

# **GRADO EN INGENIERÍA INFORMÁTICA**

## **Sistemas Operativos – Anexo 1**

### **Consola de comandos de Unix**

Juan Luis Vidal Mazón

*juanluis.vidal@uneatlantico.es*

# Historia

UNIX se desarrolla en 1968 en los Bell Telephone Laboratories.

Sus creadores, Ritchie y Thompson, buscaban desarrollar un SO para ordenadores mas sencillos en los que desarrollar software.

Para facilitar el trabajo con ficheros, Unix introdujo el sistema de directorios en formato "árbol".

Poco después, el ahora famoso lenguaje C se desarrolla por para UNIX. UNIX mismo se reescribe en C en 1973.

- Aproximadamente un 95% de UNIX esta escrito en lenguaje C. El resto es lenguaje ensamblador y afecta al núcleo, la parte que interacciona con el Hardware.

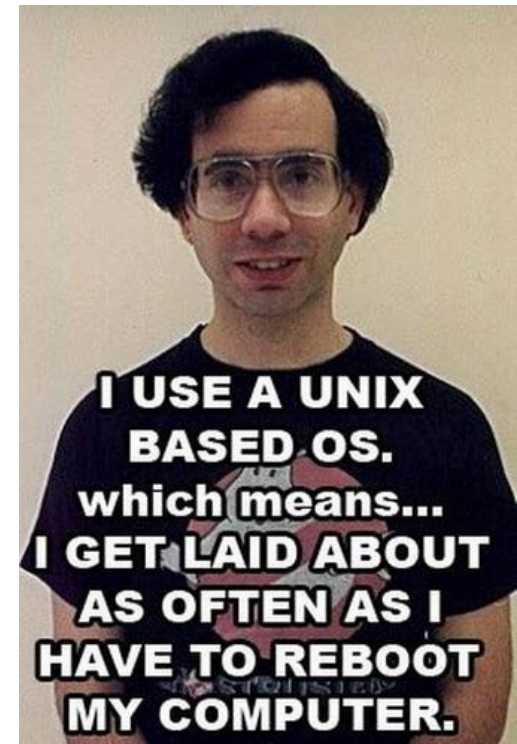
Esto trajo la portabilidad al SO, y un lenguaje de programación atractivo de usar.



## Historia (II)

Existen dos versiones principales de UNIX:

- UNIX versión V de AT&T: basado en la versión de UNIX que AT&T usaba internamente. La versión V es lanzada en 1986 y trata de fusionar lo mejor de otras versiones UNIX, como la Berkeley
- UNIX de Berkeley: desde sus comienzos el uso de UNIX ha estado muy extendido en los campus universitarios, gracias a la combinación de potencia y sencillez de manejo que ofrece. El grupo de investigación de la Universidad de California en Berkeley añadió nuevas características a UNIX en esta versión, que luego fue distribuida a otras instituciones académicas.



# Filosofía de UNIX

Comandos simples:

- UNIX se diseñó para ser un SO pequeño.
- Se diseñaron los comandos pensando en que hiciesen una sola cosa, pero que la hicieran eficientemente.
- Para desarrollar tareas mas complejas, podrían encadenarse varios comandos.
- A pesar de que ciertas distribuciones han alcanzado un tamaño considerable, estos comandos se mantienen invariables y son uno de los pilares del SO.

## Filosofía de UNIX (II)

Conectividad entre comandos:

- Para convertir comandos simples en herramientas mas potentes, el resultado de uno puede usarse como la entrada del otro.
- Se les conecta mediante los comandos de redireccionamiento  
`$ grep MINIX * | wc`

## Filosofía de UNIX (III)

Los comandos se extienden mediante parámetros opcionales:

- La funcionalidad básica de un comando puede extenderse usando los parámetros de comando.

No hay tipos de fichero asociados a una aplicación concreta:

- UNIX no presta atención a los contenidos de un fichero.
- La interpretación del contenido de un fichero la realiza cada comando encargado de manejarle.

