Práctica 1. Introducción al Sistema Operativo Unix

1.- Objetivos.

- Utilizar comandos elementales de UNIX.
- Utilizar el funcionamiento básico del editor de textos vi de UNIX.
- Compilar y ejecutar un programa escrito en lenguaje C.
- Modificar la fuente de datos de entrada o el destino de los resultados de órdenes UNIX.
- Filtrar el contenido de archivos de texto para seleccionar solamente parte de la información que contienen.

2.- El editor vi.

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

Sintaxis: `vi [nombre_de_fichero]`

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 se inicia ***vi***. Permite utilizar ciertos comandos para editar ficheros 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 ella 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.

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

`w [nombre_fichero]`	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 fichero que se edita.

4.- El compilador cc.

El compilador cc viene incluido en el sistema operativo UNIX.

Sintaxis: `cc [-o fichero_ejecutable] fichero_fuente`

- El fichero con el código fuente debe terminar obligatoriamente con los caracteres '`.c`' cuando se trata de un código escrito en lenguaje C.
- Si no se especifica el nombre del archivo ejecutable, este se llamará ***a.out***.

5.- 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 evitarse ejecutando el primer proceso en segundo plano (***background***), lo que se consigue colocando el carácter ampersand (&) inmediatamente detrás del nombre del programa.

Ejemplo de ejecución de un proceso en ***background***: `./programa &`

6.- 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 un tiempo indefinido. UNIX proporciona la orden ***kill*** para eliminar tanto estos procesos 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 con el número `pid`.

7.- Desarrollo de la práctica.

Manejo del editor vi.

1. Utiliza el editor vi para crear un documento llamado ***holamundo.c*** que contenga el siguiente código en lenguaje C:

```
/* Sistemas operativos *
 * Práctica 1          */

#include <stdio.h>
int main() {
   printf("¡Hola mundo!\n");
   return 0;
}
```

Manejo del compilador cc y ejecución de procesos.

2. Compila el archivo anterior utilizando ***cc*** para generar un archivo ejecutable llamado ***holamundo***.

3. Ejecuta el programa ***holamundo*** generado en el punto anterior.

4. Modifica el documento ***holamundo.c*** para que escriba por pantalla '*¡Hola!*', espere 1 segundo y escriba por pantalla 'Ha/n pasado 1 segundo/s', espere otro segundo y escriba por pantalla 'Ha/n pasado 2 segundo/s, así hasta que transcurridos 10 segundos, escriba por pantalla '*¡Mundo!*'.

   Para ello puedes utilizar la función `sleep(X);` de C que le dice al sistema que suspenda la ejecución del proceso actual durante `X` segundos.

   Para escribir por pantalla utiliza la función `printf("XXX\n");` de C que le dice al sistema que escriba por pantalla XXX. `\n` le indica al sistema que incluya un retorno de carro al final. También puedes indicarle al sistema que escriba el valor de una variable de tipo entero de la siguiente manera:

```
int var=0;
printf("El valor de la variable var es %d\n", var);
```

5. Guárdalo, compílalo y ejecútalo.

Eliminación de procesos.

6. Modifica una vez más ***holamundo.c*** para que espere 100 segundos en lugar de 10. Guárdalo, compílalo de nuevo y ejecútalo en segundo plano (*background*).

7. Mientras se está ejecutando ***holamundo*** utiliza el comando ***ps*** para obtener su ***pid***.

8. Mata el proceso ***holamundo*** utilizando la orden ***kill***.

9. Utiliza el proceso ***holamundo*** 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 el proceso normalmente. La ventana queda bloqueada hasta que termine de ejecutarse. Abre otra ventana de ***shell*** y mata al proceso ***holamundo*** desde ella (utiliza ***ps*** y ***kill***). Observa como la ***shell*** anterior se desbloquea.