Project 3: Linux Shell

Spring 2018

CS 4350

SUBMITTED BY:

Dillon Rowan - dhr14@txstate.edu

Jacob Benavente - jab422@txstate.edu

Jae Lee - jml209@txstate.edu

Jake Mata - jam495@txstate.edu

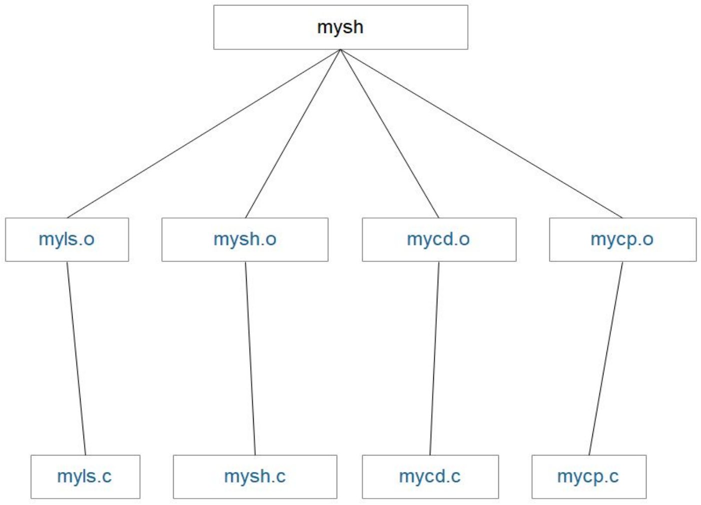
Ryan Howard - reh101@txstate.edu

Section I (Introduction):

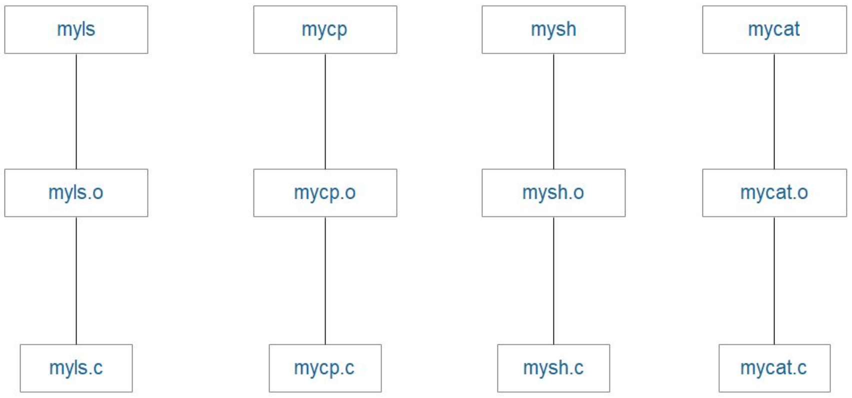
Our group met several times outside class to delegate different tasks for the project. All group members worked on seperate functions and then we combined them to a create our finish project. Ryan Howard implemented the pwd command and the mycat command. Jake Mata implemented the myls command. Jacob Benavente implemented the mycd command. Jae Lee implemented the entire shell. Dillon Rowan implemented the mycp command. All of our group members shared a Google doc and worked on the report together as a team.

Section II (Task I):

1) Dependency graph of mysh:



Dependency graph of each individual commands and the shell (pwd was included with the shell):



2)

Shell program makefile:

