|  |
| --- |
| HW6. 구조체 배열  **분반: 061**  부산대학교 국제학부  201370105  김덕률  제출일: 2018-05-28 |

# 구현 내용에 대한 설명 (50점)

1. 주요 변수 설명: extern과 static 변수는 반드시 설명함
2. Extern 변수

Phone.h 헤더파일에 struct Contact Phonebook[MAX]와 int size의 전역변수를 선언하였습니다. 이 변수들은 다른 파일에서도 모두 이용하는 변수입니다. 하지만 같은 파일에서 코드를 짜는 것이 아니기 때문에, 다른파일에서 extern 변수를 이용하여 phone.h파일에 있는 전역변수를 이용 할 수 있도록 선언해 줍니다.

Ex) extern struct Contact Phonebook[Max];

extern int size;

1. Static 변수

저는 이 정적 변수를 register.c에서 사용하였습니다. 암호가 틀릴 때 마다 다르게 (twice)와 (three times)를 적어 줘야 하기 때문에, 몇 번째 틀렸는지 이 함수가 호출 될 때 마다 몇 번째 호출되었는 지를 표시하기 위해 static int i 를 이용했습니다.

1. 주요 자료 구조 설명
2. 먼저 교수님께서 올려 놓으신 phone.h 헤더파일을 이용하여 구조체와 전역변수인 size와 phonebook 구조체 함수 선언을 만들었습니다.
3. 각각의 register, printall, search, delete의 함수를 c파일로 만들었습니다. (phone.h 헤더파일에 선언된 전역변수 두개는 extern변수를 이용하여 사용하였습니다.)
4. 그 후 main함수에서 swich문을 이용해 1-5번 항목들을 각각 이용할 수 있도록 만들었습니다.
5. 그리하여 총 phone.h, main.c, register.c, printall.c, search.c, delete.c 의 6가지 파일을 만들어 서로 연관될 수 있도록 컴파일하였습니다.
6. 주요 함수 구현 방법 설명(코드 위 #include 헤더파일과 #define 부분은 생략하겠습니다.)
7. **Phone.h**

struct Contact

{

char name[10];

char number[13];

};

struct Contact Phonebook[MAX]; **// 구조체 전역변수 선언(다른 파일에선 extern변수를 이용해 사용)**

int size; // **size 전역변수 선언(다른 파일에선 extern변수를 이용해 사용)**

1. **Register.c**

extern struct Contact Phonebook[Max]; // **phone.h 헤더파일 속 전역번수 사용**

extern int size; // **phone.h 헤더파일 속 전역변수 사용**

void registerPhoneData(void){

char passward[] = "qw12"; **// 암호 설정**

char check[10]; **// 입력 받을 암호**

static int i = 0; // **이 함수를 다시 불러도 값이 초기화 되지 않는 전역변수의** **성질을 가지는 static을 이용하여 암호 입력 실패를 위한 변수 선언**

printf("Password: ");

scanf("%s", check);

if (strcmp(passward, check)==0) **// 암호가 맞는지 확인**

{

printf("New User Name: ");

scanf("%s", Phonebook[size].name);

printf("PhoneNumber: ");

scanf("%s", Phonebook[size].number);

size = size+1; **// 사이즈를 하나 더함으로써 계속 추가 할 수 있도록**

}

1. **Printall.c**

extern struct Contact Phonebook[Max];

extern int size;

void printAll(void)

{

int i;

printf("<<Dispaly all contacts in the PhoneBook>>\n");

for(i=0;i<size;++i){ **//전역변수인 size만큼 구조체에 저장된 모든 함수 출력.**

printf("%s\t%s\n", Phonebook[i].name, Phonebook[i].number);

}

}

1. **Search.c**

extern struct Contact Phonebook[Max];

extern int size;

void searchByName(void)

