Unidad 2: Interfaz de comandos y guiones (*scripts*)

Curso de Administración de Servidores GNU/LINUX
Centro de Formación Permanente
Universidad de Sevilla

Jorge Juan <jjchico@gmail.com>

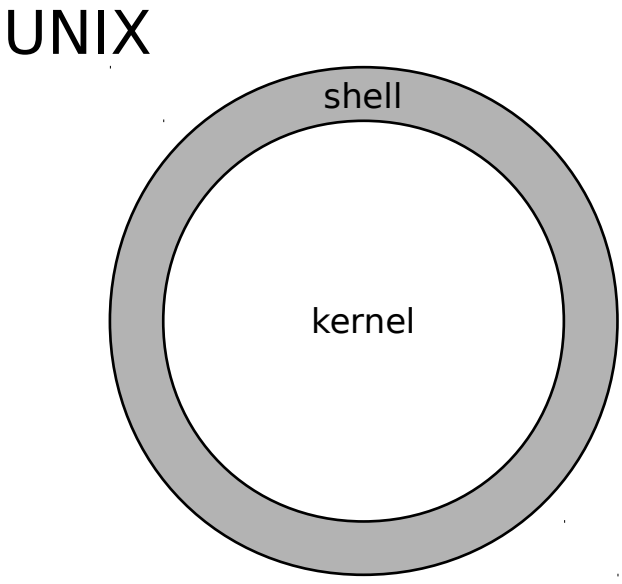
Contenidos

- **Generalidades**
- Redirección y tuberías
- Gestión de archivos
- Algunos comandos útiles
- Aplicaciones de terminal
- Scripts del shell

Terminales de texto

- Linux incluye una interfaz de comandos muy avanzada que facilita muchas tareas.
- Se puede usar el interfaz de comandos desde un terminal de consola o desde terminales virtuales
- Terminales de consola:
  - CTRL-ALT-FX
- En el terminal se ejecuta un intérprete de comandos
- El *prompt* o indica la disponibilidad para introducir un comando ($)

Shell: intérprete de comandos



Shell: intérprete de comandos

- Tipos de shells:
  - /bin/sh: shell original de UNIX
  - **/bin/bash**: Bourne again shell. (/bin/bash)
  - /bin/csh: C shell
  - Otros: tcsh, ksh, ...
- Interpreta los comandos y ejecuta, devolviendo el resultado

Formato de la línea de comandos

- Comando: archivo ejecutable
  - /bin, /usr/bin, /usr/local/bin, /sbin, /usr/sbin, /usr/local/sbin
- Formato:
  - <comando> [OPCIONES] [ARGUMENTOS]
- [OPCIONES] pueden ser:
  - Letras precedidas del símbolo '-'
  - Palabras precedidas de los símbolos '--'
- [ARGUMENTOS]: nombres de archivos o cualquier otro dato a pasar al programa.

```
$ ls -l /home
$ ls --human-readable /home
```

Caracteres especiales

- Separadores
  - Espacio: separa parámetros
  - Enter: ejecuta comandos
- Comodines: expansión de nombres de archivos
  - *: representa cualquier cadena de caracteres.
  - ?: representa cualquier carácter simple.
  - [<rango>]: cualquier carácter en <rango>.
- Escape:
  - \, ", '
- Otros
  - #, <, >, (,), {, }, $, &, [[,]]

Caracteres especiales

```
$ ls
cap1 cap2 cap3 suma.c resta.c

$ ls c*
cap1 cap2 cap3

$ ls cap?
cap1 cap2 cap3

$ ls *.c
suma.c resta.c

$ cp "Mis Documentos" /tmp
$ cp 'Mis Documentos' /tmp
$ cp Mis\ Documentos /tmp
```

Secuencias de control del terminal

- Permiten enviar señales los programas ejecutados bajo el control del terminal
  - Ctrl-C: Terminar
  - Ctrl-Z: Suspender
  - Ctrl-D: Fin de archivo
  - Ctrl-V: próximo carácter es literal (escapar sec. de control)

```
$ find / carta.txt
<Ctrl-C>

$ cat
hola
hola
<Ctrl-D>
```

