CShell Project

By Anthony Liang, Sam Xu, Shaeq Ahmed

FEATURES:

- Intro screen (Cool ASCII text and colors)
- Basic shell functionalities and built in commands
- Implemented Cd and exit
- Simple Redirection
    - Stdin (<)
    - Stdout (>)
    - Pipe (|)
        - Can link multiple pipes together
- More advanced redirection (&>,2>>,>>)
- Dynamically reallocation of user input
- Parse multiple commands with ;
- Ignores weird spacing ("ls -l")
- Prints bash prompt in linux format
    - <user>@<hostname>:<cwd>$
    - ~ if current working directory is home directory
- Implemented autocomplete binded to TAB
- Stores command history, can access with UP arrow key
- Cd prints out error statement if doesn't exist

FILES & FUNCTION HEADERS:

void introScreen();
Inputs: None
Returns: None
Explanation: Prints a few pretty lines when shell is started.

void makePrompt();
Inputs: None
Returns: None
Explanation: Prints out the user prompt in linux format, added custom colors to make it more aesthetic

int cshell_cd(char *args[]);
Inputs: char *args[]
Returns: If the directory exists, chdir into it, otherwise return -1. 1 if cd is called by itself, chdir user back to home directory
Explanation: Built-in cd command, catches and prints DNE errors

void cshell_exec(char **args);
Inputs: char **args

Returns: If pid == -1 (forking failure), return an error message. If pid == 0 (fork successful), child process runs.

Explanation: Fork parent process. Catches any signals and execvp to run the commands.

void cshell_io(char *args[], char *i, char *o, int option);
Inputs: char *args[], char *i, char *o, int option)
Returns: None
Explanation: Helper function that helps control the writing and reading of files.

void cshell_pipeHandle(char *args[]);
Inputs: char *args[]
Returns: None
Explanation: Helper function responsible for the helper.

int cshell_run(char *args[]);
Inputs: char *args[]
Returns: int
Explanation: Method used to handle the commands entered via the standard loop.

char **cshell_split_line(char *line, char *delim);
Inputs: char *line, char *delim
Returns: Array of pointers
Explanation: Function for splitting the commands by whitespace (‘\t\r\n\a’) or any specified delimiter

char **parse_semicolon(char *line);
Inputs: char *line
Returns: Array of pointers
Explanation: Parses multiple commands with ;. For example the line “ls -l;echo hello” would be split into an array of two pointers “ls -l” and “echo hello”

int is_empty(char * line)
Inputs: char *line
Returns: 0 or 1
Explanation: Checks whether or not the input line contains only whitespace. It returns 1 if is only white space, notifying the code to ask for input again. It returns 0 if it contains arguments.

`int main(int argc, char *argv[]);`
Inputs: `char argc, char **argv`
Returns: `EXIT_SUCCESS`
Explanation: Main process for program. Prints our pretty intro screen and makePrompt. Reads, parses, and executes command(s). Also implements autocompletion and command history.