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

Shell scripting es la técnica (habilidad / destreza) de diseñar y crear Script (archivo de automatización de tareas) mediante un Shell (preferiblemente) de un Sistema Operativo, o un Editor de Texto (Gráfico o Terminal). Este es un tipo de lenguaje de programación que generalmente es interpretado. Se orienta a la automatización de tareas y a aplicaciones interactivas o con interfaz gráficas. No se compilan ya que son programas interpretados

1. Investigue la funcionalidad de los comandos echo y read

Echo es un comando para la impresión de un texto en pantalla

Read lee su entrada estándar y asigna las palabras leídas en la(s) variable(s) cuyo nombre se pasa como argumento

* 1. ¿Cómo se indican los comentarios dentro de un script?

Se indican usando #

* 1. ¿Cómo se declaran y se hace referencia a variables dentro de un script?

Para declararlas haríamos NOMBRE="pepe".Para hacer referencia usamos el $, entonces seria echo $NOMBRE

1. Hecho en pc
2. 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?

Para acceder a los argumentos usamos $1,$2.. $n, dependiendo de la cantidad de argumentos que pasemos.

La variable $# contiene la cantidad de argumentos recibidos, $\* contiene la lista de todos los argumentos, $HOME es una variable que contiene la ruta /home la cual es la que contiene todos mis documentos

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

El comando exit es usado para terminar un script. Puede devolver diferentes parámetros donde cada uno tendrá un significado

* 0 indica que el script se ejecutó se form exitosa
* Un valor distinto indica un código de error

Puedo consultar el exit status imprimiendo la variable $?

1. 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 que tipo de operaciones se pueden utilizar

* Igualdad: arg1 -eq arg2
* Desigualdad: arg1 -ne arg2
* Mayor: arg1 -gt arg2
* Mayor o igual: arg1 -ge arg2
* Menor: arg1 -lt arg2
* Menor o igual: arg1 -le arg2

1. El comando “test expresión” permite evaluar expresiones y generar un valor de retorno, true o false. Este comando puede ser reemplazado por el uso de corchetes de la siguiente manera [ expresión ]. Investigar qué tipo de expresiones pueden ser usadas con el comando test. Tenga en cuenta operaciones para: evaluación de archivos, evaluación de cadenas de caracteres y evaluaciones numéricas.

El comando test permite efectuar una serie de pruebas sobre los archivos, las cadenas de caracteres, los valores aritméticos y el entorno de usuario. El comando test posee dos sintaxis: test expresión y [ expresión ], donde "expresión" representa el test que se debe efectuar.

* -b fichero: cierto si el fichero existe y es un dispositivo de bloques
* -c fichero: cierto si el fichero existe y es un dispositivo de caracteres.
* -d fichero: cierto si el fichero existe y es un directorio.
* -e fichero: cierto si el fichero existe.
* -f fichero: cierto si el fichero existe y es un fichero normal.
* -g fichero: cierto si el fichero existe y tiene el bit de grupo activado.
* -k fichero: cierto si el fichero tiene el bit de sticky activado.
* -L fichero: cierto si el fichero existe y es un enlace simbólico.
* -p fichero: cierto si el fichero existe y es una tubería nombrada.
* -r fichero: cierto si el fichero existe y es legible.
* -s fichero: cierto si el fichero existe y su tamaño es mayor que cero.
* -S fichero: cierto si el fichero existe y es un socket.
* -t [df]: cierto si df está abierto en un terminal. Si fd es omitido, se toma 1 (salida
* estándar) por defecto.
* -u fichero: cierto si el fichero existe y tiene el bit de usuario activo.
* -w fichero: cierto si el fichero existe y es escribible.
* -x fichero: cierto si el fichero existe y es ejecutable.
* -O fichero: cierto si el fichero existe y es propiedad del identificador efectivo del usuario.
* -G fichero: cierto si el fichero existe y es propiedad del identificador efectivo del grupo.
* fichero1 -nt fichero2: cierto si fichero1 es más reciente (en base a la fecha de modificación) que fichero2.
* fichero1 -ot fichero2: cierto si fichero1 es más antiguo que fichero2.
* fichero1 -ef fichero2: cierto si fichero1 y fichero2 tienen el mismo número de dispositivo y de nodo-i.
* -z cadena: cierto si la longitud de cadena es cero.
* -n cadena: cierto si la longitud de cadena no es cero.
* cadena1 = cadena2: cierto si las cadenas son iguales.
* cadena1 != cadena2: cierto si las cadenas no son iguales.
* ! expr: cierto si expr es falsa.
* expr1 -a expr2: cierto si expr1 y expr2 son ciertas
* expr1 -o expr2: si expr1 o expr2 son ciertas.
* arg1 OP arg2: OP es uno de los siguientes valores: -eq, -ne, -lt, -le, -gt, o -ge. Estos

