

### 1 *Hello, World!* in C

#### New Virtual Machine for the Summer Semester

In the context of this course, we will use an updated version of the Ubuntu VM from last semester. The newer version can be found on the C:\ drive of the workstations in the lab rooms. Add this VM to the VirtualBox Manager by clicking on the corresponding .vbox file (the one with the blue icon).

To use the VM on your own machines, please find the download link on the Moodle page of this course.

#### Exercise 1 – "Hello, World!"

Write a C program printing "Hello World!" to standard output, as seen in the lecture.

- Boot the VM image provided for this course.
- Use the terminal to create, compile, and run the program.
- In this exercise, do **not** use Eclipse or any other IDE! Instead, use **nano** and **gcc**.

#### Exercise 2 – "Hello, World!" – Reloaded

#### Eclipse CDT (C/C++ Development Tooling)

From this exercise on, you may use Eclipse to create, compile and run your C programs. To do so, you need to have Eclipse CDT (C/C++ Development Tooling) installed. The Eclipse installation within the VM image already comes with this plugin installed. For your own computer, you may either download CDT as a standalone Eclipse version or as a plugin into an already existing Eclipse installation.

In PROGRAMMING 1, we were using the Java *perspective* of Eclipse. In PROGRAMMING 2, we will now be using the C perspective. To open it, select **Window** » **Perspective** » **Open Perspective** » **Other ...** » **C/C++**

To create a new C project, choose **File** » **New** » **C/C++ Project** and select **C Managed Build**. Give your project a name and click **Finish**.

To start coding, add a new source file `main.c` by choosing **File** » **New** » **Source File**.

To be able to run a C program, you first need to build its binaries. To do so, select **Project** » **Build All**.

Write again a C program printing "Hello World!" to standard output, but this time in Eclipse.

- After building the binaries, run the program.
- Also try out the debugger by putting a breakpoint before the `printf` call.

## 2 Introduction to Git

### 2.1 Motivation

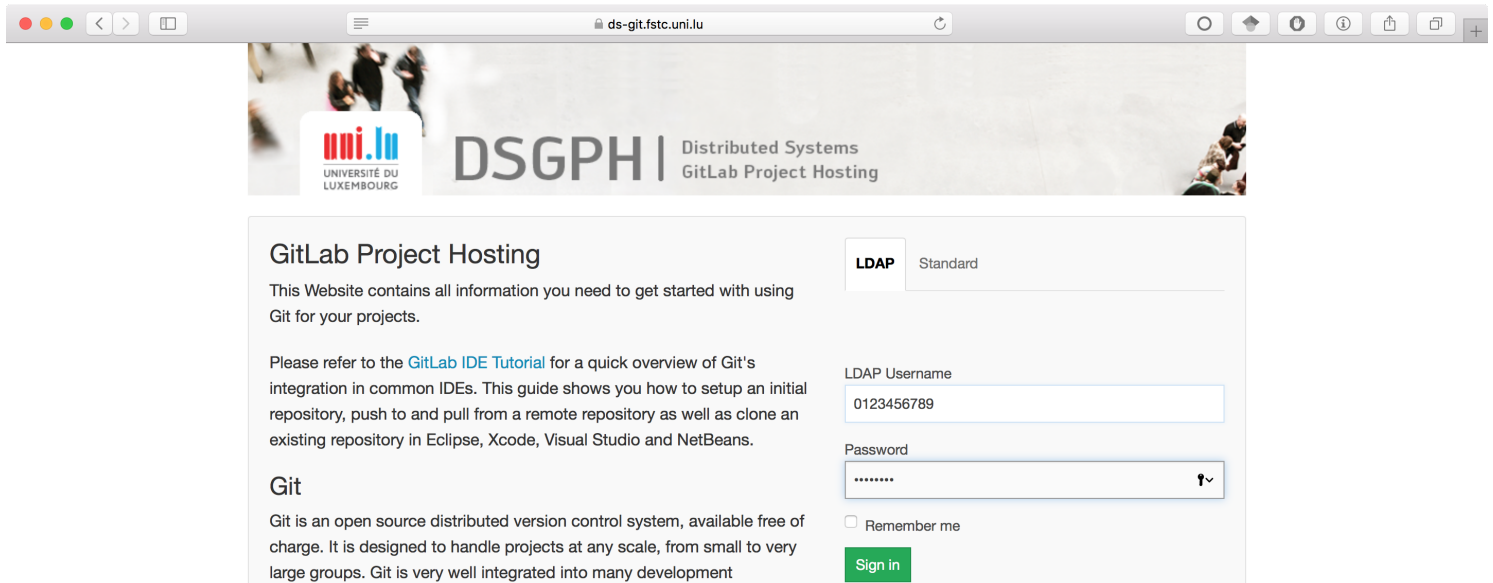
The code examples discussed in the lecture as well as the solutions of the practical labs will be put on DS-GIT, our GitLab installation. GitLab is a web-based repository manager for the Git Version Control System. We will later in the course benefit from this platform (and Git itself) to enable collaboration between team members in your implementation project.

