Resum de comandes GNU/Linux emprant la shell Bash

***Avantatges d'utilitzar la "Command Line Interface (CLI)" en comptes de la "Graphical User Interface (GUI)":**

-Més ràpida
-Més completa
-Reusable

***Cal distingir entre el shell pròpiament dita i els programes que l'envolten:**

El shell és el programa que realment entén i executa les comandes que nosaltres escrivim. Als sistemes GNU s'utilitza el shell "bash", encara que es poden utilitzar altres, com ara el "dash", el "zsh", el "ksh", el "tcsh", etc. Segons el shell utilitzat, les comandes s'hauran d'escriure seguint unes determinades regles "sintàctiques". Nosaltres només veurem com funciona el shell "bash", que és el shell més àmpliament utilitzat. Per obrir un shell "bash" podem fer dues coses diferents:

-Accedir a un terminal virtual (pulsant les tecles CTRL+ALT+F1 ó F2 ó F3...). Fent això haurem d'iniciar una nova sessió amb un nou usuari i contrasenya; un cop fet això, automàticament ens situarem a l'interior de la carpeta personal d'aquest usuari (és a dir, /home/nomdelusuari). Els terminals virtuals s'identifiquen amb el nom tty1, tty2, etc

-Executar un programa "embolcall" dins del nostre l'escriptori (com ara el "gnome-terminal" a GNOME o el "konsole" a KDE, entre molts altres), el qual té una sèrie d'opcions de configuració personalitzables (font de lletra, colors, etc). En aquest cas, accedirem directament a l'interior de la carpeta personal de l'usuari que hagi iniciat sessió a l'escriptori. Les diferents finestres "embolcall" obertes, s'identifiquen amb el nom pts/0, pts/1...

Una característica important del shell "bash" és que les comandes i els noms dels fitxers i carpetes es consideren diferents segons s'escriguin en majúscules o minúscules. És a dir, no és el mateix escriure "Hola" que "hola" que "hoLa". En general, totes les comandes s'escriuen sempre en minúscules.

***El "prompt" ens mostra molta informació:**

-Usuari actual
-Nom de la màquina
-Carpeta on estem actualment. Per defecte en entrar al shell aquesta sol ser la nostra carpeta personal /home/nomusuari, també identificada amb el símbol ~
-Si som usuari "root" (#) -és a dir, administrador- o no ($)

***Primeres comandes:**

echo “una frase” Repeteix la frase indicada.

echo -e “frase1\n\tfrase2” El símbol “\n” serveix per indicar un salt de línia. El símbol “\t” indica tabulador

echo -e "\e[0m Negre \e[31m Vermell \e[32m Verd \e[0m" Per saber tots els codis de color (de lletra, de fons) i d'efectes (negrita, subratllat, etc) es pot consultar http://misc.flogisoft.com/bash/tip_colors_and_formatting

clear Neteja la pantalla

date Mostra la data i hora actual

date +"%D %T" Mostra la data i l'hora actual. Hi ha moltes més possibilitats de formateig llistades al manual (*man date*). Destacarem %Y (any), %m (mes), %d (dia), %H (hora), %M (minuts) i %S (segons)

date -s “nºany-nºmes-nºdia nºhora:nºmin:nºseg” Canvia la data i/o hora per l'especificada. També es pot fer el mateix amb la comanda *date nºmesnºdianºhoranºminnºany.nºseg* . Altres opcions més exòtiques són: *date -s "tomorrow" ; date -s "yesterday" ; date -s "next year" ; date -s "last year" ; date -s "next month" date -s "last month" ; ...*

La comanda *date* només canvia la data mentre el sistema estigui encés. Per a fer un canvi d'hora permanent, cal fer que l'hora del sistema (ja modificada) passi a ser l'hora hardware; això es pot fer amb la comanda *hwclock -w* També es pot fer a l'inversa: fer que l'hora hardware sigui la nova hora del sistema; això es fa amb la comanda *hwclock -s* En qualsevol cas, es pot veure quina és l'hora hardware actual amb la comanda *hwclock*

A sistemes moderns, en comptes de la comanda *date* és millor fer servir la comanda *timedatectl* i per canviar-la, (ja sigui la data, l'hora o les dues coses, *timedatectl set-time "nºany-nºmes-nºdia nºhora:nºmin:nºseg"* . Aquesta comanda també és capaç de sincronitzar el nostre rellotge amb servidors NTPs públics d'Internet si s'executa així: *timedatectl set-ntp yes*

La zona horària ve indicada a un arxiu anomenat /etc/localtime, que no és més que un accés directe a /usr/share/zoneinfo/Europe/Madrid (o l'arxiu associat a la zona indicada a la instal.lació del sistema). Canviar de zona horària, doncs, seria tan senzill com fer que /etc/localtime apuntés a un altre arxiu dins de /usr/share/zoneinfo (aquests arxius s'instal.len amb el paquet "tzdata"). Això es pot fer "manualment" amb la comanda *ln -s* (que veurem posteriorment) o bé, més fàcil, amb la comanda *timedatectl set-timezone "Europe/Madrid"* (també és interessant *timedatectl list-timezones)*

cal nºmes nºany Mostra el mes en format calendari (de l'any especificat). Si no s'especifica res, es mostra el mes actual. Es pot indicar només l'any per veure el calendari complet

man comanda Mostra l'ajuda de la comanda (per a què serveix, els paràmetres que té, etc)
Notar que si els paràmetres apareixen entre claudàtors, indica que són opcionals.
Per navegar per les pàgines del manual, a més de les tecles AvPag i RePag i els cursors, són molt útils les tecles “/” (per buscar una paraula), “n” (per anar a la següent ocurrència de la paraula buscada), “N” (per anar a la ocurrència anterior de la paraula buscada) i “q” per sortir del manual

exit Tanca el shell

***Treballar amb fitxers i carpetes:**

Als sistemes Linux existeix una carpeta (anomenada “arrel” i que s'especifica amb el símbol “/”) de la qual penjen tota la resta de carpetes. A diferència de Windows, no existeixen les lletres d'unitats (C:, D:...) sinò que tot està integrat dins d'un “arbre de carpetes” que parteix d'aquest únic punt comú: la carpeta “arrel”. Directament dins de la carpeta arrel existeixen una sèrie de carpetes estàndar (https://es.wikipedia.org/wiki/Filesystem_Hierarchy_Standard), com:

/boot Aquí es troba el gestor d'arranc (Grub) i el kernel Linux (els fitxers “vmlinuz” i “initrd”)
/usr/bin Aquí es troben els executables que poden ser executats per qualsevol usuari
/usr/sbin Aquí es troben els executables que poden ser executats només per l'usuari “root”
/usr/lib Aquí es troben els fitxers necessaris per a què funcionin tots els executables (les anomenades “llibreries”)
/usr/share Aquí es troben els elements de les aplicacions que són independents de l'arquitectura (la documentació està sota "doc", les pàgines del manual estan sota "man/idioma/manN", les icones, imatges, sons, temes, etc).
/opt Aquí s'han de posar els programes instal.lats manualment (és a dir, no provinents de repositoris)
/etc Aquí es troben els fitxers de configuració del sistema i de tots els programes instal.lats
/home Aquí es troben les carpetes personals dels diferents usuaris del sistema. Cadascú té una diferent.
/root Carpeta personal de l'usuari “root”
/tmp Aquí es troben els fitxers temporals que apareixen durant el funcionament dels programes estàndar
/run Aquí es troben fitxers de tipus .pid, .lock, etc dels processos que s'estan executant en aquest moment
/media Aquí es monta automàticament el contingut dels dispositius "removibles" (llàpisos USB, CDs i DVDs, etc)
/lost+found L´usa fsck per escriure dades inconsistents.
/var/log Aquí es troben els fitxers de registre del sistema, útils per consulta què ha passat i què està passant
/var/cache Aquí es troben la "catxé" de diferents programes (és a dir, fitxers que aquests solen utilitzar més sovint)
/var/spool Aquí es troben els treballs d'impressió pendents de sortir cap a la impressora
/dev Aquí es troben els fitxers que representen els diferents dispositius que el sistema pot reconèixer i utilitzar. Aquesta carpeta normalment l'usuari no l'utilitza...és una representació interna que fa servir el kernel pel seu propi funcionament. Aquí trobem sda, sdb, sdc...(discos durs), sda1,sda2, sda3, sdb1,sdb2... (particions d'aquest discs durs), sr0, sr1 (lectores de CD/DVD), a més d'altres dispositius “especials” com ara /dev/null (“forat negre”), /dev/zero (generador de zeros) ó /dev/urandom (generador de números aleatoris), els terminals virtuals (/dev/tty1, /dev/tty2,...), els terminals gràfics (/dev/pts/0, /dev/pts/1,...), dispositius d'entrada (ratolins, teclats.../dev/input/...), entre molts altres.
/sys Aquí es troben la configuració actual "en calent" dels diferents dispositius connectats als sistema (discos durs, tarjes de xarxa, etc). No és una carpeta "estàndar" ja que el seu contingut desapareix quan s'apaga la màquina i es torna a regenerar -sempre a la memòria RAM i amb el format "sysfs"- en el següent reinici.
/proc Aquí es troba la informació "en calent" de tots els processos que estan funcionant en aquest moment al sistema, a més d'altres informacions utilitzades pel kernel. No és una carpeta "estàndar" ja que el seu contingut desapareix quan s'apaga la màquina i es torna a regenerar -sempre a la memòria RAM i amb el format "procfs"- en el següent reinici.

