Introducció a GNU/Linux

PID_00270619

Gerard Farràs Ballabriga

Temps mínim de dedicació recomanat: 4 hores

Universitat Oberta de Catalunya

Gerard Farràs Ballabriga

Enginyer tècnic en Informàtica de sistemes per la Universitat Autònoma de Barcelona (UAB). Enginyer en Informàtica i màster en Societat de la Informació i el Coneixement per la Universitat Oberta de Catalunya (UOC). Actualment treballa com a professor en una escola de secundària i formació professional. Anteriorment ha desenvolupat la seva activitat professional en l'àrea de sistemes d'informació d'un centre tecnològic i també com a professional autònom (*freelance*) treballant com administrador de sistemes i desenvolupador web.

L'encàrrec i la creació d'aquest recurs d'aprenentatge UOC han estat coordinats pel professor: Julià Minguillón Alfonso (2020)

Primera edició: febrer 2020

Av. Tibidabo, 39-43, 08035 Barcelona
Realització editorial: FUOC

Índex

1. Introducció..... 5

2. Breu història de Linux..... 6

3. Entorn gràfic contra línia de comandes..... 8

4. Usuaris del sistema..... 11

5. El sistema de fitxers..... 13

6. Cerca de fitxers..... 18

7. Manipulació de fitxers..... 20

8. Compressió de fitxers..... 26

9. Variables d'entorn..... 27

10. Editors de text..... 29

11. Processos amb GNU/Linux..... 30

12. Execució de processos automatitzats amb Cron..... 34

13. Xarxa, connectivitat i clients web..... 36

14. Creació de guions de seqüencies (*scripts*)..... 39

15. Distribucions de GNU/Linux..... 41
- 15.1. Introducció 41
- 15.2. La distribució escollida: Ubuntu 41
- 15.3. Instal·lació d'Ubuntu 42
- 15.4. Instal·lació de programari 44
  - 15.4.1. Instal·lació de programari emprant codi font 44
  - 15.4.2. Instal·lació de programari emprant repositoris d'Ubuntu 45

Bibliografia..... 49

1. Introducció

L'objectiu d'aquest document consisteix a oferir una introducció al sistema operatiu GNU/Linux per a especialistes en tractament de dades.

GNU/Linux és un sistema avançat, ric en eines i possibilitats d'automatització, estable, a més de gratuït i amb llicència de programari lliure.

És per aquests motius que es recomana als estudiants l'aprenentatge d'aquest sistema operatiu. Certament que seria possible fer la majoria d'operacions amb altres sistemes, però considerem que aquest és el més adequat. D'altra banda, l'aprenentatge sobre el funcionament d'un sistema operatiu es pot realitzar des de múltiples perspectives (per exemple, posant el focus en el funcionament intern del sistema, en el seu rendiment o en la seva seguretat). Però, l'objectiu d'aquest document és oferir unes primeres pinzellades inicials sobre què és GNU/Linux i l'enfocament en les comandes necessàries per al tractament de dades massives, deixant de banda la resta de perspectives.

L'objectiu pedagògic d'aquest document consisteix a oferir una introducció als sistemes GNU/Linux per a usuaris no experimentats i a oferir als estudiants una guia per tal que puguin emprar aquest tipus de sistemes per a les tasques relacionades amb la ciència de dades.

La nostra recomanació per a aconseguir aquest objectiu seria disposar d'un sistema GNU/Linux (la manera més senzilla és emprant una màquina virtual) i llegir, en paral·lel, aquest document a fi d'anar provant totes les comandes. Els usuaris ja més experimentats empraran altres mètodes per a l'aprenentatge. Són els usuaris més neòfits els que necessitaran una guia inicial per a començar. En cap cas, no es tracta de memoritzar cap comanda ni res: a mesura que es facin servir es recordaran per repetició. Així doncs, es recomana instal·lar un sistema, llegir aquest document i fer algunes primeres proves. L'ordre de lectura tampoc no és excessivament important: es poden llegir i provar apartats concrets: no cal una lectura lineal. A mesura que l'aprenentatge avanci, aquesta guia ja no serà de tant interès.

Usuaris experimentats

Els usuaris experimentats empraran altres mètodes per a l'aprenentatge, per exemple, utilitzant els manuals interns de la màquina, cerques avançades a internet i més «assaig-error»

2. Breu història de Linux

Linux és un sistema operatiu de programari lliure basat en el *kernel* desenvolupat per en Linus Torvalds, i alliberat per primera vegada el 17 de setembre de 1991. Linux acostuma a anar empaquetat en distribucions que contenen, a més del *kernel*, programari de sistema, llibreries i utilitats diverses, moltes de les quals del projecte GNU. Aquest GNU[1] va ser iniciat per en Richard Stallman amb l'objectiu de generar programari lliure amb la llicència pública general (GPL).

(1)Acrònim recursiu que juga amb les paraules i significa *GNU's Not Unix*.

Kernel

Amb *kernel* ens referim al cor del sistema operatiu.

El 25 d'agost de 1991, l'estudiant finlandès Linus Torvalds escrivia aquest famós missatge en la llista de notícies comp.os.minix, en què anunciava l'embrió de què seria Linux:

Minix

Minix és un altre sistema operatiu d'entorn acadèmic.

És possible llegir el fil sencer en «What would you like to see most in minix?».

```
Hello everybody out there using Minix -

I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu)
 for 386(486) AT clones. This has been brewing since April, and is starting to get ready.
 I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat
 (same physical layout of the file-system (due to practical reasons) among other things).

I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that
I'll get something practical within a few months, and I'd like to know what features most
people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)

Linus (torv...@kruuna.helsinki.fi)

PS. Yes - it's free of any minix code, and it has a multi-threaded fs. It is NOT portable
(uses 386 task switching etc), and it probably never will support anything other than
AT-harddisks, as that's all I have :-(.
```

Des que el desenvolupament del *kernel* va començar a emprar un programari que s'anomena *Git* (l'any 2005) i que ha permès, entre altres coses, veure millor les contribucions al codi font d'aquest *kernel*, es pot afirmar que un total de 15.637 desenvolupadors han contribuït al sistema. L'any 1991 aquest *kernel* contenia un petit nombre de fitxers escrits en llenguatge de programació C. La versió 4.15 de l'any 2018 contenia més de 23 milions de línies de codi.

Git

Git és un programari per al control de versions, també dissenyat per en Linus Torvalds.

Linux és un exemple prominent de **col·laboració oberta en el desenvolupament de programari**, ja que, durant anys, s'ha generat un producte altament complex gràcies a la intervenció de centenars de persones distribuïdes arreu del planeta amb un model obert i més o menys descentralitzat.

Lectura recomanada

Vegeu **J. Corbet, G. Kroah-Hartman** (2017). «Linux Kernel Development Report» [en línia]. Fundación Linux. Disponible a: go.pardot.com/l/6342/2017-10-24/3xr3f2/6342/188781/Publication_LinuxKernelReport_2017.pdf.

Respecte a la llicència pública general GNU[2] és la llicència per a programari lliure més utilitzada, que permet fer còpies, distribuir-les (tant en versions comercials com que no ho siguin) i modificar el codi font, sempre que qualsevol modificació es continuï distribuint amb la mateixa llicència. El que no permet la llicència GPL és la distribució de programari executable sense disposar del codi font corresponent. En Richard Stallman, de la Free Software Foundation, va ser el primer autor pel projecte GNU, que atorgava als destinataris d'un programa d'ordinador els drets corresponents al programari lliure. Es tracta d'una llicència *copyleft*, fet que significa que els treballs derivats s'han de distribuir amb els mateixos termes.

(2)GNU GPL, GNU *General Public License*.

Free Software Foundation

La Free Software Foundation (FSF) és una organització sense ànim de lucre global que té com a objectiu la promoció del programari lliure (en el sentit d'atorgar llibertat a l'usuari).

Figura 1. Fotografia d'en Linus Torvalds rebent un premi l'any 2018

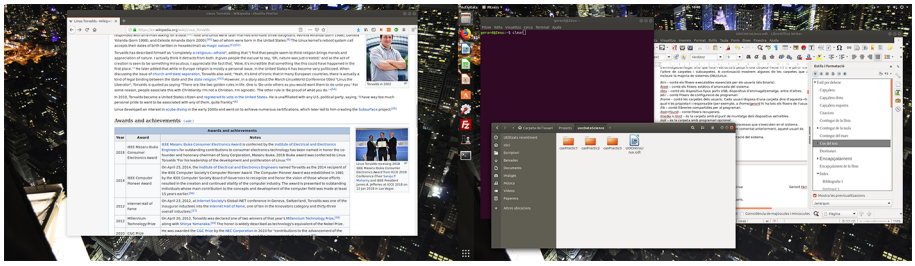
Font: imatge extreta de la Viquipèdia (en.wikipedia.org/wiki/Linus_Torvalds).

3. Entorn gràfic contra línia de comandes

La majoria de sistemes operatius actuals s'utilitzen fent servir entorns de finestres (per mitjà d'interfícies gràfiques d'usuari),[3] malgrat mantenen la possibilitat de ser utilitzats per mitjà de terminals. L'ús d'un terminal és certament més complicat i menys intuïtiu per als usuaris no experimentats però atorga més flexibilitat. A més, hi ha tasques que solament es poden fer emprant comandes de terminal. El gran avantatge de l'ús de terminals consisteix en la possibilitat d'automatització de tasques per mitjà d'*scripts* (guions de seqüències). Un altre avantatge consisteix en el fet que hi ha multitud d'eines que funcionen solament per mitjà de terminal, i no en programari basat en finestres.

[3] En anglès GUI, *Graphical User Interface*.

Figura 2. Captura de pantalla de l'escriptori d'una màquina amb la distribució Ubuntu en entorn gràfic



Aquest escriptori concret conté dues pantalles diferents.

Consell

Aconsellem treballar en una instal·lació similar amb entorn gràfic, però caldrà aprendre a utilitzar les comandes de terminal.

Així doncs, caldrà que l'alumnat es familiaritzi amb l'ús de terminals en sistemes operatius GNU/Linux. Una vegada instal·lat un sistema operatiu GNU/Linux en mode gràfic, cal cercar solament l'aplicació de «Terminal» a la barra d'aplicacions. La resta del manual que escrivim a continuació es basarà en els detalls d'aquest entorn.

Quan obrim el terminal observarem que apareix una cadena de text similar a la següent:

```
usuari@nomMaquina:~$
```

Figura 3. Captura de pantalla d'un terminal sobre un fons de pantalla amb entorn gràfic

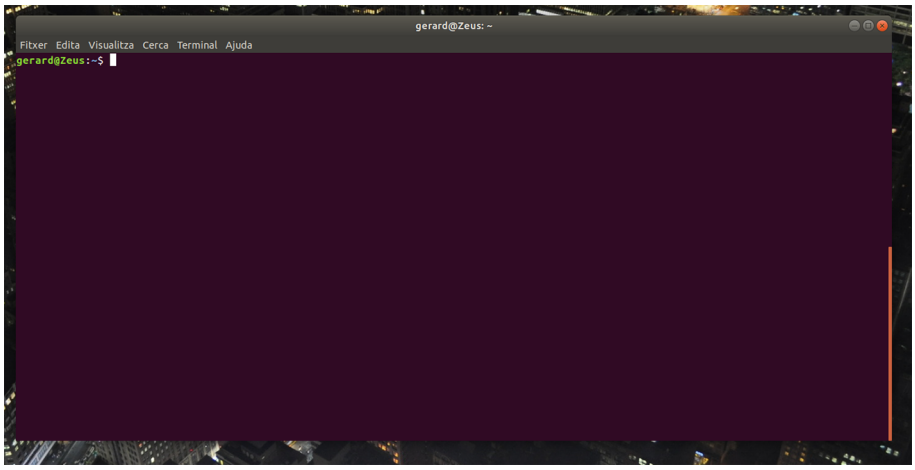


Figura 3

En aquest cas, podem observar que l'usuari és en Gerard i el nom de la màquina (hostname) és Zeus.

Això és el ***prompt***, que resta a l'espera que l'usuari introdueixi comandes i les executi amb la tecla de retorn. Aquest *prompt* conté el nom d'usuari, una arrova i el nom de la màquina (tècnicament s'anomena *hostname*). El programari que resta a l'espera i interpreta el que escriu l'usuari s'anomena ***shell*** i, en el marc d'aquesta assignatura, emprarem el programari **Bash**, malgrat n'hi ha altres.

Els terminals estan dissenyats per a executar les comandes una darrera de l'altra, sense la idea «d'anar cap endarrere». Quan el terminal estigui ple de lletres i es desitgi fer net, es pot emprar la comanda següent per a netejar-lo:

```
usuari@nomMaquina:~$ clear
```

Aquesta comanda es pot emprar tantes vegades com calgui i sense que tingui cap afectació.

