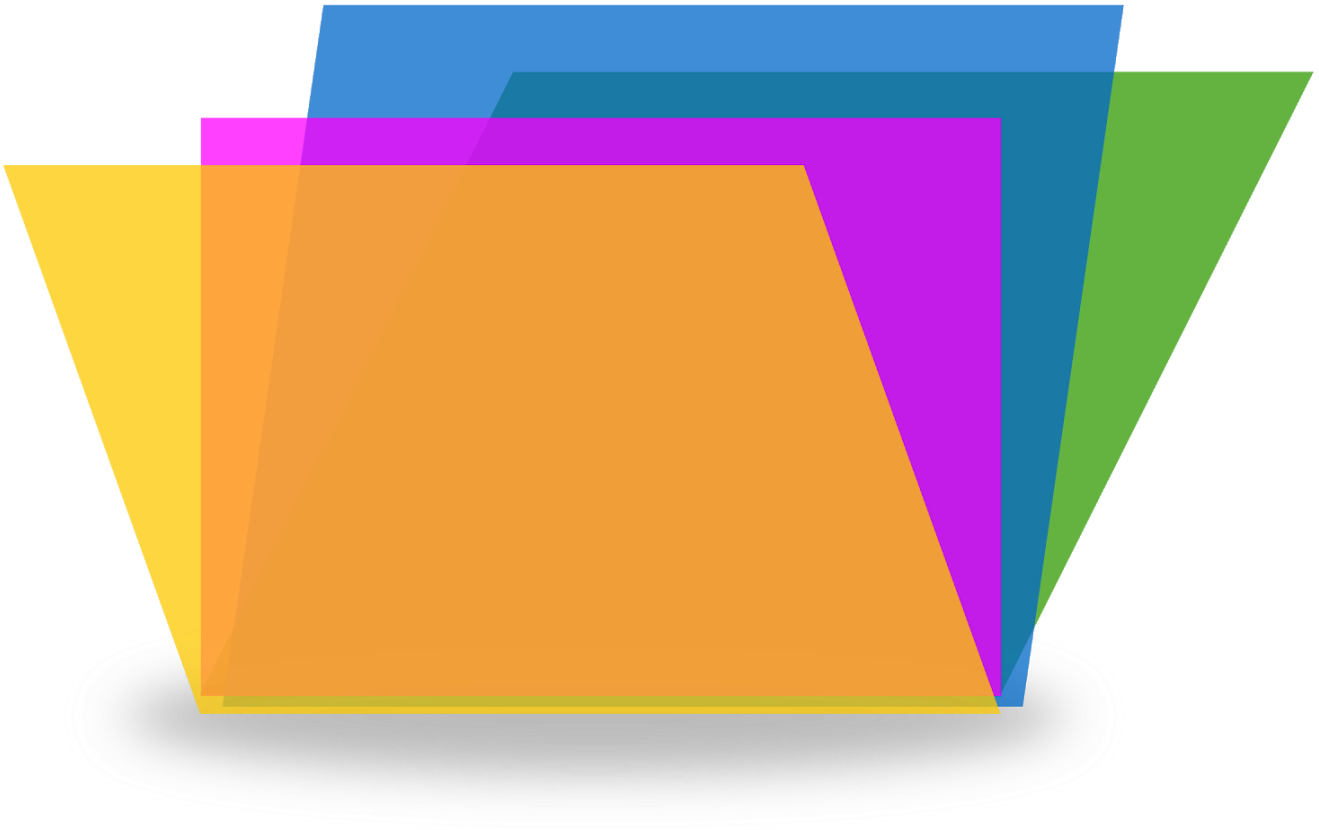
Administración de Archivos



📌 Información sobre LPI

Tema Cubierto

103.3 Realizar administración básica de archivos

Peso

4 (El peso indica la cantidad de preguntas relativas al tema en cuestión, es decir cuánto más grande el número, es más su indicencia en el resultado del examen. Su valor puede ir de 1 a 6)

Descripción

Los alumnos deberán ser capaces de interactuar con shells y comandos usando la línea de comandos. El objetivo asume la shell Bash.

Áreas Claves de Conocimiento

- **Copiar, mover y borrar archivos y directorios**
- **Copiar archivos y directorios recursivamente**
- **Borrar archivos y los directorios recursivamente**
- **Usar las especificaciones simples y avanzadas de caracteres comodines**
- **Usar find para buscar y actuar sobre archivos basado en su tipo, tamaño o fecha**
- **Usar tar, cpio y dd**

Términos y herramientas

- **cp**
- **find**
- **mkdir**
- **mv**
- **ls**
- **rm**
- **rmdir**
- **touch**
- **tar**
- **cpio**
- **dd**
- **file**
- **gzip**
- **gunzip**
- **bzip2**
- **bunzip2**
- **xz**
- **unxz**
- **Metacaracteres de archivos**

Nota: La especificación avanzada de metacaracteres se trata en el curso de Linux Shell Scripting

ADMINISTRACIÓN BÁSICA DE ARCHIVOS

Objetos del sistema de archivos

Los sistemas de archivos que vamos a explicar poseen objetos que se agrupan en un árbol jerárquico. Esta organización nos brinda la posibilidad de contener diversos archivos, donde algunos tendrán el mismo nombre, pero al estar organizados en una forma de árbol jerárquico podremos realizar diversas tareas; entre ellas, “ver” nombres de archivos iguales, (¿pero cada objeto es único?) La manera en que se administran estos objetos es mediante tablas propias del sistema de archivos, las cuales tienen diferentes propiedades. Ahora nos focalizaremos en los directorios y los archivos.

Directorios y archivos

Los directorios son contenedores que van a incluir la información de los objetos, tanto de directorios como de otros archivos. En Linux, el directorio de mayor jerarquía es la raíz “/”, a partir de ahí surgen todos los demás directorios y archivos. Un archivo, por otro lado, fes el que va a contener información; en cambio, un directorio, nos da un orden de cómo vamos a poder ver todos los archivos y/o directorios. Para poder especificar una ruta se antepone ”/” porque todo nace de ahí; en cambio, si quisiéramos nombrar un archivo de nuestro home, la ruta completa del archivo sería la siguiente: /home/nombre_usuario/test.sh No debemos confundir root filesystem con root home directory.

Inodos

La forma en que el sistema de archivo identifica a los objetos es mediante los inodos. Estos traen información acerca de los objetos; cada uno tiene distintas propiedades, que, dependiendo de su uso, va a cambiar el valor que contenga (por ejemplo: datos que posee, ubicación, fecha de modificación, tamaño, datos seguridad, etc).

