SESIÓN N° 01

COMANDOS BÁSICOS DE LINUX

I
OBJETIVOS

- Trabajar con el Shell
- Consultar el **manual** en Línea de Unix para descubrir cómo se utilizan algunos de los comandos.
- Utilizar el sistema de archivos, direccionamiento y metacaractares al aplicar los comandos básicos de Unix.

II
TEMAS A TRATAR

- ¿Por qué un sistema basado en UNIX?
- Shell
- Man
- Directorios y archivos
- Direccionamiento
- Metacaracteres
- Comandos básicos

III
MARCO TEORICO

¿Por qué un sistema basado en UNIX?

"UNIX es básicamente un sistema operativo simple, pero se debe ser genio para entender la simplicidad" Dennis Ritchie co-creador de UNIX y fundados del lenguaje C.

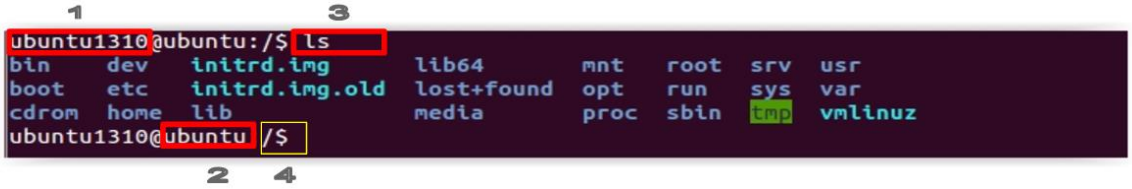
La razón radica primero en la importancia de UNIX/Linux como base de muchos otros sistemas operativos tales como Android, IOS, OS/X y como base de Apache, el software de servidor que se utiliza en la gran mayoría de los servidores de páginas web; y segundo porque se trata de un proyecto de código abierto y por lo mismo es completamente accesible para el alumno para genere ingeniería inversa en una o más de sus partes y realizar modificaciones si lo considera útil para su aprendizaje.

Shell

Son los programas que proveen una interfaz de usuario (textual o grafica) para acceder a los servicios del sistema operativos. Es un intérprete usuario-máquina.

Se llama **Terminar Virtual** a una ventana que emula por completo el funcionamiento de la terminal de computadora, es decir, la interfaz no gráfica de los SO.

1. Representa el usuario registrado en esa terminal
2. Representa a la máquina que está ejecutando el programa “Shell”
3. Comando introducido por el usuario
4. Representa el directorio en el que nos encontramos (/ -Raiz)
5. “$” indica el bash en el nos encontramos



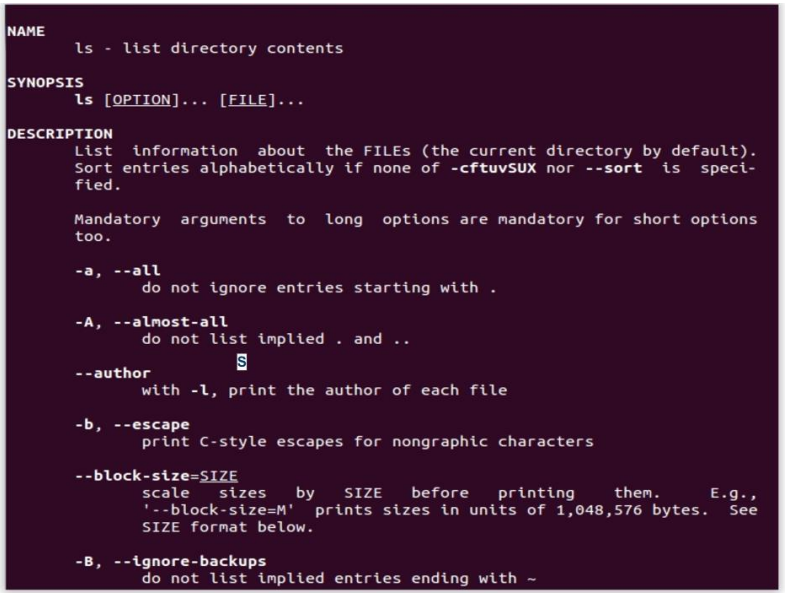
La siguiente ilustración muestra el primer Shell del sistema UNIX denominado Bourne(Sh)

Man

El comando **man**, proporciona información sobre el comando solicitado o permite a los usuarios buscar comandos relacionados con una determinada palabra clave.

