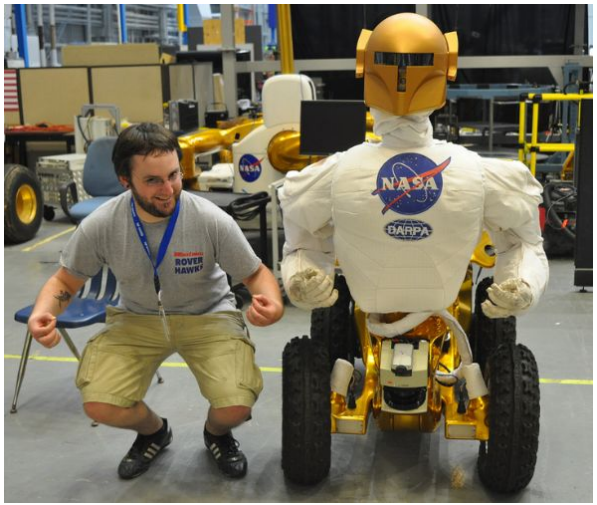
Introduction to ROS

James Kuczynski

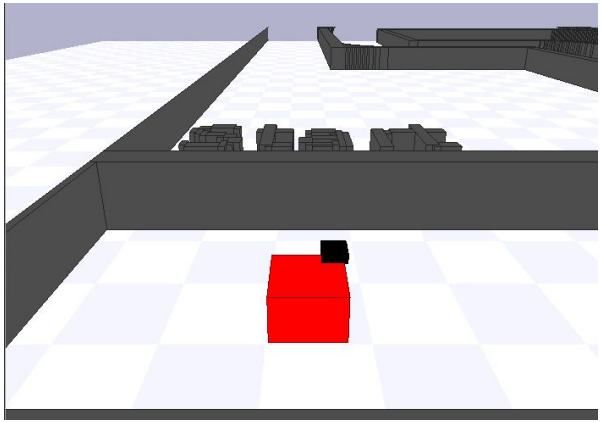
What Is ROS?

- “open-source, meta-operating system for your robot” *(ROS webpage)*
- Module-based set of libraries
- Custom environment
- Real-time simulation and visualization software
- Software stack
- Data communication framework
- Universal standard for roboticists *(sort of)*
- Runs on UNIX *(partial support for OSX and Windows)*
- Centralized system *(this will be optional in ROS2)*



What Is ROS? (contd.)

- Allows hardware abstraction
  - Same code can run on different platforms with little or no change
- Test software in simulation before running it on an expensive robot



What Isn’t ROS?

- Secure
- Flight certified
- The Skynet API

Release Cycle

We will be using ROS Indigo with CentOS 7

- ROS has a biannual release cycle
- Provides a “LTS” release once every two years
- Releases coincide with Ubuntu Linux versions
- Backwards compatibility is ***not*** guaranteed

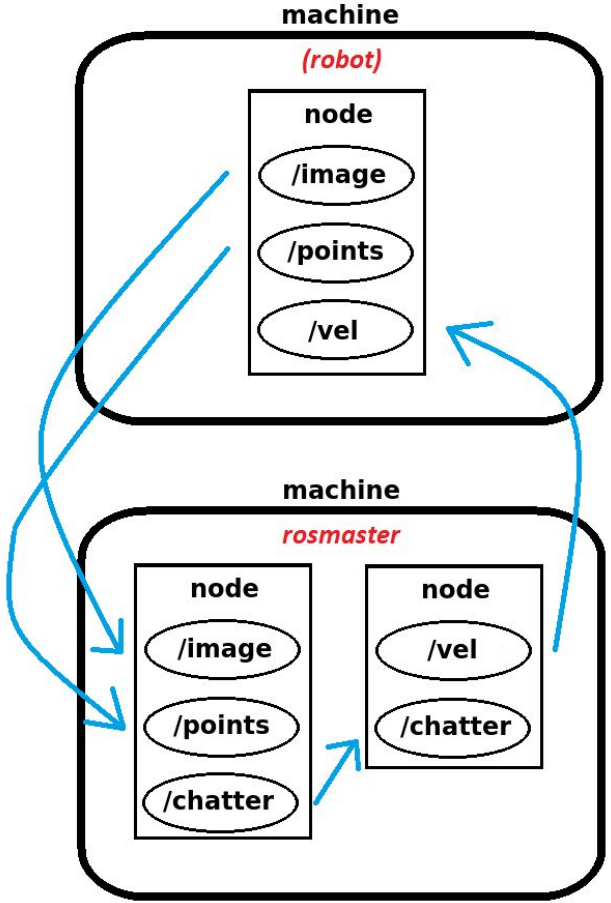
Who Uses ROS?

- Companies *(mainly for prototyping)*
- Research universities
- Robot hobbyists

- Rethink Robotics
- OSRF
- IRobot
- Google
- UC Berkeley
- Carnegie Mellon
- And many, many more...

