

Reporte de Actividad 4

Roberto Benard Orci

21/02/2018

1 Introducción

En esta actividad trabajamos con Bourne Again Shell (`\bin\bash`) para aprender a hacer programas (scripts) interpretes de comandos que recompilen, junten y clasifiquen informacion de manera automatizada.

Un Shell script es un programa que recibe lo que se escribe en la terminal y lo convierte en instrucciones para el sistema operativo. Se denomina shell porque es la capa más externa alrededor del núcleo del sistema operativo.

2 Actividad

Lo primero que hicimos fue descargar un script que entra a la página de U. de Wyoming para descargar los datos clima de una región en un año en 12 archivos, uno para cada mes. Después cambiamos el número de estación de manera que los datos que descargara el script fueran los datos de la misma región que escogimos la practica pasada.

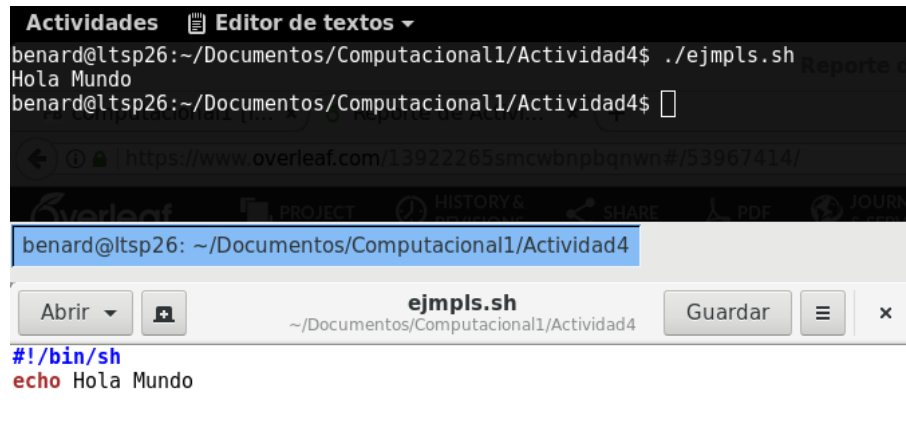
Finalmente probamos diferentes comandos que nos servirían para organizar, ver y filtrar archivos. Con la intención de usar estos comandos en un script que automatizara el proceso de organizar y filtrar los archivos.

Comandos utilizados

A continuación, daré una breve descripción sobre los comandos usados en esta práctica, estos están organizados en 3 secciones, los comandos se encuentran en la sección que mejor los describe aunque varios de estos comandos podrían estar en varias secciones a la vez.

echo

Este comando sirve para escribir algo desde la terminal a un archivo o viceversa.



```
Actividades Editor de textos
benard@ltsp26:~/Documentos/Computacional1/Actividad4$ ./ejmpls.sh
Hola Mundo
benard@ltsp26:~/Documentos/Computacional1/Actividad4$

https://www.overleaf.com/13922265smcwbnbpbnwn#/53967414/
PROJECT HISTORY & SHARE PDF JOURNALS
benard@ltsp26: ~/Documentos/Computacional1/Actividad4
ejmpls.sh
~/Documentos/Computacional1/Actividad4
Guardar x
#!/bin/sh
echo Hola Mundo
```

Obtención de datos

grep

Este comando tiene una gran cantidad de funciones, pero en esta práctica fue usado principalmente para descargar archivos en línea y para filtrar datos.

Organización de archivos

pwd

Sirve para obtener la ruta en donde se encuentra uno.

