Manipulating time with GDB

How to use GDB to perform time travel debugging

By Guinevere Larsen
time manipulation with GDB

How to use GDB to perform time travel debugging

How you can help making GDB better at manipulating time

By Guinevere Larsen
Summary

- Introduction
- How does it work
- Where the bugs come from
- A plea to help us fix them!
What are you talking about?

(gdb) reverse-continue
Continuing.

No more reverse-execution history.
0x000000000040112a in main ()
Who are you, lady?

Nice to meet you, I’m Guinevere!

I am hired to work on GDB, and have been doing this for close to 3 years.

I am the a maintainer of (one of) the relevant area of GDB.

I also like helping new contributors out, and time travel debugging is full of easy bugs to start.
And what is this GDB business?

A time wizard’s best friend!

The GNU Debugger allows you to stop time for the inferior\(^1\), or slowly execute it, and see how it ticks.

Useful for C, C++, Ada, Fortran, and much more!

1. Inferior: GDB lingo for “program being debugged”
Time travel debugging

Also called reverse debugging but that’s boring

Lets instructions be undone, meaning you can see where things went wrong

The talk related to rr explains the idea and why it is great

If you didn’t manage to catch it, just use what I teach today to see it later
How is that possible?

(gdb) help record
record, rec
Start recording.

List of record subcommands:

record btrace, record b -- Start branch trace recording.
record full -- Start full execution recording.
Example:
```
addl $0x1, -0x8(%rbp)
```

- **Memory:**
  - Address: -0x8 (%rbp)
  - Value: 00 00 00 00
  - Length: 4 bytes

- **Register:**
  - Register: %eflags
  - Values: [CF AF SF IF]

- **Register:**
  - Register: %rip
  - Values: 0x4011ef

End
GDB Recording

Pro:
● Comes in a single tool
● Fully reconstructs the state

Con:
● Slow
● Harder to support

GDB disassembles one instruction
Store all the information that is overwritten in a linked list
Tells the inferior to execute the instruction
Repeat or stop the execution

Record Details
GDB recording
Simple issues

Spaghetti code:
The main disassembly function is 3 thousand lines strong and almost unreadable.

Auxiliary functions and structs could also be better documented.

Disassembly:
We need to teach GDB about every instruction it needs to support.
Longer Example

PC = 0x00 → PC = 0x04 → PC = 0x18 → PC = 0x20 → PC = 0x24

PC 0x400400
Size 4
Type Arithmetic

PC 0x400404
Size 8
Type Control Flow

PC 0x400418
Size 8
Type Control Flow

PC 0x400420
Size 4
Type Arithmetic

PC 0x400424
Size 4
Type Arithmetic
GDB btrace

Pro:
- Comes in a single tool
- Fast

Con:
- Only restores the PC
- Only on some hardware

GDB tells the inferior to run:
The CPU stores trace data in a specific region of memory.

Once the inferior stops, GDB queries the kernel for that area of memory.

It then stores the PC, size and type of instruction for all recorded ones.

1. The region is called the Branch Trace Store (BTS) area.

Record Details
GDB btrace issues

Testsuite regressions
Assertion errors
Usability issues

Easy access to bugzilla search

Record Details
Different long Example

Checkpoint

Instructions

Checkpoint

Memory
Registers
Threads
Signals

Memory
Registers
Threads
Signals
RR can record the execution of a program outside of a debugger. It then saves the execution log to your disk. Finally it starts a gdbserver\(^1\) that is able to use this log to move back and forth.

---

\(^1\) Gdbserver is a back-end of GDB, handling the inferior and the OS, but not user commands.

**Record Details**

**udb**

Proprietary tool that seems to work similarly.

Disclaimer: These are what I understood based on colleagues explanations, no first hand checking of code was done on either.
How do we use that?

(gdb) complete reverse-
  reverse-continue
  reverse-finish
  reverse-next
  reverse-nexti
  reverse-search
  reverse-step
  reverse-steipi
GDB’s front end

2 options:
1. Explicitly using a reverse command
   `reverse-next`

2. Changing the execution direction:
   `set execution-direction reverse next`
GDB’s command handling

- If the command started with reverse-, set the execution to reverse

- GDB attempts to reuse as much code as possible for similarly named commands

- Whenever we know something works differently, we explicitly handle it with an if statement.
  - If the command started with reverse-, set the execution back to forward
RR

Uses a smart approach:
Offload as much as possible to GDB

RR, in replay mode, is a gdbserver with a reverse executing target

Meaning GDB handles the logic of understanding commands, reading debug information, etc

All RR has to do is “just” make the hardware behave correctly backwards

Gdbserver is a back-end of GDB, handling the inferior and the OS, but not user commands.

Front-end Details
What could possibly go wrong?

```
(gdb) reverse-until
Undefined command: "reverse-until". Try "help".
(gdb) frame
#0  main () at t.c:24
24       setup (n);
(gdb) set exec-direction reverse
(gdb) step
main () at t.c:23
23       int p = 0;
```
So, so many things go wrong

User experience improvements

Quick access to the bugzilla search
Commands

- Until
  - Works (badly) if setting direction manually
  - No reverse-version

- Record instruction-history
- Record function-call-history
  - only available for btrace

Issues

[QR Code: Bug reporting it]

[QR Code: Stackoverflow question]
UX

This is a real debug session

That is not what I would expect the “step” command to do

And if we decide to continue forward, the warning makes it sound like we can’t go forward anymore

Issues
Harder issues

In case you want a big challenge

Record full needs a lot of help

- Multiple inferiors
  - The history is saved as a global variable.
  - There is no way to know to whom the history belongs

- Multithreading
  - Similarly, there is no way to know which thread owns a recorded instruction

- Unusably slow
  - Needs profiling, then improving on the hotspots
Where do I come in?

Approved-By: Guinevere Larsen <blarsen@redhat.com> (record-full)
I want to help out!

Things I like:
● Reverse debugging
● Getting people into open source
● Talking about stuff I like

Reach out if you anything piqued your interest!
More questions?
Thank you!

E-mail: blarsen@redhat.com
IRC: guinevere in libera-chat, #gdb
Linkedin: Guinevere Larsen