

HW6. 함수 포인터와 void 포인터

부산대학교 정보컴퓨터공학부

2020-55565

060분반 여지수

Github ID: duwltn1301@naver.com

제출일: 2020-06-03

1. 구현 내용에 대한 설명 (60점)

- (1) 주요 변수 설명: 함수 포인터 설명
- (2) 주요 자료 구조 설명
- (3) 주요 함수 구현 방법 설명
 - Void 형 포인터 사용 시 동작 원리 자세히 설명
- (4) Makefile 설명

메인 함수가 들어있는 hw6Main.c에서는 stdio.h string.h phone.h 헤더파일을 포함하고 메인에서 쓰일 함수들(registerPhoneData, print, serachByNAme, deleteByName, sort)을 선언하며, 정적 변수 count_service와 공유될 변수 size를 0으로 선언한다.

함수 포인터는 함수의 반환값 자료형을 지정해주고, 함수 포인터 이름 앞에 * (애스터리스크)를 붙인 뒤 () (괄호)로 묶어줍니다. 그리고 다시 괄호를 붙여 함수라는 것을 알려주는데, 이러한 함수 포인터를 main에서 사용할 것이다.

main함수 이전에 함수 포인터를 선언해준다.

```
void (*pFuncs[5])() = {registerPhoneData, print, serachByNAme, deleteByName, sort};
```

pFuncs뒤의 []안의 값에 따라 함수가 호출될 것이다. 예를 들어 [1]이면 pFuncs[1]==print이므로 pFuncs[1]을 불러오면 print 함수를 호출하는 것이다. 따라서 함수포인터를 이용하면 여러 개의 함수를 배열로 관리할 수가 있다.

그래서 이후 main함수에서 1,2,3,4,5,6 서비스 중 -1만큼에 해당되는 리스트 요소를 찾아 그 함수를 호출하도록 구현했다. 6(exit)를 입력하지 않는 이상 서비스는 while문을 통해 계속된다. 1을 입력하면 resgisterPhoneData() 함수가, 2를 입력하면 print() 함수가, 3을 입력하면 searchByName()함수가, 4를 입력하면 deleteByName() 함수가, 5를 입력하면 sort()함수가 호출된다. 마지막으로 서비스가 종료될 때 서비스를 몇 번 이용했는지 service count를 출력한다.

print.c에는 extern int size를 선언하고 printArray(void *arr, int size)함수와 print()함수를 구현했다. print()함수는 printArray함수를 호출하고 호출된 함수가 phonebook내용을 출력하도록 구현했다. printArray함수는 void 포인터 매개변수와 int값을 매개변수로 받는다. arr를 같은 void 포인터 매개변수vp에 복사하고, 구조체를 출력해야하기 때문에 자료형을 구조체Contact* 로 바꾸어야 하고 이를 위해 (Contact*)vp를 한다. i를 0부터 구조체 배열 크기 만큼 (Contact*)vp+i를 하면 주소 값이 나오고, ((Contact*)vp+i)->Name, ((Contact*)vp+i)->PhoneNumber처럼 Name과 PhoneNumber을 참조하면 이름과 전화번호도 출력이 된다.

sort.c에는 extern int size와 printArray, contactCmpr, contactSwap 함수를 선언하고 typedef를 통해 *cmp와 *swap을 형 재정의 해준다.

sort()함수는 우선 sort하기 전 값들을 print해준다. 그래서 printArray(PhoneBook, size)함수를 불러와 프린트 해주고, sort한 후의 값은 sortPhoneBook 함수를 통해 버블정렬한후, 다시 printArray(PhoneBook, size)함수를 통해 프린트한다. sortPhoneBook 함수는 void 포인터, int형 값과 int 형 함수를 매개변수로 받는다. void포인터에는 PhoneBook 구조체 배열을, int형에는 구조체 배열의 크기를, int형 함수에는 contactCmpr과 contactSwap함수가 들어가 동작하게 된다. 버블정렬을 이용해 이중 for문에서 contactCmpr의 반환값이 0보다 크면 contactSwap이 호출되어 swap이 되는 것이다. 그래서 contactCmpr함수는 void 포인터 arr, int i와 j를 매개변수로 입력받고, strcmp 함수를 이용해 ((Contact*)arr+i)->Name ((Contact*)arr+j)->Name을 비교하게끔 하여 그것이 0보다 크면 1을 return 하도록 하여 알파벳순으로 정렬되도록 한다. contactSwap 함수도 void 포인터 arr, int i와 j를 매개변수로 입력받고, 새로운 Contact 구조체 tmp를 선언하고, 구조체를 역참조하여 *((Contact*)arr+i)와 *((Contact*)arr+j)가 교환되도록 한다.