El comando stat es de gran utilidad para mostrar esta información, por ejemplo:

```
# stat /etc/resolv.conf
  Fichero: /etc/resolv.conf
  Tamaño: 125           Bloques: 8          Bloque E/S: 4096   fichero regular
Dispositivo: fd00h/64768d       Nodo-i: 1050548  Enlaces: 1
Acceso: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
         Acceso: 2019-03-21 14:32:29.532283100 -0300
Modificación: 2019-03-21 14:32:29.432282543 -0300
         Cambio: 2019-03-21 14:32:29.439282582 -0300
         Creación: -
```

Es decir, este comando nos informa sobre las propiedades de un archivo:

- Nombre
- Tamaño
- Tipo de archivo
- El número de i-nodo
- La cantidad de enlaces, que en archivos regulares refiere a cuántos archivos tienen el mismo número de i-nodos.
- Permisos y propietarios
- Marcas de tiempo
    - Acceso (atime): La última vez que se accedió al archivo.
    - Modificación (mtime): La última vez que se modificó el contenido de un archivo.
    - Cambio (ctime): La última vez que se cambiaron metadatos del archivo, por ejemplo, los permisos.
    - Creación: La fecha de creación tradicionalmente no se usa en los sistemas de archivos en Unix.

.

COMANDOS BÁSICOS

Comando ls

El comando ls se utiliza para mostrar el contenido de directorios, como así también ver propiedades de los distintos archivos.

Sintaxis:
ls [opciones] [archivo1] [archivo2] [archivoN]
ls [opciones] [directorio1] [directorio2]

Las opciones más utilizadas son:
-a Muestra todos los archivos, incluso los ocultos
- l Muestra el tipo de archivo, permisos, propietarios, tamaño, fecha de modificación)
-h Muestra el tamaño de los archivos en una unidad fácil de leer
-r Ordena los archivos de manera inversa
-t Ordena los archivos por fecha de modificación

Ejemplo:

```
# ls -ltr /etc/X11/
total 24
drwxr-xr-x. 2 root root 4096 feb  3  2016 applnk
drwxr-xr-x. 2 root root 4096 ago 11  2016 fontpath.d
-rw-r--r--  1 root root  493 dic 14 08:27 Xresources
-rw-r--r--  1 root root  547 dic 14 08:27 Xmodmap
drwxr-xr-x. 2 root root 4096 mar 15 15:38 xorg.conf.d
drwxr-xr-x. 5 root root 4096 abr  8 20:51 xinit
```

Comando cp

El comando cp se utiliza para copiar archivos/directorios.

Sintaxis:

cp [opciones] archivo1 archivo2

cp [opciones] archivos directorio

Opciones más frecuentes

-f Fuerza la reescritura si ya existe el destino.

-i Me habilita el “prompt” interactivo para evitar la sobreescritura.

-p Preserva todos los permisos de la estructura del directorio y/o archivo, incluyendo la fecha de creación.

-r,-R Hace una copia recursiva, para poder copiar todo un directorio y sus subdirectorios.

-v modo “verbose” en donde muestra lo que está haciendo.

Entonces, lo que podemos hacer es copiar un archivo para que en el destino tenga otro nombre; también podemos copiar varios archivos a un directorio.

```
# cp archivo1 archivo2
```

El resultado fue la copia del archivo1 al archivo2.

```
# ls -l archivo2
-rw-r--r-- 1 root root 0 Jun 5 05:41 archivo2
```

Vemos que el archivo creado es nuevo (se darán cuenta por la fecha)

Para copiar varios archivos a un directorio.

Primero, creamos un directorio.

Segundo, copiamos los dos archivos (archivo1, archivo2) al directorio creado.

```
# mkdir dir1
# cp archivo1 archivo2 dir1
# ls -l dir1
-rw-r--r-- 1 root root 0 Jun 5 05:43 archivo1
-rw-r--r-- 1 root root 0 Jun 5 05:43 archivo2
```

Presten atención al error que aparece. Esto se debe a que le dimos un path (es decir, una ruta) absoluto y el directorio dir1 no existe en /dir1 sino que está en /root/dir1 porque fue creado en nuestro home.
Luego al ejecutar ls al directorio dir1, tampoco usamos un path absoluto (/root/test/dir1, sino ls -l dir1) porque, precisamente, estamos parados dentro de un directorio que contiene a “dir1”.
Con el comando **pwd**, nos dirá en qué directorio estamos parados.

```
# pwd
/root/test
# ls -l
total 4
-rw-r--r-- 1 root root 0 Jun 5 05:41 archivo1
-rw-r--r-- 1 root root 0 Jun 5 05:41 archivo2
drwxr-xr-x 2 root root 4096 Jun 5 05:43 dir
```

Ejemplos:

Funcionamiento de la opción -v

```
# cp -v archivo1 archivo2
`archivo1' -> `archivo2'
```

Aquí agregamos el -i ,para que vean que nos preguntará si queremos sobreescribir.

```
# cp -i archivo1 archivo2
cp: overwrite `archivo2'?
```

Aquí vemos el funcionamiento de la opción -p
Listamos con -l para que vean los permisos y la fecha

```
# ls -l archivo1
-rw-r--r-- 1 root root 0 Jun 5 05:41 archivo1
# date
Sun Jun 5 05:54:02 ART 2011
```

Ahora lo copiamos

```
# cp -p archivo1 nuevoarch
# ls -l nuevoarch
-rw-r--r-- 1 root root 0 Jun 5 05:41 nuevoarch
```

Y como ven, el archivo nuevo conserva la misma fecha; distinto sería si hubiésemos hecho esto.

```
# cp archivo1 narch
# ls -l narch
-rw-r--r-- 1 root root 0 Jun 5 05:55 narch
```

Aquí vamos a copiar recursivamente (un directorio y todo su contenido):

```
# cp -r /etc/ dir1/
# ls -l dir1/
total 4
-rw-r--r-- 1 root root 0 Jun 5 05:43 archivo1
-rw-r--r-- 1 root root 0 Jun 5 05:43 archivo2
drwxr-xr-x 34 root root 4096 Jun 5 05:57 etc
```

Al ejecutar **ls -l dir**, vemos que está el directorio “etc”, pero la fecha es de recién,(si usamos la opción -p se mantienen la fecha y demás privilegios)

Manual: **man cp**

Comando mkdir

Este comando se utiliza para crear directorios.

```
# mkdir testing
# ls -ld testing/
drwxr-xr-x 2 root root 4096 Jun 12 02:56 testing/
```

Otro caso es si queremos crear un directorio que se encuentra adentro de otro. Pero tener en cuenta que si ninguno de los directorios que ponemos existe nos dará error.

Ejemplo con error:

```
# mkdir es/otro/directorio/lejano
mkdir: cannot create directory `es/otro/directorio/lejano': No such file or directory
```

Ejemplo para crear un directorio y una cadena de subdirectorios:

```
# mkdir -p es/otro/directorio/lejano
# ls -ld es/otro/directorio/lejano/
drwxr-xr-x 2 root root 4096 Jun 12 02:57 es/otro/directorio/lejano/
```

Comando mv

Este comando se utiliza para mover o renombrar archivos y directorios.

Sintaxis:

mv [opciones] origen destino

Opciones más usadas.

-f si ya existe lo fuerza a sobreescribir.