{

int i, j = 0;

char name1[20];  **// 이름을 이용하여 찾기 위함**

printf("Enter a name to search: ");

scanf("%s", name1);

for(i=0;i<size;++i){

if(strcmp(name1,Phonebook[i].name) == 0){ **// 이름이 구조체에 있으면**

printf("%s\t%s\n", Phonebook[i].name, Phonebook[i].number);

j += 1; **// j가 0이 아니면 이름을 찾았다는 의미**

}

if(j == 0){ **// j가 0이면 for문에 들어가지 못하였으므로 이름을 못찾음**

printf("Oops! %s is not in the PhoneBook", name1);

}

}

}

1. **Delete.c**

extern struct Contact Phonebook[Max];

extern int size;

void deleteByName(void)

{

int i;

char name1[20];

printf("Enter a name to delete: ");

scanf("%s", name1); **// 이름으로 찾아서 삭제**

for(i=0;i<size;++i){

if(strcmp(name1,Phonebook[i].name) == 0){

break; **// 같은 이름을 찾으면 나온다(size번호를 찾는 것)**

}

}

if(i == size) **//변수 i가 size를 다 돌았으면 못 찾았다는 의미**

{

printf("Oops! %s is not in the PhoneBook\n", name1);

}

else

{

printf("%s is deleted...\n", name1);

for(i=i;i<size-1;i++) **//구조체를 하나만큼 줄여서 위에 덧붙이는 것(삭제)**

{

strcpy(Phonebook[i].name, Phonebook[i+1].name);

strcpy(Phonebook[i].number, Phonebook[i+1].number);

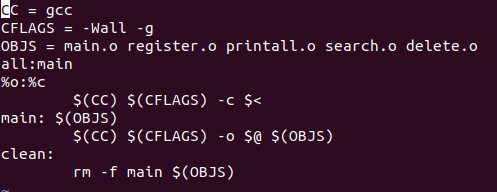
}

}

--size; **// 하나를 삭제했으므로 사이즈를 하나 줄여줌.**

}

1. Makefile 설명
2. 먼저 CC, CFLAGS, OBJS를 이용하여, 긴 문장들을 짧게 이용할 수 있도록 변수를 사용했습니다.
3. 그 다음 gcc-c의 내용을 총 4번 반복해야 하는 것을 suffix Rules를 이용하여 한번만 적어서 내용을 많이 줄였습니다.
4. 그 다음 main파일을 만들기 위해 오브젝트 파일로 변환된 파일들을 명령으로 실행할 수 있는 파일로 변환하였습니다.
5. 마지막으로 phony targets를 이용하여 all 과 clean을 이용해 보다 빠른 명령을 할 수 있도록 만들었습니다.

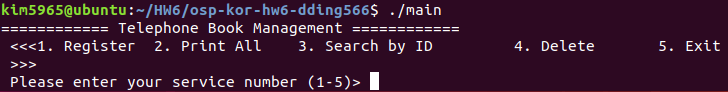


# 실행 방법 설명 (20점)

1. 사용한 운영체제 및 컴파일러의 종류

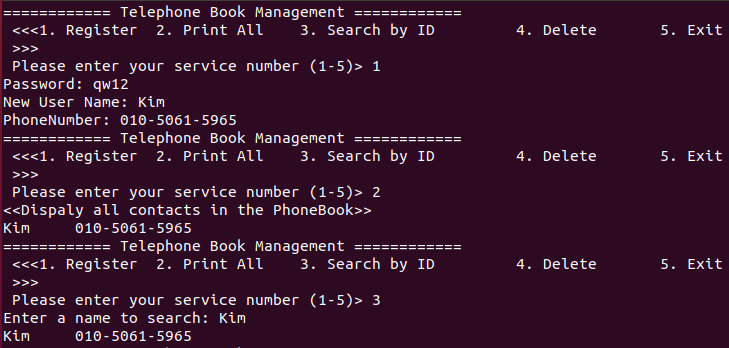
사용한 운영체제는 리눅스 우분투를 이용하였고, 컴파일러는 리눅스 터미널의 gcc c 컴파일러를 이용했습니다.

