



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: Adrián Ulises Mercado

Asignatura: Fundamentos de programación

Grupo: 7

No de Práctica(s): Práctica 2

Integrante(s): - Quintos Delgadillo Axel Alejandro
- Arzaba Ramirez Raymundo
- Mata Ramírez Andrea

Brigada: 7

Semestre: Primer semestre

Fecha de entrega: Miércoles 29 de Septiembre, 2021

Observaciones:

CALIFICACIÓN: _____

Práctica de estudio 02: GNU/Linux

Indice

- ***Introducción*3**
- ***Desarrollo*.....4**
- ***Conclusión general*.....12**
- ***Conclusiones individuales*.....12**
- ***Bibliografía*.....12**

Introducción:

El **Sistema Operativo** es el conjunto de programas y datos que administra los recursos tanto de hardware como de software 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 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:

1. Gestor de memoria,
2. Administrador y planificador de procesos,
3. Sistema de archivos y
4. 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. Es necesario el uso de dispositivos de entrada y salida (hardware) y aplicaciones en modo texto (software).

GNU/Linux es un sistema operativo libre desarrollado, de los más fiables y eficientes que podemos encontrar. Aunque su naturaleza de software libre creó inicialmente ciertas reticencias por parte de usuarios y empresas, GNU/Linux ha demostrado estar a la altura de cualquier otro sistema operativo existente. Sus principales características son:

- **Multitarea:** se pueden realizar varias actividades a la vez (navegar por Internet, editar un documento, compilar un programa,...)
- **Multiusuario:** varios usuarios pueden trabajar concurrentemente en un único ordenador con varios terminales (teclado y monitor) de forma que tengan la sensación de que es el único que está trabajando en el sistema. Cada usuario almacena sus datos (programas, documentos de texto, imágenes,...) en una cuenta privada o “home”. Notar que para que sea multiusuario es imprescindible que sea multitarea.
- **Conectividad:** permite las comunicaciones en red y el acceso a recursos remotamente. Por ejemplo, podemos acceder a nuestros datos situados en una máquina a través de otro equipo, conectados ambos a Internet .
- **Multiplataforma:** se puede instalar en multitud de dispositivos, desde todo tipo de ordenadores de sobremesa y portátiles y servidores hasta videoconsolas o incluso teléfonos móviles.
- **Libre:** su código fuente está disponible. Cualquiera puede usarlo, modificarlo y distribuirlo. Una consecuencia de esto es que es gratis.

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:

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

Desarrollo:

Se inició sesión y se abrió una terminal (figura 1)

