Contents

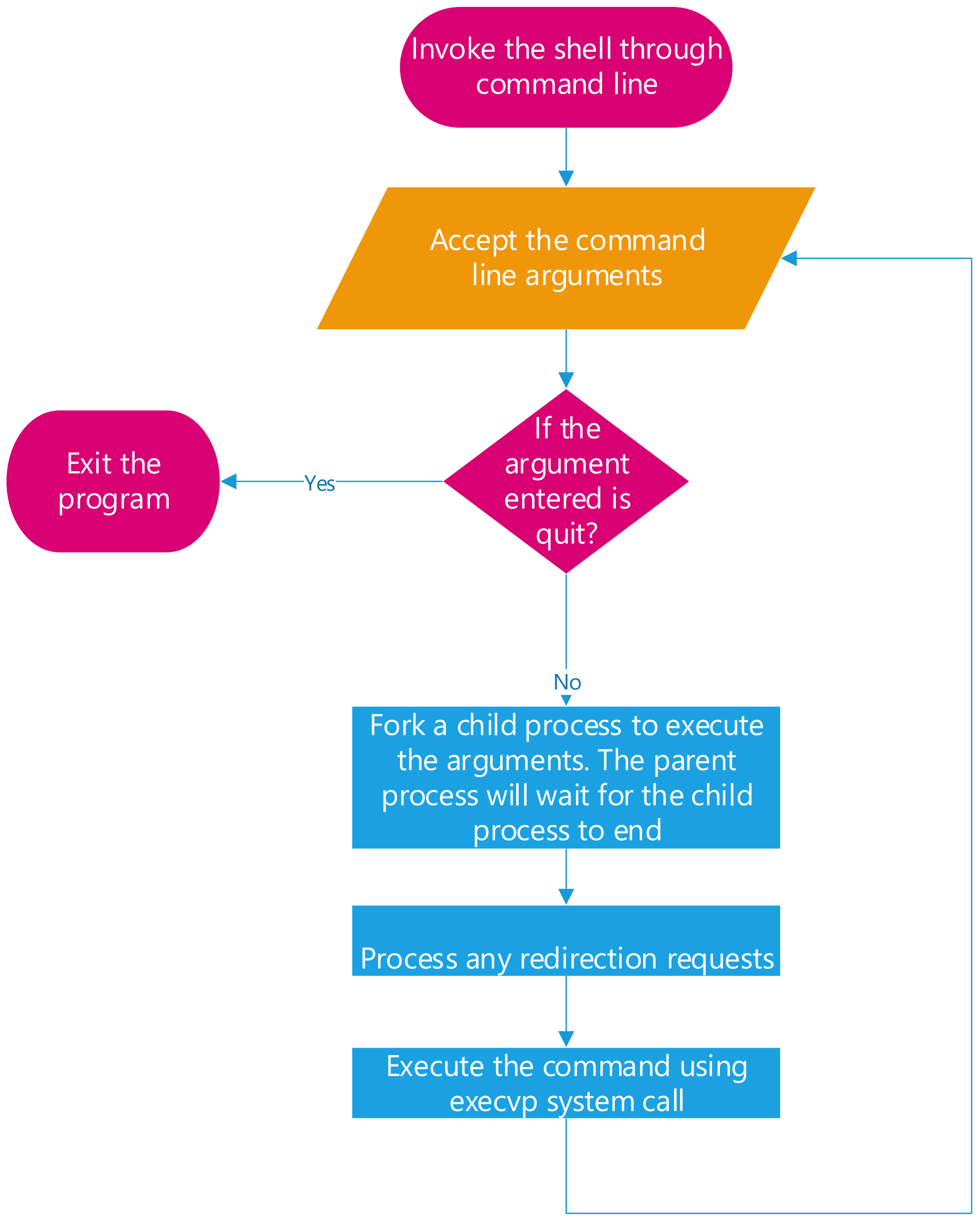
1. Introduction

As part of this programming assignment, a shell is implemented using a low-level language. The shell prompt can accept any Unix / Linux command line and invoke a POSIX system call to execute the command. The shell program is also capable of flagging and displaying errors if the command entered by the user is erroneous.

The report documented here captures the different aspects of the programming assignment in sections. The first section walks through the design of the program. The steps required to build, run, and terminate the programs are covered in the next few sections. Scope and conclusion are covered in the last 2 sections.

2. Design

A high-level design of this program is explained in the below flow chart. When the program is triggered via the command line, a shell prompt appears on the screen. Once the user enters a command, the program checks if the request is to terminate the program via the "quit" command. If so, the program terminates immediately. For all other requests from the user, a child process is forked and the program first invokes a re-direction handler routine to process any I/O file requests. Then it invokes a system call to execute the command requested by the user. Finally, after the child process is terminated the control returns to the parent process and waits for the next user request.


