

# Using Yacc

Prof. James L. Frankel  
Harvard University

Version of 8:54 PM 26-Sep-2016  
Copyright © 2016, 2015 James L. Frankel. All rights reserved.

# File Format (1 of 4)

- Extension is .y
- <declarations>  
%%  
<translation rules>  
%%  
<supporting C functions>
- Anything in the <declarations> sections that is delimited by a line with "%{" to a line with "%}" is copied directly to the output C file
- All functions used in <translation rules> should be declared in the <declarations> section and defined in the <supporting C functions> section

# File Format (2 of 4)

- Each line in the <declarations> section (other than those between "%{" and "%}") has the format:
  - %start <nonTerminal>
  - %token <listOfNames>
  - %left <listOfTerminals>
  - %right <listOfTerminals>
  - %nonassoc <listOfTerminals>
- Precedence of tokens is in the order of declaration – lowest precedence first
- All tokens on the same line have the same precedence and associativity

# File Format (3 of 4)

- Translation rules:

$$\begin{aligned} \langle \text{head} \rangle &: \langle \text{body}_1 \rangle \{ \langle \text{semantic action}_1 \rangle \} \\ &| \langle \text{body}_2 \rangle \{ \langle \text{semantic action}_2 \rangle \} \\ &\quad \dots \\ &| \langle \text{body}_n \rangle \{ \langle \text{semantic action}_n \rangle \} \\ &; \end{aligned}$$

- A single quoted character is the terminal symbol
- \$\$ is the attribute associated with the head
- \$i is the attribute associated with the *i*th grammar symbol of the body (either terminal or non-terminal)

# File Format (4 of 4)

- Unquoted strings of letters and digits not declared to be tokens are taken to be non-terminals
- Copying the value is the default action for productions with a single grammar symbol in the body ( $\$ \$ = \$ 1;$ )

# Dealing with Ambiguity in Yacc

- A reduce/reduce conflict is resolved by choosing the conflicting production listed first in the Yacc specification
- A shift/reduce conflict is resolved in favor of shift

# Associativity and Precedence

- Associativity can be assigned to terminals by using %left, %right, and %nonassoc
- As stated above, the precedence associated with tokens is determined by their declaration order – lowest precedence first
- Normally the precedence of a production is the same as that of its rightmost terminal
- This can be changed by appending  

```
%prec <terminal>
```

  
to a production body
  - This sets the precedence of that production to the same precedence as <terminal>

# Including the Lexer

- Specify  
    `#include "lex.yy.c"`  
in the third part of the Yacc input file to include the lexer built by Lex
- Declare `yylex` in the declarations section using:  
    `int yylex(void);`

# Errors Detected by Yacc

- The function `yyerror` is called by Yacc whenever an error is detected when your resulting Yacc file (*i.e.*, your parser) is executing
- A single parameter is passed to `yyerror` of the type:  
`char *`
- That string will contain a description of the error detected by Yacc
- Declare `yyerror` in the declarations section using:  
`void yyerror(char *s);`

# Removing warnings emitted by gcc

- When building the Yacc & Lex project with gcc switches `-pedantic` and `-Wall`, you may see warnings for:
  - `'yyunput' defined but not used [-Wunused-function]`
  - `'input' defined but not used [-Wunused-function]`
- These can be removed by adding the following lines to your lex file in the declarations section:
  - `%option nounput`
  - `%option noinput`

# Compiling a Yacc file

- `lex lexer.lex`
- `yacc parser.y`
- `gcc -pedantic -Wall y.tab.c -ly -lfl -o parser`
  - `-pedantic` means to issue all warnings demanded by Standard C
  - `-Wall` means to issue many warnings that some users consider questionable
  - `y.tab.c` is the output of Yacc
  - `-ly` means to link with the Yacc libraries
  - `-lfl` means to link with the flex libraries (on some systems, `-ll` may be needed to link with lex libraries)
  - `-o` is used to specify the name of the executable file

# Examining Shift/Reduce and Reduce/Reduce Conflicts

- Invoking Yacc or Bison with the -v switch will cause a y.output file to be created
- The y.output file will contain a human-readable description of what the parser will do in each of its states
- Examining the y.output file will show how Yacc or Bison is finding and resolving shift/reduce and reduce/reduce conflicts