Práctica 1: Comandos

Objetivos

- Conocer el terminal en un entorno GNU/Linux.
- Manejar el SO Ubuntu a nivel de usuario desde un terminal.

1. ¿Qué es un Sistema Operativo?

A grandes rasgos, se denomina sistema operativo (SO) a un conjunto de programas que nos permiten interactuar con nuestro sistema informático. Permite la interacción entre un programa cualquiera y el hardware del computador, gestionando los recursos del mismo.

2. GNU/Linux vs Windows

Entre otros, Windows y Linux son dos sistemas operativos que nos permiten trabajar con nuestros ordenadores. Linux se diferencia del resto de sistemas operativos en:

- es *libre*: no tenemos que pagar ningún tipo de licencia a ninguna casa desarrolladora de software por el uso del mismo.
- el sistema viene acompañado del *código fuente*.

El código fuente es el conjunto de instrucciones que forman un programa; describe detalladamente qué hace, cuál es su funcionamiento. Son las órdenes que se le dan al computador para que resuelva un problema en concreto.

El sistema operativo Linux lo forman el núcleo del sistema (*kernel*) más un gran número de programas/bibliotecas que hacen posible su utilización. Muchos de estos programas y bibliotecas han sido posibles gracias al proyecto GNU. Por este motivo, muchos llaman a Linux GNU/Linux, para resaltar que el

sistema lo forman tanto el núcleo como gran parte del software producido por el proyecto GNU. Linux se distribuye bajo la GNU *General Public License,* por lo tanto, el código fuente tiene que estar siempre accesible y cualquier modificación o trabajo derivado tiene que tener esta licencia.

A diferencia de Linux, Windows es un sistema cerrado donde no conocemos ni podemos acceder a todo el código fuente ni comprender (a bajo nivel) cómo funciona.

3. ¿Qué es Ubuntu?

Linux es un sistema de libre distribución por lo que se pueden encontrar todos los ficheros y programas necesarios para su funcionamiento en multitud de servidores conectados a Internet. La tarea de reunir todos los ficheros y programas necesarios, así como instalarlos en un sistema y configurarlo, puede ser bastante complicada. Por ello, nacieron las llamadas *distribuciones de Linux*. Es decir, hay empresas y organizaciones que se dedican a hacer el trabajo "sucio" para el beneficio y comodidad de los usuarios.

Una distribución es una recopilación de programas y ficheros, organizados y preparados para su instalación. Estas distribuciones se pueden obtener a través de Internet, o comprando los CDs de las mismas. Contienen todo lo necesario para instalar un sistema Linux bastante completo y en la mayoría de los casos proporcionan un programa que facilita la instalación. Casi todos los principales distribuidores de Linux, ofrecen la posibilidad de bajarse sus distribuciones, vía FTP (sin cargo alguno).

Ubuntu es una distribución de GNU/Linux, por lo tanto es un Sistema Operativo. Está basada en Debian, que es otra distribución de Linux. La facilidad de instalación, mantenimiento y uso lo han convertido en un sistema operativo muy extendido actualmente.

4. Entorno Ubuntu

Hay varias distribuciones de Ubuntu las cuales difieren en el tipo de paquetes que proporcionan. Todas se basan en un kernel Linux pero cada una utiliza un sistema de ventanas o un conjunto de paquetes diferentes según la filosofía o a qué público van dirigidas. Algunos ejemplos son: Ubuntu, Kubuntu, Xubuntu, etc. En nuestro caso la distribución elegida es Ubuntu y será la que utilicemos en el aula.

5. Usuarios y permisos

Linux es multitarea y multiusuario, esto quiere decir que varias personas podrían conectarse en un momento dado a la máquina para trabajar con ella. Cada vez que tengamos que acceder a una máquina, necesitaremos el usuario y la contraseña. Cada usuario tiene una serie de permisos que en un momento dado puede impedirnos la instalación o ejecución de ciertos programas.

6. ¿Qué es la *shell*?

La *shell* es una interfaz que permite la comunicación entre el usuario y el ordenador a través del sistema operativo. Esta interfaz se conoce generalmente como *shell* o línea de comandos. Dependiendo de dónde estemos trabajando, la *shell* será accesible desde un interfaz gráfico (gnome-terminal, konsole, xterm ...) o directamente se trabajará sobre la línea de comandos (típica ventana o pantalla de letras blancas/verdes con fondo negro).

Esta herramienta nos permite hacer las mismas (y más) cosas que desde el interfaz gráfico (crear carpetas, borrar carpetas, ver archivos, etc) sin tener que utilizar un ratón.

