

Administrative

- (Repeat) CS staff (not us!) will process the wait list as space becomes available.
- (Repeat) If you decide to drop, please inform TeleBEARS as soon as possible to let others in.
- Register with the course electronically from your account (whether or not enrolled yet) *by Monday*.
- CSUA (Computer Science Undergraduates Association):
 - Web site: <http://csua.berkeley.edu>
 - Mon., Jan 24 at 5PM, 337 Soda: First General Meeting (+ free dinner)
 - Tue, Jan 25 at 6PM, 306 Soda: vi and emacs Help Session
 - Thu, Jan 27, 6PM, Wozniak Lounge (4th floor Soda): Mentoring Meeting - Pair up with an UD major.

Syntactic Analysis

- First part of the "front end" of compiler.
- Purpose: convert string of characters (from file, editor input buffer, etc.) into a tree or other intermediate form.
- Like all phases of compiler:
 - Eliminates (or moves aside) information not needed by later phases.
 - Detects errors and substitutes something correct.
 - Gathers and converts other information to convenient form.
- Traditionally divided into a *lexical analyzer (scanner)* and (*context-free*) *parser*.
- Scanner converts stream of characters into stream of *tokens*: more convenient chunks.

Example of Scanner Output

- Original program might be

```
if(i== j)
    z = 0; /* No work needed */
else
    z= 1;
```

- Or as the translator sees it:

```
\tif(i== j)\n\t\tz = 0; /* No work needed */\n\telse\n\t\tz= 1;
```

- Scanner is intended to convert this to something like:

```
IF, LPAR, ID("i"), EQUALS, ID("j"), RPAR, ID("z"), ASSIGN,  
INTLIT("0"), SEMI, ELSE, ID("z"), ASSIGN, INTLIT("1"), SEMI
```

- That is, a sequence of objects, each with a *syntactic category* (IF, etc.) of interest to the parser, and possibly a *lexical value* of interest to later things.
- May also add source-location information for error messages.

Internal representations

- Syntactic categories (symbolic in example) can simply be members of a finite set of integers or other discrete values:
 - (convenient for equality comparison or indexing tables)
- The lexical value could be anything. In our example, it is the *lexeme* itself (the actual source characters).
- So one might define:

```
class Token {  
    enum SyntacticCategory { IF, LPAR, ID, EQUALS, RPAR, ASSIGN, ... };  
    SyntacticCategory syntax;  
    Object value;  
    Location sourcePosition;  
    ...  
}
```

Strategy Overview for Scanner

- *Regular expressions* can describe lexemes.
- *Finite-state automata (FSAs)*: abstract machines that can recognize languages.
- *Deterministic finite-state automata (DFAs)*: subset of FSAs easily converted into programs.
- Can convert regular expressions to FSAs.
- Any FSA can be converted to a DFA (if not already there), and hence to program.
- The total process from regular expression to program is automatable (in our course, `flex` and `jflex`).