Práctica 2.
Juan Cruz de Urquiza.
Introducción a los Sistemas Operativos.
2022.
Facultad de Informática, Universidad Nacional de La Plata.

1.

a. Tanto Vi, Vim y Emacs son editores de texto que se pueden utilizar desde la línea de comandos.

b. El comando 'cat' imprimirá por pantalla el contenido del fichero sin ningún tipo de paginación ni posibilidad de modificarlo. 'more' permite visualizar por pantalla el contenido de un fichero de texto, con la diferencia con el anterior de que éste pagina los resultados. Primero mostrará por pantalla todo lo que se pueda visualizar sin hacer scroll y después, pulsando la tecla espacio avanzará de igual modo por el fichero. 'less' es el más completo de los tres, porque puede hacer todo lo que hace 'more' añadiendo mayor capacidad de navegación por el fichero (avanzar y retroceder), además de que sus comandos están basados en el editor 'vi', del cual se diferencia en que no tiene que leer todo el contenido del fichero antes de ser abierto.
Hay dos modos de operar en vi:modo de entrada y modo de comando. El modo de entrada sirve para introducir texto en un archivo, mientras que el modo de comando se utiliza para introducir comandos que llevan a cabo funciones específicas de vi específicas. El modo de comando es el modo estándar para vi.

c.

Comando	**Significado**
vi **nombre_de_archivo**	Abrir o crear el archivo.
vi	Abrir un archivo nuevo para nombrarlo más tarde.
vi -r **nombre_de_archivo**	Recuperar un archivo de una caída del sistema.
view **nombre_de_archivo**	Abrir archivo sólo para leer .

h	Moverse un carácter hacia la izquierda.
j	Moverse una línea hacia abajo.
k	Moverse una línea hacia arriba.
l	Moverse un carácter a la derecha.
w	Moverse una palabra a la derecha.
b	Moverse una palabra a la izquierda.
e	Moverse al final de la palabra actual.
H	Moverse a la parte de arriba de la pantalla.
M	Moverse al centro de la pantalla.
L	Moverse a la parte inferior de la pantalla.
a	Insertar caracteres a la derecha del cursor.
A	Insertar caracteres al final de la línea.

i	Insertar caracteres a la izquierda del cursor.
I	Insertar caracteres al principio de línea.
o	Insertar una línea por debajo del cursor.
O	Insertar una línea por encima del cursor.
cw	Cambiar una palabra (o parte de una palabra) a la derecha del cursor.
c	Cambiar una línea.
C	Cambiar desde el cursor hasta el final de la línea.
r	Reemplazar el carácter marcado por cursor por otro carácter
J	Unir la línea actual con la línea inferior.
~	Cambiar el tipo de letra (mayúscula o minúscula).
u	Deshacer el comando anterior.

U	Deshacer todos los cambios en la línea actual.
x	Eliminar el carácter del cursor.
X	Eliminar el carácter a la izquierda del cursor.
dw	Eliminar la palabra (o la parte de la palabra a la derecha del cursor).
dd	Eliminar la línea que contiene al cursor.
D	Eliminar la parte de la línea a la derecha del cursor.
dG	Eliminar hasta el final de línea.
d1G	Eliminar desde el principio del archivo hasta el cursor.
:x,y d	Eliminar las líneas de la x a la y.
:1,2 co 3	Copiar las líneas de la 1 a la 2 y ponerlas después de la línea 3.
:4,5 m 6	Mover las líneas de la 4 a la 5 y ponerlas después de la línea 6.

:set nu	Mostrar los números de las líneas.
:set nonu	Esconder los números de las líneas.
:set ic	En la búsqueda se ignora la distinción entre mayúsculas y minúsculas.
:set noic	En la búsqueda se distingue entre mayúsculas y minúsculas.
G	Ir a la última línea del archivo
1G	Ir a la primera línea del archivo
xG	Ir a la línea x
:g/search/s//replace/g	Buscar y reemplazar.
:w	Guardar los cambios.
:wq	Guardar los cambios y salir de vi.
:q!	Salir sin guardar los cambios.
:q	Salir de vi.