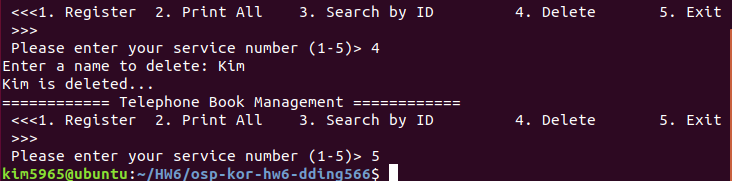
1. 컴파일 방법 및 실행 방법

위에 언급된 make 커맨드를 활용하여 손쉽게 컴파일 했습니다. 

1. 동작을 확인할 수 있는 실행 화면 캡처

1부터 5까지 각각의 동작 확인 캡처화면





# Github 화면 (20점)

(1) cloning, adding, committing, push을 위한 github 명령들을 포함 (동작화면을 실행하고 clear해서 화면캡처를 못해습니다.)

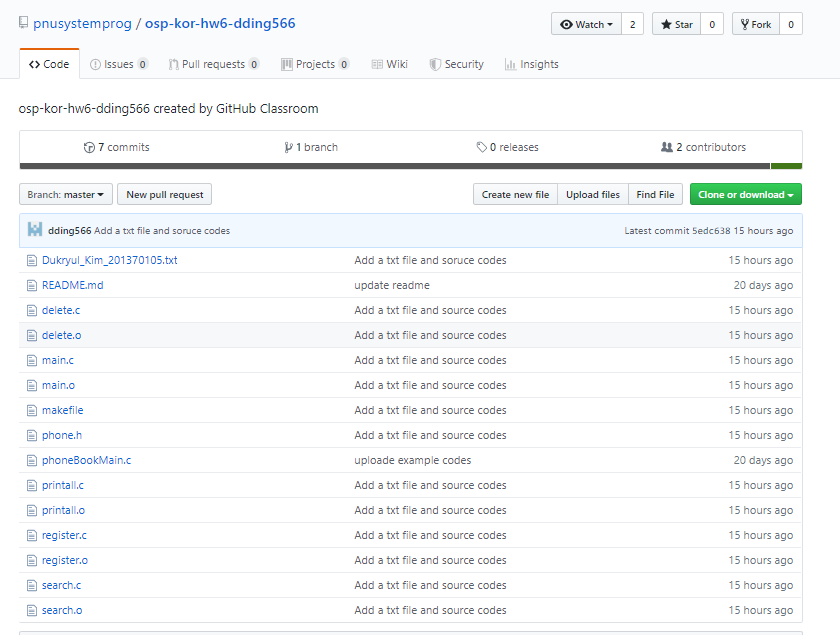
1. git clone https://github.com/pnusystemprog/osp-kor-hw6-dding566.git

2. git add \*.c / git add \*.o / git add makefile

3. git commit -m “Add a txt file and source codes”

4. git push \*.c / git push \*.o / git push makefile

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



# 논의 사항 (10점)

- 숙제를 하면서 제일 어려웠던 점은 사실 c코드를 짜는 것 그 자체였습니다. 사실 숙제를 받고나서부터 시작은 했는데, 제가 완전히 독강이라 누구에게 도움을 청하기도 어려웠습니다. 혼자 코드를 짜는데, 동작 하나하나 모두 어려웠고, 인터넷 검색을 이용하여 혼자 만들었습니다. 특히 static 함수를 이용하는 추가기능의 부분과 이름을 찾아서 삭제하는 부분은 굉장히 많은 시간이 들었습니다. 오히려 git에서는 ppt를 보고 순서대로 처리하니 훨씬 쉬웠습니다. 늦게 제출해서 죄송합니다. 그만큼 저에게 어려웠던 과제였습니다 ㅜㅜ.