ENPM702

Introductory Robot Programming

L1: Course Introduction

v2.0

Lecturer: Z. Kootbally

Semester/Year: Summer/2025

MARYLAND APPLIED GRADUATE ENGINEERING

Table of Contents

Changelog

Changelog

- **v2.0**:
  - Move the section on version control to a different slide deck.
  - Added cmake to the package installation command (slide 23)
- **v1.0**: Original version.

Conventions

Conventions

bad practice	👎
best practice	👍
code syntax	</>
example	⚗
exercise	✎
file	🗎
folder	📂
C++ guideline	🚀
note	✍
question	❓
TODO	☑
terminology	📜
warning	🔥
web link	link
package	📦
resource	▦
terminal	`command`

Prerequisites

- ☐ Get the syllabus PDF from Canvas.
- ☐ Get all lecture files from Canvas.

Note

- Slides created with Beamer LaTeX
- Slide decks have a version number.

About Me

- **Contact:** zeidk@umd.edu
- **Office Hours:** By appointment.
- **Affiliation:**
  1. National Institute of Standards and Technology (NIST)
     - Measurement Science for Agile Robotic Systems
       - P2940 - Standard for Measuring Robot Agility
       - Agile Robotics for Industrial Automation Competition (ARIAC)
     - Measurement Science for Automated Vehicles
  2. University of Maryland (UMD)
     - ENPM702: Introductory Robot Programming.
     - ENPM605: Python Applications for Robotics.
     - ENPM818T: Data Storage and Databases.
     - ENPM818Z: On-Road Automated Vehicles.

About You

- Which semester are you currently in?
- Why did you join this course?
- Do you have programming experience?
- Do you have ROS experience?
- What is a fun fact about you?

Course Overview

Operating System

- Ubuntu Desktop 24.04 LTS: Noble Numbat (recommended)
  or
- Ubuntu Desktop 22.04: Jammy Jelly-Fish.

Software

- Visual Studio Code.
- ROS 2:
  - Jazzy Jalisco (for Ubuntu 24.04)
  - Iron Irwini (for Ubuntu 22.04)
- Code documentation: Doxygen.
- Version Control:
  - Git
  - Github (account)
- Valgrind.
- Visual Studio Code extensions.

ToDo

Open **Syllabus_ENPM702_Summer2025.pdf**

Note About Online Students

“In short, the policy states that students who choose the online option may do so because of their work schedule, location, or time zone. For that reason, online section students will participate in the course asynchronously, meaning they will complete the weekly work for the course on their own time. Online students cannot be asked to attend in-person for exams, presentations, or in-class activities for any reason (however this can be offered as an option). Additionally, please allow online students at least 48 hours following the release of the recorded lecture to complete in-class quizzes and assignments.”

Linux Shell

Linux Shell

A shell is a program that provides a command-line interface (CLI) for users to interact with the operating system, allowing them to execute commands, run scripts, manage files, and control processes.

- **Bash** (Bourne Again Shell): The default shell for most Linux distributions, known for its scripting capabilities.
- **Zsh** (Z Shell): Similar to Bash, with additional features like better autocompletion and customization options.
- **Fish** (Friendly Interactive Shell): User-friendly and highly customizable, with a focus on simplicity.
- **Csh** and **Tcsh** (C Shell and Tenex C Shell): Shells with syntax similar to the C programming language, often used in certain Unix environments.
- **Ksh** (Korn Shell): A shell with advanced scripting features, sometimes used in enterprise environments.

Check your current shell with `ps -p $$`

Configuration Files

A shell configuration file is a script that runs each time a new shell session starts. A shell configuration file allows users to customize their shell environment by defining settings, aliases, functions, and environment variables.

- Bash uses **.bashrc**, located in the home directory.
- Zsh uses **.zshrc**, located in the home directory.
- Fish uses **config.fish**, located in **~/.config/fish/**
- Tcsh and Csh use **.cshrc**, located in the home directory.
- Korn uses **.kshrc**, located in the home directory.

In most Unix-like operating systems (including Ubuntu), the tilde ~ symbol refers to the current user's home directory.
`cd ~` takes you to your home directory.

Aliases

Aliases are essentially shortcuts that can save you from having to remember or type long commands.

```
alias identifier='value'
```

- An alias declaration starts with the `alias` keyword, followed by the alias name, an equal sign, and the command to be run for that alias.
- The command should be enclosed in quotes, with **no spaces around the equal sign**.
- Each alias should be declared on a separate line.
- Aliases can be declared anywhere in the configuration file.

