

Tmlz5d
Tim M. Lael
cs2750-e01
project 4

Compilation/Execution is to take place on Delmar

mysh is a minimal Unix shell written in C.

It consists of four files, a README and this report in two formats(.pdf and .docx).

The core program files are as follows:

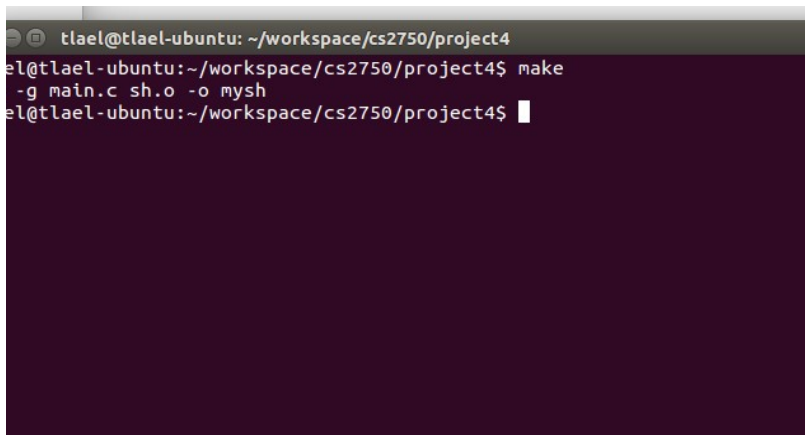
main.c – includes a simple loop and signal handlers to catch ^C.

Makefile – makefile for building project

sh.c – shell code

sh.h – shell header

Running make:



```
tlael@tlael-ubuntu: ~/workspace/cs2750/project4
tlael@tlael-ubuntu:~/workspace/cs2750/project4$ make
-g main.c sh.o -o mysh
tlael@tlael-ubuntu:~/workspace/cs2750/project4$
```

In main.c, the loop passes the prompt and command to two different functions that process and execute the terminal input. Main also implements the PS1 handling. If PS1 is null (unset) the default prompt string (“>>>”) is passed, otherwise PS1 is passed. I had to add a line to my .bashrc file to export the environment variable and capture it in the code using getenv(“PS1”). Prompt shown below:

```
project4
tlael@tlael-ubuntu: ~/workspace/cs2750/project4
tlael@tlael-ubuntu:~/workspace/cs2750/project4$ make
gcc -g main.c sh.o -o mysh
tlael@tlael-ubuntu:~/workspace/cs2750/project4$ ./mysh
\[\e]0;\u@\h: \w\a\]${debian_chroot:+($debian_chroot)}\u@\h:\w\$ 
tlael@tlael-ubuntu: ~/workspace/cs2750/project4
tlael@tlael-ubuntu:~$ cd work*/cs*/4*
tlael@tlael-ubuntu:~/workspace/cs2750/project4$ unset PS1
./mysh
>>>
```

One component

that I was unable to implement was the PATH/MYPATH search using get_path.c[h].

As you can see, both PATH and MYPATH are populated on my test machine, but I could not build the functionality in while maintaining the functionality of the rest of the program. PATH/MYPATH shown:

```
tlael@tlael-ubuntu:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
tlael@tlael-ubuntu:~$ echo $MYPATH
/home/tlael/workspace/cs2750/project3:/home/tlael/workspace/cs2750/project2:/home/tlael/bin:/home/tlael/workspace/cs2750/project3:/home/tlael/workspace/cs2750/project2:/home/tlael/bin
tlael@tlael-ubuntu:~$
```

In sh.c, I implemented a function (getcmd()) to get terminal input. Getcmd() is actually passed characters from a buffer and the prompt to be printed from main. Getcmd() also handles Ctrl+D by analyzing input for an EOF signal. The command is then passed to runcmd() to process for a built-in command (the exit command which is case insensitive). After analyzed for the built-in exit command, the command is forked and executed. Execution implements the standard PATH for executables since MYPATH functionality was omitted. Once a file is found in the cwd or if it exists in PATH, it is executed as shown below:

```

tlael@tlael-ubuntu: ~
./mysh
>>> ls -la
total 52
drwxrwxr-x 2 tlael tlael 4096 May  2 12:56 .
drwxrwxr-x 7 tlael tlael 4096 May  2 12:46 ..
-rw-rw-rw- 1 tlael tlael  790 May  2 12:54 main.c
-rw-rw-rw- 1 tlael tlael  118 May  2 09:32 Makefile
-rwxrwxr-x 1 tlael tlael 17097 May  2 12:56 mysh
-rw-rw-r-- 1 tlael tlael    0 May  2 12:48 README
-rw-rw-rw- 1 tlael tlael 3376 May  2 12:16 sh.c
-rw-rw-rw- 1 tlael tlael  202 May  2 12:14 sh.h
-rw-rw-r-- 1 tlael tlael 6344 May  2 12:52 sh.o
>>> ps -la
 F S  UID      PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S   1000    8621   8511  0  80   0 -  1083 wait  pts/1        00:00:00 mysh
0 R   1000    8623   8621  0  80   0 -  3552 -      pts/1        00:00:00 ps
>>> pwd
/home/tlael/workspace/cs2750/project4
>>>
Ctrl+D encountered... Exiting

```

The above image shows

several system commands being run (ls -la, ps -la, pwd). After each command is parsed and executed, the prompt is displayed again for further input. Also, note the Ctrl+D to exit the shell. This exits to a blank prompt since PS1 is null in my parent shell (also signaled by the “>>> “ prompt in mysh).

Also shown below is an example of other exit/non-exit requirements:

Notice above that I start several instances of mysh and provide

```

tlael@tlael-ubuntu: ~
./mysh
>>> ./mysh
>>> ./mysh
>>> ./mysh
>>> Exit
Exit encountered... Exiting
>>> EXIT
Exit encountered... Exiting
>>> exit
Exit encountered... Exiting
>>> ^C
Cannot be terminated using Ctrl+C
>>>
Ctrl+D encountered... Exiting

```

examples of different exit behaviors. First is three examples of exit to show its case insensitive behavior. Next is a Ctrl+C to demonstrate that Ctrl+C does not quit mysh. Last is Ctrl+D to show that it does terminate the last instance.

Conclusions:

This project has given me a better understanding of signal handling since we were required to work with both, standard (SIGINT) and non-standard (Ctrl+D) signal handling. I also spent a great deal of time trying to implement the `get_path` functionality and, through that process, learned a lot about some of the lesser known C functions (even though a final implementation did not include much of my trial and error). One of coolest things I implemented was the case insensitive exit string comparison. While `strcasecmp()` may be better known by some, it is new to me and works well. I will definitely be using this in the future!