ROS Program Structure

- Supports communication across **multiple machines** and **multiple processes**
- Each **process** is referred to as a **node** *(or nodelet -- more on that later)*
- ROS nodes communicate over **topics**



Environment

ROS uses environment variables

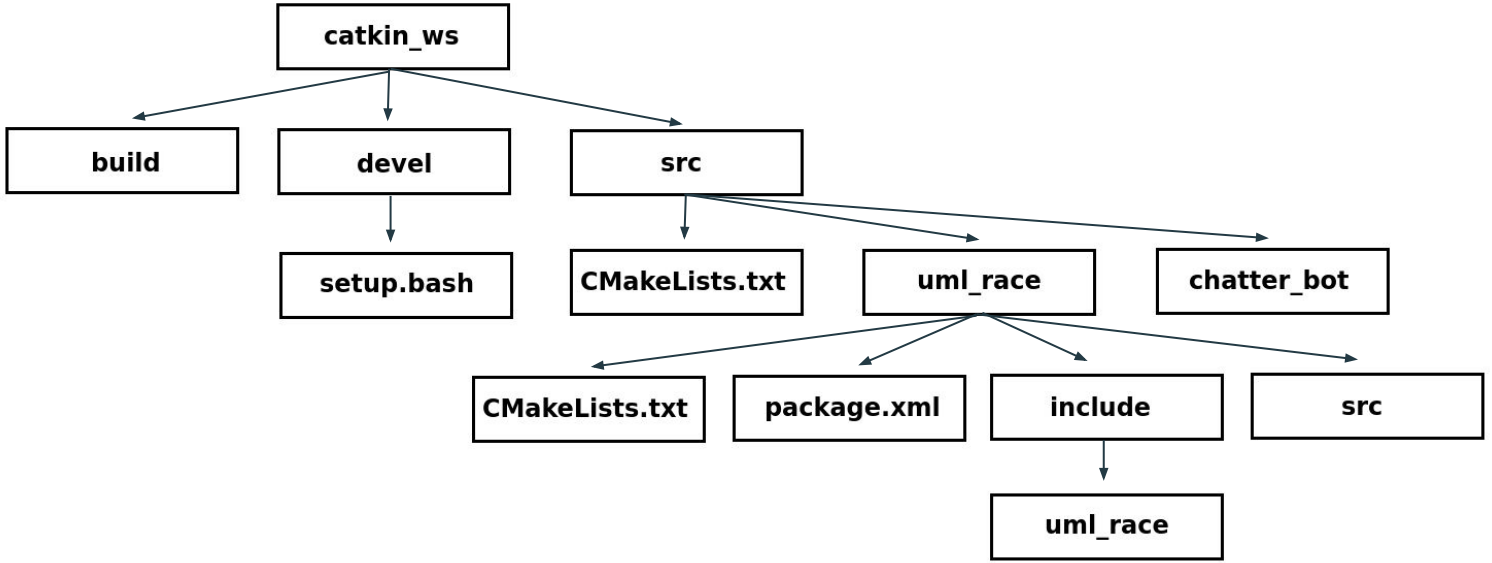
```
james@scarlet-alpha: ~$ env | grep ROS
ROS_ROOT=/opt/ros/rosdesktop_ws/install_isolated/share/ros
ROS_PACKAGE_PATH=/opt/ros/rosdesktop_ws/src/xacro:/opt/ros/rosdesktop_ws/install_isolated
/share:/opt/ros/rosdesktop_ws/install_isolated/stacks
ROS_MASTER_URI=http://localhost:11311
ROS_HOSTNAME=localhost
ROSLISP_PACKAGE_DIRECTORIES=/opt/ros/rosdesktop_ws/devel_isolated/xacro/share/common-lisp
ROS_DISTRO=indigo
ROS_ETC_DIR=/opt/ros/rosdesktop_ws/install_isolated/etc/ros
```

- **ROS_ROOT:=**where ROS is installed
- **ROS_PACKAGE_PATH:=**where ROS looks for packages/dependencies to build/run
- **ROS_MASTER_URI:=**the ROS master (remember, ROS is a centralized model)
- **ROS_HOSTNAME:=**the name of the local machine
- **ROS_DISTRO:=**the version or ROS being used

Supported Languages

- **C/C++**
  - Uses GNU gcc by default
  - Supports Clang and CUDA compilers
  - Supports C++11
- **Python**
  - Python 2 is stable
  - Python 3 is allegedly somewhat unstable
- Lisp *Don't use this. No, really, please don't.*
- Java *useful for running ROS on Android (and illustrates why Java will never replace C++)*
- Javascript *new; unadvised*
- Ruby *deprecated*
- C#/.NET *external support*

