



ECA419 - Robótica

ECA419 - Robótica

Professores:

Dr. Guilherme Sousa Bastos

(sousa@unifei.edu.br)

Ms. Audeliano Wolian Li

(audeliano@unifei.edu.br)

Bs. Adriano Henrique Rossette Leite

(adrianohrl@unifei.edu.br)



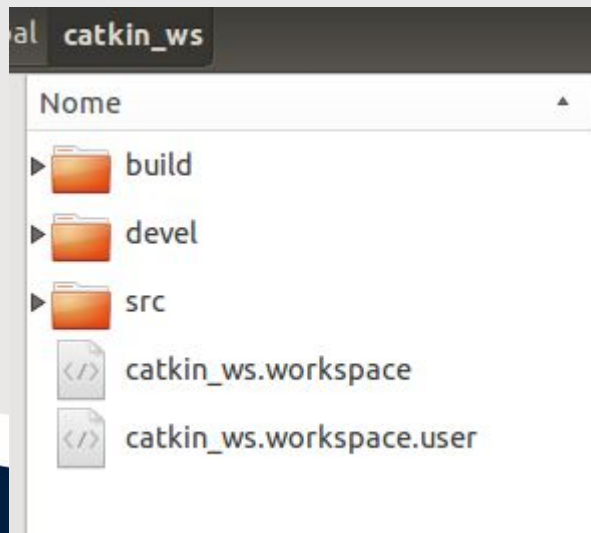
git clone

- Abra um terminal:
 - `$ cd ~/Documentos`
 - `$ git clone https://github.com/Expertinos/curso_ROS_UNIFEI`



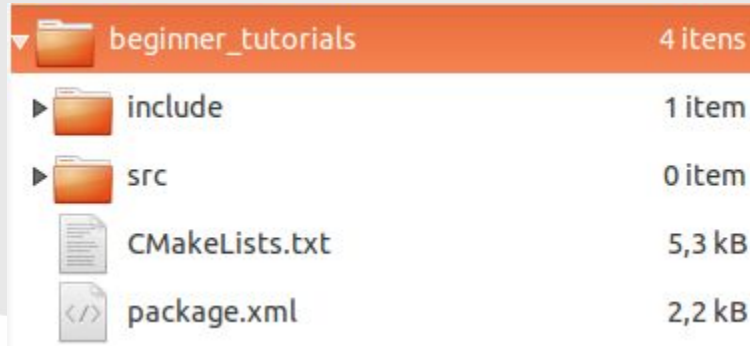
Catkin Workspace

- É a 'área de trabalho' onde todos os projetos são armazenados.
- Possui três diretórios: **build**, **devel** e **src**.
- **Build** e **devel** são gerados automaticamente pelo compilador e o **src** contém os códigos fontes de projetos desenvolvidos e/ou baixados de terceiros.



ROS Package

- Os pacotes são a principal unidade de organização no ROS.
- Possui dois arquivos essenciais para a compilação do projeto:
 - **package.xml** -> metadados para o ROS: nome do pacote, descrição, dependências,...
 - **CMakeLists.txt** -> contém regras para o compilador (catkin): diretório dos headers, bibliotecas utilizadas, endereço dos executáveis,...



| | | |
|---|--------------------|---------|
| ▼ | beginner_tutorials | 4 itens |
| ▶ | include | 1 item |
| ▶ | src | 0 item |
| | CMakeLists.txt | 5,3 kB |
| | package.xml | 2,2 kB |



package.xml

```
package.xml x
<package>
  <name>rosaria</name>
  <version>0.9.0</version>
  <description>
    <tt>ROSARIA</tt> provides a ROS interface for most Adept MobileRobots,
    MobileRobots Inc., and ActivMedia mobile robot bases including
    Pioneer 2, Pioneer 3, AmigoBot, PeopleBot, PowerBot, PatrolBot, Seekur,
    Seekur Jr., Pioneer LX,
    and any other past, current or future robot base supported by Adept MobileRobot's
    open source ARIA library.
    Information from the robot base, and velocity and acceleration control, is implemented
    via a <tt>RosAria</tt> node, which publishes topics providing data recieved from
    the robot's embedded controller by ARIA, and sets desired velocity, acceleration and
```



CMakeLists.txt

```
CMakeLists.txt x
# http://ros.org/doc/groovy/api/catkin/html/user\_guide/supposed.html
cmake_minimum_required(VERSION 2.8.3)
project(rosaria)
# Load catkin and all dependencies required for this package
# TODO: remove all from COMPONENTS that are not catkin packages.
find_package(catkin REQUIRED COMPONENTS message_generation roscpp nav_msgs geometry_msgs
sensor_msgs tf
dynamic_reconfigure)

# Set the build type. Options are:
# Coverage      : w/ debug symbols, w/o optimization, w/ code-coverage
# Debug         : w/ debug symbols, w/o optimization
# Release       : w/o debug symbols, w/ optimization
# RelWithDebInfo : w/ debug symbols, w/ optimization
# MinSizeRel    : w/o debug symbols, w/ optimization, stripped binaries
#set(ROS_BUILD_TYPE RelWithDebInfo)

#set the default path for built executables to the "bin" directory
#set(EXECUTABLE_OUTPUT_PATH ${PROJECT_SOURCE_DIR}/bin)
```



Criando um Pacote

- Abrir um terminal;
- Acessar o diretório **catkin_ws/src**;
 - `$ cd ~/catkin_ws/src`
- Criar um pacote com nome “aula2” e adicionar 3 dependências;
 - `$ catkin_create_pkg <nome_do_pacote> std_msgs roscpp rospy`
- Acessar o diretório do pacote criado;
 - `$ roscd <nome_do_pacote>`
- Compilar;
 - `$ cd ~/catkin_ws/`
 - `$ catkin_make`



Node

- Nós são processos que executam algum tipo de computação, podendo ser considerados como **executáveis**.
 - Driver de câmera, controlador de movimentos, planejador de trajetórias,...

