Facultad: Ingeniería
Escuela: Computación
Asignatura: Sistemas Operativos

Tema: Introducción a los comandos Linux

Contenido

En esta guía se tratan los metodos basicos de manejo de archivos, y uso del sistema operativos por medio de la linea de comandos de Linux en especial Bash.

Objetivo Especifico

- Conocer conceptos básicos en general de Linux.
- Utilizar los comandos básicos para el manejo de archivos y directorios.

Material y Equipo

- Sistema operativo Linux con interprete de comandos Bash
- Guía Número 1

PARTE I

Introduccion Teorica

La linea de comandos y Bash

La linea de comandos es una herramienta de interfaz de usuario existente en prácticamente todos los sistemas operativos. Esta se basa en el uso de ordenes digitadas por medio del teclado, el estándar por defecto para los sistemas Unix y en especial Linux es el shell Bash.

Bash es un software para interpretar órdenes. Está basado en la shell de Unix y es compatible con POSIX (API de Unix). Fue escrito para el proyecto GNU y es el intérprete de comandos por defecto en la mayoría de las distribuciones de Linux.

A Bash lo podemos encontrar en varios lugares dentro de GNU/Linux:

- En el momento en que el sistema a cargado completamente en

modo consola.
- Accediendo desde las interfaces gráficas por medio de los programas gnome-terminal, kterm, xterm, etc.
- Por medio de una de las 7 terminales virtuales (TTY) que provee GNU/Linux, la manera de acceder es por medio de las teclas rápidas CTRL + ALT + F***n***. Donde ***n*** es el numero de interfaz a la que queremos acceder (la interfaz 7 casi siempre es la interfaz gráfica)

En las lineas de comando se encuentra algo llamado *indicador* el cual es el lugar donde se digitan las ordenes o comandos, este indicador en GNU/Linux puede se ser configurado en su totalidad, y cada distribución lo muestra según sus necesidades, pero casi siempre se mantiene una parte del indicador el cual describe los privilegios del usuario ($ ó #), si estamos en fedora el indicador sera este:

[usuario@nombre-ordenador carpeta-actual]($/#)

Al final del indicador se encuentran los símbolos "$" y "#", estos muestran los privilegios del usuario actual, si se tiene el símbolo "$", solo podemos modificar archivos en nuestra cuenta de usuario (/home/usuario). En cambio si tenemos el símbolo "#" nuestro usuario sera el administrador *root* y podremos modificar archivos en todo el disco.

Así, sí nuestra cuenta de usuario es *user*, nuestro ordenador se tiene el nombre *miPC*, y nos hemos desplazado a la carpeta de */home/user/Documents*. El indicador se vera de la siguiente manera:
[user@miPC Documents]$

Y si accedemos a la cuenta *root*, se vería de la siguiente manera:
[root@miPC Documents]#

Procedimiento

Uso de comandos de Linux.

man [comando]: Amplia la ayuda de los comandos.

Muestra en pantalla un manual del comando, su modo de uso y sus variantes.

```
[usuario@localhost /home/usuario]$ man cat
[usuario@localhost /home/usuario]$ man ls
[usuario@localhost /home/usuario]$ man df
```

Muestra en pantalla una sinopsis de la sintaxis del comando cat, una descripción de los que hace el comando y sus variantes.

clear: limpia la pantalla.

```
[usuario@localhost /home/usuario]$ clear
```

ls [opcion] [archivo]: Listar archivos en un directorio.

ls nos permite ver el contenido de un directorio y opcionalmente sus subdirectorios. Este comando tiene muchas opciones. La forma mas corriente es simplemente

```
[usuario@localhost /home/usuario]$ ls
```

Que lista en varias columnas los nombres de los archivos en el directorio actual.
Otra variante común es

```
[usuario@localhost /home/usuario]$ ls -l
```

Proporciona un listado largo (permisos, tamaños, dueño, etc.).

```
[usuario@localhost /home/usuario]$ ls -FC
```

Coloca a los ejecutables un asterisco, a los directorios la barra /, y a los archivos comunes nada.

```
[usuario@localhost /home/usuario]$ ls -l *.f
```