Herramientas para ver/filtrar contenido

chmod

Sirve para cambiar el acceso a un archivo para el usuario y demás usuarios, la "libertad" que tienen ver, leer y/o modificar el archivo.

ls -alg

chmod es el que cambia el acceso, *ls -alg* es el que te muestra estas "libertades" de todos los archivos de una carpeta.

```
benard@ltsp26: ~/Documentos/Computacional1/Actividad4$ ls -alg
total 28648
drwxr-xr-x 1 users 4896 feb 21 15:36 .
drwxr-xr-x 1 users 4896 feb 20 11:31 ..
-rw-r--r-- 1 users 474583 feb 21 11:27 df2017_2.csv
-rw-r--r-- 1 users 474583 feb 20 11:58 df2017.csv
-rwxr-xr-x 1 users 27 feb 21 15:36 ejempls.sh
-rw-r--r-- 1 users 5 feb 15 16:58 file1.txt
-rwxr-xr-x 1 users 221 feb 21 11:26 filtro.sh
-rwxr-xr-x 1 users 147 feb 21 15:16 .fuse_hidden000007900000001c
-rwxr-xr-x 1 users 1398 feb 15 17:08 script1.sh
-rwxr-xr-x 1 users 1416 feb 15 16:42 script1.sh
-rw-r--r-- 1 users 9444587 feb 21 11:27 sondeos1.txt
-rw-r--r-- 1 users 9444587 feb 19 11:44 sondeos.txt
-rw-r--r-- 1 users 779152 feb 15 17:08 sounding?region=naconf&TYPE=TEXT%3A1IST&YEAR=2017&MONTH=01&FROM=0100&TO=3112&STNM=74560
-rw-r--r-- 1 users 710943 feb 15 17:12 sounding?region=naconf&TYPE=TEXT%3A1IST&YEAR=2017&MONTH=02&FROM=0100&TO=3112&STNM=74560
-rw-r--r-- 1 users 798051 feb 15 17:08 sounding?region=naconf&TYPE=TEXT%3A1IST&YEAR=2017&MONTH=03&FROM=0100&TO=3112&STNM=74560
-rw-r--r-- 1 users 756160 feb 15 17:11 sounding?region=naconf&TYPE=TEXT%3A1IST&YEAR=2017&MONTH=04&FROM=0100&TO=3112&STNM=74560
-rw-r--r-- 1 users 757251 feb 15 17:08 sounding?region=naconf&TYPE=TEXT%3A1IST&YEAR=2017&MONTH=05&FROM=0100&TO=3112&STNM=74560
-rw-r--r-- 1 users 757225 feb 15 17:11 sounding?region=naconf&TYPE=TEXT%3A1IST&YEAR=2017&MONTH=06&FROM=0100&TO=3112&STNM=74560
-rw-r--r-- 1 users 829805 feb 15 17:09 sounding?region=naconf&TYPE=TEXT%3A1IST&YEAR=2017&MONTH=07&FROM=0100&TO=3112&STNM=74560
-rw-r--r-- 1 users 815703 feb 15 17:09 sounding?region=naconf&TYPE=TEXT%3A1IST&YEAR=2017&MONTH=08&FROM=0100&TO=3112&STNM=74560
-rw-r--r-- 1 users 915419 feb 15 17:11 sounding?region=naconf&TYPE=TEXT%3A1IST&YEAR=2017&MONTH=09&FROM=0100&TO=3112&STNM=74560
-rw-r--r-- 1 users 761843 feb 15 17:10 sounding?region=naconf&TYPE=TEXT%3A1IST&YEAR=2017&MONTH=10&FROM=0100&TO=3112&STNM=74560
-rw-r--r-- 1 users 782161 feb 15 17:12 sounding?region=naconf&TYPE=TEXT%3A1IST&YEAR=2017&MONTH=11&FROM=0100&TO=3112&STNM=74560
-rw-r--r-- 1 users 780874 feb 15 17:11 sounding?region=naconf&TYPE=TEXT%3A1IST&YEAR=2017&MONTH=12&FROM=0100&TO=3112&STNM=74560
benard@ltsp26: ~/Documentos/Computacional1/Actividad4$
```

less

Te muestra todo lo que está escrito en un archivo empezando desde el principio, es lo único que te muestra en la terminal, puedes terminar haciendo clic en q, y al regresar a la terminal normal no te muestra el archivo.

