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).

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 can be installed on the VM with one command
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 separate from the main computer

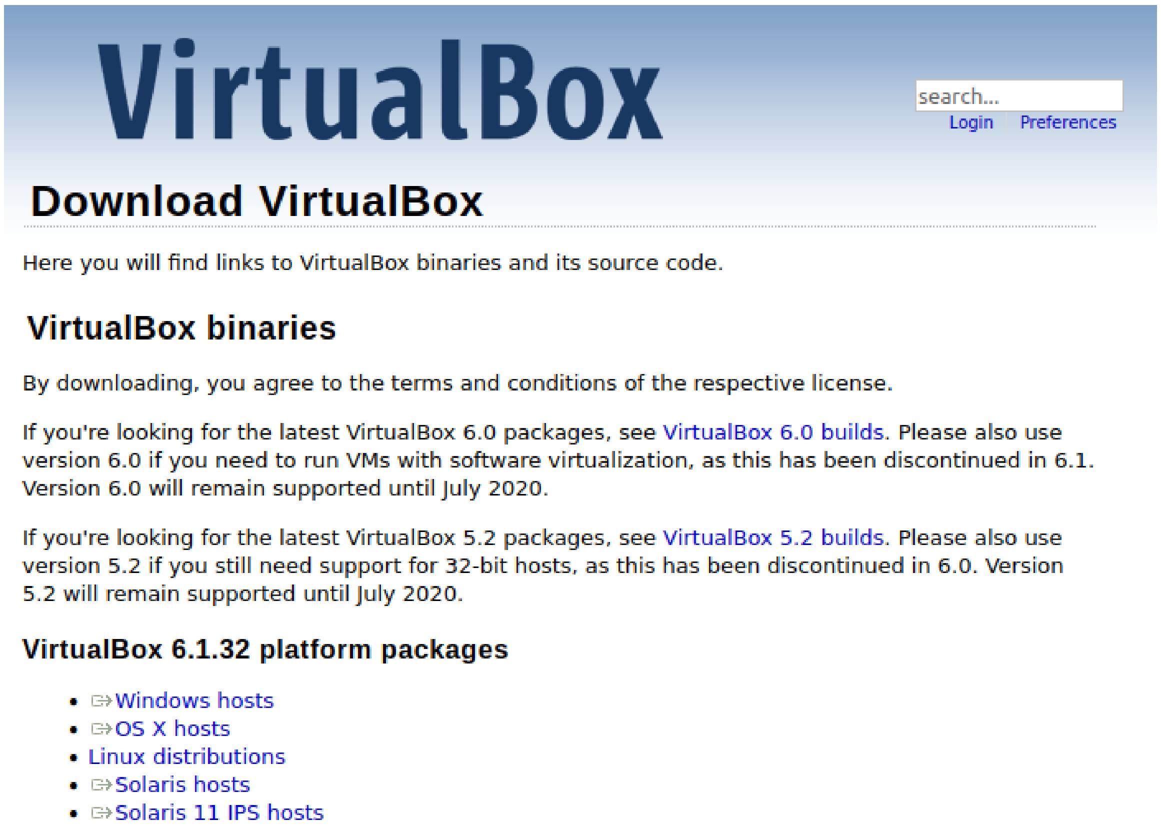
The remainder of this chapter is spent allowing for install and a quick guide to the operating system.

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).



VirtualBox

search...

Login Preferences

Download VirtualBox

Here you will find links to VirtualBox binaries and its source code.

VirtualBox binaries

By downloading, you agree to the terms and conditions of the respective license.

If you're looking for the latest VirtualBox 6.0 packages, see VirtualBox 6.0 builds. Please also use version 6.0 if you need to run VMs with software virtualization, as this has been discontinued in 6.1. Version 6.0 will remain supported until July 2020.

If you're looking for the latest VirtualBox 5.2 packages, see VirtualBox 5.2 builds. Please also use version 5.2 if you still need support for 32-bit hosts, as this has been discontinued in 6.0. Version 5.2 will remain supported until July 2020.

VirtualBox 6.1.32 platform packages

- ⇨Windows hosts
- ⇨OS X hosts
- Linux distributions
- ⇨Solaris hosts
- ⇨Solaris 11 IPS hosts

