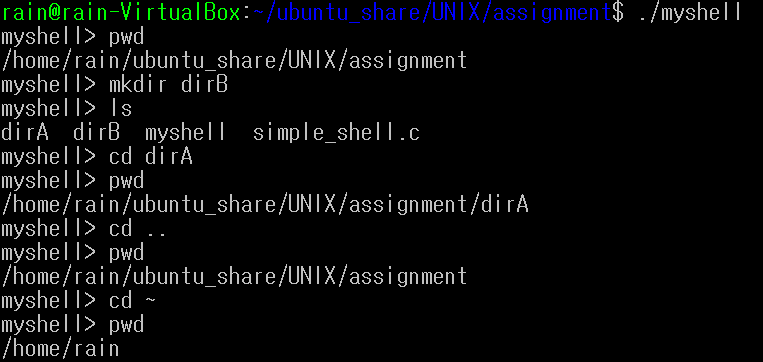
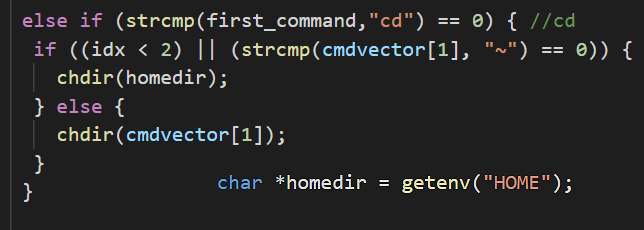
**UNIX Programming Shell 과제\_12171607 김한비**

**Project1**

**제출일: 2019.11.04**

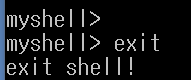
* **cd 명령어 구현**

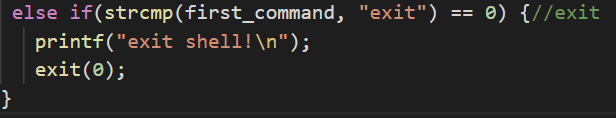




chdir()은 현재 프로세스에만 적용되기 때문에 fork 하기 전에 명령어를 실행하게 함

* **exit 명령어 구현**

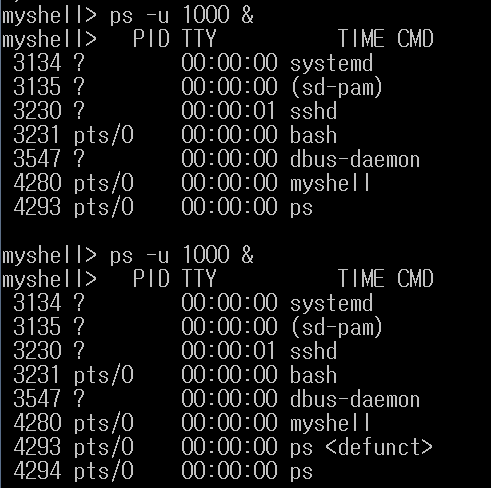




exit(0)을 이용함, 인자가 없을 때 에러 나지 않도록 그거 예외처리도 해줬음

* **background 실행 구현**





Ps를 입력하면 계속 background에서 실행되고 있는 프로세스들이 보인다.



&이 있는지 없는지 확인하는 변수를 이용, &이 있으면 부모가 wait 하지 않아 자식이 daemon process가 되게 했다.

Sleep은 int형 변수를 인자로 받기 때문에 시스템 콜과 atoi()를 이용해서 처리했다.

좀비 프로세스(<defunct>)가 생기는 이유?

-> fork()를 하면 프로세스가 자식인지 부모인지를 확인하면서 프로그램 진행.

자식프로세스가 종료되면 부모프로세스에게 SIGCHLD 시그널을 보냄

이 시그널을 받은 부모프로세스는 wait으로 자식의 상태를 받고 이 과정이 끝나야 자식이 완전히 종료됨

자식이 하나일 때는 이 과정에 문제가 없지만 자식이 여럿일 때 한 자식의 SIGCHLD를 처리하는 중이면 다른 자식이 종료되었을 때 발생하는 SIGCHLD는 처리하지 않는다.

