ASIX M01

A01-04 Shell Bàsic I:

- Pipes, Redirect, Shell variables
- Comands, alias
- Expansions

Índex de continguts

Redireccionaments, pipes, variables d'entorn i filtres

Redireccions i pipes

Dispositius estàndard.

a) Entrada estàndard: stdin (handler 0).
b) Sortida estàndard: stdout (handler 1).
c) Sortida estàndard d'error: stderr (handler 2).
d) Altres dispositius: /dev/null

```
[root@portatil mp1]# ll /dev/std
stderr  stdin   stdout

[root@portatil mp1]# ll /dev/std*
lrwxrwxrwx 1 root root 15 31 oct 04:13 /dev/stderr -> /proc/self/fd/2
lrwxrwxrwx 1 root root 15 31 oct 04:13 /dev/stdin -> /proc/self/fd/0
lrwxrwxrwx 1 root root 15 31 oct 04:13 /dev/stdout -> /proc/self/fd/1

[root@portatil mp1]# ls -la carta.txt pere
ls: no s'ha pogut accedir a pere: El fitxer o directori no existeix      // stderr
-rw-r--r-- 1 root root 56 31 oct 08:49 carta.txt                        //stdout

[root@portatil mp1]# ll /dev/null /dev/zero /dev/random
crw-rw-rw- 1 root root 1, 3 31 oct 04:13 /dev/null
crw-rw-rw- 1 root root 1, 8 31 oct 04:13 /dev/random
crw-rw-rw- 1 root root 1, 5 31 oct 04:13 /dev/zero
```

Redireccionaments bàsics..

a) Redireccionaments: de sortida estàndard: >, >>.
b) Redireccionaments de sortida d'error: 2>, 2>>.
c) Redireccionament d'entrada: <.

```
[root@portatil mp1]# ll -la > llistat
[root@portatil mp1]# cat llistat
total 12
drwxr-xr-x  2 root root 4096 31 oct 08:54 .
drwxrwxrwx 20 root root 4096 31 oct 08:47 ..
-rw-r--r--  1 root root   56 31 oct 08:49 carta.txt
-rw-r--r--  1 root root    0 31 oct 08:54 llistat

[root@portatil mp1]# ls >> llistat
[root@portatil mp1]# cat llistat
total 12
drwxr-xr-x  2 root root 4096 31 oct 08:54 .
```

```
drwxrwxrwx 20 root root 4096 31 oct 08:47 ..
-rw-r--r--  1 root root   56 31 oct 08:49 carta.txt
-rw-r--r--  1 root root    0 31 oct 08:54 llistat
carta.txt
llistat

[root@portatil mp1]# uname -a > info.txt
[root@portatil mp1]# date >> info.txt
[root@portatil mp1]# id >> info.txt
[root@portatil mp1]# cat info.txt
Linux portatil 2.6.31.12-174.2.3.fc12.i686.PAE #1 SMP Mon Jan 18 20:06:44 UTC 2010 i686 i686
i386 GNU/Linux
dg oct 31 08:56:28 EDT 2010
uid=0(root) gid=0(root) grups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
```

```
[root@portatil mp1]# ls carta.txt pere
ls: no s'ha pogut accedir a pere: El fitxer o directori no existeix
carta.txt
[root@portatil mp1]# ls carta.txt pere 2> /dev/null
carta.txt
[root@portatil mp1]# ls carta.txt pere 2> errors.txt
carta.txt
[root@portatil mp1]# cat errors.txt
ls: no s'ha pogut accedir a pere: El fitxer o directori no existeix

[root@portatil mp1]# id pere 2>> errors.txt
[root@portatil mp1]# date -pimpam 2>> errors.txt
[root@portatil mp1]# cat errors.txt
ls: no s'ha pogut accedir a pere: El fitxer o directori no existeix
id: pere: l'usuari no existeix
date: l'opció «p» no és vàlida
Proveu «date --help» per a obtenir més informació.
```

```
[root@portatil mp1]# ll /boot/ > llistat
[root@portatil mp1]# sort -r < llistat
total 49564
-rwxr-xr-x 1 root root  3649984 27 ago 02:51 vmlinuz-2.6.32.21-166.fc12.i686.PAE
-rwxr-xr-x 1 root root  3648960 30 abr  2010 vmlinuz-2.6.32.12-115.fc12.i686.PAE
-rwxr-xr-x 1 root root  3461952 18 gen  2010 vmlinuz-2.6.31.12-174.2.3.fc12.i686.PAE
-rw-r--r-- 1 root root  1670661 30 abr  2010 System.map-2.6.32.12-115.fc12.i686.PAE
-rw-r--r-- 1 root root  1652396 27 ago 02:51 System.map-2.6.32.21-166.fc12.i686.PAE
-rw-r--r-- 1 root root  1488919 18 gen  2010 System.map-2.6.31.12-174.2.3.fc12.i686.PAE
-rw-r--r-- 1 root root 11707260  6 set 13:38 initramfs-2.6.32.21-166.fc12.i686.PAE.img
-rw-r--r-- 1 root root 11686793 20 mai 11:49 initramfs-2.6.32.12-115.fc12.i686.PAE.img
-rw-r--r-- 1 root root 11344107 22 gen  2010 initramfs-2.6.31.12-174.2.3.fc12.i686.PAE.img
-rw-r--r-- 1 root root   107382 27 ago 02:51 config-2.6.32.21-166.fc12.i686.PAE
```

```
-rw-r--r-- 1 root root   107314 30 abr  2010 config-2.6.32.12-115.fc12.i686.PAE
-rw-r--r-- 1 root root   103728 18 gen  2010 config-2.6.31.12-174.2.3.fc12.i686.PAE
drwxr-xr-x 3 root root     4096 14 des  2009 efi
drwxr-xr-x 2 root root     4096  6 set 13:51 grub

[root@portatil mp1]# wc < llistat
  15  128 1123
```

```
[root@portatil mp1]# sort -r < llistat > ordenat.txt
[root@portatil mp1]# cat ordenat.txt
total 49564
-rwxr-xr-x 1 root root  3649984 27 ago 02:51 vmlinuz-2.6.32.21-166.fc12.i686.PAE
...
drwxr-xr-x 2 root root     4096  6 set 13:51 grub

[root@portatil mp1]# ls carta.txt pere > llistat 2> errors.txt
[root@portatil mp1]# cat llistat
carta.txt
[root@portatil mp1]# cat errors.txt
ls: no s'ha pogut accedir a pere: El fitxer o directori no existeix

[root@portatil mp1]# ll carta.txt pere 2> /dev/null > llistat
[root@portatil mp1]# cat llistat
-rw-r--r-- 1 root root 56 31 oct 08:49 carta.txt
```

Redireccionaments avançats.

