Robot Programming with Lisp

1. Introduction, Setup

Arthur Niedzwiecki

Institute for Artificial Intelligence
University of Bremen

21st October, 2021

General Info

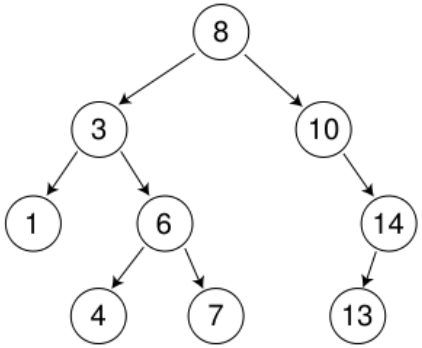
- Lecturer: Arthur (PhD student at IAI)
- Tutor: Vanessa (PhD student at IAI)
- Correspondence: `aniedz@cs.uni-bremen.de`, `hassouna@uni-bremen.de`
- Dates: Thursdays, 14:15 - 15:45, 16:15 - 17:45
- Language: English and German
- Credits: 6 ECTS (4 SWS)
- Course type: practical course
- Course number: 03-BE-710.98b
- Location: TAB Building, Room 0.36 EG

Plan

Course content

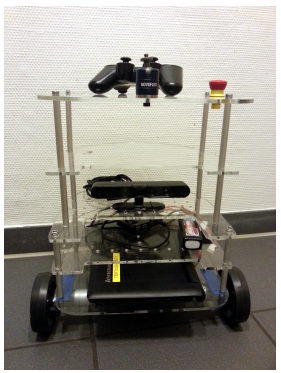
Common Lisp

Artificial Intelligence



Robot Operating System (ROS)

Robot platform



Common Lisp

- Full-featured industry-standard programming language

Common Lisp

- Full-featured industry-standard programming language
- Means for functional programming
- Means for imperative programming
- Means for OOP

Common Lisp

- Full-featured industry-standard programming language
- Means for functional programming
- Means for imperative programming
- Means for OOP
- Fast prototyping through read-eval-print loop and dynamic typing

Common Lisp

- Full-featured industry-standard programming language
- Means for functional programming
- Means for imperative programming
- Means for OOP
- Fast prototyping through read-eval-print loop and dynamic typing
- Compiles into machine code

Common Lisp

- Full-featured industry-standard programming language
- Means for functional programming
- Means for imperative programming
- Means for OOP
- Fast prototyping through read-eval-print loop and dynamic typing
- Compiles into machine code
- Best choice for symbolic processing (AI, theorem proving, etc.)

Common Lisp

- Full-featured industry-standard programming language
- Means for functional programming
- Means for imperative programming
- Means for OOP
- Fast prototyping through read-eval-print loop and dynamic typing
- Compiles into machine code
- Best choice for symbolic processing (AI, theorem proving, etc.)
- Good choice for writing domain-specific programming languages (e.g., robot programming languages)

Common Lisp

- Full-featured industry-standard programming language
- Means for functional programming
- Means for imperative programming
- Means for OOP
- Fast prototyping through read-eval-print loop and dynamic typing
- Compiles into machine code
- Best choice for symbolic processing (AI, theorem proving, etc.)
- Good choice for writing domain-specific programming languages (e.g., robot programming languages)

Applications using / written in dialects of Lisp:
Emacs, AutoCAD, Grammarly, Mirai (Gollum animation), Google ITA (airplane ticket price planner AI), DART (DARPA logistics AI), Maxima (computer algebra system), AI frameworks, NASA satellites ...

ROS

- Middleware for communication of the components of a robotic system

ROS

- Middleware for communication of the components of a robotic system
- "Meta-Operating System" for programming robotics software (configuring, starting / stopping, logging etc. software components)

ROS

- Middleware for communication of the components of a robotic system
- "Meta-Operating System" for programming robotics software (configuring, starting / stopping, logging etc. software components)
- Powerful build system (based on CMake), with a strong focus on integration and documentation

ROS

- Middleware for communication of the components of a robotic system
- "Meta-Operating System" for programming robotics software (configuring, starting / stopping, logging etc. software components)
- Powerful build system (based on CMake), with a strong focus on integration and documentation
- Language-independent architecture: C++, Python, Lisp and more

