LEX & YACC / FLEX & BISON

Uso prático...

LEX: Tokenizer

Uso do Flex:

Prompt> **flex arquivo.l**
ou
Prompt> **flex –oarquivo.c arquivo.l**

Compilando o programa gerado:

Prompt> **gcc arquivo.c –oarquivo.exe -lfl**

Formato do Arquivo *.l:

```
definitions
%%
rules
%%
user code
```

Conta.l

```
%{
/*
   Conta o total de
   Nro. de linhas e de carcateres
*/
   int nro_lin=0,nro_char=0;
%}
%%
\n      ++nro_lin;
.       ++nro_char;
%%
main()
{
   yylex();
   printf("\nNro. de Linhas : %d\n",nro_lin);
   printf(  "Nro. de Caract.: %d\n",nro_char);
}
```

LEX & YACC / FLEX & BISON

Uso prático... LEX: Tokenizer

Deshtml-v0.l

```
/*  Scanner para Des-HTML-izar  */
%{
   int nlin=0;
%}

TAG           \<(.)*\>
WHITESPACE    [\ \t]
LINHA         [\n]

%%

{TAG}               ;
{LINHA}             { nlin++; }
{WHITESPACE}        { printf("%s",yytext); }
.                   { printf("%s",yytext); }

%%

main()
{
  yylex();
  printf("\nTotal linhas processadas: %d\n",nlin);
}
```

Meu.htm

```
<HTML>
<HEAD>
<TITLE>MEU HTML<BR>
</TITLE>
</HEAD>
<BODY>
HELLO<br>
WORLD<p>
<A HREF="http://xxx.yyy.com/"> LINK
</A>
</BODY>
</HTML>
```

```
Prompt> flex -odeshtml.c deshtml-v0.l
Prompt> gcc deshtml.c -odeshtml.exe -lfl
Prompt> deshtml < meu.htm

HELLOWORLD LINK
Total linhas procesadas: 12

Prompt>
```

Problemas:

1. <...>...<...> => Lex: maior token
2. Como ler direto de um arquivo...

LEX & YACC / FLEX & BISON

Uso prático... LEX: Tokenizer

Deshtml-v1.l

```
/*  Scanner para Des-HTML-izar  */
%{
   int nlin=0;
%}

TAG           \<(.)*\>
WHITESPACE    [\ \t]
LINHA         [\n]

%%

{TAG}         { printf("Tag: %s\n",yytext); }
{LINHA}               { nlin++; }
{WHITESPACE}          { printf("%s",yytext); }
.                     { printf("%s",yytext); }

%%

main()
{
  yylex();
  printf("\nTotal linhas processadas: %d\n",nlin);
}
```

Meu.htm

```
<HTML>
<HEAD>
<TITLE>MEU HTML<BR>
</TITLE>
</HEAD>
<BODY>
HELLO<br>
WORLD<p>
<A HREF="http://xxx.yyy.com/"> LINK
</A>
</BODY>
</HTML>
```

```
Prompt> deshtml < meu.htm
Tag: <HTML>
Tag: <HEAD>
Tag: <TITLE>MEU HTML<BR>
Tag: </TITLE>
Tag: </HEAD>
Tag: <BODY>
HELLOTag: <br>
WORLDTag: <p>
Tag: <A HREF="http://xxx.yyy.com/">
 LINKTag: </A>
Tag: </BODY>
Tag: </HTML>

Total linhas processadas: 12
```

LEX & YACC / FLEX & BISON

Uso prático... LEX: Tokenizer

Deshtml-v2.l

```
/*   Scanner para Des-HTML-izar  */
%{   int nlin=0;    %}

TAG           <[^>]*>
WHITESPACE    [\ \t]
LINHA         [\n]

%%
{TAG}                 { printf("Tag: %s\n",yytext); }
{LINHA}               { nlin++; }
{WHITESPACE}          { printf("%s",yytext); }
.                     { printf("%s",yytext); }
%%

main( argc, argv )
int argc;
char **argv;
{
    ++argv, --argc;  /* pular o nome do programa */
    if ( argc > 0 )  yyin = fopen( argv[0], "rt" );
       else  yyin = stdin;
    yylex();
    printf("\nTotal linhas procesadas: %d\n",nlin);
}
```

```
Prompt> deshtml < meu.htm
Tag: <HTML>
Tag: <HEAD>
Tag: <TITLE>
MEU HTMLTag: <BR>
Tag: </TITLE>
Tag: </HEAD>
Tag: <BODY>
HELLOTag: <br>
WORLDTag: <p>
Tag: <A HREF="http://xxx.yyy.com/">
 LINKTag: </A>
Tag: </BODY>
Tag: </HTML>

Total linhas procesadas: 12
```

```
Prompt> deshtml meu.htm

MEU HTMLHELLOWORLD LINK
Total linhas procesadas: 12

Prompt>
```