phone.h 헤더파일에는 구조체를 선언하고, #ifndef와 #endif 지시자를 통해서 중복 선언을 막았다.

register.c 에는 extern int size를 선언하고 registerPhoneData()함수를 구현했다. 우선 등록할 수 있는 전화번호는 50개까지 이므로 size가 50이상이면 더이상 추가 할 수 없음을 출력한다. 만약 size가 50보다 작다면, 등록할 기회를 주는 데 우선 패스워드를 입력하도록 한다. 맞춰야 할 password를 qwerty1234로 선언하고 strcmp를 이용해 입력 받은 문자열과 맞춰야 할 password가 같은지 비교하고 같으면 등록을 받도록 하고 다르면 에러 메시지를 띄운다. 3번을 초과하여 틀리면 더이상 패스워드를 입력받지 못하도록 한다. 맞췄다면, size번째의 PhoneBook 구조체에 이름과 전화번호를 저장하고 size는 1 늘린다.

search.c에는 extern int size를 선언하고 searchByName()함수를 구현했다. 찾을 이름을 입력받고, for 문을 이용해 구조체에 접근하여 0번째 구조체 배열부터 그 구조체 배열의 name 과 입력받은 문자열을 비교한다. 같으면 0으로 선언했었던 find 변수를 1로 바꾸고 그 번째수 구조체 배열의 전화번호를 미리 선언했던 빈 phone변수에 저장해둔다. for 문을 다 돌고 나서 find가 1이 되었다면 사용자 입력한 이름이 전화번호부에 있었다는 것이고, find가 변하지 않았다면 사용자가 전화번호부에 없었다는 것이므로, 찾았을 때 찾지 못했을 때를 각각 알맞게 출력한다. 찾았을 때는 이름과 저장한 phone 변수를 출력하고 찾지 못했을 때는 없다고 출력한다.

delete.c에는 extern int size를 선언하고 deleteByName()함수를 구현했다. searchByName()함수와 마찬가지로 구현하는데, 차이점을 얘기하자면, 여기서는 for문을 돌려서 i번째수 구조체 배열의 이름과 사용자가 입력한 문자열이 일치하면, i번째부터 마지막 번째 수까지 한칸씩 앞으로 이동한다. 그럼 i번째 구조체 배열은 i+1 구조체 배열로 대체되면서 없어질 것이고, 맨 뒤에 남은 구조체 배열의 원소는 strcpy를 이용해 "" null로 복사한다. 이름이 삭제되었다는 메시지를 출력하고 하나 삭제 되었으니 size는 1 줄인다. 그리고 searchByName()가 마찬가지로 이름을 찾으면 find는 1로 하여 만일 find가 0이면 전화번호부에 그 이름이 없음을 알리는 메시지를 출력한다.

makefile에 대해 설명하자면,

CC=gcc

-> 매크로 CC를 정의한다. 컴파일러를 세팅해서 환경변수로 지정해주는 것이다. 버전을 바꿔서 컴파일할 때 유용하게 쓸 수 있다.

CFLAGS = -Wall -g -c

-> 매크로 CFLAGS를 정의한다. gcc의 옵션을 추가해주는 용도이다. -g는 디버그 정보를 추가하는 것이다.

INCLUDE = -I.

-> 디렉토리를 추가한다.

LIBS = -L. -lm

-> 링크할 때 필요한 라이브러리를 추가한다.

OBJS = hw6Main.o sort.o register.o delete.o print.o search.o

-> 오브젝트 파일들을 정의해준다.

all : main

-> make 는 makefile 을 순차적으로 읽어서 가장 처음에 나오는 규칙을 수행하게 된다. all 은 더미타겟(dummy target)이 바로 첫 번째 타겟으로써 작용하게 된다. 결과 파일이 많을 때 all 의 의존 관계(dependency)로써 정의해 두면 편리하다.

