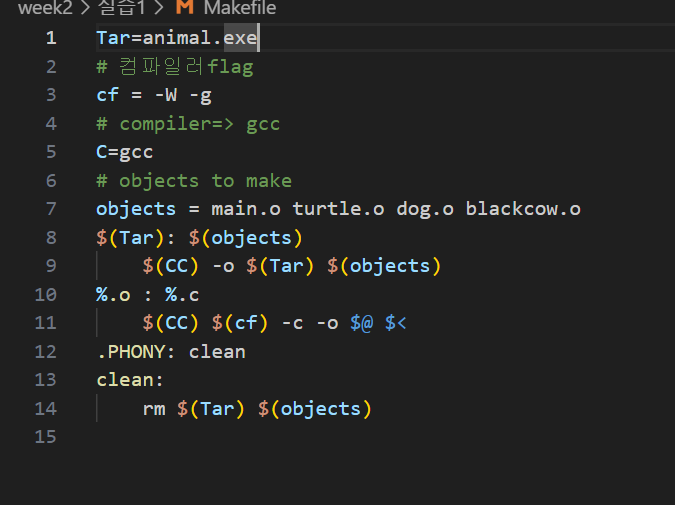
1주차 결과보고서

전공: 컴퓨터공학과 학년: 4학년 학번: 20191612 이름: 윤기웅

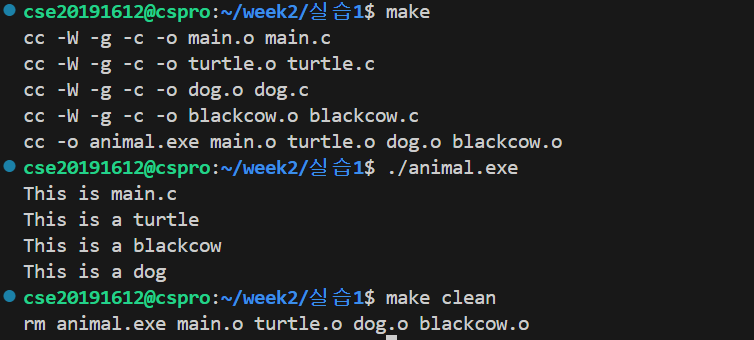
**1. (문제 및 제목은 작성하지 말 것. 답안만 작성하면 됩니다.)**

.................

