Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica
IE0117 – Programación Bajo Plataformas Abiertas
II ciclo 2023

Laboratorio 2 - Paquetería, shell/bash, "scripting", y cron.

Oscar Porras Silesky C16042
Grupo 001

Profesor: Ing. Julián Gairaud

Asistente: Felipe Agüero Peralta

27 de septiembre de 2023

1. Primera Parte: Paquetería.

1. (6 pts) "nsnake" es un juego que se ejecuta en la terminal. Se debe instalar desde la terminal el programa "nsnake". Se debe explicar detalladamente el proceso que se utilizó para instalarlo (usando apt), desde su búsqueda, su instalación y su uso.

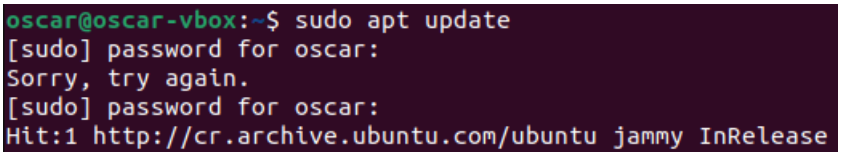

Figura 1: Comando sudo apt update.

Se actualiza la paquetería con el comando apt update.

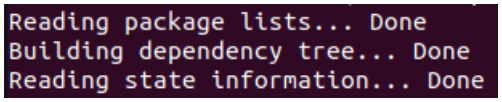

Figura 2: Verificación del comando apt udate.

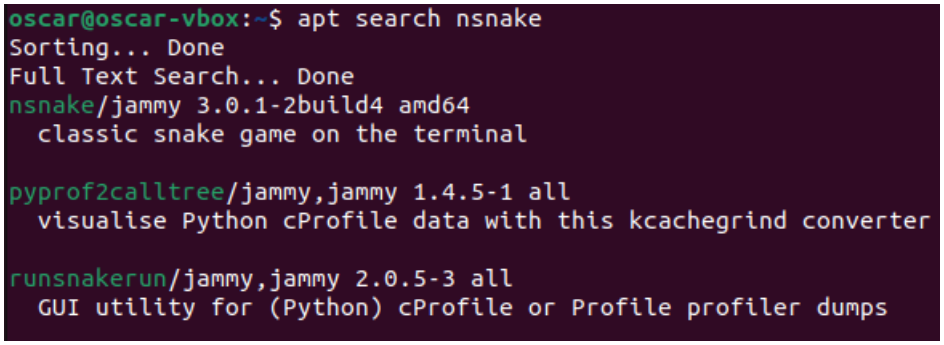

Figura 3: Búsqueda de nsnake.

Se busca el programa de nsnake utilizando apt serach nsnake.

```
oscar@oscar-vbox:~$ sudo apt install nsnake
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  nsnake
0 upgraded, 1 newly installed, 0 to remove and 23 not upgraded.
Need to get 111 kB of archives.
After this operation, 407 kB of additional disk space will be used.
Get:1 http://cr.archive.ubuntu.com/ubuntu jammy/universe amd64 nsnake amd64 3.0.
1-2build4 [111 kB]
Fetched 111 kB in 0s (297 kB/s)
Selecting previously unselected package nsnake.
(Reading database ... 184403 files and directories currently installed.)
Preparing to unpack .../nsnake_3.0.1-2build4_amd64.deb ...
Unpacking nsnake (3.0.1-2build4) ...
Setting up nsnake (3.0.1-2build4) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for mailcap (3.70+nmu1ubuntu1) ...
Processing triggers for desktop-file-utils (0.26-1ubuntu3) ...
Processing triggers for hicolor-icon-theme (0.17-2) ...
Processing triggers for gnome-menus (3.36.0-1ubuntu3) ...
oscar@oscar-vbox:~$ nsnake
```

Figura 4: Instalación de nsnake.

Se instala nsnake con el comando sudo apt install nsnake.

Figura 5: Nsnake abierto.

```
oscar@oscar-vbox:~$ which nsnake
/usr/games/nsnake
```

Figura 6: Verificación de localización de nsnake.

Se verifica la localización de nsnake con el comando which nsnake.

2. Segunda Parte: Shell/Bash

1. (2 pts) Se debe explicar la diferencia entre los conceptos de “shell” y “bash”.

