

Development using FOSS tools

Mandar Mitra

Indian Statistical Institute

Outline

- 1 Languages
- 2 Compiling
- 3 Debugging
- 4 Memory related tools
- 5 Editors
- 6 IDEs

- 1 Languages
- 2 Compiling
- 3 Debugging
- 4 Memory related tools
- 5 Editors
- 6 IDEs

Languages



+ perl, bash/shell scripts, awk, sed, grep, ...

Outline

1 Languages

2 Compiling

3 Debugging

4 Memory related tools

5 Editors

6 IDEs

GCC: commonly used flags

What to generate?

- `-o`: specify output file

```
gcc -o program program.c ⇒ program
```

- `-c`: compile / assemble but do not link

```
gcc -c program.c ⇒ program.o
```

- useful when combining multiple source files into executable / library

- `-S`: generate assembly

```
gcc -S program.c ⇒ program.s
```

- `-fverbose-asm`: put extra commentary information in the generated assembly code to make it more readable (useful if you actually need to read the generated assembly code)

GCC: commonly used flags

- `-g`: produces debugging information in the operating system's native format

```
gcc -g -o program program.c
```

- `-Wall`, `-Wextra`: enables all the warnings about constructions that some users consider questionable, and that are [usually] easy to avoid (or modify to prevent the warning)

```
gcc -Wall -g -o program program.c
```

- `-O`, `-O2`: optimise the compiled code

- Check calls to `printf` and `scanf`, etc., to make sure that the arguments supplied have types appropriate to the format string specified
- Warn if parentheses are omitted in certain contexts
- Warn when a declaration does not specify a type (assumed `int`)
- Warn whenever a function is defined without a return-type, or on return type mismatches
- Warn if an automatic variable is used without first being initialized
- ...

GCC: warnings

```
#include <stdio.h>
#include <stdlib.h>
main(int argc, char *argv[ ])
{ int i, j;
  printf("%c\n", "not a character");
  if (i = 10)
    if (j != 10)
      printf("another oops\n");
  else
    no_decl();
  return(EXIT_SUCCESS);
}
void no_decl(void) { printf("no_decl\n"); }
```

GCC: warnings

```
#include <stdio.h>
#include <stdlib.h>
main(int argc, char *argv[ ])           // return type defaults to int
{ int i, j;
  printf("%c\n", "not a character");     // wrong argument type
  if (i = 10)                            // parentheses!
    if (j != 10)                         // uninitialised j
      printf("another oops\n");
  else                                    // ambiguous else
    no_decl();                            // implicit declaration
  return(EXIT_SUCCESS);
}
void no_decl(void) { printf("no_decl\n"); }
```

Libraries, etc.

- `-I`: add a directory to the head of the list of directories to be searched for header files
- `-L`: add a directory to the list of directories to be searched for linked libraries
- `-l`: search the named library when linking
 - order is important

```
gcc -I/a/b/include -L/a/b/lib -o program program.c -lm
```

Outline

- 1 Languages
- 2 Compiling
- 3 Debugging**
- 4 Memory related tools
- 5 Editors
- 6 IDEs

GDB: getting started

- To debug a program a.out:

```
$ gdb a.out
```

- To start running the program:

```
(gdb) run
```

- To find out where a fault occurred:

```
(gdb) where
```

```
(gdb) backtrace
```

- To view code around this point:

```
(gdb) list
```

- Can use unambiguous abbreviations

GDB: example

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(int argc, char **argv)
5 {
6     char *buf;
7
8     buf = malloc(1<<31);
9
10    fgets(buf, 1024, stdin);
11    printf("%s\n", buf);
12
13    return 1;
14 }
```

GDB: example

```
$ gdb segfault
GNU gdb (Ubuntu/Linaro 7.4-2012.04-0ubuntu2) 7.4-2012.04
...
Reading symbols from /home/mandar/Dropbox/present/linux/examples/segfault..
(gdb) r
Starting program: /home/mandar/Dropbox/present/linux/examples/segfault
hallo

Program received signal SIGSEGV, Segmentation fault.
_IIO_getline_info (fp=0x7ffff7dd4340, buf=0x0, n=1023, delim=10,
    extract_delim=1, eof=0x0) at iogetline.c:91
91 iogetline.c: No such file or directory.
(gdb) bt
#0  _IIO_getline_info (fp=0x7ffff7dd4340, buf=0x0, n=1023, delim=10,
    extract_delim=1, eof=0x0) at iogetline.c:91
#1  0x00007ffff7a8bafb in _IIO_fgets (buf=0x0, n=<optimized out>,
    fp=0x7ffff7dd4340) at iofgets.c:58
#2  0x00000000004005fe in main (argc=1, argv=0x7ffffffffffd6c8) at
    segfault.c:10
(gdb)
```

GDB: breakpoints

- To stop a program at a particular position:

```
(gdb) break main
```

```
(gdb) break 8
```

```
(gdb) break segfault.c:8
```

- To continue running the program:

```
(gdb) continue
```

- To continue execution one step at a time:

```
(gdb) next
```

```
(gdb) step
```

- To continue execution until end of a called function:

```
(gdb) finish
```


- To navigate between functions (stack frames)

```
(gdb) up
```

```
(gdb) down
```

```
(gdb) frame 2
```

- To see values of variables

```
(gdb) print buf
```

GDB: example

```
(gdb) b main
Breakpoint 1 at 0x4005d3: file segfault.c, line 8.
(gdb) r
Starting program: /home/mandar/Dropbox/present/linux/examples/segfault

