mGDB (Multi-GDB)

mGDB is a user-friendly tool that manages the tedium of debugging multiple processes and its threads at the same time.

Table of Contents:

1. Usage

Please ensure that you are running all bash scripts as **root**.

1.1 Starting a Session

mGDB manages your gdb processes within a particular session. When you spawn multiple gdb processes to hook onto your target processes, it will be managed under a sessions categorized by the date and time that you run it.

To start a session, you can run `start_sessions.sh`

1.1.1 Help (-h | --help)

```
Usage: ./start_sessions.sh [ -h ] ( -n | -l ) -s [ -w ]
  -h | --help                 Show this help message and exit
  -n | --process-name         The name of the process(es)
  -l | --pid-list             A list of pids
  -s | --script               Path to gdb script
  -w | --wait                     To wait on a process to spawn.
                                  This only works with (-n | --process-name)
```

1.1.2 Process Name (-n | --process-name)

In this example, I pre-ran 3 binaries called `test` and the gdb script I am using is `gdb_scripts/debug_example_script.gdb`

```
root@gerald-ubuntu:/home/gerald/repositories/mGDB# ./start_sessions.sh -n test -s gdb_scripts/debug_exa

[INFO] Debugging Session Mappings (16-10-2023T12-47-21)
-------------------- | -------------------- | --------------------
GDB PID              | PROCESS PID          | PROCESS NAME
-------------------- | -------------------- | --------------------
9494                 | 9473                 | test
9496                 | 9443                 | test
9524                 | 9105                 | test

All gdb output is redirected to:
 - /home/gerald/repositories/mGDB/logs/16-10-2023T12-47-21/gdb_output/9473_test.log
 - /home/gerald/repositories/mGDB/logs/16-10-2023T12-47-21/gdb_output/9443_test.log
 - /home/gerald/repositories/mGDB/logs/16-10-2023T12-47-21/gdb_output/9105_test.log

[INFO] Added mappings: /home/gerald/repositories/mGDB/logs/16-10-2023T12-47-21/mappings.txt
[INFO] Added database: /home/gerald/repositories/mGDB/database/16-10-2023T12-47-21/sessions.txt
[INFO] Added gdb script contents for reference: /home/gerald/repositories/mGDB/logs/16-10-2023T12-47-2
```

1.1.3 A list of PIDs (-l | --pid-list)

Reusing the example in 1.1.2, instead of stating a process name, you can specify a list of PIDs to hook on.

```
root@gerald-ubuntu:/home/gerald/repositories/mGDB# ./start_sessions.sh -l 9473 9105 9443 -s gdb_scripts

[INFO] Debugging Session Mappings (16-10-2023T13-07-32)
-------------------- | -------------------- | --------------------
GDB PID              | PROCESS PID          | PROCESS NAME
-------------------- | -------------------- | --------------------
10254                | 9473                 | test
10256                | 9105                 | test
10283                | 9443                 | test

All gdb output is redirected to:
 - /home/gerald/repositories/mGDB/logs/16-10-2023T13-07-32/gdb_output/9473_test.log
 - /home/gerald/repositories/mGDB/logs/16-10-2023T13-07-32/gdb_output/9105_test.log
 - /home/gerald/repositories/mGDB/logs/16-10-2023T13-07-32/gdb_output/9443_test.log

[INFO] Added mappings: /home/gerald/repositories/mGDB/logs/16-10-2023T13-07-32/mappings.txt
[INFO] Added database: /home/gerald/repositories/mGDB/database/16-10-2023T13-07-32/sessions.txt
```

```
[INFO] Added gdb script contents for reference: /home/gerald/repositories/mGDB/logs/16-10-2023T13-07-3
[INFO] Type the following command to follow all logs: tail -f /home/gerald/repositories/mGDB/logs/16-1
```

1.2 Showing Sessions

Running `show_sessions.sh` will show all current sessions managed by **mGDB** currently.

```
root@gerald-ubuntu:/home/gerald/repositories/mGDB# ./show_sessions.sh

[INFO] Current Sessions:
Session              | Arguments
-------------------- | --------------------------------------------------
16-10-2023T13-13-57  | -l 9473 9443 -s gdb_scripts/debug_example_script.gdb
16-10-2023T13-17-27  | -l 9105 -s gdb_scripts/debug_example_script.gdb
```

1.3 Killing Sessions

Run `kill_sessions.sh` to end sessions managed by **mGDB** currently.

1.3.1 Help (-h | --help)

```
root@gerald-ubuntu:/home/gerald/repositories/mGDB# ./kill_sessions.sh -h
Usage: ./kill_sessions.sh [ -h ] ( -t | -a | -g )
  -h | --help                  Show this help message and exit
  -t | --session-datetime      The gdb session identified by datetime
  -a | --all                   Detaches all gdb processes from all sessions.
  -g | --global                Detaches all gdb processes, including the ones not known in database.
```

1.3.2 Killing a Particular Session (-t | --session-datetime)

You can refer to the outputs of `./show_sessions.sh` to kill off a particular session.

```
root@gerald-ubuntu:/home/gerald/repositories/mGDB# ./kill_sessions.sh -t 16-10-2023T13-13-57
[INFO] Detached gdb (10445) from test (9473)
[INFO] Detached gdb (10447) from test (9443)
[INFO] Removed session 16-10-2023T13-13-57 in database.
```

