Unidad 1

Comprensión de los Shells Linux

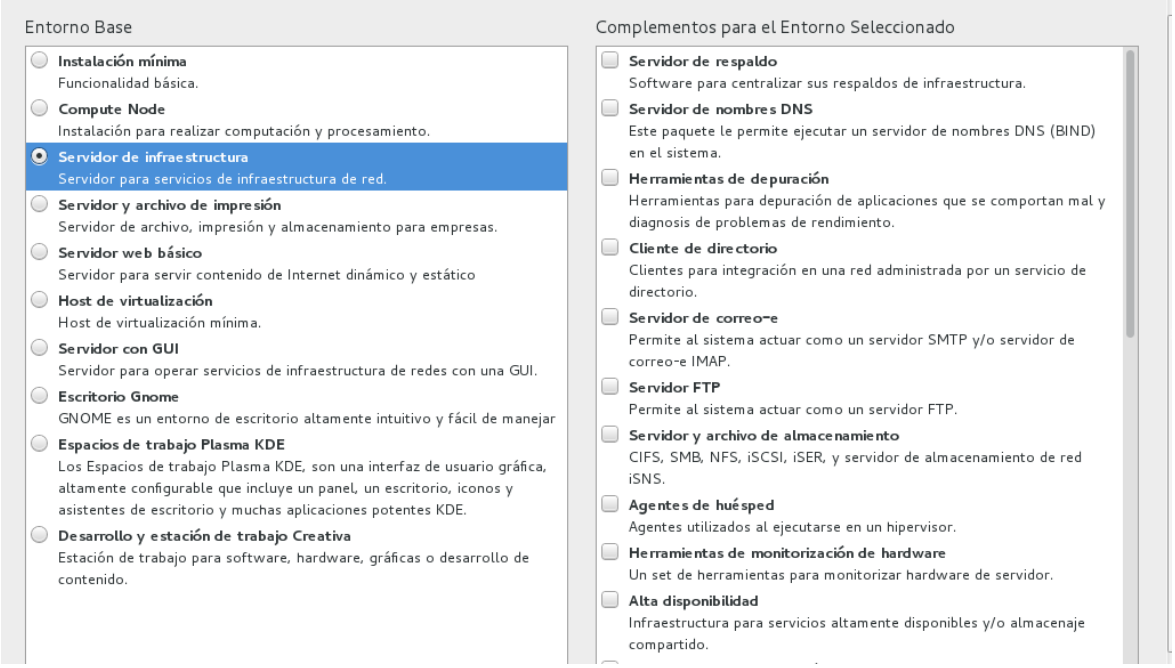
Entorno de trabajo

Para este módulo vamos a comenzar a utilizar más intensivamente la consola de Linux, la idea es que en algún momento puedan trabajar/administrar servidores Linux, lo cual significa usar la consola 100%. Linux se trata de performance, seguridad y eficiencia entre otras cosas, y eso significa tener instalado únicamente el software que necesito para la tarea que estoy llevando a cabo. En los entornos profesionales es muy raro ver un servidor Linux con entorno gráfico, eso está reservado a equipos de escritorio comúnmente.

Por lo cual vamos a instalar una máquina virtual con Centos 7 pero sin entorno gráfico lo cual nos obligará a aclimatarnos a la consola al punto de que nos sentiremos más cómodos escribiendo comandos que usando el mouse, o por lo menos esa es la idea.

El procedimiento de instalación ya lo saben así que solamente cambiaremos la parte en la que seleccionan el software a ser instalado:

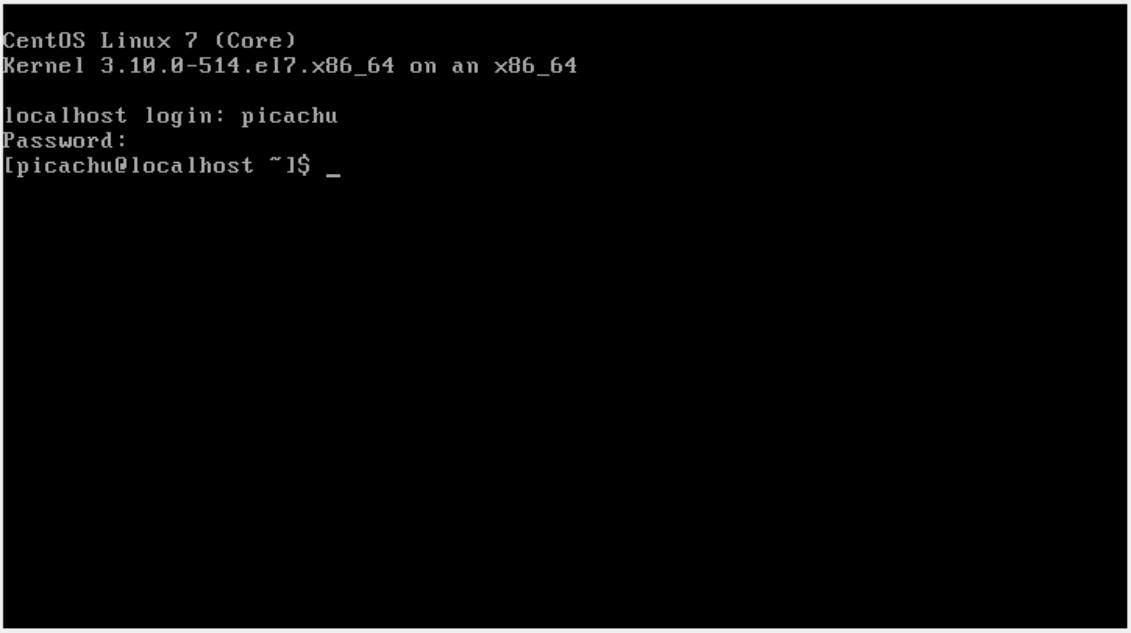
Ingresando a la sección de "Selección de Software" seleccionaremos la opción a mano izquierda "Servidor de infraestructura". No hace falta seleccionar nada del lado derecho donde se encuentran los complementos. Si hace falta algo lo instalaremos manualmente (es más divertido!!)



Después continuaremos con la instalación como lo hemos hecho hasta ahora. Recuerden crear un usuario común con su password para poder trabajar y poner la password de root para poder hacer operaciones que requieran permisos de super usuario.

Por favor no se olviden de configurar la placa de red en el virtualizador en modo puente (tal como vimos anteriormente) para que puedan tener acceso a internet y podamos instalar las herramientas necesarias como así también actualizar nuestro sistema operativo y estar resguardados contra amenazas de seguridad.

Una vez finalizada la instalación al reiniciar el sistema se encontrarán con una pantalla de login negra, sin gráficos, sin botones…. Solo letras blancas con fondo negro. Pero no se preocupen, así es como debe ser.



Aquí pondremos nuestro usuario, le daremos "Enter" y luego la contraseña que establecieron en la instalación.

Ahora como pueden ver ya pude ingresar…. "pero que son todos esos símbolos y cosas que aparecen ¡!???"… Todo tiene su razón de ser, especialmente en Linux ¡!

Lo que están viendo es el Shell o consola de comandos de Linux, que concretamente se llama BASH (Bourn Again Shell). Ya habíamos hablado de esto anteriormente, pero es bueno refrescar la memoria.

