13장 프로그래밍 도구

gcc 옵션

옵션	설명
--help	gcc 옵션에 대한 간략한 설명을 볼 수 있다.
-c	링크를 하지 않고 컴파일만 수행함, 컴파일 결과는 .o 파일로 생성됨
-D *id[=value]*	id 매크로를 정의함, value가 없으면 id는 1의 값을 가짐
-E	전처리기만 수행함, 전처리 결과는 화면에 출력됨
-g	gdb 디버거가 사용할 수 있는 디버깅 정보를 만드는 코드 생성
-I *dir*	*dir* 디렉터리에서 헤더파일을 찾음
-L *dir*	라이브러리 패스에 *dir* 디렉터리를 추가함
-l*lib*	lib로 명시된 라이브러리를 찾음
-M	makefile 생성
-o *name*	실행 파일 이름을 *name*으로 함
-O*level*	코드 최적화 수준을 선택함, level: 0, 1, 2, 3, s
-pg	gprof 프로파일러가 사용할 수 있는 프로파일 정보를 만드는 코드 생성
-S	어셈블러 코드 생성, .s 파일로 생성됨
-std=*standard*	어떤 표준으로 작성된 프로그램인지 지정함, standard: c99, c89, ansi 등

make

- 대형 프로그램을 한 파일에 저장하는 것은 매우 비효율적이고 많은 비용이 소요됨
- 대형 프로그램은 보통 여러 개의 .c와 .h 파일로 나누어 작성됨
  - 보통 한 디렉터리에 둠
- 여러 파일로 구성된 프로그램은 개별 파일을 컴파일하는 것이 훨씬 효율적임
  - make를 사용하면 쉽게 할 수 있음
  - make는 makefile(Makefile, GNUmakefile)이라는 파일을 읽고, 종속관계 트리를 만들어 필요한 동작을 수행함

makefile

- makefile은 종속관계와 동작을 명시하는 규칙이라고 하는 목록들로 구성됨
- 규칙의 일반적인 형태

  목표 : 종속관계 . . .

  　　명령

  　　. . .

  – 목표를 이루기 위해 종속관계 . . . 파일들이 관련있고 명령을 수행함

make

- makefile 예제

```
grade: main.c grade.c grade.h
    gcc -o grade main.c grade.c
```

  - grade를 만들기 위해 main.c, grade.c, grade.h 파일이 연관되고, gcc -o grade main.c grade.c를 수행함

- make 실행

  $ make 또는 $ make grade

  - makefile의 레이블(목표)는 옵션
  - make는 현 디렉터리에 있는 makefile을 읽어 수행

make

- makefile을 만들 때 각 파일을 따로 컴파일 하도록 하는 것이 좋음

```
grade: main.c grade.c grade.h
    gcc -o grade main.c grade.c
```

  - main.c와 grade.c 중 하나만 수정되어도 두 파일을 모두 컴파일 함

make

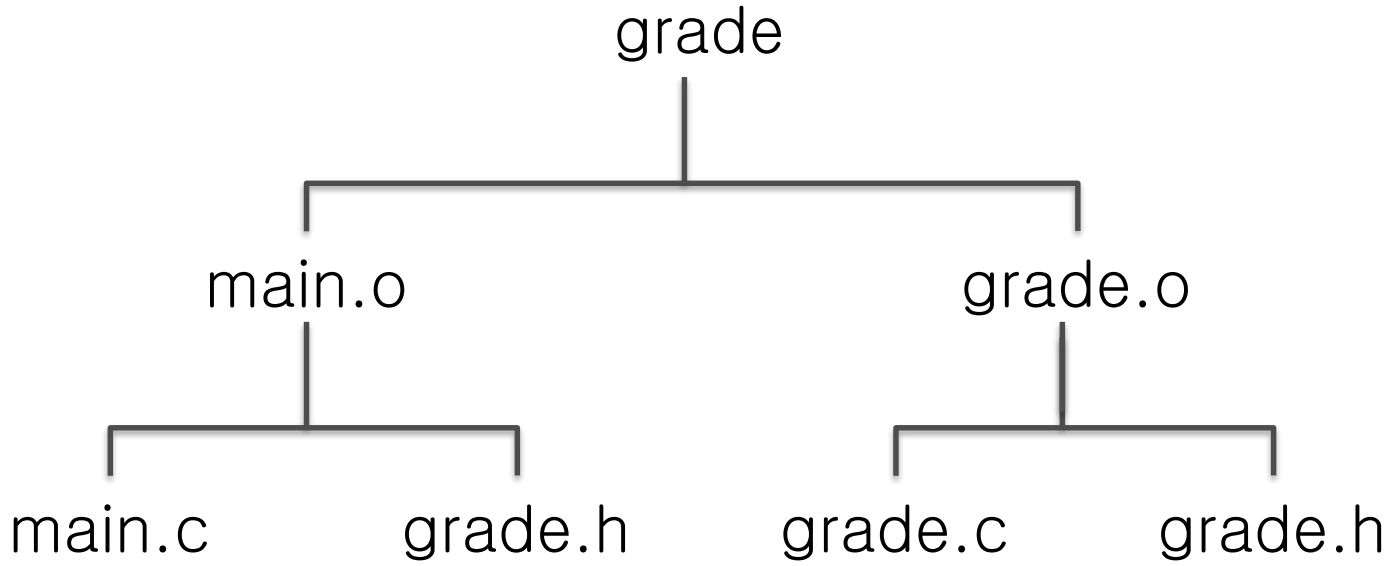
- makefile을 만들 때 각 파일을 따로 컴파일 하도록 하는 것이 좋음

```
grade: main.o grade.o
	gcc -o grade main.o grade.o
main.o: main.c grade.h
	gcc -c main.c
grade.o: grade.c grade.h
	gcc -c grade.c
```

