

Introducción básica a GNU/Linux

Agregar una cuenta de usuario

Es importante que en tu uso cotidiano del sistema, no utilices el superusuario root todo el tiempo. En vez de ello deberás crear un usuario normal. Para crear un usuario normal deberás de utilizar el comando “adduser” seguido del nombre de usuario nuevo que quieres crear.

```
root@maquina: # adduser pepito
```

El comando “adduser” te hará una serie de preguntas sencillas sobre el nuevo usuario.

Consolas virtuales

En la gran mayoría de las instalaciones GNU/Linux, cuentas con más de una consola virtual. Para cambiar de consola virtual puedes presionar las teclas Alt+Fn, donde “Fn” es cualquiera de las teclas F1, F2, F3... etc.

Por lo general, en un sistema en donde se está ejecutando un ambiente gráfico (X Window), se puede acceder a el en la consola virtual 7, es decir, presionando las teclas Alt+F7.

Apagado y reseteado del sistema

Para apagar correctamente tu computadora deberás de utilizar el comando “shutdown” seguido de los parámetros “-h now”. Por ejemplo:

```
root@maquina: # shutdown -h now
```

El proceso de apagado iniciará, espera a ver el mensaje “System halted” y entonces ya puedes presionar el botón de apagado en tu equipo.

Para resetear el sistema también se utiliza el comando “shutdown” pero ésta vez seguido de los parámetros “-r now”. Ejemplo:

```
root@maquina: # shutdown -r now
```

También es posible resetear tu computadora presionando las teclas Ctrl+Alt+Supr.

Diferencias del prompt cuando el usuario y es root o normal

Es importante que sepas diferenciar cuando estás usando el superusuario root o un usuario normal. Así puedes evitar posibles confusiones al momento de querer ejecutar un comando. Cuando estás como el usuario root el prompt tendrá al final el caracter #, y cuando se trata de un usuario normal, el caracter será un signo de pesos \$. Ejemplo:

```
root@maquina: #
```

```
...
```

```
pepito@maquina: $
```

Comandos básicos de GNU/Linux

A continuación se mencionan algunos de los comandos más utilizados:

- `pwd` – Despliega el directorio actual.

- `which comando` – Despliega la localización del comando especificado.
- `man comando` – Despliega el manual sobre el comando especificado.
- `apropos palabra` – Encuentra comandos relacionados con la palabra especificada.
- `whatis comando` – Muestra una descripción corta del comando especificado.
- `ls` – Despliega un listado de los archivos en el directorio actual.

Existen varias opciones que especifican cómo se muestra el listado.

`ls -l` Muestra el listado en formato vertical.

`ls -a` Muestra también archivos ocultos.

`ls -la` Combinando las dos opciones anteriores.

- `mkdir directorio` – Crea el directorio especificado en la carpeta actual.
- `rmdir directorio` – Elimina del disco el directorio especificado.
- `cd directorio` – Cambiarse al directorio especificado.

Algunos ejemplos:

`cd ~` Nos cambiamos al directorio que corresponde al home (~) del usuario.

`cd /` Nos vamos directamente a la raíz del sistema de archivos.

`cd hola/` Nos cambiamos a la carpeta hola que se encuentra en la carpeta actual.

`cd /var/log/` Nos cambiamos a la carpeta var que se encuentra desde la raíz del sistema de archivos, independientemente de donde nos encontremos actualmente.

- `cat archivo` – Despliega a la pantalla el contenido del archivo especificado.
- `cp original copia` – Copia un archivo.
 - `cp hola.txt adios.txt`
 - `cp hola.txt adios/hola.txt` También se puede copiar a un directorio diferente.
 - `cp -r carpeta1/ carpeta2/` Se usa la opción -r para copiar carpetas.

- `rm archivo` – Borra del disco el archivo especificado.
- `mv original copia` – Mueve un archivo a otro lugar.
 - `mv hola.txt adios/hola.txt` Mover un archivo a otra carpeta.
 - `mv hola.txt adios.txt` Para cambiarle el nombre a un archivo, solamente que “mv” en la misma carpeta.
- `chmod permisos archivo` – Modifica los permisos del archivo especificado.

Ejemplos:

`chmod 644 hola.txt` Permiso de lectura y escritura para el dueño del archivo, permisos de sólo lectura para el resto de los usuarios.

