PROCESADORES DE LENGUAJES

Análisis léxico con Lex

Prof. Dr. Nicolás Luis Fernández García

Departamento de Informática y Análisis Numérico
Escuela Politécnica Superior
Universidad de Córdoba

Contenido del tema

Lex

Características de la generación automática

- Los componentes léxicos se denotan mediante expresiones regulares.
- El generador léxico crea automáticamente el código a partir de las expresiones regulares.
- Generadores léxicos: lex, flex, pclex, jlex, ...

Lex

COMPONENTES LÉXICOS

Lex

COMPONENTES LÉXICOS

EXPRESIONES REGULARES

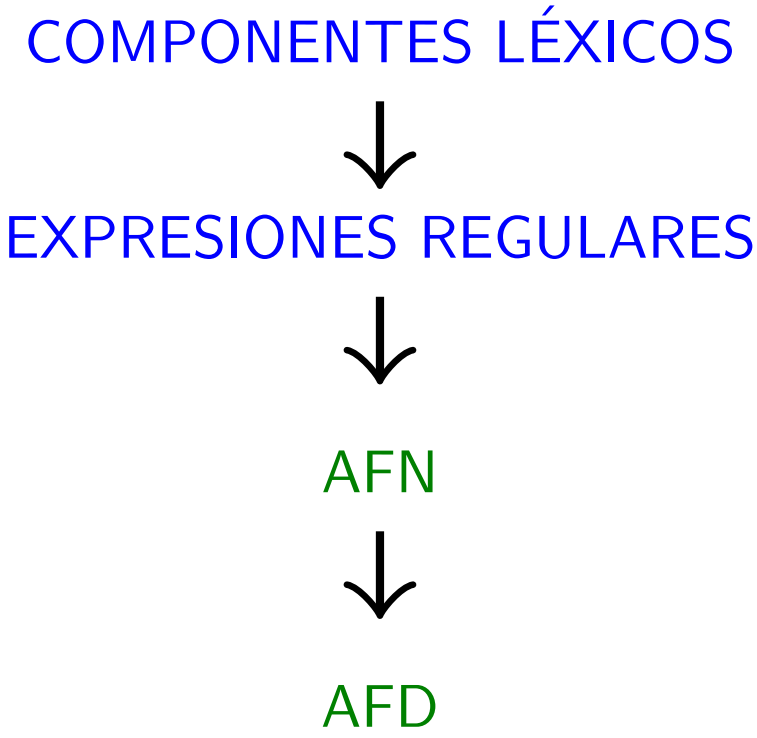
Lex

COMPONENTES LÉXICOS

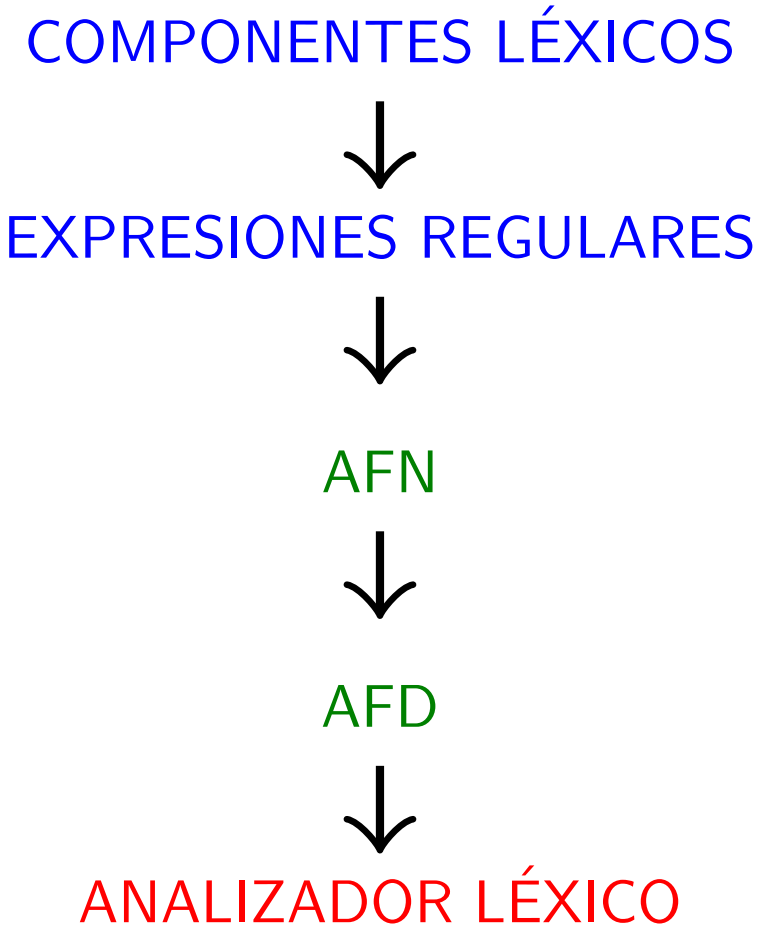
EXPRESIONES REGULARES

AFN

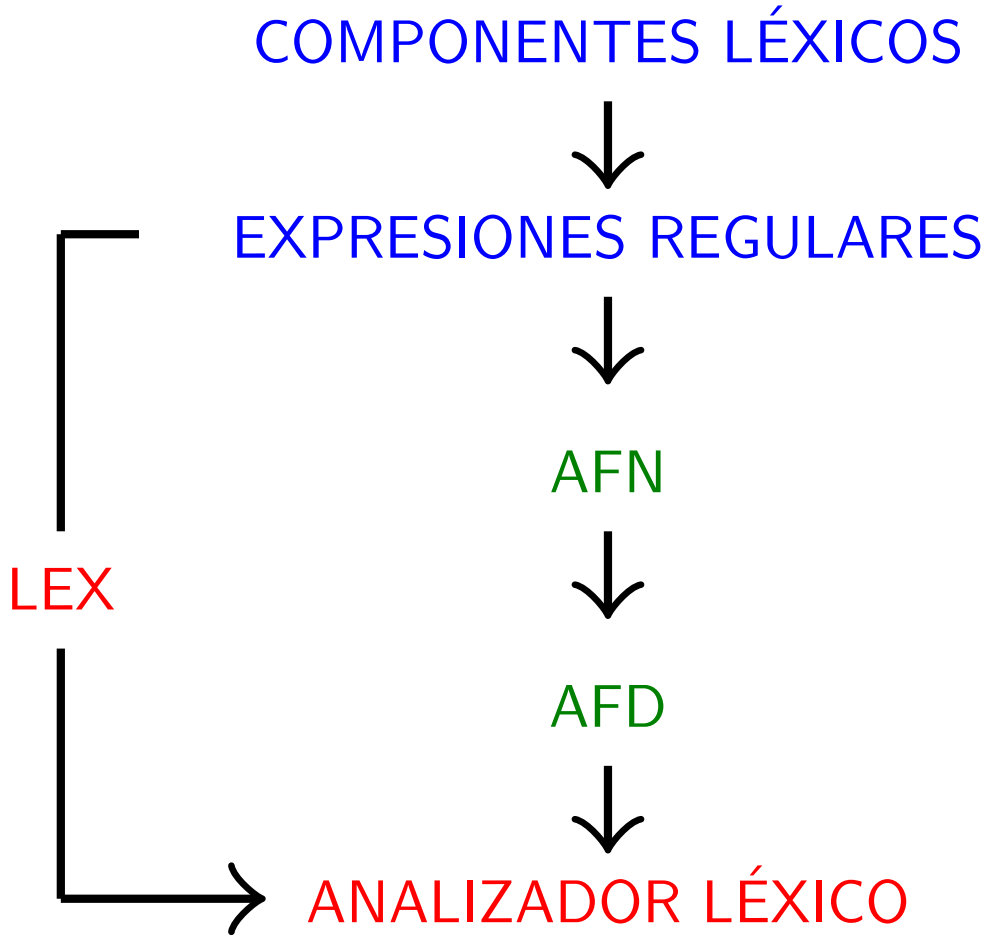
Lex



Lex


COMPONENTES LÉXICOS