La *shell* se alimenta de una serie de comandos que son los que permitirán realizar la acción deseada. Por ejemplo cuando abrimos una carpeta en un interfaz gráfico, el sistema operativo muestra todos los archivos contenidos en esa carpeta. Cuando trabajamos en una consola y estamos en una ruta determinada (posición de la carpeta en tu sistema), al escribir en la *shell* el comando `$ ls` será posible mostrar todos los archivos contenidos en esa carpeta.

El sistema operativo indica que se encuentra a la espera de órdenes en la *shell* cuando muestra el *prompt* del sistema seguido del cursor. El *prompt* es un símbolo; en nuestro caso el carácter `$`. Para indicar al sistema que ejecute una orden, hay que pulsar la tecla `intro` después del comando. Cuando finaliza, vuelve a mostrar el *prompt* y el cursor, pudiendo introducir un nuevo comando.

7. ¿Qué son los comandos?

Los comandos son una serie de órdenes que ejecutadas sobre una *shell* nos permiten que el sistema operativo lance una serie de mecanismos que devuelvan al usuario el objetivo esperado (`ls` lista los archivos de un directorio, `cd` cambia de ruta, `rm` borra ficheros, etc).

Los comandos pueden tener parámetros de entrada o argumentos que servirán de entrada o salida del comando que estamos ejecutando y que nos per-

mitirán leer o modificar los ficheros o su contenido.

Por ejemplo, si en una ruta dada del sistema (posición dentro de nuestra máquina) escribimos `$ls nombre_carpeta`, nos será mostrado el contenido de dicha carpeta sin acceder a ella. De la misma manera que cuando navegamos con el explorador de archivos pulsamos en los símbolos + que acompañan a las carpetas.

Los comandos también pueden ser comparables a los denominados ejecutables de windows. Por poner un ejemplo, cuando tenemos un archivo de texto y lo abrimos con el *notepad* (Windows) o *gedit* (Ubuntu) estamos pasándole un parámetro (el archivo) para que lo abra en el interfaz gráfico. Cuando estemos trabajando en el terminal podremos utilizar `$ nano nombre_del_archivo` para abrir el documento con el editor *nano* (existen otros como el *emacs*, *vim,* etc).

Generalmente encontraréis diferenciados unos y otros por comandos internos y comandos externos, es decir comandos que la propia *shell* interpreta (`cd`, `ls`, `rm`, etc) en cualquier parte del sistema o comando externos que la *shell* necesita saber dónde están para ejecutarse.

8. Administración desde terminal

Vamos a probar en esta clase una serie de comandos para comprobar que para la gestión de un ordenador no se necesitan poderosos interfaces gráficos. Tenemos un par de formas de implementar esta práctica:

1. Abriendo el terminal (Aplicaciones $\rightarrow$ Accesorios $\rightarrow$ Terminal) o bien pulsando en el icono de la esquina superior izquierda y escribiendo Terminal.

2. Pulsando `Ctrl+Alt+F1` (para regresar al interfaz gráfico basta con pulsar `Ctrl+Alt+F7`).

Ante cualquier duda o necesidad que vayamos encontrado a lo largo de la práctica utilizaremos el comando

```
~$ man [nombre_comando]
```

que nos muestra toda la información que contiene la *shell* acerca del comando.

Además, la orden `apropos` nos da un listado con los nombres de todos los comandos que tienen alguna relación con la palabra indicada.

```
~$ apropos [palabra]
```

9. Carpetas y directorios

La información almacenada en un dispositivo de almacenamiento (disco duro) se guarda en archivos organizados en un árbol de subdirectorios (carpetas). En Linux, el árbol de subdirectorios comienza en el directorio raíz, que se nombra con el carácter `/`. Si tenemos más de un dispositivo de almacenamiento, aparecen como si fueran subdirectorios dentro del raíz.

Denominamos ruta de un archivo (*path*) al camino que hay que seguir para encontrar dicho archivo dentro del árbol de subdirectorios. Podemos especificar una ruta de forma **absoluta** (camino completo desde el directorio raíz) o **relativa** (camino que hay que recorrer desde el subdirectorio actual). Los nombres de los distintos subdirectorios dentro del árbol se separan con el carácter `/`. Un ejemplo de ruta absoluta sería `/home/ubuntu`. Un ejemplo de ruta relativa sería `Escritorio/Carpeta/Archivo`

En todos los directorios tenemos dos archivos especiales:

- `.` Hace referencia al directorio actual.
- `..` Hace referencia al directorio padre del actual.

