ECE 250 — Winter 2024
Electrical and Computer Engineering
University of Waterloo

Due Date: Jan 29, 2024
Professor: Ziqiang Patrick Huang
Lab Instructor: Ahmed Fahmy

Lab 0:

Music Playlist

1 Overview

The objectives of Lab 0 are:

- Set up a development environment
- Implement a program of songs playlist
- Use GitLab to clone and submit project files.

2 Development Environment Set-up

How you compile, run and test your code is up to you. However, keep in mind that **your code will be tested and graded on Linux**. So, test on Linux! In this lab, we will show you how to use remote development via VS Code.

VS Code

1. Download and install VS Code from here.
2. Open VS Code and install *Remote Development* extension pack as shown in Figure 1.

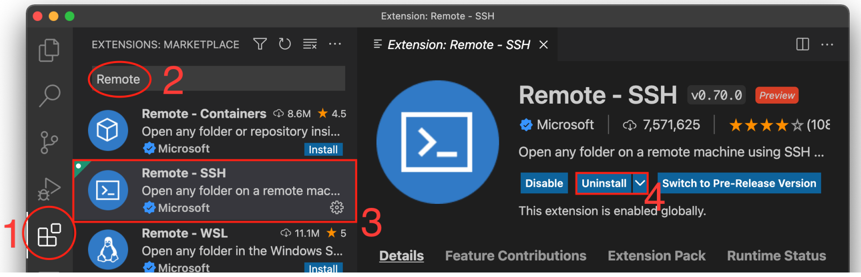

Extension: Remote - SSH
EXTENSIONS: MARKETPLACE
Extension: Remote - SSH
Remote
2
Remote - Containers 8.6M 4.5
Open any folder or repository insi...
Microsoft
Install
Remote - SSH
Open any folder on a remote mac...
Microsoft
3
1
Remote - WSL 11.1M 5
Open any folder in the Windows S...
Microsoft
Install
Remote - SSH v0.70.0 Preview
Microsoft 7,571,625
Open any folder on a remote machine using SSH ...
Disable
Uninstall
Switch to Pre-Release Version
4
This extension is enabled globally.
Details
Feature Contributions
Extension Pack
Runtime Status

Figure 1: Steps to install the *Remote Development* extension pack: (1) Click on the *Extensions* icon. (2) Search "Remote". (3) Select *Remote-SSH*. (4) Click on Install.

3. Open Command Palette, by pressing **F1** in VS Code, and select *Remote-SSH: Connect to Host*, as shown in Figure 2.
4. You should be prompted to enter your SSH[1] user@host (e.g. **userID@eceubuntu.uwaterloo.ca**).
5. Enter your password

[1]It is highly recommended to set up an SSH key to avoid being asked for the password at every step. **Note:** If you are off-campus, make sure that you are connected through the VPN.

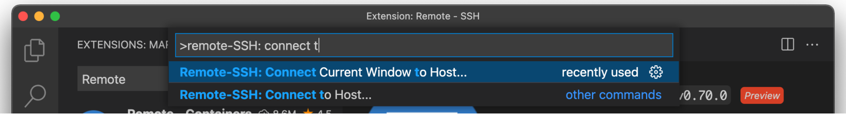


Figure 2: Command Palette in VS Code.

6. Go to *Explorer*, then click on *Open Folder*.
7. Choose a project folder/directory, or you can clone a GitLab repository (more about this later).
8. Now, you can upload project files: open *Explorer* in VS Code, then drag and drop your files into the left panel, as shown in Figure 3.

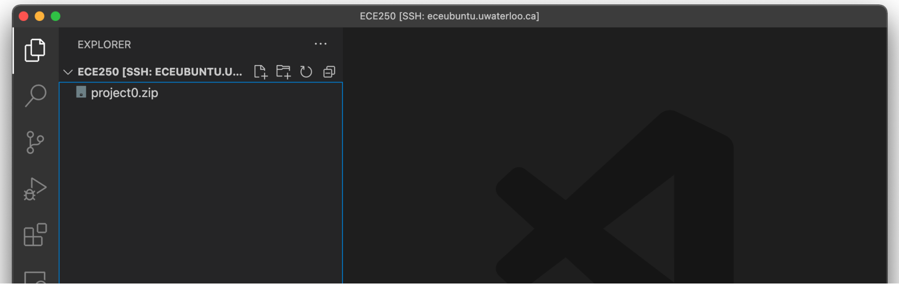
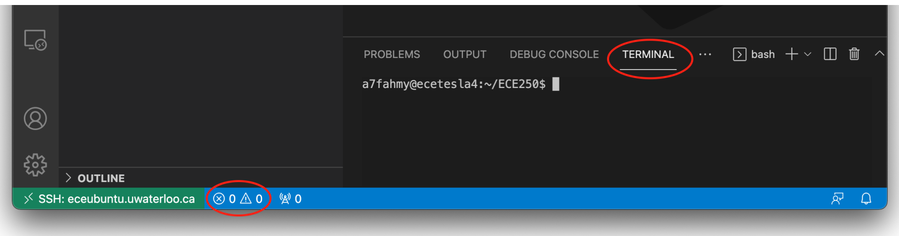

ECE250 [SSH: eceubuntu.uwaterloo.ca]
EXPLORER
ECE250 [SSH: ECEUBUNTU.U...
project0.zip

Figure 3: Drag and Drop files into the left panel.

