PRACTICA 2 ISO [GNU/Linux]

1.- Editor de Textos

a) Nombre al menos 3 editores de texto que podría utilizar desde el entorno texto
c) Enumere los modos de operación que posee el editor de textos vi
d) Nombre los comandos mas comunes que se le pueden enviar al editor de textos vi

MCEDIT, VIM, GEDIT, KATE, ETC.

Editor de textos vim (el vi mejorado):
Está presente en cualquier distribución de GNU/Linux y posee 3 modos de ejecución:

- Modo Insert
- Modo Replace
- Modo comandos

Se le pueden enviar una serie de comandos útiles:

- w: Escribir cambios
- q ó q!: Salir del Editor
- Tecla Ins ó i: Ingresar al modo edición
- Tecla Ins o r: Ingresar al modo reemplazo
- dd: Borrar una línea (copiarla al portapapeles o Cortar)
- Y: Copiar al portapapeles
- p: Pegar desde el portapapeles
- /frase: Busca la palabra frase dentro del archivo

b) ¿En que se diferencia un editor de texto a los comandos cat, more o less?

Los **comandos cat, more y less** permiten mostrar el contenido de ficheros de texto desde la línea de comandos en sistemas **Unix**, en su lugar los editores de texto, justamente nos dejan EDITAR TEXTO, ADEMÁS DE PODER VISUALIZARLO.
El comando 'cat' imprimirá por pantalla el contenido del fichero sin ningún tipo de paginación ni posibilidad de modificarlo. Básicamente concatena archivos o la salida estándar en la salida estándar.
Al igual que 'cat', 'more' permite visualizar por pantalla el contenido de un fichero de texto, con la diferencia con el anterior de que 'more' 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.
El comando 'less' es el más completo de los tres, pues 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.

2.- Proceso de Arranque

a) 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.

Proceso de Arranque System V
BootStrap:
Paso 0: Se ejecuta el código de la BIOS
Paso 1: El hardware lee el sector de arranque
Paso 2: Se carga el gestor de arranque
Paso 3: Se carga el kernel
Paso 4: Se monta el sistema de archivos raíz
Paso 5: Se ejecuta el proceso init
Paso 6: Lee el /etc/inittab
Paso 7: Ejecuta los scripts apuntados por el runlevel 1
Paso 8: El final del runlevel 1 le indica que vaya al runlevel por defecto
Paso 9: Ejecuta los scripts apuntados por el runlevel por defecto
Paso 10: El sistema está listo para usarse

b) Proceso *INIT.* ¿Quién lo ejecuta? ¿Cuál es su objetivo?
Su función es cargar todos los subprocesos necesarios para el correcto funcionamiento del SO.
Posee PID 1 y se encuentra en /sbin/init. Se lo configura a través del archivo /etc/inittab. No tiene padre y es padre de todos los procesos. Es el encargado de montar los filesystems y de hacer disponibles los demás dispositivos

c) RunLevels. ¿Qué son? ¿Cuál es su objetivo?
Es el modo en que arranca linux (3 en redhat, 2 en Debian)
Un nivel de ejecución es básicamente una configuración de programas y servicios que se ejecutarán orientados a un determinado funcionamiento.
El proceso de arranque se divide en niveles. Cada uno es responsable de levantar o bajar una serie de servicios.
Se encuentran definidos en /etc/inittab:

id:niveles_ejecucion:acción:proceso

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

d) ¿A qué hace referencia cada nivel de ejecución según el estándar? ¿Dónde se define que Runlevel ejecutar? ¿Todas las distribuciones respetan estos estándares?

Existen 7 runlevels, y permiten iniciar un conjunto de procesos al arranque

0 Parada Finaliza servicios y programas activos, así como desmonta filesystems activos y para la
CPU.
1 Monousuario Finaliza la mayoría de servicios, permitiendo sólo la entrada del administrador (*root*). Se
usa para tareas de mantenimiento y corrección de errores críticos.
2 Multiusuario sin red No se inician servicios de red, permitiendo sólo entradas locales en el sistema.
3 Multiusuario Inicia todos los servicios excepto los gráficos asociados a X Window.
4 Multiusuario No suele usarse, se deja libre para que cada usuario lo configure a su gusto.
5 Multiusuario X Igual que el 3, pero con soporte X para la entrada de usuarios (*login* gráfico).
6 Reinicio Para todos los programas y servicios. Reinicia el sistema.

Los scripts que se ejecutan están en /etc/init.d/
En /etc/rcX.d (donde X = 0..6) hay links a los archivos de /etc/init.d/
Formato de los links:
[S|K]<orden><NombreDelScript>
S: Lanza el script con el argumento start
K: Lanza el script con el argumento stop

Cuando un sistema GNU/Linux arranca, primero se carga el kernel del sistema, después se inicia el primer proceso, denominado init, que es el responsable de ejecutar y activar el resto del sistema, mediante la gestión de los niveles de ejecución (o *runlevels*).