Simple C++ Program

```
int main(int argc, char* argv[])
{
    std::cout << "Hello World!";

    return 0;
}
```

Simple C++ ROS Node

```
#include <ros/ros.h>

int main(int argc, char* argv[])
{
    ros::init(argc, argv, "hello");
    ros::NodeHandle node;

    ROS_INFO_STREAM("Hello World!");

    return 0;
}
```



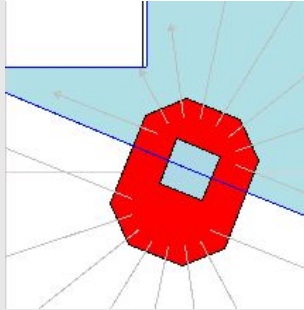
Baixando um Pacote de Terceiros

- Abrir um terminal;
- Executar o comando;
 - `$ sudo apt install ros-indigo-teleop-twist-keyboard`



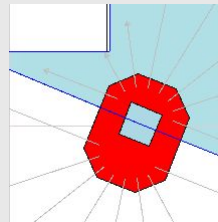
Demonstração motivadora

ROS.org





Demonstração motivadora



1) Inicie a Máquina Virtual, abra um terminal (Ctrl+Alt+T) e inicie o ROS Master:

~\$ roscore

2) No windows, abrir o simulador MobileSim e posicione o robô dentro do ambiente, se necessário.

3) Numa nova aba no terminal (Ctrl+Shift+T), conecte-se com o simulador (lembre-se sempre de usar a tecla Tab para evitar erros de digitação):

~\$ rosrund rosaria RosAria _port:=<IP do windows>:8101

4) Numa nova aba no terminal (Ctrl+Shift+T), teleopere o robô, através de:

~\$ rostopic list (qual tópico publica o comando de velocidade?)

~\$ rosrund teleop_twist_keyboard teleop_twist_keyboard.py
cmd_vel:=/RosAria/cmd_vel



Comando ROS Node

- **rostopic**
 - rostopic list
 - rostopic info <nome_do_nó>
 - rostopic kill <nome_do_nó>
 - Ctrl+C





QtCreator IDE

Lê-se: **Cute (Q. T.) Creator!!!**

É um Ambiente de Desenvolvimento Integrado (**IDE** - Integrated Development Environment), ou seja, é um programa de computador que **reúne características e ferramentas de apoio ao desenvolvimento de software** com o objetivo de agilizar este processo.

Mas por que não usar o NetBeans, o Eclipse ou outra IDE?

Porque, até agora, só no QtCreator existe um plugin para o ROS!!!



visite:

https://github.com/ros-industrial/ros_qtc_plugin



Configurando ROS Workspace no QtCreator



Projects

Examples

Tutorials

New to Qt?

Learn how to develop
your own applications and
explore Qt Creator.

Get Started Now

Qt Account

Online Community

Blogs

User Guide

+ New Project

Open Project

Sessions

default (current session)

Recent Projects

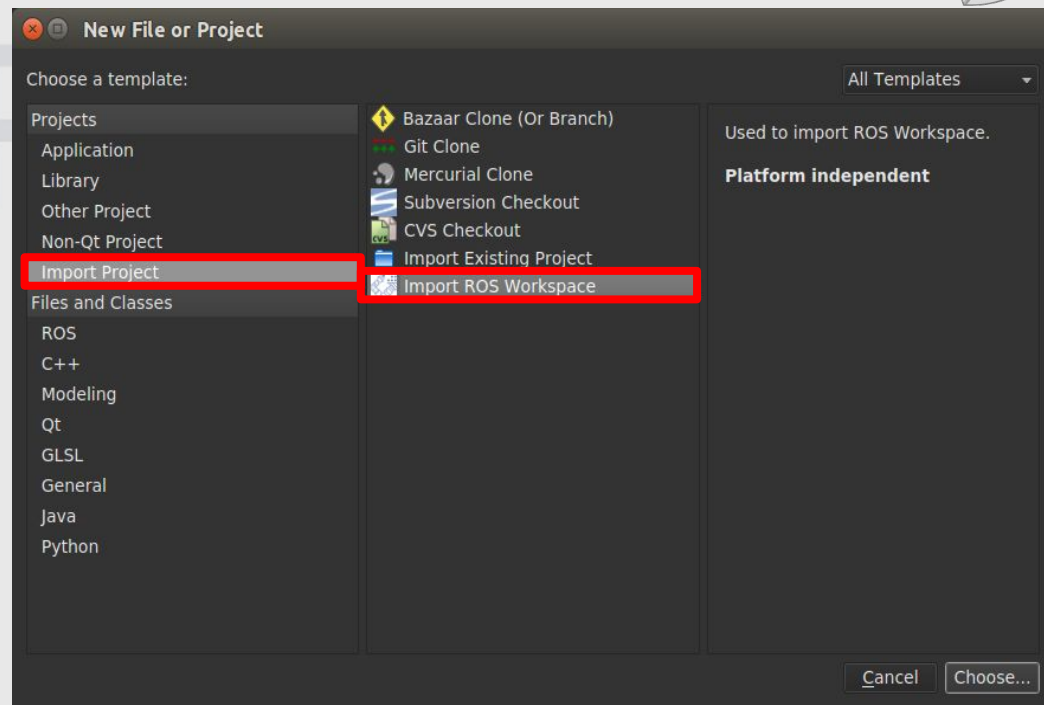
catkin_ws
Default



Importando ROS Workspace

Para importar o workspace do ROS para o QtCreator:

Abra **File** -> **New File or Project...**, ou simplesmente pressione **Ctrl+N**. Selecione as opções como mostra ao lado. Em seguida, pressione **Choose....**





Configurando Projeto

Na próxima janela, insira o **nome** que você desejar para o projeto. Selecione a ROS Distro (**Indigo**). Então, localize o caminho do ROS Worspace (**~/catkin_ws**). Finalmente, pressione o botão **Generate Project File** para poder avançar para a próxima janela. Avance para a última janela pressionando o botão **Next >**.

