Escuela Complutense
Latinoamericana

Administración Avanzada y Redes TCP/IP en UNIX

Módulo 1:
Introducción al sistema operativo UNIX

Profesor:

Dr. Rafael Moreno Vozmediano (UCM)

Contenidos

- MÓDULO 1: Fundamentos del sistema operativo UNIX/Linux
  - **Introducción al sistema Linux**
  - Órdenes básicas de acceso al sistema
  - Organización jerárquica de ficheros
  - Órdenes sobre archivos y directorios
  - Edición de textos con **vi**
  - Características generales de los shells
  - El Bourne Again shell
  - Órdenes avanzadas para el tratamiento de archivos

Introducción

¿Qué es LINUX?

- Una versión del sistema operativo Unix para PC

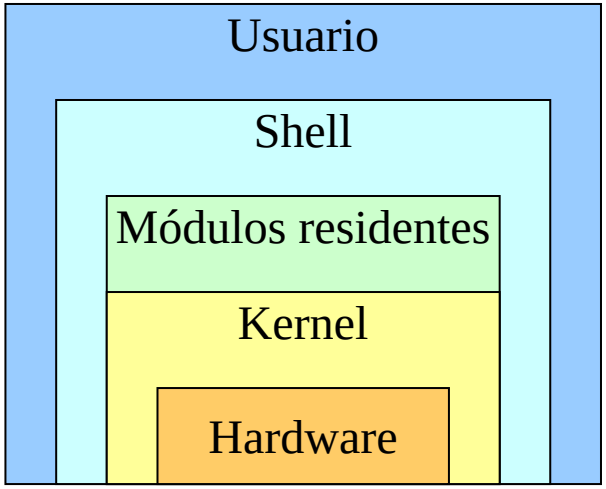
Funciones del sistema operativo

- **Gestionar los recursos** del computador: CPU, memoria y periféricos
  - Reparte el **tiempo de CPU** entre los distintos procesos
  - Supervisa y gestiona la **asignación de memoria**
  - Coordina la **utilización de dispositivos periféricos**
- **Actua como interfaz** entre el hardware del sistema y el usuario
  - Proporciona una serie de órdenes o comandos al usuario para poder usar el hardware del sistema
    - ✓ Órdenes para ejecutar programas
    - ✓ Órdenes para ver, crear, modificar o borrar archivos y directorios
    - ✓ Órdenes de acceso a periféricos: impresión, conexión a la red, acceso a cintas, ...

Características principales del sistema operativo UNIX/LINUX

- Multitarea o multiprogramado
  - Capaz de ejecutar más de un proceso o tarea simultáneamente
  - El S.O. reparte los recursos del sistema entre los distintos procesos
- Multiusuario y multiterminal
  - Pueden existir varios usuarios usando simultáneamente el computador
  - El S.O. proporciona servicios de forma concurrente a todos los usuarios
- Sistema de memoria virtual
  - Los programas se dividen en porciones pequeñas denominadas páginas
  - Las páginas de un proceso activo residen en memoria secundaria: área de swap
  - Sólo se llevan a memoria principal las páginas necesarias en un instante dado
  - La memoria virtual permite:
    - ✓ Compartir de forma eficiente la memoria principal entre todos los procesos
    - ✓ Ejecutar programas con grandes requisitos de memoria

Organización en niveles del S.O. UNIX/LINUX



- Núcleo o Kernel del sistema operativo
  - Controla y gestiona el funcionamiento del HW
  - Controla las transferencias de información entre los programas y el hardware
- Módulos residentes
  - Drivers para dispositivos
  - Componentes software para la realización de tareas específicas
    - ✓ Módulos para acceder a los distintos sistemas de ficheros, gestión de la memoria virtual y la comunicación entre procesos, gestión del entorno gráfico, etc.
- Shell o intérprete de órdenes
  - El shell actúa como interfaz entre el usuario y el Kernel
    - ✓ Acepta las órdenes que el usuario introduce en la línea de órdenes
    - ✓ Interpreta estas órdenes y las pasa al kernel

Introducción

Historia del sistema operativo LINUX

- Creador: Linus Torvalds
- Surge como un sustituto de Minix, S. O. similar a Unix para PC.

Historia del sistema operativo UNIX

- Surge en los Laboratorios Bell de AT&T en 1969
  - Programadores: Ken Thomson y Dennis Ritchie
  - S.O. flexible y compatible con las necesidades de los programadores
- En 1975 los Laboratorios Bell pusieron el sistema UNIX a disposición de las universidades de EEUU
  - En la Universidad de Berkeley se realizaron una serie de modificaciones significativas al código del S. O.
    - ✓ Principal aportación: incorporación del software de red que permitió un funcionamiento eficiente del S. O. en un entorno de red de área local (LAN)
  - Esto dio lugar a la distribución UNIX BSD

Introducción

- En la actualidad existen dos versiones principales del sistema UNIX
  - UNIX System V de AT&T
  - UNIX BSD
- Estandarización de UNIX
  - En la actualidad la versión más estandarizada de UNIX es la SVR4 (System V Release 4)
- Distribuciones comerciales del sistema operativo UNIX
  - AIX de IBM
  - HP-UX de HP
  - ULTRIX de DEC
  - Solaris de SUN (anteriormente SunOS)
  - SCO, FreeBSD, Xenix (Microsoft)
  - LINUX. Varias distribuciones:
    - ✓ Suse
    - ✓ Red Hat
    - ✓ Debian

Introducción

Ventajas del sistema operativo UNIX

- Sistema universal, válido para toda clase de computadores
- Transportable, al estar escrito en C
- Sistema abierto: las especificaciones son públicas
- Ha recogido contribuciones de múltiples personas e instituciones
- Aprovecha de forma muy eficiente los recursos del sistema

Desventajas del sistema operativo UNIX

- Dificultad en la administración
  - Las herramientas de administración del entorno de ventanas han introducido notables mejoras en este aspecto
- No está optimizado para un tipo de computador en particular
  - Puede presentar un rendimiento inferior que otros sistemas operativos específicos

Ventajas del sistema operativo Linux

- Sistema operativo tipo UNIX que corre en plataforma PC
  - Requisitos mínimos recomendados: Pentium 160 Mhz; RAM: 32 Mb; Disco: 1,2 Gb
- Se puede obtener de forma gratuita a través de Internet
  - Precio versión CD-ROM: entre 50 y 100 euros
- Distintas distribuciones
  - Red Hat, Suse, Debian, Mandrake
- Sistema operativo estable y robusto
  - Existen multitud de aplicaciones que corren en Linux sin ningún problema:
    - ✓ Communicator (Netscape), Oracle (Oracle), Wordperfect (Corel)
  - Es prácticamente imposible que el PC se quede "colgado"

Ventajas del sistema operativo Linux

- Soporta distintos tipos de interfaces gráficas (XFree)
  - KDE, GNOME, ENLIGHTENMENT, AFTERSTEP, etc.
- Existe gran cantidad de software de aplicación de libre distribución para Linux
  - Entornos de programación, software de red, aplicaciones ofimáticas, etc.
- Algunas grandes compañías van a sustituir sus sistemas UNIX por Linux
  - IBM y HP van a reemplazar sus sistemas operativos (AIX y HP-UX) por Linux
- Oracle, Dell y Red Hat han llegado a un acuerdo para ofrecer servidores Linux capaces de trabajar con bases de datos

