Project 2 – Shell

David Grisham

30 September, 2015

Usage

Run `make` in the `src/` directory to build the project. If all goes well, then you should be left with an executable called `myshell` in the `src/` directory. Running this will open the shell.

List of Files

- `shell.cpp`
  - main shell process that sets up the environment, then (basically) follows these steps:
    1. display prompt to user
    2. read in user input (currently only accepts built-in commands)
    3. tokenize input and handle the following
       - local variable assignment
       - variable substitution
       - alias substitution
    4. execute the function corresponding to the user input
- `builtins.cpp`
  - contains an implementation for each command that's built-in to the shell. see the 'Builtins' section for more details
- `builtins.h`
  - header file for the functions defined in `builtins.cpp`

Builtins

The following builtin functions are currently supported by this shell (Note: terms surround by `<>` (e.g. `<term>`) should be replaced by user input.):

- `ls`, `ls <directory>`
  - with no argument, lists the files in the current working directory
  - otherwise, lists the files in `<directory>`
- `cd <directory>`
  - if `cd` is called with no arguments, the shell will change into the directory defined by $HOME (usually `/home/<username>`)
  - otherwise, changes the current working directory to `<directory>`
- `echo <arg>`
  - prints `<arg>` to stdout, followed by a newline
- `pwd` displays the value held in the `$PWD` environment variable. It is left up to other functions (like `cd`) to update `$PWD` when necessary
- `alias`, `alias <cmd>=<def>`
  - with no arguments, `alias` will show all of the currenty defined aliases
  - otherwise, `<cmd>` is the alias name, and `<def>` is the command that should be called when `<cmd>` is called
- `unalias -a`, `unalias <cmd>`
  - with the `-a` option, unalias will remove all currently defined aliases
  - `<cmd>` is the name of the alias that should be removed from the aliases list
- `exit`
  - exists the shell with an exit code EXIT_SUCCESS
- `history`, `history <N>`
  - with no arguments, `history` displays the entire history of shell commands ran in the current session
  - `<N>` is the number of commands in the history to display, starting with the most recent
- `!!`
  - executes the last command that was ran
- `!<N>`
  - executes the `N`th command in the history
- `!-<N>`
  - executes the `N`th-preceding command (same as Bash)

Unusual / Interesting Features

As noted in the Builtins section:

- `cd` defaults to the home directory when called with no arguments
- `history <N>` can be used to see the `N` most recent lines in the history
- `!-<N>` can be used to execute the `N`th-preceding line in the history (`N` is the number of the desired line (in the history) relative to the current line)

As point of potential interest, the way in which aliases are handled allows the user to define nested aliases without issue. For example:

```
prompt > alias e=echo
prompt > e hello
hello
prompt > alias a=e
prompt > a hello
hello
```

Number of Hours Spent

Intermediate 1 – Around 6 hours