Preparación para el examen LPI 101

Tema 103.1

Trabajando en la linea de comandos

Créditos y licencia de uso

Coordinación:
Manuel Guillán (xLekOx) lpi@xlekox.org

Traducción:
Juan Maria Gil (Smooth) yo@juanmaria.com
Manuel Guillán (xLekOx) lpi@xlekox.org
Carmen Eugenio (nemrac) meneiro@ono.com
Pablo Taboada (java) ptaboada@wanadoo.es

Maquetación:
Manuel Guillán (xLekOx) lpi@xlekox.org
Pablo Taboada (java) ptaboada@wanadoo.es

ÍNDICE

Índice de contenido

Tema 103.1

Introducción

En este primer tema se aprenderá el uso básico de la linea de comandos (shell). Usar comandos de ayuda, su formato, uso de diferentes parámetros, personalizar el entorno y usar el historial de comandos.

Los comandos que se verán en este tema son:

Archivos ocultos (empiezan por un .)
bash
echo
env
exec
export
man
pwd
set
unset
~/.bash_history
~/.profile

Así mismo se harán ejercicios sobre los mismos al final del tema, que serán muy parecidos a los realizados en los exámenes.

Este tema tiene un peso (importancia) de 5 de cara al examen final de la certificación LPI 101. El total de la suma de pesos de todos los temas es de 106.

Usando la Línea de Comandos

OBJETIVO: Trabajar de forma eficaz con la línea de comandos. Incluye la escritura de comandos válidos y las secuencias de comandos, usando substitución, y aplicando comandos recursivos a través de un árbol de directorios.

Una vez que te has logeado dentro del sistema Linux, te enfrentas con el shell. Éste aparece simplemente como una interface de línea de comando. A partir del prompt del shell escribimos comandos que son interpretados por el shell y enviados al sistema. El shell que estés corriendo en ese momento configura su prompt correspondiente. La mayoría de los sistemas Linux tienen como predeterminado el shell bash, y generalmente está configurado para mostrar el nombre de usuario, nombre del servidor y directorio actual de trabajo en el prompt. Un ejemplo de este prompt sería:

```
[angie@redhat /etc]$
```

En este ejemplo **angie** es el nombre de usuario, **redhat** es el nombre del servidor y **/etc** es el directorio actual de trabajo (llamado pwd – present working directory). El prompt para la shell de bash es el símbolo **$**. El prompt se configura a través del archivo /etc/bashrc. Por medio de este archivo se puede cambiar la configuración de lo que se muestra en el prompt. Cada usuario puede personalizar el prompt a su gusto, creando para ello el archivo de configuración personalizado en ~/.bashrc

Los comandos escritos en la línea de comandos no son enviados al sistema hasta que no se haya pulsado la tecla Enter. Esto permite editar los comandos antes de que se pasen al sistema. Los comandos escritos en la línea de comandos deben seguir una sintaxis específica. La primera palabra es el comando que se debe ejecutar; esta puede incluir una ruta absoluta, que es la ruta completa al comando, que debe terminar con el nombre del comando. A continuación deben figurar las opciones que son usadas por el comando. Los argumentos del comando van a continuación de las opciones. Cada uno de estos elementos se deben separar en la línea de comandos por medio de un espacio en blanco. El formato de un comando es el siguiente:

```
$ comando opciones argumentos
```

Recordatorio de examen: Es importante recordar esta sintaxis. Algunas preguntas del examen cuestionan sobre el correcto uso de comandos, opciones y argumentos.

Por ejemplo, tomando el comando ls. Este comando lista los archivos y directorios que hay en el pwd. Para ejecutar este comando, simplemente hay que teclear:

```
$ ls
```

Las opciones son códigos de una letra precedidos por un guión (-), y modifican la acción del comando. Se pueden combinar y usar varias opciones con un mismo comando. Las opciones son sensibles a mayúsculas y, en muchas ocasiones, una letra minúscula significará una opción diferente que su correspondiente mayúscula. Una opción muy importante disponible con muchos comandos es la -R, que especifica que el comando se debe ejecutar de forma recursiva a través del árbol de directorios.

Recordatorio de examen: La opción para la función recursiva de un comando es muy importante y aparecerá en el examen.