Meu.htm

```
<HTML>
<HEAD>
<TITLE>MEU HTML<BR>
</TITLE>
</HEAD>
<BODY>
HELLO<br>
WORLD<p>
<A HREF="http://xxx.yyy.com/"> LINK
</A>
</BODY>
</HTML>
```

LEX & YACC / FLEX & BISON Uso prático... LEX: Tokenizer

Deshtml-v3.l

```
/*  Scanner para Des-HTML-izar  */
%{  int nlin=0; %}

PARAGRAFO     <p>
QUEBRA        <br>
INITITULO     <title>
FIMTITULO     <\/title>
TAG           <[^>]*>
WHITESPACE    [\ \t]
LINHA         [\n]

%%

{PARAGRAFO}          { printf("\n\n"); }
{QUEBRA}             { printf("\n");   }
{INITITULO}          { printf(">>");   }
{FIMTITULO}          { printf("<<\n"); }
{TAG}                ;
{LINHA}              { nlin++; }
{WHITESPACE}         { printf("%s",yytext); }
.                    { printf("%s",yytext); }

%%

main( argc, argv )
int argc;
char **argv;
{
    ++argv, --argc;  /* pular o nome do programa */
    if ( argc > 0 )
            yyin = fopen( argv[0], "r" );
    else
            yyin = stdin;

    yylex();
    printf("\nTotal linhas procesadas: %d\n",nlin);
}
```

Deshtml-v3.bat

```
flex -i -odeshtml.c deshtml-v3.l
gcc deshtml.c -odeshtml.exe -lfl
deshtml meu1.htm
```

Meu1.htm

```
<HTML>
<HEAD>
<TITLE>MEU HTML</TITLE>
</HEAD>
<BODY>
HELLO<br>
WORLD<p>
<A HREF="http://xxx.yyy.com/">LINK</A>
</BODY>
</HTML>
```

```
>>MEU HTML<<
HELLO
WORLD

LINK
Total linhas procesadas: 10
```

Outros programas... VER: Extract-url.l
Como extrair apenas a URL? Ou o e-mail?

LEX & YACC / FLEX & BISON

Uso prático... LEX: Tokenizer

Recursos adicionais do Lex/Flex:

- Acesso ao texto do token: yytext
- Acesso ao tamanho do token: yyleng
- Acesso ao arquivo/dispositivo de entrada:
yyin => Arquivo de entrada (stdin = teclado)
input() => Rotina usada na leitura da entrada (pode ser "sobreposta" e customizada)
unput() => Rotina de apoio usada na leitura de entrada (pode ser adaptada)
output() => Rotina usada para escrita de saída
yywrap() => EOF: executa yywrap... 0 = mais dados de entrada / 1 = fim
yymore() => hyper yymore();
text printf("Token: %s\n",yytext);

- REJECT => pink { npink++; REJECT; } /* overlap*/
ink { nink++; REJECT; }
pin { npin++; REJECT; }

- Start States: um passo em direção a uma gramática (contexto/estados)
%s ou %x
BEGIN ESTADO => Indica estado atual
BEGIN 0 => Reseta estado

- Tokens: %token

LEX & YACC / FLEX & BISON

Uso prático... LEX: Tokenizer

START STATE:

Hide.l

```
%{
  /* Exemplo: Hide/Show */
  int i;
%}

%s OCULTO

%%

<OCULTO>\<\/hide\>  { BEGIN 0; }
<OCULTO>.+          { for (i=0; i< yyleng; i++) printf("X"); }
\<hide\>            { BEGIN OCULTO;    }
.+                  ECHO;
\n                  ECHO;

%%

main()
{
  yylex();
}
```

Hide.htm

```
Este é um teste do <TAG> que
permite ocultar
uma parte de um texto
<hide>
parte oculta
do texto
</hide>
Esta parte não está oculta...
FIM
```

```
flex -i hide.l
gcc lexyy.c -ohide.exe -lfl
hide hide.htm
```

```
Este é um teste do <TAG> que
permite ocultar
uma parte de um texto

XXXXXXXXXXX
XXXXXXXX

Esta parte não está oculta...
FIM
```

LEX & YACC / FLEX & BISON

Uso prático... LEX: Tokenizer - Exemplos

token.l

```
%{
#define NUMBER  400
#define COMMENT 401
#define TEXT    402
#define COMMAND 403
%}
%%
[ \t]+                ;
[0-9]+  |
[0-9]+\.[0-9]+  |
\.[0-9]+              { return NUMBER;  }
#.*                   { return COMMENT; }
\"[^\"]*\"            { return TEXT;    }
[a-zA-Z][a-zA-Z0-9]+  { return COMMAND; }
\n                    { return '\n';    }
%%

#include <stdio.h>

main(argc,argv)
int argc;
char *argv[];
{
  int val;

  while (val = yylex())
     printf("Token: %d\n",val);
}
```

```
Prompt> token

"123"
Token: 402
Token: 10
"Texto" 1.234  ABC1
Token: 402
Token: 400
Token: 403
Token: 10
```

```
flex -i token.l
gcc lexyy.c -otoken.exe -lfl
token
```

LEX & YACC / FLEX & BISON

Uso prático...