Contenidos

- MÓDULO 1: Fundamentos del sistema operativo UNIX/Linux
  - Introducción al sistema Linux
  - **Órdenes básicas de acceso al sistema**
  - Organización jerárquica de ficheros
  - Órdenes sobre archivos y directorios
  - Nociones básicas sobre usuarios y grupos
  - Permisos de archivos y directorios
  - Edición de textos con **vi**
  - Características generales de los shells
  - El Bourne Again shell
  - Órdenes avanzadas para el tratamiento de archivos

Acceso al sistema

Cuenta de usuario

- Para acceder a un sistema UNIX es necesaria una cuenta de usuario
- La cuenta de usuario engloba los siguientes elementos
  - Nombre de usuario (Login)
  - Contraseña (Password)
  - Directorio de trabajo (Home directory)
  - Intérprete de órdenes (Shell)
- Para acceder a nuestra cuenta de usuario
  - Es necesario introducir el nombre de usuario y la contraseña
  - Automáticamente nos ubicamos en nuestro directorio de trabajo
    - ✓ Un usuario suele tener permiso total de acceso a todos los archivos y subdirectorios de su directorio de trabajo
    - ✓ El acceso a otros directorios que no pertenezcan al directorio de trabajo del usuario suele estar limitado o incluso prohibido

La cuenta del superusuario

- El superusuario es un usuario especial que actúa como administrador del sistema
  - Tiene acceso a todos los archivos y directorios del sistema
  - Tiene capacidad para crear o eliminar usarios
  - Puede detener cualquier proceso que se esté ejecutando en el sistema
  - Tiene capacidad para detener y reiniciar el sistema
- El **login** del superusuario suele ser **root**, aunque no es estrictamente necesario

Cambio de la contraseña

- La contraseña sirve para proteger nuestra cuenta de usuario de accesos no deseados
  - Sirve para proteger nuestro directorio HOME y todos sus contenidos
  - Para cambiar la contraseña se utiliza la orden **passwd**

passwd cambia la contraseña (password) del usuario

Sintaxis: **passwd**

```
prompt> passwd
Changing password for inma
(current) UNIX password:
New password:
Retype new password:
passwd: all authentication tokens updated successfully
```

Seguridad en la elección de la password

- Cuenta sin password: vía de acceso al sistema más sencilla para un usuario desautorizado
- Todas las cuentas deben tener una password
- Las passwords deben ser seguras
- Recomendaciones para elegir passwords seguras:
  - No utilizar el login como password (ni tal cual, ni invertido, ni en mayúsculas, etc.)
  - No utilizar el nombre real del usuario, ni los apellidos
  - No utilizar el nombre de los hijos ni del cónyuge
  - No utilizar ninguna información personal que se pueda obtener fácilmente (nº de DNI, nº de teléfono, matrícula del coche)

Acceso al sistema

- Recomendaciones para elegir passwords seguras:
  - No utilizar palabras contenidas en diccionarios
  - No utilizar una password con menos de seis caracteres
  - Utilizar passwords con caracteres mezclados en mayúscula y en minúscula
  - Utilizar passwords que contengan algunos caracteres no alfabéticos (números, signos de puntuación, etc.)
  - Utilizar passwords que sean sencillas de recordar, para que no sea necesario tenerlas apuntadas
  - Cambiar las passwords de forma frecuente (por ejemplo, una vez al mes)

Cambiar a otro usuario

su pasa a ser un usuario distinto (switch user)

Sintaxis: **su [- *username*]**

[- *username*] Pasa a ser el usuario especificado.
Si se omite pasa a ser superusuario

```
prompt> su – alum1
```

Terminar la sesión

exit termina la sesión

Sintaxis: **exit**

```
prompt> exit
```

logout termina la sesión

Sintaxis: **logout**

```
prompt> logout
```

Contenidos

- MÓDULO 1: Fundamentos del sistema operativo UNIX/Linux
  - Introducción al sistema Linux
  - Órdenes básicas de acceso al sistema
  - **Organización jerárquica de ficheros**
  - Órdenes sobre archivos y directorios
  - Edición de textos con **vi**
  - Características generales de los shells
  - El Bourne Again shell
  - Órdenes avanzadas para el tratamiento de archivos

Organización jerárquica de ficheros

Archivos

- **Normales**: texto, código C, órdenes shell, ejecutables, ...
- De **directorio**: Contienen nombres y punteros a los archivos y subdirectorios de ese directorio
- **Especiales**: Representan discos, terminales o impresoras (/dev)

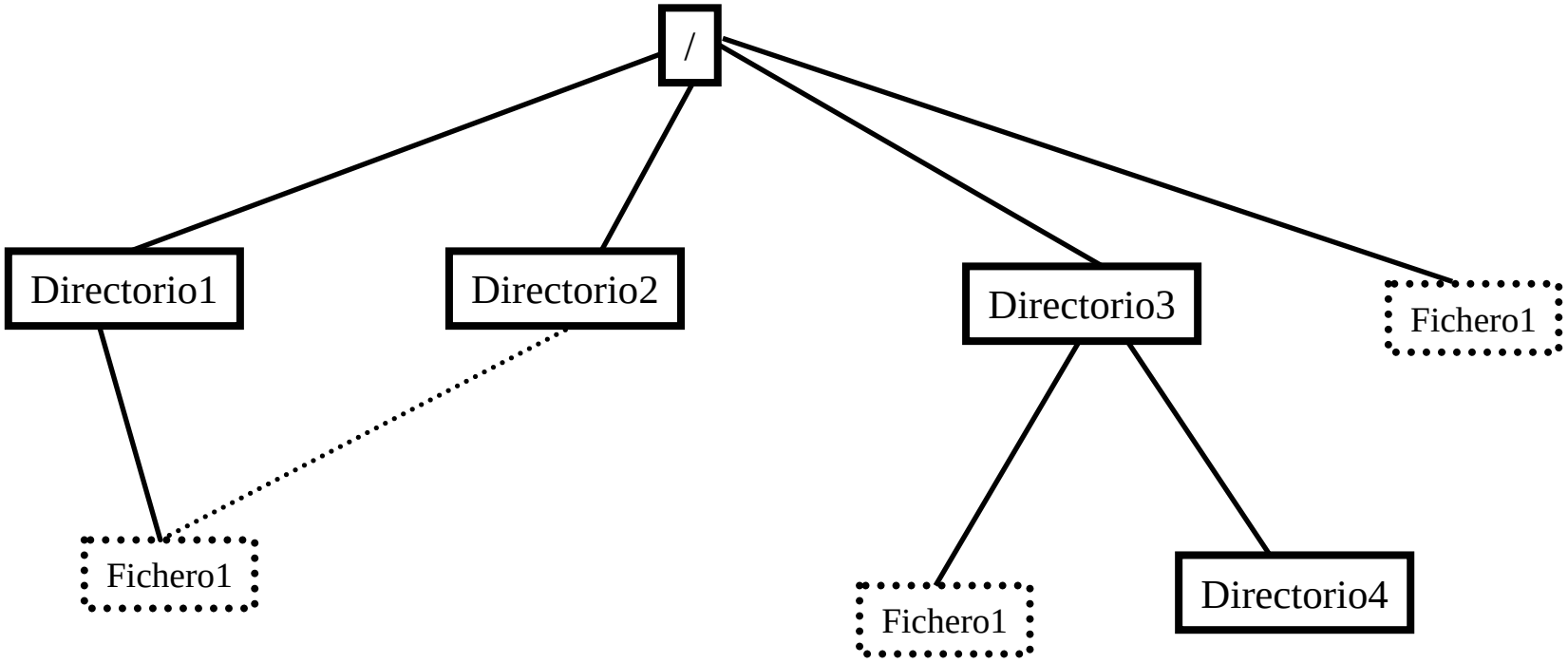
- A cada archivo se le asigna un número llamado **inodo**
  - Contiene toda la información sobre el archivo
    - ✓ Dirección de los datos en el disco
    - ✓ Tipo de archivo

