El Bourne Again Shell

¿Qué es una shell?

Sí, ¿qué es exactamente una shell? Bueno, un shell es básicamente un entorno de usuario de línea de comandos. En esencia, es una aplicación que se ejecuta cuando el usuario inicia sesión y le permite ejecutar aplicaciones adicionales. En cierto modo, es muy similar a una interfaz gráfica de usuario, ya que proporciona un marco para ejecutar comandos y ejecutar programas. Hay muchos shells incluidos con una instalación completa de Slackware, pero en este libro solo hablaremos de ***bash*** (1), el Shell Bourne Again. Los usuarios avanzados pueden querer considerar el uso del potente ***zsh*** (1), y los usuarios familiarizados con los sistemas UNIX más antiguos pueden apreciar ***ksh***. El verdadero masoquista podría elegir el ***csh***, pero los nuevos usuarios deberían limitarse a ***bash***.

Variables de entorno

Todos los shell facilitan ciertas tareas para el usuario al realizar un seguimiento de las cosas en las variables de entorno. Una variable de entorno es simplemente un nombre más corto para un poco de información que el usuario desea almacenar y utilizar más adelante. Por ejemplo, la variable de entorno PS1 le dice a **bash** cómo dar formato a su solicitud. Otras variables pueden indicar a las aplicaciones cómo ejecutarse. Por ejemplo, la variable LESSOPEN le dice a ***less*** que ejecute lesspipe.sh, ese práctico preprocesador del que ya hablamos, y LS_OPTIONS ajusta el color para ***ls***.

Configurar sus propias variables de entorno es fácil. ***bash*** incluye dos funciones incorporadas para manejar esto: ***set*** y ***export***. Además, una variable de entorno se puede eliminar utilizando ***unset***. (No entre en pánico si accidentalmente desestabiliza una variable de entorno y no sabe lo que haría. Puede restablecer todas las variables predeterminadas cerrando sesión en su terminal y volviendo a iniciar sesión). Puede hacer referencia a una variable colocando un signo dólar ($) en frente de él.

```
darkstar:~$ set FOO=bar
darkstar:~$ echo $FOO
bar
```

La principal diferencia entre ***set*** y ***export*** es que la ***export*** (naturalmente) exportará la variable a cualquier sub-shell. (Un sub-shell es simplemente otro shell que se ejecuta dentro de un shell primario). Puede ver fácilmente este comportamiento cuando trabaja con la variable PS1 que controla el indicador de ***bash***.

```
darkstar:~$ set PS1='FOO'
darkstar:~$ export PS1='FOO'
FOO
```

Hay muchas variables de entorno importantes que utilizan ***bash*** y otras shells, pero una de las más importantes con las que se encontrará es PATH. PATH es simplemente una lista de directorios para buscar aplicaciones. Por ejemplo, top (1) se encuentra en ***/usr/bin/top*** . Puede ejecutarlo

simplemente especificando la ruta completa, pero si /usr/bin está en su variable PATH, ***bash*** comprobará allí si no especifica una ruta completa. Lo más probable es que primero lo note cuando intente ejecutar un programa que no esté en su RUTA como usuario normal, por ejemplo, ***ifconfig*** (8).

```
darkstar:~$ ifconfig
bash: ifconfig: command not found
darkstar:~$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin:/usr/games:/opt/www/htdig/bin:.
```

Arriba, ve un PATH típico para un usuario normal. Puede cambiarlo por su cuenta de la misma manera que cualquier otra variable de entorno. Sin embargo, si inicia sesión como root, verá que la raíz tiene un PATH diferente.

```
darkstar:~$ su -
Password:
darkstar:~# echo $PATH
/usr/local/sbin:/usr/sbin:/sbin:/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin:
/usr/games:/opt/www/htdig/bin
```

Comodines

Los comodines son caracteres especiales que le dicen al shell que coincida con ciertos criterios. Si tiene experiencia con DOS, reconocerá * como un comodín que coincide con cualquier cosa. ***bash*** utiliza este comodín y varios otros para permitirle definir fácilmente lo que quiere hacer.