Si lo que quieres es listar los archivos y los directorios, y que los directorios aparezcan seguidos de una barra inclinada (/), se puede usar la opción -F (prueba el resultado).

```
$ ls -F
```

Ahora suponed que deseamos mostrar el contenido del directorio /etc desde otra localización. Además queremos que se muestre una barra inclinada tras todos los nombres de directorios que estén dentro de /etc. El directorio /etc se puede utilizar como argumento para el comando ls. Un argumento es otra opción que se puede utilizar con un comando. En el siguiente ejemplo, el argumento es un directorio que será examinado por el comando. Sería algo así:

```
$ ls -F /etc
```

Otro carácter especial que se puede utilizar cuando se introducen comandos es la barra invertida (\). Cuando se introduce a la derecha, antes de pulsar la tecla Enter, la barra invertida permite extender el comando a lo largo de varias líneas. La barra invertida provoca que el sistema ignore la pulsación de la tecla Enter, y trata todos los comandos como si estuvieran en una única línea. Esta función puede ser útil si tecleamos comandos muy largos, para poder partirlos en varias líneas. Un ejemplo sería el siguiente:

```
$ ls -F /etc \
ls -F /usr
```

En el mundo real: Para ejecutar comandos muy largos se utilizan scripts, en lugar de teclearlos directamente sobre la línea de comandos. Sin embargo, es importante recordar el uso de la barra invertida para el examen.

Otras opciones que se pueden usar con el comando ls pueden ser:

```
$ ls -l
```

Devuelve la lista de archivos y directorios en formato extenso, incluyendo nombre, privilegios de acceso, propietario, tamaño, fecha de última modificación, etc.

```
$ ls -a
```

Muestra todos los archivos y directorios del pwd, incluyendo los archivos ocultos.

```
$ ls *.gif
```

Muestra todos los archivos del pwd que terminen por “.gif”.

```
$ ls ga*
```

Muestra todos los archivos y directorios del pwd que comiencen por “ga”.

Completando Comandos

El shell bash incluye una características denominada “completado de comandos”. Esto nos permite teclear las primeras letras de un comando, pulsar la tecla tabulador, y dejar que el sistema complete el comando por nosotros. Si queremos ejecutar el comando dmesg para mostrar el buffer del kernel, podríamos teclear:

```
$ dm
```

y pulsar la tecla Tabulador, de forma que el sistema completará el comando:

```
$ dmesg
```

Si existe más de una coincidencia con la cadena tecleada antes de pulsar la tecla Tabulador, el sistema hará sonar un beep. Pulsando de nuevo la tecla Tabulador se mostrarán todas las posibilidades que coincidan con lo tecleado. Pulsar la tecla Esc dos veces produce el mismo efecto que pulsar la tecla Tabulador.

Conectando varios Comandos

En todos los ejemplos utilizados hasta ahora utilizamos la tecla Enter para informar al sistema de que el comando debía ser procesado. Sin embargo, no estamos limitados a ejecutar un único comando de cada vez.

Podemos ejecutar varios comandos, sin que estén conectados entre sí de ningún modo, tecleándolos en la misma línea y separándolos por punto y coma (;). Por ejemplo, es posible listar todos los archivos del directorio actual y la fecha de hoy tecleando:

```
$ ls ; date
```

El punto y coma es un carácter especial que siempre significa que hay varios comandos en la misma línea. Debido a que esto último tiene un sentido global, podemos prescindir de los espacios en blanco a ambos lados del punto y coma para obtener el mismo resultado (ls;date).

Si lo que hacen los comandos tiene algo en común – la salida de uno de ellos se convertirá en la entrada del siguiente – entonces podemos conectarlos usando una tubería (|). Por ejemplo, si la lista de de archivos de un directorio es muy larga como para poder verla en una sola pantalla, podemos ver una pantalla de cada vez usando:

```
$ ls -l | more
```

De este modo, la salida del comando ls -l será la entrada del comando more. Si falla la primera parte de la línea de comando, no se podrá ejecutar la segunda.

Comodines

Los comodines son caracteres que se utilizan en lugar de otros caracteres que el sistema rellena. Los dos comodines más frecuentes son el asterisco (*) y la interrogación (?). Aunque en ocasiones se confundan, su significado es diferente y producirán resultados totalmente distintos.

El asterisco significa ninguno, alguno o todos los caracteres:

```
$ ls s*
```

Este comando mostrará todas las entradas (archivos o directorios) dentro del directorio actual que comiencen con la letra s, y que tengan cualquier número de caracteres a continuación (incluyendo ninguno). Un posible resultado del comando puede ser:

```
s sa sam samp sampl sample samples samples.gif
```

Hay que prestar atención a que el comando encuentra la “s” sola y la “s” seguida de cualquier número de caracteres a continuación. En contraste, la interrogación (?) es un contenedor para un y sólo un carácter. Utilizando las mismas posibilidades que antes, el comando:

```
$ ls s?
```

Encontrará entradas (archivos y directorios) dentro del directorio actual que comiencen por la letra “s” y que únicamente tengan una letra más. El resultado sería:

```
sa
```

Si quisiésemos encontrar las entradas que comiencen por s y cuyo nombre tenga 5 caracteres en total, utilizaríamos:

```
$ ls s????
```

En resumen, el asterisco significa todos o ninguno, y el interrogante siempre significa uno. Estos dos comodines no son excluyentes, de modo que se pueden combinar según las necesidades. Por ejemplo, para encontrar sólo los archivos que tengan una extensión de tres letras dentro del directorio actual, utilizaremos:

```
$ ls *.???
```