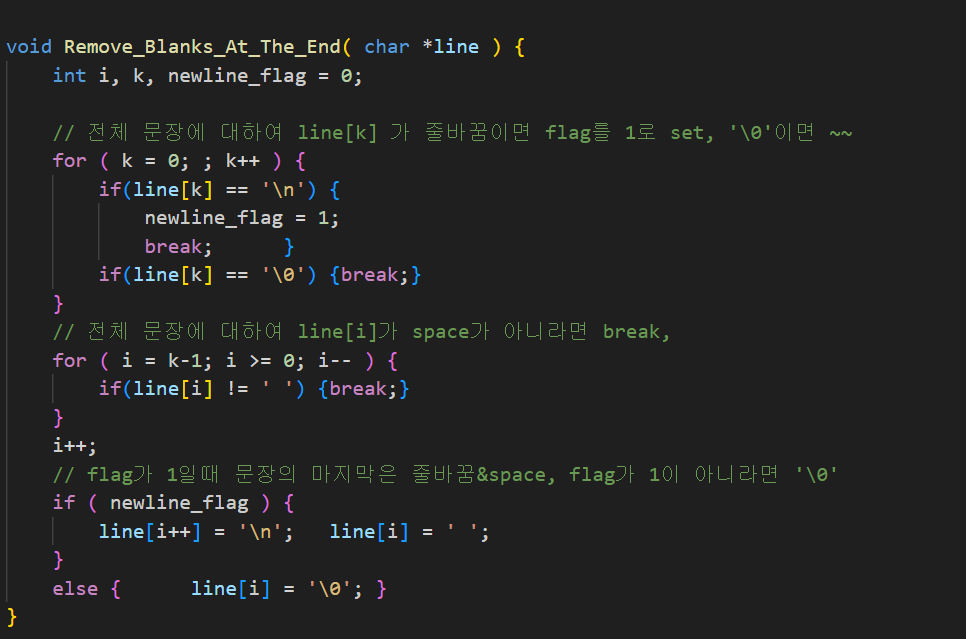
Makefile



Makefile 결과

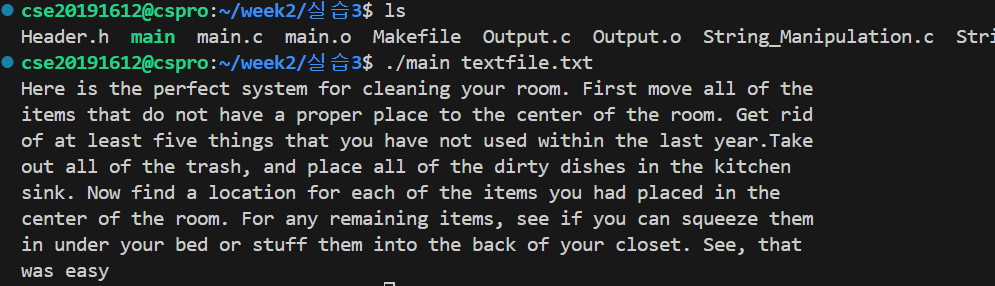


Remove blanks 함수의 코드

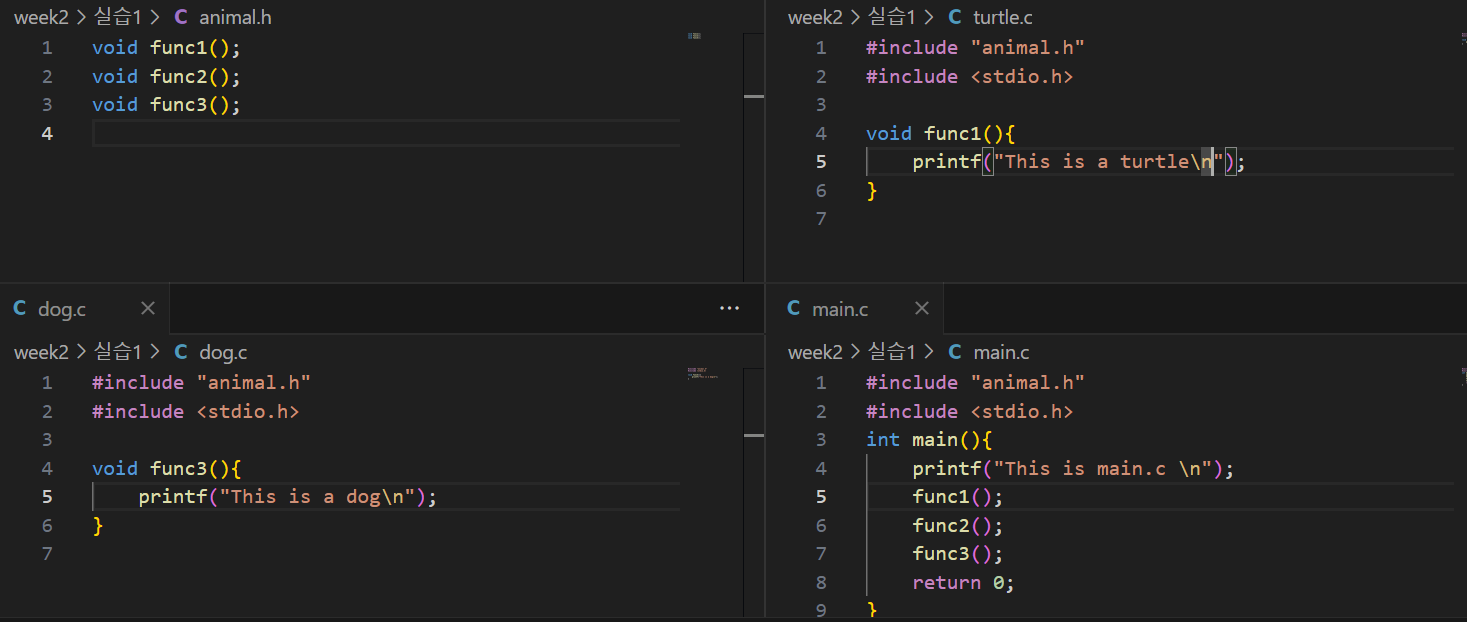


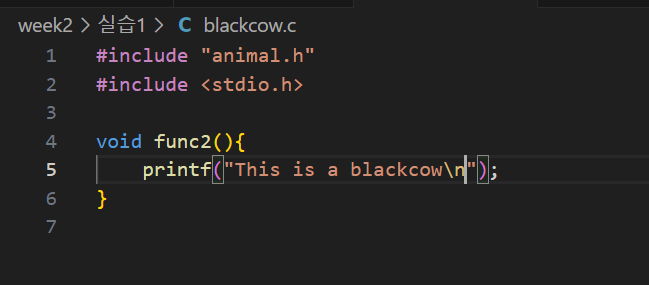
................