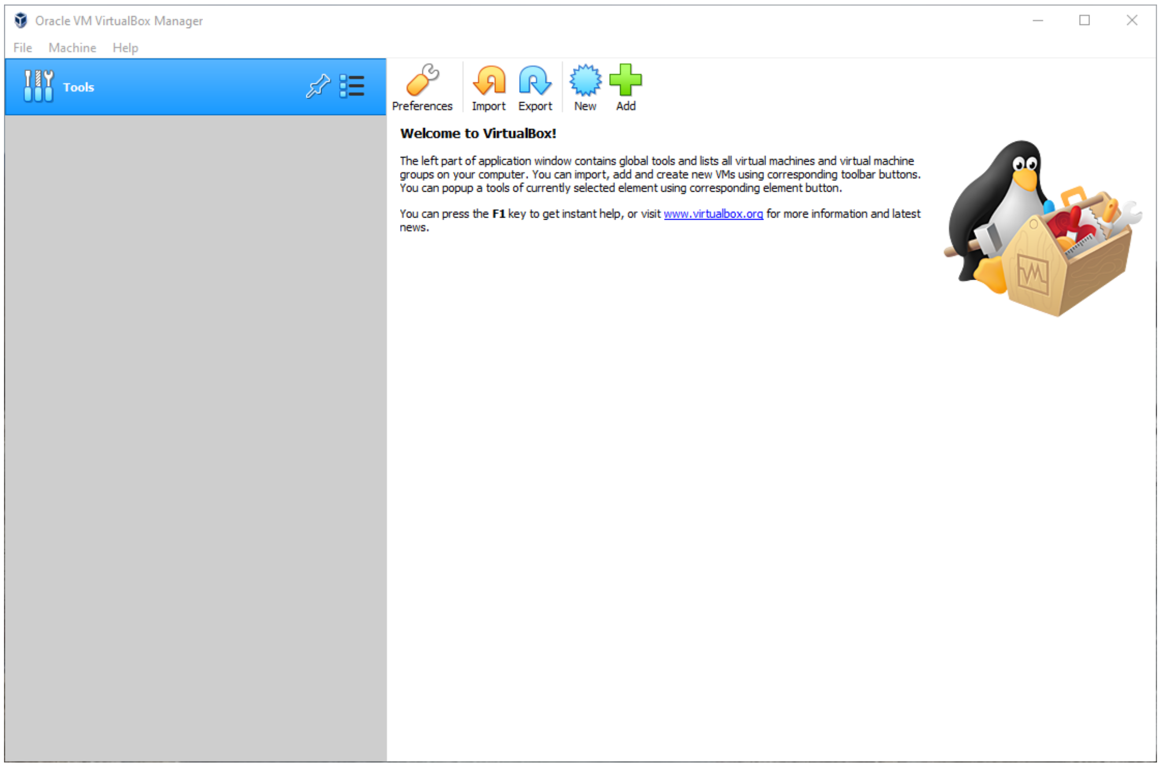
Figure 2.1: VirtualBox Download Page

Run the executable and install VirtualBox by following the instructions. (*Note: if prompted to choose which features are to be installed the following are required at a minimum; USB Support and Networking*).

2.2 Importing the Programming Merit Badge VM

In order to provide the easiest starting spot for the VM a pre-made file has been created. Use a web-browser to navigate to the https://drive.google.com/drive/folders/1rE8jUb4SAwdj-uCSRVqeH7lIxF3ttpCl?usp=sharing and download the `Scouts_Programming_MB.ova` file. This is a fairly large file (it contains an entire computer after all) and may take awhile to download. (*Note: if you are unable to download the file your councilor can provide you a USB drive which contains a copy of the file*)

Now open `VirtualBox` (installed subsection 2.1.1). You should see a screen like shown in Figure 2.2 at which point click the import arrow. This will open an import screen. Use this screen to select the `Scouts_Programming_MB.ova` we downloaded earlier and click *next*.



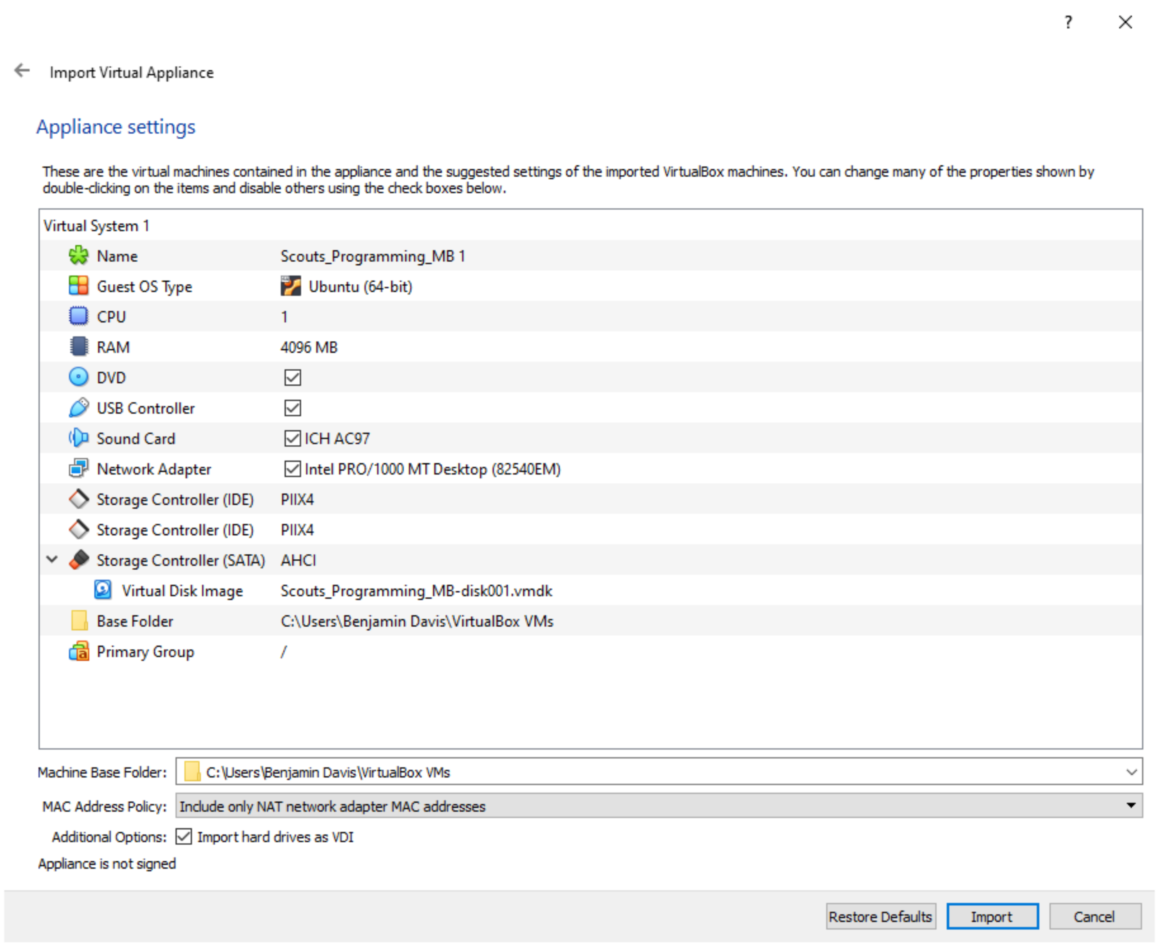
Welcome to VirtualBox!

