



SIMON FRASER UNIVERSITY
ENGAGING THE WORLD

CMPT 125 - Introduction to Computing Science and Programming II - Fall 2021

Lab 1.
Jan 18.

TA: Sepid (sepidh@sfu.ca)



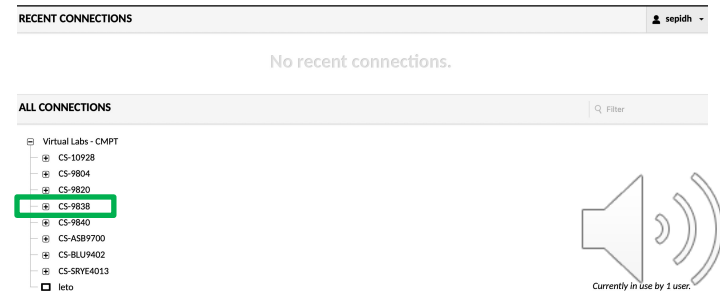
- How to connect CSIL lab computer
- Basic commands in Linux
- VSCode Configuration
- Write Hello_Word.c
- Debug and Compile using VS code
- How to compile when there is more than one file using Vs code
- Compile using Terminal
- What is the make file how to use it
- What is library and .h files
- HW1
- Examples



How to connect to CSIL computer

Using Guacamole

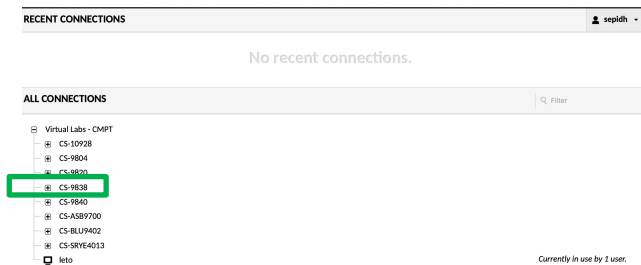
- 1: Go to this link and log in: <https://cas.sfu.ca/cas/login?service=https://gateway.its.sfu.ca/guacamole>
- 2: From the virtual labs choose any, Preferably choose cs-9838 and then
3. Choose one of the machines there



How to connect to CSIL computer

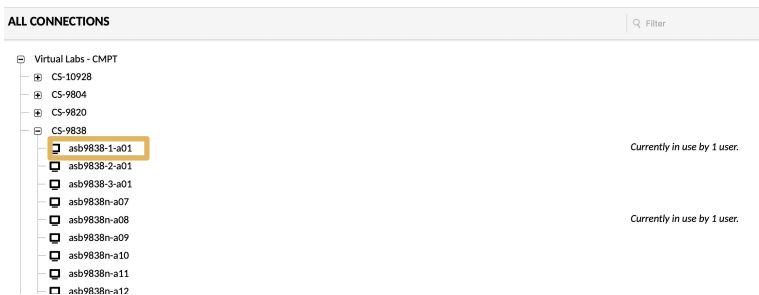


1.



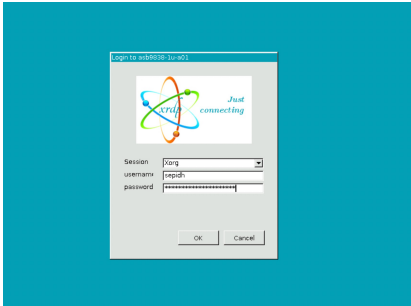
Choose any machine: ex:9838-1-a01

2.



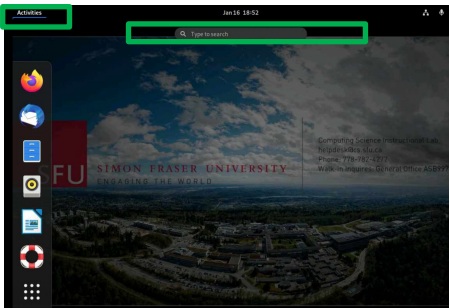
Login using sfu id and password

3.



Click on activities, then on search type terminal

4.



For more information on how to connect to csil lab computer please check :

<http://www.sfu.ca/computing/about/support/csil/unix.html#remote-access-linux-system>

For problems about it, contact helpdesk (<https://www.sfu.ca/computing/about/support/how-to-contact-helpdesk.html>)



After opening terminal lets try some simple and common linux commands:

- **pwd** : will give you the address of directory you are in.
- **ls** : use it to show the files and directories that are in the directory you are in.

For opening Terminal you also can use: **ctrl+alt+t**

```
sepidh@csil-cpu8:~$ pwd
/home/sepidh
sepidh@csil-cpu8:~$ ls
Android  Documents  Music      Public     Templates  'VirtualBox VMs'
Desktop  Downloads  Pictures   sfuhome    Videos
```



- **cd** : To navigate through the Linux directories and files. if you write **cd** and press **tab** it will show you possible directories

you can enter using **cd**, you can write part of file name to make the search space for that smaller.
- use **cd ..** to go backward.

```
sepidh@csil-cpu8:~$ ls
Android  Documents  Music      Public      Templates  'VirtualBox VMs'
Desktop  Downloads  Pictures    sfuhome     Videos

sepidh@csil-cpu8:~$ cd
Android/      Documents/    Music/        Templates/
.android/     Downloads/    Pictures/     .vagrant.d/
.cache/       .gnupg/      Public/       Videos/
.CLion/       .GoLand/     .PyCharm/    VirtualBox VMs/
.conda/       .gradle/     .RubyMine/   .vscode-server/
.config/      .IntelliJIDEA/ sfuhome/
Desktop/     .local/      .ssh/

sepidh@csil-cpu8:~$ cd D
Desktop/ Documents/ Downloads/
sepidh@csil-cpu8:~$ cd Desktop
sepidh@csil-cpu8:~/Desktop$ ls
sepidh@csil-cpu8:~/Desktop$ cd ..
sepidh@csil-cpu8:~$ ls
Android  Documents  Music      Public      Templates  'VirtualBox VMs'
Desktop  Downloads  Pictures    sfuhome     Videos
```