따라서 좀비를 만들지 않고 모든 자식을 정상적으로 종료시키고 싶으면 루프를 사용해야 한다.

Ex) 자식이 종료되었을 때 처리: while(waitpid((-1,&status,WNOHANG)>0);

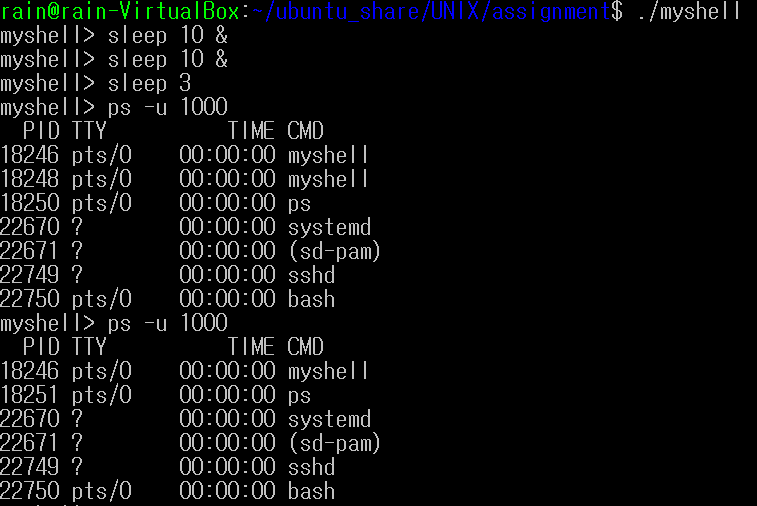
자식이 종료 또는 중지되었을 때 처리: while(waitpid(-1,&status,WNOHANG|WUNTRACKED)>0);

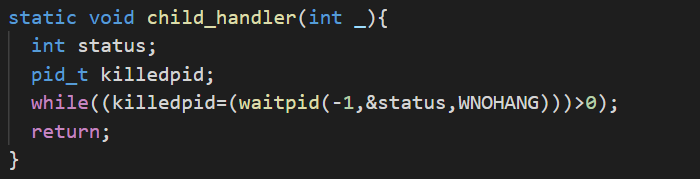
**`Project 2**

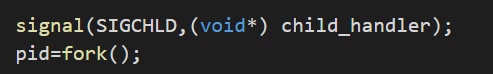
**제출일 : 2019. 11. 9**

* **SIGCHILD로 자식프로세스 wait()시 프로세스가 온전하게 수행되도록 구현**

위에 서술한 좀비 프로세스가 생기는 이유 고찰을 통해 반복문을 이용, 문제점을 처리했다.

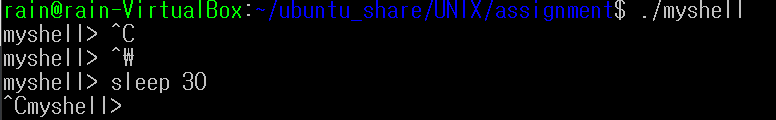


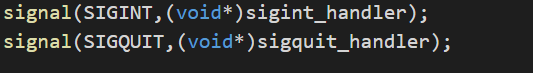




Fork 전에 signal 처리 함수를 넣었다.

* **SIGINT, SIGQUIT 사용시 쉘이 종료되지 않도록 하고 foreground 프로세스 실행 시 SIGINT를 받으면 프로세스가 끝나는 것을 구현**





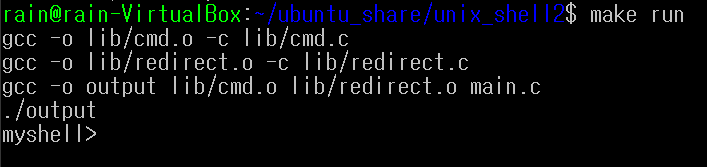
명령어 입력 후 signal 함수를 넣으니까 아무것도 입력 안 한 상태에서 CRTL+C 하면 쉘이 종료되길래 입력 전으로 함수를 뺐더니 쉘이 종료되지 않아서 조건에 맞게 구현 할 수 있었다.

