1.5 Ejecución de comandos con la shell Bash

Objetivos

Tras finalizar esta sección, los estudiantes deberían poder ahorrar tiempo en la ejecución de comandos a partir de un shell prompt con los accesos directos de Bash.

Ejecución de comandos con la shell Bash

GNU Bourne-Again Shell (**bash**) es un programa que interpreta comandos escritos por el usuario. Cada secuencia escrita en la shell puede tener un máximo de tres partes: el comando, las opciones (que comienzan con un - o --) y los argumentos. Cada palabra escrita en la shell está separada de las otras por espacios. Los comandos son nombres de programas que están instalados en el sistema. Cada comando tiene sus propias opciones y argumentos.

Cuando el usuario esté listo para ejecutar el comando, presione la tecla **Enter**. Cada comando se escribe en una línea separada y el resultado de cada comando se muestra antes de que la shell muestre un aviso. Si el usuario quiere escribir más de un comando en una sola línea, puede usarse un punto y coma **;** como separador de comando. El punto y coma pertenece a la clase de caracteres denominada *metacaracteres* que tienen un significado especial para **bash**.

Ejemplos de comandos simples

El comando **date** se usa para mostrar la fecha y hora actuales. Además, puede ser usado por el superusuario para configurar el reloj del sistema. Un argumento que comienza con el signo más (+) especifica una secuencia de formato para el comando de fecha.

```
[student@desktopX ~]$ date
Sat Apr  5 08:13:50 PDT 2014
[student@desktopX ~]$ date +%R
08:13
[student@desktopX ~]$ date +%x
04/05/2014
```

El comando **passwd** cambia la contraseña propia del usuario. La contraseña original de la cuenta debe especificarse antes de que se permita un cambio. De manera predeterminada, **passwd** se configura para solicitar una contraseña más sólida, que esté compuesta por letras minúsculas, letras mayúsculas, números y símbolos, y no que se base en una palabra del diccionario. El superusuario puede usar el comando **passwd** para cambiar las contraseñas de otros usuarios.

```
[student@desktopX ~]$ passwd
Changing password for user student.
Changing password for student.
(current) UNIX password: old_password
New password: new_password
Retype new password: new_password
passwd: all authentication tokens updated successfully.
```

Linux no requiere de extensiones de nombre de archivo para clasificar los archivos por tipo. El comando **file** detecta el comienzo de los contenidos de un archivo y muestra qué tipo de archivo es. Los archivos que se clasificarán pasan como argumentos para el comando.

```
[student@desktopX ~]$ file /etc/passwd
/etc/passwd: ASCII text
[student@desktopX ~]$ file /bin/passwd
/bin/passwd: setuid ELF 64-bit LSB shared object, x86-64, version 1
(SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.32,
BuildID[sha1]=0x91a7160a019b7f5f754264d920e257522c5bce67, stripped
[student@desktopX ~]$ file /home
/home: directory
```

Los comandos **head** y **tail** muestran el comienzo y el final de un archivo, respectivamente. De manera predeterminada, estos comandos muestran 10 líneas, pero ambos tienen la opción **-n** que permite la especificación de una cantidad diferente de líneas. El archivo que se mostrará pasa como un argumento para estos comandos.

```
[student@desktopX ~]$ head /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
[student@desktopX ~]$ tail -n 3 /etc/passwd
gdm:x:42:42::/var/lib/gdm:/sbin/nologin
gnome-initial-setup:x:993:991::/run/gnome-initial-setup/:/sbin/nologin
tcpdump:x:72:72::/:/sbin/nologin
```

El comando **wc** cuenta líneas, palabras y caracteres en un archivo. Puede usar la opción **-l**, **-w** o **-c** para mostrar solo las líneas, las palabras o los caracteres, respectivamente.

```
[student@desktopX ~]$ wc /etc/passwd
 39  70 2005 /etc/passwd
[student@desktopX ~]$ wc -l /etc/passwd
39 /etc/passwd
[student@desktopX ~]$ wc -c /etc/group /etc/hosts
843 /etc/group
227 /etc/hosts
1070 total
```

Completar con Tab

Completar con la tecla Tab permite al usuario completar comandos o nombres de archivos rápidamente una vez que haya escrito lo suficiente en el aviso como para hacerlo único. Si los caracteres escritos no son únicos, al presionar la tecla **Tab** dos veces, aparecen todos los comandos que comienzan con los caracteres ya escritos.

```
[student@desktopX ~]$ pas<Tab><Tab>
passwd     paste      pasuspender
[student@desktopX ~]$ pass<Tab>
[student@desktopX ~]$ passwd
Changing password for user student.
Changing password for student.
(current) UNIX password:
```