a) Redirigir &>, (>&) la sortida a un altre redirecció, 2>&1, (1>&2).
b) Idem affegir &>>, >>word 2>&1.
c) Redirecció <&.
d) Documents aquí (document here -).
e) Document TAG.

```
[root@portatil mp1]# ls carta.txt pere &> both.txt
[root@portatil mp1]# cat both.txt
ls: no s'ha pogut accedir a pere: El fitxer o directori no existeix
carta.txt

[root@portatil mp1]# ls carta.txt pere >& both.txt
[root@portatil mp1]# cat both.txt
ls: no s'ha pogut accedir a pere: El fitxer o directori no existeix
carta.txt

[root@portatil mp1]# ls carta.txt pere >  both.txt 2>&1
[root@portatil mp1]# cat both.txt
ls: no s'ha pogut accedir a pere: El fitxer o directori no existeix
```

```
carta.txt

[root@portatil mp1]# ls carta.txt pere 2>  both.txt 1>&2
[root@portatil mp1]# cat both.txt
ls: no s'ha pogut accedir a pere: El fitxer o directori no existeix
carta.txt

[root@portatil mp1]# ls carta.txt pere  2>&1 > mal_fet.txt       // error a stdout no al file
ls: no s'ha pogut accedir a pere: El fitxer o directori no existeix
[root@portatil mp1]# cat mal_fet.txt
carta.txt

[root@portatil mp1]# ls carta.txt pere  &> errors.txt
[root@portatil mp1]# cat errors.txt
ls: no s'ha pogut accedir a pere: El fitxer o directori no existeix
carta.txt
[root@portatil mp1]# id pere &>> errors.txt
[root@portatil mp1]# date -pimpam &>> errors.txt
[root@portatil mp1]# cat errors.txt
ls: no s'ha pogut accedir a pere: El fitxer o directori no existeix
carta.txt
id: pere: l'usuari no existeix
date: l'opció «p» no és vàlida
Proveu «date --help» per a obtenir més informació.
```

```
[root@portatil mp1]# cat llistat - llistat > resultat.txt
cal entrar per stdin el text
corresponent al document here
i acabar amb crtl+d
[root@portatil mp1]# cat resultat.txt
-rw-r--r-- 1 root root 56 31 oct 08:49 carta.txt
cal entrar per stdin el text
corresponent al document here
i acabar amb crtl+d
-rw-r--r-- 1 root root 56 31 oct 08:49 carta.txt

[root@portatil mp1]# sort > noms.txt
nom1
nom5
nom3
[root@portatil mp1]# cat noms.txt
nom1
nom3
nom5
[root@portatil mp1]# sort - > noms.txt
nom3
nom1
```

```
nom2
[root@portatil mp1]# cat noms.txt
nom1
nom2
nom3

[root@portatil mp1]# cat llistat <<FI llistat > resultat.txt
> escriure la entrada
> fins que es troba el TAG
> que ha d'estar en una línia sol
> FI
[root@portatil mp1]# cat resultat.txt
-rw-r--r-- 1 root root 56 31 oct 08:49 carta.txt
-rw-r--r-- 1 root root 56 31 oct 08:49 carta.txt

[root@portatil mp1]# sort  <<FI | wc
> linia1
> lini4
> linia3
> FI
     3       3      20
```

Pipes.

a) Concatenació d'ordres: pipe |.
b) Diferenciar entre redirecció i concatenació. No igual (ls > sort) que (ls | sort).
c) Ordre tee per duplicar la sortida.

```
[pere@portatil ~]$ cat /etc/services | grep domain | sort -r
domaintime      9909/udp                # domaintime
domaintime      9909/tcp                # domaintime
domain          53/udp
domain          53/tcp                    # name-domain server


[pere@portatil ~]$ cat /etc/services | grep domain | sort -r > llistat
[pere@portatil ~]$ cat llistat
domaintime      9909/udp                # domaintime
domaintime      9909/tcp                # domaintime
domain          53/udp
domain          53/tcp                    # name-domain server

-bash-4.0$ ll > sort
-bash-4.0$ ll | sort
-rw-rw-r-- 1 user1 user1 198  5 nov 12:15 llistat
-rw-rw-r-- 1 user1 user1 198  5 nov 12:15 majuscules.txt
-rw-rw-r-- 1 user1 user1 213  5 nov 12:17 sort
```

```
-rw-rw-r-- 1 user1 user1  50  5 nov 12:17 sortida
total 16

-bash-4.0$ ll . fit-inexistent | sort > llistat 2> /dev/null
ls: no s'ha pogut accedir a fit-inexistent: El fitxer o directori no existeix
-bash-4.0$ ll . fit-inexistent 2> /dev/null  | sort > llistat
-bash-4.0$ cat llistat
.:
-rw-rw-r-- 1 user1 user1   0  5 nov 12:19 llistat
-rw-rw-r-- 1 user1 user1 198  5 nov 12:15 majuscules.txt
-rw-rw-r-- 1 user1 user1 213  5 nov 12:17 sort
-rw-rw-r-- 1 user1 user1  50  5 nov 12:17 sortida
total 12
```

```
-bash-4.0$ cat /etc/services | grep domain | sort -r | tee llistat
domaintime      9909/udp                # domaintime
domaintime      9909/tcp                # domaintime
domain          53/udp
domain          53/tcp                  # name-domain server
-bash-4.0$ cat llistat
domaintime      9909/udp                # domaintime
domaintime      9909/tcp                # domaintime
domain          53/udp
domain          53/tcp                  # name-domain server

-bash-4.0$ cat /etc/services | grep domain | sort -r | tee llistat | wc -l
4
-bash-4.0$ cat llistat
domaintime      9909/udp                # domaintime
domaintime      9909/tcp                # domaintime
domain          53/udp
domain          53/tcp                  # name-domain server

-bash-4.0$ cat /etc/services | grep domain | sort -r | tee llistat | tr [a-z] [A-Z]
DOMAINTIME      9909/UDP                # DOMAINTIME
DOMAINTIME      9909/TCP                # DOMAINTIME
DOMAIN          53/UDP
DOMAIN          53/TCP                  # NAME-DOMAIN SERVER
-bash-4.0$ cat llistat
domaintime      9909/udp                # domaintime
domaintime      9909/tcp                # domaintime
domain          53/udp
domain          53/tcp                  # name-domain server

-bash-4.0$ cat /etc/services | grep domain | sort -r | tee llistat | tr [a-z] [A-Z] > majuscules.txt
-bash-4.0$ cat majuscules.txt
DOMAINTIME      9909/UDP                # DOMAINTIME
```

```
DOMAINTIME      9909/TCP                # DOMAINTIME
DOMAIN          53/UDP
DOMAIN          53/TCP                  # NAME-DOMAIN SERVER
-bash-4.0$ cat llistat
domaintime      9909/udp                # domaintime
domaintime      9909/tcp                # domaintime
domain          53/udp
domain          53/tcp                  # name-domain server
```

Variables d'entorn

Variables d'entorn:

a) Assignar valors: var=valor.
b) Mostrar valors: echo $valor.
c) Usar en expressions: var=”pre $var post”.
d) Anular el valor: unset.
e) Ull a la assignació:operador = sense espais al voltant (VAR=valor).

```
-bash-4.0$ echo $HOME
/tmp/user1
-bash-4.0$ echo $PWD
/tmp/m01

-bash-4.0$ NOM="alumne pere"
-bash-4.0$ echo $NOM
alumne pere
-bash-4.0$ INFO="nom: $USER shell: $BASH"
-bash-4.0$ echo $INFO
nom: user1 shell: /bin/bash
-bash-4.0$ set | grep INFO
INFO='nom: user1 shell: /bin/bash'

-bash-4.0$ unset NOM
-bash-4.0$ unset INFO
-bash-4.0$ echo $NOM $INFO
..res..
```

Variables típiques del sistema.

a) HOME
b) IFS
c) SHELL, SHLVL.

d) MAIL
e) PATH
f) PS1, PS2
g) TERM
h) USER, LOGINAME.

```
-bash-4.0$ echo $HOME $UID $EUID $SHELL
/tmp/user1 504 504 /bin/bash

-bash-4.0$ echo $USER  $TERM $BASH $SHLVL
user1 xterm /bin/bash 1

-bash-4.0$ echo $PS1
\s-\v\$
-bash-4.0$ echo $PS2
>

-bash-4.0$ set | head -25
BASH=/bin/bash
BASH_ALIASES=()
BASH_ARGC=()
BASH_ARGV=()
BASH_CMDS=()
BASH_LINENO=()
BASH_SOURCE=()
BASH_VERSINFO=([0]="4" [1]="0" [2]="38" [3]="1" [4]="release" [5]="i386-redhat-linux-gnu")
BASH_VERSION='4.0.38(1)-release'
COLORS=/etc/DIR_COLORS
COLUMNS=125
CVS_RSH=ssh
DIRSTACK=()
DISPLAY=:0.0
EUID=504
GROUPS=()
G_BROKEN_FILENAMES=1
HISTCONTROL=ignoreboth
HISTFILE=/tmp/user1/.bash_history
HISTFILESIZE=1000
HISTSIZE=1000
HOME=/tmp/user1
HOSTNAME=portatil
HOSTTYPE=i386
IFS=$' \t\n'
```

Tipus de valors de variables d'entorn.

a) 'literals'.

b) “text que permet expansió”.
c) Valors numèrics.
d) Valors no entre quotes.

