Administració local

Josep Jorba Esteve

PID_00238577

UOC
Universitat Oberta de Catalunya

Índex

Introducció

Una de les primeres tasques amb què s'haurà d'enfrontar l'administrador serà la gestió dels recursos locals presents en el sistema a administrar. En aquest mòdul veurem algunes d'aquestes tasques d'administració bàsiques, i alguns dels aspectes de personalització i rendiment dels recursos.

Abans de començar amb els aspectes més pràctics de l'administració, revisarem algunes de les eines bàsiques de què disposarà l'administrador (algunes, com els *shell scripts,* ja les hem revisades prèviament).

Posteriorment, analitzarem el procés d'arrencada d'un sistema GNU/Linux, que ens farà comprendre l'estructura inicial del sistema i la seva relació amb els serveis que proporciona.

A continuació, aprendrem com podem obtenir una visió general de l'estat actual del sistema per mitjà dels diferents procediments i comandes de què disposem per a avaluar les parts del sistema. D'aquesta manera, podrem prendre decisions d'administració si detectem algun error o deficiència de rendiment, o la falta d'algun recurs.

Nota

L'administració local engloba moltes tasques variades, que potser són les més utilitzades per l'administrador en el seu treball diari.

Un dels principals punts de l'administració és la gestió d'usuaris, ja que qualsevol configuració de la màquina estarà destinada perquè la puguin utilitzar. Veurem com podem definir nous usuaris en el sistema i controlar el seu nivell d'accés als recursos.

Quant als perifèrics del sistema, com discos i impressores, disposem de diferents possibilitats de gestió, ja sigui via diferents servidors (el cas de la impressió) o diferents sistemes d'arxius que podem tractar, i també algunes tècniques d'optimització del rendiment dels discos.

També examinarem el problema de l'actualització del sistema, i també la nova incorporació de programari d'aplicació i com el podem fer disponible als usuaris. Així mateix, analitzarem el problema d'executar treballs temporitzats en el sistema.

En el taller final examinarem l'avaluació d'estat d'una màquina, seguint els punts vistos en aquest mòdul, i durem a terme algunes de les tasques d'administració bàsiques descrites. En el desenvolupament de la unitat comentarem algunes comandes i, posteriorment, en el taller, en veurem algunes amb més detall respecte al funcionament i les opcions.

1. Eines bàsiques per a l'administrador

L'administrador de sistemes GNU/Linux s'ha d'enfrontar, diàriament, a una gran quantitat de tasques. En general, en la filosofia UNIX no hi sol haver una única eina per a cada tasca o una sola manera de fer les coses. El més comú és que els sistemes UNIX proporcionin una gran quantitat d'eines més o menys simples per a afrontar les diferents tasques.

Serà la combinació de les eines bàsiques, cada una amb una tasca molt definida, la que ens donarà la possibilitat de solucionar un problema o tasca d'administració.

En aquest apartat veurem diferents grups d'eines, identificarem algunes de les seves funcions bàsiques i veurem diversos exemples dels seus usos. Començarem per examinar alguns estàndards del món GNU/Linux, que ens permetran trobar algunes de les característiques bàsiques que esperem de qualsevol distribució de GNU/Linux. Aquests estàndards, com l'LSB (o Linux Standard Base) [Linc] i l'FHS (Filesystem Hierarchy Standard) [Linb], ens parlen d'eines que esperem trobar disponibles, d'una estructura comuna per al sistema de fitxers, i també de diferents normes que s'han de complir perquè una distribució sigui considerada un sistema GNU/Linux i mantingui regles comunes per a la compatibilitat entre aquests estàndards.

En l'automatització de tasques d'administració se solen utilitzar ordres agrupades en *shell scripts* (també anomenats *guions de comandes*), mitjançant llenguatges interpretats pel *shell* (intèrpret de comandes) del sistema. En la programació d'aquests *shell scripts* podem unir les ordres del sistema amb estructures de control de flux, i així disposar d'un entorn de prototip ràpid d'eines per a l'automatització de tasques.

Un altre esquema habitual és la utilització d'eines de compilació i depuració de llenguatges d'alt nivell (com per exemple C). En general, seran utilitzades per l'administrador per a generar nous desenvolupaments d'aplicacions o eines, o per a incorporar al sistema aplicacions que vinguin com a codi font i s'hagin d'adaptar i compilar.

També analitzarem l'ús d'algunes eines gràfiques respecte a les habituals de la línia de comandes. Aquestes eines solen facilitar les tasques a l'administrador, però el seu ús és limitat, ja que depenen fortament de la distribució de GNU/Linux, o fins i tot de cada versió. Tot i així, hi ha algunes eines útils que són compartides entre distribucions.

Finalment, analitzarem un grup d'eines imprescindibles per a mantenir el sistema actualitzat: les eines de gestió de paquets. El programari servit en la distribució GNU/Linux, o incorporat posteriorment, se sol oferir en unitats denominades *paquets*, que inclouen els arxius d'un determinat programari, més passos necessaris per a la preparació de la instal·lació, la configuració posterior o, si és el cas, l'actualització o desinstal·lació d'un determinat programari. I cada distribució sol aportar programari de gestió per mantenir les llistes de paquets instal·lats, o per instal·lar, i també el control de les versions existents, o possibilitats diverses d'actualització per mitjà de diferents fonts d'origen (repositoris de paquets).

1.1. Eines gràfiques i línies de comandes

Hi ha un gran nombre d'eines; en aquest mòdul i en els següents examinem una petita porció de les eines d'administració que són proporcionades per tercers de manera independent a la distribució o pel distribuïdor mateix del sistema GNU/Linux.

Aquestes eines poden cobrir més o menys aspectes de l'administració d'una tasca concreta, i es poden presentar amb múltiples interfícies diferents: ja siguin eines de línia d'instruccions amb múltiples opcions o fitxers de configuració associats, o eines textuals amb algun tipus de menús elaborats, o bé eines gràfiques amb interfícies més adequades per al maneig d'informació, o assistents que automatitzin les tasques, o bé interfícies web d'administració.

Funcionalitat

Les eines gràfiques d'administració no solen oferir una funcionalitat completa, i és interessant conèixer quins són els efectes de les seves accions.

Tot això ens ofereix un gran nombre de possibilitats amb vista a l'administració, però sempre n'hem de valorar la facilitat d'ús juntament amb les prestacions i els coneixements que tingui l'administrador que es dedica a aquestes tasques.

Les tasques habituals de l'administrador GNU/Linux poden implicar treballar amb diferents distribucions (per exemple, les que comentarem, Fedora [Fed] o Debian [Debb], o qualsevol altra), o fins i tot es pot treballar amb variants comercials de d'altres UNIX. Això comporta que hàgim d'establir una certa manera de treballar que ens permeti fer les tasques de la mateixa manera en els diferents sistemes.

Per aquesta raó, en els diferents apartats intentarem destacar tots aquells aspectes més comuns, i les tècniques d'administració seran fetes majoritàriament a baix nivell, mitjançant una línia d'instruccions o amb edició de fitxers de configuració associats.

Qualsevol de les distribucions de GNU/Linux sol aportar eines del tipus línia d'instruccions, textual, o en particular, gràfiques, que complementen les anteriors i simplifiquen, en més o menys mesura, l'administració de les tasques [Sm]. Però cal tenir en compte diverses puntualitzacions:

a) Aquestes eines són una interfície més o menys elaborada de les eines bàsiques de línia d'instruccions i els fitxers de configuració corresponents.

b) Normalment no ofereixen totes les prestacions o configuracions que es poden fer a baix nivell.

c) Els errors poden no gestionar-se bé, o simplement proporcionar missatges del tipus "la tasca no s'ha pogut fer".

d) L'ús d'aquestes eines oculta, de vegades completament, el funcionament intern del servei o tasca. Comprendre bé el funcionament intern és un coneixement bàsic per a l'administrador, i més si ha de desenvolupar tasques de correcció d'errors o optimització de serveis.

e) Aquestes eines són útils en la millora de la producció. Una vegada que l'administrador té els coneixements adequats, pot gestionar amb aquestes de manera més eficaç les tasques rutinàries i automatitzar-les.

f) O també el cas contrari, la tasca pot ser tan complexa, o necessitar tants paràmetres, o generar tantes dades, que es torna impossible controlar-la de manera manual. En aquests casos, les eines d'alt nivell poden ser molt útils i tornar practicables algunes tasques que d'una altra manera són difícils de controlar. Per exemple, dins d'aquesta categoria entrarien les eines de visualització, monitoratge i resum d'activitats o serveis complexos.

g) En l'automatització de tasques, aquestes eines (de nivell més alt) poden no ser les més adequades: poden no haver estat pensades per als passos que cal fer, o bé fer-ho d'una manera no eficaç. Un cas concret pot ser la creació d'usuaris; una eina visual pot ser molt atraient, per la manera d'introduir les dades, però què succeeix quan en lloc d'introduir un o pocs usuaris volem introduir una llista de desenes o centenars? L'eina, si no està preparada, es torna totalment ineficient.

h) Finalment, els administradors solen voler personalitzar les seves tasques utilitzant les eines que consideren més còmodes i fàcils d'adaptar. En aquest aspecte, sol ser habitual la utilització de les eines bàsiques de baix nivell i la utilització de *shell scripts* per a combinar-les de manera que formin una tasca.

Hem de saber valorar aquestes eines extra segons el valor afegit que tinguin per a les nostres tasques.

Podem donar a aquestes eines un ús casual (o quotidià), si tenim els coneixements suficients per a tractar els errors que es puguin produir, o bé amb l'objectiu de facilitar algun procés per al qual hagi estat pensada l'eina, però sempre controlant les tasques que implementem i el coneixement tècnic subjacent.

1.2. Documents d'estàndards

Els estàndards, ja siguin genèrics del món UNIX o particulars de GNU/Linux, ens permeten seguir uns criteris bàsics, pels quals ens guiem en el moment d'aprendre o fer una tasca, i que ens proporcionen informació bàsica per a començar la nostra feina.

En GNU/Linux ens podem trobar estàndards com l'FHS (Filesystem Hierarchy Standard) [Lin04b], que ens explica què podem trobar (o on ho podem buscar) en l'estructura del sistema de fitxers del nostre sistema. O l'LSB (Linux Standard Base), que ens comenta diferents components de programari que solem trobar en els sistemes [Lin11c].

En l'estàndard **FHS** (Filesystem Hierchachy Standard) es descriu l'estructura en arbre del sistema de fitxers principal (/), en què s'especifica l'estructura dels directoris i els principals fitxers que contindran. Aquest estàndard és usat en més o menys mesura també per als UNIX comercials, en els quals al principi hi va haver moltes diferències que van fer que cada fabricant canviés l'estructura al seu gust. L'estàndard pensat en origen per a GNU/Linux es va fer per a normalitzar aquesta situació i evitar canvis dràstics. Tot i així, l'estàndard és seguit amb diferents graus; la majoria de distribucions segueixen en un alt percentatge l'FHS, i fan canvis menors o aporten fitxers o directoris que no hi havia en l'estàndard.

L'estàndard FHS

El Filesystem Hierchachy Standard és una eina bàsica per al coneixement d'una distribució, que ens permet conèixer l'estructura i funcionalitat del sistema d'arxius principal del sistema.

Vegeu *FHS* a http://www.pathname.com/fhs

També podeu consultar les activitats del grup FHS a l'aderça següent:

http://www.linuxfoundation.org/collaborate/workgroups/lsb/fhs

Un esquema bàsic de directoris podria ser:

- **/bin**: utilitats de base del sistema, normalment programes emprats pels usuaris, des de les instruccions bàsiques del sistema (com /bin/ls, que fa una llista de directoris), passant pels intèrprets d'ordres (/bin/bash), etc.

- **/boot**: arxius estàtics relacionats amb el *boot loader*, necessaris durant l'arrencada del sistema previs a l'arrencada del *kernel*. El qual pot trobar-se a / (l'arrel) o en aquest mateix directori; per exemple, */boot/vmlinuz*.

- **/dev**: aquí trobem fitxers especials que representen els dispositius possibles en el sistema. L'accés als perifèrics en sistemes UNIX es fa com si fossin fitxers. Podem trobar fitxers com /dev/console, /dev/modem, /dev/mouse, /dev/cdrom, /dev/floppy..., que solen ser enllaços a dispositius més específics del tipus de controlador o interfície que utilitzen els dispositius, com /dev/mouse, enllaçat a /dev/psaux, que representa un ratolí de tipus PS2, o /dev/cdrom a /dev/hdc, un CD-ROM que és un dispositiu del segon connector IDE i mestre. Aquí trobem els dispositius IDE com /dev/hdx, els SCSI/sata /dev/sdx... amb *x* que varia segons el número de dispositiu. Cal esmentar que les últimes distribucions, respecte als dispositius de disc, suporten en el nucli emulació de dispositius IDE com si fossin SCSI; per això és corrent veure avui dia tots els discos com a dispositius de tipus /dev/sdx. Aquí es pot comentar que, en el seu inici, aquest directori era estàtic, amb els fitxers predefinits, o configurats en determinats moments. Actualment, s'utilitzen tecnologies dinàmiques (com *hotplug*, i principalment *udev*), que permeten detectar dispositius i crear els arxius /dev dinàmicament, bé a l'inici del sistema o durant l'execució, amb la inserció de dispositius extraïbles.

- **/etc**: fitxers de configuració. La majoria de tasques d'administració necessitaran examinar o modificar els fitxers continguts en aquest directori. Per exemple /etc/passwd conté part de la informació dels comptes dels usuaris del sistema. Es recomana que la configuració de serveis es desi en subdirectoris de */etc*, i no en el mateix */etc* directament.

- **/home**: conté els comptes dels usuaris, és a dir, els directoris personals de cada usuari. Encara que FHS no ho força, per tant és possible que les distribucions o l'administrador creïn directoris diferents (per exemple, */users* o altres). Els programes o eines d'administració no haurien de pressuposar que els comptes d'usuari es troben en aquest directori.

- **/lib**: les biblioteques del sistema, compartides pels programes d'usuari, ja siguin estàtiques (extensió .a) o dinàmiques (extensió .so). Per exemple, la biblioteca C estàndard, en fitxers libc.so o libc.a. També, en particular, se solen trobar els mòduls dinàmics del nucli Linux, en /lib/modules.

- **/mnt**: punt per a muntar (instrucció *mount*) sistemes de fitxers de manera temporal. Històricament s'havia usat per a muntar dispositius removibles,

FHS & LSB

Poden trobar-se les últimes especificacions en estat beta a:
http://www.linuxbase.org/betaspecs/fhs/
http://www.linuxbase.org/betaspecs/lsb/

però actualment es prefereix per a muntar sistemes de fitxers de manera temporal.

- **/media**: per a punt de muntatge habitual de dispositius extraïbles.

- **/opt**: s'hi sol col·locar el programari afegit al sistema posterior a la instal·lació. Una altra instal·lació vàlida és en /usr/local.

- **/run**: usat per a incorporar dades en temps d'execució, per diferents programes, especialment per a emmagatzemar informació des de l'arrencada del sistema. Típicament s'havia utilitzat */var/run* per a aquest objectiu, però s'està migrant progressivament a aquest directori. Una de les funcions és emmagatzemar PID (identificadors de procés) de programes en execució. En cas d'alguns serveis, sol disposar-se d'un subdirectori per a cadascun. També en sistemes amb systemd present, el muntatge de dispositius removibles s'ha traslladat a */run/media*.

- **/sbin**: utilitats de base del sistema. Solen ser instruccions reservades a l'administrador (*root*). Per exemple, /sbin/fsck per a verificar l'estat dels sistemes de fitxers. En algunes distribucions, aquest esquema no és tan precís, i solen estar dispersos en diversos directoris, com */sbin*, */usr/sbin*, */usr/local/sbin* principalment. Generalment s'enllaça (o es munta com a sistema de fitxers) en */usr/sbin*, i */usr/local/sbin* sol guardar-se per a eines locals instal·lades per l'administrador.

- **/svr**: directori reservat per a especificar les dades dels serveis que s'executen en el sistema. La idea és incorporar un directori per servei. Si les dades són internes al servei i no consumibles/consultables per tercers, es recomana posar-les a */var/lib*. De moment, les distribucions no han fet un gran ús d'aquest espai, mantenint molts dels o bé en directoris a */etc* o a */var/lib* quan podrien migrar-se a aquest espai.

- **/tmp**: fitxers temporals de les aplicacions o del sistema. Encara que són per a l'execució temporal, entre dues execucions l'aplicació o servei no pot assumir que trobarà els fitxers anteriors. Fins i tot es recomana a les distribucions que, entre arrencades del sistema, s'esborri aquest directori.

- **/usr**: diferents elements instal·lats en el sistema (normalment, només de lectura). Algun programari de sistema més complet s'instal·la aquí, a més de complements multimèdia (icones, imatges, sons, per exemple en /usr/share) i la documentació del sistema (/usr/share/doc). També /usr/local se sol utilitzar per a instal·lar documentació, programari local accessible als usuaris o elements complementaris. Com a cas especial important, */usr/bin*, on se solen desar la majoria d'ordres executables pels usuaris comuns del sistema. També podem destacar */usr/include* i */usr/lib*, on

s'emmagatzemen els fitxers estàndard d'*include* per al desenvolupament en llenguatge C, i les biblioteques disponibles per a programació.

- **/var**: fitxers de registre de sessió o d'estat (fitxers de tipus *log*) o errors del sistema i de diferents serveis, tant locals com de xarxa. Per exemple, fitxers de sessió en /var/log, contingut dels correus en /var/spool/mail o treballs d'impressió en /var/spool/lpd.

Aquests són alguns dels directoris definits en l'FHS per al sistema arrel; després s'especifiquen algunes subdivisions, com el contingut dels /usr i /var i els fitxers de dades o executables típics que s'esperen trobar com a mínim en els directoris (vegeu les referències en els documents FHS).

Respecte a les distribucions, Fedora o Red Hat segueix l'estàndard FHS molt de prop. Només presenta alguns canvis en els arxius presents en /usr i /var. En /etc hi sol haver un directori per component configurable, i hi ha algun directori especial, com /etc/sysconfig, en què es troba gran part de la configuració de diversos serveis bàsics del sistema. En /opt i /usr/local no hi sol haver programari instal·lat, tret que l'usuari l'instal·li. Debian, per la seva part, segueix l'estàndard, encara que afegeix alguns directoris de configuració especials en /etc. Tots dos grups de distribucions acostumen a disposar a *etc/default* d'algunes configuracions per defecte de serveis o de components del sistema.

Un altre estàndard en procés és l'LSB (Linux Standard Base) [Linc]. La idea d'aquest és definir uns nivells de compatibilitat entre les aplicacions, biblioteques i utilitats, de manera que sigui possible la portabilitat de les aplicacions entre distribucions sense gaires problemes. A més de l'estàndard, proporcionen conjunts de prova (tests) per a verificar el nivell de compatibilitat. LSB en si mateix és un recopilatori de diversos estàndards aplicats a GNU/Linux.

Nota

Estàndards d'especificacions:
http://refspecs.linuxfoundation.org
LSB:
http://refspecs.linuxfoundation.org/lsb.shtml
https://wiki.linuxfoundation.org/en/LSB

1.3. Documentació del sistema en línia

Un dels aspectes més importants per a les nostres tasques d'administració serà disposar de la documentació correcta per al nostre sistema i el programari instal·lat. Hi ha moltes fonts d'informació, entre les quals destacarem les següents:

a) man és l'ajuda per excel·lència. Ens permet consultar el manual de GNU/Linux, que està agrupat en diverses seccions, corresponents a instruccions, administració, formats de fitxers, instruccions d'usuari, crides de llenguatge C, etc. Normalment, per a obtenir l'ajuda associada, tindrem suficient amb el següent:

```
man instrucció
```

Cada pàgina descriuria la instrucció juntament amb les seves opcions i aportaria alguns exemples d'utilització. De vegades, hi pot haver més d'una entrada al manual. Per exemple, és possible que hi hagi una crida C amb el mateix nom que una instrucció. En aquest cas, cal especificar quina secció volem veure:

```
man n instrucció
```

n és el número de secció (per exemple, 2 per a les crides a sistema, o 3 per a les rutines de la biblioteca C).

Hi ha també unes quantes eines d'exploració dels manuals, per exemple *xman* i *tkman*, que mitjançant una interfície gràfica faciliten l'examen de les diferents seccions, i també índexs de les instruccions (també els entorns KDE i Gnome permeten en els seus ajuts accedir a les pàgines *man*). Una altra instrucció interessant és *apropos*, paraula que ens permetrà localitzar pàgines *man* que parlin d'un tema determinat (associat amb la paraula buscada).

b) info és un altre sistema d'ajuda habitual. És un programa desenvolupat per GNU per a la documentació de moltes de les seves eines. Es tracta d'una eina textual en la qual els capítols i les pàgines es poden recórrer per mitjà d'un sistema de navegació simple (basat en teclat).

c) Documentació de les aplicacions: a més de certes pàgines *man*, és habitual incloure en les aplicacions documentació extra, ja sigui en forma de guies d'usuari o manuals. Aquests components de documentació s'instal·len en el directori /usr/share/doc (o /usr/doc, depenent de la distribució), en què es crea un directori per paquet d'aplicació (en general, l'aplicació pot disposar de paquets de documentació separadament).

d) Sistemes propis de les distribucions. Red Hat sol venir amb uns CD de manuals de consulta que són instal·lables en el sistema i tenen formats HTML o PDF. Fedora disposa d'un projecte de documentació en el web del projecte. Debian porta els manuals com un paquet de programari més i se solen instal·lar en /usr/share/doc. D'altra banda, disposa d'eines que classifiquen la documentació present en el sistema i l'organitzen per menús per a la visualització, com *dwww* o *dhelp*, que presenten interfícies web per a examinar la documentació del sistema.

e) Finalment, **els escriptoris X**, com Gnome i KDE, també porten sistemes de documentació propis amb la seva documentació i manuals, i també informació per a desenvolupadors, ja sigui en forma d'ajuts gràfics en les seves aplicacions, o en aplicacions pròpies que recopilen els ajuts (per exemple, *devhelp* per a Gnome).

1.4. Eines de gestió de paquets

En qualsevol distribució, els paquets són l'element bàsic per a tractar les tasques d'instal·lació de programari nou, l'actualització de l'existent o l'eliminació del no utilitzat.

Bàsicament, un paquet és un conjunt de fitxers que formen una aplicació o una unió de diverses aplicacions relacionades, formant un únic fitxer (denominat *paquet*), amb un format propi i comprimit, que és el que es distribueix, ja sigui via CD, DVD o mitjançant accés a serveis d'FTP o web a repositoris públics.

L'ús de paquets facilita afegir o treure programari, en considerar-ho una unitat i no haver de treballar amb els fitxers individuals.

De vegades, algunes aplicacions o components són distribuïts en més d'un paquet, separant els components essencials dels opcionals o relacionats amb desenvolupament, *plugins* d'usuari o components modulars. Podem trobar el següent: paquet com el principal, mentre d'altres com paquet-common ens ofereixin els fitxers comuns associats; paquet-devel i/o -libs, els fitxers associats a desenvolupament i altres denominacions per a fitxers extra per a l'aplicació/component del sistema.

En el contingut de la distribució (els seus CD/DVD o imatges ISO) els paquets solen estar agrupats per categories com a) base: paquets indispensables per al funcionament del sistema (utilitats, programes d'inici, biblioteques de sistema); b) sistema: utilitats d'administració, instruccions d'utilitat; c) desenvolupament (*developement*): utilitats de programació, com editors, compiladors, depuradors...; d) gràfics: controladors i interfícies gràfiques, escriptoris, gestors de finestres, i e) altres categories.

Per a la instal·lació d'un paquet, serà necessari efectuar una sèrie de passos:

1) Previ (preinstal·lació): comprovar que hi ha el programari necessari (i amb les versions correctes) per al seu funcionament (dependències), ja sigui biblioteques de sistema o altres aplicacions que siguin usades pel programari.

2) Descompressió del contingut del paquet, copiant els fitxers a les seves localitzacions definitives, ja siguin absolutes (tindran una posició fixa) o, si es permet, resituades en altres directoris.

3) Postinstal·lació: retocar els fitxers necessaris, configurar possibles paràmetres del programari, adequar-lo al sistema...

Depenent dels tipus de paquets, aquests passos poden ser automàtics majoritàriament (així és en el cas d'RPM [Bai03] i DEB [Deb02]). També potser es necessita fer-los tots de manera manual (cas TGZ), depenent de les eines que proporcioni la distribució.

Veurem, a continuació, potser els tres paquets més clàssics de la majoria de distribucions. Cada distribució acostuma a tenir-ne un per estàndard i suporta algun dels altres. També observarem una sèrie de noves propostes per a sistemes d'empaquetat "universals" que puguin ser utilitzats de forma genèrica per més d'una distribució.

1.4.1. Paquets TGZ

Els paquets TGZ són potser els d'utilització més antiga. Les primeres distribucions de GNU/Linux els utilitzaven per a instal·lar el programari, i encara els usen diverses distribucions (per exemple, Slackware) i alguns UNIX comercials. Són una combinació de fitxers units per la instrucció *tar* en un únic fitxer .tar, que després ha estat comprimit per la utilitat *gzip*, i sol aparèixer amb l'extensió .tgz o bé .tar.gz. Així mateix, avui dia és comú trobar els tar.bz2, que utilitzen, en lloc de *gzip*, una altra utilitat anomenada *bzip2* que, en alguns casos, aconsegueix més compressió de l'arxiu.

Aquest tipus de paquet no conté cap tipus d'informació de dependències, i pot presentar tant contingut d'aplicacions en format binari com en codi font. Ho podem considerar com una espècie de col·lecció de fitxers comprimida.

Nota

Els paquets TGZ són una eina bàsica a l'hora d'instal·lar programari no organitzat i són molt útils per a fer processos de còpia de seguretat i restauració d'arxius.

En contra del que pot semblar, aquest és un format molt utilitzat i, sobretot, per creadors o distribuïdors de programari extern a la distribució. Molts creadors de programari que treballen per a plataformes diverses, com diversos UNIX comercials i diferents distribucions de GNU/Linux, ho prefereixen com a sistema més senzill i portable.

Exemple

Un d'aquests casos és el projecte GNU, que distribueix el seu programari en aquest format (en forma de codi font), ja que es pot utilitzar en qualsevol UNIX, ja sigui un sistema de propietat, una variant BSD o una distribució GNU/Linux.

Si es tracta de format binari, haurem de tenir en compte que sigui adequat per al nostre sistema; per exemple, sol ser comuna alguna denominació com la que segueix (en aquest cas, la versió 1.4 d'un paquet):

```
paquet-i686-pc-linux-gnu-1.4-installer.tar.gz
```

en què tenim el nom del paquet i l'arquitectura a la qual està destinat (*i686,* Pentium II o superiors o compatibles). Podria ser *i386*, *i586*, *i686*, *k6* (*AMD k6*), *k7* (AMD Athlon), *amd64* o *x86_64* (per a AMD64 i alguns Intel de 64 bits), o *ia64* (Intel Itanium). N'hi ha altres per a arquitectures d'altres màquines, com *sparc*, *powerpc*, *mips*, *hppa*, *alpha*... Després ens indica que és per a Linux, en una màquina PC, amb la versió del programari 1.4.

Si fos en format font, sol aparèixer com a:

```
paquet-source-1.4.tar.gz
```

on ens indiquen la paraula *source*. En aquest cas, no esmenta la versió de l'arquitectura de màquina, la qual cosa ens indica que està preparat (en principi) per a compilar-se en diferents arquitectures.

D'una altra manera, hi hauria diferents codis per a cada sistema operatiu o font: *GNU/Linux*, *Solaris*, *BSD*...

El procés bàsic amb aquests paquets consisteix en el següent:

1) Descomprimir el paquet (no solen utilitzar ruta absoluta, amb la qual cosa es poden descomprimir en qualsevol directori):

```
tar -xzvf fitxer.tar.gz (o fitxer.tgz)
```

Amb la instrucció *tar* posem les opcions *x* (extreure fitxers), *z* (descomprimir), *v* (veure els passos del procés) i *f* (donar nom del fitxer per tractar). Segons la compresió utilitzada gzip, bzip2 o altres) pot ser necessari canviar l'opció de descomprimir; consulteu la pàgina *man* de l'ordre *tar*. Per exemple, utilitzarem -j per a bzip2, -J per a xz, etc.

També es pot fer separadament (sense la *z* del *tar*):

```
gunzip fitxer.tar.gz
```

(ens deixa un fitxer *tar*)

```
tar -xvf fitxer.tar
```

2) Una vegada tenim descomprimit el *tgz*, tindrem els fitxers que contenia; normalment, el programari ha d'incloure algun fitxer de tipus Readme o Install, en què ens especificaran les opcions d'instal·lació pas per pas, i també possibles dependències del programari.

En primer lloc, caldrà verificar les dependències (se solen indicar en el fitxer amb els passos d'instal·lació), per si disposem del programari adequat. Si no, l'haurem de buscar i instal·lar.

Si es tracta d'un paquet binari, la instal·lació sol ser bastant fàcil, ja que o bé directament ja serà executable on l'hàgim deixat, o portarà algun instal·lador propi. Una altra possibilitat serà que ho hàgim de fer manualment, i llavors n'hi haurà prou de copiar (*cp -r*, còpia recursiva) o moure (instrucció *mv*) el directori a la posició volguda.

Un altre cas és el format de codi font. En aquest, abans d'instal·lar el programari, haurem de fer un pas de compilació. Per a això caldrà llegir amb cert detall les instruccions que porti el programa. Però la majoria de desenvolupadors usen un sistema de GNU anomenat *autoconf* (d'*autoconfiguració*), en el qual habitualment s'usen els passos següents (si no apareixen errors):

a) ***. /configure***: es tracta d'un *script* que configura el codi per a poder ser compilat a la nostra màquina, i verifica que hi hagi les eines adequades. L'opció *--prefix = directori* permet especificar on s'instal·larà el programari. Normalment, se suporten moltes opcions addicionals de configuració segons el programari (amb *-help* es mostren les que accepta).

b) ***make***: la compilació pròpiament dita.

c) ***make install***: la instal·lació del programari a un lloc adequat, especificat prèviament com a opció en el *configure* o assumida per defecte.

Aquest és un procés general, però depenent del programari se seguirà o no; hi ha casos bastant pitjors, en què tot el procés s'ha de fer a mà, retocant fitxers de configuració o el Makefile mateix, o compilant un per un els fitxers, però això, per sort, és com més va menys habitual.

En cas de voler esborrar el programari instal·lat, caldrà utilitzar el desinstal·lador si ens en proporcionen, o si no, esborrar directament el directori o els fitxers que es van instal·lar, anant amb compte amb les possibles dependències.

Els paquets *tgz* (y d'altres com ara tar.bz2 i tar.xz) són bastant habituals com a mecanisme de còpia de seguretat en tasques d'administració, per exemple, per a desar còpies de dades importants, fer còpies de comptes d'usuari, o guardar còpies antigues de dades que no sabem si tornarem a necessitar. Se sol utilitzar el procés següent: suposem que volem guardar una còpia del directori *dir*; llavors executem *tar -cvf dir.tar dir* (*c* per a compactar *dir* en el fitxer dir.tar) i *gzip dir.tar* (comprimir) o bé en una sola instrucció com la següent:

```
tar -czvf dir.tgz dir
```

El resultat serà un fitxer *dir.tgz*. Cal ser acurat si ens interessa conservar els atributs dels fitxers o els permisos d'usuari, i també possiblement fitxers d'enllaç (*links*) que hi pugui haver (haurem d'examinar les opcions de *tar* perquè s'ajusti a les opcions que volem).

1.4.2. Fedora/Red Hat: paquets RPM

El sistema de paquets RPM [Bai] creat per Red Hat representa un pas endavant, ja que inclou la gestió de dependències i tasques de configuració del programari. A més, el sistema desa una petita base de dades amb els paquets ja instal·lats, que es pot consultar i s'actualitza amb les noves instal·lacions.

Els paquets RPM, per convenció, solen usar un nom com el següent:

```
paquet-versio-rev.arq.rpm
```

en què *paquet* és el nom del programari, *versio* és la numeració de versió del programari, *rev* sol ser la revisió del paquet RPM, que indica les vegades que s'ha construït, i *arq*, l'arquitectura a la qual va destinat el paquet, ja sigui Intel/AMD (*i386*, *i586*, *i686*, *x86_64*, *ia64*) o altres com *alpha*, *sparc*, *ppc*... L'"arquitectura" *noarch* se sol usar quan és independent de l'arquitectura, com, per exemple, un conjunt de *scripts*, i *src* en el cas que es tracti de paquets de codi font. L'execució típica inclou l'execució del programa *rpm*, amb les opcions de l'operació que volem dur a terme, juntament amb els noms de paquets per processar junts.

Exemple

El paquet *apache-1.3.19-23.i686.rpm* indicaria que es tracta del programari Apache (el servidor web), en la seva versió 1.3.19, revisió del paquet RPM 23, per a arquitectures Pentium II o superiors.

Les operacions típiques amb els paquets RPM inclouen:

- **Gestió de dependències**: els paquets RPM incorporen la idea de gestió de dependències i de base de dades dels paquets existents.

- **Informació del paquet**: es consulta sobre el paquet una informació determinada, s'usa l'opció *-q* acompanyada del nom del paquet (amb *-p* si es fa sobre un arxiu *rpm*). Si el paquet no ha estat instal·lat encara, l'opció seria *-q* acompanyada de l'opció d'informació que es vulgui demanar, i si es volen preguntar tots els paquets instal·lats alhora, l'opció seria *-qa*. Les preguntes a un paquet instal·lat poden ser:

Consulta	Opcions RPM	Resultats
Arxius	*rpm -ql*	Llista dels arxius que conté el paquet (passat com a paràmetre)

Consulta	Opcions RPM	Resultats
Informació	*rpm -qi*	Descripció del paquet
Requisits	*rpm -qR*	Requisits previs, biblioteques o programari

- **Instal·lació:** simplement *rpm -i paquet.rpm*, o bé amb l'URL en què es troba el paquet, per a descarregar-lo des de servidors FTP o web. Només cal utilitzar la sintaxi ftp:// o http:// per a donar la localització del paquet. La instal·lació es podrà fer sempre que s'estiguin complint les dependències del paquet, ja sigui programari previ o biblioteques que haurien d'estar instal·lades. En cas de no complir-lo, ens dirà quin programari falta, i el nom del paquet que el proporciona. Es pot forçar la instal·lació (a risc que no funcioni) amb les opcions *--force* o *--nodeps,* o simplement no fer cas de la informació de les dependències. La tasca d'instal·lació (duta a terme per *rpm*) d'un paquet comporta diferents subtasques:
  - Verificar les possibles dependències.
  - Examinar conflictes amb altres paquets prèviament instal·lats.
  - Fer tasques prèvies a la instal·lació.
  - Decidir què cal fer amb els fitxers de configuració associats al paquet si existien prèviament.
  - Desempaquetar els fitxers i col·locar-los al lloc correcte.
  - Fer tasques de postinstal·lació.
  - Finalment, emmagatzemar el registre de les tasques fetes en la base de dades d'RPM.

- **Actualització:** equivalent a la instal·lació, però comprovant primer que el programari ja existeix; *rpm -U paquet.rpm*. S'encarregarà d'esborrar la instal·lació prèvia.

- **Verificació:** durant el funcionament normal del sistema, molts dels arxius instal·lats canvien. En aquest sentit, RPM permet verificar els arxius per a detectar les modificacions, bé pel procés normal, bé per algun error que podria indicar dades corrompudes. Mitjançant *rpm -V paquet* verifiquem un paquet concret, i mitjançant *rpm -Va* els verifiquem tots.

- **Eliminació:** esborrar el paquet del sistema RPM (*-e* o *--erase*). Si hi ha dependències, pot ser necessari eliminar-ne d'altres primer.

Exemple

Per a un cas remot:

```
rpm -i ftp://lloc/directori/paquet.rpm
```

ens permetria descarregar el paquet des del lloc FTP o web proporcionat, amb la seva localització de directoris, i procedir en aquest cas a la instal·lació del paquet.

Cal vigilar la procedència dels paquets, i només utilitzar fonts de paquets conegudes i fiables, ja sigui del fabricant mateix de la distribució o de llocs en què confiem. Tenim juntament amb els paquets alguna "signatura" digital perquè en puguem comprovar l'autenticitat; se solen utilitzar les sumes *md5* per a comprovar que el paquet no s'ha alterat, i altres sistemes com GPG (versió GNU de PGP) per a comprovar l'autenticitat de l'emissor del paquet. Hi ha també a Internet diferents magatzems de paquets RPM genèrics, en què estan disponibles per a diferents distribucions que usin o permetin el format RPM.

Nota

Vegeu el lloc www.rpmfind.net

Per a un ús segur de paquets, els repositoris (oficials, i alguns de tercers) firmen electrònicament els paquets, per exemple amb el GPG esmentat. Això ens permet assegurar (si disposem de les firmes) que els paquets procedeixen de la font fiable. Cada proveïdor (el repositori) inclou uns fitxers de firma PGP amb la clau per al seu lloc. Les dels repositoris oficials ja es troben instal·lades, i de les de tercers haurem d'obtenir el fitxer de clau, i incloure-la en RPM, típicament:

```
$ rpm --import GPG-KEY-FILE
```

GPP-KEY-FILE és el fitxer clau GPG o l'URL del fitxer esmentat; aquest fitxer també tindrà suma *md5* per a comprovar-ne la integritat. Podem conèixer les claus existents en el sistema amb el següent:

```
$ rpm -qa | grep ^gpg-pubkey
```

Podem observar més detalls a partir de la clau obtinguda:

```
$ rpm -qi gpg-key-xxxxx-yyyyy
```

Per a un paquet *rpm* concret podrem comprovar si disposa de firma i quina s'ha utilitzat:

```
$ rpm --checksig -v <paquet>.rpm
```

I per a verificar que un paquet és correcte partint de les firmes disponibles, es pot comprovar amb:

```
$ rpm -K < paquet >.rpm
```

Hem de ser acurats en importar només aquelles claus dels llocs en què confiem. Quan RPM trobi paquets amb firma que no tinguem en el nostre sistema, o el paquet no estigui firmat, ens avisarà, i l'acció ja dependrà de la nostra actuació.

Quant al suport RPM en les distribucions, en Fedora (Red Hat, i també en les seves derivades), RPM és el format per defecte de paquets i el que usa àmpliament la distribució per a les actualitzacions i la instal·lació de programari. En Debian s'utilitza el format denominat *DEB* (com veurem més endavant), però també hi ha suport per a RPM (existeix la instrucció *rpm*), però només per a consulta o informació de paquets. En el cas que sigui imprescindible instal·lar un paquet *rpm* en Debian, es recomana utilitzar la utilitat *alien*, que permet convertir formats de paquets, en aquest cas d'RPM a DEB, i procedir a la instal·lació amb el paquet convertit.

A més del sistema base d'empaquetament de la distribució, avui dia cada una sol aportar un sistema de gestió de programari intermedi de nivell més alt, que afegeix una capa superior al sistema base, facilita les tasques de gestió del programari i afegeix una sèrie d'utilitats per a controlar millor el procés.