Vamos a recorrer el árbol de directorios del sistema. Para ello utilizaremos los comandos `cd`, `ls` y `pwd`:

```
~$ ls
```

Nos mostrará los archivos contenidos en el *path* actual.

```
~$ pwd
```

Nos mostrará la ruta en la que nos encontramos.

```
~$ cd nombre_de_carpeta
```

Nos permitirá acceder al directorio especificado.

```
~$ cd ..
```

Podremos regresar al directorio que contiene la ruta actual (subir un nivel).

```
~$ cat [nombre_de_archivo]
```

Muestra el contenido del archivo pasado como argumento.

```
~$ more [nombre_de_archivo]
```

Muestra el contenido del archivo pasado como argumento de forma paginada.

```
~$ mkdir nombre_de_carpeta
```

Crea una carpeta de nombre `nombre_carpeta` (se puede pasar una ruta completa).

```
~$ rm nombre_de_archivo
```

Borra el archivo especificado (se puede pasar una ruta completa).

```
~$ rmdir nombre_de_carpeta
```

Borra la carpeta indicada si se encuentra vacía (se puede pasar una ruta completa).

```
~$ cp archivo_original archivo_destino
```

Crea una copia del archivo original (se pueden pasar dos rutas completas).

```
~$ mv archivo_original archivo_destino
```

Cambia el nombre de un archivo o lo mueve de un directorio a otro (se pueden pasar dos rutas completas).

10. Gestión del software

10.1. Instalación de aplicaciones

Aunque para esta práctica no va a ser necesario, existen varias maneras de instalar nuevo software en un sistema Linux.

1. Obteniendo el código fuente, compilando y ejecutando el programa.
2. Obteniendo el fichero precompilado y ejecutándolo en el sistema con `./` o `sh`
3. Utilizando la aplicación `apt-get` disponible en el sistema Ubuntu. En este caso, los siguientes comandos son de utilidad.

```
~$ wget http://lugar_web_de_la_aplicacion
```

Obtenemos el programa de la página correspondiente.

```
~$ tar [argumento] fichero.tar
```

Desempaquetamos el fichero.

```
~$ unzip fichero.zip
```

Descomprimimos el fichero `.zip`

```
~$ rar x fichero.rar
```

Descomprimimos el fichero `.rar`

```
~$ apt-get install programa
```

Instalamos el programa de los repositorios oficiales.

```
~$ apt-get update
```

Actualizamos la base de datos de los programas con respecto a los repositorios oficiales.

```
~$ apt-get upgrade
```

Actualizamos nuestro sistema con respecto a los repositorios oficiales.

11. Gestión del hardware

En los sistemas GNU/Linux se puede obtener la información del hardware de nuestro ordenador:

```
~$ lshw
```

Lista de forma general todo el hardware accesible en tu sistema operativo. La información mostrada depende del usuario que lo lanza.

`~$ lsusb`

Lista de forma genérica cualquier dispositivo conectado a los puertos USB de tu máquina.

`~$ free`

Muestra el tamaño total de la memoria RAM de tu sistema.

`~$ df`

Muestra la información de las particiones del disco duro accesibles por tu SO.

12. Otros comandos

Otros comandos interesantes son:

`~$ grep [elemento] [nombre_de_archivo]`

Muestra el elemento dentro del archivo pasado como argumento.

`~$ wc [argumento] [nombre de archivo]`

Cuenta palabras, líneas o letras.

Pipes: <, >, |

Redirecciona las salidas estándar de los comandos a donde nosotros queramos (p.e. un fichero). Por ejemplo, con la instrucción `pwd` muestra la ruta donde nos encontramos. Si la quiero guardar en un archivo, tendría que poner `pwd >archivo.txt`. Si quiero contar cuántas líneas tiene, usaría: `pwd | wc -l`

13. Usuarios y permisos

Los SSOO tienen diferentes sistemas de seguridad atendiendo al usuario que haya accedido al sistema. Es necesario presentarle las credenciales del usuario para que el SO nos permita o no hacer algo en el entorno.