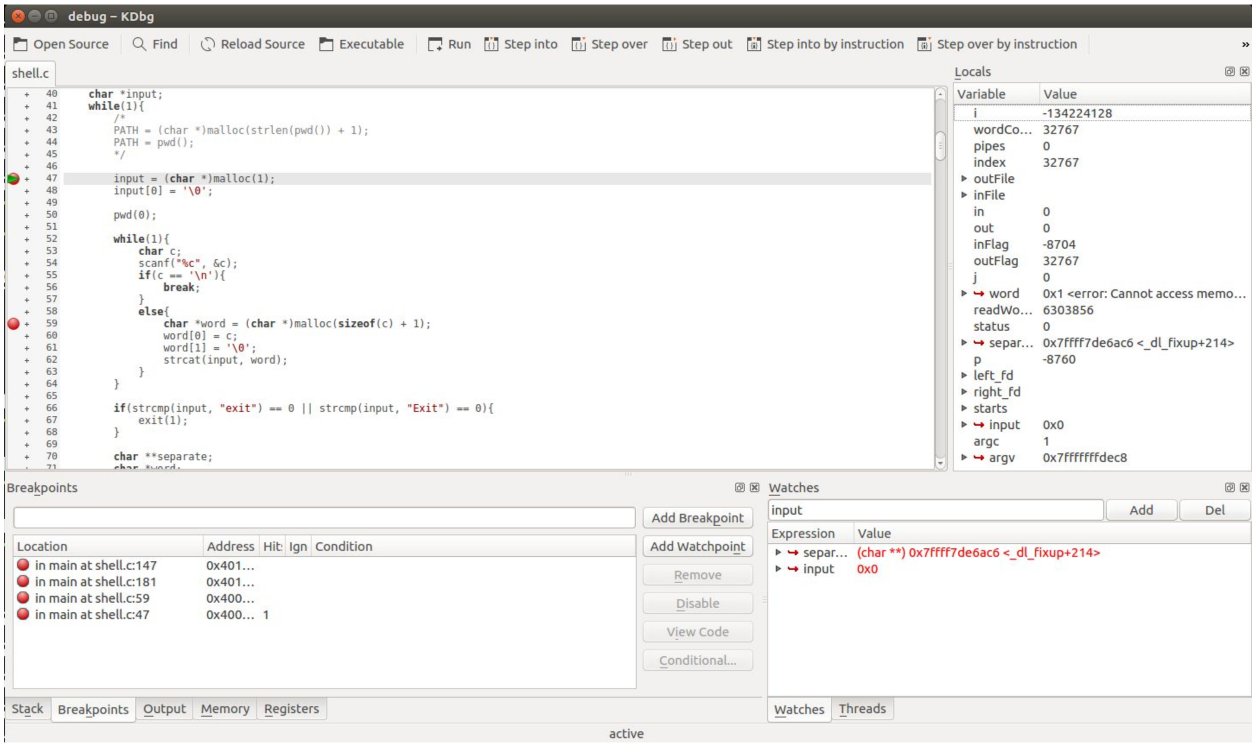
```
CFILES = mysh.c myls.c mycat.c mycp.c
OFILES = mysh.o myls.o mycat.o mycp.o
NAME = shell

all: $(NAME)
$(NAME): $(OFILES)
	gcc -o $(NAME) $(OFILES)
mysh.o: mysh.c
	gcc -c mysh.c
myls.o: myls.c
	gcc -c myls.c
mycp.o: mycp.c
	gcc -c mycp.c
mycat.o: mycat.c
	gcc -c mycat.c
clean:
	rm $(NAME) *.0
```

Mycat command makefile:

```
output: mycat.o
   gcc mycat.o -o mycat
mycat.o: mycat.c
   gcc -c mycat.c
clean:
   rm *.output
```

3)



```
    char *input;
    while(1){
        /*
        PATH = (char *)malloc(strlen(pwd()) + 1);
        PATH = pwd();
        */

        input = (char *)malloc(1);
        input[0] = '\0';

        pwd(0);

        while(1){
            char c;
            scanf("%c", &c);
            if(c == '\n'){
                break;
            }
            else{
                char *word = (char *)malloc(sizeof(c) + 1);
                word[0] = c;
                word[1] = '\0';
                strcat(input, word);
            }
        }

        if(strcmp(input, "exit") == 0 || strcmp(input, "Exit") == 0){
            exit(1);
        }

        char **separate;
```

