Práctica 1. Introducción al Sistema Operativo LINUX

1. Objetivos.

- Utilizar comandos elementales de UNIX.
- Aprender el funcionamiento básico del editor de textos `vi` de UNIX.
- Compilar y ejecutar un programa escrito en lenguaje C.
- Ejecutar procesos.

2. El *shell* de Linux

El *shell* de un sistema operativo es la interfase entre éste y sus usuarios. Se caracteriza fundamentalmente porque facilita al usuario la posibilidad de escribir órdenes en la que se conoce como línea de órdenes o línea de comandos.

La forma que tiene el sistema operativo de indicar que se encuentra a la espera de una orden es mostrar un símbolo, denominado *prompt* del sistema, seguido del cursor. Habitualmente, el *prompt* del sistema es el carácter $ o el carácter % para los usuarios y el carácter # para el administrador del sistema (usuario *root*).

Un usuario introduce una orden escribiéndola en la línea de comandos a continuación del *prompt*. Para indicar al sistema que debe ejecutarla, pulsa la tecla de retorno de carro (↵). Entonces, el sistema ejecuta la instrucción. Cuando finaliza, vuelve a mostrar el *prompt* y el cursor para esperar por otra nueva orden.

Es importante resaltar que Linux tiene diferentes tipos de shell como son, entre otros, los siguientes:

Bourne Shell	sh	Es el original de AT&T, y está disponible en todas las máquinas UNIX.
C Shell	csh	Desarrollado como parte del UNIX BSD.
Korn Shell	ksh	Mejora de AT&T del Bourne Shell.
Bourne again Shell	bash	Versión distribuida con Linux.

Todos ellos tienen características que los diferencian, pero también tienen muchos puntos en común. Uno de ellos, especialmente importante, radica en su sensibilidad al empleo de caracteres en mayúsculas o en minúsculas.

Cuando se dispone de un sistema de ventanas (Windows, X-Windows), se puede dar órdenes al sistema directamente con el sistema apuntador (ratón) y también desde una ventana de *shell*.

En Linux se abre una ventana de *shell* eligiendo en el menú *sistema* las opciones *consola* $\longrightarrow$

3. Órdenes básicas.

Cambio de la contraseña de usuario

Es la primera operación que debe realizarse cuando se obtiene una cuenta de usuario en un computador con sistema de acceso protegido mediante contraseña. El objetivo es disponer de una contraseña fácil de recordar por parte del usuario y difícil de descubrir por parte de otras personas.

Sintaxis: `passwd`

Habitualmente, el sistema solicita que se introduzca primero la contraseña antigua y, a continuación, dos veces la nueva contraseña. De esta manera, se reduce el riesgo de introducir por error una contraseña no deseada.

Ayuda de Linux

La orden man proporciona ayuda acerca de las diferentes instrucciones de Linux

Sintaxis: `man [-k] orden`

Nota: Los corchetes indican que los parámetros que aparecen entre ellos son opcionales.

Cuando se utiliza el parámetro **–k**, el resultado es un listado con los nombres de todas las instrucciones que tienen alguna relación con la palabra `orden`.

La orden apropos tiene la misma utilidad que *man* cuando se utiliza con el parámetro –k.

Sintaxis: `apropos orden`

Así por ejemplo, los resultados de ejecutar las siguientes órdenes desde la línea de comandos son los que se indican:

`man kill`	Visualiza la ayuda de la orden *kill*.
`man –k printer`	Muestra los nombres de todas las órdenes relacionadas con las impresoras.
`apropos password`	Muestra los nombres de todas las órdenes relacionadas con las constraseñas

Cierre de una ventana de shell.

La orden exit cierra la ventana de shell activa. Cuando no se dispone de un entorno de ventanas, dicha orden se utiliza para abandonar la sesión.

Sintaxis: `exit`

Almacenamiento de información: archivos y subdirectorios.

La información que se guarda en un sistema de almacenamiento masivo (disco duro) se almacena en archivos organizados en un árbol de subdirectorios.

Las tres características fundamentales de la organización de archivos UNIX son las siguientes:

1. No existe el concepto de extensión de un archivo. De hecho, el carácter punto [.] no tiene ningún significado especial. Así, por ejemplo, los siguientes nombres de archivo son válidos para un programa ejecutable:

   juego
   juego.exe
   juego.doc
   juego.al.mus.con.mis.amigos.doc.exe.bat.ya

2. El árbol de subdirectorios comienza en el directorio raíz, que se nombra con el carácter /. En el caso de disponer de varias unidades físicas de almacenamiento, éstas aparecen como subdirectorios dentro del raíz.