2.

a.

i. Se empieza a ejecutar el código del BIOS.
ii. El BIOS ejecuta el POST.
iii. El BIOS lee el sector de arranque (MBR).
iv. Se carga el gestor de arranque (MBC).
v. El bootloader carga el kernel y el initrd.
vi. Se monta el initrd como sistema de archivos raíz y se inicializan componentes esenciales.
vii. El Kernel ejecuta el proceso init y se desmonta el initrd.
viii. Se lee el /etc/inittab.
ix. Se ejecutan los scripts apuntados por el runlevel 1.
x. El final del runlevel 1 le indica que vaya al runlevel por defecto.
xi. Se ejecutan los scripts apuntados por el runlevel por defecto.
xii. El sistema está listo para usarse.

b. El proceso Init es un programa que el núcleo (kernel) ejecuta cuando arranca el sistema. Se encarga de inicializar todos los procesos normales que se necesiten ejecutar en el momento de arrancar; incluyendo los terminales que le permiten acceder al sistema, los servicios NFS, FTP, y cualquier proceso que quiera ejecutar cuando su máquina arranque. Posee el process ID (PID) 1. No tiene padre, y es el padre de todos los procesos.
Es lo primero que ejecuta el kernel cuando se inicia el sistema, y se encarga de montar el/los filesystems y de hacer disponible los demás dispositivos, cargando todos los subprocesos necesarios para el correcto funcionamiento del S.O.

c. Al ejecutar el comando pstree, la terminal muestra en forma de árbol todos los procesos del sistema.

d. Los runlevels son el modo en que arranca Linux. Al proceso de arranque se lo divide en diferentes niveles: runlevels. Cada uno es responsable de levantar (iniciar) o bajar (parar) una serie de servicios específicos.

e. Según el estándar, existen 7 runlevels que permiten iniciar un conjunto de procesos al arranque o apagado del sistema.

i. 0: halt (parada).
ii. 1: single user mode (monousuario).
iii. 2: multiuser, without NFS (modo multiusuario sin soporte de red).
iv. 3: full multiuser mode console (modo multiusuario completo por consola).
v. 4: no se utiliza.
vi. 5: X11 (modo multiusuario completo con login gráfico basado en X).
vii. 6: reboot.

Los Runlevels se encuentran definidos en /etc/inittab

No, no todas las distribuciones respetan el directorio de ubicación de los runlevels. Debian y Ubuntu, por ejemplo, no usan más el proceso de arranque SystemV, por lo que no será posible encontrar el directorio /etc/inittab, aunque los scripts que se ejecutan sí se encuentran en /etc/init.d

f. La finalidad del directorio /etc/inittab es almacenar los runlevels a ejecutarse cuando se inicia el sistema, el cual el proceso init leerá al iniciar el sistema. La estructura de los directorios que se encuentran allí es del tipo id:nivelesEjecución:acción:proceso donde

- Id: identifica la entrada en inittab (1 a 4 caracteres).
- NivelesEjecución: el/los niveles de ejecución en los que se realiza la acción.
- Acción: describe la acción a realizar.
    - wait: inicia cuando entra al runlevel e init espera a que termine
    - initdefault.
    - ctrlaltdel: se ejecutará cuando init reciba la señal SIGINT.
    - off, respawn, once, sysinit, boot, bootwait, powerwait, etc.
- Proceso: el proceso exacto que será ejecutado.

g. Suponiendo que se está en el runlevel X y se quiere cambiar al runlevel Y, se utiliza el comando init Y. Si Y=0, el sistema se apaga, ya que esta instrucción es equivalente a la instrucción halt. Si Y=6, el sistema se reinicia, y si Y no es ni 0 ni 6, el sistema cambia de runlevel. Sin embargo, este comando no es permanente sino que permanece hasta que el sistema sea apagado, es decir no cambia el nivel de ejecución por defecto que el sistema usa para arrancar.

Este comando solo sirve en procesos de arranque de SystemV o Upstart. En systemd, no se utiliza este comando.

h. Cuando init entra en un nivel de ejecución, llama al script rc con un argumento numérico que especifica el nivel de ejecución al que va. rc luego inicia y detiene los servicios en el sistema según sea necesario para llevar el sistema a ese nivel de ejecución.

Por ejemplo, si init se encuentra en el runlevel 2 y mediante el comando init el usuario decide ir al runlevel 5, el proceso init llama al script rc5, que inicia o detiene los servicios necesarios para poder llevar al runlevel 5.

Se encuentran almacenados en /etc/init.d

Es importante que los scripts comiencen y terminen en el orden correcto, ya que sino algunos procesos dentro del kernel del sistema pueden fallar. Si un script se está ejecutando, ningún otro script rc puede ejecutarse hasta que este termine.

i. insserv es un comando que analiza automáticamente los datos introducidos en la cabecera del script init y guarda los enlaces para los scripts de arranque y parada en los directorios de niveles de ejecución respectivos.
Se utiliza para administrar el orden de los enlaces simbólicos del /etc/rcX.d (X siendo un número del 0 al 6 representando los runlevels), resolviendo las dependencias de forma automática
Utiliza cabeceras en los scripts del /etc/init.d que permiten especificar la relación con otros scripts rc.
Lo que hace este comando es verificar los procesos que tienen o no que estar ejecutándose para poder ejecutar un script, cuándo ejecutarse y cuándo no, además de indicar al job en qué nivel de ejecución tendría que estar ejecutándose y en cuál nivel no.

