Computer Programming

**gdb** and **awk**  13th Lecture

엄현상 (Eom, Hyeonsang)

컴퓨터공학부

서울대학교

©Copyrights 2010 Eom, Hyeonsang All Rights Reserved
순서

- gdb
- awk
- Q&A
gdb 기본

- Features
  - Run a Program
  - Stop Execution within the Program
  - Examine and Change Variables during Execution
  - Trace How the Program Executes
  - Provide Command-Line Editing and History Features
Using gdb

Compilation with the –g Option

- Generate an Expanded Symbol Table for Use with gdb

```
martini:~$ gcc –g main.c func.o
```

Execution of gdb

- Permit Debugging a Program Compiled with the –g Option
- Permit Discovering Where and Why the Program Failed (with a Core Dump)

```
martini:~$ gdb a.out
martini:~$ gdb a.out core
```
Core dump: copy of memory image; run ‘ulimit –c unlimited’ to make the size unlimited
**gdb Basic Commands**

- **help**: Print a List of Commands or Topics
- **run**: Execute the Program
- **quit**: Exit gdb

```shell
(gdb) help
List of classes of commands
... (omitted)
running -- Running the program
...
(gdb) help running
...
run -- Start debugged program
...
(gdb) run 1
...
(gdb) quit
```

- **Argument list**: `[Ctrl]–C to stop the program`
gdb Example

```c
/* test.c */
struct {
    int *i_p;
} s;
void foo() {
    *(s.i_p) = 2;
    printf("s.i = %d\n", *(s.i_p));
}
main(int argc, char *argv[]){
    int *i_p;
    *i_p = atoi(argv[1]);
    printf("i = %d\n", *i_p);
    foo();
}
```

```
martini:~$ gcc -g test.c
martini:~$ a.out 1
i = 1
세그멘테이션 오류 (core dumped)
```
gdb Example (계속)

- **list**: Show the Lines Surrounding the Code Just Executed
- **whatis**: Print the Type of a Variable or Function
- **ptype**: Show the Contents of a Data Type

```c
martini:~$ gcc a.out core
... (omitted)
Program terminated with signal 11, Segmentation fault.
...
#0 0x0804846b in foo () at test.c:6
6     *(s.i_p) = 2;
(gdb) list
...  
6     *(s.i_p) = 2;
...  
(gdb) whatis s
  type = struct {...}
(gdb) ptype s
  type = struct {
    int *i_p;
  }
```

Call stack number 0: most recent

Line number

more powerful than whatis
**gdb Example (계속)**

- **print**: Print the Value of a Variable or Expression
- **up/down**: Move Up/Down the Stack Frame to Make another Function the Current One

```
(gdb) print s
$1 = {i_p = 0x0}
(gdb) print s.i_p
$2 = (int *) 0x0
(gdb) print *(s.i_p)
Cannot access memory at address 0x0
(gdb) up
#1 0x080484c9 in main (argc=2, argv=0xbffffce4) at test.c:15
15    foo();
(gdb) print i_p
$3 = (int *) 0xbffffffc98
(gdb) print *i_p
$4 = 1
(gdb) print $4+1
$5 = 2
```

‘print func(...’ possible
Value history identifiers
The error occurred at ‘*(s.i_p) = 2;’
gdb Example

- backtrace/where: Print the Current Location and a Stack Trace
- set variable: Assign a Value to a Variable
- print: Print the Assigned Value

martini:~$ gdb a.out
... (omitted)
(gdb) run 1
...
Program received signal SIGSEGV, Segmentation fault.
0x0804846b in foo () at test.c:6
6 *(s.i_p) = 2;
(gdb) backtrace
#0 0x0804846b in foo () at test.c:6
#1 0x080484c9 in main (argc=2, argv=0xbffffcb4) at test.c:15
(gdb) up
#1 0x080484c9 in main (argc=2, argv=0xbffffcb4) at test.c:15
15 foo();
(gdb) set variable (*i_p)++
(gdb) print *i_p += 1
$1 = 3
same as ‘where’

Assignments do not work with core
**gdb Example (계속)**

- **break**: Set Breakpoints in the Program
- **watch**: Set “Break-If” Breakpoints
- **info breakpoints**: Show All Breakpoints and Watchpoints

```
martini:~$ gdb a.out
... (omitted)
(gdb) break foo
Breakpoint 1 at 0x8048466: file test.c, line 6.
(gdb) break test.c:6
Note: breakpoint 1 also set at pc 0x8048466.
Breakpoint 2 at 0x8048466: file test.c, line 6.
(gdb) run
...
Breakpoint 1, foo () at test.c:6
6     *(s.i_p) = 2;
(gdb) watch *i_p > 0
Hardware watchpoint 3: *i_p > 0
(gdb) info breakpoints
...```
**gdb Example (계속)**

- **kill**: Abort the Running Process
- **delete**: Delete a Breakpoint or a Watchpoint
- **next**: Execute the Next Line, Executing a Function

```
(gdb) kill
Kill the program being debugged? (y or n) y
(gdb) delete 2
(gdb) delete 1
(gdb) break 11
Breakpoint 4 at 0x8048495: file test.c, line 11.
(gdb) run 1
Breakpoint 1, main (argc=2, argv=0xbffffcb4) at test.c:12
12        *i_p = atoi(argv[1]);
(gdb) print *i_p
$1 = -1073742712
(gdb) next
13        printf("i = %d \n", *i_p);
(gdb) print *i_p
$2 = 1
```
```plaintext
(gdb) kill
Kill the program being debugged? (y or n) y
(gdb) run 1
... (omitted)
(gdb) step
13    printf("i = %d \n", *i_p);
(gdb) step
i = 1
15    foo();
(gdb) step
foo () at test.c:6
6     *(s.i_p) = 2;
(gdb) continue
...```
**gdb Example (계속)**

- **info line:** Show Where the Object Code Begins and Ends
- **disassemble:** Produce a Machine Listing
- **info registers:** List Integer Registers and Their Contents

(gdb) kill
Kill the program being debugged? (y or n) y
(gdb) run 1
... (omitted)
(gdb) info line 12
Line 12 of "test.c" starts at address 0x8048495 <main+9> and ends at 0x80484af <main+35>.
(gdb) disassemble main
... 
(gdb) x 0x80484ca
0x80484ca <main+62>:    0x895590c3
(gdb) info registers
...
awk (Aho, Weinberger, Kernighan)

PL for Computing and Data-Manipulation Tasks (Especially, Columns of Data)

Example

<table>
<thead>
<tr>
<th>EXECUTION TIME = 2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXECUTION TIME = 4.2</td>
</tr>
</tbody>
</table>

```
martini:~$ awk '{sum = sum + $4} END {print "avg = " sum/NR}' temp
```

```
martini:~$ awk -f avg.awk temp
```