-i activa el modo interactivo para aceptar los cambios uno por uno.

Ejemplo para cambiar de nombre:

```
# mv archivo1.txt nuevonombre.txt
```

Ejemplo para mover:

```
# mv archivo1.txt /directorio
```

Comando cd

Este comando se utiliza para avanzar o retroceder niveles de directorios.

Ejemplo:

```
# cd dir #  (cambio a un directorio hijo)
# cd .. # (retrocedo)
# cd  #  (me lleva a mi home)
```

Comando rm

Este comando sirve para borrar tanto directorios como archivos.

rm [opciones] archivos

-f hace un borrado sin preguntar nada.

-i en modo interactivo

-r,-R para borrar recursivamente

Ejemplo para borrar un directorio y TODO su contenido:

```
# rm -rf /directorio
```

Comando rmdir

Se utiliza solamente para borrar directorios vacíos.

rmdir [opciones] directorio

```
# rmdir dir
```

Nota: generalmente se utiliza el comando rm -rf

Comando touch

Este comando cambia la fecha/hora de acceso o modificación de los archivos.

touch [opciones] archivo

Opciones:

-a solo cambia el tiempo de acceso.

-m cambia el de modificación.

-t timestamps definir la fecha a utilizar [[CC]YY]MMDDhhmm[.ss]. Por ejemplo la fecha y hora para el 12 de Enero 2001 a las 18:45 es 200101121845

Ejemplo:

```
# ls -l archivo1
-rw-r--r-- 1 root root 0 Jun 5 05:41 archivo1
# touch archivo1
# ls -l archivo1
-rw-r--r-- 1 root root 0 Jun 12 04:52 archivo1

# ls -l archivo2
-rw-r--r-- 1 root root 0 Jun 12 04:53 archivo2
# touch -t 200101121845 archivo2
```

```
# ls -l archivo2
-rw-r--r-- 1 root root 0 Jan 12 2001 archivo2
```

Comodines

Los comodines o metacaracteres (file globbing) nos van a servir para tomar acción sobre múltiples archivos/directorios que contengan cierto cadena de caracteres en su nombre. Tener en cuenta que los comodines son interpretados por la shell y no por los comandos, cuando un comando es puesto con comodines, la shell los interpreta a estos y otros tipos de expansiones, para luego ejecutar el comando. Este proceso es invisible para el usuario.

Para referirnos a uno o más archivos lo podemos hacer de una manera más acotada. La siguiente tabla nos muestra un resumen.

Comodín	Descripción	Ejemplo
*	Coincide con cualquier cadena de texto, incluso con la cadena vacía. De manera predeterminada se excluyen los nombres de archivos que comienzan con "."	`rm *` (borrará todos los archivos en el directorio en el que estemos parados) rm *ho (borrará todo lo que comience con ho) `rm *` la (borrará todos los archivos que terminen con la cadena de texto 'ho')
?	Coincide con un único carácter	`ls -l /bin/z???` va a coincidir con archivos que comienzan con z seguidos exactamente 3 caracteres
[lista o rango]	Coincide con cualquier de los caracteres encerrados entre corchetes	`ls -l /bin/[a-c][ar]??` va a coincidir con los archivos cuyo nombre empieza con a, b, ó c, el segundo carácter puede ser una a o una r. Los dos últimos caracteres pueden ser cualquiera.
[!lista o rango]	Coincide con cualquiera de los caracteres no encerrados entre corchetes	`ls -l /bin/[a-c][ar]??` **no** va a coincidir con los archivos cuyo nombre empieza con a, b, ó c, el segundo carácter puede ser una a o una r y con dos caracteres finales
{ cadena1, cadena2... }	Se usa para generar cadenas de texto arbitrarias	`mkdir dir{1,2,3,4}` generará los directorios dir1, dir2, dir3 y dir4.

Más ejemplos

```
# ls -l hola*
-rw-r--r-- 1 root root 0 Jun 12 05:08 hola1
-rw-r--r-- 1 root root 0 Jun 12 05:08 hola2
-rw-r--r-- 1 root root 0 Jun 12 05:08 hola4
-rw-r--r-- 1 root root 0 Jun 12 05:08 hola5

# ls -l ?ola1
-rw-r--r-- 1 root root 0 Jun 12 05:08 hola1
```

Busca que un caracter concuerde con alguno de los que esta adentro de la lista, siempre y cuando la misma sea equivalente a la posición de donde se lo defina.

```
# ls -l [haxt4]ola[12]
-rw-r--r-- 1 root root 0 Jun 12 05:08 hola1
-rw-r--r-- 1 root root 0 Jun 12 05:08 hola2

# ls -la hola[2-4]
-rw-r--r-- 1 root root 0 Jun 12 05:08 hola2
-rw-r--r-- 1 root root 0 Jun 12 05:08 hola4
# ls -la ho[a-z]a[2-4]
-rw-r--r-- 1 root root 0 Jun 12 05:08 hola2
-rw-r--r-- 1 root root 0 Jun 12 05:08 hola4

[!a-z] El mismo ejemplo que el anterior pero ahora negando todo.
# ls -l hola[!a-z]
-rw-r--r-- 1 root root 0 Jun 12 05:08 hola1
-rw-r--r-- 1 root root 0 Jun 12 05:08 hola2
-rw-r--r-- 1 root root 0 Jun 12 05:08 hola4
-rw-r--r-- 1 root root 0 Jun 12 05:08 hola5
```

Estas opciones me va a permitir crear string para diferentes usos.

Ejemplo:

```
# touch hola{1,2,4,5}
# ls ?[poi][!poi][a-z][1-9]
hola1 hola2 hola4 hola5
```

En un solo paso copio un archivo.

```
# cp hola1{,.bkp}
# ls -l hola1 *.bkp
-rw-r--r-- 1 root root 0 Jun 12 05:08 hola1
-rw-r--r-- 1 root root 0 Jun 12 05:45 hola1.bkp
```

Comando gzip / gunzip

Con estos comando vamos a poder comprimir y descomprimir archivos (utiliza el algoritmo Lempel-Ziv) .

gzip es uno de las herramientas más antiguas de compresión en Linux.

La sintaxis es:

gzip [opciones] [archivo]

Las opciones más comunes son:

-d Para descomprimir un archivo (Equivale al comando gunzip)

-1 La compresión más rápida

-9 La mejor compresión

```
$ gzip keepassx-0.4.3-4.1.i386.rpm
$ ls -l
-rw-rw-r--  1 rino rino 1.1M Apr 29 22:51 keepassx-0.4.3-4.1.i386.rpm.gz
$  bunzip2 keepassx-0.4.3-4.1.i386.rpm.gz
```

Comando bzip2 / bunzip2

Con este comando vamos a poder comprimir y descomprimir archivos (utiliza el algoritmo Burrows-Wheeler y Huffman) .

bzip2 es considerado uno de los más eficientes programas de compresión disponibles para Linux y su sintaxis y opciones es prácticamente idéntica a gzip/gunzip.