j. Upstart permite la ejecución de trabajos en forma asincrónica a través de eventos (event-based), a diferencia de SystemV, que es estrictamente sincrónico (dependency-based).Estos trabajos se denominan jobs, y su principal objetivo es definir servicios o tareas a ser ejecutadas por init.

k. Las principales diferencias entre Upstart y SystemV es que Upstart utiliza los jobs en forma asincrónica; scripts de texto plano que definen las acciones/tareas a ejecutar ante determinados eventos, convirtiéndolo así en un proceso de arranque event-based, mientras que SystemV es estrictamente sincrónico. En este nuevo sistema de arranque no existe más el directorio /etc/inittab previamente utilizado para almacenar los runlevels. Upstart fue el primer reemplazo propuesto para SystemV en distribuciones Debian, Ubuntu, Fedora, entre otras. Sin embargo Upstart es compatible con SystemV porque comparten configuraciones en directorios como /etc/init y los scripts (jobs) en etc/init.d, comandos como init y runlevels.

l. Lo que reemplaza a los scripts rc de SystemV en Upstart son los jobs, que también son scripts pero de texto plano que se encargan de definir acciones y tareas que se tienen que ejecutar frente a ciertos eventos. Se encuentran definidos en /etc/init.d/.conf

m. La primer línea da una descripción del job a ejecutarse.
La segunda línea da información del autor que creó este job.
La tercer línea indica cuando es que se tiene que ejecutar el job: cuando la conexión del dispositivo esté activada (net-device-up), cuando el filesystem esté correctamente ejecutandose y en los niveles de ejecución 2,3,4 y 5.
La cuarta línea indica en qué niveles el job no tendría que estar ejecutándose: en los runlevels 0,1 y 6.
La última línea indica el directorio donde esta el script a ejecutarse cuando el job tenga que ejecutarse, en este caso /usr/sbin/mysqld

n. Systemd es un sistema que centraliza la administración de demonios y librerías del sistema, mejora el paralelismo de booteo, es controlado por el comando systemctl. Es compatible con SysV si es llamado como init. El demonio systemd reemplaza al proceso init: éste pasa a tener process id 1. Los runlevels son reemplazados por targets, aunque siguen existiendo dentro del sistema en algunas distribuciones, por ejemplo Debian, que combina tanto targets como runlevels. Al igual que con Upstart, el directorio /etc/inittab no existe más.

Las unidades de trabajo ahora son denominadas units que pueden tener 2 estados: active (activas) o inactive (inactivas). Hay diferentes tipos, como por ejemplo:

- Service: controla un servicio particular (.service).
- Socket: encapsula IPC, un socket del sistema o file system FIFO (.socket).
- Target: agrupa units o establece puntos de sincronización durante el booteo (.target).
- Snapshot: almacena el estado de un conjunto de unidades que puede ser restablecido más tarde (.snapshot).

o. La activación por socket en systemd hace referencia al hecho de que no todos los servicios que se inician en el booteo se utilizan, por lo que es más eficiente usar un método de iniciación bajo demanda que inicializar todos los servicios en el arranque del sistema. Se puede ofrecer una variedad de servicios que no son usados en el arranque como impresoras o servidores en puertos, e inicializarlos solo cuando son demandados por el usuario.

Cuando el socket recibe una conexión inicia el servicio y le pasa el socket. No hay necesidad de definir dependencias entre servicios: se inician todos los sockets en primer medida.

p. Los cgroups en systemd permiten organizar grupos de procesos en forma jerárquica. Agrupa conjuntos de procesos relacionados, de este modo limita y eficientiza el uso de recursos. No utiliza los id de los procesos para buscar archivos.

3.

a. La información de los usuarios está distribuida por todo el sistema de archivos dentro del sistema de cómputo, pero la mayor parte de cuentas de usuario se almacena en el archivo passwd.

b. Un UID (identificador de usuario) es un número asignado por Linux a cada usuario del sistema. Este número se utiliza para identificar al usuario en el sistema y para determinar a qué recursos del sistema puede acceder el usuario.

