Administració de sistemes operatius

CFGS.ASX.M06/0.12

CFGS - Administració de Sistemes Informàtics en Xarxa

Generalitat de Catalunya
Departament d'Educació

ioc
institut obert
de catalunya

Aquesta col·lecció ha estat dissenyada i coordinada des de l'Institut Obert de Catalunya.

Coordinació de continguts
Mª Del Mar Sánchez-Colomer Ruiz

Redacció de continguts
Joan Muñoz Pastor
Oriol Pérez Lozano
Andrés Pérez Payeras
Mª Del Mar Sánchez-Colomer Ruiz
Miquel Tarazona Belenguer

Adaptació de continguts
Raúl Velaz May

Imatge de coberta
Chris 73 / Wikimedia Commons

Primera edició: setembre 2012

Material realitzat per Eureca Media, SL
Dipòsit legal: DL B 28670-2015

Introducció

L'administració de sistemes operatius engloba una gran quantitat de tasques que tenen com a finalitat principal instal·lar, configurar i mantenir sistemes operatius, garantint la funcionalitat i la integritat dels recursos i serveis del sistema, així com la qualitat exigida.

Avui dia hi ha una gran varietat de sistemes operatius i, fins i tot en un mateix sistema, no hi sol haver una única eina per a cada tasca ni una sola manera de fer les coses. Malgrat tot, l'assoliment correcte dels conceptes associats a la realització de les tasques ens ha de permetre poder afrontar l'administració de qualsevol sistema operatiu. En aquest mòdul s'estudien algunes de les tasques típiques d'administració de sistemes operatius i s'intenta donar un enfocament pràctic als continguts per tal d'apropar-los a situacions reals, però cal remarcar que l'administració és una disciplina que s'aprèn dia a dia, amb la feina diària i el reciclatge constant.

En la unitat "Administració de processos. Administració remota. Administració de serveis d'impressió" s'explica el funcionament i la gestió dels processos del sistema, els mètodes d'accés per poder realitzar l'administració del sistema de forma remota i la gestió dels serveis d'impressió.

La unitat "Integració de sistemes operatius en xarxa lliures i propietaris" centra els seus continguts a explicar com integrar sistemes operatius de diferents famílies, el que anomenem sistemes heterogenis.

En la unitat "Administració del servei de directori" es tracten els conceptes i procediments requerits per poder administrar correctament un servei de directori i s'explica com es pot utilitzar el directori per emmagatzemar la informació dels usuaris d'una xarxa i permetre l'autenticació centralitzada dels sistemes i les diferents aplicacions utilitzades.

La unitat "Llenguatges de guions i automatització de tasques" cobreix els continguts requerits per aprendre a programar amb llenguatges de guions de *shell* per tal de poder realitzar guions que permetin automatitzar les tasques d'administració del sistema operatiu.

Aquest mòdul té un alt component pràctic, i per aprendre els conceptes que apareixen en cada unitat i aplicar amb agilitat els procediments explicats, és imprescindible implementar els exemples il·lustratius, dur a terme totes les activitats proposades i realitzar els exercicis d'autoavaluació per tal de comprovar que s'han assolit els coneixements.

Resultats d'aprenentatge

En acabar aquest mòdul l'alumne/a:

Unitat 1: Administració de processos. Administració remota. Administració de serveis d'impressió

1. Administració de processos del sistema descrivint i aplicant criteris de seguretat i eficiència.
2. Administració remota del sistema operatiu en xarxa valorant la seva importància i aplicant criteris de seguretat.
3. Administració servidors d'impressió descrivint les seves funcions i integrant-los en una xarxa.

Unitat 2: Integració de sistemes operatius en xarxa lliures i propietaris

1. Integra sistemes operatius lliures i propietaris, justificant i garantint la seva interoperabilitat.

Unitat 3: Administració del servei de directori

1. Administra el servei de directori interpretant especificacions i integrant-lo en una xarxa.

Unitat 4: Llenguatges de guions i automatització de tasques

1. Utilitza llenguatges de guions en sistemes operatius, descrivint la seva aplicació i administrant serveis del sistema operatiu.
2. Gestiona l'automatització de tasques del sistema, aplicant criteris d'eficiència i utilitzant ordres i eines gràfiques.

Continguts

Administració avançada de sistemes operatius

Unitat 1

Administració de processos. Administració remota. Administració de serveis d'impressió

1. Administració de processos del sistema
2. Serveis d'accés i administració remota
3. Administració de servidors d'impressió

Unitat 2

Integració de sistemes operatius en xarxa lliures i propietaris

1. Xarxes heterogènies
2. Configuració i utilització de xarxes heterogènies

Unitat 3

Administració del servei de directori

1. El servei de directori
2. Instal·lació, configuració i manteniment del servei de directori
3. Integració del servei de directori

Automatització de tasques i llenguatges de guions

Unitat 4

Llenguatges de guions i automatització de tasques

1. Llenguatges de guions de shell
2. Programació de shell Bash
3. Planificació i automatització de tasques

Administració de processos. Administració remota. Administració de serveis d'impressió

Joan Muñoz Pastor i Raúl Velaz May
Adaptació de continguts: Joan Muñoz

Índex

Introducció

L'administració del sistema operatiu comprèn un conjunt de tasques i responsabilitats de gran importància pel bon funcionament i productivitat de tot l'engranatge informàtic d'una empresa. Entre aquestes responsabilitats destaquen la necessitat de conèixer el funcionament dels processos, l'administració remota del sistema o la gestió dels serveis d'impressió.

Així doncs en l'apartat "Administració de processos del sistema" ens adonarem que els sistemes operatius actuals són d'una complexitat remarcable i que poden treballar amb múltiples usuaris i tasques alhora gràcies a la gestió dels processos. El processos són tasques independents i dinàmiques que van canviant constantment, així que cal conèixer el seu funcionament, la manera com neixen, canvien d'estat i moren. Hem de saber com comunicar-nos amb ells, enviar i rebre senyals o canviar la seva prioritat d'execució en funció de les necessitats del sistema i del mateix usuari. Aquest coneixement ens farà entendre les seqüències d'arrencada i aturada, així com els nivells d'execució i els serveis que hi estan associats, que ens garanteixen el funcionament del sistema.

En l'apartat "Serveis d'accés i administració remota" veurem com la generalització de les comunicacions entre ordinadors, tant en xarxa local com mitjançant Internet, ha introduït canvis dràstics en l'administració, permetent que pugui fer-se de forma remota. Tant si es basa en la línia d'ordres, en una aplicació gràfica o si usa del navegador com a interfície gràfica, l'administració remota facilita a l'administrador la tasca de configuració i gestió del sistema informàtic, tenint sempre en compte la capacitat i ample de banda de la comunicació i la necessària atenció a la seguretat i a la privacitat.

En l'apartat "Administració de servidors d'impressió" abordarem l'administració dels serveis d'impressió, una de les tasques tradicionals que sempre han tingut espai en l'administració de sistemes, que consisteix en la conversió del resultat del procés informàtic a un suport de paper definitiu i directament interpretable pel usuari. En aquest sentit, l'evolució constant de les necessitats dels usuaris i de la tecnologia ens ha portat a sistemes que garanteixen la gestió de múltiples treballs en un entorn multisusuari, la comunicació i transmissió en xarxa amb els protocols adequats, l'administració de cues d'impressió i el suport a una tecnologia d'impressió gràfica dels perifèrics cada vegada més sofisticada.

Per treballar els continguts d'aquesta unitat, és convenient llegir amb atenció la part teòrica, dur a terme les activitats pràctiques proposades i fer els exercicis d'autoavaluació del material web.

Resultats d'aprenentatge

1. Administració de processos del sistema descrivint i aplicant criteris de seguretat i eficiència.
    - Descriu els conceptes de procés del sistema, tipus, estats i cicle de vida.
    - Utilitza interrupcions i excepcions per descriure els esdeveniments interns del processador.
    - Diferencia entre procés, fil i treball.
    - Realitza tasques de creació, manipulació i acabament de processos.
    - Utilitza el sistema de fitxers com a mitjà lògic per al registre i identificació dels processos del sistema.
    - Utilitza eines gràfiques i ordres per al control i seguiment dels processos del sistema.
    - Comprova la seqüència d'arrencada del sistema, els processos implicats i la relació entre ells.
    - Pren mesures de seguretat davant l'aparició de processos no identificats.
    - Documenta els processos habituals del sistema, la seva funció i relació entre ells.
2. Administració remota del sistema operatiu en xarxa valorant la seva importància i aplicant criteris de seguretat.
    - Descriu mètodes d'accés i administració remota de sistemes.
    - Diferencia entre els serveis orientats a sessió i els no orientats a sessió.
    - Utilitza eines d'administració remota subministrades pel propi sistema operatiu.
    - Instal·la serveis d'accés i administració remota.
    - Utilitza ordres i eines gràfiques per gestionar els serveis d'accés i administració remota.
    - Crea comptes d'usuari per a l'accés remot.
    - Realitza proves d'accés i administració remota entre sistemes heterogenis.
    - Utilitza mecanismes d'encriptació de la informació transferida.
    - Documenta els processos i serveis del sistema administrats de forma remota.
3. Administració servidors d'impressió descrivint les seves funcions i integrant-los en una xarxa.
    - Descriu la funcionalitat dels sistemes i servidors d'impressió.

- Identifica els ports i els protocols utilitzats.
- Utilitza les eines per a la gestió d'impressores integrades en el sistema operatiu.
- Instal·la i configura un servidor d'impressió en entorn web.
- Crea i classifica impressores lògiques.
- Crea grups d'impressió.
- Gestiona impressores i cues de treballs mitjançant ordres i eines gràfiques.
- Comparteix impressores en xarxa entre sistemes operatius diferents.
- Documenta la configuració del servidor d'impressió i de les impressores creades.

1. Administració de processos del sistema

Tot programari executable, inclòs el mateix sistema operatiu, s'acaba constituint en una sèrie d'unitats anomenades *processos* que s'organitzen, ordenen i consumeixen recursos, per acabar sent executats pel processador. L'administració d'aquests processos i dels seus recursos associats constitueix una de les tasques bàsiques i primordials del nucli del sistema operatiu que, entre d'altres, comprendrà funcions com ara:

- La creació i destrucció de processos
- El despatx (*dispatcher*), és a dir, l'assignació de processos a execució en el processador
- Els canvis d'estat que anomenem transicions
- La suspensió i represa de processos
- La sincronització de processos
- La comunicació entre processos
- La manipulació de blocs de control de procés (PCB)

  Moltes d'aquestes activitats estan només en mans del sistema operatiu i seran transparents per a l'usuari, però cal conèixer bé els mecanismes de l'administració de processos, ja que l'usuari pot intervenir en aspectes crucials com ara la configuració de l'arrencada del sistema, l'assignació de prioritats o la finalitzant, suspensió i represa d'aquests processos.

1.1 Conceptes bàsics sobre processos

Un procés és la instància d'un programa en execució que necessita per realitzar la seva tasca recursos com ara:

- Temps de processador
- Memòria
- Arxius
- Dispositius d'entrada/sortida

Aquests recursos es poden assignar en el moment de la creació del procés o bé durant la seva execució.

Des de aquesta perspectiva, un procés es pot considerar unes línies de codi carregades a memòria i un context associat constituït per l'activitat del processador en un moment determinat, és a dir:

- El valor del comptador del programa
- El contingut dels registres del processador

i per una sèrie de dades emmagatzemades a la memòria:

- Variables globals i memòria dinàmica
- Dades temporals a la pila (*stack*) com adreces de retorn i variables locals

1.1.1 Conceptes d'aplicació, procés i servei

Es considera generalment una **aplicació** informàtica com un programa dissenyat per interaccionar amb l'usuari, per tant s'executa en primer pla (*foreground*) amb una determinada interfície d'usuari. Són exemples típics d'aplicacions els programes d'ofimàtica, els navegadors, els reproductors multimèdia, etc.

En canvi, un **servei** és un terme que correspon al concepte i funcionalitat del **dimoni** (daemon) en la terminologia de Linux. Anomenem *servei* a un programa normalment associat al sistema operatiu, encara que també es pot engegar de forma manual, i que treballa en segon pla (*background*) sense interfície d'usuari, donant suport a altres programes. Els serveis proporcionen prestacions com serveis de pàgines web, registre d'esdeveniments, serveis d'impressió, criptografia...

El concepte de **procés,** en canvi, té més a veure amb el funcionament intern del sistema operatiu i consisteix en una sèrie d'instruccions allotjades a la memòria que, mitjançant cues i un mecanisme de planificació, accedeixen a la CPU per executar-se. Així, doncs, quan s'engega qualsevol aplicació o servei genera un o més processos en el sistema.

1.1.2 Tipus de processos i classificació

Podem fer algunes classificacions dels processos en funció del seu propietari, el seu comportament en concurrència, la forma d'execució i ubicació a la memòria o els recursos del sistema que comparteixin.

Segons del propietari del procés tenim:

- **Processosde sistema**: Associats al funcionament del nucli del sistema o dels diferents serveis en funcionament (dimonis). Molts d'aquests processos estan associats a usuaris virtuals del sistema (lp, www, mail, etc.).

- **Processos de superusuari**: Associats al compte de l'administrador arrel (*root*).
- **Processos d'usuari**: Associats a l'execució d'aplicacions d'un usuari determinat.

També podem classificar els processos que s'executen concurrentment com a:

- **Independents**: No afecten ni són afectats per altres processos.
- **Cooperants**: Poden compartir dades i per tant afectar i ser afectats per altres processos.

Segons la forma d'execució i ubicació a la memòria poden ser:

- **Residents**: Els processos residents són sempre a la memòria mentre dura l'execució.
- **Intercanviables**: Poden ser portats de la memòria principal al disc dur mentre estan bloquejats. Així, la memòria alliberada pot ser utilitzada per altres processos que la necessitin.

Finalment segons els recursos que comparteixen trobem processos:

- **Pesants**: Els processos pesants són independents i tenen tots els seus propis recursos.
- **Lleugers**: També anomenats fils d'execució o *threads*, comparteixen entre si espai de memòria i recursos d'entrada i sortida.

1.1.3 Planificació de processos

Tal com hem avançat, una de les obligacions d'un sistema operatiu com a gestor de processos és la planificació de processos: l'execució de múltiples processos optimitzant l'ús del processador.

Així, doncs, les polítiques de planificació de processos sorgeixen com a necessitat d'acomplir dos dels principals objectius funcionals dels sistema operatiu: la multiprogramació i el temps compartit:

- **Multiprogramació**: Té com a objectiu maximitzar l'aprofitament de la CPU tenint sempre en execució algun procés en tot moment.
- **Temps compartit**: Atès que una CPU només pot realitzar un procés alhora es tracta d'establir un sistema de commutació de la CPU entre els processos amb tal freqüència que l'usuari pugui interactuar-hi i tingui la impressió que s'estan executant en paral·lel.

La planificació de processos es desenvolupa a tres nivells: llarg, mig i curt termini.

Planificació a llarg termini

El planificador a llarg termini és l'encarregat de crear els processos determinant quins treballs s'admeten en el sistema per al seu processament i carregant-los en la memòria disponible. Els sistemes operatius de temps compartit gairebé no tenen algorisme de planificació a llarg termini i es limiten a crear i posar en estat "preparat" qualsevol procés nou. En canvi, té més sentit la planificació a llarg termini en sistemes que admeten processament en lots, ja que és convenient compensar tasques que demanin més temps de processador amb tasques que demanin més operacions d'entrada i sortida, per equilibrar així els recursos del sistema.

Planificació a mig termini

La planificació a mig termini regula el grau de multiprogramació fixat en un principi pel planificador a llarg termini. Si el sistema no té prou recursos, en especial de memòria, per atendre tots els processos en marxa, es poden passar alguns d'ells a l'estat de suspesos i alliberar així recursos i memòria interna.

Grau de multiprogramació

Correspon al nombre de processos actius en un moment determinat, que estan carregats a la memòria principal del sistema.

Per fer això és fa servir la tècnica de l'intercanvi (*swapping*, en anglès), que s'encarrega de suspendre processos enviant-los a memòria secundària, habitualment el disc dur (*swap out*), i reactivant-les posteriorment, tornant-los a carregar de la memòria secundària a la memòria interna (*swap in*).

Planificació a curt termini

El planificador a curt termini o *dispatcher* té la tasca essencial de decidir quin procés passa a execució d'entre els que estan a la cua dels processos preparats (*ready*). Per fer aquesta tasca, els planificadors a curt termini implementen una sèrie d'algorismes que tenen com a objectiu optimitzar l'eficiència, la productivitat i el temps de resposta de la CPU.

Aquests algorismes de planificació es poden classificar en algorismes apropiatius i no apropiatius, i també hem de considerar altres tècniques combinades, com ara les cues multinivell.

Algorismes no apropiatius

El sistema operatiu no expulsa mai el procés de la CPU fins que aquest acaba l'execució o en surt voluntàriament, per exemple, quan el procés passa a bloquejat a l'inici d'una operació d'entrada/sortida. Exemples d'aquests tipus d'algorisme poden ser:

Cues FIFO i LIFO

En una cua FIFO (*first input first output*) els elements de la cua surten en el mateix ordre que han entrat, mentre que en una cua LIFO (*last input first output*), també anomenada pila o stack, és el darrer element arribat el primer que surt de la cua.

- **FCFS** (*first come first served*). És una cua FIFO que dóna preferència segons l'ordre d'arribada a la cua.
- **SJF** (*shortest job first*). Algorisme no apropiatiu on la prioritat d'accés a l'execució depèn del temps de ràfega de CPU necessari. Així, es

beneficien aquells processos més curts d'executar i es maximitza el nombre de processos executats per unitat de temps.

Algorismes apropiatius

El sistema operatiu pot decidir expulsar un procés de la CPU i executar-ne un altre. Alguns exemples d'aquest tipus d'algorisme:

- **Round Robin**: Algorisme de roda en el qual el sistema operatiu assigna un temps determinat a cada procés. Aquest temps, anomenat **quàntum**, pot ser constant o calcular-se dinàmicament. El procés abandona la CPU quan esgota el seu quàntum. L'elecció del valor del quàntum és molt important, ja que un quàntum molt petit produeix massa canvis de context.

- **SRT** (*shortest remaining time*): Algorisme depenent del temps de ràfega com l'SJF, però en aquest cas és apropiatiu. És a dir, si arriba un nou procés a la cua de preparats i el seu temps d'execució és menor que el que li resta al que s'està executant en aquell moment, es pot apropiar de la CPU i expulsar-lo.

Cues multinivell

Consisteixen en diferents cues de processos en estat de preparats. Cada cua té la seva prioritat. Per accedir a l'execució en el processador s'escull el procés de la cua més prioritària que no estigui buida. Dins de cada cua es pot aplicar una política i un algorisme de planificació diferent.

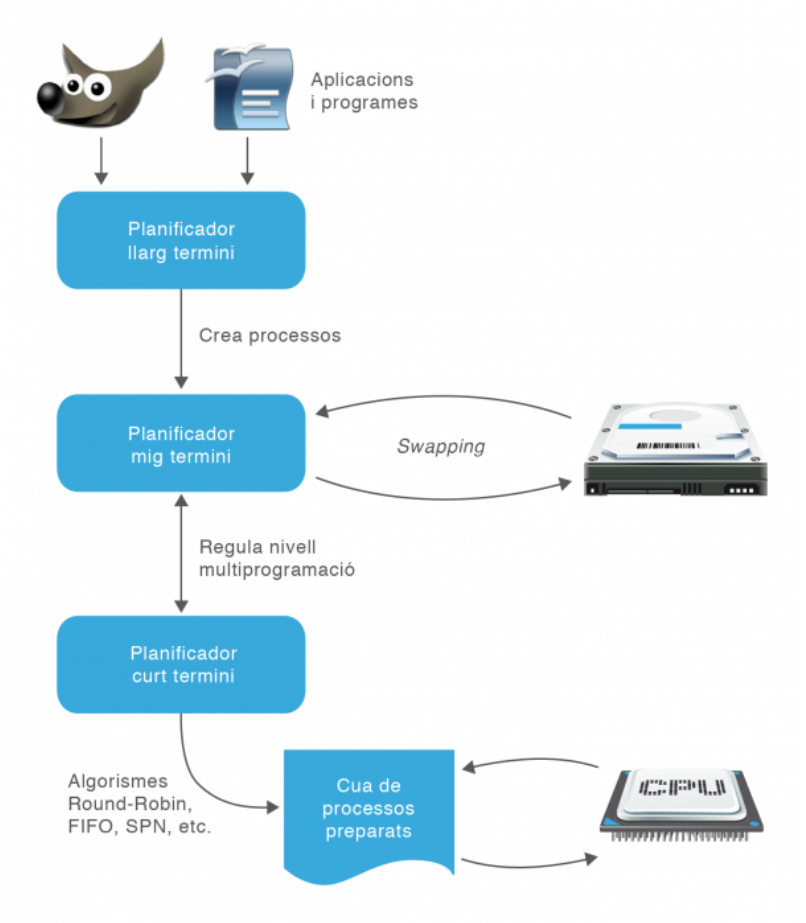
Cues multinivell realimentades

És el mateix cas de les cues multinivell, però els processos poden avançar i retrocedir per les diferents cues de prioritats. Per exemple, un procés de prioritat alta que exhaureixi el seu quàntum podria passar a una cua de preparats de menys prioritat.

El sistema operatiu Unix/Linux fa servir **cues multinivell realimentades** utilitzen l'algorisme Round Robin a cada cua. Cada segon es recalculen les prioritats dels processos en funció d'unes prioritats dinàmiques i d'unes de fixes, definides d'una banda pel sistema, segons el tipus de procés, i de l'altra pel mateix usuari mitjançant la configuració de la prioritat *nice*.

A la figura 1.1 veiem esquemàticament la intervenció dels diferents nivells de planificació en la creació de processos, la regulació del nivell de multiprogramació mitjançant l'intercanvi amb el disc dur i finalment l'assignació òptima dels processos preparats per ser executats en el processador.

FIGURA 1.1. Esquema del sistema de planificació de processos



1.1.4 Estats i transicions dels processos

Una característica fonamental d'un procés és l'estat en el qual es troba, que ve definit per la seva incorporació dins de les diferents cues que organitzen la gestió del processador. Una primera aproximació ens dóna fins a cinc estats diferents possibles per a un procés, encara que la incorporació del sistema d'intercanvi (*swapping*) ens aportarà dos possibles estats addicionals.

Estats d'un procés

Procés zombi

Podem considerar-ho com una variant de l'estat finalitzat. És un procés que ha finalitzat i ja no fa sevir la CPU, però que ha de mantenir en memòria certa informació d'utilitat per al seu procés pare. A Linux, els processos finalitzats passen directament a estat zombi fins que el procés pare se n'assabenta i els esborra definitivament.

Els cinc estats bàsics d'un procés són els següents:

- **Nou** (*new*): El procés s'està creant.
- **Preparat** (*ready*): El procés està a la cua, preparat per a l'assignació d'un processador.
- **En execució** (*running*): Les instruccions del procés estan sent executades pel processador.
- **Bloquejat** (*waiting*): El procés està parat en espera d'un esdeveniment

que el desbloquegi (finalització d'una entrada/sortida, recepció d'un senyal, etc.).

- **Finalitzat** (*terminated*): El procés està finalitzat i s'alliberen els recursos que feia servir.

Només un procés pot estar en estat d'execució en un processador i en un moment determinats, però molts processos poden estar preparats o en espera bloquejats.

Transicions d'un procés

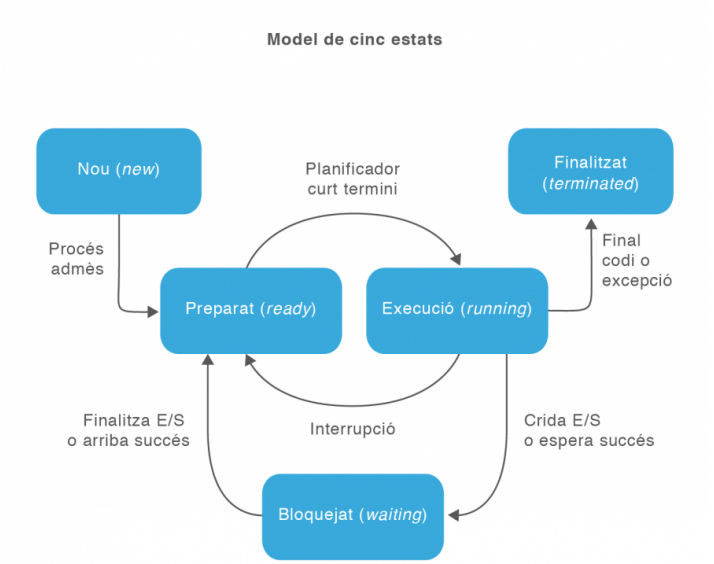
Un procés és un element essencialment dinàmic que va canviant entre els diferents estats mitjançant transicions. Partint d'un model de cinc estats com els que hem definit, les possibles transicions entre estats són les següents:

- **Nou** → **preparat**: el sistema està preparat per admetre un nou procés perquè disposa de recursos suficients.
- **Preparat** → **execució**: el planificador a curt termini, tot seguint diferents algorismes de planificació (Round Robin, SFJ, SPN, FCFS, etc.) decideix quin dels processos en cua de preparats passa a execució.
- **Execució** → **preparat**: si l'algorisme de planificació és apropiatiu, el procés en execució pot tornar a la cua de preparats si esgota el seu temps prefixat d'execució (quàntum) o si rep una interrupció per l'arribada d'un procés amb més prioritat.
- **Execució** → **bloquejat**: si un procés en execució necessita una operació d'entrada/sortida o algun altre esdeveniment per continuar, passa a l'estat de bloquejat i allibera la CPU.
- **Bloquejat** → **preparat**: si en un procés bloquejat arriba el succés o finalitza l'operació d'entrada/sortida que esperava, llavors retorna a la cua de processos preparats.
- **Execució** → **finalitzat**: si un procés finalitza l'execució de les seves línies de codi o bé rep una excepció per algun tipus d'error greu, passa a procés finalitzat i allibera els recursos emprats.

Model de cinc estats

Aquest model descrit de cinc estats i les seves transicions descrit anteriorment es pot veure gràficament en el diagrama d'estats i transicions de la figura 1.2.

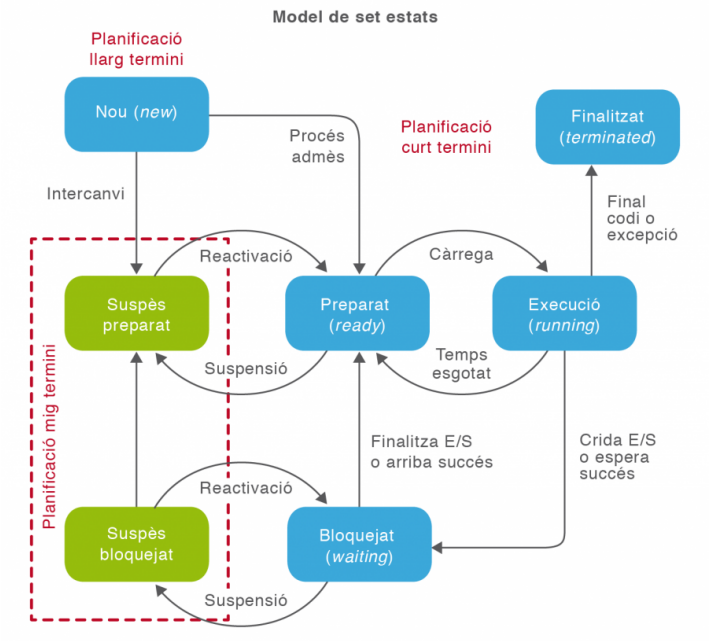
FIGURA 1.2. Model de cinc estats dels processos amb les seves transicions



Model de set estats

Si al model de cinc estats li afegim la planificació a mig termini que gestiona l'intercanvi de processos a la memòria secundària, podem considerar dos nous estats:

FIGURA 1.3. Model de set estats dels processos amb la inclusió dels estats de suspensió



- **Suspès preparat**: Aquells processos que ja estan preparats per ser executats però que, per falta de recursos del sistema, el planificador a mig termini ha decidit passar temporalment a la memòria secundària (disc dur).
- **Suspès bloquejat**: Processos que estan en espera d'un esdeveniment o de la

finalització d'una operació d'entrada/sortida i que són passats a la memòria secundària per alliberar recursos.

Així, doncs, ens queda un diagrama d'estats i transicions com el de la figura 1.3.

1.2 Control i estructura dels processos

Controlar i fer operatiu tot aquest sistema de gestió dels processos requereix la definició d'una sèrie d'estructures de dades que continguin tota la informació associada a cada procés, així com les llistes i taules per controlar les diferents cues implicades en la planificació. Dues d'aquestes estructures bàsiques d'informació són el **bloc de control de procés** (PCB) i el **bloc de control de sistema**(SCB).

D'altra banda, cal comprendre el mecanisme que permet al processador canviar el procés que està executant per un altre. Aquest mecanisme s'anomena **canvi de context**. La problemàtica de la penalització en temps de CPU que suposa el canvi de context ens portarà a l'aparició de tècniques més òptimes, com els sistemes multifil basats en fils d'execució o *threads*.

1.2.1 Bloc de control de procés

Cada procés es representa al sistema operatiu com una estructura de dades anomenada **bloc de control de processos**(en anglès *process control block* o PCB) que descriu el procés i conté una sèrie de camps d'informació:

- **Identificador** del procés.
- **Estat**: estat actual del procés (preparat, bloquejat, en execució, etc.)
- **Comptador del programa**: indica l'adreça de la següent instrucció d'aquest procés que s'executarà.
- **Registres de la CPU**: acumuladors, punters de pila (*stack*) i registres generals.
- **Informació de planificació**: com la prioritat del procés i el valor del quàntum.
- **Informació de la gestió de la memòria**: definició de la finestra de memòria amb el registre base i límit, taula de pàgines o segments.
- **Informació de comptabilitat**: temps de CPU, temps consumit, etc.
- **Informació de l'estat de les entrades/sortides**: dispositius d'entrada/sortida assignats al procés, llista d'arxius oberts, etc.

- **Punter a la memòria**: apunta al node següent de la llista enllaçada de PCB.

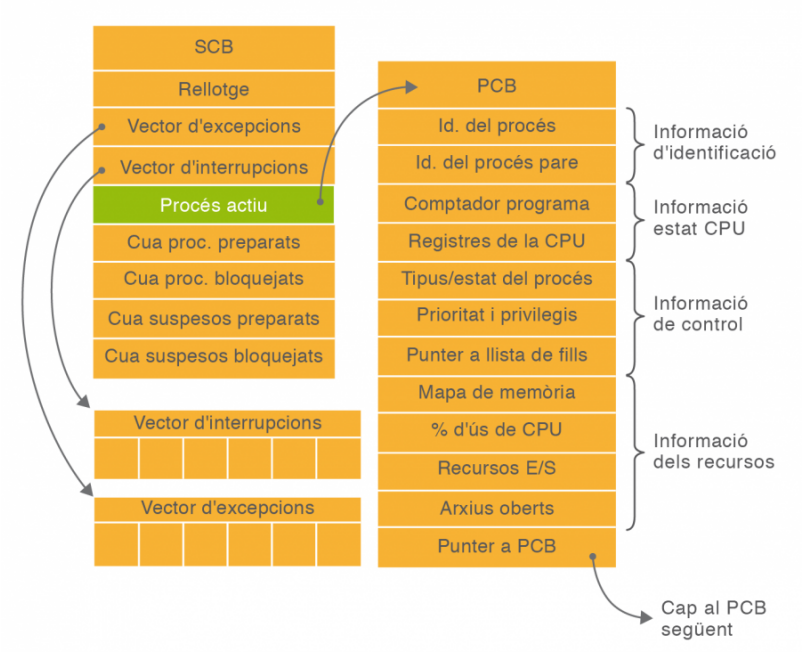
Per gestionar els processos el sistema operatiu ha de llegir, tractar i manipular aquestes estructures de dades.

1.2.2 Bloc de control del sistema

Els sistemes operatius disposen també d'una estructura anomenada **bloc de control del sistema** (de l'anglès *system control block* o SCB) per controlar i reunir informació de totes les estructures de dades dels processos, les interrupcions i les excepcions. El bloc de control de sistema conté habitualment la informació següent:

- Un punter cap al descriptor (PCB) del procés que està fent ús del processador (procés actiu)
- Un punter a una cua de descriptors dels processos preparats (*ready*)
- Un punter a una cua de descriptors dels processos bloquejats (*waiting*)
- Un punter a una cua de descriptors dels processos suspesos preparats
- Un punter a una cua de descriptors dels processos suspesos bloquejats
- Punters als vectors d'interrupcions i d'excepcions que són referències a les rutines necessàries per atendre aquests esdeveniments

FIGURA 1.4. Estructures de control de processos: bloc de control de procés i de sistema



Una **interrupció**, sigui de programari (crides al sistema) o de maquinari (rellotge, dispositius d'entrada/sortida, reinicialització, etc.), és la presència d'un esdeveniment que obliga el sistema operatiu a prendre el control del processador per analitzar i tractar la situació mitjançant rutines específiques.

Una **excepció** és un tipus d'interrupció generada al sistema per un error (divisió per zero, desbordament de memòria...).

Vector d'interrupcions/excepcions

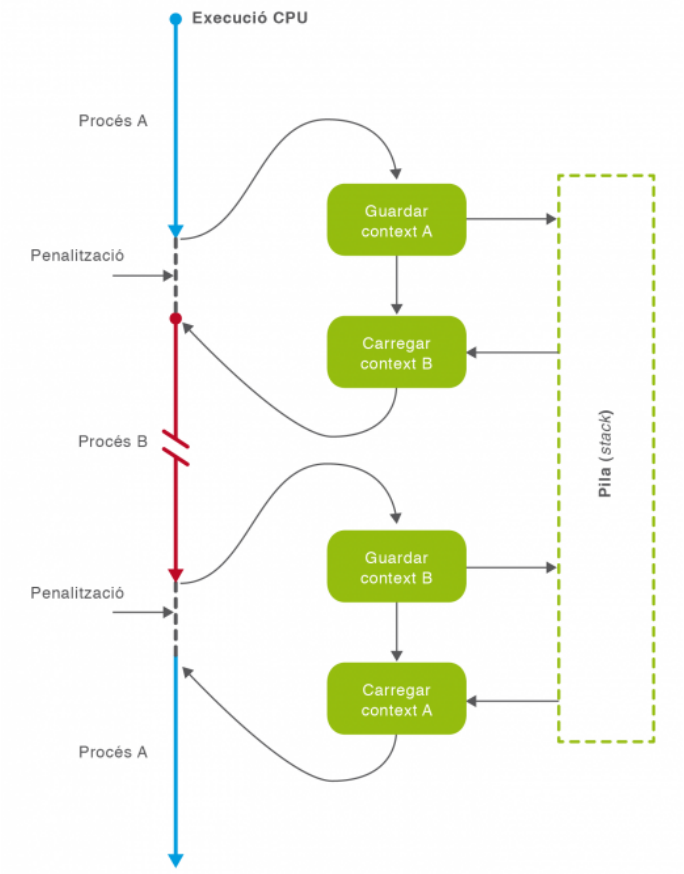
És una taula de les adreces de memòria on estan situades les rutines a executar pel tractament de la interrupció/excepció generada.

A la figura 1.4 es mostra un esquema d'aquestes estructures de dades.

1.2.3 Canvi de context

Quan un procés en execució ha d'abandonar la CPU per algun esdeveniment (interrupció, exhauriment del quàntum, operació d'entrada/sortida, etc.) i s'ha de començar o reprendre un altre procés és necessari fer un **canvi de context**. Per fer això cal guardar el context del procés que surt, és a dir, el seu estat, registres, comptador de programa, etc., que està representat pel seu PCB (bloc de control de procés) i recuperar el context del procés que entra en execució. Vegeu la figura 1.5.

Figura 1.5. Accions a efectuar en un canvi de context



Els canvis de context permeten la commutació de la CPU entre diferents processos, però també suposen una penalització pel temps necessari per efectuar-los. Per intentar reduir aquesta penalització (*overhead*) es fan servir noves estructures com els fils d'execució (*threads*).

1.2.4 Fils d'execució

La definició de procés inclou dos conceptes separats i potencialment independents: un de relatiu a la propietat dels recursos, i un altre que fa referència a la assignació de CPU per a l'execució del codi.

- **Recursos del procés**: A un procés se l'assigna sempre un espai de memòria i a vegades també altres recursos com dispositius d'entrada/sortida i arxius.
- **Assignació de CPU**: Un procés és un flux d'execució, també anomenat traça, quan aquest és assignat a un processador per a l'execució del codi.

  A la majoria de sistemes operatius aquests dos requeriments són l'essència del procés, però poden ser tractats de manera independent. Aquesta distinció ha portat, en els sistemes operatius actuals, a desenvolupar el concepte de **fil d'execució**, també anomenat procés lleuger o *thread*, en anglès. Tenint en compte aquesta divisió de característiques, podem definir fil d'execució com la unitat d'assignació de CPU.

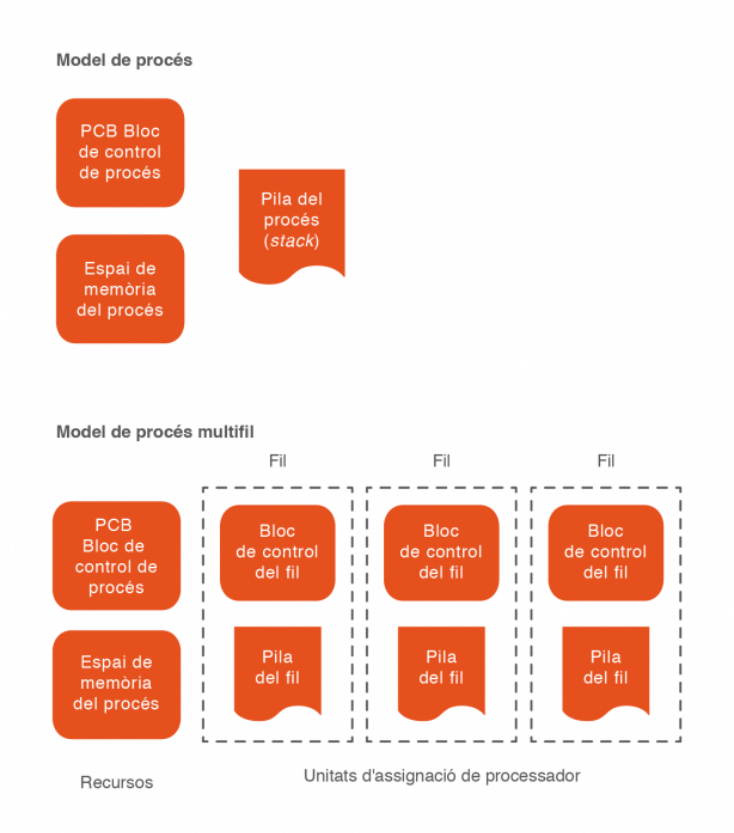
Així, doncs, en un procés hi pot haver un o més fils d'execució que disposen dels seus propis:

- Context, és a dir, el comptador de programa, estat del fil (preparat, bloquejat, etc.) i els registres de CPU, tot definit en una estructura de dades anomenada *bloc de control del fil*.
- Pila d'execució per les variables locals.

En canvi, tots els fils d'execució d'un mateix procés comparteixen certs recursos com ara l'espai de memòria, les variables globals i els arxius i dispositius d'entrada i sortida.

Els beneficis clau de la implementació de sistemes multifil rau en la millora del rendiment. És triga menys temps a crear i finalitzar fils d'execució, així com a fer canvis de context entre fils d'un mateix procés. La comunicació entre fils també és més ràpida, ja que comparteixen memòria i recursos als que poden accedir sense invocar el nucli del sistema operatiu. Finalment, en sotmetre un mateix procés a diferents fluxos d'execució es manté una única còpia del codi en memòria i, en sistemes multiprocessador, fins i tot és possible l'execució simultània de diferents fils del mateix procés en diferents processadors. Vegeu la figura 1.6.

FIGURA 1.6. Model tradicional de procés i model de procés multifil



En un sistema multifil cada fil d'execució és independent i fins i tot por executar-se simultàniament en processadors diferents, però compartint els mateixos recursos de memòria i dispositius d'entrada/sortida.

1.3 Prioritats dels processos

Un dels factors fonamentals que intervenen en la planificació a curt termini dels processos és la prioritat, que determina la cua on un procés haurà d'esperar el seu torn. A cada procés se li assigna una prioritat en funció de si és més crític o més urgent i dels recursos que necessita, la qual cosa determinarà finalment la freqüència d'accés al processador.

1.3.1 Tipus de prioritats

Podem establir una classificació de les prioritats en funció de la seva possibilitat de variació al llarg de l'execució del procés. Així, tindrem:

- **Prioritat estàtica:** Aquelles que no poden ser modificades mentre s'executa el procés.

- **Prioritat dinàmica:** Quan un procés pot modificar la seva prioritat en funció de determinats esdeveniments.

  Cada procés té assignada una prioritat que pot ser estàtica, dinàmica o una combinació de les dues. El problema de la prioritat estàtica és la inanició, és a dir, que un procés mai s'executi per no tenir mai la prioritat suficient. Per solucionar això es fa servir la prioritat dinàmica per envelliment, en la qual els processos van augmentant la seva prioritat en funció del temps que fa que estan en espera.

D'altra banda, la prioritat d'un procés està composta per dos factors:

- **Prioritat assignada pel sistema operatiu** en funció del tipus de procés, algorismes de planificació, polítiques per preveure la inanició, etc.
- **Prioritat assignada pel propi usuari**. Naturalment només afecta als processos d'usuari i permet tenir en compte els seus interessos i necessitats. Aquesta llibertat comporta un cert risc, com, per exemple, deixar bloquejada la resta del sistema si un programa amb prioritat alta assignada per l'usuari entra en un bucle infinit.

Als sistemes Unix/Linux, la prioritat d'usuari s'anomena prioritat *nice* i es pot assignar amb les ordres *nice* i *renice*.

1.3.2 Ordres per gestionar la prioritat

Més és menys

Nice vol dir 'amable' en anglès. Així s'entén millor per què quan més gran és el valor de nice menys prioritat té el procés i per tan és més "amable" amb la resta de processos que competeixen pels recursos.

Tots els processos tenen una prioritat assignada per l'usuari, anomenada **prioritat*nice***. Els processos amb més prioritat tenen valors negatius de *nice* i fan servir més recursos que els altres. La prioritat màxima assignada a *nice* és –20 i la prioritat mínima és +19. Si la prioritat no està definida, cada procés s'executarà amb una prioritat *nice* de 0.

Ordre nice

L'ordre *nice* permet arrencar un procés assignant-li d'entrada una determinada prioritat i té com a argument el nom de l'ordre que genera el procés sobre el qual volem actuar. Si no especifiquen el nivell de prioritat queda definida per defecte en 10.

```
1 $ nice -n 19 987
```

Aquesta ordre deixa amb prioritat mínima el procés de PID=987.

Ordre renice

Per canviar la prioritat *nice* d'un procés ja engegat es fa servir l'ordre ***renice***. Per exemple:.

```
$ renice –n –5 987 –u root
```

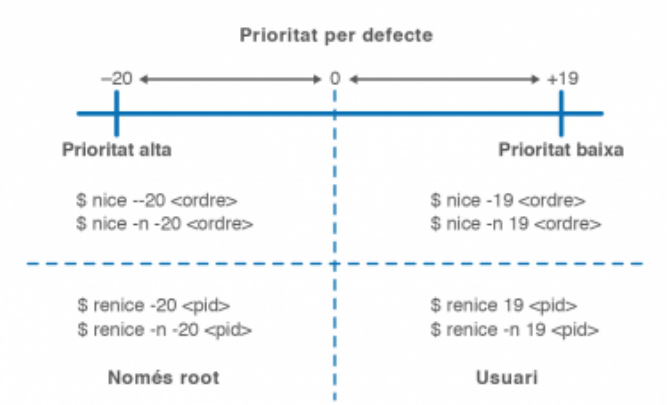
Aquesta ordre donarà prioritat –5 al procés de PID=987 i a tots els processos de l'usuari *root*.

> Qualsevol usuari pot reduir la prioritat del seus processos, però només el superusuari pot augmentar la prioritat d'un procés aplicant un valor de *nice* negatiu.

També es pot canviar la prioritat *nice* amb l'ordre interactiva ***top***.

Podem veure un esquema de la prioritat *nice* i la sintaxi de les ordres *nice* i *renice* a la figura 1.7:

Figura 1.7. Canvis de prioritat amb nice i renice



Herència de la prioritat

Els processos fill hereten la prioritat del pare, però si es canvia la prioritat del procés pare, els processos fills ja nascuts no canvien de prioritat.

1.4 Comunicació amb els processos

Hi ha diferents mecanismes que permeten la comunicació entre els processos, alguns dels quals veurem en aquest apartat. D'una banda, tot procés té associat uns arxius (stdin, stdout, stderr) que reben la comunicació d'entrada i sortida de dades. Aquests fluxos de dades dels processos poden ser intercomunicats mitjançant **canonades**.

D'altra banda, tot procés que finalitza torna informació a una **variable de retorn**. Aquesta variable pot ser llegida per un altre procés i condicionar la seva funcionalitat.

Finalment, un procés pot rebre informació i ordres mentre s'està executant mitjançant l'enviament de determinats senyals.

1.4.1 Canonades

Entre els recursos associats a un procés hi ha la interfície estàndard de comunicació amb l'exterior. Aquesta interfície consta de tres fitxers coneguts com a arxius d'entrada, de sortida i d'errors estàndard. A Unix/Linux, quan un procés obre un arxiu, el nucli del sistema li lliura un número sencer que el procés farà servir per realitzar operacions d'entrada i sortida. Els descriptors associats als arxius estàndard estan indicats a la taula 1.1.

TAULA 1.1. Arxius estàndard d'entrada i sortida

Descriptor	Denominació	Descripció
0	**stdin**	Entrada estàndard, associada per defecte al teclat del terminal.
1	**stdout**	Sortida estàndard, associada per defecte a la pantalla del terminal.
2	**stderr**	Sortida d'errors, associada per defecte a la pantalla del terminal. Mostra els missatges d'error.

1.4.2 Valors de retorn

Tot procés que finalitza torna un número comprès entre 0 o 255 que representa la causa per la qual va finalitzar. A la taula 1.2 podem veure alguns valors de retorn.

La variable d'entorn ***?*** conté el valor de retorn de la darrera ordre executada i es pot visualitzar amb *echo$?*.

TAULA 1.2. Valors de retorn d'un procés

retorn	Descripció
0	Finalització correcta del procés. Representa un valor lògic vertader si es fa servir en una comparació o iteració. Qualsevol altre valor diferent de 0 es considera fals.
1	Indica error. Finalització anormal del procés
129 ... 160	Indica que el procés ha finalitzat com a conseqüència d'un senyal. Restant 128 al valor de retorn s'obté el codi d'aquest senyal.

1.4.3 Senyals

El senyals són interrupcions que rep el procés mentre està en execució per indicar que s'ha produït algun esdeveniment significatiu davant del qual ha de respondre.

El senyals poden ser enviats pel nucli del sistema, per altres processos o pel mateix usuari. Alguns d'aquests senyals poden ser capturats (*trap*), és a dir, poden ser ignorats o rebre un tractament específic pel procés si així està programat. En canvi, els senyals `SIGKILL` i `SIGSTOP` s'han d'atendre obligatòriament i no es poden ignorar.

Hi ha un bon grapat de senyals que podem visualitzar amb l'ordre *kill* (vegeu la figura 1.8).

```
$ kill -l
```

Figura 1.8. Llistat dels senyals amb kill -l

```
root@ioc-Server:~# kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

Els senyals més importants s'han resumit a la taula 1.3.

Taula 1.3. Senyals més emprats amb la seva descripció

Núm.	Senyal	*Trap*	Descripció
1	**SIGHUP**	SÍ	Finalització per tancament terminal o del procés pare
2	**SIGINT**	SÍ	Finalització per teclat (*Ctrl+C*)
9	**SIGKILL**	NO	Finalització forçada
15	**SIGTERM**	SÍ	Finalització ordenada per defecte
18	**SIGCONT**	SÍ	Continua un procés aturat
19	**SIGSTOP**	NO	Atura el procés
20	**SIGSTP**	SÍ	Atura el procés per teclat (*Ctrl+Z*)

Hi ha un grup de senyals de finalització que tenen com a objectiu l'acabament del procés i l'alliberament del recursos que feia servir (senyals 1, 2, 9 i 15). D'altres tenen com a objectiu aturar el procés però sense treure'l de memòria per poder engegar-lo de nou en el mateix punt en què es va deixar (senyals 19 i 20).

Per obtenir informació completa dels senyals:

```
$ man -s7 signal
```

1.4.4 Ordres relacionades amb senyals

A continuació es detallen una sèrie de combinacions de tecles i ordres de consola relacionades amb l'enviament de senyals als processos.

Senyals de teclat

Hi ha una sèrie de combinacions de tecles que envien senyals directament al procés que s'està executant en primer pla:

- ***Ctrl+Z***: Envia el senyal 20 (**SIGTSTP**). Aquest senyal atura el procés, però el deixa a la memòria i es pot reprendre en el punt en el qual va quedar mitjançant un senyal 19 (SIGCONT).
- ***Ctrl+C***: Envia el senyal 2 (**SIGINT**). La majoria d'aplicacions està programades per finalitzar quan reben aquest senyal, però cal tenir en compte que es pot programar una aplicació perquè capturi aquest senyal i l'ignori.
- ***Ctrl+***: Envia el senyal 3 (**SIGQUIT**). També depèn de com estigui configurada l'aplicació, però en principi ha de produir una finalització del procés amb un bolcat a un arxiu del seu estat en el moment de la finalització.

Ordre kill

Podem fer servir l'ordre interna *kill* per enviar senyals a qualsevol procés que haguem creat tant en primer com en segon pla. A més, si som administradors podem enviar senyals a processos d'altres usuaris.

L'ordre *kill* necessita identificar el procés mitjançant el seu PID (identificador de procés), tot i que si es tracta d'un procés en segon pla del nostre terminal, també pot fer servir el seu número de treball (*job*) precedit del símbol %.

> Per defecte, tant l'ordre *kill* com *killall* envien el senyal 15 SIGTERM de finalització ordenada del programa.

Veguem-ne algun exemple:

```
# kill 2343
```

En aquest cas s'envia el senyal de finalització ordenada SIGTERM al procés de PID=2343. En alguns casos, els processos no responen al senyal de finalització per defecte i cal enviar el senyal de finalització forçada, que no és pot capturar i s'ha d'obeir. Qualsevol de les sintaxis següents és vàlida:

```
# kill -9 2343
# kill -SIGKILL 2343
```

Ordre killall

Amb l'ordre *kill* podíem enviar senyals a un procés determinat definit pel seu PID. Si volem enviar senyals a un procés a partir del seu nom hem de fer servir ***killall***.

```
# killall –9 mozilla
```

Aquesta instrucció tancarà tots els navegadors Mozilla oberts per l'usuari que executa l'ordre.

Ordre nohup

Quan tanquem una sessió, tancant la finestra del terminal si estem en l'entorn gràfic o mitjançant l'ordre *exit* o *Ctrl+D* si estem en l'entorn de línia d'ordres, el procés que gestiona el terminal (*getty*, *mingetty*) intenta tancar tots els processos associats a aquell terminal. Per fer aquest tancament envia el senyal SIGHUP a tots els processos en marxa i també als aturats en segon pla, ja que prèviament els ha activat amb un senyal de SIGCONT.

Si volem que algun dels nostres processos es mantingui actiu després de tancar la sessió, l'hem de protegir d'aquest senyal SIGHUP amb l'ordre *nohup*. La sintaxi és simple: cridem l'ordre *nohup* i li donem com a paràmetre l'ordre a protegir:

```
# nohup ./prova.sh
nohup: es descarta l'entrada i s'afegeix l'eixida a «nohup.»out
```

En aquest exemple, l'execució de l'script ./prova.sh està protegida contra el senyal SIGHUP, la seva sortida estàndard ja no és la consola (que podria estar tancada), sinó que la sortida és redreçada i afegida a un arxiu específic anomenat ***nohup.out***.

Ordre disown

L'ordre interna ***disown*** també ens permet protegir processos del senyal SIGHUP. Les diferències amb *nohup* són les següents:

1. L'ordre *disown* s'aplica a treballs en segon pla que ja existeixen.
2. A diferència de *nohup*, no canvia les sortides estàndard i d'error.

Vegem-ne alguns exemples:

```
# disown –h %3
```

Protegeix contra SIGHUP el treball 3 de segon pla.

```
# disown –a
```

Protegeix contra SIGHUP tots els treballs en segon pla.

```
# disown –r
```

Protegeix contra SIGHUP tots els treballs en segon pla que estan en marxa (*running*).

Ordre trap

És una ordre interna que ens permet capturar un senyal i especificar el que volem fer quan ens arribi. La seva sintaxi és *trap* [ordres][senyals]:

```
trap "" SIGTERM
```

Aquesta ordre dins d'un script capturarà el senyal de finalització SIGTERM i l'ignorarà, ja que com a ordres a seguir li hem posat una cadena buida.

1.5 Seqüència d'arrencada del sistema

En el moment que engeguem l'interruptor de l'ordinador, comença una llarga seqüència d'accions i execucions de processos que finalment ens porta fins que a la pantalla ens apareix l'entorn gràfic del sistema operatiu o una consola de text ens convida amb el *prompt* (indicador d'ordres) a introduir una ordre.

Fem ara un repàs resumit de tota aquesta seqüència i, al llarg dels diferents epígrafs, posarem especial atenció en els darrers esdeveniments que culminen en la creació de l'arbre de jerarquia de processos i en la càrrega de l'intèrpret d'ordres o *shell*.

1. Després de reiniciar (*reset*) o d'engegar l'interruptor de l'equip, els components electrònics reben alimentació. El microprocessador comença a executar instruccions des de una posició del mapa de memòria determinada, anomenada v*ector de reset* i comença a executar un programa especial anomenat BIOS (*basic input/output system*) que està allotjat a la memòria fixa del sistema de tipus ROM (*memòria només de lectura*).

GRUB
Acrònim de grand unified bootloader. Carregador d'arrencada múltiple del projecte GNU que es fa servir per engegar un dels sistemes operatius instal·lats en el mateix ordinador.

2. Aquest programa fix que incorpora cada placa base constitueix l'anomenat *firmware* i conté rutines de comprovació de memòria, detecció de dispositius de maquinari, rutines POST (*power-on self test*) per a la verificació del components. També ofereix a l'usuari la interacció opcional amb el sistema de memòria CMOS, que guarda la configuració de diferents característiques del sistema com ara el rellotge de temps real, la memòria secundària, la seqüència d'arrencada, etc.

3. Un cop executades les instruccions de la BIOS, la darrera acció que realitza és la cerca del sistema operatiu a la memòria secundària, habitualment el disc dur, per carregar-lo a la memòria. Per fer això s'adreça al primer sector del disc dur anomenat MBR (*master boot record*).

4. Tradicionalment, l'MBR conté un gestor d'arrencada simple i la taula de particions del disc dur, que indica quina és la partició activa que conté el sistema operatiu a carregar. Tanmateix, gairebé totes les distribucions Linux fan servir programes carregadors d'arrencada múltiple (*bootloader*) més complexos, com ara GRUB.

5. Així, doncs, en un sistema Debian, s'inicia a l'MBR la **fase 1**de**GRUB** anomenada *primer carregador de l'arrencada* (*initial program loader* o ILP).

6. Com que gairebé no hi ha espai a l'MBR per a un programa complex, cal l'execució d'una fase anomenada **fase 1.5**, que conté codi addicional i està situada als primers sectors després de l'MBR, sempre a la primera pista del disc dur per motius de compatibilitat.

7. La resta del codi de GRUB s'executa en la **fase 2** i normalment s'instal·la al sector d'arrencada de la partició on hi ha instal·lat el sistema Linux. Aquest codi interpreta la configuració de l'arxiu /boot/grub/grub.cfg, dóna suport a diferents sistemes d'arxius i permet el pas de paràmetres al nucli del sistema, a més d'interaccionar amb l'usuari mostrant el menú de selecció a pantalla.

8. Una vegada escollida la partició d'arrencada a partir del menú de GRUB es comença la càrrega del nucli del sistema operatiu pròpiament dit. En el cas de Linux, una vegada carregat el nucli del sistema operatiu comença la creació de processos que s'executen en un ordre determinat i culmina en la creació de tota una estructura jeràrquica de processos i serveis.

1.5.1 Jerarquia de processos (PID, PPID)

A Unix/Linux tots els processos s'identifiquen amb un número sencer de 16 bits que s'assigna seqüencialment a cada nou procés que es crea. Aquest número és únic per a cada procés i s'anomena *identificador de procés* o PID (de l'anglès *proces identifier*). A més, tot procés, a excepció del procés arrel *init* amb PID=1, ha estat creat per un procés pare. Així, doncs, un procés pare pot tenir molts processos fills, però qualsevol procés només té un procés pare, identificat pel seu PPID (*parent procés identifier*). Això crea una estructura jeràrquica de processos en forma d'arbre amb el procés *init* com a arrel.

Procés pare i procés fill poden o no compartir recursos i espai de memòria, i estar o no sincronitzats, és a dir, es poden executar concurrentment o bé el procés pare pot restar en espera fins a la finalització del procés fill.

1.5.2 Creació i finalització de processos. Crides al sistema

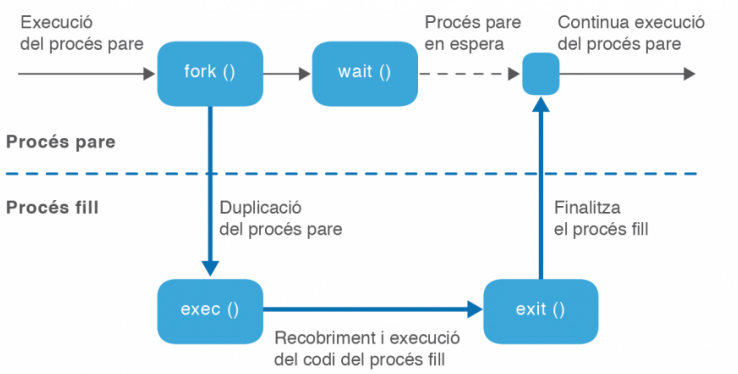
A Unix/Linux els processos en creen mitjançant crides al sistema que fan primer una duplicació del procés pare i després un recobriment del codi carregant i

executant les noves instruccions. Tot això es fa mitjançant les crides a les funcions de sistema següents:

- **Funció*fork()***: aquesta funció crea un procés fill que és una còpia pràcticament exacta del procés pare (clonació) i hereta el context del pare. Tots dos processos continuen executant-se després del punt en el qual s'ha fet la crida *fork()*, amb l'única diferència del seu PID.
- **Funció*exec()***: després de la clonació, el procés pare o bé el mateix fill han de cridar la funció *exec*, que carrega al segment de dades (recobriment) el nou codi a executar.
- **Funció*exit()***: s'invoca després de la darrera instrucció del codi per efectuar la finalització del procés i alliberar els recursos que feia servir.
- **Funció*wait()***: si el procés pare vol esperar que el procés fill finalitzi, haurà de cridar la funció *wait()*, que aturarà l'execució i el passarà a la cua de bloquejats (*waiting*) fins que el procés fill finalizi.

A la figura 1.9 es resumeixen les crides al sistema anteriors.

FIGURA 1.9. Esquema de les crides al sistema implicades en la creació d'un procés fill



A més de la finalització voluntària del procés mitjançant la crida a la funció exit(), un procés també pot finalitzar involuntàriament per:

- **Excepció**: error fatal motivat per diverses causes, instrucció privilegiada, excepció de coma flotant, violació de segment...
- **Funció *kill()***: un procés por ser finalitzat per un altre mitjançant la crida al sistema *kill()* i l'enviament del senyal de finalització corresponent.

1.5.3 Treballs en segon pla d'execució

Quan un procés pare crea un procés fill hi ha dues possibilitats en termes d'execució:

1. El procés pare espera la finalització del procés fill.
2. Pare i fill s'executen concurrentment.

D'aquesta manera en l'entorn del procés de *shell*, quan fem una crida a una ordre de sistema aquesta pot generar un procés o processos fills del *shell* que es poden executar en:

- **Primer pla (*foreground*)**: quan el *shell* queda a l'espera de la finalització de l'ordre executada i per tant el terminal no accepta noves ordres fins a la finalització del procés fill.
- **Segon pla (*background*)**: si els procés o processos fills s'executen concurrentment al procés *shell* pare, que continua acceptant noves ordres.

Directiva &

Per poder enviar l'execució d'una ordre a segon pla només cal afegir el símbol *&* al final de la línia d'ordres. Per exemple:

```
# find / -name "*.txt" &
#
```

En aquest exemple, mentre el sistema cerca en tot l'arbre de directoris els arxius amb extensió *txt*, tornem de seguida a tenir l'indicador de sistema (*prompt*), que ens acceptarà noves ordres.

El problema és que els missatges de l'ordre *find* en segon pla en apareixen a la consola i dificulten el treball, però sempre podem redreçar la sortida principal i la d'errors cap a un arxiu:

```
# find / -name "*.txt" >Llistat 2>Errors &
[1] 1544
#
```

Se'ns mostra el número de treball [1] i el PID associat al procés, 1544. Després apareix de nou l'indicador que ens convida a introduir noves ordres, com per exemple:

```
# yes
```

L'ordre *yes* genera contínuament el caràcter "y" i bloqueja el terminal. Podem fer servir la combinació de tecles *Ctrl+Z*, que enviarà aquest procés a segon pla i el deixarà aturat.

```
# ^Z
[2]+ Aturat yes
```

Ara tornem a fer la mateixa ordre, però amb més cura, enviant l'ordre directament a segon pla i la seva sortida al dispositiu *null* perquè no ens faci nosa:

```
# yes >/dev/null &
[3] 1711
```

Ja tenim un tercer treball en segon pla amb PID=1711.

Ordre jobs

L'ordre *jobs* ens permet visualitzar els treballs que tenim en segon pla i veure l'estat en el qual es troben. Si hem efectuat les anteriors ordres veurem alguna cosa semblant a:

```
# jobs
[2]+ Aturat yes
[3]– S'est? executant yes > /dev/null &
```

Com podeu veure, la primera ordre *find* ja no apareix, doncs ha finalitzat. Tenim el segon treball *yes* que hem aturat a segon pla amb la combinació *Ctrl+Z* i un tercer treball en marxa executant-se en segon pla, però amb la seva sortida redreçada al dispositiu *null*.

Ordres fg i bg

Aquestes ordres permeten reprendre una tasca aturada a segon pla en el punt on es va aturar. Aquesta tasca es pot reprendre en primer pla amb *fg* o en segon pla amb *bg*. Per indicar quina de les tasques aturades en segon pla volem reprendre cal indicar el nombre de treball precedit del símbol %.

```
# fg %2
```

Amb aquesta ordre reprendrem a primer pla el treball [2] que estava aturat. Començarà a sortir lletres "y" a la pantalla. Podem tornar a aturar el procés i enviar-lo a segon pla amb la combinació *Ctrl+Z* o parar el procés definitivament amb *Ctrl+C* i podrem tornar a cridar *jobs*:

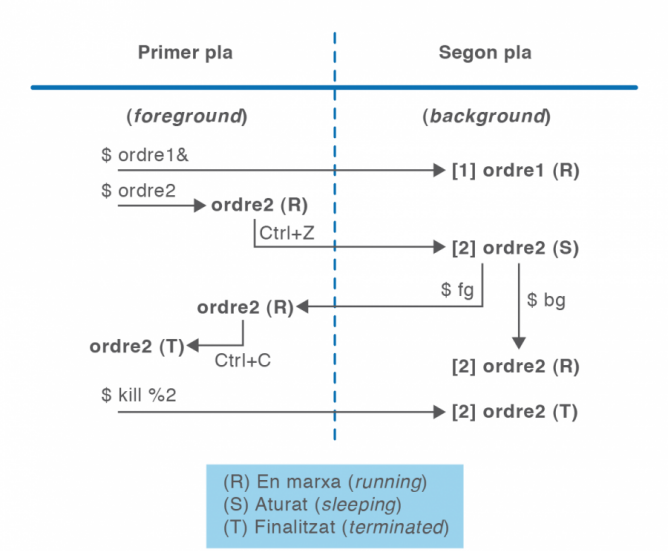
```
^C
# jobs
[3]+ S'est? executant yes > /dev/null
```

Ara només ens queda un treball en execució en segon pla. Per finalitzar definitivament un procés que s'està executant al *background* podem fer servir l'ordre *kill* tot indicant el número de PID o bé el número de treball precedit del símbol %.

```
kill %3
```

A la figura 1.11 podeu veure un resum d'aquestes ordres i de la seva funcionalitat.

FIGURA 1.10. Ordres principals per gestionar el processos a primer i segon pla



1.5.4 Dimonis

Un dimoni (de l'anglès *daemon*) és un procés no interactiu en segon pla que generalment tenim carregat a la memòria en espera d'algun senyal provinent d'un dispositiu o del mateix nucli del sistema per a despertar-se i realitzar les accions i funcions necessàries i oferir un determinat servei.

Daemon és l'acrònim de *disk and execution monitor*.

Els dimonis no disposen d'interfície amb l'usuari, per tant, no utilitzen les entrades i sortides estàndard per comunicar errors o enregistrar el seu funcionament, sinó arxius de registre (*log*), situats habitualment al directori /var/log.

Encara que un dimoni sigui un procés com qualsevol altre, que s'executa en segon pla (*background*), la manera de gestionar-lo i invocar-lo és diferent a la resta d'ordres i programes del sistema. Generalment, els dimonis tenen un guió de *shell* (*shell script*) situat al directori /etc/init.d/ que permet iniciar-los, parar-los o veure el seu estat d'execució segons la sintaxi:

```
1 # /etc/init.d/NomDimoni Accio
```

Els paràmetres d'acció bàsics que ha d'acceptar el guió del dimoni són:

- ***start***: per iniciar el dimoni. Si aquest ja s'executa es mostra un missatge d'error.
- ***stop***: per parar el dimoni. Si no s'executa es mostra un missatge d'error.
- ***restart***: reinicia el dimoni i serveix perquè es tornin a llegir els seus arxius de configuració.
- ***reload***: encara que no tots els dimonis ho permeten, aquesta acció permet recarregar els arxius de configuració sense haver de parar el dimoni.

Per exemple:

```
# /etc/init.d/cupsd start
# /etc/init.d/networking restart
```

La primera línia posa en marxa el servei d'administració d'impressores CUPS i la segona reinicia la xarxa llegint els seus arxius de configuració.

Algunes distribucions, entre elles Debian, disposen de l'ordre *service* que permet fer el mateix sense especificar la ruta completa:

```
# service cupsd stop
```

1.5.5 Nivells d'execució

Els dimonis que estan en execució en un moment determinat ens marquen els serveis que un sistema operatiu ofereix com a servidor i que rep com a client. Per organitzar adequadament la quantitat i interacció dels serveis que necessitem en un entorn o circumstància determinats disposem del nivells d'execució (*runlevels*, en anglès). Així, doncs, un nivell d'execució no és més que l'agrupació d'una sèrie de dimonis en execució que té com a finalitat crear un entorn de serveis ajustat a unes necessitats concretes.

Generalment, els sistemes Unix/Linux ens proporcionen diferents nivells d'execució, amb la funcionalitat indicada a la taula 1.4.

TAULA 1.4. Nivells d'execució (runlevels) i la seva funcionalitat

Nivel	Funcionalitat
0	El nivell d'execució 0 està configurat per aturar el sistema.
1	Nivell per a un usuari únic (*single user*). Es posen en marxa els dimonis mínims per fer tasques de manteniment i només permet l'entrada a l'arrel (*root*).
2 a 5	Nivells destinats a ser configurats segons les necessitats de cada instal·lació encara que habitualment tots són multiusuari. En algunes distribucions el nivell 2 està configurat per defecte com a multiusuari sense servei de xarxa, el nivell 3 com a multiusuari amb servei de xarxa, i el nivell 5 per engegar el sistema amb l'entorn gràfic.
6	Aquest nivell està preparat per reiniciar el sistema.

El nivell d'execució de la vostra màquina es pot consultar des de consola amb l'ordre *runlevel*, que requereix privilegis de superadministrador (root)

Cada nivell d'execució té un directori associat a `/etc/rcX.d` (on X és el número del nivell). En aquests directoris hi trobem **enllaços simbòlics** als guions de *shell* que controlen als dimonis i que ja hem vist que estan situats al directori `/etc/init.d`.

A més dels subdirectoris dels diferents nivells, n'hi ha un altre, `/etc/rcS.d`, que conté les referències als serveis bàsics que s'executen prèviament a qualsevol altre nivell.

Vegem per exemple el contingut del directori associat al nivell d'execució 0:

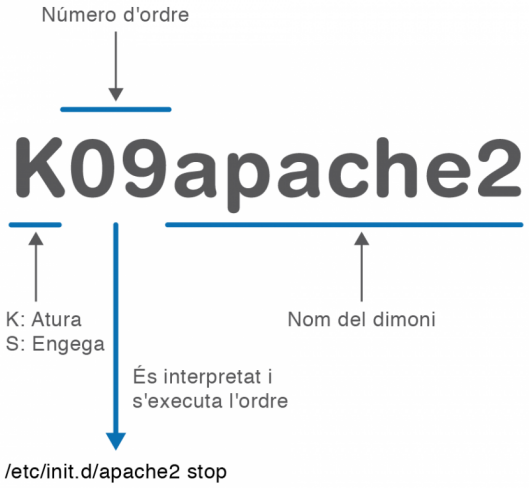
```
# ls /etc/rc0.d

K09apache2 K70vboxadd S15wpa-ifupdown S40umountfs
K10monit K70vboxadd-x11 S20sendsigs S60umountroot
K20netdiag K74bluetooth S30urandom S90halt
K20ntop S35networking
```

La primera lletra del nom d'aquestos enllaços simbòlics porta informació sobre l'acció que han de fer:

- **K**: si l'enllaç comença per *K* (*kill*) indiquem que volem aturar el dimoni.
- **S**: si l'enllaç comença per *S* (*start*) indiquem que volem activar el dimoni.

Després d'aquesta lletra es posa un número de dues xifres, entre 00 i 99, que indica l'ordre en la seqüència d'inici i aturada de serveis. Aquest ordre és important, ja que alguns dimonis tenen dependències, és a dir, necessiten que altres estiguin en execució abans de ser iniciats. Finalment trobem el nom del dimoni en qüestió que ens interessa parar o activar. Vegeu la figura 1.11:

FIGURA 1.11. Sintaxi dels enllaços simbòlics als scripts de control dels serveis



L'ordre per canviar de nivell d'execució és ***init*** i li passem com a paràmetre el nivell d'execució que volem engegar. Així, per reinicialitzar el sistema:

```
# init 6
```

El procés de canvi de nivell és el següent: el sistema inspecciona el directori corresponent al nivell que volem habilitar i començarà primer a detenir els dimonis corresponents a les entrades que comencen per K passant el paràmetre *stop* i després iniciarà els corresponents a les entrades que comencen per S mitjançant el paràmetre *start*.

La modificació de la configuració d'un nivell d'execució es pot fer manualment:

1. Incloure el script de tractament del servei en el directori `/etc/init.d`.
2. Crear els enllaços simbòlics en cada directori de nivell d'execució (`/etc/rcX.d`) donant en el mateix nom la informació de ordre seqüencial d'execució i començant per K o S si el procés s'ha d'aturar o engegar respectivament.

També es pot fer de manera més còmoda amb l'ordre *update-rc.d*.

Ordre update-rc.d

A Debian disposem de l'ordre *update-rc.d* per crear automàticament els enllaços simbòlics necessaris per a cada nivell d'execució. Vegem-ne alguns exemples:

```
# update-rc.d cups defaults
```

Inscriu el servei d'impressió CUPS amb els paràmetres per defecte, és a dir, que s'engegui en els nivells del 2 al 5 i que s'aturi en els nivells 0, 1 i 6. L'ordre d'aturada/engegada serà el 20 per defecte. L'script de servei ha d'estar a `/etc/init.d/cups`.

```
# update-rc.d cups start 10 3 . stop 05 0 1 6
```

En aquest cas només s'engegarà en l'ordre de posició 10 en el nivell 3 i s'aturarà en els nivells 0, 1 i 6 en la posició 05.

```
# update-rc.d -f cups remove
```

El paràmetre *remove* suprimeix els vincles dels diferents directoris, però l'script de servei associat (`/etc/init.d/cups`) ja no ha d'existir. En cas contrari s'ha de fer servir l'opció -f per forçar la supressió dels vincles.

chkconfig

Altres distribucions com Fedora/Red hat i openSUSE fan servir l'ordre chkconfig per configurar els serveis en els diferents nivells d'execució.

Directori /etc/default

Molt sovint cal configurar algunes variables per controlar el comportament dels scripts dels serveis que es troben al directori `/etc/init.d`.

Si aquestes variables es defineixen dins de l'script del servei s'haurien de reconfigurar cada vegada que actualitzem el paquet. Per facilitar la tasca d'administració i garantir que les variables de configuració estan sempre disponibles es poden guardar en arxius específics i independents situats al directori **/etc/default**.

Aquests arxius només han de contenir la declaració de variables i, en tot cas, comentaris explicatius. A continuació posem un exemple del contingut d'un d'aquest arxius:

```
$ cat /etc/default/cups
# Cups configure options
# LOAD_LP_MODULE: enable/disable to load "lp" parallel printer driver module
LOAD_LP_MODULE=yes
```

1.5.6 Sistema d'arrencada

La jerarquia de l'arbre de processos, el seu mecanisme de creació i finalització, el concepte de dimoni i servei i l'agrupació de serveis en diferents nivells d'execució configurables, són part fonamental de la seqüència final d'arrencada del sistema operatiu Linux i la creació del seu entorn de processos i serveis.

El nucli del sistema operatiu finalment està carregat ja a la memòria i activa dos processos previs a la generació de tot l'arbre jeràrquic de processos:

- **El planificador (*scheduler*)**: procés amb PID=0 que gestiona l'assignació de processos a la CPU per a la seva execució.
- **El procés d'inici*init*** amb PID=1. El pare de tots els processos. Tot l'arbre de processos de Linux és fill d'aquest procés.

Per generar l'arbre de processos tenim dos enfocaments: un de seqüencial, en el qual els diferents serveis s'engeguen seguint un ordre prefixat i configurat prèviament en els nivells d'execució, i un enfocament basat en esdeveniments, en el qual els serveis s'inicien depenent de l'ocurrència de determinats successos.

Sistema seqüencial: procés init

Tradicionalment, els sistemes Linux han fet servir un sistema d'arrencada heretat d'Unix System V i que està basat en el procés *init*. Aquest procés l'inicia el nucli del sistema i és l'encarregat de posar en marxa tots els serveis necessaris definits en el nivell d'execució configurat per defecte.

Per realitzar la seva tasca, el procés *init* fa servir la informació continguda en l'arxiu de configuració **/etc/inittab**, que conté, habitualment:

- El nivell d'execució per defecte en arrencar.
- Els scripts que s'han d'executar previs al nivell d'execució per defecte (continguts a /etc/rcS.d).
- La configuració dels diferents nivells d'execució disponibles al sistema.
- L'acció que s'ha de realitzar en prémer *Ctrl+Alt+Del* o amb altres combinacions de tecles.
- La definició de consoles obertes en cada nivell d'execució.

Vegem un exemple de l'arxiu /etc/inittab amb alguns comentaris:

```
1 $ cat /etc/inittab
2
3 # Es defineix el nivell d'execució per defecte.
4
```

```
id:2:initdefault:

# Aquest és el primer script que s'executa previ a qualsevol # nivell d'execuci
    ó

si::sysinit:/etc/init.d/rcS

# Engega el script /etc/init.d/rc i li passa com paràmetre el nivell d'execució
    . Aquest script rc és el que recorre el directori /etc/rcN.d corresponent
    i executa en l'ordre adequat els inicis i finalitzacions dels serveis
    indicats.

l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
l4:4:wait:/etc/init.d/rc 4
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6

# Ordre a executar en cas de CTRL-ALT-DEL

ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now

# Indicacions del terminals tty engegats en cada nivell d'execució. Es veu que
    en els nivells 2 i 3 tenim les sis consoles activades. L'acció respawn vol
     dir que si el procés finalitza, torna a arrencar automàticament.

1:2345:respawn:/sbin/getty 38400 tty1
2:23:respawn:/sbin/getty 38400 tty2
3:23:respawn:/sbin/getty 38400 tty3
4:23:respawn:/sbin/getty 38400 tty4
5:23:respawn:/sbin/getty 38400 tty5
6:23:respawn:/sbin/getty 38400 tty6
```

Un cop s'han acabat d'executar tots aquests scripts, s'executa finalment l'script d'execució local ***/etc/rc.local***.

Sistema basat en esdeveniments: procés upstart

El sistema d'arrencada de System V basat en *init* és un procés síncron que executa seqüencialment una sèrie de tasques definides amb antelació i que no activa un procés fins a haver finalitzat l'anterior. A més, aquest sistema només executa aquestes tasques quan *init* canvia d'estat, cosa que només passa quan la màquina s'encén, s'apaga o s'està reiniciant. Aquest fet fa que *init* no gestioni adequadament altres serveis que és necessari executar no quan es canvia de nivell, sinó en funció de determinats esdeveniments.

Per això, algunes distribucions de Linux estan començant a substituir el tradicional *init* de System V per un sistema basat en esdeveniments anomenat ***upstart***, però intentant sempre mantenir la compatibilitat enrere fent transparent el canvi a l'usuari. De fet, l'arxiu binari que gestiona *upstart* es diu igualment /sbin/init, encara que la seva manera de funcionar és molt diferent.

El model basat en esdeveniments d'*upstart* permet respondre de manera específica a determinats successos i no a seqüències predeterminades. Aquest asincronia fa que es puguin posar en marxa en paral·lel diferents tasques amb l'objectiu de minimitzar el temps d'arrencada.

En el sistema *upstart* l'arxiu de configuració /etc/inittab desapareix i es fan servir uns arxius de definició de treballs (*jobs*, en la terminologia *upstart*) situats al directori /etc/init.

```
ls /etc/init
acpid.conf mountall.conf rcS.conf anacron.conf mountall-net.conf rc-sysinit.
    conf apport.conf mountall-shell.conf rsyslog.conf atd.conf mounted-dev.
    conf tty1.conf
avahi-daemon.conf mounted-tmp.conf tty2.conf
...
```

Aquests arxius d'extensió *conf* defineixen les condicions i els esdeveniments que s'han de produir per engegar o parar els serveis. Vegem un exemple resumit d'un d'aquest arxius:

```
$ cat /etc/init/mysql.conf
# MySQL Service

description "MySQL Server"
author "Mario Limonciello <superm1@ubuntu.com>"

start on (net-device-up
and local-filesystems
and runlevel [2345])
stop on runlevel [016]
respawn

env HOME=/etc/mysql
umask 007
...
exec /usr/sbin/mysqld
end script
```

Veiem que l'arxiu mysql.conf defineix que el dimoni mysqld es posarà en marxa (exec /usr/sbin/mysqld) quan es detecti que hi ha xarxa (esdeveniment net-device-up), el sistema d'arxius ha estat muntat (esdeveniment local-filesystems) i el nivell d'execució sigui multiusuari (entre el 2 i el 5). Si es detecta un esdeveniment de nivell d'execució 0, 1 o 6 el dimoni s'aturarà.

També es configura amb un d'aquests arxius (control-alt-delete.conf) l'acció associada amb l'esdeveniment de teclat *Ctrl+Alt+Supr*:

```
$ cat /etc/init/control-alt-delete.conf

# control-alt-delete - emergency keypress handling
#
# This task is run whenever the Control-Alt-Delete key combination is pressed,
    and performs a safe reboot of the machine.

description"emergency keypress handling"
author"Scott James Remnant <scott@netsplit.com>"

start on control-alt-delete

task
exec shutdown -r now "Control-Alt-Delete pressed"
```

Adopció d'upstart

Les primeres distribucions en fer servir *upstart* com a sistema d'arrencada han estat Ubuntu i Fedora.

Upstart té com un dels seus objectius clau ser compatible amb el sistema Unix System V. Per això:

- Conserva el directori **/etc/init.d** per als scripts de control dels serveis que

encara no s'hagin migrat al sistema d'esdeveniments d'*upstart*.

- Implementa un treball (*job*) específic **/etc/init/rc-sysinit.conf** que simula els canvis de nivell d'execució, gestiona els scripts de /etc/init.d i permet definir el nivell d'execució per defecte.

- Manté l'ús del directori **/etc/default** per als arxius de configuració de variables que permetran ara el control del comportament tant dels scripts tradicionals de /etc/init.d com els arxius de definició de treballs d'*upstart* de /etc/init.

- Per engegar un servei manualment també podrem fer servir la instrucció ***service***.

Seqüència de processos getty, login i shell

Sigui quin sigui el procediment de posada en marxa dels serveis del sistema s'arriba finalment, dins de l'arbre de processos, a activar els terminals o consoles *tty* mitjançant el **procés*getty*** o bé *mingetty*. Aquests terminals es defineixen a l'arxiu **/etc/inittab** (als arxius /etc/init/ttyN.conf, en el cas d'*upstart*) i fan servir els dispositius /dev/ttyN, on N és el número de la consola. En aquest moment es visualitza el contingut de l'arxiu **/etc/issue** i la creació de processos s'atura en espera que l'usuari iniciï una sessió.

PAM

Acrònim de pluggable authentication modules. Consisteix en un mecanisme d'autenticació flexible que permet abstraure les aplicacions del procés d'identificació.

Per iniciar una sessió, l'usuari ha d'identificar-se amb un nom i una contrasenya mitjançant el **procés de login**. Aquestes credencials són verificades a /etc/passwd i /etc/shadow o bé fent servir mòduls PAM.

Una vegada identificat l'usuari, es presenta el contingut de l'arxiu **/etc/motd** (de l'anglès *message of the day*), i es llegeixen diferents arxius de configuració de perfil tan generals com particulars de l'usuari concret, entre ells:

- `/etc/profile`
- `/etc/bash.bashrc`
- `~/.bashrc`
- `~/.profile`

Finalment es carrega l'intèrpret d'ordres o ***shell*** que s'hagi configurat a l'arxiu /etc/passwd per a aquell usuari concret. Hi ha tota una sèrie d'alternatives de *shells* que es poden veure llistades a /etc/shells. Un dels intèrprets d'ordres més emprats és el Bash, evolució de l'original intèrpret *sh*.

La càrrega del *shell*Bash dóna per finalitzat el sistema d'arrencada de Linux i la posada en marxa de l'entorn de serveis, i el sistema operatiu queda en espera de la introducció d'ordres per part de l'usuari.

1.5.7 Aturada del sistema

Ja hem vist que *init* gestiona les aturades del sistema amb els nivells 0 i 6:

- **Nivell d'execució 0**: Para el sistema aturant els diferents processos configurats.
- **Nivell d'execució 6**: Atura el sistema i el reinicia.

Tanmateix, disposem d'una ordre específica, *shutdown*, que ens proporciona funcionalitats addicionals.

Senyals d'aturada

En la seqüència de parada del sistema s'envia primer un senyal SIGTERM a tots els processos actius i, passats uns segons, un senyal d'aturada forçada SIGKILL.

Ordre shutdown

En definitiva, *shutdown* crida a *init* 0 o *init* 6, però accepta paràmetres com ara el temps de termini per a l'apagada o reinicialització del sistema i la possibilitat d'enviar missatges d'advertiment. La seva sintaxi és:

```
shutdown <paràmetres><termini><missatge>
```

Vegeu les opcions de l'ordre en la taula 1.5.

Taula 1.5. Opcions de l'ordre shutdown

opció	Significat
-k	No atura el sistema sinó que envia un missatge d'advertiment a tots els usuaris.
-r	Reinicialització del sistema (*reboot*).
-h	Aturada del sistema (*halt*).
-f	Impedeix l'execució de la utilitat d'anàlisi i correcció del sistema d'arxius (fsck) en l'arrencada.
-F	Força l'execució d'*fsck* en l'arrencada.
-c	Cancel·la l'execució de *shutdown*.

Es pot especificar el termini de tancament del sistema de diferents maneres:

- Una hora i un minut concret amb ***hh:mm***
- Passat un nombre de minuts concret amb ***+m***
- De manera immediata amb ***now*** (àlies de +0)

  En l'exemple es programa una represa per d'aquí a 10 minuts amb un missatge d'avís:

```
# shutdown –r +10 "Represa del sistema en 10 minuts"

Broadcast message from root@ioc–Server (pts/1) (Sun Mar 1 17:58:54 2012):
Reinicialització del sistema per manteniment en 10 minuts
```

```
The system is going DOWN for reboot in 10 minutes!
```

Aquesta represa es pot cancel·lar des d'una altra consola d'arrel amb l'opció *-c*.

```
# shutdown –c "represa cancel·lada"

Broadcast message from root@ioc–Server (pts/3) (Sun Mar 4 18:03:34 2012):
reinicialització cancel·lada
```

Altres ordres de tancament del sistema que encara es conserven per garantir la compatibilitat cap enrere:

- ***halt*** és equivalent a # *shutdown -h now.*
- ***reboot*** és equivalent a # *shutdown -r now.*

1.6 Monitorització de processos a Unix/Linux

Per administrar i gestionar processos en Linux es poden fer servir directament ordres de consola, tot i que també és pot treballar des de l'entorn gràfic amb la utilitat de monitorització del sistema.

1.6.1 El directori virtual /proc

Directori virtual

Definim el directori /proc com a virtual perquè no està realment al disc dur, sinó que el nucli del sistema operatiu el crea dins de la memòria RAM.

En arrencar, el nucli del sistema (*kernel*) posa en marxa un sistema d'arxius virtual /proc que emmagatzema informació que recull del sistema. El directori /proc està implementat a la memòria i no es guarda al disc dur. Les dades que conté són tant de naturalesa estàtica com dinàmica, és a dir, que varien al llarg de l'execució.

Una de les característiques interessants del directori /proc és que hi podem trobar les imatges dels processos en execució amb tota la informació gestionada pel nucli del sistema. Cada procés es pot trobar en un directori etiquetat amb el seu identificador de procés /proc/PID_proces. Aquesta informació és útil per als programes de depuració o per a les mateixes ordres del sistema com ara *ps* o *top*, que la fan servir per veure l'estat en el qual es troben els processos.

D'altra banda, a /proc hi podem trobar també arxius amb informació sobre l'estat global del sistema com ara:

- `/proc/cpuinfo`: informació sobre el processador. Per exemple, el tipus, fabricació, model i rendiment.
- `/proc/devices`: llista de controladors de dispositiu configurats en el nucli que està funcionant actualment.

- `/proc/diskstats`: conté informació estadística de les operacions d'entrada/sortida per cada dispositiu de disc.
- `/proc/dma`: mostra quins canals de DMA (accés directe a memòria) s'estan utilitzant.
- `/proc/filesystems`: llista dels sistemes d'arxius configurats dins del nucli.
- `/proc/ide`: directori d'informació del bus IDE, característiques dels discos.
- `/proc/interrupts`: mostra quines línies d'interrupció s'estan utilitzant i, per a cadascuna, un comptador de quantes n'hi ha hagut.
- `/proc/ioports`: mostra quins ports d'entrada/sortida s'estan utilitzant.
- `/proc/loadavg`: la mitjana de la càrrega del sistema expressada en tres indicadors que representen la mitjana de processos en marxa en el darrer minut, cinc minuts i quinze minuts.
- `/proc/meminfo`: informació sobre la utilització de la memòria, tant la física com la d'intercanvi.
- `/proc/net`: directori amb arxius d'informació d'estat sobre protocols de xarxa. Els fa servir l'ordre *netstat*.
- `/proc/partitions`: conté el noms de les particions del sistema i la seva mida en blocs.
- `/proc/pci`: dispositius PCI del sistema.
- `/proc/stat`: diverses estadístiques sobre el nucli del sistema, com el temps que la CPU ha dedicat a diferents tasques, el nombre d'interrupcions ateses o el temps transcorregut des de l'arrencada del sistema.
- `/proc/version`: la versió de nucli del sistema operatiu.

Tot i que els fitxers que es poden consultar a /proc acostumen a ser arxius de text de lectura fàcil amb l'ordre cat o amb qualsevol processador de textos, de vegades el format en dificulta la interpretació. És per això que hi ha altres ordres del mateix sistema que agafen aquesta informació i la presenten a l'usuari d'una manera més elaborada perquè l'entengui millor. Per exemple, l'ordre *free* llegeix el fitxer /proc/meminfo, i l'ordre *uptime* accedeix i presenta la informació de l'arxiu /proc/loadavg.

1.6.2 Línia d'ordres

El sistema operatiu Linux disposa de diferents ordres en l'entorn de consola de text per a la gestió i monitoratge de processos. Les podríem classificar en diferents apartats funcionals:

- **Monitorització de processos**: ordres *ps*, *pstree*, *pidof*, *time* i *top*.
- **Prioritat dels processos**: ordres *nice* i *renice*.
- **Ordres relacionades amb senyals**: ordres *kill*, *killall*, *nohup*, *disown*, *trap*.
- **Execució diferida**: ordres *at*, *crontab*.
- **Tasques en segon pla**: directiva *&* i ordres *jobs*, *fg* i *bg*.
- **Altres ordres**: *wait*, *sleep*.

A continuació detallarem les ordres de monitorització de processos.

Ordre ps

A la majoria de sistemes operatius de la família Unix, l'ordre *ps* (process status) permet visualitzar els processos en execució. Aquesta ordre està basada en la informació continguda en el directori virtual /proc i admet un gran nombre d'opcions que fins i tot varien en funció dels diferents formats de sintaxi:

- **Estil Unix**: les opcions són lletres majúscules o minúscules precedides d'un guionet. Per exemple: *ps -A*.
- **Estil BSD**: les opcions no porten guionet. Per exemple: *ps r*.
- **Estil GNU**: les opcions fan servir noms llargs i porten doble guionet. Per exemple: *ps – user*.

Les opcions completes de *ps* es troben a les pàgines del manual.

```
# man ps
```

Però per veure un resum de les opcions més comuns podem fer servir l'opció *help* en estil GNU:

```
# ps -–help
```

El resultat es mostra a la figura 1.12:

FIGURA 1.12. Resum de les principals opcions de l'ordre ps

```
root@ioc-Server:~# ps --help
********* simple selection *********  ********* selection by list *********
-A all processes                      -C by command name
-N negate selection                   -G by real group ID (supports names)
-a all w/ tty except session leaders  -U by real user ID (supports names)
-d all except session leaders         -g by session OR by effective group name
-e all processes                      -p by process ID
T  all processes on this terminal     -s processes in the sessions given
a  all w/ tty, including other users  -t by tty
g  OBSOLETE -- DO NOT USE             -u by effective user ID (supports names)
r  only running processes             U  processes for specified users
x  processes w/o controlling ttys     t  by tty
*********** output format **********  ********** long options **********
-o,o user-defined  -f full            --Group --User --pid --cols --ppid
-j,j job control   s  signal          --group --user --sid --rows --info
-O,O preloaded -o  v  virtual memory  --cumulative --format --deselect
-l,l long          u  user-oriented   --sort --tty --forest --version
-F   extra full    X  registers       --heading --no-heading --context
                     ********* misc options *********
-V,V  show version      L  list format codes  f  ASCII art forest
-m,m,-L,-T,H  threads   S  children in sum    -y change -l format
-M,Z  security data     c  true command name  -c scheduling class
-w,w  wide output       n  numeric WCHAN,UID  -H process hierarchy
```

Sigui quin sigui l'estil d'ordre utilitzat, depenent de les opcions invocades apareixeran diferents columnes amb informació sobre els processos. Les dades principals que apareixen en aquests camps estan resumides en la taula 1.6:

TAULA 1.6. Descripció dels camps més habituals de l'ordre ps

Camp	Significat
PID	Identificador del procés
PPID	Identificador del pare del procés
UID	Identificador de l'usuari propietari del procés
USER	Nom de l'usuari propietari del procés
TTY	Terminal associada al procés. Sense terminal associat apareix un *?*
TIME	Temps de CPU acumulat pel procés
CMD	El nom de l'ordre que va iniciar el procés
SIZE	Mida virtual de la imatge del procés
NI	Prioritat *nice* assignada per l'usuari
%CPU	Percentatge de CPU usat pel procés
%MEM	Percentatge de memòria usat pel procés en relació a la memòria física
START	Hora d'inici del procés
VSZ	Memòria virtual en quilobytes
RSS	Memòria resident en quilobytes que fa servir el procés
STAT	Estat del procés com per exemple: * R (*running*): en execució * S (*sleeping*): procés bloquejat en espera d'un succés * T (*stopped*): procés detingut però que es pot reiniciar * Z (*zombie*): procés finalitzat però que roman a la memòria

Si no hi posem cap argument, només es mostraran aquells processos iniciats per l'usuari actual i que fan referència al terminal que aquest està utilitzant.

```
# ps
PID TTY TIME CMD
1532 pts/0 00:00:00 bash
1842 pts/0 00:00:00 ps
```

La primera columna mostra l'identificador del procés, la terminal associada al procés que en aquest cas es tracta del primer pseudoterminal (pts/0), atès que hem fet servir un terminal de l'entorn gràfic, el temps acumulat d'ús de CPU i finalment l'ordre que ha generat el procés.

Afegint algunes opcions (*aux*) en format BSD podem obtenir més informació (vegeu la figura 1.13).

```
# ps aux
```

Figura 1.13. Llistat de processos i informació obtinguda amb l'ordre ps aux

```
root@ioc-Server:~# ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   2032   704 ?        Ss   Mar04   0:03 init [2]
root         2  0.0  0.0      0     0 ?        S    Mar04   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    Mar04   0:00 [migration/0]
root         4  0.0  0.0      0     0 ?        S    Mar04   0:00 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S    Mar04   0:00 [watchdog/0]
root         6  0.0  0.0      0     0 ?        S    Mar04   0:03 [events/0]
 ...
root      2235  0.0  0.1   4784  1992 pts/3    Ss   Mar04   0:00 bash
root      4068  0.0  0.0   3176   500 pts/3    T    Mar04   0:00 yes
jmunoz    4336  0.0  2.7 103160 28600 ?        S    00:18   0:06 gedit
root      5152  0.0  0.1   4552  1760 pts/2    Ss+  01:59   0:00 bash
root      5447  0.0  0.1   3872  1040 pts/3    R+   02:45   0:00 ps aux
```

Per veure més informació, com per exemple els valors de prioritat (vegeu la figura 1.14):

```
# ps -l
```

Figura 1.14. Visualització dels valors de prioritat amb ps

```
root@ioc-Server:~/Documents# ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0  2235  2174  0  80   0 -  1196 -      pts/3    00:00:00 bash
0 T     0  4068  2235  0  80   0 -   794 -      pts/3    00:00:00 yes
4 R     0  5419  2235  0  80   0 -   915 -      pts/3    00:00:00 ps
```

Ordre pstree

L'ordre *pstree* mostra una llista de processos en forma d'arbre que segueix la jerarquia de processos Unix i que permet identificar fàcilment quin és el procés pare d'un altre.

```
$ pstree
init———NetworkManager
      |—3*[VBoxClient———{VBoxClient}]
      |—VBoxService———6*[{VBoxService}]
      |—acpid
      |—apache2———5*[apache2]
      |—atd
      |—avahi-daemon———avahi-daemon
      |—cron
      |—cupsd
      |—6*[getty]
      |—gnome-keyring-d———2*[{gnome-keyring-}]
      |—gnome-screensav
      |—gnome-settings-
      |—gnome-terminal———bash———pstree
      |                 |—bash—man-pager
      |                 |—gnome-pty-helpe
      |                 |—{gnome-terminal}
```

Analitzem la informació mostrada en aquest exemple. Podem veure que *init* és el pare de tots els processos, que hi han alguns dimonis coneguts en marxa (Apache, cron, CUPS, etc.), que tenim sis terminals en espera de login (`6*[getty]`). Al terminal gràfic (*gnome-terminal*) trobem que hi ha un parell de processos Bash en funcionament, i en un d'ells, la mateixa ordre *pstree* que genera l'arbre.

Ordre pidof

L'ordre *pidof* permet trobar els identificadors dels processos associats a una determinada ordre:

```
# pidof init
1
# pidof getty
1110 785 782 780 773 770
```

En l'exemple veiem que el procés *init* té PID=1 i també podem veure tots els PID de les sis consoles obertes.

Ordre time

Aquesta ordre permet executar qualsevol altra aplicació i enregistrar els recursos que ha emprat. Per defecte només presenta el temps real (estimat amb el rellotge del sistema) i el temps de CPU emprat tant en mode usuari com en mode sistema. Tanmateix, aquesta mateixa ordre executada amb privilegis de superadministrador i fent servir diferents modificadors pot subministrar més paràmetres d'informació.

```
# time gedit
real 0m13.289s
user 0m1.380s
sys 0m2.016s
```

Ordre sleep

Suspèn l'execució durant els segons especificats com a paràmetre.

```
# sleep 20;echo "Han passat 10 segons"
Han passat 10 segons
```

Ordre uptime

L'ordre *uptime* dóna l'hora del sistema, el temps que porta encès, la quantitat d'usuaris connectats i la càrrega mitjana del sistema durant l'últim minut, els últims cinc minuts i els últims quinze minuts.

```
# uptime
11:51:25 up 58 min,3 users, load average: 1.38, 1.35, 0.96
```

Ordre vmstat

L'ordre *vmstat* dóna informació de l'estat de la memòria física, de la memòria virtual, de l'intercanvi entre memòria interna i disc (*swapping*), de les transferències de disc, les interrupcions i ús del processador. Admet l'ús de modificadors i permet monitoratge continuat (vegeu la figura 1.15).

Figura 1.15. Resultat de l'ordre vmstat

```
root@ioc-Server:/home/jmunoz# vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa
 0  0      0 139992  14592 234084    0    0   371   225  100  329  4  5 83  9
```

La informació que mostra aquesta ordre és resumeix a la taula 1.7.

Taula 1.7. Camps d'informació de l'ordre vmstat

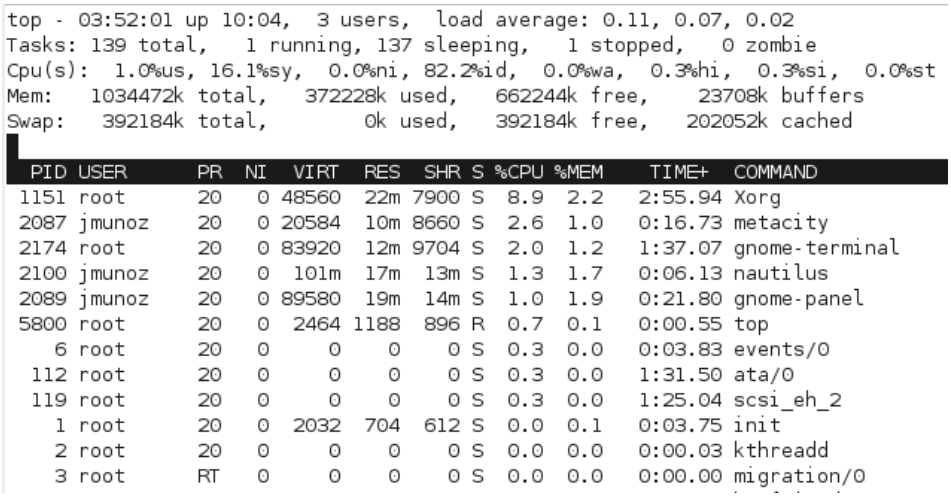
Secció	Descripció
Procs	Processos en espera de ser executats (r) i en estat d'espera (b)
Memory	Memòria virtual (swpd), memòria lliure (free), memòria intermèdia (buff) i memòria cau (cache)
IO	Blocs enviats i rebuts des de dispositius d'entrada/sortida
System	Nombre d'interrupccions per segon (in) i nombre de canvis de context (cs)
CPU	Percentatges de distribució de temps entre el mode usuari (us), mode sistema (sy) i temps ociós (id)

Ordre top

L'ordre *top* presenta la informació dels processos de manera dinàmica i interactiva, en temps real, amb una actualització per defecte cada tres segons, i ordenats pel percentatge d'ús de CPU (vegeu la figura 1.16).

```
# top
```

FIGURA 1.16. Visualització de la ordre de gestió de processos interactiva top



Com es veu al llistat, la primera línia d'informació és la mateixa que dóna l'ordre *uptime*. A continuació mostra informació dels processos en execució amb una sèrie de camps d'informació, semblants als obtinguts amb l'ordre *ps* però amb petites variacions, com podem veure a la taula 1.8.

TAULA 1.8. Descripció dels camps més habituals de l'ordre top

Camp	Significat
PID	Identificador del procés
USER	Nom de l'usuari propietari del procés
PR	Prioritat del procés
NI	Prioritat *nice* assignada per l'usuari
VIRT	Memòria virtual en quilobytes
RES	Memòria resident en quilobytes que fa servir el procés
SHR	Memòria compartida
S	Estat del procés
%CPU	Percentatge de CPU usat pel procés
%MEM	Percentatge de memòria usat en relació a la memòria física
TIME	Hora d'inici del procés
COMMAND	El nom de l'ordre que va iniciar el procés

L'ordre *top* no només mostra la informació actualitzant les dades dinàmicament, també pot interactuar amb l'usuari, que pot actualitzar la informació, ordenar els processos per qualsevol camp d'informació o, el que és més important, finalitzar el procés enviant un senyal de *kill* o fins i tot canviar-li la prioritat.

La llista de les principals ordres de *top* es mostra a continuació, a la taula 1.9:

TAULA 1.9. Descripció de les ordres interactives de l'ordre top

Ordre	Descripció
h o ?	Mostra l'ajuda.
f	Permet marcar com a visibles/ocults els diferents camps d'informació.
o	Configura l'ordre en el qual es presenten els camps d'informació.
F	Defineix el camp d'ordenació del llistat de processos.
k	Finalització d'un procés amb el senyal de *kill*.
r	Canvia la prioritat d'usuari d'un procés (*renice*).
q	Sortir de l'entorn *top*.

Per a un llistat exhaustiu de les opcions i ordres de *top*, cal fer servir l'opció *h* o *?* i s'obté la informació de la figura 1.17.

FIGURA 1.17. Llistat de les principals opcions de top

```
Help for Interactive Commands - procps version 3.2.8
Window 1:Def: Cumulative mode Off.  System: Delay 3.0 secs; Secure mode Off.