Comandos de ayuda

- man: páginas de manual. Información detallada sobre cada comando y sus opciones
  - 'q' para salir.
- info: es similar a 'man', más didáctico.
- whatis: breve descripción de cada comando.
- apropos <término>: comandos relacionados con <término>
- <comando> --help: ayuda básica integrada en muchos comandos

```
$ ls --help
...
$ man ls
...
$ whatis passwd
```

Localización de comandos

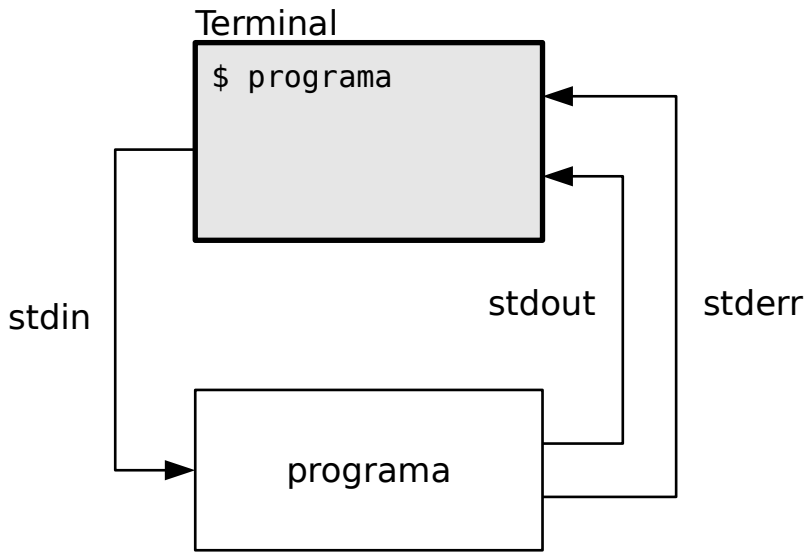
- Comandos de generales
  - /bin, /usr/bin, /usr/local/bin
- Comandos de administración
  - /sbin, /usr/sbin, /usr/local/sbin
- Comandos del usuario
  - ~/bin (si existe)
- which: localiza un comando
- whereis: localiza un comando, página de manual, datos, ...

```
$ which ls
/bin/ls

$ ls /*bin /usr/*bin | wc -l
3294
```

Contenidos

- Generalidades
- **Redirección y tuberías**
- Gestión de archivos
- Algunos comandos útiles
- Aplicaciones de terminal
- Scripts del shell

Redirección
Entrada/salida estándar




```
$ ls dir.txt list.txt
ls: no se puede acceder a list.txt: No existe el fichero ó directorio
dir.txt
```

Redirección de la salida estándar a un archivo

- >: redirección a un archivo nuevo o borra el anterior
- >>: redirección a un archivo nuevo o añade al anterior
- 2>, 2>>: igual, pero para la salida de error
- &>, &>>: igual pero para ambas salidas

- cat: lee un archivo o la entrada estándar y copia a la salida estándar
- less (more): lee texto plano de un archivo o la entrada estándar muestra el contenido en el terminal por páginas
  - desplazamiento: cursores, AvPág, RePág, h, j, k, l.
  - salir: q
  - ayuda: h

Redirección de la salida estándar a un archivo

```
$ ls /home > dir.txt
$ cat dir.txt
ana
jorge

$ ls /etc > etc.txt
$ less etc.txt

$ ls dir.txt list.txt 2> error.txt
dir.txt
$ cat error.txt
ls: no se puede acceder a list.txt: No existe el fichero ó directorio

$ ls dir.txt list.txt > out.txt 2> error.txt
$ cat out.txt
dir.txt
$ cat error.txt
ls: no se puede acceder a list.txt: No existe el fichero ó directorio
```

