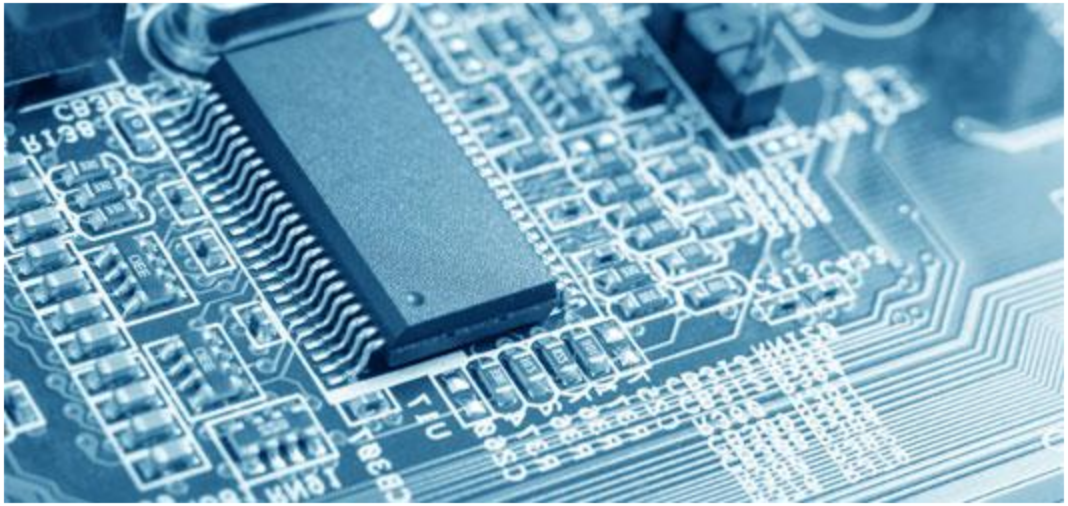
NAZARBAYEV UNIVERSITY
SCHOOL OF SCIENCE AND TECHNOLOGY



CSCI502 – Hardware/Software Co-Design

Lecture 1 – Introduction & Course Overview

Course Instructor: Dr. Almas Shintemirov

9 January 2018

Course Logistics

Reference Reading:
Real-Time Embedded Systems: Chapter 2
Exploring BeagleBone. 2nd edition: Chapters 3, 4

My Background

Education:

2009 Ph. D. in Electrical Engineering and Electronics, The University of Liverpool, UK

Working Experience:

2011 – present Associate Professor, Department of Robotics and Mechatronics, Nazarbayev University (2011–2015 – Assistant Professor)

2009 – 2011 Various administrative positions in the Nazarbayev University development team

Astana LAboratory for Robotic and Intelligent Systems (ALARIS)

- Lab location: lab C4.452
- Lab web-site www.alaris.kz

At ALARIS lab we conduct research on capacity building for developing local robotic, mechatronic and intelligent systems in the following areas:

- **Development of optimal trajectory planning and control algorithms for mobile robot/ autonomous vehicles**
- **Development of optimal and human-safe control algorithms for industrial robots (Industry 4.0 concepts)**
- **3D point cloud and machine learning based object recognition and classification**
- **Development of intelligent human-machine interfaces for assistive robotic systems**
- **Development and control of motion control platforms based on spherical parallel manipulators**
- **Design of low-cost robotic end effectors and rehabilitation robots**

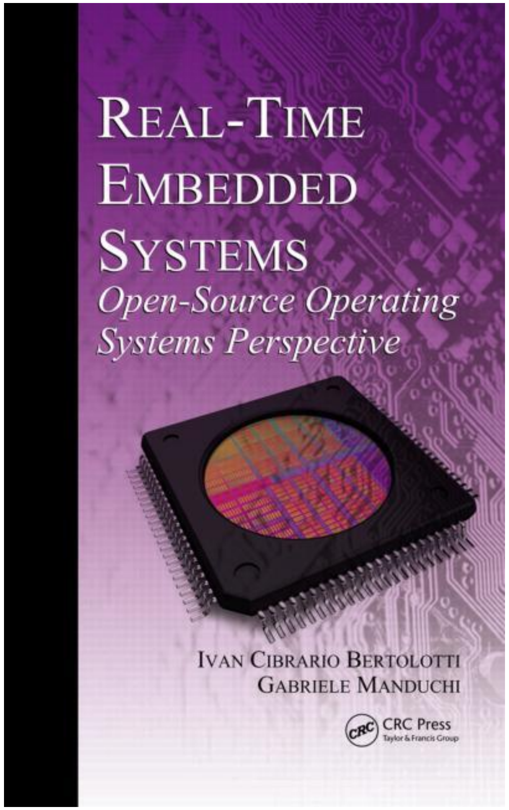
Course Prerequisites

- Background required:
  - Programming knowledge in C/C++
  - Electric and electronic circuits and sensors (desired)