Invoke the shell through command line
Accept the command line arguments
If the argument entered is quit?
Yes
Exit the program
No
Fork a child process to execute the arguments. The parent process will wait for the child process to end
Process any redirection requests
Execute the command using execvp system call

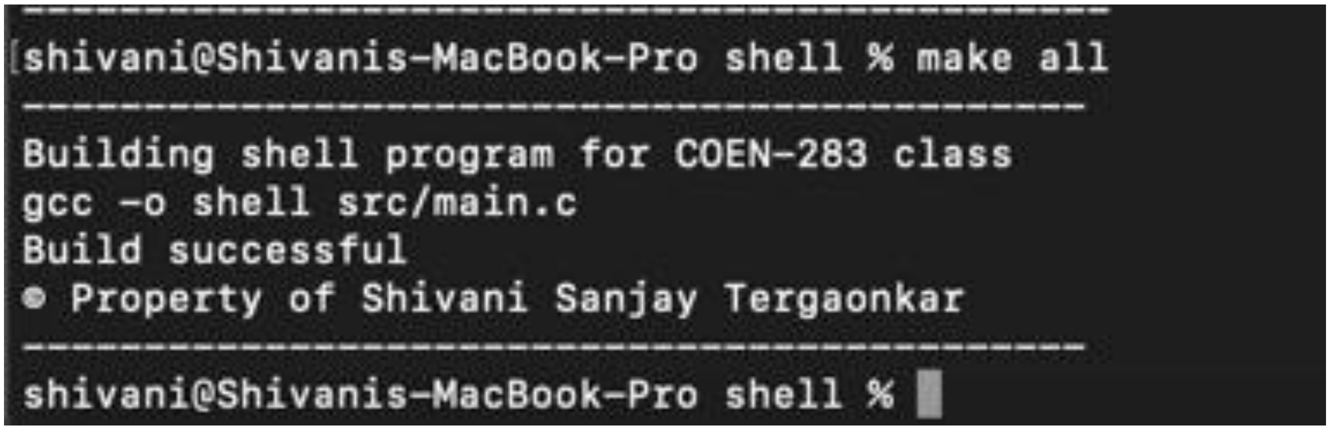
3. Files included

File name	Description
src/main.c	Shell program implementation
Makefile	Makefile to build the program
COEN-283 Programming Assignment #1.pdf	Design document for the programming assignment

4. Building the program

A Makefile is provided as part of the software package. To build the program, use the below command line.

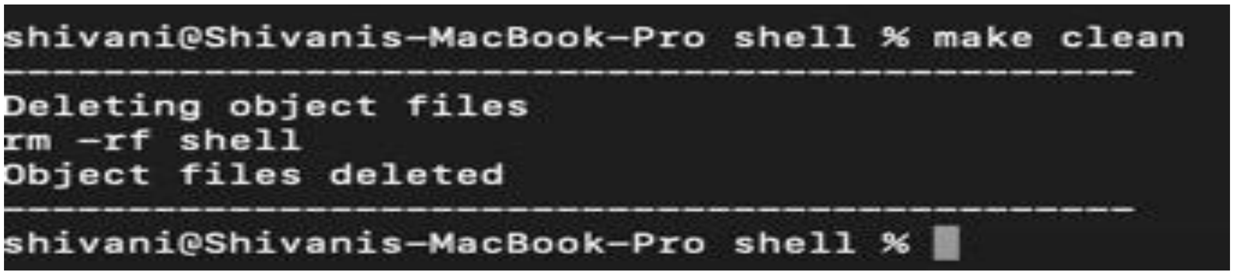
```
make all
```



```
-------------------------------------------------------------
shivani@Shivanis-MacBook-Pro shell % make all
-------------------------------------------------------------
Building shell program for COEN-283 class
gcc -o shell src/main.c
Build successful
© Property of Shivani Sanjay Tergaonkar
-------------------------------------------------------------
shivani@Shivanis-MacBook-Pro shell %
```

In the case of a clean build, after an incremental change on the source code, it may be required to clean up the previously built object. In such cases, use the below command before the build.