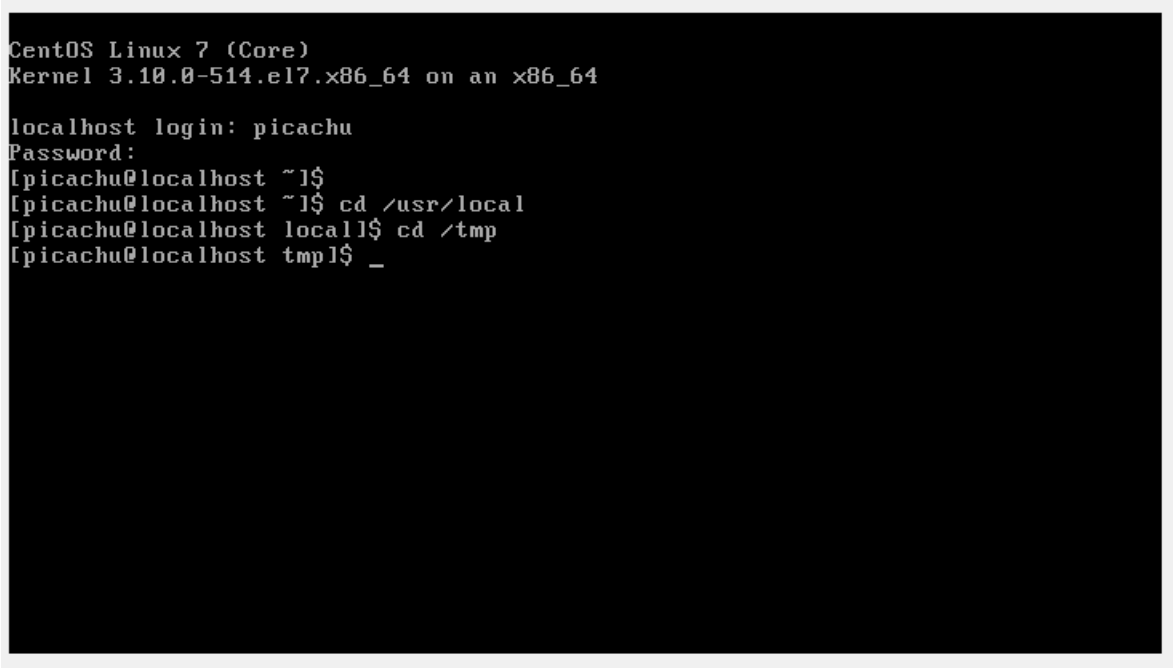
Cuando estábamos en el entorno gráfico accedíamos a ella mediante la Terminal, que es un programa o emulador que hace de interface, pero a los usos prácticos es lo mismo.

Volviendo a los símbolos, es muy simple lo que se encuentra entre corchetes [ ] se lee "el usuario" en "maquina o server" donde me encuentro. En este caso mi usuario es *picachu* y me encuentro en la maquina *localhost* que significa anfitrión (más adelante le cambiaremos el nombre a la máquina).

Y porqué necesito saber en qué maquina/server me encuentro? Cuando estemos administrando o trabajando en más de una maquina en simultáneo verán que es muy fácil perderse por lo cual podríamos

ejecutar el comando equivocado en la maquina equivocada. Presten mucha atención, es un error mucho más común de lo que se imaginan.

Volviendo a nuestro Shell verán que aparece el símbolo "~" eso es un atajo, acceso directo o sinónimo de nuestro home directory. Si cambiamos de directorio o path verán que cambiará:



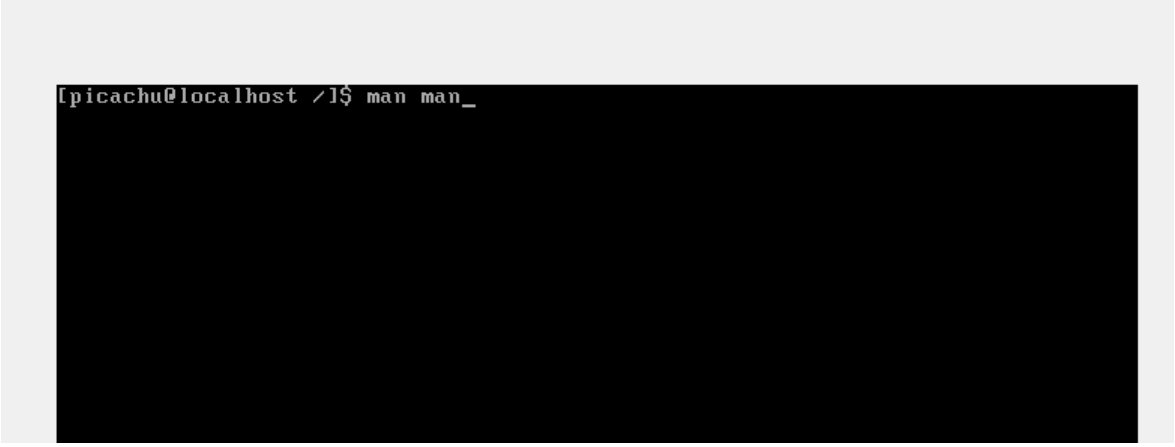
Y por último verán el símbolo de "$" que se llama prompt y lo que nos indica es que el sistema está esperando que le enviemos un comando, además nos indica que estamos usando un usuario con permisos comunes, cuando estemos logueados como super usuarios verán que estarán con el símbolo "#".

Existen tantos tipos de shell como distribuciones o sistemas operativos UNIX, para nombrar algunos: Korn Shell, C Shell, Bourn Shell, Z Shell, Alquimist Shell. Y la mayoría pueden ser instalados en Linux, pero el estándar hoy en día es BASH para casi el 100% de las distribuciones, razón por la cual nos vamos a dedicar a BASH en concreto.

Linux Man

Linux posee un manual en línea, es una de las herramientas más poderosa que vamos a usar. Dado que existen miles de comandos en Linux es imposible saber de memoria todas las opciones disponibles. Allí radica el poder de tener un manual que puedo consultar en cualquier momento aunque no tenga conexión a internet.

Para invocar el manual en Linux utilizamos el comando "man" y luego el comando que queremos investigar, así que vamos a comenzar investigando el mismo comando "man"!!!



Como podrán ver aparece una página del manual del comando "man" y tiene una estructura determinada… Esa estructura es un default y los programadores que desarrollan las herramientas deben cumplir con la misma, aunque no siempre es el caso.

Comienza por el **nombre** del comando y una pequeña descripción, luego una **sinapsis** de todos los parámetros y opciones del comando, esa parte es media difícil de entender al principio pero no se preocupen ya lo entenderán.

Luego tiene una **Descripción** donde se amplía la descripción y a continuación una sección de **Opciones** donde tienen información de cada una de las opciones que se ven en la sinapsis.

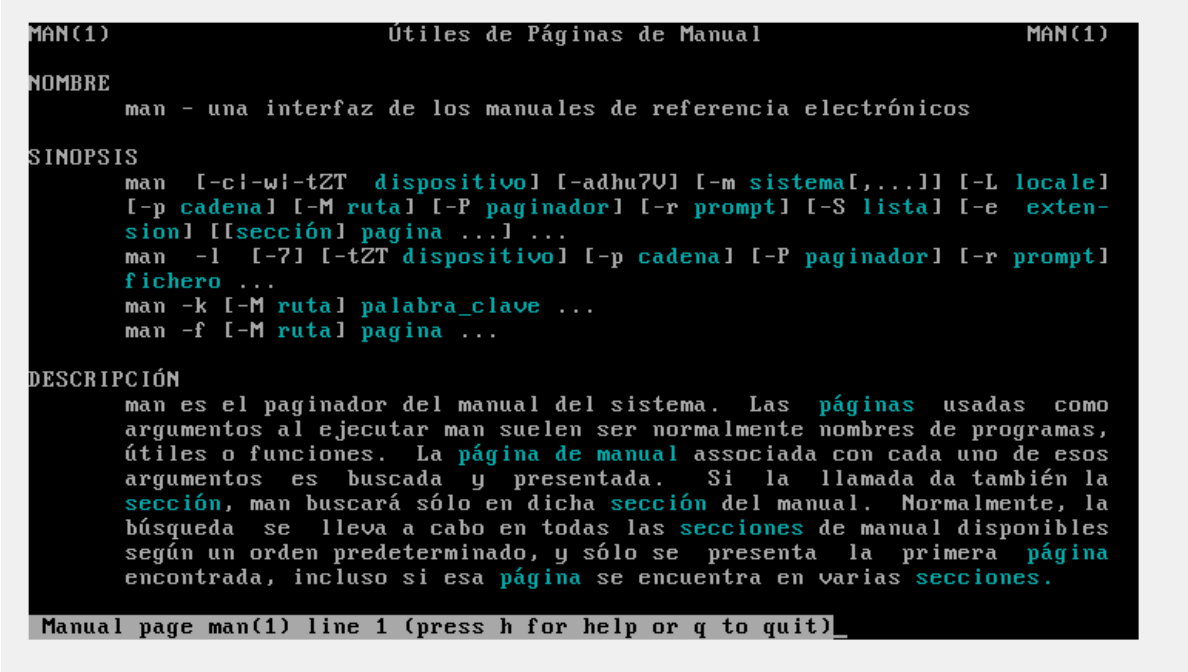
Estas opciones llamadas flags se implementan de dos formas, una forma corta que se denota con el signo menos y una letra, por

ejemplo -k o -f o -N, etc… Recuerden que Linux al igual que todos los UNIX son case sensitive, no es lo mismo N que n.