`~$ ls -l`

Lista los archivos en cierta ruta mostrando los permisos de cada uno de los elementos. En la columna de la izquierda aparece una columna con 7 caracteres (por ejemplo `drwxr-xr-x`):

- el primer carácter es `-` ó `d`. Si aparece una `d` indica que es un directorio.
- grupo de 3 caracteres siguientes: permisos del usuario sobre el archivo/directorio. Si aparece un guión (`-`) es que no tiene ese permiso. La `r` indica lectura (puede leerlo), la `w` escritura (puede modificarlo/borrarlo) y la `x` ejecución (puede ejecutarlo).
- el siguiente grupo de 3 caracteres: permisos del grupo al que pertenece el usuario sobre el archivo/directorio.
- el último grupo de 3 caracteres: permisos del resto de usuarios sobre el archivo/directorio.

`~$ chmod [argumento] [archivos]`

Cambia los permisos sobre cada uno de los archivos indicados. Los nuevos permisos se especifican en `[argumento]`. Los valores que puede tomar son `[+-wxr]` o bien `[000 777]`.

- Con `+` se asignan permisos, con `-` se quitan permisos.
- Los permisos pueden ser: de escritura (`w`), ejecución (`x`) y lectura (`r`).
- Si se le añade la letra `g` y/o la `o` especifica que se modifiquen los permisos al grupo y/o resto de usuarios. Con la letra `u` se refiere al usuario propietario del archivo. Usando la letra `a`, se refiere a todos los usuarios.
- En cuanto al uso de números del 000 al 777, cada dígito indica el permiso a cada clase de usuario. De izquierda a derecha, propietario, grupo y resto de usuarios. La relación de cada dígito con los permisos, aparece en la tabla.

Número	Binario	Lectura (r)	Escritura (w)	Ejecución (x)
0	000	No	No	No
1	001	No	No	Sí
2	010	No	Sí	No
3	011	No	Sí	Sí
4	100	Sí	No	No
5	101	Sí	No	Sí
6	110	Sí	Sí	No
7	111	Sí	Sí	Sí

14. Gestión del software

14.1. Gestión de procesos

Existen varias maneras de supervisar y gestionar los programas que se están ejecutando en el sistema. Los programas en GNU/Linux tienen un número identificativo del mismo llamado PID (*Process ID*). Este número es único para cada programa.

```
~$ top
```

Muestra todos los procesos lanzados en el sistema y el estado del mismo.

```
~$ kill [argumento] proceso
```

Manda sobre un proceso que se está ejecutando en el sistema la orden escrita en el argumento. El argumento `-9` finaliza (*mata*) el proceso especificado.

```
~$ ps [argumento]
```

Sin argumentos muestra los procesos asociados al terminal desde donde se lanza. Con el argumento `-ef` se obtiene información completa de todos los procesos.

```
~$ pstree
```

Muestra en forma de árbol todos los procesos que se están ejecutando en el sistema.

15. Ejercicios Propuestos

15.1. Ejercicio

Crea la siguiente estructura de directorios desde tu `home` siendo `CUENTA_LOGIN` el nombre de usuario con el que has entrado al sistema y GrupoX tu grupo de prácticas B3 (1, 2, 3 ó 4).

`/home/CUENTA_LOGIN/PRACTICA_GRUPOX/ALUMNO/NOMBRE_DEL_ALUMNO`

15.2. Ejercicio

Dirígete a la carpeta ALUMNO.

15.3. Ejercicio

En la carpeta ALUMNO, crea un archivo *prueba.txt* con el editor *nano* que contenga la frase *"Primera Práctica de Comandos"*

15.4. Ejercicio

Muestra el contenido del archivo sin abrirlo, con el comando *cat*

15.5. Ejercicio

Vete al directorio CUENTA_LOGIN

15.6. Ejercicio

Desde el directorio CUENTA_LOGIN, copia el archivo *prueba.txt* de la carpeta ALUMNO en la carpeta PRACTICA_GRUPOX usando rutas absolutas.

15.7. Ejercicio

Desde el directorio CUENTA_LOGIN, mueve el archivo *prueba.txt* de la carpeta ALUMNO a la carpeta NOMBRE_DEL_ALUMNO usando rutas relativas.

15.8. Ejercicio

Comprueba que se han copiado correctamente los archivos en las carpetas utilizando el comando *ls*

15.9. Ejercicio

Empaqueta toda la estructura (`PRACTICA_GRUPOX/ALUMNO/NOMBRE_DEL_ALUMNO`) en un archivo `tar`.

15.10. Ejercicio

Utilizando el comando `ps` muestra el PID correspondiente a tu propio terminal.

15.11. Ejercicio

¿Cómo podríamos saber cual es el proceso inicial del que parten el resto de los procesos lanzados en tu sistema Ubuntu? ¿Cual es su PID?

