

Universidad Tecnológica de Panamá
Facultad de Ingeniería en Sistemas Computacionales
Departamento de Arquitectura y Redes de Comunicación

Laboratorio SHELL SCRIPTS

Estudiante: Manuel Villanueva
Prof. Aris Castillo de Valencia

Cédula: E-8-172111

OBJETIVO:

- El objetivo de este documento es practicar en la creación de shell scripts bash.

MATERIALES

- Computadora, con maquina virtual y sistema operativo Linux

INFORMACION

Un **script** es un **archivo de ordenes** o pequeño **programa** con el que poder automatizar ciertas tareas o acciones en un ordenador. Intentaré explicarlo de forma más sencilla.

Un **script** suele ser un archivo de texto plano, en el que se insertan las órdenes que se quieren realizar. En Ubuntu (o **cualquier distribución Linux**) podríamos hacer un script con comandos de consola para poder, por ejemplo, actualizar el sistema, instalar alguna aplicación, hacer un **backup de archivos**, o todo a la vez. El contenido de un **script** que nos hiciese un **backup** de nuestras carpetas importantes y lo guardase en un **pendrive** podría ser del siguiente modo:

```
#!/bin/bash
cd /media/pendrive
tar -cvzf documentos.tar.gz /home/usuario/Documentos/*
tar -cvzf videos.tar.gz /home/usuario/Videos/*
tar -cvzf imagenes.tar.gz /home/usuario/Imagenes/*
tar -cvzf descargas.tar.gz /home/usuario/Descargas/*
```

De este modo, podríamos guardar y comprimir todo el contenido de nuestras carpetas importantes con un solo click o ejecutandolo en consola así: **./script.sh**

Los scripts no sólo usan **comandos Linux**, hay de muchos tipos y para todos los sistemas operativos. En Windows, por ejemplo, el tipo de scripts más común son los archivos ***.bat**.

En Linux lo más común es el **Shell Script**, scripts con comandos de **consola Linux**. Pero los scripts también pueden ser **scripts PHP**, python, java, etc.

```
cd $HOME && touch script.sh && chmod +x script.sh
cd $HOME && echo '#!/bin/bash' > script.sh && echo '# -*- ENCODING: UTF-8 -*-' >> script.sh
```

```
#!/bin/bash
# -*- ENCODING: UTF-8 -*-
echo "<° Linux es lo mejor"
exit
```

