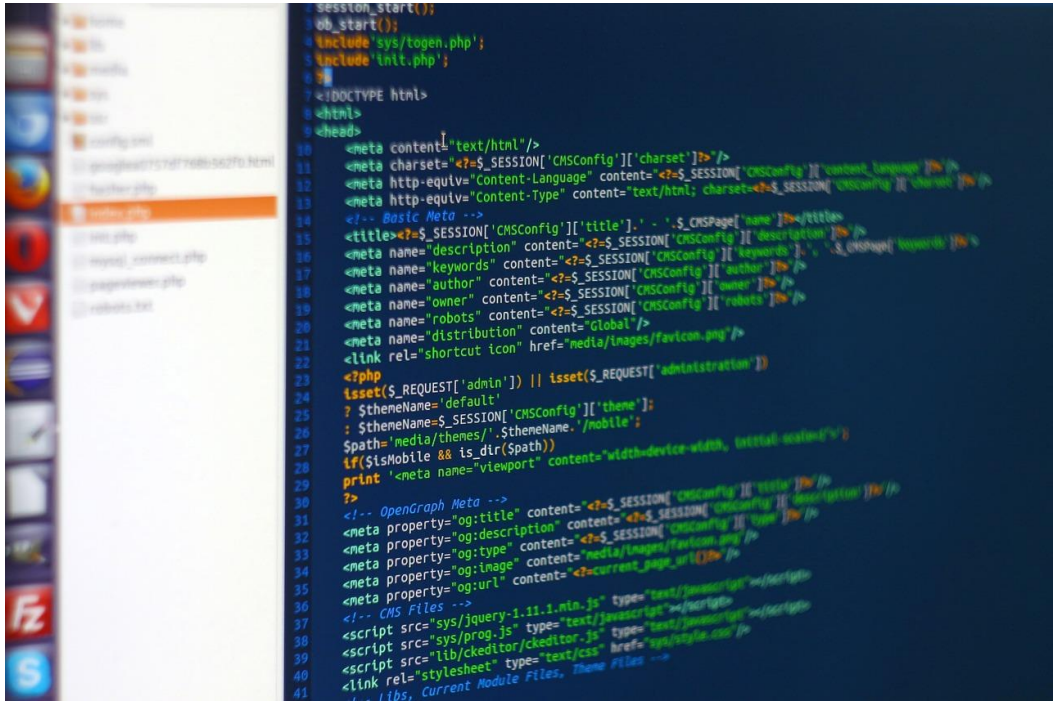
German Research Center for Artificial Intelligence

Robot Programming

Dr. rer. nat. Teena Hassan
M.Sc. Mihaela Popescu
thassan@uni-bremen.de

Prof. Dr. Dr. h.c. Frank Kirchner
Arbeitsgruppe Robotik, Universität Bremen
Robotics Innovation Center, DFKI
https://robotik.dfki-bremen.de/
robotik@dfki.de