- Teaching:
  - Lectures (small part of the course)
  - Practical projects using BeagleBoneBlack boards and embedded Linux and Robot Operating System (ROS)
  - Final project

Course Literature

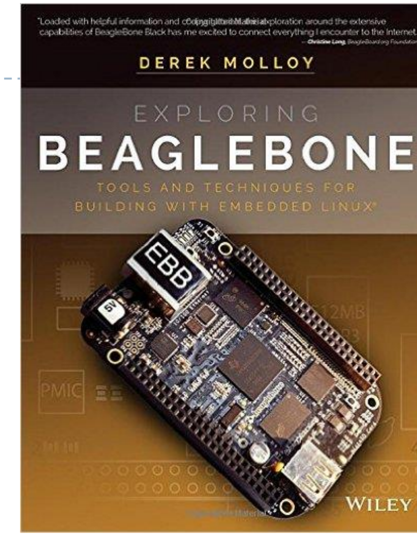
- **Real-Time Embedded Systems: Open-Source Operating Systems Perspective** by I. C. Bertolotti and G. Manduchi, 2012

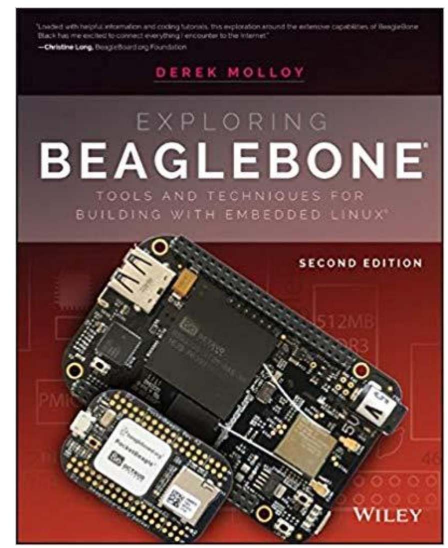


- Grab a copy for yourself from the library
- Contain material related to our course

Course Literature

- **Exploring BeagleBone: Tools and Techniques for Building with Embedded Linux** by Derek Molloy, 1st and 2nd editions.

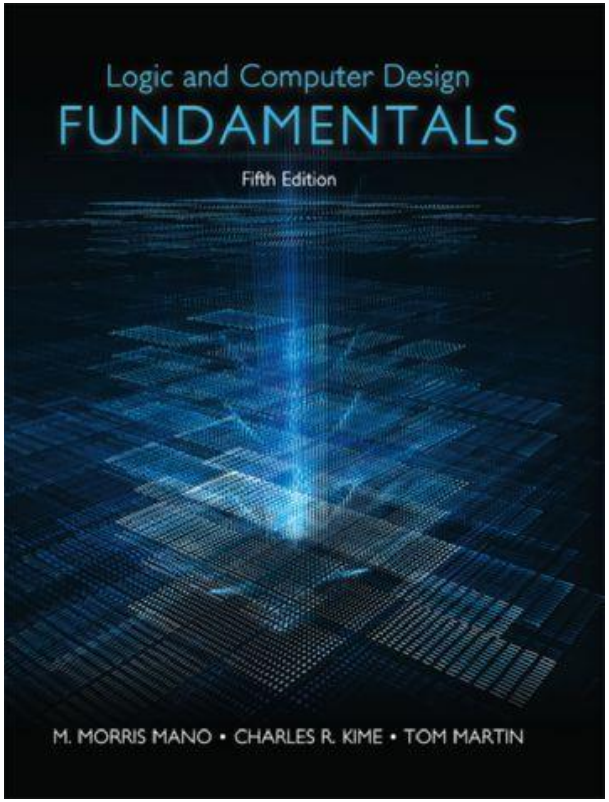




- Available in Moodle in electronic form and in the NU library (1st ed. copies)
- Contain material related to practical BeagleBone Black boards

Reference Literature

- **Logic and Computer Design Fundamentals** by M.M. Mano, C.R. Kime, T. Martin, 5th edition, 2015



