# 리시프 특강 (gcc, gdb)

## 1. GCC(5.4.0)

### 1.1. gcc warning messages

```
:~/source/gcc/warning$ vim w_ex1.c
```

filepath : warning/w_ex1.c

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

int main(int argc, char *argv[])
{
    if (argc != 2){
        printf("usage : a.out pathname\n");
        exit(1);
    }
    if (access(argv[1], R_OK) < 0)
        printf("access error for %s\n", argv[1]);
    else
        printf("read access OK\n");
    if (open(argv[1], O_RDONLY) < 0)
        printf("open error for %s\n", argv[1]);
    else
        printf("open for reading OK\n");
    exit(0);
}
```

```
:~/source/gcc/warning$ gcc w_ex1.c
```

```
:~/source/gcc/warning$ gcc w_ex1.c
w_ex1.c: In function ‘main’:
w_ex1.c:11:9: warning: implicit declaration of function ‘access’ [-Wimplicit-function-declaration]
     if (access(argv[1], R_OK) < 0)
         ^
:~/source/gcc/warning$
```

○ Message Format
- file name : In function ‘function name’: (함수안에서 발생할 경우)
- file name : line number : column number : warning: message

○ Warning Messages
- [-Wformat=]

```
:~/source/gcc/warning$ vim w_ex2.c
```

filepath : warning/w_ex2.c

```c
#include <stdio.h>

int main(void)
{
        int x = 10;
        float y = 20.0;
        printf("value x:%f\n", x, y);
        return 0;
}
```

```
:~/source/gcc/warning$ gcc w_ex2.c
```

```
:~/source/gcc/warning$ gcc w_ex2.c
w_ex2.c: In function ‘main’:
w_ex2.c:7:9: warning: format ‘%f’ expects argument of type ‘double’, but argument 2 has type ‘int’
[-Wformat=]
  printf("value x:%f\n", x, y);
         ^
w_ex2.c:7:9: warning: too many arguments for format [-Wformat-extra-args]
:~/source/gcc/warning$
```

- [-Wreturn-type]

```
:~/source/gcc/warning$ vim w_ex3.c
```

```
filepath : warning/w_ex3.c
#include <stdio.h>
int foo();

int main(void)
{
        foo();
        return 0;
}

int foo()
{
        printf("foo\n");
}
```

```
:~/source/gcc/warning$ gcc w_ex3.c
:~/source/gcc/warning$ gcc -Wreturn-type w_ex3.c
```

```
:~/source/gcc/warning$ gcc w_ex3.c
:~/source/gcc/warning$ gcc -Wreturn-type w_ex3.c
w_ex3.c: In function 'foo':
w_ex3.c:13:1: warning: control reaches end of non-void function [-Wreturn-type]
 }
 ^
:~/source/gcc/warning$
```

- [-Wunused-variable]

```
:~/source/gcc/warning$ vim w_ex4.c
```

```
filepath : warning/w_ex4.c
#include <stdio.h>

int main(void)
{
        int x = 10;
        printf("Hello World\n");
        return 0;
}
```

```
:~/source/gcc/warning$ gcc -Wunused-variable w_ex4.c
```

```
:~/source/gcc/warning$ gcc -Wunused-variable w_ex4.c
w_ex4.c: In function 'main':
w_ex4.c:5:6: warning: unused variable 'x' [-Wunused-variable]
  int x = 10;
      ^
:~/source/gcc/warning$
```

- [-Wmissing-braces]

```
:~/source/gcc/warning$ vim w_ex5.c
```

```
filepath : warning/w_ex5.c
#include <stdio.h>

int a[2][2] = { 0, 1, 2, 3 };
int b[2][2] = { { 0, 1 }, { 2, 3 } };

int main(void)
{
        printf("Hello Wolrd\n");
        return 0;
}
```

```
:~/source/gcc/warning$ gcc w_ex5.c
:~/source/gcc/warning$ gcc −Wmissing−braces w_ex5.c
```

```
:~/source/gcc/warning$ gcc w_ex5.c
:~/source/gcc/warning$ gcc −Wmissing−braces w_ex5.c
w_ex5.c:3:15: warning: missing braces around initializer [−Wmissing−braces]
 int a[2][2] = { 0, 1, 2, 3 };
               ^
w_ex5.c:3:15: note: (near initialization for ‘a’)
:~/source/gcc/warning$
```

- [-Wuninitialized]

```
:~/source/gcc/warning$ vim w_ex6.c
```

filepath : warning/w_ex6.c

```
#include <stdio.h>

int main(void)
{
        int x;
        printf("value : %d\n", x);
        return 0;
}
```

```
:~/source/gcc/warning$ gcc w_ex6.c
:~/source/gcc/warning$ gcc −Wuninitialized w_ex6.c
```

```
:~/source/gcc/warning$ gcc w_ex6.c
:~/source/gcc/warning$ gcc −Wuninitialized w_ex6.c
w_ex6.c: In function ‘main’:
w_ex6.c:6:2: warning: ‘x’ is used uninitialized in this function [−Wuninitialized]
  printf("value : %d\n", x);
  ^
:~/source/gcc/warning$
```

- [-Wsign-compare]

