CS480
Translators

Introduction to Lexical Analysis
Chap. 2
Odds and Ends

• Assignment #2 is posted
  – Please email me your teams, if not working alone
• Demo your Assignment #1
Quiz #2

Question 1
• What is the language of the following CFG:
  \[ S \rightarrow bSbb \mid A \]
  \[ A \rightarrow aA \mid \varepsilon \]
• Provide the parse tree for bbaaabbbaabbb.

Question 2
• Provide the abstract syntax tree for the following:
  \[ -1 + 2 \times 3.0^4.7 / 6 \]
• What is the post-order traversal of the tree.
• Explain how you would implement this in gforth.
Lexical Analysis

• What is its purpose?
• What is the difference in a token vs. lexeme?
• Example:
  – The Brown Fox
  – if (i !=32) then j := 12
• Are spaces important?
The Role of the Scanner...

Figure 3.1: Interactions between the lexical analyzer and the parser
while (i > 0)
  i = i - 2;
What’s new in this grammar?

\[
\begin{align*}
expr & \rightarrow \quad expr + term \quad \{ \text{print}('+') \} \\
& \quad | \quad expr - term \quad \{ \text{print}('-') \} \\
& \quad | \quad term \\

term & \rightarrow \quad term \times factor \quad \{ \text{print}('*') \} \\
& \quad | \quad term / factor \quad \{ \text{print}('/') \} \\
& \quad | \quad factor \\

factor & \rightarrow \quad ( expr ) \\
& \quad | \quad num \quad \{ \text{print}(\text{num}.\text{value}) \} \\
& \quad | \quad id \quad \{ \text{print}(\text{id}.\text{lexeme}) \}
\end{align*}
\]

Figure 2.28: Actions for translating into postfix notation
The Scanner

```c
for ( ; ; peek = next input character ) {
    if ( peek is a blank or a tab ) do nothing;
    else if ( peek is a newline ) line = line+1;
    else break;
}
```

Figure 2.29: Skipping white space

- What is the purpose of line?
- What is the purpose of peek?
Reading Ahead

• Read the next char, it is an “i”
• Could be int, if, or an identifier, so read next char, “f”
• Could be if, could still be an identifier, so read next char, “(”
• Oops, we’ve gone too far, push back “(”
Buffers

• Why is this important?

• Ways to implement:
  – Two pointers into buffer (start_char, look_ahead)
  – Push back buffer (peek)
The Lexical Analyzer

\[
\text{if ( \text{peek holds a digit} ) } \{ \\
\quad v = 0; \\
\quad \text{do } \{ \\
\quad\quad v = v \times 10 + \text{integer value of digit \text{peek};} \\
\quad\quad \text{peek} = \text{next input character;} \\
\quad\} \text{ while ( \text{peek holds a digit} );} \\
\quad \text{return token } \langle \text{num, v} \rangle; \\
\}
\]

Figure 2.30: Grouping digits into integers
Keywords vs. Identifiers

• count = count + increment;

  <id, “count”> <=> <id, “count”> <+> <id, “increment”> <;>

• How do we know count is an id vs. keyword?
• Why use a hash table?
• What is in the hash table?
How to distinguish words?

```java
if ( peek holds a letter ) {
    collect letters or digits into a buffer b;
    s = string formed from the characters in b;
    w = token returned by words.get(s);
    if ( w is not null ) return w;
    else {
        Enter the key-value pair (s, ⟨id, s⟩) into words
        return token ⟨id, s⟩;
    }
}
```

Figure 2.31: Distinguishing keywords from identifiers
Reading/Assignment

• Milestone 2
• Read Chap. 2.6 - 2.7 and Chap. 3