3. Los subdirectorios son considerados por el sistema operativo de igual forma que si fueran archivos.

Se conoce como ruta de un archivo (*path*) al camino que hay que recorrer para encontrar dicho archivo dentro del árbol de subdirectorios.

Existen dos formas diferentes de especificar la ruta de un archivo:

1. Ruta absoluta. Camino completo desde el directorio raíz.

2. Ruta relativa. Camino que hay que recorrer desde el subdirectorio actual.

Los nombres de los diferentes nodos se separan con el carácter /.

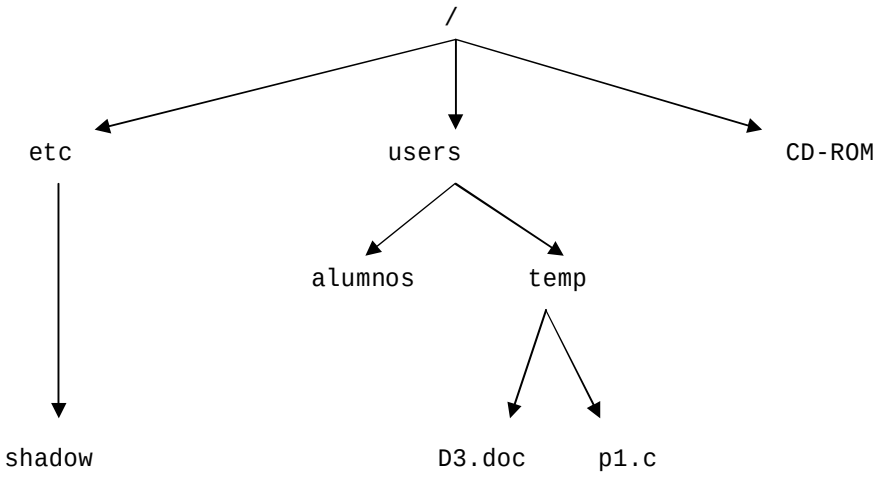
En todos los subdirectorios aparecen dos archivos especiales:

. $\longrightarrow$ Nombra al subdirectorio actual.

.. $\longrightarrow$ Nombra al subdirectorio padre del actual.

Ejemplo:

Sea el siguiente árbol de subdirectorios:



Si el subdirectorio actual es `/etc`, entonces:

Ruta absoluta de p1.c `/users/temp/p1.c`

Ruta relativa de p1.c `../users/temp/p1.c`

Ruta relativa de shadow `./shadow`

Las principales órdenes relacionadas con la jerarquía de subdirectorios son las siguientes:

`pwd` : Devuelve la ruta absoluta del subdirectorio actual:

Sintaxis: pwd

`mkdir`: Crea un subdirectorio dentro del subdirectorio actual o, si se indica la ruta, en el destino especificado.

Sintaxis: `mkdir nombre_subdirectorio`

cd: Permite navegar por la estructura de subdirectorios.

Sintaxis: `cd nombre_subdirectorio`

Así, por ejemplo, para salir del subdirectorio actual se especifica el nombre de su directorio padre:

```
cd ..
```

Para acceder al directorio propio del usuario, es posible utilizar dos variantes:

```
cd
cd $HOME
```

Nota: $HOME es una variable de entorno que almacena la ruta absoluta del directorio por defecto del usuario. Su valor se puede obtener con la orden: `echo $HOME`.

Manipulación de archivos.

ls: Vuelca en pantalla los nombres de los archivos (incluídos los subdirectorios) que hay en el subdirectorio actual.

Sintaxis: `ls [-la]`

Con la opción **`-l`** visualiza información completa de los archivos, entre la que se incluyen los permisos de acceso para cada uno de ellos.

Con la opción **`-a`** visualiza los nombres de archivos especiales (aquellos cuyo nombre comienza con el carácter punto (`.`).

touch: crea un archivo vacío.

Sintaxis: `touch nombre_archivo`

Nota: En adelante, cuando se haga referencia al nombre de un archivo, se entenderá como tal su ruta, bien absoluta, bien relativa.

rm: Borra un archivo.

Sintaxis: `rm nombre_archivo`

rmdir: Borra un subdirectorio que se encuentre vacío.

Sintaxis: `rmdir nombre_subdirectorio`

cp: Hace una copia de un archivo.

Sintaxis: `cp archivo_original archivo_destino`

mv: Cambia el nombre de un archivo o lo mueve de un subdirectorio a otro.

