Assignment – 2 SHELL

Sai Teja Reddy Moolamalla - 201564086
Siddhartha Laghuvarapu -201564122

Make instructions:
From the directory where the Makefile is present, run:

make

Running the shell:
./a.out

To delete all the object files:
make clean

File breakdown:

- main.c: Contains functions for inputting and displaying the output. Also contains implementations for some commands like cd, pwd. Manges things like semi-colon splitting, background processes and functions for other system commands.
- Makefile
- echo_implement.c: Consists of functions corresponding to the implementation of echo.
- echo_implement.h: Contains headers for echo_implement.c file.
- ls_implement.c: Consists of functions corresponding to the implementation of ls.
- ls_implement.h: Contains headers for ls_implement.c file.
- nightswatch_implement.c: Consists of functions corresponding to the implementation of nightswatch.
- nightswatch_implement.h: Contains headers for nightwatch.c file.
- pinfo_implement.c: Consists of functions corresponding to the implementation of pinfo.
- pinfo_implement.h: Contains headers for pinfo_implement.c file.
- Readme.txt : This file.

Scope of Work : All the required functionality has been implemented, all bonuses included. Appropriate error codes have been implemented wherever deemed necessary.

Implementation and functionality details:

Display:

1. Once the code is executed, the shell prompt would be displayed in the format: <username@system_name:curr_dir>
2. Some color code has been used to display the shell prompt and some messages.
3. The functionality and the output/errors displayed for all the commands have been made to resemble bash.

4. '~' corresponds to the present directory(i.e the directory from which the script is executed.)

Processing the input:

1. Getting line by line using, getline()
2. Separating the line by semicolon(), to get each separate command.
3. Processing each command using strtok(), with some DELIMITERS specified, and identifying the command, background or not etc. Also checking if two commands are separated by '&'. At this step, the command and the arguments, options are separated.
4. For each command identified,
5. Firstly, check if it is a shell builtin or not.
6. If not, execute the command using execvc, otherwise send it to the appropriate function.

cd:

1. When the function corresponding to cd is invoked, the directory is changed using chdir.
2. Cases like '~' etc. have been replaced with appropriate strings.
3. Appropriate error message will be printed if the operation fails using perror.

ls :

1. Various flags are identified, [al] in all possible combinations.
2. In case of no flags, all files in the directory are printed(excluding the hidden ones), all incase of -a.
3. In case of -l, file type,file permissions,file modified time,file size are fetched using stat.
4. When a directory is specified, files in that directory are specified.
5. In case of symlinks, link is presented in ls -l (like in bash).
6. The printing format used is to emulate 'ls -l' in bash.
7. Appropriate error messages will be printed for "file not found", wrong flags etc.

pwd:

Current working directory will be printed.(fetched using getcwd()).

echo:

1. The input is passed through a 'state-machine' type structure, so as to efficiently handle all the possible input types.
2. All the cases when characters such as "", '', spacing, multiple occurrences of quotes etc. have been handled properly including cases when newline is printed after opening quotes.
3. Memory is dynamically allocated depending upon the size of the command.
4. The format input and output would emulate bash.

pinfo:

1. Necessary information is fetched from '/proc/pid/status' and executable path from 'proc/pid/exe'
2. When no pid is specified, pid of the current process is brought from getpid().

3. Appropriate error message printed when the pid could not be identified or false string has been specified.

nightswatch:

1. Options(-n) and commands are identified firstly, using getopt, and the corresponding value, if options are present.(line -n 2).
2. If time is specified, it is used. Default value is 2 s.
3. The command output is printed and updated on a curses screen. (Emulating the watch command). The screen closes when 'q' is pressed.
4. In case of "Interrupt", appropriate information is fetched and updated from /proc/interrupts file.
5. In case of "Dirty", the information is brought from /proc/meminfo file.
6. Proper error messages have been printed in case the input is not in accord with the format.

background/foreground:

1. Identified with '&' as described above.
2. -fork() command is used and the process is run in background.
3. Also, once the process has exited(using waitpid()), an exit message is printed after the execution of the next command or when a new line is printed(like in bash).