Shells. Características Generales

Explica qué es un shell, los distintos tipos de shell y sus características principales.

Tabla de Contenidos

Autor: Lina García Cabrera & Francisco Martínez del Río

1 Introducción

Un *shell* o **intérprete de órdenes** es un ***programa que interpreta y ejecuta las órdenes conforme se proporcionan desde el terminal***. No se requiere ningún privilegio especial para ejecutar un shell; para el sistema operativo UNIX un shell es como cualquier otro programa.

Entre las características más comunes de un *shell* están

- la interpretación de **guiones de shell**,
- la **expansión de comodines** en nombres de archivos,
- la combinación de órdenes para formar **interconexiones**,
- la **recuperación de órdenes** previas,
- las construcciones **condicionales y los ciclos**, y
- las **variables** para crear abreviaturas.

Un **guión de shell** es un ***archivo que contiene secuencias de órdenes de shell***, igual que si se hubieran tecleado; los guiones de shell permiten adecuar el entorno de trabajo sin tener que realizar programación «real».

2 Tipos de Shells

Existen varios tipos de *shells*: el *shell* Bourne (**sh**), el *shell* Korn (**ksh**), el *shell* C extendido (**tcsh**), el *shell* C (**csh**) y el **shell bash**, los tres primeros se ven de forma muy breve.

- El *shell* Bourne forma parte de la séptima edición de UNIX y es el más viejo de los shells principales de UNIX.
- El *shell* C es el principal de BSD UNIX. La mayoría de los usuarios opinan que el shell C es más fácil de usar interactivamente, pero que el shell Bourne es más

sencillo para escribir guiones; por fortuna, es posible ejecutar un guión escrito para un shell estando en otro.
- El *shell* Korn, una extensión del shell Bourne, incluye la mayoría de las características mejores del shell C y sus propias adiciones. El shell Korn no se incluye en varias distribuciones de UNIX, pero si puede obtenerlo, quizá sea su mejor opción.

Con la popularización de Linux, el **shell bash** ha obtenido un gran protagonismo. Se trata de una versión mejorada y gratuita del **shell Bourne**, de hecho su nombre es acrónimo de ***Bourne-again shell***.

Como algunos recursos de shell son intrínsecos al estilo de trabajo de UNIX, y además, comunes a todos los shells, ya se han ido presentando en los temas anteriores. Ejemplos de estos recursos son las interconexiones y las formas de redireccionamiento más sencillas.

3 Guiones Shells

Los **guiones de shell**, también llamados procedimientos de shell o simplemente ***scripts***, permiten adecuar el entorno añadiendo órdenes propias. Aunque la escritura de un guión de shell requiere un poco de programación, es mucho más fácil que escribir un programa en C.

Los guiones de *shell* cortos son fáciles de crear e instalar.

> Por ejemplo, suponga que desea definir una orden **dir** que produzca un listado de directorios con un formato más agradable que el que se obtiene por defecto. Supondremos que usted usa un subdirectorio **bin** que cuelga directamente de su directorio base para almacenar su colección personal de programas y guiones de shell, y que este directorio está en su trayectoria de búsqueda (recuerde la orden ***path***). Puede crear la orden dir si teclea
>
> ```
> $ cat > $HOME/bin/dir
> ls -cF $@
> ctrl-d
> $ chmod +x $HOME/bin/dir
> ```
>
> la línea **cat** copia la segunda línea en el archivo *dir*; la línea **ls** define el significado de **dir**; y la línea **chmod** hace que el archivo sea ejecutable (condición necesaria para ejecutar un guión de shell. Los guiones de shell también deben tener permiso de lectura, pero lo más común es que un archivo de nueva creación tenga este permiso por omisión). Este guión de una línea sigue los convencionalismos del **shell Bourne**.
>
> Después de crear esta orden, puede obtener el listado mejorado si teclea **dir** seguido por un conjunto de nombres de archivos y directorios. Los nombres pueden incluir comodines. Esta orden también funciona si solamente teclea **dir**; en este caso obtendrá el contenido del directorio actual. Una versión más elaborada de **dir** podría, por ejemplo, reconocer diversas opciones.

