

Índice

Aclaraciones Generales	2
Comandos básicos de Unix	2
Comandos Extendidos de Unix	5
Ejecución automática	5
Dispositivos especiales	6
Cambio de hora	6
Lapsos y tiempos	7
Salida estándar y pipes	7
Scripting	8
Ejecución de procesos en background	8
IPC y Sincronización	9
El Kernel Linux	9
Administración del procesador	9
Administración de la memoria	11
Temas del Sistema Operativo	11

Aclaraciones Generales

Comandos Básicos de Unix

A continuación se presentan los comandos básicos utilizados y las explicaciones correspondientes:

- **pwd:** El comando `pwd` es una abreviatura de `Print Working Directory`. Como su nombre lo indica, su función es imprimir por pantalla el directorio actual.

Si se ejecuta el comando `cd /usr/bin` y luego se utiliza el comando `pwd`, justamente se imprime por pantalla `/usr/bin` ya que la función que realiza el comando `cd` es cambiar el directorio posicionandonos sobre el especificado, en este caso `/usr/bin`. Luego, el comando `pwd` nos indica la realización de este cambio de directorio.

Si ejecuta el comando `cd` sin especificar directorio al cual se desea cambiar y luego se ejecuta el comando `pwd`, entonces el mismo imprime por pantalla `/home/grupo5`. Esto sucede de esta manera, ya que si al comando `cd` no se le pasan parámetros, el mismo cambia el directorio al `home`. En nuestro caso, `/home/grupo5`. En la figura 1, se puede ver como funciona el comando.

- **cat:** El comando `cat` se utiliza, en este caso, para imprimir por pantalla el contenido del archivo especificado. De este manera, se puede ver el contenido del archivo `.profile`. Cabe destacar que el comando `cat` no solo se utiliza para este fin, el mismo también tiene otros usos como concatenar archivos en un archivo nuevo. Además, existen varios flags para modificar el comportamiento del comando, tal como además de imprimir el archivo, imprimir el número de línea. En la figura 2, se puede ver como se muestra el archivo `.profile` al utilizar el comando `cat`.
- **find:** Este comando se utiliza para buscar archivos dentro de directorios. Para ejecutar lo pedido se utilizó la sintaxis: `sudo find -name vmlinuz*`. El mismo indica que se deben buscar los archivos que coincidan en nombre con `vmlinuz` y luego el asterisco indica que puede seguir de diferentes maneras. Cabe destacar que se necesitan permisos de superusuario para poder explorar ciertas carpetas que tiene restricción.
- **mkdir:** El comando `mkdir` es una abreviatura de `Make Directory`, el mismo sirve justamente para crear directorios especificando el nombre

del directorio a crear. Este comando se puede utilizar para crear varios directorios a la vez, y también se puede hacer uso de distintos flags por ejemplo para crear directorios de solo lectura. Al utilizar el comando `mkdir tp` estando en `/home/grupo5`, se crea un directorio llamado `tp` dentro de donde se está posicionado. En la figura 3, se puede ver como el comando creo el directorio y luego se lo accede.

- `cp`: El comando `cp` proviene de la palabra `copy`, justamente la utilidad del mismo es poder copiar archivos. En el caso de ejecutar `cp /etc/passwd /home/grupo5/tp` lo que se está haciendo es copiar el archivo fuente (`/etc/passwd`) al directorio destino, que en este caso es el directorio `tp` creado recientemente. En la figura 4 se puede ver como al utilizar el comando `cp`, encontramos el archivo `passwd` dentro del directorio `tp`.
- `chgrp`, `chown` y `chmod`: Los comandos `chgrp`, `chown` y `chmod` provienen de `Change Group`, `Change Owner` y `Change Mode` respectivamente. Los mismos se utilizan justamente para cambiar el grupo de un archivo, el owner y los permisos de accesos del mismo.