Sintaxis:

```
man nombre_comando [opciones]
```



Directorios y Archivos

Estructura de directorios del GNU/LINUX

El sistema de archivos en LINUX se organiza en forma de árbol:

“/” representa el directorio raíz del sistema, y cada usuario tiene la autoridad de administrar y diseñar la organización de sus archivos y directorios a partir del directorio “/home”.

Respecto a los demás directorios estos quedan bajo el resguardo del SO y los maneja para la instalación de aplicaciones, montaje de dispositivos, repositorios entre otras cosas.

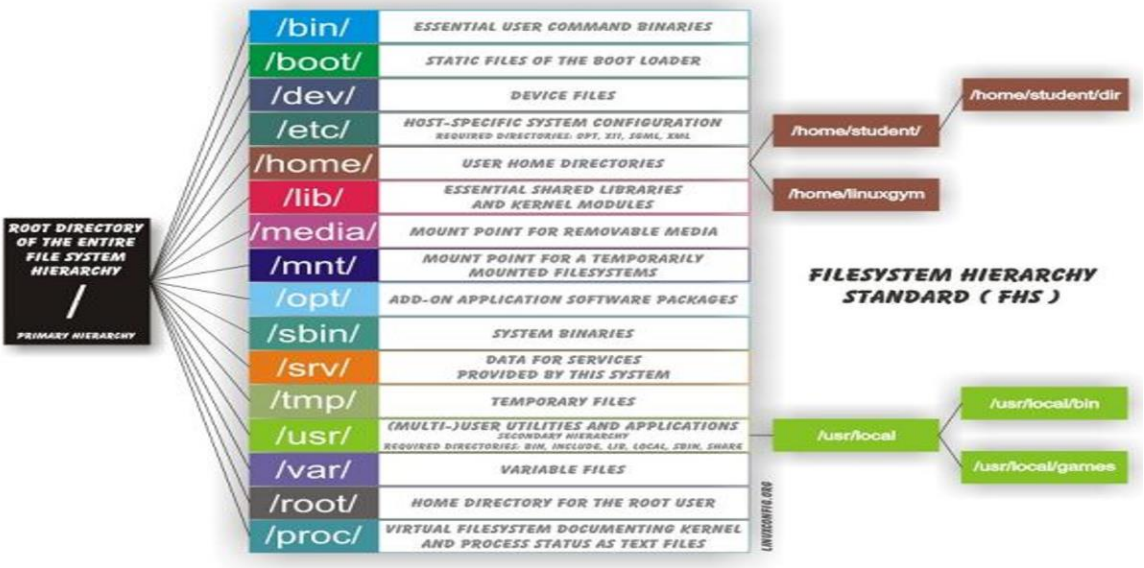
Los sistemas basados en GNU/LINUX consideran que "**todo es un archivo**", esto significa que un dispositivo cualquiera, el teclado, el mouse, monitor, tarjeta de video, drivers, y los mismos archivos tiene la misma representación dentro del sistema de archivos.

Por ejemplo:

/media/flash – podría referirse a un dispositivo de memoria flash montado en el equipo.

/home/flash – se refiere a una carpeta o archivo llamado *flash* contenido en sus archivos personales y la única diferencia es su ubicación

En la siguiente figura se muestra la estructura y el uso de los directorios importantes:



Direccionamiento

Todos los directorios poseen 2 directorios lógicos claves que son: "." (punto) y ".." (punto, punto). El primero representa al directorio actual y el segundo al directorio padre. Ambos casos pueden ser usados como elementos o argumentos al momento de usar comandos.

```
ubuntu1310@ubuntu:~$ ls
0                           Documents        Pictures    Public
asterisk-gui-2.1.0-rc1.tar.gz Downloads      prueba~     Templates
core                        examples.desktop prueba.sh   Videos
Desktop                     Music            prueba.sh~
ubuntu1310@ubuntu:~$ ls .
0                           Documents        Pictures    Public
asterisk-gui-2.1.0-rc1.tar.gz Downloads      prueba~     Templates
core                        examples.desktop prueba.sh   Videos
Desktop                     Music            prueba.sh~
ubuntu1310@ubuntu:~$ pwd
/home/ubuntu1310
ubuntu1310@ubuntu:~$ ls ../..
bin   dev  initrd.img     lib64       mnt  root  srv  usr
boot  etc  initrd.img.old lost+found  opt  run   sys  var
cdrom home lib            media       proc sbin  tmp  vmlinuz
ubuntu1310@ubuntu:~$
```

También se aprovecha para definir el "Path" o camino que es la forma en que se hace referencia a un archivo. Los "Path" pueden ser **absolutos** o

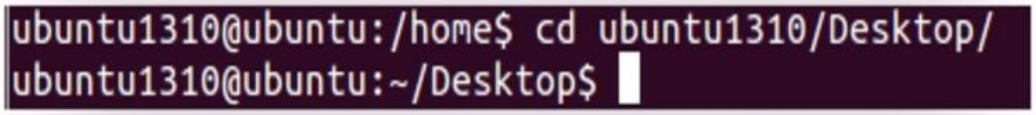
relativos.

- **Direccionamiento Absoluto**

  En el caso del sistema de archivos de Linux al ser **absoluto** se identifica cada ramificación con el símbolo **"/"**, en donde el primer "/" representa el directorio ***root*** (raíz).

- **Direccionamiento Relativo**

  En el caso de ser relativo, notara que no lleva el símbolo /" al inicio del Path y se lo accesa desde la carpeta Padre o antecesor.



Metacaracteres

El shell ofrece la posibilidad de utilizar una notación abreviada para operar sobre conjuntos de ficheros y directorios en una única orden. Esto se consigue mediante el uso de algunos caracteres que tienen significados especiales. Estos caracteres se denominan metacaracteres y se emplean para establecer una correspondencia con nombres o partes de nombres de ficheros. Los más empleados son:

- *: representa cualquier cadena de caracteres, incluyendo la cadena vacía.
- ?: representa a cualquier carácter simple.
- []: una lista de caracteres encerrada entre corchetes especifica que la correspondencia es con cualquier carácter simple de la lista.
- -: el guión se utiliza dentro de los corchetes para indicar un rango de caracteres.

Ejemplos: Supongamos que en un directorio existen los ficheros siguientes:

```
a1   a11  a111 a2   aA   aB   aa   b2
a10  a110 a12  a3   aA1  aG1  b1   b3
```

Las referencias siguientes seleccionan los ficheros que se muestran.

Referencia	Grupo de ficheros referenciados
a*	a1 a10 a11 a110 a111 a12 a2 a3 a4 aA1 aB aG1 aa
a?	a1 a2 a3 aA aB aa
a??	a10 a11 a12 aA1 aG1
a?*	a1 a10 a11 a110 a111 a12 a2 a3 aA aA1 aB aG1 aa
a?1	a11 aA1 aG1
[ab]*	a1 a11 a111 a2 aA aB aa b2 a10 a110 a12 a3 aA1 aG1 b1 b3
?1*	a1 a10 a11 a110 a111 a12 b1
a[A-Z]*	aA aA1 aB aG1
[!a]*	b1 b2 b3
a[A-D]*	aA aA1 aB
? [1-9]*	a1 a10 a11 a110 a111 a12 a2 a3 b1 b2 b3
? [!1-2]*	a3 aA aA1 aB aG1 aa b3

Comandos básicos

- **Órdenes para el manejo de directorios**

pwd
Muestra por pantalla el nombre de camino completo del directorio actual

cd [directorio]
Cambia el directorio de trabajo. Si no especifica ningún parámetro, establece como directorio de trabajo el directorio de conexión (directorio *home*) del usuario.

ls [-aAcCdfFgilLqrRstu1] [fichero(s)]
Muestra el contenido de un directorio. Algunas de las opciones más comunes son:

- F: Si el fichero es ejecutable o un directorio muestra un asterisco(*) o una barra (/) detrás del nombre, respectivamente.
- R: Listado recursivo. Lista ficheros y subdirectorios.
- a: Lista todas las entradas. Normalmente, los ficheros que empiezan por punto (.) no se muestran.
- l: Listado en formato largo. Muestra el modo, número de enlaces, propietario, tamaño en bytes y tiempo de última modificación de cada fichero.

mkdir [-p] directorio
Crea un directorio. La opción -p permite que los directorios padres que falten sean creados. Por ejemplo,

```
mkdir -p source/code/practicas
```

creará los subdirectorios source y code, si no existen, y tras ellos creará el subdirectorio practicas.

rmdir directorio
Borra un directorio, siempre y cuando esté vacío.

- **Órdenes para el manejo de ficheros**

cat [-benstuv] [fichero(s)]
Lee cada fichero especificado como parámetro y muestra sus contenidos por pantalla. Si no se introduce ningún fichero como parámetro, lee de la entrada estándar.

cp [-ip] fichero1 fichero2
cp -rR [-ip] directorio1 directorio2
cp [-iprR] fichero(s) directorio

Copia el contenido de fichero1 en fichero2. El segundo modo permite copiar recursivamente directorio1, junto con sus ficheros y subdirectorios, a directorio2. Si éste último no existe, se crea. Si existe, se realiza una copia de directorio1 dentro de directorio2 (será un subdirectorio).

Con el tercer modo, cada fichero se copia en el directorio indicado. Las opciones -r y -R indican comportamiento recursivo.

rm [-fir] fichero(s)
Borra ficheros y directorios. La opción -r indica comportamiento recursivo y se emplea para borrar directorios.

mv [-fi] fichero1 fichero2
mv [-fi] directorio1 directorio2
mv [-fi] fichero(s) directorio
Mueve ficheros y directorios dentro del sistema de ficheros. Equivale a renombrar un fichero o directorio.

ln [-fs] fichero [enlace]
ln [-fs] camino directorio
Crea un nombre adicional, llamado enlace, a un fichero. Un fichero puede tener varios enlaces.

find lista_de_directorios expresion_de_busqueda
Busca ficheros recursivamente a partir de los directorios señalados en lista_de_caminos, buscando aquellos ficheros que safisfacen una expresión_de_búsqueda. No se siguen los enlaces simbólicos hacia otros ficheros o directorios.

La expresión de búsqueda consta de una o más expresiones primarias, cada una de las cuales describe una propiedad de un fichero, aunque algunas indican una acción a tomar. Las expresiones primarias se pueden combinar mediante los operadores lógicos ! (NOT), -a (AND, que se asume por defecto) y –o (OR).

Algunas expresiones primarias son:

Expresión	Acción
· **-name** fichero	Verdadero si fichero coincide con el nombre del fichero actual. Pueden usarse metacaracteres, pero en este caso fichero debe de ir entre comillas.

· **-user** nombre	Verdadero si el fichero pertenece al usuario nombre
· **-size** n	Verdadero si el fichero tiene una longitud de mayor de n bloques (512 bytes por bloque) si n es positivo, y menor que n bloques si n es negativo.
· **-mtime** n	Verdadero si el fichero ha sido modificado en n días
· **-atime** n	Verdadero si el fichero ha sido accedido en n días
· **-print**	Siempre verdadero. Imprime el camino completo del fichero
· **-exec** orden	Ejecuta la orden sobre el fichero actual. Al final de orden es necesario introducir los caracteres de escape y punto y coma (\;). Si la orden es seguida de {}, el nombre del fichero actual es pasado como argumento a orden.
· **-newer** fichero	Verdadero si el fichero actual ha sido modificado más recientemente que fichero.

