RETO 2- SCRIPTS

DAW

1- Variables

- Variable: Lugar o espacio en la memoria, que puede contener un valor o no.
  - Si no contiene valor decimos que está vacía.
- Podemos encontrar variables locales y de entorno.
  - **LOCALES**: Solo se ven por la shell que se está ejecutando en ese momento.
  - **DE ENTORNO**: Sirven para que puedan ejecutarlas diferentes programas.

1- Variables

- **VARIABLE LOCAL**
  - No hay que definirlas, se crean al añadirles un valor:

    variable=”valor”
  - Para verla, escribimos $ delante de la variable.

    echo $variable
  - Por defecto es local, para convertirla en variable de entorno

    export variable

    export VAR=”VALOR”

1- Variables

- **VARIABLES DE ENTORNO:**
  - Servirán para que ciertos programas las utilicen o para configurar parámetros del entorno de trabajo.
  - Se suelen escribir en mayúsculas para diferenciarlas de las locales.
  - Las puede usar cualquier usuario desde cualquier terminal.
  - Algunos archivos que contienen variables de entorno son:
    - /etc/enviroment → PATH
    - /etc/default/locale → LANG
    - /etc/profile

1- Variables

- VARIABLES DE ENTORNO MÁS USADAS:
  - **HOME:** ruta de nuestro directorio personal
  - **USER:** nombre de usuario
  - **SHELL**: ruta al intérprete de órdenes que se está ejecutando
  - **HOSTNAME:** nombre asignado al equipo
  - **TERM**: tipo de terminal.
  - **LOGNAME**: nombre del usuario que ejecuta la shell.
  - **PATH**: rutas en las que el intérprete busca las órdenes a ejecutar cuando no especificamos dónde están.
  - **PWD**: directorio de trabajo actual.
  - **OLDPWD**: Para usar cd con – (Guarda el último directorio donde estuvimos)
  - **PS1**: prompt primario
  - **PS2:** promt secundario, suele ser >

1- Variables

- ALGUNOS COMANDOS:
  - **Set** → muestra variables locales y de entorno
  - **Env** → muestra las variables de entorno
  - **Export** → exporta una variable local al entorno
  - **Unset** → elimina una variable
  - **Echo $variable** → Muestra el valor de la variable

1- Variables

- EJERCICIOS- ¡TODO POR COMANDOS!
- 1) Crea una variable local asignándole un valor. Cambia de usuario y comprueba si ve la variable (con echo). Vuelve al usuario principal y exporta la variable. Cambia de usuario otra vez y comprueba si ahora se ve.
- 2) Crea una variable local asignándole un valor. Muestra su valor por pantalla. Comprueba que aparece cuando se miran las variables locales del sistema. Bórrala.
- 3) Muestra las variables de entorno y las locales. Muestra en la misma línea el contenido de PATH, HOME, SHELL, HOSTNAME y USER. Muestra el valor del prompt secundario.

2- Scripts

- Archivo de texto plano que contiene comandos que se irán interpretando por la shell.
- Para que pueda ejecutarse:
  - Tendrá permisos de ejecución.
  - Estará en un directorio incluído en la variable PATH
  - Lo indicaremos mediante ruta absoluta/relativa

```
alumno@jefe:/usr/local/sbin$ cd /usr/bin
alumno@jefe:/usr/bin$ ls
'['                                 foo2hbpl2-wrapper                   loadkeys
 2to3-2.7                           foo2hiperc                          loadunimap
 411toppm                           foo2hiperc-wrapper                  localc
 aa-enabled                         foo2hp                              locale
 aa-exec                            foo2hp2600-wrapper                  locale-check
 aconnect                           foo2lava                            localectl
 acpi_listen                        foo2lava-wrapper                    localedef
 add-apt-repository                 foo2oak                             locate
 addpart                            foo2oak-wrapper                     lodraw
 addr2line                          foo2qpdl                            loffice
 alsabat                            foo2qpdl-wrapper                    lofromtemplate
 alsaloop                           foo2slx                             logger
 alsamixer                          foo2slx-wrapper                     logname
```

2- Scripts. ¿Cómo creamos uno?