ROS Package Structure



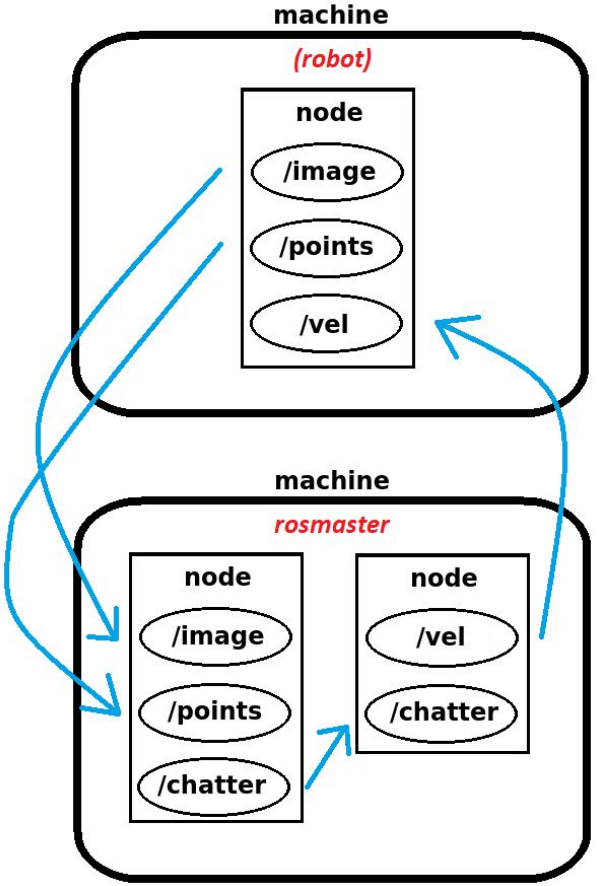
Data Communication

Rostopic

- Rostopic → XMLRPC → TCP/IP
- Message-passing system
- Includes set of tools
  - rostopic list — *Lists all topics published/subscribed*
  - rostopic info [topic-name] — *Displays information regarding a specific topic*
  - rostopic hz [topic-name] — *Displays the topic update rate in hertz*
  - etc.

Rostopic

Method of data communication across distributed systems



Nodes and Nodelets

Node

- Basically a process **which uses an internet protocol to subscribe/publish**
- Can contain multiple threads
- Can render a UI with its “main” thread
- Can contain multiple subscribers and/or publishers

Nodelet

- A node, but only has **local machine** access -- **uses shared memory to subscribe/publish**
- Provides faster data transfer

Publisher

- Transmits data
- Has a FIFO message queue
- Is an asynchronous call
- User must specify:
  - Data type to publish
  - Topic (label)
  - Queue size

Publisher (contd.)

- What will happen here? *(assume publishing one message takes 2 seconds)*

```
pub = nh.advertise<james_msgs::LargeMsg>("/slow", 10);
...
...
while(count < 100)
{
    pub.publish(msg); //msg is of type james_msgs::LargeMsg
    count++;
}
```

Subscriber

- Receives data
- Has a FIFO message queue
- Triggers a callback function
- Message queue query is does not block to wait for data
- Callback function is run on main thread *(by default)*
- User must specify:
  - Data type to publish
  - Topic (label)
  - Queue size
  - Callback function pointer

Subscriber (contd.)

- Describe the callback behavior
- How will it change if */fast* stops receiving data?

```
sub1 = nh.subscribe<std_msgs::String>("/fast", 1, callback_fast);
sub2 = nh.subscribe<std_msgs::String>("/slow", 1, callback_slow);
...
...
void callback_slow(const std_msgs::String::ConstPtr& msg)
{
    //do something really slow here...
    //...
}

void callback_fast(const std_msgs::String::ConstPtr& msg)
{
    ;//do nothing
}
```

Message

- *.msg files are formatted text files which describe the fields of a ROS message
- ROS can use these to generate custom source code for these structures
- Custom CMake functions tell ROS which of these to compile

ROS msg file:

```
Header header
 string child_frame_id
 geometry_msgs/PoseWithCovariance pose
 geometry_msgs/TwistWithCovariance twist
```

In the CMakeLists.txt file:

```
add_message_files(
  FILES
  GeoCoorSys.msg

  Telemetry.msg
)

...

generate_messages(
   DEPENDENCIES
   std_msgs
 )
```

Message (contd.)

The command “rosmsg show” [message-type] displays the .msg type information

```
$ rosmsg show [message-type]

[beginner_tutorials/Num]:
int64 num
```

Service

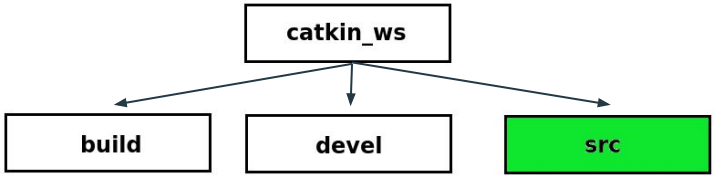
- *.srv files are similar to msg files, expect they have two parts:
  - Request
  - Response
- rossrv show [service-type] works similar to rosmsg show

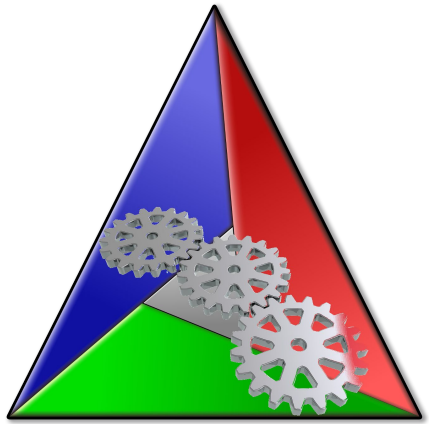
```
int64 A
int64 B
---
int64 Sum
```

Build System

Catkin

- Requires C++11
- CMake + environment and filesystem data
- Catkin → CMake → Make → a.out
- Recursively searches under [your-catkin-workspace]/src





Running Executables

Rosrun

- Runs an executable/script
- Execute “rosrun” from your ROS workspace (e.g. catkin_ws)
- “Knows” where to find your files
    - If it fails to find your exe/script, source the ROS environment; if it’s a python script, make sure it has execute permissions
- Rosrun looks recursively under your package root folder for scripts, e.g. ~/catkin_ws/src/chatter_bot

Running Executables (contd.)