%.o: %.c

\$(CC) \$(INCLUDE) \$(CFLAGS) \$<

-> 내부적으로 확장자 규칙이 적용되었다. \$(...)은 매크로의 사용을 의미한다. 매크로를 통해 복잡한 것을 간단하게 나타낼 수 있다. \$<는 입력파일로 콜론의 오른쪽에 오는 패턴을 치환한다는 의미를 담고있다. %는 일치하는 확장자를 제외한 파일명을 의미한다. 그래서 %.c를 컴파일하여 같은 이름의 오브젝트 파일을 만들겠다는 것이다. 즉 어떤 .o 파일 이 필요할 경우 이 규칙에 의해서 해당 .c 파일을 찾고 아랫줄의 명령을 실행시켜서 .o 파일을 생성해내는 것이다.

main: \$(OBJS)

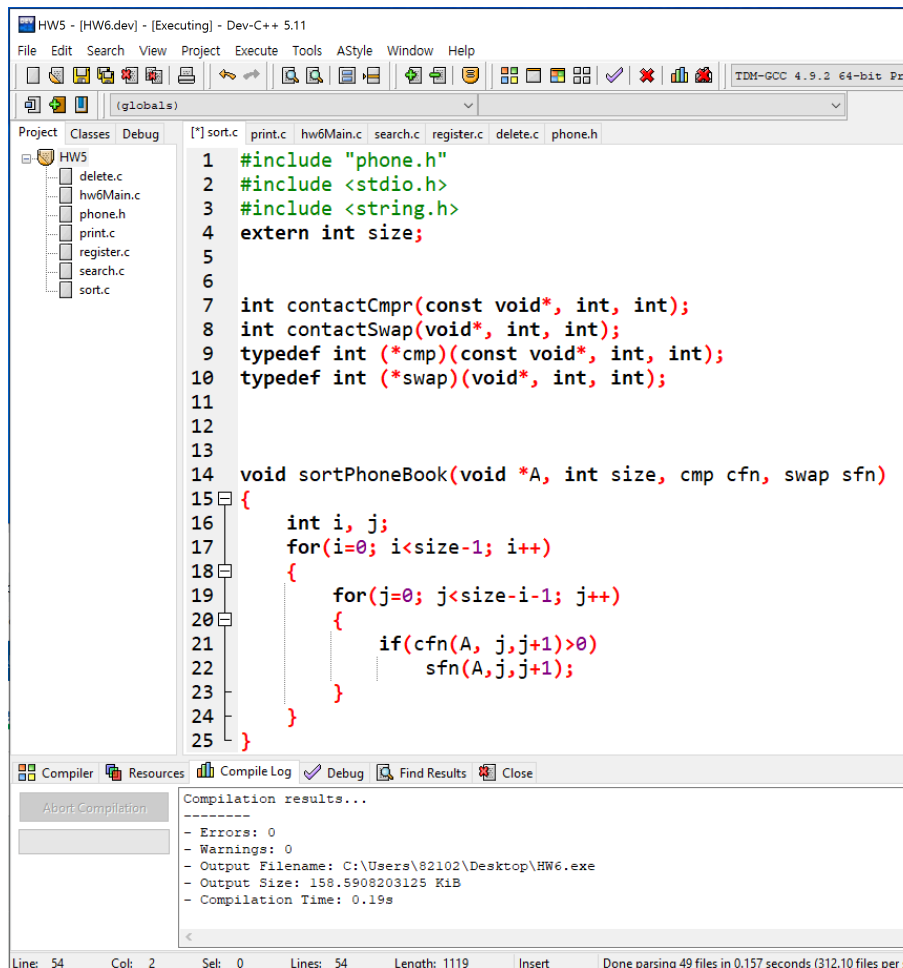
\$(CC) -o main \$(OBJS) \$(LIBS)

clean:

rm -f main \$(OBJS)

2. 실행 방법 설명 (10점)