cat

De igual manera que *less*, *cat* te muestra el contenido de un archivo, solo que en la terminal te manda de inmediato al final del archivo y este si se queda en la terminal.

diff

Muestra la diferencia entre dos archivos.

```
benard@ltsp26: ~/Documentos/Computacional1/Actividad4$ diff df2017.csv df2017_2.csv
benard@ltsp26: ~/Documentos/Computacional1/Actividad4$
```

|

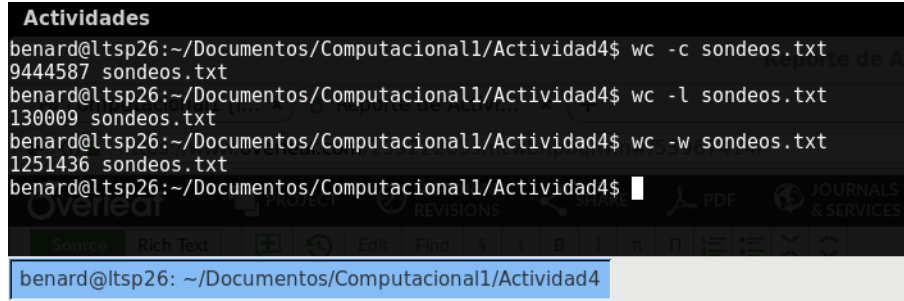
Este es un separador, sirve para separar palabras o caracteres a la hora de buscar o filtrar varios.

>

Sirve para mandar algo a un archivo, puede ser un conjunto de otros archivos, una palabra, número, etc.

wc

Sirve para contar. Puede contar el número de líneas de un archivo, el número de caracteres y los bits de este.



```
Actividades
benard@ltsp26:~/Documentos/Computacional1/Actividad4$ wc -c sondeos.txt
9444587 sondeos.txt
benard@ltsp26:~/Documentos/Computacional1/Actividad4$ wc -l sondeos.txt
130009 sondeos.txt
benard@ltsp26:~/Documentos/Computacional1/Actividad4$ wc -w sondeos.txt
1251436 sondeos.txt
benard@ltsp26:~/Documentos/Computacional1/Actividad4$
```

3 Notas de Steve Parker: Síntesis

En las referencias básicas de la practica aparece un link a un tutorial de Shell script con el cual pudimos crear el script que automatizara el proceso de hacer un archivo que tuviera los 12 archivos descargados y otro que filtrara este archivo.

A continuación, esta una síntesis de los capítulos que me sirvieron para poder entender cómo funcionaba el script descargado al inicio de la práctica, y los que también me ayudaron a crear el script antes mencionado.

3.1 Introduction

En esta sección se nos habla un poco de que son los scripts y como surgieron. También nos enseñan como mandar desde la terminal algo a un script y como imprimir texto en la terminal desde un script.

```
echo 'echo Hola Mundo' > Ejm.sh
```

```
#!/bin/sh
# Comentario
echo Hola Mundo 2.0
```

3.2 Philosophy

En esta sección se nos dice que para que un Shell script sea "bueno" este debe cumplir por lo menos con 2 características: que este "limpio" y fácil de leer, y evitar utilizar comandos innecesarios. Muestra un ejemplo de como un pequeño cambio en un código puede hacer que un programa corra más rápido.

```
#!/bin/sh
# Comentario
# Otro comentario
echo Hola Mundo

#!/bin/sh
echo Hola Mundo
```

3.3 A First Script

Esta sección habla de cómo se debe de empezar un script y el uso de comentarios, también muestra como al imprimir un mensaje en la terminal este puede cambiar dependiendo de la manera en que lo escribiste en el script.

```
#!/bin/sh
echo Hola Mundo

#!/bin/sh
echo "Holaaa      Mundooo"
```