```
:~/source/gcc/warning$ vim w_ex7.c
```

filepath : warning/w_ex7.c

```
#include <stdio.h>

int main(void)
{
        char arr[] = "Hello World\n";
        for(int i = 0 ; i < sizeof(arr) ; i++)
                printf("%c", arr[i]);
        return 0;
}
```

```
:~/source/gcc/warning$ gcc w_ex7.c
:~/source/gcc/warning$ gcc −Wsign−compare w_ex7.c
```

```
:~/source/gcc/warning$ gcc w_ex7.c
:~/source/gcc/warning$ gcc −Wsign−compare w_ex7.c
w_ex7.c: In function ‘main’:
w_ex7.c:6:20: warning: comparison between signed and unsigned integer expressions [−Wsign−compare]
  for(int i = 0 ; i < sizeof(arr) ; i++)
                    ^
:~/source/gcc/warning$
```

- [-Wunused-parameter]

```
    :~/source/gcc/warning$ vim w_ex8.c
```

```
filepath : warning/w_ex8.c
#include <stdio.h>

int main(int argc, char *argv[])
{
        printf("Hello World\n");
        return 0;
}
```

```
    :~/source/gcc/warning$ gcc w_ex8.c
    :~/source/gcc/warning$ gcc -Wunused-parameter w_ex8.c
```

```
:~/source/gcc/warning$ gcc w_ex8.c
:~/source/gcc/warning$ gcc -Wunused-parameter w_ex8.c
w_ex8.c: In function ‘main’:
w_ex8.c:3:14: warning: unused parameter ‘argc’ [-Wunused-parameter]
 int main(int argc, char *argv[])
              ^
w_ex8.c:3:26: warning: unused parameter ‘argv’ [-Wunused-parameter]
 int main(int argc, char *argv[])
                          ^
:~/source/gcc/warning$
```

- [-Wall], [-W]

| [-Wall] | [-W] or [-Wextra] | both [-Wall] and [-W] | etc |
|---|---|---|---|
| [-Wformat=] | [-Wuninitialized] | [-Wunused-parameter] | [-Wconversion] |
| [-Wimplicit-function-declaration] | [-Wsign-compare] | [-Wunused-but-set-parameter] | [-Wcast-qual] |
| [-Wreturn-type] | ... | ... | ... |
| [-Wunused-variable] | | | |
| [-Wmissing-braces] | | | |
| [-Wuninitialized] | | | |
| ... | | | |

※ https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#Warning-Options

- [-Wconversion]

```
:~/source/gcc/warning$ vim w_ex9.c
```

```
filepath : warning/w_ex9.c
#include <stdio.h>
#include <stdlib.h>

int main (void)
{
        double x = -3.14;
        double y = abs(x);  /* fabs(x)*/
        printf ("x = %g |x| = %g\n", x, y);
        return 0;
}
```

```
:~/source/gcc/warning$ gcc -W w_ex9.c
:~/source/gcc/warning$ gcc -Wall w_ex9.c
:~/source/gcc/warning$ gcc -Wconversion w_ex9.c
```

```
:~/source/gcc/warning$ gcc -W w_ex9.c
:~/source/gcc/warning$ gcc -Wall w_ex9.c
:~/source/gcc/warning$ gcc -Wconversion w_ex9.c
w_ex9.c: In function ‘main’:
w_ex9.c:7:17: warning: conversion to ‘int’ from ‘double’ may alter its value [-Wfloat-conversion]
  double y = abs(x);  /* fabs(x)*/
                 ^

:~/source/gcc/warning$
```

- [-Wcast-qual]

```
:~/source/gcc/warning$ vim w_ex10.c
```

```
filepath : warning/w_ex10.c
#include <stdio.h>

void f(const char *str)
{
        char *s = (char *)str;
        s[0] = '\0';
}

int main(void)
{
        char *a = "hello World";
        f(a);
        return 0;
}
```

```
:~/source/gcc/warning$ gcc -W -Wall w_ex10.c
:~/source/gcc/warning$ ./a.out
```

```
:~/source/gcc/warning$ gcc -W -Wall w_ex10.c
:~/source/gcc/warning$ ./a.out
세그멘테이션 오류 (core dumped)
:~/source/gcc/warning$
```

```
:~/source/gcc/warning$ gcc -Wcast-qual w_ex10.c
```

```
:~/source/gcc/warning$ gcc -Wcast-qual w_ex10.c
w_ex10.c: In function ‘f’:
w_ex10.c:5:12: warning: cast discards ‘const’ qualifier from pointer target type [-Wcast-qual]
  char *s = (char *)str;
            ^

:~/source/gcc/warning$
```

## 1.2. gcc -D option flag (defines a macro to be used by the preprocessor)

```
:~/source/gcc/define$ vim d_ex1.c
```

```
filepath : deifne/d_ex1.c
#include <stdio.h>

int main(void)
{
        int radius = 3;
        printf("radius:");
        printf("Circumference:%f\n", radius * PI);
        return 0;
}
```

