

# S.O. Unix / Editor vi

Marta Elena Zorrilla Pantaleón

Dpto. Matemática Aplicada y Ciencias de la  
Computación



# Temario

Sistemas  
Informáticos  
II

- Características básicas
- Historia
- Estándares
- Conexión y desconexión del sistema
- Ficheros y directorios
  - Organización de directorios
  - Nombres de ficheros y directorios
  - La especificación de un directorio
  - Tipos de ficheros
- Comandos básicos
  - Comienzo
  - Manejo de ficheros y directorios
  - Operadores de redirección (<,>,>>)
  - Pipes o tuberías (|)
  - Búsqueda
  - Ordenación



# Temario

---

Sistemas  
Informáticos  
II

- Otros comandos útiles
- Manuales
- Shell
- Procesos
- Administración del S.O. Unix
- Arranque de la máquina
- Desconexión de la máquina
- Seguridad
- Temporización y planificación
- Editor vi (visual)
  - Crear o editar un fichero
  - Comandos más útiles
- Correo electrónico
- Transferencia de ficheros
- Conexión remota
- LINUX



# S. O. Unix

---

## Características:

- Multiusuario: diferentes usuarios con distintos tipos de privilegios y con accesos a diferentes partes del sistema.
  - seguridad del sistema
  - privacidad de datos
  - identificación de los usuarios (login y passwd)
- Multitarea: varios procesos al mismo tiempo.
  - procesos de uno o varios usuarios
  - tareas del sistema (planificador de tareas, permisos de acceso, ....)
- Usa memoria virtual: utiliza área de swapping. Una parte del disco se configura para descargar en ella las tareas que no se están ejecutando para así liberar la memoria y permitir ejecutar otras tareas más prioritarias.



# S. O. Unix

---

- Especialización: pequeños programas que hacen una sola cosa muy bien.
- Portabilidad: trasladables a otras máquinas con pocas modificaciones (está escrito en C con cuidado de aislar las rutinas dependientes del hardware).
- Sistema de ficheros jerárquico.
- Independencia de dispositivo. Las E/S están integradas en el sistema de ficheros. Son tratados de la misma manera que los ficheros.
- Interfaz con el usuario simple e interactiva. La shell es un programa independiente que el usuario puede sustituir, manteniéndose la sintaxis de las órdenes.
- Proporciona un entorno completo de programación.
- Carácter abierto. Permite ampliar fácilmente la funcionalidad sin tener que depender de un único fabricante.



# S. O. Unix

---

## Historia

- Nació en Laboratorios Bell de AT&T en 1968 creado por Dennis Ritchie y Ken Thompson.
- La primera versión se escribió en lenguaje ensamblador.
- En 1973, Ritchie y Thompson le escribieron en lenguaje de alto nivel, lenguaje C, por lo que le hacía muy portable.
- Además, al ser inicialmente su código fuente de libre distribución, se contribuyó al desarrollo de diferentes versiones: BSD de la Universidad de Berkeley, Sistema V 3.2 de AT&T, Xenix de Microsoft, SunOS de Sun Microsystems. Todas ellas se fusionaron en Unix Sistema V version 4.0 a finales de los 80.



# S. O. Unix

---

## Estándares

- En 1984 una organización constituida por usuarios del S.O. Unix definen un estándar para garantizar la portabilidad de las aplicaciones.
- **POSIX** (Portable Operating System Interface for Computer Environments): familia de estándares que definen cualquier aplicación que interactúa con un sistema operativo. Las áreas cubiertas por este estándar son: llamadas al sistema, bibliotecas, herramientas, interfaces, verificación y prueba, características de tiempo real y seguridad. IEEE.
- **X/OPEN** (X/Open Portability Guide). Consorcio internacional de vendedores de computadoras cuyo objetivo es estandarizar las interfaces software, recoge también un conjunto importante de estándares del Sistema Unix.



# S. O. Unix

---

- **Conexión al sistema:**
  - login y password
  - prompt del sistema: \$ o %
- **Desconexión del sistema:**
  - exit





