

리눅스시스템 Lab11

분반:001

학과:컴퓨터과학전공

학번:2312282

이름: 임다희

1. 프로그램 작성 : sort.c

1) p3의 기능 구현을 위해 sort.c를 어떻게 프로그래밍 하였는지 간단하게 설명한다. 단, 설명에는 copy.c 파일의 기능을 어떻게 활용하였는지를 포함하여야 한다. (기능 구현이 100% 안 되었더라도 구현한 범위까지 설명을 작성해 주세요.)

-다음과 같이 sort.c를 작성하였다.

//001 컴퓨터과학전공 2312282 임다희

```
#include <stdio.h>
#include <string.h>
#include "copy.h"

char line[MAXLINE];
char lines[MAXLINE][MAXLINE];

int main(){
    int line_count=0;
    printf("Enter text:");
    while (fgets(lines[line_count],MAXLINE,stdin)!=NULL){
        line_count++;
    }

    for(int i=0; i<line_count-1; i++){
        for(int j=1; j<line_count-i; j++){
            if(strlen(lines[j-1])<strlen(lines[j])){
                copy(lines[j],line);
                copy(lines[j-1],lines[j]);
                copy(line,lines[j-1]);
            }
        }
        printf("-----Sorted Result-----");
        for(int i=0; i<line_count; i++){
            for(int j=0; j<strlen(lines[i]); j++){
                printf("%c",lines[i][j]);
            }
        }
        return 0;
    }
}
```

-설명

copy.h에서 지정한 MAXLINE값(100) 만큼의 문자열을 저장할 수 있으며, 저장되는 문자열의 최대 길이 또한 MAXLINE과 같은 2차원 배열 lines를 만든다. (100행 100열의 2차원 배열. 각 행마다 문자열 하나의 정보가 저장된다)

사용자에게 문자열 여러 개를 입력받아 하나의 문자열을 하나의 행에 저장하고, 문자열을 하나 입력받을 때마다 저장된 문자열의 개수를 저장하는 정수타입 변수 line_count의 숫자를 1씩 증가시킨다.

문자열 입력이 종료되면 버블 정렬을 이용해 lines 내부 문자열들을 순차적으로 방문하며 길이가 짧은 순으로 정렬한다. 버블 정렬은 안정 정렬로, 정렬 기준이 되는 문자열 길이의 값이 같은 요소들은 정렬 후에도 입력된 순서를 유지한다. copy.c에서 구현한 문자열 복사 기능을 이용해 앞 뒤 문자열의 순서가 정렬되어야 하는 경우 문자열의 값을 서로 교환하는 기능을 구현하였다.

2. 컴파일 및 실행

1) 컴파일 후, ./[실행파일이름] 형식으로 파일을 실행한 3개의 예시를 캡처한다. 단, 설명에는 p5와 같이 줄의 길이가 같으면, 먼저 입력한 줄을 먼저 출력하는 예시를 반드시 포함하여야 한다.

-예시 1

```
u2312282@dahee-VirtualBox: ~/linux/chap11
u2312282@dahee-VirtualBox:~/linux/chap11$ ./sort
Enter text:
linux
system
hi
everyone
-----Sorted Result-----
hi
linux
system
everyone
u2312282@dahee-VirtualBox:~/linux/chap11$
```

-예시 2 (p5의 예시, 줄의 길이가 같으면 먼저 입력한 줄이 먼저 출력된다.)

```
u2312282@dahee-VirtualBox:~/linux/chap11$ ./sort
Enter text:
freshman
sophomore
junior
senior
-----Sorted Result-----
junior
senior
freshman
sophomore
```

-예시 3 (줄의 길이가 같으면 먼저 입력한 줄이 먼저 출력된다.)

```
u2312282@dahee-VirtualBox:~/linux/chap11$ ./sort
Enter text:
this
is
a
test
b
-----Sorted Result-----
a
b
is
this
test
```

3. gdb 사용 (디버깅)

