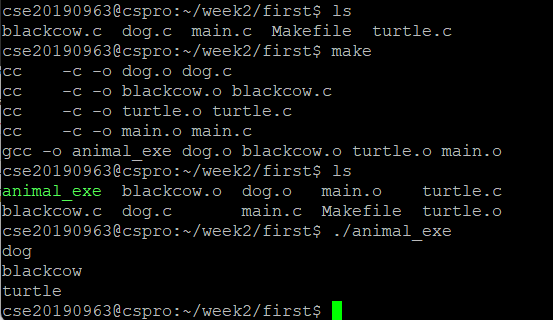
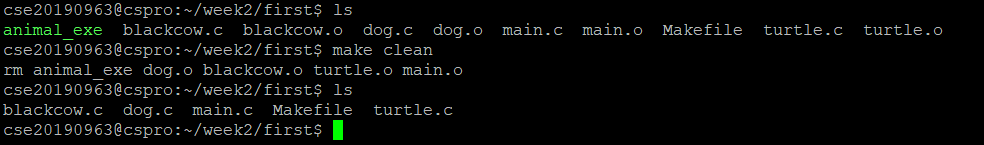
2주차 결과보고서

전공: 경영학과 학년: 3학년 학번: 20190963 이름: 한다현

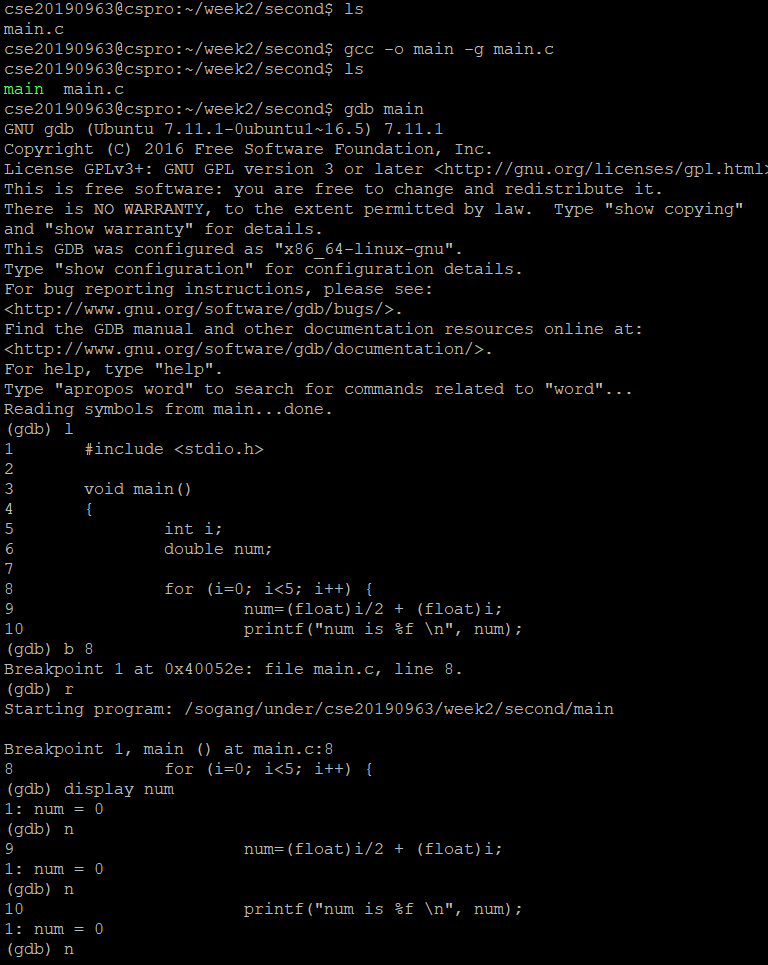
**1.**

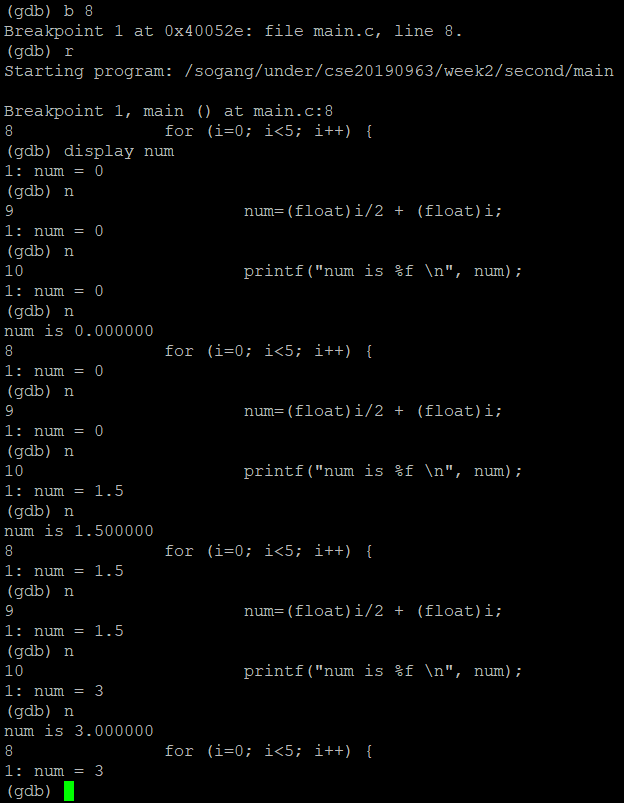
다음은 실습 1의 결과 화면이다. 먼저 make 명령어를 입력하여 animal\_exe와 네 개의 오브젝트 파일을 생성한다. animal\_exe를 실행하면 dog, blackcow, turtle이 출력되고 make clean을 입력하여 animal\_exe 파일과 네 개의 오브젝트 파일을 삭제한다.



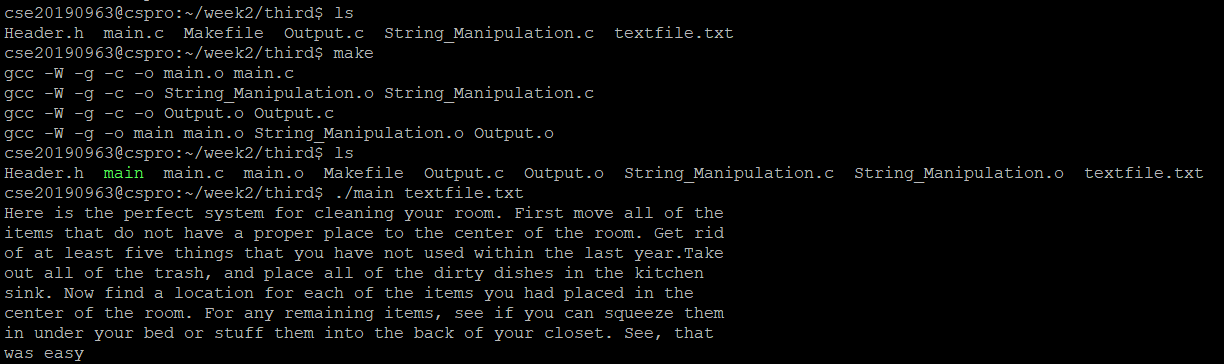


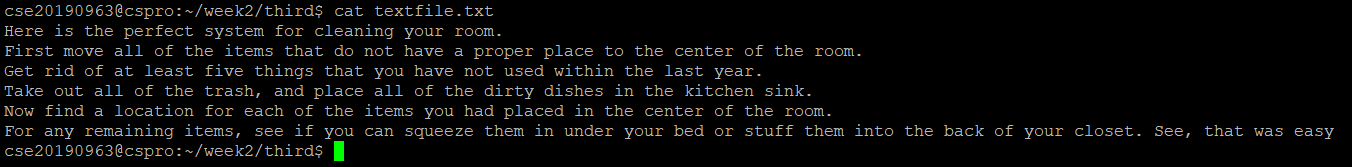
다음은 실습 2의 결과 화면이다. gcc -o main -g main.c로 gdb를 실행할 수 있는 main을 생성한다. Breakpoint를 8번 라인에 추가하고 실행한다. Display num 명령어로 num의 값을 출력하도록 하고, n을 입력하여 한 줄 씩 실행한다.





다음은 실습 3의 결과 화면이다. make로 컴파일을 한 뒤 main 파일을 실행하면 textfile에 있는 문장들이 다음과 같이 출력된다. textfile에 있는 원래 문장들은 두 번째 사진과 같다.