En el caso del modelo *runlevel* de SystemV, cuando el proceso init arranca, utiliza un fichero de configuración llamado /etc/inittab para decidir el modo de ejecución en el que va a entrar. En este fichero se define el *runlevel* por defecto (*initdefault*) en arranque (por instalación en Fedora el 5, en Debian el 2), y una serie de servicios de terminal por activar para atender la entrada del usuario.

Después, el sistema, según el *runlevel* escogido, consulta los ficheros contenidos en /etc/rcn.d, donde *n* es el número asociado al *runlevel* (nivel escogido), en el que se encuentra una lista de servicios por activar o parar en caso de que arranquemos en el *runlevel,* o lo abandonemos. Dentro del directorio encontraremos una serie de *scripts* o enlaces a los *scripts* que controlan el servicio. Cada *script* posee un nombre relacionado con el servicio, una S o K inicial que indica si es el *script* para iniciar (S) o matar (K) el servicio, y un número que refleja el orden en que se ejecutarán los servicios.

No todas las distribuciones respetan los estándares.

e) Archivo */etc/inittab.* ¿Cuál es su finalidad? ¿Qué tipo de información se almacena en el? ¿Cuál es la estructura de la información que en el se almacena?

Es el archivo de configuración de init.
Cuando el sistema se arranca, se verifica si existe un runlevel predeterminado en el archivo/etc/inittab, si no, se debe introducir por medio de la consola del sistema. Después se procede a ejecutar todos los scripts relativos al runlevel especificado.

f) Indique como haría para cambiar de un runlevel a otro.

Existen dos formas de modificar los runlevels:
→Cambiar de runlevel en ejecución
Existe una utilidad para línea de comandos que permite cambiar de un nivel de ejecución a otro. Esta es la herramienta init. Para cambiar de nivel de ejecución sólo hay que ejecutar init seguido del número del runlevel. Por ejemplo:

- init 0: Cambia al runlevel 0 (se apaga el sistema, equivalente al comando halt).
- init 2: Cambia al runlevel 2.
- init 6: Cambia al runlevel 6 (reinicia el sistema, equivalente al comando reboot).

También telinit, nos permite cambiar de nivel de ejecución, sólo tenemos que indicar el número. Por ejemplo, necesitamos hacer una tarea crítica en *root*; sin usuarios trabajando, podemos hacer un telinit 1 (también puede usarse S) para pasar a *runlevel* monousuario, y después de la tarea un telinit 3 para volver a multiusuario.

→Modificar el runlevel por defecto
Por defecto, el sistema suele arrancar en el nivel de ejecución 5 (modo gráfico). Si se quisiera modificar este comportamiento, habría que editar el fichero /etc/inittab.
Más concretamente, habría que modificar en el fichero /etc/inittab la línea donde el número 5 indica que el nivel de ejecución por defecto es el 5.

g) Scripts RC. ¿Cuál es su finalidad? ¿Dónde se almacenan?

Los scripts RC se encargan de cargar o cerrar los servicios necesarios para que el sistema funcione, de acuerdo con el runlevel que se está iniciando. Por ejemplo: lpd (servicio para imprimir), fetchmail (servicio para leer correo-e), sshd (SecureShell, para abrir sesiones remotas de una manera segura), networking (abre las conexiones de red). Todos estos servicios se encuentran en /etc/init.d/.
Sin embargo, no todos los servicios se cargan en todos los runlevels.¿Cómo sabe el RC que servicios tiene que cargar? Los servicios a cargar se encuentran en el directorio /etc/rcX.d/, donde X es el runlevel a cargar. En realidad, en estos directorios no hay más que enlaces simbólicos a /etc/init.d/
Nombres de los enlaces simbólicos

Los nombres en estos directorios tienen una sintaxis bastante concreta. Empiezan por un letra (S o K) seguidos de un número y el nombre del servicio. La letra S significa iniciar (S de start). La letra K significa acabar (K de kill). El número es de dos dígitos, de 00 a 99 e indica el orden en el que se arrancará el servicio.

Cuando un sistema GNU/Linux arranca o se detiene se ejecutan scripts, indique cómo determina qué script ejecutar ante cada acción. ¿Existe un orden para llamarlos? Justifique.

Realmente, **/etc/rc.d/rc** cuando entra en un determinado nivel de ejecución realiza las siguientes acciones:

1. Ejecuta, por orden de nombre, todos los scripts que comienzan por **K** en el directorio correspondiente al nivel, utilizando como argumento para dicho script la opción **stop**.