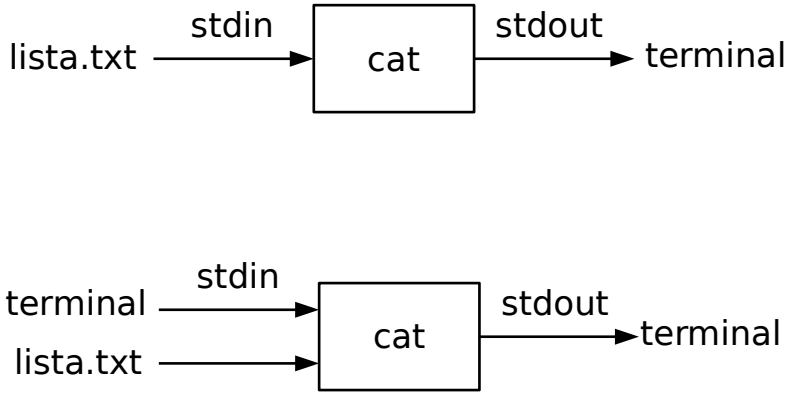
Redirección de la salida estándar a un archivo

```
$ cat > lista.txt
leche
galletas
zumo
pañales
Ctrl-D
$ cat lista.txt
leche
galletas
zumo
pañales

$ cat >> lista.txt
chocolate
Ctrl-D
$ cat lista.txt
leche
galletas
zumo
pañales
chocolate
```

Redirección de la entrada estándar desde un archivo

- <: redirección desde un archivo

```
$ cat < lista.txt
leche
galletas
zumo
pañales
chocolate

$ cat lista.txt
leche
galletas
zumo
pañales
chocolate
```



Archivos especiales /dev/null y /dev/zero

- /dev/null
  - escritura: elimina lo que se escribe
  - lectura: proporciona “fin de archivo”
- /dev/zero
  - escritura: elimina lo que se escribe
  - lectura: proporciona infinitos “ceros”

Archivos especiales /dev/null y /dev/zero

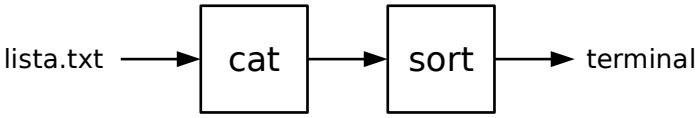
```
# Descarta la salida de un comando
$ ls /etc/fstab > /dev/null

# También la salida de error
$ ls /etc/fstab &> /dev/null

# Crea un archivo vacío o elimina su contenidos
$ cat /dev/null > dir.txt

# Mantiene ocupado al sistema
$ cat /dev/zero > /dev/null
Ctrl-C
```

Tuberías (pipes)

- Conecta la salida estándar de un comando con la entrada estándar de otro
- Permite concatenar comandos sin crear archivos intermedios
- Una de las claves de UNIX
- Especialmente útil con filtros
  - sort: ordena archivos de texto plano
  - wc: cuenta líneas, palabras y caracteres
  - grep: muestra líneas que concuerdan con un patrón
  - ...

Tuberías

```
$ cat lista.txt
leche
galletas
zumo
pañales
chocolate

$ cat lista.txt | sort
chocolate
galletas
leche
pañales
zumo

$ cat lista.txt | sort -r | less
...
```



Contenidos

- Generalidades
- Redirección y tuberías
- **Gestión de archivos**
- Algunos comandos útiles
- Aplicaciones de terminal
- Scripts del shell

Gestión de archivos

- Carpeta de trabajo
  - pwd
- Ruta absoluta: respecto a la carpeta raiz
- Ruta relativa: respecto a la carpeta de trabajo
- Abreviaturas:
  - '.' carpeta actual
  - '..' carpeta superior
  - '~' carpeta del usuario
  - '~user' carpeta del usuario 'user'

Gestión de archivos

```
$ pwd
/home/usuario
$ ls /etc/fstab
/etc/fstab
$ cd /etc
$ ls fstab
fstab
$ cd
$ ls ../../etc/fstab
../../etc/fstab
$ ls fstab
ls: no se puede acceder a fstab: No existe el fichero ó directorio
```

Gestión de archivos
Comandos básicos

- Para listar los ficheros contenidos en una carpeta (directorio) se usa el comando 'ls' (list).
- Si no se le proporciona argumentos muestra contenido de la carpeta actual.
- La opción -a lista también los archivos ocultos.
- La opción -l ofrece información adicional.

```
$ ls -la
...
```

Gestión de archivos
Comandos básicos

- El comando 'cd' (*change directory*) permite cambiar la carpeta actual.

```
$ cd carpeta_destino

$ cd .

$ cd ..
```

Gestión de archivos
Comandos básicos

- El comando 'cp' (*copy*) permite copiar archivos y carpetas.
- El comando 'mv' (*move*) permite mover archivos y carpetas.

```
$ cp archivo1 archivo2
$ cp -r carpeta1 carpeta2
$ mv archivo1 archivo2
$ mv archivo1 carpeta1
```