ROS

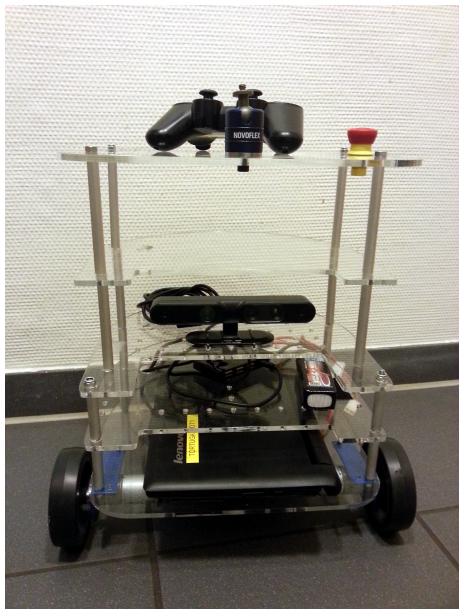
- Middleware for communication of the components of a robotic system
- "Meta-Operating System" for programming robotics software (configuring, starting / stopping, logging etc. software components)
- Powerful build system (based on CMake), with a strong focus on integration and documentation
- Language-independent architecture: C++, Python, Lisp and more
- According to ROS 2020 Community Metrics Report,
  - More than 2 million unique pageviews `wiki.ros.org` a month
  - More than 38 million downloads of `.deb` packages a month

ROS

- Middleware for communication of the components of a robotic system
- "Meta-Operating System" for programming robotics software (configuring, starting / stopping, logging etc. software components)
- Powerful build system (based on CMake), with a strong focus on integration and documentation
- Language-independent architecture: C++, Python, Lisp and more
- According to ROS 2020 Community Metrics Report,
  - More than 2 million unique pageviews `wiki.ros.org` a month
  - More than 38 million downloads of `.deb` packages a month
- *De facto* standard in modern robotics

TortugaBot

- 2 controllable wheels
- 2D laser scanner
- Thinkpad E485 PC with bluetooth
- PlayStation joystick



Why Lisp for robots?

- ROS supports a number of languages

Why Lisp for robots?

- ROS supports a number of languages
- Lisp is good for rapid prototyping

Why Lisp for robots?

- ROS supports a number of languages
- Lisp is good for rapid prototyping
- It is more suitable for symbolic reasoning and AI

Why Lisp for robots?

- ROS supports a number of languages
- Lisp is good for rapid prototyping
- It is more suitable for symbolic reasoning and AI
- There are existing robot programming languages in Lisp that automate decision making

Rough schedule

Assignments (single, this year)

- Introduction & Setup
- Lisp basics
- OOP & Failure Handling
- Functional programming
- Search Algorithms

Rough schedule

Assignments (single, this year)

- Introduction & Setup
- Lisp basics
- OOP & Failure Handling
- Functional programming
- Search Algorithms

Intermediate (until mid Jan '22)

- ROS Lisp API (*roslisp*)
- 2D world of *turtlesim*
- Coordinate frames of *TF*

Rough schedule

Assignments (single, this year)

- Introduction & Setup
- Lisp basics
- OOP & Failure Handling
- Functional programming
- Search Algorithms

Intermediate (until mid Jan '22)

- ROS Lisp API (*roslisp*)
- 2D world of *turtlesim*
- Coordinate frames of *TF*

Project (groups, Jan-Feb '22)

- Controlling TortugaBot
- Reading sensor data
- Collision avoidance
- Heuristic decision-making
- The big day: competition

Software requirements

Bringing a *personal laptop* is encouraged.

OS:	Ubuntu 18.04 (or any Linux with bootstrapped 18.04)
IDE:	Emacs 24+
Version control:	Git
Packaging system:	ROS
Lisp software:	SBCL compiler, ASDF build system, Emacs plugin for Common Lisp

Bottom line

You will learn / improve your skills in the following:

- Linux
- Git
- Emacs
- Functional programming
- Common Lisp, of course
- ROS (for future roboticists)

...and get to play with a real little robot!

Plan

Grading

- Course final grade: 100 points = 50 homework + 50 group project.

Grading

- Course final grade: 100 points = 50 homework + 50 group project.
- To participate in the project you need at least 25 points from the homeworks, otherwise it's a fail.

Grading

- Course final grade: 100 points = 50 homework + 50 group project.
- To participate in the project you need at least 25 points from the homeworks, otherwise it's a fail.
- Final grade: 50 of 100 points - 4.0, 100 of 100 points - 1.0.

Grading

- Course final grade: 100 points = 50 homework + 50 group project.
- To participate in the project you need at least 25 points from the homeworks, otherwise it's a fail.
- Final grade: 50 of 100 points - 4.0, 100 of 100 points - 1.0.
- $Grade = \frac{(100 - P_{your})}{(100 - 50)} * 3 + 1$

Homework assignments

- Homework assignments will mostly consist of filling in the missing gaps in already existing code.

Homework assignments

- Homework assignments will mostly consist of filling in the missing gaps in already existing code.
- That code will be hosted on GitLab.

Homework assignments

- Homework assignments will mostly consist of filling in the missing gaps in already existing code.
- That code will be hosted on GitLab.
- The code you write should be uploaded to GitLab (`https://gitlab.informatik.uni-bremen.de/`).