3.1 Interpretación de los Guiones de Shell

Cuando un shell detecta una orden que no es intrínseca, es decir, que **el shell no ejecuta directamente**, llama al sistema operativo para que la ejecute. La orden puede ser un programa compilado o un guión de shell. En el segundo caso, el sistema operativo tiene que seleccionar un shell para ejecutar el guión.

La forma en que el sistema operativo realiza su selección depende del sistema. Por omisión se elige una versión del **shell Bourne** o, en algunas ocasiones, el shell Korn (que es casi completamente compatible con aquel). La compatibilidad con el shell Bourne es esencial, porque históricamente éste era el único disponible en la séptima edición;

varios guiones de uso común suponen, sin indicación explícita, que son interpretados por el shell Bourne.

> La mayoría de los sistemas se adhieren al convencionalismo de BSD UNIX que especifica que **la primera línea de un guión de shell** tiene la forma
>
> **`#! shell`**
>
> donde *shell* es la **trayectoria absoluta del shell** que se usa para interpretar el guión. El espacio en blanco después de '**#!**' es opcional.

> Por ejemplo, si se escribe un guión usando el conjunto de órdenes del **shell bash**, éste deberá comenzar con la línea:
>
> **`#! /bin/bash`**

De hecho, usted puede hacer que **un guión sea interpretado por cualquier programa**.

> Por ejemplo, suponga que emite una orden que nombra a un archivo ejecutable, cuya primera línea es
>
> **`#! /usr/sibila/leer.entrañas`**
>
> Entonces, el programa **leer.entrañas** del directorio base de *sibila* asume el control, usando como entrada el archivo ejecutable. Por supuesto, **leer.entrañas** debe tener la inteligencia suficiente para reconocer la primera línea como comentario e ignorarla.

Algunos sistemas que no respetan el convencionalismo '#!' ofrecen otra manera de etiquetar los guiones de shell:

- si el primer carácter es '**:**', se considera como un guión del shell Bourne;
- si el primer carácter es '**#**', se considera como un guión del shell bash.

Todos los shells comunes proporcionan una opción **-v** para presentar cada línea de entrada conforme se lee y una opción **-x** para presentar las órdenes cuando se ejecutan. Estas opciones son muy útiles para **depurar guiones**.

A continuación describimos las características generales de los distintos shells. Dedicaremos un mayor espacio al **shell bash** . Este tema y los dos siguientes le enseñarán mediante ejemplos y propuestas las peculiaridades de los shell y cómo programar en ellos.

4 El Shell Bourne (sh)

El shell Bourne, sh, escrito por Steve Bourne en 1979, es parte de la séptima edición de UNIX y el primero de los shells principales. Los shells más recientes son más fáciles de usar, porque ofrecen recursos de los que carece el shell Bourne, como:

- la edición de las líneas de órdenes,
- la recuperación de órdenes emitidas previamente, y
- los alias para las órdenes de uso común.

No obstante, muchos usuarios de UNIX prefieren el shell Bourne para uso interactivo. Casi todos los guiones siguen los convencionalismos del shell Bourne.

Como **sh** era el único shell importante cuando se introdujo, la documentación de UNIX muchas veces hace referencia "al shell", cuando en realidad quiere decir **sh**. Muchas de estas referencias sólo se aplican a **sh**, y no a los demás shells. Lo que se dice en este tema sobre "el shell" se aplica a todos los shells principales; si un enunciado sólo es aplicable a sh, lo indicaremos de forma explícita.

Entre los recursos importantes que ofrece **sh** están los siguientes:

- Operadores para la ejecución en segundo plano, o ejecución condicional de

órdenes.
- Enunciados para repetir la ejecución de órdenes, incluida la iteración a lo largo de una secuencia de valores que pueden asignarse a una variable de iteración.
- Variables sustituibles, tanto nombradas como numeradas. Las variables numeradas, también conocidas como parámetros o parámetros de posición, contienen los argumentos de una orden.
- Exportación de variables específicas a un proceso hijo.
- Tres formas de entrecomillado.
- Ejecución de órdenes en subshells.
- Notificación automática de la llegada de correo.
- Inclusión de datos de entrada para una orden en un guión de shell como parte del guión.
- Captura de señales, y ejecución de órdenes específicas cuando ocurre una señal determinada.
- Ejecución de órdenes en archivos de iniciación antes de leer cualquier entrada. Estos archivos de iniciación pueden servir para adecuar **sh** a las necesidades propias.