Para utilizar los dos primeros comandos se debe especificar el nombre del grupo u owner nuevo y el archivo que se quiere modificar. En los casos pedidos la sintaxis sería, `chgrp grupo5 /home/grupo5/tp/passwd` y `chown grupo5 /home/grupo5/tp/passwd`. Luego de ejecutar estos dos comandos, el grupo y el owner de `passwd` pasan a ser `grupo5`. Esto se puede ver utilizando el comando `ls -l`.

Luego, para utilizar el comando `chmod`, se debe especificar cuales son los permisos a cambiar y a quien se quiere asignar o desasignar estos permisos. Para lograr lo pedido en este trabajo práctico las sentencias a utilizar serían: `chmod u+rwX passwd`, `chmod g+rx passwd`, `chmod o+x passwd`. El primer comando es para cambiar el user, y le asigna (+) `Read`, `Write` y `eXecute`. El segundo comando le asigna `Read` y `eXecute` al Grupo. Por último, el tercer comando asigna `eXecute` a los demás (`Others`). Cabe destacar que como el archivo ya traía permisos de `Read` para los `Others`, se quitaron los mismos utilizando `chmod o-r passwd`.

En la figura 5 se puede ver la utilización de todos estos comandos y se miran los atributos del archivo mediante el comando `ls -l`, para ver como van cambiando el grupo, el owner y los permisos a medida que se van utilizando los comandos explicados.

Cabe destacar que el comando `chmod` se utilizó varias veces para ir cambiando de a poco los permisos según sean para el usuario, para el

grupo o para los otros. Sin embargo, esto se puede realizar utilizando una sola vez el comando `chmod`. La sintaxis para el mismo sería `chmod 751 passwd`. El numero 751 esta es una manera de codificar los permisos, el 7(111) corresponde al user y esta poniendo en 1 los 3 permisos; el 5(101) es para el grupo, dado un orden predispuesto, lo que hace esto es setear en 1 los permisos de Read y eXecute, que son el primero y el tercero respectivamente, y setea en 0 el segundo permiso(Write). Por último, el 1(001) corresponde a setear el permiso de eXecute para los Others.

- `grep`: El comando `grep` es para buscar expresiones regulares dentro de un archivo. En el caso pedido lo que se quiere es buscar la expresion `localhost` dentro del archivo `/etc/hosts`. Para esto se utiliza el comando de la siguiente manera: `grep localhost /etc/hosts`. De esta manera, se visualiza(figura 6) como el comando `grep` imprime las lineas del archivo `hosts` en donde aparece la expresión `localhosts`.

Luego, se pide realizar la misma acción pero sobre todos los archivos del directorio `/etc`. Para esto se utiliza la siguiente sintaxis: `grep -r POSIX -- grep -v Binary`. Lo que hace este comando es primero buscar recursivamente (`-r`) los archivos que concuerden, luego con estos archivos encontrados se hace otra busqueda con `-v` que indica que hay que evitar los que coincidan con `Binary`.

- `passwd`: El comando `passwd` se utiliza para manejar contraseñas. El mismo posee varias flags para distintas opciones. Sin embargo, si se utiliza sin ninguna opción, su función por default es cambiar el password del user. Primero pide escribir el viejo password (`mesaverde`) y luego pide asignar un nuevo password y confirmarlo (`mesaverde2`).
- `rm`: El comando `rm` proviene de la palabra `remove`. El mismo se puede utilizar con diferentes opciones si se quieren eliminar directorios recursivamente por ejemplo. En el caso pedido, solo hay que eliminar el archivo `passwd`, por lo que la sintaxis es `rm passwd`. En la figura 8 se puede ver el cambio de password y la eliminacion del archivo conjuntamente.
- `ln`: El comando `ln` se utiliza para crear links entre archivos. Para utilizar el comando `ln` se debe especificar el archivo fuente y el archivo destino. Cabe destacar que el uso de este comando sin ningún flag creara un hard link. Para realizar lo pedido en el trabajo práctico la sintaxis utilizada fue la siguiente: `ln /etc/passwd /tmp/contra1` y `ln /etc/passwd /tmp/contra2`. En la figura 9 se puede ver como se utilizan estos dos

comandos y luego, realizando `ls -l`, se puede ver que ahora hay 3 links al archivo `/etc/passwd`.