operadores binarios aritméticos devuelve cierto si arg1 es igual, distinto, menor

que, menor o igual que, mayor que, o mayor o igual que arg2, respectivamente. arg1

y arg2 pueden ser enteros positivos, enteros negativos, o la expresión especial -l

cadena, la cual significa la longitud de cadena

1. Estructuras de control. Investigue la sintaxis de las siguientes estructuras de control incluidas en shell scripting:

* **if**

if [ condition ]

then

block

fi

* **case**

case $variable in

"valor 1")

block

;;

"valor 2")

block

;;

\*) #bloque usado en caso que no haya entrado en ninguno

block

;;

esac

* **while**

while [ condition ] #mientras se cumpla la condición

do

block

done

* **for**: puedo llegar a tener dos casos

#primer caso, estilo C

for ((i=0; i < 10; i++))

do

block

done

#segundo caso con lista de valores (foreach)

for i in value1 value2 value3 valueN;

do

block

done

* **select**

select variable in opcion1 opcion2 opcion3

do

#en $variable está el valor elegido

block

done

1. ¿Qué acciones realizan las sentencias break y continue dentro de un bucle? ¿Qué parámetros reciben?

**break:** usado para terminar la ejecución del loop entero después de completar la ejecución de todas las líneas de código anteriores a la sentencia break, una vez llegada a la sentencia break, saldrá del loop y ejecutará el código siguiente fuera del loop. Si indico la sentencia break n donde n es un número, n es la cantidad de bucles de los que saldré, por default es 1

**continue:**  es similar a la sentencia break pero lo que hará será salir de la interacción actual en vez de salir del loop entero. Podría servir para cuando un error ocurre pero quiero intentar ejecutar la siguiente iteración del loop. Si indico la sentencia continue n donde n es un número, n es la cantidad de loops de los que saldré, por default es 1

1. ¿Qué tipo de variables existen? ¿Es shell script fuertemente tipado? ¿Se pueden definir arreglos? ¿Cómo?

Existen las variables globales y locales. No es fuertemente tipado.

Para definir arreglos tengo dos formas

arreglo\_a=() #Se crea vacío

arreglo\_b=(1 2 3 5 8 13 21) #Inicializado

1. ¿Pueden definirse funciones dentro de un script? ¿Cómo? ¿Cómo se maneja el pasaje de parámetros de una función a la otra?

Si, puedo definir funciones dentro de un script. Puedo declararlo de dos formas function nombre {block} o como nombre() {block}.

Las variables de entorno son heredadas por los procesos hijos. Para exponer una variable global los procesos hijos se usa el comando export

export VARIABLE\_GLOBAL="Mi var global"

comando # comando verá entre sus variables de # entorno a VARIABLE\_GLOBAL

1. Evaluación de expresiones
   1. Realizar un script que le solicite al usuario 2 números, los lea de la entrada Standard

e imprima la multiplicación, suma, resta y cuál es el mayor de los números leídos

* 1. Modificar el script creado en el inciso anterior para que los números sean recibidos

como parámetros. El script debe controlar que los dos parámetros sean enviados

* 1. Realizar una calculadora que ejecute las 4 operaciones básicas: +, - ,\*, %. Esta calculadora debe funcionar recibiendo la operación y los números como parámetros

1. Uso de estructuras de control
   1. Realizar un script que visualice por la pantalla los números del 1 al 100
   2. Realizar un script que muestre 3 opciones al usuario: Listar, DondeEstoy y QuienEsta. Según la opción elegida se le debe mostrar:
      1. Listar: lista el contenido del directorio actual
      2. DondeEstoy: muestra el directorio donde me encuentro ubicado
      3. QuienEsta: muestra los usuarios conectados al sistema
   3. Crear un script que reciba como parámetro el nombre de un archivo e informe si el mismo existe o no, y en caso de afirmativo indique si es un directorio o un archivo. En caso de que no exista el archivo/ directorio cree un directorio con el nombre recibido como parámetro
2. Renombrando Archivos: haga un script que renombre solo archivos de un directorio pasado como parámetro agregándole una CADENA, contemplando las opciones:

* “-a CADENA”: renombra el fichero concatenando CADENA al final del nombre del archivo
* “-b CADENA”: renombra el fichero concatenado CADENA al principio del nombre del archivo

<https://ryanstutorials.net/bash-scripting-tutorial/bash-if-statements.php>