Instituto Politécnico Nacional

Escuela Superior de Cómputo

Asignatura	**Sistemas operativos**
Prof.	**Ana Belem Juárez Méndez**
Práctica 1	**Introducción a shell script y comandos UNIX**

Objetivo

Conocer algunos de los comandos fundamentales de UNIX y aprender a desarrollar shell scripts básicos.

Introducción

El shell es quien traduce las líneas de comandos del usuario en instrucciones del sistema operactivo. Además de comandos, el shell ofrece otros elementos para mejorar su funcionalidad, tales como variables, funciones o estructuras de control. El conjunto de comandos internos y elementos disponibles, así como su sintaxis, depende del shell empleado.

Un shell script es un simple archivo de texto que contiene uno o varios comandos. Es común que los shell scripts tengan la extensión '.sh', para ayudar a la identificación de su contenido, pero no es obligatorio colocarla.

VARIABLES

En un shell script se pueden crear y utilizar variables. Las operaciones que se pueden realizar con ellas son las siguientes:

var="" var=	Definir
var="hormiga" var=10	Inicializar o modificar
$var ${var}	Acceso a valor

El shell permite el uso de algunas variables especiales, las cuales se muestran a continuación:

$$	Número de proceso del shell script en el que se está utilizando.
$0	Nombre completo del shell script que se está utilizando
$1..$9	$n hace referencia al n–ésimo argumento de la línea de comandos.
$#	Número de argumentos.
$*	Lista de todos los argumentos. Si se usa dentro de comillas dobles, se vuelve una única variable formada por los parámetros posicionales.
$@	Lista de todos los argumentos.
$?	Valor devuelto por el último comando, script, función o sentencia de control invocado.

ARREGLOS

Existen varias maneras para asignar valores a arreglos, algunas son las siguientes:

```
dias[0]=lunes
dias[1]=martes
dias[2]=miercoles
```

o

```
dias=([0]=lunes [1]=martes [2]=miercoles)
```

Podemos acceder al valor de un elemento en la posición `i` en el arreglo de la siguiente manera: `${dias[i]}`
Podemos acceder al tamaño del arreglo de la siguiente manera: `${#dias[@]` (devolvería 3).

FUNCIONES

Para definir una función, puedes usar cualquiera de las siguientes dos formas:

```
function nomfuncion
{
    comandos shell
}
```

o

```
nomfuncion ()
{
    comandos shell
}
```

SENTENCIAS DE CONTROL

CONDICIONAL IF-ELIF-ELSE
Presenta la siguiente sintaxis:

```
if condicion
then
    sentencias
elif condicion
then
    sentencias
...
else
  sentencias
fi
```

Operadores aritméticos

-lt	Menor que
-le	Menor o igual que
-eq	igual
-ge	Mayor o igual que
-gt	Mayor que
-ne	diferente

Operadores lógicos

&&	and
||	or
!	not

Operadores de comparación de cadenas

cad1 = *cad2*	*cad1* es igual a *cad2*
cad1 != *cad2*	*cad1 no* es igual a *cad2*
cad1 < *cad2*	*cad1* es menor a *cad2*
cad1 > *cad2*	*cad1* es mayor a *cad2*
-n *cad1*	*cad1* no es null (tiene tamaño mayor a 0)
-z *cad1*	*cad1* es null (tiene tamaño 0)

Operadores de atributos de archivos

-d *archivo*	*archivo* existe y es un directorio
-e *archivo*	*archivo* existe
-f *archivo*	*archivo* existe y es un archivo regular, es decir, no es un directorio u otro tipo especial de archivo.
-r *archivo*	Tienes permiso de lectura en *archivo*
-s *archivo*	*archivo* existe y no esta vacío
-w *archivo*	Tienes permiso de escritura en *archivo*
-x *archivo*	Tienes permiso de ejecución en *archivo,* o en el caso de ser documento tienes permiso de búsqueda
archivo1 -nt *archivo2*	*archivo1* es más actual que *archivo2*
archivo1 -ot *archivo2*	*archivo1* es más antiguo que *archivo1*

BUCLE FOR
Presenta la siguiente sintaxis:

```
for i in list
do
    sentencias que usen $i
done
```

BUCLE WHILE
Presenta la siguiente sintaxis:

```
while condicion
do
    sentencias . . .
done
```

BUCLE UNTIL
Presenta la siguiente sintaxis:

```
until condicion
do
    sentencias . . .
done
```

Desarrollo

Ejercicio 1. Investiga el uso de los siguientes comandos y caracteres especiales, y muestra su funcionamiento en una terminal.

cp	who	which	du
mv	wc	w	df
date	ifconfig	sudo	exit
cal	pstree	ps	cat
uname	su	grep	rm
mkdir	head	find	sort
>	>>	<	<<
|			

Ejercicio 2. Abre una terminal y posiciónate en un directorio de trabajo vacío de tu preferencia. Crea mínimo 5 archivos con el nombre de tu preferencia. Realiza un script que le cambie el nombre a estos archivos por: Archivo1.dat, Archivo2.dat, … ArchivoN.dat, donde N es el número total de archivos que creaste. El script debe funcionar de manera dinámica, es decir, si creas un archivo adicional y vuelves a ejecutar tu script también debe renombrarlo.

Ejercicio 3. Realiza un script que reciba al menos un argumento y que devuelva en pantalla el número de argumentos introducidos, sin el uso de $#. NOTA: Válida que al menos se introduzca un argumento.

Ejercicio 4. Realiza un script que reciba mínimo 2 números y a lo sumo 8 números como argumentos, ordénalos de menor a mayor y muéstralos en pantalla. NOTA: Válida el número de argumentos, no debe aceptar menos de 2 o más de 8.

Ejercicio 5. Imagina que tienes una colección de álbumes de música de tus grupos favoritos. En un archivo tienes almacenado el número de álbumes por grupo. Un ejemplo de como se ve tu archivo, es:

```
4 sonic youth
3 The Smiths
9 phoenix
8 Beatles
```

Realiza un script que imprima las N líneas mayores, es decir los N grupos con los que tienes más álbumes. El valor de N y el nombre del archivo donde esta contenida la información deben de pasarse como parámetros.