Scouts BSA Programming Project Guide

Benjamin Davis [benjamin.j.davis96@gmail.com]

Contents

Chapter 1

Introduction

In addition to the historical and general knowledge, the Scouts BSA Programming Merit Badge requires several programming capability demonstrations. Requirement 5 outlines provides this requirement as follows:

> **5.A** With your counselor's approval, choose a sample program. Then, as a minimum, modify the code or add a function or subprogram to it. Debug and demonstrate the modified program to your counselor.
>
> **5.B** With your counselor's approval, choose a second programming language and development environment, different from those used for requirement 5a and in a different industry from 5a. Then write, debug, and demonstrate a functioning program to your counselor, using that language and environment.
>
> **5.C** With your counselor's approval, choose a third programming language and development environment, different from those used for requirements 5a and 5b and in a different industry from 5a or 5b. Then write, debug, and demonstrate a functioning program to your counselor, using that language and environment.
>
> **5.D** Explain how the programs you wrote for requirements 5a, 5b, and 5c process inputs, how they make decisions based on those inputs, and how they provide outputs based on the decision making

These requirements essentially boil down to 3 requirements in 3 different programming languages and the ability to explain them. While you are certainly able to choose any program of your desire (within the limits of your counselor's stipulations) this can be quite overwhelming and challenging for anyone without pre-existing knowledge of programming and a fair bit of programming experience. Thus I have found most scouts are more successful when they are provided with a pre-defined and recommended set of programs to use as a starting point. This guide is intended to provide all of the background information and instructions necessary for a scouts success in this portion of the merit badge.

1.1 Where to Start

The first thing we need to start programming is a computer. Modern computers come in a wide variety from industrial servers to simple laptops to gaming computers. The projects presented here are relatively simple and should be capable of operating on virtually any class of computer. That said there is a difference

between computers which will impact the development process: operating system. I cannot recommend Linux highly enough if you desire to do any substantial amount of programming; however it is still the minority in computer operating systems. Towards this end I have provided a Virtual Machine (VM) which allows you to run a Linux system on any other system with minimal overhead. I highly recommend using this approach for the projects portion of the merit badge and the instructions in the remainder of this document will be oriented towards this use case. Instructions for installing this VM and getting started can be found in chapter 2

However, if for some reason using a VM is not feasible I have verified each of the projects here to operate on Windows 10, Windows 11, Ubuntu, and Debian (Currently I do not have access to a Mac to verify these projects on that operating system, but I have no reason to believe Mac will not work with these projects). A deeper discussion of developing in these OS's can be found in Appendix A.

In order keep all of the files necessary for this merit badge in a single controlled space I have elected to use GitHub to host the files.

1.2 Projects

I highly encourage following the project requirements in order and using programs from the correct section for the respective requirement. While programs from `Project B` and `Project C` can be used interchangeably those found in `Project C` are notably more difficult, but also more rewarding. Below are the Projects which are recommended for each requirement. They have been placed in the order of difficulty that I expect them to pose to a brand new programmer. Additionally, they include the recommended program languages which could be used to most easily accomplish them (note some programming languages which we have discussed in the classroom portion are not found here, I am excluding them based on my experience that they tend to be overly complex without providing a novel advantage).

1.2.1 Project A Options

Web Calculator

Overview: Modify some code to complete a functioning calculator which operates out of a web-browser.

Programming Languages: JavaScript

1.2.2 Project B Options

Shape Area Calculator

Overview: Create a text-based program which can calculate the area of various shapes

Programming Languages: Python, Java, C++, C

Prime Number Finder

Overview: Create a text-based program which can calculate the first N primes

Programming Languages: Python, Java, C++, C, Fortran

Hang Man

Overview: Create a text-based HangMan game

Programming Languages: Python, Java, C++, C

1.2.3 Project C Options

If all of the below project options appear too difficult to accomplish in the given time or with your current school/extra-curricular activity schedule please talk to your counselor about using another option from the `Project B` list.

Sorting Numbers

Overview: Create a program which can sort a list of numbers using two different sorting algorithms.

Programming Languages: Python, C++, Java, C, Fortran

Self-Writing Program

Overview: Create a program can write its entire source code to a file

Programming Languages: C++, Java, C

Pong Game

Overview: Create a simple Pong-Style 2-D Game

Programming Languages: Python, BASIC, C++, Java

Snake Game

Overview: Create a simple Snake-Style 2-D Game

Programming Languages: Python, BASIC, C++, Java

Logic Puzzle

Overview: Create a simple program which provides the answer to a logic puzzle such as the Wolves and Sheep river crossing puzzle.