Gestión de archivos
Comandos básicos

- 'rm' (*remove*) borra archivos y carpetas.
- 'rmdir' borra carpetas.
- 'mkdir' (*make directory*) crea subcarpetas.

```
$ rm archivo

$ rm -r carpeta

$ rmdir carpeta

$ mkdir subcarcarpeta
```

Gestión de archivos
Otros comandos

- find, locate, diff, cmp, rename, ...

```
$ find . -name '*.txt' -a -newer lista.txt
...

$ locate '*.wav' | less
...

$ diff lista1.txt lista2.txt
...

$ cmp lista1.txt lista2.txt
...

$ rename 's/JPEG$/jpg/' *.JPEG
...
```

Contenidos

- Generalidades
- Redirección y tuberías
- Gestión de archivos
- **Algunos comandos útiles**
- Aplicaciones de terminal
- Scripts del shell

Filtros

- sort
- grep
- tr
- uniq
- head / tail
- sed (avanzado)
- awk (avanzado)

```
$ wget http://www.gnu.org/licenses/gpl.txt
...
$ cat gpl.txt |
tr -sc A-Za-z '\012' |
sort |
uniq -c |
sort -nr |
head
    309 the
    210 of
    177 to
    171 a
    138 or
    106 you
     97 work
     91 that
     91 and
     76 in
```

Filtros
Expresiones regulares

- Especifican patrones para localizar texto en archivos.
- Usadas por filtros como grep, sed, awk, ...
- Expresiones
  - '.' cualquier carácter
  - '[...], [^...]' uno de varios caracteres o rangos
  - '*' cero o más caracteres iguales al anterior
  - '+' uno o más caracteres iguales al anteior
  - '^' principio de la línea
  - '$' fin de la línea
  - '\' operación especial
  - '\(...\)' establece marca
  - '\n' referencia a n-ésima marca anterior

Filtros
Expresiones regulares

```
$ wget http://www.gnu.org/licenses/gpl.txt
...
$ grep '[Ss]oftware' gpl.txt | wc -l
26

$ grep '^\([aeiou]\).*\1$' /usr/share/dict/spanish
aarónica
...
ozonómetro

$ grep '^\([^aeiou]\).*\1$' /usr/share/dict/spanish
baobab
...
zarevitz

$ grep '^t.t.r.$' /usr/share/dict/spanish
tataré
...
tutora
```

Filtros
Expresiones regulares

```
$ grep '^/dev/md' /etc/fstab | tr -s ' ' ' ' | cut -d' ' -f2
/
/boot
/home
none


$ awk '/^\/dev\/md/ {print $2}' /etc/fstab
/
/boot
/home
none
```

Comandos útiles Compresión

- Compresión de archivos
  - gzip: comprime un archivo (archivo resultante con extensión .gz) y borra el fichero original no comprimido.

```
$ gzip lista.txt

$ gzip -l lista.txt.gz
```

  - gunzip: descomprime el archivo o archivos

```
$ gunzip lista.txt.gz
```

Comandos útiles Compresión

- Compresión de archivos
  - tar (tape archive): empaqueta un conjunto de archivos manteniendo la información de propiedades y permisos de cada uno.
  - Con la opción “z” además comprime con gzip

```
$ tar zcvf documentos.tar.gz docs
...
```

(c: crea un nuevo archivo)

```
$ tar zxvf documentos.tar.gz
...
```

(x: extraer archivos de un paquete)

Comandos útiles
Conexión remota

- ssh
  - Administración remota
  - Usar programas de otro ordenador
- telnet, nc
  - Conexión a puertos de otros equipos
- scp, rsync, sftp, ftp, wget, ...
  - Transferencia de archivos

Comandos útiles
Control de procesos

- Comando para el control de procesos
  - ps: listar procesos
  - nice: cambiar prioridad
  - kill/killall: enviar señales
- Control de trabajos
  - Ctrl-Z: (desde el terminal) detiene trabajo en curso
  - bg/fg: enviar a segudo/primer plano
  - '&': (al final del comando) ejecuta en segundo plano
  - jobs: lista de trabajos del terminal

Comandos útiles. Varios

- date
- cal
- bc -l
- w
- who
- uptime
- ...