In the meanwhile, the platform will allow us to extend and continue the discussion on code snippets outside the regular lecture and lab slots. The idea is to provide you with a tool where you can comment code snippets, ask questions on what you have not understood so far, discuss alternative approaches and help each other on the journey of learning a programming language. Please feel free to contribute with your questions and discussions, which we will try to answer in a timely and clear way. Also, do not hesitate to engage in a discussion and answer questions of your fellow students.

The purpose of this document is to show you step by step how to use the GitLab platform to browse through the code snippets, ask questions and get notified on new activity.

### 2.2 Account Creation & Project Membership

On <https://ds-git.fstc.uni.lu>, perform a login with your usual Uni.lu credentials, as shown in figure 1. At your first login, you will not yet adhere to any repository, as the membership is managed by the administrators. We will add you to the lecture's repository in a timely manner, based on the users both registered in the Moodle course and on the GitLab platform. If, by mistake, you are not added to the repository within a few days, please send an email to [christian.grevisse@uni.lu](mailto:christian.grevisse@uni.lu). Note that the first login is required for us to be able to add you to a repository, as the user account is created at your first login based on your Uni.lu account.



ds-git.fstc.uni.lu

**GitLab Project Hosting**

This Website contains all information you need to get started with using Git for your projects.

Please refer to the [GitLab IDE Tutorial](#) for a quick overview of Git's integration in common IDEs. This guide shows you how to setup an initial repository, push to and pull from a remote repository as well as clone an existing repository in Eclipse, Xcode, Visual Studio and NetBeans.

**Git**

Git is an open source distributed version control system, available free of charge. It is designed to handle projects at any scale, from small to very large groups. Git is very well integrated into many development

**LDAP** Standard

LDAP Username  
0123456789

Password  
.....

☐ Remember me

**Sign in**

Figure 1 – Login on <https://ds-git.fstc.uni.lu>

## 2.3 Platform Use

### 2.3.1 Browsing through projects and files

Once access has been granted to the repository, you will receive an email to your student account. On the platform, you shall now be able to see the course repository, as shown in figure 2.



Figure 2 – Overview of projects

When clicking on the project, you will see the screen shown in figure 3.

