This course has already ended.
The latest instance of the course can be found at: Object oriented programming with C++: 2023 Autumn

« Getting Started Course materials 2 Exercises »

1 Development environment¶

Contents

The recommended and supported code editor for the course is Visual Studio Code. It is free, versatile and open-source editor that works with many different languages through extensions. The exercises contain configurations for compiling and debugging with it, and it comes with git integration, which you can use for the exercises and the project.

Important

It is also possible to use some other editor or IDE through makefiles if you prefer to do so. However, if you have any issues with it, the course staff may not be able to help you.

1.1 Installing Visual Studio Code and tools¶

Please follow the instructions below to install Visual Studio Code and compiler tools for your local computer. **You only need to follow one of the instructions.**

1. Install Visual Studio Code on your local machine.

   Hint

   When prompted to **Select Additional Tasks** during installation, make sure to check the **Add to PATH** option so you can easily open a folder using `code` command.

2. Open the `Extensions` window by following any one of the following methods.
   1. Press `CTRL` + `⇧` + `P` (`⇧` + `⌘` + `P` in MacOS) and write `extensions`
   2. Press `CTRL` + `⇧` + `X` (`⇧` + `⌘` + `X` in MacOS)
   3. Look for `Extensions` on the (Left) side bar.
3. Search for `c++` by typing into the search box.
4. Install the Microsoft C/C++ extension for Visual Studio Code.
5. You will need a compiler to build your applications. The available compilers changes between platforms, and enabling/installing them require some effort.
   1. If you want to avoid installing compilers on your local computer, click **Remote connection** tab below.

      Caution

      Remote connection might have intermittent connection problems and require frequent reconnection.

   2. It is more instructive/useful to install the compiler tools on your local machine. Click on the tab of your operating system and follow the instructions to install/enable the build tools.

      Important

      There are two options listed for Windows Operating system. You only need one of them.

      If you are using Windows 10 computer, we recommend installing `gcc` **under** `Windows Subsystem for Linux (WSL)`.

      Windows 10 (WSL)

      1. Install Windows Subsystem for Linux according to instructions here.
         - Open PowerShell as Administrator and run:

           ```
           dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart
           ```

         - Restart your computer when/if you are prompted.
      2. Install Ubuntu 20.04 LTS for it.
         - Download Ubuntu 20.04 LTS from Microsoft Store and install it
         - Restart your computer when/if you are prompted.
      3. After installing, launch it by clicking `Ubuntu` icon or using the start menu item for `Ubuntu 20.04 LTS`
         - When prompted, enter a new UNIX username

           Caution

           This will be your username inside Ubuntu, and you may need it to perform some tasks.

         - When prompted, type `New password` and `Retype new password`

           Caution

           This is admin password for Ubuntu, and you will need it quite often. Select something that you will not forget.

         - Install compiler tools for WSL. In this course, we will need the `build-essential`, `git`, `valgrind` and `gdb` packages, but also `wget` and `unzip` might be handy. Run the following commands:

           ```
           sudo apt update
           sudo apt install build-essential valgrind git gdb wget unzip
           ```

         - Now you should have your Ubuntu 20.04 LTS configured!
         - To access Windows files from the Ubuntu terminal, go to folder `/mnt/`, which includes all your system drives mounted, for example `c` for your `C:` drive.
         - To access your files under WSL using windows explorer, type

           ```
           \\wsl$\<distro-name>\home\<username>
           ```

           in address bar of file explorer, after replacing `<distro-name>` with the name of your WSL distribution, for example `Ubuntu`, and `<username>` with the username you selected earlier if you are not sure about the WSL distribution name, just write

           ```
           \\wsl$
           ```

           in the address bar. This should show you all installed WSL distributions. Then, double click on `<distro-name>`, then `home` and then `<username>` you entered earlier.
      4. Install Remote Development extension pack to VS Code.
      5. Open a WSL terminal window using the start menu item for `Ubuntu 20.04 LTS`.
         - Navigate to a folder you'd like to open in VS Code. This would be the directory you would store your exercise files.
         - Install the C/C++ extension to VS Code in WSL, which should automatically popup on the `Extensions` window.
      6. All done! Now you can move onto the next section about doing the exercises.

      Windows (MinGW)

      1. Download and install Minge-w64 from here.
         - Download from the link below `MinGW-w64 Online Installer`
         - Make the following choices during the install:
           - MinGW options
         - You may choose the install location yourself. These instructions assume the default location: `C:\Program Files\mingw-w64\x86_64-8.1.0-posix-seh-rt_v6-rev0`
      2. Add the `bin` folder which is inside the `mingw64` folder inside the install location to the `PATH` environmental variable:
         - Search for `Edit environment variables` in Windows and select `Edit environment variables for your account`
         - Select the `Path` variable and press `Edit`
         - Select `New` and add the address of the `bin` folder. With the above location it would be `C:\Program Files\mingw-w64\x86_64-8.1.0-posix-seh-rt_v6-rev0\mingw64\bin`
         - Press `Ok`
         - If you have a console or Visual Studio Code open, you'll need to close and reopen them for the changes to take effect.
      3. All done! Now you can move onto the next section about doing the exercises.

      MacOS

      For MacOS, we are going to use `clang` compiler, which might might be already be installed on your Mac.

      To verify that it is, open a macOS Terminal window

      - In the `Finder`, open the `/Applications/Utilities` folder, then double-click `Terminal`

      and then, enter the following command:

      ```
      clang --version
      ```

      If Clang isn't installed, enter the following command to install the command line developer tools:

      ```
      xcode-select --install
      ```

      Linux

      Install the `build-essential` and `gdb` packages, and also the tools you will need `valgrind`, `git`, `wget`, and `unzip`.