Esto da todos los archivos que terminan en .f, con listado largo. Además, podemos ordenar la lista de varias maneras, por ejemplo por edad:

```
[usuario@localhost /home/usuario]$ ls -lt *.f
```

Esto nos lista en edad descendiente (mas viejo de ultimo) los archivos en que terminan en .f, con listado largo.

mkdir [opcion] nombredir: Crear un directorio

Crea un directorio vacío en el directorio actual, por ejemplo

```
[usuario@localhost /home/usuario]$ mkdir directorio01
[usuario@localhost /home/usuario]$ mkdir directorio02
[usuario@localhost /home/usuario]$ mkdir directorio03
```

Crea el directorio directorio01, directorio02 y directorio03 en el directorio actual.

cd nombredir: Cambio de directorio

Con **cd** cambiamos el directorio donde estamos trabajando.

Ejemplos: Cambio absoluto de directorio:

```
[usuario@localhost /home/usuario]$ cd directorio01
```

Cambia al directorio citado, en la línea de comandos, se muestra el directorio al que se ha cambiado.

```
[usuario@localhost /home/usuario/directorio01]$ cd ~
```

Cambia al "directorio base" o "*home directory*" designado por el súper-usuario..

cp [opcion] fuente destino: Copia de archivos

El comando **cp** copia archivos. El último argumento es el destino, los precedentes son el origen.
El comportamiento de **cp** depende del destino. Si el destino es un subdirectorio, los archivos son copiados a ese subdirectorio; pero si el destino no existe o es un archivo el origen es copiado al destino.
En el caso de múltiples orígenes, todos los orígenes son sucesivamente copiados al destino. Eso quiere decir que en efecto solo el penúltimo sobrevive la operación, lo cual es probablemente no deseable.
Ejemplos:

cp -i origen destino

Copia origen a destino, pero en el caso que destino ya existe pregunta si se desea sobrescribir.

Se crearan tres archivos vacíos con el comando touch y después se copiaran en el directorio02.

```
[usuario@localhost /home/usuario/directorio01]$ touch archivo01
[usuario@localhost /home/usuario/directorio01]$ touch archivo02
[usuario@localhost /home/usuario/directorio01]$ touch archivo03
[usuario@localhost /home/usuario/directorio01]$cp archivo01 archivo02
archivo03 ../directorio02
```

mv [opcion] origen destino: Mover archivos entre directorio o renombrar

El comando **mv** es similar a **cp**, excepto que borra el origen. En otras palabras, mueve archivos de un directorio a otro, o de un archivo a otro. En este último caso, como el original desaparece, **mv** puede a veces tener efectos inesperados. El último argumento de **mv** indica el destino del movimiento; los primeros son los orígenes.
Un uso muy frecuente de **mv** es de cambiar el nombre a un archivo.

Para mover archivos archivo01, archivo02 y archivo03 al directorio directorio03:

```
[usuario@localhost /home/usuario/directorio01]$ mv archivo01 archivo02 archivo03
../directorio03

[usuario@localhost /home/usuario/directorio01]$ cd ../directorio03

[usuario@localhost /home/usuario/directorio03]$ mv archivo03 archivonombrenuevo
```

rm [opcion] nombrearch: Borrar archivos

rm borra archivos y con ciertas opciones, hasta directorios. **Advertencia:** este comando es **irreversible.**
Ejemplos:

```
[usuario@localhost /home/usuario/directorio03]$ rm archivo01 archivo02
archivonombrenuevo
```

Este comando borra el archivo archivonombrenuevo, archivo01 y archivo02.

Con la opción recursiva, se puede borrar un directorio y todos los archivos dentro de ese directorio. Esto es equivalente a hacer cd a ese directorio, borrar todos los archivos, subir, y ejecutar a rmdir.

```
[usuario@localhost /home/usuario/directorio03]$ cd ..
[usuario@localhost /home/usuario/ rm -r directorio02
```

se desplegara el siguiente mensaje

rm: descend into directory ‘directorio02’?

digite la letra “y” y luego enter.