Bash conté alguns trucs que permeten als usuaris guanyar velocitat en el teclat i no haver d'escriure sempre totes les comandes. En comentem alguns a continuació:

1) Amb la fletxa superior del teclat, es reescriuran les comandes que s'han executat anteriorment.

2) La tecla «tabulador» escriu de manera automàtica el nom dels programes o fitxers que estiguin disponibles. A continuació l'usuari escriu les lletres «ho» i prem el tabulador: apareixen totes les comandes que comencen per «ho». Si escriu una lletra més, l'«s», i prem de nou el tabulador, s'escriu solament la comanda que comença per «hos», en aquest cas *host*:

```
usuari@nomMaquina:~$ ho            (l'usuari prem el tabulador)
hoichess hoixiangqi host hostid hostname hostnamectl
usuari@nomMaquina:~$ host
```

Amb la comanda `history` es poden veure les darreres comandes executades:

```
usuari@nomMaquina:~$ history
 1987 gzip -d docs.tar.gz
 1988 tar -xvf docs.tar
```

```
 1989 exit
 1990 hostclear
 1991 clear
 1992 clea
 1993 clear
 1994 history
 1995 history | more
```

I es poden executar de nou emprant un identificador de la comanda (el número que l'acompanya), precedit d'un signe d'exclamació. Per exemple:

```
usuari@nomMaquina:~$ !1991
```

De manera similar, es poden executar comandes que ja s'han executat anteriorment de manera directa emprant el caràcter «!» amb les primeres lletres de la comanda que es desitgi executar. Per exemple:

```
usuari@nomMaquina:~$ !wge
```

Finalment, amb la combinació de tecles «control» + «r», apareix un cercador de comandes en què se li poden indicar lletres concretes de comandes també anteriors:

```
(reverse-i-search)'c': clear
```

4. Usuaris del sistema

El sistema operatiu pot integrar diversos usuaris en el seu sistema. Durant el procés d'instal·lació l'assistent sol·licitarà el nom d'usuari i la contrasenya per, almenys, un usuari, a més de la contrasenya per a l'usuari administrador (que, en entorns GNU/Linux s'anomena *root*). Així doncs, en un sistema operatiu estàndard, es disposarà de, com a mínim, dos usuaris, un usuari *root* i un altre usuari normal.

La informació de tots els usuaris s'allotja en el fitxer «/etc/password», que conté els camps següents, en què el caràcter «:» actua com a delimitador. La informació en concret que emmagatzema cada línia és la següent:

- el nom d'usuari,
- la contrasenya xifrada (en les distribucions actuals aquesta informació està en el fitxer «/etc/shadow», així que ja no resideix aquí),
- un número identificador d'usuari,
- l'identificador del grup al qual pertany l'usuari,
- el nom complet,
- la carpeta amb els fitxers de l'usuari (s'anomena tècnicament la *home*), i
- el programari que actuarà d'intèrpret de les comandes (en el cas d'aquesta assignatura /«bin/bash»).

Per exemple:

```
usuari@nomMaquina:~$ cat /etc/passwd | grep usuari
usuari:x:1000:1000:usuari,,,:/home/usuari:/bin/bash
```

Amb la comanda `id` podem conèixer l'identificador d'usuari, i també els grups als quals pertany l'usuari en qüestió:

```
usuari@nomMaquina:~$ id
uid=1000(usuari) gid=1000(usuari) grups=1000(usuari),4(adm),24(cdrom),27(sudo),30(dip),
46(plugdev),113(lpadmin),128(sambashare)
```

L'usuari administrador que disposa de tots els permisos del sistema s'anomena *root* i té l'identificador «0». És possible «suplantar» l'usuari administrador amb les comandes `sudo` i `su`. Mostrem a continuació un parell d'exemples:

```
usuari@nomMaquina:~$ sudo apt-get install xboard #Comanda que accedirà com a root
(amb autenticació prèvia i instal·larà per mitjà dels repositoris de programari el paquet
xboard). Aquesta comanda solament executarà una instrucció com a superusuari.
```

Recomanació

Es recomana a tots els usuaris que sempre emprin un usuari concret (per exemple, «usuari») i que no s'acostumin a treballar habitualment amb l'usuari *root*. Passa sovint però que un usuari concret ha d'executar comandes de les quals no en disposa permisos.

En el cas que requerim executar comandes diverses com a administrador, podem convertir-nos en aquest emprant la comanda:

```
usuari@nomMaquina:~$ sudo su
```

Una vegada finalitzades les tasques d'administració es pot retornar a l'usuari normal estàndard amb la comanda següent:

```
usuari@nomMaquina:~$ exit
```

5. El sistema de fitxers

El sistema operatiu GNU/Linux disposa d'un sistema de fitxers per a l'emmagatzematge de la informació i les dades, estructurat tot plegat per mitjà de carpetes i fitxers.

Contràriament als sistemes basats en Microsoft Windows, GNU/Linux no utilitza unitats tipus «C:», «D:» o «E:» per a referir-se a espais d'emmagatzematge, sinó que tota l'estructura penja d'una carpeta inicial (/). A partir d'aquí hi ha tot l'arbre de carpetes i subcarpetes. A continuació, mostrem algunes de les carpetes que acostumen a incloure la majoria de sistemes GNU/Linux:

Carpeta	Contingut
/bin	Conté els fitxers executables essencials per als usuaris (els binaris).
/boot	Conté els fitxers estàtics d'arrancada del sistema.
/dev	Conté els dispositius tipus ports USB, dispositius d'emmagatzematge, entre d'altres.
/etc	Conté fitxers de configuració de programari.
/home	Conté les carpetes dels usuaris. Cada usuari disposa d'una carpeta dins d'aquesta «home» de la qual n'és propietari i responsable (per exemple, a «/home/usuari» hi ha tots els fitxers de l'usuari «usuari»).
/lib	Conté llibreries compartides per al programari.
/lost+found	Conté fitxers recuperats.
/media o /mnt	Carpeta amb el punt de muntatge dels dispositius extraïbles.
/opt	Carpeta amb programari opcional.
/proc	Conté fitxers del *kernel* i informació dels processos que s'executen en el sistema.
/root	«home» de l'usuari root. Tal com hem comentat anteriorment, aquest usuari és l'administrador del sistema.
/sbin	Conté fitxers executables per a l'administració del sistema.
/tmp	Conté fitxers temporals.
/usr	Conté fitxers executables (binaris) dels usuaris.
/var	Conté fitxers de dades variables.

Veurem a continuació algunes primeres comandes de terminal per a navegar per mitjà d'aquesta estructura de carpetes.

`pwd`[4] és l'eina que mostra en quina carpeta resideix l'usuari en aquell moment. Es tracta d'una eina de caire solament informatiu, que mostrarà a l'usuari en quina carpeta està i no executa cap altra acció. Per exemple:

(4)Prové de l'acrònim *Print Working Directory*.

```
usuari@nomMaquina:~$ pwd
/home/usuari
```

`ls` és una eina per llistar els fitxers que hi ha en un directori. Les opcions més utilitzades són les següents:

- `a` mostrarà tots els fitxers (i és que els fitxers que s'inicien amb un punt . són ocults i no es veuen amb un `ls` normal).
- `l` proporciona detalls amb un format de llista llarg.
- `t` ordena per data de modificació, primer el més nou.
- `S` ordena les entrades per mida, de major a menor.
- `r` inverteix l'ordre.

Tota la informació de la comanda `ls` es pot consultar a la pàgina del manual de l'eina, accessible amb la comanda:

```
usuari@nomMaquina:~$ man ls
```

També és possible veure un resum dels paràmetres disponibles emprant la comanda:

```
usuari@nomMaquina:~$ ls --help
```

Alguns exemples senzills:

```
usuari@nomMaquina:~# ls
CCA0.aux CCA0.synctex.gz CCA0.tex CCA1.tex

usuari@nomMaquina:~# ls -l
total 16
-rw-r--r-- 1 root root 454 de ma 5 21:01 CCA0.aux
-rw-r--r-- 1 root root 3803 de ma 5 21:01 CCA0.synctex.gz
-rw-r--r-- 1 root root 3497 de ma 5 21:01 CCA0.tex
-rw-r--r-- 1 root root 3497 de ma 10 19:22 CCA1.tex
```

`cd` és l'eina per a canviar de la carpeta actual. Si l'usuari està a «/home/usuari/» i, executant `ls` observem que hi ha una carpeta que s'anomena «PACS», serà possible accedir-hi fent servir la comanda `cd PACS`.

Per a retornar a la carpeta anterior, caldrà executar `cd ..`

És possible accedir directament a la *home* de l'usuari fent servir la comanda `cd` sense emprar cap paràmetre. Per exemple:

```
usuari@nomMaquina:~/Projects$ pwd
/home/usuari/Projects
usuari@nomMaquina:~/Projects$ cd
```

```
usuari@nomMaquina:~$ pwd
/home/usuari
```

`mkdir`[5] és una eina per a crear carpetes. Amb el paràmetre `-p` és possible crear més d'una carpeta a la vegada. Mostrem un parell d'exemples:

(5)De l'anglès *make directory*.

```
usuari@nomMaquina:~$ mkdir dades
usuari@nomMaquina:~$ mkdir -p PACS/PAC1/
```

`cp` és una comanda que permet copiar fitxers. Els arguments que cal indicar a aquesta comanda són, en primer lloc, el fitxer origen (què es desitja copiar) i, en segon lloc, a on es desitja realitzar la còpia. Per exemple:

```
usuari@nomMaquina:~$ cp /etc/passwd /home/usuari/PACS/ #Copiarà el fitxer passwd ubicat
dins la carpeta etc a la carpeta /home/usuari/PACS.
```

`mv` és una comanda que serveix per a moure o reanomenar fitxers. Alguns exemples:

```
usuari@nomMaquina:~$ mv enunciatPAC1.doc respostesPAC1.doc #Aquesta comanda
canviarà el nom del fitxer.
usuari@nomMaquina:~$ mv respostesPAC1.doc PAC1 #Aquesta altra comanda mouria el fitxer
a la carpeta PAC1 (que ja existeix).
```

`rm`[6] és una comanda per a suprimir fitxers. Alguns exemples:

(6)Prové de *remove*.

```
usuari@nomMaquina:~$ rm respostesPAC1.doc #Aquesta comanda suprimirà el fitxer indicat.
usuari@nomMaquina:~$ rm -R PACS/ #Aquesta carpeta suprimirà de manera recursiva
tots els fitxers que hi ha dins la carpeta PACS.
```

`rmdir` és una comanda per a suprimir directoris buits.

```
usuari@nomMaquina:~$ rmdir PACS #Aquesta comanda suprimirà la carpeta PACS.
```

En el cas que la carpeta no estigui buida, caldrà suprimir prèviament tots els fitxers:

```
usuari@nomMaquina:~$ rm -R PACS
```

`touch` permet canviar la data d'un fitxer o crear un fitxer en blanc buit nou. Aquesta comanda canviarà la data d'accés al fitxer a l'actual. Si el fitxer al qual es refereix no existeix, es crearà un fitxer en blanc.

```
usuari@nomMaquina:~$ touch fitxer.txt
usuari@nomMaquina:~$ ls -l fitxer.txt
-rwxr-x--x 1 usuari usuari 0 de se 29 18:14 fitxer.txt
```

`df` mostra l'ús d'espai de disc utilitzat en cada partició del sistema. El paràmetre `-h` mostrarà les dades en megabytes o gigabytes (mostrarà una versió més llegible per humans).

```
usuari@nomMaquina:~$ df -h
S. fitxers Mida En ús Lliure %Ús Muntat a
udev 7,8G 0 7,8G 0% /dev
tmpfs 1,6G 2,0M 1,6G 1% /run
/dev/sdb1 459G 125G 311G 29% /
tmpfs 7,9G 81M 7,8G 2% /dev/shm
tmpfs 5,0M 4,0K 5,0M 1% /run/lock
```

```
tmpfs 7,9G 0 7,9G 0% /sys/fs/cgroup
tmpfs 1,6G 16K 1,6G 1% /run/user/127
tmpfs 1,6G 40K 1,6G 1% /run/user/1000
```

`ln` és una eina per a crear enllaços entre fitxers (es tracta del concepte d'accessos directes de Windows). Acostumarem a realitzar *soft links*. Per exemple:

```
usuari@nomMaquina:~$ ln -s /etc/passwd contrasenyes.txt #Aquesta comanda crearà a la carpeta
actual un fitxer contrasenyes.txt que serà un accés directe a /etc/passwd.
usuari@nomMaquina:~$ ls -l
lrwxrwxrwx 1 usuari usuari 11 de ju 22 11:13 contrasenyes.txt -> /etc/passwd
```

El primer caràcter de la línia anterior és una «l», que indica que es tracta d'un enllaç (*link*).