Ejemplos:

· Buscar todos los ficheros cuyo nombre termine en .c a partir del directorio actual:

```
find . -name "*.c" -print
```

· Buscar todos los ficheros del sistema cuyo tamaño sea mayor de 100 bloques:

```
find / -size +100 -print
```

· Borrar todos los ficheros cuyo nombre es core a partir del directorio home:

```
find ~ -name core -exec rm {} \;
```

· Imprimir, a partir del directorio actual, todos los ficheros cuyo nombre termine en .c% o .bak y no hayan sido modificados en 20 días:

```
find . \(-name "*.c%" -o -name "*bak" \) -mtime +20 -print
```

file [-f ffich] [-cL] [-m mfile] fichero ...
Determina el tipo de un fichero examinando su contenido.

du [-s] [-a] fichero ...
Muestra el número de bloques de disco usados por un conjunto de ficheros y directorios

df [-a] [-i] [-t type] [filesystem ...] [fichero ...]
Muestra el espacio libre en el sistema de ficheros

- **Órdenes para el control de procesos**

ps [-acCegklnrStuvwxU]
Muestra el estado de los procesos del sistema. Sin argumentos, muestra información sobre los procesos asociados a la sesión del usuario.

kill [-señal] pid ...
Envía una señal a un proceso. Por defecto, se envía la señal de terminación del proceso (SIGTERM). Esta señal puede ser ignorada por el proceso, por lo que para eliminar de forma segura un proceso es necesario enviarle la señal de terminación incondicional (señal 9).

nice [-numero] orden [argumentos]
Permite modificar la prioridad con la que se ejecutará un proceso. La prioridad del proceso se aumentará en la cantidad señalada por numero. A mayor valor de numero, menor prioridad. Por defecto, numero toma el valor de 10. Es posible aumentar la

prioridad de un proceso si `numero` es un valor negativo, pero en esta posibilidad únicamente puede ser usada por el superusuario.

- **Órdenes para seguridad y protección**

chmod `[-fR] modo fichero ...`
Cambia los permisos (modo) de uno o varios ficheros o directorios. Únicamente puede ser usado por el propietario del fichero (o el superusuario). El modo del fichero se puede especificar de forma absoluta o de forma simbólica. El modo absoluto es un número octal que indica los permisos del fichero

Por ejemplo:
`chmod 444 datos`.

El modo simbólico es una cadena que tiene la forma
`[quien] operador permiso [operador permiso]`

donde quien es una combinación de las letras `u` (usuario), `g` (grupo) y `o` (otros) ó `a` (que equivale a `ugo`), operador es `+` (añade un permiso), `-` (elimina un permiso) ó `=` (establece un permiso); y permiso es una combinación de `r`, `w` ó `x`.

Ejemplos:
· Retirar el permiso de escritura, para el propietario, del fichero `datos`:
`chmod u-w datos`

· Establecer el permiso de lectura al fichero `datos`:
`chmod ugo=r datos`

· Retirar el permiso de lectura de `datos` a todos menos al propietario:
`chmod go-r datos`

· Asignar permisos de lectura a todos los usuarios y de escritura para el propietario:
`chmod a=r, u+w dataos`

chown `propietario fichero ...`
Cambia el propietario de un fichero. Sólo puede ser empleada por el propietario del fichero o por el superusuario. En la versión de UNIX SunOS, únicamente puede ser empleada por el superusuario.

passwd
Permite cambiar la palabra de paso del usuario.

- **Ordenes varias**

diff
Muestra las diferencias, línea por línea, entre dos ficheros

grep
Busca las líneas de un fichero que contienen una determinada cadena o expresión regular

head
Muestra las primeras líneas de un fichero

tail

Muestra las últimas líneas de un fichero

wc
Cuenta las líneas, palabras y caracteres de un fichero

who
Muestra los usuarios que se encuentran conectados al sistema

whoami
Muestra el nombre del usuario de la sesión activa.