```
:~/source/gcc/define$ gcc -Wall -W d_ex1.c
```

```
:~/source/gcc/define$ gcc -Wall -W d_ex1.c
d_ex1.c: In function ‘main’:
d_ex1.c:7:40: error: ‘PI’ undeclared (first use in this function)
  printf("Circumference:%f\n", radius * PI);
                                        ^
d_ex1.c:7:40: note: each undeclared identifier is reported only once for each function it appears in
:~/source/gcc/define$
```

```
:~/source/gcc/define$ gcc -Wall -W -DPI=3.14 d_ex1.c
:~/source/gcc/define$ ./a.out
:~/source/gcc/define$ gcc -Wall -W -DPI=3.14159 d_ex1.c
:~/source/gcc/define$ ./a.out
```

```
:~/source/gcc/define$ gcc -Wall -W -DPI=3.14 d_ex1.c
:~/source/gcc/define$ ./a.out
radius:Circumference:9.420000
:~/source/gcc/define$ gcc -Wall -W -DPI=3.14159 d_ex1.c
:~/source/gcc/define$ ./a.out
radius:Circumference:9.424770
:~/source/gcc/define$
```

#define PI 3.14 와 동일

○ conditionals

```
:~/source/gcc/define$ vim d_ex2.c
```

```
filepath : define/d_ex2.c
#include <stdio.h>

#ifndef PI
        #define PI 3.14
#endif

int main(void)
{
        int radius;
        printf("radius:");
        scanf("%d", &radius);
#ifdef DEBUG
        printf("address of radius:%p\n", &radius);
#endif
        printf("Circumference:%f\n", radius * PI);
        return 0;
}
```

conditionals 매크로를 이용해 컴파일할때 포함될 소스를 선택할 수 있다.

```
:~/source/gcc/define$ gcc -Wall -W d_ex2.c
:~/source/gcc/define$ ./a.out
```

```
:~/source/gcc/define$ gcc -Wall -W d_ex2.c
:~/source/gcc/define$ ./a.out
radius:3
Circumference:9.420000
:~/source/gcc/define$
```

```
:~/source/gcc/define$ gcc -Wall -W -DPI=3.14159 -DDEBUG d_ex2.c
:~/source/gcc/define$ ./a.out
```

```
:~/source/gcc/define$ gcc -Wall -W -DPI=3.14159 -DDEBUG d_ex2.c
:~/source/gcc/define$ ./a.out
radius:3
address of radius:0xbfa08e78
Circumference:9.424770
:~/source/gcc/define$
```

-Dmacro를 여러개 나열해서 사용 할수 있고, value 부분을 생략할 수 있다.

○ 디버깅 메시지 출력

```
:~/source/gcc/define$ vim d_ex3.h
```

filepath : define/d_ex3.h

```
#ifndef __DEBUG__
#define __DEBUG__

#ifdef DEBUG
#define REDS     "\x1b[31m"
#define REDE     "\x1b[0m"
#define DBGMSG_PREFX     REDS "<< DBGMSG >> "

#define DBGMSG(msg,...) fprintf(stderr, \
                        DBGMSG_PREFX"[%s %s %d] : " msg "\n" REDE, __FILE__, __func__, \
                                                                  __LINE__, ##__VA_ARGS__)

#else
#define DBGMSG(...)
#endif
#endif
```

printf처럼 사용

```
:~/source/gcc/define$ vim d_ex3.c
```

filepath : define/d_ex3.c

```
#include <stdio.h>
#include <unistd.h>
#include "d_ex3.h"

void f(void);

int main(void)
{
        int i;
        for(i = 1 ; i <= 20 ; i++) {
                DBGMSG("%d job processing", i);
                f();
        }
        printf("completed\n");
        return 0;
}

void f(void)
{
        sleep(1);
}
```

- 컴파일 및 실행

```
:~/source/gcc/define$ gcc -Wall -W d_ex3.c
:~/source/gcc/define$ ./a.out
```

```
:~/source/gcc/define$ gcc -Wall -W d_ex3.c
:~/source/gcc/define$ ./a.out
completed
:~/source/gcc/define$
```

※ ANSI_COLOR_CODE

```
#define ANSI_COLOR_GREEN        "\x1b[32m"
#define ANSI_COLOR_YELLOW       "\x1b[33m"
#define ANSI_COLOR_BLUE         "\x1b[34m"
#define ANSI_COLOR_MAGENTA      "\x1b[35m"
#define ANSI_COLOR_CYAN         "\x1b[36m"
```

```
:~/source/gcc/define$ gcc -Wall -W -DDEBUG d_ex3.c
:~/source/gcc/define$ ./a.out
```