Variable	Value
i	-134224128
wordCo...	32767
pipes	0
index	32767
outFile	
inFile	
in	0
out	0
inFlag	-8704
outFlag	32767
j	0
word	0x1 <error: Cannot access memo...
readWo...	6303856
status	0
separ...	0x7ffff7de6ac6 <_dl_fixup+214>
p	-8760
left_fd	
right_fd	
starts	
input	0x0
argc	1
argv	0x7fffffffdec8

Location	Address	Hit	Ign	Condition
in main at shell.c:147	0x401...			
in main at shell.c:181	0x401...			
in main at shell.c:59	0x400...			
in main at shell.c:47	0x400...	1		

input

Expression	Value
separ...	(char **) 0x7ffff7de6ac6 <_dl_fixup+214>
input	0x0

Section III (Task II):

1) The “myls” command supports the arguments of the “-l” option by itself which displays all the information “ls -l” in bash would display, the “-l” option accompanied with a text file that shows only the “-l” information for the specified file, and also accepts a directory as an argument by itself that displays all the contents of the specified directory. The “myls” command first parses the arguments to determine the appropriate action to take. If there is only one argument, there’s conditional if statement that accepts either “-l” or a directory name. Either case, it takes the single argument, appends a null character into a collected char pointer array, then passes the argument to the execvp() function to perform the action. If the user passes in two arguments following the command “myls”, the command accepts only “-l” and a file name. Same as before, the arguments are collected into a char pointer array with a null character, and then processes the argument into the execvp() function to to display the “-l” on the specified file.

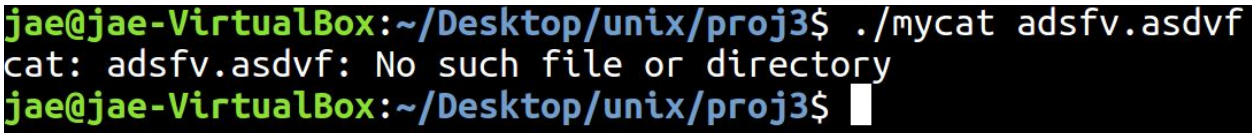
The “mycat” command takes a file as a command line argument then displays the contents to standard output.

The “mycp” command supports additional arguments such as -R. The command mycp first mallocs a char** variable called new_argv to be the same size of argv. The contents of argv are then copied over to new_argv. The characters in new_argv are then shifted left twice starting at new_argv[1]. A null character is then added to the end of this array. After this is done execvp is then executed via the following: execvp(new_argv[0], new_argv); Perror(new_argv[1]) is used to produce a message on standard error to describe the last error encountered during a call.

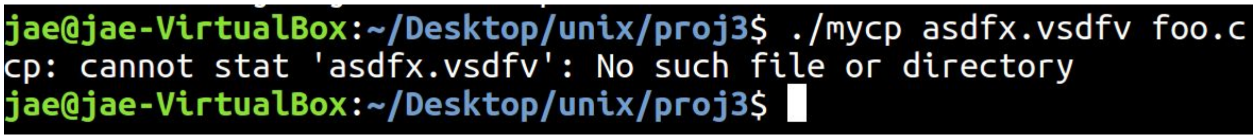
2) One issue was when implementing the “myls” command was determining if the entered directory was a directory that existed within the current directory. To ensure this a conditional if statement was used with the macro function “S_ISDIR(path_stat.st_mode)” and if true, would then perform the execvp function, if false would display an error message stating the “directory does not exist.”

An issue with implementing mycp was making sure the correct argument of the command line was being modified and that the character shifting did not go out of bound. If a stack overflow occurred a -g argument was used while compiling, so that gdb debugger could be used to detect which line was causing this.