Sintaxis: `mv archivo_original archivo_destino`

cat: Concatena varios archivos y los muestra por la salida estándar (pantalla). Se utiliza, entre otras cosas, para visualiza el contenido de archivos de texto.

Sintaxis: cat archivo_1 <archivo_i>

Nota: Cuando los parámetros de una orden aparecen entre los caracteres < >, significa que pueden utilizarse 0, 1 ó más veces.

more: Muestra por la salida estándar el contenido de un archivo de texto. A diferencia de la orden *cat*, espera por la pulsación de una tecla para continuar mostrando el contenido del documento una vez que se ha llenado la pantalla. Si se pulsa la barra espaciadora, se muestra la siguiente pantalla completa; si la tecla pulsada es el retorno de carro (↵), se muestra la siguiente línea; si se pulsa la letra v, se edita el archivo con el editor vi.

Sintaxis: more archivo_1 <archivo_i>

4. El editor vi.

El editor vi (visual editor) es el editor típico de Linux y, aunque no es sencillo de utilizar, es el único que se encuentra con seguridad en cualquier sistema Linux.

Sintaxis: `vi [nombre_de_archivo]`

Modos de trabajo.

vi se encuentra en todo momento en uno de los siguientes tres modos: modo de comandos, modo de inserción o modo de última línea.

- Modo de comandos. Es el modo por defecto en el que inicia vi. Permite usar ciertos comandos para editar archivos o para cambiar a otros modos.
- Modo de inserción. Se utiliza para insertar o editar texto. Para salir de este modo y volver al modo de comandos se pulsa la tecla escape [ESC].
- Modo de última línea. Es un modo especial para ciertos comandos extendidos. Se accede a este modo pulsando [:], a continuación se introduce el comando (que aparecerá en la última línea de la pantalla) y se pulsa retorno de carro para ejecutarlo.

Edición con vi

Para escribir texto, es necesario pasar al modo de inserción. Esto se puede conseguir con diferentes comandos (pulsación de teclas desde el modo de comandos):

`i`	Insertar texto (a la izquierda del cursor).
`a`	Añadir texto (a la derecha del cursor).
`O`	Insertar una línea por encima de la actual.
`o`	Insertar una línea por debajo de la actual.
`x`	Borra el carácter situado bajo el cursor.
`dw`	Borra la palabra en la que está situado el cursor.
`dd`	Borra la línea en la que se encuentra el cursor.
`yy`	Copia la línea en la que se encuentra el cursor.
`p`	Inserta una línea debajo de la actual y pega la que se ha copiado.
`h`	Mueve el cursor una posición hacia la izquierda.
`l`	Mueve el cursor una posición hacia la derecha.
`j`	Mueve el cursor una posición hacia abajo.
`k`	Mueve el cursor una posición hacia arriba.

Desde el modo de última línea (al que se accede pulsando [:] en modo de comandos), también se pueden realizar las siguientes tareas:

`w [nombre_archivo]`	Guarda el archivo actual. Si se indica el nombre, se guarda con el nombre suministrado.
`q!`	Abandona vi sin guardar los cambios.
`wq` o bien `x`	Abandona vi guardando los cambios.
`/patron`	Busca `patron` en el archivo que se edita.

5. El Compilador *cc*.

El compilador `cc` viene incluido en el sistema operativo UNIX.

Sintaxis: `cc [-o archivo_ejecutable] archivo_fuente`

El archivo con el código fuente debe terminar obligatoriamente con los caracteres '**`.c`**' cuando se trata de código escrito en lenguaje C.

Si no se especifica el nombre del archivo ejecutable, éste se llamará **`a.out`**.

6. Ejecución de Procesos.

La forma de ejecutar un programa es escribir su ruta en la línea de órdenes.

Ejemplo:

```
./programa
```

Es posible ejecutar varios programas secuencialmente introduciendo sus rutas en la misma línea utilizando el carácter punto y coma [;] como separador entre ellas.

Cuando se da la orden de ejecutar un programa, el *shell* se bloquea y no permite la introducción de nuevas órdenes hasta que finalice el proceso iniciado. Esta situación puede obviarse ejecutando el primer proceso en *background*, lo que se consigue colocando el carácter ampersand (&) inmediatamente detrás del nombre del programa.

Ejemplo de ejecución de un proceso en *background*:

```
./programa &
```

7. Eliminación de Procesos.

Cada proceso que se ejecuta en un instante dado se identifica por un número único denominado pid (identificador de proceso).