```
:~/source/gcc/define$ gcc -Wall -W -DDEBUG d_ex3.c
:~/source/gcc/define$ ./a.out
<< DBGMSG >> [d_ex3.c main 11] : 1 job processing
<< DBGMSG >> [d_ex3.c main 11] : 2 job processing
<< DBGMSG >> [d_ex3.c main 11] : 3 job processing
<< DBGMSG >> [d_ex3.c main 11] : 4 job processing
<< DBGMSG >> [d_ex3.c main 11] : 5 job processing
<< DBGMSG >> [d_ex3.c main 11] : 6 job processing
<< DBGMSG >> [d_ex3.c main 11] : 7 job processing
<< DBGMSG >> [d_ex3.c main 11] : 8 job processing
<< DBGMSG >> [d_ex3.c main 11] : 9 job processing
<< DBGMSG >> [d_ex3.c main 11] : 10 job processing
<< DBGMSG >> [d_ex3.c main 11] : 11 job processing
<< DBGMSG >> [d_ex3.c main 11] : 12 job processing
<< DBGMSG >> [d_ex3.c main 11] : 13 job processing
<< DBGMSG >> [d_ex3.c main 11] : 14 job processing
<< DBGMSG >> [d_ex3.c main 11] : 15 job processing
<< DBGMSG >> [d_ex3.c main 11] : 16 job processing
<< DBGMSG >> [d_ex3.c main 11] : 17 job processing
<< DBGMSG >> [d_ex3.c main 11] : 18 job processing
<< DBGMSG >> [d_ex3.c main 11] : 19 job processing
<< DBGMSG >> [d_ex3.c main 11] : 20 job processing
completed
:~/source/gcc/define$
```

○ Standard File Descriptor I/O Redirection

```
:~/source/gcc/define$ ./a.out > stdout.log
```

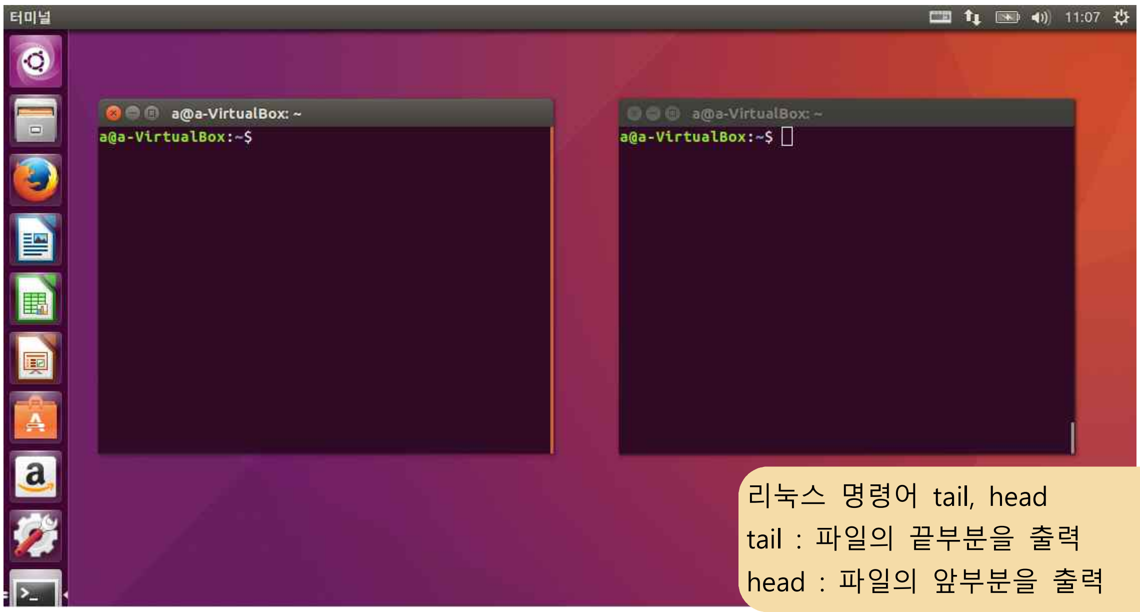
```
:~/source/gcc/define$ ./a.out > stdout.log
<< DBGMSG >> [d_ex3.c main 11] : 1 job processing
<< DBGMSG >> [d_ex3.c main 11] : 2 job processing
<< DBGMSG >> [d_ex3.c main 11] : 3 job processing
<< DBGMSG >> [d_ex3.c main 11] : 4 job processing
<< DBGMSG >> [d_ex3.c main 11] : 5 job processing
<< DBGMSG >> [d_ex3.c main 11] : 6 job processing
<< DBGMSG >> [d_ex3.c main 11] : 7 job processing
<< DBGMSG >> [d_ex3.c main 11] : 8 job processing
<< DBGMSG >> [d_ex3.c main 11] : 9 job processing
<< DBGMSG >> [d_ex3.c main 11] : 10 job processing
<< DBGMSG >> [d_ex3.c main 11] : 11 job processing
<< DBGMSG >> [d_ex3.c main 11] : 12 job processing
<< DBGMSG >> [d_ex3.c main 11] : 13 job processing
<< DBGMSG >> [d_ex3.c main 11] : 14 job processing
<< DBGMSG >> [d_ex3.c main 11] : 15 job processing
<< DBGMSG >> [d_ex3.c main 11] : 16 job processing
<< DBGMSG >> [d_ex3.c main 11] : 17 job processing
<< DBGMSG >> [d_ex3.c main 11] : 18 job processing
<< DBGMSG >> [d_ex3.c main 11] : 19 job processing
<< DBGMSG >> [d_ex3.c main 11] : 20 job processing
:~/source/gcc/define$
```