2. Ejecuta, por orden de nombre, todos los scripts que comienzan por **S** en el directorio correspondiente al nivel, utilizando como argumento para dicho script la opción **start**.

j) ¿Qué es insserv?¿Para qué se utiliza?¿Qué ventajas provee respecto del arranque tradicional?
El programa insserv se usa para actualizar el orden de los enlaces simbólicos que están en «/etc/rc?.d/» en forma dinámica, con un sysv-rc basado en dependencias especificadas en los propios scripts utilizando las cabeceras LSB de los scripts de init.d.
Esto permite a cada mantenedor de paquetes especificar en su script de init.d la relación con otros scripts y poder detectar y evitar bucles de dependencias entre scripts así como asegurarse de que todos los scripts se inician en el orden pretendido.
El programa insserv que contiene este paquete se debería utilizar con cuidado y junto al paquete sysv-rc, ya que utilizarlo incorrectamente puede llevar a un sistema inarrancable. Mejora la performance del arranque en sistemas multiprocesadores.

k) ¿Cómo maneja Upstart el proceso de arranque del sistema?

El ***upstart*** es un reemplazo **basado en eventos, y no en niveles**. Los servicios se pueden levantar o desactivar en respuesta a ciertos eventos, y este procedimiento permite por ejemplo manejar el reinicio de servicios que mueren de forma inesperada. Upstart (programado por Scott James Remnant, trabajador de Canonical Ltd) opera asíncronamente. Las tareas y servicios son ejecutados ante eventos (arranque del equipo o inserción de un dispositivo USB) definidos como tareas o Jobs. Los jobs se almacenan en el directorio /etc/init. Son scripts en texto plano que definen las acciones a ejecutar. Es compatible con el System V.

l) Cite las principales diferencias entre SystemV y Upstart

Upstart es el reemplazo de SystemV (generalmente incluido en Ubuntu y Fedora, openSUSE y otros). Permite la ejecución de tareas en forma asincrónica como principal diferencia con sysVinit que es estrictamente sincrónico. Las tareas y servicios son ejecutados ante eventos (arranque del equipo o inserción de un dispositivo USB) definidos como tareas o jobs. Los jobs se almacenan en el directorio /etc/init. Son scripts en texto plano que definen las acciones a ejecutar.
El init es el responsable de lanzar las tareas. Es compatible con el System V.

3.- Usuarios

a) ¿Qué archivos son utilizados en un sistema GNU/Linux para guardar la información de los usuarios?

/home en cuanto a sus “cosas” que quieran hacer, en cuanto a información personal son /etc/passwd o /etc/shadow

b) ¿A que hacen referencia las siglas UID y GID? ¿Pueden coexistir UIDs iguales en un sistema GNU/Linux? Justifique.

UID: USER ID
GID: GROUP ID
Se supone que no, pero hay un caso excepcional, el UID = 0 (sería el root) Pueden llegar a haber más de un root, y coexisten, pero son casos excepcionales, se supone que no pueden haber varios usuarios con un mismo ID.

c) ¿Qué es el usuario *root*? ¿Puede existir más de un usuario con este perfil en GNU/Linux? ¿Cuál es la UID del root?

Usuario root
• También llamado súper usuario o administrador.
• Su UID (User ID) y GID es 0 (cero).
• Es la única cuenta de usuario con privilegios sobre todo el sistema.
• Acceso total a todos los archivos y directorios con independencia de propietarios y permisos.
• Controla la administración de cuentas de usuarios.
• Ejecuta tareas de mantenimiento del sistema.
• Puede detener el sistema.
• Instala software en el sistema.
• Puede modificar o reconfigurar el kernel, controladores, etc.

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

- useradd ó adduser: crea un nuevo usuario, el segundo es full.
- userdel: elimina un usuario
- groupadd: crea un nuevo grupo
- groupdel: elimina un grupo
- usermod: cambia las opciones del usuario
- su: inicia con un nuevo usuario (nuevo shell)
- who: muestra usuarios en el sistema.
- passwd: cambia la contraseña del usuario

4.- FileSystem

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

Tipos de permisos
Cada archivo pertenece a un usuario y a un grupo en particular. Un grupo es un conjunto de usuarios definido (cada usuario pertenece al menos a un grupo, pero puede pertenecer a varios).
Los grupos usualmente son definidos por el tipo de usuarios que acceden al sistema. Por ejemplo, en un sistema *Unix* de una universidad, los usuarios pueden ser divididos en los grupos estudiantes, dirección, profesores e invitados. Hay también unos pocos grupos definidos por el sistema (como bin y daemon) que son usados por el propio sistema para controlar el acceso a los recursos. Normalmente los usuarios comunes no pertenecen a estos grupos.
Los permisos están divididos en tres tipos: lectura, escritura y ejecución. Estos permisos pueden ser fijados para tres clases de usuarios: el propietario del archivo o 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 (usando ls).
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.

Por último, 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 (por ejemplo, con el comando cd).

