2022-OS-HW1 : Simple MyShell

Dankook Univ.
Mobile Systems Engineering
32191818 Jueun Park
Freedays: 5

1. Project Introduction

When using a computer, the Operating System (OS) is a basic background element. The OS provides system services. For hardware features such as I/O or memory allocation, the OS acts as a mediator between processes and hardware. The user calls the kernel from the OS through the system call. It does the most basic functions. This project is to make my small 'Shell'. Shell is a small program that allows users to interact directly with the operating system. I take a command in this background and go back within a process called Shell. At this time, use the fork function to set the shell as a parent process and create a child process to execute the command instead of the child process. A fork is a system call that creates another user process.

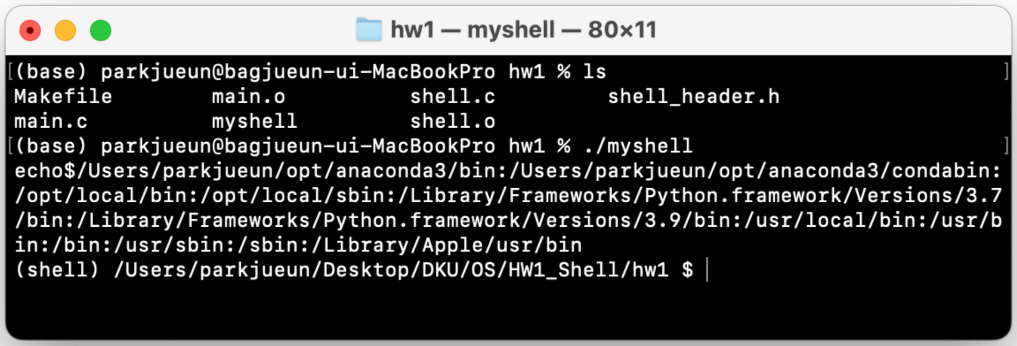
2. Instruction & Result

(1) main.c

The main function calls shell().

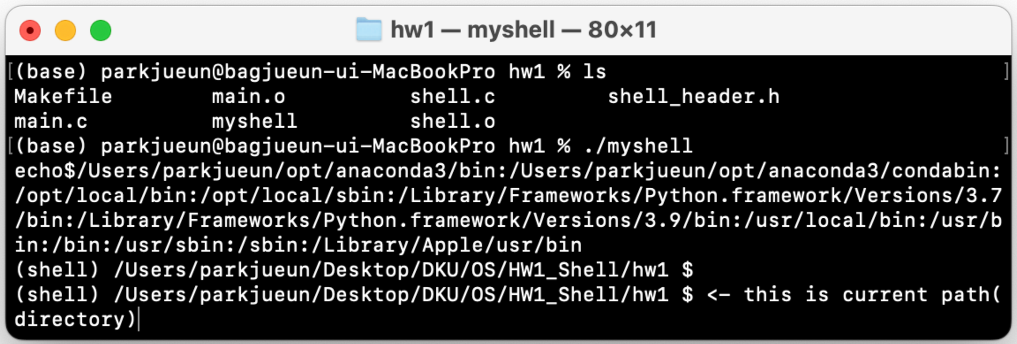
(2) shell.c

Gets the path using the getenv() function. When the Shell runs, this path appears. When the shell runs, echo$path is the first output.



```
(base) parkjueun@bagjueun-ui-MacBookPro hw1 % ls
Makefile        main.o          shell.c         shell_header.h
main.c          myshell         shell.o
(base) parkjueun@bagjueun-ui-MacBookPro hw1 % ./myshell
echo$/Users/parkjueun/opt/anaconda3/bin:/Users/parkjueun/opt/anaconda3/condabin:
/opt/local/bin:/opt/local/sbin:/Library/Frameworks/Python.framework/Versions/3.7
/bin:/Library/Frameworks/Python.framework/Versions/3.9/bin:/usr/local/bin:/usr/b
in:/bin:/usr/sbin:/sbin:/Library/Apple/usr/bin
(shell) /Users/parkjueun/Desktop/DKU/OS/HW1_Shell/hw1 $
```