1) gdb 디버거를 사용하여 p6의 (1)~(4)를 차례대로 수행한 터미널 창을 캡처하고, 4개의 각 기능을 위해 어떠한 gdb 옵션을 사용하였는지를 간단히 설명한다.

```
u2312282@dahee-VirtualBox: ~/linux/chap11
(gdb) b copy
Breakpoint 2 at 0x1475: file copy.c, line 6.
(gdb) r
Starting program: /home/u2312282/linux/chap11/sort
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Enter text:
linux
system
hi
everyone

Breakpoint 2, copy (from=0x5555555581ec <lines+300> "hi\n", to=0x555555558040 <line> "") at copy.c:6
6          i=0;
```

-(1) 설명

(gdb) b copy 를 통해(b 옵션: 정지점 설정) copy 함수의 시작 부분에 정지점을 설정한다.
(gdb) r 를 통해 프로그램을 실행시킨다. 함수 실행 도중 설정된 정지점을 만나면 함수 실행이 일시정지 된다.

```
(gdb) p from
$1 = 0x5555555581ec <lines+300> "hi\n"
(gdb) p to
$2 = 0x555555558040 <line> ""
```

-(2)설명

변수값을 프린트하는 p 옵션을 사용한다. (gdb) p from를 통해 정지한 시점에서 copy 함수의 매개 변수 from을, (gdb) p to 를 통해 copy 함수의 매개변수 to를 출력한다.
system과 hi의 순서를 바꾸기 위해 copy 함수가 실행되는 도중에 정지되었다. hi를 line으로 복사하는 과정을 나타낸다.

```
(gdb) n
7          while ((to[i]=from[i]) !='\0')
```

-(3)설명

멈춘 시점에서 다음 줄을 실행하고 재차 멈추는 n 옵션을 사용해 한 줄씩 실행한다. copy 함수의 정지점 다음 줄인 while문이 실행된다.

```
(gdb) c
Continuing.
```

```
u2312282@dahee-VirtualBox: ~/linux/chap11

Breakpoint 2, copy (from=0x55555558188 <lines+200> "system\n", to=0x555555581e
c <lines+300> "hi\n") at copy.c:6
6          i=0;
(gdb) p from
$3 = 0x55555558188 <lines+200> "system\n"
(gdb) p to
$4 = 0x555555581ec <lines+300> "hi\n"
(gdb) n
7          while ((to[i]=from[i]) !='\0')
(gdb) c
Continuing.
```

```
Breakpoint 2, copy (from=0x55555558040 <line> "hi\n", to=0x55555558188 <lines+
200> "system\n") at copy.c:6
6          i=0;
(gdb) p from
$5 = 0x55555558040 <line> "hi\n"
(gdb) p to
$6 = 0x55555558188 <lines+200> "system\n"
(gdb) n
7          while ((to[i]=from[i]) !='\0')
(gdb) c
Continuing.
```

```
u2312282@dahee-VirtualBox: ~/linux/chap11

Breakpoint 2, copy (from=0x55555558188 <lines+200> "hi\n", to=0x55555558040 <l
ine> "hi\n") at copy.c:6
6          i=0;
(gdb) p from
$7 = 0x55555558188 <lines+200> "hi\n"
(gdb) p to
$8 = 0x55555558040 <line> "hi\n"
(gdb) n
7          while ((to[i]=from[i]) !='\0')
(gdb) c
Continuing.
```

```
Breakpoint 2, copy (from=0x55555558124 <lines+100> "linux\n", to=0x55555558188
<lines+200> "hi\n") at copy.c:6
6          i=0;
(gdb) p from
$9 = 0x55555558124 <lines+100> "linux\n"
(gdb) p to
$10 = 0x55555558188 <lines+200> "hi\n"
(gdb) n
7          while ((to[i]=from[i]) !='\0')
(gdb) c
Continuing.
```

```

Breakpoint 2, copy (from=0x55555558040 <line> "hi\n", to=0x55555558124 <lines+
100> "linux\n") at copy.c:6
6         i=0;
(gdb) p from
$11 = 0x55555558040 <line> "hi\n"
(gdb) p to
$12 = 0x55555558124 <lines+100> "linux\n"
(gdb) n
7         while ((to[i]=from[i]) != '\0')
(gdb) c
Continuing.
-----Sorted Result-----
hi
linux
system
everyone
[Inferior 1 (process 3274) exited normally]
(gdb)

