

2주차 보고서

작성: 스마트보안학과 2023350018 서민주

1. 가장 어떤 라이브러리가 필요한지 GPT에게 물어봤다.

- `unistd.h`
 - 시스템 호출 관련 (`execve` , `fork` , `getcwd` , `chdir` 등)
 - POSIX 표준 함수 제공
 - 파일 디스크립터 조작 (`read` , `write` , `close`)
- `sys/wait.h`
 - 자식 프로세스의 종료 상태 관리 (`wait` , `waitpid`)
- `fcntl.h`
 - 파일 제어 (`open` , `fcntl`)
- `stdlib.h`
 - 메모리 관리 (`malloc` , `free`)
 - 프로세스 종료 (`exit`)
 - 환경 변수 (`getenv`)
- `string.h`
 - 문자열 처리 (`strlen` , `strtok` , `strcpy` , `strcmp`)
- `stdio.h`
 - 기본 입출력 (`printf` , `fgets` , `perror`)

2. main 코드 작성

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
#include <fcntl.h>

int main() {
```

```

char line[1024];

while (1) {
    printf("my_shell$ ");
    if (fgets(line, sizeof(line), stdin) == NULL) break;
    printf("Command: %s", line);
}

return 0;
}

```

3. Makefile 생성

```

myang@MYANG:/mnt/d/KUSS/Cykor/2025/week2$ make
gcc -Wall -Wextra -O2 -o myshell main.c
myang@MYANG:/mnt/d/KUSS/Cykor/2025/week2$ ls
Makefile main.c myshell

```

GPT로 Makefile을 생성하였다.

```

myang@MYANG:/mnt/d/KUSS/Cykor/2025/week2$ ./myshell
my_shell$ ls
Command: ls
my_shell$ ls
Command: ls
my_shell$ clear
Command: clear
my_shell$ exit
Command: exit
my_shell$ ^C

```

명령어를 입력하면 해당 명령어가 반영되는 모습까지 보인다.

4. 현재 디렉토리 명을 셸에 표시

```

myang@MYANG:/mnt/d/KUSS/Cykor/2025/week2$ make
gcc -Wall -Wextra -O2 -o myshell main.c
myang@MYANG:/mnt/d/KUSS/Cykor/2025/week2$ ./myshell
[week2]$ ls
Command: ls

[week2]$ pwd
Command: pwd

```

이전에 단순 printf 함수를 이용해 출력했던 것을 현재 디렉토리 명으로 출력되게 변경하였다.


5. Parsing command (analysis_prompt)


a. 다중 명령어 ; 구현

가장 먼저 line에 명령이 들어오면 이것이 단일명령인지 다중명령인지 구분해주는 것이 가장 먼저 필요하다고 생각했다. 따라서, 명령어를 구분하는 코드를 먼저 구현하였다.

[C언어/C++] strtok 함수(문자열 자르기)에 대해서.

안녕하세요. BlockDMask 입니다. 오늘 공부할 함수는 문자열을 일정 기준을 정해서 썩둑썩둑 자를 수 있는 strtok 함수입니다. C언어 strtok 함수에 대해서 한번 알아보러 가보겠습니다

 <https://blockdmask.tistory.com/382>

[C/C++] 
strtok
문자열 나누기

해당 티스토리를 참고해서 구현하였다.

리눅스에서 여러 명령어 입력은 ; 으로 구분되므로 char* delimiters 에 ; 을 넣어주었다.

```
myang@MYANG: /mnt/d/KUSS/Cykor/2025/week2$ ./myshell
[week2]$ ls;cd
Command: ls;cd

After tokenized: ls
After tokenized: cd
```

```
myang@MYANG: /mnt/d/KUSS/Cykor/2025/week2$ ./myshell
[week2]$ ls|cd;pwd
Command: ls|cd;pwd

After tokenized: ls|cd
After tokenized: pwd
```

그러나 ; 를 제외한 다른 다중명령어를 입력하면 인식이 안 된다. 따라서, 이들을 모두 파싱해주는 작업이 필요하다.

6. pwd 구현

```
void run_pwd()
{
    char cwd[BUFSIZE];
```

```

    if (getcwd(cwd, sizeof(cwd)) != NULL)
        printf("%s\n", cwd);
    else
        perror("pwd");
}

```

`getcwd()` 함수를 이용하여 현재 디렉토리 위치를 가지고 온 후 출력하였다.

7. cd 구현

`chdir()` 함수를 이용하여 구현을 진행하였다.

```

main.c: In function 'run_cd':
main.c:95:9: warning: ignoring return value of 'chdir' declared with attribute 'warn_unused_result' [-Wunused-result]
   95 |         chdir(".");
      |         ^~~~~~
main.c:97:9: warning: ignoring return value of 'chdir' declared with attribute 'warn_unused_result' [-Wunused-result]
   97 |         chdir("/");
      |         ^~~~~~
main.c:99:9: warning: ignoring return value of 'chdir' declared with attribute 'warn_unused_result' [-Wunused-result]
   99 |         chdir(getenv("HOME"));
      |         ^~~~~~

```

`chdir()` 함수는 반환값을 가지는 함수인데, 반환형으로 쓰지 않아 경고가 발생하였다. 따라서, 코드를 다음과 같이 변경하였다.

```

if (strcmp(args[1], "~") == 0)
{
    if (chdir(getenv("HOME")) != 0) perror("cd");
    return;
}

```

```

myang@MYANG: /mnt/d/KUSS/Cykor/2025/week2$ ./myshell
[week2]$ pwd
/mnt/d/KUSS/Cykor/2025/week2
[week2]$ cd .
[week2]$ cd ..
[2025]$ cd ~
[myang]$ cd /
[]$ exit
myang@MYANG: /mnt/d/KUSS/Cykor/2025/week2$

```

코드 좀 더 단순하게 수정

```

[]$ cd ~
Error: cd failed to change directory to ~.

```

이 부분만 추가로 구현 (HOME으로 이동하게)

```
myang@MYANG:/mnt/d/KUSS/Cykor/2025/week2$ ./myshe1l  
[week2]$ cd ~  
[myang]$ exit
```