En el cas de Fedora (Red Hat i derivats) s'utilitza el sistema YUM, que permet, com a eina de nivell més alt, la instal·lació i gestió de paquets en sistemes RPM, i també la gestió automàtica de dependències entre els paquets. Facilita l'accés a múltiples repositoris diferents; en algunes distribucions es basava en el fitxer */etc/yum.conf*, però actualment està repartida en diversos directoris.

Nota

Vegeu el YUM a http://yum.baseurl.org/

La configuració de YUM es basa en:

```
/etc/yum.config (fitxer d'opcions)
/etc/yum (directori per a algunes utilitats associades)
/etc/yum.repos.d (directori d'especificació de repositoris, un fitxer per a cada un,
     s'inclou informació de l'accés i localització de les firmes GPG).
```

Nota

Per a distribucions de Fedora superiors a la versió 22, el nou sistema DNF substitueix a YUM. Més sobre DNF a http://dnf.baseurl.org/

Per a les operacions típiques de YUM, un resum seria:

Ordre	Descripció
yum install <nom>	Instal·la el paquet amb el nom
yum update <nom>	Actualitza un paquet existent
yum remove <nom>	Elimina un paquet
yum list <nom>	Busca un paquet per nom (només nom)
yum search <nom>	Busca més àmpliament
yum provices <arxiu>	Busca paquets que proporcionin el fitxer

Ordre	Descripció
yum update	Actualitza tot el sistema
yum upgrade	Igual que l'anterior incloent-hi paquets addicionals

Fedora també proporciona algunes utilitats gràfiques per a l'actualització, com Yumex. Depenent de la versió han existit diferents eines, però d'existència i permanència curtes. Alguna altra (textual) com Fedup s'ha convertit en l'estàndard per a l'actualització de la distribució entre versions.

El sistema YUM, que s'utilitzava en la distribució Fedora i altres distribucions derivades de Red Hat, s'està substituint progressivament pel sistema DNF (Dandified YUM). Aquest nou sistema és l'opció per defecte, a partir de Fedora 22, per a la gestió d'alt nivell de paquets en Fedora. Les altres distribucions basades en Fedora/Red Hat s'aniran adaptant progressivament, amb migracions des de YUM a DNF.

Nota

Vegeu DNF a http://dnf.baseurl.org/.

En el nou DNF, s'han intentat millorar certes deficiències que s'observaven a YUM:

- S'han millorat les prestacions.
- S'ha millorat l'ús intensiu de memòria que es feia.
- S'ha reduït una mica la lentitud en el procés de resolució de dependències, a banda de la possible corrupció de dependències, en alguns casos.
- S'ha corregit la dependència de certes API no ben documentades i de llenguatges (pels *seus scripts*) com Python (versió 2), quan la majoria de distribucions estan migrant a Python 3.

Per millorar aquestes deficiències s'ha desenvolupat un *core* del sistema, amb un seguit de llibreries independents associades, per resoldre els problemes individuals: gestió de paquets i les seves dependències (mitjançant RPM, i les llibreries *libsolv* i *hawkey*), gestió de metadades i repositoris (*librepo*), i gestió de grups de paquets (*libcomps*). Això permet als desenvolupadors del sistema DNF disposar d'unes API clares, basades en un seguit de llibreries externes que poden millorar-se independentment.

La interfície d'ordres s'ha mantingut pràcticament compatible amb YUM i només haurà de canviar-se l'ús de l'ordre `yum` per `dnf`. Així, una instal·lació de paquet individual passa a realitzar-se amb:

```
$ dnf install package-name
```

Malgrat que DNF és habitual en les distribucions que ho han adoptat com a alternativa YUM, el propi YUM ens recorda, en l'ús de l'ordre `yum`, que es troba obsolet, i ens adverteix que cal usar les noves ordres mitjançant un avís semblant a aquest :

```
Yum command has been deprecated, use dnf instead.
See 'man dnf' and 'man yum2dnf' for more information.
To transfer transaction metadata from yum to DNF, run 'dnf migrate'
```

Nota

Pel que fa a les diferències entre CLI, DNF i YUM, pot trobar-se una llista actualitzada a: http://dnf.readthedocs.io/en/latest/cli_vs_yum.html.

Com a diferències de línia d'ordres de DNF i YUM, podem destacar:

- Les opcions *update* i *upgrade* són exactament iguals, no hi ha cap diferència a DNF.
- *dnf remove nombrepaquete* esborra tots els paquets (prèvia confirmació) amb el nom esmentat; en cas de desitjar una versió concreta, haurem d'especificar-la en l'ordre.
- Hi ha, també, diferències menors de subopcions específiques que poden consultar-se a Read the Docs o a la pàgina *man* de l'ordre: `man dnf`.

Quant a la configuració del sistema global, DNF utilitza el seu fitxer de configuració */etc/dnf/dnf.conf* i tots els fitxers de descripció de *repos* (**.repo*), que es troben a */etc/yum.repos.d*. D'altra banda, els fitxers del sistema mantinguts en cau poden trobar-se a */var/cache/dnf*.

Nota

Per a una referència completa de la configuració del sistema podem consultar: http://dnf.readthedocs.io/en/latest/conf_ref.html.

1.4.3. Debian: paquets DEB

Debian té eines interactives com *tasksel*, que permet escollir uns subconjunts de paquets agrupats per tipus de tasques: paquets per a X, per a desenvolupament, per a documentació, etc.; o com *dselect,* que facilita navegar per tota la llista de paquets disponible (n'hi ha milers) i escollir aquells que vulguem instal·lar o desinstal·lar. De fet, aquestes són només una interfície del gestor de programari de nivell intermedi APT (equivalent al YUM a Fedora).

En el nivell de línia de comandes disposa de *dpkg,* que és la comanda de més baix nivell (seria l'equivalent a *rpm*), per a gestionar directament els paquets DEB de programari [Deb], típicament *dpkg -i paquet.deb* per a fer la instal·lació. Es poden dur a terme tot tipus de tasques, d'informació, instal·lació, esborrament o canvis interns en els paquets de programari.

El nivell intermedi (com el cas de YUM a Fedora) el presenten les eines APT (la majoria són comandes *apt-xxx*). APT permet gestionar els paquets per mitjà d'una llista de paquets actuals i disponibles a partir de diverses fonts de pro-

gramari, ja sigui des dels CD de la instal·lació, llocs FTP o web (HTTP). Aquesta gestió es fa de manera transparent, i el sistema és independent de les fonts de programari.

La configuració del sistema APT s'efectua des dels arxius disponibles en /etc/apt, en què /etc/apt/sources.list és la llista de fonts disponibles. Podria ser, per exemple:

```
deb http://http.us.debian.org/debian stable main contrib non-free
debsrc http://http.us.debian.org/debian stable main contrib non-free
deb http://security.debian.org stable/updates main contrib non-free
```

en el qual hi ha recopilades diverses de les fonts "oficials" per a una Debian (*stable* en aquest cas), des d'on es poden obtenir els paquets de programari, i també les actualitzacions que estiguin disponibles. Bàsicament, s'especifica el tipus de font (Web/FTP en aquest cas), el lloc, la versió de la distribució (*stable*), i les categories del programari que es buscarà (lliure, o contribucions de tercers o de llicència no lliure o comercial).

Els paquets de programari estan disponibles per a les diferents versions de la distribució Debian. Hi ha paquets per a les versions *stable*, *testing* i *unstable*. L'ús d'uns o d'altres determina el tipus de distribució (amb canvi previ de les fonts de repositoris en sources.list). Es poden tenir fonts de paquets barrejades, però no és gaire recomanable, ja que es podrien donar conflictes entre les versions de les diferents distribucions.

Un sistema molt potent

Els paquets DEB de Debian són potser el sistema d'instal·lació més potent existent en GNU/Linux. Una de les seves prestacions més destacables és la independència del sistema de les fonts dels paquets (mitjançant APT).

Una vegada tenim les fonts de programari configurades, la principal eina per a manejar-les en el nostre sistema és *apt-get*, que ens permet instal·lar, actualitzar o esborrar des del paquet individual fins a poder actualitzar la distribució sencera. Hi ha també una interfície a *apt-get*, anomenada *aptitude*, la interfície d'opcions del qual és pràcticament igual (de fet, es podria qualificar d'emulador d'*apt-get*, ja que la interfície és equivalent). Com a avantatge, aporta una gestió millor de dependències dels paquets, i algoritmes per a solucionar els conflictes de paquets que poden aparèixer. De fet, en les últimes versions de Debian, *aptitude* és la interfície per defecte en línia de comandes per a la gestió de paquets.

Es poden continuar fent servir totes dues sense problemes, però davant de problemes complexos de resolució de conflictes entre paquets es recomana *aptitude* perquè aporta resolucions millors.

Algunes funcions bàsiques d'*apt-get* són:

1) Instal·lació d'un paquet particular:

```
apt-get install paquet
```

2) Esborrament d'un paquet:

```
apt-get remove paquet
```

3) Actualització de la llista de paquets disponibles:

```
apt-get update
```

4) Actualització de la distribució; podríem efectuar els passos combinats:

```
apt-get update
apt-get upgrade
apt-get dist-upgrade
```

Mitjançant aquest últim procés, podem mantenir la nostra distribució actualitzada permanentment, actualitzant els paquets instal·lats i verificant les dependències amb els nous. Una eina útil per a construir aquesta llista és *apt-spy*, que intenta buscar els llocs oficials més ràpids, o *netselect*, que ens permet provar una llista de llocs. D'altra banda, podem buscar les fonts oficials (configurar-les amb *apt-setup*), o bé copiar algun fitxer de fonts disponible. El programari addicional (de tercers) pot necessitar afegir altres fonts més (a /etc/apt/sources.list); es poden obtenir llistes de llocs no oficials de fonts alternatives, però cal ser curós quan es facin servir per raons de seguretat.

L'actualització del sistema en particular genera una descàrrega d'un gran nombre de paquets (en especial en *unstable*), i es fa recomanable buidar la memòria cau, el repositori local, amb els paquets descarregats (es mantenen en /var/cache/apt/archive) que ja no s'hagin d'utilitzar, bé amb *apt-get clean,* per a eliminar-los tots, o bé amb *apt-get autoclean*, per a eliminar aquells paquets no necessaris perquè ja hi ha noves versions i ja no seran necessaris (en principi). Cal tenir en compte que no tornem a necessitar aquests paquets per raons de reinstal·lació, perquè en aquest cas els hauríem de tornar a descarregar.

El sistema APT també permet el que es denomina *SecureAPT*, que és la gestió segura de paquets mitjançant verificació de sumes (*md5*) i la firma de fonts de paquets (de tipus GPG). Si durant la descàrrega no estan disponibles les firmes, *apt-get* n'informa i genera una llista amb els paquets no firmats, i demana si es deixen instal·lar o, no, i en deixa la decisió a l'administrador. S'obté la llista de fonts confiables actuals amb:

```
# apt-key list
```

Les claus GPG dels llocs oficials de Debian són distribuïdes mitjançant un paquet. Les instal·lem:

```
apt-get install debian-archive-keyring
```

Evidentment considerant que tenim sources.list amb els llocs oficials. S'espera que per defecte (depenent de la versió de Debian) aquestes claus ja s'instal·lin amb la instal·lació inicial del sistema. Per a altres llocs no oficials (que no proporcionin la clau en paquet), però que considerem confiables, en podem importar la clau, obtenint-la des del repositori (haurem de consultar on tenen la clau disponible, no hi ha un estàndard definit, encara que sol ser a la pàgina web inicial del repositori). S'utilitza *apt-key add* amb el fitxer, per a afegir la clau, o també:

```
# gpg --import arxiu.key
# gpg --export --armor XXXXXXXX | apt-key add -
```

X és un nombre hexadecimal relacionat amb la clau (vegeu les instruccions del repositori per a comprovar la manera recomanada d'importar la clau i les dades necessàries).

Una altra funcionalitat important del sistema APT són les funcions de consulta d'informació dels paquets, amb l'eina *apt-cache*, que ens permet interactuar amb les llistes de paquets de programari Debian.

Exemple

L'eina *apt-cache* disposa d'opcions que ens permeten buscar informació sobre els paquets, com per exemple:

1) Buscar paquets sobre la base d'un nom incomplet:

```
apt-cache search nom
```

2) Mostrar la descripció del paquet:

```
apt-cache show paquet
```

3) De quins paquets depèn:

```
apt-cache depends paquet
```

Una altra eina o funcionalitat d'APT interessant és *apt-show-versions*, que ens especifica quins paquets poden ser actualitzats (i per quines versions, vegeu l'opció *-u*).

Altres tasques més específiques es necessitaran fer amb l'eina de nivell més baix, *dpkg*. Es pot obtenir, per exemple, la llista d'arxius d'un paquet determinat ja instal·lat:

```
dpkg -L paquet
```

O la llista de paquets sencera amb:

```
dpkg -l
```

O buscar de quin paquet prové un element (fitxer, per exemple):

```
dpkg -S fitxer
```

Aquest en particular funciona per a paquets instal·lats; *apt-file* permet també buscar per a paquets encara no instal·lats.

Finalment, es poden esmentar també algunes eines gràfiques per a APT, com Synaptic, o les textuals ja esmentades, com *aptitude* o *dselect*.

En conclusió, es pot destacar que el sistema de gestió APT (en combinació amb el base *dpkg*) és molt flexible i potent a l'hora de gestionar les actualitzacions, i és el sistema de gestió de paquets usat a Debian i les seves distribucions derivades, com Ubuntu, Kubuntu, Knoppix, Linex, etc.

1.4.4. Nous formats d'empaquetat: Snap i Flatpak

Darrerament han sorgit noves alternatives als sistemes de paquets tradicionals per disminuir alguns dels problemes associats a l'ús dels sistemes previs. D'una banda, com a principals contendents en aquest punt, cal assenyalar Snappy/Snap (anomenat així depenent de si fem referència al sistema d'empaquetat o al paquet individual) i Flatpak.

Entre els problemes habituals dels sistemes d'empaquetat clàssics, podem destacar-ne els següents:

- Els paquets són excessivament dependents de la distribució. Malgrat ser un estàndard concret (com DEB o RPM), poden dependre de paquets específics de la distribució concreta, o pot haver-hi dependències de llibreries o versions que poden no trobar-se en la distribució destí.

- L'anterior genera una "fragmentació" dels paquets GNU/Linux, que fa gairebé impossible empaquetar aplicacions de manera vàlida per a diferents distribucions GNU/Linux.

- En una mateixa distribució, el paquet pot necessitar versions de biblioteques anteriors o posteriors a les existents, per la qual cosa crea dependències difícils (o impossibles, en alguns casos) de resoldre.

- No són possibles els paquets "universals" vàlids per a qualsevol distribució.

Els nous sistemes, com Snappy/Snap (provinent de Canonical, desenvolupador d'Ubuntu) i Flatpak (abans conegut com xdg-app i promogut per desenvolupadors de la comunitat Gnome i Red Hat), prometen, en canvi, diferents prestacions (resoltes en aquest moment amb diferents resultats):

- Facilitar al desenvolupador un empaquetament en un únic format, que pràcticament permet la instal·lació en qualsevol distribució GNU/Linux. Tot dependrà del fet que cada distribució decideixi donar suport a un o més d'aquests nous formats.

- Es promet que en aquests nous formats els paquets estaran aïllats els uns dels altres i, també, de les parts crítiques del sistema operatiu, de manera que es maximitzi la seguretat.

- Algun d'ells (Flatpak) permet empaquetar l'aplicació directament amb les llibreries que utilitza, de manera que no depengui de les llibreries finals disponibles en el sistema, sinó que l'aplicació estigui empaquetada amb tot allò necessari per funcionar. El concepte és bastant similar al d'aplicació instal·lable en MacOS X d'Apple. Els crítics d'aquest concepte addueixen que es creen infinitat de còpies de les mateixes llibreries, amb múltiples versions, difícils de controlar (*bugs*, seguretat especialment), a banda de consumir en excés tant memòria de la màquina i com disc per a l'emmagatzematge. Flatpak té alguna solució; per exemple, per a l'existència de múltiples còpies, utilitza la duplicació de llibreries: si ja existeix una còpia d'aquestes en el sistema, les usa sense afegir còpies extra.

- Flatpak són aplicacions de tipus *sandboxed*. L'aplicació només pot veure's a ella mateixa, i un nombre limitat de llibreries i interfícies (API) del sistema operatiu.

- Snappy s'usa tant en paquets per a servidor com per a aplicacions d'escriptori, mentre que Flatpak (i la seva relació amb Gnome), encara que podrà ser utilitzat en altres entorns de finestres, està pensat per executar aplicacions en sessions d'usuari, aprofitant serveis com *dbus* i *systemd*.

- En el cas de Snap, existeix el concepte de repositori centralitzat (Snap Store o Ubuntu myApps,o altres noms, depenent de la plataforma destí), controlat per Canonical, mentre que en Flatpak no existeix aquest concepte, només repositoris múltiples per afegir i tenir, d'aquesta manera, fonts de diferents paquets.

Vegem algunes proves dels nous sistemes de paquets en diferents distribucions, en què actualment es troben més estesos.

Snappy

En aquests moments Snappy/Snap es troba implementat (com a nadiu) a Ubuntu, a partir de la versió 16.04 LTS, i també està disponible, en major o menor grau, en altres distribucions. Per al seu ús en la distribució, ha d'estar

Nota

Vegeu construcció de paquets Snap per part del desenvolupador a: http://snapcraft.io/create/.

instal·lat el paquet *snapd* (o nom similar depenent de la distribució), que inclou l'ordre `snap` per a la gestió de paquets. Si hem de crear paquets, també serà necessari el paquet *snapcraft*.

Vegem a Ubuntu (>= 16.04) una petita sessió de les ordres (CLI) d'interfície d'usuari amb aquest sistema d'empaquetat.

Entre altres, en la següent sessió d'ordre *snap* observem:

`snap help`	Per obtenir informació de les opcions disponibles.
`snap find [paquet]`	Per buscar paquets disponibles, o per nom o per part d'ell. Podem combinar-lo, mitjançant `pipe`, amb `grep` per trobar en els seus comentaris altres paraules associades al paquet.
`snap install paquet`	Per instal·lar un paquet determinat.
`snap refresh paquet`	Per actualitzar un paquet prèviament instal·lat. Si està disponible una nova versió, s'actualitzarà.
`snap list`	Per llistar paquets instal·lats en el sistema. Podem combinar-lo amb `grep` per trobar un paquet/paraula determinats dels paquets instal·lats.
`snap remove paquet`	Per desinstal·lar un paquet.
`snap changes`	Per llistar canvis recents en el sistema *snap*.

```
user@ubuntu:~$ sudo snap help
Usage:
snap [OPTIONS] <command>

The snap tool interacts with the snapd daemon to control the snappy software platform.

Available commands:
  abort          Abort a pending change
  ack            Adds an assertion to the system
  change         List a change's tasks
  changes        List system changes
  connect        Connects a plug to a slot
  create-user    Creates a local system user
```

```
  disconnect     Disconnects a plug from a slot
  find           Finds packages to install
  help           Help
  install        Install a snap to the system
  interfaces     Lists interfaces in the system
  known          Shows known assertions of the provided type
  list           List installed snaps
  login          Authenticates on snapd and the store
  logout         Log out of the store
  refresh        Refresh a snap in the system
  remove         Remove a snap from the system
  run            Run the given snap command
  try            Try an unpacked snap in the system

user@ubuntu:~$ sudo snap find
Name           Version    Developer      Notes    Summary
ab             1.0        snappy-test    -        Test snap with shortest name
ag-mcphail     1.0.1      njmcphail      -        The Silver Searcher - mcphail's build and
                                                    upstream git version
alsa-utils     1.1.0-1    woodrow        -        Utilities for configuring and using ALSA
apktool        2.1.1      ligboy         -        A tool for reverse engineering 3rd party,
                                                    closed, binary Android apps.
atom-cwayne    1.9.0      cwayne18       -        Atom Text Editor
audovia        3.3.1      songbuilder    -        Database application for making music using
                                                    JFugue MusicStrings
base           16.04+20160525.05-00  canonical  -  Base content for snapd
blender-tpaw   2.77a      tpaw           -        Blender is the free and open source 3D creation
                                                    suite.
cassandra      3.7        ev             -        Cassandra distributed database
cla-check      0.1        kyrofa         -        Check if Canonical's Contributor License
                                                    Agreement has been signed
click-parser   3.0.5      bhdouglass     -        Extract data from Ubuntu's click & snap
                                                    packages
compass-straightedge  1   caozhen        -        Construct geometric figures with
                                                    compass-and-straightedge construction
dolgia-test-tools  1.0    fgimenez       -        dolgia test tools
drmips         2.0.1-4    brunonova      -        Educational MIPS simulator
eeevil         1          chipaca        -        very evil
ejabberd       16.04      wyrddreams     -        ejabberd XMPP server
electrum-tpaw  2.6.4-tpaw3  tpaw         -        Lightweight Bitcoin Client
filebot        4.7~snap1  pointplanck    9.99EUR  The ultimate TV and Movie Renamer / Subtitle
                                                    Downloader
foobar21       2.1        testuser       -        This is a test snap
freecad        0.17       vejmarie       -        This is the first beta for freecad 0.17
                                                    supporting OCCT 7 / Netgen and new Smesh
                                                    version
freechartgeany 2.0.3-1snap  lucast70     -        Technical analysis software for stocks
```

```
gdoc-html-cleaner  0.1.2  caldav         -        Download Google Docs as cleaned HTML files
ghex-udt       1          canonical      -        Hex Editor
gmailfilter    0.0.1a     thomi          -        Programmatically filter gmail messages
gnuchess       2.1        tomechangosubanana  -   Plays a game of chess, includes GUI and CLI.
                                                    Run "gnuchess.readme" for more information.
gtwang-demo    1.0        gtwang         -        G.T.Wang demo application.
hangups        0.3.6      tomdryer       -        Third-party instant messaging client for Google
                                                    Hangouts
hello          2.10       canonical      -        GNU Hello, the "hello world" snap
.................

user@ubuntu:~$ sudo snap find | grep world
hello          2.10       canonical      -        GNU Hello, the "hello world" snap

user@ubuntu:~$ sudo snap install hello
Name   Version  Rev  Developer  Notes
hello  2.10     20   canonical  -

user@ubuntu:~$ sudo snap refresh hello
error: cannot perform the following tasks:
- Download snap "hello" from channel "stable" (revision 20 of snap "hello" already installed)

user@ubuntu:~$ sudo snap list
Name         Version               Rev  Developer  Notes
hello        2.10                  20   canonical  -
ubuntu-core  16.04+20160531.11-56  122  canonical  -

user@ubuntu:~$ sudo snap remove hello

Done
user@ubuntu:~$ sudo snap changes
ID   Status  Spawn                 Ready                 Summary
20   Done    2016-07-09T15:44:25Z  2016-07-09T15:44:28Z  Install "hello" snap
21   Error   2016-07-09T15:44:36Z  2016-07-09T15:44:36Z  Refresh "hello" snap
22   Done    2016-07-09T15:46:19Z  2016-07-09T15:46:20Z  Remove "hello" snap
```

En el sistema de fitxers, els paquets es troben en el directori */snap* i, depenent del tipus, si són d'execució gràfica, s'afegeixen als menús del sistema o, si són executables via comandes, seran anomenats des del *shell* amb noms semblants al nom del paquet. Sempre podem examinar els directoris:

- */snap/bin,* per trobar els binaris disponibles per a executar.

- */snap/nomeni_paquet/current/bin*, per trobar els binaris/executables associats a un paquet, en la seva versió més actual. Encara que el llançador de l'aplicació —pot ser que en forma *de shell script* per realitzar configuracions prèvies— es troba habitualment en l'anterior */snap/bin*, és recomana-

ble l'execució de l'aplicació, via l'ordre disponible en el primer directori, o accedir als menús de sistema on s'hagi instal·lat l'aplicació.

Flatpak

En el cas de Flatpak, provarem el seu ús a partir d'una distribució Fedora, com a mínim la versió 23 (es distribueix oficialment a Fedora 24). També està disponible en altres distribucions que, en el seu repositori, inclouen el paquet Flatpak.

Nota

Podeu veure la disponibilitat actual de Flatpak a http://flatpak.org/getting.html. Pel que fa a creadors de paquets, vegeu la guia per a desenvolupadors a http://flatpak.org/index.html#developers.

A Fedora (>=23) podem realitzar la instal·lació amb:

```
# dnf install flatpak
```

Una vegada instal·lat el paquet, disposarem de l'ordre `flatpak`, mitjançant la qual realitzarem les tasques següents (normalment aquestes tasques contenen certs passos):

- Instal·lar els repositoris de paquets necessaris. Inclou importar les claus GPG per identificar el repositori i els seus paquets, i importar els repositoris concrets a què volem accedir.

- Des del repositori incorporat, instal·lar el *runtime* necessari, que proporciona totes les dependències necessàries per a les aplicacions en el repositori.

- Ja podem veure així, amb diferents opcions, quins paquets hi ha disponibles, instal·lar-los i executar les aplicacions instal·lades.

En aquest quadre-resum, veiem algunes de les ordres que farem servir en la sessió posterior d'ordres:

`flatpak remote-add --gpg-import=llavegpg repo urlrepo`	Per afegir un repositori `repo` identificat per una `llavegpg` (descarregada prèviament) i localitzat a la URL `urlrepo`.
`flatpak remote-list`	Per llistar repositoris remots disponibles.
`flatpak install repo runtime version`	Per instal·lar des del repositori repo el *runtime* `runtime` amb el nombre de versió concret version.
`flatpak remote-ls repo --app`	Per llistar les aplicacions disponibles en el `repo`.
`flatpak install repo aplicacion stable`	Per instal·lar l'aplicació en la seva versió estable des del repositori `repo`.
`flatpak list`	Per llistar aplicacions disponibles.
`flatpak run nomeni`	Per executar una aplicació `nom` disponible.
`flatpak update`	Par actualitzar des de repositori totes les aplicacions instal·lades.

Vegem una sessió de comandes per a la instal·lació d'alguns paquets de Gnome des dels repositoris Flatpak de Gnome (gnome i gnome-apps), i incorporant el *runtime* de Gnome (algunes de les comandes necessiten permisos de *root*):

```
$ wget https://sdk.gnome.org/keys/gnome-sdk.gpg
--2016-07-10 18:47:52--  https://sdk.gnome.org/keys/gnome-sdk.gpg
Resolviendo sdk.gnome.org (sdk.gnome.org)... 209.132.180.169
Conectando con sdk.gnome.org (sdk.gnome.org)[209.132.180.169]:443... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 629
Grabando a: "gnome-sdk.gpg"

gnome-sdk.gpg                   100%[=====================================================>]
629  --.-KB/s    in 0s

2016-07-10 18:47:53 (73,4 MB/s) - "gnome-sdk.gpg" guardado [629/629]

$ flatpak remote-add --gpg-import=gnome-sdk.gpg gnome https://sdk.gnome.org/repo/
$ flatpak remote-add --gpg-import=gnome-sdk.gpg gnome-apps https://sdk.gnome.org/repo-apps/

$ flatpak remote-list
gnome
gnome-apps

$ flatpak remote-ls gnome
org.freedesktop.BasePlatform
org.freedesktop.BaseSdk
org.freedesktop.Platform
org.freedesktop.Platform.Locale
org.freedesktop.Platform.Var
org.freedesktop.Sdk
org.freedesktop.Sdk.Debug
org.freedesktop.Sdk.Locale
org.gnome.Platform
org.gnome.Platform.Locale
org.gnome.Sdk
org.gnome.Sdk.Debug
org.gnome.Sdk.Locale

$ flatpak install gnome org.gnome.Platform
error: Multiple branches available for org.gnome.Platform, you must specify one of: 3.20, 3.18, 3.16

$ flatpak install gnome org.gnome.Platform 3.20
9 delta parts, 71 loose fetched; 167139 KiB transferred in 23 seconds
Installing related: org.gnome.Platform.Locale
30 metadata, 135 content objects fetched; 803 KiB transferred in 7 seconds
```

```
$ flatpak remote-ls gnome-apps --app
org.gnome.Builder
org.gnome.Calculator
org.gnome.Calendar
org.gnome.Characters
org.gnome.Dictionary
org.gnome.Epiphany
org.gnome.Evince
org.gnome.Maps
org.gnome.Polari
org.gnome.Rhythmbox3
org.gnome.Todo
org.gnome.Weather
org.gnome.bijiben
org.gnome.clocks
org.gnome.eog
org.gnome.gedit
org.gnome.iagno

$ flatpak install gnome-apps org.gnome.gedit
1 delta parts, 1 loose fetched; 1725 KiB transferred in 5 seconds
Installing related: org.gnome.gedit.Locale

$ flatpak install gnome-apps org.gnome.Calculator stable
1 delta parts, 1 loose fetched; 665 KiB transferred in 3 seconds
Installing related: org.gnome.Calculator.Locale
6 metadata, 1 content objects fetched; 20 KiB transferred in 2 seconds

$ flatpak list
org.gnome.Calculator
org.gnome.gedit

$ flatpak run org.gnome.Calculator
```

Quant a l'estructura de sistema de fitxers, Flatpak (per a les versions examinades a Fedora) genera entre d'altres:

- En l'usuari local, un directori *.var/app/nombreapp* de configuracions per a l'execució de l'aplicació.

- En el sistema en */var/lib/flatpak*, es troben les aplicacions instal·lades, els repositoris disponibles i els *runtimes* utilitzats pel sistema.

- En el sistema en */var/lib/flatpak/app/nombreapp*, es manté l'aplicació en ella mateixa amb l'estructura de fitxers necessària per a la seva execució per part de flatpak. Per exemple, per a l'última versió d'una aplicació *appname* trobaríem els seus fitxers a */var/lib/flatpak/app/appname/current/activi/files*

on, entre d' altres, podrem observar els binaris o *scripts* de l'aplicació, així com la inclusió de les llibreries que necessiten l'aplicació per a la seva execució.

Una vegada examinats els nous sistemes Snappy i Flatpak, conclourem comentant alguns aspectes sobre els nous sistemes de paquets pendents de resoldre i les problemàtiques que poden generar:

- Tant en un cas com en un altre (Snap/Flatpak), existeixen alguns problemes pendents amb els models de seguretat, especialment per a les aplicacions gràfiques en X Window que, per raons d'inseguretat de la plataforma (per exemple, una aplicació pot usar X Window per enviar falsos esdeveniments de teclat i causar que una aplicació interactuï d'una manera no desitjada), impedeixen poder disposar de cert grau de seguretat o *sandbox*, aïllant les aplicacions que facin ús de la interfície gràfica. S'espera que aquesta part es vagi resolent a mesura que l'X Window Server es substitueixi per alternatives com Wayland (que podria aprofitar Flatpak per millorar la seguretat) o Mir (en el cas d'Ubuntu i Snap).

- Pel que fa a la resta, a mesura que es vagin resolent els diversos problemes, s'espera que les distribucions adoptin els nous sistemes (Snap i Flatpak) de manera dual, o bé es decideixin per un d'ells (creant una nova fragmentació semblant al cas DEB/RPM).

- En aquest moment també existeixen altres propostes com AppImage i OrbitalApps. La primera és més simple que les noves propostes esmentades i la segona està pensada per desenvolupar aplicacions portables, com les que podríem portar a USB, entre diversos sistemes.

1.5. Eines genèriques d'administració

En el camp de l'administració, també podríem considerar algunes eines, com les pensades de manera genèrica per a l'administració. Es pot indicar, però, que per a aquestes eines és difícil mantenir-se al dia, a causa dels plans actuals de versions de les distribucions, amb evolució molt ràpida. Algunes d'aquestes eines (encara que en un moment determinat poden no ser funcionals al complet en una distribució donada) són:

a) Webmin: és una eina d'administració pensada des d'una interfície web. Funciona amb una sèrie de connectors que es poden afegir per a cada servei per administrar i té formularis en què s'especifiquen els paràmetres de configuració dels serveis. A més, ofereix la possibilitat (si s'activa) de permetre administració remota des de qualsevol màquina amb navegador.

Nota

Podeu trobar aquesta eina a Webmin, www.webmin.com

b) Altres en desenvolupament, com **cPanel**, **ISPConfig**...

D'altra banda, els entorns d'escriptori Gnome i KDE solen disposar del concepte de *centre de control*, que permet gestionar tant l'aspecte visual de les interfícies gràfiques com tractar alguns paràmetres dels dispositius del sistema.

Quant a les eines gràfiques individuals d'administració, la distribució de GNU/Linux mateixa n'ofereix algunes directament (eines que acompanyen tant Gnome com KDE), eines dedicades a gestionar un dispositiu (impressores, so, targeta de xarxa, etc.) i altres per a l'execució de tasques concretes (connexió a Internet, configurar l'arrencada de serveis del sistema, configurar X Window, visualitzar registres...). Moltes són simples *frontends* (o caràtules) a les eines bàsiques de sistema, o bé estan adaptades a particularitats de la distribució.

Es pot destacar, en especial en aquest apartat, la distribució Fedora (Red Hat i derivats), que intenta disposar de diferents utilitats (més o menys minimalistes) per a diferents funcions d'administració. Les podem trobar a l'escriptori (al menú d'administració), o en comandes com *system-config-xxxxx* per a diferents funcionalitats com la gestió de pantalla, la impressora, la xarxa, la seguretat, els usuaris, els paquets, etc. Tot i que en les darreres versions de Fedora s'estan substituint progressivament pel concepte de panell de control (sigui configuració de Gnome o el panell KDE) amb funcionalitats semblants a aquestes utilitats. En podem veure algunes a la figura:

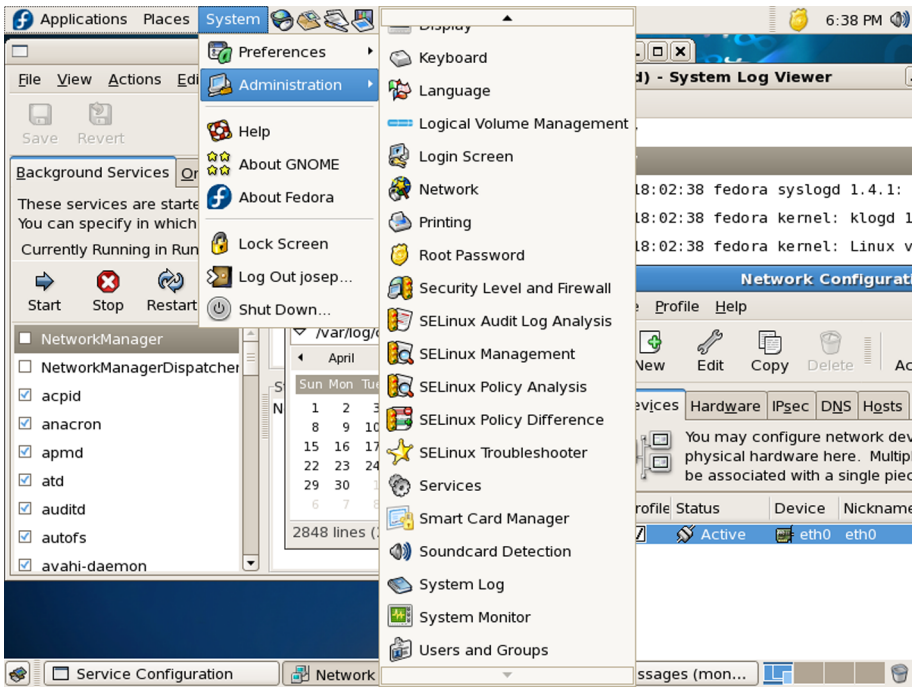


Figura 1. Algunes utilitats gràfiques d'administració en Fedora.

1.6. Altres eines

En l'espai limitat d'aquesta unitat no es poden arribar a comentar totes aquelles eines que ens poden aportar beneficis per a l'administració. Esmentarem algunes de les eines que podríem considerar bàsiques:

Nota

Consulteu les pàgines *man* de les comandes, o una referència d'eines com [Stu01].

- **Les múltiples comandes d'utilitats UNIX bàsiques**: `grep`, `awk`, `sed`, `find`, `diff`, `gzip`, `bzip2`, `cut`, `sort`, `df`, `du`, `cat`, `more`, `file`, `which`...

- **Els editors**, imprescindibles per a qualsevol tasca d'edició; tenim editors com *vi*, molt utilitzat en tasques d'administració per la rapidesa d'efectuar petits canvis als fitxers. Vim és l'editor compatible amb *vi* que sol incorporar GNU/Linux. Permet sintaxi acolorida en diversos llenguatges. L'Emacs, editor molt complet, adaptat a diferents llenguatges de programació (sintaxi i modes d'edició), disposa d'un entorn molt complet i d'una versió X denominada *XEmacs*. Joe és un editor compatible amb Wordstar. I molts d'altres...

- **Llenguatges de tipus *script***, útils per a administració, com Perl, molt adequat per a tractament d'expressions regulars i anàlisi de fitxers (filtratge, ordenació, etc.); PHP, llenguatge molt utilitzat en entorns web; Python, un altre llenguatge que permet fer prototips ràpids d'aplicacions...

- **Eines de compilació i depuració de llenguatges d'alt nivell**: GNU GCC (compilador de C i C++, entre d'altres), GDB (depurador), XXGDB (interfície *X* per a GDB), DDD (depurador per a diversos llenguatges).

2. Distribucions: particularitats

Intentem destacar ara algunes diferències tècniques menors (que com més va es redueixen més) en les distribucions (Fedora/Red Hat i Debian) [Mor03] [Soy12] utilitzades, que anirem veient amb més detall al llarg de les unitats, a mesura que vagin apareixent.

- **Ús del gestor d'arrencada GRUB** (una utilitat GNU). A diferència de versions antigues de la majoria de distribucions, que solien usar LILO, Fedora utilitza GRUB. GRUB (Grand Unified Bootloader) té una configuració en mode text (normalment en /boot/grub/grub.conf) bastant senzilla, i que es pot modificar en l'arrencada. És potser més flexible que LILO. Últimament les distribucions tendeixen a l'ús de GRUB; Debian també l'inclou ja per defecte. La majoria de distribucions actuals han migrat ja a GRUB 2, que inclou un sistema més modular de configuració, i un suport ampliat de paràmetres de *kernel* Linux, i suport millorat per a diversos sistemes operatius. El gestor GRUB 2 disposa del fitxer principal de configuració normalment a /etc/grub/grub.cfg o /etc/grub2/grub.cfg.

- **Gestió d'alternatives**. En el cas que hi hagi més d'un programari equivalent present per a una tasca concreta, mitjançant un directori (/etc/alternatives) s'indica quina és l'alternativa que s'usa. Aquest sistema es va agafar de Debian, que en fa un ús ampli en la seva distribució.

- **Programa d'escolta de ports TCP/IP basat en *xinetd***. En /etc/xinetd.d podem trobar, de manera modular, els fitxers de configuració per a alguns dels serveis TCP/IP, juntament amb el fitxer de configuració /etc/xinetd.conf. En els sistemes UNIX clàssics, el programa utilitzat és l'*inetd*, que tenia un únic fitxer de configuració en /etc/inetd.conf, com el cas, per exemple, de la distribució Debian, que utilitza *inetd* (en les versions *stable*), i deixa *xinetd* com a opció. En les darreres distribucions Fedora, xinetd ha esdevingut opcional, no hi és per defecte, però pot instal·lar-se *a posteriori* des d'un repositori. També degut a l'impacte de systemd a GNU/Linux, moltes de les distribucions, com veurem, han migrat la gestió dels serveis de l'operatiu cap a aquest nou gestor de serveis.