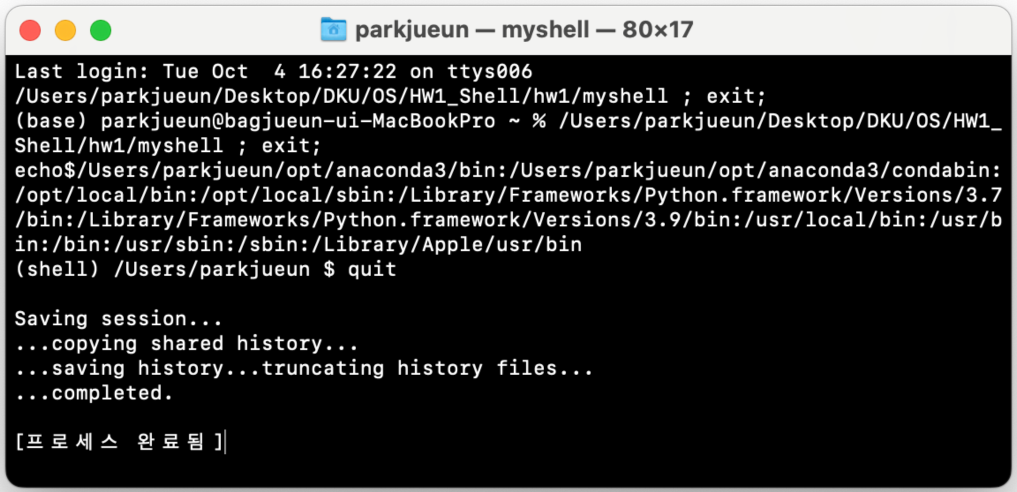
Let's look at Loop. Call the getcwd() function to find out which directory it is currently located in, and save the value to the variable pwd. Now, just like a normal terminal, let's continue to output the current path(directory).



```
[(base) parkjueun@bagjueun-ui-MacBookPro hw1 % ls                              ]
Makefile        main.o          shell.c         shell_header.h
main.c          myshell         shell.o
[(base) parkjueun@bagjueun-ui-MacBookPro hw1 % ./myshell                       ]
echo$/Users/parkjueun/opt/anaconda3/bin:/Users/parkjueun/opt/anaconda3/condabin:
/opt/local/bin:/opt/local/sbin:/Library/Frameworks/Python.framework/Versions/3.7
/bin:/Library/Frameworks/Python.framework/Versions/3.9/bin:/usr/local/bin:/usr/b
in:/bin:/usr/sbin:/sbin:/Library/Apple/usr/bin
(shell) /Users/parkjueun/Desktop/DKU/OS/HW1_Shell/hw1 $
(shell) /Users/parkjueun/Desktop/DKU/OS/HW1_Shell/hw1 $ <- this is current path(
directory)
```

The string is gotten using the fgets() function so that it can also receive white space. At this point, to save the command received by fgets() as an array and erase the newline('\n'), obtain the size of the array as a strlen() function, then enter null('\0' at the end. Then, call the strtok() function to break the input in spaces. (This is tokenized process.) Saves the string after the space in arg1.
** getenv : Obtain value of environment variable or register/change new environment variable.

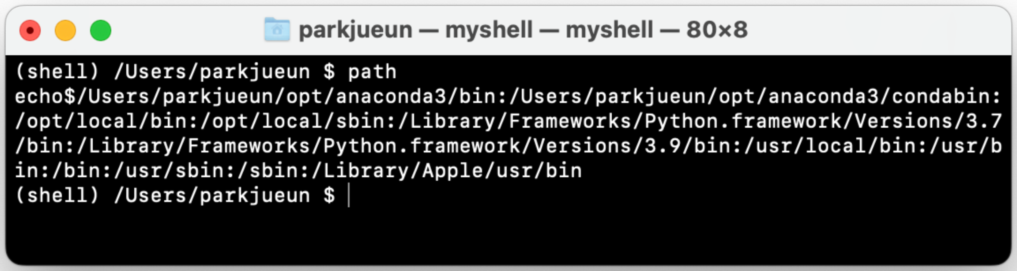
Use the strcmp() function for string comparison.
(2)-1. quit
If the input string is 'quit', exit the process.



```
Last login: Tue Oct  4 16:27:22 on ttys006
/Users/parkjueun/Desktop/DKU/OS/HW1_Shell/hw1/myshell ; exit;
(base) parkjueun@bagjueun-ui-MacBookPro ~ % /Users/parkjueun/Desktop/DKU/OS/HW1_
Shell/hw1/myshell ; exit;
echo$/Users/parkjueun/opt/anaconda3/bin:/Users/parkjueun/opt/anaconda3/condabin:
/opt/local/bin:/opt/local/sbin:/Library/Frameworks/Python.framework/Versions/3.7
/bin:/Library/Frameworks/Python.framework/Versions/3.9/bin:/usr/local/bin:/usr/b
in:/bin:/usr/sbin:/sbin:/Library/Apple/usr/bin
(shell) /Users/parkjueun $ quit

Saving session...
...copying shared history...
...saving history...truncating history files...
...completed.

[프로세스 완료됨]
```