3.4 Variables(Part 1)

Nos muestran el uso de variables y de las maneras en que estas pueden, si es que puede, interactuar entre sí, esto dependiendo del contenido de la variable.

```
#!/bin/sh
Mensaje1="Hola Mundo"
echo $Mensaje1

#!/bin/sh
Numero1=13
echo $Numero1
echo 19
echo 13 +19 =
expr $Numero1 + 19
```

3.5 Wildcards

Esta sección habla de como el uso de una sintaxis diferente afecta el resultado del script.

```
#!/bin/sh
COPY=`cp /home/benard/Documentos/Computacional1/Actividad4/Ejemplos/*
.txt /home/benard/Documentos/Computacional1/Actividad4/EjmRes/`
echo $COPY
```

3.6 Escape Characters

Esta sección nos enseña como imprimir caracteres que normalmente son usados como parte del código.

```
#!/bin/sh
echo "Hello  \"World\""
```

```
#!/bin/sh
echo "*"
```

3.7 Loops

En esta sección se nos introduce a los bucles (*loops*), que son herramientas que se usan para repetir un proceso varias veces en lugar de escribir el mismo código varias veces.

```
#!/bin/sh
for i in 5 4 3 2 1
do
    echo "Autodestrucción en $i"
done

#!/bin/sh
INPUT=Hallo
while [ "$INPUT" != "117" ]
do
    echo "Ingresa algo (117 para salir)"
    read INPUT
    echo "-----> $INPUT <-----"
done
```

3.8 Test

Esta sección nos enseña el uso de los corchetes [], los cuales son un "programa" como ls, etc.

```
#!/bin/sh
echo "hi :]"
read X
if [ "$X" = "hi :]" ]
then
    echo ":D"
else
    echo "D:"
fi
```

```
#!/bin/sh
echo Escribe un numero
read X
[ $X -ne 0 ] && echo "X isn't zero" || echo "X is zero"
```

3.9 Case

En esta sección nos enseñan la manera en la que se escribe un pequeño código de *Case*. Con estos uno puede prepararse para diferentes escenarios.

```
#!/bin/sh
echo "¿Como estas?"
while :
do
    read respuesta
    case $respuesta in
bien)
echo "Sehr gut!"
;;
mal)
echo "):"
echo "Why?"
;;
bye)
echo
echo "bon voyage!"
break
;;
adios)
echo "bon voyage!"
break
;;
*)
echo "Soy un robot, no te entiendo"
echo ":/"
;;
    esac
done

#!/bin/sh
echo "4 + 3 x 5 - 2= ??"
while :
do
    read respuesta
    case $respuesta in
17)
```

```

echo "Sehr gut!"
echo
break
;;
58)
echo "No, sigue la jerarquía de operaciones."
;;
36)
echo "No, sigue la jerarquía de operaciones."
;;
13)
echo "No, sigue la jerarquía de operaciones."
;;
*)
echo "How??"
;;
    esac
done

```

3.10 Variables(Part 2)

En esta sección nos muestran un conjunto de variables predeterminadas a las que no se les puede cambiar el valor.

```

#!/bin/sh
echo "I was called with $# parameters"
echo "My name is $0"
echo "My first parameter is $1"
echo "My second parameter is $2"
echo "All parameters are @$@"

#!/bin/sh
old_IFS="$IFS"
IFS=:
echo "Please input some data separated by colons (:)"
read x y z
IFS=$old_IFS
echo "x is $x y is $y z is $z"

```

3.11 Variables(Part 3)

En esta sección nos enseñan como escribir en la terminal sin que automaticamente se le agregue un *linebreak* al texto.

```

#!/bin/sh
echo -en "What is your name [ `whoami` ] "
read myname

```



```

if [ -z "$myname" ]; then
    myname=`whoami`
fi
echo "Your name is : $myname"

#!/bin/sh
echo -en "What is your name [ `whoami` ] "
read myname
echo "Your name is : ${myname:-`whoami`}"

