Teoría de Autómatas y Lenguajes Formales, IS17
Ingeniería Técnica en Informática de Sistemas

Práctica 1:
Introducción al Analizador Léxico FLEX

Enunciado:

El objetivo de esta práctica consiste en aprender a cómo utilizar el analizador léxico FLEX. Para este propósito se empezará describiendo brevemente cómo funciona un programa FLEX, cómo está estructurado y cómo compilarlo. El manual completo de FLEX se encuentra en la página WEB de la asignatura en el aula virtual: http://aulavirtual.uji.es bajo la etiqueta **Prácticas**.

Nota: *Al final de cada sesión de prácticas hay que entregar la solución a cada uno de los ejercicios marcados con (**) (No olvidar escribir nombre, apellido en cada hoja entregada y grupo de prácticas en el que se está apuntado).*

Breve introducción al analizador léxico FLEX.

¿Qué es un analizador léxico? Un analizador léxico es un programa que busca en, generalmente, grandes ficheros de texto, aquellas cadenas que se corresponden con unos patrones especificados en el analizador.

¿Qué es FLEX? FLEX es un analizador léxico bajo licencia GPL. Cada vez que se encuentre uno de los patrones especificados en FLEX se puede ejecutar un conjunto de acciones asociadas. FLEX es el analizador de dominio público compatible con el analizador léxico más frecuentemente utilizado: LEX (bajo sistema UNIX). FLEX (y LEX) genera, dada una especificación correcta de patrones y acciones, un programa en lenguaje C que puede ser compilado para obtener un programa ejecutable.

¿Cómo se especifican los patrones? Cada patrón es una ***expresión regular***. Algunos de los patrones que se pueden utilizar son (la lista completa está en el manual de FLEX):

Patrón	Significado	Patrón	Significado
`a`	Carácter 'a'	`r*`	Cero o más r
`.`	Cualquier carácter excepto \n	`r+`	Una o más r
`[abc]`	El carácter 'a', el 'b' o el 'c'	`r?`	Cero o una r
`abc`	La cadena 'abc'	`r{1,3}`	Entre 1 y 3 r

Patrón	Significado	Patrón	Significado
`[cv-yA]`	El carácter ‘c’, el ‘A’ o un carácter entre el ‘v’ y el ‘y’	`{nombre}`	La expansión de la definición de “nombre”
`[^aB-F]`	Cualquier carácter excepto ‘a’ o un carácter entre ‘B’ y ‘F’	`\x`	Lo que sea x, por ejemplo \n o \t o \a o \\ o \b o \c …
`" [a-wp"`	La cadena " [a-wp"	`r|s`	Una r o una s
`^r`	Una r al comienzo de línea	`r/s`	Una r sólo si le sigue una s

Más ejemplos de patrones en FLEX ¿Qué significan? ¿Alguna subcadena de la cadena indicada pertenece al patrón? Si la respuesta es sí ¿Cuáles son?

Patrón	Significado	Cadena	¿Cuáles?
`[abc]+`		Hfgccbbcbada	
`(abc)+`		hfgccbbcbaaab	
`abc[abc]*abc`		fabcbbccacabc	
`.+`		abc\naaa\n\n\n	
`0|[1-9][0-9]*`		00012ab000a90	
`0{1,3}1{2}0{,5}`		011 0 0100000	
`[ ]+`		asc a d d ff\n	
`[Bb][Ee][Gg][Ii][Nn]`		BeGin BbegiN	
`[^ \t\n.]+[.][^ \t\n.]+`		Hola.que tal.	

Ejercicio: Buscar e.r. en FLEX equivalentes a cada una de las e.r. de la tabla anterior.

Variables importantes. Hay dos variables durante el proceso de análisis léxico cuyo contenido puede resultar muy interesante Una es la variable `yytext` que almacena la cadena que ha coincidido con un patrón. La otra es la variable `yyleng` que indica la longitud de la cadena a la que apunta `yytext`. El modo de funcionamiento por defecto de FLEX consiste en intentar reconocer siempre la cadena de mayor longitud que coincide con alguno de los patrones especificados.

¿Cuál es la estructura de un programa en FLEX? Consta de tres secciones.

En la **primera sección** (opcional) se especifican definiciones básicas que serán reutilizadas en otras secciones.

La **segunda sección** está compuesta de parejas: *patrón acción*. En la que *patrón* es una expresión regular y *acción* es un conjunto se sentencias en C de la forma *{sentencia1; sentencia2; …; sentenciaN;}*. Estas sentencias especifican las acciones a realizar cuando se localiza el *patrón* asociado. Cada sección se separa de la siguiente mediante el empleo de una línea que contiene únicamente "%%".

