



INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO



SISTEMAS OPERATIVOS

PRÁCTICA 2:

Entorno de Linux

ALUMNOS:

Martínez Ramírez Sergi Alberto – 2020630260 - 2CV15

Meza Vargas Brandon David – 2020630288 - 2CM15

Peña Atanasio Alberto – 2020630367 - 2CV15

Sarmiento Gutiérrez Juan Carlos – 2020630386 - 2CV15

PROFESOR:

José Alfredo Jiménez Benítez

Índice de contenido

Glosario de términos	7
Contenido	8
Escritorios GNOME y KDE.....	8
Entornos Command Line Interface y Graphical User Interface.....	9
Terminal de Linux.....	10
Direccionamiento relativo y absoluto	11
Redireccionamiento.....	11
Clasificación de los comandos en Linux.....	11
Comandos básicos.....	12
Páginas de ayuda	12
Operaciones en el directorio.....	12
Operaciones en archivos.....	12
Gestión de los permisos.....	12
Opciones de búsqueda.....	12
Información sobre los usuarios	12
Gestión de cuentas de usuario	13
Gestión del sistema.....	13
Información del sistema	13
Información acerca del hardware	13
Gestión de procesos.....	13
Pager.....	13
Editores	13
Gestión de redes	13
Archivar y comprimir.....	14
Gestion de particiones	14
Variables de entorno	14
Desarrollo de la práctica.....	15
Comandos.....	15
Ejemplos de direccionamiento absoluto.....	28
Ejemplos de direccionamiento relativo	29
Borrado de archivos con rm y los comodines “*” y “?”.....	30

Redireccionamiento con los comandos > y >>	31
Instalación de software por línea de comandos	33
Instalación de software por el entorno gráfico	35
Ejemplos usando jerarquía en los directorios para mover y copiar archivos y directorios desde terminal	39
Agregar, verificar, entrar en sesión y borrar usuario.....	41
Hola mundo con distintos editores	42
Programa 1a	49
Programa 2a	50
Programa 3a	52
Conclusiones	54
Bibliografía	56
Anexos.....	57
Código programa 2a.....	57
Código programa 3ª.....	58

Índice de figuras

Imagen 1. Se muestra el escritorio GNOME.....	8
Imagen 2. Se muestra el escritorio KDE.....	9
Imagen 3. Se muestra un ejemplo de CLI.....	9
Imagen 4. Se muestra información de la terminal.....	10
Imagen 5. Algunas variables de entorno de Linux.....	14
Imagen 6. Comando cal en terminal Linux.....	15
Imagen 7. Comando clear en terminal Linux.....	15
Imagen 8. Comando apt con la instrucción update en terminal Linux.....	16
Imagen 9. Comando rm en la terminal de Linux.....	16
Imagen 10. Comando date en la terminal de Linux.....	16
Imagen 11. Comando ifconfig en terminal Linux.....	17
Imagen 12. Comando exit en terminal Linux.....	17
Imagen 13. Comando mv en la terminal de Linux.....	17
Imagen 14. Comando echo en la terminal de Linux.....	18
Imagen 15. Comando df en la terminal de Linux.....	18
Imagen 16. Comando ps en la terminal de Linux.....	19
Imagen 17. Comando more en la terminal Linux.....	19
Imagen 18. Comando time en la terminal Linux.....	19
Imagen 19. Comando du en la terminal Linux.....	20
Imagen 20. Comando ps -fea en la terminal Linux.....	20
Imagen 21. Comando less en la terminal Linux.....	20
Imagen 22. Comando uname en la terminal Linux.....	21
Imagen 23. Comando pstree en la terminal Linux.....	21
Imagen 24. Comando man en la terminal Linux.....	21
Imagen 25. Comando mkdir en la terminal Linux.....	22
Imagen 26. Comando w en la terminal Linux.....	22
Imagen 27. Comando kill -1 -9 en la terminal Linux.....	22
Imagen 28. Comando cat en la terminal Linux.....	23
Imagen 29. Comando pico en la terminal Linux.....	23
Imagen 30. Comando who en la terminal Linux.....	23
Imagen 31. Comando trap -1 en la terminal Linux.....	24
Imagen 32. Comando fg en la terminal Linux.....	24
Imagen 33. Comando nano en la terminal Linux.....	24
Imagen 34. Comando bash en la terminal Linux.....	25
Imagen 35. Comando pwd en la terminal Linux.....	25
Imagen 36. Comando cd en la terminal Linux.....	25
Imagen 37. Comando vi en la terminal Linux.....	26
Imagen 38. Comando wc en la terminal Linux.....	26
Imagen 39. Comando su en la terminal Linux.....	26
Imagen 40. Comando ls en la terminal Linux.....	27
Imagen 41. Comando apt-get en la terminal Linux.....	27
Imagen 42. Comando sudo en la terminal Linux.....	27
Imagen 43. Comando ls -la en la terminal Linux.....	28

Imagen 44. Ruta absoluta del archivo “hola.txt”	28
Imagen 45. Accediendo a un directorio en específico desde otro	28
Imagen 46. Accediendo a un directorio dentro de otro directorio distinto	29
Imagen 47. Accediendo a un directorio con ruta relativa.....	29
Imagen 48. Accediendo a un directorio de la misma rama	29
Imagen 49. Accediendo a un directorio con su ruta casi absoluta.....	30
Imagen 50. Comando rm.....	30
Imagen 51. Comando rm*	30
Imagen 52. Comando rm?	31
Imagen 53. Comando ifconfig redireccionado al archivo salida.txt.....	31
Imagen 54. Archivo “salida.txt” con el resultado del comando ifconfig	32
Imagen 55. Comando echo redireccionado al archivo salida.txt.....	32
Imagen 56. Palabra hola concatenada con el texto del archivo	33
Imagen 57. Comando para instalar g++.	33
Imagen 58. Primera parte de la instalación	34
Imagen 59. Segunda parte de la instalación	34
Imagen 60. Comprobando el funcionamiento del software.....	35
Imagen 61. Escritorio de Ubuntu señalando la aplicación de Ubuntu Software	35
Imagen 62. Aplicación Ubuntu Software, con el botón buscar señalado	36
Imagen 63. Barra de búsqueda con la leyenda “sublime”	36
Imagen 64. Resultados de búsqueda	37
Imagen 65. Página de Sublime lista para instalar.....	37
Imagen 66. Credenciales del usuario solicitadas.....	37
Imagen 67. Instalando Sublime text.....	38
Imagen 68. Instalación completa, botón Instalar cambia por botón Remove	38
Imagen 69. Escritorio de Ubuntu, con el menú de aplicaciones señalado	38
Imagen 70. Aplicación exitosamente instalada	39
Imagen 71. Mover todos los archivos de texto.....	39
Imagen 72. Copiar todos los archivos tipo “c” con numeración	40
Imagen 73. Mover todos los archivos de texto a un directorio, con su ruta absoluta.....	40
Imagen 74. Copiar todos los archivos de un directorio a otro.....	40
Imagen 75. Mover directorios (y su contenido) a otros directorios	41
Imagen 76. Se muestra la creación de un nuevo usuario.....	41
Imagen 77. Se muestra la verificación de que existe el usuario.	42
Imagen 78. Se ve el inicio de sesión del usuario.....	42
Imagen 79. Se ve el comando para borrar usuarios.....	42
Imagen 80. Carpeta principal que contiene la carpeta “practicas”	43
Imagen 81. Acceso a la carpeta practicas desde la terminal.....	43
Imagen 82. Apertura de nano desde la terminal.....	44
Imagen 83. Edición de un programa desde Nano.....	44
Imagen 84. Mensaje que arroja el terminal después de presionar CTRL + O.....	44
Imagen 85. Mensaje de líneas de código escritas, después de guardar	45
Imagen 86. Compilación y ejecución del programa editado en nano.....	45
Imagen 87. Apertura de pico con el terminal	45

Imagen 88. Edición del programa con pico.....	46
Imagen 89. Guardado del archivo	46
Imagen 90. Mensaje de líneas de código escritas, después de guardar	47
Imagen 91. Compilación y ejecución del programa, utilizando pico.	47
Imagen 92. Apertura del editor vi	48
Imagen 93. Código editado con vi.	48
Imagen 94. Guardado del programa y salida del editor	49
Imagen 95. Compilación y ejecución con vi.	49
Imagen 96. Código del programa que imprime la variable de entorno HOME	50
Imagen 97. Compilación y ejecución del Programa1.c.....	50
Imagen 98. Se muestra la función escribeNum().	51
Imagen 99. Compilación del programa2a.	51
Imagen 100. Se muestra la ejecución del programa2a.c	51
Imagen 101. se muestra el archivo números.txt y su contenido.	52
Imagen 102. Función leeNum() del programa3a.....	53
Imagen 103. Se muestra la función operaciones().	53
Imagen 104. Se muestra la compilación y ejecución del programa 3a.	53

Glosario de términos

GNU/Linux: GNU/Linux es la denominación técnica y generalizada que reciben una serie de sistemas operativos de tipo Unix, que también suelen ser de código abierto, multiplataforma, multiusuario y multitarea.

Comando: es una instrucción específica dada a una aplicación informática para realizar algún tipo de tarea o función.

Entorno de escritorio: es un conjunto de software para ofrecer al usuario de una computadora una interacción amigable y cómoda.

Terminal: es una forma generalizada de llamar a la interfaz de usuario de línea de comandos: una pantalla (generalmente, de color de fondo negro sobre letras blancas) donde escribiendo comandos (secuencias de palabras especiales) ordenamos al sistema realizar acciones concretas.

Editor de texto: es un programa informático que permite crear y modificar archivos digitales compuestos únicamente por textos sin formato.

GUI: El entorno gráfico de usuario (GUI) son los recursos visuales que presentan la información y procesos que se pueden ejecutar en un programa o un sistema operativo.

Contenido

Escritorios GNOME y KDE

Tanto GNOME y KDE son entornos de escritorio, los cuales son paquetes de software diseñados para interactuar de una manera gráfica con el sistema operativo. Los elementos que usa un entorno de escritorio son iconos, barras de tarea, ventanas y animaciones para que sean fáciles de usar por cualquier usuario.

GNOME nace como alternativa al escritorio KDE, GNOME fue iniciado por los desarrolladores mexicanos Federico Mena y Miguel de Icaza y actualmente está disponible en poco más de 48 idiomas, lo que aumenta su versatilidad. A continuación, podemos ver la imagen 1 que presenta el escritorio GNOME:



Imagen 1. Se muestra el escritorio GNOME

Podemos ver como luce GNOME, este escritorio es bastante completo y su interfaz de usuario es altamente configurable.

Como se mencionó KDE es un entorno de escritorio, ofrece un soporte para hasta 75 idiomas diferentes. Una de las ventajas de KDE, es que está hecho para ser personalizado. KDE es uno de los entornos de escritorio con mayor compatibilidad que se han diseñado para Linux. En la imagen 2 podemos ver el escritorio KDE:

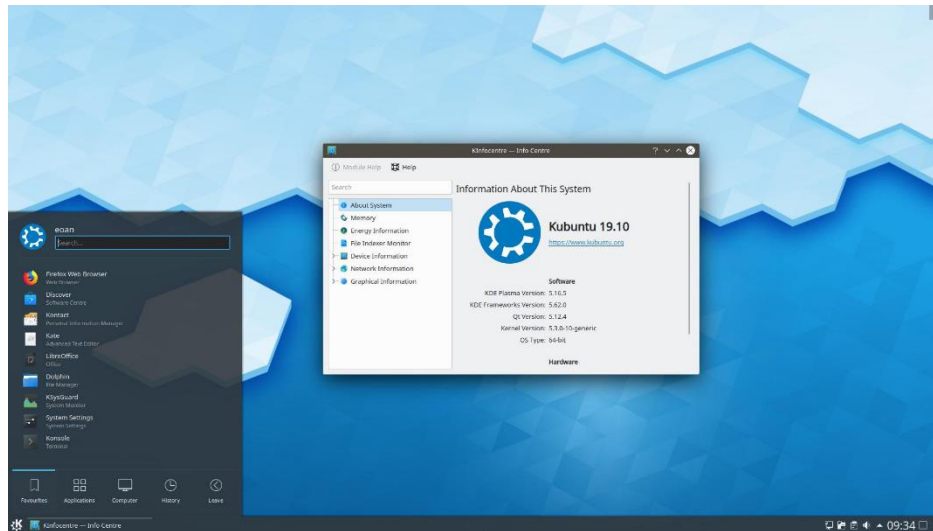


Imagen 2. Se muestra el escritorio KDE

Como se ve, KDE es muy parecido al escritorio de Windows al que estamos acostumbrados.

Entornos Command Line Interface y Graphical User Interface

El Command Line Interface (CLI) es un programa que permite a los usuarios escribir comandos de texto instruyendo a la computadora para que realice tareas específicas.

El CLI se comenzó a utilizar con mayor cantidad en la década de 1960 y era la única forma de comunicarse con la computadora. Un aspecto importante que tenemos que considerar al hablar de CLI es la shell.

La shell es una interfaz de usuario que administra el CLI y actúa como intermediario, conectando a los usuarios con el sistema operativo. entre los muchos tipos de shell, los más populares son Windows shell y Bash (para Linux y MacOS).

En la imagen 3 podemos ver un ejemplo de una CLI:

```

brandon@Brandon: ~
Archivo Editar Ver Buscar Terminal Ayuda
brandon@Brandon:~$ ls
Descargas Escritorio Música Público
Documentos Imágenes Plantillas Videos
brandon@Brandon:~$ cal
    Marzo 2021
do lu ma mi ju vi sa
   1  2  3  4  5  6
  7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
brandon@Brandon:~$

```

Imagen 3. Se muestra un ejemplo de CLI

Como se puede ver se introducen comando que deberán ser correctos para que la computadora realice la acción.

Con la llegada del mouse, se empezó a desarrollar una nueva forma para interactuar con la computadora, esto fue por medio de la GUI (Graphical User Interface). La GUI resulto increíble para los usuarios, pues era intuitiva gracias al uso de los botones y menús para representar comandos específicos.

Terminal de Linux

En la actualidad los sistemas operativos Linux tienen un intérprete de ordenes (terminal) que hace de interfaz entre el usuario y el propio sistema operativo, su nombre es Bash (Bourne Again Shell).

Bash es una herramienta Open Source perteneciente al proyecto GNU, fue escrita por Brian Fox. Bash hereda muchas propiedades de otros shells como sh, csh o zsh. Bash también es un lenguaje de scripting. Esto lo hace extremadamente potente para multitud de tareas relacionadas con la administración de sistemas, automatización de tareas, etc.

En la imagen 3 mostrada anteriormente, podemos ver como luce la terminal de Linux.

Fijándonos en la terminal podemos ver la siguiente información que se ve en la imagen 4:

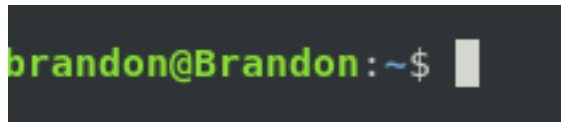


Imagen 4. Se muestra información de la terminal.

donde:

- brandon: indica el usuario conectado a la terminal
- @: significa “en”
- Brandon: indica el nombre de la máquina a la que estamos conectados
- ~: indica la ruta en la que nos encontramos
- \$: indicador para comenzar a escribir nuevos comandos

Tipos de usuario en Linux

Existen tres tipos de usuarios en Linux:

- **Usuario root:** también llamado superusuario o administrador. Es el único usuario con control total del sistema, es el único que tiene derecho a administrar las cuentas de usuario del sistema
- **Usuario normal o finales:** es el usuario habitual, tiene muy pocos permisos y pueden hacer pocas cosas como personalizar su entorno de trabajo
- **Usuarios especiales o de sistema:** los incorpora el mismo sistema, se encargan de administrar los demonios y de ciertos procesos y directorios. Ejemplos de estos son daemon, bin, sync, mail, squid, shutdown, adm, lp, apache, etc.

Direccionamiento relativo y absoluto

Cuando trabajamos con comandos es común pasar como parámetros directorios, para indicar un directorio se usa la ruta, que puede ser absoluta o relativa.

La ruta relativa indica el camino para encontrar un elemento, pero desde el directorio en el que se ejecuta el comando, es decir, desde el directorio que nos encontramos. Si me encuentro en la ruta /home y queremos ir a la ruta /home/usuario/miCarpeta tenemos que escribir: *cd usuario/miCarpeta*.

El sistema de ficheros es una estructura jerárquica que en el caso de Linux tiene una raíz que se indica cuando se pone solamente el carácter barra /. Para indicar donde se encuentra un elemento usando una ruta absoluta, indicamos todos los directorios por los que hay que pasar empezando siempre desde la raíz del sistema, sin importar de donde nos encontremos. Tomando el ejemplo de la ruta relativa, en este caso tendríamos que escribir: */home/usuario/miCarpeta*. Estas rutas a veces son muy largas, pero con la ventaja de que siempre funcionan.

Redireccionamiento

El redireccionar implica:

- Redirigir la visualización de la pantalla hacia un archivo, impresora u otro periférico
- Redirigir los mensajes de error hacia un archivo, impresora u otro dispositivo
- Sustituir la introducción vía teclado por el contenido de un archivo

Linux usa canales de entrada/salida para leer y escribir sus datos. Cualquier proceso tiene una entrada estándar, stdin, y dos salidas, la salida estándar, stdout, y la salida de errores, stderr.

Normalmente, la entrada estándar, stdin, es el teclado y la salida estándar, stdout, es la pantalla, pero si se produjo un error en la ejecución del proceso la salida por pantalla corresponderá a la salida de errores stderr. Sin embargo, puede que queramos cambiar la salida estándar o la salida de errores por un archivo. Para esto usamos el redireccionamiento con los siguientes caracteres:

- <: redirecciona la entrada estándar sustituyéndola por el archivo que se indique
- >: redirecciona la salida de un proceso al archivo que se le indique, borrando la información del archivo
- >>: redirecciona la salida de un proceso al archivo que se le indique, pero añadiéndola al final del archivo, sin borrar su contenido
- 2>: Redirecciona la salida de errores de un proceso al archivo que se le indique, borrando la información del archivo
- 2>>: redirecciona la salida de errores de un proceso al archivo que se le indique, pero añadiéndola al final del archivo, sin borrar su contenido

Clasificación de los comandos en Linux

Se recomienda usar comandos en Linux ya que son más potentes y efectivos que si usamos la GUI

Comandos básicos

Los comandos básicos incluyen las ordenes fundamentales que se usan para administrar la terminal, con los que se puede limpiar la ventana, recuperar comandos anteriores o finalizar sesión. Ejemplos de estos son: clear, exit, help, history.

Páginas de ayuda

Linux ofrece páginas de ayuda a las que podemos acceder con comandos como: apropos, info, man, pinfo, whatis.

Operaciones en el directorio

Algunos comandos permiten llevar a cabo operaciones en los directorios del sistema, como puede ser crear archivos, borrarlos y gestionarlos, así como navegar por el árbol del directorio. Ejemplos de estos comandos son: cd, chroot, ls, mkdir, mkdirhier, pwd, tree, etc.

Operaciones en archivos

Estos comandos permiten llevar a cabo operaciones sobre archivos desde la terminal, se pueden copiar, desplazar, renombrar o borrar archivos. Ejemplos de estos son: basename, cat, cmp, comm, cp, cut, dirname, file, mv, paste, rename, rm, sort, touch, etc.

Gestión de los permisos

Con comandos podemos definir los derechos de acceso y de posesión de archivos. Ejemplos de estos son: chattr, chgrp, chmod, chown, isattr.

Opciones de búsqueda

Linux dispone de diversos comandos para explorar el sistema desde la terminal. Ejemplos de estos son: find, grep, locate, uodatedb, etc.

Información sobre los usuarios

Podemos usar comandos para solicitar información detallada sobre los usuarios registrados en el sistema, así como de sus grupos y procesos. Ejemplos de estos son: finger, groups, ls, last, w, who, whoami.

Gestión de cuentas de usuario

Con comandos podemos crear, eliminar y gestionar cuentas de usuario y grupos. Ejemplos de estos son: adduser, chfn, chsh, deluser, groupadd, delgroup, groupmod, passwd, sudo, su, usermod, etc.

Gestión del sistema

Aquí se encuentran los comandos básicos de Linux para la administración del sistema, con ellos podemos apagar o reiniciar el sistema desde la consola. Ejemplos de estos son: logger, reboot, shutdown, etc.

Información del sistema

Aquí encontramos los comandos con los cuales se solicita información y mensajes de estado al sistema. Ejemplos de estos son: date, df, dmesg, du, free, hostname, uname, uptime, etc.

Información acerca del hardware

Estos comandos entregan datos sobre los componentes de nuestro hardware. Ejemplos de estos son: lscpu, lshw, lspci, lsusb.

Gestión de procesos

Estos comandos permiten supervisar los procesos. Ejemplos de estos son: chrt, ionice, kill, nice, etc.

Pager

Estos comandos permiten seleccionar las partes que se quieren mostrar en la terminal e incluso hojear archivos en el modo interactivo. Ejemplos de estos son: head, less, more, tail.

Editores

Con estos comandos podemos abrir editores de texto. Ejemplos de estos son: emacs, nano, vim.

Gestión de redes

Podemos examinar una conexión, solicitar información DNS, configurar interfaces o enviar archivos a otro ordenador con los comandos. Ejemplo de estos: arp, dig, ftp, ip, iw, netstat, nslookup, ping, route, traceroute, etc.

Archivar y comprimir

Con comandos podemos comprimir y empaquetar archivos. Ejemplos de estos son: tar, gzip, bzip2, gunzip, bunzip2, xz, etc.

Gestion de particiones

Con comandos podemos solicitar datos sobre dispositivos de bloque conectados y montarlos o desmontarlos. Ejemplos de estos son: mount, unmount, lsblk, blkid, dd.