-bash-4.0$ **NOM="molt bonic"**
-bash-4.0$ echo $NOM
molt bonic
-bash-4.0$ **NOM=poc bonic**
-bash: bonic: no s'ha trobat l'ordre
-bash-4.0$ echo $NOM
molt bonic

-bash-4.0$ **NOM="cadena de caracters amb espais que expandeix"**
-bash-4.0$ echo $NOM
cadena de caracters amb espais que expandeix
-bash-4.0$ **NOM="usuari: $USER, uid: $UID, shell: $BASH"**
-bash-4.0$ echo $NOM
usuari: user1, uid: 504, shell: /bin/bash
-bash-4.0$ **set | grep NOM**
NOM='usuari: user1, uid: 504, shell: /bin/bash'

-bash-4.0$ **NOM='literal que no expandeix'**
-bash-4.0$ echo $NOM
literal que no expandeix
-bash-4.0$ **NOM='usuari: $USER, uid: $UID, shell: $BASH'**
-bash-4.0$ **echo $NOM**
usuari: $USER, uid: $UID, shell: $BASH
-bash-4.0$ s**et | grep NOM**
NOM='usuari: $USER, uid: $UID, shell: $BASH'

-bash-4.0$ EDAT=15
-bash-4.0$ echo $EDAT
15
-bash-4.0$ EDAT=23 12
-bash: 12: no s'ha trobat l'ordre
-bash-4.0$ echo $EDAT
15

-bash-4.0$ **EDAT=$UID**
-bash-4.0$ echo $EDAT
504
-bash-4.0$ **EDAT=$UID $EUID**
-bash: 504: no s'ha trobat l'ordre
-bash-4.0$ echo $EDAT
504

Desar les variables.

e) Ordres de manipulació: set i env.
f) Filtrar per buscar una determinada variable: set | grep “VAR”.

```
-bash-4.0$ help set
-bash-4.0$ man bash

-bash-4.0$ set | grep -E "NOM|EDAT"
EDAT=504
NOM='literal que no expandeix'

-bash-4.0$ env
HOSTNAME=portatil
SHELL=/bin/bash
TERM=xterm
HISTSIZE=1000
PT5HOME=/usr/local/PacketTracer5
QTDIR=/usr/lib/qt-3.3
OLDPWD=/tmp/mp1
QTINC=/usr/lib/qt-3.3/include
USER=user1
...

-bash-4.0$ env | grep -E "NOM|EDAT"
...res...

-bash-4.0$ export NOM
-bash-4.0$ env | grep  "NOM"
NOM=literal que no expandeix

-bash-4.0$ export CURS="hisx1"
-bash-4.0$ set | grep CURS
CURS=hisx1
-bash-4.0$ env | grep CURS
CURS=hisx1
```

Àmbit de les variables.

a) Sessió local (variables del set).
b) Exportar a un sub shell amb export (variables del environment: env).
c) Desar permanentment en .bash_profile.
d) Variable SHLVL.

```
-bash-4.0$ echo $SHELL
```

```
/bin/bash
-bash-4.0$ echo $SHLVL
1
-bash-4.0$ ps
  PID TTY          TIME CMD
 2663 pts/2    00:00:00 bash
 3281 pts/2    00:00:00 ps

-bash-4.0$ bash
bash-4.0$ ps
  PID TTY          TIME CMD
 2663 pts/2    00:00:00 bash
 3282 pts/2    00:00:00 bash
 3284 pts/2    00:00:00 ps
bash-4.0$ pstree -u user1
bash───bash───pstree
bash-4.0$ echo $SHLVL
2
bash-4.0$ echo $NOM
literal que no expandeix     // ok perquè està a env
bash-4.0$ echo $EDAT
...res..
bash-4.0$ env | grep NOM
NOM=literal que no expandeix
bash-4.0$ env | grep EDAT
...res...

bash-4.0$ export EDAT=25
bash-4.0$ bash
bash-4.0$ pstree -u user1
bash───bash───bash───pstree
bash-4.0$ echo $SHLVL
3
bash-4.0$ echo $EDAT
25
bash-4.0$ env | grep EDAT
EDAT=25

bash-4.0$ export COGNOM="pou prat"
bash-4.0$ env | grep COGNOM
COGNOM=pou prat
bash-4.0$ exit
exit
bash-4.0$ env | grep COGNOM
bash-4.0$ echo $SHLVL
2
```

```
bash-4.0$ cat /etc/profile
# /etc/profile
# System wide environment and startup programs, for login setup
# Functions and aliases go in /etc/bashrc
# It's NOT good idea to change this file unless you know what you
# are doing. Much better way is to create custom.sh shell script in
# /etc/profile.d/ to make custom changes to environment. This will
# prevent need for merging in future updates.
pathmunge () {
case ":${PATH}:" in
*:"$1":*)
;;
*)
if [ "$2" = "after" ] ; then
PATH=$PATH:$1
else
PATH=$1:$PATH
fi
esac
}

if [ -x /usr/bin/id ]; then
if [ -z "$EUID" ]; then
# ksh workaround
EUID=`id -u`
UID=`id -ru`
fi
USER="`id -un`"
LOGNAME=$USER
MAIL="/var/spool/mail/$USER"
fi

# Path manipulation
if [ "$EUID" = "0" ]; then
pathmunge /sbin
pathmunge /usr/sbin
pathmunge /usr/local/sbin
else
pathmunge /usr/local/sbin after
pathmunge /usr/sbin after
pathmunge /sbin after
fi

HOSTNAME=`/bin/hostname 2>/dev/null`
HISTSIZE=1000
HISTCONTROL="ignoreboth"

export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE HISTCONTROL
```

```
for i in /etc/profile.d/colorls.sh /etc/profile.d/cvs.sh /etc/profile.d/devkit-disks-bash-completion.sh
/etc/profile.d/glib2.sh /etc/profile.d/gnome-ssh-askpass.sh /etc/profile.d/kde.sh
/etc/profile.d/kmix_pulseaudio_disable.sh /etc/profile.d/krb5-workstation.sh /etc/profile.d/lang.sh
/etc/profile.d/less.sh /etc/profile.d/mc.sh /etc/profile.d/qt.sh /etc/profile.d/vim.sh
/etc/profile.d/which2.sh ; do
if [ -r "$i" ]; then
if [ "$PS1" ]; then
. $i
else
. $i >/dev/null 2>&1
fi
fi
done
unset i
unset pathmunge
PT5HOME=/usr/local/PacketTracer5
export PT5HOME

bash-4.0$ cat /etc/bashrc
# /etc/bashrc
# System wide functions and aliases
# Environment stuff goes in /etc/profile
# It's NOT good idea to change this file unless you know what you
# are doing. Much better way is to create custom.sh shell script in
# /etc/profile.d/ to make custom changes to environment. This will
# prevent need for merging in future updates.
# By default, we want this to get set.
# Even for non-interactive, non-login shells.
# Current threshold for system reserved uid/gids is 200
# You could check uidgid reservation validity in
# /usr/share/doc/setup-*/uidgid file
if [ $UID -gt 199 ] && [ "`id -gn`" = "`id -un`" ]; then
   umask 002
else
   umask 022
fi
...
```

```
bash-4.0$ cat /etc/skel/.bash_profile
# .bash_profile
# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi
# User specific environment and startup programs
PATH=$PATH:$HOME/bin
```

```
NOM='variable definida per l'usuari permanentment'
 export PATH NOM

bash-4.0$ cat /etc/skel/.bashrc
# .bashrc
# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi
# User specific aliases and functions
alias browser='nautilus --no-desktop –browser'  #alias fet per l'usuari
```

Definir el prompt.

a) Variables del prompt: PS1, PS2.
b) Seqüències d'escape amb valors predefinits.

```
[root@portatil ~]# echo $PS1
[\u@\h \W]\$

bash-4.0$ echo $PS1
\s-\v\$

bash-4.0$ OLDPS1=$PS1
bash-4.0$ echo $OLDPS1
\s-\v\$
bash-4.0$ PS1='nou-prompt \s \u \$'
nou-prompt bash user1 $ echo $PS1
nou-prompt \s \u \$
nou-prompt bash user1 $id
uid=504(user1) gid=505(user1) grups=505(user1)

nou-prompt bash user1 $PS1=$OLDPS1
bash-4.0$
```

man bash: part de variables del sistema

PS1 The value of this parameter is expanded (see PROMPTING below) and used as the primary prompt string. The default value is ‘‘\s-\v\$ ’’.

