

Introducción a los Operativos/Conceptos

operativos

Práctica 2

Sistemas

de sistemas

**Objetivo**

El objetivo de esta práctica es que el alumno comprenda los aspectos principales acerca de la es- tructura del sistema Operativo GNU/Linux en lo que respecta a procesos, usuarios, filesystems, permisos, etc.

1. Editor de textos:
   1. Nombre al menos 3 editores de texto que puede utilizar desde la línea de comandos.
   2. ¿En qué se diferencia un editor de texto de los comandos cat, more o less? Enumere los modos de operación que posee el editor de textos vi.
   3. Nombre los comandos más comunes que se le pueden enviar al editor de textos vi.
2. VI, NANO y EMACS
3. CAT: se utiliza para mostrar el contenido de un archivo de texto en la terminal.

MORE: se utiliza para mostrar el contenido de un archivo de texto de manera paginada,

LESS: Igual que more, pero permite el desplazamiento para arriba y para abajo

Modos de operación de Vi:

1. **Modo Normal:** Este es el modo principal al abrir Vi. En este modo, puedes navegar por el texto, buscar, copiar, pegar, eliminar y realizar otras acciones sin editar directamente el texto.
2. **Modo de Inserción:** En este modo, puedes insertar y editar el texto directamente en el archivo.
3. **Modo de Comando:** Similar al Modo Normal, pero se utiliza para ingresar comandos específicos de Vi, como guardar el archivo, buscar y reemplazar, entre otros.

* C) **i:** Entra en el Modo de Inserción antes del cursor.
* **a:** Entra en el Modo de Inserción después del cursor.
* **dd:** Borra la línea actual.
* **yy:** Copia la línea actual.
* **p:** Pega el contenido copiado o cortado.
* **/texto:** Realiza una búsqueda hacia adelante de "texto".
* **?texto:** Realiza una búsqueda hacia atrás de "texto".
* **:w:** Guarda el archivo.
* **:q:** Sale de Vi (si no se han realizado cambios).
* **:q!:** Sale de Vi sin guardar cambios.
* **:wq:** Guarda el archivo y sale de Vi.
* **:x:** Similar a ":wq", guarda el archivo y sale de Vi.
* **:e archivo:** Abre otro archivo

1. Proceso de Arranque *SystemV* :
   1. Enumere los pasos del proceso de inicio de un sistema GNU/Linux, desde que se prende la PC hasta que se logra obtener el login en el sistema.
   2. Proceso **INIT**. ¿Quién lo ejecuta? ¿Cuál es su objetivo?
   3. Ejecute el comando pstree. ¿Qué es lo que se puede observar a partir de la ejecución de este comando?
   4. RunLevels. ¿Qué son? ¿Cuál es su objetivo?
   5. ¿A qué hace referencia cada nivel de ejecución según el estándar? ¿Dónde se define qué Runlevel ejecutar al iniciar el sistema operativo? ¿Todas las distribuciones respetan estos estándares?
   6. Archivo /etc/inittab. ¿Cuál es su finalidad? ¿Qué tipo de información se almacena en el? ¿Cuál es la estructura de la información que en él se almacena?
   7. Suponga que se encuentra en el runlevel <X>. Indique qué comando(s) ejecutaría para cambiar al runlevel <Y>. ¿Este cambio es permanente? ¿Por qué?
   8. Scripts RC. ¿Cuál es su finalidad? ¿Dónde se almacenan? Cuando un sistema GNU/Li- nux arranca o se detiene se ejecutan scripts, indique cómo determina qué script ejecutar ante cada acción. ¿Existe un orden para llamarlos? Justifique.
   9. ¿Qué es insserv? ¿Para qué se utiliza? ¿Qué ventajas provee respecto de un arranque tradicional?
   10. ¿Cómo maneja Upstart el proceso de arranque del sistema?
   11. Cite las principales diferencias entre SystemV y Upstart.
   12. Qué reemplaza a los scripts *rc* de *SystemV* en *Upstart*? ¿En que ubicación del filesystem se encuentran?

A)

1. Se ejecuta el código de la BIOS.
2. El hardware lee el sector de arranque.
3. Se carga el gestor de arranque.
4. Se carga el kernel.

B) El proceso INIT es ejecutado por el kernel del S.O. durante el proceso de arranque.

Su función es cargar todos los subprocesos necesarios para el correcto funcionamiento del SO

C) Cuando ejecutas **pstree**, obtienes una representación gráfica de los procesos que muestra la relación entre ellos en forma de un árbol.

D) Es el modo en que arranca Linux (3 en Redhat, 2 en Debian)

 El proceso de arranque lo dividimos en niveles

 Cada uno es responsable de levantar (iniciar) o bajar (parar) una serie de servicios

E) Según el estándar:

 0: halt (parada)

 1: single user mode (monousuario)

 2: multiuser, without NFS (modo multiusuario sin soperte de red)

 3: full multiuser mode console (modo multiusuario completo por consola)

 4: no se utiliza

 5: X11 (modo multiusuario completo con login grafico basado en X)

6:reboot

 Se encuentran definidos en /etc/inittab

**No todas las distribuciones lo respetan**

F) Su finalidad es almacenar la información del nivel de ejecución del arranque (id:nivelesEjecuci´on:acci´on:proceso)

Información:

 Id: identifica la entrada en inittab (1 a 4 car´acteres)

 NivelesEjecuci´on: el/los niveles de ejecuci´on 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 ejecutar´a cuando init reciba la se˜nal SIGINT

off, respawn, once, sysinit, boot, bootwait, powerwait, etc.

 Proceso: el proceso exacto que será ejecutado

G) sudo init Y

Es importante destacar que estos cambios de runlevel generalmente no son permanentes.

Porque no se esta modificando el inicion del sistema

H) Su finalidad es configurar y controlar la ejecución de servicios y otros procesos durante el proceso de inicio (arranque) del sistema operativo

Los scripts que se ejecutan están en /etc/init.d

Formato de los links: [S|K] <orden><nombresript>

 S: lanza el script con el argument start  K: lanza el script con el argument stop

Sí, en la mayoría de los sistemas Unix y Unix-like, existe un orden para llamar a los scripts de inicio, y esto es importante para garantizar que los servicios se inicialicen correctamente y que las dependencias entre ellos se manejen adecuadamente.