  Z,B       Global: 'Z' change color mappings; 'B' disable/enable bold
  l,t,m     Toggle Summaries: 'l' load avg; 't' task/cpu stats; 'm' mem info
  1,I       Toggle SMP view: '1' single/separate states; 'I' Irix/Solaris mode

  f,o     . Fields/Columns: 'f' add or remove; 'o' change display order
  F or O  . Select sort field
  <,>     . Move sort field: '<' next col left; '>' next col right
  R,H     . Toggle: 'R' normal/reverse sort; 'H' show threads
  c,i,S   . Toggle: 'c' cmd name/line; 'i' idle tasks; 'S' cumulative time
  x,y     . Toggle highlights: 'x' sort field; 'y' running tasks
  z,b     . Toggle: 'z' color/mono; 'b' bold/reverse (only if 'x' or 'y')
  u       . Show specific user only
  n or #  . Set maximum tasks displayed

  k,r       Manipulate tasks: 'k' kill; 'r' renice
  d or s    Set update interval
  W         Write configuration file
  q         Quit
          ( commands shown with '.' require a visible task display window )
Press 'h' or '?' for help with Windows,
any other key to continue
```

Eines sysstat

El paquet sysstat és una col·lecció d'eines de monitorització de rendiment per a sistemes Linux. Proporciona dades instantànies de rendiment, que es poden emmagatzemar en arxius històrics. En entorns de servidor, aquestes dades proporcionen informació valuosa per detectar carències i colls d'ampolla del sistema.

L'ordre d'instal·lació és la següent:

```
1 # apt-get install sysstat
```

Algunes de les eines que inclou aquest paquet són:

- ***mpstat***: recull informació del rendiment de cadascun dels processadors del sistema. Permet fer un monitoratge continuat i admet modificadors.
- ***iostat***: és una eina més completa que l'anterior, ja que a més de presentar estadístiques de la CPU, dóna informació dels dispositius d'entrada i sortida, les particions i els sistemes d'arxius.
- ***pidstat***: informació estadística dels processos de Linux.
- ***SAR*** (system activity report): recull, enregistra i presenta informació sobre l'estat dels components principals del sistema (CPU, memòria, discs, interrupcions, interfícies de xarxa, consoles, *kernel*, etc.) i de la seva càrrega en temps real i de manera continuada. Emmagatzema les dades en els fitxers /var/log/saXX on XX és el dia del mes del monitoratge. Aquest fitxers permeten l'anàlisi a posteriori de la informació. *SAR* admet modificadors molt interessants com:
  - `-P` : dóna informació per a cada processador.
  - `-r` : dóna informació del rendiment de la memòria.
  - `-B` : dóna informació relativa a la paginació de la memòria.
  - `-W` : dóna informació relativa al swapping.
  - `-d` : dóna informació del rendiment de disc.
  - `-n` : dóna informació relativa a la xarxa.

Totes les ordres accepten com a paràmetre el nombre de mostres que volem capturar i l'interval entre mostres expressat en segons.

Aquest conjunt d'utilitats ens donen eines en l'àmbit de la monitorització de l'ús del processador i del seguiment de l'activitat dels processos. Vegem-ne un parell d'exemples:

Veure l'ús dels diferents processadors del sistema:

```
# mpstat -P ALL

Linux 2.6.32-28-generic (desktop) 20/02/12 _i686_ (1 CPU)

12:56:13 CPU usr %nice %sys %iowait %irq %soft %steal %guest %
    idle
12:56:13 all 2,51 1,19 14,45 0,42 0,27 0,10 0,00 0,00 81,05
```

Veure el percentatge d'ús dels processos més actius:

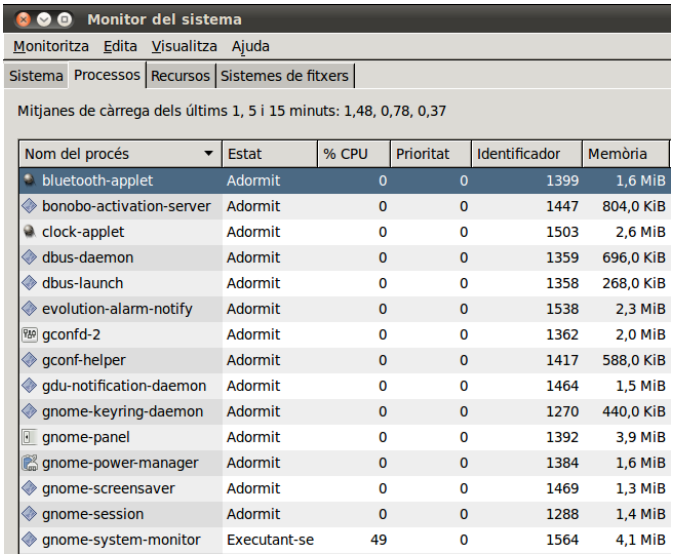
```
# pidstat 1 2

Linux 2.6.32-28-generic (desktop) 26/04/11 _i686_ (1 CPU)

13:00:09 PID %usr %system %guest %CPU CPU Command
13:00:10 984 0,00 1,72 0,00 1,72 0 Xorg
13:00:10 1357 0,00 0,86 0,00 0,86 0 wnck-applet
13:00:10 1868 0,00 3,45 0,00 3,45 0 firefox-bin
13:00:10 2508 2,59 5,17 0,00 7,76 0 pidstat

13:00:10 PID %usr %system %guest %CPU CPU Command
13:00:11 984 1,01 5,05 0,00 6,06 0 Xorg
13:00:11 1868 0,00 3,03 0,00 3,03 0 firefox-bin
```

```
13:00:11 2323 0,00 1,01 0,00 1,01 0 gnome-terminal
13:00:11 2508 0,00 9,09 0,00 9,09 0 pidstat

Average: PID %usr %system %guest %CPU CPU Command
Average: 984 0,47 3,26 0,00 3,72 - Xorg
Average: 1357 0,00 0,47 0,00 0,47 - wnck-applet
Average: 1868 0,00 3,26 0,00 3,26 - firefox-bin
Average: 2323 0,00 0,47 0,00 0,47 - gnome-terminal
Average: 2508 1,40 6,98 0,00 8,37 - pidstat
```

1.6.3 Entorn gràfic (monitor del sistema)

La mateixa eina gràfica **del monitor del sistema**, a més de la visualització en temps real dels recursos, disposa d'una pestanya on podem controlar els processos que s'estan executant i el consum de recursos del sistema que fan (vegeu la figura 1.18).

FIGURA 1.18. Finestra de gestió de processos al monitor de sistema



Per a cada procés actiu en el sistema mostra la informació següent:

- Nom del procés.
- L'estat en què es troba el procés (adormit, parat, en execució).
- El seu identificador (PID).
- El percentatge d'ocupació de processador que està utilitzant.
- La prioritat d'execució.
- La quantitat de memòria que fa servir.

Seleccionant un d'aquests processos podem efectuar diferents accions:

- Posar el procés en espera i rellançar-lo.

- Aturar-lo de manera ordenada o immediata.
- Canviar la seva prioritat d'execució.
- Visualitzar el seu mapa de memòria.
- Visualitzar els arxius oberts amb els quals treballa.

El monitor del sistema es pot configurar canviant l'interval de recollida de dades, els paràmetres de monitorització i el tipus de gràfic de presentació.

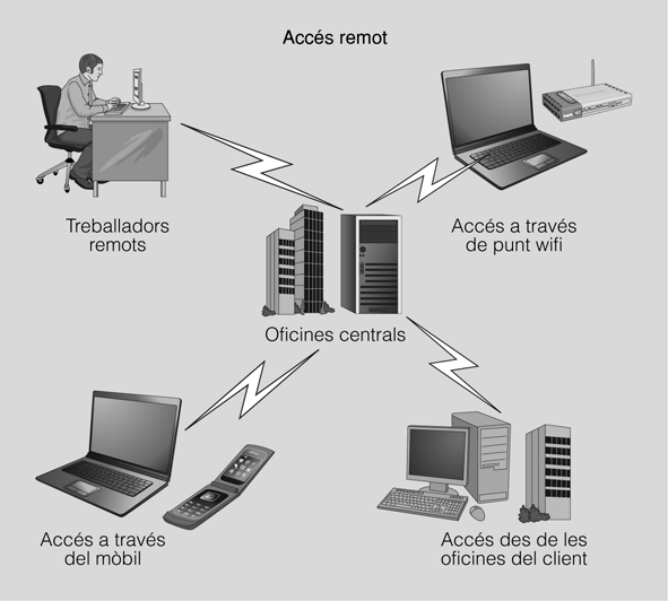
2. Serveis d'accés i administració remota

La generalització de les comunicacions entre ordinadors, tant en xarxa local com mitjançant Internet, ha introduït canvis dràstics en la forma d'accedir als equips i en la seva administració, permetent que pugui fer-se de forma remota. Tant si aquest accés es basa en la línia d'ordres, en una aplicació gràfica o en l'ús del navegador com a interfície gràfica, l'accés i l'administració remota faciliten a l'administrador la tasca de configuració i gestió del sistema informàtic, tenint sempre en compte la capacitat i ample de banda de la comunicació, i la necessària atenció a la seguretat i a la privacitat.

2.1 Introducció a l'accés remot a equips

La capacitat d'accedir remotament a arxius i informació en ordinadors a través d'Internet és interessant tant des del punt de vista de l'administració de sistemes com de l'execució i explotació de qualsevol tipus d'aplicació remota. Pot ser una eina útil per recuperar un arxiu oblidat a l'ordinador o per permetre a un administrador de sistemes modificar la configuració d'un servidor. També és utilitzat per les companyies per accedir a la informació comercial i administrativa del seus sistemes, ja sigui des de les oficines del client o des del domicili dels seus treballadors.

Figura 2.1. Accés remot des de diferents entorns



En un mateix sistema poden coexistir diferents tipus d'accés remot.

Aquest ús tan diversificat fa que hi hagi moltes tecnologies disponibles per

permetre aquest tipus d'accés: des del sistema de fitxers compartit incorporat en la majoria de sistemes operatius fins a eines més específiques desenvolupades per empreses.

L'ús tan divers fa que també hi hagi una gran diversitat de maneres i mitjans d'accedir remotament a informació, tal com podeu veure a la figura 2.1. Per fer front a aquesta diversitat hi ha nombrosos protocols i eines que donen accés remot des d'un portàtil amb connexió Wi-Fi, un telèfon mòbil UMTS o una connexió fixa d'Internet.

Wi-Fi

És l'acrònim de wireless fidelity (fidelitat sense fils) i estableix un conjunt d'estàndards de compatibilitat per a comunicacions en xarxes locals sense fils.

UMTS

És l'acrònim d'*universal mobile telecomunication system*. L'UMTS forma part dels sistemes de comunicacions mòbils de tercera generació (3G), que permeten una gran qualitat de veu, funcions multimèdia i un major ample de banda per Internet.

2.1.1 Usos més freqüents de l'accés remot a ordinadors

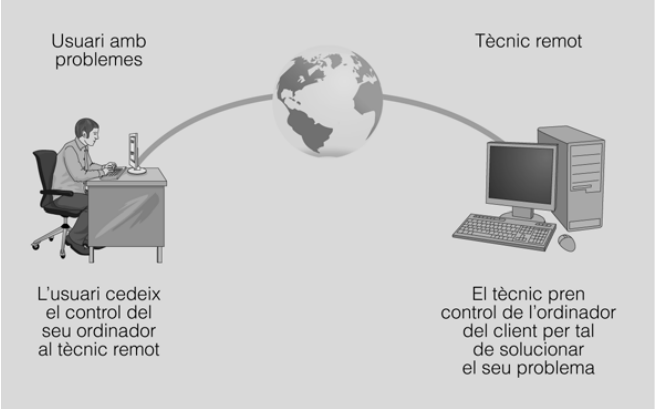
El programari que permet l'administració remota és cada vegada més comú i s'utilitza sovint quan és difícil o molt difícil estar físicament a prop d'un sistema per usar-lo o per tal d'accedir al material d'Internet que no està disponible en la mateixa ubicació.

Els servidors i altres equips de xarxa, per diverses raons, a vegades es distribueixen en distàncies considerables. Fins i tot quan estan relativament a prop, dins del mateix edifici o la mateixa planta, poden estar instal·lats en espais amb accés dificultós o restringit. Per aquestes raons l'administració remota, és a dir l'administració d'un equip des d'un altre equip, és una necessitat quotidiana.

A continuació mostrem una llista dels usos més freqüents de l'accés remot a ordinadors:

- **Administració de sistemes**: gestionar, administrar i configurar equips i servidors de forma remota en xarxa local o mitjançant Internet.
- **Suport i assistència tècnica de forma remota** (*helpdesk*): solucionar problemes tècnics a domicili sense necessitat de desplaçaments físics (vegeu la figura 2.2).
- **Treball a distància**: accés des del domicili als recursos de la xarxa de l'oficina (arxius, escriptori, correus, impressores, etc.), cosa que possibilita el teletreball.
- **Reunions i presentacions en línia**: compartir escriptori amb assistents situats en diferents llocs de treball. Inclou prestacions addicionals com videoconferència i xats de text i de veu.
- **Aplicacions d'ensenyament**: el professor pot monitoritzar les activitats escolars, compartir el seu escriptori i donar ajuda de forma remota.
- **Supervisió d'activitats**: tasques de treballadors, supervisió parental, etc.

FIGURA 2.2. Suport i assistència tècnica remota a usuaris



2.1.2 Tipus d'accés remot

Les tasques d'accés i administració remots es poden dur a terme amb mitjans diferents, però sempre estan condicionats per la xarxa que connecta l'equip que cal administrar amb l'estació on hi ha l'administrador.

La xarxa imposa condicions de:

1. **Capacitat**: si la connexió no té una amplada de banda suficient no serà pràctic treballar amb una interfície gràfica remota. Com més augmenti el retard a la xarxa, més frustrant serà el treball interactiu.
2. **Seguretat**: és obvi que la comunicació entre l'administrador i l'equip remot no ha de ser interceptada per altres usuaris de la xarxa.

Els diferents mitjans d'administració remota es poden agrupar en tres categories bàsiques:

1. Sessió de treball a la consola
2. Sessió de treball amb interfície gràfica
3. Client o eina d'administració local

Cada categoria té els seus punts forts i febles. Una **sessió de treball a la consola** mitjançant **SSH** o l'obsolet i insegur **Telnet**no exigeix grans capacitats a la xarxa. És el mètode més lleuger i fins i tot es pot utilitzar amb comoditat per administrar màquines molt distants o amb xarxes de capacitat escassa. A més, és relativament senzill automatitzar tasques mitjançant guions de programació per repetir les mateixes operacions en un conjunt de màquines.

SSH (*secure shell*) és el nom d'un protocol i del programa que l'implementa, i serveix per accedir a màquines remotes a través de la xarxa. Permet gestionar completament l'ordinador mitjançant l'intèrpret d'ordres.

TELNET

És un protocol que emula un terminal remot per connectar-se a una màquina multiusuari. El seu principal problema és la seguretat.

LAN

Una LAN (de l'anglès local area network o xarxa d'àrea local) és un tipus de xarxa informàtica caracteritzada pel seu caràcter "local" o de distància curta, com ara una casa, una oficina, un hotel, etc., és a dir, la seva extensió està limitada a uns 200 metres i podria arribar a un quilòmetre si es fessin servir repetidors.

L'administració remota amb **interfície gràfica** permet obrir a l'estació local finestres d'aplicacions que s'executen en el servidor remot. Fins i tot permet veure l'escriptori complet de l'estació remota. Tot i que pot ser un mètode de treball còmode, requereix capacitats de la xarxa que normalment només es troben disponibles dins d'una LAN.

Emprar una **eina d'administració local** feta a mida, com un assistent per configurar una impressora remota, o genèrica, com un navegador web, intenta conjugar punts forts de les dues tècniques anteriors. En aquest cas, l'administrador dialoga amb una aplicació local o una pàgina web i no necessita recordar ordres. A més, com que la representació gràfica es produeix localment, no s'exigeix gran capacitat a la xarxa. El problema de les eines específiques és que es tracta d'un programari que cal instal·lar a cada estació que l'administrador vulgui accedir. Sovint aquesta eina només està disponible per a un sistema operatiu concret o fins i tot per a una de les seves versions.

Aquest problema s'obvia amb les interfícies web, ja que un navegador web és un programari comú present en tots els equips. A la taula 2.1 podeu veure un resum dels principals avantatges i inconvenients dels tres tipus d'administració remota.

TAULA 2.1. Els tres principals mitjans d'administració remota

Mètode d'administració	Avantatges	Inconvenients
Treball a la línia d'ordres	Requisits mínims per a la xarxa. És flexible i es pot automatitzar.	Cal conèixer la sintaxi i recordar les ordres.
Interfície gràfica	Visual i flexible. No cal recordar ordres.	Imposa requisits de capacitat a la xarxa.
Client local	Visual i flexible. No cal recordar ordres. No consumeix gaires recursos de la xarxa.	Si no es tracta d'una interfície web (navegador), cal instal·lar programari.

2.2 Administració remota basada en la línia d'ordres

Les connexions remotes mitjançant la línia d'ordres són l'opció que menys capacitats exigeix a la xarxa i per tant es poden fer servir fins i tot en els casos en el qual el canal de comunicació no té una gran amplada de banda disponible o quan el retard és important. Exigeixen conèixer la sintaxi pròpia de l'intèrpret d'ordres emprat i de les seves eines, però són un mecanisme molt flexible que permet fer moltes automatitzacions.

2.2.1 Telnet

Les eines principals per iniciar una sessió de treball remota amb una línia d'ordres són **Telnet** i **SSH**. Totes dues permeten treballar com si s'estigués davant del teclat i de la pantalla de l'estació remota i estan basades en una arquitectura

client-servidor, però les diferències entre elles són importants. **Telnet** encara s'utilitza per raons històriques: és possible que trobem commutadors o servidors d'impressió als quals podem accedir remotament per fer tasques d'administració. Però el fet que la informació es transmeti com a text clar suposa un problema greu de seguretat. Avui dia el seu ús està completament desaconsellat i s'exigeix prendre mesures addicionals.

Protocol telnet

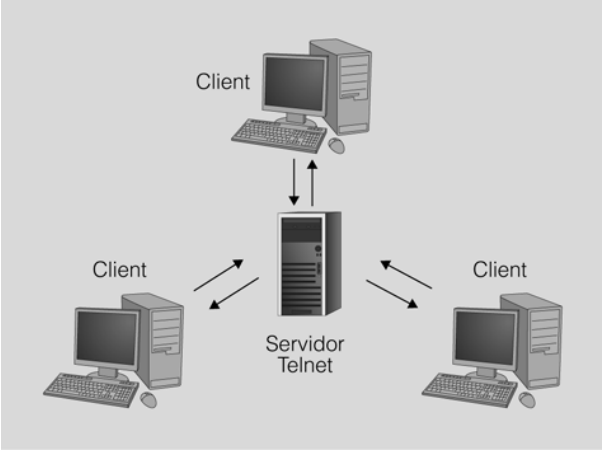
Normalment, el servidor telnet acostuma a esperar les connexions dels clients al port TCP 23. És un protocol antic, desenvolupat el 1969, la seguretat del qual pot veure's compromesa amb qualsevol programari captador de paquets (packet sniffer).

La falta de mesures de seguretat modernes impedeix emprar Telnet de manera generalitzada, però encara és una eina molt eficaç per connectar clients amb servidors que treballen amb text clar (com els servidors HTTP i SMTP) per diagnosticar problemes (figura 2.3).

HTTP

El protocol de transferència d'hipertext o HTTP (hypertext transfer protocol) estableix el protocol per a l'intercanvi de documents d'hipertext i multimèdia al web.

Figura 2.3. Arquitectura client-servidor



El servei Telnet utilitza una arquitectura client-servidor. Els clients es connectaran remotament al servidor Telnet.

Instal·lació de Telnet

En la major part de distribucions GNU/Linux, el client Telnet està instal·lat al sistema de manera predeterminada, però en cas que no ho sigui es pot instal·lar amb la instrucció següent des de l'intèrpret d'ordres:

```
root@ioc-Client:~# apt-get install telnet
```

Normalment, utilitzareu l'eina Telnet per connectar-vos a altres serveis de xarxa (com HTTP o SMTP), però també podeu instal·lar un servidor Telnet a la vostra màquina i utilitzar-lo per fer connexions remotes. Recordeu, però, que el protocol que utilitza Telnet no és segur i que per tant la vostra connexió estarà en perill de ser interceptada per tercers. Si, per exemple, escriviu alguna contrasenya en una connexió Telnet, podrà ser interceptada amb facilitat.

SMTP

SMTP és l'acrònim de simple mail transfer protocol, és a dir, protocol simple de transferència de correu, i és un protocol de xarxa basat en text utilitzat per a l'intercanvi de missatges de correu electrònic.

La instal·lació del servidor Telnet es pot fer executant la instrucció següent des de l'intèrpret d'ordres:

```
root@ioc-Server:~# apt-get install telnetd
```

S'acostuma a anomenar els dimonis amb el nom del servei acabat amb la lletra d. Així, *telnetd* és el dimoni que gestiona el servei Telnet.

Accés remot mitjançant Telnet

Un cop instal·lat el servidor Telnet, ja hi podem accedir des de qualsevol màquina que tingui el client instal·lat. La sintaxi bàsica de Telnet és la següent:

```
telnet <hostname><port>
```

- `<hostname>` representa l'adreça IP o el nom del servidor al qual ens volem connectar.
- `<port>` és el número de port del servei al qual ens volem connectar. De manera predeterminada és el port 23, que és el port utilitzat per Telnet.

A la taula 2.2 podeu veure un llistat d'alguns dels ports TCP més utilitzats habitualment.

TCP

El protocol de control de les transmissions (transmission control protocol) és un protocol orientat a la connexió dintre del nivell de transport del model OSI que permet el lliurament de paquets, en el cas de TCP anomenats segments, de manera fiable.

TAULA 2.2. Ports TCP

Número de port	Servei
21	FTP (*file transfer protocol*): sistema per transferir fitxers.
22	SSH: protocol i programa que l'implementa. Serveix per accedir a màquines remotes a través de la xarxa de manera segura.
23	Servidor Telnet.
25	SMTP: protocol de xarxa basat en text utilitzat per a l'intercanvi de missatges de correu electrònic.
53	DNS (*domain name server*): servei utilitzat per "traduir" les adreces IP a un nom de domini.
80	HTTP: protocol per a l'intercanvi de documents d'hipertext i multimèdia en el web. És el port utilitzat de manera predeterminada pels servidors de llocs web.

Per exemple, si us voleu connectar a un equip que té una IP 192.168.65.10:

```
jmunoz@ioc-Client:~$ **telnet 192.168.56.10**
Trying 192.168.56.10...
Connected to 192.168.56.10.
Escape character is '^]'.
Debian GNU/Linux 6.0

ioc-Server login: **jmunoz**
Password:

Last login: Wed Oct 26 11:28:51 CEST 2011 from ioc-Client.local on pts/1
Linux ioc-Server 2.6.32-5-686 #1 SMP Mon Oct 3 04:15:24 UTC 2011 i686

The programs included with the Debian GNU/Linux system are free software; the
    exact distribution terms for each program are described in the individual
    files in /usr/share/doc/*/copyright.
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
    applicable law.
You have new mail.

jmunoz@ioc-Server:~$
```

Com podeu veure, després de rebre un seguit de dades posem el cursor a la línia d'ordres i podem fer qualsevol tasca com si fóssim a l'equip remot.

Una manera diferent de connectar-nos és obrint una sessió de Telnet directament. Apareixerà l'indicador (*prompt*) *telnet>* i llavors es pot utilitzar la instrucció *open* per obrir la connexió.

```
jmunoz@ioc-Client:~$ **telnet**
telnet> **open 192.168.56.10**
```

Podeu trobar les instruccions més habituals de Telnet a la taula 2.3.

Taula 2.3. Algunes instruccions de Telnet

Instrucció	Servei
c - close	Tancar la connexió.
d - display	Mostrar els paràmetres configurats en la connexió.
o - open hostname [port]	Obrir una connexió en un ordinador remot. El paràmetre *port* és opcional.
q - quit	Sortir de Telnet.
?/h - help	Mostrar l'ajuda de Telnet. Es mostren totes les instruccions disponibles.

2.2.2 Introducció a SSH

SSH és un protocol de xarxa que permet l'intercanvi d'informació de manera segura. Utilitza xifrat i criptografia de clau pública per tal de fer l'autenticació de l'estació remota.

Una de les seves funcions més emprades és iniciar una sessió remota per tal d'executar ordres en substitució de Telnet. Però les seves capacitats són més àmplies, ja que també permet:

- Transmetre fitxers de manera segura.
- Fer còpies de seguretat de manera eficient i segura en combinació amb l'ordre *rsync*.
- Fer túnels per assegurar qualsevol servei que no es transmeti encriptat (HTTP, SMTP, VNC, etc.) o per travessar tallafocs que estiguin bloquejant el protocol.
- Reenviament automàtic de sessions X11 des d'un *host* remot (disposa d'aquesta funció openSSH però no altres implementacions d'SSH).
- Navegar pel web a través d'una connexió amb un servidor intermediari (*proxy)* xifrada amb programes clients que siguin compatibles amb el protocol SOCKS.

- Seguiment automatitzat i administració remota de servidors a través d'un o més dels mecanismes exposats anteriorment.
- Fent servir SSHFS (SSH File System), un sistema d'arxius basat en SSH que pot crear de manera segura un directori en un servidor remot i actuar com a sistema de fitxers en xarxa.

SSG file system

SSHFS és un sistema de fitxers en xarxa que fa servir SSH per comunicar els diferents equips.

Les seves possibilitats es poden combinar de moltes maneres diferents. Ateses les seves característiques criptogràfiques, SSH és una eina fonamental per a l'administrador de xarxa. De les capacitats i funcions esmentades, les més utilitzades són les d'inici de sessió remota, la transferència d'arxius i la tunelització d'altres serveis mitjançant redreçament estàtic de ports o bé establint un servei SOCKS amb el redreçament dinàmic de ports.

2.2.3 Instal·lació d'SSH

SSH utilitza una arquitectura client-servidor, en què el client es connecta a una màquina remota, el servidor. En la major part de distribucions GNU/Linux el client ja hi és però, si es vol accedir a una màquina de manera remota, en aquesta màquina hi haurà d'haver el servidor SSH instal·lat.

La implementació més popular d'SSH és la desenvolupada per la fundació OpenSSH, el servidor openSSH-server. Per procedir a instal·lar-la, executem l'ordre següent des de la consola de la màquina a la qual ens connectarem (servidor).

```
1 root@ioc-Server:~# apt-get install openSSH-server
```

Si no hi ha hagut cap error durant la instal·lació podreu veure un missatge en què es creen les claus necessàries per assegurar les comunicacions segures mitjançant l'encriptació.

```
1 S'està configurant openssh-server (1:5.5p1-6+squeeze1)...
2 Creating SSH2 RSA key; this may take some time ...
3 Creating SSH2 DSA key; this may take some time ...
4 Restarting OpenBSD Secure Shell server: sshd.
5 root@ioc-Server:~#
```

L'última part de la configuració bàsica es fa de manera automàtica quan intentem connectar un client per primera vegada. És la generació automàtica de la clau compartida que utilitzaran client i servidor per assegurar que les comunicacions són segures. Un cop s'ha comunicat la clau compartida entre totes dues estacions, el missatge s'encripta de forma convencional.

Arxius de configuració del client SSH

El client d'OpenSSH es pot configurar de manera prou flexible com perquè l'administrador pugui definir una configuració general per a tot el sistema, perquè

cada usuari pugui modificar els paràmetres adients per a les seves connexions o perquè pugui especificar opcions determinades per a cada connexió individual.

Configuració del client SSH

La configuració del client d'OpenSSH és força flexible. Es pot consultar la documentació oficial al manual SSH_config(5).

El client d'OpenSSH farà servir:

- Les opcions indicades a la línia d'ordres
- Els valors especificats en el fitxer de configuració de l'usuari: **$HOME/.ssh/ssh_config**
- Els valors especificats en la configuració per a tot el sistema: **/etc/ssh/ssh_config**

$HOME

Recordeu que HOME és una variable d'entorn de Linux que conté el camí absolut del directori personal de l'usuari actiu.

Per a cada paràmetre, el client farà servir el primer valor trobat. És a dir, si s'especifica un paràmetre a la línia d'ordres no se'n consultarà el valor en els fitxers de configuració. Dins dels fitxers de configuració és possible definir seccions per a diferents equips (mitjançant la paraula reservada ***host***). Les línies buides i les que comencen amb un # (comentari) seran ignorades. A la taula 2.4 podeu trobar una llista amb les opcions més bàsiques de configuració d'un client.

Taula 2.4. Algunes de les opcions més bàsiques de la configuració d'un client

Opció	Funció
Host <patró>	Permet especificar opcions que només s'aplicaran a les connexions amb l'amfitrió indicat. L'amfitrió s'indica mitjançant patrons amb els caràcters * i ?. Especifica el port de destinació per a la connexió, que per defecte és el 22.
CheckHostIP <yes|no>	El seu valor predeterminat és *yes*. Si l'opció està activada, es comprovarà l'adreça de l'estació remota mitjançant el fitxer *known_hosts* per tal d'advertir un possible enverinament de DNS.
Cipher i Chipers	Permeten especificar respectivament l'algorisme d'encriptació per les a connexions SSH1 i la precedència d'algorismes que cal emprar en les connexions SSH2.
Compression <yes|no>	Si la connexió és molt lenta, la compressió pot millorar els resultats. Si la xarxa té prou amplada de banda normalment no es recomana.
Port <port>	Especifica el port de destinació per a la connexió, que de manera predeterminada és el 22.
RekeyLimit <limit>	Especifica el volum màxim d'informació que es pot transmetre abans d'haver de renegociar la clau de sessió. Es poden fer servir els sufixos K, M o G.
User <usuari>	Especifica l'usuari per establir la connexió a l'estació remota.
SendEnv <variables>	Permet enviar el valor de les variables d'entorn especificades a l'estació remota.

A continuació podeu veure un exemple d'un arxiu `$HOME/.ssh/ssh_config`, corresponent a la configuració d'un client SSH.

```
1 Host 192.168.56.10
2 Ciphers aes128-cbc
3 Compression yes
4 User usuariSSH
```

```
5 Port 30
```

En aquest cas, el fitxer indica que, quan establim una connexió amb el servidor amb la IP 192.56.10, s'ha d'utilitzar un algoritme d'encriptació del tipus aes128-cbc i s'han de comprimir les dades que s'envien. A més, la connexió es farà utilitzant l'usuari *usuariSSH* i s'establirà la connexió amb el port 30 del servidor. Observeu que, perquè la connexió funcioni, el servidor SSH, al seu torn, haurà d'estar configurat per treballar al port 30.

Arxius de configuració del servidor SSH

Configuració del servidor SSH

Es pot trobar informació detallada sobre l'ús del servidor OpenSSH a les pàgines del manual del mateix servidor SSHd(8) i a la dedicada al seu fitxer de configuració SSHd_config(5).

La funció del servidor SSH és esperar les connexions dels clients (normalment al port TCP 22), dur a terme la seva autenticació i, si tot ha anat bé, obrir una sessió de treball, executar una ordre o bé redreçar ports.

Cada vegada que es rep un intent de connexió des d'un client, es fan en primer lloc totes les comprovacions i inicialitzacions criptogràfiques per garantir la seguretat. Després es tracta d'autenticar l'usuari i finalment se segueixen els passos d'un procés d'inici de sessió habitual.

Fitxer de configuració sshd_config

Malgrat que en el funcionament del servidor d'OpenSSH hi intervenen diversos fitxers, l'arxiu principal de configuració és **/etc/ssh/sshd_config**. Aquest fitxer conté diferents paraules clau amb el seu valor. Les línies que comencen amb **#** (comentaris) o les que estan en blanc són ignorades.

A la taula 2.5 podeu trobar els principals paràmetres de configuració d'un servidor SSH.

TAULA 2.5. Paràmetres de configuració del servidor SSH

Opció	Funció
AllowGroups	Si s'especifica seguida d'una llista de grups (separats per espais), només els usuaris que tenen algun dels grups indicats com a grup principal o suplementari podran iniciar sessió. És possible emprar els patrons *?* i * en la definició dels grups. Només es poden indicar els grups mitjançant el seu nom, no en format numèric (GID).
AllowUsers	Té la mateixa funció que *AllowGroups*, però per als usuaris. En aquest cas, a més, és possible indicar des de quins amfitrions d'origen s'acceptarà la connexió. Per exemple: *AllowUsers usuari1@192* usuari2*.
Banner <fitxer>	Envia el contingut del fitxer indicat al client abans de dur a terme l'autenticació.
Compression <yes|delayed|no>	Especifica si es farà servir la compressió. El valor *delayed*, l'opció predeterminada, indica que només es farà servir la compressió un cop s'hagi autenticat l'usuari.
DenyGroups	Permet especificar una llista de grups, separats per espais, als quals no es permetrà iniciar sessió. Es poden especificar els grups mitjançant el seu nom, emprant els patrons *?* i * de manera opcional.

TAULA 2.5 (continuació)

Opció	Funció
DenyUsers	Igual que *DenyGroups*, però per als usuaris. En aquest cas és possible indicar un equip (o subxarxa) per a cada usuari.
Port ListenAddress	Permeten especificar el port on el servidor escoltarà les connexions dels clients i les adreces on obrirà aquest port. De manera predeterminada s'utilitza el port 22 de qualsevol adreça local. És possible especificar múltiples vegades aquestes opcions, però convé que *Port* sempre aparegui abans que *ListenAddress*.
LoginGraceTime	Període de temps màxim per dur a terme l'autenticació. El valor 0 expressa que no hi ha límit.
MaxAuthTries	Nombre màxim d'intents d'autenticació que es poden fer.
MaxStartups	Nombre màxim de connexions simultànies que encara no han completat la seva autenticació.
PasswordsAuthenticati on <yes|no>	Indica si s'accepta l'autenticació mitjançant contrasenya (*password).*
PermitEmptyPasswords <no|yes>	Si s'utilitza l'autenticació mitjançant contrasenya, especifica si el servidor permet la connexió a comptes que tenen una contrasenya buida.
PermitRootLogin <yes|without-password|forced-commands-only|no>	Especifica si s'accepta la connexió de superusuari mitjançant SSH. El valor *without-password* indica que el superusuari no podrà fer servir l'autenticació basada en contrasenya i el valor *forced-commands-only*, que només es permetrà l'autenticació de clau pública per executar certes ordres de manera remota (normalment per fer còpies de seguretat).
Protocol	Especifica quins protocols es podran fer servir en les connexions dels clients (1, 2 o tots dos). És important recordar que el protocol SSH2 és força més segur que l'SSH1.
PubkeyAuthentication <yes|no>	Especifica si s'acceptarà l'autenticació de clau pública.
X11Forwarding <no|yes>	Especifica si s'acceptarà el reenviament X11 per tal que les aplicacions gràfiques executades en el servidor obrin la seva finestra en el servidor X del client.

Altres arxius de configuració

Hi ha altres arxius de configuració que permeten de forma genèrica el filtratge, control d'accés i mecanismes de protecció de diferents serveis (POP, Sendmail, Telnet, SSH, etc.) actuant de fet com un tallafocs bàsic.

Així, si volem habilitar o restringir l'accés a determinats equips i serveis podem editar els arxius de configuració **/etc/hosts.deny** i **/etc/hosts.allow** indicant en la directiva dintre de l'arxiu el servei que volem controlar, en aquest cas, el dimoni SSH. D'aquesta manera el sistema, davant d'una petició d'accés al servei, fa la cerca següent, que conclou en el moment de la primera coincidència:

1. Comprova l'arxiu /etc/hosts.allow. Si hi troba coincidència valida l'accés.
2. Comprova l'arxiu /etc/hosts/deny. Si hi troba coincidència no valida l'accés.

3. En cas de no trobar coincidència en cap dels arxius valida l'accés.

Exemples de directives d'aquests arxius:

1. sshd: ALL (permet/denega l'accés ssh a tothom)
2. sshd: 192.168.56.10 (permet/denega l'accés SSH de la IP 192.168.56.10)

Recordeu que perquè qualsevol canvi tingui efecte s'ha de reiniciar el servei:

```
/etc/init.d/ssh restart
```

Per a informació completa sobre els arxius de configuració `/etc/hosts.allow` i `/etc/hosts.deny` consulteu la informació del sistema amb l'ordre: *man hosts_access*.

2.2.4 Connexió a una estació remota amb SSH

La funció més comuna per a SSH és establir una sessió de treball remota fent ús de tècniques criptogràfiques per transmetre la informació. L'ús del client SSH és força senzill.

```
ssh user@host
```

- *user*: és l'usuari que es connectarà a la màquina remota.
- *host*: representa la IP o el nom de domini del servidor SSH al qual ens volem connectar.

És tot el que hem d'escriure per iniciar una sessió remota com a usuari (*user*) en l'equip amfitrió (*host*). En executar l'ordre ens demanarà la contrasenya de l'equip remot i, si l'escrivim de manera correcta, podrem accedir a la sessió de treball remota per escriure ordres.

```
jmunoz@ioc-Client:~$ ssh jmunoz@192.168.56.10
jmunoz@192.168.56.10's password:

Linux ioc-Server 2.6.32-5-686 #1 SMP Mon Oct 3 04:15:24 UTC 2011 i686
The programs included with the Debian GNU/Linux system are free software; the
    exact distribution terms for each program are described in the individual
    files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
    applicable law.

No mail.

Last login: Wed Oct 26 12:20:51 2011 from ioc-client.local

jmunoz@ioc-Server:~$
```

Per finalitzar la connexió escriurem *exit*.

```
jmunoz@ioc-Server:~$ exit
logout
Connection to 192.168.56.10 closed.
jmunoz@ioc-Client:~$
```

Si no s'especifica el nom d'usuari a l'hora d'invocar l'ordre SSH, intentarà fer la connexió amb l'usuari amb el qual estem connectats al terminal de GNU/Linux.

```
usuari1@ioc-Client$ ssh 192.168.56.10
usuari1@192.168.56.10's password:
```

Fixeu-vos que en aquest cas intenta connectar-se com a *usuari1*, ja que no s'ha especificat amb quin usuari s'ha de connectar.

El client d'**OpenSSH** disposa de diferents opcions que es detallen en el seu manual. Algunes de les més freqüents són les descrites a la taula 2.6.

Taula 2.6. Opcions més freqüents del client OpenSSH

Opció	Funció
-1	Força l'ús de la versió 1 del protocol SSH. Només es recomana emprar SSH1 per connectar-se a servidors antics que no són compatibles amb SSH2.
-2	Força l'ús de la versió 2 del protocol: SSH2.
-4	Força l'ús de l'adreçament IPv4.
-6	Força l'ús de l'adreçament Ipv6.
-C	Activa la compressió *gzip* en la connexió. Es recomana activar la compressió si s'està emprant SSH amb un enllaç lent, com un mòdem. Si l'enllaç és de banda ampla es recomana treballar sense compressió.
-p port	Port al qual es connectarà en l'equip remot. De manera predeterminada el servidor SSH s'executa al port TCP 22, però si es tracta d'un servidor accessible des d'Internet és recomanable escollir un altre port per evitar els intents de connexió.
-q	No imprimeix els missatges d'advertència, només els errors. Amb una altra *-q* no imprimeix ni els errors.
-X	Activa la retransmissió X11 per tal que els programes gràfics llançats en l'estació remota obrin la seva interfície gràfica en el servidor X local.
-x	Desactiva la retransmissió X11.

Algunes vegades només es desitja executar una ordre a l'estació remota, no obrir un ***shell*** (intèrpret d'ordres) per treballar. En aquest cas, és possible indicar l'ordre que cal executar en la mateixa crida de SSH.

tail

És una instrucció de GNU/Linux que permet veure les 10 últimes línies. El paràmetre *n* permet especificar el nombre de línies que es vol mostrar. En aquest cas, se'n mostren tres.

Per exemple, per tal de veure el final del fitxer de registre /var/log/messages al servidor 192.168.56.10 farem:

```
jmunoz@ioc-Client:~$ **ssh jmunoz@192.168.56.10 tail /var/log/messages**
jmunoz@192.168.56.10's password:
```

2.2.5 Transferència d'arxius entre equips

Entre les utilitats incloses en la distribució d'**OpenSSH** trobem les ordres ***scp*** i ***sftp***. Aquestes ordres permeten transferir fitxers amb totes les garanties de seguretat d'SSH.

Còpies segures (scp)

La sintaxi bàsica de la instrucció *secure copy* (*scp)* és:

```
scp [opcions] [[user@]host1:]fitxer1 [[user@]host2:]fitxer2
```

- host1: representa la màquina d'origen.
- host2: representa l'estació de destinació.

L'ordre ***scp*** es pot veure com una versió estesa de l'ordre ***cp***, que permet copiar fitxers fins i tot entre màquines diferents. De fet, és el substitut d'SSH per l'ordre ***rcp***, que és antiga i insegura. En fer una còpia mitjançant ***scp***, es pot escollir qualsevol combinació de fitxers locals o remots tant per l'origen com per la destinació.

remote copy

rcp és una instrucció GNU/Linux que permet copiar fitxers entre l'equip local i un de remot.

En l'exemple que podeu veure a continuació copiem el fitxer local manual.pdf en una estació remota amb IP 192.168.56.10.

```
jmunoz@ioc-Client:~$ **scp manual.pdf 192.168.56.10:**
jmunoz@192.168.56.10's password:
manual.pdf 100% 179KB 179.5KB/s 00:00
```

Si la contrasenya és correcta es procedirà a la transferència de l'arxiu i s'indicarà de manera interactiva el percentatge que ja s'ha enviat, la grandària en quilobytes, l'índex de transferència i el temps transcorregut des del començament de la transmissió.

No només es poden copiar fitxers locals en una estació remota, sinó que es poden copiar fitxers de l'estació remota en la local i fins i tot entre dues estacions remotes. A la taula 2.7 podeu trobar un exemple de cada tipus.

TAULA 2.7. Exemples de diferents usos de la instrucció scp

Exemple	Ordre
Copiar fitxer local a equip remot	scp fitxer usuari@192.168.56.10:/home/usuari
Copiar fitxer remot a equip local	scp usuari@192.168.56.10:/home/usuari/fitxer ./
Copiar un fitxer d'un equip remot a un altre equip remot	scp usuari@192.168.56.12:/home/usuari/fitxer usuari@192.168.56.11:/home/usuari

La pàgina del manual d'***scp*** ens mostrarà els seus paràmetres, la major part dels quals són els mateixos que té l'ordre ***ssh***. No obstant això, té algunes opcions

pròpies que són particularment útils i que podeu veure a la taula 2.8.

Taula 2.8. Opcions de la instrucció scp

Opció	Funció
-l limit	Establir un límit a l'amplada de banda en kbps.
-p	Preservar el temps de modificació, accés i els permisos dels fitxers copiats.
-r	Fer una còpia recursiva pels directoris.

Transferències segures de fitxers (sfpt)

L'ordre ***sftp*** és un substitut per al client d'FTP, que és el tradicional però que és insegur. En emprar *sftp* s'estableix una connexió al sistema remot i després, de manera interactiva, es podran indicar ordres per explorar el sistema d'arxius remot i fer modificacions.

El protocol de transferència de fitxers o FTP (de l'anglès *file transfer protocol*) és un programari estandarditzat per enviar fitxers entre ordinadors amb qualsevol sistema operatiu. Forma part de la capa d'aplicació del model TCP/IP.

La sintaxi bàsica de la instrucció *sftp* és:

```
1 sftp [opcions][[user@]host1:]fitxer1 [[user@]host2:]fitxer2
```

- host1: representa la màquina d'origen.
- host2: representa l'estació de destinació.

Un cop establerta la connexió segura, podem executar les mateixes ordres que si ens haguéssim connectat mitjançant FTP. A la taula 2.9 podeu trobar les ordres que *sftp* pot interpretar.

Taula 2.9. Ordres que poden ser interpretades per sftp

Ordre	Funció
bye, exit, quit	Finalitzar la sessió.
cd camí	Canviar el directori remot.
chgrp, chown, chmod	Canviar el grup, el propietari o els permisos en el sistema remot .
get fitxer	Transmetre el fitxer remot indicat al directori de treball local.
put fitxer	Transmetre el fitxer local indicat al directori de treball remot.
ls, mkdir, pwd, rename, rm, rmdir, ln	Elaborar llistats, crear directoris, consultar la ruta, canviar de nom, eliminar fitxers i directoris i crear enllaços simbòlics al sistema de fitxers remot.
lcd, lls, lmkdir, lpwd	Versions de les ordres *cd*, *ls*, *mkdir* i *pwd* per treballar al sistema de fitxers local.

Vegem un exemple de com fer transferències segures entre l'ordinador local i un equip amb la IP 192.168.100.10:

```
jmunoz@ioc-Client:~$ **sftp jmunoz@192.168.56.10**
jmunoz@192.168.56.10's password:
Connected to 192.168.56.10.
sftp>
```

En aquest punt ja s'ha establert la connexió segura amb l'equip remot i ja es poden començar a utilitzar les ordres pròpies d'*sftp*. En l'exemple següent es transferirà el fitxer fitxer.pdf de l'equip remot al local.

```
sftp> **cd Documents/**
sftp> **get manual.pdf **
Fetching /home/jmunoz/Documents/manual.pdf to manual.pdf
/home/jmunoz/Documents/manual.pdf 100% 179KB 179.5KB/s 00:00
sftp>
```

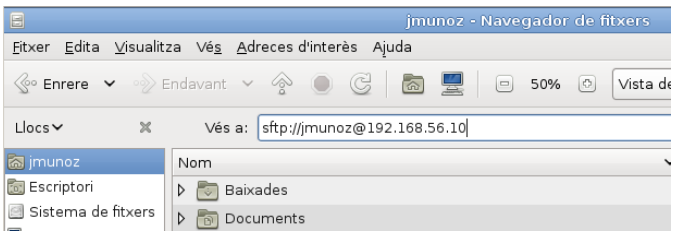
sftp a l'escriptori GNU/Linux

Els escriptoris **GNOME** i **KDE** també proporcionen un accés senzill a servidors SSH per tal de treballar en xarxa. A GNOME, el navegador d'arxius **Nautilus** pot mostrar directoris remots com si fossin locals fent servir el GNOME VFS (sistema de fitxers virtual de GNOME).

Com que la distribució Debian utilitza GNOME per defecte, per tal d'utilitzar gràficament la instrucció *sftp* al visualitzador d'arxius Nautilus, només és necessari introduir a la barra de lloc la mateixa ordre que s'utilitza per obrir una connexió SSH mitjançant ***Vés > Ubicació*** (o bé *Ctrl+L*). Vegeu la figura 2.4.

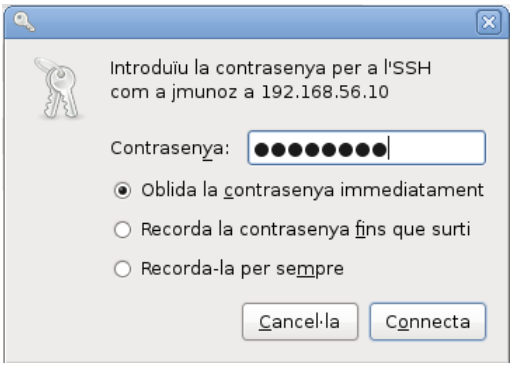
```
sftp://nomUsuari@ipServidor
```

FIGURA 2.4. Introduir a la barra d'adreces l'ordre de connexió ssh



Després d'uns segons per establir la comunicació, el navegador d'arxius Nautilus detecta que es vol fer una connexió segura i demana la contrasenya de l'usuari de l'equip remot amb una finestra similar a la de la figura 2.5.

FIGURA 2.5. Nautilus demana la contrasenya per connectar-se a l'equip remot



Si la contrasenya és correcta, s'estableix la connexió remota i Nautilus passa al mode de transferència segura d'arxius i la unitat remota queda muntada. A la figura 2.6 es pot observar que es visualitzen els arxius de l'equip remot.

FIGURA 2.6. Nautilus mostra la unitat remota per fer la transferència segura d'arxius

Utilitzant aquesta finestra, sempre que es tinguin els permisos necessaris, es poden copiar arxius, crear carpetes o fer qualsevol altra operació típica d'un navegador de fitxers.

2.2.6 Redreçament de ports TCP i túnels amb SSH

La major part dels protocols que utilitzem en les nostres comunicacions estan basats en dissenys de fa gairebé 30 anys, quan la seguretat en xarxes telemàtiques no era un problema. Telnet, FTP, POP3, protocols d'ús quotidià, descuiden la seguretat i confidencialitat de les dades que envien. No serveix de res protegir els servidors, implantar una bona política de contrasenyes i actualitzar les versions dels nostres dimonis si després, quan un usuari de POP3, per exemple, vol veure el seu correu electrònic des de la nostra xarxa, envia el seu usuari i contrasenya en text pla per la xarxa.

Fent servir SSH es poden establir túnels encriptats pels quals es pot transmetre qualsevol protocol que faci servir TCP. Així és possible, per exemple, emprar un túnel SSH per:

- descarregar el correu mitjançant POP3 i enviar-lo fent servir SMTP.

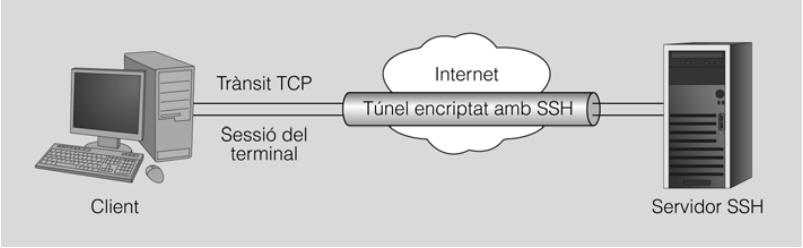
- accedir a un servidor web mitjançant HTTP.

- accedir a un sistema d'arxius remot mitjançant NFS.

Els protocols de l'exemple (POP3, SMTP, HTTP i NFS) no utilitzen tècniques criptogràfiques. El seu ús en una xarxa que no és de confiança no ens permet garantir la confidencialitat. Però com que tots aquests serveis utilitzen TCP com a protocol de transport, poden ser tunelitzats per SSH.

La idea en què es basa aquest procediment, i que es representa a la figura 2.7, és la de fer un túnel pel qual viatjaran les dades de manera segura (*tunneling*). A cada un dels extrems del túnel hi ha les aplicacions estàndard (un dimoni POP3 estàndard, el nostre client de correu preferit, FTP, un navegador web, etc.) i la comunicació s'assegura fent ús de tota la potència criptogràfica d'SSH. SSH recull les dades que el client vol enviar i les reenvia pel túnel o canal segur. A l'altre costat del túnel es recullen les dades i es tornen a enviar al servidor corresponent.

FIGURA 2.7. Túnel fet amb SSH per fer segures les comunicacions que utilitzen el protocol TCP



Redreçament estàtic de ports: túnel SSH per accedir al servei SMTP

El redreçament estàtic de ports ens permet crear un canal de comunicació segur i transparent a l'usuari entre un port de la màquina local i un altra port de la màquina remota, que pot ser un dels ports estàndards que fan servir qualsevol dels serveis de xarxa.

Per exemple, per establir un túnel SSH entre el port local 10025 i el port 25 (corresponent al servei SMTP) de l'estació 192.168.10.100 establint la connexió amb l'usuari *usuari* faríem:

```
1 ssh usuari@192.168.10.100 –L 10025:192.168.10.100:25
```

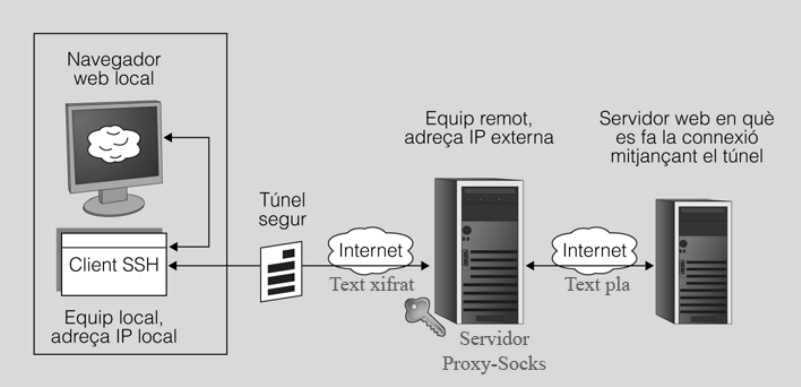
Un cop establerta la sessió SSH, i mentre es mantingui, existirà un túnel encriptat entre el port TCP local 10025 i el 25 de l'estació 192.168.10.100.

En altres paraules, per fer una connexió segura amb el servidor SMTP de l'estació remota, hauríem de connectar amb el port local 10025. SSH s'encarregarà de fer el transport de manera segura entre totes dues estacions.

Redreçament dinàmic de ports: Servidor SOCKS

Una altra de les possibles aplicacions del túnel SSH és accedir a un servidor web des d'una IP diferent de la de la nostra màquina. Com es pot veure a la figura 2.8, el navegador utilitza el túnel creat per SSH per connectar-se al servidor i navegar per pàgines web utilitzant la IP externa del servidor.

Figura 2.8. Navegar per Internet utilitzant la IP del servidor

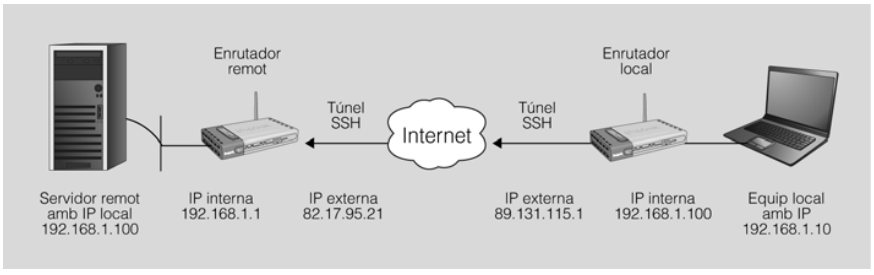


SSH permet doncs crear un túnel des de l'equip local fins a un servidor remot. Un cop connectats a aquest servidor remot utilitzem el túnel per fer la navegació per Internet. A la figura 2.9 podeu veure les IP de cada un dels equips de l'exemple. Fixeu-vos que tant la màquina local com el servidor remot tenen la seva pròpia IP local i una IP externa (NAT). Per connectar-nos al servidor remot haurem d'utilitzar l'adreça IP externa de la xarxa on està connectat, i només funcionarà si l'encaminador al qual ens connectem té el port 22 redreçat al servidor amb el qual farem el túnel. És a dir, quan l'encaminador remot rebi una connexió pel port 22 ha de redreçar aquest trànsit al port 22 del nostre servidor.

NAT (network address translation)

La traducció d'adreces de xarxa és una tècnica que amaga un espai d'adreces, que generalment consisteix en un conjunt d'adreces de xarxa privada, darrere d'una única adreça IP, sovint en l'espai d'adreces IP públiques. Aquest mecanisme s'implementa en un encaminador que utilitza unes taules per assignar les adreces "ocultes" a una sola adreça IP, de manera que els paquets a la sortida semblen originar-se en l'encaminador.

Figura 2.9. Configuració d'equips per fer un túnel SSH



Abans de fer el túnel SSH, si es fa una connexió a Internet des de l'equip local amb un navegador qualsevol, es farà des de l'adreça IP 89.131.115.1.

A continuació veiem com fer el túnel amb l'equip remot:

```
1 ssh –D 1080 –p 22 –Nf usuari@82.17.95.21
```

L'opció *-D* habilita el **redreçament dinàmic de ports locals** que a la pràctica permet fer servir SSH com un servidor SOCKS.

Un **servidor SOCKS** dóna a la intranet un servei similar al que proporciona un servidor web intermediari (*proxy*), però no està limitat al protocol HTTP/HTTPS, sinó que permet redreçar qualsevol tràfic TCP/IP. Fins i tot la versió 5 de SOCKS permet el tràfic UDP.

SOCKS

Protocol d'Internet que facilita l'encaminament de paquets de xarxa entre les aplicacions client-servidor a través d'un servidor intermediari.

El servidor SOCKS funciona mitjançant l'assignació d'un sòcol per escoltar el port local. En el nostre cas hem triat el port 1080, ja que és l'estàndard per a aquest servei. Cada vegada que s'estableix una connexió a aquest port, la connexió es transmet pel canal segur. A continuació el protocol d'aplicació és l'encarregat de determinar on es fa la connexió a la màquina remota.

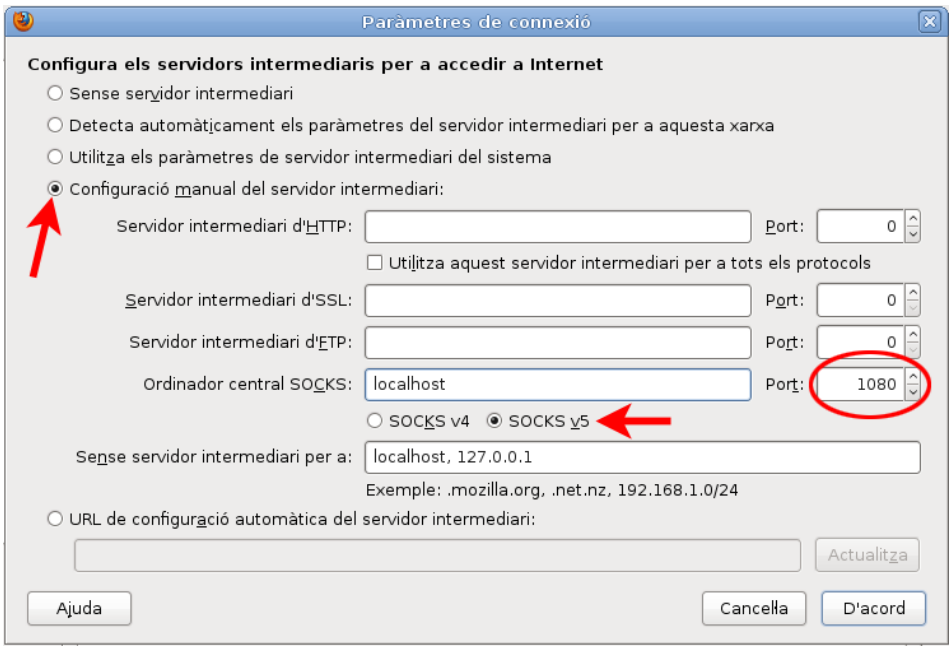
A la taula 2.10 veiem amb detall les opcions emprades.

Taula 2.10. Opcions de redreçament dinàmic de l'ordre ssh

Opció	Descripció
-D 1080	Habilita el redreçament dinàmic de ports fent servir, en aquest cas, el port 1080 estàndard per a servidors SOCKS.
-p 22	Permet indicar el port pel qual es farà el túnel SSH, habitualment el número 22.
-N	Especifica que no es vol executar cap ordre remota i no s'obrirà cap terminal interactiu. És d'utilitat quan es fa servir SSH per al redreçament de ports.
-f	Com que no ens cal interactivitat, aquesta opció fa que SSH quedi en segon pla i es dissocii de la *shell* actual, cosa que converteix el procés en un dimoni (cal l'opció -N).

Un cop establert el túnel, cal configurar a nivell d'aplicació el programa que el farà servir. En aquest cas l'aplicació que hem de configurar és el navegador local perquè utilitzi el túnel SSH per accedir a la xarxa d'Internet.

Figura 2.10. Configuració del navegador Firefox per utilitzar el servidor SOCKS



Si utilitzeu el navegador Firefox, heu d'anar a *Edita* > *Preferències* > *Avançat*, seleccionar la pestanya *Xarxa* i prémer el botó *Paràmetres*. S'ha de configurar, tal com mostra la figura 2.10: fer servir la configuració manual del servidor intermediari (*proxy*), que l'ordinador central SOCKS sigui *localhost* i que utilitzi el port 1080.

Ara el navegador ja utilitza el túnel SSH i quan es fa una connexió a Internet des de l'equip local es fa des de l'adreça pública de l'encaminador remot (en aquest cas, amb IP 82.17.95.21).

A Linux hi ha diverses aplicacions que permeten a les aplicacions no compatibles amb SOCKS fer les seves peticions TCP/IP a través d'un servidor SOCKS. Les més conegudes són tsocks i proxychains.

Hi ha aplicacions que no admeten específicament l'ús del servidor SOCKS i que no disposen de la possibilitat de configurar aquesta opció. En aquest cas es pot carregar un programa addicional que detecti peticions a la pila TCP/IP per modificar-les i redreçar-les a través del servidor SOCKS sense que la aplicació hagi de tenir implementada aquesta possibilitat.

2.3 Administració remota amb interfície gràfica

Si la xarxa de comunicacions té una amplada de banda suficient i un retard raonable, es poden fer servir eines que permeten treballar amb aplicacions que s'executen en un equip remot, però que mostren la seva interfície gràfica en un terminal local. En tot cas, la seguretat ha de ser un punt fonamental a considerar, atès que la funcionalitat administrativa de l'aplicació farà que transportem per la xarxa dades importants.

És possible que tot l'escriptori visualitzat es correspongui amb l'escriptori d'un equip remot o combinar en el mateix escriptori aplicacions locals i remotes. En aquest últim cas, fins i tot les aplicacions remotes es poden estar executant en diferents equips.

També cal diferenciar els protocols de comunicació remota, que són convencions que defineixen la comunicació entre la màquina client i la que ofereix el servei de les aplicacions client-servidor que fan servir aquests protocols per donar un servei concret d'accés gràfic remot.

Així, doncs, trobem que hi ha diferents aplicacions que implementen aquest protocols i permeten dur a terme l'administració remota d'equips mitjançant una interfície gràfica. Concretament algunes de les opcions més conegudes són:

- El sistema servidor X Window (transparència de xarxa)
- Computació virtual en xarxa VNC (*virtual network computing*)
- Webmin, eina gràfica per a l'administració remota de Linux

2.3.1 Protocols d'accés remot a interfícies gràfiques

Hi ha dues propostes tecnològiques principals quant a protocols d'accés remot a interfícies gràfiques. D'una banda, es poden transmetre directament les diferents directives que donen instruccions als motors gràfics dels servidors de finestres respectius. De l'altra, es pot transmetre informació relativa a cada píxel que permet fer una representació del mapa de memòria gràfica del gestor de finestres.

Alguns dels protocols poden incorporar les seves pròpies tècniques de seguretat i xifrat o bé incorporar una capa addicional de seguretat, per exemple fent servir túnels SSH.

Protocol X11

Protocol desenvolupat inicialment a mitjan anys vuitanta a l'Institut Tecnològic de Massachusetts (MIT) com a part del sistema de finestres X Window amb l'objectiu de proporcionar una interfície gràfica als sistemes Unix.

> La versió del protocol que es fa servir des de 1987 i fins a l'actualitat és la v11. Per aquesta raó s'acostuma a anomenar X11 tant al protocol com als servidors i clients del sistema X Window.

Aquest protocol és independent del sistema operatiu i permet la interacció gràfica remota entre un client i un servidor fent transparent la xarxa per a l'usuari, que veu el sistema com si fos un terminal gràfic virtual.

Tecnologia NX

FreeNX

És una implementació en codi lliure amb llicència GPL de la tecnologia client-servidor NX.

Més que un protocol de comunicació pròpiament dit, NX és una tecnologia que permet gestionar i millorar les connexions X Window fent servir compressió del tràfic de dades del mateix protocol X11 i afegint mecanismes de memòria cau per accelerar la comunicació i reduir el requeriments d'amplada de banda i latència.

Remote framebuffer (RFB)

El protocol de xarxa RFB (remote framebuffer protocol) també és lliure i fa servir la tècnica d'enviar informació dels píxels que conformen la memòria d'imatge gràfica. Malgrat que les funcions inicials del protocol RFB es limitaven a transmetre els esdeveniments desencadenats per l'usuari en el teclat i el ratolí locals a l'estació remota, i a enviar en direcció contrària el contingut gràfic de la pantalla, les darreres versions han estat ampliades per oferir més funcionalitats, com per exemple la transferència de fitxers.

La memòria d'imatge

La memòria d'imatge (*framebuffer*) és l'àrea de la memòria que emmagatzema les dades que defineixen la imatge gràfica que apareix a la pantalla. Normalment, les pantalles actuals estan basades en píxels, és a dir, mostren mapes de bits en comptes d'imatges vectorials.

La memòria d'imatge conté la informació necessària per definir l'estat de cada píxel de la pantalla. La quantitat d'informació necessària dependrà de la resolució i la profunditat de color (la quantitat de bits necessaris per codificar el color, i potser la transparència, de cada píxel).

Els programes més coneguts que fan servir el protocol RFB són tota la família d'aplicacions **VNC** (*virtual network computing*).

El protocol RFB fa servir per defecte el port 5900 del servidor per establir la comunicació. Alternativament, en cas de connexió a través d'un navegador es fa servir per defecte el port 5800.

Cal assenyalar que RFB no és un protocol segur. En fer servir VNC en una xarxa que no sigui de confiança, caldrà combinar el seu ús amb SSH o bé una altra tecnologia per crear una xarxa privada virtual (VPN).

VPN

Una xarxa privada virtual (VPN) és una xarxa informàtica que s'aplica en una capa lògica addicional a la d'una xarxa existent. Té el propòsit de crear un àmbit privat de les comunicacions per ordinador o fer una extensió segura d'una xarxa privada en una xarxa insegura, com Internet. Una aplicació comuna és protegir les comunicacions a través de la xarxa pública d'Internet.

Remote desktop protocol (RDP)

Protocol propietari desenvolupat per Microsoft, que el fa servir en el seu servidor de serveis d'escriptori (*terminal services*). Incorpora els seus propis sistemes de compressió i xifrat i fa servir el port TCP3389 per defecte per escoltar les peticions al servidor.

Microsoft té lògicament el seu propi client (*terminal services client*) però hi ha també clients per a una gran varietat de sistemes operatius (Windows Mobile, Linux, Unix, Mac OS X, Android, etc.).

Altres protocols

Hi han altres protocols menys coneguts, com ara ICA (*Independent Computing Architecture*), producte propietari de la companyia Citrix Systems per als seus servidors d'aplicacions, o bé AIP (A*daptive Internet Protocol*), protocol propietari que fa servir Sun.

2.3.2 El servidor X Window

El sistema X Window implementa les funcions necessàries per controlar finestres i dispositius d'entrada com el ratolí o el teclat. Aquest sistema s'utilitza en la major part de versions d'Unix per implementar la interfície gràfica. Val a dir que X Window no defineix com ha de ser aquesta interfície, només permet implementar-la. Escriptoris tan coneguts a GNU/Linux com GNOME, KDE o Xfce utilitzen el mateix sistema X Window malgrat que són escriptoris diferents amb les seves particularitats.

Entre les característiques pròpies del sistema X Window trobem que és independent del sistema operatiu emprat, només es tracta d'una capa d'aplicació que, des del principi, va ser pensada per treballar en xarxa.

X.Org

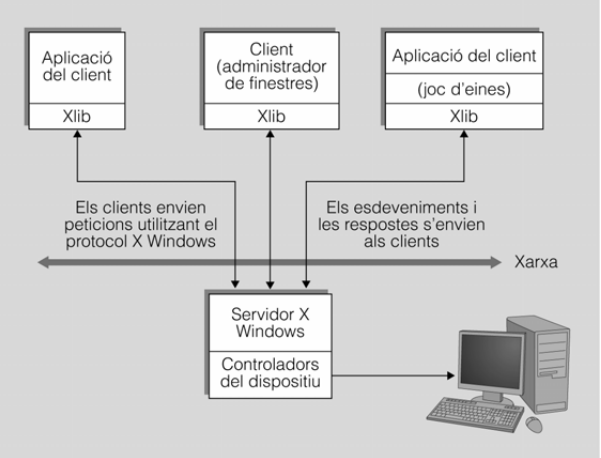
Logotip de X.Org Foundation

Existeix la fundació que gestiona el desenvolupament del sistema X Window com la implementació de referència d'aquest sistema. És programari lliure i des del seu inici com a ramificació del projecte XFree86 ha guanyat popularitat. Actualment, és la implementació del sistema X Window emprada en gairebé totes les distribucions GNU/Linux i en altres sistemes operatius.

El servei de finestres X Window fa servir el protocol X11 i permet separar en una xarxa l'estació que representa la interfície gràfica de l'estació on s'executa realment l'aplicació, de forma nativa i transparent per a l'usuari. Aquesta característica del sistema de finestres X Window s'anomena també **transparència de xarxa**.

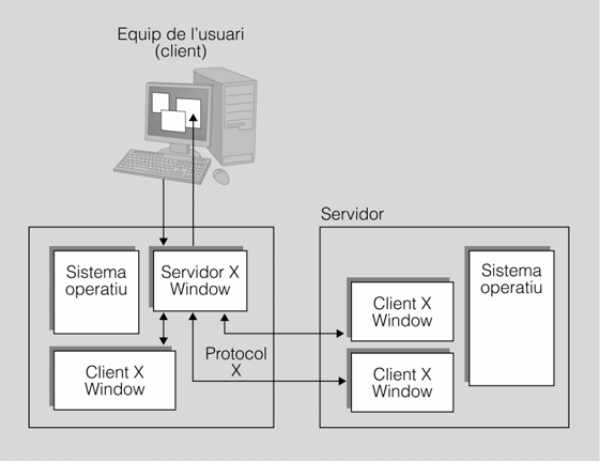
La figura 2.11 representa el funcionament d'X Window i il·lustra el comportament de la transparència de xarxa, tot i que, en molts casos, client i servidor gràfic poden estar al mateix ordinador.

FIGURA 2.11. Transparència del sistema X Window



En la nomenclatura del sistema X Window, el programari que permet dibuixar finestres rep el nom de servidor X i els programes que obren les seves finestres en el servidor X s'anomenen **clients X**. Així, el servidor X s'executarà en l'estació local de l'usuari per tal que aquest pugui interaccionar amb la interfície gràfica, l'estació remota que executa una aplicació serà el client X. Aquest ús dels termes servidor i client centrat en el punt de vista del programari sovint despista els usuaris novells. A la figura 2.12 es pot observar una arquitectura client-servidor; en aquest cas, és el client d'aquesta estructura el que proporciona el servei X Window (el client de l'aplicació fa de servidor de finestres X Window).

FIGURA 2.12. Estructura client-servidor des del punt de vista de l'aplicació



La comunicació entre el servidor X i el client X es fa sense cap tipus d'encriptació; per això només es podrà fer servir en xarxes de confiança. En la resta de casos serà necessari emprar SSH per tunelitzar el trànsit del sistema X Window, o bé crear una xarxa privada virtual (VPN).

Configuració d'un servidor X Window

Bàsicament, un servidor X11 representa un terminal gràfic amb els seus propis teclat i ratolí. Per tal que els clients puguin emprar aquest terminal és necessari fer dues accions:

1. Configurar el servidor X11 perquè permeti la connexió dels clients.
2. Indicar als programes clients on és el servidor X11.

Configurar un servidor X11 de manera poc curosa és un risc de seguretat molt greu. Qui es connecta al servidor X11 rep missatges amb notificacions, per exemple, de les tecles premudes i dels moviments del ratolí. Així, doncs, cal establir qui es pot connectar a un servidor X11.

Per fer aquesta funció es poden fer servir dos mètodes: ***xhost*** i ***xauth***. El primer mètode permet especificar permisos per a diferents equips en indicar-ne l'adreça IP o el seu DNS. Si es concedeix accés a un equip determinat, qualsevol aplicació d'aquell amfitrió (sigui quin sigui el seu usuari) podrà accedir al servidor X11.

El segon mètode està basat en una clau anomenada *galeta màgica* (*magic cookie*), que és un conjunt arbitrari de dades que es crea en activar el servei. Cada client que vulgui connectar-se al servidor ha de provar que té coneixement d'aquesta galeta.

Tanmateix, potser l'alternativa més senzilla i eficaç per solucionar la qüestió de la seguretat sigui la utilització d'un túnel fent servir un protocol de xifrat com ara SSH (secure shell).

Per tal de configurar un servidor X11 per permetre connexions dels clients, es poden emprar les ordres *xhost* i *xauth*. La primera suposa riscos de seguretat i la segona és confusa d'emprar. Per això, la millor opció és utilitzar SSH per redreçar el trànsit X11.

D'altra banda s'ha d'indicar als programes clients on és el servidor X Window que han d'emprar per obrir les seves finestres. Per fer això disposem de la variable d'entorn DISPLAY. Aquesta variable d'entorn utilitza una cadena amb la sintaxi següent:

```
host:display.screen
```

A la taula 2.11 podeu veure els principals paràmetres de configuració d'un servidor gràfic X11.

TAULA 2.11. Paràmetres de configuració d'un servidor X11

Paràmetre	Significat
Host	Adreça IP de l'estació on s'executa el servidor X11. Si no es defineix se suposa que es tracta de l'equip local.
:display	Nombre de displays dins del servidor X11. Igual que un equip físic pot tenir diferents terminals virtuals (/dev/ttyX), un servidor X11 pot tenir diferents displays. Normalment el primer és :0
.screen	Nombre de pantalles dins del display. Normalment no s'especifica, en aquest cas es fa servir la pantalla .0

Exemples

```
#export DISPLAY=:0
```

S'està indicant que s'utilitzi el primer terminal gràfic del servidor X de l'equip local.

```
#export DISPLAY=192.168.0.50:2
```

S'està indicant que s'utilitzi el segon terminal gràfic del servidor X de l'equip remot amb la IP 192.168.0.50.

```
#export DISPLAY=192.168.0.50:2.1
```

S'està indicant que s'utilitzin el segon terminal gràfic i la pantalla 1 del servidor X de l'equip remot amb la IP 192.168.0.50.

Iniciació d'un client X mitjançant SSH

Servidors X

Si estem fent servir GNU/Linux, la nostra distribució empaquetarà la versió estable d'un servidor X, probablement X.Org. Per a altres sistemes operatius disposem de programari lliure, com Cygwin/X o Xming, que permeten veure aplicacions gràfiques Unix a l'escriptori de Microsoft Windows.

Si el nostre equip està executant un servidor X, podrem establir una sessió SSH en un equip remot per executar allà qualsevol aplicació i que la seva interfície gràfica es representi al nostre servidor X.

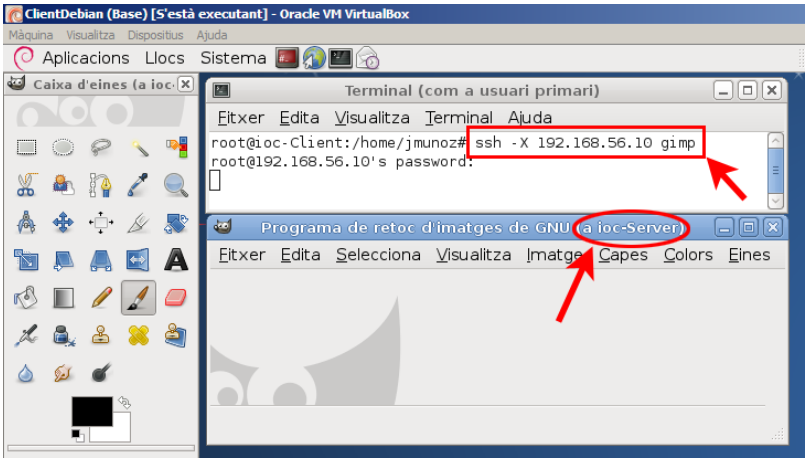
Aquesta és probablement la manera més senzilla i segura de protegir la comunicació entre servidor i client X Window. En aquest cas SSH s'encarrega de:

- Tots els procediments criptogràfics necessaris per establir un túnel entre els dos equips i dur a terme l'autenticació de l'usuari remot.
- Establir el valor adient per a la variable d'entorn DISPLAY de l'equip remot per tal que les aplicacions gràfiques utilitzin el servidor X11 local (és a dir, el que s'executa de manera local en l'estació on hem executat SSH per connectar-nos a l'equip remot).
- Transportar tot el trànsit X11 protegit dins del túnel SSH.
- Permetre l'ús del servidor X11 local per a les aplicacions de l'estació remota. Només per a l'usuari propietari de la sessió SSH, no per a la resta de processos dels altres usuaris.

A la figura **??** es mostra una captura de pantalla en què s'ha emprat SSH per iniciar una sessió de treball en una estació remota en la qual s'ha executat el programa GIMP. En aquest cas és important observar que en fer la connexió SSH s'ha indicat el paràmetre *-X* per tal d'activar el redreçament de trànsit X11.

```
#ssh –X jmunoz@192.168.56.10 gimp
```

FIGURA 2.13. Execució de GIMP en una estació remota



L'aplicació GIMP s'està executant en l'ordinador servidor remot (en aquest cas ioc-Server). Aquest envia la informació gràfica mitjançant un canal segur SSH a l'ordinador client (que actua com a servidor gràfic X Window) i mostra les finestres resultants a l'usuari.

Cal indicar que el servidor SSH haurà de tenir activat el redreçament del protocol X (normalment està activat per defecte), és a dir que haurà de tenir el paràmetre següent en una línia de l'arxiu de configuració `/etc/ssh/ssh_config`:

```
// Activació del redreçament X en /etc/ssh/sshd_config
X11Forwarding yes
```

La combinació del sistema client-servidor X Window en xarxa amb les característiques de seguretat d'SSH suposa una combinació de flexibilitat, seguretat i facilitat d'ús difícilment igualable per altres tècniques. Tot i així, cal recordar que és important que la xarxa tingui una bona amplada de banda i una latència petita, ja que la comunicació client-servidor genera gran quantitat de peticions-respostes (anomenades en anglès *roundtrip*) que poden alentir el sistema. Tal com hem comentat, per intentar millorar aquests inconvenients hi ha tecnologies i programes per gestionar connexions X Window amb eficiència, com ara FreeNX.

XDMCP per accedir a escriptoris remots

Normalment, en arrencar un ordinador amb GNU/Linux apareix una pantalla gràfica demanant nom d'usuari i contrasenya per iniciar sessió. Aquest programa, que en cas d'autenticar l'usuari carrega tot l'escriptori, és un administrador de pantalla (display manager). El protocol que controla aquest programa respon a l'acrònim XDMCP (*X display manager control protocol* o protocol de control d'un

administrador de pantalla) i funciona sobre un servidor X Window. Té diferents implementacions, de les quals les més usuals són: **gdm** per a l'escriptori GNOME, **kdm** per a l'escriptori KDE, i **xdm**, el més bàsic per llençar les X.

En la major part dels casos l'administrador de pantalles gestiona un servidor X11 que s'està executant de manera local. Però gràcies al protocol **XDMCP** també pot gestionar servidors X11 remots. En aquest cas, un equip pot mostrar la benvinguda (demanant usuari i contrasenya) en un equip remot i, després de dur a terme una autenticació positiva, mostrar a l'usuari un escriptori remot, no només la finestra d'una aplicació.

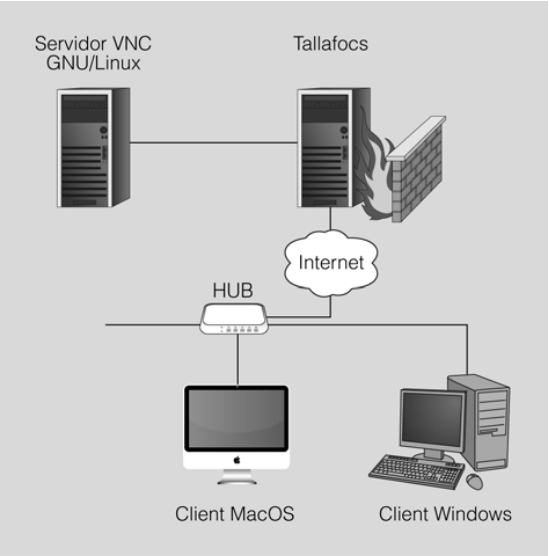
La configuració de tots els elements necessaris és força extensa per tractar-la aquí, però és convenient saber que la transparència de xarxa del sistema X Window permet mostrar tot un escriptori remot, no només les finestres aïllades de les aplicacions remotes.

2.3.3 Virtual network computing (VNC)

VNC és un sistema per veure, i si s'escau també controlar, un escriptori remot. Originalment va ser desenvolupat per Olivetti i hi ha implementacions gratuïtes de les eines que permeten treballar amb VNC en sistemes operatius diferents.

Aquesta aplicació permet accedir des d'un ordinador anomenat client a un altre ordinador anomenat servidor connectats mitjançant una xarxa. Un cop s'ha establert la comunicació, l'equip client pot utilitzar l'equip servidor sense cap limitació, llevat de les imposades per l'amplada de banda de la xarxa. El resultat és una visualització en pantalla de l'equip remot, de l'escriptori i de tots els seus programes.

FIGURA 2.14. Execució remota d'un sistema GNU/Linux en dues màquines client amb sistemes operatius diferents



Per aconseguir això, cal instal·lar dos programes: un servidor VNC a la màquina

a la qual vulguem accedir i un visualitzador VNC a la màquina client. És possible utilitzar un visualitzador en un sistema operatiu determinat i un servidor en un de diferent. D'aquesta manera, és possible executar un sistema Windows complet des d'un Macintosh o GNU/Linux, o fer qualsevol combinació que puguem imaginar. Com es pot veure a la figura 2.14, es pot executar un sistema GNU/Linux des d'un equip amb sistema operatiu Mac OS o Windows.

Funcionament de les aplicacions VNC

VNC fa servir el protocol RFB (*remote framebuffer*) que està basat en una memòria d'imatge i així pot treballar amb qualsevol sistema de finestres, ja sigui X Window, Windows o un altre. En aquest sistema sempre s'anomena client l'estació en què es troba l'usuari (equip local) i es denomina servidor l'equip que defineix el contingut de la pantalla (equip remot).

Compte! La nomenclatura per definir el client i el servidor és inversa a la que utilitza el sistema X Window.

Tot el sistema està pensat per treballar amb clients lleugers (*thin clients*) de manera que el client es pugui implementar sense gaires recursos. De fet, tot el protocol i el funcionament del sistema intenten ser molt senzills. El pilar fonamental són les actualitzacions gràfiques que el servidor envia cap al client, en forma de rectangles, per modificar les dades de la memòria d'imatge. Aquests missatges poden emprar diferents directives, però n'hi ha unes de bàsiques que tots els clients i servidors VNC han de conèixer.

Una característica fonamental del sistema VNC és que el client no conserva cap estat. Així és possible tancar un client i tornar-lo a obrir, fins i tot en un altre equip, com si es tractés d'un monitor virtual.

En els sistemes GNU/Linux és possible exportar mitjançant VNC l'escriptori actual o fins i tot un terminal de text, o bé crear un escriptori virtual nou. Els sistemes Microsoft Windows no permeten la creació d'escriptoris virtuals nous, només permeten exportar en xarxa l'únic escriptori disponible. En el moment d'exportar un escriptori es poden definir dues contrasenyes diferents, una per a aquells usuaris que podran controlar de manera remota aquest escriptori i una altra per a aquells usuaris que el podran veure però no interactuar amb el teclat i el ratolí.

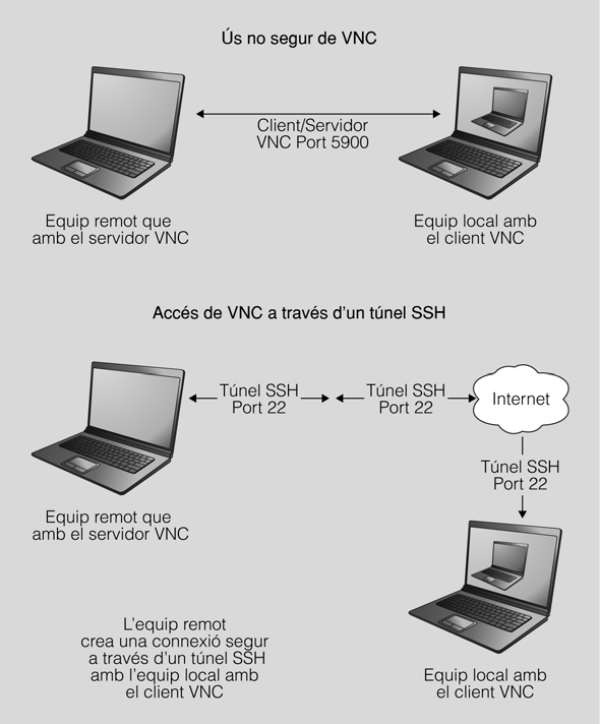
El procés per fer servir VNC és el següent:

1. A l'estació remota cal executar un servidor VNC per exportar l'escriptori de treball o un de nou.

2. A l'estació local es farà servir un client VNC per interactuar amb el servidor remot.

A la figura 2.15 es pot veure un esquema d'una connexió VNC a través d'un túnel SSH.

FIGURA 2.15. Ús no segur de VNC i ús a través d'un túnel SSH



La major part de servidors VNC inclouen un petit servidor web que ofereix la descàrrega d'una miniaplicació Java que es pot fer servir com a client VNC. D'aquesta manera, en l'ordinador local només cal un navegador compatible amb Java per poder accedir a l'escriptori remot.

Implementacions VNC

Si el desenvolupament original de VNC té algun hereu oficial, és RealVNC. Un producte desenvolupat per la companyia del mateix nom, que està formada per alguns dels desenvolupadors originals. Les eines de RealVNC s'ofereixen amb llicències diferents: l'edició Free, amb llicència GPL, i la resta, amb llicències privatives.

GPL

La llicència pública general (GPL, de l'anglès *general public license*) és un tipus de llicència per a programari que permet la còpia, distribució (comercial o no) i modificació del codi, sempre que qualsevol modificació es continuï distribuint amb la mateixa llicència GPL. La llicència GPL no permet la distribució de programes executables sense el codi font corresponent.

Una altra implementació força estesa és TightVNC, programari lliure amb llicència GPL que, tot i que és plenament compatible amb les eines originals, introdueix algunes extensions al protocol RFB. Aquestes extensions només es podran emprar en fer servir TightVNC, tant en el client com en el servidor. Entre les extensions introduïdes per TightVNC hi ha codificacions noves que redueixen la càrrega de la xarxa i la possibilitat de transmetre fitxers.

A la taula 2.12 podeu trobar un llistat dels productes més populars que implementen VNC.

TAULA 2.12. Programaris que implementen VNC

Programari	Característiques
VNC	És el programari original. En l'actualitat només s'utilitza com a referència i per a les proves de compatibilitat.

TAULA 2.12 (continuació)

Programari	Característiques
TightVNC	Una versió que s'ofereix gratuïtament amb llicència GPL. S'han millorat significativament els algorismes de compressió de VNC i permet la transferència de fitxers.
RealVNC	És una versió desenvolupada per alguns dels desenvolupadors d'AT&T. L'equip de RealVNC també està oferint-ne una versió empresarial i ha elaborat un producte molt interessant que pretén ser accessible per a qualsevol client VNC.
UltraVNC	UltraVNC ofereix una funció de transferència d'arxius, una gestió millorada de la compressió de vídeo i una funció de xat. Recentment s'ha afegit l'opció de controlar una sola finestra del programa en lloc de controlar tot l'escriptori.

Altres eines relacionades amb VNC i que cal destacar són **Vino**, una eina de GNOME per compartir l'escriptori i que acostuma a incloure Ubuntu. Tenim també diversos clients VNC com **Vinagre** i el potent client VNC que inclou la distribució Debian anomenat **Remmina**, que no només és compatible amb el protocol RFB de VNC sinó també amb la tecnologia NX o el protocol RDP de Microsoft Windows.

2.3.4 Programari d'accés remot TightVNC

El programari d'accés remot **TightVNC** és una de les implementacions de VNC més avançades. Es distribueix amb llicència GPL, la qual cosa permet un accés lliure al seu codi font, a la seva distribució i a la seva instal·lació.

Instal·lació del servidor TightVNC

El paquet del servidor TightVNC s'anomena *tightvncserver* i per instal·lar-lo només cal escriure el següent a la línia d'ordres:

```
# apt-get install tightvncserver
```

Un cop instal·lat ja podem arrencar el servidor VNC, *vncserver*, juntament amb una sèrie d'opcions.

```
# vncserver :1 -geometry 1024x768 -depth 16 -pixelformat rgb565
```

Opcions:

- `:1`: indica la pantalla en la qual s'establirà la sessió per fer les connexions remotes.
- `-geometry`: estableix les dimensions de la finestra amb què es farà la connexió.

- `-depth`: estableix la profunditat dels colors en bits per píxel. Ha de ser un valor entre 8 i 32.

Contrasenya

Atenció! Hi ha un límit de 8 caràcters per a la contrasenya de TightVNC.

- `-pixelformat`: estableix el format de la representació dels píxels.

A continuació, el programari ens demanarà dues contrasenyes, la segona de les quals és opcional. La primera és la necessària per permetre l'accés total a l'equip servidor des de l'equip remot. La segona és per permetre només un accés de visualització de l'escriptori.

```
#You will require a password to access your desktops.
Password:
Warning: password truncated to the length of 8.
Verify:
Would you like to enter a view-only password (y/n)? n
```

Per aturar el servidor TightVNC es pot fer servir l'ordre *vncserver* i s'indica la pantalla en què s'ha invocat.

```
#vncserver -kill :1
Killing Xtightvnc process ID 4223
```

Instal·lació del client TightVNC

El paquet que permet instal·lar el client TightVNC es diu *xtightvncviewer*.

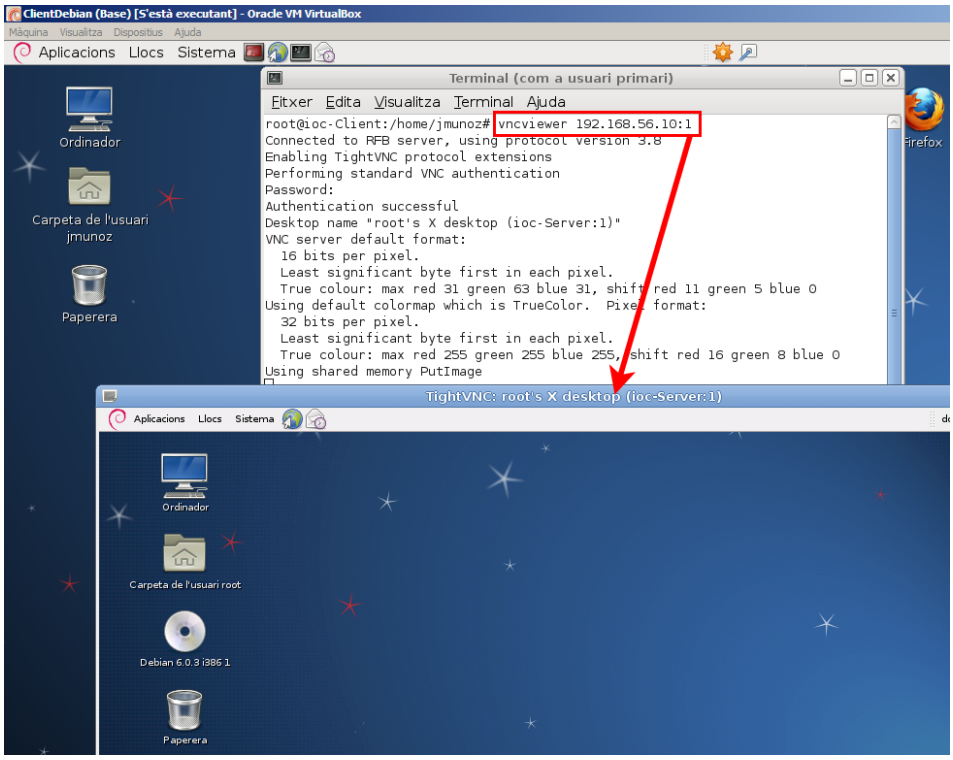
```
# apt-get install xtightvncviewer
```

A continuació, un cop s'està executant el servidor de TightVNC, es pot fer la connexió des del client remot. S'ha d'indicar l'adreça IP del servidor i la pantalla on està funcionant el servidor.

```
# vncviewer 192.168.56.10:1
```

A la figura 2.16 es pot veure com accedim des del terminal remot (ioc-Client) al servidor d'IP 192.168.56.10 i podem veure com s'obre l'escriptori d'aquest servidor (ioc-Server). Fixeu-vos que hem introduït la contrasenya que permet accedir a la sessió. Si hi haguéssim accedit amb la contrasenya de només visualització, no hauríem pogut navegar pels menús del sistema.

FIGURA 2.16. Accés remot mitjançant TightVNC



Accés remot per web i TightVNC

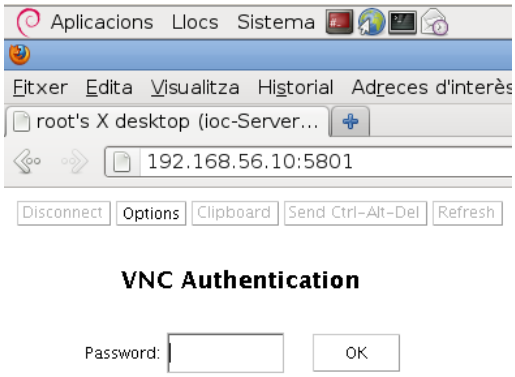
L'accés remot es pot fer també mitjançant un navegador web que sigui compatible amb les miniaplicacions (*applets*) de Java, com Firefox. Per poder accedir d'aquesta manera, el primer que s'ha de fer és instal·lar al servidor el component que permetrà aquest tipus d'accés.

```
1 # apt-get install tightvnc-java
```

A partir d'aquest moment ja es podrà accedir al servidor a través del servidor web i del port 5801. Cal tenir clar que no serà suficient tenir en compte només el component *tightvncjava* instal·lat; serà necessari, a més, tenir un servidor web també instal·lat. Per sort, la major part de distribucions GNU/Linux (inclosa Debian) porten instal·lat de manera predeterminada el servidor web Apache.

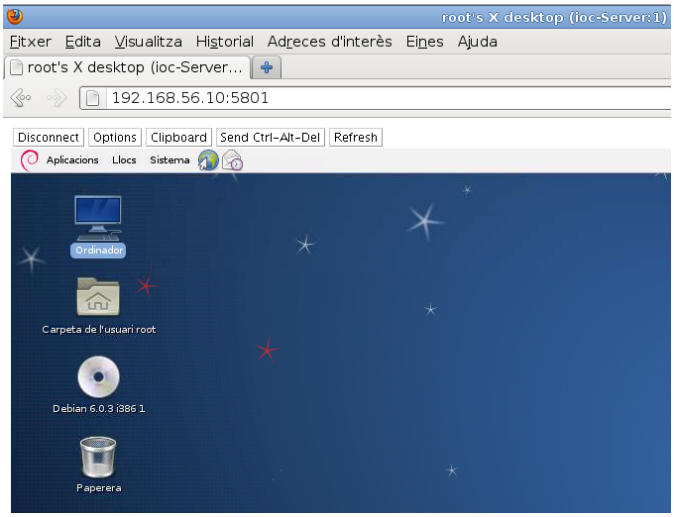
A la figura 2.17 podeu veure que, un cop accedim al servidor mitjançant l'adreça http://192.168.56.10:5801, ens demana una contrasenya d'autenticació VNC.

FIGURA 2.17. Accés remot mitjançamt un navegador web



Si introduïu bé la contrasenya, i segons si és la que permet accés total o de només visualització, s'accedirà, tal com mostra la figura 2.18 a una sessió remota.

FIGURA 2.18. Inici de sessió remota mitjançant un navegador web



Recordeu que el navegador ha de tenir configurat el component necessari per a l'execució de les miniaplicacions de Java (*applets*).

Nous clients d'escriptori remot multiprotocol

La tendència actual és que els clients d'accés a escriptoris remots funcionin amb diferents protocols i que per tant es pugui accedir a diferents plataformes i sistemes operatius. És el cas del projecte Remmina, desenvolupat en GTK+ (i per tant compatible amb els escriptoris GNOME i XFCE), que permet fins i tot mantenir diferents connexions de manera simultània.

Remmina és un client d'escriptori remot que està incorporat per defecte en les darreres versions de Linux Debian i que és compatible amb els protocols RDP (Microsoft Windows), RFB (plataforma VNC), XDMCP (protocol de gestió de finestres X), NX i SSH, entre d'altres.

2.3.5 Gestió remota mitjançant una aplicació gràfica local

Una aproximació diferent a l'administració d'equips remots consisteix a executar una aplicació local amb interfície gràfica que permeti gestionar l'estació remota. El gran avantatge d'aquesta aproximació rau en el fet que, atès que l'usuari interactua amb una aplicació local, el retard en la xarxa no suposa cap problema a l'hora de treballar. A més, com que es tracta d'una aplicació local, la interfície gràfica serà coneguda per l'usuari i no es donarà la situació, desconcertant per a usuaris no tècnics, d'haver de commutar entre la interfície de l'equip local i la de l'equip remot.

Aquesta solució s'utilitza amb molta freqüència per administrar petits equips de xarxa domèstics, als quals el fabricant de l'equip proporciona un programari de gestió remota. El problema d'aquesta solució és que evidentment el programari de gestió s'haurà d'executar de manera nativa en el sistema operatiu de l'estació de l'usuari, que en principi és diferent per a usuaris diferents.

L'ús més adient d'aquest enfocament és emprar un navegador web com a eina d'administració remota. És raonable suposar que, sigui quin sigui el sistema operatiu de l'usuari, aquest disposarà d'un navegador web. En aquest cas el fabricant del dispositiu implementarà una interfície web d'administració que permetrà la gestió del dispositiu per a tots els seus usuaris. Avui dia la major part de dispositius de xarxa, com ara els commutadors, les impressores o els punts d'accés, incorporen una interfície web per a la seva administració.

Fins i tot és possible administrar servidors complets a través del web, fent servir eines com **Webmin**.

Introducció a Webmin

Webmin és una aplicació que permet administrar un servidor Unix/Linux de manera remota mitjançant qualsevol navegador web modern.

Es tracta de programari lliure amb llicència GPL, escrit en Perl i que es pot executar en sistemes operatius com GNU/Linux, OpenSolaris i FreeBSD.

Un cop instal·lat i en funcionament, Webmin proporciona una interfície web (normalment accessible al port 10000) per administrar un gran ventall d'opcions en l'equip i els seus serveis. Es tracta d'una aplicació modular que permet la gestió de diferents àrees en funció dels mòduls instal·lats. En la distribució estàndard s'hi inclouen mòduls que permeten gestionar des del sistema operatiu (usuaris, quotes de disc, processos) fins als serveis de xarxa (Apache, Bind, Samba, etc.). I tot, amb una interfície web molt intuïtiva per a l'usuari.

Perl

És un llenguatge de programació dinàmic, dissenyat per Larry Wall i disponible en moltes

plataformes diferents. Es tracta d'un llenguatge de propòsit general amb el qual es poden implementar tot tipus d'aplicacions. Però la construcció d'aplicacions web dinàmiques i l'administració de sistemes són dues especialitats que no es poden discutir. En molts sentits Python és una alternativa més moderna a Perl, malgrat que no treu cap vigència a Perl. Tots dos són eines excel·lents.

Instal·lació de Webmin

A la pàgina de Webmin (www.webmin.com) es pot descarregar l'aplicació en diferents formats inclòs Debian (*.deb*). Atès que es tracta d'una aplicació escrita en Perl, no hi ha cap dependència d'una arquitectura específica. Només necessitarem tenir instal·lat un intèrpret de Perl amb les seves dependències. Potser la manera més senzilla d'instal·lar-lo és mitjançant l'ordre *aptitude*:

```
1 # aptitude install webmin
```

La instal·lació es fa automàticament al directori /usr/share/webmin i, un cop instal·lat, la interfície web estarà disponible al port TCP 10000. L'usuari administrador de Webmin serà el superusuari (*root*) amb la seva contrasenya.

Webmin pot funcionar mitjançant HTTP i HTTPS; després de la instal·lació inicial, només estarà disponible l'accés mitjançant HTTPS.

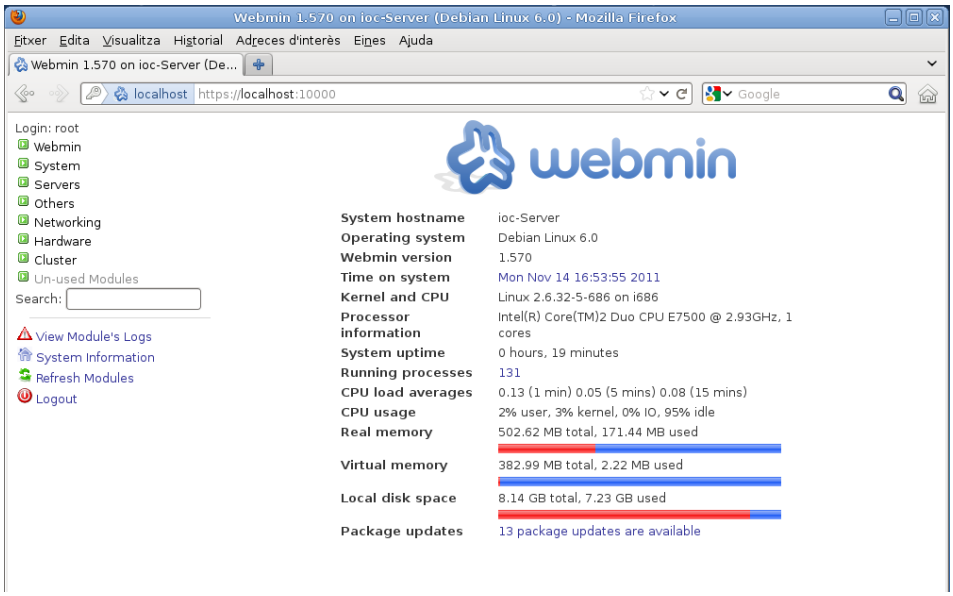
Podem treballar en mode local fent servir Webmin per administrar el propi equip. Per fer això només caldrà obrir un navegador web i dirigir-lo a l'adreça https://localhost:10000 (o bé https://127.0.0.1:10000). Si volem accedir de forma remota des de un altre ordenador connectat en xarxa cal substituir *localhost* per l'adreça IP de l'equip que volem administrar, sempre, lògicament, que aquest tingui iniciat el servei Webmin.

A la figura 2.19 podrem observar la pàgina d'inici de sessió per a Webmin que s'està executant de manera local, després d'haver-nos identificat com a superusuari.

HTTP i HTTPS

La diferència entre aquests dos protocols és que el segon consisteix a fer servir HTTP combinat amb SSL (*secure socket layer*), per tal d'encriptar tota la informació transmesa.

FIGURA 2.19. Pàgina d'entrada a Webmin



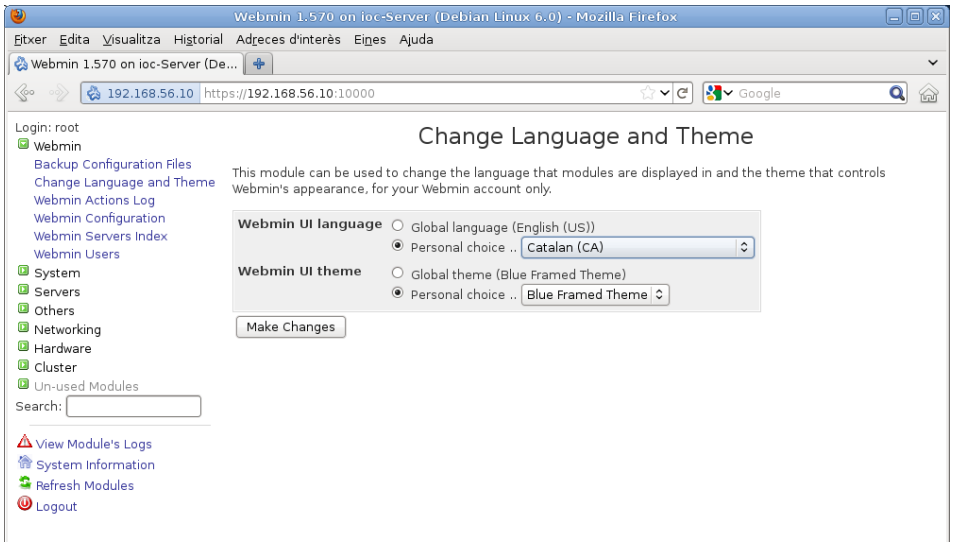
Un cop instal·lat Webmin, el fitxer **/etc/init.d/webmin** ens permetrà engegar i aturar l'aplicació com qualsevol altre servei (directives *start*, *stop*, *restart*).

Funcions de Webmin

Webmin és una aplicació modular que aporta una gran funcionalitat. Les diferents opcions s'exposen mitjançant una interfície clara i intuïtiva disponible en diferents idiomes.

Segurament, una de les primeres accions que fareu dins de Webmin serà configurar l'idioma de la interfície. Aquesta interfície està disponible en molts llenguatges diferents, que es poden seleccionar de manera individual per a cada usuari de l'aplicació o de manera general per a tota l'aplicació. Juntament amb l'idioma es pot seleccionar l'aspecte visual de la interfície. Aquests canvis es poden fer a *Webmin* > *Change Language and Theme*per a l'usuari actual, com es mostra a la figura 2.20.

Figura 2.20. Opcions d'idioma i aspecte per a la interfície de Webmin



Les opcions de la interfície estan agrupades dins de pestanyes generals, que podeu trobar a la taula 2.13.

Taula 2.13. Opcions de Webmin

Pestanya	Funcions
Webmin	Configuració general de l'aplicació: idioma, aspecte, usuaris i fitxers de registre de Webmin. Permet gestionar configuracions exportant un fitxer i tornant-lo a carregar.
Sistema	Administració del sistema informàtic en el qual s'executa Webmin. Des del control d'usuaris, les quotes de disc, les tasques periòdiques i la gestió de paquets fins a les còpies de seguretat.
Servidors	Mòduls per administrar els diferents serveis que es poden executar en l'equip. Alguns del més corrents són: Samba, Postix, OpenSSH, Squid, etc.

TAULA 2.13 (continuació)

Pestanya	Funcions
Altres	Permet gestionar fitxers, comprovar el funcionament dels serveis, instal·lar/desinstal·lar programari i fins i tot establir connexions Telnet/SSH des del mateix navegador al servidor.
Xarxa	Configurar tots els paràmetres de xarxa de l'equip i els seus adaptadors. A més, permet configurar, entre d'altres, el tallafocs o la monitorització dels adaptadors de xarxa.
Maquinari	Permet gestionar tot el maquinari: impressores, particions, RAID, gestor d'engegada GRUB, etc.
Cluster	Permet gestionar de manera unificada un grup d'equips que executen Webmin.

2.4 Tendències actuals de l'accés i administració remota d'equips

El món globalitzat i intercomunicat mitjançant Internet ha fet possible pensar en els programes d'aplicació com a serveis disponibles no en el propi ordinador sinó en el "núvol", entès com a metàfora de la xarxa global.

Quan es parla de computació en el núvol es refereix a l'accés a recursos i serveis des de qualsevol lloc, fer servir programari específic d'aplicació de forma remota mitjançant la interfície del mateix navegador d'Internet i emmagatzemant les dades en servidors externs.

Tot això fa que l'accés i l'administració remota prenguin una nova dimensió on els recursos, programes, potència de càlcul, sistemes d'emmagatzematge, etc. poden estar distribuïts i intercomunicats en qualsevol lloc de la xarxa.

El projecte eyeOS va néixer de la mà d'un grup de joves programadors d'Olesa de Montserrat (Baix Llobregat) l'agost de 2005 i ha obtingut un gran èxit i reconeixement internacional.

Dos exemples que il·lustren aquesta tendència:

- **Chrome Remote Desktop**: Es tracta d'una extensió de Chrome, el navegador de Google que permet accedir a l'escriptori des de qualsevol lloc. L'usuari pot connectar-se i gestionar diferents equips, sigui quina sigui la seva plataforma i el seu sistema operatiu, mentre disposin de navegador Chrome. Per tant, no cal la instal·lació de cap programa addicional.
- **eyeOS** és un escriptori virtual, multiplataforma i de codi lliure que inclou tota la estructura d'un sistema operatiu i algunes aplicacions ofimàtiques. El continguts, arxius i programes que conformen aquest sistema operatiu es troben en servidors privats o públics a la xarxa i es pot accedir als seus recursos des de qualsevol navegador web modern.

3. Administració de servidors d'impressió

L'administració dels serveis d'impressió ha estat sempre una tasca necessària en els processos associats als sistemes d'informació. El resultat dels càlculs i la manipulació de les dades per obtenir informació útil havien de traslladar-se, en algun moment, a un suport definitiu i directament interpretable pels usuaris: el suport de paper. En aquest sentit, l'evolució constant de les necessitats dels usuaris i de la tecnologia ens ha portat a sistemes que han de garantir la gestió de múltiples treballs d'impressió en un entorn multiusuari, la comunicació i transmissió en xarxa amb els protocols adequats, l'administració de cues d'impressió i el suport a una tecnologia d'impressió gràfica dels perifèrics cada vegada més sofisticada.

3.1 Sistemes d'impressió

En l'actualitat els sistemes d'impressió poden ser complexos de gestionar i administrar: treballs d'alta qualitat gràfica, impressores compartides, perifèrics distribuïts a la xarxa, impressores de molts tipus i tecnologies, diferents opcions de ports de connexió, cues d'impressió, prioritats, seguretat, sistemes operatius heterogenis, etc. Tota aquesta gestió fa necessària la implementació de sistemes d'impressió d'una certa complexitat que permetin una adequada administració d'aquest servei bàsic.

3.1.1 Una mica d'història

A l'inici de l'era dels ordinadors personals, la tasca d'impressió era molt simple. Les impressores eren gairebé totes matricials, es comunicaven amb l'ordinador majoritàriament pel port paral·lel i només acceptaven caràcters de text. Així, imprimir un arxiu de text en un sistema Unix es podia fer senzillament redreçant l'arxiu al dispositiu que representava el port d'impressió, generalment lp0:

```
1 $ cat mitext.txt > /dev/lp0
```

Quan els ordenadors IBM PC van aparèixer el 1981, van fer servir el mateix model d'Unix. Les aplicacions incloïen informació sobre els codis de control de les impressores més populars i conegudes i l'usuari indicava a cada aplicació quins codis d'escapada havia de fer servir per a una impressora particular.

Codi d'escapada d'impressora

Els codis d'escapada són determinades seqüències de caràcters que s'insereixen en els textos a imprimir i són interpretats per la impressora com a ordres per activar o desactivar alguna funció, per exemple la lletra negreta o itàlica, fer un salt de línia o de pàgina, etc.

Amb l'aparició dels sistemes operatius gràfics, primer Apple Mac OS i després Microsoft Windows, es va canviar la filosofia d'impressió, ja que es va abstraure la interfície d'impressió de les diferents aplicacions. Així, els programes d'aplicació

només han d'indicar al sistema d'impressió del sistema operatiu on i què imprimir, i és aquest sistema d'impressió el que trasllada la petició correctament formatada a la sortida de la impressora seleccionada.

Aquests sistemes d'impressió havien d'acceptar gràfics, fonts tipogràfiques i imatges, i van aparèixer tecnologies d'impressió com les làser o les d'injecció, juntament amb llenguatges de descripció de pàgines com PostScript, per donar resposta a aquestes noves necessitats gràfiques. També la interconnectivitat i les xarxes van aportar-hi complexitat i la necessitat de protocols per comunicar-se amb impressores i servidors d'impressió remots gestionant prioritats, usuaris i cues de treballs.

En l'actualitat, els sistemes operatius moderns, i entre ells les famílies Unix/Linux, han de resoldre i incorporar les eines d'administració per gestionar les necessitats d'impressió en aquests entorns de treball complexos.

3.1.2 Dades, interfícies i protocols

El procés d'impressió consisteix bàsicament a transmetre una sèrie de **dades** textuals, gràfiques, imatges, etc. des de l'ordinador que les ha processat fins a un determinat perifèric d'impressió. Per establir una comunicació amb qualsevol perifèric cal un canal de comunicació física, es a dir una **interfície** electrònica que permeti transmetre els senyals que porten la informació. Aquests canals de comunicació poden ser ports i bussos de comunicació local o bé una infraestructura de xarxa.

En el cas d'intercomunicar ordinadors i perifèrics mitjançant una xarxa compartida també cal tenir present la necessitat d'implementar uns **protocols** de comunicació adequats per assolir les funcionalitats necessàries en qualsevol sistema d'impressió.

Dades

En els primers sistemes d'impressió, els perifèrics emprats eren impressores de caràcters (*line printer*), que rebien informació alfanumèrica caràcter a caràcter. En aquest entorn, les dades acostumaven a estar codificades en ASCII (*American standard code for information interchange)* de 7 bits, que permetia imprimir els números, els caràcters anglesos i alguns signes de puntuació, a més d'un grapat de caràcters de control que gairebé totes les impressores reconeixien, com ara el salt de línia o el salt de pàgina.

Amb la implementació del codi ASCII de 8 bits es van poder imprimir nous caràcters particulars d'altres idiomes (vocals accentuades, lletra *ñ*, etc.), així con determinats caràcters semigràfics molt senzills.

Per fer el pas següent i poder imprimir dades més complexes com gràfics, imatges, diferents tipografies de lletra, etc. es van crear uns **llenguatges de descripció de**

pàgina. Aquest llenguatges permeten definir el format de la pàgina, la col·locació del text, els mapes de bits i els objectes vectorials per obtenir una qualitat d'impressió òptima.

En aquest cas, l'ordinador envia a la impressora un arxiu amb la descripció de les pàgines a imprimir, aquesta interpreta les instruccions i construeix una representació de la pàgina en forma de mapa de bits que la tecnologia d'impressió (làser, injecció, etc.) s'encarrega de passar al paper.

Els llenguatges de descripció de pàgina més emprats són:

- **PostScript:** Estàndard de facto desenvolupat per Adobe. La majoria d'impressores modernes entenen PostScript i, en tot cas, es poden fer servir filtres i traductors per a la majoria d'impressores.
- **PCL**(*printer command language*)**:**És un llenguatge de descripció de pàgines desenvolupat per Helwett-Packard com a protocol de comunicació per a les seves impressores làser i d'injecció (vegeu la figura 3.1). És una mica més senzill i ràpid, i necessita menys memòria, però no és tan potent com Postcript que, en teoria, pot arribar a generar impressions de més qualitat que PCL.

FIGURA 3.1. Impressora làser Hewlett-Packard amb llenguatge PCL

Interfícies de comunicació

Per transferir dades a qualsevol perifèric en general i a les impressores en particular ens cal una interfície física de comunicació que permeti la transmissió de bits d'informació per mitjans electrònics, òptics o de radiofreqüència.

Aquestes interfícies poden transmetre grups de bits simultàniament, paraula a paraula en el cas de les interfícies paral·leles, o bé bit a bit en el cas de les interfícies de sèrie.

Aquesta connexió física s'anomena genèricament **port**, encara que, en rigor, els ports són interfícies que connecten un sistema informàtic directament amb un determinat perifèric, mentre que els anomenats **bussos** són interfícies de comunicació que poden ser compartides per més d'un perifèric alhora.

Els tipus més habituals d'interfícies de comunicació per a la connexió d'impressores en l'àmbit local són:

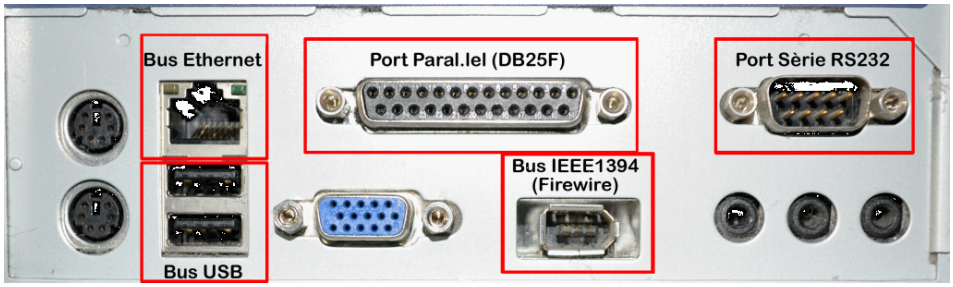
- **Port paral·lel**: també anomenat ***port Centronics***. Implementat en el disseny del primer IBM PC, és un port de comunicació paral·lel bidireccional de 8 bits de dades més un conjunt de línies de control. Aquest port va ser la interfície d'impressió local per excel·lència fins a l'aparició del port USB.
- **Port de sèrie RS232**: interfície de sèrie habitualment implementada amb connectors DB9M (nou pins mascle), dissenyada per a distàncies de fins a 15 metres i velocitats baixes, d'uns 20 Kbps (kilobits per segon). Encara avui dia es fa servir en alguns tipus d'impressores especials de tiquets i de codis de barres.
- **Bus USB** (*universal serial bus*): aquesta interfície de tipus sèrie i topologia de bus permet la connexió simultània de diferents perifèrics que comparteixen el canal de comunicació. Aquest tipus d'interfície permet la connexió de perifèrics en calent (*hot swap*) i el seu reconeixement i configuració automàtica (*plug and play*). En la seva versió 2.0 arriba a una velocitat de transmissió de 480 Mbps mentre que en la versió 3.0 pot aconseguir una velocitat de fins a 3.200 Mbps. En l'actualitat és la interfície més emprada en la connexió de perifèrics de tot tipus (ratolins, teclats, discos externs, etc.), incloses les impressores.
- **Bus FireWire:** aquesta interfície definida en la norma IEEE1394 és coneguda com a **FireWire** en l'entorn d'Apple i com a **i.link** en l'entorn de Sony. Té característiques semblants a l'USB en tant que és una interfície de sèrie de topologia bus, i les seves diferents versions permeten velocitats de transmissió que van dels 400 Mbps als 3.200 Mbps. La diferència principal és que el bus USB requereix sempre un ordinador *host*, mentre que FireWire pot funcionar de punt a punt (*peer-to-peer*), habilitant el diàleg de dos perifèrics entre si, sense la intervenció d'un ordinador. Algunes impressores disposen d'aquesta interfície, encara que el seu ús més habitual el trobem en perifèrics multimèdia, com ara càmeres de vídeo digital.

Plug and play

El sistema *plug and play* (endollar i funcionar) descriu aquells dispositius o interfícies que permeten reconèixer automàticament el maquinari d'un sistema i configurar-ho automàticament, i resolen els possibles conflictes en l'ús dels recursos. Alguns sistemes *plug and play* que es fan servir actualment són les interfícies USB, FireWire o les ranures d'expansió PCI i PCI Express.

Podem veure els connectors d'aquestes interfícies a la figura 3.2.

FIGURA 3.2. Principals interfícies de connexió per a impressores



Fora de l'àmbit de la connexió local i directa, la generalització de les infraestructures en xarxa **Ethernet** ha tingut també com a conseqüència la possibilitat de connectar les impressores directament a la xarxa com un node més. En aquest cas, a més de la connectivitat física Ethernet, usualment amb l'ús de connectors RJ45, s'ha de proveir el perifèric de la corresponent adreça IP i establir protocols de comunicació adients (IPP, SMB, HP Jetdirect, AppleTalk, etc.).

Protocols

Els protocols de comunicació descriuen el format dels missatges a intercanviar entre equips interconnectats per poder establir una comunicació i diàleg efectiu. Aquesta descripció inclou:

- **Sintaxi**: especifica com són i com es construeixen els missatges.
- **Semàntica**: descriu què significa cada ordre i la seva resposta.
- **Sincronització**: defineix la seqüència correcta de peticions i respostes.

Els protocols poden implementar-se per maquinari o per programa, poden incloure autenticació, detecció i correcció d'errors, i són necessaris en els diferents nivells OSI de la xarxa, des del nivell físic fins al nivell d'aplicació.

Els principals protocols de comunicació que es poden fer servir en l'àmbit dels sistemes d'impressió són el següents:

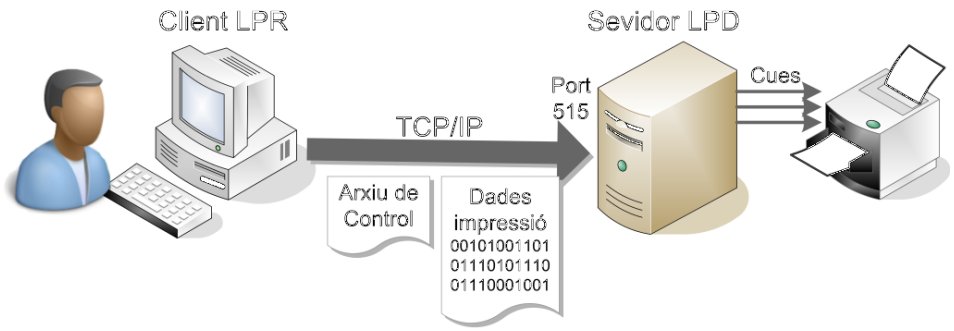
- Protocol LPD/LPR
- Protocol IPP
- AppSocket
- AppleTalk
- SMB/CIF

Protocol LPD/LPR

El Protocol LPD/LPR (*line printer daemon/line printer remote protocol*) és el protocol originalment implementat a la plataforma BDS Unix. Fa servir TCP/IP per establir connexions entre impressores i ordinadors en una xarxa (vegeu la figura 3.3). Aquest protocol treballa normalment escoltant peticions al port TCP 515 i consta de dos components:

- ***Line printer remote*(LPR)** és el terme per al procés client d'enviament de treballs d'impressió a una impressora o cua d'impressió. L'estació de treball que envia la tasca d'impressió és el client LPR.
- ***Line printer daemon*** es refereix al procés de recepció de treballs d'impressió des d'un client LPR. Les impressores o servidors d'impressió que reben els treballs d'impressió s'anomenen *servidors LPD*.

FIGURA 3.3. Impressió mitjançant el protocol LPD/LPR



Quan un usuari envia un document per imprimir, l'ordinador (LPR client) genera unt treball en un determinat format, per exemple, PostScript. Les dades d'impressió es composen d'un fitxer de dades amb el contingut real que es vol imprimir, i un fitxer de control que inclou la descripció de l'arxiu de dades, nom del treball, propietari, nombre de còpies a imprimir, cua de destí, etc. A continuació, el treball d'impressió s'envia a l'adreça IP del servidor LPD, que el rep habitualment a través del port TCP/IP 515. Es poden configurar i definir diverses cues al servidor d'impressió LPD, de manera que el fitxer de control del treball d'impressió ha d'incloure informació sobre quina cua ha de ser assignada.

Protocol IPP

El protocol d'impressió per Internet (IPP o *Internet printing protocol*) defineix extensions del conegut protocol HTTP (*hypertext transport protocol*) per donar suport als serveis d'impressió remots, configuració d'impressores i gestió de cues. A diferència d'altres protocols, IPP permet control d'accés, autenticació i xifrat per donar solucions d'impressió més completes i segures. És el protocol usat de forma nativa per CUPS i acostuma a fer servir el port 631.

HTTP

El protocol de transport d'hipertext (hypertext transport protocol) es fa servir per transferir pàgines web a través d'Internet.

AppSocket

Aquest protocol està basat en el protocol **Jetdirect** de Hewlett-Packard. Acostuma a treballar al port 9100 i es considera molt fiable, senzill i ràpid.

AppleTalk

És un conjunt de protocols de comunicació d'Apple que permet, entre altres coses, la comunicació amb impressores i servidors d'impressió

SMB/CIF

El servidor de blocs de missatges (*server message block*) i la seva evolució, el sistema d'arxius comú d'internet (*common Internet file system*), són protocols de xarxa de la capa d'aplicació del model OSI que permeten gestionar i compartir arxius i impressores entre nodes d'una xarxa. SMB fa servir el port TCP 445 i va ser ideat per IBM, però modificat i perfeccionat per Microsoft, que el fa servir en els seus sistemes operatius Windows. També hi ha implementacions del protocol en codi lliure per a sistemes Linux.

Samba

És la més popular implementació en codi lliure del protocol SMB/CIFS (entre d'altres) i permet la compartició d'arxius i impressores en xarxes heterogènies.

3.1.3 Descripció d'un sistema d'impressió

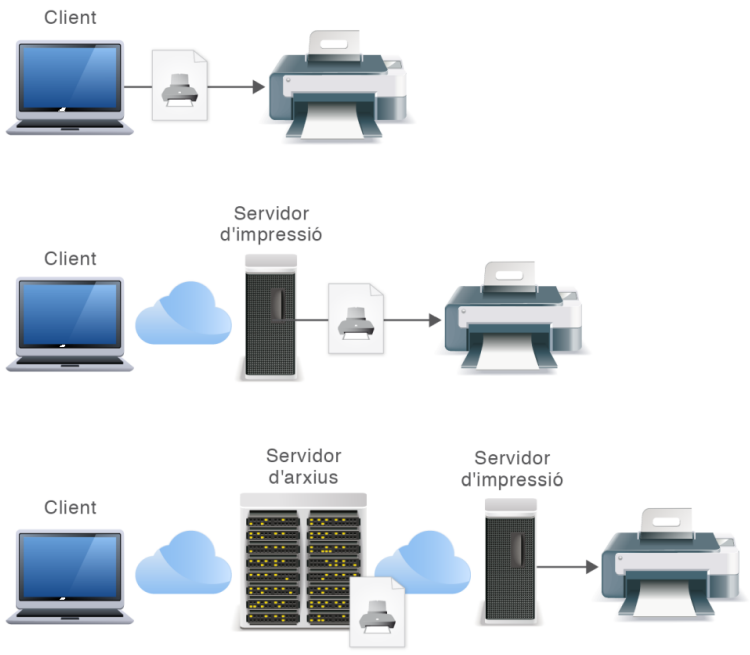
Imprimir pot semblar una tasca senzilla, però comporta una sèrie d'inconvenients i necessitats que els sistemes d'impressió actuals resolen mitjançant la implementació de diferents mòduls:

- El planificador de cues d'impressió (*spooler*)
- Els filtres
- Els controladors d'interfície o *backends*

El primer problema sorgeix perquè la majoria de les impressores no tenen memòria suficient per carregar un document complet. Aquest és un dels motius que fan necessari disposar de **cues d'impressió**i d'un sistema que les gestioni (*spooler*). El gestor de la cua monitoritza la impressora i envia el treball següent en el moment en què queda lliure. La ubicació de les cues d'impressió està determinada per la complexitat del sistema d'impressió. Les cues poden estar situades, tal com es mostra a la figura 3.4, a les localitzacions següents:

- L'ordinador client
- El servidor d'impressió
- Un servidor intermediari d'arxius

Figura 3.4. Sistemes d'impressió i ubicació de les cues d'impressió (spooler) en l'ordinador client, en el servidor d'impressió i en el servidor d'arxius

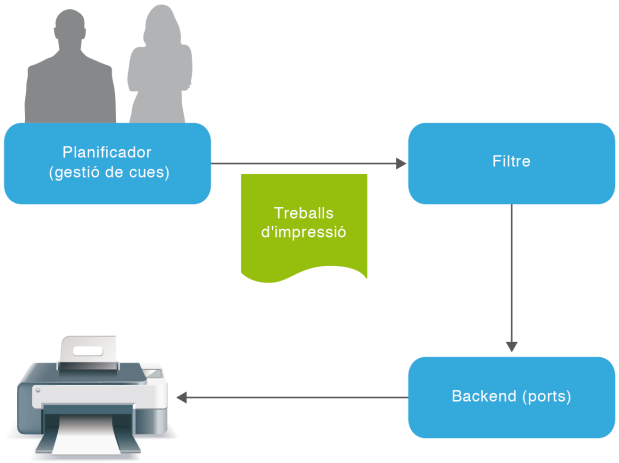


El següent problema a resoldre és el gran nombre de models d'impressora disponibles al mercat que fan servir seqüències de control diferents, és a dir que

parlen llenguatges propis que requereixen una traducció. El **filtres** són les eines, entre el planificador i les impressores, que fan aquesta traducció entre les dades d'entrada i el llenguatge que entén la impressora.

El darrer pas de la cadena és l'enviament de dades al perifèric d'impressió mitjançant algun dels canals i interfícies físiques que s'han descrit. Cadascuna d'aquestes interfícies requereix un controlador de dispositiu o ***backend*** que gestiona i controla el port de comunicació físic (USB, paral·lel, RS232...) o bé la connexió de xarxa (IPP, AppSocket, etc.). El concepte modular del controlador *backend* permet afegir nous tipus d'interfícies segons avanci la tecnologia sense afectar la resta de l'arquitectura.

FIGURA 3.5. Estructura d'un sistema d'impressió



A la figura 3.5 es mostren esquemàticament el mòduls que descriuen un sistema d'impressió modern.

3.1.4 Sistemes d'impressió Unix/Linux

La tasca d'impressió a Unix s'ha realitzat històricament fent servir un d'aquest dos sistemes de gestió d'impressió: el sistema d'impressió d'AT&T (Unix System V) i el dimoni d'impressores en línia (LPD) de Berkeley. Aquests sistemes van estar dissenyats als anys 70 amb l'únic propòsit d'imprimir text en impressores de caràcters. Per aquest motiu han aparegut noves opcions amb més prestacions, com ara LPRng o el sistema CUPS, encara que s'ha intentat sempre conservar la compatibilitat i sintaxi de les ordres de consola originals.

A continuació s'enumeren breument els principals sistemes d'impressió a Unix/-Linux:

/etc/pintcap

Aquest és l'arxiu de configuració del sistema d'impressió *lpd* que conté el registre i definició de totes les impressores del sistema, amb indicació de l'àlies, nom del dispositiu, directori de la cua de treballs, filtres, etc.

- **LPD, sistema d'impressió Berkeley**: aquest és el tradicional gestor d'impressió d'Unix de la plataforma Unix BSD. És controlat pel dimoni d'impressores de línia (*line printer daemon*) situat a /usr/sbin/lpd i fa servir

el protocol LPD/LPR. Els clients es comuniquen amb el dimoni mitjançant el dispositiu /dev/printer i fan servir l'arxiu de configuració /etc/printcap per determinar el directori de la cua de treballs d'impressió.

- **LPRng**: LPR és una implementació en codi lliure que millora el sistema d'impressió BDS però mantenint la compatibilitat, ja que també fa servir el dimoni LPD. Inclou noves funcionalitats en el control de la cua de treballs i les funcions de servidors d'impressió en xarxa.

- **CUPS:** el sistema d'impressió comú d'Unix és un potent i complet gestor d'impressió basat en el protocol IPP (*Internet printing protocol*) i integrant PostScript com el llenguatge de definició de pàgines estàndard. La majoria de distribucions Linux ja incorpora CUPS com a sistema d'impressió per defecte.

3.2 Sistema d'impressió comú d'Unix (CUPS)

CUPS (*common Unix printing system*) és un sistema d'administració i gestió d'impressió portable i extensible per als sistemes operatius de la família Unix/Linux. Va ser desenvolupat per Easy Software Products i alliberat com a programari lliure amb llicència GNU/GPL.

CUPS fa servir el protocol IPP, funciona amb LPD/LPR i proporciona accés a clients Linux, Windows i Mac OS.

3.2.1 Descripció del sistema CUPS

De manera semblant a altres sistemes d'impressió, CUPS està dissenyat al voltant d'un procés de planificació d'impressió (*scheduler*) que distribueix els treballs d'impressió, processa les ordres administratives i proporciona informació i monitorització de l'estat de les impressores i els treballs. Aquest procés correspon al dimoni anomenat **cupsd**, que s'acostuma a carregar en l'arrencada del sistema treballant en segon pla i al que farem referència com a *planificador*.

El planificador o *scheduler* gestiona les peticions que arriben tot seguint el protocol IPP. Aquest treballs poden provenir de l'ordinador local o bé de clients en xarxa de diverses plataformes (Linux, Windows, Mac OS) i generen dos arxius, un amb les dades pròpiament dites, i un arxiu de control amb informació sobre el treball d'impressió. Aquests arxius es guarden al directori `/var/spool/cups`.

Els treballs d'impressió poden incorporar text, gràfics i imatges vectorials o de mapa de bits. Aquests tipus de dades són reconegudes fent servir les entrades de l'arxiu `/etc/cups/mime.types` (o bé `/var/share/cups/mime/mime.types`).

Per descriure aquests elements, CUPS fa servir de forma estàndard el llenguatge de programació de gràfics **PostScript** desenvolupat per Adobe. Els filtres de conversió en funció del tipus de dades estan determinats a l'arxiu `/etc/cups/mime.convs` (o bé a `/var/share/cups/mime/mime.convs`).

El planificador gestiona les impressores disponibles a la xarxa local i els envia els treballs d'impressió fent servir els filtres i controladors d'interfície (*backends*) apropiats, que estan situats a `/usr/lib/cups/backend`.

```
$ ls /usr/lib/cups/backend

behdnssd hpfaxipp parallel serial socket
cups-pdf hp httplpd scsi snmp usb
```

Les impressores compatibles amb PostScript poden interpretar directament les dades rebudes del planificador sense necessitar un filtre específic. En el cas d'impressores que no siguin PostScript cal fer la traducció al llenguatge objectiu de la impressora mitjançant els filtres subministrats o amb altres aplicacions com ara **Ghostscript**, programari també de codi lliure.

Ghostscript

És un programa intèrpret d'arxius PS (PostScript) i PDF (*portable document format*) que permet representar-los en pantalla i traduir-los de manera que puguin ser impresos en una impressora amb capacitat gràfica.

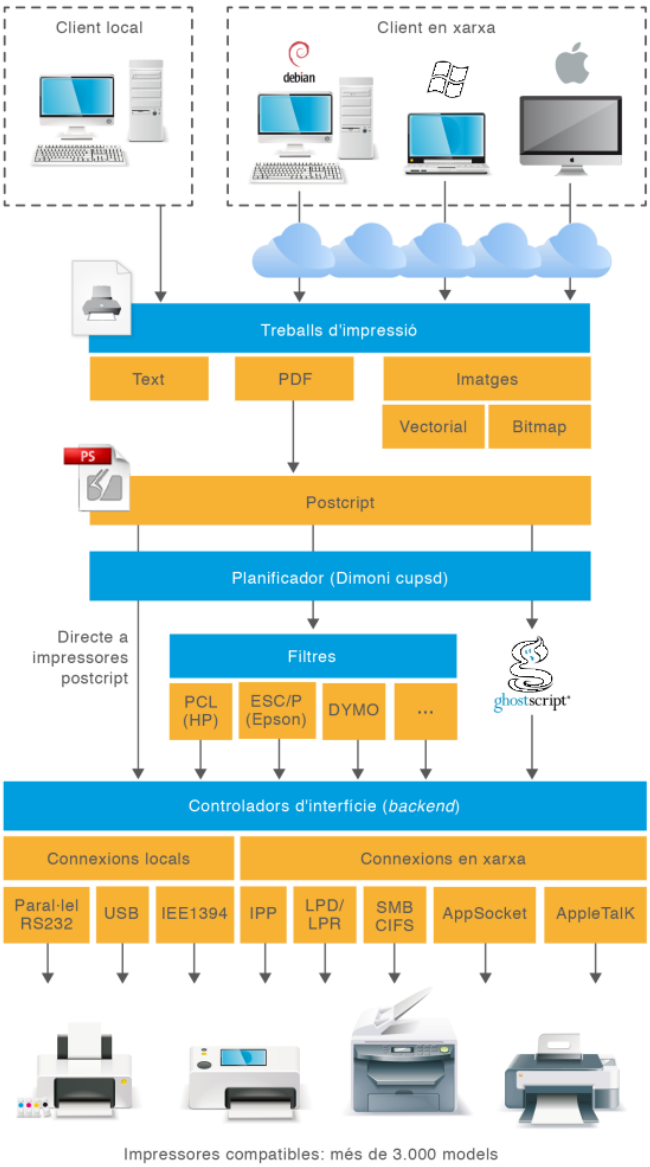
Un cop el treball ha estat transferit a la impressora, el planificador esborra l'arxiu de dades de la cua a `/var/spool/cups`, però deixa l'arxiu de control, on queden numerats correlativament (des del primer, `/var/spool/cups/c00001`).

Altres característiques addicionals proporcionades per CUPS són:

- **Servei de directori:**anomenat *examinador d'impressores*(*printer browsing*). Els clients poden trobar automàticament i fer servir impressores de qualsevol servidor de la xarxa. Si aquesta funció està activada, el servidor distribueix per la xarxa la informació de les impressores disponibles mitjançant missatges de difusió general (*broadcasting*) cada poc temps.
- **Classes**: CUPS també permet definir **classes**, que són grups d'impressores de característiques semblants. Els treballs d'impressió enviats a una classe són redreçats a la primera impressora disponible d'aquella classe. Aquest esquema també permet habilitar seguretat i repartiment de càrregues definint la mateixa impressora en diversos servidors.
- **Suport clients LPD**: fent servir un minidimoni (cups-lpd) que atén totes les tasques d'impressió rebudes per LPD i les redreça al subsistema CUPS tot convertint-les a protocol IPP.
- **Administració web**: aquest servidor també actua com a servidor web per a la documentació, monitorització d'estat i administració del sistema.
- **Impressió des de línia d'ordres**: CUPS entén directament molts tipus diferents d'arxius, incloent text, PostScript, PDF i arxius d'imatge. Això li permet imprimir aquest arxius des de les aplicacions d'usuari o directament des de la línia d'ordres.

La figura 3.6 mostra l'esquema descriptiu del sistema d'impressió CUPS.

FIGURA 3.6. Esquema descriptiu del sistema d'impressió CUPS



3.2.2 Instal·lació i configuració de CUPS

Instal·lació de CUPS

La instal·lació de CUPS no suposa cap complicació. CUPS està incorporat per defecte en la majoria de distribucions però, en qualsevol cas, només cal instal·lar el paquet:

```
1 $ apt-get install cups
```

Per instal·lar el client de CUPS a la plataforma Linux:

```
1 $ apt-get install cups-client
```

En principi CUPS està pensat perquè tant els clients com el servidor funcionin amb el mateix protocol IPP. Si els clients fan servir LPD/LPR o LPRng, cal instal·lar un dimoni de compatibilitat (cups-lpd) disponible amb el paquet:

```
$ apt-get install cups-bds
```

Convé també tenir instal·lats els paquets que contenen els arxius PPD (*PostScript printer description*) que Debian distribueix en dos paquets: openprinting-ppds per a impressores PostScript i foomatic-filters-ppds per a impressores no PostScript.

Tots aquests paquets esmentats solen estar ja instal·lats en la configuració servidor de Debian.

```
$ apt-get install openprinting-ppds foomatic-filters-ppds
```

Un altra utilitat que sol incorporar l'escriptori GNOME és system-config-printer, que permet gestionar el servei CUPS. En cas de necessitar instal·lació:

```
$ apt-get install system-config-printer
```

Finalment, és interessant també instal·lar cups-pdf, una impressora virtual per imprimir documents a un arxiu PDF, ja que resulta molt útil per fer proves d'impressió i conversió de documents a format PDF.

PDF (portable document format)

És un format obert d'emmagatzematge de documents desenvolupat per Adobe que pot incorporar text i elements multimèdia (imatges bitmap i vectorials, so, vídeo, enllaços, etc.). És un format independent del dispositiu i especialment dissenyat per a la visualització i impressió de documents.

```
$ apt-get install cups-pdf
```

Configuració i administració de CUPS

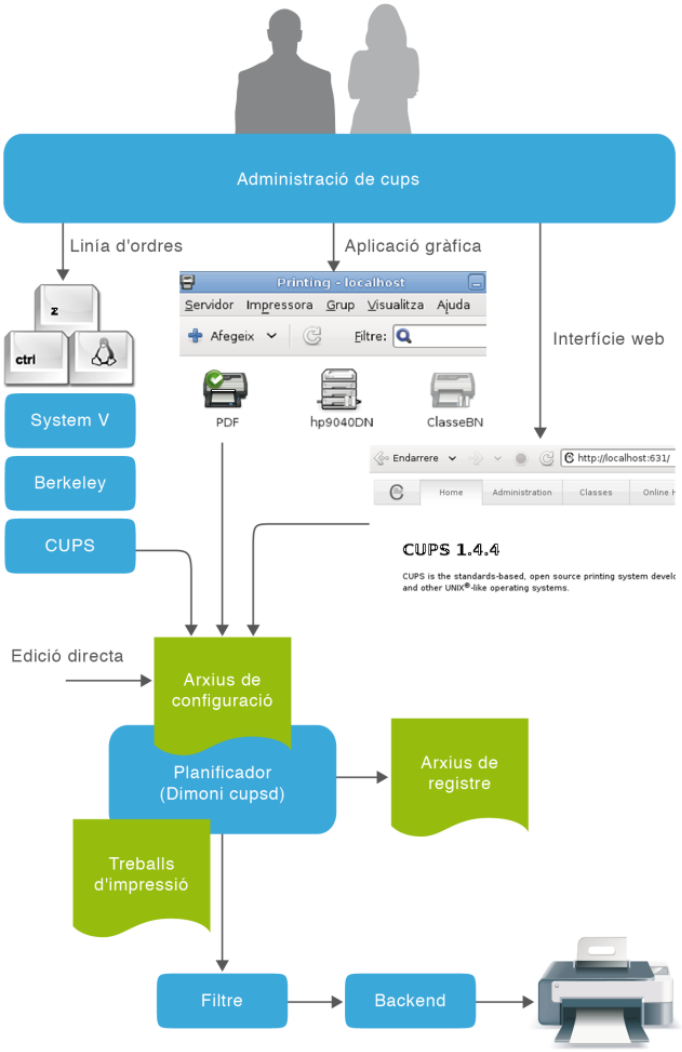
La configuració i administració de CUPS consisteix bàsicament en assignar permisos i accés, definir i configurar impressores, classes i cues d'impressió i gestionar els diferents treballs d'impressió. Aquestes tasques les podem fer de diferents maneres:

- Modificant directament els arxius de configuració
- Mitjançant ordres de consola (SystemV, BDS o pròpies de CUPS)
- Mitjançant una aplicació gràfica nativa (ex. sytem-config-printer)
- Mitjançant la interfície web que proporciona CUPS (http://localhost:631)

CUPS també genera una sèrie d'arxius on enregistra informació rellevant per al control i seguiment del seu funcionament (arxius de registre o *log*).

A la figura 3.7 es mostra un esquema amb les alternatives de configuració i administració de CUPS.

FIGURA 3.7. Alternatives per a l'administració i configuració de CUPS



Arxius de configuració del servidor CUPS

CUPS té un sistema d'administració basat en diferents arxius de configuració que s'enumeren amb detall a continuació:

- **Configuració del servidor:/etc/cups/cupsd.conf** és l'arxiu principal que centralitza la configuració del sistema d'impressió. De característiques semblants a l'arxiu de configuració del servidor Apache, defineix les característiques del servidor CUPS.

- **Definició d'impressores: /etc/cups/printers.conf** conté la llista d'impressores locals definides amb informació de la cua d'impressió associada o de com està connectada i amb quina interfície.

- **Arxius PPD**(*PostScript printer description*): cada cua d'impressió té el seu propi arxiu de configuració situat al directori **/etc/cups/ppd.**Aquests arxius contenen les opcions de configuració de la impressora (mida i orientació del paper, resolució, escala...).

Cues d'impressió

Una impressora pot tenir associada més d'una cua d'impressió. Per exemple, en les impressores en color és útil disposar de dues cues, una per als treballs en color i una altra per als treballs en blanc i negre.

- **Classes d'impressores:/etc/cups/classes.conf**conté la llista de classes d'impressores definides localment.
- **Tipus MIME**: **/etc/cups/mime.types**(o bé /usr/share/cups/mime/-mime.types)indica els tipus d'arxius MIME admesos (text/plain, application/postcript...).
- **Regles de conversió**: **/etc/cups/mime.convs**(o bé /usr/share/cups/mime/-mime.convs) defineix quin o quins filtres estan disponibles per convertir arxius d'un format a un altre. Els filtres estàndard admeten text, arxius PDF, PostScript i molts tipus de formats d'imatge.

Tipus MIME

Els tipus MIME (*multipurpose Internet mail extensions*) són unes convencions que descriuen determinats tipus d'arxius (text, àudio, vídeo, etc.) per facilitar el seu intercanvi per Internet.

Arxius de registre

El planificador manté tres arxius bàsics de registre (*log files*), que normalment es guarden al directori /var/log/cups i que són els següents:

- **access_log**: enregistra totes les peticions HTTP i IPP processades pel planificador de cues. Un exemple d'una entrada d'aquest arxiu:

```
192.168.56.11 - - [31/Jan/2012:01:52:58 +0100] "POST /printers/PDF HTTP/1.1"
    200 873 Print-Job successful-ok
```

- **error_log**: conté tots els missatges d'error del planificador per poder fer una anàlisi i seguiment dels problemes. Exemple:

```
E [30/Jan/2012:23:45:45 +0100] Unable to bind socket for address ::1:631 -
    Cannot assign requested address.
```

- **page_log**: enregistra un llistat de totes les pàgines que s'han imprès.

```
DeskJet root 2 [20/May/1999:19:21:05 +0000] 1 1 acme-123 localhost myjob letter
     one-sided
```

Els arxius de registre són gestionats pel mateix CUPS, que en controla la rotació quan la seva mida sobrepassa el màxim configurat, que per defecte és 1 Mb.

logrotate

En el cas de configurar la mida màxima dels arxius de registre a 0, el planificador deixa d'encarregar-se de la seva rotació i es poden fer servir per al manteniment d'aquests arxius altres utilitats de Linux com ara *logrotate*.

Configuració manual del servidor CUPS

El comportament del servidor CUPS es configura mitjançant les directives contingudes a l'arxiu `/etc/cups/cupsd.conf`. Aquest arxiu de configuració té la mateixa sintaxi que l'arxiu principal de configuració del servidor HTTP d'Apache.

Per exemple, la secció **location** especifica les directives de control d'accés i les opcions d'autenticació per al directori o recurs HTTP especificat. Aquests àmbits d'actuació corresponen a cada un dels directoris que s'enumeren a la taula 3.1 i que tenen correlació amb les diferents seccions de la interfície web d'administració.

Taula 3.1. Descripció dels directoris configurables a la secció location

Directori	Descripció
`/`	Directori general de totes les operacions administratives
`/admin`	Directori de totes les operacions administratives (per exemple afegir i esborrar impressores)
`/admin/conf`	Directori d'accés als arxius de configuració de CUPS (cupsd.conf, client.conf, etc.)
`/admin/log`	Directori d'accés als arxius de registre de CUPS (access_log, error_log, page_log)
`/classes`	Directori de les classes d'impressores
`/classes/name`	Directori d'una determinada classe d'impressores
`/jobs`	Directori dels treballs d'impressió
`/printers`	Directori de les impressores
`/printers/name`	Directori d'una determinada impressora
`/printers/name.ppd`	Directori d'un determinat PPD (Printer description file)

Cada secció location defineix l'accés a un directori determinat i als seus subdirectoris. En l'exemple següent la directiva ***allow*** configura l'accés administratiu a tot el directori arrel de CUPS que, en aquest cas, només es permet des de la màquina local (127.0.0.1).

```
<Location />

Order Deny,Allow
Deny From All

Allow From 127.0.0.1

</Location>
```

En aquest mateix exemple la directiva ***order*** configura el comportament de la resta de directives *allow* i *deny* amb les possibilitats següents:

- **Order Allow,Deny**: permet l'accés a totes les IP excepte aquelles que apareixen a la directiva deny.
- **Ordre Deny,Allow**: només permet l'accés a les IP llistades en directives allow.

No només podem determinar des d'on es pot accedir, sinó el nivell d'autenticació necessari mitjançant la directiva ***require***, que té les opcions següents:

- ***group***: per indicar a continuació els noms del grups autoritzats.
- ***user***: tot indicant a continuació els noms dels usuaris autoritzats o bé dels grups que han d'estar precedits del símbol arrova (@). Els grups poden ser grups predeterminats (@SYSTEM, @OWNER...) o bé grups específics (@NomGrup).

- ***valid-user***: permet l'accés a qualsevol usuari correctament identificat.

L'opció per defecte és que no es requereix autenticació.

En aquest segon exemple, la secció **location** configura la impressió i gestió d'impressores. Les directives ***allow*** permeten accedir només des del propi ordinador (*localhost*) o des d'algun ordinador de la xarxa de classe C 192.168.1.(1-254). Les directives ***require*** permeten la validació de l'usuari Joan, de qualsevol usuari dels grups alumnes i professors i del grup predeterminat de sistema.

```
<Location /printers>
Order Deny,Allow
Deny From All

Allow From 127.0.0.1
Allow From 192.168.1.*

Require group alumnes
Require user joan
Require user @professors
Require user @SYSTEM
</Location>
```

Un altra directiva important és ***listen*** que permet definir les adreces i ports que són escoltats a l'espera d'una connexió IPP. En la definició de la IP es pot fer servir el comodí (*). Per defecte s'accepten connexions de la màquina local pel port 631.

```
Listen 127.0.0.1:631
```

Es presenta un resum de les possibles directives a la taula 3.2.

TAULA 3.2. Resum d'algunes directives de l'arxiu /etc/cups/cupsd.conf

Directiva	Descripció
Location	Defineix una secció per a l'assignació de permisos d'accés i nivell d'autorització a un directori determinat.
Allow / ***Deny***	Dins d'una secció **location** especifica les adreces IP que poden accedir a un directori determinat del servidor.
Require	Defineix el conjunt d'usuaris que poden accedir al servei.
Listen	Indica les adreces/ports escoltats pels servidor.
AccesLog ***ErrorLog*** ***PageLog***	Especifica el nom i adreça absoluta de la ubicació dels arxius de registre.
DefaultLanguage	Especifica el llenguatge per defecte que faran servir les connexions del clients (opcions *de*, *en*, es, *fr*, *it*).
Loglevel	Determina el nivell d'enregistrament de l'arxiu error_log.
MaxClients	Defineix el nombre màxim de clients simultanis.
MaxJobs ***MaxJobsPerPrinter*** ***MaxJobsPerUser***	Permet limitar el nombre màxim de treballs d'impressió totals, per impressora o per usuari.
Browsing On ***Browsing Off***	Activa/desactiva l'enviament de paquets UDP broadcast per informar a la xarxa de les impressores disponibles.

Cal recordar que perquè els canvis a l'arxiu `/etc/cups/cupsd.conf` tinguin efecte és necessari reiniciar el servei:

```
/etc/init.d/cups restart
```

A més de les pàgines de *man,* hi ha una extensa descripció i exemples de totes les directives de /etc/cups/cupsd.conf al manual de referència oficial que trobareu a la secció "Adreces d'interès".

Trobareu la referència de totes les directives de configuració del servidor CUPS a la pàgina del manual associada:

```
man cupsd.conf
```

Configuració dels clients CUPS

Com ja s'ha comentat, CUPS pot treballar amb clients de diferents plataformes i protocols, però disposa del seu propi client amb protocol IPP que es configura mitjançant l'arxiu **/etc/cups/client.conf** o bé **~/.cups/client.conf** segons es vulgui configurar per a tots els usuaris o per a un usuari concret. Aquest arxiu només conté dues directives molt simples:

- ***Encryption***: defineix la configuració de xifrat del client (never-IfRequested-Required-Always). Per defecte és IfRequested.
- ***ServerName***: indica el nom o adreça del servidor que rebrà les sol·licituds del client.

Exemple de contingut de l'arxiu client.conf:

```
Encryption IfRequested
ServerName 192.168.56.10:631
```

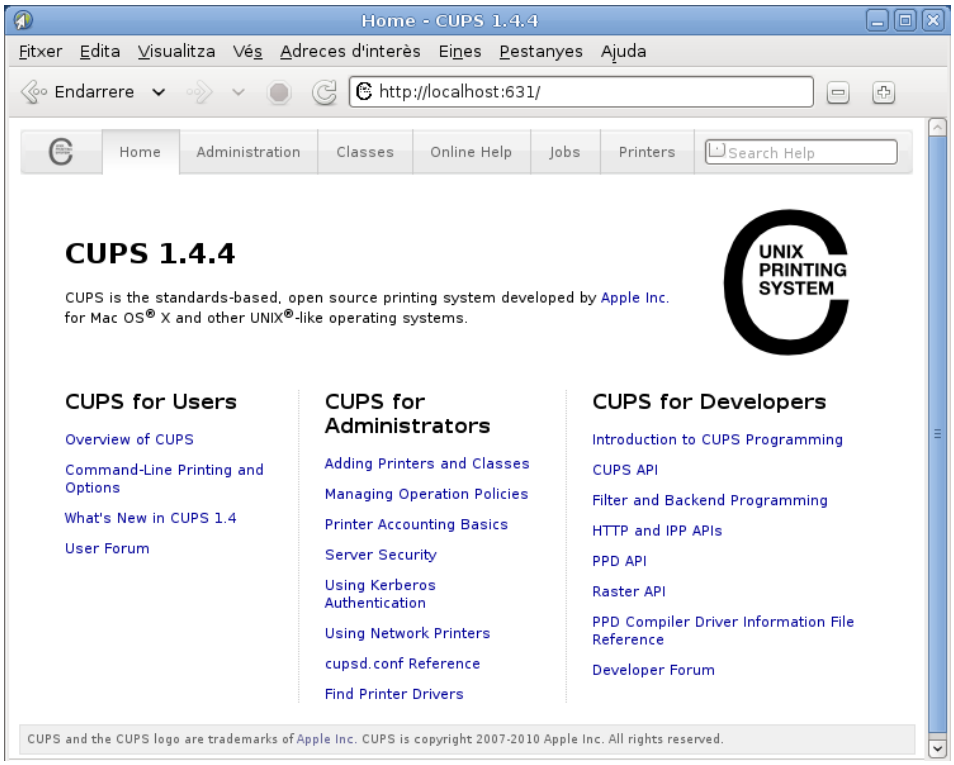
Un altre arxiu de configuració de la part client és **/etc/printcap**, que és genera i actualitza automàticament a partir de l'arxiu /etc/cups/printers.conf. Aquest arxiu és important, ja que el seu contingut és consultat per les aplicacions d'usuari (com per exemple OpenOffice) per generar el llistat d'impressores disponibles en les seves opcions i menús d'impressió.

Administració web de CUPS

El dimoni planificador (*scheduler*) de CUPS és una aplicació de servidor que gestiona peticions HTTP. A més d'atendre les peticions d'impressió rebudes pel protocol IPP, el planificador també actua com un complet servidor web per oferir documentació, monitorització de l'estat de les cues i treballs i administració del sistema d'impressió.

Per defecte s'accedeix a aquest servidor web pel port 631. Així, en la màquina local, només cal obrir qualsevol navegador indicant l'adreça **http://localhost:631** perquè aparegui la pantalla de la figura 3.8.

FIGURA 3.8. Pantalla inicial del servei d'administració web de CUPS



A la primera pàgina hi surten enllaços per arribar a diferents fonts de documentació per a usuaris, administradors i desenvolupadors. La secció que ho resumeix tot és la d'administració (vegeu la figura 3.9), on hi ha enllaços a les altres seccions (impressores, classes i treballs d'impressió) i que permet fer les tasques següents:

Kerberos

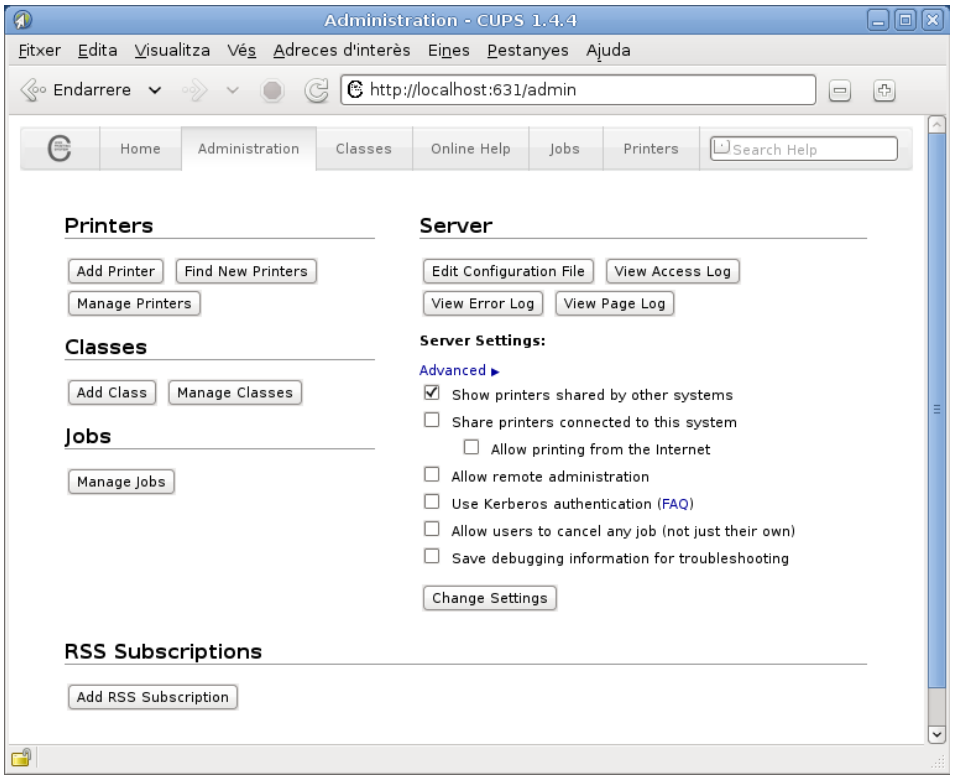
Kerberos és un protocol d'autenticació de xarxes d'ordinadors desenvolupat per l'Institut Tecnològic de Massachusetts (MIT) que permet a dos ordinadors, tant client com servidor, verificar mútuament la seva identitat en una xarxa insegura.

- **Configurar el servidor**: Hi han algunes opcions del servidor que es poden configurar automàticament marcant la casella corresponent.
  - Veure les impressores compartides per altres ordinadors de la xarxa.
  - Compartir les impressores locals.
  - Permetre l'administració des d'ordinadors remots.
  - Fer servir el sistema d'autenticació Kerberos.
  - Permetre als usuaris cancel·lar treballs d'altres usuaris.
  - Configurar en nivell d'enregistrament dels arxius de registre.
- **Edició directa de l'arxiu de configuració**: per ajustar de manera més detallada i completa les opcions de configuració es pot accedir a l'edició directa de les directives contingudes a l'arxiu /etc/cups/cupsd.conf. Un cop fetes les modificacions, el mateix sistema s'encarrega de reiniciar el servei.
- **Visualitzar els arxius de registre**: tant els de accés com els de missatges d'error i la llista de pàgines impreses.
- **Administrar impressores**: gestionar les impressores del sistema, trobar automàticament aquelles que estiguin connectades o bé donar-les d'alta manualment.

- **Administrar classes**: gestionar les classes d'impressores o gestionar les existents.

- **Administrar treball d'impressió**: visualitzar els treballs d'impressió (*jobs*), tant els finalitzats com els que estan en curs que es poden gestionar (cancel·lar, aturar, canviar de cua...).

FIGURA 3.9. Pantalla principal d'administració del servidor

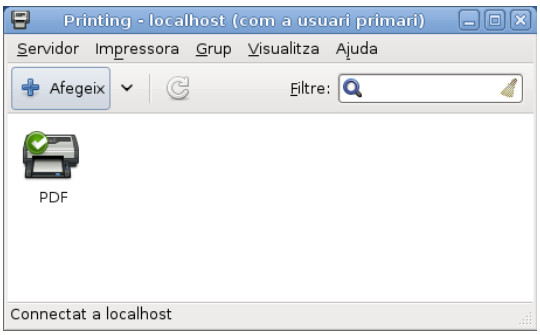


Per poder gestionar CUPS cal ser superusuari o bé usuari amb drets administració d'impressió formant part del grup *lpadmin*.

Administració gràfica del servidor

La majoria de distribucions disposen, a més de l'opció de configuració web, d'alguna eina alternativa per facilitar l'administració gràfica del sistema d'impressió. Una de les més esteses i que està configurada per defecte a la distribució Debian és system-config-printer (vegeu la figura **??**), que es pot iniciar des de consola o bé accedint al menú *Sistema* > *Administració* > *Impressió*.

FIGURA 3.10. Finestra principal del gestor d'impressió system-config-printer



L'eina administrativa system-config-printer ha estat desenvolupada en llenguatge Phyton i permet funcions semblants a l'administrador web, però amb l'avantatge de ser una aplicació nativa. D'altra banda, no està limitada a la configuració de la màquina local, ja que també pot fer servir els protocols IPP i HTTP per comunicar-se amb servidors CUPS remots.

3.2.3 Ordres de consola per a la gestió d'impressores i treballs

CUPS proporciona compatibilitat i emula les ordres de consola tradicionals de les principals plataformes Unix (Unix Berkeley BDS i Unix System V). A la taula **??** es presenten les principals ordres de consola, tot indicant la seva utilitat i de quin sistema provenen originalment.

Unix BDS i Unix System V

Berkeley Software Distribution és una de les plataformes històriques d'Unix, inicialment desenvolupada en aquesta universitat de Califòrnia. Trobem en l'actualitat evolucions i descendents seus en codi lliure com FreeBDS i OpenBDS, així com propietàries com Mac OS X.

Unix System V ésuna altra de les principals plataformes Unix que han evolucionat en implementacions propietàries com Solaris, HP/UX o SCO OpenServer.

TAULA 3.3. Principals ordres de consola reconegudes pel sistema CUPS

Ordre	Sistema	Descripció
lp	SysV	Imprimeix arxius.
lpr	BSD	L'ordre d'usuari per a tasques d'impressió.
lpc	BSD	Control d'impressores i cues (només lectura).
cupsd	CUPS	Dimoni de CUPS.
cupsaddsmb	CUPS	Exporta impressores a Samba per a clients Windows.
lpadmin	SysV	Configura les impressores i classes de CUPS.
lpinfo	CUPS	Mostra els dispositius disponibles.
lpmove	SysV	Mou treballs entre diferents cues d'impressió.
lpq	BSD	Mostra l'estat de la cua d'impressió.
lprm	BSD	Esborra un treball de la cua d'impressió.
cancel	SysV	Cancela treballs d'impressió.
disable	SysV	Atura impressores i classes d'impressora.
enable	SysV	Inicia impressores i classes d'impressora.
lpoptions	CUPS	Mostra o estableix les opcions de la impressora i les opcions per defecte.

TAULA 3.3 (continuació)

Ordre	Sistema	Descripció
lpstat	SysV	Mostra informació de la cua d'impressió.
lppasswd	CUPS	Afegeix, canvia o esborra contrasenyes.
cups-config	CUPS	Obté informació de l'API de CUPS, el compilador i els directoris.

Cal remarcar que CUPS només proporciona compatibilitat amb l'ordre *lpc* d'administració d'Unix BDS en mode de lectura. Així, doncs, per a l'administració per línia d'ordres, és més habitual fer servir l'ordre *lpadmin* de System V.

A continuació es comenten les principals tasques administratives de control d'impressió fent servir ordres de consola.

Generar treballs d'impressió

Ordres relacionades amb la generació de treballs d'impressió:

- Imprimir un arxiu en la impressora per defecte del sistema:

```
$ lp nomArxiu
$ lpr nomArxiu
```

- Imprimir un arxiu en una impressora específica:

```
$ lp -d nomImpressora nomArxiu
$ lpr -P nomImpressora NomArxiu
```

- Impressió de la sortida d'un programa: Les ordres *lp* i *lpr* accepten canonades per imprimir la sortida estàndard de qualsevol altra ordre o programa:

```
$ programa | lp
$ programa | lpr
```

A la taula 3.4 es detallen les principals opcions de les ordres *lp* i *lpr*.

TAULA 3.4. Resum d'opcions de les ordres lp i lpr

Ordre *lp* (System V)	Ordre *lpr* (BSD)	Descripció
-c		Fa una còpia de l'arxiu que està sent imprès.
-m	-m	Envia un correu electrònic en acabar la impressió.
-s		Deixa de mostrar els missatges informatius.

TAULA 3.4 (continuació)

Ordre *lp* (System V)	Ordre *lpr* (BSD)	Descripció
-w		Mostra un missatge a la pantalla en finalitzar la impressió.
-d nomImpres	-P nomImpres	Envia el treball a la impressora "nomImpres".
-n númCòpies	-# númCòpies	Nombre de còpies a imprimir.
-p númPàg		Els números de les pàgines que es volen imprimir en ordre ascendent. Es poden especificar pàgines separades, un rang de números o ambdues opcions.
-q númPrioritat		Assigna una prioritat al treball dins de la cua d'impressió mitjançant un nombre sencer entre 0 (prioritat màxima) i 39 (prioritat mínima).
-t "títol"	-T "títol"	Imprimeix el títol a la pàgina de portada del treball d'impressió.

Llistar impressores disponibles

Ordres relacionades amb el llistat d'impressores disponibles:

- Molts sistemes tenen més d'una impressora a disposició de l'usuari. Per obtenir un llistat de les impressores i classes disponibles:

```
$ lpstat -p
printer ClasseBN disabled since dt 31gen2012 18:57:57 CET
printer hp9040DN is idle. enabled since dt 31gen2012 18:23:36 CET
printer PDF is idle. enabled since dt 31gen2012 01:47:33 CE
```

- Alternativament es pot fer servir l'ordre *lpc*:

```
$ lpc status
```

Assignar impressora per defecte

Per definir quina és la impressora que rebrà els treballs per defecte:

```
$ lpoptions -d nomImpressora
```

Administració de treballs d'impressió

Ordres relacionades amb l'administració de treballs d'impressió:

- Per eliminar un treball d'impressió pendent en la cua.

```
$ cancel id_Treball
$ lprm id_Treball
```

- Cancel·lar tots els treballs d'un usuari determinat:

```
$ cancel -u NomUsuari
$ lprm NomUsuari
```

- Trasllat d'un treball d'impressió. Amb aquesta ordre es mou un treball d'impressió a una altra impressora o classe d'impressores.

```
$ lpmove id_Treball nomImpressora
```

- Podem obtenir l'identificador d'un treball (id_Treball) amb les ordres

```
$ lpq
$ lpstat
```

- Imprimir diverses còpies. Tant l'ordre *lp* com l'*lpr* tenen opcions per a la impressió de més d'una còpia d'un arxiu:

```
$ lp -n num_copies nomArxiu
$ lpr -# num_copies nomArxiu
```

Gestió de classes

Ordres relacionades amb la gestió de les classes d'impressora:

- Afegir una impressora a una classe. En aquest exemple afegim la impressora HP9040 a la classe d'impressores en blanc i negre:

```
$ lpadmin -p hp9040 -c ClasseBN
```

- Retirar una impressora d'una classe. En aquest cas eliminem la impressora virtual PDF de la classe d'impressores en color.

```
$ lpadmin -p PDF -r ClasseColor
```

Si la classe no existeix es crea automàticament en afegir la primera impressora. Si la classe queda buida s'esborra. Es poden comprovar les classes existents a l'arxiu `/etc/cups/classe.conf`.

Habilitar o denegar l'accés a usuaris

Ordres relacionades amb el control d'accés d'usuaris:

- Per determinar el control d'accés dels usuaris a una determinada impressora es fa servir l'ordre *lpadmin* amb l'opció *-u* i a continuació les llistes d'usuaris amb permís (*allow*) o sense permís (*deny*).

```
$ lpadmin -p hp9040DN -u allow:joan,laia
```

- A les llistes es por fer servir l'expressió ***all*** (tots) i ***none*** (ningú). En el següent exemple es permet l'accés a la impressora HP9040DN a tothom menys als usuaris Josep i Anna.

```
$ lpadmin -p hp9040DN -u allow:all deny:josep,anna
```

Imprimir canviant les opcions de la impressora

La configuració per defecte de la impressora pot ser suficient en la majoria d'ocasions. Si cal canviar alguna opció en imprimir un determinat arxiu podem fer servir l'opció *-o* de les ordres *lp* i *lpr*, tal com es mostra en aquest exemple on indiquem orientació de paper, escala i mida del paper:

```
lp -o landscape -o scaling=80 -o media=A4 nomArxiu.jpg
lpr -o landscape -o scaling=70 -o media=A3 nomArxiu.pdf
```

Les opcions d'impressió disponibles varien depenent de la impressora i es poden consultar a l'arxiu PPD corresponent. Les opcions d'impressió estàndard es descriuen breument a la taula 3.5.

TAULA 3.5. Resum de les opcions de configuració estàndard de la impressora

Opció	Descripció
landscape	Indica orientació apaïsada.
media=	Mida del paper. Alguns valors poden ser: *Letter, Legal, A3, A4, A5...* en funció de cada impressora.
sides=	Indica la impressió en les dues cares amb eix al costat curt (*two-sided-short-edge*) o al costat llarg (*two-sided-long-edge*)
page-ranges=	Especifica els números de pàgina (separats per coma) o rangs de pàgines (guionet) a imprimir. Ex. 1-4, 7, 9-12.
page-set=	Imprimeix només les pàgines parells o senars (*odd, even*).
outputorder=	Indica l'ordre d'impressió (*normal, reverse*).
cpi=	Caràcters per polsada (10, 12, 17...).
lpi=	Línies per polsada (6, 8...).
columns=	Formata el text en dues o més columnes.

TAULA 3.5 (continuació)

Opció	Descripció
page-left= **page-right=** **page-top=** **page-bottom=**	Especifica el marge esquerra (*left*), dreta (*right*), superior (*top*) i inferior (*bottom*) mesurat en punts (1 pt = 1/72 polsades).
scaling=	Percentatge entre 1 i 800 de mida en relació a la pàgina.
natural-scaling=	Percentatge entre 1 i 800 en relació a la mida original.
ppi=	Resolució de la imatge (1-1.200) en píxels per polsada.
job-sheets=	Impressió de pàgines de coberta (*none, standard, classified*).
brightness=	Corregeix els valors de brillantor (*brightness*). Per sota de 100 enfosqueix la imatge.
gamma=	Valors de correcció gamma. Per sota de 1.000 enfosqueix la imatge.

Canviar les opcions predeterminades

Totes les opcions anteriors s'apliquen a un treball d'impressió determinat. Si volem que una impressora quedi permanentment configurada amb una opció determinada hem de fer servir l'ordre *lpoptions* amb la sintaxi següent:

```
$ lpoptions –p NomImpressora –o NomOpcio=ValorOpcio
```

Així, en l'exemple següent es deixa com a predeterminda la mida de la pàgina per defecte de la impressora HP9040:

```
$ lpoptions –p hp9040 –o PageSize=Letter
```

Tanmateix, aquest canvi pot tenir efecte per a un conjunt d'usuaris diferent segons qui ha fet l'ordre:

- Si l'ordre ha estat feta per un usuari normal, aquesta configuració es guarda a l'arxiu personal **~/.lpoptions** i només afecta els treballs d'impressió d'aquest usuari.
- Si l'ordre la ha efectuat un usuari administrador, aquesta configuració es guarda a l'arxiu **/etc/cups/lpoptions** i afecta tots els usuaris de la màquina.
- Si la impressora és compartida i volem que aquesta configuració afecti els treballs que enviïn a la cua tots els clients de la xarxa, serà necessari canviar la configuració per defecte de l'arxiu PPD corresponent mitjançant l'ordre ***lpadmin*** i la mateixa sintaxi:

```
$ lpadmin –p hp9040 –o PageSize=Letter
```

Per esborrar un canvi d'opcions predeterminades fet amb *lpoptions* es fa servir la mateixa sintaxi però amb l'opció *-r* (*remove*).

```
$ lpoptions -p hp9040 -r PageSize=Letter
```

Finalment, si volem visualitzar la configuració completa d'una impressora determinada, per exemple:

El símbol * indica el valor actiu en aquest moment, en aquest exemple: mida A4 i resolució de 300 dpi.

```
$ lpoptions p hp9040DN -l

PageSize/Page Size: Custom.WIDTHxHEIGHT 11x14 11x17 13x19 16x20 16x24 2A 4A 8
    x10 8x12 A0 A1 A2 A3 *A4 A5 AnsiA AnsiB AnsiC AnsiD AnsiE ArchA ArchB
    ArchC ArchD ArchE C0 C1 C2 C3 C4 C5 Env10 EnvC5 EnvDL EnvMonarch Executive
     ISOB0 ISOB1 ISOB2 ISOB3 ISOB4 ISOB5 JISB0 JISB1 JISB2 JISB3 JISB4 JISB5
    Ledger Legal Letter RA0 RA1 RA2 RA3 RA4 SRA0 SRA1 SRA2 SRA3 SRA4 SuperA
    SuperB TabloidExtra Tabloid
Resolution/Output Resolution: 150dpi *300dpi 600dpi 1200dpi 2400dpi
```

Integració de sistemes operatius en xarxa lliures i propietaris

Oriol Pérez Lozano

Índex

Introducció

En els darrers anys, el continu canvi tecnològic en què s'ha vist immersa la informàtica ha modificat radicalment la manera com els usuaris intercanvien informació. Des de finals de la dècada dels seixanta fins a ben entrats els anys noranta, les xarxes informàtiques s'utilitzaven gairebé exclusivament en entorns empresarials i acadèmics. A dia d'avui l'escenari és completament diferent i poques són les empreses on les xarxes no hi són presents.

L'evolució dels sistemes operatius reflecteix aquesta tendència, i la majoria d'ells disposen de programari per a la comunicació entre els ordinadors d'una xarxa. En un entorn empresarial, l'accés als recursos s'acostuma a controlar mitjançant un sistema operatiu en xarxa (SOX) instal·lat als servidors. Aquests sistemes operatius, a diferència de la resta, permeten l'administració centralitzada dels recursos de la xarxa (generalment dades, programes i dispositius) i controlen la seguretat de la xarxa, oferint als usuaris la possibilitat de connectar-se o no als diferents recursos. Si no es disposa d'un sistema operatiu en xarxa, els equips no poden compartir els recursos i els usuaris no els poden fer servir.

Actualment, existeix un ventall molt ampli de sistemes operatius en xarxa per a servidors i resultaria pràcticament impossible llistar-los tots aquí. Tot i això, podem fer una distinció entre sistemes operatius en xarxa lliures i propietaris. Entre els SOX lliures més utilitzats podem destacar els sistemes de tipus Unix, com ara les distribucions GNU/Linux (Debian, Ubuntu, Fedora, OpenSuSE, etc.), els sistemes de la família BSD (FreeBSD, OpenBSD) i el sistema Open Solaris (Unix System V/Solaris). D'altra banda, els SOX propietaris amb més força al mercat són Microsoft Windows Server 2000/2003/2008 (Windows), Mac OS X Server (BSD/Unix) i Solaris (Unix System V).

Amb totes les opcions que existeixen, i els avantatges i inconvenients que tenen cadascun d'aquests sistemes, sembla lògic que cada vegada amb més freqüència les empreses no treballin exclusivament amb un tipus de sistema, sinó que apostin per solucions que ofereixen els diferents sistemes operatius. Aquesta unitat centra el seu contingut en explicar com integrar sistemes de diferents famílies, el que anomenem sistemes heterogenis.

Com que en l'àmbit de les estacions de treball Windows és el sistema operatiu més utilitzat i les variants d'Unix (com ara Linux) tenen una elevada quota en el món dels servidors, gran part del contingut de la unitat se centra en la interoperabilitat dels dos sistemes, i en aquest sentit, veureu com el programari **Samba** hi juga un paper important.

En el primer apartat veureu quines són les solucions destinades a la compartició de recursos en diferents famílies de sistemes operatius: Windows, Mac OS X i Unix (Linux). Alhora, es dóna una visió de les funcions que pot realitzar Samba dins d'una xarxa heterogènia com a substitut d'un servidor Windows.

En el segon apartat veureu com integrar dins d'una mateixa xarxa sistemes operatius Windows i servidors Linux amb el servei Samba. En aquest context us podeu trobar situacions molt diferents, i Samba disposa d'un nivell de configuració molt elevat que permet obtenir una bona adaptació a moltes d'aquestes situacions. Tractarem les dues situacions més comunes que us podeu trobar: Samba com a servidor independent en un grup de treball i Samba com a servidor de domini.

Resultats d'aprenentatge

En acabar aquesta unitat, l'alumne:

1. Integra sistemes operatius lliures i propietaris, justificant i garantint la seva interoperabilitat.
    - Estableix nivells de seguretat per controlar l'accés del client als recursos compartits en xarxa.
    - Comprova la connectivitat de la xarxa en un escenari heterogeni.
    - Descriu la funcionalitat dels serveis que permeten compartir recursos en xarxa.
    - Instal·la i configura serveis per compartir recursos en xarxa.
    - Comprova el funcionament dels serveis instal·lats.
    - Treballa en grup per accedir a sistemes de fitxers i impressores en xarxa des d'equips amb diferents sistemes operatius.
    - Documenta la configuració dels serveis instal·lats.

1. Xarxes heterogènies

Una xarxa és un conjunt d'equips interconnectats entre si que permet l'enviament i la recepció de dades. De manera anàloga a quan un conjunt de persones es vol comunicar entre si, en una xarxa els equips poden utilitzar diferents "idiomes" (protocols) per comunicar-se. D'aquesta manera, l'intercanvi de dades és possible només si els dos extrems de la comunicació entenen el protocol utilitzat. En aquest sentit, al llarg dels anys s'han desenvolupat enormes quantitats de protocols, més o menys especialitzats, destinats a aquest intercanvi de dades.

En els darrers anys, i gràcies al procés d'estandardització d'alguns protocols utilitzats a Internet (HTTP, FTP, SMTP, etc.), s'ha aconseguit que dispositius de diferents famílies es poguessin comunicar a través d'Internet sense problemes. Així, doncs, actualment seria impensable imaginar un escenari on per visitar una pàgina web allotjada en un servidor Linux no poguessis fer servir un equip Windows o a l'inrevés.

Com veureu a continuació, en les xarxes locals alguns dels protocols més utilitzats han anat sempre lligats al fabricant dels sistemes, fet que ha dificultat la integració entre diferents sistemes.

1.1 Descripció d'escenaris heterogenis

El model de xarxa determina quin és el rol dels equips que la integren. Aquest depèn en major o menor mesura de les necessitats de l'organització. En aquest aspecte es pot distingir entre:

1. Xarxes igualitàries
2. Xarxes centralitzades

Cada model té els seus avantatges i inconvenients, i per a cada família de sistemes operatius podem trobar diverses implementacions. Històricament, els grans fabricants han ofert solucions adaptades a la seva pròpia família de sistemes operatius i s'han preocupat poc de la compatibilitat amb la resta.

Actualment, però, són poques les xarxes homogènies que es troben en les organitzacions i cada vegada creix més la necessitat d'integrar diferents famílies de sistemes operatius en un mateix entorn.

Per això convé conèixer quines són les solucions que podem trobar en els diferents sistemes operatius.

1.1.1 Xarxes igualitàries

Originalment, les xarxes igualitàries es van dissenyar per permetre compartir recursos des de màquines d'escriptori amb altres usuaris de la xarxa.

En una xarxa igualitària, cada ordinador (*host*) pren el rol de client i servidor, de manera que cadascuna de les màquines determina els recursos que ofereix a la resta i és responsable de la seva pròpia seguretat.

La paraula *host* en aquest context fa referència a qualsevol ordinador connectat a la xarxa.

Els protocols utilitzats en aquestes xarxes també permeten visualitzar i navegar per la llista de recursos compartits sense que hi hagi un servidor central. Els usuaris poden encendre o apagar les seves màquines sense por d'interrompre altres usuaris o serveis de la xarxa (excepte quan estan accedint a recursos propis).

Aquest escenari és el més comú quan la xarxa té un nombre reduït d'ordinadors. En aquests casos també és el més simple de configurar i administrar. Alguns dels avantatges que presenten les xarxes igualitàries respecte a les xarxes centralitzades són:

1. Fiabilitat: la disponibilitat dels recursos no depenen només d'una màquina.
2. Escalabilitat: afegir ordinadors a la xarxa és una tasca simple.
3. Distribució de càrrega: la càrrega de treball es distribueix entre diferents màquines i no recau només en el servidor.
4. Disponibilitat: la caiguda d'una màquina només afecta els recursos compartits per ella.

Xarxes igualitàries en entorns Windows

Històricament, les xarxes igualitàries entre sistemes Windows es coneixen amb el nom de **grup de treball** o *workgroup* i utilitzen els protocols **NetBIOS** i **SMB/CIFS**. El grup de treball s'identifica amb un nom i determina una agrupació d'equips que comparteixen recursos, però en cap cas marca els límits d'una zona de seguretat (és a dir, un membre d'un grup de treball pot accedir a una màquina en un altre grup de treball).

Els equips que comparteixen el mateix nom de grup de treball i pertanyen a la mateixa xarxa apareixen agrupats quan s'explora l'entorn de xarxa. Un *host* Windows, independentment de la versió, ha de ser membre o bé d'un grup de treball o bé d'un domini.

Atès que cada màquina d'un grup de treball és responsable de la seva pròpia seguretat, quan un usuari vol accedir a un recurs compartit cal que tingui un compte d'usuari local a la màquina servidor.

Un recurs dins d'una xarxa s'identifica amb una ruta UNC (*universal naming convention*), que típicament pren la forma `\\NomMaquina\Recurs`. Per exemple, si suposem que dins del nostre grup de treball hi ha una màquina anomenada

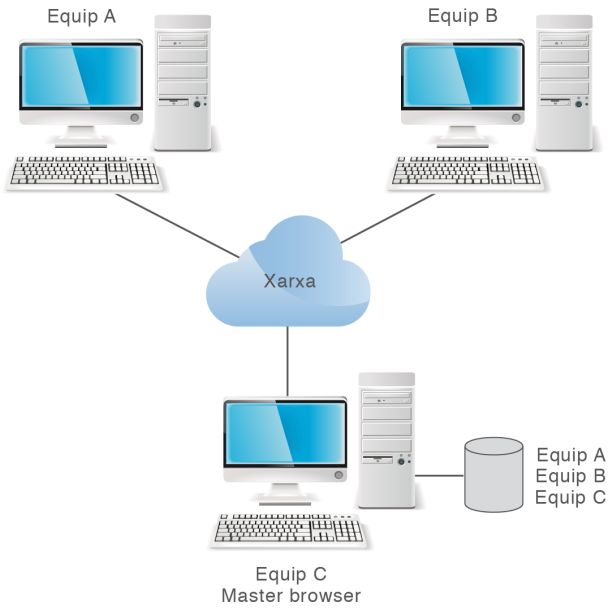
IOC que comparteix una carpeta anomenada *Public*, podrem accedir al recurs amb la UNC `\\IOC\Public`. També podem fer servir l'adreça IP del servidor (`\\192.168.0.6\Public`).

Les adreces UNC no distingeixen majúscules i minúscules.

Una de les característiques més útils dels grups de treball a Windows és la possibilitat d'explorar els recursos compartits d'una xarxa. D'aquesta manera, no és necessari que l'usuari sàpiga l'adreça UNC exacta del recurs al que vol accedir, sinó que pot "navegar" per tots els equips de la xarxa de manera similar a com ho faria en el sistema d'arxius local. Simplement cal que faci clic sobre el botó *Entorn de xarxa* i la llista d'ordinadors apareixerà a la finestra.

Per tal que tots els equips puguin explorar la xarxa d'una manera òptima i sense saturar-la, una de les màquines del grup de treball conté una base de dades amb tots els ordinadors connectats a la xarxa, com podeu observar a la figura 1.1. Aquesta màquina es coneix amb el nom de *master browser*. El *master browser* és responsable d'obtenir tota la informació necessària per crear i mantenir la llista d'ordinadors del grup de treball. La resta d'equips anuncien la seva presencia just en el moment en el qual es connecten al grup de treball i el *master browser* els afegeix a la base de dades. Qualsevol màquina d'un grup de treball és susceptible de ser *master browser* i el procés d'elecció es realitza de forma totalment transparent per als usuaris.

FIGURA 1.1. Master browser en un grup de treball



Per evitar que el *master browser* sigui un punt de fallida en cas que es perdi la connexió, també existeix el rol de *backup browser*. El *backup browser* és una altra màquina amb una còpia de la llista d'ordinadors. Cada cert temps, *master* i *backup browser* sincronitzen les seves bases de dades.

Problemes de sincronització en grups de treball

Si heu fet servir els grups de treball per compartir arxius entre sistemes Windows, probablement alguna vegada haureu vist com dos equips d'una mateixa xarxa mostren

llistes d'equips diferents sense raó aparent. Alguns equips apareixen en una llista però no en l'altra. El motiu principal acostuma a ser que el *master* i *backup browser* encara no s'han sincronitzat.

Els grups de treball funcionen relativament bé mentre hi hagi poques connexions i desconnexions a la xarxa. No passa el mateix quan ens trobem en un escenari amb canvis continus d'ordinadors, amb moltes connexions i desconnexions en intervals curts de temps. El resultat es tradueix en unes bases de dades no actualitzades i no sincronitzades, que comporten una experiència no gaire agradable per l'usuari. Actualment, amb l'auge dels dispositius portàtils, on són freqüents les connexions i desconnexions, aquest mecanisme resulta molt poc pràctic.

A partir de Windows 7 apareix un nou sistema destinat a la compartició de recursos en xarxes igualitàries domèstiques anomenat ***homegroup***. Es tracta d'una funció que simplifica la tasca de compartir recursos.

Microsoft ha publicat l'especificació de *homegroup* com a part del programa Open Specification. Aquest fet permet a qualsevol crear la seva pròpia implementació de *homegroup*, i per tant obre la porta a la implementació lliure en altres sistemes operatius. *Homegroup* fa servir una tecnologia similar a les que fan servir les xarxes P2P, com ara BitTorrent, gràcies a dos protocols:

1. ***Peer-to-peer graphing protocol*(PPGRH):** permet que tots els nodes d'una xarxa tinguin exactament la mateixa base de dades d'ordinadors (desapareix el rol de *Master Browser*).
2. ***Peer name resolution protocol*(PNRP):** permet la resolució de noms.

Aquests protocols, a diferència de NetBIOS i SMB/CIFS, s'han dissenyat de manera que la xarxa es pot expandir per Internet i sense necessitat que cap màquina actuï com a *master browser*. Per tant, s'eliminen els problemes de sincronització. La part negativa és que, de moment, el *homegroup* només està implementat en sistemes operatius Windows 7 i posteriors.

Xarxes igualitàries en entorns Unix/Linux

Tradicionalment, la compartició d'arxius entre màquines Unix s'aconsegueix amb el protocol *network file system* (NFS). L'última versió, NFSv4, incorpora moltes millores respecte a les versions prèvies i és la que es fa servir actualment en totes les distribucions de GNU/Linux.

A Debian podem instal·lar un servidor NFS afegint els paquets `nfs-kernel-server`, `nfs-common` i `portmap`:

```
1 # apt-get install nfs-kernel-server nfs-common portmap
```

La configuració d'NFS al servidor es realitza dins de l'arxiu `/etc/exports`. En aquest s'indica, per a cada línia:

1. Directori que es vol compartir
2. Nom o IP de la màquina que hi tindrà accés

3. Opcions (entre parèntesis i separades per comes)

```
# cat /etc/exports
/directori/compartit 192.168.0.0(ro,root_squash)
/directori/compartit2 192.168.0.0(ro,root_squash)
```

Els clients GNU/Linux poden accedir al servidor muntant el directori compartit tal i com es faria amb qualsevol altre dispositiu, indicant el tipus *nfs* a l'ordre `mount` i l'adreça o el nom del servidor tal i com es mostra a continuació:

```
$ mount -t nfs4 192.168.0.6:/directori/compartit /punt_muntatge
```

NFS controla qui pot muntar els sistemes de fitxers basant-se en la màquina que ho demana i no en l'usuari que farà servir el sistema de fitxers.

Tot i que el protocol NFS s'ha fet servir durant anys, en un món dominat pel sistema operatiu Windows moltes distribucions de GNU/Linux d'escriptori incorporen també Samba com a client-servidor SMB/CIFS per integrar-se a les xarxes Windows.

Xarxes igualitàries en entorns Mac

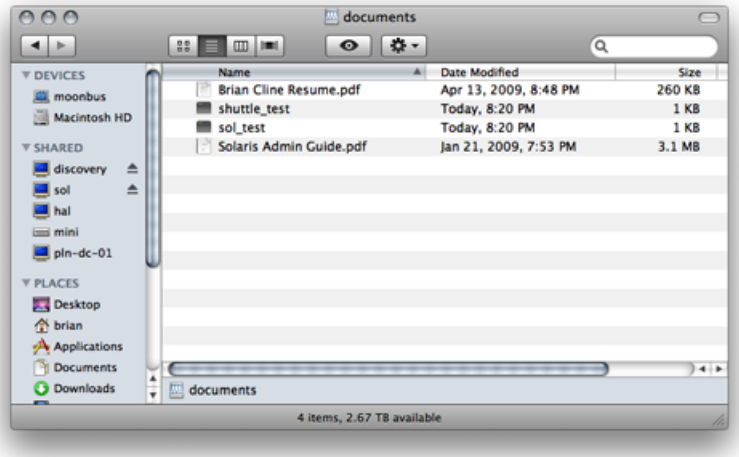
Els equips amb sistemes operatius Mac OS X treballen de forma nativa amb diversos protocols per a la compartició de recursos. En general, un sistema Mac OS X implementa:

1. El protocol *Apple filing protocol* (AFP)
2. Els protocols SMB/CIFS i NetBIOS
3. El protocol NFS

Aquesta característica permet la **compatibilitat amb sistemes operatius Unix/Linux i Windows** sense necessitat d'instal·lar cap software de tercers. Típicament, el protocol AFP s'utilitza quan es comparteixen fitxers entre equips Mac gràcies a que proveeix una interfície totalment adaptada al sistema de fitxers d'Apple (HFS+). L'adreça d'un recurs compartit per un servidor AFP comença per `afp://`. Però també podem connectar-nos a un recurs SMB i NFS fent servir les adreces `smb://` i `nfs://` respectivament.

Un dels avantatges de fer servir Mac OS és que la connexió a diferents tipus de servidors es realitza d'una manera uniforme a través del programa Finder (figura 1.2).

FIGURA 1.2. Finder permet l'accés als recursos compartits amb diferents protocols



L'accés a un recurs compartit per Windows s'especifica amb la forma `smb://servidor/recurs`, on *servidor* és la IP o el nom del servidor i *recurs* és el nom del recurs compartit.

1.1.2 Xarxes centralitzades

El model de xarxa d'igual a igual, o d'entorn de treball, funciona relativament bé quan s'implanta en un entorn amb un grup d'usuaris reduït i amb pocs equips. A mesura que el nombre d'ordinadors connectats a la xarxa va augmentant, l'administració dels recursos, dels usuaris i dels permisos comença a ser una tasca tediosa.

En les xarxes grans, els usuaris i les màquines s'acostumen a agrupar en **dominis**, de manera que l'administració dels recursos, així com els seus permisos, es pot realitzar de manera centralitzada.

> Des del punt de vista de l'administració de sistemes, un domini és un conjunt d'equips interconnectats que comparteixen informació administrativa (usuaris, grups, contrasenyes, etc.).

En aquest escenari, un servidor és el que s'encarrega del control d'accés als recursos, i no pas cada màquina, com és el cas de les xarxes igualitàries, típicament mitjançant un esquema client-servidor. Per exemple, quan un usuari vol iniciar una connexió en qualsevol dels ordinadors (clients) del domini, aquest ordinador haurà de validar les credencials de l'usuari al servidor, i obtenir d'aquest totes les dades necessàries per poder crear el context inicial de treball per a l'usuari.

D'altra banda, un dels desavantatges d'aquest model és que l'accés a la xarxa queda supeditat a la disponibilitat del servidor. És a dir, que si un client no pot contactar amb el servidor tampoc no podrà accedir als recursos. Aquest problema

es pot solucionar disposant un o diversos servidors de reserva (*backup*), que actuen en cas de caiguda del servidor principal.

Dominis en entorns Unix

Històricament, en el món Unix els dominis solien implementar-se mitjançant el conegut *network information service* (**NIS**), del qual n'existeixen actualment múltiples variants per diferents sistemes i entre les quals podem trobar versions per a GNU/Linux. El disseny original de NIS va ser desenvolupat per Sun Microsystems, i es tracta d'un protocol client-servidor que permet l'accés a un servei de directori. La finalitat d'aquest protocol és centralitzar l'administració de sistemes Unix. La implementació inicial de Sun constava d'un servidor, una llibreria per a la part dels clients i diverses eines d'administració del directori.

En general, en un domini NIS es diferencien tres tipus d'ordinadors: servidors mestres, servidors esclaus i clients. Un domini NIS ha de tenir, com a mínim, un servidor mestre. En aquest servidor s'hi emmagatzema la informació referida als usuaris i grups del domini, així com els recursos. Els servidors esclaus permeten alliberar la càrrega del servidor mestre i de cara als clients es comporten de la mateixa manera. Les actualitzacions només es poden realitzar directament al servidor mestre. Una limitació important de NIS és que tots els clients han de d'estar en la mateixa subxarxa IP que el servidor mestre.

El sistema NIS original ha demostrat tenir greus limitacions inherents al seu disseny, especialment en referència a l'escalabilitat i la seguretat. Aquest fet va provocar que el reemplacessin altres tecnologies. Com a contrapartida, i amb la intenció de millorar i corregir els problemes inicials de NIS, Sun Microsystems va introduir **NIS+**. Va afegir-hi fortes millores en seguretat, flexibilitat i escalabilitat. Aquestes millores també van anar lligades a un augment de la complexitat en l'administració d'aquests sistemes, cosa que va provocar, gairebé de forma definitiva, l'abandonament de NIS a favor d'altres tecnologies molt més potents i escalables com ara **LDAP**.

Un domini LDAP s'estructura de manera similar a un domini NIS: hi ha servidors mestres, esclaus i clients. Els clients poden obtenir informació dels mestres o esclaus, però tots els canvis han d'aplicar-se als servidors mestres. Hi ha moltes implementacions d'un servei de directori LDAP, però una de les més utilitzades és **OpenLDAP**.

OpenLDAP és un programari de codi lliure que permet adaptar-se a diverses necessitats. Es tracta d'un servei de directori robust, ràpid i escalable que no té res a envejar a altres serveis de directori. La complexitat d'OpenLDAP serà valorada per aquells administradors que vulgui construir un servei de directori personalitzat.

Dominis en entorns Windows

El servei de controlador de domini en xarxes Windows es va implementar per primera vegada a la versió 3.5 de Windows NT i va millorar posteriorment a Windows NT 4. Aquest servei es va conèixer inicialment com a *NT directory services* (NTDS). Per primera vegada s'oferia la possibilitat d'unificar i centralitzar l'administració dels equips Windows d'una xarxa. Fins aleshores, aquesta tasca s'havia de realitzar equip a equip. Entre molts altres avantatges, es permetia que els usuaris d'una xarxa poguessin accedir als recursos del domini independentment de la màquina que fessin servir.

L'estructura d'un domini Windows NT en una xarxa és molt simple. Dins de cada domini ha d'haver-hi, com a mínim, un controlador primari de domini (CPD) i, de manera opcional, un o més controladors de domini de reserva (BDC). És en aquestes màquines on es troba la informació administrativa i de seguretat del domini. Si el controlador primari deixa d'estar operatiu, els usuaris poden accedir al domini a través d'un dels controladors de reserva (en cas d'existir). Tot i així, les tasques administratives només es poden realitzar directament al controlador primari de domini.

Amb l'aparició del primer sistema operatiu de la família Windows Server (Windows 2000 Server) **es van introduir canvis importants en el servei de controlador de domini**. Aquest, que anteriorment es coneixia com a NTDS, va passar a anomenar-se *Active Directory domain services*. D'aquesta manera, podem distingir dos tipus de dominis Windows:

1. Dominis NT
2. Dominis Active Directory

La diferència més significativa entre dominis NT i active directory radica en que els segons permeten controlar una varietat molt gran de tipus d'objectes dins d'un domini: usuaris, grups, màquines, serveis, i fins i tot definir nous tipus d'objectes (NTDS només permetia definir usuaris i grups). A més, tots els objectes queden **organitzats de forma jeràrquica i accessible mitjançant el protocol LDAP**. Aquesta darrera característica va obrir la porta a la **interoperabilitat** amb aplicacions d'altres plataformes com Unix.

> Windows Server utilitza el protocol LDAP per accedir a la base de dades d'Active Directory, fet que permet la **interoperabilitat** amb aplicacions d'altres plataformes, com per exemple GNU/Linux.

Dominis en entorns Mac

El servei de directori natiu a Mac OS X s'anomena ***open directory***i qualsevol sistema actual d'Apple, sigui versió servidor o d'escriptori, inclou una base de

dades *open directory* local. En aquest directori s'emmagatzema la informació dels comptes d'usuaris locals.

Un ordinador Mac OS X Server pot prendre el rol de controlador de domini *open directory* quan es configura com a *open directory master*. *Open directory* fa servir el protocol LDAP i emmagatzema informació d'administració, usuaris, grups i comptes de màquina de forma jeràrquica. El servei de directori es pot vincular a un servidor de contrasenyes (*open directory password server*) i, opcionalment, a un regne Kerberos, que proveeix un mecanisme d'autenticació segur.

És important destacar que fins a la versió v10.6 els servidors Mac podien simular el rol d'un controlador de domini Windows NT. A partir de llavors aquesta funcionalitat, molt útil en entorns heterogenis, s'ha deixat d'implementar. De tota manera, a Mac OS X es pot fer servir Samba.

Dominis en entorns heterogenis

És cada vegada més habitual que les organitzacions tinguin simultàniament sistemes operatius de diferents famílies. En aquest tipus d'escenaris, la opció més senzilla, i moltes vegades utilitzada, és gestionar els diferents sistemes de forma independent, fent servir les eines administratives que cada fabricant proporciona.

En un entorn com el descrit, una de les solucions que es pot adoptar és crear diferents dominis per a cada tipus de sistema: un domini Unix, un domini Windows, etcètera. Aquesta és una manera simple amb la qual podem administrar, de forma centralitzada, els recursos de xarxa agrupats en famílies. És evident que aquesta solució suposa la repetició de tasques administratives per a cada domini: creació d'usuaris i grups, configuració de permisos, polítiques de seguretat, administració de recursos compartits entre sistemes...

Una opció més elegant, però més complicada, és integrar tots els equips en un mateix domini de manera que s'agrupin tots els sistemes de l'organització en una única administració centralitzada. Lamentablement, aquesta opció no és senzilla, ja que els diferents fabricants de sistemes no acostumen a fer el disseny perquè siguin compatibles entre sí (especialment quan el fabricant és una empresa que defensa un producte comercial). Fins i tot quan els sistemes es basen en tecnologies estàndard (com LDAP en el cas de Windows Server i Linux), aquesta integració no resulta fàcil. Això es degut principalment a les diferències de disseny entre sistemes i al fet que cap dels sistemes acostuma a implementar els estàndards de forma complerta fins a l'últim detall.

Tot i això, administrar un domini en el qual els sistemes no pertanyen a la mateixa família és una tasca que s'ha anat simplificant cada cop més. Si bé al principi era molt difícil integrar sistemes operatius diferents en un mateix domini, en els darrers anys han sorgit moltes solucions que faciliten aquesta integració, tant si es fan servir servidors Windows amb clients GNU/Linux com a l'inrevés.

Entre aquestes solucions podem destacar-ne dues:

1. ***Windows services for UNIX*** (**SFU**), per integrar sistemes Unix amb servidors Microsoft Windows Server.

2. **El paquet de software Samba**, per integrar sistemes Windows amb servidors Unix.

El *Windows services for UNIX* (SFU) o *Subsystem for UNIX-based applications* (SUA) és un paquet de programari produït per Microsoft que proporciona o emula parts d'un sistema Unix dins d'un sistema operatiu Windows NT o successor. La versió SFU 3.5 va passar a anomenar-se SUA (per això es coneix pels dos noms) i està inclosa a Windows Server 2003 R2 i Windows Server 2008. Dins dels sistemes Windows Server 2008 i d'algunes versions de Windows Vista i 7 (Enterprise i Ultimate) s'hi inclou un paquet mínim SUA. Aquest software es pot descarregar de manera gratuïta del lloc web de Microsoft.

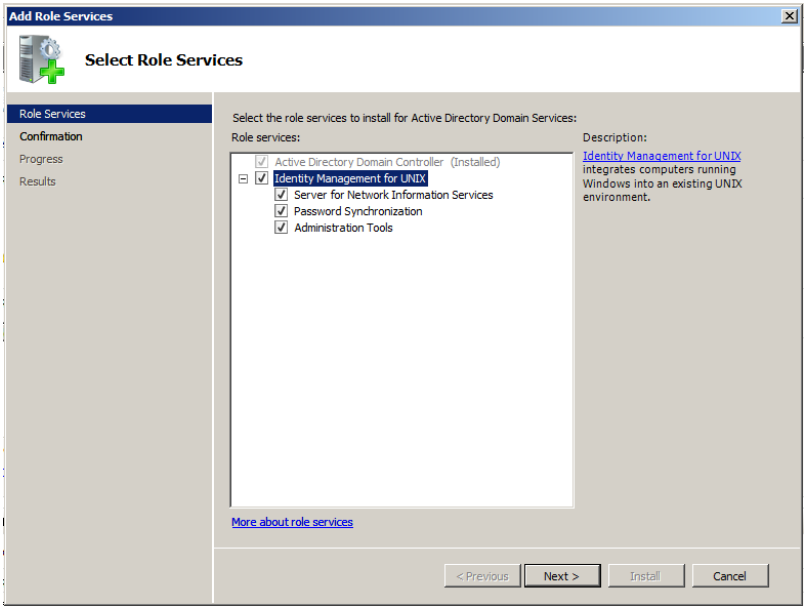
El paquet SUA de Microsoft ofereix moltes funcionalitats, però n'hi ha algunes que són molt útils a l'hora d'integrar sistemes Windows i Unix, com ara:

1. Administració d'identitats per a Unix
2. Client NFS
3. Servidor NFS
4. Servidor NIS

NFS

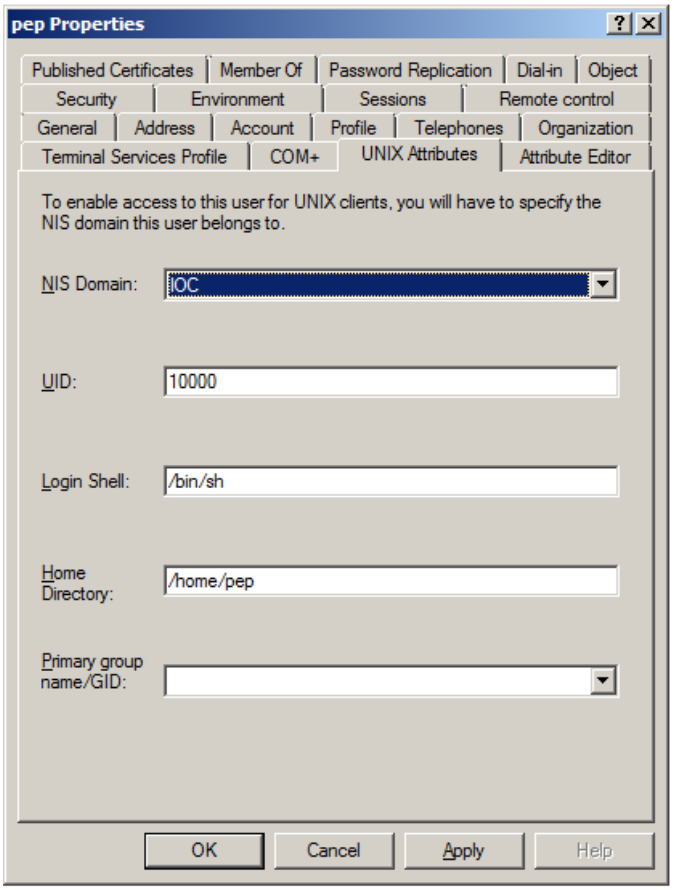
Network file system és un protocol dissenyat per accedir a sistemes d'arxius remots. NFS s'inclou per defecte en tots els sistemes operatius de la família Unix i permet compartir arxius a través de la xarxa.

FIGURA 1.3. Instal·lació del servidor NIS a Windows 2008



L'administració d'identitats per a Unix és un rol d'Active Directory que només es pot instal·lar en els controladors de domini i que **permet assignar atributs Unix als usuaris i grups del directori**. Podem afegir aquesta funcionalitat mitjançant l'eina *Administrador del servidor*, dins dels serveis de rol d'Active Directory (figura 1.3). Un cop instal·lada, podrem indicar per a cada usuari l'UID, el *shell*, el directori *home* i el GID del grup principal, tal com mostra la figura 1.4 (informació que en un sistema Unix es troba dins de l'arxiu `/etc/passwd`).

FIGURA 1.4. Atributs UNIX a Active Directory



Amb l'eina d'administració d'identitats per a Unix, una màquina Windows Server amb Active Directory treballa també com un servidor NIS i permet a les màquines Unix treballar en el mateix domini.

Windows Server pot actuar com un servidor NIS gràcies al paquet de software SUA, fent que els usuaris i grups d'Active Directory tinguin atributs específics de Unix.

Si els paquets SFU i SUA ens permeten afegir sistemes Unix a un domini administrat amb Windows, el paquet de software **Samba** ens permet tot el contrari: afegir sistemes Windows dins d'un domini administrat per màquines GNU/Linux. Samba va començar com un projecte que buscava oferir interoperabilitat als clients de Windows NT 3.x amb un servidor UNIX. Inicialment permetia compartir impressores i arxius entre Windows i Linux, però actualment **Samba pot actuar com un controlador de domini compatible amb Windows NT**. És a dir que podem unir un sistema Windows a un domini gestionat pel servei Samba, tal com ho faríem si el servidor fos Windows. Tot i això Samba té algunes limitacions que veureu més endavant i, per exemple, encara no ofereix la possibilitat de funcionar com un controlador de domini de tipus Active Directory.

1.2 El servei Samba en un escenari heterogeni

Samba és un dels grans èxits del moviment de codi obert. Prova d'això és el reconeixement i l'acceptació que ha tingut darrerament en el món de les TIC, on s'ha convertit en una espècie de servei estàndard en xarxes on conviuen sistemes operatius GNU/Linux i Windows. De fet, gran part de la seva popularitat es deu precisament a la seva capacitat per integrar-se en xarxes Windows. Samba pren el nom del protocol *server message block* (SMB), inicialment desenvolupat per Microsoft, IBM, Intel i d'altres per permetre que els sistemes operatius DOS, Xenix, OS/2 i Windows, al final dels anys 80, poguessin compartir unitats, arxius i impressores. El protocol SMB va canviar de nom l'any 1998 i va passar a anomenar-se *common Internet file system* (CIFS). A dia d'avui, aquest protocol és el que encara es fa servir als sistemes Windows per compartir arxius. L'alternativa en sistemes GNU/Linux és Samba: un paquet de software que pot estar present en situacions molt diverses, des d'un entorn domèstic per compartir arxius i impressores fins a un entorn corporatiu realitzant les tasques d'un controlador de domini.

SMB i CIFS

CIFS es pot considerar una evolució de les moltes que s'han fet del protocol SMB. El canvi de nom va ser part d'una estratègia de Microsoft, però popularment s'ha seguit anomenant SMB. Per això, en molts textos encara podem trobar les sigles SMB o CIFS, indistintament, per referir-se al mateix protocol.

Inicialment es va dissenyar per permetre la compartició de directoris i impressores entre màquines amb diferents sistemes operatius, en concret entre Windows i Unix, però a mesura que passava el temps s'hi han anat afegint més funcionalitats. Samba pot treballar en la majoria de sistemes de tipus Unix, com GNU/Linux, Solaris, AIX, BSD i Mac OS X. Actualment, Samba implementa diversos serveis i protocols que permeten, entre d'altres funcionalitats:

1. La compartició de directoris
2. L'administració i compartició d'impressores, amb controladors associats per accedir des dels clients Windows
3. L'autenticació de clients en un domini Windows
4. L'assistència a la navegació de l'entorn de xarxa (*network browsing*)

Aquestes funcions faciliten que màquines Windows i GNU/Linux **puguin conviure dins d'una mateixa xarxa**. Tant és així, que en algunes situacions és possible substituir un sistema Windows per un sistema GNU/Linux amb Samba sense que els clients se n'adonin. En realitat, un client Windows no sap si la màquina amb la que s'està comunicant és un altre sistema Windows, l'únic que cal és que la màquina de l'altre extrem es comporti com a tal, i en alguns aspectes Samba pot actuar com un servidor Windows.

Molts són els avantatges de l'ús de Samba respecte a un sistema Windows, però en destaquen sobretot:

1. El cost: fer servir els sistemes Windows requereix adquirir-ne prèviament les llicències; en canvi, podem fer servir GNU/Linux i Samba de forma gratuïta.

2. La interoperabilitat entre sistemes: podem fer servir GNU/Linux i Samba per compartir impressores o arxius accessibles des de Windows o Unix.

3. L'adaptabilitat: en tractar-se d'un projecte de codi obert, s'hi poden introduir les modificacions que convinguin.

Tot i això, no podem veure en Samba la solució gratuïta que ens permet substituir definitivament un servidor Windows. Hi ha funcionalitats que encara no estan implementades però que estan en procés, i n'hi ha d'altres que probablement no s'implementaran.

1.2.1 Rols del servei Samba en un escenari heterogeni

Heu de tenir molt clares les funcions que implementa Samba, ja que en alguns escenaris l'ús de Samba no és una bona opció i no queda més remei que fer servir un servidor Windows. El rol de Samba en una xarxa està determinat per la seva configuració, que generalment es farà a través del fitxer `/etc/samba/smb.conf`. La taula 1.1 detalla quins són els rols que compleix (fins a la versió 3.5.6).

TAULA 1.1. Rols que pot prendre Samba

Servidor de fitxers i d'impressores	Sí
Servidor independent (*stand-alone*)	Sí
Controlador de domini de tipus Windows NT	Sí
Membre de domini de Windows NT	Sí
Controlador de domini de tipus Active Directory	No
Membre de domini de tipus Active Directory	Sí
Interacció amb altres controladors de domini Windows	No
Interacció amb altres controladors de domini Samba	Sí
Master browser local	Sí
Master browser de domini	Sí

Protocols propietaris

Alguns dels protocols que fan servir els sistemes Windows són propietaris de Microsoft i no s'han publicat o estan protegits. Aquest és un dels motius que dificulta la tasca dels desenvolupadors de Samba a l'hora d'implementar les mateixes funcionalitats.

A continuació es descriu el significat de cadascun d'aquests rols:

- **Servidor de fitxers i d'impressores**: l'ús més comú de Samba és per compartir fitxers i impressores en escenaris heterogenis.

- **Servidor independent** (***stand-alone***): la forma més simple de funcionament de Samba és aquella en la qual actua com a servidor independent (*stand-alone*). Això significa que no forma part de cap domini i l'accés als recursos compartits és controlat exclusivament per Samba. Aquest rol és el que s'utilitza en escenaris amb xarxes d'igual a igual o grup de treball.

- **Samba com a membre d'un domini**: quan Samba forma part d'un domini, l'accés als recursos que ofereix és controlat pel controlador de domini i no

pas per la pròpia màquina. Samba pot unir-se a un domini NT (Samba o Windows NT) o Active Directory (Windows Server 200x).

- **Samba com a controlador de domini primari**: Samba pot actuar com un controlador primari de domini (PDC) de tipus Windows NT, però no de tipus Active Directory. En aquesta situació, Samba és responsable de l'autenticació dels clients.
- **Samba com a controlador de reserva**: els controladors de reserva (BDC) contenen una còpia de la base de dades d'usuaris i grups del controlador primari de domini, anomenada SAM (*security account manager)*. En cas de caure el PDC, el controlador de reserva s'encarregarà de substituir-lo fins que torni a estar operatiu.
- **Samba com a *local master browser***: en una xarxa d'igual a igual (sense controladors de domini) cada ordinador ha d'anunciar la seva presència a la resta d'equips de l'entorn de treball. Si el nombre d'equips és elevat es pot generar una situació on hi hagi un tràfic continu de paquets per la xarxa. Per evitar el problema, se selecciona una de les màquines del grup perquè contingui la llista de tots els equips connectats. D'aquesta manera, qualsevol una màquina pot anunciar-se i obtenir la llista de tots els equips fent una sola petició. L'equip amb la llista de màquines s'anomena *local master browser* i Samba es pot configurar per actuar com a tal.
- **Samba com a *domain master browser***: de manera similar al *local master browser*, quan ens trobem en un entorn de domini, una de les màquines es fa responsable de la llista d'equips registrats al domini. Aquesta es coneix com a *domain master browser*.

Samba4

Samba4 és el nom d'una versió de Samba en desenvolupament que implementarà un controlador de domini de tipus Active Directory.

Abans de d'aprofundir més en la configuració de Samba **és molt recomanable repassar els conceptes bàsics del seu funcionament**. En els apartats següents tractarem d'explicar alguns dels aspectes que cal conèixer prèviament a l'administració de Samba en un o altre escenari.

1.2.2 Instal·lació del servei Samba

Per instal·lar el servidor Samba en un equip amb el sistema operatiu Debian cal descarregar-se i instal·lar el paquet principal, anomenat **samba**.

```
1 # apt-get install samba
```

Aquest paquet té dues dependències que també s'instal·laran de forma automàtica: `samba-common` i `samba-common-bin`. Tot i que hi ha més paquets relacionats amb samba, com ara `smbclient`, `smbfs`, `swat`, etc., **només el paquet samba i les seves dues dependències són necessàries per establir un servidor**.

Podeu observar quins són els arxius que s'han instal·lat amb el paquet:

```
# dpkg -L samba
```

En general, el servei Samba inclou diverses aplicacions i els següents tres dimonis:

1. **smbd**: procés que rep i atén les peticions que permeten accedir als recursos compartits i autentica els clients.
2. **nmbd**: s'encarrega del registre de noms NetBIOS i participa en el procés d'eleccions a *master browser*.
3. **winbindd**: procés que es comunica amb els controladors de domini per intercanviar informació sobre els comptes d'usuari. Només és necessari quan volem que una màquina GNU/Linux formi part d'un domini Windows. Per defecte no s'inicia.

Un cop descarregat i instal·lat, el servei s'inicia de forma automàtica prenent com a configuració els paràmetres per defecte indicats a l'arxiu `/etc/samba/smb.conf`. Podeu comprovar si els dimonis estan funcionant correctament llistant tots els processos i filtrant pel nom del procés:

```
root@debian:/home/usuari# ps -A | grep nmbd
 2283 ?        00:00:00 nmbd
root@debian:/home/usuari# ps -A | grep smbd
 2287 ?        00:00:00 smbd
 2296 ?        00:00:00 smbd
```

Per aturar, iniciar o reiniciar els dimonis podem fer servir l'eina *service*, que serà necessària quan es canviïn certs aspectes de la configuració.

```
# service samba restart
Stopping Samba daemons: nmbd smbd.
Starting Samba daemons: nmbd smbd.
```

Iniciar o aturar serveis correctament

Fer servir la seqüència service `<nom_script> <paràmetre>` és la millor manera d'iniciar o aturar els serveis, ja que s'executa un script del directori `/etc/init.d/` que ho realitza de manera controlada. No és gens recomanable aturar els serveis directament fent servir l'ordre *kill*.

Opcionalment podem instal·lar un paquet amb documentació molt útil, tant en text pla com en PDF:

```
# apt-get install samba-doc samba-doc-pdf
```

Aquest paquet conté manuals i fitxers de configuració d'exemple que podem trobar dins del directori `/usr/share/doc/samba-doc`. Si disposem d'un navegador podem obrir un fitxer HTML que conté vincles a tota la documentació descarregada.

```
# firefox /usr/share/doc/samba-doc/htmldocs/index.html
```

Un cop comprovat que el servei funciona correctament, cal determinar quin rol ha d'assumir el servei Samba i configurar-lo adequadament segons l'escenari en el qual ens trobem.

Com que el procés de configuració és delicat, us recomanem que quan sigui necessari reviseu els registres (*logs*) dels dimonis **nmbd** i **smbd**. Els podeu consultar dins dels arxius `/var/log/samba/log.nmbd` i `/var/log/samba/log.smbd`

respectivament. És molt útil de cara a descobrir possibles problemes i us ajudarà en el procés d'administració.

Per defecte, Samba mostra només una quantitat molt reduïda de missatges als registres. Quan la informació que veieu al registre no us ajudi a detectar el problema podeu pujar el **nivell de *logging*** per veure missatges més detallats. El nivell dels missatges mostrats es pot canviar amb el paràmetre `log level`, i pot prendre un valor de 0 (menys detall) a 10 (més detall).

```
[global]
     ...
     log level = 3
     ...
```

En general, el nivell 3 és suficient en la majoria d'escenaris. Si feu servir l'ordre *tail* amb la opció *-f* veureu els missatges en temps real a mesura que apareixen:

```
# tail –f /var/log/samba/log.nmbd
```

O bé:

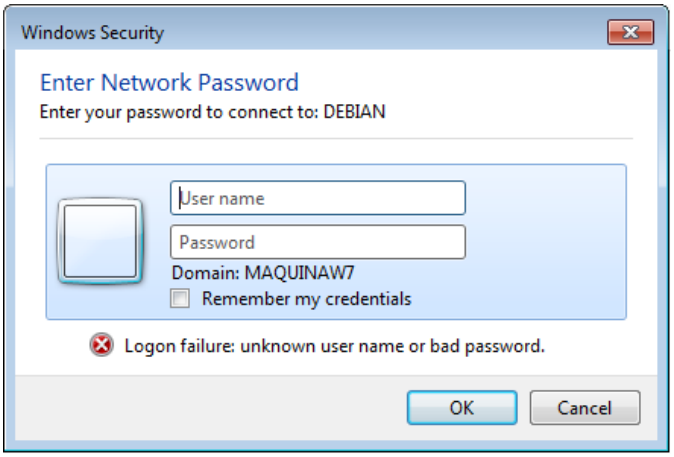
```
# tail –f /var/log/samba/log.smbd
```

1.2.3 Nivells de seguretat

Quan un client vol accedir a un recurs compartit per Samba o per Windows cal comprovar si l'usuari té els privilegis necessaris. Aquesta acció es pot dur a terme de diverses maneres, que estan determinades pel protocol SMB/CIFS. En general, SMB/CIFS defineix dos nivells de seguretat: *user* i *share*. Cadascun d'aquests implica un model d'autenticació per validar les peticions d'entrada. A continuació veiem quines són les diferències entre aquests dos nivells i com es poden configurar a Samba.

1. **Nivell de seguretat *share***. El nivell de seguretat *share* permet assignar una contrasenya a cadascun dels recursos compartits. D'aquesta manera, només aquells usuaris que coneguin la contrasenya hi podran accedir. Aquest sistema té molts inconvenients, el principal és que l'administrador no pot controlar quins usuaris saben la contrasenya. Això és especialment greu en xarxes amb un gran nombre d'usuaris. A més, si es canvia la contrasenya, cal donar-la a conèixer una altra vegada. Actualment, **el nivell *share* es considera obsolet**. Es manté per raons històriques, perquè era el sistema utilitzat en els sistemes Windows 95/98/Me i anteriors. Els desenvolupadors de Samba esperen eliminar aquesta característica en properes versions.
2. **Nivell de seguretat *user***. Quan es configura el nivell de seguretat *user*, l'autorització per accedir als recursos està determinada pels permisos de xarxa de cada usuari. **Quan un usuari (client) vol accedir a un recurs cal que s'autentiqui en el servidor**, típicament mitjançant nom d'usuari i contrasenya (vegeu figura 1.5).

FIGURA 1.5. El nivell de seguretat user requereix que el client s'autentiqui prèviament amb usuari i contrasenya



A Samba, el nivell de seguretat el determina la variable *security* dins de l'apartat `[global]`. Aquesta variable pot prendre cinc valors diferents, un corresponent al nivell de seguretat *share* i els quatre restants corresponents al nivell de seguretat *user*. Aquests valors són: *share*, *server*, *user*, *domain* i *ads*. Les opcions *server* i *share* estan obsoletes, pel que a continuació només es descriu el comportament de *user*, *domain* i *ads*.

Mode de seguretat user

```
1 [global]
2     ...
3     security = user
4     ...
```

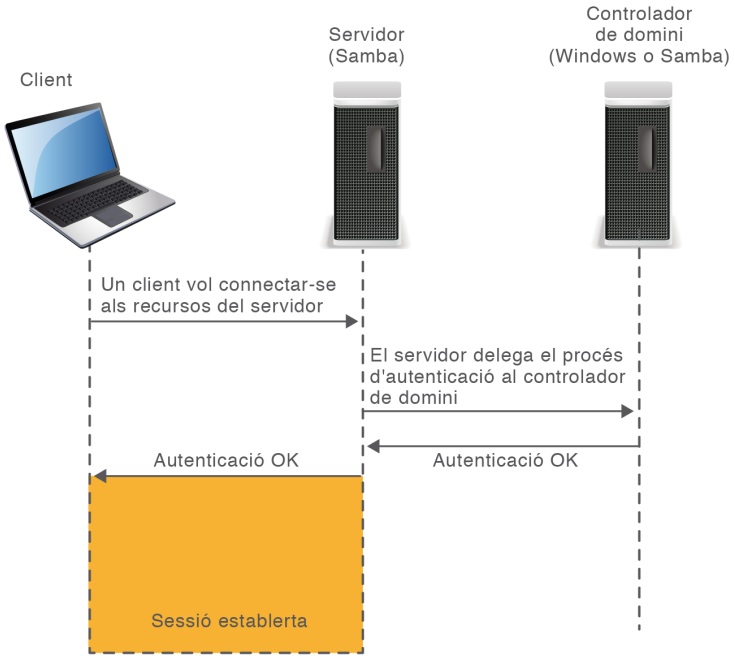
Aquest mode s'utilitza quan Samba actua com un servidor independent (*stand-alone*). Això significa que és el mateix servidor Samba el que s'encarrega de l'autenticació d'usuaris. Per tant, ***user* és el mode utilitzat en escenaris amb xarxes d'igual a igual o entorn de treball**. En aquest cas, Samba normalment disposa de la seva pròpia base de dades d'usuaris, que s'administra de forma local.

Modes de seguretat domain i ads

Quan el servidor Samba i el client es troben dins del mateix domini s'utilitzen els modes de seguretat *domain* i *ads*. D'aquesta manera, quan un usuari vol autenticar-se per accedir a un recurs del servidor Samba (directori o impressora), la petició es redirigeix a un controlador de domini i aquest determina si l'usuari és vàlid. Dit d'una altra manera, la tasca de decidir si l'usuari pot accedir es delega al controlador de domini.

El controlador de domini pot ser una màquina remota, tal i com es mostra a la figura 1.6. Però també pot ser la mateixa màquina amb el servidor Samba si aquest funciona com a CPD.

FIGURA 1.6. Modes de seguretat domain i ads



Amb el modes de seguretat //domain// i //ads//, Samba delega l'autenticació als controladors de domini.

Quina és la diferència entre els modes *domain* i *ads*? **Des del punt de vista del client no n'hi ha cap**: un servidor Samba configurat en el mode de seguretat *ads* es comporta de la mateixa manera que configurat en mode *domain*. La diferència entre aquests dos radica únicament en el procés de comunicació entre Samba i el controlador de domini, on es farà servir un protocol o un altre.

Els dos modes de seguretat, *domain* i *ads*, permeten l'ús del protocol de desafiament/resposta NTLM, que és l'utilitzat en el procés d'autenticació en dominis NT o Active Directory en mode mixt. Però el mode *ads* és l'únic que permet l'ús del protocol Kerberos, que és el que fan servir els controladors de domini Active Directory en mode natiu. En resum, el mode de seguretat *domain* es farà servir quan el controlador de domini sigui de tipus NT o Active Directory en mode mixt (opció activada per defecte a Windows Server). El mode *ads* es pot utilitzar en els mateixos casos que amb *domain* i, a més a més, quan el controlador de domini sigui Active Directory amb autenticació NTLM deshabilitada (mode natiu).

Per configurar el servidor per fer servir el mode de seguretat *domain* cal especificar les opcions següents dins del fitxer de configuració:

```
[global]
     ...
     security = domain
     encrypt passwords = yes
     workgroup = NOM_DEL_DOMINI
     ...
```

Posteriorment cal unir el servidor Samba al domini. Per això farem servir l'ordre `net join`, tot indicant-li un compte d'usuari del domini.

