제3장 C 프로그래밍 환경

학습 목표

- 문서 편집 : vi, geidt
- C 컴파일러 사용: gcc
- 컴파일 자동화: make
- 디버깅: gdb
- 통합개발환경: Eclipse
- 라이브러리 관리: ar
- 소스 관리: ctags
- 형상 관리: CVS, SVN, git

3.3 디버거

gdb

- 가장 대표적인 디버거
  - GNU debugger(gdb)
- gdb 주요 기능
  - 정지점(breakpoint) 설정
  - 한 줄씩 실행
  - 변수 접근 및 수정
  - 함수 탐색
  - 추적(tracing)
- gdb 사용을 위한 컴파일
  - -g 옵션을 이용하여 컴파일

```
$ gcc -g -o longest longest.c
```

  - 다중 모듈 프로그램

```
$ gcc -g -o main main.c copy.c
```

- gdb 실행

```
$ gdb [실행파일]      //e.g., $ gdb a.out
```

(소스 파일이 아닌 실행파일)

```
$ gdb [실행파일]
gdb 디버거는 실행파일을 이용하여 디버깅 모드로 실행한다.
```

gdb 기능

- 소스보기 : l(list)
  - l [줄번호] 지정된 줄을 프린트
  - l [파일명]:[함수명] 지정된 파일의 함수를 프린트
  - set listsize n 출력되는 줄의 수를 n으로 변경

(gdb) l copy 기본 10줄 출력

```
1 #include <stdio.h>
2 #include "copy.h"
3
4 /* copy: copy 'from' into 'to'; assume to is big enough */
5 void copy(char from[], char to[])
6 {
7 int i;
8
9 i = 0;
10 while ((to[i] = from[i]) != '₩0')
```

gdb 기능

- 정지점 : b(break), clear, d(delete)
  - b [파일:]함수 — 파일의 함수 시작부분에 정지점 설정
  - b n — n번 줄에 정지점을 설정
  - b +n — 현재 줄에서 n개 줄 이후에 정지점 설정
  - b -n — 현재 줄에서 n개 줄 이전에 정지점 설정
  - info b — 현재 설정된 정지점을 출력
  - clear 줄번호 — 해당 정지점을 삭제
  - d — 모든 정지점을 삭제

(gdb) b copy
Breakpoint 1 at 0x804842a: file copy.c, line 9.
(gdb) info b (현재 설정된 정지점의 정보 출력)
Num Type Disp Enb Address What
1 breakpoint keep y 0x0804842a in copy at copy.c:9

gdb 기능

- 프로그램 수행
  - r(run) 인수 — 명령줄 인수를 받아 프로그램 수행
  - k(kill) — 프로그램 수행 강제 종료
  - n(next) — 멈춘 지점에서 다음 줄을 수행하고 멈춤
  - s(step) — n과 같은 기능 함수호출 시 함수내부로 진입
  - c(continue) — 정지점을 만날 때 까지 계속 수행
  - u — 반복문에서 빠져나옴
  - finish — 현재 수행하는 함수의 끝으로 이동
  - return — 현재 수행중인 함수를 빠져나옴
  - quit — 종료

```
(gdb) r
Starting program: /home/chang/바탕화면/src/long
Merry X-mas !
Breakpoint 1, copy (from=0x8049b60 "Merry X-mas !", to=0x8049760 "")
at copy.c:9
9 i = 0;
```

gdb 기능

- 변수 값 프린트: p(rint)
  - p [변수명] 해당 변수 값 프린트
  - p 파일명::[변수명] 특정 파일의 전역변수 프린트
  - p [함수명]::[변수명] 특정 함수의 정적 변수 프린트
  - info locals 현재 상태의 지역변수 리스트

(gdb) p from
$1 = 0x8049b60 "Merry X-mas !“ (입력으로 Merry X-mas 입력)
(gdb) n
10 while ((to[i] = from[i]) != '₩0')
(gdb) n
11 ++i;
(gdb) p to
$2 = 0x8049760 "M“ (첫번째 글자인 M이 복사.. 이하 계속 복사 진행)

gdb 기능

(gdb) c

Continuing.