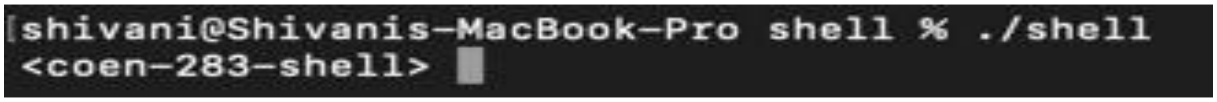
```
make clean
```



```
shivani@Shivanis-MacBook-Pro shell % make clean
-------------------------------------------------------------
Deleting object files
rm -rf shell
Object files deleted
-------------------------------------------------------------
shivani@Shivanis-MacBook-Pro shell %
```

5. Running the program

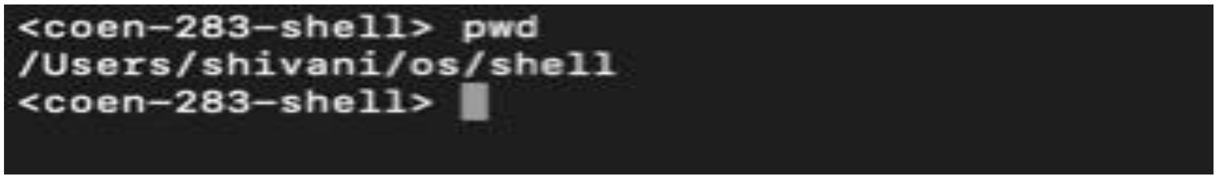
Once the program is built, the executable '`shell`' can be invoked directly from the command line. A command prompt that continuously runs until terminated, is then presented on the screen.



```
shivani@Shivanis-MacBook-Pro shell % ./shell
<coen-283-shell>
```

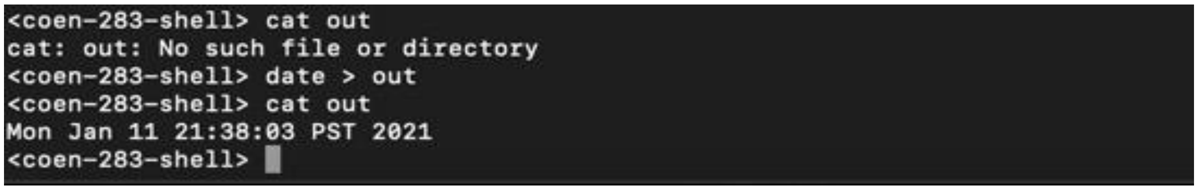
6. Verifying the program

This section walks through the different command inputs from the user. The first screenshot highlights a simple "`pwd`" Unix command input by the user.



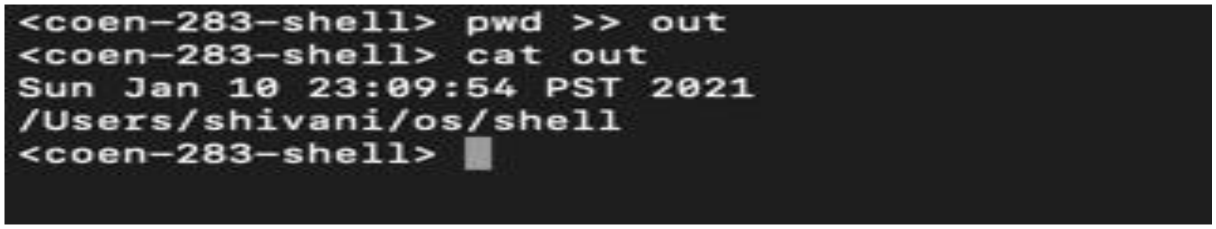
```
<coen-283-shell> pwd
/Users/shivani/os/shell
<coen-283-shell> 
```

The shell also has the feature of file re-directions as seen in the below screenshots. In the first screenshot, a single re-direction creates a new file and writes the output of the input command into the file.



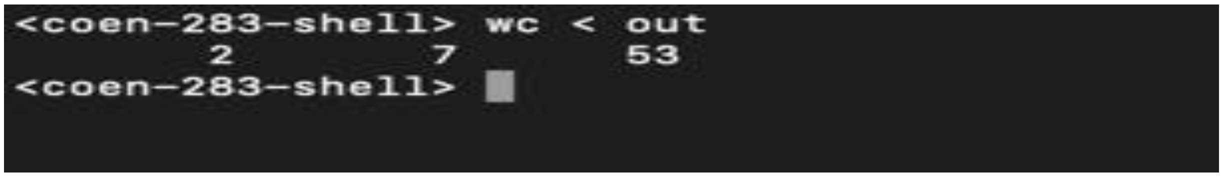
```
<coen-283-shell> cat out
cat: out: No such file or directory
<coen-283-shell> date > out
<coen-283-shell> cat out
Mon Jan 11 21:38:03 PST 2021
<coen-283-shell> 
```

In the next screenshot, when the user requests to append the output of the command to the existing file, the output file is opened in append mode, and the results of the command are written from the next line in the file specified.



