Entorn Sistema Operatiu GNU Linux

Fonaments dels Computadors – Grau en Intel·ligència Artificial – 2022-2023 Q1

Facultat d'Informàtica de Barcelona – Departament d'Arquitectura de Computadors

En aquesta pràctica de laboratori ens centrarem en agafar experiència en la distribució de Linux que tenim disponible en l'entorn dels ordinadors de la FIB (Suse), i també en l'Ubuntu, si l'heu instal·lat en el vostre ordinador.

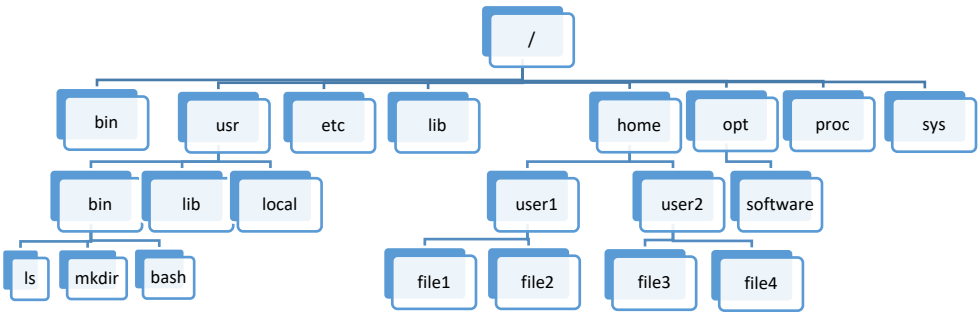
Examinarem les diferents eines que ens ofereixen els entorns GNU Linux. Obtindrem informació del Sistema Operatiu (SO), per entendre l'entorn d'execució, executar comandes i construir petits fitxers de comandes per fer tasques senzilles.

L'entorn GNU Linux

El sistema GNU Linux actual està basat en les idees desenvolupades en els sistemes operatius UNIX, que van iniciar-se el 1969 (Unix - Wikipedia), i que han anat evolucionant en diferents versions, algunes d'obertes, i altres d'empreses privades, fins que Linus Torwalds (Linus Torvalds - Wikipedia), el 1991 va iniciar el seu nucli de sistema operatiu, anomenat Linux (Linux - Wikipedia).

Quan entreu com a usuaris a un sistema GNU Linux, el sistema us posa en un directori concret dins el sistema de fitxers, habitualment amb un camí per arribar-hi similar a /home/<nom_d'usuari>. Aquest és el vostre directori d'entrada al sistema ("home directory").

El sistema de fitxers de GNU Linux té una estructura jeràrquica, molt similar independentment de la distribució que feu servir. La següent figura representa una estructura de sistema de fitxers, en general:



Durant el curs, anirem veient l'ús típic de cada directori en aquest esquema. Aquí en teniu un tastet:

- /: l'arrel del sistema de fitxers, el directori base, inicial, a partir del qual s'organitza la jerarquia de directoris i fitxers.
- bin: directori que conté comandes del sistema (ls, mkdir, vi..., bash...). N'hi ha un a l'arrel del sistema de fitxers (/bin) i un altre a dins del directori "usr" (/usr/bin).
- etc: directori on hi ha fitxers de configuració, i les bases de dades d'usuaris (/etc/passwd, /etc/shadow), i grups d'usuaris (/etc/group, /etc/gshadow).
- lib: directori que conté llibreries de suport (libc, libm, i moltes altres). També n'hi ha dos, un a l'arrel (/lib) i un altre a "usr" (/usr/lib). Pot tenir subdirectoris per classificar millor els fitxers de llibreries (/usr/lib/x86_64-linux-gnu).

- usr: inicialment pensat com un subdirectori per eines d'usuari, actualment és una extensió dels directoris del sistema operatiu amb, entre altres, una rèplica dels subdirectoris de l'arrel.
- home: el directori on es creen els subdirectoris dels usuaris del sistema.
- opt: directori on s'instal·len aplicacions de diferents origens, perquè els seus fitxers no entrin en conflicte amb fitxers del sistema operatiu.
- proc: directori on es publica informació sobre el sistema operatiu i els processos.
- sys: directori on es publica informació més estructurada i detallada del sistema operatiu.

El directori de treball d'un usuari de GNU Linux, el seu directori d'entrada o "home directori", per exemple /home/ubuntu, és on cada usuari situa els seus fitxers. Això inclou fitxers de dades, de codi font (C/C++, Python...), fitxers executables (o binaris) que són el resultat de compilar el codi font, i també fitxers de configuració del compte del mateix usuari (.login, .vimrc, .emacs...).

Quan entreu al sistema amb un entorn gràfic, el procés de login us situarà a dins del vostre directori d'entrada, i l'entorn gràfic visualitzarà les icones i fitxers que tingueu en el subdirectori "Escriptori".

Quan entreu al sistema amb una connexió de terminal, el procés de login us donarà un intèrpret de comandes (habitualment un bash, o un tcsh) que estarà en el vostre directori d'entrada. Podeu veure i crear nous fitxers i directoris a dins del vostre directori d'entrada i els seus subdirectoris. A fora del vostre directori d'entrada, habitualment només podreu veure els fitxers, però no canviar-los o esborrar-los, per no afectar el funcionament del sistema i dels altres usuaris. Aquest control d'accés als fitxers i directories es fa a través dels **permisos** que cada usuari té per accedir-hi.