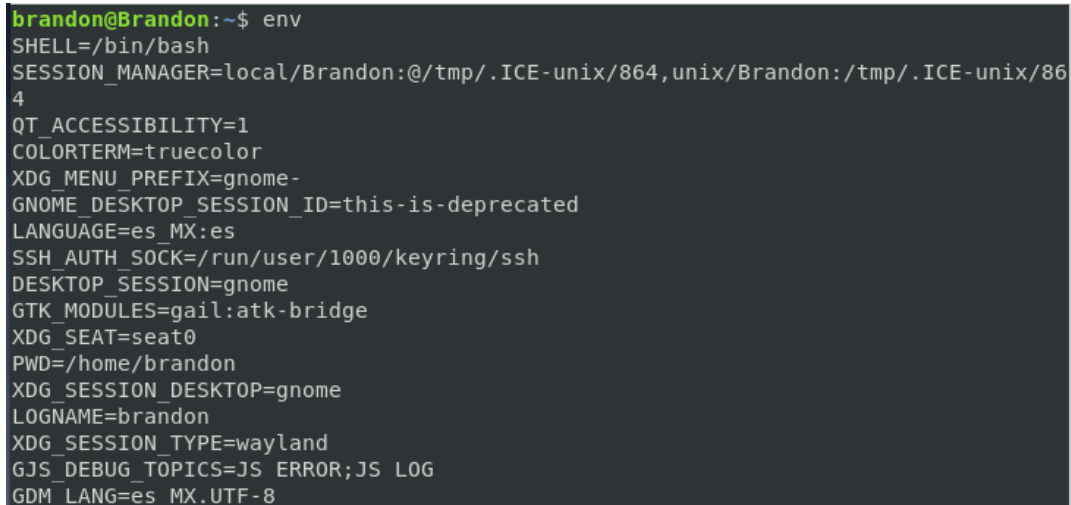
Variables de entorno

Una variable de entorno es un valor dinámico cargado en la memoria, estas se pueden crear, editar o eliminar. Son la forma más simple de pasar información de una aplicación a otra.

En Linux, una variable de entorno no es más que un nombre al que se asocia una cadena de caracteres. En bash declaramos una variable con el comando *export*.

Si queremos ver todas las variables de entorno disponibles para su uso en nuestro Linux, debemos escribir el comando: *env*

En la imagen 5 podemos ver algunas variables de entorno:



```
brandon@Brandon:~$ env
SHELL=/bin/bash
SESSION_MANAGER=local/Brandon:@/tmp/.ICE-unix/864,unix/Brandon:/tmp/.ICE-unix/864
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
LANGUAGE=es_MX:es
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
DESKTOP_SESSION=gnome
GTK_MODULES=gail:atk-bridge
XDG_SEAT=seat0
PWD=/home/brandon
XDG_SESSION_DESKTOP=gnome
LOGNAME=brandon
XDG_SESSION_TYPE=wayland
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
GDM_LANG=es_MX.UTF-8
```

Imagen 5. Algunas variables de entorno de Linux.

Se ve como el comando env muestra las variables de entorno que tenemos disponibles.

En Linux, tenemos unas variables comunes y se encuentran en la mayoría de distribuciones. Estas son las siguientes:

DISPLAY: Las salidas de X-Windows

HOME: Carpeta de usuario

HOSTNAME: Nombre del sistema

MAIL: Archivo de correo

PATH: Lista de directorios donde buscar las aplicaciones

PS1: Prompt

SHELL: Intérprete de comandos

TERM: Tipo de terminal

USER: Nombre del usuario

Desarrollo de la práctica

Comandos

cal: Se usa para mostrar el calendario actual. A continuación, una captura del funcionamiento del comando en la terminal.

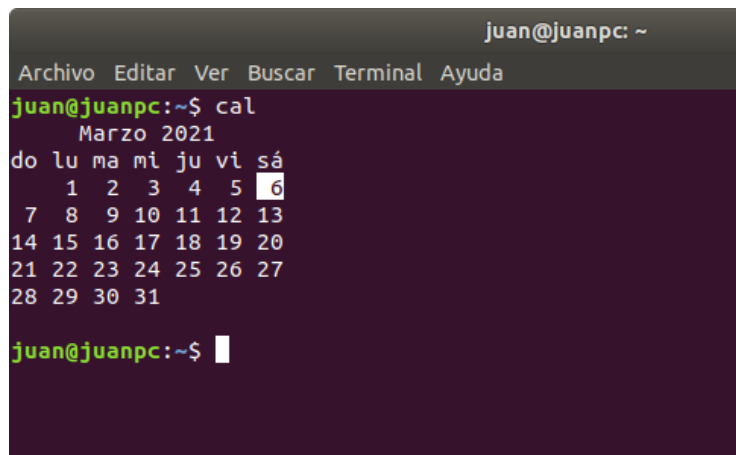
A terminal window titled 'juan@juanpc: ~' with a menu bar 'Archivo Editar Ver Buscar Terminal Ayuda'. The prompt is 'juan@juanpc:~\$'. The command 'cal' has been entered, and the output shows the calendar for March 2021. The days of the week are listed as 'do lu ma mi ju vi sa'. The numbers 1 through 31 are arranged in a grid. The number 6 is highlighted in the first row, under 'vi'. The prompt 'juan@juanpc:~\$' is shown again with a cursor.

Imagen 6. Comando cal en terminal Linux.

clear: Este comando se utiliza para limpiar el historial en la consola. A continuación, una captura del comando en la terminal.

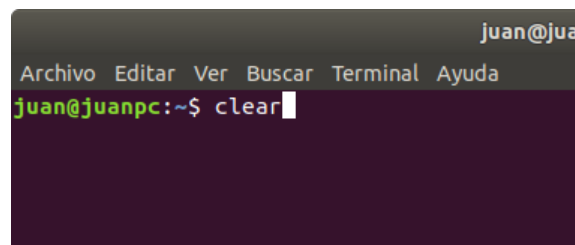
A terminal window titled 'juan@jua' with a menu bar 'Archivo Editar Ver Buscar Terminal Ayuda'. The prompt is 'juan@juanpc:~\$'. The command 'clear' has been entered, and the terminal screen is blank below it. The prompt 'juan@juanpc:~\$' is shown again with a cursor.

Imagen 7. Comando clear en terminal Linux

apt: Sirve para manejar paquetes del entorno Ubuntu, instalar, actualizar, etc. Requiere la instrucción `sudo` para confirmación del sistema. A continuación, una imagen del comando con la instrucción `update` en la terminal.

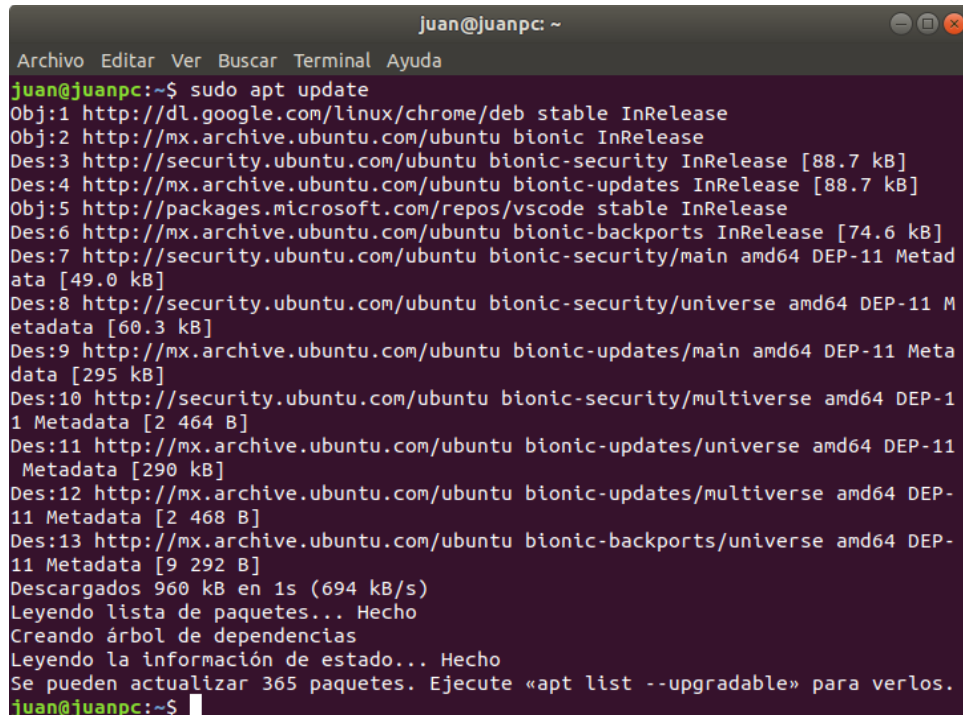
A terminal window titled 'juan@juanpc: ~' with a menu bar (Archivo, Editar, Ver, Buscar, Terminal, Ayuda). The command 'juan@juanpc:~\$ sudo apt update' has been executed. The output shows a list of update sources and their metadata sizes, followed by download statistics and a confirmation that 365 packages can be updated. The prompt returns to 'juan@juanpc:~\$'.

Imagen 8. Comando apt con la instrucción update en terminal Linux

rm: Es un comando que sirve para eliminar archivos y directorios del sistema de archivos. A continuación, una imagen con el comando en funcionamiento.

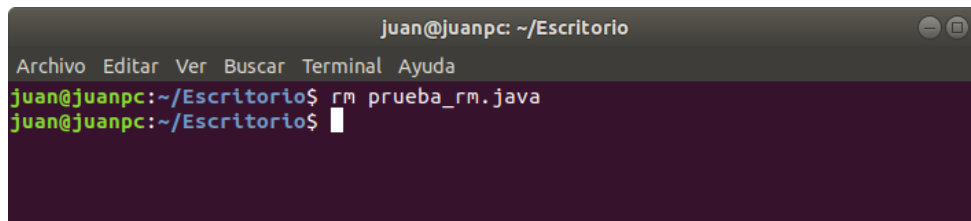
A terminal window titled 'juan@juanpc: ~/Escritorio' with a menu bar (Archivo, Editar, Ver, Buscar, Terminal, Ayuda). The command 'juan@juanpc:~/Escritorio\$ rm prueba_rm.java' has been executed. The prompt returns to 'juan@juanpc:~/Escritorio\$'.

Imagen 9. Comando rm en la terminal de Linux.

date: Es un comando que sirve para imprimir la fecha actual en distintos formatos. A continuación, una imagen del funcionamiento del comando.

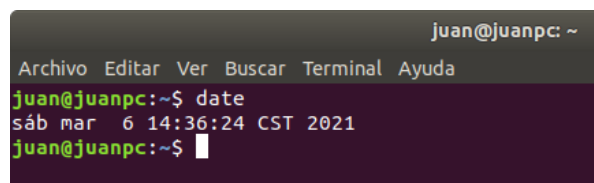
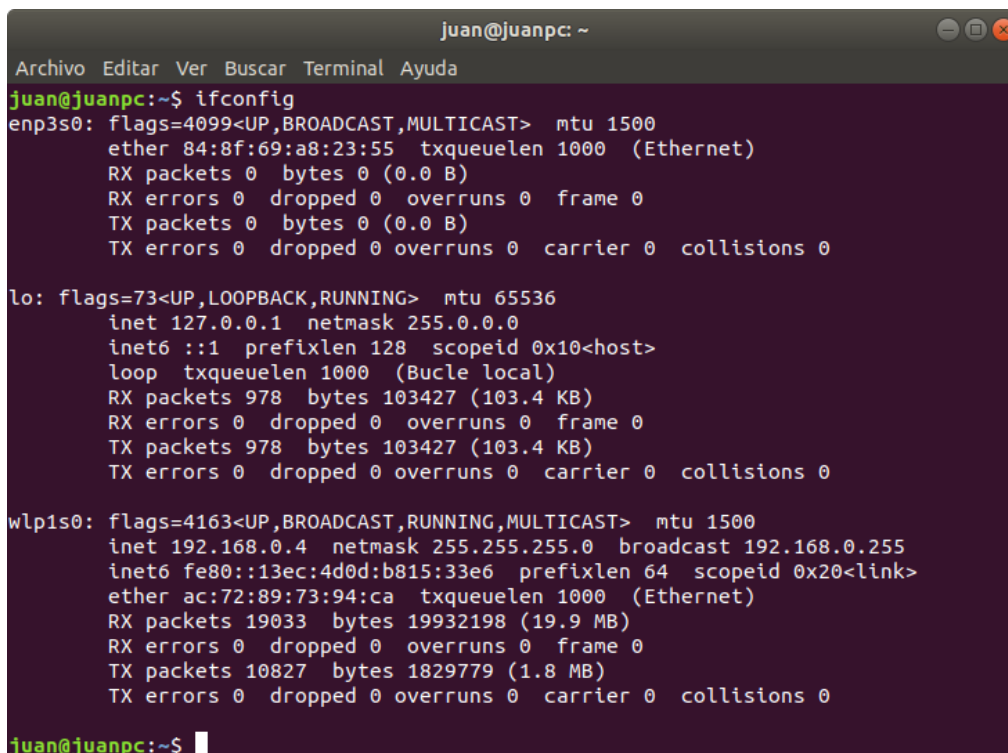
A terminal window titled 'Juan@juanpc: ~' with a menu bar (Archivo, Editar, Ver, Buscar, Terminal, Ayuda). The command 'juan@juanpc:~\$ date' has been executed. The output shows the current date and time: 'sáb mar 6 14:36:24 CST 2021'. The prompt returns to 'juan@juanpc:~\$'.

Imagen 10. Comando date en la terminal de Linux.

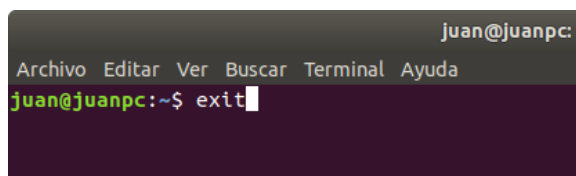
ifconfig: Es un comando que sirve para asignar direcciones IP a interfaces y configurar parámetros de interfaces manualmente. A continuación, una imagen de su funcionamiento en la terminal.



```
juan@juanpc: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
juan@juanpc:~$ ifconfig  
enp3s0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500  
    ether 84:8f:69:a8:23:55 txqueuelen 1000 (Ethernet)  
    RX packets 0 bytes 0 (0.0 B)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 0 bytes 0 (0.0 B)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
    inet 127.0.0.1 netmask 255.0.0.0  
    inet6 ::1 prefixlen 128 scopeid 0x10<host>  
    loop txqueuelen 1000 (Bucle local)  
    RX packets 978 bytes 103427 (103.4 KB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 978 bytes 103427 (103.4 KB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
wlp1s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.0.4 netmask 255.255.255.0 broadcast 192.168.0.255  
    inet6 fe80::13ec:4d0d:b815:33e6 prefixlen 64 scopeid 0x20<link>  
    ether ac:72:89:73:94:ca txqueuelen 1000 (Ethernet)  
    RX packets 19033 bytes 19932198 (19.9 MB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 10827 bytes 1829779 (1.8 MB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
juan@juanpc:~$
```

Imagen 11. Comando ifconfig en terminal Linux.

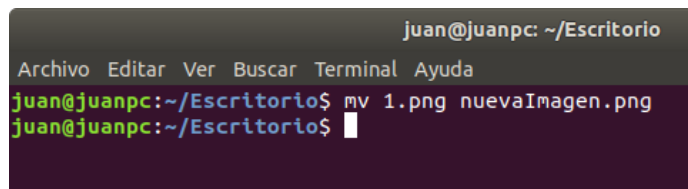
exit: Es un comando que sirve para cerrar las ventanas, las conexiones remotas o la pantalla del terminal. A continuación, una imagen del comando en la terminal.



```
juan@juanpc: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
juan@juanpc:~$ exit
```

Imagen 12. Comando exit en terminal Linux.

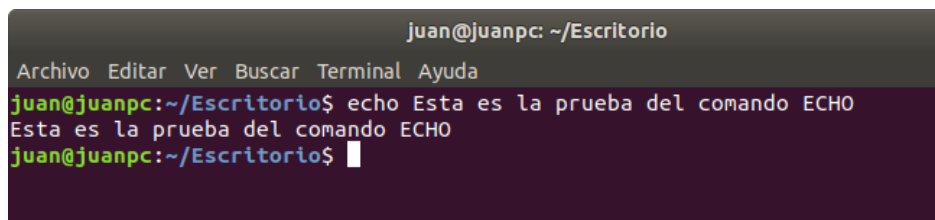
mv: Es un comando que se usa para mover o renombrar archivos o directorios del sistema de archivos. El archivo original es borrado y se crea un nuevo archivo con el mismo contenido, el nombre puede ser diferente o puede ser el mismo. A continuación, una imagen del funcionamiento del comando.



```
juan@juanpc: ~/Escritorio  
Archivo Editar Ver Buscar Terminal Ayuda  
juan@juanpc:~/Escritorio$ mv 1.png nuevaImagen.png  
juan@juanpc:~/Escritorio$
```

Imagen 13. Comando mv en la terminal de Linux.

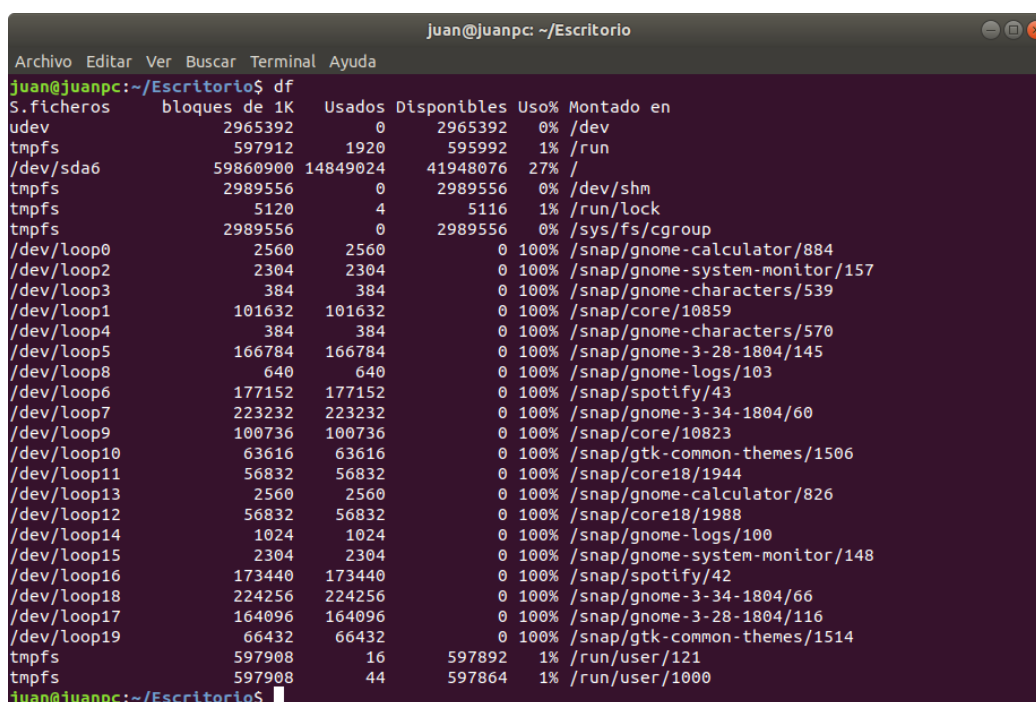
echo: Es un comando que sirve para imprimir un texto en pantalla. A continuación, una imagen del funcionamiento del comando en la terminal de Linux.

A terminal window titled 'juan@juanpc: ~/Escritorio' with a menu bar (Archivo, Editar, Ver, Buscar, Terminal, Ayuda). The prompt is 'juan@juanpc:~/Escritorio\$'. The command 'echo Esta es la prueba del comando ECHO' has been entered, and the output 'Esta es la prueba del comando ECHO' is displayed on the next line. The prompt is now 'juan@juanpc:~/Escritorio\$' with a cursor.

```
juan@juanpc: ~/Escritorio
Archivo Editar Ver Buscar Terminal Ayuda
juan@juanpc:~/Escritorio$ echo Esta es la prueba del comando ECHO
Esta es la prueba del comando ECHO
juan@juanpc:~/Escritorio$
```

Imagen 14. Comando echo en la terminal de Linux.

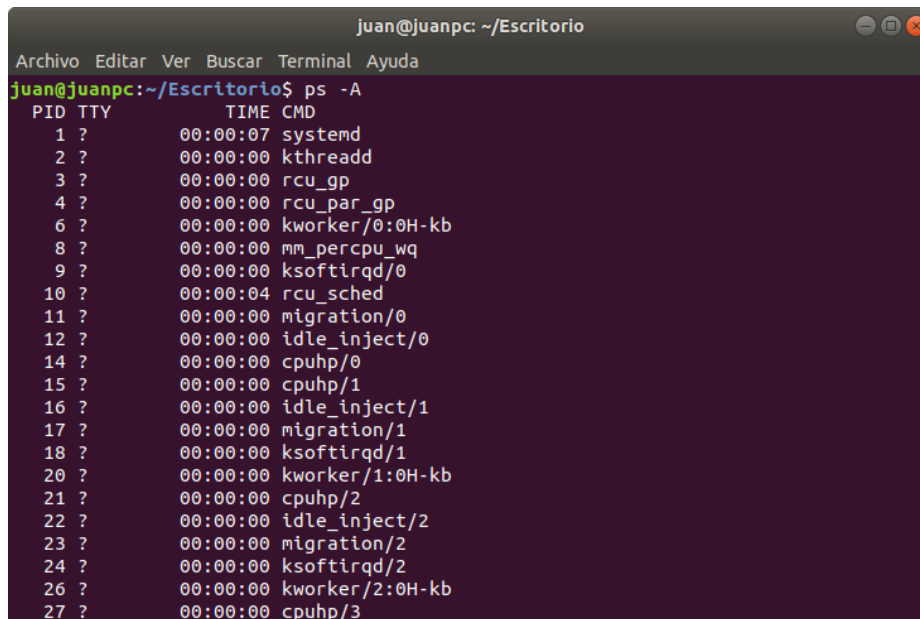
df: Este comando se utiliza para mostrar el espacio en disco utilizado por el sistema de ficheros. A continuación, una captura del funcionamiento del comando en la terminal.