(2)-2. path

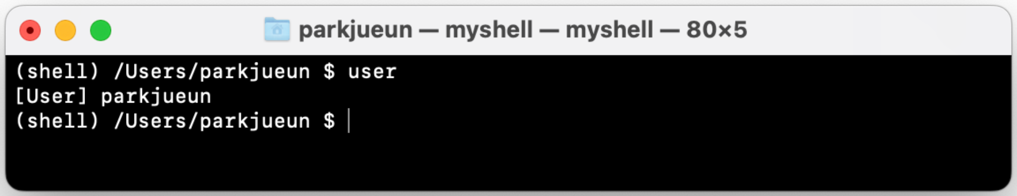
If the input string is 'path', print the path.



```
(shell) /Users/parkjueun $ path
echo$/Users/parkjueun/opt/anaconda3/bin:/Users/parkjueun/opt/anaconda3/condabin:
/opt/local/bin:/opt/local/sbin:/Library/Frameworks/Python.framework/Versions/3.7
/bin:/Library/Frameworks/Python.framework/Versions/3.9/bin:/usr/local/bin:/usr/b
in:/bin:/usr/sbin:/sbin:/Library/Apple/usr/bin
(shell) /Users/parkjueun $ 
```

(2)-3. user

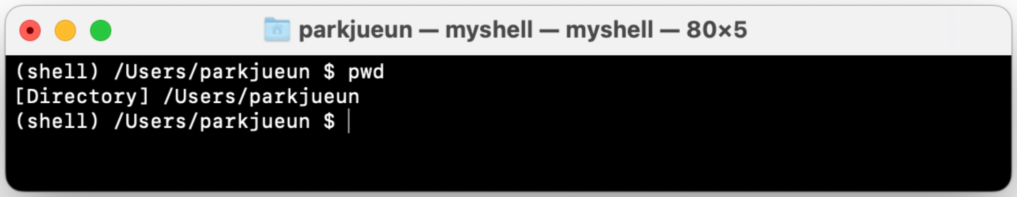
If the input string is 'user', print the username.



```
(shell) /Users/parkjueun $ user
[User] parkjueun
(shell) /Users/parkjueun $ 
```

(2)-4. pwd

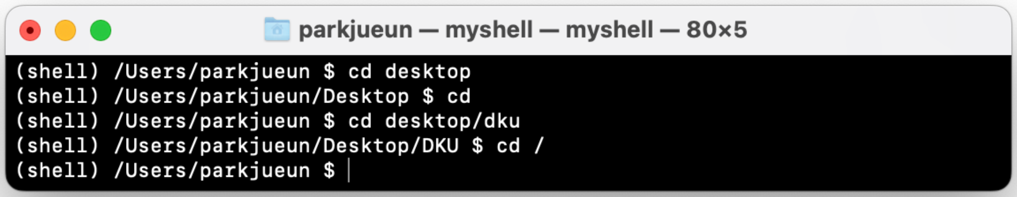
If the input string is 'pwd', print the current directory.



```
(shell) /Users/parkjueun $ pwd
[Directory] /Users/parkjueun
(shell) /Users/parkjueun $ 
```

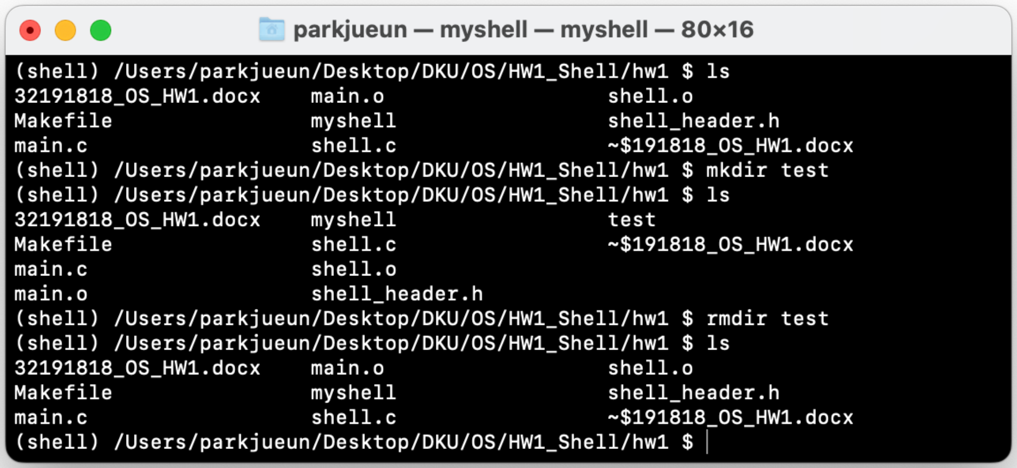
(2)-5. cd

If the input string is 'cd', using the chdir function, it is possible to move to the directory to be moved. That is, the location of the working directory is changed. If you enter a space or '/' after cd, go to home.



```
(shell) /Users/parkjueun $ cd desktop
(shell) /Users/parkjueun/Desktop $ cd
(shell) /Users/parkjueun $ cd desktop/dku
(shell) /Users/parkjueun/Desktop/DKU $ cd /
(shell) /Users/parkjueun $
```