October 26, 2022 – Bremen

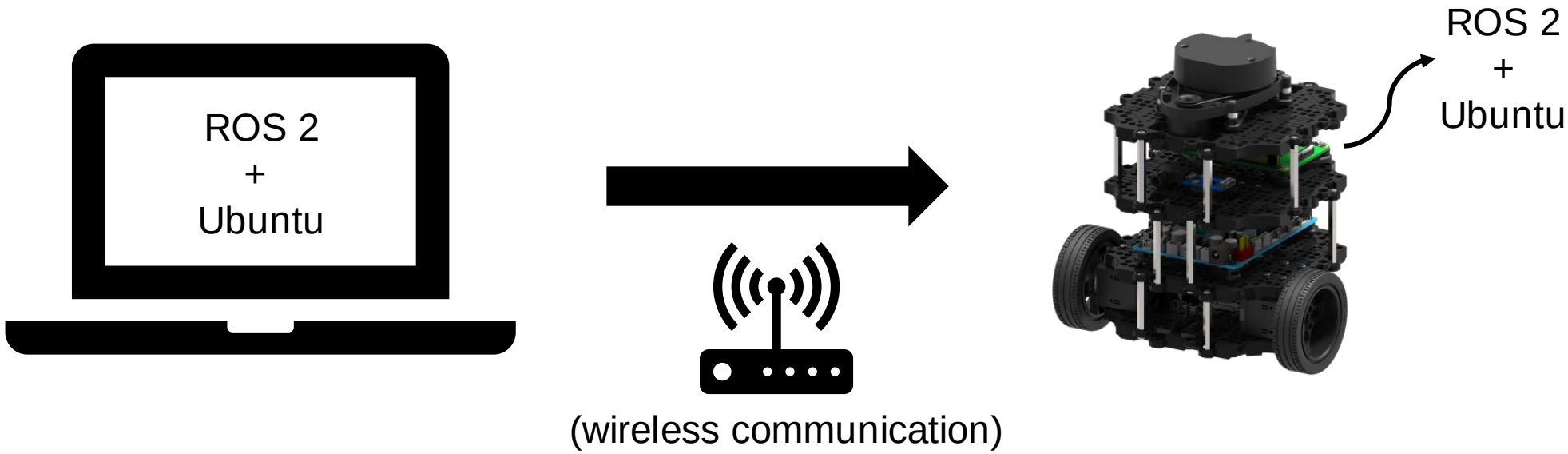


Universität Bremen

From Tutorial -- Teleoperating the TurtleBot

On Your PC
runs a teleoperation program based on ROS 2 which sends out velocity commands.

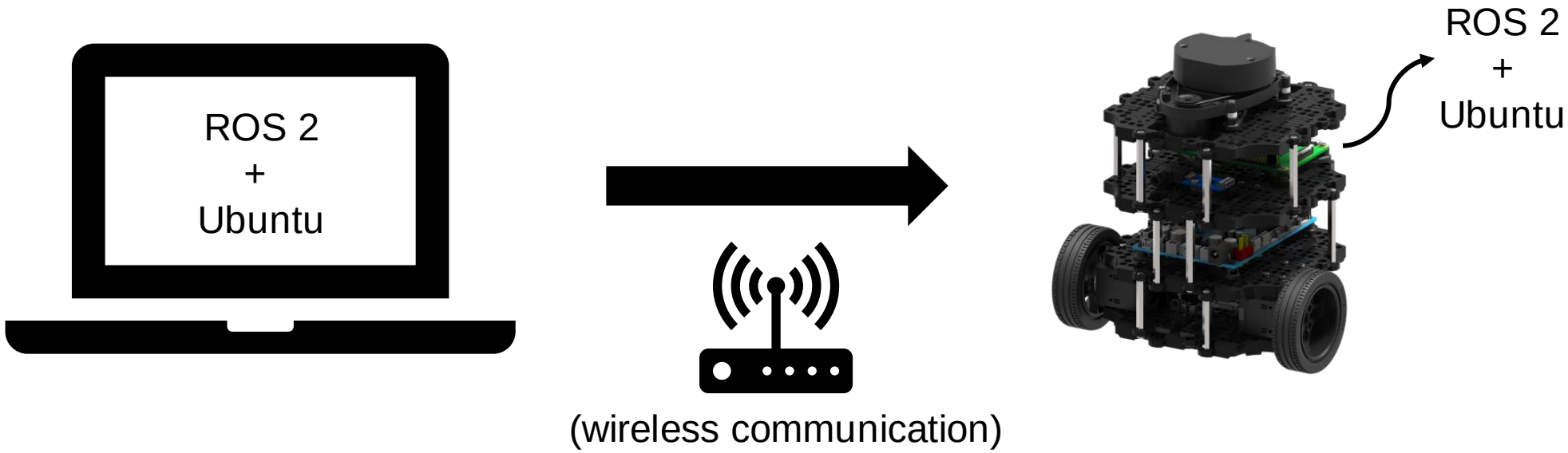
On the robot
runs several programs based on ROS 2 to receive velocity commands, read sensor data and control the wheels of the robot.