Para realizar un soft link o link simbolico se debe agregar la opción `-s` al utilizar el comando. En la figura 9 se puede ver la utilización del comando `ln -s /etc/passwd /tmp/contra3` y luego se puede ver como no cambia la cantidad de hard links usando el comando `ls -l`.

- **mount:** Para realizar lo pedido, debemos utilizar el comando `mount` de la siguiente manera: `mount /dev/cdrom` de esta manera, estamos montando el cdrom. Luego para poder ver los filesystems que estan montados debemos ver el archivo `/etc/mtab`. En la figura 10, se puede ver como se monta la unidad de cd y luego haciendo `cat /etc/mtab` se muestran los filesystems montados.
- **df y ps:** Los comandos `df` y `ps` sirven para mostrar el espacio libre de los filesystems y los procesos que se tienen ejecutando. El comando `ps`, se utiliza con la opcion `-A` para listar todos los procesos y ahi se pueden notar cuales son del sistema y cuales son de los usuarios. En la figura 11, se puede ver la utilización de los mismos.
- **umount:** El comando `umount` sirve para desmontar el dispositivo de cdrom utilizando la sintaxis `umount /dev/cdrom`. En la figura 12 se puede ver como se utiliza este comando y como luego al utilizar el comando `df`, se ve como el filesystem del cd ya no se muestra.
- **uptime y uname:** Estos comandos se utilizan para saber cuanto tiempo lleva el sistema corriendo y para saber que version del kernel. En la figura 13 se puede ver la utilización del comando `uptime` y luego el comando `uname -a` para ver toda la información, incluyendo la información sobre el sistema operativo.

Comandos Extendidos de Unix

Ejecución automática

- Para poder imprimir un hola por pantalla cada vez que un usuario se loguea, se debe modificar el archivo pertinente. En este caso, para que el hola aparezca cuando se loguea cualquier usuario se debe modificar el archivo `/etc/.profile`. En dicho archivo, se debe agregar una linea que especifique la acción requerida, la sintaxis es simplemente `echo Hola`. Luego, cada vez que un usuario se loguee, el archivo `.profile` se corra y el mismo indicará que se imprima `Hola` por pantalla.

- Buenos Dias: explicar del script en init.d y del update-rc
- Para poder imprimir un Adios por pantalla cada vez que un usuario se desloguea el proceso es diferente al realizado en el primer item de este punto. La manera debe ser diferente debido a que, a diferencia del fichero profile, no se encontro un fichero que se corra automaticamente al hacer logout.

Para poder indicarle al sistema que se quiere correr un archivo al hacer logout lo que se realizó fue modificar nuevamente el fichero `/etc/profile` con la siguiente linea de comando: `trap '. /etc/.logout;exit' 0`. Lo que hace el comando trap es darle una directiva al sistema operativo, en este caso lo que hace es darle la directiva de que cuando se reciba una señal de exit(logout) se corra el fichero `.logout`.

Luego, se creo un archivo `.logout` con los scripts que se quisiesen correr a la hora del logout, en nuestro caso el archivo consta solamente de una linea que indica la impresion de una linea de Adios. La sintaxis dentro del archivo es simplemente `echo Adios`.

Por último lo que se realizó fue hacer `chmod 755 .logut`, esto se hizo para que el archivo tuviese permisos de ejecución.

- Hasta la vista baby ver bien como hacerlo

Dispositivos Especiales

Cambio de Hora

Para esta sección lo primero que hay que hacer es crear un alias en el fichero deseado. Para eso se debe editar este archivo y poner algun alias como puede ser `alias bla = ls`. Luego hay que mantenerle el timestamp original, para esto es necesaria haberse guardado el timestamp antes de modificar el archivo, para conseguir el timestamp se utiliza el comando `stat`. Luego de modificarlo, se utiliza el comando `touch` para modificar el timestamp y poner el original, el mismo se usa con el flag `-m` que justamente indica que vamos a tocar el timestamp de modified.