```
# net join -U alumne
```

```
Enter alumne's password:
Joined domain NOM_DEL_DOMINI.
```

Configurar Samba en mode de seguretat *ads*és una mica més complex. En primer lloc, **és necessari que dins de la xarxa hi hagi un servidor DNS**, ja que les llibreries Kerberos el necessiten per resoldre les adreces dels centres de distribució de claus (*key distribution center*). Per tant, cal configurar la màquina amb Samba perquè faci servir aquests servidors (mitjançant el fitxer `/etc/resolv.conf`). A continuació se'n mostra un exemple suposant que el servidor DNS tingui l'adreça 10.0.0.1.

```
# cat /etc/resolv.conf
search NOM_DOMINI.com
nameserver 10.0.0.1
```

Active Directory i DNS

Les llibreries Kerberos necessiten el servei DNS per resoldre les adreces dels centres de distribució de claus (KDC). És per això que quan es promou una màquina Windows Server a controlador de domini (fent servir dcpromo) també s'acostuma a instal·lar el servei de DNS.

A més d'això, és indispensable que la màquina amb Samba tingui instal·lada una llibreria que implementi el protocol Kerberos versió 5. Hi ha diverses opcions, però les més conegudes són **Heimdal** (http://www.h5l.org/) i **krb5** (http://web.mit.edu/kerberos/). Als exemples fem servir krb5, però és vàlida qualsevol llibreria que implementi Kerberos v5. Tal com es mostra, krb5 es pot instal·lar fàcilment a Debian fent servir la seqüència següent:

```
# apt-get install krb5-config krb5-clients krb5-user
```

Aquestes llibreries es poden configurar a través del fitxer `/etc/krb5.conf`. En el cas que ens ocupa cal indicar, com a mínim:

1. **El regne Kerberos**: típicament coincideix amb el nom del domini, i s'especifica a la variable *default_realm*. És imprescindible que s'escrigui en majúscules.
2. **Adreces dels centres de distribució de claus Kerberos**: cal identificar quina o quines màquines dins del regne actuen com a centres de distribució de claus (KDC) Kerberos. Per esbrinar-ho de manera automàtica podem posar la variable *dns_lookup_kdc* a *true*.

```
[libdefaults]
default_realm = NOM_DOMINI.COM
dns_lookup_kdc = true
```

Per verificar que hem fet la configuració correctament, hem de poder connectar-nos fent servir l'eina `kinit` amb un nom d'usuari del domini.

```
# kinit alumne
Password for alumne@NOM_DOMINI.COM:
```

Si tot ha funcionat correctament, s'hauria d'haver obtingut un tiquet Kerberos. Aquest es pot consultar fent servir `klist`.

```
# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: alumne@NOM_DOMINI.COM
```

```
4
5  Valid starting Expires Service principal
6  12/05/11 17:32:18 12/06/11 03:32:19 krbtgt/NOM_DOMINI.COM@NOM_DOMINI.COM
7  renew until 12/06/11 17:32:18
```

Finalment cal modificar el fitxer `smb.conf` de forma similar a com queda amb el mode *domain*, però indicant el regne Kerberos a la variable *realm*.

```
1  [global]
2       ...
3       security = ads
4       encrypt passwords = yes
5       workgroup = NOM_DOMINI
6       realm = NOM_DOMINI.com
7       ...
```

A continuació podeu unir la màquina al domini fent servir l'ordre `net`.

```
1  # net ads join –U alumne
2  Enter alume's password:
3  Using short domain name —– NOM_DOMINI
4  Joined 'DEBIAN' to realm 'NOM_DOMINI.com'
```

Tanmateix, no heu d'oblidar que encara que Samba estigui connectat a un domini, internament Linux assigna permisos basant-se en els usuaris locals de `/etc/passwd`. Per això, en aquesta situació cal també un mecanisme que tradueixi els noms del domini a noms locals.

Així, doncs, encara que el controlador de domini hagi autenticat correctament un usuari que vol accedir als recursos del vostre servidor, cal que Samba sàpiga quin usuari local representa. D'altra manera no tindrà permisos per accedir-hi. Per vincular usuaris del domini a usuaris locals podeu fer servir un dels mecanismes següents:

1. Crear tots els usuaris del domini localment i definir el paràmetre *username map* al fitxer de configuració.
2. De forma totalment transparent utilitzant el dimoni winbind.

winbind

winbind actua com a intermediari entre un controlador de domini Windows i el servei d'autenticació local de Linux. D'aquesta manera, es pot fer que els usuaris del domini siguin també usuaris de Linux.

Podeu trobar força informació al respecte a les pàgines oficials de documentació de Samba.

Quan Samba forma part d'un domini, cal fer servir un procés de traducció d'usuaris del domini a usuaris locals. Aquest procés es pot dur a terme mitjançant *id mapping* o el procés winbind.

Com teniu resumit a la taula 1.2, l'ús del mode de seguretat dependrà de la situació.

TAULA 1.2. Ús dels diferents modes de seguretat en diferents escenaris

	security = user	**security = domain**	**security = ads**
Samba com a servidor independent (xarxa d'igual a igual)	Sí	No	No
Samba com a membre d'un domini NT	No	Sí	No

TAULA 1.2 (continuació)

	security = user	**security = domain**	**security = ads**
Samba com a membre d'un domini d'Active Directory	No	Sí, sempre que Windows Server accepti autenticació via NTLM (mode mixt)	Sí

1.2.4 Backends

Tradicionalment, el sistema GNU/Linux emmagatzema la informació dels comptes d'usuari, de les contrasenyes i dels grups als fitxers `/etc/passwd`, `/etc/shadow` i `/etc/group`, respectivament. De manera anàloga, els sistemes Windows ho fan dins de la base de dades **SAM**, un fitxer que podem trobar dins del directori `\windows\system32\config`. Considerem que aquests fitxers contenen la base de dades d'usuaris del sistema. D'aquesta manera, quan un usuari local s'autentica en el sistema es consulten aquests arxius i es determina si les credencials són correctes o no.

Validació d'usuaris GNU/Linux

Per validar usuaris, GNU/Linux pot fer servir altres mecanismes a més de la consulta al conegut fitxer `/etc/passwd`. Els *pluggable authentication modules* (PAM) són mòduls de software que permeten fer servir altres mecanismes d'autenticació com ara sistemes biomètrics, directoris LDAP, dominis Windows, servidors Kerberos, RADIUS, etc.

De forma similar, el servei Samba implementa un mecanisme d'autenticació d'usuaris per accedir als recursos que ofereix. Així, quan un client vol connectar-se a un recurs compartit per Samba, com ara un directori o una impressora, s'envien les credencials i Samba verifica que l'usuari estigui donat d'alta. Si les credencials rebudes són correctes, es fa el pas següent: determinar si aquest usuari té permisos o no per realitzar l'acció sol·licitada. Es fa evident, doncs, que Samba ha de disposar d'una base de dades pròpia on emmagatzemar els comptes d'usuari. Sovint, a aquesta base de dades se la coneix amb el nom en anglès de ***backend***.

Alguns usuaris, sobretot els acostumats a treballar amb Microsoft Windows, potser es plantejaran la pregunta següent: per quin motiu Samba fa servir una base de dades diferent de la del sistema GNU/Linux? No resultaria més fàcil que els mateixos usuaris del sistema, donats d'alta a `/etc/passwd`, també hi tinguessin accés com a clients Samba? La resposta és que Samba necessita emmagatzemar atributs addicionals per a cada usuari que no es poden trobar a `/etc/passwd`. Exemples d'aquests atributs són: el SID de l'usuari, les contrasenyes xifrades en NT/LM o l'adreça UNC del directori *home* (per citar-ne alguns).

> Per a cada usuari, Samba necessita emmagatzemar atributs addicionals que no pot trobar dins de `/etc/passwd`. És per això que ha de disposar d'una base de dades independent de la del sistema.

Podem especificar el *backend* que Samba farà servir a l'atribut *passdb backend*, dins de la secció `[global]` del fitxer de configuració. De manera nativa, Samba permet utilitzar tres tipus de *backends* diferents:

1. Fitxer de text pla
2. Fitxer TDB

3. Directori LDAP

Tot i això, mitjançant mòduls addicionals podem utilitzar altres *backends* com bases de dades MySQL, PostgreSQL, fitxers XML, etcètera. A continuació es detallen les característiques dels tres *backends* utilitzats per Samba.

Fitxer de text pla

És el mètode més simple. Les dades dels usuaris queden registrades en un fitxer de text, normalment a `/etc/samba/smbpasswd`. Aquest fitxer el podem visualitzar amb un editor de text.

```
[global]
     ...
     security = user
     passdb backend = smbpasswd
     encrypt passwords = yes
     ...
```

La ubicació del fitxer es pot definir manualment:

```
passdb backend = smbpasswd:/ubicacio/fitxer/smbpasswd
```

El fitxer `smbpasswd` conté una línia per a cada usuari Samba, i a cadascuna d'aquestes, els seus atributs separats per dos punts ":".

```
usuari:uid:hash_1:hash_2:flags:darrera_modificació
```

On:

1. *usuari*: és el nom d'usuari que faran servir els clients per connectar-se als recursos.
2. *uid*: l'UID del compte d'usuari al sistema.
3. *hash_1*: *hash* de la contrasenya de tipus LanMan (LM).
4. *hash_2*: *hash* de la contrasenya de tipus NT/LM.
5. *flags*: diversos caràcters que representen el tipus i l'estat del compte.
6. *darrera_modificació*: mostra quan va ser l'última vegada que es va canviar la contrasenya (LCT o *last change time*), en format hexadecimal.

```
# cat /etc/samba/smbpasswd
usuari:1000:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX:7CE21F17C0AEE7FB9CEBA532D0546AD6:[
    U ]:LCT-4EBA44DE:
lluis:1001:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX:7CE21F17C0AEE7FB9CEBA532D0546AD6:[U
     ]:LCT-4EBA6D0C:
manel:1003:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX:588FEB889288FB953B5F094D47D1565C:[U
     ]:LCT-4EBA6D15:
marta:1005:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX:05B073DAA9C1B3B909FF5AE2E4604BB5:[U
     ]:LCT-4EBA6D1C:
silvia:1004:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX:DF54DE3F3438343202C1DD523D0265BE:[
    U ]:LCT-4EBA6D22:
pep:1002:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX:37D02806094AD4CE5AEDD139D9943BED:[U
    ]:LCT-4EBA6D39:
```

Hash

Les contrasenyes dels usuaris no s'emmagatzemen mai de manera llegible, sinó que s'hi aplica un mecanisme de xifrat per evitar que ningú, ni tan sols l'administrador, les pugui llegir. Al resultat d'aplicar un mecanisme de xifrat a una contrasenya se'l coneix com a hash.

Al llarg de la seva història, Windows, i de retruc Samba, ha fet servir dos mecanismes de xifrat per emmagatzemar les contrasenyes dels seus usuaris, LM i NT/LM. El primer, LM, ha quedat en desús per problemes de debilitat. NT/LM corregeix algunes d'aquestes debilitats i és el mètode utilitzat en les darreres versions. Windows XP utilitza els dos mecanismes alhora (per compatibilitat amb versions anteriors), però a partir de Windows Vista el mecanisme de xifrat per defecte és NT/LM.

Actualment, Samba tampoc fa servir el mecanisme LM. És per això que als backends hi podem observar una filera de X on hi hauria d'haver el hash LM.

L'ús del *backend* smbpasswd és ideal per a escenaris simples amb pocs usuaris, però no és recomanable quan el nombre d'usuaris és elevat. La raó és que no es pot accedir al fitxer de manera concurrent, és a dir que quan diversos clients volen autenticar-se a la vegada, Samba atendrà les peticions de manera seqüencial, una darrera de l'altra, de manera que pot provocar un coll d'ampolla a la xarxa. Un altre inconvenient és que no hi ha lloc per a atributs addicionals com ara la data d'expiració de la contrasenya o l'adreça UNC del directori home (necessària per aplicar perfils d'usuari).

Fitxer TDB

El *backend* TDB (*trivial database*) o `tdbsam` utilitza un fitxer binari per emmagatzemar les dades dels usuaris. Com a tal, no es pot manipular ni visualitzar amb un editor de text, sinó que hem d'utilitzar les eines que proporciona el paquet de Samba.

```
[global]
     ...
     security = user
     passdb backend = tdbsam
     encrypt passwords = yes
     ...
```

Per defecte, aquest fitxer està situat en un fitxer anomenat `passdb.tdb` al directori `/var/lib/samba/`, tot i que sempre dependrà de la distribució que estem utilitzant. La ubicació del fitxer es pot definir manualment:

```
passdb backend = tdbsam:/ubicacio/fitxer/passdb.tdb
```

Té alguns avantatges respecte al *backend* `smbpasswd`: disposa de **més atributs per usuari** i permet l'**accés concurrent** al fitxer, fet que millora el rendiment en cas de múltiples peticions d'autenticació. La part negativa és que **no es pot fer servir** en un escenari amb múltiples controladors de domini. El motiu és que quan hi ha diversos controladors de domini es necessita algun mètode de replicació del *backend*. Segons els desenvolupadors de Samba, aquesta configuració és la recomanada sempre i quan el nombre d'usuaris no sigui superior a 250. Aquest *backend* el trobem seleccionat per defecte en la majoria de distribucions de GNU/Linux, com ara Ubuntu, Debian o Fedora.

El *backend* `tdbsam` està configurat per defecte a la majoria de distribucions GNU/Linux actuals, com ara Ubuntu, Debian o Fedora.

Directori LDAP

El *backend* `ldapsam` ens permet utilitzar un servei de directori LDAP per emmagatzemar els comptes d'usuari Samba. LDAP ens permet mantenir una base de dades d'usuaris centralitzada i accessible per qualsevol controlador de domini. Per configurar-lo només cal indicar la URL del servei LDAP dins del fitxer de configuració, tal com mostra l'exemple.

```
[global]
     ...
     security = user
     passdb backend = ldapsam:ldap://servidor/
     ...
```

Per utilitzar un directori LDAP com a *backend* no n'hi ha prou d'instal·lar i iniciar el servei de directori, sinó que prèviament cal preparar-lo. A grans trets, preparar el directori vol dir:

1. Carregar els esquemes propis de Samba al directori LDAP: és necessari que els elements del directori, com ara usuaris o grups tinguin els atributs específics de Samba.
2. Configurar Samba per fer servir el directori: cal que Samba sàpiga on trobar els elements dins del directori. Recordeu que el directori pot emmagatzemar moltes més dades.

Hi ha situacions en les quals és convenient disposar de més d'una còpia del servidor LDAP. D'aquesta manera ens assegurem que en cas de fallida d'un d'ells hi pugui haver una alternativa. En aquest cas es poden indicar les adreces de tots els servidors en el fitxer de configuració:

```
passdb backend = ldapsam":ldap://servidor1/ ldap://servidor2/ ldap://servidor3
    "/
```

Un altre avantatge de tenir més d'un servidor LDAP configurat és que Samba realitza, de manera automàtica, **distribució de càrrega**. Així s'evita que totes les peticions d'autenticació es redirigeixin a un mateix servidor LDAP.

En resum, veiem que cada *backend* té els seus avantatges i inconvenients; la decisió sobre quin escollir depèn de la situació. La taula 1.3 mostra en quines situacions es poden utilitzar aquests *backends*.

Taula 1.3. Ús dels backends en diferents escenaris

	smbpasswd	**tdbsam**	**ldapsam**
Samba com a servidor independent	Sí	Sí	Sí

Taula 1.3 (continuació)

	smbpasswd	**tdbsam**	**ldapsam**
Samba com a controlador de domini (PDC)	No	Sí	Sí
Múltiples controladors de domini (PDC i BDC)	No	No	Sí

2. Configuració i utilització de xarxes heterogènies

En el món empresarial, Samba és un bon aliat dels administradors de sistemes quan es disposa d'una xarxa amb diversos sistemes operatius. Com veureu tot seguit, es pot integrar d'una manera senzilla en xarxes heterogènies, però hem de ser conscients de les seves limitacions.

En aquest context us podeu pot trobar situacions molt diferents i Samba disposa d'un nivell de configuració molt elevat que permet obtenir una bona adaptació a moltes d'aquestes situacions. Fer un recull de totes les possibilitats de configuració resultaria massa extens, però tractarem d'analitzar les dues situacions més comunes que us podeu trobar:

1. Samba com a servidor independent en un grup de treball
2. Samba com a servidor de domini

A continuació veureu com configurar i administrar Samba en cadascuna d'aquestes situacions.

2.1 El servidor Samba en un grup de treball

El servei Samba permet configurar un servidor basat en Linux per tal d'integrar-lo dins d'un grup de treball Windows, de manera que pugui compartir recursos com un equip més del conjunt lògic de la xarxa. Un dels requisits imprescindibles és que aquest servidor es comporti exactament igual que un sistema Windows. Per això, cal ajustar els paràmetres de configuració de Samba tal com es detallen a continuació:

1. **Nivell de seguretat**: el nivell de seguretat determinarà la manera com els usuaris s'identifiquen. El nivell adequat a una xarxa igualitària on intervenen ordinadors Windows és el nivell `user`. Les opcions restants no són adients o bé són caduques.
2. ***Backend***: el tipus de *backend* que fem servir determina de quina manera s'emmagatzemen els usuaris. En aquest cas, l'opció més recomanable és fer servir un arxiu local, de tipus `smbpasswd` o `tdbsam`.
3. **Grup de treball:** el nom del grup de treball (workgroup).
4. **Nom de l'equip:** també conegut com el *nom NetBIOS*. Determina quin nom tindrà aquest equip quan s'explori la xarxa.

5. ***Master browser*(opcional)**: el *master browser* és l'equip que conté la llista de tots els equips connectats a la xarxa.

Per establir aquesta configuració cal modificar l'arxiu `/etc/samba/smb.conf`. Com podreu observar, en aquest arxiu trobareu centenars de paràmetres que poden donar lloc a milers de configuracions diferents.

Tot i que l'arxiu de configuració `smb.conf` és molt extens, per començar heu de realitzar molts pocs canvis respecte a l'arxiu original que s'incorpora amb el paquet.

L'arxiu es divideix en seccions identificades per un nom entre claus, `[·]`. Hi ha tres seccions especials: `[global]`, `[homes]` i `[printers]`. La secció principal s'identifica amb `[global]` i ens permet configurar els paràmetres generals del servei. La secció `[homes]` ens permet compartir els directoris d'inici (*home*) de cada usuari per tal que cada usuari pugui accedir al seu directori a través de la xarxa. La secció `[printers]` permet compartir impressores.

Dins de l'apartat `[global]` s'ha d'especificar el nom del **grup de treball**, el nom de la màquina, el **nivell de seguretat** i el tipus de ***backend*** desitjat.

```
[global]
     workgroup = IOC
     netbios name = Debian-Samba
     security = user
     passdb backend = tdbsam
     encrypt passwords = yes
```

Com veieu, la configuració és simple: el nom amb el que s'anuncia aquesta màquina és Debian-Samba, i s'inclourà al grup de treball anomenat *IOC*. El paràmetre *security* determina el nivell de seguretat i el *backend* s'emmagatzemarà en una base de dades de tipus TDB.

Amb aquesta configuració ja en tindríem prou per incorporar el servidor al grup de treball. Per assegurar-nos que les modificacions del fitxer `smb.conf` es duen a terme, és convenient reiniciar el servei.

```
# service samba restart
Stopping Samba daemons: nmbd smbd.
Starting Samba daemons: nmbd smbd.
```

2.1.1 Usuaris

Samba inclou un conjunt d'eines per manipular els comptes d'usuari emmagatzemats al *backend*. Aquestes eines s'han dissenyat de manera que funcionin igual independentment del tipus de *backend* que feu servir: fitxer TDB, fitxer smbpasswd o directori LDAP. Les eines que es fan servir per manipular usuaris Samba són, principalment, les ordres `smbpasswd` i `pdbedit`.

Per afegir usuaris a Samba farem servir l'ordre `smbpasswd`, tal com mostra l'exemple següent:

```
# smbpasswd –a usuari
New SMB password:
Retype new SMB password:
```

La parella usuari/contrasenya que introduïm aquí és aquella amb la qual s'accedirà des dels clients Windows. Podeu fer servir `smbpasswd` tantes vegades com usuaris vulgueu crear.

L'ordre `smbpasswd` es comporta de dues maneres diferenciades:

1. Si l'executa l'usuari *root*, ens permet administrar els usuaris Samba: crear, esborrar, llistar, habilitar/deshabilitar i canviar la contrasenya.
2. Si l'executa qualsevol altre usuari, li permet canviar la contrasenya Samba en servidors remots.

Les opcions més comuns d'aquesta eina es mostren a la taula 2.1.

Taula 2.1. Rols que pot prendre Samba

Opció	Descripció
`-a nom_usuari`	Afegeix un usuari.
`-d nom_usuari`	Deshabilita el compte d'usuari.
`-e nom_usuari`	Habilita el compte d'usuari.
`-n nom_usuari`	Aplica una contrasenya buida a l'usuari.
`-x nom_usuari`	Esborra el compte d'usuari.
`-h`	Mostra l'ajuda.

Tot i que l'eina *pdbedit* també es pot fer servir per realitzar les mateixes accions, normalment s'utilitza per a tasques administratives de baix nivell, com ara canviar les propietats d'un usuari o importar i exportar usuaris del *backend*.

D'altra banda, Samba necessita que tots els usuaris que accedeixen als recursos es puguin **vincular a un usuari del sistema Linux** (UID). Per aquest motiu, cal que abans d'afegir un usuari Samba el donem d'alta com a usuari del sistema.

```
# useradd usuari
# passwd usuari
Introduzca la nueva contraseña de UNIX:
Vuelva a escribir la nueva contraseña de UNIX:
passwd: contraseña actualizada correctamente
```

Aquest requisit també s'aplica als **usuaris convidats** i per tant, s'han de vincular a un compte específic de GNU/Linux. En general, quan Samba rep la petició d'un usuari que no s'ha pogut autenticar correctament, com en el cas de no trobar l'usuari al *backend*, la petició es rebutja per defecte i no s'atorga l'accés. De vegades, però, ens interessarà que un usuari sense registrar pugui accedir als recursos compartits, com, per exemple, en el cas d'una empresa amb molts

treballadors pot resultar útil que tothom tingui accés a un directori públic sense haver de donar d'alta tots els treballadors. En aquests casos és necessari **permetre l'accés a usuaris no registrats** i **habilitar l'accés de convidat**.

Si es desitja que els usuaris puguin entrar sense estar registrats, el primer que heu de fer és canviar el comportament que Samba té per defecte quan troba un usuari que no és al *backend*. Això es realitza indicant el paràmetre i valor `map to guest = bad user` dins de la secció `[global]`. D'aquesta manera li indiqueu que si no es troba l'usuari al *backend* es faci servir el compte de convidat.

Finalment, cal indicar quin usuari del sistema GNU/Linux es farà servir per vincular els usuaris invitats. Amb el paràmetre `guest account` determineu quin compte d'usuari GNU/Linux es fa servir com a convidat.

```
[global]
     workgroup = IOC
     netbios name = Debian_Samba
     security = user
     passdb backend = tdbsam
     encrypt passwords = yes
     map to guest = bad user
     guest account = invitat
```

Evidentment, el compte de convidat ha d'estar creat prèviament a Linux.

```
# useradd invitat
```

2.1.2 Recursos compartits

Els recursos compartits es defineixen afegint seccions al fitxer de configuració de Samba. Recordeu que el començament de cada secció s'indica amb un nom entre claus, `[·]`, i a continuació s'afegeix la informació relativa a un recurs compartit. Hi ha seccions especials amb noms reservats que no podem utilitzar perquè tenen un significat especial: `[global]`, `[printers]` i `[home]`.

Entre claus s'indica el nom del recurs compartit, que és aquell que veuran els clients quan hi vulguin accedir. Posteriorment, a la variable `path`, indiquem la ruta local al directori compartit.

```
[recurs_compartit]
     path = /directori
```

És indispensable que el directori existeixi prèviament dins del sistema de fitxers original.

```
# mkdir /directori
```

Aquesta configuració és la mínima necessària per compartir un recurs, però cal saber que **per defecte el recurs es compartirà amb permisos només de lectura**.

Tot i això, quan compartiu recursos amb Samba en una xarxa on s'integren sistemes operatius Windows i Linux, hi ha un seguit de **diferències que hem de tenir en consideració**. Principalment, es tracta de diferències directament relacionades amb els sistemes de fitxers.

Com sabeu, Windows i Linux disposen dels seus propis sistemes de fitxers. En el cas de Windows són FAT32 i NTFS, mentre que per a Linux els més comuns són ext2, ext3, ext4 i reiserFS. Cada sistema de fitxers té les seves característiques i limitacions. Entre aquestes limitacions trobem:

1. La distinció entre majúscules i minúscules
2. Els caràcters prohibits en els noms
3. Els accessos directes a Windows o *soft-links* i *hard-links* a Linux

Samba s'encarrega de solucionar les diferències entre sistemes de fitxers Windows i Linux.

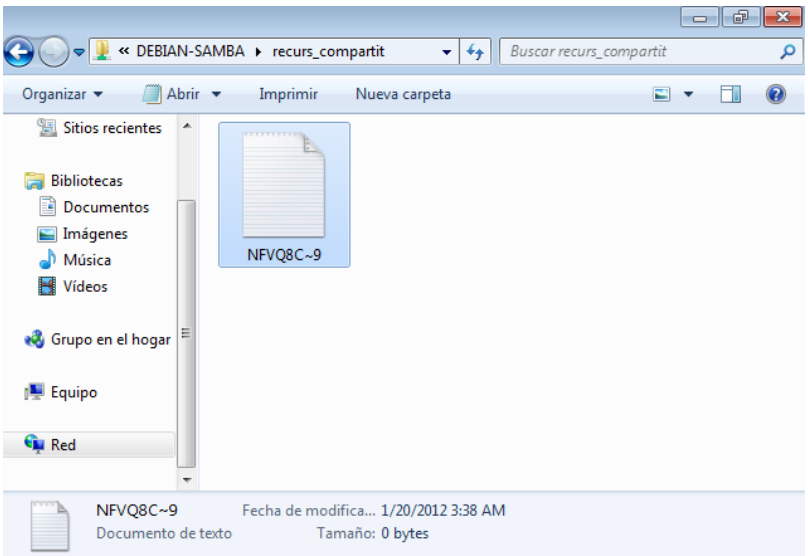
Per exemple, els noms dels fitxers en Windows no distingeixen entre majúscules i minúscules, de manera que els directoris anomenats *escriptori* i *ESCRIPTORI* són el mateix. Per això un directori no pot contenir un arxiu *document.txt* i a la vegada un *DOCUMENT.txt*. En canvi, si fem servir els sistemes de fitxers Linux sí que és possible. Com sabeu, a GNU/Linux se sol distingir majúscules i minúscules.

El sistema operatiu Windows no permet que els noms dels fitxers o directoris continguin els caràcters \/:*?"<>|, però en canvi, a Linux sí que podem crear fitxers amb els caràcters :*?"<>|. Llavors, què ocorre quan un servidor Samba comparteix algun fitxer amb un d'aquests caràcters prohibits per Windows? En aquests casos, el servidor realitza un procés de transformació anomenat *mangle* i mostra al client un nom modificat, no l'original. Aquest nom es representa en format 8.3.

Format 8.3

Els noms del fitxers amb format 8.3 tenen com a màxim 8 caràcters que indiquen el nom, seguits d'un punt i tres caràcters més que indiquen l'extensió de l'arxiu. Aquest format s'utilitzava a MS-DOS i a les primeres versions de Windows.

FIGURA 2.1. Arxiu compartit per Samba anomenat N*.txt vist des d'un client Windows



Tal com podeu observar en la figura 2.1, el procés de transformació permet que els clients Windows puguin accedir als fitxers amb caràcters prohibits, però amb un nom totalment diferent de l'inicial.

2.1.3 Permisos

L'autorització d'una acció sobre un recurs compartit està determinada pels permisos assignats al servidor. Així, si un usuari vol accedir a un recurs compartit i crear-hi fitxers haurà de tenir permisos d'escriptura dins d'aquell directori compartit.

Els recursos compartits per Samba estan afectats per dos tipus de permisos totalment independents:

1. Els permisos del sistema de fitxers
2. Els permisos Samba, també anomenats *permisos de xarxa*

Quan un client desitja realitzar una acció determinada (llegir o escriure), cal que tingui els permisos adequats tant al sistema de fitxers com al recurs compartit. No serveix de res que un usuari tingui tots els permisos a Samba si després no en té cap al sistema de fitxers. Per tant, sempre s'aplica el permís més restrictiu. Tenir això sempre present us estalviarà molts mal de caps.

> Els permisos sobre un recurs compartit depenen del sistema de fitxers i de Samba. En cas d'existir permisos oposats sempre s'aplicarà el més restrictiu dels dos.

Els **permisos del sistema de fitxers** o permisos locals es representen mitjançant 9 bits (`rwxrwxrwx`) amb els quals els sistemes Unix implementen el control d'accés als fitxers del sistema i que podem modificar amb l'ordre *chmod*.

El significat dels permisos varia depenent de si es tracta d'un fitxer o d'un directori, tal com mostra la taula 2.2.

TAULA 2.2. Permisos tradicionals Unix

Permís	Arxiu	Directori
`r`	Llegir arxius	Veure el contingut del directori
`w`	Gravar en un arxiu	Crear i esborrar arxius dins del directori
`x`	Executar com a programa	Entrar al directori
`-`	Sense permís	Sense permís

Convé recordar que en directoris, a més dels tres permisos mostrats, també podem assignar el permís de l'*sticky bit*. L'*sticky bit* permet que només els propietaris dels arxius puguin reanomenar o esborrar els seus arxius encara que la resta d'usuaris

tingui permisos d'escriptura en aquell directori. Les opcions *+t* i *-t* de l'ordre *chmod* permeten afegir o treure l'*sticky bit*.

```
# chmod +t directori
```

Els **permisos Samba** s'especifiquen en el fitxer de configuració, dins de cadascuna de les seccions on hi hagi un recurs compartit. Les variables que controlen els permisos s'indiquen a la taula 2.3. Per defecte, un recurs compartit té permisos només de lectura per a tots els usuaris del *backend*.

Taula 2.3. Variables que controlen els permisos dels recursos compartits

Variable	Definició
`admin users`	Llista d'usuaris amb control total sobre el recurs compartit.
`read list`	Llista d'usuaris que tenen accés de només lectura.
`write list`	Llista d'usuaris que tenen accés de lectura i escriptura.
`guest ok`	Permetre l'accés com a convidat al recurs.
`invalid users`	Llista d'usuaris als quals no se'ls permet accedir.
`valid users`	Llista d'usuaris que sí que hi poden entrar. Si la variable no s'especifica, tots els usuaris amb compte al *backend* hi poden accedir.
`read only`	Indica que el recurs només és de lectura.

Per exemple, la variable *read only = no* permet que els usuaris puguin llegir i escriure (sempre que també ho facin els permisos del sistema de fitxers).

```
[recurs_compartit]
     path=/directori
     read only = no
```

També podem indicar la llista d'usuaris als quals restringim l'accés amb la variable *invalid users*.

```
[recurs_compartit]
     path=/directori
     read only = no
     invalid users = alumne1, alumne2, @professors
```

Llistes d'usuaris i grups

Als permisos de Samba hi podeu indicar una llista d'usuaris amb els noms separats per comes. Els grups es mostren amb el símbol @ al davant.

Quan desitgem que els usuaris **convidats** tinguin accés a un recurs en concret, cal assignar la variable *guest ok* a *yes*. En cas contrari sempre es demanarà l'autenticació de l'usuari.

```
[public]
     path = /directori_public
     guest ok = yes
```

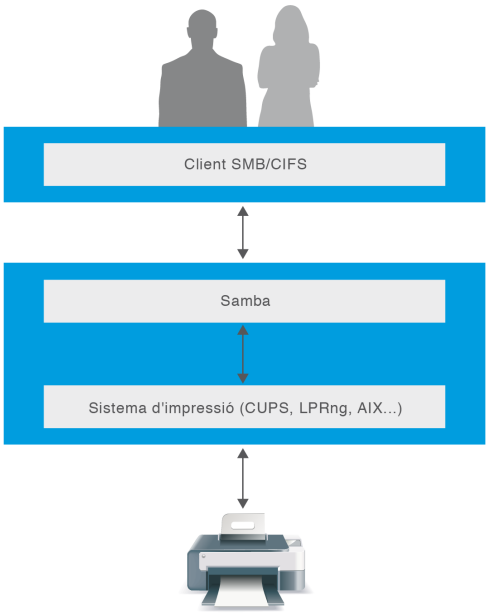
2.1.4 Impressores

Quan pensem en Samba normalment ens ve al cap la funció de compartir directoris, i de fet, aquest és l'ús principal que se li dóna. Però Samba també es pot fer servir com a servidor central d'impressores en una xarxa amb clients Windows. En aquest apartat veurem com configurar el servei per compartir impressores.

Cal dir, però, que **Samba no és un sistema d'impressió per se**, sinó que **actua com a intermediari entre les peticions dels clients iel sistema d'impressió del sistema operatiu** (figura 2.2). Això significa que Samba no es comunica directament amb les impressores, simplement executa les ordres del sistema necessàries per imprimir treballs, posar-los en pausa, cancel·lar-los, veure els treballs en cua, etcètera. En definitiva, Samba es comporta de la mateixa manera que ho faria un usuari de l'equip que vol imprimir.

> Samba no és un sistema d'impressió, sinó que actua com a intermediari entre els clients i el sistema d'impressió del sistema operatiu.

FIGURA 2.2. Esquema d'impressió del servei Samba



CUPS

CUPS (common Unix printing system) és el sistema d'impressió que per defecte es fa servir en moltes distribucions GNU/Linux actuals, entre elles Debian i Ubuntu.

Samba permet la comunicació amb diversos tipus de sistemes d'impressió Unix, entre els quals destaquem HPUX, BSD, PLP, LPRng, i el més utilitzat, **CUPS**.

Vist això, podeu suposar que **el pas previ a compartir una impressora amb Samba és tenir-la correctament configurada en el sistema d'impressió**. En el vostre cas suposarem que feu servir CUPS, i aquest es pot administrar de diverses maneres:

1. Des de les eines gràfiques que proporciona l'escriptori de Debian, fent clic a *Sistema* > *Administració* > *Impressió.*

2. Des del lloc web d'administració de CUPS: http://localhost:631/.

3. Directament des de l'arxiu de configuració a `/etc/cups/cups.conf`.

És convenient assegurar-nos que el servei CUPS està funcionant correctament.

```
# ps –A | grep cupsd
1097 ? 00:00:01 cupsd
```

Per afegir una impressora de manera senzilla podeu visitar el web http://localhost:631/, clicar a la pestanya d'*Administració* i a continuació al botó d'*Afegir impressora*. CUPS necessita que introduïu el nom i la contrasenya de l'administrador (*root*) de la màquina.

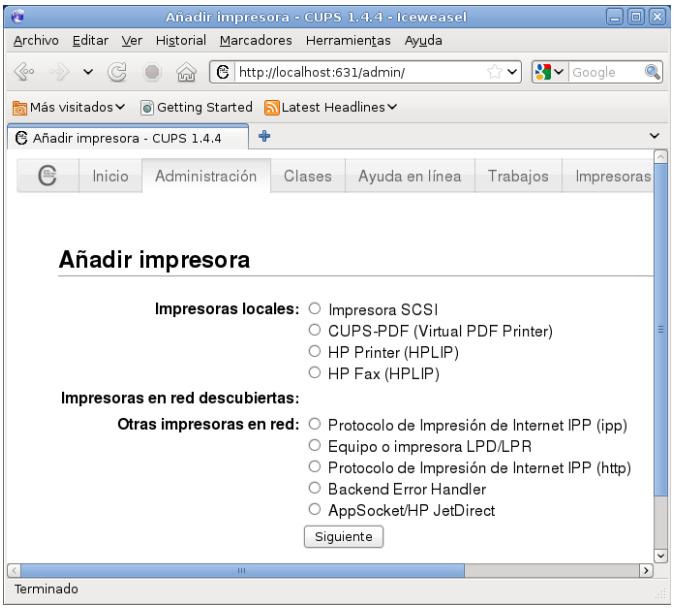
Posteriorment, i com mostra la figura 2.3, cal indicar-li on està connectada la impressora i quin protocol fa servir, el que es coneix com a *backend*. La majoria d'impressores per a entorns empresarials actuals permeten fer servir els protocols IPP, LPD/LPR o JetDirect (AppSocket).

En clicar a *Següent* us demanarà la marca i el model, el nom i la resta d'opcions. Si no és a la llista també podeu afegir un controlador específic amb extensió PPD.

Controladors (drivers) CUPS

Els controladors d'impressores per CUPS són arxius amb extensió `.ppd`. Actualment, la majoria de fabricants disposa de controladors per a Linux.

FIGURA 2.3. Aplicatiu web d'administració de CUPS



És recomanable que l'alumne revisi el funcionament dels serveis d'impressió a GNU/Linux.

Un cop heu afegit la impressora al sistema, heu d'indicar-li a Samba quin servei d'impressió fareu servir, en el nostre cas CUPS.

```
[global]
     workgroup = IOC
     netbios name = Servidor–debian
     security = user
     printing = cups
     printcap name = cups
```

Finalment, cal que afegiu la definició de la impressora compartida de manera similar a com ho faríeu amb un directori, amb les diferències següents:

1. El nom del recurs compartit ha de coincidir amb el nom de la impressora en el servei d'impressió de Linux. Així, si heu posat el nom *HP_1102W* a la impressora a CUPS, caldrà que el recurs a Samba s'anomeni igual.

2. Cal afegir la variable *print ok = yes* al recurs compartit per indicar que aquest recurs és una impressora.

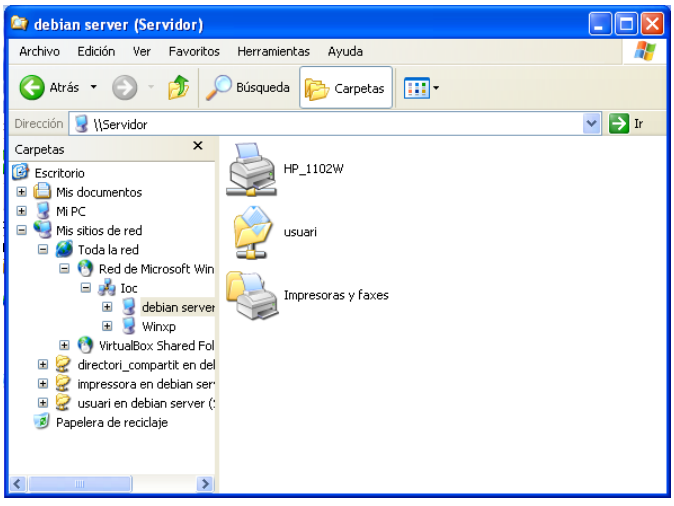
3. La variable *path* ha d'indicar el directori on s'emmagatzemen els treballs que seran impresos. És obligat que el directori indicat tingui permisos d'escriptura per a tothom que pugui imprimir, altrament els clients tindran un error en imprimir. Podem fer servir el directori `/var/spool/samba`, que ja està preparat amb el paquet Samba.

4. Típicament, els controladors d'impressió de Windows envien les dades a la impressora de manera que no és necessari cap filtre de processat, al contrari dels sistemes Linux que envien les dades en PostScript. Si no és necessari cap processat afegiu el paràmetre *cups options = "raw"*.

```
[HP_1102W]
     print ok = yes
     path = /var/spool/samba
     cups options = ""raw
```

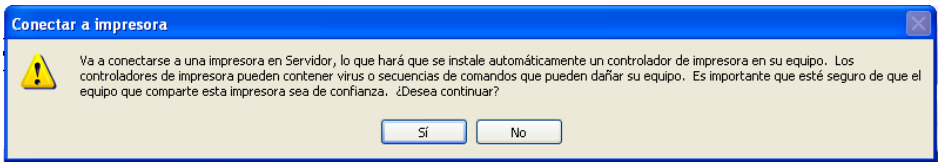
Aquesta és la configuració mínima imprescindible per tenir accessible la impressora des dels clients amb Windows, tal com podeu observar a la figura 2.4.

FIGURA 2.4. Impressora accessible des de clients Windows



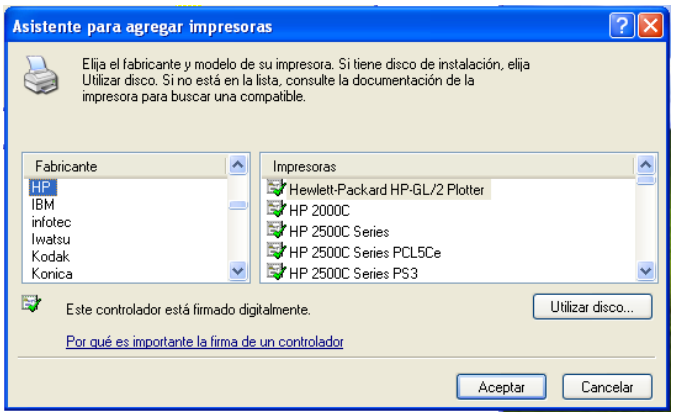
En fer doble clic en la impressora us avisarà que s'intentaran instal·lar els controladors de manera automàtica (figura 2.5).

FIGURA 2.5. Avís d'instal·lació de controladors



Com que no hem proporcionat cap controlador caldrà afegir-lo manualment (figura 2.6).

FIGURA 2.6. Addició manual del controlador



Com veieu, compartir una impressora és una tasca molt senzilla, però es pot millorar per facilitar les tasques l'administració. En primer lloc, podem fer que **Samba detecti automàticament les impressores connectades al sistema d'impressió** i les comparteixi. D'aquesta manera no cal afegir-les una a una. Per fer-ho només cal incloure el recurs compartit especial `[printers]` amb els mateixos paràmetres que una impressora.

```
[printers]
     path = /var/spool/samba
     print ok = yes
```

Aquesta petita modificació pot alliberar la càrrega de l'administrador quan hi hagi moltes impressores connectades a la màquina, com en el cas d'un servidor d'impressió d'una oficina. En segon lloc, podem facilitar la tasca d'instal·lació de les impressores als clients si permetem la **descàrrega i instal·lació automàtica de controladors**, és el que es coneix com a *Apuntar i imprimir*.

> *Apuntar i imprimir* (*Point and print*) és una característica dels dominis Windows que permet a un usuari utilitzar les impressores de xarxa fent clic sobre les seves icones.

Per habilitar aquesta característica, cal que al directori `/var/lib/samba/printers` hi hagi els controladors d'impressió que es poden descarregar els usuaris, separats segons l'arquitectura de la màquina client.

```
# ls -l /var/lib/samba/printers/
total 32
```

```
drwxr-xr-x 2 root root 4096 ene 8 16:56 COLOR
drwxr-xr-x 2 root root 4096 ene 8 16:56 IA64
drwxr-xr-x 2 root root 4096 ene 8 16:56 W32ALPHA
drwxr-xr-x 2 root root 4096 ene 8 16:56 W32MIPS
drwxr-xr-x 2 root root 4096 ene 8 16:56 W32PPC
drwxr-xr-x 2 root root 4096 ene 8 16:56 W32X86
drwxr-xr-x 2 root root 4096 ene 8 16:56 WIN40
drwxr-xr-x 2 root root 4096 ene 8 16:56 x64
```

Així, per exemple, caldrà afegir els controladors per a les màquines amb Windows XP, Vista o 7 de 32 bits dins del directori `W32X86`, els controladors per a les versions del sistema Windows de 64 bits s'hauran d'emmagatzemar dins de `x64`, i així successivament.

Aquest directori cal compartir-lo amb el nom de `[print$]`, que és el recurs on les màquines en un grup de treball Windows cerquen els controladors de les impressores. Llevat dels administradors, és important que tothom tingui accés de només lectura a aquest recurs.

```
[print$]
     path = /var/lib/samba/printers
     browseable = yes
     read only = yes
     guest ok = no
     write list = root
```

Hi ha d'haver al menys un usuari que pugui carregar els controladors en aquests directoris. Com veieu, en l'exemple assignem permisos d'escriptura a l'usuari *root*, però també podríem assignar-los a un altre usuari o grup si fos necessari, sempre i quan hi assignem els permisos d'escriptura al directori `/var/lib/samba/printers`.

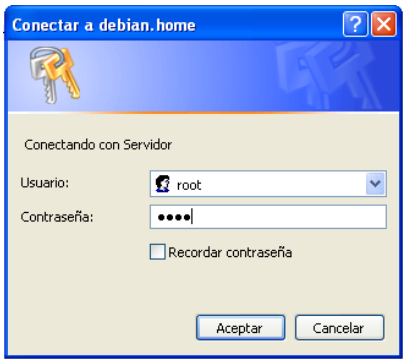
Finalment, cal donar el dret `SePrintOperatorPrivilege` a l'usuari que pugui afegir controladors.

```
# net -U root%contrasenya rpc rights grant root \ SePrintOperatorPrivilege
```

Consulteu l'apartat "Drets" per a més informació sobre com administrar els drets dels usuaris de Samba.

L'usuari *root* ja pot afegir al servidor tots els controladors que desitgi. Per fer-ho cal connectar-se al servidor des d'una màquina amb Windows (figura 2.7).

FIGURA 2.7. Connexió des d'una màquina Windows



Posteriorment cal entrar a *Impressores i faxos* (figura 2.8), fer clic amb el botó dret sobre una zona buida i entrar a *Propietats del servidor*. A Windows 7 haureu de fer clic a *Veure impressores remotes*.

Des d'aquesta finestra es poden administrar certs aspectes de les impressores en el servidor Samba de forma remota.

FIGURA 2.8. Impressores i faxos a Windows

Dins de la pestanya*Controladors* podeu veure els controladors que hi ha instal·lats al servidor Samba. Clicant a *Afegir...* podeu afegir, un per un, els controladors que necessitaran els vostres clients. Només cal indicar a quina ubicació es troben, a quina arquitectura pertanyen (figura 2.9 i figura 2.10) i finalment es carregaran automàticament al servidor, com mostra la figura 2.11.

FIGURA 2.9. Selecció del controlador

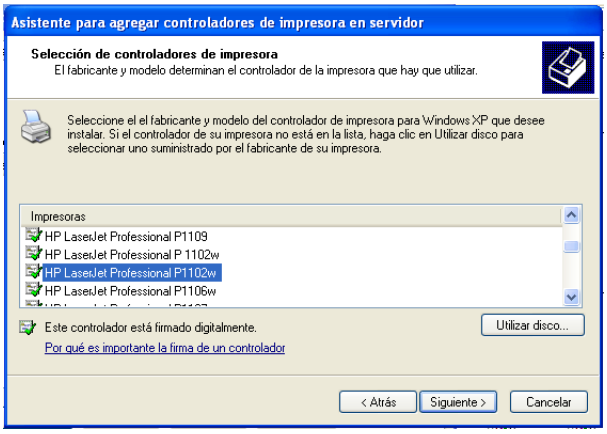


FIGURA 2.10. Selecció del SO de destinació

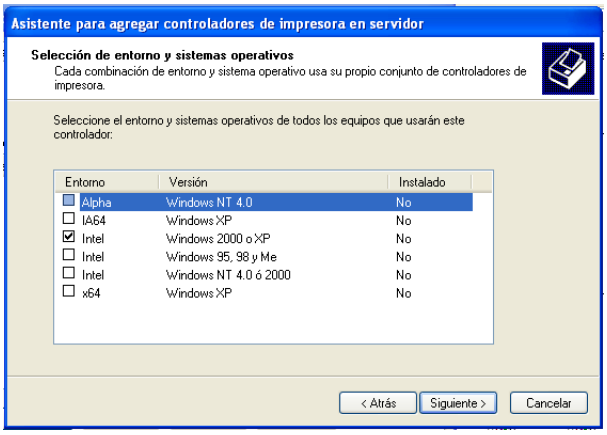
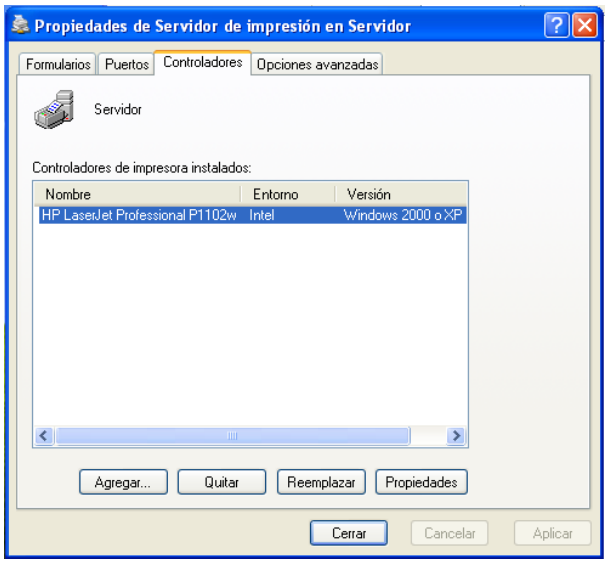


FIGURA 2.11. Controlador instal·lat

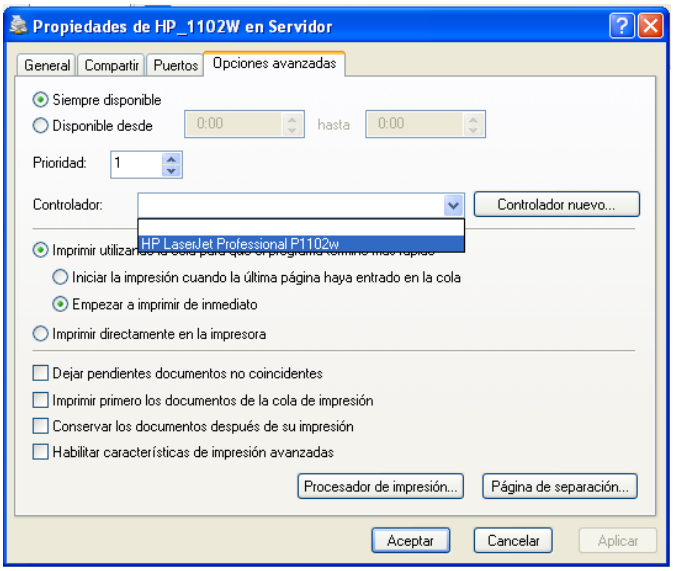


Si en aquest punt observeu el directori compartit de Samba amb els controladors, veureu que han quedat instal·lats a la carpeta corresponent.

```
# ls /var/lib/samba/printers/W32X86/3
AGACCST1.PPD hp1100sd.chm hp1100su.dll hp1600sd.sdd
     PSCRIPT.HLP
HP1100GC.DLL hp1100sd.dll hp1100su.ent PS5UI.DLL
     PSCRIPT.NTF
HP1100PP.DLL hp1100sd.sdd hp1100su.ver PSCRIPT5.DLL
```

Finalment, heu de vincular el controlador que acabeu d'instal·lar a la impressora determinada. Torneu a entra a *Impressores i faxos* (figura 2.8), feu clic amb el botó dret sobre la impressora i aneu a *Propietats*. A continuació, us avisarà que no teniu un controlador assignat i us demanarà si en voleu instal·lar un. Li haureu de dir que **no**, perquè en cas contrari instal·larà un controlador només en l'equip local, i el que voleu és vincular-li un controlador del servidor.

FIGURA 2.12. Selecció del controlador instal·lat



A la finestra que s'obre entreu a *Opcions avançades* i seleccioneu el controlador adient de la llista tal com es mostra a la figura 2.12. Si tot ha anat bé, els clients de la vostra xarxa podran fer servir les impressores simplement fent doble clic sobre elles, sense necessitat que cap tècnic instal·li els controladors ja que es descarreguen de manera automàtica.

Situacions en les quals cal l'ajuda d'un client Windows

Com veieu, en l'administració d'impressores i en altres aspectes de la configuració de Samba cal l'ajuda d'un equip amb el sistema operatiu Windows.

Realment, tota la tasca d'instal·lar els controladors es pot fer manualment des de la màquina amb Samba, però és un procés molt meticulós en el qual qualsevol errada pot provocar errors inesperats que són molt difícils de detectar. En aquest sentit, és recomanable fer servir eines que facilitin la tasca, i per això convé aprofitar les eines que ens ofereix Windows.

2.1.5 Master browser

En un entorn empresarial, els servidors acostumen a ser els equips més potents de la xarxa i els que estan disponibles la major part del temps. En aquest escenari és probable que Samba sigui un dels serveis instal·lats. Llavors, per què no aprofitar la potència d'aquestes màquines i fer que Samba actuï com a *master browser*?

Si Samba s'instal·la en una màquina amb alta disponibilitat i es configura com a *master browser* aprofitarem la potència de la màquina on està instal·lat i podrem evitar part dels problemes de sincronització deguts a la desconnexió sobtada de les màquines de l'entorn de treball.

Vegeu l'apartat "Xarxes igualitàries en entorns Windows" per a una descripció complementària sobre el rol del *master browser*.

En cas contrari podeu córrer el risc que una màquina menys potent, com la d'un treballador, assumeixi aquest rol. Si aquesta màquina no és suficientment potent o està saturada de treball, la resta del grup de treball tindrà problemes de lentitud quan vulgui accedir a la xarxa.

Qualsevol de les màquines d'un grup de treball és susceptible de prendre el rol de *master browser*. Si volem que Samba tingui aquest rol l'hem de **forçar perquè guanyi les eleccions**.

L'algorisme d'elecció està implementat en tots els sistemes Windows, de manera que quan es duu a terme, tots ells estaran d'acord en qui serà el *master* i el *backup browser*. També cal saber que en qualsevol moment es pot forçar el procés d'elecció.

El procés d'elecció es basa en alguns aspectes dels ordinadors, com ara el temps que porten encesos, el sistema operatiu o la versió del protocol que fan servir. L'algorisme de Microsoft determina la jerarquia següent:

1. Controladors de domini Windows NT/2000/2003/2008 (valor 32)

2. Windows NT/2000/2003/2008/Vista/XP/7 (valor 16)

3. Windows 95/98/Me (valor 1)
4. Windows per a grups de treball (valor 1)

El sistema operatiu que tingui un valor més alt en l'escala d'aquesta jerarquia guanya l'elecció. Així, doncs, un controlador de domini de Windows 2003, amb 32 punts, guanyarà l'elecció davant d'una màquina amb Windows 7, amb 16 punts. En cas d'empat entren en joc la resta de variables.

Problemes de seguretat en el procés d'elecció del master browser

El mecanisme utilitzat en el procés d'elecció permet que qualsevol màquina de la xarxa pugui prendre el rol de *master browser*. Això ocasiona un problema de seguretat, ja que d'aquesta manera, una màquina d'un possible atacant podria forçar l'elecció, guanyar-la i fer modificacions en la llista d'equips i suplantar així la identitat d'una de les màquines. A partir d'aquí, l'atacant podria obtenir els noms d'usuari i contrasenyes que envien els clients.

Per defecte, un servidor Samba està configurat per poder actuar com a *master browser* i s'assigna a si mateix un valor 20. Per tant, **en una xarxa igualitària** (sense controladors de dominis) **un ordinador Samba sempre guanyarà l'elecció davant d'equips Windows**. Si, en canvi, hi ha un controlador de domini, la perdrà, ja que aquests tenen assignat el valor 32.

En qualsevol cas, sempre podeu establir el valor des de l'arxiu de configuració amb la variable *os level*, que com a màxim pot prendre el valor 255.

```
1 [global]
2      ...
3      os level = 100
4      ...
```

Si a més es desitja forçar el procés d'elecció cada vegada que s'inicii el servidor Samba, cal posar la variable *preferred master* = *yes*.

```
1 [global]
2      ...
3      os level = 100
4      preferred master = Yes
5      ...
```

La taula 2.4 mostra un resum de les variables implicades en l'elecció del *master browser*.

TAULA 2.4. Variables implicades en el procés d'elecció del master browser.

Variable	Valor	Descripció
`local master`	yes/no	Determina si Samba pot actuar com a *master browser*. Per defecte, ho pot fer.
`os level`	Numèric 0-255	Valor que es tindrà en compte en el procés d'elecció. Com més elevat, més probabilitats té de guanyar-lo.
`preferred master`	yes/no	Si s'estableix a *yes*, es forcen les eleccions quan Samba s'incorpora a la xarxa.

2.1.6 Accés als recursos mitjançant clients GNU/Linux

A Linux un client disposa de diverses maneres d'accedir als recursos compartits en un grup de treball. En qualsevol distribució podeu descarregar-vos el paquet `smbclient` i accedir-hi via ordres.

```
# apt-get install smbclient
```

El paquet `smbclient` conté diverses eines:

```
# dpkg -L smbclient | grep bin
/usr/bin
/usr/bin/smbspool
/usr/bin/rpcclient
/usr/bin/smbtree
/usr/bin/smbcacls
/usr/bin/findsmb
/usr/bin/smbget
/usr/bin/smbcquotas
/usr/bin/smbclient
/usr/bin/smbtar
```

Només veureu algunes de les opcions més significatives, ja que explicar la funció de totes elles ocuparia molt temps, i en cas necessari totes disposen de pàgines al manual que podeu consultar escrivint *man <ordre>* en un terminal.

L'ordre `smbtree` mostra tots els equips del grup de treball en mode text, de manera similar a l'*Entorn de xarxa* de Windows. Dibuixa un arbre amb tots els dominis coneguts, els servidors que comparteixen recursos i els seus noms.

```
# smbtree -N
IOC
     \\WINXP
          \\WINXP\C$
          \\WINXP\ADMIN$               Admin remota
          \\WINXP\Compartit2
          \\WINXP\Compartit1
          \\WINXP\IPC$                 IPC remota
     \\SERVIDOR                   debian server
          \\SERVIDOR\HP_1102W          HP_1102W
          \\SERVIDOR\IPC$              IPC Service
          \\SERVIDOR\print$            Printer Drivers
```

Amb la opció *-N* podeu fer la consulta sense necessitat d'introduir cap contrasenya.

Per llistar els recursos compartits per una màquina es pot fer servir l'ordre `smbclient -L` seguida del nom de l'ordinador.

```
# smbclient -L WINXP -N
Domain=[WINXP] OS=[Windows 5.1] Server=[Windows 2000 LAN Manager]

     Sharename      Type      Comment
     ---------      ----      -------
     IPC$           IPC       IPC remota
     Compartit1     Disk
     Compartit2     Disk
     ADMIN$         Disk      Admin remota
     C$             Disk      Recurso predeterminado
```

En l'exemple podeu observar que la màquina comparteix cinc recursos en total, tres dels quals són ocults. Fent servir la mateixa ordre podeu accedir al recurs indicant l'adreça UNC, i amb la opció *-U* especifiqueu quin usuari feu servir per realitzar la connexió.

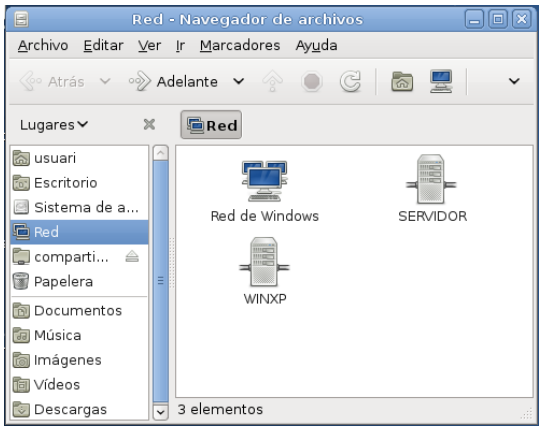
```
# smbclient //WINXP/Compartit1 -U usuari
Enter usuari's password:
Domain=[WINXP] OS=[Windows 5.1] Server=[Windows 2000 LAN Manager]
smb: \>
```

Un entorn interactiu us permet realitzar les accions desitjades en el recurs compartit. Algunes de les ordres utilitzades són semblants a les d'Unix (*cd*,*ls*,*pwd*...), mentre que per pujar o descarregar-nos fitxers podeu fer servir *put* i *get*, respectivament. En cas de dubte, escrivint *help* obtindreu la llista d'opcions.

```
smb: \> help
```

Si feu servir Debian o Ubuntu podeu accedir amb l'interfície gràfica que proporciona l'escriptori. S'hi pot accedir des de *Llocs* > *Xarxa* i s'obrirà la finestra mostrada a la figura 2.13.

FIGURA 2.13. Accés a la xarxa des de l'interfície gràfica d'Ubuntu



De manera senzilla podeu navegar pels diferents servidors de xarxa de forma molt similar a Windows.

2.1.7 Accés als recursos mitjançant clients Microsoft Windows

L'accés als recursos compartits per Samba mitjançant un client Windows es pot realitzar fent servir les utilitats gràfiques que proporciona el mateix explorador del sistema operatiu de forma totalment intuïtiva.

Per accedir al servidor Samba primer heu d'assignar un nom NetBIOS i unir l'equip Windows al mateix grup de treball. Per fer-ho, a Windows Vista i Windows 7 seguiu els passos següents:

1. Fer clic a *Inici >Equip.*

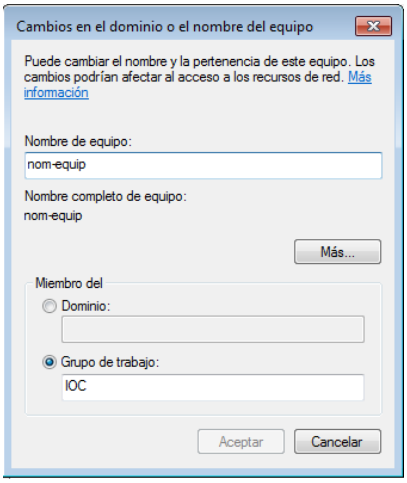
2. Fer clic al botó superior amb l'etiqueta *Propietats del sistema*.

3. Dins de la secció *Configuració del nom, domini i grup de treball del equip*, fer clic a *Canviar configuració*.

4. Fer clic al botó *Canviar*.

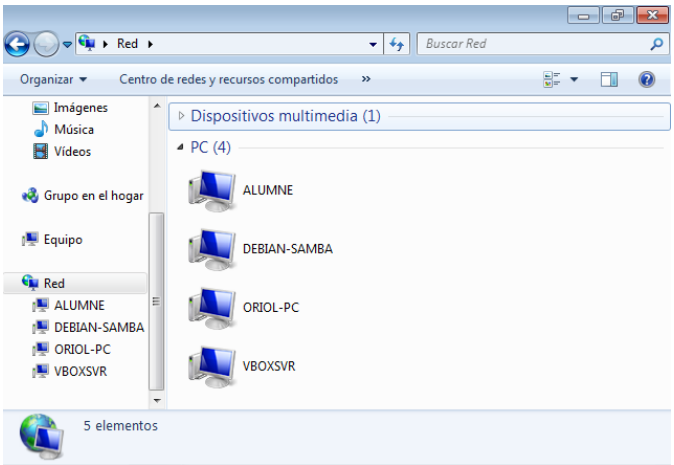
A la figura 2.14 s'observa la finestra de configuració d'un ordinador unit al grup de treball IOC amb el nom *nom-equip*. Un cop unit al grup de treball s'ha de reiniciar la màquina.

FIGURA 2.14. Finestra de configuració del grup de treball



S'accedeix als recursos compartits a través de l'explorador de la xarxa o bé escrivint l'adreça UNC del recurs compartit directament a la barra superior (per exemple, `\\Servidor\recurs`), fent servir el nom NetBIOS del servidor o bé l'adreça IP seguida del nom del recurs, tal i com mostra la figura 2.15.

FIGURA 2.15. L'accés als recursos compartits mitjançant Windows es realitza de manera intuïtiva



2.2 El servidor Samba com a controlador primari de domini

Quan el nombre d'equips d'una xarxa creix, l'administració dels recursos es fa cada vegada més complicada. Per això, en aquestes situacions és convenient fer servir un model de xarxa centralitzat amb un controlador de domini.

Per configurar Samba com a CPD cal realitzar els passos següents:

1. Especificar el nom del domini.
2. Admetre les peticions de *login*.
3. Establir el rol de *domain master browser*.
4. **Opcional:** compartir el recurs especial *netlogon*.
5. **Opcional:** habilitar els perfils mòbils.

Per establir el nom del domini es fa servir el paràmetre *workgroup*, el mateix que s'utilitza per especificar el grup de treball quan Samba és un servidor *stand-alone*. Així, si voleu crear un domini anomenat *IOC* cal que estabiu el paràmetre *workgroup = IOC*.

D'altra banda, Samba atén les peticions d'entrada al domini (*login*) quan estabiu el paràmetre *domain logons = yes*. En aquest cas també és necessari assignar-li el rol de *domain master* (`domain master = yes`) per tal que el servidor contingui la llista d'ordinadors del domini.

Domain master browser

El domain master browser conté la llista de tots els equips del domini, i no només els de la seva xarxa, com en el cas del local master browser.

```
[global]
     workgroup = IOC
     netbios name = Debian_Samba
     security = user
     encrypt passwords = yes
     domain logons = yes
     domain master = yes
```

Aquesta és la configuració mínima que requereix un servidor Samba per actuar com a controlador primari de domini. És convenient, però, que si realment voleu aprofitar les funcionalitats que ofereix un domini Windows NT afegiu també el recurs compartit *netlogon* i habiliteu els perfils mòbils.

Vegeu els apartats "El recurs *netlogon*" i "Perfils mòbils" per a més informació.

Cal reiniciar el servei per aplicar els canvis. Si us voleu assegurar que Samba s'ha configurat correctament com a controlador de domini, podeu consultar els missatges de *log* del dimoni `nmbd`.

```
# tail -f /var/log/samba/log.nmbd
```

En cas afirmatiu, obtindreu un missatge similar al següent:

```
[2012/01/25 09:03:59.523972, 0] nmbd/nmbd_logonnames.c:121(
    become_logon_server_success)
  become_logon_server_success: Samba is now a logon server for workgroup DOMINI
       on subnet 192.168.1.2
```

2.2.1 Comptes d'usuari i d'equip

Els usuaris que poden entrar al domini s'han de donar d'alta prèviament en el controlador de domini. Per tant, cal afegir-los primer com a usuaris GNU/Linux i després incorporar-los al *backend*.

El procediment per afegir usuaris és el que ja coneixeu, però abans cal considerar un aspecte relacionat amb la seguretat: la finalitat és que els usuaris puguin entrar al domini, però en cap cas que puguin fer servir el compte per iniciar sessió al servidor. Per això, podeu blocar l'entrada al servidor si no li assigneu cap *shell*. En aquest cas, *-s /bin/false*.

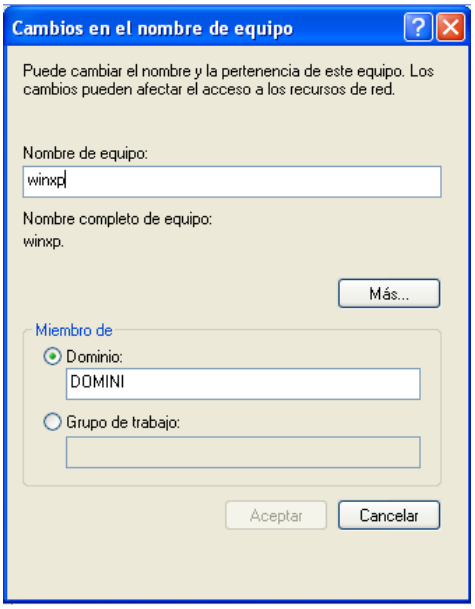
```
# useradd -d /home/usuari -s /bin/false -m usuari
# smbpasswd -a usuari
New SMB password:
Retype new SMB password:
Added user usuari.
```

Això no és suficient. A més d'afegir un compte d'usuari, per incorporar un equip al domini s'ha de crear un **compte d'equip**.

Els comptes d'equip, anomenats ***workstation trust accounts***, permeten que el controlador de domini mantingui una llista sobre quines són les màquines en les quals cal atendre els processos d'autenticació. D'aquesta manera, el controlador sap en tot moment si ha de fer cas o ha d'ignorar la petició de *login* que li arriba des d'un equip.

És indispensable crear un compte d'equip abans d'afegir-lo al domini.

FIGURA 2.16. Cada màquina dins del domini s'identifica per un nom



El compte d'un equip s'identifica per un nom i ha de coincidir amb el nom del sistema de Windows. Com qualsevol altre compte, s'ha d'emmagatzemar al *backend* de manera similar. Es podendiferenciar perquè **un compte d'equip conté el símbol del dòlar** (**$**) al final del nom. En el cas de tenir un equip Windows configurat tal com mostra la figura 2.16, hauríeu d'afegir un compte de màquina anomenat `winxp$`.

Recordeu que qualsevol compte que afegiu al *backend* ha d'estar també donat d'alta com a usuari Linux, i el cas dels comptes d'equip no és una excepció. És recomanable tenir els comptes de màquines agrupats, per això podeu crear un grup de Linux on afegirem tots els comptes de màquina.

```
# groupadd maquines
```

Posteriorment, cal afegir un nou usuari amb el mateix nom que la màquina acabat en $ que podeu assignar directament al grup de màquines.

```
# useradd -g maquines winxp$
```

A continuació ja podeu afegir el compte al *backend*, fent servir el paràmetre *-m* de l'ordre *smbpasswd* per indicar que és un compte d'equip.

```
# smbpasswd -a -m winxp
Added user winxp$.
```

Els comptes d'equip en servidors Windows

Per defecte, en dominis controlats per servidors Windows no cal crear comptes de màquina manualment com fem en Samba. La realitat és que Windows Server les afegeix de manera automàtica quan una màquina s'uneix al domini.

Amb Samba, podeu simular aquest comportament. La variable add machine script de l'arxiu de configuració permet que s'executi una ordre personalitzada cada vegada s'incorpora una màquina al domini.

Si voleu que els comptes d'equip es creïn automàticament haureu d'assignar l'ordre apropiada a la variable*add machine script*. Aquesta s'executarà cada vegada que detecti que una màquina es vol incorporar al domini. Si la màquina ja existeix no s'executa. En aquest cas es tracta d'executar l'script *useradd*.

```
[global]
     ...
     add machine script = /usr/sbin/useradd -g \ maquines -d /var/lib/samba -s/
          bin/false %u
     ...
```

2.2.2 Accés al domini mitjançant clients Microsoft Windows

Per unir una màquina Windows heu de seguir els passos següents:

1. Fer clic a *Inici* > *Equip*.

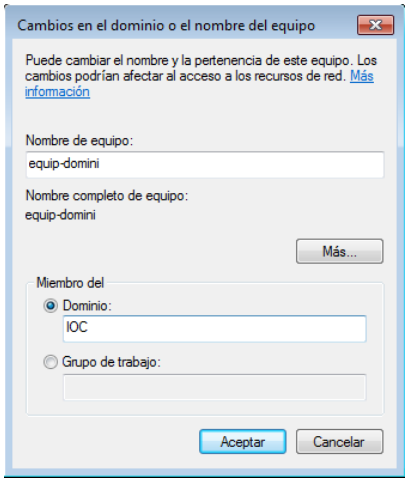
2. Fer clic al botó superior amb l'etiqueta *Propietats del sistema*.

3. Dins de la secció *Configuració del nom, domini i grup de treball del equip* fer clic a *Canviar configuració*.

4. Fer clic al botó *Canviar*.

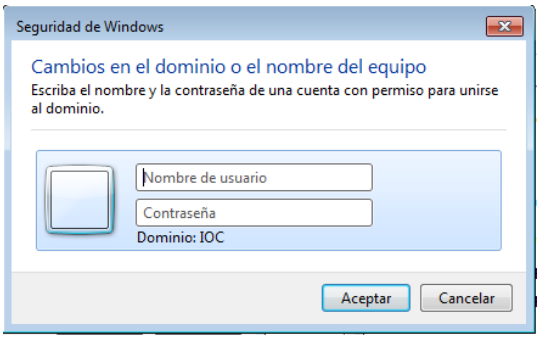
En aquesta finestra, mostrada en la figura 2.17, heu d'indicar el nom d'una màquina del domini i el nom del domini al qual voleu unir-vos. És imprescindible haver creat abans un compte de màquina en el controlador de domini.

FIGURA 2.17. Configuració del domini per accedir-hi



És important destacar que en un domini Samba, el nom del equip i el nom del domini no es poden canviar a la vegada.

FIGURA 2.18. Només un usuari amb permisos adequats pot afegir l'equip al domini



Posteriorment cal **indicar un compte d'usuari amb dret per unir màquines al domini** a la finestra mostrada en la figura 2.18. Quan el controlador de domini és Windows Server, normalment es fa servir el compte d'usuari *Administrador*, en canvi en dominis controlats per Samba es fa servir el compte *root*.

En general, l'usuari *root* té aquest dret per defecte, però si es vol delegar la feina a altres usuaris (recomanable) podeu assignar el dret *SeMachineAccountPrivilege* a qui cregueu convenient.

```
# net -U root%contrasenya rpc rights grant lluis \ SeMachineAccountPrivilege
Successfully granted rights.
```

En l'exemple s'ha donat el dret d'afegir i treure màquines del domini a l'usuari *lluis*. És evident que aquest dret no l'hauria de tenir tothom, només les persones encarregades d'administrar el sistema informàtic.

Consulteu l'apartat "Drets" per a més informació sobre com administrar els drets dels usuaris de Samba.

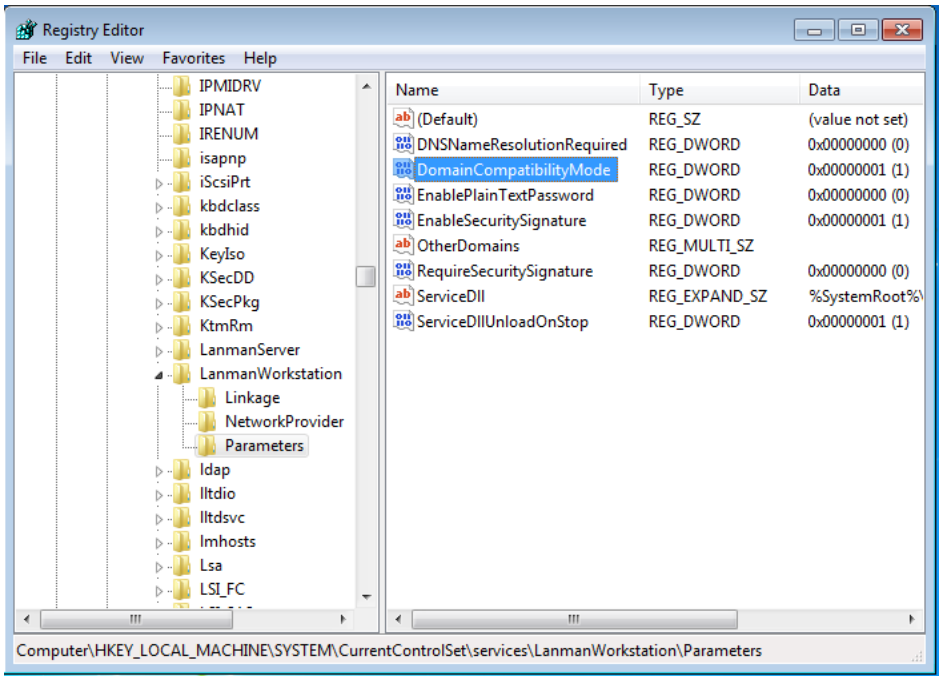
En general, no tindreu cap problema en el moment d'afegir un equip Windows excepte si ho intenteu amb un client Windows 7 o Windows 2008 R2. En aquests casos cal realitzar abans una sèrie de modificacions al registre del client (*regedit.exe*). La raó és que Microsoft ha deshabilitat per defecte la compatibilitat amb controladors de domini NT.

Cerqueu la clau següent:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\LanmanWorkstation\
    Parameters
```

En aquest punt cal afegir les dues noves variables següents amb els seus respectius valors tal com es mostra a la figura 2.19.

Figura 2.19. Addició de noves variables al registre de Windows

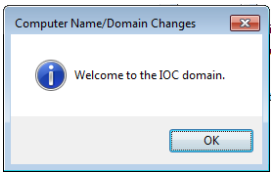



```
DWORD DomainCompatibilityMode = 1
DWORD DNSNameResolutionRequired = 0
```

Per afegir variables al registre feu clic amb el botó dret sobre la zona buida del editor i seleccioneu *Nuevo* > *Valor DWORD*, escriviu el nom i feu doble clic a sobre per assignar-li el valor.

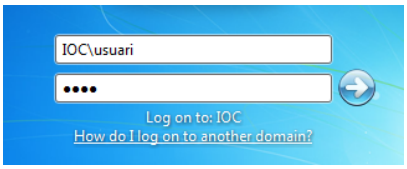
Salveu els canvis, reinicieu i podreu incorporar aquesta màquina al domini (figura 2.20).

FIGURA 2.20. Incorporació de la màquina al domini



Per entrar al domini amb un usuari heu d'indicar *NOMDOMINIusuari* a la pantalla de benvinguda del sistema operatiu, tal com mostra la figura 2.21.

FIGURA 2.21. Entrada amb un usuari



2.2.3 El recurs netlogon

En un controlador de domini, disposar del recurs compartit especial `[netlogon]` **és opcional**, però molt recomanable. Aquest recurs és un directori on s'emmagatzemen diversos arxius de gran utilitat en dominis, entre els quals trobem:

1. **El perfil per defecte dels usuaris**: quan un usuari entri al domini per primera vegada es carregarà el perfil per defecte. El perfil s'emmagatzema al subdirectori Default User.
2. **Els scripts de*login*:** Samba permet l'execució d'scripts de *login* Windows, arxius amb extensió *bat* o *cmd*que s'executen cada vegada que un client entra al domini. Aquests scripts s'emmagatzemen al recurs compartit i es transporten a la màquina client quan s'inicia sessió, i posteriorment s'executen. Cada script de *login* s'ha d'emmagatzemar al directori arrel del recurs `[netlogon]`.
3. **Les polítiques de grup**: fitxers `ntconfig.pol` o `config.pol`.

Aquest recurs es pot compartir com qualsevol altre, amb el requisit que ha de tenir permisos de **lectura per a tothom**i **opcionalment permisos totals per als administradors**. Altrament, qualsevol podria modificar els scripts de *login* i comprometre la seguretat del domini.

```
# mkdir /netlogon
# chgrp admins /netlogon
# chmod 775 /netlogon
```

El fitxer de configuració de Samba el compartirem amb el nom de `[netlogon]`.

```
# cat /etc/samba/smb.conf
```

```
[global]
     workgroup = IOC
     netbios name = Debian-Samba
     security = user
     encrypt passwords = yes
     domain logons = yes
     domain master = yes
[netlogon]
     path = /netlogon
     guest ok = yes
     read only = yes
     write list = @admins
     browseable = no
```

El paràmetre *browseable = no* amaga el recurs de l'explorador de xarxa.

Podem fer que cada vegada que un usuari entri en una màquina s'executi un script determinat. A la variable *logon script* cal indicar quin script volem que s'executi.

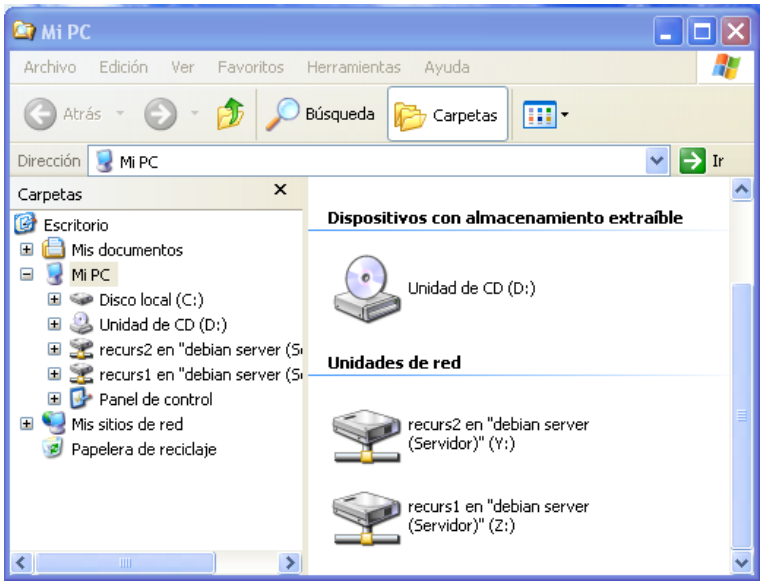
```
[global]
     ...
     logon script = logon.cmd
     ...
```

Cal que l'script estigui disponible al directori arrel del recurs *netlogon*. És a dir, si el recurs compartit és el directori `/netlogon`, llavors l'script haurà d'estar a `/netlogon/logon.cmd`.

Per exemple, si voleu muntar unitats de xarxa automàticament cada vegada que l'usuari inicia sessió (figura 2.22) podeu crear l'script següent:

```
# cat /netlogon/logon.cmd
net use z: \\Servidor\recurs1
net use y: \\Servidor\recurs2
```

FIGURA 2.22. Muntat automàtic d'unitats de xarxa



Recordeu que els scripts s'executen a la màquina del client (Windows), per tant, han d'estar en format DOS (final de línia CR/LF) i han de contenir ordres pròpies d'aquest sistema operatiu.

Els scripts del recurs *netlogon* han d'estar en format DOS, altrament no s'executaran correctament a la màquina del client.

Si genereu l'script amb un editor de Linux (**vi**, **nano**, **gedit**...), quan finalitzeu heu de canviar el format de l'arxiu d'Unix a DOS. Una de les maneres de fer-ho és amb la utilitat *todos*.

```
# apt-get install tofrodos
# todos /netlogon/logon.cmd
```

Formats DOS i Unix

Històricament, els arxius de text en DOS i Unix han fet servir codis diferents per representar el salt de línia, fet que provoca que els scripts editats en una família de sistemes operatius no es puguin executar directament en l'altre. S'han de convertir prèviament.

2.2.4 Perfils d'usuari

Els perfils d'usuari són una de les eines més importants de Windows per a la configuració de l'entorn de treball. Defineixen un entorn d'escriptori personalitzat, en el que s'inclou la configuració individual de la pantalla, així com les connexions de xarxa, les impressores, etcètera. Per defecte, cada usuari d'un sistema Windows té el perfil associat al nom d'usuari i s'emmagatzema al disc dur, normalment a `c:\documents and settings\` a Windows XP o `c:\Users` a Windows Vista i 7.

En un domini Windows distingim quatre tipus de perfils:

1. Perfils locals
2. Perfils per defecte d'un domini
3. Perfils obligatoris
4. Perfils mòbils

En un domini podeu definir un **perfil per defecte**, que és el que faran servir tots els usuaris quan es connectin per primera vegada. D'aquesta manera podem estalviar molta feina de configuració que s'hauria de realitzar màquina a màquina. Posteriorment, podeu habilitar els **perfils mòbils**, de manera que el perfil de l'usuari es desi al servidor cada vegada que tanca sessió i que es recuperi quan l'inicia. Si, en canvi, voleu que els usuaris puguin fer canvis durant la sessió però no desar els canvis, podeu establir un **perfil obligatori**.

Perfils per defecte

Quan un sistema Windows inicia sessió en un domini amb un usuari que no hi ha entrat mai, intenta descarregar-se el perfil per defecte del recurs compartit `[netlogon]` del controlador de domini. En cas de trobar-lo, fa una còpia local i inicia la sessió de l'usuari. Si no el troba s'inicia una sessió amb un perfil buit.

Hi ha dues versions de perfils diferents:

1. Perfils per a Windows XP, Windows 2000 i Windows 2003
2. Perfils versió 2, per a Windows Vista, Windows 7 i Windows 2008

Els diferents tipus de perfils són incompatibles entre si, el que significa que si al domini tenim sistemes de la família Windows XP i de Windows 7 haurem de crear dos perfils per defecte diferents.

> Els perfils de la família de sistemes operatius Windows XP, 2000 i 2003 són incompatibles amb els de Windows Vista, 7 i 2008.

Pels clients amb Windows XP, el perfil per defecte ha d'estar emmagatzemat dins de la UNC del servidor `\\<Servidor>\netlogon\Default User`. Així, si la configuració del servei *netlogon* de Samba és la següent:

```
[netlogon]
     path = /netlogon
     guest ok = yes
     read only = yes
     write list = @admins
```

Vegeu l'apartat "Identificadors de seguretat i grups" per a més informació sobre com crear un grup d'administradors a Samba.

Llavors el perfil s'haurà de situar dins de `/netlogon/Default User` i caldrà que tothom hi tingui permisos de lectura; altrament no podran descarregar-s'ho. Per fer-ho heu de seguir els passos següents:

1. **Generar un perfil local i adaptar-lo a les vostres necessitats**: inicieu sessió com un usuari local de la màquina i configureu l'entorn (configuració de programes, escriptori, documents, etcètera).
2. **Tancar la sessió d'usuari local**.
3. **Iniciar la sessió al domini amb un compte d'administrador** (figura 2.23). Si no heu generat el grup d'administradors, podeu fer servir l'usuari *root*.
4. **Copiar el perfil del pas 1 al servidor**: *Inicio* > *Mi PC* > *Propietats* > *Opcions avançades*. A la finestra mostrada a la figura 2.24 premeu el botó *Configuració* de l'apartat *Perfils d'usuari*.

   Seleccionar el perfil local que desitgem copiar al servidor i fer clic a *Copiar a...* (figura 2.25). A continuació indiqueu l'adreça al servei *netlogon* del servidor: `\\<Servidor>\netlogon\Default User\`.

   Si no podeu realitzar aquest pas segurament és perquè l'usuari amb el que us heu connectat no és administrador del domini o bé no té permisos per escriure en el recurs *netlogon*.
5. **Verificar que els arxius s'han copiat correctament al servidor**. Us podeu assegurar que el perfil s'ha copiat correctament entrant en el servidor i llistant el contingut del recurs *netlogon*. Haureu de poder veure el directori Default User.

```
# ls -l /netlogon
drwxr-xr-x 13 usuari usuari 4096 mar 2 17:53 Default User
```

FIGURA 2.23. Inici de sessió en Windows

FIGURA 2.24. opcions avançades de les Propietats del sistema

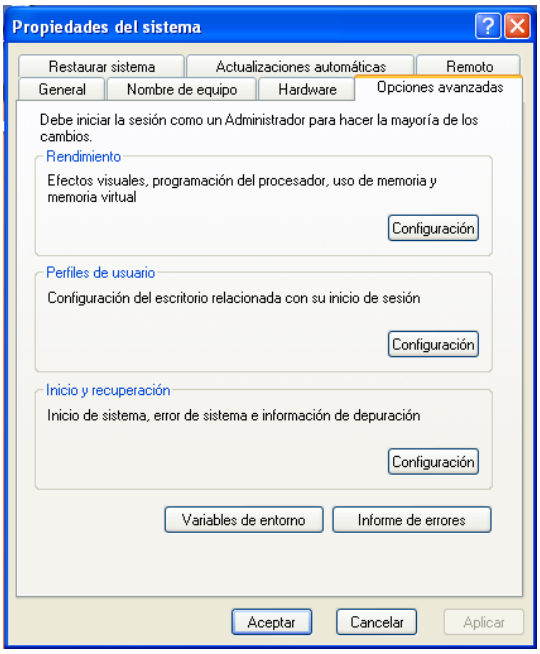
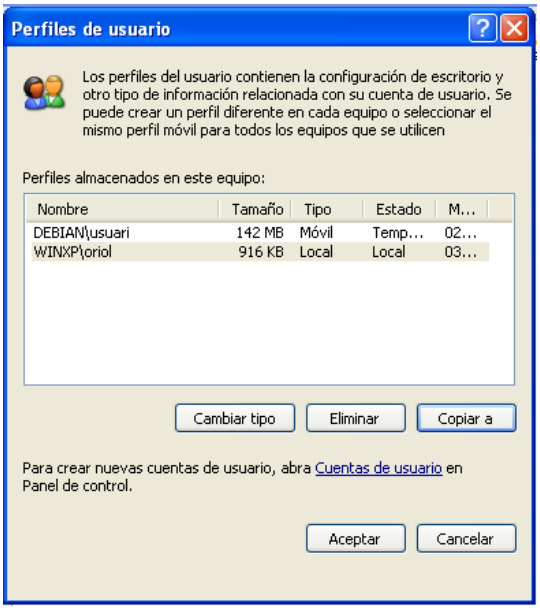


FIGURA 2.25. Perfils d'usuari



Els sistemes Windows Vista i 7 fan servir un altre tipus de perfils, coneguts com a *Perfils versió 2*. Per això, caldrà copiar-los al servidor dins del directori `Default User.v2` del recurs *netlogon* (`\\<Servidor>\netlogon\Default User.v2\`).

Els passos a seguir en sistemes Windows Vista, 7 o 2008 són molt similars als de Windows XP; l'única diferència és que només podem copiar al servidor el perfil per defecte de la màquina. En qualsevol cas, les accions que cal dur a terme són:

1. **Generar un perfil local i adaptar-lo a les vostres necessitats**: inicieu sessió com un usuari local i configureu l'entorn.

2. **Tancar la sessió d'usuari local i obrir una sessió local d'administrador**: inicieu sessió amb un compte d'usuari diferent de l'utilitzat en el pas anterior.

3. **Copiar el perfil generat en el primer pas al perfil per defecte**: obriu l'explorador, navegueu a `c:\Users` i seleccioneu el perfil de l'usuari local que heu generat en el primer pas. Canvieu el nom del perfil a *Default*. Com que aquest directori ja existeix, heu de canviar-li el nom prèviament: `Default > Copia` (figura 2.26).

4. **Iniciar sessió amb un compte d'administrador del domini**.

5. **Copiar el perfil per defecte al controlador de domini**: *Inici* > *Equip* > *Propietats* > *Opcions avançades*. Fer clic a la pestanya *Avançat*, i en la secció *Perfils d'usuari* prémer el botó *Configurar...*. Us apareixerà una finestra similar a la de la figura 2.27. Seleccioneu el perfil per defecte i copieu-lo al servidor.

6. **Verificar que els arxius s'han copiat correctament al servidor**.

```
# ls -l /netlogon
total 12
drwxr-xr-x 14 usuari usuari 4096 mar 4 10:10 Default User.v2
```

Figura 2.26. Còpia d'un perfil

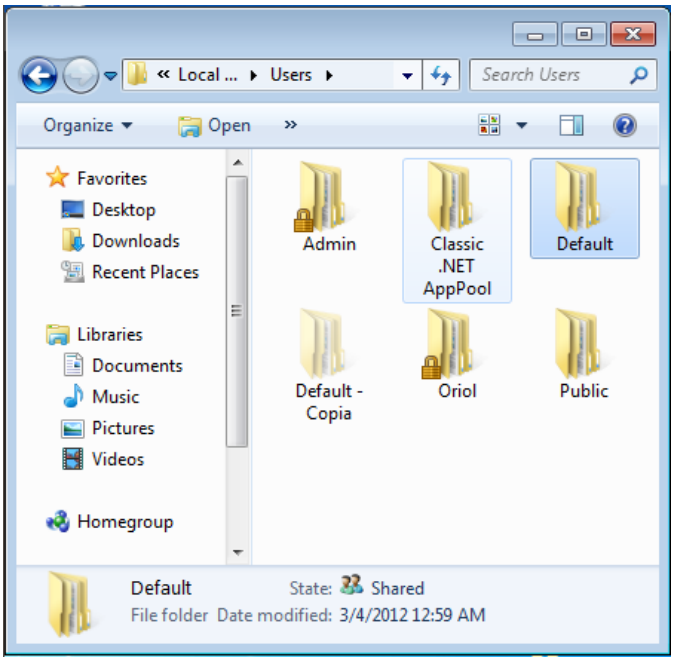
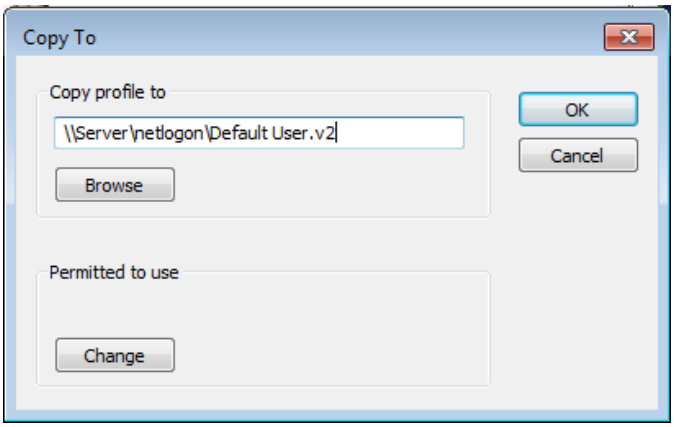


Figura 2.27. Còpia del perfil al servidor



Perfils mòbils

Gràcies als **perfils mòbils** els usuaris d'un domini poden iniciar sessió en qualsevol màquina de la mateixa xarxa i accedir als seus documents. Quan l'usuari es connecta, es recupera el seu perfil del servidor i se'n fa una còpia a la màquina local. Al finalitzar la sessió es desen els canvis del perfil al servidor. D'aquesta manera els usuaris tenen la sensació d'estar treballant sempre en el mateix equip.

Cal tenir en consideració la **càrrega extra que pateix la xarxa** quan es fan servir perfils mòbils, especialment en xarxes lentes o usuaris amb gran quantitat d'arxius al seu perfil. Per això, és recomanable que s'usi només en circumstàncies específiques.

Per habilitar l'ús dels perfils mòbils cal realitzar els passos següents:

1. Crear un recurs compartit on emmagatzemar els perfils.

2. Configurar els paràmetres adequats de Samba per habilitar els perfils mòbils.

És evident que si es vol disposar de perfils mòbils cal disposar d'un lloc en el servidor on emmagatzemar cadascun dels perfils dels usuaris. Per tant, el primer pas és crear un recurs compartit per emmagatzemar perfils. Típicament es fa servir el nom `[profiles]`, però en realitat pot anomenar-se com es desitgi.

Cal crear un directori on emmagatzemar els perfils, de manera que tothom tingui accés d'escriptura i només pugui modificar els seus arxius (*sticky bit*). En aquest cas es desitja emmagatzemar-los en un directori situat a l'arrel anomenat `perfils`.

```
# mkdir /perfils
# chmod 1777 /perfils
```

Aquest directori s'ha de compartir amb unes quantes consideracions de seguretat:

1. És recomanable que el recurs estigui ocult: *browseable = no*.
2. Els usuaris no han de poder entrar al perfil d'altres usuaris ni modificar-los: *create mask = 0600* i *directory mask = 0700*.
3. Els invitats no hi han de poder entrar: *guest ok = no*.
4. S'ha de permetre l'escriptura al recurs: *read only = no*.

Per tant:

```
[profiles]
     comment = Perfils d'usuari
     path = /perfils
     guest ok = no
     browseable = no
     create mask = 0600
     directory mask = 0700
     read only = no
```

Finalment, cal habilitar l'ús dels perfils mòbils amb el paràmetre *logon path* de la secció `[global]`. En aquest paràmetre hem d'indicar en quin lloc de la xarxa tenen els clients el seu perfil, especificant l'adreça UNC del recurs.

Els perfils són a l'adreça `\\<servidor>\profiles\`, i s'haurien d'emmagatzemar de manera separada per a cada usuari. És a dir, l'usuari *oriol* emmagatzemarà el seu perfil a `\\<servidor>\profiles\oriol` i l'usuari *lluis* a `\\<servidor>\profiles\lluis`.

Per tant, a la variable *logon path* posarem el següent:

```
[global]
     ...
     logon path = \\%N\profiles\%U
     ...
```

On *%N* substitueix el nom del servidor i *%U* substitueix el nom de l'usuari. Si s'hi connecta un usuari que no té perfil, el directori amb el seu nom es crearà automàticament.

Aquesta configuració té un problema: com heu vist, hi ha diferents tipus de perfil segons la versió de Windows, doncs un perfil de Windows XP no és compatible amb un de Windows 7. Llavors, **si en el domini hi ha diferents versions de Windows cal separar els perfils d'una família de la resta**. Amb la variable *%a* podem crear diferents directoris per als diferents tipus d'arquitectures.

```
[global]
     ...
     logon path = \\%N\profiles\%U\%a
     ...
```

L'exemple següent mostra el directori de perfils d'un usuari que ha fet servir diversos sistemes operatius per iniciar sessió al domini:

```
# ls -l /perfils/oriol/
total 8
drwx------ 2 oriol oriol 4096 mar 4 13:01 Vista.V2
drwx------ 13 oriol oriol 4096 mar 4 12:58 WinXP
```

Com veieu, quan s'habiliten els perfils mòbils cal tenir en consideració l'espai que poden arribar a ocupar al servidor i la sobrecàrrega de la xarxa.

Perfils obligatoris

Hi ha casos en què no és desitjable desar els canvis dels perfils dels usuaris. Els **perfils obligatoris**són la manera de forçar tots els usuaris a usar un perfil únic. El perfil es descarrega durant l'inici de sessió i tots els canvis realitzats són esborrats al acabar.

Per transformar un perfil normal en un perfil obligatori només cal modificar el nom de l'arxiu `NTUSER.DAT` per `NTUSER.MAN`. Aquest arxiu és a l'arrel del perfil.

```
# mv ntuser.dat ntuser.man
```

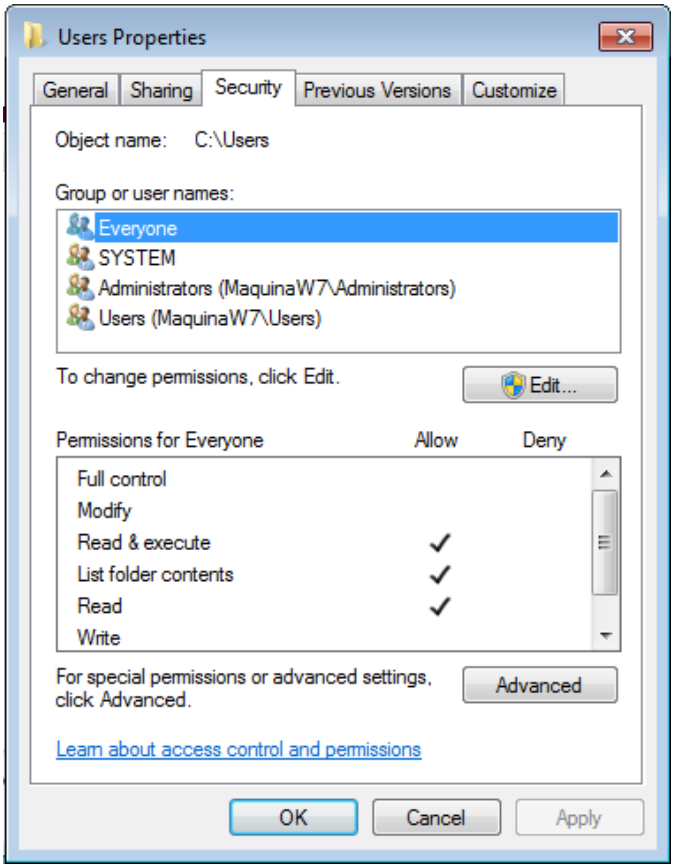
Així, doncs, si apliquem un perfil obligatori només és necessari tenir un sol perfil per usuari al servidor. Per tant, podem copiar-lo dins del recurs `profiles` i eliminar la variable del nom d'usuari del paràmetre *logon path*.

```
[global]
     ...
     logon path = \\%N\profiles\%a
     ...
```

2.2.5 Identificadors de seguretat i grups

El mecanisme de permisos de Windows està basat en les llistes de control d'accés (ACL). Com podeu veure en la figura 2.28, cada recurs (fitxer, directori, impressora, etc.) té associada una ACL en la qual es detallen el conjunt d'accions que estan autoritzades a certs usuaris o grups (llegir, escriure, esborrar...). El sistema és l'encarregat de decidir si un determinat usuari està autoritzat a fer l'acció que ha sol·licitat basant-se en l'ACL.

FIGURA 2.28. En Windows, els permisos es basen en les ACL i els identificadors de seguretat



A l'hora d'autoritzar o denegar una l'acció, Windows no ho determina fent servir el nom d'usuari o el nom del grup, sinó que, tal com ocorre a Unix, es fa servir un identificador. Aquest identificador rep el nom d'**identificador de seguretat** o **SID** i, depenent de l'àmbit, podem parlar de SID del domini o de SID local.

> A l'hora d'autoritzar o denegar una acció sobre un recurs, Windows no es basa en el nom d'usuari sinó en el seu identificador de seguretat (SID).

L'identificador de seguretat de Windows és un codi alfanumèric que **identifica usuaris o grups en un entorn local, o un objecte qualsevol dins d'un domini**: usuaris, grups, màquines, impressores, etcètera. Aquest codi és únic per cada a objecte i s'assigna en el moment de la seva creació. És similar a l'identificador d'usuari (UID) i de grup (GID) en un sistema GNU/Linux.

El mecanisme de seguretat basat en el SID també està implementat a Samba: tot usuari, grup i objecte dins del *backend* té associat un identificador de seguretat. Aquesta característica pren especial rellevància quan aquest actua com a controlador de domini.

En general, tant a Windows com a Samba, es pot distingir entre:

1. SID local
2. SID de domini

En el moment d'instal·lar un sistema operatiu Windows o iniciar per primera vegada el servei Samba en un sistema GNU/Linux, s'assigna aleatòriament un SID local que queda emmagatzemat al registre. A Windows, aquest SID només s'utilitza en un context local de cara als usuaris creats dins de la màquina, sobretot a l'hora de determinar els permisos d'accés a les particions de disc NTFS.

En el cas d'un equip que es promogui a controlador de domini, el SID local es copiarà per crear el SID de domini. En aquest cas,**el SID local serà idèntic al SID del domini** i és el que s'utilitzarà per determinar els permisos dins de l'àmbit domini.

Així, quan un usuari accedeix a un domini, en el procés de *login* el controlador li atorga una espècie de carnet d'identitat anomenat *token*. Aquest conté el SID corresponent al seu usuari i els de tots els grups als quals pertany. Quan aquesta persona vol accedir a un recurs, com ara un directori compartit, es verifica el seu *token* juntament amb l'ACL del recurs i s'autoritza o es denega l'acció que vol realitzar.

El SID local o de domini s'acostuma a representar com una cadena de caràcters separats per guions, com mostra l'exemple:

```
1 S-1-5-21-2919073133-884734282-1099352923-1104
```

Sense entrar en detall en el significat de cadascun dels caràcters, podem dividir un SID en dues parts mostrades en la figura 34. La primera, que comprèn des del primer caràcter fins a l'últim guió, conté l'identificador del domini (en l'exemple és `S-1-5-21-2919073133-884734282-1099352923`). **Aquest valor és el mateix per a tots els objectes dins d'un domini**. És a dir, els SID de dos usuari, grups o màquines pertanyents al mateix domini contenen aquests mateixos caràcters. La part restant del SID és l'identificador relatiu, o RID, i identifica l'objecte dins del domini. En l'exemple de la figura 2.29, el RID és `1104`. Aquest nombre **mai es repeteix dins d'un mateix domini**.

FIGURA 2.29. SID del domini i RID de l'objecte

Samba implementa un mecanisme per emmagatzemar identificadors de seguretat SID de manera similar a Windows. Podem obtenir el SID local o de domini mitjançant l'ordre *net*, amb les opcions *getdomainsid* o *getlocalsid*, respectivament.

```
# net getdomainsid
SID for local machine DEBIAN is: S-1-5-21-2862684848-1588976619-3858101340
SID for domain DOMINI is: S-1-5-21-2919073133-884734282-1099352923
```

Quan Samba és un controlador de domini es pot observar que el SID local i el SID de domini es corresponen.

Fent servir l'ordre *pdbedit* podem consultar el SID d'un usuari, així com el SID del seu grup principal.

```
# pdbedit -Lv lluis
Unix username: lluis
NT username:
Account Flags: [U ]
User SID: S-1-5-21-2862684848-1588976619-3858101340-1000
Primary Group SID: S-1-5-21-2862684848-1588976619-3858101340-513
Full Name:
Home Directory: \\debian\lluis
HomeDir Drive:
Logon Script:
Profile Path: \\debian\lluis\profile
Domain: DEBIAN
Account desc:
Workstations:
Munged dial:
Logon time: 0
Logoff time: never
Kickoff time: never
Password last set: sáb, 12 nov 2011 18:31:22 CET
Password can change: sáb, 12 nov 2011 18:31:22 CET
Password must change: never
Last bad password : 0
Bad password count : 0
Logon hours : FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
```

En aquest exemple podem observar que el SID del domini és:

```
S-1-5-21-2862684848-1588976619-3858101340
```

i els RID de l'usuari i grup principal són 1000 i 513, respectivament. Com veureu a continuació, l'elecció d'aquests valors no és aleatòria.

En un domini Windows, existeixen una sèrie d'identificadors de seguretat associats a usuaris o grups que es creen per defecte cada vegada que es promou un controlador de domini. Aquests identificadors, resumits a la taula 2.5, tenen un RID inferior a 1000, al contrari dels grups o usuaris creats a posteriori, que tenen un RID igual o superior a 1000.

TAULA 2.5. RID del grups coneguts

RID	Descripció
500	Compte d'usuari per a l'administrador del sistema. Per defecte, és l'únic usuari que té control total.
501	Compte d'usuari convidat, sense contrasenya. Deshabilitat per defecte.

TAULA 2.5 (continuació)

RID	Descripció
512	Administradors del domini. Grup que permet als seus membres administrar el domini.
513	Usuaris del domini. Per defecte hi pertanyen tots els usuaris creats en el domini.
514	Convidats del domini.
515	Equips del domini.
544	Administradors.
548	Operadors de comptes.
550	Operadors d'impressió.
551	Operadors de còpies de seguretat.

Podeu consultar la llista d'identificadors més coneguts al web http://support.microsoft.com/kb/243330/es.

Samba permet associar SID dels grups de Windows a GID (identificador de grup) de Linux. El paràmetre *groupmap* de l'ordre *net* es pot fer servir per determinar aquestes associacions.

Per crear grups en un domini cal establir una associació entre el grup del domini Windows, identificat pel SID, i el grup de Linux.

Per defecte, Samba no crea cap mena d'associació ni grup. És a dir, tots aquells grups que per defecte es creen en un controlador de domini Windows no hi són i els heu d'afegir manualment (si els necessiteu). Per exemple, si voleu crear el grup d'administradors del domini, identificat pel RID 512, haureu de crear un grup Linux i vincular-los.

```
# groupadd admins
# net groupmap add rid=512 unixgroup=admins ntgroup="Administradors domini"
    comment="grup d'administradors del domini"
Successfully added group Administradors domini to the mapping db as a domain
    group
```

Crear l'associació del grup d'administradors és molt important, ja que un cop s'ha fet, **automàticament tots aquells usuaris que assignem al grup `admins` seràn administradors del domini amb tots els drets que això comporta**.

Així, doncs, si volem que l'usuari *oriol* sigui un administrador del domini només caldrà afegir-lo al grup *admins*:

```
# usermod -G admins oriol
```

A partir de l'assignació, l'usuari *oriol* podrà administrar màquines, impressores, usuaris, permisos, etcètera.

En general, però, podeu crear qualsevol grup i anomenar-lo com vulgueu; no esteu limitats a crear grups per defecte. Així, doncs, podríem crear grups per a professors i alumnes, tal com es mostra a continuació:

```
# groupadd professors
# groupadd alumnes
```

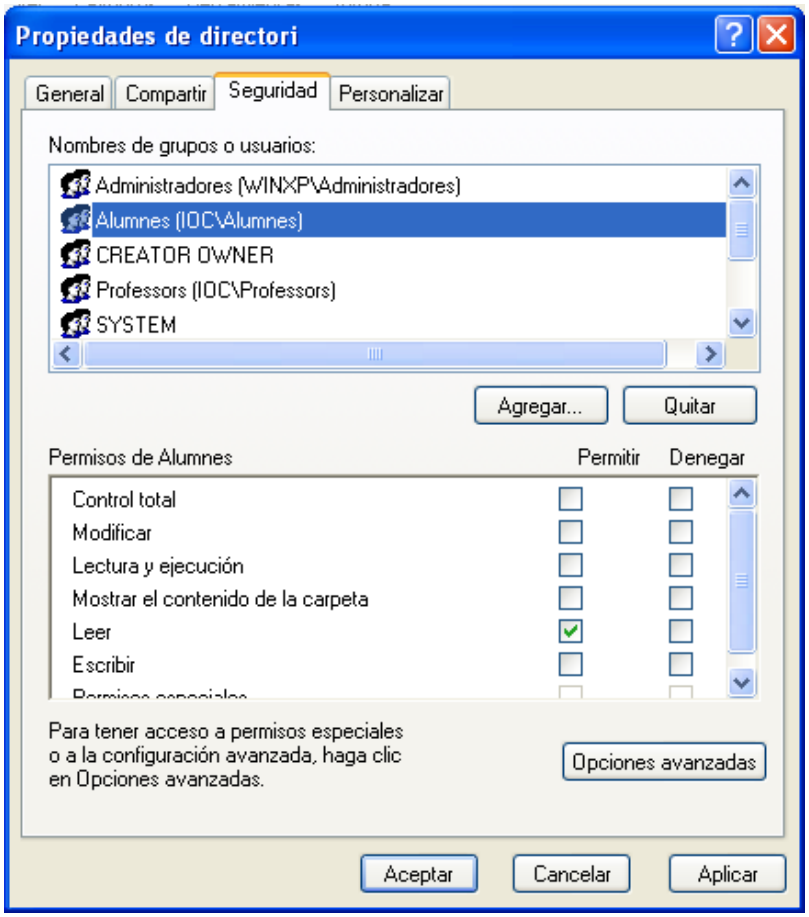
```
# net groupmap add rid=600 unixgroup=professors ntgroup="Professors"
Successfully added group Professors to the mapping db as a domain group
# net groupmap add rid=601 unixgroup=alumnes ntgroup="Alumnes"
Successfully added group Alumnes to the mapping db as a domain group
```

Llistant els grups veureu que la vinculació s'ha creat correctament:

```
# net groupmap list
Administradors domini (S-1-5-21-2768477417-3040658552-3227845813-512) -> admins
Alumnat (S-1-5-21-2768477417-3040658552-3227845813-601) -> alumnes
Professorat (S-1-5-21-2768477417-3040658552-3227845813-600) -> professors
```

Tots aquests grups, tal com podeu veure a la figura 2.30, apareixen creats correctament dins del domini.

