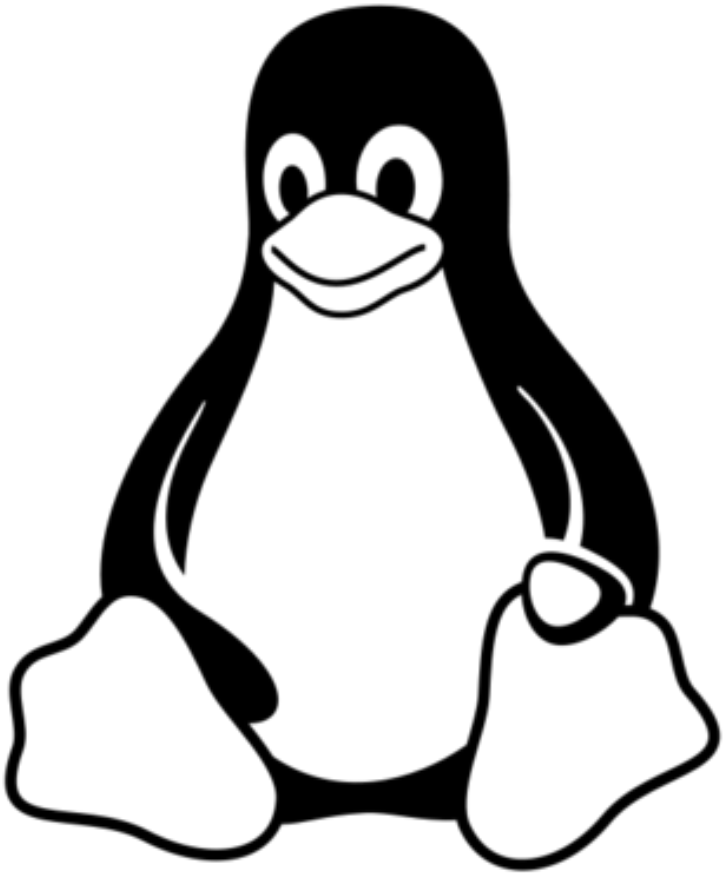
# ADMINISTRACION I

# GNU/



Linux

# CONTENIDO PROGRAMÁTICO

Unidad I. Administración de GNU/Linux

- ✓ Introducción
- ✓ Como comenzar administrar el sistema
- ✓ Administración de usuarios
- ✓ Pertenencia de los archivos
- ✓ Permisología de los archivos
- ✓ Comandos de permisología

Unidad II Introducción a la SHELL

- ✓ Variables de ambiente
- ✓ Comodines del SHELL
- ✓ Redirección del flujo de información

Unidad III. Manejo de procesos

- ✓ Definición
- ✓ Tipos de procesos
- ✓ Órdenes para el control de procesos
- ✓ Enviando señales a los procesos.
- ✓ Monitoreo de procesos

Unidad IV. Toolbox

- ✓ Definición
- ✓ Búsqueda e información
- ✓ Monitoreo local.
- ✓ Chequeo y administración
- ✓ Monitoreo de redes

Unidad V. Mecanismos de autenticación

- ✓ Cliente ssh
- ✓ Comando scp
- ✓ Herramientas de autenticación
- ✓ Utilitarios de seguridad

# UNIDAD I

## Administración de GNU/Linux

La administración de un sistema operativo como Linux es una tarea por lo regular compleja y periódica que requiere el conocimiento de ciertas herramientas y órdenes para realizar las operaciones.

**TAREAS COMUNES DE UN ADMINISTRADOR**

1. Manejo de la seguridad del sistema
2. Instalación de hardware y software
3. Manejo de usuarios del sistema
4. Configuración y entonación del Sistema Operativo
5. Manejo de las impresoras, procesos, respaldos, etc…

## ADMINISTRACION DE USUARIOS

Cada papel está en su sitio, hay carpetas y subcarpetas y todo está organizado. Ahora bien, el contable deberá tener acceso por ejemplo a las carpetas donde se encuentras las facturas y los recibos pero no tienen por qué tener acceso a la información sobre desarrollo de productos o marketing. En un sistema Linux, las carpetas y los archivos funcionan de esta manera. Por ejemplo, los archivos de configuración que se encuentran en el directorio /etc sólo pueden ser modificados por el administrador del sistema. Esto previene que cualquier usuario pueda cambiar información crítica y estropear algo.

**¿QUÉ ES EL SUPERUSUARIO?**

El superusuario, administrador del sistema o simplemente el root, es un usuario especial que tiene privilegios para cambiar la configuración, borrar y crear ficheros en cualquier directorio, crear nuevos grupos y usuarios, etc.

IMPORTANTE: ES PELIGROSO TRABAJAR COMO SUPERUSUARIO, SE PUEDE DAÑAR EL SISTEMA DE FORMA IRREVERSIBLE. EL LECTOR DEBE ESTAR SEGURO DE LO QUE HACE CUANDO TRABAJE COMO SUPERUSUARIO.

Una vez hecha esta aclaración, pasemos a hacer algo como root:

$ touch /etc/prueba.txt

touch: no se puede efectuar `touch' sobre «/etc/prueba.txt»: Permiso denegado

$ sudo touch /etc/prueba.txt

$ ls /etc/pru*

/etc/prueba.txt

El núcleo de Linux en sí trata al usuario como meros números. Cada usuario es identificado por un único numero entero, el uid (identificación de usuario) esto debido a que un número es más fácil y rápido de procesarlo que un nombre para un sistema. Una base de datos o tabla asociada a dichos UIds y GIds, por fuera del núcleo asigna un nombre textual, un único nombre de usuario para cada id. La base de datos que también contiene información adicional.

