**System Programming**

#5

**make유틸리티 사용**

**1분반**

**장경식 교수님**

**2012136020 김동우**

**2016.10.21**

**1. 샘플 소스 파일 mtest.c foo.c boo.c bar.c를 만든다.**

- foo.c : InFoo() 함수를 정의한다. 함수의 내용은 자유.

- boo.c : InBoo() 함수를 정의한다. 함수내용은 자유.

- bar.c : InBar() 함수를 정의한다. 함수내용은 자유.

- mtest.c : InFoo(), InBoo(), InBar() 함수를 호출한다.\

**2. mtest.c를 컴파일하여, mtest 라는 실행파일을 생성하는 makefile을 작성한다.**

**3. make 유틸리티를 사용하여 mtest를 생성하고, 쉘에서 실행한다.**

**4. 컴파일시 –g 옵션을 추가하여 디버그 정보를 생성한다.**

**5. gdb를 사용하여 mtest 파일에서 Infoo, InBoo, InBar 함수를 호출하는 과정을 trace 한다.**

==========================================

**foo.c**

#include<stdio.h>

void inFoo(){

printf("This is foo.c ==> function --> infoo()\n");

}

**boo.c**

#include<stdio.h>

void inBoo(){

printf("This is boo.c ==> function --> inboo()\n");

}

**bar.c**

#include<stdio.h>

void inBar(){

printf("This is bar.c ==> function --> inBar()\n");

}

**mtest.c**

#include<stdio.h>

extern int inFoo(), inBoo(), inBar();

void main(){

inFoo();

inBoo();

inBar();

printf("\n This is mtest ==> main\n");

}

**makefile**

mtest : mtest.o foo.o boo.o bar.o

gcc -o mtest mtest.o foo.o boo.o bar.o

mtest.o : mtest.c

gcc -c mtest.c

foo.o : foo.c

gcc -c foo.c

boo.o : boo.c

gcc -c boo.c

bar.o : bar.c

gcc -c bar.c

==========================================

**Makefile**

컴파일이라는 것은 프로그래밍 언어로 작성된 파일을 컴파일러를 통해 컴퓨터가 읽을 수 있는 기계어로 옮기는 과정을 말한다.

make란 리눅스, 유닉스에서 컴파일 및 각종 배치작업을 자동화 할 수 있도록 도와주는 유틸리티이다.

소수의 소스코드에서 수정 혹은 변화가 있을 경우 변화가 있는 파일과 그 파일에 의존적인 파일만을 재컴파일 하는 것을 make utility라고 한다.

* make utility를 활용하면, 컴파일 시간을 절약 할 수 있고, 파일 수가 많을 시에 빌드와 관련된 파일을 한 곳에 모을 수 있다.

구조

:<Target> : <Dependancy>

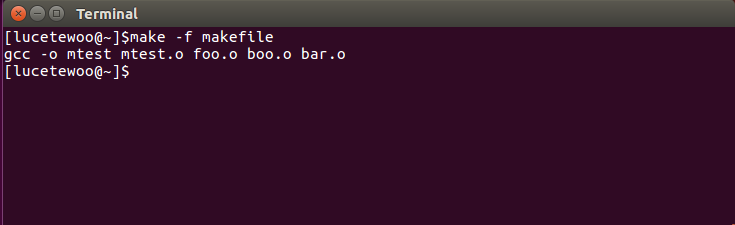
command

Target : 최종 목표

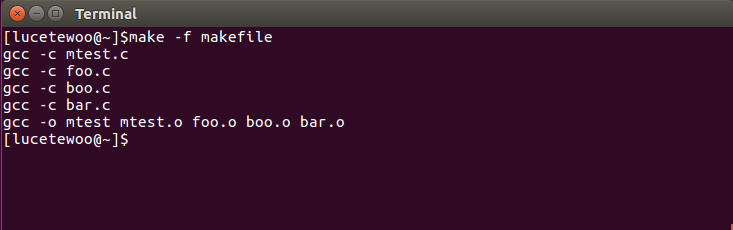
Dependancy : 하위 파일

Command : 명령

실행

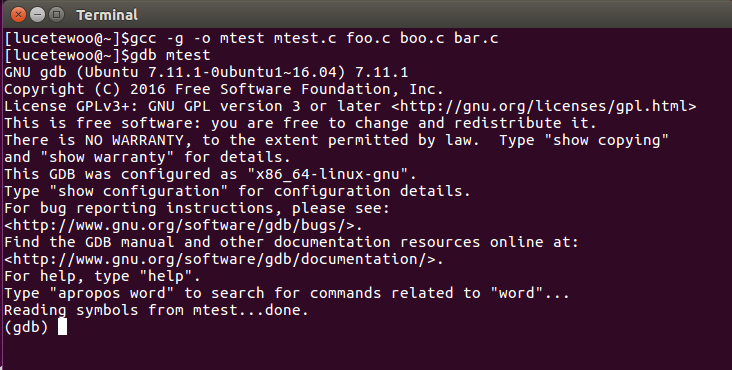


: 이 화면은 그전에 따로 foo.c, boo.c, bar.c에 대한 컴파일을 gcc명령어를 따로 해준 후 makefile유틸리티를 사용하였을 때의 모습이다. 기존의 것들은 이미 존재하고 있고, 수정이 필요 없기에 컴파일이 되어있지 않았던 mtest파일에 대해서만 컴파일 한 것을 볼 수 있다.



