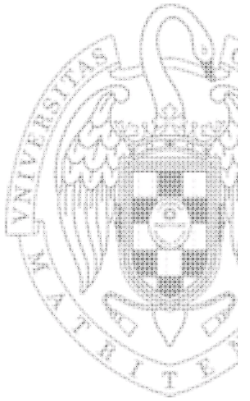


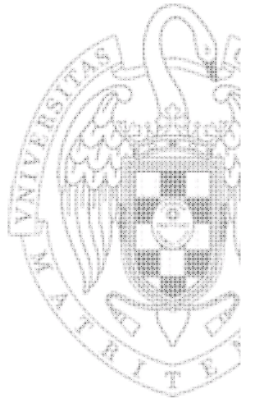
Sistemas Operativos

Curso
2015-2016

Shell

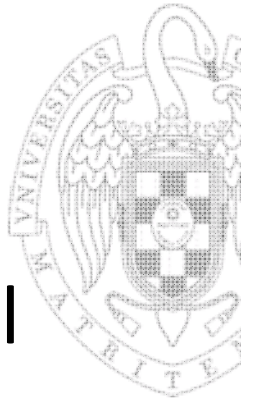
SO

Contenido

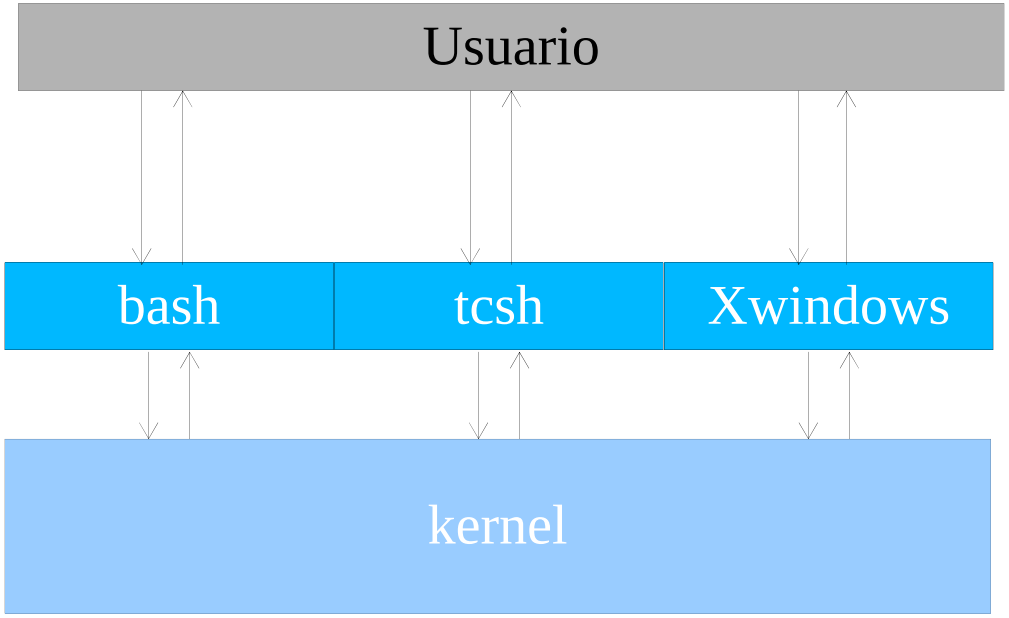


- Introducción
- Páginas de manual
- Principios de Bash

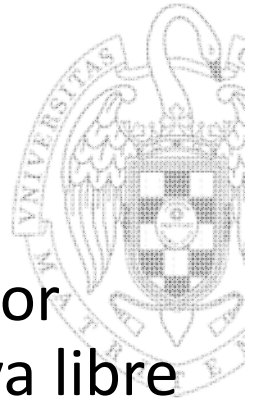
Shell



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

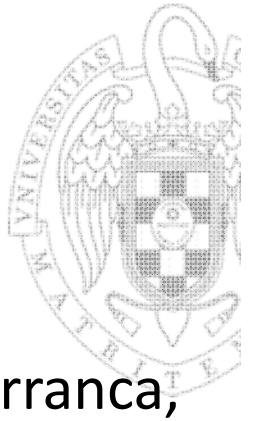


Intérprete de comandos Bash



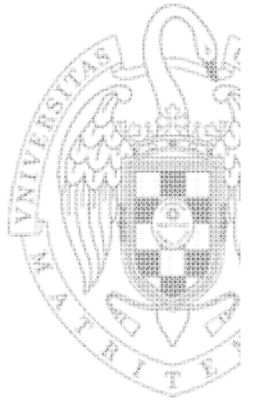
- Bash (Bourne-again shell) es una shell Unix escrita por Brian Fox para el proyecto GNU como una alternativa libre a la shell Bourne
- Bash es el intérprete predeterminado en muchos sistemas UNIX: la mayoría de sistemas GNU/Linux, Solaris y Mac OS X
- También se ha portado a Microsoft Windows (proyecto Cygwin)
- 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 por usuarios con experiencia en UNIX

Ejecución Bash



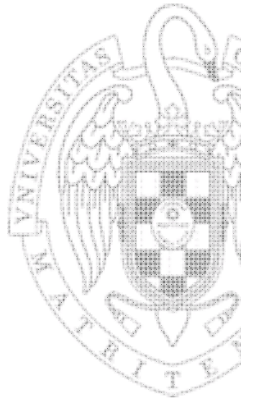
- Comportamiento del shell:
  - Cuando un shell interactivo que no es un login shell arranca, Bash lee y ejecuta órdenes desde `~/.bashrc`, si existiese.
  - Dispone de prefijos o "prompts" (`PS1 y PS2`).
