M4 – IDEs (VS Code)

CS 136L F23 – LEC 6

Yiqing Irene Huang, Qianqiu Zhang

UNIVERSITY OF WATERLOO | FACULTY OF MATHEMATICS
DAVID R. CHERITON SCHOOL OF COMPUTER SCIENCE

Disclaimer

- The following slides were not presented page by page in class.
- They are my own study notes to share with students.
- In the lab session, we will cover key points, do small demos and give hints on commonly seen errors

Main Points

Learn how to use the VS Code IDE to edit, compile and run C programs

- Setup VS Code and load/write C programs on your own machine
- Connect to remote servers through VS Code and develop code on remote host
- Leverage features of the VS Code Editor for writing C programs
- Understand basic C/C++ standards and popular compilers
- Compile and run C programs from linux command-line shell
- Compile and run C program within the VS Code IDE

Setup ssh keys

- Login onto the remote host without a password
  - Generate the private/public key pair using `ssh-keygen` (M0.4)
  - Enter an empty passphrase
  - Add the public key contents to the remote host ~/.ssh/authorized_keys
    - When you do not have **`ssh-copy-id`**

```
scp .ssh/id_rsa.pub cs136:.ssh/pub_key1
ssh into the remote host
cd .ssh; cat pub_key1 >> authorized_keys
```

    - When you do have **`ssh-copy-id`**

```
ssh-copy-id user@host
```

Remote-ssh

- Use the explicit host name and use the same one, let N be the last digit of your student ID number, then use
  - `ubuntu2204-002.student.cs.uwaterloo.ca (N=0,1)`
  - `ubuntu2204-004.student.cs.uwaterloo.ca (N=2,3)`
  - `ubuntu2204-006.student.cs.uwaterloo.ca (N=4,5)`
  - `ubuntu2204-008.student.cs.uwaterloo.ca (N=6,7)`
  - `ubuntu2204-014.student.cs.uwaterloo.ca (N=8,9)`
- Install necessary remote extensions, but not too many
- If you use too much resource, CSCF will kill your processes

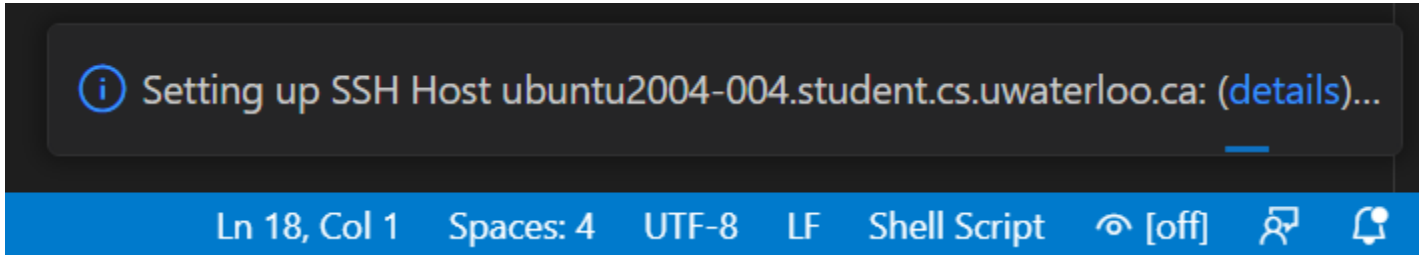
Warning from CSCF

When using VSCode, each student should consistently use the same ubtuntu machine and limit downloading extensions. If students download extensions that use lots of resources or cause problems on the server, we will kill their process.