A terminal window titled 'juan@juanpc: ~/Escritorio' with a menu bar (Archivo, Editar, Ver, Buscar, Terminal, Ayuda). The prompt is 'juan@juanpc:~/Escritorio\$'. The command 'df' has been entered, and the output is a table showing disk space usage for various filesystems. The prompt is now 'juan@juanpc:~/Escritorio\$' with a cursor.

```
juan@juanpc:~/Escritorio$ df
S.ficheros bloques de 1K Usados Disponibles Uso% Montado en
udev 2965392 0 2965392 0% /dev
tmpfs 597912 1920 595992 1% /run
/dev/sda6 59860900 14849024 41948076 27% /
tmpfs 2989556 0 2989556 0% /dev/shm
tmpfs 5120 4 5116 1% /run/lock
tmpfs 2989556 0 2989556 0% /sys/fs/cgroup
/dev/loop0 2560 2560 0 100% /snap/gnome-calculator/884
/dev/loop2 2304 2304 0 100% /snap/gnome-system-monitor/157
/dev/loop3 384 384 0 100% /snap/gnome-characters/539
/dev/loop1 101632 101632 0 100% /snap/core/10859
/dev/loop4 384 384 0 100% /snap/gnome-characters/570
/dev/loop5 166784 166784 0 100% /snap/gnome-3-28-1804/145
/dev/loop8 640 640 0 100% /snap/gnome-logs/103
/dev/loop6 177152 177152 0 100% /snap/spotify/43
/dev/loop7 223232 223232 0 100% /snap/gnome-3-34-1804/60
/dev/loop9 100736 100736 0 100% /snap/core/10823
/dev/loop10 63616 63616 0 100% /snap/gtk-common-themes/1506
/dev/loop11 56832 56832 0 100% /snap/core18/1944
/dev/loop13 2560 2560 0 100% /snap/gnome-calculator/826
/dev/loop12 56832 56832 0 100% /snap/core18/1988
/dev/loop14 1024 1024 0 100% /snap/gnome-logs/100
/dev/loop15 2304 2304 0 100% /snap/gnome-system-monitor/148
/dev/loop16 173440 173440 0 100% /snap/spotify/42
/dev/loop18 224256 224256 0 100% /snap/gnome-3-34-1804/66
/dev/loop17 164096 164096 0 100% /snap/gnome-3-28-1804/116
/dev/loop19 66432 66432 0 100% /snap/gtk-common-themes/1514
tmpfs 597908 16 597892 1% /run/user/121
tmpfs 597908 44 597864 1% /run/user/1000
juan@juanpc:~/Escritorio$
```

Imagen 15. Comando df en la terminal de Linux.

ps: Es un comando que se utiliza para conocer el estado de los procesos del sistema operativo, se pueden usar modificadores para conocer un proceso o procesos en específico. A continuación, una captura del funcionamiento del comando en la terminal.

A terminal window titled 'juan@juanpc: ~/Escritorio' showing the output of the 'ps -A' command. The output lists system processes with columns for PID, TTY, TIME, and CMD. Processes include systemd, kthreadd, rcu_gp, rcu_par_gp, kworker/0:0H-kb, mm_percpu_wq, ksoftirqd/0, rcu_sched, migration/0, idle_inject/0, cpuhp/0, cpuhp/1, idle_inject/1, migration/1, ksoftirqd/1, kworker/1:0H-kb, cpuhp/2, idle_inject/2, migration/2, ksoftirqd/2, kworker/2:0H-kb, and cpuhp/3.

```
juan@juanpc: ~/Escritorio
Archivo Editar Ver Buscar Terminal Ayuda
juan@juanpc:~/Escritorio$ ps -A
  PID TTY          TIME CMD
    1 ?            00:00:07 systemd
    2 ?            00:00:00 kthreadd
    3 ?            00:00:00 rcu_gp
    4 ?            00:00:00 rcu_par_gp
    6 ?            00:00:00 kworker/0:0H-kb
    8 ?            00:00:00 mm_percpu_wq
    9 ?            00:00:00 ksoftirqd/0
   10 ?            00:00:04 rcu_sched
   11 ?            00:00:00 migration/0
   12 ?            00:00:00 idle_inject/0
   14 ?            00:00:00 cpuhp/0
   15 ?            00:00:00 cpuhp/1
   16 ?            00:00:00 idle_inject/1
   17 ?            00:00:00 migration/1
   18 ?            00:00:00 ksoftirqd/1
   20 ?            00:00:00 kworker/1:0H-kb
   21 ?            00:00:00 cpuhp/2
   22 ?            00:00:00 idle_inject/2
   23 ?            00:00:00 migration/2
   24 ?            00:00:00 ksoftirqd/2
   26 ?            00:00:00 kworker/2:0H-kb
   27 ?            00:00:00 cpuhp/3
```

Imagen 16. Comando ps en la terminal de Linux.

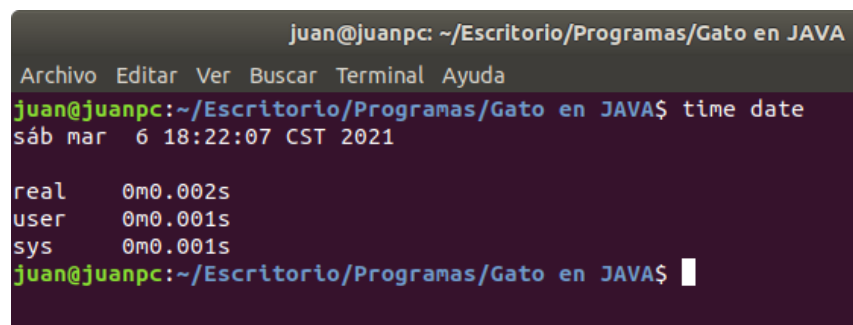
more: Es un comando para ver el contenido de un archivo o comando y visualizarlo, sin realizar modificación alguna. A continuación, una imagen del comando en la terminal.

A terminal window titled 'juan@juanpc: ~/Escritorio/Programas/Gato en JAVA' showing the output of the 'more' command. The command 'more' is used to view the contents of 'Coordenada.java'. The output shows the class definition for 'Coordenada' with attributes 'x' and 'y', a constructor, and getter methods.

```
juan@juanpc: ~/Escritorio/Programas/Gato en JAVA
Archivo Editar Ver Buscar Terminal Ayuda
juan@juanpc:~/Escritorio/Programas/Gato en JAVA$ more
Coordenada.class  Funciones.class  Gato.class        Jugador.java
Coordenada.java   Funciones.java   Gato.java
juan@juanpc:~/Escritorio/Programas/Gato en JAVA$ more Coordenada.java
public class Coordenada{
    int x,y;
    public Coordenada(int x,int y){
        this.x=x;
        this.y=y;
    }
    public int getX(){return x;}
    public int getY(){return y;}
}
juan@juanpc:~/Escritorio/Programas/Gato en JAVA$
```

Imagen 17. Comando more en la terminal Linux.

time: Es un comando que se utiliza para conocer el tiempo que dura otro comando. A continuación, una imagen que muestra el funcionamiento del comando en la terminal.

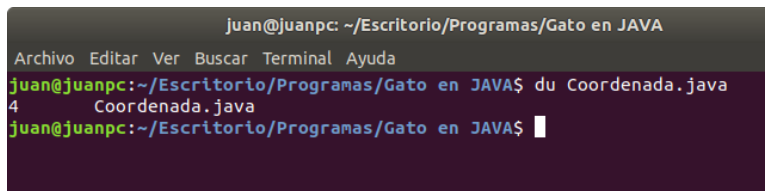
A terminal window titled 'juan@juanpc: ~/Escritorio/Programas/Gato en JAVA' showing the output of the 'time' command. The command 'time date' is used to measure the execution time of the 'date' command. The output shows the current date and time, followed by the execution time in real, user, and system time.

```
juan@juanpc: ~/Escritorio/Programas/Gato en JAVA
Archivo Editar Ver Buscar Terminal Ayuda
juan@juanpc:~/Escritorio/Programas/Gato en JAVA$ time date
sáb mar  6 18:22:07 CST 2021

real    0m0.002s
user    0m0.001s
sys     0m0.001s
juan@juanpc:~/Escritorio/Programas/Gato en JAVA$
```

Imagen 18. Comando time en la terminal Linux.

du: Es un comando que se utiliza para estimar el espacio en disco de un archivo o directorio de archivos. A continuación, una imagen que muestra el funcionamiento del comando en la terminal.

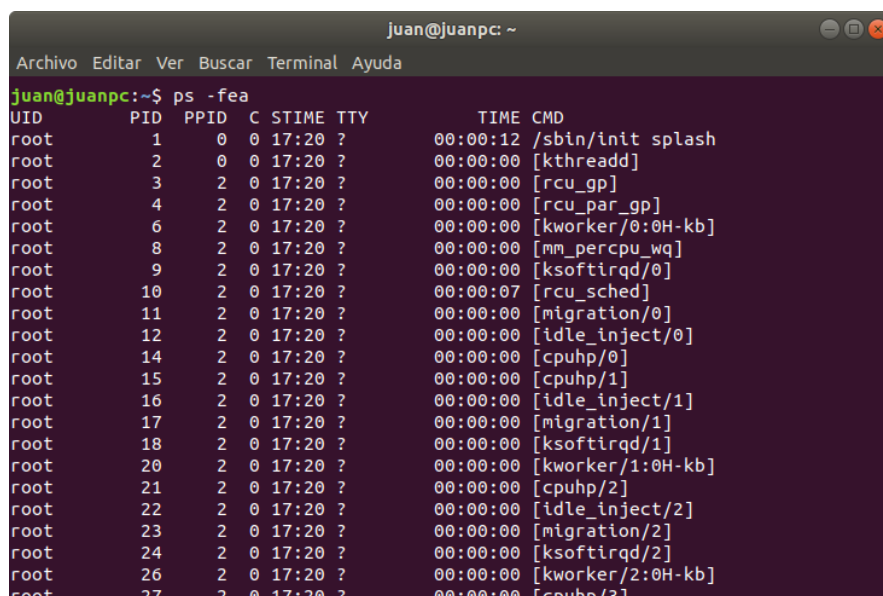


```

Juan@juanpc: ~/Escritorio/Programas/Gato en JAVA
Archivo Editar Ver Buscar Terminal Ayuda
Juan@juanpc:~/Escritorio/Programas/Gato en JAVA$ du Coordenada.java
4      Coordenada.java
Juan@juanpc:~/Escritorio/Programas/Gato en JAVA$
```

Imagen 19. Comando du en la terminal Linux.

ps -fea: Es una variación del comando ps, con los modificadores -f -e -a, que sería mostrar todos los procesos actuales de los usuarios activos junto con los parámetros con los que se inició el proceso. A continuación, una imagen que muestra el funcionamiento del comando en la terminal.

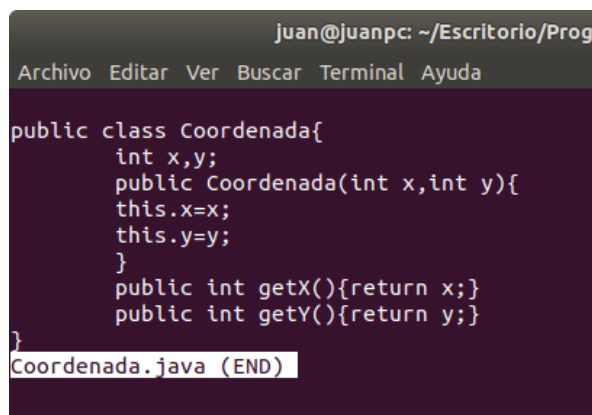


```

Juan@juanpc: ~
Archivo Editar Ver Buscar Terminal Ayuda
Juan@juanpc:~$ ps -fea
UID          PID  PPID  C  STIME TTY          TIME CMD
root           1      0  0  17:20 ?        00:00:12 /sbin/init splash
root           2      0  0  17:20 ?        00:00:00 [kthreadd]
root           3      2  0  17:20 ?        00:00:00 [rcu_gp]
root           4      2  0  17:20 ?        00:00:00 [rcu_par_gp]
root           6      2  0  17:20 ?        00:00:00 [kworker/0:0H-kb]
root           8      2  0  17:20 ?        00:00:00 [mm_percpu_wq]
root           9      2  0  17:20 ?        00:00:00 [ksoftirqd/0]
root          10      2  0  17:20 ?        00:00:07 [rcu_sched]
root          11      2  0  17:20 ?        00:00:00 [migration/0]
root          12      2  0  17:20 ?        00:00:00 [idle_inject/0]
root          14      2  0  17:20 ?        00:00:00 [cpuhp/0]
root          15      2  0  17:20 ?        00:00:00 [cpuhp/1]
root          16      2  0  17:20 ?        00:00:00 [idle_inject/1]
root          17      2  0  17:20 ?        00:00:00 [migration/1]
root          18      2  0  17:20 ?        00:00:00 [ksoftirqd/1]
root          20      2  0  17:20 ?        00:00:00 [kworker/1:0H-kb]
root          21      2  0  17:20 ?        00:00:00 [cpuhp/2]
root          22      2  0  17:20 ?        00:00:00 [idle_inject/2]
root          23      2  0  17:20 ?        00:00:00 [migration/2]
root          24      2  0  17:20 ?        00:00:00 [ksoftirqd/2]
root          26      2  0  17:20 ?        00:00:00 [kworker/2:0H-kb]
root          27      2  0  17:20 ?        00:00:00 [cpuhp/3]
```

Imagen 20. Comando ps -fea en la terminal Linux.

less: Es un comando que se utiliza para visualizar un archivo y poder modificarlo. A continuación, una imagen que muestra el funcionamiento del comando en la terminal.



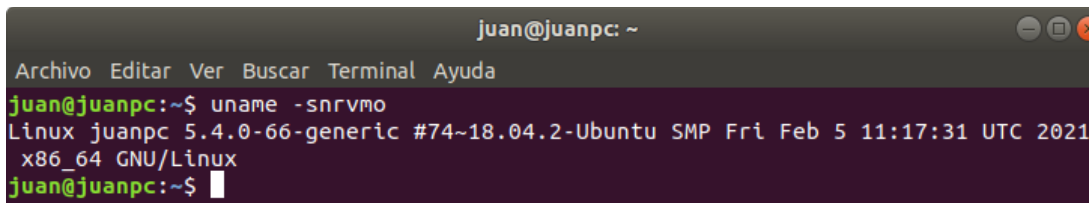
```

Juan@juanpc: ~/Escritorio/Programas/Gato en JAVA
Archivo Editar Ver Buscar Terminal Ayuda

public class Coordenada{
    int x,y;
    public Coordenada(int x,int y){
        this.x=x;
        this.y=y;
    }
    public int getX(){return x;}
    public int getY(){return y;}
}
Coordenada.java (END)
```

Imagen 21. Comando less en la terminal Linux.

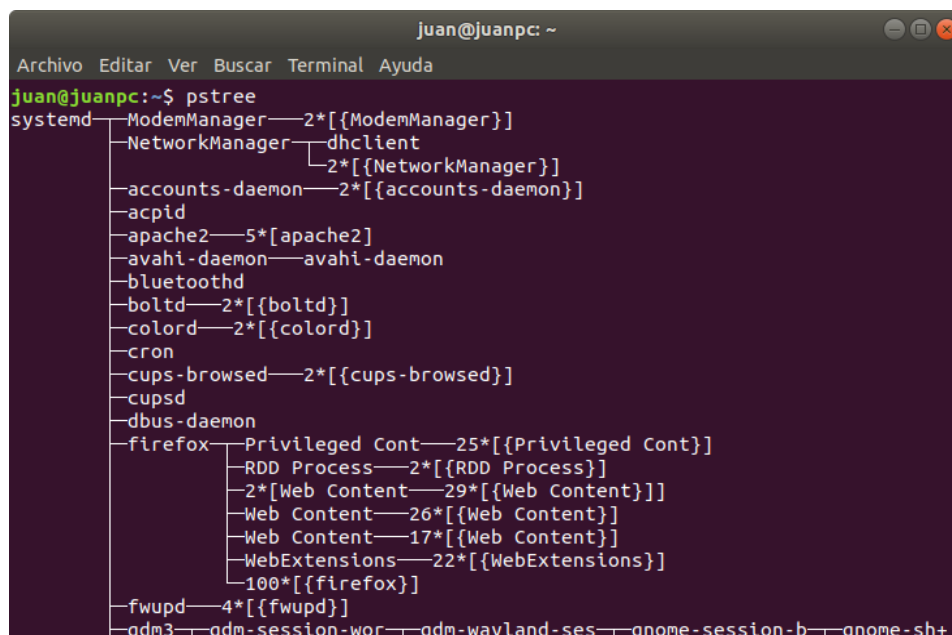
uname: Es un comando que se utiliza para mostrar información del sistema operativo como la versión del mismo, kernel y detalles del equipo. A continuación, una imagen que muestra el funcionamiento del comando en la terminal.



```
juan@juanpc: ~
Archivo Editar Ver Buscar Terminal Ayuda
juan@juanpc:~$ uname -srmvmo
Linux juanpc 5.4.0-66-generic #74~18.04.2-Ubuntu SMP Fri Feb 5 11:17:31 UTC 2021
x86_64 GNU/Linux
juan@juanpc:~$
```

Imagen 22. Comando uname en la terminal Linux.

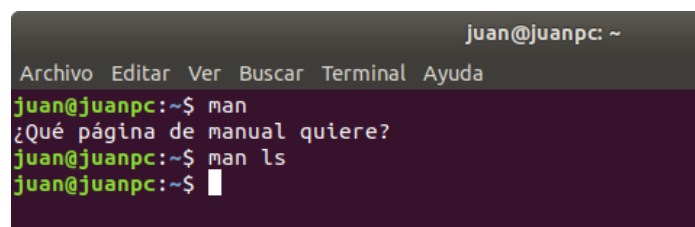
ps tree: Es un comando que se utiliza para mostrar procesos en ejecución en Linux. A continuación, una imagen que muestra el funcionamiento del comando en la terminal.



```
juan@juanpc: ~
Archivo Editar Ver Buscar Terminal Ayuda
juan@juanpc:~$ ps tree
systemd--ModemManager--2*[{ModemManager}]
      |NetworkManager--dhclient
      |                |2*[{NetworkManager}]
accounts-daemon--2*[{accounts-daemon}]
acpid
apache2--5*[{apache2}]
avahi-daemon--avahi-daemon
bluetoothd
boltd--2*[{boltd}]
colord--2*[{colord}]
cron
cups-browsed--2*[{cups-browsed}]
cupsd
dbus-daemon
firefox--Privileged Cont--25*[{Privileged Cont}]
      |RDD Process--2*[{RDD Process}]
      |2*[{Web Content--29*[{Web Content}]]
      |Web Content--26*[{Web Content}]
      |Web Content--17*[{Web Content}]
      |WebExtensions--22*[{WebExtensions}]
      |100*[{firefox}]
fwupd--4*[{fwupd}]
qdm3--qdm-session-wor--qdm-wayland-ses--gnome-session-b--gnome-sh+
```

Imagen 23. Comando ps tree en la terminal Linux.

man: Es un comando que se utiliza para acceder a la documentación disponible de sus herramientas y así aprender sobre comandos, archivos, llamadas de sistema, etc, en un sistema operativo tal y como GNU/Linux. A continuación, una imagen que muestra el funcionamiento del comando en la terminal.



```
juan@juanpc: ~
Archivo Editar Ver Buscar Terminal Ayuda
juan@juanpc:~$ man
¿Qué página de manual quiere?
juan@juanpc:~$ man ls
juan@juanpc:~$
```

Imagen 24. Comando man en la terminal Linux.

mkdir: Es un comando que se utiliza para crear directorios. A continuación, una imagen que muestra el funcionamiento del comando en la terminal.

```
juan@juanpc: ~/Escritorio/ImagenesLinux
Archivo Editar Ver Buscar Terminal Ayuda
juan@juanpc:~/Escritorio$ mkdir ImagenesLinux
juan@juanpc:~/Escritorio$ cd ImagenesLinux/
juan@juanpc:~/Escritorio/ImagenesLinux$
```

Imagen 25. Comando mkdir en la terminal Linux.

w: Es un comando que se utiliza para mostrar información sobre los usuarios actualmente conectados y lo que está haciendo cada usuario. A continuación, una imagen que muestra el funcionamiento del comando en la terminal.

```
juan@juanpc: ~
Archivo Editar Ver Buscar Terminal Ayuda
juan@juanpc:~$ w
19:25:49 up 2:04, 1 user, load average: 0.97, 0.72, 0.76
USUARIO TTY DE LOGIN@ IDLE JCPU PCPU WHAT
juan :0 :0 17:22 ?xdm? 2:08 0.01s /usr/lib/gdm3/g
juan@juanpc:~$
```

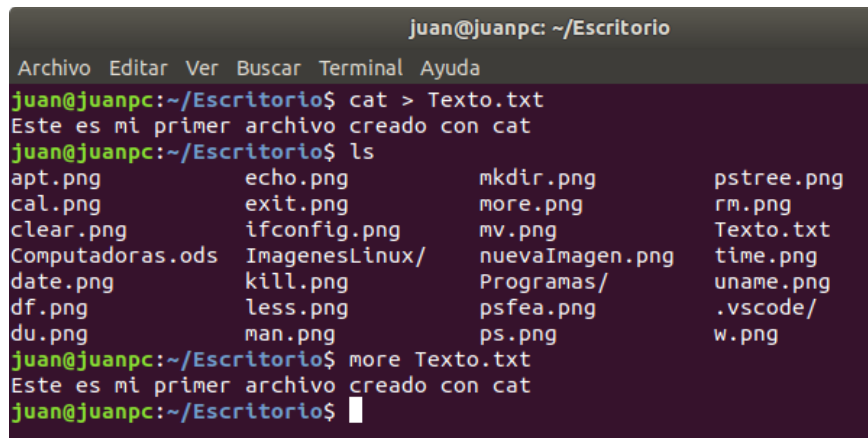
Imagen 26. Comando w en la terminal Linux.

kill -l -9: Es un comando que se utiliza para matar un proceso, el modificador -l escribe todos los valores de señal soportados y -9 fuerza a matar un proceso. A continuación, una imagen que muestra el funcionamiento del comando en la terminal.

```
juan@juanpc: ~
Archivo Editar Ver Buscar Terminal Ayuda
juan@juanpc:~$ kill
kill: modo de empleo: kill [-s id_señal | -n num_señal | -id_señal] pid | idtrab
ajo ... o kill -l [id_señal]
juan@juanpc:~$ kill -l
1) SIGHUP          2) SIGINT          3) SIGQUIT         4) SIGILL          5) SIGTRAP
6) SIGABRT         7) SIGBUS         8) SIGFPE          9) SIGKILL         10) SIGUSR1
11) SIGSEGV        12) SIGUSR2        13) SIGPIPE        14) SIGALRM         15) SIGTERM
16) SIGSTKFLT      17) SIGCHLD        18) SIGCONT        19) SIGSTOP         20) SIGTSTP
21) SIGTTIN        22) SIGTTOU        23) SIGURG         24) SIGXCPU         25) SIGXFSZ
26) SIGVTALRM      27) SIGPROF        28) SIGWINCH       29) SIGIO           30) SIGPWR
31) SIGSYS         34) SIGRTMIN       35) SIGRTMIN+1     36) SIGRTMIN+2     37) SIGRTMIN+3
38) SIGRTMIN+4     39) SIGRTMIN+5     40) SIGRTMIN+6     41) SIGRTMIN+7     42) SIGRTMIN+8
43) SIGRTMIN+9     44) SIGRTMIN+10    45) SIGRTMIN+11    46) SIGRTMIN+12    47) SIGRTMIN+13
48) SIGRTMIN+14    49) SIGRTMIN+15    50) SIGRTMAX-14    51) SIGRTMAX-13    52) SIGRTMAX-12
53) SIGRTMAX-11    54) SIGRTMAX-10    55) SIGRTMAX-9     56) SIGRTMAX-8     57) SIGRTMAX-7
58) SIGRTMAX-6     59) SIGRTMAX-5     60) SIGRTMAX-4     61) SIGRTMAX-3     62) SIGRTMAX-2
63) SIGRTMAX-1     64) SIGRTMAX
juan@juanpc:~$
```

Imagen 27. Comando kill -l -9 en la terminal Linux.

cat: Es un comando que permite crear, fusionar o imprimir archivos en la pantalla de salida estándar o en otro archivo además de otras operaciones. A continuación, una imagen que muestra el funcionamiento del comando en la terminal.



```
juan@juanpc: ~/Escritorio
Archivo Editar Ver Buscar Terminal Ayuda
juan@juanpc:~/Escritorio$ cat > Texto.txt
Este es mi primer archivo creado con cat
juan@juanpc:~/Escritorio$ ls
apt.png          echo.png         mkdir.png        pstree.png
cal.png          exit.png         more.png         rm.png
clear.png        ifconfig.png     mv.png          Texto.txt
Computadoras.ods ImagenesLinux/   nuevaImagen.png time.png
date.png         kill.png         Programas/       uname.png
df.png           less.png         psfea.png        .vscode/
du.png           man.png          ps.png           w.png
juan@juanpc:~/Escritorio$ more Texto.txt
Este es mi primer archivo creado con cat
juan@juanpc:~/Escritorio$
```

Imagen 28. Comando cat en la terminal Linux.

pico: Es un comando que se utiliza para abrir un editor de texto bastante sencillo. A continuación, una imagen que muestra el funcionamiento del comando en la terminal.

