UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
FACULTAD DE INGENIERÍA (UNIDAD MEXICALI)

Formato para prácticas de laboratorio

PROGRAMA EDUCATIVO	PLAN DE ESTUDIO	CLAVE DE UNIDAD DE APRENDIZAJE	NOMBRE DE LA UNIDAD DE APRENDIZAJE
Ingeniero en Computación	2009-2	12124	Taller de Sistema Operativo Unix

PRÁCTICA No.	LABORATORIO DE	Taller de Sistema Operativo Unix	DURACIÓN (HORAS)
14	NOMBRE DE LA PRÁCTICA	grep	2

1. INTRODUCCIÓN

Una de las consecuencias que ha tenido el uso de las computadoras es que se ha generado una gran cantidad de información y que es difícil, en ocasiones, encontrar lo que requerimos entre tantos datos. En esta práctica se utilizará el mando **`grep`** que permite buscar palabras o frases dentro de archivos o dentro de flujos.

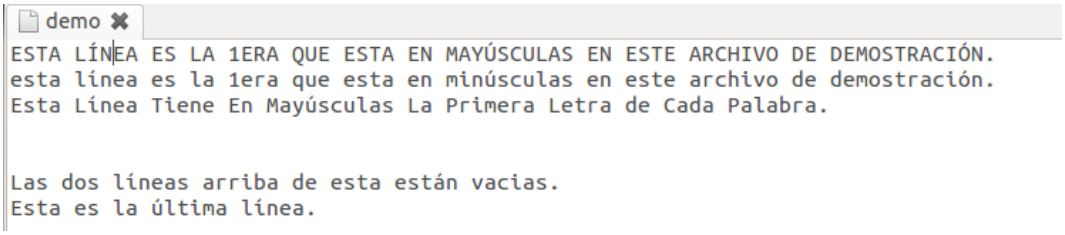
2. OBJETIVO (COMPETENCIA)

Utilizar **`grep`** para buscar patrones dentro de archivos de texto y dentro de flujos mostrando apertura hacia el autoaprendizaje.

3. FUNDAMENTO

Para el funcionamiento del mando **`grep`**, utilizaremos un archivo de nombre **demo** cuyo contenido se muestra en la siguiente figura. Para realizar los ejemplos que se muestran a continuación, se debe crear este archivo.

Formuló	Revisó	Aprobó	Autorizó
Cecilia M. Curlango Rosas	Aglay González Pacheco		David I. Rosas Almeida
Nombre y Firma del Maestro	Nombre y Firma del Responsable de Programa Educativo	Nombre y Firma del Responsable de Gestión de Calidad	Nombre y Firma del Director / Representante de la Dirección



A continuación se explicará algunos de los usos más comunes de grep.

A. Buscar un texto dentro de un archivo.

El uso más común del mando grep es para buscar un texto (también conocido como *cadena de caracteres* o simplemente *cadena*) dentro de un archivo. La sintaxis para esto se muestra a continuación:

Sintaxis

```
grep “cadena” archivo
```

Ejemplo

```
$ grep "esta" demo
esta línea es la 1era que esta en minúsculas en este archivo de demostración.
Las dos líneas arriba de esta están vacias.
```

```
$ grep esta demo
esta línea es la 1era que esta en minúsculas en este archivo de demostración.
Las dos líneas arriba de esta están vacias.
```

En el segundo ejemplo no se utilizaron las comillas (“) y generó el mismo resultado. Esto es porque sólo se busca una palabra. Si se están buscando dos o más palabras, entonces es obligatorio utilizar comillas para que **`grep`** entienda que la cadena tiene espacios en blanco. Se demuestra el uso de comillas y la falta de ellas en los siguientes ejemplos.

```
$ grep "es la" demo
esta línea es la 1era que esta en minúsculas en este archivo de demostración.
Esta es la última línea.
```

```
$ grep es la demo
grep: la: No such file or directory
demo:esta línea es la 1era que esta en minúsculas en este archivo de demostración.
demo:Las dos líneas arriba de esta están vacias.
demo:Esta es la última línea.
```

B. Buscar una cadena en varios archivos.

Con frecuencia se necesita buscar una cadena dentro de varios archivos. El mando **`grep`** permite que se le especifiquen los nombres de los archivos en los que buscará la cadena. La sintaxis para esto se muestra a continuación.

