컴퓨터공학설계및실험I 2주차 실습 결과보고서

20181202 김수미

1. 실습 결과화면 첨부

1) 실습1 : 간단하게 변환한 Makefile

```
 1 cc = gcc
 2 target = animal
 3 objects = dog.o blackcow.o turtle.o main.o
 4
 5 $(target): $(objects)
 6         $(cc) -o $(target) $(objects)
 7
 8 $(objects) : animal.h
 9
10 .PHONY : clean
11 clean :
12         rm $(target) $(objects)
```

2) 실습2 : 디버깅 과정 화면 캡처

```
cse20181202@cspro:~/week2_ex/ex_2$ gcc -g code.c -o code
code.c:3:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
 main(void)
 ^
```

```
cse20181202@cspro:~/week2_ex/ex_2$ gdb code
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from code...done.
(gdb)
```

```
(gdb) break 10
Breakpoint 1 at 0x400557: file code.c, line 10.
(gdb) run
Starting program: /sogang/under/cse20181202/week2_ex/ex_2/code

Breakpoint 1, main () at code.c:10
10              printf("num is %f \n", num);
```

```
(gdb) next
num is 0.000000
8           for (i=0; i<5; i++){
(gdb)
9               num=i/2 + i;
(gdb)

Breakpoint 1, main () at code.c:10
10              printf("num is %f \n", num);
(gdb)
num is 1.000000
8           for (i=0; i<5; i++){
(gdb)
9               num=i/2 + i;
(gdb)

Breakpoint 1, main () at code.c:10
10              printf("num is %f \n", num);
(gdb)
num is 3.000000
8           for (i=0; i<5; i++){
(gdb)
9               num=i/2 + i;
(gdb)

Breakpoint 1, main () at code.c:10
10              printf("num is %f \n", num);
(gdb)
num is 4.000000
8           for (i=0; i<5; i++){
(gdb)
9               num=i/2 + i;
(gdb)

Breakpoint 1, main () at code.c:10
10              printf("num is %f \n", num);
(gdb)
num is 6.000000
8           for (i=0; i<5; i++){
(gdb)
12      }
```

```
cse20181202@cspro:~/week2_ex/ex_2$ ./code
num is 0.000000
num is 1.000000
num is 3.000000
num is 4.000000
num is 6.000000
```

3) 실습3 : 주어진 textfile.txt를 넣고 프로그램을 돌린 결과 (LIMIT=72)

```
cse20181202@cspro:~/week2_ex/ex_3_1$ make
gcc -W -g -c -o main.o main.c
gcc -W -g -c -o String_Manipulation.o String_Manipulation.c
gcc -W -g -c -o Output.o Output.c
gcc -W -g -o fmt main.o String_Manipulation.o Output.o
cse20181202@cspro:~/week2_ex/ex_3_1$ ./fmt textfile.txt
Here is the perfect system for cleaning your room. First move all of the
items that do not have a proper place to the center of the room. Get rid
of at least five things that you have not used within the last year.Take
out all of the trash, and place all of the dirty dishes in the kitchen
sink. Now find a location for each of the items you had placed in the
center of the room. For any remaining items, see if you can squeeze them
in under your bed or stuff them into the back of your closet. See, that
was easy
```

(Remove_Blanks_At_The_End 함수 코드 내용 캡쳐)

```
#include "Header.h"

void Remove_Blanks_At_The_End( char *line ) {
        int i, k, newline_flag = 0;

        // 전체 문장에 대하여 line[k] 가 줄바꿈이면 flag를 1로 set, '\0'이면 ~~
        for ( k = 0; ; k++ ) {
                if ( line[k] == '\n') {
                        newline_flag = 1;
                        break;
                }
                else if ( line[k] == '\0') {
                        break;
                }
        }

        // 전체 문장에 대하여 line[i]가 space가 아니라면 break,
        for ( i = k-1; i >= 0; i-- ) {
                if ( line[i] != ' ') {
                        break;
                }
        }

        // flag가 1일때 문장의 마지막은 줄바꿈&space, flag가 1이 아니라면 '\0'
        if ( newline_flag == 1 ) {
                line[i+1] = '\n';
                line[i+2] = '\0';
        }
        else {
                line[i+1]= '\0';
        }
}
```