Este primero y el más común de estos es, por supuesto, *. El asterisco coincide con cualquier carácter o combinación de caracteres, incluido ninguno. Por lo tanto, b* coincidiría con cualquier archivo llamado b, ba, bab, babc, bcdb, etc. Un poco menos común es el ?. Este comodín coincide con una sóla aparición de cualquier carácter, entonces b? coincidiría con ba y bb, pero no con b o bab.

```
darkstar:~$ touch b ba bab
darkstar:~$ ls b*
b ba bab
darkstar:~$ ls b?
ba
```

¡No, la diversión no se detiene ahí! Además de estos dos, también tenemos el par de corchetes “[]” que nos permite ajustar exactamente lo que queremos hacer coincidir. Cada vez que ***bash*** ve el par de corchetes, sustituye el contenido del corchete. Cualquier combinación de letras o números puede especificarse en el corchete siempre que estén separados por comas. Además, también se pueden especificar rangos de números y letras. Esto probablemente se muestra mejor por ejemplo.

```
darkstar:~$ ls a[1-4,9]
a1 a2 a3 a4 a9
```

Dado que Linux distingue entre mayúsculas y minúsculas, las letras mayúsculas y minúsculas se tratan de manera diferente. Todas las letras mayúsculas aparecen antes de todas las letras

minúsculas en orden *"alfabético"*, por lo que cuando utilice rangos de letras mayúsculas y minúsculas, asegúrese de hacerlas correctamente.

```
darkstar:~$ ls 1[W-b]
1W 1X 1Y 1Z 1a 1b
darkstar:~$ ls 1[w-B]
/bin/ls: cannot access 1[b-W]: No such file or directory
```

En el segundo ejemplo, 1 [bW] no es un rango válido, por lo que el shell lo trata como un nombre de archivo, y como ese archivo no existe, ***ls*** lo indica.

Completar con Tab

¿Todavía piensa que el uso de comodines da demasiado trabajo? Tiene razón. Hay una forma aún más fácil cuando se trata de nombres de archivo largos: completar con tab. Completar con tab le permite escribir solo lo suficiente del nombre de archivo para identificarlo de manera única, luego, al presionar la tecla TAB, ***bash*** completará el resto por usted. Incluso si no ha escrito suficiente texto para identificar de forma única un nombre de archivo, el shell completará todo lo que pueda para usted. Si pulsa TAB una segunda vez, se le mostrará una lista de todas las coincidencias posibles.

Redirección de entrada y salida

Una de las características definitorias de Linux y otros sistemas operativos similares a UNIX es la cantidad de aplicaciones pequeñas y relativamente simples y la capacidad de apilarlas juntas para crear sistemas complejos. Esto se logra redireccionando la salida de un programa a otro, o extrayendo la entrada desde un archivo o segundo programa.

Para comenzar, vamos a mostrarle cómo redirigir la salida de un programa a un archivo. Esto se hace fácilmente con el carácter '>'. Cuando ***bash*** ve el carácter '>', redirige toda la salida estándar (también conocida como stdout) al nombre de archivo que sigue.

```
darkstar:~$ echo foo
foo
darkstar:~$ echo foo > /tmp/bar
darkstar:~$ cat /tmp/bar
foo
```

En este ejemplo, le mostramos lo que haría ***echo*** si su salida no se redirigiera a un archivo, luego lo redirigimos al archivo `/tmp/bar`. Si `/tmp/bar` no existe, se crea y la salida de ***echo*** se coloca dentro de él. Si /tmp/bar existiera, entonces su contenido es sobre-escrito. Esta podría no ser la mejor idea si desea mantener esos contenidos en su lugar. Afortunadamente, ***bash*** es compatible con '»', que agregará la salida al archivo.

```
darkstar:~$ echo foo
foo
```

```
darkstar:~$ echo foo > /tmp/bar
darkstar:~$ cat /tmp/bar
foo
darkstar:~$ echo foo2 >> /tmp/bar
darkstar:~$ cat /tmp/bar
foo
foo2
```

También puede re-direccionar el error estándar (o stderr) a un archivo. Esto es ligeramente diferente, ya que debe usar '2>' en lugar de solo '>'. (Dado que ***bash*** puede redirigir input, stdout y stderr, cada uno debe ser identificable de forma única. 0 es input, 1 es stdout y 2 es stderr. A menos que se especifique uno de estos, ***bash*** hará su mejor estimación de lo que realmente quiso decir, y asumirá que cada vez que use '>' solo desea redirigir a stdout. 1> habría funcionado igual de bien.)

```
darkstar:~$ rm bar
rm: cannot remove `bar': No such file or directory
darkstar:~$ rm bar 2> /tmp/foo
darkstar:~$ cat /tmp/foo
rm: cannot remove `bar': No such file or directory
```

También puede redirigir la entrada estándar (conocida como stdin) con el carácter '<', aunque no se usa muy a menudo.

```
darkstar:~$ fromdos < dosfile
```

