

Sistemas Operativos



Introducción
Introducción al Shell



1 ¿Qué es un intérprete de comandos?

2 Comandos y operadores del Bash

3 Páginas de manual

4 Redirecciones

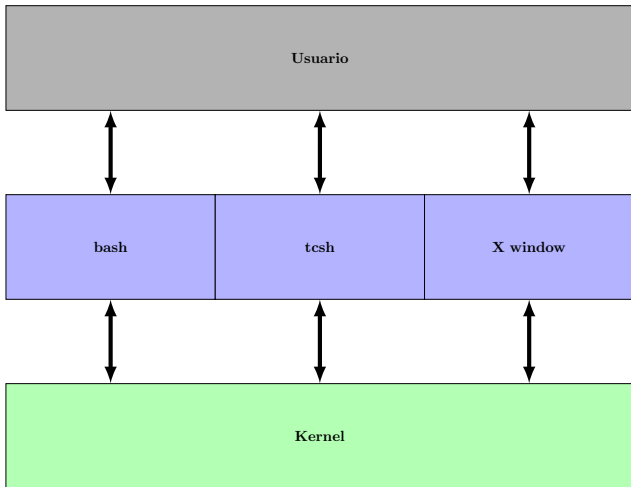
5 Guiones Shell

6 Expresiones regulares

Shell



- Programa que actúa como interfaz entre el usuario y el SO





Intérprete de comandos Bash

- Bash (*Bourne-again shell*) es una *shell* Unix escrita por Brian Fox para el proyecto GNU como una alternativa libre a la *shell Bourne*
- Bash es el intérprete predeterminado en muchos sistemas UNIX, la mayoría de sistemas GNU/Linux, Solaris y Mac OS-X
- También se ha portado a Microsoft Windows
- Otros intérpretes:
 - sh: Es el *shell Bourne*
 - tcsh o TENEX C shell: derivado de csh, es un *shell C*
 - ksh o Korn shell: en ocasiones usado en UNIX

Ejecución Bash I



- Cuando un *shell* interactivo que no es un *login shell* arranca, Bash lee y ejecuta órdenes desde `~/.bashrc`, si existiese.
- Dispone de prefijos o *prompts* (PS1 y PS2).
- Los mandatos se leen en línea (*readline*) y se ejecutan tras su lectura.
- La historia de mandatos se guarda en fichero (HISTFILE) y es posible realizar búsquedas en el historial (CTRL+R)

Ejecución Bash II



- Se permite la expansión de alias.
 - Ejemplo: `alias ls='ls --color'`
- Se pueden modificar los manejadores de señal
 - Bash maneja SIGINT (Ctrl+C)
 - La orden `trap` puede usarse para especificar comandos a ejecutar cuando se recibe alguna señal
 - Uso frecuente: `nohub`
- Se puede controlar la acción a tomar cuando el interprete de comandos recibe un carácter EOF (Ctrl+D)
 - `export IGNOREEOF=2` (ignora EOF 2 veces)



1 ¿Qué es un intérprete de comandos?

2 Comandos y operadores del Bash

3 Páginas de manual

4 Redirecciones

5 Guiones Shell

6 Expresiones regulares

Ejecutando comandos



- Tipos de comandos
 - Comandos internos (*built-in commands*): Forman parte del repertorio del propio *shell*
 - Comandos externos: Programas externos al *shell* instalados en el sistema (ficheros binarios ejecutables o *scripts*)
- Las secuencias de comandos pueden incluirse en un fichero denominado guión o *Script Bash*
 - Cuando el programa es un guión, *Bash* creará un nuevo proceso usando `fork()`

Comandos propios



- Bourne Shell built-ins ...
 - `:`, `.`, `break`, `cd`, `continue`, `eval`, `exec`,
`exit`, `export`, `getopts`, `hash`, `pwd`, `readonly`,
`return`, `set`, `shift`, `test`, `[`, `times`, `trap`,
`umask`, `unset`
- + Bash built-in commands:
 - `alias`, `bind`, `builtin`, `command`, `declare`,
`echo`, `enable`, `help`, `let`, `local`, `logout`,
`printf`, `read`, `shopt`, `type`, `typeset`, `ulimit`,
`unalias`

