Black System

Seguridad Informática

Seguridad Informática

FPTD/2021/019/2080

http://www.free-powerpoint-templates-design.com

Bash Scripting

Las funciones Bash pueden:

* Eliminar tareas repetitivas
* Ahorrar tiempo
* Proporciona una secuencia de actividades bien estructurada, modular y formateada
* Con scripts, podemos proporcionar valores dinámicos a comandos usando argumentos de línea de comando
* Puede simplificar comandos complejos en una sola unidad en ejecución
* Una vez creado, se puede ejecutar cualquier cantidad de veces por cualquier persona. Construye una vez y ejecuta muchas veces.
* Puede tener comandos de shell interactivos

Bash es definitivamente una gran herramienta para facilitar tu trabajo y mejorar tus proyectos

Crear un archivo .sh:

nano script1.sh
Después hay que darle permisos de ejecución: chmod u+x script1.sh
Para poderlo ejecutar: ./script1.sh

Todo script de bash en Linux debe comenzar con la siguiente línea:
#!/bin/bash

Variables: Podemos definir variables de la siguiente forma:

Cadena = “hola”

Para referenciarnos a esa variable y poderla usar
$Cadena

Funciones: Hay que crear primero la función antes de poder llamarla, o sea, más arriba:

```
MiprimeraFuncion (){
        comandos
        comandos
        …
}
```

Comandos más utilizados

- echo –n “Escribe tu nombre: ”
  read Minombre
  Estos dos comandos lo que hacen es imprimir por pantalla Escribe tu nombre:
  y cuando el usuario pone su nombre y le da al intro, guarda ese texto en la variable Minombre.

- Para poderle pasar a una función un parámetro, primero genero la función

```
Mifuncion (){
        Mkdir $valor
}
Mifuncion
```

- Condicionales: la estructura es la siguiente

```
If [ condición ]
then
        comandos
else
        comandos
fi
```

(para poner elseif = elif

```
echo –n “Escribe un numero: “
read resultado

if [ $resultado = “5” ]
Then
        echo “el numero es igual a 5”
Else
        echo “el numero es diferente a 5”
fi
```

Numérico
-gt mayor que (greater than) ------ $valor –gt 5
-lt menor que (lower than) ------- $valor –lt 5
-le menor o igual (lower o equal) ----- $valor –le 5
-ge mayor o igual (greater or equal) ------ $valor –ge 5
-eq igual (equal) ------ $valor –eq 5
-ne no igual ----- $valor –ne 5

Para hacer operaciones matemáticas: pondríamos la operación precedida por $((5 / 2))

Suma: +
Resta –
Multiplicación *
Division /
Resto %

Para operar con fechas tenemos el comando date:
Si calculamos la fecha dentro de un if poner $(date +%d)

Día actual: date +%d
Mes actual: date +%m
Año actual: date +%Y

Parámetros: cuando llamamos al script le podemos mandar los parámetros que queramos: ./script1.sh parametro1 parámetro2

Dentro del script podemos realizar las siguientes acciones con esos parámetros:

- Saber cuantos parámetros le hemos mandado $#: echo “Numero de parámetros “ $#
- Para referenciarnos a cada parámetro: el primero será $1, el segundo $2 : echo “el nombre del segundo parámetro es:” $2

Ejercicios

- Introduciendo un numero por teclado, que el programa nos diga si es par o impar
- Introduciendo la fecha de nacimiento que nos diga cuantos años tenemos
- Crear una calculadora que sume, reste, multiplique y divida
- Crear un pequeño tivial con 6 preguntas y 3 respuestas posibles para cada pregunta. Solo puede ser una la correcta. Por cada acierto se suma un punto y por cada equivocación se resta uno. Enseñar el total de puntos al finalizar el juego