## /ETC/PASSWD Y OTROS ARCHIVOS INFORMATIVOS/DE INFORMACIÓN /ETC/SHADOW

La base de datos básica de usuarios en un sistema Unix es un archivo de texto /etc/passwd (llamado el archivo de contraseñas), que lista todos los nombres de usuarios válidos y su información asociada. El archivo tiene una línea por usuario, y es dividido en siete colon-delimited campos.

• Nombre de usuario

• Contraseña, de modo encriptado

• Identificación (Id) de numero de usuario

• Identificación (Id) de numero de grupo

• Nombre completo u otra información descriptiva de la cuenta

• Directorio Inicio (directorio principal del usuario)

• Interprete de comandos (programa a ejecutar al ingresar al sistema)

# COMANDO PARA LA CREACION DE USUARIOS

Crear usuario:

```
# useradd jose
```

Asignarle contraseña

```
# passwd jose
```

Crear directorio de usuario

```
# mkdir /home/jose
```

Ponerle dueño y grupo:

```
# chown jose:jose -R /home/jose
```

Asignarle permisos:

```
# chmod 755 -R /home/jose
```

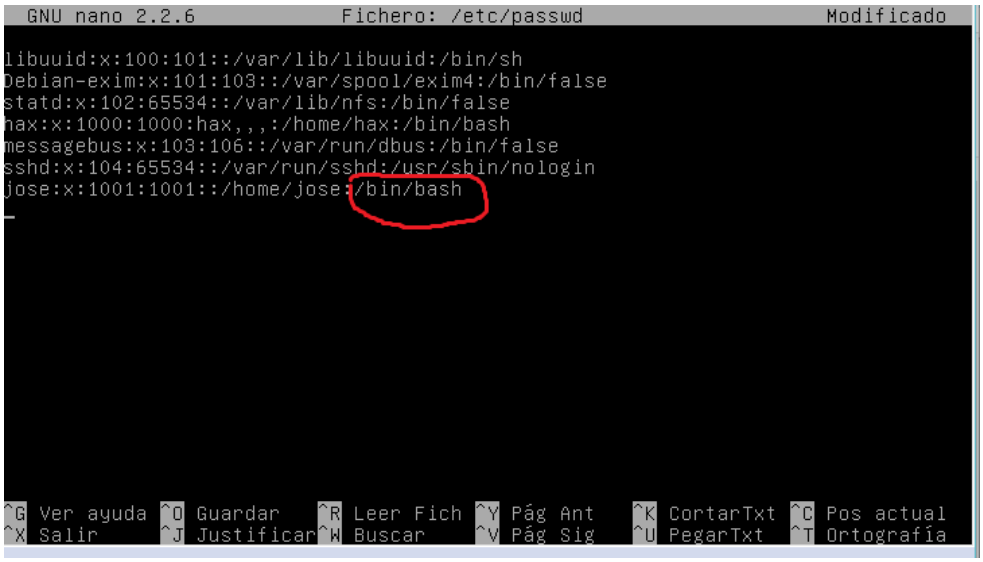
Asignarle el directorio de usuario al nuevo usuario:

```
# usermod -d /home/jose jose
```

Asegurarte de tener bash como predeterminado para tu nuevo usuario.

```
# nano /etc/passwd
```

Que la linea de tu usuario nuevo tenga en el final */bin/bash* y no */bin/sh*. Sinó cambiarla por */bin/bash*.



## CRAER GRUPOS DE USUARIOS

Los comandos groupadd, groupdel y groupmod permiten crear, borrar y modificar grupos respectivamente.

EJEMPLO:

# groupadd informática

Si queremos agregar un usuario en este caso jose a un grupo ya existente lo hacemos de la siguiente manera:

# usermod -g informática jose

La fórmula seria la siguiente: comando –opción + nombre_grupo_destino + usuario

## PERTENENCIA DE LOS ARCHIVOS

Imaginemos que el fichero informe.txt ha sido creado por el usuario pedro. Por defecto, el dueño de un archivo es el usuario que lo crea, en este caso pedro. El grupo del usuario pedro, es informática

$ su pedro

$ cd

$ pwd /home/pedro

$ touch informe.txt

$ ls -l -rw-r--r-- 1 pedro informatica0 2016-03-19 12:46 informe.txt

Todo esto se puede cambiar. Moveremos el fichero al directorio de trabajo del usuario laura y le cambiaremos el dueño.

# mv informe.txt /home/laura/

# cd /home/laura/

# chown laura informe.txt

# ls -l -rw-r--r-- 1 laura informatica0 2016-03-19 12:46 informe.txt

Ahora el fichero tiene al usuario laura como propietario. Tanto chown como chgrp se pueden usar con la opción -R para cambiar el dueño o el grupo en un directorio completo, de forma recursiva.

**PERMISOLOGIA DE LOS ARCHIVOS**

La información sobre grupos, usuarios y permisos se puede obtener mediante el comando ls junto con la opción -l. Vamos a ver los permisos que tiene establecidos el fichero whatis que se encuentra en el directorio /usr/bin.