Una manera de veure el sentit de les carpetes anteriors és observant quins fitxers es copien (i on!) quan instal.lem un paquet qualsevol. A sistemes Debian/Ubuntu això es pot veure executant la comanda *dpkg -L nompaquet* i a sistemes RedHat/Fedora executant la comanda *rpm -ql nompaquet* . D'altra banda, també és interessant fer a l'inversa: esbrinar a quin paquet pertany un determinat fitxers; a sistemes Debian/Ubuntu això es pot veure executant la comanda *dpkg -S /ruta/fitxer* i a sistemes RedHat/Fedora executant la comanda *rpm -qf /ruta/fitxer* Fent això per exemple amb la comanda *ls* (que estudiarem de seguida) es pot veure que aquesta comanda pertany al paquet "coreutils" (disponible també a Windows!) i que dins d'aquest paquet es troben moltes altres comandes que també estudiarem, com *rm, cp*...

D'altra banda, per saber la ruta on es troba un determinat executable (/usr/bin, /usr/sbin, ...) es pot executar la comanda *which nomExecutable* o bé la comanda *type [-t] nomExecutable* (aquesta darrera informa a més del tipus d'executable que és -si és un binari extern, una comanda pròpia interna del shell, un àlies, etc-).

pwd	Mostra a quina carpeta som actualment (el seu nom ve de "print working directory")

cd unacarpeta	Em mou dins la carpeta “unacarpeta” Se suposa que “unacarpeta” es troba al mateix lloc on sóc jo ara mateix
cd unacapeta/unaaltra	Em mou dins la carpeta “unaaltra”, la qual està dins de “unacarpeta” Se suposa que “unacarpeta” es troba al mateix lloc on sóc jo ara mateix
cd /unacapeta/unaaltra	Em mou dins la carpeta “unaaltra”, la qual està dins de “unacarpeta” Se suposa que “unacarpeta” es troba directament dins de la carpeta arrel (“/”).

És molt important distingir entre escriure la “/” inicial o no en una ruta d'una carpeta. Si s'escriu, aquesta ruta sempre començarà des de la carpeta arrel independentment d'on estigui situat jo en aquell moment. Per això aquesta manera d'escriure les rutes s'anomena “**absoluta**”. En canvi, si no s'escriu, aquesta ruta començarà des del lloc on estic situat; això implica que la mateixa ruta escrita en un lloc o en un altre donarà resultats diferents. Per això aquesta manera d'escriure les rutes s'anomena “**relativa**”.

cd . El símbol “.” indica sempre la carpeta on sóc jo ara mateix (sigui quina sigui)
Per tant, en aquest cas, la comanda cd no fa res, realment

cd .. El símbol “..” indica sempre la carpeta per damunt de la què jo sóc ara mateix

cd ~ El símbol “~” indica sempre la carpeta personal de l'usuari actual
També es pot indicar la carpeta personal d'un deterninat usuari així: *~nomusuari*

ls unacarpeta Mostra el contingut d'”unacarpeta” (si no s'indica, es mostrarà la carpeta on som)
No cal dir que s'apliquen les mismes regles ja vistes per tal d'indicar la ubicació d'”unacarpeta” (la “/” inicial, el símbols ., .. i ~, etc). Això serà així sempre.

La comanda *ls* té una sèrie de “modificadors” (paràmetres) interessants:

-a Mostra també els arxius ocults (que són els que el seu nom comença amb un punt)
-l Mostra informació sobre cada fitxer o carpeta llistada. Concretament, d'esquerra a dreta:
Diu si l'element es una carpeta ("d"), un fitxer ("-") o un accés directe ("l")
Diu els permisos que té l'element (veure explicació comanda *chmod*)
Diu el número d'enllaços forts que té l'element (veure explicació comanda *ln*)
Diu l'usuari propietari de l'element i el seu grup (veure explicació comanda *chown*)
Diu el tamany de l'element (només si és un fitxer; si és una carpeta informa només de l'espai reservat per metadades) en bytes
Diu la data i hora d'última modificació de l'element
Diu el nom de l'element
-h Mostra el tamany vist amb -l en sufixes (K,M,G...) en comptes de directament en bytes.
-d Combinat amb -l, mostra la informació relativa a la pròpia carpeta, no al seu contingut
-R Fa un llistat recursiu (mostra el contingut de les subcarpetes, i el d'aquestes, i el d'aquetes...)
-X Ordena el llistat per tipus de fitxer (en realitat, per extensió, de forma alfabètica)
-S Ordena el llistat per tamany, de gran a petit
-t Ordena el llistat per data d'última modificació, de més moderna a més antiga
-r Fa que -X, -S ó -t ordenin al revés (de “z” a “a”, de petit a gran o d'antiga a moderna)
-c Combinat amb -l, mostra la data de creació del fitxer (en comptes de la d'última modificació)
-u Combinat amb -l, mostra la data d'últim accés al fitxer, en comptes de la d'última modificació
Aquests paràmetres es poden escriure un rera l'altre (així: ls -a -l -S -r) o bé, de forma més compacta, escrivint un guió i després totes les lletres seguides (així: ls -alSr) . No importa l'ordre en què s'escriguin els paràmetres

chown usuari:grup unarxiu Estableix l'usuari indicat (i el seu grup) com a propietari de l'arxiu especificat.
Per defecte, l'usuari propietari d'un arxiu és aquell que ha creat l'arxiu i el que té d'especial és que té assignats en aquest arxiu uns permisos més específics que la resta d'usuaris.
Per defecte, tots els usuaris a les distribucions modernes es creen automàticament dins d'un grup homònim on només està ell mateix i ningú més (per exemple: l'usuari "root" pertany al grup "root" i així amb tots). Per tant, el més habitual serà escriure sempre a la comanda *chown* el mateix abans i després dels dos punts
NOTA: Els grups d'usuaris són una manera de assignar permisos a múltiples usuaris de cop: cada usuari que pertany a un grup obté els permissos assignats a aquest grup automàticament.
-R Si en comptes d'indicar un arxiu s'indica una carpeta, fa que el canvi de propietari afecti també a tot el contingut de la carpeta, recursivament

chmod XXX unarxiu Canvia els permisos d'un arxiu, on “XXX” són tres números. El de més a l'esquerra representa els permisos de l'usuari propietari de l'arxiu. El del centre representa els permisos de tots els usuaris que pertanyen al mateix grup que l'usuari propietari. El de més a la dreta representa els permisos de la resta d'usuaris. Existeixen tres tipus de permisos: 4 (lectura), 2 (escriptura) i 1 (execució), a més de les possibles combinacions: 3 (escriptura i execució), 5 (lectura i execució), 6 (lectura i escriptura), 7 (els tres permisos) o 0 (cap). Tenir el permís de lectura implica poder veure el contingut del fitxer; tenir el permís d'escriptura implica poder modificar aquest contingut; tenir el permís d'execució implica poder posar en marxa aquest arxiu (si és que aquest és un programa)

chmod XXX unacarpeta Canvia els permisos d'una carpeta (però no el del seu contingut). Funciona igual que la comanda anterior, però el significat dels permisos canvia: tenir el permís de lectura implica poder fer un llistat del contingut de la carpeta (“ls”), tenir el permís d'escriptura implica poder modificar el contingut de la carpeta (és a dir, poder afegir esborrar o renombrar arxius) i tenir el poder d'execució implica poder accedir a dins de la carpeta (“cd”)

-R Fa que el canvi de permisos afecti també al contingut de la carpeta, recursivament

La comanda *chmod*, ja sigui aplicada a arxius o a carpetes té una altra manera d'escriure's, alternativa a la notació numèrica de permisos. Consisteix en definir amb una lletra a quin dels tres subjectes es vol canviar el permís ("u" per l'usuari propietari, "g" pel seu grup i "o" pels "altres", per la resta), el símbol "+" o "-" per afegir o treure el permís, respectivament i una lletra més per indicar de quin permís estem parlant ("r" per lectura, "w" per escriptura, "x" per execució.). Per exemple, si volem afegir el permís d'escriptura d'un arxiu als "altres" podrem fer *chmod o+w unarxiu* Com es pot veure, aquesta notació és molt més fina ja que permet canviar només un determinat permís sense alterar la resta. No obstant, es poden assignar varis permisos a la vegada acumulant les lletres (i si cal, separant per comes), així, per exemple: *chmod ug-r,u+wx unarxiu* (en aquest cas, s'està treient el permís de lectura al propietari i el seu grup i s'està assignant els permisos d'escriptura i execució al propietari, tot de cop). Si no s'indica cap subjecte (per exemple, així, *chmod +x unarxiu*) s'entén que són tots tres (una forma equivalent seria escriure el subjecte "a").

cp unarxiu unacarpeta Copia l'arxiu “unarxiu” dins la carpeta “unacarpeta”

-p Manté el propietari, els permisos i les dates (de modificació, etc) originals a la còpia (en comptes d'establir com a nou propietari l'usuari que fa la còpia, aplicar els permisos per defecte pels arxius nous i resetejar les dates, que és el que passaria si no s'escriu aquest paràmetre)

-u Només fa la còpia si “unarxiu” no existeix ja a “unacarpeta”, o si és més nou que el que hi ha

-b Si "unarxiu" ja existeix a " unacarpeta", crea un backup d'aquest (amb l'extensió acabada en "~") abans de col.locar allà el nou "unarxiu". Es pot combinar amb -u.

