Lab-5 : make 유틸리티 사용

과목명	시스템 프로그래밍	담당 교수님	장경식
분반	01	제출일	2016. 10. 21
학번	2013136100	이름	이진솔

Ⅰ 문제

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.

Ⅱ 코드

foo.c

```c
#include <stdio.h>

void InFoo()
{
        printf("~InFoo Function~");
}
```

boo.c

```c
#include <stdio.h>

void InBoo()
{
        printf("~InBoo Function~");
}
```

bar.c

```c
#include <stdio.h>

void InBar()
{
        printf("~InBar Function~");
}
```

mtest.c

```c
#include <stdio.h>
#include <stdlib.h>

extern int InFoo(), InBoo(), InBar();

int main()
{
        InFoo();
        InBoo();
        InBar();
        printf("hello world!!!");
        exit(0);
}
```

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
```

mtest.c를 컴파일하여, mtest 라는 실행파일을 생성하는 makefile

■구조

Target : Dependency List

(Tab)Command List

Target : 생성할 타겟파일이다.

Dependency List : 타겟파일을 만드는데 필요한 파일들의 목록이다. 하나라도 수정되면 다시 타겟 파일을 만든다.

Command List : 타겟파일을 만들기 위해 사용할 명령어들이다.

Ⅲ 실행

1. make 유틸리티를 사용하여 mtest를 생성하고, 쉘에서 실행

```
jinsol@jinsol-VirtualBox:~$ make -f Makefile
gcc -c mtest.c
gcc -c foo.c
gcc -c boo.c
gcc -c bar.c
gcc -o mtest mtest.o foo.o boo.o bar.o
jinsol@jinsol-VirtualBox:~$ touch foo.c
jinsol@jinsol-VirtualBox:~$ make -f Makefile
gcc -c foo.c
gcc -o mtest mtest.o foo.o boo.o bar.o
```

$ make –f file_name
- 파일명이 Makefile이면 make 명령으로 실행이 가능하지만, 다른 이름으로 되어 있을 때는 –f 과 파일명을 주어야 한다.

■ makefile을 만들고 make 명령어를 실행하는 이유
- 각 파일에 대한 반복적 명령의 자동화로 인한 시간 절약
- 프로그램의 wdthr 구조를 빠르게 파악 할 수 있으며 관리가 용이
- 단순 박복 작업 및 재작성을 최소화

2. 컴파일시 –g 옵션을 추가하여 디버그 정보를 생성

```
jinsol@jinsol-VirtualBox:~$ gcc -g -o mtest mtest.c foo.c boo.c bar.
c
jinsol@jinsol-VirtualBox:~$ gdb mtest
GNU gdb (Ubuntu 7.7.1-0ubuntu5~14.04.2) 7.7.1
Copyright (C) 2014 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 co
pying"
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 mtest...done.
(gdb)
```

$ gdb[프로그램명]
- 실행파일을 gdb로 실행하면서 디버깅

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

```
jinsol@jinsol-VirtualBox:~$ gdb mtest
GNU gdb (Ubuntu 7.7.1-0ubuntu5~14.04.2) 7.7.1
Copyright (C) 2014 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 co
pying"
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 mtest...done.
```

■ gdb

- 컴퓨터 프로그램을 실행하여, 사용자에게 실행과정을 추적하고, 중간에 임의로 수정, 함수, 변수들을 모니터링 할 수 있도록 한다.

```
(gdb) b InFoo
Breakpoint 1 at 0x4005bc: file foo.c, line 5.
(gdb) b InBoo
Breakpoint 2 at 0x4005d1: file boo.c, line 5.
(gdb) b InBar
Breakpoint 3 at 0x4005e6: file bar.c, line 5.
(gdb) run
Starting program: /home/jinsol/mtest
```

■ run 명령어를 실행시키면 프로그램이 처음부터 끝 날 때까지 실행되기만 한다. 때문에 중간의 심볼 현황을 살펴볼 수 있도록 원하는 순간에 멈추기 위해 break (b) 명령어를 사용한다.

break (b) : run 명령어가 실행된 후 멈출 메모리 위치
b [function] // 함수 시작부분에 브레이크 포인트 설정
b [number] // 어셈블리 코드에서 10행에 브레이크 포인트 설정

//프로그램의 소스가 여러 파일로 이루어져 있을 경우
b [file.c]:[function]
b [file.c]:[number]

cl : 명령어로 브레이크 포인트를 지울 수 있다.
d : 모든 브레이크를 지운다.

■ 프로그램 진행 루틴

ni : 어셈블리 소스의 한 줄을 실행. 함수가 호출되는 부분도 한 줄로 인식
si : 어셈블리 소스의 한 줄을 실행하는데, 호출된다면 함수루틴으로 들어감

// 디버깅 정보가 없다면 run
next (n) : c언어 소소의 한 줄을 실행. 함수 호출시, 한 줄로 처리하여 함수 수행
step (s) : c언어 소스의 한 줄을 실행. 호출시 루틴 안으로 들어감.

c : 브레이크 포인트를 만날 때 까지 계속 진행

finish (f) : 함수가 끝난 지점으로 이동(함수가 끝날 때 까지 실행)
return (r) : 함수가 끝난 지점으로 이동(현재 함수 실행 안함)
until (u) : 현재 루프를 빠져나감.

```
Breakpoint 1, InFoo () at foo.c:5
5               printf("~InFoo Function~");
(gdb) c
Continuing.

Breakpoint 2, InBoo () at boo.c:5
5               printf("~InBoo Function~");
(gdb) c
Continuing.

Breakpoint 3, InBar () at bar.c:5
5               printf("~InBar Function~");
(gdb) info b
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x00000000004005bc in InFoo
                                                   at foo.c:5
        breakpoint already hit 1 time
2       breakpoint     keep y   0x00000000004005d1 in InBoo
                                                   at boo.c:5
        breakpoint already hit 1 time
3       breakpoint     keep y   0x00000000004005e6 in InBar
                                                   at bar.c:5
        breakpoint already hit 1 time
```

break에 대한 정보 출력