- **mkdir** : use this to create new directory
- **rm** : use these to remove files, and directories. For directories you need to use **rm -r**

```
sepidh@csil-cpu8:~$ ls
Android  Documents  Music      Public    Templates  'VirtualBox VMs'
Desktop  Downloads  Pictures   sfuhome   Videos
sepidh@csil-cpu8:~$ mkdir sepid
sepidh@csil-cpu8:~$ ls
Android  Documents  Music      Public    sfuhome     Videos
Desktop  Downloads  Pictures   sepid     Templates   'VirtualBox VMs'
sepidh@csil-cpu8:~$ rm -r sepid
sepidh@csil-cpu8:~$ ls
Android  Documents  Music      Public    Templates  'VirtualBox VMs'
Desktop  Downloads  Pictures   sfuhome   Videos
sepidh@csil-cpu8:~$
```



- *****: means anything. for example if you say `rm file*.txt` you mean remove all file“something”.txt

```
sepidh@csil-cpu8:~$ ls
Android  Downloads  file.txt   Public    Videos
Desktop  file2.txt  Music     sfuhome   'VirtualBox VMs'
Documents file3.txt  Pictures   Templates
sepidh@csil-cpu8:~$ rm file*.txt
sepidh@csil-cpu8:~$ ls
Android  Documents  Music     Public    Templates  'VirtualBox VMs'
Desktop  Downloads  Pictures   sfuhome   Videos
sepidh@csil-cpu8:~$
```



- **cp:** use cp for copying file and directories, for directories again you need to do cp -r. Same as other

command tab can give you possible options for being copied.

```
sepidh@csil-cpu8:~$ ls
Android  Documents  file.txt  Pictures  sfuhome  Videos
Desktop  Downloads  Music     Public    Templates 'VirtualBox VMs'
sepidh@csil-cpu8:~$ cp file.txt file2.txt
sepidh@csil-cpu8:~$ ls
Android  Downloads  Music     sfuhome  'VirtualBox VMs'
Desktop  file2.txt  Pictures   Templates
Documents file.txt   Public     Videos
sepidh@csil-cpu8:~$ cp -r Do
Documents/ Downloads/
sepidh@csil-cpu8:~$ cp -r Downloads Downloads_p
sepidh@csil-cpu8:~$ ls
Android  Downloads  file.txt  Public  Videos
Desktop  Downloads_p Music     sfuhome 'VirtualBox
Documents file2.txt  Pictures  Templates
sepidh@csil-cpu8:~$
```



- **mv:** use mv to move files and directories.

```
sepidh@csil-cpu8:~$ ls
Android    Downloads  file.txt   Public     Videos
Desktop    Downloads_p Music      sfuhome    'VirtualBox VMS'
Documents  file2.txt  Pictures   Templates
sepidh@csil-cpu8:~$ mv file.txt Downloads/
sepidh@csil-cpu8:~$ ls
Android    Downloads  Music      sfuhome    'VirtualBox VMS'
Desktop    Downloads_p Pictures    Templates
Documents  file2.txt  Public     Videos
sepidh@csil-cpu8:~$ cd Downloads
sepidh@csil-cpu8:~/Downloads$ ls
file.txt
sepidh@csil-cpu8:~/Downloads$ cd ..
sepidh@csil-cpu8:~$ mv Downloads_p Do
Documents/ Downloads/ Downloads_p/
sepidh@csil-cpu8:~$ mv Downloads_p Downloads
sepidh@csil-cpu8:~$ ls
Android    Documents  file2.txt  Pictures   sfuhome    Videos
Desktop    Downloads  Music      Public     Templates  'Virtual
sepidh@csil-cpu8:~$ cd Downloads/
sepidh@csil-cpu8:~/Downloads$ ls
Downloads_p  file.txt
sepidh@csil-cpu8:~/Downloads$
```



- **find:** use to find files and directories.

```
sepidh@csil-cpu8:~$ ls
Android      Downloads  Pictures   sfuhome    'VirtualBox VMs'
Desktop     file2.txt  Public     Templates
Documents   Music      sepid.txt  Videos
sepidh@csil-cpu8:~$ find sepid.txt
sepid.txt
sepidh@csil-cpu8:~$ find *.txt
file2.txt
sepid.txt
sepidh@csil-cpu8:~$
```