YACC: Parser

Uso do Bison:

Prompt> **bison -d arquivo.y** ou...
Prompt> **bison -y -d arquivo.y**

Compilando o programa gerado:

Prompt> **gcc y.tab.c lexyy.c –oarquivo.exe -lfl**
Prompt> **gcc yacc.c lex.c -oprog.exe -lfl**

Formato do Arquivo *.y:

```
definitions
%%
rules
%%
user code
```

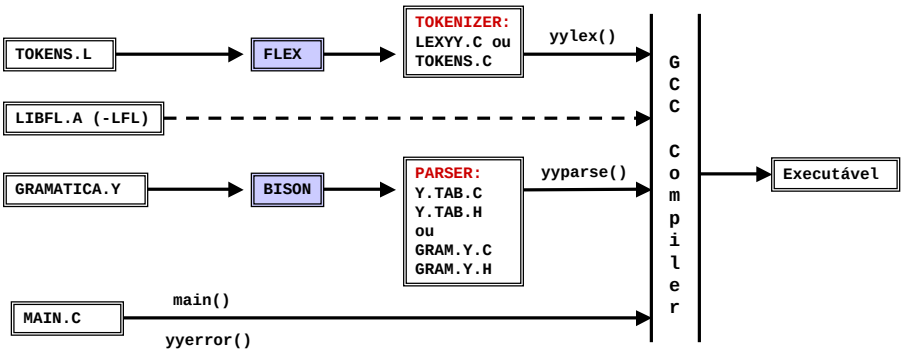
Expr-v0.y

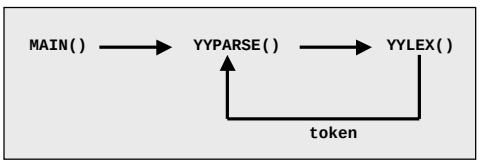
```
%token NAME NUMBER
%start statement

%%
statement: NAME '=' expr   { $$ = $1;
                             printf("(%d)\n", $3); }
   | expr { $$ = $1; printf("= %d\n", $1); }
   ;

expr: expr '+' NUMBER { $$ = $1 + $3; }
    | expr '-' NUMBER { $$ = $1 - $3; }
    | NUMBER          { $$ = $1; }
    ;

%%

#include <stdio.h>
extern FILE *yyin;

yyerror(s)
char *s;
{
  fprintf(stderr,"%"\n",s);
}

main ()
{
 do { yyparse(); }
 while (!feof(yyin));
}
```

* Exemplo da calculadora: Livro Lex/Yacc da O´Reilly Ed.

LEX & YACC / FLEX & BISON

Uso prático... LEX e YACC integrados





YACC:
- Shift / Reduce
- LALR(1) parser
- Aceita gramáticas recursivas (eliminar redundâncias)
- Varredura “left-to-right” (LR)

LEX & YACC / FLEX & BISON

Uso prático... LEX e YACC integrados

Expr-v0.y

```
%token NAME NUMBER
%start statement

%%
statement: NAME '=' expr   { $$ = $1;
                             printf("(%d)\n", $3); }
   | expr { $$ = $1; printf("= %d\n", $1); }
   ;

expr: expr '+' NUMBER { $$ = $1 + $3; }
    | expr '-' NUMBER { $$ = $1 - $3; }
    | NUMBER          { $$ = $1; }
    ;

%%

#include <stdio.h>
extern FILE *yyin;

yyerror(s)
char *s;
{
  fprintf(stderr,"%"\n",s);
}

main ()
{
 do { yyparse(); }
 while (!feof(yyin));
}
```

Nro-var.l

```
%{
#include "y.tab.h"
extern int yylval;
/* valor do token - default: int */
int error_code;
%}

%%

[0-9]+     { yylval = atoi(yytext);
             return NUMBER; }
[a-zA-Z]+  { return NAME; }
[ \t]      ; /* ignore space */
\n         { return 0; }  /* EOF */
.          { return yytext[0]; }

%%
```

```
bison -y -d expr-v0.y
flex nro-var.l
gcc y.tab.c lexyy.c -oexpr.exe -lfl
```

```
Prompt> expr

1+1
= 2
1+10+2
= 31
A=5
(5)
B=2+3
(5)
```

