Nombre: ..... Fecha: / 12 /2010

Grupo: 1 ☐ 2 ☐ 3 ☐ 4 ☐

PRÁCTICA 20
MANDATOS DE TRATAMIENTO DE CADENAS. CREACIÓN DE SCRIPTS DE SHELL SENCILLOS

Como ya hemos explicado en prácticas anteriores, los ficheros de texto son de gran importancia dentro de los SO basados en GNU/Linux. Por ejemplo, la mayor parte de la información del sistema es almacenada en ficheros de texto. Eso ha hecho que diversos mandatos que permiten buscar cadenas de texto (como grep o egrep), o modificar flujos de texto (como sed o awk) sean de uso muy extendido entre los usuarios de Linux.

Para poder hacer esas búsquedas o modificaciones de texto, se utilizan las expresiones regulares. Una expresión regular es una forma de describir una (o múltiples) cadenas de texto. La cadena de texto que estamos buscando se debe corresponder con esa expresión regular. La expresión regular "alumno" se puede entender como una forma de describir la cadena "alumno" (por eso cuando hacíamos grep "alumno" /etc/passwd obteníamos todas las líneas que contienen la cadena "alumno"). La expresión regular "p*" representa "el carácter p, cero o más veces". Si ejecutas grep h* /etc/passwd obtendrás como resultado todas las líneas del fichero (ya que todas contienen cero o más veces la letra "h").

Por tanto, un buen manejo de las expresiones regulares nos va a permitir hacer búsquedas de forma más rápida o precisa. En la página http://www.panix.com/~elflord/unix/grep.html (en inglés) puedes encontrar un buen tutorial para aprender expresiones regulares. La página del manual de grep (man grep) en su sección "Regular Expressions" también contiene información de utilidad. Una versión extendida de grep es el mandato egrep (extended grep). Se caracteriza porque su lenguaje de expresiones regulares es más expresivo que el propio de "grep".

Expresiones regulares de grep y egrep (por orden decreciente de precedencia)

c	cualquier carácter no especial c concuerda a sí mismo
\c	cancela cualquier significado especial del carácter c
^	inicio de línea
$	final de línea
\b	marca inicio o final de palabra
.	cualquier carácter individual
[…]	cualquiera de los caracteres en …; los rangos tipo a-z son legales
[^…]	cualquier carácter individual que no se encuentre en …; los rangos valen
\n	lo que concordó con el n-ésimo \(…\) (grep solamente)
r*	cero o más ocurrencias de r
r+	una o más ocurrencias de r (egrep solamente)
r?	cero o una ocurrencia de r (egrep solamente)
r{n}	n ocurrencias de r (egrep solamente)
r{n,}	n o más ocurrencias de r (egrep solamente)
r{,m}	de 0 a m ocurrencias de r (egrep solamente)
r{n,m}	de n a m ocurrencias de r (egrep solamente)
r1r2	r1 seguida de r2
r1|r2	r1 o r2 (egrep solamente)
\(r\)	expresión regular marcada r (grep solamente); puede ser una expresión
(r)	regular anidada (egrep solamente); puede anidarse

En la tabla anterior mostramos algunas de las expresiones regulares que se usarán a lo largo de la práctica.

1. A partir de la información que has leído en las páginas anteriores sobre expresiones regulares y de la tabla facilitada, trata de completar los ejercicios sobre grep que puedes encontrar en http://lem.eui.upm.es/tgrep.html.

Descarga (por medio de "wget", si no posiblemente tendrás problemas con formatos y codificaciones) el fichero que puedes encontrar en http://www.unirioja.es/cu/jearansa/1011/ficheros/historia a tu escritorio y completa las siguientes búsquedas.

2. Muestra todas las líneas que contengan la cadena "España".

($grep "España" historia)

3. Muestra todas las líneas que contengan primero la cadena "España" y luego la cadena "guerra".

($grep "España.*guerra" historia)

4. Muestra las líneas que contengan primero la cadena "guerra" y luego la cadena "España".

5. Muestra todas las líneas que contengan las cadenas "Francia" o "Italia" (puedes utilizar egrep).

6. Muestra las líneas que contengan ambas cadenas "España" y "guerra" en cualquier orden (puedes usar tuberías).

7. Muestra las líneas que contengan la cadena "guerra" pero no "España" (la opción grep –v hace búsquedas inversas, es decir, todas las líneas que no contengan la expresión regular especificada).