`chown` és l'eina que cal utilitzar per a canviar el propietari d'un fitxer. Cada fitxer pertany a un propietari i també a un grup propietari. Examinem l'exemple següent:

```
usuari@nomMaquina:~$ touch hola.txt #Amb aquesta comanda crearem un fitxer hola.txt.
usuari@nomMaquina:~$ ls -l hola.txt #I, amb aquesta altra comanda, veurem qui és el propietari.
-rw-r--r-- 1 usuari usuari 0 de ju 22 11:15 hola.txt
```

El primer «usuari» es refereix a l'usuari propietari del fitxer. El segon es refereix al grup propietari. En els fitxers «/etc/passwd» es poden veure el llistat d'usuaris existents en un sistema. En el fitxer «/etc/group» hi ha el llistat dels grups. En aquest cas concret, hi ha un grup que s'anomena igual que l'usuari, encara que no sempre és així.

Amb la comanda següent establirem el canvi de propietari i de grup propietari del fitxer «hola.txt». Per a traspassar un fitxer per tal que sigui propietat de *root*, caldrà emprar l'eina `sudo`, que permet executar comandes com si fóssim *root*.

```
usuari@nomMaquina:~$ sudo chown root:root hola.txt
usuari@nomMaquina:~$ ls -l hola.txt
-rw-r--r-- 1 root root 0 de ju 22 11:15 hola.txt
```

`chmod` és l'eina emprada per a canviar els permisos de fitxers. En entorns GNU/Linux hi ha tres permisos i tres tipus d'usuari:

Permisos	Usuaris
«r» de lectura	«u», el propietari del fitxer
«w» d'escriptura	«g», el grup propietari
«x» d'execució	«o» per a la resta

Per un fitxer «executar» significa poder executar el programa que hi ha dins. Per un directori, el permís d'execució és necessari per a fer qualsevol cosa dins de la carpeta.

```
usuari@nomMaquina:~$ ls -l script.sh
-rw-r--r-- 1 usuari usuari 0 de ju 22 11:15 script.sh
```

Amb aquesta comanda observem els permisos d'aquest hola.txt. L'usuari *root* pot fer `rw-` (pot llegir i escriure però no executar, el grup propietari pot solament llegir `r--` i, finalment, la resta d'usuaris, poden solament llegir `r--`). El primer guió serveix per a conèixer de quin tipus de fitxer es tracta (les carpetes contenen una «d» i els enllaços una «l», tal com hem vist prèviament).

Per a atorgar permís d'execució podem realitzar el següent:

```
usuari@nomMaquina:~$ chmod u+x script.sh
```

I, ara:

```
usuari@nomMaquina:~$ ls -l script.sh
-rwxr--r-- 1 usuari usuari 0 de ju 22 11:15 script.sh
```

El paràmetre `u` de la comanda `chmod` es referia a l'usuari. De manera similar, podríem haver emprat `g` per al grup i `o` per als altres.

Hi ha una altra manera d'emprar la comanda `chmod`. Considerem que els permisos consisteixen en tres nombres (observem que hi ha tres permisos per tres objectes -usuari, grup i la resta). Considerem ara aquests permisos en binari (un «0» si no se li atorga i un «1» si és el cas). De binari, traspassem a decimal.

Per exemple:

- 7, en binari, s'escriu, 111.
- 5, en binari, s'escriu, 101.
- 1, en binari, s'escriu 001.

És possible establir permisos emprant aquesta codificació:

```
usuari@nomMaquina:~$ chmod 751 fitxer.txt
usuari@nomMaquina:~$ ls -l fitxer.txt
-rwxr-x--x 1 usuari usuari 0 de se 16 06:44 fitxer.txt
```

En aquest cas, l'usuari «usuari» pot realitzar totes les operacions (llegir, escriure i executar), que es llegiria com a 111 i en binari correspon al número 7, el grup «usuari» solament llegir i executar (`r-w`), que en binari es llegiria 101 i correspon al número 5 i, finalment, la resta d'usuaris solament executar (`--x`), que en binari es llegiria 001 i correspon al nombre 1.

6. Cerca de fitxers

Per a cercar fitxers en el sistema, esmentarem un parell de comandes. La primera d'aquestes és la comanda `find` que, indicant com a paràmetre una ruta, realitzarà la cerca de fitxers que continguin certs patrons. L'esquema de la comanda és tal com segueix a continuació:

```
find opcions ruta_de_recerca expressió
```

En l'exemple que mostrem a continuació, es realitzarà una recerca del fitxer «testfile.txt» a partir de la ruta actual i de totes les subcarpetes presents en la carpeta:

```
usuari@nomMaquina:~$ find . -name testfile.txt
```

En aquest altre exemple, és el mateix però parteix de la carpeta «/var/www/» i cerca solament fitxers que finalitzin amb l'extensió «.html»:

```
usuari@nomMaquina:~$ find /var/www/ -name "*.html"
```

En l'exemple que mostrarem a continuació hi ha tres paràmetres nous. El paràmetre `user` indica l'usuari propietari del fitxer. El paràmetre `mtime` indica el nombre de dies que fa que el fitxer va ser modificat i el paràmetre `iname` és el mateix que `name` però fent que el text no sigui sensible entre minúscules i majúscules:

```
usuari@nomMaquina:~$ find /home -user exampleuser -mtime 7 -iname ".db"
```

Per tant, aquesta comanda cercaria fitxers modificats durant els darrers set dies, propietat de l'usuari exampleuser i que acabessin amb l'extensió «.db», sigui en minúscules o majúscules.

Amb el paràmetre `-delete` seria possible suprimir els fitxers que s'haguessin trobat:

```
usuari@nomMaquina:~$ find . -name "*.bak" -delete
```

La comanda `locate` és més ràpida cercant fitxers que la comanda anterior, ja que, en realitat, no els cerca, sinó que mira en una base de dades. Hi ha però dos desavantatges en aquesta comanda:

- No és tant flexible com `find` (que permet cercar per nom, dates de modificació, entre d'altres).
- Parteix d'una base de dades que cal construir i actualitzar sovint.

L'ús però de la comanda és molt senzill i és molt ràpid.

La comanda per a la creació de la base de dades s'ha de realitzar com a *root*:

```
usuari@nomMaquina:~$ sudo updatedb
```

Aquesta comanda cal executar-la sovint. Idealment es posa en el Cron per tal que s'executi de manera automàtica.

Amb la base de dades construïda ja és possible cercar fitxers:

```
usuari@nomMaquina:~$ locate mkpasswd
/usr/bin/grub-mkpasswd-pbkdf2
/usr/bin/mkpasswd
/usr/share/man/man1/grub-mkpasswd-pbkdf2.1.gz
/usr/share/man/man1/mkpasswd.1.gz
```

7. Manipulació de fitxers

Tractarem en aquest apartat el tema de la manipulació de fitxers per mitjà de les eines d'un terminal amb GNU/Linux.

Començarem primer indicant en què consisteixen els **descriptors de fitxers**: es tracta d'enters positius que representen fitxers oberts.

En el cas que el sistema disposi de cinquanta fitxers oberts, hi hauran cinquanta descriptors que apuntin a aquests. Com que en els sistemes operatius basats amb Unix tot és un fitxer, els processos incorporen un **descriptors «especials»**:

Descriptor	Tasca	Operador
stdin	Un descriptor de fitxer que representa l'entrada de dades dels processos i al qual se li assigna l'enter 0.	<
stdout	Un descriptor de fitxer que apunta on es publiquen els resultats de les execucions i al qual se li assigna l'enter 1.	>
stderr	Un descriptor que apunta al fitxer on s'incorporen els errors i al qual se li assigna el 2.	2>

Operadors

Amb els operadors és possible sobreescriure els descriptors.

En l'exemple següent el resultat de la comanda `ls` es redirigeix a un fitxer extern:

```
usuari@nomMaquina:~$ ls -l > fitxers.txt
```

Cal tenir en compte que, en el cas que «fitxers.txt» ja existeixi, se sobreescriurà i es perdrà la informació que contenia. En el cas que es desitgi agregar informació a l'existent, caldria emprar l'operador >>.

Per exemple, la comanda següent agregaria a «fitxers.txt» el resultat de la comanda `pwd`:

```
usuari@nomMaquina:~$ pwd >> fitxers.txt
```

L'entrada estàndard també es pot modificar emprant el caràcter <. Per exemple:

```
usuari@nomMaquina:~$ sort < llistat.txt
```

L'entrada de la comanda `sort` prové del fitxer «llistat.txt».

Hi ha un altre operador d'interès per a comentar en aquest apartat. Són els ***pipes*** que enllacen el resultat d'una comanda (`stdout`) amb l'entrada d'una altra (`stdin`). Aquests *pipes* es representen amb el caràcter «|». En l'exemple següent la primera comanda mostrarà tots els processos existents en el sistema. El resultat d'aquesta comanda s'enllaçarà amb un *grep* que filtrarà les línies que contenen la paraula «firefox»:

```
usuari@nomMaquina:~$ ps -a | grep firefox
```

Vist aquest tema dels descriptors dels fitxers i redireccions, vegem algunes comandes de terminal útils per a la gestió de fitxers.

`cat` és una comanda que permet mostrar el contingut d'un fitxer en un terminal. Per exemple:

```
usuari@nomMaquina:~$ cat /etc/passwd
```

`head` és una comanda que permet mostrar solament les primeres línies d'un fitxer. La comanda següent, per exemple, mostrarà solament les primeres set línies del fitxer «/etc/passwd»:

```
usuari@nomMaquina:~$ head -7 /etc/passwd
```

`tail` és una comanda similar a l'anterior, però que mostra solament les darreres línies. Per a veure les set darreres línies del mateix fitxer anterior, és possible executar:

```
usuari@nomMaquina:~$ tail -7 /etc/passwd
```

Combinant les comandes `head`, `tail` i unint-les emprant *pipes*, podem realitzar tasques com les de visualitzar solament una línia en concret. Per exemple:

```
usuari@nomMaquina:~$ head -12 /etc/passwd | tail -1
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
```

En la primera comanda, `head -12 /etc/passwd` mostra les 12 primeres línies del fitxer indicat encara que, la sortida no surt a `sdtout` sinó que es redirigeix a l'entrada de la comanda `tail`, que mostrarà solament la darrera línia.

`cut` és una comanda que permet extreure camps d'un fitxer de text delimitat per caràcters concrets. Alguns exemples:

El fitxer «/etc/password» conté diversos camps separats pel caràcter «:». L'opció `-d` de la comanda `cut` especifica aquest delimitador. L'opció `-f` indica quins camps volem mostrar. La comanda següent mostrarà els camps 6 i 7 d'aquest fitxer:

```
usuari@nomMaquina:~$ cut -d":" -f6,7 /etc/passwd
/root:/bin/bash
/usr/sbin:/usr/sbin/nologin
/home/usuari:/bin/bash
/home/sandra:/bin/bash
```

Aquest altre exemple mostrarà solament del caràcter 1 al caràcter 9 del fitxer «/etc/hosts».

```
usuari@nomMaquina:~$ cut -c 1-9 /etc/hosts
127.0.0.1
127.0.1.1
# The fol
::1 i
fe00::0 i
ff00::0 i
ff02::1 i
ff02::2 i
```

`tr` és una comanda que serveix per a substituir, suprimir o restringir alguns caràcters concrets. Vegem alguns exemples. En el primer, la comanda es prepara per a realitzar substitucions:

```
usuari@nomMaquina:~$ tr abcdefghijklmnopqrstuvwxyz ABCDEFGHIJKLMNOPQRSTUVWXYZ
grau de dades
GRAU DE DADES
```

O aquest altre que suprimeix la lletra 'a':

```
usuari@nomMaquina:~$ echo "grau de dades" | tr -d 'a'
gru de dds
```

`wc` és una comanda que recompta el nombre de línies, de paraules o de bytes que conté un fitxer determinat.

```
usuari@nomMaquina:~$ wc /etc/passwd
 49 85 2787 /etc/passwd
```

Així doncs veiem que aquest fitxer conté 49 línies, 85 paraules i ocupa 2.787 bytes. Hi ha paràmetres que permeten veure alguns camps concrets. Per exemple, aquest, que mostra solament el nombre de línies:

```
usuari@nomMaquina:~$ wc -l /etc/passwd
49 /etc/passwd
```

`tee` és una comanda que copia informació de l'entrada estàndard a l'`stdout`, però també en un fitxer extern. Així doncs, duplica la informació de sortida. La comanda següent mostra en format llista els fitxers dins la carpeta «/etc/» que comencen per *host*. Traspassa aquest llistat per mitjà d'un *pipe* a la comanda `wc` que recomptarà el nombre de línies i, el resultat d'aquesta comanda la traspassa a `tee`, que emmagatzemarà aquest resultat en el fitxer «nombre.txt», però que també el mostrarà per pantalla.

```
usuari@nomMaquina:~$ ls -l /etc/host* | wc -l | tee nombre.txt
5
usuari@nomMaquina:~$ more nombre.txt
5
```

`paste` és una comanda que serveix per a mesclar línia per línia el contingut de diversos fitxers amb un resultat de sortida. Vegem un exemple:

```
usuari@nomMaquina:~$ cat n.txt
1
2
```

```
3
4

usuari@nomMaquina:~$ cat a.txt
Gerard
Sandra
Joel
Julià

usuari@nomMaquina:~$ paste n.txt a.txt
1   Gerard
2   Sandra
3   Joel
4   Julià
```

`sort` és una comanda que serveix per a ordenar el contingut de diversos fitxers. Es poden ordenar els fitxers de manera ascendent o descendent. Vegem un parell d'exemples.

