# Programación en Shell

dcarrera@espol.edu.ec

## Ejercicio 1

- mount /mnt/floppy
- cp /etc/trabajos/archivo1  /mnt/floppy
- cd /etc/trabajos/archivo3   /mnt/floppy
- umunt /mnt/floppy

## Ejercicio 1

- crear capeta "archivos"
- mkdir archivos
- nano backup  (crea script)
  - cp /home/estud/ejercicio/opiniones.txt ../archivos/
  - cp  /home/estud/at_archivo  archivos/
- chmod a+x backup  (da permiso de ejecución)
- ./backup  (ejecuta el script)

## Shell

- es un programa que espera comandos del usuario, los procesa y los ejecuta.
- Tipos
  - sh
  - csh
- Bash (Bourne again shell) es un programa informático cuya función consiste en interpretar órdenes.

## Script

- es un conjunto de comandos escritos en archivos que permiten mejorar el performance de un proceso.
- No son programas compilados sino interprestados
- Los scripts se interpretan en shell
- Todo script se ejecuta en el servidor
- Un script es un texto y se hace más fácil por lo que son pequeños.

## Variables

- en los scripts no se declaran las variables
- Variables de sistma
- Variables creadas por el usuario
- "$" para referenciar a la variable
- Ejemplo:
  - contador = 5
  - echo $contador

## Variables en Shell

| Variable | Uso |
|---|---|
| $# | guarda los argumentos pasados en la línea de comandos |
| $? | guarda el valor de salida de la última ejecución |
| $0 | guarda el comando o programa |
| $* | guarda todos los argumentos como una lista ("$1, $2") |
| $@ | guarda todos los argumentos comoun arreglo ("$1,"$2") |

## Ejercicios

- Crear un programa llamdo "reverso"
  - echo "$2"
  - echo "$1"
  - nano reverso
- Aplicar permisos de ejercución
  - chmod a+x reverso
- Ejecutarlo
  - ./reverso hola todos
- Resultado:
  todos
  hola

## Uso de las comillas

- " " asigna valores a una variable. Esconde espacios en blanco para el shell, evalua lo que está adentro y lo asigna a la variable.
- Ejemplo:
- variable = "hola mundo"

## Uso de las comillas

- ' ' esconde todos los caracteres especialbes al shell
- Ejemplo:
- variable = "hola mundo, soy $LOGNAME" -> muestra holamundo soy estud02.
- variable = 'hola mundo, soy $LOGNAME' -> esconde y sólo muestra, hola mundo soy
- echo $variable

## Uso de comillas

- env --> muestra las variables de ambiente del sistema.
- ` ` --> sirven para ejecutar algún comando dentro de ellos
- Ej: variable= `ls`
- Sirve para esconder un caracter especial
- Ej:
  - precio=\$ 3.000 ok
  - precio='$3.000' ok
  - precio="$3.000" NO válido

## Comando test

- evalua una condición, usada en una expresión condicional

  test expresión

## Operadores del comando test (Operadores de cadena)

| Operador | Significado |
|---|---|
| str1 = str2 | V, si str1 es idéntido a str2 |
| str1 != str2 | V, si str1 es diferente a str2 |
| str | V, si str no es nulo |
| -n str | V, si la longitud str > 0 |
| -z str | V, si la longitud str=0 |

## Operadores del comando test (Operadores de enteros)

| Operador | Significado |
|---|---|
| int1 -eq int2 | V, si int1 es igual a int2 |
| int1 -ge int2 | V, si int1 es >= int2 |
| int1 -gt int2 | V, si int1 es > int2 |
| int1 -l int2 | V, si int1 es <= int2 |
| int1 -lt int2 | V, si int1 es < int2 |
| int1 -n int2 | V, si int1 != int2 |

## Operadores del comando test (Operadores de archivos)

| Operador | Significado |
|---|---|
| -d file | V, si el archivo es directorio |
| -f file | V, si el archivo es archivo común |
| -r file | V, si tiene permiso de lectura |
| -s file | V, si la longitud del archivo es != 0 |
| -w file | V, si tiene permiso de escritura |
| -x file | V, si tiene permiso de ejecución |

## Operadores del comando test (Operadores lógicos)

| Operador | Significado |
|---|---|
| !expr | V, si la expr no es verdadera |
| expr1 -a expr2 | V, si expr1 y expr2 son verdaderas |
| expr1 -o expr2 | V, si exp1 o expr2 son verdaderas |

## Comandos de salida y entrada

- echo: permite mostrar por pantalla la salida del programa
- read: tomar valores ingresados por pantalla y asignarlos a variables
- Ejemplo:
  - echo "ingrese el nombre, apellido, ciudad"
  - juan jose, ramirez, guayaquil
  - read nombre apellido ciudad

## Uso de los condicionales

- if .. then .. else .. fi
- if expresion then
- .....
- else
- ....
- fi

## Uso de los condicionales

```
if test "$name = John"    if test "$1"="$name" then
    echo "Hi John"
else
    echo "who are you"
fi
```

* Es posible anidar if

## case

- case .. in .... esac
- case string1 in
  - str1)
    - comandos;;
  - str2)
    - comandos;;
  - *)
    - comandos;;
  - esac

## Lazos repetitivos (for)

- for .. in .. do .. done
- Va ejecutando los comandos por cada elemento de la lista.

```
for vars in list
do
    comandos
done
```

- Lista: conjunto de palabras separadas por espacio y asignadas a una variable

## Lazos repetitivos (for)

- for file in *.txt

```
do
    echo "file: $file continue"
    wc -l $file
done

wc=word count
```

## Lazos repetitivos (while)

- while ... do ... done
- while expresion
  - do
    - comandos
  - done

## Lazos repetitivos (while)

- name="Barney"
- while test "$name"="Barney"
  - do
    - echo "ingres su nombre"
    - read name
  - done

## Lazos repetitivos (while)

- count=1
- while test -n "$*"  <-- guarda elementos como una lista
  - do
    - echo "parametro número $count $1"
    - count = `expr $count +1 `
  - done

## Otros

- Para ejecutar un programa en Bash, se coloca al inicio del script
- #!/bin/bash
- Para ejecutar
- #!/bin/sh