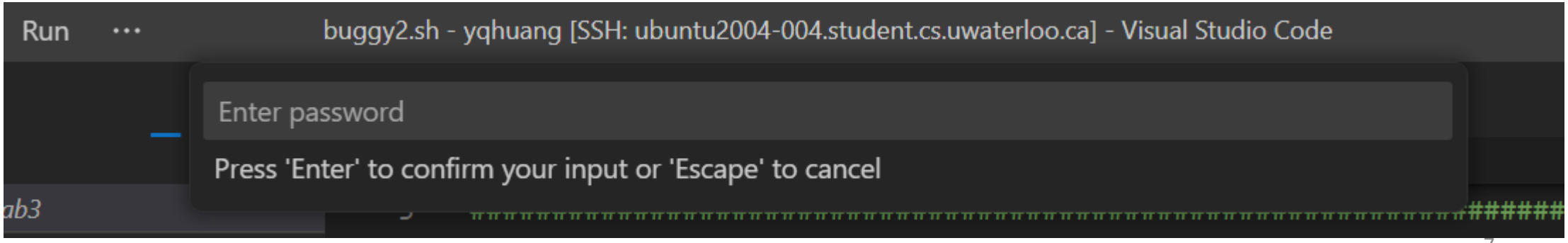
If you are going work remotely, you will need to install extensions for your remote environment AGAIN and AFTER you have established the SSH connection.

Connect to Remote Host

- There is a little window at the lower right corner when connecting the remote host, click “details” and open it to see things under the hood



- On the top, there is a prompt to ask for password



Failed to Connect to Remote Host

- The remote-ssh server puts its own meta data under ~/.vscode-server
  - Last resort to solve remote connection problem is to delete this directory
- Directly removing the .vscode-server/ takes LONG. The following is a workaround
  - `mv ~/.vscode-server ~/vscode-server-2rm`
  - `rm –rf ~/vscode-server-2rm`
  - Be cautious to use rm –rf, you cannot undo afterwards

Setup

- From Microsoft
  - Getting Started (Setup)
  - VS Code Extensions
    - C/C++ programming language extension / C/C++ Extension Pack
    - Remote –SSH

Editor

- The include path
- The IntelliSense
  - Auto completion
  - Quick info
  - Parameter help
- The .json extension configuration file
  - The include path, delimiter is comma
  - Remote extension install and configuration
- Context menu
  - Go to Definition
  - Peek into Definition
  - References

Compiler

- C/C++ Standards
  - Popular C standard: C99
  - Latest C standard: C17
  - Latest C++ standard: C++20 -> C++23
- Compilers
  - Microsoft Visual C/C++ compiler (MSVC)
  - GNU Compiler Collection (gcc)
  - clang/LLVM
    - CS136 uses clang version 14
    - The error and warning messages provided by clang are the best
    - Faster memory error checking, using `AddressSanitizer`

The compiler path and version

```
$which clang
/usr/bin/clang

$clang --version
Ubuntu clang version 14.0.6
Target: x86_64-pc-linux-gnu
Thread model: posix
InstalledDir: /usr/bin
```

Activities

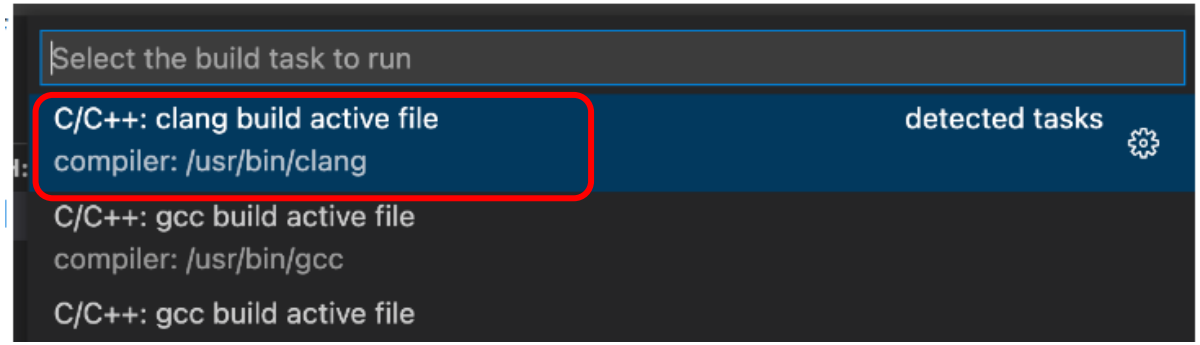
- Check the path of `diff, cat, vim, gcc, …`
- Check the version of `diff, cat, vim, gcc, …`