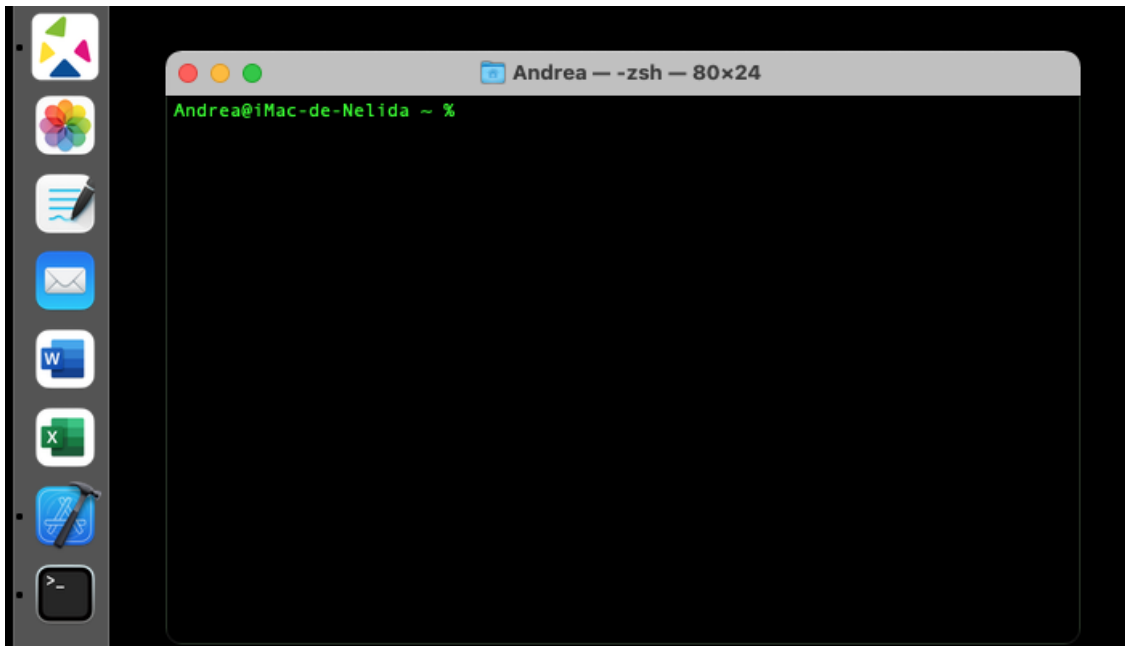


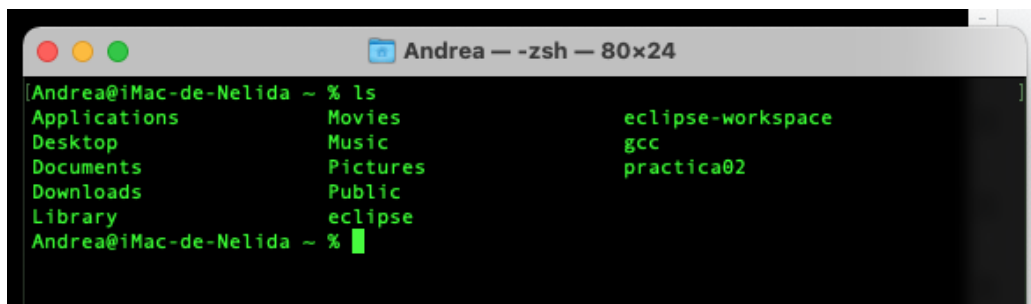
Figura 1. Terminal abierta

Posteriormente se utilizaron los comandos básicos para navegar por el sistema de archivos, 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.

Este comando 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.

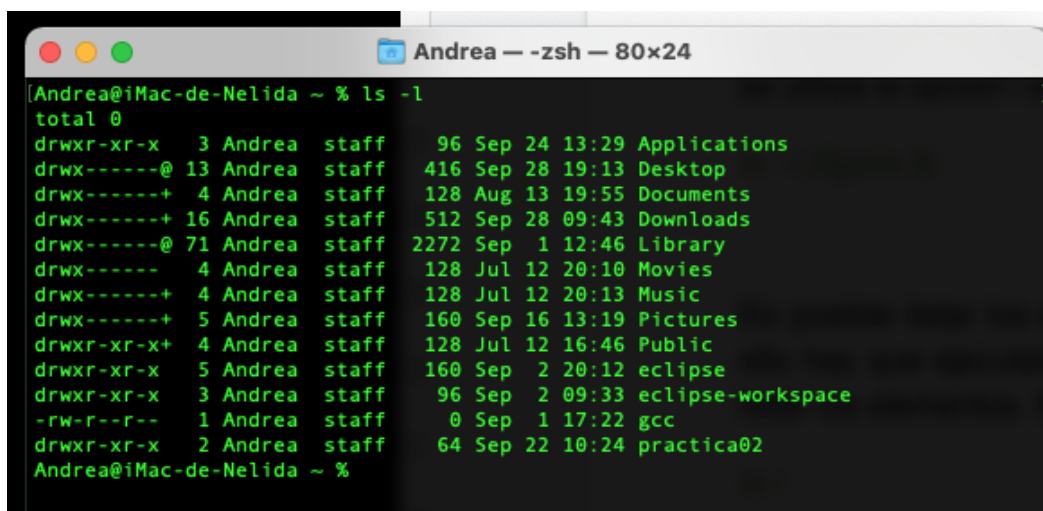
Comando `ls` (figura 2)



```
Andrea@iMac-de-Nelida ~ % ls
Applications      Movies            eclipse-workspace
Desktop           Music            gcc
Documents         Pictures         practica02
Downloads         Public
Library          eclipse
Andrea@iMac-de-Nelida ~ %
```