Programming a Robot – What Do We Need?

On Your PC
runs a teleoperation program based on ROS 2 which sends out velocity commands.

On the robot
runs several programs based on ROS 2 to receive velocity commands, read sensor data and control the wheels of the robot.



1. Operating System (OS)
2. Robot Operating System (ROS)
3. Programs based on ROS

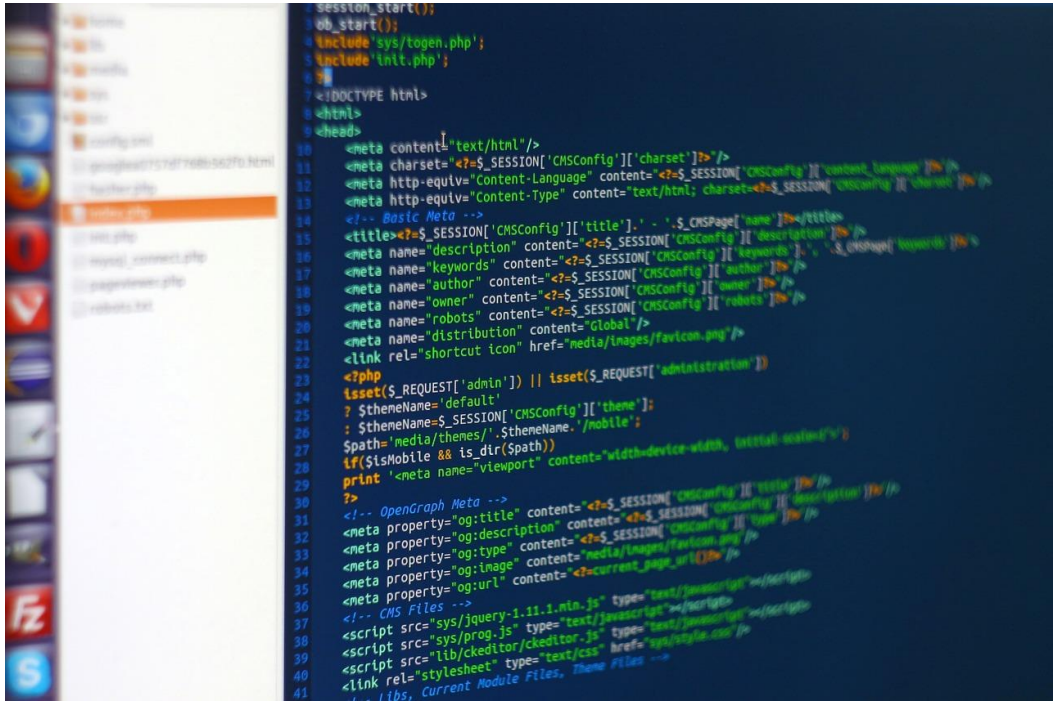
German Research Center for Artificial Intelligence

Part 1: Introduction to Operating System

Dr. rer. nat. Teena Hassan
M.Sc. Mihaela Popescu
thassan@uni-bremen.de

Prof. Dr. Dr. h.c. Frank Kirchner
Arbeitsgruppe Robotik, Universität Bremen
Robotics Innovation Center, DFKI
https://robotik.dfki-bremen.de/
robotik@dfki.de

October 26, 2022 – Bremen



Universität Bremen

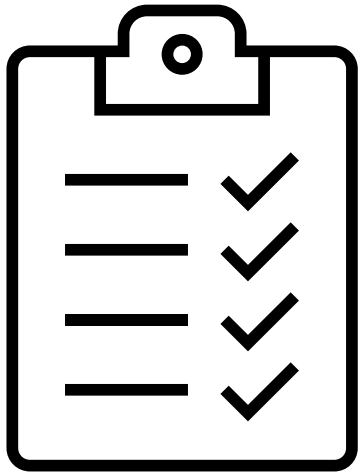
Learning Goal

- At the end of this lecture, you will be able to:
  1. Define an operating system and give examples for it.
  2. List the functions of an operating system.
  3. Understand the purpose of a command line interface.
  4. Give examples for commonly used commands in Ubuntu operating system.
  5. Define a process and describe a method for interprocess communication.
  6. Explain the concept of virtualization.