GID significa identificador de grupo. Es un valor numérico que se usa para representar un grupo específico. Este identificador numérico se utiliza para hacer referencia a grupos en los archivos /et /passwd y /etc

/group o sus equivalentes. Los archivos de contraseñas ocultas y el servicio de información de red también se refieren a GID numéricos.
Pueden haber 2 o más usuarios con el mismo UID, ya que éste es simplemente un número en un archivo de texto, y esto sirve para poder trabajar de forma remota y local en el mismo sistema, pero hacerlo no es recomendable ya que no hay forma de distinguir a las diferentes personas que utilizan el sistema y esto puede traer consecuencias de malfuncionamiento.

c. El usuario root en GNU/Linux es el usuario que tiene acceso administrativo al sistema. Los usuarios normales no tienen este acceso por razones de seguridad.
Puede existir otro usuario root en un sistema GNU/Linux, pero esto sería innecesario ya que tendría exactamente los mismos permisos que el usuario root original, y podría causar conflictos de seguridad dentro del sistema.
El UID 0 (cero) está reservado para el usuario root.

d. *Resuelto en máquina.*

e.

- **i.** Crea un nuevo usuario en el sistema. Con los parámetros se pueden especificar grupos y pasar comandos.
- **ii.** Modifica el usuario. Con los parámetros se puede dar nuevo nombre, bloquear yMod desbloquear, cambiar contraseña, entre otros.
- **iii.** Elimina un usuario. Con los parámetros se pueden eliminar ficheros y directorios, entre otros.
- **iv.** Sustituye la identidad del usuario. Con los parámetros se puede especificar grupos, pasar órdenes, y crear pseudo terminales, entre otros.
- **v.** Agrega a un usuario a un grupo. Con los parámetros se puede cambiar la contraseña del grupo, crear una cuenta del sistema, cambiar directorios, entre otros.
- **vi.** Imprime todo los usuarios que tienen una sesión iniciada. Con los parámetros se pueden ver procesos, ver el tiempo desde el último inicio del sistema, mostrar información sobre el sistema, entre otros.
- **vii.** Elimina un grupo. Con los parámetros se puede modificar los directorios del grupo antes de eliminarlo y forzar el borrado del grupo.
- **viii.** Modifica la contraseña de un usuario. Con los parámetros se puede borrar la contraseña, cambiarla, bloquear la cuenta de la contraseña indiciada, ver el estado, entre otros.

4.

a. Al ser Unix un sistema multiusuario, los archivos de cada usuario deben ser protegidos del resto de los usuarios. Lo mismo ocurre con

los archivos del sistema (programas, configuraciones, etc.). Esto tiene que ver no sólo con la confidencialidad de la información, sino también con la protección de errores involuntarios por parte de los usuarios. Para ello se utiliza un sistema de “permisos de archivos”. Este mecanismo permite que archivos y directorios “pertenezcan” a un usuario en particular. GNU/Linux permite que los archivos sean compartidos entre usuarios y grupos de usuarios.
Cada archivo pertenece a un usuario y a un grupo en particular. Los grupos usualmente son definidos por el tipo de usuarios que acceden al sistema. Hay pocos grupos definidos por el sistema (como bin y daemon) que son usados por el propio sistema para controlar el acceso a los recursos. Los usuarios comunes no pertenecen a estos grupos.

Los permisos están divididos en tres tipos: lectura (r), escritura (w) y ejecución (x). Estos permisos pueden ser fijados para tres clases de usuarios: el propietario del archivo/directorio, los integrantes del grupo al que pertenece y todos los demás usuarios.
El permiso de lectura permite a un usuario leer el contenido del archivo o en el caso de un directorio, listar el contenido del mismo.
El permiso de escritura permite a un usuario escribir y modificar el archivo (inclusive eliminarlo). Para directorios, el permiso de escritura permite crear nuevos archivos o borrar archivos ya existentes en el mismo.
El permiso de ejecución permite a un usuario ejecutar el archivo si es un programa. Para directorios, el permiso de ejecución permite al usuario ingresar al mismo.

b.

- **i.** Cambia los permisos de acceso. Con los parámetros se puede ver como se van procesando los ficheros, elegir la recursividad, informar cambios, entre otros.
- **ii.** Cambia el propietario y/o grupo de los ficheros. Con los parámetros se puede ver como se van procesando los ficheros, elegir la recursividad, informar cambios, entre otros.
- **iii.** Cambia el grupo al que pertenece un archivo o directorio. Con los parámetros se puede ver como se van procesando los ficheros, elegir la recursividad, informar cambios, entre otros.

c. El modo en octal es un número en base 8 que especifique el permiso. Los números en octal se especifican empezando el número por un 0. Por ejemplo, 0777 indica todos los permisos posibles para todos los tipos de usuario. 0666 indica que se dan permisos de lectura y escritura, pero no de ejecución. 0766 indica que se dan permisos de lectura y escritura, pero sólo tienen permiso de ejecución para los usuarios que son dueños del archivo. 0755 indica permisos para

lectura y ejecución, pero escritura sólo para el usuario que es dueño del archivo, y así con más combinaciones de permisos.