- (1) 사용한 운영체제 및 컴파일러의 종류 -> 윈도우 운영체제에서 Dev-C++컴파일러를 이용하여 프로그램을 구현했다.
- (2) 컴파일 방법 및 실행 방법 -> Dev-c++ 컴파일로 작성한 코드를 Compile & Run 한다. 컴파일은 전처리-> 컴파일->



어셈블-> 링크 이 네 가지 단계로 진행된다. 우선, 전처리 단계에서는 전처리가 소스코드에서 #으로 시작하는 지시자를 처리한다. 전처리는 #include를 만나면 뒤의 헤더파일을 그 파일의 내용을 순차적으로 삽입 한다. .c 파일로부터 .i 파일이 생성된다. 컴파일 단계에서는 컴파일러가 전처리가 끝난 파일을 컴파일하여 운영체제가 인식할 수 있는 형태의 개체파일을 만든다. .i 파일로부터 .s 파일이 생성된다. 어셈블 단계에서는 어셈블러가 완전한 기계어로 바꾸어준다. .s 파일로부터 .o 파일이 생성된다. 링크 단계에서는 오브젝트 파일에 표준 C 라이브러리, 사용자 라이브러리가 결합되어 main 함수를 호출하여 프로그램의 코드가 실행되도록 한다. .o 파일로부터 .exe 파일이 생성된다.

(3) 동작을 확인할 수 있는 실행 화면 캡처

```

C:\Users\W92102\Desktop\HW6.exe
===== Telephone Book Management =====
<<<1. Register 2. Print All 3. Search by ID 4. Delete 5. Sort 6. Exit >>>
Please enter your service number (1-6)> 1
Password: qwerty1234
New User Name:jisu
PhoneNumber:12345678
Registered...
===== Telephone Book Management =====
<<<1. Register 2. Print All 3. Search by ID 4. Delete 5. Sort 6. Exit >>>
Please enter your service number (1-6)> 1
Password: aimi
Not Matched !!!
Password: qwerty1234
New User Name:aimi
PhoneNumber:33333333
Registered...
===== Telephone Book Management =====
<<<1. Register 2. Print All 3. Search by ID 4. Delete 5. Sort 6. Exit >>>
Please enter your service number (1-6)> 2
Print all contents in the PhoneBook
Addr vp:0000000000408AC0 name:jisu phone:12345678
Addr vp:0000000000408AD7 name:aimi phone:33333333
===== Telephone Book Management =====
<<<1. Register 2. Print All 3. Search by ID 4. Delete 5. Sort 6. Exit >>>
Please enter your service number (1-6)> 5
Sort fuction is called
Before sorting
Addr vp:0000000000408AC0 name:jisu phone:12345678
Addr vp:0000000000408AD7 name:aimi phone:33333333
After sorting
Addr vp:0000000000408AC0 name:aimi phone:33333333
Addr vp:0000000000408AD7 name:jisu phone:12345678
===== Telephone Book Management =====
<<<1. Register 2. Print All 3. Search by ID 4. Delete 5. Sort 6. Exit >>>
Please enter your service number (1-6)> 1
Password: qwerty1234
New User Name:brave
PhoneNumber:23451234
Registered...
===== Telephone Book Management =====
<<<1. Register 2. Print All 3. Search by ID 4. Delete 5. Sort 6. Exit >>>
Please enter your service number (1-6)> 5
Sort fuction is called
Before sorting
Addr vp:0000000000408AC0 name:aimi phone:33333333
Addr vp:0000000000408AD7 name:jisu phone:12345678
Addr vp:0000000000408AEE name:brave phone:23451234
After sorting
Addr vp:0000000000408AC0 name:aimi phone:33333333
Addr vp:0000000000408AD7 name:brave phone:23451234
Addr vp:0000000000408AEE name:jisu phone:12345678
===== Telephone Book Management =====
<<<1. Register 2. Print All 3. Search by ID 4. Delete 5. Sort 6. Exit >>>
Please enter your service number (1-6)> 3
Enter a name to search: jisu
jisu
12345678
===== Telephone Book Management =====
<<<1. Register 2. Print All 3. Search by ID 4. Delete 5. Sort 6. Exit >>>
Please enter your service number (1-6)> 4
Enter a name to delete: brave
brave is deleted...
===== Telephone Book Management =====
<<<1. Register 2. Print All 3. Search by ID 4. Delete 5. Sort 6. Exit >>>
Please enter your service number (1-6)> 5
Sort fuction is called
Before sorting
Addr vp:0000000000408AC0 name:aimi phone:33333333
Addr vp:0000000000408AD7 name:jisu phone:12345678
After sorting
Addr vp:0000000000408AC0 name:aimi phone:33333333
Addr vp:0000000000408AD7 name:jisu phone:12345678
===== Telephone Book Management =====
<<<1. Register 2. Print All 3. Search by ID 4. Delete 5. Sort 6. Exit >>>
Please enter your service number (1-6)> 6
service count: 9
=====
Process exited after 348.2 seconds with return value 0