-i Pregunta sempre a l'usuari abans de sobrescriure un fitxer

També es pot copiar un fitxer a un altre lloc i a la vegada canviar el seu nom: per fer això sols cal executar:
cp unarxiu unacarpeta/nounomdelarxiu (amb els paràmetres necessaris si s'escau).

cp -r unacarpeta unaaltra Copia tota la carpeta “unacarpeta” (ella inclosa) dins de la carpeta “unaaltra”

-R Igual que -r

-a Combinació de -p i -r

-v Mostra els diferents fitxers que va copiant en temps real

-d Si hi ha enllaços dèbils, els copia com a tals en comptes de copiar l'arxiu apuntat

-s No copia: crea enllaços dèbils

-l No copia: crea enllaços durs

Els paràmetre -p, -u, -b, -i també s'apliquen aquí. Igualment, els paràmetres -d, -s, -l es poden aplicar allà

cp -r unacarpeta/ unaaltra* Copia tot el contingut de la carpeta “unacarpeta” dins de la carpeta “unaaltra”
NOTA: per a què "*" també "agafi els fitxers ocults, abans haurem d'executar la comanda *shopt -s dotglob*

mv unarxiu unacarpeta Mou l'arxiu “unarxiu” dins la carpeta “unacarpeta”

-u Només s'executa si “unarxiu” no existeix ja a “unacarpeta”, o si és més nou que el que hi ha

-b Si "unarxiu" ja existeix a " unacarpeta", crea un backupt d'aquest (amb l'extensió acabada en "~") abans de moure allà el nou "unarxiu". Es pot combinar amb -u.

-i Pregunta sempre a l'usuari abans de sobrescriure un fitxer

mv unarxiu unaltre Canvia el nom de l'arxiu. En realitat el que fa és moure l'arxiu al mateix lloc i

aprofita per canviar el nom. Una altra manera és fent servir la comanda *rename* ...

mv unacarpeta unaaltra Mou tota la carpeta “unacarpeta” (ella inclosa) dins de la carpeta “unaaltra”
-v Mostra els diferents fitxers que va movent en temps real

mv unacarpeta/ unaaltra* Mou tot el contingut de la carpeta “unacarpeta” dins de la carpeta “unaaltra”
NOTA: per a què "*" també "agafi els fitxers ocults, abans haurem d'executar la comanda *shopt -s dotglob*

rm unarxiu Elimina l'arxiu especificat

rm -rf unacarpeta Elimina la carpeta especificada (i tot el seu contingut)

*rm -rf unacarpeta/** Elimina tot el contingut de la carpeta especificada (però no ella mateixa, que quedarà buida)
NOTA: per a què "*" també "agafi els fitxers ocults, abans haurem d'executar la comanda *shopt -s dotglob*

Si el que es desitja és esborrar tots els fitxers recursivament d'una carpeta però només els fitxers (deixant les subcarpetes buides), es pot executar la comanda *rm unacarpeta/*** El símbol "**" és similar a "*" però afecta a totes les subcarpetes SEMPRE I QUAN abans haguem executat la comanda *shopt -s globstar* (si no, "**" és equivalent a "*", és a dir, només mira a la carpeta actual però no a les subcarpetes).

touch unarxiu Crea un arxiu (buit) . Els permisos per defecte solen ser 664 (això es podria canviar amb la comanda *umask* ... però no ho veurem)

mkdir unacarpeta Crea una carpeta (buida) . Els permisos per defecte solen ser 775 (això es podria canviar amb la comanda *umask* ... però no ho veurem)

Si s'escriu *mkdir unacarpeta/unaaltra*, es crearà “unaaltra” sempre i quan “unacarpeta” ja existeixi prèviament (si no, donarà un error). Si es volen crear les dues carpetes a l'hora sense que dóni aquest error, s'ha d'afegir el paràmetre *-p* (aquest paràmetre també serveix per a què no dóni error si pretenem crear una carpeta que ja existeix).

ln -s /ruta/arxiu nomlink Crea un link de tipus "suau" -també anomenat "dèbil" o “accés directe”- de l'arxiu
ln -s /ruta/carpeta nomlink especificat allà on s'executi la comanda. Notar que si l'arxiu original desapareix, el link "es trenca" perquè roman apuntant a un lloc inexistent. Això es pot veure fàcilment observant la sortida en colors de *ls -l*

ln /ruta/arxiu nomlink Crea un link de tipus "dur" de l'arxiu especificat allà on s'executi la comanda. Un enllaç dur té la mateixa "jerarquia" que l'arxiu original, de manera que si aquest desapareix, el contingut continua intacte gràcies a l'existència de l'enllaç dur. No cal confondre, però, un enllaç dur amb una còpia: el fitxer físicament només es troba en un lloc, només que està accessible des de diferents llocs amb diferents noms. El nº d'enllaços durs que té un determinat fitxer es pot esbrinar gràcies a la 2ª columna per l'esquerra que apareix en executar *ls -l* Limitacions dels enllaços durs són que només poden existir dins d'una mateix partició i que no poden apuntar a carpetes

És també interessant la comanda *readlink -f nomlink* , la qual mostra la ruta absoluta del fitxer real al que apunta el link indicat

tree unacarpeta Mostra recursivament tot el contingut de la carpeta indicada en forma d'arbre
-a Mostra també els fitxers ocults
-f Mostra per cada fitxer i carpeta llistat la seva ruta absoluta
-P "ptr" Mostra només els fitxers que coincideixin amb el patró indicat (p.ex: "*.png")
-I "ptr" No mostra els fitxers que coincideixin amb el patró indicat (p.ex: "*.png")
-d Mostra només les subcarpetes (no els fitxers)
-l Si es detecta un link suau a un directori, mostra el contingut d'aquest directori

-x No entra a punts de muntatge a altres particions/dispositius que hi puguin haver
-L nº Indica el número de subdirectoris que es mostraran (per defecte, fins el final)
-C Mostra el resultat amb colors
-pugshD Mostra per cada fitxer una sortida similar a *ls -l*
--du Mostra el tamany total acumulat per cada subdirectori
-o fitxer Guarda la sortida en el fitxer indicat. Útil si es combina amb -H carpetaBase, el qual fa que aquesta sortida sigui una pàgina HTML amb enllaços <a> relatius a la carpetaBase indicada

¿Què hauríem de fer si volguéssim esborrar més d'un arxiu a la vegada amb la comanda rm? ¿O copiar més d'un arxiu amb la comanda cp? Quan necessitem que la comanda a utilitzar tingui en compte més d'un arxiu, podem indicar-ho mitjançant "**comodins**":

***** : Equival a qualsevol cosa (una lletra o número, dos, tres...fins i tot res) dins del nom de l'arxiu . Per exemple, "c*sa" pot ser "casa", "cosa", "c3sa", "causa", "crosa", "cxxxsa", "c12gsa", "csa", etc . De fet, ja s'ha utilitzat en veure les comandes *cp* i *mv*

?: Equival a una sola lletra (o número) dins del nom de l'arxiu. Per exemple, "c?sa" pot ser "casa", "cosa", "c1sa" etc, però no "causa" o "csa". Es poden escriure diferents ? seguits, indicant la obligatorietat d'un número de caracters (per exemple, "c???" indica qualsevol paraula que comenci amb "c" i que tingui quatre lletres.

[⋯]: Similar a l'anterior, però obliga a què els caracters possibles només puguin ser els especificats dins dels claudàtors, i no pas qualssevol com passava amb el ?. Per exemple, "c[iu]sa" només pot ser o bé "cisa" o bé "cusa".

[!⋯]: Similar a l'anterior, però obliga a què els caracters possibles NO siguin els especificats dins dels claudàtors. Per exemple,"c[!iu]sa" pot ser "cosa", "casa", "cbsa", etc, però mai "cisa" ó "cusa" (com tampoc paraules que tinguin menys ni més que quatre lletres.

A més d'aquests comodins clàssics, al shell bash també podem fer servir els següents símbols:

{xx,yy,zz}: Millor explicar-ho amb un exemple: la comanda *mkdir carpeta{A,B,C}* crearà tres carpetes anomenades "carpetaA", "carpetaB" i "carpetaC" perquè el que passa és que la cadena "carpeta{A,B,C}" es transforma en la cadena "carpetaA carpetaB carpetaC" i sobre aquesta actua la comanda mkdir. Igualment, la comanda *rm -rf carpeta{A,B,C}* les esborraria totes de cop. Un altre exmple pot ser *cp fitxer{,.bak}* que equival a *cp fitxer fitxer.bak* .
També es pot fer servir la seqüència ".." en comptes de comes per indicar, en comptes de valors concrets, un rang, ja sigui numèric o alfabètic; per exemple , *mkdir carpeta{A..C}* farà el mateix que els exemples anteriors. Si s'escriuen varis conjunts de claus, es fan totes les combinacions possibles; per exemple: *echo {1..3}{7..9}* imprimirà 17, 18, 19, 27, 28, 29, 37, 38 i 39. També es pot afegir un tercer valor (després d'altres ..) per indicar el pas de l'increment; per exemple, *echo {1..10..2}* mostraria 1 3 5 7 i 9 i *echo {a..e..2}* mostraria a c e. Finalment dir que els rangs també poden ser decreixents.