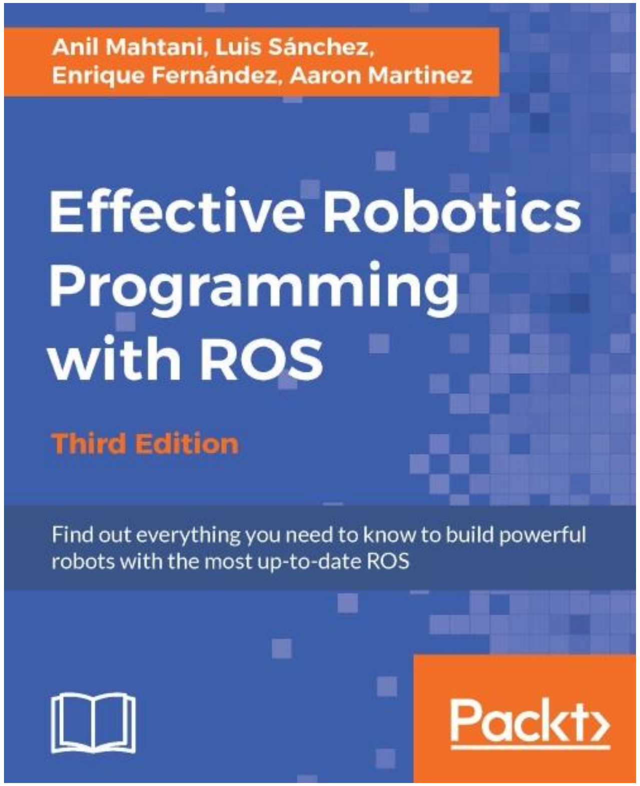
- Available in Moodle in electronic form
- Contain material related to computer architecture design

Reference Literature

- **Effective Robotics Programming with ROS.Third edition**, 2017

- Available in Moodle in electronic form

- **Other ROS books will be provided in Moodle if needed for class projects.**



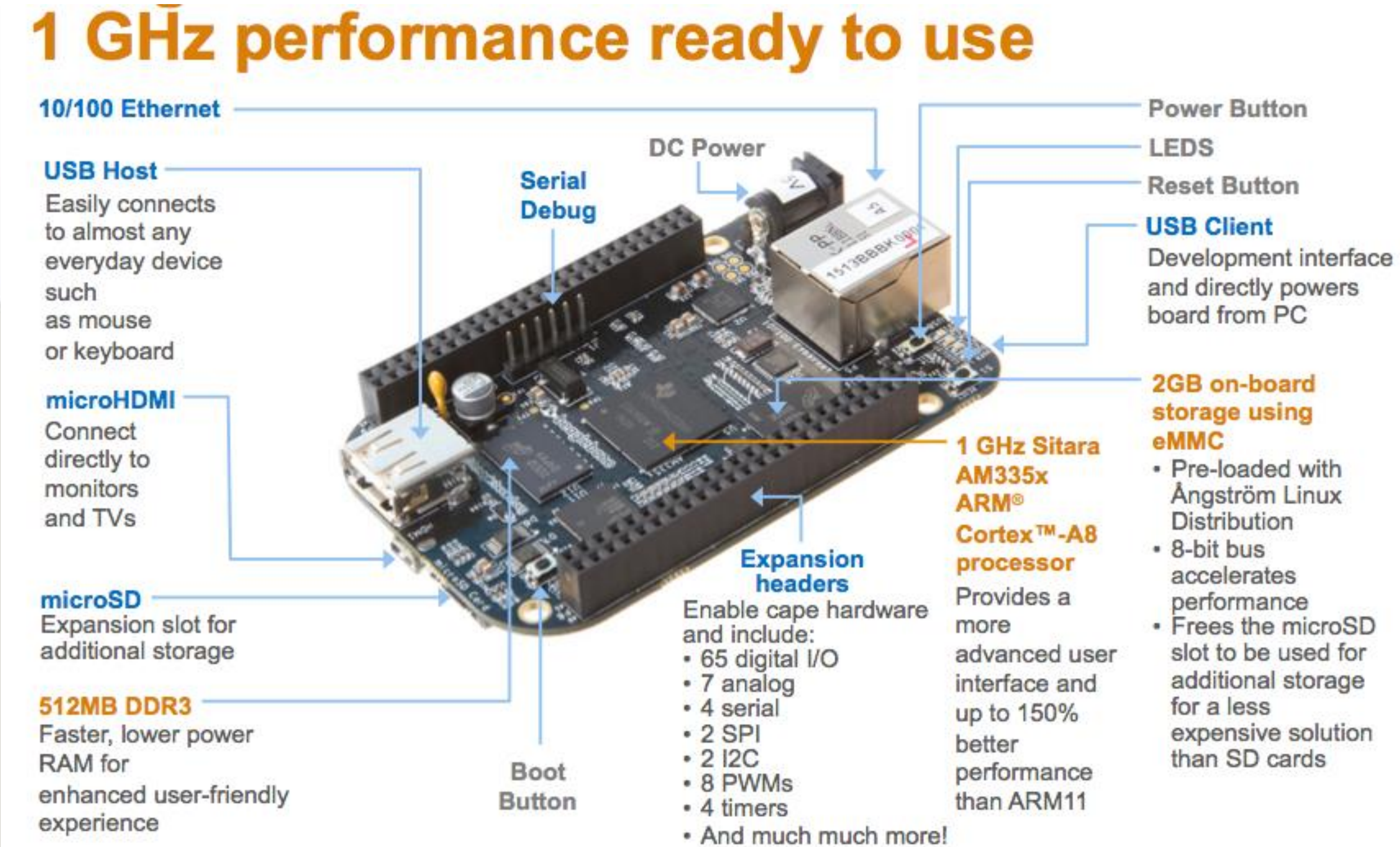
Course Oultine

- Introduction to Embedded Systems Hardware
- Operating Systems
- Embedded Linux
- Processes and Threads
- Synchronization
- BeagleBone Black assignments and projects
- ROS fundamentals and projects