Finalmente, puede redirigir la salida de un programa como entrada a otro. Esta es quizás la característica más útil de ***bash*** y otros shells, y se logra usando el caracter '|'. (Este caracter se conoce como 'tubería' (pipe). Si se habla de hacer una tubería (piping) de un programa a otro, esto es exactamente lo que significa).

```
darkstar:~$ ps auxw | grep getty
root      2632  0.0  0.0   1656   532 tty2     Ss+  Feb21   0:00
/sbin/agetty 38400 tty2 linux
root      3199  0.0  0.0   1656   528 tty3     Ss+  Feb15   0:00
/sbin/agetty 38400 tty3 linux
root      3200  0.0  0.0   1656   532 tty4     Ss+  Feb15   0:00
/sbin/agetty 38400 tty4 linux
root      3201  0.0  0.0   1656   532 tty5     Ss+  Feb15   0:00
/sbin/agetty 38400 tty5 linux
root      3202  0.0  0.0   1660   536 tty6     Ss+  Feb15   0:00
/sbin/agetty 38400 tty6 linux
```

Administración de tareas

Bash tiene otra característica interesante que ofrecer, la capacidad de suspender y reanudar tareas. Esto le permite detener temporalmente un proceso en ejecución, realizar alguna otra tarea, luego reanudarlo u, opcionalmente, hacer que se ejecute en segundo plano. Al presionar CTRL+Z, ***bash*** suspenderá el proceso en ejecución y lo devolverá a la línea de comando o símbolo de sistema

(prompt). Puede volver a ese proceso más tarde. Además, puede suspender múltiples procesos de esta manera indefinidamente. El comando incorporado de ***jobs*** mostrará una lista de tareas suspendidas.

```
darkstar:~$ jobs
[1]-  Stopped                 vi TODO
[2]+  Stopped                 vi chapter_05.xml
```

Para volver a una tarea suspendida, ejecute el comando incorporado ***fg*** para volver a poner en primer plano la tarea suspendida más reciente. Si tiene múltiples tareas suspendidas, también puede especificar un número para poner una de ellas en primer plano.

```
darkstar:~$ fg # "vi TODO"
darkstar:~$ fg 1 # "vi chapter_05.xml"
```

También puede poner en segundo plano una tarea (sorpresa) ***bg***. Esto permitirá que el proceso continúe ejecutándose sin mantener el control de su shell. Puede volver a ponerlo en primer plano con ***fg*** de la misma manera que con las tareas suspendidas.

Terminales

Slackware Linux y otros sistemas operativos similares a UNIX permiten a los usuarios interactuar con ellos de muchas maneras, pero el más común y posiblemente el más útil es la terminal. En la antigüedad, las terminales eran teclados y monitores (a veces incluso ratones) conectados a un mainframe o servidor a través de conexiones en serie. Hoy en día sin embargo, la mayoría de las terminales son virtuales; Es decir, existen solo en software. Las terminales virtuales permiten a los usuarios conectarse a la computadora sin requerir hardware costoso y, a menudo, incompatible. Más bien, un usuario solo necesita ejecutar el software y se les presenta un terminal virtual (generalmente) altamente personalizable.

Las terminales virtuales más comunes (en el sentido de que cada máquina Slackware Linux tendrá al menos una) son los gettys. ***agetty*** (8) habilita seis terminales de forma predeterminada en Slackware, y permite a los usuarios locales (aquellos que pueden sentarse físicamente frente a la computadora y escribir desde el teclado) iniciar sesión y ejecutar aplicaciones. Cada uno de estos gettys está disponible en diferentes dispositivos tty a los que se puede acceder por separado presionando la tecla ALT y una de las teclas de función desde el F1 principio F6. El uso de estos gettys le permite iniciar sesión varias veces, tal vez como diferentes usuarios, y ejecutar aplicaciones en los shells de esos usuarios de forma silenciosa. Esto se hace comunmente con servidores que no tienen ***X*** instalado, pero se puede hacer en cualquier máquina.

En computadoras de escritorio, computadoras portátiles y otras estaciones de trabajo donde el usuario prefiere una interfaz gráfica proporcionada por ***X***, la mayoría de los terminales son gráficos. Slackware incluye muchos terminales gráficos diferentes, pero los más utilizados son la ***consola*** de KDE y la ***Terminal*** de XFCE (1), así como el antiguo modo de espera, xterm (1). Si está utilizando una interfaz gráfica, verifique sus barras de herramientas o menús. Cada entorno de escritorio o administrador de ventanas tiene una terminal virtual (a menudo llamada emuladora de terminal), y todas están etiquetadas de manera diferente. Sin embargo, normalmente, las encontrará en un submenú “Sistema” en entornos de escritorio. Ejecutar cualquiera de estas opciones le dará una

terminal gráfica y automáticamente ejecutará su shell predeterminado.