En la **tercera sección** (opcional) se especifican funciones auxiliares que, entre otras cosas, pueden ser utilizadas en la parte acción de la segunda sección.

Ejemplos de programas en FLEX:

Ejemplo 1:

```
DIGITO [0-9]
%%
{DIGITO}{DIGITO}*     { printf(" \n ¡Un entero! %d",  atoi(yytext));}
[a-zA-Z_][a-zA-Z0-9_]* { printf(" \n ¡Una variable! %s", yytext);
                         printf(" que tiene %d caracteres", yyleng);
                         }
.                            ;
%%
```

Ejercicio: ¿Qué hace este programa?

Ejemplo 2:

```
%{
   int num_lineas = 0, num_caracteres = 0;
%}
%%
\n          {++num_lineas; ++num_caracteres;}
.           {++num_caracteres;}
%%
int main() {
     yylex();
     printf("\n # de lineas = %d", num_lineas);
     printf("\n # de caracteres = %d\n", num_caracteres) ;
}
```

Ejercicio: ¿Qué hace este programa?

¿Cómo se compila un programa FLEX? Supóngase que se tiene el programa "`ejemplo.fl`" que contiene una especificación correcta de un fichero FLEX. La primera acción a realizar es transformarlo en un programa en C, para ello se utiliza el siguiente comando: *`flex ejemplo.fl`* Como resultado se obtendrá un fichero denominado "`lex.yy.c`" que, al ser un programa en C, puede ser compilado mediante el siguiente comando: *`gcc lex.yy.c -lfl`*. Como resultado se obtiene un fichero ejecutable "`a.out`"

¿Cómo se especifica la entrada al analizador? Por defecto se utiliza la entrada y la salida estándar, aunque se pueden redireccionar. Por ejemplo,

```
$ a.out  <datos  >res
```

almacena en el fichero "`res`" el resultado de ejecutar el analizador `a.out` (programa flex ya compilado con gcc) con el texto almacenado en el fichero "`datos`"

Ejercicios:

1. Diseñar patrones FLEX que reconozcan

    a) Nombres de variables
    b) Números enteros con y sin signo
    c) Números reales en coma flotante con y sin signo.
    d) Números reales en notación exponencial con signo opcional y con punto también opcional
    e) Operadores matemáticos: =,+, -, *, /, DIV, MOD
    f) Paréntesis: (,)
    g) Comentarios: # esto es un comentario que acaba en una línea

2. Introducir los programas FLEX indicados en esta memoria de prácticas, compilarlos y ejecutarlos.

3. Buscar en el manual de FLEX (aulavirtual.uji.es) cómo se pueden incluir comentarios en un programa FLEX.

4. Diseñar un programa que analice un texto de entrada y sustituya dos o más blancos seguidos por un único blanco y dos o más tabuladores por un único tabulador.

5. Diseñar un programa FLEX que borre los comentarios que aparezcan en un fichero de texto (se suponen comentarios de una sóla línea que empiezan por el símbolo #, como el caso del patrón g del ejercicio 1). (**)

6. Diseñar un programa FLEX que imprima un texto tal y como está en el fichero de entrada, pero que cada vez que detecte un ";" imprima el texto que va a continuación en otra línea nueva. (**)

7. Diseñar un programa FLEX que indique cuántas veces ha detectado un número entero en un fichero de texto.

8. Diseñar un programa FLEX que, cada vez que detecte una cadena que pertenezca a uno de los patrones del ejercicio 1, imprima un mensaje que indique el tipo de cadena que ha localizado. Por ejemplo, la cadena "+542.12" pertenece al lenguaje denotado por el patrón c) y, por lo tanto, se debería de imprimir 'REAL_COMA_FLOTANTE', con la cadena "-2.4E-5" se debería de imprimir 'REAL_EXPONENCIAL'. (**)

Atención, definición importante: A las etiquetas que denotan que se ha encontrado una cadena que se corresponde con la expresión regular se les denomina "token", por lo tanto, REAL_COMA_FLOTANTE y REAL_EXPONENCIAL son tokens. Cada token tiene un valor asociado por defecto ¿Cuál puede ser?

9. Diseñar un programa FLEX que identifique e imprima los números de teléfono móvil que hay en un texto de entrada.

10. Diseñar un programa FLEX que identifique e imprima los números de teléfono móvil y fijo que hay en un texto de entrada. (**)