Comandos básicos



| Comando | Descripción |
|----------------------|---|
| ls | Lista los ficheros del directorio actual |
| pwd | Muestra en qué directorio nos encontramos |
| cd directory | Cambia de directorio |
| mv files dest | Mueve/Cambia la ruta de los ficheros |
| diff file1 file2 | Muestra las diferencias de dos ficheros |
| patch file patchfile | Aplica un parche (diff) a un fichero (ver op. -p) |
| cp files dest | Copia ficheros a una nueva ruta |
| tail file | Muestra las últimas líneas de un fichero |
| head file | Muestra las primeras líneas de un fichero |
| mkdir dirname | Crea un nuevo directorio |
| rm files | Borra ficheros (elimina su nombre, unlink) |
| file filename | Muestra el tipo de fichero dado |

Comandos básicos



| Comando | Descripción |
|----------------|--|
| cat text_file | Muestra el contenido del archivo en pantalla |
| man command | Muestra la página de manual para el comando dado |
| apropos string | Busca la cadena en la base de datos whatis |
| exit/logout | Abandona la sesión |
| grep | Busca en archivos líneas que contengan un patrón |
| echo | Muestra una línea de texto |
| env | Guarda información en el entorno |
| export | Cambia el valor de una variable de entorno |



Variables y operadores

■ Variables:

- `a=5` *#asignación*
- `echo $a` *#expansión*
- `b=$(($a+3))` *#aritmética entera*
- `b=$(($a<<1))` *#operadores de bits*

■ Operadores aritméticos y de bits:

- `+` `-` `/` `*` `%` `&` `|` `^` `<<` `>>`



Listas de órdenes

- Variable \$?
 - status de la última orden ejecutada
- `orden1 ; orden2`
 - `orden2` se ejecuta cuando acaba `orden1`
 - `$?` es el status de `orden2`
- `orden1 && orden2`
 - `orden2` sólo se ejecuta si status de `orden1 == 0` (éxito)
- `orden1 || orden2`
 - `orden2` sólo se ejecuta si status de `orden1 != 0` (fallo)

Ejecución en primer y segundo plano



- *foreground y background*

- En modo interactivo los procesos se ejecutan en primer plano (*foreground*): el *shell* no muestra el *prompt* hasta que no finaliza la ejecución de la última orden introducida.
- Si queremos dejar el proceso en segundo plano (*background*) se añade &:

```
$ xeyes &  
[2] 7584      -> [job_id] PID
```



1 ¿Qué es un intérprete de comandos?

2 Comandos y operadores del Bash

3 Páginas de manual

4 Redirecciones

5 Guiones Shell

6 Expresiones regulares

Páginas de manual en UNIX



- `man` da un formato y muestra las páginas de manual
 - Pasa a la siguiente página usando la barra espaciadora
 - Vuelve a la página anterior pulsando la tecla "b"
 - Para salir se pulsa la tecla "q"
- Las páginas de manual están divididas en secciones:
 - 1 Comandos generales
 - 2 Llamadas al sistema
 - 3 Funciones en bibliotecas, especialmente la biblioteca estándar de C
 - 4 Ficheros especiales (dispositivos ubicados en `/dev`) y controladores
 - 5 Formatos de archivo y convenciones
 - 6 Juegos y protectores de pantalla
 - 7 Miscelánea
 - 8 Comandos de administración del sistema y demonios
 - 9 API del kernel (módulos y core kernel)



Uso del programa “man”

- Manual de un comando/función:

```
man [1–9] command
```

- Buscar en descriptores y nombres de páginas de manual para la clave “keyword”:

```
man -k keyword
```

- Introducción acerca de una sección

```
man [1–9] intro
```

- Llamadas al sistema Linux:

```
man 2 syscalls
```

- Buscar texto en la página de manual:

```
/caracteres (siguiente: 'n', anterior: 'N')
```

Subsecciones del manual



- **NAME** : La primera línea contiene el nombre del comando
- **SYNOPSIS**: Notación técnica con todas las opciones y/o argumentos que este comando puede aceptar
- **DESCRIPTION**: Descripción extensa del comando
- **OPTIONS**: Opciones y su descripción
- **ENVIRONMENT**: Las variables shell que afectan al comportamiento de este comando
- **EXIT STATUS**
- **BUGS, SEE ALSO, NOTES, EXAMPLES**, etc.