```
:~/source/gcc/define$ ./a.out 2> stderr.log
```

```
:~/source/gcc/define$ ./a.out 2> stderr.log
completed
:~/source/gcc/define$
```

○ tail –f [파일명]

- 파일의 내용을 추적
- 두 개의 터미널을 A, B 준비



리눅스 명령어 tail, head
tail : 파일의 끝부분을 출력
head : 파일의 앞부분을 출력

- 터미널 A에서 실행

```
:~/source/gcc/define$ tail –f stderr.log
```

```
:~/source/gcc/define$ tail –f stderr.log
<< DBGMSG >> [d_ex3.c main 11] : 12 job processing
<< DBGMSG >> [d_ex3.c main 11] : 13 job processing
<< DBGMSG >> [d_ex3.c main 11] : 14 job processing
<< DBGMSG >> [d_ex3.c main 11] : 15 job processing
<< DBGMSG >> [d_ex3.c main 11] : 16 job processing
<< DBGMSG >> [d_ex3.c main 11] : 17 job processing
<< DBGMSG >> [d_ex3.c main 11] : 18 job processing
<< DBGMSG >> [d_ex3.c main 11] : 19 job processing
<< DBGMSG >> [d_ex3.c main 11] : 20 job processing
```

기존의 끝부분이 출력됨

- 터미널 B에서 실행

```
:~/source/gcc/define$ ./a.out 2> stderr.log
```

```
:~/source/gcc/define$ ./a.out 2> stderr.log
```

## 1.3. gcc 최적화 옵션

```
:~/source/gcc/optimum$ vim o_ex.c
```

```
filepath : optimum/o_ex.c
#include <stdio.h>
double powern (double d, unsigned n)
{
        double x = 1.0;
        unsigned j;
        for (j = 1; j <= n; j++)
                x *= d;
        return x;
}

int main (void)
{
        double sum = 0.0;
        unsigned i;
        for (i = 1; i <= 1000000000; i++)
        {
                sum += powern (i, i % 5);
        }
        printf ("sum = %g\n", sum);
        return 0;
}
```

```
:~/source/gcc/optimum$ gcc -Wall -W -O0 o_ex.c
:~/source/gcc/optimum$ time ./a.out
```

```
:~/source/gcc/optimum$ gcc -Wall -W -O0 o_ex.c
:~/source/gcc/optimum$ time ./a.out
sum = 4e+43

real    0m11.371s
user    0m11.353s
sys     0m0.000s
:~/source/gcc/optimum$
```

```
:~/source/gcc/optimum$ gcc -Wall -W -O1 o_ex.c
:~/source/gcc/optimum$ time ./a.out
```

```
:~/source/gcc/optimum$ gcc -Wall -W -O1 o_ex.c
:~/source/gcc/optimum$ time ./a.out
sum = 4e+43

real    0m2.741s
user    0m2.733s
sys     0m0.004s
:~/source/gcc/optimum$
```

```
:~/source/gcc/optimum$ gcc -Wall -W -O2 o_ex.c
:~/source/gcc/optimum$ time ./a.out
```

```
:~/source/gcc/optimum$ gcc -Wall -W -O2 o_ex.c
:~/source/gcc/optimum$ time ./a.out
sum = 4e+43

real    0m2.145s
user    0m2.142s
sys     0m0.000s
:~/source/gcc/optimum$
```

```
:~/source/gcc/optimum$ gcc -Wall -W -O3 o_ex.c
:~/source/gcc/optimum$ time ./a.out
```

```
:~/source/gcc/optimum$ gcc -Wall -W -O3 o_ex.c
:~/source/gcc/optimum$ time ./a.out
sum = 4e+43

real     0m2.137s
user     0m2.134s
sys      0m0.000s
:~/source/gcc/optimum$
```

```
:~/source/gcc/optimum$ gcc -Wall -W -O3 -funroll-loops o_ex.c
:~/source/gcc/optimum$ time ./a.out
```

```
:~/source/gcc/optimum$ gcc -Wall -W -O3 -funroll-loops o_ex.c
:~/source/gcc/optimum$ time ./a.out
sum = 4e+43