      Hint

      These are already installed on the Aalto computers. If you are using your own Linux computer, install these packages.

      If you are using `Ubuntu`, run the following commands:

      ```
      sudo apt update
      sudo apt install build-essential gdb valgrind git unzip wget
      ```

      For other distributions, run the corresponding commands

      Remote connection

      1. Install Remote Development extension pack to VS Code.
      2. Press the double arrow icon in the lower left corner and select `Remote-SSH: Connect to Host...`
      3. Select `+ Add New SSH Host...` and write

         ```
         ssh username@lyta.aalto.fi
         ```

         where `<username>` is your Aalto username. For example, if your username is `student1`, write `ssh student1@lyta.aalto.fi`.

         Alternative Aalto server

         If connection to `lyta.aalto.fi` does not seem to work, you can try `kosh.aalto.fi` instead of `lyta.aalto.fi`. Follow the same instructions by replacing `lyta.aalto.fi` with `kosh.aalto.fi`.

      4. Select an SSH configuration to update (usually the first one)
      5. A popup appears in the lower right corner, from where you can use the `Connect` button to connect to the school server.

         Later, the server can be connected by pressing `Connect` button in the lower left corner.

      6. Choose `Linux`, when you are asked about the platform of the remote host.
      7. Choose `Continue` when you are asked `Are you sure you want to continue?`
      8. Write your Aalto password to the field that opens.
      9. Install the C/C++ extension to VS Code in `lyta` (or `kosh`), which should automatically popup on the `Extensions` window.
      10. Now you're ready to do the exercises: You can download the *zip* containing the exercises templates by downloading the *.zip* file from the link at the top of the module sections.

      In the following video, you can watch how to install and configure Visual Studio Code for C/C++ programming on remote hosts.

1.2 Configuring Visual Studio Code¶

After installing the the tools, you are ready to configure Visual Studio Code. There are two ways to achieve this.

1. Using Graphical User Interface (GUI) of File Manager provided by your Operating System.

   Error

   You cannot use this option if you have selected and prepared for `Remote connection` option above.

   - For Windows, file manager is `File Explorer` or `Explorer`.
   - For MacOS, default file manager is `Finder`
   - For Linux, the default file manager depends on the distribution: `Nautilus`, `Konqueror` etc.
2. Using Command Line Terminal of your Operating System.

   Caution

   If you have Windows computer, and you are not using WSL, this option is not recommended.

   Command-line terminal can be accessed differently for different operating systems.

   - For WSL, click `Ubuntu` icon on your desktop or find `Ubuntu 20.04 LTS` in your start menu and click it.
   - For MacOS, in the `Finder`, open the `/Applications/Utilities` folder, then double-click `Terminal`.
   - For Linux, find terminal of your distribution, and click it.

   We recommend you to familiarize yourself with *command line tools* as these are handy to solve some problems you might face.

   Command Line

   1. Navigate to the directory you have write access rights and you want place the folder for the course development.
      - The terminal emulator usually starts in your `home` folder, which you have the rights. If you want to change it, use `cd` (*change directory*) command.
   2. Create a new directory using `mkdir` command. This command requires you to specify a directory name. We recommend `cpp-autumn2021`.

      ```
      mkdir cpp-autumn2021
      ```

   3. Change to the new directory using `cd` command.

      ```
      cd cpp-autumn2021
      ```

   4. Start Visual Studio Code in the directory.

      ```
      code .
      ```

      When doing this for the first time, you should see Visual Studio Code fetching components needed to run in WSL. This should only take a short while, and is only needed once.

      Error

      If this command does not work, you may need to restart your terminal or you might have forgotten to add Visual Studio Code to your `path` when installing it.

      After a moment, a new Visual Studio Code window will appear. It will now continue to configure itself and keep you informed as it makes progress.

      Note

      For WSL, once this process finishes, you can see a WSL indicator in the bottom left corner, and you'll be able to use Visual Studio Code normally.

   5. Next time you start Visual Studio Code, you can open this folder by following `File` → `Open Recent`, and selecting the directory you just created.

   Graphical User Interface

   1. Start File Manager of your Operating System.
   2. Navigate to a directory you have write access rights and you want place the course developments.
   3. Create a new directory with a name resembling the course. We recommend `cpp-autumn2021`.
   4. Start Visual Studio Code by clicking its icon
   5. In Visual Studio Code window, click `File` → `Open Folder...`
   6. Navigate to the folder you just created, and select it by pressing `Select Folder`.

Hint

Visual Studio Code provides several tools and you can find many *Extensions* for different purposes.

1. You can use integrated Terminal. If the bottom panel does not already show the terminal, you can create a new Terminal view by either one of the following options.
   - Click `Terminal` → `New Terminal` in the menu bar.
   - Use `CTRL` + `⇧` + `` ` `` (`⇧` + `⌘` + `` ` `` in MacOS) key combination.

   The terminal windows starts in the current workspace directory so that you do not need to navigate to the module directory.

2. (Left) Side Bar shows at least
   - `Explorer`
   - `Run and Debug`
   - `Source Control` (Git)
   - `Extensions`

   options. Try to familiarize yourself with these tools.
3. All commands can be entered using Command Palette. In order to activate, use either one of the following options.
   - Click `View` → `Command Palette` in the menu bar.
   - Use `CTRL` + `⇧` + `P` (`⇧` + `⌘` + `P` in MacOS) key combination.

« Getting Started Course materials 2 Exercises »