- **Alguns directoris de configuració especials**: /etc/profile.d, arxius que s'executen quan un usuari obre un intèrpret d'ordres; /etc/xinetd.d, configuració d'alguns serveis de xarxa; /etc/sysconfig, dades de configuració de diversos aspectes i serveis del sistema; /etc/cron., diversos directoris en què s'especifiquen treballs per fer periòdicament (mitjançant *crontab*); /etc/pam.d, en què *pam* són els denominats *mòduls d'autenticació*, en cada un dels arxius del qual es configuren permisos per al programa o servei

particular; /etc/logrotate.d, configuració de rotació (quan cal netejar, comprimir, etc.) d'alguns dels fitxers de registre per a diferents serveis.

En el cas de Debian:

- **Sistema d'empaquetament propi basat en els paquets DEB**, amb eines de diversos nivells per a treballar amb els paquets com: *dpkg*, *apt-get*, *dselect*, *tasksel*.

- **Debian segueix l'FHS**, sobre l'estructura de directoris, i afegeix alguns directoris particulars en /etc, com per exemple: /etc/default, arxius de configuració, i valors per defecte per a alguns programes; /etc/network, dades i guions de configuració de les interfícies de xarxa; /etc/dpkg i /etc/apt, informació de la configuració de les eines de gestió de paquets; /etc/alternatives, enllaços als programes per defecte, en aquells en què hi ha (o hi pot haver) diverses alternatives disponibles.

- **Sistema de configuració** de molts paquets de programari per mitjà de l'eina *dpkg-reconfigure*. Per exemple:

```
dpkg-reconfigure locals
```

  permet escollir les configuracions regionals d'idiomes i els jocs de caràcters que estaran disponibles, o:

```
dpkg-reconfigure tzdata
```

  ens permet reconfigurar la zona horària per defecte.

- **Utilitza configuració de serveis TCP/IP per *inetd***, amb la configuració en el fitxer /etc/inetd.conf. Disposa d'una eina *update-inetd* per a inhabilitar o crear entrades de serveis. Es pot fer servir xinetd com a alternativa per a controlar els serveis que hi estiguin suportats.

- Alguns directoris de configuració especials: /etc/cron, diversos directoris en què s'especifiquen treballs per fer periòdicament (mitjançant *crontab*); /etc/pam.d, en què *pam* són mòduls d'autenticació.

- Debian disposa de repositoris, *free*, *senar-free* i *contrib*, per a separar la procedència del programari inclòs. Mentre que Fedora disposa d'un sol genèric (en alguns casos s'afegeixen repositoris per defecte d'*updates* i extres).

- Debian tradicionalment ha usat arrencada de sistema amb sysvinit, però en les darreres versions s'està migrant progressivament cap al sistema systemd.

3. Nivells d'arrencada i serveis

Un primer punt important en l'anàlisi del comportament local del sistema és el seu funcionament en els anomenats *nivells d'execució* (o *runlevels*), que determinen (en el nivell) el mode actual de treball del sistema i els serveis que es proporcionen [Wm02].

Nota

Actualment, en les distribucions GNU/Linux coexisteixen diferents models d'arrencada del sistema i serveis. Els més destacables són *sysvinit*, *upstart* i *systemd*.

Un servei és una funcionalitat proporcionada per la màquina, normalment basada en dimonis (o processos en segon pla d'execució, que controlen peticions de xarxa, activitat del maquinari, o altres programes que controlin alguna tasca).

A partir d'aquest punt, descriurem el mètode clàssic d'arrencada per defecte del sistema i dels seus serveis (en anglès, *init system*), aquest mètode clàssic, denominat *sysvinit* (o *init* de tipus SysV), és el que clàssicament han seguit la majoria d'UNIX comercials, i al començament la majoria de distribucions GNU/Linux, progressivament s'està migrant a nous mètodes com *upstart* i *systemd*. És necessari conèixer els diversos mètodes d'arrencada, ja que alguns mantenen certa compatibilitat, i en les distribucions Debian/Fedora podem trobar diferents mètodes d'arrencada depenent de la versió de la distribució.

3.1. Arrencada SysVinit

A sysvinit, l'activació o parada de serveis es fa mitjançant la utilització de *scripts*. La majoria dels serveis estàndard, que solen tenir la seva configuració en el directori /etc, se solen controlar mitjançant els *scripts* presents en /etc/init.d/. En aquest directori solen aparèixer *scripts* amb noms similars al servei als quals estan destinats, i se solen acceptar paràmetres d'activació o parada. Es fa:

```
/etc/init.d/servei start
```

Arrencada del servei.

```
/etc/init.d/servei stop
```

Parada del servei.

```
/etc/init.d/servei restart
```

Aturada i arrencada posterior del servei.

Quan un sistema GNU/Linux arrenca, primer es carrega el nucli del sistema, i després s'inicia el primer procés, denominat *init*, que és el responsable d'executar i activar la resta del sistema, mitjançant la gestió dels nivells d'execució (o *runlevels*).

Nota

Podem utilitzar les ordres *runlevel* o *who -r* per a conèixer el *runlevel* actual en què ens trobem.

Un nivell d'execució és senzillament una configuració de programes i serveis que s'executaran orientats a un determinat funcionament.

Els nivells típics solen ser (hi pot haver diferències en l'ordre, en especial en els nivells 2-5; a la taula hi ha la configuració en Fedora, i la recomanada per l'estàndard LSB):

Runlevel	**Funció**	**Descripció**
0	Parada	Finalitza serveis i programes actius, i també desmunta sistemes d'arxius actius i para la CPU.
1	Monousuari	Finalitza la majoria de serveis, i permet només l'entrada de l'administrador (*root*). S'usa per a tasques de manteniment i correcció d'errors crítics.
2	Multiusuari sense xarxa	No s'inicien serveis de xarxa, i permet només entrades locals en el sistema.
3	Multiusuari	Inicia tots els serveis excepte els gràfics associats a X Window.
4	Multiusuari	No se sol usar, típicament és igual que el 3.
5	Multiusuari X	Igual que el 3, però amb suport X per a l'entrada d'usuaris (connexió gràfica).
6	Reinici	Para tots els programes i serveis. Reinicia el sistema.

Al contrari, es pot assenyalar que Debian usa un model en què pràcticament els nivells 2-5 són equivalents, i fan exactament la mateixa funció (encara que podria ocórrer que en alguna versió això canviï per coincidir amb l'estàndard LSB).

Aquests nivells solen estar configurats en els sistemes GNU/Linux (i UNIX) per dos sistemes diferents: el BSD i el System V (de vegades abreujat com a sysinitV o sysV). En el cas clàssic de les versions antigues de Fedora i Debian, s'utilitzava el sistema System V, que és el que mostrarem en aquesta secció (tot i que aquestes distribucions estan migrant, o ja ho han fet, cap a systemd, del qual en parlarem més endavant), però alguns UNIX utilitzen el model BSD.

En el cas del model *runlevel* de SysV, quan el procés *init* (normalment, és el procés amb identificador PID=1) arrenca, utilitza un fitxer de configuració anomenat /etc/inittab per a decidir el mode d'execució en el qual entrarà. En aquest fitxer es defineix el *runlevel* per defecte (*initdefault*) en arrencada (per a la instal·lació per defecte, en Fedora el 5, en Debian el 2) i una sèrie de serveis de terminal per activar per a atendre l'entrada de l'usuari.

Després, el sistema, segons el *runlevel* escollit, consulta els fitxers continguts en /etc/rcn.d, en què *n* és el número associat al *runlevel* (nivell escollit), en el qual es troba una llista de serveis per activar o parar en cas que arrenquem en el *runlevel* o l'abandonem. Dins del directori trobarem una sèrie de *scripts* o enllaços als *scripts* que controlen el servei.

Cada *script* té un nom relacionat amb el servei, una *S* o *K* inicial que indica si és l'*script* per a iniciar (*S*) o matar (*K*) el servei, i un número que reflecteix l'ordre en el qual s'executaran els serveis.

Una sèrie d'instruccions de sistema serveixen d'ajuda per a manejar els nivells d'execució. Es poden esmentar:

- **Els *scripts***, que ja hem vist, en /etc/init.d/ ens permeten arrencar, parar o reiniciar serveis individuals.

- ***telinit/init*** ens permet, com a *root*, canviar de nivell d'execució; només n'hem d'indicar el número. Per exemple, necessitem fer una tasca crítica en *root*; sense usuaris treballant, podem fer un *telinit 1* (també es pot usar *S*) per a passar a *runlevel* monousuari, i després de la tasca un *telinit 3* per a tornar a multiusuari. També es pot utilitzar la instrucció *init* per a la mateixa tasca, encara que *telinit* aporta algun paràmetre extra. Per exemple, el reinici típic d'un sistema UNIX es feia amb *sync*; *sync*; *sync*; *init 6*. La instrucció *sync* força el buidatge de les *memòries intermèdies* del sistema d'arxius, i després reiniciem en *runlevel* 6.

- ***shutdown*** permet parar (*'-h'* de *halt*) o reiniciar el sistema (*'–r'* de *reboot*). Es pot donar un interval de temps o fer-ho immediatament. Per a aquestes tasques també hi ha les instruccions *halt/poweroff* i *reboot*. Depenent de si el sistema té suport de gestió d'energia poden tenir efectes diferents (parar les CPU solament, en *halt*, o en *poweroff* enviar l'ordre ACPI de desconnectar l'alimentació elèctrica). Els reinicis bàsicament controlen un init 6.

- ***wall*** permet enviar missatges d'advertència als usuaris del sistema. Concretament, l'administrador pot anunciar que es pararà la màquina en un determinat moment. Instruccions com *shutdown* el solen utilitzar de manera automàtica.

- ***pidof*** permet esbrinar el PID (*process ID*) associat a un procés. Amb *ps* obtenim la llista de processos, i si volem eliminar un servei o procés mitjançant *kill*, en necessitarem el PID.

Respecte a tot el model d'arrencada, les distribucions presenten algun petit canvi:

- **Fedora/Red Hat** (només les versions amb suport SysVinit): el *runlevel* 4 no té un ús declarat. Els directoris /etc/rcn.d existeixen com a enllaços

cap a subdirectoris de /etc/rc.d, on són centralitzats els *scripts* d'arrencada. Els directoris són, així: /etc/rc.d/rcn.d; però com que hi ha els enllaços, és transparent a l'usuari. El *runlevel* per defecte és el 5, amb arrencada amb X. Les instruccions i fitxers relacionats amb l'arrencada del sistema són en els paquets de programari *sysvinit* i *initscripts*. Respecte als canvis de fitxers i guions en Fedora, es pot destacar: en /etc/sysconfig podem trobar arxius que especifiquen valors per defecte de la configuració de dispositius o serveis. El guió /etc/rc.d/rc.sysinit és invocat una vegada quan el sistema arrenca; el guió /etc/rc.d/rc.local s'invoca al final del procés de càrrega i serveix per a indicar inicialitzacions específiques de la màquina que ens interessin sense passar per tot el sistema següent.
L'arrencada real dels serveis es fa per mitjà dels guions emmagatzemats en /etc/rc.d/init.d. Hi ha també un enllaç des de /etc/init.d. A més, Fedora proporciona uns *scripts* d'utilitat per a manejar serveis: /sbin/service per a parar o iniciar un servei pel nom i /sbin/chkconfig per a afegir enllaços als fitxers *S* i *K* necessaris per a un servei, o l'obtenció d'informació sobre els serveis.

- **Debian** disposa d'instruccions de gestió dels *runlevels* com *update-rc.d*, que permet instal·lar o esborrar serveis engegant-los o parant-los en un o més *runlevels*. Una altra instrucció és *invoke-rc.d*, que permet les clàssiques accions d'engegar, parar o reiniciar el servei. El *runlevel* per defecte en Debian és el 2, l'X Window System no es gestiona des de /etc/inittab, sinó que hi ha el gestor (per exemple, *gdm* o *kdm*), com si fos un servei més del *runlevel* 2.

3.2. Upstart

Upstart és un sistema d'arrencada basat en esdeveniments, dissenyat per a substituir el procés comentat generat per l'*init* del sistema d'arrencada *sysvinit*. Un dels principals problemes del dimoni *init* de System V és que no va ser pensat per a maquinari modern, que permet dispositius removibles, o dispositius que puguin ésser connectats/substituïts en calent (*hotplug*).

Nota

Podeu obtenir més informació sobre upstart a "The Upstart Cookbook":
http://upstart.ubuntu.com/cookbook.

Aquest sistema va ser dissenyat per l'empresa Canonical, per a incorporar-se a l'Ubuntu 6.10 (2006) i ha estat incorporat posteriorment a les diferents versions d'Ubuntu, i en Fedora (a partir de la versió 9) i OpenSUSE (11.3). Debian manté l'anterior sistema (sysvinit en la versió estable) amb certes modificacions fins a les últimes versions, però havia de migrar progressivament al nou sistema (a causa dels intents, per totes dues parts, per mantenir el màxim nivell de semblança amb Ubuntu i Debian).

El 2014 totes dues distribucions van decidir migrar a systemd de manera progressiva, s'espera que aquestes distribucions vagin adaptant per complet el model systemd (amb alguns intents puntuals de mantenir al mateix temps la comptatibilitat amb SysV i Upstart). Cal assenyalar que *Upstart* en algunes versions antigues de distribucions GNU/Linux encara es fa servir, i en les nostres

tasques d'administració ens podem trobar amb versions antigues, però suportades (com algunes de les LTS) de Debian i Ubuntu que continuïn suportant *upstart*.

Identificació de tipus d'arrencada

Podem detectar la presència d'*upstart* per l'existència (en la llista de l'ordre ps) de diversos *daemons* denominats amb *upstart-* com a prefix.

Les distribucions amb suport *systemd* solen presentar el *daemon systemd* amb PID=1, i solen existir (en ps) diverses instàncies (filles) del mateix *daemon*. En algunes distribucions es manté un procés *init* amb PID=1, però existeixen diferents processos fill com el *daemon systemd* o variacions d'aquest.

Els sistemes amb sysvinit, no presenten cap dels anteriors, solament el procés init amb PID=1, i els altres serveis són arrencats des d'init.

Encara que es pot destacar que Upstart (almenys de moment) manté una compatibilitat total amb els *scripts* inicials del procés d'*init* (vistos en aquest apartat, "Nivells d'arrencada i serveis"), el que es modifica és el procés intern que gestiona els serveis d'arrencada, que passa a ser controlat pel dimoni d'Upstart.

El dimoni *init* d'Upstart està basat en esdeveniments que permeten executar programes (o accions) específiques a partir de canvis en el sistema, que solen ser típicament (respecte a l'*init* System V) *scripts* de parada o arrencada de serveis. En System V això només es produïa per canvi de nivell d'execució; en Upstart ho pot provocar qualsevol esdeveniment que succeeixi dinàmicament en el sistema. Per exemple, Upstart se sol comunicar (per esdeveniments) amb dimonis (com *udev*) que controlen els canvis del maquinari de la màquina; per exemple, si apareix/desapareix un maquinari determinat, es poden activar o desactivar els serveis de sistema associats a aquest maquinari.

La gestió de les tasques d'upstart està centralitzada en */etc/init* i, a mesura que es vagi migrant (en les distribucions) cap a Upstart, aquest directori anirà reemplaçant els continguts de */etc/init.d* i els directoris associats als nivells d'execució */etc/rc<n>.d*.

En la terminologia d'Upstart, un esdeveniment és un canvi d'estat del sistema que pot ser informat al procés *init*. Un *job* és un conjunt d'instruccions que *init* llegeix, típicament o bé un binari o un *shell script*, juntament amb el nom de l'esdeveniment. Així, quan l'esdeveniment apareix en el sistema, Upstart llança el *job* associat (els *jobs* del sistema els podem trobar definits en el directori esmentat */etc/init*). Els *jobs* poden ser de dos tipus, *task* o *service*. Una *task* és un *job* que fa la seva tasca i retorna a un estat d'espera quan acaba; un *service* no acaba per si mateix, sinó per una decisió basada en un esdeveniment o bé una intervenció manual de parada. En */etc/init* podem observar els *jobs* definits en el sistema upstart, i examinar la seva configuració, a mode d'exemple. En *upstart* també és possible definir *jobs* d'usuari, col·locant-los en un directori ".init" dins del compte de l'usuari.

Així, el procés general *init* d'Upstart és un funcionament de màquina d'estats; manté control de l'estat dels treballs (*jobs*) i els esdeveniments que dinàmicament apareixen, i gestiona els canvis d'estat dels treballs en reacció als esdeveniments apareguts.

Hi ha una sèrie de treballs (*jobs*) predefinits amb noms *rc* que serveixen per a emular els canvis de *runlevel* de System V, per a així emular el concepte de nivell d'execució.

Per a examinar el sistema Upstart, podem usar com a eina bàsica la instrucció *initctl,* que ens permet (entre altres tasques):

- initctl list (mostrar els *jobs* i el seu estat).
- initctl emit EVENT (emetre manualment esdeveniments concrets).
- Initctl start/stop/status JOB (l'acció sobre un *job*). Hi ha altres abreviatures d'ordres referents a JOB: start, stop, restart, reload, status.

Per finalitzar, podem observar que Upstart podrà tenir certa evolució futura, ja substituint completament el Sysvinit o evolucionant a noves funcionalitats, ja que el sistema basat en màquina d'estats és molt complet per integrar diferents funcionalitats que ara estan repartides per diversos elements del sistema. Es preveu que podria arribar a reemplaçar la programació de tasques periòdiques de sistema (usada actualment, com veurem, per elements com *cron*, *anacron*, *atd*) i possiblement gestions diverses de dimonis de xarxa (com les fetes per *inetd* o *xinetd*).

Upstart és un aspecte al qual haurem de parar atenció en algunes distribucions GNU/Linux antigues que continuen amb suport. D'altra banda, Sysvinit es continua utilitzant en la majoria de UNIX de propietat i en les versions empresarials de GNU/Linux.

3.3. Systemd

El nou sistema d'arrencada systemd, s'està imposant recentment a causa de la decisió de Debian i de les distribucions associades (Ubuntu, Mint) d'escollir aquesta opció, així com diverses branques del Fedora (RHEL, i CentOS) també han pres aquest camí.

Nota

Lloc oficial systemd: https://freedesktop.org/wiki/Software/systemd/

Systemd, és un *daemon* de gestió del sistema dissenyat per Linux que substitueix al procés *init*, i per tant es col·loca com a procés amb PID = 1 en l'arrencada del sistema Linux. També s'utilitza el mateix nom, *systemd*, per a denominar el paquet complet de components del sistema, que inclouen com un d'ells el *daemon systemd*.

Systemd s'ha desenvolupat per Gnu/Linux com a substitut del sistema init heretat de SysV (sysvinit). Igual que init, el *daemon systemd* controla altres *daemons* que s'encarrega d'iniciar, deixant-los com processos en segon pla; systemd és el primer *daemon* a arrencar i l'últim a acabar durant el procés d'apagament.

Els objectius principals de disseny van ser intentar expressar dependències entre *daemons* en temps d'arrencada, per a així determinar tots els processos que podien fer-se concurrentment o en paral·lel durant el procés d'arrencada, i en general reduir el cost computacional cap al *shell*.

Aquest procés d'integració de systemd també ha rebut força crítiques de la comunitat perquè és un intent d'aglutinar un gran nombre de funcionalitats en un únic sistema (en sentit contrari a la filosofia UNIX, d'unes petites eines interconnectades cadascuna realitzant tasques concretes i ben definides).

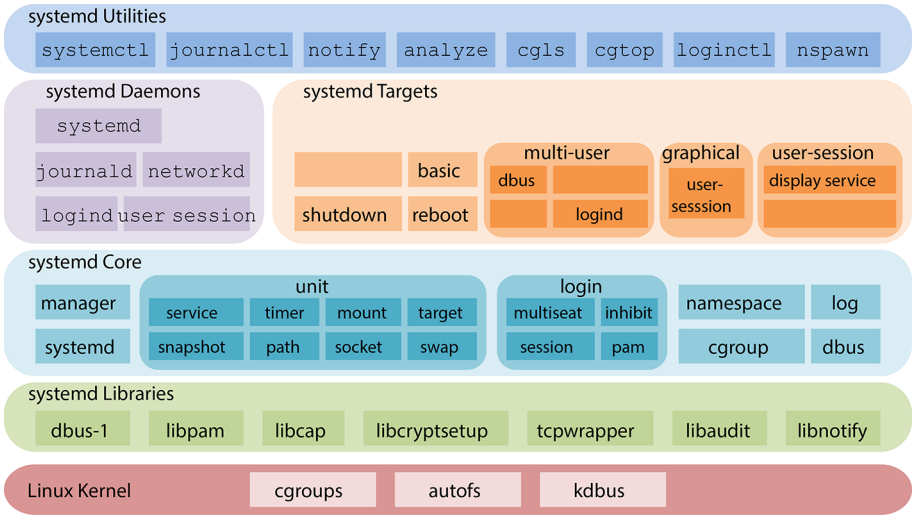
En aquests moments, systemd ha aglutinat o ofereix alternatives a tot un seguit de *daemons* clàssics, com udev, el sistema d'arrencada sysvinit, inetd, acpid, syslog, o cron entre altres. A més, últimament també s'han agregat components de configuració de xarxes, entre altres, afegint encara més components i funcionalitats disperses, a un sol bloc de funcionalitat, tot això controlat per systemd.

Nota

El lloc "Boycott systemd" va recollir gran part d'aquestes crítiques a systemd:

http://without-systemd.org/wiki/index.php/Local_copy_of_boycottsystemd.org_archive

Es poden veure més arguments i alternatives a Systemd a: http://without-systemd.org/wiki/index.php/Main_Page



Alguns components systemd. Font: figura provinent de la Wikipedia a l'entrada *systemd* sota llicència CC.

A systemd podem distingir, entre d'uns altres, components com ara:

- Fitxers **Unit**: on es recullen les instruccions d'inicialització per a cada *daemon* en un fitxer de configuració (anomenat "unit file"). Hi ha diferents tipus com: *service, socket, device, mount, automount, swap, target, path, timer, snapshot, slice* i *scope*.

- Components **core**:
  - *systemd*, el *daemon* principal del sistema *systemd*, que actua de gestor de serveis per GNU/Linux.

  - *systemctl*, com a ordre principal per controlar l'estat del sistema i del gestor de serveis, així com per gestionar les diferents *units* del sistema. Veurem *a posteriori* la gestió de serveis mitjançant aquesta ordre.

  - *systemd-analize*, per a l'anàlisi de les prestacions del sistema d'arrencada i per obtenir estadístiques sobre ell. Per exemple, amb *systemd-analize blame* obtindrem el temps en l'arrencada de cada *unit*. També és molt interessant l'opció *plot*, on obtindrem un fitxer gràfic (svg) amb el diagrama temporal del desplegament de les *units* al llarg de l'arrencada.

  - Utilitats com *systemd-cgls* i *systemd-cgtop* permeten seguir l'ús de grups de control per part de systemd, que els usa (el subsistema *cgroups* del kernel Linux) per controlar el seguiment dels processos, en lloc d'usar identificadors de processos (PID). Systemd també usa els *cgroups* per a mecanismes de contenidors Linux, mitjançant ordres com *systemd-nspawn* i *machinectl*.

- Un seguit de **components accessoris** (poden variar depenent de la versió actual del sistema systemd), la majoria dels quals són *daemons* (els finalitzats amb *d* en la següent llista) de la forma *systemd-nom*:
  - **consoled**: *systemd-consoled* proporciona un *daemon* per a la gestió de consoles textuals, amb l'objectiu de substituir les consoles/terminals virtuals de GNU/Linux.

  - **journald**: *system-journald* és el *daemon* responsable del control de *log* d'esdeveniments, gestionat en systemd per fitxers *log* binaris de només lectura. És possible gestionar els *logs* dins de systemd, amb *system-journald*, *syslog-ng* o *rsyslog*, depenent de la decisió de l'administrador.

  - **logind**: és un *daemon* encarregat de l'accés dels usuaris del sistema.

  - **networkd**: és un *daemon* encarregat de gestionar configuracions de les interfícies de xarxa.

  - **timedated**: s'encarrega de controlar les opcions relacionades amb l'hora del sistema i les zones horàries.

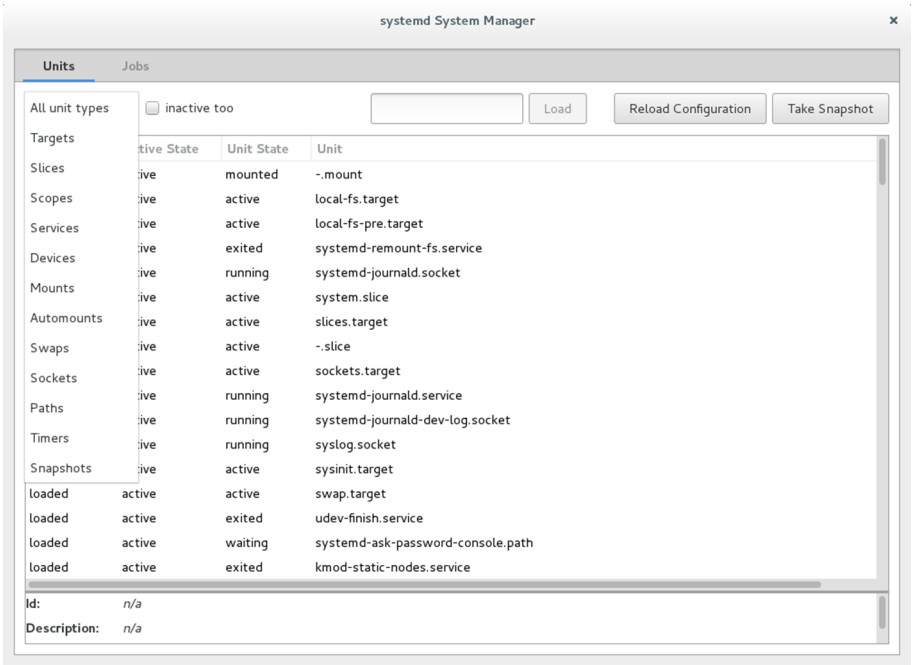
– **udevd**: és el *daemon* gestor de dispositius que s'encarrega de gestionar dinàmicament el directori */dev* del sistema, i les accions d'afegir o treure dispositius des d'usuari, així com control de càrrega de *firmwares*.

– **libudev**: en aquest cas, una llibreria estàndard per utilitzar *udev*, que permet que aplicacions de tercers puguin accedir a consultar els recursos de tipus *udev*.

– **systemd-boot**: un gestor d'arrencada del sistema (procedeix d'un ja existent prèviament *gummiboot*), amb suport de *firmare* UEFI, en principi, pensat com a substitut lleuger de GNU Grub.

A banda de tot això, es disposa també d'alguns *frontends* gràfics per gestionar els serveis i consultar les *units* disponibles, com *systemd-ui* (per a entorns Gnome, també conegut com *systemadm*) i *systems-kcm* (per a entorns KDE). Encara que aquests entorns no solen mantenir-se al mateix ritme que les versions de systemd, que es concentren en les eines textuals, com *systemctl* i *systemd-analize*.

Systemd introdueix el concepte d'*units*, que són representades per fitxers de configuració, i bàsicament representen informació sobre un servei de sistema, *sockets* d'escolta, estats de sistema, o altres objectes rellevants per a l'arrencada.



system-ui (systemadm) mostrant en una distribució Debian part de les units del sistema, així com el menú de tipus d'aquestes.

Els fitxers podem trobar-los en els llocs següents (per ordre invers de precedència dels fitxers, si es donen fitxers comuns a diferents nivells):

- */usr/lib/systemd/system/* unitats de sistema (normalment procedents de paquets).

- */run/systemd/system/* unitats creades en temps d'execució.

- */etc/systemd/system/* unitats creades i gestionades per l'administrador.

Quant als tipus d'unitats podem trobar (entre altres):

Unitat de	Extensió de fitxer	Descripció
Servei	*.service*	Un servei de sistema
Target	*.target*	Grup d'unitats systemd
Dispositiu	*.device*	Fitxer de dispositiu
Automount point	*.automount*	Un punt d'automuntatge d'un fs
Mount	*.mount*	Un punt de muntatge d'fs
Snapshot	*.snapshot*	Un estat desat del gestor systemd
Socket	*.socket*	Un *socket* de comunicació entre processos
Timer	*.timer*	Un temporitzador systemd

Per a la gestió de serveis del sistema, s'utilitzen les unitats de servei (.service), que s'utilitzen per a propòsits semblants als antics fitxers de serveis presents a /etc/init.d/. Systemd disposa de l'ordre systemctl, que s'utilitza en combinacions amb les opcions: *start*, *stop*, *restart*, *enable*, *disable*, *status*, *is-enabled* del servei corresponent:

```
# systemctl stop httpd.service
```

Per exemple, per a parar un servei, podem ometre l'extensió .service, ja que és la que assumeix *systemctl* per defecte. Respecte a altres tecles d'ordre semblants de sistemes d'arrencada anteriors (init o upstart), en algunes distribucions es mantenen les tecles d'ordre com *service* o *chkconfig*, aquestes en una distribució basada en systemd haurien d'evitar-se i usar mentre sigui possible *systemctl*.

Un tecla d'ordre com:

```
# systemctl list-units --type service --all
```

Ens proporciona l'estatus de tots els serveis (o unitats de tipus servei en terminologia systemd).

```
# systemctl list-units-files --type service
```

Ens proporciona el nom del servei amb nom complet i informació sobre si va actuar o no. Per a un servei concret disposem del següent:

```
# systemctl status name.service
```

Per preguntar si el servei està actiu:

```
# systemctl is-active name.service
```

Per preguntar si el servei està habilitat:

```
# systemctl is-enable name.service
```

Pel que fa als nivells d'execució (*runlevels*), systemd proporciona diferents unitats equivalents, en format *target*, així hi ha unitats target denominades *poweroff*, *multi-user*, *graphical* i *reboot*. Aquestes es poden escriure en una llista mitjançant:

```
# systemctl list-units --type target
```

```
# systemctl isolate name.target
```

En el segon cas ens permet canviar el *target* actual. I conèixer el per defecte (o col·locar-lo amb *set-default*):

```
# systemctl get-default
```

Per a les diferents opcions de parada i reinici de màquina es disposa de les opcions per a systemctl: *halt*, *poweroff*, *reboot*, *suspend* i *hibernate*.

Una particularitat de systemd és que també pot actuar en màquines remotes per mitjà de connexió ssh, per exemple podem llançar un tecla d'ordre amb:

```
# systemctl --host usuari@host_remot tecla d'ordre
```

Per a investigar errors de diferents unitats, podem fer el següent:

```
# systemctl --state=failed
     Ens proporciona unitats que han provocat fallades
# systemctl status  unitat_amb_fallades.extensio
     Preguntem per l'estat amb el qual ja podem examinar si ens proporciona alguna informació addicional.
```

Com veurem després a partir del PID del procés involucrat (si no el tenim podem intentar un *restart*), també podem examinar el *journal* corresponent amb:

```
# journalctl -b _PID=pid_trobat
```

Systemd té moltes possibilitats, depenent de les unitats utilitzades i els múltiples components de què disposa, alguns els comentarem en seccions posteriors.

4. Observar l'estat del sistema

Una de les principals tasques de l'administrador (*root*) en el seu dia a dia serà verificar el funcionament correcte del sistema i vigilar l'existència de possibles errors o de saturació dels recursos de la màquina (memòria, discos, etc.). Passarem a detallar, en els apartats següents, els mètodes bàsics per a examinar l'estat del sistema en un determinat moment i dur a terme les accions necessàries per a evitar problemes posteriors.

Al taller final d'aquest mòdul farem un examen d'un sistema exemple, per veure algunes d'aquestes tècniques.

4.1. Arrencada del sistema

En l'arrencada d'un sistema GNU/Linux es produeix tot un bolc d'informació interessant. Quan el sistema arrenca, solen aparèixer les dades de detecció de les característiques de la màquina, la detecció de dispositius, l'arrencada de serveis de sistema, etc., i s'esmenten els problemes apareguts. A més es crea un cert nombre de sistemes de fitxers virtuals, que generen i permeten manipular diferent quantitat d'informació de l'espai d'usuari (*user-space*) i de l'espai de *kernel* (*kernel-space*).

En la majoria de les distribucions, molta part d'aquesta informació sobre l'arrencada (com a esdeveniments) es pot veure a la consola del sistema directament durant el procés d'arrencada. Tanmateix, o la velocitat dels missatges o algunes distribucions modernes que els oculten darrere caràtules gràfiques, poden impedir seguir els missatges correctament, amb la qual cosa necessitarem una sèrie d'eines per a aquest procés.

Bàsicament, podem utilitzar:

- **Instrucció *dmesg***: dóna els missatges de l'última arrencada del nucli.

- **Fitxer /var/log/messages**: registre general del sistema, que conté els missatges generats pel nucli i altres dimonis (hi pot haver multitud d'arxius diferents de *log*, normalment en /var/log, depenent de la configuració del servei *syslog*). En algunes distribucions modernes, el sistema de *logs* (syslog per defecte en la majoria) s'ha substituït per rsyslog, que generalment té configurat el *log* de sistema principal a */var/log/syslog*. En altres distribucions que utilitzen *systemd* com a sistema d'arrencada, pot ser que no vegem cap dels fitxers anteriors (*messages* o syslog), *systemd* també s'encarrega del *log* dels esdeveniments amb un component/*daemon* denominat *Journald*.

En aquest cas es disposa de l'ordre *journalctl* per a obtenir en forma textual la llista d'esdeveniments del *log* del sistema.

- **Instrucció *uptime***: indica quant temps fa que el sistema és actiu.

A més, com hem dit, en temps d'arrencada es generen una sèrie de sistemes de fitxers virtuals que ens permeten accedir a informació del procés *de boot* i a l'estat dinàmic del sistema, tant de l'espai d'usuari com del de *kernel* i dels seus diversos components. Podem detectar aquests sistemes muntats examinant els sistemes muntats amb l'ordre *df* o amb *mount* directament. Entre els sistemes que trobarem cal destacar:

- **Sistema /proc**: pseudosistema de fitxers (*procfs*) que utilitza el nucli per a emmagatzemar la informació de processos i de sistema. Típicament es mapa en un punt de muntatge */proc* en temps d'arrencada. Es pot usar per a obtenir informació del sistema i canviar diversos paràmetres del *kernel* en temps d'execució mitjançant l'ordre *sysctl*.

- **Sistema /sys**: pseudosistema de fitxers (*sysfs*) que va aparèixer amb la branca 2.6.x del nucli, amb objectiu de proporcionar una manera més coherent d'accedir a la informació dels dispositius i els seus controladors (*drivers*). El sistema virtual de fitxers Sysfs es munta en */sys*. Bàsicament aquest sistema permet veure i manipular objectes i dades des d'espai d'usuari, que són creats i destruïts des d'espai de *kernel*.

- **Sistema /config**: *configfs* és similar a *sysfs*, però en aquest cas és complementari, ja que permet crear, gestionar i destruir objectes/dades d'espai de *kernel* des d'espai d'usuari. Típicament el trobem muntat a */config* o en altres ocasions a */sys/kernel/config*.

- **Sistema de fitxers tmpfs**: típicament s'usa per a muntar espai d'emmagatzematge temporal de manera virtual, en memòria volàtil, sense emmagatzematge en disc. Diverses distribucions l'usen per a muntar el directori */run* i subdirectoris d'aquest.

- **Sistema de fitxers devtmpfs**: es munta a */dev*, quan arrenca la màquina, i proporciona la informació dels dispositius que després utilitza el gestor de dispositius udev (mitjançant el seu *daemon* udevd). Ara udev forma part del sistema d'arrencada *systemd*.

4.2. Nucli: directori /proc

El nucli, durant la seva arrencada, posa en funcionament un sistema de fitxers virtual profcs, el munta en */proc*, on bolca la informació que recopila de la màquina, i també moltes de les seves dades internes, durant l'execució. El

directori /proc està implementat sobre memòria i no es desa en disc. Les dades contingudes són tant de naturalesa estàtica com dinàmica (varien durant l'execució).

Cal tenir en compte que com que /proc és fortament dependent del nucli, això propicia que la seva estructura depengui del nucli de què disposa el sistema i l'estructura i els fitxers inclosos poden canviar.

Una de les característiques interessants és que en el directori /proc podrem trobar les imatges dels processos en execució, juntament amb la informació que el nucli maneja sobre aquests processos. Cada procés del sistema es pot trobar en el directori /proc/<pidproces>, en què hi ha un directori amb fitxers que representen el seu estat. Aquesta informació és bàsica per a programes de depuració, o bé per a les instruccions mateixes del sistema, com *ps* o *top,* que la poden utilitzar per a veure l'estat dels processos en execució. En general, moltes de les utilitats del sistema consulten la informació dinàmica del sistema des de /proc (en especial, algunes utilitats proporcionades amb el paquet *procps).*

Nota

El directori /proc és un recurs extraordinari per a obtenir informació de baix nivell sobre el funcionament del sistema. Moltes instruccions de sistema s'hi basen per a les seves tasques.

D'altra banda, en /proc podem trobar altres fitxers d'estat global del sistema. Comentem, de manera breu, alguns fitxers que podrem examinar per a obtenir informació important:

Arxiu	Descripció
/proc/bus	Directori amb informació dels busos PCI i USB
/proc/cmdline	Línia d'arrencada del nucli
/proc/cpuinfo	Informació de la CPU
/proc/devices	Llista de dispositius del sistema de caràcters o blocs
/proc/driver	Informació d'alguns mòduls de maquinari
/proc/filesystems	Sistemes de fitxers habilitats en el nucli
/proc/scsi	SCSI, en el fitxer scsi característiques de discos
/proc/interrups	Mapa d'interrupcions de maquinari (IRQ) utilitzades
/proc/ioports	Ports E/S utilitzats
/proc/meminfo	Dades de l'ús de la memòria
/proc/modules	Mòduls del nucli
/proc/mounts	Sistemes d'arxius muntats actualment
/proc/net	Directori amb tota la informació de xarxa
/proc/scsi	Directori de dispositius SCSI, o IDE emulats per SCSI
/proc/sys	Accés a paràmetres del nucli configurables dinàmicament
/proc/version	Versió i data del nucli

A partir de la branca 2.6 del nucli, es va iniciar una transició progressiva de *procfs* (/proc) a *sysfs* (/sys) amb l'objectiu de moure tota aquella informació que no estigui relacionada amb processos, en especial dispositius i els seus controladors (mòduls del nucli) cap al sistema /sys.

4.3. Nucli: /sys

El sistema *sys* s'encarrega de fer disponible la informació de dispositius i controladors, informació de la qual disposa el nucli, a l'espai d'usuari, de manera que altres API o aplicacions puguin accedir d'una manera flexible a la informació dels dispositius (o els seus controladors). Sol ser utilitzada per capes i el servei *udev* per al monitoratge i la configuració dinàmica dels dispositius.

Dins del concepte de *sys* hi ha una estructura de dades en arbre dels dispositius i controladors (diguem-ne, el model conceptual fix), i després s'hi accedeix per mitjà del sistema de fitxers *sysfs* (l'estructura del qual pot canviar entre versions).

