|  |
| --- |
| HW6. 구조체 배열  **분반: 061**  부산대학교 경영학과  201343151  김영훈  제출일: 2019-05-20 |

# 구현 내용에 대한 설명 (50점)

1. **주요 변수 설명: extern과 static 변수는 반드시 설명함**

- static int count\_service : 전역 변수에 static을 선언하여 외부 소스 파일에서 사용하지 못하도록 제한하였음. Exit 선택 시, 총 서비스를 몇 번을 이용했는지 출력하기 위한 변수

- extern struct Contact PhoneBook[MAX] : phone.h에 있는 외부 변수를 extern 명령어를 통해 불러옴. 여러 소스 파일에서 선원된 PhoneBook 구조체 배열을 공유할 수 있게 됨.

- extern char password[100] : phoneBookMain.c에 선언된 전역 변수 password를 register.c에서 외부 변수로 불러옴. 입력된 비밀번호와 동일한지 비교하기 위함.

- wrongCount : register.c에서 비밀번호 틀린 횟수를 저장, 3이 되면 초기 메뉴로 돌아감

1. **주요 자료 구조 설명**

- struct Contact PhoneBook[MAX] : 전화번호부의 이름(Name, char[10])과 번호(PhoneNumber, char[16])를 저장하기 위한 구조체 배열. 기존의 번호를 담을 수 있는 13이 '\0'값을 포함하지 못하게 되어 제대로 저장이 안되는 문제를 해결하기 위해 크기를 16으로 늘림.

1. **주요 함수 구현 방법 설명**

- void registerPhoneData() : char inputPassword[100] 변수 선언, 비밀번호를 입력받아 기존의 비밀번호와 비교. 다를 시, wrongCount를 늘리며 3이 되면 오류 메시지와 함께 함수 종료. 비밀번호를 제대로 입력한 경우, 새로 이름과 번호를 입력받아 phoneBook[size]에 저장 후 size++ 수행.

- void printAll() : i를 0에서 size-1까지 반복하며 phoneBook[i]의 Name과 PhoneNumber를 출력함.

- void searchByName() : char searchName[10] 변수 선언. 검색할 이름을 입력받고, i를 0에서 size-1까지 반복하며 string.h의 strcmp 함수를 사용하여 PhoneBook[i]의 Name과 비교. 같을 시, 해당 PhoneBook[i]의 Name과 PhoneNumber 출력. 없을 시, 해당 이름이 없다는 메시지 출력

- void deleteByName() : char deleteName[10], int findIndex = -1 변수 선언. 제거할 이름을 입력받고, i를 0에서 size-1까지 반복하며 strcmp 함수를 통해 이름 비교. 같을 시에 findIndex = i 수행. 만약 findIndex == -1이면 오류 메시지 출력. 아니라면, i를 findIndex에서 size-2까지 반복하며 strcpy 함수를 이용하여 PhoneBook[i]에 PhoneBook[i+1]의 Name과 PhoneNumber 복사 후 size-- 수행.

1. **Makefile 설명**

- CC, CFLAGS, INCLUDE, LIBS, OBJS 매크로 선언.

- %.o: %.c의 확장자 규칙을 이용하여 각 소스파일을 따로 명시하지 않아도 gcc -c 명령어로 목적파일(.o)을 모두 생성하도록 함.

- phoneBookMain: $(OBJS)를 통해 의존해야할 파일을 OBJS 매크로로 축약. gcc를 이용하여 phoneBookMain(실행파일)을 만들도록 명시.

- clean: rm -f phoneBookMain $(OBJS)를 이용하여, make clean 명렁어 시에 소스파일을 제외하고 모두 삭제할 수 있도록 하였음.

