Compilers

Error Handling
• **Purpose of the compiler is**  
  – To detect non-valid programs  
  – To translate the valid ones  

• **Many kinds of possible errors (e.g. in C)**

<table>
<thead>
<tr>
<th>Error kind</th>
<th>Example</th>
<th>Detected by ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lexical</td>
<td>... $ ...</td>
<td>Lexer</td>
</tr>
<tr>
<td>Syntax</td>
<td>... x *% ...</td>
<td>Parser</td>
</tr>
<tr>
<td>Semantic</td>
<td>... int x; y = x(3); ...</td>
<td>Type checker</td>
</tr>
<tr>
<td>Correctness</td>
<td>your favorite program</td>
<td>Tester/User</td>
</tr>
</tbody>
</table>
Error Handling

• Error handler should
  – Report errors accurately and clearly
  – Recover from an error quickly
  – Not slow down compilation of valid code
Error Handling

- Panic mode
- Error productions
- Automatic local or global correction
• Panic mode is simplest, most popular method

• When an error is detected:
  – Discard tokens until one with a clear role is found
  – Continue from there

• Looking for synchronizing tokens
  – Typically the statement or expression terminators
• Consider the erroneous expression
  \((1 + + 2) + 3\)
• Panic-mode recovery:
  – Skip ahead to next integer and then continue
• Bison: use the special terminal error to describe how much input to skip
  
  \[
  E \rightarrow \text{int} \mid E + E \mid (E) \mid \text{error int} \mid (\text{error})
  \]
Error Handling

- Error productions
  - specify known common mistakes in the grammar

- Example:
  - Write $5 \times$ instead of $5 \times x$
  - Add the production $E \rightarrow \ldots | E E$

- Disadvantage
  - Complicates the grammar
Error Handling

• Idea: find a correct “nearby” program
  – Try token insertions and deletions
  – Exhaustive search

• Disadvantages:
  – Hard to implement
  – Slows down parsing of correct programs
  – “Nearby” is not necessarily “the intended” program
Error Handling

• Past
  – Slow recompilation cycle (even once a day)
  – Find as many errors in one cycle as possible

• Present
  – Quick recompilation cycle
  – Users tend to correct one error/cycle
  – Complex error recovery is less compelling