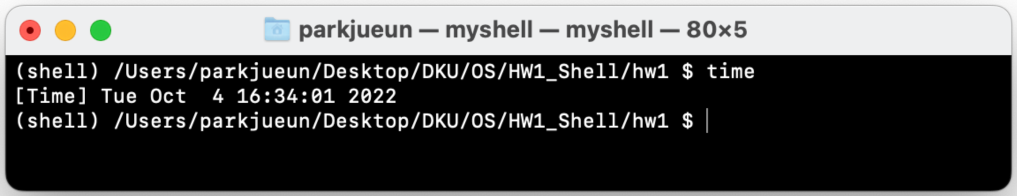
(2)-6. mkdir, rmdir
'mkdir' is a directory creation command, and 'rmdir' is a directory deletion command.



```
(shell) /Users/parkjueun/Desktop/DKU/OS/HW1_Shell/hw1 $ ls
32191818_OS_HW1.docx     main.o                  shell.o
Makefile                 myshell                 shell_header.h
main.c                   shell.c                 ~$191818_OS_HW1.docx
(shell) /Users/parkjueun/Desktop/DKU/OS/HW1_Shell/hw1 $ mkdir test
(shell) /Users/parkjueun/Desktop/DKU/OS/HW1_Shell/hw1 $ ls
32191818_OS_HW1.docx     myshell                 test
Makefile                 shell.c                 ~$191818_OS_HW1.docx
main.c                   shell.o
main.o                   shell_header.h
(shell) /Users/parkjueun/Desktop/DKU/OS/HW1_Shell/hw1 $ rmdir test
(shell) /Users/parkjueun/Desktop/DKU/OS/HW1_Shell/hw1 $ ls
32191818_OS_HW1.docx     main.o                  shell.o
Makefile                 myshell                 shell_header.h
main.c                   shell.c                 ~$191818_OS_HW1.docx
(shell) /Users/parkjueun/Desktop/DKU/OS/HW1_Shell/hw1 $
```

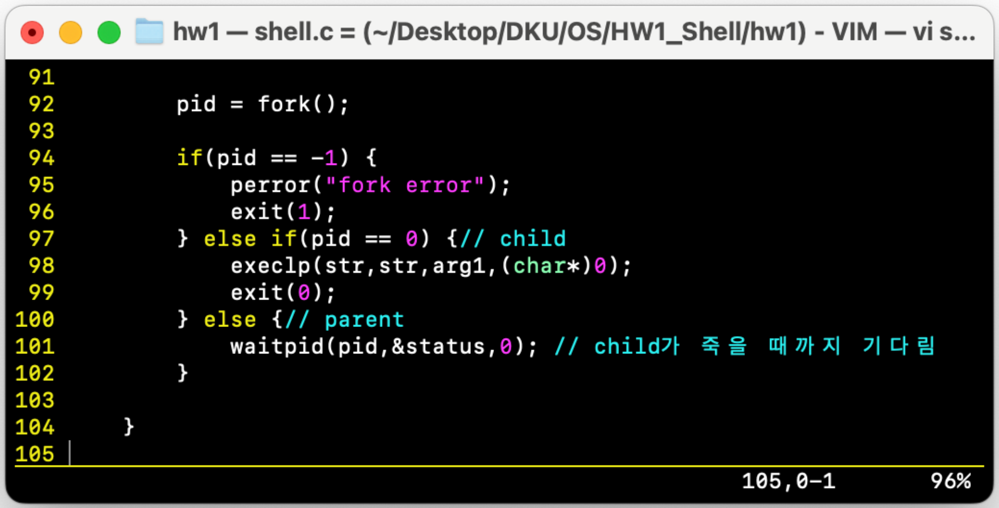
** I created 'test' directory and removed it.