8. Muestra las líneas que contengan palabras de cinco letras terminadas en "ía" (recuerda que "\b" representa el separador entre palabras).

9. Muestra líneas que contengan números de cuatro cifras (puedes crear intervalos o rangos de valores por medio de [0-9], o también puedes usar la clase de caracteres [:digit:], que representa cualquier dígito, es decir, equivale a [0-9]).

10. Muestra líneas que empiecen con "a" (el carácter de principio de línea en expresiones regulares es "^").

11. Muestra líneas que terminen con "s" (el carácter par fin de línea en expresiones regulares es $).

12. Muestra líneas que comiencen con la cadena "la".

13. Muestra líneas que terminen con la cadena "as".

14. Muestra líneas que empiecen y terminen con “s”.

15. Muestra líneas que empiecen con “a” y terminen con “s”.

16. Muestra líneas que contengan fechas relativas al siglo XX (usa la clase [[:digit:]]).

17. Muestra líneas que contengan alguna palabra de 15 letras (no de más, así que deberás usar “\b”; puedes usar la clase [:alpha:], que es equivalente al rango [a-zA-Z]).

18. Muestra el número de líneas que contengan la cadena “y” (utiliza la opción “-c” para contar el número de líneas, o una tubería y luego “wc -l” como hicimos en prácticas anteriores).

19. Muestra el número de líneas que contenga la palabra “y”.

20. Muestra el número de líneas que contenga la cadena “y” dentro de una palabra (de más de una letra).

Otro comando que también nos permite trabajar con ficheros de texto es sed (stream editor). El mandato sed nos va a permitir tomar un fichero de texto y modificarlo, de nuevo por medio de expresiones regulares o por alguna de sus opciones más conocidas. Para que puedas imaginar un caso práctico de uso, si quisieras modificar todas las apariciones en un fichero “index.html” de la expresión “Bienvenido a mi página web” por la expresión “Ésta es mi página”, podrías hacerlo por medio del mandato:

sed “s/Bienvenido a mi página web/Ésta es mi página/g” index.html > index_copia.html

La potencia de sed hace que su sintaxis sea menos intuitiva que la propia de grep. Las expresiones regulares van a seguir siendo válidas, pero hay que ser capaces también de expresar las acciones que llevaremos a cabo sobre el fichero de texto correspondiente (en grep siempre era mostrar, con sed puede ser mostrar, sustituir, añadir, eliminar…).

Puedes encontrar ayuda sobre el mandato sed en su manual de Linux (man sed). También serán de utilidad los artículos (en castellano) http://www.gentoo.org/doc/es/articles/l-sed1.xml, http://www.gentoo.org/doc/es/articles/l-sed2.xml. Las herramientas que vamos a usar de sed en esta práctica son imprimir fragmentos de texto, eliminar partes del mismo y reemplazar cadenas. Para ello, a partir del fichero “historia” haz las siguientes operaciones:

21. Muestra las 8 primeras líneas del fichero historia:

$sed –n “1,8p” historia

La opción –n consigue que las líneas no se muestren duplicadas. La opción “p” sirve para imprimir el texto seleccionado (de las líneas 1 a 8).

22. Crea un fichero llamado “fichero” que contenga las 20 primeras líneas del fichero “historia”. Realiza los siguientes ejercicios sobre el mismo, utilizando el mandato sed.

23. Muestra el contenido del fichero (opción p); utiliza la opción –n para que el texto no se muestre duplicado.

24. Muestra las líneas que contengan la cadena “España”:

$sed –n “/España/p” fichero

25. Borra las líneas comprendidas entre la 5 y la 10 (igual que mostrar un grupo de líneas pero con la opción “d”). Si usas la opción –n no verás ninguna de las líneas.