  - Los mandatos se leen en línea (`readline`) y se ejecutan tras su lectura.
  - La historia de mandatos se guarda en fichero (`HISTFILE`) y es posible realizar búsquedas en el historial (CTRL+R)
  - Se permite la expansión de alias.
  - Se pueden modificar los manejadores de señal (Ctrl+C).
  - Se puede controlar la acción a tomar cuando el interprete de comandos recibe un carácter EOF (`ignoreeof.`)

Contenido



- Introducción
- Páginas de manual
- Principios de Bash

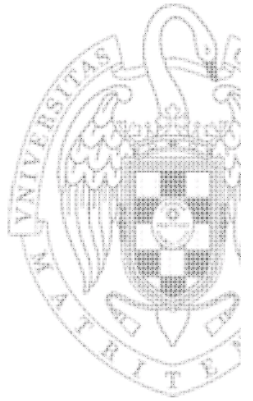
Páginas de manual en UNIX



- `man` da formato y muestra las páginas de manual en línea:
  - 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 (generalmente dispositivos, aquellos ubicados en /dev) y controladores(drivers)`
  - `(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”

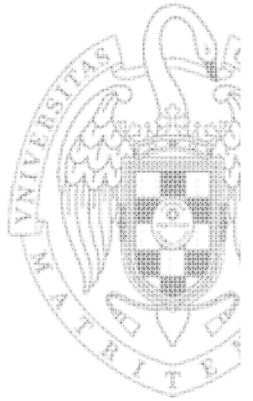


- Manual de un comando/función:
  - `man [1-9] command`
- Buscar en descriptores y nombres de páginas de manual para la clave “keyword” :
  - `man -k keyword`
- Introducción acerca de una sección
  - `man [1-9] intro`
- Llamadas al sistema Linux :
  - `man 2 syscalls`
- Buscar texto en la página de manual para los caracteres especificados:
  - `'/caracteres'` (siguiente: 'n', anterior: 'N')