# S. O. Unix

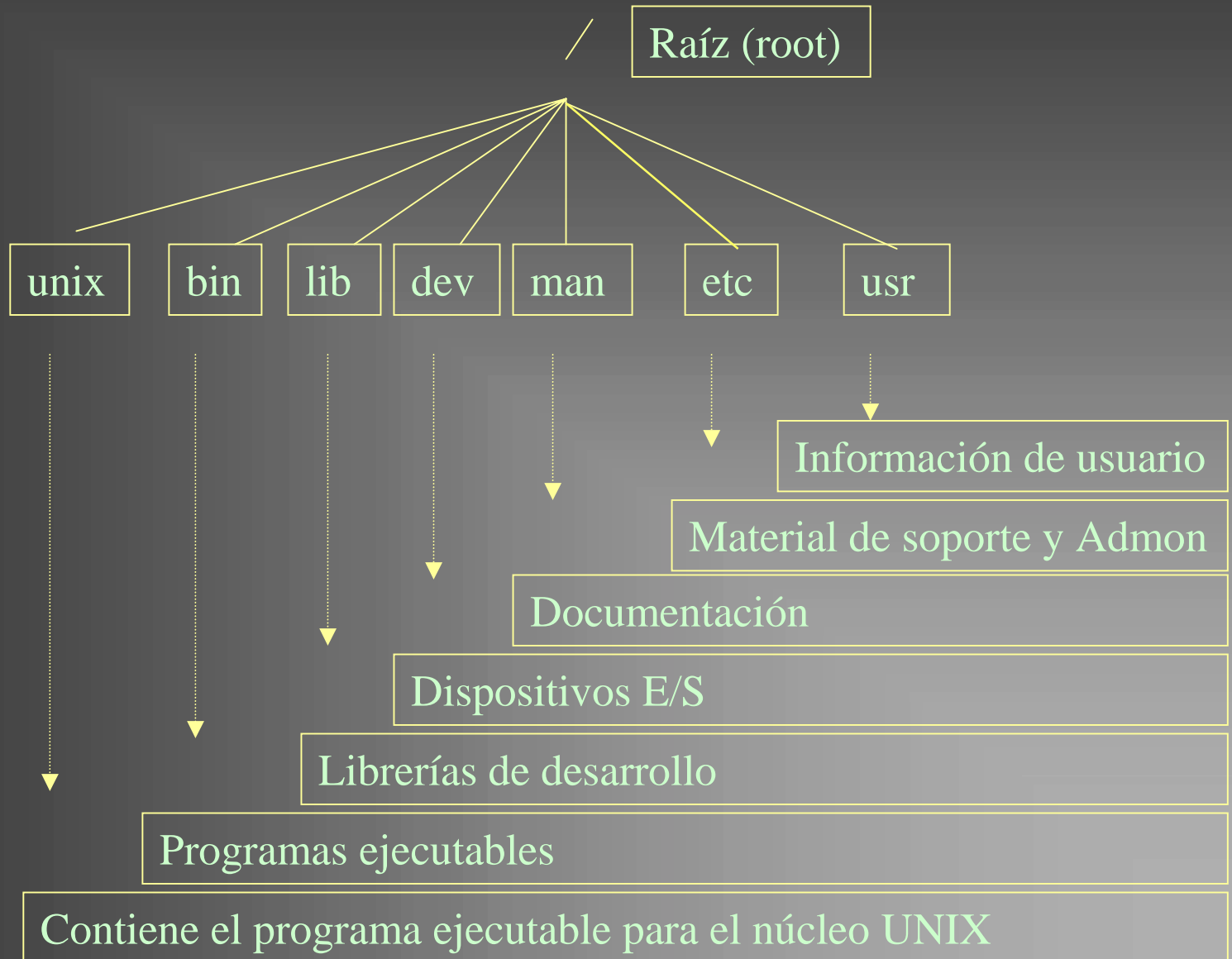
---

## Ficheros y directorios:

- Contribución del SO Unix  $\Rightarrow$  Sistema de ficheros  $\Rightarrow$  Estructura jerárquica
- En Unix todo se trata como ficheros. Los ficheros son una secuencia de bytes y los directorios son una colección de ficheros que a su vez contienen ficheros.
- **home directory**: directorio a partir del cual el usuario almacenará sus ficheros.



# S. O. Unix





# S. O. Unix

## - Nombres de ficheros y directorios:

- nombres compuestos por letras y dígitos, el punto y underline
- los ficheros que comienzan por punto no aparecen en el directorio con el comando ls
- el punto se utiliza para indicar extensión (tipo de fichero)
- unix distingue mayúsculas de minúsculas.

## - La especificación de un directorio puede ser:

- Absoluta: path completo /usr/marta
- Relativa: Empieza con el nombre del directorio debajo del directorio en curso.
  - "." : indica directorio actual
  - ".." : indica directorio padre



# S. O. Unix

## - Tipos de ficheros

### ➤ Ficheros ordinarios

### ➤ Vínculos: asignación de más de un nombre a un fichero.

➤ Físicos: Para no tener inconsistencias entre dos ficheros que son una copia del otro. No se puede utilizar para directorios ni para compartir ficheros entre computadoras distintas.

➤ Simbólicos: Permite atravesar sistemas de ficheros que residen en dispositivos físicos distintos. Se puede utilizar para ficheros y directorios.

➤ Directorios: fichero que contiene la referencia de los ficheros que contiene. Siempre tiene, al menos, dos entradas . y ..

➤ Ficheros especiales: ficheros que representan dispositivos físicos (terminal, disco, elemento de comunicaciones,...)

➤ Ficheros especiales de caracteres, c, para transmitir datos carácter a carácter (terminal, modem o impresora)  
/dev/lp0, /dev/tty00

➤ Ficheros, b, para transmisión por bloques (discos y cintas magnéticas)) /dev/fd0, /dev/hda



# S. O. Unix

---

## Comandos básicos:

### Sintaxis:

`$comando [flag1, flag2,..., flagn] arg1 arg2 ... argn`

- Los flags son precedidos normalmente por un guión para evitar que sean interpretados como un nombre de fichero. Cambian el comportamiento de los comandos.
- Los argumentos suministran información adicional para la ejecución del comando.
- Se pueden ejecutar varios comandos en la misma línea separándolos con punto y coma.