Para ejecutar el script:
./script.sh

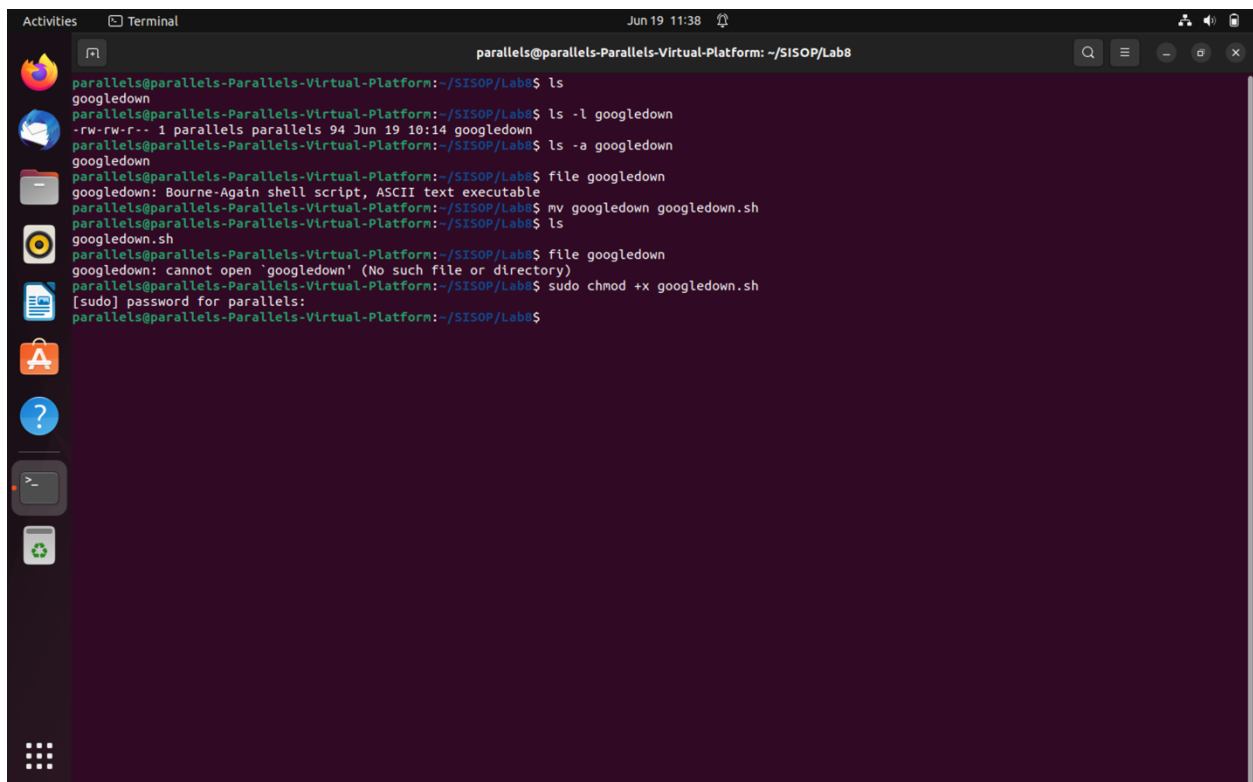
PROCEDIMIENTO

1. Verificar conectividad con un sitio web:

Primero debe entrar a un editor, escribir los comandos y guardar el archivo con extensión sh

```
#!/bin/bash
ping -c 3 www.google.com > /dev/null
if [ $? != 0 ]
then
echo "Google is down"
fi
```

- Después en la línea de comandos debe cambiar el script a modo ejecución:
chmod +x nombredelscript
- Después puede ejecutar el script colocando:
./nombredelscript.sh

