



5

Instituto Politécnico Nacional
Escuela Superior de Computo

Sistemas operativos (2CM9)

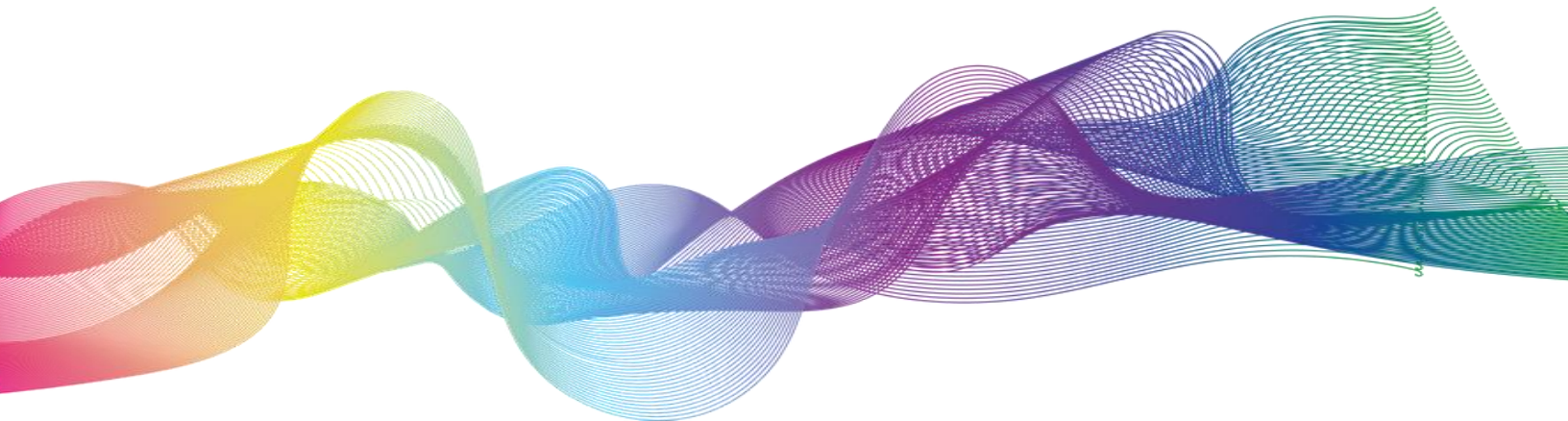
Maestra Ana Belem Juárez Méndez

Practica 1: Introducción a Shell script y comandos UNIX

Alumno:

García González Aarón Antonio

Agosto 29, 2019



Índice

Objetivo 3

Introducción 3

Desarrollo 5

 Ejercicio 1 5

 Ejercicio 2 8

 Ejercicio 3 9

 Ejercicio 4 10

 Ejercicio 5 11

Conclusiones 13

Referencias..... 13

Objetivo

Conocer algunos de los comandos fundamentales de UNIX y aprender a desarrollar Shell scripts básicos.

Introducción

Una de las ventajas de Linux es la cantidad de opciones a la hora de personalizar el sistema. Una de las más interesantes es la automatización de tareas en Linux con Bash. Descubrí cómo utilizar desde Linux el intérprete Bash y cómo crear programas que ayudan a automatizar tareas repetitivas.

Para automatizar las tareas en Linux la mejor herramienta es Bash. Lo primero que tienes que hacer es crear un 'script'. Un 'script' es un fichero de texto plano que contiene una lista de comandos de un intérprete, en este caso Bash.

A continuación, algunos de los comandos que encontré en la marcha en el desarrollo de esta practica:

sed	Para imprimir en consola la línea específica desde un archivo
cut	Para imprimir una palabra completa, desea -f 1, no -c 1. Y como el delimitador de campo predeterminado es TAB en lugar de SPACE, debe usar la opción -d, fue usado en el ejercicio 5.
rmdir	Eliminar un directorio.
chmod	Cambia los permisos de un archivo o directorio. El modo consiste en 3 partes. dueño, grupos, y otros esto hace referencia a los permisos para estos modos y debes especificarlos. Los permisos son los siguientes: ⇒ Read (Lee) =4 ⇒ Write (Escribir) = 2 ⇒ Execute (Ejecutar) =1
Tar	Combina distintos archivos en un archive comprimido si lo deseas. Parámetros ⇒ -c Crea un nuevo archivo. ⇒ -z Comprime el archivo utilizando paquetes gzip. ⇒ -j Comprime el archivo utilizando paquetes bzip2. ⇒ -v Modo Verbo se significa que puedes mostrar archivos procesados. ⇒ -f Escribe la salida a un archivo y no a la pantalla. ⇒ -x Desempaca archivos comprimidos.
Grep	Busca una cadena de caracteres en los archivos especificados y muestra que líneas contienen la cadena especificada.

