

CS2040 Development Environment

Esubmit provides little debugging support. These directions allow you to build and run programs on your machine so you can do things like set breakpoints to examine data at runtime. This uses a “virtual machine”, a program that runs software in another operating system on your Windows computer. In this case, you will install a Linux virtual machine and then build and run your code in that environment.

1. Install VirtualBox

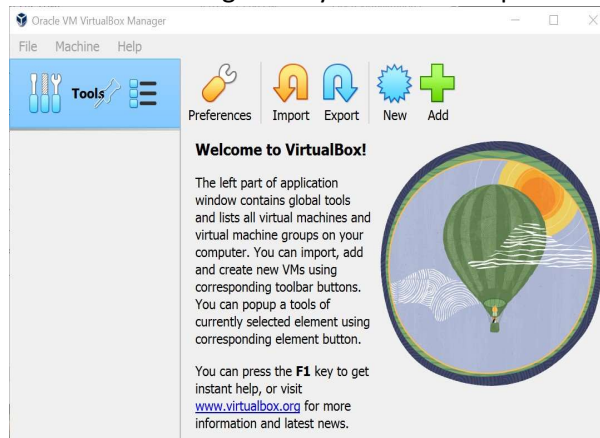
- (a) Go to <https://www.virtualbox.org/wiki/Downloads> in your browser (or search for VirtualBox and go to the downloads page)



- (b) Click the link for Windows Hosts. This will download a file called “VirtualBox-7.0.6-155176-Win.exe”. The exact numbers may vary depending on when new versions become available; download the latest version you can.
- (c) Now open the directory containing the VirtualBox installer (“VirtualBox-7.0.6-155176-Win.exe”) and run it. You can accept the default configuration.

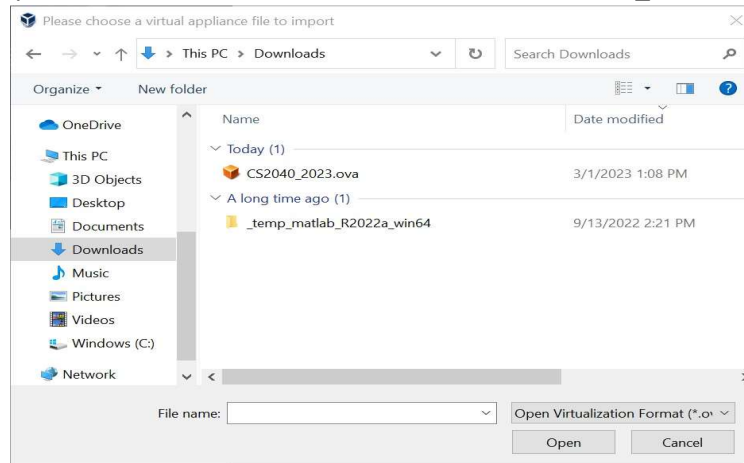
2. Download the VM

- (a) Click on this link: <https://msoe.app.box.com/s/c95d29z6ke8glnxei769ctyrb52wts6c> and then click on the Download button. This will download the image for the virtual machine you will use. This will take approximately half an hour to download and longer if lots of other students are downloading it at the same time.
- (a) Run VirtualBox. It may have started running already as the last step in the installation.



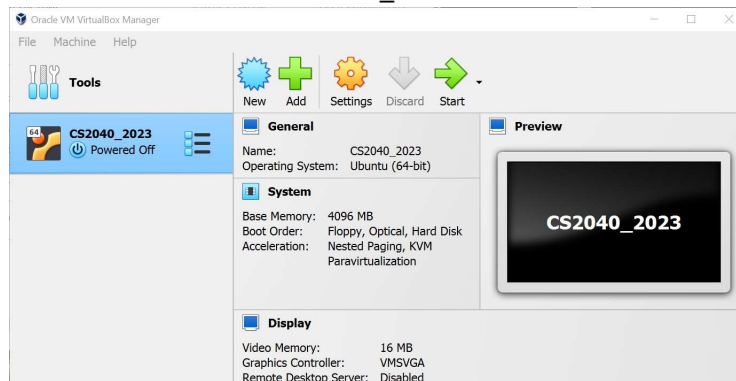
- (b) Select “Import” (the yellow U-shaped arrow).

(c) Navigate to where you downloaded the ova file. Select the file “CS2040_2023.ova”



(d) Back in the import dialog, click on the **Finish** button. An “Importing appliance” dialog will come up; wait for it to complete.

(e) You should now see a machine named “CS2040_2023” in VirtualBox.

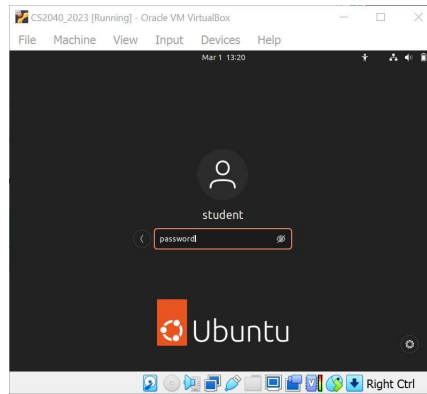


Click on the **Start** arrow to run the machine.