Programming Languages: Prolog, ALF, Python

FPGA Adder

Overview: Create device which can add two 8-bit numbers on an FPGA from switches and output the result to LEDs.

Programming Languages: SystemVerilog, Verilog, VHDL

Chapter 2

Getting Started

As discussed in section 1.1 I strongly recommend using the provided Virtual Machine (VM) for this merit badges. Doing so allows for the following notable advantages:

1. Only one installation necessary
    - All of the tools necessary for any project have been pre-installed on the VM
2. Guaranteed Compatibility
    - All of the projects for this merit badge have been verified to work on the VM
3. Easier to Follow
    - All of the instructions and examples in this project have been designed with the VM in mind
4. Improved Cyber Security
    - Using a VM keeps all of the programming and activity which is done inside the VM seperate from the main computer

The remainder of this chapter is spent allowing for install and a quick guide to the operating system.

2.1 Installing VM Software

There are a wide variety of VM software available on the market. I strongly recommend using Oracle's VirtualBox as it is free software which is created by a reliable and trusted industry member. That being said if you prefer the use of a different VM software that should be easily supported by the rest of this project guide.

2.1.1 Installing VirtualBox

To start we need to download VirtualBox. This can be done by navigating to the official webpage at https://www.virtualbox.org/wiki/Downloads (see Figure 2.1). Select the platform package which corresponds to the OS of your personal computer (macOS = OS X).

Figure 2.1: VirtualBox Download Page

2.2 Importing the Programming Merit Badge VM

2.3 Getting Familiar with the VM

When you first start the VM you will be greeted with a login screen. The username and password for the VM can be found below:

Username: scoutsbsa

Password: `goodturn`

Enter the password and you will be see the home screen. In the upper left corner of the screen you should see Figure 2.2

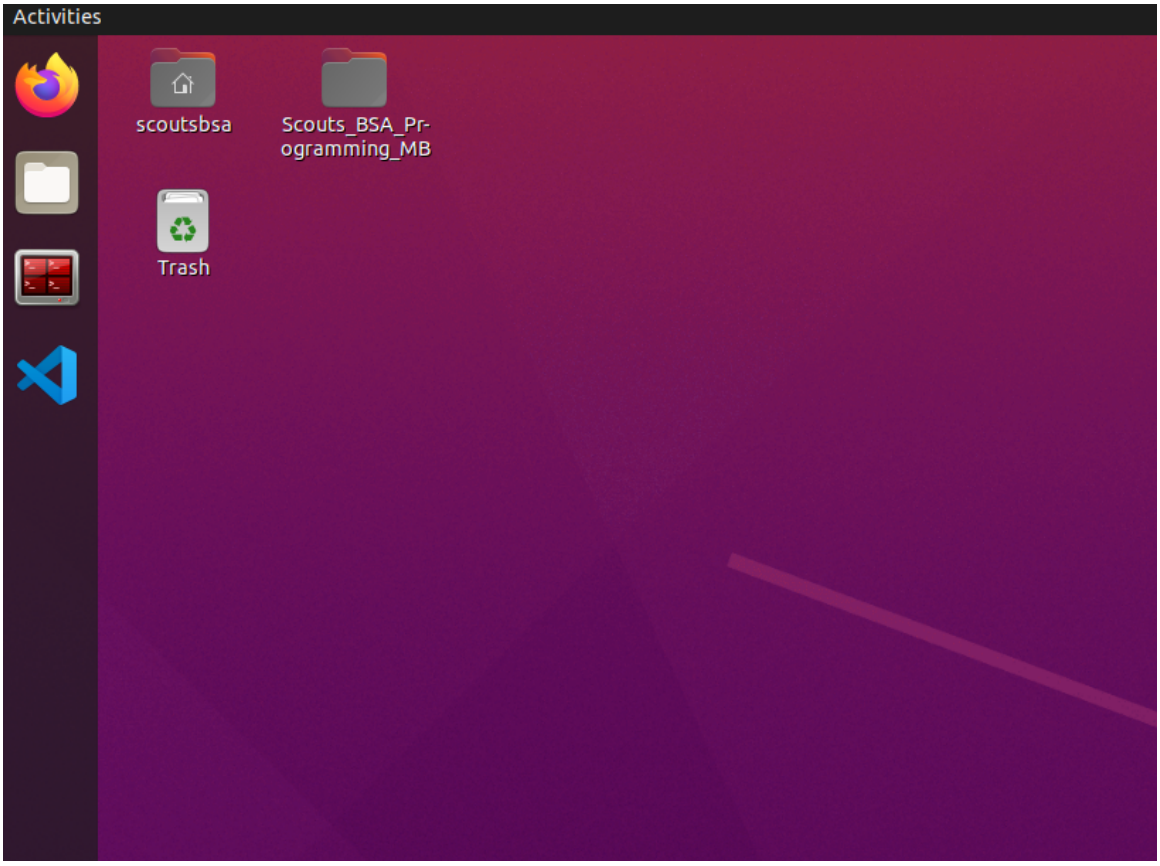
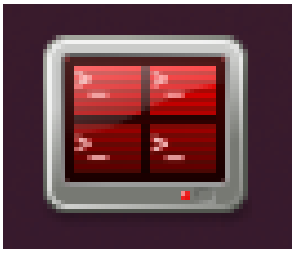


