Guia de supervivència informàtica.

Llicenciatura de Matemàtiques UAB, 2004–2005

Josep Maria Mondelo

1 Ús bàsic de Linux

1.1 Consoles virtuals

Disposeu de 6 consoles virtuals de text, hi podeu accedir fent `CTRL-ALT-F1`, `CTRL-ALT-F2`, ..., `CTRL-ALT-F6`.

Disposeu d'un número arbitrari de consoles gràfiques, començant amb la `CTRL-ALT-F7`. Aquesta és la consola en la que "aterreu" quan engegueu l'ordinador. Podeu provar de passar a la primera consola de text, amb `CTRL-ALT-F1`, després tornar a la gràfica amb `CTRL-ALT-F7`.

Només podeu entrar des de la consola gràfica (`CTRL-ALT-F7`). Per entrar, heu d'escriure, com a nom d'usuari

`NIU`

on `NIU` és el vostre NIU. Com a password, heu d'escriure el mateix que useu en Windows. Si no podeu entrar, proveu de:

- Desbloquejar les majúscules i el teclat numèric (tecles "Bloq Mayús" i "Bloq Num").
- No usar el teclat numèric (només els números de sobre les lletres).
- Seleccionar "KDE" com es tipus de sessió d'entrada.

1.2 Arbre de directoris

En Linux (de fet en qualsevol sistema Unix) no hi ha "lletres d'unitat" (`a:`,`c:`,...) com al Windows, sinó que hi ha un únic arbre de directoris. El directori arrel es representa per `/`, i aquest mateix caràcter fa de separador de directoris. Així, `/bin` és el subdirectori `bin` del directori arrel, i `/bin/ls` és el fitxer `ls` del directori `/bin`.

Existeix una operació anomenada "muntar", que consisteix a associar els continguts d'un cert dispositiu (com un floppy, un CD-ROM o una partició del disc dur) a un punt de l'arbre de directoris general. L'operació contrària es diu "desmuntar". Veurem exemples d'això més endavant.

1.3 Línia de comandes

Si des de l'entorn gràfic obriu una terminal (podeu fer doble-clic a la icona "Konsole"de l'escriptori, o bé, a l'esquerra de la barra inferior, cliqueu al logo de Mandrake [l'estrelleta], llavors "Sistema", "Terminales", "Konsole") us apareixerà alguna cosa com

```
a1234567@PC1C_40 a1234567:~$
```

D'això se'n diu "prompt", i significa que el sistema està esperant que li entreu una *comanda*. Si escriviu alguna cosa i apreteu `RETURN`, la primera paraula del que hagueu escrit s'interpretarà com el nom d'un programa a executar, i la resta s'interpretarà com arguments, que es passaran al programa en qüestió.

Així, si escriviu

```
ls
```

s'executarà la comanda `ls`, que mostra els fitxers del directori de treball (probablement el vostre directori *home*), i si escriviu

```
ls -l
```

s'executa la comanda `ls` amb l'opció `-l`, i, per tant, a més dels noms dels fitxers del directori de treball, veureu informació addicional com els seus permisos, tamany o data de darrera modificació (vegeu la següent secció).

1.4 Comandes bàsiques

- `pwd`

  Escriu el nom del directori de treball. És el que es presuposa als noms de fitxers que escriviu sense especificar el directori al qual pertanyen.

- `cd`

  Canvia el directori de treball a l'argument que se li passi. Si no se li passa cap argument, canvia el directori de treball al vostre directori *home*. Dins la llista d'arguments, "`..`" fa referència al directori pare del directori de treball, i "`.`" fa referència al directori de treball.