26. Borra todas las líneas en blanco (recuerda la expresión regular de principio y fin de línea).

27. Borra todas las líneas que contengan la cadena “en”.

28. Borra todas las líneas que contengan las cadenas “en” y “el” (trata de encadenar las dos expresiones regulares, sed admite como primer atributo varias expresiones regulares).

29. Borra la primera palabra de cada línea. Este mandato lo vamos a hacer por sustitución (s) en lugar de hacerlo por borrado (d). La sintaxis para sustituir la primera palabra de cada línea por “nada” sería:

$sed “s/^[[:alpha:]]*\b//g” fichero

(La “s” indica que hacemos una sustitución; entre los dos primeros /…/ está la expresión regular que queremos reemplazar, en este caso “/^[[:alpha:]]*\b”; entre el segundo y el tercer “//” aparece la expresión por la cual será reemplazada, en este caso por nada; la “g” final indica que el cambio debe ser global; si no indicamos nada, sólo se llevará a cabo en la primera aparición).

30. Cambia la primera aparición de la letra “a” por “@”.

31. Cambia todas las apariciones de la letra “a” por “@”.

32. Cambia todas las apariciones de la palabra España por Spain.

33. Reemplaza todas las vocales de minúsculas a mayúsculas (puedes usar tuberías, concatenar varias expresiones regulares, la opción “y/caracteres_a_reemplazar/caracteres_reemplazados/”…)

34. Elimina todas la apariciones de la letra “e”.

35. Elimina todas las vocales (mayúsculas, minúsculas, acentuadas o no).

36. Elimina todos los espacios en blanco.

Para terminar la práctica, vamos a hacer algunos pequeños ejemplos de "shell scripting". Hasta ahora hemos visto cómo usar mandatos en la shell (en el bash) y cómo los mismos se ejecutaban. Ahora vamos a ver una forma sencilla de escribir programas en la shell (llamados scripts) que se puedan ejecutar sin dificultades.

37. Ejecuta el editor de textos nano. Escribe en el mismo:

clear
echo "Estamos ejecutando un script"

Guarda el anterior texto con nombre: "script.sh".

Para poder ejecutar el fichero de texto anterior, le debemos asignar permisos de ejecución, al menos para el propietario (recuerda cómo lo hicimos en prácticas anteriores). Una vez le hayas asignado permiso de ejecución, para ejecutarlo recuerda que sólo debías hacer:

./script.sh

¿Qué ha sucedido?

38. Vamos a crear un nuevo script, de nombre vuelca_datos.sh, que cree un fichero de nombre datos (con touch, por ejemplo) y que escriba en el mismo (con echo) "Mi nombre es …", "Estamos en prácticas de SI". Cámbiale los permisos y prueba a ejecutarlo. Comprueba que ha hecho lo que esperabas.

39. Recupera el script en http://www.freeos.com/guides/lsst/ch02sec10.html; omitiendo las líneas que comienzan por "$" o por "#" cópialo en un fichero de texto (saludo.sh) y trata de ejecutarlo. Como puedes observar, en los scripts de la shell podemos utilizar variables declaradas dentro del propio script.

40. Otra opción de los scripts de shell es el paso de parámetros. Le podemos pasar a un script uno o varios parámetros a la hora de ejecutarlo (cada parámetro se denominará, dentro del script, $1, $2, $3…). Copia el script que puedes encontrar en http://www.freeos.com/guides/lsst/ch03sec02.html (omitiendo las líneas que empiezan por # o por $) en un fichero de texto mayor_que_cero.sh.

Cámbiale los permisos y ejecútalo como sigue:

$./mayor_que_cero.sh 17
$./mayor_que_cero.sh -45

¿Qué resultado has obtenido en cada uno de los casos? Como puedes observar en http://www.freeos.com/guides/lsst/ch03sec03.html, la estructura if () {…} else {…} también está admitida.

Modifica tu programa "mayor_que_cero.sh" para que muestre el mensaje "$1 number is negative" en caso de que el número sea negativo.

41. Recupera la historia de la práctica y redirígela a un fichero de nombre "mandatos_practica_20". Enlaza el fichero desde tu página web.