La opción de completar con Tab puede usarse para completar nombres de archivo cuando se escriben como argumentos para comandos. Si se presiona la tecla **Tab**, completará el nombre del archivo tanto como pueda. Si se presiona la tecla **Tab** por segunda vez, provoca que la shell enumere todos los archivos que coinciden con el patrón actual. Escriba otros caracteres hasta que el nombre sea único; a continuación, use la opción de completar con Tab para finalizar la línea de comandos.

```
[student@desktopX ~]$ ls /etc/pas<Tab>
[student@desktopX ~]$ ls /etc/passwd<Tab>
passwd   passwd-
```

Con esta opción, se puede establecer una coincidencia entre argumentos y opciones para muchos comandos. El comando **useradd** es usado por el superusuario, el usuario `root`, para crear otros usuarios en el sistema. Tiene muchas opciones que pueden usarse para controlar el comportamiento del comando. Puede usarse la

opción de completar con Tab después de una opción parcial para completar la opción sin necesidad de escribir mucho.

```
[root@desktopX ~]# useradd --<Tab><Tab>
--base-dir      --groups        --no-log-init   --shell
--comment       --help          --non-unique    --skel
--create-home   --home-dir      --no-user-group --system
--defaults      --inactive      --password      --uid
--expiredate    --key           --root          --user-group
--gid           --no-create-home --selinux-user
[root@desktopX ~]# useradd --
```

Comando history

El comando **history** muestra una lista de los comandos ejecutados anteriormente que tienen un número de comando como prefijo.

El signo de exclamación, **!**, es un metacarácter que se usa para expandir los comandos anteriores sin tener que volver a escribirlos. **!*number*** se amplía hasta el comando que coincide con el número especificado. **!*string*** expande el comando más reciente que comienza con la secuencia especificada.

```
[student@desktopX ~]$ history
...Output omitted...
  23  clear
  24  who
  25  pwd
  26  ls /etc
  27  uptime
  28  ls -l
  29  date
  30  history
[student@desktopX ~]$ !ls
ls -l
total 0
drwxr-xr-x. 2 student student 6 Mar 29 21:16 Desktop
...Output omitted...
[student@desktopX ~]$ !26
ls /etc
abrt                hosts               pulse
adjtime             hosts.allow         purple
aliases             hosts.deny          qemu-ga
...Output omitted...
```

Las teclas de flecha pueden usarse para navegar por las líneas de comandos anteriores en el historial de la shell. La tecla **Up Arrow** edita el comando anterior en la lista de historial. La tecla **Down Arrow** edita el comando siguiente en la lista de historial. Use esta tecla cuando la tecla **Up Arrow** se haya presionado demasiada veces. Use las teclas **Left Arrow** y **Right Arrow** para mover el cursor hacia la izquierda y derecha en la línea de comandos actual que se está editando.

La combinación de teclas **Esc**+ provoca que la shell copie la última palabra del comando anterior en la línea del comandos actual donde está el cursor. Si se usa en forma reiterada, seguirá avanzando hasta los comandos anteriores.

Edición de línea de comandos

Cuando se usa en forma interactiva, **bash** tiene una función de edición de línea de comandos. Esto permite al usuario utilizar los comandos del editor de texto para desplazarse y modificar el comando actual que se está escribiendo. El uso de las teclas de flecha para moverse dentro del comando actual y pasar por el historial de comando se presentó anteriormente en esta sesión. En la siguiente tabla, se presentan comandos de edición más contundentes.

Accesos directos para la edición de línea de comandos

Acceso directo	Descripción
Ctrl+**a**	Ir al inicio de la línea de comandos.
Ctrl+**e**	Ir al final de la línea de comandos.
Ctrl+**u**	Borrar desde el cursor hasta el principio de la línea de comandos.
Ctrl+**k**	Borrar desde el cursor hasta el final de la línea de comandos.
Ctrl+**Left Arrow**	Ir al inicio de la palabra anterior en la línea de comandos.
Ctrl+**Right Arrow**	Ir al final de la palabra siguiente en la línea de comandos.
Ctrl+**r**	Buscar en la lista de historial de comandos para un patrón.

Hay muchos otros comandos de edición de línea de comandos disponibles, pero estos son los más prácticos para usuarios principiantes. Los otros comandos están en la página de manual **bash**(1).

Referencias

Páginas de manual **bash**(1), **date**(1), **file**(1), **head**(1), **passwd**(1), **tail**(1) y **wc**(1)

Back Next