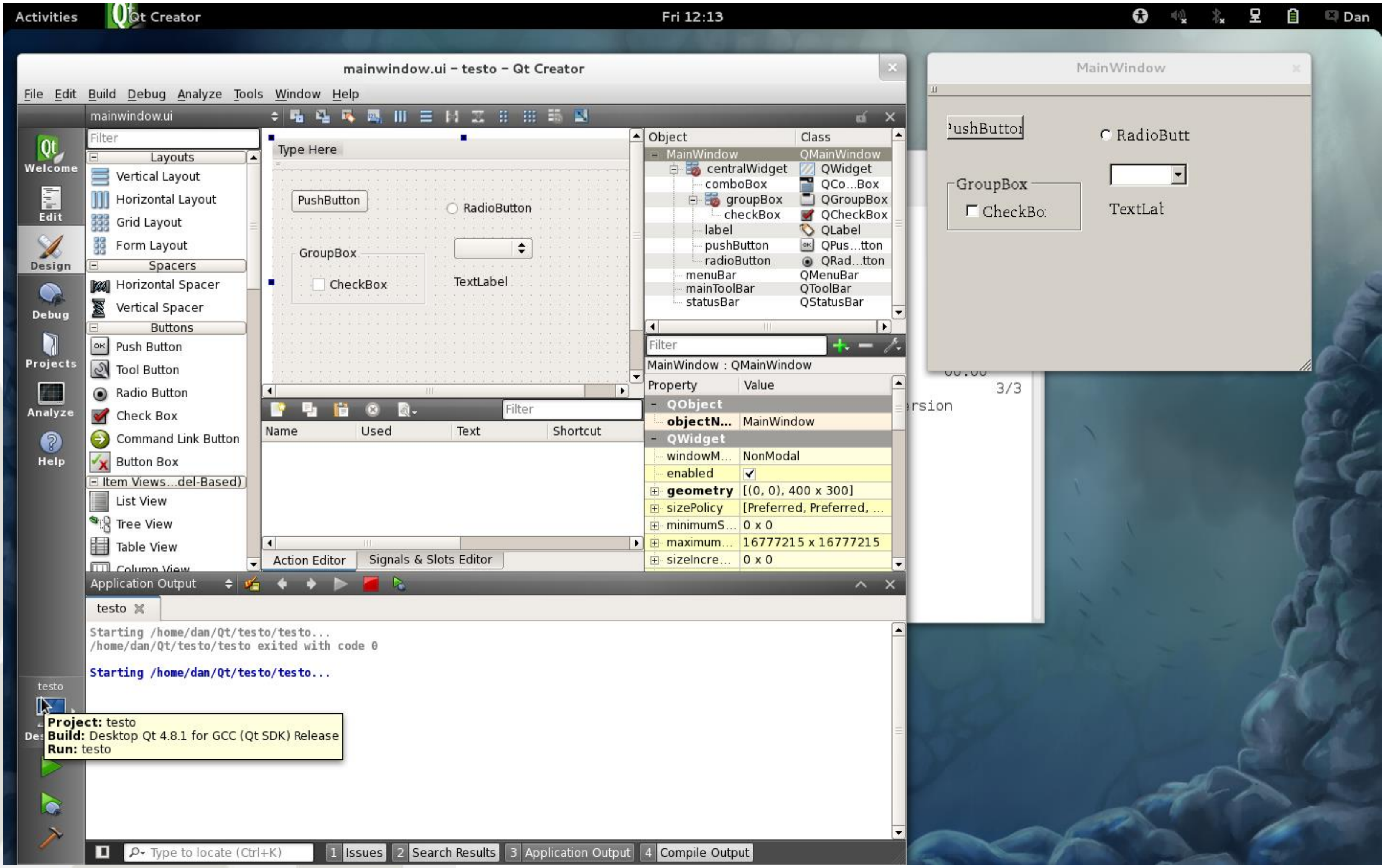
Course Assessment (Preliminary)

Activity	Quantity	Weight
Homework/Project Assignments	5	59%
Midterm Exams	3	36%
Attendance		5%

Course Lab Hardware: BeagleBoneBlack Board



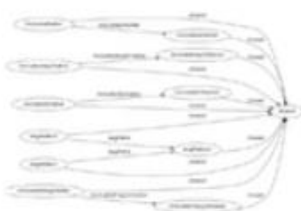
Course Lab Software: Qt GUI Design



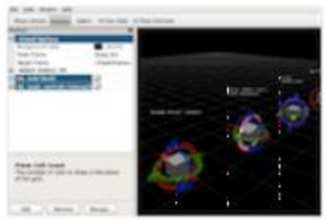
Course Software Platform: Embedded Linux