Personal Computer

- You are starting college and would like to buy yourself a new laptop.

- What aspects would you consider, when you prepare your list of requirements?



Hardware

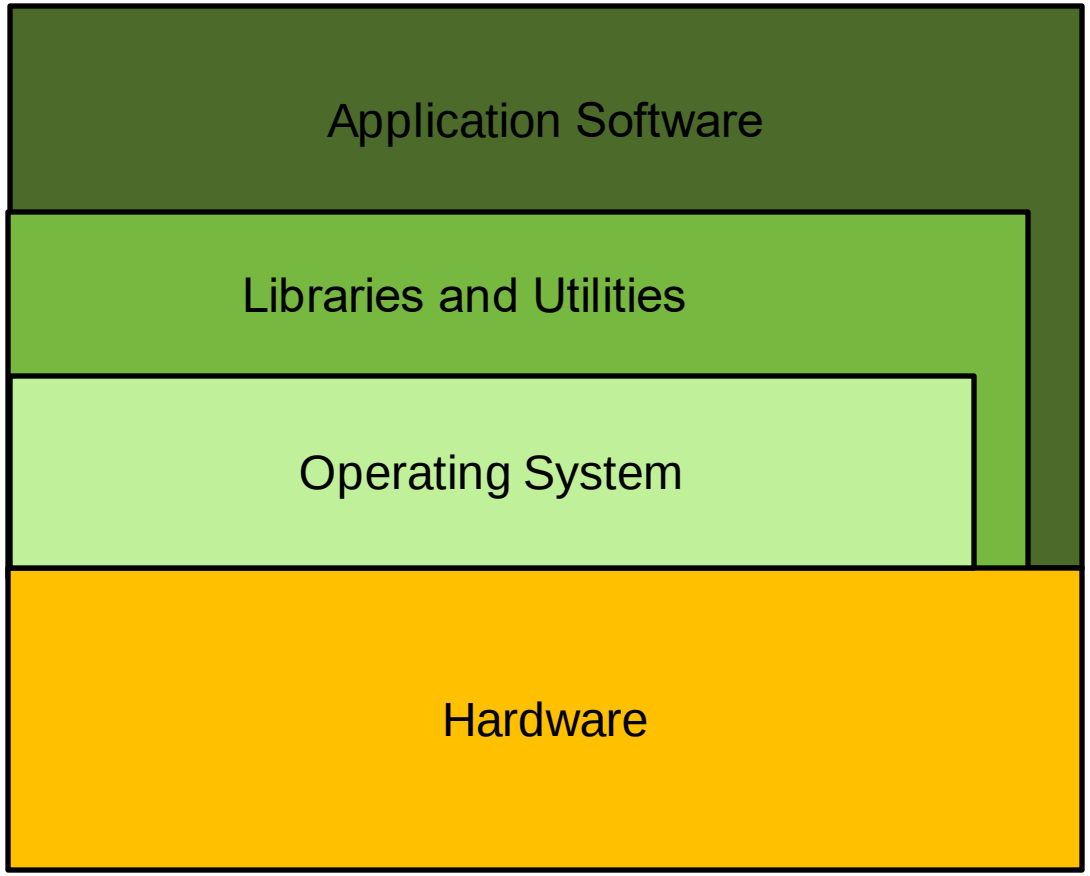
- CPU / chipset
- GPU / graphics card
- Memory (RAM, harddisk)
- I/O devices: Camera, microphone, speaker, keyboard, mousepad, monitor, etc.
- Interfaces: USB, HDMI, etc.
- Battery
- ...