La comanda següent ordenarà el contingut del fitxer especificat a partir del 5è. camp, en què els diferents camps són definits pel caràcter «:».

```
usuari@nomMaquina:~$ sort -k 5 -t ":" /etc/passwd
```

Com comentàvem, també és possible realitzar l'ordre invers:

```
usuari@nomMaquina:~$ paste n.txt a.txt | sort -r
4   Julià
3   Joel
2   Sandra
1   Gerard
```

`uniq` és una comanda que pot servir per a reportar o filtrar les línies repetides en un fitxer. Cal parar atenció, perquè la comanda examina solament les línies adjacents (línies que estan l'una sota l'altra):

```
usuari@nomMaquina:~$ more a.txt
Gerard
Gerard
Sandra
Joel
Julià

usuari@nomMaquina:~$ uniq a.txt
Gerard
Sandra
Joel
Julià

usuari@nomMaquina:~$ uniq -c a.txt
 2 Gerard
 1 Sandra
 1 Joel
 1 Julià
```

`file` és una comanda que serveix per a obtenir informació sobre el tipus de fitxer. Mostrem alguns exemples:

```
usuari@nomMaquina:~$ file /etc/passwd
/etc/passwd: ASCII text

usuari@nomMaquina:~$ file generaWeb.sh
generaWeb.sh: Bourne-Again shell script, ASCII text executable
```

```
usuari@nomMaquina:~$ file /usr/share/tuxpaint/templates/rocks.jpg
/usr/share/tuxpaint/templates/rocks.jpg: JPEG image data, JFIF standard 1.01, resolution (DPI),
density 72x72, segment length 16, baseline, precision 8, 1296x864, frames 3
```

Amb el paràmetre `-i` la comanda `file` mostrarà informació sobre la codificació del fitxer.

```
usuari@nomMaquina:~$ file -i /etc/passwd
/etc/passwd: text/plain; charset=us-ascii
```

`iconv` és una comanda que permet convertir la codificació d'un fitxer de text. Per exemple, en el cas que es desitgés convertir un fitxer de text escrit en codificació d'UTF-8 i traspassar-la a codificació ASCII.

```
usuari@nomMaquina:~$ iconv -f ISO-8859-1 -t UTF-8//TRANSLIT input.file -o out.file
```

Amb el paràmetre `-l` és possible veure tot un llistat de codificacions de caràcters (trunquem la sortida per no ocupar massa espai):

```
usuari@nomMaquina:~$ iconv -l
La llista següent conté tots els jocs de caràcters codificats coneguts. Això no significa
necessàriament que totes les combinacions d'aquests noms siguin possibles en els paràmetres
d'origen i destí de la línia d'ordres. Un joc de caràcters codificat pot estar llistat
amb noms diferents (àlies).

 437, 500, 500V1, 850, 851, 852, 855, 856, 857, 858, 860, 861, 862, 863, 864, 865, 866, 866NAV,
 869, 874, 904, 1026, 1046, 1047, 8859_1, 8859_2, 8859_3, 8859_4, 8859_5, 8859_6, 8859_7,
 8859_8, 8859_9, 10646-1:1993, 10646-1:1993/UCS4, ANSI_X3.4-1968, ANSI_X3.4-1986, ANSI_X3.4,
 ANSI_X3.110-1983, ANSI_X3.110, ARABIC, ARABIC7, ARMSCII-8, ASCII, ASMO-708, ASMO_449, BALTIC,
 BIG-5, BIG-FIVE, BIG5-HKSCS, BIG5, BIG5HKSCS, BIGFIVE, BRF, BS_4730, CA, CN-BIG5, CN-GB, CN,
 CP-AR, CP-GR, CP-HU, CP037, CP038, CP273,
```

`grep` és una comanda que processa text línia per línia cercant patrons (expressions regulars) i mostrant solament les línies que compleixen aquests patrons. Per exemple:

```
usuari@nomMaquina:~$ grep localh /etc/hosts
127.0.0.1   localhost
::1 ip6-localhost ip6-loopback
```

`jq` és una comanda de terminal per a processar fitxers en format JSON. Probablement caldrà instal·lar aquest programari a part, ja que no és habitual trobar-lo en una instal·lació estàndard. Més endavant, en aquest mateix document, es mostra com dur a terme aquesta tasca. Un programa amb Jq conté un filtre que, a partir d'una entrada, produeix una sortida.

Expressions regulars

L'aprenentatge d'expressions regulars s'escapa de l'abast d'aquest document, encara que es recomana a l'alumnat el seu aprenentatge.

Imaginem que disposem de l'exemple següent:

```
usuari@nomMaquina:~$ more f.json
["Arxiu Fotogr\u00e0fic de Barcelona",["Arxiu Fotogr\u00e0fic de Barcelona"],
["L'Arxiu Fotogr\u00e0fic de Barcelona (AFB) \u00e9s l'arxiu dedicat a la recollida,
conservaci\u00f3, organitzaci\u00f3 i difusi\u00f3 dels fons fotogr\u00e0fics de car\u00e0cter
hist\u00f2ric generats per l'Ajuntament de Barcelona a conseq\u00fc\u00e8ncia de la seva activitat,
aix\u00ed com d'aquells fons i col\u00b7leccions fotogr\u00e0fiques, de car\u00e0cter
no municipal, d'inter\u00e8s per a la hist\u00f2ria de la ciutat i per a la hist\u00f2ria
de la fotografia."],["https://ca.wikipedia.org/wiki/Arxiu_Fotogr%C3%A0fic_de_Barcelona"]]
```

Amb la comanda `jq` podem obtenir camps concrets d'aquest fitxer en format JSON. Per exemple:

```
usuari@nomMaquina:~$ jq .[2] f.json
[
L'Arxiu Fotogràfic de Barcelona (AFB) és l'arxiu dedicat a la recollida, conservació,
organització i difusió dels fons fotogràfics de caràcter històric generats per l'Ajuntament
de Barcelona a conseqüència de la seva activitat, així com d'aquells fons i col·leccions
fotogràfiques, de caràcter no municipal, d'interès per a la història de la ciutat
i per a la història de la fotografia.
]
usuari@nomMaquina:~$ jq .[3] f.json
[
 "https://ca.wikipedia.org/wiki/Arxiu_Fotogr%C3%A0fic_de_Barcelona"
]
```

`xmlstarlet` és un conjunt de comandes de terminal que es poden usar per a transformar, seleccionar, validar o editar fitxers XML.[7] Es tracta d'eines que poden fer quelcom similar a la comanda `grep`, però en fitxers en format XML. Per mitjà de la comanda següent es mostrarien els noms d'aquests *xpaths* del fitxer «museus.xml».

(7) De l'anglès *Extensible Markup Language*.

```
usuari@nomMaquina:~$ xmlstarlet sel -t -v "/xml/dadesMuseus/Equipament" museus.xml
```

Aquí s'ha ofert una descripció molt i molt breu de les darreres tres comandes (`grep`, `jq` i `xmlstarlet`), però aprofundir-hi més surt de l'objectiu d'aquest document d'introducció a GNU/Linux. S'han esmentat solament com a eines disponibles, encara que l'aprenentatge per al seu ús és complicat i es deixa per altres espais de documentació.

8. Compressió de fitxers

És habitual en sistemes informàtics comprimir un conjunt de fitxers, sigui per a emmagatzemar-los en còpies de seguretat, traspassar-los per xarxa o el que sigui. En entorns GNU/Linux és habitual que diversos fitxers s'empaquetin en un de sol primer i, després, es comprimeixin.

Windows

En sistemes Windows normalment hi ha solament una operació: es genera un sol fitxer «.zip» a partir de fitxers diversos i ja està.

La comanda emprada per a empaquetar o desempaquetar fitxers serà la utilitat `tar`. Per a comprimir i descomprimir hi ha comandes diverses, encara que les més habituals són les següents: `gzip`, `bzip2`, `zip` i `unzip`.

Mostrem a continuació alguns exemples. En la primera comanda, s'empaqueta en un fitxer «docs.tar» tot el contingut de la carpeta «Documents». El paràmetre `-cf` cal llegir-lo com a «crear fitxer». Després, es comprimeix en format zip:

```
usuari@nomMaquina:~$ tar -cf docs.tar Documents/
usuari@nomMaquina:~$ ls docs.tar
docs.tar
usuari@nomMaquina:~$ gzip docs.tar
usuari@nomMaquina:~$ ls *.tar.gz
docs.tar.gz
```

Ara fem el procés contrari: a partir d'un fitxer «.tar.gz» el descomprimim. El paràmetre `-d` a `gzip` és per a descomprimir. El paràmetre `-xvf` serveix per a extreure un fitxer.

```
usuari@nomMaquina:~$ gzip -d docs.tar.gz
usuari@nomMaquina:~$ tar -xvf docs.tar
Documents/
```

Hi ha la possibilitat d'efectuar ambdues operacions d'una manera molt elegant emprant un *pipe*:

```
usuari@nomMaquina:~$ gzip -dc docs.tar.gz | tar -xv
```

9. Variables d'entorn

En els sistemes GNU/Linux hi ha **variables d'entorn** per a emmagatzemar dades i opcions de configuració que depenen de l'usuari.

Per a conèixer el llistat de variables d'entorn disponibles podem executar les comandes `env` o `printenv`. En l'exemple següent hem truncat la informació per a facilitar-ne la lectura.

```
usuari@nomMaquina:~$ env
LANG=ca_ES.UTF-8
DISPLAY=:1
GNOME_SHELL_SESSION_MODE=ubuntu
GTK2_MODULES=overlay-scrollbar
COLORTERM=truecolor
USERNAME=usuari
XDG_VTNR=2
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path
XDG_SESSION_ID=4
USER=usuari
DESKTOP_SESSION=ubuntu
QT4_IM_MODULE=xim
TEXTDOMAINDIR=/usr/share/locale/
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/7e2d77d9_3137_4a30_8957_2f5409826d1e
DEFAULTS_PATH=/usr/share/gconf/ubuntu.default.path
QT_QPA_PLATFORMTHEME=appmenu-qt5
PWD=/home/usuari
HOME=/home/usuari
TEXTDOMAIN=im-config
SSH_AGENT_PID=3019
QT_ACCESSIBILITY=1
XDG_SESSION_TYPE=x11
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share:/usr/share:/var/lib/snapd/desktop
XDG_SESSION_DESKTOP=ubuntu
GJS_DEBUG_OUTPUT=stderr
GTK_MODULES=gail:atk-bridge
WINDOWPATH=2
TERM=xterm-256color
SHELL=/bin/bash
VTE_VERSION=5202
QT_IM_MODULE=ibus
XMODIFIERS=@im=ibus
IM_CONFIG_PHASE=2
XDG_CURRENT_DESKTOP=ubuntu:GNOME
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
GNOME_TERMINAL_SERVICE=:1.104
XDG_SEAT=seat0
SHLVL=1
GDMSESSION=ubuntu
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
LOGNAME= usuari
PATH=/home/usuari/bin:/home/usuari/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/
bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
SESSION_MANAGER=local/Zeus:@/tmp/.ICE-unix/2911,unix/Zeus:/tmp/.ICE-unix/2911
LESSOPEN=| /usr/bin/lesspipe %s
GTK_IM_MODULE=ibus
_=/usr/bin/env
```

Entre d'altres, les quatre variables d'entorn que considerem de més interès són les següents:

Variables d'entorn	Descripció
HOSTNAME	Conté el nom de l'ordinador.
HOME	Conté el directori de l'usuari.
PATH	Conté els directoris que es recorreran per a la recerca de les comandes.
EDITOR	Conté el nom de l'editor de text per defecte.