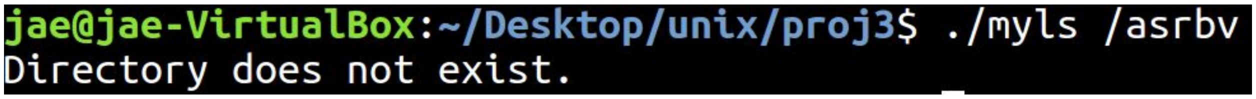
3)



```
jae@jae-VirtualBox:~/Desktop/unix/proj3$ ./mycat adsfv.asdvf
cat: adsfv.asdvf: No such file or directory
jae@jae-VirtualBox:~/Desktop/unix/proj3$
```



```
jae@jae-VirtualBox:~/Desktop/unix/proj3$ ./mycp asdfx.vsdfv foo.c
cp: cannot stat 'asdfx.vsdfv': No such file or directory
jae@jae-VirtualBox:~/Desktop/unix/proj3$
```

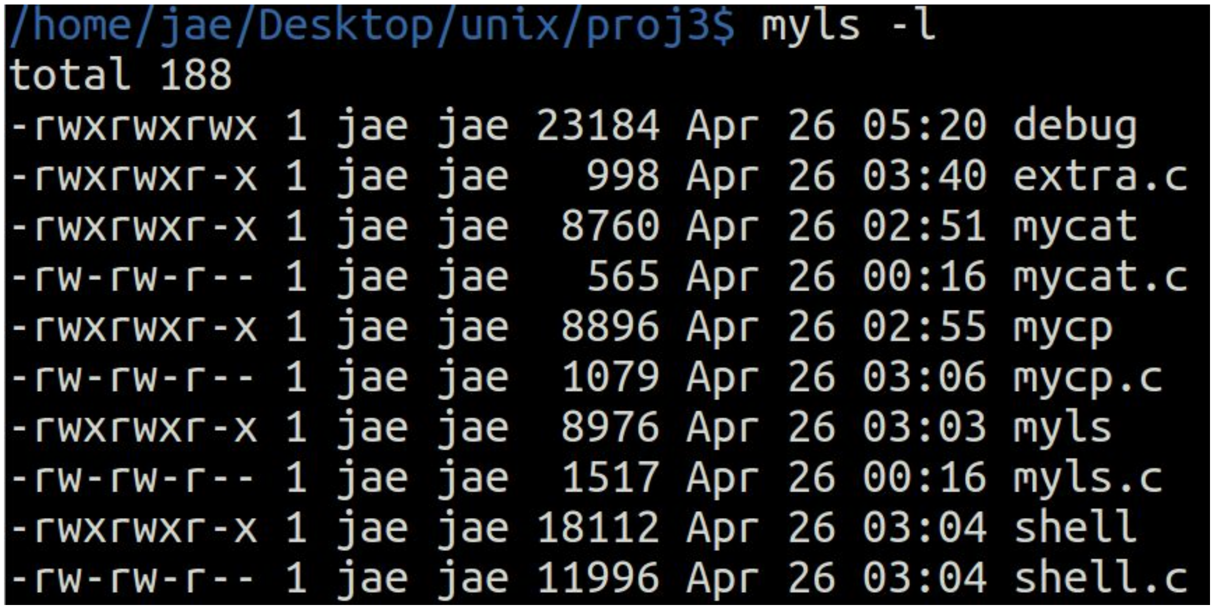


```
jae@jae-VirtualBox:~/Desktop/unix/proj3$ ./myls /asrbv
Directory does not exist.
```

Section IV(Task III):