Happy New Year ! (입력)

Breakpoint 1, copy
(from=0x8049b60 "Happy New Year !",
to=0x8049760 "Merry X-mas !") at copy.c:9

9 i = 0;

(gdb) p from

$3 = 0x8049b60 "Happy New Year !"

(gdb) n

10 while ((to[i] = from[i])!='₩0')

(gdb) n

11 ++i;

(gdb) p to (Happy의 H 복사)

$4 = 0x8049760 "Herry X-mas !"

...

입력의 끝을 의미하는 control-D를
입력하면 가장 긴 줄 출력 후 종료

(gdb) c

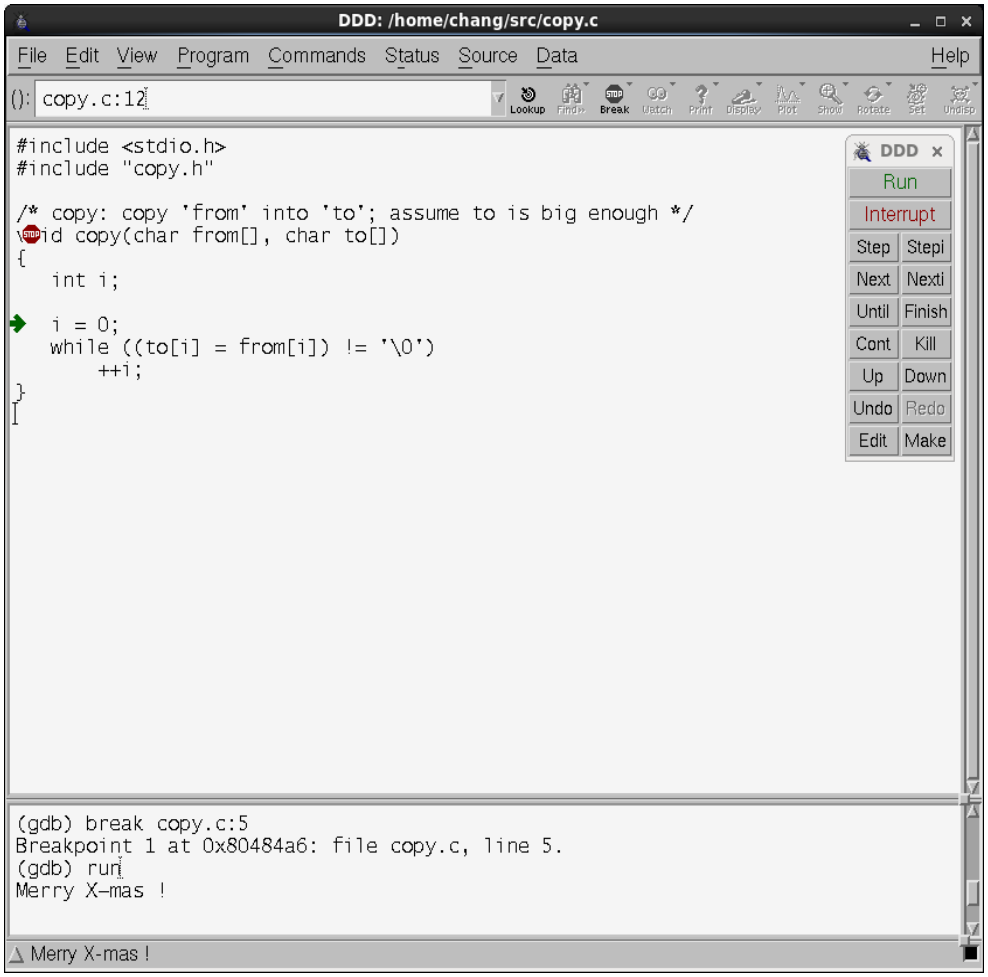
Continuing.

Happy New Year !

Program exited normally.

