Compiladores
Análise Sintática
YACC – Gerador de *Parser*

Prof. Dr. Luiz Eduardo G. Martins
(adaptado por Profa Dra Ana Carolina Lorena)
UNIFESP

YACC – Gerador de *Parser*

- Muitos *parsers* podem ser gerados automaticamente, a partir da especificação da GLC
- YACC
  - *Yet Another Compiler-Compiler*
- Existem várias implementações do YACC
  - O GCC disponibiliza a versão *Bison*
  - *Bison* gera uma rotina chamada *yyparse()*
    - *yyparse()* faz a análise sintática de um arquivo de entrada (programa fonte)
      - Retorna 0 (não identificou erro)
      - Retorna 1 (identificou erro)

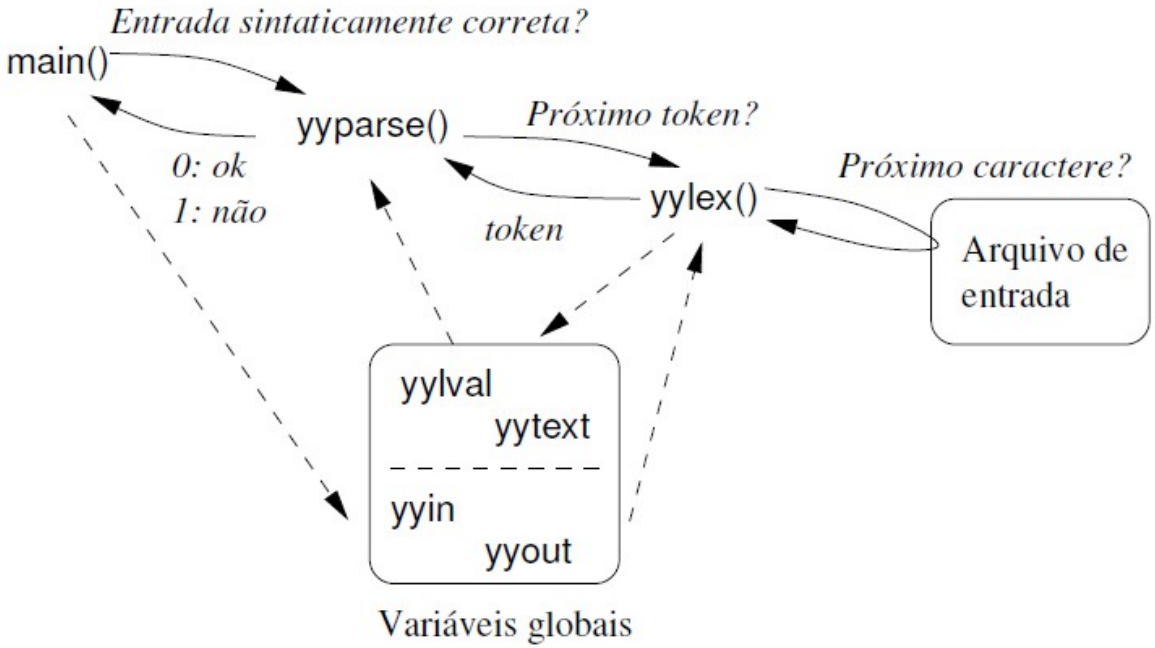
YACC – Gerador de *Parser*

- YACC - *Bison*
  - *yyparse()*
    - É uma função gerada em linguagem C
    - Implementa um *parser LALR(1)*
      - *LA: Look Ahead* (verificação à frente para se obter o próximo token)
      - *L: Left* (entrada processada da esquerda para a direita)
      - *R: Right* (derivação à direita)
      - *(1):* verificação de um símbolo à frente
    - *LALR(1)* é um caso especial de *LR(1)*
    - *LR* é considerado o método de análise sintática mais geral que pode ser aplicado a linguagens e gramáticas passíveis de análise determinística