# 실행 방법 설명 (20점)

1. **사용한 운영체제 및 컴파일러의 종류**

- ubuntu 16.04에서 vi를 이용하여 소스 파일 생성

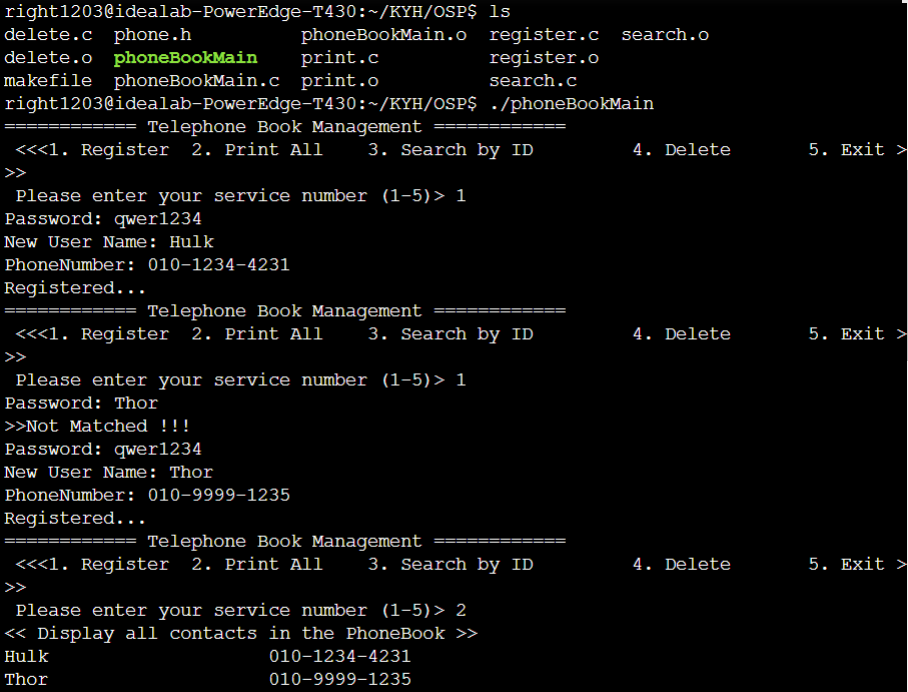
- gcc 컴파일러 사용

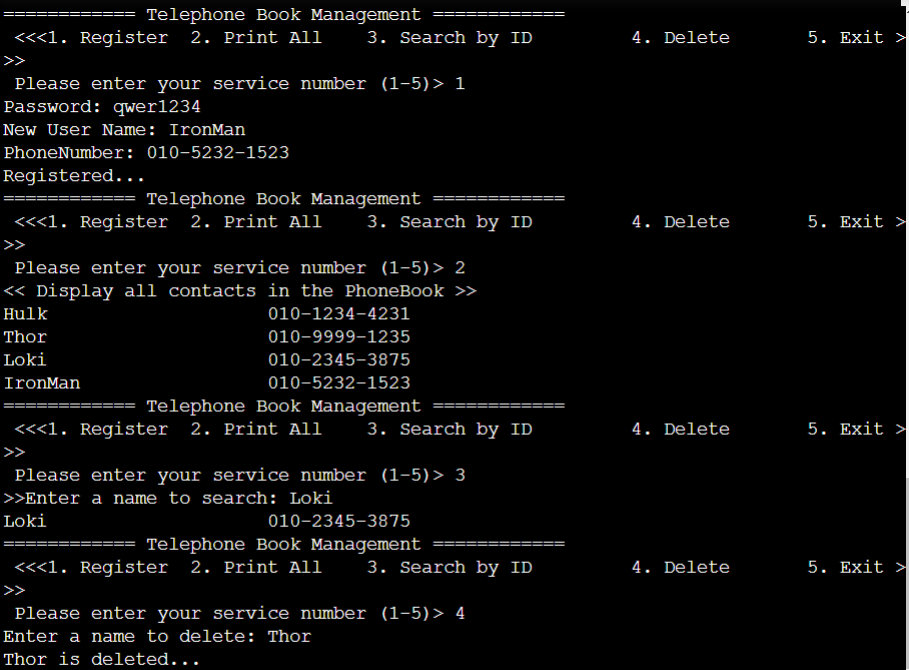
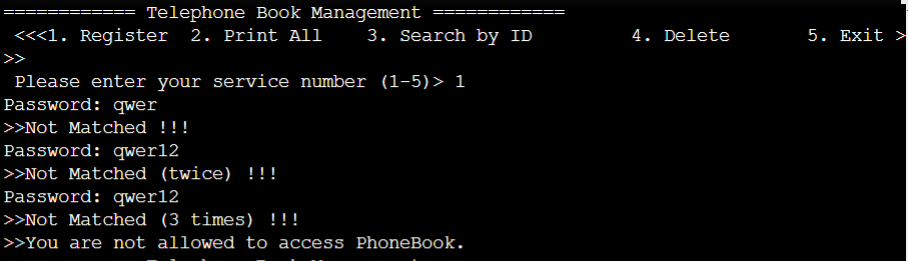
1. **컴파일 방법 및 실행 방법**