2. remove_blanks_at_the_end 함수 구현 설명 (코드 붙여넣기 및 주석 작성)

```
void Remove_Blanks_At_The_End( char *line ) {
    int i, k, newline_flag = 0;
    // 문장이 끝날 때까지 k 값을 증가시키며 문장을 훑는다.
    // 전체 문장에 대하여 line[k] 가 줄바꿈이면 newline_flag 값을 1로 바꾼다.
    // line[k] 가 '₩0'인 경우 아무런 변화 없음. (newline_flag 값 그대로 0)
    // 아래 루프가 끝나면 k는 문장의 가장 마지막 부분 바로 다음을 가리키는 index가 됨
    for ( k = 0; ; k++ ) {
            if ( line[k] == '₩n') {
                    newline_flag = 1;
                    break; }
            else if ( line[k] == '₩0') { break; }
    }
    // 따라서 k 값에서 -1을 하여 사용한다.
    // 문장의 가장 마지막 부분에서 처음까지 훑어 가면서 '(뒤에서부터)글자가 시작되는
    부분'이 어디인지 찾는다.
    // 중간에 찾으면 (line[i]가 space(' ')가 아니라면) break. 이 때 i 값은 '(뒤에서부터)글자가
    시작되는 부분'을 가리키는 index 값이다.
    // 문장의 가장 끝에서 가장 처음으로 갈 때 까지 글자가 하나도 없다면 이 루프가
    끝났을 때 i 값은 -1 이 된다.
    for ( i = k-1; i >= 0; i-- ) {
            if ( line[i] != ' ') { break; }
    }
    // newline_flag 값이 1일때 문장의 마지막, 즉 line[i+1]의 위치에 줄바꿈('₩n')과 그 다음
    위치 line[i+2] 에 '₩0' 표시를 넣어주고, newline_flag 값이 0이라면 line[i+1] 위치에 '₩0'
    표시만 넣는다.
    if ( newline_flag == 1 ) {
            line[i+1] = '₩n';
            line[i+2] = '₩0'; }
    else { line[i+1]= '₩0'; }}
```

3. 실습1 makefile의 한 줄 한 줄 의미를 설명

```
1 cc = gcc
2 target = animal
3 objects = dog.o blackcow.o turtle.o main.o
4
5 $(target): $(objects)
6         $(cc) -o $(target) $(objects)
7
8 $(objects) : animal.h
9
10 .PHONY : clean
11 clean :
12         rm $(target) $(objects)
```

1 cc = gcc

>> 환경변수 cc는 gcc를 의미한다. gcc를 이용하여 컴파일 하겠다는 뜻이다.

2 target = animal

>> 환경변수 target은 Makefile을 실행시켰을 때 만들어지는 파일의 이름이다.

3 objects = dog.o blackcow.o turtle.o main.o

>> 환경변수 objects는 위의 네가지 .o 파일들을 가리킨다. Make에서 사용되는 object file 들의 모음이다.

5 $(target): $(objects)

6 $(cc) -o $(target) $(objects)

>> 환경변수 target이 저 자리에 놓이게 됨으로써 이 Makefile의 최종 목표가 되는 파일의 이름은 animal이 된다. 최종 목표 파일 이름과 object files, 컴파일러 등을 위와 같은 형식으로 불러옴으로써 기존 Makefile에서와 동일하게 명령을 수행한다.

8 $(objects) : animal.h

>> objects들의 함수는 animal.h 헤더파일 안에서 호출되거나(main외), 호출하여(main) 컴파일된다.

10 .PHONY : clean

>> 아랫줄의 clean 명령을 수행할 때, 파일 중에 clean이라는 이름을 가진 것이 있는 등의 예상치 못한 상황으로 오류가 발생하는 것을 방지하기 위해 PHONY를 사용해주는 것이 좋다.

11 clean :

12 rm $(target) $(objects)

>> 현재 디렉토리에 있는 모든 target 파일과 object 파일들을 지운다.

4. make의 주요 옵션 정리

-C dir : Makefile을 계속 읽지 말고 dir로 이동시키는 옵션. 순환 make에 사용된다.

-d : Makefile을 수행하면서 각종 정보를 모조리 출력해 준다.

결과를 파일로 저장해서 읽어보면 make 의 동작을 이해하는데 도움이 된다.

-h : 옵션에 관한 도움말을 출력한다. (-help)

-f file : file 에 해당하는 파일을 Makefile로써 취급한다. (-file)

-r : 내장하고 있는 각종 규칙(Suffix rule 등)을 없는 것으로(-no-builtin-rules) 간주한다.

즉 사용자가 규칙을 새롭게 정의해 주어야 한다.

-t : 파일의 생성 날짜를 현재 시간으로 갱신한다. (-touch)

-v : make의 버전을 출력한다. (-version)

-p : make에서 내부적으로 세팅되어 있는 값들을 출력한다. (-print-data-base)

-k : make는 에러가 발생하면 도중에 실행을 포기하게 되는데 -k 는 에러가 나더라도 멈추지 말고 계속 진행시키라는 명령이다. (-keep-going)