Si es té l'opció *extglob* de Bash activada (això es pot veure amb la comanda *shopt extglob* ; si no ho està, es pot activar executant *shopt -s extglob* i, si es volgués desactivar, llavors es faria així: *shopt -u extglob)* es pot fer servir els símbols

!(expressió_amb_comodins) : Nega l'expressió indicada dins els parèntesis. Per exemple, ls !(*.txt) mostraria tots els fitxers excepte els acabats en "txt".

En general, si es vol fer que un comodí perdi el seu significat especial i passi a ser un caràcter "tal qual", s'ha d'"escapar". Per escapar un caràcter simplement cal precedir-lo del símbol "\". Així, doncs, per fer que el símbol "*" representi un simple asterisc caldria escriure "*".

file unarxiu Informa sobre quin tipus de fitxer és l'indicat (això ho aconsegueix estudiant el seu contingut intern, ja que les extensions poden enganyar).
-b Modifica la sortida per a què només tregui el tipus, sense acompanyar-lo del nom del fitxer
-i Mostra el tipus MIME oficial en comptes del tipus més "d'estar per casa" mostrat per defecte

***Àlies, arxius .bashrc (i similars) i missatges de benvinguda:**

Als sistemes Ubuntu (i també Fedora) existeix una "comanda" anomenada *ll* que és idèntica a *ls -l* . No obstant, *ll* no és una comanda sinò un àlies ja predefinit: és a dir, un "sinònim" d'una comanda real (i, generalment, més curt d'escriure i fàcil de recordar). De fet, *ls* és també un àlies de sí mateix amb un paràmetre per veure els colors (concretament, *--color=auto*). A continuació es llisten les comandes que ens permetran gestionar els àlies (tant els que ja venen predefinits com els nostres propies que creem):

alias	Mostra els àlies reconeguts pel shell actual
alias unnom="comanda"	Crea un nou nom inventat per nosaltres per la comanda especificada, la qual pot venir acompanyada dels paràmetres que es vulguin afegir. Útil quan s'utilitza moltes vegades una comanda molt llarga
unalias unnom	Destrueix l'àlies especificat (ja sigui predefinit o propi). De totes formes, en tancar el shell, els àlies propis creats en aquesta sessió es destruiran igualment

¿Com es pot fer per a què els àlies propis siguin permanents a l'igual que ho són els predefinits del sistema ? El truc està en escriure la comanda de creació d'àlies al final d'alguns dels següents fitxers de texte:

/etc/profile : el seu contingut s'executa cada cop que qualsevol usuari inicia sessió en un terminal virtual
~/.bash_profile : el seu contingut s'executa cada cop que aquest usuari concret inicia sessió en un " "
/etc/bash.bashrc : el seu contingut s'executa cada cop que qualsevol usuari obre una finestra bash a l'escriptori
~/.bashrc : el seu contingut s'executa cada cop que aquest usuari concret obre una finestra bash a "

De fet, al final dels fitxers anteriors també es pot escriure qualsevol comanda. D'aquesta manera, podem executar automàticament allò que volguem cada cop que s'inicïi una shell bash. Per exemple, podem fer servir les següents comandes per mostrar un bonic missatge de benvinguda:

figlet -f nomFont "Missatge"	Mostra el missatge amb la font indicada. Les fonts Figlet es troben a /usr/share/figlet (són els arxius *.tlf) però també es pot usar el nom d'una font que es trobi a la carpeta actual (o en qualsevol altre lloc si s'indica la ruta absoluta)
-l, -r, -c	Mostra el missatge alineat a l'esquerra, dreta o centrat, respectivament
-w nº	Defineix l'amplada de l'espai on es mostrarà el missatge. Si és massa petit, es partirà
-k	S'activa el "kerning" (això afecta a l'espaiat entre lletres)

toilet -f nomFont "Missatge"	Igual que Figlet (de fet, fa servir les mateixes fonts). Admet els mateixos paràmetres que Figlet però també un parell més que són interessants
-F list	Mostra la llista de filtres ("efectes") que es poden aplicar
-F nomfiltre:unaltre:...	Aplica el/s filtre/s indicat/s (metal, border, flip, flop, gay...)
-E list	Mostra la llista de formats de sortida en què es pot exportar el missatge
-E nomformat	Exporta a un fitxer del format indicat el missatge tal com quedaria gràficament. Formats interessants són HTML, SVG ó PostScript.
cowsay -f dibuix "Missatge"	Mostra el missatge com si el digués el dibuix indicat (per defecte, una vaca) Els dibuixos disponibles es troben a /usr/share/cowsay (es poden veure també amb *cowsay -l)*
boxes -d dibuix unarxiu	Mostra el missatge -que ha d'estar guardat prèviament dins d'"unarxiu"- dins d'un requadre amb la forma indicada. Els requadres disponibles es poden veure executant *boxes -l* però, de fet, es troben tots dins de l'arxiu /usr/share/boxes/boxes
-a h{l|r|c}v{t|c|b}	Indica l'alineament horitz i vert del text dins del requadre
-p hnºvnº	Indica el "padding" horitz i vert del text dins del requadre. Hi ha + opcions
-s nºxnº	Indica el tamany del requadre
tput ...	S'estudiarà als exercicis

Variables d'entorn i de shell:

Cada cop que s'obre un shell (ja sigui en un terminal virtual, dins del gnome-terminal o similars o bé executant un script -ja que els scripts obren un shell intern "invisible" dins del qual s'executen, retornant al shell "pare" en acabar-), es produeix una recopilació d'informació de diferents llocs (fitxers, memòria RAM, etc) que estableix com serà el comportament d'aquest nou shell. A aquesta informació, que podem modificar-la un cop dins d'aquest shell segons les nostres necessitats, se l'anomena genèricament "variables de sistema" justament per això: perquè pot ser "variada". Que s'utilitzi el plural denota que les variables del sistema en realitat són un conjunt de diferents parelles de nom<->valor, on cada parella és una determinada variable concreta que serveix per definir una característica determinada del shell (així doncs, depenent del cas ens interessarà conèixer i/o modificar el valor d'una variable o d'una altra). El nom de cada variable normalment és escrit en majúscules, i per accedir al seu valor haurem de precedir aquest nom pel símbol $.

En realitat podem distingir dos tipus de variables del sistema: les anomenades variables "d'entorn" i les anomenades variables "del shell". Les primeres estan disponibles pel shell actual i a més per tots els processos que s'executin a partir d'ell (com ara programes gràfics, scripts o fins i tot, shells "fills"); aquest tipus de variables s'usen per passar informació del shell actual al procés fill engegat allà mateix (a la inversa no és possible: no es pot passar el valor d'una variable d'entorn del procés fill al shell "pare"...si fos possible un programa podria alterar el comportament del sistema molt fàcilment). Les segones només estan disponibles pel shell actual; habitualment s'usen per emmagatzemar informació temporal.

Cal tenir clar que els valors de les variables d'entorn són locals; és a dir, específiques per cada shell que obrim. Això vol dir que si obrim dos finestres de terminal (és a dir, dos processos *bash* diferents) i canviem el valor d'una variable d'entorn en una de les finestres, aquest valor no es canviarà al shell de l'altra finestra.

Les següents comandes serveixen per gestionar tant les variables d'entorn com les de shell:

printenv	Mostra la llista actual de variables d'entorn (nom i valor) No existeix cap comanda que només mostri la llista actual de variables de shell: la comanda *set* les mostra però mesclades amb les d'entorn i més coses (funcions)
echo $NOMVARIABLE	Mostra el valor de la variable especificada (d'entorn o de shell) També es podria fer servir *printenv NOMVARIABLE* (només per les d'entorn, clar)
NOMVARIABLE=*valor*	Si no existeix, crea una variable de shell amb un valor inicial. Si existeix, canvia el seu valor

NOTA: Si el valor d'una variable conté espais en blancs, s'haurà d'escriure entre cometes
NOTA: Es pot assignar més d'un valor a una variable; per fer això cal escriure cada valor separat de l'anterior pel símbol dels dos punts, així: *NOMVARIABLE=valor1:valor2:valor3...*
NOTA: Es pot deixar "en blanc" el valor d'una variable simplement escrivint *NOMVARIABLE=*

export NOMVARIABLE=valor	Si no existeix, crea una variable d'entorn amb un valor inicial. Si existeix, canvia el seu valor i, si era de shell, la transforma en d'entorn.
export -n NOMVARIABLE	"Degrada" la variable d'entorn indicada a variable de shell (mantenint el seu valor)
unset NOMVARIABLE	Destrueix la variable especificada (d'entorn o de shell)

A continuació es mostren algunes de les variables d'entorn més importants (encara que hi ha moltes més; per exemple es pot consultar a https://help.ubuntu.com/community/EnvironmentVariables#List_of_common_environment_variables una llista més extensa):