Generalmente tiene la extensión bz2

Ejemplos de como usarlo:

La opción -d es para descomprimir, también se puede usar el comando bunzip2

Con la opción -9 (va de 1 a 9) se define el nivel de compresión. El 9 será la compresión máxima y el 1 la mínima.

Tengan en cuenta que el archivo mantendrá el nombre y le agregará la extensión bz2.

```
$ bzip2 keepassx-0.4.3-4.1.i386.rpm
$ ls -l
-rw-rw-r--  1 rino rino 1.1M Apr 29 22:51 keepassx-0.4.3-4.1.i386.rpm.bz2
$  bunzip2 keepassx-0.4.3-4.1.i386.rpm.bz2
```

Comando xz

Esta herramienta usa el algoritmo LZMA, ofrece altos grados de compresión de archivos, así como también, velocidad para la compresión y descompresión.

Para **comprimir** un archivo se debe hacer lo siguiente:

```
$ xz lxle.iso
$ ls -l
-rw-rw-r--  1 sergio sergio 1.2G Feb 24 22:51 lxle.iso..xz
```

Para descomprimir un archivo:

```
$ xz -d lxle.iso.xz
$ ls -l
-rw-rw-r--  1 sergio sergio 1.4G Feb 24 22:51 lxle.iso..xz
```

Se puede lograr lo mismo usando el comando **unxz**:

```
$ unxz lxle.iso.xz
```

```
$ ls -l
-rw-rw-r--  1 sergio sergio 1.4G Feb 24 22:51 lxle.iso..xz
```

Al igual que los comandos **gzip** y **bzip2**, **xz** se puede combinar con tar para generar un archivo con extensión .**tar.xz.**

Comando tar

Los comandos bzip, gzip y xz pueden comprimir un único archivo a la vez. Este comando sirve para poder agrupar archivos y directorios, y comprimirlos con esas herramientas. El archivo resultante suele llamarse tarball.

Opciones:

c para crear

x para extraer

v modo verbose, muestra lo que se está haciendo

f define el archivo.tar

z comprime/descomprime con gzip

j comprime/descomprime con bzip2

J comprime/descomprime con xz

a Usa la extensión del archivo para elegir la herramienta de compresión

t Muestra el contenido del tarball

```
# tar -cvf boinc.tar ../Programas/BOINC
../Programas/BOINC/
../Programas/BOINC/run_client
../Programas/BOINC/get_project_config.xml
../Programas/BOINC/lookup_website.html
../Programas/BOINC/statistics_www.worldcommunitygrid.org.xml
../Programas/BOINC/time_stats_log
../Programas/BOINC/installingwwer_eng
```

Generalmente las opciones de un comando van precedidas por guiones medios (-), sin embargo para el comando tar podemos prescindir de ellos en muchos casos.

En este ejemplo se agrupó todo el contenido del directorio provisto en un archivo tar. Para ver el contenido del archivo.

```
# tar tf boinc.tar
Programas/BOINC/
Programas/BOINC/run_client
Programas/BOINC/get_project_config.xml
Programas/BOINC/lookup_website.html
Programas/BOINC/statistics_www.worldcommunitygrid.org.xml
Programas/BOINC/time_stats_log
Programas/BOINC/installingwwer_eng
# ls -lh boinc.tar
-rw-rw-r-- 1 rino rino 623M Sep  2 21:47 boinc.tar
```

Para comprimirlo

```
# bzip2 -9 boinc.tar
# ls -lh boinc.tar.bz2
-rw-rw-r-- 1 rino rino 179M Sep  2 21:47 boinc.tar.bz2
```

Para recuperar los datos

Descomprimir:

```
# bunzip2 boinc.tar.bz2
# tar xvf boin.tar.bz2
```

Otro método

Crea el tar y lo comprime con bzip2

```
# tar cvjf etc.tar.bz2 /etc
```

Luego:

```
# ls -lh etc.tar.bz2
-rw-rw-r-- 1 rino rino 7.9M Sep  2 21:56 etc.tar.bz2
```

Extraer y descomprimir

```
# tar xvjf etc.tar.bz2
```

Comando cpio

📌**Este comando se usa para crear y extraer archivos o copiar archivos desde un lugar hacia otro. No utiliza compresión por sí solo, para comprimirlo hay que hacerlo con gzip/bzip2. Tener en cuenta que nos referimos a archivos en el sentido de paquetes de archivos**

Sintaxis:

cpio –o [options] < [filenames …] > [archive]
cpio –i < [archive]
cpio –p [destination-directory] < [filenames…]

-o Copy-out mode. Este modo se usa para crear un archivo.

-i Copy-in mode. Este modo se usa para copiar todo que creamos antes.

-p Copy-pass mode. No crea un archivo , solo lo copia archivos de un lado a otro.

Ejemplos:

Para archivar:

```
# ls |cpio -ov > /media/extdisk/muestra.cpio
athelstan
rino
rino1
rino2
rino3
1 block
```

📌 **El significado del signo '>' significa que volcamos la salida del comando ubicado a la izquierda en el archivo de la derecha.**

Para extraer:

```
# cpio  -iv < /media/extdisk/muestra.cpio
athelstan
rino
rino1
rino2
rino3
1 block
```

📌 **El significado del signo '<' significa que el comando de la izquierda recibe como argumentos el archivo de la derecha.**

```
# ls -la
```

```
total 12
drwxr-xr-x  3 root root 4096 Sep  1 21:11 .
drwxr-xr-x 11 rino rino 4096 Sep  1 21:10 ..
drwxr-xr-x  2 root root 4096 Sep  1 21:11 athelstan
-rw-r--r--  1 root root    0 Sep  1 21:11 rino
-rw-r--r--  1 root root    0 Sep  1 21:11 rino1
-rw-r--r--  1 root root    0 Sep  1 21:11 rino2
-rw-r--r--  1 root root    0 Sep  1 21:11 rino3
```

Otras opciones importantes del comando cpio son:
-u Sobreescribe archivos existentes al extraer
-m Preserva los permisos
-d Como cpio extrae de manera 'plana' los archivos, con esta opción respeta la estructura original de directorios del paquete.

La ventaja de este comando con respecto al tar es que se puede tener mayor control, cuando realizamos scripts, sobre los archivos o tipos de archivos que no se desean copiar. Suele combinarse con el comando find. No obstante, su uso es menos frecuente que el de tar ya que este último es mucho más sencillo de usar.

Comando dd

Este comando es muy útil para realizar copias de seguridad/clonado de dispositivos de almacenamiento porque copia en bloques.

Sintaxis dd [options]

if=archivo	toma como entrada un archivo o dispositivo.
of=archivo	salida a un archivo en vez de la salida estándar.
ibs=n	leer de a "n" bytes.
obs=n	escribir de a n bytes
conv=list	convertir de ciertos aspecto de lo que copio.