`$who ; date`



# S. O. Unix

---

- **Comienzo:**

\$login : para introducirse en el sistema

\$passwd : para cambiar la palabra clave de entrada

- **Manejo de ficheros y directorios:**

\$cat > fich1 : crea el fichero fich1 donde se  
escribe líneas y se termina con Ctrl-D

\$cat fich1 : visualiza su contenido en pantalla

\$cat fich1 fich2 : visualiza la concatenación del fichero  
1 y el 2

\$cp fich1 fich2 : copiar fich1 con nombre fich2



# S. O. Unix

---

\$rm fich1 : borra fich1

\$mkdir dir1 : crea directorio dir1

\$rm -R dir1 : borra dir1 con todos los ficheros  
y subdirectorios que contiene

\$rmdir dir1 : borra dir1 si está vacío

\$ls : listar contenido de directorio

Opciones:

- a lista todos los ficheros
- c ordena por fecha de creación
- l da toda la información
- r invierte el orden del listado
- s indica el tamaño en bloques



# S. O. Unix

---

- \$cd /usr/marta : para cambiar al directorio usr/marta
- \$cd .. : para ir al directorio jerárquicamente superior
- \$cd marta : para ir al directorio marta a partir del directorio en curso
- \$pwd : muestra el directorio de trabajo actual
- \$ln fich1 ./dir1/fich3 : vincula fich1 en el directorio actual con el fichero fich3 en el directorio dir1
- \$lp fich1 : imprimir fich1





# S. O. Unix

---

## Comodines o metacaracteres (\*,?,[ ]):

- |                   |   |
|-------------------|---|
| \$ls -l fich*     | : lista ficheros que comienzan por fich   |
| \$ls -l fich?     | : lista ficheros que comienzan por fich y le sigue un único carácter.               |
| \$ls -l fich[1-5] | : lista ficheros que comienzan por fich y le sigue un carácter en el rango indicado |



# S. O. Unix

```
Telnet - cuss5
Conectar Edición Terminal Ayuda
44% ls -al
total 50
drwxr-xr-x  6 marta    informix  512 Ene 26 20:14 .
drwxrwxr-x 26 root     auth      512 Sep  4 1998 ..
-rw-r--r--  1 marta    informix 1390 Oct  1 1997 .cshrc
-r-----  1 marta     auth         0 Ene 26 19:48 .lastlogin
-rw-----  1 marta     informix   934 Oct  1 1997 .login
-rw-r--r--  1 marta     informix    5 Sep  3 1998 .mosaicpid
drwxr-xr-x  4 marta     informix   512 Sep  3 1998 .odtpref
-rw-r--r--  1 marta     informix    38 Sep  3 1998 .scohelp-hotlist
drwx----- 2 marta     informix   512 Sep  3 1998 .scohelp-pers-annot
-rw-----  1 marta     informix    49 Sep  3 1998 .Xauthority
-rw-r--r--  1 marta     informix    0 Sep  2 1998 .xdtsupCheck
drwxr-xr-x  5 marta     informix   512 Sep  2 1998 .xdt_dir
drwxr-xr-x  2 marta     informix   512 Ene 26 20:16 dir1
-rw-r--r--  2 marta     informix    68 Ene 26 20:02 fich1
-rw-r--r--  1 marta     informix    68 Ene 26 20:03 fich2
-rw-r--r--  1 marta     informix 1594 Sep  3 1998 Main.dt
-rw-----  1 marta     informix   424 Sep  2 1998 mbox
-rw-r--r--  1 marta     informix   648 Sep  2 1998 numped.sql
-rw-r--r--  1 marta     informix   137 Sep  2 1998 NUMPED.sql
-rw-r--r--  1 marta     informix   231 Sep  2 1998 Personal.dt
-rw-r--r--  1 marta     informix 3983 Sep  2 1998 trash.dt
-rw-r--r--  1 marta     informix   843 Sep  2 1998 ventas.sql
45% █
```

permisos    propietario grupo    tamaño    nombre  
            vínculos                              modificación

- |   |       |       |        |
|---|-------|-------|--------|
| - | rwx   | rwx   | rwx    |
|   | owner | group | others |
- fichero  
d directorio  
l vinculación simbólica  
b ó c especiales.



# S. O. Unix

---

\$chmod : para cambiar permisos en ficheros y directorios

Actúa sobre:

u Propietario

g Grupo

o Otros

– Códigos de operación

+ Otorga el permiso

- Deniega el permiso

– Permisos:

r	Lectura	4
w	Escritura	2
x	Ejecución	1
	Sin permiso	0



# S. O. Unix

---

`$chmod o-rwx fich1`

: quita todos los permisos a los usuarios que no pertenecen al grupo del fichero llamado fich1.

`$chmod 777 dir1`

: otorga todos los permisos a todo el mundo (suma 4+2+1)

`$chmod 700 dir1`

: otorga permisos al propietario



# S. O. Unix

---

`$chown nombre_usuario fich1` : cambia de propietario al fichero fich1

`$chgrp nombre_usuario fich1` : cambia de grupo al fichero fich1

**umask:** permite especificar los permisos de todos los ficheros y directorios que se creen después de ejecutar esta orden. Para cambiar los permisos se le indica a umask que reste de 777.