Remove blanks at the end 함수를 적용한 text file 출력하기

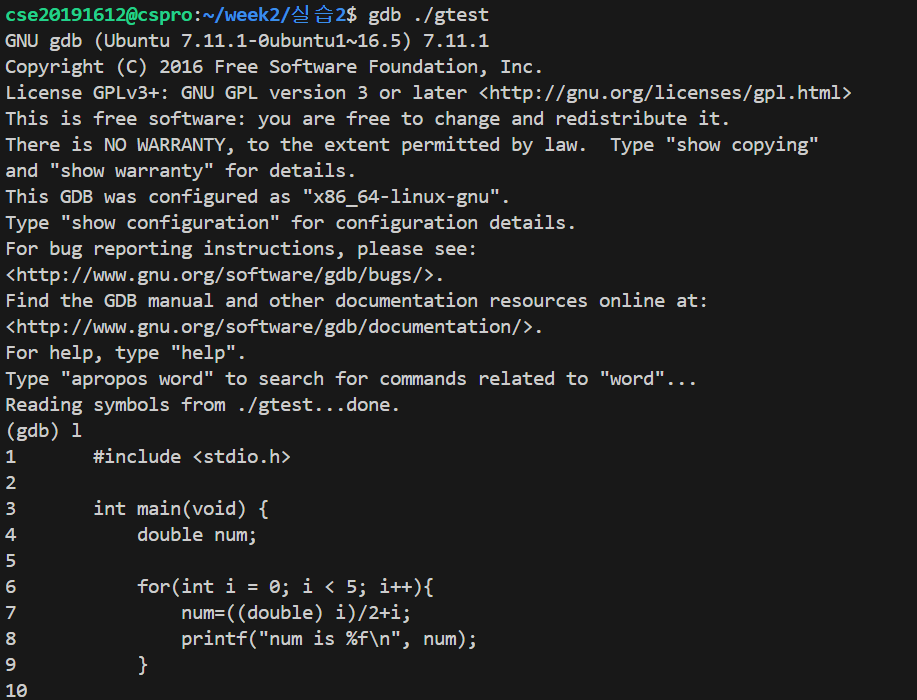


Makefile에 대상이 되는 코드 파일들의 모습이다

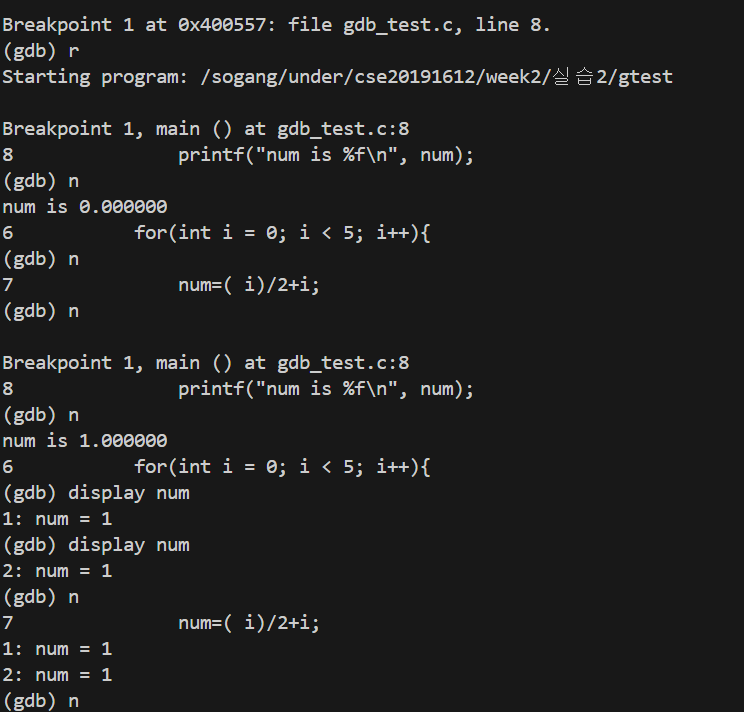




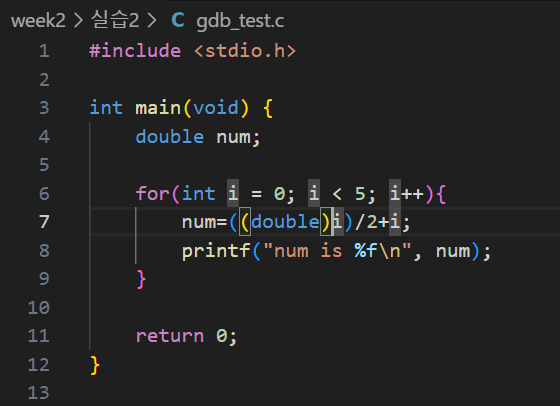
Gdb를 통해서 코드의 에러 찾기 1



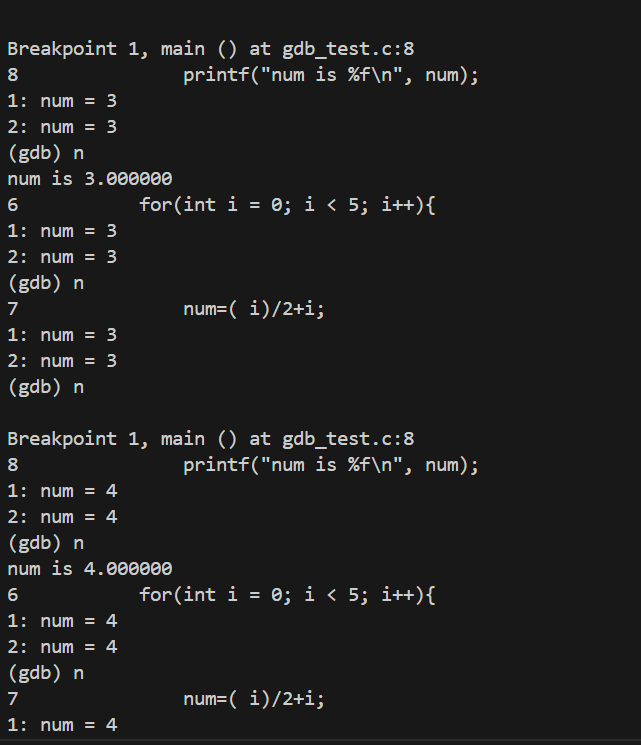
Gdb를 통해서 코드의 에러 찾기 2



Gdb code fix



Gdb를 통해서 코드의 에러 찾기 3



**2.**

.......................

Main 함수: 텍스트 파일을 받아온다. 만약 파일이 없는 경우는 에러 코드를 실행한다.

Print\_Line 함수: 현재 줄에 문자 개수를 확인해서 최대 제한을 넘어서 출력되지 않도록 조정한다. Flag의 값에 따라서 출력을 달리 하는데 만약 플래그가 1인 경우는 같이 이어서 출력하고, 0인 경우는 다시 count에 따라서 출력이 달라진다. 만약, 최대 길이와 비교해서 출력 여유가 있는 경우는 계속 출력하고 아닌 경우는 다음 줄로 출력을 넘긴다.

Get\_Blanks\_Chars 함수: 텍스트의 문자와 빈칸을 카운트하는 함수이다. 텍스트의 시작 지점에서 반복문을 순환하면서 줄 바꿈이나 문장의 끝인 경우는 반복문을 종료하고 문자가 나올 때 까지 공백과 문자의 개수를 각각 구한다.