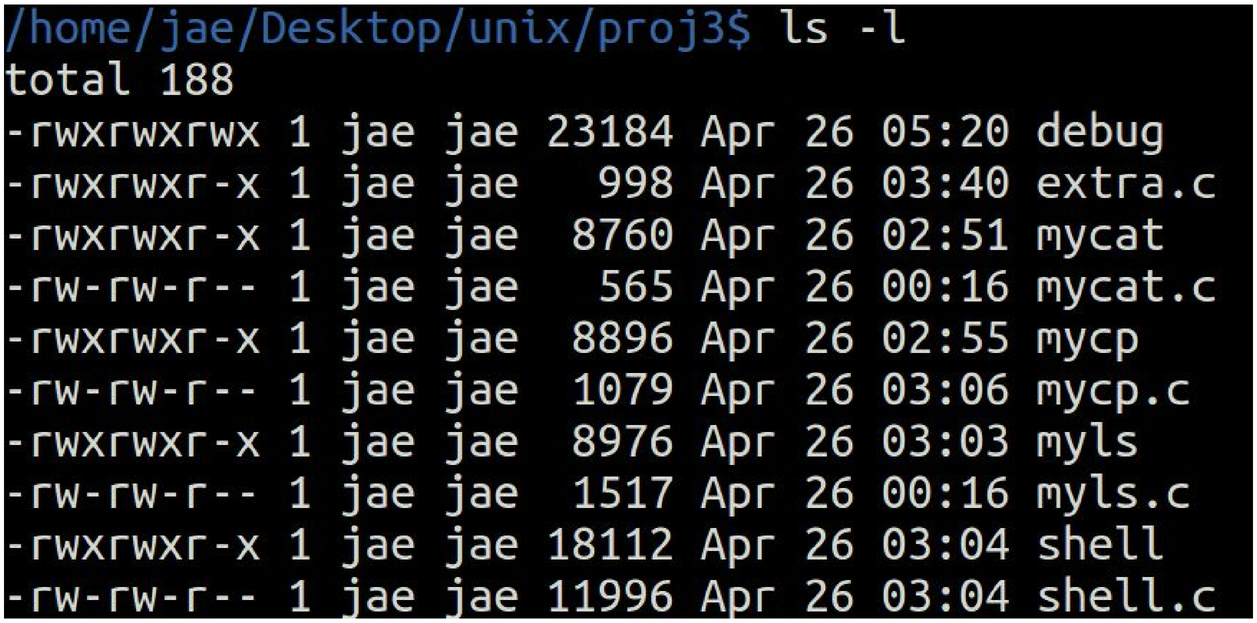
1) The shell program first interprets user input by adding up each character into one string then creating an array of strings that separates each word in the input by a space. Then it searches through the array of strings to find either a pipe symbol or a redirection symbol. If a redirection is found the program flags that it happened and changes a 2 int array to be equal to the word after the redirection. If a pipe is found, then the number of pipes and the placement of the pipes are saved. Then I loop through each amount of pipes, if any, and create a child. If the child is the first one created then I dup the output to a variable for the remaining child, if it is the middle children (not first and not last) then it receives from the output of the previous and writes to the next remaining childs, and the last child

receives the output that went through all other pipes as input, and if redirection flags were in the input, then it got the input for the child process from the input redirection via open, and/or wrote the output to the redirection by using a creat to make a write file. If no pipes were detected, then a few keywords like cd, pwd, myls, mycat, mycp, were specially checked because cd and pwd were implemented in the shell and the other "my" files were not in the PATH, but in the same repository, so I had to run them using the "./" prefix. Then I caught all other possible inputs and sent it to bash, so if it was a legitimate command, it would run, but if it wasn't it would show appropriate error messages.

2) 