$ ls -l /usr/bin/whatis

-rwxr-xr-x 1 root root 87792 2008-03-12 14:24 /usr/bin/whatis

En la primera columna aparecen los permisos, en la tercera se indica el usuario (en este caso es el administrador del sistema) y en la cuarta columna aparece el nombre del grupo (que en este caso coincide con el de usuario).

Vamos a ver qué significan exactamente los caracteres de la primera columna:

| - | r | w | x | r | - | x | r | - | x |
|---|---|---|---|---|---|---|---|---|---|
| Tipo de fichero. | Permisos para el dueño del fichero. | | | Permisos para el grupo al que pertenece el fichero. | | | Permisos para el resto de usuarios | | |

| | |
|---|---|
| r | Permiso de **lectura**. |
| w | Permiso de **escritura**. |
| x | Permiso de **ejecución**. |

A este método, que utiliza los caracteres rwx se le denomina método simbólico. Podemos utilizar de forma análoga el método numérico.

| 4 2 1 | Total |
|---|---|
| r w x | 4 + 2 + 1 = **7** |
| r w - | 4 + 2 + 0 = **6** |
| r - x | 4 + 0 + 1 = **5** |
| r - - | 4 + 0 + 0 = **4** |
| - w x | 0 + 2 + 1 = **3** |
| - w - | 0 + 2 + 0 = **2** |
| - - x | 0 + 0 + 1 = **1** |

## COMANDOS DE PERMISOLOGIA

Cambio de permisología en modo Octal

# chmod 755 hola_mundo.rb que sería el equivalente a rwxr-xr-x

Sería equivalente a estas tres

$ chmod u+rwx hola_mundo.rb

$ chmod g+rx-w hola_mundo.rb

$ chmod o+rx-w hola_mundo.rb

**UNIDAD II**

## INTRODUCCION A LA SHELL

SHELL es un programa que interactúa vía teclado con el sistema operativo a través de órdenes, es un intérprete de órdenes.

El Shell más utilizado en Linux es el BASH (Bourn Again Shell), aunque también existen otros tipos de Shell tales como el sh, ksh, csh, entre otros.

### CARACTERISTICAS DE BASH

- Soporta la introducción de ordenes por parte del usuario
- Provee un poderoso lenguaje de guiones
- Mantiene un registro histórico de ordenes (HISTORY)
- Permite la completación automática de los nombres de archivos
- Soporta el redireccionamiento y las tuberías

## VARIABLES DE ENTORNO

Las variables de entorno y configuraciones son aquellas que tienen un significado propio para la SHELL o algún otro programa. Ciertos programas leen el contenido de las variables de entorno para modificar su comportamiento, entre ellos la propia SHELL. Entre las variables de entorno más importantes se pueden citar:

• PATH, indica la ruta de búsqueda de programas ejecutables. Está constituida por una lista de directorios separados por dos puntos (:). El directorio actual, de forma predeterminada, no viene incluida en PATH.

• PS1, especifica el indicador del sistema. Lo habitual es que PS1 sea el símbolo $ para usuarios normales y # para usuario root.

LANG, especifica el lenguaje que se aplica al usuario; para español se utiliza es.

• TERM, almacena el tipo de terminal desde el que se está trabajando.

• EDITOR, especifica el editor por omisión del sistema. Lo habitual en los sistema Unix es que el editor por omisión sea vi.

• DISPLAY, especifica qué equipo muestra la salida que se efectúa en modo gráfico.

Ese equipo deberá tener un servidor gráfico.

• PWD, contiene el directorio de trabajo efectivo.

Para ver el contenido de una variable, se puede usar el comando echo de la Siguiente manera:

*echo $VARIABLE*

Para eliminar una variable, se utiliza el comando interno del intérprete bash, llamado unset pasándole como parámetro el nombre de la variable

- Los nombres de las variables pueden consistir de letras, números o símbolos, pero no deben comenzar con un número
- Use el símbolo "=" para asignar un valor a una variable
- Use el símbolo "$" para accesar a contenido de una variable.
- Si el valor tiene espacios en blancos debe entrecomillarse

Ejemplo;

$VARIABLE_1=curso

$echo $VARIABLE_1

$curso

$VARIABLE_2="curso de Linux"

## USO DEL ENTRECOMILLADO EN EL SHELL

EL SHELL interpreta los textos entrecomillados de la siguiente manera:

- Las comillas dobles “” expanden los caracteres $,\,* que estén en el texto
- Las comillas simples ‘’ no expanden ningún carácter en el texto
- Las comillas tildes `` ejecutan como una orden lo que está en el texto

Ejemplo:

$CURSO=Linux

$echo “El curso es $CURSO”

$ EL curso es Linux

$echo ‘El curso es $CURSO’

$el curso es $CURSO

$ echo “El directorio es `pwd`”

$el directorio es /home/curso

## COMODINES DEL SHELL

Los comodines sustituyen la lista de archivos que se pasan como argumentos a las ordene.

- * sustituye cero o más caracteres a partir de la posición que se encuentra
- ? sustituye un solo carácter a partir de la posición que se encuentra
- [ rango ] sustituye un solo carácter basado en el rango a partir de la posición que se encuentra

  Ejemplo:

  *$ ls a*  #listar todo lo que comience por a*

  *$ ls a?c  #listar cualquiera en el segundo*

  *$ ls a [c-e] #listar los nombres de archivos del segundo carácter entre c y e*

## HISTORIA DE ÓRDENES EN EL SHELL

El bash salva las órdenes entradas por el usuario en un archivo (bitacora) que es almacenado de acuerdo a la variable del ambiente HISTFILE.

