Bloque II. Herramientas

Capítulo 9. Uso de sistemas operativos

- Introducción
- Sistemas operativos comunes
- El sistema de ficheros
- Uso de la memoria USB
- El intérprete de órdenes
- Ejecución de programas
- Guiones (scripts)

9..1 Introducción

El ***sistema operativo*** es un programa que:

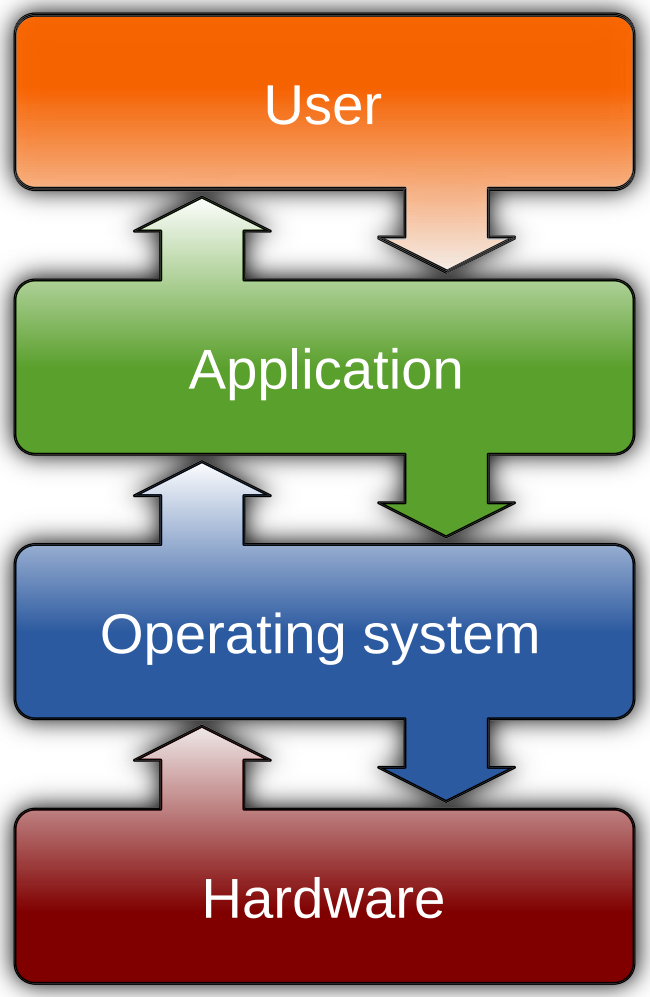
- Controla el acceso a todos los ***recursos del sistema*** (memoria, teclado, pantalla, etc.)
- Implementa un ***sistema de ficheros*** para el almacenamiento de información en la memoria secundaria o masiva
- Tiene un ***intérprete para las órdenes*** que el usuario introduce por teclado ("***shell***") o ratón y pantalla (gestor de ventanas). Con él podemos:
  - gestionar la información almacenada en el disco
  - ejecutar programas de aplicación

El sistema operativo se carga en la memoria del computador al encenderlo

Ubicación lógica del sistema operativo

Principales componentes del computador

- Usuarios
  - personas, otros computadores
- Aplicaciones
  - conjuntos de instrucciones que facilitan la realización de tareas
  - utilizan los recursos del sistema
  - ejemplos: editor de textos, navegadores, juegos
- Sistema operativo
  - controla y coordina el uso del hardware
  - uso concurrente
- Hardware
  - CPU, memoria, dispositivos de E/S



Fuente: wikipedia

9.2 Sistemas operativos comunes

De propósito general

UNIX/Linux
estándar POSIX
versiones libres
(Linux, BSD, ...)

