|  |
| --- |
| HW6. 구조체 배열  **분반: 061**  부산대학교 정보컴퓨터공학부  2017-24650  김선우  제출일: 2019-05-20 |

# 구현 내용에 대한 설명

1. 주요 변수 설명: extern과 static 변수는 반드시 설명함]
2. 상수  
   MAX : 저장할 수 있는 번호의 최대 수 (50)
3. Static 변수  
   input : 메인 메뉴에서 입력한 명령어를 저장 (1 ~ 5)
4. Extern 변수  
   password : 프로그램에서 사용할 비밀번호를 저장  
   size : 현재 저장된 연락처들의 수 (= 구조체 배열의 크기)  
   contact book : 연락처들을 저장한 구조체 배열
5. 주요 자료 구조 설명

struct contact { 전화번호부에 저장할 정보의 구조체  
 char name [10]; 저장할 인물의 이름  
 char pNumber[13]; 저장할 인물의 전화번호

};

struct contact book[MAX]; 위의 각 인물의 정보의, 크기가 50인 구조체 배열

1. 주요 함수 구현 방법 설명
2. registerPhonedata() 전화번호 정보를 저장하는 함수

int errors = 0 암호 입력 횟수를 체크하는 변수

char name[], phone[], pass[] 등록할 인물의 이름과 전화번호, 입력할 비밀번호

while (error != 3) {

scan pass 암호를 입력받음

if (strcmp(pass,password){ 입력받은 암호가 일치하지 않는 경우

error ++;

switch(error)

case1 : error msg once

case2 : twice

caes3 : three times then break

else 일치함

print guide msg then

scan name, phone

strcpy name , phone to book[size].name and book[size].pNumber

size ++;

1. printAll() 저장 된 모든 정보를 출력

print guide msg

for (int i = 0 to size) print book[i].name book[i].pNumber;

1. searchByName() 이름을 기준으로 탐색을 실시함

char search 검색하려는 사람의 이름

int count = 0 검색이 안될 경우를 확인하기 위한 변수

print guide msg then scan search 안내메시지 출력 후 search 입력

for (int i = 0 to size)

if (!strcmp(search, book[i].name) then 이름이 일치하는 정보를 확인

print book[i].name book[i].pNumber

else count ++

if (count == size) then print error msg 일치하는 검색 결과가 없음

1. deleteByName() 이름을 기준으로 탐색, 대상을 삭제함

char search

int count = 0

print guide msg then scan search

for (int i = 0 to size)

if (!strcmp(search, book[i].name) then

for (int j = i to size)

book[j] = book[j +1] 삭제할 대상의 값을 다음의 것으로 덮어씌움

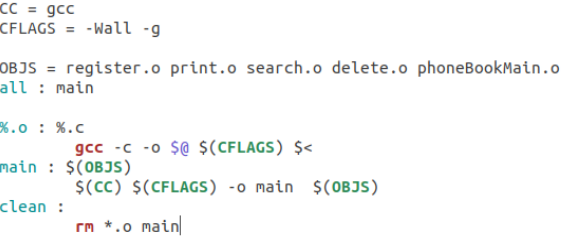
size --; 삭제 후 구조체배열의 size를 하나 줄임

break;