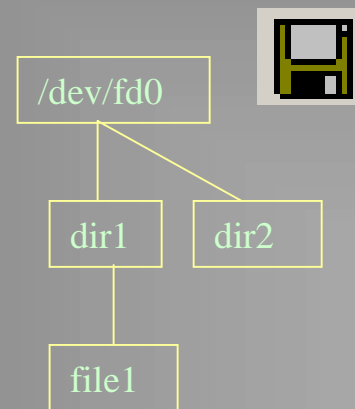
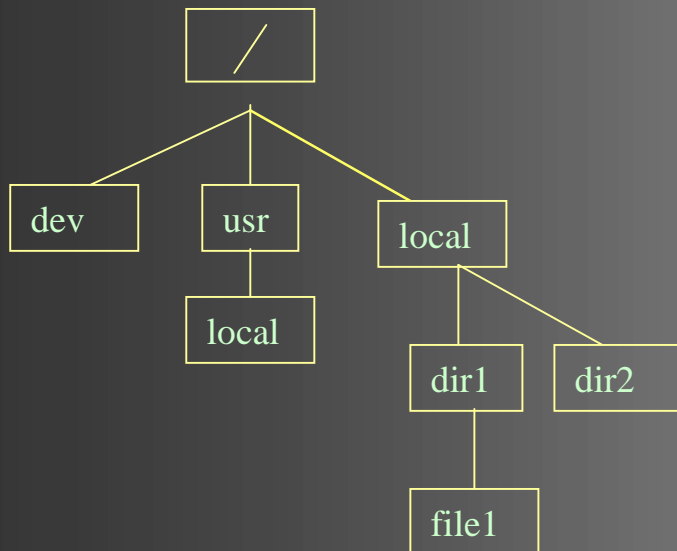
`$umask 077` : todos los ficheros tendrán permiso únicamente para el propietario.



# S. O. Unix

Permite añadir un sistema de ficheros a un directorio de otro sistema de ficheros:

`$mount /dev/fd0 /local` : integra el floppy disk en el sistema de archivos bajo el directorio `/local`.





# S. O. Unix

---

- Operadores de redirección (<,>,>>):

\$ls > fich1

: crea fichero fich1 con el resultado del comando ls

\$cat fich1 fich2 >> fich3

: concatena fich1 y fich2 y añádelo al final de fich3



# S. O. Unix

- Pipes o tuberías (|)

- Es posible conectar dos o más procesos de tal forma que la salida de un proceso se use como la entrada de otro proceso.

`$comando1 | comando2 | ... | comandoN`

- Ej: `$who | wc -l` : cuanta los usuarios que están conectados

- Se evita el uso de ficheros intermedios
- El comando **tee** permite derivar una copia de los datos pasados entre comandos a un fichero sin modificar la acción pipe.

- Ej: `$who | tee fich_datos | wc`





# S. O. Unix

- **Búsqueda:**

**find:** Busca ficheros que coinciden con un patrón dentro de subdirectorios:

- name fich : con nombre fich
- size n : con tamaño de n bloques
- mtime n : ficheros modificados hace n días
- print : imprimir la ruta
- exec comando : ejecuta el comando una vez localizado el fichero, debe ir seguido de {} que reemplaza al path en curso y la secuencia de escape \
- ok comando : igual al anterior pero antes de ejecutar pregunta al usuario
- type f : buscar ficheros
- type d : buscar directorios
- xargs : construye y ejecuta comandos



# S. O. Unix

---

- **Búsqueda:**

## Ejemplos:

```
$find /usr/marta -name 'f*' -print
```

busca a partir del directorio /usr/marta los ficheros que comienzan por f e imprime las ruta de los ficheros encontrados en pantalla

```
$find . -name '*.ec' -print -exec rm {} \
```

borra los ficheros con extensión ec del directorio actual

```
$find . -type f -print | xargs grep texto
```

localiza los ficheros del directorio actual que contenga la palabra 'texto'



# S. O. Unix

**grep:** Busca patrones de texto en ficheros y muestra el nombre de los ficheros que contienen el patrón.

- b antecede cada línea localizada con el número de bloque en el que se encuentra
- n antecede cada línea con el número de línea en el que se encuentra
- c imprime el nº de líneas que coinciden con el patrón de búsqueda
- v imprime todas las líneas que no coinciden con el patrón de búsqueda
- i ignora mayúsculas y minúsculas

`$grep palabra fich1` : busca el texto palabra en el fichero fich1



# S. O. Unix

- **Ordenación:**

sort: ordena el contenido de ficheros alfabéticamente, numéricamente o por diferentes campos.

- b : ignora blancos iniciales
- f : no distingue mayúsculas de minúsculas
- n : clasifica por orden numérico
- o fich1 : guarda el resultado en fich1
- r : clasifica en orden inverso
- u : suprime líneas repetidas

`$sort -fr -o ordenado lio`

guarda en ordenado el resultado de la ordenación del fichero lio en orden inverso, no diferenciando mayúsculas de minúsculas.

