**YACC**

The computer program **YACC** is a parser generator developed by Stephen C. Johnson at AT&T for the Unix operating system. The name is an acronym for “Yet Another Compiler Compiler”. It generates a parser (the part of a compiler that tries to make syntactic sense of the source code) based on an analytic grammar written in a notation similar to BNF. Historically, YACC has generated the code for the parser in the C Programming language.

YACC used to be available as the default parser generator on most Unix systems. It has since been supplanted as the default by more recent, largely compatible, programs such as Berkeley Yacc, GNU bison, MKS yacc and Abraxas pcyacc. An updated version of the original AT&T version is included as part of Sun’s OpenSolaris project. Each offers slight improvements and additional features over the original YACC, but the concept has remained the same. YACC has also been rewritten for other languages, including Ratfor, ML, Ada, Pascal, Java, Python and Common Lisp.

The parser generated by YACC requires a lexical analyzer. Lexical analyzer generators, such as LEX or Flex are widely available. The IEEE POSIX P1003.2 standard defines the functionality and requirements for both LEX and YACC. Some versions of AT&T YACC have become open source. For example, source code (for different implementations) is available with the standard distributions of Plan 9 and OpenSolaris.

**Structure of YACC file**

A YACC program consists of three sections separated with the %% delimiter:

*definitions*

*%%*

*rules*

*%%*

*auxiliary procedures*

The YACC language is free-format: whitespace (blanks, tabs and newlines) is ignored, except if it serves as a delimiter. Comments have the C-like format /\* ... \*/. They are treated as whitespace. Grammar symbols are denoted by identifiers which have the usual form (letter, including underscore, followed by a sequence of letters and digits; upper- and lowercase is distinct). The YACC language also has some keywords which always start with the % character. Literals are denoted by characters enclosed in single quotes. The usual C-like escapes are recognized:

* \n denotes newline
* \r denotes carriage return
* \t denotes tab
* \b denotes backspace
* \f denotes form feed
* \nnn denotes character no. nnn in octal base

1. **Definitions**

The first section of a YACC grammar serves to define the symbols used in the grammar. It may contain the following type of definitions:

* 1. start symbol definition
  2. terminal definition
  3. precedence definition
  4. type definition
  5. turbo pascal definition

1. **Rules**

Grammar rules have the format

*name : symbol ... ;*

The left-hand side of a rule must be an identifier (which denotes a non-terminal symbol). The right-hand side may be an arbitrary (possibly empty) sequence of non-terminal and terminal symbols (including literals enclosed in single quotes). The terminating semicolon may also be omitted. Different rules for the same left-hand side symbols may be written using the ‘|’ character to separate the different alternatives.

1. **Auxiliary Procedures**

The third section of a YACC program is optional. If it is present, it may contain any Turbo Pascal code (such as supporting routines or a main program) which is tacked on to the end of the output file.

**Usage**

*yacc [options] yacc-file[.y] [output-file[.pas]]*

Options

1. -v “Verbose:" YACC generates a readable description of the generated parser, written to yacc-file with new extension .lst.
2. -d “Debug:" YACC generates parser with debugging output.