# Estructura

Unix se compone de 3 componentes esenciales:

- Sistema de ficheros (file system). Estructura para la organización y archivo de los datos.
- Kernel. Conjunto de programas que controla la gestión de recursos y periféricos del sistema. El usuario se comunica con el Kernel a través del Shell.
- Interprete de comandos (Shell). Programa de interface entre el usuario y el kernel. Existen varios tipos de shell como el Bourne o el Cshell.

## Estructura (II)

Unix es un sistema multiusuario.

Tiene todas las características asociadas con este tipo de SOs, como el uso compartido de:

- Software
- Impresoras
- Colas de ejecución de procesos

Cada usuario posee un directorio propio al que accede al entrar al sistema.

No solo es multiusuario sino que es también multitarea (es capaz de realizar varios procesos a la vez)



## Estructura (III)

Para poder acceder a un sistema UNIX, previamente debe darse de alta cada cuenta de usuario.

Existe un “superusuario” que es quien se encarga de estas labores:

- Abre y cierra el sistema.
- Crea usuarios nuevos, los elimina o modifica.
- Hace copias de seguridad de los ficheros
- Tiene pleno acceso al sistema de ficheros (a diferencia de los usuarios normales)

Los usuarios están a su vez divididos en grupos de usuarios. Estos grupos son mantenidos también por el superusuario.

# Sistema de archivos

La estructura de directorios de UNIX tiene forma de árbol.

El directorio de acceso de cada usuario no es el raíz (como ocurriría en DOS), sino uno propio que tiene asignado.

Nombres de ficheros:

- Hasta 14 caracteres sin restricciones.
- Se recomienda no usar símbolos especiales como espacios, tabulador, \*...
- Distingue entre mayúsculas y minúsculas.

## Sistema de archivos (III)

Ficheros:

- Un fichero es un conjunto de datos asociados con un nombre.
- En UNIX existen 3 tipos principales de ficheros
  - Ordinarios
  - Directorios
  - Especiales (los dispositivos como impresoras o discos)
- Un directorio es por tanto un fichero que contiene información sobre otros ficheros.

# Comandos UNIX

## pwd

Muestra por pantalla la ruta absoluta actual en la que nos encontramos dentro del árbol de directorios de UNIX

```
sdf:/udd/j/j1> pwd  
/udd/j/j1  
sdf:/udd/j/j1> █
```

## clear

Limpia el contenido actual de la pantalla. Equivalente a la instrucción `cls` de MSDOS

# Comandos UNIX

*ls [-ltasdrv] [nombre\_dir]*

lista los contenidos de un directorio, y ofrece información sobre los ficheros

```
sdf:/udd/j/j1> ls
.          .hushlogin      nuevoDirectorio
..         .profile      prueba
.history   .signature      prueba2
sdf:/udd/j/j1> █
```

# Comandos UNIX

- Modificadores de **ls**
  - -l Formato largo
  - -t Ordenado por tiempo de acceso
  - -a Lista todas las entradas
  - -s Tamaño en bloques
  - -d Información sobre el estado del directorio
  - -r En orden inverso
  - -v Según último acceso
  - -R Lista de manera recursiva todos los subdirectorios que encuentre.

# Comandos UNIX

Ejemplo: ls -l

```
sdf:/udd/j/j1> ls -l
total 96
drwxrwx--x    3 j1  users   512 Feb 25 09:59 .
drwxr-xr-x  583 new  wheel 25600 Feb 25 04:20 ..
-rw-r--r--    1 j1  users  4520 Feb 25 10:59 .history
-rw-rw----    1 j1  users     0 Feb 18 18:18 .hushlogin
-rw-r-----    1 j1  users  2794 Feb 18 18:18 .profile
-rw-rw----    1 j1  users    76 Feb 18 18:18 .signature
drwxr-xr-x    2 j1  users   512 Feb 18 20:29 nuevoDirectorio
-rw-r--r--    1 j1  users    43 Feb 18 18:49 prueba
-rw-r--r--    1 j1  users    35 Feb 23 22:54 prueba2
-rw-----    1 j1  users    47 Feb 25 09:59 prueba_clase.save
sdf:/udd/j/j1> █
```

# Comandos UNIX

*cat [-v] nombre\_fichero*

Visualiza por pantalla el contenido de un fichero

```
sdf:/udd/j/j1> cat texto
Este es el primer fichero de prueba
sdf:/udd/j/j1> █
```

- El modificador `-v` añade el numero de cada línea al comienzo.