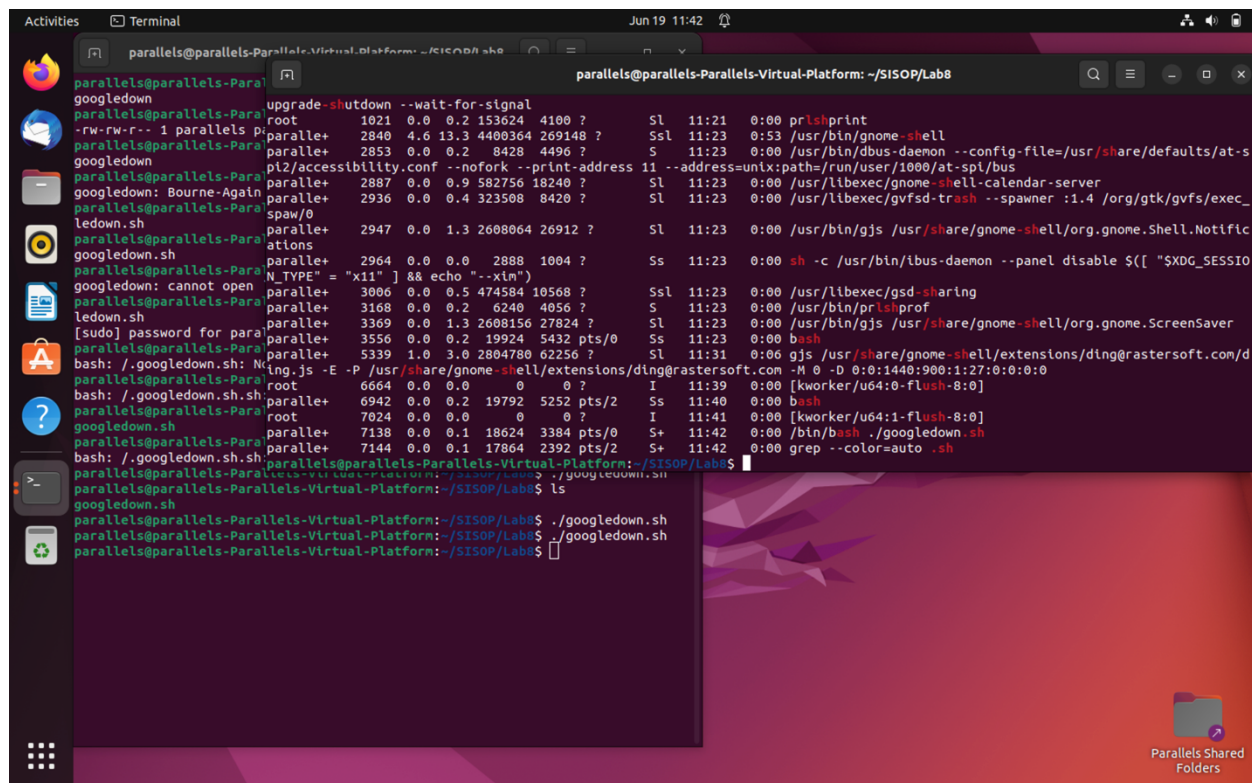


```
parallels@parallels-Parallels-Virtual-Platform: ~/SISOP/Lab8
parallels@parallels-Parallels-Virtual-Platform:~/SISOP/Lab8$ ls
googledown
parallels@parallels-Parallels-Virtual-Platform:~/SISOP/Lab8$ ls -l googledown
-rw-rw-r-- 1 parallels parallels 94 Jun 19 10:14 googledown
parallels@parallels-Parallels-Virtual-Platform:~/SISOP/Lab8$ ls -a googledown
googledown
parallels@parallels-Parallels-Virtual-Platform:~/SISOP/Lab8$ file googledown
googledown: Bourne-Again shell script, ASCII text executable
parallels@parallels-Parallels-Virtual-Platform:~/SISOP/Lab8$ mv googledown googledown.sh
parallels@parallels-Parallels-Virtual-Platform:~/SISOP/Lab8$ ls
googledown.sh
parallels@parallels-Parallels-Virtual-Platform:~/SISOP/Lab8$ file googledown
googledown: cannot open 'googledown' (No such file or directory)
parallels@parallels-Parallels-Virtual-Platform:~/SISOP/Lab8$ sudo chmod +x googledown.sh
[sudo] password for parallels:
parallels@parallels-Parallels-Virtual-Platform:~/SISOP/Lab8$
```

2. Verificar que un servicio/proceso se está ejecutando

Si queremos saber si existen procesos de algún servicio o programa ejecutándose podríamos usar este script (en este caso verifica que el demonio de *Apache* esté corriendo):

```
#!/bin/sh
SERVICE='httpd'
if ps ax | grep -v grep | grep $SERVICE > /dev/null
then
echo "El servicio $SERVICE esta ejecutandose"
else
echo "El servicio $SERVICE esta detenido"
fi
```



The screenshot shows a terminal window with the following output:

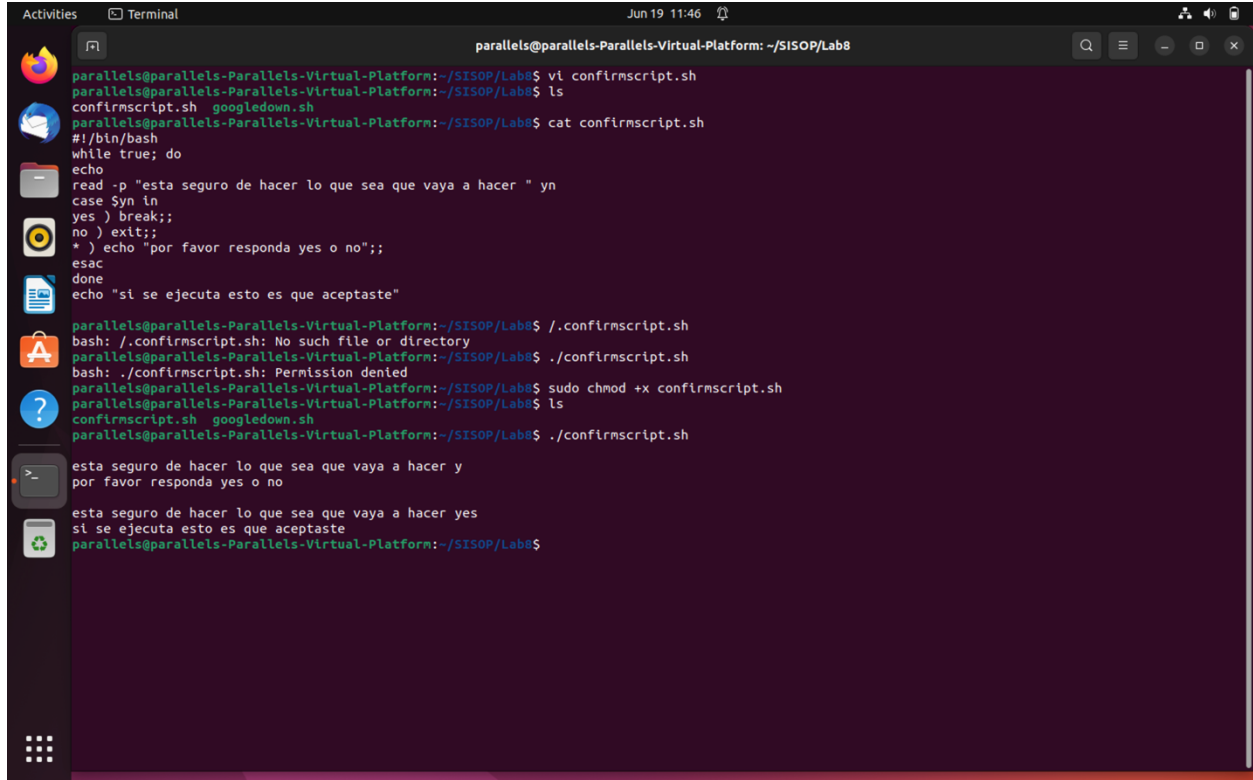
```
parallels@parallels-Parallels-Virtual-Platform: ~/SISOP/Lab8
googleltdown
parallels@parallels-Parallels-Virtual-Platform: ~/SISOP/Lab8$ ps ax | grep -v grep | grep httpd
root      1021  0.0  0.2 153624  4100 ?        Ssl   11:21   0:00 prishprint
parallels 2840  4.6 13.3 4400364 269148 ?        Ssl   11:23   0:53 /usr/bin/gnome-shell
parallels 2853  0.0  0.2   8428  4496 ?        S     11:23   0:00 /usr/bin/dbus-daemon --config-file=/usr/share/defaults/at-s
googleltdown
parallels 2887  0.0  0.9 582756 18240 ?        Ssl   11:23   0:00 /usr/libexec/gnome-shell-calendar-server
googleltdown: Bourne-Again shell
parallels 2936  0.0  0.4 323508  8420 ?        Ssl   11:23   0:00 /usr/libexec/gvfsd-trash --spawner :1.4 /org/gtk/gvfs/exec_
parallels 2947  0.0  1.3 2608064 26912 ?        Ssl   11:23   0:00 /usr/bin/gjs /usr/share/gnome-shell/org.gnome.Shell.Notific
parallels 2964  0.0  0.0   2888  1004 ?        Ss    11:23   0:00 sh -c /usr/bin/ibus-daemon --panel disable ${XDG_SESSIO
parallels 3006  0.0  0.5 474584 10568 ?        Ssl   11:23   0:00 /usr/libexec/gsd-sharing