File: sepid.txt

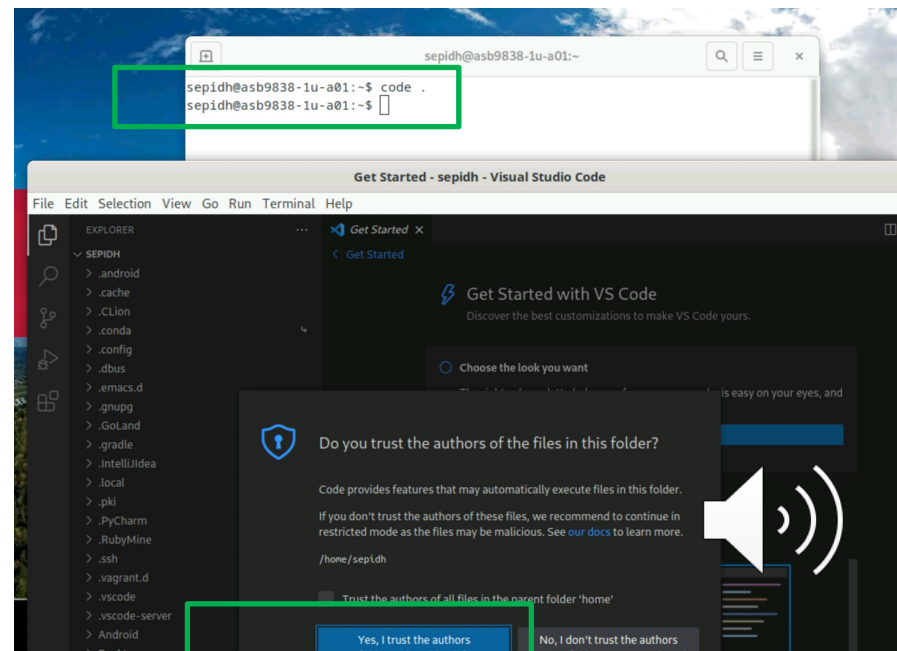


VSCode Configuration, and run “Hello Word”

For opening VS code there is two ways

- 1: Similar to terminal just search for it.
- 2: in terminal type down **code** .

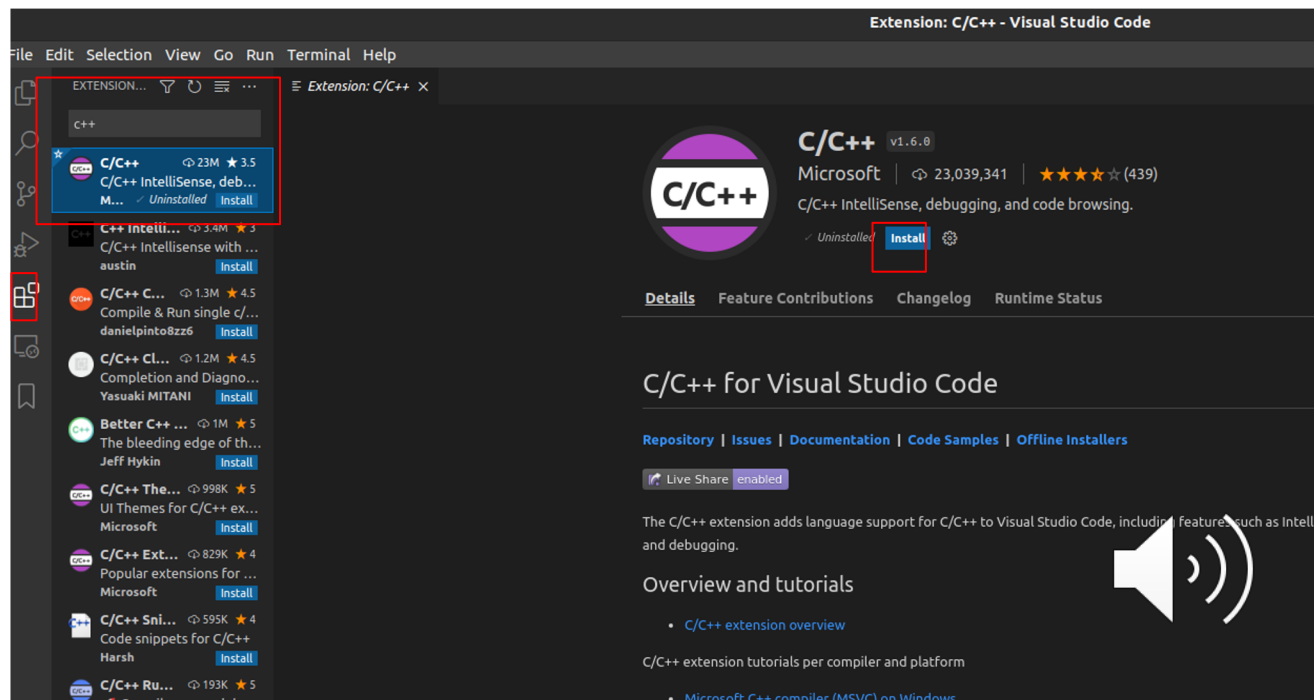
After opening it click on yes, I trust.



VSCode Configuration, and run "Hello Word"

Open VS code, go to extensions and install c/c++ extension

If it was already installed it is totally fine.



VSCode Configuration, and run “Hello Word”

Go to File, create new file. paste “**hello world**” program below in the file and save is as hello.c on your machine.

Don’t save in in desktop or in home, go to documents and create a folder with a proper name and save it there, this will save you from future headaches.

