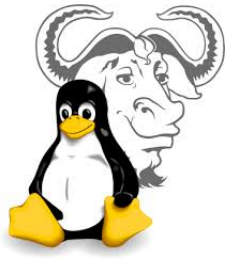
UNIVERSIDAD DE MENDOZA

FACULTAD DE INGENIERIA



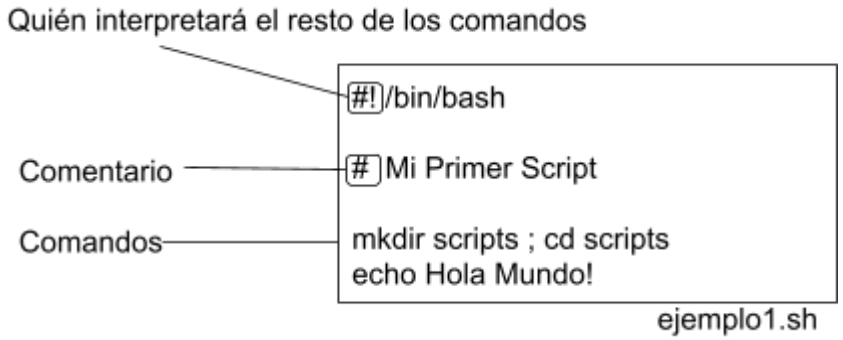
SISTEMAS OPERATIVOS
GUÍA PRÁCTICA DE LABORATORIO Nº5

GNU-LINUX

ejemplo1.sh

1.1. Editar un archivo llamado ejemplo1.sh

$vi ejemplo1.sh



1.2. Ejecutar el archivo ejemplo1.sh

$. ejemplo1.sh

¿Cómo se ejecuta un comando en shell?

1) Lee la entrada desde un archivo como un argumento o desde la terminal
2) Divide la entrada en tokens de acuerdo a las reglas de quoting. Se hacen expansiones de metacaracteres y alias
3) Se analizan los tokens y se dividen en comandos simples y compuestos

4) Se hacen expansiones separando los tokens expandidos en listas de nombres de archivo y comandos con sus argumentos
5) Se realizan redireccionamientos eliminando operadores y operandos de redirección
6) Se ejecuta el comando
7) Opcionalmente se espera la finalización del mismo para recoger su exit status

Caracteres especiales / metacaracteres

2.1. Analizar la salida y el comportamiento de los caracteres especiales:

$pwd > path.bck

$find /home/ -name gu* &

$echo El Dijo \"Hola\"

$echo Fecha y Hora actual: `date`

$echo Hay `wc -l /etc/passwd | awk '{print $1}'` usuarios en el sistema

2.2. ¿qué sucede con estos comandos?

$echo 2 * 3 > 1 es cierto

$echo El valor de este producto es $120

Esto se debe a Quoting. Los mecanismos de quoting deshabilitan el comportamiento por defecto o imprimen textualmente un metacaracter. Protegen metacaracteres dentro de una cadena a fin de evitar que se reinterpreten o expandan por acción del shell. Existen 3 mecanismos de quoting:

- Carácter de Escape \
- Comillas dobles "
- Comillas Simples '

2.3. Probemos los comandos anteriores utilizando quoting

2.4. ANSI-C Quoting nos permite que las cadenas de forma: $'texto' sean consideradas de forma especial.

- \a Alerta
- \b Retroceso
- \n Nueva Línea
- \t tab horizontal
- \v tab vertical
- \\ barra invertida "\"

$echo Se escucha… se escucha… $'\a' (Sonará un Beep)

$echo Hola $'\n'Mundo!

2.5. Utilizar o quitar los caracteres especiales y quoting para que estos ejemplos se puedan mostrar:

$echo <-$1250. **>; (update?) [y|n]

$echo La variable '$UID' contiene el valor --\> "$UID"

$echo It's <party> time!

Variables

La utilización de variables permite crear scripts flexibles y depurables. Una variable posee un *nombre* y un valor (*$nombre*). Se puede concatenar con una cadena: ${variable}cadena. Bash es case sensitive. Es un lenguaje No fuertemente tipado. Las variables con shell pueden:

- Crearse: $var1=10
- Asignarse: $var2=var1
- Borrarse: $unset var1

Las variables solo pueden contener letras números (las var con nombres numéricos están reservadas) o guiones bajos.

3.1. Es posible almacenar en una variable el contenido de la ejecución de un comando

$listar_archivos=$(ls)

$echo $listar_archivos

$listar_archivos=$(ls /home/mz/)

$echo $listar_archivos

Parámetros Posicionales o argumentos

4.1. Son variables cuyos nombres son números. Estas referencian a los argumentos de los comandos.

$find /home -name ejemplo1.sh

4.2. Visto el comando anterior… Completar:

- Nombre del Comando:
- 1º Argumento:
- 2º Argumento:
- 3º Argumento:

Arrays

Es una serie de casillas, cada una contiene un valor. Se accede a los elementos mediante índices. Éstos comienzan en 0 y hay más de 5k billones.