Luego, para cambiar el horario del sistema linux se utiliza el comando `date -set=` y la fecha que se quiere poner, si el cambio es hacia atras no se nota ningún problema porque el sistema operativo reacomoda su hora con la hora del hardware cada cierto intervalo de tiempo. Si se hace un `date` rapidamente despues del cambio, se vera la fecha que fue modificada, pero si se deja un lapso de tiempo, el sistema se actualiza automaticamente. Lo

que se podría hacer es utilizar el comando `hwclock` para setear el horario del hardware.

Lapsos y Tiempos

Para conocer toda la información que se pide en esta sección hay varios comandos. En primer lugar se puede utilizar el comando `uptime` de la misma manera que se utilizó en la primer sección de este trabajo, para tener información de cuando se prendió la maquina y cuantos usuarios hay logueados.

Sin embargo, el comando que más información pertinente nos da, es el comando `last`. El mismo lista todos los últimos movimiento del sistema, mostrando quien se logueo, en que terminal y a que hora. Así como otros datos pertinentes.

Salida Estándar y Pipes

STDOUT:

En primer lugar se utilizó el siguiente comando para volcar la información pedida: `ls -R >> /home/grupo5/tp/config`. El mismo se utilizó estando en el directorio `/etc`, de esta manera el comando `ls -R` lo que hizo fue listar recursivamente todos el contenido del directorio y su subdirectorios. Mientras que `>>` lo que hizo fue volcar esta información al archivo especificado.

Para obtener la información sobre la cantidad de líneas, palabras y caracteres se utilizó el comando `wc` con las opciones necesarios en cada caso (`-l`, `-w`, `-m`). El resultado fue: 665 líneas, 593 palabras y 7305 caracteres.

En tercer lugar, se buscar ordenar el contenido de `/etc/passwd` y ponerlo al final del archivo `config` anteriormente nombrado. Para ordenar se utilizó el comando `sort`, mientras que para volcar en el archivo nuevamente se utilizo `>>`. De esta manera la sintaxis utilizada fue: `sort /etc/passwd >> /home/grupo5/tp/config`

Por último se procedió a contar las líneas, palabras y caracteres del archivo de la misma forma que se habia realizado anteriormete, el resultado fue: 688 líneas, 621 palabras y 8238 caracteres.

Pipes: Para realizar lo pedido, sin utilizar archivos temporales, se ejecutó la siguiente linea de comando que hace lo que el ejercicio pide, en el orden requerido. La sintaxis es: `ls -al /usr/bin/a* | grep apt | wc`. La misma es una combinación de los comandos utilizados en puntos anteriores, para lograr todos los efectos requeridos.

Scripting

3.3.1: En este punto se debe hacer un script que cada 5 minutos diga HOLA por pantalla. Para esto, se utilizaron dos resoluciones diferentes, una que es totalmente automática y la otra semiautomática.

Por un lado, se realizó un script de shell que, una vez ejecutado, corriese un ciclo infinito; dentro de este ciclo, el script espera 300 segundos mediante el comando `sleep 300` y luego hace un `echo HOLA` para imprimir por pantalla la palabra requerida. Esta solución requiere que el usuario ejecute el script. Dicho script se encuentra en la carpeta `/home/grupo5/tp`.

Por otro lado, se encuentra otra solución totalmente automática, es decir que no requiere que el usuario ejecute el script. La misma consiste en agregar una entrada a la crontable que indique que se debe imprimir HOLA cada 5 minutos. Cabe destacar que la distribución de ubuntu utilizada para este trabajo, no cuenta con el cron por lo que fue necesario instalarlo. Una vez instalado, se agrega en la tabla la entrada correspondiente. Dicha entrada tiene la siguiente sintaxis: `* /5 * * * * root echo HOLA >> /dev/console`. Esta sintaxis dice que, cada 5 minutos, a cualquier hora, en cualquier día, en cualquier mes y en cualquier día de la semana; el usuario root corra el comando `echo HOLA` y lo imprima en la consola.