Figure 2.2: Desktop View

There are several icons on this screen. The bar on the left contains all of the programs necessary for the projects in this merit badge. The top image is the `FireFox` web-browser (like Google Chrome, Safari, or Internet Explorer). One icon down is the file explorer which allows you to see the files on the VM. Below the file explorer is the terminal which allows you to compile and execute programs (see Figure 2.3a). Finally, is the the VS Code icon (see Figure 2.3b). VS Code is a text editor which allows code to be highlighted so it is easier to read.



(a) Terminal Icon

(b) VS Code Icon

Figure 2.3: Program Icons

Now click on the terminal open it. This will place you in your *home* folder. Type the `ll` command to view a list of all the files and folders in your *home*.

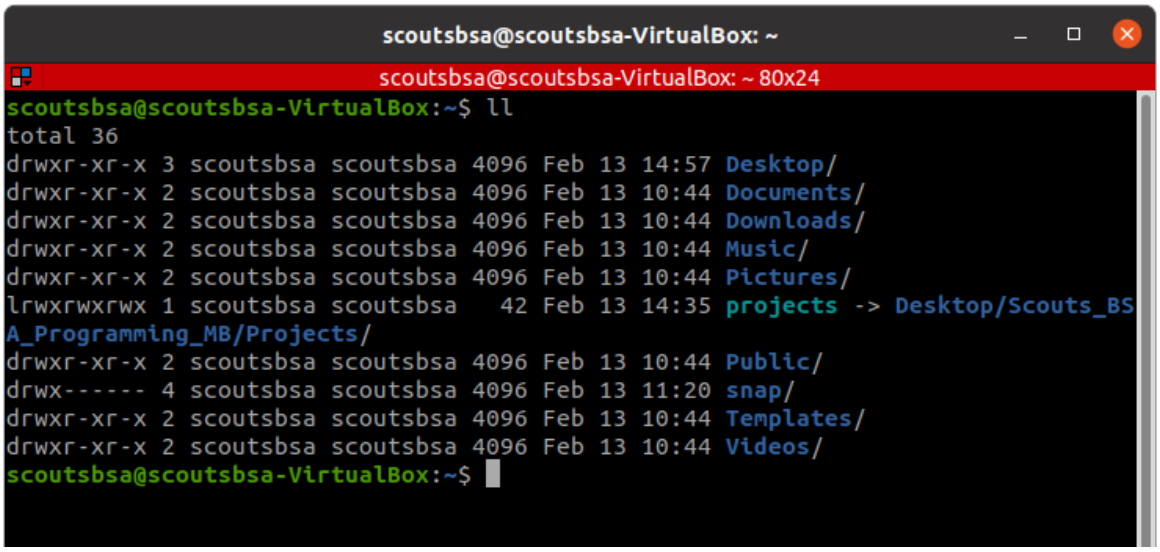


Figure 2.4: Home Directory File List

Notice there is a folder called projects. This is where all of the files you will need for your projects live. To get there in the terminal type `cd projects`. The `cd` command changes the directory you are in. Now find a list of the files in the projects directory.