Para complicar un poco más las cosas también podemos utilizar los corchetes ([]) para especificar posibles valores. Todos los valores posibles deben estar dentro de los corchetes, y el shell los tratará individualmente:

```
$ ls [de]*
```

Este ejemplo encontrará todas las entradas que comiencen por “d” o por “e” y que contengan un número ilimitado de caracteres. Para encontrar las entradas de longitud de 3 caracteres que comiencen por “d” o por “e”, utilizaremos:

```
$ ls [de]??
```

El número de caracteres que podemos incluir dentro de los corchetes es teóricamente ilimitado. Sin embargo, si lo que queremos es encontrar todas las entradas que comiencen por una letra minúscula pero no por un número u otro carácter, podemos utilizar [abcdefghijklmnopqrstuvwxyz]. Debido a que esto es un rango, una forma mucho más simple de obtener el mismo resultado es poniendo:

```
$ ls [a-z]*
```

Los rangos no tienen que ser series completas de números o caracteres, podemos expresar subconjuntos de ellos. Por ejemplo, si queremos buscar entradas que comiencen por alguna letra entre la "d" y la "t", podemos utilizar indistintamente [defghijklmnopqrst] o [d-t]. Si la entrada puede comenzar por esas letras tanto en mayúsculas como en minúsculas, podemos usar [DEFGHIJKLMNOPQRSTdefghijklmnopqrst] o [D-Td-t]

Otros ejemplos son:

- Todas las letras (mayúsculas y minúsculas): [A-z]
  [A-z] es lo mismo que [A-Z] y [a-z]
- Todos los números: [0-9]
- Cualquier carácter que no sea un número: [!0-9]
- Cualquier carácter que no sea una letra: [!A-z]

Alias de comandos

Aunque el sistema operativo y la shell ofrecen multitud de comandos y utilidades, puedes crear alias con nombres que tengan más sentido para ti o que sean más pequeños y así teclear menos caracteres. Por ejemplo, si estás familiarizado con la línea de comandos de Windows o del DOS estarás acostumbrado a escribir **dir** para obtener una lista de ficheros de un directorio, para obtener lo mismo en Linux se escribiría **ls -l**. Podrías crear un alias de tal forma que cada vez que escribieses **dir** se ejecutase **ls -l**, utilizando la siguiente expresión:

```
alias dir="ls -l"
```

La sintaxis será siempre el alias seguido del comando que habrá de ejecutarse cuando se teclee el alias separados por el signo igual (=). En raras ocasiones podrías hacerlo sin encerrar el comando entre comillas, pero como norma general siempre deberías incluirlas.

¡Ojo! Para que los alias no se pierdan al finalizar la sesión deben añadirse al archivo .bashrc que se encuentra en el directorio home del usuario.

Path y otras variables

Cuando tecleas un comando en el prompt la shell primero busca entre sus comandos internos y de no encontrar el comando se buscará una utilidad(comando) externa con ese mismo nombre. Esto se realiza buscando en los directorios incluidos en la variable **PATH** y en el mismo orden en el que han sido definidos en dicha variable hasta que se encuentra el primer archivo ejecutable cuyo

nombre coincide con el tecleado, en caso de no encontrarse ninguno después de buscar en todos los directorios indicados aparecerá el mensaje (“command not found”).

A continuación se remarcan algunas cosas importantes que deben conocerse respecto al path:

- Puedes consultar el valor actual de PATH con el siguiente comando:

```
echo $PATH
```

En el path no se incluye por defecto el directorio actual, por tanto, podrías tener un fichero ejecutable en tu directorio, ver que está ahí (tecleando **ls**) pero al escribir su nombre obtener un error de “command not found”.

Para solucionar esto podrías escribir el pathname completo del fichero a ejecutar, añadir este directorio a la variable **PATH**, mover el fichero a un directorio incluido en dicha variable, o añadir el símbolo del directorio actual (.) a la variable **PATH**.

Las distintas entradas en el path irán separadas por dos puntos (:).

El orden de búsqueda en la variable **PATH** debería comenzar por los directorios donde se encuentran los comandos más comunes (los directorios bin) y terminar por los directorios donde se encuentren los comandos del usuario, si existiesen.

Para añadir un directorio al path puedes redefinir la declaración completa o, simplemente, añadir el nuevo directorio con el comando:

```
PATH=$PATH:{nuevo directorio}
```

Por tanto, para añadir el directorio /home/fulanito al path, el comando sería:

```
PATH=$PATH:/home/fulanito
```

Si quieres añadir una entrada tal que el directorio donde estás trabajando en ese momento esté siempre incluido en el path de búsqueda, escribe:

```
PATH=$PATH:./
```

Por motivos de seguridad te recomendamos que no hagas ésto último, pero si no tienes más remedio que hacerlo, colócalo siempre al final de la declaración de **PATH** como se comentó anteriormente y no al principio.