La orden ps permite visualizar la información (incluido el pid) de los procesos que se están ejecutando en un sistema:

Sintaxis: `ps [-ef] [-u login]`

Con los parámetros –ef se obtiene información completa de todos los procesos. Con el parámetro –u se obtiene información de todos los procesos del usuario identificado por *login*.

Hay ocasiones en las que un proceso no finaliza adecuadamente y permanece en el sistema durante tiempo indefinido. UNIX proporciona la orden kill para eliminar tanto estos como cualesquiera otros procesos.

Sintaxis: `kill –9 pid`

La orden kill se utiliza de modo genérico para enviar señales a procesos. El parámetro –9 especifica que la señal enviada *mata* al proceso identificado por el número pid.

8. Comunicación entre usuarios.

Existen diferentes formas de comunicación entre los usuarios de un sistemaLinux:

write: Permite enviar un mensaje a un usuario que se encuentre conectado en otro terminal.

Sintaxis: `write login_destinatario`

El shell no muestra el prompt del sistema, sino que lee líneas desde el teclado y las envía al destinatario. El proceso finaliza cuando el remitente pulse la combinación de teclas [Control]+[D].

talk: Establece una comunicación bidireccional entre dos usuarios.

Sintaxis: `talk login_usuario`

La orden talk avisa al usuario destinatario de que otro usuario solicita comunicarse con él. Para comenzar la comunicación, el destinatario debe ejecutar también la orden talk indicando el login del usuario solicitante. Entonces, las líneas de texto que escriba cada uno de los usuarios aparecerá en la pantalla del otro. Para finalizar la conversación, se pulsan simultáneamente las teclas [Control]+[C].

mail: Utilidad de correo electrónico.

mesg: Permite habilitar / deshabilitar la recepción de mensajes vía *write* o *talk*.

Sintaxis: `mesg [y | n]`

Sin parámetros, devuelve el estado actual.

El parámetro **y** habilita la recepción de mensajes.

El parámetro **n** inhibe la recepción de mensajes.

who: Muestra una lista de los usuarios conectados al sistema en el momento actual.

Sintaxis: `who`

whodo:Muestra una lista de los usuarios actualmente conectados y los procesos que están ejecutando.

Sintaxis: `whodo`

whoami: Muestra el *login* del usuario que la ejecuta.

Sintaxis: `whoami`

who am i: Muestra información más detallada que *whoami*.

Sintaxis: `who am i`

9. Desarrollo de la práctica.

1. Entra en el sistema con tu login y tu contraseña.
2. Cambia tu contraseña de usuario:

   `passwd`
3. Obtén la ruta completa del subdirectorio en el que te encuentras, es tu directorio por defecto:

   `pwd`
4. Crea un subdirectorio de nombre SO dentro de tu directorio por defecto:

   `mkdir SO`
5. Entra en el nuevo subdirectorio:

   `cd SO`
6. Comprueba que te encuentras en el subdirectorio SO:

   `pwd`
7. Sal al directorio padre de SO:

   `cd ..`
8. Crea otro subdirectorio dentro de SO llamado Practica1 y entra en él.

   `mkdir SO/Practica1`

   `cd SO/Practica1`
9. Crea un archivo vacío dentro de Práctica1 llamado *nulo*:

   `touch nulo`
10. Comprueba que este archivo se ha creado y bórralo:

    `ls`

    `rm nulo`
11. Sal del directorio Practica1 y bórralo:

    `cd ..`

    `rmdir Practica1`

12. Inicia el editor de textos vi para editar el archivo *duerme.c*:

    `vi duerme.c`

13. Inserta la siguiente línea de texto:

    `/* Practica 1 */`

14. Vuelve al modo de comandos pulsando [ESC], desplaza el cursor y colócalo justo antes de la palabra `Practica`. Inserta a la izquierda del cursor las palabras: `Esta es la`.

15. Inserta al principio del documento la siguiente línea utilizando el comando O de vi:

    `#include <stdio.h>`

16. Inserta al final del documento el siguiente texto utilizando el comando o de vi:

    `Void main()`

17. Borra la letra V e inserta la letra v (minúscula) antes de *oid*.

18. Borra la palabra main empleando el comando dw.

19. Elimina la línea `#include <stdio.h>` utilizando el comando dd.

20. Guarda el archivo sin salir de vi. Para ello, se debe teclear :w desde el modo de comandos (pasa al modo de última línea y ejecuta la orden w)

21. Completa el documento hasta tener el siguiente código fuente en lenguaje C:

```
/* Practicas de Linux */
/* Práctica 1 */

void main(){
   sleep(10);
   exit(0);
}
```