```

3. Github 화면 (20점)

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

(2) 소스 코드와 makefile을 push한 후, 본인의 Github repository를 스크린 캡처하여 포함
mkdir homework2

cd homework2

로 homework2 디렉토리를 만들고 그 디렉토리로 옮겨가서 cloning 한다.

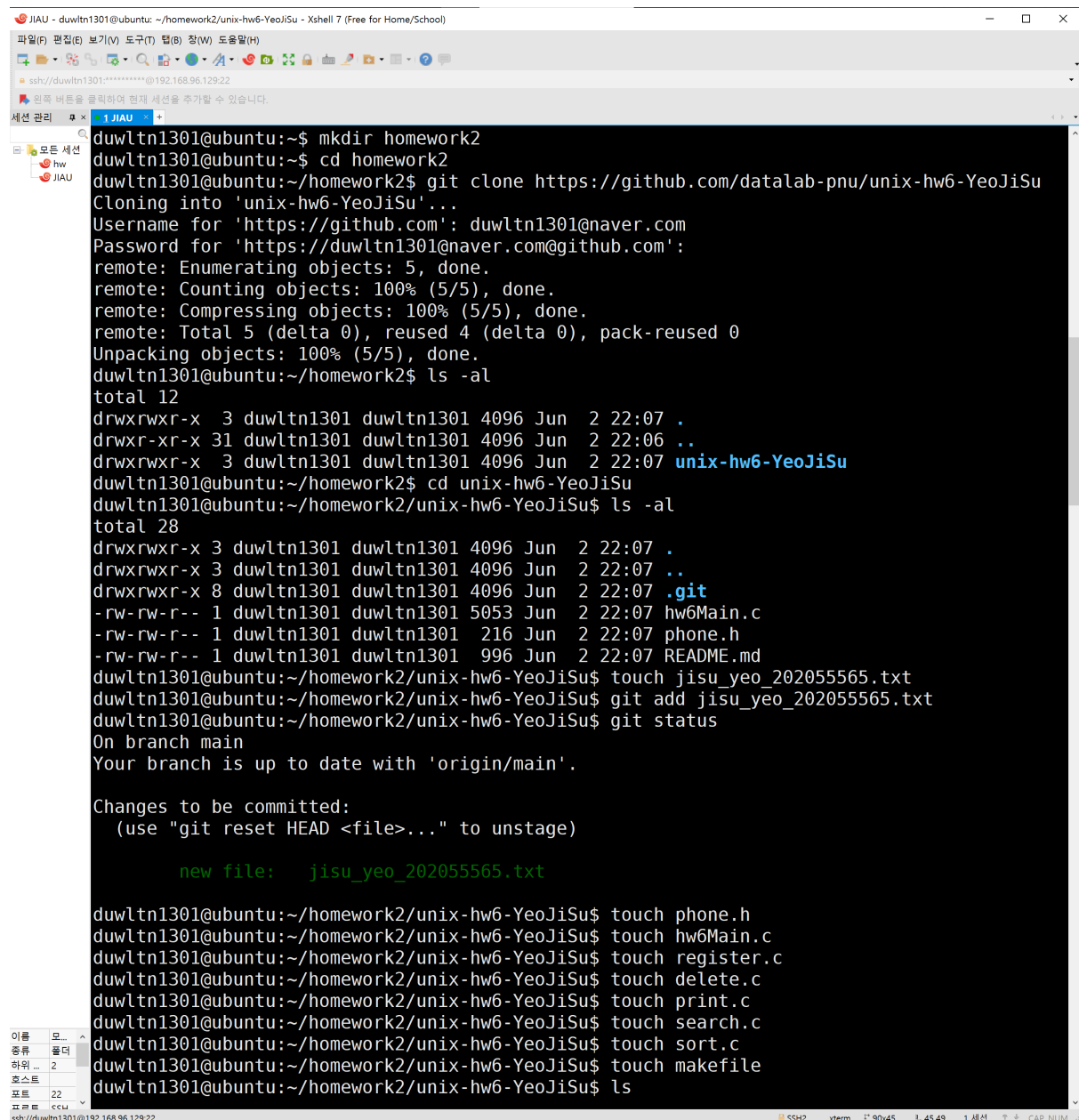
git clone https://github.com/datalab-pnu/unix-hw6-YeoJiSu