Enlaces

- Todos los sistemas UNIX permiten realizar enlaces (links) de un archivo
  - Un enlace consiste en crear un nombre alternativo para un determinado archivo
  - Podemos acceder a un mismo archivo usando cualquiera de los enlaces existentes

Organización jerárquica de ficheros

- Estructura en árbol única
  - Un sólo directorio raíz (/)
  - Los directorios se tratan como ficheros que contienen descripciones de otros ficheros
  - Los discos se ven como ficheros del árbol
  - Posibilidad de realizar enlaces entre archivos (**links**)



Organización jerárquica de ficheros

Rutas de acceso

- Pueden especificarse de dos modos
  - **Ruta absoluta**: Tomando como punto de partida el directorio raíz

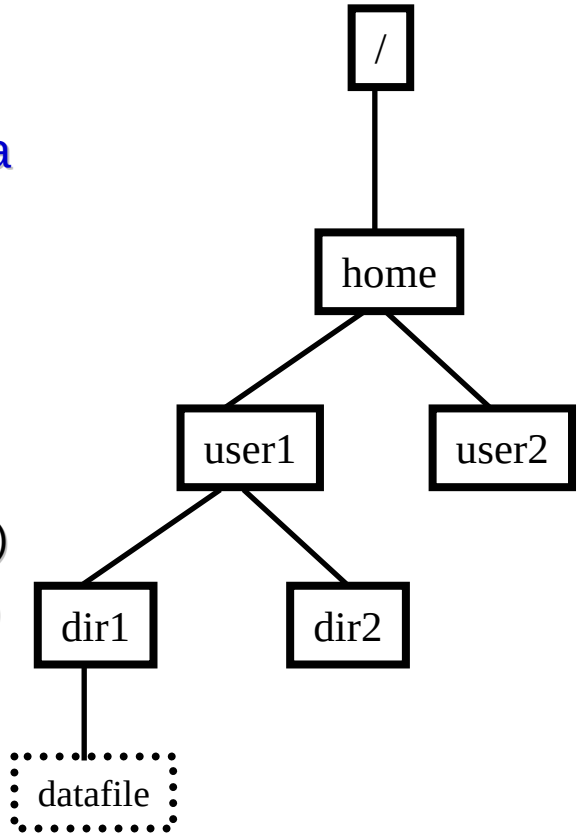
    `/home/user1/dir1/datafile`
  - **Ruta relativa**: Tomando como punto de partida el directorio actual

    `home/user1/dir1/datafile` (desde el directorio /)

    `user1/dir1/datafile` (desde el directorio /home)

    `dir1/datafile` (desde el directorio /home/user1)

    `datafile` (desde el directorio /home/user1/dir1)



Organización jerárquica de ficheros

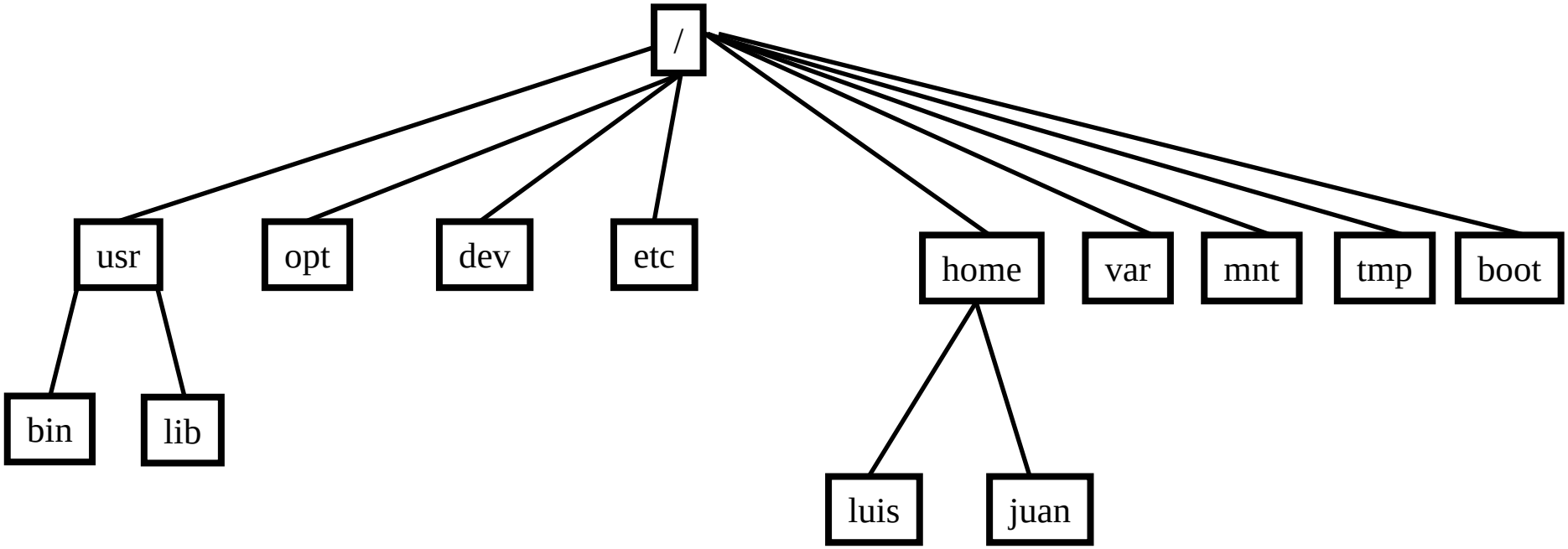
Reglas sintácticas

- Archivos
  - Linux limita la longitud de los nombres de archivos a 256 caracteres
  - Los nombres de archivo no pueden contener caracteres con significado especial para el shell

    ! @ # $ % ^ & * ( ) [ ] { } ` ” \ / | ; < > ‘
  - No tienen extensión obligatoria y la extensión no tiene significado intrínseco
- Rutas de acceso
  - Los sucesivos subdirectorios hasta llegar al archivo se separan con el carácter ***/*** (no \ como en MS-DOS)

Organización jerárquica de ficheros

El árbol de directorios de Linux



Organización jerárquica de ficheros

- Directorio **/usr**
  - Contiene las órdenes del sistema, las bibliotecas de rutinas y las páginas del manual
  - Sus contenidos no deben cambiar durante el uso normal del sistema
- Directorio **/var**
  - Contiene información que puede cambiar durante el uso del sistema
    - ✓ Directorios de spool (colas del trabajo de impresión, de correo electrónico)
    - ✓ Archivos tipo LOG (informan sobre los eventos que se producen en el sistema)
    - ✓ Archivos temporales
- Directorio **/etc**
  - Contiene información sobre la configuración del sistema
    - ✓ Cuentas de usuario
    - ✓ Discos montados en el sistema
    - ✓ Impresoras instaladas

Organización jerárquica de ficheros

- Directorio **/dev**
  - Contiene los archivos que hacen referencia a todos los dispositivos del sistema
    - ✓ En Unix todos los dispositivos (puertos, terminales, discos, etc) se representan como archivos del árbol de directorios
- Directorio **/tmp**
  - Contiene archivos temporales del sistema
- Directorio **/boot**
  - Contiene archivos necesarios para el arranque del computador (GRUB)
- Directorio **/opt**
  - Contiene software opcional ajeno al sistema operativo
- Directorio **/mnt**
  - Contiene los directorios de acceso a la unidad de CD-ROM y disquete
    - ✓ /mnt/cdrom
    - ✓ /mnt/floppy