Sintaxis

```
grep “cadena” lista_de_archivos
```

Para especificar los archivos en los que se buscará la cadena, se puede escribir directamente la lista como se muestra en el primer ejemplo, o se pueden utilizar los comodines que reconoce Linux como se muestra en el segundo ejemplo. En los ejemplos, los archivo **demo**, **demo1** y **demo2** tienen el mismo contenido.

```
$ grep esta demo demo1 demo2
demo:esta línea es la 1era que esta en minúsculas en este archivo de demostración.
demo:Las dos líneas arriba de esta están vacias.
demo1:esta línea es la 1era que esta en minúsculas en este archivo de demostración.
demo1:Las dos líneas arriba de esta están vacias.
demo2:esta línea es la 1era que esta en minúsculas en este archivo de demostración.
demo2:Las dos líneas arriba de esta están vacias.
```

```
$ grep esta demo*
demo:esta línea es la 1era que esta en minúsculas en este archivo de demostración.
demo:Las dos líneas arriba de esta están vacias.
demo1:esta línea es la 1era que esta en minúsculas en este archivo de demostración.
demo1:Las dos líneas arriba de esta están vacias.
demo2:esta línea es la 1era que esta en minúsculas en este archivo de demostración.
demo2:Las dos líneas arriba de esta están vacias.
```

C. Búsquedas sin distinción entre mayúsculas y minúsculas

Se puede buscar una cadena e indicarle a **`grep`** que la encuentre sin importar si la cadena está escrita con letras mayúsculas, minúsculas o una mezcla de ambas. La sintaxis para esto se muestra a continuación y después se muestra un ejemplo de su uso.

Sintaxis

```
grep -i “cadena” archivo
```

```
$ grep -i "la" demo
ESTA LÍNEA ES LA 1ERA QUE ESTA EN MAYÚSCULAS EN ESTE ARCHIVO DE DEMOSTRACIÓN.
esta línea es la 1era que esta en minúsculas en este archivo de demostración.
Esta Línea Tiene En Mayúsculas La Primera Letra de Cada Palabra.
Las dos líneas arriba de esta están vacias.
Esta es la última línea.
```

D. Encontrar una expresión regular en un archivo

En esta sección presentaremos una introducción a expresiones regulares. Las expresiones regulares son una forma de describir una cadena utilizando símbolos especiales similares a los comodines que se emplean para describir los nombres de los archivos. En muchos manuales se emplea la expresión ***REGEX*** cuando se refiere a expresiones regulares.

Sintaxis

```
grep “REGEX” archivo
```

El siguiente ejemplo demuestra como utilizar una expresión regular para expresar que se buscan cadenas que empiecen con la palabra esta y terminen con la palabra demostración y que tengan cualquier cosa en medio. El punto y el asterisco son los símbolos con los que se forma la expresión

regular. El punto significa "*cualquier caracter*" y el asterísco significa "*cero o mas repeticiones del caracter anterior*". Analice el ejemplo hasta comprender por qué arroja el resultado que se muestra.

```
$ grep "esta.*demostración" demo
esta línea es la 1era que esta en minúsculas en este archivo de demostración.
```

La siguiente tabla muestra algunos otros símbolos que tienen significado especial en las expresiones regulares. El manual de **grep** contiene una explicación de los símbolos que se utilizan para formar expresiones regulares con una mayor complejidad. Para aquellos que están marcados en **azul** en Ubuntu se debe utilizar el mando **egrep** o **grep -E**.