d. Depende de los permisos exactos que el usuario tenga sobre el directorio. Si el usuario tiene permiso para entrar al directorio, puede acceder a los archivos dentro de él con los nombres de los archivos, sin embargo no puede listarlos si no tiene permisos de lectura. Si el usuario no tiene permiso para entrar al directorio donde se encuentran los archivos, no los puede leer de ninguna forma.

e. El full path marca la ubicación del directorio o archivo independientemente del directorio donde se esté trabajando en el momento. Su nombre viene de que incluye la dirección completa del archivo/directorio buscado. Siempre empiezan en la misma dirección: la root. Los full path traen consigo todos los detalles necesarios para localizar lo buscado. Para escribirlo se necesita usar / al principio, ya que esto representa el directorio del root.

Los relative path especifican la ubicación de un directorio o archivo en base al directorio donde estamos trabajando en el momento. Solo incluyen la información de la ubicación buscada a partir del directorio actual, sin la necesidad de un full path. En términos básicos, los full path son relativos a la posición en la que se encuentra el usuario al momento de la búsqueda.

Por ejemplo, si estamos buscando el archivo llamado ‘prueba.txt’ dentro de la carpeta Facultad, y el directorio actual es Documentos.

El absolute path sería:

/home/iso/Documentos/Facultad/prueba.txt

El relative path sería:

/Facultad/prueba.txt

f. Para saber en que directorio me encuentro actualmente, se utiliza el comando pwd. Se puede acceder a un directorio personal sin necesidad de escribirlo por completo de 2 formas diferentes:

La primera es con el comando cd, y se escriben los primeros caracteres distintivos del directorio al cual quiere acceder. Una vez hecho esto, se apreta tab y el sistema autorellenerá el resto del director, ahorrando la necesidad de escribir el resto.

Por ejemplo, si necesito ir a /home/iso/Documentos/Facultad/ y me encuentro en /home, puedo ejecutar el comando cd /i y apretar tab (suponiendo que en /home no hay ningún otro directorio o archivo que empiece con i),Doc y apretar tab (suponiendo lo mismo adaptado a este directorio) y escribir Fac (misma suposición) y apretar tab nuevamente. Ejecuta el comando y se va a encontrar en esa carpeta específica y no fue necesario escribir el absolute path.

La otra es también usando el comando cd, pero esto no se puede en todos los casos. Cuando se quiere ir a un directorio inmediatamente anterior al cual está parado, puede ingresar cd .., y ascenderá un nivel

de directorio. Esto se puede anidar, realizando cd../../ , las veces que sea necesario y válido.

El atajo cd ~ vuelve al directorio de login independientemente de donde esté ubicado.

Siguiendo con el ejemplo anterior, si me encuentro en /home/iso/Documentos/Facultad y quiero ir a /home, puedo ejecutar el comando cd/../../.. , sin escribir el full path name y ejecutar el comando.

g.

- **i.** Modifica el directorio de trabajo de la shell. Con los parámetros se puede decidir si seguir o no enlaces simbólicos.
- **ii.** Desmonta sistemas de ficheros. Con los parámetros se puede elegir si desmontar todos los sistemas de ficheros, desmontar a la fuerza, mostrar lo que está haciendo, entre otros.
- **iii.** Crea directorios. Con los parámetros se pueden establecer permisos, crear directorios padres, y mostrar mensajes por cada directorio creado.
- **iv.** Muestra una lista con los tamaños que ocupa cada directorio de forma recursiva. Con los parámetros se pueden mostrar resultados para todos los ficheros (no solo los directorios), producir un total, cambiar la unidad de tamaño, entre otros.
- **v.** Borra un directorio si está vacío. Con los parámetros se pueden borrar a los padres de ese directorio, ignorar fallos si es que se encuentra vacío y se puede hacer un seguimiento de la eliminación.
- **vi.** Muestra información sobre el sistema de ficheros en el que reside cada fichero, o por omisión sobre todos los sistemas de ficheros. Con los parámetros se puede acceder a archivos semi-inaccesibles, mostrar información de nodos, cambiar la forma en que se imprimen los tamaños, entre otros.
- **vii.** Monta un sistema de ficheros. Con los parámetros se pueden hacer pruebas de este comando, mostrar las etiquetas de los sistemas de ficheros, montar el sistema de ficheros para sólo lectura, entre otros.
- **viii.** Crea un link simbólico a un archivo. Con los parámetros se puede crear una copia de seguridad de cada fichero de destino que ya exista, borrar los ficheros destinos que ya existan, preguntar si se borran los destinos, entre otros.
- **ix.** Muestra información acerca de los ficheros (del directorio actual por defecto) ordenado alfabéticamente si no se indica lo contrario. Con los parámetros se puede cambiar el orden de muestra, clasificar, mostrar el tamaño de cada bloque, filtrar los elementos, entre otros.