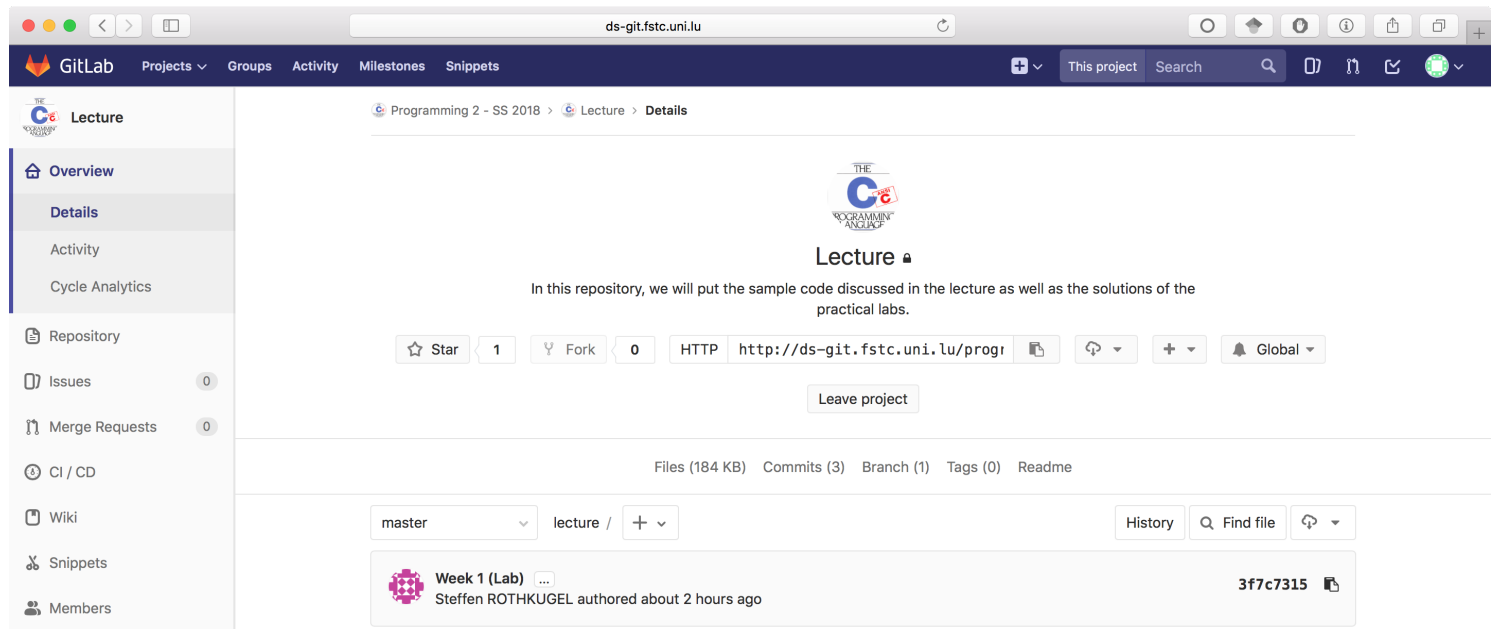


Figure 3 – Project Home

You can browse through the current state of the repository by clicking on *Files*. As shown in figure 4, the code examples from the lecture and lab solutions are organized in different folders. Inside these folders, you can see the contained files and directly from your web browser inspect the source and header files. Note that syntax highlighting is also provided.