**I)**

**insserv** es una herramienta utilizada en sistemas Linux que se utiliza para gestionar los scripts de inicio (init scripts) y las dependencias entre servicios durante el proceso de arranque del sistema.

Se utiliza para administrar el orden de los enlaces simbólicos del /etc/rcX.d, resolviendo las dependencias de forma automática

VENTAJAS:

1. Gestión de dependencias automatizada.
2. Compatibilidad con arranque paralelo.
3. Mayor flexibilidad.
4. Facilita la eliminación de servicios obsoletos.
5. Mejora la legibilidad de los scripts de inicio.

J,K) Upstart fue el primer reemplazo propuesto para SystemV (Ubuntu, Fedora, Debian, etc.)

 Permite la ejecución de trabajos en forma asincrónica a través de eventos (event-based) como **principal diferencia con sysVinit** que es estrictamente sincrónico (dependency-based)

 Estos trabajos se denominan jobs

 El principal objetivo de un job es definir servicios o tareas a ser ejecutadas por init

 Son scripts de texto plano que definene las acciones/tareas (unidad de trabajo) a ejecutar ante determinados eventos

 Cada job es definido en el /etc/init (.conf)

 Suelen ser de dos tipos:

Task: ejecuci´on finita (task) → not respawning → exit 0 o uso de stop

Service: ejecuci´on indeterminada → respawning

1. K) **Modelo de inicio:**
   * **SystemV:** SystemV init utiliza un enfoque secuencial basado en scripts almacenados en el directorio **/etc/init.d/**. Los scripts se ejecutan en un orden predefinido según su número de secuencia y nivel de ejecución.
   * **Upstart:** Upstart utiliza un enfoque basado en eventos y tareas. Las tareas se configuran en archivos **.conf** ubicados en el directorio **/etc/init/** y se ejecutan en respuesta a eventos específicos del sistema.
2. **Gestión de dependencias:**
   * **SystemV:** La gestión de dependencias en SystemV se realiza manualmente mediante comentarios en los scripts de inicio (**# Required-Start**, **# Required-Stop**, etc.). Esto puede ser propenso a errores y no siempre es eficiente.
   * **Upstart:** Upstart maneja automáticamente las dependencias entre tareas. Puede iniciar tareas en paralelo cuando no tienen dependencias entre sí, lo que mejora la eficiencia del arranque.
3. **Flexibilidad:**
   * **SystemV:** SystemV init es menos flexible en comparación con Upstart. Requiere configuración adicional para gestionar eventos específicos y dependencias complejas.
   * **Upstart:** Upstart es más flexible y permite definir fácilmente eventos y tareas para adaptarse a las necesidades del sistema. Es especialmente eficaz en la gestión de eventos y servicios de manera receptiva.
4. **Compatibilidad con arranque paralelo:**
   * **SystemV:** SystemV init no es nativamente compatible con el arranque paralelo, lo que significa que los servicios se inician en secuencia, uno después del otro, lo que puede aumentar el tiempo de arranque.
   * **Upstart:** Upstart está diseñado para admitir el arranque paralelo siempre que no haya dependencias entre tareas, lo que puede acelerar significativamente el proceso de arranque.
5. **Migración y estandarización:**
   * **SystemV:** SystemV init ha sido ampliamente utilizado en muchas distribuciones de Linux, pero su falta de un estándar bien definido llevó a la diversidad de implementaciones.
   * **Upstart:** Upstart fue adoptado por algunas distribuciones, como Ubuntu, pero no logró una estandarización completa en el ecosistema de Linux. Muchas distribuciones han migrado a Systemd como sistema de inicio predeterminado debido a la necesidad de un estándar común.

L) En Upstart, los scripts RC del estilo SystemV, ubicados en **/etc/init.d/**, son reemplazados por archivos de configuración con extensión **.conf**. Estos archivos **.conf** se encuentran en el directorio **/etc/init/** de sistemas que utilizan Upstart

1. Dado el siguiente *job* de upstart perteneciente al servicio de base de datos del mysql indique a qué hace referencia cada línea del mismo:

# MySQL Se r v i c e

d e s c r i p t i o n "MySQL Server " author " i n f o autor "

s t a r t on ( net device up

*− −*

and l o c a l f i l e s y s t e m s and r u n l e v e l [ 2 3 4 5 ] )

*−*

stop on r u n l e v e l [ 0 1 6 ]

[ . . . ]

exec / usr / sbin / mysqld [ . . . ]

1. ¿Qué es *sytemd*?

(ñ) ¿A qué hace referencia el concepto de activación de socket en systemd?

1. ¿A qué hace referencia el concepto de *cgroup*?

# MySQL Se r v i c **e ;Comentario**

d e s c r i p t i o n "MySQL Server "

author " i n f o autor **" ;Indica Autor**

s t a r t on ( net device up

*− −*

and l o c a l f i l e s y s t e m s and r u n l e v e l [ 2 3 4 5 ] ) **;Indica que inicia cuando la red y los archivos estén listos en un runlevel 2,3,4,5**

*−*

stop on r u n l e v e l [ 0 1 6 ] **;Indica que se dentendra en runlevel 0,1,6**

exec / usr / sbin / mysqld **;Indica el archivo a ejecutar**

B)  No todos los servicios que se inician en el booteo se utilizan: impresora,servidor, ETC.