x. Muestra el nombre del directorio de trabajo actual. Con los parámetros se puede mostrar el directorio físico sin enlaces simbólicos.

xi. Copia origen a destino, o varios orígenes a directorio. Con los parámetros se pueden hacer copias de seguridad, copiar el contenido de los ficheros especiales, crear enlaces duros de los ficheros en vez de copiarlos, entre otros.

xii. Renombra origen a destino, o mueve orígenes a directorio. Con los parámetros se puede preguntar antes de sobreescribir, reemplazar sufijos, ver lo que se está procesando, entre otros.

5.

a. Un proceso es un programa en ejecución. Es dinámico, tiene program counter y su ciclo de vida comprende desde que se lo “dispara” hasta que termina. Si se tiene sólo una CPU, sólo uno se encontrará activo en cualquier instante. Es una entidad de abstracción, para poder ejecutarse necesita una sección de código (texto), una sección de datos (variables globales) y stacks (datos temporales como parámetros, variables temporales y direcciones de retorno).
Las siglas PID significan Process Identification, es un identificador numérico único del proceso.
Las siglas PPID significan Process Parent ID, y es el identificador del proceso padre.
Todos los procesos que se ejecutan en nuestro sistema GNU/Linux poseen estos identificadores, que son necesarios para distinguir a los procesos en memoria.

b. Para ver qué procesos están en ejecución en el sistema operativo, se utiliza el comando ps.
Con el parámetro -ef se puede ver todos los procesos llevándose a cabo en el momento, con el parámetro -fu username se pueden observar todos los procesos que está ejecutando un usuario, y también se pueden filtrar algunas de las columnas con parámetros como -user,-fname, entre otros.

c. Cuando un proceso está en ejecución sin que sea mostrado en la terminal, se dice que se está ejecutando en el background. Si se muestra la ejecución del comando dentro de la terminal se dice que está en el foreground.

d. Los comandos fg y bg permiten traer un proceso a primer plano o enviarlo a segundo plano respectivamente. Hace falta ejecutar la instrucción seguido de % y el número del proceso que obtenemos de jobs.
Si se quisiera traer el proceso 2 a primer plano, sólo hay que ejecutar fg %2. Una vez en primer plano se devuelve a segundo plano utilizando el atajo de teclado Ctrl+Z. Sin embargo al utilizar el atajo de

teclado el proceso se detiene, y esa no era la intención. Simplemente hay que utilizar bg %2 y el proceso se pondrá en marcha de nuevo.
Resumidamente:
Para traer un proceso a primer plano: se utiliza fg % processname
Para devolver el proceso a segundo plano: se utiliza el atajo de teclado Ctrl+Z
Para poner el proceso en marcha una vez que está en segundo plano: se utiliza bg %processname

e. El | (pipe) permite comunicar 2 procesos por medio de un pipe desde la shell. Conecta la salida estándar (stdout) del primer comando con la entrada estándar (stdin) del segundo. Se pueden anidar tantos pipes como se desee.
Por ejemplo: ls | more
Se ejecuta el comando ls y la salida del mismo, es enviada como entrada del comando more.

f. Las redirecciones consisten en trasladar información de un tipo a otro. Existen 2 tipos de redirecciones: destructivas y no destructivas.
Para utilizar redirecciones destructivas se utiliza >: Si el archivo de destino no existe, se lo crea. Si el archivo existe, se lo trunca y se escribe el nuevo contenido.
Para utilizar redirecciones no destructivas se utiliza >>: Si el archivo de destino no existe, se lo crea. Si el archivo existe, se agrega la información al final de éste.
ls -la ~ > (nombre del archivo)
Si realizamos la ejecución de esa forma, el contenido de nuestro archivo será reemplazado, cada vez por la salida del comando ya que se está utilizando una redirección destructiva.
Si lo que quisiéramos es que se adicione dicha salida en el archivo, se tendría que usar una redirección no destructiva, y entonces la ejecución sería de la siguiente manera:
ls -la ~ >> (nombre del archivo)

g. Es un comando utilizado para enviar mensajes sencillos a los procesos ejecutándose en el sistema. Por defecto, el mensaje que se envía es la señal de terminación (SIGTERM), que solicita al proceso limpiar su estado y salir.
Por ejemplo, si se quiere matar al proceso con el PID 1543, se tiene que ejecutar el comando kill 1543

h. Todos estos comandos son los que interactúan de forma directa o indirecta con los procesos de Linux, aunque todos actúan de diferente forma.

