Búsqueda de expresiones regulares.
El *shell*. Grep y sus derivados.

Teoría de Autómatas y lenguajes formales
Federico Simmross Wattenberg (fedesim@infor.uva.es)
Universidad de Valladolid

En un entorno de trabajo UNIX, las expresiones regulares están presentes siempre que uno trata de buscar, extraer o aislar información dentro de un texto. Existen multitud de herramientas que automáticamente analizan ficheros de texto y nos proporcionan como salida una porción de ellos (la información que pretendemos aislar). Cada una de ellas tiene una especial utilidad según cada caso concreto, pero sobresale sin duda el comando `grep`, con sus variantes `egrep` y `fgrep`. Veremos que la potencia de estas herramientas puede multiplicarse combinando el uso de varias de ellas mediante las posibilidades que nos ofrece el *shell*.

1. El *shell*

Podemos decir que el *shell* es un intérprete de comandos; es la aplicación que permite al usuario comunicarse con el sistema operativo y darle órdenes. No vamos a explicar aquí todas sus funcionalidades, sino simplemente las que más directamente nos afectan ahora; sin embargo, se anima a los alumnos a explorar sus posibilidades en la documentación (`man sh`, `man bash`, `man tcsh`,...). Seguiremos la sintaxis del *Bourne shell* y sus derivados.

Algunas posibilidades interesantes:

Nombre	**Sintaxis**	**Descripción**
Redirección	`comando < fichero`	toma la entrada estándar de comando del fichero (en vez de desde el teclado)
Redirección	`comando > fichero`	dirige la salida estándar de comando al fichero (en vez de a la pantalla)
Redirección	`comando >> fichero`	idem, pero añade al final del fichero en vez de sobrescribirlo.
Redirección	`comando 2> fichero`	dirige la salida de errores estándar al fichero (en lugar de a la pantalla)
Pipe	`comando1 | comando2`	Ejecutar comando2 de forma que su entrada estándar sea la salida estándar producida por comando1
Sustitución de comando	`` `comando` `` ó `$(comando)`	Se sustituye por la salida estándar de comando.
Quoting	`'literal'`	impide que el shell interprete el significado especial de los caracteres del literal.
Quoting	`"literal"`	impide que el *shell* interprete el significado especial de los caracteres del literal, excepto de $, \ y `
-	`comando1 ; comando2`	ejecutar comando1 y a continuación, comando2
-	`comando &`	ejecutar comando en *background*

Es posible guardar en un fichero una secuencia de comandos del *shell* para su posterior ejecución. De esta forma podemos crear scripts a los que se puede llamar como a cualquier otro comando del sistema operativo. Para ello, debemos dar permisos de ejecución al fichero (con `chmod`), e insertar el siguiente texto en su primera línea:

```
#!/ruta/completa/al/intérprete/de/comandos
```

Por ejemplo:

```
#!/bin/sh
#
#Script de prueba
#Se ejecuta con /bin/sh

echo "Hola, Mundo"
```

Éste es un script interpretable por el shell `/bin/sh`. Si lo hubiéramos escrito en un lenguaje para otro intérprete, bastaría con cambiar la primera línea:

```
#!/bin/perl
#
#Script de prueba
#Se ejecuta con /bin/perl

print "Hola, Mundo\n";
```

2. La herramienta `grep`

grep procesa las líneas de un fichero indicado (o de la entrada estándar si no se indica ninguno), y pone en la salida estándar aquellas que se ajusten a un patrón dado, es decir, a una expresión regular.

2.1. Variantes

En todos los sistemas UNIX existen al menos estas tres versiones de grep:

- **grep**: Busca expresiones regulares básicas.
- **egrep (extended grep)**: Busca expresiones regulares completas. Es más potente pero utiliza un algoritmo distinto, que consume más recursos.
- **fgrep (fast grep)**: Busca cadenas de texto fijas. Es la variante más rápida pero ignora el significado especial de los metacaracteres.

Estas variantes no siempre existen:

- **rgrep**: Busca en el directorio especificado y en todos sus subdirectorios.
- **zgrep**: busca en ficheros comprimidos.

2.2. Sintaxis básica

Véase la página *man* correspondiente para más detalles.

```
[efrz]grep [opciones] <patrón> [fichero...]
```

Opciones:

`-c`: muestra sólo el número de líneas que coinciden con el patrón.
`-n`: muestra el número línea de cada línea que coincide con el patrón.
`-i`: ignora la diferencia entre mayúsculas y minúsculas.
`-v`: muestra las líneas que NO coinciden con el patrón, en vez de las que coinciden.
`-w`: el patrón debe coincidir con una palabra completa del fichero para que se muestre.
`-l`: muestra sólo el nombre de los ficheros en los que se ha encontrado alguna coincidencia.

2.3. Ejemplos

- Mostrar los usuarios cuyo *login* comienza por a:
  ```
  grep '^a' /etc/passwd
  ```

- Mostrar el número de usuarios cuyo *login* empieza por a:
  ```
  grep –c '^a' /etc/passwd
  ```

- Mostrar los ficheros que contienen la cadena 'printf' en los ficheros de /usr/include y sus subdirectorios, cuyo nombre empieza por 's':
  ```
  rgrep –l printf /usr/include/s*
  ```
  ó
  ```
  grep –l printf `find /usr/include –name 's*'`
  ```

3. Ejercicios

Utilícese el fichero de texto 'quijote.txt' para realizar estos ejercicios. Puede descargarse de la página web de la asignatura: http://duero.lab.fi.uva.es/~valecar/talf

1- Mostrar las líneas que contienen la letra 'f' (mayúscula o minúscula).
2- Quitar las líneas vacías.
3- Mostrar las líneas que únicamente contienen uno o más puntos (.)
4- Mostrar las líneas que contengan exactamente 3 letras 'c'.
5- Mostrar las líneas que contengan palabras que empiecen por 'qui' o por la letra 'f' (en mayúsculas o minúsculas).
6- Mostrar las líneas que contengan un número par de letras 'a' (al menos 2).