Quan es detecta o apareix en el sistema un objecte afegit, en l'arbre del model de controladors (controladors, dispositius incloent-hi les seves diferents classes), es crea un directori en *sysfs*. La relació pare/fill es reflecteix amb subdirectoris sota /sys/devices/ (s'hi reflecteix la capa física i els seus identificadors). En el subdirectori /sys/bus es col·loquen enllaços simbòlics, reflectint la manera en la qual els dispositius pertanyen als diferents busos físics del sistema. I /sys/class mostra els dispositius agrupats d'acord amb la seva classe, com per exemple *xarxa*, mentre que /sys/block/ conté els dispositius de blocs.

Alguna de la informació proporcionada per /sys es pot trobar també en /proc, però es va considerar que aquest estava barrejant diferents coses (dispositius, processos, dades de maquinari, paràmetres del nucli) de manera no coherent, i això va ser un dels motius per a crear /sys. S'espera que, progressivament, es migri informació de /proc a /sys per a centralitzar la informació dels dispositius.

4.4. Udev: gestió de dispositius /dev

A diferència dels sistemes tradicionals, en què els nodes de dispositius presents en el directori */dev* eren considerats com un conjunt estàtic de fitxers, el sistema *udev* proporciona dinàmicament els nodes per als dispositius presents en el sistema.

Udev és el gestor de dispositius per al *kernel* Linux, que substitueix algunes iteracions anteriors, com els sistemes *hotplug* i *devfsd*, tractant la gestió de nodes de dispositiu en el directori /dev, i al mateix temps gestiona els esdeveniments generats en espai d'usuari, a causa de la nova presència (o absència) de dispositius de maquinari inclosos en el sistema, incloent-hi la cara de microprogra-

mari (*firmware*) necessària per a certs dispositius (encara que aquesta ultima funció es desplacés progressivament a l'espai de *kernel*, reservant-la als *drivers* dels dispositius).

Entre altres característiques udev, suporta:

- Denominació persistent de dispositius, per exemple evitant l'ordre d'arribada (o connexió) dels dispositius al sistema. Entre ells es proporcionen noms persistents per als dispositius d'emmagatzematge, cada disc reconegut disposa d'un ID de sistema de fitxers, el nom del disc, i la localitat física del maquinari on està connectat.

- Notificació, a sistemes externs, dels canvis dels dispositius.

- Crear un directori /dev dinàmic.

- S'executa en espai d'usuari, llevant del *kernel* la responsabilitat de nomenar els dispositius, per tant pot usar-se denominació de dispositius específics a partir de les propietats del dispositiu.

El sistema udev està compost bàsicament de tres parts:

- La biblioteca *libudev*, que permet l'accés a la informació del dispositiu. Depenent de la implementació, aquesta biblioteca pot trobar-se aïllada o, en el cas de distribucions amb arrencada systemd, haver passat a incloure's com una funcionalitat inclosa en aquest sistema.

- Un *daemon* d'espai d'usuari, *udevd*, que gestiona el directori virtual /dev. El *daemon* escolta el *socket* de comunicació entre *kernel* i espai d'usuari, per determinar quan un dispositiu és afegit o llevat del sistema, i udev gestiona la resta d'operacions: creació del node en /dev, càrrega del mòdul/*driver* necessari del *kernel*, etc. En sistemes amb systemd d'arrencada, el funcionament del *daemon* és controlat per systemd-udevd.

- I una col·lecció d'utilitats administratives, *udevadm*, per a diagnòstics.

Clàssicament, tots els fitxers de dispositiu del sistema s'han de trobar en el directori */dev* en una màquina Linux. Mitjançant un fitxer de dispositiu, és com un programa d'espai d'usuari, ha d'accedir a un dispositiu de maquinari o una funció d'aquest. Per exemple, el fitxer de dispositiu /dev/sda s'usa tradicionalment per a representar el primer disc del sistema. El nom *sda* correspon a un parell de nombres denominats el més gran i el més petit del dispositiu, i que són utilitzats pel *kernel* per a determinar amb quin dispositiu de maquinari interacciona. Cada nombre més gran i més petit és assignat a un nom que es

correspon amb el tipus de dispositiu. Aquesta correspondència és mantinguda per l'autoritat LANANA, i la llista d'aquestes assignacions es pot trobar a http://www.lanana.org/docs/device-list/devices-2.6+.txt.

A partir de certs desenvolupaments del *kernel* es va comprovar que aquesta situació no era pràctica, a causa de la limitació de l'esquema de nombres, i a assignar aquests nombres a situacions dinàmiques en què els dispositius són removibles, o a la gran quantitat de nombres necessaris. A més, a causa de la situació estàtica de /dev inicialment, aquest era sobrecarregat per les distribucions, per tots els possibles dispositius que ens podríem trobar al llarg del temps, creant una llarga llista d'entrades en el directori que no es farien servir.

Udev, després d'unes iteracions prèvies amb altres intents de solució, com *hotplug* i *devfs*, es va proposar com a solució suportar els punts que hem comentat anteriorment.

Nota

Es recomana consultar les pàgines man, corresponents de la distribució, per a: udev, udevd (o systemd-udevd), i udevadm.

Per a la denominació del dispositiu, se segueixen una sèrie de passos per a intentar determinar un nom únic indicant el següent: 1) etiqueta (*label*) o número de sèrie que el pugui identificar segons la classe de dispositiu; 2) número del dispositiu en el seu bus on està connectat (per exemple, en un bus PCI, el seu identificador); 3) topologia en el bus (posició en el bus a què pertany); 4) nom de reemplaçament (per si hi ha coincidència entre diversos dispositius presents); 5) nom en el *kernel*, si els passos anteriors no poden proporcionar un nom, finalment s'utilitza el disponible en el *kernel*.

El sistema udev, la informació del qual sol residir a /lib/udev i /etc/udev (també és possible que existeixi un /run/udev en temps d'execució), inclou entre altres una sèrie de regles per a la denominació i les accions que cal prendre sobre els dispositius, que podem observar en les distribucions a:

/lib/udev/rules.d

/etc/udev/rules.d

En el primer cas són les regles per defecte incloses en el sistema, mentre en el segon són les que l'administrador pugui arribar a incloure com a noves per a identificar algun dispositiu especial o canviar-ne la denominació (si hi ha regla a /etc/udev/rules.d, aquesta té preferència sobre la que ho és per defecte en el sistema a /lib/udev/rules.d). El primer nombre en el nom d'aquests fitxers de regles es refereix a l'ordre d'aplicació de les regles. I les regles poden contenir canvis de nom previstos, o bé actuacions a dur a terme segons esdeveniments d'afegit de dispositius, canvis o extracció d'aquests.

En general, el funcionament del *daemon udevd*, gestiona els esdeveniments mitjançant:

- En procés d'arrencada, es munta el directori /dev com a sistema virtual d'arxius.

- udevd copia els nodes de dispositiu estàtics que té a /lib/udev/devices en el directori /udev (depenent de la distribució).

- udevd queda a les escoltes dels esdeveniments del *kernel* (uevents), per als dispositius connectats en el sistema.

- udevd passa les dades uevents que li arriben i les intenta fer correspondre amb les regles especificades a /etc/udev/rules.d (si no, amb /lib/udev/rules.d).

- Es creen els nodes de dispositius i els enllaços segons especifiquin les regles.

- udevd llegeix les regles /etc/udev/rules.d/*.rules (o /lib/udev/...) i les emmagatzema en la memòria (en algunes distribucions incorpora fitxers binaris amb les regles precarregades, per exemple /lib/udev/hwd.d).

- udevd pot rebre notificacions si les regles canvien, s'encarrega de llegir els canvis i d'actualitzar la còpia de memòria.

Si cal carregar algun *driver*, udev també usa els *modalias* per a carregar el *driver* correcte, aquests es troben a partir de la informació dels mòduls que acaben creant (via depmod en instal·lar nous mòduls) el fitxer corresponent a /lib/modulis/`uname –r`/moduli.alies.

D'altra banda, la utilitat udevadm, entre altres coses, ens permet obtenir informació del dispositiu, amb els camps que són usats en les regles udev, de manera que podem crear noves regles que ens permetin canviar el comportament depenent dels esdeveniments del dispositiu. S'obté la informació mitjançant el paràmetre *info* i el seu devpath (el camí dins de /sys per a arribar al). Per exemple, per a la primera targeta de xarxa:

Nota

Algunes referències sobre l'escriptura de regles udev:

https://wiki.debian.org/udev

http://www.reactivated.net/writing_udev_rules.html

```
# udevadm info -a -p /sys/class/net/eth0/
looking at device '/devices/pci0000:00/0000:00:03.0/net/eth0':
    KERNEL=="eth0"
    SUBSYSTEM=="net"
    DRIVER==""
    ATTR{addr_assign_type}=="0"
    ATTR{addr_len}=="6"
    ATTR{dev_id}=="0x0"
    ATTR{ifalias}==""
    ATTR{iflink}=="2"
    ATTR{ifindex}=="2"
    ATTR{type}=="1"
    ATTR{link_mode}=="0"
```

```
    ATTR{address}=="08:00:27:37:1a:ce"
    ATTR{broadcast}=="ff:ff:ff:ff:ff:ff"
    ATTR{carrier}=="1"
    ATTR{speed}=="1000"
    ATTR{duplex}=="full"
    ATTR{dormant}=="0"
    ATTR{operstate}=="up"
    ATTR{mtu}=="1500"
    ATTR{flags}=="0x1003"
    ATTR{tx_queue_len}=="1000"
    ATTR{netdev_group}=="0"

  looking at parent device '/devices/pci0000:00/0000:00:03.0':
    KERNELS=="0000:00:03.0"
    SUBSYSTEMS=="pci"
    DRIVERS=="e1000"
    ATTRS{vendor}=="0x8086"
    ATTRS{device}=="0x100e"
    ATTRS{subsystem_vendor}=="0x8086"
    ATTRS{subsystem_device}=="0x001e"
    ATTRS{class}=="0x020000"
    ATTRS{irq}=="19"
    ATTRS{local_cpus}=="00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,
                        00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000001"
    ATTRS{local_cpulist}=="0"
    ATTRS{numa_node}=="-1"
    ATTRS{dma_mask_bits}=="32"
    ATTRS{consistent_dma_mask_bits}=="32"
    ATTRS{enable}=="1"
    ATTRS{broken_parity_status}=="0"
    ATTRS{msi_bus}=="""

  looking at parent device '/devices/pci0000:00':
    KERNELS=="pci0000:00"
    SUBSYSTEMS=="""
    DRIVERS=="""
```

En què podem observar les característiques del dispositiu, des del punt final, fins a ascendir pel bus PCI (fins a l'ID 0 del PCI, posició del bus on està connectada la targeta).

4.5. Processos

Els processos que es trobin en execució en un determinat moment seran, en general, de diferent naturalesa. Podem trobar:

- **Processos de sistema**, ja siguin processos associats al funcionament local de la màquina, el nucli, o bé processos (denominats *dimonis*) associats al control de diferents serveis. D'altra banda, poden ser locals, o de xarxa, si estem oferint el servei (actuem de servidor) o rebent els resultats del servei (actuem de clients). La majoria d'aquests processos de sistema apareixeran associats a l'usuari *root* (encara que se solen migrar a pseudousuaris especialitzats per servei). Hi pot haver alguns serveis associats a altres usuaris de sistema (*lp*, *bin*, *www*, *mail*, etc.). Aquests són pseudousuaris "virtuals", no interactius, que utilitza el sistema per a executar certs processos.

- **Processos de l'usuari administrador**: en cas d'actuar com a *root*, els nostres processos interactius o aplicacions llançades també apareixeran com a processos associats a l'usuari *root*.

- **Processos d'usuaris del sistema**: associats a l'execució de les seves aplicacions, ja siguin tasques interactives en mode text o en mode gràfic.

Com a instruccions ràpides i més útils, podem utilitzar:

- ***ps***: la instrucció estàndard, mostra els processos amb les seves dades d'usuari, temps, identificador de procés i línia d'instruccions usada. Una de les opcions més utilitzada és *ps -ef* (o *-ax*), però hi ha moltes opcions disponibles (vegeu *man ps*).

- ***top***: una versió que ens dóna una llista actualitzada a intervals, monitoritzant dinàmicament els canvis. I ens permet ordenar la llista de processos per diferents categories, com despesa de memòria, d'ús de CPU, amb propòsit d'obtenir un rànquing dels processos que acaparen els recursos. Molt útil per a donar indicis en situacions extremes de saturació d'ús de recursos, de la possible font de problemes.

- ***kill***: ens permet eliminar processos del sistema mitjançant la tramesa de senyals al procés com, per exemple, la d'acabament *kill -9 pid_del_proces* (*9* correspon a *SIGKILL*), on indiquem l'identificador del procés. Resulta útil per a processos amb comportament inestable o programes interactius que han deixat de respondre. Podem veure una llista dels senyals vàlids en el sistema amb *man 7 signal*.

4.6. Registres del sistema

Tant el nucli com molts dels dimonis de serveis, i també diferents aplicacions o subsistemes de GNU/Linux, poden generar missatges que van a parar a fitxers *log*, ja sigui per a tenir una traça del seu funcionament, o bé per a detectar errors o advertències de mal funcionament o situacions crítiques. Aquest tipus

de registre és imprescindible, en molts casos, per a les tasques d'administració, i se sol emprar bastant temps d'administració en el processament i l'anàlisi dels seus continguts.

Nota

Hi ha diversos gestors de *logs* depenent de la distribució que es faci servir, el clàssic UNIX i de GNU/Linux a l'inici és *Syslogd*, però progressivament s'han començat a usar els sistemes *rsyslog* i *Journald* (part de *systemd*) com a alternatives, o forma conjunta amb *syslogd*.

La major part dels registres es generen en el directori /var/log, encara que algunes aplicacions poden modificar aquest comportament. La majoria de registres del sistema sí que es troben en aquest directori.

Un dimoni particular del sistema (important) és el *syslogd* (més endavant comentrem les alternatives *rsyslog* i *journald*), que s'encarrega de rebre els missatges que envia el nucli i altres dimonis de serveis i els envia a un fitxer *log* que es troba en /var/log/messages. Aquest és el fitxer per defecte, però *syslogd* és també configurable (en el fitxer /etc/syslog.conf), de manera que es poden generar altres fitxers, depenent de la font, segons el dimoni que envia el missatge, i així dirigir-lo a un *log* o a un altre (classificant així per font), o també classificar els missatges per importància (nivell de prioritat): *alarm, warning, error, critical*, etc.

Nota

El dimoni *syslogd* és el servei més important d'obtenció d'informació dinàmica de la màquina. El procés d'anàlisi dels *logs* ens ajuda a entendre el funcionament, els possibles errors i el rendiment del sistema.

Depenent de la distribució (cal indicar que *Syslog* cada vegada es fa servir menys a favor de les seves alternatives *rsyslog* i *journald*), pot estar configurat de diferents maneres per defecte; en /var/log sol generar (per exemple) fitxers com kern.log, mail.err, mail.info..., que són els registres de diferents serveis. Podem examinar la configuració per a determinar d'on provenen els missatges i en quins fitxers els desa. Una opció que sol ser útil és la possibilitat d'enviar els missatges a una consola virtual de text (en /etc/syslog.conf s'especifica per als tipus de missatge una consola de destinació, com /dev/tty8 o /dev/xconsole), de manera que podrem anar veient els missatges a mesura que es produeixin. Això sol ser útil per a monitoritzar l'execució del sistema sense haver d'estar mirant els fitxers de registre a cada moment. Una modificació simple d'aquest mètode podria ser introduir, des d'un terminal, la instrucció següent (per al *log* general):

```
tail -f /var/log/messages
```

Aquesta sentència ens permet deixar el terminal o finestra de terminal, de manera que aniran apareixent els canvis que es produeixin al fitxer.

A banda dels registres de *syslog* hi ha altres ordres de sistema que ens poden ajudar en altres àmbits:

- ***uptime***: temps que fa que el sistema està actiu. Útil per a comprovar que no hi ha existit alguna rearrencada del sistema inesperat.

- ***last***: analitza el registre d'entrades/sortides del sistema (/var/log/wtmp) dels usuaris, i les arrencades del sistema. O *lastlog,* el control de l'última vegada que els usuaris han estat vistos en el sistema (informació en /var/log/lastlog).

- **Diverses utilitats per a processament combinat de *logs***, que emeten resums (o alarmes) del que ha succeït en el sistema, com per exemple *logwatch, logcheck*...

Com a nous sistemes de generació de *logs* de sistema, cal destacar rsyslogd i journald (associat al systemd), que progressivament estan substituint syslogd com a gestió dels *logs* del sistema.

En el cas de rsyslogd, el fitxer de la configuració, el trobarem a /etc/rsyslog.conf, essent la configuració per defecte suficient per a la majoria dels casos. Com pot observar-se, la majoria de la configuració és similar a la que s'ha explicat en syslogd, ja que es manté la compatibilitat amb aquest agregant una sèrie de noves prestacions. Entre aquestes, la possibilitat d'utilitzar filtres basats en context, més riquesa de filtres i la utilització de TCP com a mecanisme de comunicació, la qual cosa ens permet monitorar remotament els *logs* d'un sistema (fora del seu segment de xarxa, prèviament només era possible UDP) o disposar d'un sistema amb servidor centralitzat de *logs* de múltiples màquines client, a més de tenir suport natiu per a bases de dades MySQL i Postgres, la qual cosa permet emmagatzemar els *logs* directament en d'aquestes bases de dades (entre d'altres).

Exemple

Un exemple de servidor centralitzat de *logs* amb Rsyslog:

http://tecadmin.net/setup-centralized-logging-server-using-rsyslogd/

Quant a la posició dels *logs*, aquests se solen generar com abans preferentment a /var/log , i algunes distribucions que usen rsyslog mantenen /var/log/messages, mentre altres han canviat el fitxer a /var/log/rsyslog.

Quant a Journald, està inclòs en el systemd i, no sense crítiques (sobretot per la seva conversió dels anteriors (r)syslog basats en fitxers i *logs* text a formats binaris), està tenint un ressò important a mesura que les distribucions adopten el sistema d'arrencada de systemd.

En aquest cas, systemd ja no requereix l'execució de *daemons* de tipus syslog. Per a la lectura de *logs* només cal utilitzar:

```
# journalctl
```

En aquest cas, els *logs* del sistema es desen (depèn de la distribució) normalment a /var/log/journal, i en format binari, no llegible directament, si no són processats amb journalctl. Es disposa d'un fitxer de configuració a /etc/systemd/journald.conf (consulteu pàgina man d'aquest últim per als paràmetres configurables).

Amb journalctl, podem fer diferents consultes, algunes plantilles destacables:

 # journalctl -b Missatges de l'últim *boot*.

 # journalctl -b -p err Missatges d'error en l'últim *boot*.

 # journalctl -since=yesterday Missatges apareguts des d'ahir (combinar –since –until per a un període).

 # journalctl -f Quedar pendents dels pròxims missatges a mesura que es generin.

 # journalctl _PID=1 Fer una llista de missatges del procés amb aquest PID (en aquest cas és el mateix systemd), també _UID= o _GID=.

 # journalctl /usr/sbin/cron Missatges provinents d'un executable concret.

 # journalctl /dev/sda Missatges referents al disc.

 # journalctl --disk-usage Espai de disc utilitzat pels *logs*.

 # journalctl _TRANSPORT=kernel Missatges procedents del *kernel* (equival a -k).

 # journalctl --list-boots Llista dels últims *boots* de sistema i les seves marques de temps.

 # journalctl _SYSTEMD_UNIT=crond.service Missatges d'un servei systemd, o unitat controlada per systemd, el següent ens permet conèixer les unitats disponibles.

 # journalctl -F _SYSTEMD_UNIT Unitats disponibles per a *log* (possibles valors de _SYSTEMD_UNIT pels quals podem preguntar).

 # journalctl -F <campo> Com en el cas anterior però per al camp concret (vegeu man 7 systemd.journal-fields per conèixer els possibles camps).

Es recomana que consulteu les pàgines man: journalctl, 7 systemd.journal-fields, per una descripció exhaustiva dels camps i opcions disponibles.

4.7. Memòria

Respecte a la memòria del sistema, haurem de tenir en compte que disposem de la memòria física de la màquina mateixa i de la memòria virtual, que pot ser direccionada pels processos. Normalment (tret que estiguem tractant amb servidors empresarials), no disposarem de quantitats gaire grans, de manera que la memòria física serà menor que la mida de memòria virtual necessària (4 GB en sistemes de 32 bits, o en 64 bits un espai teòric de 16 exabytes, 2^{64} bytes, que en els sistemes físics actuals sol estar limitat a 256 terabytes). Això obligarà a utilitzar una zona d'intercanvi (*swap*) sobre disc, per a implementar els processos associats a memòria virtual, quan ultrapassen l'ús de la memòria física existent.

Aquesta zona d'intercanvi (*swap*) es pot implementar com un fitxer en el sistema d'arxius, però és més habitual trobar-la com una partició d'intercanvi (anomenada de *swap*), creada durant la instal·lació del sistema. En el moment de particionar el disc, es declara com de tipus Linux *swap*.

Per a examinar la informació sobre memòria, tenim diversos mètodes i instruccions útils:

- **Fitxer /etc/fstab**: apareix la partició *swap* (si n'hi ha). Amb una instrucció *fdisk* en podem esbrinar la mida (o consultar-la en /proc/swaps).

- **Instrucció *ps***: permet conèixer quins processos tenim, i amb les opcions de percentatge i memòria usada.

- **Instrucció *top***: és una versió de *ps* dinàmica actualitzable per períodes de temps. Pot classificar els processos segons la memòria que s'usa o el temps de CPU.

- **Instrucció *free***: informa sobre l'estat global de la memòria. Dóna també la mida de la memòria virtual.

- **Instrucció *vmstat***: informa sobre l'estat de la memòria virtual, i l'ús que s'hi dóna.

- **Alguns paquets** com *dstat* permeten recollir dades dels diferents paràmetres (memòria, *swap* i d'altres) a intervals de temps (de manera semblant a *top*).

4.8. Discos i sistemes d'arxius

Examinarem quins discos tenim disponibles, com són organitzats i de quines particions i quins sistemes d'arxius (*filesystems*) disposem. Quan disposem d'una partició i d'un determinat *filesystem* accessible, haurem de fer un procés

de muntatge per a integrar-la en el sistema, ja sigui explícitament o bé programada en arrencada. En el procés de muntatge, es connecta el sistema d'arxius associat a la partició a un punt de l'arbre de directoris.

Per a conèixer els discos (o dispositius d'emmagatzematge) que tenim en el sistema, ens podem basar en la informació d'arrencada del sistema (instrucció *dmesg* o /var/log/messages), en què es detecten els presents, com els /dev/hdx per als dispositius IDE o els SCSI amb dispositius /dev/sdx. Últimament els discos SATA i antics IDE són presentats com si fossin SCSI, a causa d'una capa d'emulació del nucli que tracta els discos com a SCSI. Altres dispositius, com discos durs connectats per USB, discos flaix (els de tipus *pen drive*), unitats extraïbles o CD-ROM externs, solen ser dispositius amb algun tipus d'emulació SCSI, per la qual cosa també es veuran com a dispositius d'aquest tipus.

Qualsevol dispositiu d'emmagatzematge presentarà una sèrie de particions del seu espai. Típicament, un disc amb el format clàssic MBR suporta un màxim de quatre particions físiques, o més si són lògiques (permeten col·locar diverses particions d'aquest tipus sobre una de física). Cada partició pot contenir tipus de *filesystems* diferents, ja sigui d'un mateix operatiu o d'operatius diferents.

El format clàssic de particions, desat en l'estructura MBR (*master boot record*), es va introduir el 1983, i és un tipus especial de sector d'arrencada (512 bytes o més en el primer sector del disc), que guarda com estan organitzades les particions del disc, a més de contenir codi que permet des del disc arrencar en sistema operatiu resident en alguna de les particions, a partir de petició del BIOS. Però MBR té algunes limitacions importants, com la de només quatre particions físiques (expandible per lògiques) i una limitació de capacitat del disc a 2 TB. A causa d'aquestes limitacions, l'esquema de partició dels discos actuals s'està transferint al nou esquema GPT (*GUID partition table*) superant aquestes limitacions. Els reemplaçaments per al BIOS de PC-Compatibles com EFI i UEFI tenen suport natiu per a GPT, a més de permetre l'accés a discos de grandàries superiors a 2 TB. Encara que cal assenyalar que durant la transició s'han realitzat diverses modificacions a l'esquema d'MBR per a permetre certa compatibilitat entre els dos esquemes. I alguns BIOS d'alguns fabricants també disposen de suport GPT.

GPT és part de l'especificació UEFI (*unified extensible firmware interface*) per a substituir els antics BIOS PC. Aquest esquema de partició de discos és usat per a l'arrencada en sistemes operatius amb suport EFI, UEFI, com per exemple US X i Windows, però en el cas del Linux i alguns BSD, poden arrencar-se de GPT, amb el suport d'EFI/UEFI o sense ell, en BIOS antics. En el cas de Linux, versions superiors a per exemple Fedora 8, Ubuntu 8, suporten ja GPT, i

proporcionen eines com gdisk, GNU Parted, fdisk, i l'arrencada de GRUB 2 amb suport GPT inclòs. En el cas de fdisk, el suport no és complet, i es recomana usar GNU parted per a un suport complet de discos amb GPT.

Tornant a la nostra anàlisi dels discos, per a conèixer-ne l'estructura o canviar la disposició de la partició del disc, podem utilitzar l'ordre fdisk (en discos MBR, en GPT el suport és parcial i es recomana GNU gparted) o qualsevol de les variants d'fdisk, més o menys interactives (com cfdisk, sfdisk). Per exemple, en examinar un disc (antic) de de tipus IDE /dev/hda, ens dóna la informació següent:

```
# fdisk -l /dev/hda
Disk /dev/hda: 20.5 GB, 20520493056 bytes 255 heads, 63 sectors/track, 2494 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Device Boot  Start  End   Blocks Id System
/dev/hda1    *      1     1305   10482381   7 HPFS/NTFS
/dev/hda2    *      1306  2429   9028530    83 Linux
/dev/hda3           2430  2494   522112+    82 Linux swap
```

Disc MBR de 20 GB amb tres particions (s'identifiquen amb el número afegit al nom del dispositiu), en què observem dues particions amb arrencada (columna *Boot* amb *) de tipus NTFS i Linux, fet que representa l'existència d'un Windows NT/2000/XP/Vista/7 juntament amb una distribució GNU/Linux, i l'última partició, que és usada de *swap* per a Linux. A més, tenim informació de l'estructura del disc i de les mides de cada partició. També es pot obtenir informació de totes les particions presents en el sistema consultat el fitxer /proc/partitions.

En aquest altre exemple observem un disc SATA, /dev/sda en un altre sistema:

```
$ fdisk -l /dev/sda

Disc /dev/sda: 500.1 GB, 500107862016 bytes
255 caps, 63 sectors/pista, 60801 cilindres, 976773168 sectors en total
Unitats = sectors d'1 * 512 = 512 bytes
Grandària de sector (lògic / físic): 512 bytes / 512 bytes
Grandària I/S (mínim/òptim): 512 bytes / 512 bytes
Identificador del disc: 0x000aba55

Dispositiu Inici    Començament      Fi      Blocs  Id  Sistema
/dev/sda1   *        2048   960016383   480007168   83  Linux
/dev/sda2       960018430   976771071     8376321    5  Extendida
/dev/sda5       960018432   976771071     8376320   82  Linux swap
```

En aquest disc de 500 GB podem observar tres particions (sda1, sda2 i sda5) de les quals una és lògica estesa (sda2) que en realitat conté l'sda5. Per tant tenim dues particions finals, l'sda1 de tipus *boot* que ens arrenca Linux, i l'sda5

que és utilitzada per swap. L'estesa (en aquest cas, sda2) ens permetrà en el cas MBR crear particions addicionals (si disposem d'espai en el disc), com per exemple, sda6, sda7 i sda8, dins de l'estesa sda2, superant així el límit de quatre particions físiques amb què ens trobem en discos MBR.

En un sistema amb discos amb particions GPT:

```
# fdisk -l /dev/sdb

WARNING: GPT (GUID Partition Table) detected on '/dev/sdb'! The util fdisk doesn't support GPT.
                   Use GNU Parted.

Disc /dev/sdb: 5999.5 GB, 5999532441600 bytes
255 heads, 63 sectors/track, 729401 cylinders
Units = cilindres of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

Disposit. Inici    Començament      Fi      Blocs  Id  Sistema
/dev/sdb1               1      267350  2147483647+  ee  GPT

# parted
GNU Parted 2.1
Usant /dev/sda
Welcome to GNU Parted! Type 'help' to view a list of commands.

(parted) select /dev/sdb
Usando /dev/sdb

(parted) print
Model: DELL PERC 6/i (scsi)
Disk /dev/sdb: 6000GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt

Número  Inici  Fi     Grandària  Sistema de fitxers  Nom  Banderes
 1      1049kB  6000GB  6000GB  ext4
```

En aquest sistema, es disposa d'un *storage*, que es veu com un dispositiu /dev/sdb amb un total de 6 terabytes disponibles, en una partició /dev/sdb1 que és de tipus GPT, fdisk ens avisa que no disposa de suport de GPT i que usem alternativament GNU parted. En la segona part es crida parted, com a eina de gestió de GPT. Aquesta eina és interactiva (ordre *help*, opcions disponibles), seleccionem el dispositiu que ens interessa (*select*) i visualitzem les particions (*print*).

Dels discos i particions de què disposem, alguns es trobaran muntats en el nostre sistema de fitxers, o estaran preparats per a muntar-los sota demanda o bé muntar-los en el moment en què en disposem (en el cas de dispositius extraïbles).

Aquesta informació la podem obtenir de diferents maneres (ho veurem amb més detall en el taller final):

- **Fitxer /etc/fstab**: indica dispositius que estan preparats per a muntar-se en l'arrencada o els extraïbles que podran ser muntats. No han d'estar necessàriament tots els del sistema, sinó només aquells que vulguem tenir en arrencada. Els altres els podem muntar sota demanda amb la instrucció *mount*, o desmuntar-los amb *umount*.

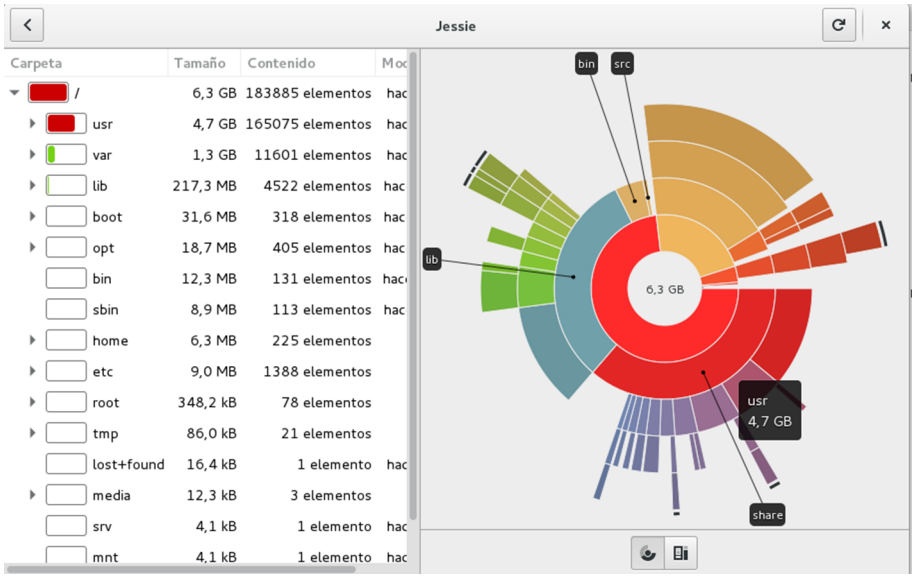
- **Instrucció *mount***: ens informa dels *filesystems* muntats en aquell moment (ja sigui dispositius reals o *filesystems* virtuals, com /proc). Podem obtenir aquesta informació també des del fitxer /etc/mtab.

- **Instrucció *df -k***: ens informa dels *filesystems* d'emmagatzemament, i ens permet verificar l'espai usat i disponible. Es tracta d'una instrucció bàsica per a controlar l'espai de disc disponible. Respecte a aquesta instrucció *df -k*, una de les nostres tasques bàsiques d'administració de la màquina és controlar-ne els recursos, i en aquest cas l'espai disponible en els *filesystems* utilitzats. Aquestes mides cal monitoritzar-les amb certa freqüència per a evitar la caiguda del sistema; mai no s'hauria de deixar un *filesystem* (sobretot si és el /) per sota d'un 10-15%, ja que hi ha molts processos (dimonis de serveis) que escriuen informació temporal o registres, que poden consumir molt espai. Un cas particular el formen els fitxers *core,* generats per errors de programari i que contenen informació de depuració dels errors que els han provocat juntament amb la imatge del procés, els quals poden tenir (depenent del procés) mides molt grans d'arxiu. Normalment, caldrà tenir algunes precaucions de "neteja del sistema" si es detecten situacions de saturació del *filesystem*:

- **Eliminar temporals antics**. Els directoris /tmp i /var/tmp solen acumular molts arxius generats per diferents usuaris o aplicacions. Alguns sistemes o distribucions ja prenen mesures de neteja, com netejar /tmp en cada arrencada del sistema.

- **Registres**: cal evitar-ne el creixement excessiu, ja que segons la configuració del sistema (per exemple, de *syslogd*) la informació generada de missatges pot ser molt gran. Caldrà netejar periòdicament en arribar a determinades mides, i en tot cas, si necessitem la informació per a anàlisis posteriors, podem fer còpies de seguretat en mitjans extraïbles. Aquest procés es pot automatitzar mitjançant ús de *scripts cron*, o bé per mitjà d'eines especialitzades com *logrotate*.

- Hi ha altres punts del sistema que solen créixer molt, com poden ser:
  - Fitxers *core* dels usuaris: els podem eliminar periòdicament o fer que no es generin.
  - El sistema de correu electrònic: emmagatzema tots els correus enviats i rebuts; podem demanar als usuaris que facin neteja periòdica, o bé posar sistemes de quotes.
  - Les memòries cau dels navegadors o altres aplicacions: també solen tenir mides grans, una altra neteja que caldrà fer periòdicament.
  - Els comptes dels usuaris mateixos: poden tenir quotes per a no superar les mides prefixades...
- A banda de comandes (com `du`) per a l'anàlisi d'espai dels sistemes de fitxers, també existeixen diferents eines per a l'anàlisi gràfica de l'ocupació dels fitxers en els sistemes d'arxius. Aquestes eines permeten una visió global de l'administrador, en integrar informació tant local com remota (integrant en la visualització sistemes d'arxius per NFS, per exemple). Algunes eines que destaquen per a aquesta funcionalitat: *kdirstat* (o la seva substituta *k4dirstat*), *baobab*, *gdmap*, entre altres.



Anàlisi del *filesystem* arrel (/) en una distribució Debian amb baobab

4.9. Upower, Udisks

Com a casos especials de control de dispositius, semblant a udev, però no integrat en aquest, hi ha els casos de gestió de dispositius relacionats amb l'energia, i els discos.

Aquests dos components estan integrats en la iniciativa freedesktop.org, que entre els seus objectius inclou el desenvolupament de tecnologies interoperables per a compartir bases per a desenvolupar entorns d'escriptori per a X win-

dow per a entorns de GNU/Linux i altres operatius de tipus UNIX. La idea és que hi ha molts entorns de treball (*frameworks*) de desenvolupament per a X, però l'objectiu és intentar que les diferències de desenvolupament no siguin visibles a l'usuari, mentre es puguin utilitzar una sèrie de tecnologies base comunes.

freedesktop.org

Grup freedesktop.org, els seus projectes i especificacions:

http://www.freedesktop.org/wiki/Software/.

http://www.freedesktop.org/wiki/Specifications/.

asi com el projecte DeviceKit: http://www.freedesktop.org/wiki/Software/DeviceKit.

Diversos projectes d'entorns d'escriptori, com GNOME, KDE i Xfce, estan col·laborant amb freedesktop.org (que també és coneguda com a X Desktop Group o XDG).

També un dels projectes importants que mantenen per a GNU/Linux és D-bus, que és un mecanisme de comunicació interprocés (IPC), que permet a processos que s'executen concurrentment comunicar-se entre ells per a intercanviar informació o demanar-se serveis. En el GNU-Linux, s'utilitza per a comunicar aplicacions d'escriptori en execució en la mateixa sessió d'escriptori i per a comunicar la sessió d'escriptori amb el sistema operatiu, tant si és el *kernel* com *daemons* o processos propis. El D-Bus s'utilitza en aplicacions d'escriptori de KDE, Gnome i Xfce. El D-Bus és bàsicament un sistema de bus per a intercanviar missatges, de manera que dues aplicacions o dos processos poden intercanviar missatges un amb l'altre mitjançant el *daemon* de pas de missatges. En systemd, es va reescriure el codi de D-Bus i es va fer més eficient, augmentant les prestacions. Paral·lelament, també hi ha un projecte semblant, denominat kdbus (*Linux Kernel D-Bus implementation*), que oferirà una alternativa en el futur, mitjançant comunicacions d'igual a igual (peer-to-peer) per a implementar IPC amb mediació del *kernel*.

L'arquitectura D-Bus consta principalment de tres capes:

Nota:

D-Bus tutorial: http://dbus.freedesktop.org/doc/dbus-tutorial.html.

Howto use dbus-monitor: https://wiki.ubuntu.com/DebuggingDBu.

- **libdbus**. Biblioteca que permet a dues aplicacions connectar-se l'una amb l'altra i intercanviar missatges.

- **dbus-daemon**. *Daemon* executable que implementa el bus de missatges i que, basant-se en libdbus, permet que múltiples aplicacions es connectin, i s'encarrega de rootejar els missatges d'una aplicació a zero o més aplicacions destinatàries mitjançant mecanismes publish/subscribe.

- **Biblioteques construïdes per a entorns de treball d'aplicacions específics**. Des del punt de vista d'administrador, desenvolupador d'aplicacions

és útil l'eina dbus-monitor, que permet veure els missatges que circulen pel bus de missatges tant en el cas de sistema com de sessió d'escriptori.

En un dels projectes mantinguts per freedesktop.org, *DeviceKit*, entre altres es van desenvolupar Upower i Udisks com a interfícies i serveis, mitjançant D-Bus, per a manejar la gestió d'energia i els dispositius d'emmagatzematge.

Upower, ens permet enumerar dispositius d'energia (AC o bateries), escoltant pels esdeveniments del dispositiu, i preguntar pel seu històric o estadístiques. Alguns exemples:

- **# upower -e** Enumerar dispositius de bateria o AC
- **# upower -d** Informació de bateria.
- **# upower -i <nom_dispositiu>** Obtenir informació especifica d'un dispositiu (obtingut de –e).