real     0m2.628s
user     0m2.625s
sys      0m0.000s
:~/source/gcc/optimum$
```

일반적으로
디버깅시 : -O0 /  배포시 : -O2

※ https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html

## 2. gdb

○ gdb 시작과 종료

- $ gdb [ prog [ core | procID ] ] [option]
  - ✓ prog
    - 디버깅 할 프로그램 이름
  - ✓ core
    - 프로그램 실행 중에 "segmentation fault" 등의 오류로 비정상 종료할 때 생성되는 파일
    - 비정상 종료 시 시스템 내부 상태 저장
    - 비정상 종료된 곳의 소스코드 위치 표시 가능
  - ✓ procID
    - 이미 실행 중인 프로그램을 디버깅하고 싶을 때 사용
    - 디버깅할 process의 id(PID)가 인자로 사용
    - 인자로 사용한 PID와 같은 이름의 파일이 있을 경우, gdb는 core 파일로 인식함
  - ✓ option
    - -q : 시작메세지 숨김
    - -tui : curses gui 모드로 실행

| 명령어 | 설 명 | 예 |
|---|---|---|
| gdb [ prog [ core \| procID ] ] | gdb 시작 | gdb ./a.out |
| q(uit), Ctrl + d | gdb 종료 | q |

○ 소스코드 출력

| 명령어 | 설 명 | 예 |
|---|---|---|
| l(ist) | main함수를 기점으로 소스 출력 | l |
| l(ist) - | 출력된 행의 이전 행을 출력 | l - |
| l(ist) [N] | N행을 기준으로 출력 | l 10 |
| l(ist) [FUNC] | FUNC함수의 소스를 출력 | l main |
| l(ist) [FILE]:[N] | FILE의 N행을 기준으로 출력 | l test.c:10 |
| l(ist) [FILE]:[FUNC] | FILE의 FUNC함수의 소스를 출력 | l test.c:main |
| set listsize [N] | 출력되는 행의 수를 변경 (기본은 10행 출력) | set listsize 15 |

○ debugging 진행

| 명령어 | 설 명 | 예 |
|---|---|---|
| start | 프로그램 실행과 동시에 break | start |
| r(un) | 프로그램 실행 | r |
| r(un) arg1 arg2 ... | 인자를 이용한 프로그램 실행 | r 10 20 |
| k(ill) | 프로그램 종료 | k |
| s(tep) | 현재 행 수행 후 멈춤,<br>함수 호출 시 함수 내부로 들어감 | s |
| n(ext) | 현재 행 수행 후 멈춤,<br>함수 호출 시 함수 수행 후 다음 행으로 감 | n |
| c(ontinue) | 다음 브레이크포인트를 만날 때까지 계속 지행 | c |
| u(ntil) | 현재 loop를 빠져 나감 | u |
| finish | 현재 함수를 수행하고 빠져 나감 | finish |
| return | 현재 함수를 수행하지 않고 빠져 나감 | return |
| advance [LINE#] | 현재 파일의 LINE#행을 만날 때까지 진행 | advance 10 |

○ break point 추가

| 명령어 | 설 명 | 예 |
|---|---|---|
| b(reak) [FUNC] | FUNC 함수 시작 부분에 break point 설정 | b main |
| b [LINE#] | LINE#에 break point 설정 | b 15 |
| b [FILE:FUNC] | FILE의 FUNC 함수에 break point 설정 | b test.c:func |
| b [FILE:LINE#] | FILE의 LINE#에 break point 설정 | b test.c:15 |
| b +N | 현재 행 N개 행 이후 지점에 break point 설정 | b +2 |
| b -N | 현재 행 N개 행 이전 지점에 break point 설정 | b -2 |
| b *[ADDRESS] | ADDRESS 주소에 break point 설정<br>(어셈블리로 디버깅 시 사용함) | b *0x8060000 |
| b [LINE#] if [CONDITION] | LINE#에 break point설정,<br>단,CONDITION이 참일 경우에만 동작 | b 15 if i == 100 |
| condition [#] [CONDITION] | #번 break point에 CONDITION 설정 | condition 2 i == 100<br>condition 3 func(3) == 10 |
| ignore [#] [N] | #번 break point를 N번 무시 | ignore 1 10 |
| tb | 한번만 동작하는 break point 설정<br>사용법은 b와 동일 | tb 15 |
| rb [정규표현식] | 정규표현식에 일치하는 심볼에 모두<br>break point 설정 | rb oslab* / rb ^oslab |

○ break point 삭제

| 명령어 | 설 명 | 예 |
|---|---|---|
| cl(ear) [LINE#] | LINE#행의 break point를 삭제 | cl 10 |
| cl [FUNC] | FUNC의 break point를 삭제 | cl main |
| cl [FILE:FUNC] | FILE의 FUNC의 break point를 삭제 | cl test.c:func |
| cl [FILE:LINE#] | FILE의 LINE#행의 break point를 삭제 | cl test.c:15 |
| d | break point를 모두 삭제 | d |

○ break point 확인

| 명령어 | 설 명 | 예 |
|---|---|---|
| I(nfo) b(reakpoints) | 현재 설정된 break point 확인<br>(watchpoint확인) | info b |

○ break point 활성화/비활성화

| 명령어 | 설 명 | 예 |
|---|---|---|
| dis(able) b(reakpoints) | 모든 break point 비활성화 | disable b |
| dis b [#] | #번 break point 비활성화 | disable b 2<br>disable b 2 4 |
| en(able) b(reakpoints) | 모든 break point 활성화 | enable b |
| en b [#] | #번 break point 활성화 | enable b 2 |
| en b once [#] | #번 break point 한번만 활성화 | enable b once 1 |
| en b delete [#] | #번 break point 한번 작동 후 삭제 | enable b delete 1 |

○ 변수 값 확인

| 명령어 | 설 명 | 예 |
|---|---|---|
| info locals | 지역변수의 현재 값을 출력 | info locals |
| info variables | 전역변수의 현재 값을 출력 | info variables |
| p(rint) [변수명] | 변수의 현재 값을 출력 | p i |
| p(rint) [함수명] | 함수의 주소 값을 출력 | p func |
| p [변수명] = [값] | 변수를 값으로 변경함 | p i = 20 |
| display [변수명] | 변수 값을 매번 화면에 출력 | display var |
| undisplay [#] | #번호의 display를 설정을 없앰 | undisplay 1 |

○ watch point 설정

| 명령어 | 설 명 | 예 |
|---|---|---|
| watch [변수명] | 변수의 값이 변경될 때 멈춤 | watch sum |
| rwatch [변수명] | 변수의 값이 읽혀질 때 멈춤 | watch sum |
| awatch [변수명] | 변수가 변경되거나 읽혀질 때 모두 멈춤 | watch sum |

○ TUI 모드 Key Bindings

| 키 | 설 명 |
|---|---|
| Ctrl + x Ctrl + a | TUI 모드 전환 |
| Ctrl + x 1 | 1개 윈도우 |
| Ctrl + x 2 | 2개 윈도우 |
| Ctrl + x o | active 윈도우 변경 |
| Ctrl + x s | SingleKey 모드로 전환 |
| Ctrl + l | 화면 refresh |

