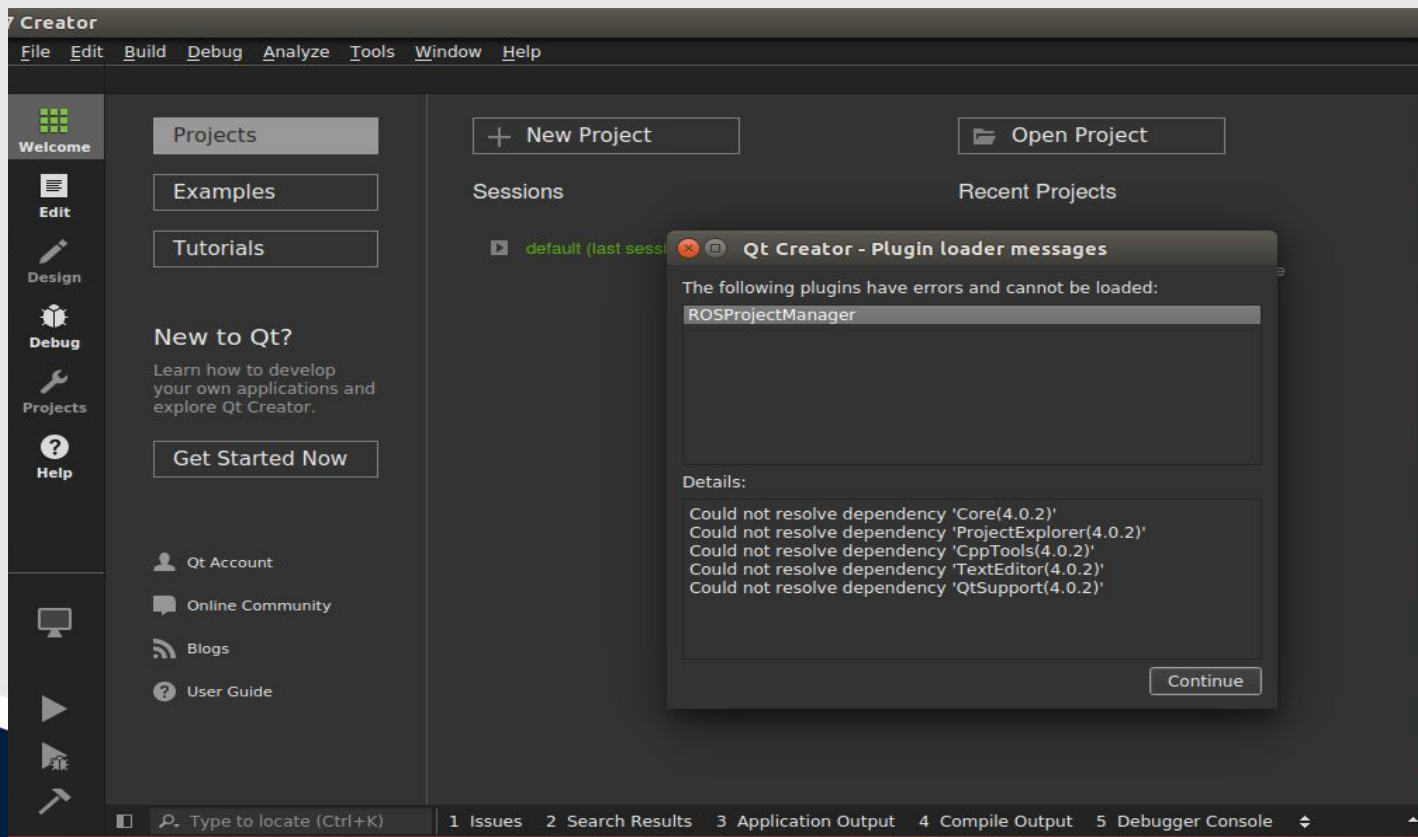




## Aula 3

**ECA419 - Robótica**

# Correção dos Erros do Qt



# Correção dos Erros do Qt

- Acesse o link e execute em um terminal os 3 comandos da seção 1.1:
  - [https://github.com/ros-industrial/ros\\_qtc\\_plugin/wiki/1.-How-to-Install-\(Use rs\)#section1.1](https://github.com/ros-industrial/ros_qtc_plugin/wiki/1.-How-to-Install-(Use rs)#section1.1)
- Abra o qt:
  - File>New File or Project>Projects>Import Project>Import ROS Workspace



# Correção dos Erros do Terminal

- Caso ocorra o erro “Warning: error while crawling /home/viki: boost::filesystem::status: Permission denied: "/home/viki/.gvfs"” após apertar “tab” para completar um comando, execute os comandos no terminal:
  - *sudo umount /home/viki/.gvfs*
  - *sudo rm -rf .gvfs*



# Objetivos da aula



# Mensagem

- Para representar os valores dos dados transferidos, são usados as **mensagens**.
- Conforme o tipo do dado, a mensagem acompanha o seu tipo, podendo ser:
  - Integer, boolean, array, etc.
- Pode ser também um conjunto de tipos primitivos (**struct**):
  - geometry\_msgs/Pose2D
  - std\_msgs/String
- Pode-se também criar sua própria mensagem (será abordado mais a frente).