Figura 2. comando `ls`

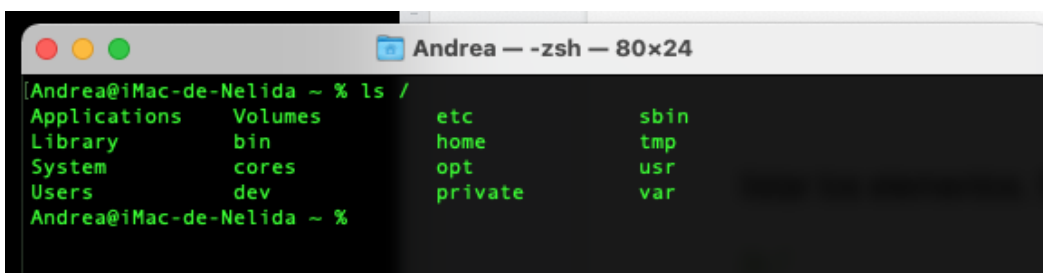
El comando `ls` realiza acciones distintas dependiendo de las banderas que utilice, por ejemplo, si se utiliza la opción `l` se genera un listado largo de la ubicación actual: `ls -l` (figura 3)



```
Andrea@iMac-de-Nelida ~ % ls -l
total 0
drwxr-xr-x  3 Andrea  staff   96 Sep 24 13:29 Applications
drwx-----@ 13 Andrea  staff  416 Sep 28 19:13 Desktop
drwx-----+  4 Andrea  staff  128 Aug 13 19:55 Documents
drwx-----+ 16 Andrea  staff  512 Sep 28 09:43 Downloads
drwx-----@ 71 Andrea  staff 2272 Sep  1 12:46 Library
drwx-----  4 Andrea  staff  128 Jul 12 20:10 Movies
drwx-----+  4 Andrea  staff  128 Jul 12 20:13 Music
drwx-----+  5 Andrea  staff  160 Sep 16 13:19 Pictures
drwxr-xr-x+  4 Andrea  staff  128 Jul 12 16:46 Public
drwxr-xr-x  5 Andrea  staff  160 Sep  2 20:12 eclipse
drwxr-xr-x  3 Andrea  staff   96 Sep  2 09:33 eclipse-workspace
-rw-r--r--  1 Andrea  staff    0 Sep  1 17:22 gcc
drwxr-xr-x  2 Andrea  staff   64 Sep 22 10:24 practica02
Andrea@iMac-de-Nelida ~ %
```

Figura 3. Comando `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 /` (Figura 4)



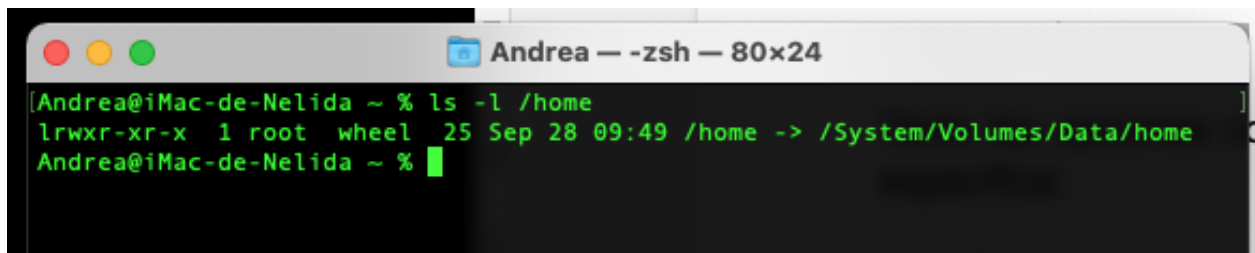
```
Andrea@iMac-de-Nelida ~ % ls /
Applications  Volumes      etc          sbin
Library       bin          home         tmp
System        cores        opt          usr
Users         dev          private      var
Andrea@iMac-de-Nelida ~ %
```