Variables Comunes

Podemos ver el valor de cualquier variable existente con el siguiente comando:

```
echo ${nombre_de_variable}
```

Por tanto, el comando:

```
echo $MAIL
```

mostrará el directorio de correo, $HOME, el directorio home, y así sucesivamente. Para obtener una lista completa de todas las variables definidas en el entorno puedes utilizar cualquiera de estos dos comandos: **env** y **set**.

Aunque las salidas de los mismos puedan variar ligeramente (variables del entorno contra variables locales), en su mayor parte la salida de **env** es un subconjunto de la salida de **set**.

Algunas de las variables que podemos visualizar son:

- **HOME**—El directorio donde inicias la sesión y a donde llegas si tecleas **cd** sin ningún parámetro adicional.
- **LINES**—El número de líneas de pantalla que se visualizarán entre cada pausa (comando more).
- **LOGNAME**— El nombre de usuario con el que has conectado.
- **PWD**— El directorio de trabajo actual, el directorio donde te encuentras en este momento.
- **SHELL**— El intérprete de comandos que estás utilizando.
- **TERM**— El tipo de terminal o emulación seleccionado.
- **USER**— Suele ser igual que **LOGNAME**

Como regla general, las variables del sistema siempre aparecen en mayúsculas.

Puedes cambiar el valor de las variables según lo necesites o añadir tus propias variables para que sean utilizadas por otros programas o desde la línea de comando. Por ejemplo para crear una nueva variable llamada **HOY**, solo tendrías que escribir:

```
HOY=Viernes
```

Ahora puedes ver el valor de esa variable escribiendo:

```
echo $HOY
```

El resultado es **Viernes**. Si ahora empleas el comando:

```
set
```

la variable aparecerá ahí, sin embargo si usas el comando:

```
env
```

no aparecerá. La variable ha sido creada localmente y solo podrá ser referenciada localmente. Para que sea accesible por subrutinas o procesos hijos debes utilizar el comando **export** que hará que pase de ser local a ser una variable del entorno:

```
export HOY
```

Esto pondrá la variable en el entorno donde podrá ser encontrada tanto localmente como por

las subrutinas y procesos hijos, la variable y su valor serán accesibles durante toda la sesión y se perderán una vez desconectes.

Para que el valor sea permanente debes añadir la definición a un perfil, por ejemplo puedes cambiar el valor de **PATH** para todos los usuarios del sistema, conectándote como root y editando el archivo /etc/profile modificando la línea donde se define la variable **PATH.** Ésto podrías hacerlo utilizando el editor vi con el siguiente comando:

```
vi /etc/profile
```

Si quisieses hacer la modificación solo para un usuario en particular tendrías que editar el archivo /home/nombre_del_usuario/.bash_profile (el punto al inicio del nombre del fichero indica que este es un fichero oculto, tendrías que teclear ls -a para poder verlo).

Las modificaciones en los perfiles solo se harán efectivas tras la desconexión y nueva conexión del usuario.

Para cambiar el valor de una variable, simplemente, vuelve a definirla con el nuevo valor:

```
HOY=Lunes
```

Como ya la habíamos exportado anteriormente ya no tenemos por que hacerlo otra vez y el nuevo valor quedará accesible tanto localmente como en el entorno.

Si fuese necesario eliminar una variable puedes utilizar el comando **unset**.

Configurando el prompt

Entre las variables presentes y definibles, están aquellas que definen el prompt. El prompt es el mensaje que visualiza la shell cuando está lista para recibir un comando.

Los prompts por defecto incluyen:

- $ El último carácter para sh, bash, y ksh
- % El último carácter para csh y zsh
- > El último carácter para tcsh

El prompt primario puede ser tanto la variable **PS1** o la variable **prompt**, dependiendo de que shell estés utilizando. En bash, un valor típico para **PS1** sería:

```
[\u@\h \W]\$
```

Componente por componente, **PS1** sería igual a lo siguiente:

- El corchete izquierdo ([)
- El nombre del usuario actual (**\u**)

- La arroba (@)
- El nombre del host actual (**\h**)
- Un espacio
- El directorio de trabajo actual (**\W**)
- El corchete derecho (**]**)
- El signo del dolar (**$**)

Un ejemplo de este prompt sería:

```
[leko@pentimVIII home]$
```

La barra invertida (\) indica la utilización de un valor especial. Algunos de estos valores se muestran en la tabla 1.1.