```

-(4)설명

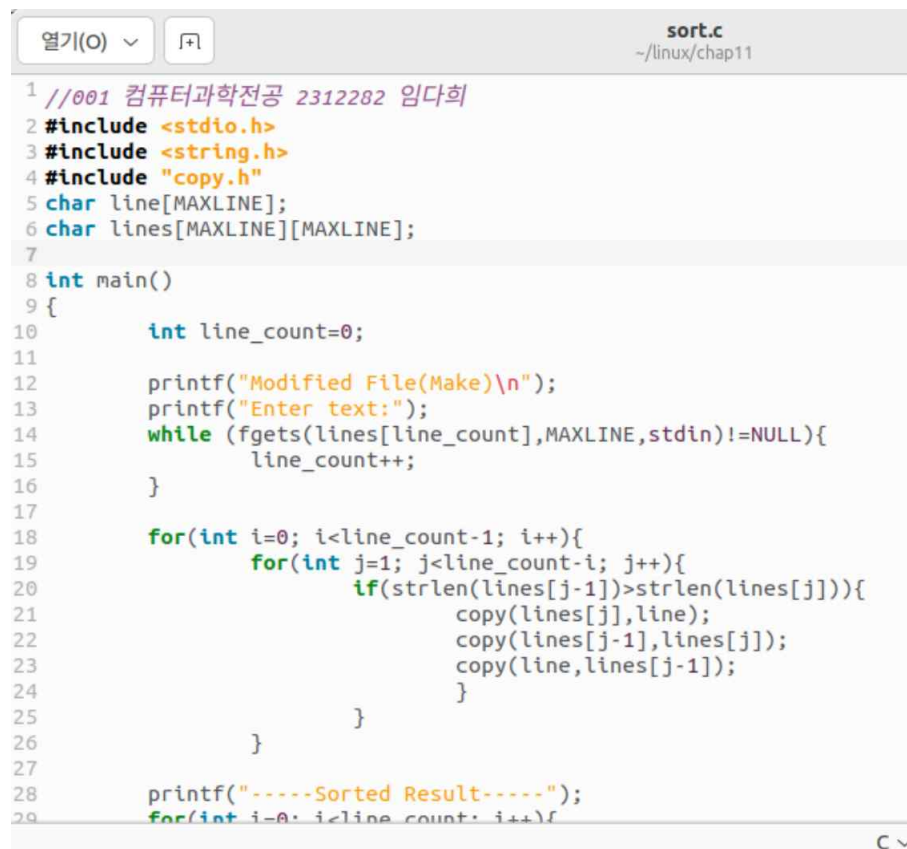
정지점을 만날 때까지 계속 실행하도록 하는 c 명령어를 사용해 계속 실행하고, (2),(3)의 과정을 반복하여 정지점을 만날 때마다 from과 to 값을 출력한다. 실행되는 도중 copy문 내의 정지점을 만나면 정지가 이루어지게 된다. system과 hi의 순서를 바꾸기 위한 두 번째 copy문, 세 번째 copy문, linux와 hi의 순서를 바꾸기 위해 copy문 세 번이 더 실행되어 5번의 정지점을 더 만난 후 마지막으로 c 명령어를 실행하면 정상적으로 프로그램이 실행 완료된다.