  - main.c와 grade.c를 따로 컴파일 하게 함

종속트리

- make는 makefile을 읽어 종속 트리를 만든 다음 파일이 만들어진 시간을 비교하여 명령을 실행할 지를 결정함



makefile

- make는 다양한 편의 기능을 가지고 있음
  - .o 파일은 대응되는 .c 파일에 종속됨

```
grade: main.o grade.o
        gcc -o grade main.o grade.o
main.o: grade.h
        gcc -c main.c
grade.o: grade.h
        gcc -c grade.c
```

makefile

- make는 여러 가지 내장된 매크로를 지원함

```
grade: main.o grade.o
        gcc -o grade main.o grade.o
main.o grade.o: grade.h
        gcc -c $*.c
```

makefile

- makefile에서 매크로를 선언할 수 있음

```
# grade Makefile
BASE    =  /home/kmh/grade
CC      =  gcc
CFLAGS  =  -O -Wall
EFILE   =  $(BASE)/bin
OBJS    =  main.o  grade.o

$(EFILE): $(OBJS)
        @echo "linking ..."
        @$(CC)  $(CFLAGS) -o $(EFILE)  $(OBJS)
$(OBJS): grade.h
        $(CC)  $(CFLAGS)  -c $*.c
```

touch

- 파일에 새로운 시간 설정
- make는 파일 시간을 보고 동작할 것인지 결정
- 다시 전체를 컴파일 하고 싶을 때 touch를 수행 후 make 함
- 예

```
$ touch *.c
$ make
```

gprof

- gprof는 GNU 프로파일러로 프로그램의 실행 프로파일을 보여줌
- gprof를 사용하기 위해서는 프로그램을 컴파일할 때 -pg 옵션을 사용해야 함
- -pg 옵션을 설정하면 컴파일러는 각 함수의 호출 횟수와 실행 시간을 수집하는 코드를 실행 파일에 삽입함
- 실행 정보는 gmon.out 파일에 저장됨
- gprof는 gmon.out 파일을 사용하여 실행 프로파일을 생성함

gprof

- gprof 사용

```
$ gcc -pg -o quicksort quicksort.c
$ quicksort
$ gprof -b quicksort
---- 프로파일 출력 ------
```

- gprof는 플랫 프로파일과 호출 그래프를 출력함
  - 플랫 프로파일은 각 함수의 호출 횟수와 실행 시간을 보여줌
  - 호출 그래프는 각 함수가 어떤 함수를 얼마나 호출하는 지를 보여줌

prof 출력

```
$ gprof -b quicksort
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
 50.00      0.06     0.06     9999     0.01     0.01  partition
 25.00      0.09     0.03    19999     0.00     0.00  find_pivot
 16.67      0.11     0.02        1    20.00    20.00  write_data
  8.33      0.12     0.01        1    10.00    10.00  gen_data
  0.00      0.12     0.00  1581610     0.00     0.00  swap
  0.00      0.12     0.00        1     0.00    90.00  quicksort
```