”Shell” es una interfaz de usuario que permite interactuar con el sistema operativo y puede ser tanto gráfica como de línea de comandos. ”Bash”, por otro lado, es una versión específica de una shell de línea de comandos, utilizada para ejecutar comandos y gestionar el sistema.

2. (5 pts) Se debe declarar una variable de ambiente (usar cualquier nombre que se quiera) con export. Se debe demostrar su existencia utilizando el comando printenv. Se puede utilizar grep para buscar solamente la nueva variable. Se debe declarar otra variable (igualmente usar cualquier nombre), pero esta vez sin utilizar export. Se debe investigar cómo obtener las variables de ambiente en Python3, abrir un Shell de Python (o crear un script), e intentar obtener ambas variables previamente creadas. Aquí se da un ejemplo de cómo hacerlo desde el shell interactivo de Python. Se debe explicar brevemente lo que sucede ¿cuál(es) variable(s) se puede obtener desde Python y por qué?

```
oscar@oscar-vbox:~$ export oscar_variableambiente=C16042
oscar@oscar-vbox:~$ printenv | grep oscar_variableambiente
oscar_variableambiente=C16042
```

Figura 7: Variable exportada y confirmación.

```
oscar@oscar-vbox:~$ oscar_variablenoexport=C16042v2
oscar@oscar-vbox:~$ echo $oscar_variablenoexport
C16042v2
```

Figura 8: Variable no exportada y confirmación.

Para la variable exportada se utiliza export, la variable y su valor. Para la variable no exportada se declara con su valor solamente.

```
oscar@oscar-vbox:~$ python3
Python 3.10.12 (main, Jun 11 2023, 05:26:28) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from os import getenv
>>> variable_ambiente = "oscar_variableambiente"
>>> valor_ambiente = getenv(variable_ambiente)
>>> print('El valor de la variable de ambiente {} es {}'.format(variable_ambient
e, valor_ambiente))
El valor de la variable de ambiente oscar_variableambiente es C16042
>>> variable_noexport = "oscar_variablenoexport"
>>> valor_noexport = getenv(variable_noexport)
>>> print('El valor de la variable no exportada {} es {}'.format(variable_noexpo
rt, valor_noexport))
El valor de la variable no exportada oscar_variablenoexport es None
```

Figura 9: Creación de código de python con ambas variables a trabajar.

Lo que sucede es que cuando se declaran variables en la línea de comandos, solo aquellas marcadas con export se vuelven accesibles para otros procesos, como los scripts de Python, ya que se convierten en variables de entorno.

3. Se debe explicar la diferencia entre stdout y stderr.

Se utiliza stdout para mostrar resultados normales de un programa, mientras que stderr se emplea para emitir mensajes de error. La separación de estos dos canales permite dirigir la salida normal y los mensajes de error a destinos diferentes si se desea.

4. Se deben utilizar ejemplos para demostrar cómo se puede redirigir a un archivo:

a. solamente stdout

Para redirigir stdout, simplemente se utiliza el símbolo >. Este símbolo redirige la salida estándar de un comando a un archivo, reemplazando cualquier contenido existente en dicho archivo. Como se muestra en la Figura (10), una vez realizada la redirección y tras leer el archivo, se constató que la información fue almacenada correctamente. Si se quiere evitar la sobreescritura del contenido, basta con usar ≫.

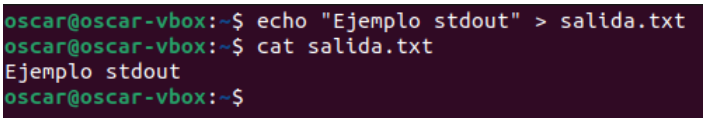

Figura 10: Redirigir solamente stdout.

b. solamente stderr

Para realizar una redirección de stderr, se emplea el símbolo "2>". Este símbolo conduce la salida de error, o stderr, hacia el archivo señalado. Al acceder al archivo, se puede confirmar que los datos de error han sido guardados correctamente. Si se busca preservar el contenido original del archivo, es necesario usar el símbolo "2>>". Este proceso está representado en la Figura (11).