2) gdb 디버거 사용을 위한 컴파일 명령을 작성한다.

```
$ gcc -g -o sort sort.c copy.c copy.h
```

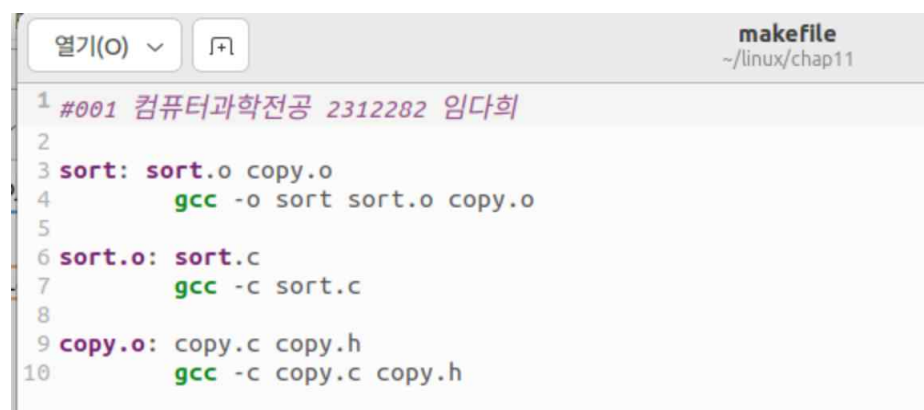
4. make 사용

1) makefile 내용을 작성한 뒤 이를 이용하여 컴파일하고, make 명령어를 수행하는 진행 과정을 설명한다.



```
1 //001 컴퓨터과학전공 2312282 임다희
2 #include <stdio.h>
3 #include <string.h>
4 #include "copy.h"
5 char line[MAXLINE];
6 char lines[MAXLINE][MAXLINE];
7
8 int main()
9 {
10     int line_count=0;
11
12     printf("Modified File(Make)\n");
13     printf("Enter text:");
14     while (fgets(lines[line_count],MAXLINE,stdin)!=NULL){
15         line_count++;
16     }
17
18     for(int i=0; i<line_count-1; i++){
19         for(int j=1; j<line_count-i; j++){
20             if(strlen(lines[j-1])>strlen(lines[j])){
21                 copy(lines[j],line);
22                 copy(lines[j-1],lines[j]);
23                 copy(line,lines[j-1]);
24             }
25         }
26     }
27
28     printf("-----Sorted Result-----");
29     for(int i=0; i<line_count; i++){
```

sort.c 파일에 printf("Modified File(Make)\n"); 구문을 추가한다.



```
1 #001 컴퓨터과학전공 2312282 임다희
2
3 sort: sort.o copy.o
4     gcc -o sort sort.o copy.o
5
6 sort.o: sort.c
7     gcc -c sort.c
8
9 copy.o: copy.c copy.h
10    gcc -c copy.c copy.h
```

컴파일할 파일들이 있는 위치에 makefile을 다음과 같이 작성한다. sort의 의존 리스트에 sort.o와 copy.o를 넣어 gcc -o 옵션으로 컴파일하고, sort.o의 의존 리스트에 sort.c를 넣어 gcc -c 옵션으로 컴파일, copy.o의 의존 리스트에 copy.c, copy.h를 넣어 gcc -c 옵션으로 컴파일하도록 한다.

```
u2312282@dahee-VirtualBox:~/linux/chap11$ make
gcc -c sort.c
gcc -c copy.c copy.h
gcc -o sort sort.o copy.o
```

make 명령어를 실행하면 작성한 makefile을 이용해 실행 파일을 빌드할 수 있다.

```
u2312282@dahee-VirtualBox:~/linux/chap11$ ./sort
Modified File(Make)
Enter text:
hello
hi
make system
practice
-----Sorted Result-----
hi
hello
practice
make system
```

sort를 실행하면 다음과 같이 sort.c파일의 변경사항이 반영되어 컴파일된 것을 볼 수 있다.

5. 이클립스 통합개발환경

1) eclipse로 1번에서 작성한 프로그램을 빌드 및 실행시키고 화면을 캡처한다.