4. A bit about your new friend, Ubuntu

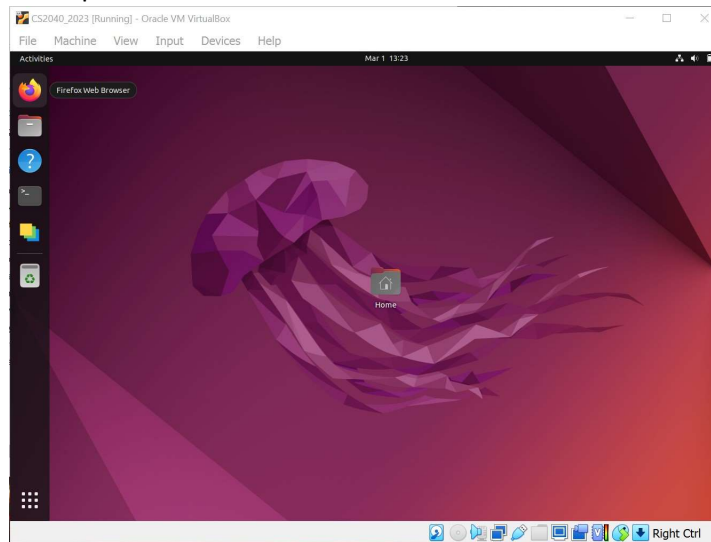
(a) Make sure the VM is running. If it isn't, you can use the Start arrow or just double-click to run it. It takes a while for the system to boot. If there are messages about auto-capturing the mouse and keyboard, you can close them. Those messages are more for people who have used VirtualBox in the past.

(b) The login name is “student”; the password is “password”.



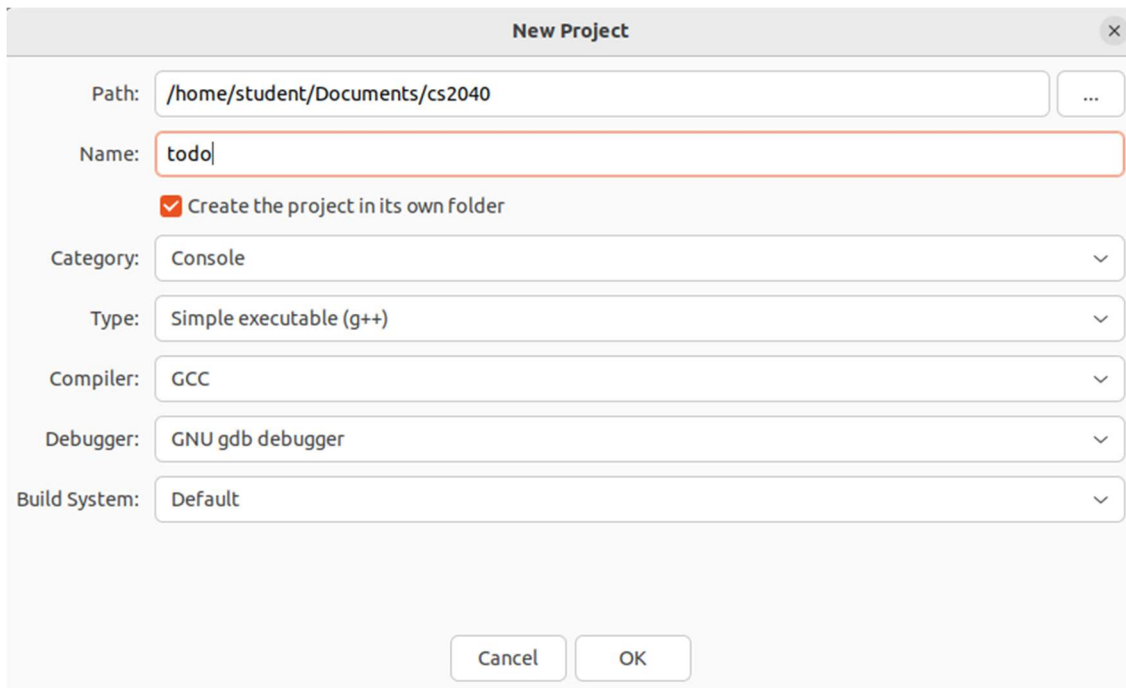
Do not use this such a simple password for non-virtual machines (or any machines you want to keep secure)!

- (c) Once you have logged in, you will see a desktop. Some common applications are on the left. Notably, the left-top is Firefox, and the bottom color icon is the CodeLite IDE. We will use CodeLite for C/C++ development.