TABLA 1.1

Valor	Resultado
\d	Fecha actual
\h	Nombre del host hasta el primer punto
\n	Interlínea
\s	Shell
\t	Hora actual
\u	Nombre usuario
\W	Directorio actual
\w	Ruta completa de directorios
\!	Número del historial (se comentará mas adelante)
\#	Número del comando
\$	Prompt por defecto—$ para los usuarios normales y # para root
****	Barra invertida literal
ABC	Texto libre
\[	Secuencia de caracteres no imprimibles
\]	Fin de secuencia de caracteres no imprimibles
$date	Salida del comando date (o cualquier otro)

Para modificar el prompt de forma permanente para todos los usuarios hace falta editar el fichero /etc/bashrc (como root) y modificar su contenido. Teniendo en cuenta que si un usuario tiene en su directorio personal el fichero ~/.bashrc se sobreescribirá el contenido en /etc/bashrc.

Si te fijas en las variables del sistema verás que, además de **PS1,** puedes encontrar **PS2**.

Anteriormente se comentó que podía terminarse una línea de comandos con una barra invertida para indicarle a la shell que aun no se había terminado de introducir todo el comando, si observamos un ejemplo podemos ver lo siguiente:

```
[leko@pentimVIII home]$ ls -l *.gif \
> *.fig \
> *.bmp
```

Observa como el prompt cambió de lo indicado en **PS1** a un signo de mayor (**>**). Si hubiese seguido siendo el mismo **PS1** no podrías saber si se trata del mismo comando o de uno nuevo, por eso se cambia del prompt primario a otro secundario definido, precisamente por la variable **PS2**. Su valor podrá ser cambiado de la misma forma que el de **PS1,** incluyendo los valores especiales de la Tabla 1.1. La mayoría de las shells admiten tres o cuatro niveles de prompts.

Otras Variables

A estas alturas ya te habrás dado cuenta de que el signo del dolar ($) se utiliza para obtener el valor de una variable; si tienes una variable llamada **EJEMPLO**, puedes ver su contenido examinando **$EJEMPLO**.

Hay otras tres variables que pueden resultar prácticas para determinar tu entorno.

La primera—$$—muestra el ID del proceso correspondiente a la shell que se ejecuta actualmente:

```
echo $$
```

La segunda—$?—muestra el resultado del último comando ejecutado. Este resultado puede ser correcto (**0**) o incorrecto (**1**). Por ejemplo, el comando **ls** admite la opción **-F** que diferenciará entre ficheros y directorios insertando una barra invertida detrás del nombre de los directorios, sin embargo este comando no incluye la opción **-z**.

Dada esta información, en el ejemplo siguiente podremos comprobar la utilización de la variable $?

```
[leko@pentimVIII home]$ ls -F
Desktop\ sample snapshot01.gif snapshot02.gif
[leko@pentimVIII home]$ echo $?
0
[leko@pentimVIII home]$ ls -z
ls: invalid option — z
[leko@pentimVIII home]$echo $?
1
```

La tercera variable—$!—mostrará el ID del último proceso hijo que se inició en el background . Si no se hubiese iniciado ningún proceso de background, entonces la variable no tendría valor.

Los procesos se describen con mayor detalle en otro capítulo pero para esta explicación es

suficiente con saber que poniendo un ampersand (**&**) al final de un comando le indicamos a la shell que ejecute dicho comando en el background:

```
[leko@pentimVIII home]$ echo $!
[leko@pentimVIII home]$ ls -F &
[leko@pentimVIII home]$ echo $!
19321
[leko@pentimVIII home]$
```

Usando el historial de comandos

El archivo del historial contiene una lista de los comandos introducidos en la línea de comando. La variable **HISTSIZE** define el número de comandos que se almacenarán en dicho archivo durante la sesión actual, esta variable puede estar definida tanto en /etc/profile como en ~/.profile (~ equivale al directorio home del usuario) y su valor por defecto es de 1000 entradas. El comando history muestra todas las entradas del archivo del historial que se guarda en ~/.bash_history.

Puedes navegar por las entradas del histórico con las flechas del teclado. La flecha hacia arriba mostrará las entradas anteriores, mientras que la flecha hacia abajo avanzará hacia adelante en el historial. De esta forma podemos ahorrarnos volver a escribir comandos que ya habíamos escrito con anterioridad. Mientras navegamos por los comandos podemos editarlos antes de ejecutarlos de nuevo.

La variable **HISTCMD** proporciona el índice dentro del historial comando que se está ejecutando. La variable **HISTFILE** especifica el nombre del fichero que contendrá el historial (~/.bash_history por defecto). La variable **HISTFILESIZE** especifica el máximo número de líneas que contendrá el fichero especificado en **HISTFILE** y, aunque suele tener el mismo valor que **HISTSIZE**, podría ser diferente ya que ésta última se refiere solo al histórico de la sesión actual y no al tamaño total del archivo histórico**.**

```
fc
```

La utilidad fc proporciona otra opción para editar los comandos del fichero histórico antes de ejecutarlos. La utilidad fc abre el editor de textos por defecto con el comando especificado y ahí podemos editarlo y salvarlo antes de ejecutarlo disponiendo de toda la potencia de un editor de textos. Podemos llamar a fc utilizando como parámetro el número del comando que queremos editar o, también, con un rango de comandos para, de esta forma, editarlos y ejecutarlos en conjunto. También es posible especificar el editor de textos a utilizar. Una vez que se ha llamado a fc el fichero de comandos puede ser editado como cualquier otro fichero de texto y los comandos editados se ejecutarán al salir del editor. La opción -l se utiliza para mostrar una lista de los valores especificados a continuación, podéis escribir, por ejemplo, fc -l 50 60 y obtendréis la lista de los comandos del 50 al 60 en el historial.