EXPRESIONES REGULARES
AFN
AFD
ANALIZADOR LÉXICO

Lex


COMPONENTES LÉXICOS
EXPRESIONES REGULARES
AFN
AFD
LEX
ANALIZADOR LÉXICO

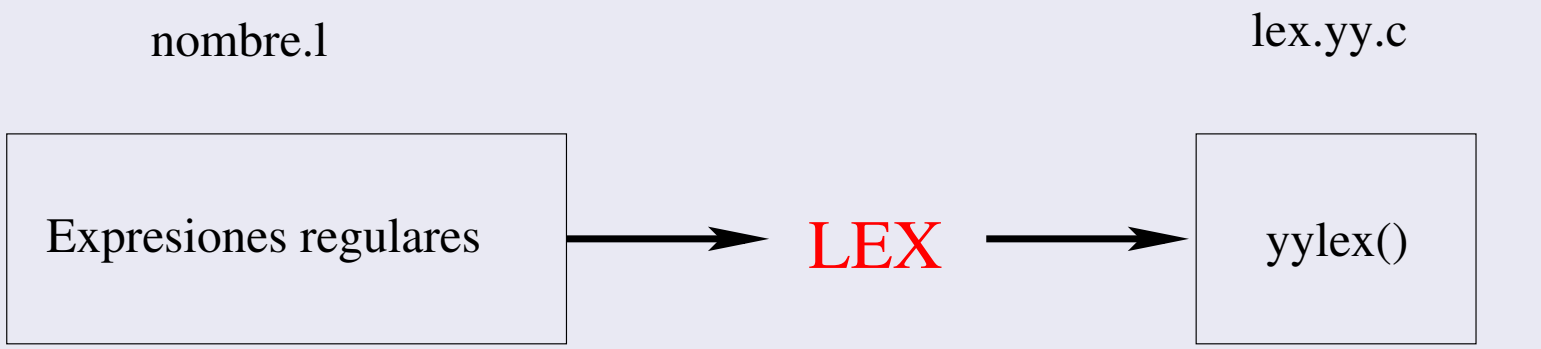
Lex

LEX

- Creado por M. E. Lesk y E. Schmidt (Bell Laboratories).
- Genera analizadores léxicos para C, Fortran, Raftor.
- Hay versiones para Unix, Linux, DOS, etc.

Lex

Funcionamiento de LEX



Lex

Funcionamiento de LEX

- Unix
  - > lex nombre.l
  - > cc -g lex.yy.c -ll -o nombre.exe
- Linux
  - > flex nombre.l
  - > gcc -g lex.yy.c -lfl -o nombre.exe

Lex

Ejecución

- $>$ ./nombre.exe
- Redirigiendo la entrada y la salida
  - $>$ ./nombre.exe $<$ fichero_entrada
  - $>$ ./nombre.exe $<$ fichero_entrada $>$ fichero_salida
- Usando argumentos desde la línea de comandos
  - $>$ ./nombre.exe fichero_entrada
  - $>$ ./nombre.exe fichero_entrada fichero_salida

Lex

Estructura del fichero de LEX

declaraciones `(opcional)`

%%

reglas de traducción de las expresiones regulares

%%

funciones auxiliares `(opcional)`

Lex

LEX: expresiones regulares

- Símbolos especiales:
  - |: disyunción
  - ( ): agrupación de expresiones regulares
  - *: repetición de un patrón cero o más veces.
  - +: repetición de un patrón una o más veces.
  - ?: el patrón puede aparecer cero o una vez.
  - " ": delimitadores de cadenas
  - .: cualquier carácter distinto del salto de línea (\n).
  - \n: salto de línea
  - $: carácter de final de línea
  - [ ]: delimitadores de clases de caracteres
  - ^ : inicio de línea y complementario de una clase.

