Guía 1: Manejo básico en la línea de comandos

1. GNU/Linux es un sistema operativo con muchas características, una de ellas es estar diseñado para ser utilizado por múltiples usuarios. Sin embargo, existe un usuario que está por encima del resto, que es capaz de manejar el resto de los usuarios. Este es el usuario `root` que además tiene un completo control del sistema y de los usuarios.

   a) ¿Para qué fines es necesario utilizar la cuenta de `root`?

   - Para fines administrativos, ya sea de usuario o de sistema. Eso contempla editar archivos de configuración, manejo de paquetes de instalación, manejo de programas instalados, actualizaciones, agregar/modificar/eliminar usuarios, niveles de ejecución, administración de archivos de grupo, etcétera.

   b) Si necesito leer las noticias de `www.lun.cl` (Las Últimas Noticias), ¿por qué no es recomendable utilizar la cuenta de `root`?, refiérase al punto de vista de seguridad.

   - `root` tiene permiso de acceso a todas las partes del sistema, y esto puede ser explotad por algún malware instalado en alguna página web. También hay que tomar en cuenta que para eso requiero (en cierto punto) usar el entorno gráfico, que no está diseñado para ser seguro, ni menos para ser utilizado por `root`.

   c) Como un usuario distinto de `root`, ¿por qué no es posible eliminar un archivo ubicado e el directorio `/etc/`?

   - Porque `/etc` tiene permisos de escritura sólo para `root`. Eso es verificable haciendo `ls -l /etc`.

2. Los comandos de Linux pueden provocar rechazo en una primera instancia. Sin embargo, con la práctica se vuelve una necesidad escencial utilizar una consola para realizar cualquie operación en el sistema.

   a) El comando `ls` lista los archivos presentes en el directorio actual, pero, ¿qué parámetro debo pasarle al comando para ver los permisos, el nombre de usuario y grupo que pertenec el archivo, tamaño, fecha, etc.? (Ayuda: `man ls`).

   - `ls -l`. El parámetro `-l` muestra el listado de archivos de forma larga.

   b) ¿Qué hace el comando `ls -la`? ¿Qué significan los archivos de la forma `.nombre-`

`archivo`?

- `ls -la` significa que mostrará el listado de archivos de forma larga (`-l`) y además mostrando **todos** los archivos (`-a`, de *all*), incluyendo los archivos ocultos.

  Los archivos ocultos, sean archivos, links o carpetas, comienzan con un punto. Por ejemplo, `.nombre-archivo` es un archivo oculto.

c) ¿Qué muestra el comando `ps`? Investigue que despliega el comando `ps aux`.

- `ps` muestra información acerca de los procesos que se encuentran corriendo en la máquina, dependiendo del usuario que lo ejecute.

  `ps aux` muestra información de los procesos completa, de todos los usuarios, de forma extendida.

d) ¿Qué hace el comando `init`? ¿Qué significa cada nivel del sistema? (Ayuda: revise el archivo `/etc/inittab`).

- El comando `init` permite cambiar de **nivel de ejecución** o **runlevel** dentro del equipo. La idea de esto es poder lograr diferentes niveles de administración y servicios para la máquina, y cómo esto afecta al entorno de múltiples usuarios.

  Los diferentes runlevels son:

  - 0: `halt`. Apaga el equipo.
  - 1: Modo monousuario. Sólo `root` tiene acceso a este nivel.
  - 2: Modo multiusuario sin red. Ahora ya hay acceso para múltiples usuarios.
  - 3: Modo multiusuario completo. Con acceso a la red.
  - 4: Este runlevel es personalizable, y usualmente se deja sin utilizar para que el sysadmin lo configure como quiera.
  - 5: Modo multiusuario + `X.org`. Aquí ya hay interfaz gráfica servida por `X.org`.
  - 6: `reboot`. Reinicia la máquina.

e) ¿Qué hacen los siguientes comandos: `reboot`, `chkconfig` y `pwd`?

- `reboot` reinicia la máquina. Sólo `root` puede ejecutar este comando.
- `chkconfig` sirve para administrar los diferentes servicios que se ejecutan y en qué runlevels. Sólo `root` puede ejecutar este comando.
- `pwd` significa *print working directory* y mostrará la ruta **absoluta** del directorio en el cual nos encontremos trabajando ahora. Es ejecutable por cualquier usuario.

f) El comando `echo` sirve para imprimir un comentario por pantalla, por ejemplo, `echo Hola Mundo`, imprimirá `Hola Mundo` en la consola. Ahora, ¿qué imprime `echo $HOME`?. Investigue que significan las variables de entorno.

- `echo $HOME` mostrará el contenido dentro de la **variable de entorno** `$HOME`.

  Las **variables de entorno** son variables que se establecen para ciertos objetivos dentro de la terminal que se ejecuta, y son útiles en la creación de *scripts* para la automatización de tareas. Por ejemplo, la variable `$HOME` contiene la ruta absoluta donde se ubica el directorio de usuario del **usuario que lo ejecuta**.

  Existen otras variables de entorno como `$PATH`, `$EDITOR`, así como también se pueden crear propias variables mediante el comando `export $MI_VARIABLE="mis valores"`.

g) ¿Por qué para eliminar un directorio con el comando `rm` es necesario añadir el parámetro `-r`, es decir, `rm -r`?

- `rm` establece que para poder borrar un directorio, primero debe borrarse su contenido. El parámetro `-r` hace que el borrado sea recursivo, entonces borrando primero el contenido del directorio, y luego borrando el directorio. Esto es completamente independiente de la profundidad del directorio.

h) ¿Qué hace el comando `alias`? Ejecútelo en una consola e interprete los resultados.

- `alias` sirve para poder asociar un nombre arbitrario a un comando que sea demasiado largo. Si se ejecuta sin parámetros, `alias` mostrará los *alias* ya configurados.

  Para crear un nuevo *alias* se debe ejecutar `alias mi-alias="comando largo"`.

i) Los comandos `cat`, `more` y `less` permiten ver el contenido de un archivo en la consola. Por ejemplo, para revisar el contenido del archivo ya mencionado, `/etc/inittab`, lo podemos revisar con `cat /etc/inittab`, `more /etc/inittab` o `less /etc/inittab`. Compare estos 3 comandos mencionando las ventajas y desventajas que encuentra en cada uno. ¿Cuál es más completo?

- `cat` vuelca el contenido de un archivo a la terminal, sin importar lo que éste archivo contenga. Esto es útil para cuando se necesita volcar contenidos de un archivo hacia o desde la entrada o salida estándar (`stdin` y `stdout`). También se pueden volcar contenidos desde archivos binarios, imágenes de disco e incluso dispositivos de sistema.

  Su desventaja radica en que, al ser un programa de volcado, éste no dará tiempo para navegar un archivo. Además, en momentos puede corromper la salida de la terminal cuando se vuelcan cierto tipo de archivos o dispositivos, como `/dev/urandom`.
- `more` es un programa dedicado a la lectura de archivos de texto, pero de forma paginada. Es un avance sustancial desde `cat`, en términos que permite ver un archivo con más tiempo, pero no permite una navegación completa, sino linealmente y de forma paginada.

- `less` es también un programa dedicado a la lectura de archivos de texto, con navegación completa en 4 direcciones, además de propiedades de búsqueda de texto. Tiene una interfaz y comandos similares al editor `vi`.
- Si lo vemos desde el punto de vista de lectura de archivos, `less` es más completo debido a sus capacidades de navegación y búsqueda. Sin embargo no permite el volcado de un archivo hacia la entrada/salida estándar, cosa que `more` tampoco es capaz.