- `ls`

  Sense cap més argument, llista el contingut del directori de treball. Si li poseu un nom de de directori o fitxer, llista aquest directori o fitxer (p.ex., feu "`ls /bin`", "`ls /bin/ls`"). Amb l'opció `-l` us mostra informació addicional, com hem vist abans. L'opció `-d` no llista el contingut dels directoris especificats, sinó els directoris mateixos. Per exemple, si feu

```
ls -ld /bin/ls /bin
```

(noteu que es poden barrejar opcions, "`-ld`" equival a "`-l -d`") obtindreu

```
drwxr-xr-x    2 root    root     4096 Feb 17 13:32 /bin/
-rwxr-xr-x    1 root    root    43784 Mar 18  2002 /bin/ls
```

En aquestes línies, la primera columna són els permisos, la tercera el *propietari* del fitxer, la quarta el *grup* del fitxer, a continuació ve la data d'última actualització i finalment el nom.

La columna de permisos s'interpreta com segueix: la primera lletra ens indica si l'entrada corresponent és un directori (`d`) o un fitxer (`-`)[1]. Les tres següents fan referència al permsisos del propietari del fitxer: `r` vol dir que (el propietari) pot veure el seu contingut, `w` vol dir que pot modificar el seu contingut, i `x` vol dir que el pot executar. Per a directoris, "executar" vol dir "poder fer-lo de treball" (amb la comanda `cd`). Les lletres 5–7 fan referència al grup, i les lletres 8–10 fan referència a la resta d'usuaris.[2]

- `cp`

  Copia fitxers. Té dues formes principals:

  `cp fitx1 fitx2`

  Copia `fitx1` a sobre de `fitx2` (compte!).

  `cp fitx1 fitx2 directori`

  Copia els fitxers `fitx1` i `fitx2` al directori `directori`.

  Amb l'opció `-r` s'accepten directoris a la llista de fixters, i llavors es copia tot el seu contingut també.

  A la llista d'arguments s'accepten els anomenats *caracters comodí*, que són `*` i `?`. L'intèrpret de comandes,[3] abans d'exectuar cap comanda expandeix els caràcters comodí, d'acord amb les següents regles:

  `*` representa qualsevol conjunt de caràcters de qualsevol longitud.

[1] Hi ha més codis: `l` vol dir *link simbòlic*, `p` vol dir *pipeline*, `s` vol dir *socket*, ara no ens hi posem.

[2] Com heu endevinat, en un sistema Unix, els usuaris estan agrupats en grups.

[3] El programa ens mostra el prompt i interpreta les nostres comandes, també conegut com a *shell*.

`?` representa exactament un caràcter.

Així, "`*.c`" fa referència a tots els fitxers que acabin en "`.c`", i "`solucio*`" fa referència a qualsevol fitxer que comenci per `solucio`. Així, per exemple, si al directori de treball teniu uns fitxers `solucio1.txt`, `solucio2.txt`, `solucio3.txt` i els voleu copiar al directori pare del directori de treball, heu de fer

```
cp solucio?.txt ..
```

suposant que no hi ha cap altre fitxer que correspongui al patró `solucio?.txt` (si n'hi hagués algun, aquest també es copiaria).

La opció `-i` us demana confirmació abans de fer qualsevol acció potencialment perillosa, per exemple si feu

```
cp -i fitx1 fitx2
```

i `fitx2` ja existeix.

- `mv`

  Mou fitxers i/o directoris ("moure" vol dir copiar i esborrar l'original). Admet els arguments de `cp` que hem vist. També s'usa per canviar un fitxer de nom.

- `rm fitx1 fitx2`

  Esborra els fitxers `fitx1` i `fitx2`. Amb l'opció `-r`, esborra també directoris i el seu contingut.

- `mkdir`

  Crea directori(s).

- `rmdir`

  Esborra directori(s).

- `scp`

  És una comanda que funciona com `cp`, només que un dels seus arguments pot referir-se a un ordinador remot, en aquest cas s'ha de prefixar per `usuari@maquina:~`. Per exemple, si esteu treballant a la peixera, i teniu un fitxer `programa.c` que voleu copiar al subdirectori `practica1` del directori home del servidor de fitxers del laboratori docent, heu de fer

```
scp programa.c inedp01@labmat07.uab.es:~/practica1
```

(la tilde (~) representa el vostre directori home).