DDD(Data Display Debugger)

- gdb 디버거의 그래픽 사용자 인터페이스
  - GNU DDD 사용
  - http://www.gnu.org/software/ddd
  - DDD 프로그램의 소스코드도 함께 배포



GDB 실습 1

- Wrong program

```
/* simple gdb example code */
# include <stdio.h>
main()
{
    int i;
    double j;
    char *buf = NULL;
    for(i=0; i<5; i++){
                j = i/2 + i;
                printf(" j is %lf ₩n", j);
    }
    strcpy(buf, "hi");
    printf("buf is %s ₩n", buf);
    return;
}
```

(double)i/2 + i

char *buf = (char *)malloc(10 * sizeof(char)); // 10바이트 크기 할당

i/(double)2 + i

i/2.0 + i

프로그램은 에러없이 컴파일 되었음. 하지만,과연 올바른 프로그램인가?

GDB 명령어 종류

- 실행 및 트레이스에 관련된 명령들
  - run [arglist]
  - break {[file:] function | line number }
  - continue
  - next [n]
  - step [n] 함수 내부로 들어감
  - quit

- 데이터와 관련된 명령들
  - whatis variable-name
  - print
  - display
  - list [first, last]

GDB 명령어

- 실행 및 트레이스
  - run : 현재의 인수를 사용하여 프로그램을 실행
  - run <args> : 새로운 <인수>를 가지고 프로그램을 실행
  - continue : 현재 위치에서 프로그램을 계속 실행 (약자 c)

GDB 명령어

- next : 한 줄씩 실행 시킨다. 이 때 함수를 포함하고 있으면 함수를 수행시킨다. (약자) n
- next <n> : <n>줄을 실행시킨다.
- step : 한 줄씩 실행 시킨다. 이 때 함수를 포함하고 있으면 함수 내부로 들어가서 한 줄씩 실행한다. (약자) s
- step <n> : <n>줄을 실행시킨다.
- **(주의) step 사용시 원치 않은 함수로 들어가지 않도록 주의**

GDB 명령어

- break <line number> : 라인 번호에서 프로그램 실행을 멈추게 한다.
- break <함수 명> : 함수 내부의 첫번째 라인에서 프로그램의 실행을 멈추게 한다.
- quit : gdb를 종료 시킨다.

GDB 명령어

- 데이터에 관련된 명령들
  - whatis <expr> : 지정한 <변수>에 관련된 정보를 보여준다.
  - print <expr> : <expr>에 지정된 식의 값을 보여준다. (약자) p
  - display : 현재 지정된 display 명령의 목록을 보여준다.
  - display <expr> : 새로운 <expr>을 display목록에 추가
    - 지정된 목록의 반복 출력 기능 제공 display한 후 next하면 display 했던 값들이 나옴
    - undisplay, disable display, enable display
  - list : 현재 위치에서 소스 파일의 내용을 10줄 보여준다.
  - list <first>,<last> : <시작줄>과 <끝줄>사이의 소스파일 내용을 보여준다.

GDB의 사용예

```
1 #include <stdio.h>
2
3 main()
4 {
5        int         i;
6       double j;
7        char    *buf = NULL;
8
9       for( i = 0; i < 5; i++) {
10         j = i/2 + i;
11         printf(" j is %lf ₩n", j );
12       }
13       strcpy(buf, "hi");
14       printf("buf is %s ₩n", buf);
15
16       return;
17       }
<끝>
```

GDB의 사용예

- 컴파일과 실행

```
[sugar@hussein gdb]$ cc -g test.c
[sugar@hussein gdb]$ ./a.out
 j is 0.000000
 j is 1.000000
 j is 3.000000
 j is 4.000000
 j is 6.000000
Segmentation fault (core dumped)
[sugar@hussein gdb]$
[sugar@hussein gdb]$ ls
a.out  core  test.c
```

GDB의 사용예

- GDB의 시작
  - $ gdb <실행화일>

```
[sugar@hussein gdb]$ gdb a.out
GNU gdb 4.17.0.11 with Linux support
Copyright 1998 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "i386-redhat-linux"...
(gdb)
(gdb)
```