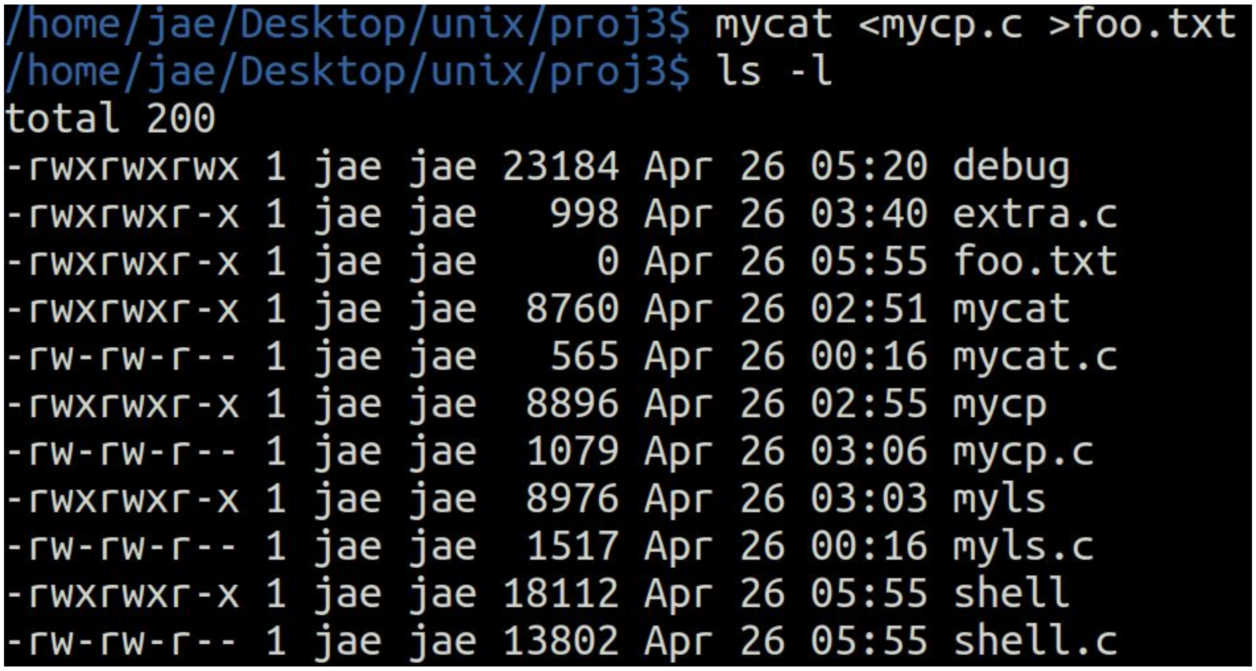
```
/home/jae/Desktop/unix/proj3$ myls -l
total 188
-rwxrwxrwx 1 jae jae 23184 Apr 26 05:20 debug
-rwxrwxr-x 1 jae jae   998 Apr 26 03:40 extra.c
-rwxrwxr-x 1 jae jae  8760 Apr 26 02:51 mycat
-rw-rw-r-- 1 jae jae   565 Apr 26 00:16 mycat.c
-rwxrwxr-x 1 jae jae  8896 Apr 26 02:55 mycp
-rw-rw-r-- 1 jae jae  1079 Apr 26 03:06 mycp.c
-rwxrwxr-x 1 jae jae  8976 Apr 26 03:03 myls
-rw-rw-r-- 1 jae jae  1517 Apr 26 00:16 myls.c
-rwxrwxr-x 1 jae jae 18112 Apr 26 03:04 shell
-rw-rw-r-- 1 jae jae 11996 Apr 26 03:04 shell.c
```

3)



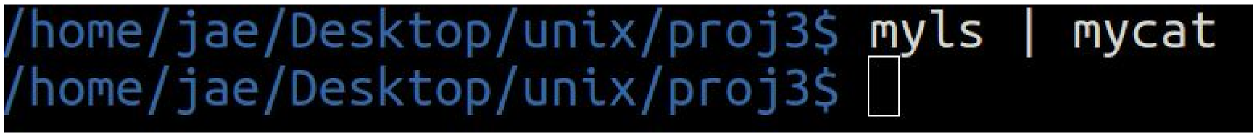
```
/home/jae/Desktop/unix/proj3$ ls -l
total 188
-rwxrwxrwx 1 jae jae 23184 Apr 26 05:20 debug
-rwxrwxr-x 1 jae jae   998 Apr 26 03:40 extra.c
-rwxrwxr-x 1 jae jae  8760 Apr 26 02:51 mycat
-rw-rw-r-- 1 jae jae   565 Apr 26 00:16 mycat.c
-rwxrwxr-x 1 jae jae  8896 Apr 26 02:55 mycp
-rw-rw-r-- 1 jae jae  1079 Apr 26 03:06 mycp.c
-rwxrwxr-x 1 jae jae  8976 Apr 26 03:03 myls
-rw-rw-r-- 1 jae jae  1517 Apr 26 00:16 myls.c
-rwxrwxr-x 1 jae jae 18112 Apr 26 03:04 shell
-rw-rw-r-- 1 jae jae 11996 Apr 26 03:04 shell.c
```

