VE482 — Introduction to Operating Systems

Lab 7

Manuel — UM-JI (Fall 2019)

Goals of the lab

- Find and read documentation
- Understand how to use GDB
- Run GDB on a basic exmaple

1 Generalities on GDB

1. How to enable built-in debugging in `gcc`?
2. What is the meaning of GDB?
3. Compile the C program from homework 4 with debugging enabled.

2 Basic use of GDB

1. Find the homepage of the GDB project.
2. What languages are supported by GDB?

In a terminal navigate to the folder where the previously mentioned program is located. Enter `gdb` in the terminal. An interactive shell should open, under the condition that GDB has been installed. First note that this shell recalls history when using the arrow keys and auto-completes command using the `TAB` key.

At any stage `help [command]` can be typed to get general information or details on a specific command. To debug the program cthread input `file cthread`, the program is loaded but not run. In order to run it input the command `run`.

The command `break cthread.c:17` sets a breakpoint at line 17 in the file cthread.c. Upon executing `run`, the program pauses each time it encounters a breakpoint. There is no restriction on the number of breakpoints added to a program.

An alternative strategy consists in breaking at a certain function. In that case input for instance `break thread_count`, in order to stop each time the function `karatsuba` is called.

When blocked at a breakpoint computation can be resumed using the command `continue` but will stop at the next breakpoint. In order to progress line-by-line use the command `step`. If more than one instruction is written on a line input `next` to only run a single instruction at a time. Since typing in `next` or `step` many time can be tedious GDB offers the possibility the repeat the previous command by just pushing the `ENTER` key.

In order for debugging to be effective it is necessary to observe things at breakpoints: use `print tmp` to display the value of the variable `tmp`. Special breakpoints, called watchpoints, can be used to track a variable. This is achieved through the `watch tmp` command. Each time `tmp` is modified the program stops and display both its old and new values.

For a pointer `p`, `print p` prints the memory address of the pointer. Accessing each element of a structure is done in a similar way as in C, e.g. `p->s` displays the field `s` referenced by pointer `p`. Note that it is also possible to input `print *p` even if `p` is a pointer on a structure (this not easily done in C).

3. What are the following GDB commands doing:
    - `backtrace`
    - `where`
    - `finish`
    - `delete`
    - `info breakpoints`
4. Search the documentation and explain how to use conditional breakpoints.

Watch this youtube video and answer the following questions.

5. What is `-tui` option for GDB?
6. What is the "reverse step" in GDB and how to enable it. Provide the key steps and commands.