else count ++

if (count == size) then print error msg

1. Makefile 설명



CC / CFLAGS : 컴파일에 사용할 명령어를 매크로로 지정

OBJS : 만들어질 오브젝트파일 명들을 매크로로 지정

%.o : %.c ~ : 폴더 내의 모든 .c 파일들을 컴파일 후 .o파일을 형성함

main : $(OBJS) ~ : 메인 파일을 실행할 수 있도록 컴파일을 완료함

clean : 컴파일한 파일들을 모두 삭제함

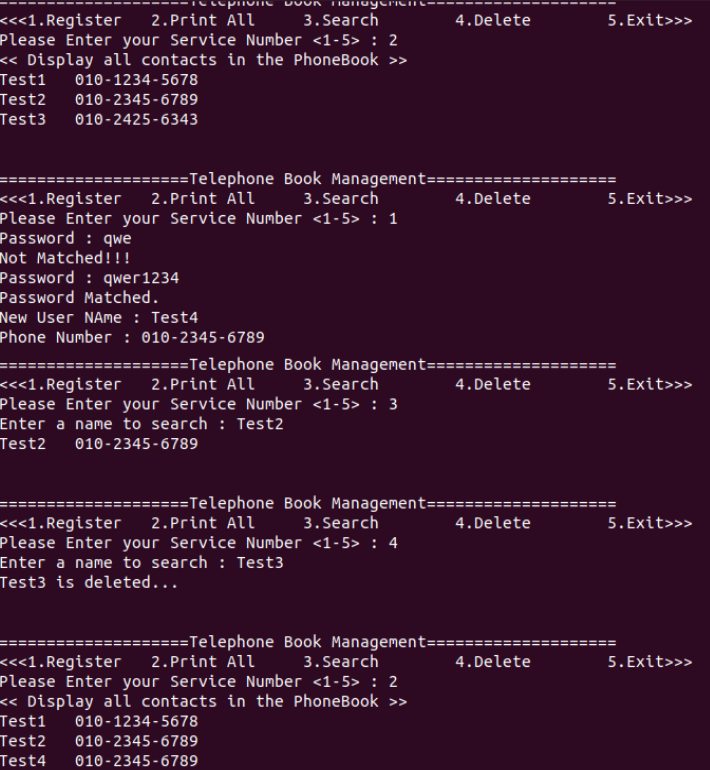
# 실행 방법 설명

1. 사용한 운영체제 및 컴파일러의 종류

완성한 파일은 Linux-Ubuntu 운영체제에서 GNU-GCC를 사용하여 컴파일하였다.

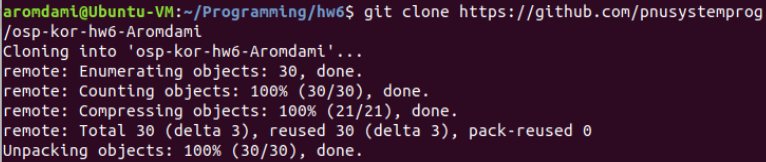
1. 컴파일 방법 및 실행 방법

위의 makefile을 생성 후, 쉘 스크립트에서 make 명령어를 통해 컴파일  
./main 을 입력하여 프로그램을 실행시킨다.