Podeu navegar amb l'intèrpret de comandes cap a dins dels vostres directoris, per exemple, si teniu un directori per una aplicació que esteu desenvolupant anomenat "app_c", podeu executar:

```
$ cd app_c
```

i entrareu al directori "app_c". Executant "ls" podreu veure els fitxers i directoris que composen aquesta aplicació "app_c".

Si ets a dins del teu directori d'entrada (/home/ubuntu, per exemple), també pots "sortir" del vostre directori, executant:

```
$ cd ..     # que us portarà al directori /home
```

o bé

```
$ cd /     # que us portarà al directori arrel del sistema de fitxers (/)
```

Tots els directoris tenen dos fitxers especials que s'anomenen "." i "..":

"." : fa referència al propi directori. Per tant, la comanda $ cd "." es quedarà al propi directori on és.

".." : fa referència al directori pare de l'actual. Per tant, $ cd ".." ens portarà al directori pare.

Per referenciar fitxers en el sistema de fitxers, podem usar camins absoluts o relatius:

- el camí per arribar a un fitxer és **absolut** si indica els subdirectoris que cal travessar per arribar-hi des de l'arrel del sistema de fitxers (/). Per exemple "/usr/lib/x86_64-linux-gnu/libc.so" és el nom absolut de la llibreria que conté les funcions de suport al llenguatge C (típicament anomenada libc). Observeu que els camins absoluts sempre comencen amb l'arrel (/...).
- El camí per arribar a un fitxer és **relatiu** si indica els components que cal travessar per arribar-hi des del directori actual. En aquest cas necessitem saber on som dins el sistema de fitxers per trobar el fitxer que volem. Per exemple, si som a casa (/home/ubuntu) i volem arribar al mateix

fitxer “libc”, farem servir el camí “../../usr/lib/x86_64-linux-gnu/libc.so”. Els camins relatius comencen per un component diferent de l’arrel (../..., app_c.c, ./dades/clients.dat, ...)

Si estàs treballant amb l’OpenSUSE a la FIB, tots els comptes d’usuari tenen un directori “dades”, que té l’avantatge que la FIB en fa una còpia de seguretat (backup) cada nit. Pots posar dins aquest directori “dades” la feina que vulguis guardar per més endavant, preferiblement en algun subdirectori dedicat a l’assignatura. Per exemple, els codis font que desenvolupis a l’assignatura.

De tota manera, tingues també en compte que és millor desenvolupar les sessions de laboratori en un directori a fora del directori “dades”. La raó és que en aquest directori, la FIB proporciona accés a un altre sistema de fitxers, que ve proporcionat per una màquina Windows, i com a conseqüència, té un format sobre el qual no funciona alguna pràctica de laboratori. Per exemple, hem observat en altres quadrimestres, que configurar i compilar algunes aplicacions pot fallar per causes no massa ben determinades.

Els comptes que teniu a OpenSUSE estan limitats en la quantitat de dades que hi podeu guardar. Poden tenir un límit al voltant de 3GBytes. Podeu consultar la quota que teniu disponible i ocupada a través del Racó. Si puntualment necessiteu més espai, podeu utilitzar l’espai que hi ha al directori “/tmp”, durant la sessió actual. Però el contingut d’aquest directori s’esborrarà automàticament en reiniciar o apagar el PC. Podeu trobar els detalls aquí:

(https://www.fib.upc.edu/ca/la-fib/serveis-tic/entorn-de-treball-als-ordinadors)

Treballant amb l’intèrpret de comandes (shell)

Per començar, executarem un intèrpret de comandes (o Shell). Un intèrpret de comandes és un programa que ofereixen els sistemes operatius, i que ens permet treballar interactivament en mode text. L’entorn de text pot semblar menys intuïtiu i efectiu que l’entorn gràfic, però és realment senzill d’utilitzar, és més potent, i et permet conèixer molt millor l’entorn de l’ordinador on estàs treballant.

Hi ha diversos intèrprets de comandes disponibles en els sistemes operatius. A OpenSUSE, per defecte teniu l’intèrpret “tcsh”. A Ubuntu, disposeu per defecte del “bash” (GNU-Bourne Shell). Ens referirem a tots ells de forma general amb el nom d’intèrpret de comandes o Shell. La majoria de conceptes que expliquem en aquesta sessió poden fer-se amb tots els intèrprets de comandes, però la sintaxi que fem servir és la del “bash”. Si el sistema us ofereix un “tcsh”, podeu entrar en un “bash”, simplement teclejant:

$ bash

bash_$

També podeu trobar informació sobre els conceptes presentats en aquesta pràctica de laboratori, usant la comanda “man” del sistema. Per exemple:

$ man bash

Per executar un intèrpret de comandes, necessites obrir un terminal. Per exemple, usant l’aplicació “konsole”, que trobareu a l’Escriptori, o en el menú d’inici d’OpenSUSE o Ubuntu. Si no la trobeu, la podeu buscar en el cercador del menú d’inici. Aquesta aplicació obre una consola, una nova finestra, en la que s’executa el vostre intèrpret de comandes per defecte (el que indica el vostre usuari). Si us n’obre un de diferent del “bash”, us suggerim executar a sobre un “bash”, com hem indicat abans.