- 1) Avisamos al sistema que shell vamos a utilizar.
  - #!/bin/bash
  - Después de esa línea, todos los # son comentarios
- 2) Añadimos la extensión .sh
  - No siempre hace falta, pero da información al usuario que va a utilizarlo.
- 3) Añadimos comandos que queremos que se ejecuten en el orden lógico que deben seguir.

2- Scripts. ¿Cómo creamos uno?

- EJERCICIO
- Crea un script que se llame borra.sh, que borre la pantalla y te diga la fecha de hoy.
  - ¿Cómo cambias los permisos para que se ejecute?
  - ¿Cómo lo ejecutas?

2- Scripts. Variables

Variable	Función
$0	Nombre del *shell script*.
$1, $2, $3,...	Parámetros o argumentos posicionales que se introducen desde la línea de comandos.
$#	Número de parámetros o argumentos posicionales.
$*	Variable que recoge el valor de todos los argumentos.
$?	Valor devuelto por el último comando ejecutado.
$$	PID del shell *script*.

- El cuadro muestra las varaibles introducidas como argumentos.
- Si queremos que el script sea interactivo utilizamos el comando ***read variable***
- El numero máximo de parámetros son 9, por lo que si queremos utilizar más, usaremos el comando ***shift → Desplaza los parámetros una posición, EJ: El $1 desaparece y el $2 se convierte en $1.***

2- Scripts. Estructuras de control.

if-then-fi

```
if [ condición ]
then
        comando/s
fi
```

if-then-else-fi

```
if [ condición ]
then
        comando/s
else
        comando/s
fi
```

if-then-elif-then-else-fi

```
if [ condición ]
then
        comando/s
        elif [ condición ]
        then
                comando/s
        else
                comando/s
fi
```

case-in-esac

```
case $variable in
        expr1)comando/s;;
        expr2)comando/s;;
        ...
        *) comando/s;;
esac
```

for-in-do-done

```
for variable in valores
do
        comando/s
done
```

while-do-done

```
while [condición]
do
        comando/s
done
```

2- Scripts. Expresiones condicionales

- Podemos añadir las siguientes condiciones entre los [ ]

CONDICIÓN	Devuelve true o verdadero si...
-f $variable	...variable es un fichero
-d $variable	---variable es un directorio
-r $variable	...variable tiene permiso de lectura
-w $variable	---variable tiene permiso de escritura
-x $variable	...variable tiene permiso de ejecución
-e $variable	...variable es un archivo que existe
$var1 = $var2	...var1 es una cadena igual que var2
$var1 != $var2	...var 1 es una cadena distinta de var2
-z $variable	...variable es una cadena vacía
-n $variable	...variable es una cadena no vacía

2- Scripts. Expresiones condicionales.

- Si queremos comparar dos valores numéricos
- *$var1 -eq $var2* → *La variable 1 es igual a la variable 2*
  - -ne → no es igual
  - -lt → less than, 1 es menor que 2
  - -gt → greater than, 1 es mayor que 2
  - -le → less equal, 1 es menor o igual que 2
  - -ge → greater equal, 1 es mayor o igual que 2
- Podemos añadir operaciones lógicas
- *$var1 -eq $var2 && $var1 -gt $var3* → *var1 es igual a var2 Y var1 es mayor que var3*
  - II → or, una condición Ó otra
  - ! → NOT

2- Scripts. Funciones.

- Podemos añadir funciones o subprogramas del principal.
- Se definirán antes del programa principal
- Para definirla:
  - Function nombre {comandos}
- Lo explicamos con un ejemplo:
  - ENLACE AL EJEMPLO

```
#!/bin/bash

function ahora {
#Definimos una variable a partir de la ejecución de un comando, observar que no hay espacio antes y después del igual
fecha=$(date +%H:%M:%S)
sleep 5
echo Tarea $1 completada a las $fecha


}

#Programa principal, al escribir ahora, hace referencia a la función definida arriba
ahora desayuno
ahora comida
ahora cena
```

```
alumno@jefe:~$ ./comida.sh
Tarea desayuno completada a las 12:41:09
Tarea comida completada a las 12:41:14
Tarea cena completada a las 12:41:19
alumno@jefe:~$
```

2- Scripts.