La versión de **sh** que se describe aquí es la de System V. La versión que existe en otros sistemas, en particular en BSD UNIX, es una anterior a la séptima edición que carece de algunos de los recursos de la versión de System V. En estos sistemas, es probable que la versión más nueva exista con el nombre **sh5**. No tiene sentido usar la versión antigua si está disponible la más reciente.

5 El Shell Bash (bash)

El *shell* bash, disponible a través de la orden **bash**, se desarrolló como parte del proyecto GNU. Está basado en la shell **sh** de Unix, y por lo tanto se puede ver como un superconjunto de éste.

Algunas diferencias que podemos encontrar entre ambos son:

- La posibilidad de recuperar órdenes previas mediante un mecanismo de “historia”.
- Facilidades de edición de la línea de comandos.
- La capacidad de conmutar entre procesos y controlar su avance (control de trabajos).
- Formas más flexibles de sustitución de variables.
- Operadores adicionales, de sintaxis similar a la de C.
- Alias para órdenes de uso frecuente, sin tener que usar guiones.

Varias de las órdenes de Bourne again sh **bash** se comportan igual que sus contrapartes de **sh**. Es por ello que en nuestra descripción de **bash** haremos algunas referencias a **sh** para presentar detalles y ejemplos.

En la siguiente tabla se muestra una breve comparativa entre similitudes y diferencias de los shells más populares:

Tabla 1 Algunas diferencias entre los principales shells

	sh	csh	tcsh	ksh	bash
histórico de órdenes	No	Sí	Sí	Sí	Sí
alias	No	Sí	Sí	Sí	Sí
shell scripts	Sí	Sí	Sí	Sí	Sí
autocompletado de archivos	No	Sí	Sí	Sí	Sí
edición de línea de órdenes	No	No	No	Sí	Sí
control de trabajos	No	Sí	Sí	Sí	Sí

6 El Shell Korn (ksh)

El *shell* Korn, **ksh**, ofrece una síntesis de las características de los shells Bourne y C, además de otras propias. Fue desarrollado por David Korn, de ATT Bell Laboratories, en 1982, presentando versiones mejoradas en 1986 y 1988. Se incluye como característica estándar de la versión 4 de System V y otros sistemas; también se puede comprar por separado.

El shell Korn sigue de cerca los convencionalismos del *shell* Bourne, y casi todos los guiones escritos para el primero funcionan con el segundo.

Las características principales que se adoptaron del *shell* C son:

- Listas históricas para la recuperación de órdenes previas.
- Control de trabajos, con la capacidad para pasar trabajos específicos al primer o segundo plano.
- Alias para los nombres de órdenes.
- Empleo de '~' para representar el directorio base del usuario, o, al combinarse con un nombre de usuario, el de otro usuario.
- Capacidad para calcular expresiones numéricas generales, y asignar el resultado a una variable.

Algunas de las características nuevas de **ksh** son:

- Edición interactiva de la línea de órdenes, incluida la complementacion de nombres de fichero con las mismas características de **bash** y la posibilidad de editar la lista histórica.
- Mejores definiciones de funciones, que ofrecen variables locales y permiten escribir funciones recursivas.
- Comparación extendida de patrones para nombres de ficheros y otras construcciones, parecidas a la de **egrep**.
- Capacidad para extraer la porción de una cadena especificada por un patrón.
- Capacidad para cambiar fácilmente de un directorio a otro.

7 El Shell C mejorado (tcsh)

El shell **tcsh** es una versión mejorada del shell C, que ha adquirido mucha popularidad. Algunos de los recursos adicionales que ofrece son:

- Capacidad para editar la línea de órdenes interactivamente.
- Llamada sencilla de órdenes ejecutadas con anterioridad, las cuales se pueden editar.
- Complementacion interactiva de nombres de ficheros y órdenes.
- Consulta de la documentación sobre una orden cuando es tecleada.
- Capacidad para programar la ejecución periódica de una orden.
- Marcas de la hora en la lista histórica.