Per exemple:

```
# upower -e
/org/freedesktop/UPower/devices/line_power_AC

# upower -d
Device: /org/freedesktop/UPower/devices/line_power_AC
  native-path:          AC
  power supply:         yes
  updated:              dom 20 jul 2014 17:57:59 CEST (23889 seconds ago)
  has history:          no
  has statistics:       no
  line-power
    online:             yes

Daemon:
  daemon-version:  0.9.23
  on-battery:      no
  on-low-battery:  no
  lid-is-closed:   no
  lid-is-present:  no
  is-docked:       no

# upower -i /org/freedesktop/UPower/devices/line_power_AC
  native-path:          AC
  power supply:         yes
  updated:              dom 20 jul 2014 17:57:59 CEST (23905 seconds ago)
  has history:          no
  has statistics:       no
```

```
    line-power
      online:             yes
```

En aquest cas en trobem una línia d'AC, el seu estat, i des de quan està activa; en casos de bateries, és possible fer-ne un seguiment de la càrrega o descàrrega, a partir dels esdeveniments d'energia que es generen, amb:

```
  # upower --monitor-detail
```

o directament *-i <nom_bateria>* per a conèixer les estadístiques generals.

D'altra banda, udisks, s'implementa un *daemon*, udisksd, que implementa interfícies D-Bus, que es poden fer servir per a consultar i manipular els dispositius d'emmagatzematge. I amb la utilitat de línia de tecles d'ordre, udisksctl o udisks (depenent de la distribució, o fins i tot pot disposar de tots dos), es pot utilitzar per a preguntar i es pot utilitzar el *daemon* per a consultar els discos, per exemple amb udiskctl, que és l'eina de línia de tecles d'ordre per a interactuar amb el procés de *daemon* d'udisksd:

Nota

Vegeu el manual de referència Udisks a: https://udisks.freedesktop.org/docs/latest/.

Si l'ordre `udisks` no està disponible pot utilitzar-se `udiskctl`: https://udisks.freedesktop.org/docs/latest/udisksctl.1.html.

- **# udisks -dump** Informació dels dispositius presents.

- **# udisksctl status** Mostra informació d'alt nivell dels discos i els seus dispositius de blocs.

- **# udisksctl mount** Munta un dispositiu que li passem, en /media (desmuntar amb umount)

- **# udisksctl monitor** Visualitza els esdeveniments que es produeixen en el *daemon*.

D'altra banda, udisks com a utilitat de línia de tecles d'ordre si està present (o els seus equivalents a *udiskctl*):

- **# udisks --enumerate** Llista ids dels dispositius presents (com a alternativa, *udiskctl status*, però només obté informació de primer nivell, no particions per exemple; en aquest cas *udiskctl dump* obté informació més detallada).

- **# udisks --enumerate-device-files** Llista els dispositius i els seus identificadors coneguts (com els UUID entre d'altres; com a alternativa, *udiskctl dump*).

- **# udisks --dump** Informació extensa de tots els dispositius presents (com a alternativa *udiskctl dump*).

- **# udisks --show-info** Informació d'un dispositiu particular (com a alternativa *udiskctl -b dispositiu info*).

- **# udisks --monitor-detail** Monitorar amb detall l'activitat del *daemon* de discos (udisksd) (com a alternativa *udiskctl monitor*).

Un exemple d'una sessió, en què s'analitza /dev/sda i /dev/sda1, d'una determinada màquina, amb l'enumeració prèvia de dispositius, a més en el cas del disc se n'obté la informació SMART, que ens pot ser útil per a conèixer la "salut" del seu estat físic actual:

```
# udisks --enumerate
/org/freedesktop/UDisks/devices/sda5
/org/freedesktop/UDisks/devices/sr0
/org/freedesktop/UDisks/devices/sda
/org/freedesktop/UDisks/devices/sda1
/org/freedesktop/UDisks/devices/sda2

# udisks --enumerate-device-files
/dev/sda5
/dev/disk/by-id/ata-MB0500EBNCR_WMAP146173-part5
/dev/disk/by-id/scsi-SATA_MB0500EBNCR_WMAP146173-part5
/dev/disk/by-id/wwn-0x50014ee002c9780b-part5
/dev/disk/by-uuid/10628c07-51f6-4bb6-babe-082526be81d7
/dev/disk/by-path/pci-0000:00:1f.2-scsi-0:0:0:0-part5
/dev/sr0
/dev/disk/by-id/ata-DV-18S-A_11022334083404
/dev/disk/by-path/pci-0000:00:1f.5-scsi-0:0:0:0
/dev/sda
/dev/disk/by-id/ata-MB0500EBNCR_WMAP146173
/dev/disk/by-id/scsi-SATA_MB0500EBNCR_WMAP146173
/dev/disk/by-id/wwn-0x50014ee002c9780b
/dev/disk/by-path/pci-0000:00:1f.2-scsi-0:0:0:0
/dev/sda1
/dev/disk/by-id/ata-MB0500EBNCR_WMAP146173-part1
/dev/disk/by-id/scsi-SATA_MB0500EBNCR_WMAP146173-part1
/dev/disk/by-id/wwn-0x50014ee002c9780b-part1
/dev/disk/by-uuid/b16ebe2f-5072-4e00-a1e0-19aebf2be5e8
/dev/disk/by-path/pci-0000:00:1f.2-scsi-0:0:0:0-part1
/dev/sda2
/dev/disk/by-id/ata-MB0500EBNCR_WMAP146173-part2
/dev/disk/by-id/scsi-SATA_MB0500EBNCR_WMAP146173-part2
/dev/disk/by-id/wwn-0x50014ee002c9780b-part2
/dev/disk/by-path/pci-0000:00:1f.2-scsi-0:0:0:0-part2

# udisks --show-info /dev/sda
Showing information for /org/freedesktop/UDisks/devices/sda
```

```
  native-path:            /sys/devices/pci0000:00/0000:00:1f.2/host0/target0:0:0/0:0:0:0/block/sda
  device:                      8:0
  device-file:                 /dev/sda
    presentation:              /dev/sda
    by-id:                     /dev/disk/by-id/ata-MB0500EBNCR_WMAP14612773
    by-id:                     /dev/disk/by-id/scsi-SATA_MB0500EBNCR_WMAP1461773
    by-id:                     /dev/disk/by-id/wwn-0x50014ee002c9780b
    by-path:                   /dev/disk/by-path/pci-0000:00:1f.2-scsi-0:0:0:0
  detected at:                 lun 21 jul 2014 00:51:05 CEST
  system internal:             1
  removable:                   0
  has media:                   1 (detected at lun 21 jul 2014 00:51:05 CEST)
    detects change:            0
    detection by polling:      0
    detection inhibitable:     0
    detection inhibited:       0
  is read only:                0
  is mounted:                  0
  mount paths:
  mounted by uid:              0
  presentation hide:           0
  presentation nopolicy:       0
  presentation name:
  presentation icon:
  automount hint:
  size:                        500107862016
  block size:                  512
  job underway:                no
  usage:
  type:
  version:
  uuid:
  label:
  partition table:
    scheme:                    mbr
    count:                     3
  drive:
    vendor:                    ATA
    model:                     MB0500EBNCR
    revision:                  HPG0
    serial:                    WMAP146173
    WWN:                       50014ee002c9780b
    detachable:                0
    can spindown:              1
    rotational media:          Yes, at 7200 RPM
    write-cache:               disabled
    ejectable:                 0
```

```
    adapter:                    /org/freedesktop/UDisks/adapters/0000_3a00_3a1f_2e2
    ports:
      /org/freedesktop/UDisks/adapters/0000_3a00_3a1f_2e2/host0
    similar devices:
    media:
      compat:
    interface:                  ata
    if speed:                   (unknown)
    ATA SMART:                  Updated at lun 21 jul 2014 01:21:05 CEST
      overall assessment:       Good
================================================================================
 Attribute       Current|Worst|Threshold  Status   Value       Type     Updates
================================================================================
 raw-read-error-rate         200|200| 51   good    0           Pre-fail Online
 spin-up-time                100|253| 21   good    0           Pre-fail Online
 start-stop-count            100|100|  0    n/a    6           Old-age  Online
 reallocated-sector-count    200|200|140   good    0 sectors   Pre-fail Online
 seek-error-rate             200|200| 51   good    0           Pre-fail Online
 power-on-hours               97| 95|  0    n/a    120,7 days  Old-age  Online
 spin-retry-count            100|253| 51   good    0           Pre-fail Online
 calibration-retry-count     100|253| 51   good    0           Pre-fail Online
 power-cycle-count           100|100|  0    n/a    5           Old-age  Online
 unused-reserved-blocks      200|200|100   good    0           Pre-fail Online
 end-to-end-error            100|100| 97   good    0           Pre-fail Online
 reported-uncorrect          100|100|  0    n/a    0 sectors   Old-age  Online
 command-timeout             100|100|  0    n/a    0           Old-age  Online
 airflow-temperature-celsius  70| 61| 45   good    30C / 86F   Old-age  Online
 power-off-retract-count     200|200|  0    n/a    4           Old-age  Online
 load-cycle-count            200|200|  0    n/a    1           Old-age  Online
 temperature-celsius-2       113|104|  0    n/a    30C / 86F   Old-age  Online
 hardware-ecc-recovered      200|200|  0    n/a    0           Old-age  Online
 reallocated-event-count     200|200|  0    n/a    0           Old-age  Online
 current-pending-sector      200|200|  0    n/a    0 sectors   Old-age  Online
 offline-uncorrectable       100|253|  0    n/a    0 sectors   Old-age  Offline
 udma-crc-error-count        200|200|  0    n/a    0           Old-age  Online
 multi-zone-error-rate       100|253|  0    n/a    0           Old-age  Offline

# udisks --show-info /dev/sda1
Showing information for /org/freedesktop/UDisks/devices/sda1
  native-path:                /sys/devices/pci0000:00/0000:00:1f.2/host0/target0:0:0/0:0:0:0/block
                                     /sda/sda1
  device:                       8:1
  device-file:                  /dev/sda1
    presentation:               /dev/sda1
    by-id:                      /dev/disk/by-id/ata-MB0500EBNCR_WMAP146173-part1
    by-id:                      /dev/disk/by-id/scsi-SATA_MB0500EBNCR_WMAP146173-part1
    by-id:                      /dev/disk/by-id/wwn-0x50014ee002c9780b-part1
```

```
  by-id:                      /dev/disk/by-uuid/b16ebe2f-5072-4e00-a1e0-19aebf2be5e8
  by-path:                    /dev/disk/by-path/pci-0000:00:1f.2-scsi-0:0:0:0-part1
detected at:                  lun 21 jul 2014 00:51:05 CEST
system internal:              1
removable:                    0
has media:                    1 (detected at lun 21 jul 2014 00:51:05 CEST)
  detects change:             0
  detection by polling:       0
  detection inhibitable:      0
  detection inhibited:        0
is read only:                 0
is mounted:                   1
mount paths:              /
mounted by uid:               0
presentation hide:            0
presentation nopolicy:        0
presentation name:
presentation icon:
automount hint:
size:                         491527340032
block size:                   512
job underway:                 no
usage:                        filesystem
type:                         ext4
version:                      1.0
uuid:                         b16ebe2f-5072-4e00-a1e0-19aebf2be5e8
label:
partition:
  part of:                    /org/freedesktop/UDisks/devices/sda
  scheme:                     mbr
  number:                     1
  type:                       0x83
  flags:                      boot
  offset:                     1048576
  alignment offset:           0
  size:                       491527340032
```

5. Sistema de fitxers

A cada màquina amb un sistema GNU/Linux podem veure sistemes de fitxers de diferents tipus [Hin] [Soy12]. Per començar, és habitual trobar-se amb els sistemes de fitxers propis de Linux creats en diferents particions dels discos [Koe].

La configuració habitual sol ser de dues particions:

1) la corresponent a "/" (*root filesystem*) i

2) la corresponent al fitxer d'intercanvi o de *swap*.

Amb tot, en configuracions més professionals sol ser habitual separar particions amb parts "diferenciades" del sistema. Una tècnica habitual és, per exemple (en veurem més opcions després), crear particions diferents per al següent:

```
/    /boot /home      /opt  /tmp  /usr  /var  swap
```

que segurament es trobaran muntades des de diferents orígens (diferents discos, o fins i tot xarxa en alguns casos).

Les particions es fan per a separar clarament parts estàtiques i dinàmiques del sistema, per a permetre d'una manera més fàcil, davant de problemes de saturació, estendre les particions o aïllar més fàcilment parts per a fer còpies de seguretat (per exemple, els comptes dels usuaris en la partició /home).

El tipus de particions *swap* és de tipus *Linux swap*, i la corresponent a / sol ser d'algun dels sistemes de fitxers estàndard, ja sigui *ext2* (el tipus per defecte fins als nuclis 2.4), *ext3* o el nou *ext4*, que són millores de l'*ext2* compatibles però amb *journaling*, la qual cosa permet tenir un registre del que va passant al sistema de fitxers, per a recuperacions més ràpides en cas d'error. També poden ser habituals altres sistemes d'arxius, com Reiser (endesús per manca de suport), XFS i d'altres més utilitzats amb un suport important com Btrfs i ZFS.

Una altra configuració habitual pot ser de tres particions: /, *swap*, /home, en què /home es dedicarà als comptes dels usuaris. Això permet separar els comptes dels usuaris del sistema, aïllant en dues particions separades, i podem donar l'espai necessari per als comptes en una altra partició.

Un altre esquema molt utilitzat és el de separar en particions les parts estàtiques del sistema de les dinàmiques; per exemple, una partició per a / amb la part estàtica (/bin /sbin i /usr en alguns casos) que s'espera que no creixerà o ho farà molt poc, i una altra o diverses amb la part dinàmica (/var /tmp /opt), suposant que /opt, per exemple, és el punt d'instal·lació del programari nou. Això permet ajustar millor l'espai de disc i deixar més espai per a les parts del sistema que en necessitin.

Respecte als sistemes de fitxers suportats, n'hem de destacar la gran varietat; actualment, podem trobar (entre d'altres):

- **Sistemes associats a GNU/Linux**, com l'estàndard *ext2*, *ext3*, evolució de l'anterior amb concepte de *journaling* (suport de registre d'operacions fetes en el sistema de fitxers que en pot permetre la recuperació en cas d'algun desastre que ho faci inconsistent). En les noves distribucions tenen un pes important ext4 com a evolució d'ext3 (amb millores en les prestacions), i btrfs, un nou disseny (que aporta fitxers més grans i prestacions addicionals amb tolerància a fallades i reparació) que s'espera que sigui el sistema de fitxers per defecte en la majoria d'aquestes.

- **Compatibilitat amb entorns no GNU/Linux**: *msdos*, *vfat*, *ntfs*, accés als diferents sistemes de *fat16*, *fat32* i *ntfs*. En particular, es pot destacar que, en distribucions antigues, el suport del nucli, en el cas de l'ntfs, estava limitat a lectura. Però com ja hem dit, hi ha solucions en l'espai d'usuari (mitjançant FUSE, un component que permet gestionar sistemes de fitxers en espai d'usuari), que permeten l'escriptura, com l'*ntfs-3g* ja esmentat. També es disposa de compatibilitat en altres entorns com Mac amb *hfs* i *hfsplus*.

- **Sistemes associats a suports físics**, com el cas de CD/DVD, amb els *iso9660* i *udf*.

- **Sistemes usats en diferents UNIX**, que ofereixen generalment un rendiment millor (de vegades, a costa de més consum de recursos, en CPU per exemple), com JFS2 (IBM), XFS (SGI), o ReiserFS, Zfs (Oracle Solaris).

- **Sistemes de fitxers en xarxa** (més tradicionals): NFS, Samba (*smbfs*, *cifs*), permeten accedir a sistemes de fitxers disponibles en altres màquines de manera transparent per xarxa.

- **Sistemes distribuïts en xarxa**: com GFS, Lustre, Ceph, HDFS o Coda.

- **Pseudosistemes de fitxers**, com procfs (/proc) o sysfs (/sys).

En la majoria (excepte algun cas especial) d'aquests sistemes de fitxers, GNU/Linux ens permetrà crear particions d'aquests tipus, construir el sistema de fitxers del tipus requerit i muntar-les com a part integrant de l'arbre de directoris, ja sigui de manera temporal o permanent.

5.1. Punts de muntatge

A part del *filesystem* principal / i de les seves possibles divisions en particions extres (/usr, /var, /tmp, /home), es pot tenir en compte la possibilitat de deixar punts de muntatge preparats per al muntatge d'altres sistemes de fitxers, ja siguin particions de disc o altres dispositius d'emmagatzematge.

A les màquines en les quals GNU/Linux comparteix la partició amb altres sistemes operatius, mitjançant algun sistema d'arrencada (lilo, grub o grub 2), hi pot haver diverses particions assignades als diferents operatius. Moltes vegades és interessant compartir dades amb aquests sistemes, ja sigui per a llegir els seus fitxers o modificar-los. A diferència d'altres sistemes (que només tenen en compte les seves pròpies dades i els seus sistemes de fitxers, i en els quals en algunes versions no se suporten alguns dels seus sistemes de fitxers propis), GNU/Linux és capaç de tractar, com hem vist, amb una quantitat enorme de sistemes de fitxers de diferents operatius i poder compartir la informació.

Exemple

Si en els PC personals hem instal·lat GNU/Linux, segurament trobarem més d'un operatiu; per exemple, una altra versió de GNU/Linux amb *ext2* o *ext3* de sistema de fitxers; podríem trobar un antic MSDOS amb el seu sistema de fitxers FAT, un Windows98/ME/XP Home amb FAT32 (o *vfat* per a Linux) o un Windows NT/2000/XP/Vista/7/8.x amb sistemes NTFS (*ntfs* per a Linux) i FAT32 (*vfat*) alhora.

El nostre sistema GNU/Linux pot llegir dades (disponibles com a fitxers i directoris) de tots aquests sistemes de fitxers i escriure en la majoria.

En el cas d'NTFS, fins a certs moments hi va haver problemes en l'escriptura, que estava en forma experimental en la majoria de controladors del nucli apareguts, a causa, principalment, de les diferents versions que van apareixent del sistema de fitxers, ja que hi ha dues versions principals anomenades *NTFS* i *NTFS2*, i algunes extensions com els anomenats *volums dinàmics*, o els sistemes de fitxers xifrats. Accedir amb segons quina opció de controladors presentava certes incompatibilitats, que podrien causar corrupcions de dades o errors en el sistema de fitxers.

A causa de FUSE, un mòdul integrat en el nucli (a partir del 2.6.11), s'ha permès un desenvolupament més flexible de sistemes de fitxers, directament en l'espai d'usuari (de fet, FUSE actua com un "pont" entre les peticions del nucli i l'accés que es fa des del controlador).

Gràcies a les possibilitats de FUSE, es té un suport més o menys complet d'NTFS (mentre Microsoft no faci més canvis en l'especificació), en especial des de l'aparició del controlador (basat en FUSE) *ntfs-3g* i la combinació amb les utilitats *ntfsprogs*.

Perquè es puguin llegir o escriure les dades, la partició ha d'estar disponible dins del nostre sistema de fitxers arrel (/). Per tant, cal dur a terme un procés de "muntatge" del sistema de fitxers en algun punt del nostre arbre de directoris.

Depenent de la distribució, s'usen uns sistemes o d'altres, o també els podem crear nosaltres. Normalment, solen existir o bé com a subdirectoris de l'arrel, per exemple /cdrom, /win, /floppy, o bé com a subdirectoris dins de /mnt, el punt estàndard de muntatge (apareixen com a /mnt/cdrom, /mnt/floppy...), o el directori /media, que és el preferit últimament per les distribucions. Segons l'estàndard FHS /mnt s'hauria d'usar per a muntatges temporals de sistemes d'arxiu, mentre que /media s'utilitzaria per a muntar dispositius extraïbles.

El procés de muntatge es fa mitjançant l'ordre *mount* amb el format següent:

```
mount -t filesystem-type device mount-point
```

El tipus de *filesystem* pot ser: *msdos* (FAT), *vfat* (FAT32), *ntfs* (NTFS de lectura), *iso9660* (per a CDROM), *ext2*, *ext3*, *xfs*... (dels disponibles).

El dispositiu (*device*) és l'entrada corresponent en el directori /dev a la localització del dispositiu; els IDE tenien /dev/hdxy, en què *x* és *a*, *b*, *c* o *d* (1 mestre, 1 esclau, 2 mestre, 2 esclau) i *y*, el número de partició; en els SCSI (/dev/sdx), *x* és *a, b, c, d*... (segons l'ID SCSI associat 0, 1, 2, 3, 4...).

En veurem alguns casos:

```
mount -t iso9660 /dev/sdc /mnt/cdrom
```

muntaria el CD-ROM (si és el dispositiu sdc, per exemple si es disposa d'altres dos discos prèviament) en el punt /mnt/cdrom.

```
mount -t iso9660 /dev/cdrom /mnt/cdrom
```

muntaria el CD-ROM; /dev/cdrom s'usa com a sinònim (és un enllaç) del dispositiu on està connectat.

```
mount -t vfat /dev/fd0H1440 /mnt/floppy
```

muntaria el disquet, /dev/fd0H1440. Seria la disquetera A en alta densitat (1,44 MB). També es pot usar /dev/fd0. Els disquets pràcticament han desaparegut dels PC actuals, però encara en podem trobar en PC antics o en alguns servidors antics.

```
mount -t ntfs /dev/sda2 /mnt/win8
```

muntaria la segona partició del primer dispositiu (la C:), de tipus NTFS (per exemple, un Windows 8.x, en una màquina que tingui arrencada dual).

Si aquestes particions són més o menys estables en el sistema (és a dir, no canvien freqüentment) i les volem utilitzar, el millor serà incloure els muntatges perquè es facin en temps d'execució, en iniciar el sistema, mitjançant la configuració del fitxer /etc/fstab:

```
# /etc/fstab: Informació estàtica del sistema de fitxers
#
#<Sis. fitxers><Punt muntatge><Tipus><Opcions> <Bolcat>  <Passada>
/dev/sda2            /                    ext3       errors = remountro 0           1
/dev/sdb3            none                 swap       sw                 0           0
proc                 /proc                proc       defaults           0           0
/dev/fd0             /floppy              auto       user,noauto        0           0
/dev/cdrom           /cdrom               iso9660    ro,user,noauto     0           0
/dev/sdb1            /mnt/usb             vfat       user,noauto        0           0
```

Per exemple, aquesta configuració inclou alguns dels sistemes estàndard, com l'arrel en /dev/sda2, la partició de *swap* que està en *sdb3*, el sistema *proc* (que utilitza el nucli per a desar la seva informació). I el disquet, el CD-ROM, i en aquest cas un disc USB de tipus flaix (que es detecta com un dispositiu SCSI). En alguns casos, s'especifica *auto* com a tipus de *filesystem*. Això permet que s'autodetecti el sistema de fitxers. Si es coneix, és millor indicar-ho en la configuració i, d'altra banda, el *noauto* en les opcions permet que no sigui muntat de manera automàtica sempre, sinó sota petició (o accés al directori).

Nota

El format de */etc/fstab*, especialment en la primera columna (*device*), depèn del tipus de dispositius que trobem, tant si són particions físiques com especials de tipus RAID programari (*/dev/mdx*) o LVM (de tipus */dev/mapper*), o UUID, que són identificadors únics assignats a discos o particions, utilitzats en temps d'arrencada. Podem usar l'ordre *blkid* o accedir a */dev/disk/by-uuid* per a conèixer aquests últims UUID associats.

Si tenim aquesta informació en el fitxer, el procés de muntatge se simplifica molt, ja que es farà o bé en execució, en arrencada, o bé sota demanda (per als *noauto*). I es pot fer ara simplement demanant que es munti el punt de muntatge o el dispositiu:

```
mount /mnt/cdrom
mount /dev/fd0
```

ja que el sistema ja té la resta de la informació. El procés contrari, el desmuntatge, és bastant senzill, amb la instrucció *umount* amb el punt o dispositiu:

```
umount /mnt/cdrom
umount /dev/fd0
```

En el cas de mitjans extraïbles, de tipus CD-ROM (o d'altres), es pot usar *eject* per a l'extracció del suport físic:

```
eject /dev/cdrom
```

o, en aquest cas, només:

```
eject
```

Les instruccions *mount* i *umount* munten o desmunten tots els sistemes disponibles. En el fitxer /etc/mtab es manté una llista dels sistemes muntats. En un moment concret es pot consultar o executar *mount* sense paràmetres per a obtenir aquesta informació.

5.2. Permisos

Un altre assumpte que caldrà controlar, en el cas dels fitxers i directoris, és el dels permisos que volem establir en cada un; cal recordar que cada fitxer pot disposar de la sèrie de permisos *rwxrwxrwx*, que són: *rwx* del propietari, *rwx* del grup a què l'usuari pertany i *rwx* per a altres usuaris. En cada un es pot establir el permís de lectura (*r*), escriptura (*w*) o execució (*x*). En el cas d'un directori, *x* denota el permís per a poder entrar en aquest directori (amb la instrucció *cd*, per exemple).

Per a modificar els drets sobre un directori o fitxer, tenim les instruccions:

- ***chown***: canvia el propietari dels fitxers.
- ***chgrp***: canvia el grup propietari dels fitxers.
- ***chmod***: canvia els permisos específics (*rwx*) dels arxius.

Aquestes instruccions també permeten l'opció *-R*, que és recursiva si es tracta d'un directori.

5.3. Sistemes de fitxers: Xfs, Zfs i Btrfs

A més dels clàssics sistemes de fitxers del Linux, ext2, 3 o 4, apareixen cada vegada amb més força nous sistemes, molts procedents d'altres sistemes UNIX, com XFS (creat per Silicon Graphics) o ZFS (creat per Sun Microsystems, ara

mantingut per Oracle, originalment es va crear amb llicència oberta), i Btrfs, que va ser desenvolupat per Oracle originalment com a nou sistema de fitxers per al Linux, per a substituir les sèries ext2, 3, 4.

L'XFS va ser creat el 1993 per SGI, com a sistema de fitxers de 64 bits, d'altes prestacions, que incloïa funcionalitats de *journaling*, es va portar al *kernel* Linux el 2001, i algunes distribucions ja el comencen a incloure com a sistema per defecte (RHEL i algunes versions de CentOS). Entre les capacitats a destacar de XFS, s'inclou el fet de ser un sistema de fitxers de 64 bits, la qual cosa li permet arribar fins a 8 ExbiBytes d'emmagatzematge (2^63 -1 bytes), poden generar-se arxius de fins a 500 TB, el seu gran rendiment en operacions E/S en paral·lel, i la consistència de les dades, gràcies al seu sistema de *journaling* (que facilita la recuperació en cas de fallada) i a diverses optimitzacions per a suportar àmplies amplades de banda d'E/S, en maquinari dedicat.

Quant a l'administració, algunes tecles d'ordre:

mkfs.xfs

Crear un sistema de fitxers XFS en la unitat donada. Un paràmetre important és la grandària de bloc, que s'especifica amb –b size= , essent el rang vàlid normalment entre 512 bytes i 4.096 bytes (4 kB), el bloc depèn de la grandària total del sistema de fitxers (*filesystem*) i de la composició respecte al percentatge de fitxers petits. Si es hi ha un gran nombre d'aquests, es recomanen grandàries de bloc petits (512), però, si no, estem creant problemes de prestacions. La grandària per defecte és de 4.096, però es pot incloure qualsevol potència de 2 en la grandària de bloc.

L'opció, -l size, ens permet definir la grandària reservada als registres del *journal* que es desaran en el *log* del sistema. Això requereix espai de disc, que no es veurà com a espai disponible (per exemple, amb tecles d'ordre *df*). L'espai pot estar en la mateixa partició, o en partició o volum separats (es pot especificar amb l'opció –l), el defecte és en la mateixa partició. La grandària màxima per a un cas intern a la partició està entre 64 K blocs.

Hi ha altres opcions útils per a definir els grups d' *allocation* i les unitats de *stripe*, que són dos paràmetres que afecten les prestacions en funció del patró d'accessos de què es disposi. Els valors per defecte haurien de ser suficients per a un ús comú. En tot cas, consulteu la pàgina man d'mkfs.xfs per a aquests paràmetres. Amb mount, umount muntarem (o desmuntarem) les particions creades.

Un exemple de creació:

```
# mkfs.xfs -b size=1k -l size=10m /dev/sdc1
meta-data=/dev/sdc1 isize=256    agcount=18, agsize=4194304 blks
data =                       bsize=1024   blocks=71687152, imaxpct=25
```

```
       =                       sunit=0      swidth=0 blks, unwritten=0
naming =version 2               bsize=4096
log    =internal log            bsize=1024   blocks=10240, version=1
        =                       sunit=0 blks
realtime =none                    extsz=65536  blocks=0, rtextents=0

# mkdir -p /mnt/space

# mount /dev/sdc1 /mnt/space

(també podríem incloure en /etc/fstab perquè es muntés en arrencada):

/dev/sdc1  /mnt/space  xfs  defaults  0  0
```

Altres tecles d'ordre:

 # xfs_quota Permet el control de quotes d'espai per als usuaris

 # xfs_growfs /mount/point –D size Permet el creixement dels sistemes de fitxers dins del dispositiu de bloc on resideixen (l'espai és indicat en –D en blocs), fins i tot quan està muntat. Els sistemes de fitxers XFS no poden reduir-se *a posteriori*.

 # xfs_repair /dev/device Permet reparar el sistema de fitxers a partir del *log*. Si aquest està malmès, s'ha d'inicialitzar amb opcions com –L , encara que en perdre el *log* podem tenir algun dany o pèrdua de dades.

 # xfs_freeze –f | -u /mount/point Permet suspendre les escriptures (-f) per exemple per a fer un *snapshot* (els quals no suporta XFS, els delega al gestor de volums) de l'estat actual, i després tornar activar (-u). Si el sistema XFS està sobre volums LVM, no fa falta aquesta operació *xfs_freeze*, ja que els *snapshots* d'LVM ja s'encarreguen de suspendre i reassumir després de realitzar el *snapshot* LVM.

 # xfsdump -l 0 -f /dev/device /path/to/filesystem Permet una còpia de seguretat (*backup*) del sistema de fitxers (*file system*) sobre el (-f) dispositiu /dev/device (una cinta, un fitxer o un còpia de seguretat remota). Amb –l s'especifica el nivell d'abocament, 0 és un full còpia de seguretat, els següents 1-9 són còpies de seguretat incrementals a partir d'un full previ realitzat. Aquestes còpies de seguretat es poden restaurar *a posteriori* amb xfsrestore.

 # xfs_info Proporciona informació sobre els sistemes de fitxers:

Per al sistema de fitxers ZFS [Pow12] següent, es tracta d'una combinació de sistema de fitxers i gestor de volums lògics, dissenyat per Sun (i posteriorment mantingut per Oracle). Inclou proteccions contra corrupció de dades, compressió de dades, *snapshots* i suport de *copy-on-write*, COW, (tècnica que opti-

mitza múltiples còpies de les mateixes dades amb una única font, amb protecció durant escriptures), RAID-Z (model no estàndard de RAID semblant a RAID 5) amb suport natiu en ZFS i suport natiu d'ACL (llistes d'accés) per a NFSv4.

Per a ZFS també hi ha un projecte alternatiu denominat OpenZFS (suport ZFS-Like per a múltiples UNIX i Linux). De les implementacions possibles, ZFS original no s'acostuma a utilitzar a causa de llicències incompatibles; ZFS és llicenciat en CDDL, que és incompatible amb GPL. Una segona possibilitat és Native ZFS on Linux, versió creada pel Lawrence Livermore Labs (LLNL), normalment denominada ZFS on Linux , i que suporten directament Ubuntu i Gentoo des dels seus repositoris. També és possible trobar solucions basades en FUSE, de manera que el sistema de fitxers s'executi en espai d'usuari (de fet de manera semblant al cas de ntfs-3g). Però, en aquests cas, el suport té menys prestacions que un suport natiu de *kernel*.

Nota

ZFS on Linux http://zfsonlinux.org/

ZFS en Ubuntu: https://wiki.ubuntu.com/Kernel/Reference/ZFS

Si utilitzem ZFS on Linux en una distribució Fedora, podem consultar el procés bàsic d'instal·lació a http://zfsonlinux.org/fedora.html. Una vegada fet, ja disposem de suport natiu en *kernel* i podem començar a administrar ZFS. En el cas d'Ubuntu (a partir de la versió 15.10), ZFS està disponible com a paquet en la distribució (podem instal·lar-lo amb `sudo apt install zfsutils-linux`), i el podem utilitzar com a sistema de fitxers per a qualsevol disc o sistema de discos, a excepció del sistema arrel (tot i que aquesta opció està disponible a partir de la versió 16.04). Algunes tecles d'ordre útils, assumint que en aquesta sessió disposem de discos sdb i sdc totalment lliures per a implementar un *pool* d'emmagatzematge ZFS, són:

Nota

Els exemples següents destrueixen completament el contingut previ dels discos usats, es recomana solament usar discos extres per fer les proves, o per contra usar discos virtuals extres dins d'una maquina virtual.

```
# zpool create pdiscos /dev/sdb /dev/sdc
```

Pot ser que ens diguin que els discos són de tipus MBR. En aquest cas, s'haurà d'utilitzar *parted* per a posar-los una etiqueta de tipus GPT i inicialitzar la taula de particions nova; i tornem a procedir (en utilitzar *parted*, compte amb seleccionar, *select*, els discos adequats per a inicialitzar).

L'ZFS també pot utilitzar particions, però si es fa sobre el disc sencer, automàticament es creen particions i s'utilitza l'espai sencer. En aquest exemple s'ha creat un *pool* d'emmagatzematge denominat *pdiscos*, ZFS automàticament el muntarà sobre /pdiscos , que serà una col·lecció dels discos que integrarà l'espai de tots dos discos (semblant en concepte a un RAID0).

```
# zpool create mirror pdiscos /dev/sdb /dev/sdc
```

Ens crearia un *mirror* entre els dos discos (semblant en concepte a un RAID1). O en el cas d'un RAID-Z (amb al menys tres discos) podem:

```
# zpool create raidz pdiscos /dev/sdb /dev/sdc /dev/sdd
```

O també podem aconseguir una cosa semblant a RAID 6, amb un raidz-2 i un disc més (quatre discos mínim).

```
# zpool status
```

Estatus dels *pools* disponibles.

```
# zfs set mountpoint=/mnt/pdiscos pdiscos
```

Canviar el punt de muntatge del *pool* (per defecte la /).

```
# zfs list pdiscos
```

Llista de subvolums que inclou el sistema de fitxers.

```
# zfs create pdiscos/snaps
```

Crear un *sub-pools* de pdiscos denominat *snaps*.

```
# zfs snapshot pdiscos/snaps@avui
```

Crear un *snapshot* denominat *avui*. Per exemple, podríem usar la data amb *date* per a identificar-lo per estampa de temps.

El qual es pot muntar en qualsevol moment per a veure l'estat del moment en què es va realitzar:

```
# mount -t zfs pdiscos/snaps@avui /mnt/tmp
```

Muntar un *snapshot* per a examinar-lo.

```
# zfs list
```

Llista de *pools* i *subpools* presents.

```
# zfs list -t snapshot
```

Llista de *snapshots* presents.

```
# zfs destroy pdiscos/snaps@avui
```

Esborrar *snapshot*.

```
# zpool destroy pdiscos
```

Esborrar el *pool* sencer.

Btrfs (*B-tree file system*), el sistema de fitxers següent, és amb llicència GPL, que implementa *copy-on-write* per a Linux. L'han desenvolupat, entre d'altres, Oracle, Fujitsu i Red Hat. La idea principal d'aquest sistema és esdevenir el substitut dels ext2, 3 i 4 afegint tecnologies modernes que permetin *pooling*, *snapshots*, sumes de verificació (*checksums*), i suport de sistemes de fitxer en multidispositiu, opcions que són vitals per a necessitats d'emmagatzematge altes. La majoria de distribucions tenen ara suport natiu de Btrfs, i s'espera que, a mesura que s'incorporin noves funcionalitats planejades en el seu desenvolupament, es converteixi en el sistema de fitxers per defecte en gran part d'aquestes.

Per a l'administració, a més del suport del *kernel*, que en molts casos ja està actiu (si no *modprobre btrfs*), haurem d'instal·lar el paquet *btrfs-progs*, que inclou diverses utilitats per a la gestió del sistema de fitxers. Una vegada les tinguem (suposant que hi ha dos discos /dev/sdb i /dev/sdc amb partició primària, prèviament preparats amb fdisk o parted):

Nota

Més exemples estesos de Btrfs: http://www.funtoo.org/BTRFS_Fun

```
# mkfs.btrfs -L/diskb /dev/sdb1
# mkfs.btrfs -L/diskc /dev/sdc1
```

Vam crear el seus sistemes de fitxers amb les respectives etiquetes (-L). També podríem haver creat un *pool* amb:

```
# mkfs.btrfs -L/disks /dev/sdb1 /dev/sdc1
```

O un *mirror* amb:

```
# mkfs.btrfs -L/disks -draid1 /dev/sdb1 /dev/sdc1
```

Se suportaran els diferents raid0, 1, 5, 6 i 10. Una vegada disposem del sistema de fitxers creat podem examinar-ho amb:

```
# btrfs filesystem show /disks
```

Per exemple:

```
# btrfs filesystem show /disks
Label: '/disks'  uuid: 52eab416-66a7-4581-9487-243d84861331
 Total devices 2 FS bytes used 112.00KiB
 devid    1 size 4.40GiB used 929.00MiB path /dev/sdb1
 devid    2 size 4.40GiB used 909.00MiB path /dev/sdc
```

Que podem muntar després a partir de l'uuid que veiem amb una línia /etc/fstab com:

```
UUID=&lt;camp_uuid_anterior&gt;  /disks btrfs defaults 1 2
```

(col·locant en /disks com a punt de muntatge, encara que podria ser qualsevol altre punt).

Podem crear *snapshots* amb (/disks aquí fa referència al punt on es troba muntat el sistema de fitxers):

```
# btrfs subvolume snapshot /disks /disks/snapshot
```

I fer una llista dels *snapshots* amb

```
# btrfs subvolume list /disks
```

O destruir els *snapshots* amb:

```
# btrfs subvolume delete /disks/snapshot
```

Com hem vist al llarg d'aquests exemples, XFS, ZFS i Btrfs ofereixen força possibilitats, moltes més o menys equivalents, i amb graus diferents de suport, des del consolidat XFS, fins a l'experimental Btrfs, que segurament s'introduirà com a sistema de fitxers per defecte en el futur.

6. Usuaris i grups

Els usuaris d'un sistema GNU/Linux disposen d'un compte associat (definit amb algunes de les seves dades i preferències), juntament amb l'espai en disc perquè puguin desenvolupar els seus arxius i directoris. Aquest espai està assignat a l'usuari, i només el pot usar ell (tret que els permisos especifiquin coses diferents).

Dins dels comptes associats a usuaris, en podem trobar diferents tipus:

- **El de l'administrador**, amb identificador *root*, que només és (o hauria de ser) utilitzat per a les operacions d'administració. L'usuari *root* és el que disposa de més permisos i accés complet a la màquina i als arxius de configuració. Per tant, també és el que més mal pot causar per errors o omissions. És millor evitar usar el compte de *root* com si fos un usuari més, per la qual cosa es recomana deixar-lo només per a operacions d'administració.

- **Comptes d'usuaris**: els comptes normals per a qualsevol usuari de la màquina tenen els permisos restringits a l'ús de fitxers del seu compte, i a algunes altres zones particulars (per exemple, els temporals en /tmp), i també a utilitzar alguns dispositius per als quals s'hagin habilitat permisos.