hello world:

```
#include <stdio.h>

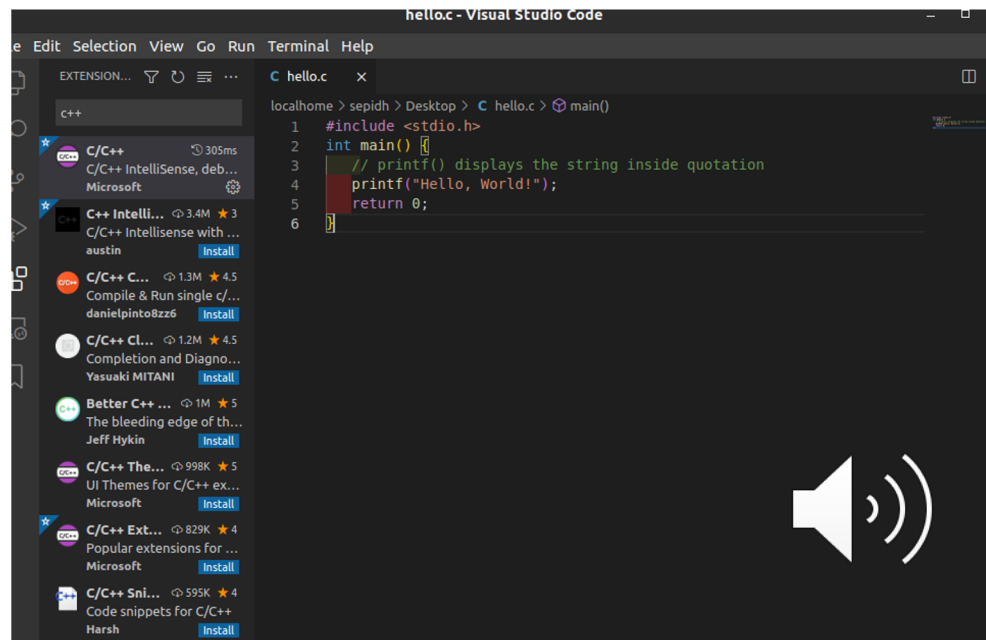
int main() {

    // printf() displays the string inside quotation

    printf("Hello, World!\n");

    return 0;

}
```



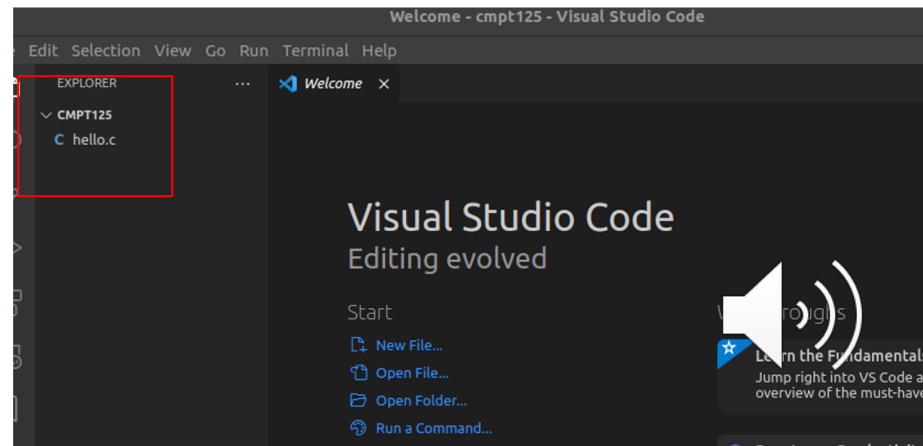
VSCode Configuration, and run “Hello Word”

Then go to the directory that file is saved there. right click and select open

terminal. in the terminal write code . and enter.

new vs code should open and your hello.c should be there. chose it.

```
sepidh@cs-vml-42:~/Desktop/cmpt125$ code .
```



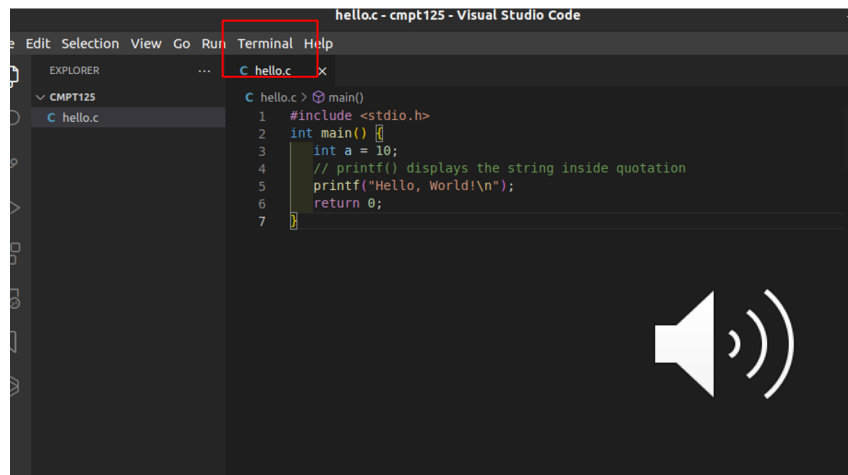
VSCode Configuration, and run “Hello Word”

from terminal choose “Run Build Task”.

then choose “C/C++: gcc build active file”

now your code is ready to run and debug using

“Run” in vs code.