YACC – Gerador de *Parser*

- YACC
  - *Bison* gera a rotina *yyparse()*
  - *Flex* gera a rotina *yylex()*

Integração lex-yacc



YACC – Gerador de *Parser*

Especificação do analisador

Formato yacc

Como arquivo lex, três seções separadas por %%:

1. Definições e declarações
2. Regras da gramática e ações associadas
3. Código complementar

YACC – Gerador de *Parser*

- Formato básico de arquivo para Yacc (sufixo .y):

{definições} → contém informações de marcas, tipos de dados e código C que deve ir para início da saída (entre %{ e %})
Pode ser vazia

%%

{regras} → regras em formato BNF e ações em C executadas quando a regra é reconhecida (entre chaves)

%%

{rotinas auxiliares} → declarações de procedimentos e funções
Pode ser vazia

YACC – Gerador de *Parser*

Especificação da gramática

Notação estilo BNF:

```
símbolo   :   expansão   ;
```

símbolo lado esquerdo da produção, um símbolo não-terminal

expansão lado direito da produção, a expansão de símbolo em termos de outros símbolos, sejam eles terminais ou não-terminais (definidos em outra produção da gramática)

YACC – Gerador de *Parser*

Produções

- Símbolo sentencial: declaração `%start`
  - Na ausência da declaração, lado esquerdo da primeira produção é considerado o símbolo sentencial
- Se produção é recursiva, recursão à esquerda é preferível
- Expansões alternativas podem ser separadas com o símbolo |

YACC – Gerador de *Parser*

Tokens

Os símbolos terminais das produções podem ser representados de duas formas:

- nomes simbólicos, declarados com o comando `%token`, ou
- representação direta do caractere, se for um único caractere

YACC – Gerador de *Parser*

Tokens

Operadores

Símbolos terminais para operadores podem usar, ao invés de `token`, declaração com associatividade explícita:

%left	operador com associatividade à esquerda
%right	operador com associatividade à direita
%nonassoc	operador não-associativo

Estratégia para eliminar ambigüidade

- Operadores declarados na mesma linha têm mesma precedência
- Operadores declarados na última linha têm maior precedência

YACC – Gerador de *Parser*

- Exemplo

A gramática de expressões (soma ou multiplicação) é usada neste exemplo como a semente para uma calculadora:

$$
\begin{aligned}
E &\rightarrow E + E \\
E &\rightarrow E \times E \\
E &\rightarrow (E) \\
E &\rightarrow v
\end{aligned}
$$

```
%start   entrada
%token   VALOR FIMLIN
%left    SOMA
%left    MULT
%token   ABRPAR FECPAR
```

YACC – Gerador de *Parser*

- Exemplo

A gramática de expressões (soma ou multiplicação) é usada neste exemplo como a semente para uma calculadora:

$$
\begin{aligned}
E &\rightarrow E + E \\
E &\rightarrow E \times E \\
E &\rightarrow (E) \\
E &\rightarrow v
\end{aligned}
$$

%left SOMA Exemplo: A - B - C é interpretada como (A - B) - C
%right SOMA Exemplo: A - B - C é interpretada como A - (B - C)
%nonassoc SOMA Exemplo: A - B - C é interpretada como incorreta

YACC – Gerador de *Parser*

Manipulação das sentenças reconhecidas

Ação semântica

- Código C cuja execução está associada à aplicação da produção
- Especificado entre chaves após a expansão da produção
- Pode referenciar o valor semântico de um token da expansão
  - Notação posicional: $1 é o primeiro token, $2 o segundo...
  - Para tokens, analisador léxico deve definir esse valor por meio da variável `yylval`
  - Um valor semântico pode ser associado ao lado esquerdo ($$), para uso em outras expansões
  - Tipo padrão do valor semântico é `int`, mas pode ser alterado com a redefinição da *string* `YYSTYPE`