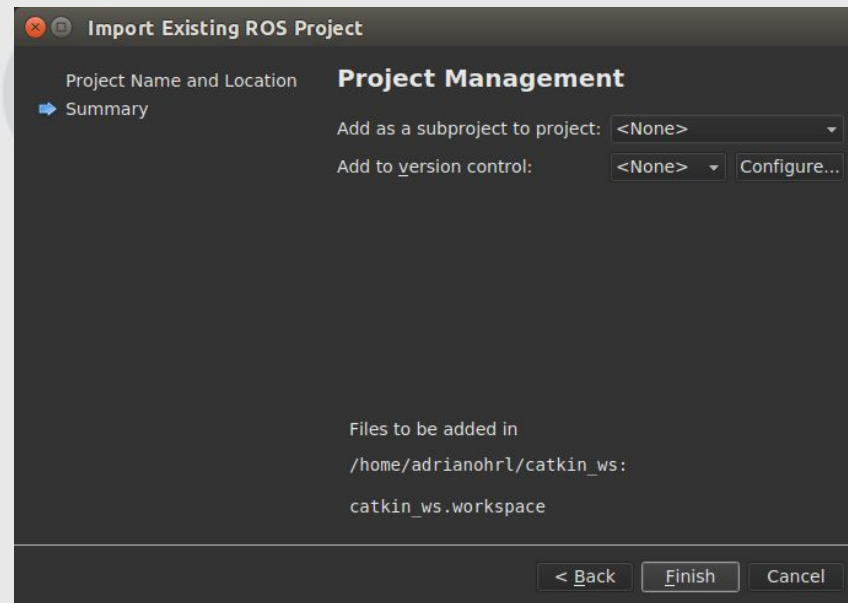
A screenshot of the 'Import Existing ROS Project' dialog box. The dialog has a title bar with a close button and a maximize button. The main content area is divided into two panes. The left pane is titled 'Project Name and Location' and contains a 'Summary' section. The right pane is also titled 'Project Name and Location' and contains the following fields: 'Name:' with the value 'catkin_ws', 'Distribution:' with a dropdown menu showing 'indigo', 'Workspace Path:' with the value 'drianohrl/catkin_ws' and a 'Browse...' button, and 'Output:' with an empty text area. At the bottom right of the right pane is a 'Generate Project File' button. At the bottom of the dialog are two buttons: 'Next >' and 'Cancel'. Red rectangular boxes highlight the 'Name' field, the 'Distribution' dropdown, the 'Workspace Path' field and its 'Browse...' button, the 'Generate Project File' button, and the 'Next >' button.

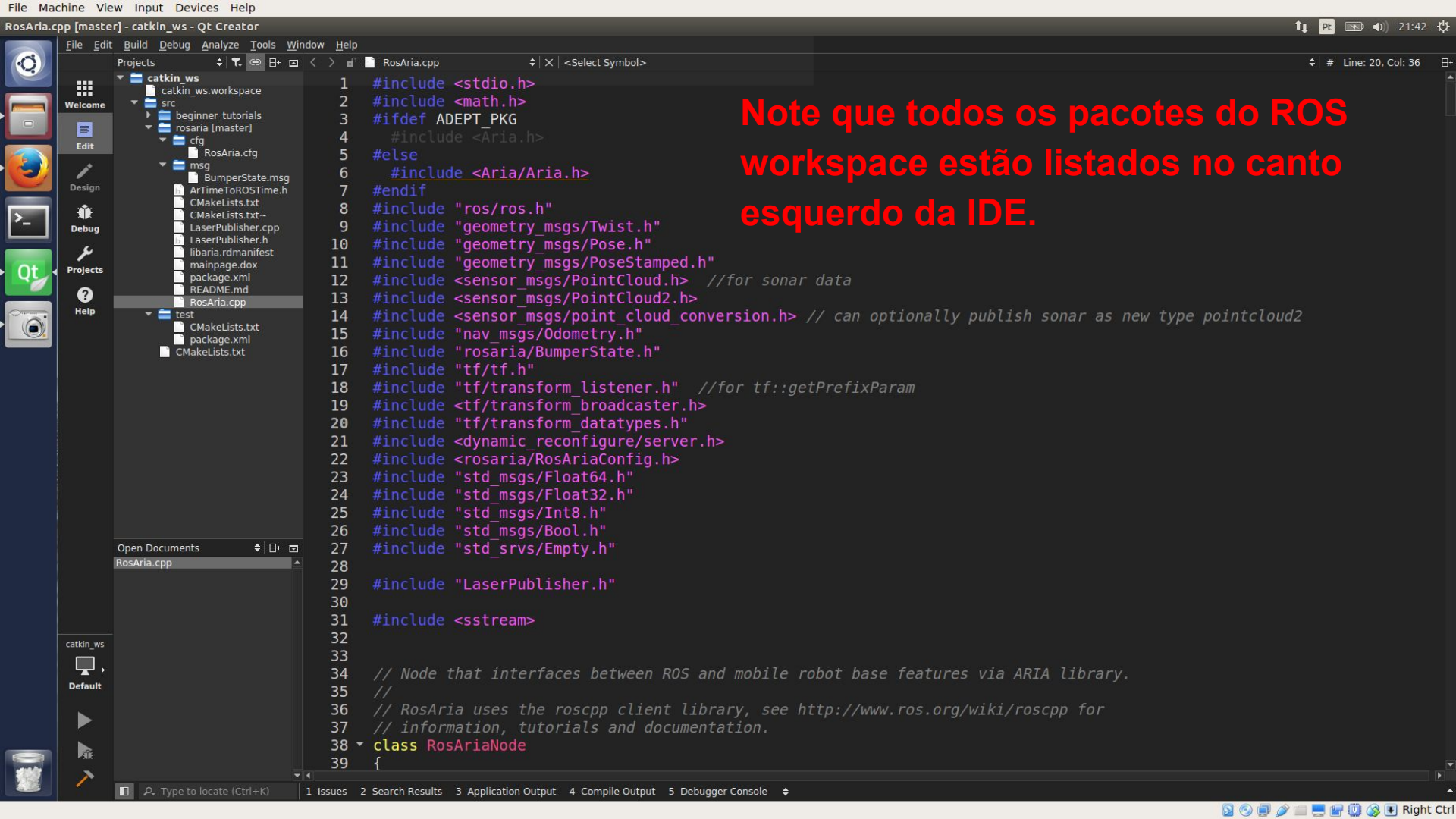


Configurando Projeto

Apenas pressione o botão **Finish** para encerrar a configuração do Projeto ROS no QtCreator.