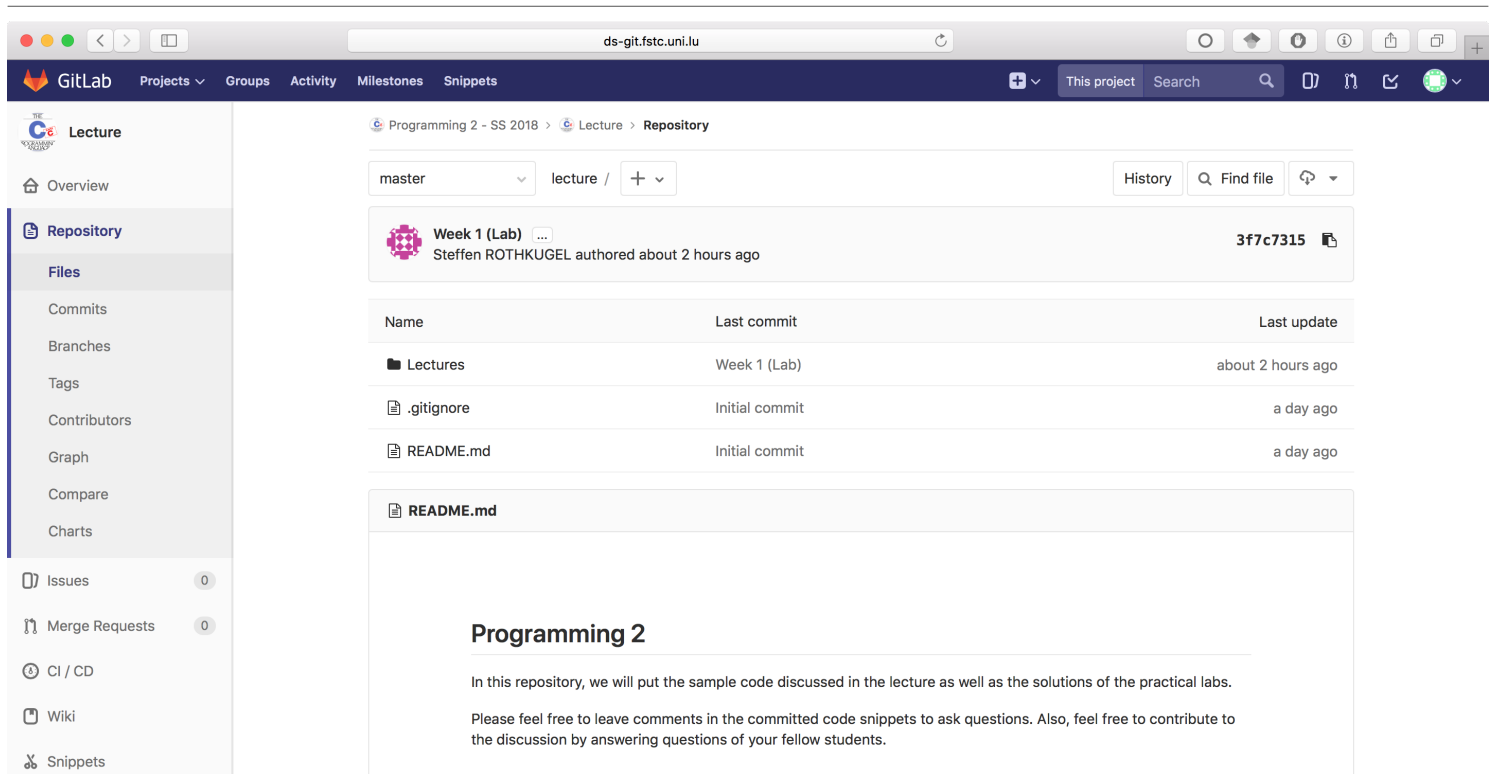


Figure 4 – Project Files

### 2.3.2 Commenting Code

As already mentioned, the main purpose is to enable you to ask questions on aspects you have not understood so far and discuss alternative approaches with the lecturer, the teaching assistants and even your fellow students. Assume you are inspecting a source file from a lab solution and do not remember the explanation given in the subsequent session. As shown in figure 5, there is a hash value in the upper-right corner. The value links to the latest *commit* of the file on the version control system.

When clicking on this link, the subsequent screen shows the changes performed on this file during the last commit. Positioning the cursor on the left margin of the code snippet shows, on a line-basis, a speech bubble icon. Clicking on latter opens a text field which you can use to ask questions or discuss alternative approaches, as shown in figure 6. Note that you can benefit from the *Markdown* markup language in your comments, but this is not an obligation. Click on the *Comment* button to submit.

### 2.3.3 Notifications

When a comment is written on a code snippet, the teaching team will be notified to be able to answer your questions in a timely manner. Of course, you are also kindly invited to participate in the discussion initiated by a fellow student. To be notified on a new comment from any project member, you have to change the notification level in the project home, from *Global* to *Custom*, as seen in figure 7. In the modal dialog, as shown in figure 8, choose *New note*. This setting is highly recommended to follow the discussion and learn from other people's questions. **Remember: There is no such thing as a stupid question!**

