Alberto Gómez Morales 1º DAW Sistemas Informáticos

EJERCICIOS REPASO **de Linux. Expresiones regulares**

Una expresión regular es una cadena de caracteres que es capaz de representar múltiples cadenas de caracteres.

Para crear expresiones regulares se utilizan metacaracteres, también conocidos a veces como comodines.

Cuando un comando de shell admite como argumento múltiples archivos, podemos utilizar los comodines:

* que significa "0 ó más caracteres",

? que representa "un único carácter"

Los metacaracteres vistos hasta ahora (* ?) representan a cualquier conjunto de caracteres.

¿Cómo podría listar los archivos terminados en número, o que comiencen por cierta letra? La respuesta es encerrando entre corchetes una lista o rango de caracteres.

- Mediante los corchetes [ ] podemos representar a un único carácter dentro de una lista o rango:

Ejemplos: [aeiou] es cualquier vocal (pero sólo una)

[a-z] es cualquier letra minúscula (cualquiera entre la a y la z)

[A-D] es una letra mayúscula entre la A y la D

Ejercicio 1. Listado de archivos que tengan en su nombre una 'a', mayúscula o minúscula:

```
$ ls -d *[aA]*
$ find -maxdepth 1 -name "*[aA]*"
$ find -maxdepth 1  -iname "*a*"
$ ls | grep "[aA]"
$ ls | grep "a" -i
```

Ejercicio 2. Lista los archivos con 2 dígitos numéricos detrás de prov: prov00.txt, prov01.txt,...

```
$ ls prov[0-9][0-9].txt
```

Ejercicio 3. Listado de archivos que tengan una h,i,j,k,l,m ó H,I,J,K,L,M en su nombre:

```
$ ls -d *[h-M]*
$ ls -d *[h-M]*
$ echo *[h-M]*
$ find -maxdepth 1 -name "*[h-M]*"
```

Podemos elegir los caracteres complementarios a una lista precediéndola del carácter:

! si es una expresión regular dirigida a la shell

^ si la expresión regular es interpretada por los comandos find o grep

Archivos del directorio actual que no comiencen por ninguna letra comprendida entre la 'a' y la 'f':

```
$ ls -d [!a-F]*
$ echo [!a-F]*
$ find -maxdepth 1 -name "[^A-f]*"
```

EJERCICIOS RESUELTOS **2: echo, cat y variables de entorno**

2.1. Mediante el comando echo, visualice en pantalla la siguiente frase:"esta es una frase de prueba"

Usando el comando -> echo “esta es una frase de prueba”

2.2. Visualice la misma frase pero en dos líneas usando el caracter del control \n

echo –e “esto es una frase \n de prueba”

2.3. Ejecute las siguientes líneas:

No me aparecen líneas a ejecutar en este pedido.

2.4. Recuerda lo que es una variable de entorno. ¿Cómo se puede ver el contenido de una variable de entorno?

Se puede ver con la orden “*echo $HOME”*

2.5. Mostrar el listado de todas las variables de entorno del sistema

env

2.6. Solicite el listado de su directorio usando la variable $HOME y ~

ls ~
ls $HOME

2.7. ¿Cómo modificar una variable de entorno?. (RESUELTO PARA MEJOR COMPRENSIÓN)

Para configurar la señal de ejecución en la shell, hacemos uso de PS1, un ejemplo:

PS1 = ‘[inma \d]$’

La siguiente linea modifica la variable de entorno del prompt ($PS1) para modificar el color del prompt:

PS1="\[\033[0;32m\][\$(date +%H%M)]$\[\033[0m\]"

2.8. Crear un archivo llamado CurSO en el directorio /home utilizando el comando cat y la redirección de la salida.

cat > CurSO
Estamos creando este archivo usando el comando cat y el símbolo de redireccionamiento de la salida.
Este texto será almacenado en el archivo CurSO
Y pulsar (CTRL.+D) para salir

2.9. Añadir, utlizando los símbolos de redireccionamiento algun texto al archivo:

cat >> CurSO
Este texto está siendo agregado al archivo CurSO
Y pulsar (CTRL.+D) para salir

2.10. Por medio de los símbolos de redirección copie del contenido de CurSO a CurSO_1

cat CurSO > CursoSO_1

2.11. Añada una línea a CurSO_1

cat >> CurSO_1
Estamos viendo el contenido del archivo CurSO_1
Y pulsar (CTRL.+D) para salir

2.12. Utilice el comando cat para concatenar el archivo CurSO y CurSO_1

cat CurSO CurSO_1

2.13. Repita el ejercicio anterior pero escribiendo mal el nombre del archivo CurSO_1.

cat CurSO CurSO_2
(muestra mensaje de error: no such file or directory)

2.14. Redirección del error estándar redireccionando los errores al archivo CurSO_err. Que contiene el nuevo archivo?

cat CurSO CurSO_2 > CurSO_err

cat CurSO_err

EJERCICIOS RESUELTOS 3: Comando grep y find

Para el comando **grep** tenemos los siguientes patrones:

^expr_inicial => Selecciona las líneas que comienzan con esa expresión regular
expr_final$ => Selecciona las líneas que finalizan con esa expresión

Nota: para buscar varios patrones se puede utilizar tanto grep como egrep así:

grep -e patron1 -e patron2 .... archivo_donde_buscar.

egrep "pollo | pavo" archivo_donde_buscar

Egrep es una versión más completa que el grep original, que mantiene las funcionalidades originales y además le añade unas nuevas. Esta herramienta es muy útil para realizar consultas avanzadas que grep por sí sólo no es capaz de realizar.

