Shell Project

Group Names: Richard Tran & Alexandra Zhang

Parsing

- parser.c
- cmd.h

cmd.h

- holds only the cmdListStruct:
    - int argc;
        - The number of arguments including the command itself
    - char* argv[51] ;
        - The array of char ptrs. Used 51 to prevent error rather than 50. If parseCmds exceeds 50 in it's buffer, it returns NULL back to our shell and prints "Too many arguments. Only 50 are allowed"
    - struct cmdListStruct* next;
        - Points to the next element in the linked_list.

parser.c

- cmdList* insertCmd(char* arguments[], int count, cmdList* cmdTail);
    - called by parseCmd when the buffer is ready to be loaded into a cmdList struct
    - Use cmdTail to keep track of the last node in the linked-list to keep a fast insertion
    - returns NULL if empty string
- cmdList* parseCmds(char* input);

  - Uses a ptrFlag, quoteFlag, buffer, bufferIndex, cmdListHead, cmdListTail
  - The flags are used to handle cases:
    - ptrFlag
      - 0 - If nothing was put into buffer yet
      - 1 - A char ptr is found, and allows program to search for a blank space or pipe if not in quote to turn into \0 to mark end of string.
    - quoteFlag
      - Allows immunity to all characters within the string. No character in string between quotes with stop program from setting ptrFlag to 0 except another corresponding quote.
      - 0 : Off
      - 1: Single Quote (‘)
      - 2: Double Quote (“)
  - Loads the very buffer cmd in the linked list, and ptrs cmdListTail points to it at first. Then with every insertion, cmdListTail points to tail to allow faster insertion into the linked list
- Methods void runsrc,void runsrc2, void rundest, void rundest2, void transfer:
  - Take the arguments: (int fd1[], int fd2[], char* cmd1[])
    - fd1 and fd2 are piped file descriptors
    - char* cmd1[]
  - Called by piping to handle the possibilities of 2 - 3 commands separated by 1-2 pipes, with each pipe between each command.
  - runsrc and rundest are used in the case of two commands separated by one pipe
  - runsrc2, transfer, rundest2 are called in order to pass values from one end to the next in the case of three commands separated by two pipes.
- void piping(cmdList head*)
  - Decides which case of piping would occur. Determines the number of cmds by counting the number of nodes in the linked list of cmdLists, and will decide which case to use.

handling built-in commands

- builtin.c
- builtin.h

builtin.c

- `int call_exit(int status)`
    - wrapper function for exit(0)
    - arg: int status
        - the argument for exit()
    - returns 0 after exiting
- `int call_cd(char * path)`
    - wrapper function for chdir(path)
    - arg: char * path
        - argument for chdir()
    - returns what chdir() returns
    - also handles if the path given is NULL
    - in that case, it will cd to HOME
- `int is_builtin(cmdList *c)`
    - checks if certain command (c) is a built-in command or not
    - arg: cmdList * c
        - command that user enters
    - returns -1 if not built-in command, 0 if built-in command
- `void builtin(cmdList *c)`
    - executes the built-in command
    - arg: cmdList * c
        - command that user types in
    - returns nothing
    - sets up the function pointers to `call_exit()` and `call_cd()` wrapper functions
    - populates the `struct builtin * b_fx[2]` struct that will hold all the built-in commands (their names and their function pointers)
    - for loop through all the built-in commands to see if there's a match

        - if there is, calls that built-in command's function pointer with the arguments given to the command c

builtin.h

- created `struct builtin` to hold
    1. name of the command (`char *`)
    2. pointer to the built-in function (`int(*f)()`)