The left part of application window contains global tools and lists all virtual machines and virtual machine groups on your computer. You can import, add and create new VMs using corresponding toolbar buttons. You can popup a tools of currently selected element using corresponding element button.

You can press the **F1** key to get instant help, or visit www.virtualbox.org for more information and latest news.

Figure 2.2: VirtualBox Home Screen

This should bring up a screen like the one shown in Figure 2.3. Here you are able to set a lot of options about how you want your VM to behave. The two we are most interested in are *CPU* and *RAM*. If you know how many CPU cores or how much ram you have in your computer you can use up to half of each without having any issues. However, if you are unsure how your computer is equipped talk to your councilor or choose the safe *CPU = 2* and *RAM = 2048 MB*.



Appliance settings

These are the virtual machines contained in the appliance and the suggested settings of the imported VirtualBox machines. You can change many of the properties shown by double-clicking on the items and disable others using the check boxes below.

Virtual System 1

Name	Scouts_Programming_MB 1
Guest OS Type	Ubuntu (64-bit)
CPU	1
RAM	4096 MB
DVD	☑
USB Controller	☑
Sound Card	☑ ICH AC97
Network Adapter	☑ Intel PRO/1000 MT Desktop (82540EM)
Storage Controller (IDE)	PIIX4
Storage Controller (IDE)	PIIX4
Storage Controller (SATA)	AHCI
Virtual Disk Image	Scouts_Programming_MB-disk001.vmdk
Base Folder	C:\Users\Benjamin Davis\VirtualBox VMs
Primary Group	/

Machine Base Folder: C:\Users\Benjamin Davis\VirtualBox VMs

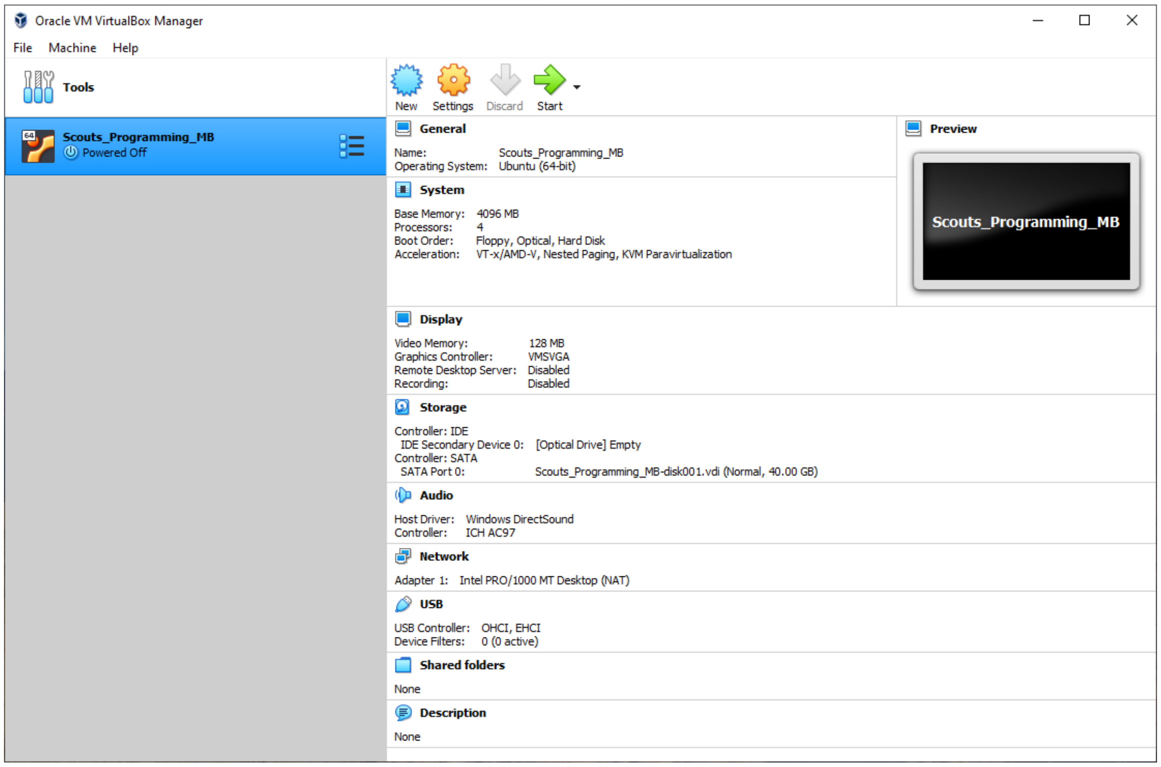
MAC Address Policy: Include only NAT network adapter MAC addresses