Para hacer lo mismo pero a una hora determinada, se debe explicitar la hora requerida en la entrada de la crontable. Por ejemplo, en la imagen se encuentra la entrada `25 7 * * * root echo HOLA >> /dev/console`. Esta entrada tiene la misma acción que la anterior, pero se ejecuta todos los días a las 7 y 25.

3.3.2

Hay que robarse este script!!!

Ejecución de Procesos en Background

En primer lugar, se procedió a transcribir el programa loop, se guardó el mismo en el directorio `/home/grupo5/tp`. Una vez transcrita, se procedió a compilar el mismo utilizando el gcc.

En primer lugar el proceso se corrió en foreground. Esto quiere decir que el proceso toma el control de la terminal que se está utilizando, es decir que se corre en un primer plano. El problema con este proceso es que el mismo contiene un ciclo infinito que escribe en la pantalla. Como el proceso está corriendo en foreground, y el ciclo es infinito, el mismo nunca devuelve el control de la terminal y sigue ensuciando la consola sin poder realizar otra

acción. Luego, se mató el proceso mediante Ctrl+C para poder interrumpirlo y así volver a tomar el control de la terminal.

En segundo lugar, el proceso se corrió en background. Esto quiere decir que el proceso queda corriendo en un segundo plano, mientras que el usuario sigue teniendo control de la terminal. Para solucionar el hecho de que el proceso ensucia la terminal, el mismo se corrió mandando su salida a /dev/null. Al realizar esta acción, el proceso queda corriendo, pero se puede seguir haciendo uso de la terminal.

Como este proceso no está corriendo en foreground, Ctrl+C para matarlo no servirá. Es por esto que se debe matar el proceso mediante el comando kill, el mismo necesita del ID del proceso que se quiere matar. Este ID se obtiene al correr el programa, ya que al correr un proceso en background, por consola se devuelve el número de proceso asignado para, por ejemplo, saber que número tiene al querer matarlo.

IPC y Sincronización

Facu acá me tenes que decir que hay que poner y si el código está o que carajo...

El Kernel Linux

Administración del procesador

En cuanto a la administración del procesador de Linux refiere, el sistema operativo maneja los procesos o tareas. Los procesos o tareas, son partes de un programa, o un programa entero, que cumplen alguna función específica mediante el manipulamiento de ciertas variables y funciones.

La administración del procesador justamente se basa en ver cómo manejar estos procesos para lograr que todos funcionen correctamente en tiempo y forma.

Los procesos en un sistema operativo de tipo Linux tienen varios estados:

- **Running:** Es cuando el proceso se encuentra ejecutando, tiene todos los recursos que necesita para poder funcionar correctamente.
- **Sleeping:** Estos procesos se encuentran dormidos, puede ser porque son procesos interactivos que no se utilizan hace cierta cantidad de tiempo por ejemplo; o bien porque es un proceso que está esperando algún suceso para continuar.

- **Zombie:** Un proceso zombie es aquel que ya no esta en funcionamiento pero que tiene una entrada en la tabla de procesos. Esto puede suceder cuando un proceso hijo termina, pero el padre todavia es capaz de ver su estado de salida por algún motivo en particular.
- **Stopped:** Son procesos detenidos totalmente, pero que pueden ser reiniciados para continuar con su ejecución.
- **Uninterruptible sleep:** Son procesos que se encuentran durmiendo pero que no pueden ser interrumpidos, generalmente tienen que ver con entrada/salida del sistema operativo mismo.
- **Dead:** proceso terminado que puede seguir apareciendo en el listado de procesos.

Si se esta trabajando en un sistema operativo con interfaz gráfica como ubuntu, se puede utilizar alguna herramienta como el `gnome-system-monitor` para ver todos estos procesos. A su vez, a un costado del mismo, aparecerá el estado del proceso, que será alguno de los anteriormente explicados.

Por otro lado, en un sistema sin interfaz gráfica como el utilizado para este trabajo práctico se puede utilizar el comando `ps` con algún flag que nos de toda la información posible. Usualmente en estos casos aparece una letra que indica el estado del proceso: R de running, S de sleeping, Z de zombie, T de stopped, U de Uninterruptible, X de dead.