Figura 4. Comando `ls /`

Para ver los usuarios del equipo local, revisamos el directorio home que parte de la raíz (/): *ls /home* (Figura 5)

Figura 5. Comando *ls /home*

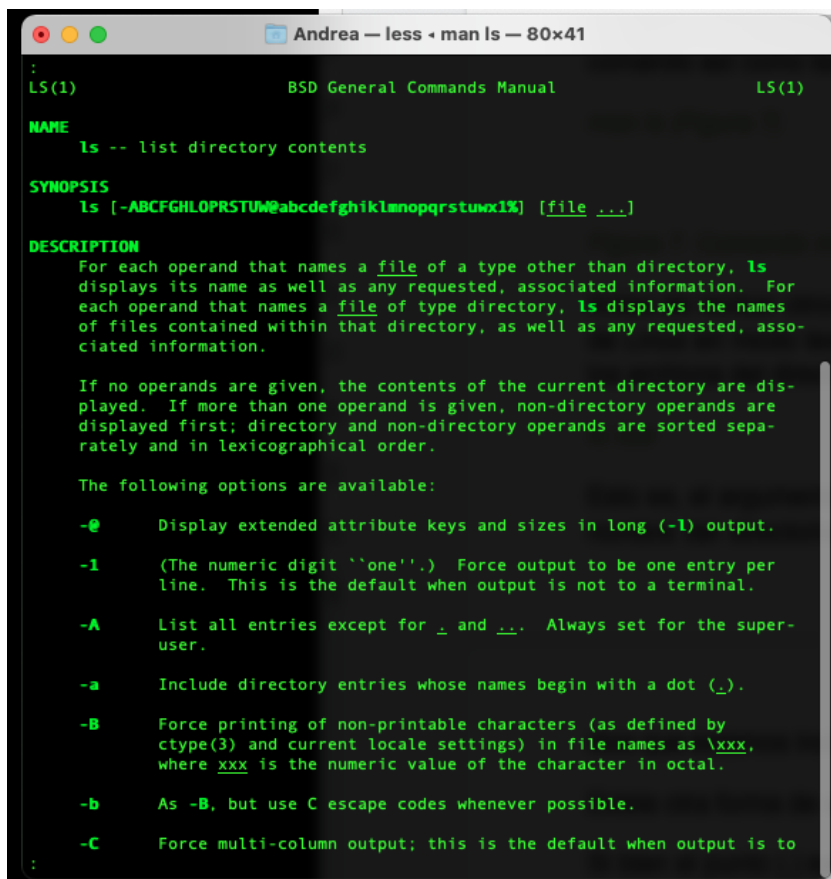
Tanto las opciones como los argumentos se pueden combinar para generar una ejecución más específica: *ls -l /home* (Figura 6)



```
Andrea@iMac-de-Nelida ~ % ls -l /home
lrwxr-xr-x  1 root  wheel  25 Sep 28 09:49 /home -> /System/Volumes/Data/home
Andrea@iMac-de-Nelida ~ %
```