 Es un mecanismo de iniciación bajo demanda → podemos ofrecer una variedad de servicios sin que realmente esten iniciados

 Cuando el sockect recibe una conexión spawnea el servicio y le pasa el socket

 No hay necesidad de definir dependencias entre servicios → se inician todos los sockets en primer medida

C) Permite organizar un grupo de procesos en forma jerárquica

 Agrupa conjunto de procesos relacionados (por ejemplo, un servidor web Apache con sus dependientes)

 Tareas que realiza: Tracking mediante subsistema cgroups → no se utiliza el PID → doble fork no funciona para escapar de systemd , Limitar el uso de recursos, etc

1. Usuarios:
   1. ¿Qué archivos son utilizados en un sistema GNU/Linux para guardar la información de los usuarios?
   2. ¿A qué hacen referencia las siglas *UID* y *GID*? ¿Pueden coexistir UIDs iguales en un sistema GNU/Linux? Justifique.
   3. ¿Qué es el usuario root? ¿Puede existir más de un usuario con este perfil en GNU/Li- nux? ¿Cuál es la *UID* del *root*?.
   4. Agregue un nuevo usuario llamado iso2017 a su instalación de GNU/Linux, especifique que su home sea creada en /home/iso\_2017, y hágalo miembro del grupo catedra (si no existe, deberá crearlo). Luego, sin iniciar sesión como este usuario cree un archivo en su home personal que le pertenezca. Luego de todo esto, borre el usuario y verifique que no queden registros de él en los archivos de información de los usuarios y grupos.

e)Investigue la funcionalidad y parámetros de los siguientes comandos:

**useradd y adduser:** Ambos comandos se utilizan para crear nuevos usuarios en un sistema Linux. Sin embargo, **adduser** es una interfaz de usuario más amigable y simplificada para agregar usuarios, mientras que **useradd** proporciona opciones más avanzadas y personalizadas para la creación de usuarios. Algunos parámetros comunes incluyen:

* + **-m**: Crea el directorio de inicio del usuario.
  + **-g**: Especifica el grupo principal del usuario.
  + **-G**: Define los grupos secundarios a los que pertenece el usuario.
  + **-s**: Establece la shell predeterminada para el usuario.
  + **-c**: Agrega un comentario o información sobre el usuario.

1. **usermod:** Este comando se utiliza para modificar las propiedades de un usuario existente en el sistema. Algunos parámetros comunes incluyen:
   * **-l**: Cambia el nombre de inicio del usuario.
   * **-d**: Cambia el directorio de inicio del usuario.
   * **-G**: Agrega o quita el usuario de grupos secundarios.
   * **-aG**: Agrega el usuario a un grupo secundario sin quitarlo de otros grupos.
   * **-e**: Establece la fecha de vencimiento de la cuenta del usuario.
2. **userdel:** Este comando se utiliza para eliminar usuarios del sistema. Por lo general, no se utiliza para eliminar usuarios por completo, ya que podría dejar archivos huérfanos en el sistema. Se usa con precaución.
3. **su:** El comando **su** (Switch User) se utiliza para cambiar de usuario en una terminal. Normalmente, se usa **su** seguido del nombre de usuario al que se desea cambiar. Puede ser útil para ejecutar comandos como otro usuario, incluido el usuario root.
4. **groupadd y groupdel:** Estos comandos se utilizan para crear y eliminar grupos de usuarios en el sistema, respectivamente. **groupadd** se utiliza para agregar grupos y **groupdel** se utiliza para eliminarlos.
5. **passwd:** El comando **passwd** se utiliza para cambiar la contraseña de un usuario en el sistema. Puedes usarlo para cambiar tu propia contraseña o, como superusuario, para cambiar la contraseña de otros usuarios.
6. **who:** El comando **who** muestra información sobre quién está conectado al sistema en ese momento, incluyendo sus nombres de usuario y tiempos de inicio de sesión.

A) Se almacena en /etc (Password, Group, Shadow)

B) **UID (User Identifier)**, **GID (Group Identifier), No pueden exisir 2 usuarios iguales, ya que no se sabría quien seria**

C) El usuario root es el superusuario (El admin), no es posible quee existan varios usuarios con estos pemisos, El superusuario (root) generalmente tiene un UID de 0,

4.Archivos:

A)¿Cómo son definidos los permisos sobre archivos en un sistema GNU/Linux?

B,C,D)Investigue la funcionalidad y parámetros de los siguientes comandos relacionados con los permisos en GNU/Linux:

chmod chown chgrp

* 1. Al utilizar el comando chmod generalmente se utiliza una notación octal asociada para definir permisos. ¿Qué significa esto? ¿A qué hace referencia cada valor?
  2. ¿Existe la posibilidad de que algún usuario del sistema pueda acceder a determinado archivo para el cual no posee permisos? Nombrelo, y realice las pruebas correspondien- tes.
  3. Explique los conceptos de “full path name” y “relative path name”. De ejemplos claros de cada uno de ellos.
  4. ¿Con qué comando puede determinar en qué directorio se encuentra actualmente?

¿Existe alguna forma de ingresar a su directorio personal sin necesidad de escribir todo el path completo? ¿Podría utilizar la misma idea para acceder a otros directorios?

¿Cómo? Explique con un ejemplo.

* 1. Investigue la funcionalidad y parámetros de los siguientes comandos relacionados con el uso del FileSystem:

1. **cd (Change Directory):**
   * **Funcionalidad:** Se utiliza para cambiar el directorio actual en el que te encuentras.
   * **Parámetros:** No tiene parámetros adicionales. Simplemente proporciona la ruta al directorio al que deseas cambiar.
   * **Ejemplo:** **cd /ruta/a/directorio**
2. **umount (Unmount):**
   * **Funcionalidad:** Se utiliza para desmontar sistemas de archivos previamente montados en puntos de montaje.
   * **Parámetros:** Puedes especificar el dispositivo de almacenamiento o el punto de montaje a desmontar.
   * **Ejemplo:** **umount /mnt/punto\_de\_montaje**
3. **mkdir (Make Directory):**
   * **Funcionalidad:** Crea un nuevo directorio en la ubicación especificada.
   * **Parámetros:** Puedes proporcionar opciones para establecer permisos u otros atributos del directorio.
   * **Ejemplo:** **mkdir nuevo\_directorio**
4. **du (Disk Usage):**
   * **Funcionalidad:** Muestra el uso del espacio en disco de archivos y directorios en una ubicación específica.
   * **Parámetros:** Puede incluir opciones para personalizar la salida, como **-h** para mostrar tamaños en formato legible por humanos.
   * **Ejemplo:** **du -h /ruta/a/directorio**
5. **rmdir (Remove Directory):**
   * **Funcionalidad:** Elimina un directorio vacío.
   * **Parámetros:** Puedes especificar el directorio a eliminar.
   * **Ejemplo:** **rmdir directorio\_vacio**
6. **df (Disk Free):**
   * **Funcionalidad:** Muestra información sobre el espacio en disco disponible en sistemas de archivos montados.
   * **Parámetros:** Puede incluir opciones para personalizar la salida, como **-h** para mostrar tamaños en formato legible por humanos.
   * **Ejemplo:** **df -h**
7. **mount (Mount):**
   * **Funcionalidad:** Monta sistemas de archivos en puntos de montaje especificados.
   * **Parámetros:** Debes proporcionar la ruta al dispositivo de almacenamiento y el punto de montaje.
   * **Ejemplo:** **mount /dev/sda1 /mnt/punto\_de\_montaje**
8. **ln (Link):**
   * **Funcionalidad:** Crea enlaces (vinculaciones) a archivos o directorios. Hay dos tipos principales: enlaces simbólicos (**ln -s**) y enlaces duros (**ln** sin **-s**).
   * **Parámetros:** Depende de si deseas crear un enlace simbólico o duro y especificas la ruta al archivo o directorio original y la ubicación del nuevo enlace.
   * **Ejemplo de enlace simbólico:** **ln -s /ruta/al/archivo /ruta/del/enlace**
   * **Ejemplo de enlace duro:** **ln /ruta/al/archivo /ruta/del/enlace**
9. **ls (List):**
   * **Funcionalidad:** Muestra una lista de archivos y directorios en un directorio específico.
   * **Parámetros:** Puede incluir opciones para personalizar la salida, como **-l** para mostrar información detallada y **-a** para mostrar archivos ocultos.
   * **Ejemplo:** **ls -l /ruta/a/directorio**
10. **pwd (Print Working Directory):**
    * **Funcionalidad:** Muestra la ruta completa del directorio actual en el que te encuentras.
    * **Parámetros:** No tiene parámetros adicionales.
    * **Ejemplo:** **pwd**
11. **cp (Copy):**
    * **Funcionalidad:** Copia archivos o directorios de una ubicación a otra.
    * **Parámetros:** Debes proporcionar la ubicación de origen y destino.
    * **Ejemplo:** **cp archivo\_origen /ruta/de/destino**
12. **mv (Move):**
    * **Funcionalidad:** Mueve (cambia el nombre) de archivos o directorios de una ubicación a otra. También se utiliza para renombrar archivos y directorios.
    * **Parámetros:** Debes proporcionar la ubicación de origen y destino.
    * **Ejemplo:** **mv archivo\_origen /ruta/de/destino**

A)

1. **Propietario (Owner):** El propietario del archivo es el usuario que creó el archivo o el usuario que tiene la capacidad de cambiar sus permisos y contenido. Cada archivo tiene un único propietario.
2. **Grupo (Group):** Un grupo es un conjunto de usuarios en el sistema. Los permisos de grupo se aplican a todos los usuarios que pertenecen a ese grupo. Cada archivo está asociado a un grupo y los usuarios que pertenecen a ese grupo tienen ciertos derechos sobre el archivo.
3. **Otros (Others):** "Otros" se refiere a cualquier usuario que no sea el propietario del archivo ni pertenezca al grupo asociado al archivo. Los permisos de "otros" se aplican a todos los demás usuarios en el sistema que no son el propietario ni pertenecen al grupo.

Los permisos en un archivo se definen mediante una combinación de tres tipos de acceso (lectura, escritura y ejecución) para cada uno de los tres componentes mencionados anteriormente (propietario, grupo y otros). Estos permisos se representan en notación de tres caracteres (r, w y x) y se organizan en el siguiente orden:

* El primer carácter representa los permisos del propietario.
* El segundo carácter representa los permisos del grupo.
* El tercer carácter representa los permisos de otros.

Cada carácter puede tener uno de los siguientes valores:

* **r**: Permite la lectura del archivo o directorio.
* **w**: Permite la escritura en el archivo o directorio.
* **x**: Permite la ejecución del archivo o el acceso al directorio.
* **-**: Indica que el permiso correspondiente está deshabilitado.

Por EJ **-rwxr-xr—**

**B)**

1. **chmod (Change Mode):**
   * **Funcionalidad:** El comando **chmod** se utiliza para cambiar los permisos (modo) de archivos y directorios en sistemas GNU/Linux. Permite modificar quién puede leer (r), escribir (w) y ejecutar (x) un archivo o directorio.
   * **Parámetros comunes:**
     + **u** (User): Modifica los permisos para el propietario del archivo.
     + **g** (Group): Modifica los permisos para el grupo al que pertenece el archivo.
     + **o** (Others): Modifica los permisos para otros usuarios que no son el propietario ni están en el grupo.
     + **a** (All): Modifica los permisos para todos (equivalente a **ugo**).
     + **+** (Add): Agrega permisos.
     + **-** (Remove): Elimina permisos.
     + **=** (Assign): Asigna permisos específicos, reemplazando los permisos existentes.
2. **chown (Change Owner):**
   * **Funcionalidad:** El comando **chown** se utiliza para cambiar el propietario y/o el grupo de un archivo o directorio en sistemas GNU/Linux.
   * **Parámetros comunes:**
     + **usuario** (User): Especifica el nuevo propietario por nombre de usuario.
     + **:grupo** (Group): Especifica el nuevo grupo por nombre de grupo.
3. **chgrp (Change Group):**
   * **Funcionalidad:** El comando **chgrp** se utiliza específicamente para cambiar el grupo de un archivo o directorio en sistemas GNU/Linux.
   * **Parámetros comunes:**
     + **grupo** (Group): Especifica el nuevo grupo por nombre de grupo.

**E)** La notación octal es una forma común de representar permisos al utilizar el comando **chmod** en sistemas Unix y GNU/Linux. Esta notación se basa en números en base octal (base 8) y permite definir de manera concisa los permisos para un archivo o directorio. Cada dígito en la notación octal representa un conjunto específico de permisos.