- You can resize this window as necessary.
 - You can also upgrade the Guest Additions if you want by choosing that option from the “Devices” menu.) This can be used to enable accessing files on your host computer, but such options are not mandatory (and not described here).
 - Clicking the nine dots in the far bottom left will list all applications.
5. Your virtual machine is now ready. Testing it:
6. Using Firefox, visit <https://faculty-web.msoe.edu/hasker/cs2040/samples/todo> and download the file todo.zip. If accessing the web page doesn't work, try opening a terminal window and entering the command
- ```
wget https://faculty-web.msoe.edu/hasker/cs2040/samples/todo/todo.zip
```
- A simple way to transfer files between your VM and your laptop host is to upload the files to a server like Box or GoogleDocs and then download it from there to your VM or laptop. You can also set up git on your VM; see the CS 2040 resources page for details.

7. Start **CodeLight** and click on the **New Workspace** button, set the workspace type to **C++**, the default Workspace Path, and use **cs2040** as the Workspace Name.
8. Right click on **cs2040** (under the Workspace View tab on the left) and select **New Project**.
9. In the New Project dialog, use the 3 dots to the right of **Path** to browse to where you want the project to be located; probably in the cs2040 workspace. Name the project “todo”. Set the **Category** to **Console**, **Type** to **Simple executable (g++)**, **Compiler** to **GCC**, and **Debugger** to **GNU gdb debugger**, and **Build System** to **Default**. Note you might need to grab the bottom edge of the window and expand it to be able to set the Build System. Click **OK**.



The image shows a 'New Project' dialog box with the following fields and values:

- Path: /home/student/Documents/cs2040
- Name: todo
- Create the project in its own folder
- Category: Console
- Type: Simple executable (g++)
- Compiler: GCC
- Debugger: GNU gdb debugger
- Build System: Default

Buttons: Cancel, OK

10. Browse into the **src** folder of the **todo** project and open **main.cpp**. It’s a simple hello world program. Run it by clicking **Build | Run**. This will pop up a terminal window that shows the hello message followed by **Hit any key to continue...** Press the **Enter** key.
11. By default, the CodeLite window fills your screen. Click on the double-square button in the upper right to get out of that mode. This will show an Output View window. A known annoyance is that you can't move this window, but you can at least move the main window so that the output view can be clicked into the foreground when you need it and you can resize the Output View so the full line is visible.
12. At this point you have a building project, but it’s just a hello world program. Let’s replace this project by the code from the todo project you downloaded earlier. Open the Files window (click on the gray file folder icon that should be near the top left corner) and browse to your todo.zip – it is likely in your Downloads folder. Double-click on todo.zip to open it, click on **Extract** in the upper left corner, select a location to extract it to (perhaps Documents), then click the green **Extract** button in the upper right. Then browse into the extracted folder and select all of the files but the Makefile. You can do this by control-clicking on each source file (.h and .cpp). Right click and select **Copy**. Then browse to your new **todo** project *in the cs2040 workspace*, right click, and select **Paste**.
13. In CodeLite, select the **todo** project, right click on **main.cpp**, and select **Remove**. If prompted, it’s advisable to remove it from the disk, though you can make things work without deleting it.

14. Right click on the **src** icon and select **Add an Existing File...** Use control-clicks to select just the .h and .cpp files and add them to the project by clicking on the Open button in the upper right.
15. Select **Build | Run** to run the project. Note you can have it always **Build and Execute** when you run projects. You should see a Terminal window displaying “Total time: 23.67” followed by the standard “Hit any key” prompt.
16. Add a breakpoint: click to the right of line 15 in day.cpp so there’s an orange circle:

```

5
6 #include "task.h"
7 #include "tasklist.h"
8 #include <iostream>
9 using namespace std;
10
11 int main()
12 {
13 TaskList *tasks = new TaskList();
14 tasks->add(new Task("sleep", 6.5));
15 tasks->add(new Task("shower", 0.17));
16 tasks->add(new Task("program", 16.5));
17 tasks->add(new Task("eat", 0.5));
18 cout << "Total time: " << tasks->totalHours()
19 << endl;
20 return 0;
21 }

```

17. From the **Debugger** menu, select **Start/Continue Debugger**. The program will start, and the terminal window will contain the warning “GDB: Failed to set controlling terminal: Operation not permitted.” You can ignore this warning. Note there is a green arrow on line 15. At the bottom of the screen is a Debugger window with a Locals tab. Click on that tab (if it’s not already selected) and click on the “>” to the left of tasks. This allows you to view the data in the class instance tasks. Note that the count is set to 1.
18. There are run controls at the top of the CodeLite window. Click on the **downarrow** to execute adding one more task. You’ll note the debugger still shows the count at 1. Click the **refresh loop** in the **Debugger** window to update the information. Explore the other debug controls as well. Note that the **run** button in this group (an open green triangle) will run the program to completion (and automatically close the output Terminal) unless you add a breakpoint at the final return.

You will want to explore other features of CodeLite. Learning to use IDEs effectively and learning debugging techniques are both important parts of computing!