Figura 6. Comando *ls -l /home*

GNU/Linux proporciona el comando *man*, que permite visualizar la descripción de cualquier comando así como la manera en la que se puede utilizar. *man ls* (Figura 7)



```
Andrea — less — man ls — 80x41
:
LS(1)                                BSD General Commands Manual                                LS(1)

NAME
  ls -- list directory contents

SYNOPSIS
  ls [-ABCFGHLOPRSTUW@abcdefghiklmnopqrstuwx1] [file ...]

DESCRIPTION
  For each operand that names a file of a type other than directory, ls
  displays its name as well as any requested, associated information. For
  each operand that names a file of type directory, ls displays the names
  of files contained within that directory, as well as any requested, asso-
  ciated information.

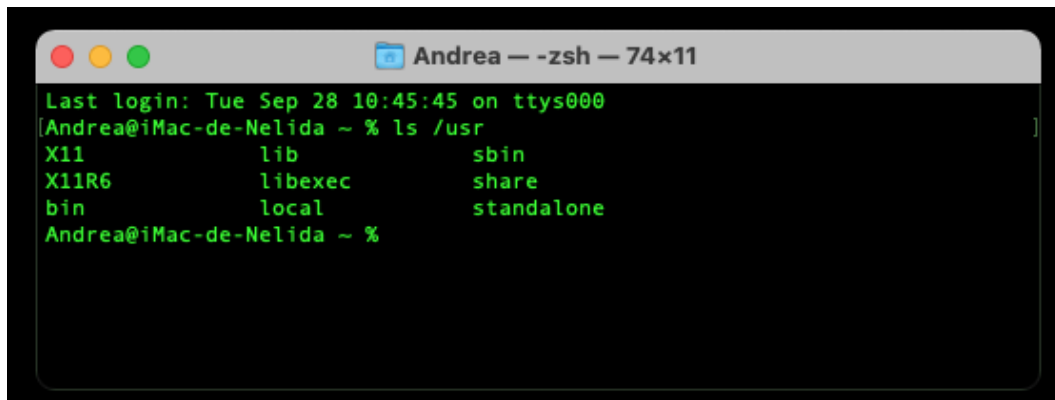
  If no operands are given, the contents of the current directory are dis-
  played. If more than one operand is given, non-directory operands are
  displayed first; directory and non-directory operands are sorted sepa-
  rately and in lexicographical order.

  The following options are available:

  -e      Display extended attribute keys and sizes in long (-l) output.
  -l      (The numeric digit 'one'.) Force output to be one entry per
  line. This is the default when output is not to a terminal.
  -A      List all entries except for . and .. Always set for the super-
  user.
  -a      Include directory entries whose names begin with a dot (.).
  -B      Force printing of non-printable characters (as defined by
  ctype(3) and current locale settings) in file names as \xxx,
  where xxx is the numeric value of the character in octal.
  -b      As -B, but use C escape codes whenever possible.
  -C      Force multi-column output; this is the default when output is to
  :
```

Figura 7. Comando *man ls*

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` (figura 8)



```
Andrea — -zsh — 74x11
Last login: Tue Sep 28 10:45:45 on ttys000
[Andrea@iMac-de-Nelida ~ % ls /usr
X11      lib      sbin
X11R6    libexec  share
bin      local    standalone
Andrea@iMac-de-Nelida ~ %
```

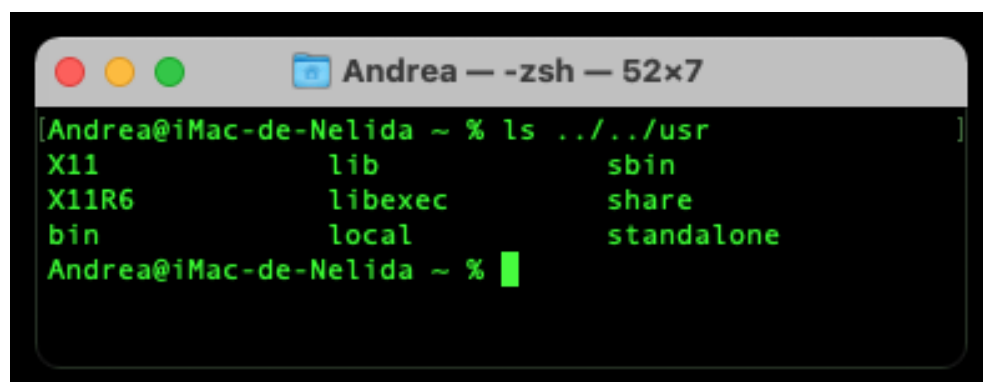
Figura 8. Comando `ls /usr`

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. Si nuestra cuenta depende de `home`, la ruta relativa para listar los archivos de del directorio `usr` es: `ls ../../usr` (figura 9)



```
Andrea — -zsh — 52x7
[Andrea@iMac-de-Nelida ~ % ls ../../usr
X11      lib      sbin
X11R6    libexec  share
bin      local    standalone
Andrea@iMac-de-Nelida ~ %
```

Figura 9. Comando `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.

