Introducción a los Sistemas Operativos

# Práctica 2

1. ¿Qué es el Shell Scripting? ¿A qué tipos de tareas están orientados los script? ¿Los scripts deben compilarse? ¿Por qué?

El Shell Scripting es la creación de scripts (programas pequeños escritos en un lenguaje de programación interpretado) que se ejecutan en un shell o intérprete de comandos. El objetivo de un script es automatizar tareas repetitivas o complejas que normalmente se realizan manualmente en la línea de comandos.

Los scripts pueden estar orientados a una amplia gama de tareas, desde la administración del sistema hasta el procesamiento de datos y la automatización de tareas de desarrollo. Por ejemplo, un script puede automatizar la copia de archivos, la eliminación de archivos antiguos, la instalación de paquetes, el procesamiento de datos, la manipulación de archivos de configuración y mucho más.

A diferencia de los lenguajes de programación compilados, los scripts no necesitan ser compilados antes de su ejecución. En cambio, se ejecutan directamente por el intérprete de comandos. Esto ofrece varias ventajas, como la capacidad de depurar rápidamente y modificar el script sobre la marcha. Además, los scripts son portables y se pueden ejecutar en diferentes sistemas operativos sin necesidad de realizar cambios de código.

2. Investigar la funcionalidad de los comandos echo y read

(a) ¿Como se indican los comentarios dentro de un script?

El comando "echo" es un comando utilizado en Shell Scripting que se utiliza para imprimir texto en la pantalla o en un archivo de salida. Puede imprimir una cadena de texto, variables, comodines y cualquier otro tipo de contenido. Por ejemplo:

#!/bin/bash

echo "Hola, este es un ejemplo de texto impreso en pantalla"

El comando "read" se utiliza en Shell Scripting para leer una entrada de texto del usuario y asignarla a una variable. Esto es útil cuando se necesita que el usuario proporcione información al script. Por ejemplo:

#!/bin/bash

echo "¿Cuál es tu nombre?"

read nombre

echo "Hola, $nombre. Bienvenido!"

Para indicar un comentario dentro de un script de Shell, se utiliza el carácter "#". Todo lo que sigue después del carácter "#" en una línea se considera un comentario y se ignora durante la ejecución del script. Por ejemplo:

#!/bin/bash

# Este es un comentario

echo "Hola, mundo" # Este es otro comentario

(b) ¿Cómo se declaran y se hace referencia a variables dentro de un script?

En Shell Scripting, se pueden declarar variables utilizando un nombre de variable y un valor asignado a esa variable. No es necesario declarar el tipo de variable, ya que las variables son de tipo dinámico. Se pueden asignar valores de texto o numéricos a las variables.

Para hacer referencia a una variable en un script, se puede utilizar el nombre de la variable precedido por el signo de dólar "$". Por ejemplo:

#!/bin/bash

nombre="Juan"

echo "Hola, $nombre"

También es posible utilizar el valor de una variable en una operación matemática. Para hacerlo, es necesario utilizar el signo de dólar y paréntesis alrededor del nombre de la variable. Por ejemplo:

#!/bin/bash

numero1=10

numero2=5

resultado=$(($numero1 + $numero2))

echo "El resultado es $resultado"

3. Crear dentro del directorio personal del usuario logueado un directorio llamado practica-shell-script y dentro de él un archivo llamado mostrar.sh cuyo contenido sea el siguiente:

Texto

Descripción generada automáticamente

(a) Asignar al archivo creado los permisos necesarios de manera que pueda ejecutarlo

chmod +x archivo.sh

(b) Ejecutar el archivo creado de la siguiente manera: ./mostrar

(c) ¿Qué resultado visualiza?

(d) Las backquotes (`) entre el comando whoami ilustran el uso de la sustitución de comandos. ¿Qué significa esto?

(e) Realizar modificaciones al script anteriormente creado de manera de poder mostrar distintos resultados (cuál es su directorio personal, el contenido de un directorio en particular, el espacio libre en disco, etc.). Pida que se introduzcan por teclado (entrada estándar) otros datos

4. Parametrización: ¿Cómo se acceden a los parámetros enviados al script al momento de su invocación? ¿Qué información contienen las variables $#, $\*, $? Y $HOME dentro de un script?

Los parámetros enviados al script al momento de su invocación se pueden acceder a través de las variables especiales $1, $2, $3, ..., $n, donde "n" es el número de parámetros que se pasan al script. La variable $1 contiene el primer parámetro, $2 contiene el segundo, y así sucesivamente.

La variable $# contiene el número total de parámetros pasados al script. La variable $\* contiene todos los parámetros pasados al script como una única cadena de texto. Es decir, $\* incluye todos los parámetros separados por el primer caracter en la variable de entorno IFS (Internal Field Separator), que por defecto es un espacio en blanco. La variable $? contiene el estado de salida del último comando ejecutado. Si el comando se ejecutó correctamente, su valor será 0. De lo contrario, contendrá un valor diferente de 0 que indica el código de error del comando.

La variable $HOME contiene la ruta del directorio home del usuario actual. Es importante destacar que estas variables sólo están disponibles dentro del script y no afectan el entorno general del sistema.

5. ¿Cuál es la funcionalidad de comando exit? ¿Qué valores recibe como parámetro y cuál es su significado?

El comando "exit" en un script de shell se utiliza para salir del script y devolver un estado de salida específico. El estado de salida se utiliza para indicar si el script se ejecutó correctamente o no, y puede ser utilizado por otros scripts o programas que llamen al script para determinar si se debe tomar alguna acción adicional.

El comando "exit" puede recibir un valor numérico como parámetro, que es el estado de salida que se devuelve al sistema operativo. El valor por defecto es 0, que indica que el script se ejecutó correctamente sin errores. Si se produce algún error o excepción durante la ejecución del script, se puede devolver un valor diferente de 0 para indicar que algo salió mal.

Por ejemplo, si se tiene un script que realiza una operación y se desea devolver un estado de salida de 1 si la operación falla, se puede utilizar el siguiente comando:

if [ $resultado -ne 0 ]

then

echo "La operación falló"

exit 1

fi

En este caso, si la variable $resultado es diferente de 0, se imprime un mensaje de error y se sale del script con un estado de salida de 1. El valor de 1 se puede utilizar más adelante por otros scripts o programas que llamen a este script para tomar una acción adicional si es necesario.

Es importante destacar que el estado de salida debe ser un número entero entre 0 y 255, ya que estos son los valores permitidos por el sistema operativo.

6. El comando expr permite la evaluación de expresiones. Su sintaxis es: expr arg1 op arg2, donde arg1 y arg2 representan argumentos y op la operación de la expresión. Investigar qué tipo de operaciones se pueden utilizar.