PS2 The value of this parameter is expanded as with PS1 and used as the secondary prompt string. The default is ‘‘> ’’.

PS3 The value of this parameter is used as the prompt for the select command (see SHELL GRAMMAR above).

PS4 The value of this parameter is expanded as with PS1 and the value is printed before each command bash displays during an execution trace. The first character of PS4 is replicated multiple times,

man bash: PROMPTING

When executing interactively, bash displays the primary prompt PS1 when it is ready to read a command, and the secondary prompt PS2 when it needs more input to complete a command. Bash allows these prompt strings to be customized by inserting a number of backslash-escaped special characters that are decoded as follows:

\a an ASCII bell character (07)
\d the date in "Weekday Month Date" format (e.g., "Tue May 26")

\D{format}
the format is passed to strftime(3) and the result is inserted into the prompt string; an empty format results in a locale-specific time representation. The braces are required
\e an ASCII escape character (033)
\h the hostname up to the first '.'
\H the hostname
\j the number of jobs currently managed by the shell
\l the basename of the shell's terminal device name
\n newline
\r carriage return
\s the name of the shell, the basename of $0 (the portion following the final slash)
\t the current time in 24-hour HH:MM:SS format
\T the current time in 12-hour HH:MM:SS format
\@ the current time in 12-hour am/pm format
\A the current time in 24-hour HH:MM format
\u the username of the current user
\v the version of bash (e.g., 2.00)
\V the release of bash, version + patch level (e.g., 2.00.0)
\w the current working directory, with $HOME abbreviated with a tilde (uses the $PROMPT_DIRTRIM variable)
\W the basename of the current working directory, with $HOME abbreviated with a tilde
\! the history number of this command
\# the command number of this command
\$ if the effective UID is 0, a #, otherwise a $
\nnn the character corresponding to the octal number nnn
\\ a backslash
\[begin a sequence of non-printing characters, which could be used to embed a terminal control sequence into the prompt
\] end a sequence of non-printing characters

The command number and the history number are usually different: the history number of a command is its position in the history list, which may include commands restored from the history file (see HISTORY below), while the command number is the position in the sequence of commands executed during the current shell session. After the string is decoded, it is expanded via parameter expansion, command substitution, arithmetic expansion, and quote removal, subject to the value of the promptvars shell option (see the description of the shopt command under SHELL BUILTIN COMMANDS below).

Filtres

Filtres bàsics:.

a) Retallar per columnes: cut.
b) Comptar paraules i línies: wc.
c) Ordenar lexico-gràficament: sort.
d) Buscar cadenes amb patrons: grep.

Ordres:

stdin, stdout, stderr, null, >, >>, 2>, 2>>, <, >&, 2>& , - (document here), document TAG, pipe, tee, shell variables, env, set, unset, .bashrc, .bash_profile, prompt, expressions (literals, cadenes, valors), valor de retorn $?.
Filtres: grep, sort, cut, wc

Shell Bàsic

Shell bàsic

Path.

a) Llista de camins alternatius. Indica una precedència.
b) Execució d'ordres: $ ordre -enter. 1) interpretar i expandir l'ordre. 2) executar.
c) Execució: 1) ordres internes. 2) llista de camins alternatius del path.
d) Les ordres externes són fitxers executables (normals).
e) Unix NO executa per defecte fitxers del directori actiu.
f) Tota ordre es pot executar si s'indica la seva trajectòria absoluta.
g) Path permet usar diferents versions (personalitzar) d'ordres.

```
bash-4.0$ echo $PATH
/usr/lib/qt-3.3/bin:/usr/kerberos/sbin:/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:/usr/local/sbin: \
/usr/sbin:/sbin

[root@portatil ~]# echo $PATH
/usr/lib/qt3.3/bin:/usr/kerberos/sbin:/usr/kerberos/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin: \
/usr/sbin:/usr/bin:/root/bin

bash-4.0$ tree -d -L 3 / | grep bin
├── bin
│   │   ├── bin
│   ├── bin
│   ├── sbin
│   │   ├── bin
│   │   ├── sbin
├── sbin
│   ├── bin
│   │   ├── bin
│   │   ├── sbin
│   │   ├── kconf_update_bin
│   │   ├── bin
│   │   ├── sbin
│   ├── sbin
│   │   ├── rpcbind
│   │   ├── cgi-bin
│   │   └── binding
│   │   ├── bin
```

```
# ordres externes i alias
[root@portatil ~]# whereis pwd
```

```
pwd: /bin/pwd /usr/include/pwd.h /usr/share/man/man1/pwd.1.gz /usr/share/man/man1p/pwd.1p.gz
[root@portatil ~]# which pwd
/bin/pwd

[root@portatil ~]# whereis ls
ls: /bin/ls /usr/share/man/man1/ls.1.gz /usr/share/man/man1p/ls.1p.gz

[root@portatil ~]# which ls
alias ls='ls --color=auto'
        /bin/ls

bash-4.0$ pwd
/tmp/m01
bash-4.0$ /bin/pwd
/tmp/m01

[ecanet@portatil ~]$ ls
Baixades  Documents  edt  Escriptori  Imatges  llistat  Música  Plantilles  Públic  Vídeos
[ecanet@portatil ~]$ /bin/ls
Baixades  Documents  edt  Escriptori  Imatges  llistat      MúsicaPlantilles  Públic  Vídeos

bash-4.0$ /usr/bin/id
uid=504(user1) gid=505(user1) grups=505(user1)
bash-4.0$ cd /usr/bin
bash-4.0$ ./id
uid=504(user1) gid=505(user1) grups=505(user1)
```

```
# els programes del directori actiu no s'executen per defecte, cal la ruta
bash-4.0$ cat <<FI > programa.sh
> #! /bin/bash
> echo "hola soc un programa"
> echo $USER $UID $EUID
> date
> FI
bash-4.0$ chmod 755 programa.sh
bash-4.0$ programa.sh
bash: programa.sh: no s'ha trobat l'ordre
bash-4.0$ ./programa.sh
hola soc un programa
user1 504 504
dv nov  5 19:07:49 EDT 2010
```

```
# ordres internes
[root@portatil ~]# whereis cd
cd: /usr/share/man/man1/cd.1.gz /usr/share/man/man1p/cd.1p.gz
[root@portatil ~]# which cd
```

```
/usr/bin/which: no cd in (/usr/lib/qt-3.3/bin:/usr/kerberos/sbin:/usr/kerberos/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin)
[root@portatil ~]# help cd
cd: cd [-L|-P] [dir]
    Change the shell working directory.
    Change the current directory to DIR.  The default DIR is the value of the
    HOME shell variable.
    ...

bash-4.0$ help
GNU bash, version 4.0.38(1)-release (i386-redhat-linux-gnu)
These shell commands are defined internally.  Type `help' to see this list.
Type `help name' to find out more about the function `name'.
Use `info bash' to find out more about the shell in general.
Use `man -k' or `info' to find out more about commands not in this list.
A star (*) next to a name means that the command is disabled.
job_spec [&]                history [-c] [-d offset] [n] or history -anrw [filename] o>
...

bash-4.0$ man builtins
BASH_BUILTINS(1)                                BASH_BUILTINS(1)
NAME
    bash,  :,  .,  [,  alias, bg, bind, break, builtin, caller, cd, command, compgen, complete, compopt, con-
      tinue, declare, dirs, disown, echo, enable, eval, exec, exit, export, false, fc, fg, getopts, hash, help,
      history,  jobs, kill, let, local, logout, mapfile, popd, printf, pushd, pwd, read, readonly, return, set,
      shift, shopt, source, suspend, test, times, trap, true, type, typeset,  ulimit,  umask,  unalias,  unset,
      wait - bash built-in commands, see bash(1)
```

```
# activar desactivar el builtin o l'executable
bash-4.0$ enable -a
enable .
enable :
enable [
enable alias
enable bg
enable bind
enable break
enable builtin
enable caller
enable cd
...

bash-4.0$ enable -a | grep test
enable test
bash-4.0$ which test
/usr/bin/test
bash-4.0$ enable -n test
bash-4.0$ enable -a | grep test
```

```
enable -n test
```