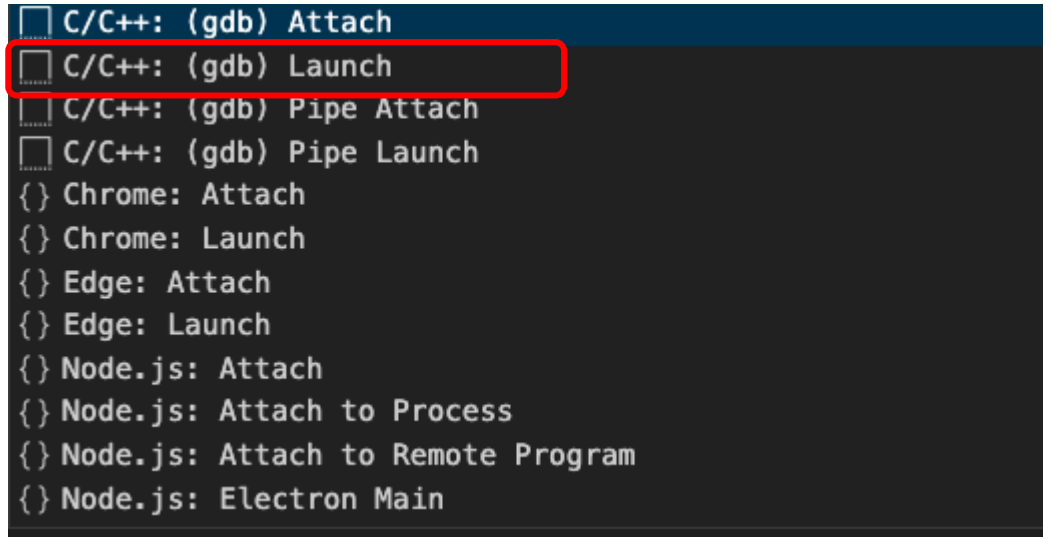
The clang compiler

- `-o`: name the output of the compiler (i.e. the executable, default is `a.out`)
- `-std=c99`: use c99
- `-Wall`: turn on all warnings
- `-I`: specify the include path
  - Use multiple `-I` to include multiple include paths
  - Only specify the directory name, do not include the file name in the path.
- Except for `-o`, order of arguments does not matter
- Examples
  - `clang main.c -o myexe1 -std=c99 -I/u2/cs136l/pub/common`
  - `clang -std=c99 -I/u2/cs136l/pub/common main.c -o myexe1`
  - `clang main.c -std=c99 -I/u2/cs136l/pub/common -o myexe1`

Using VS Code

- You need to have a folder open
- You need to have your main.c open
- Create the tasks.json file to compile
  - Terminal -> Configure Tasks
  - The “label” attribute
  - The “args” attribute
    - The include path argument
    - The object file argument
- Create the launch.json file
  - The “program” attribute
  - The “preLaunchTask” attribute





Lab4 Q1

./compile exec [.in] [.args] file.{c,o} [compiler arguments]

Examples
./compile exe1 main.c
./compile exe2 main.c file1.c file2.c /u2/cs136l/pub/common/cs136.o
./compile exec test.in main.c -g file2.c ~cs136l/pub/common/cs136.o
./compile yolo test.args main.c file1.c -std=c99 -Wall /u2/cs136l/pub/common/cs136.o -g
./compile a.out test.in test.args main.c /u2/cs136l/pub/common/cs136.o -I/u2/cs136l/pub/common/

Lab4 Q1 Cont'd

- Test cases: `/u2/cs136l/pub/lab4/q1/`
- Coding tips
  - `[ "${filename##*\.}" == "in" ]:` test file extension
  - `shift`: move all arguments to the left by one
  - `$@:` give a space delimited list of all the arguments
  - `[ -z $INFILE ]:` test string is empty

Lab4 Q2

- You need to put the run.txt and *.json files in the same directory and zip.
- Do not nest .json files in the .vscode directory
- You may name the .zip file any name you want if we do not specify the submission file name

Acknowledgement

- Slides by courtesy of Carmen Bruni and Dave Tompkins
- Demo notes from Dave Tompkins
- Demo lectures by Carmen Bruni, Dave Tompkins and Nomair Naeem

References

- CS 136L edX notes at https://online.cs.uwaterloo.ca/