1.3.3 Killing all sessions managed by mGDB (-a | --all)

This feature just provides an easy way for lazy people like myself to kill off all gdb processes in all sessions.

```
root@gerald-ubuntu:/home/gerald/repositories/mGDB# ./kill_sessions.sh -a
[INFO] Detaching all gdb processes in session 16-10-2023T13-17-27...
[INFO] Detached gdb (10579) from test (9105)
[INFO] Removed session 16-10-2023T13-17-27 in database.
--------------------------------------------------------------
```

```
[INFO] Detaching all gdb processes in session 16-10-2023T13-30-54...
[INFO] Detached gdb (10835) from test (9473)
[INFO] Detached gdb (10838) from test (9443)
[INFO] Removed session 16-10-2023T13-30-54 in database.
--------------------------------------------------------------
```

1.3.4 Killing All GDB Processes (-g | --global)

WARNING: Please use this command only when you are very sure that you are the only one doing the debugging on the system **or** if there are unwanted GDB processes hooking onto your target processes.

The `-g | --global` option just provides an easy way to hard reset the enviroment such that there won't be any gdb processes running anymore.

```
root@gerald-ubuntu:/home/gerald/repositories/mGDB# ./kill_sessions.sh -g
[INFO] Detaching all gdb processes in session 16-10-2023T13-34-34...
[INFO] Detached gdb (10980) from test (9473)
[INFO] Detached gdb (10982) from test (9443)
[INFO] Removed session 16-10-2023T13-34-34 in database.
--------------------------------------------------------------
[INFO] Detaching all gdb processes in session 16-10-2023T13-34-37...
[INFO] Detached gdb (11048) from test (9105)
[INFO] Removed session 16-10-2023T13-34-37 in database.
--------------------------------------------------------------
[ERROR] There are no gdb processes outside of known database.
```

2. Extensions

As GDB's default commands are not the nicest and easiest to look at, there is a need to extend its capabilities with the help of python scripting.

An example would be `info proc mappings` which shows you the different start addresses and end addresses of a particular binary.

```
(gdb) info proc mappings
process 9473
Mapped address spaces:

          Start Addr           End Addr       Size     Offset  Perms  objfile
      0x5589a58d7000     0x5589a58d8000     0x1000        0x0  r--p   /home/gerald/repositories/mGDB/te
      0x5589a58d8000     0x5589a58d9000     0x1000     0x1000  r-xp   /home/gerald/repositories/mGDB/te
      0x5589a58d9000     0x5589a58da000     0x1000     0x2000  r--p   /home/gerald/repositories/mGDB/te
      0x5589a58da000     0x5589a58db000     0x1000     0x2000  r--p   /home/gerald/repositories/mGDB/te
      0x5589a58db000     0x5589a58dc000     0x1000     0x3000  rw-p   /home/gerald/repositories/mGDB/te
      0x5589a5aa9000     0x5589a5aca000    0x21000        0x0  rw-p   [heap]
      0x7f6a81c00000     0x7f6a81c28000    0x28000        0x0  r--p   /usr/lib/x86_64-linux-gnu/libc.s
      0x7f6a81c28000     0x7f6a81dbd000   0x195000    0x28000  r-xp   /usr/lib/x86_64-linux-gnu/libc.s
      0x7f6a81dbd000     0x7f6a81e15000    0x58000   0x1bd000  r--p   /usr/lib/x86_64-linux-gnu/libc.s
      0x7f6a81e15000     0x7f6a81e19000     0x4000   0x214000  r--p   /usr/lib/x86_64-linux-gnu/libc.s
      0x7f6a81e19000     0x7f6a81e1b000     0x2000   0x218000  rw-p   /usr/lib/x86_64-linux-gnu/libc.s
```

If I wanted the start address of `/usr/lib/x86_64-linux-gnu/libc.so.6` , I would have to manually look through the output and probably assign `0x7f6a81c00000` to a variable. However, we want to be able to do this programmatically.

Under `gdb_scripts/py_commands` , there are several useful commands such as:

- get_base_addr
- load_library_symbol
- strlen

To import in these commands, you can put the following line in your gdb script:

```
source gdb_scripts/py_commands/<command_name>.py
```

Example of getting base address of a binary and printing its address using the return variable:

```
(gdb) source gdb_scripts/py_commands/get_base_addr.py
[INFO] Example usage of get_base_addr:
Template: get_base_addr
Eg. get_base_addr /usr/lib/example_lib.so
    - command name: get_base_addr
    - number of arguments: 1
    - ret variable in gdb: $get_base_addr_ret
      -> returns 0x0 on failure, otherwise a valid address

(gdb) get_base_addr /home/gerald/repositories/mGDB/test_binaries/test
[INFO] Found base address of /home/gerald/repositories/mGDB/test_binaries/test: 0x5589a58d7000

(gdb) printf "%p\n", $get_base_addr_ret
0x5589a58d7000
```

3. Contributing

I am currently accepting pull requests for `gdb_scripts/py_commands` if you wish to implement an extension that is useful for the community. Please follow the `gdb_scripts/py_commands/template.py` as a guideline on writing extensions.