**Project 3**

**제출일: 2019.11.20**

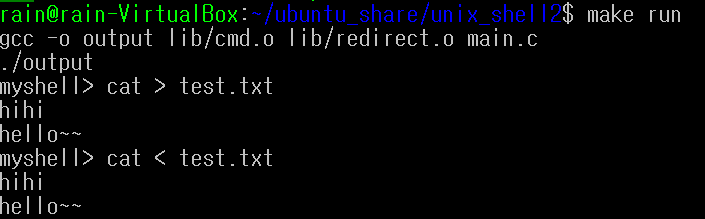
계속 한 파일에서 코딩을 하고 메인 함수에 모든 코드를 넣다 보니 짜는데 어려움이 생겨서 코드를 리팩토링 했습니다.

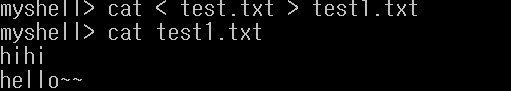
* **컴파일 방법: make run을 입력**



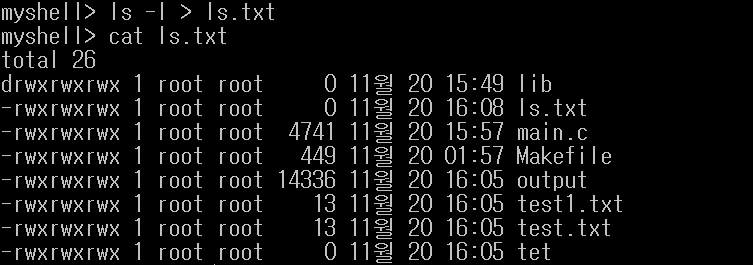
* **실행결과**

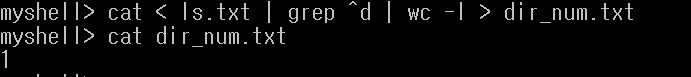