`$sort +1 lista` : ordena lista por el primer campo (comienza en cero)

+1 : salta al primer campo

-2 : parar justo antes de comparar el segundo

+0 : realizar una segunda pasada y ordenar a partir del inicio de la línea actual



# S. O. Unix

---

- Otros comandos útiles

**alias** permite abreviar la ejecución de un comando  
\$alias ls 'ls -al'

**compress** fich1 permite comprimir un fichero  
**uncompress** fich1.Z (extensión .Z)

**diff** determina diferencias entre dos ficheros o directorios  
-b ignora blancos  
\$diff fich1 fich2

**wc** fich1 cuenta líneas, palabras y caracteres que contiene el fich1



# S. O. Unix

---

**more**

paginador

ls -al | more

**du**

imprime el nº de bloques (512 bytes) de cada fichero y directorio

**echo**

imprime en pantalla una cadena de caracteres  
\$echo escribe esto en pantalla

**clear**

limpia la pantalla

**tar**

guarda y extrae archivos de ficheros en disco

tar -cvf archivo1.tar \* : crea archivo1.tar con todos los  
ficheros del directorio actual

tar -xf archivo1.tar : recupera los ficheros contenidos  
en archivo1.tar



# S. O. Unix

- **Manuales**

- Los comandos en Unix están descritos en línea en una colección de ficheros conocidos como **man pages**.

Hay ocho secciones:

- |                                |                                     |
|--------------------------------|-------------------------------------|
| 1 : Comandos - C               | 5 : Misceláneos - M                 |
| 2 : Llamadas al sistema - S    | 6 : Juegos - F                      |
| 3: Librerías de funciones - CP | 7 : Archivos especiales - HW        |
| 4: Formatos de archivos - CT   | 8 : Mantenimiento del sistema - ADM |

- Para ver la explicación de un comando:

`man [sección] nombre_comando`

`man C cp`

- Para buscar un tema dentro de las cabeceras de las páginas del manual:

`apropos copy`

`man -k copy`



# S. O. Unix

- **Shell:**

- Intérprete de comandos, es decir, la interfaz entre el usuario y el SO. Traduce las órdenes dadas por el usuario a instrucciones en la sintaxis interna del sistema.
- Hay diferentes shells:
  - Bourne shell    /bin/sh            .profile
  - C shell         /bin/csh            .cshrc    .login
  - Korn shell      /bin/ksh            .profile .login
- La ksh y csh son ampliaciones de la sh para proporcionar historia de órdenes, edición de línea de órdenes, alias, control de trabajos, protección de archivos frente a sobreescritura y características de comodidad.
- Para conocer el shell, se utiliza el comando:  
echo \$SHELL





# S. O. Unix

- El entorno ksh guarda en memoria los comandos introducidos (recuperar con `esc k`) y es muy compatible con sh del Unix Sistema V, por ello muchos usuarios la prefieren.
- Se puede cambiar de shell con `$sh`, `$csh`, `$ksh`
- Las características del entorno se definen por las variables de entorno:
  - `HOME` directorio del usuario
  - `LOGNAME` nombre del usuario
  - `PATH` directorios de búsqueda de comandos
  - `TERM` tipo de terminal
  - `PS1` define prompt primario



# S. O. Unix

---

<b>env</b>	visualiza variables de entorno
<b>echo \$PATH</b>	visualiza el valor de la variable PATH
<b>history</b>	visualiza la últimas instrucciones ejecutadas
<b>!num_hist</b>	ejecuta el comando con nº num_hist (csh)
<b>history -num_hist</b>	ejecuta el comando con nº num_hist (ksh)



# S. O. Unix

## Fichero .profile

```
HOME=/home/$LOGNAME
```

```
PATH=$PATH:/bin:/usr/bin
```

```
EDITOR=vi          export EDITOR
```

```
TERM=vt100         export TERM
```

```
umask 066
```

```
stty erase "^H" kill "^U" intr "^C" eof "^D"  
tabs
```

```
# Cierre de todos los procesos asociados al terminal al
```

```
# recibir interrupción
```

```
set -u
```

```
trap "echo 'logout'" 0
```

```
# evitar sobre escribir ficheros
```

```
set noclobber
```

```
#registro de comandos ejecutados
```

```
set history=20
```



# S. O. Unix

## Fichero .cshrc

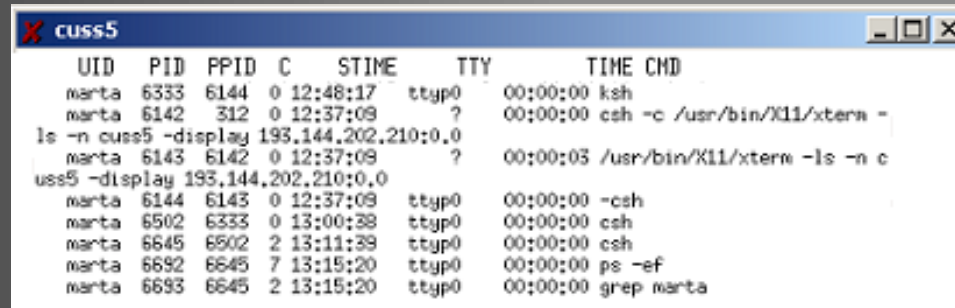
```
set noclobber                # don't allow '>' to overwrite
set history=20               # save last 20 commands
if ($?prompt) then
    set prompt=\\!%\\        # set prompt string
alias print 'pr -n \\!:* | lp' # print command alias
setenv INFORMIXDIR /u/informix9
setenv ONCONFIG onconfig.ows
setenv INFORMIXSERVER macc_bd1
setenv INFORMIXSQLHOSTS /u/informix7/etc/sqlhosts
setenv INFORMIXTERM terminfo
setenv TERMINFO /usr/share/lib/terminfo
setenv PATH ./bin:/u/informix7/bin:./usr/ccs/bin:/usr/ccs/lib:/bin
```



# S. O. Unix

- **Procesos**

- Un proceso es un programa en memoria que se está ejecutando. Puede ser un comando simple, un conjunto de ellos o un programa.
- A cada proceso, el sistema le asigna un identificador denominado PID.