GDB의 사용예

```
(gdb) list main
1   #include <stdio.h>
2   main()
3   {
4       int i;
5       double j;
6       char *buf = (char *)malloc(3 * sizeof(char));
7       for (i = 0; i<5; i++){
8           j = i/2 + i;
9           printf("j is %lf \n", j);
10      }
```

- list명령

```
(gdb) list
1       #include <stdio.h>
2
3       main()
4       {
5               int     i;
6               double j;
7               char    *buf = NULL;
8
9               for( i = 0; i < 5; i++) {
10                      j = i/2 + i;
(gdb)
```

함수의 처음부터 10줄(기본)

(gdb) list [행번호] | [함수명]

위아래 5줄씩(기본)

```
(gdb) list 4, 13
4       {
5               int     i;
6               double j;
7               char    *buf = NULL;
8
9               for( i = 0; i < 5; i++) {
10                      j = i/2 + i;
11                      printf(" j is %lf \n", j );
12              }
13              strcpy(buf,"hi");
(gdb)
```

GDB의 사용예

- break , run 명령
  - break point의 지정, run으로 break point까지 실행

```
(gdb) break 9
Breakpoint 1 at 0x804840d: file test.c, line 9.
(gdb) run
Starting program: /home/users/phd/sugar/bit/gdb/a.out

Breakpoint 1, main () at test.c:9
9                   for( i = 0; i < 5; i++) {
(gdb)
```

GDB의 사용예

- next, print 명령

```
(gdb) n
10                      j = i/2 + i;
(gdb) p i
$1 = 0
(gdb) p j
$2 = 4.8699524093964861e-270
(gdb) n
11                      printf(" j is %lf ₩n",
    j );
(gdb) p i
$3 = 0
```

```
(gdb) p j
$4 = 0
(gdb) n
 j is 0.000000
9               for( i = 0; i < 5; i++) {
(gdb)
```

GDB의 사용예

- display 명령 : 변수명과 함께 다음 실행에서도 계속 결과 제시

```
(gdb) display i
1: i = 0
(gdb) display j
2: j = 0
(gdb) n
10                  j = i/2 + i;
2: j = 0
1: i = 1
(gdb) n
11                  printf(" j is %lf ₩n", j );
2: j = 1
1: i = 1
```

```
(gdb) n
 j is 1.000000
9           for( i = 0; i < 5; i++) {
2: j = 1
1: i = 1
(gdb) n
10                  j = i/2 + i;
2: j = 1
1: i = 2
(gdb) n
11                  printf(" j is %lf \n", j );
2: j = 3
1: i = 2
(gdb)
```

GDB의 사용예

...next명령어 계속...

```
(gdb) n
10                      j = i/2 + i;
2: j = 4
1: i = 4
(gdb) n
11                      printf(" j is %lf ₩n",
    j );
2: j = 6
1: i = 4
(gdb) n
 j is 6.000000
9               for( i = 0; i < 5; i++) {
2: j = 6
1: i = 4
```

GDB의 사용예

```
(gdb) n
13              strcpy(buf,"hi");
2: j = 6
1: i = 5
(gdb) n

Program received signal SIGSEGV,
   Segmentation fault.
strcpy (dest=0x0, src=0x80484ec "hi")
   at ../sysdeps/generic/strcpy.c:38
../sysdeps/generic/strcpy.c:38: No such
   file or directory.
(gdb) q
```

J값이 이상하다!

→가령, 1.000000의 값을 위해 j를 double 로 선언했을까?

→ 만약 다음과 같이 사용되었다면 1.500000이 출력됨.

j = (double)i/2 + i;

```
j is 0.000000
j is 1.000000
j is 3.000000
j is 4.000000
j is 6.000000

j is 0.000000
j is 1.500000
j is 3.000000
j is 4.500000
j is 6.000000
```

GDB의 사용예

```
[sugar@hussein gdb]$ gdb a.out
GNU gdb 4.17.0.11 with Linux support
...생략...
(gdb) b 13
Breakpoint 1 at 0x8048460: file test.c,
    line 13.
(gdb) run
Starting program:
    /home/users/phd/sugar/bit/gdb/a.o
    ut
 j is 0.000000
 j is 1.000000
 j is 3.000000
 j is 4.000000
 j is 6.000000
```

```
Breakpoint 1, main () at test.c:13
13              strcpy(buf,"hi");
(gdb) p buf
$1 = 0x0

(gdb)
```

buf의 주소값이 잘못되어 있다!