	<p>Parámetros</p> <ul style="list-style-type: none"> ⇒ -R Búsqueda recursiva dentro de subdirectorios si existen. ⇒ -i Búsqueda que ignora las mayúsculas. ⇒ -l Muestras el nombre del archivo, no las líneas de textos.
reboot	Reinicia el sistema inmediatamente. Solo escribe reboot.
halt	Apaga el sistema, pero se asegura de cerrar todos tus archivos para evitar la pérdida de datos.

Desarrollo

Ejercicio 1

Investiga el uso de los siguientes comandos y caracteres especiales, y muestra su funcionamiento en una terminal.

Cp: Este comando sirve para copiar archivos y directorios.	<pre>MacBook-Pro-de-Aaron:p1 aarongarcia\$ ls ejemplos pract1.pdf practica_1.docx MacBook-Pro-de-Aaron:p1 aarongarcia\$ cp pract1.pdf ejemplos MacBook-Pro-de-Aaron:p1 aarongarcia\$ ls ejemplos pract1.pdf practica_1.docx MacBook-Pro-de-Aaron:p1 aarongarcia\$</pre>
-b	Crea un backup en el destino en el caso en el que exista un archivo llamado igual que el que queremos generar.
-f	Fuerza el borrado de los archivos destino sin consultar o avisar al usuario.
-i	Informa antes de sobrescribir un archivo en el destino indicado.
-l	Realiza un link en vez de copiar los ficheros.
-p	Realiza la copia de los ficheros y directorios conservando la fecha de modificación de los archivos y carpetas originales.
-r	Copia de forma recursiva.
-u	El comando cp en Linux no copia un archivo o directorio a un destino si este destino tiene la misma fecha de modificación o una fecha de modificación posterior comparándola con el archivo o directorio que queremos mover.
-v	Muestra lo que se está ejecutando.
Mv: se utiliza para mover archivos y directorios de una ubicación a otra, también es utilizado para renombrar tanto archivos como directorios	<pre>MacBook-Pro-de-Aaron:ejemplos aarongarcia\$ mv pract1.pdf practica1.pdf MacBook-Pro-de-Aaron:ejemplos aarongarcia\$ ls practica1.pdf MacBook-Pro-de-Aaron:ejemplos aarongarcia\$ MacBook-Pro-de-Aaron:ejemplos aarongarcia\$ mv pract1.pdf ./carpeta MacBook-Pro-de-Aaron:ejemplos aarongarcia\$ ls carpeta MacBook-Pro-de-Aaron:ejemplos aarongarcia\$ cd carpeta MacBook-Pro-de-Aaron:carpeta aarongarcia\$ ls pract1.pdf</pre>
Date: Gestión de fecha y hora de sistema, se puede ver y establecer	<pre>MacBook-Pro-de-Aaron:carpeta aarongarcia\$ date Sun Aug 25 23:44:39 CDT 2019</pre>
Cal: Se imprime en la terminal el mes en que estamos. Como opción estándar, el día en que nos encontramos aparece de forma más visible para distinguirse de los demás.	<pre>MacBook-Pro-de-Aaron:carpeta aarongarcia\$ ncal grep Su Su 4 11 18 25 MacBook-Pro-de-Aaron:carpeta aarongarcia\$ cal August 2019 Su Mo Tu We Th Fr Sa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 MacBook-Pro-de-Aaron:carpeta aarongarcia\$ cal -j 2019 2019 January February Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa 6 7 8 9 10 11 12 34 35 36 37 38 39 40 13 14 15 16 17 18 19 41 42 43 44 45 46 47 20 21 22 23 24 25 26 48 49 50 51 52 53 54 27 28 29 30 31 55 56 57 58 59</pre>
Uname: El comando linux uname sirve para imprimir información del sistema linux. La opción -a nos del comando uname nos da toda la información disponible.	<pre>MacBook-Pro-de-Aaron:carpeta aarongarcia\$ uname Darwin MacBook-Pro-de-Aaron:carpeta aarongarcia\$ uname -a Darwin MacBook-Pro-de-Aaron.local 18.7.0: Darwin Kernel Version 18.7.0: Thu Jun 28 18:42:21 PDT 2019; root:xnu-4903.270.47~4/RELEASE_ARM_T8020 MacBookPro16,1</pre>
mkdir [parametros] [directorio] Este comando nos permite crear directorios en Linux, tanto con una ruta relativa como con una ruta absoluta. Esto cambios afectan al modo gráfico.	<pre>MacBook-Pro-de-Aaron:carpeta aarongarcia\$ mkdir carpeta1 carpeta2 MacBook-Pro-de-Aaron:carpeta aarongarcia\$ ls carpeta1 carpeta2 pract1.pdf</pre>

