CSCB09 - Lab 3: Basic C

Introduction

In this lab you will get some practice with simple C programs.

Remember, the TAs are there to help you. You should work in pairs and are welcome to ask other students for help too.

Preparation

The C programming concepts you will be using in the lab today are:

- simple types
- arrays
- functions
- compiling and running a program from the command line
- using gdb (a debugger)

1. Writing some simple array functions

The file `arrays.c` contains some starter code for you to complete. You are asked to write three simple array functions. Copy the file from (`/courses/courses/cscb09w19/nizamnau/labs/lab3/arrays.c`) into your directory, compile and run the program as follows:

```
prompt> gcc -Wall -g -o arrays arrays.c
prompt> ./arrays
```

Now finish the four functions. Remember that when we pass arrays to functions we pass in the name of the array as a pointer, and we also have to pass in the size, because the array itself does not contain size information after we have passed it into the function. Within the function, you do not need to use pointers to access the array elements.

2. Array overflow and gdb

The file `overflow.c` contains a program that you can play with. It can be found here:

```
/courses/courses/cscb09w19/nizamnau/labs/lab3/overflow.c
```

You may not have seen the `#define` macro yet. It is the typical way to create constants in C. For this part of the lab you will be changing the values of `SIZE` and `OVERFLOW` to see what happens when `OVERFLOW` is bigger than `SIZE`

First read through the program. Describe to your partner or some unsuspecting bystander what the program is doing. Notice that we are printing the addresses of the variables. The purpose of doing that is to show how the variables are laid out in memory.

Next, compile and run the program as shown here:

```
prompt> gcc -Wall -g -o overflow overflow.c
prompt> ./overflow
```

Don't miss the new `-g` flag or gdb won't work properly below.

Notice the initial value for `i`. Did the program behave as you expected?

Now change the value of `OVERFLOW` to 5. Compile the program and run it again. What changed?

The next step is to run the program in gdb. gdb is a debugger. Type `gdb overflow`. Set a breakpoint in main by typing `break main`, and then start the program running by typing `run`. You want to watch the values of a few variables, so use `display a>` to show the value of variable a (for example). Do this for each variable you want to watch. Step through the program one line at a time using `next`. Notice when `k` changes, and notice the final value of `i`.

The last step in this section is to try to make your program crash. (for example, set overflow to something like 5000 to get it to crash with a Segmentation fault.)