→ buf = (char *)calloc(3, sizeof(char));
buf에 메모리 공간을 할당 후 사용해야 함.

GDB 실습 2

- 예제프로그램
  - (vi debug.c에서 set number 명령 실행)

```
_____1_#include <stdio.h>
_____2_
_____3_void print_sum(sum)
_____4_int sum
_____5_int      sum;
_____6_{
_____7_         printf("Total sum : %d₩n", sum);
_____8_}
_____9_
_____10_main()
_____11_{
_____12_        int     i, sum;
_____13_
_____14_        sum = 0;
_____15_        for(i=0;i<5;i++) {
_____16_                printf("%dth interation\n", i );
_____17_                sum += i;
_____18_        }
_____19_        print_sum(sum);
_____20_}
```

GDB 실습 2

[sugar@hussein bit]$ gcc -g debug.c

[sugar@hussein bit]$ gdb a.out

GNU gdb 4.17.0.11 with Linux support

Copyright 1998 Free Software Foundation, Inc.

....생략....

(gdb) **break 14**

Breakpoint 1 at 0x80483ee: file debug.c, line 14.

(gdb) **run**

Starting program: /home/users/phd/sugar/bit/a.out

Breakpoint 1, main () at debug.c:14

14sum = 0;

(gdb) **next**

15for(i=0;i<5;i++) {

(gdb) **next**

16printf("%dth interation\n", i);

(gdb) **n**

0th interation

17sum += i;

(gdb)

GDB 실습 2

```
(gdb) n
15for(i=0;i<5;i++) {
(gdb) n
16printf("%dth interation₩n", i );
(gdb) n
1th interation
17sum += i;
(gdb) n
15for(i=0;i<5;i++) {
(gdb) n
16printf("%dth interation₩n", i );
(gdb) n
2th interation
17sum += i;
```

```
(gdb) n
15for(i=0;i<5;i++) {
(gdb) n
16printf("%dth interation\n", i );
(gdb) n
3th interation
17sum += i;
(gdb)
```

GDB 실습 2

(gdb) n
15for(i=0;i<5;i++) {
(gdb) **print sum**
$1 = 6
(gdb) **print i**
$2 = 3
(gdb) **display sum**
1: sum = 6
(gdb) **display i**
2: i = 3

(gdb) next
16printf("%dth interation\n", i);
2: i = 4
1: sum = 6
(gdb) next
4th interation
17sum += i;
2: i = 4
1: sum = 6
(gdb)

GDB 실습 2

(gdb) next
15for(i=0;i<5;i++) {
2: i = 4
1: sum = 10
(gdb) next
19print_sum(sum);
2: i = 5
1: sum = 10

(gdb) next
Total sum : 10
20}
2: i = 5
1: sum = 10
(gdb) **c**
Program exited with code 017.
(gdb) **quit**
[sugar@hussein bit]$