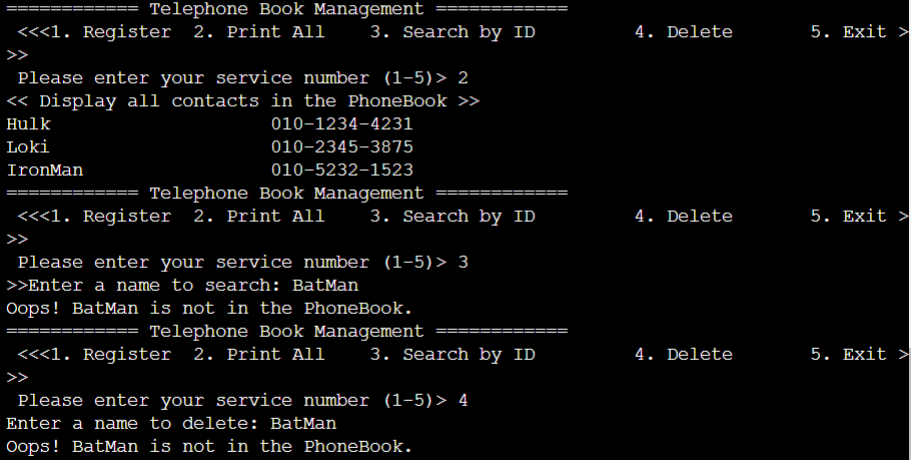
- 리눅스의 gcc 컴파일러를 이용하여 컴파일

- makefile 파일을 생성하고, make 명령어로 생성된 phoneBookMain을 실행

**(3) 동작을 확인할 수 있는 실행 화면 캡처**