Caracter	Significado
?	El caracter anterior es opcional y se buscará como máximo una vez.
*	El caracter anterior se buscará cero o más veces.
+	El caracter anterior se buscará una o más veces.
{n}	El caracter anterior se buscará exactamente *n* veces.
{n,}	El caracter anterior se buscará *n* o más veces.
{,m}	El caracter anterior se buscará como máximo *m* veces.
{n,m}	El caracter anterior se buscará al menos *n* veces, pero no más de *m* veces.

La siguiente figura muestra un ejemplo del uso de estos caracteres especiales con **egrep**. En el ejemplo, se buscan todas las líneas que tengan el caracter **í** 1 o mas veces. Esto se hace de dos formas diferentes.

```
 $ egrep "í{1,}" demo
esta línea es la 1era que esta en minúsculas en este archivo de demostración.
Esta Línea Tiene En Mayúsculas La Primera Letra de Cada Palabra.
Las dos líneas arriba de esta están vacias.
Esta es la última línea.
 $ egrep "í+" demo
esta línea es la 1era que esta en minúsculas en este archivo de demostración.
Esta Línea Tiene En Mayúsculas La Primera Letra de Cada Palabra.
Las dos líneas arriba de esta están vacias.
Esta es la última línea.
```

D. Búsqueda de palabras completas

Como se muestra en la siguiente figura, cuando busca una cadena, **grep** no distingue entre palabras completas y partes de palabras; la segunda línea que encuentra contiene la palabra *"es"*.

```
$grep es demo
esta línea es la 1era que esta en minúsculas en este archivo de demostración.
Las dos líneas arriba de esta están vacias.
Esta es la última línea.
```

Para indicar a **grep** que debe buscar la cadena como una palabra completa se utiliza la siguiente sintaxis:

Sintaxis
```
grep -w “cadena” archivo
```

Compare los resultados de la figura anterior con la figura siguiente. ¿Cuál es la diferencia entre las dos búsquedas?

```
$grep -w es demo
esta línea es la 1era que esta en minúsculas en este archivo de demostración.
Esta es la última línea.
```

E. Mostrar las líneas alrededor de la cadena

En ocasiones se requiere saber qué líneas rodean a la línea en la que se encuentra la cadena que buscamos. Para ello, **grep** tiene tres opciones que se explican a continuación. Los ejemplos siguientes utilizarán el archivo **auth.log** que se encuentra en el subdirectorio **/externos/home/clases/compartido** y su contenido es la bitácora generada por el servidor de computación cada vez que se accedió al servidor durante un periodo de tiempo.

Sintaxis
```
grep -A <N> “cadena” archivo
```

La opción **-A** muestra las **<N>** líneas que se encuentran después (AFTER) de la cadena buscada.

```
curlango@computacion:/externos/home/clases/compartido$ grep -A 2 Webmin auth.log
Jul 26 04:42:34 computacion webmin[12855]: Webmin starting
Jul 26 04:50:01 computacion CRON[18937]: pam_unix(cron:session): session opened for user root by (uid=0)
Jul 26 04:50:01 computacion CRON[18937]: pam_unix(cron:session): session closed for user root
```

Sintaxis
```
grep -B <N> “cadena” archivo
```

La opción **-B** muestra las **<N>** líneas que se encuentran antes (BEFORE) de la cadena buscada.

```
curlango@computacion:/externos/home/clases/compartido$ grep -B 2 Webmin auth.log
Jul 26 04:30:02 computacion CRON[8018]: pam_unix(cron:session): session closed for user root
Jul 26 04:42:32 computacion perl: pam_unix(webmin:auth): authentication failure; logname= uid=0 euid=0 tty= ruser= rhost=  user=root
Jul 26 04:42:34 computacion webmin[12855]: Webmin starting
```

