



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: M.I. Marco Antonio Martínez Quintana

Asignatura: Fundamentos de Programación

Grupo: 3

No de Práctica(s): 2. GNU/Linux

Integrante(s): Hernández Vázquez Daniela

*No. de Equipo de
cómputo empleado:* No aplica

No. de Lista o Brigada: 26

Semestre: 1

Fecha de entrega: Lunes 12 de octubre de 2020

Observaciones:

CALIFICACIÓN: _____

Práctica 2: GNU / Linux

Objetivo:

Conocer la importancia del sistema operativo de una computadora, así como sus funciones. Explorar un sistema operativo GNU/Linux con el fin de conocer y utilizar los comandos básicos en GNU/Linux.

Actividades:

- Iniciar sesión en un sistema operativo GNU/Linux y abrir una "terminal"
- Utilizar los comandos básicos para navegar por el sistema de archivos.
- Emplear comandos para manejo de archivos.

Introducción

El Sistema Operativo es el conjunto de programas y datos que administra los recursos tanto de hardware (dispositivos) como de software (programas y datos) de un sistema de cómputo y/o comunicación. Además, funciona como interfaz entre la computadora y el usuario o aplicaciones.

En la actualidad existen diversos sistemas operativos; por ejemplo, para equipos de cómputo están Windows, Linux, Mac OS entre otros. Para el caso de dispositivos móviles se encuentran Android, IOS, Windows Phone entre otros. Cada uno de ellos tiene diferentes versiones y distribuciones que se ajustan a los diversos equipos de cómputo y comunicación en los que trabajan.

Los componentes de un sistema operativo, de forma general, son:

- Gestor de memoria,
- Administrador y planificador de procesos,
- Sistema de archivos y
- Administración de E/S.

Comúnmente, estos componentes se encuentran en el kernel o núcleo del sistema operativo.

En cuanto a la Interfaz con el usuario, las hay de tipo texto y de tipo gráfico. En la actualidad, es común trabajar con la interfaz gráfica ya que facilita mucho seleccionar la aplicación a utilizar; inclusive esta selección se hace "tocando la pantalla" (técnica touch).

Sin embargo, cuando se desarrollan proyectos donde se elaborarán documentos y programas es necesario el uso de dispositivos de entrada y salida (hardware) y aplicaciones en modo texto (software).

Sistema Operativo Linux

Linux es un sistema operativo tipo Unix de libre distribución para computadoras personales, servidores y estaciones de trabajo.

El sistema está conformado por el núcleo (kernel) y un gran número de programas y bibliotecas. Muchos programas y bibliotecas han sido posibles gracias al proyecto GNU, por lo mismo, se conoce a este sistema operativo como GNU/Linux.

Software libre

Un software libre es aquel que se puede adquirir de manera gratuita, es decir, no se tiene que pagar algún tipo de licencia a alguna casa desarrolladora de software por el uso del mismo.

Además, que un software sea libre implica también que el software viene acompañado del código fuente, es decir, se pueden realizar cambios en el funcionamiento del sistema si así se desea.

Linux se distribuye bajo la Licencia Pública General de GNU por lo tanto, el código fuente tiene que estar siempre accesible y cualquier modificación o trabajo derivado debe tener esta licencia.

Licencia GNU

La Licencia Pública General de GNU o GNU General Public License (GNU GPL) es una licencia creada por la Free Software Foundation en 1989 y está orientada principalmente a proteger la libre distribución, modificación y uso de software.

Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

Kernel de GNU/Linux

El kernel o núcleo de linux se puede definir como el corazón del sistema operativo. Es el encargado de que el software y el hardware del equipo se puedan comunicar. Sus componentes son los que se mencionaron en la introducción de esta práctica.



Figura 1: Capas que componen al sistema operativo GNU/Linux.

De la figura 1, se puede observar que entre el kernel y las aplicaciones existe una capa que permite al usuario comunicarse con el sistema operativo y en general con la computadora, a través de programas que ya vienen instalados con la distribución de Linux (Debian, Ubuntu, Fedora, etc.) y trabajan ya sea en modo gráfico o en modo texto. Uno de estos programas es el Shell.

La estructura de Linux para el almacenamiento de archivos es de forma jerárquica; por lo que la carpeta o archivo base es "root" (raíz) la cual se representa con una diagonal (/). Desde este archivo raíz, parten todos los demás. Los archivos pueden ser carpetas (directorios), de datos, aplicaciones, programas, etc.