**2.**

fmt를 구현하기 위해 사용한 함수는 다음과 같다.

void Remove\_Blanks\_At\_The\_End( char \*line );

void Print\_Line( char \*line, int \*Count, int \*B\_Flag );

void Get\_Blanks\_Chars( char \*line, int Start, int \*N\_Blanks, int \*N\_Chars );

먼저 Remove\_Blanks\_At\_The\_End 함수는 문장의 끝에 blank(‘ ‘)가 있을 때 이를 삭제하기 위한 함수이다. 또한 문장의 끝에 있는 blank를 모두 제거한 뒤에 문장의 마지막에 \n이나 \0를 추가하기도 한다. Print\_Line 함수는 fmt의 룰에 맞는 형식으로 한 문장을 출력하기 위한 함수이다. 마지막으로 Get\_Blanks\_Chars 함수는 Print\_Line 함수의 구현을 위해 사용되는 함수이다. fmt에서는 단어의 중간에서 줄을 바꿀 수 없기 때문에 한 글자씩 출력하는 것이 아니라 한 단어씩 출력해야 한다. Get\_Blanks\_Chars는 문장 속에서 단어를 구별하는 역할을 하고, Print\_Line 함수는 Get\_Blanks\_Chars 함수를 다음 단어의 글자수가 0이 될 때까지 반복하여 한 문장을 출력한다.

