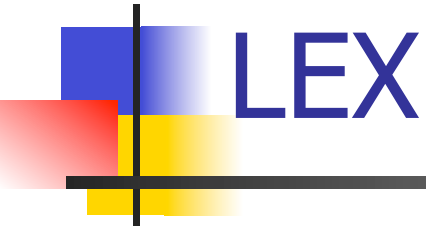
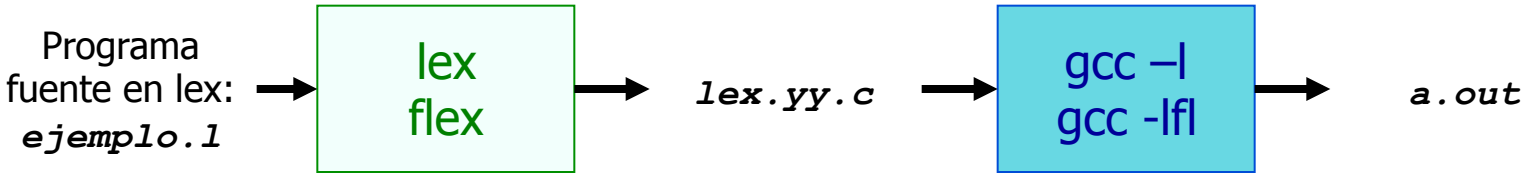


LEX

Generador de analizadores léxicos.

Un analizador léxico (*scanner)* es un programa que reconoce **patrones léxicos** en un texto.

Funcionamiento de lex




```
flex ejemplo.l && gcc –oejemplo lex.yy.c -lfl
```

Formato del archivo fuente

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

NO PUEDE FALTAR!!!

En la sección de reglas, a cada patrón se asocia una acción. Cuando el patrón se encuentra en el texto, se ejecuta la acción asociada.

Si para un patrón no hay acción asociada, no se hace nada.

Regla por default: copiar todo carácter de la entrada en la salida.

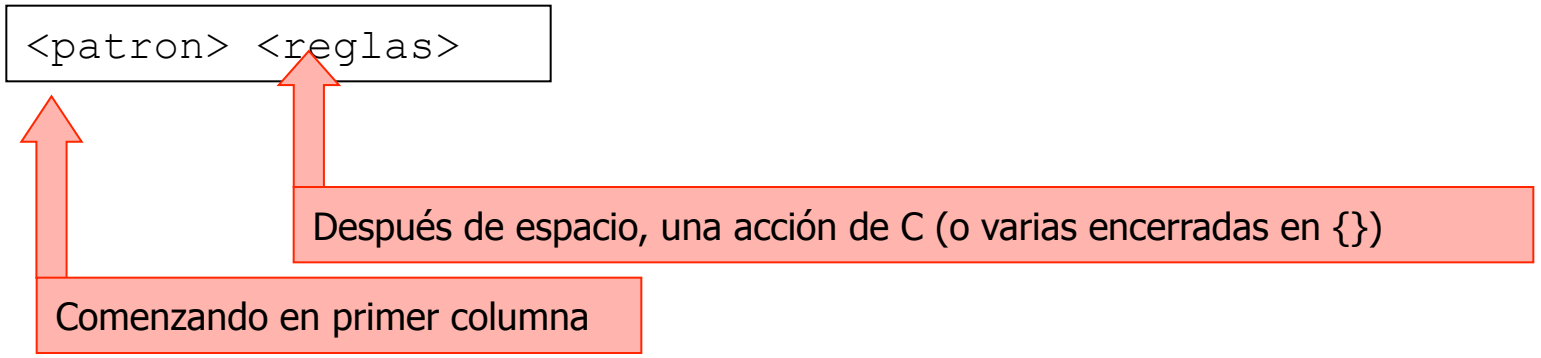
Ejemplos de archivos fuente

```
%%
.       ECHO;
\n      ECHO;
%%
int yywrap(void){
return 1;
}
int main(void){
yylex();
return 0;
}
```

Copia toda la entrada en la salida

El comportamiento del programa generado es el siguiente:
- Imprimir en la salida estándar los lexemas que no se adapten a ningún patrón (ECHO)
- Realizar la acción indicada para los lexemas que se ajustan a un patrón.

Sección de Reglas



Si lo primero que aparece en una línea no está en la primer columna, se copia textual en el archivo C generado.

Patrones

Expresión	*Significado*
.	Cualquier carácter excepto \n
\	Para usar un carácter operador literalmente
R*	Cero o más apariciones de la expresión R
R+	Una o más apariciones de la expresión R
R?	Cero o una aparición de la expresión R
^R	Si R se da al comienzo de la línea
R$	Si R se da al final de la línea
R|S	Si se da la expresión R o la expresión S
R/S	Si se da la expresión R seguida de S
(RS)+	Una o más apariciones de la concatenación
“R+S”	Literalmente el texto R+S
[…]	Todo carácter que esté comprendido dentro de la clase
[^a-z]	Todo carácter que no sea letra minúscula
R{n,m}	De n a m apariciones de R

Ejemplos de archivos fuente

```
%%
[ \t]+  printf(“ “);
```

Si la entrada coincide con uno o más espacios o tabulaciones seguidos, muestra uno solo

```
%%
```

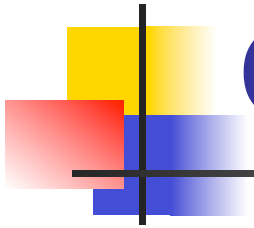
Copia toda la entrada en la salida

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

Copia toda la entrada en la salida

```
%%
[0-9]+   printf(“digito”);
.        ;
```

Si la entrada coincide con uno o más dígitos seguidos, escribe “digito”. Lo demás no lo muestra



Coincidencia con un patrón

- Cuando el scanner se ejecuta, analiza la entrada en búsqueda de cadenas que coincidan con algún patrón.
  - Si una cadena coincide con más de un patrón, se toma la regla que coincida con el token más largo.
  - Si los dos patrones son de igual longitud, se toma el primero que aparezca en la sección
  - Si un patrón no tiene regla asociada, se elimina
  - Si un patrón tiene asociado una “|”, se ejecuta la regla del patrón que sigue.