Ejemplo: comando touch

El comando touch permite crear un archivo de texto, su sintaxis es la siguiente: *touch nombre_archivo[.ext]* (Figura 10)

```
C:\Users\Ray>touch hola.txt
C:\Users\Ray>ls ../Ray
'3D Objects'
AppData
'Application Data'
'Configuraci'$'\303\263''n local'
Contacts
Cookies
'Datos de programa'
Desktop
Documents
Downloads
'Entorno de red'
Favorites
Impresoras
Links
'Men'$'\303\272'' Inicio'
'Mis documentos'
Music
NTUSER.DAT
NTUSER.DAT{53b39e88-18c4-11ea-a811-000d3aa4692b}.TM.blf
NTUSER.DAT{53b39e88-18c4-11ea-a811-000d3aa4692b}.TMContainer000000000000000001.regtrans-ms
NTUSER.DAT{53b39e88-18c4-11ea-a811-000d3aa4692b}.TMContainer000000000000000002.regtrans-ms
OneDrive
Pictures
Plantillas
Reciente
'Saved Games'
Searches
SendTo
Videos
hola.txt
ntuser.dat.LOG1
ntuser.dat.LOG2
ntuser.ini
C:\Users\Ray>
```

Figura 10. Comando touch

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 “tareas” se escribe el siguiente comando: *mkdir tareas* (figura 11)

```
Andrea — -zsh — 52x7
[Andrea@iMac-de-Nelida ~ % mkdir tareas
Andrea@iMac-de-Nelida ~ %]
```

Figura 11. 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 “tareas” creada anteriormente, se escribe el comando: `cd tareas` (Figura 12)

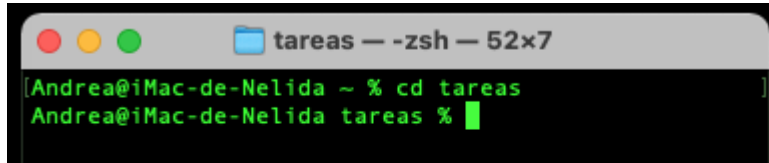
A terminal window titled 'tareas — -zsh — 52x7'. The prompt is 'Andrea@iMac-de-Nelida ~ %'. The user enters 'cd tareas' and presses enter. The prompt changes to 'Andrea@iMac-de-Nelida tareas %'.

Figura 12. Comando cd tareas

Ahora, si deseamos situarnos en la carpeta de inicio de nuestra cuenta, que es la carpeta padre, escribimos el comando: `cd ..` (figura 13)

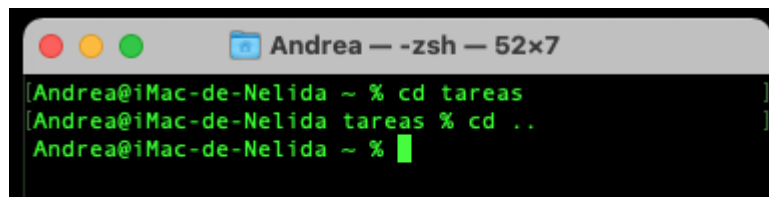
A terminal window titled 'Andrea — -zsh — 52x7'. The prompt is 'Andrea@iMac-de-Nelida ~ %'. The user enters 'cd tareas' and presses enter. The prompt changes to 'Andrea@iMac-de-Nelida tareas %'. The user then enters 'cd ..' and presses enter. The prompt returns to 'Andrea@iMac-de-Nelida ~ %'.

Figura 13. Comando cd ..