Course Lab Software:
Robot Operating System (ROS) www.ros.org

ROS = Robot Operating System



+



+

+

ros.org

Plumbing
- Process management
- Inter-process communication
- Device drivers

Tools
- Simulation
- Visualization
- Graphical user interface
- Data logging

Capabilities
- Control
- Planning
- Perception
- Mapping
- Manipulation

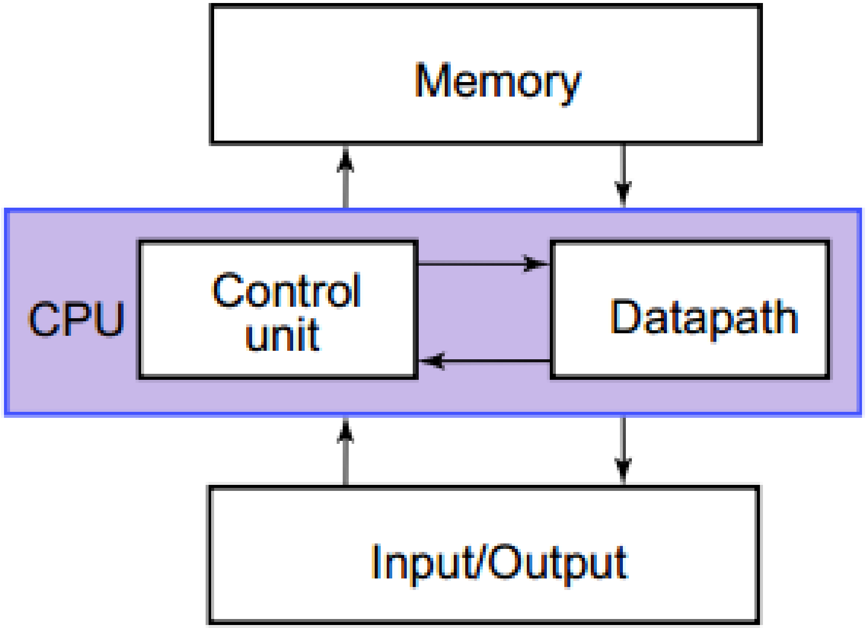
Ecosystem
- Package organization
- Software distribution
- Documentation
- Tutorials

- ROS is becoming a standard hardware interfacing and programming environment in for robotics research (machine learning, AI for robots, autonomous vehicles, robot control).
- Most of CS based robotics research engineer and PhD positions worldwide require/desire ROS experience

@Péter Fankhauser

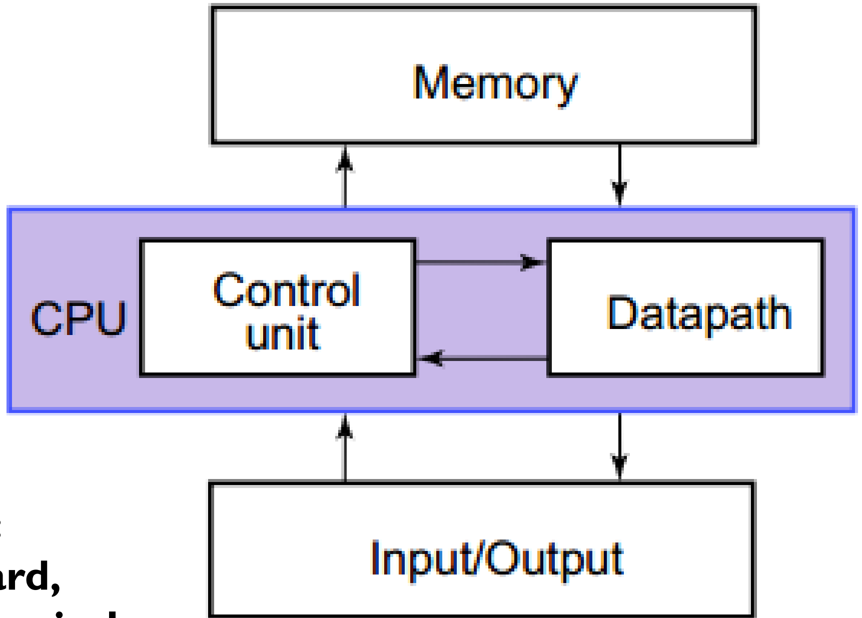
General Computer Architecture

- CPU: the "brain" of a computer
  - Control unit does calculations on data in datapath
- Memory: stores data (for later use)
- Input/Output: interface to outside (disk, network, monitor, keyboard, mouse, etc.)