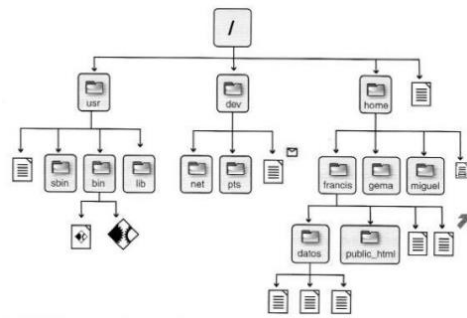


Figura 2: Una parte del sistema de archivos jerárquico en GNU/Linux.

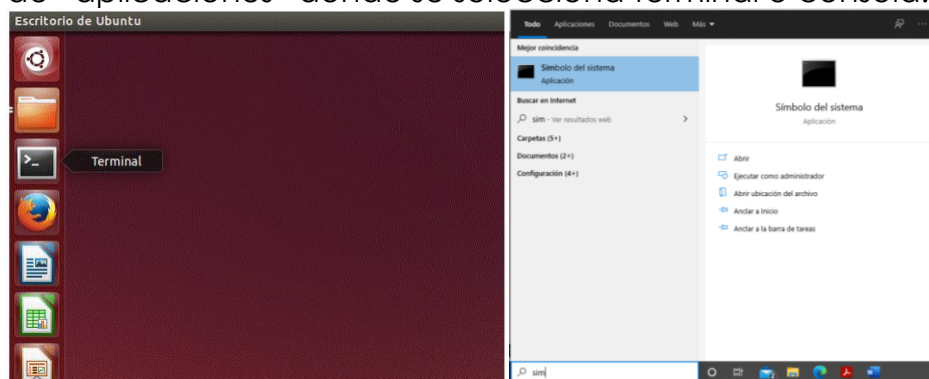
Interfaz de línea de comandos (CLI) o shell de GNU/Linux

El Shell de GNU/Linux permite introducir órdenes (comandos) y ejecutar programas en el sistema operativo. Todas las órdenes de UNIX/Linux son programas que están almacenados en el sistema de archivos y a los que llamamos comandos, por lo tanto, todo en GNU/Linux se puede controlar mediante comandos.

Comandos básicos

Para trabajar en Linux utilizando comandos, se debe abrir una “terminal” o “consola” que es una ventana donde aparece la “línea de comandos” en la cual se escribirá la orden o comando. La terminal permite un mayor grado de funciones y configuración de lo que queremos hacer con una aplicación o acción en general respecto a un entorno gráfico.

El proceso de abrir una terminal varía dependiendo del entorno gráfico. Por lo general hay un área de “aplicaciones” donde se selecciona terminal o consola.



O bien en el ícono de aplicaciones en la línea de “buscar” escribir “terminal” si es que no está a la vista el ícono de terminal.



Una vez teniendo una terminal abierta, estamos listos para introducir comandos.



```
Microsoft Windows [Versión 10.0.19041.508]
(c) 2020 Microsoft Corporation. Todos los derechos reservados.

C:\Users\HP>

fp03alu26@samba:~
Microsoft Windows [Versión 10.0.19041.508]
(c) 2020 Microsoft Corporation. Todos los derechos reservados.

C:\Users\HP>ssh -99 fp03alu26@132.248.59.142
unknown option -- 9
usage: ssh [-46AaCfGgKkMnqsTtVvXxYy] [-B bind_interface]
          [-b bind_address] [-c cipher_spec] [-D [bind_address]:port]
          [-E log_file] [-e escape_char] [-F configfile] [-I pkcs11]
          [-i identity_file] [-J [user@]host[:port]] [-L address]
          [-l login_name] [-m mac_spec] [-O ctl_cmd] [-o option] [-p port]
          [-Q query_option] [-R address] [-S ctl_path] [-W host:port]
          [-w local_tun[:remote_tun]] destination [command]
```

C:\Users\HP>ssh -p 99 fp03alu26@132.248.59.142
fp03alu26@132.248.59.142's password:
Last login: Mon Oct 12 11:00:31 2020 from 201.141.84.82

SAMBA

[fp03alu26@samba ~]\$

La sintaxis que siguen los comandos es la siguiente:

comando [-opciones] [argumentos]

Esto es, el nombre del comando, seguido de algunas banderas (opciones) para modificar la ejecución del mismo y, al final, se puede incluir un argumento (ruta, ubicación, archivo, etcétera) dependiendo del comando. Tanto las opciones como los argumentos son opcionales.