- `mount`, `umount`

  Serveixen per muntar i desmuntar dispositius (vegeu la secció 1.2). L'administrador del sistema pot muntar el dispositiu que vulgui al directori que vulgui, mentre que els usuaris normals només poden muntar dispositius específics a llocs específics, triats per l'administrador.

  Per exemple, poseu un disquet a la disquetera i proveu de fer

  ```
  ls -l /mnt/floppy
  mount /mnt/floppy
  ls -l /mnt/floppy
  umount /mnt/floppy
  ls -l /mnt/floppy
  ```

  Podeu fer el mateix amb un CD, llavors heu de canviar `floppy` per `cdrom`.

  IMPORTANT!! Heu de tenir en compte que, mentre un dispositiu està muntat, el sistema operatiu suposa que hi pot escriure quan li sembli, així que MAI NO TREGUEU UN DISQUET DE LA DISQUETERA MENTRE ESTÀ MUNTAT, perquè el podríeu deixar en estat corrupte.[4]

- `mdir`, `mcopy`, `mdel`, `mmd`, `mrd`,...

  Són un conjunt de comandes, que formen part d'unes utilitats anomenades MTOOLS, que permeten manipular un disquet sense muntar-lo. Són equivalents a les comandes corresponents de MS–DOS (o del "Símbolo del sistema" de Windows) sense la `m`. Només heu de tenir en compte que el separador de directoris és `/`, com a Unix, i no \ com a MS–DOS i Windows.

  Així, per veure el contingut d'un disquet heu de fer

  ```
  mdir a:
  ```

[4] Amb un CD no tindreu aquest problema perquè, mentre estigui muntat, encara que apreteu el botó no sortirà. Amb la disquetera això no es pot fer perquè el mecanisme d'extracció és mecànic.

i per copiar un fitxer `programa.c` al directori `practica1` del disquet (suposant que existeixi), heu de fer

```
mcopy programa.c a:/practica1
```

Si voldreu mirar el contingut del disquet des de Windows, podeu usar la opció `-t` de `mcopy` per fer la conversió entre els formats de texte de Linux (Unix, en general) i Windows. Per exemple, si feu

```
mcopy -t programa.c a:/practica1
```

se suposa que `programa.c` (al directori de treball) està en format de Unix i es copiarà fent la conversió Unix→Windows, mentre que si feu

```
mcopy -t a:/practica1/programa.c .
```

se suposa que `programa.c` (al disquet) està en format de Windows i es copiarà fent la conversió Windows→Unix.

ATENCIÓ!!! MAI NO HEU D'USAR l'opció `-t` amb fitxers que no siguin de texte (com els programes C), per que els corromperíeu. Per exemple, NO l'heu d'usar amb fitxers `.gif`, `.doc`, `.xls`, `.jpg`, entre molts d'altres.