General Computer Architecture

- Computer processes programs (stored in memory)
  - program made up of sequences of instructions
- Programs modify data also stored in memory



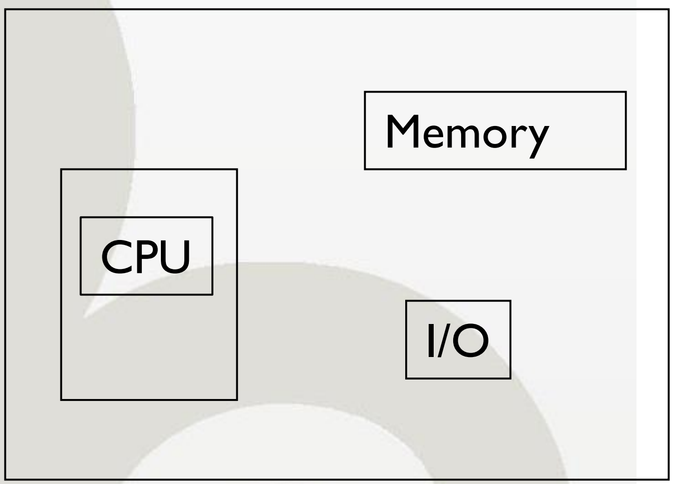
Inputs: keyboard, mouse, wireless, microphone