Ejemplo (comando ls)

El comando `ls` permite listar los elementos que existen en alguna ubicación del sistema de archivos de Linux. Por defecto lista los elementos que existen en la ubicación actual; Linux nombra la ubicación actual con un punto (`.`) por lo que

`ls`

y

`ls .`

realizan exactamente lo mismo.

El comando `ls` realiza acciones distintas dependiendo de las banderas que utilice, por ejemplo, si se utiliza la opción `l` se genera un listado largo de la ubicación actual:

`ls -l`

Es posible listar los elementos que existen en cualquier ubicación del sistema de archivos, para ello hay que ejecutar el comando especificando como argumento la ubicación donde se desean listar los elementos. Si queremos ver los archivos que se encuentran en la raíz, usamos:

`ls /`

Para ver los usuarios del equipo local, revisamos el directorio `home` que parte de la raíz (`/`):

`ls /home`

Tanto las opciones como los argumentos se pueden combinar para generar una ejecución más específica:

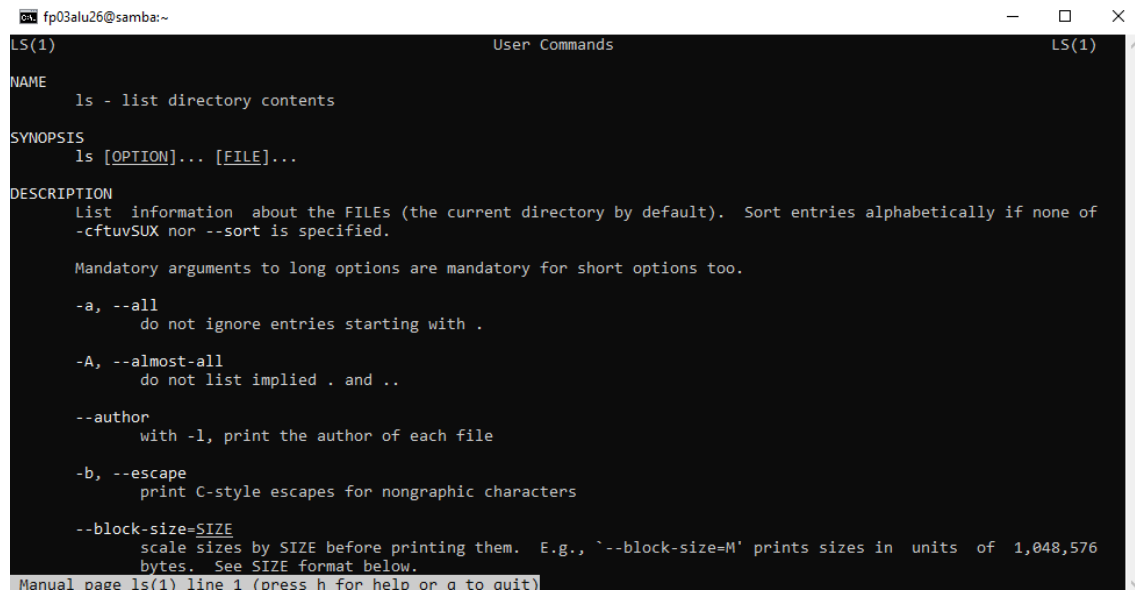
`ls -l /home`

GNU/Linux proporciona el comando `man`, el cual permite visualizar la descripción de cualquier comando así como la manera en la que se puede utilizar.