La otra forma de invocar una opción de un comando es con dos signos menos y una palabra, pero depende del comando si esta opción está disponible o no. Un ejemplo de eso es el flag -e para el comando man. Puedo ejecutar "man -e <sub-extensión>" o puedo ejecutar "man -extension=<sub-extension>" el resultado del comando es exactamente el mismo. Por cierto, lo que se encuentra entre <…> en este caso la palabra sub_extension significa que el usuario debe completar esa parte con la información que se le enviará al comando.

Generalmente luego de eso viene una sección de ejemplos y por último las referencias a otros manuales o comandos que pueden servirnos.

Con la flecha hacia abajo ir bajando y viendo la información:



Para salir del manual solo hace falta apretar la letra "q" y si quieren más información pulsen la letra "h".

Un flag muy interesante para usar con el comando man es -k que lo que nos permite es buscar cierta palabra dentro de todos los manuales, lo mismo sucede con el comando "apropos":

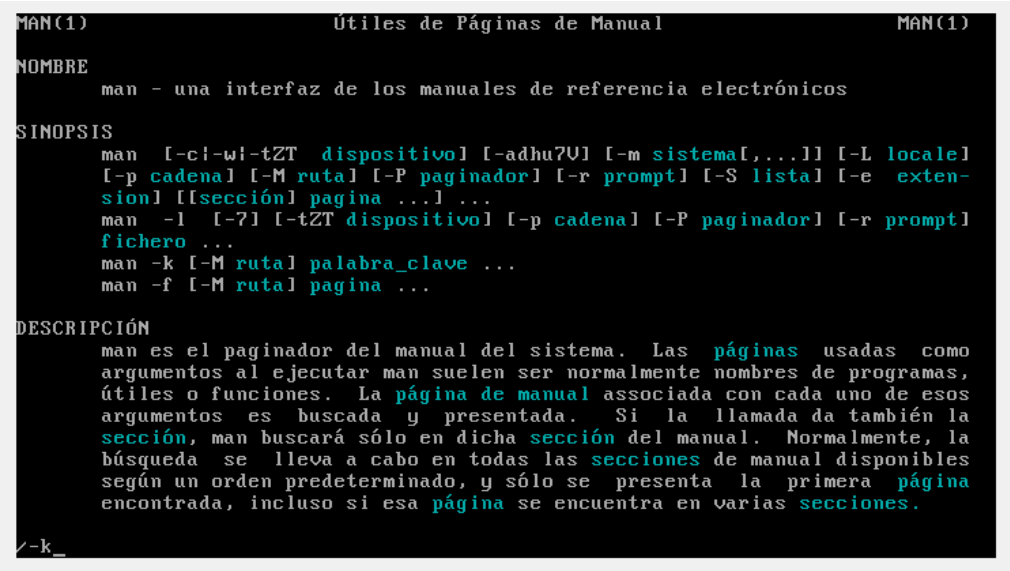
```
[picachu@localhost /]$ man -k printf
asprintf (3)         - imprimen en una cadena reservada
dprintf (3)          - imprimen en un descriptor de fichero
fprintf (3)          - conversión de salida formateada
fwprintf (3)         - conversión con formato de la salida de caracteres anchos
printf (3)           - conversión de salida formateada
snprintf (3)         - conversión de salida formateada
sprintf (3)          - conversión de salida formateada
swprintf (3)         - conversión con formato de la salida de caracteres anchos
vasprintf (3)        - imprimen en una cadena reservada
vdprintf (3)         - imprimen en un descriptor de fichero
vfprintf (3)         - conversión de salida formateada
vfwprintf (3)        - conversión con formato de la salida de caracteres anchos
vprintf (3)          - conversión de salida formateada
vsnprintf (3)        - conversión de salida formateada
vsprintf (3)         - conversión de salida formateada
vswprintf (3)        - conversión con formato de la salida de caracteres anchos
vwprintf (3)         - conversión con formato de la salida de caracteres anchos
wprintf (3)          - conversión con formato de la salida de caracteres anchos
fprintf (3p)         - print formatted output
fwprintf (3p)        - print formatted wide-character output
printf (1)           - format and print data
printf (1p)          - write formatted output
printf (3p)          - print formatted output
```

En este ejemplo podemos ver todos los manuales que contienen esa palabra.

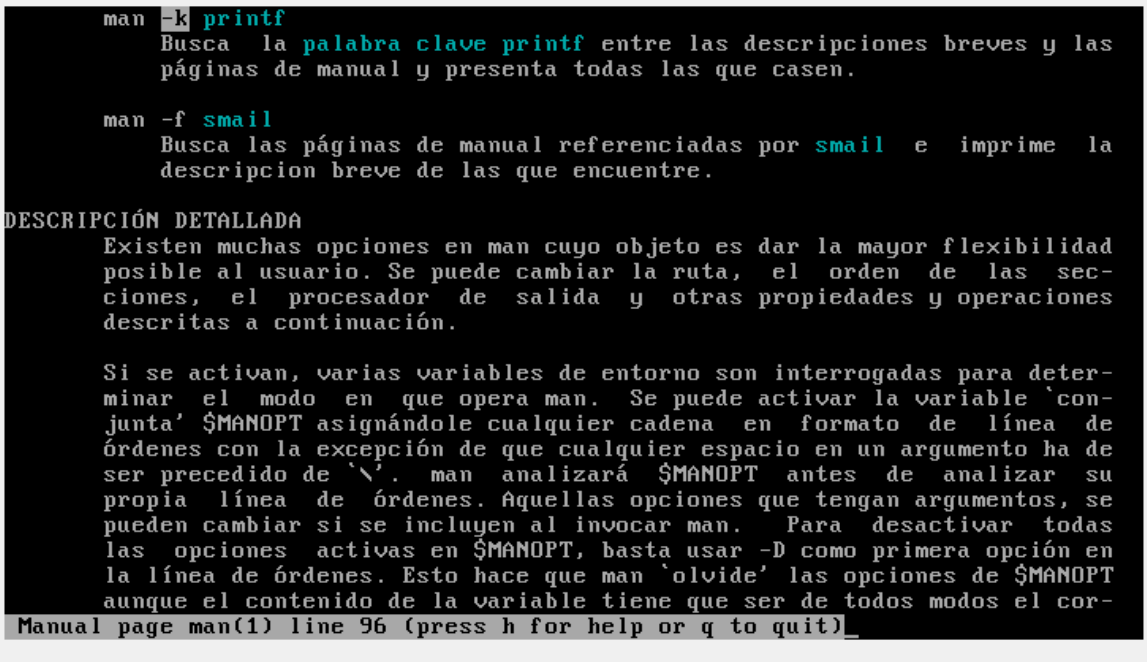
```
[picachu@localhost /]$
[picachu@localhost /]$ apropos printf
asprintf (3)         - imprimen en una cadena reservada
dprintf (3)          - imprimen en un descriptor de fichero
fprintf (3)          - conversión de salida formateada
fwprintf (3)         - conversión con formato de la salida de caracteres anchos
printf (3)           - conversión de salida formateada
snprintf (3)         - conversión de salida formateada
sprintf (3)          - conversión de salida formateada
swprintf (3)         - conversión con formato de la salida de caracteres anchos
vasprintf (3)        - imprimen en una cadena reservada
vdprintf (3)         - imprimen en un descriptor de fichero
vfprintf (3)         - conversión de salida formateada
vfwprintf (3)        - conversión con formato de la salida de caracteres anchos
vprintf (3)          - conversión de salida formateada
vsnprintf (3)        - conversión de salida formateada
vsprintf (3)         - conversión de salida formateada
vswprintf (3)        - conversión con formato de la salida de caracteres anchos
vwprintf (3)         - conversión con formato de la salida de caracteres anchos
wprintf (3)          - conversión con formato de la salida de caracteres anchos
fprintf (3p)         - print formatted output
fwprintf (3p)        - print formatted wide-character output
printf (1)           - format and print data
printf (1p)          - write formatted output
printf (3p)          - print formatted output
```

