GNU Debugger Workshop

Jürgen Weigert

2011, opensuse.org, RWX³

GNU Debugger Workshop

Jürgen Weigert

SUSE Software Archeology: the '80ies

What to Expect from GDB ?

```
$ 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

“Oh no, it's an old-fashioned command line tool!”

What is a Bug?

… where gdb might help you …

- Program crash
  segmentation fault, signal 11

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

- Misbehavior
  faulty logic, corrupt data, eating 100% CPU, …

```
while (select(10, … ) read(10, … );
```

- *And much more …*

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

gdb limitations

- Gdb cannot find syntax errors
  - Use e.g. **`lint`** and **`gcc -Wall -O2`**
- Gdb does not mix well with optimization
  - Use **`gcc -g -O0`**
- Preprocessor macros are invisible to gdb
- Gdb cannot step backwards

- With gdb you often just explore symptoms
  - The cause may remain hidden
- Beware of interpreters written in C
  - Perl, python, ruby, javascript ... have their own debuggers

General Bug Hunting Techniques

- Reproduce & reduce the bug
  - What is needed to repeat the bug?
  - What can be removed before the bug disappears?

- Data collection (symptoms)
  - Locate logfiles, config files, take screenshots

- Check your expectations
  - Define expected outcome, read documentation

General Bug Hunting -2-

- Increase output verbosity
  - **`--verbose`** / **`-v`** options,
  - Add **`printf()`**s
- Compare other versions
  - Same bug in older versions? (Patches?)
  - Other revisions (**`svn co -r`**)
- Narrow a location by bisecting
  - Comment out code systematically
  - Use revision control systems (**`git bisect`**)

Working With GDB
an example

```
$ cat furlong.c
main() {
  const char yards_fu = 220;
  int ft_fu = 3 * yards_fu;
  printf("feet per furlong: %d\n",
    ft_fu);  // 660, no?
}
$ gcc -O0 -g -o furlong -c furlong.c
./furlong
feet per furlong: -108
```

(qqp) p ft fu

(gdp) p ft fu

Working With GDB -2-
expression syntax

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

As known from C:

```
(gdb) p/t (3*32|0x10)>>4
```

Array printing:

```
(gdb) what Prime
type = int [50]
(gdb) p Prime[0]@50
```

Working With GDB -3-
important commands

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

Working With GDB -4-
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!

Let's get our hands dirty!

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!

Thank You!

GNU Debugger Workshop

Jürgen Weigert

2011, opensuse.org, RWX³

GNU debugger Workshop

Jürgen Weigert

2011, opensuse.org, RWX³

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`**

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

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`**; …