Ejemplo: comando pwd, comando find

El comando `pwd` permite conocer la ubicación actual(ruta), su sintaxis es la siguiente: `pwd` (figura 14)

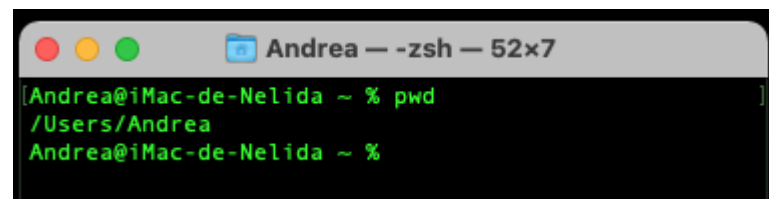
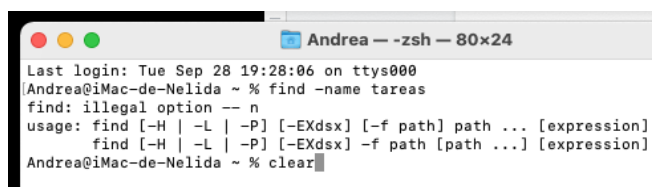
A terminal window titled 'Andrea — -zsh — 52x7'. The prompt is 'Andrea@iMac-de-Nelida ~ %'. The user enters 'pwd' and presses enter. The output is '/Users/Andrea'. The prompt returns to 'Andrea@iMac-de-Nelida ~ %'.

Figura 14. Comando pwd

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` (Figura 15)



```
Andrea — zsh — 80x24
Last login: Tue Sep 28 19:28:06 on ttys000
Andrea@iMac-de-Nelida ~ % find -name tareas
find: illegal option -- n
usage: find [-H | -L | -P] [-EXdsx] [-f path] path ... [expression]
       find [-H | -L | -P] [-EXdsx] -f path [path ...] [expression]
Andrea@iMac-de-Nelida ~ % clear
```

Figura 15. Comando `find . -name tareas`

El comando `clear` permite limpiar la consola, su sintaxis es la siguiente: `clear` (Figura 16)



```
Andrea — zsh — 80x24
Last login: Tue Sep 28 19:28:06 on ttys000
Andrea@iMac-de-Nelida ~ % find -name tareas
find: illegal option -- n
usage: find [-H | -L | -P] [-EXdsx] [-f path] path ... [expression]
       find [-H | -L | -P] [-EXdsx] -f path [path ...] [expression]
Andrea@iMac-de-Nelida ~ % clear

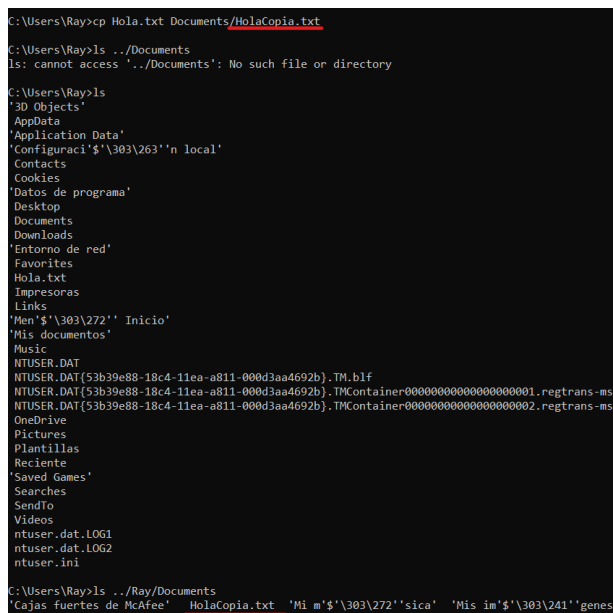
Andrea — zsh — 80x24
Andrea@iMac-de-Nelida ~ %
```