   i. Muestra información sobre una selección de procesos activos. Con los parámetros se puede ver en forma de árbol, información de seguridad y ver todos los procesos que el usuario root está ejecutando, entre otros

ii. Manda una señal a un proceso. Con los parámetros se mandan las señales que se quieran mandar, como también ver todas las señales posibles.
iii. Muestra un árbol de los procesos que se están ejecutando. Con los parámetros se pueden usar caracteres ASCII, se puede desactivar la compresión de sub-árboles idénticos, agregar colores, etc.
iv. Manda una señal a todos los procesos ejecutando cualquiera de los comandos especificados. Con los parámetros se puede matar a un grupo de procesos, pedir una confirmación antes de matar, ver todas las señales posibles, entre otros.
v. Muestra una vista dinámica en tiempo real de los procesos del sistema. Con los parámetros se puede establecer un tiempo de demora entre las pantallas que van a aparecer, establecer un número máximo de iteraciones que va a hacer el proceso, mostrar los procesos de un usuario en específico, entre otros.
vi. Ejecuta un programa con la prioridad de planificación modificada. Con los parámetros se le puede asignar el valor N a la prioridad del proceso, que por defecto es 10.

6.

a. Empaquetar archivos en GNU/Linux significa unir varios archivos en uno solo.
b. Cuando se empaquetan 4 archivos, la característica que se nota es que la suma del tamaño de los 4 archivos por separado es menor al tamaño del archivo empaquetado, es decir el empaquetado ocupa más espacio que los archivos juntos.
c. Si se quisiese empaquetar 4 archivos en uno solo, se tiene que runnear el comando:
tar -cvf empaquetado.tar comprimir1.txt comprimir2.txt comprimir3.txt comprimir4.txt
Así, se crea un archivo .tar gracias al comando tar que almacena el contenido de los comprimirX.txt (1<=X<=4). El parámetro c significa comprimir en 1 y el parámetro v ver el proceso. El último parámetro siempre es f.
d. Sí, se puede. En el ejemplo anterior está hecho de esa forma. También se puede empaquetar utilizando el mismo comando muchas veces.
e.
i. Guarda múltiples archivos en uno solo y puede restaurar archivos individuales desde ese mismo archivo.
ii. Busca patrones en cada archivo.
iii. Comprime o expande ficheros.
iv. Busca una expresión regular en ficheros posiblemente comprimidos.

v. Imprime el número de líneas, de palabras y de bytes de un archivo.

7.

a. Lista el contenido del directorio donde está situado y lo guarda en el archivo ‘prueba’.
b. Guarda la información de los procesos del directorio donde está situado y lo guarda en el archivo ‘PRUEBA’.
c. Cambia los permisos del archivo ‘prueba’. Al propietario (primer número del parámetro) le otorgará todos los permisos (eso significa el 7), al grupo del usuario (segundo número del parámetro) le va a otorgar el permiso de sólo ejecutar (eso significa el 1), a los otros usuarios (último número del parámetro) no le va a otorgar ningún permiso (eso significa el 0).
d. Cambia el propietario y grupo del archivo ‘PRUEBA’ a root.
e. Cambia los permisos del archivo ‘prueba’. Tanto al propietario, al grupo del usuario y al resto de los usuarios le otorgará todos los permisos.
f. Modifica la contraseña del usuario root. Este comando no se puede ejecutar si no es usuario root.
g. Elimina el archivo ‘PRUEBA’.
h. Abre el manual de /etc/shadow. Este comando no se puede ejecutar si no es usuario root.
i. Busca todos los archivos de tipo .conf y muestra los directorios. Muchos de estos archivos tendrán el permiso denegado.
j. Modifica el usuario root, y le cambia el directorio de inicio a /home/newroot.
k. Se posiciona en el directorio /root. Este comando no se puede ejecutar si no es usuario root.
l. Elimina todos los archivos del directorio actual.
m. Se posiciona en el directorio /etc.
n. Copia todos los archivos y subdirectorios de /home.
o. Apaga el sistema.