Roslaunch

- Runs one or more executables/scripts, using your *.launch configuration file
- XML syntax
- Allows specific failure behaviors, etc.
- Implicitly starts a roscore, if none are running

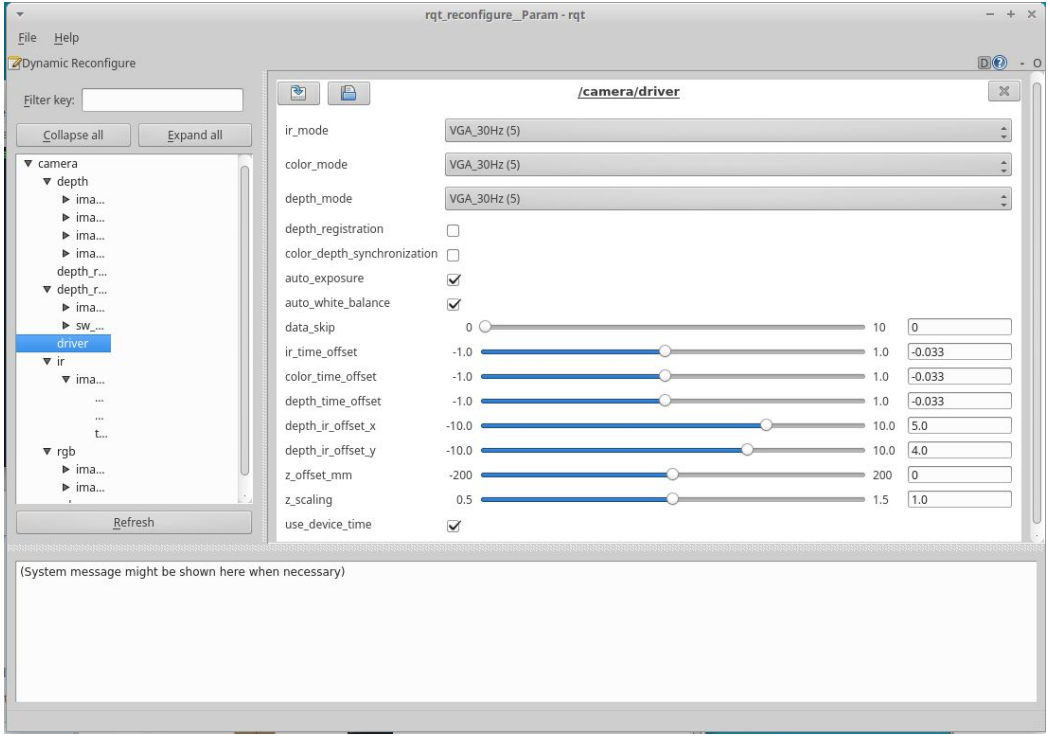
Tools

- rqt_*
- RViz
- Gazibo
- Stage
- Rosbag

Rqt_* Tools

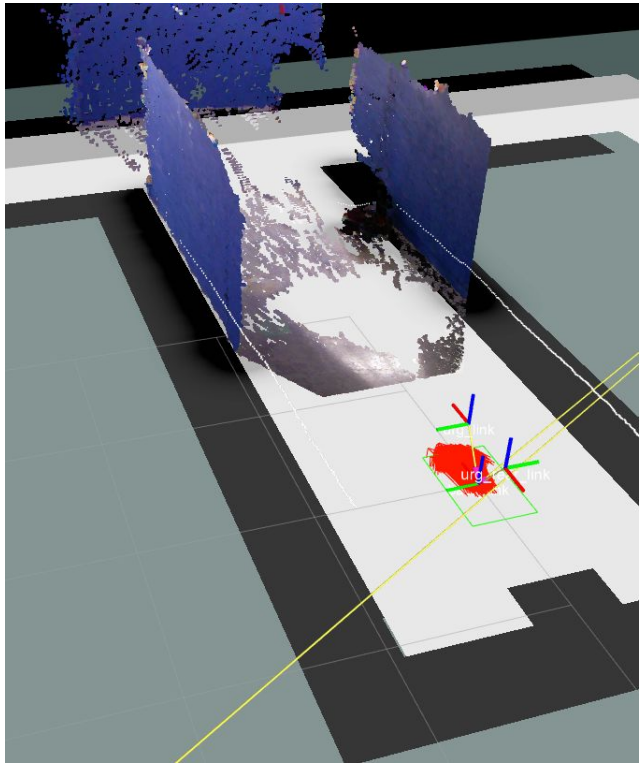
UI tools for performing various tasks:

- rqt_reconfigure
- rqt_image_view
- rqt_plot
- rqt_graph
- rqt_bag



RViz

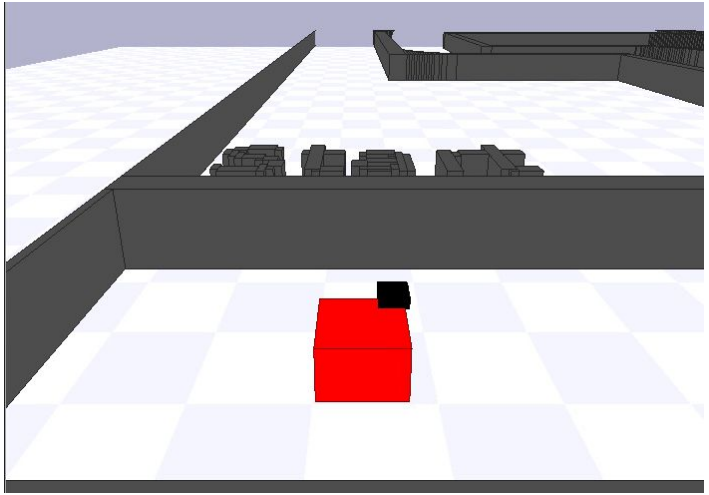
- Data visualization tool
- A must-have for visualizing real-time data from a real robot
- Can import pre-made *.rviz configuration files
  - Default location: ~/home/[your-username]/default.rviz
- https://www.youtube.com/watch?v=H4e16pptt1U



Stage

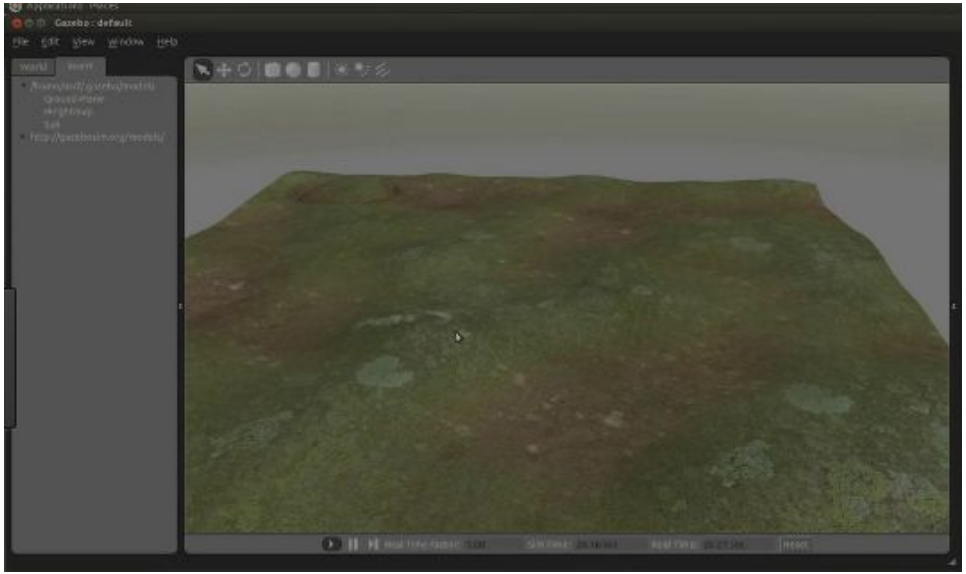
“All the world’s a stage, And all the men and women merely players” --Shakespeare

- 2.5D robot simulator
- Reliable
- Little overhead



Gazibo

- 3D robot simulator
- More features than stage, but more complexity and overhead



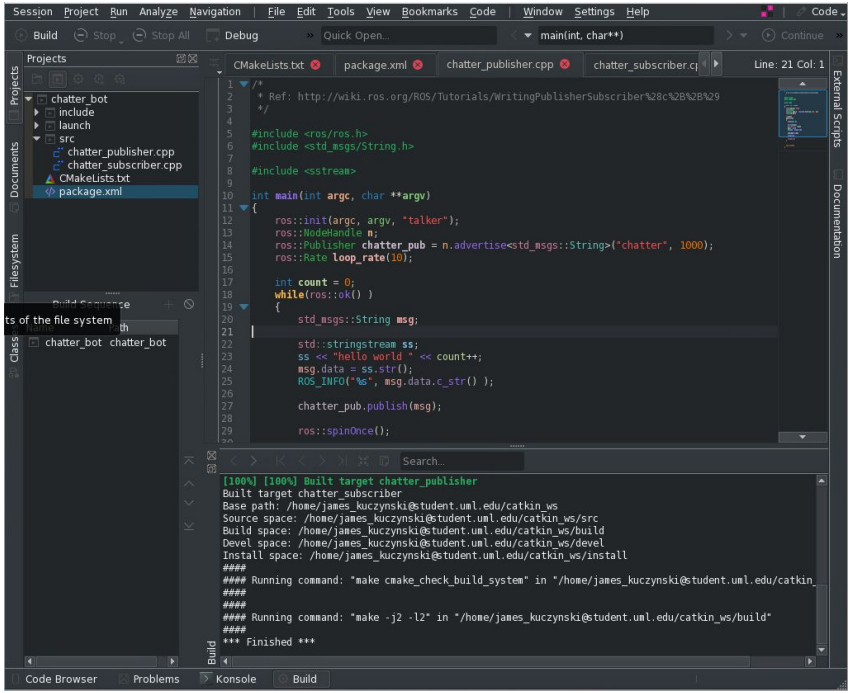
Rosbag

- Rosbag record: records rostopic data
- Rosbag play: plays back prerecorded rostopic data from a *.bag file
- Rqt_bag is the (optional) RQT UI frontend

Text Editors/IDEs

My personal experience

- **ViM/Emacs** **(yes)**
- **CLion** **(yes)**
- **KDevelop** **(yes)**
- QtCreator (yes)
- Codeblocks (sort of)
- Eclipse (soft of)
- NetBeans (allegedly)
- ...and more: http://wiki.ros.org/IDEs



Common Problems

- **Q**: *My program segfaults, but it totally isn't my fault!*
- **A**: Yes it is, but it might not be your code's fault, but a library version clash. ROS installs its own dependencies (OpenCV, PCL, Eigen, etc.); double-check your CMakeLists.txt file
- **Q**: *RViz keeps crashing!*
- **A**: Probably a race condition.
- **Q**: *Catkin can't find my node!*
- **A**: Run "source devel/setup.bash".
- **Q**: *I can't communicate with my robot (or my robot isn't moving)!*
- **A**: Check ROS_MASTER_URI, ROS_HOSTNAME, and ROS_IP; one of these might not be set correctly.