Additional Options: ☑ Import hard drives as VDI

Appliance is not signed

Figure 2.3: VirtualBox Advanced Settings

Now hit import and wait a couple of minutes while the VM imports. Once the import is complete you should see the home screen again. This time there should be a Scouts_Programming_MB VM on the left side (as seen in Figure 2.4). Click this VM and then hit start. Now that you've got the VM imported head down to section 2.3 to continue!



Scouts_Programming_MB
Powered Off

General
Name: Scouts_Programming_MB
Operating System: Ubuntu (64-bit)

System
Base Memory: 4096 MB
Processors: 4
Boot Order: Floppy, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, KVM Paravirtualization

Display
Video Memory: 128 MB
Graphics Controller: VMSVGA
Remote Desktop Server: Disabled
Recording: Disabled

Storage
Controller: IDE
IDE Secondary Device 0: [Optical Drive] Empty
Controller: SATA
SATA Port 0: Scouts_Programming_MB-disk001.vdi (Normal, 40.00 GB)

Audio
Host Driver: Windows DirectSound
Controller: ICH AC97

Network
Adapter 1: Intel PRO/1000 MT Desktop (NAT)

USB
USB Controller: OHCI, EHCI
Device Filters: 0 (0 active)

Shared folders
None

Description
None

Figure 2.4: VirtualBox Import Complete

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 2.5

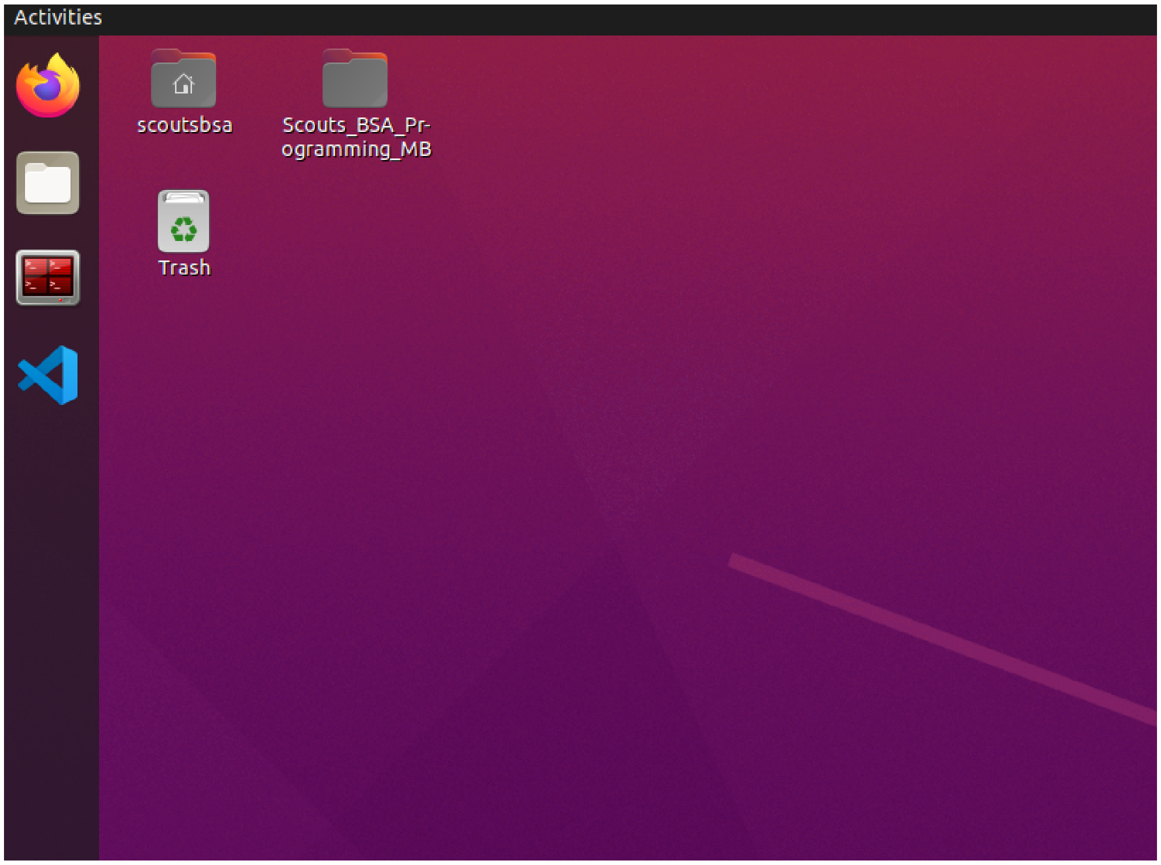
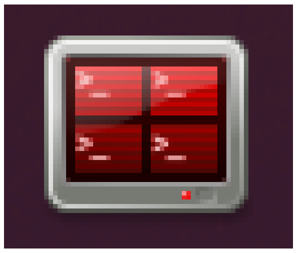