(2)-7. time
It the input string is 'time', print the current time.



```
(shell) /Users/parkjueun/Desktop/DKU/OS/HW1_Shell/hw1 $ time
[Time] Tue Oct  4 16:34:01 2022
(shell) /Users/parkjueun/Desktop/DKU/OS/HW1_Shell/hw1 $
```

Duplicate the process by calling the fork() function on the pid. Since the parent process and the child process do different things, the pid return value is used. If the return value of the pid is -1, since this is the case where the fork call fails, an error message is output. Then, call exit (1) and forcefully terminate it. If the return value of the pid is 0, it is a part to be performed by the child process, and if it is greater than 0, it is a part to be performed by the parent process. The parent process uses the waitpid() function to put it on standby. In the child process, the execlp() function is called to execute str(file), and str and arg1 are transferred as factors. (The last factor of the function must be specified as a NULL pointer.)


hw1 — shell.c = (~/Desktop/DKU/OS/HW1_Shell/hw1) - VIM — vi s...

```

        pid = fork();

        if(pid == -1) {
            perror("fork error");
            exit(1);
        } else if(pid == 0) {// child
            execlp(str,str,arg1,(char*)0);
            exit(0);
        } else {// parent
            waitpid(pid,&status,0); // child가 죽을 때까지 기다림
        }

    }

                                                  105,0-1       96%
```

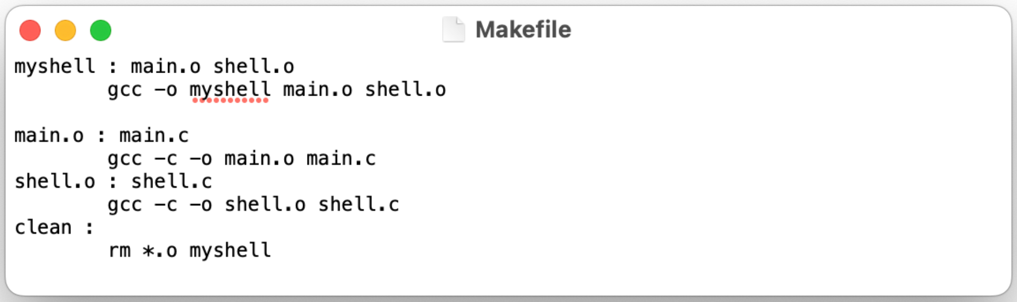
The reason for using fork() and execlp() together is to keep both parent and child alive while doing different tasks. It replicates using fork(), and executes other programs by calling execlp() in the child process. After that, when the process is terminated, the status value is delivered to the parent process. Therefore, the reason using waitpid() is to receive a return value from the terminated child process when the process is divided into parent and child.

** fork() : Create exactly the same process as oneself
** execlp() : Create a process from the specified executable
** waitpid : It will be in a waiting state until the specified own child process ends.

3. Optional

(1) make & compile: write a makefile that compiles your code. I will j ust type make on the terminal.

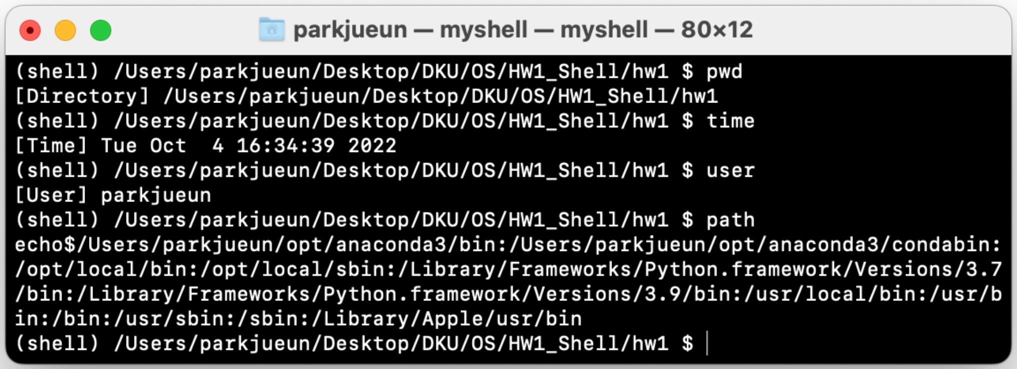
>>



```
myshell : main.o shell.o
	gcc -o myshell main.o shell.o

main.o : main.c
	gcc -c -o main.o main.c
shell.o : shell.c
	gcc -c -o shell.o shell.c
clean :
	rm *.o myshell
```

(2) You can specify different shell prompt using getenv function.
    (e.g. your current working directory (PWD, TIME, USER, etc.)