UID	PID	PPID	C	STIME	TTY	TIME	CMD
marta	6333	6144	0	12:48:17	ttyp0	00:00:00	ksh
marta	6142	312	0	12:37:09	?	00:00:00	csh -c /usr/bin/X11/xterm -
ls -n cuss5	-display	193.144.202.210:0.0					
marta	6143	6142	0	12:37:09	?	00:00:03	/usr/bin/X11/xterm -ls -n c
uss5	-display	193.144.202.210:0.0					
marta	6144	6143	0	12:37:09	ttyp0	00:00:00	-csh
marta	6502	6333	0	13:00:38	ttyp0	00:00:00	csh
marta	6645	6502	2	13:11:39	ttyp0	00:00:00	csh
marta	6692	6645	7	13:15:20	ttyp0	00:00:00	ps -ef
marta	6693	6645	2	13:15:20	ttyp0	00:00:00	grep marta

UID    Usuario

PPI    ID del proceso

PPID   ID del proceso padre (1 es el padre de todos los procesos)

C    cantidad de recursos de CPU (los más bajos entran antes al scheduler)

STIME   hora a la que se inició el proceso

TTY    terminal

TIME   tiempo que llevó la ejecución

COMMAND   comando



# S. O. Unix

- Un proceso puede ejecutarse en background o foreground. Para ejecutar en background se añade el carácter & al final de la línea del comando (sólo ksh).

```
$du | sort > disco_uso &
```

[1] 6543 : devuelto por la shell indicando nº de trabajo y nº de proceso.

\$fg %num\_trabajo : se trae a la pantalla el nº de trabajo lanzado en background. Para llevarlo a background, se suspende el proceso (Ctrl-Z) y se escribe:

```
$bg
```

- Para eliminar un proceso se utiliza el comando **kill** junto con el pid. Para conocer PID comando **ps**.

```
$ps -ef|grep marta : busca procesos en curso del usuario marta
```

```
$kill n_pid : mata el proceso n_pid
```

```
$kill -9 0 : mata todos los procesos de un tty
```



# S. O. Unix

---

- El comando `jobs` lista los procesos en curso.

`$jobs` : muestra entre corchetes el número del proceso del usuario. El signo `+` indica proceso en curso y `-` indica un proceso previo. `Done` y `Running` indica el estado del proceso. Se puede suspender un proceso con `Crtl-Z`. Solo disponible en `ksh`.

- Cada proceso abre tres ficheros estándar:
  - entrada (`stdin`), de donde lee datos de entrada. Por defecto, teclado
  - salida (`stdout`), donde escribe el resultado. Por defecto, pantalla.
  - errores (`stderr`), donde escribe mensajes de error. Por defecto, pantalla



# S. O. Unix

---

- A los procesos se les puede asignar prioridad. El comando **nice** reduce la prioridad en 10 entre 0 y 19 del planificador (scheduler) arbitrario.

```
$nice -14 fichero.sh
```

- Se pueden tener procesos en ejecución aún cuando se cierre la sesión. Comando **nohup** para que al cerrar la sesión tome el padre (pid 1 ) el control.

```
$nohup fichero.sh &
```





# S. O. Unix

- Señales

Evento que interrumpe la ejecución normal de un proceso.

- Puede ser originada por:
  - Sistema operativo
  - Propio proceso
  - Otro proceso
  - El usuario
- Señales predefinidas por POSIX
  - `SIGINT`      <ctrl> C interrupción por teclado
  - `SIGHUP`      conexión de comunicación rota
  - `SIGFPE`      excepción de coma flotante
  - `SIGBUS`      error de bus, violación de memoria
  - ...



# S. O. Unix

---

- Administración del S.O. Unix

- su o root                      prompt #  
  homepath /
- orden **su**: conmuta de estado de un usuario normal a superusuario.
- creación de mensajes y noticias: **motd**, **wall**



# S. O. Unix

- tareas del admon ( **sysadm**)
  - copias de seguridad
  - cambio de contraseña
  - hora y fecha
  - operaciones de disco
  - operaciones de sistemas de fichero (crear, montar y desmontar)
  - preparación del correo
  - preparación de periféricos (impresora, puerto serie,...)
  - operaciones de impresora ( cola, status,...)
  - restauración de medios extraíbles
  - información del sistema (**uname**)
  - información de usuarios
  - gestión de software.



# S. O. Unix

- **Arranque de la máquina**

- primary boot-strap : carga de la ROM la primera parte del SO del disco. De hecho, carga el programa cargador que a su vez carga el SO Unix.