로 cloning하고

touch jisu_yeo_202055565.txt

git add jisu_yeo_202055565.txt

하여 txt파일을 add한다.



```
JIAU - duwlt1301@ubuntu: ~/homework2/unix-hw6-YeoJiSu - Xshell 7 (Free for Home/School)
파일(F) 편집(E) 보기(V) 도구(T) 팀(B) 창(W) 도움말(H)
ssh://duwlt1301@192.168.96.129:22
원격 버튼을 클릭하여 현재 세션을 추가할 수 있습니다.
세션 관리
모든 세션
hw
JIAU
duwlt1301@ubuntu:~$ mkdir homework2
duwlt1301@ubuntu:~$ cd homework2
duwlt1301@ubuntu:~/homework2$ git clone https://github.com/datalab-pnu/unix-hw6-YeoJiSu
Cloning into 'unix-hw6-YeoJiSu'...
Username for 'https://github.com': duwlt1301@naver.com
Password for 'https://duwlt1301@naver.com@github.com':
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 5 (delta 0), reused 4 (delta 0), pack-reused 0
Unpacking objects: 100% (5/5), done.
duwlt1301@ubuntu:~/homework2$ ls -al
total 12
drwxrwxr-x 3 duwlt1301 duwlt1301 4096 Jun  2 22:07 .
drwxr-xr-x 31 duwlt1301 duwlt1301 4096 Jun  2 22:06 ..
drwxrwxr-x 3 duwlt1301 duwlt1301 4096 Jun  2 22:07 unix-hw6-YeoJiSu
duwlt1301@ubuntu:~/homework2$ cd unix-hw6-YeoJiSu
duwlt1301@ubuntu:~/homework2/unix-hw6-YeoJiSu$ ls -al
total 28
drwxrwxr-x 3 duwlt1301 duwlt1301 4096 Jun  2 22:07 .
drwxrwxr-x 3 duwlt1301 duwlt1301 4096 Jun  2 22:07 ..
drwxrwxr-x 8 duwlt1301 duwlt1301 4096 Jun  2 22:07 .git
-rw-rw-r-- 1 duwlt1301 duwlt1301 5053 Jun  2 22:07 hw6Main.c
-rw-rw-r-- 1 duwlt1301 duwlt1301 216 Jun  2 22:07 phone.h
-rw-rw-r-- 1 duwlt1301 duwlt1301 996 Jun  2 22:07 README.md
duwlt1301@ubuntu:~/homework2/unix-hw6-YeoJiSu$ touch jisu_yeo_202055565.txt
duwlt1301@ubuntu:~/homework2/unix-hw6-YeoJiSu$ git add jisu_yeo_202055565.txt
duwlt1301@ubuntu:~/homework2/unix-hw6-YeoJiSu$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   jisu_yeo_202055565.txt

duwlt1301@ubuntu:~/homework2/unix-hw6-YeoJiSu$ touch phone.h
duwlt1301@ubuntu:~/homework2/unix-hw6-YeoJiSu$ touch hw6Main.c
duwlt1301@ubuntu:~/homework2/unix-hw6-YeoJiSu$ touch register.c
duwlt1301@ubuntu:~/homework2/unix-hw6-YeoJiSu$ touch delete.c
duwlt1301@ubuntu:~/homework2/unix-hw6-YeoJiSu$ touch print.c
duwlt1301@ubuntu:~/homework2/unix-hw6-YeoJiSu$ touch search.c
duwlt1301@ubuntu:~/homework2/unix-hw6-YeoJiSu$ touch sort.c
duwlt1301@ubuntu:~/homework2/unix-hw6-YeoJiSu$ touch makefile
duwlt1301@ubuntu:~/homework2/unix-hw6-YeoJiSu$ ls
```