- Use la orden **history** para ver la bitácora de la ordenes
- Use el carácter "!" con un numero o string para ejecutar una orden de la bitácora
- La combinación de las teclas control (ctrl) y la "r" realiza una búsqueda en orden reverso por la bitácora.

## AYUDAS EN EL SHELL

El man (manual electrónico) , permite obtener una ayuda detallada y precisa. Está compuesto de varias secciones de clasificación.

Sintaxis: man orden

El info permite obtener una ayuda detallada y precisa, además de permitir navegar por tópicos seleccionados.

Sintaxis: info orden

La opción help o –h que viene construida internamente en las ordenes y presenta por lo general una ayuda resumida…

Sintaxis: orden--help

## REDIRECCION DEL FLUJO DE INFORMACION

Una vez que se ejecuta un comando, se crea un proceso. Este proceso abre tres flujos:

1) stdin, denominado entrada estándar, en cuyo caso el proceso lee los datos de entrada. De manera predeterminada, stdin se refiere al teclado. stdin se identifica con el número 0.

2) stdout, denominado salida estándar, en cuyo caso el proceso escribe los datos de salida. De manera predeterminada, stdout se refiere a la pantalla. stdout se identifica con el número 1.

3) stderr, denominado error estándar, en cuyo caso el proceso escribe los mensajes del error. De manera predeterminada, stderr se refiere a la pantalla. stderr se identifica con el número 2.

Por lo tanto, de manera predeterminada, cada vez que se ejecuta un programa, los datos se leen desde el teclado y el programa envía su salida y sus errores a la pantalla.

Sin embargo, también es posible leer datos desde cualquier dispositivo de entrada, incluso desde un archivo, y enviar la salida a un dispositivo de visualización, un archivo, entre otros.

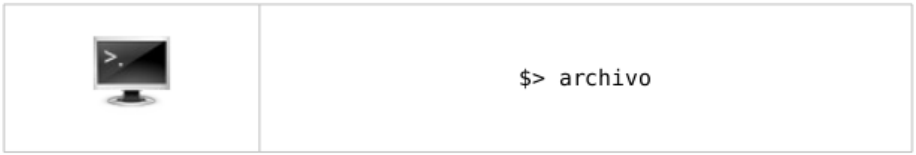
**Redirecciones**

Como cualquier sistema Unix, Linux posee mecanismos que permiten redirigir la entrada-salida estándar a archivos.

Por lo tanto, si se usa el carácter ">", se puede redirigir la salida estándar de un comando que se encuentra a la izquierda a un archivo que se encuentra a la derecha:

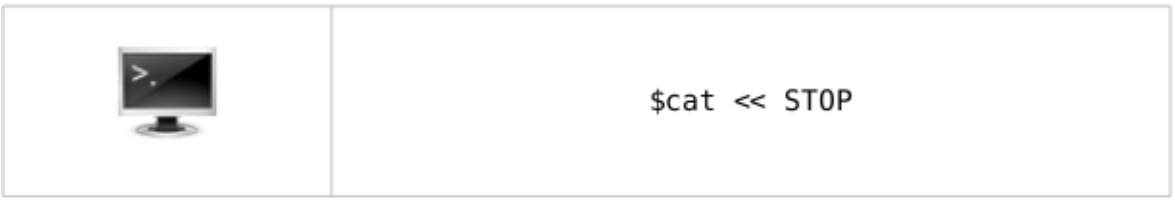

El siguiente comando equivale a una copia de los archivos:


El propósito de la redirección ">" es el de crear un archivo nuevo. En el caso de que un archivo ya exista con el mismo nombre, se lo debe eliminar. El siguiente comando simplemente crea un archivo vacío:




El uso del carácter doble "**>>**" permite agregar la salida estándar al archivo, es decir, permite agregar la salida después del archivo sin eliminarlo. De manera similar, el carácter "**<**" indica una redirección de la entrada estándar. El siguiente comando envía el contenido del archivo *toto.txt* con el comando *cat*, cuyo único propósito es mostrar el contenido en la salida estándar (el ejemplo no es útil, pero es instructivo):

Por último, el uso de la redirección "<<" permite la lectura, en la entrada estándar, hasta que se encuentre la cadena ubicada a la derecha. En el siguiente ejemplo, se lee la entrada estándar hasta que se encuentra la palabra STOP. Después, se muestra el resultado:




**Salida de errores**.

Si quisiéramos realizar un listado de un directorio y, en caso de producirse un error, este fuese redirigido a un archivo, haremos lo siguiente:


Esta simple redirección solo tendrá efecto sobre el error estándar (stderr) o como también se denomina, *descriptor de archivo nº 2*. Con esta redirección los posibles errores serían redirigidos al archivo /tmp/error.ls. Si quisiéramos dividir tanto la salida por pantalla como el error en dos archivos separados podemos hacerlo de esta manera:


**UNIDAD III**

## MANEJO DE PROCESOS

Los procesos en Linux son programas ejecutándose en la memoria RAM, a cada una de ellas se les asigna un numero identificador, conocido como Process ID (PID) y un propietario (owner). El proceso init es el primer proceso del sistema y el padre de los procesos.

**TIPOS DE PROCESOS**

- Sin control de terminal o Background (Demonios, Script arranque del sistema, etc).
- Con control del terminal o Foregraouund (iniciados desde una consola por un usuario).