9. Open the terminal from within VS Code by clicking on the error icon and then on the terminal tab, as shown in Figure 4.



```
a7fahmy@ecetesla4:~/ECE250$
```

Figure 4: Open the terminal in VS Code.

10. If your files are *zipped*, unzip using the following command: `unzip FILENAME.zip`.
11. Install C/C++ in VS Code by going to *Extensions*, search “C++”, select *C/C++*, then select *install on eceubuntu.uwaterloo.ca*.
12. Open your “.cpp” file in VS Code from the *Explorer*.
13. Go to *Run and Debug*, the “play and bug” icon on the left bar, then click *Run and Debug*.
14. Choose C++ (GDB) from the Command Palette. Then, choose the g++ debugger (`/usr/bin/g++`).
15. Open the terminal in VS Code to see that your app is compiled and running!

3 Music Playlist

The goal of this project is to write a C++ program that manages a playlist of songs, where the user can:

1. **Add** a song to the end of the list, **if possible**
2. **Select** a song to be played, **if it exists**
3. **Delete** a song from the list, **if it exists**

Requirements

You **must use an array** to store the playlist. **Using the STL library is not allowed**. Each entry in the array must store a song's title and artist. In addition, the program has restrictions such that the following songs can never be added to the playlist:

- "Baby" by Justin Bieber[2]
- "My Heart Will Go On" by any artist.

Your program must read commands from standard input (e.g. `cin`) and write to standard output (e.g. `cout`). The program is required to respond to input commands with the corresponding output messages as outlined in Table 1. The outputs specified in the "Output" column must be displayed as indicated. For example, in the first row, the expected output is the string "success" in lowercase.

Command	Parameters	Description	Output
`m`	`N`	Sets the maximum playlist size to N **Note:** This command is executed exactly once at the beginning.	`success`

`i`	`t;a`	Adds song (`t`) by (`a`) at the end of the playlist **only if:**
		• it is not already in the list,
		• it is not a restricted entry, and
		• the list is not full
			`success`
			OR
			`can not insert t;a`

`p`	`n`	Plays song at position `n`, if it exists	`played n title;artist`
			OR
			`can not play n`

`e`	`n`	Erases song at position `n`, moves any songs after position `n` up by 1	`success`
			OR
			`can not erase n`

`done`	N/A	Ends the program	N/A

Table 1: Input/Output Requirements

Provided Files

You are provided with the following files: `driver.cpp` where the `main()` function is implemented, `playlist.h` where your classes are *defined*, and `playlist.cpp` where your classes are *implemented*. **Do not create, delete or change the name of the provided files**. In addition, there are some examples of input files along with their corresponding output files. The test files are named `test01.in`, `test02.in`, and so on, with their respective output files named `test01.out`, etc.

[2]The official music video for "Baby" was the most disliked clip on YouTube until 2018. It was voted the worst song ever in a 2014 Time Out poll.

4 GitLab Basics

We are going to use GitLab for code management and submission. For every lab, a repository is created for you with the URL:

```
https://git.uwaterloo.ca/ece250-w24/labX/ece250-w24-labX--USERID
```

where `X` is the lab number, and `USERID` is your user ID.

Clone

A clone is a full copy of a repository, including all logging and versions of files. To clone a lab project repository, type the following command in your terminal:

```
git clone https://git.uwaterloo.ca/ece250-w24/labX/ece250-w24-labX--USERID
```

Commit

One of the core functions of Git is the concept of *Commit*. Adding commits helps you keep track of your progress and changes as you work. In Git, each commit serves as a change point or "save point," acting as a reference point in the project that can be revisited if needed, such as for bug fixes or modifications. When committing changes, it's essential to include a meaningful message. Clear and descriptive commit messages make it easy to understand what changes were made and when they occurred.

The `commit` command performs a commit, and the `-m "message"` adds a message. Example:

```
git commit -a -m "First release"
```

Tha flag `-a` skips *staging*.

Revert

We use the command `revert` when we want to take a previous `commit` and add it as a new commit. To see your latest commits:

```
git log -oneline
```

The to `revert` to the `X`th latest commit:

```
git revert HEAD x
```

where `X = 0` means the latest commit.

Push/Pull

The command `git pull` is used to bring remote changes to your local repository, ensuring your local branch is up-to-date. On the other hand, `git push` is used to send your local changes to the remote repository, making them accessible to others that have access to the project, including graders. Only last pushed version will be graded.

More

For more information about Git repositories, you can check w3schools.