Figure 2.5: Desktop View

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 2.6a). Finally, is the the VS Code icon (see Figure 2.6b). VS Code is a text editor which allows code to be highlighted so it is easier to read.


(a) Terminal Icon

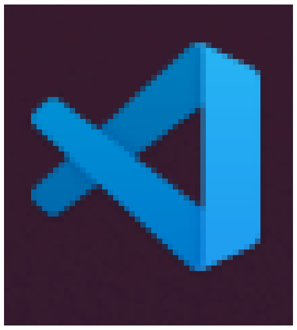
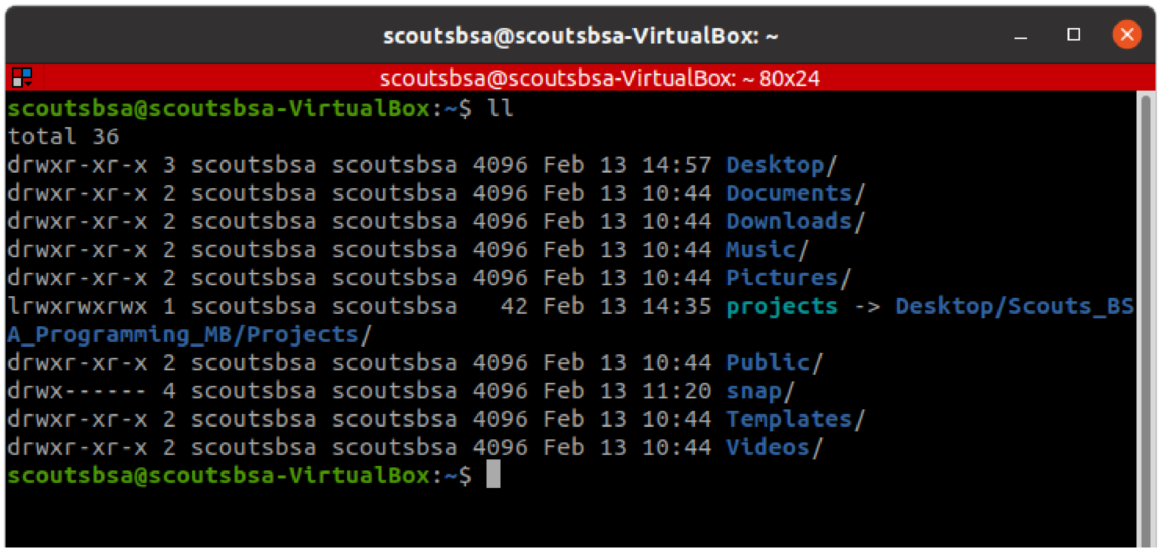

(b) VS Code Icon

Figure 2.6: Program Icons

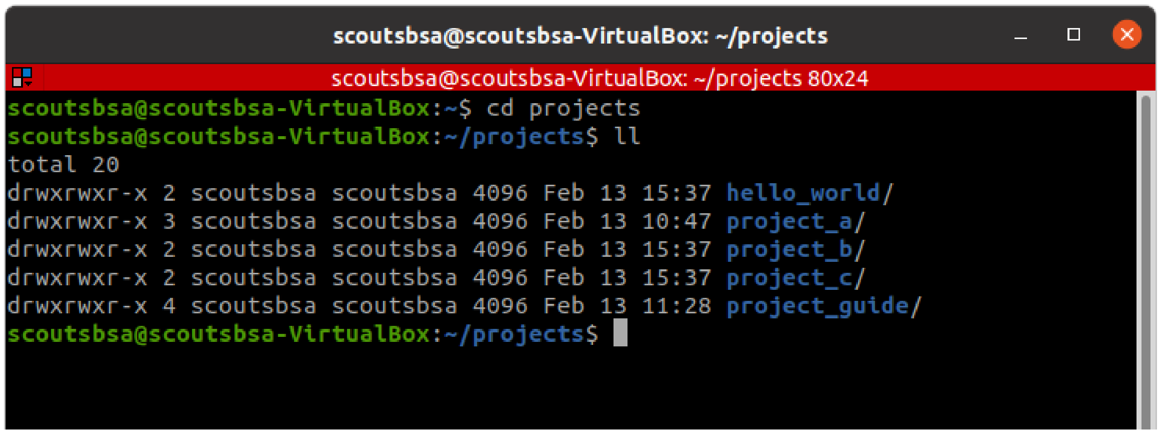
Now click on the terminal open it. This will place you in your *home* folder. Type the `ll` command to view a list of all the files and folders in your *home*.


