EECS 192: Mechatronics Design Lab

Discussion 8: GDB Debugging

GSI: Andrew Barkan

10 Mar 2019 (Week 8)

- GDB and Core Dump
- GDB Demo
GNU Debugger
GDB and Core Dump

Debugging

- General idea: locate bugs in your program by stopping it at particular points and looking at values
- GDB (gdb) "GNU Debugger" for C, C++, and other languages:
  - GDB should already be installed w/ ESP-IDF)
  - Run GDB script with core dump
  - Examine program with GDB
- (Documentation link)

What’s wrong with my code?
Setting Up GDB

- You may have to set your paths to run from terminal
- Alternatively, you can run the ESP-IDF export script after opening a new terminal
  - Navigate to `.platformio/packages/framework-esp32df`
  - Run the export script (On Windows => `export.bat`)
- The export script may have you run `install.bat` first
- You should then be able to run the `gdb core dump` script
Preparing Your ESP32

- We need to enable core dump
- Compiler config variables
  - `./platformio/penv/Scripts/pio.exe run -t menuconfig`
  - Make sure you run from your project folder!
- Take a look at the SkeletonHuzzah32 wiki for more information

Setting compiler config variables
Preparing Your ESP32

- Here are the specific variables you need to set:
  - `CONFIG_ESP_COREDUMP_TO_FLASH_OR_UART` (change to UART)
  - `CONFIG_ESP_SYSTEM_PANIC_PRINT_REBOOT` or `ESP_SYSTEM_PANIC_PRINT_HALT` (change to print registers and halt)
  - `CONFIG_ESP_COREDUMP_DATA_FORMAT` (set to ELF)
  - Component config `-q` Core dump `-q` ELF format ELF format (Executable and Linkable Format file for core dump)
  - Compile with debugger (default option)

- After setting variables, save (s) and quit (q)

- Restart VS Code and allow CMakeLists to rebuild
Generating A Core Dump

- Probably already seen a core dump from serial output
- Big wall of seemingly meaningless characters (actually base64)
- What if I want to examine a specific point?
  - `*((int *) 0) = 0;`
  - Will cause a panic and core dump!

Some core dump text
When your program generates a core dump, the result is printed to over UART to whatever serial interface you are using.

Copy and save the core dump to a .txt file.

Remove:

```
================= CORE DUMP START =================
```

and

```
================= CORE DUMP END =================
```

from text file.
Now that we have the core dump saved, we can use the ESP-IDF core dump script to begin debugging

- Located in
  ./platformio/packages/framework-esp-idf/components/espcoredump

- Here is the command template:

```bash
python espcoredump.py dbg_corefile -t b64 -c
</path/to/saved/base64/text> </path/to/program/elf/file>
```
GDB Commands

- Keep in mind that you are NOT running the program; you’re just looking at a snapshot taken at the instant of core dump

- Useful commands for debugging core dump with GDB:
  - `help`: Brings up help interface w/ more commands
  - `help <command>`: Gives helpful info on given command
  - `list`: Lists 10 lines around the error line
  - `list <line #>`: Lists 10 lines around the given line
  - `bt`: Backtrace function calls from error location
  - `info <subject>`: Print out info on given subject (e.g. locals)
More GDB Commands

- You can also examine specific frames and variables!
- Some more commands:
  - frame <frame #> : Brings you to specified frame #
  - print <expression> : Print the value of an expression (e.g. a variable)
  - thread <thread #> : Brings you to specified thread
Some Debugging Tips

- When working on new functionality, take small controlled steps
  - Start with functional code (make sure it is committed)
  - Make small change so that ensuing errors can be isolated
- Math is hard
  - Implicit casting and integer math can cause issues
  - If debugging math, inspect intermediate values
  - e.g. \( a, b, c, (a + b), c \times (a + b) \)
- Print statements are still useful tools!
- Preemptive error handling can be powerful (even when prototyping)
GDB Example