Pots també intentar canviar el teu intèrpret de comandes per defecte perquè sigui el "bash". Fes-ho usant la comanda change-shell ("chsh"). Desafortunadament, a l'entorn OpenSUSE de la FIB, la comanda "chsh" no funciona correctament, perquè la informació dels usuaris la proporciona el servei distribuït LDAP, enlloc del tradicional fitxer "/etc/passwd". I nosaltres, com a usuaris normals no podem canviar la informació a la base de dades del LDAP. Si ho voleu intentar, feu:

```
$ chsh
Password:       <enter your password here>
Changing the login shell for <your username>
Enter the new value, or press ENTER for the default
      Login Shell [/usr/bin/tcsh]:    /bin/bash
Unable to modify the user's shell, because the user is not in the
passwd file.
```

Si funcionés, el nou intèrpret de comandes s'engegaria el següent cop que entressiu al sistema, amb la qual cosa haurieu de sortir i tornar a entrar, perquè el canvi es fes efectiu.

Tots els intèrprets de comandes tenen dos tipus de comandes: externs i interns. Les comandes externes són qualsevol programa instal·lat en el sistema. Per executar-les, l'intèrpret de comandes crea un nou procés i l'utilitza per executar el programa. Les comandes internes són funcions implementades pel propi intèrpret de comandes. D'internes, cada intèrpret implementa les seves, algunes són comunes i d'altres particulars a l'intèrpret. Penseu per exemple, en la comanda "cd", que s'utilitza per canviar el directori actual. Es pot implementar com a una comanda externa?

Aconseguint ajuda

A Linux hi ha diverses comandes que poden ser executades localment al propi ordinador per obtenir ajuda interactivament:

- La comanda "man", que ofereix ajuda sobre els intèrprets de comandes, les comandes externes i qualsevol aplicació instal·lada que porti ajuda en el format "man".
- La comanda "info", de GNU, que també proporciona informació sobre les comandes externes i altres aplicacions que incorporin manuals en el format "info".
- La comanda "help", del "bash", que proporciona informació sobre les seves comandes internes.

Saber com fer servir el manual de GNU Linux ("man") és bàsic ja que, encara que us explicarem algunes comandes, no podem fer-ho amb totes (n'hi ha centenars!). Hauràs d'aprendre a fer servir el "man" i així completar els teus coneixements de les que no podem explicar, però tot i això poden ser-te força útils. La comanda del manual està també auto-documentada, executeu $ man man.

La informació que mostra la comanda "man" està organitzada en capítols. I cada comanda o peça del manual s'anomena una pàgina del manual (man page). Els capítols contenen comandes (capítol 1), crides a sistema (capítol 2), crides a serveis de llibreria (capítol 3), informació sobre drivers (capítol 4), fitxers de configuració (capítol 5), jocs (capítol 6), components del sistema (capítol 7) i comandes administratives (capítol 8). Hi ha també capítols que poden tenir una lletra associada al número del capítol ("1p", "3p"...), indicant que les comandes, crides, eines explicades en aquests capítols segueixen l'standard POSIX (The Open Group Base Specifications Issue 7, 2018 edition). Totes les pàgines del manual segueixen un format similar, organitzades en diferents seccions.

A la primera secció de les pàgines de manual, per exemple:

VIM(1) General Commands Manual VIM(1)

NAME

vim - Vi IMproved, a programmer's text editor

SYNOPSIS

vim [options] [file ..]

vim [options] –

vim [options] -t tag

vim [options] -q [errorfile]

...

hi trobem el nom de la comanda (vim), la seva descripció i un esquema (SYNOPSIS) de com utilitzar-la. En aquesta part, la pàgina mostra si la comanda accepta opcions, un o diversos noms de fitxers, si els paràmetres són obligatoris o opcionals... Les comandes acostumen a acceptar opcions curtes, d'una sola lletra (-t, -q...), amb arguments o sense, o arguments opcionals com [errorfile]. O també poden acceptar opcions llargues, habitualment amb dos guions (--verbose, --color...). O noms de fitxers, o altres cadenes de caràcters, per ser utilitzats/utilitzades durant l'execució de la comanda.

La següent secció de les pàgines de manual és la descripció (DESCRIPTION) de la comanda, amb explicacions sobre cadascuna de les opcions i fitxers que accepta i què fa en cada cas. En aquesta secció s'hi pot trobar també una explicació del valor que pot retornar la comanda quan acabi. Per exemple, el codi de retorn del procés que està executant la comanda pot ser 0 (zero), en cas que la comanda acabi correctament. O un número positiu 1, 2... indicant un error particular amb què s'ha trobar la comanda. 1 pot significar algun problema menor, 2 un problema més seriós, com que no hagi trobat un dels fitxers que li hem passat com a argument...

