Robot Programming with Lisp

7. Lisp Packaging and Introduction to ROS

Gayane Kazhoyan

Institute for Artificial Intelligence
University of Bremen

November 29th, 2018
Outline

Lisp Packages and ASDF Systems
   Lisp Packages
   ASDF Systems

Robot Operating System
   What is a Robot?
   ROS Overview
   ROS Communication Layer
   ROS Build System
   Programming with ROS

Organizational Info
Lisp Packages

Lisp packages define namespaces. They are used to avoid naming clashes and control access permissions.

CL-USER> (defun lambda () #\L)
Lock on package COMMON-LISP violated when proclaiming LAMBDA as ...
CL-USER> (defpackage :i-want-my-own-lambda)
CL-USER> (in-package :i-want-my-own-lambda)
#<COMMON-LISP:PACKAGE "I-WANT-MY-OWN-LAMBDA">
I-WANT-MY-OWN-LAMBDA> (common-lisp:defun lambda () #\L)
LAMBDA
I-WANT-MY-OWN-LAMBDA> (common-lisp:in-package :cl-user)
#<PACKAGE "COMMON-LISP-USER">
CL-USER> (describe *)
#<PACKAGE "COMMON-LISP-USER">
Documentation:
  public: the default package for user code and data
Nicknames: CL-USER
Use-list: COMMON-LISP, SB-ALIEN, SB-DEBUG, SB-EXT, SB-GRAY, SB-PROFILE

Lisp Packages and ASDF Systems
Robot Operating System
Organizational

Gayane Kazhoyan
November 29th, 2018
Lisp Packages [2]
Defining a Package

defpackage defined-package-name [[option]] => package

option::= (:nicknames nickname*)* |
          (:documentation string) |
          (:use package-name*)* |
          (:shadow symbol-name*)* |
          (:shadowing-import-from package-name symbol-name*)* |
          (:import-from package-name symbol-name*)* |
          (:export symbol-name*)* |
          (:intern symbol-name*)* |
          (:size integer)
Lisp Packages [3]

Example Package Definition

```
CL-USER> (defpackage :homework
       (:nicknames :hw)
       (:documentation "A namespace for my homework assignments")
       (:use :common-lisp))

#<PACKAGE "HOMEWORK">
CL-USER> (in-package :homework)

#<PACKAGE "HOMEWORK">
HW> (defun say-hello () (print "hello"))
HW> (say-hello)
"hello"
HW> (in-package :common-lisp-user)

#<PACKAGE "COMMON-LISP-USER">
CL-USER> (say-hello)
The function COMMON-LISP-USER::SAY-HELLO is undefined.
CL-USER> (hw:say-hello)
The symbol "SAY-HELLO" is not external in the HOMEWORK package.
CL-USER> (hw::say-hello)
"hello"
```

Lisp Packages and ASDF Systems

Robot Operating System

Organizational

Gayane Kazhoyan

November 29th, 2018
Symbol Namespaces

symbol-package

CL-USER> (in-package "HOMEWORK")
#<PACKAGE "HOMEWORK">
HW> (describe 'say-hello)
HOMEWORK::SAY-HELLO
HW> (describe 'defun)
COMMON-LISP:DEFUN
HW> (describe :hello)
:HELLO
HW> (symbol-package 'say-hello)
#<PACKAGE "HOMEWORK">
HW> (symbol-package :hello)
#<PACKAGE "KEYWORD">
HW> (eql ':hello :hello)
T
HW> keyword:hello
:HELLO
HW> (eql :hello keyword:hello)
T

Lisp Packages and ASDF Systems  Robot Operating System  Organizational
Symbol Namespaces [2]

Uninterned symbols, find-package, intern

HW> '#:hello
#:HELLO
HW> (symbol-package '#:hello)
NIL
HW> (eql '#:hello '#:hello)
NIL
HW> (gensym)
#:G1008
HW> (find-package :homework)
#<PACKAGE "HOMEWORK">
HW> (intern "HELLO" (find-package :homework))
HELLO
NIL
HW> (describe 'hello)
HOMEWORK::HELLO
HW> (loop for i from 1 to 5 
collect (intern (format nil "NAME-~a" i)))
(NAME-1 NAME-2 NAME-3 NAME-4 NAME-5)
Outline

Lisp Packages and ASDF Systems
  Lisp Packages
  ASDF Systems

Robot Operating System
  What is a Robot?
  ROS Overview
  ROS Communication Layer
  ROS Build System
  Programming with ROS

Organizational Info
ASDF System Definition

(in-package :cl-user)
(asdf:defsystem my-system
   :name "My Super-Duper System"
   :description "My Super-Duper System is for doing cool stuff."
   :long-description "Here's how it does cool stuff: ..."
   :version "0.1"
   :author "First Last <email@bla.bla>"
   :licence "BSD"
   :depends-on (alexandria and-another-system)
   :components ((:file "package")))
ASDF Systems [2]

ASDF keeps a *registry* of all the paths where it expects to find `.asd` files. A registry is a list of paths.