- **Comptes especials dels serveis**: *lp*, *wheel*, *www-data*... comptes que no són usats per persones, sinó per serveis interns del sistema, que els usa sota aquests noms d'usuari. Alguns dels serveis també són usats sota l'usuari de *root* (encara que per raons de seguretat s'hauria d'evitar).

Un usuari normalment es crea mitjançant l'especificació d'un nom (o identificador d'usuari), una paraula de pas (*contrasenya*) i un directori personal associat (el compte).

La informació dels usuaris del sistema (local, si no és que es fan servir sistemes externs de directori, com ara els YP NIS, NIS+ o LDAP, que veurem més endavant) està inclosa en els arxius següents:

```
/etc/passwd
/etc/shadow
/etc/group
/etc/gshadow
```

Unes línies de /etc/passwd podrien ser:

```
joan:x:1000:1000:Joan Garcia,,,:/home/joan:/bin/bash
```

```
root:x:0:0:root:/root:/bin/bash
```

en què s'indica (si apareixen :: seguits és que el camp és buit):

- *joan*: identificador d'usuari en el sistema.
- *x*: paraula de pas de l'usuari codificada, si hi ha una *x* és que es troba en el fitxer /etc/shadow.
- *1000*: codi de l'usuari; l'usa el sistema com a codi d'identitat de l'usuari.
- *1000*: codi del grup principal a què pertany; la informació del grup es troba en /etc/group.
- *Joan García*: comentari; se sol col·locar el nom complet de l'usuari, o algun comentari per a identificar l'objectiu del compte.
- */home/juan*: directori personal associat al seu compte.
- */bin/bash*: intèrpret d'ordres interactiu que utilitzarà l'usuari en interactuar amb el sistema, en mode text, o amb el terminal gràfic. En aquest cas, l'intèrpret Bash de GNU, que és l'utilitzat per defecte. El fitxer /etc/passwd solia contenir les paraules de pas dels usuaris en forma xifrada, però el problema era que qualsevol usuari podia veure el fitxer, i en el seu moment es van dissenyar *cracks* que intentaven trobar en forma bruta la paraula de pas, mitjançant la paraula de pas xifrada com a punt de partida (paraula codificada amb el sistema *crypt)*.

Per a evitar això, avui dia ja no es col·loquen les paraules de pas en aquest arxiu, sinó només una *x* que indica que es troben en un altre fitxer, que és només de lectura per a l'usuari *root*, */etc/shadow,* el contingut del qual podria ser semblant al següent:

```
juan:a1gNcs82ICst8CjVJS7ZFCVnu0N2pBcn/:12208:0:99999:7:::
```

en què es troba l'identificador de l'usuari juntament amb la paraula de pas xifrada. A més, apareixen (com a camps separats per ":" amb informació sobre la contrasenya):

- Dies des de l'1 de gener de 1970 en què la paraula de pas es va canviar per última vegada.
- Dies que falten perquè es canviï (*0* vol dir que no s'ha de canviar).
- Dies després dels quals cal canviar-la (és a dir, termini de canvi).
- Dies en què l'usuari serà avisat abans que li expiri.
- Dies, una vegada expirat, que es produirà la deshabilitació del compte.
- Dies des de l'1 de gener de 1970 en què el compte està deshabilitat.

- I un camp reservat.

A més, les claus de xifratge poden ser més difícils, ja que ara es poden utilitzar diversos mètodes com md5, sha 256 o 512, blowfish o DES (acostumen a identificar-se en la clau per un codi x, on x = 1, 5, 6 2a, o altres per als indicats) per a protegir les paraules de pas dels usuaris.

En /etc/group hi ha la informació dels grups d'usuaris:

```
jose:x:1000:
```

en què tenim:

```
nom-grup:contrasenya-grup:identificador-del-grup:llista-usuaris
```

La llista d'usuaris del grup pot ser present o no, ja que la informació ja està en /etc/passwd, i no se sol posar en /etc/group. Si s'hi posa, sol aparèixer com una llista d'usuaris separada per comes. Els grups també poden tenir una contrasenya associada (encara que no sol ser tan normal), com en el cas dels d'usuari, i llavors també hi ha un fitxer de tipus shadow: /etc/gshadow.

Altres fitxers interessants són els del directori /etc/skel, en què es troben els fitxers que s'inclouen en cada compte d'usuari en crear-lo. Recordeu que, com hem vist amb els intèrprets d'ordres interactius, podem tenir uns *scripts* de configuració que s'executen en entrar o sortir del compte. En el directori *skel* es desen els "esquelets" que es copien al directori de cada usuari en crear-lo. Sol ser responsabilitat de l'administrador crear uns fitxers adequats per als usuaris, posant les rutes necessàries d'execució, la inicialització de variables de sistema, les variables que es necessitin per al programari, etc.

A continuació, veurem una sèrie d'instruccions útils per a aquesta administració d'usuaris (n'esmentem la funcionalitat i en el taller farem algunes proves):

- ***useradd***: afegeix un usuari al sistema.
- ***userdel***: esborra un usuari del sistema.
- ***usermod***: modifica un usuari del sistema.
- ***groupadd, groupdel, groupmod***: el mateix per a grups.
- ***newusers, chpasswd***: poden ser d'utilitat en grans instal·lacions amb molts usuaris, ja que permeten crear diversos comptes des de la informació introduïda en un fitxer (*newusers*) o bé canviar les contrasenyes a un gran nombre d'usuaris (*chpasswd*).
- ***chsh***: canvia l'intèrpret d'ordres per defecte de l'usuari.
- ***chfn***: canvia la informació de l'usuari, present en el comentari del fitxer /etc/passwd.
- ***passwd***: canvia la contrasenya d'un usuari. Es pot executar com a usuari, i llavors demana la contrasenya antiga i la nova. En el cas de fer-ho, *root* ha d'especificar l'usuari a qui canviarà la contrasenya (si no, estaria canviant

la seva) i no necessita la contrasenya antiga. És potser la instrucció més usada per *root*, quan als usuaris se'ls oblida la contrasenya antiga.

- ***su***: una espècie de canvi d'identitat. L'utilitzen tant usuaris com *root* per a canviar l'usuari actual. En el cas de l'administrador, és bastant utilitzat per a provar que el compte de l'usuari funcioni correctament. Hi ha diferents variants: *su* (sense paràmetres, serveix per a passar a usuari *root*, prèvia identificació, i permet, quan estem en un compte d'usuari, passar a *root* per a fer alguna tasca). La instrucció *su iduser* canvia l'usuari a *iduser*, però deixant l'entorn com està, és a dir, en el mateix directori. L'ordre *su - iduser* fa una substitució total, com si el segon usuari hagués entrat en el sistema fent una *connexió*.

Respecte a l'administració d'usuaris i grups, el que hem comentat aquí fa referència a l'administració local d'una sola màquina. En sistemes amb múltiples màquines que comparteixen els usuaris se sol utilitzar un altre sistema de gestió de la informació dels usuaris. Aquests sistemes, denominats genèricament *sistemes d'informació de xarxa* o *serveis de directori*, com NIS, NIS+ o LDAP, utilitzen bases de dades per a emmagatzemar la informació dels usuaris i grups, de manera que s'utilitzen màquines servidores, en què s'emmagatzema la base de dades, i altres màquines clients, en què es consulta aquesta informació. Això permet tenir una sola còpia de les dades dels usuaris (o diverses de sincronitzades), i que aquests puguin entrar a qualsevol màquina disponible del conjunt administrat amb aquests sistemes. A més, aquests sistemes incorporen conceptes addicionals de jerarquies o dominis/zones de màquines i recursos, que permeten representar adequadament els recursos i el seu ús en organitzacions amb diferents estructures d'organització interna del seu personal i les seves seccions internes.

Podem comprovar si estem en un entorn de tipus NIS si en les línies *passwd* i *group* de l'arxiu de configuració /etc/nsswitch.conf apareix *files* en primer terme, si estem treballant amb els fitxers locals, o bé *nis* o *nisplus*, segons el sistema amb què estiguem treballant. En general, per a l'usuari simple no representa cap modificació, ja que la gestió de les màquines li és transparent, i més si es combina amb fitxers compartits per NFS, que permet disposar del seu compte sense que importi amb quina màquina treballa. La major part de les instruccions anteriors es poden continuar usant sense problema sota NIS o NIS+; són equivalents a excepció del canvi de contrasenya, que en lloc de *passwd* es fa amb *yppasswd* (NIS) o *nispasswd* (NIS+), encara que sol ser habitual que l'administrador els rebategi (amb un enllaç) a *passwd*, amb la qual cosa els usuaris no notaran la diferència.

Veurem aquest i altres modes de configuració en les unitats d'administració de xarxa.

7. Servidors d'impressió

El sistema d'impressió de GNU/Linux [Gt] [Smi02] està heretat de la variant BSD de UNIX. Aquest sistema es denominava LPD (Line Printer Daemon). És un sistema d'impressió molt potent, ja que integra capacitats per a gestionar tant impressores locals com de xarxa i ofereix tant el client com el servidor d'impressió. De manera semblant també UNIX ha disposat generalment del System V Line Printer (o LPR), que era el sistema comú en les altres variants de UNIX. GNU/Linux ha integrat originalment tots dos sistemes, bé usant principalment LPD i emulant LPR, o depenent de la distribució integrant-ne per defecte un o un altre.

Potència i flexibilitat

Els sistemes UNIX disposen, potser, dels sistemes d'impressió més potents i complexos, que aporten una gran flexibilitat als entorns d'impressió.

LPD és un sistema bastant antic, ja que es remunta als orígens de la branca BSD de UNIX (mitjan anys vuitanta). Per tant, a LPD li sol faltar suport per als dispositius moderns, ja que en origen el sistema no va estar pensat per als tipus d'impressores actuals. Tampoc no va ser concebut com un sistema basat en controladors de dispositiu, ja que es produïen només impressores en sèrie o paral·lel d'escriptura de caràcters de text.

Per a la situació actual, el sistema LPD es combina amb un altre programari comú, com el sistema Ghostscript, que ofereix sortida de tipus PostScript per a un rang molt ampli d'impressores per a les quals té controladors. A més, se sol combinar amb algun programari de filtratge, que segons el tipus de document per imprimir, selecciona filtres adequats per a adaptar la impressió de documents o formats binaris al sistema d'impressió de destinació. Així, normalment el procés que se segueix és (bàsicament):

1) El treball és iniciat per una instrucció del sistema LPD.

2) El sistema de filtre identifica quin tipus de treball (o fitxer) és utilitzat i converteix el treball a un fitxer PostScript de sortida, que és el que s'envia a la impressora. En GNU/Linux i UNIX, la majoria d'aplicacions suposen que la sortida serà cap a una impressora PostScript, i moltes generen sortida PostScript directament, i per aquesta raó es necessita el pas següent.

3) Ghostscript s'encarrega d'interpretar el fitxer PostScript rebut, i segons el controlador de la impressora al qual ha estat enviat el treball, fa la conversió al format propi de la impressora. Si és de tipus PostScript, la impressió és directa; si no, caldrà fer-ne la traducció. El treball s'envia a la cua d'impressió.

Com hem dit, a més del sistema d'impressió LPD (amb origen en els BSD UNIX), també hi ha el denominat sistema *System V* (d'origen en l'altra branca UNIX System V) o LPR. Per compatibilitat, actualment la major part de UNIX els integra tots dos, de manera que o bé un o un altre és el principal, i l'altre se

simula sobre el principal. En el cas de GNU/Linux, passa una cosa semblant; segons la instal·lació que fem podem tenir només les instruccions LPD de sistema d'impressió, però també serà habitual disposar de les instruccions System V. Una manera senzilla d'identificar els dos sistemes (BSD o System V) és amb la instrucció principal d'impressió (la que envia els treballs al sistema), que en BSD és *lpr*, i en System V és *lp*.

Aquest era el panorama inicial dels sistemes d'impressió de GNU/Linux, però en els últims anys han sorgit més sistemes, que permeten més flexibilitat i disposició de controladors per a les impressores. Els dos principals sistemes són CUPS i, en grau menor, LPRng (de fet, ja obsolet, que es va utilitzar en algunes versions de Fedora, i ja no el comentarem en aquesta revisió del material; es pot trobar en edicions anteriors). Últimament és CUPS l'estàndard *de facto* per a GNU/Linux, encara que els altres sistemes han de ser suportats per compatibilitat amb sistemes UNIX existents.

Els dos (tant CUPS com LPRng) són una espècie de sistema de nivell més alt, però que no es diferencien gaire amb vista a l'usuari respecte als BSD i System V estàndard. Per exemple, s'utilitzen les mateixes instruccions clients (o compatibles en opcions) per a imprimir. Per a l'administrador sí que representen diferències, ja que els sistemes de configuració són diferents. En certa manera, podem considerar LPRng i CUPS com a noves arquitectures de sistemes d'impressió, que són compatibles per a l'usuari amb les instruccions antigues.

En les distribucions GNU/Linux actuals podem trobar els diferents sistemes d'impressió. Si la distribució és antiga, pot ser que porti incorporat tan sols el sistema BSD LPD. En les actuals, tant Debian com Fedora/Red Hat utilitzen CUPS. En algunes versions de Red Hat hi havia una eina, *Print switch*, que permitia canviar el sistema, commutar de sistema d'impressió, encara que últimament només està disponible CUPS. En Debian es poden instal·lar tots dos sistemes, però són exclusius, i només un pot gestionar la impressió.

En el cas de Fedora, el sistema d'impressió per defecte és CUPS (LPRng va desaparèixer en Fedora Core 4), i l'eina *Print switch* ja no existeix perquè no és necessària. S'utilitza *system-config-printer* per a la configuració de dispositius. Debian, per defecte, utilitzava BSD LPD, però ja és comú instal·lar CUPS (i és l'opció per defecte en noves versions), i també pot utilitzar LPRng. A més, es pot recordar que també teníem la possibilitat (vista en la unitat de migració) d'interaccionar amb sistemes Windows mitjançant protocols Samba, que permetien compartir les impressores i accedir-hi.

Respecte a cada un dels sistemes [Gt]:

- **BSD LPD**: és un dels estàndards de UNIX, i algunes aplicacions assumeixen que tindran les instruccions i el sistema d'impressió disponibles, per la qual cosa, tant LPRng com CUPS emulen el funcionament i les instruccions de BDS LPD. El sistema LPD és utilitzable, però no gaire configurable,

sobretot en el control d'accés; per això les distribucions s'han mogut als altres sistemes més moderns.

- **LPRng**: es va dissenyar per a ser un reemplaçament del BSD; per tant, la major part de la configuració és semblant i només difereix en alguns fitxers de configuració.

- **CUPS**: es tracta d'una desviació més important del BSD original, i la configuració és pròpia. Es proporciona informació a les aplicacions sobre les impressores disponibles (també en LPRng). En CUPS, tant el client com el servidor han de disposar de programari CUPS.

Els dos sistemes tenen emulació de les instruccions d'impressió de System V.

Per a la impressió en GNU/Linux, cal tenir en compte diversos aspectes:

- **Sistema d'impressió que s'utilitza**: BSD, CUPS, o LPRng (avui pràcticament obsolet).

- **Dispositiu d'impressió** (impressora): pot disposar de connexió local a una màquina o estar col·locada en xarxa. Les impressores actuals poden ser col·locades per connexions locals a una màquina mitjançant interfícies en sèrie, paral·lel, USB, etc., o disponibles simplement en xarxa, com una màquina més, o amb protocols especials de propietat. Les connectades a xarxa poden actuar elles mateixes de servidor d'impressió (per exemple, moltes de làser són servidors BSD LPD), o bé es poden penjar d'una màquina que actuï de servidor d'impressió.

- **Protocols de comunicació** utilitzats amb la impressora o el sistema d'impressió: ja sigui TCP/IP directe (per exemple, una HP amb LPD), o bé altres de més alt nivell sobre TCP/IP, com IPP (CUPS), JetDirect (algunes impressores HP), etc. Aquest paràmetre és important, ja que l'hem de conèixer per a instal·lar la impressora en un sistema.

- **Sistema de filtres usat**: cada sistema d'impressió en suporta un o diversos.

- **I els controladors de les impressores**: en GNU/Linux n'hi ha bastants tipus diferents; podem esmentar, per exemple, controladors de CUPS, propis o dels fabricants (per exemple, HP i Epson en proporcionen); Gimp, el programa de retoc d'imatges, també té controladors optimitzats per a la impressió d'imatges; Foomatic, un sistema de gestió de controladors que funciona amb la majoria de sistemes (CUPS, LPD, LPRng i d'altres); els controladors de Ghostscript, etc. Gairebé totes les impressores tenen un o més controladors d'aquests conjunts.

Nota

Podeu trobar informació de les impressores més adequades i dels controladors a: http://www.openprinting.org/printers

Respecte a la part client del sistema, les instruccions bàsiques són iguals per als diferents sistemes. Aquestes són les instruccions del sistema BSD (cada sistema suporta emulació d'aquestes instruccions):

- ***lpr***: envia un treball a la cua de la impressora per defecte (o a la que se selecciona); el dimoni d'impressió (*lpd*) s'encarrega d'enviar-lo a la cua corresponent i assigna un número de treball, que serà usat amb les altres instruccions. La impressora, per defecte, estaria indicada per una variable de sistema PRINTER, o s'utilitzarà la primera que estigui definida. En alguns sistemes s'utilitza la cua *lp* (com a nom per defecte).

```
lpr -Pepson dades.txt
```

Aquesta instrucció enviaria el fitxer *dades.txt* a la cua d'impressió associada a una impressora que hem definit com a "epson".

Nota

Si no es disposa d'impressora, per a provar el sistema d'impressió es pot instal·lar el paquet cups-pdf (si la distribució suporta CUPS) que instal·la una impressora virtual en el sistema amb sortida en fitxers PDF.

- ***lpq***: ens permet examinar els treballs existents en la cua.
  Aquesta instrucció ens mostra els treballs en cua, amb l'ordre i les mides. Els fitxers poden aparèixer amb noms diferents, ja que depèn de si els hem enviat amb *lpr* o amb una altra aplicació que pot canviar els treballs de nom en enviar-los, o si han hagut de passar per algun filtre en convertir-los.

```
# lpq -P epson
Rank Owner      Job    Files        Total Size
1st  juan       15     dades.txt    74578 bytes
2nd  marta      16     fpppp.F      12394 bytes
```

- ***lprm:*** elimina treballs de la cua. Podem especificar un número de treball, o un usuari per a cancel·lar els treballs.

```
lprm -Pepson 15
```

Elimina el treball amb ID 15 de la cua.

Respecte a la part administrativa (en BSD), la instrucció principal seria *lpc*. Aquesta instrucció permet activar i desactivar cues, moure treballs en l'ordre de les cues i activar o desactivar les impressores (es poden rebre treballs a les cues, però no s'envien a les impressores).

Es pot esmentar, així mateix que, per al cas de System V, les instruccions d'impressió solen també estar disponibles, simulades sobre les de BSD. En el cas client, les instruccions són *lp, lpstat, cancel* i, per a temes d'administració, *lpadmin, accept, reject, lpmove, enable, disable, lpshut*.

En els apartats següents veurem com cal configurar un servidor d'impressió per a dos dels sistemes principals. Aquests servidors serveixen tant per a la impressió local com per a atendre les impressions de clients de xarxa (si estan habilitats).

7.1. BSD LPD

En el cas del servidor BSD LPD, hi ha dos fitxers principals per examinar: per una part, la definició de les impressores en /etc/printcap i, per l'altra, els permisos d'accés per xarxa en /etc/hosts.lpd.

Respecte als permisos, per defecte BSD LPD només deixa accés local a la impressora, i per tant, cal habilitar-lo expressament en /etc/hosts.lpd.

Exemple

El fitxer podria ser:

```
#arxiu hosts.lpd
second
first.the.com
192.168.1.7
+@groupnis
-three.the.com
```

que indicaria que està permesa la impressió en una sèrie de màquines, mostrades bé pel seu nom DNS o per l'adreça IP. Es poden afegir grups de màquines que pertanyin a un servidor NIS (com en l'exemple *groupnis*) o bé permetre accés a determinades màquines indicant-ho amb un guionet "-".

Quant a la configuració del servidor en /etc/printcap, es defineixen entrades, en què cada una representa una cua del sistema d'impressió a la qual poden anar a parar els treballs. La cua pot estar tant associada a un dispositiu local com a un servidor remot, ja sigui una impressora o un altre servidor.

En cada entrada, hi pot haver les opcions:

- ***lp*=**: ens indica a quin dispositiu està connectada la impressora; per exemple *lp = /dev/lp0* indicaria el primer port paral·lel. Si la impressora és de tipus LPD, per exemple una impressora de xarxa que accepta el protocol LPD (com una HP), llavors podem deixar el camp buit i emplenar els següents.

- ***rm*=**: adreça amb nom o IP de la màquina remota que disposa de la cua d'impressió. Si es tracta d'una impressora de xarxa, serà l'adreça d'aquesta.

- ***rp*=**: nom de la cua remota, a la màquina indicada abans amb *rm*.

```
# Entrada d'una impressora local
lp|epson|Epson C62:\
 :lp=/dev/lp1:sd=/var/spool/lpd/epson:\
 :sh:pw#80:pl#72:px#1440:mx#0:\
 :if = /etc/magicfilter/StylusColor@720dpi-filter:\filtro
 :af = /var/log/lp-acct:lf = /var/log/lp-errs:
# Entrada d'impressora remota
hpremota|hpr||hp remota del departament:\
 :lp = :\
 :rm = servidor:rp = cuahp:\
 :lf = /var/adm/lpd_rem_errs:\arxiu de log.
 :sd = /var/spool/lpd/hpremota:gestió de cues local associada
```

7.2. CUPS

CUPS és una nova arquitectura per al sistema d'impressió bastant diferent; té una capa de compatibilitat amb BSD LPD, que li permet interaccionar amb servidors d'aquest tipus. Suporta també un nou protocol d'impressió anomenat *IPP* (basat en HTTP), però només disponible quan client i servidor són de tipus CUPS. A més, utilitza un tipus de controladors denominats *PPD* que identifiquen les capacitats de la impressora. CUPS ja porta alguns d'aquests controladors, i alguns fabricants també n'ofereixen (com HP i Epson, entre altres).

CUPS té un sistema d'administració completament diferent, basat en diferents fitxers: /etc/cups/cupsd.conf centralitza la configuració del sistema d'impressió, /etc/cups/printers.conf controla la definició d'impressores i /etc/cups/classes.conf els grups d'impressores.

En /etc/cups/cupsd.conf configurem el sistema segons una sèrie de seccions de l'arxiu i les directives de les diferents accions. L'arxiu és bastant gran; destacarem algunes directives importants:

- ***Allow***: ens permet especificar quines màquines podran accedir al servidor, ja sigui grups o màquines individuals, o segments IP de xarxa.

- ***AuthClass***: permet indicar si es demanarà que s'autentifiquin els usuaris clients o no.

- ***BrowseXXX***: hi ha una sèrie de directives relacionades amb la possibilitat d'examinar la xarxa per a trobar impressores servides. Aquesta possibilitat està activada per defecte (*browsing* en *on*); per tant, trobarem disponibles totes les impressores disponibles a la xarxa. La podem desactivar, per a observar només les impressores que hàgim definit. Una altra opció important és *BrowseAllow*, que diu a qui li donem la possibilitat de preguntar per les nostres impressores. Per defecte està habilitada, per la qual cosa qualsevol pot veure la nostra impressora des de la xarxa.

Es pot assenyalar que CUPS, en principi, està pensat perquè tant els clients com el servidor funcionin sota el mateix sistema; si els clients utilitzen LPD, cal instal·lar un dimoni de compatibilitat anomenat *cups-lpd* (en paquets com *cupsys-bsd*). En aquest cas, CUPS accepta treballs que provinguin d'un sistema LPD, però no controla els accessos (*cupsd.conf* només serveix per al sistema CUPS mateix), per la qual cosa caldrà implementar alguna estratègia de control d'accés, de tipus tallafocs.

Per a l'administració des de línia d'instruccions, CUPS és una mica peculiar, ja que accepta tant instruccions LPD com System V en els clients, i l'administració se sol fer amb la instrucció *lpadmin* de System V. Quant a ei-

nes gràfiques, disposem de *gnome-cups-manager*, *gtklp*, utilitats com *system-config-printers* o la interfície per web que porta el sistema CUPS mateix, accessible en http://localhost:631.

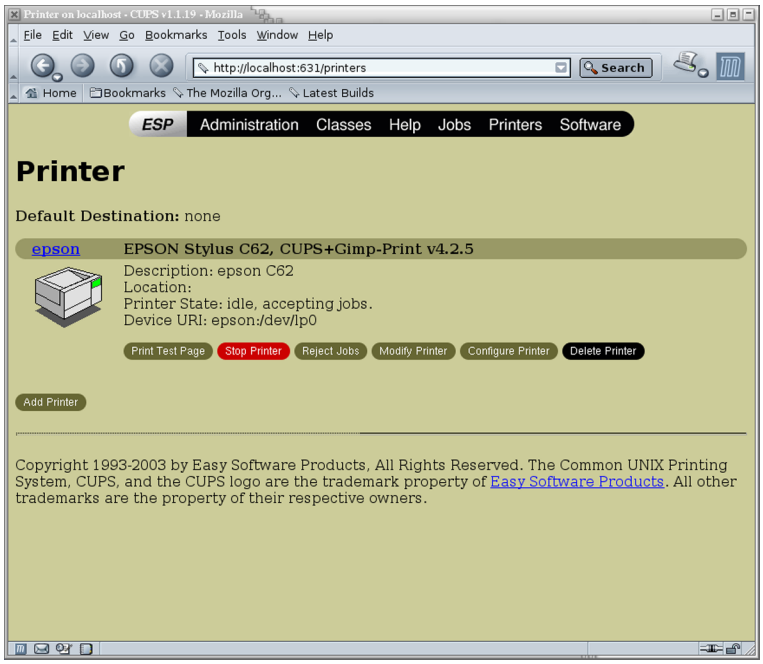


Figura 2. Interfície per a l'administració del sistema CUPS.

8. Discos i gestió de *filesystems*

Respecte a les unitats d'emmagatzemament, com hem vist, tenen una sèrie de dispositius associats, depenent del tipus d'interfície:

- **IDE** (només en PC antics): dispositius com:
  - /dev/hda *disk master*, primer connector IDE;
  - /dev/hdb *disk slave*, del primer connector;
  - /dev/hdc *master*, segon connector;
  - /dev/hdd *slave*, segon connector.
- **SCSI**: dispositius /dev/sda /dev/sdb... seguint la numeració que tinguin els perifèrics al bus SCSI. Els discos SATA i IDE del mateix sistema també solen seguir aquesta nomenclatura, a causa de la capa d'emulació SCSI present en el nucli per a aquests dispositius.
- **Disquets** (obsolets, però que es poden trobar en PC antics): dispositius /dev/fdx, amb *x* com a número de disquetera (començant pel 0). Hi ha diferents dispositius, depenent de la capacitat del disquet; per exemple, el disquet de 1,44 MB a la disquetera A seria */dev/fd0H1440*.

Respecte a les particions presents, el número que segueix el dispositiu representa l'índex de la partició dins del disc, i és tractat com un dispositiu independent: /dev/hda1 és la primera partició del primer disc IDE, o /dev/sdc2, la segona partició del tercer dispositiu SCSI. En el cas dels discos amb BMR, permeten quatre particions denominades *primàries* i un major nombre d'esteses (o lògiques). Així, si en /dev/sdan, *n* és inferior o igual a 4, es tractarà d'una partició primària; si no, es tractarà d'una partició lògica amb *n* superior o igual a 5.

Amb els discos i els sistemes de fitxers (*filesystems*) associats, els processos bàsics que podem fer s'engloben en els següents:

- **Creació de particions**, o modificació. Mitjançant instruccions com *fdisk* o semblants (*cfdisk, sfdisk*). I *parted*, com ja hem comentat, per a màquines amb suport de GPT (nous PC amb *firmware* UEFI).
- **Formatació de disquets**: en cas de disquets, es poden utilitzar diferents eines: *fdformat* (formatació de baix nivell), *superformat* (formatació a diferents capacitats en format MSDOS), *mformat* (formatació específica creant un *filesystem* MSDOS estàndard).

- **Creació de *filesystems* Linux**, en particions, mitjançant la instrucció *mkfs*. Hi ha versions específiques per a crear *filesystems* diferents: *mkfs.ext2*, *mkfs.ext3*, i també *filesystems* no Linux: *mkfs.ntfs*, *mkfs.vfat, mkfs.msdos*, *mkfs.minix* o d'altres. Per a CD-ROM, com *mkisofs,* a l'hora de crear els ISO9660 (amb extensions *joliet* o *rockridge*), que puguin ser una imatge del que després s'acabarà gravant sobre un CD/DVD, i juntament amb instruccions com *cdrecord* (o *wodim*), es podrà finalment crear/gravar els CD/DVD. Un altre cas particular és l'ordre *mkswap,* que permet crear àrees de *swap* en particions que, més tard, es poden activar o desactivar amb *swapon* i *swapoff*.

- **Muntatge dels *filesystems***: instruccions *mount, umount*.

- **Verificació d'estat**: la principal eina de verificació de *filesystems* Linux és la instrucció *fsck*. Aquesta instrucció comprova les diferents àrees del sistema de fitxers per a verificar la consistència i comprovar possibles errors i, en els casos en els quals sigui possible, corregir-los. El sistema mateix activa automàticament la instrucció *fsck* en l'arrencada quan detecta situacions en què s'ha produït una parada incorrecta (una apagada elèctrica o accidental de la màquina), o bé ha passat un cert nombre de vegades des que el sistema s'ha engegat. Aquesta comprovació sol comportar cert temps, normalment alguns minuts (depenent de la mida de dades). També hi ha versions particulars per a altres sistemes de fitxers: *fsck.ext2*, *fsck.ext3*, *fsck.vfat*, *fsck.msdos*, etc. El procés de l'*fsck* es fa amb el dispositiu en mode de "només lectura" amb particions muntades. Es recomana desmuntar les particions per a fer el procés si es detecten errors i cal aplicar correccions. En determinats casos, per exemple, si el sistema per a comprovar és l'arrel / i es detecta algun error crític, ens demanarà que canviem de mode d'execució del sistema (*runlevel*) a un mode només *root* i fem allà la verificació. En general, si cal fer la verificació, es recomana fer-les en mode superusuari (podem commutar de mode de *runlevel* amb les instruccions *init* o *telinit*).

- **Processos de còpia de seguretat**: ja siguin del disc, blocs de disc, particions, *filesystems*, fitxers... Hi ha diverses eines útils per a això: *tar* ens permet copiar fitxers cap a un fitxer o a unitats de cinta; *cpio,* de manera semblant, pot fer còpies de fitxers cap a un fitxer; tant *cpio* com *tar* mantenen la informació de permisos i propietaris dels fitxers; *dd* permet còpies, ja sigui de fitxers, dispositius, particions o discos a fitxer; és una mica complex i cal conèixer informació de baix nivell, tipus, mides, nombre de blocs o sectors... Es pot enviar també a cintes.

- **Utilitats diverses**: certes ordres individuals (algunes dependran d'utilitats exclusives per a un sistema de fitxers concret), algunes utilitzades pels processos anteriors per a fer tractaments diferents: *badblocks* per a trobar blocs defectuosos al dispositiu; *dumpe2fs* per a obtenir informació sobre *filesys-*

tems Linux; *tune2fs* permet fer processos de *tunning* de *filesystems* Linux de tipus *ext2*, *ext3* o *ext4* i ajustar diferents paràmetres de comportament.

A continuació, destaquem dos temes relacionats amb la concepció de l'espai d'emmagatzemament, que són utilitzats en diversos ambients per a la creació base de l'espai d'emmagatzemament: l'ús de RAID en programari i la creació de volums dinàmics.

8.1. RAID en programari

La configuració de discos mitjançant esquemes RAID és un dels esquemes d'emmagatzemament d'alta disponibilitat més usats actualment, quan disposem de diversos discos per a implementar els nostres sistemes de fitxers.

L'enfocament principal de les diferents tècniques existents és la tolerància a errors que es proporciona des d'un nivell de dispositiu, el conjunt de discos, a diferents tipus possibles d'errors tant físics com de sistema, per a evitar les pèrdues de dades o els errors de coherència en el sistema. Així, també alguns esquemes que estan dissenyats per a augmentar les prestacions del sistema de discos, ampliant l'amplada de banda disponible cap al sistema i les aplicacions.

Típicament, avui dia el RAID en maquinari (mitjançant targetes controladores de maquinari) es pot trobar en servidors empresarials (i comencen a tenir certa presència en equips d'escriptori, amb plaques base amb capacitats per a alguns RAID bàsics), en què es troben disponibles diferents solucions de maquinari que compleixen aquests requisits de fiabilitat i de maximitzar prestacions. En particular, per a ambients amb aplicacions intensives en disc, com reproducció d'àudio o vídeo en temps real, o grans bases de dades.

Convé destacar un error comú en el tema RAID, que proporciona certes capacitats de tolerància a errors, però no evita haver de fer còpies de seguretat de les dades disponibles de manera periòdica. Si se supera la capacitat de tolerància a errors, també es perden dades (en alguns casos, completament).

En general, aquest maquinari es troba en forma de targetes (o integrat a la màquina) de tipus controladora RAID de discos, que implementen la gestió d'un o més nivells (de l'especificació RAID), sobre un conjunt de discos (des d'un mínim de dos discos) administrat per aquesta controladora.

En RAID es distingeix una sèrie de nivells (o configuracions possibles) que es poden proporcionar (cada fabricant de maquinari, o el programari concret, pot suportar un o diversos d'aquests nivells). Cada nivell de RAID s'aplica sobre un conjunt de discos, de vegades denominat *array* RAID (o matriu de discos RAID), que solen ser (idealment) discos iguals en mida (o iguals en mides per grups). Per exemple, per a fer un cas de matriu es podrien utilitzar quatre discos de 500 GB, o en un altre cas, dos grups (a 500 GB) de dos discos, un de 150 GB i un altre de 350 GB. En alguns casos de controladors de maquinari, no

es permet que els discos (o en grups) siguin de diferents mides; en d'altres es poden utilitzar, però la matriu queda definida per la mida del disc (o grup) més petit.

Descrivim conceptes bàsics d'alguns nivells en la llista següent (teniu en compte que, en alguns casos, la terminologia no és plenament acceptada, i pot dependre de cada fabricant):

- **RAID 0**: es distribueixen les dades equitativament entre un o més discos sense informació de paritat o redundància; no s'està oferint tolerància a l'error. Només s'estan repartint dades; si el disc falla físicament la informació es perd i l'hem de recuperar a partir de còpies de seguretat. El que sí que augmenta és el rendiment, depenent de la implementació de RAID 0, ja que les operacions de lectura i escriptura es dividiran entre els diferents discos.

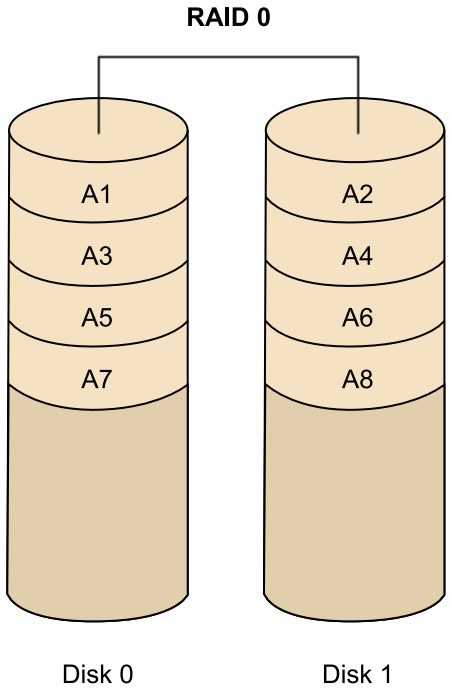


Figura 3

- **RAID 1**: es crea una còpia exacta (*mirror)* en un conjunt de dos o més discos (denominada *matriu RAID*). En aquest cas, resulta útil per al rendiment de lectura (que es pot arribar a incrementar de manera lineal amb el nombre de discos), i en especial per a disposar de tolerància a l'error d'un dels discos, ja que (per exemple, amb dos discos) es disposa de la mateixa informació. RAID 1 sol ser adequat per a alta disponibilitat, com en entorns de 24 x 7, en què hem de disposar críticament dels recursos. Aquesta configuració ens permet també (si el maquinari ho suporta) l'intercanvi en calent (*hot-swap*) dels discos. Si detectem l'error en un, el podem substituir, sense apagar el sistema, per un disc nou.

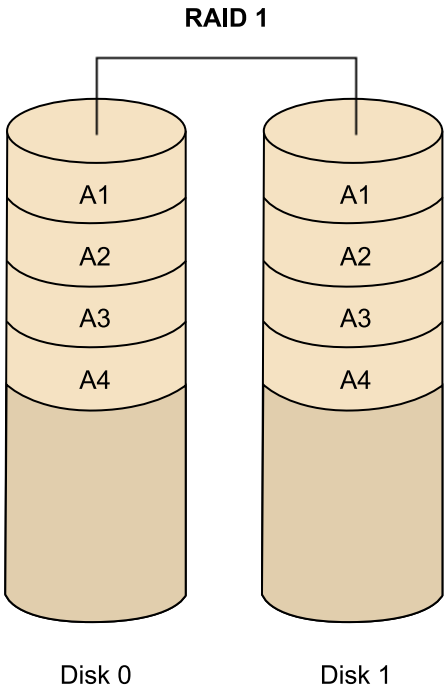


Figura 4

- **RAID 2**: en els anteriors es divideixen les dades en blocs per repartir. Aquí es divideix en bits i s'utilitzen codis de redundància per a la correcció de dades. No és utilitzat pràcticament, malgrat les altes prestacions que assoliria, ja que necessita idealment un nombre molt alt de discos, un per bit de dades i diversos per al càlcul de la redundància (per exemple, en un sistema de 32 bits, arribaria a usar 39 discos).

- **RAID 3**: utilitza divisió en bytes amb un disc dedicat a la paritat dels blocs. Tampoc no és gaire utilitzada, ja que segons la mida de les dades i posicions no permet accessos simultanis. RAID 4 és semblant, encara que divideix en l'àmbit de blocs en lloc de bytes, la qual cosa permet que sí que es puguin servir peticions simultànies quan se sol·licita un únic bloc.

- **RAID 5**: s'usa divisió en l'àmbit de blocs, distribuint la paritat entre els discos. Té un ús ampli, a causa de l'esquema senzill de paritat, i al fet que aquest càlcul s'implementa de manera senzilla per maquinari, amb bones prestacions.