YACC – Gerador de *Parser*

- Exemplo:

```
%{
#include <stdio.h>
#include <ctype.h>
%}
%token NUMBER
%left '+' '-'
%left '*'
%%
command: exp {printf("%d\n",$1);};
exp: NUMBER {$$ = $1;} |
     exp '+' exp {$$ = $1 + $3;} |
     exp '-' exp {$$ = $1 - $3;} |
     exp '*' exp {$$ = $1 * $3;} |
     '(' exp ')' {$$ = $2;} ;
%%
main(){ return yyparse(); }
```

```
int yylex(void){
   int c;
   while((c = getchar()) == ' ');
   if(isdigit(c)){
      ungetc(c,stdin);
      scanf("%d",&yylval);
      return(NUMBER);
   }
   if(c == '\n') return 0;
   return(c);
}
int yyerror(char *s){
   fprintf(stderr,"%s\n",s);
   return 0;
}
```

Assume que expressão está em uma linha

YACC – Gerador de *Parser*

Tratamento de erros

error	símbolo não-terminal pré-definido que está associado a sentenças não reconhecidas pela gramática especificada
yyerrok	macro definida em C que limpa a entrada de caracteres para que a análise possa prosseguir
yyerror()	função definida pelo usuário, invocada pelo analisador para apresentação de mensagens de erro

YACC – Gerador de *Parser*

- Recuperação de erros no Yacc
  - Quando analisador detecta erro, fica em estado de erro até encontrar sequência de três marcas legais consecutivas
  - Podem ser usadas produções de erros, que dizem que marcas errôneas podem ser removidas até encontrar ponto de sincronização

YACC – Gerador de *Parser*

- Ambiguidades no Yacc
  - Yacc tem regras internas para eliminar ambiguidades
  - Entre reduzir e empilhar, prefere empilhar
  - Entre duas possíveis reduções, escolhe aquela pela regra que aparece antes no arquivo de especificação
  - Tem também mecanismos ad hoc para especificar precedência e associatividade de uma gramática ambígua

YACC – Gerador de *Parser*

Integração com lex

- Arquivo de cabeçalho produzido com opção `-d`
- Definições de valores para os nomes simbólicos associados aos tokens, declarados na primeira seção do arquivo de especificação de yacc (`acumuly.y`)
- Execução com `bison` (implementação Gnu de yacc)

```
> bison -d acumuly.y
```

> bison –d –v -g acumuly.y (opção –v gera o arquivo acumuly.output, com informações sobre o *parser;* opção –g gera acumuly.dot, com DFA)

YACC – Gerador de *Parser*

Integração com lex

Exemplo: arquivo acumuly.l (6)

acumuly.l

```
%{
#include "acumuly.tab.h"
extern YYSTYPe yylval;
%}
%%
[0-9]+ { yylval = atoi(yytext)
         return NUM;};
\+ {return SOMA;}
\- {return SUBT;}
\* {return MULT;}
\( {return PARE;}
\) {return PARD;}
\n {return LINH;}
%%
```

YACC – Gerador de *Parser*

Geração da aplicação

- Analisador léxico é um programa C, produzido com `flex`

```
> flex acumuly.l
> gcc -c lex.yy.c
```

- Compilação do analisador sintático

```
> gcc -o acc lex.yy.o acumuly.tab.h -ly -lfl
```

YACC – Gerador de *Parser*

Exemplo

Execução (7)

```
> ./acc
1+2+4*5
Resposta: 23
2+2
Resposta: 4
1=2+3+4
=syntax error: 2
1+2
Resposta: 3
[^d]
>
```

Exercício:

Incluir as operações de divisão (x/y) e exponenciação (xy) na calculadora

YACC – Gerador de *Parser*

Sugestões de leitura (Web)

- Bison: Gnu parser generator
  `http://www.gnu.org/software/bison/`

YACC – Gerador de *Parser*

- Bibliografia consultada

LOUDEN, K. C. **Compiladores: princípios e práticas.** São Paulo: Pioneira Thompson Learning, 2004. (Cap. 5)

RICARTE, I. **Introdução à Compilação.** Rio de Janeiro: Editora Campus/Elsevier, 2008. (Cap. 4)