There are different types of registries: for users, for administrators, etc. But the simplest is to work with the `*central-registry*`.

Managing the Registry

```
CL-USER> asdf:*central-registry*
(#P"/some/path/
  #P"/some/other/path/")
CL-USER> (push "~/path/to/dir/of/my-system/" asdf:*central-registry*)
("~/path/to/dir/of/my-system/
  #P"/some/path/
  #P"/some/other/path/")
CL-USER> (asdf:load-system :my-system)
T
```

The trailing slash is important ("/some/path/")!
Outline

Lisp Packages and ASDF Systems
  Lisp Packages
  ASDF Systems

Robot Operating System
  What is a Robot?
  ROS Overview
  ROS Communication Layer
  ROS Build System
  Programming with ROS

Organizational Info

Gayane Kazhoyan
November 29th, 2018
Industrial Robots

Logistics

Automotive

• Extremely heavy, precise and dangerous, not really smart
• Mostly no sensors, only high-precision motor encoders
• Programmable through PLCs (using block diagrams or Pascal / Basic like languages)

Image courtesy: BIBA

Image courtesy: Mercedes Benz Bremen
Industrial Light-weight Robots

Production: 

Production: 

Medicine: 

Copyright: Universal Robots

Copyright: Universal Robots

Copyright: Intuitive Surgical

Copyright: Intuitive Surgical

Copyright: KUKA Roboter GmbH

Copyright: KUKA Roboter GmbH

- Very precise, moderately dangerous, somewhat smart
- High-precision motor encoders, sometimes force sensors, cameras
- Native programming and simulation tools (C++, Java, Python, GUIs)
Service Robots

Autonomous aircrafts

- Usually not very precise
- Not really dangerous
- Usually cognition-enabled
- Equipped with lots of sensors
- Usually running a Linux

Mobile platforms

 Manipulation platforms

 Courtesy DJI

 Humanoids

 Courtesy NASA/JPL-Caltech

Lisp Packages and ASDF Systems

Robot Operating System

Organizational

Gayane Kazhoyan
November 29th, 2018

Robot Programming with Lisp
Service Robots with Light-weight Arms

DLR Justin

Rosie / Boxy

• Moderately precise and dangerous
• Cognition-enabled
• Equipped with lots of sensors
• Usually running a combination of a real-time and non real-time OS.

Courtey of DLR
Outline

Lisp Packages and ASDF Systems
  Lisp Packages
  ASDF Systems

Robot Operating System
  What is a Robot?
  ROS Overview
    ROS Communication Layer
    ROS Build System
  Programming with ROS

Organizational
  Info
Motivation

• Numerous different robotics labs, each with their own robot platforms, different operating systems and programming languages but similar software and hardware modules for most of them.
Motivation

- Numerous different robotics labs, each with their own robot platforms, different operating systems and programming languages but similar software and hardware modules for most of them.
- Each lab reinventing the wheel for their platforms.
Motivation

• Numerous different robotics labs, each with their own robot platforms, different operating systems and programming languages but similar software and hardware modules for most of them.

• Each lab reinventing the wheel for their platforms.

• **Idea:** provide a unified software framework for everyone to work with.

Requirements:
Motivation

• Numerous different robotics labs, each with their own robot platforms, different operating systems and programming languages but similar software and hardware modules for most of them.

• Each lab reinventing the wheel for their platforms.

• Idea: provide a unified software framework for everyone to work with.

Requirements:
  – Support for different programming languages
Motivation

• Numerous different robotics labs, each with their own robot platforms, different operating systems and programming languages but similar software and hardware modules for most of them.

• Each lab reinventing the wheel for their platforms.

• **Idea**: provide a unified software framework for everyone to work with.

  **Requirements:**
  
  – Support for different programming languages
  – Different operating systems
Motivation

• Numerous different robotics labs, each with their own robot platforms, different operating systems and programming languages but similar software and hardware modules for most of them.
• Each lab reinventing the wheel for their platforms.
• Idea: provide a unified software framework for everyone to work with.