+

Software

- **Operating system (macOS/ Windows 11/ Ubuntu)**
- Device drivers
- Graphical user interface
- Software development tools
- Application software (PDF/image/text editors, Web browsers, Skype/Zoom, etc.)
- ...

Computer Hardware – Software Structure



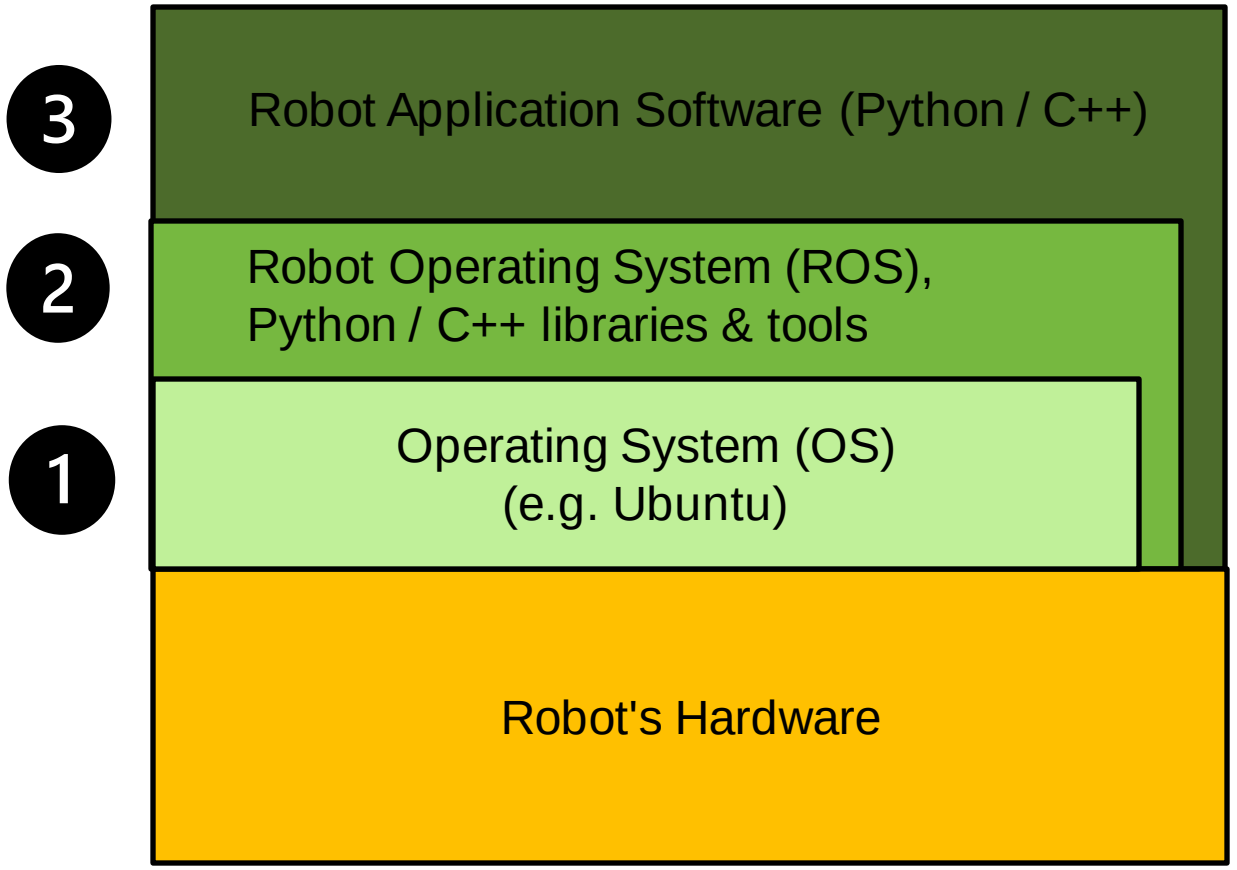
Based on Figure 2.1 in "Operating Systems: Internals and Design Principles", Seventh Edition, by William Stallings. Published by Prentice Hall. Copyright © 2012 by Pearson Education, Inc.