És possible veure el valor d'aquestes variables d'entorn emprant la comanda `echo` i agregant el caràcter «$» al nom de la variable. Per exemple:

```
usuari@nomMaquina:~$ echo $HOME
/home/usuari

usuari@nomMaquina:~$ echo $PATH
/home/usuari/bin:/home/usuari/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/
sbin:/bin:/usr/games:/usr/local/games:/snap/bin
```

Sovint pot ser necessari canviar o agregar informació a aquestes variables d'entorn. Per exemple, en el cas que desitgéssim agregar una nova carpeta a la variable d'entorn **PATH**. Per a realitzar aquesta operació caldrà emprar la comanda `export`:

```
usuari@nomMaquina:~$ export PATH=$PATH:/home/usuari/novacarpeta
```

Aquesta comanda configura la variable d'entorn **PATH** amb la informació que ja contenia i també hi agrega aquesta nova carpeta «/home/usuari/novacarpeta».

De manera similar, podem crear noves variables:

```
usuari@nomMaquina:~$ export VARIABLE="hola !"
usuari@nomMaquina:~$ echo $VARIABLE
hola !
```

En el cas que desitgem que aquestes noves variables siguin persistents (que no es perdin quan es reinicïi l'ordinador), caldrà que les emmagatzemem en fitxers. En concret, podem agregar les comandes al final del fitxer «$HOME/.bashrc»:

```
usuari@nomMaquina:~$ cd $HOME
usuari@nomMaquina:~$ tail -2 .bashrc
export VARNOVA=" adeu !"
usuari@nomMaquina:~$ echo $VARNOVA
adeu !
```

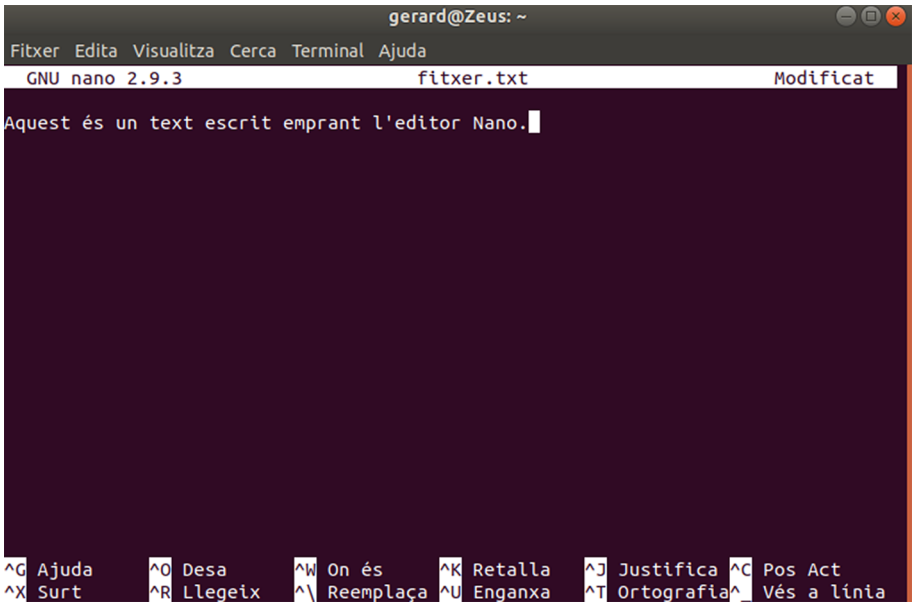
10. Editors de text

És important saber crear fitxers de text per mitjà del terminal. Hi ha diversos editors. Un dels més antics i que es manté en la majoria de terminals és el **Vi**, encara que es desaconsella en el marc d'aquesta assignatura. Recomanem, per la seva simplicitat, la utilitat **Nano**. Així doncs, per a l'edició d'un fitxer de text caldrà solament executar la comanda:

```
usuari@nomMaquina:~$ nano fitxer.txt
```

Una vegada dins de l'editor de text s'observarà que les comandes de control es realitzen de manera conjunta amb les tecles «control» + una tecla concreta. Per exemple, per a desar el fitxer es fa amb «control+o» i, per a sortir, «control+x». En aquest cas concret, la distinció entre minúscules i majúscules no són importants. Excepte per aquestes combinacions de tecles, per la resta s'escriu normalment, tal com es faria amb altres editors de text.

Figura 4. Captura de pantalla en què s'observa la interfície de l'editor de textos Nano



11. Processos amb GNU/Linux

Quan l'usuari executa un programari es crea un **procés**.

Els sistemes operatius actuals són capaços d'executar múltiples processos a la vegada (amb tècniques diverses de planificació de la CPU). Però aquí la idea principal és que cada programa és un procés que s'identifica amb un número (el PID -identificador del procés). Exemplificarem aquest apartat amb comandes.

La comanda `sleep` executa un procés que no fa res durant els segons indicats, en aquest cas 10 segons. Observem que, mentre s'executa aquest procés, l'usuari no pot fer cap altra operació mentre no finalitzi aquest.

```
usuari@nomMaquina:~$ sleep 10
```

Es tracta d'una execució en primer pla. És possible executar el mateix procés però «de fons» (el que tècnicament s'anomena en *background*):

```
usuari@nomMaquina:~$ sleep 10 &
[1] 9244
```

Quan s'executa un procés amb aquesta modalitat, es mostrarà l'identificador del procés que se li haurà assignat, en aquest darrer cas, el 9244. És possible veure els processos que estan en marxa en una màquina amb les comandes `top` i `ps`:

```
usuari@nomMaquina:~$ ps
 PID TTY TIME CMD
 6306 pts/0 00:00:00 bash
 7671 pts/0 00:00:00 ps
usuari@nomMaquina:~$ sleep 10 &
[1] 7672
usuari@nomMaquina:~$ ps
 PID TTY TIME CMD
 6306 pts/0 00:00:00 bash
 7672 pts/0 00:00:00 sleep
 7675 pts/0 00:00:00 ps
```

La comanda `ps` pot mostrar informació diversa en funció dels paràmetres que se li assignin. Per exemple:

Paràmetre	Informació
UID	Identificador d'usuari al qual pertany el procés (o sigui, la persona que l'està executant).
PID	Identificador del procés.

Paràmetre	Informació
PPID	Identificador del procés pare (en el cas que es tracti d'un altre procés que l'hagi començat).
C	Grau d'utilització de la CPU per part del procés.
STIME	Data d'inici del procés.
TTY	Tipus de terminal associat al procés.
TIME	Temps de CPU emprada pel procés.
CMD	Comanda que s'ha emprat per a iniciar el procés.

La comanda `ps` disposa de diversos paràmetres disponibles. Entre d'altres:

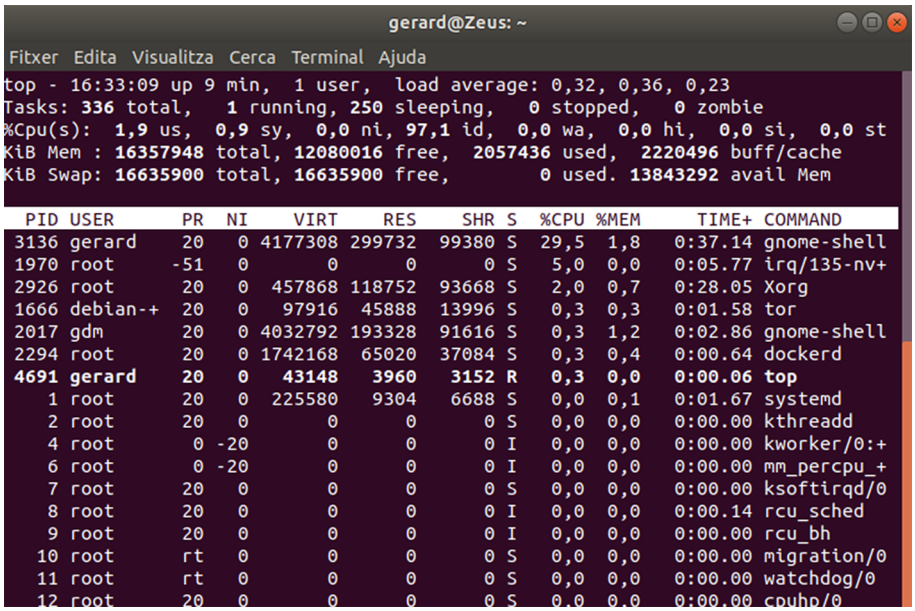
- **a**: Mostra informació de tots els usuaris.
- **x**: Mostra informació dels processos sense els terminals.
- **u**: Mostra informació addicional.

La comanda `top` és una eina d'administració del sistema útil per a conèixer l'activitat del processador, saber quins processos estan en marxa i quants recursos ocupen. Per a executar-la cal simplement escriure en un terminal:

```
usuari@nomMaquina:~$ top
```

En la pantalla de la figura 5 s'observa el llistat de processos actuals, quin tant per cent de memòria RAM ocupa cadascun, el grau d'utilització de la CPU, la comanda en concret que ha generat el procés, la prioritat amb la qual s'executa, entre d'altres. Per a sortir de la comanda cal prémer la tecla «q».

Figura 5. Captura de pantalla en què s'observa el resultat de l'execució de la comanda `top`



També és possible filtrar quins processos observar, per exemple, els d'un usuari en concret:

```
usuari@nomMaquina:~$ top -u usuari
```

I, fins i tot, generar un fitxer de text pla amb la informació que genera aquesta comanda:

```
usuari@nomMaquina:~$ top -n 1 -b > sortida-top.txt
```

Als processos se'ls pot enviar senyals diverses, per exemple, senyals per tal que s'aturin. Quan s'executa un procés en primer pla, amb la combinació de les tecles «control + c» s'envia una senyal per tal que el procés pari. Quan el procés s'executa en segon pla (en *background*), en canvi, caldrà emprar la comanda `kill`. Vegem uns exemples:

```
usuari@nomMaquina:~$ sleep 10 &
[1] 7734
```

Aquest 7734 es refereix al procés d'identificador del procés (el PID) i, de fet, el podem veure executant un *process status*:

```
usuari@nomMaquina:~$ ps
 PID TTY TIME CMD
 6306 pts/0 00:00:00 bash
 7734 pts/0 00:00:00 sleep
 7735 pts/0 00:00:00 ps
```

A continuació, enviarem una senyal amb la comanda `kill` al procés en concret, per tal de demanar-li que s'aturi:

```
usuari@nomMaquina:~$ kill -TERM 7734
usuari@nomMaquina:~$ ps
 PID TTY TIME CMD
 6306 pts/0 00:00:00 bash
 7736 pts/0 00:00:00 ps
[1]+ Terminat sleep 10
```

A vegades hi ha processos que, pel motiu que sigui, no s'aturen. Hi ha altres senyals menys amigables. En concret, aquesta de `KILL`:

```
usuari@nomMaquina:~$ kill -KILL 7734
```

Hi ha diverses senyals que podem trametre als processos encara que, les més emprades habitualment són les dues que hem comentat.

És possible veure el llistat de senyals disponibles emprant la comanda següent:

```
usuari@nomMaquina:~$ kill -l
 1) SIGHUP     2) SIGINT     3) SIGQUIT    4) SIGILL     5) SIGTRAP
 6) SIGABRT    7) SIGBUS     8) SIGFPE     9) SIGKILL   10) SIGUSR1
11) SIGSEGV   12) SIGUSR2   13) SIGPIPE   14) SIGALRM   15) SIGTERM
16) SIGSTKFLT 17) SIGCHLD   18) SIGCONT   19) SIGSTOP   20) SIGTSTP
21) SIGTTIN   22) SIGTTOU   23) SIGURG    24) SIGXCPU   25) SIGXFSZ
26) SIGVTALRM 27) SIGPROF   28) SIGWINCH  29) SIGIO     30) SIGPWR
31) SIGSYS    34) SIGRTMIN  35) SIGRTMIN+1   36) SIGRTMIN+2   37) SIGRTMIN+3
38) SIGRTMIN+4   39) SIGRTMIN+5   40) SIGRTMIN+6   41) SIGRTMIN+7   42) SIGRTMIN+8
43) SIGRTMIN+9   44) SIGRTMIN+10  45) SIGRTMIN+11  46) SIGRTMIN+12  47) SIGRTMIN+13
```

```
48) SIGRTMIN+14  49) SIGRTMIN+15  50) SIGRTMAX-14  51) SIGRTMAX-13  52) SIGRTMAX-12
53) SIGRTMAX-11  54) SIGRTMAX-10  55) SIGRTMAX-9   56) SIGRTMAX-8   57) SIGRTMAX-7
58) SIGRTMAX-6   59) SIGRTMAX-5   60) SIGRTMAX-4   61) SIGRTMAX-3   62) SIGRTMAX-2
63) SIGRTMAX-1   64) SIGRTMAX
```

La comanda `killall` permet matar els processos mitjançant el seu nom, sense haver de conèixer el seu identificador:

```
usuari@nomMaquina:~$ sleep 10 &
[1] 7790
usuari@nomMaquina:~$ killall sleep
[1]+ Terminat sleep 10
```

També hi ha la comanda `nohup` que permet executar una altra comanda instruint al sistema que la continuï executant fins i tot quan es desconnecti la sessió. Aquest «nohup» prové de «no hangup», que es refereix a una senyal de sistema que mata els processos quan l'usuari es desconnecta. Aquesta comanda permet l'execució en segon pla de comandes que poden requerir molt de temps. Per exemple:

```
usuari@nomMaquina:~$ nohub ping -i 10 www.uoc.edu
```

Podem emprar la comanda `time` per a determinar quant de temps requereix una comanda per a la seva execució. Pot ser útil per a testejar el rendiment d'un guió de seqüències o d'una comanda concreta. Imagineu que disposeu de dos *scripts* diferents i desitgeu saber quin dels dos és més ràpid. Això es podria saber amb la comanda `time`. El resultat d'aquesta comanda pot variar en funció de la *shell* utilitzada. Malgrat això, amb Bash, apareixen els resultats següents:

- `real`: és el temps des que ha començat el procés fins que ha finalitzat.
- `user`: és el temps que el procés ha estat en espai de mode d'usuari.
- `system`: és el temps que el procés ha estat en l'espai del *kernel*.