```
:~/source/gdb$ vim bug.c
```

filepath : gdb/bug.c

```
#include <stdio.h>
#define NUM 5

int score[NUM];

int sum(int cnt){
        int i;
        int sum;

        for(i = 0; i < cnt ; i++){
                sum += score[i];
        }
        return sum;
}

int main()
{
        int i = 0;
        int cnt = 0;

        printf("input scores. input -1 to finish.\n");

        for(i = 0;i < NUM;i++)  {
                printf("score #%d : ", cnt+1);
                scanf("%d", score[cnt]);
                if(score[cnt] == -1)
                        break;
                cnt++;
        }

        printf("%d scores read.\n", cnt);
        printf("--- result ---\n");
        printf("sum : %d avg : %d\n", sum(cnt), sum(cnt)/cnt);

        return 0;
}
```

```
:~/source/gdb$ gcc -Wall -W -g bug.c
:~/source/gdb$ ./a.out
```

```
:~/source/gdb$ gcc -Wall -W -g bug.c
bug.c: In function ‘main’:
bug.c:25:9: warning: format ‘%d’ expects argument of type ‘int *’, but argument 2 has type ‘int’ [-Wformat=]
   scanf("%d", score[cnt]);
         ^
:~/source/gdb$
:~/source/gdb$ ./a.out
input scores. input -1 to finish.
score #1 : 102
세그멘테이션 오류 (core dumped)
:~/source/gdb$
```

```
:~/source/gdb$ gdb a.out -q
```

```
:~/source/gdb$ gdb a.out -q
Reading symbols from a.out...done.
(gdb) r
Starting program: /home/oslab/source/gdb/a.out
input scores. input -1 to finish.
score #1 : 102

Program received signal SIGSEGV, Segmentation fault.
_IO_vfscanf_internal (s=0xb7fbb5a0 <_IO_2_1_stdin_>, format=0x804865f "%d", argptr=0xbffff604 "", errp=0x0)
    at vfscanf.c:1902
1902    vfscanf.c: 그런 파일이나 디렉터리가 없습니다.
(gdb) bt
#0  _IO_vfscanf_internal (s=0xb7fbb5a0 <_IO_2_1_stdin_>, format=0x804865f "%d", argptr=0xbffff604 "",
    errp=0x0) at vfscanf.c:1902
#1  0xb7e6513e in __isoc99_scanf (format=0x804865f "%d") at isoc99_scanf.c:37
#2  0x08048520 in main () at bug.c:25
(gdb) q
A debugging session is active.

        Inferior 1 [process 3812] will be killed.