Ejemplos

Crear una imagen de un CD/DVD

```
# dd if=/dev/cdrom of=/tmp/cd.iso
```

Crear una imagen de un disco rígido.

```
# dd if=/dev/sda of=/direnotrodisco/disk1.img
```

Crear una imagen de una partición

```
# dd if=/dev/sda1 of=/direnotraparticion/disk2.img
```

Recuperar el contenido de un disco

```
# dd if=disk1.img of=/dev/sda
```

Recuperar el contenido de una partición

```
# dd if=disk2.img of=/dev/sda1
```

Copiar el archivo file a file2 convirtiendo todo su contenido a minúsculas.

```
# dd if=/tmp/file of=/tmp/file2 conv=lcase
```

Copiar una determinada cantidad de bytes de un archivo:

```
# dd if=/dev/zero of=/root/bigfile bs=100M count=10
```

En este último caso se crea un archivo de 1000M con bytes nulos.

Comando file

Este comando nos va a devolver qué tipo de archivo es el que el le preguntemos, sin importar su extensión, lo detecta por su contenido.

Sintaxis file [options] [file]

Opciones

-f busca por cada línea que hay en el archivo que se le invoque

-z intenta ver adentro de los archivos comprimidos

```
# cat rino.txt
/etc/passwd
/etc/shadow
```

Vemos que anteriormente mostramos el contenido del archivo en donde esta el path de dos archivos, luego con la opción -f examina cada línea.

```
# file -f rino
/etc/passwd: ASCII text
/etc/shadow: regular file, no read permission
```

```
# file symvers-2.6.18-164.el5.gz
symvers-2.6.18-164.el5.gz: gzip compressed data, from Unix, last modified: Tue Aug 18 16:55:52 2009, max compression
```

Comando find

El comando **find** es una herramienta muy potente que permite buscar entre otras cosas:

- En un determinado directorio en la jerarquía del sistema de archivos.
- Archivos por nombre
- Archivos por tipo
- Archivos por marcas de tiempo
- Archivos por permisos
- Archivos de acuerdo a su dueño

Sintaxis

find [options] [path…] [expression]

-mount no va a seguir buscando recursivamente por directorios que fueron montados.

-maxdepth X le indico cual es la cantidad máxima de subdirectorios que quiero que recorra

-perm para buscar por permisos (-perm 777)

Buscar por nombre

```
#  find /home/rino/scripts/ -name “*.sh”
/home/rino/scripts/vpn.sh
/home/rino/scripts/irssi_notify.sh
/home/rino/scripts/rino.sh
```

📌 Es conveniente encerrar entre comillas para que la shell no reemplace automáticamente los metacaracteres en el directorio actual.

Buscar aquellos archivos que se hayan creado hace más de siete días, y máximo un subdirectorio de profundidad

```
# find /tmp -maxdepth 1 -type f -daystart -ctime +7
/tmp/soDumpciZ4eX
/tmp/sakis3g.dialog.6712
/tmp/salida.out
/tmp/sakis3gz.6677.sakis3g
```

Buscar archivos .tmp sin importar mayúsculas o minúsculas y ejecutar una acción sobre los archivos encontrados

```
# find /home -iname "*.tmp" -exec rm -i '{}' \;
```

La cadena de texto '{}' es reemplazada por el archivo procesado por find, y el \; significa que no hay más argumentos para el comando después del exec (en este caso rm). Tanto las comillas simples como la contrabarra se ponen para que bash interprete de manera literal esos caracteres.

Buscar por tamaño:

```
# find /usr -size +110M
/usr/lib/jvm/java-10-openjdk-10.0.2.13-1.fc28.x86_64/lib/modules
/usr/lib/locale/locale-archive
/usr/share/windows95/resources/app.asar.unpacked/src/images/windows95.img
/usr/local/share/.config/yarn/global/node_modules/puppeteer/.local-chromium/linux-579032/chrome-linux/chrome
```

Buscará los archivos con un tamaño mayor a 110 MB.

Más ejemplos:

Buscar por nombre de archivo	find /etc -name "*.sh"	Usar comillas al usar metacaracteres para evitar que la shell realice autocompletado de rutas
Buscar por tipo y nombre de archivo	find / -type d -name images	
Buscar por fecha de modificación	find /var -mtime -2	Buscará todos los archivos que fueron modificados hace menos de 2x24hs exactamente
	find /var -mtime +2	Buscará todos los archivos que fueron modificados hace más de 2x24hs exactamente
	find /var -mtime 2	Buscará todos los archivos que fueron modificados hace 2x24hs exactamente

Manuales

Las páginas del manual o páginas **man** de GNU/Linux son el mejor lugar para resolver cuestiones relativas a la sintaxis y opciones de los comandos y utilidades del sistema. Los documentos de las páginas **man** están almacenados en un formato comprimido. El comando **man** descomprime y da formato a estas páginas para su correcta visualización. Se accede a estas páginas utilizando el comando **man** seguido del nombre del comando que se quiere consultar.

Un ejemplo de la sintaxis te este comando es el siguiente:

$ man ls

Este comando buscará todas las páginas del manual relativas al comando ls. Cuando abras las páginas del manual, lo primero que se

visualizará es un banner con el comando y la página man que está siendo consultada. También se muestra aquí el logo FSF de Free Software Foundation.

Ejemplo:

LS(1) FSF LS(1)

Esto irá seguido del nombre del comando y de su función.

NAME

ls – list directory contents

A continuación, se muestra la sintaxis del comando:

SYNOPSIS

ls [OPTION]… [FILE]…

A esto le sigue una descripción del comando. Tras la descripción se muestran y explican las opciones del mismo.

DESCRIPTION

List information about the FILEs (the current directory by default).
Sort entries alphabetically if none of -cftuSUX nor –sort.
Mandatory arguments to long options are mandatory for short options
too.
-a, –all
do not hide entries starting with.
-A, –almost-all
do not list implied . and ..
-b, –escape
print octal escapes for nongraphic characters
–block-size=SIZE
use SIZE-byte blocks
-B, –ignore-backups
do not list implied entries ending with ~

La página man termina con información relativa al autor de la página, bugs conocidos e información sobre como reportar nuevos bugs, copyright, e instrucciones para obtener más información sobre el comando.

AUTHOR

Written by Richard Stallman and David MacKenzie.
REPORTING BUGS
Report bugs to .
COPYRIGHT
Copyright © 1999 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR
A PARTICULAR PURPOSE.
SEE ALSO
The full documentation for ls is maintained as a Texinfo manual.
If the info and ls programs are properly installed at your site,
the command
info ls
should give you access to the complete manual.

GNU fileutils 4.0p March 2000 1

Para avanzar páginas en el **man**, se utiliza la barra espaciadora, que avanzará una página por cada pulsación. La tecla "**q**" provoca la salida del **man**. Si queremos buscar algún texto dentro de la página man, puedes utilizar las expresiones regulares. a continuación veremos un ejemplo de cómo buscar la palabra "option".