Who: muestra los usuarios de sistema que han iniciado una sesion.	<pre>MacBook-Pro-de-Aaron:carpeta aarongarcia\$ who aarongarcia console Aug 25 22:26 aarongarcia ttys000 Aug 25 22:26</pre>
Wc: calcula número de palabras y otros datos similares de un fichero.	<pre>MacBook-Pro-de-Aaron:carpeta aarongarcia\$ wc pract1.pdf 1117 7911 159085 pract1.pdf MacBook-Pro-de-Aaron:carpeta aarongarcia\$ wc -l pract1.pdf 1117 pract1.pdf</pre>
Ifconfig: interfaz config. configuración de interfaces de red, modems, etc.	<pre>MacBook-Pro-de-Aaron:carpeta aarongarcia\$ ifconfig lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384 options=1203<RXCSUM, TXCSUM, TXSTATUS, SW_TIMESTAMP> inet 127.0.0.1 netmask 0xffff0000</pre>
Pstree: Muestra los procesos en forma de árbol	<pre>MacBook-Pro-de-Aaron:~ aarongarcia\$ pstree --+= 00001 root /sbin/launchd ---= 00039 root /usr/sbin/syslogd ---= 00040 root /usr/libexec/UserEventAgent (System)</pre>
Su: es utilizando para cambiar el usuario actual a otro usuario desde SSH, es decir, nos permite asumir la identidad de otro usuario (si conocemos su password claro)	<pre>MacBook-Pro-de-Aaron:~ aarongarcia\$ su Password: su: Sorry</pre>
Head: muestra las primeras líneas o bytes de uno o más archivos de texto. -n número: imprime el número indicado de líneas. -c número: imprime el número indicado de bytes.	<pre>MacBook-Pro-de-Aaron:ejemplos aarongarcia\$ head -c 34 man.txt Another one got caught today, it'sMacBook-Pro-de-Aaron:ejemplos aarongarcia\$ head -n 2 man.txt Another one got caught today, it's all over the papers. "Teenager Arrested in Computer Crime Scandal", "Hacker Arrested after Bank Tampering"...</pre>
Which: Nos sirve para averiguar donde se encuentra instalando un determinado programa	<pre>MacBook-Pro-de-Aaron:ejemplos aarongarcia\$ which usage: which [-as] program ... MacBook-Pro-de-Aaron:ejemplos aarongarcia\$ which -a find /usr/bin/find</pre>
W: muestra información sobre los usuarios que están conectados en ese momento a la máquina y sobre sus procesos. La cabecera muestra, en este orden, el tiempo actual, cuanto lleva el sistema funcionando, cuantos usuarios están conectados y las cargas medias en los anteriores 1, 5 y 15 minutos.	<pre>MacBook-Pro-de-Aaron:ejemplos aarongarcia\$ w 0:06 up 1 day, 1:40, 2 users, load averages: 2.38 1.99 1.96 USER TTY FROM LOGIN@ IDLE WHAT aarongarcia console - Sun22 25:39 - aarongarcia s000 - Mon22 - w</pre>
Sudo: permite a los usuarios ejecutar programas con los privilegios de seguridad de otro usuario (normalmente el usuario root) de manera segura, convirtiéndose así temporalmente en superusuario.	<pre>MacBook-Pro-de-Aaron:ejemplos aarongarcia\$ sudo usage: sudo -h -K -k -V usage: sudo -v [-AknS] [-g group] [-h host] [-p prompt] [-u user] usage: sudo -i [-AknS] [-g group] [-h host] [-p prompt] [-U user] [-u user] [command] usage: sudo [-AbEHknPS] [-C num] [-g group] [-h host] [-p prompt] [-u user] [VAR=value] [-i -s] [command] usage: sudo -e [-AknS] [-C num] [-g group] [-h host] [-p prompt] [-u user] file ...</pre>
Ps: permite visualizar el estado de un Proceso	<pre>MacBook-Pro-de-Aaron:ejemplos aarongarcia\$ ps PID TTY TIME CMD 3834 ttys000 0:00.28 -bash</pre>