VSCode Configuration

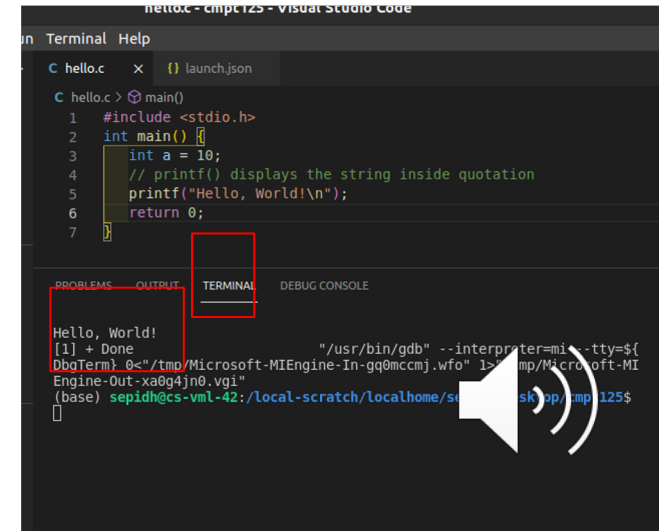
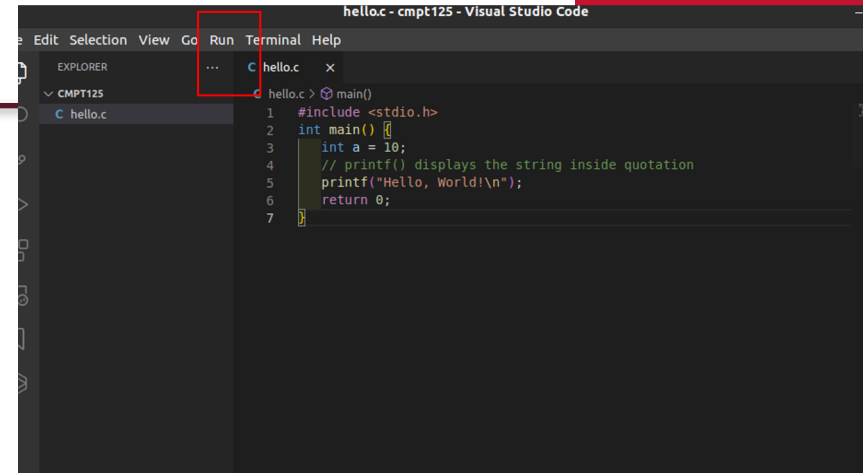
in “Run”, menu at top choose “Run Without Debugging”.

Then choose C++(GDB, LLDB), and

then choose “gcc build and debug active file”.

at terminal down you should be able to see output.

Change “Hello, World!\n” to any sentence you like and see
how printed output changes.



VSCode Configuration, and run “Hello V

To debug the code first click on the lines that you want your code stop on. you should see small red circles there after clicking.

Then from Run menu choose Start Debugging.

```
C hello.c (deleted) •
C hello.c
1  #include <stdio.h>
2  int main() {
3      printf("Start");
4      // printf() displays the string inside quotation
5      printf("Hello, World!\n");
6      return 0;
7  }
```

Run Terminal Help

C hello.c x

C hello.c > main()

```
1  #include <stdio.h>
2  int main() {
3      printf("Start");
4      // printf() displays the string inside quotation
5      printf("Hello, World!\n");
6      return 0;
7  }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

C/C++: ...

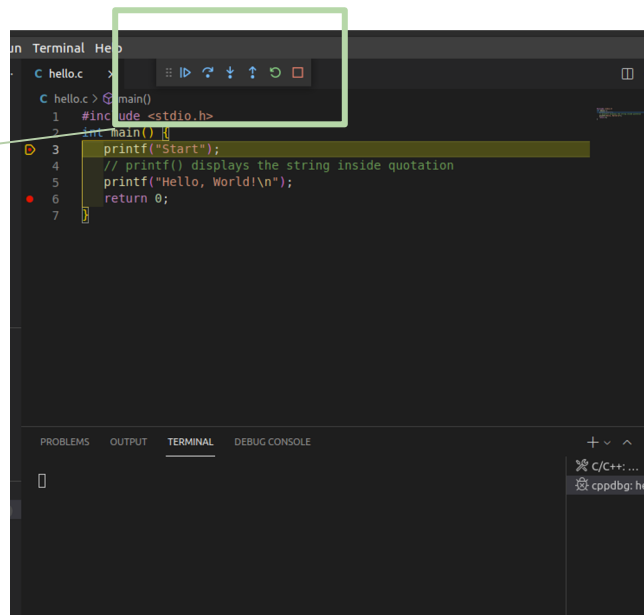
gdb: hello



VSCode Configuration, and run "Hello Word"

It should stop in the selected lines.

Then by using options at top you can navigate through your code and move inside your code.



Run “Hello Word” using terminal.

Other way to run your code is using terminal and gcc there.

open a terminal and go the the directory that your file is in.

Then, write this command in the terminal:

```
gcc hello.c -o hello.o
```

with above command you created hello.o file. then you can run

it using command `./hello.o` and see the output.