Finalment, les pàgines de manual acostumen a acabar-se, incloent els autors de la comanda i de la pàgina de manual, com informar els desenvolupadors de possibles errors en la implementació, algun exemple d'ús i referències a altres comandes relacionades en les que podem estar interessats.

La comanda "man" és una eina de sistema que sap mostrar per la pantalla els fitxers on hi ha la informació, en un forman determinat. Els autors de les pàgines de manual, indiquen el text a mostrar i en quin format, amb les seccions indicades, i l'eina "man" el mostra. La comanda "man", mentre està mostrant una pàgina accepta una sèrie de caràcters d'entrada, que permeten navegar per la pàgina amunt i avall, línia a línia o pantalla a pantalla, cercar text, i sortir del "man", entre d'altres. Molts cops, la comanda "man", de fet, usa una altra comanda típica de GNU Linux per fer aquesta visualització, com és el cas de les comandes "less" o "more". Les opcions més típiques per recordar són:

- Com que la major part de les pàgines de manual ocupen més d'una pàgina, per avançar a la següent pàgina, pots usar la tecla de l'espai.
- Per anar a la pàgina anterior, pots usar la tecla <b> (back).
- Les tecles <d> (down) i <u> (up) es mouen de mitja pàgina en mitja pàgina.
- La tecla <retorn> (<enter>) i la <j> van a la línia següent, i la <k> a l'anterior.
- Per buscar un text i anar directament allà on és, fes servir el caràcter </>, i teclejeu el text a continuació, seguit de la tecla de <retorn>. Per exemple, si teclejem "/SEE ALSO", anirem a la primera aparició del text "SEE ALSO", que habitualment serà la secció de referències d'altres comandes.
- Per anar a la següent aparició del mateix text, prem la <n> (next).

- I per sortir de la pàgina de manual, usa la tecla <q> (quit).

Exercici 1

Crea una jerarquia de directories per guardar-hi els exercicis que facis a cada sessió durant el curs. Pots anomenar "Sx" al directori de la sessió "x". Fes-ho amb la comanda "mkdir" i amb el gestor de fitxers de l'entorn gràfic. Practica també esborrant els directoris amb "rmdir".

Llista el contingut del directori actual amb la comanda llistar (ls). Hi ha diversos fitxers "ocults", que no es llisten per defecte. Tots els fitxers i directoris que comencen amb el caràcter "." són fitxers ocults, que la comanda "ls" no mostra per defecte. Habitualment tenen alguna característica especial. Busca quina opció de la comanda "ls" et permet veure tots els fitxers, inclosos els que comencen per ".". Els fitxers que hauries de veure llistats ara són:

- Directori ".": referencia el propi directori on és. Executant $ cd . veuràs que et quedes en el mateix directori. Després veurem la seva utilitat.
- Directori "..": referencia el directori immediatament superior al directori on és. Executant $ cd .. veuràs que canvies de directori al directori superior.
- Si ets al teu directori d'entrada al sistema (home), segurament veuràs els fitxers de comandes que donen suport a l'entrada al sistema (.profile, .cshrc...), i altres.

Nota que aquests fitxers i directoris ocults tampoc apareixen en el gestor de fitxers de l'entorn gràfic. Si entres al directori S1, possiblement apareixerà com a buit. En el gestor de fitxers també hi ha una opció de configuració que et permet habilitar que es vegin els fitxers "ocults".

Exercici 2

Obre el gestor de fitxers de l'entorn gràfic, i ves als directoris que has creat abans (S1, S2, ...). Observa que els fitxers que comencen amb "." no es mostren en el gestor, i això també passa amb noms com ".prova". Modifica la configuració del gestor de fitxers perquè sí que mostri aquests fitxers "ocults".

Edició Bàsica de fitxers

Per crear fitxers de text, ja sigui per editar les respostes a aquestes pràctiques de laboratori, o per editar fitxers de C o Python, tenim diverses opcions. GNU Linux incorpora una llarga llista d'aplicacions per a l'edició de fitxers: vi, joe, gedit, emacs...

Per exemple, obre un editor i escriu un text:

```
$ gedit  document.txt
```

En aquest cas, l'editor s'executarà i **l'intèrpret de comandes s'esperarà a que s'acabi la comanda actual** (l'edició en aquest cas). Per executar altres comandes mentre l'editor és obert tenim dues opcions:

- Obrir un altre terminal, per executar les comandes en ell.
- Posar l'editor en **segon pla** (background), amb la seqüència de control "control-Z" (^Z)

Això últim podem fer-ho automàticament, executant de bon començament:

```
$ gedit document.txt &
[1] 326
$
```

amb el caràcter "&" al final de la comanda, l'intèrpret de comandes executa la comanda en **segon pla** automàticament. És a dir, no s'espera a que acabi el "gedit". I ens indica que s'ha creat el job [1], que té d'identificador de procés el número 326. Amb aquest mètode, l'intèrpret de comandes executa la comanda i immediatament mostra l'indicador d'entrada de comandes (el prompt), esperant per la propera comanda. Per defecte, l'intèrpret de comandes executa les comandes en **primer pla** (foreground).