	<pre>MacBook-Pro-de-Aaron:ejemplos aarongarcia\$ ps aux USER PID %CPU %MEM VSZ RSS Tt STAT STARTED TIME COMMAND root 51 33.2 0.1 4345768 11480 ?? Us Sun10PM 0:26.13 /usr/sbin/systemstats --d root 443 9.0 0.0 4385892 736 ?? Ss Sun10PM 0:03.96 /usr/libexec/sysmond root 1 4.3 0.1 4328748 11620 ?? Ss Sun10PM 1:22.00 /sbin/launchd</pre>
Grep: se utiliza para hacer coincidir e imprimir un patrón de búsqueda o una expresión regular de un solo archivo o varios archivos de texto. Buscará el patrón de los archivos mencionados e imprimirán el resultado en la pantalla o en un archivo de salida.	<pre>MacBook-Pro-de-Aaron:ejemplos aarongarcia\$ grep phone man.txt the phone line like heroin through an addict's veins, an electronic pulse is Damn kid. Tying up the phone line again. They're all alike...</pre>
Find: comando más común utilizado para encontrar y filtrar archivos. <ul style="list-style-type: none"> f – archivo normal d – directorio o carpeta l – enlace simbólico c – dispositivos de caracteres b – dispositivos de bloque 	<pre>MacBook-Pro-de-Aaron:ejemplos aarongarcia\$ find . -iname "*.txt" ./man.txt ./especificacion.txt</pre> <p>find <startingdirectory> <options> <search term></p>
Du: Muestra el espacio que esta ocupado en disco.	<pre>MacBook-Pro-de-Aaron:OneDrive - Instituto Politecnico Nacional aarongarcia\$ du 196624 ./probabilidad 32 ./algoritmos/ordenamiento 36952 ./algoritmos/p1 96184 ./algoritmos 104 ./so/practicas/lab 0 ./so/practicas/p1/ejemplos/carpeta/carpeta2 0 ./so/practicas/p1/ejemplos/carpeta/carpeta1 312 ./so/practicas/p1/ejemplos/carpeta 384 ./so/practicas/p1/ejemplos 24200 ./so/practicas/p1</pre>
Df: nos informa acerca del espacio total, ocupado y libre en nuestro sistema	<pre>Filesystem 512-blocks Used Available Capacity iused ifree %iused Mounted on /dev/disk1s1 236568496 99113352 131925144 43% 1135375 9223372036853640432 0% / devfs 373 373 0 100% 648 0 100% /dev /dev/disk1s4 236568496 4194392 131925144 4% 2 9223372036854775805 0% /private/var/vm map -hosts 0 0 0 100% 0 0 100% /net map auto_home 0 0 0 100% 0 0 100% /home MacBook-Pro-de-Aaron:practicas aarongarcia\$ df -h Filesystem Size Used Avail Capacity iused ifree %iused Mounted on /dev/disk1s1 113Gi 47Gi 63Gi 43% 1135375 9223372036853640432 0% / devfs 187Ki 187Ki 0Bi 100% 648 0 100% /dev /dev/disk1s4 113Gi 2.0Gi 63Gi 4% 2 9223372036854775805 0% /private/var/vm map -hosts 0Bi 0Bi 0Bi 100% 0 0 100% /net map auto_home 0Bi 0Bi 0Bi 100% 0 0 100% /home</pre>
Exit: termina la sesión del shell.	<pre>Last login: Tue Aug 27 17:31:12 on ttys000 MacBook-Pro-de-Aaron:practicas aarongarcia\$ exit logout Saving session... ...copying shared history... ...saving history...truncating history files... ...completed. [Proceso completado]</pre>
Cat: muestra todo el contenido de un fichero sin pausa alguna.	<pre>MacBook-Pro-de-Aaron:lab aarongarcia\$ ls a.out cont.sh fork.c parametros.sh xd.sh archivo.txt fork hola.sh prueba.c MacBook-Pro-de-Aaron:lab aarongarcia\$ cat prueba.c int main(){ printf("%gf\n");</pre>