15.12. Ejercicio

¿Cómo podrías hacer para volcar el comando `free` sobre un fichero `.txt`? Llámalo `free_output.txt`

15.13. Ejercicio

¿Qué comando lanzarías para saber el número de líneas que contiene actualmente tu fichero `free_output.txt`?

15.14. Ejercicio

¿Qué permisos tiene el archivo `free_output.txt` creado?

A continuación, modifica sus permisos. Si es de sólo lectura, dale permisos de escritura. Si se puede escribir, cámbialo para que sea de sólo lectura.

15.15. Ejercicio

Abre un terminal y abre un editor con el comando `gedit`. Abre otro terminal y mata el proceso `gedit` lanzado en el terminal anterior con el comando `kill`.

15.16. Ejercicio

Indique el comando necesario para crear mediante un único comando la carpeta */home/practicas/grupo3/examen/carpeta* usando la línea de comandos.

15.17. Ejercicio

Indique el comando necesario para guardar la salida del comando *pstree* en el archivo *arbol.txt* en el directorio *grupo3* creado anteriormente usando la línea de comandos y suponiendo que el directorio actual es

/home/practicas/grupo3/examen/carpeta

15.18. Ejercicio

Indique el comando necesario para copiar el archivo *arbol.txt* en el directorio examen creado anteriormente usando la línea de comandos y rutas relativas, suponiendo que el directorio actual es

/home/practicas/grupo3/examen

15.19. Ejercicio

Indique el comando necesario para asignar permisos de lectura a aquellos usuarios que no pertenecen al grupo del usuario que lo creó ni son el propio usuario al archivo *arbol.txt* creado anteriormente usando la línea de comandos.

15.20. Ejercicio

Indique el comando necesario (y emplee sólo uno) utilizando pipes para contar el número de líneas que contienen la palabra *network* de la salida del comando *lshw*.

15.21. Ejercicio

Indique el comando necesario para empaquetar la carpeta *examen* en un archivo *examen.tar* suponiendo que el directorio actual es */home/practicas/grupo3/*

15.22. Ejercicio

Indique el comando necesario (y utilice sólo uno) para borrar la carpeta *grupo3* creada anteriormente usando la línea de comandos suponiendo que el directorio actual es */home/practicas*.

15.23. Ejercicio

Indique la secuencia de comandos necesaria para matar el proceso correspondiente a la terminal en la que se está trabajando usando la línea de comandos. Explique cómo se decide qué argumentos se emplean (especificando el fragmento de la salida del comando que da esa información).

15.24. Ejercicio

Indique el comando necesario para crear mediante un único comando la carpeta */home/practicas/laboratorio/examen/carpeta* usando la línea de comandos.

15.25. Ejercicio

Indique el comando necesario para guardar la salida del comando `lsusb` en el archivo `usb.txt` en el directorio examen creado anteriormente usando la línea de comandos y suponiendo que el directorio actual es
/home/practicas/laboratorio/examen

15.26. Ejercicio

Indique el comando necesario para mover el archivo `usb.txt` en el directorio laboratorio creado anteriormente usando la línea de comandos y rutas relativas, suponiendo que el directorio actual es */home/practicas/laboratorio/examen/carpeta*

15.27. Ejercicio

Indique el comando necesario para asignar al archivo `usb.txt` permisos de escritura a todos los usuarios que pertenezcan al mismo grupo que el propietario del archivo usando la línea de comandos.

15.28. Ejercicio

Indique el comando necesario (y emplee sólo uno) utilizando pipes para contar el número de palabras de las líneas del archivo `usb.txt` que contienen la palabra bus usando la línea de comandos y utilizando pipes.

15.29. Ejercicio

Indique el comando necesario para empaquetar la carpeta laboratorio en un archivo `laboratorio.tar` suponiendo que el directorio actual es */home/practicas/*

15.30. Ejercicio

Indique el comando necesario (y utilice sólo uno) para borrar la carpeta laboratorio creada anteriormente usando la línea de comandos suponiendo que el directorio actual es */home/practicas*.

15.31. Ejercicio

Indique la secuencia de comandos necesaria para matar el proceso correspondiente al padre de la terminal en la que se está trabajando usando la línea de comandos. Explique cómo se decide qué argumentos se emplean, cómo se llama el proceso que hay que buscar, etc (especifique además el fragmento de la salida del comando que da esa información).

Referencias:

http://www.linuxtotal.com.mx/?cont=info_admon_002
http://linuxcommand.org/lc3_writing_shell_scripts.php