Introducción a la Programación en Shell Scripting

El *shell* es un intérprete de línea de comandos que proporciona una interfaz entre el usuario y el núcleo del sistema operativo (*kernel*). Permite al usuario ingresar una serie de instrucciones a través de la entrada estándar (*stdin*), las cuales el *shell* interpreta y ejecuta mediante el sistema operativo, generando la salida correspondiente que se mostrará en la pantalla o salida estándar (*stdout*). De hecho, hasta que la ejecución del comando no llega a su fin, el *shell* no volverá a permitir que el usuario pueda introducir otro comando. Aunque la ejecución generalmente es rápida, en numerosas ocasiones se requiere ejecutar un número elevado de comandos, ya sea de forma repetitiva o que tomen tiempo en completarse. En tales casos, parece necesario encontrar una manera de introducir todas estas órdenes de forma única y dejar que se ejecuten sin intervención adicional. Los scripts de shell, conocidos como ***shells scripts***, resuelven estos problemas permitiendo la automatización y combinación de comandos y estructuras lógicas de control, creando programas personalizados que ejecutan una secuencia de instrucciones almacenadas en un archivo de texto plano.

¿Por qué usar los shells scripts?

El uso de *shells* scripts ofrece numerosas ventajas, tales como:

- **Automatización:** La capacidad de automatizar tareas repetitivas ahorra tiempo y reduce la posibilidad de errores humanos.
- **Estructuración Modular:** Los *scripts* proporcionan una manera estructurada y modular de organizar las instrucciones, facilitando la comprensión y el mantenimiento del código.
- **Manejo de Argumentos:** La capacidad de introducir argumentos (valores dinámicos) a los comandos dentro de un *script* permite personalizar la ejecución del programa.
- **Simplificación de Comandos Complejos:** La capacidad de encapsular comandos complejos en un solo *script* facilita la ejecución y comprensión de tareas complejas.
- **Reutilización:** Un *script* puede ejecutarse repetidamente por cualquier persona, siguiendo el principio "construye una vez y ejecuta muchas veces".

Tipos de Shell

El universo de *shells* en sistemas UNIX es diverso, y la elección del *shell* depende en gran medida de las preferencias y necesidades del usuario. Los shells se clasifican principalmente en dos categorías: basado en Bourne y basados en C.

Shells basados en Bourne:

- **Bourne Shell (sh):** Creado por Steven Bourne, es considerada como la primera *shell* tradicional de los sistemas UNIX y generalmente se asigna por defecto al superusuario (root).
- **Korn Shell (ksh):** Desarrollado por David Korn, añade características como historial de órdenes, edición en línea y funcionalidades adicionales de programación.
- **Bourne Again Shell (bash):** Desarrollado por el Proyecto GNU. Actualmente, es el *shell* de *facto* de la gran mayoría de las distribuciones Linux. Es compatible con **sh** y presenta características de **ksh** y **csh**, como la edición de línea de comandos, historial ilimitado, control de trabajos y procesos, funciones y alias, cálculos aritméticos con números enteros, etc.

Shells basados en C:

- **C Shell (csh):** Desarrollado por Bill Joy en la Universidad de Berkeley, es el más utilizado en sistemas BSD (Berkeley Software Distribution). Ofrece características útiles para programadores que trabajan en C, con mejoras en la edición de línea de comandos y el historial, aunque no es compatible con **sh**.
- **Tenex C Shell (tcsh):** Un superconjunto de **csh** que mejora la velocidad y soluciona muchos problemas de **csh**.

Todas estas *shells* y otras menos habituales comparten características comunes como la forma para redireccionar la salida estándar con los caracteres **>** y **>>** o la entrada estándar con el carácter **<**, así como el uso del carácter **|** (pipe) para encadenar comandos o el uso de los operadores AND **(&&)** y OR (**||**), que permiten controlar la ejecución de diferentes instrucciones. Sin embargo, difieren en aspectos como la asignación de variables, variables de entorno predefinidas, la creación de aliases o incluso el uso de comandos específicos.