Windows
propietario

Mac OS
propietario
basado en UNIX BSD

Para móviles

Android
gratuito
basado en Linux
lenguaje Java

iOS
basado en UNIX BSD
lenguaje Swift
(antes Objective C)

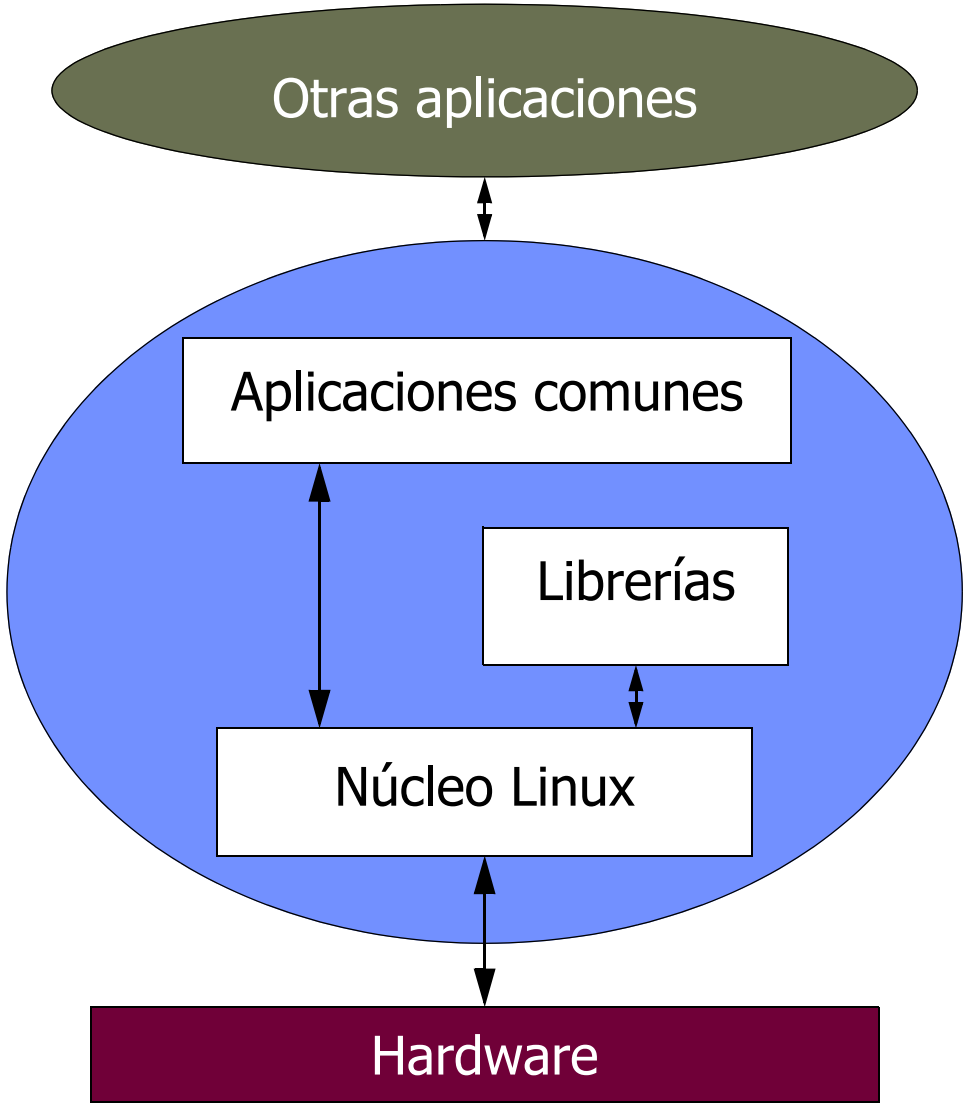
Para sistemas empotrados

De tiempo real
VxWorks, QNX
RTEMS, ...
MaRTE OS

Sist. pequeños
Raspbian, ROS,
TinyOS, ...

sin SO
Arduino
PIC

El sistema operativo Linux



Distribución Linux

Ejemplos de distribuciones

fedora

debian

ubuntu

9.3 El sistema de ficheros

Los programas y la información no volátil se almacenan en el ***sistema de ficheros*** del computador, basado en memoria secundaria (discos, memorias USB, DVD, etc.)

La información se almacena en ***ficheros***, que se identifican por un ***nombre***, con el siguiente formato:

- cualquier secuencia de hasta 256 caracteres (excepto "`/`" en UNIX y "`\/:?*"<>`" en Windows)
- se suele identificar el tipo de fichero con una ***extensión***:
  - programa python: `nombre.py`
  - fotografía jpeg: `nombre.jpg` o `nombre.jpeg`
  - página web: `nombre.html` o `nombre.htm`
  - etc.

