Practica 3- Explicación

jueves, 15 de septiembre de 2022 09:50

SHELL

Shell es un intérprete de comandos interactivo que en los sistemas operativos *nix es configurable. Proveen estructuras de control que permiten programar Shell scripts

OBJETIVOS

- automatizar tareas
- hacer aplicaciones interactivas
- hacer aplicaciones con interfaz grafica

existen muchas shells. Sus diferencias consisten principalmente en sintaxis. A continuación se listan las más utilizadas:

- sh: shell por defecto en Unix
- bash: cómoda, instalada por defecto en la mayoría de las distribuciones
- dash: eficiente, parcialmente compatible con bash
- csh: sintaxis incompatible con bash/dash

ALGUNOS COMANDOS ÚTILES

OPERACIÓN	SINTAXIS
imprimir el contenido de un archivo	cat archivo
imprimir el texto	echo "Hola mundo"
leer una línea desde entrada estándar en la variable var	read var
quedarme con la primer columna de un texto separado por: desde entrada estándar	cut -d: -fi
contar la cantidad de líneas que se leen desde entrada estándar	wc -l
buscar todos los archivos que contengan la cadena **pepe** en el directorio /tmp	grep pepe /tmp/*
buscar todos los archivos dentro del *home* del usuario cuyo nombre termine en .doc	find $HOME -name "*.doc"
buscar todos los archivos dentro del directorio actual que sean enlaces simbólicos	find -type l
EMPAQUETADO: une varios archivos en uno solo. El comando tar puede invocar a gzip por nosotros mediante el argumento z	tar -cvf archivo.tar archivo1 archivo2 archivo3
COMPRESIÓN: reduce el tamaño de un archivo	gzip archivo.tar #genera archivo.tar.gz comprimido ////////// gzip -d archivo.tar.gx #descomprime archivo.tar

REDIRECCIONES Y PIPES

" > "	redirección destructiva. Si el archivo no existe se crea y si existe, se sobrescribe
>>	redirección no destructiva. SI el archivo no existe se crea y si existe, agrego al final

los *pipes* conectan la salida de un comando con la entrada de otro

VARIABLES

a tener en cuenta

- bash soporta string y arrays
- los nombres son case sensitive
- para crearla haré `NOMBRE= "pepe" #sin espacios alrededor del =`
- para accederla se usa `$: echo $NOMBRE`

ARREGLOS

- para crearlos hay dos maneras
  - `arreglo_a=() #se crea varío`

    - `arreglo_b=(1 2 3 4 5 9 10) #inicializado`
- asignación de un valor en una posición concreta `arreglo_b[2]= spam`
- acceso a un valor del arreglo (en esta caso las llaves no son opcionales
    - `echo ${arreglo_b[2]}`
    - `copia= ${arreglo_b[2]}`
- acceso a todos los valores del arreglo `echo ${arreglo[@]}`
- tamaño del arreglo `${#arreglo[@]}`
- borrado de un elemento `unset arreglo[2]`
- los indices de los arreglos empiezan en 0

COMILLAS
no hacen falta a menos que

- el string tenga espacios
- que sea una variable cuyo contenido pueda tener espacios
- son importantes en las condiciones de los if, while, etc.

SCRIPT

- puedo crearlo en cualquier editor de texto
- debo indicar `#!  /bin/bash`, especifica el intérprete que usaré para ejecutar el script

ESTRUCTURAS DE CONTROL

Decisión:

```
if [ condition ]
then
        block
fi
```

Selección:

```
case $variable in
    "valor 1")
            block
    ;;
    "valor 2")
            block
    ;;
    *)
            block
    ;;
esac
```

while

```
while [ condition ] #Mientras se cumpla la condición
do
        block
done
```

until

```
until [ condition ] #Mientras NO se cumpla la condición
do
        block
done
```

EVALUACIONES LÓGICAS

OPERADOR	CON STRINGS	CON NÚMEROS
igualdad	"$nombre" = "Maria"	$edad -eq 20
desigualdad	"$nombre" != "Maria"	$edad -ne 20
mayor	A > Z	5 -gt 20
mayor o igual	A >= Z	5 -ge 20
menor	A < Z	5 -lt 20

`break [n]` corta la ejecución de n niveles de *loop*
`continue [n]` salta a la siguiente iteración del n-ésimo *loop* que contiene esta instrucción

menor o igual	A <= Z	5 -le 20

ARGUMENTOS Y VALOR DE RETORNO

los script pueden recibir argumentos en su invocación. Para accederlos se usan variables especiales

- $0 contiene la invocación al script
- $1, $2, $3… contienen cada uno de los argumentos
- $# contiene la lista de todos los argumentos
- #? contiene en todo momento el valor de retorno del último comando ejecutado

FUNCIONES

las funciones permiten modularizar el comportamiento de los scripts

puedo declararlos de dos formas

- `function nombre{ block}`
- `nombre() {block}`

con la sentencia return se retorna un valor entre 0 y 255. El valor de retorno se puede evaluar mediante la variable $?

las variables no inicializadas son reemplazadas por un valor nulo o 0. Por defecto las variables son globales