Shell del sistema

Explorar los *shells* disponibles en un entorno de trabajo Linux es crucial para personalizar la experiencia del usuario. Para conocer los *shells* disponibles en el entorno de trabajo Linux, se debe consultar el archivo **/etc/shells**:

```
[usuario@maquina ~]$ more /etc/shells

/bin/sh
/bin/bash
/usr/bin/sh
/usr/bin/bash
/usr/bin/zsh
/bin/zsh
```

Tras la instalación de un sistema UNIX, se establece un *shell* por defecto. De hecho, cuando se accede a la terminal en el ambiente de trabajo y se inicia el *shell*, este asigna a una variable de entorno global el valor del *shell* actual, que se puede consultar utilizando el comando `**echo $SHELL**`:

```
[usuario@maquina ~]$ echo $SHELL

/bin/bash
```

La salida anterior nos informa que el valor de la *shell* actual en el ambiente de trabajo definido es `/bin/bash`, que hace referencia a la **Bash shel**l. Podemos consultar la versión instalada para esta instancia de bash en la salida estándar con el comando `**bash**` seguido de la opción `--version`.

```
[usuario@maquina ~]$ bash --version

GNU bash, version 4.2.46(2)-release (x86_64-koji-linux-gnu)
Copyright (C) 2011 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law
```

Otra forma de consultar la versión de bash instalada sería consultar directamente la variable de entorno global `**BASH_VERSION**`. El valor de dicha variable se puede consultar utilizando el comando `**echo**`:

```
[usuario@maquina ~]$ echo $BASH_VERSION

4.2.46(2)-release
```

Recuerda que otro tipo de información puede ser consultada para conocer más especificidades sobre nuestra shell; para ello consulta su manual en línea con el comando `**man**`:

```
[usuario@maquina ~]$ man bash
```

La página del manual divide la información del comando en secciones. Las páginas del manual se pueden recorrer presionando la barra espaciadora o usando las teclas de flecha para desplazarse hacia adelante y hacia atrás a través del texto. Para finalizar la visualización del manual del comando consultado deberás presionar la tecla `q` (quit).

La capacidad de cambiar el *shell* por defecto y de especificar el *shell* deseado en un *script* proporciona flexibilidad y adaptabilidad al entorno de trabajo. Específicamente, se puede cambiar el *shell* por defecto empleando el comando `**chsh**` (change s*hell*). Es importante reiniciar la sesión para que los cambios surtan efecto.

Este cambio se puede realizar de forma interactiva (si no añadimos ninguna opción) o si le añadimos la opción `-s` se realizará de forma directa, indicándole el cambio a otra *shell* validada en la lista de *shells* disponibles.

```
[usuario@maquina ~]$ chsh -s bin/bash
```

Además, en los *shell* scripts, se puede especificar el nuevo *shell* deseado directamente en la primera línea del script con `#!` (shebang). Por ejemplo, un usuario bash podría utilizar el intérprete csh en un script que comenzara por:

```
#! bin/csh
```

En lo sucesivo, utilizaremos la palabra *shell* para referirnos específicamente a la Bash shell.

BASH Shell

Esta asignatura se va a focalizar en **BASH,** ya que es el intérprete de comandos más utilizado en los sistemas GNU/Linux. Además de ser un intérprete de comandos, Bash ofrece un completo lenguaje de programación estructurado con una amplia variedad de funciones internas.

Las principales características del intérprete BASH son las siguientes:

- **Ejecución secuencial o paralela de órdenes:** Permite la ejecución de comandos de forma secuencial o paralela, ofreciendo flexibilidad en la gestión de procesos.
- **Disposición de distintos tipos de redirecciones de entrada y salida:** Facilita el control y filtrado de información mediante la manipulación de la entrada y salida de los comandos.
- **Control del entorno de los procesos:** Proporciona herramientas para supervisar y gestionar el entorno en el que se ejecutan los procesos.
- **Ejecución interactiva:** Permite la ejecución interactiva de comandos, aceptando entradas desde archivos, teclado, entre otras fuentes.
- **Órdenes internas:** Ofrece comandos internos para la manipulación directa del intérprete y del entorno de operación, proporcionando mayor control y personalización.
- **Variables y Estructuras de Control:** Incluye una amplia gama de elementos que facilitan la programación, desde variables y operadores hasta estructuras de control de flujo y funciones.

- **Control de Procesos:** Permite gestionar procesos tanto en primer plano *(foreground)*, donde interactúan directamente con el usuario, como en el segundo plano *(background)*, permitiendo la ejecución sin bloquear la terminal.
- **Creación de alias:** Facilita la creación de alias, palabras que agrupan uno o más comandos, simplificando la ejecución de secuencias de instrucciones complejas.

En resumen, Bash no solo sirve como un intérprete de comandos estándar, sino que también proporciona un entorno robusto para el desarrollo de scripts y la automatización de tareas en sistemas operativos basados en GNU/Linux. Su versatilidad y potencia lo convierten en una herramienta fundamental para administradores de sistemas y usuarios avanzados.