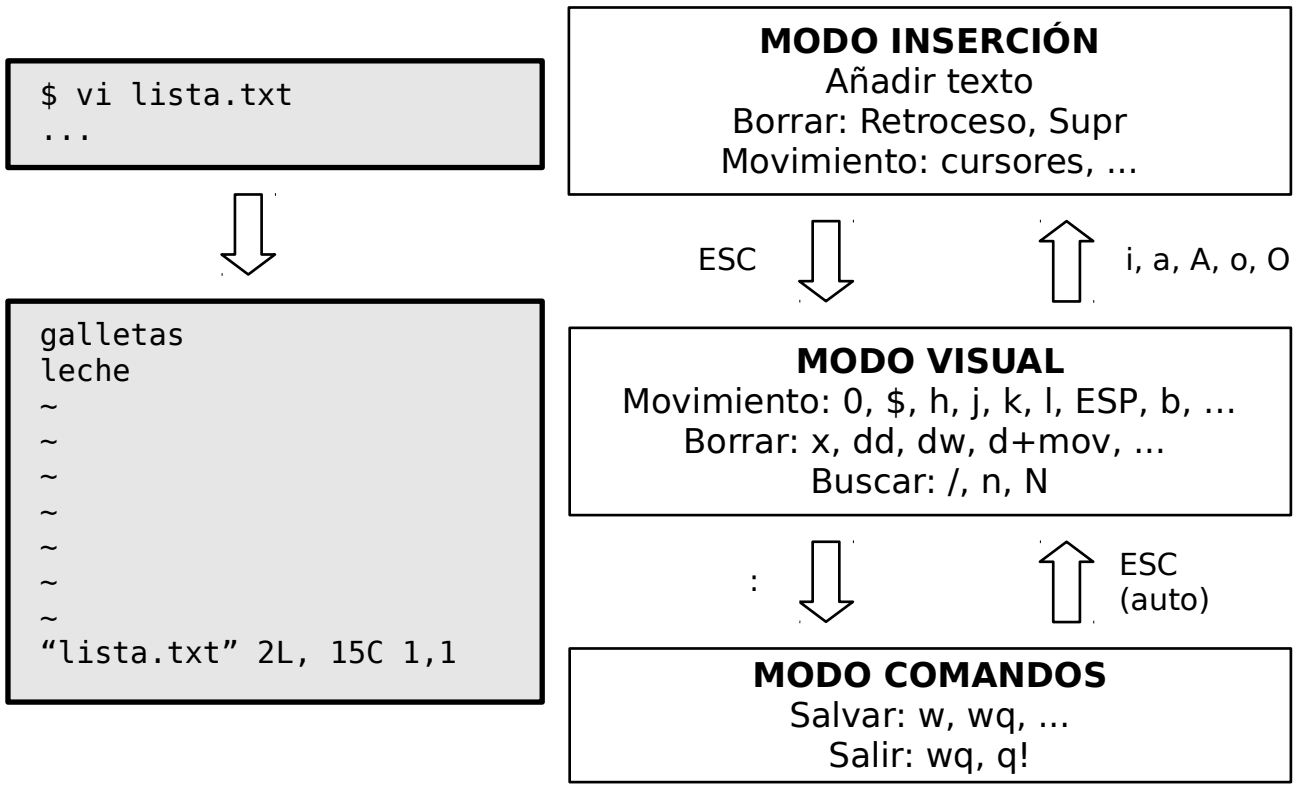
Contenidos

- Generalidades
- Redirección y tuberías
- Gestión de archivos
- Algunos comandos útiles
- **Aplicaciones de terminal**
- Scripts del shell

Aplicaciones de terminal

- Editores de texto
  - Editor 'nano':
    - Intuitivo y fácil de utilizar
  - Editor 'vi':
    - Uso muy extendido en sistema UNIX/Linux.
    - Tiene varios modos: modo comando y modo de inserción y modo de línea.
    - Si no se conoce es difícil de utilizar.
    - Potente y rápido

Vi esencial



Aplicaciones de terminal

- top, htop
  - Control de procesos e información del sistema
- lynx
  - Navegador web
- mutt
  - Cliente de correo electrónico
- ...

