

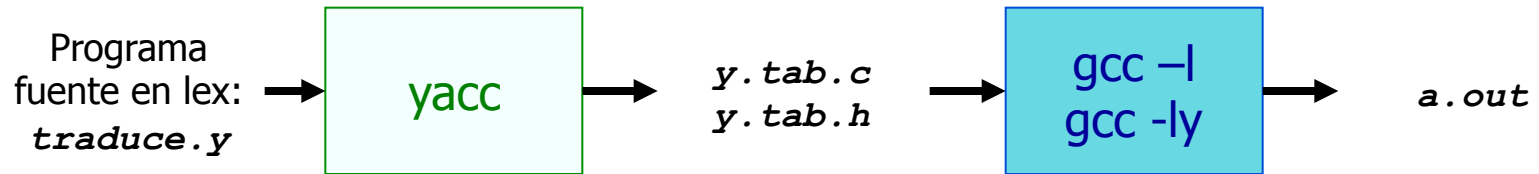


Yet Another Compiler Compiler  
LALR parser generator.



# Funcionamiento de yacc

---



```
yacc -d traduce.y && gcc -ocompiler y.tab.c -ly
```



```
y.tab.h contiene las definiciones de los tokens compartidas con LEX
```



# Declaraciones

---

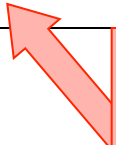
```
{declaraciones}  
%%  
{reglas de traducción}  
%%  
{rutinas de C}
```



# Declaraciones

---


```
%token DIGIT PROGRAM STATEMENT  
%start PROGRAM
```



*%token* se definen los componentes léxicos compartidos entre LEX y YACC y que son generados en el archivo `y.tab.h` para importar desde LEX. Con *%start* se marca cual el es token raíz de todo el árbol sintáctico.

# Reglas de traducción

```
expr      :      termino '*' factor
          {
            $$ = $1 * $3;
          }
          |      factor
          ;
```



Al aplicar la producción indicada se ejecuta la sentencia entre {} asociada a esa producción. \$1 indica el primer operador del lado izquierdo (término), \$2 el segundo ( '\*' ) y \$3 el tercero. Estos operadores son tomados del valor de la variable global yyval que se setea desde LEX cuando se reconoce ese lexema. \$\$ indica el valor que se le asigna a expr.



# Variables globales

---

<i>Nombre</i>	<i>Significado</i>
char <b>*yytext</b>	Puntero al inicio del texto que se ha hecho coincidir con un patrón (puntero al inicio del token) También puede ser un arreglo (char [])
<b>yylen</b>	Longitud de yytext
FILE <b>*yyout</b>	Output file (stdout por default)
FILE <b>*yyin</b>	Input file (stdin por default)
yyval	Variable global que puede utilizarse desde LEX para pasar el valor correspondiente asociado al reconocimiento de un lexema.



# Ejemplos de archivos fuente

---

```
%{
#include <ctype.h>
int yydebug=1;  // útil para debugging..
%}

%token DIGITO
%start linea
%%
linea      :      expr '\n'          { printf("%d\n", $1); }
           ;
Expr       :      expr '+' termino { $$ = $1 + $3; }
           |      termino
Termino    :      termino '*' factor { $$ = $1 * $3; }
           |      factor
           ;
Factor     :      '(' expr ')'      { $$ = $2; }
           |      DIGITO
           ;
%%
.....
```



# Ejemplos de archivos fuente

---

```
.....  
%%  
Yylex() {  
    int c;  
    c = getchar();  
    if (isdigit(c) ) {  
        yylval = c-'0';  
        return DIGIT;  
    }  
    return c;  
}
```