prof 출력

Call graph

```
granularity: each sample hit covers 4 byte(s) for 8.33% of 0.12 seconds

index % time    self  children    called     name
                                                 <spontaneous>
[1]    100.0    0.00    0.12                 main [1]
                0.00    0.09       1/1           quicksort [2]
                0.02    0.00       1/1           write_data [5]
                0.01    0.00       1/1           gen_data [6]
-----------------------------------------------
                                19998            quicksort [2]
                0.00    0.09       1/1           main [1]
[2]     75.0    0.00    0.09       1+19998   quicksort [2]
                0.06    0.00    9999/9999        partition [3]
                0.03    0.00   19999/19999       find_pivot [4]
                                19998            quicksort [2]
-----------------------------------------------
                0.06    0.00    9999/9999        quicksort [2]
[3]     50.0    0.06    0.00    9999         partition [3]
                0.00    0.00 1573023/1581610     swap [9]
```

prof 출력

```
                0.03    0.00   19999/19999       quicksort [2]
[4]     25.0    0.03    0.00   19999          find_pivot [4]
                0.00    0.00    8587/1581610      swap [9]
-----------------------------------------------
                0.02    0.00       1/1           main [1]
[5]     16.7    0.02    0.00       1          write_data [5]
-----------------------------------------------
                0.01    0.00       1/1           main [1]
[6]      8.3    0.01    0.00       1          gen_data [6]
-----------------------------------------------
                0.00    0.00    8587/1581610      find_pivot [4]
                0.00    0.00 1573023/1581610      partition [3]
[9]      0.0    0.00    0.00 1581610          swap [9]
-----------------------------------------------


Index by function name

   [4] find_pivot              [3] partition               [9] swap
   [6] gen_data                [2] quicksort               [5] write_data
```

gdb

- 대화식 디버거
- gdb는 프로그램을 실행하고, 원하는 곳에서 실행을 중지하고, 또 재실행하며, 변수의 값을 설정하거나 볼 수 있는 기능을 제공 함
- gdb를 사용할 때에는 컴파일 할 때 -g 옵션을 사용해야 함

gdb 실행

```
$ gcc -g -o sort sort.c
$ gdb sort
GNU gdb 6.5.50.20060706-cvs (cygwin-special)
Copyright (C) 2006 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 "i686-pc-cygwin"...
(gdb)
```

← gdb 프롬프트

gdb 실행

```
(gdb) run      ← gdb에서 프로그램 실행
Starting program: /home/tool/sort.exe
Loaded symbols for /cygdrive/c/WINDOWS/system32/ntdll.dll
Loaded symbols for /cygdrive/c/WINDOWS/system32/kernel32.dll
Loaded symbols for /usr/bin/cygwin1.dll
Loaded symbols for /cygdrive/c/WINDOWS/system32/advapi32.dll
Loaded symbols for /cygdrive/c/WINDOWS/system32/rpcrt4.dll

Program received signal SIGSEGV, Segmentation fault.
0x004010d9 in sort (data=0x7, n=7) at sort.c:13
13                  data[j] = tmp;
(gdb)
```

gdb 실행

```
(gdb) list 13
8           for (i = 0; i < n - 1; i++)
9              for (j = n - 1; j < n; j--)
10                if (data[j - 1] > data[j]){
11                   tmp = data[j - 1];
12                   data[j - 1] = data[j];
13                   data[j] = tmp;
14                }
15       }
16       int main(void)
17       {
(gdb) l 7, 13
7           int  i, j, tmp;
8           for (i = 0; i < n - 1; i++)
9              for (j = n - 1; j < n; j--)
```

← 소스 코드 보기(list 또는 l)

← 구간 지정 가능

gdb 실행

```
(gdb) print n
$1 = 7
(gdb) p i
$2 = 0
(gdb) p j
$3 = -15
(gdb) p main::a
$4 = {3379, 4346, 7005, 8489, 5214, 8806, 5844, 3411, 8422, 7729}
(gdb)
```

← 변수 값 보기(print 또는 p)

gdb 실행

```
(gdb) set var j = 10
(gdb) p j
$5 = 10
(gdb) p data[0]
Cannot access memory at address 0x7
(gdb) set var data = main::a
(gdb) p data[0]
$6 = 3379
(gdb)
```