What is an Operating System?

"Operating Systems are those programs that **interface the machine with the application programs**. The main function of these systems is to dynamically allocate the shared system resources to the executing programs. As such, research in this area is clearly concerned with the management and scheduling of memory, processes, and other devices."

-- slightly adapted from *What can be automated?: The Computer Science and Engineering Research Study, MIT Press, 1980.*

Programming Robots: OS, ROS, Programming Language

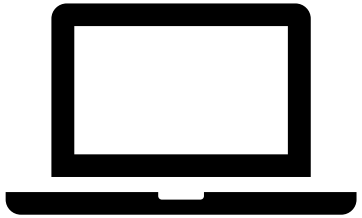


What do you need to create robot application software?

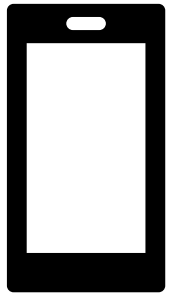
Inspired by Figure 2.1 in "Operating Systems: Internals and Design Principles", Seventh Edition, by William Stallings. Published by Prentice Hall. Copyright © 2012 by Pearson Education, Inc.

Examples of Operating Systems

- Personal computers
  - Microsoft Windows (proprietary)
  - Apple's macOS (proprietary)
  - Linux Operating System (open-source)



- Mobile Devices
  - Apple's iOS
  - Google's Android OS



Functions of an Operating System (1/2)

German Research Center for Artificial Intelligence

- Development of programs
  - Tools such as editors, compilers, interpreters, debuggers to help users to develop new programs.
- Execution of programs
  - Load the program into memory and run it on the CPU/GPU.
- Access to Input and Output (I/O) devices
- Access to stored files
  - Enable storage and retrieval of files on different storage media.

Reference: Section 2.1 in "Operating Systems: Internals and Design Principles", Seventh Edition, by William Stallings. Published by Prentice Hall.

Functions of an Operating System (2/2)

- Detect and respond to hardware and software errors.

- Resource management
  - Decide which executing program gets access to CPU or RAM when and for how long.
  - Decide if and when a program can be given access to I/O devices during execution.
  - Control which program gets access to which files.

Reference: Section 2.1 in "Operating Systems: Internals and Design Principles", Seventh Edition, by William Stallings. Published by Prentice Hall. Copyright © 2012 by Pearson Education, Inc.

Operating System: Ubuntu

- Open source
- Based on Linux operating system
- Debian family

Command Line Interface

- How can users interact with their computers?
  1. Graphical User Interface (GUI)
     - Allows interaction through audiovisual and haptic interfaces (e.g. clicking on icons, giving voice commands).
  2. Command Line Interface (CLI)
     - Allows only text-based interaction.
     - CLI applications can be launched via GUI.
       - » In Ubuntu and macOS: Terminal
       - » In Windows: Command Prompt

CLI: Shell and Commands

German Research Center for Artificial Intelligence

- A shell is a program that allows access to the services of the OS.
- Each open CLI (Terminal or Command Prompt) runs the shell and allows to invoke OS services via text-commands.

- Some common shell commands in Ubuntu:
  - mkdir : To create a new directory (folder) inside the current directory
  - ls: To list the contents of the current directory
  - cd: To change the current directory to another directory
  - pwd: To know the current working directory
  - For more details, check out:
    - https://assets.ubuntu.com/v1/f401c3f4-Ubuntu_Server_CLI_pro_tips_2020-04.pdf
    - https://ubuntu.com/tutorials/command-line-for-beginners#1-overview

CLI: Shell Scripting in Ubuntu

German Research Center for Artificial Intelligence

- Instead of issuing individual commands sequentially from the command line, users can also write "shell scripts" to execute a sequence of commands at once or to do more complex tasks.