FIGURA 2.30. Grups d'usuaris creats



Els vincles s'emmagatzemen en una base de dades, que per defecte és dins de l'arxiu `/var/lib/samba/group_mapping.ldb` i no al *backend*. Podeu afegir o treure usuaris dels grups fent servir l'ordre de Linux *usermod*.

```
# usermod -G alumnat -a usuari
```

Per a més informació sobre la manipulació de grups escriviu

```
# net groupmap
```

i obtindreu la llista d'opcions que es poden realitzar.

2.2.6 Drets

L'assignació de drets als usuaris permet alliberar la càrrega de l'usuari *root*. Fins fa poc, a Samba el superusuari era l'únic que podia realitzar algunes accions d'administració. En les darreres versions de Samba podem atorgar certes accions a usuaris o grups. Per defecte, Samba no assigna cap dret als usuaris. La taula 2.6 mostra el nom i la descripció de tots ells.

Taula 2.6. Drets d'usuari

Dret	Descripció
`SeMachineAccountPrivilege`	Afegir màquines al domini.
`SePrintOperatorPrivilege`	Administrar impressores.
`SeAddUsersPrivilege`	Afegir usuaris i grups al domini.
`SeRemoteShutdownPrivilege`	Forçar l'apagada des d'una màquina remota.
`SeDiskOperatorPrivilege`	Administrar un recurs compartit.
`SeTakeOwnershipPrivilege`	Prendre possessió d'un fitxer o directori del domini independentment del seu propietari.
`SeRestorePrivilege`	Assignar el propietari d'un fitxer o directori a un altre usuari.

Per defecte, només un administrador del domini pot assignar o treure els drets d'un usuari.

L'eina que es fa servir per administrar els drets és*net rpc rights*. Aquesta ordre permet assignar (*grant*), treure (*revoke*) o llistar (*list*) els drets en un servidor.

Per llistar tots els drets que permet assignar el servidor:

```
# net -S localhost -U root rpc rights list
Enter root's password:
     SeMachineAccountPrivilege
      SeTakeOwnershipPrivilege
             SeBackupPrivilege
            SeRestorePrivilege
     SeRemoteShutdownPrivilege
      SePrintOperatorPrivilege
           SeAddUsersPrivilege
       SeDiskOperatorPrivilege
```

Les opcions *-S* i *-U* indiquen el servidor i l'usuari que fem servir per realitzar l'acció. En aquest cas fem servir l'usuari arrel, però pot ser qualsevol administrador del domini. Si voleu assignar o treure un determinat dret a un usuari hem d'indicar el domini, el nom i el privilegi:

```
# net -S localhost -U root rpc rights grant 'IOC\usuari'
    SeDiskOperatorPrivilege
# net -S localhost -U root rpc rights revoke 'IOC\usuari'
    SeDiskOperatorPrivilege
```

Administració del servei de directori

Andrés Pérez Payeras i Miquel Tarazona Belenguer

Administració de sistemes operatius

Índex

Introducció

Poques vegades els usuaris de sistemes informàtics ens preguntem, quan escrivim un nom d'usuari i una contrasenya, on està emmagatzemada aquesta informació o com s'hi accedeix. Una característica desitjable per a qualsevol sistema és no duplicar informació per tal de facilitar-ne el manteniment. Emmagatzemar la informació de l'organització en un directori i que aquest sigui el punt central on consultar-la sembla una bona opció.

En aquesta unitat s'estudien els conceptes requerits per poder administrar correctament un servei de directori, des de la part teòrica necessària per comprendre el seu funcionament fins a la descripció d'exemples pràctics, similars als que ens podem trobar a la vida real, per mostrar la part més procedimental dels continguts.

En l'apartat "El servei de directori" es tracta la part més teòrica de la unitat. S'hi descriuen tots els conceptes necessaris per entendre els apartats posteriors. Aquest apartat presenta els fonaments dels directoris i el protocol més utilitzat per accedir-hi, l'LDAP.

En l'apartat "Instal·lació, configuració i manteniment del servei de directori" es tracta una implementació específica del protocol LDAP, concretament l'OpenLDAP. Primerament s'expliquen les característiques bàsiques de la implementació triada per veure'n després la instal·lació, configuració i manteniment tot centrant-nos en un supòsit concret per ajudar a assimilar els conceptes que s'expliquen. S'ha triat l'OpenLDAP perquè és una implementació lliure però àmpliament estesa i que disposa de suport comercial.

En l'apartat "Integració del servei de directori" es treballa un dels aspectes més utilitzats i funcionals de l'ús d'un servei de directori com a dipòsit on emmagatzemar la informació: l'autenticació centralitzada. Aquesta autenticació centralitzada facilita enormement la tasca de manteniment d'usuaris i contrasenyes en un sistema informàtic en el qual requereixen autenticació diverses aplicacions. L'apartat cobreix la integració del directori per fer l'autenticació amb un sistema operatiu client, sigui Windows o Linux; amb un servei, prenent com a exemple un servidor FTP; i amb una aplicació web, prenent com a exemple un gestor de contingut.

Els continguts d'aquesta unitat tenen una orientació bàsicament pràctica. Es proposa una formació pràctica i aplicada amb l'objectiu que l'alumnat aprengui i interioritzi els conceptes fent les coses. És molt recomanable llegir els continguts de cada apartat i anar realitzant els exemples proposats, així com els exercicis i les activitats del web de la unitat, per tal de posar en pràctica i comprovar els coneixements adquirits.

Resultats d'aprenentatge

En acabar aquesta unitat, l'alumne:

1. Administra el servei de directori interpretant especificacions i integrant-lo en una xarxa.
    - Identifica la funció, els elements i les estructures lògiques del servei de directori.
    - Determina i crea l'esquema del servei de directori.
    - Realitza la instal·lació del servei de directori al servidor.
    - Realitza la configuració i personalització del servei de directori.
    - Integra el servei de directori amb altres serveis.
    - Aplica filtres de cerca en el servei de directori.
    - Utilitza el servei de directori com a mecanisme d'acreditació centralitzada dels usuaris en una xarxa.
    - Realitza la configuració del client per a la seva integració en el servei de directori.
    - Utilitza eines gràfiques i comandaments per a l'administració del servei de directori.
    - Documenta l'estructura i implantació del servei de directori.

1. El servei de directori

Avui dia les persones i empreses confien en els sistemes informàtics connectats en xarxa per utilitzar aplicacions distribuïdes. La informació de la xarxa pot ser recollida en una base de dades especial que es diu *directori*. Un directori s'utilitza habitualment per emmagatzemar la informació dels usuaris, com ara el nom de l'usuari i la contrasenya. També es possible emmagatzemar-hi informació com les dades de contacte dels usuaris (directori d'una empresa) i també s'utilitza sovint per inventariar recursos (inventari de màquines, impressores, servidors i altres recursos de xarxa).

A mesura que el nombre de xarxes i aplicacions creix, el nombre de directoris especialitzats d'informació també augmenta, cosa que dóna lloc a "illes" d'informació que són difícils de compartir i gestionar. El Lightweight Directory Access Protocol (LDAP) és un estàndard obert que ha evolucionat per proporcionar un mètode estàndard per accedir a la informació en un directori i actualitzar-la. L'LDAP ha obtingut una gran acceptació com a mètode d'accés a directoris dins de les xarxes corporatives. Està recolzat per un gran nombre de proveïdors de programari i ha estat incorporat a un nombre creixent d'aplicacions.

1.1 Servei de directori

Un directori és un llistat d'informació sobre uns objectes disposats en un ordre que dóna detalls sobre cada objecte. Exemples comuns de directoris són una guia telefònica d'una ciutat o un catàleg d'una biblioteca. En una guia telefònica, els objectes de la llista són les persones, els noms estan ordenats alfabèticament i els detalls guardats sobre cada persona són la direcció i el número de telèfon. Els llibres d'un catàleg d'una biblioteca estan ordenats per autor o per títol, i se'n guarda informació com l'ISBN o altres característiques de la publicació.

En termes informàtics, un **directori** és una base de dades especialitzada (també anomenada *repositori de dades*) que emmagatzema informació sobre objectes i que està específicament dissenyada per optimitzar les operacions de consulta. Un **servei de directori** és el programari que emmagatzema, organitza i facilita l'accés a la informació d'un directori.

Per exemple, un directori pot incloure informació sobre les impressores (els objectes), que consistirà en dades com ara la ubicació (una cadena de caràcters), la velocitat en pàgines per minut (numèric), els fluxos d'impressió compatibles (per exemple, PostScript o ASCII) i així successivament.

Els directoris permeten als usuaris o a les aplicacions trobar informació que compleixi unes característiques determinades. Per exemple, un directori es pot utilitzar per buscar una adreça de correu electrònic d'una persona concreta, per trobar una impressora en color propera, per consultar la informació de facturació d'un client específic...

En un directori es pressuposa un nombre molt més alt de lectures que d'escriptures, ja que generalment la informació continguda canvia rarament (per exemple el nombre de telèfon d'una persona). Aquest aspecte és important i el diferencia d'una base de dades de propòsit general, en la qual les operacions són tant de lectura com d'escriptura. En el disseny d'un directori, els esforços d'optimització es concentren en les cerques i les lectures, i no és cap problema que això penalitzi les actualitzacions.

Els directoris poden contenir molts tipus d'informació i estendre la seva utilitat a diverses aplicacions, però complint sempre aquestes característiques:

1. La informació que contenen està basada en objectes (cada objecte és una entrada del directori) i en els atributs d'aquests objectes.
2. Les actualitzacions i modificacions de les dades són simples i no gaire freqüents.
3. Estan optimitzats per donar una resposta ràpida a operacions de cerca i revisió.
4. Poden replicar la informació per augmentar-ne la disponibilitat i la fiabilitat alhora que disminueix el temps de resposta a les consultes.

Alguns exemples d'ús de directori són:

1. Autenticació d'usuaris.
2. Autenticació de màquines.
3. Llibreta d'adreces.
4. Representació de l'estructura de l'organització.
5. Emmagatzematge d'informació telefònica.
6. Cerca d'adreces de correu electrònic.

1.1.1 Clients i servidors de directori

API i protocol

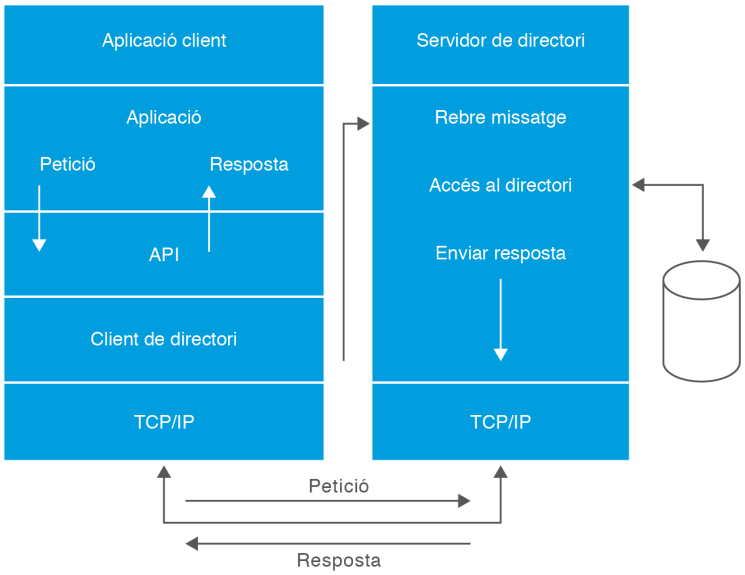
Una API defineix la interfície de programació que un llenguatge de programació particular utilitza per accedir a un servei. El format i contingut dels missatges intercanviats entre el client i el servidor s'han d'adherir al protocol acordat.

Els serveis de directori solen implementar-se seguint el model client-servidor, de manera que una aplicació que vol accedir al directori no accedeix directament a la base de dades, sinó que, tal com mostra la figura 1.1, crida a una funció de l'API (Application Programming Interface) proporcionada pel servei, que envia

un missatge a un procés del servidor. Aquest procés accedeix al directori i retorna el resultat de l'operació.

Algunes vegades, el servidor pot convertir-se en el client d'un altre servidor per aconseguir la informació necessària per processar la petició que se li ha realitzat.

FIGURA 1.1. Arquitectura client-servidor del directori



Seguint aquest model, el client no depèn de l'arquitectura del servidor i el servidor pot implementar el directori de la manera més convenient.

El servidor del directori i el client han d'implementar un protocol comú per comunicar-se i és aquest el que caracteritza les diferents solucions que s'han desenvolupat. El **Lightweight Directory Access Protocol** (**LDAP**) facilita serveis centralitzats i distribuïts d'informació mitjançant una xarxa TCP/IP i s'ha convertit en l'estàndard de facto d'accés a la informació emmagatzemada en un servidor de directori per a usuaris i aplicacions.

1.1.2 Avantatges dels serveis de directori

Un directori específic d'una aplicació només emmagatzema la informació necessària per a aquella aplicació en particular i no és accessible per altres aplicacions. La mateixa adreça de correu electrònic emmagatzemada per una aplicació de calendari també pot ser emmagatzemada per una aplicació de correu electrònic i per una aplicació que avisa els operadors del sistema de problemes en els equips. Cada aplicació crea i administra el seu propi directori.

En un entorn on només una aplicació ha d'accedir a les dades, potser l'esforç necessari per implantar un servei de directori electrònic estàndard és innecessari, però l'experiència demostra que tard o d'hora diverses aplicacions acaben utilit-

zant aquestes dades, i en el cas de no haver implantat un directori centralitzat ens trobem amb diversos directoris que han d'estar sincronitzats i que accedeixen de maneres diferents a les dades contingudes en directoris creats a mida. Això implica un major esforç per fer el manteniment i un fre al desenvolupament d'aplicacions basades en el directori.

El fet de disposar d'un servei de directori comú, multiplataforma, accessible mitjançant un protocol estàndard i amb una API estàndard, permet als programadors desenvolupar les aplicacions sense haver de crear directoris específics.

Des del punt de vista de l'administració de sistemes, configurar el sistema perquè les aplicacions utilitzin un servei de directori comú, dissenyat de manera adequada, permet controlar més fàcilment els riscos de fallada per inconsistència de dades i concentrar els esforços en millorar l'administració i la tolerància a fallades del servei.

1.2 El protocol LDAP

LDAP són les inicials de Lightweight Directory Access Protocol (protocol lleuger d'accés a directoris). Com el seu nom indica, és un protocol pensat per permetre l'accés a serveis de directori.

L'LDAP funciona amb un esquema client-servidor, en el qual un (o diversos) servidors mantenen la mateixa informació de directori (actualitzada mitjançant rèpliques) i els clients fan consultes a qualsevol d'ells.

A l'LDAP la informació està estructurada en forma d'arbre. Aquesta estructura s'anomena *directori* i cada node s'anomena *entrada*. Per la seva part, cada entrada està formada per un conjunt d'atributs, cadascun dels quals és d'un tipus i conté un o més valors.

1.2.1 Origen de l'LDAP

El 1984, la UIT-T (llavors anomenada CCITT) va decidir desenvolupar una especificació de directori de propòsit general. La necessitat més immediata era proporcionar un directori per a la gestió de l'estàndard X.400 de missatges de correu electrònic. Al mateix temps, l'ISO/IEC JTC1 va iniciar una activitat similar. Les dues organitzacions van acordar fusionar els dos treballs en un de sol per evitar la producció de dues normes diferents per al mateix propòsit. Aquesta col·laboració va donar com a resultat la norma X.500. Aquesta norma és un conjunt d'estàndards de xarxes d'ordinadors per a serveis de directori. La norma defineix un protocol anomenat DAP (Directory Access Protocol o protocol d'accés a directoris).

Organitzacions relacionades amb el protocol LDAP

UIT-T és el Sector de Normalització de la Unió Internacional de Telecomunicacions.

CCITT és l'acrònim de Comitè Consultiu Internacional Telegràfic i Telefònic.

ISO és l'acrònim de la International Organization for Standardization i **IEC** ho és de la International Electrotechnical Commission. **ISO/IEC** treballen conjuntament en la preparació, adopció i utilització de diferents estàndards.

ISO/IEC JTC1 és el comitè tècnic que treballa en l'estandardització dins del camp de les tecnologies de la informació.

El DAP és un protocol molt complex (i d'elevat cost computacional), ja que està definit sobre la pila completa del model OSI (*Open System Interconnection*, en català 'Interconnexió de Sistemes Oberts'). L'LDAP és un protocol més lleuger, gairebé equivalent al DAP, més senzill i eficient, ja que està dissenyat per funcionar directament per TCP/IP.

L'LDAP és un subconjunt del DAP que s'ha desenvolupat per al seu ús amb TCP/IP, de manera que hereta els conceptes fonamentals de la norma X.500, però simplifica la comunicació entre el servidor i el client i no inclou les característiques menys utilitzades del DAP.

L'especificació LDAP es defineix a l'RFC 4510 de l'Internet Engineering Task Force (IETF).

1.2.2 Característiques de l'LDAP

Existeixen altres tecnologies que ofereixen informació a un client, però els serveis de directori que implementen el protocol LDAP ofereixen alguns avantatges significatius:

1. **Consulta ràpida de dades**: l'objectiu d'un servidor de directori és la recuperació ràpida d'informació. L'LDAP ofereix un mètode lleuger de connexió a un dipòsit, lectura de dades i tancament de la connexió.
2. **Solució distribuïda**: com que l'LDAP és un protocol natiu de xarxa d'arquitectura distribuïda, pot distribuir a tota la xarxa informació llesta per ser usada per totes les aplicacions. Fins i tot pot reproduir una part del directori per donar un servei amb certes característiques en un entorn determinat. O moure (delegar) la informació a diversos llocs sense afectar cap accés extern a aquestes dades.
3. **Independència de la plataforma**: com que l'LDAP és un protocol estàndard, que proporciona un mètode d'accés a dades remot i local, és possible intercanviar completament la implementació de l'LDAP sense afectar la forma en que es podrà accedir a les dades. Hi ha una àmplia gamma de implementacions. Per suposat, el client i el servidor es poden executar en sistemes operatius diferents.

4. **Seguretat integrada en el repositori**: la informació d'accés s'emmagatzema en el mateix repositori. Alguns productes poden treballar amb drets i permisos de granularitat fina i també proporcionen la capacitat de controlar els drets d'accés en funció de factors externs, com ara l'adreça IP del client. Com que aquests controls s'executen de manera centralitzada pel servidor, són fàcils de mantenir. Els clients, per tant, no s'han de preocupar d'aquests detalls.

5. **Implementació fàcil del client**: la disponibilitat d'interfícies de programació (API) de l'LDAP per a gairebé qualsevol llenguatge de programació facilita la compatibilitat de l'LDAP amb pràcticament totes les aplicacions.

6. **Gran difusió**: com que l'LDAP és un protocol estàndard, molts proveïdors l'han adoptat al seu programari. Per exemple, IBM Tivoli, Microsoft Active Directory, Novell eDirectory o RedHat Directory Server.

7. **Baix cost**: l'LDAP és un protocol estàndard i no cal pagar llicències pel seu ús ni per disposar de clients o servidors (de codi font obert).

Característiques de l'LDAPv3

L'LDAPv2 es va dissenyar amb l'objectiu de ser implementat amb facilitat. Encara que ja realitza les principals operacions, va ser substituït per l'LDAPv3 perquè tenia algunes limitacions. Actualment només es recomana la compatibilitat amb la versió 3 de l'LDAP.

Algunes de les característiques particulars de l'LDAPv3 són:

1. La majoria dels elements de dades del protocol es poden codificar com a cadenes normals.

2. Els valors dels atributs i els noms distintius s'han internacionalitzat mitjançant l'ús del conjunt de caràcters ISO 10646 (UTF-8).

3. El protocol pot ser ampliat per permetre noves operacions.

4. El resultat retornat poden ser referències. Això significa que un servidor de directoris que no contingui les dades sol·licitades les pot demanar a un altre servidor de directori o pot informar el client d'on les pot trobar.

5. Els clients poden demanar l'esquema que descriu les estructures de dades disponibles al directori.

6. Hi ha atributs operacionals (amb finalitats administratives) gestionats pel servidor.

7. Disposa de protecció per contrasenya, ús de certificats digitals i accés SSL.

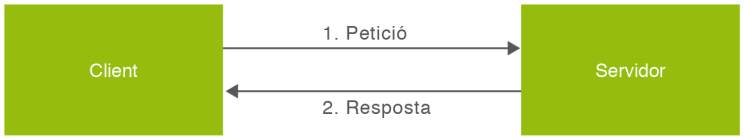
8. Els mecanismes SASL (capa d'autenticació i seguretat) es poden utilitzar amb l'LDAP per proporcionar serveis de seguretat.

1.2.3 Funcionament de l'LDAP

En el model client-servidor de l'LDAP, davant d'una consulta concreta d'un client, el servidor contesta amb la informació sol·licitada. La resposta pot ser, de forma alternativa (o a més de la informació que s'havia sol·licitat), un punter que indica on aconseguir aquesta informació o dades addicionals (habitualment, el punter apunta a un altre servidor de directori).

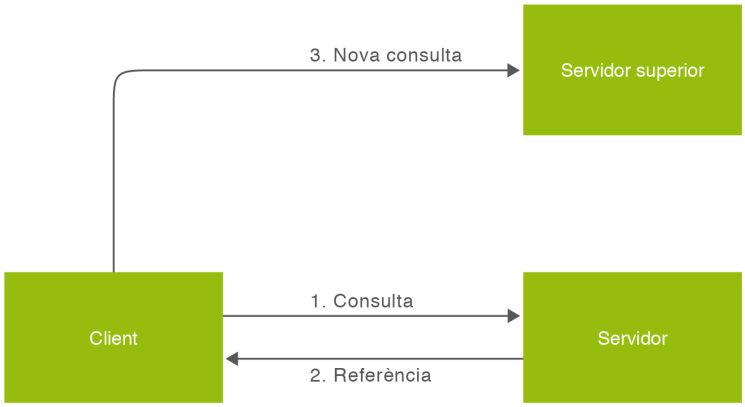
Aquesta configuració client-servidor es pot fer amb un únic servidor, com es pot veure en la figura 1.2. Amb aquest funcionament es proveeix el servei de directori per a una sola organització.

FIGURA 1.2. L'LDAP com a servei local



Es pot decidir separar el directori entre diversos servidors per motius organitzatius o per facilitar-ne la gestió. Es aquest cas es poden delegar parts del directori a un altre servidor, com s'observa en la figura 1.3. Aquesta configuració també es fa servir si es vol que el directori formi part d'un directori global. El servidor està configurat per tornar referències a altres servidors LDAP en el cas que se li demani informació de la qual no disposa, però que sap on aconseguir.

FIGURA 1.3. LDAP amb delegació

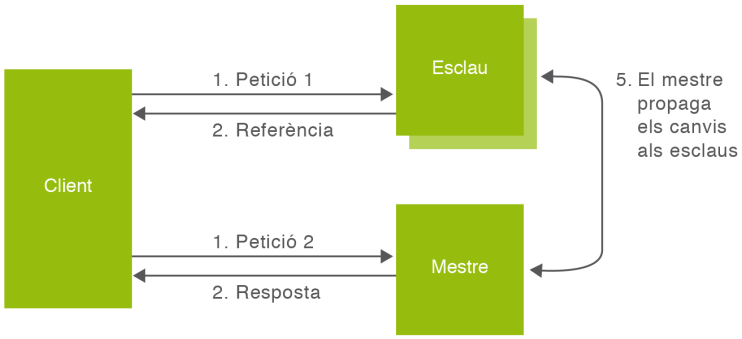


Es pot augmentar la disponibilitat i la fiabilitat del directori fent servir més d'un servidor LDAP per mantenir la informació. D'altra banda, durant un temps limitat pot mancar sincronització entre mestre i esclaus.

L'actualització de dades es produeix a nivell d'entrada i no d'atribut, de manera que si es modifica un atribut, el mestre envia als esclaus tota l'entrada, cosa que és problemàtica quan hi ha entrades grans o modificacions freqüents.

A més de la configuració mestre-esclau com la que mostra la figura 1.4, hi ha implementacions amb replicació multimestre (les actualitzacions es repliquen a diversos nodes que poden actuar com a mestre).

FIGURA 1.4. LDAP amb replicació



1.2.4 Missatges LDAP

L'LDAP s'implementa directament a TCP/IP i fa servir cadenes simples per transportar les dades. És un protocol orientat a missatges, és a dir, tan bon punt arriba un missatge al receptor, l'operació de recepció ha acabat, perquè ja s'ha rebut tota la informació referent a aquell missatge.

Tots els missatges d'anada i tornada entre el servidor i el client —les peticions que el client envia al servidor, els resultats que el servidor envia al client i els codis de resultat—viatgen en aquest format.

El procés de negociació és el següent:

1. El client LDAP envia una sol·licitud al servidor LDAP.
2. El servidor executa una o diverses operacions (segons el que s'especifiqui a la sol·licitud).
3. En el cas d'una transmissió correcta, el servidor envia els resultats. En cas d'error s'envia al client el codi d'error corresponent.

Cada missatge LDAP, tant si és una petició com una resposta, conté la identificació del missatge, un codi d'operació i les dades. L'identificador del missatge es necessita per determinar la sol·licitud a la qual correspon una resposta, dada molt important, perquè el mode l'LDAP no requereix que les sol·licituds i respostes es facin de forma síncrona.

1.3 Els models LDAP

L'LDAP és un estàndard i no pas un maquinari o programari que es pot comprar. El que s'instal·la en l'equip client o servidor és la implementació d'aquest protocol; la qüestió de com emmagatzemar o tractar les dades es deixa als proveïdors de l'aplicació de la norma final.

Implementacions del protocol LDAP

Existeixen diverses implementacions del protocol LDAP realitzades per diferents companyies, entre d'altres:

- Active Directori: és la implementació de Microsoft en els seus sistemes operatius Windows Server.
- RedHat Directory Server o 389 Directory Server: una implementació realitzada per RedHat/Fedora.
- ApacheDS: un servei de directori que ofereix l'Apache Software Foundation.
- OpenDS: una implementació Java del protocol LDAP.
- OpenLDAP: una implementació lliure de l'estàndard.

Tot i la llibertat d'implementació, el sistema pot caracteritzar-se segons algun dels quatre models següents:

1. El **model d'informació** descriu l'estructura de la informació emmagatzemada en el directori LDAP.
2. El **model de noms** descriu com s'organitza i identifica la informació en el directori LDAP.
3. El **model funcional** descriu quines operacions poden ser realitzades amb la informació emmagatzemada en el directori LDAP.
4. El **model de seguretat** descriu com es pot protegir la informació continguda en el directori LDAP davant d'intents d'accés no autoritzats.

1.3.1 Model d'informació

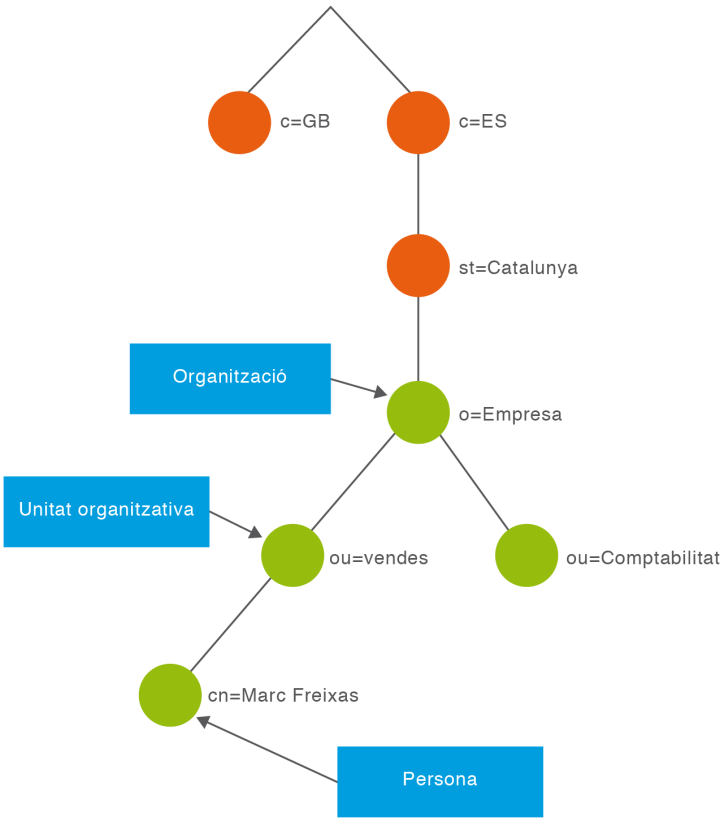
El model d'informació defineix quin tipus d'informació es pot emmagatzemar en el directori i les unitats bàsiques en què l'LDAP estructura la informació.

DIT

A l'LDAP la informació està estructurada en forma d'arbre. Aquest arbre s'anomena **arbre d'informació del directori** o **DIT** (Directory Information Tree).

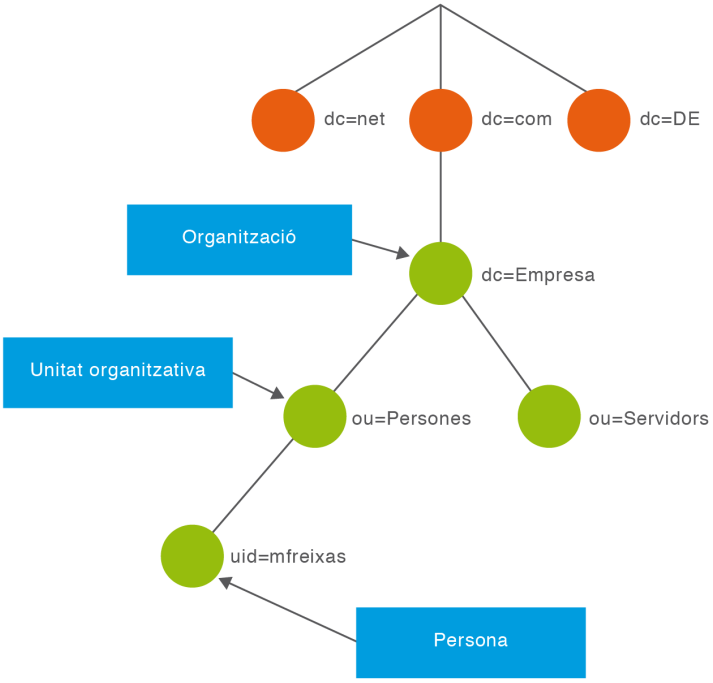
Tradicionalment, aquesta estructura ha reflectit límits geogràfics i organitzatius. A la part superior de l'arbre s'han representat els països, i a sota, els estats, les organitzacions , les unitats organitzatives, les persones, els ordinadors, les impressores, els documents o altres conceptes. Aquesta estructura es mostra en la figura 1.5.

FIGURA 1.5. Arbre de directori LDAP amb nomenclatura tradicional



L'arbre del directori també es pot estructurar en funció dels noms de domini d'Internet (DNS). Aquesta pràctica està cada vegada més estesa, ja que permet trobar un servidor LDAP realitzant una consulta DNS. En la figura 1.6 es pot veure un exemple d'arbre de directori amb una organització basada en el DNS. Aquest tipus de representació és el més habitual a dia d'avui a l'hora d'implementar serveis de directori per administrar dominis informàtics.

Figura 1.6. Arbre de directori LDAP amb nomenclatura DNS



Entrada (entry)

Cada node de l'arbre s'anomena *entrada*. Una entrada és una col·lecció d'atributs identificada per un **nom distintiu** (*distinguished name* o **DN**). Aquest DN és únic al directori i fa referència a l'entrada de manera unívoca (de la mateixa manera que un identificador caracteritza de manera unívoca un registre en una base de dades relacional).

El DN de cada entrada del directori es representa mitjançant una cadena de caràcters formada per parells de la forma:

```
<tipus_atribut>=<valor>[,<tipus_atribut>=<valor>]
```

Aquest nom representa la ruta des de la posició de l'entrada fins a l'arrel de l'arbre (*base*, segons la nomenclatura LDAP). Com que se suposa que un directori es fa servir per emmagatzemar els objectes d'una organització determinada, la base serà la ubicació d'aquesta organització. Així, la base es converteix en el sufix de totes les entrades del directori.

Cada entrada té una entrada pare (*parent*) i pot tenir o no entrades fills (*child*). Cada entrada fill és un germà (*sibling*) de les altres entrades fills del mateix pare.

Exemples de noms distintius

En l'arbre de la figura 5 la base seria "o=Empresa,st=Catalunya,c=ES" i el DN de la persona representada seria "cn=Marc Freixas,ou=Vendes,o=Empresa,st=Catalunya,c=ES".

En l'arbre de la figura 6 la base seria "dc=Empresa,dc=com" i el DN de la persona representada seria "uid=mfreixas,ou=Persones,dc=Empresa,dc=com".

Atributs (attributes)

Les entrades estan formades per un conjunt d'atributs. Cada atribut d'una entrada té un tipus i un valor (SINGLE-VALUE) o més (MULTI-VALUE, que és el comportament predeterminat). El tipus d'atribut té associat els valors permesos i el seu comportament a l'hora de fer comparacions, ordenacions o treballar amb subcadenes.

Els tipus són habitualment cadenes de text mnemòniques, com poden ser *cn* per fer referència a *common name* (nom comú), *sn* a*surname* (cognom) o *mail* a *email address* (adreça de correu electrònic).

La sintaxis dels valors dependrà del tipus de l'atribut, per exemple, l'atribut *cn* podria contenir el valor "Marc Freixas", l'atribut *mail* podria contenir un valor com "mailto:mfreixas@empresa.com" i un atribut *jpegPhoto* contindria una imatge en format binari.

Classes d'objecte (objectClass)

Els atributs que conté una entrada estan condicionats per les classes d'objecte a les quals pertany. Una classe d'objecte agrupa conjunts d'atributs que poden ser opcionals (indicat amb MAY) o obligatoris (indicat amb MUST) i la unió de tots els atributs de totes les classes d'objecte de les quals forma part una entrada són els atributs permesos per aquesta entrada.

La classe d'objecte es reconeix amb un identificador (OID) i opcionalment el nom (NAME).

L'atribut *objectClass* pot ser de tres tipus:

1. *objectClass* estructural: s'utilitza per crear entrades (objectes de dades). Cada entrada ha de tenir una classe d'aquest tipus i prou.
2. *objectClass* auxiliar: es pot afegir a qualsevol entrada. Per exemple, *dcObject* (que inclou l'atribut *dc*, *domain component*, per fer referència a noms DNS).
3. *objectClass* abstracta: utilitzada per tractar entitats inexistents.

La classe d'objecte abstracta més comú és top, que constitueix el nivell més alt de les jerarquies d'objecte estructurals. Cal tenir en compte que una entrada només pot tenir una classe d'objecte abstracta.

Herència

Totes les classes d'objectes tenen un origen comú, la classe d'objecte top. També és possible que una classe d'objecte sigui part d'una jerarquia. En aquest cas hereta totes les característiques de les classes d'objecte pare (classes superiors o SUP). En particular cal fer servir tots els atributs obligatoris de les classes superiors, a més dels atributs addicionals necessaris que defineixen la nova classe.

Exemple d'herència a les classes d'objecte

La classe d'objecte inetOrgPerson deriva de la classe d'objecte organizationalPerson, que al seu torn deriva de person, que, a la fi, té el pare top. Per tant, inetOrgPerson inclourà els atributs de les classes d'objecte top, person, organizationalPerson i les que ella mateixa hagi definit.

Si una classe d'objecte és part d'una jerarquia ha de ser del mateix tipus (estructural o auxiliar) que l'*objectClass* superior. L'excepció a aquesta regla és si el superior és un tipus abstracte (per exemple, top).

Regles de coincidència (matching rules)

Les regles de coincidència defineixen els mètodes de comparació disponibles al servidor LDAP.

En general no cal definir-les de manera explícita, sinó que el servidor ja les inclou per defecte.

Les regles de coincidència es fan servir per a cada atribut mitjançant les propietats opcionals EQUALITY, SUBSTR i ORDERING, a les que s'assigna els seus noms auto-descriptius o el seu OID.

Consulteu el document "Sintaxi de l'esquema" de la secció "Annexos" del material web.

Esquemes (schemas)

Els esquemes són unitats d'embalatge: en general totes les classes d'objecte i atributs estan definits dins dels esquemes. S'ha de considerar que, inicialment, el servidor de directori no coneix cap esquema, de manera que cal carregar-hi els arxius d'esquema que contenen la descripció de les classes d'objecte que el directori ofereix.

Els esquemes següents són d'ús habitual:

1. *core.schema*: per a les classes i els atributs bàsics.
2. *cosine.schema*: per a algunes extensions útils com es defineix en l'RFC 1274, com ara la identificació d'usuari, el correu...
3. *inetorgperson.schema*: útil en alguns casos quan es necessiten més atributs, com s'especifica en l'RFC 2798.

Els esquemes són reutilitzables: la creació d'un *objectClass* dins d'un nou esquema pot fer servir atributs d'*objectClass* definits en altres esquemes.

Excepcionalment, hi ha algunes classes d'objecte i atributs definits com a operacionals, que estan implícits en el programari del servidor LDAP i no necessiten un arxiu d'esquema. Aquests afecten totes les entrades d'un servidor i es fan servir a l'entrada "subschema", localitzable pel valor de l'atribut *subschemaSubentry* de qualsevol entrada.

Consulteu el document "Sintaxi de l'esquema" de la secció "Annexos" del material web.

Exemple de creació d'una classe d'objecte dins d'un esquema

A continuació es mostra la definició de la classe d'objecte *person*. Aquesta classe d'objecte està definida en l'esquema *core.schema*; és, per tant, una classe d'objecte bàsica.

```
attributetype ( 2.5.4.41 NAME 'name'
EQUALITY caseIgnoreMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{32768} )

attributetype (2.5.4.3 NAME ( 'cn' 'commonName' ) SUP name )

attributetype (2.5.4.4 NAME ( 'sn' 'surname' ) SUP name )

objectclass ( 2.5.6.6 NAME 'person' SUP top STRUCTURAL
  MUST ( sn $ cn )
  MAY ( userPassword $ telephoneNumber $ seeAlso $ description )
      )


dn: uid=jperez,ou=people,dc=exemple,dc=edu
objectclass: top
objectclass: person
cn: José Pérez
sn: Pérez
```

En fer servir *objectclass: person* per declarar una entrada, automàticament es requereix que els atributs *sn* (cognom) i *cn* (nom comú) continguin un valor, ja que així està establert en la definició de la classe d'objecte. A més, permet l'ús dels atributs *userPassword*, *telephoneNumber*, *seeAlso* i *description*, però no obliga a assignar-hi un valor.

Informació no textual en directoris LDAP

Base64

Base64 és un sistema de numeració posicional que fa servir 64 com a base. És la major potència de dos que pot ser representada usant únicament els caràcters imprimibles de ASCII i això ha propiciat el seu ús per la codificació de diversos tipus d'informació.

Tradicionalment s'associen els directoris amb informació de tipus text. Cada implementació ofereix un ventall de possibles tipus de dades. En general n'hi ha de molts tipus diferents, incloses les dades binàries. Però s'han desenvolupat opcions en el cas que calgui incloure informació que no s'ajusti als tipus disponibles (per exemple, mitjançant la codificació Base64).

1.3.2 Model de nomenclatura

El model de nomenclatura defineix com s'organitza i es referencia la informació. Les entrades de directori es disposen en una estructura arborescent jeràrquica per tal de reproduir fronteres polítiques, geogràfiques, d'organització o altres criteris.

Com cal esperar, igual que els noms de domini (`www.google.com`) o la ruta completa d'un fitxer (`/etc/passwd`), el nom d'una entrada ha de ser únic en cada servidor LDAP.

A diferència d'altres models, en aquest tots els nodes poden tenir contingut.

1.3.3 Nom distintiu (DN)

La posició d'una entrada en la jerarquia del DIT ve determinada pel seu nom complet (DN). Cada component d'un DN es diu *nom distintiu relatiu* (RDN).

No hi ha normes especifiques sobre com construir aquest nom distintiu, l'única cosa important és que aquest nom ha de ser únic dins de l'espai de noms. És a dir, tots els fills d'una entrada han de tenir RDN diferents. Un RDN pot estar compost per un o més parells atribut/valor. És similar als noms de domini, però el separador és una coma (*dn: uid=mfreixas,ou=Personal,dc=empresa,dc=com*). També hi poden haver RDN multivalor (*dn: cn=Marc Freixas+uid=mfreixas,ou=Personal,dc=empresa,dc=com*)

Consulteu el document "Sintaxi del DN" de la secció "Annexos" del material web.

Però en el cas que la entrada serveixi per a la autenticació de l'usuari (operació *bind*), cal indicar el paràmetre "Bind DN" i opcionalment una contrasenya. Com que no es produeix cap operació de cerca, cal que el "Bind DN" coincideixi amb el DN de creació de l'entrada.

1.3.4 Sufix de directori (directory suffix)

Totes les entrades del directori tenen un node comú (arrel) que es diu *sufix del directori* o *base*.

L'LDAP permet construir el sufix del directori amb qualsevol d'aquests tres estils:

1. L'estil de noms X.500 (el nom de l'organització seguit del codi de país): `o=nom_empresa,c=es`.
2. L'estil de DNS. La mateixa entrada ara queda: `o=nom_empresa.es`.
3. L'estil de component de domini utilitza l'atribut *dc* per separar el nom de domini en components *dc*: `dc=nom_empresa,dc=es`.

1.3.5 Àlies (alias)

Hi ha situacions en què és útil estalviar-se duplicar la inserció d'una entrada real en el DIT. Es pot crear una entrada àlies, amb un comportament similar al dels enllaços simbòlics (accessos directes) en un sistema de fitxers.

Un àlies pot apuntar fins i tot a un servidor de directori. Com que un àlies erroni pot provocar un desastre, no totes les implementacions de directoris permeten l'ús d'àlies. Si la implementació no permet àlies, pot utilitzar referències.

El procediment de creació és el mateix que s'usa per afegir una nova entrada: amb

un DN i classe d'objecte àlies, i només hi ha un atribut necessari, *aliasedObject-Name*, que indica on és l'entrada real.

Exemple de creació d'un àlies

```
dn: cn=barcelona,dc=empresa,dc=com
objectClass: top
objectClass: alias
objectClass: extensibleObject
cn: barcelona
aliasedObjectName: cn=bcn,dc=empresa,dc=com
```

1.3.6 Referències (referrals)

Pot arribar un punt en què ja no és pràctic o eficient mantenir l'arbre del directori en un únic servidor. Per motius de rendiment pot interessar fer particions (*partitioning*), ja que tenir més servidors que ofereixen serveis de directori permet distribuir les peticions dels clients entre aquests servidors. O pot ser interessant per raons administratives, per permetre diferents polítiques per a diferents parts de l'arbre de directoris.

El procediment de creació és el mateix que s'usa per afegir una nova entrada: amb un DN i classe d'objecte referral, i només hi ha un atribut necessari, *ref*, que indica on resideix l'entrada real amb una URL LDAP.

Exemple on es referencia un subarbre del directori

Un servidor gestiona `dc=empresa,dc=com`, però un altre servidor, anomenat `srvB`, conté la informació de `dc=bcn,dc=empresa,dc=com`:

```
dn: DC=bcn,DC=exemple,DC=edu
objectClass: referral
objectClass: extensibleObject
dc: bcn
ref: [[ldap://srvB/DC=bcn,DC=exemple,DC=edu|ldap://srvB/dc=bcn,dc
    =exemple,dc=edu]]
```

1.3.7 Arrel DSE (rootDSE)

La informació operacional del servidor és en una entrada anomenada *arrel DSE*. DSE vol dir "entrada DSA específica". DSA vol dir "agent de servidor de directori" i conté atributs del servidor.

L'entrada "arrel DSE" té un DN de longitud zero i conté:

1. La versió LDAP compatible amb el servidor. Si el client no aconsegueix

recuperar aquesta informació, l'usuari pot concloure que és l'LDAP versió 2, que no és compatible amb aquesta funció.

2. Totes les classes d'objecte i tipus d'atributs reconeguts pel servidor.
3. Els sufixos emmagatzemats al servidor de directori.
4. Les operacions ampliades i els controls compatibles amb el servidor de directori.
5. Informació sobre els mecanismes de suport SASL.
6. Una llista de servidors LDAP per consultar la informació que el servidor original no pot proporcionar.

1.3.8 Model funcional

El model funcional defineix com es recupera i modifica la informació del directori, descrivint les operacions que es poden realitzar en l'accés, el manteniment i la gestió del directori.

Hi ha funcions que permeten buscar, comparar, afegir, modificar i esborrar entrades al directori. Totes són atòmiques, és a dir, l'operació s'executa en la seva totalitat o s'avorta si ocorre un error. Acompanyant al paquet d'instal·lació d'un servidor LDAP, s'acostuma a oferir un conjunt de programes per fer servir des de l'intèrpret de línia d'ordres que implementen l'accés a aquestes funcions.

A continuació descrivim les operacions agrupades per funcionalitat, sense aprofundir en la gestió d'errors.

Operacions d'autenticació i control

La primera operació és obrir la connexió amb el servidor (*bind*); després el client pot proporcionar al servidor les credencials de l'usuari (per exemple, *uid* i *userPassword*) per provar la seva identitat. Si el servidor accepta aquestes credencials, associa o "uneix" certs drets d'accés a les credencials d'usuari fins que es tanca la connexió (*unbind*) o s'envien noves credencials per obtenir drets d'accés diferents.

Connexió (bind)
Permet que el client s'autentiqui al servidor de directori.

Paràmetres:

1. *version*: la versió de l'LDAP que el client vol utilitzar.
2. *name*: nom de l'objecte de directori al qual el client desitja unir-se (un DN). En cas de referències o àlies no es resoldrà.

3. *authentication*: autenticació d'elecció, que té dos valors possibles:
   (a) *simple*: indica que el missatge viatja en text clar.
   (b) *sasl*: utilitza el mecanisme d'autenticació i seguretat de la capa SASL com es descriu en l'RFC 4752.

Desconnexió (unbind)
Allibera els recursos assignats al client, descarta la informació d'autenticació i tanca la connexió. No cal cap paràmetre ni es retorna cap valor.

Abandonar (abandon)
S'informa al servidor que aturi una operació prèviament sol·licitada. El servidor no envia cap resposta.

Paràmetre:

1. *operationID*: identificació de l'operació que s'abandona.

Operacions interrogatives

Les operacions interrogatives són les que s'utilitzen tant per buscar com per llegir les entrades. Són les següents:

1. Cercar (*search*)
2. Comparar (*compare*)

Cercar (search)
Sol·licita a un servidor que retorni el conjunt d'entrades coincidents amb un criteri de cerca complex.

Paràmetres:

1. *baseObject*: un DN base per iniciar la recerca.
2. *filter*: un filtre que defineix les condicions que ha de complir la cerca perquè coincideixi amb una entrada determinada.
3. *scope*: àmbit d'aplicació: en el nivell del DN (*baseObject*), en un nivell inferior (*singleLevel*) o en el conjunt del subarbre (*wholeSubtree*).
4. *Attributes*: quins atributs s'han de retornar. Hi ha dos casos especials:
   (a) llista buida (sense atributs) o *%x2A*, és a dir, un asterisc, *. Ambdós valors indiquen que es retornin tots els atributs de les entrades coincidents.
   (b) *%x31.2E.31*, és a dir, *1.1*: si és l'únic OID de la llista, no es retorna cap atribut, però sí els DN. Si no és l'únic s'ignora.

5. *derefAliases*: indica com s'ha de fer la resolució de les referències:
   (a) *neverDerefAliases*: no resol les referències dels àlies.
   (b) *derefInSearching*: resol la referència d'àlies en les entrades subordinades a l'objecte base en la cerca, però no per localitzar l'objecte base de la cerca.
   (c) *derefFindingBaseObj*: resol la referència d'àlies per localitzar l'objecte base de la cerca, però no en les entrades subordinades a l'objecte base en la cerca.
   (d) *derefAlways*: resol sempre les referències.
6. *sizelimit*: el nombre màxim d'entrades a retornar. "zero" significa que el client no imposa cap límit de mida. El servidor pot imposar un límit màxim.
7. *timeLimit*: nombre màxim de segons que una consulta pot trigar. "zero" significa que el client no imposa cap límit de temps. El servidor pot imposar un límit màxim.
8. *typesOnly*: un valor booleà. Establert a "true" indica que només es retornen els tipus d'atributs. En cas d'estar establert a "false", retorna els tipus d'atributs i els valors dels atributs.

Comparar (compare)
Permet a un client comparar una afirmació de valor amb els valors d'un atribut en particular en una entrada particular. El resultat és `compareTrue` en cas positiu, `compareFalse` en cas negatiu, o algun codi d'error, si no.

Paràmetres:

1. *entry*: el nom de l'entrada que s'ha de comparar (un DN). En cas de referències o àlies no es resol.
2. *ava*: una afirmació de valor d'un atribut que serà comparada.

Operacions d'actualització

Tot i no ser les més habituals sobre un directori, les operacions d'actualització tenen una gran importància a l'hora de mantenir la correcció de les dades.

Les operacions d'actualització són les següents:

1. Afegir (*add*)
2. Esborrar (*delete*)
3. Modificar (*modify*)
4. Modificar DN (*modify DN*)

Afegir (add)
Demana l'addició d'una entrada.

Paràmetres:

1. *entry*: nom complet de la nova entrada (un DN). En cas de referències o àlies no es resol.

2. *attributes*: una llista de parells nom-valor dels atributs continguts en l'entrada.

Esborrar (delete)
Demana l'eliminació d'una entrada.

Paràmetre:

1. *entry*: nom complet de la entrada a eliminar (un DN). En cas de referències o àlies no es resol.

Modificar (modify)
Modifica una entrada.

Paràmetres:

1. *object*: nom complet de la entrada a canviar (un DN). En cas de referències o àlies no es resol.

2. *changes*: una llista de modificacions a realitzar. Les opcions són:

3. *operation*: tipus d'operació que s'executa en aquesta entrada, amb tres valors possibles:

4. *add*: afegeix un nou atribut.

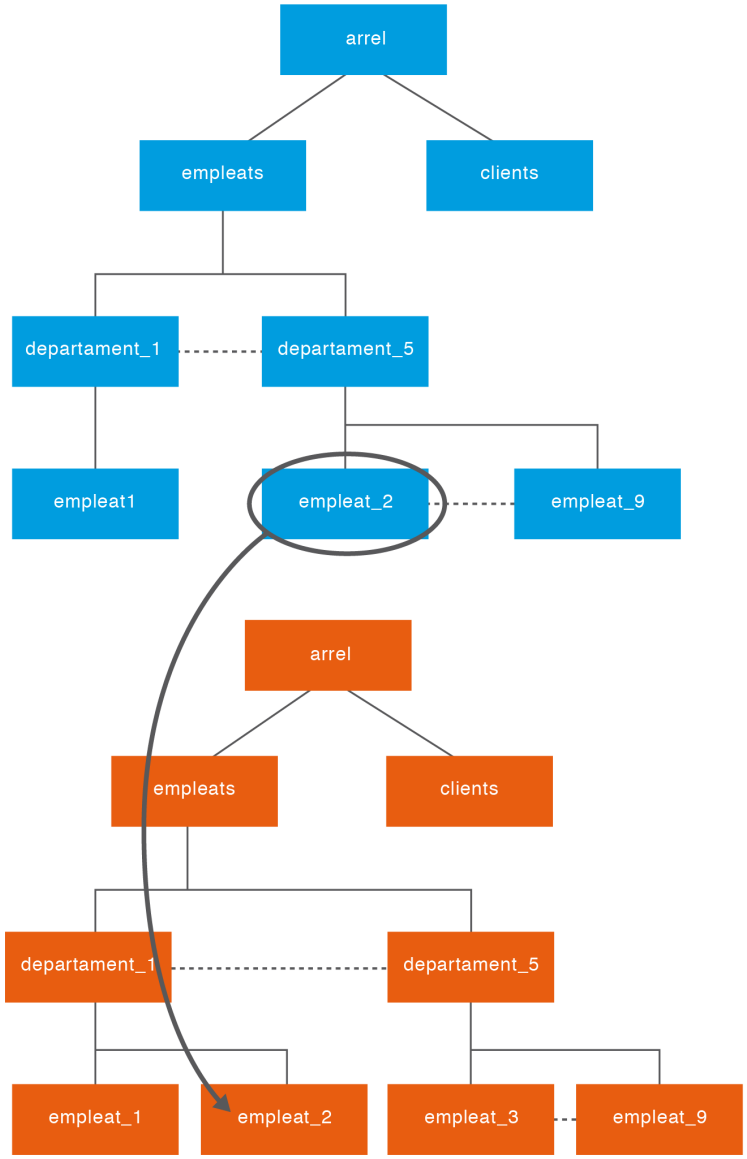
5. *delete*: elimina un atribut.

6. *replace*: modifica un atribut.

7. *modification*: una llista de parelles nom-valor que s'afegeix o modifica.

Modificar DN (modify DN)
Canvia l'RDN d'una entrada o passa un subarbre d'entrades a una nova ubicació al directori (vegeu la figura 1.7).

Figura 1.7. Modificació del DN



Paràmetres:

1. *entry*: nom complet de la entrada a canviar (un DN). En cas de referències o àlies no es resol. En l'exemple és `uid=empleat_2,ou=departament5,ou=empleats, dc=exemple,dc=edu`.

2. *newrdn*: nou nom complet relatiu (un RDN). En l'exemple és `uid=empleat_2`.

3. *deleteoldrdn*: valor booleà que indica si l'RDN vell s'ha de mantenir en el directori.

4. *newSuperior*: és un paràmetre opcional que indica quin serà el superior immediat d'*entry*. En cas de referències o àlies no es resol. En l'exemple és `ou=departament1,ou=empleats, dc=exemple,dc=edu`.

Operacions ampliades

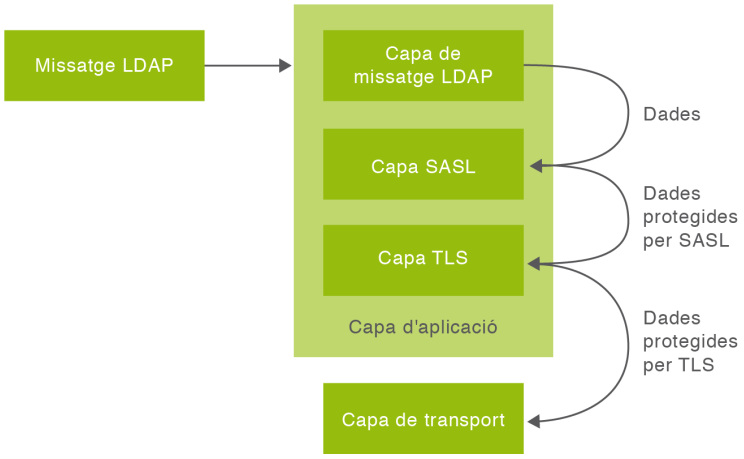
Les operacions ampliades es poden utilitzar per definir les noves operacions que no eren part de l'especificació del protocol original.

L'operació estesa més habitual és StartTLS.

StartTLS

L'operació StartTLS (definida a l'RFC 2830) permet a un client sol·licitar una connexió de tipus Transport Layer Security (el descendent d'SSL) mitjançant la transmissió d'una petició ampliada (ExtendedRequest, LDAPv3). Si el servidor és capaç i està en disposició de negociar TLS, el client ha de començar la negociació TLS (RFC 5246) o tallar la connexió. En la figura 1.8 es mostren les capes de seguretat en una sessió LDAP.

FIGURA 1.8. Capes de seguretat SASL i TLS en una sessió LDAP



1.3.9 Model de seguretat

El model de seguretat mostra com es controla l'accés a la informació continguda en el directori.

> Abans que un client pugui accedir a les dades d'un servidor LDAP, s'han de dur a terme dos processos: autenticació i autorització.

El model de seguretat (RFC 4513) descriu aquests processos juntament amb la integritat i confidencialitat en la transmissió de dades o la limitació en l'ús de recursos.

Autenticació

L'autenticació és el procés que assegura que les identitats dels usuaris i màquines (servidors o clients) estan correctament validades.

Normalment, la seva funció es redueix a verificar la identitat de l'usuari abans de permetre-li l'accés al sistema. Aquest control permet definir el nivell d'accés de cada usuari i objecte del directori.

Els serveis de directori s'utilitzen habitualment com a eines d'autenticació que emmagatzemen credencials de manera centralitzada, no només contrasenyes per autenticar els usuaris del mateix directori, sinó també mecanismes per verificar contrasenyes que autentiquin usuaris d'altres sistemes. Tanmateix, el protocol no imposa que les contrasenyes associades amb un usuari estiguin al servidor, sinó que permet que s'integrin amb serveis d'autenticació externs (per exemple, SASL).

Autenticació anònima
Tots els servidors LDAP han d'acceptar el tipus d'autenticació "sense autenticació en absolut", també anomenat *usuaris anònims*, ja que el servidor no sap qui està demanant una connexió. Aquest tipus d'autenticació té associada una autorització de tipus anònima.

S'aconsegueix escollint la opció d'autenticació simple a *bind*, però sense proporcionar credencials d'usuari al servidor.

Connexió no autenticada
Una connexió no autenticada dóna lloc a una autorització anònima, però la identificació facilita la traçabilitat de la connexió.

S'aconsegueix escollint la opció d'autenticació simple a *bind*, però sense proporcionar cap contrasenya al servidor.

Autenticació bàsica
L'autenticació bàsica també s'utilitza en altres protocols com HTTP. El client simplement envia les credencials d'usuari per la xarxa en format clar.

S'aconsegueix escollint l'opció d'autenticació simple a *bind*, proporcionant nom distintiu i contrasenya al servidor. El servidor busca un atribut anomenat *userPassword* en l'entrada corresponent al nom distintiu i el compara amb l'entrada enviada pel client. Si la contrasenya coincideix, s'estableix la connexió amb el servidor LDAP. Si no, s'envia un missatge d'error i es tanca la connexió.

SASL
És el mètode recomanat pel protocol LDAP. SASL Separa els mecanismes d'autenticació dels protocols de l'aplicació, permetent que qualsevol protocol d'aplicació que faci servir SASL utilitzi qualsevol mecanisme d'autenticació suportat per

SASL (RFC 4422 i 4752). La capa d'autenticació i seguretat (SASL) proporciona serveis d'autenticació a un protocol orientat a la connexió, com el LADP. Un cop el servidor i el client estan connectats, estableixen un acord sobre el mecanisme de seguretat.

Els clients poden determinar els mecanismes SASL que un servidor admet llegint l'atribut *supportedSASLMechanisms* des de l'arrel DSE.

A més de l'autenticació, ofereix seguretat de dades (integritat i confidencialitat).

Alguns mecanismes d'autenticació definits per SASL són:

- EXTERNAL (per exemple, IPSec o TLS)
- ANONYMOUS
- PLAIN (text clar)
- OTP (One Time Password)
- SKEY
- CRAM-MD5
- DIGEST-MD5
- NTLM
- GSSAPI
- GATEKEEPER

Contrasenyes
El protocol LDAP (RFC 4519) ofereix *userPassword*, un tipus d'atribut multivalor definit per donar suport a l'operació *bind* amb autenticació simple i accessible únicament pel propi usuari. En ser multivalor, per validar l'usuari es prova la contrasenya candidata amb cada valor (útil en períodes de transició). Aquest atribut apareix en classes d'objecte com organizationalUnit o person i a la seva definició es diu que s'emmagatzema com a text clar.

Algunes implementacions aprofiten el text clar per fer ús de més tipus de mecanismes d'autenticació, fent servir una funció resum (*one-way has*h) i guardant el resultat codificat en Base64.

L'RFC 3112 oficialitza aquesta pràctica i defineix l'*authentication password schema* per guardar valors derivats de les contrasenyes de l'usuari amb un nou tipus d'atribut, *authPassword*, que permet esquemes d'emmagatzemament múltiple i regles de coincidència per validar que una contrasenya en text clar coincideix amb un dels valors de l'atribut. Aquest atribut apareix a classes d'objecte com posixAccount, posixGroup o shadowAccount. Els clients poden determinar els esquemes que un servidor suporta llegint l'atribut *supportedAuthPasswordSchemes* des de l'arrel DSE (per exemple, BASE64, CLEAR, CRYPT, MD5, SHA, SMD5, SSHA, SSHA256, SSHA384, SSHA512 o SASL).

Per exemple, si la contrasenya és `abcd123`:

```
abcd123 xifrat amb SHA = fDYHuOYbzxlE6ehQOmYPIfS28/E=
userPassword: {SHA}fDYHuOYbzxlE6ehQOmYPIfS28/E=

{SHA}fDYHuOYbzxlE6ehQOmYPIfS28/E= codificat en Base64 =
    e1NIQX1mRFlIdU9ZYnp4bEU2ZWhRT21ZUElmUzI4L0U9

authPassword:: e1NIQX1mRFlIdU9ZYnp4bEU2ZWhRT21ZUElmUzI4L0U9
```

La creació d'atributs nous en esquemes personalitzats per guardar contrasenyes no es considera una bona pràctica.

En cap cas no és convenient fer la transferència de la contrasenya ni del *hash*. Per això és important saber que si no s'utilitza SSL/TLS o SASL amb algorisme de negociació segur el client LDAP enviarà les credencials amb text clar i serà el servidor qui calcularà el valor de *hash* i el comparará amb el valor emmagatzemat.

D'altra banda, és important crear una política de contrasenyes, encara que la seva implementació i compliment no sigui fàcil (implica que els usuaris canviïn les seves contrasenyes periòdicament, requisits de construcció de les contrasenyes, restringir la reutilització d'una contrasenya vella, impedir els atacs d'endevinació de contrasenyes...).

Consulteu l'adreça web "Model de polítiques de contrasenyes" a la secció "Adreces d'interès" del material web per accedir a la desena versió d'un *Internet draft* que recull un conjunt de regles que controla com s'utilitzen i administren les contrasenyes en un servei de directori basat en l'LDAP.

Internet draft

Internet draft són esborranys vàlids per a un màxim de sis mesos que poden ser actualitzats o reemplaçats per altres documents o esdevenir obsolets en qualsevol moment. No és apropiat l'ús d'*Internet drafts* com a material de referència ni citar-los d'altra manera que com a "treballs en curs".

La classe d'objecte pwdPolicy es defineix amb atributs que mantenen la informació de l'estat de la directiva de contrasenyes general per a cada usuari. La seva declaració és:

```
( 1.3.6.1.4.1.42.2.27.8.2.1
NAME 'pwdPolicy'
SUP top
AUXILIARY
MUST ( pwdAttribute )
MAY ( pwdMinAge $ pwdMaxAge $ pwdInHistory $ pwdCheckQuality $
pwdMinLength $ pwdMaxLength $ pwdExpireWarning $
pwdGraceAuthNLimit $ pwdGraceExpiry $ pwdLockout $
pwdLockoutDuration $ pwdMaxFailure $ pwdFailureCountInterval $
pwdMustChange $ pwdAllowUserChange $ pwdSafeModify $
pwdMinDelay $ pwdMaxDelay $ pwdMaxIdle ) )
```

pwdAttribute és el nom de l'atribut al qual s'aplica la directiva (per exemple, *userPassword*). Es recomana exigir un nivell de qualitat de les contrasenyes, canvis periòdics i bloqueig (temporal o no) de comptes en superar el límit d'intents.

Autorització i control d'accés

Una política de control d'accés és un conjunt de regles que defineixen la protecció dels recursos en termes de les capacitats de les persones o entitats que accedeixen a aquests recursos.

La sol·licitud pot estar associada a una àmplia varietat de factors de control d'accés (ACFs), per exemple, l'adreça IP d'origen, el nivell de xifrat, el tipus d'operació que se sol·licita, l'hora del dia...

Cada sessió LDAP té associat un estat d'autenticació vinculat a una identitat d'autorització.

L'autenticació anònima i la connexió no autenticada comporten un estat d'autorització anònima, mentre que una autenticació bàsica comporta un estat d'autorització autenticada. SASL permet, opcionalment, que el client comuniqui la identitat d'autorització desitjada.

Cada implementació estableix els seus propis mecanismes d'administració i emmagatzemament. L'RFC 2820 descriu els requeriments de control d'accés, però no en facilita la implementació, cosa que dificulta la interoperabilitat en aquest aspecte.

El control d'accés es gestiona normalment en forma de llistes de control d'accés (ACL). Així, les polítiques de control d'accés sovint s'expressen en termes d'identitats d'autorització: "L'entitat X pot realitzar l'operació Y del recurs Z".

Consulteu el document "Sintaxi de configuració dinàmica del control d'accés a l'OpenLDAP" de la secció "Annexos" del material web.

Per exemple:

```
access to * by * read

access to *
    by self write
    by anonymous auth
    by * read

olcAccess: to *
    by users read
```

Privacitat i integritat de les dades

És important assegurar que les dades no es modifiquen ni es donen a conèixer durant la seva transmissió.

L'LDAP amb SSL/TLS

El protocol SSL (Secure Sockets Layer) implementa mecanismes de seguretat basats en criptografia de clau pública a la pila de protocols TCP/IP entre la capa de transport i la capa d'aplicació, és a dir, la capa LDAP.

TLS proporciona un mecanisme de seguretat que permet l'autenticació mútua del servidor i el client, la negociació segura i fiable del protocol d'encriptació i les claus criptogràfiques, tot això abans que el protocol d'aplicació (LDAP) envii o rebi el seu primer *byte* de dades.

L'LDAP li dóna suport mitjançant l'operació ampliada StartTLS i, opcionalment, ofereix autenticació quan es combina amb SASL EXTERNAL.

El protocol TLS 1

Transport Layer Security va néixer a partir de la versió 3 de l'SSL per oferir confidencialitat i integritat a la comunicació entre dues aplicacions. Es descriu als RFC 5246, 5746, 5878 i 6176. Les característiques del TLS, ordenades per prioritats del disseny, són: seguretat criptogràfica, interoperabilitat, extensibilitat i eficiència.

SASL

Aquest tipus de connexió ja s'ha presentat com a opció d'autenticació.

El client sol·licita una connexió segura mitjançant l'operació ampliada *StartTLS*. Si el servidor respon que està llest per negociar amb el protocol *TLS Handshake* de TLS, primer es posen d'acord en la versió del TLS i després en el mecanisme de seguretat que s'utilitzarà per a la comunicació. Ambdues parts decideixen si accepten o no el nivell de seguretat assolit. Si el servidor o el client decideixen que el nivell no és suficient, es tanca la connexió (TLS).

Protecció contra la monopolització

Per tal de garantir la disponibilitat del servei, cal limitar l'ús de recursos per evitar que un únic usuari monopolitzi el sistema.

L'LDAP ofereix límits d'autocontrol en la mida del resultat o temps d'algunes operacions. D'altra banda, l'estàndard recomana definir límits administratius mitjançant els controls de servei per tal de protegir el servidor, però no ho concreta, de manera que cada implementació ha desenvolupat la seva pròpia solució.

1.4 El format d'intercanvi de dades LDIF

L'LDIF (LDAP Data Interchange Format o format d'intercanvi de dades LDAP) és un estàndard de format de text pla utilitzat per representar el contingut d'un directori LDAP i les sol·licituds d'actualització del directori (afegir una entrada, modificar-la, eliminar-la...).

Podem crear fitxers de text pla amb format LDIF per representar la informació que conté un directori i també es poden fer servir per modificar el contingut del directori.

LDIF representa el contingut d'un directori amb una sèrie de registres, cadascun dels quals representa una entrada del directori. De la mateixa manera, una sol·licitud d'actualització del directori (adició, modificació, eliminació) es representa també com un registre (de fet, hi haurà un registre per a cada sol·licitud).

L'ús d'un fitxer de text pla és molt útil. En primer lloc és autoexplicatiu, fàcil de llegir i d'entendre. A més, és molt fàcil de produir o processar utilitzant scripts o algun llenguatge de programació. D'altra banda, com que la compatibilitat entre

les diferents implementacions de servei de directori no està garantida, sempre es poden fer servir els fitxers en aquest format per transportar informació d'un servei de directori a un altre, és a dir, per importar i exportar informació entre servidors LDAP.

1.4.1 Format d'un fitxer LDIF

El format de fitxer LDIF està definit a l'RFC 2849.

Els fitxers LDIF estan formats per un conjunt de línies de diversos tipus. Els tipus de línies que es distingeixen són els següents:

- **Línia de directiva:** una línia que comença amb qualsevol caràcter, excepte espai o coixinet. Aquestes són les línies realment importants, ja que contenen la informació de les entrades i les operacions a realitzar.
- **Línia en blanc:** les línies en blanc s'utilitzen normalment per separar les diferents seqüències d'entrada.
- **Línia de comentari:**una línia que comença amb un coixinet.
- **Línia de continuació:** una línia que segueix una línia de directiva i comença amb un espai. Se suposa que els caràcters següents són part de la línia anterior.
- **Línia de separació:** una línia que comença amb un guió (–). Les línies de separació són utilitzades típicament per acabar les seqüències d'operador.

Els fitxers LDIF són molt sensibles als espais i les línies en blanc. S'ha de ser molt respectuós amb la sintaxi del format.

S'anomena **seqüència** a l'**agrupació de directives**separades per una o més línies en blanc. Existeixen dos tipus de seqüències: d'entrada i d'operador.

Seqüència d'entrada

Les seqüències d'entrada defineixen una entrada del directori. Són un grup de directives que comencen amb una directiva **DN (nom distintiu)**, seguida d'altres directives que descriuen els diferents atributs d'una entrada al DIT. Les seqüències d'entrada sempre comencen amb la directiva DN i acaben amb una línia en blanc.

Exemple de seqüència d'entrada

```
1 dn: ou=Persones, dc=empresa,dc=com
2 ou: Persones
3 objectClass: organizationalUnit
```

L'exemple mostra una entrada del directori, concretament una unitat organitzativa. Inclou **tresdirectives:**la primera defineix el DN i les altres dues descriuen diferents atributs de l'entrada o objecte. Especifiquen que l'objecte és una unitat organitzativa i donen valor a l'atribut *ou*.

Seqüència d'operador

Una seqüència d'operador és un grup de directives que inclou la directiva***changetype***. Una seqüència d'operador acaba amb una línia de separació o una línia en blanc.

La directiva *changetype* admet com a valors:

- ***add***: afegeix entrades.
- ***delete***: elimina entrades.
- ***modify***: modifica entrades, però cal especificar quin atribut volem modificar i quina és la modificació que cal realitzar. Aquestes modificacions poden ser *add*, *delete* o *replace*. Per fer diverses modificacions en una entrada es fa servir el separador -.

Exemple de seqüència d'operador changetype: modify

```
dn: cn=Marc Freixas,ou=Persones,dc=empresa,dc=com
changetype: modify
add: work-phone
work-phone: 931234561
work-phone: 931234562
-
delete: home-fax
-
replace: home-phone
home-phone: 415/697-8899
```

En l'exemple es modifica l'entrada "Marc Freixas". S'hi han afegit dos telèfons de la feina, s'ha esborrat el fax del domicili i s'ha modificat el telèfon particular. Es pot observar com després del DN apareix la directiva changetype. Com que hi ha més d'una modificació, cada modificació se separa de la següent mitjançant un guionet (-).

- ***modrdn***: modifica el DN relatiu, és a dir, sense canviar la base. Cal fer servir la directiva ***newrdn***per definir el nou DN. Es pot decidir si es vol mantenir o eliminar l'antic DN amb la directiva ***deleteoldrdn***.

Exemple de directiva modrdn

```
dn: cn=Marc Freixas,ou=people,dc=example,dc=com
changetype: modrdn
newrdn: Marc Freixas Vila
# deletes old RDN entry
deleteoldrdn: 1
```

En l'exemple s'ha canviat el nom de la persona de "Marc Freixas" a "Marc Freixas Vila" i s'ha esborrat l'anterior DN.

- ***moddn***: modifica el DN complet. Això implica moure l'objecte a un altre node del DIT. S'ha d'especificar el nou pare de l'objecte amb la directiva ***newsuperior***. Opcionalment, es pot mantenir o esborrar l'entrada anterior amb la directiva ***deleteoldrdn***.

Exemple de directiva moddn

```
dn: cn=Marc Freixas,ou=Persones,dc=empresa,dc=com
changetype: moddn
newsuperior: ou=exempleats,dc=empresa,dc=com
# Mantenim l'anterior entrada RDN
deleteoldrdn: 0
```

En l'exemple, Marc Freixas ha passat a tenir com a nou DN "cn=Marc Freixas,ou=exempleats,dc=empresa,dc=com".

2. Instal·lació, configuració i manteniment del servei de directori

El programari OpenLDAP és una implementació del protocol LDAP que ens permet instal·lar, configurar i mantenir el servei de directori d'una organització. Aquest programari és compatible amb la majoria de distribucions de Linux i amb altres sistemes operatius, com Android, Mac OS X, Solaris, Microsoft Windows (NT i derivats, és a dir, 2000, XP, Vista, Windows 7...) i z/OS.

2.1 Introducció a l'OpenLDAP

L'OpenLDAP és una implementació lliure, de codi obert, del Lightweight Directory Access Protocol (LDAP) desenvolupat pel projecte OpenLDAP. Disposa d'una llicència anomenada OpenLDAP Public License. El projecte el va iniciar Kurt Zeilenga clonant el codi font de referència de la Universitat de Michigan, que és on es va desenvolupar originalment el protocol LDAP.

Els components bàsics de la implementació OpenLDAP són els següents:

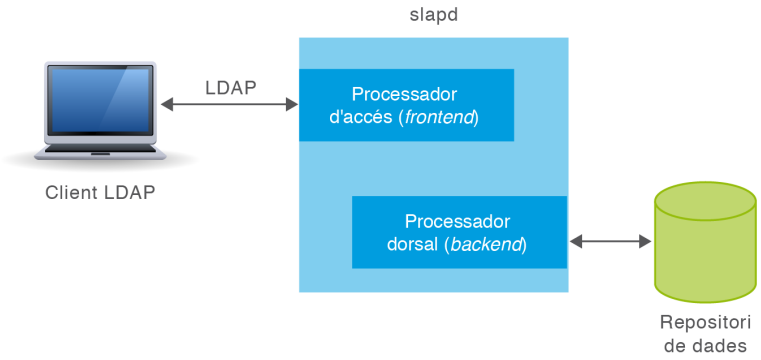
1. El **servidor slapd** (*stand-alone LDAP daemon*): un dimoni que escolta per diferents ports (per defecte pel port 389) peticions de connexió LDAP.

2. Les llibreries de client LDAP: llibreries que implementen el protocol LDAP i que poden ser utilitzades per desenvolupar programari client amb accés al protocol LDAP.

3. Utilitats, eines i diversos clients de mostra.

L'arquitectura del servidor slapd està dividida en una secció de processament frontal (*frontend*), que controla les connexions de xarxa i el processament del protocol, i una secció de processament dorsal o de segon pla (*backend*), que s'encarrega de l'emmagatzematge i la recuperació de les dades en resposta a les peticions que es reben. Aquesta arquitectura es mostra en la figura 2.1.

FIGURA 2.1. Arquitectura OpenLDAP



Vegeu l'apartat "Configuració dinàmica del servei" d'aquesta unitat per a més informació sobre la configuració cn=config.

Com que l'arquitectura és modular, hi ha una gran varietat de *backends* disponibles per interactuar amb diferents tecnologies, no només amb bases de dades tradicionals. Entre d'altres, podem emmagatzemar el directori (o també generar de forma automàtica la informació del directori) mitjançant:

1. **back-bdb**: el primer *backend* transaccional per a l'OpenLDAP, construït a partir d'Oracle BerkeleyDB. Es tracta d'una base de dades molt utilitzada a la indústria.
2. **back-hdb**: una variant de back-bdb que és totalment jeràrquica i permet reanomenar subarbres.
3. **back-ldif**: fa servir fitxers LDIF de text. És el *backend* per defecte per a la configuració cn=config.
4. **back-ndb**: un *backend* transaccional construït a partir del motor d'emmagatzematge de dades NDB de MySQL. NDB (Network DataBase) permet l'emmagatzematge de dades de forma distribuïda en una xarxa.
5. **back-ldap**: servidor intermediari simple a altres servidors LDAP. És a dir, utilitza altres servidors LDAP per emmagatzemar i recuperar la informació.
6. **back-sql**: estableix connexions a bases de dades SQL fent que puguin ser utilitzades com a magatzem de dades.

A més de la secció de processament frontal i les de processament dorsal o de segon pla, hi poden haver *overlays*. Els *overlays* són components de programari situats en el processador dorsal que n'augmenten les funcionalitats. Així, es pot modificar el comportament del processadorsense haver-ne de reescriure el codi o modificar-ne la resposta abans de tornar-la al processador frontal.

Existeix una gran quantitat d'*overlays*. Entre d'altres:

1. **Password Policy**: defineix polítiques de contrasenyes (longitud mínima, caducitat...).
2. **AccessLogging**: manté un registre dels accessos a un *backend* determinat en una base de dades diferent.

3. **Audit Logging**: manté un registre de les modificacions efectuades sobre un *backend* determinat.

4. **Value Sorting**: permet ordenar els valors d'un atribut multivalor.

2.2 Planificació del servei de directori

Abans d'implementar un directori cal fer una planificació anticipada d'alguns aspectes:

1. Definir per a què s'utilitzarà, és a dir, definir els continguts del directori.

2. Decidir quina serà l'organització de les dades. Amb l'LDAP la informació es representa en forma d'arbre, per tant, abans de començar hem de pensar quina forma tindrà aquest arbre d'informació del directori o DIT (Directory Information Tree).

3. Definir aspectes de seguretat de les dades.

4. Definir aspectes de rendiment.

La definició d'aspectes de seguretat i rendiment queden fora de l'abast d'aquesta unitat i només tractarem els dos primers punts.

2.2.1 Escenari d'exemple

L'RFC 2307bis fa una sèrie de recomanacions de cara a fer servir un servidor LDAP per realitzar una acreditació centralitzada d'usuaris.

Vegeu a la secció d'adreces d'interès del web del mòdul l'enllaç al text complet de l'RFC 2307bis.

Un dels usos més habituals dels directoris és permetre l'autenticació centralitzada dels usuaris d'una xarxa. Veurem un exemple de com fer servir el servidor OpenLDAP per realitzar l'acreditació centralitzada dels usuaris d'una organització.

Suposem que l'organització té les característiques següents:

1. El nom de l'empresa és *Empresa*.

2. L'empresa té comprat un domini Internet anomenat *empresa.com*.

3. L'empresa vol mantenir al directori les dades dels seus empleats, però també les dels clients i proveïdors.

4. L'empresa té dos departaments: administració i departament comercial.

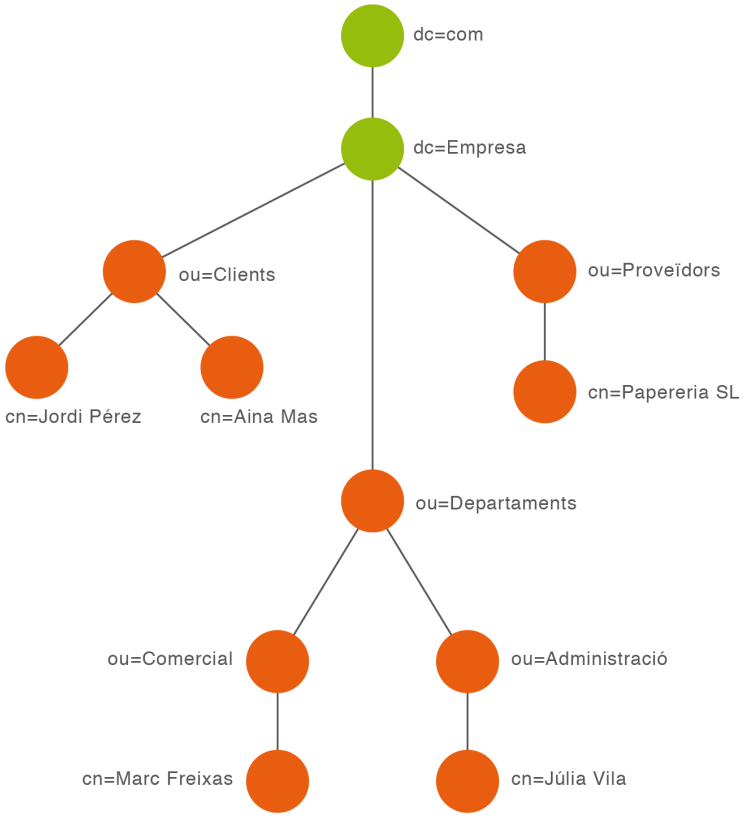
Partint d'aquest supòsit es poden fer diversos dissenys per organitzar les dades del directori.

El primer que cal és establir l'arrel. La manera més habitual de fer-ho és utilitzant el nom del domini per definir la base del DIT. En aquest cas l'arrel

serà “dc=empresa,dc=com”. Una vegada definida l’arrel, cal estructurar la resta d’elements. Una classe d’objecte que permet organitzar les entrades de l’arbre per al nostre cas és la classe organizationalUnit (unitat organitzativa).

En la figura 2.2 es pot observar una proposta d’organització del directori. Aquesta figura està simplificada, ja que el DIT pot contenir molta més informació i cada entrada també té molts més atributs.

FIGURA 2.2. DIT d’una organització

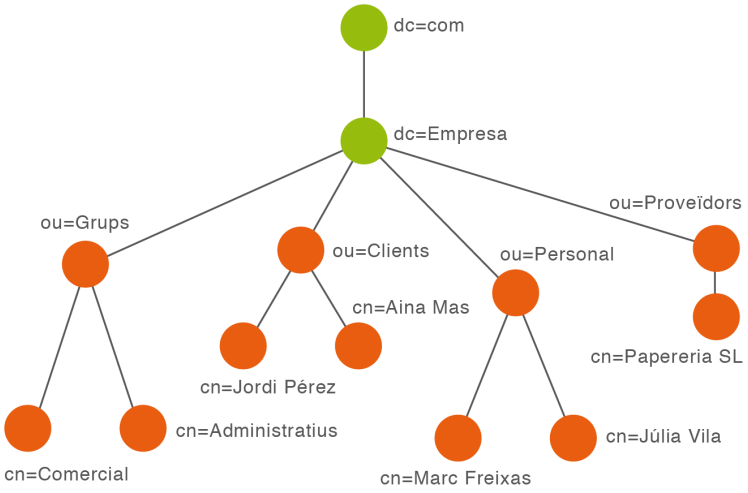


L’organització de la informació de la figura 2.2 té l’inconvenient que no segueix les recomanacions de l’RFC 2307bis. Bàsicament s’ha de tenir en compte que les organitzacions són canviants al llarg del temps, els departaments poden canviar de nom, els empleats poden canviar de departament... Per aquesta raó es proposa crear un DIT on hi hagi una unitat organitzativa per englobar els departaments (usualment l’anomenem *Grups*) i una sola unitat organitzativa que contingui tots els usuaris de l’empresa (nosaltres l’hem anomenat *Personal* però sovint la trobarem amb el nom de *People*). Aquesta estructura es mostra en la figura 2.3. A banda de Grups i Personal, crearem una unitat organitzativa addicional per a cada entitat externa a la nostra empresa. En aquest cas en creem una per als clients i una altra per als proveïdors.

L’estructura de la figura 2.3 és molt més adequada per construir un DIT per mantenir la informació dels usuaris i els departaments de l’organització, i també la podrem fer servir per configurar el sistema per fer l’acreditació centralitzada dels usuaris a partir del servei de directori. Per tant, aquesta estructura és la que pren-

drem com a model per implementar el directori amb el programari OpenLDAP i la que prendrem com a referència per entendre i seguir les explicacions d'aquesta unitat.

Figura 2.3. DIT d'una organització amb les recomanacions de l'RFC 2307bis



2.3 Instal·lació de l'OpenLDAP

La versió actual de l'OpenLDAP és la 2.4, que compleix l'LDAPv3. En el nostre cas en considerarem la instal·lació per al sistema operatiu Debian (6.0.3-Squeeze).

El programari es pot descarregar des del web del projecte OpenLDAP, però aquest només l'ofereix en forma de codi font. Podem optar per compilar el programari o per usar alguna compilació realitzada per tercers. En el cas de Debian GNU/Linux, el programari és compatible i està disponible als repositoris de la versió estable de Debian, així que se'n simplifica la instal·lació i configuració inicial.

2.3.1 Requeriments previs

Primerament cal disposar d'accés amb drets d'administrador (*root*) a un equip amb sistema operatiu Debian, preferiblement amb connexió a Internet i que no tingui instal·lat prèviament OpenLDAP.

Abans de fer servir apt-get o aptitude, ens hem d'assegurar que el sistema operatiu té el nom de la màquina i l'FQDN correctament configurats. Per comprovar-ho farem servir l'ordre *hostname*. Amb aquesta ordre coneixerem tant el nom de la màquina com l'FQDN. Instal·lem una màquina anomenada **servidor** al domini **empresa.com**. Podem comprovar que és correcte amb les ordres següents:

```
1 root@servidor:~# hostname
2 servidor
```

```
root@servidor:~# hostname -f
servidor.empresa.com
```

FQDN

FQDN (Fully Qualified Domain Name) és un nom de màquina que inclou a més el nom del domini associat a l'equip, i especifica la situació exacta de la màquina en la jerarquia DNS. Per exemple, si l'ordinador s'anomena *servidor* i el nom de domini és *domini.com*, l'FQDN serà *servidor.domini.com*.

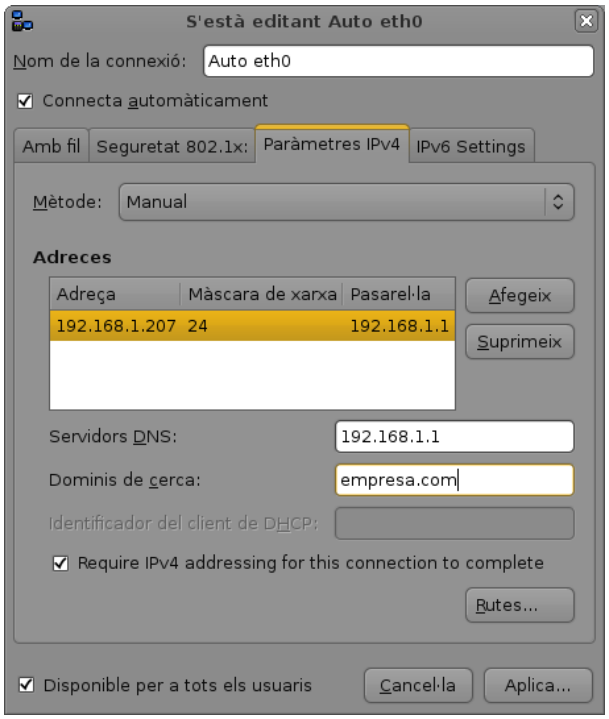
És absolutament necessari tenir correctament configurat el nom de la màquina i del domini abans de fer la instal·lació del servidor OpenLDAP. Per fer-ho a Debian Squeeze podem fer servir els fitxers /etc/hostname, on únicament hi ha el nom de la màquina, i /etc/hosts, que té un contingut similar a aquest:

```
127.0.0.1localhost
127.0.1.1servidor.empresa.com servidor

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

L'altra cosa totalment necessària és configurar correctament la xarxa. No es pot tenir un servidor OpenLDAP amb una adreça IP dinàmica, ja que els clients han de conèixer aquesta adreça, ja sigui per la IP directament o per la consulta a un servidor DNS. Per fer aquesta configuració tenim dues opcions.

Si tenim l'entorn gràfic instal·lat i fem servir NetworkManager, l'haurem de fer servir per configurar la xarxa com mostra la figura 2.4. En aquest exemple estem suposant que el servidor té l'adreça IP fixa 192.168.1.207 i màscara per defecte.

FIGURA 2.4. Configuració TCP en mode gràfic



Si, per contra, l'entorn gràfic no està instal·lat, no disposarem del NetworkManager ni de la seva miniaplicació (*applet*) per a Gnome. Aleshores farem servir el fitxer `/etc/network/interfaces`:

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
allow-hotplug eth0
iface eth0 inet static
    address 192.168.1.207
    netmask 255.255.255.0
    gateway 192.168.1.1
```

2.3.2 Instal·lació del programari

Als repositoris de Debian es pot trobar el servidor OpenLDAP disponible per ser instal·lat mitjançant apt-get o aptitude. Com a superusuari executem l'ordre següent:

Abans de fer qualsevol instal·lació és convenient actualitzar la informació dels paquets dels repositoris mitjançant *apt-get update* o *aptitude update*.

```
root@servidor:~# aptitude install slapd
...
```

La configuració per defecte es fa mitjançant `debconf`. L'únic que el sistema ens preguntarà és quina contrasenya volem definir per a l'administrador del servidor OpenLDAP.

debconf

debconf (Debian Package Configuration System) és un programa que s'utilitza per fer tasques de configuració del sistema. Està desenvolupat principalment per a la distribució de Linux Debian, i està integrat amb el sistema de gestió de paquets dpkg de Debian. Quan els paquets estan essent instal·lats, debconf pregunta a l'usuari qüestions que determinen el contingut de tot el sistema de fitxers de configuració associats al paquet. Després de la instal·lació del paquet, és possible tornar enrere i canviar la configuració d'un paquet utilitzant dpkg-reconfigure o un altre programa com Synaptic.

`debconf` tria per defecte les millors opcions per fer la configuració de l'slapd. Aquesta configuració està determinada principalment per l'FQDN del sistema; és per això que cal realitzar correctament la configuració del nom del sistema.

Si tot ha funcionat correctament, les darreres línies de la instal·lació ens comunicaran que tot ha anat bé i que s'ha iniciat el servidor `slapd`.

```
...
S'està configurant slapd (2.4.23-7.2)...
  Creating new user openldap... done.
  Creating initial configuration... done.
  Creating LDAP directory... done.
Starting OpenLDAP: slapd.
root@servidor:~#
```

Aquestes línies informen de les modificacions realitzades al sistema: s'ha creat un usuari, anomenat *openldap*, que és l'usuari amb el queal s'executarà el servei `slapd`, i s'ha creat la configuració inicial del servei i el directori LDAP.

Vegeu l'apartat "Verificació de la instal·lació" d'aquesta unitat per aprendre com fer una verificació de la configuració inicial.

Podem saber quins executables s'han instal·lat amb el paquet slapd mitjançant l'ordre següent:

```
root@servidor:~# dpkg -L slapd | grep bin
/usr/share/man/man5/slapo-pbind.5.gz
/usr/sbin
/usr/sbin/slappasswd
/usr/sbin/slapindex
/usr/sbin/slapd
/usr/sbin/slapauth
/usr/sbin/slapcat
/usr/sbin/slapacl
/usr/sbin/slapadd
/usr/sbin/slapdn
/usr/sbin/slaptest
```

Les llibreries estan situades al directori `/usr/lib/`.

Vegeu l'apartat "Utilitats de línia d'ordres" d'aquesta unitat per conèixer la utilitat dels executables instal·lats amb el paquet slapd i el paquet `ldap-utils`.

Abans de prosseguir la configuració, és convenient instal·lar el paquet `ldap-utils`, un conjunt d'eines que podem utilitzar per operar com a clients de l'OpenLDAP i poder accedir al directori, afegir dades, modificar-les... Executem l'ordre següent:

```
root@servidor:~# aptitude install ldap-utils
```

Podem conèixer els executables instal·lats amb el paquet ldap-utils mitjançant l'ordre següent:

```
root@servidor:~# dpkg -L ldap-utils | grep bin
/usr/bin
/usr/bin/ldapdelete
/usr/bin/ldappasswd
/usr/bin/ldapexop
/usr/bin/ldapcompare
/usr/bin/ldapmodify
/usr/bin/ldapsearch
/usr/bin/ldapwhoami
/usr/bin/ldapmodrdn
/usr/bin/ldapurl
/usr/bin/ldapadd
```

2.3.3 Verificació de la instal·lació

Podem comprovar que el servei `slapd` s'està executant amb l'ordre següent:

```
root@servidor:~# ps -ef|grep slapd
```

O bé, com és habitual amb tots els serveis, mitjançant l'ordre:

```
root@servidor:~# service slapd status
```

També podem veure el port que fa servir (normalment és el 389):

```
root@servidor:~# netstat -tunlp | grep slapd
```

A més de comprovar que el dimoni slapd s'està executant i està connectat al port adequat, també hem de comprovar que el servei funciona correctament. Per fer aquesta comprovació es pot usar l'ordre *ldapsearch* (del paquet `ldap-utils`) amb els paràmetres següents:

```
# ldapsearch -x -LLL -H ldap:/// -b dc=empresa,dc=com dn
```

Vegeu l'apartat "Utilitats de línia d'ordres" d'aquesta unitat per a una explicació més completa de la sintaxi de l'ordre *ldapsearch*.

L'ordre anterior ens dóna un resultat com aquest:

```
dn: dc=empresa,dc=com

dn: cn=admin,dc=empresa,dc=com
```

Aquesta sortida mostra que la instal·lació del paquet slapd ha creat per defecte una configuració bàsica del directori amb un DIT basat en l'FQDN del sistema (en aquest cas l'arrel és "dc=empresa,dc=com") i que ha creat un usuari administrador per al directori (tal com mostra l'entrada "cn=admin,dc=empresa,dc=com") que ens permetrà realitzar operacions de modificació en el directori.

Per defecte, el dimoni slapd també desa tota la informació relacionada amb la seva configuració (opcions globals de configuració, definicions d'esquemes, definicions de *backends* i bases de dades...) en objectes del directori a partir de la base anomenada *cn=config*. Podem comprovar la configuració per defecte del dimoni `slapd` amb l'ordre:

```
# ldapsearch -Q -Y EXTERNAL -LLL -H ldapi:/// -b cn=config dn
```

La sortida de l'ordre és:

```
dn: cn=config

dn: cn=module{0},cn=config

dn: cn=schema,cn=config

dn: cn={0}core,cn=schema,cn=config

dn: cn={1}cosine,cn=schema,cn=config

dn: cn={2}nis,cn=schema,cn=config

dn: cn={3}inetorgperson,cn=schema,cn=config

dn: olcBackend={0}hdb,cn=config

dn: olcDatabase={-1}frontend,cn=config

dn: olcDatabase={0}config,cn=config

dn: olcDatabase={1}hdb,cn=config
```

Vegeu l'apartat "Configuració dinàmica del servei" d'aquesta unitat per a una explicació més detallada de cn=config.

2.3.4 Reconfigurar el programari

La instal·lació del paquet `slapd` en un sistema operatiu configurat correctament crea un servidor OpenLDAP amb una configuració per defecte. Ara bé, el nom de la màquina podria no estar correctament configurat, o potser es vol fer alguna variació sobre la configuració estàndard. Si es vol repassar la configuració de l'slapd podem fer servir l'ordre:

```
# dpkg-reconfigure slapd
```

En aquest cas, debconf no només pregunta la contrasenya d'administrador del directori, sinó que fa més preguntes:

1. La primera és si volem ometre la configuració inicial. S'ha de contestar que no i procedir a fer-la.
2. La segona pregunta vol establir la relació entre el nom DNS i el directori que s'està creant. Demana el nom DNS del domini que s'administrarà per construir-ne la base (arrel del DIT). Per al cas de l'exemple s'especificarà "empresa.com". Si el *hostname* està configurat correctament, l'instal·lador pot agafar directament el nom correcte.
3. En tercer lloc demana el nom de l'organització. Pot ser "Empresa Exemple S.L." o qualsevol altre nom que es vulgui donar a l'organització. És habitual escriure el mateix que al punt anterior.
4. Ara ens demana la contrasenya de l'administrador del domini i la confirmació.
5. El protocol LDAP estableix quina informació s'ha d'emmagatzemar, però no defineix com s'ha de fer. L'OpenLDAP permet fer servir dos *backends* per emmagatzemar les dades: BDB (Oracle Berkeley DB) i HDB (Hierarchical Data Base, una variant jeràrquica de BDB), que és la més recomanada.
6. En aquest punt, l'slapd ens demana si volem eliminar la base de dades en el cas de purgar el paquet (eliminar el programa i els seus fitxers de configuració). És tasca de l'administrador decidir si es pot arriscar a perdre tota la informació en cas de desinstal·lar el paquet o si prefereix, per contra, mantenir totes les dades.
7. Si s'està fent una reconfiguració, l'instal·lador detectarà que ja hi ha dades d'una instal·lació anterior. No és convenient barrejar dades de dues instal·lacions, per tant és recomanable acceptar moure la base de dades antiga a un altre lloc. És molt **important**, si s'havia fet anteriorment una reconfiguració de la base de dades de l'OpenLDAP i per tant s'havia creat una còpia de seguretat, **moure o eliminar** aquesta **còpia de seguretat** abans de fer la reconfiguració, ja que si no apareixerà l'error següent:

```
Moving old database directory to /var/backups:
Backup path /var/backups/unknown-2.4.23-7.2.ldapdb exists. Giving up...
```

Per finalitzar, no és necessària la compatibilitat amb l'LDAPv2, al menys en la majoria de les situacions que ens podrem trobar al llarg de la nostra feina com a administradors de directori.

2.3.5 Desinstal·lació del programari

Podem eliminar el programari OpenLDAP del sistema amb el gestor `aptitude`. Es pot utilitzar l'opció *remove* o l'opció *purge*. La diferència és que amb *remove* només eliminem el programari, mentre que amb *purge* eliminem també els fitxers de configuració i la base de dades del directori. Per exemple:

```
# aptitude purge slapd
```

Igualment, si hem instal·lat el paquet `ldap-utils` i el volem eliminar del sistema, podem fer-ho amb l'ordre `aptitude` i les opcions *remove* o *purge*:

```
# aptitude remove ldap-utils
```

2.3.6 Aturada i arrencada del servei

Per defecte, el sistema operatiu Debian 6 s'inicia en el nivell d'execució 2. Si comprovem els serveis que Debian inicia en aquest nivell, veurem que el dimoni `slapd` s'activa per defecte en iniciar-se el sistema:

```
root@servidor:~# ls /etc/rc2.d/S*slapd
/etc/rc2.d/S18slapd
```

Com qualsevol altre servei de Debian, es pot aturar el servei `slapd` mitjançant l'ordre *stop*:

```
root@servidor:~# service slapd stop
Stopping OpenLDAP: slapd.
```

I es pot tornar a activar manualment amb l'ordre *start*:

```
root@servidor:~# service slapd start
Starting OpenLDAP: slapd.
```

2.4 Configuració del servei de directori

Vegeu l'apartat "El format d'intercanvi de dades LDIF" d'aquesta unitat per a més informació sobre el format de fitxer LDIF i la seva funcionalitat.

Si bé la instal·lació del programari OpenLDAP no presenta cap dificultat, la correcta configuració del dimoni `slapd` requereix el coneixement i domini de

diversos conceptes, tant de l'estructura i funció dels directoris creats en temps d'instal·lació com del format de fitxer LDIF.

2.4.1 Configuració dinàmica del servei

Tradicionalment, la configuració del servidor OpenLDAP es realitzava mitjançant el fitxer `/etc/slapd.conf`. Es tractava d'una configuració estàtica, de manera que qualsevol canvi en la configuració comportava reiniciar el servei. Si el directori contenia moltes dades, aquesta aturada i posada en marxa necessitava un temps que podia ser inacceptable en determinats entorns.

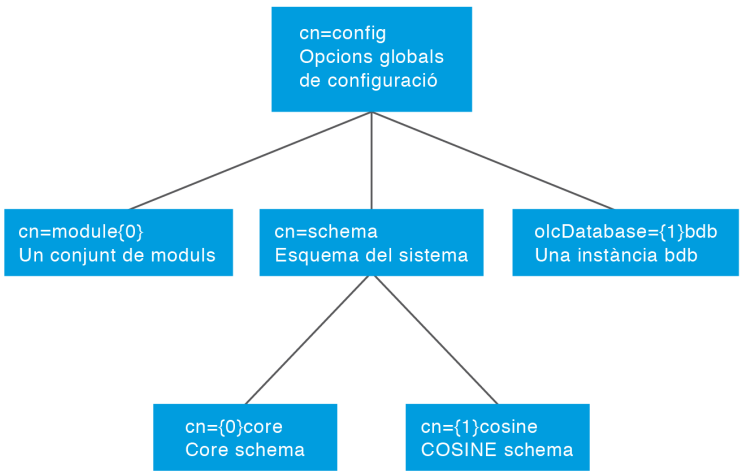
A partir de la versió 2.3, l'OpenLDAP permet la configuració dinàmica del dimoni `slapd`. Aquesta configuració és coneguda com a *run-time configuration* (RTC) o**cn=config**. La configuració de l'slapd s'emmagatzema en un directori especial LDAP amb un esquema i un DIT predefinits amb arrel a cn=config.

Hi ha entrades d'aquest directori que s'utilitzen per guardar les opcions de configuració global, les definicions d'esquemes, les definicions de *backends* i bases de dades i altres elements. La figura 2.5 mostra un exemple simplificat de configuració del directori.

Configuració estàtica o dinàmica

Tot i que en la versió 2.4 de l'OpenLDAP encara es pot fer servir el fitxer de configuració estàtica `/etc/slapd.conf`, en futures versions es retirarà el suport per a aquesta configuració. Debian 6 fa servir per defecte la configuració dinàmica `cn=config`.

FIGURA 2.5. DIT de configuració de l'slapd



El DIT de configuració de l'slapd té una estructura molt específica. La base de l'arbre és `cn=config`, que conté les opcions de configuració globals. La resta de configuracions estan a les entrades filles:

1. Mòduls carregats dinàmicament.
2. Definicions d'esquemes: l'entrada "cn=schema,cn=config" conté l'esquema del sistema (tot l'esquema que està programat directament al dimoni

slapd). Les entrades filles contenen els esquemes de l'usuari que poden ser afegits per configuració o de manera dinàmica durant l'execució del servei.

3. Configuració específica del *backend*.
4. Configuració específica de la base de dades: com a entrades filles d'aquesta entrada es defineixen els *overlays*.

Aquesta configuració s'emmagatzema en un *backend* LDIF. En el cas de Debian aquesta base de dades està situada al directori `/etc/ldap/slapd.d`. Amb l'ordre *tree* es pot visualitzar l'estructura de directoris i fitxers que conté el directori `/etc/ldap/slapd.d`:

Podeu instal·lar l'ordre *tree* mitjançant l'ordre**aptitude install tree**.

```
root@servidor:~# tree /etc/ldap/slapd.d/
/etc/ldap/slapd.d/
|-- cn=config
| |-- cn=module{0}.ldif
| |-- cn=schema
|   | |-- cn={0}core.ldif
|   | |-- cn={1}cosine.ldif
|   | |-- cn={2}nis.ldif
|   | |-- cn={3}inetorgperson.ldif
| |-- cn=schema.ldif
| |-- olcBackend={0}hdb.ldif
| |-- olcDatabase={0}config.ldif
| |-- olcDatabase={-1}frontend.ldif
| |-- olcDatabase={1}hdb.ldif
|-- cn=config.ldif

2 directories, 11 files
```

Es pot navegar pel directori `slapd.d`, però no és gens convenient fer les modificacions directament sobre els fitxers LDIF de configuració, sinó que s'han de fer servir les eines pròpies de l'OpenLDAP (*ldapmodify*,...). Igualment, per afegir nous esquemes al directori cal fer servir les ordres correctes; no n'hi ha prou de copiar els diferents fitxers de configuració a una carpeta de l'arbre de directoris.

Amb la configuració `cn=config` modifiquem el comportament o la configuració del servidor `slapd` simplement modificant les entrades necessàries al DIT.

Exemple de configuració del nivell de registre (log) del dimoni slapd

Per defecte, el dimoni `slapd` té el nivell de registre definit a "none" (ho podeu comprovar al fitxer `/etc/ldap/slapd.d/cn=config.ldif`). Aquesta configuració provoca que pràcticament no quedi registrada cap activitat. La definició del nivell de registre es fa a l'atribut *olcLogLevel* de l'entrada `cn=config` (configuració general). Aquesta entrada és la següent:

```
dn: cn=config
objectClass: olcGlobal
cn: config
olcArgsFile: /var/run/slapd/slapd.args
olcLogLevel: none
olcPidFile: /var/run/slapd/slapd.pid
olcToolThreads: 1
```

Per modificar l'entrada, creem un fitxer LDIF (en aquest exemple el creem al nostre directori de treball amb el nom `fitxer.ldif`) amb el contingut següent:

```
# Modificació del nivell de registre.
dn: cn=config
changetype: modify
replace: olcLogLevel
olcLogLevel: stats
```

El nou nivell de registre el definim amb `stats`. Els diferents nivells de registre estan definits al manual d'administració de l'OpenLDAP. Per realitzar la modificació, farem servir el fitxer LDIF que hem creat i l'ordre *ldapmodify* del paquet `ldap-utils`:

```
# ldapmodify -QY EXTERNAL -H ldapi:/// -f ~/fitxer.ldif

modifying entry "cn=config"
```

A partir d'aquest moment, i sense necessitat de reiniciar el servei, el nivell de registre de l'slapd quedarà definit a "stats" i el dimoni enregistrarà els esdeveniments al fitxer definit a l'atribut *olcLogFile* de l'objecte `cn=config`.

2.5 Manteniment del directori

Un directori correctament instal·lat i perfectament configurat no servirà de res si no es fa servir per emmagatzemar informació i consultar-la. Si s'ha fet la instal·lació del programari a partir dels repositoris Debian, el mateix programa d'instal·lació realitza una configuració bàsica, que crea l'arrel del DIT per a l'organització especificada durant la instal·lació i una entrada amb un usuari que permet l'administració del directori.

L'administració del directori es pot fer amb una eina gràfica, ja sigui una aplicació nativa per un determinat sistema operatiu o una aplicació web, que es pot fer servir des de qualsevol navegador. També és possible realitzar-la mitjançant utilitats de línia d'ordres, que es poden fer servir de manera interactiva des d'un terminal, ja sigui local o remot, o que es poden utilitzar en scripts per automatitzar tasques.

2.5.1 Utilitats de línia d'ordres

Les utilitats de línia d'ordres són principalment usades per programadors i administradors de sistemes com a eina de treball, especialment en sistemes operatius basats en Unix, ja sigui localment o amb una connexió remota. El programari openLDAP ofereix utilitats per poder administrar el directori des d'una interfície de línia d'ordres. A més de les utilitats proporcionades pel paquet **slapd**, el paquet de programari **ldap-utils** proporciona una sèrie d'eines per a línia d'ordres que permeten la interacció amb el directori.