- redirection test





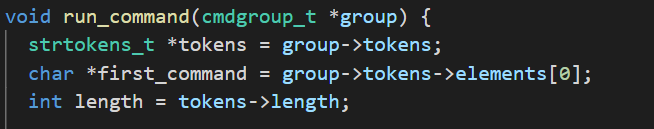
-redirection & pipe test



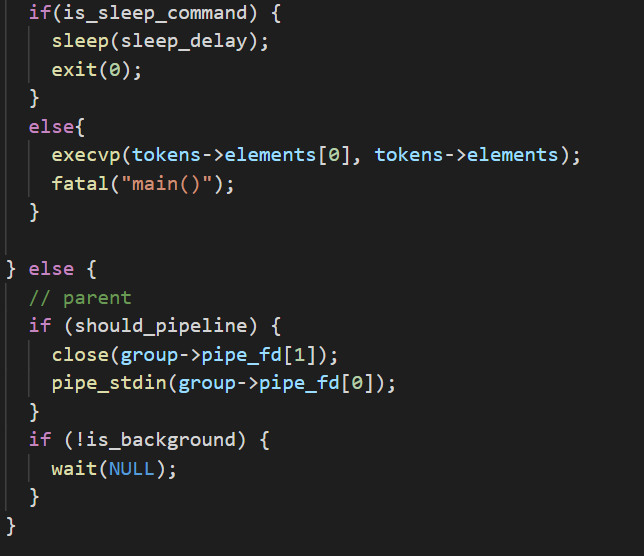


* **Redirection & Pipe 코드**

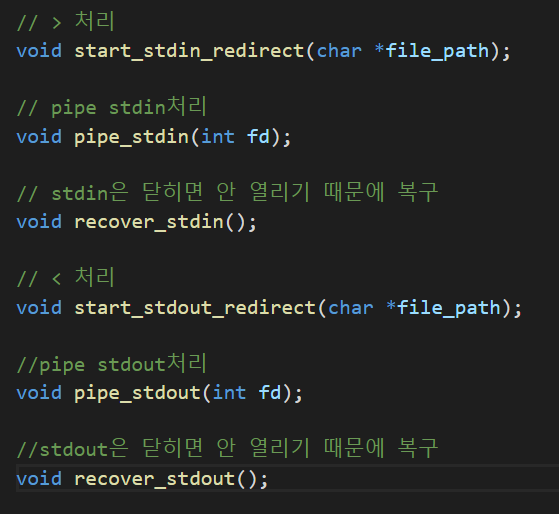
- cmdline에서 pipe, redirection 구분



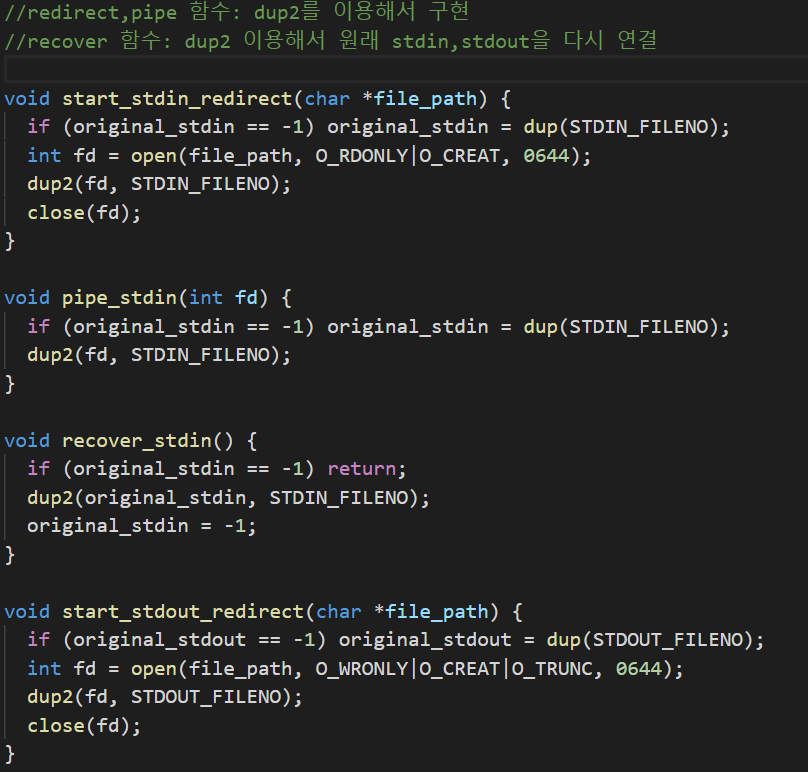