[http://docs.ros.org/kinetic/api/std\\_msgs/html/msg/String.html](http://docs.ros.org/kinetic/api/std_msgs/html/msg/String.html)

[http://docs.ros.org/kinetic/api/geometry\\_msgs/html/msg/Pose2D.html](http://docs.ros.org/kinetic/api/geometry_msgs/html/msg/Pose2D.html)



# Tópicos

- Os nós trocam informações através dos tópicos, que podem ser considerados como um **barramento de dados**.
- Os nós que colocam dados em um certo tópico são chamados de **publisher**.
- Enquanto os nós que fazem a leitura dos dados presentes no tópico, são os **subscriber**.
- Em um mesmo nó podemos ter tanto publishers quanto subscribers.
- OBS: o tipo do tópico deverá ser **igual** ao tipo da mensagem.



# Nós

- Nós são processos que executam algum tipo de computação, podendo ser considerados como **executáveis**.





# Criação dos nós emissor/receptor

- Os passos foram baseados no tutorial do wiki.ros presente em:
  - <http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscribe%28c%2B%2B%29>




# Nó emissor



```
emissor.cpp
1  #include <ros/ros.h>
2  #include "std_msgs/String.h"
3  #include <sstream>
4
5  int main(int argc, char **argv)
6  { // Set up ROS.
7      ros::init(argc, argv, "emissor");
8      ros::NodeHandle n;
9
10     ros::Publisher conversa_pub = n.advertise<std_msgs::String>("conversa",1000);
11
12     ros::Rate loop_rate(10);
13
14     int cont = 0;
15     while(ros::ok())
16     {
17         std_msgs::String msg;
18
19         std::stringstream ss;
20         ss << "Hello World_" << cont;
21         msg.data = ss.str();
22
23         ROS_INFO ("%s", msg.data.c_str());
24
25         conversa_pub.publish(msg);
26         ros::spinOnce();
27         loop_rate.sleep();
28         ++cont;
29         if(cont==50) n.shutdown();
30     }
31     return 0;
32 }
33
```

# Nó receptor



```
< > [lock icon] [file icon] receptor.cpp [dropdown] [X] <No Symbols>

1  #include <ros/ros.h>
2  #include "std_msgs/String.h"
3
4  void conversaCallback(const std_msgs::String::ConstPtr& msg)
5  {
6      ROS_INFO("I heard: [%s]",msg->data.c_str());
7  }
8
9  int main(int argc, char **argv)
10 {
11     // Set up ROS.
12     ros::init(argc, argv, "receptor");
13     ros::NodeHandle n;
14
15     ros::Subscriber conversa_sub = n.subscribe("conversa", 1000, conversaCallback);
16     ros::spin();
17     return 0;
18 }
19
```

# Editando o CMakeLists.txt

```
105  ## Declare a cpp executable
106  add_executable(emissor_node src/emissor.cpp)
107
108  ## Add cmake target dependencies of the executable/library
109  ## as an example, message headers may need to be generated before nodes
110  add_dependencies(emissor_node beginner_tutorials_generate_messages_cpp)
111
112  ## Specify libraries to link a library or executable target against
113  target_link_libraries(emissor_node
114    ${catkin_LIBRARIES}
115  )
116
117
118  add_executable(receptor_node src/receptor.cpp)
119
120  ## Add cmake target dependencies of the executable/library
121  ## as an example, message headers may need to be generated before nodes
122  add_dependencies(receptor_node beginner_tutorials_generate_messages_cpp)
123
124  ## Specify libraries to link a library or executable target against
125  target_link_libraries(receptor_node
126    ${catkin_LIBRARIES}
127  )
128
```



# Testando os nós

- Abrir um terminal e execute a master:
  - `roscore`
- Segundo terminal para o emissor:
  - `roslaunch [nome_do_pacote] [nome_do_nó]`
  - `roslaunch aula3 emissor_node`
- Terceiro terminal para o receptor:
  - `roslaunch aula3 receptor_node`