`chmod 600 hola.txt` Permiso de lectura y escritura para el dueño del archivo, y ningún permiso para ningún otro usuario. Sólo el dueño del archivo lo puede leer y modificar.

`chmod 755 hola.sh` Permisos de lectura, escritura y ejecución para el dueño del archivo, permisos de ejecución y lectura para el resto de los usuarios.

- `top` – Muestra información general y los procesos activos en el sistema de forma dinámica.
- `ps` – Realiza un listado de los procesos activos en el sistema.

Algunos ejemplos de opciones que acepta `ps` son:

`ps` Por default, `ps` solo muestra los procesos de tu usuario que están ligados a una terminal.

`ps a` Muestra los procesos de todos los usuarios del sistema.

`ps x` Muestra también los procesos que no están ligados a una terminal.

`ps u` Muestra los resultados de forma mas amigable.

`ps aux` Combinamos las opciones anteriores para ver todos los procesos de todos los usuarios en un formato fácil de leer.

Existen muchas otras opciones, las cuales puedes ver con “`man ps`”.

- `kill proceso` – Detiene un proceso que se encuentra actualmente ejecutándose.

Se utiliza `kill` para enviar una señal a un proceso. La señal es interpretada por el proceso y se espera que actúe de acuerdo a dicha señal. Por ejemplo, para indicar a un proceso con el `pid` 345 que debe de resetearse, se le envía la señal `SIGHUP`:

```
kill -HUP 345
```

Para pedirle a un proceso que termine su ejecución, se le manda la señal `SIGTERM`, la cual es la señal por defecto, por lo que no es necesario especificarla:

```
kill -TERM 345
```

```
kill 345
```

 Es equivalente a `kill -TERM`

Si el proceso no responde se puede enviar la señal `SIGKILL`, la cual no es interceptable por el proceso y que ocasiona que el sistema operativo mate al proceso.

```
kill -KILL 345
```

- `grep patron archivo` – Busca un patron en el archivo o archivos especificados.

Ejemplo, para buscar si el archivo `hola.txt` contiene el texto “`pepito`”

```
grep pepito hola.txt
```

El comando anterior va a desplegar todas las líneas del archivo que contengan el patrón especificado. Otro ejemplo mas útil es buscar un texto en un conjunto de archivos:

```
grep pepito *.txt
```

Con el comando anterior se busca el texto `pepito` en todos los archivos con la extensión `.txt` en la carpeta actual. Se desplegará todas las líneas que contengan el patrón especificado.

Sin embargo, es difícil saber en que archivo se encontraba el texto que se estaba buscando, para resolver eso se puede utilizar la opción `-H`, la cual despliega el nombre del archivo al inicio de la línea desplegada con el texto que se está buscando:

```
grep -H pepito *.txt
```

- `gzip archivo` – Comprime el archivo especificado.

- `gunzip archivo.gz` – Descomprime un archivo previamente comprimido con `gzip`.
- `bzip2 archivo` – Comprime el archivo especificado. Ofrece una mejor compresión que `gzip`.
- `bunzip2 archivo.bz2` – Descomprime un archivo previamente comprimido con `bzip2`.
- `tar` – Utilería para empaquetar múltiples archivos y directorios en un solo archivo.

El comando `tar` es útil para cuando se tienen muchos archivos y se desea copiarlos a otra computadora. Con `tar` se puede crear un sólo archivo que contenga todos los archivos y carpetas que se desean manejar. Por ejemplo, si se desea empaquetar la carpeta `proyecto` y todo su contenido en un archivo `miproyecto.tar`, se puede hacer lo siguiente:

```
tar cvf miproyecto.tar proyecto/
```

Hay que hacer notar que éste empaquetado NO hace ninguna compresión, sin embargo es posible indicarle a `tar` que deseamos hacer una compresión del archivo final llamando ya sea `gzip` o `bzip2`. Ejemplo:

```
tar cvfz miproyecto.tar.gz proyecto/
```

Que es exactamente lo mismo que hacer lo siguiente:

```
tar cvf miproyecto.tar proyecto/
```

```
gzip miproyecto.tar
```

Para realizar la compresión con `bzip2` se utiliza la opción `j` en lugar de `z`:

```
tar cvfj miproyecto.tar.bz2 proyecto/
```

Para desempaquetar el archivo en la carpeta actual se utiliza la opción `x` en lugar de la `c`:

```
tar xvf miproyecto.tar
```

En el caso de que no se hizo ninguna compresión.

```
tar xvfz miproyecto.tar.gz
```

Archivo comprimido con `gzip`.

```
tar xvfj miproyecto.tar.bz2
```

Archivo comprimido con `bzip2`.