`man ls`



```
fp03alu26@samba:~  
SAMBAA  
[fp03alu26@samba ~]$ ls  
Escritorio  
[fp03alu26@samba ~]$ ls .  
Escritorio  
[fp03alu26@samba ~]$ ls -l  
Escritorio  
[fp03alu26@samba ~]$ ls /  
bin  database  etc  lib  lost+found  mnt  proc  run  sitio  srv  tmp  usr  webserver  
boot  dev  home  lib64  media  opt  root  sbin  software  sys  users  var  
[fp03alu26@samba ~]$ ls/home  
-bash: ls/home: No existe el fichero o el directorio  
[fp03alu26@samba ~]$ ls /home  
administrador  mena  MNR  squid  
[fp03alu26@samba ~]$ ls -l /home  
administrador  
mena  
MNR  
squid  
[fp03alu26@samba ~]$
```



```
fp03alu26@samba:~
LS(1) User Commands LS(1)
NAME
  ls - list directory contents
SYNOPSIS
  ls [OPTION]... [FILE]...
DESCRIPTION
  List information about the FILES (the current directory by default). Sort entries alphabetically if none of
  -cftuvSUX nor --sort is specified.

  Mandatory arguments to long options are mandatory for short options too.

  -a, --all
      do not ignore entries starting with .
  -A, --almost-all
      do not list implied . and ..
  --author
      with -l, print the author of each file
  -b, --escape
      print C-style escapes for nongraphic characters
  --block-size=SIZE
      scale sizes by SIZE before printing them. E.g., '--block-size=M' prints sizes in units of 1,048,576
      bytes. See SIZE format below.
Manual page ls(1) line 1 (press h for help or q to quit)
```

Antes de revisar otros comandos, es importante aprender a “navegar” por el sistema de archivos de Linux en modo texto. Basándonos en la Figura 2 de esta práctica, si deseamos ver la lista de los archivos del directorio usr, podemos escribir el comando:

```
ls /usr
```

Esto es, el argumento se inicia con / indicando que es el directorio raíz, seguido de usr que es el nombre del directorio. Cuando especificamos la ubicación de un archivo partiendo de la raíz, se dice que estamos indicando la “ruta absoluta” del archivo. Existe otra forma de especificar la ubicación de un archivo, esto es empleando la “ruta relativa”.

Si bien el punto (.) es para indicar la ubicación actual, el doble punto (..) se utiliza para referirse al directorio “padre”. De esta forma si deseamos listar los archivos que dependen de mi directorio padre se escribe el siguiente comando:

```
ls ..
```

O

```
ls ../
```

Se pueden utilizar varias referencias al directorio padre para ir navegando por el sistema de archivos, de tal manera que se realice la ubicación de un archivo a través de una ruta relativa. De la Figura 2, si nuestra cuenta depende de home, la ruta relativa para listar los archivos de del directorio usr es:

```
ls ../../usr
```

Con los primeros dos puntos se hace referencia al directorio home, con los siguientes dos puntos se refiere al directorio raíz, y finalmente se escribe el nombre del directorio usr.

```
fp03alu26@samba:~$ man ls
[fp03alu26@samba ~]$ ls /usr
bin  etc  games  include  lib  lib64  libexec  local  sbin  share  src  tmp
[fp03alu26@samba ~]$ ls ..
bash: ls: comando no encontrado...
[fp03alu26@samba ~]$ ls ..
fp03alu01 fp03alu07 fp03alu13 fp03alu19 fp03alu25 fp03alu31 fp03alu37 fp03alu43 fp03alu49 fp03alu55
fp03alu02 fp03alu08 fp03alu14 fp03alu20 fp03alu26 fp03alu32 fp03alu38 fp03alu44 fp03alu50 fp03alu56
fp03alu03 fp03alu09 fp03alu15 fp03alu21 fp03alu27 fp03alu33 fp03alu39 fp03alu45 fp03alu51 fp03alu57
fp03alu04 fp03alu10 fp03alu16 fp03alu22 fp03alu28 fp03alu34 fp03alu40 fp03alu46 fp03alu52
fp03alu05 fp03alu11 fp03alu17 fp03alu23 fp03alu29 fp03alu35 fp03alu41 fp03alu47 fp03alu53
fp03alu06 fp03alu12 fp03alu18 fp03alu24 fp03alu30 fp03alu36 fp03alu42 fp03alu48 fp03alu54
[fp03alu26@samba ~]$ ls ../usr
ls: no se puede acceder a ../usr: No existe el fichero o el directorio
[fp03alu26@samba ~]$
```

Ejemplo (comando touch)

El comando touch permite crear un archivo de texto, su sintaxis es la siguiente:

`touch nombre_archivo[.ext]`

En GNU/Linux no es necesario agregar una extensión al archivo creado, sin embargo, es recomendable hacerlo para poder identificar el tipo de archivo creado.