Homework assignments

- Homework assignments will mostly consist of filling in the missing gaps in already existing code.
- That code will be hosted on GitLab.
- The code you write should be uploaded to GitLab (`https://gitlab.informatik.uni-bremen.de/`).
- Homework is due in one week.

Homework assignments

- Homework assignments will mostly consist of filling in the missing gaps in already existing code.
- That code will be hosted on GitLab.
- The code you write should be uploaded to GitLab (`https://gitlab.informatik.uni-bremen.de/`).
- Homework is due in one week.
- Solutions are discussed in the tutorial.

Homework assignments

- Homework assignments will mostly consist of filling in the missing gaps in already existing code.
- That code will be hosted on GitLab.
- The code you write should be uploaded to GitLab (`https://gitlab.informatik.uni-bremen.de/`).
- Homework is due in one week.
- Solutions are discussed in the tutorial.
- Can get 60 of 50 points in homework (can skip one homework).

Homework assignments

- Homework assignments will mostly consist of filling in the missing gaps in already existing code.
- That code will be hosted on GitLab.
- The code you write should be uploaded to GitLab (`https://gitlab.informatik.uni-bremen.de/`).
- Homework is due in one week.
- Solutions are discussed in the tutorial.
- Can get 60 of 50 points in homework (can skip one homework).
- Bonus points for very good homework solutions.

Links

- Emacs cheat sheet:

  `https://www.gnu.org/software/emacs/refcards/pdf/refcard.pdf`

- Git reference book:

  `http://git-scm.com/book/de`

- Lisp books:

  `http://landoflisp.com/`, `http://www.paulgraham.com/onlisp.html`, `http://www.gigamonkeys.com/book/`

Info summary

Next class:

- Date: 28.10
- Time: 14:15 (14:00 - 14:15 for questions)
- Place: same room (TAB 0.36)

Assignment:

- Due: 27.10, Wednesday, 23:59
- Points: 3 points
- For questions: write an email to Vanessa or Arthur

Plan

Assignment goals

Set up your working environment

Set up your Git repositories

Get comfortable with Emacs

Task 1: Install Ubuntu: General

We need Ubuntu 18.04. Depending on your current system, do this:

- Ubuntu 18.04
  Congrats, go further to Task 2: Install ROS.

Task 1: Install Ubuntu: General

We need Ubuntu 18.04. Depending on your current system, do this:

- Ubuntu 18.04
  Congrats, go further to Task 2: Install ROS.
- Windows
  Install Ubuntu alongside Windows as Dual-Boot
  Or use Ubuntu with WSL in Windows
  `https://jack-kawell.com/2020/06/12/ros-wsl2/` (skip ROS install for now)

Task 1: Install Ubuntu: General

We need Ubuntu 18.04. Depending on your current system, do this:

- Ubuntu 18.04
  Congrats, go further to Task 2: Install ROS.
- Windows
  Install Ubuntu alongside Windows as Dual-Boot
  Or use Ubuntu with WSL in Windows
  `https://jack-kawell.com/2020/06/12/ros-wsl2/` (skip ROS install for now)
- Linux
  Dual-Boot or this, for a smooth solution:
  `https://ai.uni-bremen.de/wiki/infrastructure/schroot-18.04-on-20.04`

Task 1: Install Ubuntu: General

We need Ubuntu 18.04. Depending on your current system, do this:

- Ubuntu 18.04
  Congrats, go further to Task 2: Install ROS.
- Windows
  Install Ubuntu alongside Windows as Dual-Boot
  Or use Ubuntu with WSL in Windows
  `https://jack-kawell.com/2020/06/12/ros-wsl2/` (skip ROS install for now)
- Linux
  Dual-Boot or this, for a smooth solution:
  `https://ai.uni-bremen.de/wiki/infrastructure/schroot-18.04-on-20.04`
- MacOS
  Dual-boot or VM with VirtualBox (if Mac is incompatible)

Task 1: Install Ubuntu: Download

- Check your gear for how old it is.
  Especially new machines don't support 18.04 anymore.

Task 1: Install Ubuntu: Download

- Check your gear for how old it is.
  Especially new machines don't support 18.04 anymore.
- $<$ 2019 like Intel CPU 9th gen and older:
  Download Ubuntu 18.04 installation `.iso` (Bionic Beaver)
  (`ubuntu-18.04.6-desktop-amd64.iso`)
  `https://releases.ubuntu.com/18.04/`

Task 1: Install Ubuntu: Download

- Check your gear for how old it is.
  Especially new machines don't support 18.04 anymore.
- $<$ 2019 like Intel CPU 9th gen and older:
  Download Ubuntu 18.04 installation `.iso` (Bionic Beaver)
  (`ubuntu-18.04.6-desktop-amd64.iso`)
  `https://releases.ubuntu.com/18.04/`
- For newer machines, download 20.04 (Focal Fossa):
  (`ubuntu-20.04.3-desktop-amd64.iso`)
  `https://releases.ubuntu.com/20.04/`