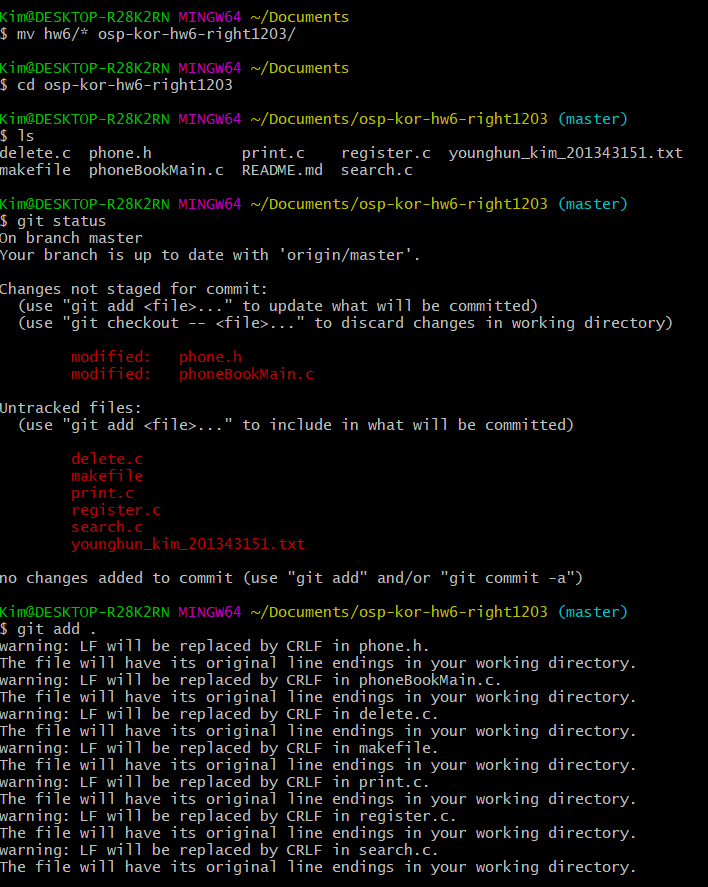


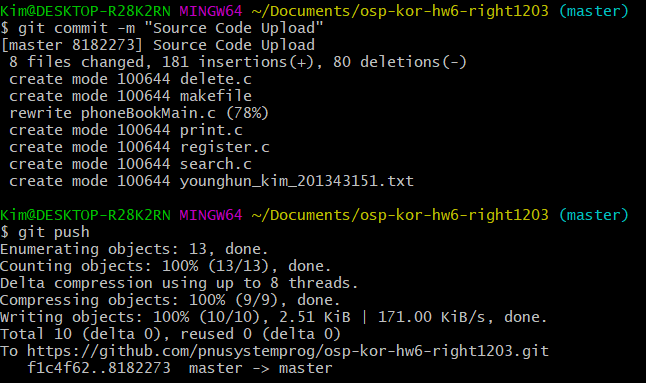


- 설명 파일에 있는 조건들을 지키고, 모든 수행 사항을 프로그래밍하였음.

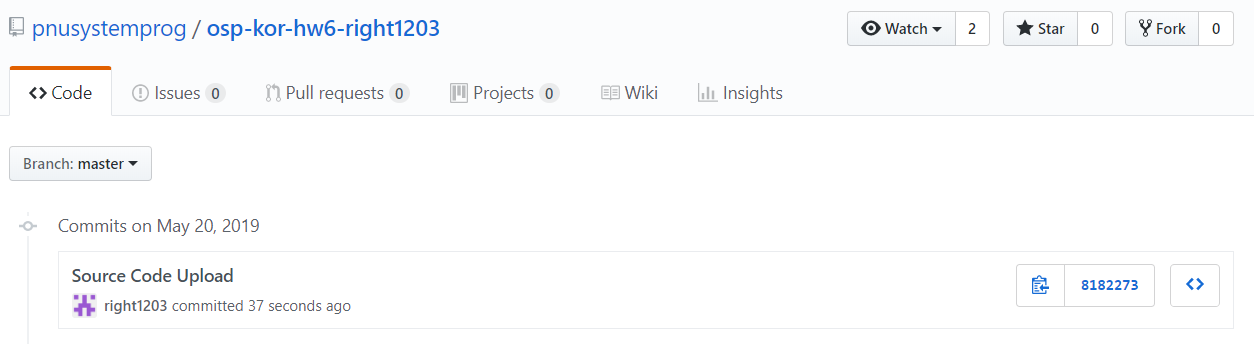
# Github 화면 (20점)