Per veure ràpidament el contingut d'un fitxer, sense obrir l'editor un altre cop, podem usar les comandes que hem comentat abans: *less*, *more* i també el *cat*. Afegeix al fitxer document.txt 3 o 4 pàgines de text (pots copiar i enganxar text des d'aquest mateix document, per exemple). I llavors prova:

```
$ cat document.txt   # veuràs que el "cat" simplement volca tot el fitxer a la sortida, sense cap control

$ less document.txt   # en aquest cas tindràs el mateix control que amb les pàgines de manual

$ more document.txt  # amb more, tindràs una sortida similar al less, també amb control de pàgina

$ cat document.txt | less
```

Aquest últim exemple és el primer encadenament de comandes que fas (anomenats pipeline). Amb els pipelines, la sortida d'una comanda (com el cat) es passa directament a la comanda que hi ha després del símbol de pipe (|), i aquesta rep la informació i pot formatar-la per visualitzar-la amb comoditat.

```
$ cat document.txt | sort | less
```

Amb aquest encadenament de comandes, la informació obtinguda del fitxer document.txt (cat) es passa a la comanda "sort", que ordenarà les línies alfabèticament i les passarà al "less" per visualitzar-les.

Copia (*cp*) el fitxer "document.txt" diversos cops amb diferents noms (per exemple, afegint un número diferent al final – document1.txt, document2.txt...). Què passaria si els noms dels fitxers origen i destí tinguessin el mateix nom? Què diu la comanda "cp"?

Què fa la comanda "cp" si el fitxer destí ja existeix? Busca una opció de la comanda "cp" que eviti sobreescriure el fitxer destí, si aquest ja existeix.

Crea un àlies (comanda alias) que et permeti usar "cp" amb aquesta opció que has trobat per evitar sobreescriure els fitxers destí.

Prova d'esborrar (rm) alguns dels fitxers que has creat, i canvia el nom (mv) d'altres fitxers. Fes-te un àlies de la comanda "rm" que inclogui l'opció "-i" (pots dir-li "rmi"), i busca també el significat de l'opció "-i" de la comanda "mv".

Pots fer que aquests canvis siguin permanents per les teves sessions de treball futures, incorporant les comandes "alias" en els teus fitxers d'entrada al sistema. Tingues en compte que la sintaxi canvia entre els diferents intèrprets de comandes:

- **$HOME/.bashrc** (si uses bash, per exemple " alias rmi='rm -i' ")
- **$HOME/.cshrc** (usant csh o tcsh, per exemple " alias rmi 'rm -i' ").

Fitxers i Permisos

La comanda "ls -l" visualitza els permisos que tenen els fitxers i directoris. En GNU Linux, els permisos més bàsics s'apliquen en 3 nivells:

- El propietari (usuari) del fitxer (owner, o "u" per usuari)
- El grup d'usuaris (group, o "g")
- La resta d'usuaris (other, o "o")

I els permisos fan referència a 3 operacions possibles, o modes d'accés: lectura (read – r), escriptura (write – w), i execució (execute – x). Per exemple, si en el directori actual hi ha només un fitxer (f1), i aquest fitxer té permisos de lectura i escriptura pel propietari del fitxer, de només lectura pels membres del grup i cap permís per la resta d'usuaris del sistema, la sortida de la comanda "ls -l" donarà el següent resultat:

FTYPE/PERM	NLINKS	OWNER	GROUP	SIZE	DATE	FILENAME
- rw- r-- ---	1	xavi	users	17410312	Feb 3 21:12	f1

La primera columna de la sortida indica el tipus de fitxer i els permisos d'accés que té. El primer caràcter indica el tipus de fitxer, que pot ser:

- "-": l'entrada és un fitxer normal, de dades. Ho són els fitxers de text, executables i de dades
- "d": l'entrada és un subdirectori
- "p": l'entrada és una pipe amb nom, que es pot fer servir per passar dades entre processos
- "s": l'entrada és un socket amb nom, que es pot fer servir per comunicar diferents processos
- "c": l'entrada representa un dispositiu de caràcter, com "/dev/null"
- "b": l'entrada representa un dispositiu de bloc, habitualment un disc o partició, com "/dev/sda1"

Els següents caràcters, agrupats de 3 en 3, representen, en ordre, els permisos que el propietari (rw-), el grup d'usuaris (r--) i la resta d'usuaris (---) tenen sobre el fitxer. En l'exemple, l'usuari té permisos de lectura(r) i escriptura(w), però no d'execució(-).

Aquests permisos es poden canviar amb la comanda "chmod" (change-mode). La comanda permet especificar els permisos de diferents maneres, però la més intuïtiva usa els mateixos símbols per afegir o treure les proteccions. Per exemple, si volem afegir permisos d'execució pel propietari pel fitxer f1, faríem:

```
$ chmod  u+x  f1
```

```
$ ls -l  f1
```

FTYPE/PERM	NLINKS	OWNER	GROUP	SIZE	DATE	FILENAME
- rwx r-- ---	1	xavi	users	17410312	Feb 3 21:12	f1

Si ara volem donar permisos d'escriptura i execució al membres del grup, faríem:

```
$ chmod  g+wx  f1
```

```
$ ls -l  f1
```

FTYPE/PERM	NLINKS	OWNER	GROUP	SIZE	DATE	FILENAME
- rwx rwx ---	1	xavi	users	17410312	Feb 3 21:12	f1