Estados 'init'

- desconectar 0
  - monousuario 1
  - multiusuario 2
  - xxx
  - xxx
  - desconexión y arranque 5
- fichero **/etc/inittab**: contiene los procesos que deben estar en ejecución. Se arrancan con **init**.
  - El proceso init se está ejecutando constantemente, se lee solo al arrancar la máquina. Si se hacen modificaciones se debe recargar.



# S. O. Unix

---

- Desconexión de la máquina

`shutdown -i0 -y -g0`

-i0: llevarlo al estado 0

-y: sin confirmación

-g0: intervalo de tiempo 0

– comandos **who** y **ps -ef** dicen la actividad del sistema



# S. O. Unix

- Seguridad

- virus e instalación de productos
- permisos a nivel de usuario, grupo y otros (**umask**)
- cifrado de ficheros crypt y pack
- contraseñas /etc/passwd (**passwd**)
- limitar la shell del usuario poniendo permisos para el admon y órdenes como limitar moverse de su HOMEPATH, ejecutar programas, etc...
- password de arranque de la máquina



# S. O. Unix

- Seguridad

Detección de ataques al sistema:

- script que recorra todos los ficheros que han cambiado de permisos y programas que han crecido mucho
- contolar cambios en /etc/initab y en /etc/profile y en los ficheros cron, /bin/login, /bin/getty
- caballo de troya: alteración de parte clave del software que generalmente se tiene ya copiado en back-ups. Se aconseja reinstalar.



# S. O. Unix

- **Temporización y planificación**

- La planificación de tareas es una parte importante de cualquier SO multitarea.
- La temporización es tan importante en Unix que casi todas las máquinas tienen en hardware reloj/calendario interno.

`at -f scrip.sh 6:00 am Friday` : ejecutar el fichero script.sh  
a las 6:00 am del viernes

`at -l` : ver cola de trabajo

`batch < script.sh` : se ejecuta el trabajo cuando el  
sistema reconoce baja carga





# S. O. Unix

- cron demonio que se despierta cada minuto y mira si hay algún proceso que ejecutar. Planifica trabajos para hacerlos diariamente, semanalmente o mensualmente.

/usr/lib/cron: directorio que contiene los ficheros de permisos de quien puede ejecutar tareas en el cron. Solo lo modifica el admon.

- cron.allow y cron.deny ficheros con los usuarios que pueden utilizar el cron y que no se les permite, respectivamente.

/usr/spool/cron/crontabs: contiene los ficheros crontab del sistema.

\$crontab cron\_file : carga el fichero al cron

Formato cron\_file:

min	hora	dia	mes	dia_sem	tarea
(0-59)	(0-23)	(1-31)	(1-12)	(0-6)	



# Editor vi

---

- Editor críptico, interactivo y visual, aprovecha toda la pantalla del terminal para desplegar el texto que se está editando.
- No es un programa de formateo de texto (justificar márgenes, centrar títulos,...). Para ello habría que utilizar el nroff.
- El vi lo incorporan la mayoría de los sistemas operativos Unix.
- Buffer de trabajo: vi lee el contenido de un fichero y hace una copia. Sobre ésta trabaja y al salir del editor guardando, vuelca los cambios al fichero del disco.



# Editor vi

---

- Modos de operación:
  - modo comando - vi acepta como órdenes la información tecleada
  - modo texto - vi acepta como texto la información tecleada
- ▶ Al iniciar, vi está en modo comando.
- ▶ Cambio de comando a modo texto: `<escape>`
- ▶ Volver a modo comando : `<escape><escape>`



# Editor vi

---

- Crear o editar fichero  
\$vi nomb\_fich
- Comandos más útiles:
  - Para introducir texto:
    - esc i : inserta en la posición del cursor
    - esc a : inserta en la posición siguiente del cursor
    - esc o : abre una línea por debajo del cursor
    - esc O : abre una línea por encima del cursor
    - esc I : inserta a principio de la línea
    - esc A : inserta al final de la línea



# Editor vi

---

– Para borrar texto:

- `esc x` : borra carácter situado en la posición del cursor
- `esc dd` : borra la línea donde está situado el cursor
- `esc d)` : borra el resto de la frase
- `esc d}` : borra el resto del párrafo
- `esc dw` : borra la palabra
- `esc D` : borra hasta el final de la línea

todos estos comandos pueden ir precedidos por un entero que indique el número de caracteres, palabras, líneas, etc. a los que debe afectar.



# Editor vi

## – Para desplazar texto:

- `esc yy` : copia la línea y la coloca en el buffer
- `esc p` : coloca después del cursor el fragmento copiado en el buffer
- `esc P` : coloca antes del cursor el fragmento copiado en el buffer
- `esc "cyy` : copia la línea en el buffer denominado c, donde c puede ser una letra de a-z
- `esc "cp` : pega el contenido del buffer c después del cursor.