Note que foi gerado um arquivo de configuração do projeto (de extensão `.workspace` e com o mesmo nome do projeto) no diretório escolhido.

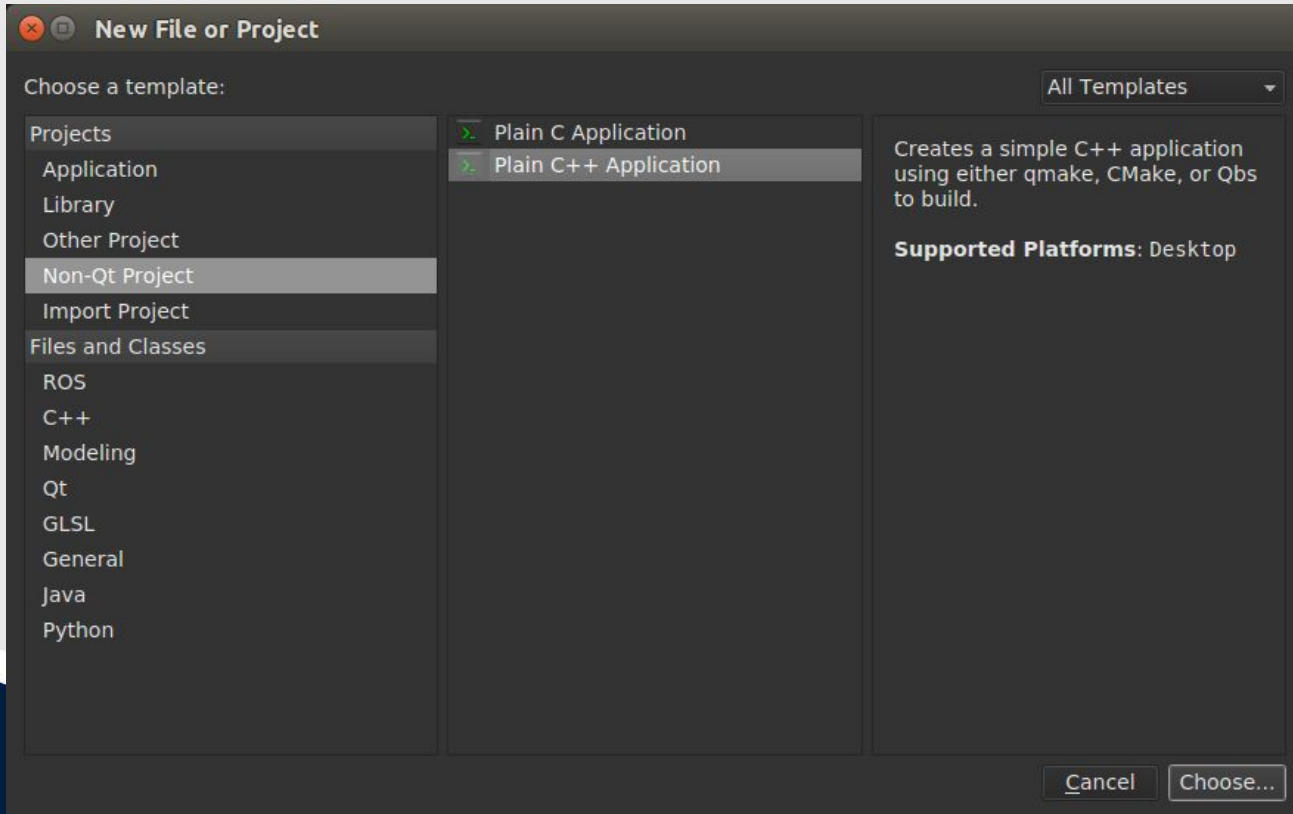






Configurando Projeto em CMake no QtCreator

Criando novo projeto CMake



Configurando projeto CMake



Plain C++ Application

Location
Build System
Kits
Summary

Project Location

Creates a simple C++ application using either qmake, CMake, or Qbs to build.

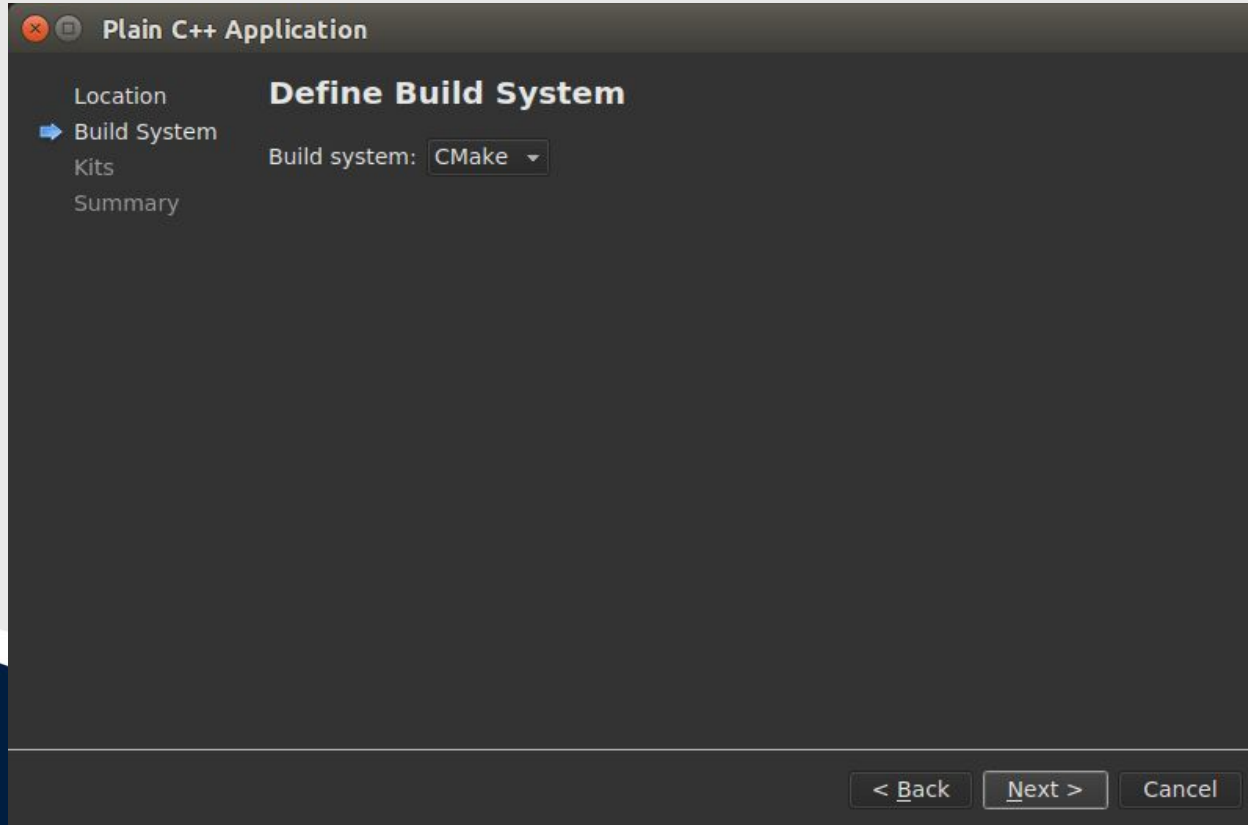
Name:

Create in:

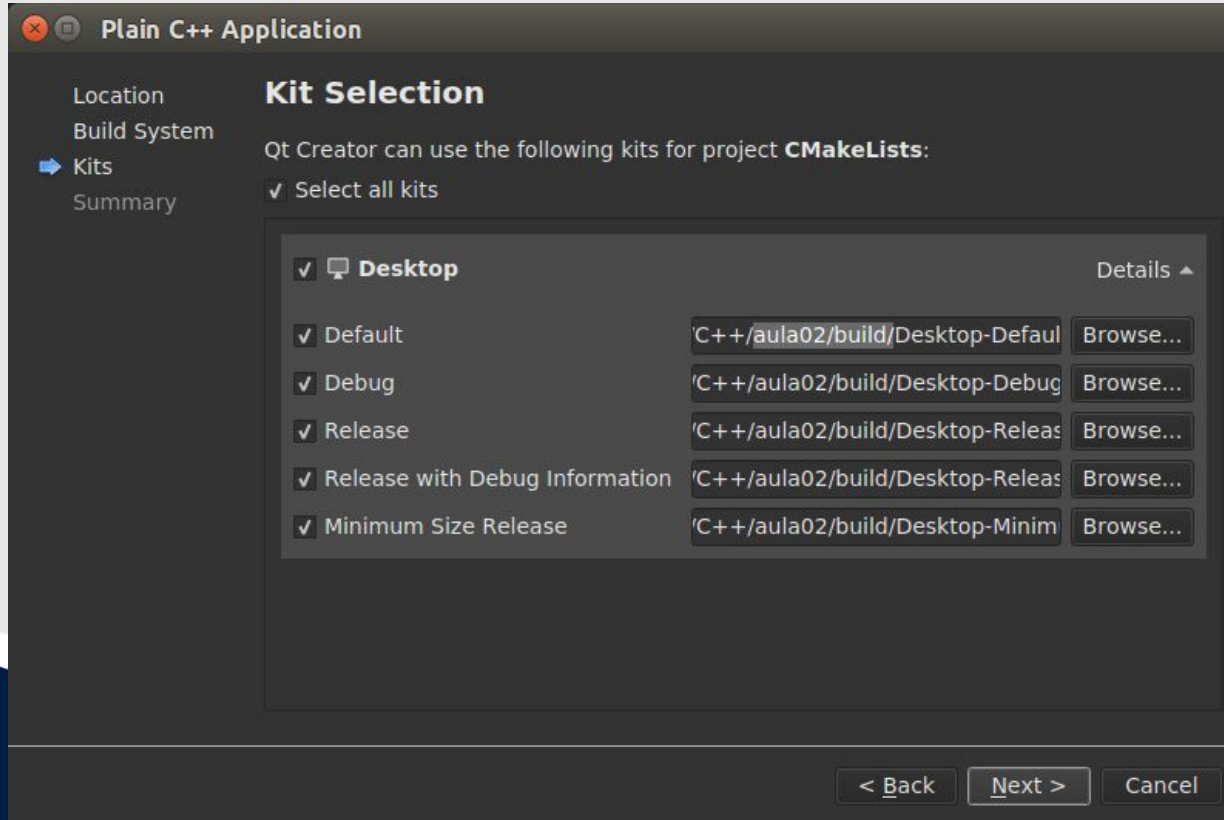
☒ Use as default project location



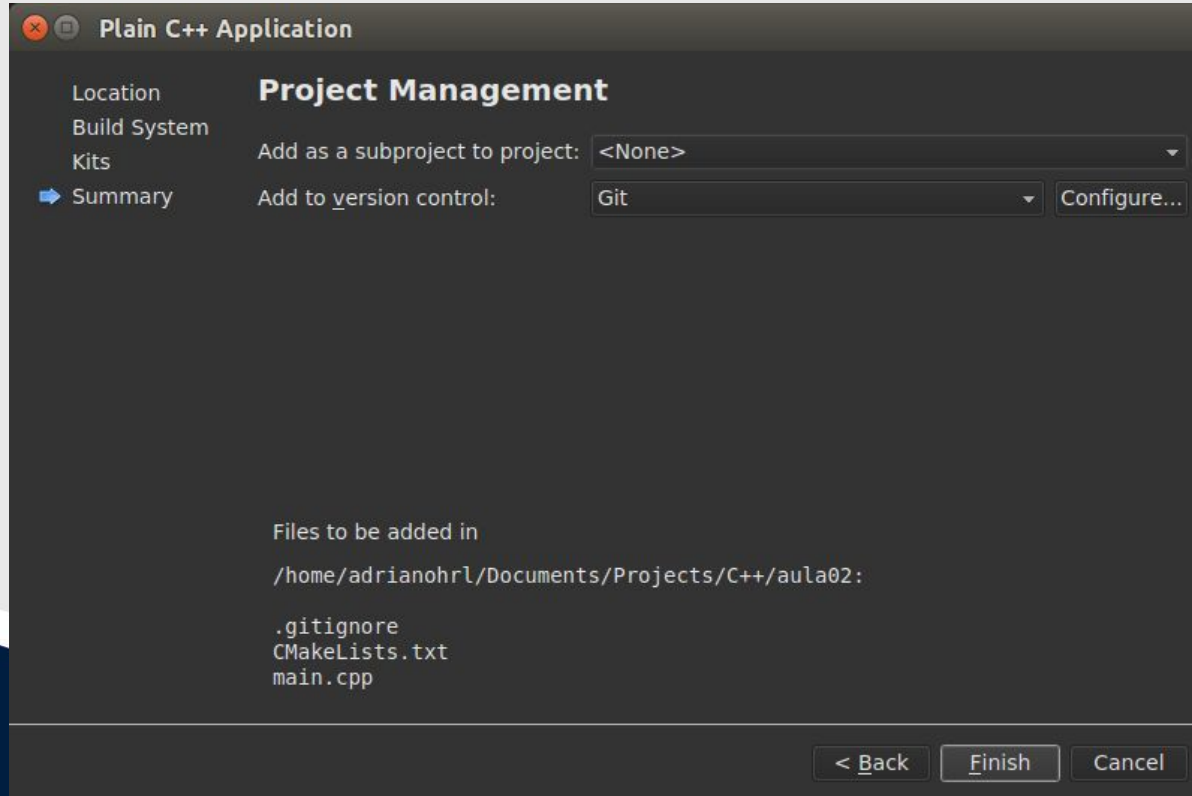
Configurando projeto CMake



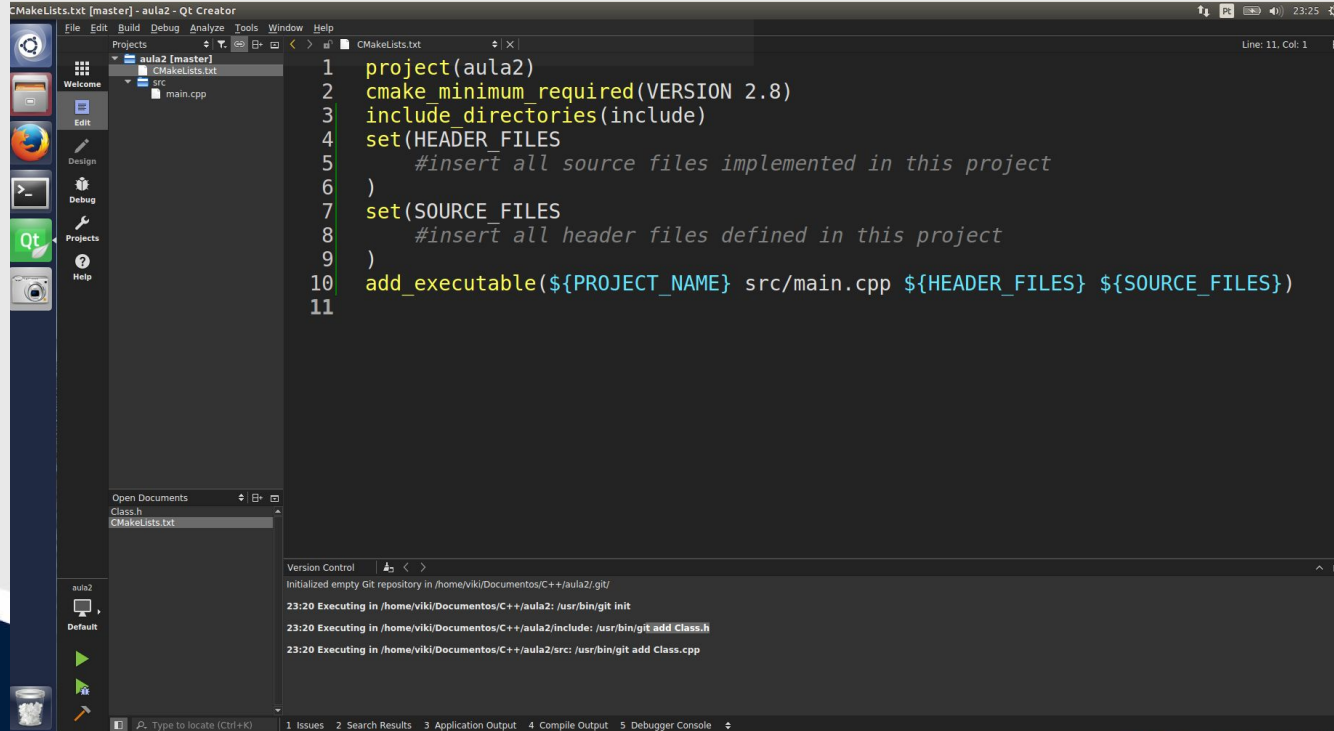
Configurando projeto CMake



Configurando projeto CMake



Configurando o arquivo CMakeLists.txt



The image shows the Qt Creator IDE interface. The main editor window displays the CMakeLists.txt file for a project named 'aula2'. The code defines the project, sets the minimum CMake version to 2.8, includes directories, sets header and source files, and adds an executable. The left sidebar shows the project structure with 'aula2 [master]' and 'CMakeLists.txt'. The bottom status bar shows the execution path and output.

```
1 project(aula2)
2 cmake_minimum_required(VERSION 2.8)
3 include_directories(include)
4 set(HEADER_FILES
5     #insert all source files implemented in this project
6 )
7 set(SOURCE_FILES
8     #insert all header files defined in this project
9 )
10 add_executable(${PROJECT_NAME} src/main.cpp ${HEADER_FILES} ${SOURCE_FILES})
11
```

