Aprendendo a depurar programas com o GDB

Edjunior Machado Sérgio Durigan Júnior

26 de novembro de 2012

GDB
The GNU Project
Debugger

Licença

- Licença: **Attribution-ShareAlike 3.0 Unported (CC BY-SA 3.0)**
- `http://creativecommons.org/licenses/by-sa/3.0/`

Antes de mais nada...

- O `TAB` é seu amigo
- `help` e `apropos` também
- A tecla `enter` repete o comando anterior

Compilando o Programa com Informações para Depuração

- `gcc -g3 -O0 fonte.c -o programa`
- `CFLAGS="-g3 -O0" ./configure; make`
  - O segundo parâmetro da variável `CFLAGS` é **menos ó zero**

Executando o GDB

- `gdb ./programa`
- `gdb --args ./programa <arg1> <arg2>`
- `gdb`
  `(gdb) file programa`
  `(gdb) run <arg2> <arg2> ...`
  ou
  `(gdb) start <arg2> <arg2> ...`

Examinando Dados

- `print <variável>`
  - `p *array@<elementos>`
- `x <endereço>`
- `whatis <variável>`
- `display <variável>` e `info display`
- Configure a exibição de `structs` através de `set print pretty on`

TUI (Text User Interface)

- Embora não muita difundida, é uma interface bastante útil
- Pressione `CTRL+x a` (ou `CTRL+x 1` ou `CTRL+x 2`)
- Pode ter alguns problemas caso o programa sendo depurado utilize a saida padrão ou a janela do terminal seja redimensionada
- *Workaround*: sair e entrar novamente na interface

Parando e Continuando a Execução

- Breakpoints
  - `break`
    - `break 7` (parar na linha 7)
    - `break foo` (parar na função `foo`)
    - `break fonte.c:bar` (parar na função `bar`, no arquivo `fonte.c`)
    - `break fonte.c:7` (parar na linha 7 do arquivo `fonte.c`)
  - `tbreak` (breakpoint temporário); mesmos argumentos que `break`
- Watchpoints
  - `watch` (escrita), `rwatch` (leitura) e `awatch` (acesso)
  - Watchpoints condicionais
- Catchpoints
  - `catch syscall`
  - `catch fork`, com `follow-fork-mode`

Parando e Continuando a Execução[2]

- `continue`
- `next`, `nexti`
- `step`, `stepi`
- `finish`
- Loop infinito? `CTRL+c`

Examinando o Código

- `list`
- `disassemble`

Examinando Dados[2]

- `bt`
- `frame`
- `up`, `down`

Alterando o Programa Depurado

- Alterando os Dados
  - `set var <variável> = <valor>`
  - `set {int}<endereço> = <valor>`
- Alterando o Fluxo
  - `jump`
  - `return`

Corefiles

- Representação do estado do um programa em determinado momento
- Pode ser gerado manualmente (GDB) ou automaticamente em caso de falha do programa
- Habilite a criação de `corefiles` com
  `ulimit -c unlimited`
- Executando o GDB
  `gdb programa core.PID`
- Manualmente, de dentro do GDB
  `generate-core-file`
  `core`

Outras Informações

- `info`
  - `info breakpoints`
  - `info watchpoints`
  - `info locals`
  - `info registers`
  - `...`

Depuração Reversa(!!)

- Suportada desde 2008 (mas pouca gente sabe)
- ```
  (gdb) start
  (gdb) target record
  (gdb) next
  (gdb) ...
  (gdb) reverse-next
  ```
- Altere a direção de execução com
  `set exec-direction [forward,reverse]`
  ```

Em Dúvida Sobre Algum Comando?

- `help`
- `apropos`

Outras Funcionalidades

- `Tracepoints`
- `forks e threads`
- `gdbserver`

O papel da Red Hat nessa história

- GDB, GCC, glibc, Kernel Linux, virtualização...
- Participa ativamente do projeto (2 mantenedores globais no GDB).
  - Suporte a extensões em Python
  - SystemTap probes em userspace
  - Melhorias no desempenho (utilizando probes)
  - Várias melhorias no suporte a C++ (Projeto Archer)
  - ...

Referências

- **GDB: The GNU Project Debugger**
  http://sourceware.org/gdb/
- **Debugging with GDB**
  http://sourceware.org/gdb/current/onlinedocs/gdb/
- **GDB Wiki**
  http://sourceware.org/gdb/wiki/
- canal #gdb na Freenode

Obrigado!

Dúvidas?

Edjunior Machado

Sérgio Durigan Júnior
sergiodj@redhat.com