Luego, el comando pregunta si se quiere remover los archivos dentro del directorio, escriba “y”.

```
rm: remove ‘directorio02/archivo01’ ? Y
```

```
rm: remove 'directorio02/archivo02' ? y
rm: remove 'directorio02/archivo03'? y
```

rmdir: Borrar un directorio

Este comando borra un subdirectorio vacío. Si no esta vacío, rmdir despliega un mensaje de error y no efectúa la operación.

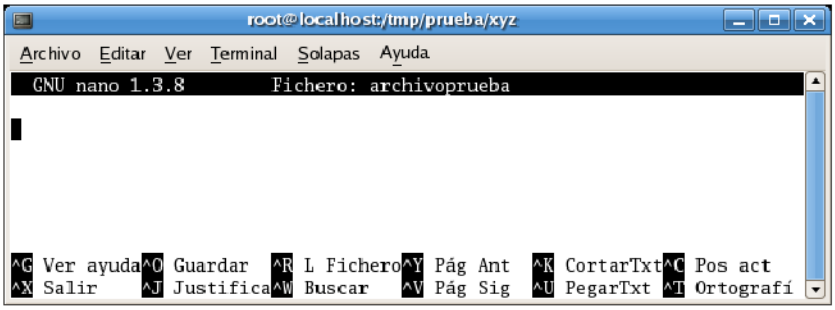
```
[usuario@localhost /home/usuario/]$  rmdir directorio01
```

Borra el directorio directorio01.

nano [opcion] [archivo]: Editor de texto.

Nano es un editor de texto fácil de utilizar.

```
[usuario@localhost /home/usuario/]$ nano archivoprueba
```



Otros editores de texto que posee Linux son (si están instalados por supuesto): vi/vim, emacs.

cat [opcion] [archivo] : Concatenar (o ver) archivos.

El objetivo principal de **cat** es de pegar o encadenar archivos. El archivo resultado va a stdout. Cuando hay un solo archivo este aparece por pantalla. Por eso, **cat** se usa mucho para ver el contenido de un archivo, aunque para eso es mejor el comando **more**.

Ejemplos:

```
[usuario@localhost /home/usuario/]$ cat archivoprueba
```

[Se muestra el contenido del archivoprueba]

find [ruta] [opcion] patron: Encontrar un archivo con ciertas características.

El comando **find** se usa para encontrar archivos en el árbol de directorios de Linux. La estructura de directorios puede ser arbitraria. **find** requiere un punto de partida y las características del archivo a

encontrarse. Después, **find** revisa ese directorio y todos los directorios subordinados, buscando los archivos que cumplan la condición(es) citada(s).

```
[usuario@localhost /home/usuario/]$ find / -name "*c" -print
```

Muestra todos los archivos que terminan con la letra "c" a partir del directorio raiz "/".

Esto busca en el directorio actual (.) todos los archivos o directorios de nombre perdido y pone el resultado en pantalla (-print). En algunas versiones modernas de Linux, la opción -print no es necesaria, pero en otras si.

df: Características de un sistema de archivos

Este comando da las características de un sistema de archivos. Su primer uso es de ver que es lo que esta montado; su segundo uso es de ver el espacio libre de un sistema de archivos.
En mi sistema, el comando

```
[usuario@localhost /home/usuario/]$ df
produce
Filesystem    1k-blocks     Used          Available     Use%   Mounted on
/dev/hda6     19424         18165         1259          94%    /
/dev/hda7     51718         42116         9602          81%    /usr
```

De lo cual se puede ver, por ejemplo, que el sistema raíz (/) esta a punto de llenarse. La apariencia de la salida de este comando varía entre diferentes versiones de Linux.

PARTE II

Introduccion Teorica

Linux es un sistema operativo multiusuario y multitarea, lo que le permite a varios usuarios estar utilizando el sistema simultáneamente y ejecutando varios procesos a la vez.

Multiusuario.

Debido a su característica de ser multiusuario, existe una administración de usuarios y grupos; para ingresar a un sistema Linux, es necesario conocer un login ID y un password (que debe ser de al menos 6 caracteres).