LEX & YACC / FLEX & BISON

Uso prático... LEX e YACC integrados

GRAMÁTICA AMBÍGÜA

```
%token NAME NUMBER
%start statement

%%
statement: NAME '=' expr   { $$ = $1;
                             printf("(%d)\n", $3); }
   | expr { $$ = $1; printf("= %d\n", $1); }
   ;

expr: expr '+' expr { $$ = $1 + $3; }
    | expr '-' expr { $$ = $1 - $3; }
    | expr '*' expr { $$ = $1 * $3; }
    | expr '/' expr { if ($3 == 0)
                        yyerror("Divisão por 0");
                      else
                        $$ = $1 / $3;
                    }
    | -expr         { $$ = -$2; }
    | '(' expr ')'  { $$ = $2;  }
    | NUMBER        { $$ = $1;  }
    ;

%%

/*  2+3*4  =>  (2+3)*4  ou   2+(3*4)  ??  */
```

SOLUÇÃO:

1. Reescrever a gramática eliminando a ambigüidade
2. Usar a definição de precedência do YACC

Exemplo:

```
%left '+' '-'
%left '*' '/'
%nonassoc UMINUS
```

Gramática completa em:
CalcInt.Y

GRAMÁTICA AMBÍGÜA
&
PRECEDÊNCIA DE OPERADORES

EXEMPLO COMPLETO: Ver...

- CalcInt.y
- CalcInt.l

LEX & YACC / FLEX & BISON

Uso prático... LEX e YACC integrados

VALORES DOS SÍMBOLOS/TOKENS

O Yacc possui definida uma maneira de passar os valores dos símbolos (p.exemplo: valor do token NUMBER, ou nome de um token IDENTIFIC).

Tipo pré-definido:

YYSTYPE => Typedef Int, double, char * ou...
%Union

Variável usual: yylval é do tipo YYSTYPE

EXEMPLO COMPLETO: Ver...
- Scalc.y
- Scalc.l

EXEMPLO:

"Typed Tokens"

%union => gera o typedef!

```
typedef union {
  double dval;
  int    ival;
  char   id[30];
} YYSTYPE;

extern YYSTYPE yylval;

...

/* Tokens... Símbolos terminais */

%token <dval> NUMBER
%token <id>   NAME

/* Símbolos não terminais */

%type <dval> expression
```

LEX & YACC / FLEX & BISON

Uso prático... LEX e YACC

• **Conceitos Importantes...**

TABELA DE SÍMBOLOS
- Look Up (Procura se já existe)
- Add Symbol (Insere na Tabela de Símbolos)

```
/* Header for calculator program */
#define NSYMS 20  /* max. number of symbols */

struct symtab {
          char *name;
          double value;
} symtab[NSYMS];

struct symtab *symlook();
```
calcvar.h

Exemplo:

```
%{  #include "calcvar.h"
    #include <string.h>
%}
%union {
          double dval;
          struct symtab *symp;
}
%token <symp> NAME
%token <dval> NUMBER
%left '-' '+'
%left '*' '/'
%nonassoc UMINUS

%type <dval> expression
%%
```
calcvar.y

```
%%
/* look up a symbol table entry, add if not present */
struct symtab *
symlook(s)
char *s;
{
  char *p;
  struct symtab *sp;

  for(sp = symtab; sp < &symtab[NSYMS]; sp++)
  { /* is it already here? */
    if (sp->name && !strcmp(sp->name, s))
       return sp;
    /* is it free */
    if(!sp->name)
    {  sp->name = strdup(s);  return sp; }
    /* otherwise continue to next */
   }
   yyerror("Too many symbols");
   exit(1); /* cannot continue */
} /* symlook */
```
calcvar.y (cont.)

LEX & YACC / FLEX & BISON

Uso prático... LEX e YACC

• **Conceitos Importantes...**

FUNÇÕES E PALAVRAS RESERVADAS...

Exemplo: Yacc

```
%token  SQRT  LOG  EXP
...
%%
expression:...
    |    SQRT '(' expression ')'    { $$ = sqrt($3); }
    |    LOG  '(' expression ')'    { $$ = log($3);  }
    |    EXP  '(' expression ')'    { $$ = exp($3);  }
```

Exemplo: Lex

```
sqrt   return SQRT;
log    return  LOG;
exp    return  EXP;
```

Solução genérica?

Vide Livro Lex/Yacc
Levine; Mason; Brown
O´Reilly Ed. – Pags. 71-77