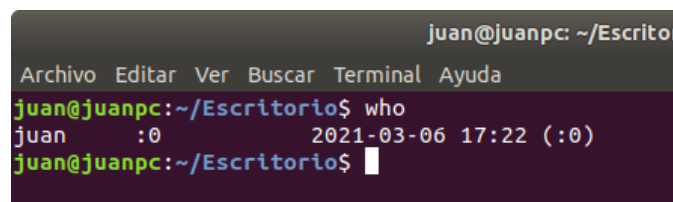


```
juan@juanpc: ~/Escritorio
Archivo Editar Ver Buscar Terminal Ayuda
GNU nano 2.9.3 pico.txt
Este es el edir de archivos pico

[ 1 línea leída ]
^G Ver ayuda ^O Guardar ^W Buscar ^K Cortar Tex ^J Justificar ^C Posición
^X Salir ^R Leer fich.^_ Reemplazar ^U Pegar txt ^T Ortografía ^_ Ir a línea
```

Imagen 29. Comando pico en la terminal Linux.

who: Es un comando que se utiliza para aportar datos sobre los usuarios del sistema. A continuación, una imagen que muestra el funcionamiento del comando en la terminal.



```
juan@juanpc: ~/Escritorio
Archivo Editar Ver Buscar Terminal Ayuda
juan@juanpc:~/Escritorio$ who
juan      :0                2021-03-06 17:22 (:0)
juan@juanpc:~/Escritorio$
```

Imagen 30. Comando who en la terminal Linux.

trap -l: Es un comando que se utiliza para especificar las acciones a realizar cuando se reciban las señales, además se usa el modificador -l para ver los números de señal y los nombres asociados en un incitador de comandos. A continuación, una imagen que muestra el funcionamiento del comando en la terminal.

```

juan@juanpc: ~/Escritorio
Archivo Editar Ver Buscar Terminal Ayuda
juan@juanpc:~/Escritorio$ trap -l
1) SIGHUP      2) SIGINT      3) SIGQUIT     4) SIGILL      5) SIGTRAP
6) SIGABRT     7) SIGBUS     8) SIGFPE      9) SIGKILL     10) SIGUSR1
11) SIGSEGV    12) SIGUSR2    13) SIGPIPE    14) SIGALRM     15) SIGTERM
16) SIGSTKFLT  17) SIGCHLD    18) SIGCONT     19) SIGSTOP     20) SIGSTP
21) SIGTTIN    22) SIGTTOU    23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM  27) SIGPROF    28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS     34) SIGRTMIN   35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4 39) SIGRTMIN+5 40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9 44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6 59) SIGRTMAX-5 60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1 64) SIGRTMAX
juan@juanpc:~/Escritorio$

```

Imagen 31. Comando trap -l en la terminal Linux.

fg: Es un comando que se utiliza para traer a primer plano un trabajo que está ejecutándose en segundo plano. También se puede usar para reanudar en primer plano un trabajo que está suspendido o detenido. A continuación, una imagen que muestra el funcionamiento del comando en la terminal.

```

juan@juanpc: ~/Escritorio
Archivo Editar Ver Buscar Terminal Ayuda
juan@juanpc:~/Escritorio$ fg %1

```

Imagen 32. Comando fg en la terminal Linux.

nano: Es un comando que se utiliza para abrir un editor de texto que está basado en pico. A continuación, una imagen que muestra el funcionamiento del comando en la terminal.

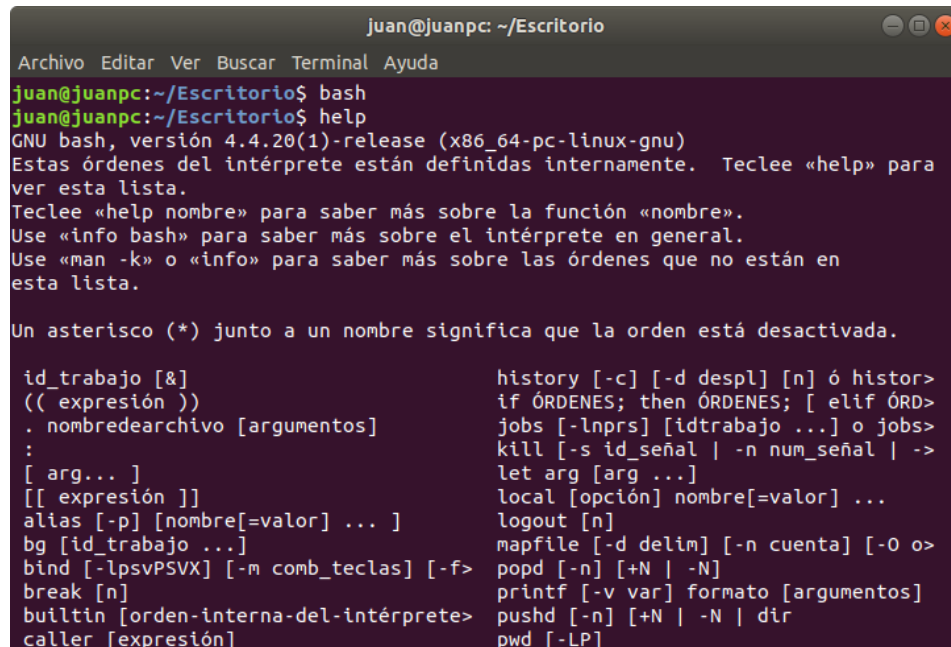
```

juan@juanpc: ~/Escritorio
Archivo Editar Ver Buscar Terminal Ayuda
GNU nano 2.9.3          Nuevo búfer          Modificado
Hola desde nano
^G Ver ayuda  ^O Guardar  ^M Buscar   ^K Cortar Tex^J Justificar^C Posición
^X Salir      ^R Leer fich.^_ Reemplazar^U Pegar txt ^T Ortografía^_ Ir a línea

```

Imagen 33. Comando nano en la terminal Linux.

bash: Son un conjunto de comandos que se utilizan para para la administración y configuración del sistema, así como un conjunto de combinaciones especiales de teclas para realizar tareas específicas en entornos Linux/Unix. A continuación, una imagen que muestra el funcionamiento del comando en la terminal.



```

juan@juanpc: ~/Escritorio
Archivo Editar Ver Buscar Terminal Ayuda
juan@juanpc:~/Escritorio$ bash
juan@juanpc:~/Escritorio$ help
GNU bash, versión 4.4.20(1)-release (x86_64-pc-linux-gnu)
Estas órdenes del intérprete están definidas internamente.  Teclee «help» para
ver esta lista.
Teclee «help nombre» para saber más sobre la función «nombre».
Use «info bash» para saber más sobre el intérprete en general.
Use «man -k» o «info» para saber más sobre las órdenes que no están en
esta lista.

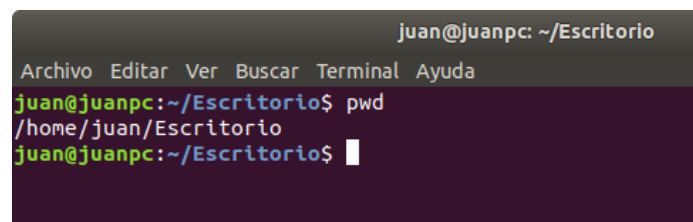
Un asterisco (*) junto a un nombre significa que la orden está desactivada.

id_trabajo [&]
(( expresión ))
. nombredearchivo [argumentos]
:
[ arg... ]
[[ expresión ]]
alias [-p] [nombre[=valor] ... ]
bg [id_trabajo ...]
bind [-lpsvPSVX] [-m comb_teclas] [-f>
break [n]
builtin [orden-interna-del-intérprete>
caller [expresión]
history [-c] [-d despl] [n] ó histor>
if ÓRDENES; then ÓRDENES; [ elif ÓRD>
jobs [-lnprs] [idtrabajo ...] o jobs>
kill [-s id_señal | -n num_señal | ->
let arg [arg ...]
local [opción] nombre[=valor] ...
logout [n]
mapfile [-d delim] [-n cuenta] [-O o>
popd [-n] [+N | -N]
printf [-v var] formato [argumentos]
pushd [-n] [+N | -N | dir
pwd [-LP]

```

Imagen 34. Comando bash en la terminal Linux.

pwd: Es un comando que se utiliza para imprimir el nombre del directorio actual en una sesión de comandos bajo un sistema operativo Unix o derivado. A continuación, una imagen que muestra el funcionamiento del comando en la terminal.



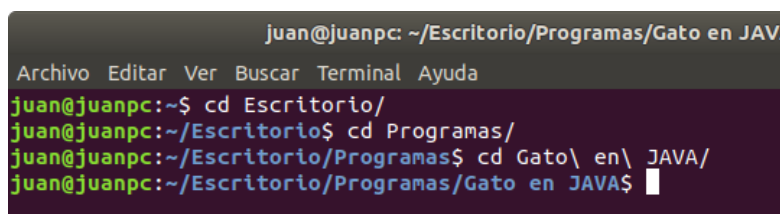
```

juan@juanpc: ~/Escritorio
Archivo Editar Ver Buscar Terminal Ayuda
juan@juanpc:~/Escritorio$ pwd
/home/juan/Escritorio
juan@juanpc:~/Escritorio$

```

Imagen 35. Comando pwd en la terminal Linux.

cd: Es un comando que se utiliza para moverse a través de los directorios. A continuación, una imagen que muestra el funcionamiento del comando en la terminal.



```

juan@juanpc: ~/Escritorio/Programas/Gato en JAVA
Archivo Editar Ver Buscar Terminal Ayuda
juan@juanpc:~$ cd Escritorio/
juan@juanpc:~/Escritorio$ cd Programas/
juan@juanpc:~/Escritorio/Programas$ cd Gato\ en\ JAVA/
juan@juanpc:~/Escritorio/Programas/Gato en JAVA$

```

Imagen 36. Comando cd en la terminal Linux.

vi: Es un comando que se utiliza para abrir un editor de texto que guarda los archivos en memoria. A continuación, una imagen que muestra el funcionamiento del comando en la terminal.



Imagen 37. Comando vi en la terminal Linux.

wc: Es un comando que se utiliza para realizar diferentes conteos desde la entrada estándar, ya sea de palabras, caracteres o saltos de líneas. A continuación, una imagen que muestra el funcionamiento del comando en la terminal.

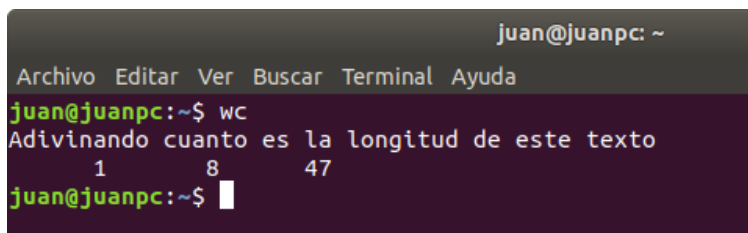


Imagen 38. Comando wc en la terminal Linux.

su: Es un comando es una utilidad de los sistemas operativos del tipo Unix que permite usar el intérprete de comandos de otro usuario sin necesidad de cerrar la sesión. A continuación, una imagen que muestra el funcionamiento del comando en la terminal.

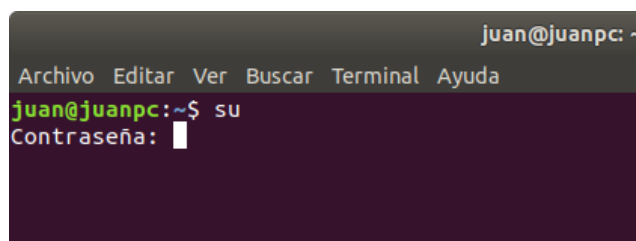


Imagen 39. Comando su en la terminal Linux.

ls: Es un comando que se utiliza para listar los archivos contenidos en un fichero. A continuación, una imagen que muestra el funcionamiento del comando en la terminal.

```
juan@juanpc: ~/Escritorio
Archivo Editar Ver Buscar Terminal Ayuda
juan@juanpc:~/Escritorio$ ls
apt.png      df.png      less.png     pico.txt     Texto.txt
bash.png     du.png      man.png      Programas    time.png
cal.png      echo.png    mkdir.png    psfea.png    trapl.png
cat.png      exit.png    more.png     ps.png       uname.png
cd.png       fg.png      mv.png       pstree.png   vi.png
clear.png    ifconfig.png nano.png      pwd.png      wc.png
Computadoras.ods ImagenesLinux nuevaImagen.png rm.png       who.png
date.png     kill.png    pico.png     su.png       w.png
juan@juanpc:~/Escritorio$
```

Imagen 40. Comando ls en la terminal Linux.

apt-get: Es un comando bastante robusto que se utiliza para descargar archivos de internet e instalarlos. A continuación, una imagen que muestra el funcionamiento del comando en la terminal.

```
juan@juanpc: ~/Escritorio
Archivo Editar Ver Buscar Terminal Ayuda
juan@juanpc:~/Escritorio$ apt-get
apt 1.6.12ubuntu0.2 (amd64)
Uso: apt-get [opciones] orden
      apt-get [opciones] install|remove paq1 [paq2 ...]
      apt-get [opciones] source paq1 [paq2 ...]

apt-get es una interfaz de línea de órdenes para la obtención de
paquetes y de información sobre ellos de orígenes autenticados y
para la instalación, actualización y eliminación de paquetes junto
a sus dependencias.

Órdenes más utilizadas:
update - Descarga nuevas listas de paquetes
upgrade - Realiza una actualización
install - Instala nuevos paquetes (paquete es libc6 y no libc6.deb)
remove - Elimina paquetes
purge - Elimina y purga paquetes
autoremove - Elimina automáticamente todos los paquetes sin utilizar
dist-upgrade - Actualiza la distribución, vea apt-get(8)
dselect-upgrade - Sigue las selecciones de dselect
build-dep - Configura las dependencias de construcción para paquetes fuente
clean - Elimina los archivos descargados
autoclean - Elimina los archivos descargados antiguos
check - Verifica que no haya dependencias incumplidas
```

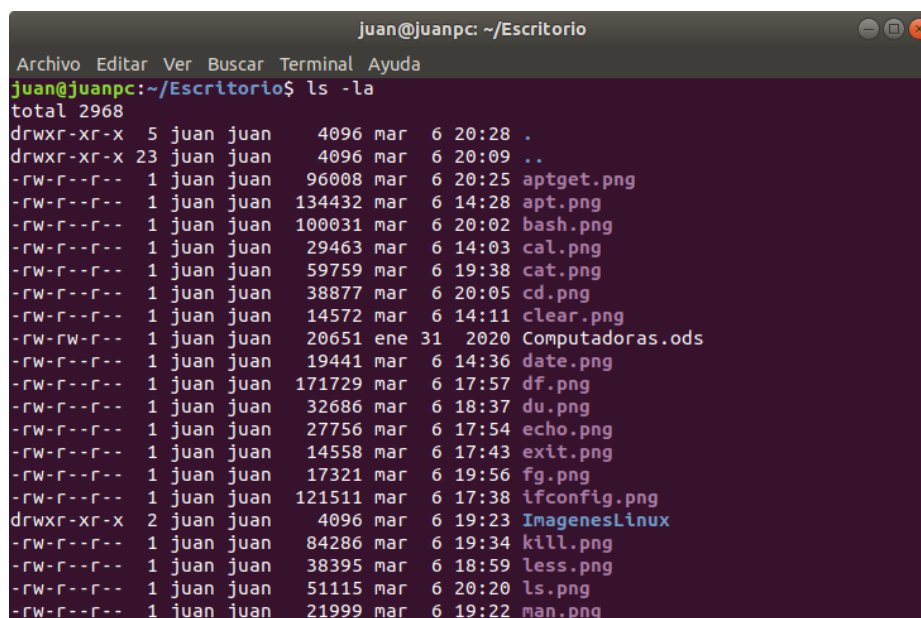
Imagen 41. Comando apt-get en la terminal Linux.

sudo: Es una utilidad de los sistemas operativos tipo Unix, como GNU/Linux, BSD, Mac OS X o Mac OS 11, que permite a los usuarios ejecutar programas con los privilegios de seguridad de otro usuario normalmente el usuario root de manera segura. A continuación, una imagen que muestra el funcionamiento del comando en la terminal.

```
juan@juanpc: ~/Escritorio
Archivo Editar Ver Buscar Terminal Ayuda
juan@juanpc:~/Escritorio$ sudo
usage: sudo -h | -K | -k | -V
usage: sudo -v [-AknS] [-g group] [-h host] [-p prompt] [-u user]
usage: sudo -l [-AknS] [-g group] [-h host] [-p prompt] [-U user] [-u user]
[command]
usage: sudo [-AbEHknPS] [-r role] [-t type] [-C num] [-g group] [-h host] [-p
prompt] [-T timeout] [-u user] [VAR=value] [-i|-s] [<command>]
usage: sudo -e [-AknS] [-r role] [-t type] [-C num] [-g group] [-h host] [-p
prompt] [-T timeout] [-u user] file ...
juan@juanpc:~/Escritorio$
```

Imagen 42. Comando sudo en la terminal Linux.

ls -la: Es un comando que se utiliza para listar los archivos de un directorio, pero con una descripción más detallada. A continuación, una imagen que muestra el funcionamiento del comando en la terminal.

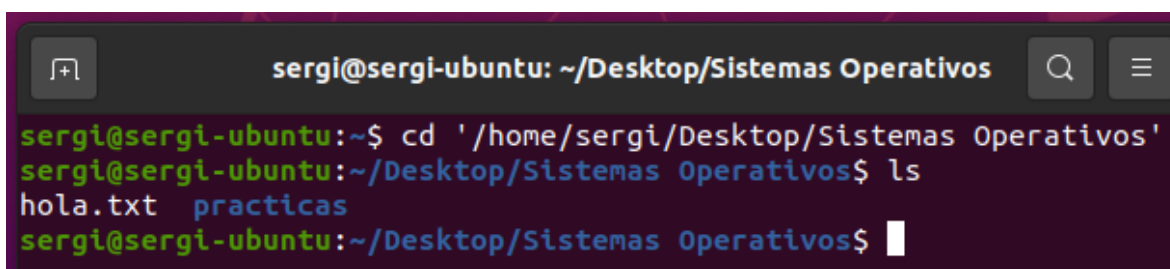


```
juan@juanpc: ~/Escritorio
Archivo Editar Ver Buscar Terminal Ayuda
juan@juanpc:~/Escritorio$ ls -la
total 2968
drwxr-xr-x  5 juan juan   4096 mar  6 20:28 .
drwxr-xr-x 23 juan juan   4096 mar  6 20:09 ..
-rw-r--r--  1 juan juan  96008 mar  6 20:25 aptget.png
-rw-r--r--  1 juan juan 134432 mar  6 14:28 apt.png
-rw-r--r--  1 juan juan 100031 mar  6 20:02 bash.png
-rw-r--r--  1 juan juan  29463 mar  6 14:03 cal.png
-rw-r--r--  1 juan juan  59759 mar  6 19:38 cat.png
-rw-r--r--  1 juan juan  38877 mar  6 20:05 cd.png
-rw-r--r--  1 juan juan  14572 mar  6 14:11 clear.png
-rw-rw-r--  1 juan juan  20651 ene 31 2020 Computadoras.ods
-rw-r--r--  1 juan juan  19441 mar  6 14:36 date.png
-rw-r--r--  1 juan juan 171729 mar  6 17:57 df.png
-rw-r--r--  1 juan juan  32686 mar  6 18:37 du.png
-rw-r--r--  1 juan juan  27756 mar  6 17:54 echo.png
-rw-r--r--  1 juan juan  14558 mar  6 17:43 exit.png
-rw-r--r--  1 juan juan  17321 mar  6 19:56 fg.png
-rw-r--r--  1 juan juan 121511 mar  6 17:38 ifconfig.png
drwxr-xr-x  2 juan juan   4096 mar  6 19:23 ImagenesLinux
-rw-r--r--  1 juan juan  84286 mar  6 19:34 kill.png
-rw-r--r--  1 juan juan  38395 mar  6 18:59 less.png
-rw-r--r--  1 juan juan  51115 mar  6 20:20 ls.png
-rw-r--r--  1 juan juan  21999 mar  6 19:22 man.png
```

Imagen 43. Comando ls -la en la terminal Linux.