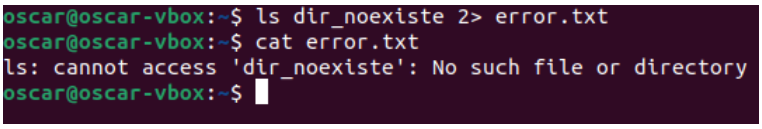

Figura 11: Redirigir solamente stderr.

c. ambos

Para desviar tanto stdout como stderr a un solo archivo, se emplea el símbolo "&>". Este símbolo posibilita dirigir ambas salidas, stdout y stderr, al archivo designado. Se confirma que funciona al ponerle la salida de stdout y un error de un documento que no existe. Y los dos funcionan perfectamente.

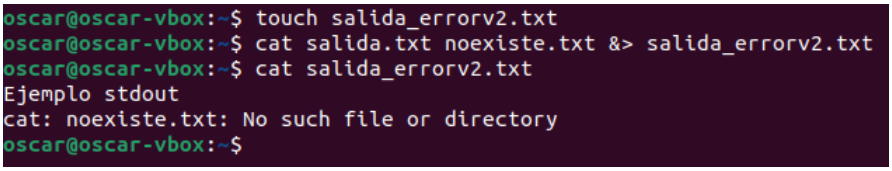

Figura 12: Redirigir stout y stderr.

3. Tercera Parte.

1. (10 pt) Se debe crear un programa (conocido como "script") de bash (recordar la línea she-bang, o "`#!/usr/bin/bash)` ". El programa debe recibir un parámetro, el propósito del programa es escribir en terminal (stdout) el mensaje proporcionado en el enunciado. Se debe guardar el programa en un archivo "`miprograma.bash` "dentro del directorio "home". Se deben dar permisos de ejecución para todos. Se debe ejecutar el programa, y mostrar la salida.

```
oscar@oscar-vbox:~$ nano /home/oscar/miprograma.bash
oscar@oscar-vbox:~$ sudo chmod +x /home/oscar/miprograma.bash
oscar@oscar-vbox:~$ ./miprograma.bash Oscar Oscar
Error: Solo se debe ingresar un argumento.
oscar@oscar-vbox:~$ ./miprograma.bash Oscar
Hoy es lunes, 25 de septiembre, del año 2023, y mi nombre es Oscar. El número de
 inodo de este script es 527444.
oscar@oscar-vbox:~$
```

Figura 13: Creación de miprograma.bash y verificación de funcionamiento.

```
#!/bin/bash

# Verificar el número de parámetros
if [ $# -ne 1 ]; then
    echo "Error: Solo se debe ingresar un argumento."
    exit -1
fi

# Guardar el parámetro en una variable
nombre="$1"

# Obtener fecha y tiempo
dia_semana=$(date +%A)
dia_mes=$(date +%d)
mes=$(date +%B)
ano=$(date +%Y)

# Obtener el número de inodo
num_inodo=$(ls -i $0 | cut -d " " -f 1)

# Imprimir el mensaje
echo "Hoy es $dia_semana, $dia_mes de $mes, del año $ano, y mi nombre es $nombre. El número de inodo de este script es $num_inodo."
```

Figura 14: Creación del código del script.

En la Figura (13) se puede apreciar la creación de mi programa.bash y luego como se le otorga permisos de ejecución. Se corrobora que tire error si se ingresan dos argumentos y que funcione si solo se coloca uno.

En el script de la Figura (14), se empieza por verificar la cantidad de parámetros ingresados durante su ejecución. Si no se introduce exactamente un parámetro, el script terminará su ejecución inmediatamente, retornando un código de error -1 y mostrando un mensaje de error en la terminal que indica que solo se debe ingresar un argumento. Si se introduce un parámetro correctamente, este se almacena en una variable llamada "nombre". Posteriormente, el script recoge información acerca de la fecha actual, separando el día de la semana, el día del mes, el mes, y el año en variables diferentes. Además, se obtiene el número de inodo del archivo del script mediante comandos de lista y corte de texto; el comando "`ls -i $0`" lista el archivo del script actual, representado por "`$0`", y "`-i`" hace que se muestre el número de inodo del archivo junto con su nombre. Posteriormente, el comando "`cut -d " " -f 1`" procesa esta

salida: "`cut`" se usa para dividir una línea de texto, "`-d " "` " especifica que el delimitador es un espacio, y "`-f 1`" selecciona el primer campo de la línea, que en este caso es el número de inodo. Finalmente, se imprime un mensaje en la terminal que incorpora toda la información recopilada, incluyendo la fecha actual, el nombre introducido como parámetro y el número de inodo del script.