Para empezar, se pueden ver las tareas y las subtareas en una estructura anidada mediante el comando **pstree** es decir, permite visualizar un árbol de procesos. Asimismo, **pstree -p** muestra entre paréntesis el número identificador (PID) de los procesos, algo muy importante cuando se quiere pasar de actuar de forma pasiva a interactuar con los procesos, cosa que normalmente se hace señalando sus PIDs. Aunque dicha información estructurada resulta interesante, existen dos comandos muy conocidos que muestran una cantidad ingente de información sobre los procesos.

## GESTIÓN DE PROCESOS

**COMANDO PS**

Muestra una lista de los procesos en ejecución. Las opciones más habituales son: ps u → que muestra los procesos que pertenecen al usuario actual, ps aux → muestra información detallada de todos los procesos en ejecución. Algunos de los campos más importantes mostrados por ps son:

• USER - usuario dueño del proceso.

• PID - número identificador del proceso.

• %CPU - porcentaje de uso del microprocesador por parte de este proceso.

• %MEM - porcentaje de la memoria principal usada por el proceso.

• VSZ - tamaño virtual del proceso (lo que ocuparía en la memoria principal si todo él estuviera cargado, pero en la práctica en la memoria principal solo se mantiene la parte que necesita procesarse en el momento).

• RSS - tamaño del proceso en la memoria principal del sistema (generalmente son KBytes, cuando no lo sea, se indicará con una M detrás del tamaño).

TTY - número de terminal (consola) desde el que el proceso fue lanzado. Si no aparece, probablemente se ejecutó durante el arranque del sistema.

• STAT - estado del proceso.

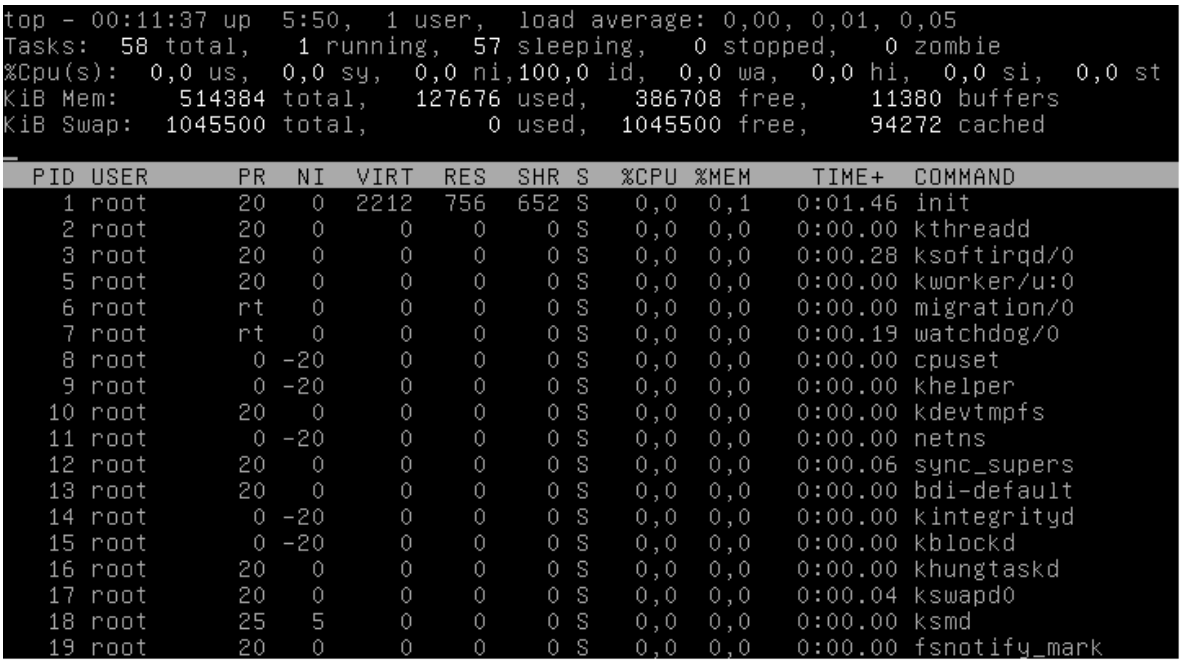
• START - cuándo fue iniciado el proceso.

• TIME - el tiempo de CPU (procesador) que ha usado el proceso.

• COMMAND - el comando que inició el proceso.

**COMANDO TOP**

Es la versión interactiva de ps, y tiene algunas utilidades interesantes añadidas. Si se ejecuta en una terminal y sin opciones, aparecerá arriba información del sistema: usuarios, hora, información del tiempo de funcionamiento de la máquina, número de procesos, uso de CPU, uso de memoria y uso del swap y a continuación muestra una lista de procesos similar a la que se muestra con ps, la diferencia entre ambos radica en que ésta se actualiza periódicamente, permitiendo ver la evolución del estado de los procesos.