Sintaxis
```
grep -C <N> “cadena” archivo
```

La opción **-C** muestra las **<N>** líneas que se encuentran tanto antes como después de la cadena buscada.

```
curlango@computacion:/externos/home/clases/compartido$ grep -C 2 Webmin auth.log
Jul 26 04:30:02 computacion CRON[8018]: pam_unix(cron:session): session closed for user root
Jul 26 04:42:32 computacion perl: pam_unix(webmin:auth): authentication failure; logname= uid=0 euid=0 tty= ruser= rhost=  user=root
Jul 26 04:42:34 computacion webmin[12855]: Webmin starting
Jul 26 04:50:01 computacion CRON[18937]: pam_unix(cron:session): session opened for user root by (uid=0)
Jul 26 04:50:01 computacion CRON[18937]: pam_unix(cron:session): session closed for user root
```

F. Invertir una búsqueda

En ocasiones lo que se requiere es encontrar aquellas líneas que no contienen una cadena y para ello grep también tiene una forma de realizar esta búsqueda.

Sintaxis

```
grep -v “cadena” archivo
```

En el siguiente ejemplo se muestran únicamente las líneas que *no* tienen la palabra *“esta”*.

```
$grep -v esta demo
ESTA LÍNEA ES LA 1ERA QUE ESTA EN MAYÚSCULAS EN ESTE ARCHIVO DE DEMOSTRACIÓN.
Esta Línea Tiene En Mayúsculas La Primera Letra de Cada Palabra.


Esta es la última línea.
```

G. Contar el número de líneas en que aparece una cadena

En ocasiones se requiere conocer en cuántas líneas aparece una cadena, por ejemplo, para conocer cuántos usuarios pertenecen a un grupo particular de usuarios. A continuación se muestra la sintaxis que tiene **grep** para conocer en cuántas líneas aparece una cadena.

Sintaxis

```
grep -c “cadena” archivo
```

```
$grep -c línea demo
3
```

El ejemplo anterior indica que en el archivo demo la palabra línea aparece 3 veces. Verifique este resultado encontrando estas líneas en el archivo.

H. Combinación de opciones

Como en otros mandos de Linux, se pueden combinar las opciones de **grep** para realizar búsquedas sofisticadas. En el siguiente ejemplo se muestra como visualizar la cantidad de líneas que no tienen la palabra línea en el archivo demo.

```
$grep -cv línea demo
6
```

I. Mostrar números de línea

El mando **grep** también permite agregar el número de la línea en la que encontró la cadena. Su sintaxis se muestra a continuación, seguida de un ejemplo.

Sintaxis

```
grep -n “cadena” archivo
```

```
$grep -n línea demo
2:esta línea es la 1era que esta en minúsculas en este archivo de demostración.
6:Las dos líneas arriba de esta están vacias.
7:Esta es la última línea.
```

J. Buscar en directorios

Para buscar una cadena dentro de todos los archivos que se encuentran en un directorio y sus subdirectorios, se utiliza la sintaxis que se muestra a continuación.

Sintaxis

```
grep -r “cadena” directorio
```

```
$ grep -rw "one" /etc/rc*
/etc/rc0.d/S20sendsigs:         # the list.  If we did miss one starting up, this beats waiting
/etc/rc0.d/K02vmware:            echo 'At least one instance of '"$PRODUCT_NAME"' is still running.' 1>&2
/etc/rc0.d/K02vmware:            echo 'At least one instance of '"$PRODUCT_NAME"' is still running.'
/etc/rc0.d/S60umountroot:       # will act on a bind mount of / if there is one.
/etc/rc2.d/S11vmware:            echo 'At least one instance of '"$PRODUCT_NAME"' is still running.' 1>&2
/etc/rc2.d/S11vmware:            echo 'At least one instance of '"$PRODUCT_NAME"' is still running.'
/etc/rc3.d/S11vmware:            echo 'At least one instance of '"$PRODUCT_NAME"' is still running.' 1>&2
/etc/rc3.d/S11vmware:            echo 'At least one instance of '"$PRODUCT_NAME"' is still running.'
/etc/rc5.d/S11vmware:            echo 'At least one instance of '"$PRODUCT_NAME"' is still running.' 1>&2
/etc/rc5.d/S11vmware:            echo 'At least one instance of '"$PRODUCT_NAME"' is still running.'
/etc/rc6.d/S20sendsigs:         # the list.  If we did miss one starting up, this beats waiting
/etc/rc6.d/K02vmware:            echo 'At least one instance of '"$PRODUCT_NAME"' is still running.' 1>&2
/etc/rc6.d/K02vmware:            echo 'At least one instance of '"$PRODUCT_NAME"' is still running.'
/etc/rc6.d/S60umountroot:       # will act on a bind mount of / if there is one.
```