Ficheros y directorios

Los ficheros pueden ser:

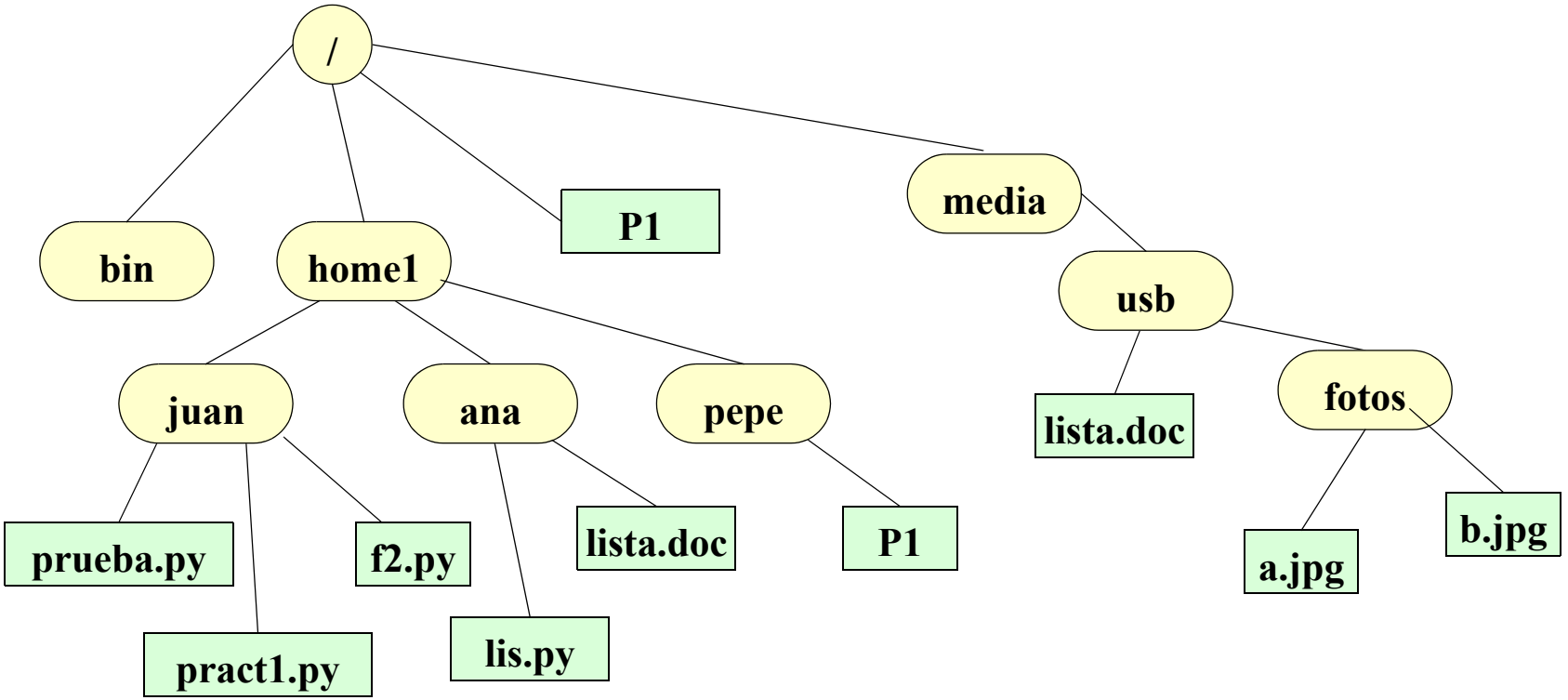
- ficheros de información (programas o datos)
- directorios (o carpetas), que a su vez contienen otros ficheros

Los ficheros y directorios se organizan con una estructura jerárquica, en forma de árbol

- La ***raíz*** del árbol se denomina en UNIX: "/"
  - Al revés que en Windows: "\"

Árboles de ficheros en UNIX

Los directorios dan lugar a una estructura en forma de un único árbol, con todas las unidades de memoria secundaria (disco, USB, ...)