Common Problems (contd.)

- **Q**: *Roscore won't stop running, and [CTRL]+\ isn't working!*
- **A**: Run killall -9 roscore "*This is an offer the **program** can't refuse*" -- Prof. Moloney
- **Q**: *move_base isn't using the current yaml parameters!*
- **A**: restart all ros nodes, including the roscore.
- **Q**: *I'm getting a AF_INET error!*
- **A**: Check ROS_MASTER_URI, ROS_HOSTNAME, and ROS_IP; one of these is probably wrong.
- **Q**: *I've discovered artificial general intelligence, and am being forced to clean the floor for my Roomba overload!*
- **A**: Sorry, can't help you with that one.

Common Problems (contd.)

- **Package clashing**

/usr/lib[...] *OR* /usr/local/lib

/opt/ros/[...]

~/home/james/[...]/CxxLibraries/[...]

Don’t install external libraries unless you really need them

- **CMake “random” version selection**

/usr/lib/pcl1.8

/opt/ros/.../pcl1.7

Useful Resources

- https://github.com/ros/cheatsheet/releases
- https://www.cse.sc.edu/~jokane/agitr/
- http://wiki.ros.org/ROS/Tutorials
- http://wiki.ros.org/Books

Babbling incoherently in response to an undergrad's question, the grad student is alarmed to watch the class write everything down.

Getting Started

UMass Lowell VLabs

- https://vlabs.uml.edu
- Provides access to CentOS virtual machines with ROS Indigo preinstalled
- See the tutorial for information on using it
- *Backup your data (e.g. to a **private** GitHub repository)*
    - *“The virtual machine ate my homework!”*

.bashrc

- Use it to set environment variables
- Is sourced when you open a terminal

```
125 export QT5_DIR=/opt/Qt/5.9/gcc_64/lib/cmake/Qt5
126 export QT_QMAKE_EXECUTABLE=/opt/Qt/5.9/gcc_64/bin/qmake
127
128 export ROS_OS_OVERRIDE=ubuntu:16.04
129
130 export JAVA_HOME=/home/johndoe/jdk1.8.0_111
131 export PATH=$PATH:/home/johndoe/jdk1.8.0_111
132
133 export ROS_MASTER_URI=http://machine-name:11311
134 export ROS_HOSTNAME=10.10.10.114
135
136 source /opt/ros/indigo/setup.bash
```

Creating a ROS Workspace

```
1  james@scarlet-alpha: ~$ source /opt/ros/rosdesktop_ws/devel_isolated/setup.bash
2  james@scarlet-alpha: ~$ mkdir -p catkin_ws/src
3  james@scarlet-alpha: ~$ cd catkin_ws/src/
4  james@scarlet-alpha: ~/catkin_ws/src$ catkin_init_workspace
5  Creating symlink "/home/james@scarlet-alpha/catkin_ws/src/CMakeLists.txt" pointing to "/opt/ros/rosdesktop_ws/
   install_isolated/share/catkin/cmake/toplevel.cmake"
6  james@scarlet-alpha: ~/catkin_ws/src$ cd ..
7  james@scarlet-alpha: ~/catkin_ws$ catkin_make
8  ...
9  ...
10 ...
11 james@scarlet-alpha: ~/catkin_ws$ ls
12 build  devel  src
13 james@scarlet-alpha: ~/catkin_ws$ echo $ROS_PACKAGE_PATH
14 /opt/ros/rosdesktop_ws/src/xacro:/opt/ros/rosdesktop_ws/install_isolated/share:/opt/ros/rosdesktop_ws/
   install_isolated/stacks
15 james@scarlet-alpha: ~/catkin_ws$ source devel/setup.bash
16 james@scarlet-alpha: ~/catkin_ws$ echo $ROS_PACKAGE_PATH
17 /home/james@scarlet-alpha/catkin_ws/src:/opt/ros/rosdesktop_ws/src/xacro:/opt/ros/rosdesktop_ws/
   install_isolated/share:/opt/ros/rosdesktop_ws/install_isolated/stacks
```

Creating a ROS Workspace (contd.)

```
james@scarlet-alpha: ~/catkin_ws$ roscore
... logging to /home/james@scarlet-alpha/.ros/log/bec932.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://localhost:96787/
ros_comm version 1.11.21


SUMMARY
========
```

Creating a ROS Workspace (contd.)

```
PARAMETERS
 * /rosdistro: indigo
 * /rosversion: 1.11.21

NODES

auto-starting new master
process[master]: started with pid [14323]
ROS_MASTER_URI=http://localhost:11311/

setting /run_id to bec932e2-123a-23e1-9e2f-0123398f97e
process[rosout-1]: started with pid [12898]
started core service [/rosout]
```