Ejemplo (comando mkdir)

El comando mkdir permite crear una carpeta, su sintaxis es la siguiente:

`mkdir nombre_carpeta`

Para crear una carpeta en nuestra cuenta, que tenga como nombre "tarefas" se escribe el siguiente comando:

`mkdir tareas`

Ejemplo (comando cd)

El comando cd permite ubicarse en una carpeta, su sintaxis es la siguiente:

`cd nombre_carpeta`

Por lo que, si queremos situarnos en la carpeta "tarefas" creada anteriormente, se escribe el comando:

`cd tareas`

Ahora, si deseamos situarnos en la carpeta de inicio de nuestra cuenta, que es la carpeta padre, escribimos el comando:

`cd ..`

Ejemplo (comando pwd)

El comando pwd permite conocer la ubicación actual(ruta), su sintaxis es la siguiente:

`pwd`

Ejemplo (comando find)

El comando find permite buscar un elemento dentro del sistema de archivos, su sintaxis

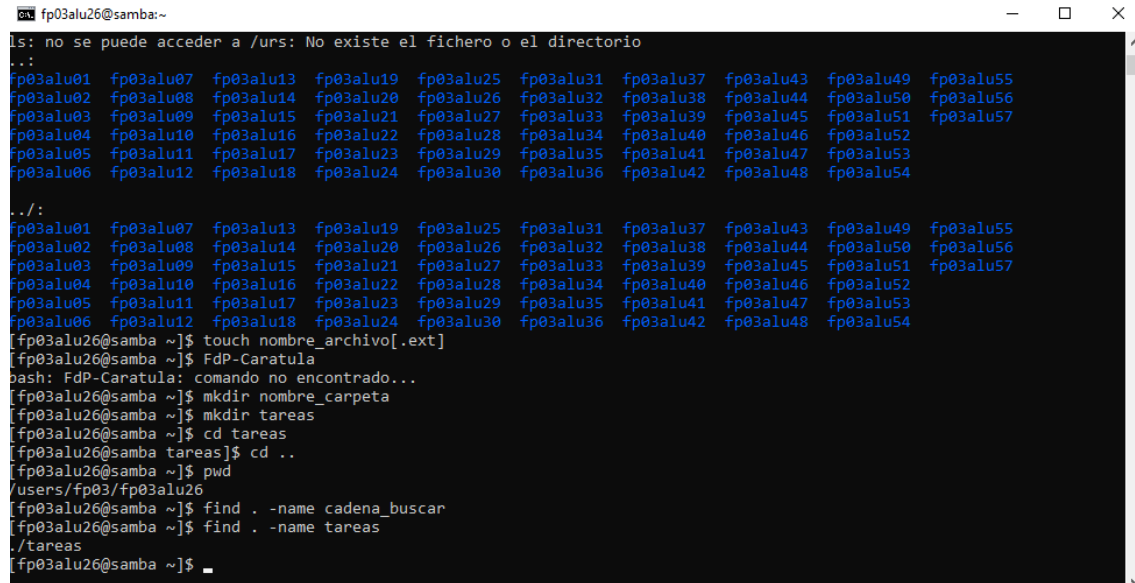
es la siguiente:

```
find . -name cadena_buscar
```

Al comando find hay que indicarle en qué parte del sistema de archivos va a iniciar la búsqueda. En el ejemplo anterior la búsqueda se inicia en la posición actual (uso de .). Además, utilizando la bandera -name permite determinar la cadena a buscar (comúnmente es el nombre de un archivo).