scoutsbsa@scoutsbsa-VirtualBox: ~
scoutsbsa@scoutsbsa-VirtualBox: ~ 80x24

```
scoutsbsa@scoutsbsa-VirtualBox:~$ ll
total 36
drwxr-xr-x 3 scoutsbsa scoutsbsa 4096 Feb 13 14:57 Desktop/
drwxr-xr-x 2 scoutsbsa scoutsbsa 4096 Feb 13 10:44 Documents/
drwxr-xr-x 2 scoutsbsa scoutsbsa 4096 Feb 13 10:44 Downloads/
drwxr-xr-x 2 scoutsbsa scoutsbsa 4096 Feb 13 10:44 Music/
drwxr-xr-x 2 scoutsbsa scoutsbsa 4096 Feb 13 10:44 Pictures/
lrwxrwxrwx 1 scoutsbsa scoutsbsa   42 Feb 13 14:35 projects -> Desktop/Scouts_BS
A_Programming_MB/Projects/
drwxr-xr-x 2 scoutsbsa scoutsbsa 4096 Feb 13 10:44 Public/
drwx------ 4 scoutsbsa scoutsbsa 4096 Feb 13 11:20 snap/
drwxr-xr-x 2 scoutsbsa scoutsbsa 4096 Feb 13 10:44 Templates/
drwxr-xr-x 2 scoutsbsa scoutsbsa 4096 Feb 13 10:44 Videos/
scoutsbsa@scoutsbsa-VirtualBox:~$
```

Figure 2.7: 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.

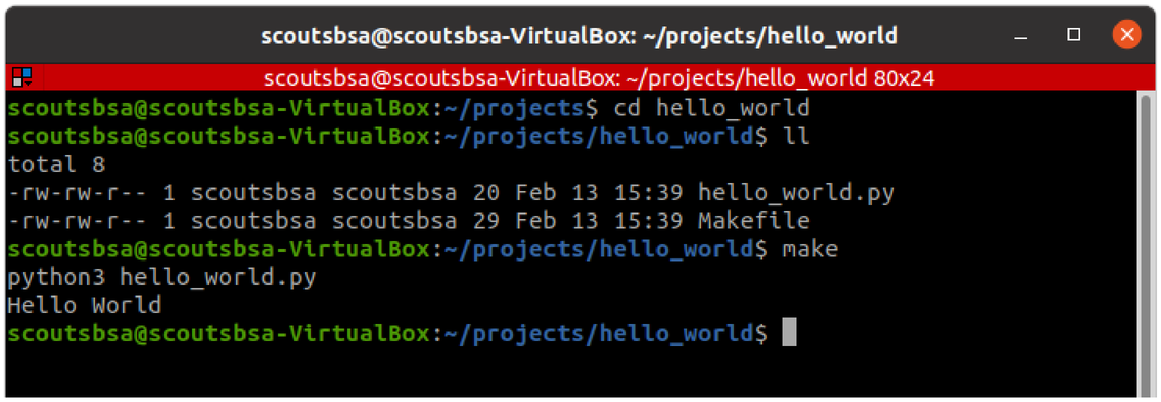

scoutsbsa@scoutsbsa-VirtualBox: ~/projects
scoutsbsa@scoutsbsa-VirtualBox: ~/projects 80x24

```
scoutsbsa@scoutsbsa-VirtualBox:~$ cd projects
scoutsbsa@scoutsbsa-VirtualBox:~/projects$ ll
total 20
drwxrwxr-x 2 scoutsbsa scoutsbsa 4096 Feb 13 15:37 hello_world/
drwxrwxr-x 3 scoutsbsa scoutsbsa 4096 Feb 13 10:47 project_a/
drwxrwxr-x 2 scoutsbsa scoutsbsa 4096 Feb 13 15:37 project_b/
drwxrwxr-x 2 scoutsbsa scoutsbsa 4096 Feb 13 15:37 project_c/
drwxrwxr-x 4 scoutsbsa scoutsbsa 4096 Feb 13 11:28 project_guide/
scoutsbsa@scoutsbsa-VirtualBox:~/projects$
```

Figure 2.8: 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.


scoutsbsa@scoutsbsa-VirtualBox: ~/projects/hello_world
scoutsbsa@scoutsbsa-VirtualBox: ~/projects/hello_world 80x24

```
scoutsbsa@scoutsbsa-VirtualBox:~/projects$ cd hello_world
scoutsbsa@scoutsbsa-VirtualBox:~/projects/hello_world$ ll
total 8
-rw-rw-r-- 1 scoutsbsa scoutsbsa 20 Feb 13 15:39 hello_world.py
-rw-rw-r-- 1 scoutsbsa scoutsbsa 29 Feb 13 15:39 Makefile
scoutsbsa@scoutsbsa-VirtualBox:~/projects/hello_world$ make
python3 hello_world.py
Hello World
scoutsbsa@scoutsbsa-VirtualBox:~/projects/hello_world$
```