En la notación octal, se utilizan tres dígitos, y cada dígito se asocia con un grupo de usuarios:

* El primer dígito (el más a la izquierda) se refiere al propietario del archivo.
* El segundo dígito se refiere al grupo al que pertenece el archivo.
* El tercer dígito se refiere a los otros usuarios en el sistema (no son el propietario ni están en el grupo).

Cada dígito se compone de una combinación de tres bits, que pueden tener uno de los siguientes valores:

* **4 (r):** Representa el permiso de lectura. Permite a un usuario ver el contenido de un archivo o listar el contenido de un directorio.
* **2 (w):** Representa el permiso de escritura. Permite a un usuario modificar un archivo o crear, eliminar y renombrar archivos en un directorio.
* **1 (x):** Representa el permiso de ejecución. Permite a un usuario ejecutar un archivo o ingresar a un directorio.

La combinación de estos valores se suma para representar los permisos. Por ejemplo, si un dígito tiene el valor 7, significa que se han otorgado permisos de lectura (4) y escritura (2) y permisos de ejecución (1), lo que permite todas las operaciones posibles (lectura, escritura y ejecución).

Ejemplos de notación octal:

* **750:** El propietario tiene permisos de lectura, escritura y ejecución (7), el grupo tiene permisos de lectura y ejecución (5), y otros no tienen ningún permiso (0).

F) Si, el que tenga permisos de superusuario

G)

* **Full Path Name (Nombre de Ruta Completa):** Especifica la ubicación completa y precisa de un archivo o directorio desde el directorio raíz del sistema de archivos, utilizando una ruta que comienza con "/". Ejemplo: **/home/usuario/proyectos/documento.txt**
* **Relative Path Name (Nombre de Ruta Relativa):** Describe la ubicación de un archivo o directorio en relación con el directorio actual del usuario o programa. Es una ruta más concisa que no comienza desde la raíz. Ejemplo: Si el usuario está en el directorio "usuario," el relative path name para "proyectos/documento.txt" se refiere al archivo "documento.txt" en el directorio "proyectos" desde el directorio actual.

H)Con pwd se ve la ruta acual, con cd ~ se va al directorio personal desde cualquier usuario, Para navegar a un directorio específico utilizando rutas relativas, simplemente proporciona la ruta relativa al directorio que deseas. Por ejemplo, si estás en **/home/usuario** y deseas ir a **/var/log**, puedes hacerlo de la siguiente manera:

cd /var/log

1. Procesos:
   1. ¿Qué es un proceso? ¿A que hacen referencia las siglas *PID* y *PPID*? ¿Todos los procesos tienen estos atributos en GNU/Linux? Justifique. Indique qué otros atributos tiene un proceso.
   2. Indique qué comandos se podrían utilizar para ver qué procesos están en ejecución en un sistema GNU/Linux.
   3. ¿Qué significa que un proceso se está ejecutando en Background? ¿Y en Foreground?
   4. ¿Cómo puedo hacer para ejecutar un proceso en Background? ¿Como puedo hacer para pasar un proceso de background a foreground y viceversa?
   5. Pipe ( **|** ). ¿Cuál es su finalidad? Cite ejemplos de su utilización.
   6. Redirección. ¿Qué tipo de redirecciones existen? ¿Cuál es su finalidad? Cite ejemplos de utilización.
   7. Comando *kill*. ¿Cuál es su funcionalidad? Cite ejemplos.
   8. Investigue la funcionalidad y parámetros de los siguientes comandos relacionados con el manejo de procesos en GNU/Linux. Además, compárelos entre ellos:
2. **ps (Process Status):**
   * **Funcionalidad:** Muestra información sobre los procesos en ejecución en el sistema.
   * **Parámetros:** Puede incluir opciones como **-e** (mostrar todos los procesos), **-f** (muestra información detallada), **-u** (filtrar por usuario) y más.
   * **Ejemplo:** **ps -ef | grep proceso**
3. **kill (Kill Process):**
   * **Funcionalidad:** Se utiliza para enviar señales a procesos específicos para detenerlos o controlar su comportamiento.
   * **Parámetros:** Debes proporcionar el PID del proceso y la señal que deseas enviar (por defecto, es la señal SIGTERM para terminar).
   * **Ejemplo:** **kill -SIGTERM PID**
4. **pstree (Process Tree):**
   * **Funcionalidad:** Muestra los procesos en una estructura de árbol que representa la jerarquía de procesos en el sistema.
   * **Parámetros:** No requiere parámetros adicionales.
   * **Ejemplo:** **pstree**
5. **killall (Kill All Processes by Name):**
   * **Funcionalidad:** Se utiliza para enviar señales a todos los procesos que coinciden con un nombre de proceso específico.
   * **Parámetros:** Debes proporcionar el nombre del proceso.
   * **Ejemplo:** **killall nombre\_del\_proceso**
6. **top:**
   * **Funcionalidad:** Muestra una lista en tiempo real de los procesos en ejecución junto con información detallada sobre el uso de la CPU y la memoria.
   * **Parámetros:** Puedes incluir opciones como **-d** (intervalo de actualización) y más.
   * **Ejemplo:** **top**
7. **nice:**
   * **Funcionalidad:** Ajusta la prioridad de ejecución de un proceso, permitiendo que los procesos se ejecuten con prioridades diferentes en el sistema.
   * **Parámetros:** Se utiliza con un valor numérico para establecer la prioridad (números negativos para prioridades más altas y positivos para prioridades más bajas).
   * **Ejemplo:** **nice -n -10 comando**

Comparación:

* **ps** y **top** son comandos para mostrar información sobre procesos en ejecución. **ps** proporciona una instantánea en el momento de su ejecución, mientras que **top** ofrece una vista en tiempo real y es interactivo.
* **kill** se utiliza para enviar señales a procesos específicos para controlar su comportamiento o detenerlos, mientras que **killall** envía señales a todos los procesos que coinciden con un nombre de proceso específico.
* **pstree** muestra la jerarquía de procesos en forma de árbol, lo que facilita la visualización de las relaciones entre los procesos.
* **nice** se utiliza para ajustar la prioridad de ejecución de un proceso, lo que puede ser útil para asignar recursos de manera eficiente en sistemas multiusuario.