Metacaràcters.

a) Metacaràcters: *, ?, [2-9], [abc], [!abc]

```
bash-4.0$ man bash   # buscar l'apartat de Pathname Expansion i file globbing

bash-4.0$ ls -l /usr/bin/ch*
-rwxr-xr-x 1 root root  12128 24 mar  2010 /usr/bin/chacl
-rwsr-xr-x 1 root root  60044 28 abr  2010 /usr/bin/chage
-rwxr-xr-x 1 root root   1271 24 jul  2009 /usr/bin/changetracker-console
lrwxrwxrwx 1 root root      9  6 gen  2010 /usr/bin/charmap -> gucharmap
-rwxr-xr-x 1 root root  10476  5 jul 16:35 /usr/bin/chattr
-rwxr-xr-x 1 root root  13280  6 abr  2010 /usr/bin/chcat
-rwxr-xr-x 1 root root  60364 28 abr  2010 /usr/bin/chcon
-rwxr-xr-x 1 root root  19531 15 feb  2010 /usr/bin/checkbashisms
-rwxr-xr-x 1 root root   3329 19 set  2006 /usr/bin/check-binary-files
-rwxr-xr-x 1 root root  17732 26 jul  2009 /usr/bin/checkisomd5
-rwxr-xr-x 1 root root 274672 24 jul  2009 /usr/bin/checkmodule
-rwxr-xr-x 1 root root 282896 24 jul  2009 /usr/bin/checkpolicy
-rwxr-xr-x 1 root root 180180 19 mar  2010 /usr/bin/cheese
-rws--x--x 1 root root  16168 12 abr  2010 /usr/bin/chfn
-rwxr-xr-x 1 root root     84 24 jun 17:11 /usr/bin/chktrust
-rwxr-xr-x 1 root root   9768 12 abr  2010 /usr/bin/chrt
-rws--x--x 1 root root  14984 12 abr  2010 /usr/bin/chsh
-rwxr-xr-x 1 root root   7216 26 ago  2009 /usr/bin/chvt

bash-4.0$ ls -l /usr/bin/cha*
-rwxr-xr-x 1 root root 12128 24 mar  2010 /usr/bin/chacl
-rwsr-xr-x 1 root root 60044 28 abr  2010 /usr/bin/chage
-rwxr-xr-x 1 root root  1271 24 jul  2009 /usr/bin/changetracker-console
lrwxrwxrwx 1 root root     9  6 gen  2010 /usr/bin/charmap -> gucharmap
-rwxr-xr-x 1 root root 10476  5 jul 16:35 /usr/bin/chattr

bash-4.0$ ls -l /usr/bin/ch[arv]*
-rwxr-xr-x 1 root root 12128 24 mar  2010 /usr/bin/chacl
-rwsr-xr-x 1 root root 60044 28 abr  2010 /usr/bin/chage
-rwxr-xr-x 1 root root  1271 24 jul  2009 /usr/bin/changetracker-console
lrwxrwxrwx 1 root root     9  6 gen  2010 /usr/bin/charmap -> gucharmap
-rwxr-xr-x 1 root root 10476  5 jul 16:35 /usr/bin/chattr
-rwxr-xr-x 1 root root  9768 12 abr  2010 /usr/bin/chrt
-rwxr-xr-x 1 root root  7216 26 ago  2009 /usr/bin/chvt

bash-4.0$ ls -l /usr/bin/ch[f-s]*
-rws--x--x 1 root root 16168 12 abr  2010 /usr/bin/chfn
-rwxr-xr-x 1 root root    84 24 jun 17:11 /usr/bin/chktrust
```

```
-rwxr-xr-x 1 root root  9768 12 abr  2010 /usr/bin/chrt
-rws--x--x 1 root root 14984 12 abr  2010 /usr/bin/chsh

bash-4.0$ ls -l /usr/bin/ch[!a-e]*
-rws--x--x 1 root root 16168 12 abr  2010 /usr/bin/chfn
-rwxr-xr-x 1 root root    84 24 jun 17:11 /usr/bin/chktrust
-rwxr-xr-x 1 root root  9768 12 abr  2010 /usr/bin/chrt
-rws--x--x 1 root root 14984 12 abr  2010 /usr/bin/chsh
-rwxr-xr-x 1 root root  7216 26 ago  2009 /usr/bin/chvt

bash-4.0$ ls -l /usr/bin/ch??
-rws--x--x 1 root root 16168 12 abr  2010 /usr/bin/chfn
-rwxr-xr-x 1 root root  9768 12 abr  2010 /usr/bin/chrt
-rws--x--x 1 root root 14984 12 abr  2010 /usr/bin/chsh
-rwxr-xr-x 1 root root  7216 26 ago  2009 /usr/bin/chvt

bash-4.0$ ls -l /usr/bin/ch?t
-rwxr-xr-x 1 root root 9768 12 abr  2010 /usr/bin/chrt
-rwxr-xr-x 1 root root 7216 26 ago  2009 /usr/bin/chvt

bash-4.0$ ls -l /usr/bin/ch??t
-rwxr-xr-x 1 root root 13280  6 abr  2010 /usr/bin/chcat

bash-4.0$ ls -l /usr/bin/ch*t
-rwxr-xr-x 1 root root 13280  6 abr  2010 /usr/bin/chcat
-rwxr-xr-x 1 root root    84 24 jun 17:11 /usr/bin/chktrust
-rwxr-xr-x 1 root root  9768 12 abr  2010 /usr/bin/chrt
-rwxr-xr-x 1 root root  7216 26 ago  2009 /usr/bin/chvt

bash-4.0$ ls -l /usr/bin/ch*fn
-rws--x--x 1 root root 16168 12 abr  2010 /usr/bin/chfn

bash-4.0$ ls -l /usr/bin/ch?fn
ls: no s'ha pogut accedir a /usr/bin/ch?fn: El fitxer o directori no existeix
```

Alias.