Variables de entorno

Las variables de entorno sirven para indicar o modificar el comportamiento de ciertos comandos, por ejemplo la variable de entorno `PATH` indica dónde se buscarán los comandos cuando se desea ejecutar uno. Para ver el valor de una variable de entorno se utiliza el comando “`echo`”:

```
pepito@maquina: $ echo $PATH
```

Para establecer el valor de una variable de entorno se utiliza el comando “`export`”, por ejemplo:

```
pepito@maquina: $ export MI_NOMBRE=pepito
```

```
pepito@maquina: $ echo $MI_NOMBRE
```

Para ver una lista con las variables de entorno definidas se utiliza el comando “`env`”:

```
pepito@maquina: $ env
```

Control de ejecución

Cuando se está ejecutando una aplicación en la terminal, es posible mandar dicha aplicación “al fondo” para poder así realizar otras tareas. Para mandar al fondo (background) una aplicación que está actualmente corriendo en tu terminal, presiona las teclas `Ctrl+Z`. La aplicación será enviada al

background y su ejecución será suspendida.

Para ver una lista de aplicaciones que han sido suspendidas se utiliza el comando “jobs”.

Para hacer que la ejecución de la aplicación continúe en el background se utiliza el comando “bg”, indicando el número de la aplicación, el cual es el número que se obtiene con el comando “jobs”.

Para traer al frente una aplicación que se encuentra en el background, se utiliza el comando “fg”, indicando el número de la aplicación.

Para lanzar una aplicación nueva, y hacer que se vaya directamente al background y continúe su ejecución ahí, se pone un & al final del comando. Ejemplo:

```
pepito@maquina: $ bzip2 archivo_grande.txt &
```

Condiciones de ejecución

El condicional && ocasiona que se ejecute primero un comando, y sólo cuando éste haya acabado de manera exitosa, se ejecuta un segundo comando. Por ejemplo:

```
pepito@maquina: $ tar archivo.tar carpeta/ && bzip2 archivo.tar
```

El condicional || ocasiona que se ejecute primero un comando, y cuando éste acabe se ejecute un segundo comando, independientemente del resultado del primero. Ejemplo:

```
pepito@maquina: $ gzip hola.txt || gzip adios.txt
```

Redirección del resultado de un comando

Para hacer que el resultado (salida a stdout – que normalmente es la pantalla) de un comando se introduzca como entrada para un segundo comando se utiliza el control de flujo “|”, normalmente llamado pipeline. Por ejemplo:

```
pepito@maquina: $ cat archivo.txt | grep hola
```

Para mandar a un archivo el resultado de la ejecución de un comando se utiliza el redireccionamiento “>”, por ejemplo:

```
pepito@maquina: $ ps aux > procesos.txt
```

En el ejemplo anterior se reemplaza el contenido del archivo procesos.txt con el resultado del comando utilizado. Se puede utilizar el redireccionamiento “>>” para mandar al final de un archivo y no reemplazar el contenido ya existente:

```
pepito@maquina: $ ps aux >> procesos.txt
```

Aliases de comandos

Una forma de evitar teclear muchos caracteres para comandos utilizados frecuentemente es el uso de “alias”. Por ejemplo, sabemos que para desplegar el contenido de la carpeta actual en formato vertical utilizamos el comando “ls -la”, se puede declarar un comando “ll” que sea un sinónimo del comando “ls -la”.

```
pepito@maquina: $ alias ll="ls -la"
```

Ligas con mas información

<http://www.gulag.org.mx/>

<http://www.linuxlaguna.com/>

<http://www.cofradia.org>

<http://www.barrapunto.com/>

<http://www.unixmexico.org/>

<http://es.tldp.org/>

<http://www.demiurgo.org/doc/cubasl/cubasl.html>

<http://www.debian.org/doc/index.es.html>

<http://www.google.com/search?q=curso+basico+de+linux>

<http://www.linuxquestions.org/>

<http://www.faqs.org/docs/>

Comentarios y dudas

Jorge Gajón <gajon@gajon.org>