USER : Guarda el nom de l'usuari actual
HOME : Guarda la ruta de la carpeta personal de l'usuari actual
PWD : Guarda la ruta de la carpeta de treball (és a dir, la de la carpeta actual)
LANG : Guarda l'idioma actual del sistema. Altres variables relacionades (més específiques) són les **LC_***
DISPLAY: Guarda la pantalla on es veuen els programes gràfics (per defecte, és la pantalla 0)
SHELL : Guarda la ruta de l'executable corresponent al shell actual (normalment, /bin/bash)
LS_COLORS: Indica quins colors es veuen en fer un *ls*, segons el tipus de fitxer. Cada tipus de fitxer està separat dels altres per ":", i els codis de color són els mateixos que els utilitzats a la variable PS1 (veure més avall)
PATH : Guarda la llista de carpetes (separades per ":") on el sistema anirà a buscar la comanda escrita per l'usuari en un moment donat. Si aquesta comanda es troba, s'executarà; si no, s'indicarà amb un missatge del tipus "command not found"

A continuació es mostren algunes de les variables de shell més importants:

PS1 : Guarda l'aparença visual del prompt actual. Pot tenir com a valor, a més d'una frase qualsevol, una sèrie de signes concrets, com per exemple:

\d : data actual \t : hora, minuts i segons actuals \h : nom de la màquina
\u : usuari actual \w : carpeta actual \$: indica si s'és root (#)o no ($)
\e[0;30m untextequalsevol : color de lletra (negre)
\e[0;31m " : color de lletra (vermell) ...hi ha fins el 37
\e[1;30m " : color de lletra (negre més clar)
\e[1;31m " : color de lletra (vermell més clar)...hi ha fins el 37
\e[0;40m " : color de fons (negre)
\e[1;40m " : color de fons (negre més clar)...hi ha fins el 47
\e[4,30m " : subratllat

Per veure més combinacions, veure https://en.wikipedia.org/wiki/ANSI_escape_code a partir de la taula "SGR (Select Graphic Rendition) parameters" cap endavant

PS2 : Guarda l'aparença visual dels prompts quan una comanda es parteix en diferents línies
RANDOM : El seu valor és aleatori cada cop que es vol consultar
TMOUT : Guarda el nº de segons que han de passar sense que hagi activitat per a què es tanqui automàticament la sessió iniciada en un terminal virtual o bé a través de SSH
IFS : Indica els caràcters que es faran servir com a separadors en les dades que entrin al shell actual.
Aquesta variable se sol definir dins d'scripts per tal d'indicar els separadors en les dades que entrin en ells (a través de paràmetres, via comanda *read*, etc)
Els separadors per defecte són el salt de línia (\n), el tabulador (\t) i l'espai en blanc.

Les variables relacioades amb l'historial de comandes:
HISTFILE : Nom del fitxer on es guardarà l'historial -fins llavors a la RAM- quan es tanqui el terminal (per defecte, aquest fitxer és ~/.bash_history). Si no es vol utilitzar, es pot fer *unset HISTFILE* o *HISTFILE=/dev/null*
HISTFILESIZE : Número de línies que com a màxim es poden guardar al fitxer anterior
HISTSIZE : Número de línies que com a màxim poden estar disponibles a la memòria RAM
HISTCONTROL : Pot valer, entre altres "ignoredups" o "ignorespace". El primer valor no guarda a l'historial -en RAM- les comandes duplicades (que siguin consecutives, això sí). El segon no guarda les que estiguin precedides d'un espai en blanc (un bon truc per ocultar-les, doncs). Per gaudir dels 2 valors es pot usar el valor "ignoreboth". També està "erasedups", que només guarda comandes úniques (d'aquesta manera les repetides no cal que siguin consecutives per ser esborrades com passava amb "ignoredups").
HISTIGNORE : Llista les comandes (separades per :) que no es volen guardar a l'historial -en RAM-. Es poden fer servir comodins. Un exemple seria *HISTIGNORE="ls*:cp*"* . El símbol "&" aquí significa "darrera comanda executada": posar-lo seria equivalent al valor ignoredups de HISTCONTROL
HISTTIMEFORMAT :Defineix el format (idèntica al de la comanda *date*) de la data i hora que es guardarà a l'historial juntament amb la comanda corresponent. Si no està definida, només es guardarà la comanda sense data/hora

Relacionada amb l'historial està la següent comanda:

history Mostra l'historial de les comandes executades anteriorment (emmagatzemat a la RAM).
Atenció: l'historial en RAM és independent per cada terminal que obrim en paral.lel.
Si s'escriu *history nº* només es mostren les darreres nº entrades de l'historial a RAM
El paràmetre *-c* serveix per esborrar tot l'historial de la RAM
El paràmetre *-d nº* per esborrar-ne l'entrada indicada amb un número concret.
El paràmetre *-a* serveix per afegir ja a l'arxiu .bash_history les comandes actualment llistades a l'historial de la RAM sense que calgui tancar el terminal.
El paràmetre *-w* fa el mateix que *-a* però sobreescrivint en comptes d'afegir.

Per tornar a executar una comanda concreta de l'historial, s'ha d'escriure *!nºcomanda*.
Pulsant CTRL+R i escrivint el començament d'una comanda es pot anar a la darrera comanda que coincideix. Pulsant més vegades CTRL+R es va més enrera. Quan s'arribi a la comanda desitjada es pot pulsar ESC per si opcionalment es vol editar i finalment ENTER per executar-la
!! executarà la darrera comanda escrita (útil per si hem oblidat escriure sudo, p. ex: *sudo !!*)
Un equivalent a !! és !*
!paraula executarà la darrera comanda que comenci per "paraula"
!paraula:p mostrarà la darrera comanda que comenci per "paraula" i l'afegirà a l'historial
!$* conté els paràmetres passats a la darrera comanda (*ls !$, per exemple)
*!^* conté el primer paràmetre passat a la " "
!?paraula? executarà la darrera comanda que contingui "paraula"

^*paraula1*^*paraula2*^ executarà la darrera comanda que contingui "paraula1" però la substituirà per "paraula2". Molt útil quan només volem canviar un detall de tota la comanda Per més, veure l'apartat "Word designators and modifiers" de http://www.catonmat.net/blog/the-definitive-guide-to-bash-command-line-history/

Un programa molt interessant per gestionar l'historial, d'altra banda, és https://github.com/dvorka/hstr

Desgraciadament, si modifiquem el valor d'una variable d'entorn o de shell, veureu que quan tanquem el shell actual i tornem a obrir un de nou, aquest nou valor desapareixerà i la variable tornarà a tenir el valor que tenia per defecte. El mateix passa si creem nosaltres variables pròpies: si tanquem el shell on s'han definit, desapareixeran ¿Com es pot fer doncs per a què els valors modificats de les variables (o fins i tot, noves variables pròpies que creem nosaltres mateixos) siguin permanents? El truc està en escriure la definició de la variable (ja sigui de tipus shell o bé, mitjançant la comanda *export*, de tipus d'entorn) al final d'alguns dels següents fitxers de text:

/etc/profile : el seu contingut s'executa cada cop que qualsevol usuari inicia sessió (gràfica o no) al sistema
~/.profile : el seu contingut s'executa cada cop que aquest usuari concret inicia sessió (" " ") al sistema

***Treballar amb contingut textual:**

cat unarxiu Mostra el contingut de l'arxiu especificat. Es poden especificar més d'un arxiu: en aquest cas, es mostraria el contingut de cadascun un darrera l'altre.

-s No mostra les línies en blanc repetides que pugui haver dins l'arxiu

-n Mostra a més, el número de línia

nano unarxiu Edita el contingut de l'arxiu especificat. De fet, és un editor de text tan complet com el gedit, però en mode texte. Les opcions més importants del programa són
CTRL+O (guardar), CTRL+X (sortir)
CTRL+W (buscar) i ALT+W (seguir buscant), ALT+\ (reemplaçar)
ALT+A (sel.leccionar text), CTRL+K (tallar), ALT+6(copiar), CTRL+U (pegar)
CTRL+C (informa de la posició actual del cursor)

less unarxiu Mostra el contigut de l'arxiu especificat de forma paginada. A més de fer servir les tecles AvPag, RePag i els cursors, es poden fer servir les mateixes tecles de navegació de les pàgines del manual ("/", "n","N", "q", entre altres).

-*R* Interpreta els codis ANSI correctament per poder mostrar colors

head -n nº unarxiu Mostra només les primeres nº línies de l'arxiu especificat

head -n -nº unarxiu Mostra les línies de l'arxiu especificat excepte les darreres nº línies

tail -n nº unarxiu Mostra només les últimes nº línies de l'arxiu especificat

tail -n +nº unarxiu Mostra el contingut de l'arxiu especificat a partir de la línia nº

-f Mostra les últimes línies de l'arxiu especificat en temps real. Molt útil per observar logs "en calent"

hexdump unarxiu Mostra el contingut de l'arxiu indicat directament en hexadecimal (juntament amb numeració de files a l'esquerra de tot; si aquesta no es vol veure, es pot fer llavors *hexdump unarxiu | cut -c9-*)

-C Mostra, a la dreta de tots (després de la numeració de files i dels valors en hexadecimal), la possible correspondència de cadascun dels bytes amb un caracter ASCII

-c Mostra la numeració de les files i la correspondència amb ASCII però no els valors hexadecimals
Es pot canviar la manera de mostrar la sortida especificant el format desitjat amb el paràmetre -e
Per veure la llista de formats possibles, consulteu la pàgina del manual.