Utilitats proporcionades per l'slapd

Les utilitats proporcionades pel paquet `slapd` permeten controlar el funcionament del servidor. Són les següents:

1. ***slappasswd***: es fa servir per generar una contrasenya vàlida per fer servir amb *ldapmodify* per als atributs *userPassword* dels objectes del directori, per a la directiva *rootpw* del fitxer de configuració slapd.conf (ja no es fa servir) o per a la directiva de configuració *olcRootPW* de la configuració slapd-config.
2. ***slapindex***: reindexa les entrades en la base de dades de l'slapd.
3. ***slapd***: Stand-alone LDAP Daemon, el servidor LDAP.
4. ***slapauth***: serveix per comprovar el funcionament de l'autenticació dels usuaris.
5. ***slapacl***: permet comprovar l'accés a una llista d'atributs, obrint el *backend* de l'slapd-config i mirant les directives *olcAccess*.
6. ***slapadd***: serveix per afegir entrades al directori mitjançant fitxers LDIF.
7. ***slapdn***: serveix per comprovar si una entrada de l'arbre segueix les normes dels esquemes definits.
8. ***slaptest***: comprova que la configuració del fitxer `slpad.conf` sigui correcte, però aquest mètode de configuració ja no es fa servir. L'ordre també es pot utilitzar per convertir fitxers `.schema` en `.ldif`.
9. ***slapcat***: aquesta ordre retorna en format LDIF tot el contingut de la base de dades LDAP. Per exemple:

```
root@servidor:~# slapcat
dn: dc=empresa,dc=com
objectClass: top
...
modifyTimestamp: 20120312011231Z
```

Utilitats proporcionades per ldap-utils

A més de les eines pròpies del servidor `slapd`, hi ha una sèrie d'utilitats per accedir com a client al servidor. Per fer-les servir s'ha d'instal·lar el paquet `ldap-utils`. Tot i que estan pensades per fer-se servir des d'un client, la seva instal·lació al servidor en facilitarà l'administració. Les més utilitzades són:

1. ***ldapdelete***: permet eliminar una entrada del directori.
2. ***ldappasswd***: canvia la contrasenya d'una entrada LDAP.
3. ***ldapexop***: permet iniciar operacions ampliades LDAP.

4. ***ldapcompare***: permet realitzar comparacions al directori LDAP.
5. ***ldapmodify***: modifica entrades al servidor LDAP.
6. ***ldapsearch***: és l'eina de cerca per al servidor LDAP.
7. ***ldapwhoami***: retorna l'usuari amb el qual s'està treballant. També permet comprovar la connexió d'usuari i el mot de pas.
8. ***ldapmodrdn***: utilitat per reanomenar entrades del directori.
9. ***ldapurl***: permet compondre o descompondre URIs LDAP.
10. ***ldapadd***: afegeix entrades al servidor LDAP.

Exemples d'ús de les utilitats de línia d'ordres

Un cop s'ha configurat correctament un directori OpenLDAP, és el moment d'estructurar i emmagatzemar la informació de l'organització, com ara la informació relativa a usuaris i grups.

Vegeu l'apartat "Escenari d'exemple" d'aquesta unitat per veure l'estructura del DIT a partir del qual crearem els elements.

Creació d'unitats organitzatives, grups i usuaris

Per crear les unitats organitzatives des de la línia d'ordres podem fer servir l'ordre ***ldapadd***. Aquesta ordre necessita un fitxer LDIF amb la informació que volem afegir al directori.

Tot i que amb una sola ordre LDIF es podria afegir tota la informació del directori, pot ser interessant a efectes didàctics o de depuració d'errors separar els diversos objectes que es volen inserir al directori en fitxers diferents. D'aquesta manera es pot entendre molt millor l'estructura de cada objecte i, a més, és més fàcil detectar errors de sintaxi o entrades duplicades.

Processament de fitxers LDIF sense aturada

Per evitar que s'aturi l'ordre ***ldapadd*** en afegir entrades si hi ha algun error al fitxer LDIF i que es continuïn afegint la resta d'entrades, cal fer servir el paràmetre ***-c***.

Per emmagatzemar objectes al directori cal definir una estructura. La manera habitual d'emmagatzemar els objectes en un directori és mitjançant **unitats organitzatives**. Les més comunes són **People** per als usuaris i **Groups** per als grups. Evidentment, cada situació o problemàtica requereix una solució adaptada a les necessitats particulars. L'administrador sempre té la potestat per decidir l'estructura organitzativa del DIT.

En l'exemple actual simplement s'insereixen dues unitats organitzatives, una per als usuaris anomenada *Personal* i una per als grups anomenada *Grups*.

Organització de l'entorn de treball

Quan es treballa en qualsevol sistema es fan servir fitxers temporals, diferents versions de documents, diferents versions de guions per automatitzar tasques i molts altres tipus de fitxers. Pot ser convenient crear una subcarpeta per emmagatzemar els fitxers LDIF que es van creant o recuperant del directori LDAP dins de la carpeta de treball personal pel simple fet de tenir-los localitzats i mantenir l'entorn mínimament ordenat.

En primer lloc crearem un fitxer LDIF (`ouPersonal.ldif`) amb el contingut següent per inserir-hi la unitat organitzativa Personal:

```
#Unitat organitzativa Personal
dn: ou=Personal,dc=empresa,dc=com
objectClass: organizationalUnit
ou: Personal
```

L'explicació és la següent:

1. La primera línia és un comentari que defineix el fitxer.
2. La segona línia defineix el DN o nom distintiu de l'objecte. Aquest nom serà el que identificarà l'objecte dins del directori. Per tant, ha de ser únic.
3. La tercera línia indica a quina classe pertany l'objecte. Es pot definir més d'una classe simplement afegint més línies.
4. La quarta línia defineix un atribut de l'objecte. En aquest cas és l'atribut *ou*. Sempre s'han de definir els atributs obligatoris; en cas contrari es produirà un error.

Per afegir l'entrada simplement s'executa l'ordre ***ldapadd*** assignant-li com a paràmetre el fitxer que s'ha creat. Per exemple:

```
root@servidor:~# ldapadd –x –D cn=admin,dc=empresa,dc=com –W –f ouPersonal.ldif

Enter LDAP Password: ********
adding new entry "ou=People,dc=example,dc=com"
```

Aquesta és l'explicació dels paràmetres de l'ordre:

1. ***-x***: defineix la forma de connexió al servidor com a simple i no fa servir la connexió SASL per defecte.
2. ***-D***: defineix amb quin usuari es realitzarà la connexió. Ha de ser un usuari amb els privilegis suficients per afegir una entrada a la part del directori corresponent. En l'exemple es connecta l'usuari administrador **cn=admin,dc=empresa,dc=com** especificant el seu DN.
3. ***-W***: fa que es demani la contrasenya per la línia d'ordres. En cas contrari s'hauria d'especificar la contrasenya a la línia d'ordres mitjançant l'opció *-w* o en un fitxer de contrasenyes mitjançant l'opció *-y*.
4. ***-f***: defineix la ruta del fitxer LDIF (en aquest cas **ouPersonal.ldif**) que conté la informació que volem afegir al directori.

Si tot ha funcionat correctament es pot afegir la unitat organitzativa següent, en el nostre cas, la que es farà servir per emmagatzemar els grups. Es crearà un fitxer que es pot anomenar *ouGrups.ldif*.

```
#Unitat organitzativa Grups
dn: ou=Grups,dc=empresa,dc=com
objectClass: organizationalUnit
ou: Grups
```

S'afegeix el fitxer al directori mitjançant l'ordre *ldapadd*.

Si es vol afegir més d'un objecte al directori en un sol fitxer LDIF només cal tenir en compte que els diferents objectes han d'estar separats per una línia en blanc. Així, les dues unitats organitzatives que s'han creat es podrien haver definit en un sol fitxer anomenat, per exemple, *ou.ldif*, de la manera següent:

```
#Unitat organitzativa Personal
dn: ou=Personal,dc=empresa,dc=com
objectClass: organizationalUnit
ou: Personal

#Unitat organitzativa Grups
dn: ou=Grups,dc=empresa,dc=com
objectClass: organizationalUnit
ou: Grups
```

Una vegada creades les unitats organitzatives es pot dir que ja està creada l'estructura bàsica del directori. És el moment de situar els grups (departaments) i usuaris (empleats). Sembla lògic començar per crear un grup abans de crear un usuari, ja que habitualment un usuari sempre pertany com a mínim a un grup.

Per crear el primer grup del directori es crea un fitxer que es pot anomenar *grupcomercials.ldif*, en el qual es defineix un grup anomenat *comercials*, que contindrà tots els comercials de l'empresa.

```
dn: cn=comercials,ou=Grups,dc=empresa,dc=com
objectClass: posixGroup
cn: comercials
gidNumber: 5001
```

En aquest cas, la classe d'objecte varia, ja que passa a ser un posixGroup (ja no és una organizationalUnit). En canviar la classe d'objecte, ja no és necessari definir un atribut *ou*, sinó que la nova classe demana els atributs *cn* (*commonName*) i *gidNumber*, que contindrà el GID (identificador de grup) amb el qual s'identificarà aquest grup LDAP al sistema.

És molt important fer servir un GID que no coincideixi amb els GID definits al fitxer `/etc/group` del sistema client. És per això que habitualment es fan servir nombres elevats.

Una vegada definit el grup, cal definir els usuaris. Es pot fer amb la definició de fitxers LDIF (que poden tenir el format *usuariMarcFreixas.ldif*) amb el contingut següent:

```
dn: uid=mfreixas,ou=Personal,dc=empresa,dc=com
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
uid: mfreixas
sn: Freixas
givenName: Marc
cn: Marc Freixas
displayName: Marc Freixas
uidNumber: 10001
gidNumber: 5001
userPassword: marcpassword
gecos: Marc Freixas
loginShell: /bin/bash
homeDirectory: /home/comercials/mfreixas
```

Cal analitzar aquesta definició, ja que té alguns punts importants:

1. En primer lloc cal destacar que es defineixen tres classes d'objecte. Això vol dir que aquest objecte haurà de contenir els atributs definits a les tres classes.

2. L'atribut *uid* definirà el nom de l'usuari que podrà iniciar sessió a un sistema client.

3. Existeixen molts atributs que donen informació sobre l'usuari: *sn* defineix el cognom, *givenName*, el nom propi, *displayName*, el nom que es mostrarà, i *cn*, el nom complet.

4. Els atributs *uidNumber* i *gidNumber* fan referència als identificadors d'usuari i de grup que aquest usuari tindrà en iniciar sessió des d'un sistema client. En aquest cas es fa que l'usuari "mfreixas" tingui com a grup primari el que hem creat, és a dir que és un comercial.

5. La contrasenya està en text pla en el fitxer LDIF. Sempre queda l'opció de no assignar-li cap contrasenya i després executar l'ordre de canvi de contrasenya de l'objecte.

6. *gecos*, *loginShell* i *homeDirectory* fan referència a la resta d'atributs posix necessaris, com són el nom complet, el *shell* de connexió al sistema i el directori de connexió al sistema o HOME.

És molt important fer servir un UID que no coincideixi amb els definits al fitxer **/etc/passwd** del sistema client. És per això que habitualment es fan servir nombres elevats.

Després d'afegir aquest darrer fitxer cal comprovar que tot està correctament situat. Per fer-ho es pot utilitzar l'ordre *slapcat* o realitzar una cerca sobre un objecte concret dels que s'han creat:

```
root@servidor:~# ldapsearch -x -LLL -b dc=empresa,dc=com 'uid=mfreixas' cn
    gidNumber

dn: uid=mfreixas,ou=Personal,dc=empresa,dc=com
cn: Marc Freixas
gidNumber: 5001
```

Els paràmetres que es fan servir són:

1. ***-x***: per fer l'autenticació simple en comptes de la SASL per defecte.

2. ***-LLL***: per mostrar el resultat de la cerca en format LDIF sense comentaris i sense mostrar la versió del format LDIF.

3. ***-b***: és molt important, és la base des d'on comença la cerca, és a dir, l'arrel del DIT.

4. ***uid=mfreixas***: és el filtre de cerca. Indica que s'està buscant aquest usuari i no un altre.

5. ***cn*** **gidNumber**: són els atributs concrets que es demanen a la cerca que retorni. Si no es defineix cap atribut es retornaran tots els atributs, que és el comportament per defecte.

És molt important crear els objectes en l'ordre correcte. Per exemple, no podem afegir l'objecte **dn: uid=mfreixas,ou=Personal,dc=empresa,dc=com** si

prèviament no hem afegit l'objecte **dn: ou=Personal,dc=empresa,dc=com**, ja que intentar afegir un objecte a una unitat organitzativa que encara no està creada donaria error.

Realització de cerques en el directori

Un directori està pensat principalment per realitzar operacions de cerca i consulta. Per això l'ordre *ldapsearch* és una de les més potents del conjunt d'utilitats de l'OpenLDAP. L'ajuda de l'ordre en mostra la seva sintaxi particular:

```
usage: ldapsearch [options] [filter [attributes...]]
where:
  filter     RFC 4515 compliant LDAP search filter
  attributes     whitespace-separated list of attribute descriptions
    which may include:
      1.1   no attributes
      *     all user attributes
      +     all operational attributes
```

Aquesta ordre accepta unes opcions, un filtre de cerca i uns atributs determinats. Les opcions modifiquen el comportament de l'eina i permeten diferents configuracions de la cerca. El filtre de cerca defineix allò que s'està buscant en el directori, i els atributs són els atributs que es retornaran en la cerca. Les principals opcions són les següents:

1. *-A* retorna només el nom dels atributs, no el seu valor.
2. *-b basedn* indica a partir de quina base es realitzarà la cerca.
3. *-c* estableix un mode d'execució continu, no s'atura en els errors.
4. *-L* mostra les respostes en format LDIFv1.
5. *-LL* mostra les respostes en format LDIF sense comentaris.
6. *-LLL* mostra les respostes en format LDIF sense comentaris ni versió.
7. *-P versió* indica la versió del protocol (per defecte és la 3).
8. *-d nivell* estableix el nivell de depuració LDAP a "nivell".
9. *-D binddn* indica el DN amb el qual es connectarà al directori per fer la cerca.
10. *-h host* indica el servidor LDAP a consultar.
11. *-H URI* indica l'adreça del servidor LDAP a consultar en format Uniform Resource Identifier, és a dir, "ldap://servidor/".
12. *-n* mostra el que es faria, però no ho fa. Serveix per comprovar que la sintaxi de l'ordre és correcta.
13. *-p port* és el port on està connectat el servidor LDAP.
14. *-Q* fa servir el mode silenciós SASL.

15. *-v* s'executarà de manera detallada, mostrant els diagnòstics per la sortida estàndard.
16. *-w passwd* contrasenya de connexió per a l'autenticació simple.
17. *-W* demana la contrasenya de connexió.
18. *-x* autenticació simple.
19. *-X authzid*: la identitat d'autorització SASL ("dn:<dn>" o "u:<user>").
20. *-y fitxer* permet obtenir la contrasenya a partir d'un fitxer.

A continuació es mostren alguns exemples d'ús de l'ordre.

Exemple de consulta de tot el directori

```
root@servidor:~# ldapsearch -LLL -D "cn=admin,dc=empresa,dc=com"
    -H ldap:/// -W -b dc=empresa,dc=com cn
Enter LDAP Password:
dn: dc=empresa,dc=com

dn: cn=admin,dc=empresa,dc=com
cn: admin

dn: ou=Personal,dc=empresa,dc=com

dn: ou=Grups,dc=empresa,dc=com

dn: cn=comercials,ou=Grups,dc=empresa,dc=com
cn: comercials

dn: uid=mfreixas,ou=Personal,dc=empresa,dc=com
cn: Marc Freixas

dn: cn=administratius,ou=Grups,dc=empresa,dc=com
cn: administratius
```

Aquesta consulta té aquests paràmetres:

- **-LLL** per obtenir un text de sortida sense comentaris.
- **-D**, que defineix amb quin objecte es fa la connexió al directori.
- **-H**, ldap:///, especifica que el servidor LDAP a consultar serà localhost.
- **-W** perquè demani la contrasenya.
- **-b**, que defineix la base de cerca.
- **cn** és l'atribut pel qual s'està preguntant.

Es pot observar que la resposta retorna els diferents objectes del directori i, en aquells objectes que contenen l'atribut *cn*, també aquest atribut. Si l'objecte no conté l'atribut *cn* (com és el cas de les unitats organitzatives Personal i Grups) no pot retornar l'atribut i només mostra el nom de l'objecte.

Es poden realitzar diferents tipus de consulta:

1. Igualtat: (givenname=Marc)
2. Presència: (givenname=*)

3. Subcadena: (givenname=J*)
4. Semblança: (givenname~=Joan)

Exemple de consulta d'un objecte concret (consulta d'igualtat)

```
root@servidor:~# ldapsearch -LLL -D "cn=admin,dc=empresa,dc=com"
    -H ldap:/// -W -b 'dc=empresa,dc=com' '(uid=mfreixas)'
Enter LDAP Password:
version: 1

dn: uid=mfreixas,ou=Personal,dc=empresa,dc=com
cn: Marc Freixas
displayName: Marc Freixas
gecos: Marc Freixas
gidNumber: 5001
givenName: Marc
homeDirectory: /home/comercials/mfreixas
loginShell: /bin/bash
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
objectClass: top
userPassword:: bWFyY3Bhc3N3b3Jk
sn: Freixas
uidNumber: 10001
uid: mfreixas
```

Aquesta consulta té els paràmetres:

- **-LL**, per tenir un text de sortida sense comentaris, però que mostrarà la versió del format LDIF.
- **-D**, que defineix amb quin objecte es fa la connexió al directori.
- **-H**, ldap:///, especifica que el servidor LDAP a consultar serà localhost.
- **-W** perquè demani la contrasenya.
- **-b**, que defineix la base de cerca.
- **"uid=mfreixas"** és el filtre de cerca que es fa servir, en aquest cas d'igualtat.

Es pot observar que la resposta retorna tots els atributs dels objectes que compleixen el filtre (en aquest cas només un objecte el compleix), ja que no es demana cap atribut en particular.

Els filtres de cerca ofereixen una gran versatilitat a l'hora de fer les consultes. Per fer-los servir se segueix una norma especificada a l'RFC 1558.

La forma d'expressar consultes complexes en l'LDAP és mitjançant la notació prefixa, en la qual s'avantposa l'operador lògic. Els operadors vàlids són:

- *!* per a NOT
- | per a OR
- *&* per a AND

Exemple d'ús d'operadors lògics

```
root@servidor:~# ldapsearch -LLL -D "cn=admin,dc=empresa,dc=com"
    -H ldap:/// -W -b 'dc=empresa,dc=com' '(|(ou=P*)(ou=G*))'
Enter LDAP Password:
dn: ou=Personal,dc=empresa,dc=com
objectClass: organizationalUnit
objectClass: top
ou: Personal

dn: ou=Grups,dc=empresa,dc=com
objectClass: organizationalUnit
objectClass: top
ou: Grups
```

Al filtre s'observa l'utilització d'un operador lògic, |, així com dues consultes de subcadena, P* i G*.

Es pot combinar l'ús d'operadors lògics i d'atributs per obtenir únicament els resultats que ens interessin.

Exemple de consulta d'un atribut determinat

```
root@servidor:~# ldapsearch -LLL -D "cn=admin,dc=empresa,dc=com"
    -H ldap:/// -W -b 'dc=empresa,dc=com' '(&(cn=Marc*)(sn=F*))'
    uid
Enter LDAP Password:
dn: uid=mfreixas,ou=Personal,dc=empresa,dc=com
uid: mfreixas
```

La consulta demana l'atribut uid per als objectes el cn dels quals comença per "Marc" i l'sn dels quals comença per "F".

2.5.2 Utilitat gràfica phpLDAPadmin

Tot i que les utilitats bàsiques per a la línia d'ordres permeten realitzar totes les operacions necessàries en el servidor OpenLDAP, sempre resulta més còmode fer servir una interfície gràfica per a l'administració del directori. Hi ha diverses opcions d'eines d'administració gràfica, però la utilització d'un entorn web és molt recomanable, ja que permet l'accés al directori des de qualsevol ordinador de la xarxa amb un simple navegador.

Una de les opcions més conegudes i provades com a eina d'administració gràfica basada en web per a l'OpenLDAP és l'aplicació `phpLDAPadmin`, també coneguda com a **PLA**.

PLA és una eina d'administració per al LDAP que té les característiques següents:

1. Permet administrar entrades en un servidor LDAP, és a dir, permet crear, modificar i esborrar les entrades.

2. És compatible amb els estàndards oberts, de manera que pot treballar amb entrades o registres de qualsevol servidor LDAP que compleixi la normativa.

3. Està pensada per ser usada pels administradors, que ja tenen cert coneixement de l'LDAP.

4. És flexible: pot ser configurada per adaptar-se a l'entorn sense necessitat de canviar el codi.

PLA ha estat dissenyada perquè la facin servir els administradors. Per tant, és útil tant per administrar tota la base de dades LDAP com només una part. Tot i que la resta dels usuaris poden fer servir l'aplicació per, per exemple, editar les seves entrades o registres al directori, l'estructura, la terminologia i el procés a seguir no resulten senzills.

Instal·lació de phpLDAPadmin a Debian

La instal·lació de `phpLDAPadmin` a Debian és molt senzilla. Com que el paquet es troba als repositoris, només caldrà instal·lar-lo mitjançant l'ordre corresponent:

```
root@servidor:~# aptitude install phpldapadmin
```

Com que es tracta d'una aplicació per a entorn web, les dependències del paquet fan que també s'instal·len el servidor web Apache i el suport PHP per al servidor. El pas següent és comprovar si tot ha anat correctament i si el servidor Apache està connectat al port esperat, el 80:

```
root@servidor:~# nmap -p 80 localhost
...
PORT STATE SERVICE
80/tcp open http
...
```

També es pot comprovar que funciona correctament amb una simple consulta des del navegador web, com mostra la figura 2.6.

FIGURA 2.6. Pàgina per defecte del servidor web Apache

Ara cal comprovar que el PHP –que és el llenguatge amb el qual s'ha programat l'aplicació– s'ha instal·lat correctament.

```
root@servidor:~# php -v
PHP 5.3.3-7+squeeze8 with Suhosin-Patch (cli) (built: Feb 10 2012 14:12:26)
...
```

Evidentment, no ens serveix de gaire tenir l'Apache i el PHP instal·lats si el mòdul d'Apache per fer servir el PHP no està habilitat. Una simple ullada al directori de configuració de mòduls de l'Apache ens permet comprovar-ho:

```
root@servidor:~# ls /etc/apache2/mods-enabled/ |grep php
php5.conf
php5.load
```

En el cas que el mòdul no estigui habilitat, s'ha d'habilitar amb l'ordre d'Apache, en aquest cas *a2enmod*. Quan s'habilita un mòdul al servidor web Apache, cal reiniciar el servidor per tornar a carregar els fitxers de configuració:

```
root@servidor:~# a2enmod php5
Enabling module php5.
Run '/etc/init.d/apache2 restart' to activate new configuration!
root@servidor:~# /etc/init.d/apache2 restart
Restarting web server: apache2 ... waiting .
```

L'aplicació phpLDAPadmin se situa al directori `/usr/share/phpldapadmin/` i la seva configuració a `/etc/phpldapadmin/`, on el fitxer més important és `/etc/phpldapadmin/apache.conf`, ja que és aquí on hi ha la configuració perquè Apache pugui servir les pàgines de l'aplicació. De fet, aquest fitxer està enllaçat des del directori de configuració d'Apache, que és `/etc/apache2/conf.d/`.

```
root@servidor:~# ls -l /etc/apache2/conf.d/php*
lrwxrwxrwx 1 root root 29 4 abr 13:16 /etc/apache2/conf.d/phpldapadmin -> /etc/
    phpldapadmin/apache.conf
```

En iniciar el servei Apache, aquest processa tots els fitxers que hi ha al directori `/etc/apache2/conf.d/` i carrega tots els llocs web que serveix.

Al fitxer de configuració de phpLDAPadmin per a Apache `/etc/phpldapadmin/apache.conf` és on es defineix si phpLDAPadmin apareixerà com a àlies (és l'opció per defecte) o si es crearà un VirtualHost.

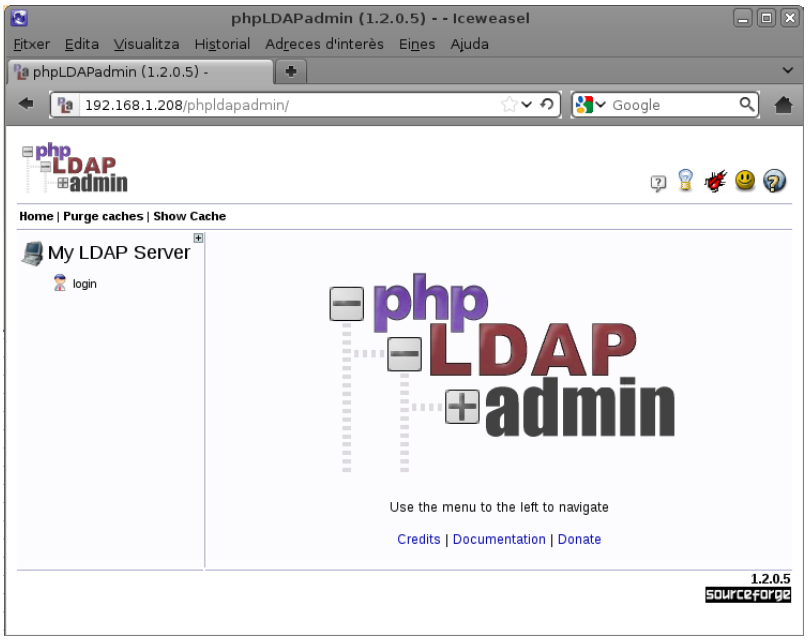
1. Si es crea un àlies, l'adreça web per accedir a l'aplicació serà http://servidor.empresa.com/phpldapadmin, on es pot canviar el nom del servidor per l'adreça IP.

2. Si pel contrari es crea un VirtualHost, l'adreça podria ser de l'estil http://ldap.empresa.com.

Configuració de phpLDAPadmin a Debian

La instal·lació de phpLDAPadmin (PLA) és molt senzilla i la configuració encara ho és més. De fet, el programari ja funciona sense haver de fer cap configuració, simplement amb la instal·lació per defecte. Tan sols s'ha d'accedir a l'adreça per defecte i l'aplicació ja està disponible, com s'aprecia en la figura 2.7.

FIGURA 2.7. phpLDAPadmin en una finestra d'un navegador



Tot i estar disponible des d'un principi, sí que és necessari configurar certs paràmetres perquè funcioni correctament. El fitxer de configuració és `/etc/phpldapadmin/config.php` i cal editar-lo.

Les diferents opcions de configuració estan explicades en el mateix fitxer mitjançant comentaris. La millor opció és cercar dins del fitxer les directives que es volen configurar. Les més interessants són aquestes:

1. **$config->custom->appearance['language'] = 'auto';** Permet configurar l'idioma de l'aplicació. Si està en mode "auto", l'aplicació intentarà determinar l'idioma en funció de l'idioma del sistema. En cas de voler forçar un idioma, els disponibles són *ct*, *de*, *en*, *es*, *fr*, *it*, *nl* i *ru*.

2. **$servers->setValue('server','base',array('dc=example,dc=com'));** Aquesta línia defineix el nom de domini que es mostra en la vista d'arbre, a la part esquerra de la pantalla, com es pot observar en la figura 18. Si no està definit, l'aplicació la intenta detectar, però és convenient afegir el domini que ha estat configurat. En l'exemple actual seria: **$servers->setValue('server','base',array('dc=empresa,dc=com'));** Com a orientació, aquesta línia és la 283 del fitxer de configuració.

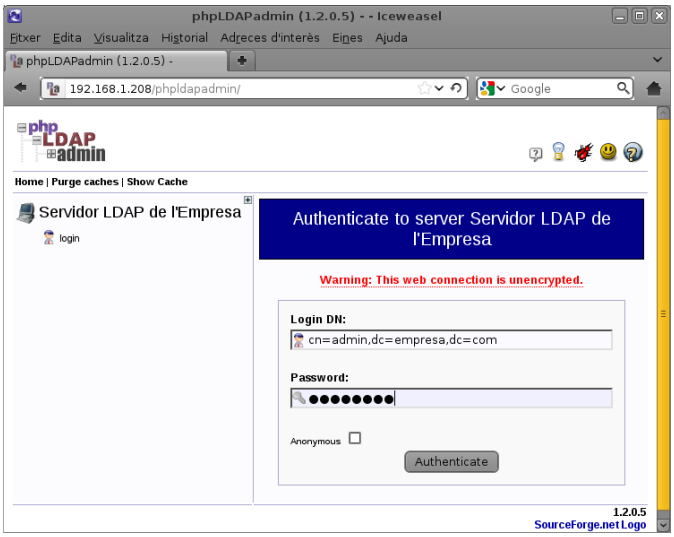
3. **$servers->setValue('login','bind_id','cn=admin,dc=example,dc=com');** Aquesta directiva defineix l'usuari mitjançant el qual phpLDAPadmin es connecta al servidor OpenLDAP. Si es defineix, en accedir a l'aplicació, apareix directament, per la qual cosa només cal escriure la contrasenya. Pot ser una bona mesura de seguretat no definir aquesta entrada o definir un usuari fals. La directiva és a la línia 306. Per a l'exemple que s'ha fet servir al llarg de la unitat caldria configurar: **$servers->setValue('login','bind_id','cn=admin,dc=empresa,dc=com');**. Si s'ha definit aquesta entrada, també es pot definir la contrasenya a la directiva següent: **$servers->setValue('login','bind_pass','secret');** on "secret"

s'hauria de canviar pel mot de pas de l'administrador.

4. **$servers->setValue('server','name','Servidor LDAP de l'Empresa');** En la línia 270 es pot incloure una descripció per identificar el directori al qual es connecta aquesta aplicació.

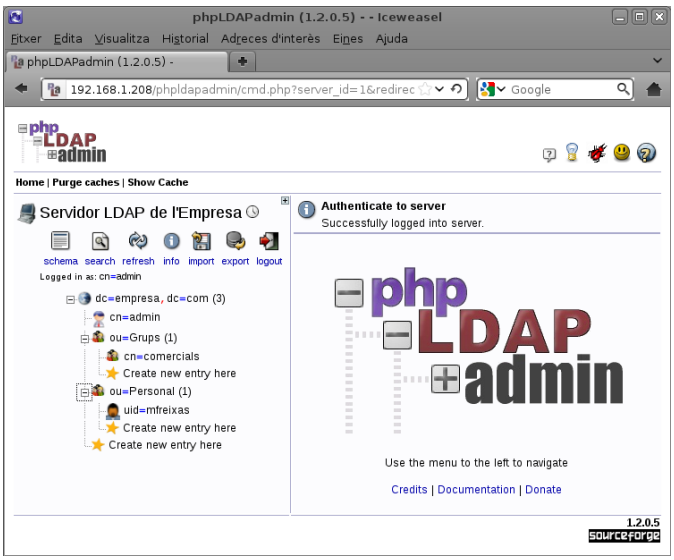
Una vegada ha estat configurada l'aplicació, s'hi pot accedir mitjançant la pantalla de connexió.

FIGURA 2.8. Pantalla de connexió de phpLDAPadmin



En la figura 2.8 es pot observar que al camp *Login DN* ja apareix definit l'usuari mitjançant el qual es farà la connexió al servidor. Tan sols queda escriure la contrasenya per poder-hi accedir. També hi ha una casella que podem activar per connectar-nos de manera anònima al directori, sempre i quant aquest ho permeti (l'OpenLDAP ho permet per defecte), que és útil per fer cerques en el directori.

FIGURA 2.9. Pantalla principal de phpLDAPadmin



Una vegada s'inicia la sessió, la pantalla de l'aplicació queda dividida, com s'aprecia en la figura 2.9. En la part esquerra, a més d'un menú amb algunes

opcions, hi apareix un arbre amb tots els objectes del directori. En la figura també s'aprecia que els objectes que ja hi ha al directori estan disponibles per ser consultats o modificats.

Administració del directori mitjançant phpLDAPadmin

L'ús de **phpLDAPadmin** facilita molt les tasques de consulta, creació i modificació d'entrades al directori, tot i que es poden seguir fent servir fitxers LDIF per fer aquestes operacions.

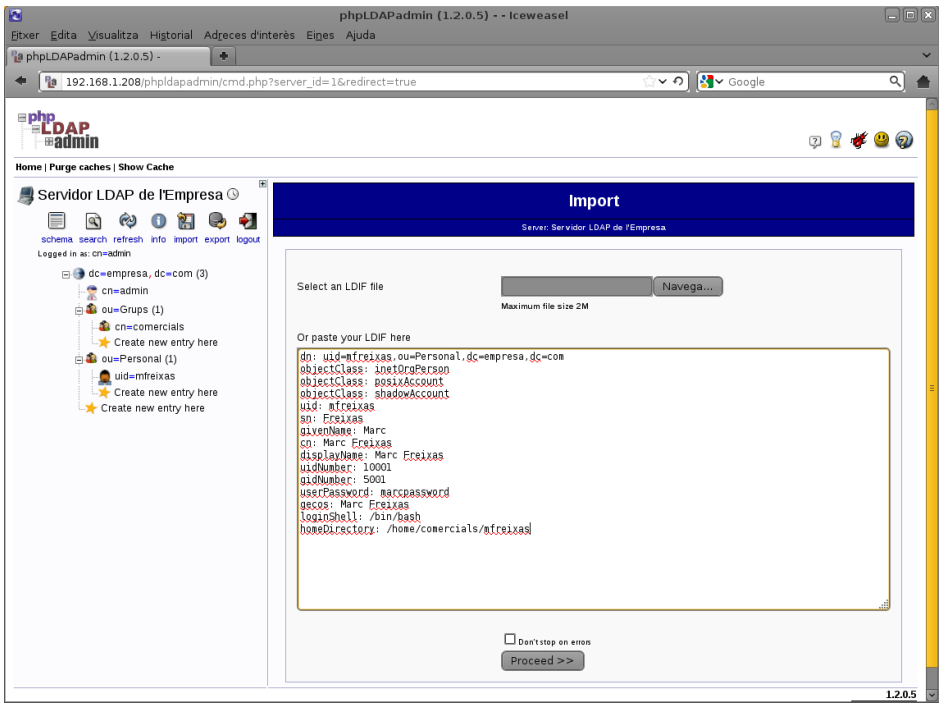
Ús de fitxers LDIF

Com que el format de fitxer per defecte per definir entrades i operacions en un servidor LDAP és el LDIF, phpLDAPadmin permet la importació d'un fitxer o d'un text en aquest format. Per fer-ho, simplement cal fer clic sobre l'opció *import* del menú. Quan se selecciona aquesta opció es pot afegir el contingut LDIF al directori de dues maneres:

1. Seleccionant un fitxer LDIF del disc dur.

2. Escrivint el text amb format LDIF directament en l'espai disponible.

FIGURA 2.10. LDIF a phpLDAPadmin



En la figura 2.10 es mostra com afegir un fitxer LDIF.

No només es pot importar un fitxer, sinó que també es pot exportar el contingut del directori, d'una part del directori, d'una cerca concreta o una entrada determinada a format LDIF.

FIGURA 2.11. Exportació de dades a phpLDAPadmin

Quan es fa clic a la icona *export* del menú, apareix la pantalla de la figura 2.11, on es pot definir la cerca a realitzar. Els objectes se cercaran des del punt que es defineix a *Base DN*. Habitualment aquest punt és l'arrel de l'arbre. També es defineix si es buscarà en els fills d'aquest objecte mitjançant la directiva *Search Scope*. Es defineix el filtre de la cerca, que delimita quins objectes es mostren, i els atributs d'aquests objecte que ens interessen. Per acabar, també es pot definir el format d'exportació del fitxer, ja que no està limitat al LDIF.

Modificació d'una entrada del directori

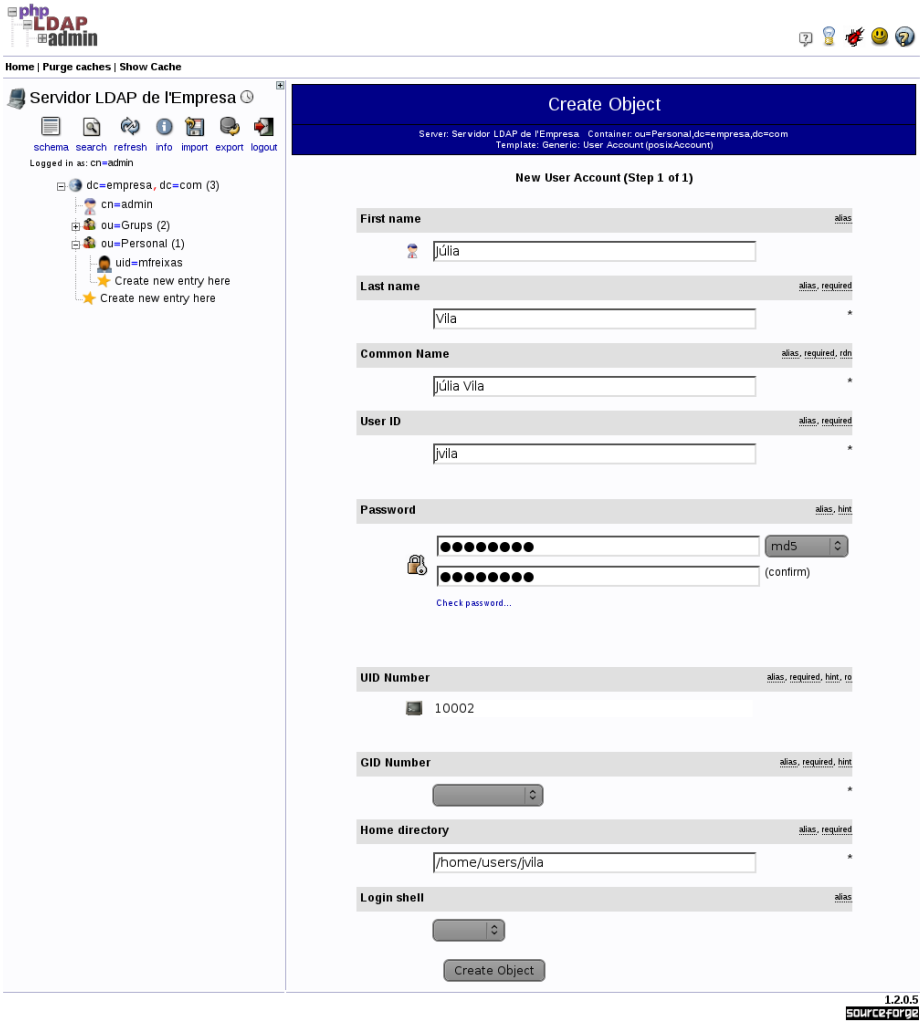
Per modificar una entrada del directori únicament cal fer clic al seu damunt. Quan es fa així, l'aplicació sempre pregunta quina plantilla es vol utilitzar per editar l'entrada. Depenent de la plantilla que se seleccioni es visualitzen, i per tant es poden editar, uns atributs diferents. Per exemple, per a l'entrada "mfreixas", que és un usuari, si es fa servir la plantilla *address book entry* es pot accedir a o modificar la fotografia de l'usuari, però si es fa servir la plantilla *default* no es té accés a aquest atribut (tot i que sí que es podrà accedir a d'altres, com per exemple *homeDirectory*).

A més, una vegada s'està editant una entrada, a la part superior apareixen més opcions. Es permet canviar el nom de l'entrada, copiar o moure l'entrada, canviar la plantilla que s'està fent servir, exportar l'entrada, comparar-la, afegir-li un atribut o esborrar-la.

Creació d'una entrada en el directori

Per crear una entrada només cal fer clic sobre el lloc de l'arbre on es vol crear, on diu *Create new entry here*, o, si l'objecte accepta entrades fill, quan s'està editant l'objecte, fer clic a *Create a child entry*. En tots dos casos, la icona és una estrella groga de cinc puntes. En fer-hi clic apareix un formulari com el de la figura 2.12.

FIGURA 2.12. Creació d'un objecte a phpLDAPadmin



Igual que passa quan s'està editant una entrada, en crear-ne una de nova l'aplicació pregunta quina és la plantilla a usar per al procés de creació, i quins seran, per tant, els atributs que contindrà l'entrada. Depenent de quina plantilla i per tant de quina classe d'objecte s'ha triat, uns atributs seran obligatoris, i d'altres, opcionals.

3. Integració del servei de directori

Un servei de directori ofereix accés a un o més directoris. La informació continguda en aquests directoris pot ser consultada amb eines concretes, però també pot ser utilitzada per part d'aplicacions client configurades específicament per a aquest propòsit.

El cas més habitual és fer servir el directori per fer una gestió centralitzada dels usuaris d'una organització i, més concretament, per muntar un sistema d'autenticació centralitzada. El directori contindrà les dades que l'organització hagi decidit, així com aquelles necessàries perquè els sistemes i les aplicacions clients puguin autenticar als usuaris.

3.1 Autenticació centralitzada amb un servei de directori

Poder realitzar consultes en un directori amb una interfície gràfica o amb la línia d'ordres és útil, però el que és realment interessant és que diferents aplicacions puguin fer servir les dades d'un únic directori per tal de no haver de tenir informació duplicada en diferents llocs i possibilitar el manteniment únic i centralitzat d'aquestes dades.

Una de les utilitats més comunes d'un servei de directori és l'autenticació centralitzada dels usuaris d'una xarxa en els diferents serveis (inici de sessió en el sistema, accés a un servei FTP, accés a un gestor de continguts...). La implementació de l'autenticació centralitzada d'usuaris a partir del directori té diversos avantatges:

1. Els noms d'usuaris i contrasenyes són únics en tots els entorns.
2. Se simplifiquen les eines de gestió i administració de comptes d'usuaris (altes i baixes, canvis de contrasenyes...).
3. Permet implementar sistemes d'alta disponibilitat d'una manera més fàcil.
4. Es poden establir polítiques de seguretat coherents i universals.

Un sistema d'alta disponibilitat és un sistema dissenyat per assegurar una operació continuada sense errors durant un determinat període de temps.

És evident que en una organització en la qual els usuaris han d'autenticar-se en diferents serveis és convenient mantenir la informació centralitzada i unificada per evitar duplicitats al sistema d'informació de l'empresa i facilitar-ne el manteniment.

Igualment, el directori LDAP pot contenir totes aquelles dades que l'organització cregui necessàries, com ara informació sobre edificis, departaments, ordinadors,

telèfons, impressores..., a banda de la informació necessària per implementar l'autenticació centralitzada.

3.2 Autenticació a Debian 6 amb l'OpenLDAP

El sistema operatiu Debian 6 fa per defecte l'autenticació bàsica d'usuaris mitjançant uns fitxers determinats: `/etc/passwd`, `/etc/group`, `/etc/shadow` i d'altres. Aquests fitxers són consultats durant el procés d'inici de sessió per autenticar un usuari i obtenir les dades relatives al compte de l'usuari per crear el seu context inicial, com ara el directori de treball. Aquest procés fa servir dues biblioteques: la PAM i l'NSS.

La PAM (Pluggable Authentication Module) és una biblioteca d'autenticació genèrica que qualsevol aplicació pot utilitzar per validar usuaris, utilitzant múltiples esquemes d'autenticació alternatius (fitxers locals, Kerberos, LDAP, dispositius biomètrics...). Aquesta biblioteca és utilitzada pel procés d'inici de sessió (*login*) del sistema operatiu per comprovar si les credencials introduïdes per l'usuari (nom i contrasenya) són correctes.

Mòduls PAM

Als començaments d'Unix, per autenticar un usuari calia que aquest introduís una contrasenya. Llavors el sistema comprovava si la contrasenya coincidia amb la contrasenya xifrada emmagatzemada a `/etc/passwd` (més tard a `/etc/shadow`). La idea era que un usuari era realment aquest usuari només si podia entrar la seva contrasenya secreta correctament.

Amb l'evolució dels sistemes informàtics s'han popularitzat noves formes d'autenticació d'usuaris, entre d'altres: fitxers alternatius i més complicats que l'arxiu `/etc/passwd`, lectors d'empremtes dactilars, reconeixement òptic de cares, certificats USB, claus intel·ligents...

El problema és que cada vegada que es desenvolupa un nou esquema d'autenticació requereix que tots els programes que el volen fer servir (login, ftpd...) s'hagin de reescriure.

La PAM ofereix una capa intermèdia entre l'usuari i l'aplicació en el moment de l'autenticació. Aquesta capa intermèdia està basada en un sistema modular que li permet oferir diferents funcionalitats.

L'NSS (Name Service Switch) és una interfície genèrica per obtenir els paràmetres d'un compte (com l'identificador d'usuari, l'identificador de grup, el *shell* inicial, el directori de treball...), que és utilitzada pel procés de *login* per crear el procés d'inici de sessió de l'usuari.

Els paràmetres d'un compte d'usuari es poden obtenir tradicionalment de l'arxiu `/etc/passwd`. Aquest fitxer és de text pla i conté una línia per a cada usuari donat d'alta en el sistema. Cada línia està formada per set camps separats per dos punts:

1. El nom d'usuari.
2. *x* si s'usen *shadow passwords* (el més habitual), si no, la contrasenya codificada.

3. L'identificador numèric de l'usuari (UID).
4. L'identificador numèric del grup primari de l'usuari (GID).
5. El nom complet de l'usuari i, opcionalment, informació addicional.
6. El directori d'inici (també anomenat *directori de connexió* o *directori de treball*) de l'usuari.
7. L'intèrpret d'ordres per defecte de l'usuari.

L'avantatge fonamental d'aquestes dues biblioteques és que es poden reconfigurar dinàmicament mitjançant fitxers, sense necessitat de tornar a compilar les aplicacions que les utilitzen. Per tant, l'únic que es necessita és configurar les dues biblioteques perquè utilitzin el servidor LDAP a més dels fitxers locals (`/etc/passwd`...) de cada sistema.

3.2.1 Comprovacions preliminars

Per instal·lar el suport LDAP per a l'autenticació d'usuaris caldrà comprovar la correcta instal·lació bàsica del sistema client, és a dir, la configuració IP, el nom de la màquina i la connexió amb el servidor OpenLDAP. Una forma ràpida de fer-ho és, en primer lloc, confirmar que el ping retorna del client al servidor:

```
root@client:~# ping servidor.empresa.com
PING servidor.empresa.com (192.168.1.208) 56(84) bytes of data.
64 bytes from servidor.empresa.com (192.168.1.208): icmp_req=1 ttl=64 time
    =0.638 ms
64 bytes from servidor.empresa.com (192.168.1.208): icmp_req=2 ttl=64 time=1.24
     ms
64 bytes from servidor.empresa.com (192.168.1.208): icmp_req=3 ttl=64 time
    =0.697 ms
^C
--- servidor.empresa.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2005ms
rtt min/avg/max/mdev = 0.638/0.859/1.244/0.274 ms
```

Vegeu l'apartat "Instal·lació de l'OpenLDAP" d'aquesta unitat per a més detalls sobre la instal·lació i configuració d'un servidor OpenLDAP.

Una segona comprovació és fer una consulta a la informació del directori des del client al servidor OpenLDAP. Per fer-ho, en primer lloc cal instal·lar en la màquina client les utilitats bàsiques LDAP per poder realitzar la consulta:

```
root@client:~# aptitude install ldap-utils
```

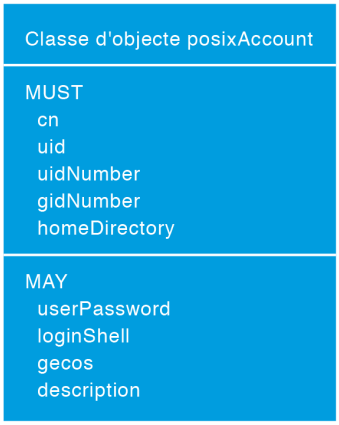
Una consulta molt útil sobre el servidor OpenLDAP és la que retorna els identificadors (UID) dels usuaris donats d'alta. A més de comprovar la connexió amb el servidor, també proporciona la llista d'usuaris que haurien de poder iniciar sessió a la màquina client:

```
root@client:~# ldapsearch -LLL -x -D 'cn=admin,dc=empresa,dc=com' -W -H ldap://
    servidor/ -b 'dc=empresa,dc=com' objectClass=posixAccount uid
Enter LDAP Password:
dn: uid=mfreixas,ou=Personal,dc=empresa,dc=com
uid: mfreixas
```

La consulta anterior dóna com a resultat un únic usuari, que és el que podrà iniciar sessió.

El filtre de cerca que s'ha utilitzat demana els objectes del tipus**posixAccount**, ja que aquest tipus d'objecte defineix una sèrie d'atributs que són necessaris per poder fer servir l'objecte per autenticar usuaris d'un sistema Linux. Els atributs que defineix aquesta classe d'objecte, tant els obligatoris (MUST) com els opcionals (MAY), es mostren en la figura 3.1. La classe d'objecte posixAccount proporciona els mateixos atributs que es poden trobar al fitxer `/etc/passwd`.

FIGURA 3.1. Classe d'objecte posixAccount



La classe que permet guardar els atributs necessaris per a un usuari d'un sistema operatiu de tipus Unix és posixAccount. Per poder fer servir un objecte del directori per fer l'autenticació d'un usuari, aquest ha de ser de la classe posixAccount.

Una vegada s'ha comprovat que el sistema client és capaç d'accedir a la informació del directori caldrà configurar el sistema perquè en comptes d'utilitzar els fitxers clàssics per validar els usuaris faci una consulta al servidor LDAP.

3.2.2 Instal·lació del programari client LDAP

El programari necessari per configurar un sistema operatiu Debian GNU/Linux com a client d'un servidor OpenLDAP està inclòs als repositoris de programari oficials. L'ordre d'instal·lació és:

```
1 root@client:~# apt-get install libpam-ldap libnss-ldap nscd
```

Problemes amb el dimoni nscd

En algunes ocasions, el dimoni nscd pot provocar problemes durant la configuració. Es pot aturar amb l'ordre **/etc/init.d/nscd stop**, configurar el sistema i reiniciar-lo amb **/etc/init.d/nscd start**.

Aquests paquets instal·len el suport LDAP tant per al mòdul PAM com per a l'NSS, i també el dimoni `nscd` (*name service cache daemon*), que fa de cau de noms per al servei NSS per accelerar les consultes.

Després de la instal·lació caldrà contestar algunes preguntes per realitzar la configuració inicial dels paquets.

Configuració inicial de libnss-ldap

Per configurar el paquet `libpam-ldap` es demanen una sèrie de dades:

1. **URI (Uniform Resource Identifier)** del servidor LDAP. En aquest punt s'ha d'introduir una cadena de la forma ldap://<hostname_o_ip>:<port>/ com pot ser "ldap://ldap.empresa.com/" o "ldap://192.168.1.208/". Opcionalment es pot incloure el port on es connecta el servidor, ldap://192.168.1.208:389/, però si no s'especifica, es fa servir el port per defecte.

   S'ha de fer servir ldap:// per una connexió estàndard per TCP, *ldaps: si la connexió està xifrada per SSL i* ldapi: si es fa per IPC (fa servir un sòcol Unix). Aquesta última és l'opció per defecte i funciona si el client i el servidor són la mateixa màquina. El més habitual és fer servir l'LDAP per TCP.

2. **DN (nom distintiu)** de la base del DIT. Com que la majoria de llocs fan servir la notació basada en DNS és una bona opció seguir la mateixa norma. En qualsevol cas, la base aquí definida ha de ser la mateixa que es va definir en el servidor LDAP i ha d'estar escrita en la notació correcta: *dc=empresa,dc=com*.

3. **Versió LDAP**que s'utilitza. Excepte en situacions molt excepcionals (per mantenir la compatibilitat amb instal·lacions antigues), es fa servir sempre la versió 3 del protocol.

4. **Compte LDAP per a l'administrador**. Quan es fa la configuració inicial del servidor LDAP es creen dues entrades: la base i l'administrador. Aquest administrador és el que crea l'instal·lador del dimoni slapd per defecte, i és de l'estil *cn=admin,dc=empresa,dc=com*. El DN de l'entrada coincideix amb la base que s'ha definit en el pas anterior de la instal·lació.

5. **Contrasenya LDAP per a l'administrador**. La contrasenya que es va definir en el moment de la instal·lació i configuració del dimoni slapd per a l'administrador.

Una vegada entrades aquestes dades procedim a configurar el paquet libpam-ldap.

Configuració inicial de libpam-ldap

El paquet `libpam-ldap` realitza dues preguntes durant la configuració bàsica:

1. **Permetre a l'administrador** de l'LDAP actuar com a **administrador local**. Es pot decidir si es vol concedir permisos a l'administrador LDAP per realitzar també canvis en els comptes locals del sistema. A aquesta qüestió es contesta habitualment que **no**.

2. Requisit d'**inici de sessió**per accedir al directori. La configuració per defecte del directori OpenLDAP permet la consulta sense validació (la

validació s'anomena *bind* a l'LDAP) de les dades del directori. **No és necessària cap autenticació**, però si s'ha canviat la configuració per no permetre una consulta anònima, en aquest punt es pot definir un usuari amb el qual poder fer les consultes. L'usuari administrador sempre podrà fer-les.

3.2.3 Configuració de l'autenticació LDAP

Amb el programari instal·lat i configurat, és el moment de configurar el sistema perquè faci servir els nous mòduls d'autenticació.

El primer fitxer que cal modificar és `/etc/ldap/ldap.conf`. Aquest fitxer únicament conté comentaris. Serveix per definir les opcions per defecte per a tots els clients LDAP.

Les directives que cal modificar són:

1. BASE, que defineix la base del DIT.
2. URI, que defineix l'adreça en la qual es connecta el servidor OpenLDAP.

Exemple de fitxer ldap.conf

El fitxer de configuració general LDAP per a Debian serà un fitxer com aquest:

```
#
# LDAP Defaults
#

# See ldap.conf(5) for details
# This file should be world readable but not world writable.

BASEdc=empresa,dc=com
URIldap://192.168.1.208
```

Les diferents opcions de configuració del fitxer es poden consultar en la seva pàgina del manual:

```
root@client:/etc# man ldap.conf
```

El fitxer `nsswitch.conf` està situat en el directori `/etc`. Aquest fitxer configura el Name Service Switch. Com que hem instal·lat el suport LDAP per a l'NSS, caldrà configurar l'NSS perquè a més de les bases de dades tradicionals (bàsicament `/etc/passwd`) faci servir l'LDAP per a les seves operacions.

El fitxer per defecte és:

```
# Example configuration of GNU Name Service Switch functionality.
# If you have the 'glibc-doc-reference' and 'info' packages installed, try:
# 'info libc "Name Service Switch"' for information about this file.

passwd: compat
group: compat
shadow: compat

```

```
hosts: files mdns4_minimal [NOTFOUND=return] dns mdns4
networks: files

protocols: db files
services: db files
ethers: db files
rpc: db files

netgroup: nis
```

Per configurar la nova base de dades caldrà modificar tres línies i informar l'NSS que, a més de les bases de dades tradicionals, també ha de fer servir l'LDAP:

```
passwd: compat ldap
group: compat ldap
shadow: compat ldap
```

Com que s'ha canviat la configuració NSS, és una bona pràctica reiniciar el dimoni `nscd` que fa de memòria cau de noms:

```
root@client:/etc# /etc/init.d/nscd restart
Restarting Name Service Cache Daemon: nscd.
```

Ara que l'NSS ja sap on obtenir els atributs del usuaris, cal configurar l'LDAP perquè realitzi l'autenticació en el servidor LDAP. La llibreria PAM es configura en el directori `/etc/pam.d/`. En aquest directori hi ha diversos fitxers, molts dels quals són específics per a una aplicació determinada que desitja fer servir aquesta llibreria per realitzar l'autenticació dels usuaris.

Els fitxers de configuració de la PAM són:

1. **common-auth**: on es defineix com saber si la contrasenya d'un usuari és correcta.
2. **common-account**: on es defineix com saber si un compte d'usuari és vàlid, comprovant si ha expirat o si s'hi ha d'aplicar alguna restricció de temps.
3. **common-password**: on es defineix com pot un usuari canviar la seva contrasenya.
4. **common-session**i **common-session-noninteractive**: on es defineix la configuració de sessió per a l'usuari amb diverses accions, com pot ser informant si té correu al sistema, creant el seu directori de connexió o mostrant l'últim inici de sessió.

Aquests fitxers ja estan configurats per permetre l'ús dl'LDAP per a l'autenticació dels usuaris i inclouen el mòdul `pam_ldap.so`. Aquest mòdul és el que proporciona autenticació, autorització i canvi de contrasenya fent servir un servidor LDAP.

Caldrà, però, afegir una línia al final del fitxer `/etc/pam.d/common-session`:

```
session required pam_mkhomedir.so
```

Aquesta línia té una gran importància i forçarà la creació del directori de treball de l'usuari que inicia sessió en el cas que aquest directori no existeixi.

El directori de treball (en anglès, *home directory*) té molta importància, ja que és el directori de connexió de l'usuari en iniciar una sessió interactiva. Sense aquest directori, l'usuari no pot treballar normalment i alguns sistemes fins i tot li impediran iniciar la sessió.

Com que fem servir una base de dades centralitzada d'usuaris, els usuaris es donen d'alta únicament al servidor LDAP. Cada usuari, per poder iniciar sessió en un ordinador client, hauria de tenir creat un directori de treball. Això implicaria crear aquest directori en totes les màquines client. Evidentment no és la solució desitjable, ja que possiblement aquest usuari no iniciarà sessió en tots els ordinadors de l'organització.

El que fa la solució proposada és una crida al mòdul `pam_mkhomedir.so` de la PAM, que comprovarà si existeix o no el directori definit en l'atribut *homeDirectory* (vegeu la figura 3.1 per recordar els diferents camps obligatoris) i si no existeix el crearà, fent servir el directori `/etc/skel/` com a base.

3.2.4 Verificació de la configuració

La comprovació bàsica del funcionament de la configuració es realitza amb l'ordre ***getent***. Aquesta ordre obté entrades d'una base de dades administrativa que s'especifica.

La base de dades pot ser passwd, group, hosts, services, protocols o networks. La que ens interessa és passwd, ja que ens proporcionarà les entrades d'aquesta base de dades.

Exemple d'ús de l'ordre getent

```
1 root@client:~# getent passwd
2 root:x:0:0:root:/root:/bin/bash
3 daemon:x:1:1:daemon:/usr/sbin:/bin/sh
4 bin:x:2:2:bin:/bin:/bin/sh
5 ...
6 mitabe:x:1000:1000:Miquel Tarazona Belenguer,,,:/home/mitabe:/bin
      /bash
7 sshd:x:109:65534::/var/run/sshd:/usr/sbin/nologin
8 mfreixas:x:10001:5001:Marc Freixas:/home/comercials/mfreixas:/bin
      /bash
9 jvila:*:10002:5002:Júlia Vila:/home/users/jvila:/bin/sh
```

Els dos usuaris ressaltats no són al fitxer /etc/passwd sinó que s'obtenen de la consulta al servidor LDAP.

El resultat de l'ordre *getent passwd* és el conjunt d'usuaris que es poden autenticar en el sistema. És la unió del fitxer /etc/passwd amb la consulta LDAP realitzada per l'NSS. Les dues últimes entrades són els usuaris creats en el servidor LDAP amb els atributs d'objecte posixAccount, que, com s'observa, són totalment compatibles amb els atributs que hi ha en el fitxer `/etc/passwd`.

Es pot obtenir el mateix resultat si es consulta la base de dades dels grups fent

servir el paràmetre *group*.

Exemple de l'ordre getent group

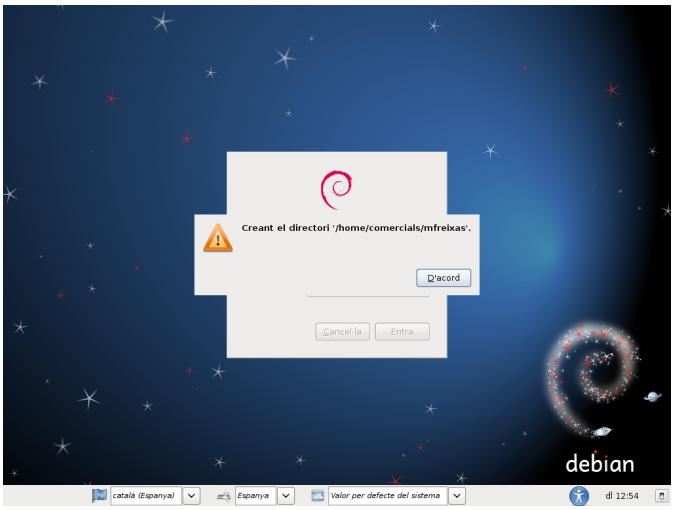
```
root@client:~# getent group
root:x:0:
daemon:x:1:
...
mitabe:x:1000:
comercials:*:5001:
administratius:*:5002:
```

A més dels grups locals apareixen els dos grups creats en el directori.

Per iniciar sessió en el client amb un usuari del directori, simplement s'ha d'introduir el nom d'usuari i la contrasenya. Per a l'usuari és totalment transparent, ja que no sap si la seva informació d'usuari està en el sistema local Linux o en un servidor remot amb LDAP.

Quan un usuari inicia sessió per primera vegada en una màquina configurada en el domini es mostra un missatge advertint de la creació del directori personal, com s'aprecia en la figura 3.2.

Figura 3.2. Creació d'un directori personal en un entorn gràfic



3.2.5 Reconfiguració de l'autenticació

Si les coses no han funcionat correctament, sempre es pot tornar a realitzar la configuració de les utilitats PAM i NSS per al LDAP fent servir l'ordre *dpkg-reconfigure*:

```
# dpkg-reconfigure libpam-ldap
# dpkg-reconfigure libnss-ldap
```

Les preguntes que ens fan són les mateixes que en la instal·lació inicial, i es prenen per defecte els valors que ja s'havien introduït.

3.3 Integració de Samba amb l'OpenLDAP

Samba és un conjunt de programes que implementa en sistemes basats en Unix (GNU/Linux) una sèrie de serveis i protocols que permeten compartir fitxers i impressores entre els equips d'una xarxa local.

La característica principal de Samba és que permet compartir recursos entre màquines Windows i GNU/Linux connectades en xarxa. Possibilita, per tant, compartir recursos en un escenari heterogeni.

Samba permet configurar un servidor GNU/Linux com a controlador primari de domini. D'aquesta manera s'autenticaran i compartiran recursos de màquines tant Windows com GNU/Linux. Si teniu un servidor Samba configurat a la xarxa, per facilitar la gestió del domini podeu centralitzar la gestió dels usuaris i les màquines, emmagatzemant aquesta informació al directori LDAP en comptes de fer servir una base de dades local.

Per realitzar aquesta configuració s'ha de configurar Samba com a client LDAP. Per aconseguir-ho el primer que caldrà fer és afegir l'esquema `samba.schema` al directori LDAP per poder tenir accés a tots els objectes i atributs necessaris. Una vegada el directori posseeix les característiques necessàries, es pot configurar Samba com a client LDAP.

3.3.1 Preparació del servidor

Vegeu l'apartat "Configuració de l'autenticació LDAP" d'aquesta unitat per a una explicació més detallada.

Com que l'autenticació de Samba amb l'LDAP està molt relacionada amb l'autenticació del mateix sistema servidor, cal configurar, en primer lloc, el servidor com a client LDAP, en aquest cas d'ell mateix.

En primer lloc cal instal·lar el programari necessari:

```
root@servidor:~# aptitude install libnss-ldap libpam-ldap ldap-utils
```

Quan el programa de configuració pregunta quin serà el servidor LDAP per fer l'autenticació, es pot contestar tant amb l'adreça IP del servidor, com amb "localhost" o "127.0.0.1".

Cal configurar el fitxer `/etc/nsswitch.conf` per habilitar l'autenticació LDAP al sistema.

```
root@servidor:~# cat /etc/nsswitch.conf
# /etc/nsswitch.conf
#
# Example configuration of GNU Name Service Switch functionality.
# If you have the 'glibc-doc-reference' and 'info' packages installed, try:
# 'info libc "Name Service Switch"' for information about this file.

passwd: compat ldap
group: compat ldap
```

```
shadow: compat ldap

hosts: files dns
networks: files

protocols: db files
services: db files
ethers: db files
rpc: db files

netgroup: ldap
```

En el fitxer `/etc/pam.d/common-password` s'ha de modificar una línia canviant:

```
password [success=1 user_unknown=ignore default=die] pam_ldap.so use_authtok
    try_first_pass
```

per:

```
password[success=1 user_unknown=ignore default=die]pam_ldap.so try_first_pass
```

En el fitxer `/etc/pam.d/common-session` s'ha d'afegir al final la línia per habilitar la creació automàtica de directoris:

```
session optional pam_mkhomedir.so skel=/etc/skel umask=077
```

En aquest cas, es defineixen, a més de l'ús de la llibreria, dues opcions: el directori a partir del qual es crearà el directori de treball i la màscara de creació del directori.

3.3.2 Afegir samba.schema al directori LDAP

El fitxer LDIF necessari per al suport de Samba està disponible a la secció "Annexos" del material web del mòdul.

La configuració de l'OpenLDAP amb el mètode cn=config permet realitzar canvis dinàmics en la configuració del directori. Així es pot afegir el suport Samba sense necessitat de reiniciar el servei; simplement s'ha d'afegir un fitxer LDIF amb la definició de les estructures desitjades. Aquest fitxer LDIF no està disponible als repositoris de Debian i per crear-lo hi ha dues opcions:

1. Navegar per la xarxa i trobar-lo en algun lloc on ja estigui creat i a la nostra disposició.
2. Crear-lo a partir dels antics fitxers de definicions d'esquema `samba.schema`.

Per crear el fitxer `samba.ldif`, en primer lloc cal obtenir el fitxer `samba.schema` per agafar-lo com a base. Aquest fitxer es troba al paquet `samba-doc`:

```
root@servidor:~# apt-get install samba-doc
```

Se situa el fitxer en el directori corresponent:

```
root@servidor:~# cp /usr/share/doc/samba-doc/examples/LDAP/samba.schema.gz /etc
    /ldap/schema/
```

Es descomprimeix el fitxer `samba.schema.gz`:

```
root@servidor:~# gzip -d /etc/ldap/schema/samba.schema.gz
```

Cal crear un arxiu de configuració temporal on es defineixen els diferents fitxers amb format schema que es faran servir per generar un directori de configuració des d'on es podran obtenir els fitxers LDIF necessaris. Aquest fitxer pot tenir qualsevol nom, però és molt important l'ordre de les línies *include*, ja que en l'LDAP hi ha herència de classes i d'atributs i les definicions s'han de fer en l'ordre correcte. En cas contrari es pot produir algun error.

```
root@servidor:~# cat schema_convert.conf
include /etc/ldap/schema/core.schema
include /etc/ldap/schema/collective.schema
include /etc/ldap/schema/corba.schema
include /etc/ldap/schema/cosine.schema
include /etc/ldap/schema/duaconf.schema
include /etc/ldap/schema/dyngroup.schema
include /etc/ldap/schema/inetorgperson.schema
include /etc/ldap/schema/java.schema
include /etc/ldap/schema/misc.schema
include /etc/ldap/schema/nis.schema
include /etc/ldap/schema/openldap.schema
include /etc/ldap/schema/ppolicy.schema
include /etc/ldap/schema/samba.schema
```

Amb el fitxer de configuració personalitzat es fa servir l'ordre *slapcat* per generar un directori temporal de configuració de l'estil cn=config, des del qual es poden obtenir els fitxers LDIF necessaris.

```
root@servidor:~# mkdir -p ./tmp/ldif_output
root@servidor:~# slapcat -f schema_convert.conf -F ./tmp/ldif_output -n0 -s "cn
    ={12}samba,cn=schema,cn=config" > ./tmp/cn=samba.ldif
```

Ara en el directori temporal hi ha un nou fitxer que cal editar per eliminar l'índex de l'entrada i informació extra que no és necessària:

```
root@servidor:~# vim ./tmp/cn\=samba.ldif
```

En aquest fitxer cal modificar dues línies, la 1 i la 3:

```
dn: cn={12}samba,cn=schema,cn=config
...
cn: {12}samba
```

Que passen a ser:

```
dn: cn=samba,cn=schema,cn=config
...
cn: samba
```

El que s'ha fet és eliminar els índexs del fitxer temporal, quan es tornen a afegir al directori tornaran a tenir índex, però possiblement serà diferent (i correcte, que és l'important).

Per acabar, s'eliminen les línies de la 186 al final, que es mostren a continuació:

```
structuralObjectClass: olcSchemaConfig
entryUUID: 01a09774-3073-1031-9c6c-798a169c02ec
creatorsName: cn=config
createTimestamp: 20120512114024Z
entryCSN: 20120512114024.039233Z#000000#000#000000
modifiersName: cn=config
modifyTimestamp: 20120512114024Z
```

Cal afegir el fitxer `cn=samba.ldif` al directori una vegada editat.

```
root@servidor:~# ldapadd -Y EXTERNAL -H ldapi:/// -f ./tmp/cn=samba.ldif
```

Amb el nou esquema afegit al directori, és convenient configurar l'slapd perquè l'indexi a partir dels nous atributs definits en l'esquema de Samba. D'aquesta manera es millorarà el rendiment de bases de dades grans. Aquesta configuració es fa mitjançant un fitxer LDIF en el qual es defineixen els nous índexs:

```
root@servidor:~# cat samba_indexes.ldif
dn: olcDatabase={1}hdb,cn=config
changetype: modify
add: olcDbIndex
olcDbIndex: uidNumber eq
olcDbIndex: gidNumber eq
olcDbIndex: loginShell eq
olcDbIndex: uid eq,pres,sub
olcDbIndex: memberUid eq,pres,sub
olcDbIndex: uniqueMember eq,pres
olcDbIndex: sambaSID eq
olcDbIndex: sambaPrimaryGroupSID eq
olcDbIndex: sambaGroupType eq
olcDbIndex: sambaSIDList eq
olcDbIndex: sambaDomainName eq
olcDbIndex: default sub
```

Aquest fitxer configura la base de dades amb els nous índexs. Simplement cal afegir-lo a la configuració de l'slapd:

```
ldapmodify -Y EXTERNAL -H ldapi:/// -f samba_indexes.ldif
```

No cal reiniciar el dimoni `slapd`; la configuració ja està aplicada i en funcionament.

A partir d'aquest moment, l'OpenLDAP està preparat per tenir un client Samba.

3.3.3 Configurar Samba com a client LDAP

Per configurar Samba com client LDAP hi ha un paquet anomenat *smbldap-tools*, que disposa dels fitxers necessaris per facilitar la configuració que s'ha d'instal·lar.

```
root@servidor:~# aptitude install samba smbldap-tools
```

Aquest paquet conté un fitxer amb un exemple de configuració per a Samba: "/usr/share/doc/smbldap-tools/examples/smb.conf". Aquest fitxer es pot fer servir

com a punt de partida de la configuració de Samba si no està correctament configurat. Si ja ho està, seria convenient agafar-lo com a referència per modificar la configuració de Samba.

El fitxer `/etc/samba/smb.conf` permet definir un directori LDAP com a processador dorsal (*backend*) per emmagatzemar els usuaris Samba. Serà en aquest moment quan Samba es convertirà en client LDAP.

Les línies que cal configurar (prenent com a referència el fitxer d'exemple) per definir aquest processador dorsal són les següents:

```
ldap admin dn = cn=admin,dc=empresa,dc=com
ldap group suffix = ou=Grups
ldap idmap suffix = ou=Idmap
ldap machine suffix = ou=Computers
ldap suffix = dc=empresa,dc=com
ldap ssl = no
ldap user suffix = ou=Personal
```

Consulteu la secció "Annexos" del material web del mòdul per obtenir el llistat complet del fitxer de configuració /etc/samba/smb.conf.

Una vegada configurat correctament Samba, cal reiniciar el servei per carregar el fitxer de configuració i afegir la contrasenya Samba a l'administrador del directori:

```
root@servidor:~# service samba restart
Stopping Samba daemons: nmbd smbd.
Starting Samba daemons: nmbd smbd.
root@servidor:~# smbpasswd -W
Setting stored password for "cn=admin,dc=empresa,dc=com" in secrets.tdb
New SMB password:
Retype new SMB password:
```

Per a la correcta integració entre Samba i l'LDAP, les utilitats *smbldap-tools* han d'estar configurades correctament. Aquesta configuració es realitza mitjançant un script que proporcionen els mantenidors del paquet.

L'script llegeix la configuració de Samba i adapta la configuració de les utilitats.

```
root@servidor:~# gzip -d /usr/share/doc/smbldap-tools/configure.pl.gz
root@servidor:~# perl /usr/share/doc/smbldap-tools/configure.pl
```

La configuració ofereix una sèrie d'opcions en la majoria de les quals cal deixar l'opció per defecte, ja que aquestes opcions s'obtenen a partir de les configuracions actuals del sistema. En qualsevol cas, cada opció està perfectament explicada.

Podem veure un exemple de configuració de les utilitats smbldap-tools en el codi que apareix a continuació:

```
root@servidor:~# perl /usr/share/doc/smbldap-tools/configure.pl
$# is no longer supported at /usr/share/doc/smbldap-tools/configure.pl line
    314.
==========================================================
smbldap-tools script configuration
==========================================================
Before starting, check
. if your samba controller is up and running.
. if the domain SID is defined (you can get it with the 'net getlocalsid')

. you can leave the configuration using the Crtl-c key combination
. empty value can be set with the "." character
==========================================================
```

```
Looking for configuration files...

Samba Configuration File Path [/etc/samba/smb.conf] >

The default directory in which the smbldap configuration files are stored is
    shown.
If you need to change this, enter the full directory path, then press enter to
    continue.
Smbldap-tools Configuration Directory Path [/etc/smbldap-tools/] >
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Let's start configuring the smbldap-tools scripts ...

. workgroup name: name of the domain Samba act as a PDC
workgroup name [EMPRESA] >
. netbios name: netbios name of the samba controler
netbios name [SERVIDOR] >
. logon drive: local path to which the home directory will be connected (for NT
     Workstations). Ex: 'H:'
logon drive [H:] >
. logon home: home directory location (for Win95/98 or NT Workstation).
(use %U as username) Ex:'\\SERVIDOR\%U'
logon home (press the "." character if you don't want homeDirectory) [\\
    SERVIDOR\%U] > .
. logon path: directory where roaming profiles are stored. Ex:'\\SERVIDOR\
    profiles\%U'
logon path (press the "." character if you don't want roaming profile) [\\
    SERVIDOR\profiles\%U] > .
. home directory prefix (use %U as username) [/home/%U] >
. default users' homeDirectory mode [700] >
. default user netlogon script (use %U as username) [logon.bat] >
default password validation time (time in days) [45] >
. ldap suffix [dc=empresa,dc=com] >
. ldap group suffix [ou=Grups] >
. ldap user suffix [ou=Personal] >
. ldap machine suffix [ou=Computers] >
. Idmap suffix [ou=Idmap] >
. sambaUnixIdPooldn: object where you want to store the next uidNumber
and gidNumber available for new users and groups
sambaUnixIdPooldn object (relative to ${suffix}) [sambaDomainName=EMPRESA] >
. ldap master server: IP adress or DNS name of the master (writable) ldap
    server
ldap master server [192.168.1.208] >
. ldap master port [389] >
. ldap master bind dn [cn=admin,dc=empresa,dc=com] >
. ldap master bind password [] >
. ldap slave server: IP adress or DNS name of the slave ldap server: can also
    be the master one
ldap slave server [192.168.1.208] >
. ldap slave port [389] >
. ldap slave bind dn [cn=admin,dc=empresa,dc=com] >
. ldap slave bind password [] >
. ldap tls support (1/0) [0] >
. SID for domain EMPRESA: SID of the domain (can be obtained with 'net
    getlocalsid SERVIDOR')
SID for domain EMPRESA [S-1-5-21-972867998-1773855070-1288752139] >
. unix password encryption: encryption used for unix passwords
unix password encryption (CRYPT, MD5, SMD5, SSHA, SHA) [SSHA] > MD5
. default user gidNumber [513] >
. default computer gidNumber [515] >
. default login shell [/bin/bash] >
. default skeleton directory [/etc/skel] >
. default domain name to append to mail adress [] >
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Use of uninitialized value $# in concatenation (.) or string at /usr/share/doc/
    smbldap-tools/configure.pl line 314, <STDIN> line 33.
backup old configuration files:
/etc/smbldap-tools/smbldap.conf->/etc/smbldap-tools/smbldap.conf.old
/etc/smbldap-tools/smbldap_bind.conf->/etc/smbldap-tools/smbldap_bind.conf.old
writing new configuration file:
/etc/smbldap-tools/smbldap.conf done.
```

```
/etc/smbldap-tools/smbldap_bind.conf done.
```

Es defineixen tots els paràmetres. Com que Samba ja estava correctament configurat, en la majoria dels casos caldrà escollir les opcions per defecte. L'únic que s'ha fet en l'exemple és desactivar els perfils mòbils.

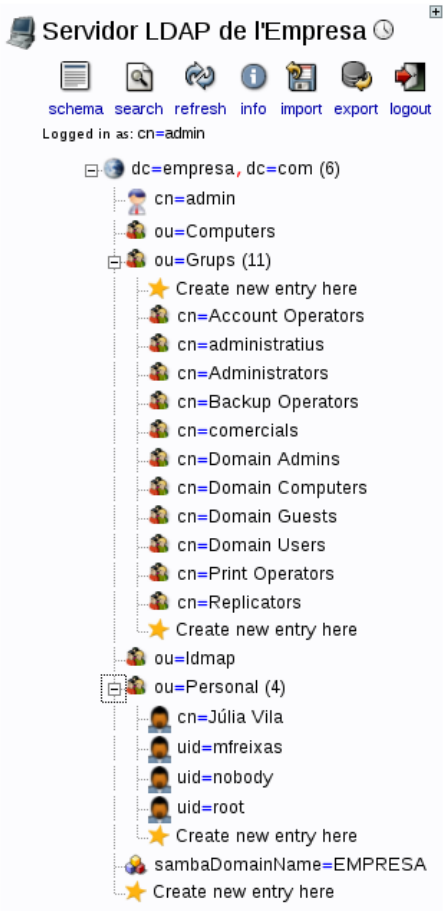
Amb les utilitats correctament configurades, el pas següent és crear en el directori l'estructura necessària d'entrades (unitats organitzatives i grups) per al correcte funcionament de Samba. Aquesta creació es fa amb l'ordre *smbldap-populate*:

```
root@servidor:~# smbldap-populate
```

Al final de l'execució d'aquesta ordre se'ns demana la contrasenya de l'usuari *root* per al sistema Unix i per a Samba.

En la figura 3.3 s'observen els objectes que s'han creat en el directori per permetre que Samba realitzi les autenticacions amb la base de dades LDAP.

FIGURA 3.3. DIT resultant de realitzar smblap-populate



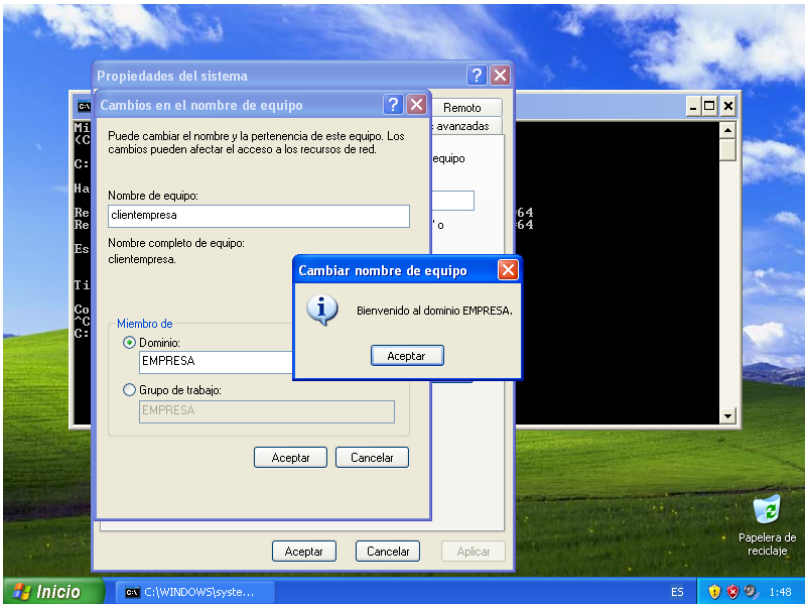
Abans d'afegir les màquines Windows al domini, cal crear els directoris que s'han definit en el fitxer de configuració de Samba.

```
root@servidor:/etc/samba# mkdir /srv/samba
root@servidor:/etc/samba# mkdir /srv/samba/printers
root@servidor:/etc/samba# mkdir /srv/samba/spool
root@servidor:/etc/samba# mkdir /srv/samba/profiles
```

```
root@servidor:/etc/samba# mkdir /srv/samba/netlogon
```

A partir d'aquest moment es pot afegir una màquina Windows al domini, com mostra la figura 3.4.

Figura 3.4. Afegir una màquina Windows a un domini



3.3.4 Comprovació de la configuració

El principal motiu de fer servir Samba és permetre la integració entre totes les màquines de la xarxa que estem administrant, ja que aquest programari facilita la integració de sistemes GNU/Linux, Windows i OS X. És habitual combinar un servidor Samba i una base de dades LDAP per permetre la gestió centralitzada d'usuaris i equips, simulant el comportament d'un domini creat amb Active Directory de Microsoft.

Es pot afegir una màquina Windows al domini que s'ha configurat de manera totalment transparent, de la mateixa manera que s'afegiria a un domini d'Active Directory. A més, tots els usuaris de l'LDAP podran iniciar sessió a la màquina Windows.

3.4 Autenticació a Pure-FTPd amb l'OpenLDAP

Pure-FTPd és un servidor FTP lliure amb llicència BSD que ha estat dissenyat pensant en la seguretat. Ha estat portat a un gran nombre de sistemes operatius com Linux, OpenBSD, NetBSD, FreeBSD, DragonFly BSD, Solaris, Tru64, Darwin, IRIX, HP-UX i fins i tot Android.

Els repositoris de Debian contenen un paquet compilat i preparat per fer servir en

el sistema.

El dimoni `Pure-FTPd` permet l'autenticació dels usuaris mitjançant LDAP. Per fer aquesta autenticació només caldrà instal·lar el paquet `pure-ftpd-ldap`:

```
root@servidor:/etc/moodle# aptitude install pure-ftpd-ldap
```

Aquest paquet és el dimoni `Pure-FTPd` compilat amb suport LDAP. Els fitxers de configuració del dimoni estan a `/etc/ftpd/`.

No es tractarà en aquest apartat la configuració del comportament del dimoni `Pure-FTPd` més enllà de l'autenticació per al LDAP.

El fitxer on es configura l'autenticació LDAP és `/etc/pure-ftpd/db/ldap.conf`.

A continuació podem veure un exemple de configuració LDAP per al dimoni `pure-ftpd`:

```
root@servidor:/etc/pure-ftpd/db# cat ldap.conf
###########################################
# #
# Sample Pure-FTPd LDAP configuration file. #
# See README.LDAP for explanations. #
# #
###########################################

# Optional : name of the LDAP server. Default : localhost
LDAPServer servidor.empresa.com

# Optional : server port. Default : 389
LDAPPort 389

# Mandatory : the base DN to search accounts from. No default.
LDAPBaseDN dc=empresa,dc=com

# Optional : who we should bind the server as.
# Default : binds anonymously or binds as FTP users

LDAPBindDN cn=admin,dc=empresa,dc=com


# Password if we don't bind anonymously
# This configuration file should be only readable by root
LDAPBindPW contrasenya

# Optional : default UID, when there's no entry in a user object
# LDAPDefaultUID 500

# Optional : default GID, when there's no entry in a user object
# LDAPDefaultGID 100

# Filter to use to find the object that contains user info
# \L is replaced by the login the user is trying to log in as
# The default filter is (&(objectClass=posixAccount)(uid=\L))
# LDAPFilter (&(objectClass=posixAccount)(uid=\L))

# Attribute to get the home directory
# Default is homeDirectory (the standard attribute from posixAccount)
# LDAPHomeDir homeDirectory

# LDAP protocol version to use
# Version 3 (default) is mandatory with recent releases of OpenLDAP.
# LDAPVersion 3

```

```
47 # Optional: use TLS to connect to the LDAP server
48 # LDAPUseTLS True
49
50 # Can be PASSWORD or BIND.
51 # PASSWORD retrieves objects and checks against the userPassword attribute
52 # BIND tries to bind
53 LDAPAuthMethod PASSWORD
```

La majoria d'opcions permeten definir quin atribut de l'objecte servirà per definir les opcions específiques de la connexió. L'explicació de les opcions és la següent:

1. **LDAPServer servidor.empresa.com** és la URI del servidor o l'adreça IP.
2. **LDAPPort 389** és el port on se suposa que està connectat el servidor LDAP.
3. **LDAPBaseDN dc=empresa,dc=com** és la base a partir de la qual se cercaran els objectes. Es pot definir un subarbre, si es vol. Per exemple, es pot definir una unitat organitzativa especial per als usuaris que tindran accés al servei FTP.
4. **LDAPBindDN cn=admin,dc=empresa,dc=com** és l'usuari amb el qual el servidor fa la consulta al directori. També es pot definir una consulta anònima si el directori ho permet.
5. **LDAPBindPW contrasenya** és la contrasenya de l'usuari amb el qual es fa la consulta.
6. **LDAPDefaultUID 500**. En el cas que l'usuari que s'ha validat no tingués definit l'atribut uidNumber es faria servir aquest número.
7. **LDAPDefaultGID 100**. En el cas que l'usuari que s'ha validat no tingués definit l'atribut gidNumber es faria servir aquest número.
8. **LDAPFilter (&(objectClass=posixAccount)(uid=\L))**. Quan es fa una consulta al directori sempre es fa servir un filtre. Aquesta opció permet definir quin serà.
9. **LDAPHomeDir homeDirectory** és l'atribut on està definit el directori de connexió de l'usuari que s'ha autenticat.
10. **LDAPVersion 3** és la versió del protocol. Habitualment és la 3.
11. **LDAPUseTLS True** defineix si es fa servir autenticació segura.
12. **LDAPAuthMethod PASSWORD** permet canviar el mètode d'autenticació, que pot ser comprovant la contrasenya o fent una connexió al directori (BIND).

3.4.1 Comprovació de la connexió FTP

Una simple connexió al servidor FTP permet comprovar el correcte funcionament de l'autenticació mitjançant LDAP.

Exemple de connexió FTP

```
usuari@hp:~$ ftp 192.168.1.208
Connected to 192.168.1.208.
220———————— Welcome to Pure-FTPd [privsep] [TLS] ————————
220-You are user number 1 of 50 allowed.
220-Local time is now 13:53. Server port: 21.
220-This is a private system - No anonymous login
220-IPv6 connections are also welcome on this server.
220 You will be disconnected after 15 minutes of inactivity.
Name (192.168.1.208:usuari): mfreixas
331 User mfreixas OK. Password required
Password:
230-User mfreixas has group access to: comercials
230 OK. Current directory is /home/comercials/mfreixas
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

A l'exemple s'observa la connexió al servidor FTP amb un usuari anomenat mfreixas que està donat d'alta al directori.

3.5 Autenticació a Joomla amb l'OpenLDAP

Més enllà de l'autenticació d'usuaris que inicien sessió en una màquina determinada i fan servir un sistema operatiu, actualment existeixen multitud d'aplicacions web que requereixen l'autenticació de l'usuari. En són exemples Moodle, Joomla, MediaWiki i Zimbra.

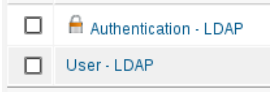
Unes d'aquestes aplicacions són els anomenats *sistemes de gestió de continguts* o CMS (Content Management System), d'entre els quals un dels més utilitzats és Joomla.

3.5.1 Configuració de Joomla

Joomla fa servir per defecte MySQL per gestionar tot el contingut, incloent la base de dades dels usuaris i les seves contrasenyes. Aquest mètode d'autenticació es pot canviar triant l'autenticació LDAP com a opció preferent. De fet, el disseny de Joomla permet fer servir més d'un mètode.

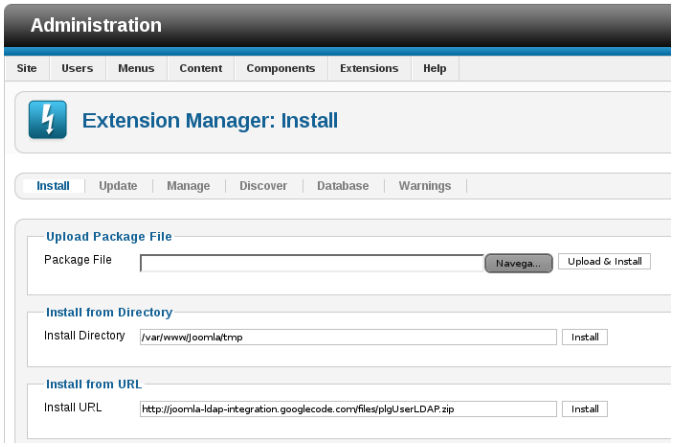
Per fer la configuració cal entrar en la configuració de Joomla, en l'apartat de gestió d'extensions, on s'han d'habilitar els dos connectors (*plug-in*) que es mostren a la figura 3.5.

FIGURA 3.5. Connectors LDAP de Joomla



Authentication - LDAP està inclòs per defecte en la instal·lació de Joomla, però si no es disposa de *User - LDAP* es pot instal·lar fàcilment com es mostra en la figura 3.6. La instal·lació es fa amb la utilitat *extension manager*, introduint la URL directament.

FIGURA 3.6. Instal·lació de User - LDAP a Joomla



Una vegada habilitats, *User - LDAP* no requereix cap configuració, simplement l'habilitació. La configuració d'*Authentication - LDAP* sí que conté els paràmetres necessaris per activar l'autenticació.

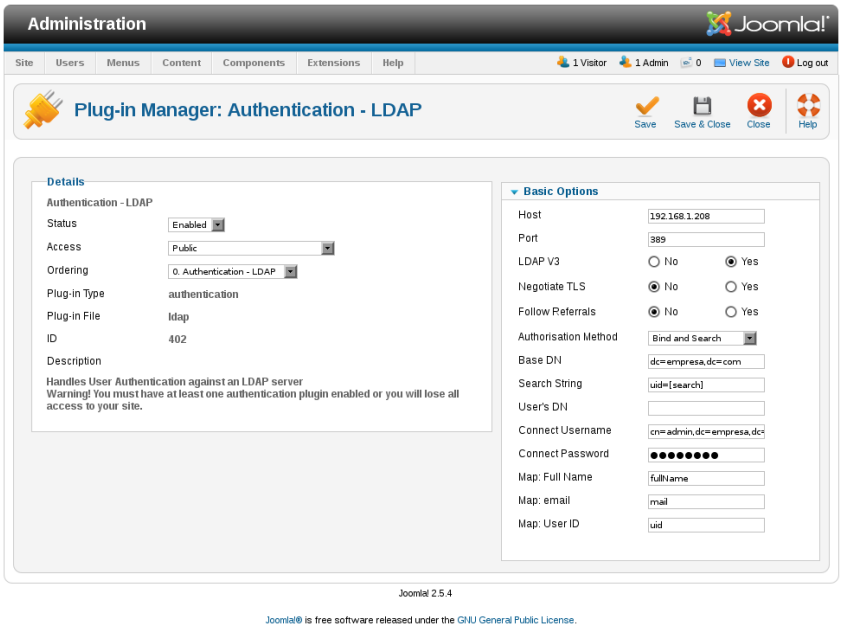
Exemple de configuració del connector Authentication - LDAP

L'exemple connecta Joomla al servei LDAP que s'ha fet servir al llarg de la unitat.

En la figura 3.7 es poden observar els paràmetres més importants de connexió de Joomla per al LDAP. Són:

1. ***Host***: l'adreça del servidor LDAP.
2. ***Port***: el port de connexió per defecte.
3. ***LDAP V3***: informa de la versió del protocol.
4. ***Negotiate TLS***: especifica si es farà servir una connexió segura o no.
5. ***Authorisation Method***: permet definir si es farà una consulta anònima al directori o si, per contra, caldrà connectar-se amb un usuari per fer la consulta.
6. ***Connect Username*** i ***Connect Password***: l'objecte del directori i la contrasenya que es faran servir per realitzar la consulta.
7. ***Map***: permet definir els atributs LDAP que es faran coincidir amb els atributs Joomla.

FIGURA 3.7. Configuració d'Authentication - LDAP



Amb una correcta configuració es pot accedir a Joomla amb l'usuari donat d'alta al LDAP, tal com s'aprecia en la figura 3.8 i figura 3.9.

FIGURA 3.8. Formulari d'inici de sessió de Joomla

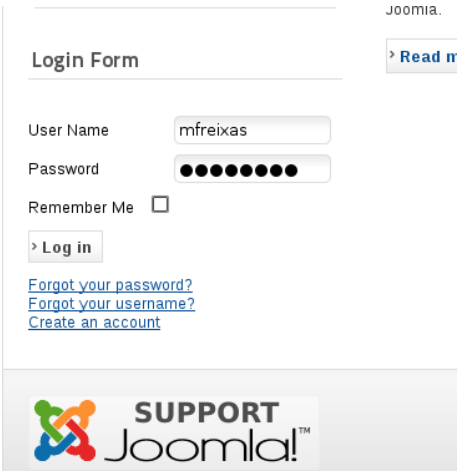


FIGURA 3.9. Sessió iniciada a Joomla

Joomla permet anar una mica més enllà en la integració amb l'LDAP, ja que té

definit el seu propi esquema. Cal afegir aquest esquema al directori si volem permetre la total integració de Joomla i l'LDAP i no només l'acreditació dels usuaris.

Per convertir el fitxer esquema al format LDIF cal crear un fitxer de configuració temporal anomenat *schema_convert.conf* amb el contingut següent:

```
root@servidor:~# cat schema_convert.conf
include /etc/ldap/schema/core.schema
include /etc/ldap/schema/cosine.schema
include /etc/ldap/schema/inetorgperson.schema
include /etc/ldap/schema/nis.schema
include /etc/ldap/schema/joomla.schema
```

A partir d'aquest fitxer de configuració es pot generar, amb l'ordre *slapcat*, el fitxer LDIF amb la informació de l'esquema.