parallels 3168  0.0  0.2   6240  4056 ?        S     11:23   0:00 /usr/bin/prishprof
googleltdown.sh
parallels 3369  0.0  1.3 2608156 27824 ?        Ssl   11:23   0:00 /usr/bin/gjs /usr/share/gnome-shell/org.gnome.ScreenSaver
parallels 3556  0.0  0.2 19924   5432 pts/0    Ss    11:23   0:00 bash
parallels 5339  1.0  3.0 2804780 62256 ?        Ssl   11:31   0:06 gjs /usr/share/gnome-shell/extensions/ding@rastersoft.com/d
parallels 6664  0.0  0.0     0     0 ?        I     11:39   0:00 [kworker/u64:0-flush-8:0]
parallels 6942  0.0  0.2 19792   5252 pts/2    Ss    11:40   0:00 bash
parallels 7024  0.0  0.0     0     0 ?        I     11:41   0:00 [kworker/u64:1-flush-8:0]
googleltdown.sh
parallels 7138  0.0  0.1 18624   3384 pts/0    S+    11:42   0:00 /bin/bash ./googleltdown.sh
parallels 7144  0.0  0.1 17864   2392 pts/2    S+    11:42   0:00 grep --color=auto .sh
parallels@parallels-Parallels-Virtual-Platform: ~/SISOP/Lab8$ ls
googleltdown.sh
parallels@parallels-Parallels-Virtual-Platform: ~/SISOP/Lab8$ ./googleltdown.sh
parallels@parallels-Parallels-Virtual-Platform: ~/SISOP/Lab8$ ./googleltdown.sh
parallels@parallels-Parallels-Virtual-Platform: ~/SISOP/Lab8$
```