Si queremos encontrar la ubicación del archivo tareas, se escribe el siguiente comando:

```
find . -name tareas
```



```
fp03alu26@samba:~  
ls: no se puede acceder a /urs: No existe el fichero o el directorio  
.:  
fp03alu01 fp03alu07 fp03alu13 fp03alu19 fp03alu25 fp03alu31 fp03alu37 fp03alu43 fp03alu49 fp03alu55  
fp03alu02 fp03alu08 fp03alu14 fp03alu20 fp03alu26 fp03alu32 fp03alu38 fp03alu44 fp03alu50 fp03alu56  
fp03alu03 fp03alu09 fp03alu15 fp03alu21 fp03alu27 fp03alu33 fp03alu39 fp03alu45 fp03alu51 fp03alu57  
fp03alu04 fp03alu10 fp03alu16 fp03alu22 fp03alu28 fp03alu34 fp03alu40 fp03alu46 fp03alu52  
fp03alu05 fp03alu11 fp03alu17 fp03alu23 fp03alu29 fp03alu35 fp03alu41 fp03alu47 fp03alu53  
fp03alu06 fp03alu12 fp03alu18 fp03alu24 fp03alu30 fp03alu36 fp03alu42 fp03alu48 fp03alu54  
./:  
fp03alu01 fp03alu07 fp03alu13 fp03alu19 fp03alu25 fp03alu31 fp03alu37 fp03alu43 fp03alu49 fp03alu55  
fp03alu02 fp03alu08 fp03alu14 fp03alu20 fp03alu26 fp03alu32 fp03alu38 fp03alu44 fp03alu50 fp03alu56  
fp03alu03 fp03alu09 fp03alu15 fp03alu21 fp03alu27 fp03alu33 fp03alu39 fp03alu45 fp03alu51 fp03alu57  
fp03alu04 fp03alu10 fp03alu16 fp03alu22 fp03alu28 fp03alu34 fp03alu40 fp03alu46 fp03alu52  
fp03alu05 fp03alu11 fp03alu17 fp03alu23 fp03alu29 fp03alu35 fp03alu41 fp03alu47 fp03alu53  
fp03alu06 fp03alu12 fp03alu18 fp03alu24 fp03alu30 fp03alu36 fp03alu42 fp03alu48 fp03alu54  
[fp03alu26@samba ~]$ touch nombre_archivo[.ext]  
[fp03alu26@samba ~]$ FdP-Caratula  
bash: FdP-Caratula: comando no encontrado...  
[fp03alu26@samba ~]$ mkdir nombre_carpeta  
[fp03alu26@samba ~]$ mkdir tareas  
[fp03alu26@samba ~]$ cd tareas  
[fp03alu26@samba tareas]$ cd ..  
[fp03alu26@samba ~]$ pwd  
/users/fp03/fp03alu26  
[fp03alu26@samba ~]$ find . -name cadena_buscar  
[fp03alu26@samba ~]$ find . -name tareas  
./tareas  
[fp03alu26@samba ~]$
```

Ejemplo (comando clear)

El comando clear permite limpiar la consola, su sintaxis es la siguiente:

```
clear
```



```
fp03alu26@samba:~  
[fp03alu26@samba ~]$  
clear  
[fp03alu26@samba ~]$
```

Ejemplo (comando cp)

El comando cp permite copiar un archivo, su sintaxis es la siguiente:

```
cp archivo_origen archivo_destino
```

Si queremos una copia del archivo datos.txt con nombre datosViejos.txt en el mismo directorio, entonces se escribe el comando

```
cp datos.txt datosViejos.txt
```

Ahora, si requerimos una copia de un archivo que está en la carpeta padre en la ubicación actual y con el mismo nombre, entonces podemos emplear las rutas relativas de la siguiente forma:

```
cp ../archivo_a_copiar .
```

Es muy importante indicar como archivo destino al punto (.) para que el archivo de copia se ubique en el directorio actual.

Ejemplo (comando mv)

El comando mv mueve un archivo de un lugar a otro, en el sistema de archivos; su sintaxis es la siguiente:

```
mv ubicación_origen/archivo ubicación_destino
```

El comando mueve el archivo desde su ubicación origen hacia la ubicación deseada(destino).

Si queremos que un archivo que está en la carpeta padre, reubicarlo en el directorio actual y con el mismo nombre, entonces podemos emplear las rutas relativas de la siguiente forma:

```
mv ../archivo_a_reubicar .
```

Este comando también puede ser usado para cambiar el nombre de un archivo, simplemente se indica el nombre actual del archivo y el nuevo nombre:

```
mv nombre_actual_archivo nombre_nuevo_archivo
```

Ejemplo (comando rm)