- EJERCICIO SCRIPTS 1:
- Cambia el valor de la variable PATH para que te incluya tu directorio personal.
- A continuación crea un shell script, param.sh, que muestre los parámetros que ha recibido, cuántos parámetros son, el nombre del shell script y el PID del proceso.

2- Scripts.

- EJERCICIO SCRIPTS 2:
- Crea un shell script, fich.sh, que te pregunte el nombre de un fichero. Si el fichero existe, te debe mostrar información sobre él en formato largo. Si no existe o no es un fichero, te mostrará un mensaje de que no existe.

2- Scripts.

- EJERCICIO SCRIPTS 3:
- Crea un shell script, fichodir.sh, que reciba una serie de parámetros por la línea de comandos, los muestre y te diga si son ficheros o directorios.

2- Scripts.

- EJERCICIO SCRIPTS 4:
- Crea un shell script, reciente.sh, que te pregunte el nombre de un fichero. Si existe, para cada entrada del directorio personal, te dirá si es más reciente o no que el fichero que has escrito.
- NOTA, para comprar si es mas reciente $var1 -nt $var2

2- Scripts.

- **SCRIPTS DE INICIO DE SESIÓN:**
  - Al arrancar el sistema o al iniciar sesión se ejecuta una shell que lee los archivos **/etc/environment, /etc/profile, /etc/bash.bashrc**
  - Después se ejecutan los archivos **~/.bash_profile, ~/.bash_login** o **~/.profile**, que son parámetros de la cuenta del usuario.
  - Los ficheros **/etc/bash.bashrc y ~/.bashrc** se ejecutan cada vez que el usuario llama a la shell, lo que nos da la posibilidad de ejecutar comandos diferentes para el inicio de sesión de cada usuario.
  - Existen **.bash_login o .profile** en caso de que .bash profile no exista o no utilicemos la shell bash.

2- Scripts.

- **SCRIPTS DE FIN DE SESIÓN:**
  - Cuando salimos del sistema el fichero que lee el shell bash es .bash_logout.
  - Podemos utilizarlo para incluir comandos que queremos que se ejecuten cuando salimos.
  - Si el fichero no existe o no contiene ningún comando, no se ejecutará ningún comando al terminar la sesión.

2- Scripts.

- SCRIPTS DEL SISTEMA: Pueden ser de inicio de sesión o de fin de sesión.
- **SCRIPTS DE INICIO DE SESIÓN:** Al arrancar el sistema o cada vez que un usuario inicia una sesión se ejecutan estos scripts.
- **SCRIPTS DE FIN DE SESIÓN**: Al cerrar la sesión o apagar el sistema se ejecutan estos scripts para que al arrancar el sistema otra vez tengamos todo preparado.

2- Scrips

- EJERCICIO SCRIPTS 5:
- Mira en tu directorio personal los archivos ocultos para ver cuales de los scripts de inicio de sesión tienes.

2- Scripts

- EJERCICIO SCRIPTS 6:
- Modifica los ficheros .profile y .bashrc de tu directorio personal y los ficheros /etc/profile y /etc/bashrc añadiéndoles una línea al inico que muestre por pantalla el nombre del fichero que se va a ejecutar.
- Para ver el orden en que se ejecutan los ficheros inicia una nueva sesión en la primera terminal virtual.

2- Scripts

- EJERCICIO SCRIPTS 7:
- Modifica los ficheros .profile y .bash_logout de tu directorio personal de manera que cada vez que se inicie una sesión se escriba en un fichero llamado .movimientos (si se entra, si se sale del sistema o si se inicia una nueva sesión) el nombre del fichero shell script que se está ejecutando y la fecha.
- PISTA → $0 = NOMBRE DE LA SHELL

THE END