Remove\_Blanks\_At\_The\_End 함수: 텍스트 파일을 열어서 다른 함수들을 사용해서 limit 글자수 제한에 맞추어서 줄 바꿈을 수행한다. 그리고 줄 바꾸기 전에 공백을 제거해줘야 한다. 텍스트에서 줄 바꿈이나 문장 끝을 찾으면 거기서부터 다시 idx를 줄이면서 공백을 찾는다.

.........................

3.

Tar=animal.exe # 컴파일러로 생성할 최종파일의 이름을 지정한다.

cf = -W -g # flag를 만들어서 파일 내의 생성 규칙 코드에 사용한다.

C=gcc # gcc를 c언어 컴파일러로 지정한다.

objects = main.o turtle.o dog.o blackcow.o# 최종파일 생성에 필요한 오브젝트 파일들을 정의한다.

$(Tar): $(objects) : 목표 파일 만들기 위한 코드로 오브젝트 파일들이 있다면 정상 작동한다.

$(CC) -o $(Tar) $(objects)

%.o : %.c : 각 소스파일에서 오브젝트 파일을 만드는 코드로 이름은 동일하고 파일 형식만 변경된다.

$(CC) $(cf) -c -o $@ $<

.PHONY: clean #phony 사용해서 make clean할 때 제거하고 싶은 파일을 잘 지정하기(제거 명령을 위한 가상 타겟)

clean: # make clean에서 수행할 내용을 다음에 받는다.

rm $(Tar) $(objects) #make clean에서 제거할 파일들을 적는다.

4. 입력 받은 텍스트를 보면서 처음이 빈칸으로 시작하면 합쳐지지 아니하고, 공간이 하나 이상 있는 경우는 다음 번 줄에 출력되도록 넘긴다. Flag를 1혹은 0으로 정해야 하는데 텍스트를 계속 확인하면서 만약에 줄 바꿈이 있는 경우는 flag를 1로 하고 아닌 경우는 0으로 한다. 그리고 다시 break된 곳부터 다시 역으로 ­살펴보면서 공백이 아닌 경우는 다시 break를 한다. 한 단어인 경우는 쪼개서 출력하면 안 되기 때문이다. 그리고 flag가 1인 경우에 대해 확인하면서 마지막 조건문을 수행하는데 플래그가 1인 경우는 이제 그 다음부터 다음 줄로 출력되어야 하니 줄 바꿈 문자를 바로 다음에 넣어준다. 아닌 경우는 다음 줄로 출력하는 것이 아니 라서 그냥 공백을 넣어야 한다.

5.

Make란 파일 관리 유틸리티로 파일에 적힌 코드대로 컴파일러가 명령을 수행하도록 한다. 단순 반복 작업을 하지 않아도 되고 코드 파일들의 관리가 편하다. make에는 다양한 옵션들이 존재한다.

* C dir : makefile을 계속 수행하지 아니하고 디렉토리 경로로 이동하는 것이라서 순환 생성에 사용된다.
* d,debug : makefile 수행하면서 여러 정보들을 출력한다. 그래서 출력되는 양이 이전보다 더 많아진다.
* h,help : make의 옵션과 기능에 대한 도움들을 볼 수 있다.
* File<filename> : filename에 해당하는 파일을 makefile로 지정한다.
* r,no-builtin-rules : 기본으로 적용되는 규칙들을 수행하지 않게 만들고 만약 새로운 규칙을 적용하려면 이용자가 직접 새로운 규칙을 정의해야 한다.
* t,touch : 파일의 생성 날짜를 지금 시간으로 최신화 한다.
* v,version : make의 해당 버전을 출력한다.
* p,print-daata-base : make에서 안에 설정되어 있는 값을 보는 것이다.
* k,-keep-going : 에러가 나더라도 정지하지 말고 그냥 진행하는 명령어이다.

참고자료:

1) 2주차\_unix(linux) 에서의 프로그래밍.pptx

2)부록\_Makefile.pdf

3)UNIX-2.pdf