	UNIVERSIDAD DE CÓRDOBA FACULTAD DE INGENIERÍAS PROGRAMA DE INGENIERÍA DE SISTEMAS Y TELECOMUNICACIONES

Asignatura: LINUX AVANZADO I	Semestre: V	**Laboratorio No. 1**
Profesor: Jose Waldo de la Ossa	Temática**: Programación Shell Script**	

Consideraciones Generales

La presente actividad debe ser desarrollada y entregada en formato digital (PDF) en grupo (2 personas). *FECHA ENTREGA 25-08-2015 HASTA LAS 11:00 PM.*

Conceptualización

Shells y scripts

El *shell*, como programa, es una utilidad que permite a los usuarios interactuar con el *kernel* por interpretación de comandos que el mismo usuario introduce en la línea de comandos o en los ficheros de tipo *Shell script.*

Los *shell scripts* son ficheros de texto que contienen secuencias de comandos de sistema, más una serie de comandos propios del *shell* interactivo, más las estructuras de control necesarias para procesar el flujo del programa (tipo *while*, *for*, etc.).

Además, los *shells* suelen proporcionar los tres mecanismos siguientes:

- ✓ **Redirección:** dado que los dispositivos de E/S y los ficheros se tratan de la misma manera en UNIX, el *shell* los trata a todos simplemente como ficheros. Desde el punto de vista del usuario, se pueden redefinir los descriptores de los ficheros para que los flujos de datos de un descriptor vayan a cualquier otro descriptor; a esto se le llama *redirección*. Por ejemplo, nos referiremos a la redirección de los descriptores 0 o 1 como a la redirección de la E/S estándar.

- ✓ **Tuberías (*pipes*):** la salida estándar de un programa puede usarse como entrada estándar de otro por medio de *pipes.* Varios programas pueden ser conectados entre sí mediante *pipes* para formar lo que se denomina un *pipeline.*

- ✓ **Concurrencia de programas de usuario:** los usuarios pueden ejecutar varios programas simultáneamente, indicando que su ejecución se va a producir en segundo plano (*background*), en términos opuestos a primer plano (o *foreground*), donde se tiene un control exclusivo de pantalla. Otra utilización consiste en permitir

trabajos largos en segundo plano cuando interactuamos con el *shell* con otros programas en primer plano.

Variables de sistema

Algunas variables de sistema útiles que pueden consultarse (con el comando *echo*), son:

Variable	Valor ejemplo	Descripción
HOME	/home/juan	Directorio raíz del usuario
LOGNAME	juan	Id del usuario en el *login*
PATH	/bin:/usr/local/bin:/usr/X11/bin	Caminos
SHELL	/bin/bash	*Shell* del usuario
PS1	$	*Prompt* del *shell*, el usuario puede cambiarlo
MAIL	/var/mail/juan	Directorio del buzón de correo
TERM	xterm	Tipo de terminal que el usuario utiliza
PWD	/home/juan	Directorio actual del usuario

Las diferentes variables del entorno pueden verse con el comando *env*.

```
$ env
```

Tuberías

La **tubería** es una herramienta que permite utilizar la salida normal de un programa como entrada de otro, por lo que suele usarse en el filtrado y depuración de la información, siendo una de las herramientas más potentes de la programación con intérpretes Unix.

Pueden combinarse más de una tubería en la misma línea de órdenes, usando el siguiente formato:

Mandato1 | *Mandato2* ...

Todos los dialectos Unix incluyen gran variedad de filtros de información. La siguiente tabla recuerda algunos de los más utilizados.

Mandato	Descripción
`head`	Corta las primeras líneas de un fichero.
`tail`	Extrae las últimas líneas de un fichero.
`grep`	Muestra las líneas que contienen una determinada cadena de caracteres o cumplen un cierto patrón.
`cut`	Corta columnas agrupadas por campos o caracteres.
`uniq`	Muestra o quita las líneas repetidas.
`sort`	Lista el contenido del fichero ordenado alfabética o numéricamente.
`wc`	Cuenta líneas, palabras y caracteres de ficheros.
`find`	Busca ficheros que cumplan ciertas condiciones y posibilita ejecutar operaciones con los archivos localizados
`sed`	Edita automáticamente un fichero.
`awk`	Procesa el fichero de entrada según las reglas de dicho lenguaje.

El siguiente ejemplo muestra una orden compuesta que ordena todos los ficheros con extensión ".txt", elimina las líneas duplicadas y guarda los datos en el fichero "resultado.sal".

```
cat *.txt | sort | uniq >resultado.sal
```

Introducción a Bash

En Unix existen 2 familias principales de intérpretes de mandatos: los basados en el intérprete de Bourne (BSH, KSH o BASH) y los basados en el intérprete C (CSH o TCSH).

Shell haciendo referencia especialmente a **BASH** (*Bourne Again Shell*) –evolución de BSH, con características de KSH y CSH–, ya que es el intérprete de mandatos más utilizado en Linux e incluye un completo lenguaje para programación estructurada y gran variedad de funciones internas.

El shell que estamos usando

Si queremos saber que versión de shell tenemos instalado podemos usar el comando:

```
$ echo $SHELL
```

Este comando nos indica que shell estamos usando y en que directorio está instalado.

Si queremos conocer la versión de Bash podemos usar el comando:

```
$ echo $BASH_VERSION
```

También podemos saber donde está instalado Bash con el comando:

```
$ whereis bash
```

Puede conocer todos los shell de que dispone su máquina con el comando:

```
$ cat /etc/shells
```

Programación *scripts* en Bash

Todos los *scripts* Bash tienen que comenzar con la línea:

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

El *script* puede ejecutarse de dos modos diferentes:

1) Por ejecución directa desde la línea de comandos, siempre que tenga permiso de ejecución. Si no se da el caso, ponemos el permiso con: chmod +x script.

2) Por ejecución mediante el *shell*, llamamos al *shell* explícitamente:

```
bash script.
```

Variables en Bash

La asignación de variables se realiza por:

```
variable = valor
```

El valor de la variable se puede ver con:

```
echo $variable
```

donde '$' nos hace referencia al valor de la variable. La variable por defecto sólo es visible en el *script* (o en el *shell*). Si la variable tiene que ser visible fuera del *script*, a nivel de *shell* o de cualquier *shell* hijo (o *subshell*) que se genere *a posteriori*, será necesario "exportarla" además de asignarla. Podemos hacer dos cosas:

• Asignar y exportar después:

```
var = valor
export var
```

• Exportar en la asignación:

```
export var = valor
```

Parte 1 – Ejecución de comandos

1) Capture pantallazos para mostrar los comandos que satisfagan lo solicitado.
2) Listar de un directorio de varios archivos, los 10 primeros.
3) Realice las modificaciones necesarias al comando anterior, a fin que sean mostrados los 10 archivos de mayor tamaño, ordenados alfabéticamente.
4) ¿Cuál es el directorio que más espacio ocupa dentro de /usr ?
5) Que resultado produce la ejecución de los siguientes comandos (explique cada una de las partes del comando):

- ✓ `cat /etc/passwd | cut -d ":" -f1 | sort`
- ✓ `ps -ef | cut -d " " -f1 | grep root`

Parte 2 – Primeros script

1) Elabore un Shell script que permita enviar el día y hora actuales al fichero *ahora.txt.* Debe capturar un pantallazo una vez ejecutado.
2) Shell que no muestre las líneas que terminan por **bash** en el fichero **/etc/passwd**
3) Shell que nos entregue el número de líneas y de palabras del fichero /etc/passwd