8.

a. kill 23
b. kill init. No está permitido.
c. find / −name* .conf
d. ps > /home/iso/procesos
e. chmod 751 /home/iso/xxxx
f. chmod 650 /home/iso/yyyy
g. rm /tmp
h. chown iso2010 : iso2010 /opt/isodata
i. pwd >> /home/iso/donde

9.

a. su root
b. useradd jdeUrquiza

passwd iso

c. Fue modificado el archivo /etc/passwd y /etc/groups ya que ahí se lista la información del nuevo usuario, y se crea el directorio de inicio del usuario.

d. mkdir /tmp/cursada2017

e. cp /var/log /tmp/cursada2017

f. chown jdeUrquiza : users /tmp/cursada2017

g. chmod -R 723 /tmp/cursada2017

h. gnome-terminal
su - jdeUrquiza

i. hostname

j. ps r

k. w

l. sudo
write jdeUrquiza < “Se va a apagar el sistema”>

m. shutdown

10.

a. mkdir 19497/8
cd 19497/8

b. vi LEAME
ESC + a : Juan Cruz de Urquiza
ESC + o
ESC + a: 19497/8
ESC + o
ESC + a: juancruzdeu@gmail.com
ESC + : w
ESC + : q

c. chmod 017 LEAME

d. cd /etc
touch /home/iso leame
ls > leame
Se pudo realizar ya que GNU/Linux es case sensitive y diferencia las mayúsculas de las minúsculas.

e. find <nombre del archivo>
find <*característica similar>
Este comando hace que se busquen archivos dentro del filesystem con su nombre. Por ejemplo, si quiero buscar todos los archivos .txt del directorio donde estoy parado ejecuto: find *.txt

f. find *.so > | tee /home/iso/19797/8/ejercicio_f

11.

i. Crea un directorio llamado iso en el directorio actual.

ii. Accede a la carpeta iso y guarda los procesos activos en un nuevo archivo llamado f0.

iii. Guarda la lista de los directorios y archivos de iso en un nuevo archivo llamado f1.
iv. Se posiciona en /.
v. Imprime la variable $HOME que tiene el directorio personal del usuario, en este caso /home/iso
vi. Guarda el listado detallado del directorio / en un nuevo archivo llamado ls en la carpeta recientemente creada iso.
vii. Se posiciona en el directorio personal y crea el directorio f2.
viii. Lista en formato de listado largo los directorios de f2, no sus contenidos.
ix. Cambia los permisos del directorio f2. Al usuario actual lo deja escribir y ejecutar, al grupo del usuario actual solo leer y al resto de los usuarios sólo ejecutar.
x. Crea un archivo llamado dir.
xi. Se posiciona en el directorio f2.
xii. Vuelve a /home/iso.
xiii. Guarda en un nuevo archivo llamado f3 el directorio actual donde estoy parado.
xiv. Crea un directorio llamado f2 en el directorio personal donde allí creó un archivo llamado f3 que almacena la cantidad de archivos que almacenen el texto ps.
xv. Cambia los permisos del directorio f2. Al usuario actual lo deja escribir, ejecutar y leer mientras que al grupo y a los otros usuarios no les deja hacer nada. Luego se mueve al directorio personal.
xvi. Busca los archivos que contengan en su nombre "." en el directorio etc/passwd y devuelve su ubicación.
xvii. Busca los archivos que contengan en su nombre "/" en el directorio etc/passwd y devuelve su ubicación.
xviii. Crea un directorio llamado ejercicio5 en el directorio actual.

b. *Resuelto en máquina.*

c. 19: cp -r iso/ Documentos/Facultad/19497
20: cp -r ejercicio5/ Documentos/Facultad/19497
cp -r f2/ Documentos/Facultad/19497
cp dir Documentos/Facultad/19497

d. *Resuelto en máquina.*

12.

i. mv /home/iso/dir1/f3 /home/iso
ii. cp /home/iso/dir2/f4 /home/iso/dir1/dir11
iii. mv /home/iso/dir1/f3 /home/iso/f7
cp /home/iso/dir2/f4 home/iso/dir1/dir11/f7
iv. mkdir /home/iso/copia ; cp -r /home/iso/dir1 /home/iso/copia
v. mv /home/iso/f1 /home/iso/archivo ; ls -l /home/iso/archivo
vi. chmod 617 archivo

vii. mv /home/iso/f7 /home/iso/dir1/f3.exe ; mv /home/iso/dir2/f4 /home/iso/dir2/f4.exe

viii. chmod 023 /home/iso/dir1/f3.exe /home/iso/dir2/f3.exe

13.

a. cd / ; mkdir logs

b. cp /var/log /tmp/logs

c. tar -cvf misLogs.tar /tmp/logs

d. tar -cvzf misLogs.tar.zip /tmp/logs

e. cp /var/log/misLogs.tar /home/iso ; cp /var/log/misLogs.tar.zip /home/iso

f. rm -R /tmp/logs

g. tar -xvf /home/iso/misLogs.tar > /home/iso/primero ; tar -xvf /home/iso/misLogs.tar.zip > /home/iso/segundo