Lex

LEX: expresiones regulares

Nota

Si se antepone la barra \ delante de un símbolo especial entonces sólo se representa a sí mismo:

$\backslash . \rightarrow$ *sólo representa el punto.*

Lex

Ejemplo (LEX: expresiones regulares 1 / 3)

Expresión regular	Significado
$a\|b$	a, b
$[ab]$	a, b
ab	ab
$ab+$	ab, abb, $abbb$, $\cdots$
$(ab)+$	ab, $abab$, $ababab$, $\cdots$
$ab*$	a, ab, abb, $\cdots$
$(ab)*$	ϵ, ab, $abab$, $\cdots$
$ab\{1,3\}$	ab, abb, $abbb$

Lex

Ejemplo (LEX: expresiones regulares 2 / 3)

Expresión regular	Significado
[a-z]	$a, b, c, \cdots, z$
[a \-z]	$a, -, z$
[-az]	$-, a, z$
[az-]	$a, z, -$
[a-zA-Z]	$a, b, \cdots, z, A, B, \cdots, Z$
[a-zA-Z0-9]	$a, b, \cdots, z, A, B, \cdots, Z, 0, 1, \cdots, 9$
*[a-zA-Z0-9]**	*cero o más veces* $a, b, c, \cdots, 9$
[a-zA-Z0-9]+	*una o más veces* $a, b, c, \cdots, 9$

Lex

Ejemplo (LEX: expresiones regulares 3 / 3)

Expresión regular	Significado
[\t \n]	*espacio en blanco, tabulador y salto de línea*
[^ ab]	*cualquier carácter distinto de a o b*
[a ^ b]	*a, acento circunflejo, b*
a/b	*a sólo si va seguido de b*
a$	*a si va seguido del carácter \n*
a/ \n	*a si va seguido del carácter \n*
^ abc	*abc si está escrito al principio de la línea*
[\40-\176]	*caracteres ASCII imprimibles desde octal 40 (espacio) hasta octal 176 (tilde ~)*

Lex

LEX: Zona de declaraciones (opcional)

- Código extendido de lenguaje C delimitado por **%**{ y }**%**
  - + Ficheros de cabecera.
  - + Macros.
  - + Prototipos de funciones.
  - + Variables globales
  - + Etc.
- Directivas de lex: %a, %n %o, %p, ...
- Declaración de definiciones regulares.

Lex

LEX: Zona de declaraciones (opcional)

- Directivas de lex (tablas internas):
  - %x ESTADO: permite activar reglas condicionales (véase el ejemplo del comentario).
  - %a número: cambia el número de transiciones empaquetadas.
  - %n número: cambia el número de transiciones.
  - %e número: cambia el número de nodos.
  - %p número: cambia el número de posiciones.
  - Etc

Lex

Ejemplo (Definiciones regulares 1/2)

numero	`[0-9]`
letra	`[a-zA-Z]`
identificador	`{letra}({letra}|{numero})*`

Lex

Ejemplo (Definiciones regulares 2/2)

La definición regular **identificador** *definida como*

$$\{letra\}(\{letra\}|\{numero\})*$$

es transformada en

$$[a-zA-Z]([a-zA-Z]|[0-9])*$$

Lex

LEX: Zona de reglas de traducción

expresión regular	sentencia de lenguaje C
expresión regular	{ sentencias de lenguaje C }

Lex

Ejemplo (Lex: zona de reglas)

```
%%
[ \t]          { ; } /* saltar los espacios y los tabuladores */
{numero}+\.?|{numero}*\.{numero}+   {
                                     sscanf(yytext,"%lf",&yylval.val);
                                     return NUMBER;
                                    }

\n      {lineno++; return FIN;}
.       {return yytext[0];} /* Devuelve cualquier otro carácter */
```

Lex

LEX: resolución de ambigüedades

- Si una secuencia de caracteres se puede emparejar con **varias expresiones regulares**,
  1. tiene preferencia la expresión regular que denote la cadena de caracteres de **mayor longitud**.
  2. si la longitud de la cadena es igual, tiene preferencia la que aparezca en **primer lugar**.