A veces queremos encontrar que es lo que hace un flag determinado de un comando que encontramos en alguna documentación pero el man del comando es muy extenso… en ese caso lo que pueden hacer es utilizar el comando "/" mientras están dentro del manual seguido

de la palabra clave que quieren buscar:



Al ingresar "/-k" en el manual de man, les seleccionara la primera ocurrencia encontrada, para ver la siguiente solamente deben apretar la letra "n" y para ir a la ocurrencia anterior utilicen la letra "N" (en mayúscula)



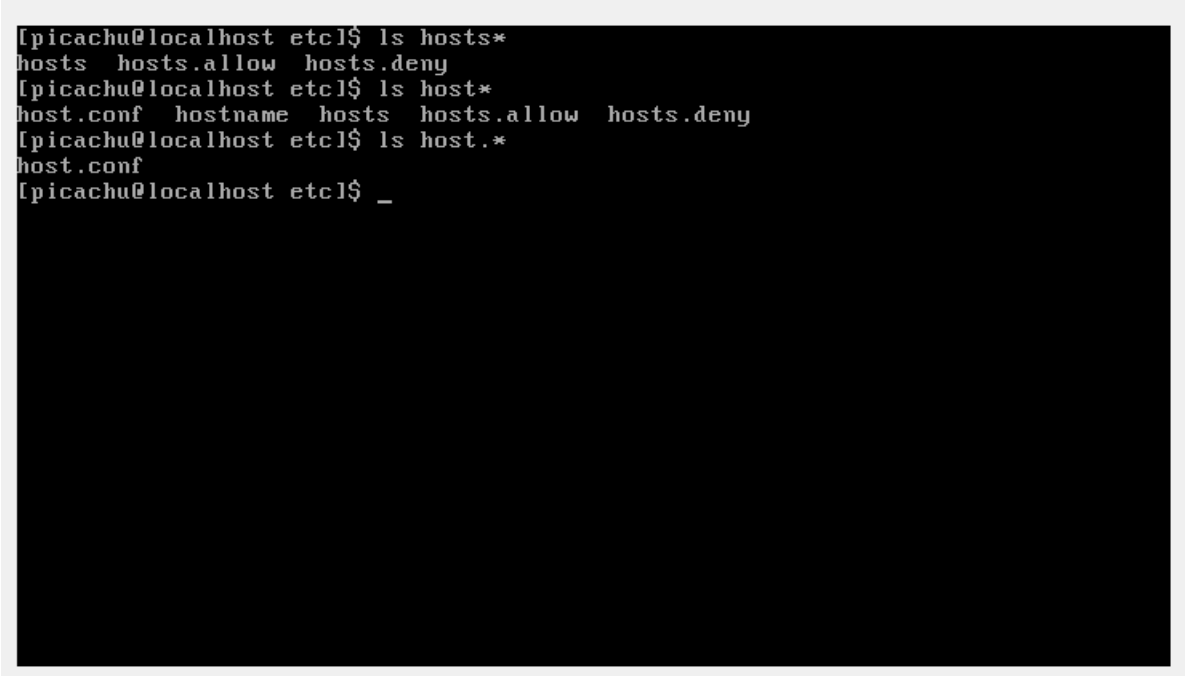
Globbing

Los shells son intérpretes de comandos, esto quiere decir que son capaces de hacer cadenas de operaciones complejas… Incluso algunos autores consideran que son similares a lenguajes de programación ya que esas cadenas se pueden almacenar en archivos ejecutables llamados scripts.

El globbing consiste en utilizar caracteres especiales interpretados por el Shell para realizar diferentes tareas. La combinación de estos caracteres especiales nos permite realizar tareas complejas en pocos pasos y llegar a resultados útiles de una forma sencilla. Aquí es donde radica el poder de los shells en UNIX y por ello los administradores necesitan saber cómo utilizar estas herramientas:

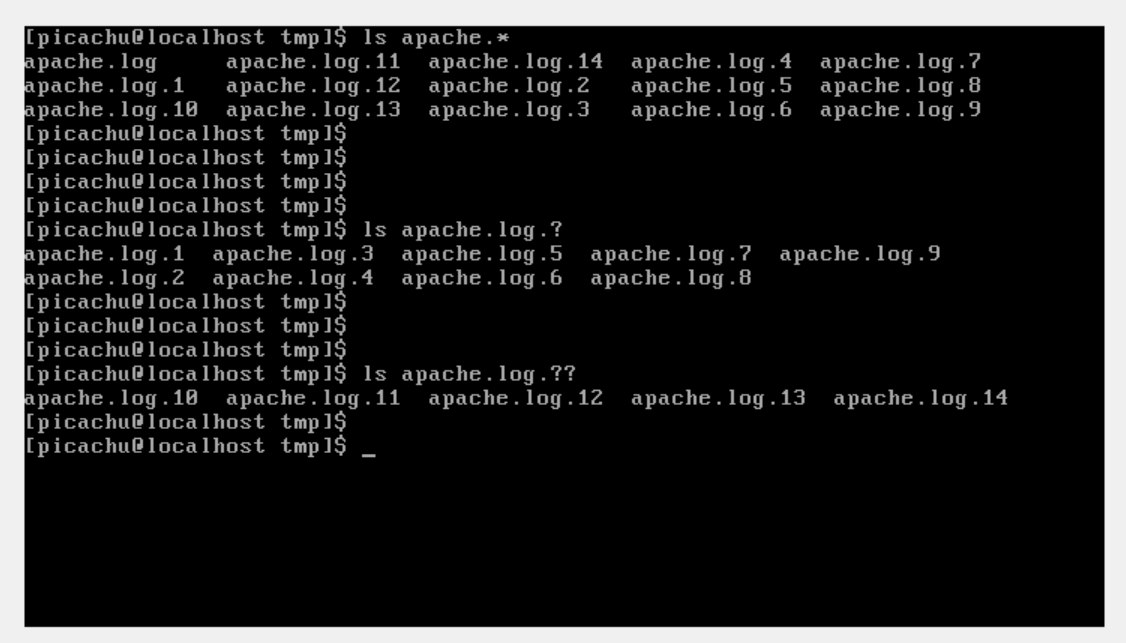
Los primeros caracteres especiales que utilizaremos son los denominados wildcards o comodines.

El asterisco (*) es el comodín universal que nos permite hacer cosas como buscar todos los archivos que comienzan o finalizan con ciertos caracteres, por ejemplo:



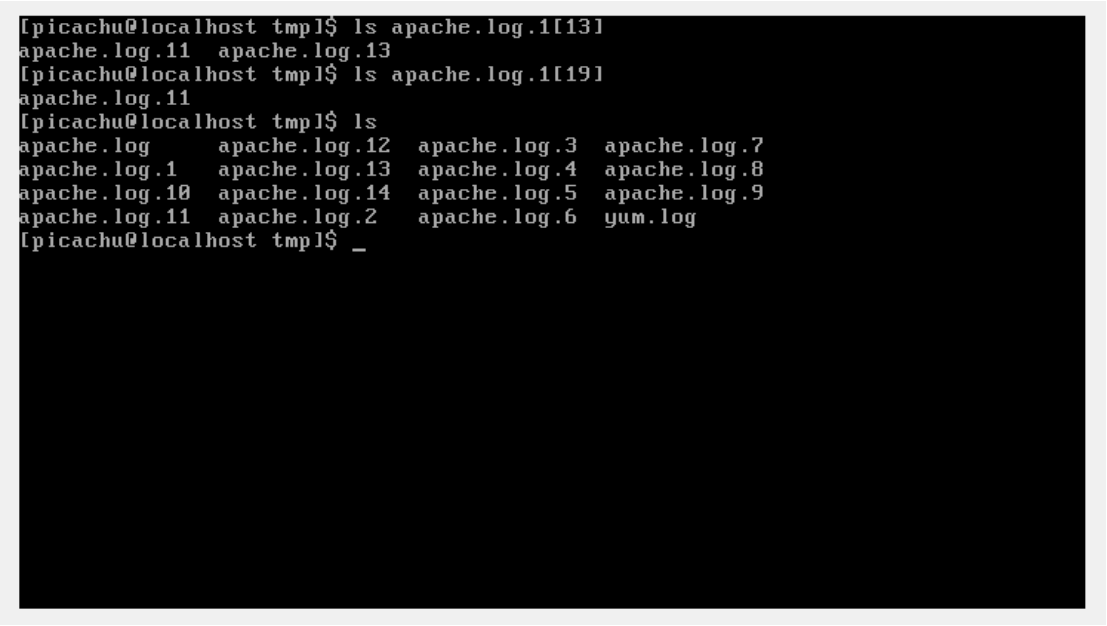
Nótese la diferencia en los resultados dependiendo de la ubicación del *.