Contenidos

- MÓDULO 1: Fundamentos del sistema operativo UNIX/Linux
  - Introducción al sistema Linux
  - Órdenes básicas de acceso al sistema
  - Organización jerárquica de ficheros
  - **Órdenes sobre archivos y directorios**
  - Edición de textos con **vi**
  - Características generales de los shells
  - El Bourne Again shell
  - Órdenes avanzadas para el tratamiento de archivos

Órdenes sobre archivos y directorios

Formato de las órdenes de Linux

- **Sintaxis:** *órdenes [opciones] argumentos*
  - La orden es el comando que se va a ejecutar
  - Los argumentos son los archivos o directorios sobre los que se va a ejecutar el comando
  - Las opciones modifican la forma de ejecutar la orden. Suelen ir precedidas de un signo **–**

```
prompt> ls -a /tmp
prompt> cat /etc/passwd /usr/include/time.h
prompt> ls -a -l /tmp  ó  ls –al /tmp
```

La ayuda de Linux

- **man:** muestra información sobre el formato, opciones, utilidad y ejemplos de una orden

```
prompt> man  orden
```

Órdenes sobre archivos y directorios

Moverse por el árbol de directorios

- **cd** cambia el directorio de trabajo

  cd [directorio]

  - Si no se indica directorio, cambia al HOME del usuario
  - El directorio **.** representa el directorio actual
  - El directorio .. representa el directorio padre del directorio actual

```
prompt> cd /etc
prompt> cd ..
```

- **pwd** muestra el directorio en el que el usuario se encuentra

  pwd

```
prompt> pwd
```

Órdenes sobre archivos y directorios

Listar contenidos de directorios

- **ls [-aAcCdfFgilLmnpqrRstux1]** [*ruta* ...]

-a	lista todas las entradas	**-1**	fuerza el formato de un nombre de fichero en cada línea
-F	pone '/' al final de directorios, '*' al final de ejecutables y '@' al de enlaces simbólicos	**-A**	igual que –a pero no lista los directorios '.' y '..'
-i	muestra el número de inodo en la columna 1	**-L**	sigue los enlaces simbólicos
-l	listado largo	**-o**	como –l pero no muestra grupo
-n	con –1 muestra UID/GID	**-q**	muestra los caracteres no visualizables
-p	pone '/' al final de los directorios	**-R**	lista recursivamente directorios
-r	invierte el sentido de ordenación	**-t**	ordena por fechas
-s	muestra tamaño en bloques	**-x**	salida multicolumna ordenada horizontalmente
-u	sentido de ordenación por la fecha del último acceso		

```
prompt> ls –alF /etc
```

Órdenes sobre archivos y directorios

Crear y borrar directorios

- **mkdir** Crea un nuevo directorio

  mkdir [-p] *directorio*

  - **-p** Crea los directorios necesarios para construir la ruta del nuevo directorio

```
prompt> cd
prompt> mkdir tmp
prompt> cd tmp
prompt> pwd
```

- **rmdir** Borra un directorio

  rmdir *directorio*

```
prompt> cd ..
prompt> rmdir tmp
```

Órdenes sobre archivos y directorios

Mostrar contenidos de archivos

- **cat** Mostrar ficheros de texto

  cat [-s] [-v[et]] [*fichero* ...]

  -v Muestra caracteres de control (no imprimibles)

  -s Reemplaza varias líneas en blanco por una única línea

  -t como **–v** pero además imprime tabuladores como ^I

  -e lo mismo que **–v** pero también imprime $ al final de cada línea

```
prompt> cat -se /etc/passwd
```

Órdenes sobre archivos y directorios

Mostrar contenidos de archivos

- **more** Mostrar ficheros de texto en pantalla página a página

  more [**-dpcsu**] [**-num**] [+/*patrón*] [+**linenum**] *fichero ...*

-num	Especifica el tamaño de pantalla (en líneas)
-d	Visualiza el mensaje "[Press space to continue, 'q' to quit]"
-p	No realiza desplazamiento, sino que limpia la pantalla y visualiza el texto
-c	No realiza "scroll", visualiza línea a línea de arriba a abajo
-s	Sustituye varias líneas en blanco consecutivas por una sola
-u	Suprime subrayado
+/*patrón*	Empieza por la página que contiene la palabra patrón
+**linenum**	Comienza en la línea **linenum**

```
prompt> more  /etc/passwd
```

Mostrar contenidos de archivos

- **tail** Mostrar las últimas líneas o caracteres de un fichero

  tail [+- n] [f] *fichero*

  -n Muestra las **n** últimas líneas (o caracteres) de un fichero (por defecto n=10)

  +n Muestra las últimas líneas (o caracteres) de un fichero saltando las **n-1** primeras

  -f Si aparece este argumento fichero no acaba: duerme 1 s, y muestra nuevas líneas si las hay

```
prompt> tail –5 /etc/passwd
prompt> tail -20f /etc/passwd
```

Órdenes sobre archivos y directorios

Mostrar contenidos de archivos

- **head** Mostrar las primeras líneas o caracteres de un fichero

  head [- n] *fichero*

  -n Muestra las **n** primeras líneas (o caracteres) de un fichero (por defecto son 10)

```
prompt> head –5 /etc/passwd
```

- **wc** Contar líneas, palabras y caracteres de un archivo

  wc [-clw] *fichero ...*

  -c Cuenta caracteres

  -l Cuenta líneas

  -w Cuenta palabras

```
prompt> wc -l /etc/passwd
```

Órdenes sobre archivos y directorios

Crear archivos de texto

tee Crea un archivo de texto a partir de datos introducidos por la entrada estándar (teclado)

tee [-a] *[fichero]*

-a Concatena los nuevos datos al final del fichero, en lugar de sobreescribir el mismo

```
prompt> tee prueba.txt
        Hola que tal
        Hasta luego
        ^C
prompt> more prueba.txt
```

Órdenes sobre archivos y directorios

Crear archivos de texto

cat (con redirección) Crear un fichero de texto a partir de los datos introducidos por la entrada estándar (teclado)

- **>** Sobreescribe el archivo si éste ya existía. En caso contrario crea un archivo nuevo
- **>>** Concatena los nuevos datos al final del archivo, en lugar de sobreescribir el mismo

```
prompt> cat > prueba2.txt
        Hola de nuevo
        Adios
        ^C
prompt> more prueba2.txt
```

Órdenes sobre archivos y directorios

Crear, renombrar y borrar archivos y directorios

cp Copiar ficheros o directorios

cp [-i] *fuente destino*

cp [-i] *fuente1 [fuente2 ...] directorio*

cp [-i]rR *directorio1 directorio2*

-i Interactivo. Pide confirmación antes de sobrescribir un archivo o directorio

-r-R Recursivo. Copia los subdirectorios y todos sus contenidos

```
prompt> cp /etc/passwd mi.passwd
```

Órdenes sobre archivos y directorios

Crear, renombrar y borrar archivos y directorios

mv Renombrar y mover ficheros y directorios

mv [-f] *fuente1 [fuente2 ...] destino*

-f Suprime el destino sin preguntas

```
prompt> mv mi.passwd ./destino
```

Órdenes sobre archivos y directorios

Crear, renombrar y borrar archivos y directorios

rm Borra ficheros y directorios

rm [-if] *fichero1 [fichero2 ...]*

rm [-ifrR] *directorio1 [directorio2 ...]*

-i Interactivo (pide confirmación)