I si ara volem treure el permís d'escriptura als membres del grup, faríem:

```
$ chmod  g-w  f1
$ ls -l  f1
```

FTYPE/PERM	NLINKS	OWNER	GROUP	SIZE	DATE	FILENAME
- rwx r-x ---	1	xavi	users	17410312	Feb 3 21:12	f1

I si ara volem donar permisos de lectura i execució també a la resta d'usuaris del sistema, faríem:

```
$ chmod  o+rx  f1
$ ls -l  f1
```

FTYPE/PERM	NLINKS	OWNER	GROUP	SIZE	DATE	FILENAME
- rwx r-x r-x	1	xavi	users	17410312	Feb 3 21:12	f1

Els permisos són un mapa de bits, de forma que si el permís està a 0, no es pot accedir, i si està a 1, sí que es pot. Si agrupem els bits de 3 en 3, podem interpretar un mapa de bits com 111 101 101 (=755 en base octal) i usar la comanda chmod així:

```
$ chmod  755  f1
```

per aconseguir els mateixos permisos que en el darrer exemple, rwx r-x r-x.

Exercici 3

*Crea un fitxer anomenat "**respostes.txt**" i explica com modificar els permisos del fitxer "document.txt" per tenir només permís d'escriptura pel propietari, el grup d'usuaris i la resta d'usuaris del sistema. Explica què passa quan intentes fer un "$ cat document.txt".*

Caràcters Especials de l'Intèrpret de Comandes

En les seccions anteriors hem introduït l'ús del caràcter &, com la manera d'executar una comanda en segon pla. Altres caràcters o combinacions de caràcters molt útils quan donem comandes a l'intèrpret de comandes són:

- * (asterisc): l'intèrpret de comandes substitueix el caràcter '*' per qualsevol grup de caràcters (excepte un '.' al començament) que trobi en els noms dels fitxers que hi ha en el directori que s'analitza. Per exemple, si executem $ ls d*, veurem tots els fitxers i directoris que comencen per "t". Aquests caràcters especials de l'intèrpret de comandes es poden fer servir amb qualsevol comanda, perquè és l'intèrpret qui els resol, abans d'executar la comanda.
- ? (interrogant): l'intèrpret de comandes substitueix el caràcter '?' per un sol caràcter que trobi en els noms dels fitxers que hi ha en el directori que s'està analitzant. Per exemple, si executem $ ls document?.txt, veurem tots els fitxers que coincideixin amb el nom "document" + 1 caràcter qualsevol + ".txt", en particular, "document**2**.txt" o "document**h**.txt".
- > "fitxer": la sortida per defecte de la comanda afectada per aquesta seqüència es redirecciona al "fitxer" indicat. Si no la fem servir, la sortida de les comandes va habitualment al terminal on les executem. Amb aquesta seqüència podem canviar aquesta associació. Aquesta acció s'anomena "redirecció de la sortida" (output redirection). Per exemple, "$ ls > output_ls", guarda la sortida de la comanda "ls" en el fitxer "output_ls". Prova d'executar aquesta

comanda. Després, prova la mateixa redirecció amb una altra comanda. I comprova el contingut del fitxer "output_ls". Què ha passat?

- >> "fitxer": la sortida per defecte de la comanda afectada per aquesta seqüència s'afegeix al fitxer indicat ("fitxer"). La diferència amb l'anterior '>' és que el fitxer no es trunca, sinó que es manté la informació que ja hi havia i s'afegeix la nova al final. Repeteix l'exemple anterior amb les dues comandes redireccionades cap a "output_ls" ara usant '>>' per comprovar que efectivament la informació s'afegeix sense esborrar l'anterior.
- | (encadenament, o *pipe*): com hem vist abans, aquest caràcter permet que la informació de la sortida de la comanda de la seva esquerra, passi a l'entrada de la comanda que posem a la dreta. És com si estiguessin connectades a través d'una canonada.

Durant el curs també veurem aquests, que són prou útils:

- < "fitxer": aquest caràcter s'usa per indicar que l'entrada per defecte de la comanda afectada, vindrà del "fitxer", enlloc de fer la lectura del terminal.
- 2> "fitxer": aquesta combinació indica que la sortida d'error de la comanda afectada vagi al "fitxer". Per defecte la sortida d'error va al mateix lloc que la sortida estàndard, al terminal.
- >& "fitxer": aquesta combinació indica que es redireccioni tan la sortida estàndard, com la sortida d'error, al "fitxer".
- Una comanda pot executar-se usant diverses d'aquestes seqüències, de forma que tenim la màxima flexibilitat per fer redireccions:

```
$ cat  < document.txt  | grep "sortida" >output.txt   2>error.txt
```

Canals estàndard d'entrada i sortida

Amb el que hem comentat a la secció anterior sobre les redireccions d'entrada i de sortida, observeu que tant l'intèrpret de comandes, com les comandes que executem, com tots els processos que s'estan executant en el nostre sistema, tenen 3 canals de comunicació oberts per defecte que, per convenció, s'utilitzen de la següent manera:

- stdin (canal 0, descriptor de fitxer 0, entrada estàndard): el canal 0, stdin (en C) o std::cin (en C++) és el canal que s'utilitza per defecte per llegir informació de forma estàndard. Per defecte està lligat amb el terminal on s'executa la comanda. Un procés pot obrir altres fitxers per lectura, que usaran altres identificadors de canal (3, 4, 5...), a no ser que es tanqui prèviament el canal 0.
- stdout (canal 1, descriptor de fitxer 1, sortida estàndard): el canal 1, stdout (en C) o std::cout (en C++) és el canal que s'utilitza per defecte per escriure informació de forma estàndard. Per defecte està lligat amb el terminal on s'executa la comanda. Un procés pot obrir altres fitxers per escriptura, que usaran altres identificadors de canal (3, 4, 5...), a no ser que es tanqui prèviament el canal 1.
- stderr (canal 2, descriptor de fitxer 2, sortida d'error estàndard): el canal 2, stderr (en C) o std::cerr (en C++) és el canal que es recomana per escriure per defecte els missatges d'error dels programes que funcionen sobre GNU Linux. Per defecte està lligat amb el terminal on s'executa la comanda.
- C++ també ofereix el canal std::clog, que es mapeja sobre el mateix canal d'error (std::cerr, stderr, canal 2).

Filtering/Searching

La comanda *grep* ens permet buscar seqüències de caràcters en fitxers de text. Poden ser seqüències constants o patrons. Per provar-ho, pots afegir una paraula concreta en un dels fitxers document*.txt

que has copiat. Prova de buscar-la amb la comanda *grep*. Per exemple, afegeix la paraula "hola" en un dels fitxers i executa aquesta prova:

```
$ grep "hello" document*.txt
```

Exercici 3

*Respon en el teu fitxer "**respostes.txt**" com filtrar la llista de fitxers i directoris en el directori actual, per mostrar només els subdirectoris. Explica com fer-ho i les comandes encadenades per una pipe que en resulten. Pista: busca a la pàgina de manual del grep, com buscar un text que apareix al principi de la línia (anchoring).*

En el fitxer de suport d'aquesta pràctica trobareu exemples per fer el mateix en C++17 i Python.

Sistemes de Fitxers

Els fitxers estan organitzats en sistemes de fitxers. Un sistema de fitxers és una col·lecció de fitxers, directoris i fitxers especials (de caràcter, bloc, pipes, sockets...) que comença en una arrel (root directory, top level directory). Compara els sistemes de fitxers amb les diferents particions que tenim en el sistema MS Windows: C:, D:, ...

La comanda *mount* llista el conjunt de sistemes de fitxers disponibles en l'entorn d'execució, i el directori on estan muntats. Per exemple:

```
# /proc és el sistema de fitxers que conté informació sobre els processos
  proc          on    /proc  type    proc ...
# /sys és el sistema de fitxers que conté informació sobre el sistema operatiu
  sysfs         on    /sys   type    sysfs ...
# / és l'arrel (root) absoluta de tots els sistemes de fitxers en l'entorn, muntada en el disc local
  /dev/sda1  on   /        type    ext4 ...
# el sistema de fitxers "dades" està muntat en una unitat de disc en xarxa
  //pax/dades   on   ...
# el vostre directori d'entrada al sistema està muntat en una unitat de disc en xarxa
  libra.fib.upc.edu:/... on /home2/...
```

Exercici 5

*Mira el llistat del sistema de fitxers arrel (root - /). Intenta entendre l'ús dels diferents subdirectoris que hi trobes (només els de primer nivell). Explica el que trobes en el teu fitxer "**respostes.txt**".*

Configuració d'àlies per comandes freqüents

Com hem explicat abans, per comandes usades freqüentment, és possible definir àlies que simplifiquin la quantitat de caràcters que hem d'escriure al terminal. Podem considerar que un àlies ens abreujarà comandes concretes amb els seus arguments més habituals. Els àlies consisteixen en la definició d'un nom que l'intèrpret de comandes reconeixerà i substituirà pel la traducció que li hem proporcionat en

la seva definició. Per exemple, si veiem que molts cops executem la comanda "ls" amb les opcions "-la", és fàcil definir-la en el nostre fitxer d'entrada al sistema, d'aquesta manera:

(bash) $ alias ls='ls -la'

(csh, tcsh) $ alias ls 'ls -la'

Executa aquest àlies i després executa "ls" sense opcions. Comprova que es tradueix automàticament per "ls -la" i la sortida és la corresponent al format llarg de l'"ls" i mostra els fitxers ocults.

Pots fer aquest tipus de definicions permanents per les futures sessions, afegint-la en els fitxers d'entrada al sistema: **$HOME/.bashrc** (usant bash), i **$HOME/.cshrc** (usant csh o tcsh).

Variables d'Entorn

Els programes s'executen en un entorn i context determinats: un procés del sistema operatiu. Un procés pertany a un usuari i un grup, té un directori de treball determinat (el directori actual del procés, o current directory), amb una sèrie de límits en el temps d'ús de processador, de memòria, de mida de fitxers, de quantitat de fitxers oberts... Més detalls al respecte aniran apareixent durant el curs.