```
sepidh@cs-vml-42: ~/Desktop/cmpt125
(base) sepidh@cs-vml-42:~/Desktop/cmpt125$ gcc hello.c -o hello.o
(base) sepidh@cs-vml-42:~/Desktop/cmpt125$ ./hello.o
Hello, World!
(base) sepidh@cs-vml-42:~/Desktop/cmpt125$
```



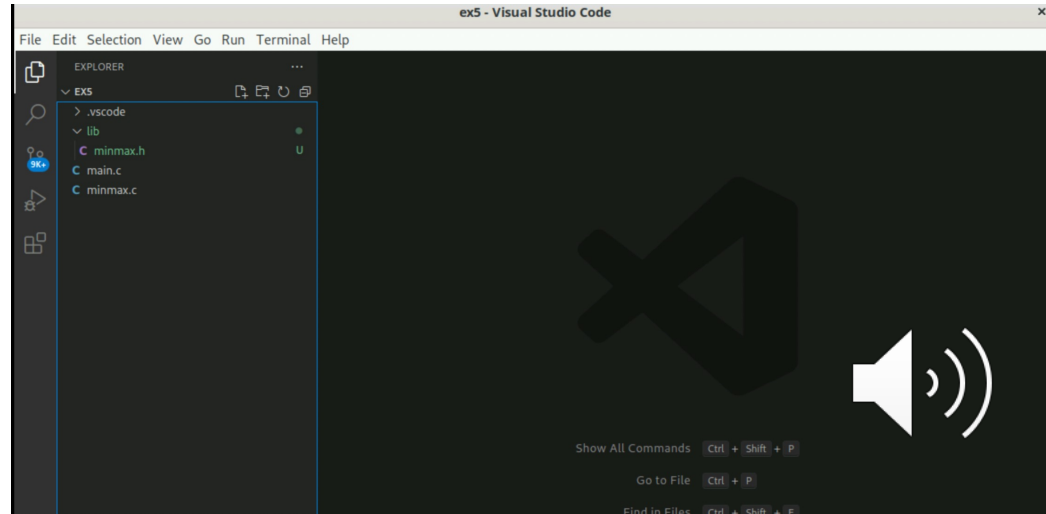
Practice examples:

- What if there are several .c and .h file in different directories?
 - Downloads minmax.zip file, and save it in a file. Open the terminal from the folder, then open vscode by typing **code .**

In terminal, something like this should show up.

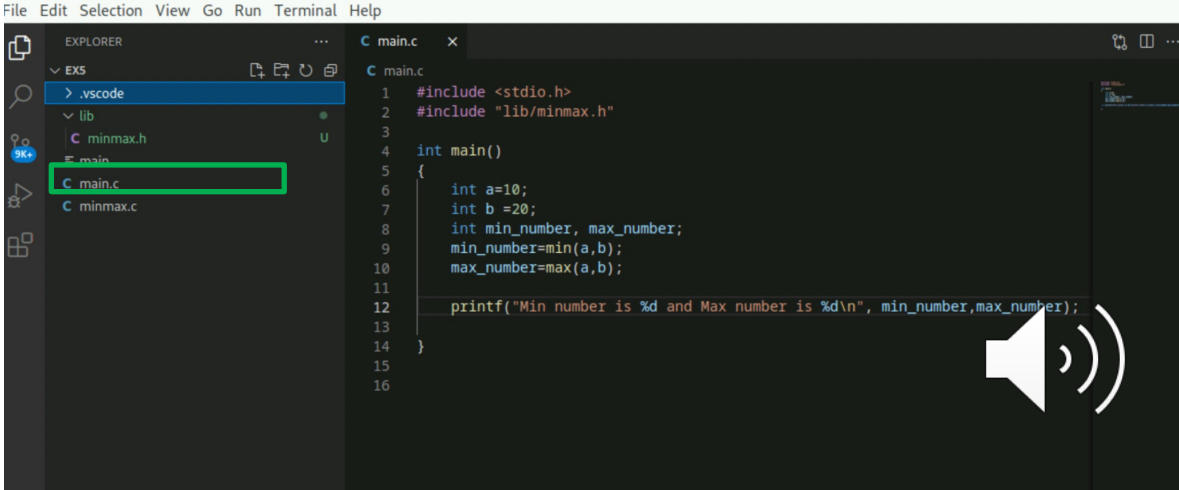
As you can see there is main.c and minmax.c and also a directory called lib with minmax.h inside.

For building main.c we need our compiler see all those files, so we need to follow a few steps.



Practice examples:

- First clean on main.c, as it opens in editor you can see that it includes lib/minmax.h, in the other word if we want that to compile we need compiler see minmax.h too



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays a file tree with the following structure:

- EXS
 - .vscode
 - lib
 - minmax.h
 - main
 - C main.c** (highlighted with a green rectangle)
 - minmax.c

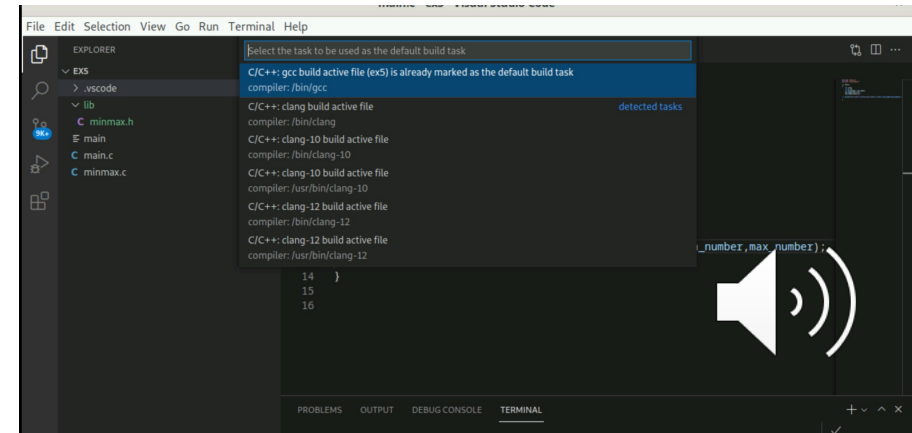
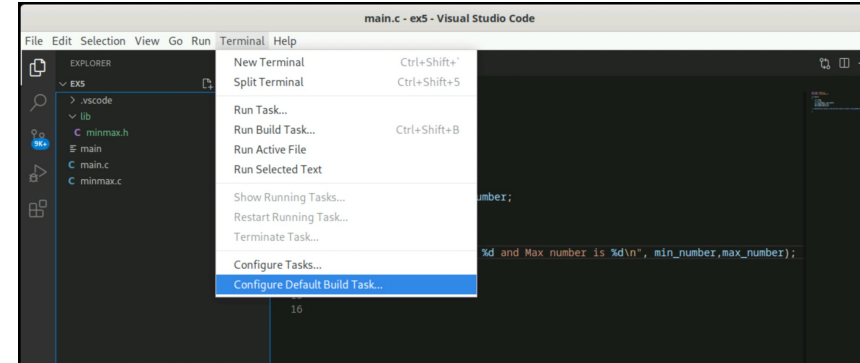
The main editor window shows the content of `C main.c`:

```
1 #include <stdio.h>
2 #include "lib/minmax.h"
3
4 int main()
5 {
6     int a=10;
7     int b =20;
8     int min_number, max_number;
9     min_number=min(a,b);
10    max_number=max(a,b);
11
12    printf("Min number is %d and Max number is %d\n", min_number,max_number);
13
14 }
15
16
```

A speaker icon is visible in the bottom right corner of the editor area.

Practice examples:

- Click on terminal, then open “Configure Default Build Task”,
- Then choose the one with **c/c++: gcc** in it.



Practice examples:

So what comes after “-g” is what compiler is going to see.

You need to put all of your input files after that -g.

By default it is `${file}`, which means it only see the open file “main.c” by default.

You need to edit it and write down all filenames that compiler need to see.

To do see in each line you start by “`${fileDirname}/`” this is the folder you are working inside and your main.c has been saved in.

Then you add path to each file.

For example as my minmax.h is in lib directory under, I wrote “`${fileDirname}lib/minmax.h`”.

After doing that you can build your code as previous examples.

```
tasks: [
  {
    "type": "cppbuild",
    "label": "C/C++: gcc build active file",
    "command": "/bin/gcc",
    "args": [
      "-fdiagnostics-color=always",
      "-g",
      "${file}",
      "-o",
      "${fileDirname}/${fileBasenameNoExtension}"
    ],
    "options": {
      "cwd": "${fileDirname}"
    },
    "problemMatcher": [
      "$gcc"
    ],
    "group": {

```

```
main.c tasks.json x
tasks.json > [ ] tasks > { } 0 > [ ] args
4 {
5   "type": "cppbuild",
6   "label": "C/C++: gcc build active file",
7   "command": "/bin/gcc",
8   "args": [
9     "-fdiagnostics-color=always",
10    "-g",
11    "${fileDirname}/lib/minmax.h",
12    "${fileDirname}/main.c",
13    "${fileDirname}/minmax.c",
14    "-o",
15    "${fileDirname}/${fileBasenameNoExtension}"
16  ],
17  "options": {
18    "cwd": "${fileDirname}"
19  },
20  "problemMatcher": [
21    "$gcc"
22  ],
23  "group": {
24    "kind": "build",
25    "isDefault": true

```



What is .h file (header files):

- Header file included at the top of any C program. as `#include <file>` and they end with “.h”



Some useful system header files:

- **<stdio.h>** : have many standard library functions for file input and output
- **<math.h>** : support are mathematical related functions in c
- **<time.h>**: interact with system time
- **<string.h>**: support string handling function

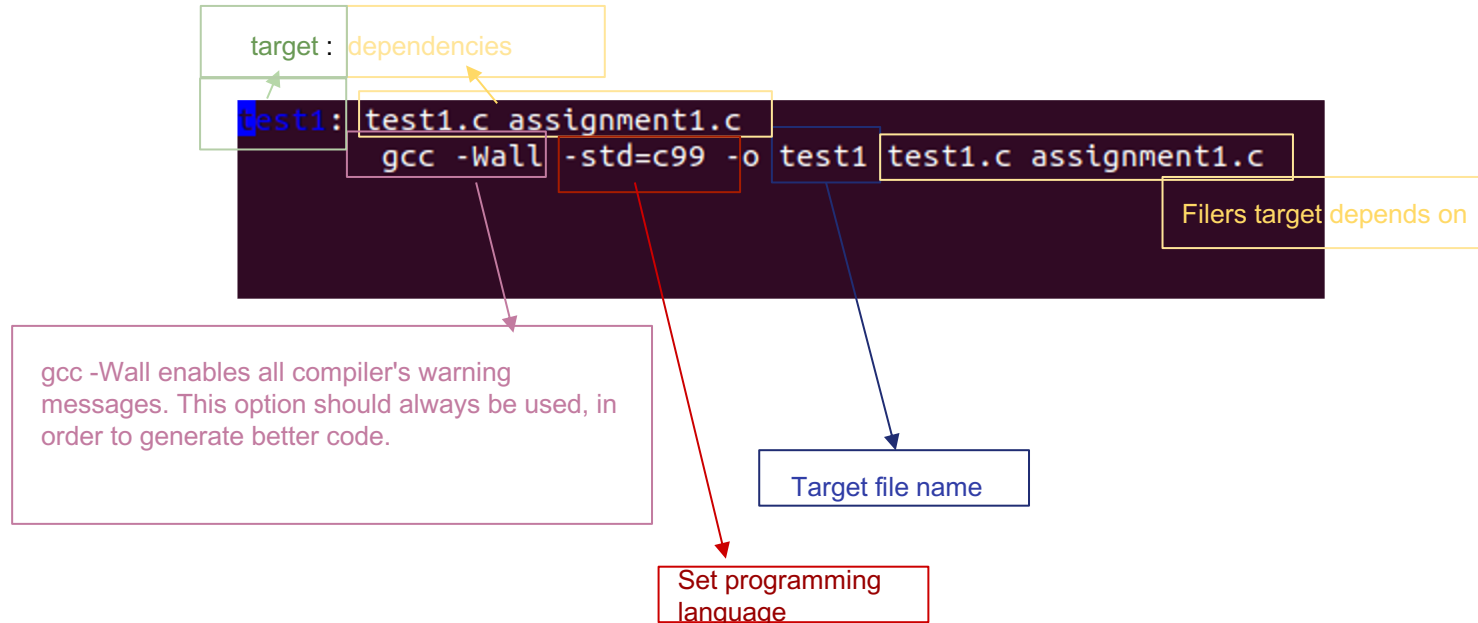


What is in makefile:

- **rules : implicit, explicit**
- **variables (macros)**
- **directives (conditionals)**
- **# sign – comments everything till the end of the line**
- **\ sign - to separate one command line on two rows**



Simple makefile



Why do we use .h file?

- When dealing with larger projects, it's advisable to divide your code into separate files based on their purpose (function sets)
- You can use these self-made libraries to quickly add desired functionality to your code typically used files

assignment1.h – header file, typically contains function prototypes, macros, [classes,] structures

assignment1.c – Code that implements the functions declared in the header file

To run and check your assignment.

you need to first: change and implement your functions/methods in **assignment1.c**.

then using terminal go to the directory that your assignment has been saved.

use command **make** to make executable **test1**.

Then use command **./test1** to run test1 and check test case.