**ORDENES PARA EL CONTROL DE PROCESOS**

**Comando bg.**

Nos permite reanudar un proceso y que este se ejecute en segundo plano, permitiéndonos hacer uso del prompt mientras dicho proceso avanza. Para iniciarlos utilice el carácter & al final de la orden

Ejemplo: $./proceso&

[1 ] 5561

**Comando fg**

Este comando nos sirve para reanudar un proceso que se haya detenido y lo envía al primer plano nuevamente para que continúe su ejecución. Con las teclas ctrl+z los detiene (stop) y ctrl+c los interrumpe

**Jobs:** se utiliza para ver la lista de los procesos en background o detenidos

**Enviando señales a los procesos**

**COMANDO KILL**

Este comando sirve para matar o anular procesos indeseados. Se debe tener en cuenta que cada proceso lleva su usuario y por tanto solo él (o el superusuario) pueden anularlo. Normalmente, si los programas que componen el grupo de procesos son civilizados, al morir el padre mueren todos ellos siempre y cuando el padre haya sido señalizado adecuadamente. Para ello, se emplea el comando $kill - <número_señal> PID, siendo PID el número del proceso o del grupo de procesos. Los números de señales (número_señal) utilizados con más frecuencia son:

| Número | Descripción |
|---|---|
| 15 | TERM o terminación, se manda para que el proceso cancele ordenadamente todos sus recursos y termine. |
| 1 | HUP[6], normalmente utilizado en procesos de servicios para que hagan una nueva lectura de sus archivos de configuración. |
| 2 | Interrupción |
| 3 | Salir (QUIT) |
| 5 | TRAP. Señal para hacer que el programa depurador que supervisa el proceso informe sobre los puntos de chequeo del mismo. |
| 9 | (KILL) La más enérgica de todas las señales, esta evita que los procesos mueran ordenadamente. El proceso que la recibe finaliza inmediatamente. |

La señal puede ser por ejemplo:

$ kill -9 3054 (elimina el proceso 3054 inmediatamente)

$ kill –SIGTERM 1901 (termina la ejecución del proceso)

$ kill –SIGSTOP %1 (detiene el primer trabajo en la cola de los backgrounds)

La variable killall permite enviar un SIGTERM a un conjunto de variables.

**UNIDAD IV**

## TOOLBOX

TOOLBOX: es el conjunto de órdenes de uso periódico que permiten las funciones de monitoreo, administración y corrección de posibles errores en un entorno local o de red. Su objetivo es proveer al administrador de un set de órdenes básicas que faciliten sus tareas rutinarias y que sean susceptibles de ser usadas en cualquier entorno y/o distribución.

**Búsqueda e información**

| ORDENES | FUNCION BASICA | SINTAXIS Y EJEMPLO |
|---|---|---|
| FIND | buscar archivos en un árbol de directorios en base a la evaluación de expresión y operadores | Find [path inicial opciones parámetros<br>Find / -name core –and –size +10] |
| GREP | Busca las líneas en un archivo que concuerden con el patrón dado | grep patrón [archivos<br>ls \| grep prueba |
| df | Informa sobre la utilización de espacio en disco de un sistema de archivos | df [opciones] [archivos]<br>df –k |

**Monitoreo Local**

| Ordenes | Función básica | Sintaxis y ejemplo |
|---|---|---|
| last | Muestra la lista de los últimos usuarios conectados por el /var/log/utmp | last [opciones] |
| lastlog | Muestra el log de los últimos accesos por usuario por el /var/log/lastlog | lastlog [opciones] |
| pstree | Despliega el arbol de procesos actuales en el sistema | pstree [opciones<br>pstree -p] |
| ps | Despliega el status de los procesos actuales en el sistema | ps [opciones<br>ps aux] |

**Monitorio local y otros**

| **Ordenes** | **Función básica** | **Sintaxis y ejemplo** |
|---|---|---|
| uptime | muestra el tiempode encendido del equipo, usuarios y carga actual del sistema | uptime [-V]<br>uptime |
| top | Muestra los procesos que más consume CPU en forma ciclica | top [opciones] |
| tail | Despliega las últimas líneas de un archivo | Tail [opciones] [archivo<br>Tail–f /var/log/messages] |
| watch | Muestra la ejecución de un programa con intervalos de segundos | Watch [-n segundos] programa<br>Watch pstree |

**Chequeo y administración**

| **Ordenes** | **Función básica** | **Sintaxis y ejemplo** |
|---|---|---|
| Md5sum | Genera una suma de control usando el algoritmo MD5 | Md5sum –c archivo |
| diff | Muestra las diferencias entre dos archivos línea por línea | diff [opciones] archivo1 archivo2 |
| Dpkg | Administra paquetes debían de nivel medio | Dpkg [opciones] acción<br>Dpkg –i paquetes |
| rpm | Administra paquetes Red Hat Package | Rpm –i archivo-paquetes |

| Ordenes | Función básica | Sintaxis y ejemplo |
|---|---|---|
| netstat | Muestra conexiones de red, tablas de encaminamiento, estadísticas y mensajes. | netstat -ie |
| Traceroute | Realiza un trazado de las rutas que va cursando un paquete hasta un host. | Traceroute hostname |
| Ping | Envía paquetes tipo ICMP a un host de la red para probar tiempos y conectividad | Ping localhost |
| Bing | Envía paquetes tipo ICMP entre dos host remotos para probar el ancho de banda | Bing host1 host2 |

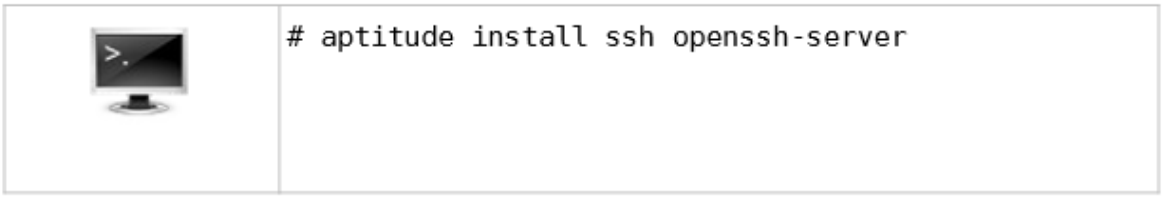
**UNIDAD V**

## MECANISMOS DE AUTENTICACION

SSH ( Secure SHell ) es una implementación segura de sesiones de intérprete de comando remotas. SSH es la manera segura de comunicarse a través de Internet para la administración de un computador ejecutando GNU/Linux. La implementación más popular de SSH consta de una variante llamada OpenSSH. La cual incluye una colección de herramientas para realizar conexiones a través de este protocolo.