En aquesta sessió, introduïm les variables d'entorn, que també formen part de l'entorn del procés. Són similars a les constants que es poden definir dins dels programes en C i C++, però en el cas de les variables d'entorn es defineixen abans d'executar el programa. Habitualment fan referència a algun aspecte important de l'execució. Per exemple, poden referir-se a directoris per defecte útils durant l'execució, o valors concrets per configurar aspectes de l'execució, com l'idioma a fer servir pels missatges, colors en els que presentar la informació, etc. Algunes de les variables d'entorn habituals les fan servir també els intèrprets de comandes, per variar el seu comportament. Per exemple, l'indicador d'entrada (prompt) habitualment es treu de la variable d'entorn PS1 (bash).

Les variables d'entorn habitualment es defineixen usant lletres majúscules i dígits, però no és necessari. Els valors assignats a les variables d'entorn poden ser accedits durant l'execució d'un programa a través de funcions de la llibreria de C (getenv, setenv). Per aprendre més detalls sobre les variables d'entorn que l'intèrpret de comandes fa servir, pots buscar-les a les pàgines de manual de l'intèrpret que fas servir. Per exemple pel "bash": $ man bash i busca "Shell Variables".

Executa la comanda "env" per veure la llista de variables (amb els seus valors) que estan definides en l'entorn de l'intèrpret actual. Per indicar a l'intèrpret de comandes que volem veure el valor d'una variable, usarem el símbol $ al davant del nom.

L'intèrpret de comandes té una comanda (habitualment interna), echo, que permet veure aquests valors:

```
$ echo $PS1
```

Amb el símbol de $ diferenciem entre el text literal "PS1", de la variable d'entorn PS1. Altres exemples poden ser:

```
$ echo $USERNAME        or        $ echo $PWD
```

També podem definir noves variables o modificar les existents, usant la següent comanda interna de l'intèrpret de comandes (observeu que per assignar-los un valor no s'usa el símbol de "$"):

```
$ export   VARIABLE_NAME=value    # (sense deixar espais al voltant del símbol '=').
```

Exercici 6

Algunes variables d'entorn són actualitzades automàticament per l'intèrpret de comandes en executar comandes internes. Per exemple, canviar el directori actual (cd) canvia el valor de la variable PWD. Prova-ho, mira la variable PWD, canvia de directori i torna a mirar la variable PWD.

Comprova també els valors de les variables USERNAME, PATH i HOME. Aquestes són fixes per la sessió, a no ser que les canviem manualment.

*Afegeix al teu fitxer "**respostes.txt**" una explicació de per a què serveix cadascuna d'aquestes variables: PWD, USERNAME, PATH i HOME.*

Alguns cops és interessant mirar en detall el contingut dels fitxers que fem servir. Una primera observació te la pot donar la comanda "file", que informa l'usuari sobre el contingut aproximat dels fitxers que rep com a argument:

```
# file  *.c
program.c:              C source, ASCII text
extractsymbols.c:       C source, UTF-8 Unicode text
```

Una segona opció, més detallada i de vegades difícil d'entendre és observar el contingut exacte dels fitxers en decimal, hexadecimal, octal o en format caràcter:

```
# od  -t  cd1x1   program.c   # mostra el contingut en caràcters(c),
                              # bytes en decimal (d1) i bytes en hexadecimal(x1)
0000000     #     i     n     c     l     u     d     e           <     s     t     d     i     o     .
         35   105   110    99   108   117   100   101    32    60   115   116   100   105   111    46
         23    69    6e    63    6c    75    64    65    20    3c    73    74    64    69    6f    2e
0000020     h     >    \n     #     i     n     c     l     u     d     e           <     s     t     d
        104    62    10    35   105   110    99   108   117   100   101    32    60   115   116   100
         68    3e    0a    23    69    6e    63    6c    75    64    65    20    3c    73    74    64
...
```

Una altra comanda semblant a l'"od" és "xxd", per mostrar el contingut exacte dels fitxers en format hexadecimal.

Exercici 7

Executa les següents línies de comanda:

```
$ echo hello world > fitxer.txt

$ echo goodbye world >> fitxer.txt      # observa la redirecció per afegir (>>) aquí
```

per crear un fitxer anomenat "fitxer.txt", amb el contingut "hello world". Amb aquest fitxer, compara la sortida de les comandes:

```
$ file fitxer.txt

$ od -xc fitxer.txt

$ od -t x1c fitxer.txt

$ xxd  fitxer.txt
```

Veureu que les sortides de l'od i l'xxd donen un parell de caràcters '\n' (en hexadecimal 0a) a dins del fitxer.

Explica les conclusions a les que arribes amb la comparativa, i què pot significar el caràcter '\n' a dins del teu fitxer "**respostes.txt**".

Envia'ns l'informe de la pràctica

*Empaqueta el fitxer "**respostes.txt**" en un fitxer empaquetat i comprimit, pots fer servir aquesta comanda:*

```
$ tar  czvf  sessio2.tar.gz  respostes.txt
```

(pots veure què fa la comanda "tar", amb la seva pàgina de manual: $ man tar)

I ara pots anar al RACÓ i enviar-nos el fitxer comprimit (sessio2.tar.gz) a la pràctica corresponent.