La existencia de usuarios exige implementar una fuerte seguridad a nivel de archivos, de tal forma que un usuario puede o no conceder derecho a otros usuarios de leer, ejecutar o modificar sus archivos y carpetas.

Para administrar dicha seguridad, se utiliza el sistema de archivos extendido 2 o ext2fs (por sistema de archivo se entiende a la forma que el sistema operativo organiza y administra los archivos), el sistema de archivos ext2fs permite hasta 256 caracteres en los nombres de archivos y 4 Terabytes máximo en espacio por archivo.

Un archivo es una colección de información almacenada en algún medio secundario. Existen 2 tipos de archivos de sistema:

- **Archivos ordinarios**: archivos que contienen datos, código ejecutable, etc.
- **Archivos de directorio**: archivos que contienen nombres de archivos, este tipo de archivos se utiliza para organizar los archivos en grupos llamados comúnmente directorios.

La protección de archivos en Linux esta relacionada con los usuarios del sistema. Existen 3 clases diferentes de usuarios de un archivo y 3 modos diferentes de acceder al archivo:

Clase de usuario	Descripción
Propietario	Usuario que creo el archivo. Tiene derecho de controlar quien accede el archivo.
Grupo	Grupo de usuarios. Se utiliza para organizar a los usuarios del sistema según la ubicación geográfica

	o funciones; de esta forma, es mucho más fácil para el propietario del archivo asignar permisos grupales en vez de asignarlos individualmente.
Otros	Cualquier otro usuario del sistema.

Para cada una de las 3 clases de usuarios existen 3 modos de acceso diferente:

Modo	Archivo ordinario	Archivo de directorio
Lectura (r)	Permite examinar el contenido del archivo.	Permite listar los archivos contenidos en el directorio.
Escritura (w)	Permite cambiar el contenido del archivo.	Permite crear y eliminar directorio.
Ejecución (x)	Permite ejecutar el archivo como un comando.	Permite buscar en el directorio.

Habitualmente, cada usuario normal tiene un directorio HOME, donde tiene control total sobre los archivos propios, luego se le restringe el acceso a solo lectura o denegado para archivos del sistema o archivos de otros usuarios. Esto asegura que un usuario no pueda eliminar o modificar archivos ajenos o archivos esenciales del sistema operativo.

Multiproceso.

En Linux, se pueden tener varios procesos o tareas ejecutándose simultáneamente. Un comando puede ejecutarse en foreground o en background.

Control de procesos.

Un comando puede ejecutarse en foreground o en background. Cuando un comando se ejecuta en foreground, el prompt de sistema no retorna hasta que el comando haya finalizado su ejecución, sin embargo, si se ejecuta en background, entonces la tarea se ejecuta en segundo plano. Para correr un trabajo en background es necesario poner un ampersand (&) al final de la línea.

Por ejemplo, imagine que se desea hacer ping a una maquina por un tiempo indefinido:

```
[usuario@localhost /home/usuario/]$ ping localhost
```

Al ejecutarse el programa anterior, observara al prompt del sistema ocupado, lo que significa que se esta ejecutándose en primer plano. Observe que el comando esta escribiendo constantemente información en la pantalla que el ping suministra. Presione CTRL+C para

cancelar la tarea.

```
[usuario@localhost /home/usuario/]$ ping localhost >/dev/null &
```

Observara que el prompt regresa al sistema operativo, sin embargo si ud. ejecuta el comando **jobs**, Linux le muestra las tareas que se están ejecutando.

Para mover una tarea que se esta ejecutando en background a foreground, solo es necesario digitar **jobs** y luego **fg no_tarea**, como se muestra a continuación:

```
[usuario@localhost /home/usuario/]$  jobs
[1]+ Running    ping localhost > /dev/null &
[usuario@localhost /home/usuario/]$  fg 1
ping localhost > /dev/null
[CTRL+C]
[usuario@localhost /home/usuario/]$
```

Para visualizar todos los procesos que se están ejecutando en Linux, existen los comandos **ps [-all]** o listar el contenido del directorio **/proc** con **ls**.

