YACC

SSN

Introduction

- What is **YACC ?**

  A tool for automatically generating a parser given a grammar written in a yacc specification (.y file)

- YACC (Yet Another Compiler Compiler) is a program designed to compile a LALR(1) grammar and to produce the source code of the syntactic analyzer of a language produced by this grammar.
- Yacc reads the grammar and generate C code for a parser .

Yacc

- Input to yacc is divided into three sections.

... definitions ...

%%

... rules ...

%%

... subroutines ...

Yacc

- **The definitions section consists of:**
  - token declarations .
  - C code bracketed by **“%{“** and **“%}”.**

- **the rules section consists of**:
  - BNF grammar .

- **the subroutines section** consists of:
  - user subroutines .

yacc& lex in Together

- **The grammar:**

  program -> program expr | ε

  expr -> expr + expr | expr - expr | id

- **Program and expr are nonterminals.**
- **Id are terminals (**tokens returned by lex**)**

- **expression may be :**
  - sum of two expressions .
  - product of two expressions .
  - Or an identifiers

Lex file

```
%{
#include <stdlib.h>
void yyerror(char *);
#include "y.tab.h"
%}

%%

[0-9]+       {
                 yylval = atoi(yytext);
                 return INTEGER;
             }

[-+\n]       return *yytext;

[ \t]        ; /* skip whitespace */

.            yyerror("invalid character");

%%

int yywrap(void) {
    return 1;
}
```

Yacc file

```
%{
    #include <stdio.h>
    int yylex(void);
    void yyerror(char *);
%}

%token INTEGER

%%

program:
        program expr '\n'           { printf("%d\n", $2);
        |
        ;

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

%%

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

int main(void) {
    yyparse();
    return 0;
}
```

Linking lex&yacc