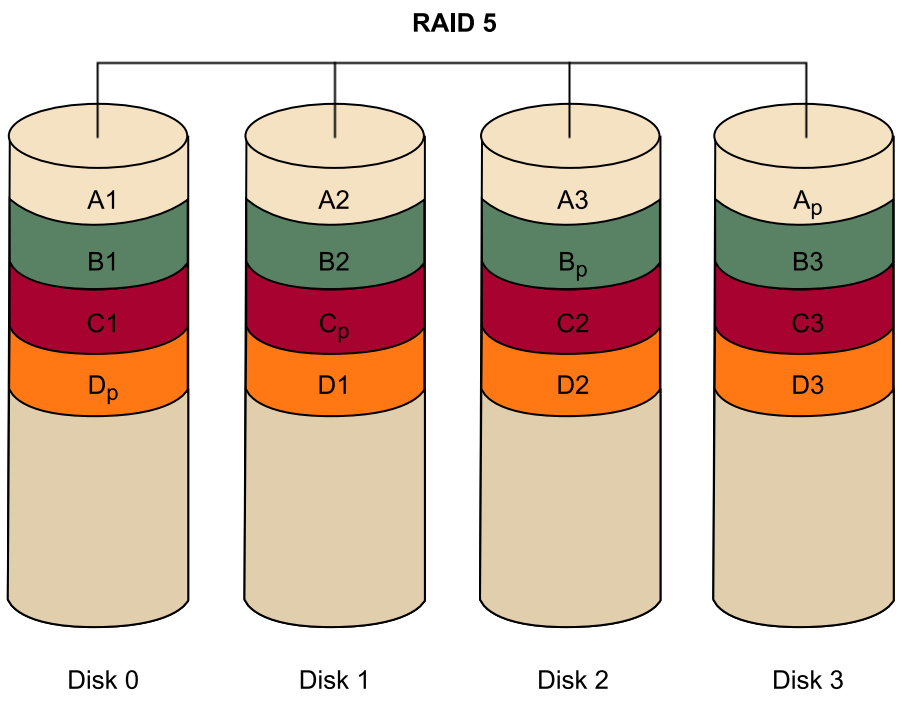


Figura 5

- **RAID 0 + 1 (o 01)**: un *mirror* de divisions és un nivell de RAID niat. S'implementen, per exemple, dos grups de RAID 0, els quals són usats en RAID 1 per a crear *mirror* entre ells. Un avantatge és que, en cas d'error, es pot reconstruir el nivell de RAID 0 usat gràcies a l'altra còpia, però si es volen afegir discos cal afegir-los a tots els grups de RAID 0 de la mateixa manera.

- **RAID 10 (1 + 0)**: divisió de *mirrors*, grups de RAID 1 sota RAID 0. Així, en cada grup de RAID 1 pot arribar a fallar un disc sense que es perdin dades. És clar que això obliga a reemplaçar-los, ja que si no el disc que quedi al grup es converteix en possible punt d'error de tot el sistema. És una configuració que se sol usar per a base de dades d'altes prestacions (per la tolerància a errors i la velocitat, en no estar basada en càlculs de paritat).

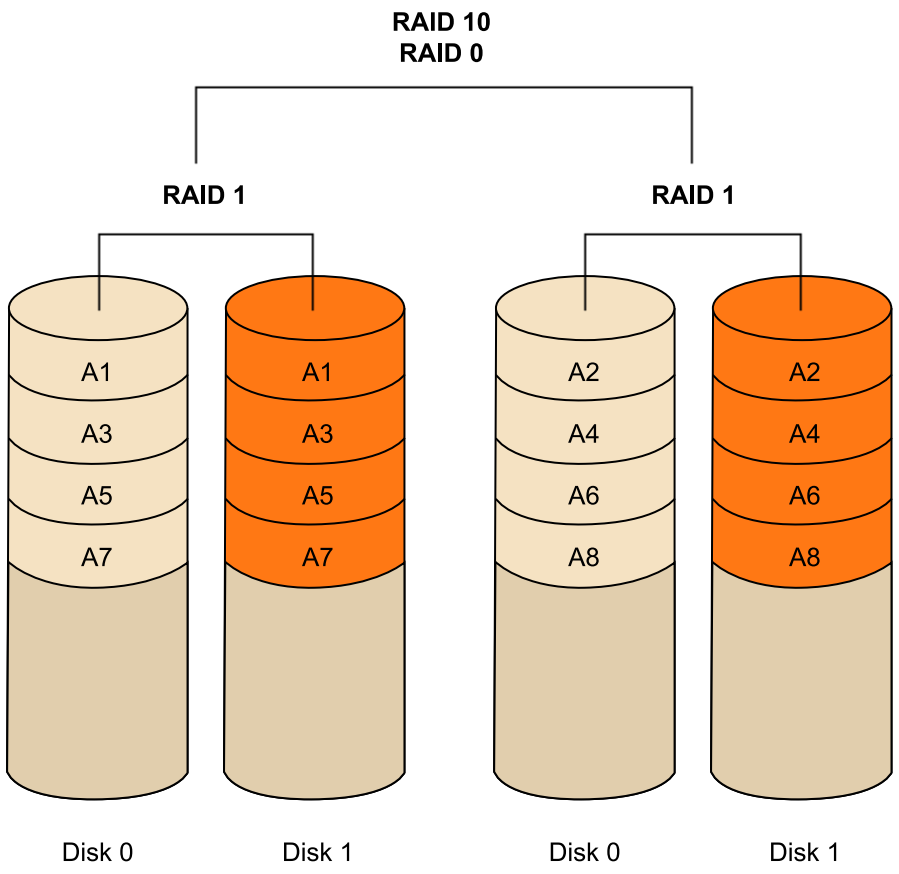


Figura 6

Algunes consideracions per tenir en compte sobre RAID en general:

- RAID millora l'*uptime* del sistema, ja que alguns dels nivells permeten que els discos fallin i el sistema continuï essent consistent, i depenent del maquinari, fins i tot es pot canviar el maquinari problemàtic en calent sense necessitat de parar el sistema, qüestió especialment important en sistemes crítics.

- RAID pot millorar el rendiment de les aplicacions; en especial, en els sistemes amb implementacions de *mirror* és possible que la divisió de dades permeti que les operacions lineals de lectura s'incrementin significativament, a causa de la possibilitat que els discos ofereixin, de manera simultània, parts d'aquesta lectura, amb l'augment de la taxa de transferència de dades.

- RAID no protegeix les dades; evidentment, la destrucció per altres mitjans (virus, mal funcionament general o desastres naturals) no està protegida. Ens hem de basar en esquemes de còpies de seguretat. Tinguem en compte que alguns esquemes protegeixen contra un o dos errors de discos de la matriu, però si n'hi ha més, o depenent de l'esquema, directament (el cas del RAID 0) es perdran dades.

- No se simplifica la recuperació de dades; si un disc pertany a una matriu RAID, s'ha d'intentar recuperar en aquest ambient. Es necessita programari específic o els controladors de maquinari per a accedir a les dades.

- Al contrari, no sol millorar aplicacions típiques d'usuari –d'escriptori– a causa que aquestes aplicacions tenen components alts d'accés aleatori a dades, o a conjunts de dades petites; pot ser que no es beneficiïn de lectures lineals o de transferències de dades sostingudes. En aquests ambients, és possible que no es noti a penes millora de prestacions.

- Alguns esquemes augmenten de velocitat les operacions de lectura, però d'altra banda penalitzen les d'escriptura (el cas del RAID 5 per al càlcul de paritat que cal escriure). Si l'ús és bàsicament d'escriptura, caldrà buscar quins esquemes no penalitzen o aconsegueixen la proporció d'escriptura que necessitem (alguns casos, com RAID 0,1, o algunes modalitats de RAID 10, són equivalents a escriure en un disc únic, o fins i tot augmenten les prestacions en aquest sentit).

- No es facilita el trasllat d'informació; sense RAID és bastant fàcil traslladar dades, simplement movent el disc d'un sistema a un altre. En el cas de RAID és gairebé impossible (tret que disposem del mateix maquinari controlador) moure una matriu de discos a un altre sistema.

En el cas de GNU/Linux, es dóna suport al maquinari RAID mitjançant diversos mòduls del nucli, associats a diferents conjunts de fabricants o circuits base, joc de xips, d'aquestes controladores RAID. Es permet així al sistema abstreure's dels mecanismes de maquinari i fer-los transparents al sistema i a l'usuari final. Així, aquests mòduls del nucli ens permeten l'accés als detalls d'aquestes controladores, i a la configuració de paràmetres de nivell molt baix, que en alguns casos (especialment en servidors que suporten càrrega elevada d'E/S), poden ser interessants per a processos de *tunning* del sistema de discos que usi el servidor. Es busca maximitzar les prestacions del sistema.

L'altra possibilitat que analitzarem aquí és la realització d'aquests processos mitjançant components de programari, concretament el component programari RAID de GNU/Linux.

GNU/Linux disposa en el nucli del controlador anomenat *multiple device* (*md*), que podem considerar com el suport del nucli per a RAID. Mitjançant aquest controlador podem implementar nivells de RAID, generalment 0, 1, 4, 5, 6 i niats (per exemple, RAID 10) sobre diferents dispositius de bloc com discos IDE, SATA o SCSI. També disposa del nivell *linear*, com a nivell en què es produeix una combinació lineal dels discos disponibles (és igual que siguin de diferents mides), de manera que s'escriu consecutivament en els discos.

Per a la utilització del RAID programari en Linux, hem de disposar del suport RAID en el nucli, i en el seu cas els mòduls *md* actius (a més d'alguns controladors específics segons el nivell; vegeu els controladors disponibles associats a RAID, per exemple en Debian amb *modconf*). El mètode preferit per a la implementació de matrius de discos RAID, mitjançant el programari RAID ofert per Linux, és mitjançant el procés d'instal·lació del sistema inicial, o bé mitjançant la utilitat *mdadm*. Aquesta utilitat ens permet crear les matrius i gestionar-les.

Webs recomanats

Per a consultar conceptes de programari RAID, vegeu el *Linux RAID wiki*:

https://raid.wiki.kernel.org/index.php/Linux_Raid

http://www.linuxfoundation.org/collaborate/workgroups/linux-raid

Vegeu també els nivells RAID suportats:

https://raid.wiki.kernel.org/index.php/Introduction#The_RAID_levels

Observem-ne alguns casos pràctics. Suposem uns discos SCSI /dev/sda, /dev/sdb... en els quals disposem de diverses particions disponibles per a implementar RAID:

Creació d'una matriu *linear:*

```
# mdadm --create --verbose /dev/md0 --level=linear --raid-devices=2 /dev/sda1 /dev/sdb1
```

en què es genera una matriu *linear* a partir de les particions primeres de /dev/sda i /dev/sdb, creant el nou dispositiu /dev/md0, que ja pot ser usat com a nou disc (suposant que existeixi el punt de muntatge /mnt/discRAID):

```
# mkfs.ext2fs /dev/md0
# mount /dev/md0 /mnt/discRAID
```

Per a un RAID 0 o RAID 1 podem canviar simplement el nivell (*--level*) a *raid0* o *raid1*. Amb *mdadm --detail /dev/md0* podrem comprovar els paràmetres de la matriu nova creada.

També podem consultar l'entrada *mdstat* en /proc per a determinar les matrius actives, i també els seus paràmetres. En especial, en els casos amb *mirror* (per exemple, en els nivells 1, 5...) podrem observar en la seva creació la construcció inicial de la matriu de les còpies, en /proc/mdstat indicarà el nivell de reconstrucció (i el temps aproximat d'acabament).

mdadm disposa de moltes opcions que ens permeten examinar i gestionar les diferents matrius RAID de programari creades (en podem veure una descripció i exemples en *man mdadm*).

Una altra qüestió important és que les matrius RAID per programari no són automàticament reconegudes en l'arrencada (com sí que passa amb el RAID de maquinari suportat), ja que de fet depenen de la construcció amb *mdadm*. Perquè la definició d'una matriu de programari sigui persistent, cal registrar-la al fitxer de configuració /etc/mdadm.conf (la ubicació pot dependre de la distribució). Un pas senzill és crear-lo automàticament a partir del resultat de l'ordre.

```
mdadm --detail --scan (també es pot afegir --verbose)
```

Una altra consideració important són les optimitzacions a què es poden sotmetre les matrius RAID per a millorar-ne el rendiment, tant per a monitoritzar el comportament com per a optimitzar paràmetres del sistema de fitxers, per a fer un ús més efectiu dels nivells RAID i les seves característiques.

Nota

L'optimització de les matrius RAID pot ser una font important de sintonització del sistema. Examineu algunes qüestions en https://raid.wiki.kernel.org/index.php/Performance o en la pàgina *man* mateixa d'*mdadm*.

Passarem a detallar un altre cas de configuració RAID més elaborat basat en l'elaboració d'un RAID 5.

Aquest nivell RAID (com a mínim de tres discos) presenta una configuració distribuïda de dades i paritat en els discos que formen la matriu. Es presenta molt bon rendiment en lectura, però disminueix l'escriptura (respecte a un únic disc), ja que cal calcular la paritat i distribuir-la entre els discos (en casos RAID de maquinari, aquest cas de càlcul de paritat s'accelera mitjançant maquinari). Una altra dada que cal tenir en compte és que la capacitat final obtinguda és la suma de $N - 1$ discos, i aquesta configuració és resistent a un error de disc present en la matriu, el qual es pot reemplaçar i tornar a reconstruir la matriu. El nivell no suporta dos errors de disc (probabilitat que, de fet, pot augmentar si en la matriu RAID s'integra un nombre elevat de discos), amb la qual cosa, com ja comentem, disposar de RAID no ens inhibeix del procés de còpia de seguretat de dades. Si volguéssim disposar de més fiabilitat, ens podríem moure a RAID 6 (molt recomanable davant de RAID5 per qüestions de fiabilitat, sempre que es disposi d'un mínim de quatres discos), que té càlcul amb paritat dual, a costa de baixar una mica el rendiment, o a configuracions amb RAID 10, que té un bon compromís entre rendiment i fiabilitat.

Farem, en aquest cas, el procés de construcció amb quatre discos SATA d'alta capacitat 1,5 TB, amb un emmagatzemament final de 4,5 TB (aproximadament). La matriu RAID és complementària al sistema existent; en casos en què hagi d'incloure particions de *boot*, el procés és més delicat (vegeu les recomanacions del *RAID software wiki*, comentat anteriorment).

Utilitzarem quatre discos SATA presents en el sistema com a /dev/sdb, sdb, sdc, sdd, sde. El disc /dev/sda se suposa que inclou el sistema i les particions d'arrencada, i no forma part del RAID. Primer hem de passar per la inicialitzacio dels discos (aquest procés esborra qualsevol contingut previ); definirem una partició única (en aquest cas, integrarem tot l'emmagatzemament disponible; en altres esquemes es podrien fer diverses particions), creada amb *fdisk*.

Fonamentalment, seleccionem una nova partició primària, requisits per defecte i canviem el tipus de partició a *Linux Raid autodetect* (codi *fd*), fent el procés amb *fdisk*, si *x* és cada disc diferent de la matriu, i el parèntesi inclou les opcions mitjançant teclat de *fdisk*:

```
# fdisk /dev/sdx         (d n p 1 ENTER ENTER t fd w)
```

(per al nostre cas, 4 vegades amb *x=b,c,d,e*).

Respecte a aquesta inicialització, cal anar amb compte amb les mides de disc, ja que *fdisk* amb particions de tipus MBR només suporta particions fins a 2 TB. Si els discos fossin majors ens hauríem de moure a altres eines de particionament, com *gparted/parted,* que suporta nous tipus de particions com GPT, que ja no disposa d'aquestes restriccions de mida.

Una vegada feta aquesta inicialització, ja podem passar a construir la matriu:

```
# mdadm --create /dev/md0 --level=6 --verbose --force --chunk=512
--raid-devices=4 /dev/sdb1 /dev/sdc1 /dev/sdd1 /dev/sde1
```

Un paràmetre important en RAID més avançats és la mida del paràmetre *chunk* (per defecte, 64 unitats en kB), que té especial rellevància en les prestacions del RAID. Hi ha estudis que determinen mides adequades de *chunk* en funció de la funcionalitat final del RAID (escriptura o lectura, mida mitjana dels accessos, accés principalment aleatori o seqüencial, etc.). En general, en RAID 6 es recomanen mides de 128 o més, i fins i tot més elevades, de 512-1024, depenent de la mida mitjana dels arxius. Si disposem de més discos que els mínims necessaris, també podem afegir discos com a *spare*, en l'ordre anterior per exemple amb `--spare-devices=1`. Els discos de suport tipus *spare* no formen part de l'*array* RAID inicial, però si un dels discos actius falla, quan la fallada és detectada, el disc és marcat com a *bad*, es treu de l'*array* i la reconstrucció de l'array comença immediatament en el primer disc *spare* disponible.

Una vegada llançada la instrucció anterior, començarà la construcció del RAID en segon pla. Podem anar consultant /proc/mdstat, que ens esmentarà la velocitat de construcció i el percentatge, i també una estimació de temps per a finalitzar. Els temps de construcció són bastant grans per a altes capacitats d'espai, i poden anar des d'algunes hores fins a dies, depenent de les capacitats dels discos i la màquina.

Podríem estar veient en /proc/mdstat:

```
# cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4]
md0 : active raid6 sdb1[0] sde1[3] sdd1[2] sdc1[1]
   4395406848 blocks level 6, 512k chunk, algorithm 2 [4/3] [UUU_]
  [==>..................] recovery = 12.6% (37043392/292945152)
```

```
   finish=127.5min speed=33440K/sec

unused devices: <none>
```

I una vegada finalitzat:

```
# cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4]
md0 : active raid6 sdb1[0] sde1[3] sdd1[2] sdc1[1]
   4395406848 blocks level 6, 512k chunk, algorithm 2 [4/4] [UUUU]

unused devices: <none>
```

El pas següent és crear un *filesystem* en el nostre recentment creat *array*, vist pel sistema com el dispositiu /dev/md0 (aquest pas es podria iniciar simultàniament durant la reconstrucció, però per velocitat i seguretat en alguns passos crítics inicials es recomana fer-ho al final del procés).

En el nostre cas, optimitzarem lleugerament aquesta creació examinant alguns paràmetres; crearem un *filesystem ext3* en la matriu completa (és recomanable usar un gestor de volums primer, com LVM, que veurem a continuació, perquè ens permetrà crear diverses particions lògiques, que es puguin expandir en el futur o contreure's segons les necessitats).

Per a la creació del *filesystem* hi ha algunes recomanacions generals que provenen a partir de la mida de *chunk* (nombre de dades consecutives que resideix en un disc en kB), que de fet ens defineix com s'accedeix a la matriu, i dels discos presents, un possible esquema de creació d'un *filesystem ext3*:

```
mkfs.ext3 -v -b 4096 -E stride=128,stripe-width=384 /dev/md0
```

en què *-b* 4096 és la mida de bloc de disc, recomanada per a *filesystems* molt grans, i *-E* defineix opcions del *filesystem* per a optimitzar-ne el rendiment. Aquestes opcions es poden variar després amb la instrucció *tune2fs*.

En aquestes opcions se sol suggerir un càlcul relacionat amb el *chunk*, que pot ser en el nostre exemple:

- *chunk size* = 512 kB (col·locat en la instrucció *mdadm* en la creació de la matriu)
- *block size* = 4 kB (recomanat per a grans *filesystems*)
- *stride* = *chunk / block* = 512 kB / 4k = 128 kB
- *stripe-width* = *stride* * ((*n* discos en el raid 6) – 2) = 128 kB * ((4) – 2) = 128 kB * 2 = 256 kB

Això ens permet ajustar els paràmetres del sistema de fitxers a l'ús del *chunk* escollit.

Una vegada creat el sistema de fitxers, ja tenim la matriu disponible perquè pugui ser muntada en una ubicació del sistema (suposant un directori previ */mnt/raid6*):

```
# mount -t ext3 /mnt/raid6 /dev/md0
```

I ja el tenim disponible en el sistema. Si el volem fer fix, recordeu introduir els paràmetres en /etc/fstab perquè es munti en l'arrencada. I en aquest cas de matriu de programari és recomanable (encara que alguns nuclis recents ja suporten autodetecció de RAID de programari), col·locar la informació RAID en el fitxer /etc/mdadm.conf, mitjançant la consulta de l'identificador de la matriu amb les instruccions *# mdadm -detail -scan* (amb *-verbose* si es necessita la identificació de dispositius):

```
#mdadm --detail --scan --verbose
ARRAY    /dev/md0    level=raid6    num-devices=4    metadata=0.90
UUID=dcc77e17:94093185:66731ad6:6353ec0b
   devices=/dev/sdb1,/dev/sdc1,/dev/sdd1,/dev/sde1
```

En el reinici següent del sistema ja disposarem de la matriu muntada en arrencada. També, com a punt final, podem obtenir la informació de la matriu amb *mdadm -detail*:

```
# mdadm --detail /dev/md0
/dev/md0:
     Version : 0.90
 Creation Time : Sat May 8 09:33:06 2010
Raid Level : raid6
   Array Size : 4395406848 (4191.79 GiB 4500.90 GB)
 Used Dev Size : 1465135616 (1397.26 GiB 1500.30 GB)
 Raid Devices : 4
 Total Devices : 4
Preferred Minor : 0
  Persistence : Superblock is persistent

  Update Time : Fri May 21 12:14:31 2010
     State : clean
 Active Devices : 4
Working Devices : 4
 Failed Devices : 0
  Spare Devices : 0

      Layout : left-symmetric
    Chunk Size : 512K
```

```
     UUID : dcc77e17:94093185:66731ad6:6353ec0b (local to host kaoscore)
    Events : 0.125

Number  Major  Minor  RaidDevice  State
 0       8      17          0    active sync  /dev/sdb1
 1       8      33          1    active sync  /dev/sdc1
 2       8      49          2    active sync  /dev/sdd1
 3       8      65          3    active sync  /dev/sde1
```

Respecte al funcionament del sistema, cal examinar tant aquest informe com el proporcionat per /proc/mdstat per a controlar de manera periòdica (manualment, mitjançant *cron*, o per notificacions mitjançant el correu) l'estat del RAID (en aquest cas actiu), per a determinar si es produeix algun error de disc. En aquest estat la matriu passaria a *degraded,* i el sistema perdria la seva capacitat de tolerància a un error següent. És llavors necessari detectar quin disc està fallant i substituir-lo, si no es tenen discos *spare* en l'*array*, i en aquest cas la reconstrucció començaria automàticament. Si no es disposa de *spare*, examinem quin disc és, per a substituir-lo, perquè comenci la reconstrucció de la matriu (mitjançant *mdadm* es pot eliminar un disc de la matriu, i afegir una vegada fet el canvi de maquinari el nou disc). També és possible mitjançant *mdadm* simular degradacions o errors, la qual cosa ens permet provar la nostra matriu davant de condicions extremes.

Finalment, destaquem que la creació de matrius RAID és molt interessant per a la realització de grans suports de discos per a entorns empresarials en què s'intenta maximitzar la tolerància a errors, la recuperació ràpida davant de problemes i un suport continu de serveis sense interrupcions. És un camp també en què es necessita un entorn acurat d'anàlisi de rendiment de les solucions, dels requisits inicials del servei i experimentació amb les diferents solucions possibles.

8.2. Volums lògics (LVM)

En un moment determinat, sorgeix la necessitat d'abstreure's del sistema físic de discos, i de la seva configuració i nombre de dispositius, que el sistema (operatiu) s'encarregui d'aquesta feina i no ens hàgim de preocupar d'aquests paràmetres directament. En aquest sentit, es pot veure el sistema de volums lògics com una capa de virtualització de l'emmagatzemament que permet una visió més simple que faciliti la utilització fluida i senzilla.

En el nucli Linux es disposa d'LVM (*logical volume manager*) [War13], que es va basar en idees desenvolupades de gestors de volums d'emmagatzemament usats en HP-UX (una versió UNIX propietat d'HP). Actualment n'hi ha dues versions, de les quals LVM2 és la més utilitzada, per una sèrie de prestacions afegides que incorpora.

L'arquitectura d'una LVM consisteix típicament en els components (principals):

1) Volums físics (PV): són els discos durs, o particions d'aquests discos, o qualsevol altre element que aparegui com un disc dur amb vista al sistema (per exemple, un RAID per programari o maquinari).

2) Volums lògics (LV): és l'equivalent a la partició del disc físic. Aquesta LV és visible en el sistema com un dispositiu de blocs (absolutament equivalent a una partició física), i pot contenir un sistema de fitxers (per exemple, el /home dels usuaris). Els volums tenen més sentit per als administradors, ja que es poden usar noms per a identificar-los (així podem utilitzar un dispositiu lògic, anomenat *stock* o *marketing* en lloc d'*sda6* o *sdc3*).

3) Grups de volums (VG): és l'element de la capa superior. La unitat administrativa que engloba els nostres recursos, ja siguin volums lògics (LV) o físics (PV). En aquesta unitat es desen les dades dels PV disponibles, i com es formen les LV a partir dels PV. Evidentment, per a poder utilitzar un grup VG, hem de disposar de suports físics PV, que s'organitzen en diferents unitats lògiques LV.

A la figura següent observem un grup de volums, en què disposem de set PV, en forma de particions de discos, que s'han agrupat per a formar dos volums lògics (que s'han acabat utilitzant per a formar els sistemes de fitxers /usr i /home):

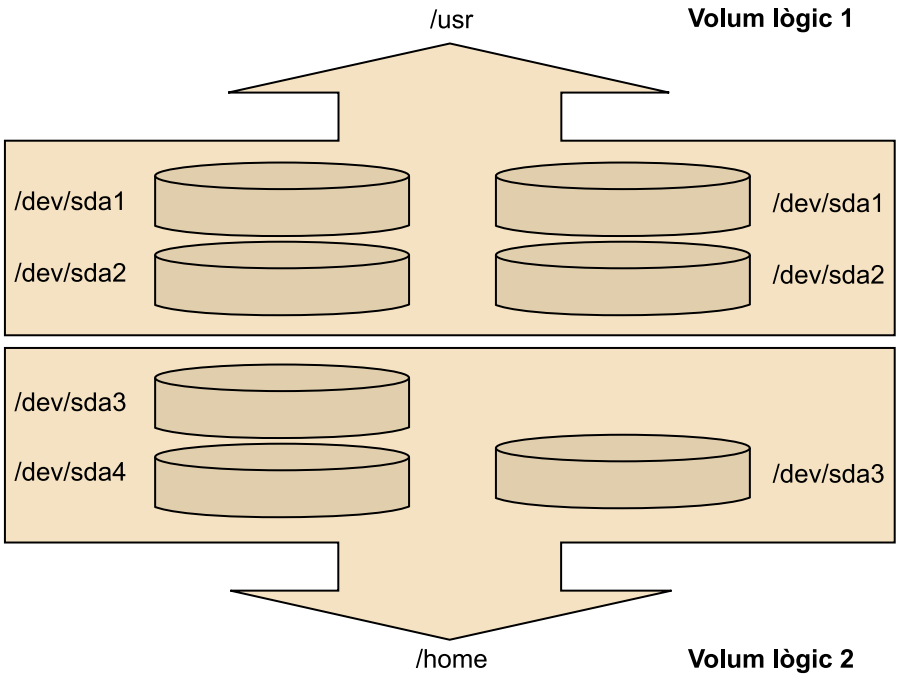


Figura 7. Esquema d'un exemple d'LVM.

Amb l'ús dels volums lògics permetem un tractament més flexible de l'espai en el sistema d'emmagatzematge (que podria tenir un gran nombre de discos i particions diferents), segons les necessitats que ens apareguin. Podem gestionar l'espai, tant amb identificadors més adequats com amb operacions que ens permetin adequar les necessitats a l'espai disponible en cada moment.

Els sistemes de gestió de volums ens permeten:

1) Redimensionar dinàmicament grups i volums lògics, aprofitant nous PV, o extraient-ne alguns dels disponibles inicialment.

2) Instantànies (*snapshots*) del sistema d'arxius (lectura en LVM1 i lectura o escriptura en LVM2). Això facilita la creació d'un nou dispositiu que sigui una instantània en el temps de la situació d'una LV. Es pot crear, per exemple, la instantània, muntar-la, provar diverses operacions o una configuració nova de programari, o altres elements, i si no funciona com esperàvem, tornar el volum original al seu estat abans de les proves.

3) RAID 0 de volums lògics.

En LVM no s'implementen configuracions de RAID de tipus 1 o 5, si són necessàries (és a dir, redundància i tolerància a error). El procés és utilitzar programari de RAID o controladora de maquinari RAID que l'implementi en un determinat nivell de RAID, i després col·locar LVM com a capa superior al RAID creat prèviament.

Fem un breu exemple de creació típica (en molts casos, l'instal·lador de la distribució fa un procés semblant, si permetem un LVM com a sistema inicial d'emmagatzemament). Bàsicament, es fa:

a) la creació dels volums físics (PV),

b) creació del grup lògic (VG),

c) creació del volum lògic (LV), i

d) utilització per a creació d'un sistema de fitxers, i muntatge posterior:

1) Disposem, per exemple, de tres particions de diferents discos. Creem tres PV (alerta: els passos inicials esborren qualsevol contingut dels discos) i n'inicialitzem el contingut:

```
# dd if=/dev/zero of=/dev/sda1 bs=1k count=1
# dd if=/dev/zero of=/dev/sda2 bs=1k count=1
# dd if=/dev/zero of=/dev/sdb1 bs=1k count=1

# pvcreate /dev/sda1
```

```
Physical volume "/dev/sda1" successfully created
# pvcreate /dev/sda2
Physical volume "/dev/sda2" successfully created
# pvcreate /dev/sdb1

Physical volume "/dev/sdb1" successfully created
```

2) Col·loquem en un VG creat dels diferents PV:

```
# vgcreate grup_discos /dev/sda1 /dev/sda2 /dev/sdb1

Volume group "grup_discos" successfully created
```

3) Creem l'LV (en aquest cas, de mida 1 GB) a partir dels elements que tenim en el grup VG (*-n* indica el nom del volum):

```
# lvcreate -L1G -n volum_logic grup_discos
lvcreate -- doing automatic backup of "grup_discos"
lvcreate -- logical volume "/dev/grup_discos/volum_logic" successfully created
```

4) I finalment, creem un sistema de fitxers (un xfs en aquest cas):

```
# mkfs.xfs /dev/grup_discos/volum_logic
```

Que, per exemple, podríem col·locar d'espai per a còpies de seguretat:

```
mkdir /mnt/backup
mount -t xfs /dev/grup_discos/volum_logic /mnt/backup
```

I disposem finalment del dispositiu com un volum lògic que implementa un sistema de fitxers.

Posem un altre exemple d'escenari d'LVM, en què canviem dinàmicament els esquemes de disc usats per un sistema, perquè no sabem quina imatge final tindrà el sistema al llarg de la seva història.

Suposem, per exemple, que hi ha un entorn de portàtil en què de partida tenim un esquema de particions:

1) C:
2) D: (incloent dades)
3) Partició invisible de recuperació del sistema.

Analitzant, una possible instal·lació de GNU/Linux, determinem que la millor solució és usar la partició 2 havent desat prèviament les dades. Però GNU/Linux sol necessitar com a mínim dues particions (/ root i swap) i, si el volguéssim alterar amb aquest esquema, hauríem de definir noves particions, posar-ne alguna com a estesa i moure la resta o copiar particions.

En aquest esquema, per evitar aquests problemes, decidim generar volums sobre la partició física 2, que ens permetran col·locar a posteriori les particions de root i swap associades als volums corresponents.

Amb això, el curs de l'acció seria:

1) Generem un PV amb la /dev/sda2.
2) Generem un VG "grup" amb la PV anterior.
3) Generem dos volums a partir del grup anterior: LV "system" i LV "swap".

La seqüència de tecles d'ordre podria ser:

```
# pvcreate /dev/sda2
# vgcreate grupo /dev/sda2
# lvcreate -n system -L <grandària_system>
# lvcreate -n swap -L <grandària_swap>
```

Amb això, ara /dev/sda2 se'ns presentarà com dos volums:

/dev/grup/system
/dev/grup/swap

Que podem usar en una instal·lació de GNU/Linux per a fer les particions *root* i *swap* (sempre que la distribució suporti instal·lació sobre volums LVM). Amb la qual cosa, amb aquestes particions, una vegada formatades, i sistemes de fitxers de *root* (possiblement un ext4) i *swap*, podríem instal·lar un sistema GNU/Linux que residirà sobre els dos volums que al seu torn estan sobre la partició física /dev/sda2.

Ara suposem que ens canvia la imatge del sistema final, per exemple al llarg del temps ens adonem que voldríem utilitzar el disc complet per a la distribució GNU/Linux que vam instal·lar en el passat, amb la qual cosa podríem recuperar les particions /dev/sda1 i sda3.

Amb l'LVM, això ho podem plantejar amb:

1) Nous volums físics PV amb /dev/sda1 i /dev/sda3.

2) Estendre el grup VG amb els nous PV.

3) Estendre els LV amb el nou espai disponible, i els sistemes de fitxers existents.

Bàsicament:

```
# pvcreate /dev/sda1
# pvcreate /dev/sda3
# vgextend group /dev/sda1
# vgextend group /dev/sda3
```

Abans de desmuntar els sistemes de fitxers, per a les redimensions, i utilitzar un liveCD o USB, podríem en el sistema:

```
# lvextend /dev/grupo/system +<grandària_sda1>G
# lvextend /dev/grupo/system +<grandària_sda3>G
```

Afegint així tota la grandària disponible a l'LV "system", encara que no caldria afegir-ho tot, podríem deixar alguna cosa, preveient nous volums LV necessaris en el futur, etc.

I finalment redimensionar el sistema de fitxers (suposant que sigui de tipus ext4).

```
# resize2fs /dev/grupo/system <grandària total>
```

En tot cas, com a mesura de precaució, abans d'utilitzar aquest tipus de procediments seria recomanable fer una còpia de seguretat de les dades sensibles o perdudes, o directament dels sistemes de fitxers sencers o el mateix disc.

En el següent cas d'ús amb volums, en una distribució Fedora, combinarem LVM2 amb XFS per realitzar operacions de creació de volums, expansió de sistemes d'arxius, i ús de *snapshots* de LVM2.

Nota

Igual que en altres casos, les operacions d'aquest cas destrueixen les dades anteriors (en el nostre cas farem servir el disc */dev/sdb*).

Comencem examinant els dispositius de bloc disponibles:

```
# lsblk
NAME       MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda          8:0    0   20G  0 disk
├─sda1       8:1    0  500M  0 part /boot
└─sda2       8:2    0 19,5G  0 part
  ├─fedora-root
  │        253:0    0 17,5G  0 lvm  /
  └─fedora-swap
           253:1    0    2G  0 lvm  [SWAP]
sdb          8:16   0    8G  0 disk
```

El disc */dev/sdb* no està en ús: crearem una partició, amb tot l'espai, on esborrarem les dades.

```
# fdisk /dev/sdb

Bienvenido a fdisk (util-linux 2.28).
Los cambios solo permanecerán en la memoria, hasta que decida escribirlos.
Tenga cuidado antes de utilizar la orden de escritura.



Orden (m para obtener ayuda): n
Tipo de partición
   p   primaria (0 primaria(s), 0 extendida(s), 4 libre(s))
   e   extendida (contenedor para particiones lógicas)
Seleccionar (valor predeterminado p): p
Número de partición (1-4, valor predeterminado 1): 1
Primer sector (2048-16777215, valor predeterminado 2048):
Último sector, +sectores o +tamaño{K,M,G,T,P} (2048-16777215, valor predeterminado 16777215):

Crea una nueva partición 1 de tipo 'Linux' y de tamaño 8 GiB.

Orden (m para obtener ayuda): w
Se ha modificado la tabla de particiones.
Llamando a ioctl() para volver a leer la tabla de particiones.
Se están sincronizando los discos.

# dd if=/dev/zero of=/dev/sdb1 bs=1k count=1
1+0 registros leídos
1+0 registros escritos
1024 bytes (1,0 kB, 1,0 KiB) copied, 0,00265721 s, 385 kB/s
```

Passem a crear un dispositiu físic, un grup de volums *volgroup* i un volum dins d'ell amb el 50% d'espai disponible (4 GB en el nostre cas):

```
# pvcreate /dev/sdb1
  Physical volume "/dev/sdb1" successfully created.

# vgcreate volgroup /dev/sdb1
  Volume group "volgroup" successfully created

# lvcreate -L4G -n vol volgroup
  Logical volume "vol" created.
```

Creem un sistema de fitxers XFS sobre el volum i el muntem a */mnt/extra*.

```
# mkfs.xfs /dev/volgroup/vol
meta-data=/dev/volgroup/vol      isize=512    agcount=4, agsize=262144 blks
```

```
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=0
data     =                       bsize=4096   blocks=1048576, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=1
log      =internal log           bsize=4096   blocks=2560, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0

# mkdir /mnt/extra

# mount -t xfs /dev/volgroup/vol /mnt/extra

# df -k
S.ficheros               bloques de 1K  Usados Disponibles Uso% Montado en
/dev/mapper/volgroup-vol       4184064   32960     4151104   1% /mnt/extra
```

Mentrestant, podem realitzar algunes operacions amb el nostre sistema de fitxers, com copiar arxius temporals/de prova, per disposar d'alguns arxius en el sistema de fitxers.