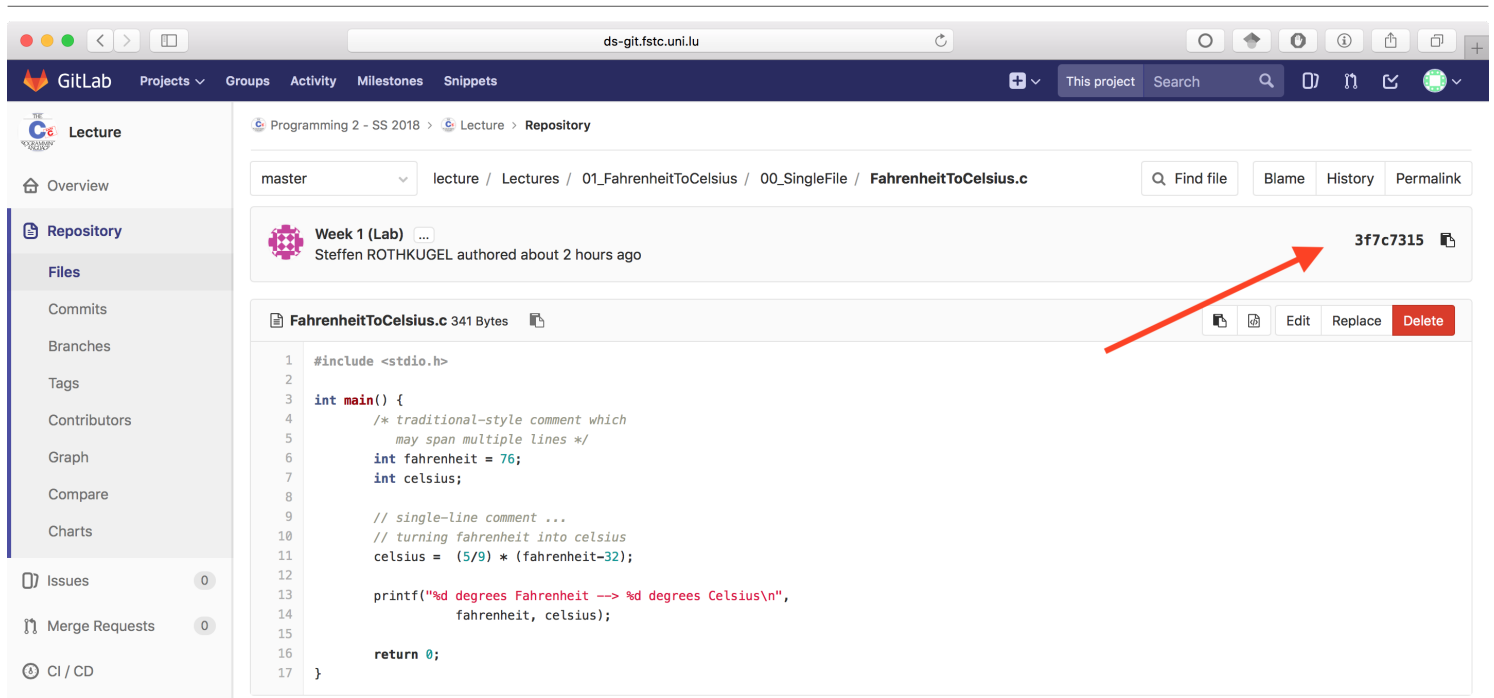


Figure 5 – File Inspection

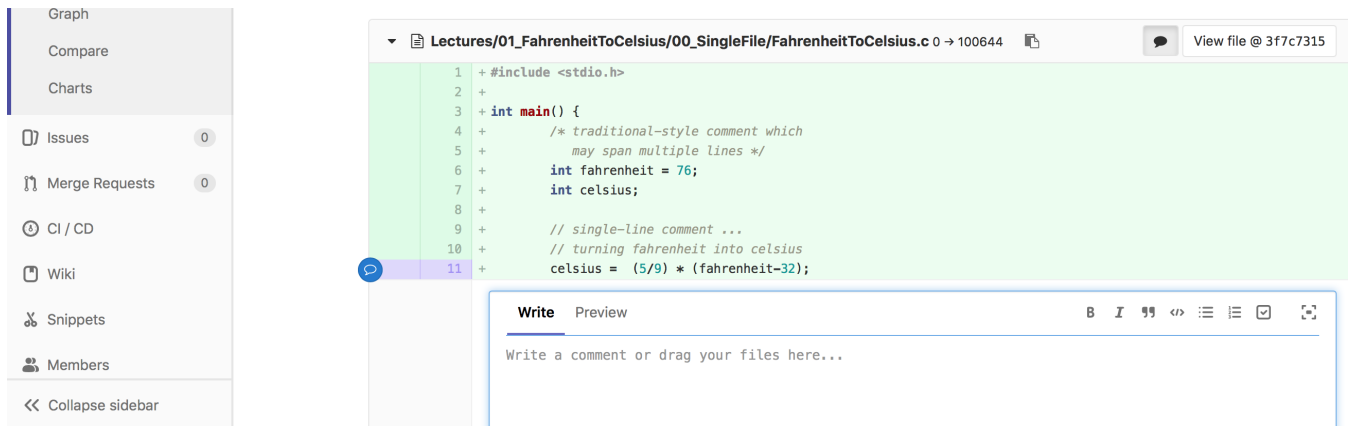


Figure 6 – Commenting a line of code

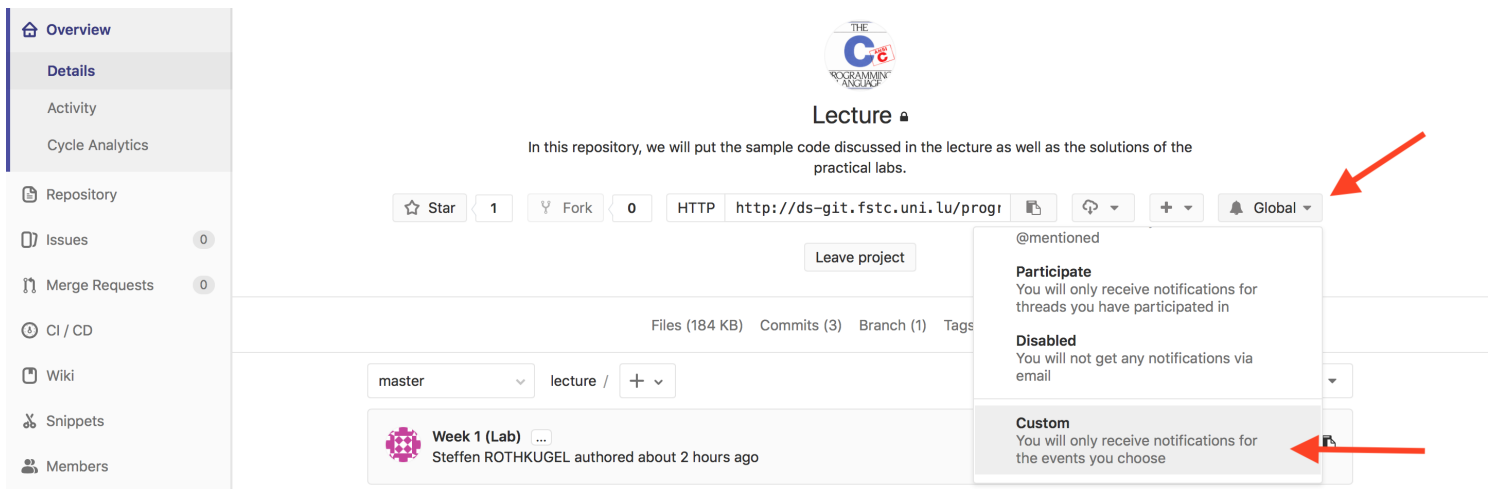


Figure 7 – Notification Levels

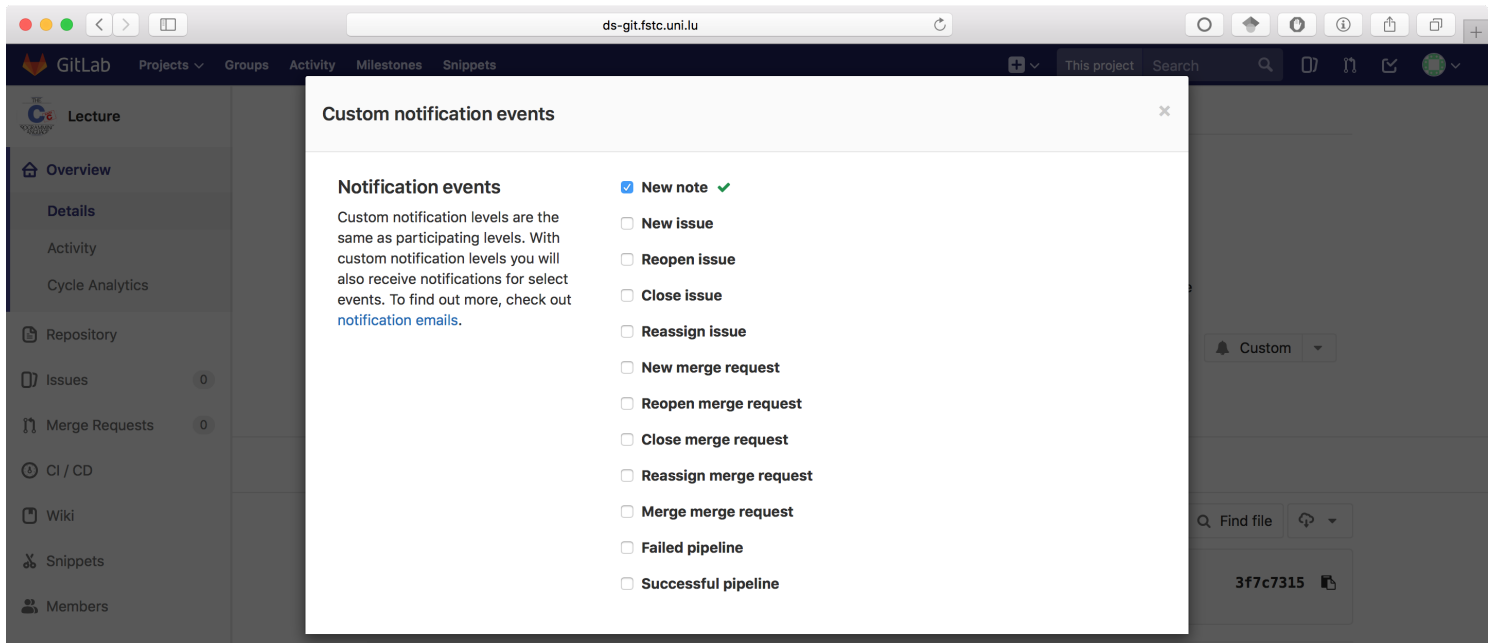


Figure 8 – Custom Notification Setting

### Exercise 3 – Temperature Conversion

[#VariableDeclaration](#) [#VariableInitialization](#) [#printf](#)

In this exercise, you will take first steps with Git and GitLab.

1° Begin by *cloning* the Git repository of the lecture and import it into Eclipse:

- Choose **File** > **Import...**, then select **Git** > **Projects from Git**, **Next**, **Clone URI**, **Next**
- As *URI*, put [https://ds-git.fstc.uni.lu/programming2\\_2018/lecture.git](https://ds-git.fstc.uni.lu/programming2_2018/lecture.git) and under *Authentication*, enter your Uni.lu credentials (**do not store** your credentials if using a VM on a lab workstation!), **Next**, choose *master*, **Next**
- Set a directory of your choice, e.g. `/home/student/programming2`, to clone the Git repository into, **Next**
- Select *Import using the New Project wizard*, **Finish**
- **C** > **C Project**, **Next**
- Set a project name, set the location to the subfolder `Lectures/01_FahrenheitToCelsius/00_SingleFile` (e.g. `/home/student/programming2/Lectures/01_FahrenheitToCelsius/00_SingleFile`), **Finish**

