CS-701 Lecture 22

April 26, 2001

Reading a Command Line

- Goal is to get an array of strings representing command-line tokens.
  - Allow quotes, redirection symbols, pipes, semicolons, ampersands (, …?) as tokens.
  - Allow user to edit the command line.
  - Support arrow keys.
- Low-level line reader
  - Different from *fgets()* in that the program examines each character as it is typed.
    - Advantage: Support for arrow keys
    - Disadvantage: Have to write the line editing functions
- High-level line processor
  - Takes the line returned by the low-level reader and tokenizes it.
    - Has to process the line a character at a time
    - Has to recognize quotes (three kinds) and escapes (\)

Assignment 5

- Add features to your shell.
  - I will recommend a “standard set” of features to add, but you are free to extend or substitute this set as you wish.
  - Might serve as the basis for a CS-731 project.
- Standard Set (tentative)
  - Define and substitute environment variables
  - Define and substitute command aliases
  - Process double quotes.

Three Kinds of Quotes

- Double Quotes (")
  - Allows variable substitution inside them
    - print “The exit code was $?”
- Single Quotes (')
  - Same as double, but suppress variable substitution.
- Back Quotes (`)
  - Command substitution
    - POSIX and ksh prefer $(…)
- Escapes (\ )
  - Suppresses any special meaning of the character immediately following it.
    - Print “The value of \$? is $?”
    - Allows you to split long lines by escaping newlines

Command Line Features

- Multiple commands per line, separated by semicolons or pipes.
  - Pipelines (Stevens section 14.2) can be medium; doing it “right” is big.
  - Parenthesis and brace grouping (big)
- Substitutions:
  - Aliases (small)
  - Environment and shell variables (medium)
  - Pattern matching (big)
  - Command substitution (fun)
- Redirection and background processing.
  - Job control is big