IV
ACTIVIDADES

Manipulando el shell

01. El "shell" es el comando que interpreta y ejecuta todos los otros comandos. Cuando arrancas linux, el shell se comienza a ejecutar automáticamente y se queda esperando a que teclees otros comandos. Hay varios "shells" que se han desarrollado a lo largo de los años. Verificar el shell, asegurarse que se está utilizando el shell "bash", ejecutar:

```
$ echo $BASH
/bin/bash
```

02. Si no aparece la segunda línea significa que nos hallamos en otro shell, para escoger BASH bastará con ejecutar el comando "bash" como se muestra en el siguiente ejemplo:

```
$ echo $BASH
/usr/bin/sh
$ bash
$ echo $BASH
/bin/bash
```

03. Con el shell bash ya inicializado escriba "wh" y a continuación presione la tecla TAB en dos ocasiones. Notará que esto provoca que se desplieguen todos los comandos que inician con "wh" como se muestra a continuación:

```
$ wh
whatis    whereis       which    while    whiptail  who  whoami
```

A excepción de `whiptail, while y whois`, la mayoría son comandos útiles para ver status de usuario, descripciones de programas y ubicación de los mismos.

04. En base a los comandos mostrados hasta este punto y la información que proporcionen sus respectivos MAN, conteste las siguientes preguntas:

a) ¿Cuáles comandos nos muestran el listado de usuarios activos en el sistema?

b) ¿Cuál sería el comando para desplegar la fecha del último "boot" (Reinicio) del sistema? Si el comando requiere determinadas opciones, inclúyelas

c) Si un archivo tuviese exclusivamente 3 líneas de texto, ¿cuál sería la diferencia de utilizar los comandos `head, tail, more y cat`?

d) Si queremos leer el archivo `/etc/passwd` (el cual contiene el listado de usuarios del sistema) ¿Cuál sería el más apropiado entre los comandos `head`, `tail`, `more` y `cat`? ¿Por qué?

e) ¿Cuál es el comando que se recomienda utilizar en lugar de more?

¿Dónde estamos?

05. Supongamos que nuestro usuario de nombre "fulano" tiene la estructura en su directorio "`/home`" -obtenida mediante el comando *tree*- de la siguiente forma:

```
├── Desktop
├── Documents
│   └── Pruebas
├── Downloads
├── Music
├── Pictures
├── Public
├── Sesion1
│   ├── Archivo1
│   └── Archivo2.txt
├── Templates
└── Videos
```

Conteste las siguientes preguntas

a) ¿Qué diferencia existe entre `Archivo1` y `Archivo2.txt`? (pista: En linux las "extensiones" como .txt no indican el tipo de archivo, solo se utilizan como convenciones)

Si la línea en bash aparece como: `fulano@host: /etc$`

b) ¿Cuál es el comando para desplegar todo el contenido de `Archivo2.txt` utilizando direccionamiento relativo al directorio en el que nos encontramos? Si el comando requiere determinadas opciones, inclúyelas

c) ¿Cuál es el comando para desplegar el contenido del folder o directorio `Sesion1`, incluyendo los directorios lógicos (también llamados simbólicos) y en orden alfabético, utilizando direccionamiento absoluto (es decir, comenzando por la raíz de todos, "`/`")?

d) ¿Cuál es el comando para duplicar la información liberada por `tree`?

e) Valide su respuesta anterior con su propio directorio `home`, utilizando tanto `tree` como el comando sugerido por usted.

Manipulado archivos y directorios

Teniendo en cuanta todo lo visto en esta práctica, realice las siguientes actividades.

06. Ejecute los siguientes comandos

```
$ mkdir $HOME/Operativos
$ touch $HOME/Operativos/Arch1
$ touch $HOME/Operativos/Arch2
$ touch $HOME/Operativos/Arch3
```

07. Conteste las siguientes preguntas:
    a) Comando para copiar el contenido del archivo `/etc/passwd` a `Arch1`
    b) Comando que copie el archivo `Arch1` del paso anterior con el nombre `Arch4`.
    c) Desde `$HOME/Operativos` ejecute el comando

```
$ mkdir ./Acto1
```

    d) Mueva el archivo `Arch4` al directorio creado en el paso anterior.
    e) Despliegue la primera línea de `Arch4` con direccionamiento absoluto
    f) Utilice solamente un único comando para borrar todo el contenido del directorio `Operativos`

NOTA: Para crear y editar archivos utilice el editor que le parezca más conveniente. Si no conoce algún editor se recomienda utilizar ***gedit*** en modo gráfico o ***nano*** en consola.