>> When environmental variables such as 'pwd', 'time', 'user', and 'pa th' are input into the command, the environmental variables are outp ut through getenv().



```
(shell) /Users/parkjueun/Desktop/DKU/OS/HW1_Shell/hw1 $ pwd
[Directory] /Users/parkjueun/Desktop/DKU/OS/HW1_Shell/hw1
(shell) /Users/parkjueun/Desktop/DKU/OS/HW1_Shell/hw1 $ time
[Time] Tue Oct  4 16:34:39 2022
(shell) /Users/parkjueun/Desktop/DKU/OS/HW1_Shell/hw1 $ user
[User] parkjueun
(shell) /Users/parkjueun/Desktop/DKU/OS/HW1_Shell/hw1 $ path
echo$/Users/parkjueun/opt/anaconda3/bin:/Users/parkjueun/opt/anaconda3/condabin:
/opt/local/bin:/opt/local/sbin:/Library/Frameworks/Python.framework/Versions/3.7
/bin:/Library/Frameworks/Python.framework/Versions/3.9/bin:/usr/local/bin:/usr/b
in:/bin:/usr/sbin:/sbin:/Library/Apple/usr/bin
(shell) /Users/parkjueun/Desktop/DKU/OS/HW1_Shell/hw1 $ 
```

(3) You can take additional input parameters for the executing progra m, and pass them to the created process. Please find manual pages of execve system call.

>> When the command is input, I added additional factors to enable additional functions. (e.g. mkdir test, rmdir test, cd /)

4. Program Structure

```
----------------shell()----------------
Get Environmental Variables
---------------------------------------
Loop
(1) getcwd()
(2) Get command
(3) Tokenize the command
(4) Execute the command
(5) pid = fork()
(6) if pid ==-1 : exit(1)
     else if pid == 0 : execlp() and then exit(0)
     else : wait
end Loop
```

5. Build Environment

MacOS Terminal, Visual Studio Code

6. Problems and Solutions

After receiving the string from fgets() and storing it in an array, I had a problem using it. Because I didn't know that 'newline' ('\n') should not exist. I didn't know even that a 'newline' would be attached when I received a string through fgets(), and it didn't work as I wanted. Therefore, I added null('\0') at the end of the array to remove this.
When implementing the environmental variable 'time', other environmental variables could be output through the getenv function. However, the output of getenv ("TIME") was a NULL value. Therefore, I decided to use the time function, 'localtime()'. This function returns time structure. And I used the 'asctime()' function to output the time. This function converts time into a

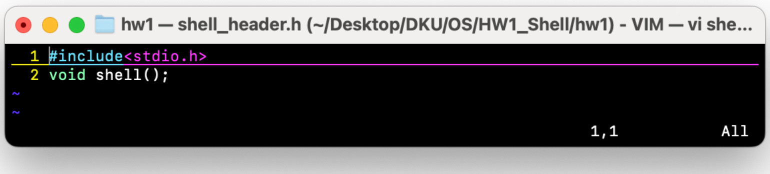
string.
I didn’t know how to make the ‘Makefile’. To make the ‘Makefile’, I had to learn some grammar. After then, I wrote making code on the ‘vi’.

7. Personal Feeling

When I first read the assignment, it was unfamiliar to me, so I was at a loss at first. And in order to do the task, I had to understand what a shell was. I've been using commands without any questions, but I was interested in receiving commands through Shell and seeing the process of operating differently depending on them. While studying Shell, I thought the most important thing to be dealt with was forks. When creating a process with fork, the execution state at the time of fork invocation is inherited, and it was interesting that changes made in the child process after fork did not affect the parent process. So I thought that the order of process progress would be important.

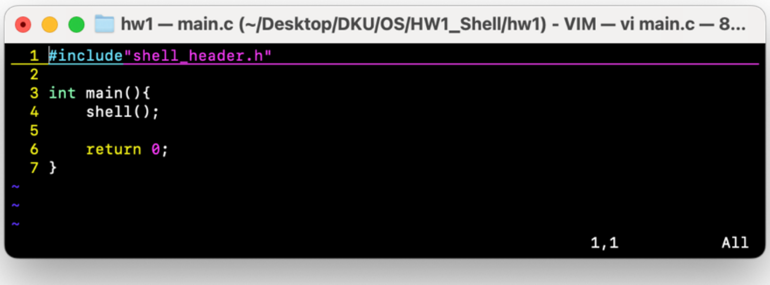
8. Code

shell_header.h


hw1 — shell_header.h (~/Desktop/DKU/OS/HW1_Shell/hw1) - VIM — vi she...