a) Llistar els alias definits: alias.
b) Provar-ne la seva execució.
c) Definir alies i provar-los. Alias hola='whoami'.
d) Desar permanentment un alies: .bashrc.

```
[root@portatil ~]# alias
alias cp='cp -i'
alias l.='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
```

```
alias ls='ls --color=auto'
alias mc='. /usr/libexec/mc/mc-wrapper.sh'
alias mv='mv -i'
alias rm='rm -i'
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot --show-tilde'

[pere@portatil ~]$ alias browser='nautilus --no-desktop --browser'
[pere@portatil ~]$ alias
alias l.='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
alias ls='ls --color=auto'
alias mc='. /usr/libexec/mc/mc-wrapper.sh'
alias vi='vim'
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot –show-tilde'
bash-4.0$ unalias browser

bash-4.0$ alias quisoc='whoami; who am i; id; finger $(whoami)'
bash-4.0$ quisoc
user1
ecanet   pts/2        2010-11-05 12:06 (:0.0)
uid=504(user1) gid=505(user1) grups=505(user1)
Login: user1                            Name:
Directory: /tmp/user1                   Shell: /bin/bash
Never logged in.
No mail.
No Plan.
bash-4.0$ alias
alias quisoc='whoami; who am i; id; finger $(whoami)'

bash-4.0$ alias psu='pstree user1'    // funcionara només per a user1
bash-4.0$ alias
alias psu='pstree user1'
bash-4.0$ psu
bash―――bash―――pstree

bash-4.0$ alias psu='pstree $(whoami)'    //funcionarà per a un user qualsevol
bash-4.0$ alias
alias psu='pstree $(whoami)'
bash-4.0$ psu
bash―――bash―――pstree

bash-4.0$ alias psu='pstree -a'
bash-4.0$ psu user1
bash
  └─bash
      └─pstree -a user1

bash-4.0$ unalias psu
```

// definicions per al pròpi usuari
bash-4.0$ **cat ~/.bashrc**
# .bashrc
# Source global definitions
if [-f /etc/bashrc]; then
        . /etc/bashrc
fi
User specific aliases and functions
alias browser='nautilus --no-desktop –browser' #alias fet per l'usuari

// les definicions per a tots els usuaris van a:
bash-4.0$ ls -l **/etc/bashrc**
-rw-r--r-- 1 root root 2615 24 mai 09:08 /etc/bashrc

Valor de retorn.

a) Codi d'estat o errorlevel o valor de retorn de tot programa.
b) Correcte: 0. Incorrecte: <> 0.
c) $? Variable que conté el codi d'estat de l´última ordre efectuada.

bash-4.0$ date
dv nov 5 18:18:25 EDT 2010
bash-4.0$ **echo $?**
0
bash-4.0$ date --opcionovalida
date: l’opció «--opcionovalida» no és reconeguda
Proveu «date --help» per a obtenir més informació.
bash-4.0$ **echo $?**
1

bash-4.0$ **ls llistat fit-inexistent**
ls: no s’ha pogut accedir a fit-inexistent: El fitxer o directori no existeix
llistat
bash-4.0$ **echo $?**
2

bash-4.0$ **mkdir /tmp/prova; echo $?**
0
bash-4.0$ **mkdir /tmp/prova; echo $?**
mkdir: no s’ha pogut crear el directori «/tmp/prova»: El fitxer ja existeix
1

EXIT STATUS
The exit status of an executed command is the value returned by the waitpid system call or equivalent function. Exit statuses fall between 0 and 255, though, as explained below, the shell may use values

above 125 specially. Exit statuses from shell builtins and compound commands are also limited to this range. Under certain circumstances, the shell will use special values to indicate specific failure modes.

For the shell's purposes, a command which exits with a zero exit status has succeeded. An exit status of zero indicates success. A non-zero exit status indicates failure. When a command terminates on a fatal signal N, bash uses the value of 128+N as the exit status.

If a command is not found, the child process created to execute it returns a status of 127. If a command is found but is not executable, the return status is 126.

If a command fails because of an error during expansion or redirection, the exit status is greater than zero.

Shell builtin commands return a status of 0 (true) if successful, and non-zero (false) if an error occurs while they execute. All builtins return an exit status of 2 to indicate incorrect usage.

Bash itself returns the exit status of the last command executed, unless a syntax error occurs, in which case it exits with a non-zero value. See also the exit builtin command below.

Command substitution.

a) Execució d'ordres dins d'altres ordres. `ordre`.
b) Assignar valors resultants d'ordres a variables. VAR=`cat /etc/passwd | wc -l`.

```
bash-4.0$ which ls
/bin/ls
bash-4.0$ file $(which ls)
/bin/ls: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.18, stripped
bash-4.0$ file `which ls`
/bin/ls: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.18, stripped

bash-4.0$ finger $(whoami)
Login: user1                            Name:
Directory: /tmp/user1                   Shell: /bin/bash
Never logged in.
No mail.
No Plan.

bash-4.0$ file $(ls)
llistat:       ASCII text
majuscules.txt: ASCII text
programa.sh:    Bourne-Again shell script text executable
sort:          ASCII text
sortida:       ASCII text

bash-4.0$ DATA="$(date)"
bash-4.0$ echo $DATA
dv nov 5 19:35:09 EDT 2010
bash-4.0$ echo $DATA
dv nov 5 19:35:09 EDT 2010
```

```
bash-4.0$ PROC=$(pstree user1)
bash-4.0$ echo $PROC
bash---bash---pstree

bash-4.0$ USUARIS=$(users)
bash-4.0$ echo $USUARIS
user1 pere marta root user1

bash-4.0$ echo $(date) $(whoami) $(id)
dv nov 5 19:40:43 EDT 2010 user1 uid=504(user1) gid=505(user1) grups=505(user1)
bash-4.0$ DADA="info:  $(date) $(whoami) $(id)"
bash-4.0$ echo $DADA
info: dv nov 5 19:40:59 EDT 2010 user1 uid=504(user1) gid=505(user1) grups=505(user1)
```

Interpretació de les comandes.

a) Expansió
b) Execució
c) si l'expansió falla l'ordre no s'executa. Per exemple si molts fitxers en fer rm * no els va esborrant de mica en mica.

```
bash-4.0$ echo ls
ls
bash-4.0$ echo $(ls)
llistat majuscules.txt programa.sh sort sortida

bash-4.0$ ls *
llistat  majuscules.txt  programa.sh  sort  sortida
bash-4.0$ echo *
llistat  majuscules.txt  programa.sh  sort  sortida

bash-4.0$ echo "arguments: *"
arguments: *
bash-4.0$ echo "arguments: " *
arguments:  llistat majuscules.txt programa.sh sort sortida


bash-4.0$ echo  dir{1..5}
dir1 dir2 dir3 dir4 dir5

bash-4.0$ echo  dir{1..50}
dir1 dir2 dir3 dir4 dir5 dir6 dir7 dir8 dir9 dir10 dir11 dir12 dir13 dir14 dir15 dir16 dir17 dir18 dir19
dir20 dir21 dir22 dir23 dir24 dir25 dir26 dir27 dir28 dir29 dir30 dir31 dir32 dir33 dir34 dir35 dir36
dir37 dir38 dir39 dir40 dir41 dir42 dir43 dir44 dir45 dir46 dir47 dir48 dir49 dir50
```

bash-4.0$ **echo dir{1..5000000000000000}**
echo error, massa arguments!

Expansion

Expansion:

a) Pathname expansion
b) Brace expansion.
c) Arimetic expansion.
d) Tilde expansion.
e) Command substitution.
f) Quoting.

Pathname Expansion.

Pathname expansion

Pathname Expansion
After word splitting, unless the -f option has been set, bash scans each word for the characters *, ?, and [. If one of these characters appears, then the word is regarded as a pattern, and replaced with an alphabetically sorted list of file names matching the pattern. If no matching file names are found, and the shell option nullglob is not enabled, the word is left unchanged. If the nullglob option is set, and no matches are found, the word is removed. If the failglob shell option is set, and no matches are found, an error message is printed and the command is not executed. If the shell option nocaseglob is enabled, the match is performed without regard to the case of alphabetic characters. When a pattern is used for pathname expansion, the character ‘.’ at the start of a name or immediately following a slash must be matched explicitly, unless the shell option dotglob is set. When matching a pathname, the slash character must always be matched explicitly. In other cases, the ‘.’ character is not treated specially. See the description of shopt below under SHELL BUILTIN COMMANDS for a description of the nocaseglob, nullglob, failglob, and dotglob shell options.