Ejemplos de direccionamiento absoluto

Ejemplo 1: /home/sergi/Desktop/Sistemas Operativos/hola.txt

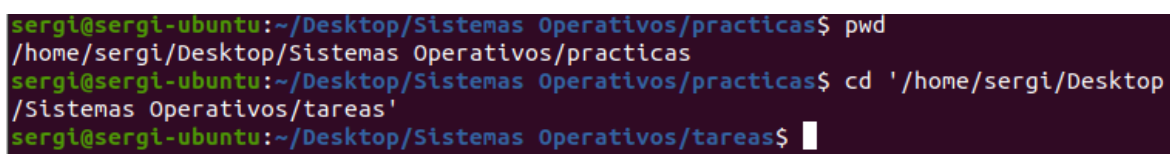


```
sergi@sergi-ubuntu: ~/Desktop/Sistemas Operativos
sergi@sergi-ubuntu:~$ cd '/home/sergi/Desktop/Sistemas Operativos'
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos$ ls
hola.txt  practicas
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos$
```

Imagen 44. Ruta absoluta del archivo “hola.txt”

En la primera línea de comando de la imagen anterior se puede apreciar que desde la carpeta raíz el usuario salta hasta el directorio “Sistemas Operativos”, en la segunda línea se lista todo el contenido del directorio y en el listado se puede encontrar el archivo “hola.txt”, por lo que se puede deducir que la ruta absoluta del archivo es la mencionada al principio.

Ejemplo 2: cd '/root/docs/Sistemas Operativos/practicas'



```
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos/practicas$ pwd
/home/sergi/Desktop/Sistemas Operativos/practicas
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos/practicas$ cd '/home/sergi/Desktop/Sistemas Operativos/tareas'
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos/tareas$
```

Imagen 45. Accediendo a un directorio en específico desde otro

En este comando se puede apreciar el uso de una ruta absoluta. Como se dijo en el punto anterior este comando puede ser útil cuando se quiere llegar a esa ruta específica desde otro directorio distinto. Aunque se puede llegar de otras formas (por ejemplo con el uso del `cd ..` o `cd ~`) esta forma es más rápida y práctica cuando se conoce la ruta absoluta.

Ejemplo 3: `cd '/home/sergi/Desktop/Sistemas Operativos/tareas'`

```
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos/practicas$ pwd
/home/sergi/Desktop/Sistemas Operativos/practicas
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos/practicas$ cd '/home/sergi/Desktop
/Sistemas Operativos/tareas'
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos/tareas$
```

Imagen 46. Accediendo a un directorio dentro de otro directorio distinto

En la imagen anterior se presenta el escenario en el que el sistema se encuentra en un directorio distinto al que se quiere acceder, lo que cualquier persona pensaría que se puede hacer es ejecutar el comando “`cd ~`”, pero utilizar la ruta absoluta con este comando es mucho más útil y rápido.

Ejemplos de direccionamiento relativo

Ejemplo 1: `cd practicas`

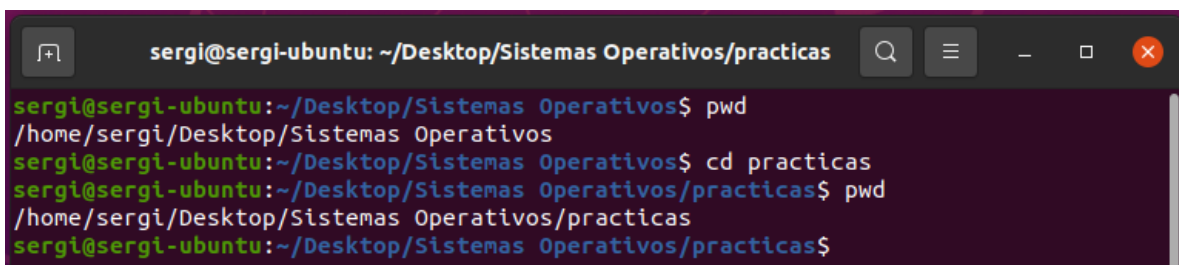
A terminal window titled 'sergi@sergi-ubuntu: ~/Desktop/Sistemas Operativos/practicas'. The terminal shows the following commands and output: `pwd` returns `/home/sergi/Desktop/Sistemas Operativos`; `cd practicas` is executed; `pwd` returns `/home/sergi/Desktop/Sistemas Operativos/practicas`; and the prompt shows the current directory as `~/Desktop/Sistemas Operativos/practicas$`.

Imagen 47. Accediendo a un directorio con ruta relativa

Contrario a lo que se pueda pensar, la ruta anterior es completamente válida (y se puede apreciar en la imagen), siempre y cuando exista el directorio en la dirección de la cual se manda a llamar. A diferencia de las rutas absolutas, las relativas sólo funcionan cuando se está en la misma dirección en la cual se encuentra el archivo o directorio al que se quiere acceder. En ocasiones, las rutas relativas son más rápidas de ejecutar que las absolutas.

Ejemplo 2: `cd 'Sistemas Operativos/practicas'`

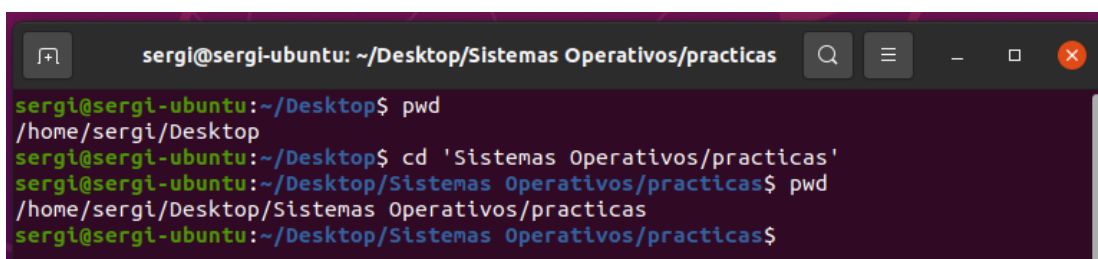
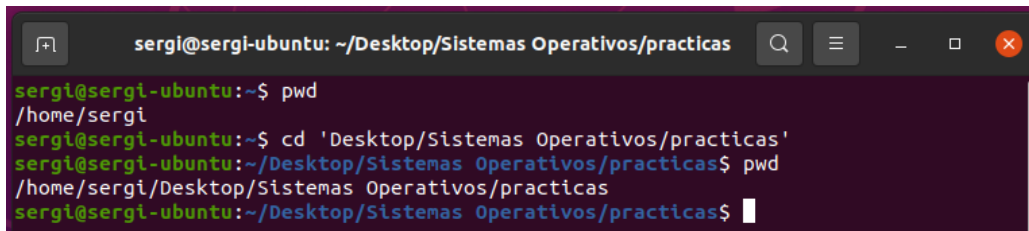
A terminal window titled 'sergi@sergi-ubuntu: ~/Desktop/Sistemas Operativos/practicas'. The terminal shows the following commands and output: `pwd` returns `/home/sergi/Desktop`; `cd 'Sistemas Operativos/practicas'` is executed; `pwd` returns `/home/sergi/Desktop/Sistemas Operativos/practicas`; and the prompt shows the current directory as `~/Desktop/Sistemas Operativos/practicas$`.

Imagen 48. Accediendo a un directorio de la misma rama

Como se puede observar en este comando, y a diferencia del ejemplo 2. de direccionamiento absoluto, este comando solo contiene 2 directorios; esto es posible porque el sistema se encuentra en el mismo directorio que contiene al otro llamado “Sistemas Operativos”. Hacer uso de este comando puede ser más rápido que utilizar la ruta absoluta.

Ejemplo 3: cd ‘Desktop/Sistemas Operativos/practicas’



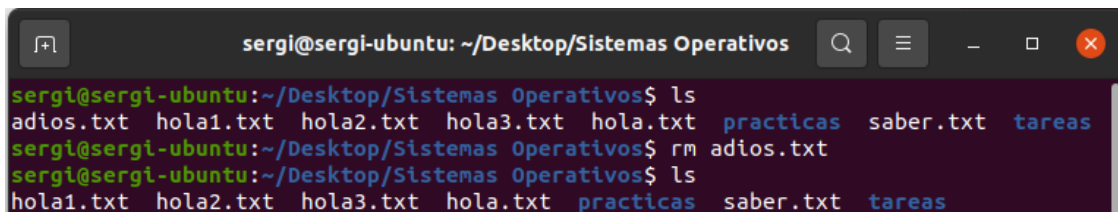
```
sergi@sergi-ubuntu: ~/Desktop/Sistemas Operativos/practicas
sergi@sergi-ubuntu:~$ pwd
/home/sergi
sergi@sergi-ubuntu:~$ cd 'Desktop/Sistemas Operativos/practicas'
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos/practicas$ pwd
/home/sergi/Desktop/Sistemas Operativos/practicas
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos/practicas$
```

Imagen 49. Accediendo a un directorio con su ruta casi absoluta

En la imagen anterior se puede apreciar la ejecución del comando cd haciendo uso de la ruta relativa del directorio “practicas”. Aunque parezca que se utilizó la ruta absoluta en realidad no se trata de esta, si no de una ruta relativa con muchos directorios. Resulta, que para que pueda ser una ruta absoluta se debe partir desde la carpeta raíz y esta ruta parte desde el directorio “Desktop”, es por eso que no es una ruta absoluta; también se puede descubrir que es una ruta relativa porque si se ejecuta esta línea de comando en otro usuario, saltaría un error.

Borrado de archivos con rm y los comodines “*” y “?”

Ejemplo 1: rm adios.txt

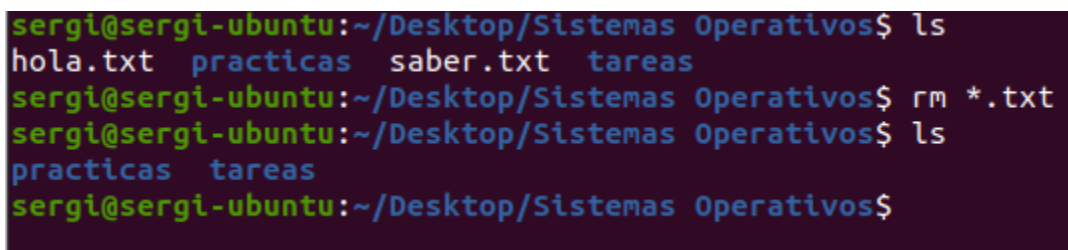


```
sergi@sergi-ubuntu: ~/Desktop/Sistemas Operativos
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos$ ls
adios.txt hola1.txt hola2.txt hola3.txt hola.txt practicas saber.txt tareas
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos$ rm adios.txt
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos$ ls
hola1.txt hola2.txt hola3.txt hola.txt practicas saber.txt tareas
```

Imagen 50. Comando rm

El comando anterior elimina el archivo “adios.txt” ubicado en el mismo directorio que el usuario.

Ejemplo 2: rm *.tx



```
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos$ ls
hola.txt practicas saber.txt tareas
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos$ rm *.txt
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos$ ls
practicas tareas
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos$
```

Imagen 51. Comando rm*

Este comando elimina todos los archivos de tipo texto que haya en el directorio actual, si hay otro tipo de archivos o directorios estos no sufrirán ninguna modificación ni serán eliminados.

Ejemplo 3: rm hola?.txt

```
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos$ ls
hola1.txt hola2.txt hola3.txt hola.txt practicas saber.txt tareas
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos$ rm hola?.txt
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos$ ls
hola.txt practicas saber.txt tareas
```

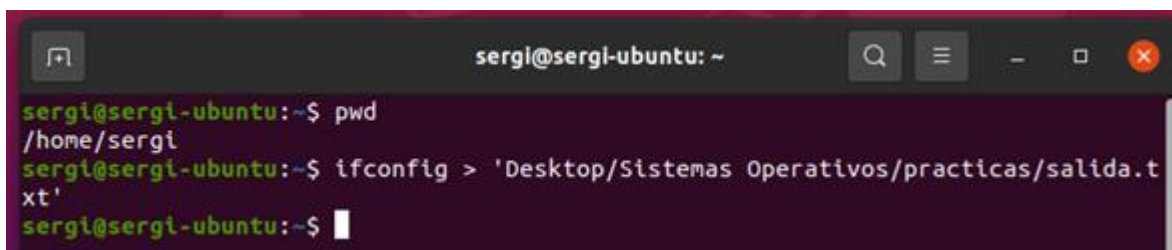
Imagen 52. Comando rm?

El último comando elimina todos los archivos llamados “hola_.txt” que contengan un número o carácter en el guion bajo, pero no elimina el archivo llamado “hola.txt”.

Redireccionamiento con los comandos > y >>

Comando >

El comando > se utiliza para escribir lo que muestre la terminal (o el resultado de algún comando ejecutado) en un archivo de texto. En este caso será utilizado para ejecutar el comando “ifconfig” y escribir el resultado en el archivo “salida.txt” ubicado en la dirección “/home/sergi/Desktop/Sistemas Operativos/practicas”.

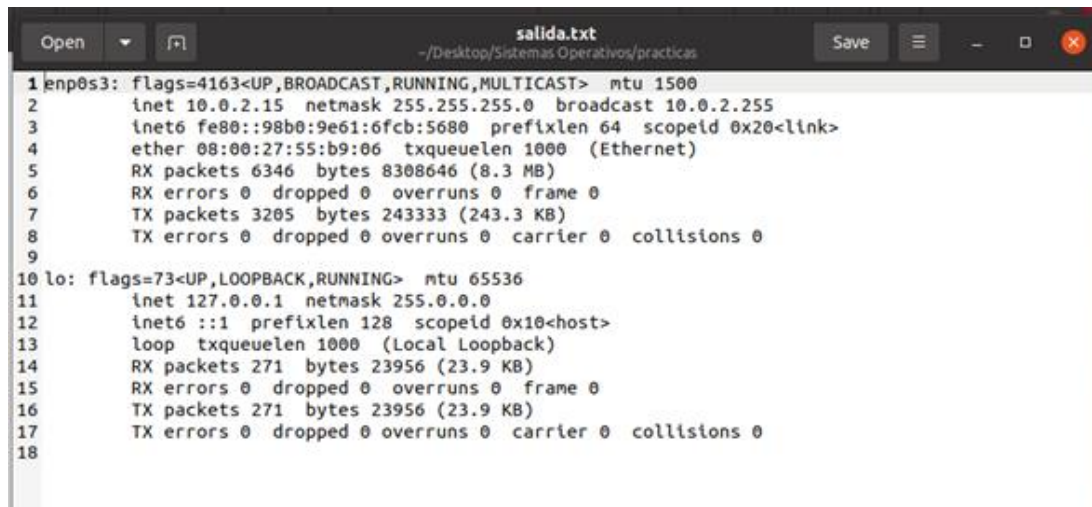
A terminal window titled 'sergi@sergi-ubuntu: ~' with search, menu, and window control icons. The terminal shows the following commands and output: 'sergi@sergi-ubuntu:~\$ pwd' followed by '/home/sergi'; 'sergi@sergi-ubuntu:~\$ ifconfig > 'Desktop/Sistemas Operativos/practicas/salida.txt'' followed by 'xt''; and 'sergi@sergi-ubuntu:~\$' followed by a cursor. The output of 'ifconfig' is not visible, indicating it was redirected to the file.

```
sergi@sergi-ubuntu:~$ pwd
/home/sergi
sergi@sergi-ubuntu:~$ ifconfig > 'Desktop/Sistemas Operativos/practicas/salida.t
xt'
sergi@sergi-ubuntu:~$
```

Imagen 53. Comando ifconfig redireccionado al archivo salida.txt

En la imagen anterior se pueden apreciar varios comandos, el primero muestra la dirección en la que se encuentra el usuario, tal y como lo muestra el resultado de este comando, el usuario se encuentra en una ruta diferente a la del archivo al que se quiere llegar, por lo que tiene que hacer uso de la ruta absoluta del mismo para llegar a él. El siguiente comando que es ejecutado es un ifconfig con redireccionamiento al archivo anteriormente mencionado, se puede observar que el comando anterior no imprimió ningún resultado en la terminal, por lo que se puede deducir que el resultado fue escrito en el archivo de texto.

Abriendo el archivo se encuentra lo siguiente.



```
1 enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
2     inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
3     inet6 fe80::98b0:9e61:6fcb:5680 prefixlen 64 scopeid 0x20<link>
4     ether 08:00:27:55:b9:06 txqueuelen 1000 (Ethernet)
5     RX packets 6346 bytes 8308646 (8.3 MB)
6     RX errors 0 dropped 0 overruns 0 frame 0
7     TX packets 3205 bytes 243333 (243.3 KB)
8     TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
9
10 lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
11     inet 127.0.0.1 netmask 255.0.0.0
12     inet6 ::1 prefixlen 128 scopeid 0x10<host>
13     loop txqueuelen 1000 (Local Loopback)
14     RX packets 271 bytes 23956 (23.9 KB)
15     RX errors 0 dropped 0 overruns 0 frame 0
16     TX packets 271 bytes 23956 (23.9 KB)
17     TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
18
```

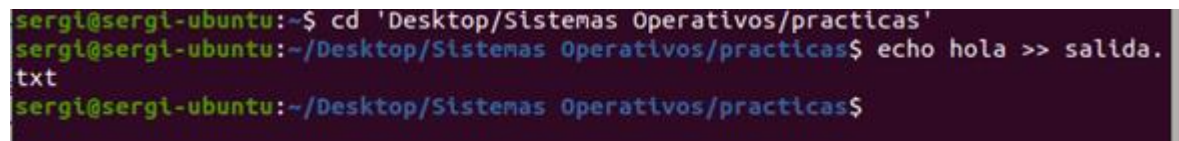
Imagen 54. Archivo “salida.txt” con el resultado del comando ifconfig

Dentro del archivo se encuentra el resultado del comando ifconfig, esto ocurre gracias al redireccionamiento.

Comando >>

El comando >> realiza lo mismo que el comando anterior, la diferencia se encuentra en que el comando > sobrescribe el archivo, es decir, borra lo que hay en el archivo y escribe el resultado del comando ejecutado, mientras que el comando >> solo concatena al final del archivo la salida de la terminal.

En este ejemplo se ejecutará el comando “echo” con una cadena para que se escriba en el archivo de texto.

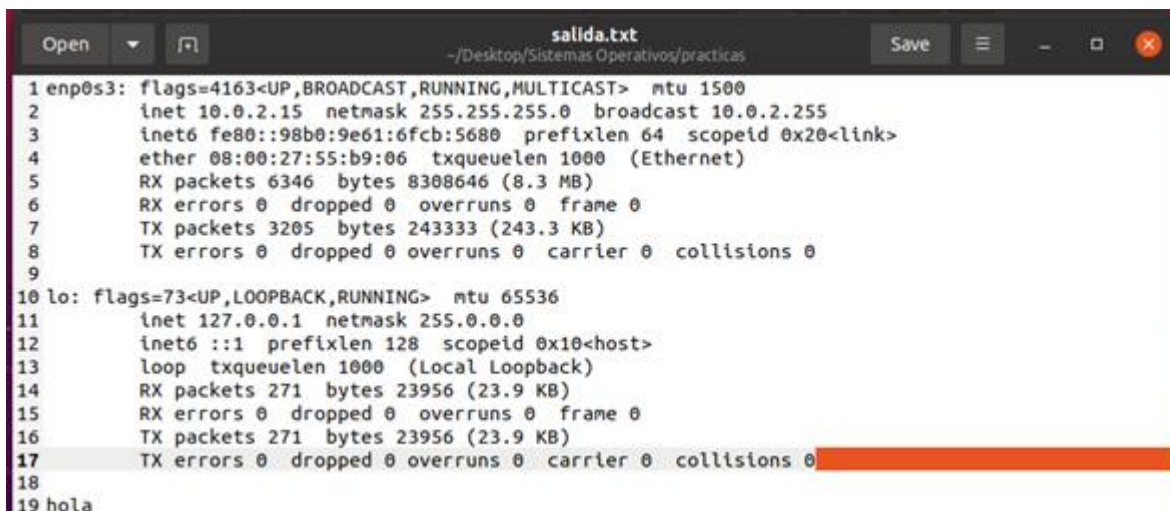


```
sergi@sergi-ubuntu:~$ cd 'Desktop/Sistemas Operativos/practicas'
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos/practicas$ echo hola >> salida.txt
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos/practicas$
```

Imagen 55. Comando echo redireccionado al archivo salida.txt.

En la imagen anterior se tienen dos líneas de comando, la primera muestra el comando cd el cual posiciona al usuario en el mismo directorio que el archivo “salida.txt”, la segunda línea de comando redirecciona la salida del comando echo con la cadena “hola” al archivo mencionado, haciendo uso solamente de la ruta relativa del archivo.