A)

Un proceso en un sistema operativo, incluyendo GNU/Linux, es una instancia en ejecución de un programa o una tarea. Cada proceso tiene su propio espacio de memoria y recursos asignados, lo que le permite ejecutarse de manera independiente de otros procesos.

1. **PID (Process ID):** PID es el identificador único de un proceso. Cada proceso en el sistema tiene su propio PID que lo distingue de otros procesos. Los PID son números enteros positivos asignados por el sistema operativo y se utilizan para identificar y administrar procesos. Los PID pueden ser utilizados para enviar señales a procesos específicos, detener procesos y realizar un seguimiento de su estado.
2. **PPID (Parent Process ID):** PPID es el identificador del proceso padre del proceso actual. Indica qué proceso creó o engendró el proceso en cuestión. El proceso padre suele ser el que inició o lanzó el proceso hijo. Los PPID son útiles para establecer la jerarquía de procesos en el sistema y determinar cómo interactúan los procesos entre sí.

B)

**ps (Process Status): Muestra los procesos en ejecucion**

**top:** El comando **top** muestra una lista en tiempo real de los procesos en ejecución junto con información detallada sobre el uso de la CPU y la memoria.

**Pstree: Muestra los procesos en forma de arbol**

**pgrep:** El comando **pgrep** se utiliza para buscar procesos por su nombre o por otros criterios y muestra los PIDs de los procesos coincidentes.

C) Un proceso en "Background" se ejecuta en segundo plano sin interferir con otras tareas en primer plano y se inicia con el símbolo "&" al final del comando. Por otro lado, un proceso en "Foreground" se ejecuta en primer plano, generalmente requiere la atención y la interacción del usuario, y bloquea el terminal hasta que finaliza o se detiene manualmente.

D) Para ejecutar un proceso en background, puedes agregar el símbolo **&** al final del comando al ejecutarlo en el shell. Ejemplo: comando\_en\_background &

Para cambiar un proceso de background a foreground, utiliza el comando "fg" seguido del número de trabajo del proceso (%X). Para mover un proceso de foreground a background, suspende el proceso con Ctrl+Z y luego utiliza "bg" para que continúe en segundo plano.

E) El “|” nos permite comunicar dos procesos por medio de unpipe o tubería desde la shell

 El pipe conecta stdout (salida estándar) del primer comando con la stdin (entrada estándar) del segundo

Ej: $ ls | more

F) Al utilizar redirecciones mediante > (destructiva):

 Si el archivo de destino no existe, se lo crea

 Si el archivo existe, se lo trunca y se escribe el nuevo contenido

Al utilizar redirecciones mediante >> (no destructiva):

 Si el archivo de destino no existe, se lo crea

 Si el archivo existe, se agrega la información al final

Ej: comando\_que\_genera\_errores 2> errores.txt

G) Hecho en h

.

.

.

.

.

.

1. Otros comandos de Linux (Indique funcionalidad y parámetros):
   1. ¿A qué hace referencia el concepto de empaquetar archivos en GNU/Linux?
   2. Seleccione 4 archivos dentro de algún directorio al que tenga permiso y sume el tamaño de cada uno de estos archivos. Cree un archivo empaquetado conteniendo estos 4 archivos y compare los tamaños de los mismos. ¿Qué característica nota?
   3. ¿Qué acciones debe llevar a cabo para comprimir 4 archivos en uno solo? Indique la secuencia de comandos ejecutados.
   4. ¿Pueden comprimirse un conjunto de archivos utilizando un único comando?
   5. Investigue la funcionalidad de los siguientes comandos: **tar (Tape Archive):** Se utiliza para crear, ver, extraer o manipular archivos empaquetados, también conocidos como "archivos tar". Puede comprimir archivos opcionalmente.

**grep (Global Regular Expression Print):** Se utiliza para buscar patrones de texto en archivos o en la salida de otros comandos.

**gzip (GNU Zip):** Se utiliza para comprimir archivos utilizando el formato de compresión GZIP.

**zgrep (Zipped Grep):** Similar a **grep**, pero se utiliza específicamente para buscar patrones de texto en archivos comprimidos con GZIP.

**wc (Word Count):** Se utiliza para contar palabras, líneas y caracteres en archivos o en la entrada estándar.

* 1. El empaquetamiento de archivos en GNU/Linux se refiere a agrupar uno o varios archivos y/o directorios en un único archivo
  2.  Compresión: Se reduce el tamaño de un archivo (gzip/bzip2/etc.)

gzip archivo.tar # Genera archivo.tar.gz comprimido

gzip -d archivo.tar.gz # Descomprime archivo.tar

* 1. Si

7-Indique qué acción realiza cada uno de los comandos indicados a continuación considerando su orden. Suponga que se ejecutan desde un usuario que no es root ni pertenece al grupo de root. (Asuma que se encuentra posicionado en el directorio de trabajo del usuario con el que se logueó). En caso de no poder ejecutarse el comando, indique la razón:

l s l > prueba **;Lista archivos y redirige salida a archivo prueba**

ps > PRUEBA ; **Lista los procesos en ejecución y los guarda en "PRUEBA**

chmod 710 prueba; **; Cambia los permisos, usuario todo los permisos (7), grupo solo ejecución(1), resto nada (0)**

*−*

chown root : root PRUEBA; **Cambia el propietario y el grupo del archivo "PRUEBA" a "root**

chmod 777 PRUEBA**; No tiene mas permisos, error**

chmod 700 / e tc / passwd **; No tiene mas permisos, error**

passwd root**; No tiene mas permisos, error**

rm PRUEBA**; No tiene mas permisos, error**

man / e tc / shadow; **; No tiene mas permisos, error**

f i n d /-name ∗ . conf; Realiza una búsqueda en todo el sistema de archivos ("/"

usermod root-d /home/ newroot-L, **; No tiene mas permisos, error**

cd / root; **Cambia el directorio actual al directorio "/root,"**

rm ∗**; No tiene mas permisos, error**

cd / e tc**; Cambia el directorio actual al directorio "/etc,**

cp ∗ /home-R ; Copia todos los archivos y directorios en el directorio "/etc" al directorio "/home"

shutdown**; No tiene mas permisos, error**

8)Indique qué comando sería necesario ejecutar para realizar cada una de las siguientes ac -ciones:

* 1. Terminar el proceso con *PID* 23. kill 23
  2. Terminar el proceso *init*. ¿Qué resultados obtuvo? killall init (Cae sist)
  3. Buscar todos los archivos de usuarios en los que su nombre contiene la cadena “.conf” find /home -name "\*\.conf"
  4. Guardar una lista de procesos en ejecución el archivo **/home/<su nombre de usua- rio>/procesos,** ps aux > /home/<su nombre de usuario>/procesos
  5. Cambiar los permisos del archivo **/home/<su nombre de usuario>/xxxx** a:

Usuario: Lectura, escritura, ejecución Grupo: Lectura, ejecución

Otros: ejecución

x

* 1. Cambiar los permisos del archivo **/home/<su nombre de usuario>/yyyy** a: Usuario: Lectura, escritura.

Grupo: Lectura, ejecución Otros: Ninguno

chmod u=rw,g=r-x,o=--- /home/<su nombre de usuario>/yyyy

* 1. Borrar todos los archivos del directorio **/tmp** rm -f /tmp/\*
  2. Cambiar el propietario del archivo **/opt/isodata** al usuario **iso2021**

chown iso2021 /opt/isodata

* 1. Guardar en el archivo **/home/<su nombre de usuario>/donde** el directorio donde me encuentro en este momento, en caso de que el archivo exista no se debe eliminar su contenido anterior.

pwd >> /home/<su nombre de usuario>/donde

9)Indique qué comando sería necesario ejecutar para realizar cada una de las siguientes ac- ciones:

* 1. Ingrese al sistema como usuario “root” su
  2. Cree un usuario. Elija como nombre, por convención, la primer letra de su nombre seguida de su apellido. Asígnele una contraseña de acceso.
* useradd -m -s /bin/bash a\_apellido

passwd a\_apellido

* 1. ¿Qué archivos fueron modificados luego de crear el usuario y qué directorios se crearon?

**/etc/passwd**, **/etc/shadow**, **/etc/group**, y posiblemente **/home/a\_apellido** si se ha creado un directorio de inicio para el nuevo usuario. Los directorios creados incluirán el directorio de inicio del usuario, que en este caso sería **/home/a\_apellido**.

* 1. Crear un directorio en /tmp llamado cursada2021 mkdir /tmp/cursada2021
  2. Copiar todos los archivos de /var/log al directorio antes creado.

cp -r /var/log/\* /tmp/cursada2021/

* 1. Para el directorio antes creado (y los archivos y subdirectorios contenidos en él) cambiar el propietario y grupo al usuario creado y grupo users.

chown -R a\_apellido:users /tmp/cursada2021/

* 1. Agregue permiso total al dueño, de escritura al grupo y escritura y ejecución a todos los demás usuarios para todos los archivos dentro de un directorio en forma recursiva.

chmod -R u+rwx,g+rw,o+rw /tmp/cursada2021/

* 1. Acceda a otra terminal virtual para loguearse con el usuario antes creado. Ctrl+Alt+F2
  2. Una vez logueado con el usuario antes creado, averigüe cuál es el nombre de su terminal. tty
  3. Verifique la cantidad de procesos activos que hay en el sistema. ps aux | wc -l
  4. Verifiqué la cantidad de usuarios conectados al sistema. who
  5. Vuelva a la terminal del usuario root, y envíele un mensaje al usuario anteriormente creado, avisándole que el sistema va a ser apagado.

wall "El sistema se apagará en breve. Por favor, guarde su trabajo."

* 1. Apague el sistema. shutdown -h now

10)Indique qué comando sería necesario ejecutar para realizar cada una de las siguientes ac- ciones:

* 1. Cree un directorio cuyo nombre sea su número de legajo e ingrese a él.

mkdir 12345 cd 12345

* 1. Cree un archivo utilizando el editor de textos vi, e introduzca su información personal: Nombre, Apellido, Número de alumno y dirección de correo electrónico. El archivo debe llamarse "LEAME".

vi LEAME, Luego, dentro de Vi, presione "i" para entrar en modo de inserción, escriba su información y luego presione "Esc" seguido de ":wq"

* 1. Cambie los permisos del archivo LEAME, de manera que se puedan ver reflejados los siguientes permisos:

Dueño: ningún permiso Grupo: permiso de ejecución Otros: todos los permisos

chmod 701 LEAME

* 1. Vaya al directorio /etc y verifique su contenido. Cree un archivo dentro de su directorio personal cuyo nombre sea leame donde el contenido del mismo sea el listado de todos los archivos y directorios contenidos en /etc. ¿Cuál es la razón por la cuál puede crear este archivo si ya existe un archivo llamado "LEAME.en este directorio?.
* cd /etc

ls -l > ~/leame

* 1. ¿Qué comando utilizaría y de qué manera si tuviera que localizar un archivo dentro del filesystem? ¿Y si tuviera que localizar varios archivos con características similares? Explique el concepto teórico y ejemplifique.

find /var -name "\*.log"

* 1. Utilizando los conceptos aprendidos en el punto e), busque todos los archivos cuya extensión sea .so y almacene el resultado de esta búsqueda en un archivo dentro del directorio creado en a). El archivo deberá llamarse .ejercicio\_f".

find / -name "\*.so" > 12345/.ejercicio\_f

Cuando crea un archivo en su directorio personal, como el archivo "leame," está creando ese archivo en su espacio personal y no en un directorio compartido como /etc. Por lo tanto, no afecta a otros usuarios ni interfiere con archivos fuera de su directorio personal.