Tabla 1-2. Operadores de expansión del historial de comandos

Operador	Descripción
!!	También conocido como bang-bang,[1] este operador hace referencia al comando más reciente del historial.
!n	Hace referencia al comando número n del historial. Puedes utilizar el comando history para conocer estos números.
!-n	Hace referencia al comando actual menos n en el historial.
! cadena	Hace referencia al comando más reciente que comience por cadena.
!? cade	Hace referencia al comando más reciente que contenga cadena.
^ cadena1^cadena2	Sustitución rápida. Repite el último comando reemplazando la primera aparición de cadena1 por cadena2.

[1] Es común llamar bang al signo de admiración en los sistemas Linux y Unix.

– **A** l trabajar con Linux habrá ocasiones en las que necesites una información más amplia sobre la utilización de algún comando, utilidad o configuración del sistema. Aunque este y otros libros pueden ser muy útiles, ningún libro puede tener la información totalmente actualizada. Afortunadamente existen magníficos recursos para cuando necesitemos más información. Algunas de estas fuentes de información las podremos encontrar en nuestro sistema local, mientras que otras estarán disponibles en Internet. Este capítulo te informará sobre algunos de los lugares más útiles para buscar información. Este conocimiento te hará ahorrar mucho tiempo cuando trabajes con sistemas Linux y es esencial para la preparación del examen.

Obteniendo Ayuda con las Páginas Man

• Uso y administración de la documentación del Sistema Local. El uso y administración de los recursos de man y del material de /usr/doc/. Incluye el encontrar las páginas man más relevantes, buscar por las secciones de man, encontrar comandos y las páginas man relativas a éstos, configurar el acceso a las fuentes de man y al sistema man, utilizar la documentación del sistema almacenada en /usr/doc/ y otros lugares relacionados y determinar que documentación mantener en /usr/doc/.

Las páginas del manual o páginas man de 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 formatea 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.

Todo esto aparecería de esta forma:

LS(1) FSF LS(1)

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

NAME
ls - list directory contents
Objective

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.
-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 reltiva 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 <bug-fileutils@gnu.org>.
COPYRIGHT
Copyright (c) 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 desplazarse hacia abajo por las páginas man se utiliza la barra espaciadora, que avanzará una página por cada pulsación. La tecla Q provoca la salida del proceso de visualización de la página. Si quieres buscar algún texto dentro de la página man puedes utilizar las expresiones regulares, a continuación se muestra un ejemplo de como 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. Por defecto las páginas man se guardan en los siguientes lugares.

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

El significado de los números se comentará en la siguiente sección del capítulo, “Buscando secciones de las páginas man”

Pregunta de Examen: Asegurate que sabes la ubicación por defecto de los ficheros fuentes de las páginas man. Esta pregunta es muy probable que salga en el examen.

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. El comando man admite distintas opciones, una de ellas permite usar un path distinto al indicado en MANPATH. Las opciones del comando man se muestran en la tabla 1-3.

Tabla 1-3

Opciones utilizadas con man

Opción	Uso
-C fichero-de-configuración	Indica un fichero de configuración distinto a /etc/man.conf.
-M path	Indica en que directorios se buscarán las páginas man.

Opción	Uso
-P paginador	Indica el paginador o el programa utilizado para formatear y visualizar las páginas man. El paginador por defecto es el indicado en la variable de entorno PAGER Los paginadores more y less son los más frecuentemente utilizados.
-S Lista-de-secciones	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	Indica que debe mostrarse 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 en lugar de las páginas.

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:

Tabla 1-4

Secciones de las Páginas man

Sección	*Uso*
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 retorna 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 quieres buscar. Si quisieses buscar información sobre la utilidad crontab en la sección cinco utilizarías el siguiente comando:

```
# man 5 crontab
```

Si usas la opción -a tal como se muestra en la Tabla 7-1 podrás examinar todas las entradas coincidentes. Siguiendo el ejemplo de la utilidad crontab tendrías que utilizar el siguiente comando:

```
# man -a crontab
```

Buscando con 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 puedes 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.