Si todo funcionó correctamente la palabra “hola” debería aparecer en el archivo.



```
1 enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
2   inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
3   inet6 fe80::98b0:9e61:6fcb:5680 prefixlen 64 scopeid 0x20<link>
4   ether 08:00:27:55:b9:06 txqueuelen 1000 (Ethernet)
5   RX packets 6346 bytes 8308646 (8.3 MB)
6   RX errors 0 dropped 0 overruns 0 frame 0
7   TX packets 3205 bytes 243333 (243.3 KB)
8   TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
9
10 lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
11   inet 127.0.0.1 netmask 255.0.0.0
12   inet6 ::1 prefixlen 128 scopeid 0x10<host>
13   loop txqueuelen 1000 (Local Loopback)
14   RX packets 271 bytes 23956 (23.9 KB)
15   RX errors 0 dropped 0 overruns 0 frame 0
16   TX packets 271 bytes 23956 (23.9 KB)
17   TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
18
19 hola
```

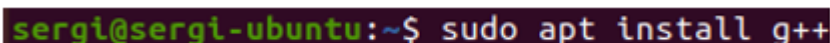
Imagen 56. Palabra hola concatenada con el texto del archivo

La imagen anterior muestra el archivo salida.txt con la salida del comando ifconfig del punto anterior y la palabra hola concatenada al final del archivo, resultado del comando ejecutado anteriormente.

Instalación de software por línea de comandos

Las siguientes imágenes mostrarán el proceso para instalar g++, el cual es un compilador para el lenguaje de c++, desde la terminal de Ubuntu, Linux.

La imagen que se puede ver a continuación presenta el comando necesario para realizar esta instalación.



```
sergi@sergi-ubuntu:~$ sudo apt install g++
```

Imagen 57. Comando para instalar g++.

Como se mencionó anteriormente el comando de la imagen anterior realiza la instalación del software g++; la palabra sudo quiere decir “Super User DO”, que básicamente funciona como lo haría un perfil de administrador en windows, apt significa Advanced Packaging Tool la cual es una herramienta que simplifica la eliminación e instalación (en este caso) de cualquier aplicación, la palabra install sirve para instalar cualquier software y por último el nombre del software que se quiere instalar.

Después de ejecutar el comando anterior y si no hay errores de sintaxis, comenzará la instalación como lo muestra la siguiente imagen.

```

sergi@sergi-ubuntu:~$ sudo apt install g++
[sudo] password for sergi:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libfprint-2-tod1 libllvm10
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  g++-9 libstdc++-9-dev
Suggested packages:
  g++-multilib g++-9-multilib gcc-9-doc libstdc++-9-doc
The following NEW packages will be installed:
  g++ g++-9 libstdc++-9-dev
0 upgraded, 3 newly installed, 0 to remove and 4 not upgraded.
Need to get 10.1 MB of archives.
After this operation, 46.7 MB of additional disk space will be used.
Do you want to continue? [Y/n]

```

Imagen 58. Primera parte de la instalación

Tal y como se puede apreciar en la imagen anterior cuando el comando fue aceptado el sistema solicita la contraseña del superusuario (o administrador), una vez aceptada la contraseña proporcionada por el usuario comienza la descarga de los archivos necesarios, después de completar la descarga aparece un mensaje de advertencia que dice “Después de esta operación, 46.7 MB de espacio adicional del disco será usado. ¿Quieres continuar? [Y/n]”, el usuario toma la decisión que prefiera, si su opción fue si (y) entonces comenzará la segunda parte de la instalación.

Como se puede prever, la segunda instalación será el proceso de instalar los datos descargados en la imagen anterior.

```

Do you want to continue? [Y/n] y
Get:1 http://mx.archive.ubuntu.com/ubuntu focal-updates/main amd64 libstdc++-9-dev amd64 9.3.0-17ubuntu1-20.04 [1 714 kB]
Get:2 http://mx.archive.ubuntu.com/ubuntu focal-updates/main amd64 g++-9 amd64 9.3.0-17ubuntu1-20.04 [8 405 kB]
Get:3 http://mx.archive.ubuntu.com/ubuntu focal/main amd64 g++ amd64 4:9.3.0-1ubuntu2 [1 604 B]
Fetched 10.1 MB in 23s (449 kB/s)
Selecting previously unselected package libstdc++-9-dev:amd64.
(Reading database ... 186314 files and directories currently installed.)
Preparing to unpack .../libstdc++-9-dev_9.3.0-17ubuntu1-20.04_amd64.deb ...
Unpacking libstdc++-9-dev:amd64 (9.3.0-17ubuntu1-20.04) ...
Selecting previously unselected package g++-9.
Preparing to unpack .../g++-9_9.3.0-17ubuntu1-20.04_amd64.deb ...
Unpacking g++-9 (9.3.0-17ubuntu1-20.04) ...
Selecting previously unselected package g++.
Preparing to unpack .../g++_4%3a9.3.0-1ubuntu2_amd64.deb ...
Unpacking g++ (4:9.3.0-1ubuntu2) ...
Setting up libstdc++-9-dev:amd64 (9.3.0-17ubuntu1-20.04) ...
Setting up g++-9 (9.3.0-17ubuntu1-20.04) ...
Setting up g++ (4:9.3.0-1ubuntu2) ...
update-alternatives: using /usr/bin/g++ to provide /usr/bin/c++ (c++) in auto mode
Processing triggers for man-db (2.9.1-1) ...
sergi@sergi-ubuntu:~$

```

Imagen 59. Segunda parte de la instalación

Una vez se hayan terminado de instalar y verificar todos los archivos el software está listo para su uso.

Para comprobar que el software g++ se ha instalado satisfactoriamente se probará compilando un archivo de lenguaje c

```

Processing triggers for man-db (2.9.1-1) ...
sergi@sergi-ubuntu:~$ cd Desktop
sergi@sergi-ubuntu:~/Desktop$ cd 'Sistemas Operativos'
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos$ ls
a.out  archivos.c  hola.txt
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos$ g++ archivos.c
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos$

```

Imagen 60. Comprobando el funcionamiento del software.

Como se puede apreciar en la imagen anterior el archivo llamado archivos.c compiló perfectamente y no hubo errores de sintaxis con el comando g++. Por lo que se puede concluir que el software ha sido instalado con éxito.

Instalación de software por el entorno gráfico

Al igual que en el punto pasado las siguientes imágenes mostrará la instalación de un software, pero esta vez en el entorno gráfico. El software que se instalará será el editor de texto llamado “Sublime”. Para realizar lo último, se debe ir al escritorio de Linux y ubicar la aplicación llamada “Ubuntu Software”



Imagen 61. Escritorio de Ubuntu señalando la aplicación de Ubuntu Software

Una vez se ha localizado la aplicación, como en la imagen anterior, hay que abrir dicha aplicación.

Al abrir la aplicación aparecerá una nueva ventana.

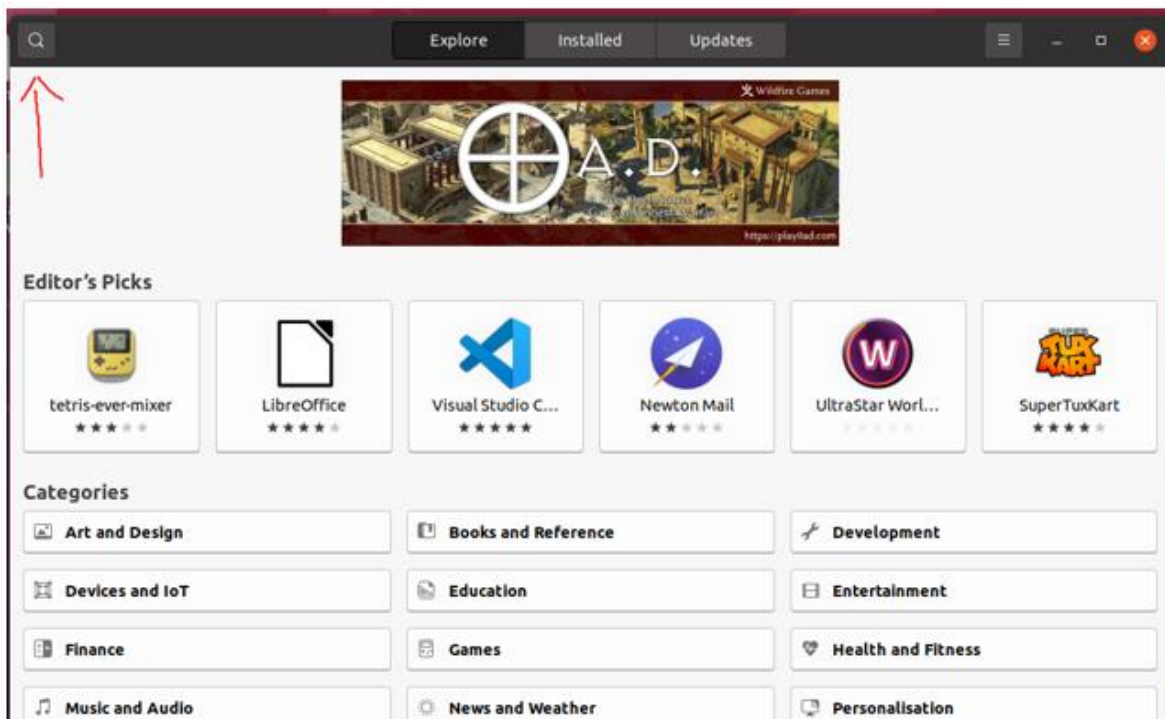


Imagen 62. Aplicación Ubuntu Software, con el botón buscar señalado

En la ventana anterior hay que presionar el botón de búsqueda, para poder buscar la aplicación

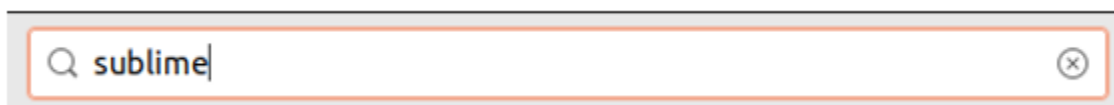


Imagen 63. Barra de búsqueda con la leyenda "sublime"

Ya que se ha escrito el nombre de la aplicación se tiene que dar enter para realizar la búsqueda.

Una vez realizada la búsqueda aparecerán distintas aplicaciones similares a las palabras de la búsqueda

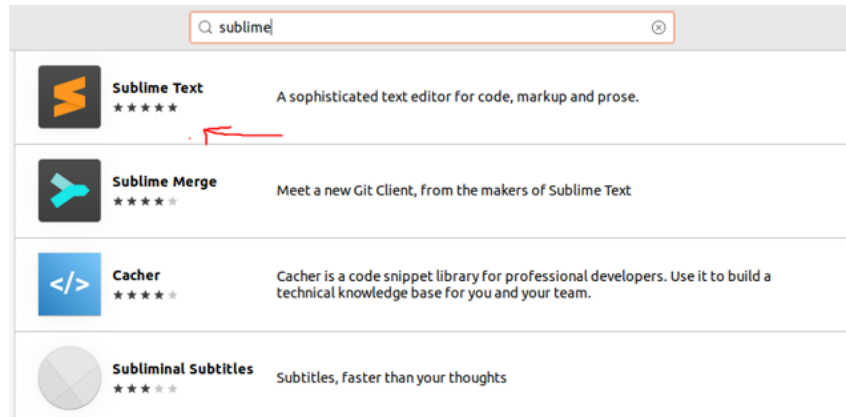


Imagen 64. Resultados de búsqueda

Como se mencionó anteriormente, aparecen distintas aplicaciones simultáneamente, pero la aplicación que se busca instalar es la primera opción.

Ya que se ha identificado la aplicación correcta se da clic en ella para así poder acceder a la página de la aplicación.



Imagen 65. Página de Sublime lista para instalar

Como se puede apreciar en la imagen anterior aparece un gran botón color verde que dice “Install”, se da clic en ese botón para instalar la aplicación.

Después de dar clic en el botón verde install, el sistema solicitará las credenciales del superusuario

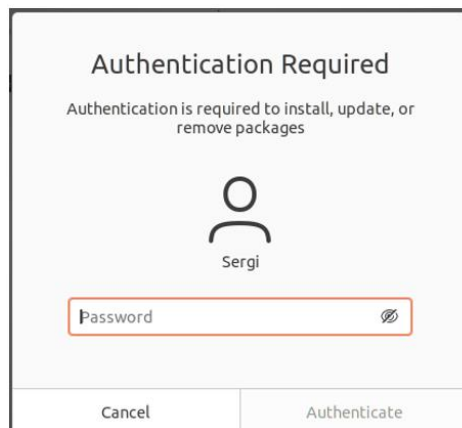


Imagen 66. Credenciales del usuario solicitadas

Una vez el usuario haya ingresado las credenciales correctas, la instalación comenzará.

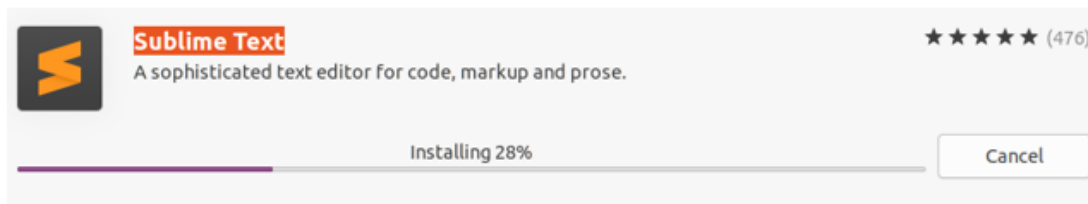


Imagen 67. Instalando Sublime text

Como se aprecia en la imagen anterior la instalación de Sublime ha comenzado.

Una vez que haya terminado la instalación la barra morada desaparecerá y será reemplazada por un botón rojo que dice Remove. Véase la siguiente imagen.



Imagen 68. Instalación completa, botón Instalar cambia por botón Remover

Esta última imagen quiere decir que la aplicación ha sido instalada con éxito.

Ahora solo hay que buscarla en el menú de aplicaciones.

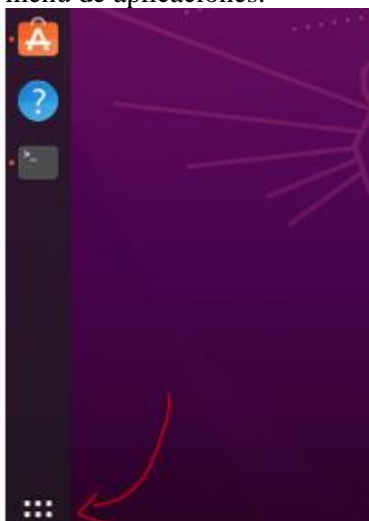


Imagen 69. Escritorio de Ubuntu, con el menú de aplicaciones señalado

Se busca en el escritorio el menú de aplicaciones.

Una vez encontrado se da clic en él y se busca la aplicación recién instalada.

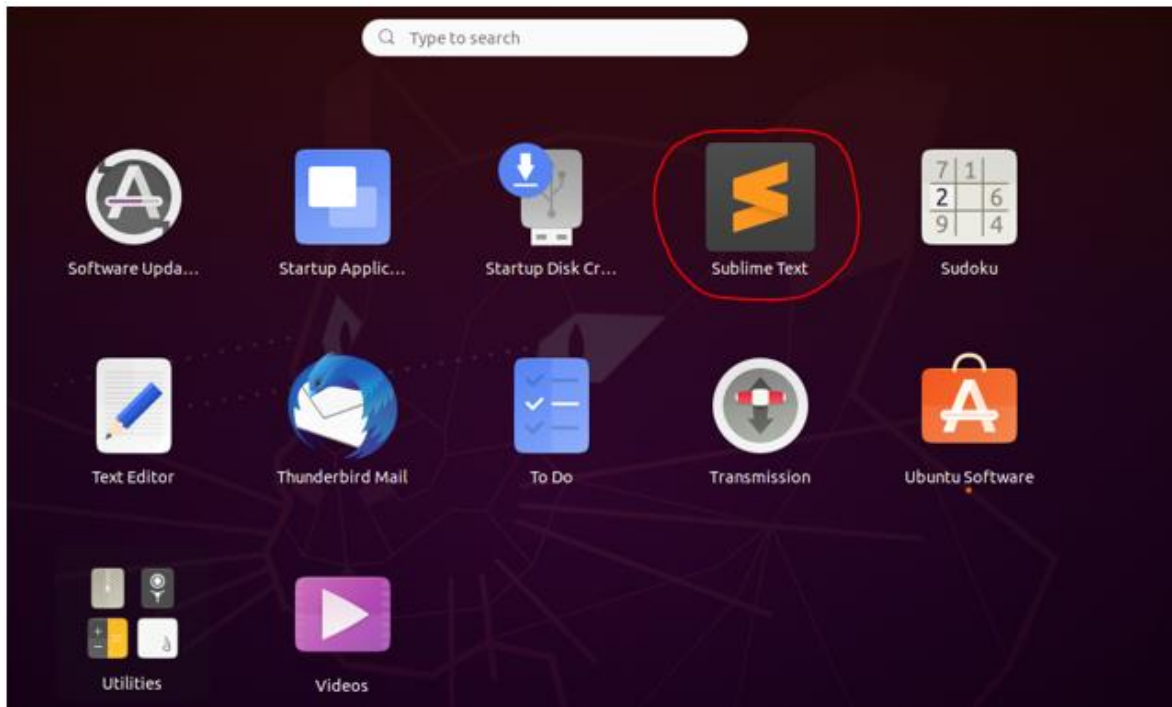


Imagen 70. Aplicación exitosamente instalada

Al encontrar la aplicación en el menú de aplicaciones, se puede concluir que el software se ha instalado con éxito.

Ejemplos usando jerarquía en los directorios para mover y copiar archivos y directorios desde terminal

Ejemplo1: `mv *.txt ..`

```
sergi@sergi-ubuntu: ~
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos/tareas$ pwd
/home/sergi/Desktop/Sistemas Operativos/tareas
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos/tareas$ ls
archivo2.txt archivo3.txt archivo4.txt archivo.txt
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos/tareas$ mv *.txt ..
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos/tareas$ ls
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos/tareas$ cd ..
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos$ ls
archivo2.txt archivo4.txt directorio1 directorio3 hola2.c hola.c tareas
archivo3.txt archivo.txt directorio2 hola1.c hola3.c practicas
```

Imagen 71. Mover todos los archivos de texto

Este comando mueve todos los archivos de texto existentes en el directorio, al directorio padre de este.

Ejemplo 2: cp hola?.c /home/sergi/Desktop/Proyecto

```
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos$ ls
archivo2.txt  archivo4.txt  directorio1  directorio3  hola2.c  hola.c  tareas
archivo3.txt  archivo.txt   directorio2  hola1.c     hola3.c  practicas
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos$ cp hola?.c /home/sergi/Desktop/Proyecto
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos$ cd /home/sergi/Desktop/Proyecto
sergi@sergi-ubuntu:~/Desktop/Proyecto$ ls
hola1.c  'reporte (3rd copy).txt'  'reporte (copy).txt'
hola2.c  'reporte (4th copy).txt'  reporte.txt
hola3.c  'reporte (another copy).txt'
```

Imagen 72. Copiar todos los archivos tipo “c” con numeración

El comando anterior copia todos los archivos llamados “hola_.” (excepto el archivo hola.c) del directorio actual al directorio “Proyecto”.

Ejemplo 3: mv *.txt '/home/sergi/Desktop/Sistemas Operativos/practicas'

```
sergi@sergi-ubuntu:~/Desktop/Proyecto$ ls
hola1.c  'reporte (3rd copy).txt'  'reporte (copy).txt'
hola2.c  'reporte (4th copy).txt'  reporte.txt
hola3.c  'reporte (another copy).txt'
sergi@sergi-ubuntu:~/Desktop/Proyecto$ mv *.txt '/home/sergi/Desktop/Sistemas Operativos/practicas'
sergi@sergi-ubuntu:~/Desktop/Proyecto$ ls
hola1.c  hola2.c  hola3.c
sergi@sergi-ubuntu:~/Desktop/Proyecto$ cd '/home/sergi/Desktop/Sistemas Operativos/practicas'
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos/practicas$ ls
'reporte (3rd copy).txt'  'reporte (another copy).txt'  reporte.txt
'reporte (4th copy).txt'  'reporte (copy).txt'         salida.txt
```

Imagen 73. Mover todos los archivos de texto a un directorio, con su ruta absoluta

Este otro comando mueve todos los documentos de texto del directorio actual al directorio “practicas”.

Ejemplo 4: cp /home/sergi/Desktop/*

```
sergi@sergi-ubuntu: ~/Desktop/Sistemas Operativos/directorio3
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos/practicas$ pwd
/home/sergi/Desktop/Sistemas Operativos/practicas
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos/practicas$ ls /home/sergi/Desktop
escritorioenc.c  escritorioword.docx  'Sistemas Operativos'
escritorio.txt   Proyecto
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos/practicas$ cp /home/sergi/Desktop/* ..
cp: -r not specified; omitting directory '/home/sergi/Desktop/Proyecto'
cp: -r not specified; omitting directory '/home/sergi/Desktop/Sistemas Operativos'
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos/practicas$ cd ..
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos$ ls
archivo2.txt  archivo.txt  directorio3  escritorioword.docx  hola3.c  tareas
archivo3.txt  directorio1  escritorioenc.c  hola1.c           hola.c
archivo4.txt  directorio2  escritorio.txt  hola2.c           practicas
```

Imagen 74. Copiar todos los archivos de un directorio a otro

Este comando copia todos los archivos (excepto directorios) del directorio “Desktop” a la carpeta “Sistemas Operativos” (la cual es padre del directorio “practicas” que es la ruta en la que se encuentra el usuario).