Contenidos

- Generalidades
- Redirección y tuberías
- Gestión de archivos
- Algunos comandos útiles
- Aplicaciones de terminal
- **Scripts del shell**

Scripts del shell

- bash es un shell programable
- Muy útil para automatizar tareas y hacer pequeños programas
- Potente pero lento: no apto como lenguaje de programación general
- Desarrollo muy rápido de pequeñas aplicaciones
- Referencias
  - Advanced Bash-Scripting Guide
    - http://tldp.org/LDP/abs (¡Muy recomendable!)
  - Bash Reference Manual
    - http://www.gnu.org/software/bash/manual/bashref.html

Aspectos básicos. Variables

- Variables del shell
  - Permiten almacenar valores, cadenas, etc.
  - Asignación: <variable>=<valor> (sin espacios)
  - Uso: $variable
- Variables especiales
  - $HOME, $PWD, $PATH, $?, ...

```
$ item=galletas
$ echo $item
galletas
$ grep $item lista.txt
galletas
```

Aspectos básicos. Variables

- Las variables en general son locales al intérprete y no son heredadas por los procesos hijos.
- Las variables de entorno sí son heredadas por los procesos hijos.
- Las variables se “pasan” al entorno con “export”

```
$ item=galletas
$ export item
. . .
$ export search=donuts
```

Aspectos básicos. Valor de retorno

- Cada programa devuelve un valor de retorno que se almacena en la variable ($?)
  - 0: ejecución correcta
  - !=0: algún error

```
$ cat lista.txt
galletas
leche
$ grep galletas lista.txt
galletas
$ echo $?
0
$ grep tarta lista.txt
$ echo $?
1
```

Aspectos básicos. Configuración de bash

- Varios archivos del sistema y del usuario configuran bash
  - ~/bashrc
  - ~/bash_alias
  - ~/.profile
  - /etc/bash.bashrc
  - /etc/profile

Guiones (scripts)

- Archivo de texto con lista de comandos y estructuras de control que puede ejecutar un intérprete
- El archivo debe tener permiso de ejecución
- La primera línea permite seleccionar el intérprete (#!)
- En bash, # indica un comentario.
- Un script se comporta como cualquier otro progrma.
- Es conveniente poner los scripts personales en ~/bin

Guiones (scripts)

```
#!/bin/bash
# numrep-v1.sh: cuenta repeticiones

tr -sc A-Za-z '\012' |
sort |
uniq -c |
sort -nr |
head
```

```
$ cat gpl.txt | numrep-v1.sh
    309 the
    210 of
    177 to
...
```

¿Qué ocurre si ejecutamos numrep-v1.sh a secas?

Paso de parámetros

- $1, $2, $3, ...: parámetros en la línea de comandos
- $*: todos los parámetros
- "$@" = "$1" "$2" "$3" ...
- $#: número de parámetros
- shift: desplaza los parámetros

```
#!/bin/bash
# numrep-v2.sh: cuenta repeticiones

cat "$1" |
tr -sc A-Za-z '\012' |
sort |
uniq -c |
sort -nr |
head
```

¿Qué ocurre si no pasamos ningún parámetro?

Condiciones

- Estructura if-then-fi
- Evaluación de condiciones
  - test (comando empotrado)
  - [...] (igual que 'test')
  - [[...]] (palabra clave, más rápido, menos portable)
- Expresiones condicionales
  - <expresion 1> || <expresion 2>
  - <expresion 1> && <expresion 2>

Condiciones

```
#!/bin/bash
# numrep-v3.sh: cuenta repeticiones

if [ $# -lt 1 ]
# if test $# -lt 1
# if [[ $# -lt 1 ]]
then
  echo "Necesito al menos un parámetro."
  exit 1
fi

cat "$@" |
tr -sc A-Za-z '\012' |
sort |
uniq -c |
sort -nr |
head
```

Condiciones

```
#!/bin/bash
# numrep-v4.sh: cuenta repeticiones

[ $# -lt 1 ] || {
  echo "Necesito al menos un parámetro."
  exit 1
}

cat "$@" |
tr -sc A-Za-z '\012' |
sort |
uniq -c |
sort -nr |
head
```

Otra forma de poner la condición, pero esta vez hemos cometido un pequeño error. Corrígelo.