Otro carácter especial muy útil es el signo de interrogación (?), lo que nos permite es utilizar como comodín un solo carácter, por ejemplo:



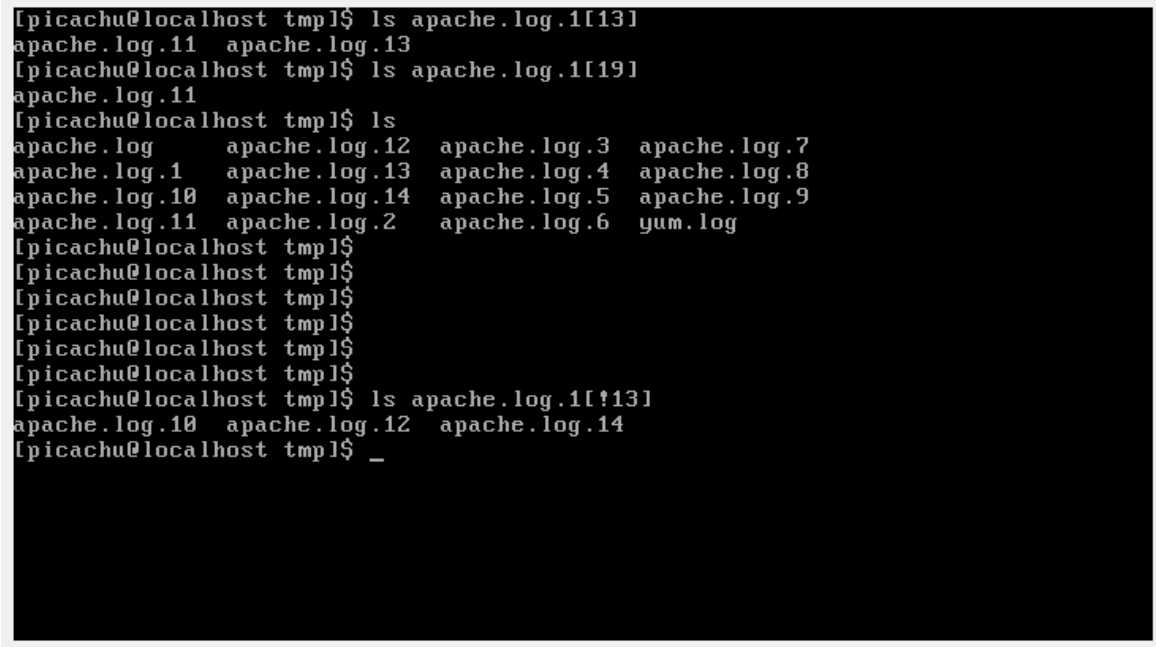
Nótese la diferencia de utilizar "*" versus "?" , en el primer caso me muestra tordos los resultados de cualquier archivo que contenga la cadena de caracteres "apache." Al comienzo, en cambio con un solo "?" solamente muestra los resultados que contienen un único carácter luego del "apache.log." pero si utilizo dos "??" muestra únicamente los resultados que tienen dos caracteres luego de "apache.log."

Otra opción muy útil son los corchetes, simbolizan opciones, es decir puedo poner dos o más caracteres dentro de los corchetes y el Shell mostrará los resultados encontrando uno u otro carácter en la posición que hayamos establecido, por ejemplo:



Como pueden ver aquí cuando pusimos como opción [13] en la segunda posición del número de log encontró las dos combinaciones, ya que existían ambos archivos, terminado en 11 y en 13. En cambio cuando pusimos como opción [19] solamente mostro el archivo en 11 ya que no existe el que termina en 19.

Existe una variante de esto que es utilizar el símbolo de exclamación (!) que significa negación.



Por ultimo veremos una opción muy interesante que es la de series, consiste en indicar un rango de opciones entre corchetes (siempre reemplazando al carácter en esa posición) por ejemplo [0-9] se leería “de cero a nueve”.

```
[picachu@localhost tmp]$ ls
apache.log     apache.log.13  apache.log.33  apache.log.5   apache.log.8
apache.log.1   apache.log.14  apache.log.35  apache.log.56  apache.log.89
apache.log.10  apache.log.2   apache.log.37  apache.log.6   apache.log.9
apache.log.11  apache.log.3   apache.log.39  apache.log.7   apache.log.99
apache.log.12  apache.log.31  apache.log.4   apache.log.75  yum.log
[picachu@localhost tmp]$
[picachu@localhost tmp]$
[picachu@localhost tmp]$ ls apache.log.[3-9][9-9]
apache.log.39  apache.log.89  apache.log.99
[picachu@localhost tmp]$
[picachu@localhost tmp]$
[picachu@localhost tmp]$
[picachu@localhost tmp]$ ls apache.log.[3-9]?
apache.log.31  apache.log.35  apache.log.39  apache.log.75  apache.log.99
apache.log.33  apache.log.37  apache.log.56  apache.log.89
[picachu@localhost tmp]$
[picachu@localhost tmp]$
[picachu@localhost tmp]$
[picachu@localhost tmp]$ ls apache.log.?[5-9]
apache.log.35  apache.log.39  apache.log.75  apache.log.99
apache.log.37  apache.log.56  apache.log.89
[picachu@localhost tmp]$ _
```

Y por supuesto que se pueden combinar todas las opciones anteriores para obtener los resultados que queremos en situaciones mucho más complejas de que los ejemplos que estamos viendo.

Está en ustedes aplicar de formas creativas las opciones de globbing que hemos visto para resolver las situaciones que afronten en la operatoria diaria de un administrador Linux.

Un tip que puede ser de mucha ayuda es que existe el flag “-d” para el comando ls que previene que muestre los directorios que se encuentran en el path donde estamos trabajando.

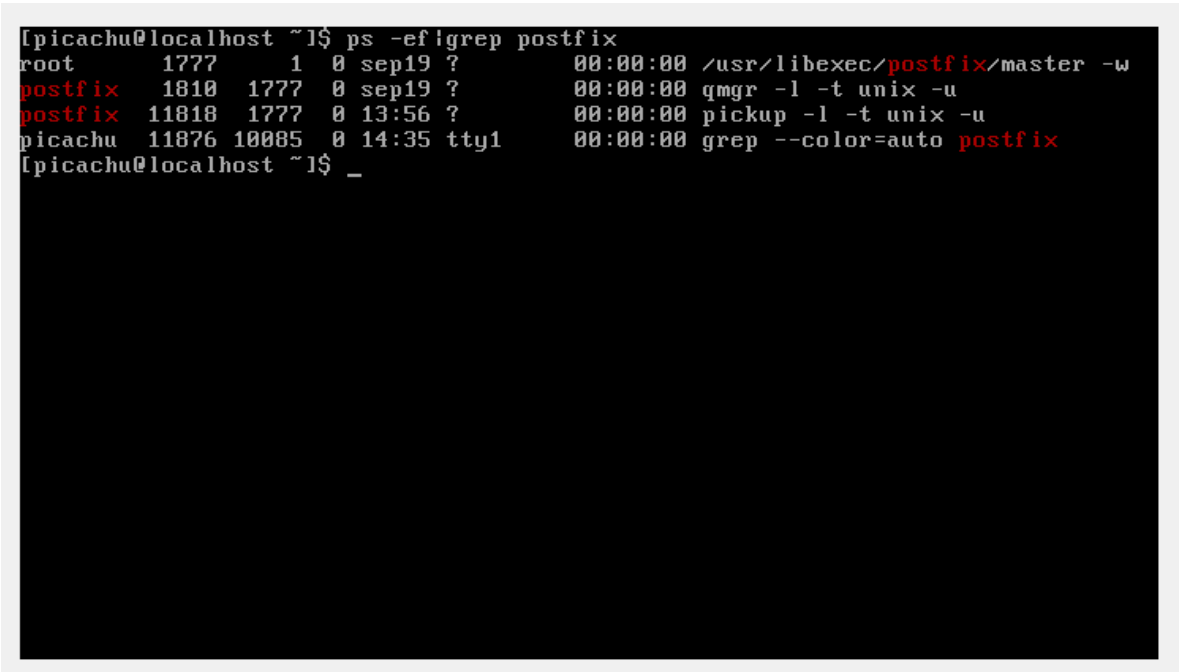
Redireccionamientos

Otra de las herramientas que deben aprender a usar son las redirecciones de comandos, se trata de combinar la salida de un comando para usarla como entrada de otro comando y así obtener un resultado deseado como pueden ser datos más reducidos u ordenados.