Ejemplos:

```
%%
“alfa”          putchar(‘a’);
“alfabeto”      printf(“de la ‘a’ a la ‘z’”);
```

```
%{
#include <stdio.h>
int lineno;
%}

%%
begin|end    {printf("%s:%d\n",yytext,lineno);}
\n           lineno++;
.*             ;
```

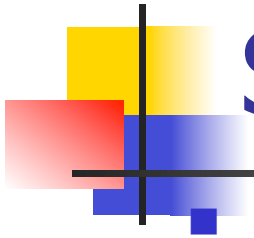
```
%%
“ ”      |
“\t”     |
“\n”     ;
```

```
%%
[ \t\n] ;
```

REJECT

- Indica que hay que ejecutar la siguiente acción que matchea y que pueda corresponderse con la entrada.

```
%{
int s = 0;
int h = 0;
%}
%%
she     {s++;REJECT;}
he      {h++;REJECT;}
\n      |
.       ;
```



Sección de Código de Usuario

- Código que se copia textual en lex.yy.c
  - Ej:

```
int
main(int argc, char **argv)
{
if(argc > 0)
        yyin = fopen(argv[1]. “r”);
else
        yyin = stdin;
yylex();
return 0;
}
```

Sección de código de usuario

- El programa por default es:

```
int yywrap(void){
return 1;
}
int main(void){
yylex();
return 0;
}
```

yylex() es la función que contiene el automata que se define a partir de las reglas.

yywrap() es la función que se invoca cuando se agota la entrada. Retorna 0 si se necesita procesar más cosas.

Variables de lex

Nombre	*Significado*
`char *yytext`	Puntero al inicio del texto que se ha hecho coincidir con un patrón (puntero al inicio del token) También puede ser un arreglo (char [])
`yyleng`	Longitud de yytext
`FILE *yyout`	Output file (stdout por default)
`FILE *yyin`	Input file (stdin por default)
`YY_START`	Indicador de la condición de arranque actual.

Funciones de lex

Nombre	*Significado*
`yymore()`	La proxima vez que se compruebe una coincidencia, el token se anexará al actual yytext.
`yyless(n)`	Retorna todo excepto los n primeros caracteres del token
`unput(c)`	El caracter c se vuelve a colocar en el input. Será el próximo carácter a procesar por el scanner.
`input()`	Lee el siguiente carácter del input stream
`Yyrestart(FILE*)`	Toma el puntero al archivo indicado como nueva entrada.

```
%%
mega-   ECHO;yymore();
kludge  ECHO;
```

Si la entrada es mega-kludge muestra **mega-mega-kludge**

```
%%
foobar  ECHO;yyless(3);
[a-z]+  ECHO;
```

Si la entrada es foobar muestra **foobarbar**

Sección de Definiciones

- Contiene declaraciones de definiciones más simples de nombres (sinónimos)
  - Ej:
```
DIGITO  [0-9]
```
- Contiene declaraciones de condiciones de inicio (start conditions)(%s o %x)
  - Ej:
```
%s AMAYUSC
```
- Contiene código que se copia textual en el archivo .c generado.
  - Ej:
```
%{
#include <ctype.h>
%}
```

Ejemplos:

```
DIGITO  [0-9]
%%
{DIGITO}+       printf(“entero”);
```

```
%{
#include <ctype.h>
%}
%x      PALABRA
MINUSCULA       [a-z]
MAYUSCULA       [A-Z]
%%
<PALABRA>[ \t\n]        ECHO;BEGIN(INITIAL);
<INITIAL>{MAYUSCULA}    ECHO;BEGIN(PALABRA);
<INITIAL>{MINUSCULA} putchar(toupper(*yytext));BEGIN(PALABRA);
```

Condiciones de Arranque

- %S CONDICION

•Se definen en la sección de definiciones com %Start CONDICION
•Una condición de arranque se activa utilizando la acción ”BEGIN(CONDICION)” .
•El estado inicial es siempre por default INTIAL.
•Las reglas se condicionan con: <CONDICION> { regla }
•Hasta que se ejecute la próxima acción BEGIN, las reglas con la condición de arranque dada estarán activas y las reglas con otras condiciones de arranque estarán inactivas.

Diferencia entre %x y %s

- %s Declara condiciones de arranque *inclusivas*
- %x Declara condiciones de arranque *exclusivas*.

•Si la condición de arranque es *inclusiva*, entonces las reglas sin condiciones de arranque también estarán activas.
•Si es *exclusiva*, entonces *solamente* las reglas calificadas con la condición de arranque estarán activas.

Ejemplo %x vs %s

```
%s ejemplo
%% <ejemplo>foo hacer_algo();
bar algo_mas();
```

```
%x ejemplo
%% <ejemplo>foo hacer_algo();
<INITIAL,ejemplo>bar algo_mas();
```

Sin el calificador `<INITIAL,example>',
el patrón `bar' en el segundo ejemplo no estará activo