CS341: Computer Architecture Lab

Lab 0: Debugging

Report

Richeek Das (190260036)

Department of Computer Science and Engineering
Indian Institute of Technology Bombay
2021-2022

Contents

Abstract

This is the introductory lab of CS341 (Computer Architecture Lab) to get people accustomed to using essential tools like GDB and Valgrind and also introduce some key concepts of objdump and linker-loader.

1. Problems

1.1 A0-GDB

Files submitted:

- `heap.cpp`
- `heap.hpp`
- `dijkstra.cpp`
- `prettyprinters.py`
- `gdb.txt`

Objectives of this problem were to introduce the utility of pretty printers in GDB and to get comfortable with using GDB to walkthrough the code step-by-step and find out whats going wrong.

Firstly we check out the python API for gdb and build a pretty printer for a min-heap. The code can be found in: `prettyprinters.py`.

Once we are done with getting the contents of `VertexHeap` in a readable format, we proceed to dig out the bugs.

Finally we do a walkthrough step by step, using `step`, `next` and `break` statements and finally find the two bugs in `heap.hpp` and `heap.cpp`. We log the commands used to find these bugs in `gdb.txt`.

1.2 A0-Valgrind

Files submitted:

- `argparse.c`
- `answers.pdf`

For every `malloc` or `strdup` there should be a `free`.

We go by the above rule :) With Valgrind we know the source of the allocation. As in, we know when the memory was allocated. But we don't know, where we should `free` it! Thats left to our discretion.

We know `getArg()` is the last place where we'll be using the `argParser` var. So, naturally its a good place to `free` all `malloc'd` and `strdup'd` memory.

Issues:

- `argParser.argList = malloc(sizeof(struct arg) * argParser.capacity);` - Allocation done, but never free'd.
- `struct arg *tempArgs = malloc(sizeof(struct arg) * argParser.capacity);` - Allocation done, but never free'd.
- `argParser.argList[argParser.len].name = strdup(name);` - Allocation done but never free'd.
- `argParser.argList[j].result = strdup(argv[i + 1]);` - Allocation done but never free'd. To get an error because of this allocation, we'll need to run `valgrind ./main -J sth`

Solutions:

- Initialise `argParser.argList[i].result` to `NULL` in `addArg()` function (whenever `argParser.argList[i].name` is initialised). This resolves the later warnings of uninitialised variables.
-
  ```
  for (int i = 0; i < argParser.len; i++)
  {
      if (argParser.argList[i].name)
          free(argParser.argList[i].name);
      if (argParser.argList[i].result)
          free(argParser.argList[i].result);
  }
  free(argParser.argList);
  ```

  Later in the `getArg()` function we use this piece of code to free all the allocations we had made with the struct `argParser`.
- Remember we are returning `strdup(argParser.argList[i].result);` if we find a value for the given argument. Since we are invoking `strdup` here, its necessary to `free` the `char* result` in `main.c` as well.