Mostrem a continuació un exemple senzill:

```
usuari@nomMaquina:~$ time ping -c 1 www.google.es
PING www.google.es(ocsp.pki.goog (2a00:1450:4003:80b::2003)) 56 data bytes
64 bytes from ocsp.pki.goog (2a00:1450:4003:80b::2003): icmp_seq=1 ttl=53 time=43.7 ms
--- www.google.es ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 43.749/43.749/43.749/0.000 ms
real    0m0,051s
user    0m0,005s
sys     0m0,000s
```

La darrera comanda que esmentem en aquest apartat és `nice`, que permet canviar la prioritat amb la qual s'executarà un procés. Amb el nombre `-20` s'indica màxima prioritat (fet que significa que la CPU prioritzarà l'execució d'aquest procés en comptes de d'altres processos existents), fins al rang del nombre 19 (que indica prioritat mínima). Per exemple:

```
usuari@nomMaquina:~$ nice -n 13 nano h.txt
```

12. Execució de processos automatitzats amb Cron

Cron prové del terme grec *chronos*, que significa «temps». En sistemes operatius de tipus Unix, aquest Cron és un programa que executa processos o guions de seqüencia en segon pla (això significa, de manera automàtica sense intervencions de l'usuari). Aquest procés està instal·lat en tots els sistemes operatius GNU/Linux i es configura per mitjà del fitxer «crontab», ubicat habitualment a «/etc/crontab».

Aquest fitxer conté unes quantes línies i cada línia especifica una tasca a executar. Els camps concrets són els següents:

```
# m h dom mon dow user command
```

on:

- `m` correspon al minut en el qual s'executarà l'*script*, amb un valor entre 0 i 59.
- `h`, l'hora exacta, amb un valor entre 0 i 23.
- `dom` fa referència al dia del mes (*day of month*).
- `dow` fa referència al dia de la setmana (*day of week*). S'indica amb un valor numèric (on el 0 serà el diumenge, 1 dilluns i així successivament) o amb les tres primeres lletres del nom en anglès: mon, tue, wed, thu, fri, sat, sun.
- `user` es refereix a l'usuari que executarà el procés o l'*script*.
- `command` es refereix a la comanda que es desitja executar. S'aconsella indicar aquí la ruta completa. Per exemple: «/home/usuari/scripts/descarrega.sh».

No és obligatori indicar tots els camps. Alguns d'aquests es poden substituir amb un asterisc. Alguns exemples:

```
00 22 * * * usuari /home/usuari/scripts/descarrega.sh
```

L'usuari «usuari» executarà tots els dies a les 22:00 h aquest *script* «descarrega.sh» ubicat en la carpeta assenyalada (concretament «/home/usuari/scripts/»).

O, aquest altre exemple:

```
30 3 * * sun root apt-get -y update
```

On tots els diumenges a les 3:30 h de la nit l'usuari *root* executarà aquesta comanda `apt-get -y update`.

13. Xarxa, connectivitat i clients web

Indiquem en aquest apartat algunes comandes d'interès per a la xarxa i la connectivitat a internet. Hi ha qui pot considerar que algunes d'aquestes comandes poden ser fora de l'abast d'aquesta assignatura, però s'agreguen aquí amb l'únic objectiu que l'usuari disposi dels coneixements adequats per a poder realitzar tests senzills sobre la connectivitat a la xarxa i a internet. Després d'aquestes comandes més tècniques, s'anomenen dues comandes més molt útils per a descarregar fitxers via web, per exemple.

Les comandes que hem decidit incorporar en aquesta secció són les següents:

`ip address` és la comanda que mostrarà informació sobre les adreces de xarxa assignades a les diferents interfícies de l'ordinador.

```
usuari@nomMaquina:~$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
 link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
 inet 127.0.0.1/8 scope host lo
 valid_lft forever preferred_lft forever
 inet6 ::1/128 scope host
 valid_lft forever preferred_lft forever
2: enp0s31f6: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN group
default qlen 1000
 link/ether b0:6e:bf:61:9c:18 brd ff:ff:ff:ff:ff:ff
3: wlp4s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default
qlen 1000
 link/ether 1c:87:2c:b8:3d:b5 brd ff:ff:ff:ff:ff:ff
 inet 192.168.1.106/24 brd 192.168.1.255 scope global noprefixroute wlp4s0
 valid_lft forever preferred_lft forever
 inet6 2a02:2e02:9e3c:8700:7c1c:f871:6d61:345a/64 scope global temporary dynamic
 valid_lft 880sec preferred_lft 280sec
 inet6 2a02:2e02:9e3c:8700:1d2e:ea31:a1:c293/64 scope global dynamic mngtmpaddr noprefixroute
 valid_lft 880sec preferred_lft 280sec
 inet6 fe80::dcfb:a25a:71c5:d60f/64 scope link noprefixroute
 valid_lft forever preferred_lft forever
4: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
 link/ether 02:42:88:28:f6:09 brd ff:ff:ff:ff:ff:ff
 inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
 valid_lft forever preferred_lft forever
```

En aquest cas concret s'observen quatre interfícies de xarxa (la primera, la de *loopback*, que sempre disposa de l'adreça 127.0.0.1, la segona, que en aquest cas no disposa d'adreça de xarxa, la tercera, que és un enllaç via wifi i que disposa de l'adreça 192.168.1.106 i, finalment, una quarta virtual).

Amb la comanda `ip route` és possible conèixer quines són les màquines que actuen de porta d'enllaç de la nostra xarxa.

```
usuari@nomMaquina:~$ ip route
default via 192.168.1.1 dev wlp4s0 proto dhcp metric 600
169.254.0.0/16 dev docker0 scope link metric 1000 linkdown
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1 linkdown
192.168.1.0/24 dev wlp4s0 proto kernel scope link src 192.168.1.106 metric 600
En aquest cas concret, la màquina que actua de porta d'enllaç disposa de l'adreça de xarxa
```

```
192.168.1.1.
```

`ping` és la millor comanda per a testejar la connectivitat entre diversos nodes d'una xarxa informàtica, tant es tracti d'una xarxa d'abast local (LAN) com d'una d'abast global (WAN). `Ping` utilitza paquets del tipus ICMP[8] per a comunicar-se amb altres dispositius. En concret, es tracta de paquets ICMP Echo Reply. Vegem a continuació un parell d'exemples.

(8)De l'anglès *Internet Control Message Protocol*.

```
usuari@nomMaquina:~$ ping www.google.es
PING www.google.es(www.google.es (2a00:1450:4003:801::2003)) 56 data bytes
64 bytes from www.google.es (2a00:1450:4003:801::2003): icmp_seq=1 ttl=53 time=44.7 ms
64 bytes from www.google.es (2a00:1450:4003:801::2003): icmp_seq=2 ttl=53 time=44.6 ms
64 bytes from www.google.es (2a00:1450:4003:801::2003): icmp_seq=3 ttl=53 time=45.5 ms
64 bytes from www.google.es (2a00:1450:4003:801::2003): icmp_seq=4 ttl=53 time=44.3 ms
^C

--- www.google.es ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 44.377/44.846/45.567/0.464 ms
```

Observem que aquest *ping* no s'atura. Per a fer-ho, caldrà trametre al procés una senyal **TERM** per tal que ho faci emprant la combinació de tecles «control + c».

Amb el paràmetre `-c` podem indicar el nombre exacte de paquets ICMP que desitgem trametre:

```
usuari@nomMaquina:~$ ping -c 1 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=1.72 ms
--- 192.168.1.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.726/1.726/1.726/0.000 ms
```

Amb la comanda `traceroute` l'usuari pot saber per quantes (i per quines) màquines viatgen les seves peticions. Per exemple, des de l'ordinador on escrivim aquest text fins al servidor web de la UOC (www.uoc.edu), hi ha vuit màquines:

```
usuari@nomMaquina:~$ traceroute www.uoc.edu
traceroute to www.uoc.edu (213.73.40.242), 30 hops max, 60 byte packets
 1 Livebox (192.168.1.1) 1.739 ms 2.945 ms 4.075 ms
 2 192.0.0.1 (192.0.0.1) 21.776 ms 22.141 ms 23.234 ms
 3 10.34.35.134 (10.34.35.134) 26.595 ms 26.761 ms 27.691 ms
 4 10.34.35.206 (10.34.35.206) 38.789 ms 39.234 ms 39.435 ms
 5 anella01.01.catnix.net (193.242.98.38) 40.333 ms 41.607 ms 42.500 ms
 6 in3-anella.cesca.cat (84.88.18.42) 45.243 ms 34.008 ms 35.061 ms
 7 * * *
 8 www.uoc.edu (213.73.40.242) 27.121 ms 25.073 ms 25.665 ms
```

`wget` és una utilitat de terminal que permet descarregar fitxers de servidors web remots, emprant protocols com HTTP, HTTPS o també FTP. `wget` també pot ser útil per a descarregar fitxers de manera automàtica dins dels *scripts*. Amb la comanda següent descarregarem un fitxer concret del servidor web especificat:

```
usuari@nomMaquina:~$ wget http://tldp.org/Linux-HOWTO-text.tar.gz
--2019-10-01 06:42:38-- http://tldp.org/Linux-HOWTO-text.tar.gz
S'està resolent tldp.org (tldp.org)… 152.19.134.41
```

```
S'està connectant a tldp.org (tldp.org)|152.19.134.41|:80… conectat.
HTTP: s'ha enviat la petició, s'està esperant una resposta… 200 OK
Mida: 9048579 (8,6M) [application/x-gzip]
S'està desant a: «Linux-HOWTO-text.tar.gz»
Linux-HOWTO-text.ta 100%[===================>] 8,63M 2,56MB/s in 3,4s
2019-10-01 06:42:42 (2,56 MB/s) - s'ha desat «Linux-HOWTO-text.tar.gz» [9048579/9048579]
```

A vegades succeeix que desitgem descarregar fitxers de servidors web amb HTTPS i el client desconeix el certificat digital. Per aquest motiu es recomana agregar el paràmetre `--no-check-certificate`.

```
usuari@nomMaquina:~$ wget https://tldp.org/Linux-HOWTO-text.tar.gz
--2019-10-01 06:43:22-- https://tldp.org/Linux-HOWTO-text.tar.gz
S'està resolent tldp.org (tldp.org)… 152.19.134.41
S'està connectant a tldp.org (tldp.org)|152.19.134.41|:443… conectat.
ERROR: cap nom comú alternatiu del certificat concorda amb el nom del servidor demanat «tldp.org».
Per a connectar a tldp.org de manera insegura, useu «--no-check-certificate».
```

En canvi:

```
usuari@nomMaquina:~$ wget --no-check-certificate https://tldp.org/Linux-HOWTO-text.tar.gz
--2019-10-01 06:44:01-- https://tldp.org/Linux-HOWTO-text.tar.gz
S'està resolent tldp.org (tldp.org)… 152.19.134.41
S'està connectant a tldp.org (tldp.org)|152.19.134.41|:443… conectat.
AVÍS: cap nom comú alternatiu del certificat concorda amb el nom del servidor demanat «tldp.org».
HTTP: s'ha enviat la petició, s'està esperant una resposta… 200 OK
Mida: 9048579 (8,6M) [application/x-gzip]
S'està desant a: «Linux-HOWTO-text.tar.gz»
Linux-HOWTO-text.ta 100%[===================>] 8,63M 4,05MB/s in 2,1s
2019-10-01 06:44:03 (4,05 MB/s) - s'ha desat «Linux-HOWTO-text.tar.gz» [9048579/9048579]
```

`curl` és una comanda similar a l'anterior, amb lleugeres diferències, que emprarem de manera indistinta amb l'anterior. Per exemple:

```
usuari@nomMaquina:~$ curl -o Linux-HOWTO-text.tar.gz http://tldp.org/Linux-HOWTO-text.tar.gz
 % Total % Received % Xferd Average Speed Time Time Time Current Dload Upload Total Spent Left Speed
100 8836k 100 8836k 0 0 3397k 0 0:00:02 0:00:02 --:--:-- 3397k
```

on el paràmetre `-o` indica el nom del fitxer de sortida.