```

3.12 External Programs

En esta sección se nos introduce a grep, uno de los programas "externos" que son comando que no están incorporados en Shell. Este lo usamos principalmente para filtrar, aunque tiene mucho otros usos.

```

#!/bin/sh
GREP=`grep -w PRES sounding\?region\=naconf\&TYPE\=TEXT%3ALIST\
&YEAR\=2017\&MONTH\=01\&FROM\=0100\&TO\=3112\&STNM\=74560`
echo $GREP

#!/bin/sh
GREP=`grep -R "Texto de prueba" /home/benard/Documentos/
Computacional1/Actividad4`
echo $GREP
#Aparecera este archivo

```

3.13 Functions

En esta sección nos muestran que los Shell scripts tambien pueden ser muy utiles a la hora de hacer funciones, e incluye varios ejemplos.

```

#!/bin/sh
# A simple script with a function...

add_a_user()
{
    USER=$1
    PASSWORD=$2
    shift; shift;
    # Having shifted twice, the rest is now comments ...
    COMMENTS=$@
    echo "Adding user $USER ..."
    echo useradd -c "$COMMENTS" $USER
    echo passwd $USER $PASSWORD
    echo "Added user $USER ($COMMENTS) with pass $PASSWORD"
}

```

```

###
# Main body of script starts here
###
echo "Start of script..."
add_a_user bob letmein Bob Holness the presenter
add_a_user fred badpassword Fred Durst the singer
add_a_user bilko worsepassword Sgt. Bilko the role model
echo "End of script..."

#!/bin/sh
echo Factorial
factorial()
{
    if [ "$1" -gt "1" ]; then
        i=`expr $1 - 1`
        j=`factorial $i`
        k=`expr $1 \* $j`
        echo $k
    else
        echo 1
    fi
}

while :
do
    echo "Enter a number:"
    read x
    factorial $x
done

```

3.14 Hints and Tips

Como el nombre de esta sección sugiere, nos muestran unos cuantos tips al usar ciertos comandos.

```

#!/bin/sh
steves=`grep -i steve /etc/passwd | cut -d: -f1`
echo "All users with the word "steve" in their passwd"
echo "Entries are: "
echo "$steves" | tr ' ' '\012' | tr '[a-z]' '[A-Z]'

#!/bin/sh
host=127.0.0.1
port=23
login=steve

```

```

passwd=hellothere
cmd="ls /tmp"

echo open ${host} ${port}
sleep 1
echo ${login}
sleep 1
echo ${passwd}
sleep 1
echo ${cmd}
sleep 1
echo exit

```

3.15 Quick Reference

Esta sección es nos muestran una lista de comandos y variables con una pequeña descripción de que hacen o que significan.

4 Conclusión

Los programas (scripts) pueden ser de gran ayuda al automatizar diferentes procesos a la hora de trabajar con una gran cantidad de datos. No solo hacen el proceso de obtener, organizar y filtrar datos más fácil, sino mucho más rápido en comparación a hacerlo de otra manera.

5 Bibliografía

Shell Scripting Tutorial. (n.d.). Retrieved February 21, 2018, from <https://www.shellscript.sh/external.html>

Shell script. (2018, February 18). Retrieved February 21, 2018, from https://en.wikipedia.org/wiki/Shell_script

6 Apéndice

1.- ¿Qué fue lo que más te llamó la atención en esta actividad?

Lo sencillo que es el crear un script que automatice diferentes procesos.

2.- ¿Qué consideras que aprendiste?

Varios comandos nuevos para las terminales y el uso de Shell scripts para automatizar el proceso de recolectar, organizar, y seleccionar información.

3.- ¿Cuáles fueron las cosas que más se te dificultaron?

Al principio pensé que sería difícil el crear el script que automatizara los puntos 13, 14 y 15 de la práctica, pero después de leer tutorial de Shell scripting me di cuenta que era mucho más fácil de lo parecía.

4.- ¿Cómo se podría mejorar en esta actividad?

En general, me pareció muy bien explicada.

5.- ¿En general, cómo te sentiste al realizar en esta actividad?

Al principio un poco extrañado, pero este sentimiento se fue rápidamente.