

C++ Programming Course: Assignment 0

Requirement to take part in this course: Complete the Task 0 (Getting started) and Task 3 (GitLab/GIT setup) until 08.11.23 at 11:59 pm


Introduction

The tasks given here were devised to run in either the provided virtual machine(s) or an appropriately configured *unix* machine. Conduct the lecture slides for a specification of the technical setup. If you use private-owned laptops you need to adjust the setup accordingly. We cannot and will not support any other operating system.

Task 0: Getting started

Carefully read and understand our sheet containing **General remarks** along with important rules and hints for participating in this course. The sheet is available from the materials page in the CMS.

Task 1: Some shell commands

1. open a terminal
2. to see the path where you are at the moment type: `pwd`
3. create an empty file: `touch test.txt`
4. edit file (e.g. using vim) `vim test.txt`;
 - enter insertion mode by pressing `i`
 - write something in your file
 - leave insertion mode by using `Esc`
 - write file to disk and quit by entering `:wq` 
5. view content of file: `less test.txt`; use `q` to leave less
6. output content of file into shell: `cat test.txt`
7. copy file: `cp test.txt test2.txt`
8. move file: `mv test2.txt test3.txt`
9. use `ls` or `ls -alrth` to see the files in the current folder

10. create folder: `mkdir new_folder`
11. move file into new folder: `mv test3.txt new_folder/`
12. change into new folder: `cd new_folder`
13. look at content `ls -al`
14. return to parent folder `cd ..`
15. delete file `rm test.txt`
16. delete folder that is not empty (be careful when using this command!): `rm -r new_folder`

Task 2: Using SSH

You can use the CIP-Pool machines via ssh.

CIP-Pool:

1. connect a the CIP-Pool machine:
`ssh your_SIC_login@bio01.studcs.uni-sb.de`
with the 01 being replacable by 01 - 24.
2. log out again: `logout`
3. generate ssh-key for connecting without password: `ssh-keygen -t rsa`
4. when asked for passphrase leave this empty
5. you should now have a file with your private key (`id_rsa`) and public key (`id_rsa.pub`) in `~/.ssh/` ; note: "`~`" is a shortcut for your home-folder (`/home/your_login/`); folders/files starting with a "." are not visible with the normal `ls` command, use `ls -al` to see these files; the public key can be used to connect with other computers over ssh without using a password; to this end your public key must be copied to the remote you want to connect to
6. copy public ssh key:
`ssh-copy-id -i ~/.ssh/id_rsa.pub your_SIC_login@bio01.studcs.uni-sb.de`
7. after that you should be able to connect with
`ssh your_SIC_login@bio01.studcs.uni-sb.de`
without entering a password
8. we will also use the created public key for gitlab

Task 3: GitLab/GIT setup

1. go to: <https://ccb-gitlab-stud.cs.uni-saarland.de/>

2. login with our university username and the password you received (After registering in CMS a personalized GitLab account is going to be created; Please be patient as this can take several hours to complete)
3. follow the slides of the intro lecture to add your ssh key
4. look up the git commands for cloning a repository, adding files, committing files, pulling files, pushing files (e.g. <https://git-scm.com/book/en/v2>)
5. clone the git repository into your home folder using the terminal
6. add a .gitignore file to the freshly cloned repository which should at least contain patterns for ignoring backup files (*~) and object files (*.o)
7. add, commit and push the .gitignore file
8. see the changes of your repository on the gitlab website
9. create a folder `assignment0` in your cloned repository
10. download the file *Assignment_0_Material.zip* from our CMS
11. unzip the file using the terminal into the `assignment0` folder
12. add everything contained in this folder to your git repository, commit it and push it
13. take a look at the gitlab website if the checks for this task will pass; if not try to find out what went wrong

Task 4: Setup development environment

1. get a students license at <https://www.jetbrains.com/shop/eform/students> and register an account
2. download CLion (<https://download-cf.jetbrains.com/cpp/CLion-2021.2.3.tar.gz>)
3. go to the download location and uncompress the file: `tar xzvf CLion-2021.2.3.tar.gz`
4. `cd clion-2021.2.3/bin` and execute clion script with `./clion.sh`
5. login with your students account created in the first step.
6. File -> New Project -> C++ Executable: enter a location and use C++17 ; this creates an automatic "Hello World" example for you including CMakeLists.txt for running cmake
7. build the example by clicking the "build" button (upper right) or via Run -> Build and run the example by clicking the "run" button

8. take a look at your project folder; the created Hello World file is named `main.cpp`. The build folder (`cmake-build-debug`) contains a Makefile, the executable, and some other files and folders. You can run the executable also with `./name_of_executable` if you are in the same folder
9. now delete the build folder (`rm -r cmake-build-debug`) and we try to create the same output that CLion gave us by clicking two buttons manually
10. create a folder `build` and change into this directory
11. call `cmake ..` to create a Makefile for your project
12. call `make` to build your project/executable
13. you should now have again an executable that you can execute :)
14. This was to demonstrate what an IDE is doing automatically for you. Nevertheless, you should still be able to build programs "manually" by using `cmake` and `make`. How the compiling and linking is completely manually done without using Makefiles is shown in the next task. Of course, this only makes sense for very small projects consisting of only a few files and having not many dependencies. The difference of compiling and linking in C++ is however important and should be clear (this will be re-explained in the lecture). Otherwise, it can happen that you search for errors at the completely wrong place in your programs.

Task 5: Compiling/Linking

1. The above "Hello World" example can also be easily compiled/linked manually, because it consists only of one C++ file.
2. To compile "main.cpp" manually, type: `g++ -c main.cpp`. This creates the object file `main.o`.
3. To get an executable the object file must be linked (with other libraries or other object files). To do that for our example, write: `g++ main.o -o main` to get an executable named `main`
4. For this simple case compiling/linking can also be done in one command:
`g++ main.cpp -o main`

Task 6: Debug code snippets

1. Unzip the file `handson_code.zip`.
2. Create a new project in CLion by opening the `CMakeLists.txt` file.

3. Start debugging the 4 code snippets.
 - Check if the code compiles and/or links
 - If you run the code, does it produce an output as expected?
 - Try to find programming errors by inspecting the code and running it on custom input that confirms your ideas.