Profesor: Victor Valenzuela

Scripts en Bash

Para ejecutar un programa en Bash

- **Agregar en la primera línea del archivo la cadena #!/bin/bash, dar permiso de ejecución al archivo.**
- Emplear **./** seguido del nombre del archivo.
- **Los archivos de bash tienen la extensión .sh**
- Para ejecutar también se puede escribir **bash** seguido de un espacio y el nombre del archivo.
  - Bash archivo.sh

Variables

- Definición => Variable=/home
- Uso => ls –l $Variable
- El caracter '$' es empleado para distinguir variables o bien parámetros de un script.

Sustitución de Variables

- Variables de ambiente de bash => set
  - Ejemplos: $HOME, $USER, $LINES,$HOSTNAME,$HISTFILE
- Operaciones matemáticas => $((operación))
- Si un valor de una variable debe tener espacios, se encierra todo el valor entre comillas.

Sustitución de Variables

- Variables especiales:
- $1, $2, $3… => parámetros de entrada del script.
- $# => Es remplazado por la cantidad de parámetros que el script recibe.
- $* => Que se expande a todos los parámetros que el script haya recibido, un parámetro se separa de otro con el valor de la variable IFS que normalmente es un espacio.
- $? => Todo programa al terminar debe retornar un número al sistema operativo, por convención o significa operación exitosa y números diferente representan errores. $? se expande al número retornado por el último programa ejecutado en primer plano. Un script puede retornar un 3 en lugar de 0 con exit 3

Sustitución de Variables

- $$ => Identificación del proceso que se está ejecutando.
- $! => Identificación del proceso del último comando que se ejecutó en segundo plano.
- $0 => Nombre del script (de hecho, es el nombre del programa, o mejor dicho, lo ingresado para ejecutar el programa)

Sustitución de comandos

- Un comando encerrado entre apóstrofes invertidos (i.e `*comando*`) o entre las cadenas "$(" y ")", será expandido al resultado que tal comando envíe a salida estándar cuando es ejecutado. Por ejemplo:
  - TEXTOS=`ls *.txt` asignará a la variable TEXTOS los nombres de los documentos tipo texto (el resultado de **ls *.txt**).
  - ls $(cat rutas.txt) presentará los archivos de los directorios que estén en el archivo rutas.txt.

Citas de Texto

- Un texto que se encierra entre apóstrofes (') no es expandido. Esto es útil cuando se requiere una cadena que tiene algunos caracteres reservados para expansiones. Por ejemplo
- **N=10**
- **echo '$N' es $N** enviará a salida estándar **$N es 10**

Comandos y programas útiles de Bash

- **read :** Lee una línea de entrada estándar y asigna las palabras a las variables que sigan al comando **read**. Puede especificarse un mensaje que se presentará como prompt antes de empezar a leer con la opción -p *mensaje*. El siguiente ejemplo lee dos palabras en las variables NOMBRE y APELLIDO:
- read -p "Teclee nombre y apellido: " NOMBRE APELLIDO

Comandos y programas útiles de Bash

- **dirname:** Recibe como primer parámetro el nombre completo de una archivo, incluyendo su ruta y envía a salida estándar sólo la ruta. Por ejemplo:
- dirname /usr/doc/xterm/README.Debian presenta en salida estándar /usr/doc/xterm.
- **basename:** Análogo a **dirname**, pero envía a salida estándar el nombre del archivo.

Comandos y programas útiles de Bash

- **cat [archivo 1] [archivo 2]…[archivo n] > [archivo combinado]:** permite mirar, modificar o combinar un archivo.
- **>**: redirecciona la salida standard (salida por pantalla) hacia un archivo. Si el archivo existe y tiene información adentro, la reemplaza. **Si el archivo no existe, lo crear e ingresa la información**.
- **>>**: redirecciona la salida standard a un archivo, pero la información se escribe al final del archivo.
- **<** : se usa para utilizar la información de un archivo como entrada de la orden especificada
  - Sort < archivo

Manejo de procesos

- Top -> programas en tiempo real corriendo en el sistema
- ps aux -> foto de procesos
- killall <nombre aplicación> -> mata proceso
- kill -9 <pid> -> mata proceso con process id

Ejercicios (bash)

1. Realizar un programa que guarde en un archivo la lista de archivos .txt existentes en la carpeta actual
2. Realice un programa que entregue en pantalla el promedio entre 2 números dados como argumento.
3. Realice un programa que le pregunte el nombre de un archivo a crear, lo cree en la carpeta de usuario, e ingrese en el los datos de: nombre de usuario y nombre de maquina.
4. Realice un programa que elimine el contenido del historial de bash.

Ejercicios (bash)

5. Crear un programa que ingrese datos **al principio** de un archivo (antes del contenido ya existente). Qué archivo y los datos a escribir serán pasados por argumento (1er y 2do argumento respectivamente).

Ejercicios (C, Python, Perl + bash)

1. crear programa que cuenta infinitamente y muestra por pantalla
   - Matarlo a través de otra consola
2. crear programa que cuenta **escribiendo en un archivo*** y ejecútelo en segundo plano (& después del comando para iniciar el programa).
   - Utilice **tail** -> para ver contenido de un archivo en tiempo real
   - volver a matar el proceso desde otra consola.
3. Crear un programa que se base en el código anterior, pero que obtenga el pid del proceso y lo guarde en un archivo antes de seguir con el proceso normal.
   - (<) (en c: getpid() perl:$$ Python:os.getpid()).
4. matar el proceso de acuerdo a pid guardado
5. **hacer un script bash para iniciar y detener procesos en segundo plano (esto es un programa al que por argumentos se le ingrese el nombre de un programa, pueda iniciarlo en segundo plano y apretar cualquier botón, detenga el programa que inició y a si mismo)**.

*Nota:
c: https://www.includehelp.com/c/getpid-and-getppid-functions-in-c-linux.aspx
Python: https://appdividend.com/2019/02/06/python-os-module-tutorial-with-example/