INTRODUCCIÓN A LA COMPUTACIÓN
PRÁCTICO – 3 LINUX SCRIPTS

VARIABLES EN LINUX

1 - a) Defina la variables *var1* con el valor entero 5 y la variable *var2* con el contenido alfanumérico (string) *abcdefg*. Despliegue por consola su contenido.

b) Despliegue por consola el mensaje “El contenido de var1 es: ” seguido por el contenido de var1, seguido por “ y el de var2 es: ”, seguido finalmente por el contenido de var2.

2 - a) Despliegue el contenido de la información relativa a sus usuario guardada en las variables del sistema: USER, HOME, HOSTNAME, PWD, OSTYPE, SHELL. Analice a que se refiere cada variable.

b) Despliegue el mensaje “Estoy parado en el directorio: ”, seguido del directorio donde se encuentra. El comando debe ser el mismo independiente de cual directorio sea en el que lo ejecuta.

3 – a) Defina una variable de tipo array llamada *arrayVar* con 2 elementos. Llamarlos *primer elemento* y *segundo elemento* (el índice de un array comienza en cero). Despliegue en la consola, los dos contenidos del array.

b) Despliegue todos los contenidos del array con el comodín “*”

SUSTITUCIONES Y COMILLAS EN LINUX

3 - Defina la variables var con el contenido alfanumérico “soy una variable”. Ejecute y analice los comandos:

- echo $var
- echo 'imprimo los siguiente: $var en la fecha `date`'
- echo “imprimo lo siguiente: $var en la fecha `date`”
- echo “Puedo concatenar strings,” “ separando comillas”
- echo “La sustitución de una variable no tiene porque ir entre comillas: ”$var
- echo “este ejemplo muestra varias sustituciones de comandos: `cd /home; ls` `cd ; pwd`”
- homedir=/home ; echo “Hago una doble sustitución, de variable y comando: `ls $homeDir`”
- Cuando quiero imprimir explícitamente dentro de comillas dobles algún carácter reservado del sistema como los de sustitución, se anteceden por \, fijarse que se imprime con: echo “imprimo: \$ y \" ” y con echo “”imprimo: $ y ``”

4 – El caracter $ también sirve para sustituir expresiones aritméticas y no solo de variables, las sintaxis es **$((expresión arit.))**. Pruebe y analice los comandos:

- echo $((6+3))
- echo 'entre comillas verticales vimos que nada se sustituye: $((6+3))'

- echo “entre comillas dobles el caracter $ y las comillas verticales, se sustituyen: `echo suma=$((6+3))`”
- Se pueden efectuar cálculos con enteros con el comando expr :
  echo “la suma de 6+3 es: `expr 6 + 3`” .Se debe dejar espacio entre 6,+,3 pues son argumentos del comando expr.
- echo “lo anterior es equivalente a la sustitución de la expresión aritmética $ ((5+3))”

SCRIPTS BÁSICOS

5 -a) ¿Qué hace el siguiente script?

```
#!/bin/bash
echo $USER@$HOSTNAME:$PWD$
```

b) ¿Qué se debe agregar a la línea del script para que imprima la fecha actual?

6) Escribir un scipt que verifique si existe una archivo de texto con un nombre pasado como argumento al script (que está en el mismo directorio que le archivo). De existir, concatenarle el texto “final de archivo” al final. Si no existe, crearlo y agregarle el mismo texto. (Sugerencia: usar el comando if con alguna de las opciones para archivos/directorios).

7 -a) Escribir un script que reciba 2 argumentos numéricos y devuelva la suma de ellos. (Sugerencia: puede utilizar el comando **expr** o sustitución de expresiones aritméticas).

b) Modifique el script anterior para que verifique que la cantidad de argumentos sea exactamente dos. De no ser así que devuelva el siguiente mensaje “uso: ” seguido del nombre del script, seguido de “ argnro1 argnro2”. El mensaje debe ser independiente del nombre del script. (Sugerencia: Utilice la variable reservada de shell “#” para saber cuantos argumentos pasó y la “0” para el nombre del script).

8 - a) Exit Status**.**
Todos los comandos de Linux cuando se ejecutan, pueden finalizar en forma correcta o devolver un error. El *“Estado de Salida”* (Exit Status) se guarda en una variable del sistema “?”. Cuando el resultado de un comando (o script) es exitoso, devuelve 0 y si es erróneo, devuelve 1 o mayor. Ejecutar los comandos en consola:

- pwd ; echo $?
- pwdirectory ; echo $?
- ls *.dat ; echo $? (si no existen archivos que con extensión .dat, debe dar error)

b) Se puede forzar la salida de un script en caso de que no se cumpla algún requerimiento, con el comando *exit int* (donde int es un entero mayor que cero). (Es posible forzar la salida en caso de éxito con *exit 0*). Escriba, ejecute con y sin argumentos y analice el script, finalmente en consola vea cual es el exit status de su script (echo $?):

```
#!/bin/bash
# Este script debe recibir dos argumentos de entrada, de no ser así
```

```
# devuelve un error

nroArgs=$#

if  [ $nroArgs –ne 1 ]
then
    echo “Error, cantidad de argumentos incorrecta”
    echo “Uso: $0 arg”
     # salgo del programa devolviendo un error status 1 (error es  >
0)
    exit  1
else
    echo “Correcto: $0 $1”
fi
```