/option

Encontrando las páginas man

Las páginas **man** se almacenan en el sistema. La variable **MANPATH** indica la ubicación de estos archivos. Las páginas **man** se guardan en los siguientes lugares.

/usr/share/man/man1
/usr/share/man/man2
/usr/share/man/man3
/usr/share/man/man4
/usr/share/man/man5
/usr/share/man/man6
/usr/share/man/man7
/usr/share/man/man8
/usr/share/man/man9

El usuario puede especificar un **MANPATH** diferente. Esto permitiría utilizar un conjunto diferente de páginas man. Esto es práctico porque algunos comandos podrían almacenar sus páginas man en lugares distintos a los estándar.

Opción	Descripción
-C configuración	Indica un fichero de configuración distinto a /etc/man.conf.
-M ruta	Indica en qué directorios se buscarán las páginas man.
-P paginador	Indica el paginador (Programa que da formato y visualiza las páginas). Por defecto es el indicado en la variable de entorno **PAGER**. Los paginadores **more** y **less** son los más utilizados.
-S lista	Indica una lista de las secciones a buscar separadas por dos puntos (**:**)
-a	Indica que han de mostrarse todas las entradas coincidentes y no solo la primera.
-c	Indica que la página fuente ha de ser reformateada.
-d	Mostará información de debug en lugar de las páginas man.
-f	Indica que el programa man debe comportarse como el programa **whatis** (se explicará mas adelante).
-h	Muestra información sobre el comando man.
-k	Indica que el programa man debe comportarse como el programa apropos (se explicará mas adelante).
-K	Busca una cadena especificada en las páginas man. Por cada entrada encontrada se le pregunta al usuario si desea verla
-m	Indica un conjunto alternativo de páginas man basado en el sistema especificado.
-w	Indica que ha de visualizarse el path de las páginas man

A continuación se muestra un ejemplo del uso de la opción -a. Ésta hace que las páginas coincidentes sean mostradas en el orden en el que han sido encontradas. En primer lugar, se le muestra al usuario la entrada correspondiente a crontab en la sección uno. Cuando el usuario pulsa la tecla Q para salir de ésta página , se mostrará la entrada encontrada en la sección cinco.

 # man –a crontab

La opción -w es práctica para encontrar la ubicación de las entradas de las páginas man. Si utilizásemos esta opción con la utilidad crontab obtendríamos lo siguiente:

 # man -w crontab
/usr/man/man1/crontab.1.gz

Pregunta de Examen: Asegurate que conoces las opciones de búsqueda y sus funciones, entre ellas -a, -K, y -k.

Buscando secciones de las páginas man

La información de las página man de Linux están contenidas en un conjunto de archivos. Estos archivos están agrupados en secciones y cada sección contiene un tipo específico de información. La Tabla 1-4 lista éstas secciones y su uso:

Secciones de las páginas man

Sección	Descripción
1	Comandos y aplicaciones del usuario.
2	Llamadas del sistema y errores del Kernel.
3	Llamadas a librerías.
4	Drivers de dispositivos y protocolos de red.
5	Formatos estándar de archivos.
6	Juegos y demos.
7	Ficheros y documentos misceláneos.
8	Comandos de administración del sistema.
9	Especificaciones e interfaces oscuros del kernel.

Cuando se le pasa un argumento al comando man, éste busca dentro de las secciones siguiendo un orden específico y se devuelve la primera coincidencia. El orden de búsqueda por defecto es el siguiente:

1, 8, 2, 3, 4, 5, 6, 7, 9.

También es posible indicar la sección en la que queremos buscar. Si quisiéramos buscar información sobre la utilidad crontab en la sección cinco utilizarías el siguiente comando:

 # man 5 crontab

Si usamos la opción **-a** podremos examinar todas las entradas coincidentes. Siguiendo el ejemplo de la utilidad crontab, tendríamos que utilizar el siguiente comando:

 # man -a crontab

whatis

La utilidad **whatis** no se menciona específicamente en los objetivos del examen, pero conocer su uso puede venir bien en el examen. La utilidad **whatis** se usa para buscar entradas coincidentes en la base de datos **whatis**. Esta base de datos se crea utilizando el comando **/usr/bin/makewhatis.** Esta base de datos contiene las descripciones cortas que se encuentran en las páginas **man** de los comandos del sistema. Un ejemplo de su uso es el siguiente:

 # whatis passwd
passwd (1) – update a user's authentication tokens(s)
passwd (1ssl) – compute password hashes
passwd (5) – password file
passwd.nntp [passwd] (5) – passwords for connecting to remote NNTP servers

Como podemos ver en este ejemplo, el comando passwd tiene entradas en las secciones uno y cinco de las páginas man. También ha sido encontrado en la sección uno de las páginas man del comando ssl.

El comando **man -f** busca en esta base de datos entradas coincidentes con la palabra clave indicada. A continuación, tenemos un ejemplo de la salida producida por este comando.

 # man -f passwd
passwd (1) – update a user's authentication tokens(s)
passwd (1ssl) – compute password hashes
passwd (5) – password file
passwd.nntp [passwd] (5) – passwords for connecting to remote NNTP servers

Estos comandos realizan la misma búsqueda. Se muestran los comandos y las páginas man donde han sido encontrados. Esto puede ser práctico para localizar secciones de páginas man y variantes de los comandos.

apropos

Al igual que whatis, el comando apropos no se menciona específicamente en los objetivos del examen, pero conocer su uso puede venir bien en el examen. Y también como la utilidad whatis, el comando apropos utiliza la base de datos whatis. Este comando se emplea para buscar tanto los nombres de comando como las descripciones para la palabra clave indicada. A continuación vemos un ejemplo del comando apropos:

 # apropos password
chpasswd (8) – update password file in batch
gpasswd (1) – administer the /etc/group file
htpasswd (1) – Create and update user authentication files
nwpasswd (1) – Change a user's password
…(Salida del comando recortada)

A continuación tenemos un ejemplo del comando man -k :

 # man -k password
chpasswd (8) – update password file in batch
gpasswd (1) – administer the /etc/group file
htpasswd (1) – Create and update user authentication files
nwpasswd (1) – Change a user's password
passwd (1) – update a user's authentication tokens(s)
passwd (1ssl) – compute password hashes
passwd (5) – password file
passwd.nntp [passwd] (5) – passwords for connecting to remote NNTP servers
…(Salida del comando recortada)

Como podemos ver, estos comandos hacen lo mismo. Esto puede ser práctico cuando buscamos comandos a partir de determinadas palabras clave.