Nombres de ficheros UNIX: rutas absolutas

El directorio raíz se llama "/"

Nombre completo de un fichero ("*pathname*" o "*Ruta absoluta*"):

- empieza en la raíz: "/"
- siguen los nombres de los directorios de los que depende en el árbol de ficheros, separados por "/"
- termina con el nombre del fichero o directorio

Ejemplo: `/home1/juan/prueba.py`

En UNIX son distintas las mayúsculas de las minúsculas

Directorio de trabajo y rutas relativas

Para no escribir el nombre completo del fichero existe un ***directorio de trabajo***:

- si el fichero está en el directorio de trabajo, se puede omitir el nombre de este para formar *rutas relativas*

Ejemplo: suponiendo que el directorio de trabajo es `/home1/juan`, son equivalentes:

```
/home1/juan/prueba.py
prueba.py
```

Ejemplo: suponiendo que el directorio de trabajo es `/home1`, son equivalentes:

```
/home1/ana/lista.doc
ana/lista.doc
```

Navegación por el árbol de ficheros

Existen estas notaciones para usar con rutas relativas:

- El ***directorio de trabajo*** se denomina "`.`"
- El ***padre*** de un directorio se llama "`..`"

Ejemplos:

```
../ana/lis.py
../../bin
```

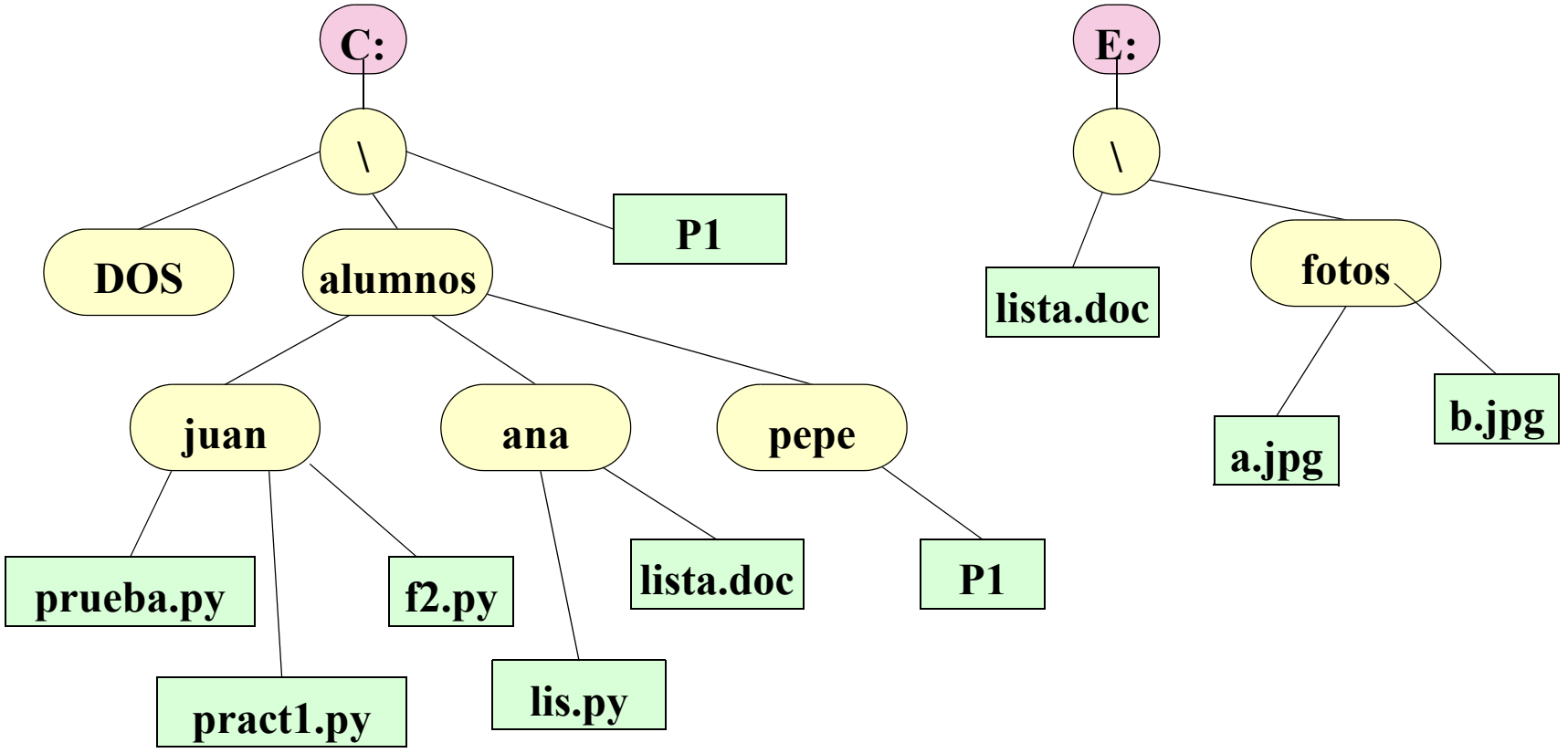
Atajo para acceder al directorio de usuario: `'~'`