Subsecciones de una página de manual

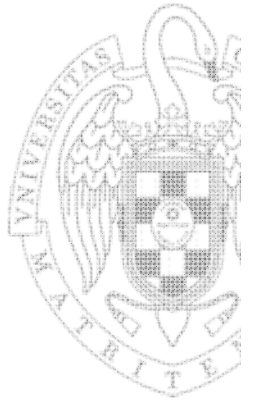
- **NAME** : La primera línea contiene el nombre del comando
- **SYNOPSIS**: Notación técnica con todas las opciones y/o argumentos que este comando puede aceptar
- **DESCRIPTION**: Descripción extensa del comando
- **OPTIONS**: Opciones y su descripción
- **ENVIRONMENT**: Las variables shell que afectan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

Contenido



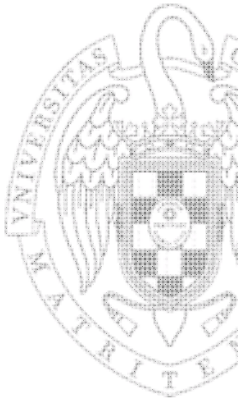
- Introducción
- Páginas de manual
- Principios de Bash

Ejecutando comandos



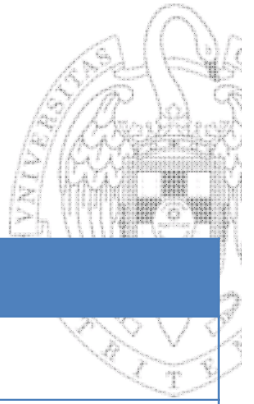
- Tipos de comandos
  - **Comandos internos** (*built-in commands*): Forman parte del repertorio del propio shell
  - **Comandos externos**: Programas externos al shell instalados en el sistema (ficheros binarios ejecutables o scripts)
- Las secuencias de comandos pueden inclurse en un fichero denominado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

Comandos propios



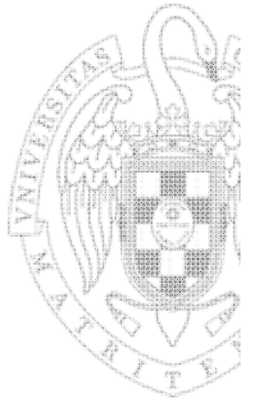
- Bourne Shell built-ins …
  - `:, ., break, cd, continue, eval, exec, exit, export, getopts, hash, pwd, readonly, return, set, shift, test, [, times, trap, umask and unset.`
- + Bash built-in commands:
  - `alias, bind, builtin, command, declare, echo, enable, help, let, local, logout, printf, read,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

Comandos básicos



Comando	Descripción
`ls`	Lista los ficheros del directorio actual
`pwd`	Muestra en qué directorio nos encontramos
`cd directory`	Cambia de directorio
`man command`	Muestra la página de manual para el comando dado
`apropos string`	Busca la cadena en la base de datos whatis
`file filename`	Muestra el tipo de fichero dado
`cat textfile`	Muestra el contenido del archivo en pantalla
`exit/logout`	Abandona la sesión
`grep`	Busca en archivos líneas que contengan un patrón de búsqueda dado
`echo`	Muestra una línea de texto
`env`	Guarda información en el entorno
`export`	Cambia el valor de una variable de entorno

Variables y operadores



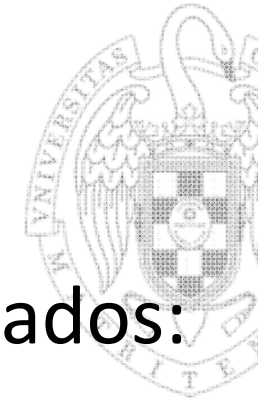
- Variables

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

- Operadores aritméticos y de bits:

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

Redirecciones



- Tres descriptores de ficheros predeterminados:

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

- Redirección de la salida estándar:

  ```
  orden > fichero
  ```

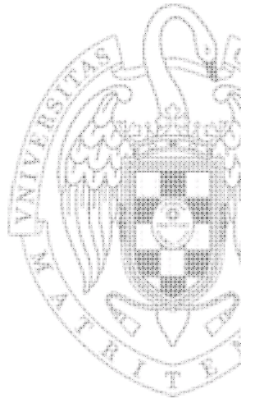
- Redirección de la salida de error:

  ```
  orden 2> fichero
  ```

