Compiler Design
Lexical Analysis
Input Bufering

conf. dr. ing. Ciprian-Bogdan Chirila

chirila@cs.upt.ro

http://www.cs.upt.ro/~chirila

Outline

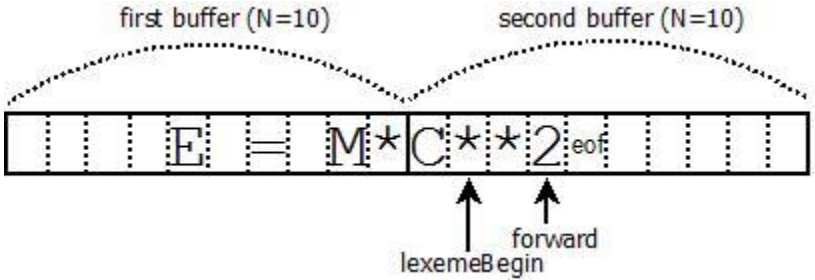
- Input Buffering
  - Buffer Pairs
  - Sentinels

Input Buffering

- How to speed the reading of source program ?
- to look one additional character ahead
- e.g.
  - to see the end of an **identifier** you must see a character
    - which is not a letter or a digit
    - not a part of the lexeme for **id**
  - in C
    - - ,= , <
    - ->, ==, <=
- two buffer scheme that handles large lookaheads safely
- sentinels – improvement which saves time checking buffer ends

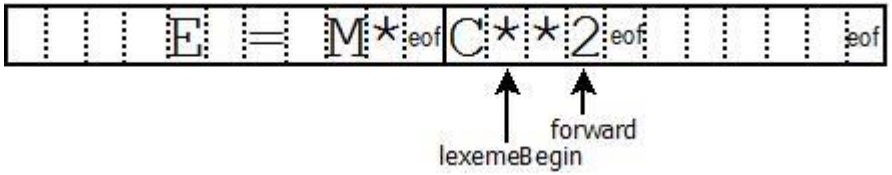
Buffer Pairs

- Buffer size N
- N = size of a disk block (4096)
- read N characters into a buffer
  - one system call
  - not one call per character
- read < N characters we encounter **eof**
- two pointers to the input are maintained
  - *lexemeBegin* – marks the beginning of the current lexeme
  - *forward* – scans ahead until a pattern match is found



Sentinels

- *forward* pointer
  - to test if it is at the end of the buffer
  - to determine what character is read (multiway branch)
- sentinel
  - added at each buffer end
  - can not be part of the source program
  - character **eof** is a natural choice
    - retains the role of entire input end
    - when appears other than at the end of a buffer it means that the input is at an end



Lookahead Code with Sentinels

```
switch(*forward++)
{
    case eof:
        if(forward is at the end of the first buffer)
        {
            reload second buffer;
            forward = beginning of the second buffer;
        }
        elseif(forward is at the end of the second buffer)
        {
            reload first buffer;
            forward = beginning the first buffer;
        }
        else
        /* eof within a buffer marks the end of input */
            terminate lexical analysis;
        break;
    cases for the other characters
}
```

Potential Problems

- usually
  - lexemes are short
  - 1-2 characters lookahead are sufficient
- problem: running out of buffer space
  - when N = 3,4,5 x 1000
  - long character strings > N
- solution: concatenation of string components by grammar rules (like in Java using the + operator to catenate multiline strings)
- long lookahead
  - languages where keywords are not reserved
  - in PL/I:
    - DECLARE (ARG1, ARG2,…)
    - ambiguous identifier resolved by the parser (keyword or procedure name)

Bibliography

- Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman – Compilers, Principles, Techniques and Tools, Second Edition, 2007