Una vez explicados como son los procesos en linux, es importante ver como se organizan estos para su ejecución. Linux se basa en lista de prioridades para poder manejar sus procesos. Para esto, linux divide sus tareas en dos grupos:

- Procesos del sistema
- Procesos de usuarios

Procesos del sistema: Para los procesos del sistema linux posee 100 listas de prioridad, donde a menor número, mayor prioridad tiene el proceso. Para este tipo de procesos linux posee dos políticas diferentes para manejarlos:

- **FIFO:** First in, First out. Es simplemente una cola, donde el primer proceso que entra, es el primer proceso que sale de la cola de prioridad.
- **Round Robin:** Es un sistema que se basa en darle un tiempo de ejecución a cada uno de los procesos. Es decir, se entrega un quantum a cada proceso, si el proceso finaliza o el quantum se agota, el proceso es desalojado.

Procesos de usuarios: Para estos procesos linux posee 40 listas de prioridad, numeradas desde -20 hasta 19. Para estos procesos el sistema operativo se encarga de alternarlos de manera que no haya ninguno que se encuentre mucho tiempo inactivo. Para esto el sistema se encarga de penalizar los procesos que tienen mucho tiempo de cpu, y trata de darle más prioridad a los procesos que tienen más entrada/salida, que se supone que estuvieron bloqueados una mayor cantidad de tiempo.

Por defecto, el proceso cuando se ejecuta tiene nivel de prioridad cero. Si se quiere que un proceso tenga una prioridad diferente, se puede utilizar el comando `nice` para realizar un cambio de prioridad. Si la prioridad a asignar es positiva, es decir una prioridad menor, se puede hacer directamente. Si, por el contrario, se quiere asignar una prioridad mayor, es decir negativa, se necesitan permisos de superusuario.

En el caso que un proceso ya se encuentre en ejecución, se puede cambiar la prioridad del mismo sin necesidad de matarlo. Esto se puede hacer con el comando `renice`, que reasigna prioridad en tiempo de ejecución.

Administración de la memoria

La administración de la memoria en linux, al igual que en la mayoría de los sistemas operativos modernos, es pagina por demanda debido a que resulta efectivo poder simular mayor cantidad de memoria ram utilizando otro dispositivo como el disco duro. Linux posee paginas de 4kb o 4mb dependiendo la arquitectura del procesador y contiene un espacio especial de memoria en el disco denominado memoria swap donde justamente hace el intercambio entre las paginas que se encuentran en memoria principal y las que se encuentran en memoria secundaria según las necesidades requeridas. Esta memoria swap se puede asignar manualmente al instalar el sistema operativo y tiene como consecuencia una mejor o peor eficiencia del manejo de la memoria virtual.

Cabe destacar, que al momento de realizar los intercambios, el sistema operativo linux toma una política LRU, last recently used, es decir que la pagina elegida para ser removida de memoria principal es la que más tiempo lleva sin ser referenciada.

Sistema de archivos

El sistema de archivos de linux es, basicamente, el heredado del sistema UNIX. El mismo es un sistema de archivo jerarquico, donde todo se toma o como un directorio o como un archivo. Este sistema de archivo es jerarquicos, ya que los directorios con sus subdirectorios correspondientes tiene

una topología de árbol donde se encuentra el directorio root y luego todos los demás, colgando de este primero.

Es importante destacar que para el filesystem de linux, todo es un archivo o un directorio, es por esto que se puede acceder por ejemplo a los dispositivos de la misma manera que a cualquier otro directorio. Por ejemplo, si se escribe información en el directorio `/dev/dsp`, se podrá escuchar esta información por los parlantes dado que este directorio corresponde al dispositivo de sonido.

Al heredar este filesystem de UNIX, el mismo posee las mismas propiedades a bajo nivel, es decir que los medios de representación son los i-nodos. Además, el sistema posee un Virtual file system que se encarga de ver los filesystem que se pueden ir montando sobre el original.

Temas del Sistema Operativo