# Comandos UNIX

*more [-c][-num\_lineas][+numero\_linea] nombre\_fichero*

Visualiza el contenido de un fichero por pantalla página a página.

Opciones:

- c Limpia el contenido de la pantalla, no la desplaza.
- num\_lineas Muestra un número determinado de líneas.
- +numero\_linea Comienza en esta línea.

## Comandos UNIX

*mkdir nombre\_dir*

Crea un nuevo directorio.

```
sdf:/udd/j/j1/nuevoDirectorio> ls
.  ..
sdf:/udd/j/j1/nuevoDirectorio> mkdir miDirectorio
sdf:/udd/j/j1/nuevoDirectorio> ls
.  ..          miDirectorio
sdf:/udd/j/j1/nuevoDirectorio> █
```

# Comandos UNIX

## *cd nombre\_directorio*

Cambia nuestra posición actual dentro del sistema de ficheros al directorio indicado.

```
sdf:/udd/j/j1> cd nuevoDirectorio
sdf:/udd/j/j1/nuevoDirectorio> pwd
/udd/j/j1/nuevoDirectorio
sdf:/udd/j/j1/nuevoDirectorio> cd ..
sdf:/udd/j/j1> █
```

De manera similar a MS-DOS, el directorio actual se representa por ".", y el directorio padre o inmediatamente superior por "..". De esta forma, si queremos colocarnos en nuestro 'directorio padre', usaremos la instrucción `cd ..`

## Comandos UNIX

*rm [-ir] nombre\_fichero*

Elimina el fichero especificado

-i pide confirmación antes de eliminar cualquier archivo

-r Borra el directorio especificado y todos los archivos o subdirectorios que se encuentren en él.

```
sdf:/udd/j/j1/nuevoDirectorio> ls
.      ..      fichero
sdf:/udd/j/j1/nuevoDirectorio> rm fichero
sdf:/udd/j/j1/nuevoDirectorio> ls
.      ..
sdf:/udd/j/j1/nuevoDirectorio> █
```

# Comandos UNIX

*rmdir nombre\_directorio*

elimina el directorio especificado, siempre y cuando esté vacío.

```
sdf:/udd/j/j1/nuevoDirectorio> cd miDirectorio
sdf:/udd/j/j1/nuevoDirectorio/miDirectorio> cd ..
sdf:/udd/j/j1/nuevoDirectorio> rmdir miDirectorio
sdf:/udd/j/j1/nuevoDirectorio> cd miDirectorio
/usr/pkg/bin/psh[443]: cd: /udd/j/j1/nuevoDirectorio/miDirectorio - No such file
or directory
sdf:/udd/j/j1/nuevoDirectorio> █
```

# Comandos UNIX

*cp origen destino*

Copia el fichero origen en el destino especificado.

*mv origen destino*

Mueve el fichero especificado en destino al directorio y nombre de fichero especificado. Puede usarse también para renombrar un fichero.

# Comandos UNIX

## who

la orden who lista por pantalla la relación de los usuarios que están en ese instante conectados al sistema. La orden muestra el nombre de usuario, el nombre de terminal, y la hora/fecha en que dicho usuario se conectó.

## cal [-mes] [-año]

Visualiza el calendario del mes actual, o del año y mes indicados

```
sdf:/udd/j/j1> cal
      February 2006
  S   M Tu   W Th   F   S
           1   2   3   4
  5   6   7   8   9  10  11
12  13  14  15  16  17  18
19  20  21  22  23  24  25
26  27  28

sdf:/udd/j/j1> █
```

# Comandos UNIX

*bc [nombre\_fichero]*

El comando `bc` es una sencilla calculadora de línea de comandos. Los comandos pueden estar previamente guardados en un fichero: en ese caso, `bc` mostrará el resultado de las operaciones almacenadas.

Para salir de la calculadora, hemos de usar `ctrl.+D`.



# Comandos UNIX

## du

Esta orden informa del espacio total ocupado por cualquier directorio, sus subdirectorios o cada archivo.