The GLOBIGNORE shell variable may be used to restrict the set of file names matching a pattern. If GLOBIGNORE is set, each matching file name that also matches one of the patterns in GLOBIGNORE is removed from the list of matches. The file names ‘.’ and ‘..’ are always ignored when GLOBIGNORE is set and not null. However, setting GLOBIGNORE to a non-null value has the effect of enabling the dotglob shell option, so all other file names beginning with a ‘.’ will match. To get the old behavior of ignoring file names beginning with a ‘.’, make ‘.*’ one of the patterns in GLOBIGNORE. The dotglob option is disabled when GLOBIGNORE is unset.

Pattern Matching

Any character that appears in a pattern, other than the special pattern characters described below, matches itself. The NUL character may not occur in a pattern. A backslash escapes the following character; the escaping backslash is discarded when matching. The special pattern characters must be quoted if they are to be matched literally.

The special pattern characters have the following meanings:
* Matches any string, including the null string. When the globstar shell option is enabled, and * is used in a filename expansion context, two adjacent *s used as a single pattern will match all files and zero or more directories and subdirectories. If followed by a /, two adjacent *s will match only directories and subdirectories.
? Matches any single character.
[...] Matches any one of the enclosed characters. A pair of characters separated by a hyphen denotes a range expression; any character that sorts between those two characters, inclusive, using the cur-

rent locale's collating sequence and character set, is matched. If the first character following the [is a ! or a ^ then any character not enclosed is matched. The sorting order of characters in range expressions is determined by the current locale and the value of the LC_COLLATE shell variable, if set. A - may be matched by including it as the first or last character in the set. A] may be matched by including it as the first character in the set.

Within [and], character classes can be specified using the syntax [:class:], where class is one of the following classes defined in the POSIX standard:
alnum alpha ascii blank cntrl digit graph lower print punct space upper word xdigit
A character class matches any character belonging to that class. The word character class matches letters, digits, and the character _.

Within [and], an equivalence class can be specified using the syntax [=c=], which matches all characters with the same collation weight (as defined by the current locale) as the character c.

Within [and], the syntax [.symbol.] matches the collating symbol symbol.

If the extglob shell option is enabled using the shopt builtin, several extended pattern matching operators are recognized. In the following description, a pattern-list is a list of one or more patterns separated by a |. Composite patterns may be formed using one or more of the following sub-patterns:

?(pattern-list)
Matches zero or one occurrence of the given patterns
*(pattern-list)
Matches zero or more occurrences of the given patterns
+(pattern-list)
Matches one or more occurrences of the given patterns
@(pattern-list)
Matches one of the given patterns
!(pattern-list)
Matches anything except one of the given patterns

* ? [...] [!...] [^...]
? * + @ !

Brace Expansion:

Brace expansion

Brace Expansion
Brace expansion is a mechanism by which arbitrary strings may be generated. This mechanism is similar to pathname expansion, but the filenames generated need not exist. Patterns to be brace expanded take the form of an optional preamble, followed by either a series of comma-separated strings or a sequence expression between a pair of braces, followed by an optional postscript. The preamble is prefixed to each string contained within the braces, and the postscript is then appended to each resulting string, expanding left to right.

Brace expansions may be nested. The results of each expanded string are not sorted; left to right order is preserved. For example, a{d,c,b}e expands into 'ade ace abe'.

A sequence expression takes the form {x..y[..incr]}, where x and y are either integers or single characters, and incr, an optional increment, is an integer. When integers are supplied, the expression expands to each number between x and y, inclusive. Supplied integers may be prefixed with 0 to force each term to have the same width. When either x or y begins with a zero, the shell attempts to force all generated terms to contain the same number of digits, zero-padding where necessary. When characters are supplied, the expression expands to each character lexicographically between x and y, inclusive. Note that both x and y must be of the same type. When the increment is supplied, it is used as the difference between each term. The default increment is 1 or -1 as appropriate.

Brace expansion is performed before any other expansions, and any characters special to other expansions are preserved in the result. It is strictly textual. Bash does not apply any syntactic interpretation to the context of the expansion or the text between the braces.

A correctly-formed brace expansion must contain unquoted opening and closing braces, and at least one unquoted comma or a valid sequence expression. Any incorrectly formed brace expansion is left unchanged.

A { or , may be quoted with a backslash to prevent its being considered part of a brace expression. To avoid conflicts with parameter expansion, the string ${ is not considered eligible for brace expansion. This construct is typically used as shorthand when the common prefix of the strings to be generated is longer than in the above example:

mkdir /usr/local/src/bash/{old,new,dist,bugs}
or
chown root /usr/{ucb/{ex,edit},lib/{ex?.?*,how_ex}}

Brace expansion introduces a slight incompatibility with historical versions of sh. sh does not treat opening or closing braces specially when they appear as part of a word, and preserves them in the output. Bash removes braces from words as a consequence of brace expansion. For example, a word entered to sh as file{1,2} appears identically in the output. The same word is output as file1 file2 after expansion by bash. If strict compatibility with sh is desired, start bash with the +B option or disable brace expansion with the +B option to the set command (see SHELL BUILTIN COMMANDS below).

[root@portatil ~]# **echo nom{1,2}**
nom1 nom2

[root@portatil ~]# **echo nom{1..5}**
nom1 nom2 nom3 nom4 nom5
[root@portatil ~]# echo nom{1-5}
nom{1-5}

[root@portatil ~]# **echo nom{a,m}**
noma nomm

[root@portatil ~]# **echo nom{a..m}**
noma nomb nomc nomd nome nomf nomg nomh nomi nomj nomk noml nomm

[root@portatil ~]# **echo nom{a..d,1,5,10..12}**
noma..d nom1 nom5 nom10..12
[root@portatil ~]# **echo nom{a..d} nom{1,5} nom{10..12}**
noma nomb nomc nomd nom1 nom5 nom10 nom11 nom12
bash-4.0$ **echo nom{{a..d},{1,5},{10..12}}**
noma nomb nomc nomd nom1 nom5 nom10 nom11 nom12

bash-4.0$ **echo nom{primer,segon,{1..7},quart}**
nomprimer nomsegon nom1 nom2 nom3 nom4 nom5 nom6 nom7 nomquart

bash-4.0$ **mkdir dir{1,2}**
bash-4.0$ ls
dir1 dir2 llistat majuscules.txt programa.sh sort sortida
bash-4.0$ rmdir dir{1,2}

Arithmetic Expansion:

 # Arithmetic expansion

Arithmetic Expansion
Arithmetic expansion allows the evaluation of an arithmetic expression and the substitution of the result. The format for arithmetic expansion is:

$((expression))
The expression is treated as if it were within double quotes, but a double quote inside the parentheses is not treated specially. All tokens in the expression undergo parameter expansion, string expansion, command substitution, and quote removal. Arithmetic expansions may be nested.

The evaluation is performed according to the rules listed below under ARITHMETIC EVALUATION. If expression is invalid, bash prints a message indicating failure and no substitution occurs.

bash-4.0$ $((2*2))
bash: 4: no s'ha trobat l'ordre

bash-4.0$ **echo $((2*2))**
4

bash-4.0$ echo $((2*2+$UID))
508

bash-4.0$ **NUM=$((2*$UID+300+$SHLVL))**
bash-4.0$ echo $NUM
1310
bash-4.0$ **echo $((1000/20))**
50

ARITHMETIC EVALUATION
The shell allows arithmetic expressions to be evaluated, under certain circumstances (see the let and declare builtin commands and Arithmetic Expansion). Evaluation is done in fixed-width integers with no check for overflow, though division by 0 is trapped and flagged as an error. The operators and their precedence, associativity, and values are the same as in the C language. The following list of operators is grouped into levels of equal-precedence operators. The levels are listed in order of decreasing precedence.

id++ id--
variable post-increment and post-decrement
++id --id
variable pre-increment and pre-decrement
- + unary minus and plus
! ~ logical and bitwise negation
** exponentiation
* / % multiplication, division, remainder
+ - addition, subtraction
<< >> left and right bitwise shifts
<= >= < >
comparison
== != equality and inequality
& bitwise AND
^ bitwise exclusive OR
| bitwise OR
&& logical AND
|| logical OR
expr?expr:expr
conditional operator
= *= /= %= += -= <<= >>= &= ^= |=
assignment
expr1 , expr2
comma
Shell variables are allowed as operands; parameter expansion is performed before the expression is evaluated. Within an expression, shell variables may also be referenced by name without using the parameter expansion syntax. A shell variable that is null or unset evaluates to 0 when referenced by name without using the parameter expansion syntax. The value of a variable is evaluated as an arithmetic expression when it is referenced, or when a variable which has been given the integer attribute using declare -i is assigned a value. A null value evaluates to 0. A shell variable need not have its integer attribute turned on to be used in an expression.