14. Creació de guions de seqüencies (*scripts*)

És possible elaborar sistemes que permetin l'execució de les comandes anteriors en bloc, de manera continuada, amb l'execució d'una darrera l'altra i sense que l'usuari hi hagi d'intervenir. Aquests programes són els ***scripts***, en català, **guions de seqüencies**. Es tracta de fitxers de text que contenen les instruccions. Elaborar aquests *scripts* pot ser útil per a l'execució de tasques que es realitzen de manera repetitiva o que poder ser útils per a l'automatització de tasques diverses (per exemple, la creació de còpies de seguretat nocturnes, la descàrrega de grans fitxers d'internet, entre d'altres, tot de manera automàtica).

Per a l'elaboració d'*scripts*, caldrà emprar un editor de text tipus el que hem esmentat abans i escriure quelcom similar al següent:

```
#!/bin/bash
echo "Aquest és el meu primer script. Mostrarà per pantalla el fitxer de passwd"
cat /etc/passwd
```

La primera línia indica quin ha de ser el programa que interpreti les instruccions que seguiran a continuació (en el cas que ens ocupa serà «/bin/bash»). Abans de poder executar aquest *script*, caldrà atorgar-li el permís d'execució:

```
usuari@nomMaquina:~$ chmod +x script.sh
```

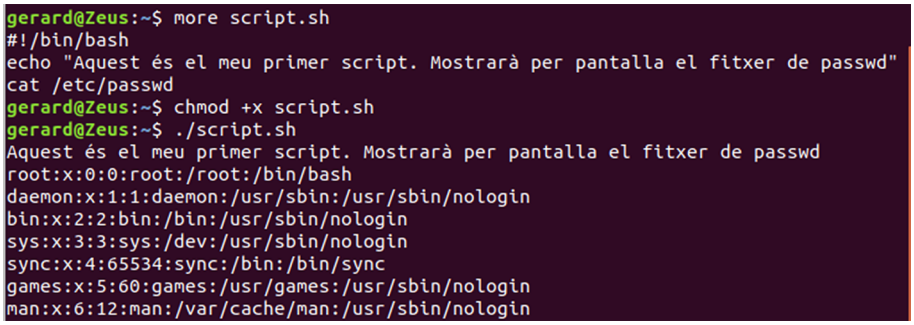
Ara, ja serà possible executar-lo amb:

```
usuari@nomMaquina:~$ ./script.sh
```

El `./` inicial indica on està el fitxer en concret. En aquest cas, en la mateixa carpeta on està l'usuari. També seria possible executar-lo indicant la ruta completa d'ubicació del fitxer. Per exemple:

```
/home/usuari/script.sh
```

Figura 6. Captura de pantalla en què s'observa el procés d'execució d'aquest `script.sh`



De manera similar es poden executar guions escrits amb altres llenguatges de programació. Per exemple, podríem executar *scripts* amb Python. La primera línia de l'*script* conté l'intèrpret que serà l'encarregat d'executar les comandes que vindran a continuació. Vegem el cas següent:

```
usuari@nomMaquina:~$ more scriptenPython.sh
#!/usr/bin/env python3
print('Hola a tothom des de Python!')

usuari@nomMaquina:~$ chmod +x scriptenPython.sh

usuari@nomMaquina:~$ ./scriptenPython.sh
Hola a tothom des de Python!
```

15. Distribucions de GNU/Linux

15.1. Introducció

Una **distribució de GNU/Linux** consisteix en un sistema operatiu generat per mitjà d'un conjunt de programari que es basa, principalment, en el *kernel* de Linux i, normalment, amb un sistema de gestió de paquets de programari.

Habitualment els usuaris de GNU/Linux descarreguen una ISO amb la seva distribució preferida, primer l'enregistren en un suport extraïble tipus memòria flash o CDROM i, després, la instal·len en el seu maquinari. Hi ha distribucions per a tot tipus de maquinari: per a dispositius encastats, ordinadors personals o supercomputadors. També hi ha distribucions dissenyades amb recopilacions de programari específic per a certes tasques (per exemple, distribucions amb aplicacions educatives, científiques, de seguretat informàtica, entre d'altres).

Una distribució típica de GNU/Linux conté, tal com hem comentat, un *kernel*, eines i llibreries GNU, programari, documentació i un entorn gràfic (malgrat que és quelcom opcional). La majoria del programari que s'inclou és programari lliure i està disponible per mitjà del codi font o compilat amb fitxers binaris executables. Algunes distribucions inclouen de manera opcional programari propietari, sense disposar de codi font, normalment per a alguns controladors de dispositius (el que tècnicament s'anomena *drivers*).

Normalment, els responsables de la distribució s'encarreguen d'adaptar el programari, posar-lo en paquets i distribuir-lo en línia mitjançant el que s'anomena **repositoris** (servidors amb programari) distribuïts arreu del món.

Actualment hi ha centenars de distribucions diferents. Algunes d'aquestes mantingudes amb **suport comercial** com, per exemple, Fedora (Red Hat), openSUSE (SUSE) o Ubuntu (Canonical Ltd.) i altres mantingudes completament per la **comunitat**, com ara Slackware o Debian.

15.2. La distribució escollida: Ubuntu

Ubuntu és una distribució de GNU/Linux basada en una altra de més antiga que s'anomena Debian. Es tracta d'una distribució gratuïta i de programari lliure amb diferents versions: una per a entorns d'escriptori (*Desktop*), una altra per a entorns de servidor (*Server*) i una altra per a dispositius IoT (*Internet of Things*) i robots (*Core*). S'allibera una versió d'Ubuntu cada sis mesos, amb una

Figura 7. Logotip d'Ubuntu

versió amb suport per a llarg termini cada dos anys. En el moment d'escriure aquests apunts (estiu de 2019), la darrera versió és la 19.04 (titulada «Disco Dingo») i la versió per a llarg termini és la 18.04 (anomenada «Bionic Beaver»).

Ubuntu està desenvolupada per a tota una comunitat d'usuaris i l'empresa Canonical amb un model de governança basat en la **meritocràcia**. El nom Ubuntu prové del terme filosòfic africà que Canonical va traduir com «la humanitat cap als altres» o «jo soc qui soc en funció del que totes les persones som».

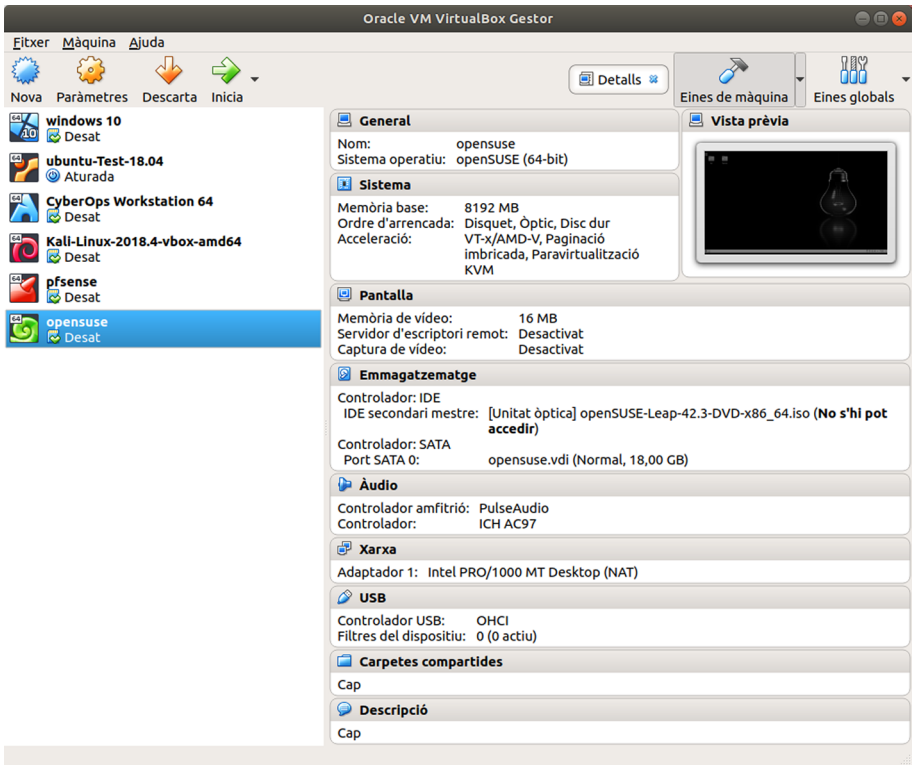
15.3. Instal·lació d'Ubuntu

Hi ha diferents maneres per a disposar d'una distribució Ubuntu accessible. Una d'aquestes consisteix a descarregar la imatge d'instal·lació del lloc web oficial, gravar aquest fitxer en un dispositiu extern tipus un USB, un CD-ROM o un DVD i iniciar un ordinador que no disposi de sistema operatiu per mitjà d'aquests mitjans extraïbles. El sistema iniciaria un procés d'instal·lació a l'ordinador i ja el tindríem disponible.

Una altra opció, que s'aconsella per als estudiants menys experimentats, consisteix a portar a terme la mateixa operació però en un entorn de màquines virtuals. Una màquina virtual és un programari que permet l'execució simulada de diferents sistemes operatius sobre un altre sistema operatiu ja existent. L'avantatge és que no cal disposar d'una màquina completa per a la instal·lació del sistema i es pot mantenir, per exemple, un sistema base amb Microsoft Windows i també executar-hi un sistema Ubuntu per mitjà d'aquest programari de màquines virtuals.

Hi ha diferents programaris per a la instal·lació de màquines virtuals, per exemple, VirtualBox o VMware Workstation Player.

Figura 8. Captura de pantalla de l'execució del programari VirtualBox



En aquesta instal·lació hi ha diverses màquines virtuals (un Windows 10 i una distribució Linux anomenada Kali Linux amb programari especialitat de seguretat informàtica).

Un altre avantatge d'aquest sistema de virtualització consisteix en la possibilitat de poder descarregar i instal·lar sistemes operatius ja preparats. El tipus de fitxer **OVA**, per exemple, és un estàndard obert que consisteix en un format de virtualització que empaqueta sistemes operatius virtuals. Així, és possible descarregar un fitxer «.OVA» amb Ubuntu, incorporar-la al programari de màquines virtuals i, ràpidament, disposar de la distribució accessible.

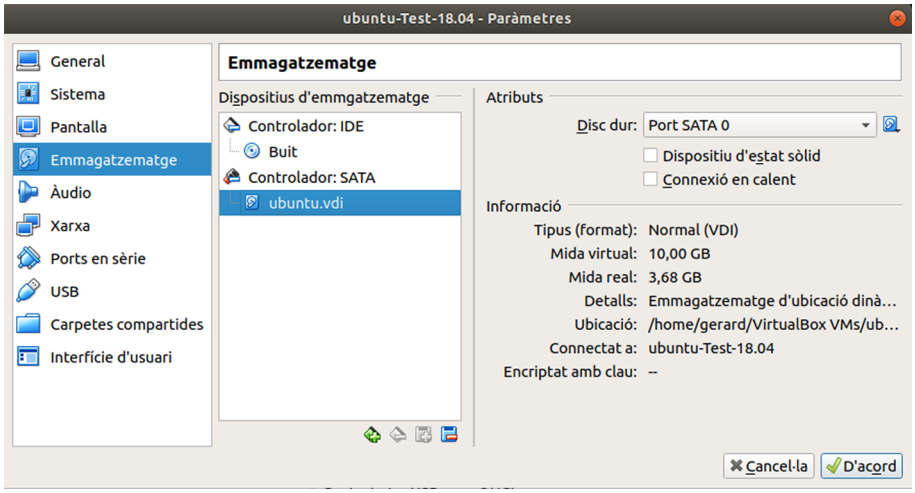
Per a dur a terme aquesta operació amb el programari de VirtualBox cal solament anar a «Fitxer» --> «Importa una aplicació virtual».