2. (5 pt) Se debe editar el crontab del sistema, agregando el nuevo programa, y redirigiendo su salida al archivo "`/home/<SU USUARIO>/bitacora.log` "(usar >>). Dentro de crontab, se debe añadir la siguiente línea (sustituir USUARIO, y NOMBRE por los valores correctos, se puede usar cualquier nombre o lugar):

"`* * * * * /home/<USUARIO>/miprograma.sh NOMBRE >> /home/< USUARIO>/bitacora.log`"

Se debe esperar unos minutos, y mostrar los contenidos del archivo bitacora.log. Se debe explicar brevemente qué sucede.

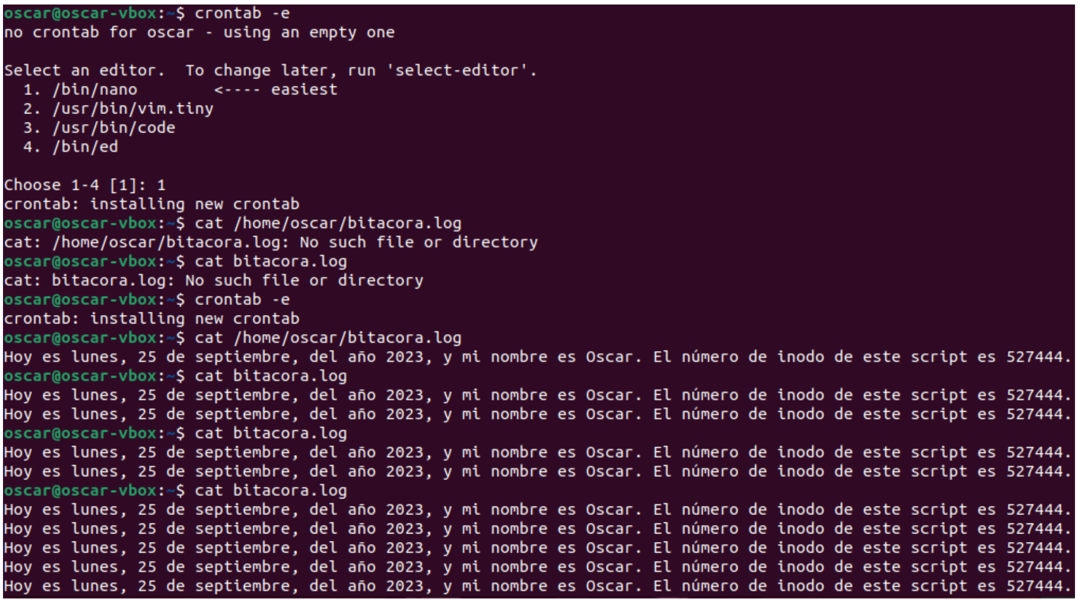

Figura 15: Edición del crontab -e y verificación.

La línea que se añadió a crontab programa el script de Bash para que se ejecute cada minuto. Cada vez que se ejecuta el script, la salida se añade al archivo bitacora.log en el directorio home correspondiente. Se especificó que el script se ejecute cada minuto (* * * * *), por lo que se verá una nueva entrada en el archivo bitacora.log cada minuto.

3. (2 pt) Se debe modificar la línea del cron del punto anterior, pero esta vez cambiar la periodicidad para ejecutar de jueves a domingo, a las 7:45pm, durante el mes de setiembre. Se debe mostrar la línea modificada. No es necesario mostrar los contenidos de "`bitácora.log` "para este punto.

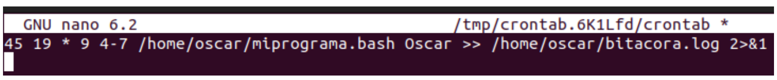

Figura 16: Cambio al crontab para que ejecute el script en la hora solicitada.

La línea añadida en crontab está programada para ejecutar el script miprograma.bash residente en el directorio home de oscar, específicamente a las 7:45 PM, de jueves a domingo, durante todo el mes de septiembre. Cada ejecución del script resultará en que la salida del script, junto con cualquier error que pueda surgir durante su ejecución, sea registrada y añadida al archivo bitacora.log, también localizado en el directorio home de oscar.