Para realizar dichos redireccionamientos vamos a utilizar el carácter reservado "|", en inglés se llama pipe que traducido literalmente significa tubería o caño en castellano.

Por ejemplo, el comando "ps" proporciona una "foto" de los procesos que se están ejecutando en este momento en nuestro sistema operativo. Y por otro lado el comando "grep" sirve para buscar cadenas de caracteres.

Lo que vamos a hacer es combinar ambos para obtener los resultados que queremos.



Aquí por ejemplo estamos filtrando la salida del comando "ps" con los flags "-ef" y estamos filtrando por la palabra clave "postfix"

En este caso por ejemplo estamos usando el comando “cat” que nos imprime en la pantalla los contenidos del archivo “/etc/passwd” y lo filtramos por la palabra clave “bash”.

Si analizamos el resultado, resulta interesante ver que en la primera columna vemos que menciona el usuario “root” y luego debajo el usuario “picachu” y esto es porque el archivo “passwd” es clave para nuestro sistema y es que contiene información de los usuarios que hemos configurado en el sistema.

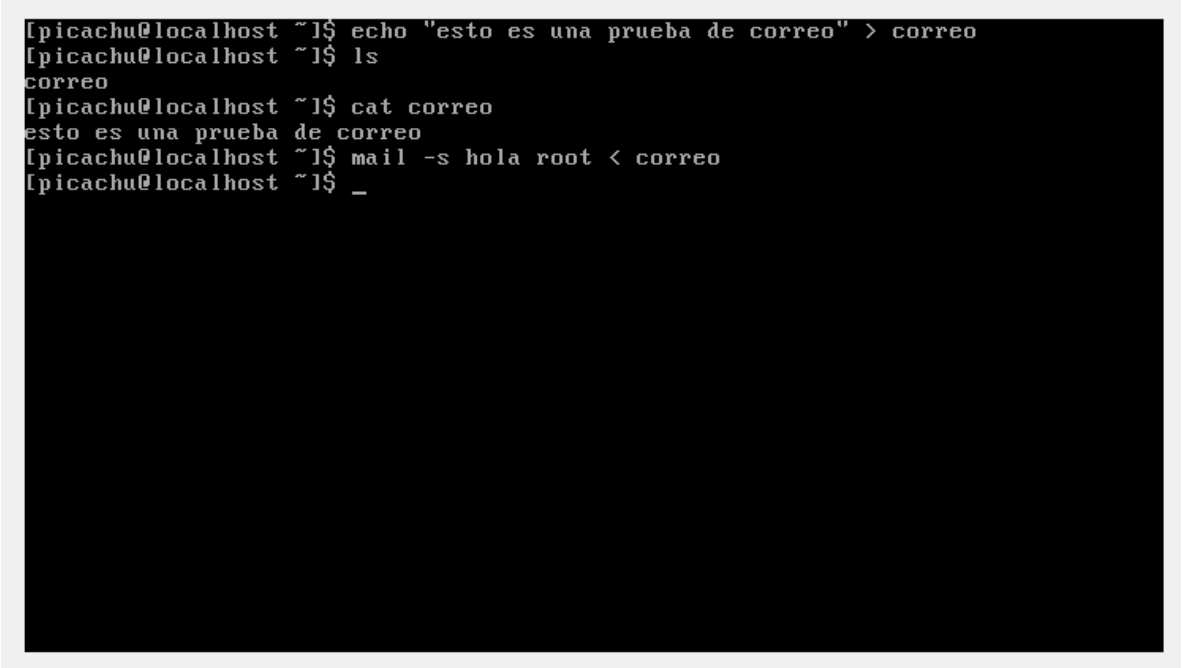
En la línea pueden ver datos separados por “:” como el home del usuario. En el caso de picachu es “/home/picachu” y en el caso de root es “/root”. También se puede ver que id de usuario tienen (es un número identificatorio único de cada usuario) y el gid (grup id) , en el caso de root el id es 0 y el gid también 0 dado que es el primero usuario que se crea.

Al final de la línea dice con qué Shell comenzaran estos usuarios cuando inicien su sesión, podríamos cambiarlo a otro Shell, pero no lo haremos por ahora ya que si cometemos algún error podríamos dejarnos fuera del sistema.

Flujos

En Linux todo proceso tiene 3 flujos de datos los cuales podemos manipular, el primero es el Standar Input (STDIN) que vendría a ser el flujo de información hacia el proceso, luego tenemos el Standar Output (STDOUT) que es el flujo de salida de información del proceso y por último el Standar Error (STDERR) que es donde el proceso va a enviar los mensajes de error de ejecución.

EL STDIN sirve para enviar información a los procesos o comandos, para ello utilizamos el carácter reservado "<" un ejemplo sería utilizando el comando "mail".



En el ejemplo pueden ver que con el comando "echo" enviamos un mensaje y lo direccionamos ">" a un archivo nuevo llamado correo.

Ejecutando el comando "cat correo" vemos que el contenido se guardó exitosamente y luego con "mail -s hola root < correo" estamos pasándole al comando mail como parámetro el mensaje que queremos enviar como contenido del correo.

Luego podemos loguearnos como usuario root "su -"y comprobar que el correo llegó al usuario de forma exitosa con el contenido que le quisimos enviar

```
[picachu@localhost ~]$ su -
Contraseña:
Último inicio de sesión:mié sep 20 11:55:09 ART 2017en tty1
[root@localhost ~]# mail
Heirloom Mail version 12.5 7/5/10.  Type ? for help.
"/var/spool/mail/root": 1 message 1 new
>N  1 picachu              Wed Sep 20 15:20  18/636   "hola"
& 1
Message  1:
From picachu@localhost.localdomain  Wed Sep 20 15:20:57 2017
Return-Path: <picachu@localhost.localdomain>
X-Original-To: root
Delivered-To: root@localhost.localdomain
Date: Wed, 20 Sep 2017 15:20:57 -0300
To: root@localhost.localdomain
Subject: hola
User-Agent: Heirloom mailx 12.5 7/5/10
Content-Type: text/plain; charset=us-ascii
From: picachu@localhost.localdomain (picachu)
Status: R

esto es una prueba de correo

& _
```

Por supuesto que esto es solamente un ejemplo y no quiere decir que sea la forma en que se manejarán para enviar un correo electrónico de ahora en adelante , claramente no es práctico con los sistemas de correo web que existen hoy en día, pero así es como se hacía antaño.

Algo importante para saber del STDOUT es que existen dos opciones, la primera es la que utilizamos “>” y la segunda es “>>”. En la primera opción la información que obtenemos “pisa” la información existente y en la segunda la concatena, es decir que la adjunta al final de cualquier información que previamente existiera.