Procedim a crear un volum *snapshot* de LVM, amb 1 GB assignat (Compte!, per evitar problemes, la grandària hauria de ser la mateixa que la de *filesystem* original. En aquest cas, estem pressuposant que, en aquests moments, un 25% del sistema original no arriba a estar ocupat. Si el sistema de fitxers origen estigués ple, hauríem de crear un *snapshot* també de 4 GB):

```
# lsblk /dev/sdb
NAME             MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sdb                8:16   0    8G  0 disk
└sdb1              8:17   0    8G  0 part
  └volgroup-vol 253:2    0    4G  0 lvm  /mnt/extra

# lvcreate --size 1G -s -n vol_snap /dev/volgroup/vol
  Logical volume "vol_snap" created.

# lvdisplay
  --- Logical volume ---
  LV Path                /dev/volgroup/vol
  LV Name                vol
  VG Name                volgroup
  LV UUID                yJiHYl-onqI-QR6M-NK85-0Dr1-2Qre-Eu1LtB
  LV Write Access        read/write
  LV Creation host, time localhost.localdomain, 2016-07-11 13:18:09 +0200
  LV snapshot status     source of
                         vol_snap [active]
  LV Status              available
```

```
  # open                 1
  LV Size                4,00 GiB
  Current LE             1024
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           253:2

  --- Logical volume ---
  LV Path                /dev/volgroup/vol_snap
  LV Name                vol_snap
  VG Name                volgroup
  LV UUID                LCZqyU-aWb3-mwvV-8yzx-cS4t-KdAH-Jl31wy
  LV Write Access        read/write
  LV Creation host, time localhost.localdomain, 2016-07-11 13:26:01 +0200
  LV snapshot status     active destination for vol
  LV Status              available
  # open                 0
  LV Size                4,00 GiB
  Current LE             1024
  COW-table size         1,00 GiB
  COW-table LE           256
  Allocated to snapshot  0,00%
  Snapshot chunk size    4,00 KiB
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           253:5


# lsblk
NAME                    MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sdb                       8:16   0    8G  0 disk
└sdb1                     8:17   0    8G  0 part
  ├volgroup-vol-real   253:3    0    4G  0 lvm
  | ├volgroup-vol      253:2    0    4G  0 lvm  /mnt/extra
  | └volgroup-vol_snap 253:5    0    4G  0 lvm
  └volgroup-vol_snap-cow
                       253:4    0    1G  0 lvm
    └volgroup-vol_snap 253:5    0    4G  0 lvm
```

Generem un punt de muntatge i muntem l'*snap*. En aquest cas, pot incorporar-se una opció de tan sols lectura *-ro*, si l'objectiu és fer només una còpia de seguretat. L'opció *nouuid* és necessària, en cas contrari, *mount* no deixaria muntar dos sistemes d'arxius XFS que tenen el mateix *uuid*. Una alternativa és

realitzar l'ordre `xfs_admin -O generate /dev/volgroup/vol_snap` per generar un nou *uuid*, però és recomanable utilitzar la primera opció per evitar modificar el volum de *snapshot*.

```
# mkdir /mnt/snap
# mount -o nouuid /dev/volgroup/vol_snap /mnt/snap

# lvs
  LV       VG       Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  vol      volgroup owi-aos---  4,00g
  vol_snap volgroup swi-aos---  1,00g      vol    0,20

# lsblk
NAME                    MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sdb                       8:16   0    8G  0 disk
└sdb1                     8:17   0    8G  0 part
  ├volgroup-vol-real   253:3    0    4G  0 lvm
  │ ├volgroup-vol      253:2    0    4G  0 lvm  /mnt/extra
  │ └volgroup-vol_snap 253:5    0    4G  0 lvm  /mnt/snap
  └volgroup-vol_snap-cow
                       253:4    0    1G  0 lvm
    └volgroup-vol_snap 253:5    0    4G  0 lvm  /mnt/snap
```

Ara podem realitzar una còpia de seguretat de la nostra partició aprofitant l'*snapshot* creat (assumim que la partició on hi ha */tmp* té espai suficient per a aquesta còpia de seguretat):

```
# tar -cvzf /tmp/backup_vol.tgz /mnt/snap
```

Quan la còpia de seguretat s'hagi realitzat, podem desmuntar l'*snapshot* i treure'l del sistema; d'aquesta manera, evitarem la pèrdua de prestacions per mantenir alhora actius la còpia original i l'*snapshot*.

```
# umount /mnt/snap
# lvremove /dev/volgroup/vol_snap
Do you really want to remove active logical volume vol_snap? [y/n]: y
  Logical volume "vol_snap" successfully removed
```

Ara realitzem una extensió del volum *vol* per augmentar en 2G la seva grandària:

```
# lvextend -L+2G /dev/volgroup/vol
Size of logical volume volgroup/vol changed from 4,00 GiB (1024 extents) to 6,00 GiB (1536 extents).
  Logical volume vol successfully resized.
```

Ara podem estendre el sistema d'arxius existent (en el nostre cas XFS) amb:

```
# xfs_growfs /mnt/extra

# lsblk /dev/sdb
NAME             MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sdb                8:16   0    8G  0 disk
└─sdb1             8:17   0    8G  0 part
  └─volgroup-vol 253:2    0    6G  0 lvm  /mnt/extra

# df -hT /mnt/extra
S.ficheros               Tipo Tamaño Usados  Disp Uso% Montado en
/dev/mapper/volgroup-vol xfs    6,0G    33M  6,0G   1% /mnt/extra
```

9. Programari: actualització

Per a l'administració de la instal·lació o actualització de programari en el nostre sistema, dependrem en primera instància del tipus de paquets de programari que utilitzi el nostre sistema:

Vegeu també

En l'apartat 1, "Eines bàsiques per a l'administrador," desenvolupem en profunditat aquests conceptes.

- **RPM**: paquets que utilitza la distribució Fedora/Red Hat (i derivades). Se solen manejar mitjançant la instrucció *rpm*. Contenen informació de dependències del programari. A alt nivell, la utilitat principal d'actualització és YUM.

- **DEB**: paquets de Debian, se solen manejar amb un conjunt d'eines que treballen a diferents nivells amb paquets individuals o grups. Entre aquestes eines, es poden esmentar *dselect, tasksel, dpkg, apt-get* i *aptitude*.

- **Tar**, o bé els *tgz* (també *tar.gz*): són purament paquets de fitxers units i comprimits mitjançant instruccions estàndard com *tar* i *gzip* (s'usen els mateixos per a la descompressió). Aquests paquets no contenen informació de dependències i es poden instal·lar en diferents llocs, si no és que porten informació de ruta (*path*) absoluta.

Per a manejar aquests paquets hi ha diverses eines gràfiques, com, per a RPM, Kpackage; per a DEB, Synaptic o Gnome-apt; per a *tgz*, Kpackage o des del gestor de fitxers gràfics mateix (en el Gnome o el KDE). També hi sol haver utilitats de conversió de paquets. Per exemple, en Debian tenim la instrucció *alien,* que permet convertir paquets RPM a DEB. Encara que cal prendre les precaucions oportunes perquè el paquet, per tenir una distribució de destinació diferent, no modifiqui algun comportament o fitxer de sistema no esperat.

Depenent de l'ús dels tipus de paquets o eines, l'actualització o instal·lació de programari del nostre sistema es podrà produir de diferents maneres:

1) Des dels CD mateixos d'instal·lació del sistema. Totes les distribucions busquen el programari en els seus CD. Però cal tenir en compte que aquest programari no sigui antic i no inclogui, per aquesta raó, alguns pegats com a actualitzacions, o noves versions amb més prestacions, amb la qual cosa, si s'instal·la a partir de CD, és bastant comú verificar després que no hi hagi alguna versió més recent disponible en els repositoris oficials de la distribució o en repositoris alternatius que proporcionin programari addicional (sempre amb les mesures de seguretat oportunes per a establir que siguin repositoris de confiança).

2) Mitjançant serveis d'actualització o cerca de programari, ja sigui de manera gratuïta, com el cas de l'eina *apt-get* de Debian, o yum/dnf a Fedora, o serveis de subscripció (de pagament o amb facilitats bàsiques), com el Red Hat Network de les versions Red Hat comercials.

3) Amb repositoris de programari que ofereixen paquets de programari preconstruïts per a una distribució determinada.

4) El creador o distribuïdor mateix del programari, que ofereix una sèrie de paquets d'instal·lació del seu programari. Podem no trobar el tipus de paquets necessari per a la nostra distribució.

5) Programari sense empaquetament o amb un empaquetament només de compressió sense cap tipus de dependències.

6) Només codi font, en forma de paquet o bé fitxer comprimit.

En les actualitzacions basades en punts com 3, 4, 5 i 6, és imprescindible mantenir certa seguretat en la distribució, per assegurar-se que són paquets "oficials" o d'una font de reputació coneguda, i comprovar l'autenticitat dels paquets mitjançant signatures o comprovacions de *hash*. Així mateix és possible que els paquets disposin d'actualitzacions freqüents que haurem d'integrar periòdicament, per exemple les diferents correccions o *updates*, relacionats amb seguretat, amb correccions de vulnerabilitats detectades.

Cal recordar, a més a més, que Snap/Flatpack, com a nous formats d'empaquetat ens permetrien instal·lar paquets de programari en principi independents de la distribució, i que podrem incorporar a qualsevol distribució que suporti el sistema utilitzat (sigui Snappy o Flatpak). Caldrà esperar quina és l'evolució futura d'aquests sistemes d'emmagatzematge "universal".

10. Feines no interactives

En les tasques d'administració, sol ser necessària l'execució a intervals temporals de certes tasques, ja sigui per a programar les tasques per a fer-les en horaris de menys ús de la màquina, o bé per la naturalesa periòdica mateixa de les tasques que es vulguin desenvolupar.

Per a dur a terme aquest tipus de feines "fora d'hores", com a serveis periòdics o programats, hi ha diversos sistemes que ens permeten construir un tipus d'agenda de tasques (planificació d'execució de tasques):

- ***nohup***: és potser el cas més simple utilitzat pels usuaris. Els permet l'execució d'una tasca no interactiva una vegada que hagin sortit del seu compte. En sortir del compte, l'usuari perd els seus processos, i *nohup* permet deixar-los en execució, malgrat que l'usuari es desconnecti.

- ***at***: ens deixa llançar una acció per a executar-la més tard, programant un determinat instant en què s'iniciarà, especificant l'hora (*hh:mm*) i la data, o bé si es farà avui (*today*) o demà (*tomorrow*).

```
at 10pm tasca
```

fer la tasca a les deu de la nit.

```
at 2am tomorrow tasca
```

fer la tasca a les dues de la matinada.

- ***cron***: permet establir una llista de feines per fer amb la seva programació; aquesta configuració es desa en /etc/crontab. Concretament, en cada entrada d'aquest fitxer tenim: minuts i hora que s'efectuarà la tasca, quin dia del mes, quin mes, quin dia de la setmana, juntament amb què (ja sigui una tasca, o bé un directori en què hi haurà les tasques per executar).

De manera estàndard, el contingut és semblant al següent:

```
25 6 * * * root test -e /usr/sbin/anacron || run-parts --report
/etc/cron.daily
47 6 * * 7 root test -e /usr/sbin/anacron || run-parts --report
/etc/cron.weekly
52 6 1 * * root test -e /usr/sbin/anacron || run-parts --report
/etc/cron.monthl
```

en què s'està programant que una sèrie de tasques es faran: cada dia (* indica *qualsevol*), setmanalment (el setè dia de la setmana), o mensualment (el primer de cada mes). Les feines serien executades per la instrucció *crontab,* però el sistema *cron* implica que la màquina està sempre encesa. Si no és així, és millor utilitzar *anacron,* que verifica si l'acció no es va fer quan l'hauria hagut de fer, i llavors l'executa. En cada línia del fitxer anterior es verifica que hi hagi la instrucció *anacron* i s'executen els *scripts* associats a cada acció, que en aquest cas estan desats en uns directoris assignats per a això, els *cron*.

També hi pot haver uns fitxers *cron.allow, cron.deny*, per a limitar qui pot col·locar (o no) tasques en *cron*. Mitjançant la instrucció *crontab,* un usuari pot definir tasques en el mateix format que hem vist abans, que es guardaran en /var/spool/cron/crontabs. En alguns casos, hi ha també un directori /etc/cron.d on es poden col·locar feines i que és tractat com si fos una extensió del fitxer /etc/crontab. En algunes versions, el sistema ja especifica les seves feines periòdiques de sistema directament sobre subdirectoris de */etc* com cron.hourly/, cron.daily/, cron.weekly i cron.montly/, on es col·loquen les feines de sistema que necessiten aquesta periodicitat.

En sistemes amb arrencada *systemd*, aquest ofereix *systemd-cron*, com un servei basat en les seves possibilitats de temporitzadors i calendaris, que li permet usar els directoris estàndard *cron.hourly*, *daily*, *monthly*, i amb `$ systemctl start crond.service` o *stop*, podem arrencar o parar el servei. Depenent de la distribució i el suport de *systemd*, s'utilitza aquesta capa de compatibilitat amb el cron clàssic, o pot utilitzar-se la definició de *timers* propis de *systemd*, per a crear tasques usant directament només *systemd*. Es recomana consultar les pàgines man de la distribució, referents a *systemd*, systemd.timer, cron, i crond, per a conèixer els mètodes habilitats.

11. Taller: pràctiques combinades dels apartats

Començarem per examinar l'estat general del nostre sistema. Farem els diferents passos en un sistema Debian. Encara que es tracta d'un sistema Debian, els procediments són majoritàriament traslladables a altres distribucions, com Fedora/Red Hat (esmentarem alguns dels canvis més importants). El maquinari consisteix en una antiga màquina Pentium 4 a 2.66 MHz amb 768 MB i diversos discos, DVD i gravador de CD, a més d'altres perifèrics, però ja anirem obtenint aquesta informació pas per pas. En aquest cas particular hem seleccionat una màquina bastant antiga per als estàndards actuals. Precisament una de les versatilitats de GNU/Linux ens permet recuperar màquines obsoletes (segons altres entorns operatius) i convertir-les en funcionals per a tasques determinades: servidors de fitxers, impressores, servidors web, *streaming* àudio/vídeo, etc.

Vegem primer com ha engegat el nostre sistema l'última vegada:

```
# uptime

17:38:22 up 2:46, 5 users, load average: 0.05, 0.03, 0.04
```

Aquest instrucció ens dóna el temps que fa que funciona el sistema des que es va arrencar l'última vegada: 2 hores i 46 minuts. En el nostre cas tenim cinc usuaris. Aquests no han de ser necessàriament cinc usuaris diferents, sinó que seran les sessions d'usuari obertes (per exemple, mitjançant un terminal). La instrucció *who* permet mostrar aquests usuaris. El *load average* és la càrrega mitjana del sistema en els últims 1, 5 i 15 minuts.

Vegem el registre de l'arrencada del sistema (instrucció *dmesg*), les línies que s'anaven generant en la càrrega del sistema (s'han suprimit diferents línies per claredat):

```
Linux version 2.6.20-1-686 (Debian 2.6.20-2) (waldi@debian.org) (gcc
version 4.1.2 20061115 (prerelease) (Debian 4.1.1-21)) #1 SMP Sun Apr 15 21:03:57 UTC
BIOS-provided physical RAM map:
BIOS-e820: 0000000000000000 - 000000000009f800 (usable)
BIOS-e820: 000000000009f800 - 00000000000a0000 (reserved)
BIOS-e820: 00000000000ce000 - 00000000000d0000 (reserved)
BIOS-e820: 00000000000dc000 - 0000000000100000 (reserved)
BIOS-e820: 0000000000100000 - 000000002f6e0000 (usable)
BIOS-e820: 000000002f6e0000 - 000000002f6f0000 (ACPI data)
BIOS-e820: 000000002f6f0000 - 000000002f700000 (ACPI NVS)
BIOS-e820: 000000002f700000 - 000000002f780000 (usable)
BIOS-e820: 000000002f780000 - 0000000030000000 (reserved)
```

```
BIOS-e820: 00000000ff800000 - 00000000ffc00000 (reserved)
BIOS-e820: 00000000fffffc00 - 0000000100000000 (reserved)
0MB HIGHMEM available.

759MB LOWMEM available.
```

Aquestes primeres línies ja ens indiquen diverses dades interessants: la versió del nucli Linux és la 2.6.20-1-686, una versió 2.6 revisió 20 amb revisió 1 de Debian, i per a màquines 686 (arquitectura Intel x86 de 32 bits). Indica, també, que estem arrencant un sistema Debian, amb aquest nucli, que va ser compilat amb un compilador GNU GCC versió 4.1.2. A continuació, es veu un mapa de zones de memòria usades (reservades) per la BIOS, i a continuació el total de memòria detectada a la màquina: 759 MB, als quals caldria sumar el primer 1 MB, amb un total de 760 MB.

```
Kernel command line: BOOT_IMAGE=LinuxNEW ro root=302 lang=es acpi=force
Initializing CPU#0
Console: colour dummy device 80x25
Memory: 766132k/777728k available (1641k kernel code, 10968k reserved, 619k
data, 208k init, 0k highmem)
Calibrating delay using timer specific routine.. 5320.63 BogoMIPS (lpj=10641275)
```

Aquí ens refereix com ha estat l'arrencada de la màquina, quina línia d'instruccions se li ha passat al nucli (es poden passar diferents opcions, per exemple des del LILO o GRUB). I estem arrencant en mode consola de 80 × 25 caràcters (això es pot canviar). Els *BogoMIPS* són una mesura interna del nucli de la velocitat de la CPU; hi ha arquitectures en què és difícil detectar a quants MHz o GHz funciona la CPU, i per això s'utilitza aquesta mesura de velocitat. Després ens dóna més dades de la memòria principal, i ens diu per a què s'està usant en aquest moment de l'arrencada.

```
CPU: Trace cache: 12K uops, L1 D cache: 8K
CPU: L2 cache: 512K
CPU: Hyper-Threading is disabled
Intel machine check architecture supported.
Intel machine check reporting enabled on CPU#0.
CPU0: Intel P4/Xeon Extended MCE MSRs (12) available
CPU0: Intel(R) Pentium(R) 4 CPU 2.66GHz stepping 09
```

A més, ens proporciona dades diverses de la CPU, la mida de les memòries cau de primer nivell, la memòria cau interna de la CPU, L1 dividida en una *Trace-Cache* del Pentium 4 (o *instruction cache*), i la cau de dades, i la cau unificada de segon nivell (*L2*), el tipus de CPU, la seva velocitat i la del bus del sistema.

```
PCI: PCI BIOS revision 2.10 entry at 0xfd994, last bus=3
Setting up standard PCI resources
...
```

```
NET: Registered protocol
IP route cache hash table entries: 32768 (order: 5, 131072 bytes)
TCP: Hash tables configured (established 131072 bind 65536)
checking if image is initramfs... it is
Freeing initrd memory: 1270k freed
fb0: VESA VGA frame buffer device
Serial: 8250/16550 driver $Revision: 1.90 $ 4 ports, IRQ sharing enabled
serial8250: ttyS0 at I/O 0x3f8 (irq = 4) is a 16550A
00:09: ttyS0 at I/O 0x3f8 (irq = 4) is a 16550A
RAMDISK driver initialized: 16 RAM disks of 8192K size 1024 blocksize
PNP: PS/2 Controller [PNP0303:KBC0,PNP0f13:MSE0] at 0x60,0x64 irq 1,12
i8042.c: Detected active multiplexing controller, rev 1.1.
serio: i8042 KBD port at 0x60,0x64 irq 1
serio: i8042 AUX0 port at 0x60,0x64 irq 12
serio: i8042 AUX1 port at 0x60,0x64 irq 12
serio: i8042 AUX2 port at 0x60,0x64 irq 12
serio: i8042 AUX3 port at 0x60,0x64 irq 12
mice: PS/2 mouse device common for all mice
```

Continuen les inicialitzacions del nucli i els dispositius; esmenta la inicialització de protocols de xarxa. Els terminals, els ports sèrie *ttyS0* (seria el *COM1*), *ttyS01* (el *COM2*), dóna informació dels discos RAM que usem i detecta dispositius PS2, teclat i ratolí.

```
ICH4: IDE controller at PCI slot 0000:00:1f.1

  ide0: BM-DMA at 0x1860-0x1867, BIOS settings: hda:DMA, hdb:pio
  ide1: BM-DMA at 0x1868-0x186f, BIOS settings: hdc:DMA, hdd:pio
Probing IDE interface ide0...
hda: FUJITSU MHT2030AT, ATA DISK drive
ide0 at 0x1f0-0x1f7,0x3f6 on irq 14
Probing IDE interface ide1...
hdc: SAMSUNG CDRW/DVD SN-324F, ATAPI CD/DVD-ROM drive
ide1 at 0x170-0x177,0x376 on irq 15
SCSI subsystem initialized
libata version 2.00 loaded.
hda: max request size: 128KiB
hda: 58605120 sectors (30005 MB) w/2048KiB Cache, CHS=58140/16/63<6>hda: hw_config=600b
, UDMA(100)
hda: cache flushes supported
hda: hda1 hda2 hda3
kjournald starting. Commit interval 5 seconds
EXT3-fs: mounted filesystem with ordered data mode.
hdc: ATAPI 24X DVD-ROM CD-R/RW drive, 2048kB Cache, UDMA(33)
Uniform CD-ROM driver Revision: 3.20
Addinf 618492 swap on /dev/hda3.
```

Detecció de dispositius IDE, detecta el xip IDE al bus PCI i informa que està controlant dos dispositius: *hda* i *hdc*, que són, respectivament, un disc dur (Fujitsu), un segon disc dur, un DVD Samsung i una enregistradora de CD (ja que en aquest cas es tracta d'una unitat combinada). Indica particions actives en el disc. Més endavant, detecta el sistema principal de fitxers de Linux, un *ext3* amb *journal*, que activa i afegeix l'espai de *swap* disponible en una partició.

```
usbcore: registered new interface driver usbfs
usbcore: registered new interface driver hub
usbcore: registered new device driver usb
input: PC Speaker as /class/input/input1
USB Universal Host Controller Interface driver v3.0
hub 1-0:1.0: USB hub found
hub 1-0:1.0: 2 ports detected
uhci_hcd 0000:00:1d.1: UHCI Host Controller
uhci_hcd 0000:00:1d.1: new USB bus registered, assigned bus number 2
uhci_hcd 0000:00:1d.1: irq 11, io base 0x00001820
usb usb2: configuration #1 chosen from 1 choice
hub 2-0:1.0: USB hub found
hub 2-0:1.0: 2 ports detected
hub 4-0:1.0: USB hub found
hub 4-0:1.0: 6 ports detected
```

Més detecció de dispositius, USB en aquest cas (i els mòduls que hi corresponen). Ha detectat dos dispositius *hub* (amb un total de 8 ports USB).

```
parport: PnPBIOS parport detected.
parport0: PC-style at 0x378 (0x778), irq 7, dma 1 [PCSPP,TRISTATE,COMPAT,EPP,ECP,DMA]
input: ImPS/2 Logitech Wheel Mouse as /class/input/input2
ieee1394: Initialized config rom entry `ip1394'
eepro100.c:v1.09j-t 9/29/99 Donald Becker
Synaptics Touchpad, model: 1, fw: 5.9, id: 0x2e6eb1, caps: 0x944713/0xc0000
input: SynPS/2 Synaptics TouchPad as /class/input/input3

agpgart: Detected an Intel 845G Chipset
agpgart: Detected 8060K stolen Memory
agpgart: AGP aperture is 128M
eth0: OEM i82557/i82558 10/100 Ethernet, 00:00:F0:84:D3:A9, IRQ 11.
 Board assembly 000000-000, Physical connectors present: RJ45
e100: Intel(R) PRO/100 Network Driver, 3.5.17-k2-NAPI
usbcore: registered new interface driver usbkbd
Initializing USB Mass Storage driver...
usbcore: registered new interface driver usb-storage
USB Mass Storage support registered.

lp0: using parport0 (interrupt-driven).
```

```
ppdev: user-space parallel port driver
```

I la detecció final de la resta de dispositius: port paral·lel, model de ratolí, port Firewire (*ieee1394*), targeta de xarxa (Intel), un tauler tàctil, la targeta de vídeo AGP (*i845*). Més dades de la targeta de xarxa: una Intel Pro 100, registre d'USB com a *mass storage* (indica un dispositiu d'emmagatzematge per USB, com un disc extern) i detecció del port paral·lel.

Tota aquesta informació, que hem vist mitjançant la instrucció *dmesg,* també la podem trobar bolcada en el registre principal del sistema, /var/log/messages. En aquest registre, entre altres, trobarem els missatges del nucli i dels dimonis, i errors de xarxa o dispositius, els quals comuniquen els seus missatges a un dimoni especial anomenat *syslogd,* que és l'encarregat d'escriure els missatges en aquest fitxer. Si hem arrencat la màquina recentment, observarem que les últimes línies contenen exactament la mateixa informació que la instrucció *dmesg*; per exemple, si ens quedem amb la part final del fitxer (sol ser molt gran):

```
tail 200 /var/log/messages
```

Observem les línies d'abans, i també algunes informacions més, com per exemple:

```
shutdown[13325]: shutting down for system reboot
kernel: usb 4-1: USB disconnect, address 3
kernel: nfsd: last server has exited
kernel: nfsd: unexporting all filesystems
kernel: Kernel logging (proc) stopped.
kernel: Kernel log daemon terminating.

exiting on signal 15
syslogd 1.4.1#20: restart.

kernel: klogd 1.4.1#20, log source = /proc/kmsg started.
Linux version 2.6.20-1-686 (Debian 2.6.20-2) (waldi@debian.org) (gcc version
4.1.2 20061115 (prerelease) (Debian 4.1.1-21)) #1 SMP Sun Apr 15 21:03:57

kernel: BIOS-provided physical RAM map:
```

La primera part correspon a l'aturada anterior del sistema; ens informa que el nucli ha deixat de col·locar informació en /proc, s'està parant el sistema... Al principi de l'arrencada nova, s'activa el dimoni *syslogd*, que genera el registre i comença la càrrega del sistema, que ens diu que el nucli començarà a escriure informació en el seu sistema /proc. Veiem les primeres línies de *dmesg* d'esment de la versió que s'està carregant de nucli, i després trobarem el que hem vist amb *dmesg*.

En aquest punt, una altra instrucció útil per a saber com s'ha produït la càrrega és *lsmod*, que ens permetrà saber quins mòduls dinàmics s'han carregat amb el nucli (versió resumida):

```
# lsmod
Module          Size      Used by
nfs             219468    0
nfsd            202192    17
exportfs        5632      1 nfsd
lockd           58216     3 nfs,nfsd
nfs_acl         3616      2 nfs,nfsd
sunrpc          148380    13 nfs,nfsd,lockd,nfs_acl
ppdev           8740      0
lp              11044     0
button          7856      0
ac              5220      0
battery         9924      0
md_mod          71860     1
dm_snapshot     16580     0
dm_mirror       20340     0
dm_mod          52812     2 dm_snapshot,dm_mirror
i810fb          30268     0
vgastate        8512      1 i810fb
eeprom          7184      0
thermal         13928     0
processor       30536     1 thermal
fan             4772      0
udf             75876     0
ntfs            205364    0
usb_storage     75552     0
hid             22784     0
usbkbd          6752      0
eth1394         18468     0
e100            32648     0
eepro100        30096     0
ohci1394        32656     0
ieee1394        89208     2 eth1394,ohci1394
snd_intel8x0    31420     1
snd_ac97_codec  89412     1 snd_intel8x0
ac97_bus        2432      1 snd_ac97_codec
parport_pc      32772     1
snd             48196     6 snd_intel8x0,snd_ac97_codec,snd_pcm,snd_timer
ehci_hcd        29132     0
ide_cd          36672     0
cdrom           32960     1 ide_cd
soundcore       7616      1 snd
psmouse         35208     0
```

```
uhci_hcd        22160     0
parport         33672     3 ppdev,lp,parport_pc
intelfb         34596     0
serio_raw       6724      0
pcspkr          3264      0
pci_hotplug     29312     1 shpchp
usbcore         122312    6 dvb_usb,usb_storage,usbkbd,ehci_hcd,uhci_hcd
intel_agp       22748     1
agpgart         30504     5 i810fb,drm,intelfb,intel_agp
ext3            121032    1
jbd             55368     1 ext3
ide_disk        15744     3
ata_generic     7876      0
ata_piix        15044     0
libata          100052    2 ata_generic,ata_piix
scsi_mod        133100    2 usb_storage,libata
generic         4932      0 [permanent]
piix            9540      0 [permanent]
ide_core        114728    5 usb_storage,ide_cd,ide_disk,generic,piix
```

Veiem que disposem dels controladors per al maquinari que hem detectat, i d'altres de relacionats o necessaris per dependències.

Ja tenim, doncs, una idea de com s'han carregat el nucli i els seus mòduls. En aquest procés pot ser que ja hàgim observat algun error; si hi ha maquinari mal configurat o mòduls del nucli mal compilats (no eren per a la versió del nucli adequada), inexistents, etc.

El pas següent serà l'observació dels processos en el sistema, amb la instrucció *ps* (*process status*), per exemple (només s'han mostrat els processos de sistema, no els dels usuaris):

```
#ps -ef
UID PID PPID C STIME TTY TIME CMD
```

Informació dels processos: *UID*, usuari que ha llançat el procés (o amb quin identificador s'ha llançat); *PID*, codi del procés assignat pel sistema –són consecutius a mesura que es llancen els processos, el primer sempre és el 0, que correspon al procés d'*init*–; *PPID* és l'ID del procés pare de l'actual; *STIME*, el temps en què va ser engegat el procés; *TTY*, el terminal assignat al procés (si en té algun); *CMD*, línia d'instruccions amb què va ser llançat.

```
root    1    0 0 14:52 ?       00:00:00 init [2]
root    3    1 0 14:52 ?       00:00:00 [ksoftirqd/0]
root   143   6 0 14:52 ?       00:00:00 [bdflush]
root   145   6 0 14:52 ?       00:00:00 [kswapd0]
root   357   6 0 14:52 ?       00:00:01 [kjournald]
```

```
root   477   1 0 14:52 ?       00:00:00 udevd --daemon
root   719   6 0 14:52 ?       00:00:00 [khubd]
```

Diversos dimonis de sistema, com *kswapd,* dimoni que controla l'intercanvi de pàgines amb memòria virtual. Gestió de memòries intermèdies del sistema (*bdflush*). Gestió de *journal* de *filesystem* (*kjournald*), gestió d'USB (*khubd*). O el dimoni *udev*, que controla la connexió en calent de dispositius. En general (no sempre) els dimonis se solen identificar per una *d* final, i si porten una *k* inicial, normalment són fils (*threads*) interns del nucli.

```
root      1567 1 0 14:52 ?    00:00:00 dhclient -e -pf ...
root      1653 1 0 14:52 ?    00:00:00 /sbin/portmap
root      1829 1 0 14:52 ?    00:00:00 /sbin/syslogd
root      1839 1 0 14:52 ?    00:00:00 /sbin/klogd -x
root      1983 1 0 14:52 ?    00:00:09 /usr/sbin/cupsd
root      2178 1 0 14:53 ?    00:00:00 /usr/sbin/inetd
```

Tenim: *dhclient*, que indica que aquesta màquina és clienta d'un servidor DHCP, per a obtenir la seva IP; *syslogd*, dimoni que envia missatges al registre; el dimoni de *cups*, com hem vist, està relacionat amb el sistema d'impressió; i *inetd*, que, com veurem en la part de xarxes, és una espècie de "superservidor" o mediador d'altres dimonis relacionats amb serveis de xarxa.

```
root      2154 1 0 14:53 ?    00:00:00 /usr/sbin/rpc.mountd
root      2241 1 0 14:53 ?    00:00:00 /usr/sbin/sshd
root      2257 1 0 14:53 ?    00:00:00 /usr/bin/xfs -daemon
root      2573 1 0 14:53 ?    00:00:00 /usr/sbin/atd
root      2580 1 0 14:53 ?    00:00:00 /usr/sbin/cron
root      2675 1 0 14:53 ?    00:00:00 /usr/sbin/apache
www-data  2684 2675 0 14:53 ? 00:00:00 /usr/sbin/apache
www-data  2685 2675 0 14:53 ? 00:00:00 /usr/sbin/apache
```

També hi ha *sshd*, servidor d'accés remot segur (una versió millorada que permet serveis compatibles amb Telnet i FTP); *xfs* és el servidor de tipus de lletra d'X Window. Les instruccions *atd* i *cron* serveixen per a manejar tasques programades en un moment determinat. Apache és el servidor web, que pot tenir diverses cadenes actives (*threads*) per a atendre diferents peticions.

```
root   2499 2493 0 14:53 ?       00:00:00 /usr/sbin/gdm
root   2502 2499 4 14:53 tty7    00:09:18 /usr/bin/X :0 -dpi 96 ...
root   2848   1 0 14:53 tty2     00:00:00 /sbin/getty 38400 tty2
root   2849   1 0 14:53 tty3     00:00:00 /sbin/getty 38400 tty3
root   3941 2847 0 14:57 tty1    00:00:00 -bash
root   16453 12970 0 18:10 pts/2 00:00:00 ps -ef
```

Així, *gdm* és la connexió gràfica del sistema d'escriptori Gnome (l'entrada que ens demana la connexió i la contrasenya); els processos *getty* són els que gestionen els terminals virtuals de text (els que podem veure amb les tecles Alt +Fn, o Ctrl+Alt+Fn si estem en mode gràfic); *X* és el procés del servidor gràfic X Window System, imprescindible perquè s'executi qualsevol entorn d'escriptori a sobre. Un intèrpret obert (*bash*), i finalment el procés que hem generat en demanar aquest *ps* des de la línia d'instruccions.

La instrucció *ps* té moltes opcions de línia d'ordres per a ajustar la informació que volem de cada procés, ja sigui el temps que fa que s'executa, el percentatge de CPU usat, la memòria utilitzada, etc. (vegeu *man ps*). Una altra instrucció molt interessant és *top,* que fa el mateix que *ps* però de manera dinàmica; s'actualitza cada cert interval; podem classificar els processos per ús de CPU, de memòria, i també ens dóna informació de l'estat de la memòria global.

Nota

Vegeu el *man* de les instruccions per a interpretar les sortides.

Altres instruccions útils per a l'ús de recursos són *free* i *vmstat*, que ens donen informació sobre la memòria utilitzada i el sistema de memòria virtual:

```
# free
             total    used    free    shared    buffers    cached
Mem:         767736    745232    22504          0     89564    457612
-/+ buffers/cache:   198056   569680
Swap:               618492    1732   616760

# vmstat
procs -----------memory----------   ---swap-- -----io-- --system-- ----cpu----
 r b swpd free      buff    cache   si so bi bo    in   cs us sy  id wa

 1 0 1732 22444     89584  457640   0   0  68 137  291 418 7  1   85 7
```

En la instrucció *free* també es pot observar la mida del *swap* present, aproximadament d'uns 600 MB, que de moment no s'està usant intensament per tenir suficient espai de memòria física. Encara en queden uns 22 MB lliures (fet que indica una alta utilització de la memòria física i un ús proper de *swap*). L'espai de memòria i el *swap* (a partir dels nuclis 2.4) són additius per a compondre el total de memòria del sistema, i en aquest cas fa un total d'1,4 GB de memòria disponible.

Activitats

1. Feu una lectura bàsica de l'estàndard FHS, que ens servirà per a tenir una bona guia a l'hora de buscar arxius per a la nostra distribució.

Nota

Vegeu FHS a: http://www.pathname.com/fhs

2. Per als paquets RPM, com faríeu algunes de les tasques següents?

a) Conèixer quin paquet va instal·lar una determinada instrucció.

b) Obtenir la descripció del paquet que va instal·lar una instrucció.

c) Esborrar un paquet del qual no coneixem el nom complet.

d) Mostrar tots els arxius que eren al mateix paquet que un determinat arxiu.

3. Efectueu les mateixes tasques que en l'activitat anterior, però per a paquets Debian, usant eines APT.

4. Actualitzeu una distribució Debian (o Fedora).

5. Instal·leu en la nostra distribució alguna eina genèrica d'administració, com Webmin. Què ens ofereix? És fàcil comprendre les tasques executades i els efectes que provoquen?

6. L'espai de *swap* permet complementar la memòria física per a disposar de més memòria virtual. Depenent de les mides de memòria física i *swap*, es pot arribar a esgotar la memòria? Ho podem solucionar d'una altra manera que no sigui afegint més memòria física?

7. Suposem que tenim un sistema amb dues particions Linux, una / i l'altra de *swap*. Com solucionaríeu el cas que els comptes dels usuaris esgotessin l'espai de disc? I en el cas de tenir una partició /home aïllada que s'estigués esgotant, com ho solucionaríeu?

8. Instal·leu el sistema d'impressió CUPS, definiu la nostra impressora perquè funcioni amb CUPS i proveu l'administració via interfície web. Tal com està el sistema, seria recomanable modificar d'alguna manera la configuració que porta per defecte CUPS? Per què?

9. Analitzeu el sistema Upstart, o systemd presents en una distribució Debian, Ubuntu o Fedora. Quins esdeveniments i *jobs* porta predefinits? Hi ha compatibilitat amb l'*init* de System V (sysvinit)?

10. Examineu la configuració per defecte que porti el sistema GNU/Linux per a feines no interactives amb *cron* (o systemd-cron). Quines feines s'estan fent? Quan s'estan fent? Alguna idea per a noves feines que calgui afegir?

11. Reproduïu l'anàlisi del taller (més els altres apartats de la unitat) sobre una màquina disponible; s'observen en el sistema alguns errors o situacions anòmales? En aquest cas, com n'orientem la resolució?

Bibliografia

[Bai03] Bailey, E. C. (2003). RedHat Maximum RPM. Ofereix una àmplia visió dels sistemes de paquets de programari de les distribucions Debian i Fedora/Red Hat.

[Bas] Mike, G. "BASH Programming - Introduction HOWTO". *The Linux Documentation Project*. Ofereix una àmplia introducció (i conceptes avançats) a la programació de *shell scripts* en Bash, i també nombrosos exemples.

[Coo] Cooper, M. (2006). "Advanced Bash Scripting Guide". *The Linux Documentation Project* (guies). Ofereix una àmplia introducció (i conceptes avançats) a la programació de *shell scripts* en Bash, i també nombrosos exemples.

[Deb] Debian. "Lloc de Seguretat de Debian". Ofereix una àmplia visió dels sistemes de paquets de programari de les distribucions Debian i Fedora/Red Hat.

[Debb] Comunitat Debian. "Distribució Debian". <http://www.debian.org>

[Fed] The Fedora Project. <http://fedoraproject.org/>

[Fri02] Frisch, A. (2002). *Essential System Administration*. O'Reilly. Administració de GNU/Linux i UNIX. Comenta de manera àmplia aspectes d'administració local i gestió de sistemes d'impressió.

[Gt] Taylor, G.; Allaert, D. "The Linux Printing HOWTO". *The Linux Documentation Project*. Ofereix informació actualitzada dels sistemes d'impressió i la seva configuració, i també detalls d'algunes impressores. Per a detalls concrets de models d'impressora i controladors, us podeu dirigir a: <http://www.linuxprinting.org/>

[Hin00] Hinner, M. "Filesystems HOWTO". *The Linux Documentation Project*. Informació sobre els diferents sistemes de fitxers disponibles i els esquemes de creació de particions per a la instal·lació del sistema.

[Koe] Koehntopp, K. "Linux Partition HOWTO". *The Linux Documentation Project*. Informació sobre els diferents sistemes de fitxers disponibles i els esquemes de creació de particions per a la instal·lació del sistema.

[Lin04b] *Filesystem Hierarchy Standard (FHS)*. http://www.pathname.com/fhs

[Lin11c] *Linux Standard Base (specifications Archive)*. <http://refspecs.linuxfoundation.org/lsb.shtml>

[Linc] *Linux Standards Base project*. <http://www.linux-foundation.org/en/LSB>

[Mor03] Morill, D. (2003). *Configuración de sistemas Linux*. Anaya Multimedia.

[Nem06] Nemeth, E.; Snyder, G.; Hein, T. R. (2006). "Linux Administration Handbook" (2a. ed.). Prentice Hall. Tracta de manera àmplia de la majoria d'aspectes d'administració i és una bona guia genèrica per a qualsevol distribució.

[Pow12] Powell, Howard (2012). "ZFS and Btrfs: a Quick introudction to Modern Filesystems". *Linux Journal* (issue 218, pàg. 104-111).

[Qui01] Quigley, E. (2001). *Linux shells by Example*. Prentice Hall. Comenta els diferents intèrprets de programació en GNU/Linux, i també les seves semblances i diferències.

[Sob10] Sobell, M. G. (2010). *A Practical Guide to Fedora and Red Hat Enterprise Linux*. Prentice Hall. És una bona guia d'administració local per a distribucions Red Hat i Fedora.

[SM02] Schwartz, M. i altres (2002). *Multitool Linux - Practical Uses for Open Source Software*. Addison Wesley.

[Smi02] Smith, R. (2002). *Advanced Linux Networking*. Addison Wesley. Administració de GNU/Linux i UNIX. Comenta de manera àmplia aspectes d'administració local i gestió de sistemes d'impressió.

[Soy12] Soyinka, W. (2012). *Linux Administration. A Beginner's Guide* (6a. ed.). McGraw Hill.

[Stu] Stutz, M. "The Linux Cookbook: Tips and Techniques for Everyday Use". *The Linux Documentation Project* (guies). És una àmplia introducció a les eines disponibles en GNU/Linux.

[War13] Ward, Allan (2013). "How To Use the Logic Volume Manager". *Full Circle Magazine* (issue #80, pàg. 12-18).

[Wm02] Welsh, M. i altres (2002). *Running Linux* (4a. ed.). O'Reilly. Administració de GNU/Linux i UNIX. Comenta de manera àmplia aspectes d'administració local i gestió de sistemes d'impressió.