```
root@servidor:~# slapcat -f schema_convert.conf -F ./tmp/ldif_output/ -n0 -s "
    cn={4}joomla,cn=schema,cn=config" > ./tmp/cn=joomla.ldif
```

Abans d'afegir-lo al directori, cal modificar-lo per eliminar els índexs, ja que el dimoni `slapd` serà l'encarregat d'afegir aquests índexs. Les primeres línies queden així:

```
dn: cn=joomla,cn=schema,cn=config
...
cn: joomla
```

També cal esborrar les últimes línies, ja que la informació que contenen no és necessària i produiria un error si intentéssim afegir el fitxer al directori. Cal eliminar les línies finals:

```
structuralObjectClass: olcSchemaConfig
...
modifyTimestamp: 20120513173054Z
```

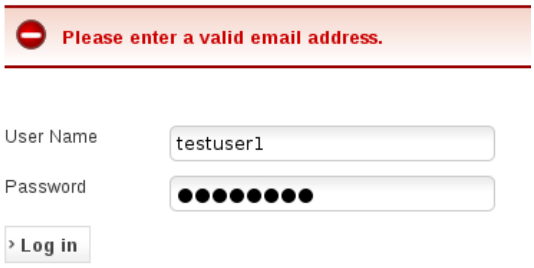
L'últim pas és afegir el fitxer al directori. Es fa amb l'ordre *ldapadd*:

```
root@servidor:~# ldapadd -Y EXTERNAL -H ldapi:/// -f ./tmp/cn\=joomla.ldif
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
adding new entry "cn=joomla,cn=schema,cn=config"
```

Amb l'extensió de l'esquema, Joomla ja pot treballar amb els objectes de l'LDAP com si fossin usuaris i grups tradicionals.

Quan es dona d'alta un usuari al LDAP que es vol fer servir amb Joomla, és molt important que aquest usuari tingui al menys una adreça de correu electrònic definida. L'atribut és *mail*. Si no és així, tot i estar correctament configurada la integració, l'usuari no podrà iniciar sessió, com s'aprecia en la figura 3.10.

FIGURA 3.10. Usuari sense correu electrònic a Joomla



Llenguatges de guions i automatització de tasques

Mª Del Mar Sánchez-Colomer Ruiz

Administració de sistemes operatius

Índex

Introducció

L'intèrpret d'ordres o *shell* és un programari que proporciona una interfície als usuaris en un sistema operatiu i els proveeix accés als serveis del nucli. La majoria d'intèrprets d'ordres incorporen un llenguatge de programació propi que permet crear programes que anomenem *guions de shell* o *shellscripts*. En l'administració de sistemes operatius, entendre i saber fer guions de *shell* és fonamental perquè és la manera principal d'automatitzar tasques de manteniment i configuració del sistema. Algunes d'aquestes tasques, com ara les còpies de seguretat, s'han de poder executar de manera periòdica i sense que hi intervingui l'usuari. Mitjançant la programació de *shellscripts* i un planificador de tasques aconseguim aquest propòsit.

Al llarg de la unitat "Llenguatges de guions i automatització de tasques" tractarem els continguts relacionats amb la matèria de manera genèrica per als sistemes operatius de servidor més representatius en l'actualitat i aprofundirem en el sistema operatiu Unix. En els sistemes de tipus Unix gairebé tots els fitxers de dades de configuració del sistema són fitxers de text i una bona part del codi d'inicialització del sistema pren la forma de guions de *shell*. Això fa d'Unix un sistema operatiu idoni per aprendre els conceptes i els procediments implicats en l'administració de sistemes en general i, en particular, dels que ens ocupen en aquesta unitat.

En l'apartat "Llenguatges de guions de *shell*" veurem alguns conceptes previs sobre llenguatges de programació i la diferència entre els llenguatges compilats i els llenguatges interpretats. A continuació introduirem què són els llenguatges de guions, els *shells*, els guions de *shell*, quins tipus de *shell* hi ha, i ens centrarem en el Bash, l'intèrpret d'ordres predeterminat de gairebé totes les distribucions GNU/Linux, així com del sistema operatiu Mac OS X. No farem un estudi de totes les característiques del *shell* Bash atès que queda fora de l'abast d'aquesta unitat, però veurem aquelles funcionalitats que és necessari conèixer per poder abordar correctament la programació de *shellscripts* i l'automatització de tasques del sistema.

En l'apartat "Programació del *shell* Bash" introduirem el llenguatge de programació de Bash i estudiarem els aspectes essencials per crear i executar guions de *shell*, interaccionar amb l'usuari, tractar paràmetres, especificar i avaluar condicions, utilitzar les estructures de control i fer funcions.

En l'apartat "Planificació i automatització de tasques" veurem què són els sistemes de planificació de tasques i estudiarem la configuració i el funcionament del planificador de tasques dels sistemes Unix i derivats, anomenat *cron*. Arribats a aquest punt estarem en disposició d'automatitzar i planificar tasques del sistema, de manera que acabarem analitzant alguns casos pràctics d'automatització amb les eines que hem estudiat. En concret, veurem exemples de procediments de còpies

de seguretat, manteniment de fitxers de registre, gestió massiva d'usuaris i inici automàtic de serveis.

Per treballar els continguts d'aquesta unitat, és necessari haver assimilat una sèrie de continguts bàsics relacionats amb el sistema operatiu Unix: el funcionament de la línia d'ordres, l'organització del sistema de fitxers i coneixements d'administració bàsica del sistema, tals com la gestió d'usuaris i de grups. Igualment, és convenient haver assolit els coneixements de fonaments de programació estructurada i haver fet programes en algun llenguatge d'alt nivell.

Aquesta unitat és eminentment pràctica, per això a mesura que aneu avançant en la lectura dels diferents apartats és molt recomanable que aneu provant els exemples que hi apareixen, i que aneu fent els exercicis i les activitats d'aprenentatge proposades a la web del mòdul.

Resultats de l'aprenentatge

1. Utilitza llenguatges de guions en sistemes operatius, descrivint la seva aplicació i administrant serveis del sistema operatiu.
    - Utilitza i combina les estructures del llenguatge per crear guions.
    - Utilitza eines per depurar errors sintàctics i d'execució.
    - Interpreta guions de configuració del sistema operatiu.
    - Realitza canvis i adaptacions de guions del sistema.
    - Crea i prova guions d'administració de serveis.
    - Crea i prova guions d'automatització de tasques.
    - Implanta guions en sistemes lliures i propietaris.
    - Consulta i utilitza llibreries de funcions.
    - Documenta els guions creats.
2. Gestiona l'automatització de tasques del sistema, aplicant criteris d'eficiència i utilitzant ordres i eines gràfiques.
    - Descriu els avantatges de l'automatització de les tasques repetitives en el sistema.
    - Utilitza les ordres del sistema per a la planificació de tasques.
    - Estableix restriccions de seguretat.
    - Realitza planificacions de tasques repetitives o puntuals relacionades amb l'administració del sistema.
    - Automatitza l'administració de comptes.
    - Instal·la i configura eines gràfiques per a la planificació de tasques.
    - Utilitza eines gràfiques per a la planificació de tasques.
    - Documenta els processos programats com a tasques automàtiques.

1. Llenguatges de guions de shell

En l'àmbit de la informàtica, un programa és una seqüència d'instruccions que un ordinador ha d'executar per dur a terme una tasca determinada. El llenguatge de programació determina la forma amb què el programador ha d'escriure les operacions que l'ordinador ha de realitzar. De llenguatges de programació n'existeixen centenars, i cada any en sorgeixen de nous o versions millorades dels existents, que n'amplien les característiques per adaptar-se a les necessitats tecnològiques de cada moment.

Els intèrprets d'ordres o *shells* són programes que permeten la interacció dels usuaris amb el sistema operatiu i, més enllà d'aquesta funció, també incorporen llenguatges de programació que permeten crear programes que anomenem *guions*. Els guions de *shell* són molt útils per fer tasques d'administració del sistema i altres treballs repetitius que no requereixen un llenguatge de programació més sofisticat.

1.1 Conceptes previs

Abans d'abordar l'estudi dels llenguatges de guions de *shell*, fem un repàs d'alguns conceptes previs.

1.1.1 Llenguatges de programació

Un llenguatge de programació és un llenguatge informàtic usat per controlar el comportament d'una màquina, normalment un ordinador. Cada llenguatge té una sèrie de regles sintàctiques i semàntiques estrictes que cal seguir per escriure un programa informàtic, i que en descriuen l'estructura i el significat respectivament. Aquestes regles permeten especificar les dades amb què treballarà el programa i les accions que realitzarà.

El **llenguatge de màquina** o codi màquina és l'únic llenguatge de programació que poden entendre directament els circuits microprogramables de l'ordinador, com ara el microprocessador. El codi màquina està format exclusivament per codi binari, és a dir, per zeros (0) i uns (1). Un programa escrit en codi màquina consisteix en un seguit d'instruccions i dades codificats en binari. El llenguatge de màquina és específic de cada màquina, malgrat que el conjunt d'instruccions disponibles pugui ser similar.

Codi binari

Els circuits interns de l'ordinador treballen amb dos nivells de tensió que de manera abstracta representem amb el 0 i l'1. Per això el llenguatge màquina només utilitza aquests dos símbols.

Els llenguatges de programació se solen classificar principalment en **llenguatges de nivell baix**, que són molt propers al codi màquina utilitzat internament per

un tipus d'ordinador determinat, i **llenguatges de nivell alt**, que són més propers al llenguatge humà (normalment l'idioma anglès) i més independents del tipus d'ordinador.

Un llenguatge de programació d'alt nivell és un llenguatge que, en comparació amb els llenguatges de nivell baix, ofereix un nivell d'abstracció més alt, és més fàcil d'utilitzar i més portable entre plataformes de maquinari. L'objectiu d'aquests llenguatges de programació és alliberar als programadors de tasques complexes i augmentar la productivitat i l'eficiència en la generació de codi. Per posar un exemple, els llenguatges de programació de nivell alt no s'encarreguen directament de la gestió dels mapes de memòria. En canvi, en un programa escrit en un llenguatge de nivell baix, les dades utilitzades són referenciades per la seva posició en memòria, és a dir, és responsabilitat del programador controlar el mapa de memòria i l'assignació de memòria a cada dada.

Avui dia existeix una gran quantitat de llenguatges de programació de nivell alt. Es tracta d'un grup molt heterogeni amb una gran diversitat de característiques i objectius. Alguns exemples de llenguatges de nivell alt, més o menys especialitzats en diferents tasques, són Java, PHP, C, C++, Python, Visual Basic .NET, C#, Ruby, Cobol, Perl, JavaScript, etc.

Llenguatges de programació

La diferència entre llenguatges de nivell baix i alt es fa evident comparant dos programes que escriuen "Hola" a la pantalla, el primer usant llenguatge assemblador per màquines x86 (nivell baix) i el segon utilitzant el llenguatge de programació Python (nivell alt).

Programa en llenguatge assemblador

```
.MODEL SMALL
IDEAL
STACK 100H

DATASEG
HW DB 'Hola!$'

CODESEG
MOV AX, @data
MOV DS, AX
MOV DX, OFFSET HW
MOV AH, 09H
INT 21H
MOV AX, 4C00H
INT 21H
END
```

Programa en llenguatge d'alt nivell Python

```
print "Hola!"
```

1.1.2 Codi font

El **codi font** d'un programa es refereix a la sèrie d'instruccions escrites en algun llenguatge de programació llegible per l'home que conformen el programa. El codi font no és directament executable per l'ordinador, sinó que ha de ser traduït a

llenguatge de màquina que sí podrà ser executat pel maquinari de l'ordinador. Per a aquesta traducció, depenent del tipus de llenguatge utilitzat, s'usen els anomenats compiladors i intèrprets.

Programari lliure i programari propietari

Un aspecte a tenir en compte quan es parla del codi font d'un programa informàtic és si la seva llicència permet que aquest codi font estigui disponible perquè qualsevol pugui estudiar-lo, modificar-lo o reutilitzar-lo lliurement. Quan es compleix aquest aspecte es parla de programari lliure, en contraposició al programari propietari (o privatiu), que, normalment, no va acompanyat del codi font ni de cap de les llibertats esmentades.

1.1.3 Llenguatges compilats i llenguatges interpretats

Els programes escrits en llenguatges de programació d'alt nivell no es poden executar directament a la màquina. És necessari executar un procés de traducció del llenguatge d'alt nivell a llenguatge màquina. Aquesta traducció pot ser una compilació (llenguatges compilats), una interpretació (llenguatges interpretats) o una combinació de les dues opcions anteriors (llenguatges híbrids).

Alguns exemples de llenguatges compilats són Fortran, C, C++, Ada, Cobol i Visual Basic.

La implementació d'un llenguatge de programació és la que proveeix una manera perquè s'executi un programa. Un **llenguatge compilat** es refereix a un llenguatge de programació que típicament s'implementa mitjançant un compilador.

Un **compilador** d'un determinat llenguatge de programació és un programa que llegeix un fitxer en codi font escrit en aquest llenguatge de programació i el converteix en una seqüència de codi màquina per a una plataforma determinada (Intel, Sparc, etc.). El codi traduït pot servir com a entrada d'un altre intèrpret o un altre compilador. Un compilador anomenat *de codi natiu* és el que tradueix el codi font d'un programa directament a codi màquina executable. Sovint, però, aquest procés se separa en més d'una fase, per exemple, en una generació de codi objecte i un enllaçat, tal com es veu a la figura 1.1.

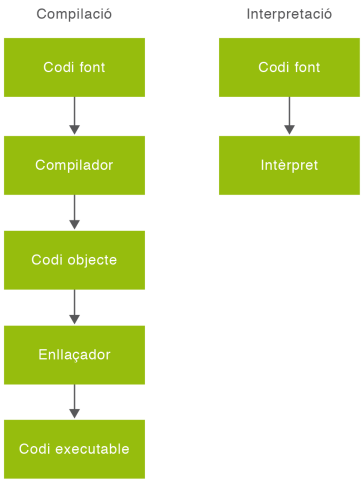
Alguns exemples de llenguatges interpretats són Perl, PHP, Python, Bash i JavaScript.

Un **llenguatge interpretat** és aquell en què les instruccions es tradueixen o interpreten una per una en temps d'execució a un llenguatge intermedi o llenguatge màquina.

Un **intèrpret** d'un llenguatge de programació és un programa que llegeix les ordres del codi font, en comprova la sintaxi i executa el codi relacionat amb aquestes ordres.

Els principals avantatges d'una implementació interpretada respecte d'una compilada són la seva independència respecte de la plataforma i que permet disminuir el cost de programació, en el sentit que permet desenvolupar i provar el programa més ràpidament. El desavantatge és que l'execució d'un programa interpretat normalment és més lenta que la d'un programa compilat i necessita més recursos (memòria, CPU, etc.).

FIGURA 1.1. Compilació i interpretació



1.1.4 Llenguatges de guions

Els **llenguatges de guions**, també coneguts amb el nom de **llenguatges d'*scripting***, són un tipus de llenguatges de programació d'alt nivell que poden ser de propòsit general o de propòsit específic i gairebé sempre són llenguatges interpretats. Els programes escrits amb aquests llenguatges s'anomenen **guions** o **scripts**.

Treball per lots

Es coneix com a treball per lots o batch job un tipus de programes que s'executen sense el control o supervisió directa de l'usuari.

Els llenguatges de guions tenen el seu origen en els llenguatges de control de tasques, concretament en el JCL (*job control language*), utilitzat als *mainframe*d'IBM per a la programació de treballs per lots.

El JCL és un llenguatge amb poques capacitats de programació i amb un propòsit molt específic. En canvi, molts dels llenguatges de guions actuals, com ara el Python o el Perl, són llenguatges de programació potents que permeten desenvolupar aplicacions de propòsit general.

Els llenguatges de guions sovint s'utilitzen per ampliar les prestacions que ofereix un altre llenguatge, entorn o aplicació. En aquest sentit són molt utilitzats en el desenvolupament d'aplicacions web:

1. **Scripts de navegadors web**. S'utilitzen per ampliar les capacitats de l'HTML i per inserir accions en pàgines web. Permeten crear efectes especials i aporten interactivitat. Els scripts són interpretats i executats en la màquina client pel navegador web, el qual ha d'incorporar l'intèrpret del llenguatge. Un exemple de llenguatge d'aquest tipus molt utilitzat és JavaScript.

2. **Scripts de servidor**. Són programes que permeten donar funcionalitats a les pàgines web que no es poden resoldre només amb els scripts de navegador. Els scripts de servidor permeten dotar de certa "intel·ligència" els llocs web, la qual cosa fa que generin pàgines diferents segons les circumstàncies. Un

dels llenguatges més utilitzats en aquest àmbit és PHP.

Llenguatge PHP

Alguns llenguatges de guions serveixen per a més d'un propòsit. Per exemple, encara que en el disseny del llenguatge PHP tot està orientat a facilitar la creació de llocs web, és possible crear aplicacions amb una interfície gràfica per a l'usuari, utilitzant l'extensió PHP-Qt o PHP-GTK. També pot ser usat des de la línia d'ordres amb el PHP-CLI (*command line interface*).

En l'àmbit dels sistemes operatius, els **intèrprets d'ordres** incorporen un tipus de llenguatges de guions que anomenem **llenguatges de guions de*shell***, que s'utilitzen amb diversos propòsits.

1.2 Intèrprets d'ordres o shells

Un intèrpret de línia d'ordres o simplement **intèrpret d'ordres**, és un programa informàtic que té la funció de llegir línies de text (ordres) escrites en un terminal o en un fitxer de text i interpretar-les.

Dispositius de xarxa

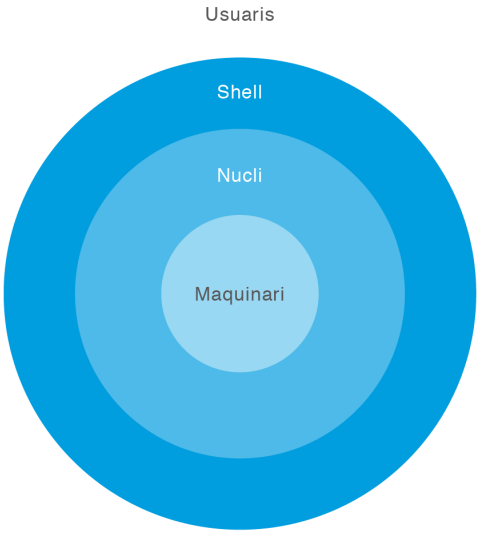
Els dispositius de xarxa que poden ser configurats per programari normalment disposen d'una interfície de línia d'ordres. Per exemple, el sistema operatiu IOS de Cisco disposa d'una interfície de línia d'ordres que és utilitzada per executar ordres de configuració, supervisió i manteniment dels dispositius de Cisco, ja sigui amb una consola de l'encaminador (router), amb un terminal o a partir de mètodes d'accés remot.

Les ordres s'escriuen seguint la sintaxi incorporada per aquest intèrpret. En llegir l'ordre, l'intèrpret analitza la seqüència de caràcters i, si la sintaxi de l'ordre és correcta, passa l'ordre interpretada al sistema operatiu o al programa que representa (una sessió d'FTP, una sessió d'SSH, etc.) perquè l'executi.

Els intèrprets d'ordres dels sistemes operatius també es coneixen amb el nom de ***shell***. El *shell* és un programa encarregat de llegir les ordres que tecleja l'usuari i convertir-les en instruccions que el sistema operatiu pot executar.

Antigament la interacció dels usuaris amb el sistema operatiu es feia únicament mitjançant interfícies de línia d'ordres, atès que no existien ordinadors amb la capacitat de mostrar gràfics o imatges. El terme *shell*va aparèixer a la dècada dels 70 amb el sistema operatiu Unix, que va ser pioner en el concepte d'un entorn de línia d'ordres potent. La traducció de *shell* seria "embolcall o closca", perquè envolta la resta de capes del sistema a mode de closca. Tal com es pot apreciar a la figura 1.2, el *shell* és la capa més externa i actua com a interfície entre l'usuari i la resta del sistema.

Figura 1.2. Capes del sistema operatiu Unix

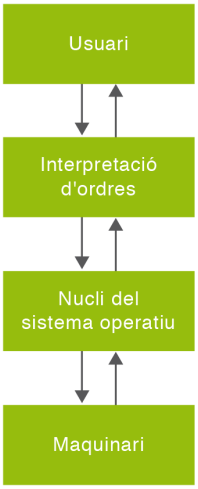


De vegades es confonen els termes d'intèrpret d'ordres i sistema operatiu i, de manera errònia, s'identifiquen l'un amb l'altre.

> L'intèrpret d'ordres és el que els usuaris veuen del sistema i està escrit de la mateixa manera que un programa d'usuari. No està integrat en el nucli del sistema operatiu, sinó que s'executa com un programa més de l'usuari.

El sistema operatiu sempre està situat en un nivell inferior als programes, i l'intèrpret d'ordres o *shell* és un programa més que té la tasca de llegir les ordres que li dóna l'usuari, realitzar una serie de funcions d'anàlisi i passar les ordres interpretades al nucli del sistema operatiu perquè les executi, com es veu a la figura 1.3.

Figura 1.3. Interpretació d'ordres



En l'actualitat gairebé tots els sistemes operatius disposen d'interfícies gràfiques d'usuari (GUI, *graphical user interface*) que permeten interactuar amb el sistema d'una manera molt més senzilla que amb la línia d'ordres, cosa que facilita les tasques de l'usuari. Amb tot, les interfícies de línies d'ordres continuen sent una eina de treball molt utilitzada en l'àmbit de l'administració de sistemes i xarxes, especialment per a l'automatització de tasques.

1.2.1 Llenguatges de guions de shell

La majoria dels intèrprets d'ordres de tots els sistemes operatius compten amb un llenguatge de programació propi per escriure programes que anomenem de diverses maneres: *programes per lots*, *arxius d'ordres*, ***guions de shell*** o ***shell scripts***.

Els **llenguatges de guions de *shell*** o de ***shell scripting*** són molt utilitzats en l'àmbit de l'administració de sistemes operatius, perquè permeten fer programes combinant crides a ordres del sistema, programes compilats, d'altres guions, etc. En principi són llenguatges pensats per a l'automatització de tasques en un sistema operatiu però realment la majoria permeten crear aplicacions de propòsit general.

El terme script

El terme anglès script es va agafar del guió escrit de les arts escèniques, el qual és interpretat per una sèrie d'actors i actrius (o, en el nostre cas, programes) seguint un ordre establert. En alguns textos es tradueix script com a guió i en d'altres s'utilitza l'expressió arxiu d'ordres.

Els intèrprets d'ordres interpreten guions escrits en el llenguatge que saben interpretar. Alguns intèrprets d'ordres també incorporen el motor intèrpret d'altres llenguatges a més del seu propi, la qual cosa permet l'execució d'scripts en aquests llenguatges directament al mateix intèrpret.

1.2.2 Automatització de tasques amb guions de shell

En general, no hi ha una manera única de fer les coses, però normalment algunes eines són més indicades per a determinats propòsits que d'altres.

Els llenguatges de guions de *shell* ofereixen un entorn de programació ràpid per a l'automatització de tasques de manteniment i configuració del sistema.

Les principals raons que porten l'administrador del sistema a implementar l'automatització de tasques mitjançant la creació de *shell scripts* són les següents:

- Execució de tasques repetitives. Els administradors de sistemes sovint repeteixen seqüències d'ordres (com una alta massiva d'usuaris) o bé alguna línia d'ordres llarga i complexa. Fer un script proporciona una manera ràpida i segura d'executar aquestes seqüències o línies d'ordres.

- Planificació de tasques. Hi ha moltes tasques que han de dur-se a terme amb una base regular (fer còpies de seguretat, netejar fitxers de registre o fitxers temporals, etc.). Mitjançant la realització de guions de *shell* i el planificador de tasques es poden programar aquestes tasques perquè es duguin a terme de manera automàtica i periòdica.

  – Delegar tasques. Hi ha tasques de manteniment del sistema que es poden o que s'han de delegar a d'altres usuaris (operadors, administradors menys experimentats, etc.). El fet d'utilitzar *shell scripts* per implementar aquestes tasques proporciona una manera perquè els usuaris menys experts puguin aprendre, ja que en poden analitzar el codi i fins i tot mantenir-lo a mesura que el seu aprenentatge i habilitats progressen.

  – Personalització de l'inici de serveis. És habitual que en administrar un servidor haguem de dissenyar els nostres propis serveis per fer alguna tasca concreta. En la majoria de sistemes operatius (Unix, Windows, etc.), la realització de guions de *shell* ens permet l'automatització de l'inici i l'aturada de serveis.

Vegeu la implementació de *shell scripts* per automatitzar tasques d'administració del sistema en l'apartat "Automatització de tasques del sistema" d'aquesta unitat.

1.2.3 Shells del sistema operatiu Windows

En els sistemes operatius Windows de Microsoft la interacció de l'usuari amb el sistema operatiu es fa generalment utilitzant la interfície gràfica integrada en el propi sistema operatiu, però també disposem d'intèrprets d'ordres que ens permeten interaccionar i crear guions per automatitzar tasques. Els intèrprets d'ordres existents per a aquests sistemes són els següents:

Es pot accedir a l'intèrpret d'ordres cmd.exe dels sistemes Windows a *Inici* > *Executar* > *cmd*.

- **COMMAND.COM** és el nom de l'intèrpret d'ordres per a DOS i per a versions de Windows de 16/32 bits (95/98/98 SE/Me). Té dos modes d'execució, el mode interactiu, en el qual l'usuari escriu ordres per ser executades, i el mode per lots (*batch*), que executa una seqüència predefinida d'ordres guardada en un arxiu de text amb l'extensió *.bat* i que se sol anomenar *programa per lots*.

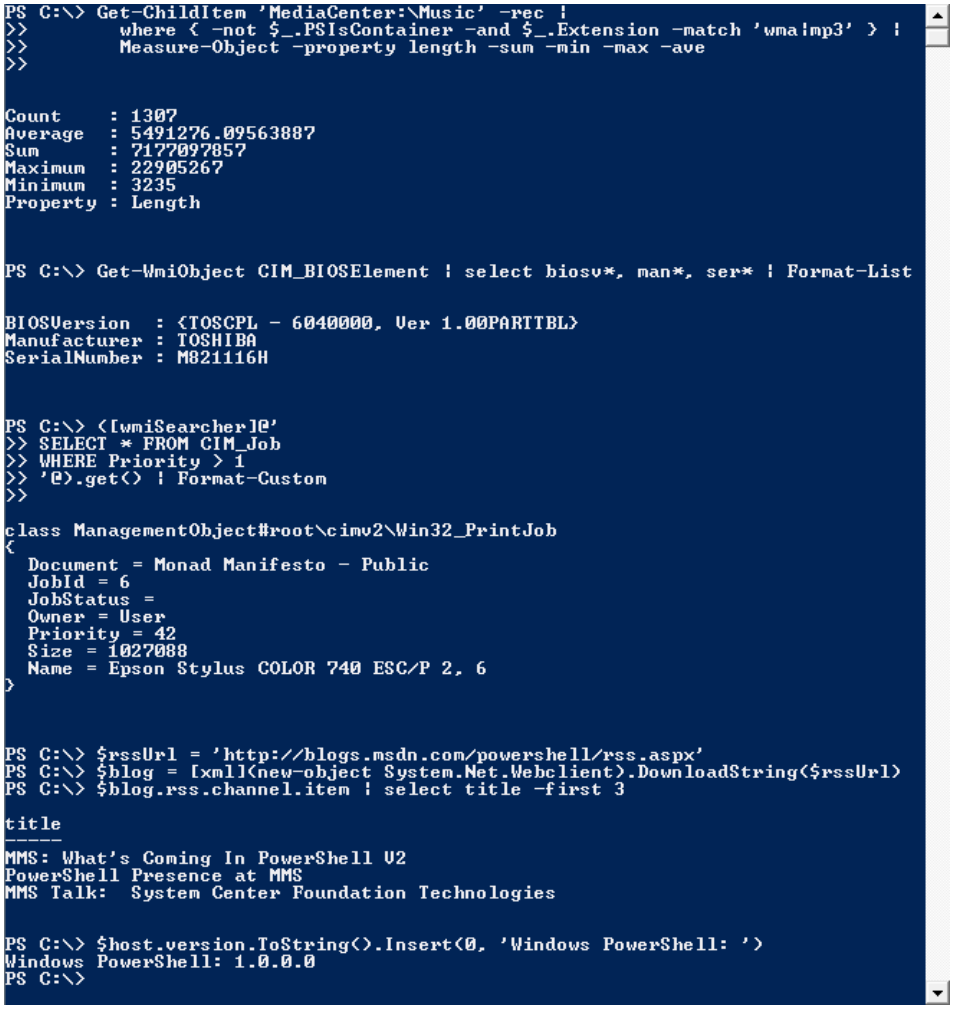
- L'executable **cmd.exe** és l'intèrpret d'ordres dels sistemes basats en Windows NT (incloent Windows 2000, XP, Server 2003, Vista i 7). És l'equivalent de COMMAND.COM a MS-DOS i sistemes de la família Windows 9x. En realitat, cmd.exe és un programa de Windows que actua com un intèrpret d'ordres de tipus DOS. En general és compatible, però proporciona extensions que eliminen algunes de les limitacions de COMMAND.COM.

- Windows **PowerShell**és una interfície de consola per a sistemes operatius Windows llançada el 2006 que té com a utilitat principal l'automatització de tasques administratives. Les funcions d'interpretació i de programació de guions estan molt millorades respecte a cmd.exe. Aquesta interfície no està instal·lada per defecte en el sistema i requereix de la instal·lació prèvia

del *framework* (entorn de treball) .NET versió 2.0 per al seu funcionament. Permet interactuar amb el sistema operatiu i amb programes de Microsoft com SQL Server, Exchange o IIS. La característica distintiva d'aquest intèrpret d'ordres respecte als tradicionals és que està **orientat a objectes**. La figura 1.4 mostra una imatge d'una finestra de Windows PowerShell.

Vegeu més informació sobre el llenguatge de guions Windows PowerShell a la pàgina oficial de Microsoft "Scripting with Windows PowerShell", que trobareu a la secció d'enllaços d'interès dels materials web del mòdul.

FIGURA 1.4. Finestra de Windows PowerShell



1.2.4 Shells del sistema operatiu Unix i derivats

Un sistema operatiu de tipus Unix o derivat és aquell que comparteix moltes de les característiques del sistema operatiu Unix, que va ser escrit al 1969 per, entre d'altres, Ken Thompson, Dennis Ritchie i Douglas McIlroy als Laboratoris Bell. En l'actualitat hi ha molts sistemes operatius de tipus Unix, com ara totes les distribucions GNU/Linux o el sistema operatiu Mac OS X.

Sistemes operatius de tipus Unix

Una distribució GNU/Linux o simplement distribució Linux és un sistema operatiu de tipus Unix que consisteix en el nucli de Linux, llibreries i utilitats del projecte GNU, i un conjunt de programari de tercers que permet afegir una sèrie d'aplicacions d'ús molt ampli, com ara escriptori gràfic, ofimàtica, navegadors, etc. Tant el nucli com la majoria de programari que

formen les distribucions Linux són de codi lliure. Existeixen centenars de distribucions, en constant revisió i desenvolupament. Alguns dels noms més coneguts són Fedora (Red Hat), openSUSE (Novell), Ubuntu (Canonical Ltd), Mandriva Linux (Mandriva) i les distribucions Debian i Gentoo les quals tenen un model de desenvolupament independent d'empreses i estan creades pels seus propis usuaris.

Mac OS X és el sistema operatiu de codi propietari (amb part de codi obert) basat en Unix que utilitzen els ordinadors Macintosh de la companyia Apple Inc. A la X que acompanya el nom Mac OS se li atribueixen dos significats: el de numeral romà 10 (ja que les versions del Mac OS anomenat "Classic" acaben al 9) i la darrera lletra del mot Unix.

En els sistemes Unix i derivats, la interacció de l'usuari amb el sistema operatiu es pot fer amb mode gràfic o amb mode text. En aquests tipus de sistemes hi ha una gran varietat d'intèrprets d'ordres. A la taula 1.1 en podem veure el nom dels més rellevants amb una breu descripció i les seves característiques.

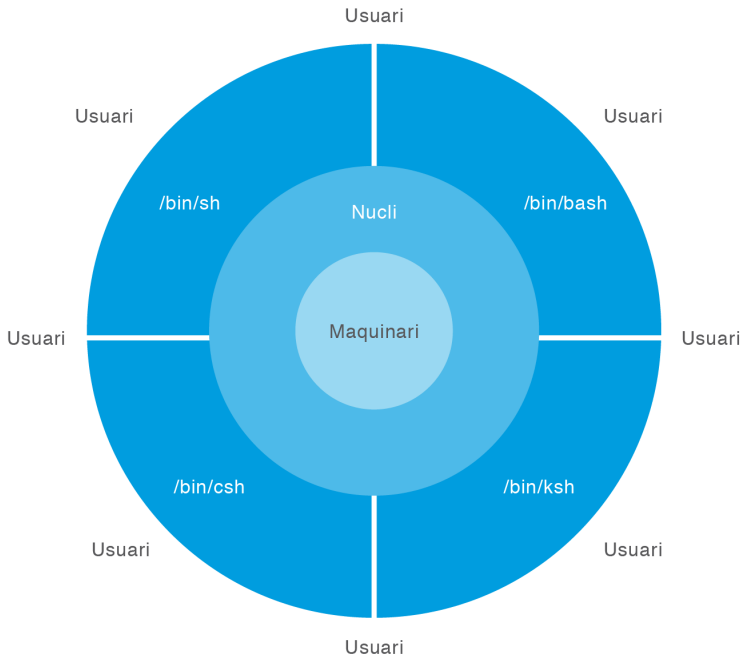
TAULA 1.1. Alguns dels shells disponibles als sistemes de tipus Unix

Nom del *shell*	**Programa**	**Descripció i característiques**
Bourne	/bin/sh	És un dels primers *shells* que es van desenvolupar i és un dels intèrprets més coneguts. Va ser escrit per Steve Bourne als Laboratoris Bell i es va convertir en un estàndard de facto.
Almquist	/bin/ash	Va ser escrit com una substitució del *shell* Bourne però amb llicència BSD.
Bash	/bin/bash	Basat en el *shell* Bourne, va ser escrit com a part del projecte GNU. És el *shell* estàndard de la majoria de sistemes GNU/Linux i de Mac OS X.
Debian Almquist	/bin/dash	Una substitució del *shell* ash per a les distribucions Debian i basades en Debian, com ara Ubuntu.
C	/bin/csh	Es tracta del segon *shell* d'Unix. Fou dissenyat per facilitar l'ús interactiu afegint-hi noves funcions. La seva sintaxi és molt semblant al llenguatge C de programació i el seu ús és més comú en els sistemes Unix de Berkeley.
TENEX C	/bin/tcsh	Essencialment és el *shell* C amb alguna característica millorada de la línia d'ordres.
Korn	/bin/ksh	És el tercer *shell* d'Unix, fruit de l'evolució del *shell* Bourne amb característiques del *shell* C. És un *shell* estàndard del sistema Unix System V.
Z	/bin/zsh	És considerat el *shell* més complet, en el sentit que és el que té més funcionalitats. Engloba característiques d'sh, ash, bash, csh, ksh i tcsh.

Un dels aspectes que caracteritza un sistema operatiu de tipus Unix és el fet que és multiusuari, és a dir, permet l'ús del mateix ordinador a més d'un usuari al mateix temps (per exemple, a través d'SSH). Per a cada sessió d'usuari, el sistema

genera un *shell* propi per a ell que actua com a intèrpret interactiu de les ordres que l'usuari executa, la qual cosa permet que els usuaris puguin triar un *shell* o un altre d'acord amb les seves necessitats. La figura 1.5 mostra gràficament l'existència de diferents *shells* en el diagrama de capes d'Unix.

Figura 1.5. Diagrama de capes d'Unix amb diferents shells



El fet que el *shell* estigui separat de la resta del sistema operatiu proporciona flexibilitat, perquè permet seleccionar la interfície més adequada a les necessitats de cada usuari.

El fitxer `/etc/shells` dóna informació sobre els *shells* vàlids per iniciar sessió en un sistema Unix determinat. Que siguin vàlids no significa que estiguin instal·lats, si es volen utilitzar cal que prèviament ens assegurem que estan instal·lats al sistema.

Exemple de contingut del fitxer /etc/shells d'un sistema Debian

```
# /etc/shells: valid login shells
/bin/csh
/bin/sh
/usr/bin/es
/usr/bin/ksh
/bin/ksh
/usr/bin/rc
/usr/bin/tcsh
/bin/tcsh
/usr/bin/esh
/bin/dash
/bin/bash
/bin/rbash
```

El *shell* que s'executa per defecte quan un usuari inicia sessió queda definit en el fitxer `/etc/passwd`, en el darrer camp de la línia corresponent a l'usuari.

Exemple de línia d'un fitxer /etc/passwd

La següent és una línia d'una usuària anomenada "mia" del fitxer `/etc/passwd`. En el darrer camp hi ha escrit `/bin/bash`, cosa que indica el shell per defecte de la usuària:

```
mia:x:1011:1000:Mia Maya:/home/mia:/bin/bash
```

Normalment, l'administrador del sistema és l'encarregat de fixar el *shell* d'inici de sessió dels usuaris. En alguns entorns, està habilitat l'ús d'ordres com `chsh` per permetre als usuaris canviar el seu *shell* per defecte d'inici de sessió.

Exemple d'utilització de l'ordre /usr/bin/chsh

Imaginem que la usuària "mia" està treballant en un entorn en què se li permet fer el canvi del seu shell d'inici i que el shell /bin/tcsh està instal·lat al sistema. La usuària té com a shell d'inici el bash i té la necessitat de canviar-lo pel tcsh. Per això executa l'ordre següent:

```
1 chsh –s /bin/tcsh
```

El resultat de l'ordre anterior és un canvi en la línia del fitxer `/etc/passwd` corresponent a la usuària "mia". En el darrer camp ara hi haurà escrit `/bin/tcsh` i a partir d'aquell moment totes les sessions noves executaran aquest *shell* per defecte.

```
mia:x:1011:1000:Mia Maya:/home/mia:/bin/tcsh
```

Si hem iniciat sessió com a usuaris amb un *shell* determinat, en la mateixa sessió podem arrencar un nou *shell* –al qual anomenem ***subshell*** o ***shell* fill**–, que pot ser del mateix tipus o d'un de diferent disponible en el sistema. Per iniciar un *subshell* senzillament hem d'escriure el nom del programa (sh, bash, csh, ksh, etc.) a la línia d'ordres i prémer la tecla de retorn. Si hem canviat de *shell*, en general, apareixerà un indicador de línia d'ordres diferent, ja que cada *shell* té una aparença diferent. Podem finalitzar l'execució del *subshell* i retornar al *shell* pare escrivint l'ordre *exit*.

1.3 El shell Bash

El *shell* **Bash** és l'intèrpret d'ordres predeterminat de gairebé totes les distribucions GNU/Linux, així com de Mac OS X, i pot executar-se en la majoria dels sistemes operatius tipus Unix. També s'ha portat a Microsoft Windows per al projecte Cygwin.

El nom Bash

És un acrònim de Bourne-Again Shell ("un altre *shell* Bourne"), fent un joc de paraules (*born-again* significa "renaixement") sobre el *shell* Bourne (sh), que va ser un dels primers intèrprets importants d'Unix.

El Bash és un intèrpret d'ordres compatible amb el *shell* Bourne (sh), que incorpora característiques útils del *shell* Korn (ksh) i del *shell* C (csh), amb l'objectiu de complir amb l'estàndard **POSIX**. Bash ofereix millores funcionals sobre sh tant per a la programació com per a l'ús interactiu i s'ha convertit en l'estàndard de facto per a la programació de guions de *shell*.

Com que el *shell* Bash (bash) és compatible amb el *shell* Bourne (sh), les ordres i els programes escrits per a sh poden ser executats amb bash sense cap modificació, però el contrari no sempre és cert.

Queda fora de l'abast d'aquesta unitat fer un estudi de totes les característiques del *shell* Bash i únicament ens centrarem en aquelles funcionalitats que són necessàries per poder abordar correctament la programació de *shell scripts* i l'automatització de tasques del sistema.

POSIX

És l'acrònim de *portable operating system interface*. La X prové d'Unix, com a símbol d'identitat de l'API. Una traducció aproximada de l'acrònim podria ser "interfície de sistema operatiu portàtil basat en Unix".

1.3.1 Obrir una sessió amb Bash

Quan arrenca, el sistema presenta als usuaris una interfície determinada que pot ser de text o gràfica, depenent dels sistemes operatius o de com estigui configurat el mode d'arrencada.

La interacció amb Bash es fa a través d'un emulador de terminal. Si hem iniciat sessió al sistema en mode consola (text), una vegada validats obtindrem l'accés directe a un *shell*. Si hem iniciat sessió en mode gràfic, haurem d'executar algun dels programes d'emulació de terminal disponibles. La majoria de sistemes Unix disposen d'una aplicació de terminal accessible per alguna de les opcions del menú principal, per exemple, a la distribució de Linux Debian accedim a un terminal des d'*Aplicacions* > *Accessoris* > *Terminal*.

Els usuaris de Mac OS X podeu accedir a un *shell* Bash via *Finder* > *Aplicacions* > *Utilitats* > *Terminal*.

Un altre cas d'obtenció d'un intèrpret d'ordres interactiu és l'accés remot a la màquina, tant per qualsevol de les possibilitats de text (Telnet, rlogin, SSH) com per les gràfiques (per exemple, amb emuladors X Window).

Sigui quin sigui el mode d'accés, en iniciar sessió el sistema genera un *shell* per a nosaltres que farà d'intèrpret de les ordres que executem en aquella sessió. Per a això el *shell* fa el següent:

- Mostra per pantalla l'indicador de la línia d'ordres (el *prompt*) assenyalant que està llest per acceptar ordres.
- Quan l'usuari introdueix una ordre, el *shell* la interpreta (busca les ordres, fa l'expansió de noms de fitxers, substitueix els valors de variables per variables referenciades, etc.).
- Si troba algun error mostra a l'usuari un missatge d'error.
- Si l'ordre està ben escrita, aleshores localitza el programa que cal executar i demana al sistema operatiu que l'executi, passant-li a ell el control.
- Quan finalitza l'execució del programa, el control retorna al *shell*, que torna a mostrar el *prompt* esperant una nova ordre.

Vegeu més informació sobre el *shell* per defecte a l'apartat "*Shells* del sistema operatiu Unix i derivats" d'aquesta unitat.

Després d'iniciar sessió, podem comprovar quin és el *shell* que tenim establert per defecte i amb el que estem treballant executant l'ordre següent:

```
1 echo $SHELL
```

1.3.2 Interpretació d'ordres

La funció principal del *shell* és interpretar ordres, ja sigui les ordres escrites de manera interactiva a la interfície de la línia d'ordres que proporciona, o les ordres escrites en un fitxer de text (guió de *shell*).

Les ordres que escrivim, ja sigui en l'indicador de la línia d'ordres del *shell* o en un programa de *shell*, tenen el format següent:

```
nom_ordre [–opcions] [arguments] <return>
```

On:

- *nom_ordre*: és el nom de l'ordre que volem executar
- *opcions*: les ordres poden o no portar opcions. Normalment les opcions s'escriuen amb un guió davant.
- *arguments*: depenent de l'ordre, es poden posar arguments que moltes vegades representen una cadena de caràcters, el nom d'un fitxer o directori.

El *shell* interpreta sempre l'espai en blanc com a separador d'ordres, opcions o arguments. Si no posem els espais correctament, obtindrem un missatge d'error.

Exemple d'execució d'ordres

És possible que una mateixa ordre accepti diferents modes d'execució, a soles, amb opcions, amb arguments. Per exemple l'ordre ls:

```
ls
ls –l
ls /etc/shells
ls –l /etc/shells
```

L'ordre següent dóna error perquè no hem posat l'espai requerit per separar l'opció i l'argument:

```
ls –l/etc/shells
```

1.3.3 Expansió de noms de fitxers

El *shell* ens proporciona una característica que s'anomena **generació de noms defitxers** o **expansió de noms de fitxers**,que ens estalvia temps a l'hora de teclejar els noms dels fitxers amb els quals operen les ordres.

La generació de noms de fitxers permet utilitzar uns caràcters especials per especificar grups de noms de fitxers.

Es poden trobar noms de fitxers o directoris que compleixen un patró determinat, per exemple, tots els noms que acaben en *.c*, o tots els que comencen per *test*, o tots els que tenen tres caràcters. Mitjançant l'ús de les **expressions regulars**, podem referir-nos als noms dels fitxers que compleixen el patró determinat. El *shell* expandeix el patró corresponent als noms de fitxers abans d'executar l'ordre. La taula 1.2 mostra les expressions regulars utilitzades per a la generació de noms de fitxers.

Expressions regulars

En informàtica, una expressió regular és una representació, segons unes regles sintàctiques d'un llenguatge formal, d'una porció de text genèric a buscar dins d'un altre text, com per exemple uns caràcters, paraules o patrons de text concrets. El text genèric de l'expressió regular pot representar patrons amb determinats caràcters que tenen un significat especial, com ara el caràcter interrogant, ?,per representar un caràcter qualsevol; el caràcter comodí, *, per representar un nombre qualsevol de caràcters, etc.

Taula 1.2. Expressions regulars usades en la generació de noms de fitxers

Expressió	Descripció
?	Coincidència amb qualsevol caràcter simple, excepte el punt a l'inici del nom.
*	Coincidència amb zero o més caràcters (excepte el punt inicial).
[]	Coincidència amb qualsevol dels caràcters tancats.
[!]	Coincidència amb qualsevol dels caràcters no tancats.
[-]	Coincidència amb qualsevol dels caràcters del rang.

Les expressions regulars de generació de noms de fitxers no generen mai noms de fitxers que comencen per punt, el punt sempre s'ha d'indicar explícitament.

Exemple de generació de noms de fitxers amb expressions regulars

Situeu-vos en el directori /usr/bin i llisteu els noms de fitxers i directoris que comencin per c:

```
ls c*
```

Els que comencin per la lletra y o z:

```
ls [yz]*
```

Els que acabin en .sh:

```
ls *.sh
```

Els que comencin per r i acabin en e:

```
ls r*e
```

Els que comencin per alguna lletra compresa entre la a i la d:

```
ls [a-d]*
```

Tots aquells fitxers el nom dels quals tingui quatre lletres:

```
ls ????
```

1.3.4 Variables del shell

Al *shell*, una variable és un nom que representa un valor. La sintaxi per definir i assignar un valor a una variable és:

```
nom_variable=valor
```

Fixeu-vos que no hi ha cap espai abans ni després del signe igual. Si se'n posa algun obtindrem un error.

Els següents són exemples vàlids de definició de variables:

```
NOM=Marc
COGNOMS="Ros Roig"
EDAT=31
```

El valor assignat a la variable COGNOMS l'hem tancat entre cometes dobles perquè el *shell* no interpreti l'espai que hi ha entre els dos cognoms. Per evitar la possibilitat de cometre errors, podem optar per posar el valor sempre entre cometes dobles:

```
nom_variable="valor"
```

El valor de la variable sempre el podem canviar assignant-li un altre valor. Per exemple:

```
NOM=Mia
```

El valor associat a una variable pot ser utilitzat mitjançant el nom de la variable precedit amb el símbol dòlar: `$nom_variable`. Aquest mecanisme es diu **substitució de variables**.

> El *shell* realitza la substitució de variables a qualsevol línia d'ordres que contingui un símbol $ seguit d'un nom de variable vàlid.

Podem visualitzar el valor d'una variable definida amb l'ordre `echo` i utilitzant el mecanisme de substitució de variables:

```
echo $nom_variable
```

Per exemple:

```
echo $COGNOMS
```

Després d'executar l'ordre anterior, el valor associat a la variable anomenada COGNOMS es mostra per pantalla.

En la substitució de variables es pot fer ús de claus per delimitar el nom de la variable: `${nom_variable}`. La utilització de claus ens permet, per exemple, fer la substitució de la variable seguida d'un text. Per exemple:

```
PREFIX=extra
echo ${PREFIX}ordinari
```

L'ordre anterior mostra per pantalla la paraula "extraordinari". Fent ús de la mateixa variable PREFIX podem escriure l'ordre següent:

```
echo "Mots que comencen per $PREFIX: ${PREFIX}ordinari, ${PREFIX}polar, ${
    PREFIX}murs, etc".
```

L'ordre anterior mostra per pantalla la frase "Mots que comencen per extra: extraordinari, extrapolar, extramurs, etc.".

Per convenció, els noms de les variables s'escriuen en majúscules, però es poden posar en minúscules. Els noms de les variables **han de començar obligatòriament** per caràcter alfabètic (*a-z*, *A-Z*) i poden contenir caràcters alfabètics, numèrics i subratllats. No hi ha restriccions respecte al nombre de caràcters que pot tenir el nom d'una variable.

Les variables poden ser de dos tipus, variables locals o variables d'entorn:

- **Variables locals**: només són visibles pel *shell* en el qual estem treballant, no són visibles per cap *shell* fill.
- **Variables d'entorn**: són visibles tant pel *shell* pare com pels *shells* fills. En ser creat, el *shell* fill "hereta" les variables d'entorn del pare (en rep una còpia). El *shell* fill pot canviar les variables heretades del *shell* pare, però, com que són una còpia, els canvis fets en el fill no afecten el pare.

> Una variable definida en un procés fill, ja sigui local o d'entorn, no és visible pel seu procés pare.

La manera de fer una variable d'entorn és mitjançant l'ordre `export`:

```
export NOM_VAR
```

L'assignació i exportació es pot fer en un sol pas:

```
export NOM_VAR=valor
```

Exemple de manipulació de variables locals i d'entorn

Executeu de manera seqüencial les línies d'ordres següents:

```
x=11    # Definim una variable x amb valor 11
echo $x # Mostrem el valor de x
bash    # Obrim un shell fill
echo $x # El shell fill no coneix el valor de x
exit    # Sortim del shell fill i retornem al pare
export x# Exportem la variable x a l'entorn
bash    # Obrim un shell fill
echo $x # Ara el shell fill sí coneix el valor de x
x=12    # Modifiquem en el shell fill el valor de x
exit    # Tornem al shell pare
echo $x # El valor de x segueix sent 11
```

Hi ha diverses ordres relacionades amb les variables:

- Ordre `set`: permet veure totes les variables (locals i d'entorn) definides en una sessió.
- Ordre `env`: permet veure les variables d'entorn definides en una sessió.
- Ordre `unset nom_variable`: elimina la variable i el valor associat a la variable.

En qualsevol sessió de *shell* Bash hi ha presents una sèrie de **variables d'entorn predefinides** pel sistema que ens poden resultar de molta utilitat en la programació de guions de *shell*, ja que guarden informació genèrica de l'entorn dels usuaris.

La taula 1.3 mostra el nom i la descripció d'algunes d'aquestes variables.

Localització

Localització es refereix al conjunt de convencions que es fan servir en una zona geogràfica o cultural determinada, com ara el format de la data i l'hora, el nom de la moneda, el separador per a desenes i centenes, etc. L'ordre locale mostra informació sobre el valor de les variables de localització del sistema.

Exemple d'ús de variables d'entorn predefinides

Fent ús de la substitució de variables i utilitzant variables d'entorn predefinides escriviu una línia d'ordres que en executar-se mostri per pantalla el text:

"Sóc nom_usuari i el meu directori de treball és directori".

Els valors de nom_usuari i directori s'han d'obtenir de les variables adequades i predefinides pel sistema.

```
echo "Sóc $USER i el meu directori de treball és $HOME"
```

Feu el mateix, però ara heu de mostrar:

"Treballo amb un sistema sistema_operatiu i un shellnom_shell"

```
echo "Treballo amb un sistema $OSTYPE i un shell $SHELL"
```

Taula 1.3. Algunes de les variables predefinides del shell Bash

Nom variable	Descripció
BASH	Camí del programa Bash que s'està executant.
BASH_VERSION	Versió del Bash que utilitzem.
COLUMNS	Nombre de columnes a la pantalla.
HISTFILE	Fitxer on es guarda l'històric de les ordres executades per l'usuari.
HOME	Directori d'inici de l'usuari.
HOSTNAME	Nom de l'ordinador.
LANG	Usada per determinar el valor de localització per defecte per qualsevol categoria no especificada amb la corresponent variable LC_ (LC_NUMERIC, LC_TIME, LC_MONETARY, etc.).
LC_ALL	Aquesta variable sobreescriu el valor de LANG i de qualsevol variable LC_ que especifiqui una categoria de localització.
LOGNAME	Nom de l'usuari connectat.
MACHTYPE	Arquitectura de l'ordinador.
OSTYPE	Tipus de sistema operatiu que estem utilitzant.
PATH	Llista de directoris on el *shell* ha de localitzar els programes quan els escrivim sense indicar el camí on es troben.
PPID	Identificador del procés pare.
PS1	Indicador de la línia d'ordres primari.
PS2	Indicador de la línia d'ordres secundari.
PWD	Camí del directori actual, és a dir, el directori on estem situats.
RANDOM	Número aleatori.
SECONDS	Temps en segons des que s'ha iniciat el *shell*.
UID	Identificador de l'usuari actual.

1.3.5 Substitució d'ordres

La substitució d'ordres s'utilitza per reemplaçar la sortida de l'execució d'una ordre dins de la mateixa línia d'ordres.

Es pot fer amb la sintaxi següent:

```
1 $(ordre)
```

O bé tancant l'ordre entre accents greus: `` `ordre` ``.

Les dues maneres són vàlides però si estem escrivint un *shell script* és millor triar-ne una i fer servir la mateixa manera al llarg de tot el programa.

En els exemples d'aquesta unitat utilitzarem `$(ordre)` i no `` `ordre` `` per fer la substitució d'ordres.

De la mateixa manera que en la substitució de variables, la substitució d'ordres es fa sempre abans d'executar la línia d'ordres. Quan el *shell* troba a la línia d'ordres

un *$* seguit d'un parèntesi obert, executa tot el que troba fins a arribar al parèntesi tancat. El resultat el posa a la línia d'ordres en el lloc on hi havia l'ordre que substituïa.

Exemple de substitució d'ordres

Fent ús de la substitució d'ordres escriviu una línia d'ordres que en executar-la mostri per pantalla el text: "La data del sistema és: data". El valor de data s'ha d'obtenir del resultat d'executar l'ordre adequada.

```
1 echo "La data del sistema és: $(date)"
```

També es podria haver fet així: `echo ‘‘La data del sistema és: `date`’’`.

La substitució d'ordres és un mecanisme molt utilitzat en la programació de guions de *shell*. Moltes vegades es fa servir en l'assignació de valors a variables. Per exemple:

L'ordre `date` admet diversos modificadors de format de sortida que s'indiquen amb *+%*. Per exemple, `date +%x` indica la data del sistema amb format dd/mm/aa.

```
1 HORA=$(date +%H)
```

Si la sortida de l'ordre que substituïm conté salts de línia, el *shell* els substituirà per espais en blanc. Per exemple, executeu l'ordre següent:

```
1 seq 10
```

L'ordre `seq` és utilitzada per generar seqüències de números, vegeu la seva sintaxi amb man seq.

L'ordre anterior mostra per pantalla els números de l'1 al 10 separats per salts de línia. Ara poseu la mateixa ordre en una substitució d'ordres, per exemple:

```
1 echo $(seq 10)
```

El resultat de l'ordre anterior és la llista de números separats per espais.

Exemple d'ús de variables i substitució d'ordres

Assigneu a una variable la llista de nombres parells que hi ha entre el 0 i el 20 (inclosos) separats per espais:

```
1 PARELLS=$(seq 0 2 20)
```

1.3.6 Expansió aritmètica

El mecanisme d'expansió aritmètica del Bash permet l'avaluació d'una expressió aritmètica i la substitució del resultat.

El format per fer l'expansió aritmètica és:

```
$((expressió_aritmètica))
```

No confongueu el mecanisme de substitució d'ordres, `$(ordre)`, amb el d'expansió aritmètica, $((expressió)).

L'avaluació d'expressions aritmètiques només admet **nombres sencers**. Els operadors admesos són gairebé els mateixos que en el llenguatge de programació C. La taula 1.4 mostra la llista d'operadors existents (no incloem els que operen amb bits) **en ordre decreixent de precedència**. Podem utilitzar parèntesis per alterar l'ordre de precedència dels operadors.

Taula 1.4. Operadors aritmètics

Operador	Descripció
VAR++, VAR–	Postincrement i postdecrement de la variable
++VAR, –VAR	Preincrement i predecrement
-, +	Menys i més unaris
!	Negació lògica
**	Potència
*, /, %	Multiplicació, divisió i residu
+, -	Suma i resta
<=, >=, <, >	Operadors de comparació
==, !=	Igualtat i desigualtat
&&	AND lògic
||	OR lògic
expr ? expr : expr	Avaluació condicional
=, *=, /=, %=, +=, -=	Assignacions
,	Separador d'expressions

En les expressions s'admeten variables del *shell* com a operands i podem utilitzar-les fent l'expansió de la variable $NOM_VAR, o bé només amb el seu nom, NOM_VAR. Per exemple, donades dues variables amb els valors següents:

```
X=2
Y=3
```

L'expressió següent:

```
Z=$((X+Y))
```

També la podem escriure així:

```
Z=$(($X+$Y))
```

El mecanisme d'expansió aritmètica ens permet realitzar operacions i substituir el resultat en una altra ordre, per exemple:

```
echo “El resultat de sumar $X i $Y és: $((X+Y))”
```

En l'exemple anterior fem servir l'ordre *echo* i en l'argument utilitzem l'expansió aritmètica directament per visualitzar el resultat de l'operació X+Y.

1.3.7 Tractament dels caràcters especials

Els caràcters especials són aquells que tenen algun significat concret per al *shell*, per exemple:

- El caràcter espai indica separació d'ordres, opcions o arguments.
- El caràcter salt de línia indica final d'ordre.
- El caràcter *$* davant d'un nom indica referenciar el valor d'una variable amb aquest nom.
- El caràcter * és utilitzat com a expressió regular en la generació de noms de fitxers.

Quan no volem que el *shell* interpreti aquests caràcters de manera especial sinó com a caràcters normals, podem anul·lar el seu significat de les maneres següents:

- Precedint el caràcter del símbol\. Aquesta tècnica anul·la el significat especial del caràcter que va darrera.
- Amb cometes dobles, " ". Aquesta tècnica anul·la el significat de tots els caràcters especials que estiguin dins les cometes dobles excepte el caràcter especial dòlar, *$*, la contrabarra, \, l'accent greu, `, i les cometes dobles, " ".
- Amb cometes simples,' '. Aquesta tècnica anul·la el significat de tots els caràcters especials que estiguin dins les cometes simples.

Exemple d'anul·lació del significat dels caràcters especials

Escriviu una ordre que mostri una frase com la següent: Estic llegint el llibre "Córrer o morir". El títol "Córrer o morir" ha d'aparèixer entre cometes dobles.

```
1 echo Estic llegint el llibre \"Córrer o morir\".
```

Es pot aconseguir el mateix amb les cometes simples:

```
1 echo 'Estic llegint el llibre "Córrer o morir".'
```

Escriviu una ordre que mostri la frase següent: El contingut de *$USER és: nom_usuari*. S'ha de substituir *nom_usuari* pel valor de la variable USER.

```
1 echo El contingut de \$USER és $USER.
```

1.3.8 Redirecció de l'entrada i la sortida

A Unix els dispositius d'entrada i sortida (E/S) i els fitxers es tracten de la mateixa manera i el *shell* els tracta a tots com a fitxers. Tots els programes executats mitjançant un *shell* inclouen tres fitxers predefinits, especificats pels descriptors de fitxers (*file handles*) corresponents. Per defecte, aquests fitxers són els següents:

- Entrada estàndard (*standard input*). Normalment està assignada al teclat. Utilitza el descriptor número 0.
- Sortida estàndard (*standard output*). Normalment està assignada a la pantalla. Utilitza el descriptor 1.
- Sortida estàndard d'errors (*standard error*). Normalment, està assignada a la pantalla. Utilitza el descriptor 2.

Això ens indica que, per defecte, qualsevol programa executat des del *shell* tindrà l'entrada associada al teclat, la seva sortida a la pantalla i, si es produeixen errors també els enviarà a la pantalla.

Una característica que ens proporciona el *shell* és poder redirigir l'entrada o la sortida estàndards d'una ordre, és a dir, ens permet fer que una ordre rebi la seva entrada o enviï la seva sortida des de o cap a altres fitxers o dispositius.

De manera que:

- La **redirecció d'entrada** permet que les ordres agafin les dades d'un fitxer enlloc de des del teclat.
- La **redirecció de sortida** ens permet enviar la sortida d'una ordre a un fitxer enlloc de a la pantalla.
- La **redirecció de la sortida d'errors** ens permet enviar la sortida d'errors d'una ordre a un fitxer enlloc de a la pantalla.

La taula 1.5 mostra els caràcters utilitzats per redirigir l'entrada i la sortida de les ordres.

TAULA 1.5. Redireccionament d'entrada i sortida

Caràcter	Descripció	Exemple
<	Redirecció d'entrada.	write usuari < "Hola"
>	Redirecció de sortida. Si el fitxer no existeix el crea i si ja existeix en sobreescriu el contingut.	date > registre.log
>>	Redirecció de sortida. Si el fitxer no existeix el crea i si ja existeix l'afegeix a continuació.	who >> registre.log
2>	Redirecció de sortida d'errors. Si el fitxer no existeix el crea i si ja existeix en sobreescriu el contingut.	ls /tmp/kk 2> registre.log
2>>	Redirecció de sortida d'errors. Si el fitxer no existeix el crea i si ja existeix l'afegeix a continuació.	ls /tmp/kk 2>>registre.log
2>&1	Redirecció de la sortida d'errors a la sortida estàndard.	ls /tmp/kk 2>&1
1>&2	Redirecció de la sortida estàndard a la sortida d'errors.	ls /tmp/kk 1>&2
>&	Redirecció de sortida i de sortida d'errors a un fitxer.	ls /tmp/kk >& registre.log

Si volem que l'execució d'una ordre no generi activitat per pantalla (execució silenciosa), hem de redirigir totes les seves sortides a /dev/null. Per exemple:

```
1 ls /tmp/kk >& /dev/null
```

Si volem redirigir la sortida estàndard i la sortida d'errors a un fitxer amb la doble redirecció, per tal que si el fitxer no existeix el creï i que si ja existeix afegeixi les dues sortides, podem fer-ho així:

```
1 ls /tmp/kk >> registre.log 2>> registre.log
```

1.3.9 Canonades o 'pipes'

El *shell* permet enllaçar la sortida d'una ordre com a entrada d'una altra ordre mitjançant el que anomenem **canonades** o ***pipes***.

La sintaxi és la següent:

```
1 ordre1 | ordre2
```

La sortida de l'ordre 1 s'utilitza com a entrada de l'ordre 2. El símbol | s'anomena *pipe* ("canonada" en anglès). Es poden posar espais entre la canonada i les ordres per fer més llegible la línia d'ordres, però no és obligatori, els espais són opcionals.

Per exemple, executem una ordre *echo* per mostrar per pantalla una línia de text:

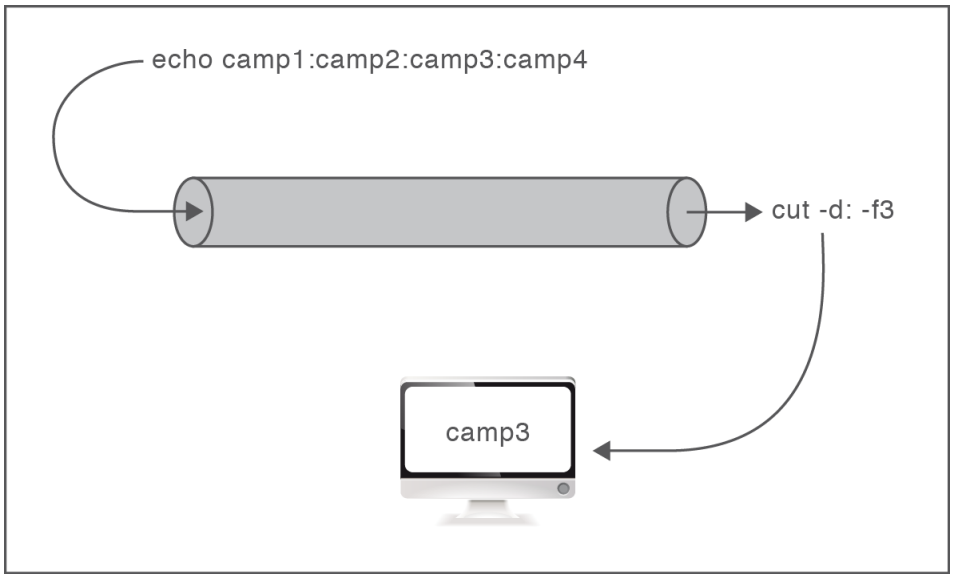
```
1 echo "Alba:Gomez:08004:BCN"
```

L'ordre anterior mostra per pantalla una serie de dades separades per dos punts així: "Nom:Cognoms:CP:Ciutat". Si d'aquesta sortida només volem prendre el tercer camp (el de codi postal), la podem redirigir amb una canonada cap a l'ordre *cut* perquè seleccioni únicament el camp que ens interessa de la manera següent:

```
1 echo "Alba:Gomez:08004:BCN" | cut –d: –f3
```

A la figura 1.6 podem veure aquest exemple de manera gràfica. La sortida de l'ordre *echo* passa per la canonada i la rep com a entrada l'ordre *cut*, la qual la processa i mostra per pantalla la sortida processada.

FIGURA 1.6. Funcionament de les canonades



Veiem un altre exemple senzill d'utilització de canonades:

```
1 cat /etc/passwd | more
```

En l'exemple anterior la sortida de l'ordre cat /etc/passwd s'utilitza com a entrada de l'ordre more per veure el contingut del fitxer /etc/passwd per pàgines, és a dir, fent una pausa cada vegada que s'omple la pantalla.

Amb l'ordre *more* avancem de pàgina amb la tecla d'espai.

Una línia d'ordres escrita amb una pipe es coneix com una ***pipeline***. La *pipeline* no està limitada només a dues ordres. Es poden fer *pipelines* més llargues associant la sortida d'una ordre amb l'entrada de la següent i així successivament.

```
1 ordre1 | ordre2 | ordre3 | ordre4 ...
```

Per exemple:

```
1 cat /etc/passwd | sort | more
```

En l'exemple anterior hi ha un encadenament de tres ordres amb transmissió de les seves dades d'una a l'altra: la sortida de cat s'envia d'entrada a sort i la sortida de sort s'envia d'entrada a more. El resultat final consisteix en mostrar les línies del fitxer /etc/passwd, ordenades i fent una pausa cada vegada que s'omple la pantalla.

La manera com enllacem les diferents ordres no és aleatòria, sinó que s'ha de fer de manera adequada per obtenir els resultats esperats. En la construcció de *pipelines* llargues convé ser metòdics per evitar errades, és preferible fer la *pipeline* pas a pas i anar comprovant els resultats intermedis per corregir els possibles errors. El procés a seguir seria el següent:

- Comprovem les dues primeres ordres de la *pipeline*: `ordre1 | ordre2`
- Si el resultat és correcte, aleshores afegim l'ordre següent del procés i comprovem el resultat intermedi: `ordre1 | ordre2 | ordre3`
- Si el resultat és correcte, afegim la següent: `ordre1 | ordre2 | ordre3 | ordre4`
- I així successivament fins a obtenir la *pipeline* final.

Quan encara no s'està familiaritzat amb l'ús de les canonades, a vegades es tendeix a confondre el seu propòsit i no s'utilitzen bé. No totes les ordres es poden enllaçar de qualsevol manera amb canonades. Per fer una *pipeline* hem de tenir en compte les consideracions següents:

- Una ordre pot tenir a la dreta una canonada si admet sortida estàndard. Per exemple, les ordres *ls*, *who*, *date*, *cat*, etc.
- Una ordre pot tenir a l'esquerra una canonada si admet entrada estàndard. Per exemple, les ordres *wall*, *write*, etc.
- Una ordre pot tenir una canonada a la seva esquerra i a la seva dreta si admet tant entrada com sortida estàndards. Per exemple, les ordres *sort*, *cut*, *grep*, etc.

One liners

Entre els administradors de sistemes és comú el terme one liners. Es refereix a un conjunt d'ordres unides generalment per canonades que enllaçades adequadament produeixen un resultat útil i pràctic. Per exemple, si volem saber quins són els cinc processos que consumeixen més CPU podem executar el següent:

```
1 ps -eo pcpu,user,pid,cmd | sort -r | head -6
```

Fixeu-vos que hem utilitzat l'opció o de l'ordre ps, que permet personalitzar la sortida, i hem indicat les columnes que volem que es mostrin. L'opció e de l'ordre indica extended, és a dir, els processos de tots els usuaris. Tota la sortida de ps és enviada a sort, que en fa un ordenament invers basant-se en la primera columna (pcpu). Per acabar, la sortida ordenada és enviada a l'ordre head, que mostrarà les sis primeres línies (la capçalera i els cinc processos amb més consum).

1.3.10 Filtres i 'pipelines'

A Unix hi ha unes ordres que es fan servir molt a les *pipelines* i que anomenem **filtres**. Els filtres són ordres que accepten dades de l'entrada estàndard, escriuen

la sortida a la sortida estàndard i escriuen els errors a la sortida estàndard d'errors, i mai modifiquen les dades d'entrada. A causa del seu funcionament, és freqüent utilitzar filtres a les *pipelines*.

Per exemple, un filtre molt utilitat és l'ordre *sort*, que serveix per ordenar línies de text. L'ordre accepta dades des de l'entrada estàndard (el teclat). Per introduir-les només cal escriure l'ordre i prémer la tecla de retorn:

```
1 sort
```

El cursor se situa a sota esperant línies d'entrada. Escrivim les línies de text que volem que l'ordre ordeni separades per salts de línia i, en acabar, hem de prémer la combinació de tecles *Ctrl+D* per indicar la finalització de l'entrada de dades. A continuació ens apareixeran per pantalla les línies ordenades.

Com que els filtres accepten les dades del teclat (entrada estàndard) i mostren el resultat per la pantalla (sortida estàndard), podem utilitzar-los en *pipelines* per processar mitjançant una canonada la sortida de qualsevol ordre que doni el resultat en línies de text. Per exemple:

Per comprendre les *pipelines* de l'exemple és recomanable que les executeu comprovant els resultats intermedis, seguint el procés descrit en l'apartat "Canonades o *pipes*" d'aquesta unitat.

```
1 who | sort
```

L'ordre *who* mostra una línia per a cada usuari connectat al sistema. En enviar aquesta sortida a l'entrada de *sort*, les línies es mostren ordenades per pantalla.

Vegem un altre exemple:

```
1 cat /etc/group | sort
```

L'ordre *cat* mostra les línies del fitxer /etc/group. En enviar aquesta sortida a l'ordre *sort*, les línies del fitxer es mostren ordenades per pantalla, però el fitxer no es modifica.

Els filtres no només accepten dades des de l'entrada estàndard, sinó que també accepten dades des d'un o més fitxers posats com arguments en la línia d'ordres. Aleshores, l'ordre anterior també es pot escriure així:

```
1 sort /etc/group
```

En general els filtres són molt útils per processar el contingut dels fitxers de text. Alguns dels més utilitzats són *wc*, *sort*, *grep* i *cut*:

- Comptar el nombre de línies d'un fitxer:

  `wc -l /etc/passwd`

- Ordenar alfabèticament les línies del fitxer:

  `sort /etc/passwd`

- Fer la recerca de línies que continguin un patró determinat:

  `grep root /etc/passwd`

- Extreure parts de les línies del fitxer:

```
cut -d:  -f1 /etc/passwd
```

Exemple de construcció de pipelines amb filtres

Compteu els fitxers del directori actual.

```
ls | wc -l
```

Mostreu els noms (només els noms) dels usuaris donats d'alta al sistema ordenats alfabèticament.

```
cat /etc/passwd | cut -f1 -d: | sort
```

Mostreu els noms dels usuaris donats d'alta en el sistema i el seu shell d'inici, ordenats alfabèticament i separant els dos camps per un tabulador.

```
cat /etc/passwd | cut -f1,7 -d: | sort | tr ":" "\t"
```

Mostreu l'identificador (PID) i el nom (CMD) de tots els processos que pertanyen a l'usuari root.

```
ps -ef | grep "^root " | tr -s " " | cut -f2,8 -d" "
```

Mostreu una llista amb el propietari, grup i nom de fitxer de tots els fitxers que hi ha a /etc. La llista ha d'estar ordenada pel nom del grup del fitxer.

```
ls -l /etc | tr -s " " | cut -f3,4,9 -d" " | sort -k2 -t" "
```

2. Programació del shell Bash

El *shell* Bash (Bourne-Again *shell*) és un intèrpret d'ordres, un programa informàtic que té la funció d'interpretar ordres. El *shell* ens proporciona la possibilitat de programar l'execució d'un conjunt d'ordres amb el seu propi llenguatge i d'emmagatzemar-les en un fitxer, que executarem com qualsevol altra ordre del sistema. Aquest fitxer d'ordres s'anomena *guió de shell oshell script*. Es poden escriure guions de *shell* complexos, ja que el *shell* admet variables, paràmetres, entrada i sortida de dades interactiva, comparacions, bifurcacions, bucles, etc.

L'única manera d'aprendre *shell scripting* és fent guions de *shell*. Per això és molt recomanable que a mesura que aneu avançant en la lectura d'aquest apartat copieu els exemples i els proveu al vostre sistema, i que aneu fent les activitats d'aprenentatge proposades al web de la unitat.

2.1 Creació i execució d'un guió de shell

Un **guió de *shell*** o ***shell script*** és un conjunt d'ordres emmagatzemades en un fitxer de text pla per poder ser executades posteriorment amb el nom del fitxer. En el guió podem incloure la crida a qualsevol programa que sigui executable pel *shell* (ordres del sistema, altres guions de *shell*, etc.), així com crides a funcions, estructures de control del llenguatge del *shell*, comentaris, etc.

> Cada ordre que escrivim en un guió ha d'anar separada per un salt de línia o bé pel caràcter *;* (punt i coma) si està a la mateixa línia.

Les instruccions del guió s'executen seguides una darrere de l'altra en l'ordre en que estan escrites, com si les estiguéssim escrivint una a una a la línia d'ordres, i el salt de línia o el caràcter punt i coma s'interpreten com si preméssim *Retorn* després de cada ordre.

2.1.1 Creació i nom del fitxer

Per crear un *shell script* n'hi ha prou amb obrir un nou fitxer buit en un editor de text, escriure la seqüència d'ordres que volem que s'executin i desar el fitxer amb el nom que li volem donar al guió.

El fitxer del guió de *shell* ha d'estar compost únicament per text sense format i per això hem d'utilitzar un **editor de text pla**.

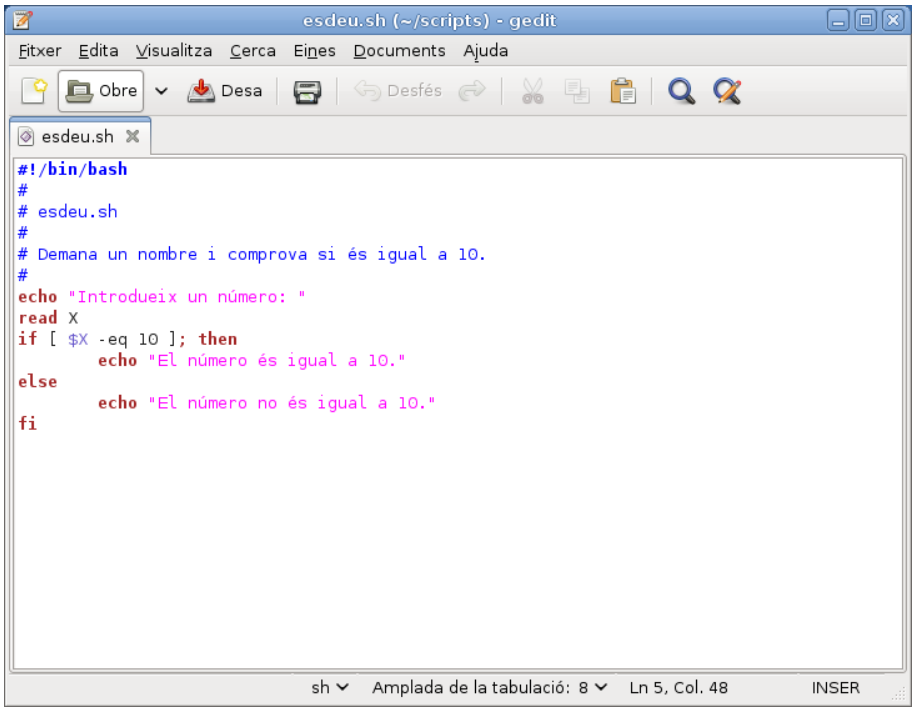
Editors de text avançats

El vi (o el vim a Linux) i l'emacs són editors de text molt potents que s'han utilitzat històricament en entorns Unix, però que poden resultar incòmodes per a usuaris novells. En teniu d'altres més fàcils d'utilitzar, com ara el gedit que es troba per defecte en l'entorn d'escriptori GNOME. I n'hi ha molts altres que no hi són per defecte però que us podeu instal·lar a part.

Els editors de texts plans es distingeixen dels processadors de textos en que es fan servir per escriure només text, sense format i sense imatges, és a dir sense diagramació. També podem optar per fer servir un editor de text avançat que reconegui el llenguatge del *shell* i utilitzi diferents colors per al ressaltat de la sintaxi, com el de la figura 2.1. Aquesta mena d'editors són especialment útils quan som principiants, ja que ens ajuden a prevenir errors de sintaxi, com ara oblidar de posar una clau de tancament, unes cometes dobles, etc.

El nom *vim* ve de *vi improved*. És compatible amb el vi però hi afegeix algunes funcionalitats.

Figura 2.1. Editor gedit de l'escriptori GNOME



El nom que donem al fitxer pot ser qualsevol nom vàlid d'acord amb les normes que hi ha per anomenar fitxers en el sistema en què estem treballant. Normalment triarem un nom que sigui representatiu del que fa el guió, per exemple, si fa una còpia de seguretat, podem anomenar-lo *copia*. En tot cas convé que ens assegurem que el nom no entra en conflicte amb d'altres programes o ordres existents al sistema. Per garantir això, els noms dels guions sovint acaben amb l'extensió *.sh*.

Podem anomenar els nostres guions amb l'extensió *.sh*. Ara bé, això només és una convenció i no és obligatori fer-ho.

Noms de fitxers

Les ordres which, whereis i locate ens serveixen per verificar si ja existeixen fitxers amb un nom determinat. Per exemple, si executem which date ens retorna /bin/date, que indica el camí complet on podem trobar aquest fitxer. Si no ens retorna res significa que no localitza cap fitxer amb aquest nom.

Per començar, obriu un editor de text i creeu un nou fitxer que contingui les quatre línies següents:

```
1 #!/bin/bash
2 # El meu primer guió de shell
3 echo "Hola, món"
4 echo "Sóc $USER"
```

Tot i que ara no volem analitzar a fons el significat de cada una de les línies anteriors, en fem una descripció breu per saber què fa el guió:

- La primera línia indica al sistema que el programa Bash ubicat al directori /bin ha d'executar la resta d'instruccions que hi ha a l'script.
- La línia següent és un comentari i, per tant, el shell la ignora.
- La tercera línia és una ordre *echo*, que permet mostrar text per pantalla, en aquest cas, la frase "Hola, món".
- La darrera línia és una altra ordre *echo* que mostra per pantalla el missatge ''Sóc $USER'' substituint el valor de la variable USER pel nom de l'usuari que executa el guió.

Una vegada hem escrit les línies sense errors, desem el fitxer i li donem un nom per poder executar-lo des de la línia d'ordres.

2.1.2 Execució del guió de shell

La manera més habitual d'executar un script és obrir una sessió de terminal i executar-lo com qualsevol altra ordre del sistema, és a dir, escrivint el nom del fitxer que conté l'script en la línia d'ordres seguit d'un salt de línia. Per poder-ho fer cal que el fitxer tingui permís d'execució per als usuaris que han d'executar-lo. En el cas més simple, donarem permís d'execució al propietari del fitxer amb l'ordre següent:

Vegeu com obrir una sessió amb Bash en l'apartat "Obrir una sessió amb Bash" d'aquesta unitat.

```
1 chmod u+x nom_fitxer
```

Després d'afegir el permís d'execució al fitxer, podem executar-lo amb el seu nom tenint en compte les consideracions següents:

- Si el directori que conté el fitxer del *shell script* és a PATH, podem posar només el nom del fitxer que conté el guió perquè s'executi:

  ```
  # Executar script pel nom

  nom_fitxer
  ```

- Si el directori que conté el fitxer del *shell script* no és a PATH, hem d'escriure el nom del fitxer indicant on es troba (amb camí relatiu o camí absolut):

  ```
  # Executar un guió de shell des del directori actual

  ./nom_fitxer

  \\# Executar un guió de shell ubicat a /home/usuari

  /home/usuari/nom_fitxer
  ```

Variable PATH

La variable PATH conté la llista de directoris on el shell s'ha d'adreçar per localitzar les ordres que executem. Això ens permet escriure les ordres escrivint només el seu nom i sense especificar el directori on estan situades, és a dir, sense preocupar-nos d'on es troben al disc. La recerca de l'ordre sol·licitada en els directoris que s'especifiquen en la variable PATH es fa d'esquerra a dreta. Si no troba l'ordre en cap dels directoris especificats en la variable PATH, el shell avisa amb un missatge d'error que no s'ha trobat l'ordre. Podeu visualitzar el valor de la variable PATH amb l'ordre: `echo $PATH`.

Exemple de creació i execució d'un guió de shell

Obriu una sessió de terminal, assegureu-vos que esteu situats al vostre directori d'inici (~) executant l'ordre següent:

```
cd
```

Amb un editor de text pla, creeu un fitxer que contingui les línies següents:

```
#!/bin/bash
# holamon.sh
echo "Hola, món"
echo "Avui és $(date +%x)"
echo "Sóc $USER"
echo "Treballo amb el sistema $(uname -sr)"
echo "Adéu!"
```

Deseu el fitxer al mateix directori d'inici i anomeneu-lo holamon.sh. Doneu permís d'execució al fitxer:

```
chmod u+x holamon.sh
```

Executeu el shell script, feu la crida indicant el nom d'on es troba amb camí relatiu:

```
./holamon.sh
```

Executeu una altra vegada el shell script. Ara feu la crida indicant el nom d'on es troba el fitxer amb camí absolut:

```
~/holamon.sh
```

Creeu un directori ~/scripts per guardar els vostres programes, moveu el fitxer holamon.sh al directori ~/scripts i afegiu el directori al contingut de la variable PATH amb les ordres següents:

```
mkdir ~/scripts
mv ~/holamon.sh ~/scripts
export PATH="$PATH:~/scripts"
```

Ara podeu executar holamon.sh cridant-lo només pel seu nom, sense indicar on es troba (ni amb camí relatiu ni amb camí absolut), perquè està en un directori contingut a la variable PATH.

```
holamon.sh
```

Podeu provar de situar-vos a qualsevol punt de la jerarquia de directoris del sistema i executar holamon.sh posant només el seu nom. Per exemple:

```
cd /tmp
holamon.sh
```

El canvi que heu fet a la variable PATH només afecta a la sessió de shell que teniu activa. Si voleu que la variable PATH tingui el directori ~/scripts en totes les sessions que obriu, cal que editeu el fitxer ~/.bashrc, afegiu al final la línia amb l'ordre export i deseu els canvis.

```
export PATH="$PATH:~/scripts"
```

Un guió de *shell* també es pot executar explícitament amb un *shell* determinat, posant el nom del *shell* i a continuació el nom del fitxer que conté el guió de *shell*:

```
bash holamon.sh
```

Aquest mètode generalment només el fem servir quan volem comprovar que el programa funciona amb un altre *shell* o quan volem depurar (*debug*) el guió de *shell*. Per exemple:

```
rbash nom_script   # Executar el guió amb rbash
sh nom_script      # Executar el guió amb sh
bash -x nom_script # Executar el guió amb mode debug
```

Quan el Bash executa un *shell script*, crea un procés fill que executa un altre Bash, el qual llegeix les línies de l'arxiu (una línia per vegada), les interpreta i executa com si vinguessin del teclat. El procés Bash pare espera mentre el Bash fill executa l'script fins al final, i en aquell moment el control torna al procés pare, el qual torna a posar l'indicador o *prompt*.

Els canvis en l'entorn d'un *shell* fill no afecten a l'entorn del *shell* pare.

Per tant, si en el guió hi ha ordres que modifiquen l'entorn, tals com canviar de directori actual, modificar el valor d'una variable d'entorn, crear una nova variable, etc., aquests canvis només tenen efecte a l'entorn del *shell* fill i desapareixen una vegada finalitza l'execució del guió.

Vegeu el significat, la definició i utilització de les variables d'entorn i locals en l'apartat "Variables del *shell*" d'aquesta unitat.

Si volem executar un guió en el *shell* actual en lloc d'amb un *shell* fill i, per tant, modificar l'entorn actual, hem d'executar el guió amb l'ordre `source`.

```
source nom_script # Executar en el shell actual
```

L'ordre del Bash anomenada `source` és un sinònim de l'ordre . (punt) del *shell* Bourne:

```
. nom_script
```

No confongueu l'ordre punt (.) del *shell* Bourne amb el directori punt, que indica el directori actual.

Exemple d'execució d'un guió de shell amb o sense modificació de l'entorn actual

Creeu un guió de shell en el vostre directori de treball que contingui les línies següents:

```
#!/bin/bash
cd /etc
echo El directori actual és:
pwd
```

Deseu el fitxer i anomeneu-lo canvi.sh. Doneu permís d'execució al fitxer:

```
chmod u+x canvi.sh
```

Executeu el guió des del directori actual mitjançant un shell fill:

```
./canvi.sh
```

La sortida us mostra que en el shell fill es canvia de directori. Ara bé, quan finalitza l'execució del guió, el directori on estem situats continua sent el nostre directori de treball, perquè el canvi en el shell fill no ens afecta. Ho comprovem executant l'ordre:

```
1 pwd
```

Executeu el guió en el shell actual (enlloc d'en un shell fill) mitjançant l'ordre source o l'ordre punt (.):

```
1 source canvi.sh
```

En aquest cas, en finalitzar l'execució ens trobem al directori /etc, ja que les ordres del guió s'han executat en el nostre shell. Ho comprovem executant l'ordre:

```
1 pwd
```

2.1.3 Definició del shell d'execució

En escriure guions de *shell* és recomanable que indiquem el *shell* que ha d'executar el guió. Per això els dos primers caràcters de la primera línia han de ser *#!*, seguits del nom del *shell* que ha d'interpretar les ordres que venen a continuació.

Per exemple:

```
1 #!/bin/bash
```

No comenceu el *shell script* amb una línia en blanc, perquè aquestes línies també són tingudes en consideració.

Si ometem la definició del *shell* d'execució, el *shell script* s'executa amb el *shell* que hi ha establert per defecte i, en el cas de no ser el *shell* per al qual s'ha escrit el guió, les ordres contingudes poden donar error en ser executades.

2.1.4 Comentaris al guió de shell

Els comentaris són útils per ajudar a entendre els scripts als lectors. Cal tenir en compte que probablement no serem els únics que llegirem el codi dels guions de *shell* que fem, o que, passat un temps, la memòria ens pot fallar i no entendre el programa que nosaltres mateixos hem escrit.

Scripts d'inici

Vegeu els scripts d'inici del sistema al directori /etc/init.d. Aquests guions estan molt ben comentats perquè siguin fàcilment llegibles pels administradors i els programadors del sistema.

Normalment, les primeres línies després de la línia que indica el *shell* d'execució són un comentari sobre la funcionalitat del programa. La resta del programa es comenta com calgui per a una major claredat i comprensió del codi.

Per afegir comentaris al programa només cal posar el símbol # seguit del text que es vulgui. Cada nova línia ha d'anar precedida del símbol #. Es pot posar el símbol

\ al final d'una línia per indicar que el text no ha acabat i continua a la línia següent. Per exemple, les primeres línies del guió de *shell* /etc/rc.init/kerneloops són:

```
#!/bin/bash
#
# kerneloops
#
# chkconfig: 345 90 88
# description: A tool that collects and submits \
# kernel crash signatures to the kerneloops.org \
# website for use by the Linux kernel developers.
# ...
```

2.1.5 Tabulació del codi

En el cas més simple, un guió no és més que una llista d'ordres del sistema escrites una darrere de l'altra que s'executen de manera seqüencial. Ara bé, en programar *shell scripts* sovint utilitzem estructures de control de flux (condicionals i iteratives) que ens permeten trencar el flux d'execució, repetir una part de les ordres, etc., i que fan que el guió de *shell* no sigui simplement una enumeració d'ordres.

Per tal de facilitar la lectura del codi del guió de *shell*, és molt important que el codi estigui ben tabulat. Al Bash no li cal que hi hagi tabuladors per poder interpretar el codi, però a les persones sí que ens va bé que el codi estigui organitzat amb tabuladors per tal de llegir-lo d'una manera còmoda.

> El codi d'un *shell script* ha d'estar ben tabulat per tal que el programa sigui llegible.

Per exemple, considerem el guió de *shell* següent, escrit sense tabular:

```
#!/bin/bash
echo -n "Escriu un número: "
read X
if [ $X -lt 10 ]; then
echo "X és més petit que 10"
else
if [ $X -gt 10 ]; then
echo "X és més gran que 10"
else
echo "X és igual a 10"
fi
fi
```

La lectura i la comprensió de les estructures condicionals imbricades del guió de *shell* anterior són difícils. El mateix codi, tabulat i comentat, queda així:

```
#!/bin/bash
#
# esdeu.sh
# Demana un número i comprova si és igual a 10.
#
echo -n "Escriu un número: "
read X
```

```
if [ $X -lt 10 ]; then
  echo "X és més petit que 10"
else
  if [ $X -gt 10 ]; then
    echo "X és més gran que 10"
  else
    echo "X és igual a 10"
  fi
fi
```

La tabulació ens permet llegir el codi i seguir la lògica del programa amb més comoditat, així com evitar errades de programació (oblidar una paraula clau de tancament d'una estructura de control, situar una paraula clau en algun lloc indegut, etc.).

No hi ha cap norma que fixi com s'ha de tabular el codi, hi ha qui utilitza tabuladors de dos espais i qui prefereix quatre espais. Tant és, el que importa és utilitzar un estil homogeni al llarg de tot el *shell script*.

Si usem un editor de text pla, hem de sagnar el codi a mà, o bé amb espais o bé amb tabuladors. Alguns editors avançats ens faciliten aquesta feina fent que a mida que escrivim el codi es vagi sagnant de manera automàtica sense que nosaltres ens haguem de preocupar de les tabulacions.

2.1.6 Depurar un guió de shell

De vegades el funcionament d'un guió de *shell* no és l'esperat i hem de determinar quina és la causa d'aquest funcionament incorrecte, és a dir, hem de depurar el *shell script*. Una tècnica consisteix a fer un seguiment pas a pas de les ordres que executa per tal de veure on es produeix l'error. Per fer aquest seguiment podem executar l'*script* amb *bash-x* seguit del nom del guió així:

```
bash -x nom_fitxer
```

O bé podem incloure *-x* a la primera línia:

```
#!/bin/bash -x
```

Si només volem depurar una part del programa, podem fer-ho de la manera següent:

```
...
# activar depuració des d'aquí
set -x

codi per depurar

# parar la depuració
set +x
...
```

L'opció *-x* de Bash indica que s'han d'imprimir per pantalla les ordres i els seus arguments mentre s'executen. Així podem veure a quin punt de l'execució s'ha

arribat quan es produeix un error.

Exemple d'execució pas a pas d'un guió de shell

Feu un guió de shell que es digui prova.sh amb les línies següents:

```
#!/bin/bash
# Shell script de prova
N=5
echo "El valor de N és: $N"
```

Executeu-lo de manera normal i veieu que la sortida és simplement el missatge següent:

```
El valor de N és:  5
```

Ara l'executem mostrant cada pas de l'execució de la manera següent:

```
bash –x prova.sh
```

El resultat és que ens apareixen les línies que es van executant amb un símbol + al davant:

```
+ N=5
+ echo 'El valor de N és:  5'
El valor de N és:  5
```

Podem afegir l'opció -v per mostrar totes les línies d'entrada al shell, incloent comentaris, a mesura que les va llegint:

```
bash –xv prova.sh
```

El resultat és:

```
#!/bin/bash
# Shell script de prova
N=5
+ N=5
echo ''El valor de N és:  $N''
+ echo 'El valor de N és:  5'
El valor de N és:  5
```

2.2 Interacció amb l'usuari

Un guió de *shell* pot requerir interacció amb l'usuari, és a dir, sol·licitar-li l'entrada de dades o mostrar-li dades de sortida. Per poder interaccionar amb el programa normalment utilitzem les ordres ***echo*** i ***read***, que ens permeten mostrar dades a la pantalla del terminal i llegir dades del teclat en mode text.

Si el que volem és fer una aplicació gràfica, aleshores Bash no és l'eina indicada. Tanmateix, podem tenir una interacció bàsica amb mode gràfic i utilitzar caixes de diàleg senzilles amb algun programari que ho permeti, com ara el Zenity.

2.2.1 Ordres echo i read

L'ordre ***echo*** mostra una cadena de text afegint un salt de línia per la sortida estàndard. La sintaxi de l'ordre és:

```
1 echo [opció] [cadena]
```

Executeu `man echo` per consultar la llista de caràcters d'escapament possibles del vostre sistema.

Les opcions que es poden utilitzar amb *echo* són:

- *-n* perquè no afegeixi un salt de línia després de mostrar la cadena.
- *-e* per habilitar la interpretació dels caràcters d'escapament (\n per afegir un salt de línia, \t per afegir un tabulador, etc.). Si utilitzem aquesta opció cal que posem el text de la cadena entre cometes dobles.

Per exemple:

```
1 echo –e "\n\nHola, món!!\n\"n
```

L'ordre ***read*** permet llegir dades de l'entrada estàndard. La utilització més habitual de l'ordre *read* és amb la sintaxi següent:

```
1 read nom_variable
```

L'ordre *read* llegeix el que l'usuari introdueix pel teclat fins que hi ha un salt de línia i assigna les dades a la variable *nom_variable*. Per exemple:

Executeu `man read` per veure la sintaxi completa d'aquesta ordre.

```
1 read N
```

Exemple de guió de shell que interacciona amb l'usuari amb les ordres echo i read.

Feu un guió de shell que s'anomeni salutacio.sh que demani un nom i a continuació mostri un missatge de salutació amb el nom que hem introduït.

```
1 #!/bin/bash
2 #
3 # salutacio.sh
4 #
5 # Exemple d'ús de les ordres echo i read
6 #
7 echo "Com et dius?"
8 read NOM
9 echo "Hola, $NOM"
```

Eines GTK

Les GTK o grup d'eines del GIMP (GIMP toolkit, en anglès) són unes llibreries pensades per al desenvolupament d'aplicacions gràfiques amb facilitat. Disposen de llicència GPL i hi ha nombrosos projectes que les utilitzen, com el GIMP (d'aquí li prové el nom, ja que inicialment es van crear per a aquest projecte), el GNOME, l'Eclipse o el Firefox, entre d'altres.

2.2.2 Interacció en mode gràfic

El *shell* Bash és un *shell* de línia d'ordres pensat per interactuar en mode text, però podem tenir una interacció bàsica en mode gràfic utilitzant altres programes, com

ara Zenity. Zenity permet utilitzar caixes de diàleg basades en GTK+ a la línia d'ordres i en els *shell scripts*.

Podem usar Zenity per crear diàlegs simples que interactuïn gràficament amb l'usuari, ja sigui per obtenir informació de l'usuari o per proporcionar-li informació. Per exemple, podem demanar a l'usuari que seleccioni una data d'un diàleg del calendari o que seleccioni un arxiu d'un diàleg de selecció d'arxiu. O es pot usar un diàleg de progrés per indicar l'estat actual d'una operació o usar un diàleg d'alerta per notificar a l'usuari algun error.

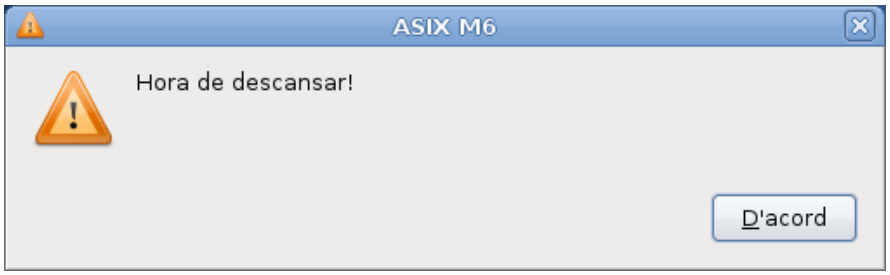
Per exemple, obriu una sessió de terminal a l'escriptori GNOME i executeu la línia d'ordres següent:

Vegeu la documentació oficial de Zenity a la secció del web "Adreces d'interès".

```
1 /usr/bin/zenity --warning --title="ASIX M6" --text="Hora de descansar\!" --
      width=500
```

El resultat és una caixa de diàleg com la de la figura 2.2.

FIGURA 2.2. Caixa de diàleg de Zenity



2.3 Paràmetres i variables especials

La utilització de **paràmetres** és un mètode per passar dades al programa de manera no interactiva. El llenguatge del *shell* disposa d'una serie de variables especials que permeten treballar amb aquests paràmetres.

2.3.1 Ús de paràmetres

Un **paràmetre** o **argument** de la línia d'ordres no és més que un valor que li passem al programa en el moment de la seva crida. Un programa pot tenir qualsevol nombre d'arguments a la línia d'ordres.

La majoria d'ordres d'Unix poden fer accions diferents en funció dels paràmetres que els donem en executar-les. Per exemple, escriviu l'ordre:

```
1 ls /etc/profile
```

`ls` és el nom de l'ordre que s'ha d'executar. Tot el que ve a continuació a la línia d'ordres es pren com a arguments per a aquesta ordre. Cada un dels arguments està separat per un o més espais. En aquest cas hi ha un únic argument, el nom del fitxer /etc/profile.

Considerem un exemple de crida d'una ordre amb dos arguments:

```
1 tail +10 /var/log/messages
```

`tail` és el nom de l'ordre, *+10* és el primer argument i el nom del fitxer /var/log/messages és el segon argument.

A la taula 2.1 podeu veure més exemples de crida d'ordres amb arguments.

TAULA 2.1. Exemples d'execució d'ordres i paràmetres

Ordre	Nom de l'ordre	Nombre d'arguments	Nom dels arguments
ls	ls	0	N/A
ls /etc/resolv.conf	ls	1	/etc/resolv.conf
cp /etc/resolv.conf /tmp/test.txt	cp	2	/etc/resolv.conf /tmp/test.txt
sort -r -n nom_fitxer	sort	3	-r -n nom_fitxer
date +"%d-%m-%Y"	date	1	+"%d-%m-%Y"

De la mateixa manera que ho fan la majoria de les ordres, els guions de *shell* poden acceptar paràmetres quan s'executen. Un paràmetre o argument és un valor que li donem al guió de *shell* en el moment de la seva crida. Els arguments d'un *shell script* es poden referenciar dins del programa mitjançant una sèrie de **variables especials**.

2.3.2 Variables especials

En executar un guió de *shell* amb paràmetres, hi ha un conjunt de variables especials del *shell* anomenat **paràmetres posicionals** que es fixen automàticament perquè coincideixin amb els paràmetres donats al programa. S'anomenen *paràmetres posicionals* perquè l'assignació de cada variable depèn de la posició d'un paràmetre en la línia d'ordres. Els noms d'aquestes variables es corresponen amb el valor numèric de la seva situació en la línia d'ordres: 0, 1, 2, 3... I així fins a l'últim paràmetre que es passa.

Els paràmetres posicionals es poden utilitzar dins del guió de *shell* com qualsevol altra variable del *shell*, és a dir, per saber el seu valor utilitzarem el símbol *$*. A partir del desè paràmetre, el nombre s'ha de tancar entre claus.

Els paràmetres dins del programa són accessibles utilitzant les variables: `$0`, `$1`, `$2`, `$3`... `${10}`, `${11}`, `${12}`...

Exemple d'ús de paràmetres en un shell script

Creeu un shell script que es digui args.sh amb el contingut següent:

```
#!/bin/bash
#
# args.sh
#
# Exemple d'ús de paràmetres
#
echo "Nom del guió de shell: $0"
echo "Valor del primer paràmetre del guió de shell: $1"
echo "Valor del segon paràmetre del guió de shell: $2"
echo "Valor del tercer paràmetre del guió de shell: $3"
```

Deseu el fitxer. Executeu-lo així:

```
chmod +x args.sh
./args.sh blau verd vermell
```

La sortida del programa és la següent:

```
Nom del guió de shell:  ./args.sh

Valor del primer paràmetre del guió de shell:  blau

Valor del segon paràmetre del guió de shell:  verd

Valor del tercer paràmetre del guió de shell:  vermell
```

Podeu provar diferents execucions del programa que acabeu de fer i comprovar-ne els resultats, per exemple:

```
./args.sh A B C
./args.sh Alba 32 Barcelona
```

A banda dels paràmetres posicionals, hi ha unes altres variables especials definides en qualsevol script que podem usar segons les nostres necessitats. La taula 2.2 mostra el nom i la descripció de les variables especials més utilitzades.

Taula 2.2. Variables especials

Variable	Descripció
`$0`	Nom del *shell script* que s'està executant
`$n`	Paràmetre passat al *shell script* en la posició *n*
`$*`	Cadena que conté tots els paràmetres rebuts, començant per *$1*
`$@`	Igual que *$**, excepte quan es posa entre cometes
`$#`	Nombre de paràmetres
`$$`	PID del procés del *shell* que s'està executant
`$!`	PID de l'últim procés executat
`$?`	Codi de sortida de la darrera ordre executada

Vegeu el significat de la variable especial *$?* a l'apartat "Codis de sortida" d'aquesta unitat.

Tingueu en compte les observacions següents:

- *$** i *$@* són el mateix quan no van entre cometes.
- "*$**" expandeix els paràmetres en una cadena: "par1 par2…". És una

sola cadena que comprèn tots els paràmetres units per espais en blanc. Per exemple '1 2' 3 esdevé "1 2 3".

- *"$@"* expandeix els paràmetres en cadenes diferenciades: "par1" "par2"... És a dir, la llista de cadenes resultant coincideix exactament amb allò que s'ha donat al guió de *shell*. Per exemple, *'1 2' 3* esdevé "1 2" "3".

El valor de les variables especials es pot guardar en altres variables, per exemple:

```
NOM=$1
```

Ara bé, l'assignació de valors a les variables especials no està permesa, per exemple:

```
# operació no permesa
$1=Alba
```

Exemple de visualització dels paràmetres i d'altres variables especials.

Modifiqueu el guió de shell args.sh i afegiu-li al final les línies que apareixen en negreta:

```
#!/bin/bash
#
# args.sh
#
# Exemple d'ús de paràmetres
#
echo "Nom del guió de shell: $0"
echo "Valor del primer paràmetre del guió de shell: $1"
echo "Valor del segon paràmetre del guió de shell: $2"
echo "Valor del tercer paràmetre del guió de shell: $3"
echo "Nombre de paràmetres passats al guió de shell: $#"
echo "Llista de tots els arguments rebuts: $*"
```

Executeu-lo:

```
./args.sh a b c
```

La sortida del programa és la següent:

```
Nom del guió de shell: ./cmdargs.sh
Valor del primer paràmetre del guió de shell: a
Valor del segon paràmetre del guió de shell: b
Valor del tercer paràmetre del guió de shell: c
Nombre de paràmetres passats al guió de shell: 3
Llista de tots els arguments rebuts: a b c
```

2.3.3 Control del nombre de paràmetres

La majoria d'ordres mostren un missatge d'error o informatiu quan els arguments requerits per l'ordre no s'han especificat en la seva crida. Per exemple, executeu l'ordre:

```
rm
```

La sortida que us dóna és:

```
rm: missing operand
Try 'rm --help' for more information.
```

Anàlogament, si un guió de *shell* espera rebre paràmetres, hem de verificar dins del programa que la crida al guió de *shell* s'ha fet amb el nombre de paràmetres esperat i, en cas contrari, mostrar un missatge d'error. Aquest control el podem dur a terme mitjançant l'estructura condicional if i la variable especial $#.

Vegeu el funcionament de l'estructura if en l'apartat "Estructures condicionals" d'aquesta unitat.

Exemple d'un shell script amb control de nombre de paràmetres

Creeu un shell script amb el codi següent i anomeneu-lo suma.sh. Aquest script rep dos números per paràmetre i mostra per pantalla el resultat de la suma dels dos números. A l'inici controla que el nombre d'arguments rebuts és correcte i dóna un error si no és així.

```
#!/bin/bash
#
# suma.sh
#
# Rep dos nombres per paràmetre i mostra la suma per
# la sortida estàndard.
#
# Control del nombre de paràmetres:
if [ $# -ne 2 ]; then
echo "Error: s'esperaven dos paràmetres."
echo "Ús del programa: $0 num1 num2"
exit 1
fi
# Rebuts dos paràmetres, fem la suma
(( SUMA = $1 + $2 ))
echo "La suma de $1 i $2 és: $SUMA"
```

Doneu permís d'execució al fitxer i executeu-lo passant-li dos números com a paràmetres:

```
chmod +x suma.sh
./suma.sh 10 5
```

La sortida del programa és la següent:

```
15
```

Executeu-lo passant-li un nombre erroni de paràmetres (diferent de 2), per exemple:

```
./suma.sh
```

La sortida del programa és la següent:

```
Error: s'esperaven dos paràmetres.
Ús del programa: suma.sh num1 num2
```

2.4 Codis de sortida

En finalitzar l'execució totes les ordres generen un **codi de sortida** (en anglès, *exit status* o *exit code*) que és un nombre sencer entre 0 i 255.

Per convenció, si l'ordre acaba bé, normalment retorna un zero (0) i si acaba malament retorna un valor diferent de zero (entre 1 i 255). Moltes vegades, el valor retornat per l'ordre representa l'error generat. Per exemple, els errors de sintaxi gairebé sempre fan que les ordres retornin el valor 1.

El *shell* ens proporciona una variable especial anomenada *?* (signe d'interrogació), que conté el codi de sortida de l'ordre executada anteriorment.

El codi de sortida pot ser utilitzat pel Bash, el podem mostrar o podem controlar el flux del guió amb ell. Per visualitzar el valor utilitzem l'ordre *echo*:

```
1 echo $?
```

En els guions de *shell*, la majoria de les decisions de programació es controlen analitzant el valor dels codis de sortida. Quan avaluem condicions, el codi de sortida ens indica si la condició és **vertadera** (retorna 0) o **falsa** (retorna un valor diferent de zero).

Exemple d'execució d'ordres i visualització dels codis de sortida

Obriu una sessió de terminal i executeu l'ordre ls:

```
1 ls
```

Visualitzeu el codi de sortida de l'ordre que acabeu d'executar:

```
1 echo $?
2 0
```

La sortida és un zero, fet que indica que no hi ha hagut cap error. Executeu l'ordre ls amb algun nom de fitxer inexistent per provocar una errada:

```
1 ls asdfg
2 ls: no s'ha pogut accedir a asdfg: El fitxer o directori no
      existeix
```

Visualitzeu el codi de sortida de l'ordre que acabeu d'executar:

```
1 echo $?
2 2
```

La sortida és un valor diferent de zero (un 2), fet que indica que hi ha hagut un error.

Executeu l'ordre cp sense arguments per provocar una errada:

```
cp
cp: manca un operand fitxer
Proveu «cp ——»help per a obtenir més informació.
```

Visualitzeu el codi de sortida de l'ordre que acabeu d'executar:

```
echo $?
1
```

La sortida és un 1, fet que indica error de sintaxi.

De la mateixa manera que ho fan les ordres, els *shell scripts* retornen un codi de sortida que és el de la darrera ordre executada. El codi de sortida d'un *shell script* també es pot consultar just després de la finalització del programa mitjançant la variable *$?*.

Exemple de visualització del codi de sortida d'un guió de shell

Creeu un guió de shell senzill anomenat hola.sh:

```
#!/bin/bash
echo "Hola, món"
```

Doneu permís d'execució al fitxer i executeu el guió:

```
chmod u+x hola.sh
./hola.sh
```

Consulteu el codi de sortida des de la línia d'ordres:

```
echo $?
0
```

El codi retornat és un zero, el de la darrera ordre executada en el guió de *shell*, és a dir, l'ordre `echo ‘‘Hola, món’’`.

Podeu modificar el guió i afegir al final una ordre que doni error per comprovar l'efecte que té sobre el codi de sortida.

2.4.1 Ordre exit

L'ordre **exit** provoca la finalització del *shell script* i de manera opcional permet fixar el codi de sortida amb un valor determinat. La seva sintaxi és:

```
exit [ n ]
```

essent *n* un nombre enter entre 0 i 255.

Per convenció el valor que es retorna és un zero si ha anat bé o un enter del rang entre 1 i 255 si hi ha hagut algun error. Normalment, el rang de 126 a 255 es reserva per ser utilitzat directament pel *shell* o per a finalitats especials, i els codis del rang de 0 a 125 es deixen per ser utilitzats pel programa.

En tot guió de *shell* convé proporcionar un codi de sortida significatiu, com a mínim un **0 (zero) si finalitza bé** i un **1 (en general, un valor diferent de zero) si hi ha algun error**. Això permetrà que el procés que ha fet la crida pugui verificar com ha anat l'execució del guió de *shell*.

Si no es passa cap paràmetre a *exit*, el codi retornat pel guió de *shell* és el de la darrera ordre executada just abans de l'*exit*. Per exemple:

```
#!/bin/bash
ordre_1
.. .
ordre_N
# Acaba amb el codi de sortida de l'ordre_N.
exit
```

L'exemple anterior, hauria estat exactament igual si s'hagués especificat el valor del codi de retorn de la manera següent:

```
#!/bin/bash
ordre_1
.. .
ordre_N
# Acaba amb el codi de sortida de l'ordre_N.
exit $?
```

Anàlogament, si un *shell script* finalitza sense especificar l'ordre *exit*, el codi de retorn és el de la darrera ordre executada al *shell script*.

```
#!/bin/bash
ordre_1
.. .
ordre_N
# Acaba amb el codi de sortida de l'ordre_N.
```

Exemple d'utilització de l'ordre exit i comprovació dels codis de sortida.

Feu un guió de shell que s'anomeni sortida.sh amb les línies següents:

```
#!/bin/bash

echo "Això és una prova."
# Retornem codi de sortida 0
exit 115
```

Deseu el fitxer i executeu-lo:

```
chmod +x sortida.sh
./sortida.sh
```

La sortida del programa és la següent:

```
Això és una prova.
```

Visualitzeu el codi de sortida del guió de *shell*:

```
echo $?
```

El valor de sortida és un 115, atès que l'hem forçat amb l'ordre *exit*.

```
115
```

En la programació de guions de *shell* és molt freqüent utilitzar l'ordre *exit* per finalitzar el programa en qualsevol punt sense seguir la norma de la programació estructurada que diu que un programa ha de tenir un únic punt de sortida. Per exemple, un *shell script* fet amb programació estructurada seria així:

```
if condicio_error1; then
  codi=1
else
  if condicio_error2; then
    codi=2
  else
    if condicio_error3; then
      codi=3
    else
      # No hi ha errors
      instruccions
      codi=0
    fi
  fi
fi
exit $codi
```

Però habitualment ens trobarem el guió de *shell* anterior escrit amb un estil similar al següent:

```
if condicio_error1; then
  exit 1
fi
if condicio_error2; then
  exit 2
fi
if condicio_error3; then
  exit 3
fi
# No hi ha errors
instruccions
exit 0
```

Tot i que els dos programes fan el mateix, el primer segueix les normes de la programació estructurada i té un únic punt de sortida en el darrer *exit*, mentre que el segon té quatre possibles punts de sortida, un per a cada ordre d'*exit*.

2.5 Avaluació aritmètica i lògica

En programar gairebé sempre apareix la necessitat d'operar amb nombres així com d'avaluar el resultat d'expressions aritmètiques o lògiques per prendre decisions.

A Bash disposem del mecanisme d'**expansió aritmètica** per operar i avaluar expressions amb nombres enters però també tenim altres mètodes, dels quals en veurem dos: **l'ordre** `let` **i la construcció doble parèntesi**. Per aquells casos que necessitem operar amb nombres amb decimals veurem com fer-ho utilitzant l'ordre `bc`.

Vegeu el mecanisme d'expansió aritmètica descrit a l'apartat "Expansió aritmètica" d'aquesta unitat.

Una altra ordre molt utilitzada en la programació de guions de *shell* per fer avaluacions lògiques és l'**ordre** `test`. Aquesta ordre ens permet avaluar expressions amb tres tipus d'elements: nombres enters, fitxers i cadenes de caràcters.

2.5.1 Ordre let

L'ordre `let` permet als programes de *shell* avaluar expressions aritmètiques amb els mateixos operadors que en el mecanisme d'expansió aritmètica i amb la sintaxi següent:

```
let expressió_aritmètica
```

L'ordre avalua d'esquerra a dreta l'expressió i torna un codi de sortida igual a 0 (vertader) o 1 (fals). Malgrat que no sempre cal tancar l'expressió entre cometes dobles, podem optar per posar-les per defecte per evitar errades.

L'expressió aritmètica pot constar de nombres enters, variables i operadors de *shell*. També podem utilitzar parèntesis per canviar l'ordre de preferència d'una expressió. Per exemple:

```
#!/bin/bash
# Definició de variables
x=12
y=2
# Utilització de l'ordre let
let "z = x / y + 1"
echo $z# z val 7, primer s'ha fet la divisió
# Alterar precedència amb parèntesis
let "z = x / (y + 1)"
echo $z# z val 4, primer s'ha fet la suma
```

Quan en l'expressió utilitzem operadors lògics o relacionals (*!*, <=, >=, <, >, ==, *!*=) avaluem el codi de sortida tornat pel *shell*. Per exemple:

```
let "11 < 10"
echo $?
```

L'ordre anterior retorna un 1 indicant que l'expressió s'ha avaluat com a falsa, ja que 11 no és més petit que 10.

Algunes consideracions a tenir en compte quan utilitzem l'ordre *let* són:

- L'ordre *let* no mostra cap resultat per pantalla. Un error comú és esperar que *let* retorni el resultat d'una operació per la sortida estàndard, per exemple, executar `let 4+2` i esperar un 6 per pantalla. Per operar amb *let* cal utilitzar variables i assignar el resultat a les variables, per exemple, `let x=4+2`.
- L'expressió de *let* s'escriu sense espais. Si voleu posar espais per separar els operands dels operadors perquè quedi més clar, heu de posar l'expressió entre cometes dobles. Per exemple, let ‘‘`x = x + 2`’’.
- Si en una expressió voleu utilitzar parèntesis per alterar l'ordre de precedència dels operadors, heu de posar doble cometa per anul·lar el significat especial dels parèntesis. Per exemple, `let ‘‘x=x/(y+1)’’`.
- Les variables que s'utilitzen dins d'una expressió poden anar referenciades o no. Per exemple, let x=x+2 és el mateix que `let x=$x+2`.

2.5.2 La construcció doble parèntesi

La construcció doble parèntesi, *(( ))*, permet avaluar expressions aritmètiques de manera equivalent a l'ordre *let* però amb la sintaxi següent:

```
(( expressió_aritmètica ))
```

Per exemple:

```
(( x = x + (y / 2) ))
```

És equivalent a:

```
let "x = x + (y / 2)"
```

Sovint trobem la construcció doble parèntesi formant part de bucles *while* amb l'estil del llenguatge de programació C. Per exemple:

Alguns autors recomanen utilitzar preferentment els dobles parèntesis enlloc de l'ordre *let*.

```
x=1
while (( x < 10 )); do
  ...
  ordres
  ...
  (( x++ ))
done
```

Igual que amb l'ordre *let*, les variables utilitzades com a operands en una expressió poden anar precedides del símbol *$* o no. Per exemple, les dues expressions següents són correctes:

Vegeu l'explicació dels bucles *while* utilitzant la construcció *(( ))* a l'apartat "Estructures repetitives" d'aquesta unitat.

```
(( x = x + 1 ))   # Correcte
(( x = $x + 1))   # Correcte
```

Però aneu amb compte i no cometeu errades com aquesta:

```
(( $x = x + 1 ))  # Incorrecte!!
```

2.5.3 Operacions amb nombres amb decimals

L'ordre *bc* és un programa molt potent que incorpora un llenguatge de programació propi i que permet fer càlculs amb precisió. Aquesta ordre ens pot ser de molta utilitat si necessitem fer operacions amb nombres amb decimals.

Si executem *bc* a la línia d'ordres, ens apareix una informació sobre la versió del programa similar a la següent:

Executeu *man bc* per veure'n el funcionament i la descripció de totes les opcions.

```
bc 1.06.95
Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006 Free Software Foundation, Inc
    .
This is free software with ABSOLUTELY NO WARRANTY.
For details type 'warranty'.
```

El programa queda esperant l'entrada de dades de manera interactiva des del teclat, de manera que podem introduir les operacions que volem fer, prémer *Retorn* i a continuació obtenim el resultat, per exemple:

```
6*3/2
9
```

Per sortir de la calculadora hem d'escriure la paraula *quit* seguida de *Retorn*.

Podem utilitzar *bc* dins d'un *shell script* mitjançant una canonada per redirigir les dades d'entrada al programa. Per exemple:

```
echo "(2 + 3) * 5" | bc
```

En l'exemple anterior utilitzem l'ordre *echo* per passar les dades d'entrada a l'ordre *bc*. El resultat de l'operació ens surt per pantalla, en aquest cas un 25. Per treballar amb nombres amb decimals especifiquem la precisió (quantitat de decimals) amb l'opció *scale*. Per exemple:

```
echo "scale=2; 7*5/3" | bc
```

Nombre π

En matemàtiques, π o pi és la constant d'Arquimedes, una constant que relaciona el diàmetre de la circumferència amb la longitud del seu perímetre. És un nombre irracional, és a dir, la seva part fraccionària té un nombre de xifres infinit i a més no té cap període. Per fer càlculs pràctics s'agafa un valor simplificat de pi, com per exemple 3,14159265.

L'ordre anterior ens mostra un 11,66 per pantalla, que és el resultat de l'operació que hem realitzat.

Veiem un darrer exemple en què mostrem el nombre pi (π) per pantalla utilitzant l'opció *-l* en la crida de *bc* per poder usar funcions matemàtiques, en aquest cas la funció *a(x)*, que ens retorna l'arc tangent d'*x* en radiants:

```
echo "scale=9; 4*a(1)" | bc -l
```

2.5.4 Ordre test

Vegeu el significat de la variable especial *?* a l'apartat "Codis de sortida" d'aquesta unitat.

L'ordre `test` avalua expressions lògiques i genera un codi de sortida que indica si l'expressió és **certa** (codi de sortida igual a zero) o **falsa** (codi de sortida diferent de zero). Aquesta ordre no escriu res a la sortida estàndard. Per determinar el resultat de l'ordre *test* cal avaluar el valor del codi de sortida amb la variable *?*.

La sintaxi de l'ordre test és la següent:

```
test expressió_test
```

O en forma abreujada:

```
[ expressió_test ]
```

Cal deixar un espai després del símbol *[* i abans del símbol *]*.

Quan utilitzeu la forma abreujada de *test* (l'expressió entre els claudàtors) cal que aneu amb molt de compte i no oblideu que hi ha espais entre l'expressió i els claudàtors. Si no poseu els espais es produirà un error de sintaxi.

A la taula 2.3 podeu veure les expressions que es poden avaluar amb l'odre *test*.

Taula 2.3. Avaluació d'expressions amb l'ordre test

Avaluació d'expressions amb nombres entersable	
[num1 -eq num2]	cert si num1 i num2 són iguals
[num1 -ne num2]	cert si num1 i num2 són diferents
[num1 -gt num2]	cert si num1 és més gran que num2
[num1 -ge num2]	cert si num1 és més gran o igual que num2
[num1 -lt num2]	cert si num1 és més petit que num2
[num1 -le num2]	cert si num1 és més petit o igual que num2
Avaluació d'expressions amb cadenes de caràcters	
[cad]	cert si és una cadena no buida
[-n cad]	cert si és una cadena no buida
[-z cad]	cert si és una cadena buida
[cad1 = cad2]	cert si cadena1 i cadena2 són iguals
[cad1 != cad2]	cert si cadena1 i cadena2 són diferents
Avaluació d'expressions amb fitxers	
[-e fit]	cert si el fitxer existeix
[-d fit]	cert si el fitxer existeix i és un directori
[-f fit]	cert si el fitxer existeix i és regular
[-L fit]	cert si el fitxer existeix i és un enllaç simbòlic
[-r fit]	cert si el fitxer existeix i té permís de lectura
[-w fit]	cert si el fitxer existeix i té permís d'escriptura
[-x fit]	cert si el fitxer existeix i té permís d'execució
[fit1 -nt fit2]	cert si fitxer1 és més nou que fitxer2
[fit1 -ot fit2]	cert si fitxer1 és més antic que fitxer2
Operadors lògics	
[exp1 -a exp2]	AND. Cert si l'expressió1 i l'expressió2 són certes.
[exp1 -o exp2]	OR. Cert si l'expressió1 o l'expressió2 són certes.
[! exp]	Negació. Cert si l'expressió és falsa.

Exemple d'avaluació de diferents tipus d'expressions amb test

Amb l'ordre test, comproveu si el nombre 11 és més gran que el nombre 15.

```
1 test 11 -gt 15
```

Avalueu el codi de sortida de l'ordre que acabeu d'executar per saber si l'expressió anterior és vertadera (0) o falsa (diferent de 0):

```
1 echo $?
```

```
1
```

La sortida és un 1, fet que indica que l'expressió "11 és més gran que 15" és falsa.

Amb la forma abreviada de l'ordre test, comproveu si el número 15 és igual que el número 15.