Constants with a leading 0 are interpreted as octal numbers. A leading 0x or 0X denotes hexadecimal. Otherwise, numbers take the form [base#]n, where base is a decimal number between 2 and 64 representing

the arithmetic base, and n is a number in that base. If base# is omitted, then base 10 is used. The digits greater than 9 are represented by the lowercase letters, the uppercase letters, @, and _, in that order. If base is less than or equal to 36, lowercase and uppercase letters may be used interchangeably to represent numbers between 10 and 35.

Operators are evaluated in order of precedence. Sub-expressions in parentheses are evaluated first and may override the precedence rules above.

Tilde Expansion:

Tilde expansion

Tilde Expansion
If a word begins with an unquoted tilde character (‘~’), all of the characters preceding the first unquoted slash (or all characters, if there is no unquoted slash) are considered a tilde-prefix. If none of the characters in the tilde-prefix are quoted, the characters in the tilde-prefix following the tilde are treated as a possible login name. If this login name is the null string, the tilde is replaced with the value of the shell parameter HOME. If HOME is unset, the home directory of the user executing the shell is substituted instead. Otherwise, the tilde-prefix is replaced with the home directory associated with the specified login name.

If the tilde-prefix is a ‘~+’, the value of the shell variable PWD replaces the tilde-prefix. If the tilde-prefix is a ‘~-’, the value of the shell variable OLDPWD, if it is set, is substituted. If the characters following the tilde in the tilde-prefix consist of a number N, optionally prefixed by a ‘+’ or a ‘-’, the tilde-prefix is replaced with the corresponding element from the directory stack, as it would be displayed by the dirs builtin invoked with the tilde-prefix as an argument. If the characters following the tilde in the tilde-prefix consist of a number without a leading ‘+’ or ‘-’, ‘+’ is assumed.

If the login name is invalid, or the tilde expansion fails, the word is unchanged.

Each variable assignment is checked for unquoted tilde-prefixes immediately following a : or the first =. In these cases, tilde expansion is also performed. Consequently, one may use file names with tildes in assignments to PATH, MAILPATH, and CDPATH, and the shell assigns the expanded value.

bash-4.0$ **echo ~**
/tmp/user1

bash-4.0$ **echo ~root**
/root

bash-4.0$ **echo ~pere**
/home/pere

[root]# **cp ~pere/.bash_profile ~marta/.bash_profile**

Command Expansion:

Command substitution

Command Substitution
Command substitution allows the output of a command to replace the command name. There are two forms:

$(command)

or
`command`

Bash performs the expansion by executing command and replacing the command substitution with the standard output of the command, with any trailing newlines deleted. Embedded newlines are not deleted, but they may be removed during word splitting. The command substitution $(cat file) can be replaced by the equivalent but faster $(< file).

When the old-style backquote form of substitution is used, backslash retains its literal meaning except when followed by $, `, or \. The first backquote not preceded by a backslash terminates the command substitution. When using the $(command) form, all characters between the parentheses make up the command; none are treated specially.

Command substitutions may be nested. To nest when using the backquoted form, escape the inner backquotes with backslashes.

If the substitution appears within double quotes, word splitting and pathname expansion are not performed on the results.

$(ordre)
`ordre`

bash-4.0$ **file $(which ls)**
/bin/ls: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.18, stripped

bash-4.0$ **file `which ls`**
/bin/ls: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.18, stripped

Quoting Expansion:

Quoting

QUOTING
Quoting is used to remove the special meaning of certain characters or words to the shell. Quoting can be used to disable special treatment for special characters, to prevent reserved words from being recognized as such, and to prevent parameter expansion.

Each of the metacharacters listed above under DEFINITIONS has special meaning to the shell and must be quoted if it is to represent itself.

When the command history expansion facilities are being used (see HISTORY EXPANSION below), the history expansion character, usually !, must be quoted to prevent history expansion.

There are three quoting mechanisms: the escape character, single quotes, and double quotes.

A non-quoted backslash (\) is the escape character. It preserves the literal value of the next character that follows, with the exception of <newline>. If a \<newline> pair appears, and the backslash is not itself quoted, the \<newline> is treated as a line continuation (that is, it is removed from the input stream and effectively ignored).

Enclosing characters in single quotes preserves the literal value of each character within the quotes. A single quote may not occur between single quotes, even when preceded by a backslash.

Enclosing characters in double quotes preserves the literal value of all characters within the quotes, with the exception of $, `, \, and, when history expansion is enabled, !. The characters $ and ` retain their special meaning within double quotes. The backslash retains its special meaning only when followed by one of the following characters: $, `, ", \, or <newline>. A double quote may be quoted within double

quotes by preceding it with a backslash. If enabled, history expansion will be performed unless an ! appearing in double quotes is escaped using a backslash. The backslash preceding the ! is not removed.

The special parameters * and @ have special meaning when in double quotes (see PARAMETERS below).

Words of the form $'string' are treated specially. The word expands to string, with backslash-escaped characters replaced as specified by the ANSI C standard. Backslash escape sequences, if present, are decoded as follows:

\a alert (bell)
\b backspace
\e an escape character
\f form feed
\n new line
\r carriage return
\t horizontal tab
\v vertical tab
\\ backslash
\' single quote
\nnn the eight-bit character whose value is the octal value nnn (one to three digits)
\xHH the eight-bit character whose value is the hexadecimal value HH (one or two hex digits)
\cx a control-x character

The expanded result is single-quoted, as if the dollar sign had not been present.

A double-quoted string preceded by a dollar sign ($) will cause the string to be translated according to the current locale. If the current locale is C or POSIX, the dollar sign is ignored. If the string is translated and replaced, the replacement is double-quoted.

bash-4.0$ **echo hola**
hola
bash-4.0$ **echo hola que tal**
hola que tal

bash-4.0$ **echo "hola que tal $USER"**
hola que tal user1

bash-4.0$ **echo 'hola que tal $USER'**
hola que tal $USER

bash-4.0$ **echo "hola que tal $USER *"**
hola que tal user1 *
bash-4.0$ e**cho "hola que tal $USER *" ***
hola que tal user1 * llistat majuscules.txt programa.sh sort sortida

bash-4.0$ **echo "hola que tal $USER \$USER"**
hola que tal user1 $USER
bash-4.0$ **echo "hola que tal $USER \$USER $HOME \home\user1"**
hola que tal user1 $USER /tmp/user1 \home\user1

bash-4.0$ **echo "el nom 'pere' es entre cometes"**
el nom 'pere' es entre cometes
bash-4.0$ **echo 'el nom "pere" es entre cometes'**
el nom "pere" es entre cometes
bash-4.0$ echo 'el nom 'pere' es entre cometes' #això és un error
el nom pere es entre cometes

bash-4.0$ **echo el nom \'pere\' es entre cometes**

el nom 'pere' es entre cometes

Quote Removal
After the preceding expansions, all unquoted occurrences of the characters \, ', and " that did not result from one of the above expansions are removed.

Ordres:

PATH (ordre aplicació), alias, unalias, whereis, which, enable, help, man builtin, .bashrc, .bash_profile, prompt, expressions (literals, cadenes, valors), valor de retorn $?, Command substitution.

Expansion: filename, brace, arithmetic, tilde, command, quoting.