- por ejemplo, si el directorio de usuario es `/home1/juan`, son equivalentes:

```
/home1/juan/prueba.py
~/prueba.py
```

Árboles de ficheros en Windows

Los directorios dan lugar a una estructura en forma de árbol, con un árbol por cada unidad de memoria secundaria



Rutas absolutas en: Windows

Cada unidad de almacenamiento secundario (disco, memoria USB, DVD) tiene como nombre una letra, seguida de ":"

- memoria USB: `e:` disco duro: `c:`

El directorio principal (raíz) se llama "`\`"

El nombre completo de un fichero ("*ruta absoluta*") tiene:

- nombre de la unidad
- nombres de los directorios de los que depende en el árbol de ficheros, separados por "`\`"
- nombre del fichero

Ejemplo: `c:\alumnos\juan\prueba.py`

En los nombres no se distinguen mayúsculas de minúsculas

Unidad y directorio de trabajo

Para no escribir la ruta absoluta del fichero existen rutas relativas basadas en estos conceptos:

- una unidad de trabajo: si el fichero está en la unidad de trabajo esta se puede omitir
- un directorio de trabajo: idem.

Ejemplo: suponiendo que la unidad de trabajo es `c:` y el directorio de trabajo es `\alumnos\juan`, son equivalentes:

```
c:\alumnos\juan\prueba.py
\alumnos\juan\prueba.py
prueba.py
```

Navegación: El directorio padre se representa por "`..`"; ejemplos:

```
..\ana\lis.py          ..\..\dos
```

Caracteres comodín

En ocasiones podemos referirnos globalmente a múltiples ficheros cuyos nombres se parecen. Para ello se usan los caracteres comodín:

- "?" puede ser sustituido por cualquier carácter
- "*" por cualquier secuencia de cero o más caracteres
- Ejemplos UNIX:

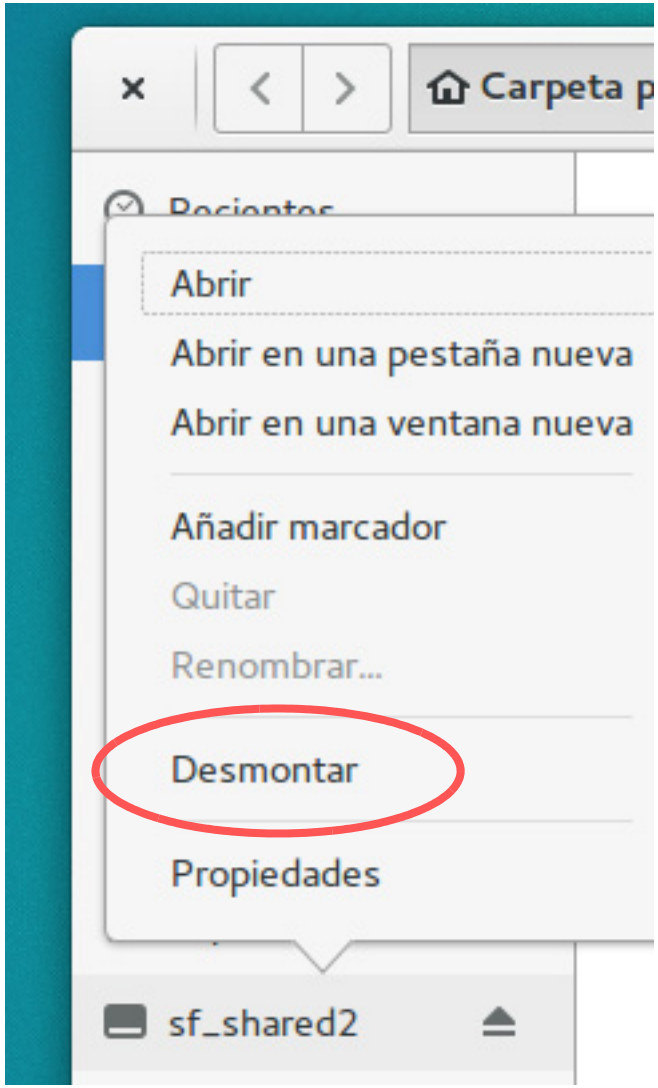
  `/home1/juan/*.py` (prueba.py, f2.py y pract1.py)

  `/home1/*` (juan, ana y pepe)

9.4 Uso de la memoria USB en LINUX

Desde el gestor gráfico de ficheros

- es imprescindible acordarse de ***desmontar*** el dispositivo
  - desde el gestor gráfico de ficheros seleccionar "***quitar de forma segura***" o "***desmontar***" o "***expulsar***" con el botón derecho del ratón sobre la memoria USB
- si no lo hacemos se puede corromper la información



9.5 Intérprete de órdenes

El sistema operativo dispone de un intérprete de las órdenes que se introducen por teclado ("***shell***")

- llamado "***Símbolo del sistema***" en Windows (ver ***accesorios***)
- llamado "***terminal***" en OS X (ver ***aplicaciones*** => ***utilidades***)
- llamado "***sh***", "***csh***", "***bash***", ..., en Unix (***terminal*** en el menú Ubuntu)

Cuando el intérprete está listo para recibir una orden, muestra el símbolo de preparado ("***prompt***"), generalmente con el directorio de trabajo o el nombre del computador. Ejemplo:

- `pc37 usr>`

Bajo el intérprete se puede:

- introducir una orden o ejecutar un programa

Órdenes de navegación: listar
ls

Orden	`ls`	**Opciones**	`-l`
Función	Muestra una lista del contenido del directorio		
Sintaxis	`ls nombreDirectorio`* `ls -l nombreDirectorio`		
Ejemplos			
`ls`		Muestra el contenido del directorio actual	
`ls -l prueba`		Muestra el contenido del directorio `prueba`	
`ls -l -a prueba`		Muestra el contenido *completo* del directorio `prueba`	

* Este color representa un argumento opcional: `nombreDirectorio`

Facilidades del intérprete de órdenes

Historial de órdenes (LINUX):

- para avanzar atrás o adelante en el historial de órdenes pulsar ↑ o ↓
- la orden se puede editar con ← y →

Completar nombres de fichero (LINUX):

- pulsar los primeros caracteres del nombre y luego `<TAB>`

Interrumpir la ejecución de un programa:

`<ctrl>c`

Órdenes de navegación: directorio de trabajo: pwd

Orden	pwd		
Función	Muestra el nombre completo del directorio de trabajo		
Sintaxis	pwd		
Ejemplos			
pwd			

Órdenes de navegación: cambiar directorio: cd

Orden	`cd`		
Función	Cambiar el directorio de trabajo		
Sintaxis	`cd nombreDirectorio`		
Ejemplos			
`cd`		Hace que el directorio de trabajo sea el inicial del usuario	
`cd prueba`		Hace que el directorio de trabajo sea `prueba`	