```
[ 15 -eq 15 ]
```

Avalueu el codi de sortida de l'ordre que acabeu d'executar per saber si l'expressió anterior és vertadera (0) o falsa (diferent de 0):

```
echo $?
0
```

La sortida és un 0, fet que indica que l'expressió "15 és igual que 15" és certa.

Amb la forma abreviada de *test*, comproveu si la cadena "hola" és igual a la cadena "HOLA":

```
[ "hola" = "HOLA" ]
```

Avalueu el codi de sortida per saber si l'expressió anterior és vertadera (cadenes iguals) o falsa (cadenes diferents):

```
echo $?
1
```

La sortida és un 1, fet que indica que l'expressió és falsa, ja que "hola" en minúscules no és igual a "HOLA" en majúscules.

Amb l'ordre test, comproveu si existeix un fitxer anomenat /etc/passwd:

```
test -f /etc/passwd
```

Avalueu el codi de sortida de l'ordre que acabeu d'executar per saber si l'expressió anterior és vertadera (el fitxer existeix) o falsa (el fitxer no existeix):

```
echo $?
0
```

La sortida és un zero, fet que indica que l'expressió és vertadera (el fitxer existeix).

Assigneu a una variable N el valor 15, a continuació amb l'ordre test comproveu si el valor d'N està entre 10 i 20:

```
N=15
[ $N -gt 10 -a $N -lt 20 ]
echo $?
0
```

El codi de sortida és un zero, fet que indica que l'expressió és vertadera.

Escriviu amb la forma abreviada de *test* una comprovació que doni cert si el directori /tmp *no* existeix:

```
[ ! -d /tmp ]
echo $?
1
```

El codi de sortida és un 1, fet que indica que l'expressió és falsa, atès que el directori /tmp sí que existeix.

Quan avaluem el valor d'una variable en una expressió, hem d'assegurar-nos que la variable sempre conté algun valor, perquè si no, la variable valdrà *null* i l'script ens donarà un error. Per exemple:

```
[ $X -eq 3 ]
bash: [: -eq: s'esperava un operador unari
```

La variable *X* no té cap valor, per tant el *shell* ha interpretat *[-eq 3]* i ha donat un error.

Si treballem amb cadenes de caràcters, podem evitar aquesta errada posant sempre les variables entre cometes dobles (“”). Així ens assegurem que la variable conté almenys el valor *null* i el shell interpretarà la cadena com a buida. Per exemple:

```
[ "$X" = 3 ]
```

En aquest cas, si *X* no té cap valor, el *shell* interpreta *[”” = 3]* i no dóna error.

> El *shell* interpreta els valors d'una expressió de *test* com a enters o com a cadenes de caràcters segons els operadors que utilitzem.

Les expressions que s'utilitzen amb l'ordre *test* poden ser connectades lògicament utilitzant els operadors propis de *test -a* i *-o*, però la manera més recomanable de fer-ho és utilitzar les ordres de *test* de manera independent i combinar-les amb els operadors lògics *&&* (AND lògic) i || (OR lògic). Per exemple, en lloc d'escriure

```
[ $N -gt 10 -a $N -lt 20 ]
```

és millor separar les expressions i connectar-les així:

```
[ $N -gt 10 ] && [ $N -lt 20 ]
```

Però aneu amb compte; els operadors lògics no poden anar dins dels claudàtors de *test*:

```
# Operació errònia
[ $N -gt 10 && $N -lt 20 ]
```

2.6 Estructures de control

Les estructures de control (*if*, *case*, *while*, *for*, etc.) permeten canviar el flux seqüencial d'un programa en funció de l'avaluació d'unes condicions i bifurcar-lo cap a un cantó o cap a un altre o repetir l'execució d'unes instruccions.

Sovint utilitzem l'ordre *test* abreujada amb claudàtors o la construcció doble parèntesi per avaluar expressions i prendre decisions. Però ni els claudàtors de l'ordre *test* ni els parèntesis de la construcció doble parèntesi **no formen part de la sintaxi de cap estructura de control**del *shell*. No ho confongueu amb altres llenguatges de programació en què l'especificació de condicions en les estructures de control s'ha de fer entre parèntesis perquè la sintaxi ho requereix.

2.6.1 Estructures alternatives

Les estructures alternatives són aquelles que ens permeten executar una part del programa en funció de si es compleix o no una condició.

Estructura if

L'estructura *if* proporciona un control de flux basat en el codi de retorn d'una ordre. La sintaxi és:

```
if condició; then
  ordre1
  ordre2
  ...
  ordreN
fi
```

El *shell* executa l'ordre que estableix la condició potserior a *if* i avalua el codi de retorn resultant:

- Si el valor del codi de retorn és igual a zero, aleshores s'executen les ordres que hi hagi entre les paraules clau *then* i *fi.*
- Si el valor del codi de retorn és diferent de zero, no s'executen les ordres que hi hagi entre les paraules clau *then* i *fi* i el programa segueix executant el que hi hagi darrera de la paraula clau *fi*.

Habitualment utilitzem les ordres *test* i *let* (o el doble parèntesi equivalent) per especificar les condicions. Per exemple, el codi següent comprova si existeix el fitxer /etc/profile:

```
if test -f /etc/passwd; then
  echo "El fitxer /etc/passwd existeix."
fi
```

O bé amb la forma abreviada de *test*:

```
if [ -f /etc/passwd ]; then
  echo "El fitxer /etc/passwd existeix."
fi
```

Però la sintaxi d'*if* accepta qualsevol ordre, perquè totes les ordres generen un codi de retorn. Per exemple:

```
if grep ^root /etc/passwd > /dev/null; then
  echo "L'usuari root existeix."
fi
```

Noteu que l'estructura *if* implica una execució de l'ordre que estableix la condició i una avaluació del codi de retorn de manera implícita. De manera equivalent, es pot executar l'ordre que estableix la condició abans de l'*if* i avaluar el codi de retorn de manera explícita, per exemple:

```
test -f /etc/passwd
if [ $? -eq 0 ]; then
  echo "El fitxer /etc/passwd existeix."
fi
```

I també:

```
grep root /etc/passwd > /dev/null
if [ $? -eq 0 ]; then
  echo "L'usuari root existeix."
fi
```

Ara bé, la manera més habitual d'utilitzar l'estructura *if* és amb l'avaluació del codi de sortida de manera implícita.

Exemple d'utilització de l'estructura de control alternativa if

Feu un guió de shell que s'anomeni esfitx.sh que indiqui si un nom donat com a paràmetre és un fitxer regular.

```
#!/bin/bash
#
# esfitx.sh
#
# Rep un paràmetre i comprova si és un fitxer.
#
if [ -f $1 ]; then
  echo "$1 és un fitxer."
fi
exit 0
```

L'script anterior es pot millorar fent una comprovació del nombre de paràmetres rebuts.

```
#!/bin/bash
#
# esfitx.sh
#
# Rep un paràmetre i comprova si és un fitxer.
#
#
# Control del nombre de paràmetres
if [ $# -ne 1 ]; then
  echo "Nombre d'arguments erroni."
  echo "Ús del programa: $0 nom"
  exit 1
fi
#
# Comprovar si el paràmetre és un fitxer
if [ -f $1 ]; then
  echo "$1 és un fitxer."
```

```
fi
exit 0
```

Estructura if-else

L'estructura *if-else* permet prendre un curs d'acció si el codi de retorn de l'ordre que controla la condició és zero (veritat) i un altre curs d'acció si el codi de retorn és diferent de zero (fals). La sintaxi és la següent:

```
if condició; then
  ordres1
else
  ordres2
fi
```

Per veure el funcionament fem un guió de *shell* senzill que demani l'entrada d'un número per teclat i que a continuació ens digui si el número llegit és igual a 10 o no:

```
#!/bin/bash
#
# esdeu.sh
#
# Demana un nombre i comprova si és igual a 10.
#
echo "Introdueix un número: "
read X
if [ "$X" = 10 ]; then
  echo "El número és igual a 10."
else
  echo "El número no és igual a 10."
fi
```

En l'exemple que acabem de veure hem utilitzat l'ordre *test* per especificar la condició de l'estructura condicional, però també es podria haver fet amb l'ordre *let* o amb la construcció doble parèntesi de la manera següent:

```
echo "Introdueix un número: "
read X
if (( "$X" == 10 )); then
  echo "El número és igual a 10."
else
  echo "El número no és igual a 10."
fi
```

Exemple d'utilització de l'estructura de control alternativa if-else.

Feu un guió de shell que s'anomeni esdir.sh que rebi un argument de manera que si és un directori mostri el missatge "El contingut del directori <nom_directori> és:" i llisti el seu contingut. Si l'argument no és cap directori ha de donar un missatge informatiu. Feu el control del nombre d'arguments.

```
#!/bin/bash
#
# esdir.sh
#
# Rep el nom d'un directori per paràmetre i en mostra
# el contingut.
#
# Control del nombre de paràmetres.
if [ $# -ne 1 ]; then
```

```
10    echo "Nombre d'arguments erroni."
11    echo "Ús del programa: $0 nom_directori"
12    exit 1
13  fi
14  #
15  # Comprovar si el paràmetre és un directori.
16  if [ -d $1 ]; then
17    echo "El contingut del directori $1 és:"
18    ls $1
19  else
20    echo "$1 no és un directori."
21  fi
22  exit 0
```

Estructura if-elif-else

L'estructura *if-elif-else* es pot utilitzar per construir una bifurcació amb múltiples direccions. La sintaxi és:

```
1  if condició1; then
2    ordres1
3  elif condició2; then
4    ordres2
5  elif condició3; then
6    ordres3
7  ...
8  else
9    ordresN
10 fi
```

Noteu que la manera recomanada de tabular l'estructura *if-elif-else* és diferent de l'habitual.

Amb aquesta estructura, coneguda com a escala *if-else-if*, el *shell* avalua les expressions condicionals començant per la primera i continuant per la següent de forma descendent fins a trobar una condició veritable (codi de retorn igual a zero), moment en què s'executa la llista d'ordres associada a aquesta condició i se salta la resta de l'escala. Si cap condició és veritable s'executaran les ordres associades a l'*else* final. Si totes les condicions són falses i no hi ha *else* final, no fa res.

En realitat, l'estructura *if-then-elif* no és més que una manera abreviada d'escriure el mateix codi amb estructures *if-else* imbricades, és a dir, és equivalent al següent:

```
if condició1; then
  ordres1
else
  if condició2; then
    ordres2
  else
    if condició3; then
      ordres3
    else
      if
        ...
      else
        ordresN
      fi
    fi
  fi
fi
```

Per exemple, el programa següent llegeix un nombre del teclat i mostra per pantalla si és més petit, més gran o igual que 10, utilitzant una estructura *if-elif-else*:

Fixeu-vos en la forma d'escala que adopta el codi quan s'implementa amb estructures *if-else* i es tabula de la manera habitual. Per això es coneix amb el nom d'escala *if-else*.

```
echo "Introdueix un número: "
read X
if [ $X -lt 10 ]; then
  echo "El número és més petit que 10."
elif [ $X -gt 10 ]; then
  echo "El número és més gran que 10."
else
  echo "El número és igual a 10."
fi
```

Es podria haver escrit el mateix codi utilitzant estructures *if* i *if-else* de la manera següent:

```
echo "Introdueix un número: "
read X
if [ $X -lt 10 ]; then
  echo "El número és més petit que 10."
else
  if [ $X -gt 10 ]
  then
    echo "El número és més gran que 10."
  else
    echo "El número és igual a 10."
  fi
fi
```

Si el nombre de condicions és elevat, l'estructura *if-elif-else* permet compactar el codi.

Exemple d'utilització d'una escala if-else

Feu un shell script anomenat nota.sh que demani el valor d'una nota (un nombre enter) i ens digui si la nota és una D (0, 1, 2), una C- (3, 4), una C+ (5, 6), una B (7, 8) o una A (9, 10).

```
#!/bin/bash
#
# nota.sh
#
```

```
# Demana el valor d'una nota (enter) i diu si és
# D (0, 1, 2), C- (3 o 4), C+ (5 o 6),
# B (7 o 8) o A (9 o 10).
#
echo "Quina nota tens (un enter de 1 a 10)?"
read NOTA
if [ $NOTA -lt 0 ] || [ $NOTA -gt 10 ]; then
  echo "Nota fora de rang."
elif [ $NOTA -lt 3 ]; then
  echo "Tens una D."
elif [ $NOTA -lt 5 ]; then
  echo "Tens una C-."
elif [ $NOTA -lt 7 ]; then
  echo "Tens una C+."
elif [ $NOTA -lt 9 ]; then
  echo "Tens una B."
else
  echo "Tens una A."
fi
```

Estructura case

L'estructura *case* és útil quan tenim múltiples bifurcacions **basades en avaluacions de cadenes de caràcters**. La seva sintaxi és:

```
case "$NOM_VAR" in
  patró1)
    ordre1
    ...
    ordreN
    ;;
  patró2)
    ordre1
    ...
    ordreN
    ;;

  ...

  patróN)
    ordre1
    ...
    ordreN
    ;;
esac
```

Es fa una comparació seqüencial de la cadena que hi ha darrera de la paraula clau *case* amb els patrons. Quan es troba la primera coincidència s'executa la corresponent llista d'ordres i cap altra. Si no es troba cap coincidència no s'executa res.

Veiem-ne un exemple:

Noteu l'ús del patró *, que es pot utilitzar per indicar patrons per defecte.

```
echo "Tria una opció de 1 a 3: "
read OPCIO
case "$OPCIO" in
  1)echo "Has triat l'opció 1." ;;
  2)echo "Has triat l'opció 2." ;;
  3)echo "Has triat l'opció 3." ;;
  *)echo "Opció incorrecta."
esac
```

Els patrons són cadenes de caràcters i podem utilitzar els mateixos caràcters amb significat especial que utilitzem en la generació de noms de fitxers:

- *, per indicar coincidència amb qualsevol cadena de caràcters, inclòs el caràcter *null*.
- *?*, per indicar coincidència amb qualsevol caràcter simple.
- *[ ]*, per indicar coincidència amb qualsevol dels caràcters que hi hagi entre els claudàtors. Els caràcters de la llista van seguits un darrere de l'altre , no van separats per comes ni per espais ni per cap altre caràcter delimitador. S'accepten rangs amb el símbol menys (-).

També es pot utilitzar el símbol | si es vol coincidència amb més d'un patró (s'interpreta com un *o* lògic):

```
case "$NOM_VAR" in
  patró1|patró2|patró3)
    ordre1
    ...
    ;;
  patró4|patró5)

  ...
esac
```

L'exemple següent mostra l'ús de patrons amb caràcters amb significat especial. Noteu que en alguns casos pot haver més d'una manera d'expressar un mateix patró, per exemple, el patró [0-2] també es podria haver escrit *[012]* o bé *0|1|2*.

Exemple d'utilització d'una estructura case i ús de patrons

Feu el mateix guió de shell notes.sh que heu implementat amb una escala if-else, però ara utilitzant una estructura case.

```
#!/bin/bash
#
# nota2.sh
#
# Demana el valor d'una nota (enter) i diu si és
# D (0, 1, 2), C- (3 o 4), C+ (5 o 6),
# B (7 o 8) o A (9 o 10).
#
echo "Quina nota tens (enter de 1 a 10)?"
read NOTA
case "$NOTA" in
  [0-2])
    echo "Tens una D."
    ;;
```

```
  [34])
    echo "Tens una C-."
    ;;
  [56])
    echo "Tens una C+."
    ;;
  [78])
    echo "Tens una B."
    ;;
  9|10)
    echo "Tens una A."
    ;;
  *)
  echo "Nota fora de rang."
esac
```

2.6.2 Operadors && i ||

Els operadors de control && (AND) i || (OR) permeten fer un control de flux bàsic.

Operador &&

L'**operador *&&*** s'utilitza per executar una ordre, i, si té èxit (codi de sortida igual a zero), executar la propera ordre de la llista. La sintaxi és:

```
ordre1 && ordre2
```

L'ordre2 s'executa si i només si l'ordre1 torna un estat de sortida de zero (vertader). En altres paraules, s'executa ordre1 si el codi de sortida és igual a zero, llavors s'executa ordre2. És a dir, és equivalent a la utilització de l'estructura *if* així:

```
if ordre1; then
  ordre2
fi
```

Per exemple:

```
rm /tmp/fitxer && echo "Fitxer esborrat."
```

L'ordre *echo* només s'executa si l'ordre *rm* s'executa amb èxit (amb un codi de sortida de zero). Si el fitxer s'elimina correctament, l'ordre *rm* dóna un codi de sortida igual a zero i a continuació es mostra el missatge "Fitxer esborrat". Per evitar els possibles missatges d'error de l'ordre *rm*, podem redirigir la sortida d'errors a /dev/null així:

```
rm /tmp/fitxer 2>/dev/null && echo "Fitxer esborrat."
```

Exemple d'utilització de l'operador &&

Escriviu les ordres necessàries per comprovar si hi ha un directori anomenat /tmp/foo i donar un missatge d'error si no existeix.

```
test ! -d /tmp/foo && echo "El directori no existeix."
```

O amb la forma abreviada de test:

```
[ ! -d /tmp/foo ] && echo "El directori no existeix."
```

Operador ||

L'operador `||` s'utilitza per executar una ordre, i, si no té èxit (codi de sortida diferent de zero), executar la propera ordre de la llista. La sintaxi és:

```
ordre1 || ordre2
```

L'ordre2 s'executa si i només si l'ordre1 retorna un estat de sortida diferent de zero. En altres paraules, s'executa ordre1, si el codi de sortida de ordre1 és diferent de zero, llavors s'executa ordre2. Per exemple:

```
cat /etc/shadow 2>/dev/null || echo "No s'ha pogut obrir el fitxer."
```

L'ordre cat intentarà llegir el fitxer /etc/shadow i si falla mostrarà el missatge "No s'ha pogut obrir el fitxer".

Exemple d'utilització de l'operador OR

Escriviu les ordres necessàries per cercar el nom d'un usuari anomenat "messi" al fitxer /etc/passwd i donar un missatge si no es troba.

```
grep "^messi" /etc/passwd || echo "No s'ha trobat messi a /etc/
    passwd."
```

Combinació dels operadors && i ||

La llista d'ordres unides per operadors `&&` i `||` pot ampliar-se. Sovint s'utilitza la combinació dels operadors de la manera següent:

```
ordre_condició && ordre_cert || ordre_fals
```

És a dir, s'executa l'*ordre_condició.* Si el codi de sortida és:

- cert (0), llavors s'executa ordre_cert.
- fals (diferent de 0), llavors s'executa ordre_fals.

És a dir, és equivalent a la utilització de l'estructura *if-else* així:

```
if ordre_condicio; then
  ordre_cert
else
  ordre_fals
fi
```

Exemple d'utilització dels operadors OR i AND combinats

Escriviu les ordres necessàries per cercar el nom d'un usuari anomenat "messi" al fitxer /etc/passwd i donar missatges diferents segons si es troba o no.

```
grep "^messi" /etc/passwd || echo "No s'ha trobat messi a /etc/
    passwd." && echo "Trobat messi a /etc/passwd."
```

Escriviu les ordres necessàries per assegurar-vos que el superusuari root és qui està executant el guió de shell i donar un missatge depenent de si ho és o no.

```
test $(id -u) -eq 0 && echo "S'executa l'script amb el
    superusuari root." || echo "Només l'usuari root pot executar
    aquest script."
```

Escriviu les ordres necessàries per comprovar si el fitxer /etc/resolv.conf existeix i donar un missatge si hi és i un altre missatge si no hi és.

```
[ -f /etc/resolv.conf ] && echo "El fitxer /etc/resolv.conf
    existeix." || echo "El fitxer /etc/resolv.conf no existeix."
```

2.6.3 Estructures iteratives

Les estructures iteratives són aquelles que ens permeten executar diversos cops una part de codi.

Estructura while

L'estructura *while* permet l'execució repetitiva d'unes sentències sempre que l'ordre de control del bucle *while* sigui certa (codi de sortida igual a zero). La sintaxi és:

```
while condició; do
  ordre1
  ordre2
  ...
  ordreN
done
```

Normalment, s'utilitzen les ordres *test* o *let* per controlar la condició del bucle, però podem utilitzar qualsevol ordre que retorni un valor.

Veiem-ne un exemple senzill, un bucle que mostra per pantalla els números de l'1 al 10:

```
X=1
while (( X <= 10 )); do
  echo X val $X
  (( X=X+1 ))
done
```

Exemple d'utilització de l'estructura de control while

Feu un guió de shell que s'anomeni endevina.sh que assigni un número aleatori entre 1 i 6 a una variable N i que a continuació ens demani que l'encertem. El programa finalitza quan encertem el número i ens diu quants intents hem necessitat per endevinar-lo.

```
#!/bin/bash
# endevina.sh
# Joc per endevinar un número entre 1 i 6
#
INTENTS=1
# Calculem un número aleatori entre 1 i 6
# Mòdul 6 dóna un número entre 0 i 5
(( N = $RANDOM % 6 + 1 ))
echo –n "Endevina un número entre 1 i 6: "
read TRIA
while (( "$TRIA" != $N )); do
  echo
  echo –n "El número no és $TRIA, torna–ho a intentar: "
  read TRIA
  (( INTENTS++ ))
done
echo
echo "L'has endevinat, era el $N!!"
echo "Has necessitat $INTENTS intents."
exit 0
```

Estructura until

L'estructura *until* és idèntica a *while*, excepte que la condició és negada. La llista d'ordres s'executa sempre que la condició retorni un codi de sortida diferent de zero.

```
until condició; do
  ordre1
  ordre2
  ...
  ordreN
done
```

Per exemple:

```
X=1
until (( X > 10 )); do
  echo X val $X
  (( X=X+1 ))
done
```

Estructura for

L'estructura *for* permet especificar una llista de valors i executar les sentències per a cada valor de la llista. La sintaxi d'aquest bucle és la següent:

```
for NOM_VAR in llista_de_valors; do
  ordre1
  ordre2
  ...
  ordreN
done
```

La llista de valors és una seqüència de valors separats per espai o tabulador que s'aniran assignant a la variable *NOM_VAR* en cada iteració del bucle *for*. Per tant, les sentències que hi ha entre les paraules reservades *do* i *done*, s'executaran tantes vegades com valors hi hagi a la llista de valors.

La construcció clàssica de bucle *for* del *shell* difereix significativament de la del seu homòleg C i d'altres llenguatges de programació.

Per exemple, el següent és un bucle *for* que simplement mostra el valor de cada un dels ítems de la llista de valors (mostra els números de l'1 al 5):

```
for X in 1 2 3 4 5; do
  echo $X
done
```

L'estructura *for* és un mecanisme de bucle molt flexible. Es pot construir un bucle amb qualsevol llista que es pugui generar. Podem generar llistes fàcilment, per exemple, mitjançant la llista de paràmetres passats en la crida al *shell script* o mitjançant la substitució d'ordres. Així, l'exemple que acabem de veure es podria haver expressat obtenint la llista de valors amb el resultat d'executar l'ordre *seq* de la manera següent:

```
for X in $(seq 1 5); do
  echo $X
done
```

I en el cas que el nostre guió s'executi amb paràmetres, per exemple així:

```
nom_guió 1 2 3 4 5
```

El codi següent tindrà el mateix efecte que els anteriors:

```
for X in $*; do
  echo $X
done
```

La variable especial *$** conté la llista de paràmetres passats en la crida al programa. Teniu més informació al respecte a l'apartat "Paràmetres i variables especials" d'aquesta unitat.

Exemple d'utilització de l'estructura de control for

Feu un guió de shell que calculi l'ocupació de disc de cada directori de /home. Podeu obtenir la llista d'aquests directoris executant l'ordre ls dins del directori /home. Per calcular l'ocupació del directori podeu utilitzar l'ordre du (disk usage):

```
#!/bin/bash
# ocupacio.sh
# Calcula l'ocupació dels directoris de /home
#
cd /home
for DIR in $(ls); do
```

```
  if [ -d $DIR ]; then
    # Calcular espai ocupat pel directori
    du -sh $DIR
  fi
done
```

Feu el mateix guió de shell, però rebent per paràmetre el nom dels directoris dels quals volem conèixer l'espai que ocupen al disc.

```
#!/bin/bash
# Nom: espai.sh
# Mostra ocupació dels directoris rebuts per paràmetre
# Crida: espai.sh DIR1 DIR2 DIR3 ...
#
for DIR in $*; do
  if [ -d $DIR ]; then
    # Calcular espai ocupat pel directori
    du -sh $DIR
  else
    echo "Error: $DIR no és cap directori."
  fi
done
```

El *shell* Bash també permet utilitzar la construcció *for* a l'estil del llenguatge de programació C amb una sintaxi com la següent:

La sintaxi de *for* a l'estil de C que permet Bash no funciona amb sh (*shell* Bourne).

```
for (( expr1 ; expr2 ; expr3 )) ; do
  ordre1
  ordre2
  ...
  ordreN
done
```

En aquest cas les expressions només poden ser expressions aritmètiques, no poden ser qualsevol ordre i s'utilitzen amb el propòsit següent:

- *expr1*: per inicialitzar el bucle.
- *expr2*: per comprovar si la llista d'ordres entre *do* i *done* s'ha d'executar.
- *expr3*: per canviar la condició després de cada iteració del bucle.

És equivalent a:

```
(( expr1 ))
while (( expr2 )); do
  ordre1
  ordre2
  ...
  ordreN
  (( expr3 ))
done
```

Per exemple, per mostrar els números de l'1 al 5:

```
for ((x=1; x<=5; x++))
{
  echo $x
}
```

Veiem un altre exemple en què utilitzem bucles *for*imbricats (un bucle *for* dins d'un altre bucle *for*):

```
#!/bin/bash
#
# estrelles.sh
# Exemples de bucles for a l'estil de C
#
N=10
for (( i=1; i<=$N; i++ ))
do
  for (( j=1; j<=i; j++ ))
  do
    echo -n " *"
  done
  echo
done
for (( i=$N; i>=1; i-- ))
do
  for (( j=1; j<=i; j++ ))
  do
    echo -n " *"
  done
  echo
done
```

El resultat del guió de *shell* és:

```
*
* *
* * *
* * * *
* * * * *
* * * * * *
* * * * * * *
* * * * * * * *
* * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * *
* * * * * * * *
* * * * * * *
* * * * * *
* * * * *
* * * *
* * *
* *
*
```

2.7 Funcions

El llenguatge del *shell* Bash permet definir funcions, encara que amb una implementació una mica limitada.

Una **funció** és un bloc de codi que implementa un conjunt d'operacions, una "caixa negra" que duu a terme una tasca específica. La utilització de funcions en un *shell script* és una manera d'agrupar una serie d'ordres perquè puguin ser executades posteriorment utilitzant un sol nom.

Considerarem l'ús de funcions en els casos següents:

- Quan el programa de *shell* és complex, és recomanable fer una programació modular, és a dir, agrupar codi en funcions o mòduls per estructurar el programa.
- Sempre que tinguem codi repetitiu, és a dir, una part de codi que s'ha d'utilitzar més d'una vegada, cal considerar l'ús d'una funció.

Fitxers de funcions

Hi ha un munt de scripts en el sistema que utilitzen les funcions com una manera estructurada d'agrupar una sèrie d'ordres. En alguns sistemes Linux, per exemple, es troba l'arxiu /etc/rc.d/init.d/functions, un arxiu de definició de funcions que s'inclou al principi de tots els scripts d'inici. Usant aquest mètode, les tasques comunes, com la comprovació de si un procés s'executa, iniciar o aturar un dimoni, etc., només cal escriure-les una vegada.

Com a administradors del sistema, podeu crear el vostre propi fitxer de funcions, per exemple /usr/local/bin/funcions, amb totes les funcions que utilitzareu amb freqüència en els vostres guions de shell. Després, en cada guió que hagi d'utilitzar les funcions, només cal que escriviu la línia següent:

```
. /usr/local/bin/funcions
```

No deixeu de posar el punt i l'espai (equivalent a l'ordre source) abans del nom del fitxer que conté les funcions perquè el shell actual reconegui les funcions.

La sintaxi per definir una funció és:

```
1 function nom_de_la_funcio {
2   codi de la funcio
3 }
```

O bé:

```
1 nom_de_la_funcio () {
2   codi de la funcio
3 }
```

Cal tenir en compte les consideracions següents:

- El nom de la funció ha de ser únic en el *shell* o script.
- El caràcter *{* d'obertura de la funció pot estar a la segona línia.
- La definició de la funció ha d'estar feta abans de la primera crida a la funció que hi hagi al programa.

Es recomana fer totes les definicions de les funcions necessàries abans del codi principal del guió que s'executa, tret que hi hagi raons concretes per no fer-ho.

Això dóna una visió molt millor del programa i assegura que tots els noms de les funcions es coneixen abans de ser utilitzats. Generalment, l'estructura bàsica d'un guió de *shell* que implementa funcions és:

```
#!/bin/bash

VARIABLES_CONFIGURACIÓ

DEFINICIO_FUNCIONS

CODI_PRINCIPAL
```

Per cridar una funció des del programa de shell, simplement posem el nom de la funció, igual com fem amb qualsevol ordre. Quan es crida la funció, la llista d'ordres relacionades amb el nom de la funció s'executa.

```
#!/bin/bash
# funcs.sh
# Guió de shell per provar les funcions
#
# Definició de funcions
saludaUsuari() {
  echo "Execució de la funció $FUNCNAME..."
  echo Hola $USER!!
}
mostraData() {
  echo "Execució de la funció $FUNCNAME..."
  echo Avui és $(date)
}
### Programa principal ###
saludaUsuari
mostraData
```

2.7.1 Paràmetres a les funcions

Les funcions accepten paràmetres de la mateixa manera que les ordres o que els *shell scripts*. Per passar paràmetres a una funció només cal que en el moment de la crida a la funció especifiquem al costat del nom els valors dels paràmetres separats per espais. Per exemple:

```
nom_funcio param1 param2
```

La funció referencia els paràmetres rebuts per la posició que ocupen, igual que els paràmetres posicionals: `$1`, `$2`, etc. Tots els paràmetres es passen per valor, és a dir, quan retornem de la funció el valor dels paràmetres no haurà canviat.

Vegeu la descripció dels paràmetres posicionals i les variables especials a l'apartat "Paràmetres i variables especials" d'aquesta unitat.

Exemple de funció amb paràmetres

Feu una funció que faci la suma de dos números rebuts per paràmetre i en mostri el resultat. El programa principal ha de demanar els números per teclat.

```
#!/bin/bash
# suma.sh
# Demana dos números per teclat i en mostra la suma
#
# Definició de funcions
suma() {
  (( TOTAL = $1 + $2 ))
  echo "La suma de $1 i $2 és: $TOTAL"
}
### Programa principal ###
echo -n "Introdueix un número: "
read X
echo -n "Introdueix un altre número: "
read Y
# Crida a la funció suma() amb paràmetres
suma $X $Y
```

En executar una funció, els paràmetres passats a la funció esdevenen els paràmetres posicionals durant l'execució de la funció. Les variables especials *, @ i # també s'actualitzen per reflectir els canvis. La variable especial *0*, que té el nom del guió de *shell*, no canvia. La variable del Bash anomenada *FUNCNAME* s'estableix amb el nom de la funció mentre dura l'execució de la funció.

Quan es completa l'execució de la funció, els valors dels paràmetres posicionals i de les variables especials *, @ i # es restauren als valors que tenien abans de l'execució de la funció. Per tant, si necessitem accedir des d'una funció als valors dels paràmetres posicionals i de les variables especials *, @ i # del programa que crida a la funció, haurem de guardar-los prèviament a la crida de la funció.

Exemple de funció amb paràmetres

Un exemple típic de funció amb paràmetres és una funció per mostrar els missatges d'error d'un guió de shell. Imagineu que esteu fent un guió de shell i que teniu múltiples punts on el programa dóna missatges d'error:

```
if [ -z "$DIR" ] ; then
  echo "$0: especifiqueu el nom del directori."
  exit 1
fi
if [ ! -d $DIR ] ; then
  echo "$0: el directori no existeix."
  exit 1
fi
```

Quan això succeeix, és millor que creeu una funció per a aquest propòsit, per exemple:

```
mostraError() {
  echo "$0: $*"
  exit 1
}
...
if [ -z "$DIR" ] ; then
  mostraError "especifiqueu el nom del directori."
fi
if [ ! -d "$DIR" ] ; then
  mostraError "el directori no existeix."
fi
```

Noteu que la variable especial `$0` ha mantingut el valor dins de la funció i que la variable `$*` ha agafat el valor de la llista de paràmetres passats a la funció.

2.7.2 Codis de retorn

Les funcions sempre tornen un valor al *shell* que les crida anomenat *codi de retorn*, que és anàleg al codi de sortida de les ordres. El codi de retorn d'una funció pot ser especificat de manera explícita amb l'ordre return; altrament, el valor retornat per la funció és el valor del codi de sortida de l'última ordre executada. El codi de retorn de la funció pot utilitzar-se en el *shell script* mitjançant *$?*, de la mateixa manera que el codi de sortida de qualsevol altra ordre.

Ordre return

L'ordre **return** interromp l'execució d'una funció. La sintaxi de l'ordre és:

```
return [n]
```

De manera opcional accepta un nombre enter com a paràmetre que és retornat al guió de *shell* que fa la crida com el codi de retorn de la funció i és assignat a la variable *$?*. Si no indiquem paràmetre, el valor retornat és el valor del codi de retorn de l'última ordre executada abans del *return*. L'ordre return utilitzada fora del context d'una funció fa el mateix que l'ordre *exit*.

Exemple de funció amb codi de retorn

Modifiqueu la funció suma.sh perquè en lloc de mostrar el resultat de la suma dels dos números per pantalla, el retorni mitjançant l'ordre return. En el programa principal accediu al valor retornat per la funció amb la variable `$?`.

```
#!/bin/bash
# suma.sh
# Demana dos nombres per teclat i mostra la suma
#
# Definició de funcions
# Rep dos nombres i retorna la suma
suma() {
  (( TOTAL = $1 + $2 ))
  return $TOTAL
}
### Programa principal ###
echo -n "Introdueix un número: "
read X
echo -n "Introdueix un altre número: "
read Y
suma $X $Y
RESULTAT=$?
echo "La suma de $X i $Y és: $RESULTAT"
```

3. Planificació i automatització de tasques

En l'administració de sistemes sol ser necessària l'automatització de certes tasques a causa de la seva naturalesa repetitiva. Algunes d'aquestes tasques s'han de poder executar de manera no interactiva i s'han de poder planificar per fer-les sota algunes condicions, com ara, en horaris de menys ús de la màquina. Per dur a terme aquest tipus de feines com a serveis periòdics i programats hi ha diversos sistemes que ens permeten construir una mena d'agenda de tasques, és a dir, una planificació d'execució de les tasques.

3.1 Planificació de tasques

Un planificador de tasques és un programari que permet l'execució de tasques de manera desatesa (sense la intervenció de l'usuari) i d'acord amb les condicions descrites en funció d'un calendari o d'altres esdeveniments.

Les característiques que s'esperen d'un programari de planificació de tasques són:

- Permetre l'execució de tasques de manera automàtica.
- Tenir interfícies que ajudin a definir els fluxos de treballs o dependències entre tasques.
- Tenir interfícies que permetin la monitorització i el seguiment de les execucions de les tasques.
- Disposar d'algun mecanisme de prioritats o cues per controlar l'ordre d'execució dels treballs no relacionats.

Qualsevol programari que inclou totes o alguna d'aquestes característiques el podem considerar com un programari amb capacitats de planificació de tasques.

Des del punt de vista històric, podem distingir dues èpoques principals pel que fa als planificadors de tasques: l'era de l'ordinador central (*mainframe*) i l'era dels sistemes oberts i la informàtica distribuïda.

L'època de l'ordinador central es caracteritza pel desenvolupament de solucions de planificació sofisticades que inclouen característiques de planificació avançades i que formen part de les eines de gestió del propi sistema del *mainframe*. En l'ordinador central d'IBM, per exemple, el *job control language* (JCL) oferia des de bon principi la funcionalitat de gestionar dependències entre treballs.

Ordinador central

Un ordinador central o *mainframe* és un ordinador gran, potent i costós. En l'actualitat és utilitzat per grans companyies per a processaments de grans quantitats de dades; per exemple, per al processament de transaccions bancàries.

En l'època dels sistemes oberts, una gran diversitat de programari ofereix capacitats bàsiques de planificació de tasques:

- La majoria de sistemes operatius, com ara Windows o Unix i derivats, proporcionen eines de planificació que segueixen un model senzill per a l'execució de les tasques basat en els esdeveniments d'un calendari o en la càrrega del sistema en un moment determinat.
- Els serveis d'allotjament web ofereixen capacitats de planificació de treballs per mitjà d'un tauler de control o una solució de tipus webcron.
- D'altres aplicacions de diverses àrees, com ara programaris de còpies de seguretat, ERP, etc., incorporen facilitats per planificar les tasques pròpies de l'aplicació.

ERP

Els sistemes de planificació de recursos empresarials o ERP (de l'anglès enterprise resource planning) pretenen integrar totes les dades i processos d'una organització en un sistema unificat. Un sistema ERP típic utilitza múltiples components de programari i maquinari per aconseguir la integració de la producció, l'inventari, la distribució, els enviaments, les factures, la comptabilitat, les comandes, els lliuraments, els pagaments, etcètera.

En general, els sistemes operatius i aplicacions de l'era dels sistemes oberts no ofereixen la possibilitat de planificar l'execució de tasques més enllà d'una única instància de sistema operatiu o fora de l'àmbit del programa específic. A mesura que el nombre i la varietat de plataformes s'estén, les capacitats bàsiques de planificació de tasques es queden curtes i sorgeix la necessitat de sistemes de planificació avançats propers als planificadors estàndards dels ordinadors centrals i que a més proporcionin la possibilitat d'integrar diferents tipus de plataformes.

3.1.1 Sistemes distribuïts de planificació

Els sistemes distribuïts de planificació de treballs s'utilitzen en organitzacions amb un nombre de plataformes elevat i variat, per simplificar la gestió de la càrrega de treball de tota l'empresa i per tenir la capacitat de definir treballs en sistemes distribuïts, assegurant que s'executen en el temps i en la seqüència correcta.

En aquest tipus de sistemes se sol parlar de "gestió de càrrega de treball" en lloc de "planificació de tasques".

Normalment, les organitzacions mitjanes i grans disposen d'un nombre elevat de servidors amb diferents sistemes operatius i aplicacions, i un conjunt molt variat i molt ampli de treballs per executar, que poden tenir dependències complexes entre ells més enllà de la dimensió del temps o la càrrega del sistema. Per satisfer les demandes d'aquest tipus de centres, hi ha programaris específics de planificació de treballs –o de gestió de càrrega de treball– que són compatibles amb i que integren diverses plataformes i aplicacions i que disposen de característiques avançades com ara:

- Planificació de temps real basada en esdeveniments imprevisibles externs.
- Reinicialització automàtica de tasques i recuperació en cas de fallades.
- Alerta i notificació al personal d'operacions.

- Generació d'informes d'incidents.
- Registres d'auditoria per a propòsits de compliment de normatives.

En aquests sistemes hi ha diversos esquemes per decidir quin treball s'ha d'executar en un moment determinat. Per exemple, alguns paràmetres que poden ser considerats són:

- La prioritat de la feina.
- Els recursos de còmput existents.
- L'existència de la clau de llicència, si el treball està utilitzant programari amb llicència.
- El temps d'execució assignat a un usuari.
- El nombre de treballs permesos simultàniament a un mateix usuari.
- El temps estimat d'execució de la tasca.
- El temps transcorregut de la tasca.
- La disponibilitat de perifèrics.
- L'ocurrència d'esdeveniments requerits.

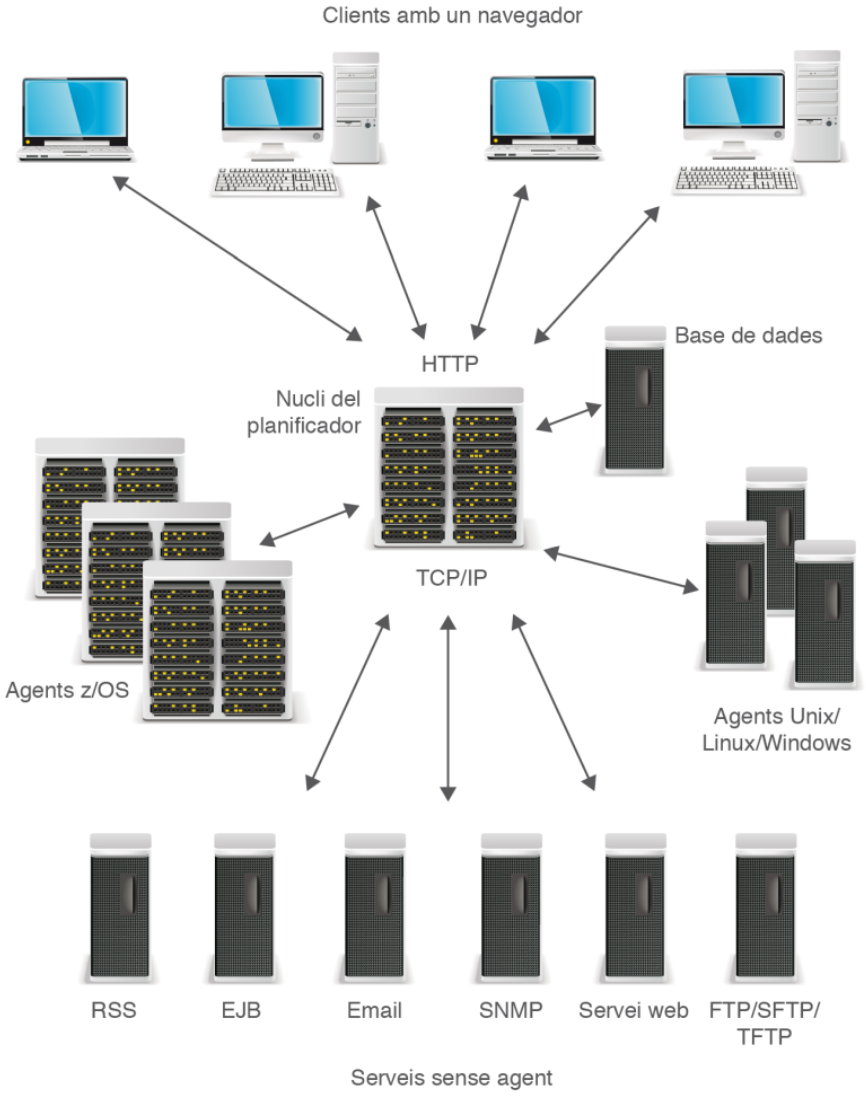
Aquests productes són complexos i solen oferir una interfície gràfica d'usuari amb un únic punt de control per a la definició i seguiment de les execucions de les tasques de tota la xarxa distribuïda d'ordinadors. La interfície d'usuari del programari sol ser de tipus web i normalment també incorpora una interfície de línia d'ordres.

La figura 3.1 mostra gràficament un entorn amb un sistema distribuït de planificació de treballs i una arquitectura típica implementada en aquests tipus de planificadors anomenada *arquitectura de mestre/agent*. El nucli del programari de planificació de treballs s'instal·la en una sola màquina (mestre), mentre que en les màquines de producció només s'instal·la un component molt petit (agent) que espera les ordres del mestre, les executa i li retorna el codi de sortida. Determinats serveis (FTP, SNMP, programes Java, etc.) no requereixen cap agent. Qualsevol màquina o dispositiu client amb un navegador permet el control i seguiment de les tasques.

Hi ha un ampli ventall de programari especialitzat en planificació de tasques per a sistemes distribuïts. L'elecció d'una eina sòlida continua sent essencial per a qualsevol empresa amb un nombre significatiu de servidors. Aquestes eines poden ser molt cares quan s'adquireixen de venedors tradicionals de programari de gestió de les TIC (tecnologies de la informació i la comunicació), però també hi ha eines de venedors independents a preus més competitius.

Vegeu una comparativa de programaris de planificació de tasques a la secció Adreces d'interès del web del mòdul.

FIGURA 3.1. Planificador de tasques distribuït



3.1.2 Planificadors del sistema operatiu

Les eines bàsiques de planificació proporcionades amb sistemes operatius com ara Unix i derivats o Windows s'utilitzen en entorns amb un nombre reduït de servidors que no requereixen planificació de tasques distribuïda. A continuació descriurem les més utilitzades en l'actualitat.

Sistema operatiu Windows

El planificador de tasques dels sistemes Windows de Microsoft s'anomena **task scheduler**, s'instal·la automàticament en el sistema operatiu com un servei i s'inicia cada vegada que el sistema és arrencat.

El task scheduler es pot utilitzar des de la línia d'ordres amb el programa `schtasks.exe`, però el més habitual és fer servir la interfície gràfica d'usuari

a la qual hi accedim a partir de *Panell de control* > *Sistema i seguretat* > *Eines administratives* > *Programador de tasques*.

La versió actual (2012) del task scheduler és la 2.0. Va ser introduïda amb Windows Vista i incorporada a Windows Server 2008. En aquesta versió la interfície d'usuari té un nou disseny basat en la consola de gestió de Microsoft (MMC, *Microsoft management console*) i les capacitats de planificació de tasques es milloren respecte la versió anterior, la 1.0.

La consola MMC

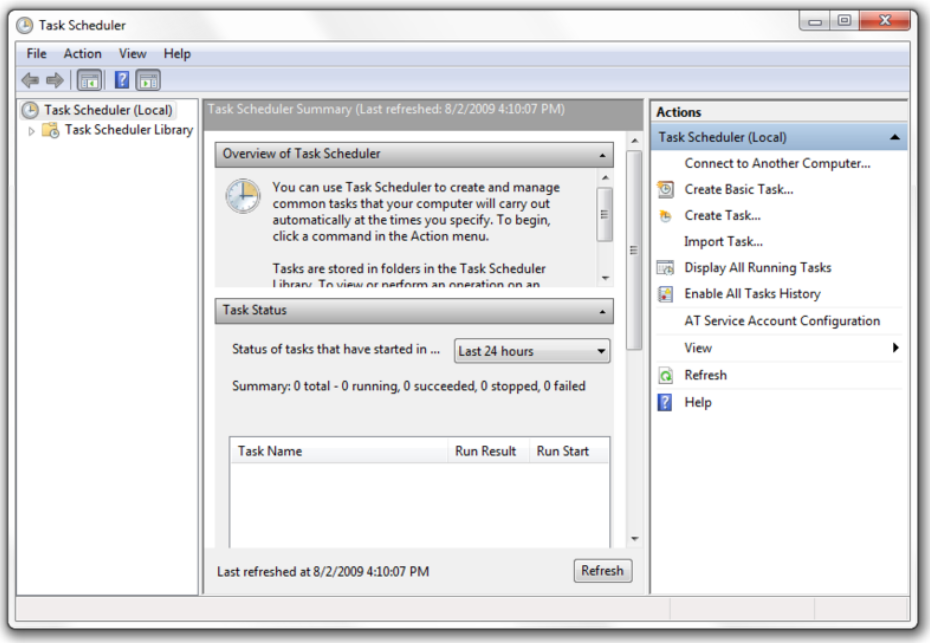
MMC (Microsoft management console) és un component de Windows 2000 i dels seus successors que proveeix als administradors del sistema una interfície per configurar i monitoritzar el sistema.

El task scheduler 2.0 permet seleccionar el tipus d'acció que volem planificar d'entre les següents:

- Executar una aplicació o guió de *shell*.
- Enviar un correu electrònic.
- Mostrar una caixa de diàleg amb un missatge.

La figura 3.2 mostra una finestra amb el task scheduler executant-se en un sistema operatiu Windows 7.

Figura 3.2. Planificador de tasques de Windows 7



Les accions es poden planificar perquè siguin executades quan es produeixi alguna de les condicions següents:

- D'acord amb una programació (a una hora específica una vegada, diàriament, setmanalment o mensualment).
- En obrir o tancar la sessió.
- En iniciar el sistema.
- Quan el sistema entra en estat inactiu.

- En produir-se un esdeveniment determinat del sistema.
- En bloquejar-se o desbloquejar-se l'estació de treball.

Sistemes de tipus Unix

En els sistemes de tipus Unix hi ha diverses eines relacionades amb la planificació de tasques:

- El programa cron, un planificador basat en el temps que assumeix que el sistema està sempre en funcionament.
- El programa anacron, que fa les tasques de cron però sense assumir que el sistema està sempre en funcionament. Aquest programa no substitueix a cron, és una eina complementària.
- L'ordre *at*, la qual es fa servir per fer execucions retardades i planificar tasques que es volen executar una sola vegada a una hora determinada en el futur.

Càrrega del sistema

La càrrega del sistema és un paràmetre que ens indica el grau d'activitat de l'ordinador. En sistemes de tipus Unix podem veure aquesta càrrega de manera interactiva amb l'ordre `top`.

Mentre que cron, anacron i *at* són eines per executar tasques quan arriba un determinat moment en el temps, hi ha d'altres eines que permeten executar tasques quan es dóna un esdeveniment, per exemple:

- L'ordre *batch* s'utilitza per executar tasques quan el nivell de càrrega del sistema ho permeti.
- El servei incron està dissenyat per executar tasques quan es produeix un esdeveniment en el sistema de fitxers. incron pot examinar un fitxer específic o un directori sencer i reaccionar a diferents esdeveniments com ara la creació, la modificació o l'esborrat de fitxers, etc.

Si bé tenim a l'abast aquestes i altres eines, el planificador per excel·lència és el cron i és en el que ens centrarem.

3.2 El planificador cron

Cron és el nom del programa que permet als usuaris de sistemes Unix i derivats planificar l'execució d'ordres o guions de *shell* de manera automàtica a una data i temps específics. És utilitzat sovint pels administradors de sistemes com a eina per automatitzar les tasques de manteniment, com ara les còpies de seguretat, però pot ser utilitzat per a qualsevol altre objectiu.

El nom de cron ve de cronògraf, l'aparell per enregistrar intervals de temps.

Cron ha estat reescrit diversos cops durant la seva història i en podem trobar implementacions, com ara la d'AT&T, que poden diferir lleugerament del **cron vixie**, que és el que utilitzarem i descriurem en els apartats següents.

Cron vixie

El seu autor, Paul Vixie, ho és també de Bind, el servidor de DNS lliure més utilitzat.

3.2.1 Iniciar el servei cron

Cron és un dimoni (servei) i generalment s'instal·la automàticament en fer la instal·lació del sistema operatiu i queda configurat per ser iniciat amb l'arrencada del sistema. Es pot comprovar l'estat del servei executant l'ordre:

```
1 /etc/rc.d/init.d/cron status
```

Executeu man cron per veure la pàgina de manual de l'ordre cron.

També podem veure si el dimoni està en execució mitjançant l'ordre *ps*:

```
1 ps -ef | grep cron
```

Si per alguna raó cron no està funcionant es pot iniciar el servei amb el superusuari *root* i amb el guió de *shell* d'inicialització corresponent:

```
1 /etc/rc.d/init.d/cron start
```

En altres sistemes potser anomenen al dimoni *crond* (*cron daemon*).

3.2.2 El fitxer de crontab

Anomenem genèricament el fitxer que utilitzem per configurar la planificació de les tasques que ha d'executar cron *fitxer de **crontab***.

Un fitxer de crontab conté instruccions per al servei cron en la forma general: "Executa aquesta ordre a aquesta hora en aquesta data".

La planificació de tasques amb cron es pot fer de dues maneres, a nivell d'un usuari particular i a nivell de tot el sistema i els fitxers de crontab en cada cas difereixen lleugerament.

Executeu `man 5 crontab` per veure la pàgina de manual sobre el format dels fitxers de crontab.

Quan el cron és utilitzat **a nivell d'usuari**, les línies d'un fitxer de crontab es formen amb sis camps separats per espais o tabuladors de la manera següent:

```
1 minut hora diaDelMes mes diaSetmana ordre
```

Els cinc primers camps indiquen el calendari d'execució i el sisè camp especifica la tasca que s'ha d'executar. Cada línia correspon a una tasca i no hi ha límit de tasques.

En el cas que el cron sigui utilitzat **a nivell de sistema**, afegim a les línies del fitxer de crontab un sisè camp per indicar el nom de l'usuari que ha d'executar l'ordre:

```
1 minut hora diaDelMes mes diaSetmana usuari ordre
```

En tots els casos, el significat de cada camp és el següent:

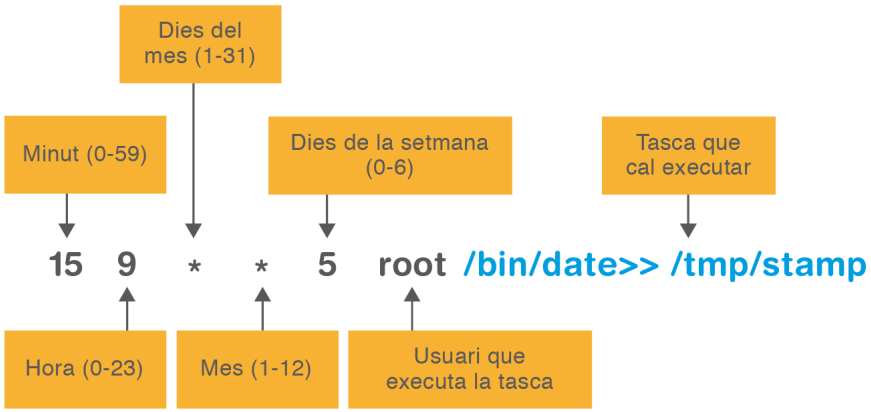
- *minut*: indica el minut de l'hora en què l'ordre serà executada. Ha de ser un valor entre 0 i 59.
- *hora*: indica l'hora en què l'ordre serà executada. Ha de ser un valor entre 0 i 23, éssent 0 la mitjanit.
- *diadelmes*: indica el dia del mes en què l'ordre serà executada. Ha de ser un valor entre 1 i 31.
- *mes*: Indica el mes en què l'ordre serà executada. Pot ser indicat numèricament (un valor entre 1 i 12) o amb les tres primeres lletres del nom del mes en anglès (*jan*, *feb*, *mar*, etc.).
- *diasetmana*: indica el dia de la setmana en què l'ordre serà executada. Pot ser indicat numèricament (un valor entre 0 i 7, sent tant el 0 com el 7 el diumenge) o amb les tres primeres lletres del nom del dia en anglès (*mon*, *tue*, etc.).
- *usuari*: indica l'usuari que executarà l'ordre (només a /etc/crontab).
- *ordre*: ordre, script o programa que es vol executar. Aquest camp ocupa fins al final de la línia; pot contenir múltiples paraules i espais.

Es poden utilitzar els símbols especials següents per indicar els valors dels camps:

- Un asterisc, *, indica tots els valors possibles.
- Llistes de valors separats per comes. Per exemple: *1,2,5,9*.
- Rangs de valors separats pel guió. Per exemple: *8-11* (indica 8, 9, 10 i 11).
- Intervals periòdics mitjançant **/valor*. Per exemple: */5 en el camp *minut* indica cada 5 minuts: 5, 10, 15, 20, 25…

La figura 3.3 mostra gràficament el format d'una línia d'un fitxer de crontab de sistema.

FIGURA 3.3. Format d'una línia de crontab de sistema



La taula 3.1 mostra exemples d'expressions temporals de cron.

Taula 3.1. Exemples d'expressions de cron

minut hora diaDelMes mes diaSetmana	Descripció
17 * * * *	En el minut 17 de cada hora de tots els dies.
25 6 * * *	A les 6.25 am cada dia.
47 6 * * 7	A les 6.47 am tots els diumenges.
52 6 1 * *	A les 6.52 am del primer de cada mes.
* 5 * * *	Cada minut de 5 a 5.59 am.
59 11 * 1-3 1,2,3,4,5	A les 11.59 am de dilluns a divendres, de gener a març.
10,30,50 * * * 1,3,5	En el minut 10, 30 i 50 de totes les hores dels dilluns, dimecres i divendres.
*/15 10-14 * * *	Cada quinze minuts de les 10.00 am a les 2.00 pm.
0 */5 1-10,15,20-23 * 3	Cada 5 hores dels dies 1 al 10, del dia 15 i del dia 20 al 23 de cada mes i que el dia sigui dimecres.

Exemple de crontab d'un usuari

El següent és un exemple del contingut d'un fitxer de crontab d'un usuari. Les línies precedides d'un símbol # són comentaris i són ignorades:

```
# Executar 5 minuts després de la mitjanit cada dia
5 0 * * * $HOME/bin/diari.sh
# Executar a les 2.15 pm el primer dia de cada mes
15 14 1 * * $HOME/bin/mensual.sh
```

Hi ha uns valors predefinits que es poden utilitzar per substituir tota l'expressió de cron. La taula 3.2 mostra quins són aquests valors.

Taula 3.2. Valors especials de cron

Entrada	Descripció	Equivalència
@yearly	S'executa un cop a l'any.	0 0 1 1 *
@annualy	Igual que @yearly.	0 0 1 1 *
@monthly	S'executa un cop al mes.	0 0 1 * *
@weekly	S'executa un cop a la setmana.	0 0 * * 0
@daily	S'executa un cop al dia.	0 0 * * *
@midnight	Igual que @daily.	0 0 * * *
@hourly	S'executa un cop cada hora.	0 * * * *
@reboot	S'executa cada vegada que el servei de cron es reinicia, normalment coincidirà amb la reinicialització del servidor.	Sense equivalència.

Hi ha diverses variables d'entorn que són establertes de manera predeterminada pel servei de cron. Per exemple, la variable *SHELL* s'estableix per defecte a /bin/sh per indicar que les tasques s'executin amb aquest *shell*, o la variable *PATH* s'estableix a /usr/bin:/bin.

Les ordres o programes que funcionen en un *shell* interactiu poden no funcionar correctament en ser executades amb cron ja que determinades variables d'entorn no tenen els valors adequats. Al fitxer de crontab es poden canviar els valors per defecte de les variables d'entorn afegint les línies de definició de variables amb la forma NOM_VAR=valor.

Exemple de crontab d'usuari amb definició de variables d'entorn

El següent és un exemple del contingut d'un fitxer de crontab d'un usuari que inclou definició de variables d'entorn:

```
SHELL=/bin/bash
PATH=~/bin:/usr/bin/:/bin
5 0 * * * $HOME/bin/diari.sh
15 14 1 * * $HOME/bin/mensual.sh
```

Planificació de tasques amb el crontab d'usuari

El planificador de tasques cron permet que cada usuari tingui el seu propi fitxer de planificació de tasques o crontab. Els crontab dels usuaris es guarden al directori `/var/spool/cron/crontabs` (aquest directori pot variar segons la versió d'Unix/Linux) amb el nom de l'usuari del sistema que ha generat el fitxer. Per exemple, hi podem trobar un fitxer anomenat root per al crontab del superusuari *root*, un fitxer anomenat alba per al crontab de la usuària *alba*, i així per a cada usuari del sistema.

Executeu `man crontab` per veure la pàgina de manual de l'ordre `crontab`.

En lloc que cada usuari pugui modificar al seu gust els fitxers que hi ha al directori `/var/spool/cron/crontabs`, existeix un programa denominat `crontab` que serveix per gestionar aquests fitxers, però d'una forma controlada.

Amb l'ordre `crontab` podem crear o modificar un fitxer de planificació de tasques, llistar-ne el contingut o esborrar-lo.

La sintaxi de l'ordre `crontab` és la següent:

```
crontab [ -u usuari ] fitxer
crontab [ -u usuari ] { -e | -l | -r }
```

L'opció -u només la pot utilitzar *root* i permet gestionar el crontab d'un altre usuari en lloc del propi.

La primera forma d'ús de crontab permet a l'usuari que executa l'ordre generar un fitxer de crontab a partir d'un fitxer creat prèviament. L'usuari crea un fitxer de text amb les línies corresponents a les tasques que vol planificar amb el format adequat, és a dir:

```
minut hora diaDelMes mes diaSetmana ordre
```

Un cop creat el fitxer de planificació de tasques, l'usuari executa l'ordre *crontab* posant com a argument el nom del fitxer:

```
crontab fitxer
```

Per exemple, la usuària *maria* crea un fitxer anomenat *tasques* que conté la línia següent:

```
0 22 * * * /home/maria/proces.sh
```

I a continuació executa l'ordre:

```
crontab tasques
```

En utilitzar l'ordre *crontab*, en gravar el fitxer es comprova automàticament que la sintaxi és correcta.

Si el fitxer de tasques no conté errades sintàctiques, l'ordre `crontab` generarà el fitxer `/var/spool/cron/crontabs/maria` amb la tasca planificada per la usuària *maria*. En cas contrari, l'ordre dóna un missatge informatiu indicant l'errada trobada i no instal·la el fitxer de crontab.

L'ordre `crontab` és pot utilitzar sense necessitat de rebre un fitxer com a argument. En aquest cas la sintaxi és:

```
crontab -e
```

L'ordre anterior obre l'editor preestablert de l'usuari i permet crear o modificar el fitxer de planificació de tasques directament. En sortir i desar el fitxer, si crontab no hi troba errades, crearà o modificarà el fitxer corresponent a `/var/spool/cron/crontabs` amb el nom de l'usuari.

Editor preestablert

Podeu canviar l'editor preestablert pel sistema amb la variable d'entorn EDITOR o la variable VISUAL. Per exemple:

```
export EDITOR=/usr/bin/vi
```

Amb l'opció *-l*, crontab permet llistar les tasques que té planificades l'usuari:

```
crontab -l
```

Amb l'opció *-r* s'eliminen totes les tasques de cron de l'usuari:

```
crontab -r
```

Exemple de gestió d'un fitxer crontab d'usuari amb l'ordre //crontab//

Genereu un crontab per al vostre usuari que guardi cada minut la data del sistema en un fitxer del vostre directori d'inici anomenat *provacron.log*.

Primerament, modifiqueu la variable d'entorn *EDITOR* per tal que s'obri l'editor gedit enlloc del preestablert pel sistema:

```
export EDITOR=/usr/bin/gedit
```

A continuació executeu l'ordre *crontab* amb l'opció *-e* per editar directament el fitxer de crontab:

```
crontab -e
```

S'obre el gedit amb un fitxer nou buit. Introduïu la línia desitjada:

```
* * * * * date >> /home/usuari/provacron.log
```

Deseu els canvis i sortiu. Si no hi ha cap errada sintàctica en els camps requerits, us sortirà un missatge informant-vos que s'ha instal·lat el vostre crontab.

Amb **l'usuari *root***podeu comprovar que s'ha creat un fitxer /var/spool/cron/crontabs/nom_usuari, que és un fitxer de text que conté el crontab que acabeu de generar.

Podeu verificar que el fitxer provacron.log va guardant a cada minut la data i hora del sistema obrint un nou terminal i executant:

```
tail -f /home/usuari/provacron.log
```

En una altra finestra de terminal, podeu llistar el contingut del vostre crontab amb l'ordre `crontab -l`.

Si hi voleu fer canvis, podeu tornar a editar el vostre fitxer de crontab amb `crontab -e`.

Finalment, després de fer totes les proves, podeu eliminar el vostre crontab amb l'ordre `crontab -r`.

Els fitxers de crontab ubicats al directori `/var/spool/cron/crontabs` són fitxers de text pla. No obstant això, quan treballem amb l'usuari *root* i volem modificar el seu crontab o el de qualsevol altre usuari, no hem d'editar directament aquests fitxers, sinó que hem d'utilitzar l'ordre `crontab` igual que fan la resta d'usuaris per assegurar-nos que no fem errades de sintaxi en els camps de planificació de les tasques.

Planificació de tasques amb el crontab del sistema

El fitxer de crontab del sistema es diu `/etc/crontab`. És un fitxer de text que només pot modificar l'usuari *root*. Té el mateix format que els crontab d'usuari, però s'afegeix un camp a les línies per especificar amb quin usuari s'executa cada tasca.

El format de les línies del fitxer és:

```
minut hora diaDelMes mes diaSetmana usuari ordre
```

Malgrat que tenen el mateix nom, no confongueu el fitxer de configuració `/etc/crontab` amb l'ordre de gestió del cron dels usuaris, `/usr/bin/crontab`.

Els canvis en el fitxer /etc/crontab prenen efecte pel sol fet d'editar-lo i modificar-lo, és a dir, a diferència del que fem amb els crontab d'usuari, no cal executar l'ordre *crontab* per instal·lar-lo.

El fitxer `/etc/crontab` es crea amb un contingut per defecte similar al següent:

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the 'crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow usercommand
17 * * * * root run-parts --report /etc/cron.hourly
25 6 * * * root run-parts --report /etc/cron.daily
47 6 * * 7 root run-parts --report /etc/cron.weekly
```

```
14 52 6 1 * * root run-parts --report /etc/cron.monthly
```

Les primeres línies sense símbol de comentari són definicions de variables. La variable *SHELL* indica el *shell* amb el qual s'han d'executar les tasques, i la variable *PATH*, els camins dels directoris en els quals cron buscarà les tasques que s'han d'executar.

Després de les variables, hi ha les línies que executen les tasques planificades:

- La primera línia indica que s'executi la tasca en el minut 17 de cada hora de tots els dies.
- La segona línia indica que s'executi a les 6.25 am de cada dia.
- La tercera línia, a les 6.47 am tots els diumenges.
- La darrera, a les 6.52 am del primer de cada mes.

Programa run-parts

L'ordre `run-parts` serveix per executar tots els scripts i programes situats al directori especificat. Executeu `man run-parts` per veure la pàgina de manual de l'ordre.

L'usuari especificat per executar totes les tasques és *root* i l'ordre planificada sempre és `run-parts`, però posant com a argument en cada cas el nom d'un directori diferent:

- /etc/cron.hourly
- /etc/cron.daily
- /etc/cron.weekly
- /etc/cron.monthly

Com a administradors del sistema, podem desar un fitxer executable pel *shell* (guió de *shell*, programa compilat, etc.) dins de qualsevol d'aquests directoris perquè sigui executat una vegada cada hora, dia, setmana o mes, respectivament, a l'hora configurada en el fitxer /etc/crontab. Per exemple, si es deixa un *shell script* dins de `/etc/cron.daily` s'executarà a les 6.25 am de cada dia.

En el fitxer `/etc/crontab` es poden afegir línies addicionals per executar qualsevol altra ordre, però no és recomanable, ja que aquest fitxer pot ser sobreescrit en fer actualitzacions del sistema. És més recomanable posar les tasques en els directoris /etc/cron.hourly, /etc/cron.daily, etc. o bé utilitzar els crontab d'usuari.

Exemple de planificació de tasques amb el fitxer /etc/crontab

Tot i que no és recomanable, podem trobar sistemes en què s'han afegit línies per planificar tasques concretes al fitxer /etc/crontab del sistema. Per exemple:

```
1 0 22 * * * root /usr/local/diari.sh
2 0 8,20 * * * helena -s "Sistema viu" helena@ioc.cat
```

En aquest exemple, la primera tasca l'executa l'usuari *root* i la segona la usuària *helena*. La primera línia indica que s'executi l'script diari.sh tots els dies a les 10 pm. La segona línia indica que s'enviï un correu tots els dies a les vuit del matí i a les vuit del vespre amb l'assumpte "Sistema viu" a l'adreça helena@ioc.cat. És un mètode senzill d'estar assabentat de si un sistema remot està actiu a les hores indicades. Si la usuària no rep un correu en aquestes hores, significa que alguna cosa no va bé.

Crontabs de /etc/cron.d

En alguns sistemes, hi ha també un directori anomenat `/etc/cron.d` on es poden col·locar fitxers de crontab. Els fitxers d'aquest directori tenen el mateix format que el fitxer /etc/crontab, és a dir, les línies inclouen el camp de l'usuari:

```
minut hora diaDelMes mes diaSetmanausuari ordre
```

En general, l'administrador no hauria d'utilitzar el directori /etc/cron.d/. La finalitat d'aquest directori és permetre que les aplicacions que requereixen un control més fi de la seva planificació que el proporcionat pels directoris /etc/cron.daily, /etc/cron.weekly, etc, afegeixin un fitxer propi de crontab a /etc/cron.d. Aquests fitxers han de dur el nom del paquet de programari que els instal·la i sovint són enllaços a fitxers on, tant l'enllaç com el fitxer, tenen com a propietari l'usuari *root*.

Exemple de contingut del directori /etc/cron.d

El següent és un exemple dels fitxers trobats en un directori /etc/cron.d d'un sistema determinat. Llistem el contingut del directori amb l'ordre:

```
ls /etc/cron.d/
```

I obtenim aquesta sortida:

```
anacron mdadm php5
```

Veiem que hi ha tres programaris que han instal·lat un crontab. Examinem el contingut d'un dels fitxers anteriors, per exemple php5, per veure'n el format:

```
cat php5
```

Podem apreciar que les línies no comentades tenen el mateix format que les del fitxer /etc/crontab:

```
# /etc/cron.d/php5: crontab fragment for php5
# This purges session files older than X, where X is defined in
    seconds
# as the largest value of session.gc_maxlifetime from all your
    php.ini
# files, or 24 minutes if not defined. See /usr/lib/php5/
    maxlifetime

# Look for and purge old sessions every 30 minutes
09,39 * * * * root [ -x /usr/lib/php5/maxlifetime ] && [ -d /var/
    lib/php5 ] && find /var/lib/php5/ -type f -cmin +$(/usr/lib/
    php5/maxlifetime) -print0 | xargs -n 200 -r -0 rm
```

Un exemple d'aplicació que trobarem que instal·la un fitxer de crontab al directori /etc/cron.d és anacron. És un programari que no pretén substituir a cron, sinó que funciona amb cron. El sistema de cron implica que la màquina està sempre encesa, cosa que és certa en la majoria dels casos quan parlem de servidors, però per a aquells casos que això no és així, per exemple quan es tracta d'estacions de treball, és millor utilitzar anacron, que verifica si l'acció no es va fer quan l'hauria hagut de fer, i llavors l'executa.

Executeu *man anacron* per saber més sobre el funcionament d'aquest programari.

3.2.3 Sortida de les tasques de cron

El cron es desperta cada minut, examina tots els fitxers de crontab existents i executa les tasques que tenen els camps que es compleixen en aquell precís minut. Qualsevol sortida de la tasca s'envia per correu electrònic a l'usuari que l'executa o, si existeix, a l'usuari indicat en la variable d'entorn del crontab anomenada *MAILTO*.

En planificar tasques de manteniment del sistema, sovint volem guardar el rastre de les accions que es fan en fitxers de registre. Per redirigir la sortida d'una tasca que executem amb cron i enviar-la a un fitxer enlloc de al correu electrònic de l'usuari, utilitzem redireccions, ja sigui dins del guió de *shell* o en escriure l'ordre en la línia del crontab.

Tenim diverses opcions de redirecció. Les més utilitzades són:

- Ignorar la sortida redirigint a /dev/null: ordre > /dev/null
- Crear o afegir a un fitxer de registre: ordre >> nom_fitxer
- Redirigir també la sortida d'errors: ordre >> nom_fitxer 2>&1

Per exemple:

```
1 5 0 * * * $HOME/bin/diari.sh >> $HOME/tmp/diari.out 2>&1
```

La línia anterior especifica que s'executi cada dia a les 0.05 el guió de *shell* $HOME/bin/diari.sh. La sortida d'aquesta tasca està redirigida amb l'operador >> cap a un fitxer anomenat *$HOME/tmp/diari.out*, de manera que cada vegada que el guió de *shell* s'executa, la seva sortida es va afegint al final del fitxer. La sortida d'errors també està redirigida amb *2>&1* per tal que, si hi ha errors, s'afegeixin al mateix fitxer de sortida.

Vegeu més opcions de redirecció d'entrada i sortida a l'apartat "Redirecció de l'entrada i la sortida" d'aquesta unitat.

3.2.4 Notificacions del servei cron

El cron utilitza el servei anomenat ***syslog*** per enregistrar la seva activitat. Per defecte, està configurat per enregistrar un nivell de detall bàsic al fitxer `/var/log/syslog`.

Per exemple, podem trobar un rastre com el següent:

```
Feb 27 12:33:02 saturn crontab[2671]: (umart) BEGIN EDIT (umart)
Feb 27 12:33:23 saturn crontab[2671]: (umart) REPLACE (umart)
Feb 27 12:33:23 saturn crontab[2671]: (umart) END EDIT (umart)
Feb 27 12:34:01 saturn /usr/sbin/cron[1253]: (umart) RELOAD (crontabs/umart)
Feb 27 12:34:01 saturn /USR/SBIN/CRON[2677]: (umart) CMD (echo hola)
```

En les línies anteriors podem llegir que l'usuari *umart* està treballant en el servidor *saturn* i ha editat el seu crontab el 27 de febrer. Després el cron s'ha recarregat i s'ha executat una tasca del mateix usuari.

Podem configurar syslog perquè es creï un fitxer de registre exclusiu per a cron. Per fer-ho cal editar el fitxer de configuració `/etc/rsyslog.conf` i treure el símbol de comentari, #, de la línia següent:

```
#cron.*   /var/log/cron.log
```

Vegeu més informació sobre el servei syslog a l'apartat "Manteniment dels fitxers de registre" d'aquesta unitat.

Si volem activar un nivell superior de detall de registre de l'activitat de cron, hem d'editar el fitxer de configuració de cron anomenat `/etc/default/cron` i eliminar el símbol de comentari, #, de la línia EXTRA_OPTS="-L 2". El contingut del fitxer quedarà així:

```
# Cron configuration options
...
# Extra options for cron, see cron(8)
# Set a higher log level to audit cron's work
EXTRA_OPTS="-L 2"
```

3.2.5 Control d'accés a cron

És possible controlar quins usuaris poden o no utilitzar els serveis de cron d'una manera molt senzilla amb els fitxers `/etc/cron.allow` i `/etc/cron.deny`.

Si el fitxer /etc/cron.allow existeix, llavors l'usuari ha d'estar llistat (un usuari per línia) dins d'aquest fitxer per poder fer servir l'**ordre *crontab***. Si el fitxer /etc/cron.allow no existeix però existeix el fitxer /etc/cron.deny, llavors l'usuari no ha d'estar llistat dins de /etc/cron.deny per poder usar l'ordre *crontab*. Si es vol evitar que tots els usuaris utilitzin cron es pot escriure *ALL* dins del fitxer /etc/cron.deny.

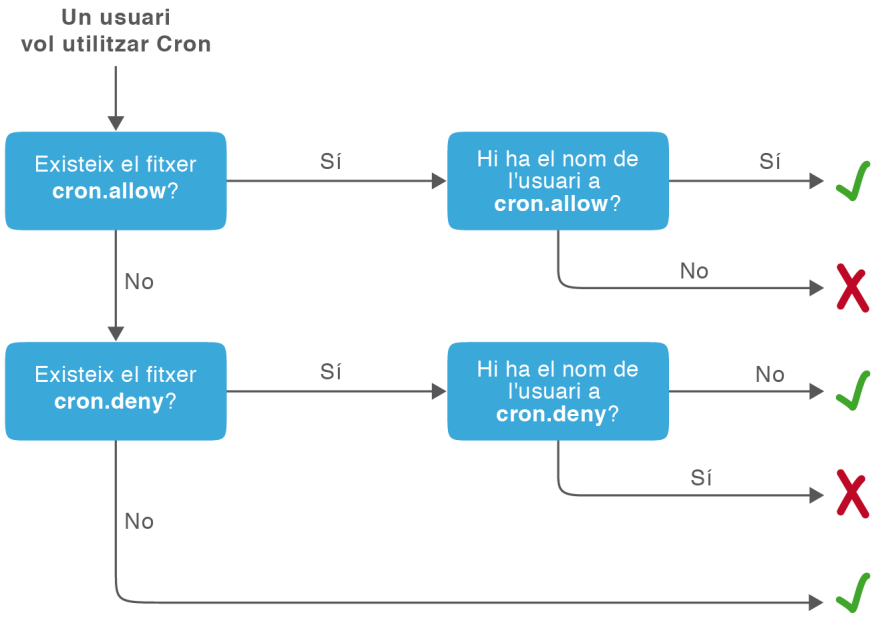
Vegeu el propòsit i el funcionament de l'ordre *crontab* en l'apartat "Crontabs d'usuari" d'aquesta unitat.

Cal tenir en compte les consideracions següents:

- Si cap dels dos fitxers existeix, per defecte tots els usuaris del sistema poden planificar treballs utilitzant l'ordre *crontab*.
- Si els dos fitxers existeixen, llavors el fitxer /etc/cron.allow té precedència, és a dir, el fitxer /etc/cron.deny s'ignora i l'usuari ha d'estar llistat dins de /etc/cron.allow per poder usar l'ordre crontab.

La figura 3.4 il·lustra gràficament el funcionament del control d'accés a cron.

Figura 3.4. Funcionament del control d'accés a cron



Independentment de l'existència d'aquests dos fitxers, l'usuari *root* sempre pot planificar tasques, ja sigui amb l'ordre *crontab* o amb el fitxer de sistema /etc/crontab i els directoris de cron situats a /etc.

3.2.6 Eines gràfiques

Hi ha diverses interfícies que es poden utilitzar per configurar cron de manera gràfica. Els entorns d'escriptori porten els seus paquets respectius per fer la configuració de cron gràficament. Per exemple:

- Amb GNOME podem instal·lar el paquet gnome-schedule. Una vegada instal·lat l'executem des d'*Aplicacions* > *Eines del sistema* > *Tasques programades*.
- Amb KDE tenim el paquet kde-config-cron, que és part del mòdul d'administració de KDE.

La figura 3.5 mostra l'aspecte del planificador de tasques de GNOME visualitzant el següent crontab d'un usuari:

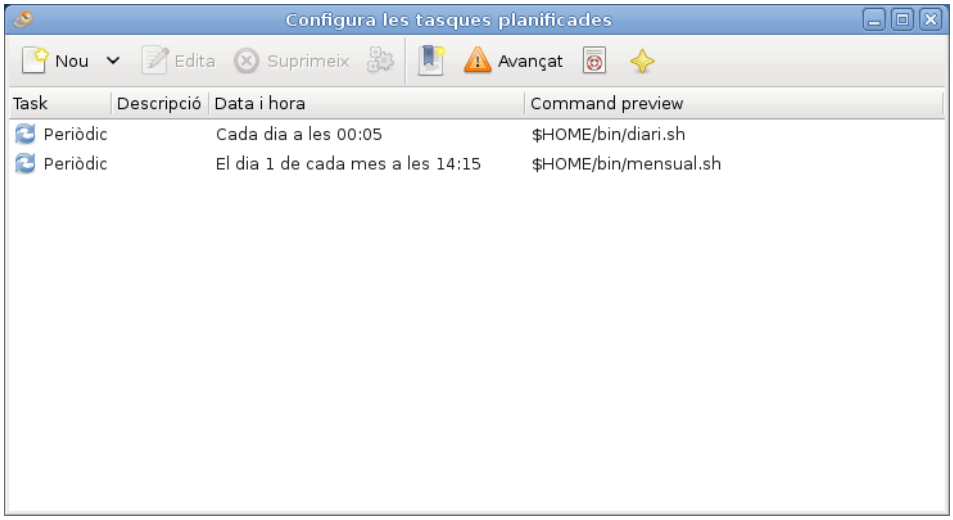
```
5 0 * * * $HOME/bin/diari.sh
15 14 1 * * $HOME/bin/mensual.sh
```

La majoria d'eines gràfiques d'administració del sistema també permeten configurar cron. Per exemple, l'eina d'administració Webmin incorpora un mòdul que permet configurar la planificació de tasques de cron de manera gràfica, en remot i des d'un navegador web.

Vegeu l'apartat "Gestió remota mitjançant Webmin" de la unitat "Administració de processos. Administració remota. Administració de serveis d'impressió".

Si sabem treballar amb cron i configurar-lo des de la línia d'ordres, l'ús de qualsevol d'aquestes eines gràfiques és immediat i explicar el seu funcionament és innecessari.

FIGURA 3.5. Planificador de tasques de GNOME



3.3 Automatització de tasques del sistema

Ubicació dels scripts d'automatització de tasques

En els sistemes de tipus Unix, després de provar i depurar un guió de *shell* d'administració, és habitual moure'l a /usr/local/bin amb el propietari, grup i permisos establerts de manera adequada. En el cas que el guió sigui una utilitat que voleu que estigui disponible per a tots els usuaris del sistema, cal que us assegureu que doneu permís d'execució a *others*.

La programació de *shell scripts* i la utilització d'un planificador de tasques ens facilita l'automatització de moltes de les tasques requerides per al manteniment i la configuració del sistema. A continuació veurem exemples de tasques que típicament requereixen ser automatitzades i utilitzarem el llenguatge de guions Bash per implementar-les.

En primer lloc veurem tasques que s'automatitzen i que normalment també es planifiquen per ser executades de manera periòdica, com són les còpies de seguretat i el manteniment dels fitxers de registre del sistema. Per fer la planificació de l'execució d'aquestes tasques utilitzarem cron.

Després veurem l'automatització de tasques del sistema que en general no requereixen d'un planificador de tasques per ser executades, com ara la gestió d'usuaris, que normalment executem de manera interactiva quan sorgeix la necessitat, i l'automatització de l'inici de serveis que es realitza amb l'arrencada del sistema.

3.3.1 Còpies de seguretat

En informàtica, una còpia de seguretat (en anglès *backup*) és la còpia d'informació que es realitza per tal de ser restaurada en cas de pèrdua de dades o en cas de ser requerida posteriorment.

Les còpies de seguretat poden ser del sistema o de les dades. Les còpies del sistema tenen com a objectiu poder arrencar un sistema després d'un incident de seguretat i per tant realitzen una còpia dels fitxers del sistema operatiu i del programari instal·lat. D'altra banda, les còpies de seguretat de dades pretenen recuperar informació i realitzen còpies de fitxers de dades o bases de dades.

Els administradors de sistemes han de proporcionar els mecanismes per realitzar còpies de seguretat del sistema i de les dades, així com per restaurar-les.

Normalment les dades es copien en un mitjà d'emmagatzemament diferent al de l'origen de les dades, com poden ser discos durs externs, CD-ROM, cintes magnètiques (DAT), etc. i, cada vegada més, s'utilitzen sistemes de còpia de seguretat remota que realitzen les còpies de manera automàtica a través de la xarxa o Internet.

Per tal de decidir quina tecnologia utilitzarem per dur a terme les còpies de seguretat hi ha tres factors clau a tenir en compte: el volum de dades a copiar, el cost econòmic del mitjans d'emmagatzemament i l'operativitat de la solució escollida tant pel que fa al temps de còpia com al de recuperació. Algunes decisions que s'han de prendre són:

- La periodicitat de les còpies. Com més alta és la freqüència més capacitat de recuperació es té.
- El nombre de còpies. Si es realitza més d'una còpia i aquestes es desen en ubicacions separades s'augmenta la seguretat.
- La retenció de les còpies. Com més còpies antigues es guarden es té més capacitat de recuperar informació de fa més temps.
- El model de còpia. Hi ha diversos models (còpia completa, diferencial o incremental) que analitzem més endavant i que són estratègics pel que fa al temps de còpia i recuperació, així com a la quantitat d'espai requerit per guardar les còpies.

Eines

Existeixen molts programaris específics de còpies de seguretat i recuperació. Ara bé, en els sistemes de tipus Unix aquestes tasques també poden fer-se fàcilment

mitjançant scripts planificats amb cron que utilitzen les eines bàsiques que ens proporciona el sistema, com *tar*, *cpio* o el parell *dump/restore*.

En els exemples que mostrarem utilitzarem l'eina *tar* per empaquetar fitxers i l'eina *gzip* per comprimir les dades. Empaquetar vol dir ajuntar dos o més fitxers en un de sol (paquet) i comprimir vol dir agafar el fitxer o paquet i comprimir-lo. A la taula 3.3 es mostra un resum del funcionament de les eines *tar* i *gzip* i l'extensió que se sol posar als fitxers que es generen.

TAULA 3.3. Eines per empaquetar i comprimir dades

Ordre *tar*: empaquetar dades		
Empaquetar	tar cf fitxer.tar /dades	Copia (c) tots els fitxers de /dades i els empaqueta al fitxer (f) fitxer.tar. L'extensió *.tar* li posem per convenció, no és obligatòria. Després de /dades pot haver-hi més noms de directoris o fitxers a empaquetar separats per espais.
Veure el contingut	tar tvf fitxer.tar	Llista (t) i mostra per pantalla (v) les dades empaquetades al fitxer (f) fitxer.tar.
Desempaquetar totes les dades	tar xf fitxer.tar	Extreu (x) totes les dades empaquetades al fitxer (f) fitxer.tar al directori actual.
Desempaquetar algunes dades	tar xvf fitxer.tar f1 f2 ...	Extreu i mostra per pantalla (v) només el fitxer o fitxers especificats (f1, f2...) al directori actual.
Desempaquetar en un directori concret	tar xf fit.tar -C dir tar xf fit.tar f1 f2 -C dir	Extreu les dades al directori especificat després de -C en lloc de al directori actual.
Ordre *gzip*: comprimir dades		
Comprimir	gzip -q fitxer	Comprimeix el fitxer i el reanomena com a fitxer.gz.
Descomprimir	gzip -d fitxer.gz	Descomprimeix fitxer.gz i el deixa com a fitxer.
Ordre *tar* amb *gzip*: empaquetar i comprimir dades. Funciona igual que *tar* però afegint una *z* a les opcions		
Empaquetar i comprimir	tar czf fitxer.tar.gz /dades	Copia (c) i comprimeix amb *gzip* (z) tots els fitxers de /dades i els empaqueta al fitxer (f) fitxer.tar. L'extensió *.tgz* li posem per convenció, no és obligatòria. També podem trobar l'extensió *.tar.gz* en aquests fitxers.
Veure el contingut	tar tzvf fitxer.tar.gz	Llista sense extreure (t) i mostra per pantalla (v) les dades comprimides (z) i empaquetades al fitxer (f) fitxer.tar.gz.
Desempaquetar i descomprimir	tar xzf fitxer.tar.gz	Extreu (x) i descomprimeix (z) les dades del fitxer (f) fitxer.tar.gz.

L'eina *tar* (*tape archiver*) va ser dissenyada originalment per transferir fitxers de disc a una cinta magnètica i viceversa, tot i que ara s'utilitza per empaquetar dades directament sobre qualsevol fitxer o dispositiu. Per exemple, si en el sistema hi ha muntada una unitat de cintes disponible a partir del nom de fitxer de dispositiu /dev/st0 i hi tenim posada una cinta, podem copiar-hi les dades del directori anomenat /dades així:

```
1 tar cf /dev/st0 /dades
```

Una vegada copiades les dades, podem utilitzar les diferents opcions de *tar* que ens permeten veure i extreure les dades.

Estratègies i implementació

L'estratègia més simple per realitzar una còpia de seguretat és copiar totes les dades diàriament. Aquest tipus de còpies són factibles per a sistemes amb un volum de dades petit, però quan la quantitat de dades és gran, cal combinar altres estratègies per dur a terme les còpies. Els tipus de còpies que es poden fer són els següents:

- Copia de seguretat completa. És aquella que inclou tots els fitxers d'un determinat conjunt de dades sense tenir en compte les còpies de seguretat prèvies. També les podem anomenar *còpies de nivell zero*.
- Còpia de seguretat diferencial. És una còpia de tots els fitxers d'un conjunt de dades determinat que ha estat afegit o modificat des de la darrera còpia de seguretat completa. Per restaurar un conjunt de dades, s'ha de restaurar el nivell zero i el diferencial més recent. Un diferencial inclou els mateixos fitxers que el diferencial anterior. Així, en el cas típic, la mida de cada diferencial augmentarà fins que s'executi el proper nivell zero.
- Còpia de seguretat incremental. Es tracta d'una còpia de seguretat de tots els fitxers que hi ha en un determinat conjunt de dades que han canviat des de la darrera còpia de seguretat (de qualsevol tipus). Així, donada una còpia de nivell zero el dilluns, si fem una incremental el dimarts només obtindríem els fitxers nous o canviats des del dilluns. Una incremental feta el dimecres inclouria els fitxers que es van afegir o modificar des del dimarts. Això contrasta amb la diferencial. El que convé recordar és l'ordre de restauració. Per restaurar completament el conjunt de dades, s'ha de recuperar en primer lloc la còpia de nivell zero i després tots els increments en la mateixa seqüència en què van ser creats.

Tenint en compte les definicions que acabem de donar, en funció dels requeriments de cada sistema, les estratègies que s'implementen amb més freqüència són tres:

- Còpia completa diària.
- Còpia completa setmanal i còpia diferencial diària
- Còpia completa setmanal i còpia incremental diària.

Còpia completa diària

La primera estratègia consisteix a fer còpies completes diàries, o, el que és el mateix, fer una còpia de nivell zero de dilluns a diumenge. En la seva forma més simple aquesta còpia és:

```
1 cd /
2 tar czpf /copies/home.tgz home
```

El primer que fem és situar-nos al directori / per evitar posar el símbol / a l'inici dels noms dels fitxers o directoris que volem copiar (l'opció *-C* de *tar* també fa

aquest servei). A continuació fem una còpia del directori /home empaquetant les dades i comprimint-les en un fitxer anomenat home.tgz i ubicat a /copies. L'opció *p* és per preservar els atributs de seguretat (propietari, grup i permisos) dels fitxers que copiem. Assumim que /copies és un sistema de fitxers creat en un disc extern.

Afegim aquestes dues línies en un guió de *shell* que podem ubicar al directori /usr/local/bin i l'anomenem ***completa.sh***. Després planifiquem l'execució d'aquesta tasca perquè s'executi diàriament i triem una hora en la qual no hi hagi activitat en el sistema per dues raons:

- Per garantir la integritat de les dades que es copien.
- Perquè el procés de còpia no afecti al rendiment global del sistema.

La línia del crontab podria quedar així:

En tots els exemples, *root* és el propietari dels guions de *shell* i l'usuari que els planifica i els executa amb cron.

```
0 2 * * */usr/local/bin/completa.sh
```

Fixeu-vos que si no fem res més, la retenció de les còpies amb aquest sistema és només d'un dia, ja que cada dia estem guardant la còpia completa amb el mateix nom, i, per tant, estem sobreescrivint el fitxer. Generalment voldrem mantenir les còpies de més d'un dia d'antiguitat.

En el nostre cas, podem fer una modificació molt senzilla per mantenir sempre les set darreres còpies realitzades. Consisteix a afegir un número al nom del fitxer que indiqui el dia de la setmana en què es fa la còpia, per exemple:

```
cd /
tar czpf /copies/home$(date +%u).tgz home
```

Amb aquesta modificació, el nom del fitxer generat conté el resultat de l'execució de l'ordre `date +%u` per indicar el dia de la setmana (home1.tgz els dilluns, home2.tgz els dimarts, etc.). Així, cada dia sobreescriurem la còpia de fa una setmana i mantindrem les dels sis dies anteriors.

Còpia completa setmanal i diferencial diària

La segona estratègia consisteix a fer una còpia setmanal completa (per exemple el diumenge) i una còpia de nivell 1 (diferencial) la resta de dies de la setmana. En aquesta estratègia, a la còpia de nivell 0 realitzada el diumenge que implementem com acabem de veure, li hem d'afegir la còpia diferencial diària, que en la seva forma més simple és:

```
find /home -type f -newer /copies/home0.tgz >/tmp/llista
tar czpTf /tmp/llista /copies/home$(date +%u).tgz
```

L'ordre *find* permet cercar fitxers que compleixen unes característiques determinades amb diversos criteris de selecció.

Veiem que ara estem utilitzant l'ordre *find* per generar una llista de tots aquells fitxers de /home que siguin més nous que el fitxer que conté la còpia completa. En el *tar* hem afegit l'opció *T* per llegir els fitxers a copiar des de la llista generada en el pas anterior.

Afegim les dues línies a un *shell script* que anomenem ***diferencial.sh*** i planifiquem la còpia completa perquè s'executi únicament els diumenges i la diferencial perquè s'executi la resta de dies de la setmana. Per exemple:

```
0 2 * * 0/usr/local/bin/completa.sh
0 2 * * 1-6/usr/local/bin/diferencial.sh
```

Còpia completa setmanal i incremental diària

La tercera estratègia consisteix a fer una còpia completa setmanal (per exemple el diumenge) i una incremental la resta de dies de la setmana (nivell 1 el dilluns, nivell 2 el dimarts, nivell 3 el dimecres, etc.). En aquesta estratègia, a la còpia realitzada el diumenge de nivell 0 que implementem com en els altres casos, li hem d'afegir la còpia incremental diària, que en la seva forma més simple és:

```
find /home -type f -mtime 1 > /tmp/llista
tar czpTf /tmp/llista /copies/home$(date +%u).tgz
```

En aquest cas estem utilitzant l'ordre *find* per generar una llista de tots aquells fitxers de /home que s'han modificat en les darreres 24 hores. El *tar* s'ha d'interpretar igual que en el cas anterior.

De manera anàloga al cas anterior, afegim les dues línies a un guió de *shell* que anomenem ***incremental.sh*** i planifiquem la còpia completa per als diumenges i la incremental per a la resta de dies. Per exemple:

```
0 2 * * 0/usr/local/bin/completa.sh
0 2 * * 1-6/usr/local/bin/incremental.sh
```

Millora dels guions de shell

Podem millorar els procediments per implementar les diferents estratègies de còpia (completa, diferencial i incremental) si en els *shell scripts* utilitzem variables, afegim control d'errors, etc. A continuació, i a tall d'exemple, veiem una versió millorada del *shell script* que hem anomenat *completa.sh*:

Vegeu exemples de guions de *shell* per copiar un servidor web i una base de dades local a la secció d'adreces d'interès del web del mòdul.

```
#/bin/bash
# completa.sh
# Còpia completa de dades
#
# Directoris a copiar, no incloem la / inicial
DIRS="etc home var"
# Directori de destinació de la còpia
BACKUPDIR="/copies"
# Nom del fitxer de la còpia
AVUI=$(date +"%Y-%m-%d")
FITXER="completa_$AVUI.tar.gz"
# Nombre de dies per guardar còpies antigues
ANTIGUES=14
#
# Missatge d'inici
echo "$(date +%X) Inici del procediment de còpia."
#
# Comprovació que l'eina tar hi és
TAR=$(which tar)
```

```
20 if [ -z "$TAR" ]; then
21   echo "Error: no s'ha trobat tar."
22   exit 1
23 fi
24 # Comprovació que el directori de còpies hi és
25 if [ ! -d $BACKUPDIR ] ; then
26   echo "Error: no s'ha trobat $BACKUPDIR"
27   exit 1
28 fi
29 # Fem la còpia
30 $TAR -zcPf $BACKUPDIR/$FITXER -C / $DIRS
31 if [ $? -ne 0 ]
32 then
33 # El tar ha fallat
34   echo "Error: hi ha hagut algun error en fer la còpia."
35   exit 1
36 fi
37 # Esborrem les còpies anteriors a $ANTIGUES
38 find $BACKUPDIR/ -name "*.gz" -type f -mtime +$ANTIGUES -delete
39 if [ $? -ne 0 ]
40 then
41   echo "Error: eliminant les còpies antigues."
42   exit 1
43 fi
44 # Missatge de finalització
45 echo "$(date +%X) Final correcte de la còpia."
46 exit 0
```

Aquest guió de *shell* permet guardar tantes copies com s'hagin definit a la variable *ANTIGUES*. El nom del fitxer de còpia que es genera és: completa_AAA-MM-DD.tar.gz. Per exemple, si fem una còpia el dia 11 de març de 2012 tindrem un fitxer que es dirà *completa_2012-03-11.tar.gz*.

Planifiquem l'execució diària del guió de *shell* a l'hora que convingui. Per exemple:

```
1 0 2 * * */usr/local/bin/completa.sh
```

Si volem guardar un registre de l'activitat d'aquest *shell script* hem de redirigir les sortides estàndard i d'errors a un fitxer. Per exemple:

```
1 0 2 * * * /usr/local/bin/completa.sh >>fitxer 2>>fitxer
```

El fitxer de registre usualment el crearem en el directori de registre del sistema, /var/log. En aquest cas podem crear un directori anomenat /var/log/local i desar allà els fitxers de registre dels procediments de còpia i d'altres procediments locals de manteniment del servidor. Per al guió de *shell* completa.sh, el nom del fitxer de sortida pot ser /var/log/local/completa.log.

3.3.2 Manteniment dels fitxers de registre

Els fitxers de registre o de *log* (de l'anglès *log files*) són fitxers de traça que generen el sistema o les aplicacions per enregistrar esdeveniments i deixar constància de les accions que es fan. Són molt útils per veure l'activitat d'un servei o aplicació i per diagnosticar problemes.

syslog

syslog és un estàndard de facto per a l'enviament de missatges de registre en una xarxa informàtica IP. Va ser desenvolupat el 1980 per Eric Allman com a part del projecte Sendmail. Posteriorment es va comprovar que era molt útil i d'altres aplicacions també van començar a usar syslog.

En els sistemes Unix i derivats hi ha un servei anomenat **syslog** que és l'encarregat d'enregistrar l'activitat del sistema operatiu i de les aplicacions que utilitzen aquest servei, com ara el servei cron (planificació de treballs) o lpr (subsistema d'impressió) entre d'altres.

Generalment, tots els fitxers de registre del sistema, siguin o no gestionats per syslog, se solen emmagatzemar al directori `/var/log/`. Encara que la majoria de fitxers de registre són de text i els podem veure amb qualsevol editor, en podem trobar algun d'especial que no desi les seves dades en aquest format, com ara els fitxers `/var/log/wtmp` i `/var/log/btmp`, que són els registres d'entrada d'usuaris en el sistema i d'entrades errònies respectivament. Per veure aquests dos fitxers, podem utilitzar les ordres `last` i `lastb`. Si tinguéssim configurats aquests registres en algun altre fitxer, també els podríem veure passant el paràmetre amb l'ordre `last -f nom_fitxer`.

La mida dels fitxers de registre sempre va creixent, ja que les dades enregistrades es van afegint al final del fitxer. Per evitar saturar el disc, cal automatitzar alguna tasca de manteniment d'aquests fitxers.

rsyslogd

Hi ha diverses implementacions de syslog. La distribució Linux Debian utilitza rsyslogd. Executeu `man rsyslogd` per veure el funcionament i la personalització d'aquest servei.

Normalment s'utilitza un sistema de rotació de registres, que consisteix a anar comprimint cada cert temps aquests fitxers i a desar-los fins que tinguin una antiguitat determinada. Per exemple, es poden comprimir cada setmana i desar només els d'un o dos mesos anteriors. Segons el servidor que estiguem administrant haurem de tenir en compte la legalitat vigent, que en alguns casos obliga a conservar els fitxers de registre durant un període de temps determinat.

Consola de logs

Podem configurar una consola del sistema per veure tots els registres que es van generant, afegint la línia *.* /dev/ttySX (X és la consola en què volem veure els registres) al fitxer /etc/rsyslog.conf i reiniciant el dimoni rsyslogd.

Hi ha programes que ens permeten fer ús del sistema de *log* que podem utilitzar en els nostres *shell scripts* per crear els nostres propis fitxers de registre o, si escau, manipular manualment els del sistema: amb `logger` podem escriure en el syslog del sistema i amb `savelog` podem desar i opcionalment comprimir els fitxers de registre.

Gestió dels fitxers de registre amb logrotate

El programa `logrotate` està dissenyat per facilitar les tasques de gestió de registres. Permet la rotació automàtica, compressió, eliminació i l'enviament per correu dels fitxers. Cada fitxer de registre pot ser tractat diàriament, setmanal, mensual o quan es fa massa gran.

Executeu `man logrotate` per veure la pàgina de manual del programa.

Normalment, logrotate s'executa com una tasca diària de cron. Al directori `/etc/cron.daily` hi ha un guió de *shell* anomenat `logrotate` que conté les línies següents:

```
#!/bin/sh
test -x /usr/sbin/logrotate || exit 0
/usr/sbin/logrotate /etc/logrotate.conf
```

El fitxer `/etc/logrotate.conf` és on es configura el mode d'operació de logrotate. Per defecte té un contingut similar al següent (els comentaris originals són en anglès):

```
# rotar els fitxers de registre setmanalment
weekly
# mantenir 4 setmanes de fitxers antics
rotate 4
# crear fitxers nous buits després de rotar
create
# comprimir els fitxers
#compress
# directori on desar fitxers de configuració específica de les aplicacions
include /etc/logrotate.d
# configuració per als serveis wtmp i btmp
/var/log/wtmp {
  missingok
  monthly
  create 0664 root utmp
  rotate 1
}
/var/log/btmp {
  missingok
  monthly
  create 0660 root utmp
  rotate 1
}
# altres logs específics del sistema es poden configurar aquí
```

Les aplicacions poden deixar els seus propis fitxers al directori /etc/logrotate.d per especificar les seves opcions. Per exemple, el servei CUPS instal·la per defecte aquest fitxer:

```
/var/log/cups/*log {
  daily
  missingok
  rotate 7
  sharedscripts
  postrotate
    if [ -e /var/run/cups/cupsd.pid ]; then
      invoke-rc.d --quiet cups force-reload > /dev/null
      sleep 10
    fi
  endscript
  compress
  notifempty
  create 640 root lpadmin
}
```

Gestió dels fitxers de registre amb guions de shell propis

El programa logrotate ens dóna la possibilitat d'automatitzar la gestió dels fitxers de registre del sistema i de les aplicacions, però la manipulació d'aquests fitxers també es pot fer amb guions de *shell* creats per nosaltres amb aquest propòsit.

A continuació veiem un exemple senzill per esborrar els registres del fitxer de *log* del sistema anomenat `/var/log/messages`. El programa esborra el fitxer i es queda amb 50 línies o les que indiqui l'usuari en la línia d'ordres.

```
#!/bin/bash
# borralog.sh
# Neteja fitxers de registre del sistema
#
# Definició de variables
LOG_DIR=/var/log
FITXER=messages
ROOT_UID=0
LINIES=50
# Definició de funcions
missatge () {
  DATA=$(date +"%b %x - %X")
  echo "$0: $DATA ---> $1"
}
############## Programa principal ##############
# Missatge d'inici
missatge "Inici esborrat de logs."
# Comprovem que som root.
if [ "$UID" -ne "$ROOT_UID" ]; then
  missatge "Error: heu de ser root."
  exit 1
fi
# Establir el nombre de línies que cal preservar
case "$1" in
  "")
    linies=$LINIES
    ;;
  *[!0-9]*)
    echo missatge "Error: $1 argument invàlid."
    exit 1
    ;;
  *)
    linies=$1
    ;;
esac
# Canviar al directori de logs
cd /var/log || {

  missatge "Error: no puc anar a $LOG_DIR." >&2
  exit 1;
}
# Guardar les línies que volem del fitxer de log
tail -n $linies $FITXER > $FITXER.tmp
mv $FITXER.tmp $FITXER
# Missatge de final
missatge "Final de l'esborrat de logs."
exit 0
```

El guió de *shell* anterior pot ser executat de manera interactiva o bé es pot planificar amb cron per ser executat periòdicament. Per exemple:

```
30 1 * * */usr/loc/bin/borralog.sh
```

Si volem guardar el registre de l'activitat d'aquest *shell script*, hem de redirigir les sortides estàndard i d'errors a un fitxer. Per exemple:

```
30 1 * * * /usr/local/bin/borralog.sh >>fitxer 2>>fitxer
```

El nom del fitxer de sortida podria ser: `/var/log/local/borralog.log`.

3.3.3 Gestió d'usuaris

Els usuaris dels sistemes de tipus Unix tenen assignat un compte o nom d'usuari que els identifica. Cada compte d'usuari també té una contrasenya que els permet accedir al sistema anomenada clau d'accés (*password*). A més a més, a Unix cada usuari pertany com a mínim a un grup d'usuaris, que anomenem el seu *grup primari* o *principal*. En cas de ser necessari, un usuari pot pertànyer a més d'un grup i en aquest cas es diu que són els seus grups secundaris o addicionals.

Alguns usuaris i grups es creen en el moment d'instal·lar el sistema operatiu, per exemple: *root*, *bin*, *sys*, *mail*, *uucp*, etc.

Una de les funcions de l'administrador del sistema és donar d'alta els grups i els usuaris al sistema. Hi ha un usuari especial anomenat superusuari o *root*, que es crea amb la instal·lació del sistema operatiu i disposa dels màxims drets al sistema.

> És convenient fer servir el superusuari només quan fem funcions d'administració de sistemes que requereixen els drets de *root*. En tots els altres casos és més adient emprar un altre compte. Per evitar l'ús de *root*, també podem configurar `sudo` per permetre que un compte d'usuari normal tingui drets per fer determinades tasques d'administració, com ara gestionar usuaris.

sudo

sudo és una ordre que permet que els usuaris executin ordres que inicialment només pot executar *root*. Al fitxer de configuració /etc/sudoers és on s'indica quins usuaris poden executar quines ordres.

Hi ha moltes eines que permeten fer la gestió d'usuaris i grups des d'una interfície gràfica, per exemple l'eina d'administració Webmin o les interfícies dels escriptoris de GNOME i KDE, entre d'altres. Alternativament, per treballar des de línia d'ordres, en algunes distribucions de Linux, entre d'elles a Debian, hi ha els programes useradd, userdel i usermod per a la gestió d'usuaris, i groupadd, groupdel i groupmod per a la gestió de grups.

La gestió d'usuaris pot donar molta feina tenint en compte que, per exemple, ens podem trobar que volem donar d'alta grups de més de 20 persones. En aquests casos el més habitual és fer guions de *shell* que permeten automatitzar aquesta tasca i donar d'alta usuaris nous de forma massiva a partir d'un fitxer de text amb un format especial.

A tall d'exemple mostrem un guió de *shell* que dóna d'alta tots els usuaris descrits en un fitxer de text amb format CSV. Els camps de les línies del fitxer d'usuaris són els següents:

```
1 nom_usuari,grup_primari,contrasenya_inicial
```

Fitxer CSV

Els fitxers CSV (de l'anglès comma-separated values) són un tipus de document que representa dades en forma de taula, en la qual les columnes se separen per comes (o punt i coma) i les files per salts de línia.

Per exemple, si volem donar d'alta la usuària *angela* i l'usuari *david* amb grup primari *alumnat* i contrasenya *123* i *456* respectivament, i l'usuari *marius* amb grup *professorat* i contrasenya *789*, el fitxer CSV ha de contenir les línies:

```
1 angela,alumnat,123
2 david,alumnat,456
3 marius,professorat,789
```

El codi del programa és el següent:

```
#/bin/bash
# altausu.sh
# Dóna d'alta usuaris a partir de la informació d'un
# fitxer CSV. El format de les línies del fitxer és:
# nom,grup,contrasenya
#
# Nom del fitxer CSV
FITXER=usuaris.csv
ROOT_UID=0
#
# Comprovacions
if [ "$UID" -ne "$ROOT_UID" ]; then
  echo "Error: heu de ser root."
  exit 1
fi
if [ ! -f "$FITXER" ]; then
  echo "Error: no s'ha trobat $FITXER."
  exit 1
fi
# Inici del procés
for linia in $(cat $FITXER); do
  echo $linia
  USUARI=$(echo $linia | cut -f1 -d",")
  GRUP=$(echo $linia | cut -f2 -d",")
  PASSWD=$(echo $linia | cut -f3 -d",")
  echo "Creant l'usuari $USUARI..."
  /usr/sbin/useradd -m -g $GRUP $USUARI
  echo "Assignant contrasenya a $USUARI..."
  echo "$USUARI:$PASSWD" | chpasswd
done
```

Aquest *shell script* automatitza i simplifica considerablement la tasca de creació d'usuaris. Podem millorar el codi del programa afegint, si cal, el tractament de més camps del fitxer d'informació d'usuaris (directori de treball, *shell* d'inici, etc.), fent control d'errors, guardant els missatges de sortida en un fitxer de registre, etc.

L'ordre *chpasswd* fa el mateix que *passwd*, però pot ser utilitzada dins d'un guió de *shell* de manera no interactiva.

3.3.4 Engegada i aturada automàtica de serveis

Durant el procés d'arrencada del sistema s'inicien els processos que anomenem *dimonis* (*daemons* en anglès). Generalment tots els dimonis tenen un guió de *shell* situat al directori `/etc/init.d/` que ens permet iniciar-los, aturar-los o veure el seu estat d'execució. La majoria d'aquests scripts utilitzen un programa anomenat `start-stop-daemon` que ens proporciona el sistema operatiu i que serveix per al tractament d'aquests processos.

En administrar un servidor es pot donar el cas que ens haguem de dissenyar els nostres propis dimonis per fer alguna tasca concreta. Al directori on se situen tots els guions de *shell* dels dimonis, se'n sol trobar un d'exemple anomenat `/etc/init.d/skeleton` perquè el puguem utilitzar quan en necessitem configurar un de nou. El codi del dimoni d'exemple és similar al següent:

Vegeu més informació sobre els processos que anomenem *dimonis* a l'apartat "Dimonis" de la unitat "Administració de processos. Administració remota. Administració de serveis d'impressió".

```
#!/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin: /usr/sbin:/usr/bin DAEMON=/usr/
    sbin/daemon
NAME=daemon
DESC="some daemon"
test -x $DAEMON || exit 0
set -e
case "$1" in
  start)
    echo -n "Starting $DESC: $NAME"
    start-stop-daemon --start --quiet --pidfile \
    /var/run/$NAME.pid --exec $DAEMON echo "."
    ;;
  stop)
    echo -n "Stopping $DESC: $NAME "
    start-stop-daemon --stop --quiet --pidfile \
    /var/run/$NAME.pid --exec $DAEMON echo "."
    ;;
  restart|force-reload)
    echo -n "Restarting $DESC: $NAME"
    start-stop-daemon --stop --quiet --pidfile \
    /var/run/$NAME.pid --exec $DAEMON
    sleep 1
    start-stop-daemon --start --quiet --pidfile \
    /var/run/$NAME.pid --exec $DAEMON echo "."
    ;;
  *)
    N=/etc/init.d/$NAME echo " Usage: $N {start|stop| \
    restart|force-reload}" >&2
    exit 1
    ;;
esac
exit 0
```

La primera línia indica quin és el *shell* d'execució. Gairebé en tots els scripts d'inici trobarem el shell /bin/sh. Com que Bash és compatible amb sh, les ordres i els programes escrits per a sh poden ser executats amb Bash sense cap modificació. Però al revés no és cert. Per tant, si necessitem escriure un *shell* d'inici amb característiques específiques del llenguatge Bash hem de canviar la primera línia i posar */bin/bash*.

En les variables declarades a l'inici del guió de *shell* especifiquem quin *PATH* és necessari per al procés del dimoni, el programa que executarem (*DAEMON*), el nom que li donem (NAME), que ha de ser igual que el nom del *shell script*, i la descripció (*DESC*).

En arrencar el dimoni la única cosa que fem és escriure al directori /var/run/ un fitxer amb el PID del procés. En aturar-lo, es buscarà aquest PID i s'enviarà el senyal d'acabament al procés corresponent.

Trobarem guions de *shell* preparats per a fer moltes més operacions amb el nostre dimoni, però, com a mínim, tots han de tenir aquesta estructura.