### Iniciando ssh

Para comenzar, instale el cliente y el servidor OpenSSH. (en Canaima GNU/Linux tanto el cliente ssh como el servidor vienen instalados por defecto)




### Servidor SSH

El servidor ssh, al funcionar en un computador con GNU/Linux, permite la conexión de clientes a un intérprete de comandos en el sistema anfitrión. Por otro lado, sus características de encriptación y autenticación de varias vías le dan mucha flexibilidad cuando se trata de controlar el acceso de los clientes con métodos de autenticación que van desde la simple solicitud de contraseñas, hasta el bloqueo de clientes desconocidos a la red donde opera.

**Opciones de configuración**

Las configuraciones del servidor ssh, como sus mecanismos de autenticación y el puerto donde este opera pueden cambiarse editando el archivo /etc/ssh/sshd_config. Explicamos a continuación los más importantes:

• Port: Define el puerto donde el servicio funcionará. El puerto estándar de ssh es el puerto 22 TCP.
• Protocol: Define la versión de protocolo que se acepta por defecto, el valor por defecto es 2, ya que actualmente, es poco común conseguir un cliente ssh que no soporte esta versión del protocolo.
• UsePAM: Este parámetro determina si el servidor ssh permitirá la entrada a usuarios válidos del sistema donde se está ejecutando, de estar establecido a "no" la autenticación del servidor deberá configurarse para utilizar otra base de datos de usuarios y contraseñas, o bien, un método de autenticación por validación de llave pública.

• PubkeyAuthentication: autenticación del usuario basada en una clave pública

• Hostbased Authentication: autenticación basada en.rhosts o /etc/hosts.equiv combinada con la autenticación de la clave pública de la máquina cliente (desactivada).

• Password Authentication: autenticación basada en contraseña.

• Challenge Response Authentication: autenticación basada en challenge response.

• /etc/ssh/ssh_config es el archivo de configuración más importante, las entradas son:

• Host: Restringe las siguientes declaraciones (up to the next Host keyword) siendo nada mas los host que declaremos aquí los que se les dará una llave de auntentificacion ssh para entablar comunicación.

• Protocol: especifica la versión del protocolo SSH. Valor predeterminado "2,1".

• Preferred Authentications: especifica el método de autenticación para el cliente SSH2. Por defecto: "hostbased,publickey,keyboard-interactive,password".

• ForwardX11: desactivado por defecto. Se puede no tener en cuenta mediante la opción "-X" de la línea de comandos

• Otro archivo de interés: /etc/ssh/sshd_config: valores predeterminados del servidor SSH. Las entradas más importantes son:

• ListenAddress: especifica las direcciones locales que sshd debe escuchar. Se permiten múltiples opciones.

• AllowTcpForwarding: desactivado por defecto. • X11Forwarding: desactivado por defecto.

**Cliente SSH**

Mediante el uso del cliente ssh es que en práctica, se hace uso de las bondades de esta herramienta administrativa. El cliente ssh se invoca en cualquier consola del sistema por su nombre

Lo siguiente iniciará una conexión ssh desde un cliente a un servidor en la máquina miservidor.ssh.com

```
usuario@maquina$ ssh nombre_usuario@miservidor.ssh.com
password: (digite su contraseña)
nombre_usuario@miservidor.ssh.com:$
```

Como puede verse, se obtiene una sesión remota en un intérprete de comandos en otro computador.

El cliente y servidor ssh resulta de especial importancia para la administración de sistemas GNU/Linux remotos, aunque se encuentren próximos al administrador. Ya que permiten una gran flexibilidad y comodidad para la administración y operación de los ambientes productivos.

**El comando SCP**

Parte del cliente ssh es el comando scp ( secure copy) que permite la copia segura de archivos entre computadores.

Por ejemplo, deseamos copiar el archivo “pepe.txt” desde el directorio /tmp de una máquina al home del usuario “sutano” en la máquina con dirección “mi.ssh.net”.

```
$ scp /tmp/pepe sutano@mi.ssh.net:/home/sutano/
sutano@mi.ssh.net's password:
pepe.txt 100%|*****************************| 50165
```

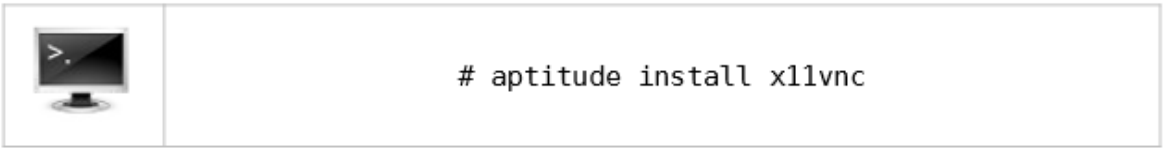
De nuevo, se le pedirá una clave. La orden scp muestra el progreso de la copia por omisión. Puede copiar un archivo desde un host remoto con la misma facilidad; simplemente especificando su nombre de host y ruta como origen y la ruta local como destino. También se puede copiar un archivo desde un host remoto a otro host remoto, pero habitualmente no necesitará hacer eso, porque todos los datos viajan a través de su host.