- 1 ¿Qué es un intérprete de comandos?
- 2 Comandos y operadores del Bash
- 3 Páginas de manual
- 4 **Redirecciones**
- 5 Guiones Shell
- 6 Expresiones regulares



Redirecciones

- Tres descriptores de ficheros predeterminados:

`stdin (0) stdout (1) stderr (2)`

- Redirección de la salida estándar:

`orden > fichero`

- Redirección de la salida de error:

`orden 2> fichero`

`orden > fichero 2>&1`

- Redirección de la entrada estándar:

`orden < fichero`

Ejemplos I



Salida por terminal:

```
$ ls -l > listado
$ cat listado
total 0
-rw-r--r-- 1 usuarios usuarios 0 Jan 25 17:27 f1.txt
-rw-r--r-- 1 usuarios usuarios 0 Jan 25 17:27 f2.txt
-rw-r--r-- 1 usuarios usuarios 0 Jan 25 17:27 listado
$ ls -l /home >> listado
$ cat listado
total 0
-rw-r--r-- 1 usuarios usuarios 0 Jan 25 17:27 f1.txt
-rw-r--r-- 1 usuarios usuarios 0 Jan 25 17:27 f2.txt
-rw-r--r-- 1 usuarios usuarios 0 Jan 25 17:27 listado
total 12
drwx----- 58 usuarios usuarios 12288 Jan 24 21:46 usuarios
```

Ejemplos II



Salida por terminal:

```
$ ls /bin/basha > error
ls: cannot access '/bin/basha': No such file or directory
$ ls
error f1.txt f2.txt listado
$ cat error
$ ls /bin/basha > error 2>&1
$ cat error
ls: cannot access '/bin/basha': No such file or directory
```



Cauces, tuberías o Pipes

- La salida estándar de una orden sirve como entrada estándar de otra:

```
ls -l | more
```

- Se combinan cauces y redirecciones:

```
ps aux | grep -v root > ps.out
```

Comodines



- Permiten referirnos a un conjunto de ficheros con características comunes en sus nombres.
 - * corresponde con cualquier conjunto de caracteres.
 - ? corresponde con cualquier carácter individual
 - [conjunto] corresponde con cualquier carácter dentro de conjunto.
- Ejemplo:
 - ?[a-c]*.h
 - Ficheros cuyo nombre comience por un carácter cualquiera seguido de las letras a, b ó c y que acabe en .h



Expansión de órdenes

- Podemos guardar en una variable la salida estándar de una orden o lista de órdenes.

- Ejemplo:

```
num=$( ls a* | wc -w )
```

- Forma equivalente:

```
num=` ls a* | wc -w `
```



1 ¿Qué es un intérprete de comandos?

2 Comandos y operadores del Bash

3 Páginas de manual

4 Redirecciones

5 Guiones Shell

6 Expresiones regulares

Guiones Shell



- Un guión *shell* es un fichero que contiene una secuencia de órdenes *shell*.
- Se crea un proceso *shell* que interpreta las líneas
- Los comentarios comienzan por el carácter `#`
- Ejemplo:

```
#!/bin/bash  
mkdir tmp  
cd tmp  
touch hola  
cd ..
```

Guiones Shell



- Un guión es más versátil si su ejecución depende de parámetros.
- Los parámetros posicionales se denotan por
\$1, \$2, \$3 ... \$9
- Pueden usarse como si fueran variables normales pero además:
 - \$# es el número total de parámetros.
 - shift desplaza a la izquierda los parámetros (decrementando \$#).