Buscando con 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
        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
```

pg_passwd (1) - Manipulate the flat password file
pwupdate (8) - updates passwd and shadow NIS map
rpc.yppasswdd [rpc] (8) - NIS password update daemon
smbpasswd (5) - The Samba encrypted password file
smbpasswd (8) - change a users SMB password
ypchfn [yppasswd] (1) - change your password in the NIS database
ypchsh [yppasswd] (1) - change your password in the NIS database
yppasswd (1) - change your password in the NIS database

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
pg_passwd (1) - Manipulate the flat password file
pwupdate (8) - updates passwd and shadow NIS map
rpc.yppasswdd [rpc] (8) - NIS password update daemon
smbpasswd (5) - The Samba encrypted password file
smbpasswd (8) - change a users SMB password
ypchfn [yppasswd] (1) - change your password in the NIS database
ypchsh [yppasswd] (1) - change your password in the NIS database
yppasswd (1) - change your password in the NIS database

Como puedes ver estos comandos hacen lo mismo. Esto puede ser práctico cuando busques comandos a partir de determinadas palabras clave.

Configurando el acceso a las páginas man

Como se mencionó con anterioridad en este capítulo, en el directorio /usr/man se guardan por defecto los ficheros fuente de las páginas man. La variable de entorno MANPATH puede ser utilizada para cambiar el path de búsqueda por defecto de los ficheros fuente de las páginas man. La variable MANPATH sobreescribirá el path de búsqueda por defecto de las páginas man, así que es importante incluir el path de las páginas man existentes. Abajo tenemos un ejemplo de una definición de la variable MANPATH añadida a un fichero /home/user/.profile.