## HERRAMIENTAS DE AUTENTICACION

### Servicio VNC

Hay muchos servidores VNC, pero explicaremos el uso de x11vnc porque es el más sencillo de utilizar. La mayoría de servidores VNC requieren un display de las X particular y, aunque ofrecen un escritorio remoto, lo que hacen es iniciar una nueva sesión gráfica en vez de ofrecer acceso a una sesión ya existente. Con x11vnc podremos permitir el acceso a una sesión X ya existente de una forma.

Instalamos el paquete para el Servidor:




### Arrancando el servidor

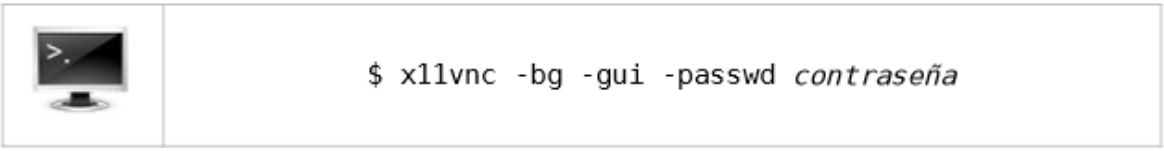
Para arrancar el servidor, abriremos una consola y escribiremos el comando x11vnc. Esto nos iniciará un servidor básico, sin contraseña, que permite el acceso a todo el mundo y que una vez ha desconectado el cliente, se cierra.

A continuación, veremos los parámetros que podemos pasarle al inicio, para configurar el servidor de una forma más razonable:

• bg: Nos inicia el servidor en segundo plano. Para poder cerrar la consola y que siga en marcha. • passwd: Establece la contraseña que se pedirá a los clientes al conectar

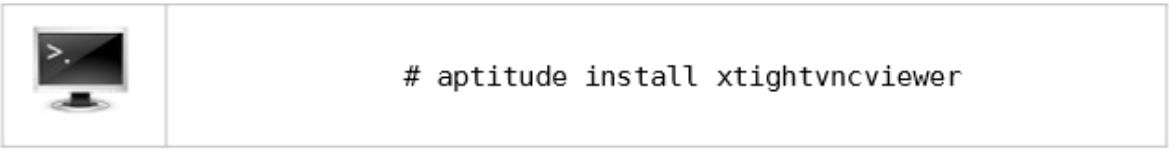
• gui: Inicia una pequeña interfaz gráfica del lado del servidor.

Sabiendo estos parámetros, podríamos iniciar el servidor de VNC de esta manera, para que nos aparezca su ventana de configuración:




Esto iniciará el servidor VNC y nos abrirá la pantalla de configuración, en la que podremos configurar opciones avanzadas del servidor.

Por el lado del cliente utilizaremos tighvncviewer, una implementación de cliente VNC muy robusta y multiplataforma. Para instalar el paquete cliente, se utiliza la siguiente orden:




Se puede ejecutar xtightvncviewer desde el enlace del menú que nos crea en nuestro computador cliente, sin embargo, también puede ejecutarse a conveniencia desde la consola:


Cuando esté ejecutándose, nos solicitará el nombre o dirección IP del host con el que deseamos conectarnos. Una vez establecida la conexión, nos preguntará por la contraseña previamente definida cuando configuramos y ejecutamos el servidor. Esto es igualmente válido para servidores VNC a los cuales tengamos acceso pero no necesariamente hayan sido instalados por nosotros.

**HARRAMIENTA PUTTY**

PuTTy es un cliente de red que soporta los protocolos SSH, Telnet y Rlogin y sirve principalmente para iniciar una sesión remota con otra máquina o servidor. Es de licencia libre y está diseñado y mantenido principalmente por Simon Tatham desde Gran Bretaña. A pesar de su sencillez es muy funcional y configurable.

**Diferencia entre ssl Shell y telnet**

Telnet sólo debe ser usado por usuarios avanzados que necesitan ejecutar shell scripts o utilizar comandos shell. Desgraciadamente, por este método tradicional, tanto el login como el password (así como el resto de la sesión) se transmiten en texto claro a través de nuestra red local o incluso a

través de routers y nodos ajenos al nuestro. Esto quiere decir que cualquiera que tenga activado un sniffer puede capturar nuestras sesiones con el potencial peligro que ello conlleva. Si deseas acceder remotamente a un servidor de forma segura, utiliza el protocolo SSH (Secure Shell). Es similar a una sesión telnet tradicional, pero con la particularidad de que todas las comunicaciones son cifradas para evitar que terceras personas puedan descubrir información sensible como el usuario, la contraseña o lo que se escribe durante una sesión.

## UTILITARIOS DE SEGURIDAD

- Crack : permite chequear claves débiles en el sistema
- Chrootkit : chequea posibles contaminaciones de virus y troyanos
- Logcheck : creado para verificar problemas de violación de seguridad por medio del monitoreo de los archivos de log
- Port sentry : diseñado para detectar y responder a buscadores de puertos en tiempo real
- Trippwaire : es un IDS (instrusion detection system)