Documentos Info

<PARAHACER>

Más recursos

Libros:

- [CompTIA Linux+ / LPIC-1 Cert Guide: (Exams LX0-103 & LX0-104/101-400 & 102-400)](https://www.lpimarketplace.com/CompTIA-Linux-LPIC-1-Cert-Guiden-p/978-0-78-975455-4.htm)
- [CompTIA Linux+ /LPIC-1 Certification All-In-One Exam Guide, Premium Second Edition With Online Practice Labs (Exams LX0-103 & LX0-104/101-400 & 102-400)](https://www.lpimarketplace.com/CompTIA-Linux-LPIC-1-All-In-One-Exam-Guide-p/978-1-259-86369-1.htm)
- [LPIC-1 Certification All-in-One Exam Guide Exams LPIC-1 book | LPI Marketplace](https://www.lpimarketplace.com/LPIC-1-Certification-All-in-One-Exam-Guide-p/978-0-071-84168-9.htm)

Paginas:

- [COMANDOS LINUX PARA LA GESTION DE ARCHIVOS Y DIRECTORIOS [WIKI. IES Haría. Informática]](https://smr.iesharia.org/wiki/doku.php/ssv:ut3:c_linux_archivos)
- [Secretos del comando man - ForoSUSE](http://www.forosuse.org/forosuse/showthread.php?t=13975&langid=1)
-
- [Beginner’s Bash](https://linux.org.mt/article/terminal/)
- [Linux Guide/merge/Linux For Newbies/Command Line - Wikibooks, open books for an open world](https://en.wikibooks.org/wiki/Linux_Guide/merge/Linux_For_Newbies/Command_Line)

BÚSQUEDA Y UBICACIÓN DE ARCHIVOS

📌 Información sobre LPI

Tema Cubierto

103.3 Buscar archivos del sistema y ponerlos en la ubicación correcta

Peso

2 (El peso indica la cantidad de preguntas relativas al tema en cuestión, es decir cuánto más grande el número, es más su incidencia en el resultado del examen. Su valor puede ir de 1 a 6)

Descripción

Los alumnos deberían de estar familiarizado con la Jerarquía del Sistema de archivos Estándar clas(en inglés, FHS) incluyendo la clasificación ubicaciones de los archivos comunes y la clasificación de los directorios.

Áreas Claves de Conocimiento

- **Terms and Utilities:**

- `find`
- `locate`
- `updatedb`
- `whereis`
- `which`
- `type`
- `/etc/updatedb.conf`

¿Qué es FHS?

En este tópico vamos a desarrollar el tema de la estructura de archivos del sistema y algunos comando básicos para poder encontrar archivos.

Antes de empezar a explicar, hay que introducir el término de FHS (Filesystem Hierarchy standard –>http://refspecs.linuxfoundation.org/fhs.shtml), un estándar a seguir para las distribuciones de Linux, que se refiere a la forma en que se utiliza el sistema de archivos en Linux.
Vamos a mencionar algunas partes de FHS, pero para mas información, pueden leer el link mencionado anteriormente.

Estructura Básica según FHS

Hay dos grupos independientes de archivos, shareables/unshareables y variables/static

Shareable: Los datos pueden ser usados por múltiples sistemas en una red. Los archivos que se pueden compartir y que son información de propósito general, en donde existe ninguna atadura en ningún host. Un ejemplo de estos archivos pueden ser archivos de datos, archivos de programas ejecutables y documentación del sistema.

Non-Shareable: Aquella información que no se puede compartir, dado que está atada a un equipo en particular, como por ejemplo, archivos de configuración.

Variable: Los datos son considerados variables cuando estos cambian sin intervención del usuario/root. Ejemplo de estos pueden ser archivos de usuarios, registros del sistema (logs). Ejemplo /var/log/syslog, /var/log/messages.

Static: Son aquellos archivos que no interactúan con nadie y permanecen siempre iguales durante el día a día o años enteros. Un ejemplo de estos pueden ser los binarios de determinados comandos (ls,bash) en donde solo cambian cuando se actualiza el sistema.

Algunos directorios en el sistema de ficheros de Linux están destinados a contener los tipos específicos de datos. Por ejemplo, los archivos ejecutables en **/usr** ,rara cambian sin la intervención del administrador, por lo que pueden definirse como algo estático, ya que son necesarios para todos los usuarios en una red. Antes de que los discos fueran tan grandes como lo son hoy, los archivos que se encontraban comúnmente en **/usr** se montaban a menudo desde servidores remotos para conservar espacio en el disco local. Así, además de ser estático, **/usr** se dice que es compartible.
Mantener los archivos organizados con respecto a estos atributos puede simplificar el intercambio de archivos, administración de sistemas, y la complejidad de copia de seguridad, así como reducir los requerimientos de almacenamiento.
El FHS organiza las categorías de datos anteriores en una matriz de 2 × 2.

	Shareable	Unshareable

static	/usr /opt	/etc /opt
variable	/var/mail /var/cache/fonts /home	/var/run /var/log

En muchas redes, **/usr** y **/usr/local** se montan por las estaciones de trabajo individuales de un servidor **NFS**. Esto puede ahorrar una cantidad considerable de almacenamiento local en las estaciones de trabajo. Más importante aún, la colocación de estos directorios en otro sistema puede hacer que las actualizaciones y upgrades sean mucho más simples. Estos directorios suelen ser compartidos como sistemas de archivos de solo lectura porque no son modificados por la mayoría de usuarios finales.
El directorio **/var/mail** y directorios **/home**, por el contrario, se comparten pero son cambiados regularmente por los usuarios. El directorio **/etc** y **/boot** contienen archivos que son estáticos en el sentido que sólo el administrador puede cambiarlo, pero el uso compartido de ellos no es necesario o aconsejado, porque son archivos de configuración local. El directorio **/var/log** y **/proc** son directorios muy dinámicos, pero también son solo de interés local.

Directorios Esenciales

El sistema de archivos raíz /

El FHS ofrece un significativo nivel de detalle que describe la localización exacta de los archivos, utilizando la lógica derivada de la definición static/variable y shareable/unshareable. Esta sección trata sobre las partes más importantes de la jerarquía de directorios FHS.
El sistema de archivos raíz se encuentra en la parte superior de la jerarquía de directorios. El FHS define los objetivos para el sistema de archivos raíz:

- Arrancar el sistema, deben estar el software y los datos suficiente en la partición raíz para montar otros sistemas de archivos. Esto incluye herramientas, configuración, información del cargador de arranque, y otros datos del arranque. Los directorios /usr, /opt y /var están diseñados de modo tal poder ubicarse en otras particiones o sistemas de archivos.
- Posibilitar la recuperación y/o reparación de un sistema, aquellas herramientas necesarias para que un administrador experimentado diagnostique y reconstruya un sistema dañado.
- Restaurar un sistema, aquellas herramientas necesarias para recuperar backups.
- Las aplicaciones ni la distribución deberían crear archivos o subdirectorios adicionales en el directorio raíz
- Su partición y debería ser lo más chico posible, siempre que permite realizar las funciones mencionadas arriba mencionadas.

Aunque un sistema Linux con todo en una sola partición raíz se puede crear, el hacerlo así no cumpliría con estos objetivos. En cambio, el sistema de archivos raíz debe contener sólo los esenciales directorios del sistema, junto con los puntos de montaje para otros sistemas de archivos.