11)Indique qué acción realiza cada uno de los comandos indicados a continuación considerando su orden. Suponga que se ejecutan desde un usuario que no es root ni pertenece al grupo de root. (Asuma que se encuentra posicionado en el directorio de trabajo del usuario con el que se logueó). En caso de no poder ejecutarse el comando indique la razón:

mkdir i s o Se crea un directorio llamado "iso"

cd ./ i s o ps > f 0 Cambia al

directorio "iso." Y muestra una lista de procesos en la terminal y redirige la salida al archivo "f0."

l s > f 1muestra el contenido del directorio y redirige la salida al archivo "f1."

cd /Cambia al directorio raíz

echo $HOME Muestra el valor de la variable de entorno "$HOME

l s l &> $HOME/ i s o / l s Ejecuta el comando **ls -l** y redirige tanto la salida estándar como la salida de error al archivo "ls" en el directorio "iso" dentro del directorio de inicio del usuario.

*−*

cd $HOME; mkdir f 2 Cambia al directorio de inicio del usuario y crea un directorio llamado "f2" en ese directorio.

l s ld f 2 Muestra la información detallada del directorio "f2", incluyendo sus permisos.

*−*

chmod 341 f 2 Cambia los permisos del directorio "f2" a "341". Esto dará permisos de lectura y escritura al propietario, permisos de escritura al grupo y ningún permiso al resto.

touch d i r Crea un archivo vacío llamado "dir" en el directorio de trabajo actual.

cd f 2

cd ~/ i s o Cambia al directorio "f2" que se encuentra en el directorio de inicio del usuario.

pwd >f 3 **cd ~/iso**: Obtiene la ruta actual (el directorio "iso" en el directorio de inicio del usuario) y la guarda en un archivo llamado "f3".

ps | grep ' ps ' | wc l >> . . / f 2 / f 3 Ejecuta el comando **ps**, filtra las líneas que contienen la palabra "ps" con **grep**, cuenta el número de líneas con **wc -l** y agrega ese número al final del archivo "../f2/f3" (subiendo dos niveles de directorio).

*−*

chmod 700 . . / f 2 ; cd . . ; EROOR

f i n d . name e tc / passwd Busca el archivo "passwd" en el directorio actual y sus subdirectorios, pero este comando no proporcionará resultados si el archivo no existe en el directorio actual.

*−*

*−*

mkdir e j e r c i c i o 5 Crea un directorio llamado "ejercicio5" en el directorio de trabajo actual.

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

* 1. Inicie 2 sesiones utilizando su nombre de usuario y contraseña. En una sesión vaya siguiendo paso a paso las órdenes que se encuentran escritas en el cuadro superior. En la otra sesión, cree utilizando algún editor de textos un archivo que se llame

.ejercicio10\_explicacion"dentro del directorio creado en el ejercicio 9.a) y, para cada

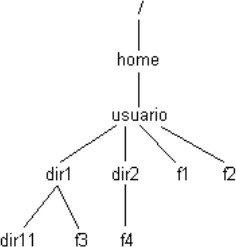
una de las órdenes que ejecute en la otra sesión, realice una breve explicación de los resultados obtenidos.

* 1. Complete en el cuadro superior los comandos 19 y 20, de manera tal que realicen la siguiente acción:

19: Copiar el directorio iso y todo su contenido al directorio creado en el inciso 9.a).

20: Copiar el resto de los archivos y directorios que se crearon en este ejercicio al directorio creado en el ejercicio 9.a).

* 1. Ejecute las órdenes 19 y 20 y comentelas en el archivo creado en el inciso a).



12)Cree una estructura desde el directorio /home que incluya varios directorios, subdirectorios y archivos, según el esquema siguiente. Asuma que “usuario” indica cuál es su nombre de usuario. Además deberá tener en cuenta que dirX hace referencia a directorios y fX hace referencia a archivos:

* 1. Utilizando la estructura de directorios anteriormente creada, indique que comandos son necesarios para realizar las siguientes acciones:

Mueva el archivo "f3.al directorio de trabajo /home/usuario. Copie el archivo "f4.en el directorio "dir11".

Haga los mismo que en el inciso anterior pero el archivo de destino, se debe llamar "f7".

Cree el directorio copia dentro del directorio usuario y copie en él, el contenido de "dir1".

Renombre el archivo "f1"por el nombre archivo y vea los permisos del mismo. Cambie los permisos del archivo llamado archivo de manera de reflejar lo siguiente:

* + - Usuario: Permisos de lectura y escritura
    - Grupo: Permisos de ejecución
    - Otros: Todos los permisos

Renombre los archivos "f3 te.

"f4"de manera que se llamen "f3.exe

"f4.exerespectivamen-

Utilizando un único comando cambie los permisos de los dos archivos renombrados en el inciso anterior, de manera de reflejar lo siguiente:

* + - Usuario: Ningún permiso
    - Grupo: Permisos de escritura
    - Otros: Permisos de escritura y ejecución

13)Indique qué comando/s es necesario para realizar cada una de las acciones de la siguiente secuencia de pasos (considerando su orden de aparición):

* 1. Cree un directorio llamado logs en el directorio /tmp. mkdir /tmp/logs
  2. Copie todo el contenido del directorio /var/log en el directorio creado en el punto anterior. cp -r /var/log/\* /tmp/logs/
  3. Empaquete el directorio creado en 1, el archivo resultante se debe llamar "misLogs.tar". tar -cf misLogs.tar -C /tmp logs
  4. Empaquete y comprima el directorio creado en 1, el archivo resultante se debe llamar "misLogs.tar.gz". tar -czf misLogs.tar.gz -C /tmp logs
  5. Copie los archivos creados en 3 y 4 al directorio de trabajo de su usuario.

cp /tmp/misLogs.tar .

cp /tmp/misLogs.tar.gz

* 1. Elimine el directorio creado en 1, logs. rm -r /tmp/logs
  2. Desempaquete los archivos creados en 3 y 4 en 2 directorios diferentes.

mkdir logs\_tar

tar -xf misLogs.tar -C logs\_tar

mkdir logs\_targz

tar -xzf misLogs.tar.gz -C logs\_t