LOOPS EN SCRIPTS

9 – a) Con el comando **for in lista do**…, escribir un script que liste todos los archivos que hay en su directorio home. (Sugerencia: exprese la lista de archivos, construyendo u path con la variable de shell del sistema HOME y usando el comodín *).

b) Modifique la parte a para que liste solo los archivos que terminan en “.sh”

c) Modifique la parte a para listar solo aquellos que son ejecutables. (Sugerencia use el comando if dentro del cuerpo del bucle para saber si el archivo actual es ejecutable). (Crear algún archivo y cambiar todos sus permisos a no-ejecución para probarlo).

10 – a) Escribir un script que devuelva la suma de todos los enteros positivos hasta un número dado (inclusive) como argumento.

b) Escribir un script para calcular el factorial de un número dado como argumento.

c) El comando **read var** espera a que el usuario ingrese un valor en la stdin y lo asigna a la variable var (probar: **read var ; echo $var**). Escribir un script que pida en la entrada un número al usuario, hasta que el número sea mayor a 10. Por ejemplo:

```
user@hostname:~$ ./ej10c.sh
Ingrese un número:
5
Ingrese otro número:
8
Ingrese otro número:
10
Ingrese otro número:
11
el número que ingresó es mayor que 10
user@hostname:~/pr3$
```

(Sugerencia usar el comando while)

APLICACIÓN DE UN SCRIPT DE LINUX

11 - Este ejercicio es una aplicación práctica real del uso que se le puede dar a un script.
Ver http://www.cv.nrao.edu/~rfisher/Ephemerides/ephem_descr.html
El planteo del problema es el siguiente:

El catálogo de Efemérides del Sistema Solar de JPL (Jet Propulsión Laboratory), especifica el pasado y futuro del Sol y los planetas del sistema solar. El estado del arte de estos catálogos es la versión DE405. La estructura de uno de los archivos del catálogo es (es irrelevante para nuestro propósito):

- Interval start Julian date
- Interval end Julian date
- Mercury sub-block
    - Subinterval 1
        - X-coordinate coefficients
        - Y-coordinate coefficients
        - Z-coordinate coefficients
    - Subinterval 2
        - X-coordinate coefficients
        - Y-coordinate coefficients
        - :
    - Subinterval 3
    - :
- Venus sub-block
    - Subinterval 1
        - X-coordinate coefficients
        - Y-coordinate coefficients
        - Z-coordinate coefficients
    - Subinterval 2
    - :
    - etc.

El encabezado (junto con otros más) muestra la fecha de inicio y finalización de los tiempos discretos a los que corresponden las coordenadas del archivo. Estos catálogos planetarios son accesibles de JPL, a través de un *ftp anonimo* en la url:

ftp://ssd.jpl.nasa.gov/pub/eph/planets/ascii/

Allí se encuentra un serie de archivos con nombre *ascpXXX.405* con la estructura especificada arriba, donde XXXX es el año de inicio. Los archivos varían de a 20 años. Para producir un archivo de formato estándar o canónico efemérides, se debe bajar de la dirección de arriba el archivo *header.405* y uno o más de estos archivos, luego concatenar header.405 con los archivos bajados. Finalmente debemos bajar el programa *asc2eph* (archivo ASCII o sea texto “to” ephemerides) y correrlo. El programa requiere como argumentos el archivo concatenado y genera un archivo efemérides estándar JPLEPH.

Vamos a resolver todo el problema con un solo script de Linux. Con un browser de Internet, ir al sitio: ftp://ssd.jpl.nasa.gov/pub/eph/planets/ascii/ y ver los archivos

ascpXXX.405 y header.405 (solo para entender el problema). Luego crear un directorio *ephemerides* dentro de su home de trabajo.

Debemos crear un script *ephemerides.sh* en el directorio *ephemerides* que haga:

- Crear dos directorios: “downloads” y “progs”

- Baje el archivo *header.405* de la dirección antes dada, en *downloads.*

- Reciba como argumento dos números que delimiten el rango de años de archivos que queremos bajar; por ej.: 2000 2040, entonces debemos bajar los archivos *ascp2000.405 ascp2020.405 ascp2040.4.* en el directorio *downloads.*

- Luego debe concatenar header.405 ascp2000.405 …etc. en ese orden, en un archivo que llamaremos *entrada.405.*

- Finalmente el script debe ejecutar el programa *asc2eph*, este programa debe ser bajado en el directorio “progs”. El programa recibe como argumento, el contenido de *entrada.405* o sea debemos redireccionar *entrad.405* a *asc2eph* cuando lo ejecutemos.

Podemos utilizar toda la artillería de herramientas del sistema operativo en forma de programa, como servicios de red (wget), filtrado de texto (concatenar los archivos), ejecución de un programa dado, de forma encadenada con pocas líneas de código de script. Esto hace de Linux una herramienta invaluable para este tipo de tareas.