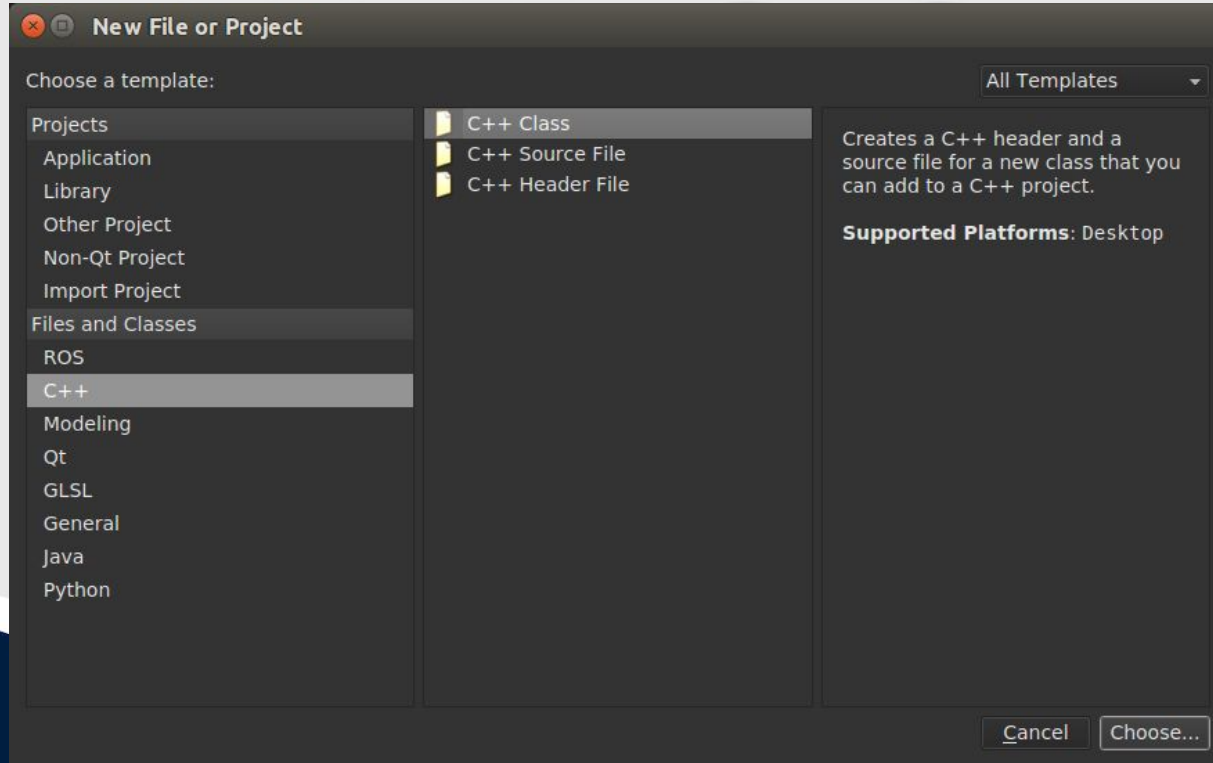
Version Control
Initialized empty Git repository in /home/viki/Documents/C++/aula2/.git/
23:20 Executing in /home/viki/Documents/C++/aula2: /usr/bin/git init
23:20 Executing in /home/viki/Documents/C++/aula2/include: /usr/bin/git add Class.h
23:20 Executing in /home/viki/Documents/C++/aula2/src: /usr/bin/git add Class.cpp





**Criando nova classe no
projeto CMake**

Criando nova classe no projeto CMake



Criando nova classe no projeto CMake



C++ Class

Details
Summary

Define Class

Class name:

Base class:

☐ Include QObject

☐ Include QWidget

☐ Include QMainWindow

☐ Include QDeclarativeItem - Qt Quick 1

☐ Include QQuickItem - Qt Quick 2

☐ Include QSharedData

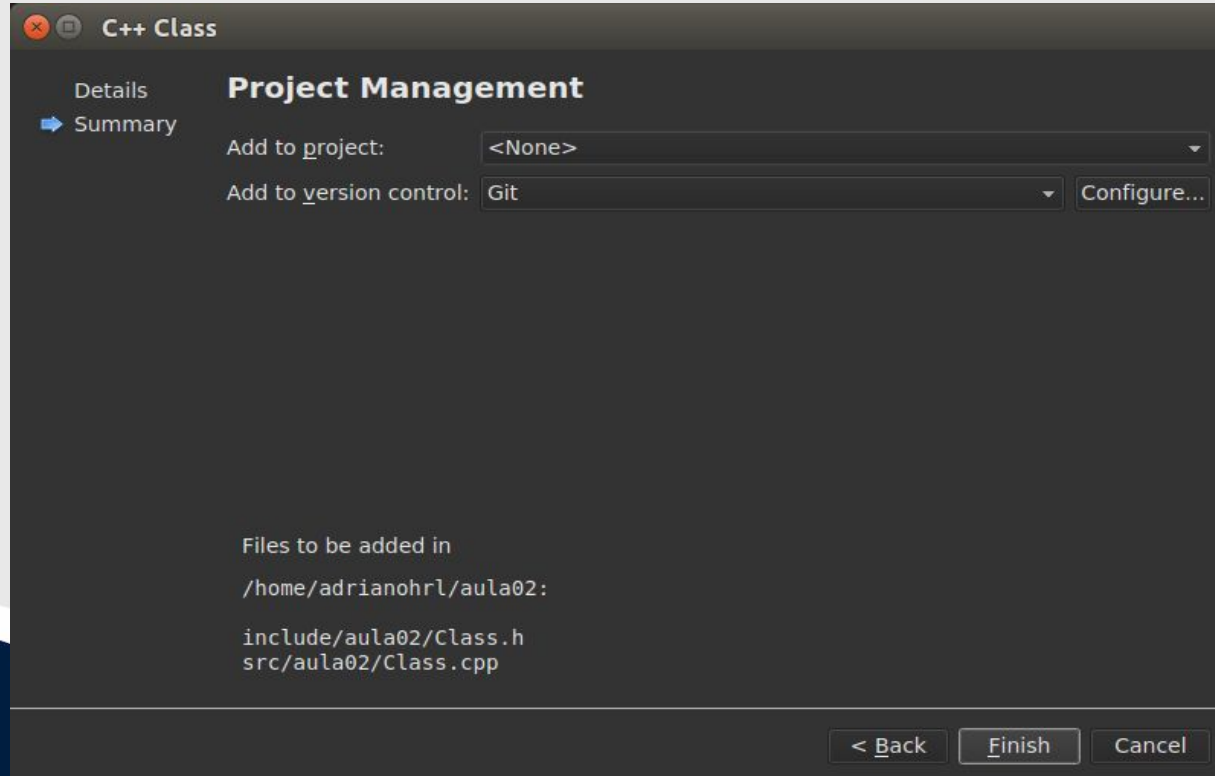
Header file:

Source file:

Path:

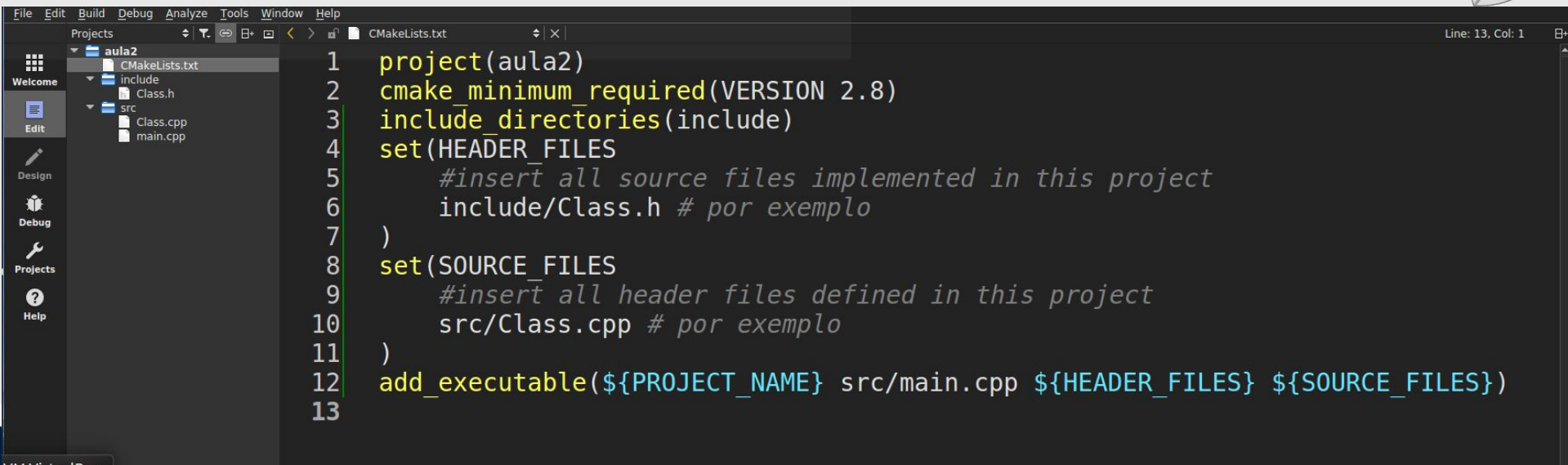


Criando nova classe no projeto CMake





Reconfigurando CMakeLists.txt

A screenshot of the Qt Creator IDE. The left sidebar shows the 'Projects' panel with a tree view containing 'aula2', 'CMakeLists.txt', 'include', 'Class.h', 'src', 'Class.cpp', and 'main.cpp'. The main editor window displays the 'CMakeLists.txt' file with the following content:

```
1 project(aula2)
2 cmake_minimum_required(VERSION 2.8)
3 include_directories(include)
4 set(HEADER_FILES
5     #insert all source files implemented in this project
6     include/Class.h # por exemplo
7 )
8 set(SOURCE_FILES
9     #insert all header files defined in this project
10    src/Class.cpp # por exemplo
11 )
12 add_executable(${PROJECT_NAME} src/main.cpp ${HEADER_FILES} ${SOURCE_FILES})
13
```





Atalhos no QtCreator

Ctrl+Space

mostra possíveis opções de código;

Alt+0

mostra/esconde Sidebar (painél lateral esquerdo);

Esc

esconde os Output Panes (painéis de saída inferior);

Ctrl+F

procura expressão no código fonte;

Ctrl+Shift+R

renomeia variável (**utilizar só para variáveis locais**);

Ctrl+B

compila projeto ativo;

Ctrl+R

roda executável escolhido em configuração (**Projects->Run**);

F5

depura código, se devidamente configurado (**Projects->Build**).





Executando via terminal

```
viki@c3po:~$ cd Documentos/C++/  
aula2/  
build-aula2-Desktop-Default/  
viki@c3po:~$ cd Documentos/C++/build-aula2-Desktop-Default/  
viki@c3po:~/Documentos/C++/build-aula2-Desktop-Default$ ls  
aula2      CMakeCache.txt  cmake_install.cmake  
aula2.cbp  CMakeFiles      Makefile  
viki@c3po:~/Documentos/C++/build-aula2-Desktop-Default$  
./aula2  
Hello World!  
viki@c3po:~/Documentos/C++/build-aula2-Desktop-Default$
```





**Linguagem C++ e Paradigma de
Programação Orientada à Objetos**



Trabalho 1

Parte 1

Trabalho 1 (parte 1)

Multi-Robot System (MRS): a group of robots that are designed aiming to perform some collective behavior (e.g. rescue and search).

One of the most challenging problems of MRS is how to optimally assign a **set of robots** to a **set of tasks** in such a way that optimizes the **overall system performance** subject to a **set of constraints**. This problem is known as **Multi-Robot Task Allocation (MRTA)**.



Trabalho 1 (parte 1)

Pode ser realizado com até **4 integrantes**.

Enviar e-mail com os integrantes do grupo para execução do trabalho 1 até dia **12/09** (para o email: **eca419.unifei@gmail.com**).

Entregar a versão final da parte 1 do trabalho 1 até dia **22/09** (para o email: **eca419.unifei@gmail.com**), pois este trabalho será utilizado na aula do dia **23/09**.

Recomenda-se utilizar tecnologia **git** para versionar o código (vide **GitHub**).