Sentencias condicionales I



- Estructura if-then-else:

```
if condicion ; then
    bloque then
else
    bloque else
fi
```
- Nota importante: en la condición, 0 significa *verdadero*, otro valor significa *falso*

Sentencias condicionales II



- Ejemplo:

```
if test -x /bin/bash ; then  
    echo "/bin/bash es ejecutable"  
else  
    echo "/bin/bash no es ejecutable"  
fi
```

- También

```
if [ -x /bin/bash ] ; then ...  
fi
```

Condiciones I



■ Cadenas:

`cadena1 = cadena2`
`cadena1 != cadena2`

Verdadero si son iguales
Verdadero si no son iguales

`¬n cadena`
`¬z cadena`

Verdadero cadena no nula
Verdadero si cadena nula

■ Ficheros

`¬d fichero`
`¬e fichero`
`¬f fichero`
`¬r fichero`
`¬s fichero`
`¬w fichero`
`¬x fichero`

es un directorio
existe
es un fichero regular
tiene permisos de lectura
tiene longitud > 0
tiene permisos de escritura
tiene permisos de ejecución

Condiciones II



■ Aritméticas:

| | |
|--|--|
| <code>expresión1 -eq expresión2</code> | <i>#ambas expresiones son iguales</i> |
| <code>expresión1 -ne expresión2</code> | <i>#ambas expresiones no son iguales</i> |
| <code>expresión1 -gt expresión2</code> | <i>#expresión1 > expresión2</i> |
| <code>expresión1 -ge expresión2</code> | <i>#expresión1 >= expresión2</i> |
| <code>expresión1 -lt expresión2</code> | <i>#expresión1 < expresión2</i> |
| <code>expresión1 -le expresión2</code> | <i>#expresión1 <= expresión2</i> |
| <code>! expresión</code> | <i>#expresión es falsa</i> |

Bucles for



```
for variable in valores  
do
```

```
    cuerpo del for  
done
```

```
for (( i=0 ; $i<10; i++
```

```
do  
    echo $i  
done
```

Salida por terminal:

```
$ for i in `seq 0 1 5`  
> do  
>     echo $i  
> done  
0  
1  
2  
3  
4  
5
```



Bucles while

```
while condición ; do  
    cuerpo del while  
done
```

- Ejemplo:

```
while [ $# -gt 0 ] ; do  
    echo $1 ; shift  
done
```



- 1 ¿Qué es un intérprete de comandos?
- 2 Comandos y operadores del Bash
- 3 Páginas de manual
- 4 Redirecciones
- 5 Guiones Shell
- 6 **Expresiones regulares**

Expresiones regulares I



- Son un mecanismo muy potente para la búsqueda de patrones en cadenas de caracteres.
- Bloques básicos:
 - carácter: coincide con un carácter concreto (p.ej. 'a')
 - . : (punto) coincide con cualquier carácter.
 - ^ : principio de línea.
 - \$: final de línea.
 - [lista] : cualquier carácter dentro de lista
 - [^lista] : cualquier carácter fuera de lista

Expresiones regulares II



- Operadores de repetición. El elemento precedente concuerda:
 - $?$: como mucho una vez (puede ser ninguna).
 - $*$: cero o más veces.
 - $\{n\}$: exactamente n veces.
 - $\{n, \}$: n o más veces.
 - $\{, m\}$: como mucho m veces.
 - $\{n, m\}$: al menos n veces y no más de m .

Ejemplos



- a : cualquier cadena que contenga al menos una a
- ab^* : cualquier cadena que contenga al menos una a
- ab : cualquier cadena que contenga la subcadena ab
- $a.b$: cualquier cadena que tenga una a y una b separadas por un carácter cualquiera
- $\wedge[abc]$: cualquier línea que comience por a , b ó c
- $[\wedge abc]$: cualquier cadena que contenga cualquier carácter distinto de a , b ó c