Creating a ROS Package

Do **NOT** create a package this way!

```
james@scarlet-alpha: ~/catkin_ws/src$ catkin_create_pkg chatter_bot roscpp rospy std_msgs
Traceback (most recent call last):
  File "/usr/bin/catkin_create_pkg", line 75, in <module>
    main()
  File "/usr/bin/catkin_create_pkg", line 63, in main
    boost_comps=args.boost_comps)
  File "/usr/lib/python2.7/site-packages/catkin_pkg/package_templates.py", line 142, in
_create_package_template
    urls=[])
  File "/usr/lib/python2.7/site-packages/catkin_pkg/package_templates.py", line 55, in __init__
    self.validate()
  File "/usr/lib/python2.7/site-packages/catkin_pkg/package.py", line 244, in validate
    raise InvalidPackage('\n'.join(errors))
catkin_pkg.package.InvalidPackage: Invalid email "james_kuczynski@student.uml.edu@todo.todo" for
person "james_kuczynski@student.uml.edu"
```

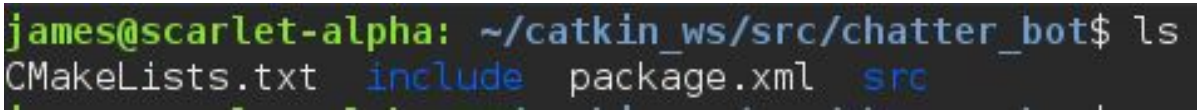
Creating a ROS Package (contd.)

Use this method

```
james@scarlet-alpha: ~/catkin_ws/src$ catkin_create_pkg -m James chatter_bot roscpp rospy std_msgs
Created file chatter_bot/CMakeLists.txt
Created file chatter_bot/package.xml
Created folder chatter_bot/include/chatter_bot
Created folder chatter_bot/src
Successfully created files in /home/james_kuczynski@student.uml.edu/catkin_ws/src/chatter_bot.
Please adjust the values in package.xml.
```

Creating a ROS Package (contd.)

- **package.xml**: Internal ROS dependencies
- **CMakeLists.txt**: CMake file
- **src**: C++ Source files and python scripts
- **include**: Headers



CMakeLists.txt

- Generates makefiles
- Generates msg and srv
- Code
- ROS provides custom macros

```
cmake_minimum_required(VERSION 2.8.3)
project(chatter_bot)

find_package(catkin REQUIRED COMPONENTS
  roscpp
  rospy
  std_msgs
)

catkin_package()

include_directories(${catkin_INCLUDE_DIRS})

add_executable(chatter_publisher src/chatter_publisher.cpp)
add_executable(chatter_subscriber src/chatter_subscriber.cpp)

target_link_libraries(chatter_publisher ${catkin_LIBRARIES})
target_link_libraries(chatter_subscriber ${catkin_LIBRARIES})
```

package.xml

- Contains meta information
  - Maintainer name, etc.
- Contains internal dependencies

```xml
<?xml version="1.0"?>
<package>
  <name>chatter_bot</name>
  <version>0.0.0</version>
  <description>The chatter_bot package</description>

  <maintainer email="James@todo.todo">James</maintainer>

  <license>MIT</license>

  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>roscpp</build_depend>
  <build_depend>rospy</build_depend>
  <build_depend>std_msgs</build_depend>
  <run_depend>roscpp</run_depend>
  <run_depend>rospy</run_depend>
  <run_depend>std_msgs</run_depend>

  <export>
    <!-- Other tools can request additional information be placed here -->
  </export>
</package>
```

Publisher.cpp

```cpp
#include <ros/ros.h>
#include <std_msgs/String.h>

#include <sstream>

int main(int argc, char **argv)
{
    ros::init(argc, argv, "talker");
    ros::NodeHandle n;
    ros::Publisher chatter_pub = n.advertise<std_msgs::String>("chatter", 1000);
    ros::Rate loop_rate(10);

    int count = 0;
    while(ros::ok() )
    {
        std_msgs::String msg;

        std::stringstream ss;
        ss << "hello world " << count++;
        msg.data = ss.str();
        ROS_INFO("%s", msg.data.c_str() );

        chatter_pub.publish(msg);

        ros::spinOnce();

        loop_rate.sleep();
    }


    return EXIT_SUCCESS;
}
```

Publisher.py

```python
#!/usr/bin/env python

import rospy
from std_msgs.msg import String

def talker():
    pub = rospy.Publisher('chatter', String, queue_size=10)
    rospy.init_node('talker', anonymous=True)
    rate = rospy.Rate(10) # 10hz
    while not rospy.is_shutdown():
        hello_str = "hello world %s" % rospy.get_time()
        rospy.loginfo(hello_str)
        pub.publish(hello_str)
        rate.sleep()

if __name__ == '__main__':
    try:
        talker()
    except rospy.ROSInterruptException:
        pass
```

Publisher.lsp

This is included merely to show RosLisp exists. Please do <u>not</u> use it for homework assignments!!!

```
(in-package :roslisp-tutorials-basics)

(defun talker ()
  "Periodically print a string message on the /chatter topic"
  (with-ros-node ("talker")
    (let ((i 0) (pub (advertise "chatter" "std_msgs/String")))
      (loop-at-most-every .1
          (publish-msg pub :data (format nil "foo ~a" (incf i)))))))
```