-n nº Monstra només els primers nº bytes del fitxer

-s nº Se "salta" el nº de bytes indicats del principi del fitxer i comença a mostrar a partir d'all

wc unarxiu Conta el número de caracters, paraules i línies que conté l'arxiu especificat

-c Conta només els caracters

-w Conta només les paraules

-l Conta només les línies

-L Conta els caràcters de la línia més llarga

cut -c 1-4,6 unarxiu De cada línia de l'arxiu només mostra (en aquest exemple) els caràcters 1,2,3,4 i 6

cut -f 1-4,6 -d ":" unarxiu De cada línia de l'arxiu només mostra (en aquest exemple) les columnes 1,2,3,4 i 6, suposant que el separador de columnes és el caracter ":"

sort unarxiu Ordena les línies de l'arxiu alfabèticament per la primera lletra de cada línia
Es pot indicar més d'un arxiu: llavors ordenarà tot el conjunt com si només fos un

-r Ordena al revés

-n Si hi ha números, serveix per ordenar correctament (és a dir, fer 100>12, i no al revés)

-h Combinat amb -n, serveix per ordenar correctament números del tipus 3K, 5G, 2M, etc

-f Ordena independentment de si els caràcters són en majúscules o minúscules
-k 3 -t ";" Ordena segons la columna nº 3 (en l'exemple), suposant que el caràcter separador és ";"
-c Comprova si l'arxiu indicat ja està ordenat o no però no fa res més
-R Ordena aleatòriament
-u Si després d'ordenar apareixen línies consecutives repetides, només mostra una

uniq unarxiu Mostra només les línies de l'arxiu indicat que siguin úniques (és a dir, no repetides)
Atenció: per a què funcioni correctament, les línies idèntiques han de ser consecutives (per tant, moltes vegades s'hauràn d'ordenar prèviament amb la comanda *sort)*
-d Mostra (una vegada) només les línies que estan repetides
-i No té en compte si són majúscules o minúscules
-c Mostra per cada línia el número total de línies repetides que s'han trobat

paste unarxiu unaltre Concatena la 1ª línia del primer fitxer amb la 1ª línia del segon (separant-les per un tabulador) en una sola línia, la 2ª línia del primer amb la 2ª del segon en una línia següent, i així

comm unarxiu unaltre Compara línia a línia el primer fitxer amb el segon. Aquestes línies han hagut de ser prèviament ordenades i sense duplicats. Mostra tres columnes: les línies que només estan al primer arxiu a la primera, les línies que només estan al segon arxiu a la segona i les que estan a totes dues a la tercera. El paràmetre -1 oculta la primera columna, el paràmetre -2 oculta la segona i -3 oculta la tercera

diff unarxiu unaltre Compara línia a línia el primer fitxer amb el segon, però d'una forma més avançada que comm. Aquestes línies han hagut de ser prèviament ordenades i sense duplicats.Si els fitxers són iguals, no mostra res. Si no són iguals, aquesta comanda informa de les accions que s'haurien d'aplicar per fer que el primer arxiu sigui igual al segon. Per exemple, "10c10" significaria "reemplaçar la línia nº10 del primer fitxer (indicada pel símbol '<') per la línia nº10 del segon fitxer (indicada pel símbol '>')". "24,26d23" significaria "esborrar des de la línia 24 a la 26 del primer fitxer". "5a6" significaria "afegir la línia indicada rera "<" després de la línia 5.
-s Si els fitxers són iguals, en comptes de no mostrar res mostra un missatge informatiu
-y Mostra tot el contingut dels dos fitxers analitzats un al costat de l'altre
-q No mostra res mai. Per saber si ha trobat alguna diferència caldrà veure el valor de la variable especial $? (que valdrà 0 si els fitxers són iguals i 1 si no). Això és útil per scripts.

Els "redireccionadors" són un element molt important del intèrpret de comandes. A continuació llistem els més comuns:

comanda > fitxer (Redireccionador de sortida): Normalment serveix per guardar en un fitxer el resultat visible a la pantalla de l'execució d'una comanda. Per exemple, *ls > unarxiu* gravarà en "unarxiu" la informació que mostra la comanda *ls*. És equivalent a **1>**

comanda >> fitxer (Redireccionador de sortida doble): Serveix pel mateix que el redireccionador anterior (gravar en un fitxer la sortida de pantalla d'una comanda), però amb la diferència que no sobreescriu el que ja hi havia d'abans al fitxer, sinò que el nou contingut s'afegeix al final. Per exemple, *ls >> unarxiu* afegeix a "unarxiu"la informació que mostra la comanda *ls* cada cop que s'executa . És equivalent a **1>>**

comanda | comanda (Redireccionador d'entrada i sortida -"tuberia"-): Redirecciona la sortida d'una comanda (que es mostraria normalment a la pantalla) per portar-la com a entrada de dades d'una segona comanda posterior. Per exemple, *ls | less* traspassa la informació que mostraria la comanda ls a la comanda less, per tal de què aquesta mostri la informació paginada. Un altre exemple: *ls | wc -l* traspassa la informació que mostraria la comanda *ls* a la comanda *wc*, per tal de què aquesta conti el número de línies que hi conté (per tant, de fet, estaria contant el número de fitxers i carpetes existents en el llistat). Un altre exemple (veure més endavant): *ps | grep "firefox"* mostra només les línies que genera la comanda *ps* que contenen la paraula "firefox" (és a dir, de totes les línies que ens mostra la comanda *ps*, només veurem les filtrades per *grep* segons la paraula especificada). Altres exemples més complexes poden ser: *ls | grep -v "^d" | wc -l* , o *date | cut -f 4 -d " "* o *grep "usuari" /etc/shadow | cut -f 2 -d ":"*, etc

NOTA: La comanda *pv* (que potser s'ha d'instal.lar a part) es pot utilitzar entre dues canonades per mostra per pantalla el progrés de les dades transferides entre la primera i la segona (velocitat de tranferència, temps trigat i el que falta, etc). Per exemple: *cat fitxer.gros | pv | gzip > fitxer.gros.gz* Un paràmetre interessant és *-s nºbytestotalstransferits*, el qual serveix per a què pv pugui dibuixar la barra de progrés creixent (ja que sap on ha d'acabar la transferència)...aquest nº es pot saber prèviament si és el tamany d'un fitxer, per exemple. Un altre paràmetre interessant és *-L nº* que defineix el màxim de tranferència (en bytes/s) entre les canonades (útil per limitar la velocitat de les dades entre la comanda anterior i la darrera. Més a http://www.catonmat.net/blog/unix-utilities-pipe-viewer/

comanda 2> fitxer (Redireccionador d'error): Els missatges d'error (com per exemple el que apareix en oblidar-se algun paràmetre obligatori) també solen aparèixer per pantalla. Si es volen redireccionar a un altre lloc, es fa servir els símbols "2>", que serveixen específicament pels missatges d'error. És molt comú fer la redirecció en comptes de a un fitxer, al dispositiu /dev/null, per tal de no veure cap missatge d'error que molesti. També està **2>>**

comanda < fitxer (Redireccionador d'entrada): No s'utilitza gaire. Serveix per fer que una comanda rebi instruccions a partir del contingut d'un fitxer en comptes del teclat. D'altra banda, si el contingut d'entrada prové d'una canonada en comptes d'un fitxer (o del teclat), moltes comandes (no totes, cal mirar el manual) necessiten tenir un guió "-" com a paràmetre per gestionar aquest tipus d'entrada correctament.

En general, si es vol fer que un redireccionador perdi el seu significat especial i passi a ser un caràcter "tal qual", s'ha d'"escapar". Per escapar un caràcter simplement cal precedir-lo del símbol "\".

A continuació llistem altres redireccionadors més especialitzats:

comanda <<< "cadena": Serveix per redireccionar d'entrada directament una cadena sense necesitat de cap comanda: ordre <<< "cadena" D'aquesta manera no cal fer echo "cadena" | comanda. Per exemple: *bc <<< "3+6"*

comanda <<MARCA cadena MARCA : Similar a l'anterior: serveix per redireccionar d'entrada directament una cadena (la indicada entre les marques) sense necesitat de cap comanda. La diferència més important és que ara aquesta cadena pot ser arbitràriament llarga i incloure salts de línia. Un exemple (la comanda bc és explicada a l'apartat "Altres" de més endavant):
bc <<HOLA
scale=4;
3/6
HOLA

comanda &> fitxer : Redirecciona a un fitxer les dues sortides: l´estàndar i l´error. També està **&>>** . També es poden entubar les dues sortides a una única entrada de la següent comanda: comanda **|&** següentcomanda

comanda | tee fitxer : La sortida estàndar de la comanda es veurà a la pantalla però a més es guardarà al fitxer indicat