Example

```
alias cdd='cd ~/Documents'
alias meminfo='free -m -l -t' # display memory usage
alias weather='curl wttr.in' # display weather information
```

Functions

Functions are reusable blocks of code that allow you to group commands and execute them by calling the function’s name.

```
# Method 1: Using the function keyword
function function_name {
    commands
}

# Method 2: Without the function keyword
function_name() {
    commands
}
```

Examples

```
function greeting {
    echo "Hello $1 $2"
}
```

Call the function: `greeting John Doe`

```
print_arguments() {
    echo "Function or script name: $0"
    echo "Number of arguments: $#"
    echo "All arguments (\$*): $*"
    echo "Each argument separately:"
    for arg in "$@"; do
        echo "$arg"
    done
}
```

Call the function: `print_arguments 1 2 hello world`

Visual Studio Code (VSCode)

Visual Studio Code (VSCode)

- Free code editor that works on Windows, Mac, and Linux.
- Used by development teams at major tech companies including Google (Chrome and Angular) and Netflix (for internal infrastructure tools)
- Consistently ranked as the most popular code editor in developer surveys worldwide.
- Supports virtually every programming language they might learn.
- Makes coding faster and less error-prone with helpful features.
- 36,000 extensions for specific programming languages, 19,000 snippet collections, 13,000 formatters, 13,000 linters, 9,400 debuggers, and 8,700 themes.

Installation

- Download .deb file at https://code.visualstudio.com/download
- `sudo apt install code_1.100.2-1747260578_amd64.deb`

Run

- `code` or run it for the Ubuntu Applications.

Essential Interface

- **Activity Bar** (left side): File explorer, search, source control, extensions.
- **Editor**: Where you will write code.
- **Terminal**: For running commands and programs.
- **Status bar**: Shows file info and problems.

Resources

- keyboard-shortcuts-linux.pdf

Visual Studio Code (VSCode) ▸ Workspace

Workspace

A container that holds a set of related projects or code files. In this course we will use the following workspace structure for C++ lectures.

```
enpm702_summer2025_cpp
└─ lectureX
   ├─ src
   │  └─ *.cpp
   └─ include
      └─ *.hpp
```

ToDo

- Create folder structure for **lecture1**
- Place **lecture1.cpp** in **lecture1/src**
- Open the workspace in VSCode:
  - `cd <workspace>`
  - `code .`

Configuration

- **File** →**Preferences** →**Settings** or ⌨ **Ctrl** + ,
    - Configure VS Code for the current workspace, e.g., change the default font size.
- Changes are stored either in the workspace 🗎 **settings.json** (📂 **.vscode** folder) or in the user 🗎 **settings.json**
    - Workspace 🗎 **settings.json** takes precendence over user 🗎 **settings.json**

.vscode Folder

The **.vscode** folder is a special directory that VSCode creates to store workspace-specific configuration and settings.

- **Purpose and Location:** The **.vscode** folder appears in your project's root directory and contains configuration files that customize VSCode's behavior for that specific workspace. These settings apply only to the current workspace, not globally across all VSCode instances.
- **Key Files Inside .vscode:**
  - **settings.json**: Workspace-specific settings that override user settings.
  - **launch.json**: Debug configurations for your project.
  - **tasks.json**: Automates common development workflows.
  - **extensions.json**: Recommended extensions for the workspace.

ToDo

- Place **extensions.json** in **.vscode**
- **Ctrl + Shift + P**→**Extensions: Show Recommended Extensions**→Click on the cloud icon to install all the recommended extensions.

The Command Palette

The Command Palette displays a searchable list of all available commands in VS Code, including those from installed extensions. Think of it as a universal search box for actions rather than files. The Command Palette is one of VS Code's most powerful features.

```
Ctrl + Shift + P
```

CMake

CMake is an open-source, cross-platform family of tools designed to build, test, and package software.

```
sudo apt install cmake
cmake --version
```

CMakeLists.txt is used by CMake to define the build configuration for a software project. It contains a series of commands that specify how the project should be built, including which source files to compile, which libraries to link against, and other build-related settings.

Resources

- CMakeLists.txt documentation
- Create a CMake hello world project with CMake Quick Start

ToDo

1. `sudo apt install build-essential gdb`
2. Place **CMakeLists.txt** in your workspace
3. **Ctrl + Shift + P → CMake: Set Build Target**
4. **Ctrl + Shift + P → Developer: Reload Window**
   - Ensure **CMake** is in the side bar.
5. **Configure** (do this only once).
6. **Build** (do this only once).
7. From now on, clicking ▶ (at the bottom of the window) will build and run the executable.

Next Class

- Lecture2: Introduction to C++