Parte I: Analizador léxico-sintáctico

Objetivo:

Aprender a implementar analizadores léxico sintácticos usando las herramientas Flex y Bison.

Fecha límite entrega Parte I (A. Léxico-Sintáctico):

29/10/2017

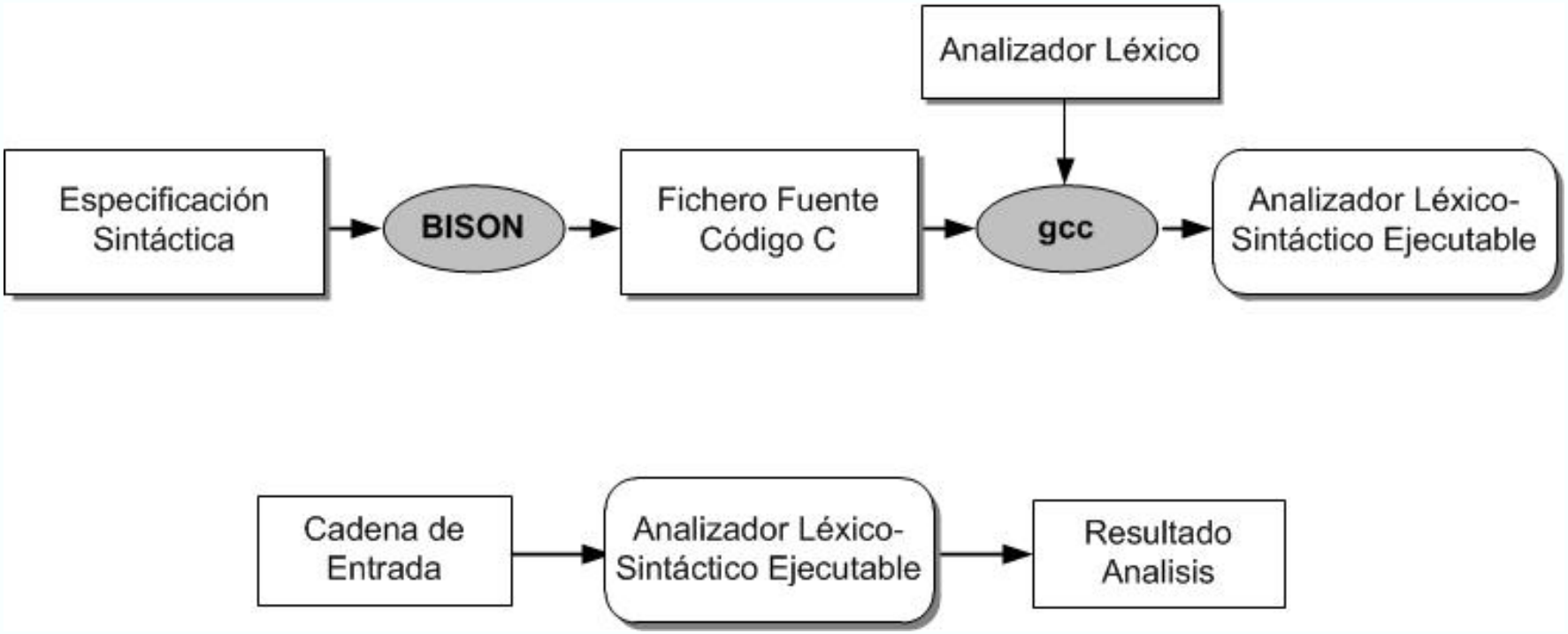
Modo de entrega:

- Grupo
- A través de una tarea PoliformaT

Bison I: Introducción

Lenguajes de Programación y Procesadores de Lenguajes

Uso de Bison


Analizador Léxico
Especificación Sintáctica
BISON
Fichero Fuente Código C
gcc
Analizador Léxico-Sintáctico Ejecutable
Cadena de Entrada
Analizador Léxico-Sintáctico Ejecutable
Resultado Analisis

Especificación Sintáctica

- Fichero de texto con extensión .y

- Dividido en 3 partes (separadas por %%):

  1. Definiciones: Declaraciones C y Bison

  2. Reglas: Reglas gramaticales

  3. Funciones de usuario: Funciones C opcionales

Sección de definiciones

Preámbulo C:

```
%{
  #include <stdio.h>
%}
```

Declaraciones Bison

Identificadores para cada símbolo terminal (token)

%token *nombre_token nombre_token …*

Ejemplo:

```
%token OPSUMA_   OPMULT_
%token CET_   ID_
```

Sección de reglas

Contiene reglas de la forma:

noterminal **:** *lado_derecho* **{** *acciones* **}**

- Primera regla es la del símbolo inicial de la gramática.

- Símbolos no-terminales: Minúsculas
- Símbolos terminales:
  - Identificador en mayúsculas (declarado con %token)
  - Carácter entre comillas simples (‘+’)
  - Cadena de caracteres (“<=“)

Sección de reglas

- Reglas de mismo no-terminal separadas por barra vertical (**|**)
- Última regla de un no-terminal acaba en punto y coma (**;**)

Ejemplo:

```
expresion: termino
         | expresion OPMAS_ termino
         ;
termino: factor
         | termino OPMULT_ factor
         ;
tipo : CTE_
     | ID_
     ;
```

…

Sección código usuario

- Opcional
- El código se copiará en el fichero de salida.
- Se usa para incorporar funciones que aparecen en las acciones de las reglas.

```
main ()
{
  yyparse();
}
```

