9th February 2012

Role of the Parser

In the compiler model, the parser obtains a string of tokens from the lexical analyser, and verifies that the string can be generated by the grammar for the source language.
The parser returns any syntax error for the source language.

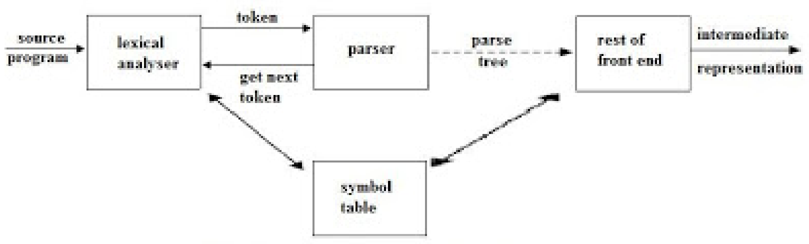

Fig 2.1 Position of parser in compiler model

[http://1.bp.blogspot.com/-rQj15mJzm2I/TzOMd_8xSfI/AAAAAAAAAF4/h60sPn1m9Og/s1600/aas.jpg]

There are three general types' parsers for grammars.
Universal parsing methods such as the *Cocke-Younger-Kasami algorithm* and *Earley's algorithm* can parse any grammar. These methods are too inefficient to use in production compilers.
The methods commonly used in compilers are classified as either *top-down parsing* or *bottom-up parsing.*
***Top-down parsers* build parse trees from the *top (root)* to the *bottom (leaves)*.**
***Bottom-up parsers* build parse trees from the *leaves* and work up to the *root*.**
In both case input to the parser is scanned from left to right, one symbol at a time.
The output of the parser is some representation of the parse tree for the stream of tokens.
There are number of tasks that might be conducted during parsing. Such as;

- **Collecting information about various tokens into the symbol table.**
- **Performing type checking and other kinds of semantic analysis.**
- **Generating intermediate code.**

Syntax Error Handling:

- **Planning the error handling right from the start can both simplify the structure of a compiler and improve its response to errors.**
- **The program can contain errors at many different levels. e.g.,**
  - § ***Lexical* – such as misspelling an identifier, keyword, or operator.**
  - § ***Syntax* – such as an arithmetic expression with unbalanced parenthesis.**
  - § ***Semantic* – such as an operator applied to an incompatible operand.**
  - § ***Logical* – such as an infinitely recursive call.**

Much of the error detection and recovery in a compiler is centered on the syntax analysis phase.

- **One reason for this is that many errors are syntactic in nature or are exposed when the stream of tokens coming from the lexical analyser disobeys the grammatical rules defining the programming language.**

- Another is the precision of modern parsing methods; they can detect the presence of syntactic errors in programs very efficiently.

The error handler in a parser has simple goals:

- It should the presence of errors clearly and accurately.
- It should recover from each error quickly enough to be able to detect subsequent errors.
- It should not significantly slow down the processing of correct programs.

Error-Recovery Strategies:

- There are many different general strategies that a parser can employ to recover from a syntactic error.
    - § Panic mode
    - § Phrase level
    - § Error production
    - § Global correction
- ***Panic mode:***
    - § This is used by most parsing methods.
    - § On discovering an error, the parser discards input symbols one at a time until one of a designated set of synchronizing tokens (delimiters; such as; semicolon or end) is found.
    - § Panic mode correction often skips a considerable amount of input without checking it for additional errors.
    - § It is simple.
- ***Phrase-level recovery:***
    - § On discovering an error; the parser may perform local correction on the remaining input; i.e., it may replace a prefix of the remaining input by some string that allows the parser to continue.
    - § e.g., local correction would be to replace a comma by a semicolon, deleting an extraneous semicolon, or insert a missing semicolon.
    - § Its major drawback is the difficulty it has in coping with situations in which the actual error has occurred before the point of detection.
- ***Error productions:***
    - § If an error production is used by the parser, can generate appropriate error diagnostics to indicate the erroneous construct that has been recognized in the input.
- ***Global correction:***
    - § Given an incorrect input string x and grammar G, the algorithm will find a parse tree for a related string y, such that the number of insertions, deletions and changes of tokens required to transform x into y is as small as possible.

Posted 9th February 2012 by Arun Patrick

Labels: compiler design, compiler design notes, compiler notes, role of parser

Enter your comment...

Comment as: sagar giri (Goo ▼

Sign out

Publish Preview

☐ Notify me