

# Lab 0: C Warmup

Assigned	Monday, March 30th
Due Date	<i>Monday, April 6th, 2015 at 5:00pm</i>
Files	See "Acquiring the code" below. (Direct link here: <a href="#">arrays.c</a> )
Submissions	Submit your completed <a href="#">arrays.c</a> <a href="#">here</a> .

## Overview

The purpose of this assignment is to gain some very basic experience with C programming, including such topics as arrays, stack allocation, heap allocation, function calls, and memory safety (or the lack thereof). It is a credit/no credit exercise, and should take a short time to finish. Instructions on what to do are in the comments of the code you will download below.

## Setting up your working environment

To do the programming assignments in this course, you will either need 1) a virtual machine that you can run on your own laptop or desktop (if it supports 64-bit x86) or 2) access to the CSE instructional cluster (named "attu") or 3) the instructional Linux machines in the lab. **Only CSE majors** will be able to run on attu or on the instructional Linux machines in the lab as they require a CSENetID.

All of you should be able to use a virtual machine. To install a virtual machine player on your own computer, follow the instructions at this [link](#). **(NOTE that the provided image is missing at least one tool that you will need for lab 0. You will need to follow the instructions on the GoPost that tell you how to install the valgrind tool.)** You need to download the rather large VM file itself (it is 2.1 GB, so do this when you have good high bandwidth connectivity). This is a full linux environment with all the software we'll need this quarter pre-installed. You can find that file [here](#).

## Acquiring the code for this assignment

The first thing to do is to acquire the code for this lab. We provide several options for acquiring the code with and without CSE accounts. We encourage you to try multiple options in order to become more familiar with moving files between machines.

You can download [arrays.c](#) from a browser. Alternatively, use `wget` to download it from the command line. In a terminal, run:

```
wget http://www.cs.washington.edu/education/courses/cse351/15sp/labs/lab0/
```

This should download `arrays.c`, storing it in your current working directory. A lecture on some basic Unix commands is available [here](#) if you need help with this step.

## Editing, compiling, and running the code

Now that you have acquired the source file, open `arrays.c` in your favorite text editor. The `arrays.c` file contains a number of TODOs, which you are expected to complete. Most have a proceeding line that says "Answer:", which is where you should leave an answer to the question(s) posed in the TODO. One TODO requires you to write some code, which you should place immediately after the comment block that describes what to do.

The source file `arrays.c` won't do you any good by itself; you need a compiler (specifically the GNU C compiler) to compile it to an executable format. The GNU C compiler is available on the [CSE home VM](#), `attu`, the instructional Linux machines in the lab, and most popular variants of Linux, such as Ubuntu and Fedora. You are free to use whichever machine you like, although we will only provide support for the CSE home VM, `attu`, and the instructional Linux machines in the lab.

Using any one of these machines, open a terminal and execute `gcc -v`. On my machine, here is what I see:

```
$ gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-linux-gnu/4.6/lto-wrapper
Target: x86_64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu/Linaro
4.6.3-1ubuntu5' --with-bugurl=file:///usr/share/doc/gcc-4.6/README.Bu
gs --enable-languages=c,c++,fortran,objc,obj-c++ --prefix=/usr --prog
ram-suffix=-4.6 --enable-shared --enable-linker-build-id --with-syste
m-zlib --libexecdir=/usr/lib --without-included-gettext --enable-thre
ads=posix --with-gxx-include-dir=/usr/include/c++/4.6 --libdir=/usr/l
ib --enable-nls --with-sysroot=/ --enable-clocale=gnu --enable-libstd
cxx-debug --enable-libstdcxx-time=yes --enable-gnu-unique-object --en
able-plugin --enable-objc-gc --disable-werror --with-arch=32=i686 --w
ith-tune=generic --enable-checking=release --build=x86_64-linux-gnu -
-host=x86_64-linux-gnu --target=x86_64-linux-gnu
Thread model: posix
gcc version 4.6.3 (Ubuntu/Linaro 4.6.3-1ubuntu5)
```

The output tells me a bunch of the configuration options for the my installation of GCC as well as the version number, which is 4.6.3. Assuming that you have saved the `arrays.c` source file somewhere on your machine, navigate to that directory, and then use GCC to compile it with the following command:

```
gcc -g -Wall -std=gnu99 -o arrays arrays.c
```

It's not that important right now for you to know what all of these options do, but `-g` tells the compiler to include debugging symbols, `-Wall` says to print warnings for all types of potential problems, `-`

`std=gnu99` says to use the C99 standard (now only 14 years old!), `-o arrays` instructs the compiler to output the executable code to a file called `arrays`, and `arrays.c` is the source file being compiled.

Having executed that command, you should be able to see a file named `arrays` in the same directory:

```
$ ls
arrays  arrays.c
```

The `arrays` file is an executable file, which you can run using `./arrays`. On my machine, here is what I see:

```
$ ./arrays
Filling an array at address 0x7fff6b4e3e60 with 10 values
Done!
Filling an array at address 0x7fff6b4e3eac with 1 values
Done!
Filling an array at address 0x7fff6b4e3e90 with 4 values
Done!
Filling an array at address 0x11ca010 with 5 values
Done!
```

If you look through the source code of the `arrays.c` file, you should be able to match up each line that is printed with the output that you see in the console.

With that, you should have everything you need to complete the assignment. Note that every time you want to test a modification to the source file, you will need to use the `gcc -g -Wall -std=gnu99 -o arrays arrays.c` command to produce an updated `arrays` executable file.

## Checking your work

When you have finished the exercise, please take the time to ensure that your code both compiles without warnings (the GCC command will output lines that say "warning:" if you have warnings) and is readable. In particular, please try to make the format of your code match that of what was given to you.

## Submitting the exercise

Once you have completed the exercise, submit your `arrays.c` source file to the [Catalyst Dropbox for this lab](#).