Ejemplo 5: mv directorio1 directorio2 directorio3

```
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos$ ls
archivo2.txt  archivo.txt  directorio3  escritorioenc.c  hola3.c  tareas
archivo3.txt  directorio1  escritorioenc.c  hola1.c  hola.c
archivo4.txt  directorio2  escritorio.txt  hola2.c  practicas
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos$ mv directorio1 directorio2 directorio3
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos$ ls
archivo2.txt  archivo.txt  escritorio.txt  hola2.c  practicas
archivo3.txt  directorio3  escritorioenc.c  hola3.c  tareas
archivo4.txt  escritorioenc.c  hola1.c  hola.c
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos$ cd directorio3
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos/directorio3$ ls
directorio1  directorio2
sergi@sergi-ubuntu:~/Desktop/Sistemas Operativos/directorio3$
```

Imagen 75. Mover directorios (y su contenido) a otros directorios

El comando anterior mueve ambos directorios “directorio1” y “directorio2” al directorio llamado “directorio3”

Agregar, verificar, entrar en sesión y borrar usuario

Primeramente, agregamos a un usuario nuevo a nuestro sistema, para esto usamos el siguiente comando: *sudo adduser nombre*.

En la imagen 76 podemos ver que se añade un nuevo usuario llamado usuario:

```
brandon@BDMV:~$ sudo adduser usuario
Añadiendo el usuario `usuario' ...
Añadiendo el nuevo grupo `usuario' (1002) ...
Añadiendo el nuevo usuario `usuario' (1002) con grupo `usuario' ...
Creando el directorio personal `/home/usuario' ...
Copiando los ficheros desde `/etc/skel' ...
Nueva contraseña:
Vuelva a escribir la nueva contraseña:
passwd: contraseña actualizada correctamente
Cambiando la información de usuario para usuario
Introduzca el nuevo valor, o pulse INTRO para usar el valor predeterminado
Nombre completo []: Brandon Meza
Número de habitación []:
Teléfono del trabajo []:
Teléfono de casa []:
Otro []:
¿Es correcta la información? [S/n] s
brandon@BDMV:~$
```

Imagen 76. Se muestra la creación de un nuevo usuario.

Como se ve en la figura, para crear un usuario nos piden una contraseña para este, además de datos extra.

Para verificar que existe un usuario usamos el siguiente comando: ***getent passwd nombre***

A continuación, se muestra la imagen 77 en donde verificamos que el usuario creado anteriormente exista:

```
brandon@BDMV:~$ getent passwd usuario
usuario:x:1002:1002:Brandon Meza,,,:/home/usuario:/bin/bash
brandon@BDMV:~$ getent passwd usuarios
```

Imagen 77. Se muestra la verificación de que existe el usuario.

Como se ve, si el usuario existe nos muestra sus datos, datos que introducimos al momento de crearlo, si no existe simplemente no nos muestra nada.

Ahora iniciaremos sesión con el usuario creado, para esto se usa el siguiente comando: ***sudo login nombre***

En la imagen 78 podemos ver el inicio de sesión del usuario:

```
brandon@BDMV:~$ sudo login usuario
[sudo] password for brandon:
Contraseña:
Linux BDMV 4.19.0-14-amd64 #1 SMP Debian 4.19.171-2 (2021-01-30) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
usuario@BDMV:~$
```

Imagen 78. Se ve el inicio de sesión del usuario.

Como vemos, al ingresar el comando nos pide la contraseña del super usuario y posteriormente la contraseña del usuario que quiere iniciar sesión, cuando inicia sesión cambia el usuario de la terminal.

Finalmente, vamos a borrar el usuario creado, para esto usamos el siguiente comando: ***sudo userdel nombre***.

Para borrar el usuario tenemos que cerrar sesión en este, en la imagen 79 se ve como se elimina el usuario:

```
brandon@BDMV:~$ sudo userdel usuario
[sudo] password for brandon:
```

Imagen 79. Se ve el comando para borrar usuarios.

Como se ve, nos pide la contraseña del sudo para poder borrar el usuario.

Hola mundo con distintos editores

En los siguientes apartados se mostrará cómo crear, compilar y ejecutar el programa básico de la programación el “Hola mundo” apoyándose de los distintos editores y herramientas del sistema

operativo Linux Ubuntu. Para el anterior proceso se creó una carpeta llamada “practicass” dentro de la carpeta personal del sistema operativo.

A continuación, se presenta la imagen de la nueva carpeta **prácticas**.

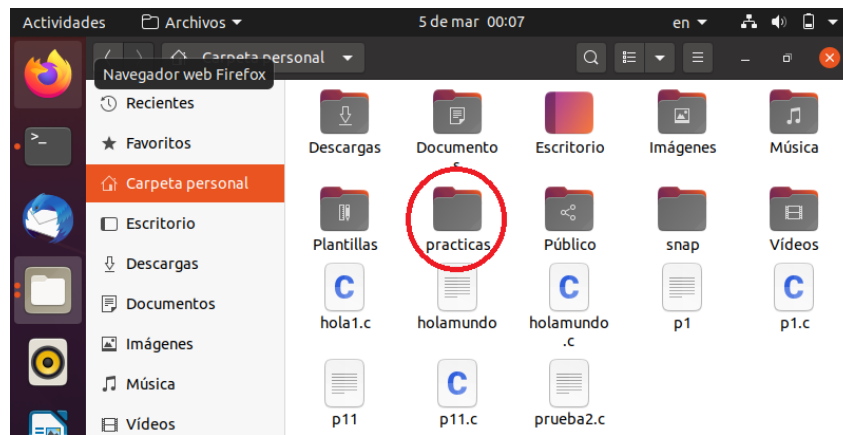


Imagen 80. Carpeta principal que contiene la carpeta “practicass”

Se puede apreciar que es muy sencillo crear una carpeta en el sistema operativo, simplemente se da clic a los 9 puntos de la esquina inferior izquierda, se busca la “carpeta principal”, una vez ubicados en esta carpeta se da un clic derecho en cualquier espacio en blanco del lado derecho y se selecciona la opción “crear nueva carpeta”.

Una vez creada esta carpeta, accederemos a ella, a través de la terminal, escribiendo el comando **cd** + nombre de la carpeta. Lo anterior con la intención de guardar los archivos que se vayan creando.

Debajo se presenta la captura del anterior proceso mencionado.

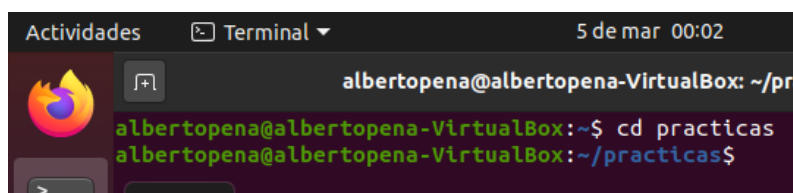


Imagen 81. Acceso a la carpeta practicas desde la terminal

Como se puede notar es muy sencillo acceder a la carpeta desde la terminal.

Hola mundo utilizando nano

Nano es un editor muy parecido a pico que sirve de apoyo para crear los programas que se ocupen en cierto momento. Para acceder a este editor, a través de la terminal, se escriba lo siguiente: nano + nombre que se dará al archivo.extensión , en este caso se colocó **nano + h1.c**.

Abajo se presenta dicho proceso.

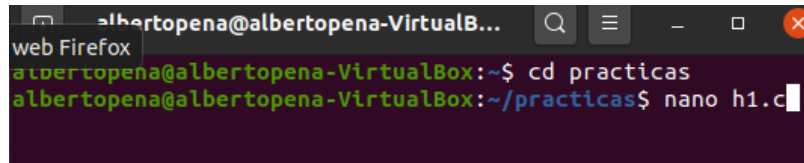


Imagen 82. Apertura de nano desde la terminal

Se nota que el acceder y crear ciertos archivos desde la terminal es similar a como se hace en el cmd de Windows.

Después de escribir nano y el nombre del archivo, se da **ENTER** y automáticamente se abre el editor nano. Dentro del editor nano, se escribe lo necesario para ejecutar un programa.

La siguiente imagen muestra la edición dentro del editor.

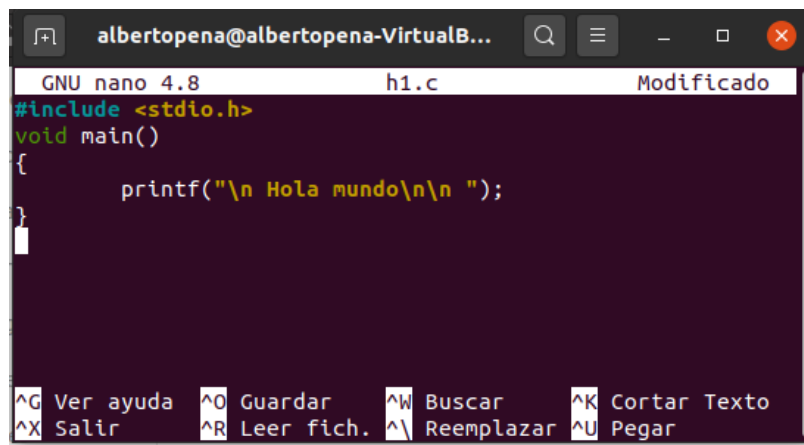


Imagen 83. Edición de un programa desde Nano

Una vez escrito el programa dentro del editor, se presionan las teclas **CTRL + O** para guardar el archivo dentro de la carpeta que se seleccionó al principio. A continuación, se puede ver que el terminal arrojó un mensaje en la parte inferior de la captura. Dicho mensaje indica el nombre con el que se guardará el archivo.

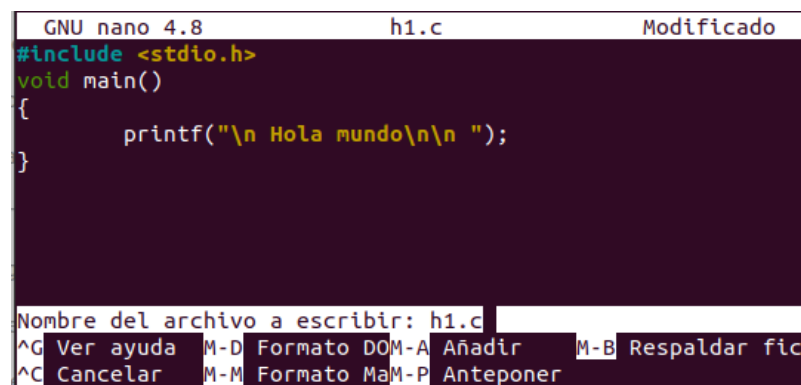
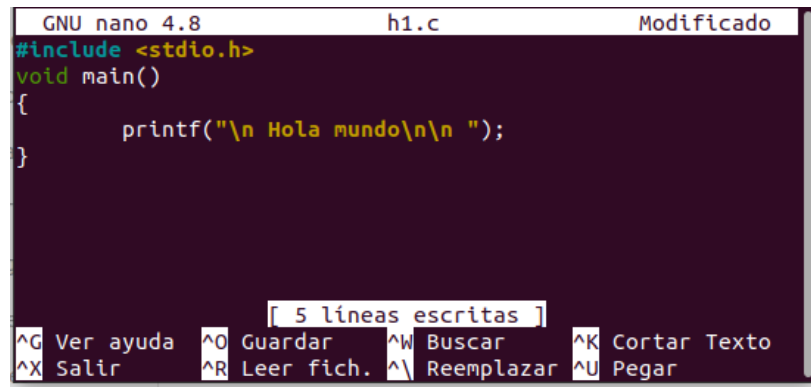


Imagen 84. Mensaje que arroja el terminal después de presionar CTRL + O

Después de presionar esas teclas y después de que el mensaje apareciera en la parte inferior, se presiona **ENTER** para que finalice el proceso de guardado. Si el archivo se guardó correctamente en la parte inferior aparecerá un nuevo texto que indique el número de líneas del código.



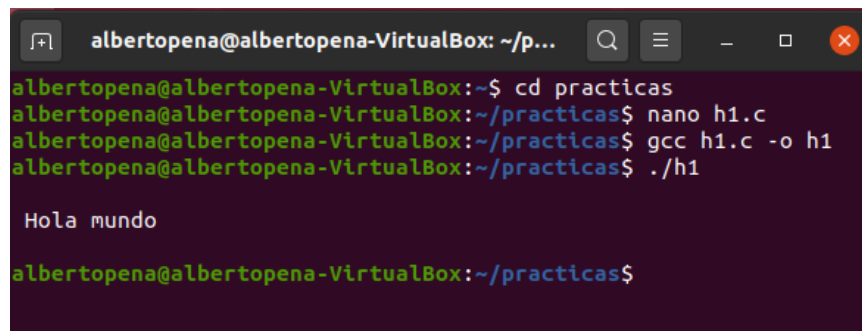
```
GNU nano 4.8 h1.c Modificado
#include <stdio.h>
void main()
{
    printf("\n Hola mundo\n\n ");
}

[ 5 líneas escritas ]
^G Ver ayuda  ^O Guardar    ^W Buscar     ^K Cortar Texto
^X Salir      ^R Leer fich. ^\ Reemplazar ^U Pegar
```

Imagen 85. Mensaje de líneas de código escritas, después de guardar

Finalmente, para ver que el programa fue guardado y editado correctamente, se regresa el terminal inicial presionando las teclas **CTRL +X**. Estando en el terminal se compila y ejecuta el programa.

Seguidamente se muestran los comandos que se utilizaron para compilar y ejecutar el programa h1.c



```
albertopena@albertopena-VirtualBox: ~/p...
albertopena@albertopena-VirtualBox:~$ cd practicas
albertopena@albertopena-VirtualBox:~/practicas$ nano h1.c
albertopena@albertopena-VirtualBox:~/practicas$ gcc h1.c -o h1
albertopena@albertopena-VirtualBox:~/practicas$ ./h1

Hola mundo

albertopena@albertopena-VirtualBox:~/practicas$
```

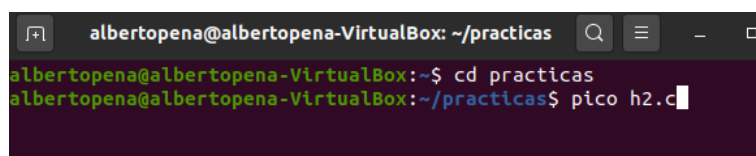
Imagen 86. Compilación y ejecución del programa editado en nano

Hola mundo utilizando pico

Pico es un editor muy parecido a nano que de igual forma sirve de herramienta para la creación de los programas en Linux.

La manera de abrirlo desde el terminal es la misma que cuando se abre el editor nano. Se coloca la palabra pico + el nombre del nuevo archivo junto con su terminal.

Debajo se puede ver la apertura del editor.



```
albertopena@albertopena-VirtualBox: ~/practicas
albertopena@albertopena-VirtualBox:~$ cd practicas
albertopena@albertopena-VirtualBox:~/practicas$ pico h2.c
```

Imagen 87. Apertura de pico con el terminal

En seguida se abre el editor en el cual se puede comenzar a escribir el programa, este editor es como cualquier otro, al parecer de los que realizaron la práctica es muy cómodo con el usuario.



```
albertopena@albertopena-VirtualBox: ~/practicas
GNU nano 4.8 h2.c Modificado
#include <stdio.h>

void main()
{
    printf("\n Hola mundo\n\n");
}

^G Ver ayuda ^O Guardar ^W Buscar ^K Cortar Tex ^J Justificar
^X Salir ^R Leer fich. ^\ Reemplazar ^U Pegar ^T Ortografia
```

Imagen 88. Edición del programa con pico

Como sucedió con nano, para guardar el archivo.c se presiona **CTRL+O** y debajo del editor aparece el nombre con el que se guardará.

Abajo se puede ver la acción mencionada.



```
GNU nano 4.8 h2.c Modificado
#include <stdio.h>

void main()
{
    printf("\n Hola mundo\n\n");
}

Nombre del archivo a escribir: h2.c
^G Ver ayuda M-D Formato DOS M-A Añadir M-B Respalda fich
^C Cancelar M-M Formato Mac M-P Anteponer ^T A Ficheros
```

Imagen 89. Guardado del archivo

Después que aparece el texto en la parte inferior, se presiona la tecla **ENTER** para que finalice el guardado del archivo, si el guardado fue exitoso, entonces aparecerá un mensaje en pantalla indicando el número de líneas que tiene el programa.



```
albertopena@albertopena-VirtualBox: ~/practicass
GNU nano 4.8 h2.c
#include <stdio.h>

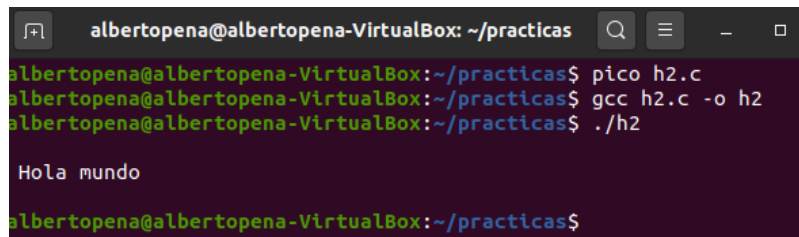
void main()
{
    printf("\n Hola mundo\n\n");
}

[ 7 líneas escritas ]
^G Ver ayuda ^O Guardar ^W Buscar ^K Cortar Text ^J Justificar
^X Salir ^R Leer fich.^_ Reemplazar ^U Pegar ^T Ortografía
```

Imagen 90. Mensaje de líneas de código escritas, después de guardar

Posteriormente, el siguiente paso es compilar y ejecutar el código que se ha guardado, por lo tanto, se empiezan a introducir una serie de comandos para realizar dichas acciones.

La imagen, a continuación, muestra cómo hacerlo.



```
albertopena@albertopena-VirtualBox: ~/practicass
albertopena@albertopena-VirtualBox:~/practicass$ pico h2.c
albertopena@albertopena-VirtualBox:~/practicass$ gcc h2.c -o h2
albertopena@albertopena-VirtualBox:~/practicass$ ./h2

Hola mundo

albertopena@albertopena-VirtualBox:~/practicass$
```

Imagen 91. Compilación y ejecución del programa, utilizando pico.

Hola mundo utilizando vi

A la hora de realizar el programa con vi, la verdad es que se batalló un poco más, debido a que es un editor algo robusto, para poder escribir en él se tienen que presionar ciertos comandos.

Lo primero que se hace es abrir el editor, escribiendo vi + nombre del archivo + . extensión.

La imagen que está abajo lo muestra de mejor manera.

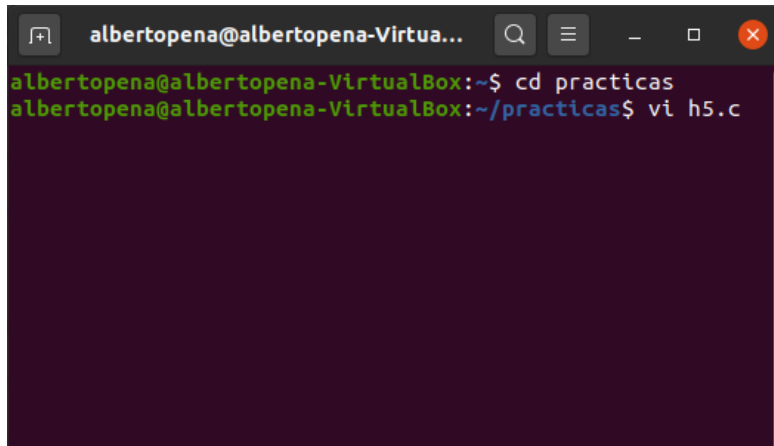
A terminal window with a dark background. The title bar shows 'albertopena@albertopena-Virtua...'. The prompt is 'albertopena@albertopena-VirtualBox:~\$'. The command 'cd practicas' has been entered, and the prompt has changed to 'albertopena@albertopena-VirtualBox:~/practicas\$'. The command 'vi h5.c' has been entered, and the prompt is now 'albertopena@albertopena-VirtualBox:~/practicas\$'.

Imagen 92. Apertura del editor vi

Una vez dentro del editor, se tienen que presionar una serie de comandos para poder crear el programa. Los comandos que se ocuparon fueron:

- “i” → Editar por la derecha
- “o” → Salto de la línea en la edición
- “r” → Reemplazar
- “ESC” → Se deja de editar

A continuación, se presenta el código resultante.

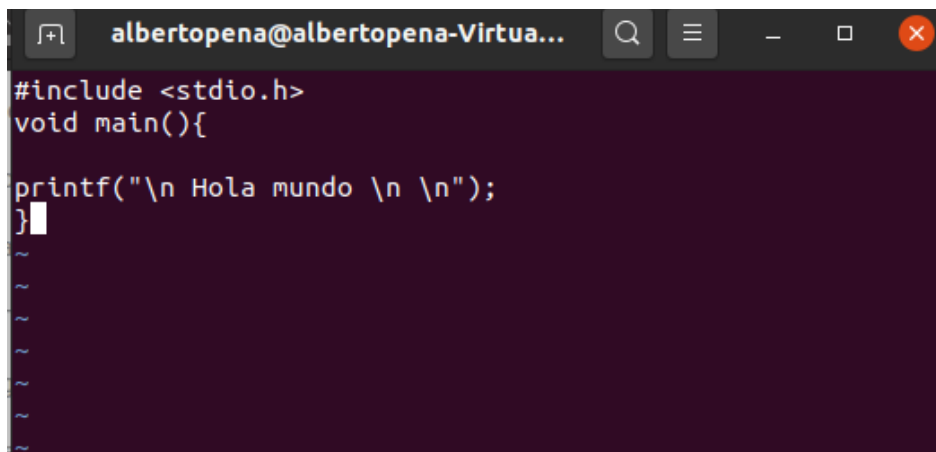
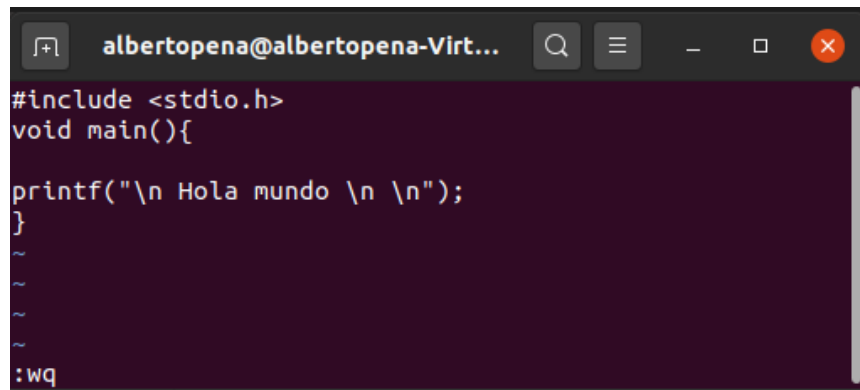
A terminal window with a dark background. The title bar shows 'albertopena@albertopena-Virtua...'. The prompt is 'albertopena@albertopena-VirtualBox:~/practicas\$'. The code entered is: '#include <stdio.h>', 'void main(){', 'printf("\n Hola mundo \n \n");', and '}'. The cursor is at the end of the closing brace.