Task 1: Install Ubuntu: Download

- Check your gear for how old it is.
  Especially new machines don't support 18.04 anymore.
- $<$ 2019 like Intel CPU 9th gen and older:
  Download Ubuntu 18.04 installation `.iso` (Bionic Beaver)
  (`ubuntu-18.04.6-desktop-amd64.iso`)
  `https://releases.ubuntu.com/18.04/`
- For newer machines, download 20.04 (Focal Fossa):
  (`ubuntu-20.04.3-desktop-amd64.iso`)
  `https://releases.ubuntu.com/20.04/`
- When in doubt, try 18.04 first, we'll need it anyway.
  Follow the steps on the next slide and do
  'Try out Ubuntu' instead of installing.
  Press the Super-Key (Windows-Key) and search for 'About'.
  If the graphics driver is **not** llvmpipe, your machine supports 18.04.

Task 1: Install Ubuntu: Installation

- Create a boot USB with the `.iso` (or burn a DVD).
  Hint: In Windows use the Universal USB installer:

  `http://www.pendrivelinux.com/universal-usb-installer-easy-as-1-2-3/`

  In Linux, use the Startup Disk Creator or `unetbootin`.

Task 1: Install Ubuntu: Installation

- Create a boot USB with the `.iso` (or burn a DVD).
  Hint: In Windows use the Universal USB installer:

  `http://www.pendrivelinux.com/universal-usb-installer-easy-as-1-2-3/`

  In Linux, use the Startup Disk Creator or `unetbootin`.
- Reboot with the USB plugged in to install Ubuntu.
  Dual boot installation with default settings is a one click thing.

Task 1: Install Ubuntu: For new hardware

- Follow this for new hardware that doesn't support 18.04 anymore.

Task 1: Install Ubuntu: For new hardware

- Follow this for new hardware that doesn't support 18.04 anymore.
- Go to this guide to install 18.04 within your current Linux system:

  `https://ai.uni-bremen.de/wiki/infrastructure/schroot-18.04-on-20.04`

Task 1: Install Ubuntu: For new hardware

- Follow this for new hardware that doesn't support 18.04 anymore.
- Go to this guide to install 18.04 within your current Linux system:

  `https://ai.uni-bremen.de/wiki/infrastructure/schroot-18.04-on-20.04`
- Remeber to stay in the 18.04 'jail' for everything regarding this lecture.

Task 1: Install Ubuntu: FAQ

- *How do I boot from USB / CD?*
  You should enter either "Boot Menu" or "BIOS Menu" during reboot
  `https://www.brynux.com/tutorials/boot-keys`

Task 1: Install Ubuntu: FAQ

- *How do I boot from USB / CD?*
  You should enter either "Boot Menu" or "BIOS Menu" during reboot
  `https://www.brynux.com/tutorials/boot-keys`
- *Windows doesn't let me into "BIOS Menu"!*
  You should restart into the "Boot Options Menu" of your Windows:
  hold down "Shift" while pressing "Restart".

Task 1: Install Ubuntu: FAQ

- *How do I boot from USB / CD?*
  You should enter either "Boot Menu" or "BIOS Menu" during reboot
  `https://www.brynux.com/tutorials/boot-keys`
- *Windows doesn't let me into "BIOS Menu"!*
  You should restart into the "Boot Options Menu" of your Windows:
  hold down "Shift" while pressing "Restart".
- *My BIOS supports UEFI, Ubuntu won't install!*
  It should work but if you can't get it to run turn off the UEFI mode:
  restart into the "Boot Options Menu" of your Windows,
  choose "Troubleshoot", then "UEFI Firmware Settings"

Task 1: Install Ubuntu: FAQ

- *How do I boot from USB / CD?*
  You should enter either "Boot Menu" or "BIOS Menu" during reboot
  `https://www.brynux.com/tutorials/boot-keys`
- *Windows doesn't let me into "BIOS Menu"!*
  You should restart into the "Boot Options Menu" of your Windows:
  hold down "Shift" while pressing "Restart".
- *My BIOS supports UEFI, Ubuntu won't install!*
  It should work but if you can't get it to run turn off the UEFI mode:
  restart into the "Boot Options Menu" of your Windows,
  choose "Troubleshoot", then "UEFI Firmware Settings"
- *It still doesn't work!*
  Write an email to Vanessa or Arthur

Task 2: Install ROS

Consult the official installation instructions for troubleshooting:
`http://wiki.ros.org/melodic/Installation/Ubuntu`
In short, it boils down to executing the following in the terminal (*hint*: to open a fresh terminal press `<Ctrl>+<Alt>+t`):