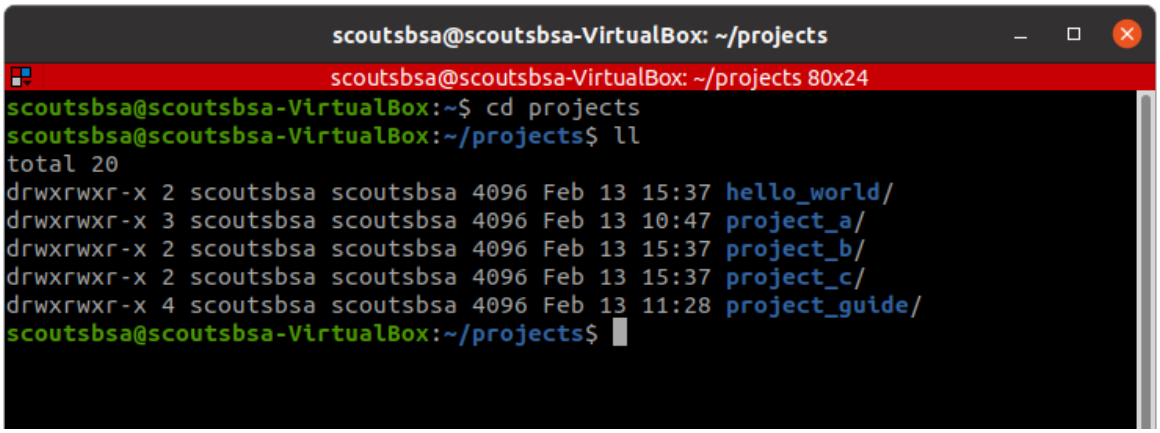


Figure 2.5: Project Directory File List

Now we want to go into the into the *hello_world* folder. Again use `cd` to do this.

Once in the *hello_world* folder view the two files using `ll`. Notice a file called *Makefile*. This is the file which controls how all of your projects are run. Use the command `make` to view what it does in this folder.

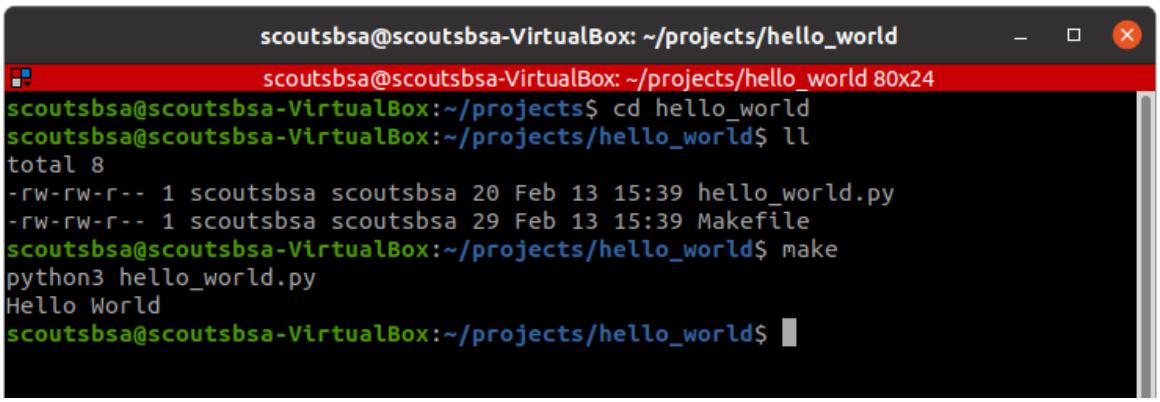


Figure 2.6: Project Directory File List

Now that you've seen how we are going to run our projects let's explore how we are going to change the code. Open the VS Code window by clicking on the VS Code icon on the left. The left hand side should contain a list of files and folders. At the top of this section the folder name which you are currently looking at should be displayed. VS Code should have automatically opened to the `projects` folder. If not use *File* → *Open Folder* to open the projects folder in your home directory.

Now click the hello world folder on the left-hand bar. You should see the same two files we saw in the terminal. Click `hello_world.py` to open the file. This file is extremely simple, so for now just change the "Hello World" message to another message of your choosing. Then rerun make in the terminal. You should now see your message displayed!

2.4 Next Steps

Now you should be ready to go! Head on over to the Project A chapter to get started.

If you've had any troubles in this section let your councilor know before moving on.

Chapter 3

Project A

Chapter 4

Project B

Chapter 5

Project C

Appendix A

Developing in Windows vs Linux vs Mac

A.1 WSL: The Best of Both Worlds?

Appendix B

Tool Installations

This appendix contains the instructions necessary to install the various tools on a variety of operating systems. Currently it is heavily lacking, but can be easily updated upon request if a scout is opting to use an operating system and language which are not currently listed.

B.1 JavaScript

B.2 Python

B.3 C++

B.4 C

B.5 BASIC

B.6 Fortran

B.7 Prolog

B.8 SystemVerilog, Verilog, VHDL