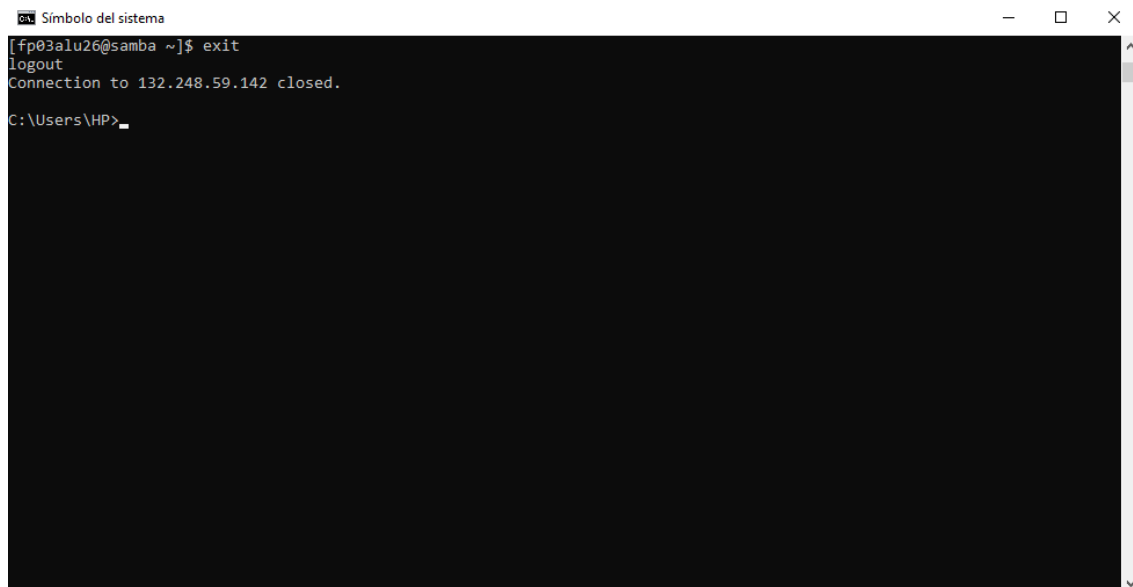
El comando rm permite eliminar un archivo o un directorio, su sintaxis es la siguiente:

```
rm nombre_archivo
```

```
rm nombre_carpeta
```

Cuando la carpeta que se desea borrar contiene información, se debe utilizar la bandera -f para forzar la eliminación. Si la carpeta contiene otras carpetas, se debe utilizar la opción -r, para realizar la eliminación recursiva.

```
fp03alu26@samba:~$ cp archivo_origen archivo_destino
cp: no se puede efectuar 'stat' sobre «archivo_origen»: No existe el fichero o el directorio
fp03alu26@samba:~$ cp datos.txt datosViejos.txt
cp: no se puede efectuar 'stat' sobre «datos.txt»: No existe el fichero o el directorio
fp03alu26@samba:~$ cp datos.txt datosViejos.txt
cp: no se puede efectuar 'stat' sobre «datos.txt»: No existe el fichero o el directorio
fp03alu26@samba:~$ cp ../archivo_a_copiar .
cp: falta el fichero de destino después de «../archivo_a_copiar»
Pruebe 'cp --help' para más información.
fp03alu26@samba:~$ mv ubicacion_origen/archivo ubicacion_destino
mv: no se puede efectuar 'stat' sobre «ubicacion_origen/archivo»: No existe el fichero o el directorio
fp03alu26@samba:~$ mv ../archivo_a_reubicar .
mv: falta el fichero de destino después de «../archivo_a_reubicar»
Pruebe 'mv --help' para más información.
fp03alu26@samba:~$ mv nombre_actual_archivo nombre_nuevo_archivo
mv: no se puede efectuar 'stat' sobre «nombre_actual_archivo»: No existe el fichero o el directorio
fp03alu26@samba:~$ rm nombre_archivo
rm: no se puede borrar «nombre_archivo»: No existe el fichero o el directorio
fp03alu26@samba:~$ rm nombre_carpeta_
```



```
Símbolo del sistema
[fp03alu26@samba ~]$ exit
logout
Connection to 132.248.59.142 closed.
C:\Users\HP>
```

Conclusiones

Al hacer esta práctica ahora puedo conocer algunos de los comandos para manejar algunos aspectos del equipo de cómputo. Me hubiera gustado que dentro de esta práctica se hubieran creado más carpetas para que así se pudiera haber evaluado mejor. Debido a esto es porque en algunas capturas de pantalla de los comandos estos no funcionaban como debían, todos los comandos respondieron, aunque fue algo complicado entrar a las máquinas de la facultad.

Bibliografía

- Óscar Vicente Huguet Soriano, Sonia Doménech Gómez. Introducción a Linux. [Figura 1]. Consulta: Junio de 2015. Disponible en: http://mural.uv.es/oshuso/81_introduccion_a_linux.html
- Pablo Delgado. Integración de sistemas. Linux y su sistema gestor de ficheros (descripciones). [Figura 2]. Consulta agosto de 2016. Disponible en: <http://todobytes.es/2014/09/integracion-de-sistemas-linux-y-su-sistema-gestor-de-ficheros-descripciones/>