1.5 Una sessió d'exemple

A continuació teniu una sessió d'exemple per exercitar les comandes que hem vist. Tot el que hi hagi darrera un caràcter `#` és un comentari i no ho heu d'escriure.

```
cd                      # ens posem al directori home
pwd                     # ara el veiem
cd /bin                 # ens canviem al directori /bin
pwd                     # voilà
cd ..                   # canviem al directori pare de /bin
pwd                     #    que és /
cd                      # tornem al directori home
pwd

ls -ld /bin/ls /bin     # /bin/ls es un fitxer
                        # /bin un directori
unalias cp              # per desactivar la protecció de
```

```
                        #    sobreescriptura
mkdir dirproves         # fem un direcori anomenat "dirproves"
touch f1 f2             # creem dos fitxers f1, f2
ls -l f1 f2             # ... que ara veiem
cp f1 f2                # sobreescrivim f2 amb f1
cp -i f1 f2             # pregunta abans de matxacar
ls -l dirproves         # dirproves és buit
cp -i f1 f2 dirproves   # hi copiem f1 i f2
ls -l dirproves         # ... que ara s'han de veure
cp -i f1 f2 dirproves   # f1, f2 ja són a dirproves
                        #    demana confirmació abans de matxacar
cp f1 f2 dirproves      # matxaca sense preguntar
rm dirproves/*          # esborra tots els fitxers de dirproves
touch a1                # fem un fitxer "a1"
ls -l . dirproves       # llista el directori de treball
                        #    i dirproves
cp * dirproves          # copia tots els fitxers del directori
                        #    de treball a dirproves,
                        #    es queixa perquè no pot
                        #    copiar directoris.
ls -l dirproves
rm dirproves/*
cp f? dirproves         # només copiem f1, f2
ls -l dirproves         # ara ho veiem
rm dirproves/*
ls -l . dirproves
mv f1 dirproves         # f1 desapareix de . i passa a dirproves
ls -l . dirproves
mv f2 f3                # f2 passa a ser f3, matxacant
                        #    els continguts
ls -l
rmdir dirproves         # no podem!
ls -l dirproves         # és perquè no és buit
rm dirproves/*
rmdir dirproves         # ara sí

touch f1 f2
mkdir dirproves
cp * dirproves
ls -l . dirproves
rm -r dirproves         # esborrem dirproves recursivament
```

```
ls -l                   # ... ja no hi és

mdir a:                 # si hi ha un disquet a la disquetera
                        #   veiem el seu contingut
mmd a:/pract0           # fem un subdirectori anomenat
                        #    pract0
mdir a:                 # el veiem
mcopy f? a:/pract0      # hi copiem f1, f2
mdir a:/pract0
mdel a:/pract0/*        # esborrem el contingut de pract0
                        #    (al disquet!)
mrd a:/pract0           # esborrem el subdirectori pract0
mdir a:                 # ja no hi és
```

1.6 Particularitats de la instal·lació de la peixera

Sota el vostre directori home, trobareu un directori `nwhome`, (ruta completa: `/home/NIU/nwhome`, on `NIU` és el vostre NIU). Des d'una terminal us hi podeu posar fent

```
cd nwhome
```

També us hi podeu posar clicant a la icona "nwhome" de l'escriptori de l'entorn gràfic.

Aquest directori és el vostre espai personal de 7MB que també manipuleu des de Windows. Heu de tenir en compte dues coses

- Que és limitat, i només hi podreu guardar 7MB (que són menys de 7 disquets).
- Qualsevol canvi que feu des de Linux també afectarà al que veieu des de Windows. Si hi poseu un nou fitxer, el veureu des de Windows. Si esborreu un que ja hi és, des de Windows no el tornareu a veure.

2 Ús de l'editor

Per escriure programes, podeu fer servir un editor anomenat "KWrite" (n'hi ha molts més). El podeu engegar clicant al logo de Mandrake (l'estrelleta de l'esquerra de la barra de sota), llavors "Más aplicaciones", "Editores", "KWrite". D'entre les seves característiques, ens interessen:

- Que té "syntax highlighting", o sigui, que escriu cada paraula d'un color d'acord amb el llenguatge (C en el nostre cas). Perquè ho faci, ha de saber de quin tipus de fitxer es tracta, així que, si esteu editant un fitxer nou, no tindreu colors fins què el graveu.
- A l'esquerra del texte teniu icones que us permenten "amagar" blocs de programa.
- Cada cop que el cursor és a sota d'un parèntesi, clau o claudàtor, s'ilumina el parèntesi, clau o claudàtor oposat. Això és útil, per exemple, per no oblidar-vos de tancar parèntesis a expressions aritmètiques complicades.

3 Ús del compilador de C

Fem anar el compilador de C del projecte GNU, que és el compilador standard als sistemes Linux. De fet és el mateix compilador que fa servir el Dev–C++. Es diu `gcc`, i una invocació típica és:

```
gcc -g -Wall -oexecutable font.c -lm
```

El significat de cada argument és el següent:

- `-g` és per incloure informació simbòlica, de manera que el programa es pugui executar sota un debugger.
- `-Wall` és perquè el compilador doni tots els warnings. N'hi ha alguns, relacionats amb construccions potencialment perilloses, que per defecte no es treuen.
- `-oexecutable` vol dir que guardi el codi executable produït a un fitxer anomenat `executable`.
- `font.c` és el fitxer amb el codi font en C.
- `-lm` és perquè, en generar l'executable, es linki contra les llibreries matemàtiques (les que contenen les funcions `sin()`, `fabs()`, etc).