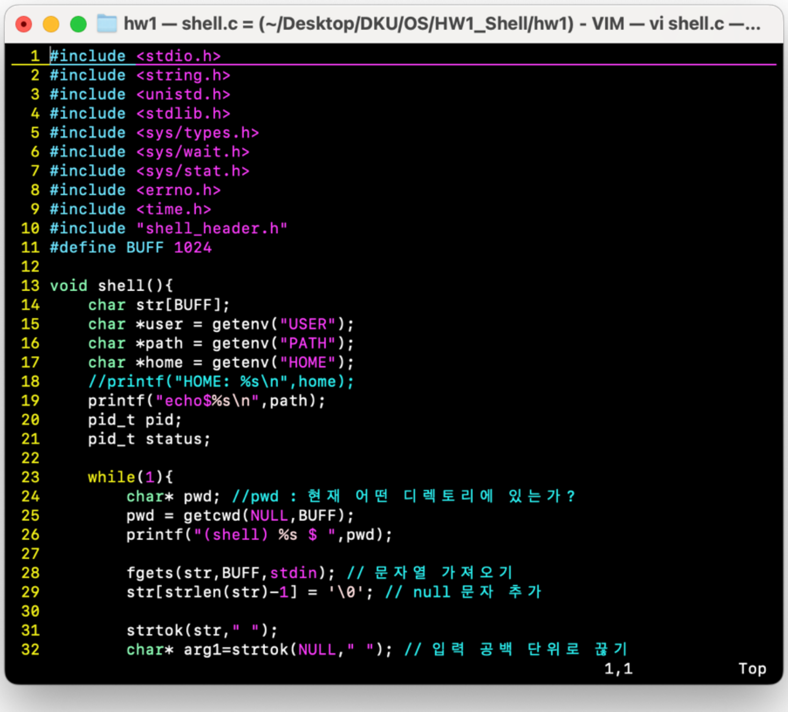
```c
#include<stdio.h>
void shell();
~
~
                                                    1,1           All
```

main.c


hw1 — main.c (~/Desktop/DKU/OS/HW1_Shell/hw1) - VIM — vi main.c — 8...

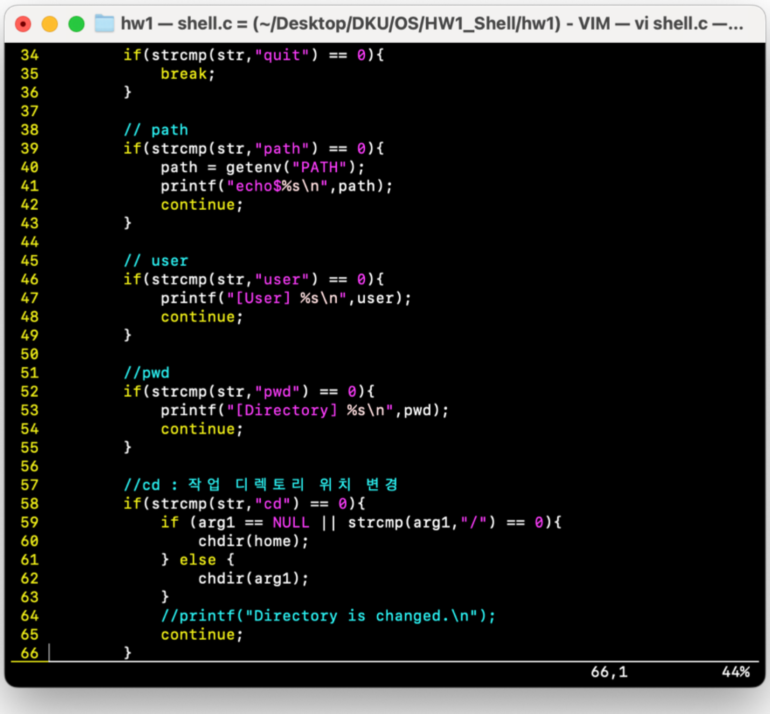
```c
#include"shell_header.h"

int main(){
    shell();

    return 0;
}
~
~
~
                                                    1,1           All
```

shell.c


hw1 — shell.c = (~/Desktop/DKU/OS/HW1_Shell/hw1) - VIM — vi shell.c —...