Subscriber.cpp

```cpp
#include <ros/ros.h>
#include <std_msgs/String.h>


void chatterCallback(const std_msgs::String::ConstPtr& msg)
{
  ROS_INFO("I heard: [%s]", msg->data.c_str() );
}


int main(int argc, char **argv)
{
    ros::init(argc, argv, "listener");
    ros::NodeHandle n;
    ros::Subscriber sub = n.subscribe("chatter", 1000, chatterCallback);


    ros::spin();

    return 0;
}
```

Subscriber.py

```python
#!/usr/bin/env python

import rospy
from std_msgs.msg import String

def callback(data):
    rospy.loginfo(rospy.get_caller_id() + "I heard %s", data.data)

def listener():
    rospy.init_node('listener', anonymous=True)

    rospy.Subscriber("chatter", String, callback)

    # spin() simply keeps python from exiting until this node is stopped
    rospy.spin()

if __name__ == '__main__':
    listener()
```

Subscriber.lsp

This is included merely to show RosLisp exists. Please do not use it for homework assignments!!!

```
(in-package :roslisp-tutorials-basics)

(defun listener ()
  (with-ros-node ("listener" :spin t)
    (subscribe "chatter" "std_msgs/String" #'print)))
```

beginner_tutorials.launch

```
<!-- :set filetype=xml -->

<launch>
    <node name="chatter_subscriber" pkg="chatter_bot" type="chatter_subscriber" output="screen" required="true"/>
    <node name="chatter_publisher"  pkg="chatter_bot" type="chatter_publisher" output="screen" required="true"/>
</launch>

<!-- respawn;=false by default-->
<!-- can also specify parameters, inline YAML, configuration files, etc. -->
```

Build

Run “catkin_make”

```
james@scarlet-alpha: ~/catkin_ws$ catkin_make
Base path: /home/james_kuczynski@student.uml.edu/catkin_ws
Source space: /home/james_kuczynski@student.uml.edu/catkin_ws/src
Build space: /home/james_kuczynski@student.uml.edu/catkin_ws/build
Devel space: /home/james_kuczynski@student.uml.edu/catkin_ws/devel
Install space: /home/james_kuczynski@student.uml.edu/catkin_ws/install
####
#### Running command: "make cmake_check_build_system" in "/home/james_k
####
####
#### Running command: "make -j2 -l2" in "/home/james_kuczynski@student.
####
Scanning dependencies of target chatter_publisher
Scanning dependencies of target chatter_subscriber
[100%] Building CXX object chatter_bot/CMakeFiles/chatter_subscriber.di
[100%] Building CXX object chatter_bot/CMakeFiles/chatter_publisher.dir
Linking CXX executable chatter_publisher
Linking CXX executable chatter_subscriber
[100%] Built target chatter_publisher
[100%] Built target chatter_subscriber
james@scarlet-alpha: ~/catkin_ws$
```

Run (C/C++)

Run Individual Nodes

- roscore
- rosrun chatter_bot chatter_subscriber
- rosrun chatter_bot chatter_publisher

Using Roslaunch

- Roslaunch chatter_bot chatter_bot.launch

Run (Python)

Run Individual Nodes

- roscore
- rosrun chatter_bot chatter_subscriber.py
- rosrun chatter_bot chatter_publisher.py

Using Roslaunch

- Roslaunch chatter_bot chatter_bot.launch

Make sure to set executable permissions, otherwise:

```
james@scarlet-alpha: ~/catkin_ws$ rosrun chatter_bot chatter_subscriber.py
[rosrun] Couldn't find executable named chatter_subscriber.py below /home/j
[rosrun] Found the following, but they're either not files,
[rosrun] or not executable:
[rosrun]   /home/james_kuczynski@student.uml.edu/catkin_ws/src/chatter_bot/
```

Race Conditions

- What happens when this program runs?
- Is the behavior predictable?

```
pcl::PointCloud g_cloud;
sensor_msgs::LaserScan g_scan;

void callback_pcd(const pcl::PointCloud& cloud)
{
    g_cloud = cloud;
}

void callback_ldr(const sensor_msgs::LaserScan& scan)
{
    g_scan = scan;
    fuseData();
}

void fuseData()
{
    pcl::PointCloud newCloud = g_scan.toPcd() + g_cloud;
}
```

Supplemental Exercises

Core ROS Tutorials 1.1 Beginner Level http://wiki.ros.org/ROS/Tutorials

uml_race

uml_race

- Autonomous navigation of a robot with a LiDAR unit using Stage
- You will be subscribing to the sensor (/robot/base_scan) and publishing to the motors (/robot/move_base)
- *(hint)* Print sensor_msgs::Twist messages to get sensor info
  http://docs.ros.org/api/sensor_msgs/html/msg/LaserScan.html
- Objective is to successfully traverse the race course as fast as possible
  - Teleportation is not allowed
  - “Referee” node sets maximum velocity allowed, so algorithm efficiency is important
  - No recovery behaviors are provided (or should be needed)
- *time* > 60s := ok 60 > *time* < 55 := good *time* < 55 := very good

uml_race

- Go to https://github.com/uml-robotics/uml_race and clone or download the repository
- Have fun :)