L'opció `-g` es pot canviar per `-O3`, que vol dir "optimitzar a nivell 3". Llavors no podreu executar el programa sota el debugger,[5] però, depenent de com sigui el codi, el temps d'execució es reduirà notablement (són freqüents factors entre 2 i 3, de vegades s'aconsegueix més).

[5]Bé, de fet sí que podreu, però no podreu veure el codi C, només el codi màquina desensamblat, i per tant no sabreu a quina línia esteu, ni podreu posar breakpoints referents al codi C, etc.

4 Ús del debugger

Disposeu d'un debugger fet pel projecte GNU, que s'anomena `gdb`. Funciona amb línia de comandes. Per exemple, per començar una sessió de debugging amb el programa generat a dalt, heu de fer

```
gdb executable
```

Llavors us apareixerà un prompt "`(gdb)`" i podeu entrar comandes.

Per executar pas a pas, heu de començant posant un breakpoint a la rutina `main()`:

```
b main
```

Llavors podeu engegar l'execució amb:

```
r
```

Observareu com s'atura a `main()`. Podeu veure on esteu del programa fent

```
bt
```

(vol dir "backtrace"). Per executar una instrucció, hi ha dues comandes: la comanda

```
s
```

(vol dir "step"), i la comanda

```
n
```

(vol dir "next"). La diferència entre les dues és que, quan esteu a sobre d'una rutina, la `s` entra a dins, mentre que la `n` no ho fa.

Si voleu continuar l'execució del programa fins a un altre lloc, hi heu de posar un altre breakpoint ("`b 150`" per parar a la línia 150, "`b jacobi`" per parar a la rutina `jacobi()`, etc) i llavors escriure la comanda

```
c
```

(vol dir "continue").

Podeu veure el contingut de qualsevol variable amb la comanda

```
p
```

(vol dir "print"). Per exemple, per veure el contingut de la variable `index` escriuríeu "`p index`", i per veure la segona component del vector `u` posaríeu "`p u[1]`". De fet la comanda `p` no és només veure el contingut de variables, sinó que admet qualsevol expressió. Així, si voleu canviar el valor de la segona component del vector `u`, podeu fer "`p u[1]=3`".

Quan engegueu el `gdb`, si es troba al mateix directori on l'engegueu un fitxer anomenat `.gdbinit`, hi executa les comandes que trobi a dintre. Així, si hi poseu (dins `.gdbinit`)

```
b main
r
```

llavors, només engegar el `gdb` començarà l'execució del programa i s'aturarà a la rutina `main()`.

Una situació en la qual el debugger és especialment útil és quan el programa casca amb el missatge "`Segmentation fault`". Dues de les causes més freqüents d'aquest error és accedir a un vector amb un índex fora de rang o usar un punter amb una adreça invàlida (p.ex. sense inicialitzar). Si llavors correu el programa sota el debugger (sense cap breakpoint), quan s'aturi podreu veure la línia de codi on s'ha produït el `Segmentation fault` amb la comanda `bt`. Llavors amb la comanda `p` podeu veure valors d'índexs, etc., per veure si és aquesta línia on s'ha produït l'accés incorrecte a la memòria.[6]

I recordeu: la millor manera de depurar un programa és no equivocar-se. Així que repasseu el codi un parell de vegades abans de llançar-vos al debugger.

Trobareu més informació fent "`info gdb`" des d'una terminal (en particular, trobareu una sessió d'exemple molt instructiva).

5 Ús de gnuplot

És un programa per graficar, basat en línia de comandes. Permet dibuixar funcions definides per expressions que involucrin funcions elementals, però la seva potència radica en la facilitat amb la que es poden dibuixar fitxers de dades.