**3.**

cc=gcc

target = animal\_exe

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

$(target): $(objects)

    $(cc) -o $(target) $(objects)

dog.o blackcow.o turtle.o main.o : header.h

.PHONY : clean

clean :

    rm $(target) $(objects)

위 코드는 실습 1의 Makefile이다. 먼저 cc=gcc는 gcc를 이용하여 컴파일 한다는 것을 의미하는데 gcc를 cc라는 변수에 저장하는 형식이다. target=animal\_exe은 animal\_exe가 Makefile의 최종 목표가 되는 파일이라는 의미이다. objects = dog.o blackcow.o turtle.o main.o는 make에서 사용되는 모든 오브젝트 파일들을 말한다. 실습1에서는 dog.o, blackcow.o, turtle.o, main.o 총 4개의 오브젝트 파일이 사용된다.

$(target): $(objects)

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

는 각각의 오브젝트 파일을 target 파일로 통합하는 과정이다. dog.o를 예로 들면,

animal\_exe: dog.o

gcc -o animal\_exe dog.o

이다. blackcow.o, turtle.o, main.o 모두 위와 같은 과정을 거쳐 animal\_exe 파일로 컴파일 된다. dog.o blackcow.o turtle.o main.o : header.h는 header.h 파일이 변경되면 object 파일을 새로 생성한다는 것을 의미한다.

.PHONY : clean에서 clean은 실제 파일의 이름을 의미하는 것이 아니라 다음에 나올 recipe를 대표하는 이름이다.

clean :

rm $(target) $(objects)

.PHONY : clean에서 clean에 대한 recipe이다. make clean이라는 명령어를 입력하면 target(animal\_exe)와 make에 사용된 모든 오브젝트 파일을 삭제한다는 내용이다.

**4.**

규칙 R5는 입력줄의 첫 글자가 blank이면 앞줄과 합쳐지지 않도록 줄을 바꿔 출력하는 것이다. 이를 실행하기 위해서 두 문장의 상태를 알아야 한다. 즉, 현재 출력 중인 줄 line1과 다음 줄 line2의 상태를 모두 알 필요가 있다. 다음 줄이 blank로 시작하는지 아닌지에 따라 line1의 출력이 다르게 처리되어야 하기 때문이다. 이러한 과정을 처리하기 위해서 Count와 B\_Flag라는 변수가 필요하다. Count는 현재 줄에 실제로 출력할 글자의 개수를 저장하는 변수이고, B\_Flag는 줄바꿈을 할 것인지, 현재 줄에 이어서 출력할 것인지 결정하는 변수이다. B\_Flag의 값이 1이면 현재 줄에서 계속 출력하고, B\_Flag의 값이 0이면 새로운 줄에서 출력한다. 먼저, line1의 상태를 파악하고 line1의 끝에 있는 blank를 제거한다. 이 후 line1을 출력하고 line2의 상태를 파악한다. line2의 끝에 있는 blank를 제거한 후, line2의 첫 글자가 blank인지 확인한다. 이 때, line2의 첫 글자가 blank라면 putchar(‘\n’)을 사용해 줄바꿈을 하고, B\_Flag의 값을 0으로, count의 값을 0으로 설정한다. 이 과정이 끝나면 line2를 line1으로 바꾸고 위와 동일한 과정을 입력이 종료될 때까지 반복한다.

**5.**

make의 옵션은 unix에서 make –help 명령어를 사용하여 확인할 수 있다. make -f makefile 옵션을 사용하면 특정 파일명을 지정할 수 있다. 즉, 여러 개의 makefile이 있을 때 특정 파일을 makefile로 읽을 수 있는 옵션이다. -i 옵션은 오류 코드를 무시하는 옵션이고, -s 옵션은 make를 실행할 때 특별한 표시를 하지 않고 실행하는 옵션이다. -r 옵션은 기존 규칙을 사용하지 않는 옵션이고, -n 옵션은 명령어를 실행하지 않고 표시만 해주는 옵션이다. -C DIRECTORY 옵션은 실행하기 전에 DIRECTORY로 이동하는 옵션이고, -d 옵션은 디버깅 정보를 출력하는 옵션이며 -p 옵션은 make의 내부 데이터베이스를 출력하는 옵션이다. 이외에도 make의 버전 정보를 출력하는 -v 옵션과 현재 디렉토리를 출력하는 -w 옵션 등이 있다.