-f No emite mensajes de error cuando el archivo o directorio no existe

-r-R Recursivo. Borra un directorio y todos sus contenidos

```
prompt> rm -i destino/output
```

Órdenes sobre archivos y directorios

Comparar archivos

cmp Compara dos ficheros. Muestra los números de línea y el byte donde tiene lugar la primera diferencia

cmp [-l] *fichero1 fichero2*

-l Muestra todas las diferencias (por omisión sólo muestra la primera)

```
prompt> cmp prueba.txt prueba2.txt
```

Órdenes sobre archivos y directorios

Comparar archivos

diff Compara dos ficheros de texto e informa de lo que hay que hacer en uno para que sea como el otro

diff [-ibw] *fichero1 fichero2*

-b Ignora los blancos (espacio y tabulador) a final de línea

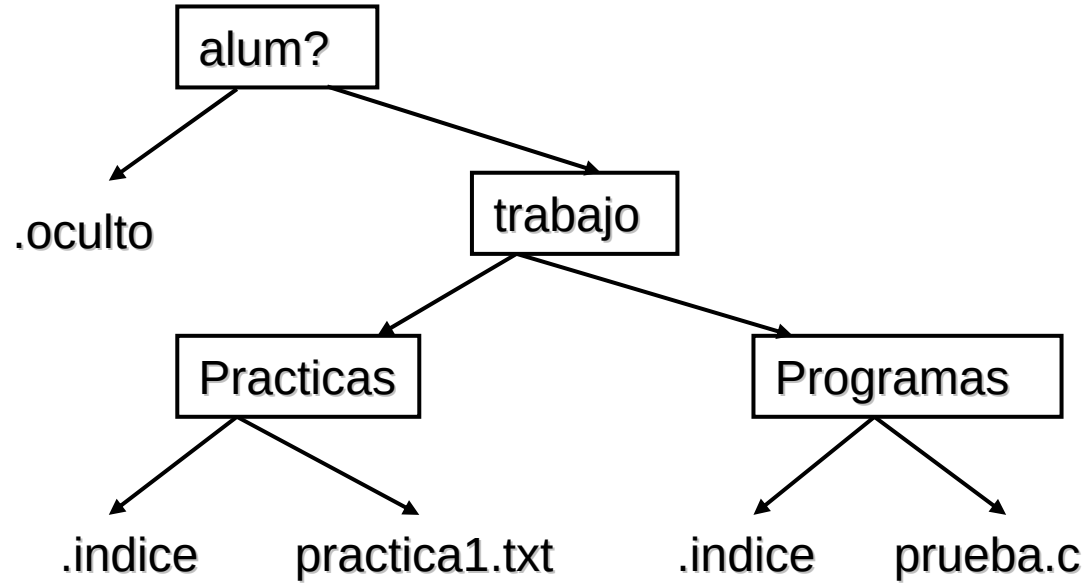
-w Ignora todos los blancos

-i Ignora diferencias entre mayúsculas y minúsculas

```
prompt> diff prueba.txt prueba2.txt
```

Práctica: Árbol de directorios

- Crear el siguiente árbol de directorios
  - **mkdir**: trabajo, Practicas, Programas
  - **cat**: .oculto, .indice, practica1.txt, prueba.c
  - Usar **cp** para crear el fichero .indice del directorio Programas



Órdenes sobre archivos y directorios

Práctica: Árbol de directorios

- Comprobar el resultado y el tamaño de los distintos directorios y archivos creados
  - Los archivos que comienzan por "." son archivos ocultos
  - Para visualizar los resultados usar el comando **ls**
- Cambiar el nombre del archivo .indice del directorio *trabajo*
  - Comprobar que se ha producido el cambio
- Borrar los archivos .indice y practica1.txt del directorio *Practicas*
- Borrar el directorio *Practicas*

Contenidos

- MÓDULO 1: Fundamentos del sistema operativo UNIX/Linux
  - Introducción al sistema Linux
  - Órdenes básicas de acceso al sistema
  - Organización jerárquica de ficheros
  - Órdenes sobre archivos y directorios
  - **Edición de textos con vi**
  - Características generales de los shells
  - El Bourne Again shell
  - Órdenes avanzadas para el tratamiento de archivos

Edición de textos con vi

Modos en *vi*

- Estados en los que el editor puede encontrarse
  - **Modo de órdenes**: Introducir órdenes dirigidas al editor
  - **Modo de entrada de texto**: Introducir texto

- Conmutación de modos en *vi*
  - De modo TEXTO a modo ÓRDENES: pulsando la tecla ESC
  - De modo ÓRDENES a modo TEXTO: invocando alguna orden de edición
    - ✓ **i** (insertar)
    - ✓ **a** (añadir)
    - ✓ **r** (reemplazar)

Edición de textos con vi

Modo ÓRDENES

- Borrar un carácter **x**
- Borrar *n* caracteres **nx**
- Reemplazar un carácter **r**
- Borrar una línea **dd**
- Borrar n líneas **ndd**
- Insertar una línea debajo **o** (entramos en modo texto)
- Insertar una línea encima **O** (entramos en modo texto)
- Pasar a modo de inserción **i** (insertamos a partir de la posición del cursor)
- Pasar a modo de inserción **a** (insertamos a partir de la siguiente posición del cursor)
- Pasar a modo de sobreescritura **R**
- Dehacer **u**
- Salvar y salir **ZZ** ó **:wq**
- Salir sin salvar **:q!**

Modo TEXTO

- Pasamos a este modo con:
  - i, a, o, O, ...
- Pasamos a modo sobreescritura
  - R
- Volvemos a modo órdenes con:
  - ESC

Edición de textos en vi

Práctica: Creación de un archivo en vi

- Iniciar **vi**
  - Escribir **vi** y presionar <Intro>
- Pasar a modo texto (insertar)
  - <i>
- Introducir texto
  - Tareas para hoy:

    a. Practicar con el vi

    b. Clasificar datos e imprimir resultados
- Ir a modo órdenes
  - <ESC>
- Grabar el archivo
  - <ESC> <:w practica1> <Intro>
- Salir del vi
  - <ESC> <:q> <Intro>

Edición de textos en vi

Práctica: Modificaciones sobre un archivo vi

- Abrir el fichero con el editor vi
  - vi practica1
- Situar el cursor a continuación de la palabra “tareas” y añadir el texto “a realizar”
  - <a>
- Borrar la palabra “para” de la primera línea
  - <ESC> <x>
- Borrar la tercera línea entera
  - <ESC> <dd>
- Sobreescribir la segunda línea con: “Modificar un texto en vi”
  - <ESC> <R>
- Grabar y salir
  - <ESC> <:wq>

Contenidos

- MÓDULO 1: Fundamentos del sistema operativo UNIX/Linux
  - Introducción al sistema Linux
  - Órdenes básicas de acceso al sistema
  - Organización jerárquica de ficheros
  - Órdenes sobre archivos y directorios
  - Edición de textos con **vi**
  - **Características generales de los shells**
  - El Bourne Again shell
  - Órdenes avanzadas para el tratamiento de archivos

Características generales de los shells

Tipos de shell

- **Bourne shell**: */bin/sh*
  - Primer shell diseñado para el sistema operativo Unix (1978, Steve Bourne)
- **C-shell**: */bin/csh*
  - Historia, alias, estructuras de control adicionales (similares a las de C)
- **Tab C-shell**: */bin/tcsh*
  - Historia mejorada, completa automáticamente los nombres con ESC
  - Compatible con C-shell
- **Korn shell**: */bin/ksh*
  - Mejores características de programación
- **Bourne again shell**: */bin/bash*
  - Introduce las mejoras de *csh* y *ksh*, compatible con Bourne shell