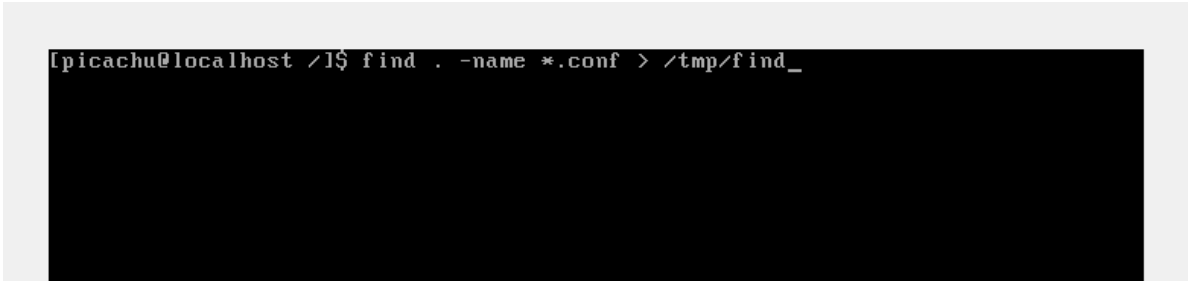
En este ejemplo lo verán claramente.

```
[picachu@localhost ~]$ ls
[picachu@localhost ~]$ echo "Esto es una prueba" > prueba.txt
[picachu@localhost ~]$ ls
prueba.txt
[picachu@localhost ~]$ cat prueba.txt
Esto es una prueba
[picachu@localhost ~]$ echo "Vamos a sumar texto a la prubea" >> prueba.txt
[picachu@localhost ~]$ ls
prueba.txt
[picachu@localhost ~]$ cat prueba.txt
Esto es una prueba
Vamos a sumar texto a la prubea
[picachu@localhost ~]$ echo "Rompé pepe , rompé" > prueba.txt
[picachu@localhost ~]$ ls
prueba.txt
[picachu@localhost ~]$ cat prueba.txt
Rompé pepe , rompé
[picachu@localhost ~]$ _
```

Por ultimo vamos a ver el STDERR que es el que utilizan los procesos para direccionar la información de errores que suceden mientras se están ejecutando. Por lo general estos flujos están dirigidos a los logs que podemos encontrar en "/var/log/", pero a veces puede ser de utilidad redirigir el flujo a otro lugar donde queramos guardar la información o en caso de que esté dirigido a nuestra pantalla por default y sea una molestia, lo mandamos a otro lugar.

Nuevamente con el ejemplo lo van a comprender muy claramente.

Lo que vamos a hacer es pararnos en el "/" con nuestro usuario picachu y vamos a ejecutar una búsqueda con el comando "find . -name *.conf", esto se lee buscar en donde estoy parado "." cualquier archivo que con el nombre * (comodín) y termine con la extensión ".conf" y luego lo metemos en el archivo "/tmp/find"



El resultado que verán en pantalla es una lista de errores de la siguiente forma:

Estos errores se deben a que estamos buscando en directorios para los cuales no tenemos permisos así que vamos a hacer lo siguiente, redirigir el STDERR.

```
[picachu@localhost /]$ find . -name *.conf 2> /tmp/errores_
```

Ahora sí, la información que vemos es el resultado que deseábamos y toda la información molesta la enviamos a un archivo para ser analizada más tarde.

```
./usr/share/alsa/pcm/hdmi.conf
./usr/share/alsa/pcm/iec958.conf
./usr/share/alsa/pcm/modem.conf
./usr/share/alsa/pcm/rear.conf
./usr/share/alsa/pcm/side.conf
./usr/share/alsa/pcm/surround21.conf
./usr/share/alsa/pcm/surround40.conf
./usr/share/alsa/pcm/surround41.conf
./usr/share/alsa/pcm/surround50.conf
./usr/share/alsa/pcm/surround51.conf
./usr/share/alsa/pcm/surround71.conf
./usr/share/alsa/smixer.conf
./usr/share/alsa/topology/broadwell/broadwell.conf
./usr/share/alsa/firmware/usx2yloader/us122.conf
./usr/share/alsa/firmware/usx2yloader/us224.conf
./usr/share/alsa/firmware/usx2yloader/us428.conf
./usr/share/abrt/conf.d/abrt-action-save-package-data.conf
./usr/share/abrt/conf.d/abrt.conf
./usr/share/abrt/conf.d/gpg_keys.conf
./usr/share/abrt/conf.d/plugins/xorg.conf
./usr/share/abrt/conf.d/plugins/oops.conf
./usr/share/abrt/conf.d/plugins/CCpp.conf
./usr/share/abrt/conf.d/plugins/vmcore.conf
./usr/share/abrt/conf.d/plugins/python.conf
[picachu@localhost /]$ _
```

Como pueden ver a continuación en el archivo /tmp/errores encontraremos toda la información que enviamos del STDERR

```
[picachu@localhost tmp]$ head errores
find: ‘./boot/grub2’: Permiso denegado
find: ‘./proc/tty/driver’: Permiso denegado
find: ‘./proc/1/task/1/fd’: Permiso denegado
find: ‘./proc/1/task/1/fdinfo’: Permiso denegado
find: ‘./proc/1/task/1/ns’: Permiso denegado
find: ‘./proc/1/fd’: Permiso denegado
find: ‘./proc/1/map_files’: Permiso denegado
find: ‘./proc/1/fdinfo’: Permiso denegado
find: ‘./proc/1/ns’: Permiso denegado
find: ‘./proc/2/task/2/fd’: Permiso denegado
[picachu@localhost tmp]$ _
```

Por ultimo veremos un comando que es muy útil llamado “sort” que sirve para ordenar.

En este ejemplo veremos cómo podemos utilizar este comando para ordenar un archivo y guardarlo en otro con los resultados en un solo paso.

```
[picachu@localhost ~]$ cd /etc/
[picachu@localhost etc]$ ls > ~/archivos.txt
[picachu@localhost etc]$ sort < ~/archivos.txt > ~/archivos_ordenados.txt
[picachu@localhost etc]$ cat ~/archivos_ordenados.txt _
```

Como pueden ver en con la línea “sort < ~/archivos.txt > ~/archivos_ordenados.txt” hacemos un STDIN y un STDOUT en una sola línea.

Ejercicios

- Encontrar utilizando el comando find todos los archivos que terminen con la extensión .log , guardar su nombre dentro de un archivo llamado logs_del_sistema dentro de nuestro home directory. El listado de archivos debe estar ordenado alfabéticamente.
- Imprimir dentro de un archivo llamado procesos_root en nuestro home directory el STDOUT del comando “ps –aux”, pero solamente la última columna.
    - Pista : investigar el comando cut y sus flags.

Shell Seguro

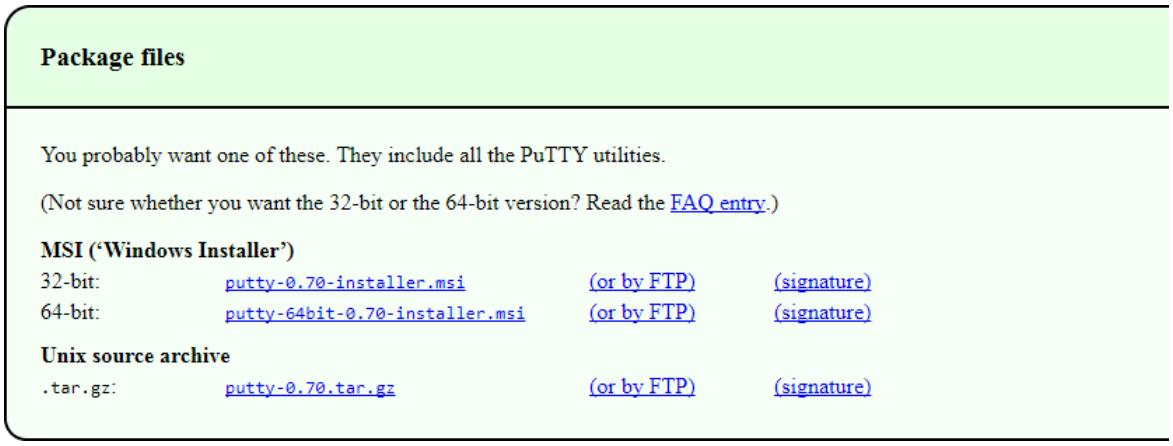
En esta sección veremos que es SSH o Secure Shell (Shell Seguro) y como lo utilizaremos para conectarnos a nuestros servidores de forma remota y segura.

SSH es un protocolo de comunicaciones que permite enviar y recibir información por un medio de forma encriptada, esto quiere decir que será mucho más difícil que puedan descifrar a información que estamos transmitiendo entre nuestros servidores y sus “clientes”.

Para poder utilizar SSH debemos tenes 2 cosas, un servidor de SSH y un software de cliente ssh que nos permita conectarnos al servidor.

Para este curso usaremos un software llamado Putty que es ampliamente conocido y utilizado en el ambiente, aunque no es el único ni el mejor.

Para poder descargarlo deben ingresar al siguiente link y descargar el archivo que dice installer 64bits



Package files

You probably want one of these. They include all the PuTTY utilities.

(Not sure whether you want the 32-bit or the 64-bit version? Read the FAQ entry.)