# Editor vi

## – Para alterar texto:

- `esc r` : reemplaza el carácter donde está situado el cursor
- `esc R` : reemplaza el texto anterior a partir de la posición del cursor
- `esc cw` : cambia la palabra a partir del carácter donde está situado el cursor
- `esc c}` : cambia la frase a partir del carácter donde está situado el cursor
- `esc J` : une la línea siguiente a aquella en la que está el cursor
- `esc cc` : reemplaza la línea
- `esc C` : reemplaza hasta final de línea



# Editor vi

– Para guardar texto y abandonar el editor:

- `esc :w` : guarda los cambios realizados hasta el momento
- `esc :q` : sale del editor sino se han realizado cambios
- `esc :q!` : sale del editor sin guardar cambios
- `esc :wq` : sale del editor y guarda los cambios
- `esc ZZ` : sale del editor y guarda los cambios
- `esc :n, k w fich2` : escribe las líneas n a k en el fichero fich2
- `esc :n, k w >> fich2` : añade las líneas n a k en el fichero fich2





# Editor vi

## – Para desplazamiento del cursor:

- `esc j` : baja el cursor de línea
- `esc k` : sube el cursor de línea
- `esc l` : desplaza el cursor a la derecha
- `esc h` : desplaza el cursor a la izquierda
- `esc <control-d>` : desplaza media página hacia abajo.
- `esc <control-u>` : desplaza media página hacia arriba.
- `esc <control-b>` : retrocede página completa
- `esc <control-f>` : avanza página completa
- `esc nG` : desplaza el cursor hasta la línea n del fichero
- `esc <control-l>` : redibuja la pantalla



# Editor vi

- Para buscar:
  - `esc /patron` : busca la siguiente aparición de patrón
  - `esc ?patron` : busca la anterior aparición de patrón
  - `esc n` : repite el último comando de búsqueda
  
- Para deshacer:
  - `esc u` : deshace el último comando
  - `esc U` : deshace todos los cambios en la línea actual
  
- Para resaltar en pantalla:
  - `esc :set redraw` : mantiene actualizada la pantalla en todo momento
  - `esc :set number` : muestra los números de línea
  - `esc :set nonumber` : desactiva los números de línea



# Correo electrónico

- Correo electrónico

Leer correo: **mail** o **mailx**

<return> o +	: lee el siguiente mensaje
-	: vuelve al mensaje anterior
p	: imprime de nuevo el mensaje
s fich	: salva el mensaje en fich (fichero por defecto mbox)
dp	: borra el mensaje y da paso al siguiente
q	: abandona mail
r usuario	: responder a usuario
?	: lista los comandos disponibles



# Correo electrónico

- Enviar correo: **mail**

```
mail nombre_login o mailx nombre_login  
aquí va el texto  
<ctrl-d>
```

```
mail nom1 nom2 nomN < fich : se envía a los N usuarios  
el contenido del fichero
```

```
mail zorrillm@unican.es : envío de correo a otra  
máquina
```

Si la dirección o usuario de correo es desconocida el sistema te avisa enviándote un correo con el error.



# ftp (file transfer protocol)

---

- **Transferencia de ficheros con ftp**

- ftp: permite transferir ficheros desde/hacia una máquina remota.
- Se requiere conocer el login y passwd de la cuenta a la que se va a conectar.
- Sintaxis:  
ftp nombre\_máquina\_remota

Ej:

\$ftp cuss5

nombre de la máquina

\$ftp 150.0.1.51

dirección IP



# ftp (file transfer protocol)

---

- Se puede transferir en modo binario (binary) o texto (ascii).
  - ftp> ascii
  - ftp> binary
- Para transferir de máquina remota a local
  - ftp> get fichero\_remoto
- Para transferir de máquina local a remota
  - ftp> put fichero\_local
- El fichero en ambos casos se guarda en el directorio activo.



# ftp (file transfer protocol)

## Comandos útiles:

- Para conocer el directorio remoto activo  
ftp> pwd
- Para conocer el directorio local activo  
ftp> !pwd (en Windows lcd)
- Para cambiar el directorio remoto activo  
ftp> cd nombre\_dir\_remoto
- Para cambiar el directorio local activo  
ftp> lcd nombre\_dir\_local
- Para cambiar el directorio remoto activo  
ftp> cd nombre\_dir\_remoto
- Para salir de la conexión  
ftp> bye
- Para conocer los comando disponibles  
ftp> ?



# telnet (file transfer protocol)

- **Conexión remota (telnet)**

- telnet: permite conectarse a una máquina remota para ejecutar procesos en ella.
- Se requiere conocer el login y passwd de la cuenta a la que se va a conectar.

- Sintaxis:

telnet nombre\_máquina\_remota

\$telnet cuss5                      nombre de la máquina

\$telnet 150.0.1.51                dirección IP

- El sistema responde preguntando el login y passwd y, una vez aceptados, nos muestra el prompt del S.O.





# LINUX



- S.O. desarrollado por Linus Torvalds en 1994 y miles de colaboradores por medio de Internet.  
<http://www.linux.org/>
- Unix para procesadores intel.
- Compatible con Unix.
- Se reconoce por su mascota: Pingüino Linux.
- Fabricantes están adaptando sus sistemas a Linux (Oracle, Informix, Corel, Netscape,...)
- Licencia GNU: programas fuentes públicos y no se puede hacer dinero con él.
- Puede bajarse de la red : Red Hat, Debian, Slackware,...