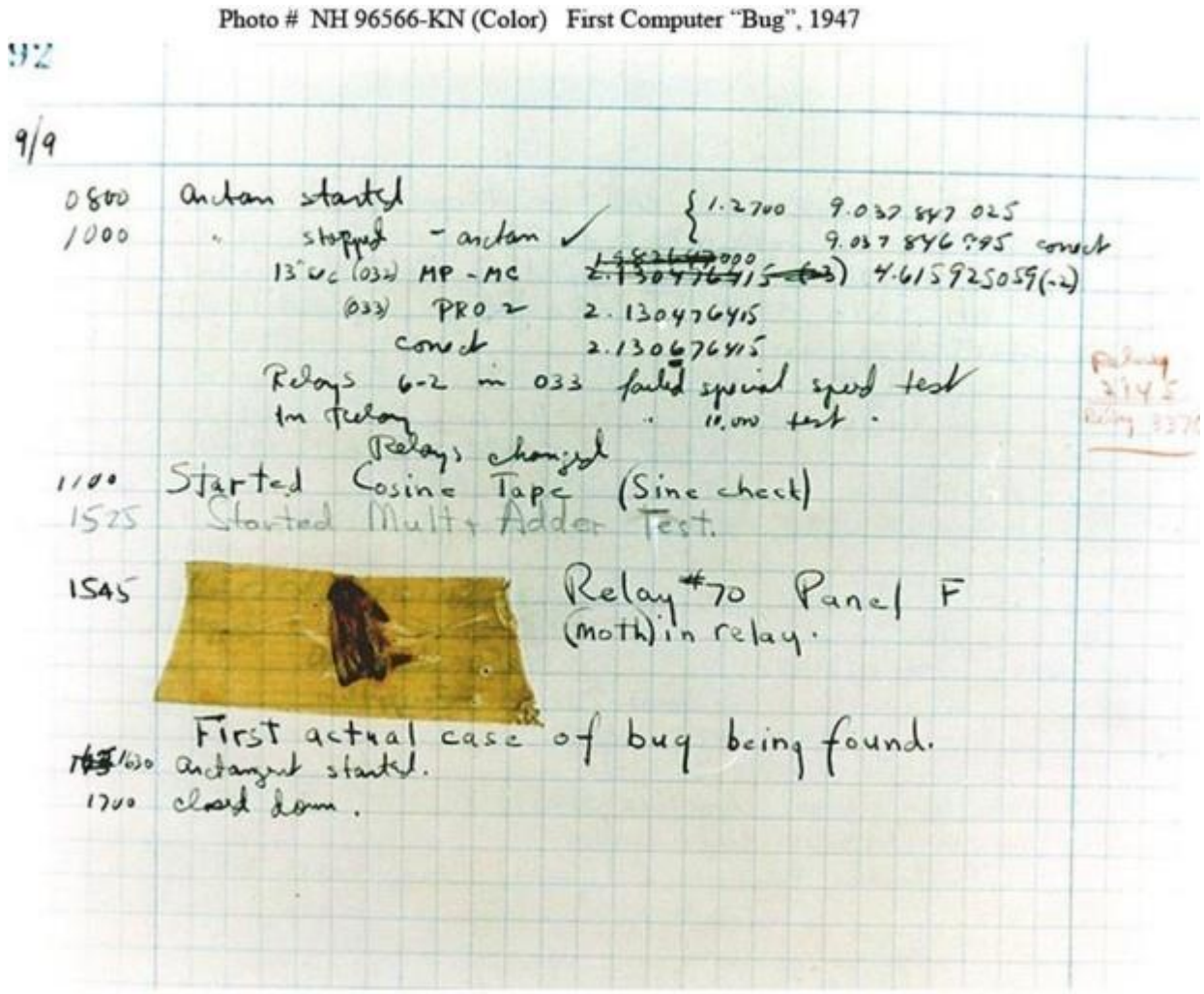
버그가 없이 동작함을 확인!

GDB 실습 3 : Finding Bugs…

```
#include <stdio.h>
#define S_TO_M 60

void  main(void)
{
        int sec, min, left;
          sec = 1;
        printf("Enter the number of seconds\n");
        printf("Enter 0 to end the program\n");
        while(sec > 0) {
                scanf("%d",  sec);   &sec
                min = sec / S_TO_M;
                left = sec % S_TO_M;
                printf("%d sec is %d min, %d sec\n",sec, min, left);
                printf("Next input?\n");
        }
    }
```