- Shell scripts are usually saved with the ".sh" extension and run using the ***source*** command.
  - e.g. source test.sh
- Shell scripts start with the line #!/bin/sh

- A special shell script called ".bashrc" located inside the user's home directory is executed automatically, every time a Terminal window is opened.

- To learn how to write shell scripts: https://www.shellscript.sh/

What is a Process?

- Simply put, a process is a program in execution.
- You can execute a program from CLI or via GUI.
  - When you open any application e.g. a Terminal or a Web browser through GUI, or when you enter a command on the CLI, you start a process or multiple related processes.

- Each process is assigned a process ID by the OS.
  - In Ubuntu, the command "ps" can be used to retrieve the process IDs
- The OS dynamically allocates memory, CPU time, I/O access, etc. to processes with the goals of load balancing, efficient use of resources, enhanced responsivity, etc.

Termination of a Process

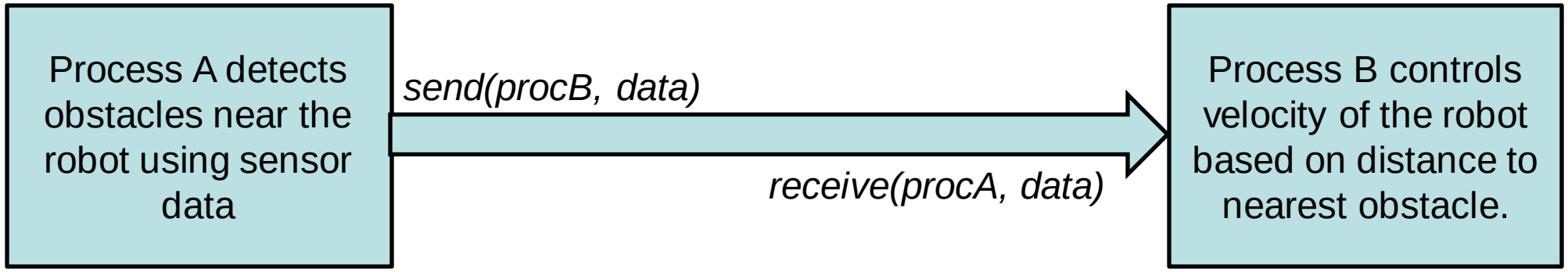
- A process may terminate by itself after its execution is completed.
- If an error is encountered (e.g. division by zero; memory access violation, etc.), then the OS terminates the execution of the process, unless such errors are handled explicitly in the program.

- A process may also be programmed to run indefinitely, unless interrupted by the user. E.g.
  - User can stop such processes via the GUI by choosing "Exit" option from the menu or by clicking on the "Close" button.
  - In CLI in Ubuntu/Linux, the key combination "Ctrl+C" is commonly used to abort execution of a process.
    - Alternatively, "kill –9 <process-id>" command can be used, especially for processes running in the background or on other Terminal windows.

Inter-Process Communication

- Several processes run simultaneously on our computers, and on robots too.
- Sometimes processes may need to communicate with each other, e.g. to exchange data, to coordinate their tasks, or to share some resource such as a file or a device.

- A commonly used method for inter-process communication is **message passing**.
  - Sender / Producer: Sends message to destination
  - Receiver / Consumer: Receives message from sender
  - Depending on the context: Receiver can either block until message is received or can be interrupted to handle the message whenever it arrives.



Shell Piping in Ubuntu

German Research Center for Artificial Intelligence

- In Ubuntu, the piping operator (|) helps to redirect the output of one command to the input of the other.
- It is an implementation of the message passing method for inter-process communication.