Directorios No Esenciales

El resto de directorios de nivel superior en el sistema de archivos raíz se consideran no esenciales para los procedimientos de emergencia:

- /boot
- /home
- /opt
- /tmp
- /usr
- /var

El sistema de archivos /usr

El sistema de archivos **/usr** contiene utilidades de sistema y programas que no pueden no estar en la partición raíz. Por ejemplo, los programas de usuario, tales como **less** y la **tail** se encuentran en **/usr/bin.** El directorio **/usr/sbin** contiene comandos de administración del sistema, tales como **adduser** y **traceroute**, y una serie de demonios sólo necesarios cuando el sistema funciona

normalmente. No hay datos específicos del host o datos variables que se almacenen en **/usr**. También es rechazada la colocación de directorios directamente en **/usr** para los paquetes de software grandes. Una excepción a esta regla está hecha para X11.

Los siguientes subdirectorios se deben encontrarse en /usr:

- /usr/bin
- /usr/include
- /usr/lib
- /usr/local
- /usr/sbin
- /usr/share

El sistema de archivos /var: El sistema de archivos **/var** contiene datos tales como, el de la cola de impresión y los archivos de registro de bitácoras que varían con el tiempo. Puesto que los datos variables siempre están cambiando y creciendo, **/var** es por lo general contenida en una partición separada para evitar el llenado de la partición raíz.

Los siguientes sub-directorios se requieren en /var:

1. /var/cache
2. /var/lib
3. /var/local
4. /var/lock
5. /var/log
6. /var/opt
7. /var/run
8. /var/spool
9. /var/tmp

Binarios

Los directorios **bin** y **sbin**, se llaman así porque poseen archivos ejecutables, los cuales la mayoría están ya compilados como binarios.
En esta tabla, se enumeran estos directorios y muestra cómo se usa cada uno.

	Comandos de usuario	Comandos de administración del sistema
Suministrados por la distribución (sistema de archivos raíz)	/bin	/sbin
Suministrados por la distribución (sistema de archivos /usr)	/usr/bin	/usr/sbin
Suministrados localmente (por ejemplo, compilados a mano), no son esenciales (sistema de archivos /usr/local)	/usr/local/bin	/usr/local/sbin

Herramientas de Búsqueda

Ubicando los archivos:
FHS ofrece a la comunidad Linux un excelente recurso que asegura la coherencia entre las distintas distribuciones y otros sistemas operativos. En la práctica, sin embargo, la ubicación del archivo puede ser frustrante, y surge la necesidad de buscar archivos en el sistema rápidamente.
Estas herramientas de localización de archivos son necesarias para el examen 101: **which, find, locate, whereis, y type.**

- **which** utiliza la variable PATH para localizar los archivos ejecutables.
- **find** realiza búsquedas en áreas especificadas en el sistema de archivos.
- **whereis** busca en un pequeño subconjunto de directorios comunes.
- **locate** ofrece una alternativa rápida para encontrar por nombre de archivo y es adecuado para la localización de los archivos que no se mueven en el sistema de archivos. Genera una base de datos, la cual se actualiza periódicamente. Si se busca un archivo creado antes de la actualización de la base de datos, este no aparecerá. Se puede forzar la actualización con el comando **updatedb.** El comportamiento de updatedb se puede adaptar editando el archivo **/etc/updatedb.conf**.

Locate

Sintaxis locate patrón

La búsqueda se realiza en un índice, creado con el comando **updatedb**, busca los archivos cuyos nombres coincidan con uno o más patrones.
Buscar el patrón sh:

```
# locate zyxel
/usr/share/wireshark/radius/dictionary.zyxel
```

El comando **locate** debe tener una base de datos reciente para la búsqueda, y la base de datos debe ser actualizada periódicamente para incorporar los cambios en el sistema de archivos. Si la base de datos está demasiado desactualizada, el comando puede advertirnos como se muestra a continuación:

 # locate tcsh
locate: warning: database /var/lib/slocate/slocate.db' is more than 8 days old

Para actualizarla hay que ejecutar el comando **updatedb**

Updatedb

Sintaxis updatedb [opciones]

Actualizar (o crear) la base de datos slocate en /var/lib/slocate/slocate.db.

Ejemplo para actualizar sólo lo contenido en un grupo de directorios

```
# updatedb -e ”/dev /proc /home”
```

Hay parámetros especiales, donde podemos hacer que realice la búsqueda por red o que no busque en determinados sistemas de archivos.

```
# cat /etc/updatedb.conf
PRUNEFS = "auto afs gfs gfs2 iso9660 sfs udf"
PRUNEPATHS = "/afs /media /net /sfs /tmp /udev /var/spool/cups /var/spool/squid /var/tmp"
```

Type

Sintaxis

type [options] nombre de archivo

Type no es realmente un programa independiente, sino una parte integrada de la shell bash. **Type** le dirá cómo un nombre de archivo se interpretaría si se usa como un nombre de comando.

Ejemplo:

```
# type ls
ls is aliased to `ls --color=tty'

# type grep
grep is /bin/grep
```

Whereis

Sintaxis

whereis [options] nombre de archivo

whereis localiza source/binarios y manuales para los archivos especificados.

Un ejemplo:

```
# whereis ls
ls: /bin/ls /usr/share/man/man1p/ls.1p.gz /usr/share/man/man1/ls.1.gz
```

Which

Sintaxis

which comando

Determina la ubicación del comando y muestra la ruta completa del programa ejecutable que la shell lanzaría para ejecutarlo. Este sólo busca la rutas definidas en la variable PATH.

```
# which bash
/bin/bash
```

Más recursos

Libros:

- Página del manual de **FHS**, se accede con `man 7 hier`
- [CompTIA Linux+ / LPIC-1 Cert Guide: (Exams LX0-103 & LX0-104/101-400 & 102-400)](https://www.lpimarketplace.com/CompTIA-Linux-LPIC-1-Cert-Guiden-p/978-0-78-975455-4.htm)
- [CompTIA Linux+ /LPIC-1 Certification All-In-One Exam Guide, Premium Second Edition With Online Practice Labs (Exams LX0-103 & LX0-104/101-400 &

102-400)](https://www.lpimarketplace.com/CompTIA-Linux-LPIC-1-All-In-One-Exam-Guide-p/978-1-259-86369-1.htm)
- [LPIC-1 Certification All-in-One Exam Guide Exams LPIC-1 book | LPI Marketplace](https://www.lpimarketplace.com/LPIC-1-Certification-All-in-One-Exam-Guide-p/978-0-071-84168-9.htm)

Páginas:

- [FHS Referenced Specifications](https://refspecs.linuxfoundation.org/fhs.shtml)
- [El estándar FHS | Tutorial de GNU/Linux](http://fpg.site11.com/GNU-Linux/el_estndar_fhs.html)
- [FilesystemHierarchyStandard - Debian Wiki](https://wiki.debian.org/FilesystemHierarchyStandard)