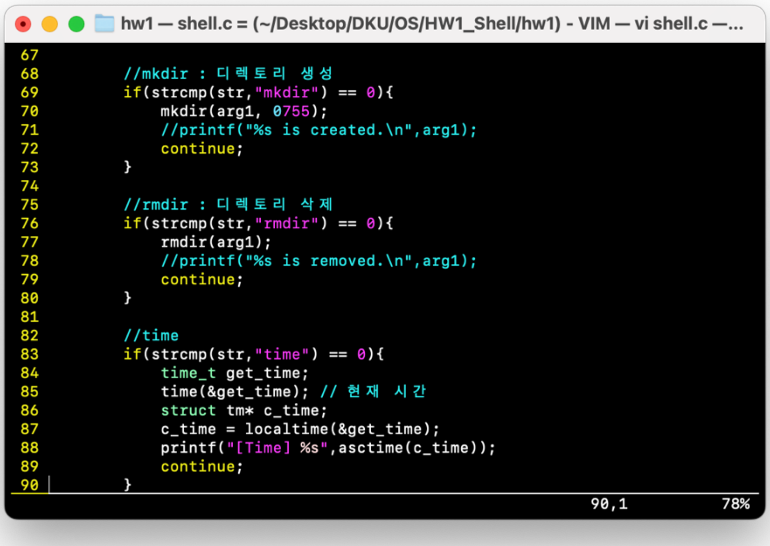
```c
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/stat.h>
#include <errno.h>
#include <time.h>
#include "shell_header.h"
#define BUFF 1024

void shell(){
    char str[BUFF];
    char *user = getenv("USER");
    char *path = getenv("PATH");
    char *home = getenv("HOME");
    //printf("HOME: %s\n",home);
    printf("echo$%s\n",path);
    pid_t pid;
    pid_t status;

    while(1){
        char* pwd; //pwd : 현재 어떤 디렉토리에 있는가?
        pwd = getcwd(NULL,BUFF);
        printf("(shell) %s $ ",pwd);

        fgets(str,BUFF,stdin); // 문자열 가져오기
        str[strlen(str)-1] = '\0'; // null 문자 추가

        strtok(str," ");
        char* arg1=strtok(NULL," "); // 입력 공백 단위로 끊기
```

```
1,1           Top
```


hw1 — shell.c = (~/Desktop/DKU/OS/HW1_Shell/hw1) - VIM — vi shell.c —...

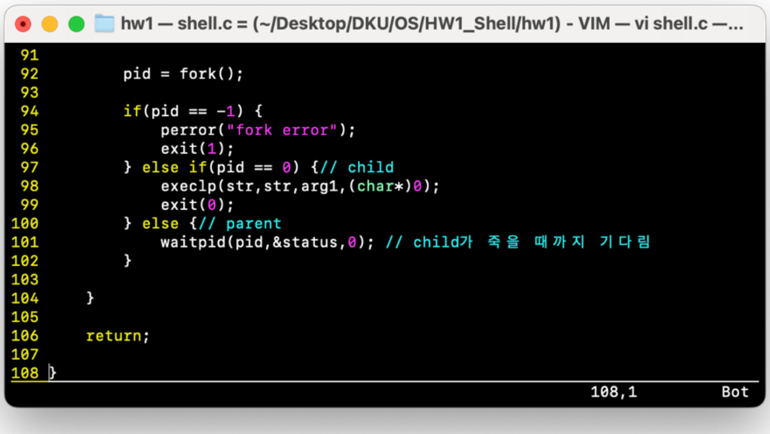
```c
        if(strcmp(str,"quit") == 0){
            break;
        }

        // path
        if(strcmp(str,"path") == 0){
            path = getenv("PATH");
            printf("echo$%s\n",path);
            continue;
        }

        // user
        if(strcmp(str,"user") == 0){
            printf("[User] %s\n",user);
            continue;
        }

        //pwd
        if(strcmp(str,"pwd") == 0){
            printf("[Directory] %s\n",pwd);
            continue;
        }

        //cd : 작업 디렉토리 위치 변경
        if(strcmp(str,"cd") == 0){
            if (arg1 == NULL || strcmp(arg1,"/") == 0){
                chdir(home);
            } else {
                chdir(arg1);
            }
            //printf("Directory is changed.\n");
            continue;
        }
```

```
66,1          44%
```



hw1 — shell.c = (~/Desktop/DKU/OS/HW1_Shell/hw1) - VIM — vi shell.c —...

```c

        //mkdir : 디렉토리 생성
        if(strcmp(str,"mkdir") == 0){
            mkdir(arg1, 0755);
            //printf("%s is created.\n",arg1);
            continue;
        }

        //rmdir : 디렉토리 삭제
        if(strcmp(str,"rmdir") == 0){
            rmdir(arg1);
            //printf("%s is removed.\n",arg1);
            continue;
        }

        //time
        if(strcmp(str,"time") == 0){
            time_t get_time;
            time(&get_time); // 현재 시간
            struct tm* c_time;
            c_time = localtime(&get_time);
            printf("[Time] %s",asctime(c_time));
            continue;
        }
                                                        90,1          78%
```



hw1 — shell.c = (~/Desktop/DKU/OS/HW1_Shell/hw1) - VIM — vi shell.c —...

```c

        pid = fork();

        if(pid == -1) {
            perror("fork error");
            exit(1);
        } else if(pid == 0) {// child
            execlp(str,str,arg1,(char*)0);
            exit(0);
        } else {// parent
            waitpid(pid,&status,0); // child가 죽을 때까지 기다림
        }

    }

    return;

}
                                                        108,1         Bot
```