Características generales de los shells

Entrada/salida estándar

- La entrada/salida por defecto de una orden Unix es el terminal (teclado+pantalla)
  - Si una orden genera una salida, ésta se visualiza por pantalla
    - ✓ Ejemplos: cat, ls, pwd, who, date, ...
  - Si una orden requiere una entrada, ésta se introduce por teclado
    - ✓ Ejemplos: cat, tee, write
- El shell permite redirigir tanto la entrada como la salida estándar
  - Se puede redirigir la salida para que la orden vuelque los datos a un fichero (operadores > y >>)
  - Se puede redirigir la entrada para que la orden tome los datos de un fichero (operador <)

Redirección de la salida

- Los operadores de redirección de la salida son: **>** y **>>**
- Formatos de redirección de la salida
  - **Orden > archivo de salida**
    - ✓ Si el archivo de salida no existía, el shell crea uno
    - ✓ Si el archivo ya existía, se sobrescribe y pierde su contenido anterior
  - **Orden >> archivo de salida**
    - ✓ Si el archivo de salida no existía, el shell crea uno
    - ✓ Si el archivo ya existía, el shell añade la salida al final del archivo, pero no pierde su contenido anterior

```
prompt> date > salida
prompt> who > salida
prompt> more salida
inma   pts/0     Jun  21 14:42
alum1  pts/1     Jun  21 16:29
```

```
prompt> date > salida
prompt> who >> salida
prompt> more salida
vie jun 21 18:45:50 CEST 2002
inma   pts/0     Jun  21 14:42
alum1  pts/1     Jun  21 16:29
```

Redirección de la entrada

- El operador de redirección de la entrada es <
- Formato de redirección de la entrada
  - **orden < archivo entrada**
    - ✓ El fichero de entrada debe existir y debe contener los datos de entrada al programa
- Ejemplo:
  - write inma < mensaje
    - ✓ El fichero mensaje debe contener el texto que queremos enviar

Características generales de los shells

Elementos de una orden

orden –opciones argumentos

- Orden a ejecutar
  - ✓ ls, who, ps, ...
- Opciones o modificadores de la orden
  - ✓ Modifican el significado o el formato de salida de la orden a ejecutar
  - ✓ En ocasiones llevan el signo "-" delante
    - ➢ *ls –als*
    - ➢ *ps aux*
  - ✓ Las distintas opciones pueden ponerse juntas o separadas
    - ➢ *ls –a –l –s*
- Argumentos de la orden
  - ✓ Son los elementos sobre los que se ejecuta la orden (uno o varios)
    - ➢ *Nombre del fichero o directorio, identificador de proceso, nº tarea, ...*
    - ➢ *Ejemplos: rm <fichero>*
      cp <fichero1> <fichero2>

Características generales de los shells

Secuenciamiento de órdenes

- Se puede introducir una serie de órdenes en la misma línea separadas por punto y coma
- El shell las ejecuta secuencialmente de izquierda a derecha
- Ejemplo:

```
prompt> ps; date; who
PID TTY STAT TIME COMMAND
31527  p3 S    0:00 -tcsh
32031  p3 R    0:00 ps
Sat Jun  5 14:14:26 CEST 1999
manuel   tty1     Jun  4 18:31
rafa     ttyp3    Jun  5 11:57 (pc-rafa.dacya.ucm.es)
```

Características generales de los shells

Agrupamiento de órdenes

- Una secuencia de órdenes puede agruparse colocándolas entre paréntesis
- Un grupo de órdenes puede redireccionarse como si se tratase de una única orden

```
prompt> (ps; date; who)> output
prompt> more output
PID TTY          TIME CMD
2871 pts/1    00:00:00 bash
3197 pts/1    00:00:00 bash
3198 pts/1    00:00:00 ps
vie jun 21 18:27:30 CEST 2002
inma     pts/0    Jun 21 14:42 (pc-inma.dacya.ucm.es)
inma     pts/1    Jun 21 16:29 (equipo23)
Alum1    pts/2    Jun 21 16:29 (pc-inma.dacya.ucm.es)
```

Agrupamiento de órdenes

- Observar la diferencia con esta otra orden sin paréntesis

```
prompt> ps; date; who > output
PID TTY          TIME CMD
2871 pts/1    00:00:00 bash
3202 pts/1    00:00:00 ps
vie jun 21 18:31:50 CEST 2002
```

  - El fichero **output** sólo almacena la salida de la orden **who**. Las salidas de **ps** y **date** se visualizarán por pantalla

Características generales de los shells

Expansión de la línea de órdenes

- El shell proporciona 3 tipos de **comodines** que permiten especificar múltiples nombres de ficheros como argumentos de una orden
- Si se utilizan comodines, el shell expande el argumento y aplica la orden sobre todos los archivos cuyo nombre pueda sustituirse por los caracteres comodín
- El **comodín ***
  - Puede sustituirse por cualquier cadena de caracteres
    - ✓ **ls *** → Lista TODOS los ficheros del directorio actual
    - ✓ **rm *.c** → Elimina todos los ficheros que terminan con la cadena "c"
    - ✓ **ls dat*** → Lista todos los ficheros que comienzan por la cadena dat
    - ✓ **cp p*.o dir** → Copia en el directorio dir todos los ficheros que comienzan por "p" y terminan por ".o"

Características generales de los shells

- El **comodín ?**
  - Este comodín puede sustituirse por cualquier carácter (uno solo)
    - ✓ **ls prueba?** → Lista todos los ficheros del tipo prueba1, prueba2, pruebaK
    - ✓ **rm lista?.dat** → Borra todos los ficheros del tipo lista1.dat, listaK.dat
- **Caracteres entre corchetes**
  - Se puede especificar una lista de caracteres entre corchetes que pueden corresponder con un carácter del nombre del fichero
    - ✓ **ls prueba[12AB]** → Lista los ficheros prueba1, prueba2, pruebaA y pruebaB
    - ✓ **rm lista[XYZ].dat** → Borra los ficheros listaX.dat, listaY.dat y listaZ.dat si existen

```
prompt> ls *
contA.c   contabilidad   contar.c  contar  contB.c  prueba
prompt> ls cont*
contA.c   contabilidad  contar   contar.c   contB.c
prompt> ls contar*
contar   contar.c
prompt> ls cont?.c
contA.c   contB.c
prompt> ls cont[ABCD].c
contA.c  contB.c
```

Expansión de la línea de órdenes

- Argumentos ambiguos
  - Existen determinadas órdenes que aceptan comodines siempre que no planteen ambigüedad
  - Por ejemplo: orden **cd** (en el C-shell)
    - ✓ Supongamos dos directorios que se llaman pruebas y proyectos respectivamente

```
prompt> cd pr*
pru*: Ambiguous.
prompt> cd pro*
Directory: /home/inma/proyectos
```

    - ✓ En el bash shell ⇒ si hay ambigüedad accede al primer directorio

Características generales de los shells

Caracteres especiales usados como argumentos

- **Directorio actual:** se representa con "."
- **Directorio padre** (anterior al actual): se representa con ".."
- **Directorio HOME:** se representa por el símbolo "~"

```
prompt> mkdir trabajo
prompt> cd trabajo
Directory: /home/inma/trabajo
prompt> mkdir proyecto
prompt> cd proyecto
Directory: /home/inma/trabajo/proyecto
prompt> cp ../../contar .
```

```
prompt> cd /tmp
Directory: /tmp
prompt> cp datos ~/trabajo/proyecto
```

