Introducción a FLEX
Una herramienta para el análisis léxico

Construcción de compiladores

- Los compiladores diseñados a principios de 1950 utilizaban técnicas exclusivamente creadas para analizar el código fuente del lenguaje
- Durante los 60's esta área tuvo mucha atención por parte de la comunidad académica
- Finalmente en los 70's el anális léxico y sintáctico eran problemas muy bien entendidos
- Uno de los avances más importantes fue dividir el trabajo de análisis en dos partes: análisis léxico y análisis sintáctico

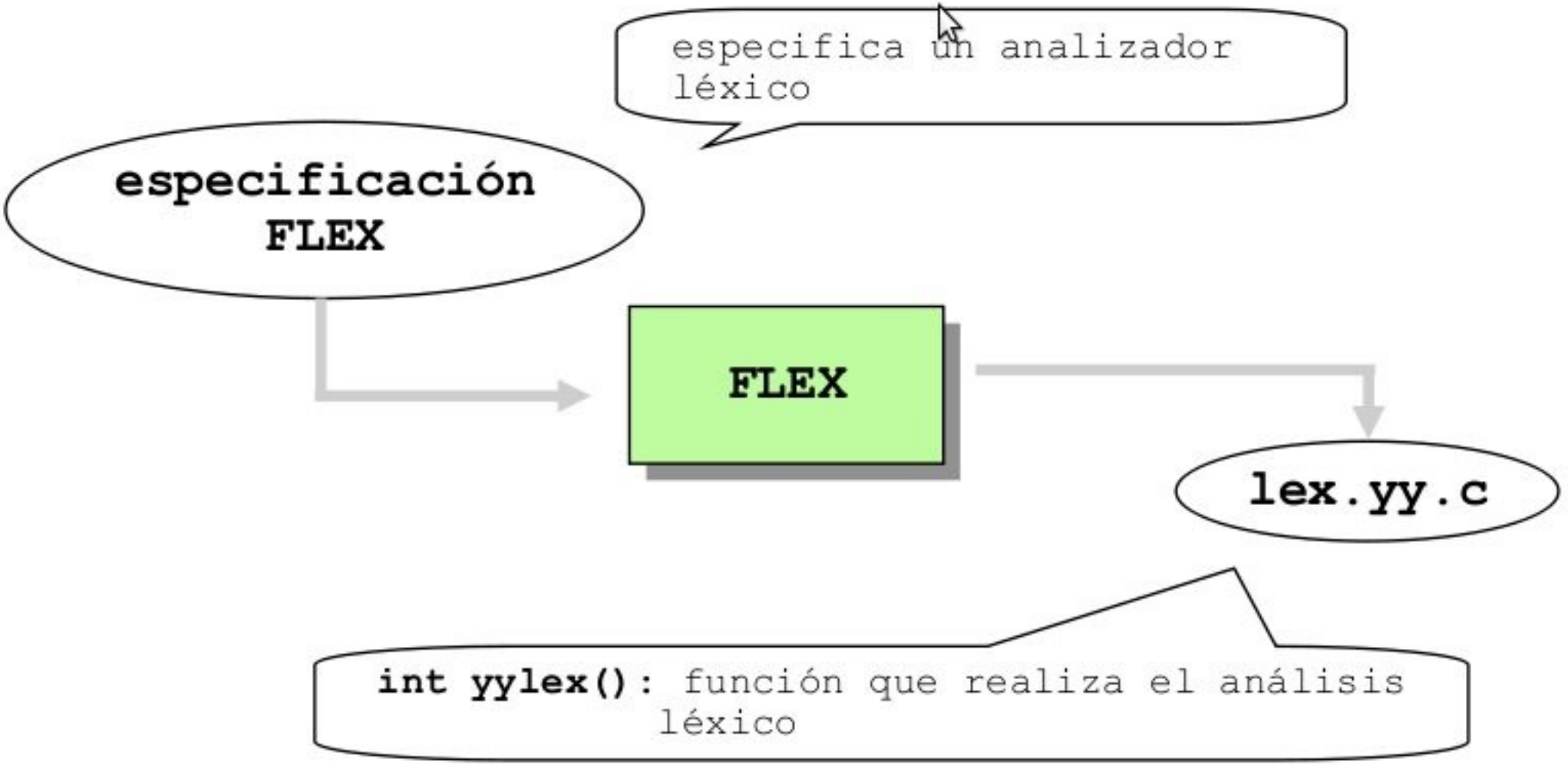
Lex

- En 1975 Mike Lesk y Erick Schmidt diseñaron Lex
- Lex es un generador de analizadores léxicos
- Se utiliza como programa independiente o como complemento del generador de analizadores sintácticos YACC
- Lex era una herramienta lenta y con varios errores, sin embargo se hizo popular