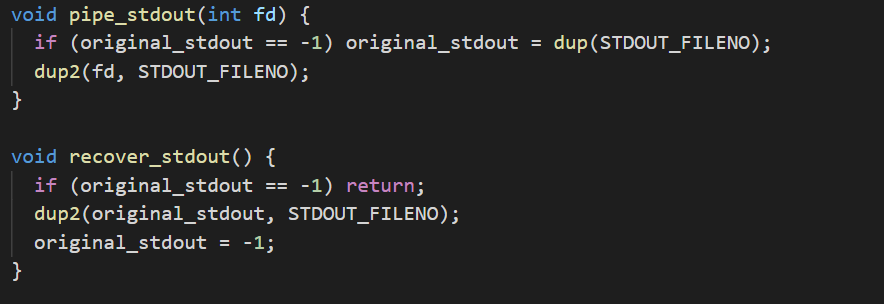


- Redirect.h



- Redirect.c





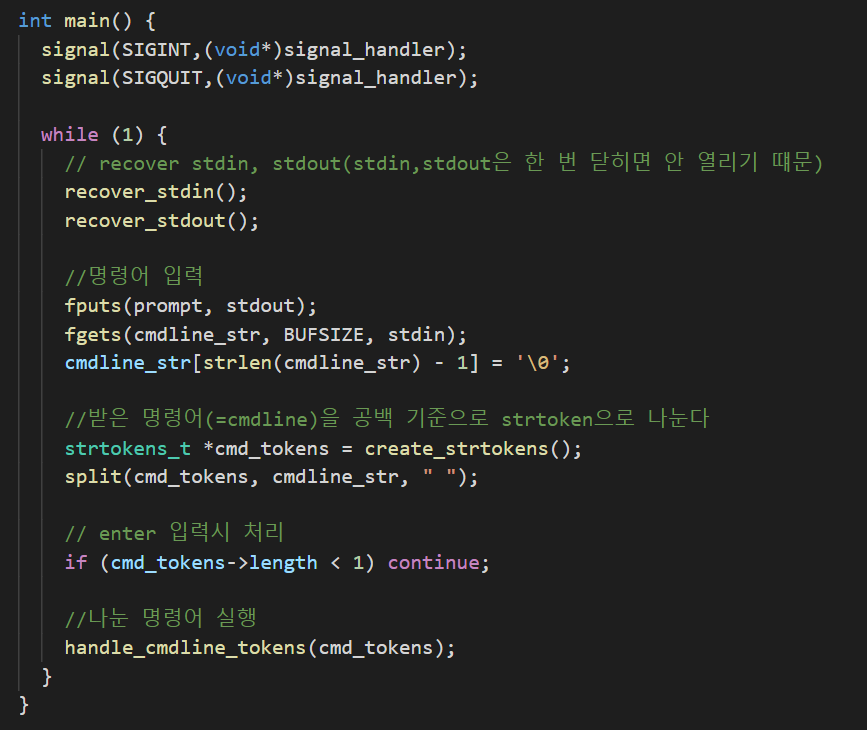
* 리팩토링 코드 설명

- main.c , lib폴더 안의 redirect.c & redirect.h, cmd.h & cmd.c로 구성

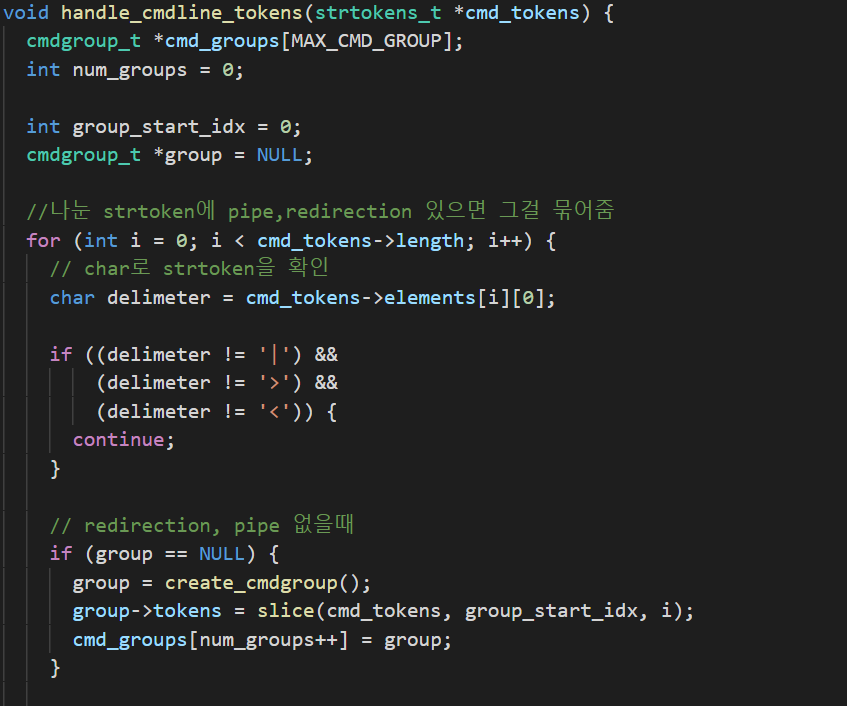
위에 redirect.h,redirect.c 코드는 캡쳐했으므로 나머지만 캡쳐했습니다.