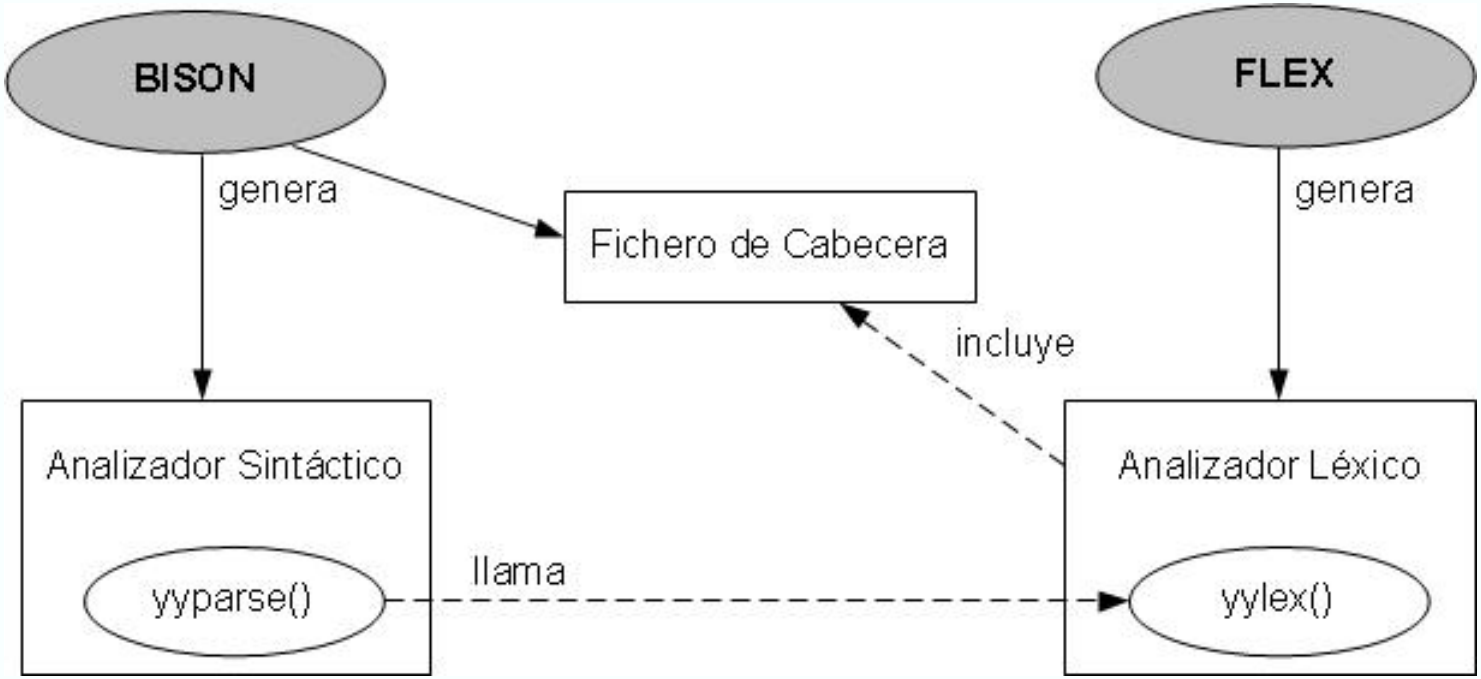
Analizador sintáctico Bsion

- *El AS generado por Bison llama a **yylex()** para obtener un token.*
  - Escrita por el programador o usar la generada por Flex

- *Incluir una llamada a **yyparse()** en **main()** para comenzar el análisis.*

- ***yyparse()** devuelve 0 si llega a fin de fichero yel análisis fue correcto.*

Integración Bison - Flex



Integración Bison - Flex

- Compilar Bison con la opción **'-d'**.

  Genera fichero (por defecto <nombre>.tab.h) con las definiciones de los tokens de las declaraciones %token de Bison

  Ejemplo:

  bison –d <f_entrada>.y  <f_entrada>.tab.c
  <f_entrada>.tab.h

- Incluir este fichero en Flex

  Escribir en la sección de preámbulo de C de flex:

```
#include "<f_entrada>.tab.h"
```

Integración Bison - Flex

Resumen modificaciones a realizar para la integración Bison-Flex

1. Incluir el fichero de cabecera generado por Bison.

2. Las acciones de las reglas léxicas deben ejecutar la sentencia `return` seguida del token detectado.

3. La funcion main debe llamar a `yyparse()` en lugar de llamar a `yylex()`.

Ejemplo Flex

```
%{
  #include <stdio.h>
  #include "asin.h"

%}%
option yylineno
letra [a-zA-Z]
digito [0-9]
%%
[ \t]+                          { }
"+"                             {return(OPSUMA_);}
"*"                             {return(OPMULT_);}
{digito}+                       {return(CTE_);}
{letra}({letra}|{digito})*      {return(ID_);}
.                               {yyerror("Caracter no Valido:");}
%%
```

Ejemplo Bison (1/2)

```
%{
  #include <stdio.h>
  extern int yylineno ;
  extern FILE *yyin ;
  int numErrores ;
%}
%token ID_ CTE_ OPMAS_ OPMULT_
%%
expresion:  expresion OPMAS_ termino
       |    termino
       ;
termino: termino OPMULT_ factor
       |  factor
       ;
factor: CTE_
     |  ID_
     ;
%%
```

Ejemplo Bison (2/2)

```
/* Llamada a yyparse ante un error */
void yyerror (char *msg)  {
   numErrores++;
   fprintf(stdout, "\nError at line %d: %s\n", yylineno, msg);}
}

int main (int argc, char **argv){
   if ((yyin = fopen (argv[1], "r")) == NULL)
        fprintf (stderr, "Fichero no valido \%s", argv[1]);
   yyparse();
}
```