Procedimiento

who [opciones]: informa de quien esta actualmente en el sistema…

y proporciona información de otro usuario y de entrada del sistema.

- Comando **whoami** muestra quienes somos en la terminal en la que estamos trabajando.
- Comando Who muestra quienes están conectados al sistema en ese momento.

du [opciones] [archivo]

El comando **du** se utiliza para conocer el espacio de disco ocupado por un archivo. En el caso de que el archivo sea de tipo directorio, calcula el espacio utilizado por todos sus subdirectorios de forma recursiva. Su sintaxis es:

Un ejemplo de su utilización (con la opción -s y - h se muestra un resumen de la salida en unidades de facil interpretación humana, en mega M, kilo K, Giga G, entre otros) sería:

```
[usuario@localhost /home/usuario/]$ du  -sh  /etc
101M   /etc
```

pwd

Este comando imprime el directorio actual en pantalla, es la abreviación de **P**rint **W**orking **D**irectory.

La sintaxis es

```
[usuario@localhost /home/usuario/]$ pwd
/home/usuario
```

ls -a: listar archivos invisibles.

Archivos invisibles. Los archivos cuyos nombres comienzan con un punto (.) se llaman archivos invisibles, por que normalmente no aparecen con la instrucción **ls** a menos que se especifique el parámetro "-a". por lo general, estos archivos invisibles se utilizan para almacenar información del sistema.
Ejemplo del uso de "-a":

```
```

```
[usuario@localhost /home/usuario/]$ ls -a
.     .bash_logout      .gnome_private
..    .bash_profile     .gnp
bin
```

more

Se utiliza para visualizar archivos en pantalla. La ventaja sobre el comando **cat** es que hace una pausa al llenar la pantalla; con la barra espaciadora se avanza un pagina, con **<Enter>** se avanza una línea y con **q** o **del** se finaliza el comando.

```
[usuario@localhost /home/usuario/]$ more archivomuestra
[ Se muestra contenido archivo]
```

less

muy similar a **more**, sin embargo nos permite visualizar libremente un documento.

```
[usuario@localhost /home/usuario/]$ less archivomuestra
```

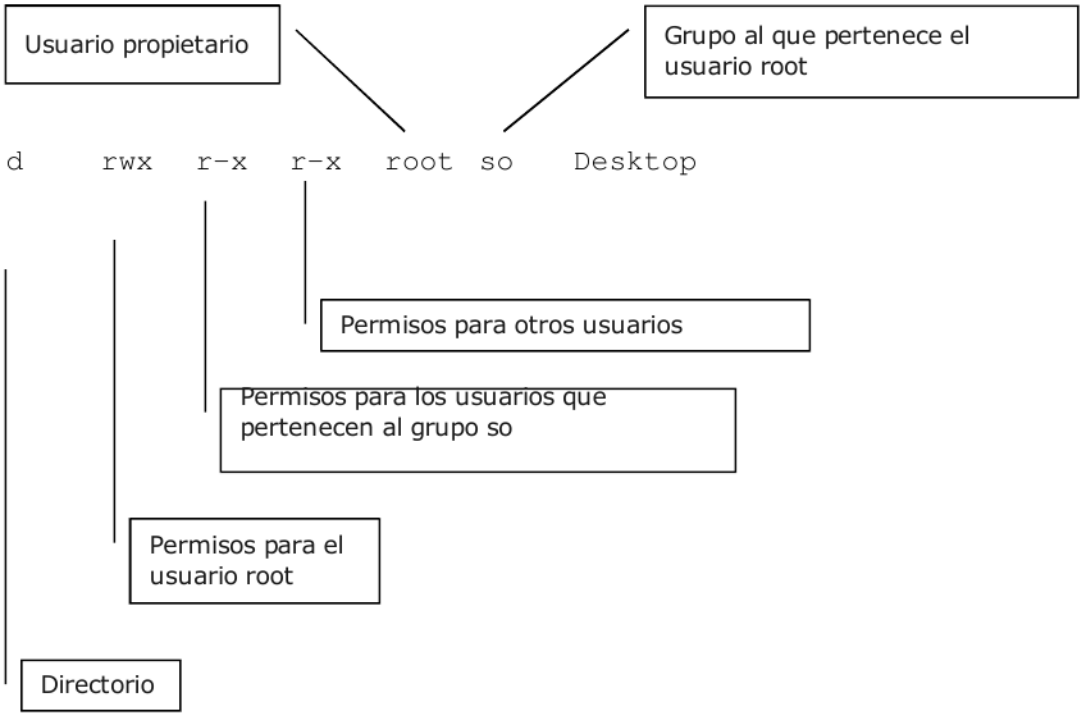
Permisos
Para mostrar en pantalla los permisos de los archivos o directorios, se utiliza el comando **ls** seguido de la opción - l