[6]Heu de tenir en compte que això no sempre passa: pot ser que en un punt del programa feu un accès invàlid a memòria que "desestabilitzi" el sistema de gestió de memòria dinàmica, de manera que el programa no casca aquí sinó més endavant. Llavors l'error és molt més difícil de trobar.

5.1 Dibuixos bidimensionals

Un fitxer de dades per al `gnuplot` és un fitxer de text que consta de vàries línies, on cada línia serà un punt de la gràfica. Un fitxer de dades pot constar de diversos "datablocks", que estan separats per línies en blanc.

Per a gràfiques bidimensionals s'usa la comanda `plot`. Per exemple, suposem que tenim un fitxer de text anomenat `dades.dat` amb contingut

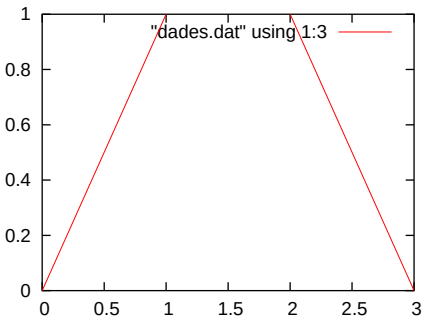
```
# Aixo es un comentari
# Primer datablock
0 1 0
1 1 1

# Segon datablock
2 1 1
3 1 0
```

Si entrem dins `gnuplot` i escrivim

```
plot "dades.dat" using 1:3 with lines
```

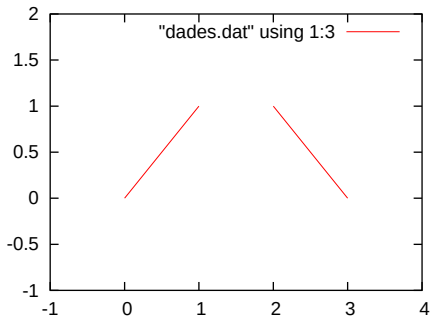
llavors gnuplot recorrerà el fitxer, per cada línia considerarà el punt amb coordenada x la columna 1 i coodenada y la columna 3, i unirà amb línies els punts consecutius de cada datablock (o sigui, a cada línia en blanc "aixequem el llapis"). El dibuix que obtindrem és:



Observeu que els rangs s'han ajustat perquè hi càpiguen exactament els 4 punts que s'han dibuixat. Els podem ampliar fent

```
set xrange [-1:4]; set yrange [-1:2]
replot
```

i surt



A `with lines` podeu canviar "`lines`" per "`points`", "`linespoints`", "`dots`", "`impulses`" i altres. Podeu comprovar que fan fent

```
set xrange [-pi:3*pi]
set autoscale y
plot sin(x) with lines
plot sin(x) with points
plot sin(x) with linespoints
plot sin(x) with dots
plot sin(x) with impulses
```

Sempre i quan no hi hagi lloc a confusió, les comandes es poden abreujar. Així, es pot fer

```
set autoscale
plot "dades.dat" u 1:3 w l
```

5.2 Sortida a fitxer

Una característica molt pràctica de gnuplot es que no només sap pintar a la pantalla, sinó que també sap pintar a fitxers en diversos formats gràfics. Per exemple, per guardar una gràfica de $\sin(x)$ en un fitxer `dibuix.png` en format PNG, heu de fer

```
set termimal png     # format: GIF
set out 'dibuix.png' # sortida a dibuix.png
plot sin(x)
set out              # per tancar el fitxer dibuix.png
set term x11         # per tornar a dibuixar per pantalla
```

Llavors, des d'una terminal, podeu mirar el dibuix que acabeu de generar fent

```
display dibuix.png
```

(dins gnuplot podeu fer "`!display dibuix.png`"). També podeu convertir aquest fitxer a qualsevol altre format gràfic amb la comanda `convert`. Per exemple, per convertir-lo en TIFF heu de fer (a una terminal, o dins gnuplot però amb "`!`" al davant)

```
convert dibuix.png dibuix.tif
```

Per saber en quins formats podeu gravar des de gnuplot, feu des de dins de gnuplot "`help set terminal`".