Órdenes de navegación: mostrar texto less

Orden	`less`		
Función	Muestra en pantalla un fichero de texto Permite la navegación arriba y abajo (***flechas***) Salir con `q` (***quit***)		
Sintaxis	`less nombreFichero`		
Ejemplos			
`less pepe.txt`		Muestra el contenido del fichero `pepe.txt`	

Para ver el contenido de un fichero de texto también pueden usarse las órdenes `more` y `cat`

Otras órdenes de navegación

Orden	Función	Ejemplo de Sintaxis
`man`	Pedir info sobre una orden	`man orden`
`diff`	Comparar dos ficheros y ver sus diferencias	`diff fichero1 fichero2`
`grep`	Buscar texto en múltiples ficheros	`grep "texto" *.txt`
`find`	Busca ficheros en un directorio	`find dir -name *.txt` `find . -newer fichero`
`tail`	Mostrar las últimas líneas de un fichero	`tail -n 20 fichero`
`head`	Mostrar las primeras líneas de un fichero	`head -n 15 fichero`
`wc`	Cuenta líneas y palabras	`wc fichero.txt`

Hay muchísimas más

Órdenes de gestión de ficheros: borrar rm

Orden	`rm`	**Opciones**	`-r`
Función	Borra ficheros		
Sintaxis	`rm nombreFichero` `rm -r nombreDirectorio`		
Ejemplos			
`rm pepe`		Borra el fichero `pepe`	
`rm *.java`		Borra todos los ficheros del directorio de trabajo que acaben en `.java`	
`rm -r proyecto1`		Borra el directorio `proyecto1` y todos sus contenidos, recursivamente	

Órdenes de gestión de ficheros: crear directorio: mkdir

Orden	`mkdir`		
Función	Crea un directorio		
Sintaxis	`mkdir nombreDirectorio`		
Ejemplos			
`mkdir proyecto3`		Crea un directorio vacío llamado `proyecto3`	

Órdenes de gestión de ficheros: borrar directorio vacío: rmdir

Orden	`rmdir`		
Función	Borra un directorio vacío		
Sintaxis	`rmdir nombreDirectorio`		
Ejemplos			
`rmdir proyecto1`		Borra el directorio `proyecto1`, si está vacío	

Órdenes de gestión de ficheros: copiar cp

Orden	`cp`	**Opciones**	`-r`
Función	Copia ficheros		

Sintaxis	`cp ficheroOrigen ficheroDestino`
	`cp ficheroOrigen dirDestino`
	`cp -r dirOrigen dirDestino`		

Ejemplos			
`cp pepe juan`	Copia el fichero `pepe` en otro llamado `juan`		
`cp Hola.* proyecto1`	Copia los ficheros que empiecen por `Hola.`, con el mismo nombre, en el directorio llamado `proyecto1`		
`cp -r proyecto1 proyecto2`	Copia recursivamente (con sus subdirectorios) el directorio `proyecto1`, en el directorio `proyecto2`		

Órdenes de gestión de ficheros: mover mv

Orden	`mv`	**Opciones**	
Función	Cambia de nombre un fichero o lo mueve a otro directorio		

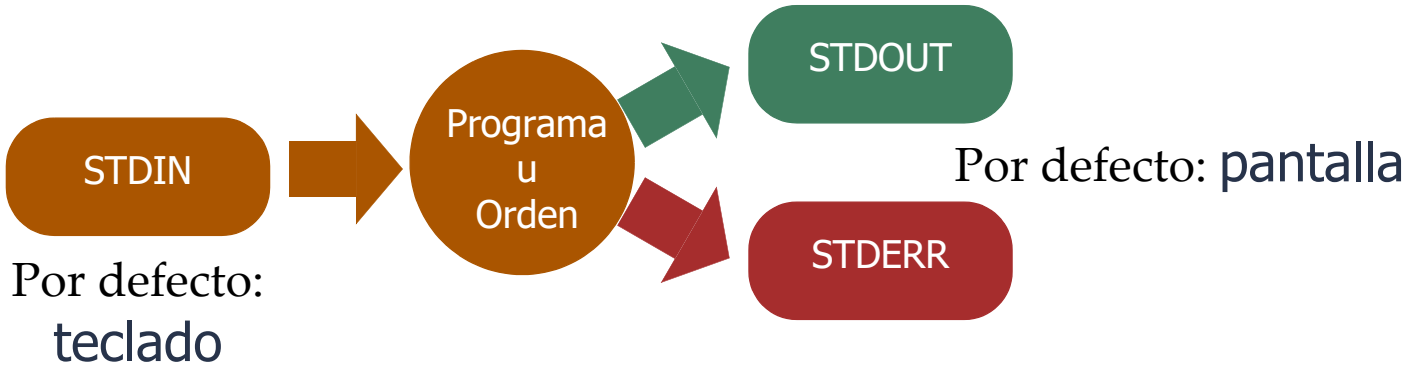
Sintaxis	`mv nombreViejo nombreNuevo`
	`mv nombreFichero nombreDirectorio`
	`mv nombreDirectorioOrigen nombreDirectorioDestino`		