22. Salva el programa y sal del editor. Para ello teclea la orden :wq.

23. Haz una copia del documento de texto duerme.c:

    `cp duerme.c duerme.old`

24. Genera una lista detallada de los archivos que hay en el directorio actual:

    `ls -l`

25. Borra el documento duerme.old:

    `rm duerme.old`

26. Comprueba con la instrucción *ls* que realmente se ha borrado *duerme.old*.

27. Compila el archivo *duerme.c* y genera un ejecutable de nombre *duerme*.

```
cc –o duerme duerme.c
```

28. Ejecuta el programa:

```
./duerme
```

29. Cambia el archivo *duerme.c* para que la instrucción *sleep* reciba como parámetro el valor 100000.

30. Guárdalo, compílalo y ejecútalo en *background*:

```
./duerme &
```

31. Aparentemente, no ocurre nada, pero el programa se está ejecutando. Para comprobarlo, utiliza la instrucción *ps* con tu login de usario.

```
ps –u login
```

32. Busca la fila en la que aparece el proceso `duerme`, anota su pid.

33. Mata al proceso `duerme` utilizando la orden *kill*:

```
kill –9 pid
```

34. Comprueba que el proceso duerme ya no está ejecutándose.

35. Utiliza el programa `duerme` para simular el caso de un proceso que está colgado, es decir, que está bloqueado y no se comporta correctamente. Entonces, habrá que matarlo desde otra ventana de shell.

    Ejecuta: `./duerme`

    La ventana queda bloqueada. Abre otra y mata al proceso duerme desde ella (utiliza, *ps* y *kill*). Observa cómo la shell anterior se desbloquea.

Resumen de órdenes Linux:

Instrucción	Utilidad
cat file1 file2 ...	Concatena varios archivos (o uno solo) y los muestra por pantalla.
cc –o archivo archivo.c	Compila *archivo.c* y genera un ejecutable con nombre *archivo*.
cd	Entra a un subdirectorio.
clear	Borrar la pantalla.
cp archivo1 archivo2	Copia archivo1 con el nombre archivo2
date	Fecha y hora.
exit	Abandona la sesión de UNIX o cierra la ventana de shell.
kill –9 pid	Mata el proceso identificado por pid.
ls	Ver los archivos contenidos en el subdirectorio actual.
ls –l	Ver una descripción detallada de los archivos.
ls –la	Ver una descripción detallada incluyendo los archivos ocultos.
mail usuario	Enviar un correo a un usuario (Ctrl-D para salir y enviar).
man [-k] orden apropos palabra	Ver la ayuda de una orden o un listado de órdenes
mesg n / mesg y	Desactiva / Activa la recepción de mensajes.
mkdir	Crea un subdirectorio.
more archivo	Vuelca página a página el contenido de un archivo de texto.
ps –ef	Listado completo de los procesos que se están ejecutando en la máquina.
ps –lc	Información detallada de los procesos hijos de la shell en la que se está trabajando.
ps –u login	Listado de los procesos de un usuario concreto.
pwd	Ver el path o ruta del subdirectorio actual.
rm archivo	Borra un archivo.
rmdir subdirectorio	Borra un subdirectorio que esté vacío.
talk usuario	Hablar con un usuario que esté conectado.
touch file	Crea un archivo vacío de nombre *file*.
uname –a	Ver el nombre y las propiedades del sistema.
vi archivo	Edita *archivo* con el editor vi.
who	Ver los usuarios conectados al sistema.
whoami who am i	Ver las propiedades del usuario.
whodo	Ver qué hacen los usuarios que están conectados.
write usuario	Escribir un mensaje a un usuario que esté conectado.

Resumen de comandos vi:

Comando	Utilidad
`i`	Insertar texto delante del cursor.
`a`	Insertar texto detrás del cursor.
`h`	Desplazar el cursor hacia la izquierda.
`j`	Desplazar el cursor hacia abajo.
`k`	Desplazar el cursor hacia arriba.
`l`	Desplazar el cursor hacia la derecha.
`O`	Inserta una línea encima de la actual.
`o`	Insertar una línea debajo de la actual.
`x`	Borrar el carácter situado bajo el cursor.
`dw`	Borrar una palabra completa.
`dd`	Borrar una línea completa.
`:w`	Guardar el archivo sin salir de vi.
`:wq` *o bien* `:x`	Guardar el archivo y salir de vi.
`:q!`	Salir de vi sin guardar el archivo.
`yy`	Copiar la línea actual
`p`	Pegar

Nota: Al modo de comandos se accede pulsando [ESC]