Interpretando los permisos de archivos
Veamos un ejemplo del uso de permisos de archivos. Usando el comando ls con la opción "-l" se mostrara un listado "largo" de los archivos, el cual incluye los permisos.

fabrizio@debian: /$ ls -l
-rwxr-xr-- 1 fabrizio users 505 May 5 19:05 prueba.exe

El primer campo representa los permisos del archivo. El tercer campo es el propietario del mismo (fabrizio), el cuarto es el grupo al cual pertenece el archivo (users) y el último campo es el nombre del archivo (prueba.exe).
La cadena "-rwxr-xr--" nos informa, por orden, los permisos para el propietario, los usuarios del grupo y el resto de los usuarios.
El primer carácter de la cadena de permisos ("-") representa el tipo de archivo. El "-" significa que es un archivo regular, "d" indicaría que se trata de un directorio. Los siguientes tres caracteres ("rwx") representan los permisos para el propietario del archivo, fabrizio. Éste tiene permisos para leer (r), escribir (w) y ejecutar (x) el archivo prueba.exe.
Los siguientes tres caracteres, "r-x", representan los permisos para los miembros del grupo al que pertenece el archivo (en este caso, users). Como sólo aparece "r-x" cualquier usuario que pertenezca al grupo users puede leer este archivo, y ejecutarlo, pero no modificarlo.
Los últimos tres caracteres, "r--", representan los permisos para cualquier otro usuario del sistema (que no sea fabrizio ni pertenezca al grupo users). Como sólo está presente la "r", los demás usuarios pueden leer el archivo, pero no escribir en él o ejecutarlo.
Aquí tenemos otros ejemplos de permisos de grupo:
-rw-------
El propietario del archivo puede leer y escribir. Nadie más puede acceder al archivo.
-rwxrwxrwx
Todos los usuarios pueden leer, escribir y ejecutar el archivo.
drwxr-xr-x
El propietario del directorio puede leer, escribir y entrar al mismo. Los usuarios pertenecientes al grupo del directorio y todos los demás usuarios pueden leer e ingresar al directorio.

b) Investigue la funcionalidad y parámetros de los siguientes comandos relacionadolos con los permisos en GNU/Linux:

chmod: modifica los permisos de un archivo especificado.
chown: cambia el propietario de un fichero/directorio.
chgrp: cambia el grupo propietario de un fichero/directorio.

c) Al utilizar el comando chmod generalmente se utiliza una notación octal asociada para definir permisos. ¿Qué significa esto? ¿A qué hace referencia cada valor?

R: read: solo lectura = 4
W: write: solo escritura = 2
X: execute: solo ejecución = 1

d) ¿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 correspondientes

Existe la posibilidad, si es que el usuario puede utilizar el comando su, o sudo. Sino, sin poseer los permisos necesarios no puede acceder al archivo. Sólo root.

e) Explique los conceptos de "*full path name*" y "*relative path name*". De ejemplos claros de cada uno de ellos.

Nombre de ruta completo: Un nombre de vía de acceso completo empieza en el directorio raíz y efectúa un rastreo de todos los directorios que quedan por debajo hasta llegar al archivo y directorio de destino.
Por ejemplo, **/etc/uucp/Devices** hace referencia al archivo **Devices** del directorio **uucp** del directorio raíz **etc**.
Para indicar un directorio raíz, siempre es preciso especificar delante el carácter de barra inclinada (/). Separe siempre los elementos de la vía de acceso mediante el carácter de barra inclinada (/).

Nombre de ruta relativo: El nombre de vía de acceso relativo sólo contiene los directorios que dependen del directorio actual.
Por ejemplo, si el directorio actual es **/usr/bin** y el directorio de destino es **/usr/bin/reports**, escriba el nombre de vía de acceso relativo **reports** (sin la barra inclinada inicial).

f) ¿Con qué comando puede determinar en qué directorio se encuentra actualmente?
Pwd
¿Existe alguna forma de ingresar a su directorio personal sin necesidad de escribir todo el path completo?
Si: /home o cd
¿Podría utilizar la misma idea para acceder a otros directorios? ¿Cómo? Explique con un ejemplo.
Sí, como por ejemplo /etc

g) Investigue la funcionalidad y parámetros de los siguientes comandos relacionados con el uso del FileSystem:

- cd: cambia de directorio
- mkdir: crea un directorio
- rmdir: elimina un directorio
- mount: monta un dispositivo
- umount: desmonta un dispositivo
- du: muestra lo que ocupa y el tamaño total de los directorios dentro del directorio donde me encuentro
- df: muestra los sistemas de ficheros montados
- ln: crea enlaces a archivos (crea un fichero que apunta a otro)
- ls: lista los archivos y directorios dentro del entorno de trabajo
- pwd: muestra el directorio actual de trabajo
- cp: copia archivos en el directorio indicado
- mv: renombra un conjunto

5.- Procesos

a) ¿Qué es un proceso? ¿A que hacen referencia las siglas PID y PPID? ¿Todos los procesos tienen estos atributos en GNU/Linux? Justifique.Indique que otros atributos tiene un proceso.