Ejemplos			
`mv pepe juan`	Cambia el nombre del fichero `pepe` al nuevo `juan`		
`mv juan proyecto1`	Mueve el fichero `juan` al interior del directorio `proyecto1`		
`mv proyecto1 proyecto3`	Mueve recursivamente el directorio `proyecto1` al interior del directorio `proyecto3`		

Redirección de entrada/salida

Flujo de datos vinculados a los programas y las órdenes de la *shell*. Habitualmente, los programas:

- leen datos de su entrada estándar (STDIN)
- muestran resultados en su salida estándar (STDOUT)
- muestran notificaciones de error en su salida estándar (STDERR)



Redirección de entrada/salida

Es posible redirigir la entrada o la salida hacia otro lugar

- por ejemplo, un fichero u otra orden

Destino	Oper.	Función	Ejemplo
Ficheros	>	STDOUT a un fichero	wc --help > wc.txt
	>>	STDOUT a un fichero existente	man wc >> wc.txt
	<	usa fichero como STDIN	cat < wc.txt
	&>	STDOUT y STDERR a un fichero	mv -v *.txt dir &> errors.txt
Orden	|	STDOUT a otra orden	cat wc.txt | tail -n 20 | wc -w

9.6 Ejecución de programas

Para ejecutar un programa basta escribir su nombre bajo el intérprete de órdenes (sin extensión):

- UNIX: nombre del fichero ejecutable; por ejemplo si se ha creado el programa `practi1`:
  - Usando una ruta absoluta:
    ```
    /home1/juan/practi1
    ```
  - Usando una ruta relativa. Debe comenzar por el directorio actual `'.'`:
    ```
    ./practi1
    ```

Ruta de acceso ("PATH")

La ruta de acceso (PATH) es una variable de entorno que almacena una lista de directorios donde se buscan los programas a ejecutar

Si el programa está en el PATH basta escribir su nombre simple:

- `practi1`

La ruta de acceso se puede consultar con la orden:

- `echo $PATH`

Cambiar la ruta de acceso (“PATH”)

En Linux el cambio del PATH depende de la shell. Puede consultarse en

https://rootsudo.wordpress.com/2014/04/06/el-path-la-ruta-de-linux-variables-de-entorno/

http://es.ccm.net/faq/315-bash-la-variable-de-entorno-path#v-anadir-un-directorio-a-la-variable-path

9.7 Guiones (*scripts*) en Unix

Es posible crear archivos con secuencias de órdenes

- nos evita tener que teclearlas de nuevo
- automatización de tareas repetitivas

Las órdenes se escriben en un archivo de texto

- se puede usar el editor ***gedit*** o similar:
- `gedit nombre_archivo`

Para poder ejecutarlas es preciso dar permiso de ejecución al archivo

- `chmod +x nombre_archivo`

Luego ejecutarlo así:

```
./nombre_archivo
```

Comentarios y escritura en pantalla en *scripts*

Es conveniente explicar al lector de un script lo que éste va haciendo

- con comentarios, usando este formato:

```
# comentario hasta el final de la línea
```

Alternativamente es conveniente explicar al usuario de un script lo que éste va haciendo, poniendo mensajes en pantalla

- Usar para ello esta orden:

```
echo "mensaje en pantalla"
```