Quit anyway? (y or n) y
```

```
:~/source/gdb$ vim bug.c
```

~~25          scanf("%d", score[cnt]);~~

```
25          scanf("%d", &score[cnt]);
```

```
:~/source/gdb$ gcc -Wall -W -g bug.c
:~/source/gdb$ ./a.out
```

```
:~/source/gdb$ gcc -Wall -W -g bug.c
:~/source/gdb$ ./a.out
input scores. input -1 to finish.
score #1 : 90
score #2 : 80
score #3 : 70
score #4 : 60
score #5 : 50
5 scores read.
--- result ---
sum : -1217277252 avg : -243455520
:~/source/gdb$
```

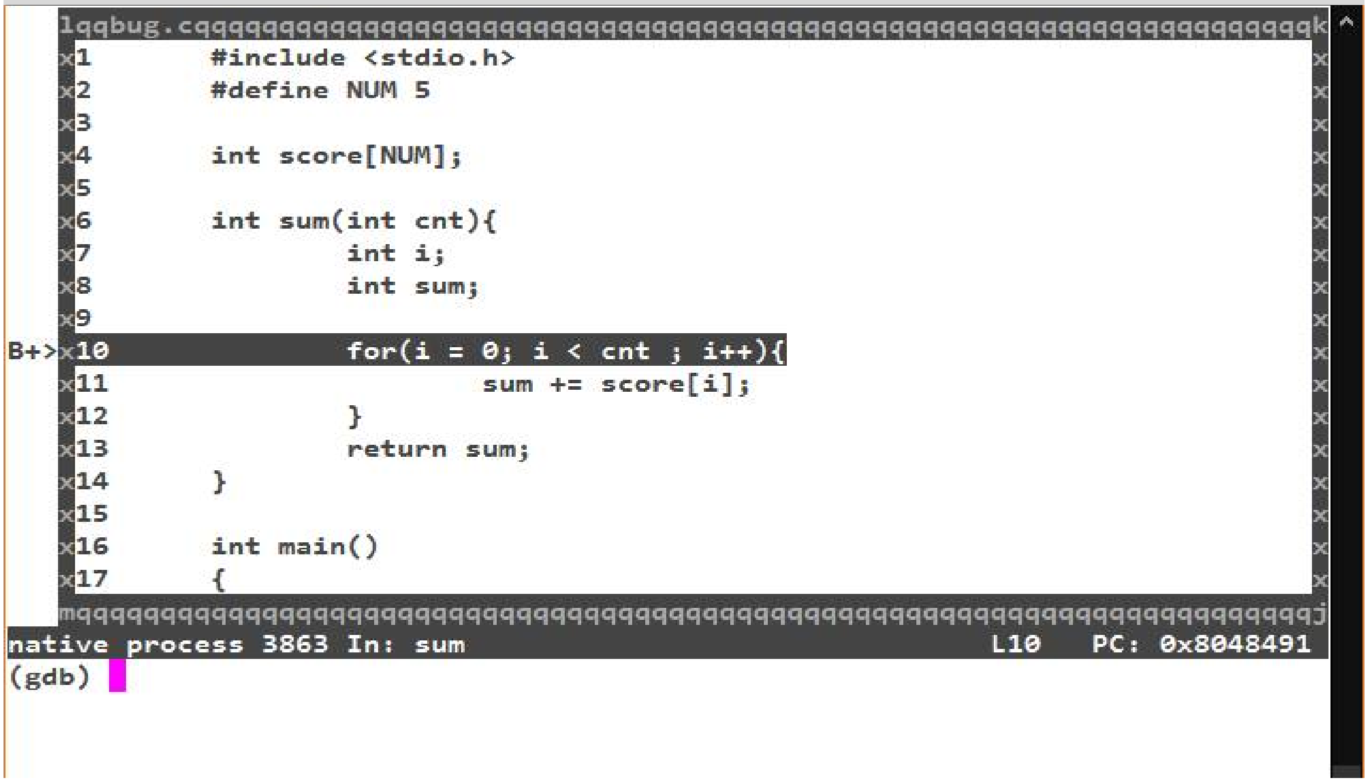
```
:~/source/gdb$ gdb a.out -q
```

```
:~/source/gdb$ gdb a.out -q
Reading symbols from a.out...done.
(gdb) b sum
Breakpoint 1 at 0x8048491: file bug.c, line 10.
(gdb) info b
Num     Type           Disp Enb Address    What
1       breakpoint     keep y   0x08048491 in sum at bug.c:10
(gdb) r
Starting program: /home/oslab/source/gdb/a.out
input scores. input -1 to finish.
score #1 : 90
score #2 : 80
score #3 : 70
score #4 : 60
score #5 : 50
5 scores read.
--- result ---

Breakpoint 1, sum (cnt=5) at bug.c:10
10              for(i = 0; i < cnt ; i++){
(gdb)
```

- TUI 모드로 전환

```
Ctrl + x, Ctrl + a
```



```
lqqbug.cqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqk
   x1       #include <stdio.h>                                                x
   x2       #define NUM 5                                                     x
   x3                                                                         x
   x4       int score[NUM];                                                   x
   x5                                                                         x
   x6       int sum(int cnt){                                                 x
   x7               int i;                                                    x
   x8               int sum;                                                  x
   x9                                                                         x
B+>x10              for(i = 0; i < cnt ; i++){                                x
   x11                      sum += score[i];                                  x
   x12              }                                                         x
   x13              return sum;                                               x
   x14      }                                                                 x
   x15                                                                        x
   x16      int main()                                                        x
   x17      {                                                                 x
   mqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqj
native process 3863 In: sum                                L10   PC: 0x8048491
(gdb)
```

```
(gdb) p sum
$1 = -1208242176
(gdb) n
11                      sum += score[i];
(gdb) p sum
$2 = -1208242176
(gdb) n
10              for(i = 0; i < cnt ; i++){
(gdb) p sum
$3 = -1208242086
(gdb)
```

```
:~/source/gdb$ vim bug.c
```

~~8      int sum;~~

```
8      int sum = 0;
```