← 변수 값 설정(set var)

gdb 실행

```
(gdb) break main
Breakpoint 1 at 0x401119: file sort.c, line 17.
(gdb) break sort
Breakpoint 2 at 0x401057: file sort.c, line 8.
(gdb) break 9
Breakpoint 3 at 0x40106b: file sort.c, line 9.
(gdb) info break
Num Type           Disp Enb Address    What
1   breakpoint     keep y   0x00401119 in main at sort.c:17
2   breakpoint     keep y   0x00401057 in sort at sort.c:8
3   breakpoint     keep y   0x0040106b in sort at sort.c:9
```

← 검사점 설정(break 함수명)

← 검사점 설정(break 행번호)

← 검사점 확인(info break)

gdb 실행

```
(gdb) clear 17
Deleted breakpoint 1
(gdb) clear sort
Deleted breakpoint 2
(gdb) info break
Num Type           Disp Enb Address    What
3   breakpoint     keep y   0x0040106b in sort at sort.c:9
(gdb)
```

← 검사점 삭제(clear)

gdb 실행

```
(gdb) run
Starting program: /home/tool/sort.exe
Loaded symbols for /cygdrive/c/WINDOWS/system32/ntdll.dll
Loaded symbols for /cygdrive/c/WINDOWS/system32/kernel32.dll
Loaded symbols for /usr/bin/cygwin1.dll
Loaded symbols for /cygdrive/c/WINDOWS/system32/advapi32.dll
Loaded symbols for /cygdrive/c/WINDOWS/system32/rpcrt4.dll

Breakpoint 1, main () at sort.c:19      ← 검사점에서 중지
19          srand(time(NULL));
(gdb) step                  ← 한 행씩 실행(step)
20          for (i = 0; i < N; ++i)
(gdb) step
21              a[i] = rand() % 10000;
(gdb) continue              ← 다음 검사점까지 실행(continue)
Continuing.
```

gdb 실행

```
(gdb) break 10 if j < 0
Breakpoint 1 at 0x40107a: file sort.c, line 10.
(gdb) run
Starting program: /home/tool/sort.exe
Loaded symbols for /cygdrive/c/WINDOWS/system32/ntdll.dll
Loaded symbols for /cygdrive/c/WINDOWS/system32/kernel32.dll
Loaded symbols for /usr/bin/cygwin1.dll
Loaded symbols for /cygdrive/c/WINDOWS/system32/advapi32.dll
Loaded symbols for /cygdrive/c/WINDOWS/system32/rpcrt4.dll

Breakpoint 1, sort (data=0x23cc70, n=10) at sort.c:10
10                if (data[j - 1] > data[j]){
(gdb) p j
$1 = -1
(gdb)
```

← 검사점에 조건 명시(if) 가능

gdb 실행

```
(gdb) backtrace
#0  sort (data=0x23cc70, n=10) at sort.c:10
#1  0x0040118c in main () at sort.c:22
(gdb)
```

← 함수 호출 관계 보기(backtrace)

gdb 실행

```
(gdb) help      ← help 기능 (help)
List of classes of commands:

aliases -- Aliases of other commands
breakpoints -- Making program stop at certain points
data -- Examining data
files -- Specifying and examining files
internals -- Maintenance commands
obscure -- Obscure features
running -- Running the program
stack -- Examining the stack
status -- Status inquiries
support -- Support facilities
tracepoints -- Tracing of program execution without stopping the program
user-defined -- User-defined commands

Type "help" followed by a class name for a list of commands in that class.
Type "help" followed by command name for full documentation.
Command name abbreviations are allowed if unambiguous.
(gdb)
```

gdb 실행

```
(gdb) help breakpoints
(gdb)
Making program stop at certain points.
List of commands:
awatch -- Set a watchpoint for an expression
break -- Set breakpoint at specified line or function
catch -- Set catchpoints to catch events
clear -- Clear breakpoint at specified line or function
commands -- Set commands to be executed when a breakpoint is hit
condition -- Specify breakpoint number N to break only if COND is true
. . .
rwatch -- Set a read watchpoint for an expression
tbreak -- Set a temporary breakpoint
tcatch -- Set temporary catchpoints to catch events
thbreak -- Set a temporary hardware assisted breakpoint
watch -- Set a watchpoint for an expression

Type "help" followed by command name for full documentation.
Command name abbreviations are allowed if unambiguous.
(gdb)
```

← 각 항목 자세히 보기 기능 (help)