C Debugging Workshop
using gdb

Jürgen Weigert
Documentation & Legal Team
openSUSE.org

2011-06-30

Novell

Overview

- What is a Bug?
- General bug hunting techniques
- What to expect from GDB
- Working with GDB
- Bug hunting by example
- Further outlook

What is a Bug?

… where gdb might help …

- Program crashes
  segmentation fault, signal 11
- The infinite loop
  eating 100% CPU, permanently
- Misbehavior
  faulty logic, corrupt data
- Blocking
  program waits without success

What is a Bug? -2-

Other bugs ...

- Web interface issue
- Slow execution
- Memory leak
- Compile time error
- Documentation error
- Configuration error
- Architectural/Design flaw

... need other tools

firebug

strace
ltrace
valgrind
printf()

lint

...

lots of practice

General Bug Hunting Techniques (preparation)

- Reproduce & reduce the bug
  - What is needed to repeat the bug reliably?
  - What can be removed before the bug disappears?
- Data collection (symptoms)
  - Locate logfiles, config files, make screenshots
- Check your expectations
  - Define expected outcome, read documentation
- Install a build environment
  - Unpack sourcecode, take care of dependencies,

General Bug Hunting -2-
(initial steps)

- Increase output verbosity
  - **`--verbose`** / **`-v`** options, (shell script '**`set -x`**')
  - Add **`printf()`**s
- Compare other versions
  - Same bug in older versions? (Patches?)
  - Other distributions, other revisions (**`svn co -r`**)
- Narrow a location by bisecting
  - Within a file: comment out systematically
  - Use revision control systems (**`git bisect`**)

General Bug Hunting -3-
(typical steps)

- Increase output verbosity
  - Write own **`main()`** for code fragments/libraries
  - Write a wrapper shell script, for easy reproduction
- Log protocols
  - Systemcalls (**`strace`**), Library calls (**`ltrace`**)
  - Memory usage (**`valgrind`**)

```
int a[10]; a[10] = 13;
char *u; if (strlen(u) > 0) ...
```

  - Crashdumps, collect stack backtraces (**`gdb`**)

General Bug Hunting -4-
(advanced steps)

- Study reference documentation
  - Description of library functions (**man 3**)
  - Know your system calls (**man 2**)
- Call for help
  - Query an expert
  - Use bugzilla
    - https://bugzilla.novell.com/page.cgi?id=bug-writing.html
    - https://bugzilla.novell.com/docs/html/bugreports.html
    - http://en.opensuse.org/Bugs#Reporting_a_Bug
    - https://innerweb.novell.com/organizations/engineering/pqsc/Defect+Management+Process.pdf

General Bug Hunting -5-
(wrapping up)

- Document your surgery
  - Add comments, ChangeLog entries
- Regression testing
  - Run the existing test-suite, (if any)
  - Write a new test that would reproduce the now fixed bug
- Submit code
  - Increment version number?
  - Create patch, send it upstream
  - **`svn checkin`**; **`osc ci`**; **`git commit`** , **`push`**; ...

What to Expect from GDB

"Oh no, it's an old command line tool!"

$ gdb *program core*

- List source code, see stack backtrace, inspect variables (Post Mortem Analysis)

$ gdb *program processID*
$ gdb –args *program parameters ...*

- Start, interrupt, list code, inspect state
- Change variables, make function calls
- Single step, continue to run, breakpoints

Working With GDB

```
$ gdb
(gdb) print 3*4
$1 = 12
```

Important commands

run	**set**	**print**	**list**	***CTRL-C***
where	**up**	**down**	**step**	**next**
break	**cont**	**disable**	**enable**	**help**

Expression syntax

- As known from C: (gdb) **`p/t (3*32|0x10)>>4`**
- Array printing: (gdb) **`p Prime[0]@50`**

Working With GDB -2-
environment

$ **ulimit -c unlimited**

- allow coredumps

$ **gcc -g -Wall -O0**

- tune Makefile: CFLAGS, LDFLAGS
  compile with debuginfo, without optimization

Install debuginfo packages

- for inspecting libraries

Prepare two or three shell windows

- to see your editor, compiler, and debugger all at once

Bug Hunting by Example

```
$ wget ftp.suse.de:/pub/people/jw/gdb/prime-0.3.tar.gz

$ tar xvf prime-0.3.tar.gz
$ cd prime-0.3
$ cc -o prime main.c prime.c

$ ./prime
```

Bitte obere Schranke eingeben: **10**

2 ist Primzahl

3 ist Primzahl

5 ist Primzahl

7 ist Primzahl

... that is what we want to see!

Try it – good luck!

Further Outlook

Avoiding bugs

- Test driven development, **`assert()`**
- Respect compiler warnings & **`lint`**

C++ demangling

- Symbol names and signatures, QT4 debugging

Network debugging

- Multiple interacting programs, Web UI

Graphical interfaces to gdb

- **`ddd`**, **`eclipse`**

Further Outlook -2-
gdb limitations

- Gdb cannot find syntax errors
  - use **`lint`** and **`gcc -Wall -O2`** for this
- Gdb can change variables, but not code
- Optimized code breaks the model
- Gdb cannot step backwards
- Preprocessor macros are invisible
- The cause of a bug often remains hidden
- No support for scripting languages
  - Perl, python, ruby, ... have their own debuggers, which often work similar.

References

$ info gdb

http://www.gnu.org/software/gdb/documentation

https://bugzilla.novell.com/page.cgi?id=bug-writing.html

http://en.opensuse.org/openSUSE:Submitting_bug_reports

ftp://ftp.suse.com/pub/people/jw/gdb