El ejemplo anterior recorre todos los subdirectorios dentro de **/etc** cuyos nombres empiezan con **rc** y dentro de estos busca en los archivos la palabra “*one*”.

4. PROCEDIMIENTO (DESCRIPCIÓN)

A) EQUIPO NECESARIO	**MATERIAL DE APOYO**

Computadoras con Linux instalado y atado al servidor.
Los siguiente archivos: **demo** y **auth.log**.

B) DESARROLLO DE LA PRÁCTICA

Utilice el mando **grep** para solucionar los ejercicios planteados en esta sección. En algunos ejercicios utilizará **grep** en combinación con otros mandos presentados en prácticas anteriores.

Algunos de los procesos que se ejecutan en los servidores de Unix registran sus actividades en archivos de bitácora que se encuentran en el directorio `/var/log`. Una de las bitácoras es auth.log donde se registra todas la veces en que se accede al servidor para ingresar a él. Es muy importante para los administradores de los servidores revisar esta bitácora con frecuencia para detectar ataques a los servidores y tomar medidas preventivas o correctivas.

En el directorio `/externos/home/clases/compartido` se tiene una copia de esta bitácora auth.log que se generó en julio del 2012. En los siguientes ejercicios se utilizará este archivo.

1. Verifique si se intentó ingresar al servidor con los siguientes nombres de usuario: *admin*, *apache*, *mysql*, *juan*, *bd103*.

2. ¿Cuántas veces se hizo el intento con cada usuario del ejercicio anterior?

3. Muestre un listado de todos los intentos por ingresar al servidor con un nombre de usuario inexistente.

4. ¿Cuantas veces intentaron ingresar al servidor con nombres de usuario válidos?

5. ¿Con qué usuario lograron ingresar al servidor (*hackerar*)?

6. ¿En qué fecha *hackearon* el servidor?

7. ¿Desde qué direcciones de Internet se estuvo lanzando el ataque al servidor?

8. ¿Cuáles usuarios ingresaron al servidor el 28 de julio?

9. En un archivo guarde los nombres de los usuarios que se utilizaron para intentar ingresar al servidor.

10. Muestre con números de línea las 4 líneas anteriores y las 4 líneas posteriores al ingreso del usuario **prueba**.

11. ¿Cuántos procesos ejecutó el usuario **prueba**?

12. El directorio **/etc/init.d** contiene archivos para controlar el inicio y término de algunos servicios que se ejecutan en el servidor. Genere un listado de todas las veces que aparece la palabra *"start"* en cada archivo de este directorio. Incluya el número de línea en la que aparece la palabra dentro del archivo.

13. ¿Cuáles archivos contienen la palabra *"el"* en los directorios del grupo **tu100**?

14. Utilizando el redireccionamiento de error, elimine los mensajes de error que se mostraron en el ejercicio anterior.

15. Muestre los números de línea en los que aparece la palabra *"linux"* en los archivos que se encuentran en los directorios del grupo tu200 sin que aparezcan mensaje de error.

C) CÁLCULOS (SI APLICA) Y REPORTE

5. RESULTADOS Y CONCLUSIONES

6. ANEXOS

7. REFERENCIAS

1. Man page grep http://manpages.ubuntu.com/manpages/precise/man1/grep.1.html