Lex

Ejemplo (Lex: resolución de la ambigüedad de **if**)

```
%%
if                  { return IF; }
{identificador}     { Symbol *s;
                      if ((s=lookup(yytext)) == 0)
                          s = install (yytext, INDEFINIDA, 0.0);
                      yylval.sym = s;
                      return s->tipo == INDEFINIDA ? VAR : s->tipo;
                    }
```

Lex

LEX: comandos especiales

- ECHO: imprime por pantalla el texto reconocido.
- BEGIN: cambia a un estado definido por el programador (véase el ejemplo del *comentario*).
- REJECT: rechaza el texto reconocido para que pueda ser procesado por otra regla (véase el ejemplo de *pink*).

Lex

Ejemplo (ECHO)

- *Imprime por pantalla todos los caracteres*

```
%%
.|\n    ECHO;
%%
```

Equivalencia

```
%%
.|\n    printf("%s",yytext");
%%
```

Lex

LEX: Zona de funciones auxiliares (opcional)

- Código de funciones auxiliares utilizadas por las reglas de traducción
- También se pueden incluir
  - Ficheros de cabecera.
  - Macros.
  - Prototipos de funciones.
  - Declaración de variables globales
  - Etc.

Lex

LEX: variables globales predefinidas

- yytext: cadena que contiene el texto reconocido (tipo: char *)
- yyleng: longitud de yytext (tipo: int)
- yyin:
  - Puntero al fichero de entrada
  - Tipo: FILE *
  - Valor por defecto: **stdin**, el teclado
- yyout:
  - Puntero al fichero de salida
  - Tipo: FILE *
  - Valor por defecto: **stdout**, la pantalla

Lex

LEX: funciones predefinidas

- yylex(): contiene el analizador léxico generado por flex o lex
- yymore(): indica a lex que añada el siguiente componente léxico al componente léxico actual (véase el ejemplo *hiper*).
- yywrap(): se ejecuta cuando el analizar léxico encuentra el fin de fichero:
  - Si devuelve 0, el analizador léxico continúa explorando.
  - Si devuelve 1 (valor por defecto), el analizador léxico devuelve un componente léxico nulo para indicar el fin del fichero.
  - Esta función puede ser redefinida por el programador.
- yyless(n): retiene los primeros *n* caracteres de yytext y devuelve el resto al dispositivo de lectura.

Lex

Ejemplo (Lex: zona de declaraciones 1/3)

```
%{
#include "macros.h"
#include "hoc3.h"
#include "y.tab.h"
extern char *progname;
extern int lineno;
%}

/* definiciones regulares */
numero          [0-9]
letra           [a-zA-Z]
identificador   {letra}({letra}|{numero})*
```

Lex

Ejemplo (Lex: zona de reglas 2/3)

```
%%
[ \t]           { ; } /* saltar los espacios y los tabuladores */
{numero}+\.?|{numero}*\.{numero}+   {
                                     sscanf(yytext,"%lf",&yylval.val);
                                     return NUMBER;
                                    }
{identificador}    { Symbol *s;
                     if ((s=lookup(yytext)) == 0)
                         s = install (yytext, INDEFINIDA, 0.0);
                     yylval.sym = s;
                     return s->tipo == INDEFINIDA ? VAR : s->tipo;
                    }
;          {return FIN ;}
\n         {lineno++; return FIN;}
.          {return yytext[0];} /* Devuelve cualquier otro carácter */
```

Lex

Ejemplo (Lex: zona de funciones auxiliares 3/3)

```
/***** Zona de funciones auxiliares *****/
extern FILE *yyin, *yyout;
main(int cantidad, char *palabras[])
 {   switch(cantidad)
    {
     case 2: yyin=fopen(palabras[1],``r'');
        break;
     case 3: yyin=fopen(palabras[1],``r'');
        yyout=fopen(palabras[2],``w'');
    }
yylex();
}
```

PROCESADORES DE LENGUAJES

Análisis léxico con Lex

Prof. Dr. Nicolás Luis Fernández García

Departamento de Informática y Análisis Numérico
Escuela Politécnica Superior
Universidad de Córdoba