#### Easy, huh?

Admittedly, Eclipse sometimes requires a lot of navigating between dialog windows. A somehow easier way would be to use the command line tool `git`:

```
mkdir Lecture && cd Lecture
git clone https://ds-git.fstc.uni.lu/programming2_2018/lecture.git
```

Now, you can create a new C project with its location set to the above mentioned subfolder `00_SingleFile` and benefit from the already contained file.

#### Multiple source files with `main` function in Eclipse

Due to the way Eclipse builds the project, you are likely to get an error when having multiple source files with a `main` function in your project. Therefore, you will need to create a different project for each C console application.

As the above git repository obviously contains several source files with a `main` function, it is important to create a new C project for the specific subfolder.

2° Build and run the `FahrenheitToCelsius.c` source file. What do you observe? What might be the issue and what solution do you propose?

3° Go back to GitLab, look for the source file there, and comment, as described before, on the line(s) that cause(s) the issue. Feel free to discuss with fellow students in these comments about different ways to solve the problem!

If you don't understand something in the code, consult the learning material related to the ALMA hashtags next to the title of this exercise (you can directly click on them in the PDF, which will open a web browser). If there are still understanding issues, feel free to leave a comment on GitLab, such that a teacher or even a fellow student may respond to your question!

### Exercise 4 – Temperature Conversion – Reloaded

[#Typedef](#) [#FunctionDefinition](#) [#PreprocessorInclude](#)

Using the tags attached to this exercise, consult material on these topics to create a new C console application that converts temperatures between Celsius and Fahrenheit units. Particularly:

- *define a new type* `Temperature` as a synonym for the already existing type `float`;
- write two *functions*
  - `Temperature toFahrenheit(Temperature celsius)`
  - `Temperature toCelsius(Temperature fahrenheit)`

that calculate the respective conversion;

- put the *type definition* and function *prototypes* in a separate *header file* `temperature.h`, the function implementations in a separate source file `temperature.c` and the `main` function, which demonstrates the correct behavior calling the previous two functions, in a separate source file `main.c`