3. Pedir confirmación antes de ejecutar un script

A veces es útil hacer que el usuario confirme la ejecución de un lote de sentencias, es decir, el típico mensaje que pide al usuario escribir yes o no. Esto lo podemos hacer así:

```
#!/bin/bash
while true; do
echo
read -p "esta seguro de hacer lo que sea que vaya a hacer " yn
case $yn in
yes ) break;;
no ) exit;;
* ) echo "por favor responda yes o no";;
```

```
esac
done
echo "si se ejecuta esto es que aceptaste"
```



The screenshot shows a terminal window titled "parallels@parallels-Parallels-Virtual-Platform: ~/SISOP/Lab8". The user creates a file named `confirmscript.sh` using `vi`. The script's content is as follows:

```
#!/bin/bash
while true; do
  echo
  read -p "esta seguro de hacer lo que sea que vaya a hacer " yn
  case $yn in
    yes ) break;;
    no ) exit;;
    * ) echo "por favor responda yes o no";;
  esac
done
echo "si se ejecuta esto es que aceptaste"
```

The user then attempts to run the script with `./confirmscript.sh`, which fails with "No such file or directory" and "Permission denied". After running `sudo chmod +x confirmscript.sh`, the script is executed successfully, displaying the prompts and the final echo message.

4. Realice un Shell Script sobre operaciones matemáticas, en donde el usuario ingrese dos números enteros y mediante un menú escoja que operación desea realizar.

```
Activities Terminal Jun 19 11:49 parallels@parallels-Parallels-Virtual-Platform: ~/SISOP/Lab8

# Pregunta por la opción de operación
read -p "Ingresa tu elección (1-4): " choice

# Realiza la operación seleccionada
case $choice in
    1)
        result=$(echo "$num1 + $num2" | bc)
        operator="+"
        ;;
    2)
        result=$(echo "$num1 - $num2" | bc)
        operator="-"
        ;;
    3)
        result=$(echo "$num1 * $num2" | bc)
        operator="*"
        ;;
    4)
        result=$(echo "scale=2; $num1 / $num2" | bc)
        operator="/"
        ;;
    *)
        echo ";Opción inválida!"
        exit 1
        ;;
esac

# Muestra el resultado
echo "Resultado: $num1 $operator $num2 = $result"

parallels@parallels-Parallels-Virtual-Platform:~/SISOP/Lab8$ sudo chmod +x calscript.sh
parallels@parallels-Parallels-Virtual-Platform:~/SISOP/Lab8$ ls
calscript.sh confirmscript.sh googledown.sh
parallels@parallels-Parallels-Virtual-Platform:~/SISOP/Lab8$ ./calscript.sh
Ingresa el primer número: 2
Ingresa el segundo número: 4
Selecciona una operación:
1. Suma
2. Resta
3. Multiplicación
4. División
Ingresa tu elección (1-4): 3
Resultado: 2 * 4 = 8
parallels@parallels-Parallels-Virtual-Platform:~/SISOP/Lab8$
```

INFORME

1. Enliste comandos de red de Linux con sus funciones.
 - **ifconfig:** Muestra y configura información de red, como direcciones IP, máscaras de red y configuración de interfaz de red.
 - **ip:** Similar a ifconfig, muestra y configura información de red, como direcciones IP, rutas y configuración de interfaz de red. Es más moderno y poderoso que ifconfig.
 - **ping:** Envía un mensaje de solicitud de eco ICMP a una dirección IP específica para verificar la conectividad de red y medir el tiempo de respuesta.
 - **traceroute:** Muestra la ruta tomada por los paquetes de red desde tu dispositivo hasta un destino, mostrando cada salto y el tiempo de respuesta en cada uno.

- **netstat:** Muestra estadísticas de red, como conexiones activas, tablas de enrutamiento y puertos en uso.
- **ss:** Similar a netstat, muestra información detallada sobre las conexiones de red, puertos y sockets.
- **nslookup:** Realiza consultas de DNS (Sistema de Nombres de Dominio) para obtener información sobre registros de dominio, como direcciones IP asociadas a un nombre de dominio.
- **dig:** Similar a nslookup, realiza consultas de DNS más avanzadas y muestra información detallada sobre los registros de dominio.
- **wget:** Descarga archivos desde la web mediante el protocolo HTTP, HTTPS o FTP.
- **curl:** Permite transferir datos hacia o desde un servidor mediante varios protocolos, como HTTP, HTTPS, FTP y muchos otros.
- **ssh:** Inicia una sesión de shell segura en un servidor remoto mediante el protocolo SSH (Secure Shell).
- **scp:** Copia archivos de forma segura entre el sistema local y un servidor remoto utilizando el protocolo SSH.
- **nc:** Utilizado para leer y escribir datos desde y hacia conexiones de red utilizando cualquier protocolo.

- **route:** Muestra y configura la tabla de enrutamiento del sistema, que determina cómo se dirigen los paquetes de red.
- **iptables:** Configura el firewall de red en sistemas Linux para filtrar, redirigir o modificar el tráfico de red.

2. Muestre capturas que comprueben la realización de los scripts

¡Los scripts que realicé se pueden ver en las capturas de los puntos previos! ☺

3. Conclusión y recomendaciones

Esta actividad me gustó mucho ya que yo previamente he hecho scripts para diferentes funciones. Recomendaría que sería interesante agregar scripts sencillos para automatizar procesos dentro de Linux, o scripts maliciosos y como estos interactúan con el OS. Además adjunto el NSE (Nmap Scripting Engine), que es un repositorio de scripts de mapeo de redes que posee nmap y pueden ser usado con la instalación de nmap en el OS:

<https://nmap.org/book/nse.html>

REFERENCIA

<https://www.javatpoint.com/linux-networking-commands>

<https://blog.desdelinux.net/scripts-shell-utiles-en-cualquier-programa/>

<https://iterm2.com/documentation-scripting-fundamentals.html>