Requirements:
– Support for different programming languages
– Different operating systems
– Distributed processing over multiple computers / robots
Motivation

- Numerous different robotics labs, each with their own robot platforms, different operating systems and programming languages but similar software and hardware modules for most of them.
- Each lab reinventing the wheel for their platforms.
- **Idea**: provide a unified software framework for everyone to work with.

Requirements:
- Support for different programming languages
- Different operating systems
- Distributed processing over multiple computers / robots
- Easy software sharing mechanisms
At 2007 Willow Garage, a company founded by an early Google employee Scott Hassan at 2006 in the Silicon Valley, starts working on their Personal Robotics project and ROS.
Robot Operating System [2]

ROS core components:

- Meta-Operating System for programming robotics software (configuring, starting / stopping, logging etc. software components)
- Middleware for communication of the components of a robotic system (distributed inter-process / inter-machine communication)
- A collection of packaging / build system tools with a strong focus on integration and documentation
- Language-independent architecture (C++, Python, Lisp, Java, JavaScript, ...)

ROS core software developed and maintained by OSRF and some externals.
In addition, developed by the ROS community:

- hardware drivers
- libraries (PCL, OpenCV, TF, ...)
- capabilities (navigation, manipulation, control, ...)
- applications (fetching beer, making popcorn, ...)

Robot Operating System [3]
## ROS Community

From the community report:

<table>
<thead>
<tr>
<th>Location</th>
<th>Visitors</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>United States</td>
<td>34,710</td>
<td>16.86%</td>
</tr>
<tr>
<td>China</td>
<td>31,946</td>
<td>15.75%</td>
</tr>
<tr>
<td>Japan</td>
<td>15,518</td>
<td>7.63%</td>
</tr>
<tr>
<td>Germany</td>
<td>12,711</td>
<td>6.30%</td>
</tr>
<tr>
<td>India</td>
<td>8,400</td>
<td>4.06%</td>
</tr>
<tr>
<td>Philippines</td>
<td>7,235</td>
<td>3.49%</td>
</tr>
<tr>
<td>South Korea</td>
<td>6,790</td>
<td>3.19%</td>
</tr>
<tr>
<td>United Kingdom</td>
<td>4,325</td>
<td>2.03%</td>
</tr>
<tr>
<td>Taiwan</td>
<td>4,233</td>
<td>2.02%</td>
</tr>
<tr>
<td>France</td>
<td>3,723</td>
<td>1.74%</td>
</tr>
<tr>
<td>Canada</td>
<td>3,354</td>
<td>1.58%</td>
</tr>
<tr>
<td>Spain</td>
<td>2,955</td>
<td>1.38%</td>
</tr>
<tr>
<td>Singapore</td>
<td>2,842</td>
<td>1.35%</td>
</tr>
<tr>
<td>Italy</td>
<td>2,744</td>
<td>1.28%</td>
</tr>
<tr>
<td>Russia</td>
<td>2,455</td>
<td>1.16%</td>
</tr>
<tr>
<td>Indonesia</td>
<td>2,451</td>
<td>1.16%</td>
</tr>
<tr>
<td>Australia</td>
<td>2,436</td>
<td>1.15%</td>
</tr>
<tr>
<td>Brazil</td>
<td>2,231</td>
<td>1.07%</td>
</tr>
<tr>
<td>Hong Kong</td>
<td>2,147</td>
<td>1.03%</td>
</tr>
<tr>
<td>Turkey</td>
<td>1,928</td>
<td>0.90%</td>
</tr>
<tr>
<td>Netherlands</td>
<td>1,511</td>
<td>0.72%</td>
</tr>
<tr>
<td>Thailand</td>
<td>1,437</td>
<td>0.68%</td>
</tr>
<tr>
<td>Poland</td>
<td>1,335</td>
<td>0.64%</td>
</tr>
<tr>
<td>Switzerland</td>
<td>1,242</td>
<td>0.59%</td>
</tr>
<tr>
<td>Vietnam</td>
<td>1,125</td>
<td>0.53%</td>
</tr>
</tbody>
</table>

wiki.ros.org visitor locations:

Source: Google Analytics
Site: wiki.ros.org in July 2018
Some robots supporting ROS (data from November 2014):

- Fraunhofer IPA Care-O-bot
- Aldebaran Nao
- Willow Garage PR2
- Merlin miroBotPro
- Clearpath Robotics Husky
- Gostai Jazz
- Neobotix mpa-700
- Neobotix mpa-500
- ROS-Industrial
- Robotnik Summit
- Adept MobileRobots Pioneer family (P3DX, P3AT, PioneerLB, PioneerInd, AmigoBot, Seekur, Seekur Jr.)
- Adept MobileRobots Seekur family (Seekur, Seekur Jr.)
- Adept MobileRobots Pioneer LX
- Otto Bock SensorHand Speed
- Robonaut 2
- Robotnik Modular Arm
- Denso VS9000
- PAL Robotics REEM-C
- Kinaova JACO
- LiZi
- Nav2
- Robotnik Agus
- Kinga MICO