(comanda1;comanda2) > fitxer : Redirecciona la sortida de totes les comandes de dins del parèntesis al mateix fitxer
Una altra manera de fer el mateix (més eficient) és **{ comanda1; comanda2; } > fitxer**

comanda1 > >(comanda2) 2> >(comanda3) : Redirecciona la sortida estàndar de comanda1 a l'entrada de comanda2 (és a dir, el mateix que comanda1 | comanda2) però a més, redirecciona la sortida d'error de comanda1 a l'entrada de comanda3. En realitat, el que passa aquí és que comanda2 i comanda3 -gràcies a escriure's dins de ">(...)" - s'executen "a l'espera" degut a tenir la seva entrada vinculada a sengles canonades associades a la sortida estàndar de comanda1 (via ">") i la sortida d'error (via "2>") respectivament. Quan aquestes canonades són omplertes en executar-se comanda1, és quan les altres dues comandes reben les dades necessàries per poder "arrencar".

comanda1 <(comanda2) <(comanda3): La sortida estàndar de comanda2 i comanda3 es guarda respectivament en un fitxer temporal (en realitat, una canonada amb nom, veure més avall), la ruta dels quals (que és similar a /proc/nº/fd) serà el valor que tindrà el primer i segon paràmetre, respectivament, de comanda1. Un exemple: *comm <(sort unarxiu) <(sort altrearxiu)*

Canonades amb nom: Amb un parell d'exemples es veurà clar
En un terminal: mkfifo untub ; echo pepe > untub En un altre terminal: *cat untub*
En un terminal: *mkfifo untub ; ls > untub* En un altre terminal: *head -n 5 < untub*
Un exemple interessant d'això és per exemple dir a la comanda wodim -un gravador de CD per terminal- que gravi al fifo untub (es quedarà esperant a rebre contingut) i després fer: *wget arxiu.iso -O untub* El que estaríem fent és gravar l'arxiu al mateix temps que el descarreguem.

comanda nº>&nº : Redirecciona el que anava inicialment a un destí indicat pel primer número (on 1 és la sortida

estàndar, també anomenada "stdout" i 2 la d'error, també anomenada "stderr"; el nº0 està reservat per l'entrada estàndar, "stdin") a un altre destí indicat pel segon número. Per exemple, per redireccionar "stderr" a "stdout" es pot fer *echo "hola" 2>&1* De fet, una alternativa a *comanda &> fitxer* és *comanda > fitxer 2>&1* i una alternativa a *comanda1 |& comanda2 és comanda1 2>&1 | comanda2*

NOTA: La gràcia del redireccionador nº>&nº està sobre tot en la posibilitat de fer servir més destins que no només stdout i stderr. Això sí, aquests destins s'han de crear primer; això es fa amb la comanda **exec,** així: *exec 3>unfitxer* (per crear el destí nº3 i associar-lo a un determinat fitxer). També es poden assignar noms en comptes de números, així : *exec {nom}>unfitxer* (en aquest cas, Bash assignarà a aquest nou destí un número intern). A partir d'aquí, si volguéssim redireccionar la sortida estàndar d'una comanda qualsevol a aquest destí, simplement caldria executar *comanda >&3* (o *comanda >&{nom}*). Per destruir el destí cal fer *exec 3>&-* (el símbol "-" aquí significa "tancar"). Amb aquest tècnica (concretament, executant *exec 2> unfitxer*) podríem fer que totes les comandes executades a partir de llavors redireccionessin automàticament els seus missatges d'error a un fitxer sense haver d'estar indicant la cueta "2> unfitxer" a cadascuna d'elles.
Si volguéssim redireccionar l'entrada, primer executaríem *exec 4<unfitxer* i llavors faríem *comanda <&4*.

***Treballar amb expressions regulars:**

grep "paraula" unarxiu Mostra les línies de l'arxiu especificat on es troba la paraula buscada

-i	Trobarà la paraula buscada ja sigui escrita amb majúscules o minúscules. Per defecte grep només troba les paraules que siguin escrites tal qual s'especifiquen
-c	Conta el número de línies que contenen la paraula especificada
-n	Mostra a més de les línies on es troba la paraula buscada, el número de cada línia
-v	Inverteix la recerca: mostra les línies que NO contenen la paraula buscada
-o	En comptes de mostrar la línia sencera on es troba la paraula, només mostra la paraula
-r	Si en comptes d'indicar un arxiu es pot indicar una carpeta, aquest paràmetre realitza una recerca de la paraula a tot el seu contingut de forma recursiva. Es pot combinar amb -n, -v...
-l	En comptes de mostrar la línia on es troba la paraula, només mostra els noms del fitxers on està. Sols té sentit si es combina amb -r o fent ús de comodins
-A nº	Mostra el nº de línies següents a cada línia on s'hagi trobat la paraula buscada
-B nº	Mostra el nº de línies anteriors a cada línia on s'hagi trobat la paraula buscada
-q	No mostra res. Per saber si ha trobat alguna cosa caldrà veure el valor de la variable especial $? (que valdrà 0 si sí ha trobat alguna cosa i 1 si no s'ha trobat res). Això és útil per scripts.
-a	Permet buscar el patró textual indicat dins d'un fitxer binari (és a dir, que no sigui de text).

La comanda *grep* no només és capaç de buscar paraules exactes, sinò també patrons (també anomenats “expressions regulars”) com ara “paraules que comencen per tal lletra”, ó “paraules que tenen tantes lletres” ó "paraules que estiguin al començament de la línia", etc, etc. Per a què *grep* pugui buscar expressions regulars en comptes de no només paraules, això sí, cal afegir el paràmetre -E.

Les expressions regulars és un tema que sobrepassa la comanda *grep* (i alguna altra que també veurem, com *sed)* ja què són estàndar independent de Linux que està presents en molts altres àmbits com ara la programació, les bases de dades, la web, etc. Això vol dir que els símbols utilitzats per construir expressions regulars, encara que coincideixin amb alguns dels que ja hem vist (asterisc, interrogant, etc) no tenen per res el mateix significat. A continuació es presenten alguns d'aquests símbols però a no ser que s'indiqui el contrari, el significat explicat aquí només tindrà validesa en el context d'ús de les comandes indicades en aquest apartat: *grep, sed*... i ja està).

. : Equival a un sol caràcter, qualsevol. Per exemple, “c.sa” pot ser “casa”, “cosa”, “c1sa” etc, però no “causa” o “csa”. Es poden escriure diferents "." seguits, indicant la obligatorietat d'un número de caracters (per exemple, “c...” indica qualsevol paraula que comenci amb “c” i que tingui quatre lletres.

[⋯] :Similar a l'anterior, però obliga a què els caracters possibles només puguin ser els especificats dins dels claudàtors, i no pas qualssevol com passava amb el ".". Per exemple, “c[iu]sa” només pot ser o bé “cisa” o bé “cusa”. També es pot posar un rang de caràcters: per exemple "[a-z]" indica totes les lletres minúscules, "[A-Z]" indica totes les majúscules, "[0-9]" indica tots els dígits, etc

[^⋯] :Similar a l'anterior, però obliga a què els caracters possibles NO siguin els especificats dins dels claudàtors. Per exemple,”c[^iu]sa” pot ser “cosa”, “casa”, “cbsa”, etc, però mai “cisa” ó “cusa” (com tampoc paraules que tinguin menys ni més que quatre lletres. També es pot posar un rang de caràcters no desitjat: per exemple "[^a-d]" indica qualsevol caràcter excepte "a","b","c" o "d".

NOTA: Si es vol incloure el símbol "-" com un símbol més dins del claudàtors i fer llavors que perdi el seu significat especial d'indicador de rang de caràcters, s'ha d'"escapar". "Escapar un caràcter" (en aquest cas el guió) vol dir fer-li perdre el seu significat especial i convertir-lo en el caràcter tal qual; això es pot fer per qualsevol altre símbol que tingui algun significat concret en les expressions regulars, com el punt, els claudàtors i els propers que veurem. Per escapar un caràcter simplement cal precedir-lo del símbol "\". Així, doncs, en aquest exemple, per incloure el símbol "-" com un símbol més caldria escriure "\-".

^⋯ : Indica que els caràcters escrits després del ^ han d'aparèixer obligatòriament al principi de la línia
⋯$: Indica que els caràcters escrits abans del $ han d'aparèixer obligatòriament al final de la línia
^⋯$: Combinació dels dos anteriors: indica que els caràcters escrits han de ser els únics existents a la línia
\<... : Indica que els caràcters escrits després del \< han d'aparèixer obligatòriament al principi d'una paraula
...\> : Indica que els caràcters escrits abans del \> han d'aparèixer obligatòriament al final d'una paraula

***** : Indica que el caràcter anterior pot aparèixer 0 o més vegades repetit consecutivament (per tant,pot no aparèixer)
+ : Indica que el caràcter anterior pot aparèixer 1 o més vegades repetit consecutivament
? : Indica que el caràcter anterior pot aparèixer 0 o 1 vegades, no més.
{x} : Indica que el caràcter anterior ha d'aparèixer exactament un número x de vegades consecutivament.
{x,} : Indica que el caràcter anterior ha d'aparèixer com a mínim un número x de vegades consecutivament. Es pot

veure que l'expressió "{0,}" seria equivalent a "*" i "{1,}" a "+"
{x,y} : Indica que el caràcter anterior ha d'aparèixer com a mínim un número x de vegades consecutivament i com a màxim un número y de vegades. Es pot veure que l'expressió "{0,1}" seria equivalent a "?"

(...) : Serveix per agrupar expressions. S'usa molt juntament amb els símbols *,+,?,{nº...} per a què aquests no afectin només al caràcter d'abans sinò a tot el conjunt dins dels parèntesis. Per exemple, l'expressió "(ab){3}" equival a "abababab".