Ejemplo.

```
[usuario@localhost /home/usuario/]$ ls -l
total 40
drwxr - xr - x         5 usuario   so    4096  Aug 25 13:17      Desktop
drwxr - xr - x         2 usuario   so    4096  Jan 21      15:04 bin
-rw - r - - r - -      1 usuario   so    17    Jan 15      16:42 archivo01
-rw - r - - r - -      1 usuario   so    23    Jan 15      16:42 archivo02
-rw - r - - r - -      1 usuario   so    40    Jan 15      16:42 archivo03
-rwxr - xr - x         1 usuario   so    13410 Jan 21      10:50 archivo04
-rw - r - - r - -      1 usuario   so    82    Jan 15      16:42 archivo05

 . . . . . .
```

Explicación



Cambio de permisos.

chmod: cambia los permisos.

El comando **chmod** (change mode) se utiliza para cambiar los permisos de un archivos ordinario y un directorio.

Existen dos formas de cambiar los permisos, el modo simbólico y el modo absoluto.

Modo simbólico: cuando se utiliza este modo, se puede añadir o quitar permisos a los archivos y directorios. El formato del comando es:

Chmod [who] codigo_operador permisos archivo

Who: tipo de usuario, que puede ser:
- U: propietario del archivo
- G: grupo al que pertenece el propietario
- O: usuarios clasificados como otros
- A: todos los usuarios del sistema (propietario, grupo y otros)

Codigo_operador: el símbolo + se utiliza para añadir permisos, el símbolo - los remueve.
Permisos: tipo de permiso el cual puede ser de solo lectura (r), escritura (w) o ejecución (x).

Modo absoluto: el modo absoluto se especifica con 3 dígitos numéricos.

Cada número representa el permiso de cada tipo de usuario.

Estos dígitos se obtienen, para cada clase de usuario a partir de los siguientes valores:

1: ejecución
2: escritura
4: lectura

Sintaxis del comando:

chmod modo archivo

De donde:

0: ningún permiso
1: permiso de ejecución
2: permiso de escritura
3: permiso de ejecución + escritura (1+2)
4: permiso de lectura
5: permiso de lectura + ejecución (4+1)
6: permiso de lectura + escritura (4+2)
7: permiso de lectura, escritura y ejecución (4+2+1).

Ejemplo:

```
[usuario@localhost /home/usuario/]$ chmod 777 archivomuestra
[usuario@localhost /home/usuario/]$ ls
```

Hoja de cotejo: 2

Guía 2: Introducción a los comandos Linux

Alumno: Maquina No:

Docente: GL: Fecha:

EVALUACION					
	%	**1-4**	**5-7**	**8-10**	**Nota**
CONOCIMIENTO	Del 20 al 30%	Conocimiento deficiente de los fundamentos teóricos	Conocimiento y explicación incompleta de los fundamentos teóricos	Conocimiento completo y explicación clara de los fundamentos teóricos	
APLICACIÓN DEL CONOCIMIENTO	Del 40% al 60%				
ACTITUD					
	Del 15% al 30%	No tiene actitud proactiva.	Actitud propositiva y con propuestas no aplicables al contenido de la guía.	Tiene actitud proactiva y sus propuestas son concretas.	
TOTAL	100%				