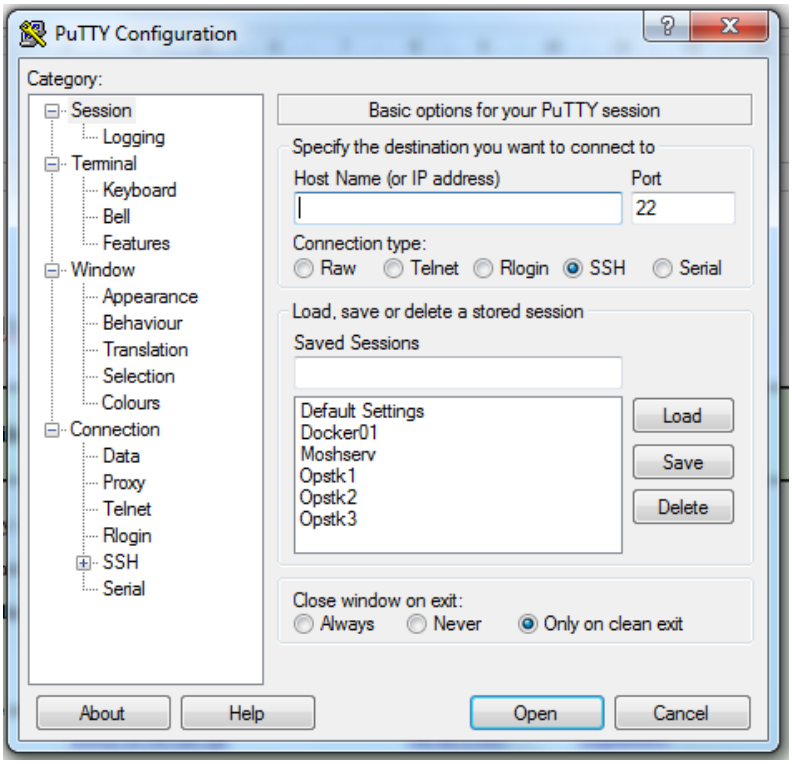
MSI ('Windows Installer')

32-bit:	putty-0.70-installer.msi	(or by FTP)	(signature)
64-bit:	putty-64bit-0.70-installer.msi	(or by FTP)	(signature)

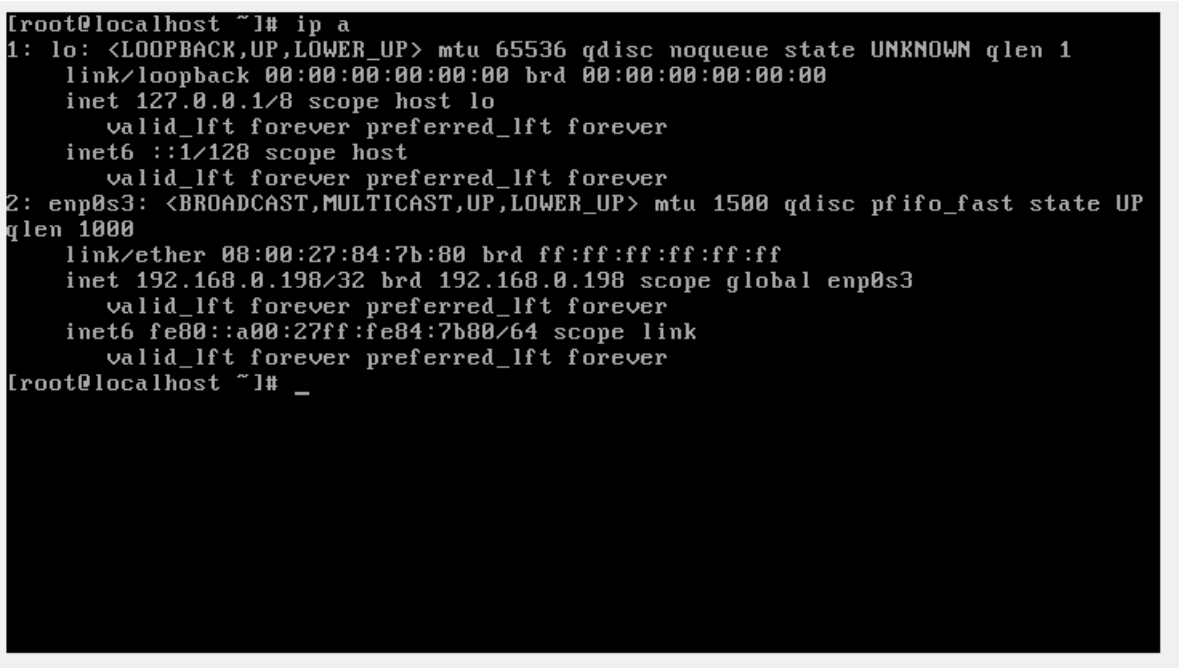
Unix source archive

.tar.gz:	putty-0.70.tar.gz	(or by FTP)	(signature)

Luego de instalarlo al ingresar a Putty verán la siguiente pantalla:

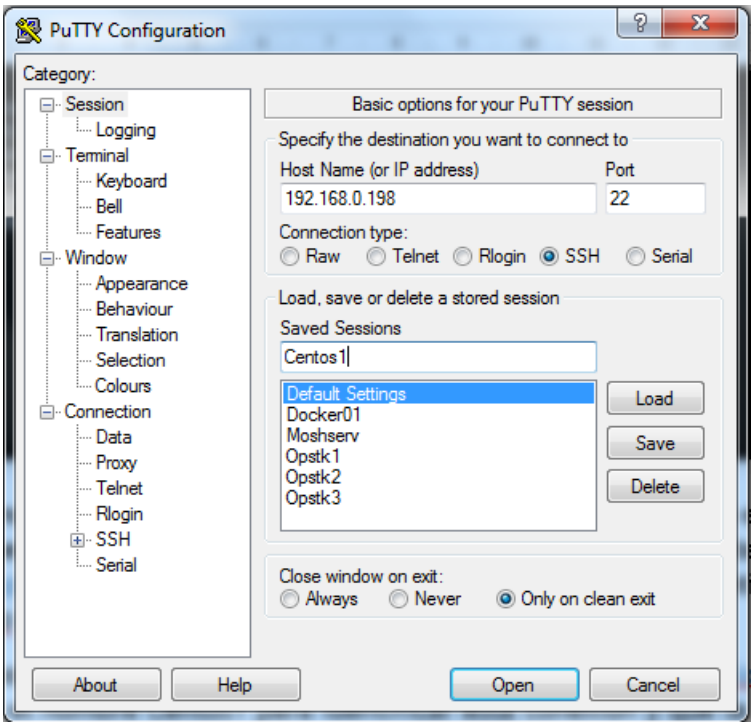


Lo que necesitamos ahora es la IP de nuestro servidor, para ver que ip tenemos usaremos el comando "ip a" que nos dará la siguiente información :



Aquí podemos ver que hay 2 placas de red (en realidad la llamada lo es la loopback pero ahora no entraremos en detalle de eso), la que nos interesa es la llamada enp0s3 en la segunda línea pueden ver que dice inet y a continuación la dirección ip que en mi caso es "192.168.0.198".

Ingresaremos esa dirección en putty y dentro del campo "Saved Sessions" pondremos el nombre Centos1 para identificar esta conexión y que la podamos utilizar más adelante sin tener que recordar la IP de memoria, para ello haremos click en "Save" y se agregará a la lista de conexiones conocidas.



Después hacemos click en Open y nos abrirá una pantalla similar a la Terminal que habíamos usado cuando teníamos el entorno gráfico en Linux, aunque primero debemos aceptar la advertencia de seguridad que nos mostrará:

Ahora si nos hemos conectado exitosamente a nuestro servidor a través de un túnel de SSH.

Ejercicio

- Crear una máquina virtual ya sea clonando la existente o instalando desde cero una nueva, obtener la dirección ip de la misma y realizar una conexión ssh desde una virtual hasta la otra.
  Pista: deberán utilizar el man del comando ssh y encontrar en alguna sección algo que les pueda servir de ejemplo.

Bibliografía Recomendada

Aprendiendo a Aprender Linux
- Vladimir Támara
- Jaime Irving Dávila
- Pablo Chamorro
- Igor Támara
Febrero de 2003