## df

Informa del espacio libre en el sistema.

```
sdf:/udd/j/j1> df
Filesystem      512-blocks    Used   Avail  Capacity
/dev/wd0a       28861356    447048  26971240    1%
/dev/wd0d       26797180     15064  25442256    0%
ol1:/sys        70038596  18698408  47838256   28%
ol1:/udd        70010036  18379152  48130380   27%
ol1:/arpa/af    70010036  57913056   8596476   87%
```

# Comandos UNIX

## *ps [-efl]*

Muestra por pantalla un listado de los procesos que se están ejecutando en el sistema

- e Presenta información de todos los procesos, no solo los asociados al terminal actual en que nos encontremos
- f Genera un listado completo
- l Genera un listado largo

```
sdf:/udd/j/j1> ps
  PID TT  STAT      TIME COMMAND
   532 p9  Ss+    0:00.22 /bin/ksh /usr/pkg/bin/psh
19538 p9  R+     0:00.00 ps
```

## El editor VI

VI es un editor de texto orientado a pantalla que se encuentra disponible en la mayoría de las versiones de UNIX

Se disponen mas de 100 ordenes en Vi, lo que hacen de este editor una herramienta realmente potente, y una de las características mas conocidas del SO.

Vi tiene dos modos básicos de operación:

- Modo de orden: Cuando se inicia Vi, se entra en el modo de orden. En este modo, las entradas desde teclado se interpretan como ordenes, y las teclas pulsadas no se ven reflejadas en pantalla. Aquí se pueden realizar tareas como salir del editor, guardar el fichero, buscar palabras, etc. etc. etc.
- Modo inserción: En este modo el teclado se convierte en una maquina de escribir, y las teclas pulsadas no se interpretan como ordenes sino como texto a escribir dentro del archivo.

## El editor VI

Cambio de un modo a otro:

- Para entrar en el modo de inserción de texto (por ejemplo nada mas entrar en el editor) tenemos varias ordenes posibles. La mas común es la tecla “i”, que comienza la inserción justo antes de la situación actual del cursor.
- Para volver al modo de orden, basta pulsar la tecla “Esc” (escape) del teclado.

## El editor VI

Las teclas de entrada en el modo inserción:

- **i** Inserta en texto antes del carácter donde este situado el cursor.
- **I** Sitúa el texto de entrada al comienzo de la línea actual
- **a** Añade el texto después del carácter en el que esta situado el cursor.
- **A** Añade el texto al final de la línea actual
- **o** Abre una línea en blanco debajo de la actual y situa el cursor en su comienzo.
- **O** abre una línea en blanco por encima de la actual y situa el cursor en su comienzo.

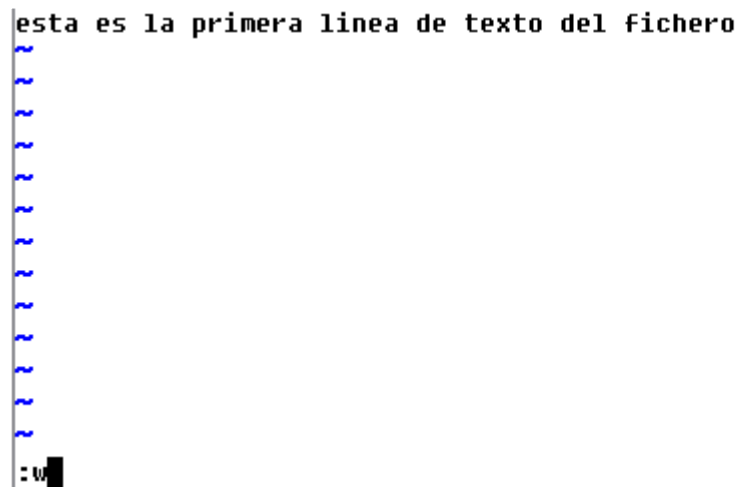
```
$ vi numero fichero
```

Para poder introducir texto, pulsamos la tecla “i”. A continuación, ya podemos escribir.

```
esta es la primera linea de texto del fichero
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
-- INSERTAR --
```

## El editor VI

Si queremos guardar el fichero, primero entraremos en el modo de orden presionando la tecla <Esc>, e introduciendo la orden :w



esta es la primera linea de texto del fichero

:w

The screenshot shows a terminal window with the text "esta es la primera linea de texto del fichero" on the first line. Below this, there are several lines of blue wavy lines representing empty space. At the bottom left, the command ":w" is entered in command mode, indicated by a black cursor block.