- Add ROS repositories to your sources list:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu bionic main" > /etc/apt/sources.list.d/ros-latest.list'
```

Task 2: Install ROS

Consult the official installation instructions for troubleshooting:
`http://wiki.ros.org/melodic/Installation/Ubuntu`
In short, it boils down to executing the following in the terminal (*hint*: to open a fresh terminal press `<Ctrl>+<Alt>+t`):

- Add ROS repositories to your sources list:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu bionic main" > /etc/apt/sources.list.d/ros-latest.list'
```

- Add their key to your trusted public keys:

```
sudo apt install curl

curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -
```

Task 2: Install ROS

Consult the official installation instructions for troubleshooting:
`http://wiki.ros.org/melodic/Installation/Ubuntu`
In short, it boils down to executing the following in the terminal (*hint*: to open a fresh terminal press `<Ctrl>+<Alt>+t`):

- Add ROS repositories to your sources list:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu bionic main" > /etc/apt/sources.list.d/ros-latest.list'
```

- Add their key to your trusted public keys:

```
sudo apt install curl

curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -
```

- Update your Debian package index:

```
sudo apt-get update
```

Task 2: Install ROS

Consult the official installation instructions for troubleshooting:
`http://wiki.ros.org/melodic/Installation/Ubuntu`
In short, it boils down to executing the following in the terminal (*hint*: to open a fresh terminal press `<Ctrl>+<Alt>+t`):

- Add ROS repositories to your sources list:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu bionic main" > /etc/apt/sources.list.d/ros-latest.list'
```

- Add their key to your trusted public keys:

```
sudo apt install curl

curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -
```

- Update your Debian package index:

```
sudo apt-get update
```

- The version of ROS distributed with Ubuntu 18.04 is **ROS Melodic**. Install the **desktop** package.

```
sudo apt-get install ros-melodic-desktop
```

Task 2: Install ROS

Consult the official installation instructions for troubleshooting:
`http://wiki.ros.org/melodic/Installation/Ubuntu`
In short, it boils down to executing the following in the terminal (*hint*: to open a fresh terminal press `<Ctrl>+<Alt>+t`):

- Add ROS repositories to your sources list:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu bionic main" > /etc/apt/sources.list.d/ros-latest.list'
```

- Add their key to your trusted public keys:

```
sudo apt install curl

curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -
```

- Update your Debian package index:

```
sudo apt-get update
```

- The version of ROS distributed with Ubuntu 18.04 is **ROS Melodic**. Install the **desktop** package.

```
sudo apt-get install ros-melodic-desktop
```

- Install the workspace management tools:

```
sudo apt install python-rosdep python-rosinstall python-rosinstall-generator python-wstool build-essential
```

Task 3: Setup ROS

Consult the official installation instructions for troubleshooting:
`http://wiki.ros.org/melodic/Installation/Ubuntu`

In short, it boils down to executing the following in the terminal:

- Setup rosdep:

```
sudo rosdep init && rosdep update
```

Task 3: Setup ROS

Consult the official installation instructions for troubleshooting:
`http://wiki.ros.org/melodic/Installation/Ubuntu`

In short, it boils down to executing the following in the terminal:

- Setup rosdep:

  ```
  sudo rosdep init && rosdep update
  ```

- Initialize the ROS environment for this particular terminal:

  ```
  source /opt/ros/melodic/setup.bash
  ```

Task 3: Setup ROS

Consult the official installation instructions for troubleshooting:
`http://wiki.ros.org/melodic/Installation/Ubuntu`

In short, it boils down to executing the following in the terminal:

- Setup rosdep:
  ```
  sudo rosdep init && rosdep update
  ```
- Initialize the ROS environment for this particular terminal:
  ```
  source /opt/ros/melodic/setup.bash
  ```
- Create a directory where the code you'll write will be stored (the name `ros_ws` and the location ~ can be changed):
  ```
  mkdir -p ~/ros_ws/src
  ```

Task 3: Setup ROS

Consult the official installation instructions for troubleshooting:
`http://wiki.ros.org/melodic/Installation/Ubuntu`

In short, it boils down to executing the following in the terminal:

- Setup rosdep:
  ```
  sudo rosdep init && rosdep update
  ```
- Initialize the ROS environment for this particular terminal:
  ```
  source /opt/ros/melodic/setup.bash
  ```
- Create a directory where the code you'll write will be stored (the name `ros_ws` and the location ~ can be changed):
  ```
  mkdir -p ~/ros_ws/src
  ```
- Initialize the workspace:
  ```
  cd ~/ros_ws && catkin_make
  ```

Task 3: Setup ROS

Consult the official installation instructions for troubleshooting:
`http://wiki.ros.org/melodic/Installation/Ubuntu`

In short, it boils down to executing the following in the terminal:

- Setup rosdep:
  ```
  sudo rosdep init && rosdep update
  ```
- Initialize the ROS environment for this particular terminal:
  ```
  source /opt/ros/melodic/setup.bash
  ```
- Create a directory where the code you'll write will be stored (the name `ros_ws` and the location ~ can be changed):
  ```
  mkdir -p ~/ros_ws/src
  ```
- Initialize the workspace:
  ```
  cd ~/ros_ws && catkin_make
  ```
- Update your bash startup script and make sure it worked:
  ```
  echo -e "\n# ROS\nsource $HOME/ros_ws/devel/setup.bash\n" >> ~/.bashrc && tail ~/.bashrc && source ~/.bashrc
  ```

Task 4: Git and GitLab

- Log into university GitLab with your LDAP / TZI account:

  `https://gitlab.informatik.uni-bremen.de/`

Task 4: Git and GitLab

- Log into university GitLab with your LDAP / TZI account:
  `https://gitlab.informatik.uni-bremen.de/`
- Click on "+ New Project", call the project `lisp_course_exercises` and make sure it is private.

Task 4: Git and GitLab

- Log into university GitLab with your LDAP / TZI account:
  `https://gitlab.informatik.uni-bremen.de/`
- Click on "+ New Project", call the project `lisp_course_exercises` and make sure it is private.
- Once created, in "Members" add "Arthur Niedzwiecki" and "Vanessa Hassouna" as collaborators. "Project Access" is master.

Task 4: Git and GitLab

- Log into university GitLab with your LDAP / TZI account:

  `https://gitlab.informatik.uni-bremen.de/`
- Click on "+ New Project", call the project `lisp_course_exercises` and make sure it is private.
- Once created, in "Members" add "Arthur Niedzwiecki" and "Vanessa Hassouna" as collaborators. "Project Access" is master.
- Install Git:

  ```
  sudo apt-get install git
  ```

Task 4: Git and GitLab

- Log into university GitLab with your LDAP / TZI account:

  `https://gitlab.informatik.uni-bremen.de/`
- Click on "+ New Project", call the project `lisp_course_exercises` and make sure it is private.
- Once created, in "Members" add "Arthur Niedzwiecki" and "Vanessa Hassouna" as collaborators. "Project Access" is master.
- Install Git:

  ```
  sudo apt-get install git
  ```
- Download the course material into your ROS workspace:

  ```
  roscd && cd ../src
  git clone https://gitlab.informatik.uni-bremen.de/lisp-course/lisp_course_exercises.git && ll
  ```

Task 4: Git and GitLab

- Log into university GitLab with your LDAP / TZI account:

  `https://gitlab.informatik.uni-bremen.de/`

- Click on "+ New Project", call the project `lisp_course_exercises` and make sure it is private.
- Once created, in "Members" add "Arthur Niedzwiecki" and "Vanessa Hassouna" as collaborators. "Project Access" is master.
- Install Git:

  ```
  sudo apt-get install git
  ```

- Download the course material into your ROS workspace:

  ```
  roscd && cd ../src
  git clone https://gitlab.informatik.uni-bremen.de/lisp-course/lisp_course_exercises.git && ll
  ```

- Define a remote target with the address of your new GitLab repo:

  ```
  cd lisp_course_exercises
  git remote add my-repo https://gitlab.informatik.uni-bremen.de/YOUR_GITLAB_USERNAME/lisp_course_exercises.git
  ```

Task 4: Git and GitLab

- Log into university GitLab with your LDAP / TZI account:

  `https://gitlab.informatik.uni-bremen.de/`

- Click on "+ New Project", call the project `lisp_course_exercises` and make sure it is private.
- Once created, in "Members" add "Arthur Niedzwiecki" and "Vanessa Hassouna" as collaborators. "Project Access" is master.
- Install Git:

  ```
  sudo apt-get install git
  ```

- Download the course material into your ROS workspace:

  ```
  roscd && cd ../src
  git clone https://gitlab.informatik.uni-bremen.de/lisp-course/lisp_course_exercises.git && ll
  ```

- Define a remote target with the address of your new GitLab repo:

  ```
  cd lisp_course_exercises
  git remote add my-repo https://gitlab.informatik.uni-bremen.de/YOUR_GITLAB_USERNAME/lisp_course_exercises.git
  ```

- Upload the files to your new GitLab repo:

  ```
  git push -u my-repo master
  ```

Task 4 (alternative): Git and GitHub

- Create an account on GitHub and get a student discount:

  `https://education.github.com/`

Task 4 (alternative): Git and GitHub

- Create an account on GitHub and get a student discount:
  `https://education.github.com/`
- Click on "Start a project", call the project `lisp_course_exercises`. Once you get student discount, make the project private.

Task 4 (alternative): Git and GitHub

- Create an account on GitHub and get a student discount:
  `https://education.github.com/`
- Click on "Start a project", call the project `lisp_course_exercises`. Once you get student discount, make the project private.
- In project "Settings" → "Collaborators" add "Vanessa Hassouna" and "Arthur Niedzwiecki" as collaborators.

Task 4 (alternative): Git and GitHub

- Create an account on GitHub and get a student discount:

  `https://education.github.com/`
- Click on "Start a project", call the project `lisp_course_exercises`. Once you get student discount, make the project private.
- In project "Settings" → "Collaborators" add "Vanessa Hassouna" and "Arthur Niedzwiecki" as collaborators.
- Install Git:

  ```
  sudo apt-get install git
  ```

Task 4 (alternative): Git and GitHub

- Create an account on GitHub and get a student discount:

  `https://education.github.com/`
- Click on "Start a project", call the project `lisp_course_exercises`. Once you get student discount, make the project private.
- In project "Settings" → "Collaborators" add "Vanessa Hassouna" and "Arthur Niedzwiecki" as collaborators.
- Install Git:

  ```
  sudo apt-get install git
  ```
- Download the course material into your ROS workspace:

  ```
  roscd && cd ../src
  git clone https://github.com/lisp-course/lisp_course_exercises.git && ll
  ```

Task 4 (alternative): Git and GitHub

- Create an account on GitHub and get a student discount:

  `https://education.github.com/`
- Click on "Start a project", call the project `lisp_course_exercises`. Once you get student discount, make the project private.
- In project "Settings" → "Collaborators" add "Vanessa Hassouna" and "Arthur Niedzwiecki" as collaborators.
- Install Git:

  ```
  sudo apt-get install git
  ```
- Download the course material into your ROS workspace:

  ```
  roscd && cd ../src
  git clone https://github.com/lisp-course/lisp_course_exercises.git && ll
  ```
- Define a remote target with the address of your new GitHub repo:

  ```
  cd lisp_course_exercises
  git remote add my-repo https://github.com/YOUR_GITHUB_USERNAME/lisp_course_exercises.git
  ```

Task 4 (alternative): Git and GitHub

- Create an account on GitHub and get a student discount:

  `https://education.github.com/`
- Click on "Start a project", call the project `lisp_course_exercises`. Once you get student discount, make the project private.
- In project "Settings" → "Collaborators" add "Vanessa Hassouna" and "Arthur Niedzwiecki" as collaborators.
- Install Git:

  ```
  sudo apt-get install git
  ```
- Download the course material into your ROS workspace:

  ```
  roscd && cd ../src
  git clone https://github.com/lisp-course/lisp_course_exercises.git && ll
  ```
- Define a remote target with the address of your new GitHub repo:

  ```
  cd lisp_course_exercises
  git remote add my-repo https://github.com/YOUR_GITHUB_USERNAME/lisp_course_exercises.git
  ```
- Upload the files to your new GitHub repo:

  ```
  git push -u my-repo master
  ```

Task 5: Install the IDE

- Install the editor itself (Emacs), the Common Lisp compiler (SBCL), the linker (ASDF) and the Emacs Common Lisp plugin (Slime):

```
sudo apt-get install ros-melodic-roslisp-repl
```

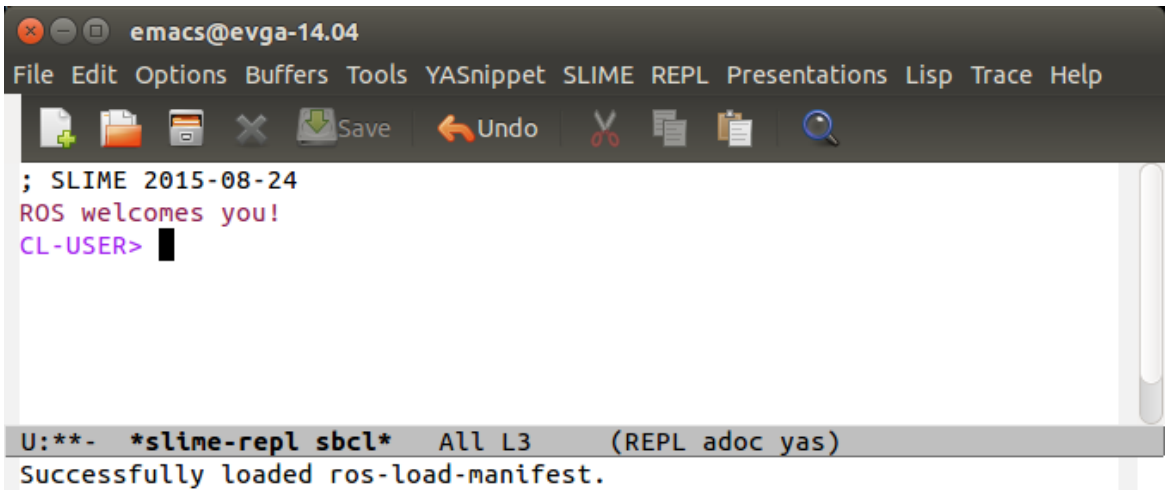
Task 5: Install the IDE

- Install the editor itself (Emacs), the Common Lisp compiler (SBCL), the linker (ASDF) and the Emacs Common Lisp plugin (Slime):