Condiciones

```
#!/bin/bash
# numrep-v5.sh: cuenta repeticiones

if [ "X$*" == "X" ]; then
  cat
else
  cat "$@"
fi |
tr -sc A-Za-z '\012' |
sort |
uniq -c |
sort -nr |
head
```

Esta versión es mejor. ¿Por qué?

Condiciones if-grep

- grep devuelve '0' en caso de que encuentre el patrón indicado y '1' en otro caso, lo cual tiene mucha utilidad en los scripts.
- (-q suprime la salida de grep)

```
#!/bin/bash
# softraid-v1.sh
# Comprueba uso de RAID software

if grep -q '^/dev/md' /etc/fstab
then
  echo "Su sistema usa RAID por software"
  exit 0
else
  echo "Su sistema NO usa RAID por software"
  exit 1
fi
```

Funciones

```
#!/bin/bash
# softraid-v2.sh
# Comprueba uso de RAID software mediante función

use_software_raid ()
{
  if grep -q '^/dev/md' /etc/fstab; then
    return 0
  else
    return 1
  fi
}

if use_software_raid; then
  [ "X$1" != "X-q" ] && echo "Su sistema usa RAID por
software"
  exit 0
else
  [ "X$1" != "X-q" ] && echo "Su sistema NO usa RAID
por software"
  exit 1
fi
```

Bucles for y while

- Dos bucles equivalentes

```
#!/bin/bash
# Demostración de bucles for y while

echo "Bucle for"
for i in "$@"
do
  echo "$i"
done

echo "Bucle while"

while [[ $# -ne 0 ]]
do
  echo "$1"
  shift
done
```

Expansión aritmética

- ((<expresión>))
  - evalúa el contenido de forma aritmética
- $((<expresión>))
  - evalúa y sustituye el resultado
- let "<expresión>"
  - evalúa de forma aritmética
  - es un comando empotrado

```
#!/bin/bash
# Demostración de expansión aritmética

echo "Bucle for"
n=1
for i in "$@"
do
  echo "Parámetro $n: $i"
  ((n = n + 1))    # n=$((n + 1))
done

echo "Bucle while"
n=1
while [[ $# -ne 0 ]]
  do
    echo "Parámetro $n: $1"
    shift
    let "n = n + 1"  # let n=n+1
done
```

Sustitución de comandos

- `comando`
- $(comando)

```
#!/bin/bash
# Sustitución de comandos (V1)

list=`find . -name '*.txt'`

for i in $list
do
  size=`ls -s "$i" | cut -d' ' -f1`
  echo "El tamaño de $i es $size"
done
```

¿Qué ocurre si 'find' encuentra archivos con espacios en su nombre?

Sustitución de comandos

```
#!/bin/bash
# Sustitución de comandos (V2)

find . -name '*.txt' | while read i
do
  size=`ls -s "$i" | cut -d' ' -f1`
  echo "El tamaño de $i es $size"
done
```

Esta versión es mejor porque se ejecuta más rápido, consume menos memoria y funciona bien con cualquier nombre de archivo.

Case

```
#!/bin/bash
# numrep-v6.sh: cuenta repeticiones

prog=`basename $0`; lines=10

while [[ "$1" =~ ^- ]]; do
  case "$1" in
    -h) echo "$prog: cuenta repeticiones de
   palabras"
        echo "Uso: $prog [-h] [-n N]
   <archivo>"
        exit 1
        ;;
    -n) lines=$2; shift
        ;;
    -*) echo "$prog: opción desconocida."
        echo "Pruebe: '$prog -h' para ayuda."
        exit 1
        ;;
  esac
  shift
done
```

```
if [ "X$*" == "X" ]
then
  cat
else
  cat "$@"
fi |
tr -sc A-Za-z '\012' |
sort |
uniq -c |
sort -nr |
head -$lines
```

Comandos útiles para scripts

- ¡Todos!
- Filtros
- basename
- dirname
- ...

Consejos útiles

- Claridad
- Comentarios: los precisos, breves.
- Cuidado con los espacios en los nombres de archivos
- Incluir comprobaciones. Pensar en el peor caso
  - ¿y si no existe el archivo?
  - ¿y si no se puede escribir en el archivo?
  - ¿y si el comando devuelve un error?
  - ¿y si faltan parámetros?
  - ¿y si el número no es un número?
  - ...