Figure 2.9: 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

Project A is intended to fulfill Requirement `5.A` of the programming merit badge:

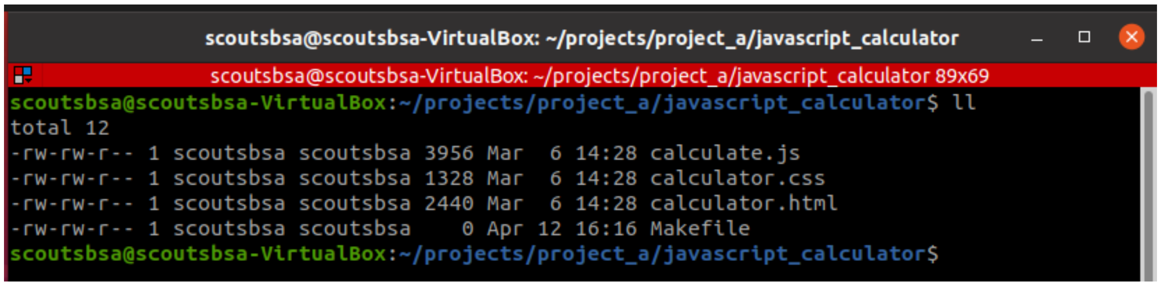
5.A With your counselor's approval, choose a sample program. Then, as a minimum, modify the code or add a function or subprogram to it. Debug and demonstrate the modified program to your counselor.

This is the only project which limits your choices. In fact there is currently only one option: JavaScript Calculator.

3.1 Getting Started

To start navigate to the projects directory like you did in section 2.3. There you should see (using `ll`) a folder called *project_a*; use `cd project_a` to enter it. Here `ll` should reveal there is only one folder *javascript_calculator*, enter it using `cd`. Now you are in the folder which allows you to do Project A.

Listing the files should show that there are 4 files here (see Figure 3.1). These files are discussed in more detail below.


scoutsbsa@scoutsbsa-VirtualBox: ~/projects/project_a/javascript_calculator
scoutsbsa@scoutsbsa-VirtualBox: ~/projects/project_a/javascript_calculator 89x69

```
scoutsbsa@scoutsbsa-VirtualBox:~/projects/project_a/javascript_calculator$ ll
total 12
-rw-rw-r-- 1 scoutsbsa scoutsbsa 3956 Mar  6 14:28 calculate.js
-rw-rw-r-- 1 scoutsbsa scoutsbsa 1328 Mar  6 14:28 calculator.css
-rw-rw-r-- 1 scoutsbsa scoutsbsa 2440 Mar  6 14:28 calculator.html
-rw-rw-r-- 1 scoutsbsa scoutsbsa    0 Apr 12 16:16 Makefile
scoutsbsa@scoutsbsa-VirtualBox:~/projects/project_a/javascript_calculator$
```

Figure 3.1: Project A Files

calculator.js is the *JavaScript* file for the calculator. This file is where you will do all of your work for this project.

calculator.css is the *Cascading Style Sheets* file for the calculator. This file specifies certain things about how the calculator looks such as the colors. You are welcome to edit this file to change colors, but not required.

calculator.html is the *HTML* file which serves as the entry point for this project. *HTML* has been the backbone of the internet for decades, so feel free to look into this file, but there is no need to modify it.

Makefile is a file I have created to help you run this project more easily. That being said it is extremely small and simple so feel free to look into it.