```
sudo apt-get install ros-melodic-roslisp-repl
```

- Start the editor (after compilation is finished you'll see the Lisp shell):

```
roslisp_repl &
```



Task 6: Get familiar with Emacs

The following notation is used in Emacs for keyboard shortcuts:

- `C` for `<Ctrl>`
- `M` for `<Alt>`
- `SPC` for `<Space>`
- `RET` for `<Enter>`
- `-` for when two keys are pressed together (e.g. `C-x` for `<Ctrl>+x`)

The basic shortcuts you will need are listed below:

- `C-x C-f` opens a file
- `C-x 3` or `C-x 2` opens a new tab, `C-x 0` closes it, `C-x 1` maximizes
- `C-x o` switches between tabs
- `C-x b` switches buffers, `C-x C-b` lists all open buffers, `C-x k` kills
- `C-g` cancels a command half-way, `C-x C-c yes` exits Emacs

Task 6: Get familiar with Emacs

The following notation is used in Emacs for keyboard shortcuts:

- `C` for `<Ctrl>`
- `M` for `<Alt>`
- `SPC` for `<Space>`
- `RET` for `<Enter>`
- `-` for when two keys are pressed together (e.g. `C-x` for `<Ctrl>+x`)

The basic shortcuts you will need are listed below:

- `C-x C-f` opens a file
- `C-x 3` or `C-x 2` opens a new tab, `C-x 0` closes it, `C-x 1` maximizes
- `C-x o` switches between tabs
- `C-x b` switches buffers, `C-x C-b` lists all open buffers, `C-x k` kills
- `C-g` cancels a command half-way, `C-x C-c yes` exits Emacs

Open the file with your first assignment and follow the instructions:

`ROS_WORKSPACE/src/lisp_course_exercises/assignment_1/src/orc-battle.lisp`

Task 7: Get familiar with Git

- Once done editing `orc-battle.lisp`, check what's new in your local repo (the one on your hard drive):

```
cd ROS_WORKSPACE/src/lisp_course_exercises && git status
```

Task 7: Get familiar with Git

- Once done editing `orc-battle.lisp`, check what's new in your local repo (the one on your hard drive):

```
cd ROS_WORKSPACE/src/lisp_course_exercises && git status
```

- To see which exactly lines changed ask for the `diff` (q to exit):

```
git diff
```

Task 7: Get familiar with Git

- Once done editing `orc-battle.lisp`, check what's new in your local repo (the one on your hard drive):

```
cd ROS_WORKSPACE/src/lisp_course_exercises && git status
```

- To see which exactly lines changed ask for the `diff` (q to exit):

```
git diff
```

- The red files are the new untracked ones, the green ones are already in the Git index. To add new files to the index use

```
git add .
```

Task 7: Get familiar with Git

- Once done editing `orc-battle.lisp`, check what's new in your local repo (the one on your hard drive):

```
cd ROS_WORKSPACE/src/lisp_course_exercises && git status
```

- To see which exactly lines changed ask for the `diff` (q to exit):

```
git diff
```

- The red files are the new untracked ones, the green ones are already in the Git index. To add new files to the index use

```
git add .
```

- If you deleted some files, to remove them from the index use

```
git add -u
```

Task 7: Get familiar with Git

- Once done editing `orc-battle.lisp`, check what's new in your local repo (the one on your hard drive):

```
cd ROS_WORKSPACE/src/lisp_course_exercises && git status
```

- To see which exactly lines changed ask for the `diff` (*q* to exit):

```
git diff
```

- The red files are the new untracked ones, the green ones are already in the Git index. To add new files to the index use

```
git add .
```

- If you deleted some files, to remove them from the index use

```
git add -u
```

- Once you're sure the changes are final, commit locally:

```
git commit -m "A meaningful commit message."
```

Task 7: Get familiar with Git

- Once done editing `orc-battle.lisp`, check what's new in your local repo (the one on your hard drive):

```
cd ROS_WORKSPACE/src/lisp_course_exercises && git status
```

- To see which exactly lines changed ask for the `diff` (*q* to exit):

```
git diff
```

- The red files are the new untracked ones, the green ones are already in the Git index. To add new files to the index use

```
git add .
```

- If you deleted some files, to remove them from the index use

```
git add -u
```

- Once you're sure the changes are final, commit locally:

```
git commit -m "A meaningful commit message."
```

- Finally, to upload your local commits to the GitLab server, push the changes upstream:

```
git push # or git push my-repo master
```

Q & A

Thanks for your attention!