Week 6 Lab

Debugging with gdb

Task 1. Debugging with gdb

Go to the Moodle page for FIT3042 and download the file `buggy.c`. This program is supposed to display a list of numbers and their squares.

a) You will notice that `buggy.c` does not have a `Makefile`. Write one that will compile `buggy.c` to an executable file called `squares`. Add another target called `squares-debug` that will compile with the `-g` flag. It doesn't matter whether you give the debuggable version the same name or a different name – do whatever you find more convenient. You will probably need to compile to a debug version a lot to get this exercise completed, so it's worth making that operation as simple as possible.

b) Use `ls -l` to look at the sizes of the files compiled with and without the `-g` flag. How much difference does it make?

c) If you compile and run `squares`, you will see that it does not work correctly. Create a debug version and use `gdb` to find and fix the bugs in it.

Task 2. Catch up on lab work

Many students are not up to date with their lab work. If you are in this situation, please make use of any time you have left in this week's lab to catch up on exercises from previous weeks. If there were any programs that you could not finish because they had bugs you couldn't find, use your newfound `gdb` knowledge to find and fix these bugs. As ever, your demonstrator is there to help you if you get stuck.

Before you begin on Assignment 2, you should make sure you are confident with, *at least*, dynamic memory allocation, bitwise operations, the use of multiple source files, and the tools `gdb`, `valgrind`, and `make`. You should also try to fix any memory leaks or errors in your Assignment 1 code.