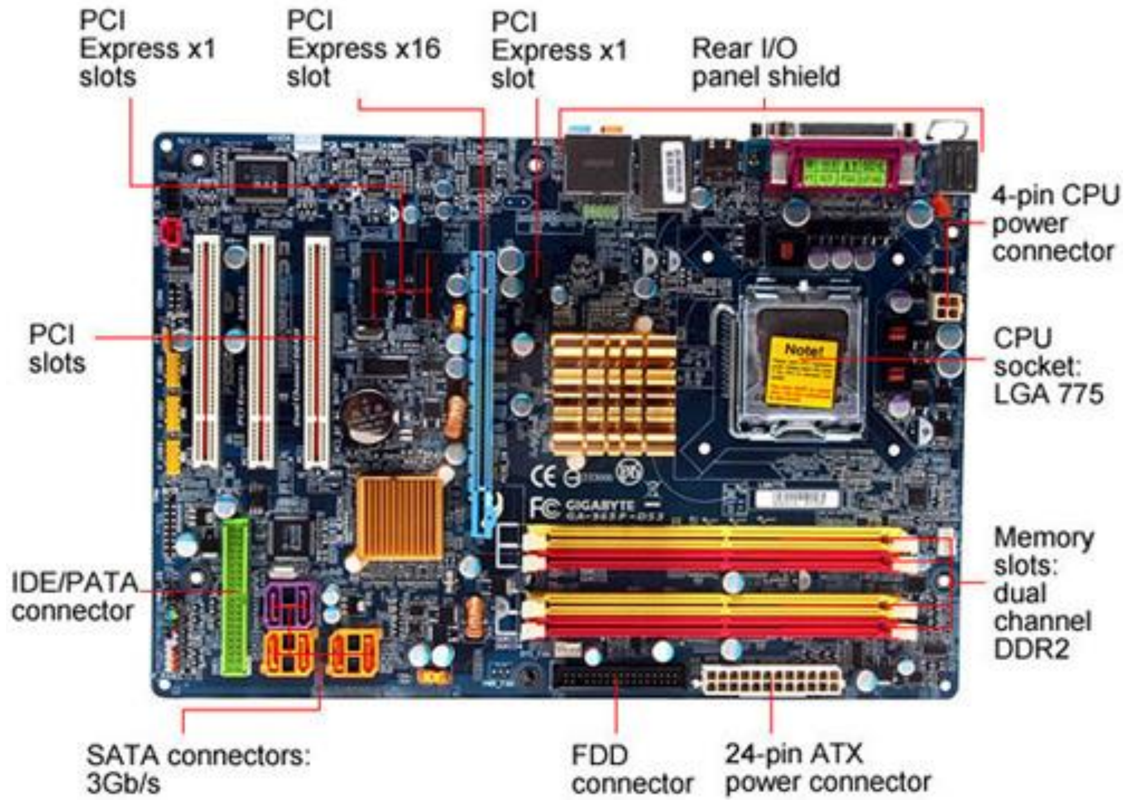
Outputs: LCD screen, wireless, speakers

More on this later in term…

Digital Computers: Basics

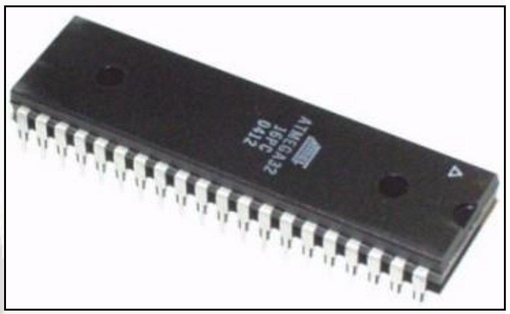
- CPU
- Memory
- I/O



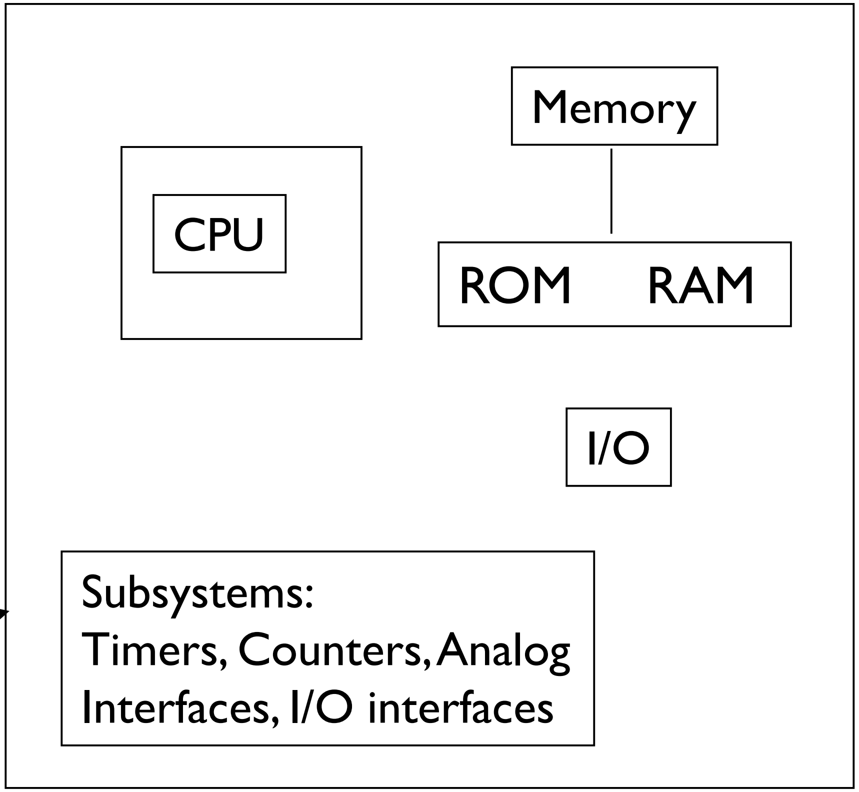


Could be a chip, a board, or several boards

Microcontrollers



A single chip



Microcontrollers

Microcontrollers are 'single chip' computers specifically designed to:

- Read input devices, such as buttons and sensors.
- Process data or information.
- Control output devices, such as lights, displays, motors and speakers.

And Beyond – Embedded Systems

Embedded systems (ES) = **information processing systems embedded into a larger product**

Examples:

Main reason for buying is **not** information processing

Embedded Systems & Cyber-Physical Systems

<u>“Dortmund“ Definition</u>: [Peter Marwedel]

Embedded systems are information processing systems embedded into a larger product

<u>Berkeley:</u> [Edward A. Lee]:
Embedded software is software integrated with physical processes. The technical problem is managing time and concurrency in computational systems.

Cyber-Physical (cy-phy) Systems (CPS) are integrations of computation with physical processes [Edward Lee, 2006].

Cyber-physical system (CPS) =
Embedded System (ES) + *physical environment*

Embedded Systems

- *An **embedded system** is a computer system designed to perform one or a few dedicated functions, often with real-time computing constraints. It is usually embedded as part of a complete device including hardware and mechanical parts.*
- *Embedded systems span all aspects of modern life and there are many examples of their use: mobile phones, MP3 players, digital cameras, GPS receivers, household appliances, avionics system, medical systems, etc.*
- *Embedded processors can be microprocessors or microcontrollers.*
- *The program instructions written for embedded systems are referred to as firmware, and are stored in read-only memory or Flash memory chips. They run with limited computer hardware resources: little memory, small or non-existent keyboard and/or screen.*