Main.c

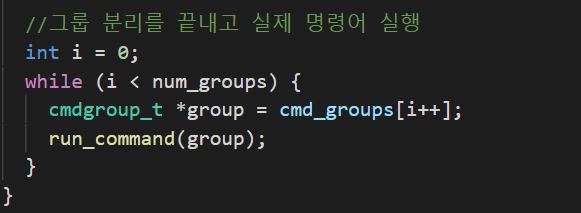
- main



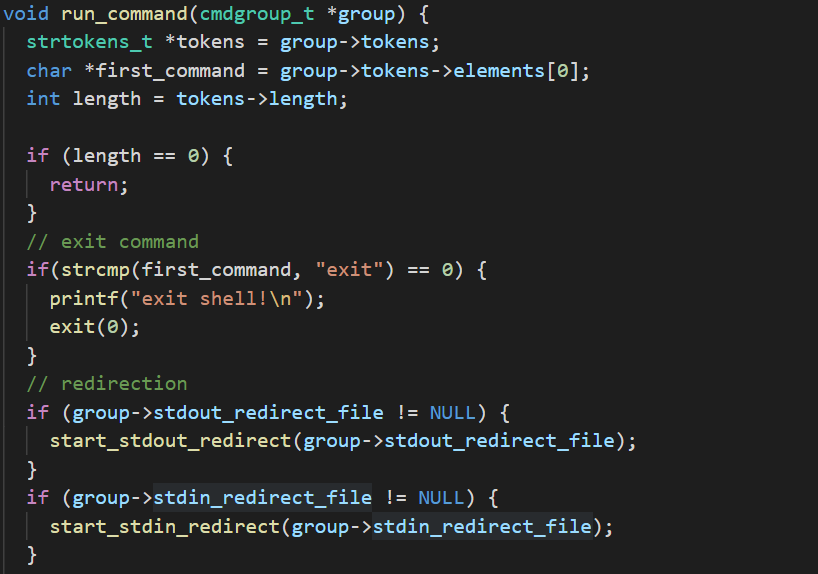
- handle\_cmdline\_tokens