Imagen 93. Código editado con vi.

Luego, para poder salir del editor, se tiene que presionar la tecla de los dos puntos + wq. Lo anterior para que se pueda salir del editor y guardar el programa.

A continuación, la imagen para que se pueda ver de mejor forma.



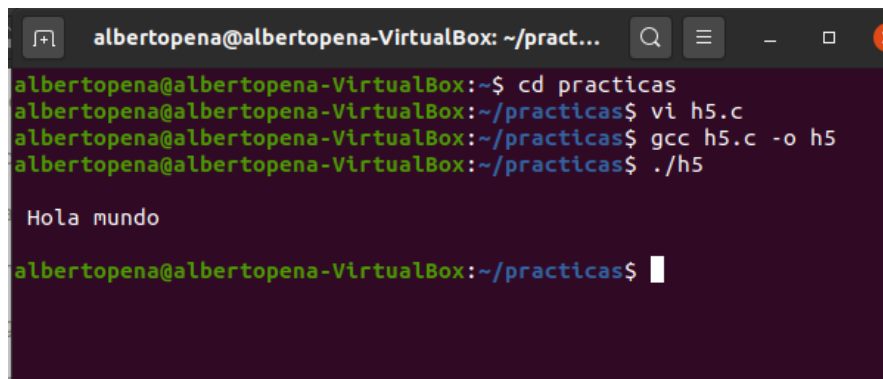
```
#include <stdio.h>
void main(){

printf("\n Hola mundo \n \n");
}
~
~
~
~
:wq
```

Imagen 94. Guardado del programa y salida del editor

Una vez de regreso en el terminal, se vuelve a compilar y ejecutar con una serie de comandos anteriormente utilizada.

La captura de abajo, lo explica, visualmente, mejor.



```
albertopena@albertopena-VirtualBox: ~/pract...
albertopena@albertopena-VirtualBox:~$ cd practicas
albertopena@albertopena-VirtualBox:~/practicas$ vi h5.c
albertopena@albertopena-VirtualBox:~/practicas$ gcc h5.c -o h5
albertopena@albertopena-VirtualBox:~/practicas$ ./h5

Hola mundo

albertopena@albertopena-VirtualBox:~/practicas$
```

Imagen 95. Compilación y ejecución con vi.

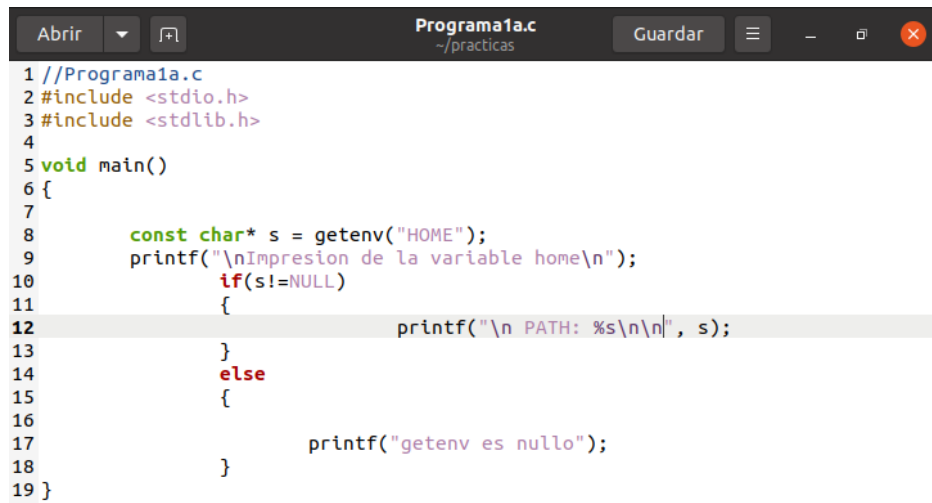
Programa 1a

Este programa permite imprimir la variable de entorno “HOME” de Linux. Como ya se había visto anteriormente, una variable de entorno es un valor dinámico cargado a la memoria que puede ser utilizado por varios procesos que funcionan simultáneamente.

La variable de entorno HOME, en Linux, muestra la ruta de acceso al directorio actual del usuario.

La función getenv en C recibe un apuntador constante a una variable de entorno, en este caso, un apuntador a HOME. Devuelve un puntero a la entrada de la tabla de entorno que contiene varname.

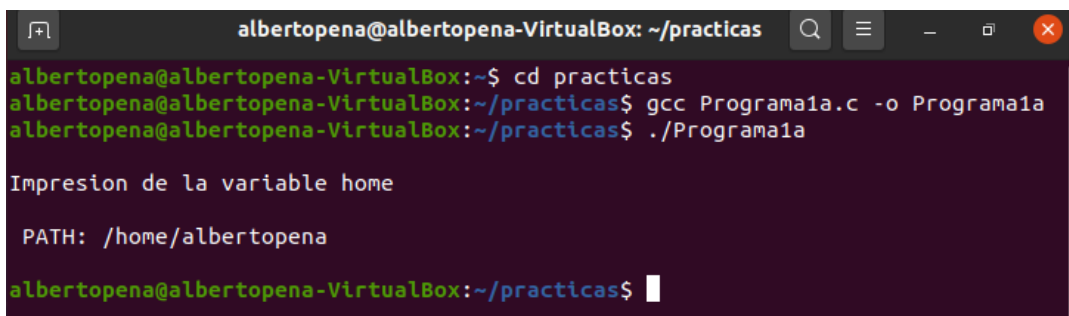
A continuación, se muestra el programa que se realizó para poder visualizar la información de la variable de entorno.



```
1 //Programa1a.c
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 void main()
6 {
7
8     const char* s = getenv("HOME");
9     printf("\nImpresion de la variable home\n");
10    if(s!=NULL)
11    {
12        printf("\n PATH: %s\n\n", s);
13    }
14    else
15    {
16
17        printf("getenv es nullo");
18    }
19 }
```

Imagen 96. Código del programa que imprime la variable de entorno HOME

Para finalizar, se compiló y ejecutó el programa, se tuvo el resultado esperado debido a que se imprimió en pantalla la ruta de la variable HOME.



```
albertopena@albertopena-VirtualBox: ~/practicas
albertopena@albertopena-VirtualBox:~$ cd practicas
albertopena@albertopena-VirtualBox:~/practicas$ gcc Programa1a.c -o Programa1a
albertopena@albertopena-VirtualBox:~/practicas$ ./Programa1a

Impresion de la variable home

PATH: /home/albertopena

albertopena@albertopena-VirtualBox:~/practicas$
```

Imagen 97. Compilación y ejecución del Programa1.c

Programa 2a

Se tiene que realizar un programa en C que haga lo siguiente:

- Crear un archivo
- Introducir diez números dados por el usuario
- Grabar los números del punto anterior en el archivo creado
- Cerrar en archivo

En la imagen 98 a continuación podemos ver la función escribeNum de nuestro programa:

```

int numeros[10], i;
FILE * ptr; //puntero de tipo archivo

ptr=fopen("numeros.txt", "w+"); //w+ crea un archivo de lectura y escritura

if(ptr==NULL){
    printf("El archivo no se creo correctamente\n"); //si no se puede abrir o
                                                    //crear el archivo se sale del programa
    return;
}

for(i=0; i<10;i++){
    printf("Ingresa un numero: \n");
    scanf("%d", &numeros[i]); //leemos los numeros ingresados por el usuario
    fprintf(ptr, "%d ",numeros[i]); //Se van guardando los números en el archivo
}
printf("Datos guardados\n");
fclose(ptr); //Cerramos el archivo

```

Imagen 98. Se muestra la función escribeNum().

Como se ve, en esta función creamos el archivo usando un puntero que será de tipo archivo para posteriormente ir almacenando la cantidad de datos solicitada e irlos guardando en el archivo llamado números.txt.

Una vez realizado nuestro programa procedemos a compilarlo para verificar que no tenga algún error, la compilación la podemos ver en la imagen 99:

```

brandon@BDMV:~$ gcc programa2a.c -o prog2
brandon@BDMV:~$

```

Imagen 99. Compilación del programa2a.

En la imagen 100 podemos ver la ejecución de nuestro programa:

```

Ingresa un numero:
1
Ingresa un numero:
2
Ingresa un numero:
3
Ingresa un numero:
4
Ingresa un numero:
5
Ingresa un numero:
6
Ingresa un numero:
7
Ingresa un numero:
8
Ingresa un numero:
9
Ingresa un numero:
10
Datos guardados

```

Imagen 100. Se muestra la ejecución del programa2a.c

Para demostrar el correcto funcionamiento del programa, se ve en la imagen 101 el archivo creado por el programa2a, además de mostrar el contenido de este:



Imagen 101. se muestra el archivo números.txt y su contenido.

Como se ve, nuestro programa2a creo el archivo de forma correcta y guardo los números ingresados por el usuario igual de forma correcta.

El código completo de este programa se encuentra en los anexos del presente reporte.

Programa 3a

Para el programa 3a se solicita hacer un programa que haga lo siguiente:

- Abrir el archivo del programa anterior (Programa2a) en modo lectura
- Leer los números del archivo
- Mostrar los números en pantalla
- Calcular promedio, suma, multiplicación, desviación estándar y varianza de los números
- Mostrar los resultados en pantalla
- Cerrar el archivo

Nuestro programa consta de dos funciones, una se encarga de leer los números del archivo del programa anterior, esta es la función leeNum que vemos en la imagen 102:

```
FILE * ptr; //puntero de tipo archivo
int num[10], i;
ptr= fopen("numeros.txt", "r"); //Abrimos el archivo del programa2 en modo lectura

if(ptr==NULL){
    printf("El archivo no se abrio correctamente\n"); //si no se puede abrir o
                                                    //crear el archivo se sale del programa
    return;
}
for(i=0; i<10; i++){
    fscanf(ptr, "%d ", &num[i]); //leemos los datos del archivo
    printf("%d\n", num[i]);
}
fclose(ptr);
operaciones(num); //llamamos a la funcion que hace las oeperaciones
```

Imagen 102. Función leeNum() del programa3a.

Se ve que en esta función lo único que hacemos es leer los números del archivo.txt y llamamos a la función operaciones, la cual se encarga de realizar las operaciones que se piden para el programa.

La imagen 103 muestra las operaciones que se realizan con los datos leídos del archivo:

```
float suma, mult, prom, desvEst, var, numf[10], desvi;
int i=0;
for(i=0; i<10; i++){
    numf[i]= (float)num[i]; //pasamos nuestro arreglo a flotantes para realizar las operaciones
    suma+=numf[i]; //hacemos la suma
    mult*=numf[i]; //hacemos la multiplicaciones
    desvi+= pow(numf[i] - (suma/10), 2); //calculamos las desviaciones necesarias para la varianza
}
prom=suma/10; //Calculamos el promedio
var=desvi/9; //calculamos la varianza
desvEst= sqrt(var); //calculamos la desviación estándar

printf("El promedio es: %0.2f\nLa suma es: %0.2f\nLa multiplicacion es: %e\n", prom, suma, mult);
printf("La desviacion estandar es: %0.2f\nLa varianza es: %0.2f\n", desvEst, var);
```

Imagen 103. Se muestra la función operaciones().

Como se ve en la imagen, en la función operaciones se hacen y se muestran en pantalla los resultados de las operaciones solicitadas.

A continuación, la imagen 104 que muestra la compilación y ejecución del programa 3a:

```
brandon@BDMV:~$ gcc programa3a.c -o prog3 -lm
brandon@BDMV:~$ ./prog3
1
2
3
4
5
6
7
8
9
10
El promedio es: 5.50
La suma es: 55.00
La multiplicacion es: -inf
La desviacion estandar es: 3.70
La varianza es: 13.71
```

Imagen 104. Se muestra la compilación y ejecución del programa 3a.

Como se ve en la imagen, para compilar este programa usamos -lm para que reconozca la librería mat.h para realizar distintas operaciones matemáticas.

El código completo de este programa se encuentra en los anexos del presente reporte.

Conclusiones

Martínez Ramírez Sergi Alberto

Esta práctica me fue de gran ayuda para entender un poco más el entorno de Linux, a pesar de que llevo trabajando en distintas distribuciones de Linux desde hace años, no he logrado entender por completo este sistema operativo, si no fuese por el gran parecido a Mac OS y un poco a Windows, no sabría moverme por este sistema. Hasta hace poco Linux era un sistema ajeno a mí, un sistema que solo utilizaba para ejecutar un par de comandos sencillos, como `ifconfig`, `ls`, `cd`, `mkdir`, etcétera. Pero ahora debido a que tengo que ocupar más este sistema realizar ejercicios como este son de mucha ayuda, ya que sirven para familiarizarte más que nada con la terminal de Linux, porque el entorno gráfico, al fin y al cabo, es como el de otros sistemas operativos, claro tiene sus diferencias, pero si ya has manejado otros sistemas, aprender Linux será un poco intuitivo (hablando del entorno gráfico), pero aprender a ocupar la terminal de Linux es un proceso un poco más difícil a mi parecer. Pero hacer el esfuerzo vale la pena, porque este sistema operativo es uno de los más poderosos y seguros, además que la gran mayoría de servidores en la industria están hechos con este sistema.

Meza Vargas Brandon David

El realizar esta práctica caí en cuenta de la utilidad que tiene el entorno de Linux y lo fácil que pueden ser algunas tareas en este sistema operativo con los distintos comandos que nos ofrece, eso sí, no es una tarea fácil el acostumbrarse a este sistema, pero la práctica sí que me ayudo bastante para familiarizarme de una mejor manera con este entorno.

El trabajar con comandos en Linux es de suma importancia, pues procesos sencillos como ver el calendario se pueden realizar desde la terminal de una manera más rápida que con el entorno gráfico, cosas como desde navegar entre directorios, ver archivos en el directorio actual, hasta configurar y administrar el sistema o ver procesos del sistema se pueden hacer desde la terminal.

No solo eso, como se realizó en la práctica también podemos borrar archivos de distintas formas desde la terminal, incluso insertar cosas dentro de un archivo sin necesidad de abrir un editor de textos. Como se ha mencionado, trabajar con la terminal de Linux nos ahorra mucho tiempo al momento de hacer ciertas tareas, por ejemplo, el instalar un software, porque sí, es posible instalar un software en específico desde la terminal y no necesariamente ir al entorno gráfico.

En adición a lo anterior, también podemos administrar los usuarios de nuestro sistema, esto nos trae ventajas de ahorro de tiempo y de recursos de nuestra computadora.

Entre otras cosas, Linux incorpora distintos editores de texto en los que podemos escribir desde una carta hasta un programa en un lenguaje de programación como en esta práctica, estos programas pueden ser compilados y ejecutados desde la misma terminal de una manera rápida y eficiente, pero tenemos que saber que es necesario haber instalado un compilador antes, como `gcc` o `g++` en el caso de programar en `c` o `c++`.

Con todo lo visto podemos decir que el conocer el entorno de Linux, y más aún, sus distintos comandos y el saber manejar archivos nos permitirá navegar por el sistema de una manera más sencilla y eficaz que al hacerlo de una manera gráfica.

Peña Atanasio Alberto

El utilizar un sistema operativo como lo es Linux es un gran reto, debido a que la mayoría venía con la idea de que sólo existían los sistemas operativos Windows y MacOS.

Leyendo un poco más acerca de Linux se pudo notar que este tiene infinidad de aplicaciones, principalmente para los Ingenieros en Sistemas Computacionales.

Para un usuario promedio, tal vez, al principio, le cueste trabajo adaptarse a Linux, ya que presenta algunos detalles como por ejemplo la distribución del teclado, entre otras. Pero en general cuenta con una gran interfaz gráfica y es muy amigable con el usuario.

A la hora de realizar la práctica algo que resultó interesante fue el hecho de que se pueden instalar ciertos paquetes de instalación desde la terminal, tan solo con escribir algunos comandos, además de que al escribir en la terminal la palabra -help se puede consultar qué hace el comando en cuestión.

La práctica, en ocasiones, se tornó complicada debido a que no se encontraba información en español tan explícita como en inglés. Otro factor que hizo un tanto más difícil a la práctica fue que no existen tutoriales o vídeos que explicaran de una manera concisa cómo trabajar en Linux, por lo que se tuvo, simplemente, que practicar y practicar.

A la hora de estar trabajando con los distintos editores como lo son nano, pico y vi se pudo ver que hay distintos caminos para realizar una misma tarea. Estos editores son de mucha ayuda a la hora de estar trabajando en tiempo real, lo anterior por que en la misma terminal se pueden crear, compilar y ejecutar archivos de una manera muy rápida y eficiente.

Con los comandos básicos aprendidos y el manejo de archivos se podría decir que se puede trabajar con más profundidad con este nuevo sistema operativo. Esperando que se cumplan con todos los objetivos de la materia.

Sarmiento Gutiérrez Juan Carlos

En lo personal siento que esta práctica me ayudó bastante a comprender cómo funciona la terminal de Linux, así como los comandos básicos, en total revise 35 comandos y 4 variaciones. Encontré bastante útiles los comandos que abrían un editor de texto tales como pico, nano o vi ya que como mencione en la práctica anterior Linux es una interfaz de desarrollo bastante más ágil, y el hecho de tener un editor al alcance de un comando la verdad es que vuela el proceso de codificación mucho más ágil. Además, existen otros comandos para manejar los archivos resultantes como more, mv, mkdir, ls, less etc. que nuevamente complementan mi idea anterior. Finalmente investigué más por mi cuenta y descubrí que existen los modificadores los cuales nos pueden hacer más detallado el resultado de un comando y esto tiene una importante aplicación en el desarrollo de scripts ya que constantemente hay que estar gestionando archivos, así como monitoreando procesos de nuestro sistema.

Bibliografía

- [1] MASLINUX, “GNOME vs KDE”, 2017. [En línea]. Disponible: <https://maslinux.wordpress.com/2017/10/04/gnome-vs-kde-cual-es-el-mejor-escritorio/>
- [2] ComputerNewAge, “GNU/Linux. La terminal o Línea de Comandos”. 2018. [En línea]. Disponible: <https://computernewage.com/gnu-linux/terminal/>
- [3] R, Velasco, “La Terminal de Linux”, 2021. [En línea]. Disponible: <https://www.programoergosum.com/cursos-online/raspberry-pi/243-terminal-de-linux-en-raspberry-pi/que-es-una-terminal-en-linux>
- [4] R, Morales, “Tipos de usuarios en Linux”, 2018. [En línea]. Disponible: <https://www.ticarte.com/contenido/tipos-de-usuarios-y-grupos-directorios-e-ids>
- [5] J, Blanco, “Rutas relativas y rutas absolutas”, 2016. [En línea]. Disponible: <https://sospedia.net/rutas-relativas-rutas-absolutas/>
- [6] S, González, “Redireccionamiento en Linux”. [En línea]. Disponible: <https://www.linuxtotal.com.mx/index.php?cont=redireccionamiento-en-linux>
- [7] ionos, “Los comandos en Linux”. [En línea]. Disponible: <https://www.ionos.mx/digitalguide/servidores/configuracion/comandos-de-linux-la-lista-fundamental/>
- [8] sololinux, “Variables de entorno”. [En línea]. Disponible: <https://www.sololinux.es/variables-de-entorno-que-son-y-para-que-sirven/>

Anexos

Código programa 2a

```
#include <stdio.h>

void escribeNum();

void escribeNum(){
    int numeros[10], i;
    FILE * ptr; //puntero de tipo archivo
    ptr=fopen("numeros.txt", "w+"); //w+ crea un archivo de lectura y escritura
    if(ptr==NULL){
        printf("El archivo no se creo correctamente\n"); //si no se puede abrir o crear el
        archivo se sale del programa
        return;
    }
    for(i=0; i<10;i++){
        printf("Ingresa un numero: \n");
        scanf("%d", &numeros[i]); //leemos los numeros ingresados por el usuario
        fprintf(ptr, "%d ",numeros[i]); //Se van guardando los números en el archivo
    }
    printf("Datos guardados");
    fclose(ptr); //Cerramos el archivo
}

int main(){
    escribeNum();
    return 0;
}
```

Código programa 3ª

```
#include <stdio.h>
#include <math.h>

void leeNum();
void operaciones(int num[]);

void leeNum(){
    FILE * ptr; //puntero de tipo archivo

    int num[10], i;

    ptr= fopen("numeros.txt", "r"); //Abrimos el archivo del programa2 en modo lectura
    if(ptr==NULL){
        printf("El archivo no se abrio correctamente\n"); //si no se puede abrir o crear el
        //archivo se sale del programa
        return;
    }

    for(i=0; i<10; i++){
        fscanf(ptr, "%d ", &num[i]); //leemos los datos del archivo
        printf("%d\n", num[i]);
    }

    fclose(ptr);

    operaciones(num); //llamamos a la funcion que hace las oepraciones
}

void operaciones(int num[]){
    float suma, mult, prom, desvEst, var, numf[10], desvi;

    int i=0;

    for(i=0; i<10; i++){
        numf[i]= (float)num[i]; //pasamos nuestro arreglo a flotantes para realizar las
        //operaciones

        suma+=numf[i]; //hacemos la suma

        mult*=numf[i]; //hacemos la multiplicaciones
    }
}
```

```

        desvi+= pow(numf[i] - (suma/10), 2); //calculamos las desviaciones necesarias para
la varianza
    }
    prom=suma/10; //Calculamos el promedio
    var=desvi/9; //calculamos la varianza
    desvEst= sqrt(var); //calculamos la desviación estándar
    printf("El promedio es: %0.2f\nLa suma es: %0.2f\nLa multiplicacion es: %e\n", prom,
suma, mult);
    printf("La desviacion estandar es: %0.2f\nLa varianza es: %0.2f", desvEst, var);
}
int main(){
    leeNum();
    return 0;
}

```