The Bug by Grace Hopper



이클립스 통합개발환경

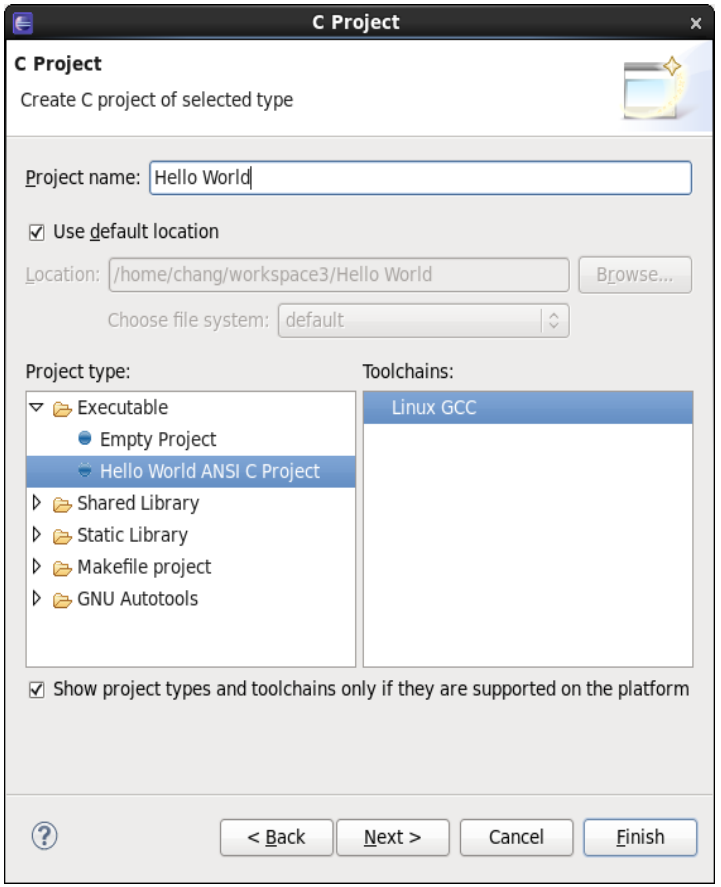
이클립스(Eclipse)

- 통합 개발 환경
  - 윈도우, 리눅스, 맥 등의 다양한 플랫폼에서 사용 가능
  - 다양한 언어(C/C++, Java 등)를 지원
  - 막강한 기능을 자랑하는 Free Software