Un programa es un archivo ejecutable, un proceso es un programa que esta siendo ejecutado. Cada proceso tiene su propio medio ambiente. La sigla PID hace referencia al ID del Proceso y la sigla PPID hace referencia al ID del Proceso Padre del proceso. Todos los procesos tienen estos atributos, además de estos (que son los más

importantes pero no todos): Usuario (UID), Grupo (GID), Prioridad, etc. Con ps –ejH PUEDO VER EL PPID.

b) Indique que comandos se podrían utilizar para ver que procesos están en ejecución en un sistema GNU/Linux.

Ps, pstree y top
Las opciones más importantes y utilizadas de este comando son: **-a** para mostrar los procesos de todos los teminales, **-u** para mostrar el usuario al que pertenece el proceso y la hora de inicio, y -x para mostrar procesos que no estan controlados por ningún terminal. Suelen usarse combiandas para tener una visión global de los procesos que están en ejecutan. Otros dos comandos útiles a la hora de visualizar los procesos son: pstree, que nos muestra los procesos en una estructura de árbol; y top.

c) ¿Qué significa que un proceso se esta ejecutando en Background? ¿Y en Foreground?

En Linux, podemos colocar un proceso en background, esto es, el proceso continúa la ejecución mientras que el shell se libera para otras actividades; Foreground es exactamente lo contrario.

d) ¿Cómo puedo hacer para ejecutar un proceso en Background? ¿Cómo puedo hacer para pasar un proceso de background a foreground y viceversa?

Foreground: Los procesos se inician de este modo simplemente escribiendo el comando en la línea de comandos.
Background: Los procesos se inician de este modo escribiendo un & al final de la línea del comando.
También se puede colocar un proceso en background a través de la utilización de las teclas Ctrl+z seguido del comando bg. Para retornar el proceso al shell principal, utilizamos el comando fg [N*], de esta forma el proceso vuelve a shell inicial.
N*: número de orden de entrada en el sistema

e) Pipe (|). ¿Cual es su finalidad? Cite ejemplos de su utilización.

Un pipe es un par de "archivos inexistentes", que tienen la cualidad de que lo que se escribe en uno se lee en el otro.
Este mecanismo nos permite pasar la salida de un comando a otro. Para ello se usa la sintaxis: **<comando1> | <comando2>**. Con esto, la salida de *comando1* será la entrada de *comando2*. Vamos a ver unos ejemplos:

```
$ rpm -qa | grep <nombre_paquete>
```

El primero de los dos comandos nos haría una lista de todos los paquetes instalados. Imaginemos que sólo queremos saber si tenemos instalado uno en concreto. Con el segundo comando limitamos la salida a los paquetes que en el nombre que contengan el *patrón* que especificamos en *<nombre_paquete>*. Por ejemplo, para saber si tenemos instalado algún paquete llamado **glibc** haríamos:

```
$ rpm -qa | grep "glibc"
```

grep es un *parseador* de expresiones regulares, es decir, le damos un patrón y un fichero (o introducimos lo que sea por consola, o lo pasamos con un *pipe*) y de ese texto nos devuelve sólo lo que coincide con el patrón.
Otro ejemplo útil seria, por ejemplo, cuando queremos saber el PID de un proceso. En vez de mostrarlos todos y tener que buscarlo podríamos hacer:

```
$ ps -e | grep <nombre_proceso>
```

y así nos mostraría sólo las líneas que contuvieran *<nombre_proceso>* (es decir, limitaríamos la salida al proceso que queremos).

f) Redirección. ¿Qué tipo de redirecciones existen? ¿Cuál es su finalidad?
Cite ejemplos de utilización.

En Linux, al final todo es tratado como si fuera un fichero y como tal, tenemos descriptores de fichero para aquellos puntos donde queramos acceder. Hay unos descriptores de fichero por defecto:

- **0**: Entrada estándar (normalmente el teclado).
- **1**: Salida estándar (normalmente la consola).
- **2**: Salida de error.

Para redirigir las salidas utilizaremos el **descriptor de fichero** seguido del símbolo '>' (o < si redirigimos la entrada hacia un comando). Veamos unos ejemplos:

$ ls -l >fichero
Guarda la salida de *ls -l* en *fichero*. Si no existe lo crea, y si existe lo sobreescribe.
$ ls -l >>fichero
Añade la salida del comando a *fichero*. Si no existe lo crea, y si existe, lo añade al final.
$ ls -l 2>fichero
Si hay algún error, lo guarda en *fichero* (podría salir un error si no tuviéramos permiso de lectura en el directorio).

Es importante ver que si no se especifica el descriptor de fichero se asume que se redirige la salida estándar. En el caso del operador < se redirige la entrada estándar, es decir, el contenido del fichero que especificáramos, se pasaría como parámetro al comando.
Si quisiéramos redirigir todas las salidas a la vez hacia un mismo fichero, podríamos utilizar >&. Además, con el carácter & podemos redirigir salidas de un tipo hacia otras, por ejemplo, si quisiéramos redirigir la salida de error hacia la salida estándar podríamos indicarlo con: 2>&1. Es importante tener en cuenta que el orden de las redirecciones es significativo: se ejecutarán de izquierda a derecha.

g) Comando Kill. ¿Cuál es su funcionalidad? Cite ejemplos.