<p>Rm: se usa para eliminar objetos como archivos, directorios, enlaces simbólicos, etc.</p>	<pre>MacBook-Pro-de-Aaron:lab aarongarcia\$ ls a.out cont.sh fork.c parametros.sh xd.sh archivo.txt fork hola.sh prueba.c MacBook-Pro-de-Aaron:lab aarongarcia\$ rm a.out MacBook-Pro-de-Aaron:lab aarongarcia\$ ls archivo.txt cont.sh fork fork.c hola.sh MacBook-Pro-de-Aaron:lab aarongarcia\$</pre>
<p>Sort: se utiliza para ordenar líneas de texto a partir de varios criterios, su sintaxis es similar a la de todos los comandos: sort [opción...] [archivo...]</p>	<pre>MacBook-Pro-de-Aaron:lab aarongarcia\$ cat fork.c sort //sleep(30); exit(-1); exit(0); // proceso terminado de manera exitosa perror("\nError en la llamada fork"); pidh = wait(NULL); // obtener variable de terminacion del hijo printf("\nHijo > El pid de mi padre es %d", getppid()); printf("\nHijo > El pid es %d", getpid()); printf("\nHijo > El valor de num es %d", num);</pre>
<p>>: redirige la salida a un archivo, sobrescribiendo el archivo</p>	<pre>MacBook-Pro-de-Aaron:lab aarongarcia\$ cat fork.c >> xd.txt MacBook-Pro-de-Aaron:lab aarongarcia\$ ls archivo.txt fork hola.sh prueba.c xd.txt cont.sh fork.c parametros.sh xd.sh MacBook-Pro-de-Aaron:lab aarongarcia\$ cat xd.txt #include<stdio.h> #include<unistd.h> #include<sys/types.h> #include<stdlib.h> #include<sys/wait.h></pre>
<p>>>: redirige la salida a un archivo que agrega la salida redirigida al final.</p>	<pre>MacBook-Pro-de-Aaron:lab aarongarcia\$ cat archivo.txt >> xd.txt MacBook-Pro-de-Aaron:lab aarongarcia\$ ls archivo.txt fork hola.sh prueba.c xd.txt cont.sh fork.c parametros.sh xd.sh</pre>
<p><: redireccionando la entrada estándar</p>	<pre>MacBook-Pro-de-Aaron:lab aarongarcia\$ wc < archivo.txt 6 47 312</pre>
<p><<: como una instrucción para leer la entrada hasta que encuentra una línea que contiene el delimitador especificado. Todas las líneas de entrada hasta la línea que contiene el delimitador se alimentan a la entrada estándar del comando.</p>	
<p> : la salida del comando anterior pasa a ser parámetro del comando siguiente</p>	<pre>MacBook-Pro-de-Aaron:lab aarongarcia\$ cat fork.c sort //sleep(30); exit(-1); exit(0); // proceso terminado de manera exitosa perror("\nError en la llamada fork"); pidh = wait(NULL); // obtener variable de terminacion del hijo printf("\nHijo > El pid de mi padre es %d", getppid()); printf("\nHijo > El pid es %d", getpid()); printf("\nHijo > El valor de num es %d", num);</pre>

Ejercicio 2

Abre una terminal y posícónate en un directorio de trabajo vacío de tu preferencia. Crea mínimo 5 archivos con el nombre de tu preferencia. Realiza un script que le cambie el nombre a estos archivos por: Archivo1.dat,

Archivo2.dat, ... ArchivoN.dat, donde N es el número total de archivos que creaste. El script debe funcionar de manera dinámica, es decir, si creas un archivo adicional y vuelves a ejecutar tu script también debe renombrarlo.

```
#!/bin/bash
# Aaron Antonio Garcia Gonzalez

#num=$(ls "pruebas" | wc -l)
# echo $num
contador=1

for i in $( ls "pruebas"); do
    mv "./pruebas/"$i "./pruebas/archivo"$contador"."${i##*.}
    let contador+=1
done

[MacBook-Pro-de-Aaron:ejercicios aarongarcia$ ls ./pruebas
a.txt  b.txt  c.txt  d.txt  e.txt  f.txt  g.xls  h.docx  i.c    j.php  l.py  m.js  n.pptx  o.h    p.sql
[MacBook-Pro-de-Aaron:ejercicios aarongarcia$ sh p2.sh
[MacBook-Pro-de-Aaron:ejercicios aarongarcia$ ls ./pruebas
archivo1.txt  archivo12.js  archivo15.sql  archivo4.txt  archivo7.xls
archivo10.php  archivo13.pptx  archivo2.txt  archivo5.txt  archivo8.docx
archivo11.py  archivo14.h  archivo3.txt  archivo6.txt  archivo9.c
```