(...|...) : Indica que o bé és una expressió -la de l'esquerra de la barra vertical- o bé una altra -la de la dreta-. Per exemple, l'expressió "(ca|co)sa" pot equivaler tant a "casa" com a "cosa".

A més dels símbols anteriors, també es poden escriure expressions especials que permeten simplificar una mica la construcció d'expressions regulars. Per exemple:

[[:alnum:]] : Equival a [0-9A-Za-z]
[[:alpha:]] : Equival a [A-Za-z]
[[:lower:]] : Equival a [a-z]
[[:upper:]] : Equival a [A-Z]
[[:digit:]] : Equival a [0-9]
[[:xdigit:]] : Equival a [0-9A-Fa-f]
[[:punct:]] : Equival a un dels caràcters següents: `! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [ \ ] ^ _ ` { | } ~`
[[:graph:]] : Equival a [[:alnum:]] + [[:punct:]]
[[:print:]] : Equival a [[:graph:]] + l'espai en blanc
[[:blank:]] : Equival a l'espai en blanc i al tabulador
[[:space:]] : Equival a l'espai en blanc, al tabulador, al salt de línia, retorn de carro i tabulador vertical

Un exemple: per trobar codis postals, podríem utilitzar l'expressió "[[:digit:]]{5}"

Per saber més sobre expressions regulars, recomano la consulta de http://www.regular-expressions.info

NOTA: Per a què grep accepti els símbols \t o \n s'ha d'escriure el paràmetre -P en comptes de -E. Una altra opció seria usar "ansi c quotes" així (¡aquí les cometes simples són importants!): *grep -E $'.{2}\t.{3}' fitxer.txt* (https://www.gnu.org/software/bash/manual/html_node/ANSI_002dC-Quoting.html) ; bàsicament la idea és que escrivint $'...' l'interior de les cometes s'interpreta pel shell directament. Si no es vol aquesta opció tampoc, sempre es pot pulsar el tabulador explícitament amb CTRL+V+TAB

La comanda *sed* és un editor de text no interactiu. És a dir, serveix pel mateix que un *nano* o un *gedit* però executant les ordres d'edició directament sense "obrir" mai l'arxiu a modificar. Això fa que sigui molt útil per scripts, per exemple. A continuació es mostren alguns exemples d'ús:

sed -i "s/antiga/nova/g" unarxiu Canvia la paraula “antiga” per la “nova” a tot arreu de l'arxiu
Si no s'escriu el "/g", el canvi només es farà a la 1ª ocurrència d'"antiga" que hi hagi a cada línia, no a totes les que hi puguin haver a cada línia.
El paràmetre -i és per a què es faci efectiu el canvi: si no només es veurà per pantalla però no es gravarà al fitxer. Si es vol guardar una còpia de l'arxiu abans de fer la modificació, s'haurà d'indicar just darrera de -i el sufixe que es vol afegir després del nom de l'arxiu (creant-se llavors una còpia anomenada "unarxiu.sufixe")

La paraula "antiga" podria haver sigut una expressió regular. Per a què *sed* les pugui reconèixer, cal que afegim el paràmetre *-r* (a l'igual que passava amb la comanda grep i el seu paràmetre -E) . Per exemple, *sed -ir "s/^$/###/" unarxiu* substitueix les línies en blanc per línies de tres coixinets. Un altre exemple: *sed -ir "s/^[\t]*//" unarxiu* esborra tots els espais en blanc o tabuladors que hi hagi al començament de totes les línies. Es poden combinar més d'una acció SED si s'escriuen separades per punt i comes; per exemple *sed -ir "s/^[\t]*//;s/[\t]*$//" unarxiu* esborra tants els espais i tabuladors del començament de les línies com dels seus finals.

NOTA: En realitat, existeix un subconjunt bàsic d'expressions regulars que es poden utilitzar sense indicar el paràmetre -r (o el -E del grep) però per accedir al conjunt complet (anomenat "POSIX Extended") sí que cal utilitzar-los, així que per simplicitat farem servir sempre aquest/s paràmetre/s i ja està.
NOTA: En realitat, el símbol "/" per separar la "s" d'"antiga" i aquesta de "nova" i aquesta de "g" podria ser qualsevol altre, però per conveni sempre s'utilitza aquest (a no ser que hi hagi algun "/" en "antiga" o "nova"...en aquest cas, per evitar confusions, se sol fer servir el símbol "|", així, per exemple: s|http://|https://|g"

Altres exemples de la comanda *sed* són els següents:

sed -i "56d" unarxiu : Esborra la línia número 56 de l'arxiu.

*Per indicar un rang de línies (per exemple de la 4ª a la 7ª, ambdues incloses, s'ha d'escriure *sed -i "4,7d" unarxiu* .
*Per indicar un rang amb una expressió regular d'inici i una altra de final, així: *sed -nr "/start/,/end/d" unarxiu* (en aquest cas, esborraria totes les línies entre la primera on apareix la paraula "start" i la primera on apareix "end").
*Es pot fer servir també expressions regulars per només esborrar les línies que coincideixin amb el patró (per exemple, *sed -ir "/^$/d" unarxiu* esborra les línies en blanc.
*O bé indicar un número de línies determinades a partir d'una dada, així: *sed -i "4,+7d" unarxiu* (en aquest cas, esborraria la quarta línia i les set següents).
D'altra banda, es pot fer servir el símbol especial "$", que indica "la última línia", així: *sed -i "$d" unarxiu

sed -i "3i frase" unarxiu : Afegeix la frase indicada a la línia nº 3 (desplaçant la que fins llavors era la 3ª a la 4ª posició la 4ª a la 5ª, etc).

*Es pot fer servir també expressions regulars per només afegir la frase a la posició de les línies que -fins llavors- coincideixen amb el patró (per exemple, *sed -ir "/^$/i Hola" unarxiu* afegeix la paraula "Hola" just a la línia d'abans de les línies en blanc

sed -n "4p" unarxiu : Mostra només la línia indicada. El paràmetre -n és per a què no es mostrin les línies originals de l'arxiu abans de tractar.

*També es poden fer servir rangs (del tipus "4,7" o "4,+7")...
*...i expressions regulars (per exemple, per mostrar només les línies que tinguin 65 caràcters o més, es pot fer: *sed -nr "/^.{65,}/p" unarxiu*.
*També es pot indicar un rang amb una expressió regular d'inici i una altra de final, per exemple: *sed -nr "/start/,/end/p" unarxiu*.
*D'altra banda, es pot fer servir a més la notació *x~y* on "x" indica un número de línia inicial i "y" indica l'intèrval a partir d'ella (per exemple, *sed -n "1~2" unarxiu* mostra només les línies impars).

sed -i "y/abcde/nopqr/" unarxiu : Substitueix cada ocurrència de "a" per "n", cada "b" per "o", "c" per "p","d" per "q" i "e" per "r".

Aquesta comanda es podria fer servir per canviar de minúscules a majúscules (o viceversa), per exemple:
sed -i "y/abcdefghijklmnopqrstuvwxyz/ABCDEFGHIJKLMNOPQRSTUVWXYZ/ El que és important és que hi hagi el mateix número de caràcters a substituir que caràcters subtituts.

***Altres:**

comanda1 && comanda2 El símbol && serveix per executar primer comanda1, i si aquesta s'ha executat bé, (és a dir si el seu valor de retorn $? és igual a 0) passa a executar comanda2

comanda1 || *comanda2* El símbol || serveix per executar primer comanda1, i només si aquesta s'ha executat mal (és a dir si el seu valor de retorn $? és diferent de 0), passa a executar comanda2

comanda1 ; comanda2 El símbol ; serveix per executar primer comanda1, i independenment de si aquesta s'ha executat bé o no, passa sempre seguidament a executar comanda2

sleep nº S'espera el número de segons indicat. Útil quan es fa servir amb &&

watch -n nº "comanda" Repeteix l'execució de la comanda indicada infinites vegades (fins pulsar CTRL+C) cada nº segons. Per exemple: *watch -t -n 1 "date '+%D%n%T'| figlet -k"*

-t No mostra l'encapçalament on es veu el pas del temps

-d Assenyala gràficament a cada repetició les diferències de la sortida actual respecte l'anterior

echo "scale=4;4/3" | bc La comanda *bc* serveix per fer càlculs matemàtics a la consola. Pot fer-se servir interactivament o bé passar-li les operacions a través d'una tuberia. En aquest exemple, abans de realitzar la divisió indicada s'informa a *bc* del valor de la variable interna "scale", el qual correspon al número de decimals que es vol mostrar al resultat (per defecte és 0). Es poden executar vàries operacions si s'escriuen una darrera de l'altra separades per punt i coma. Per executar operacions més sofisticades (trigonomètriques, logarítmiques...) cal afegir-li el paràmetre -l (veure apartat "Math library" del manual)

Per saber més, recomano consultar les webs http://wiki.bash-hackers.org i http://mywiki.wooledge.org/FullBashGuide (també és molt interessant la llista de preguntes més freqüents a http://mywiki.wooledge.org/BashFAQ i la lllista d'errors més freqüents a http://mywiki.wooledge.org/BashPitfalls). Un altre lloc més senzill però també molt recomanable és https://linuxjourney.com