GDB guide

jorge.gonzalez, martin.carrasco

April 16, 2020

What is GDB?

- It's the GNU Project Debugger

What is GDB?

- It's the GNU Project Debugger
- Allows you to analyze a running program

What is GDB?

- It's the GNU Project Debugger
- Allows you to analyze a running program
- Print variable names, change variable values, stop at certain conditions, etc.

How to use GDB

- Start GDB session for a program: `gdb <program>`

How to use GDB

- Start GDB session for a program: `gdb <program>`
- Start program execution `run`

How to use GDB

- Start GDB session for a program: `gdb <program>`
- Start program execution `run`
- Setup a breakpoint where execution will stop
  `break <file>:<line>` `disable <num>` `info <breakpoint>`

How to use GDB

- Start GDB session for a program: `gdb <program>`
- Start program execution `run`
- Setup a breakpoint where execution will stop
  `break <file>:<line>` `disable <num>` `info <breakpoint>`
- Execute current line and break at next `next`

How to use GDB

- Start GDB session for a program: `gdb <program>`
- Start program execution `run`
- Setup a breakpoint where execution will stop
  `break <file>:<line>` `disable <num>` `info <breakpoint>`
- Execute current line and break at next `next`
- Run a trace of the calls that led to current point `backtrace`

How to use GDB

- Start GDB session for a program: `gdb <program>`
- Start program execution `run`
- Setup a breakpoint where execution will stop
  `break <file>:<line>` `disable <num>` `info <breakpoint>`
- Execute current line and break at next `next`
- Run a trace of the calls that led to current point `backtrace`
- Print a variable content `print <var>` `set <expression>`

Configurations for GDB

- Install GDB
- Define GDBMacros variable in the local enviroment
- Run docker container (if you are using one) with

```
docker run --cap-add=SYS_PTRACE --security-opt
seccomp=unconfined
```

GDB in PintOS

- **Shell1**: Start PintOS `pintos --gdb -- run mytest`

GDB in PintOS

- **Shell1**: Start PintOS `pintos --gdb -- run mytest`
- **Shell2**: Load binary `pintos-gdb kernel.o`

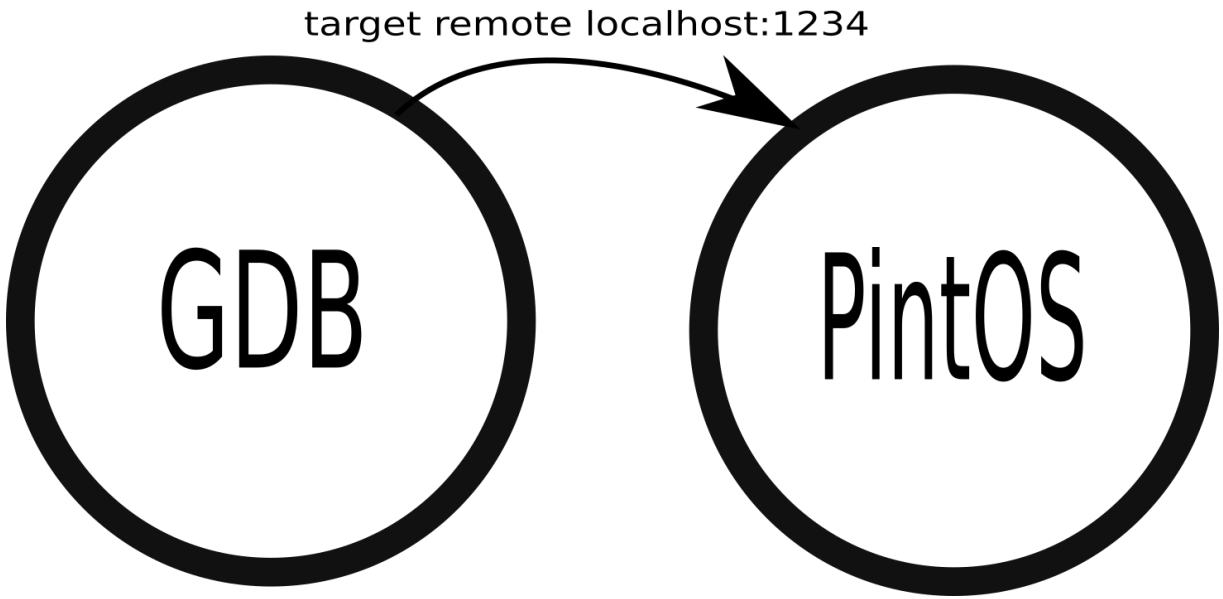
GDB in PintOS

- **Shell1**: Start PintOS `pintos --gdb -- run mytest`
- **Shell2**: Load binary `pintos-gdb kernel.o`
- **Shell2**: Attach to proccess `target remote localhost:1234`

GDB in PintOS

- **Shell1**: Start PintOS `pintos --gdb -- run mytest`
- **Shell2**: Load binary `pintos-gdb kernel.o`
- **Shell2**: Attach to proccess `target remote localhost:1234`
- Start debbuging by pressing `c`

Configuration for GDB: Graphical



Usefull PintOS-GDB macros

- **btthread thread**: Run a backtrace on a specific thread
- **dumplist list-name list-type element**: Print the element of a given list
- **hook-stop**: This is called everytime a pagefault occurs
- **btthreadlist list-name element**: Shows the backtrace of all the threads contained in the list
- **btpagefault**:Backtrace of a thread after a page fault.

Usage example: Alarm Clock

Live example

Bibliography

https://web.stanford.edu/class/cs140/projects/pintos/pintos.pdf