Una altra possibilitat consisteix a descarregar una còpia d'un disc dur virtual (es tracta de fitxers amb les extensions VDI o VMDK). Llavors, una vegada descarregat, cal crear una màquina nova a Ubuntu i configurar-la per tal que utilitzi el disc dur carregat.

Fitxers VDI

En www.osboxes.org/ubuntu podeu descarregar fitxers VDI per a màquines Ubuntu amb el sistema operatiu ja instal·lat.

Figura 9. Captura de la pantalla del programari VirtualBox on és possible configurar els discs durs de les màquines virtuals



15.4. Instal·lació de programari

Almenys hi ha dues vies per a la instal·lació de programari amb sistemes GNU/Linux. La primera d'aquestes consisteix a descarregar el codi del programa, compilar-lo i obtenir un fitxer executable. Es tracta d'un mètode tradicional però per a usuaris més avançats. Recordem que el **procés de compilació** consisteix a convertir un codi escrit per humans en un codi executable per una màquina. Un avantatge d'aquest mètode és que permet controlar tot el procés i configurar el programari tal com desitja l'usuari, però cal estar molt més al cas de tots els detalls.

El segon sistema, molt més senzill per a usuaris no tan experimentats i el recomanat en el marc d'aquesta assignatura, utilitza els **repositoris de paquets de programari** mantinguts per la mateixa distribució. Per a la gestió d'aquests paquets, Ubuntu empra un sistema desenvolupat per Debian que s'anomena *APT*.[9] Es tracta d'un sistema capaç de gestionar programari en el nostre ordinador. Per exemple, descarregant paquets en servidors externs (repositoris) i instal·lant-los en el sistema.

[9] De l'anglès *Advanced Packaging Tool*.

15.4.1. Instal·lació de programari emprant codi font

Mostrarem a continuació un exemple de com instal·lar programari descarregant el codi font de la xarxa. Es tracta d'un sistema avançat, ja que, malgrat que permet conèixer tots els detalls de la instal·lació, requereix dependències d'altres programes.

En aquest cas concret descarregarem i compilarem el programa **Grep**. La primera instrucció descarrega el fitxer del web (amb l'opció `-q` mostra menys informació per pantalla). La segona instrucció desempaqueta i descomprimeix el fitxer). I la tercera instrucció accedeix dins la carpeta descomprimida.

```
usuari@nomMaquina:~$ wget -q https://ftp.gnu.org/gnu/grep/grep-3.3.tar.xz
usuari@nomMaquina:~$ tar -xvf grep-3.3.tar.xz
```

```
usuari@nomMaquina:~$ cd grep-3.3/
```

A continuació s'executa un *script* (habitualment s'anomena *configure*) que s'assegura de disposar de totes les dependències de programari en el nostre ordinador i que genera un fitxer «Makefile» que conté les instruccions de compilació per a la nostra màquina en concret. De nou, l'opció `-q` l'afegim aquí per a obtenir una sortida amb menys línies.

```
usuari@nomMaquina:~/grep-3.3$ ./configure -q
config.status: creating po/POTFILES
config.status: creating po/Makefile
```

Amb el «Makefile» construït ja és possible compilar el programari amb la comanda `make`. Una vegada finalitzat el procés, el programa resideix dins la carpeta «src» i es pot executar ja el binari. Recordem que, com que aquesta carpeta no resideix en la variable d'entorn **PATH**, cal indicar la ubicació exacta del fitxer emprant el «./» (que indica la carpeta actual):

```
usuari@nomMaquina:~/grep-3.3$ make
usuari@nomMaquina:~/grep-3.3$ cd src
usuari@nomMaquina:~/grep-3.3/src$ ./grep
Usage: grep [OPTION]... PATTERNS [FILE]...
Feu servir «grep --help» per a obtenir més informació.
```

En el cas que es provi aquest mateix procés en un altre ordinador, pot succeir que no es disposi de tot el programari requerit per a la compilació del programa (compilador, *make*, llibreries que requereix el programa, etc.). És per aquest motiu que s'aconsella als estudiants emprar aquest mètode solament en el cas que el programari que desitgem instal·lar no estigui en els repositoris de programari mantinguts per la distribució.

15.4.2. Instal·lació de programari emprant repositoris d'Ubuntu

Recordem que aquest és el mètode recomanat per a la instal·lació de programari, ja que, a part de ser molt més senzill, els administradors de la distribució s'hauran assegurat que tot funciona adequadament.

El llistat de repositoris estan definits en el fitxer «/etc/apt/sources.list» i en la carpeta «/etc/apt/sources.list.d», encara que no hem de manipular directament aquests fitxers.

```
usuari@nomMaquina:/etc/apt$ cat /etc/apt/sources.list | grep -v ^# | sed '/^$/d'
deb http://es.archive.ubuntu.com/ubuntu/ bionic main restricted
deb http://es.archive.ubuntu.com/ubuntu/ bionic-updates main restricted
deb http://es.archive.ubuntu.com/ubuntu/ bionic universe
deb http://es.archive.ubuntu.com/ubuntu/ bionic-updates universe
deb http://es.archive.ubuntu.com/ubuntu/ bionic multiverse
deb http://es.archive.ubuntu.com/ubuntu/ bionic-updates multiverse
deb http://es.archive.ubuntu.com/ubuntu/ bionic-backports main restricted universe multiverse
deb http://security.ubuntu.com/ubuntu bionic-security main restricted
deb http://security.ubuntu.com/ubuntu bionic-security universe
deb http://security.ubuntu.com/ubuntu bionic-security multiverse
```

En la comanda anterior, el primer `cat` mostra el contingut del fitxer «/etc/apt/sources.list», que traspassa emprant un *pipe* el resultat a la comanda `grep`. Aquesta comanda exclou les línies que comencin pel caràcter coixinet (el paràmetre `-v` és per a excloure, el caràcter «^» significa inici de línia, seguit per un «#»). Finalment, el darrer `sed`, suprimeix les línies que estan en blanc.

Vegem a continuació algunes comandes d'`apt`:

```
usuari@nomMaquina:~$ apt-cache search nomdeprograma #Aquesta comanda cercarà
en els repositoris programari que contingui la paraula «nomdeprograma». Per exemple:

usuari@nomMaquina:/etc/apt$ apt-cache search data science
algotutor - program for observing the intermediate steps of algorithm
beneath-a-steel-sky - classic 2D point and click science fiction adventure game
bibutils - interconvert various bibliographic data formats
cernlib - CERNLIB data analysis suite - general use metapackage
cernlib-base - CERNLIB data analysis suite - common files
cernlib-base-dev - CERNLIB data analysis suite - dependencies checking script
cernlib-core - CERNLIB data analysis suite - main libraries and programs
cernlib-core-dev - CERNLIB data analysis suite - core development files
cernlib-extras - CERNLIB data analysis suite - extra programs
cernlib-montecarlo - CERNLIB Monte Carlo libraries
cwltool - Common Workflow Language reference implementation
daligner - local alignment discovery between long nucleotide sequencing reads
datalad - data files crawler and data distribution
dict-wn - electronic lexical database of English language for dict
```

També és possible obtenir informació d'un programari en concret:

```
usuari@nomMaquina:~$ apt-cache show jq
Package: jq
Architecture: amd64
Version: 1.5+dfsg-2
Multi-Arch: foreign
Priority: optional
Section: universe/utils
Origin: Ubuntu
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Original-Maintainer: ChangZhuo Chen <czchen@debian.org>
Bugs: https://bugs.launchpad.net/ubuntu/+filebug
Installed-Size: 88
Depends: libjq1 (= 1.5+dfsg-2), libc6 (>= 2.4)
Filename: pool/universe/j/jq/jq_1.5+dfsg-2_amd64.deb
Size: 45646
MD5sum: 9a9c411594822e8c69d82d2d7aa97766
SHA1: d03195dec7219ca7d3a9fbf712a8c92d0c2b7801
SHA256: 7da6b3fe4de02f50169923134b1e6c4cbf4ee693afa963d02d1eb8a0e3adb472
Homepage: https://github.com/stedolan/jq
Description-en: lightweight and flexible command-line JSON processor
 jq is like sed for JSON data - you can use it to slice
 and filter and map and transform structured data with
 the same ease that sed, awk, grep and friends let you
 play with text.
 .
 It is written in portable C, and it has minimal runtime
 dependencies.
 .
 jq can mangle the data format that you have into the
 one that you want with very little effort, and the
 program to do so is often shorter and simpler than
 you'd expect.
Description-md5: fd8d7d97b13012ce68c52772c1ce56aa
```

```
usuari@nomMaquina:~$ apt-get install nomdeprograma #Aquesta comanda descarregarà el programa
del repositori i l'instal·larà en el nostre sistema. En el cas que aquest programari tingui
dependències, també les instal·larà. També podria succeir que el programari ja estigués instal·lat.
Així doncs, no realitzaria cap tasca. Per exemple:
```

```
usuari@nomMaquina:~$ sudo apt-get install jq
[sudo] contrasenya per a usuari:
S'està llegint la llista de paquets… Fet
S'està construint l'arbre de dependències
S'està llegint la informació de l'estat… Fet
jq ja està en la versió més recent (1.5+dfsg-2).
Els paquets següents s'han instal·lat automàticament i ja no seran necessaris:
 liblog4cxx10v5 libqt5script5 libqt5scripttools5 libzzip-0-13
Empreu «sudo apt autoremove» per a suprimir-los.
0 actualitzats, 0 nous a instal·lar, 0 a suprimir i 0 no actualitzats.
```

```
usuari@nomMaquina:~$ apt-get remove nomdelprograma #Aquesta comanda suprimirà del nostre
ordinador el programa mencionat. Per exemple:

usuari@nomMaquina:/etc/apt$ sudo apt-get remove jq
[sudo] contrasenya per a usuari:
S'està llegint la llista de paquets… Fet
S'està construint l'arbre de dependències
S'està llegint la informació de l'estat… Fet
Els paquets següents s'han instal·lat automàticament i ja no seran necessaris:
 libjq1 liblog4cxx10v5 libonig4 libqt5script5 libqt5scripttools5 libzzip-0-13
Empreu «sudo apt autoremove» per a suprimir-los.
Es SUPRIMIRAN els paquets següents:
 jq
0 actualitzats, 0 nous a instal·lar, 1 a suprimir i 0 no actualitzats.
Després d'aquesta operació s'alliberaran 90,1 kB d'espai en disc.
Voleu continuar? [S/n] S
(S'està llegint la base de dades… hi ha 291242 fitxers i directoris instal·lats actualment.)
S'està desinstal·lant jq (1.5+dfsg-2)…
S'estan processant els activadors per a man-db (2.8.3-2ubuntu0.1)…
```

```
usuari@nomMaquina:~$ apt-get update #Amb aquest paràmetre el programa apt-get actualitzarà
els índex amb els repositoris que disposa.
```

Cal comentar que algunes d'aquestes comandes realitzen tasques d'administració en el sistema i que realitzen canvis importants. Per tant, caldrà executar-les emprant permisos d'administració. Per exemple, amb la comanda `sudo`.

Bibliografia

Garrels, M. (2008). «Introduction to Linux. A Hands on Guide» [en línia]. [Data de consulta: 1 d'octubre de 2019]. tldp.org/LDP/intro-linux/intro-linux.pdf

GNU nano (2016). «nano Command Manual» [en línia]. [Data de consulta: 1 d'octubre de 2019]. www.nano-editor.org/dist/v2.1/nano.html

Oracle Corporation (2019). «Oracle® VM VirtualBox®. User Manual» [en línia]. [Data de consulta: 1 d'octubre de 2019]. www.virtualbox.org/manual/

Peek, J.; Strang, J.; Todino, G. (2014). *Learning the Unix Operating System. A concise Guide for the New User*. California: O'Reilly Media.

UBUNTU Documentation (2013). «CompilingEasyHowTo» [en línia]. Ubuntu documentation. [Data de consulta: 1 d'octubre de 2019]. help.ubuntu.com/community/CompilingEasyHowTo

UBUNTU Documentation (2016). «CronHowto» [en línia]. Ubuntu documentation. [Data de consulta: 1 d'octubre de 2019]. help.ubuntu.com/community/CronHowto

UBUNTU Documentation (2016). «Getting Started with Ubuntu 16.04» [en línia]. [Data de consulta: 1 d'octubre de 2019]. files.ubuntu-manual.org/manuals/getting-started-with-ubuntu/16.04/en_US/screen/Getting%20Started%20with%20Ubuntu%2016.04.pdf

Zonker (2010). «Writing A Simple Bash Script» [en línia]. Linux.com. [Data de consulta: 1 d'octubre de 2019]. www.linux.com/tutorials/writing-simple-bash-script/.