Flex

- En 1987 Vern Paxson tomo una versión de lex y la reescribio en C llamando Flex (Fast Lexical Analizer Generator)
- Debido a su rapidez y disponibilidad terminó suplantando completamente a Lex

Funcionamiento de Flex



Expresiones regulares

- El mecanismo más importante para describir patrones es la expresión regular
- Las expresiones regulares usan un metalenguaje basado en POSIX-extendido
- El metalenguaje utiliza caracteres de texto estandar, algunos de los cuales se representan a si mismo y otros representan patrones

Expresión regular	Descripción
x	Empata con la cadena “x”
.	Cualquier carácter, excepto \n
[xyz]	Una clase; en este caso una de las letras x, y, z
[a-zA-Z]	Una clase con un rango; empata con cualquier letra
[^A-Z]	Una clase complementada esto es, todos los caracteres que no están en la clase. En este caso cualquier carácter, excepto las letras mayúsculas.
r*	Cero o más veces una r
r+	Una o más veces una r
r?	Cero o una r
r{2,5}	Entre 2 y 5 apariciones de r
r{2,}	2 o más r. r{4} Exactamente 4 r
{nombre_macro}	La expansión de nombre_macro mediante su definición_regular
“[xyz]\”foo”	Exactamente la cadena [xyz]”foo
(r)	Los paréntesis son utilizados para cambiar la precedencia
rs	Concatenación.
r|s	Empata con r o s.
^r	Empata con r, al comienzo de una linea.
r$	Empata con r, al final de una linea.

Secciones de un programa en Flex

- El archivo de entrada en Flex consiste en tres secciones separadas por una línea conteniendo solo '%%'

```
%{
  Declaraciones en C
%}
definiciones
%%
reglas
%%
código de usuario
```

Sección de definiciones

- Contiene declaraciones de definiciones de nombre utilizadas para simplificar las especificaciones del analizador, así como declaraciones de condiciones iniciales
- Las definiciones de nombre tienen la siguiente sintaxis:

```
nombre definición
```

- Donde
  - `nombre` es una palabra que inicia con una letra o guión bajo (‘_’) seguido de cero o mas letras, dígitos, ‘_’,‘-’
  - *definición* se toma a partir del primer carácter distinto de espacio en blanco después de “nombre” y continua hasta el fin de línea
- Ejemplo:

```
DIGITO   [0-9]
```

Sección de definiciones

- Se puede hacer referencia a una definición anterior mediante la sintaxis {nombre}
- Ejemplo:

```
NUM {DIGITO}+"."{DIGITO}*
```

Sección de reglas

- Esta sección contiene una serie de reglas de la forma:

```
patrón acción
```

- Donde patrón debe estar sin sangrar y la acción debe estar en la misma línea
- Ejemplo:

{NUM} {return(NUMERO);}

Sección código

- Esta sección de copia literalmente al archivo de salida de Flex
- Se utiliza para rutinas de complemento que llaman al analizador o son llamadas por este
- Esta sección es opcional

- Ejemplo de uso de LEX:

La función yylex

- El *archivo.l* donde se especifica la tarea que realizará el analizador sintáctico incluye una función entera llamada *yylex()*
- La función yylex() analiza las entradas, buscando la secuencia más larga que empata con alguna de las expresiones regulares y ejecuta la correspondiente acción
- Si no encuentra ningún emparejamiento ejecuta la regla por defecto *printf("%s", yytext)*

La función yytext

- Si ecuentra dos expresiones regulares con las que la cadena más larga casa, elige la que aparece primero en el programa lex
- Una vez que se encuentra alguna coincidencia con una epxresión regular, la cadena queda disponible a través del apuntador global *yytext* y su longitud en el apuntador *yyleng*