Breakpoint 1, main (argc=1, argv=0x7fffffff6c8) at segfault.c:8
8      buf = malloc(1<<31);
(gdb) n
10     fgets(buf, 1024, stdin);
(gdb)
```

GDB: example

```
(gdb) cont
```

```
Continuing.
```

```
abc
```

```
Program received signal SIGSEGV, Segmentation fault.
```

```
_IO_getline_info (fp=0x7ffff7dd4340, buf=0x0, n=1023, delim=10,  
    extract_delim=1, eof=0x0) at iogetline.c:91
```

```
91    iogetline.c: No such file or directory.
```

```
(gdb) frame 2
```

```
#2  0x0000000004005fe in main (argc=1, argv=0x7fffffff6c8) at segfault.c:
```

```
10    fgets(buf, 1024, stdin);
```

```
(gdb) p buf
```

```
$1 = 0x0
```

GDB: example

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(int argc, char **argv)
5  {
6      char *buf;
7
8      buf = malloc(1<<31);
9
10     fgets(buf, 1024, stdin);
11     printf("%s\n", buf);
12
13     return 1;
14 }
```

GDB: more about breakpoints

- `(gdb) break file1.c:6 if i >= ARRAYSIZE`
- `(gdb) condition 1 (i >= ARRAYSIZE)`
- `(gdb) delete 1` (use (optional) breakpoint number)
`(gdb) clear main` (use breakpoint location)
- `(gdb) disable 1`
`(gdb) enable 1`
- `(gdb) tbreak`
- `(gdb) info breakpoints`

GDB: more about printing

- print accepts expressions (including type casts, &, *, etc.)

```
(gdb) print (char) x
```

- To print an array:

```
(gdb) print buffer[2]@16
```

- To find out type of a variable:

```
(gdb) whatis buf
```

```
(gdb) ptype argc
```

- To stop execution whenever the value of an expression changes:

```
(gdb) watch x
```

```
(gdb) watch *(int *)0x12345678
```

```
(gdb) watch a*b + c/d
```

- To stop execution when an expression is read by the program:

```
(gdb) rwatch x
```

- To stop execution when an expression is read / written:

```
(gdb) awatch x
```

- Type control-C to interrupt an infinite loop
- Use `quit` or control-D to exit
- Type return to repeat previous command
- `(gdb) help <command name>`

Outline

- 1 Languages
- 2 Compiling
- 3 Debugging
- 4 Memory related tools**
- 5 Editors
- 6 IDEs

<http://dmalloc.com/>

- In source: `#include "dmalloc.h"`
- Link the dmalloc library into your program.
- Output:
`not freed: '0x45048' (10 bytes) from 'argv.c:1077'`
`WARNING: tried to free(0) from foo.c:708`
`ERROR: heap_check: free space was overwritten`

<http://valgrind.org/>

■ Usage: `valgrind leak-check=yes myprog arg1 arg2`

■ Output:

```
==19182== Invalid write of size 4
==19182==    at 0x804838F: f (example.c:6)
==19182==    by 0x80483AB: main (example.c:11)
==19182== Address 0x1BA45050 is 0 bytes after a block of size 40 alloc'd
==19182==    at 0x1B8FF5CD: malloc (vg_replace_malloc.c:130)
==19182==    by 0x8048385: f (example.c:5)
==19182==    by 0x80483AB: main (example.c:11)
```

- 1 Languages
- 2 Compiling
- 3 Debugging
- 4 Memory related tools
- 5 Editors**
- 6 IDEs

- bluefish <http://bluefish.openoffice.nl/index.html>
- gedit <http://projects.gnome.org/gedit/>
- jEdit <http://www.jedit.org/>
- kate <http://kate-editor.org/>
- nano <http://www.nano-editor.org/>
- SciTE <http://www.scintilla.org/SciTE.html>
- vim <http://www.vim.org/>



http://en.wikipedia.org/wiki/Comparison_of_text_editors

Outline

- 1 Languages
- 2 Compiling
- 3 Debugging
- 4 Memory related tools
- 5 Editors
- 6 IDEs**

- Eclipse <http://www.eclipse.org/>
- Geany <http://www.geany.org/>
- KDevelop <http://kdevelop.org/>
- NetBeans <http://netbeans.org/>



- Content-sensitive editing modes, including syntax coloring
- Highly customizable, using Emacs Lisp
- Many extensions
- Complete built-in documentation, including a tutorial for new users
- Full Unicode support for nearly all human languages and their scripts

- Automatic compiling / building tools: make, ant
- Version control systems: bazaar, cvs, git, mercurial, subversion
- Bug tracking: bugzilla, trac

References

- Overview:

<http://www.slideshare.net/sagara10/foss-tools>

- GCC

<http://www.pearsonhighered.com/samplechapter/0672320215.pdf>

- GDB

<http://www.dirac.org/linux/gdb/>

Search for “gdb tutorial”

- Editors

http://en.wikipedia.org/wiki/List_of_text_editors

<http://www.linuxlinks.com/article/20080824052425167/Editors.html>

<http://tuxarena.blogspot.in/2009/04/>

[14-most-popular-text-editors-for-linux.html](http://tuxarena.blogspot.in/2009/04/14-most-popular-text-editors-for-linux.html)

- IDEs

http://en.wikipedia.org/wiki/Comparison_of_integrated_development_environments

<http://www.linuxlinks.com/article/20090620114618990/IDE.html>

<http://wiki.python.org/moin/IntegratedDevelopmentEnvironments>

<http://www.mojavelinux.com/wiki/doku.php?id=javaidecomparison>