Cauces o "pipes"

- La salida de una orden puede usarse como entrada de otra orden
  - Ejemplo: si queremos ver un listado de procesos página a página

```
prompt> ps aux > salida
prompt> more salida
prompt> rm salida
```

  - El shell ofrece una forma más potente y elegante de hacer esto: los pipes o cauces
    - ✓ El operador de pipe es " | "
    - ✓ Formato general de un pipe:

      orden A | orden B → la salida estándar de la orden A se utiliza como entrada de la orden B
    - ✓ Se pueden encadenar tantas órdenes como se quiera
    - ✓ Usando un pipe, el ejemplo anterior se podría hacer así:

```
prompt> ps aux|more salida
```

Características generales de los shells

Variables de los shells

- Una variable es un elemento con un determinado nombre al que se le asigna un cierto valor
- Para referirnos a una variable, utilizar el símbolo "$" delante de la variable
  - **echo $nombre**
- El shell utiliza variables estándar que permiten personalizar el entorno
  - Número de terminal → variable **tty**
  - Tipo de terminal → variable **term**
  - Directorio HOME → variable **home**
  - Directorio actual → variable **cwd**
  - Tipo de shell → variable **shell**
  - Nombre de usuario → variable **user**
  - Número de usuario → variable **uid**
  - Nombre de grupo → variable **group**
  - Número de grupo → variable **gid**
  - camino → variable **path**
  - Tipo de prompt → variable **prompt**

Características generales de los shells

Definición de nuevas variables

- Una variable no puede utilizarse como una orden sino como argumento de una orden
  - Ejemplo: **cd $HOME**
- El usuario puede definir sus propias variables
  - Para facilitar o acelerar ciertas operaciones
  - Pueden ser necesarias para que algunos programas funcionen correctamente
- Asignar un valor a una variable

  Sintaxis: ***nombre=valor***
  - **set** sin argumentos, visualiza una lista con todas las variables definidas
- **unset** elimina una variable existente

  Sintaxis: ***unset nombre***

Ejemplo de definición de variables

- Supongamos un directorio *$home/trabajo/proyecto/ejecutables* al que se accede muy a menudo
  - El nombre del directorio se puede sustituir por una variable

```
prompt> exe=$home/trabajo/proyecto/ejecutables
prompt> echo $exe
/home/inma/trabajo/proyecto/ejecutables
prompt> cd $exe
Directory: /home/inma/trabajo/proyecto/ejecutables
```

- También se pueden definir variables numéricas

```
prompt> n=5
prompt> head -$n /etc/passwd
```

- Muchos programas requieren que el usuario defina ciertas variables
  - Directorios donde se encuentran ejecutables o librerías
  - Variables numéricas necesarias para la ejecución del programa

Características generales de los shells

La orden echo

- Permite visualizar información por pantalla
  - Cadenas de caracteres
  - Valores de las variables
  - Resultados de ejecución de órdenes
- Ejemplos
  - La cadena puede ir entre comillas o sin ellas

```
prompt> echo Hola, Buenos dias
Hola, Buenos dias
prompt> echo "Hola, Buenos dias”
Hola, Buenos dias
```

  - Visualizar el valor de una variable: se pone delante “$”

```
prompt> echo Mi nombre de usuario es USER
Mi nombre de usuario es USER
prompt> echo Mi nombre de usuario es $USER
Mi nombre de usuario es inma
```

La orden echo

- Visualizar el resultado de una orden: se pone entre comillas invertidas (`orden `)

```
prompt> echo Mi nombre de usuario es `whoami`
Mi nombre de usuario es inma
prompt> echo El directorio actual es `pwd`
El directorio actual es /home/inma
```

- '.....' ⇒ Deshabilita TODOS los metacaracteres ($ \ !)

```
prompt> echo '$USER'
$USER
```

- "....." ⇒ No deshabilita los metacaracteres ($ \ !)

```
prompt> echo "$USER"
inma
```

Contenidos

- MÓDULO 1: Fundamentos del sistema operativo UNIX/Linux
  - Introducción al sistema Linux
  - Órdenes básicas de acceso al sistema
  - Organización jerárquica de ficheros
  - Órdenes sobre archivos y directorios
  - Edición de textos con **vi**
  - Características generales de los shells
  - **El Bourne Again shell**
  - Órdenes avanzadas para el tratamiento de archivos

Funciones principales del *bash* shell

- Funciones de personalización para que el usuario cree un espacio de trabajo a su gusto
  - Mediante ficheros especiales (**.bashrc, .bash_profile**)
  - Variables locales y de entorno (**PATH**, **HOME**, **PS1**, ...)
  - Órdenes especiales (**umask**, **nice**, ...)
- Funciones interactivas que permiten
  - Usar comodines para referirse a grupos de ficheros
  - Ejecutar una orden para proporcionar argumentos a otra
  - Conectar la salida de una orden a la entrada de otra (**pipes**)
  - Historial de órdenes pasadas que puedan ser reutilizadas
  - Completar el nombre de las variables de entorno y archivos y directorios con el tabulador
  - Asignar nombres breves a ficheros u órdenes complejas (**alias**)

Funciones principales del *bash* shell

- Funciones de programación
  - El *bash* shell se puede utilizar como un lenguaje de propósito especial
  - Permite escribir programas llamados “guiones shell”
  - Los guiones shell son interpretados (no hace falta compilar)
  - Tiene utilidades de depuración (bash –xv)
  - Los guiones shell juegan un papel muy importante en el control de trabajos, control de errores o gestión de ficheros
  - Las estructuras de programación son muy semejantes a las del lenguaje C

Tipos de variables del bash shell

- Variables locales o privadas del shell
  - Se definen, utilizan y suprimen dentro del shell
  - Suelen emplearse para configurar los principales aspectos del shell particular
  - Sus valores no se transmiten a los subshells hijos
  - Se definen con una **asignación** y se suprimen con la orden **unset**
  - Suelen ser nombres con letras minúsculas
- Variables de entorno o exportadas
  - Se definen normalmente antes de entrar en el shell
  - Suelen emplearse para configurar los aspectos principales de la sesión
  - Sus valores SÍ se trasmiten a los subshells hijos
  - Se definen con la orden **export** y se suprimen con **unset**
  - Suelen usar nombres con letras mayúsculas

Bourne Again Shell

- **Definición de variables locales**
  - Asigna un valor a una variable local

    Sintaxis: ***nombre=valor***
  - **unset** Elimina una variable local existente

    Sintaxis: ***unset nombre***
  - **set** Visualiza todas las variables del sistema

    Sintaxis: ***set*** (sin argumentos)
- **Definición de variables de entorno**
  - **export** Asigna un valor a una variable de entorno

    Sintaxis: ***export nombre=valor***
  - **unset** Elimina una variable de entorno existente

    Sintaxis: ***unset nombre***
  - **env** Visualización de las variables de entorno

    Sintaxis: **env** (sin argumentos)

Principales variables de entorno del *bash* shell

- TERM → Tipo de terminal
- HOSTNAME → Nombre de la máquina
- HOME → Directorio HOME
- USER → Nombre del usuario
- PATH → Camino donde se buscan los ficheros ejecutables
- MAIL → Directorio donde se comprueba la existencia de correo
- SHELL → Tipo de shell
- HISTSIZE → Se le da un valor numérico ***n*** y el shell conserva las últimas ***n*** instrucciones ejecutadas

- Si una variable de entorno tiene el mismo nombre que una variable local, un cambio de valor de la variable de entorno también afecta a la variable local del mismo nombre, y a la inversa.