Para salir del editor, tenemos que introducir la orden :q

## El editor VI

Otros comandos del modo orden:

- **x** borra el carácter donde este situado el cursor.
- **1G** coloca el cursor en la primera línea del fichero.
- **nG** coloca el cursor en la línea *n* del fichero.
- **G** coloca el cursor en la ultima línea del fichero.
- **cc** sustituir una línea entera por el texto entrado a continuación.
- **ZZ** salvar el fichero y salir del editor.
- **dd** borrar la línea debajo del cursor.
- **ndd** borrar *n* líneas debajo del cursor.



# Seguridad en UNIX

Los usuarios de UNIX se encuentran divididos en grupos. Esta distribución permite establecer diferentes niveles de acceso para los diferentes ficheros, directorios o programa del sistema.

De manera general, los usuarios pueden dividirse en:

- Propietario (*u*)
- Grupo (*g*)
- Todos los demás (*o*)
- Todos los usuarios (*a*)

El acceso a un fichero puede ser con el objeto de leerle (read), modificarle (write) o ejecutarle en el caso de un programa (eXecute).

# Seguridad en UNIX

Según esto, se puede tener acceso a un fichero en los siguientes modos:

- Lectura *r*
- Escritura *w*
- Ejecución *x*

Combinando los diferentes grupos de usuarios, con estos diferentes tipos de accesos, podremos establecer los permisos de un fichero dado.

# Seguridad en UNIX

Si empleamos la orden `ls -l`, obtendremos un resultado similar a este:

```
sdf:/udd/j/jl> ls -l
total 104
drwxrwx--x   3 jl   users   512 Feb 25 18:36 .
drwxr-xr-x 588 new  wheel 25600 Feb 25 15:39 ..
-rw-r--r--   1 jl   users 10607 Feb 25 18:37 .history
-rw-rw----   1 jl   users    0 Feb 18 18:18 .hushlogin
-rw-r-----   1 jl   users  2794 Feb 18 18:18 .profile
-rw-rw----   1 jl   users   76 Feb 18 18:18 .signature
drwxr-xr-x   2 jl   users   512 Feb 25 12:34 nuevoDirectorio
-rw-r--r--   1 jl   users   35 Feb 23 22:54 prueba2
-rw-r--r--   1 jl   users   36 Feb 25 11:16 texto
sdf:/udd/j/jl> █
```

# Seguridad en UNIX

- La primera letra de la columna indica el tipo de fichero que es (una d indica directorio).
- A continuación aparecen 3 grupos de 3 caracteres cada uno.
  - El primer grupo son los permisos concedidos al propietario del fichero (en este caso jl). Una r significa permiso de lectura, una w permiso de escritura y una x permiso de ejecución. Un guión indica ausencia de permiso.
  - El usuario jl pertenece al grupo de usuarios “users”. El segundo grupo nos indica los permisos concedidos a todos los usuarios que pertenecen a dicho grupo.
  - El tercer grupo nos informa de los permisos que poseen todo el resto de usuarios, los no pertenecientes al grupo “users” al que pertenece el propietario del fichero, “jl”.

# Seguridad en UNIX

– Ejemplo:

```
-rw-r--r--  1 jl  users   35 Feb 23 22:54 prueba2
-rw-r--r--  1 jl  users   36 Feb 25 11:16 texto
```

El primer carácter del ejemplo es `-`, indicando que es un tipo de fichero ordinario.

A continuación, aparece el primer grupo de 3 caracteres, con los valores `rw-`

Por tanto el propietario del fichero tiene permiso de lectura y escritura del fichero. No aparece el permiso de ejecución `x` porque estos ficheros no son programas ejecutables.