Figura 16. Comanco `clear`

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` (Figura 17)



```
C:\Users\Ray>cp Hola.txt Documents/HolaCopia.txt
C:\Users\Ray>ls ../Documents
ls: cannot access '../Documents': No such file or directory
C:\Users\Ray>ls
'3D Objects'
'AppData'
'Application Data'
'Configuraci'$'\303\263''n local'
'Contacts'
'Cookies'
'Datos de programa'
'Desktop'
'Documents'
'Downloads'
'Entorno de red'
'Favorites'
'Hola.txt'
'Impresoras'
'Links'
'Men'$'\303\272'' Inicio'
'Mis documentos'
'Music'
'NTUSER.DAT'
'NTUSER.DAT{53b39e88-18c4-11ea-a811-000d3aa4692b}.TM.blf'
'NTUSER.DAT{53b39e88-18c4-11ea-a811-000d3aa4692b}.TMContainer000000000000000001.regtrans-ms'
'NTUSER.DAT{53b39e88-18c4-11ea-a811-000d3aa4692b}.TMContainer000000000000000002.regtrans-ms'
'OneDrive'
'Pictures'
'Plantillas'
'Reciente'
'Saved Games'
'Searches'
'SendTo'
'Videos'
'ntuser.dat.LOG1'
'ntuser.dat.LOG2'
'ntuser.ini'
C:\Users\Ray>ls ../Ray/Documents
'Cajas fuertes de McAfee'
'HolaCopia.txt'
'Mi m'$'\303\272''sica'
'Mis in'$'\303\241''genes'
```

Figura 17. Comando `cp`

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 (Figura 18)

```
C:\Users\Ray>ls ../Ray/Documents
'Cajas fuertes de McAfee' 'Mi m'$'\303\272''sica' 'Mis v'$'\303\255''deos' desktop.ini
HoloCopia.txt 'Mis im'$'\303\241''genes' 'Plantillas personalizadas de Office'

C:\Users\Ray>mv Holo.txt Documents/Adios.txt

C:\Users\Ray>ls ../Ray/Documents
Adios.txt HoloCopia.txt 'Mis im'$'\303\241''genes' 'Plantillas personalizadas de Office'
'Cajas fuertes de McAfee' 'Mi m'$'\303\272''sica' 'Mis v'$'\303\255''deos' desktop.ini
```

Figura 18. Comando mv

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 (Figura 19)

```
Andrea — -zsh — 64x9
Andrea@iMac-de-Nelida ~ % rm tareas
rm: tareas: is a directory
Andrea@iMac-de-Nelida ~ %
```

Figura 19. Comando rm

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.

Conclusión general:

En esta práctica se pudieron lograr los dos objetivos que se plantearon al principio, ya que pudimos identificar la importancia que tiene el sistema operativo de una computadora, así como sus funciones, también logramos explorar un sistema operativo GNU/Linux para conocer y utilizar los comandos básicos.

Conclusiones individuales:

Andrea Mata Ramírez:

Me pareció muy útil e interesante esta práctica ya que pude identificar algunos beneficios de aprender a usar la línea de comandos, los beneficios son que se obtiene un mayor control sobre el sistema operativo, también gracias a la línea de comandos se pueden realizar desde operaciones sencillas hasta operaciones más complejas, que puede ser que al realizarlas de manera manual llevarán más tiempo, y para finalizar de hablar de los beneficios que pude identificar, al buscar en internet pude ver que la interfaz de línea de comandos podría ser necesaria para otras herramientas que un desarrollador necesita, así que saber manejarla e identificar los comandos es de gran ayuda para la carrera que estoy cursando actualmente.

Raymundo Arzaba Ramírez:

La práctica fue interesante y útil, considero que es una buena introducción para lo que veremos a futuro, además que es de ayuda para el entendimiento sobre los archivos que se encuentran en el SO y el cómo manejarlos mediante los comandos vistos en la práctica.

Quintos Delgadillo Axel Alejandro:

Esta práctica nos dio una buena introducción al uso de los comandos, como poder utilizar los diferentes archivos, crear carpetas y navegar por todas estas secciones desde la terminal. Es un buen comienzo para empezar a trabajar con los diferentes comandos de los lenguajes de programación, desde el lenguaje C, hasta los que veremos posteriormente.

Referencias:

- Viñas, R. B. (2003, noviembre). El Sistema Operativo GNU/Linux. VOC. <https://softlibre.unizar.es/manuales/linux/868.pdf>
- Deyimar A. (2021, 29 julio). 35 comandos básicos de Linux que todo usuario debe saber. hostinger.mx. <https://www.hostinger.mx/tutoriales/linux-comandos>