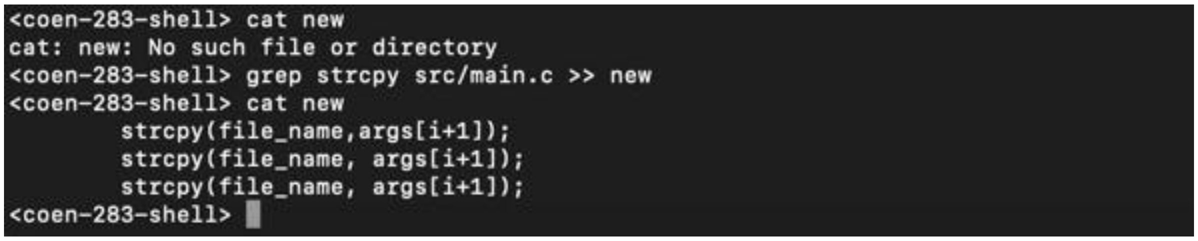
```
<coen-283-shell> pwd >> out
<coen-283-shell> cat out
Sun Jan 10 23:09:54 PST 2021
/Users/shivani/os/shell
<coen-283-shell> 
```

The below screenshot highlights the input re-direction. The word count of the input file is output to the terminal.



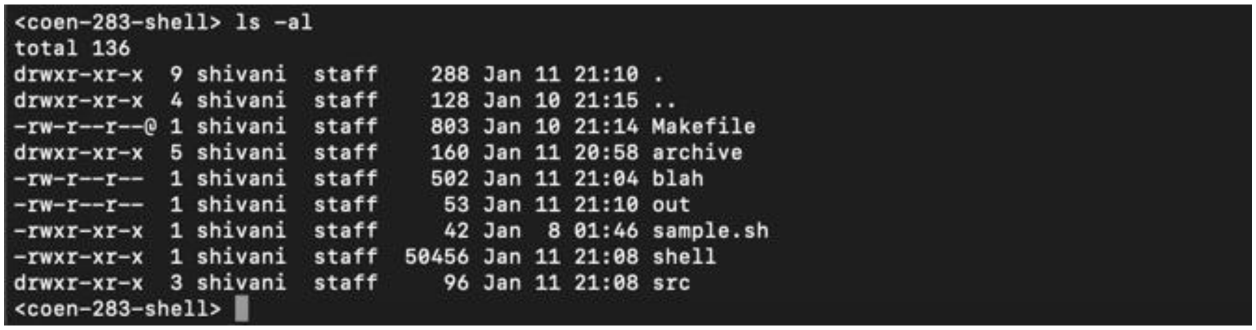
```
<coen-283-shell> wc < out
       2       7      53
<coen-283-shell> 
```

The shell program also handles scenarios where an append operation is requested by the user on a file that doesn't exist. In this case, a new file is created, and the result of the input command is written to the file.



```
<coen-283-shell> cat new
cat: new: No such file or directory
<coen-283-shell> grep strcpy src/main.c >> new
<coen-283-shell> cat new
        strcpy(file_name,args[i+1]);
        strcpy(file_name, args[i+1]);
        strcpy(file_name, args[i+1]);
<coen-283-shell>
```

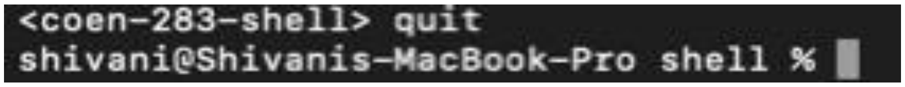
The shell program also supports multi-parameter commands as shown in the below example.



```
<coen-283-shell> ls -al
total 136
drwxr-xr-x  9 shivani  staff    288 Jan 11 21:10 .
drwxr-xr-x  4 shivani  staff    128 Jan 10 21:15 ..
-rw-r--r--@ 1 shivani  staff    803 Jan 10 21:14 Makefile
drwxr-xr-x  5 shivani  staff    160 Jan 11 20:58 archive
-rw-r--r--  1 shivani  staff    502 Jan 11 21:04 blah
-rw-r--r--  1 shivani  staff     53 Jan 11 21:10 out
-rwxr-xr-x  1 shivani  staff     42 Jan  8 01:46 sample.sh
-rwxr-xr-x  1 shivani  staff  50456 Jan 11 21:08 shell
drwxr-xr-x  3 shivani  staff     96 Jan 11 21:08 src
<coen-283-shell>
```

7. Terminating the program

To terminate the program, the user can type in "`quit`" and the program exits.



```
<coen-283-shell> quit
shivani@Shivanis-MacBook-Pro shell %
```

8. Scope for future work

The program can be enhanced to add new options such as background process and pipes.

9. Conclusion

This exercise gives a good insight into low-level system calls carried out by an operating system. This also introduces the process and inter-process communication.