4)



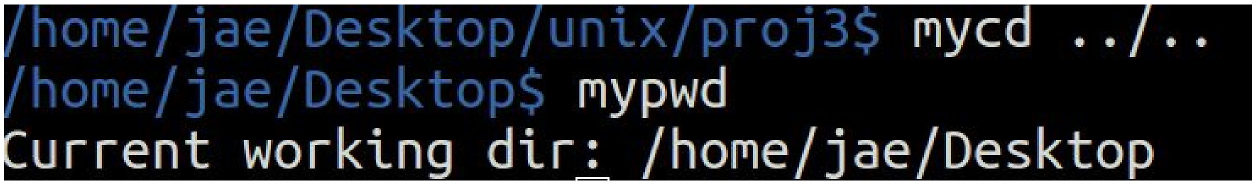
```
/home/jae/Desktop/unix/proj3$ mycat <mycp.c >foo.txt
/home/jae/Desktop/unix/proj3$ ls -l
total 200
-rwxrwxrwx 1 jae jae 23184 Apr 26 05:20 debug
-rwxrwxr-x 1 jae jae   998 Apr 26 03:40 extra.c
-rwxrwxr-x 1 jae jae     0 Apr 26 05:55 foo.txt
-rwxrwxr-x 1 jae jae  8760 Apr 26 02:51 mycat
-rw-rw-r-- 1 jae jae   565 Apr 26 00:16 mycat.c
-rwxrwxr-x 1 jae jae  8896 Apr 26 02:55 mycp
-rw-rw-r-- 1 jae jae  1079 Apr 26 03:06 mycp.c
-rwxrwxr-x 1 jae jae  8976 Apr 26 03:03 myls
-rw-rw-r-- 1 jae jae  1517 Apr 26 00:16 myls.c
-rwxrwxr-x 1 jae jae 18112 Apr 26 05:55 shell
-rw-rw-r-- 1 jae jae 13802 Apr 26 05:55 shell.c
```

5)



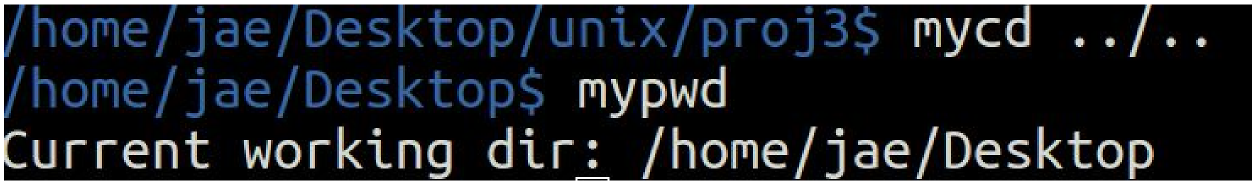
```
/home/jae/Desktop/unix/proj3$ myls | mycat
/home/jae/Desktop/unix/proj3$ 
```

6)



```
/home/jae/Desktop/unix/proj3$ mycd ../..
/home/jae/Desktop$ mypwd
Current working dir: /home/jae/Desktop
```

7)



```
/home/jae/Desktop/unix/proj3$ mycd ../..
/home/jae/Desktop$ mypwd
Current working dir: /home/jae/Desktop
```