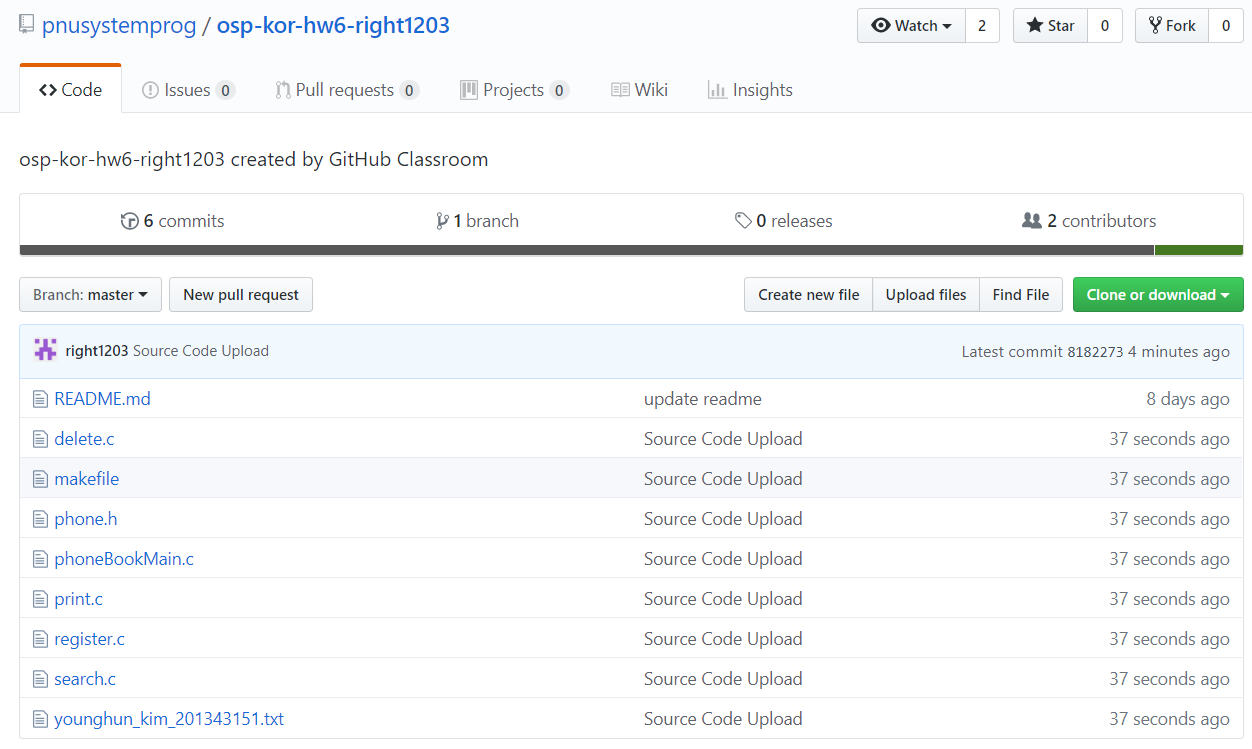
**(1) cloning, adding, committing, push을 위한 github 명령들을 포함**





**(2) 소스 코드와 makefile을 push한 후, 본인의 Github repository를 스크린 캡쳐하여 포함**





# 논의 사항 (10점)

**- 숙제를 하면서 새로이 알게 된 내용(수업 시간 이외의 내용)**

각 함수를 소스코드마다 따로 구현하여 gcc해본 경험이 처음이었습니다. 그래서 어떻게 해야하는지, 외부 변수를 어떻게 선언하고 static을 어떻게 활용하는지를 배울 수 있었습니다. 그리고 실제 작성한 파일로 makefile을 작성한 경험도 처음이기에 많은 공부가 되었습니다.

또한 vi 에디터를 좀 더 프로그래밍에 적합하도록 바꿀 수 있는 방법들을 찾아보며 알게 되었습니다.

**- 숙제를 하는 중에 어려웠던 점 등을 기술**

vi 에디터로 소스코드를 편집하려니 어려웠습니다. 그래서 vi 편집기를 조금 더 편하게 쓸 수 있는 방안들을 찾고 실행하는 것에 시간이 많이 소모되었습니다. 아직은 codeblocks와 같은 IDE가 더 편하지만 이 경험으로 인해 vi 에디터를 좀 더 능숙하게 다룰 수 있게 된 것 같습니다.