```
(base) sepidh@cs-vml-42:~/Desktop/cmpt125/hw1-cmpt125-fall21$ make
gcc -Wall -std=c99 -o test1 test1.c assignment1.c
(base) sepidh@cs-vml-42:~/Desktop/cmpt125/hw1-cmpt125-fall21$ ./test1
01-1 ok
01-2 ok
02-1 ok
02-2 ok
03-1 ok
03-2 ok
04-1 ok
04-2 ok
05-1 ok
05-2 ok
(base) sepidh@cs-vml-42:~/Desktop/cmpt125/hw1-cmpt125-fall21$
```



For Assignment 1:

- You need fill assignment1.c (parts that said `// implement me`), do not touch assignment1.h
- your code must be compile in **csil** systems using make file.
- Do not write too much explanation for your code, please have a clean readable code.
- Use comments, not too much but enough
- **Only submit assignment1.c**, if you submit something with wrong name our grading system cannot compile that which means you will receive 0.
- **Do not put int main function in assignment1.c**, it will cause compile error which mean 0!
- **If your code does not compile you will automatically receive 0. if you got 0 because of compile error you have a chance to resubmit until one week after a grade release but you will receive only 50% of your grade.**



Practice examples:

- Implement a program that get array of integers and print the max:

```
#include <stdio.h>
int main()
{
    int array[10];

    // getting array element using scanf

    for ( int c = 0; c < 10; c++)
    {printf("enter a number: ");
      scanf("%d", &array[c]); }
    int max_temp=0;

    //*****//
    // write down a for loop that find the max value here
    //*****//

    printf("Maximum element is %d.\n", max_temp);
    return 0;
}
```



Practice examples:

- Answer:

```
#include <stdio.h>
int main()
{
    int array[10];

    // getting array element using scanf

    for ( int c = 0; c < 10; c++)
        {printf("enter a number: ");
        scanf("%d", &array[c]); }
    int max_temp=0;

    //*****
    for (int c = 0; c < 10 ; c++)
        if (array[c] > max_temp)
            max_temp = array[c];
    //*****

    printf("Maximum element is %d.\n", max_temp);
    return 0;
}
```



Practice examples:

- Implement a program that get an array of ints and increase each value by 1 and print the array:

```
#include <stdio.h>
int main()
{
    int array[10];
    // getting array element using scanf
    for ( int c = 0; c < 10; c++)
        {printf("enter a number: ");
        scanf("%d", &array[c]); }

    //*****
    // implement a for loop that add 1 to each entry in array
    //*****

    // printing out array
    for (int c = 0; c < 10 ; c++)
        printf("%d ",array[c]);
    return 0;
}
```



- Answer:

```
#include <stdio.h>
int main()
{
    int array[10];
    // getting array element using scanf
    for ( int c = 0; c < 10; c++)
    {printf("enter a number: ");
      scanf("%d", &array[c]); }

    //*****
    for (int c = 0; c < 10 ; c++)
        array[c]++;
    //*****

    // printing out array
    for (int c = 0; c < 10 ; c++)
        printf("%d ",array[c]);
    return 0;
}
```



- Implement swap of two numbers (using pointers)

```
#include <stdio.h>

int main()
{

    int x, y, *a, *b, tmp;

    printf("Enter the value for x:\n");
    scanf("%d", &x) ;

    printf("Enter the value for y:\n");
    scanf("%d", &y);

    //*****
    // implement the part that swap two numbers using pointer
    //*****

    printf("After: \n \"x\" = %d\n \"y\" = %d : \n", x, y);

    return 0;
}
```



- Answer

```
#include <stdio.h>
int main()
{
    int x, y, *a, *b, tmp;
    printf("Enter the value for x:\n");
    scanf("%d", &x) ;
    printf("Enter the value for y:\n");
    scanf("%d", &y);

    //*****
    a = &x;
    b = &y;
    tmp = *b;
    *b = *a;
    *a = tmp;

    //*****
    printf("After: \n \"x\" = %d\n \"y\" = %d :\n", x, y);
    return 0;
}
```



- Compute length of a string

```
#include <stdio.h>
int main() {

    char* s = "Programming is FUN!!";
    int i=0;

    //*****
    // while loop that change i value as the length of
    //*****

    printf("Length of the string: %d", i);
    return 0;
}
```



- Answer

```
#include <stdio.h>
int main() {

    char* s = "Programming is FUN!";
    int i=0;

    //*****
    while(s[i] != '\0')
        i++;
    //*****

    printf("Length of the string: %d", i);
    return 0;
}
```