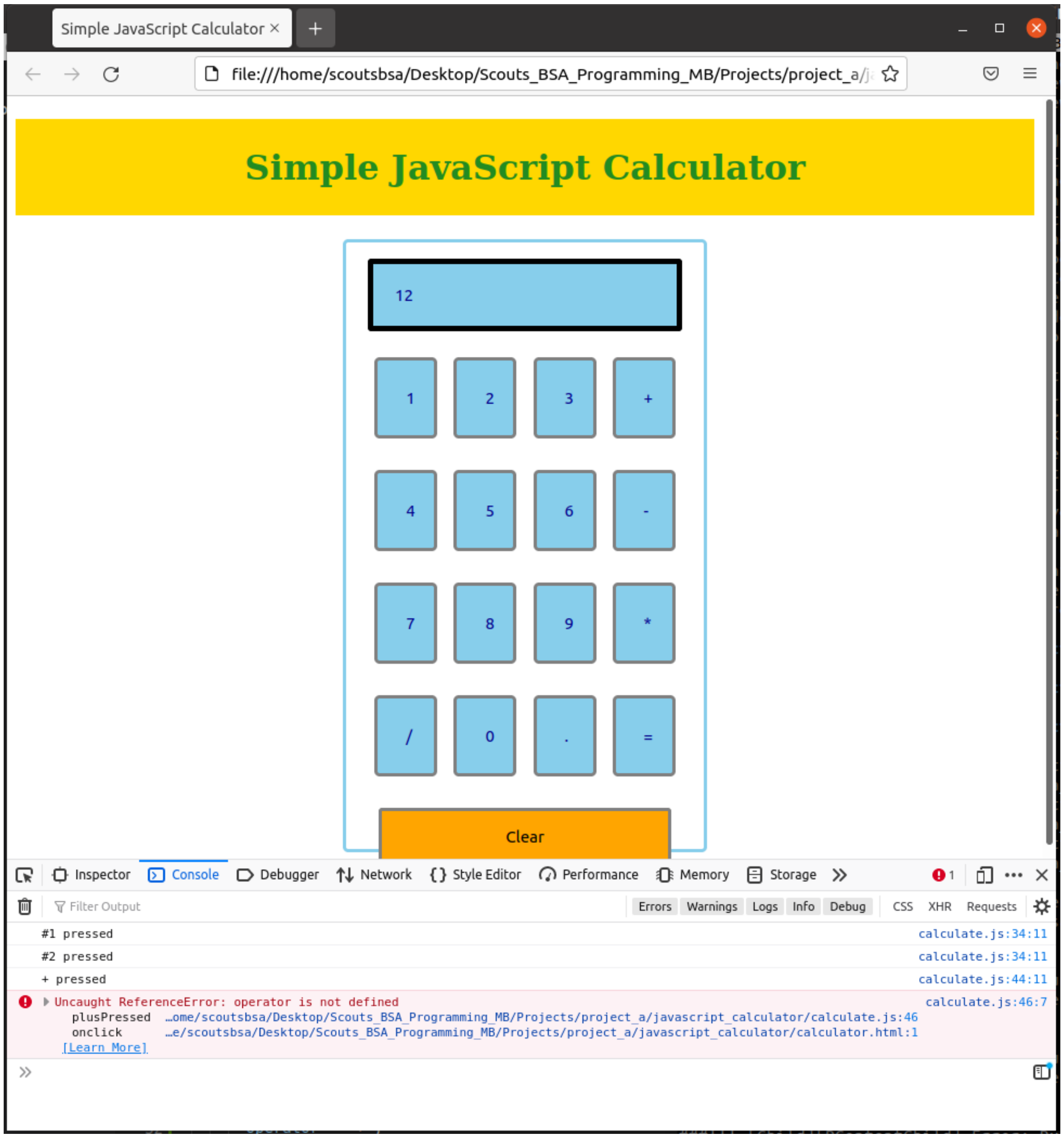
Now that you know what files are in the folder let's do some programming! To start VSCode run `code . &`. Once VSCode is open click the *calculate.js* file on the left-hand side.

3.2 ToDo 1 - Creating a Variable

Throughout this code you will see 4 comments which contain ToDo's. I would recommend working through them in the order provided.

To start run the calculator using `make`. This will open a browser window (like the one in Figure 3.2). When you hit a number you should see a message in the console (bottom of the screen telling you which number is pressed).

Now try pressing '+'. What do you see in the console? If you haven't hit the *clear* button yet then you should see a message followed by an error that says `operator is not defined`.



```
#1 pressed                                                        calculate.js:34:11
#2 pressed                                                        calculate.js:34:11
+ pressed                                                         calculate.js:44:11
Uncaught ReferenceError: operator is not defined                  calculate.js:46:7
  plusPressed  …ome/scoutsbsa/Desktop/Scouts_BSA_Programming_MB/Projects/project_a/javascript_calculator/calculate.js:46
  onclick      …e/scoutsbsa/Desktop/Scouts_BSA_Programming_MB/Projects/project_a/javascript_calculator/calculator.html:1
  [Learn More]
```

Figure 3.2: Project A Files

So let's fix this. Find the comment which contains the ToDo 1 task '*Create a variable called 'operator' to represent the last operator clicked*'. The two lines above the comment declare other variables. Can you figure out how to create the new one?

Once you think you've got it try running `make` again. You shouldn't see the same error and all the numbers should work. Additionally, '+', '=', and clear should work for addition.

3.3 ToDo 2 - Variable Assignment and Display

Once you've got addition working let's get subtraction going. Find ToDo #2 in the `minusPressed` function.

For this part of the task we are doing two things:

1. Set the previous value variable to the current value
2. Display the current value

This was done in the `plusPressed` function. Can you use it to perform this task?

Once you've worked through this ToDo, run `make` again. Now addition and subtraction should both be working!

3.4 ToDo 3 - Add Multiplication Functionality

This next step should enable multiplication to work correctly. Find the `calculate` function and ToDo #3 inside. This task is to add capability for the calculate function to handle multiplication and division.

Again this function already contains how it is done for addition and subtraction. How do you think it is done for multiplication and division?

Once you've gotten it done try multiplying with the calculator. Be ready to explain why changing this function seems to change how the `timesPressed` function works.

3.5 ToDo 4 - Add Division Functionality

This final step is the most complex. Use everything you've learned so far to create a function called dividePressed which performs division. Once this is working you are done with Project A!

Discuss the findings from this project with your councilor.

Chapter 4

Project B

Chapter 5

Project C