- e.g. cat song.txt | grep "robot"
  - The command cat outputs the contents of the given file.
  - The command grep searches for a specific pattern in a stream of characters.
  - Here, the piping operator gives the content of song.txt to grep to check if the word "robot" is present in the lyrics of the song.

Virtualization of Operating System

German Research Center for Artificial Intelligence

- Suppose you are running Microsoft Windows OS on your PC, but would like to work with Ubuntu in your study course. What will you do?

- You have three options:
  1. Get a second PC and install Ubuntu as operating system.
     - Expensive!
  2. Configure your PC for dual boot, so that you can switch between Windows and Ubuntu.
     - Cons: You have to reboot to switch between OS!
     - Pro: Ubuntu gets direct access to the real hardware.
  3. **Virtualization**: Create a virtual environment that simulates the hardware through software.
     - Install Ubuntu on this virtual machine.
     - Virtualization software: Virtual box (https://www.virtualbox.org/wiki/VirtualBox)
     - Pro: You can switch between Windows and Ubuntu without the need to reboot the PC.
     - Cons: It is slower, since the Ubuntu is effectively running on top of Windows.

Remote Access

- We learned that we can run programs either via GUI or CLI on the PC.
- But, how can we run programs on the robot?
  1. Connect an external monitor and keyboard to the robot.
     - CLI appears on the monitor and commands can be given via the keyboard.
  2. Or, login to the robot from another PC which already has these I/O devices.
     - This can be done using the command "ssh".
     - From the CLI of your PC, type: ssh username@robot-ip-address
     - Whatever commands you execute from this CLI window will be executed on the robot.

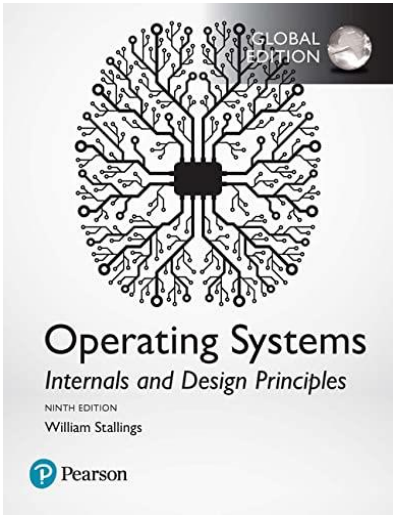
Conclusion

- In this lecture, you learnt to:
  1. Define an operating system and give examples for it.
  2. List the functions of an operating system.
  3. Understand the purpose of a command line interface.
  4. Give examples for commonly used commands in Ubuntu operating system.
  5. Define a process and describe a method for interprocess communication.
  6. Explain the concept of virtualization.

References and Recommended Reading

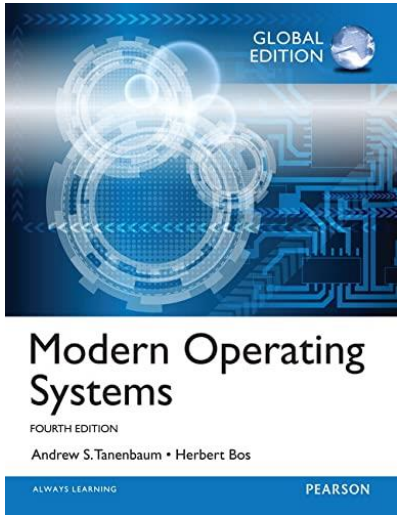
German Research Center for Artificial Intelligence

- William Stallings. Operating Systems: Internals and Design Principles, Global Edition (2018)



- https://repository.dinus.ac.id/docs/ajar/Operating_System.pdf

- Andrew S. Tannenbaum & Herbert Bos. Modern Operating Systems: Global Edition (2014)



- https://csc-knu.github.io/sys-prog/books/Andrew%20S.%20Tanenbaum%20-%20Modern%20Operating%20Systems.pdf

Next Part: Robot Operating System (ROS)