Los dos siguientes grupos tienen la misma forma `r--`

Esto significa que tanto los usuarios del mismo grupo, como todos los demás usuarios, poseen tan solo permiso de lectura, pero no de escritura sobre los dos ficheros mostrados.

# Seguridad en UNIX

## Chmod

- Chmod es el comando del SO que nos permite asignar permisos a un fichero del que seamos propietarios
- Existen 2 maneras de asignar mediante chmod estos permisos:
  - Asignando directamente a cada usuario o grupo los diferentes permisos  
`chmod g-r prueba`
  - Usando los códigos numéricos asociados a los permisos que queremos asignar  
`chmod 777 prueba`

# Seguridad en UNIX

- Si queremos asignar a cada usuario o grupo los permisos:
  - Debemos indicar a quien se lo concedemos  
u → propietario      g → grupo del propietario  
o → los demás usuarios a → todos los usuarios
  - Debemos indicar el operador  
+ → Añade permiso    - → Quita el permiso
  - Debemos indicar qué permiso estamos modificando  
r → lectura      w → escritura    x → ejecución
  - Ejemplos:

`chmod a+w prueba`

Da derecho de escritura para el fichero prueba a todos los usuarios

`chmod o-x programa`

Quitamos el derecho de ejecución de un programa a los usuarios que no pertenezcan a nuestro mismo grupo.

# Seguridad en UNIX

- Si queremos usar los códigos numéricos:
  - Los códigos de cada permiso posible son los siguientes  
4 → lectura      2 → escritura      1 → ejecución
  - El código de autorización se representa mediante 3 cifras. La primera son los permisos asociados al propietario del fichero, la segunda al grupo al que pertenece el propietario y la tercera los permisos concedidos al resto de usuarios.  
Cada una de esas cifras se obtiene sumando los códigos de los permisos concedidos.  
Por ejemplo, si queremos dar permiso de lectura y escritura la cifra seria 6 (4+2). Si queremos dar permisos totales seria 7 (4+2+1). Si quisiéramos dar permisos de lectura y ejecución pero no de escritura, la cifra a emplear seria 5 (4+1).



# Seguridad en UNIX

- Ejemplos:

- Si hemos creado un programa y queremos tener pleno acceso a el (lectura, escritura y ejecución), y permitir al resto de los usuarios ejecutarle pero no modificarle la orden seria:

```
chmod 755 programa.out
```

- Si tenemos un documento clasificado al que no queremos que accedan nadie mas aparte de nosotros mismos, la orden sería la siguiente:

```
chmod 700 documento_topsecret
```

Si quisiéramos ahora añadir permiso solo de lectura para el resto de los usuarios de nuestro grupo, la orden es:

```
chmod 740 documento_topsecret
```

## Anexo I: Programando C en UNIX

Tal y como se ha visto antes, UNIX y el lenguaje de programación C están íntimamente ligados.

Desde el Shell de UNIX es posible crear programas en C con rapidez.

El compilador de C esta normalmente integrado por defecto en todas las distribuciones UNIX. Para invocarlo, basta ejecutar el comando `cc`, indicándole el fichero que contiene el código fuente.

# Anexo I: Programando C en UNIX

Vamos a crear un sencillo programa:

- Para ello utilizaremos vi para crear un fichero `codigo.c` con las siguientes instrucciones

```
#include <stdio.h>
main ()
{
    printf("Hola UNIX");|
}
~
~
~
~
~
~
~
~
-- INSERTAR --
```

4,22-29      Todo

## Anexo I: Programando C en UNIX

- Una vez salvado vamos a llamar al compilador de C. La sintaxis básica es la siguiente:

*cc fic\_fuente [-o nombre\_ejecutable]*

fic\_fuente es el nombre del fichero que contiene el código fuente C. Si queremos asignar un nombre al programa resultante, usaremos la opción `-o` seguida del nombre que queremos emplear. En nuestro caso:

```
cc codigo.c -o programa
```

Si el proceso de compilación termina sin errores, habrá generado un programa llamado “saludo”, que estará listo para ser invocado desde el Shell de UNIX y nos mostrará en pantalla el mensaje:

```
Hola UNIX
```



Universidad  
Europea  
del Atlántico

[www.uneatlantico.es](http://www.uneatlantico.es)