Ejercicio 3

Realiza un script que reciba al menos un argumento y que devuelva en pantalla el número de argumentos introducidos, sin el uso de \$#. NOTA: Valida que al menos se introduzca un argumento.

```
#!/bin/bash
# Aaron Antonio Garcia Gonzalez

echo "número de parámetros = $#"
```

si número de parámetros igual con 0

```
while [ $# -le 0 ]
do
    echo "Hay que introducir al menos un argumento, intente nuevamente"
    echo "Escribe tus parametros: "
    read script
    sh p3.sh $script
    if [ $# -le 0 ]; then
        exit 1
    fi
done
```

done

```
[MacBook-Pro-de-Aaron:ejercicios aarongarcia$ sh p3.sh
número de parámetros = 0
Hay que introducir al menos un argumento, intente nuevamente
Escribe tus parametros:
6 2 9
número de parámetros = 3
```

Ejercicio 4

Realiza un script que reciba mínimo 2 números y a lo sumo 8 números como argumentos, ordénalos de menor a mayor y muéstralos en pantalla. NOTA: Valida el número de argumentos, no debe aceptar menos de 2 o más de 8.

```
#!/bin/bash

# Aaron Antonio Garcia Gonzalez

echo "número de parámetros = $#"
```

si número de parámetros igual o menor con 1

```
function ordena {
    contador=0

    for i in "$@"
    do
        arreglo[$contador]=$i
        let contador=contador+1
    done

    for ((i=0; i <= ((${#arreglo[@]} - 2)); ++i))
    do
        for ((j=((i + 1)); j <= ((${#arreglo[@]} - 1)); ++j))
        do
            if [[ ${arreglo[i]} -gt ${arreglo[j]} ]]
            then
                # echo $i $j ${arr[i]} ${arr[j]}

                tmp=${arreglo[i]}
                arreglo[i]=${arreglo[j]}
                arreglo[j]=$tmp
            fi
        done
    done
```

```

done

for var in "${arreglo[@]}"
do
    echo "${var}"
done
}

if [ $# -lt 2 ]
then
    echo "ERROR - Hay menos de DOS argumentos"
elif [ $# -gt 8 ]
then
    echo "ERROR - Hay mas de OCHO argumentos"
else
    ordena $*
fi

```

```

[MacBook-Pro-de-Aaron:ejercicios aarongarcia$ sh p4.sh 3
número de parámetros = 1
ERROR - Hay menos de DOS argumentos
[MacBook-Pro-de-Aaron:ejercicios aarongarcia$ sh p4.sh 9 8 7 6 5 4 3 2 1
número de parámetros = 9
ERROR - Hay mas de OCHO argumentos
[MacBook-Pro-de-Aaron:ejercicios aarongarcia$ sh p4.sh 8 7 6 5 4 3 2 1
número de parámetros = 8
1
2
3
4
5
6
7
8

```

Ejercicio 5

Imagina que tienes una colección de álbumes de música de tus grupos favoritos. En un archivo tienes almacenado el número de álbumes por grupo. Un ejemplo de como se ve tu archivo es:

- ⇒ 4 Sonic youth
- ⇒ 3Te Smiths

⇒ 9 Phoenix

⇒ 8 Beatles

Realiza un script que imprima las N líneas mayores, es decir los N grupos con los que tienes más álbumes. El valor de N y el nombre del archivo donde esta contenida la información deben de pasarse como parámetros.

```
#!/bin/bash
# Aaron Antonio Garcia Gonzalez

num_lineas=$(cat $1 | wc -l)
contador=0
auxiliar=0
posicion=0
num=$2

for i in $( cut -d ' ' -f1 albumes.txt )
do
    arreglo[$contador]=$i
    let contador+=1
done

if [ $num -gt $num_lineas ]; then
    echo "Numero solicitado es mayor al total de bandas" $2 ">" $num_lineas
else
    for j in `seq 1 $num`;do
        let auxiliar=${arreglo[0]}
        let posicion=0

        for i in `seq 0 $num_lineas`;do
            if [ ${arreglo[$i]} -ge $auxiliar ]; then
                let auxiliar=${arreglo[$i]}
                let posicion=$i+1
            fi
        done
        sed -n ${posicion}p "albumes.txt"
        let arreglo[$posicion-1]=0
    done
fi
```