- Redirección de la entrada estándar:

  ```
  orden < fichero
  ```

Ejemplos



- Redirecciones:

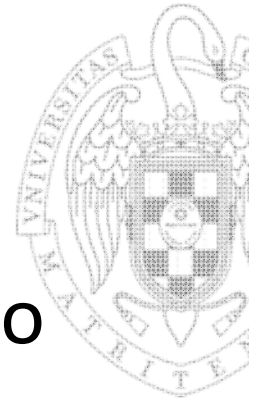
```
ls -l > listado

ls -l /etc >> listado

ls /bin/basha 2> error

find / -name 'lib*' -print > librerias 2>&1
```

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
```

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

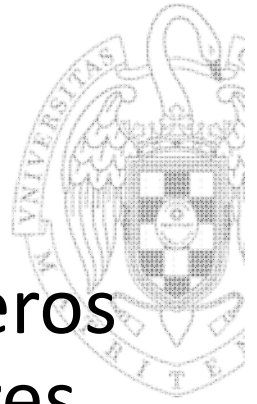
- *foreground* y *background*
  - En modo interactivo los procesos se ejecutan en primer plano (*foreground*): la *shell* no muestra el *prompt* hasta que no finaliza la ejecución de la última orden introducida.
  - Si queremos dejar el proceso en segundo plano (background) se añade **&**:

```
$ xeyes &
[2] 7584
```

PID

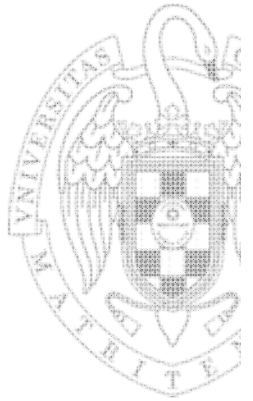
Job_ID

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`** cualquier fichero cuyo nombre comience por un carácter cualquiera seguido de las letras **`a`**, **`b`** ó **`c`** y que acabe en **`.h`**

Expansión de órdenes



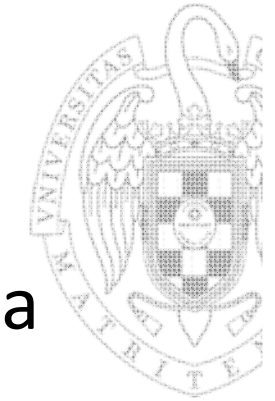
- Podemos guardar en una variable la salida estándar de una orden o lista de órdenes.
- Ejemplo:

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

- Forma equivalente:

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

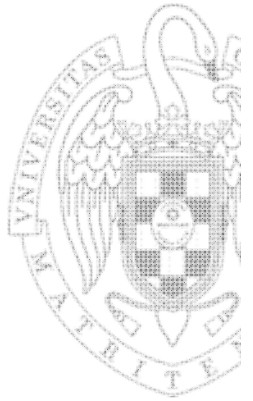
Guiones Shell



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

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

Guiones Shell I



- 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.

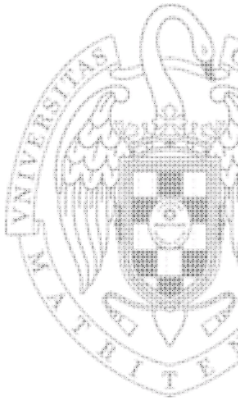
Sentencias condicionales

- 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



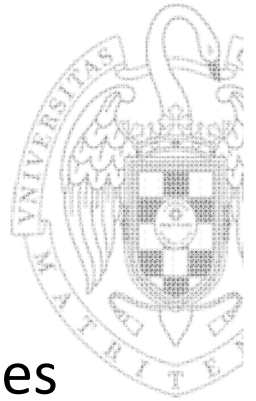
- Ejemplo:

```
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...
```

Condiciones



- Cadenas:

`cadena1 = cadena2`	Verdadero si son iguales
`cadena1!= cadena2`	Verdadero si no son iguales
`-n cadena`	Verdadero cadena no nula
`-z cadena`	Verdadero si cadena nula

- Ficheros

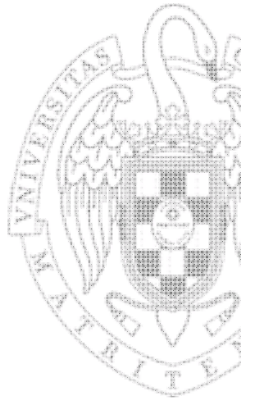
`-d fichero`	es un directorio
`-e fichero`	existe
`-f fichero`	es un fichero regular
`-r fichero`	tiene permisos de lectura
`-s fichero`	tiene longitud > 0
`-w fichero`	tiene permisos de escritura
`-x fichero`	tiene permisos de ejecución

Condiciones II

- Aritméticas

`expresión1 -eq expresión2`	ambas expresiones son iguales
`expresión1 -ne expresión2`	ambas expresiones no son iguales
`expresión1 -gt expresión2`	`expresión1` > `expresión2`
`expresión1 -ge expresión2`	`expresión1` ≥ `expresión2`
`expresión1 -lt expresión2`	`expresión1` < `expresión2`
`expresión1 -le expresión2`	`expresión1` ≤ `expresión2`
`! expresión`	`expresión` es falsa

Bucles for



- Bucles
  - Bucle **for (I)**:

```
for variable in valores
do
  cuerpo del for
done
```

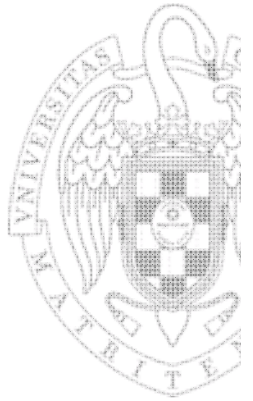
  - Ejemplo

```
for i in `seq 0 1 9`
do
    echo $i
done
```

  - Bucle **for (II)**:

```
for (( i=0 ; $i<10; i++ ))
do
  echo $i
done
```

Bucles While



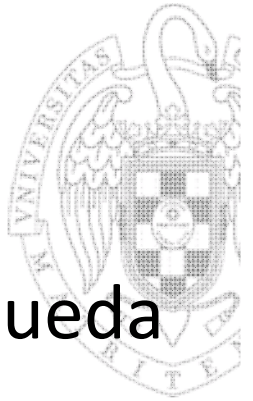
- Bucle **`while`**:

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

- Ejemplo:

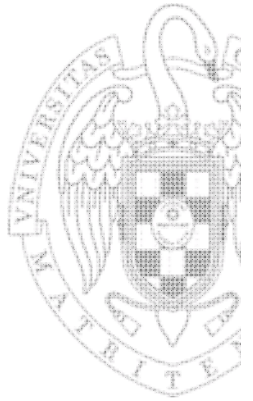
```
while [ $# -gt 0 ] ; do
     echo $1 ; shift
done
```

Expresiones regulares



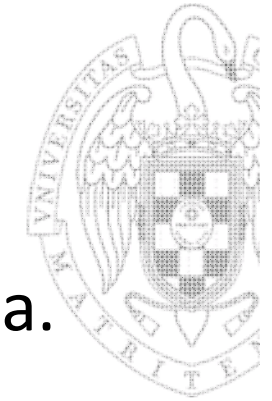
- Son un mecanismo muy potente para la búsqueda de patrones en cadenas de caracteres.
- Bloques básicos:
  - **`carácter`**: coincide con un carácter concreto. P.e. a
  - **`.`** : (punto) coincide con cualquier carácter.
  - **`^`** : principio de línea.
  - **`$`** : final de línea.
  - **`[lista]`** : 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.

^[abc] : cualquier línea que comience por **a**, **b** ó **c**.

[^abc] : cualquier cadena que contenga cualquier carácter distinto de **a**, **b** ó **c**.