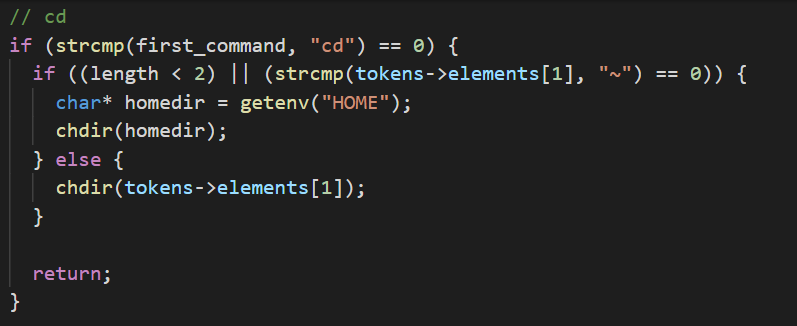




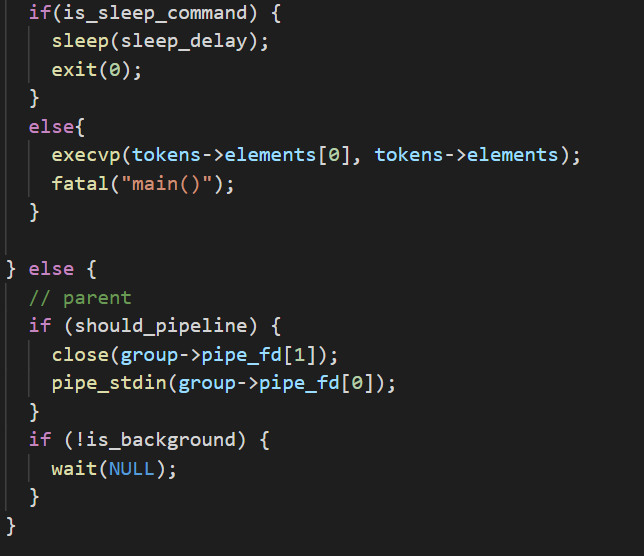


-run\_command

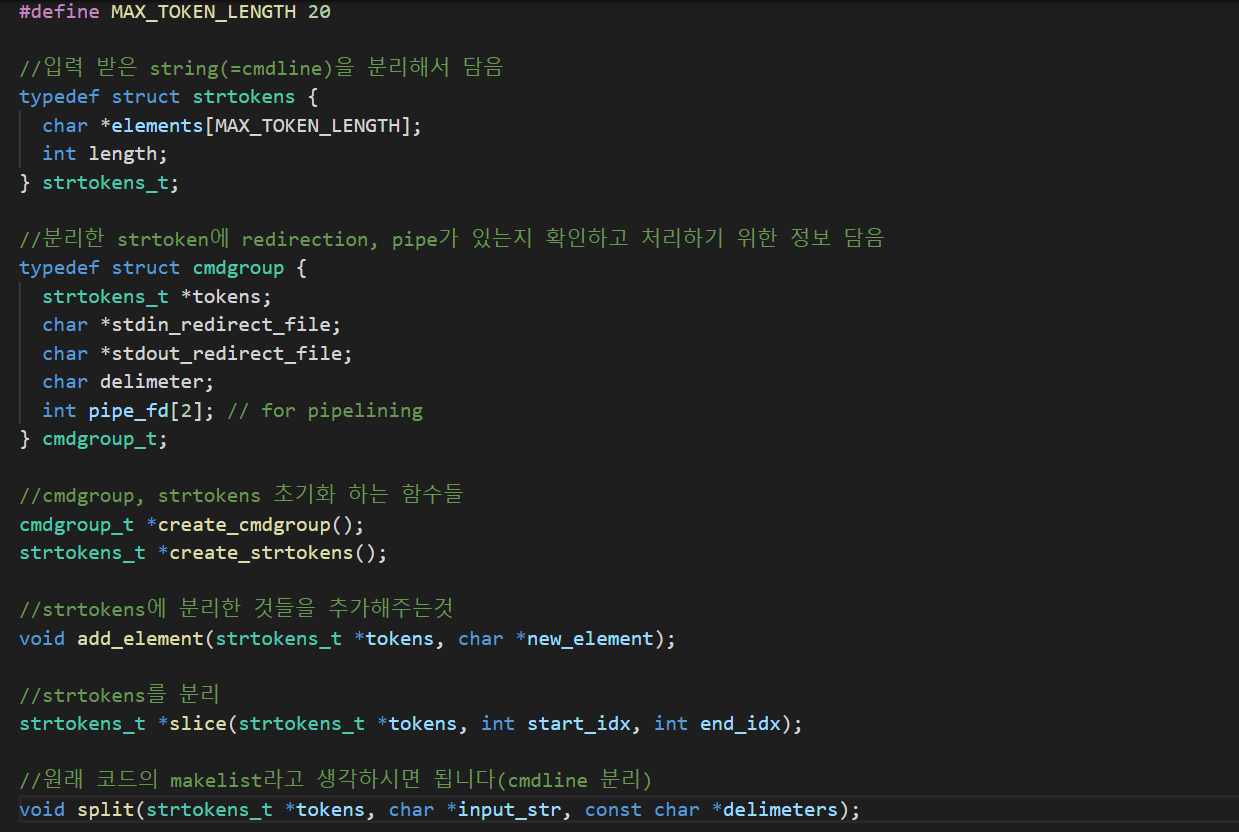








cmd.h



cmd.c