Conclusiones

Cuando curse introducción a los sistemas operativos en la educación media superior, realice archivos .bat o archivos batch, en el sistema operativo Windows, que fueron un tanto similar a estos, pero sin la lógica de programación y únicamente los comandos propios del sistema operativo, gracias a el bash de linux se pueden automatizar varios procesos cotidianos y no tan cotidianos, este tipo de comandos son muy útiles cuando se esta trabajando en el nivel bajo, Unix es muy poderoso, por lo que hay comandos bastante interesantes. Me gusto mucho que lo que se realizo en Shell script, tiene mucha sintaxis de lenguaje c que ya es bastante conocido en esta profesión.

```
[MacBook-Pro-de-Aaron:ejercicios aarongarcia$ cat albumes.txt
24 Celtic Thunder
45 Celtic woman
23 Kodaline
2 Savage Garden
56 Pentatonix
6 Ten Tenors
31 Hillsong Worship
12 Westlife
14 Beret
67 Boyce Avenue
56 Owl city
29 Maroon five
11 Dream sweet
4 Sonic youth
3 The Smiths
9 Phoenix
8 Beatles
[40] jump5MacBook-Pro-de-Aaron:ejercicios aarongarcia$ sh p5.sh albumes.txt 5
67 Boyce Avenue
56 Owl city
56 Pentatonix
45 Celtic woman
40 jump5
```

Referencias

Automatiza tareas Linux y ahorra tiempo con Bash. (2015, 5 julio). Recuperado 30 agosto, 2019, de <https://computerhoy.com/paso-a-paso/apps/automatiza-tareas-linux-ahorra-tiempo-bash-30751>

Capy | Linux (Consola): Imprimir una línea específica de un archivo de texto. (2011, 14 julio). Recuperado 30 agosto, 2019, de <http://ecapy.com/linux-consola-imprimir-una-linea-especifica-de-un-archivo-de-texto/index.html>

WC - Contar directorios / ficheros / Líneas de un fichero. (s.f.). Recuperado 30 agosto, 2019, de <https://www.cambiatealinux.com/wc-contar-directorios-ficheros-lineas>

Código de Linux/Unix Shell Scripting - Script que cuenta el número de líneas de una lista de archivos. (s.f.). Recuperado 30 agosto, 2019, de <https://www.lawebdelprogramador.com/codigo/Linux-Unix-Shell-Scripting/2705-Script-que-cuenta-el-numero-de-lineas-de-una-lista-de-archivos.html>

Alvaro Escarcha, A. E. (s.f.). [Solucionado] ¿Cómo contar el número total de líneas de | linux? Recuperado 30 agosto, 2019, de <https://www.enmimaquinafunciona.com/pregunta/37884/como-contar-el-numero-total-de-lineas-de-archivos-encontrados>

Emc2Net. (s.f.). Recuperado 30 agosto, 2019, de <https://e-mc2.net/es/bash-iv-estructuras-de-control-y-bucles>

bash - ¿Cómo cambiar la extensión de varios archivos utilizando un script de bash? (2018, 24 diciembre). Recuperado 30 agosto, 2019, de <https://rstopup.com/como-cambiar-la-extension-de-varios-archivos-utilizando-un-script-de-bash.html>

How to declare 2D array in bash. (2013, 10 mayo). Recuperado 30 agosto, 2019, de <https://stackoverflow.com/questions/16487258/how-to-declare-2d-array-in-bash>

How to increment a variable in bash? (2013, 3 diciembre). Recuperado 30 agosto, 2019, de <https://askubuntu.com/questions/385528/how-to-increment-a-variable-in-bash>

imprimiendo la primera palabra en cada línea de un archivo txt bash de unix - Código de registro. (s.f.). Recuperado 30 agosto, 2019, de <https://codeday.me/es/qa/20190323/362828.html>