Lisp Packages and ASDF Systems

Robot Operating System

Organizational
Outline

Lisp Packages and ASDF Systems
  Lisp Packages
  ASDF Systems

Robot Operating System
  What is a Robot?
  ROS Overview
  ROS Communication Layer
  ROS Build System
  Programming with ROS

Organizational Info
Robotic software components

Robot PCs
- Camera
- Motor Drivers
- Controllers
- Image Processing

Vision PC
- Navigation System

AI PC
- Knowledge Base
- Planning System

Lisp Packages and ASDF Systems
Robot Operating System
Organizational

Gayane Kazhoyan
November 29th, 2018
Robotic software components

→ Processes distributed all over the place.

Lisp Packages and ASDF Systems
Robot Operating System
Organizational

Gayane Kazhoyan
November 29th, 2018
Connecting Pieces Together [2]
Connecting Pieces Together [2]

- Camera
- Direct Connection
- Motor Drivers
- Knowledge Base
- Planning System
- Navigation System
- Controllers
- Image Processing

Lisp Packages and ASDF Systems
Robot Operating System
Organizational

Gayane Kazhoyan
November 29th, 2018

Robot Programming with Lisp
34
Connecting Pieces Together [2]
Distributed Hosts

Robot

- roscore
- Controllers
- Camera
- Motor Drivers
- Navigation System

Network Bus (TCP/IP)

Host B
- Image Processing

Host A
- Knowledge Base
- Planning System
roscore

- **ROS master**
  - A centralized XML-RPC server
  - Negotiates communication connections
  - Registers and looks up names of participant components

- **Parameter Server**
  - Stores persistent configuration parameters and other arbitrary data

- **rosout**
  - Distributed stdout
Terminology

- **Nodes** are processes that produce and consume data
- **Parameters** are persistent data stored on parameter server, e.g. configuration and initialization settings

Node communication means:

- **Topics**: asynchronous many-to-many “streams-like”
  - Strongly-typed (ROS .msg spec)
  - Can have one or more *publishers*
  - Can have one or more *subscribers*

- **Services**: synchronous blocking one-to-many “function-call-like”
  - Strongly-typed (ROS .srv spec)
  - Can have only one *server*
  - Can have one or more *clients*

- **Actions**: asynchronous non-blocking one-to-many “function-call-like”
  - Built on top of topics but can be canceled
Establishing Communication

```lisp
(ros "master")
(camera
  viewer)
```
Establishing Communication

```lisp
advertise("images")

camera

ros "master"

viewer
```
Establishing Communication

```
ros
"master"
topic:images
```

camera

viewer
Establishing Communication

```lisp
(subscribe "images")

(ros "master"
  topic:images)
```

camera

viewer
Establishing Communication

camera

subscribe("images")

viewer

ros "master"

topic:images
Establishing Communication

```
ros "master"
topic:images

images(tcp)
camera
viewer
```
Establishing Communication

```
ros
"master"

images(tcp)

camera

viewers

publish(img)
```
Establishing Communication

```
ros "master"
```

```
topic:images
```

```
camera
```

```
images(tcp)
```

```
viewer
```

```
publish(img)
```
Establishing Communication

camera --> images(tcp) --> viewer

ros "master"

publish(img)

viewer too
Establishing Communication

Lisp Packages and ASDF Systems

Robot Operating System

Organizational

Gayane Kazhoyan
November 29th, 2018

Robot Programming with Lisp

48
Establishing Communication

![Diagram showing communication setup]

- `ros "master"
- `topic:images`
- `images(tcp)`
- `camera`
- `viewer`
- `publish(img)`
- `viewer_too`
ROS Graph

- Starting the core:
  
  ```
  $ roscore
  ```

- Starting a node:
  
  ```
  $ rosrun turtlesim turtlesim_node
  ```

- Starting another node:
  
  ```
  $ rosrun turtlesim turtle_teleop_key
  ```

- Examining the ROS Graph:
  
  ```
  $ rqt_graph
  ```
Tools

- **rosnode**: gives the user information about a node
  
  $ rosnode -h
  
  cleanup, info, kill, list, machine, ping

- **rostopic**: gives publishers, subscribes to the topic, datarate, the actual data
  
  bw, echo, find, hz, info, list, pub, type

- **rosservice**: enables a user to call a ROS Service from the command line
  
  call, find, list, type, uri

- **rosmsg**: gives information about message types
  
  list, md5, package, packages, show

- **rossrv**: same as above for service types
  
  list, md5, package, packages, show

- **roswtf**: diagnoses problems with a ROS network
Outline

Lisp Packages and ASDF Systems
  Lisp Packages
  ASDF Systems

Robot Operating System
  What is a Robot?
  ROS Overview
  ROS Communication Layer
  ROS Build System
  Programming with ROS

Organizational Info
Packages and Metapackages

- *Packages* are a named collection of software that is built and treated as an atomic dependency in the ROS build system.