En Unix y los sistemas operativos tipo Unix, **kill** 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. Pero kill no tiene por que tener que ver con matar al proceso. El comando **kill** es un wrapper alrededor de la llamada al sistema kill(), que envía señales a procesos o grupos de procesos en el sistema, referenciados por sus IDs de proceso (PIDs) o IDs de grupo de procesos (PGIDs). kill se ofrece siempre como programa independiente, pero la mayoría de las shells tienen intrínsecamente comandos kill que pueden diferir levemente de ella.
Pueden enviarse numerosas señales. La señal por defecto es SIGTERM. Los programas que cazan esta señal pueden limpiar su estado (como guardar datos de la configuración a un archivo) antes de terminarse. Para los programas que no capturan esta señal, se utiliza una gestión por defecto. En otras ocasiones, un proceso capaz de capturar la señal puede quedar en un estado que le impide manejarla.
Todas las señales excepto SIGKILL y SIGSTOP pueden ser interceptadas por el proceso, esto es, tener una función especial que es llamada cuando el programa recibe tales señales. Sin embargo, SIGKILL y SIGSTOP sólo las ve el kernel, lo que provee formas seguras de controlar la ejecución de los procesos. SIGKILL finaliza el proceso, y SIGSTOP lo pone en pausa hasta que se reciba una señal SIGCONT.
Unix cuenta con mecanismos de seguridad para evitar que usuarios no autorizados puedan finalizar otros procesos. Básicamente, para que un proceso pueda enviar una señal a otro, deben tener el mismo propietario, o ser enviada por el superusuario.
Las señales disponibles tienen distintos nombres, asignados a determinados números. El número de las señales puede cambiar entre distintas implementaciones de Unix. SIGKILL suele tener el número 9 y SIGTERM el 15.

h) Investigue la funcionalidad y parámetros de los siguientes comandos relacionados con el manejo de procesos en GNU/Linux. Además compárelos entre ellos:

- ps: muestra la lista de procesos del usuario
- pstree: ps en forma de árbol
- top: ps en la Terminal, tipo texto
- kill: Permite interactuar con cualquier proceso mandando señales. Kill <pid> termina un proceso y Kill -9 <pid> fuerza a terminar un proceso en caso de que la anterior opción falle.
- killall: mata todos los procesos que el usuario es capaz de matar (Sistem V) o solo los procesos especificados en la línea de comandos (psmisc).
- nice: es usado para iniciar un proceso con una determinada prioridad. nice -10 named: Esto *bajaría* la prioridad de named en 10 unidades(Si estaba en -10, pasará a -20)MENOS GENTIL = MAS PRIORIDAD.

6.- Otros comandos de Linux (Indique funcionalidad y parámetros):
a) ¿A qué hace referencia el concepto de empaquetar archivos en GNU/linux?

Los archivos TAR no son archivos comprimidos sino empaquetados. TAR es un empaquetador, es algo más parecido a un compresor como "arj" ó "zip" pero sin compresión. Su función es la de incluir todos los ficheros juntos en el mismo archivo, conservando las estructuras de directorios y permisos de los mismos.
Para crear un archivo .tar
tar –cvf mi_archivo.tar /directorio/a/empaquetar

Para extraer el contenido de un archivo .tar ya creado
tar –xvf mi_archivo.tar

Para actualizar el contenido de un archivo .tar ya existente
tar –uvf mi_archivo.tar

Para agregar un directorio a un archivo .tar ya existente
tar –rvf archivo.tar /directorio/a/agregar

Para empaquetar y comprimir el archivo .tar
tar –cvzf archivo.tgz /directorio/a/empaquetar/y/comprimir

Para desempaquetar el archivo .tgz
tar –xvzf archivo.tgz

Ojo : Observa la extensión "**.tgz**" para el tar comprimido. Aunque también podemos ponerle como extensión "**.tar.gz**"

b) Seleccione 2 archivos dentro de algún directorio al que tenga permiso y sume el tamaño de cada uno de estor archivos. Cree un archivo empaquetado conteniendo estos 2 archivos y compare los tamaños de los mismos. ¿Qué característica nota?

El tamaño del archivo empaquetado es menor al tamaño de los 2 archivos por separado, sin empaquetar.

c) ¿Qué acciones debe llevar a cabo para comprimir 4 archivos en uno solo?
Indique la secuencia de comandos ejecutados

tar –cvzf mi_archivo_comprimido.tgz /home/fabrizio/archivo1o2o3o4

d) ¿Pueden comprimirse un conjunto de archivos utilizando un único comando?