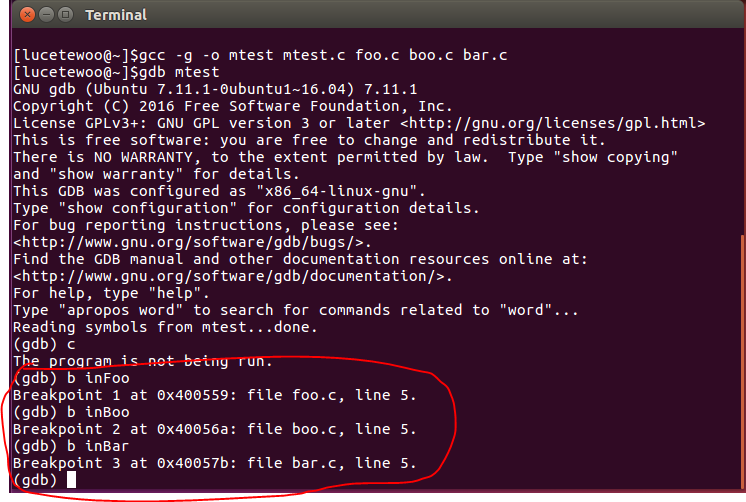
: 전체적인 makefile유틸리티 사용 화면을 보고 싶어, 원래 있던 파일들을 모두 지우고 다시 실행 하였을 때의 화면이다. 모든 부분에 대해 잘 컴파일되고, 출력 된 것을 볼 수 있다.

gdb로 디버깅 하기



: -g 명령어를 추가한 컴파일 후 gdb 명령을 통하여 디버깅을 실행하였다.

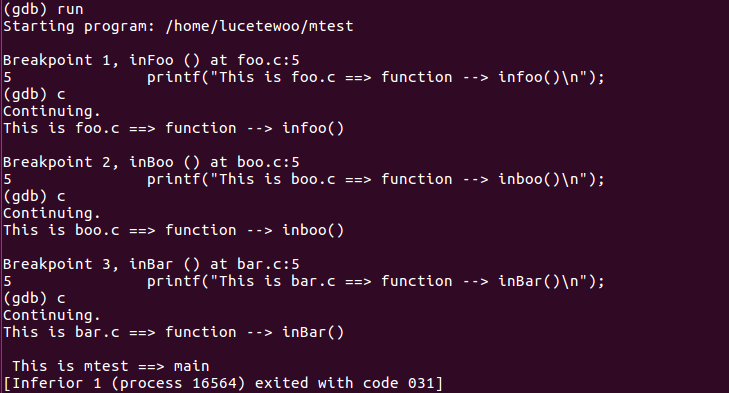
다음과 같은 화면이 뜨게 되며, 명령 또한 다르게 실행된다.



: breakpoint를 설정

각각의 함수에 대해 breakpoint를 설정하였다

.



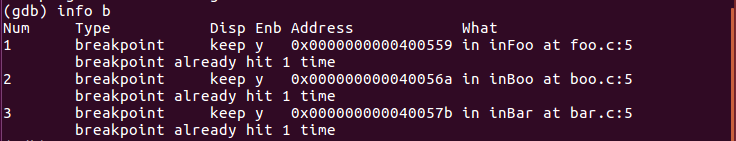
: run명령을 이용하여 mtest.c를 실행시킨다.

처음 breakpoint가 inFoo()함수에서 실행된다.

두번째 breakpoint는 inBoo()함수에서 실행되는 것을 확인 할 수 있다.

마지막으로 inBar()함수가 실행된다.

inBar()함수가 실행되는 위치까지 확인이 되었으니 모든 내용이 출력되고 종료되었다.



:info명령어를 통해 breakpoint에 대한 정보를 확인해 보았다. 전부 한번 씩 실행 되었음을 확인해 볼 수 있었다.

**느낀점**

: 이번 과제는 시험기간에 진행 되었기에, 조금 늦게 과제를 하였다. 지금까지는 gcc명령어를 활용하여 하나하나 컴파일하도록 하여, .o 목적 파일도 만들고 실행파일도 만들었었지만, 이번과제는 makefile이라는 유틸리티를 활용하여 컴파일하는 것이 새로웠다.

makefile유틸리티는 평소에 일일이 목적 파일을 만들어 주어야 했던 것 과는 달리 visualstudio에서 컴파일하는 것같이 많은 편리함을 안겨 줄 수 있는 도구라는 생각이 든다. 이 makefile유틸리티에 적응이 되면, 명령어를 하나하나치는 불편함을 줄이고, 더 편하게 컴파일 할 수 있지 않을까 하는 생각이든다. 하지만, gdb디버깅은 기존의 명령어와는 조금 다른 부분이 많았기에, 새로운 명령어들을 공부할 필요가 있다는 생각이 들었다.