Introduction to GDB

(or how to debug like a pro)

Mistral Contrastin

November 13, 2014

"What is wrong with printf?" – *Radu*

GDB as a Swiss Army Knife

- Debugger (duh)
- Memory inspector
- Disassembler
- Runtime
- Patcher
- Hex editor
- ...

Start with the basics

Compilation	`$ gcc -g <file>`
Start	`$ gdb <binary>`
Get to `main`	`> break main`
Inspect code	`> list`
Execute	`> run`
Run line by line	`> next` / `> step`
Inspect variable	`> print var`

Unleash the power of **breakpoints** and **watchpoints**

Lets you stop a program at arbitrary lines or at even deeper granularity.

Set	`> break [filename:]<line number> [if expr]`
	`> watch [filename:]<line number> [if expr]`
Conditional	`> cond <breakpoint number> expr`
List	`> info breakpoints`
Delete	`> delete <breakpoint number>`
Temporary	`> tbreak <line number>`
Function	`> break <function name>`
Instruction	`> break *<memory address>`

Inspect memory

Inspect locals `> info local`

Inspect arguments `> info args`

Stack trace `> where / backtrace / info stack`

Stack trace limit `> set backtrace limit <n>`

Inspect memory, cont'd

General memory inspection format is `p x/nfu`, where n is repeat count, f is display format and u is the unit size.

Options for display format

bytes	b
halfwords	h
words	w
giant words	g

Common display formats

decimal	d
hexadecimal	x
octal	o
string	s
instruction	i

Text User Interface

Enter	`C-x a` within GDB or `$ gdb -tui`
Exit	`C-x a`
Single key	`C-x s`

Single key mode bindings:

run	r
continue	c
next	n
step	s
where	w
info locals	v
exit	q

Use **core files** to save time

- `$ ulimit -c <number of bytes>`
- Linux dumps at the current working directory with name `core`
- For better core file name on Linux (`<filename>_<pid>.core`)
  ```
  $ sudo su
  $ echo %e_%p.core > /proc/sys/kernel/core_pattern
  ```
- Mac dumps at `/cores` with name `core.<PID>`
- Task manager > Right click > Create Dump File
  (please use Cygwin instead)
- `gdb <file> <core>`

C debugging tips and tricks

- Principle of Confirmation
- "Premature optimization is the root of all evil."

  – *Donald Knuth*
- ↑ Except when you use it as a lynt
- Always compile with -Wall -Wextra -pedantic

Go down the rabbit hole

- Attaching processes
- Remote debugging
- Kernel debugging
- Hardware and memory breakpoints
- Multiple TUI windows
- LLDB

Thank you!