Elección del camino (path)

- La variable **$PATH** contiene una lista de directorios en los que el shell busca directamente los ficheros ejecutables
  - Búsqueda ordenada: Comienza por el primer directorio, cuando encuentra el comando lo ejecuta y termina la búsqueda
- Se trata de una variable tipo lista a la que se pueden añadir más elementos
  - PATH=$PATH:newpath
    - ✓ $PATH ⇒ representa la ruta actual
    - ✓ :newpath ⇒ lo que se añade a la ruta actual

```
prompt> echo $PATH
/usr/bin:/bin:/usr/local/bin:/usr/X11R6/bin:/home/inma/bin
prompt> PATH=$PATH:/usr/games
prompt> echo $PATH
/usr/bin:/bin:/usr/local/bin:/usr/X11R6/bin:/home/inma/bin:/usr/games
```

Elección del indicador de la línea de órdenes (prompt)

- El indicador de las líneas de órdenes se puede cambiar a gusto del usuario mediante la variable local **PS1**
  - **PS1= "<nuevo prompt>"**
- Si queremos visualizar el número de la orden, podemos usar el operador "!" en la definición de la variable **prompt**
- **Ejemplos**

```
prompt> PS1="HOLA > "
HOLA > PS1="HOLA \! > "
HOLA 143 > PS1="\u : \! >"
inma : 144 > PS1="\h : \! >"
equipo23 : 145 > PS1="[\u@\h: \W]$"
[inma@equipo23 inma]$
```

\u	nombre de usuario
\h	nombre de la máquina
\w	directorio de trabajo (todo el path)
\W	sólo directorio de trabajo
\!	Número de la orden

El historial de órdenes

- El *bash* shell dispone de una lista histórica de órdenes que permite recordar las últimas órdenes introducidas
- La historia de órdenes permite volver a repetir las últimas órdenes de forma rápida y también permite editar la línea de órdenes
- La longitud de la lista histórica de órdenes se define mediante la variable global **$HISTSIZE**
  - **HISTSIZE = <longitud>** define la longitud de la lista histórica de órdenes
  - **history** visualiza la lista histórica de órdenes

```
prompt> export HISTSIZE = 4
prompt> history
9  export HISTSIZE=4
10  pwd
11  ls -l
12  history
```

Opciones del historial de órdenes

Opción	*Función*
!!	Ejecuta la última orden
!<*num*>	Ejecuta la orden especificada en <*num*>
!<*cadena*>	Ejecuta la última orden que comience por <*cadena*>
!$	Repite los argumentos de la orden anterior
!<*num*>:p	Visualiza la orden especificada en <*num*> pero sin ejecutarla
Tecla ↑	Va pasando, de una en una, a las órdenes anteriores cada vez que se pulsa
^<cad1>^<cad2>	Ejecuta la orden anterior, sustituyendo <cad1> por <cad2> (edición de la línea de órdenes)

Utilización de “alias”

- El *bash* shell permite asignar un seudónimo (alias) a una orden
- El uso del alias es útil para órdenes largas usadas con frecuencia
- Se puede definir un alias con el nombre de una orden que ya existe
  - En este caso, al introducir el nombre de la orden se ejecuta el alias

alias <nombre>=< "orden“ > define un alias para la orden especificada

- **Ejemplo:**

```
alias del="rm -i"
del memo.txt
rm: remove memeo.txt? y
```

- El alias puede definirse de forma permanente
  - ✓ Introducirlo en el fichero de configuración **.bashrc**

unalias <nombre> elimina un alias previamente definido

alias <nombre> muestra la orden que ejecuta el alias

- Sin <nombre> muestra el valor de todos los alias

Contenidos

- MÓDULO 1: Fundamentos del sistema operativo UNIX/Linux
  - Introducción al sistema Linux
  - Órdenes básicas de acceso al sistema
  - Organización jerárquica de ficheros
  - Órdenes sobre archivos y directorios
  - Nociones básicas sobre usuarios y grupos
  - Características generales de los shells
  - El Bourne Again shell
  - **Órdenes avanzadas para el tratamiento de archivos**

Órdenes avanzadas para el tratamiento de archivos

Buscar archivos y directorios

find Busca un archivo dentro del árbol de directorios

find [*directorio_de_busqueda*] [–name *patrón_de_busqueda*] –print

-print	Imprime el nombre y la localización de los archivos seleccionados en pantalla
-name *patrón*	Busca ficheros cuyo nombre coincide con el patrón
-links n	Selecciona ficheros que tengan **n** o más enlaces
-size n	Selecciona ficheros que ocupen **n** o más bloques de 512 bytes
-atime n	Selecciona cualquier archivo que haya sido accedido en los últimos **n** días
-exec *orden*	Después de seleccionar los archivos, ejecuta una orden que los usa como argumento

```
prompt> find /usr/lib –name '*.a' -print
```

Órdenes avanzadas para el tratamiento de archivos

Buscar cadenas de texto

grep Busca una determinada cadena de texto dentro de los archivos especificados

grep *[opciones]* *'cadena_texto' fichero1, fichero2, ...*

- **-c** Muestra sólo el número total de líneas que contienen el patrón
- **-i** ignora la distinción entre mayúsculas y minúsculas
- **-h** Evita que aparezca el nombre del fichero que contiene el patrón
- **-l** Muestra una sola vez el nombre de ficheros que contienen las líneas seleccionadas
- **-n** Precede cada línea con su número de línea dentro del fichero
- **-s** Suprime los mensajes de error referentes a ficheros no existentes
- **-v** Muestra todas las líneas excepto las que contienen el patrón
- **-w** Muestra las líneas que satisfacen el patrón con una palabra completa

Buscar cadenas de texto

- Ejemplos:
  - **prompt>** grep 'Pardines Lence' /etc/passwd
  - **prompt>** grep 'alum' /etc/passwd ⇒ Probar con todas las opciones
- La orden **grep** resulta muy útil junto con la utilización de **pipes**
  - Cuando una orden genera una salida muy larga, podemos filtrar esta salida mediante la orden **grep** y encontrar una información específica
  - Ejemplos:
    - ✓ **prompt>** `ps –aux | grep inma` ⇒ Muestra todos los procesos que pertenecen al usuario inma
    - ✓ **prompt>** `ls –al | grep rwxr-xr-x` ⇒ Muestra todos los archivos del directorio actual con permisos rwxr-xr-x

Órdenes avanzadas para el tratamiento de archivos

Ordenar archivos de texto

- **sort** Ordena un archivo de texto

  sort [-u] [-o *fichero_salida*] [-bdfinr] [-t*caracter*] [+*m*] [-*N*] [*fichero* ...]

  - **-u** No muestra líneas duplicadas
  - **-o fichero** Muestra la salida ordenada a ***fichero_salida***
  - **-t*carácter*** Utiliza ***carácter*** como separador de campos
  - **+m** Salta los ***m*** primeros campos y comienza a ordenar a partir del ***m+1***
  - **-N** Ignora todos los campos a partir del campo **N** (no inclusive)
  - **-f** No distingue entre mayúsculas y minúsculas
  - **-n** Compara campos numéricos
  - **-r** Invierte el orden de ordenamiento

Ordenar archivos de texto

- Ejemplos:

```
prompt> sort +1 –2 fichero
```

  - Ordena por el segundo campo (salta el 1º campo e ignora todos a partir del 2º campo)

```
prompt> sort –t: -n +2 –3 /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
```

  - Ordena por el tercer campo (uid), usando como delimitador el carácter ":" y considerando este campo como un valor numérico