touch 명령어로 헤더파일, c파일, makefile을 생성하고 이를 모두 git add 했다.

```
duwltn1301@ubuntu:~/homework2/unix-hw6-YeoJiSu$ ls
delete.c  jisu_yeo_202055565.txt  phone.h  README.md  search.c
hw6Main.c  makefile                print.c  register.c  sort.c
duwltn1301@ubuntu:~/homework2/unix-hw6-YeoJiSu$ git add *.c
duwltn1301@ubuntu:~/homework2/unix-hw6-YeoJiSu$ git add *.h
duwltn1301@ubuntu:~/homework2/unix-hw6-YeoJiSu$ git add makefile
duwltn1301@ubuntu:~/homework2/unix-hw6-YeoJiSu$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   delete.c
    new file:   jisu_yeo_202055565.txt
    new file:   makefile
    new file:   print.c
    new file:   register.c
    new file:   search.c
    new file:   sort.c

duwltn1301@ubuntu:~/homework2/unix-hw6-YeoJiSu$ git commit -m "Add a txt file and source codes"
[main 3cf729e] Add a txt file and source codes
 7 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 delete.c
 create mode 100644 jisu_yeo_202055565.txt
 create mode 100644 makefile
 create mode 100644 print.c
 create mode 100644 register.c
 create mode 100644 search.c
 create mode 100644 sort.c
```

touch 명령어로 생성한 파일들에는 아무 내용이 없기에

내가 기존에 작성했던 헤더파일, c파일, makefile을 /home/duwltn1301/homewok/unix-hw6-YeoJiSu 디렉토리로 옮기고

git commit -m "Add a txt file and source codes" 명령을 했다. 그 후

git push

```
duwltn1301@ubuntu:~/homework2/unix-hw6-YeoJiSu$ git commit -m "Add a txt file and source codes"
On branch main
Your branch is ahead of 'origin/main' by 2 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
duwltn1301@ubuntu:~/homework2/unix-hw6-YeoJiSu$ git push
Username for 'https://github.com': duwltn1301@naver.com
Password for 'https://duwltn1301@naver.com@github.com':
Counting objects: 12, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (11/11), done.
Writing objects: 100% (12/12), 2.89 KiB | 493.00 KiB/s, done.
Total 12 (delta 3), reused 0 (delta 0)
remote: Resolving deltas: 100% (3/3), completed with 1 local object.
To https://github.com/datalab-pnu/unix-hw6-YeoJiSu
   8cb79d7..003c65d  main -> main
duwltn1301@ubuntu:~/homework2/unix-hw6-YeoJiSu$
```

를 하면 파일들이 모두 잘 업로드 되어 있다.

4. 논의 사항 (10점)

- 이번 과제를 하면서 함수 포인터를 처음 접해보고 사용해보았다. 함수 포인터를 왜 굳이 써야하는지 의문이 생겼었는데, 콜백(callback) 매커니즘을 구현할 때 함수를 또 다른 함수의 인자(argument)로 넘겨줄 수도 있고, 여러 개의 함수를 배열로 관리하고자 할 때 사용할 수 있어서 더 편리하다는 것을 깨달았다. 새롭게 알게된 사실은 함수 포인터도 포인터이기 때문에, 일반적인 포인터와 마찬가지로 메모리 주소를 가리키지만, 일반적인 포인터와 달리, 함수 포인터는 데이터가 아닌 코드의 위치를 가리킨다는 점이다. 마치 배열을 가리키는 포인터가 배열의 시작 부분을 가리키는 것과 같이, 함수 포인터도 코드를 가리킬 때 코드의 시작 부분을 가리켰다. 그리고 이러한 함수 포인터를 통해 메모리를 할당하거나 회수하는 것이 불가능해서 함수 포인터를 대상으로 malloc(), free() 함수는 사용할 수 없다는 것을 알게 되었다.