Personalización

A estas alturas ya debería estar bastante familiarizado con ***bash*** y es posible que haya notado algún comportamiento extraño. Por ejemplo, cuando inicia sesión en la consola, aparece un mensaje que se parece un poco a esto.

```
alan@darkstar:~$
```

Sin embargo, a veces verá un mensaje mucho menos útil como este.

```
bash-3.1$
```

La causa aquí es una variable de entorno especial que controla el indicador de ***bash***. Algunos shells se consideran shells de *"inicio de sesión"* y otros son shells *"interactivos"*, y ambos tipos leen diferentes archivos de configuración cuando se inician. Los shells de inicio de sesión leen /etc/profile y ~/.bash_profile cuando se ejecutan. Los shell interactivos leen en su lugar ~/.bashrc. Esto tiene algunas ventajas para los usuarios avanzados, pero es una molestia común para muchos usuarios nuevos que desean el mismo entorno cada vez que ejecutan ***bash*** y no importa la diferencia entre inicio de sesión y shells interactivos. Si esto se aplica a usted, simplemente edite su propio archivo ~ /.bashrc e incluya las siguientes líneas. (Para obtener más información sobre los diferentes archivos de configuración utilizados, lea la sección INVOCACIÓN de la página de manual de ***bash***).

```
# ~/.bashrc
. /etc/profile
. ~/.bash_profile
```

Cuando utilice lo anterior, todos sus shells de inicio de sesión e interactivos tendrán la misma configuración de entorno y se comportarán de manera idéntica. Ahora, cada vez que deseemos personalizar una configuración de shell, solo tendremos que editar ~/.bash_profile para los cambios específicos del usuario y /etc/profile para la configuración global. Comencemos configurando el prompt.

Los mensajes de ***bash*** vienen en todas las formas, colores y tamaños, y cada usuario tiene sus propias preferencias. Personalmente, prefiero mensajes cortos y simples que ocupen un mínimo de espacio, pero muchas veces he visto y usado mensajes de línea múltiple. Un amigo mío incluso incluyó ASCII -art en su mensaje de bash. Para cambiar su aviso, solo necesita cambiar su variable PS1. De forma predeterminada, Slackware intenta configurar su variable PS1 de esta manera:

```
darkstar:~$ echo $PS1
\u@\h:\w\$
```

Sí, esta pequeña pieza de figuras de aspecto gracioso controla su indicador de bash. Básicamente, todos los caracteres de la variable PS1 se incluyen en el indicador, a menos que sea un escape por a \, que le dice a ***bash*** que lo interprete. Hay muchas secuencias de escape diferentes y no podemos discutirlas todas, pero las explicaré. El primer "\u" se traduce al nombre de usuario del usuario actual. "\H" es el nombre de host de la máquina a la que está conectado el terminal. "\W" es el directorio de

trabajo actual, y "\$" muestra un signo # o un signo $, dependiendo de si el usuario actual es root o no. Una lista completa de todas las secuencias de escape rápido se encuentra en la página de manual de ***bash*** en la sección PROMPTING.

Ya que hemos pasado por todo este problema para discutir el indicador predeterminado, pensé que me tomaría un tiempo mostrarle un par de ejemplos y los valores de las variables PS1 necesarios para usarlos.

```
Wed Jan 14 12:08 AM
alan@raven:~$ echo $PS1
\d \@\n\u@\h:\w$
HOST: raven - JOBS: 0 - TTY: 3
alan@~/Desktop/sb_3.0:$ echo $PS1
HOST: \H - JOBS: \j - TTY: \l\n\u@\w:\$
```

Para obtener más información sobre cómo configurar el mensaje de bash, incluida la información sobre cómo configurar mensajes de colores, consulte /usr/doc/Linux-HOWTOs/Bash-Prompt-HOWTO. Al poco tiempo de leerlo, tendrá una idea de cuán poderosas pueden ser sus ordenes de ***bash***. ¡Una vez incluso tuve un aviso que me proporcionó información meteorológica actualizada, como la temperatura y la presión barométrica!

Navegación del capítulo

Capítulo anterior: Comandos básicos de shell

Siguiente capítulo: Control de procesos

Fuentes

- Fuente original: http://www.slackbook.org/beta
- Originalmente escrito por Alan Hicks, Chris Lumens, David Cantrell, Logan Johnson

— *M3rsy 2019/02/03 17:45 (UTC)*