No, debo comprimir de a uno, aunque sea engorroso, es así. Lo que si puedo es empaquetar, primero, varios archivos en uno, y a esto comprimirlo, con lo cual estaría comprimiendo un conjunto de archivos, pero antes empaquetandolo, como dijimos.

e) Investigue la funcionalidad de los siguientes comandos:

- tar: empaqueta o comprime archivos usando la extensión .tar
- gzip: comprime solo archivos utilizando la extensión .gz
- wc: cuenta los caracteres, palabras y líneas del archivo de texto.
- grep: escribe en la salida estándar aquellas líneas que coincidan con un patrón
- zgrep: Busca archivos comprimidos para una expresión.

7.- Indique que 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 logueo). En caso de no poder ejecutarse el comando indique la razón:

a) ls -l > prueba

Genera un archivo de nombre prueba que contiene un listado detallado con los contenidos del directorio home del usuario. Se redirije la salida estandar de ls mediante el carácter > hacia el archivo prueba.

b) ps > PRUEBA

Genera un archivo de nombre PRUEBA que contiene un listado de los procesos en ejecución en el directorio home del usuario. AL igual que en el ejemplo anterior, se redirije la salida estandar mediante >.

c) chmod 710 prueba

Cambia los permisos del archivo prueba a 710 para UGO (usuario, Grupo, Otros).

d) chown root:root PRUEBA

Se intenta cambiar el propietario del archivo prueba pero la operación no está permitida.

e) chmod 777 PRUEBA

Cambia los permisos del archivo PRUEBA a 777. Es decir, todos los usuarios pueden leer, escribir y ejecutar el archivo.

f) chmod 700 /etc/passwd

Intenta cambiar los permisos a 700, pero la operaciòn no está permitida para un usuario que no es root, esto por el archivo que esta intentando cambiar.

g) passwd root

passwd: No debe ver o cambiar la información de la contraseña para root.

h) rm PRUEBA

Se elimina el archivo PRUEBA.

i) man /etc/shadow

Permiso denegad, porque “man” no debe recibir una ruta, si hago “man shadow” si anda.

j) find / -name *.conf

Lista todos los archivos cuyo nombres terminan con .conf, empezando la búsqueda en el directorio raíz /.

k) usermod root –d /home/newroot –L

l) cd /root

Se intenta acceder a la carpeta root, pero la operación falla porque el usuario no tiene los permisos.

m) rm *

Borra todos los archivos del directorio donde está posicionado el usuario.

n) cd /etc

Cambia el directorio a /etc, osea “se mueve” a /etc

o) cp * /home –R

Intenta copiar el contenido de /etc a home, pero el usuario no tiene los permisos necesarios para crear archivos en el directorio /home.

p) shutdown

Apaga el equipo.

8.- Indique que comando seria necesario ejecutar para realizar cada una de las siguientes acciones:

a) Terminar el proceso con PID 23
KILL -9 23
b) Terminar el proceso llamado *init*. ¿Qué resultados obtuvo?
KILL -9 1. Nada, no hace nada, porque el proceso init, no puede terminarse, así que ni responde al comando.
c) Buscar todos los archivos de usuarios en los que su nombre contiene la cadena ".conf"
find / -name *.conf
d) Guardar una lista de procesos en ejecución el archivo */home/<su nombre de usuario>/procesos*
ps > /home/fabrizio/procesos
e) Cambiar los permisos del archivo */home/<su nombre de usuario>/xxxx* a:
a. Usuario: Lectura, escritura, ejecución
b. Grupo: Lectura, ejecución
c. Otros: ejecución
chmod 751 /home/fabrizio/xxxx
f) Cambiar los permisos del archivo */home/<su nombre de usuario>/yyyy* a:
a. Usuario: Lectura, escritura.
b. Grupo: Lectura, ejecución
c. Otros: Ninguno
chmod 650 /home/fabrizio/yyyy
g) Borrar todos los archivos del directorio */tmp*
rm /tmp/* o me paro en tmp con (cd /tmp) y hago rm *
h) Cambiar el propietario del archivo */opt/isodata* al usuario *iso2010*
chown iso2010 /opt/isodata
i) 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/fabrizio/donde