*Nota: **escapar un carácter** para que grep no lo interprete como especial se hace : grep 'hola/$' prueba.txt*

Esto te busca la cadena hola$ y no la cadena que acabe en hola.

3.1. Busca la cadena de texto HOLA en Archivo.txt. Cada línea de este archivo que contiene la cadena HOLA será impresa en pantalla.

$ grep 'HOLA ' Archivo.txt

3.2. Busca HOLA en todos los archivos del directorio actual.

*$ grep 'HOLA ' **

3.3. Liste los nombres de los archivos en el directorio actual que contengan la cadena de texto HOLA . Esta sentencia solo listara los nombres de los archivos, no las líneas individuales que contienen la cadena.

*$ grep -l HOLA ' **

3.4. Busque la cadena de texto "yo estudio Software Libre" en todos los archivos e n el directorio actual que sus nombres terminan con .txt. Ignore la distinción de mayúscula/minúscula de los caracteres.

*$ grep -i 'yo estudio software libre ' *.txt*

3.5. Busque la cadena de texto "final de la oración termina con ." en Archivo.txt.

$ grep 'final de la oración termina con \.' Archivo.txt

3.6. Utilice el comando grep para buscar las líneas en todos los archivos de $HOME/Prueba que contengan la expresión regular 'cadena'

grep cadena $HOME/Prueba/*

3.7. Muestre todas las líneas que NO posean la expresión regular ‘‘Luisa Ruiz’ en el archivo ‘agenda’

grep –v ´Luisa Ruiz´ agenda

3.8. Muestre solamente las líneas que comiencen con el patrón ‘Est’ en el archivo curso

grep '^Est' curSO

3.9. M*uestra cualquier línea en la que "b" sea su último carácter.*

grep 'b$' curSO

3.10. Buscar en el archivo articulosamericanos.txt los artículos que cuesten 100 doláres ($)

grep '100/$' prueba.txt

Nota: tenemos que “escapar” el símbolo $ para que no lo interprete como “fin de cadena”.

4. EJEMPLOS PRACTICOS DEL COMANDO FIND

4.1. Encuentre todos los archivos bajo /etc que sean de propiedad del usuario bin.

find /etc –user bin

4.2. Encuentre todos los archivos bajo su directorio home que hayan sido modificados en los ultimos tres días.

find $HOME –mtime 3

4.3. Encuentre todos los archivos bajo /var y bajo su directorio home que hayan sido modificados en los ultimos tres días.

find /var $HOME –mtime 3

4.4. Búsqueda de directorios (sólo directorios) que empiecen por E en el directorio actual.

*find . -type d E**

4.5. Búsqueda de directorios vacíos en el directorio actual.

find . -depth -type d -empty

4.6. Búsqueda de archivos vacíos.

find . -depth -type f -empty

4.7. Búsqueda de ficheros cuyo nombre acabe en .txt. Incluye búsqueda en subcarpetas

```
find . -type f -name "*.txt"
```

4.8. Búsqueda de ficheros cuyo nombre acabe en .txt. Incluye búsqueda solo en el directorio y subdirectorios del primer nivel.

```
find . -maxdepth 2 -type f -name "*.txt"
```

4.9. Encuentre todos los archivos bajo /usr llamados lp.

```
find /usr –name lp
```

4.10. Buscar todos los archivos cuyo nombre es respaldo.txt y contiene tanto letras mayúsculas y minúsculas en el diretorio $HOME.

```
# find $HOME -iname respaldo.txt
```

4.11. Buscar todos los directorios cuyo nombre es respaldo en / directorio.

```
#find / -type d -name respaldo
```

4.12. Buscar todos los archivos php cuyo nombre es respaldo.php en el directorio de trabajo actual.

```
# find . -type f -name respaldo.php
```

4.13. Buscar todos los archivos PHP en Directorio actual.

```
# find . -type f -name "*.php"
```

4.14. Buscar los ficheros cuyo tamaño esté entre 10M y 20M

```
$ find / -size +10240k -size -20480k
```

4.15. Buscar todos los ficheros que se nombren core y los borra interactivamente (consultando al usuario).

```
# find / -name core -exec rm -i "{}" ";"
```

4.16. Buscar todos los archivos cuyos permisos son 777 en el directorio actual.

```
# find . -type f -perm 0777
```

4.17. Buscar todos los archivos cuyos permisos NO son 777 en el directorio actual.

```
# find . -type f ! -perm 0777
```

4.18. Buscar todos los archivos del sistema con permisos de sólo de lectura para el grupo.

```
# find / -perm / g=r
```

4.19. Buscar archivos ejecutables en todos los grupos de permisos.

```
# find / -perm / a=x
```

4.20. Buscar todos los archivos con permisos 777 y utilizar el comando chmod para establecer permisos a 644 .

```
# find / -type f -perm 0777 -print -exec chmod 644 {} \;
```

4.21. Encontrar y eliminar un solo archivo

```
# find . -type f -name "archivo.txt" -exec rm -f {} \;
```

4.22. Listar todos los archivos vacíos bajo /tmp.

```
# find /tmp -type f -empty
```

4.23. Buscar todos los directorios vacíos

```
# find /tmp -type d -empty
```

4.25. Buscar todos los archivos propiedad del grupo “developer”

```
# find /home -group developer
```

4.26. Buscar los archivos modificados los últimos 50 dias

```
# find / -mtime 50
```

4.27. Buscar los archivos accedidos los últimos 50 días

```
# find / -atime 50
```

4.28. Buscar los archivos modificados entre los últimos 50-100 días

```
# find / -mtime +50 -mtime -100
```

4.29. Buscar archivos modificados en los últimos 60 minutos, ie. 1 hora

```
# find / -cmin -60
```