```
Export
MANPATH=/usr/local/man:/usr/man/preformat:/usr/man:/usr/X11R6/man
```

La mayoría de los documentos almacenados en /usr/man están comprimidos y sin formatear. El comando man utiliza el fichero /etc/man.config para obtener información acerca de la correcta visualización de esos fichero.

Este fichero contiene la definición de MANPATH, así como, información relativa a las opciones de compresión, formateo y paginación. Mediante la opción -C mostrada en la Tabla 7-1,

podemos especificar un fichero de configuración diferente.

El comando man se encuentra en /usr/bin. Este directorio debe estar incluido en la variable de entorno PATH, de lo contrario deberemos ejecutar el comando utilizando el path absoluto /usr/bin/man.

Preguntas Pre-TEST

1. Que fichero contiene la configuración por defecto del shell para cada usuario?
2. Cual es la mejor forma de añadir la variable de entorno PATH para todos los usuarios en un sistema?
3. Como es el tamaño del fichero del historial bash?
4. Que tecla se usa para completar comandos con el bash shell?
5. Que comando permite editar el ultimo comando que introdujiste usando el editor por defecto?
6. Que fichero contiene el historial de comandos?
7. Que comando usuarias para ver la configuración de la variable de entorno HOME?
8. Como ejecutas un programa localizado en el pwd?
9. Que función permite la salida desde un comando para ser usada en lugar del comando?
10. Que opción se usa para operaciones recursivas de un comando?

Preguntas TEST

1. Que shell es usada por defecto en los sistemas GNU/Linux?
   A. csh
   B. rsh
   C. bash
   D. tcsh

2. Que fichero contiene los shells disponibles para el sistema?
   A. /etc/passwd
   B. /etc/command
   C. /etc/bash
   D. /etc/shells

3. Cual es el comando usado para cambiar el shell por defecto para el BASH2?
   A. chng -s /bin/bash2
   B. chsh -s /bin/bash2
   C. shell -c /bin/bash2
   D. default -shell /bin/bash2

4. Cual es el formato correcto para introducir un comando en la linea de comandos?
   A. command
   B. command options
   C. command arguments options
   D. command options arguments

5. Se pueden poner varios comandos en la misma linea separados porque carácter?
   A. .
   B. ;
   C. ,
   D. \

6. Que combinación de teclas es usada para introducir un comando en varias lineas?
   A. ""
   B. \ Enter
   C. / Enter
   D. Tab-Enter

7. Que tecla, presionada una vez, es usada para completar los comandos?
   A. Tab
   B. Esc
   C. Enter
   D. Ctrl

8. Que tecla, cuando es presionada dos veces, es usada para completar los comandos?
   A. Tab
   B. Esc
   C. Enter
   D. Ctrl

9. Que fichero contiene las variables del sistema que tiene el shell bash?
   A. /bash
   B. /bin/bash
   C. /etc/bash
   D. /etc/profile

10.. _____hará que se ejecute el script llamado update_mozilla guardado en tu directorio home, desde ese directorio.

11. Cuando se hacen cambios a las variables del entorno, que comando debe de usarse para asegurarse de que los cambios están disponibles para la shell?
   A. save
   B. remember
   C. export
   D. echo

12. Que variable de entorno se usa para personalizar el prompt?
   A. PS1
   B. prompt
   C. shell
   D. display

13. Que fichero contiene la asignación a la carpeta del usuario home?
   A. /etc/home
   B. /etc/profile
   C. /etc/passwd
   D. /etc/users

14. Que comando lista los comandos ejecutados anteriormente?
    A. commands
    B. review
    C. history
    D. export

15. Que tecla permite ver el último comando ejecutado?
    A. down arrow
    B. up arrow
    C. right arrow
    D. left arrow

16. Que comando te permite usar el editor por defecto para editar varios comandos del archivo historico?
    A. history
    B. edit
    C. fc
    D. view

17. Es posible ejecutar comandos que no están en el PATH, si conoces la ruta completa y el nombre del comando
    A. Verdadero
    B. Falso

Respuestas Pre-Test

1. El fichero /etc/passwd contiene el shell por defecto para cada usuario.

2. Añadiendo a la variable PATH editando el fichero /etc/profile, añadira el PATH a todos los usuarios del sistema.

3. La variable HISTSIZE indica el numero de entradas a almacenar en el fichero .bash_history

4. La tecla TAB presionada una vez, y la tecla ESC presionada dos veces completa los comandos, si solo hay un posible comando lo iguala.

5. El comando fc permite editar las entradas desde el fichero ~/.bash_history usando el editor por defecto del sistema.

6. El fichero .bash_history, localizado en el directorio home de cada usuario, contiene el historial de comandos.

7. Se pueden ver las variables usando el comando echo, como echo $HOME.

8. Para ejecutar un comando desde pwd, escribe el comando precedido de ./, como en ./commandname

9. La sustitución de comandos permite la salida de un comando en lugar del nombre del comando.

10.La opción -R permite funciones recursivas de un comando.

Respuestas TEST

1. C. El shell por defecto usado en los sistemas Linux es bash. Ver la sección "Entendiendo Shells" para mas información.

2. D. El fichero /etc/shells contiene una lista de los shells disponibles. Ver la sección de "Entendiendo Shells".

3. B. El comando chsh se usa para ver y cambiar la configuración del shell. Ver la sección "Entendiendo Shells" para mas información.

4. D. Las opciones del comando preceden a los argumentos cuando se entran en la linea de comandos. Ver la sección "Usando la linea de comandos" para mas información.

5. B. El punto y coma (;) se usa para separar multiples comandos en una sola linea. Ver la sección "Usando la linea de comandos" para mas información.

6. B. La tecla \ causa que cualquier tecla que le siga sea ignorada. Esto permite a la tecla Enter ser ignorada y que continue el comando de la linea siguiente. Ver la sección "Usando la linea de comandos" para mas información.

7. A. La tecla TAB completa comandos cuando se pulsa una vez. Ver la sección "completar comandos" para mas información.

8. B. La tecla escape completa comandos cuando se pulsa dos veces. Ver la sección "completar comandos" para mas información.

9. D. El fichero /etc/profile almacena la declaración de las variables de todo el sistema usando el shell bash. Ver la sección "Entendiendo Shells" para mas información.

10. ./update_mozilla. Cuando ejecutamos un comando desde pwd, el nombre del comando se precede por un punto y una barra (./), que especifica que el comando esta en el directorio actual. Ver la sección "Editando la variable path" para mas información.

11.C. El comando export causa cambios en una variable de entorno para que sea accesible por el shell del usuario. Ver la sección "Variables de entorno y configuraciones" para mas información.

12.A. La variable PS1 se usa para hacer cambios al prompt. Ver la sección "Prompt" para mas información.

13.C. El fichero /etc/passwd contiene la asignación de directorios del home del usuario. Ver la sección "variables de entorno y configuraciones" para mas información.

14. C. El comando history muestra los comandos introducidos previamente que se encuentran almacenados en el fichero del usuario .bash_history. Ver la sección “”Usando el fichero history” para mas información.

15. B. La flecha hacia arriba se usa para pasearse por los comandos introducidos previamente. Ver la sección “Usando el fichero history” para mas información.

16. C. El comando fc permite editar los comandos introducidos previamente usando el editor por defecto. Ver la sección “fc” para mas información.

17. A. Los comandos se pueden ejecutar usando el path completo si el usuario tiene los permisos necesarios. Ver la sección “Editando la variable PATH” para mas información.

Bibliografia y enlaces recomendados

LPIC 1 Certification Bible (Bible) by Angie Nash, Jason Nash
John Wiley & Sons; Bk&CD-Rom edition (July 1, 2001) ISBN: 0764547720

LPI Linux Certification in a Nutshell by Jeffrey Dean
O'Reilly & Associates; 1st ed edition (May 15, 2001) ISBN: 1565927486

CramSession's LPI General Linux Part 1 : Certification Study Guide
CramSession.com; ISBN: B000079Y0V; (August 17, 2000)

Referencias Unix Reviews
http://www.unixreview.com/documents/s=7459/uni1038932969999/

Página LPI: www.lpi.org

Apuntes IBM: http://www-106.ibm.com/developerworks/edu/l-dw-linux-lpir21-i.html

Manuales GPL: http://www.nongnu.org/lpi-manuals/