- 이클립스 설치
  - 리눅스용 이클립스를 다운받아 설치가능
  - https://www.eclipse.org

새로운 C 프로젝트를 생성하기

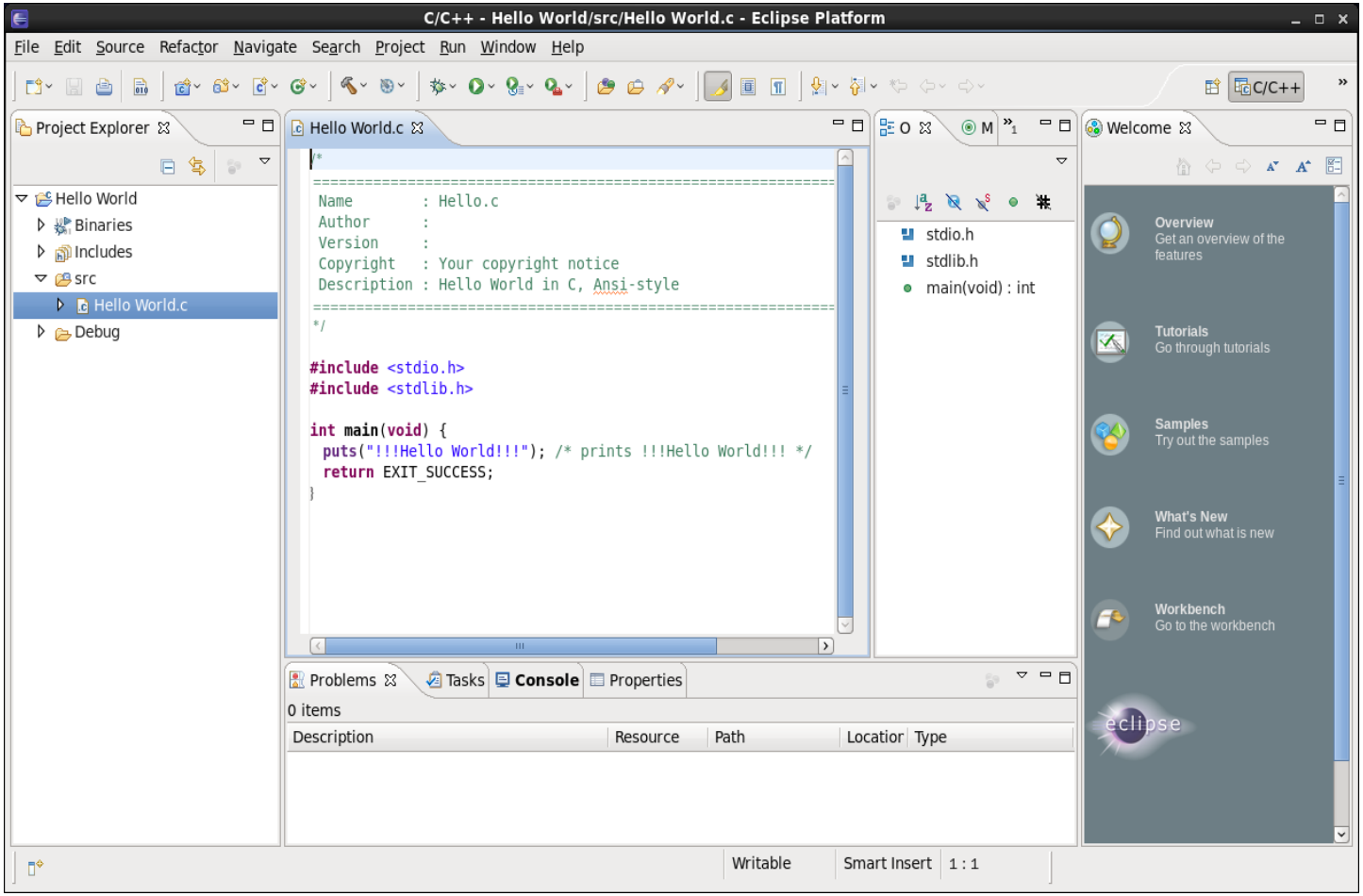
- 'File →New → C/C++ Projects 선택
- 프로젝트 이름을 지정
- 프로젝트 타입 Hello World ANSI C Project 선택
- Finish 버튼 클릭하면 간단한 HelloWorld.c 프로그램 자동 생성
- 프로젝트 타입 Empty Project 선택하면 빈 프로젝트 생성



이클립스 메인화면

- 좌측 탐색 창:
  - 새로 생성된 프로젝트 확인 및 프로젝트, 파일 탐색
  - 소스 파일은 src 폴더에 헤더 파일은 include 폴더에 저장됨
- 중앙
  - 상단은 소스 및 각종 파일 등을 편집 수정할 수 있는 창
  - 하단은 C 파일을 컴파일 혹은 실행한 결과를 보여주는 창
- 화면의 우측
  - 이클립스 사용법을 보여준다.

이클립스 메인화면


C/C++ - Hello World/src/Hello World.c - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
C/C++
Project Explorer
Hello World
Binaries
Includes
src
Hello World.c
Debug
Hello World.c
/*
Name : Hello.c
Author :
Version :
Copyright : Your copyright notice
Description : Hello World in C, Ansi-style
*/
#include <stdio.h>
#include <stdlib.h>
int main(void) {
puts("!!!Hello World!!!"); /* prints !!!Hello World!!! */
return EXIT_SUCCESS;
}
stdio.h
stdlib.h
main(void) : int
Welcome
Overview
Get an overview of the features
Tutorials
Go through tutorials
Samples
Try out the samples
What's New
Find out what is new
Workbench
Go to the workbench
eclipse
Problems
Tasks
Console
Properties
0 items
Description
Resource
Path
Locatior
Type
Writable
Smart Insert
1 : 1