- *Metapackages* are dummy “virtual” packages that reference one or more related packages which are loosely grouped together. Similar to Debian packages.

Actually released through the Debian packaging system.
ROS Workspace

Packages are stored in ROS workspaces:

```
$ roscd
```

Workspaces have a specific structure:

```
build
  devel
  install
  src
  CMakeLists.txt
  ros_package_1
  metapack_repo_1
    metapackage_1
    ros_package_2
    ros_package_3
```

```
CMakeLists.txt
package.xml
asdf-system.asd
src
```
Managing Packages

• Creating a package:
  $ ros cd && cd src/lisp_course_material
  $ catkin_create_pkg assignment_6 roslisp turtlesim geometry_msgs

• Compiling a package:
  $ ros cd && catkin_make

• Moving through ROS workspaces:
  $ ros cd assignment_6

Naming convention: underscores (no CamelCase, no-dashes)!

All the packages in your workspace are one huge CMake project.

→ Multiple workspaces chained together.
<?xml version="1.0"?>
<package>
  <name>assignment_6</name>
  <version>0.0.0</version>
  <description>The assignment_6 package</description>
  <maintainer email="kazhoyan@cs.uni-bremen.de">Gaya</maintainer>
  <license>Public domain</license>
  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>geometry_msgs</build_depend>
  <build_depend>roslisp</build_depend>
  <build_depend>turtlesim</build_depend>
  <run_depend>geometry_msgs</run_depend>
  <run_depend>roslisp</run_depend>
  <run_depend>turtlesim</run_depend>
</package>
CMakeLists

```plaintext
assignment_6/CMakeLists.txt

cmake_minimum_required(VERSION 2.8.3)
project(assignment_6)
find_package(catkin REQUIRED COMPONENTS
   roslisp
   geometry_msgs
)
catkin_package(
   CATKIN_DEPENDS roslisp geometry_msgs
)
```

Lisp Packages and ASDF Systems  Robot Operating System  Organizational

Gayane Kazhoyan

November 29th, 2018
Launch Files
Automated Starting, Stopping and Configuring the Nodes

XML files for launching nodes:

• automatically set parameters and start nodes with a single file
• hierarchically compose collections of launch files
• automatically re-spawn nodes if they crash
• change node names, namespaces, topics, and other resource names
• without recompiling
• easily distribute nodes across multiple machines
Launch Files [2]
Automated Starting, Stopping and Configuring the Nodes

Example

```xml
<launch>
  <!-- Starting nodes-->
  <node pkg="turtlesim" type="turtlesim_node" name="sim"/>
  <node pkg="turtlesim" type="turtle_teleop_key" name="teleop"
       output="screen"/>

  <!-- Setting parameters -->
  <param name="some_value" type="double" value="2.0"/>
</launch>
```

Using the launch file:

```
$ roslaunch package_name launch_file_name
```
Outline

Lisp Packages and ASDF Systems
- Lisp Packages
- ASDF Systems

Robot Operating System
- What is a Robot?
- ROS Overview
- ROS Communication Layer
- ROS Build System
- Programming with ROS

Organizational Info
ROS API

ROS API provides the programmer with means to

• start ROS node processes
• generate messages
• publish and subscribe to topics
• start service servers
• send service requests
• provide and query action services
• find ROS packages
• ...

ROS APIs: roscpp, rospy, rosjava, rosjs, roslib
Links

• ROS documentation
  http://wiki.ros.org/

• ROS community support
  http://answers.ros.org/questions/
Outline

Lisp Packages and ASDF Systems
  Lisp Packages
  ASDF Systems

Robot Operating System
  What is a Robot?
  ROS Overview
  ROS Communication Layer
  ROS Build System
  Programming with ROS

Organizational Info
Organizational Info

• Tutorial link:

• Grades: 5 points for this assignment
• Due: 05.12, 23:59 AM German time
• Next class: 06.12, 14:15
Thanks for your attention!

Special thanks to Lorenz Mösenlechner and Jan Winkler for providing illustrations!