5.1. Asignemos colores:
$colores=(blanco negro azul cyan rojo)

5.2. Llamando a una variable

$echo mi impresora siempre necesita tinta ${colores[3]} y no lo uso!

5.3. ¿Qué sucede si...?

$echo ${colores[*]}

6.1. El comando *set* despliega las variables de shell junto a sus valores. Permite definir el comportamiento del bash (opciones)

$set `date`

$echo La hora Actual es: $4

Operadores

7.1. juguemos al Verdadero-Falso:

$true ; echo $? $'\n' ; false ; echo $?

$ var1=21 ; var2=22

$(("var1" > "var2"))

$echo $?

$ var1=Argentina ; var2=Argentina

$(("var1" > "var2"))

$echo $?

7.2. ¿Qué operación evalúa? ¿cuál devuelve?

$let "t1 = ((5+5,7-1,15-4))"

$echo "t1 = $t1"

$let "t1 = ((i=(5+5),7-1,15-4))"

$echo "t1 = $t1 i = $i"

Construcciones Condicionales:

if

```
if condición1;
then
        comandos1
elif condición2;
then
        comandos2
else
        comandos3
fi
```

8.1. Crear un script que me siga si soy el único usuario conectado

$vi usr_con.sh

___

```
#!/bin/bash
xuser=`who | wc -l`

if [ $xuser -gt 1 ] ;
then
      echo "Hay Más de un usuario conectado"
else
      echo "Estoy mas solo que kung fu"
fi
```

___

$. usr_con.sh

Case

```
case palabra in
        patron1)comandos1 ;;
        patros2)comandos2 ;;
        …
esac
```

8.2. Crea un script que me describa sobre el color que le asigne a una variable:

$color=cyan

$vi nousocase.sh

___

```
#!/bin/bash

if [ "$color" = blanco ] ; then
        echo "Que aburrido"
elif [ "$color" = negro ] ; then
        echo "como mi corazón"
elif [ "$color" = cyan ] ; then
        echo "Nunca Uso el cyan ¿porque me falta?"
fi
```

___

```
$. nousocase.sh

$vi usocase.sh
```

___

```
#!/bin/bash

case "$color" in
        blanco) echo "Que aburrido" ;;
        negro)  echo "como mi corazón" ;;
        cyan)   echo "Nunca Uso el cyan ¿porque me falta?" ;;
esac
```

___

```
$. usocase.sh
```

Construcciones Iterativas

for

```
for arg in lista ;
do
          comando (s) …
done
```

9.1. veamos los archivos .sh de la carpeta actual utilizando un script

```
$vi usofor.sh
```

___

```
#!/bin/bash

ruta=`pwd`
```

```
for archivo in "$ruta/"*.sh ;
do
        cat $archivo ; echo "_____"
done
```

___

$. usofor.sh

Select

```
select name [in lista] ;
do
          comando (s) …
done
```

9.2. Elijamos nuestra comida preferida mediante un script

$vi usoselect.sh

___

```
#!/bin/bash

ps3='Elija su comida preferida: '
echo $ps3
select food in "Carne" "Pescado" "Vegetal" "Pasta"
        do
              [ -e "$food" ] && continue
              echo "El finde haremos $food."
              break
        done
```

___

$. usoselect.sh

while

```
while comando ; # comando ~ condición ~ comando test
do
          comando (s) …
done
```

9.3. Contemos del 0 al 9…

$ vi uso while

___

```
#!/bin/bash
```

```
x=0
while [ $x -lt 10 ] ;
do
        echo -n $x $'\n'
        x=$(( $x + 1 ))
done
```

___

$. usowhile.sh

until

```
until comando ; # comando ~ condición ~ comando test
do
           comando (s)
done
```

9.4. ¿se cansaron de contar? Hora de hacerlo nuevamente pero con la negación del while ;)

$vi usountil.sh

___

```
#!/bin/bash

x=1

until [ $x -gt 10 ]
do
        echo -n $x $'\n'
        x=$(( $x + 1 ))
done
```

___

$. usountil.sh

9.5. Como vemos… el resultado no es el mismo. ajustar el script a fin de que ambos arrojen el mismo resultado.

Ejercicios… ahora solos…

10.1 Realiza un script que compruebe si el usuario actual es phonguito. Si es así, salúdalo sino despídete.

10.2. Realiza un script que permita el ingreso de un número. devolver el valor por pantalla y antes de salir consultar si desea volver a jugar.