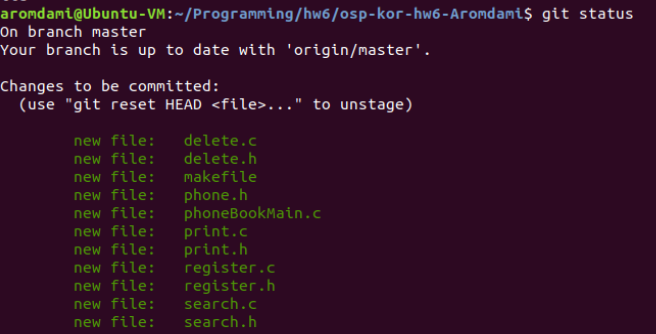
# Github 화면

(1) git 명령어를 통한 파일 전송



1. git cloning을 통한 git 전용 폴더를 생성

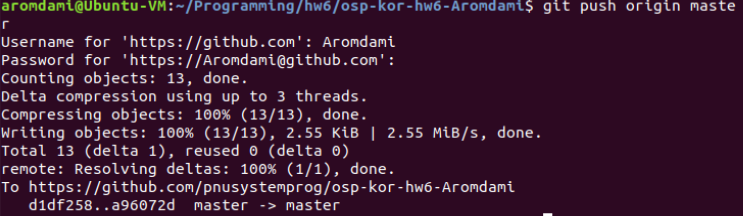




2. 생성한 모든 \*.c파일과 \*.h파일, 그리고 makefile을 git에 add후 등록된 파일들의 상태를 확인

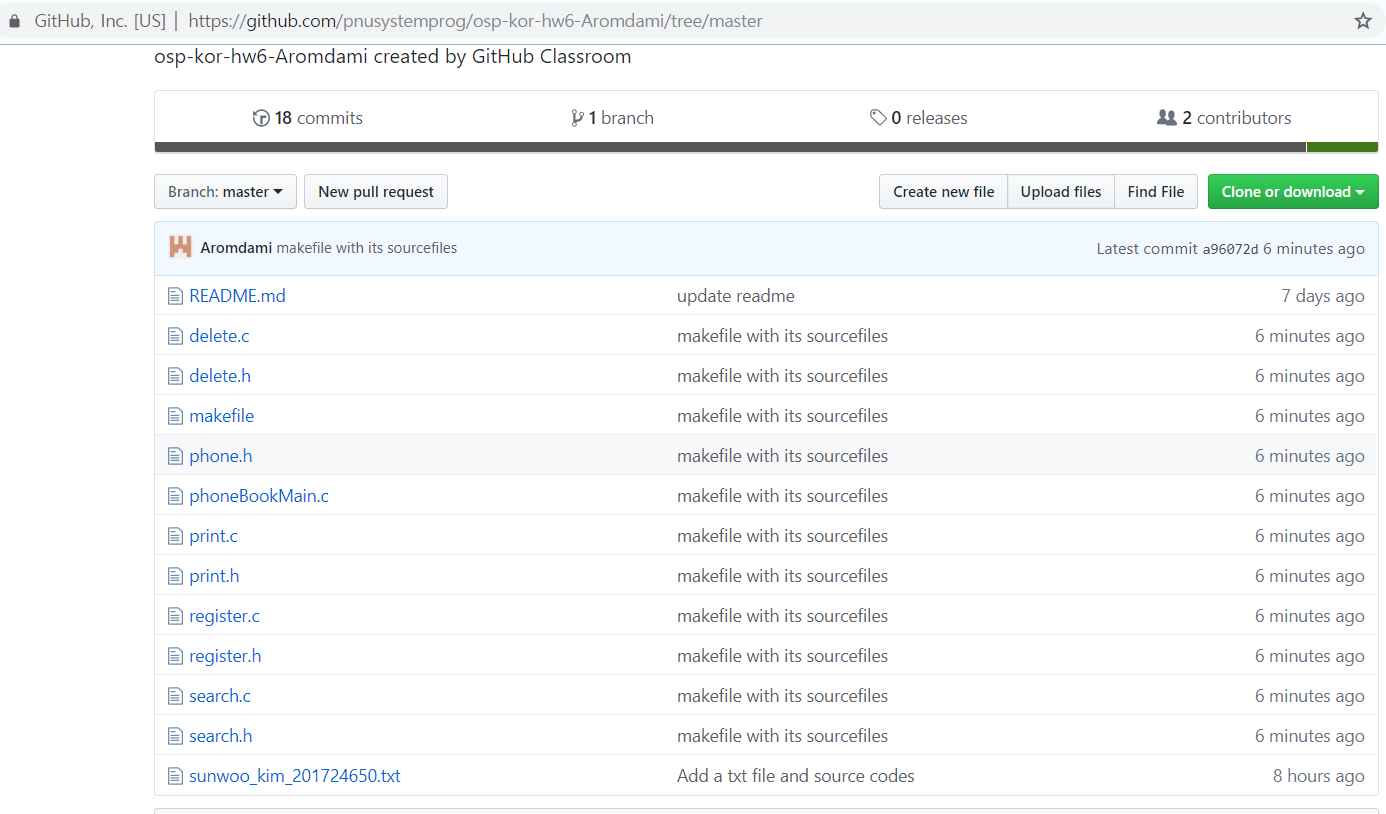


3. 간단한 정보입력과 함께 새로이 등록한 파일들을 git으로 제출함



4. git으로 모든 파일들을 전송 완료 한 상태

(2) Git Hub Repository



리눅스의 쉘에서 작업한 파일들이 github에 등록이 완료된 사진

# 논의 사항

기존에 사용하던 Visual Studio로 코딩을 할 경우, extern 등의 변수나 외부 함수들을 활용 할 때에 컴파일러 측에서 자동으로 링크를 시켜주었기 때문에 이번 과제를 진행하는 동안 링크의 필요성과 그 방법을 다시 확인하게 되었다.

또한, 기존의 컴파일러를 사용했을 경우 #include “register.c”와 같이 .c 소스파일을 포함시켰을 때에도 그 소스를 문제 없이 활용할 수 있었던 반면, 리눅스 환경 및 gnu-gcc를 통한 컴파일을 할 경우엔 오로지 .h 확장자인 헤더파일만 포함하도록 하여 각 함수 원형을 호출하기위해 각 소스파일들과 동일한 명칭의 헤더파일들을 추가하게 되었다.