9.- Indique que comando seria necesario ejecutar para realizar cada una de las siguientes acciones:

a) Ingrese al sistema como usuario "*root*"
su root

b) Cree un usuario. Elija para como nombre, por convención, la primer letra de su nombre seguida de su apellido. Asígnele una contraseña de acceso.
adduser fgelsi

c) ¿Qué archivos fueron modificados luego de crear el usuario y qué directorios se crearon?
Se modificaron los archivos de etc/shadow y se creó el directorio home/fgelsi

d) Crear un directorio en */tmp* llamado *cursada2010*
mkdir /tmp/cursada2010

e) Copiar todos los archivos de */var/log* al directorio antes creado.
cp /var/log/* /tmp/cursada2010

f) 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 adibello:users /tmp/cursada2010

g) 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 723 /directorio/ -R

h) Acceda a otra terminal virtual para loguearse con el usuario antes creado.
screen su fgelsi

i) Una vez logueado con el usuario antes creado, averigüe cual es el nombre de su terminal.
who "root **pts/0** 2012-05-05 19:39 (:tty1:S.0)"

j) Verifique la cantidad de procesos activos que hay en el sistema.
ps

k) Verifiqué la cantidad de usuarios conectados al sistema.
who

l) Vuelva a la terminal del usuario root, y envíele un mensaje al usuario anteriormente creado, avisándole que el sistema va a ser apagado.
write fgelsi
echo "El sistema será apagado, guarde sus trabajos"

m) Apague el sistema.
Shutdown

10.- Indique que comando seria necesario ejecutar para realizar cada una de las siguientes acciones:
a) Cree un directorio cuyo nombre sea su número de legajo e ingrese a él.
mkdir 306; cd 306

b) 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; apreto "i" para ingresar todo lo necesario, escape y :wq.

c) 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 017 LEAME

d) 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?
Se puede porque unix es case sensitive y distingue entre mayúsculas y minúsculas.

e) ¿Qué comando utilizaría y de qué manera si tuviera que localizar un archivo dentro del file system? ¿Y si tuviera que localizar varios archivos con características similares? Explique el concepto teórico y ejemplifique
Encontrar un archivo en el filesystem: sudo find / -name “archivo” –type f
Encontrar todas las imágenes en el filesystem: sudo find / -name “*.jpg” –type f

f) 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” –type f > /home/fabrizio/306/ejercicio_f

11.- Indique que 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:
1) mkdir iso
Crea el directorio iso

2) cd ./iso; ps > f0
Cambia al directorio iso, crea una lista de los procesos de iso y lo guarda en un archivo que se llama f0.

3) ls > f1
Lista el contenido de directorio actual (/iso) y lo guarda en el archivo f1.

4) cd /
Me voya raíz.

5) echo $HOME
Imprime el directorio home del usuario

6) ls -l > $HOME/iso/ls
Guarda el listado detallado del directorio actual en el archivo llamado ls contenido en el directorio iso

7) cd $HOME; mkdir f2
Va a home y crea el directorio f2

8) ls –ld f2
Lista los detalles del directorio f2

9) chmod 341 f2
Cambia los permisos del directorio f2. escritura y ejecución al dueño, lectura al grupo y ejecución a otros.

10) touch dir
Cambia (actualiza) la fecha al directorio.

11) cd f2
Se posiciona en el directorio f2

12) cd ~/iso
Cambia al directorio /home/user/iso

13) pwd >f3
Lista el directorio actual (/home/user/iso) y lo guarda en un archivo llamado f3

14) ps | grep 'ps' | wc -l >> ../f2/f3

Guarda en el archivo f3, sin sobreescritura, ubicado en el directorio f2, la cantidad de líneas que contienen la palabra “ps” entre la lista de procesos.

15) chmod 700 ../f2; cd ..
Modifica los permisos del directorio f2: total para usuario, nada para grupo y nada para otros. Vuelve a home.

16) find . -name etc/passwd

17) find / -name etc/passwd
Permiso denegado, no puede un usuario comun, buscar en la raíz. Sólo root.

18) mkdir ejercicio5
Crea el directorio ejercicio5 si que esta parado dentro su home.

19) **cp /home/iso/* /home/306**

20) **cp ejercicio5 /home/306**

b) 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)*

13.- Indique que comando/s es necesario para realizar cada una de las acciones de la siguiente secuencia de pasos (considerando su orden de aparición):
_ Cree un directorio llamado *logs* en el directorio */tmp*.
_ Copie todo el contenido del directorio */var/log* en el directorio creado en el punto anterior.
_ Empaquete el directorio creado en 1, el archivo resultante se debe llamar *misLogs.tar*
_ Empaquete y comprima el directorio creado en 1, el archivo resultante se debe llamar *misLogs.tar.gz*
_ Copie los archivos creados en 3 y 4 al directorio de trabajo de su usuario
_ Elimine el directorio creado en 1, *logs*.
_ Desempaquete los archivos creados en 3 y 4 en 2 directorios diferentes.

Me logueo como root
su –
contraseña
cd /tmp
mkdir logs
cp /home/root/var/log/ * /Home/root/tmp/logs
tar –cvf misLogs.tar /home/root/tmp/logs
tar –cvzf misLogs.tar.gz /home/root/tmp/logs
cp /home/root/tmp/logs/misLogs.tar /home/root
cp /home/root/tmp/logs/misLogs.tar.gz /home/root
cd /home/root/tmp
rmdir logs
cd
tar –xvf misLogs.tar > /home/root/directorio1
tar-xvzf misLogs.tar.gz > /home/root/directorio2