Application Area: Automotive Electronics

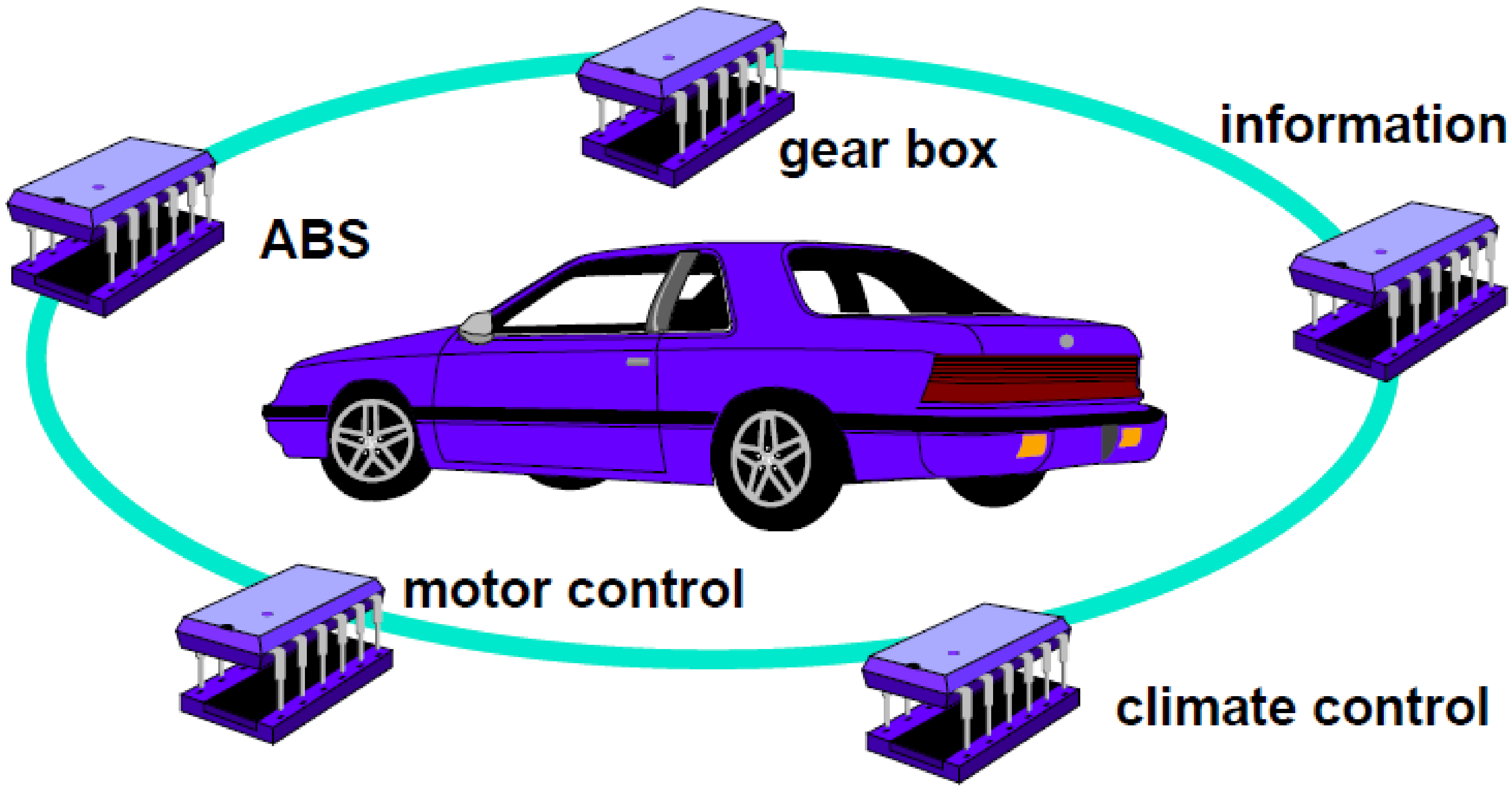
Functions by embedded processing:

- ABS: Anti-lock braking systems
- ESP: Electronic stability control
- Airbags
- Efficient automatic gearboxes
- Theft prevention with smart keys
- Blind-angle alert systems
- ... etc ...

- Multiple networks
- Multiple networked processors

Examples of Embedded Systems

Car as an integrated control-, communication and information system.



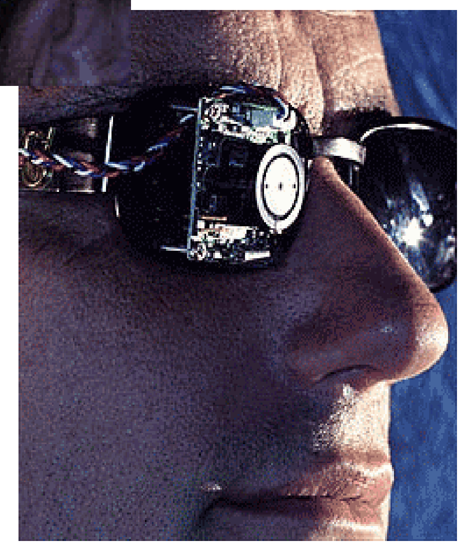
Application Area: Avionics

- Flight control systems,
- anti-collision systems,
- pilot information systems,
- power supply system,
- flap control system,
- entertainment system,
- …

- Dependability is of outmost importance.

Medical Systems

- For example:
  - Artificial eye: several approaches, e.g.:
    - Camera attached to glasses; computer worn at belt; output directly connected to the brain, "pioneering work by William Dobelle". Previously at [www.dobelle.com]



- Translation into sound; claiming much better resolution. [http://www.seeingwithsound.com/etumble.htm]

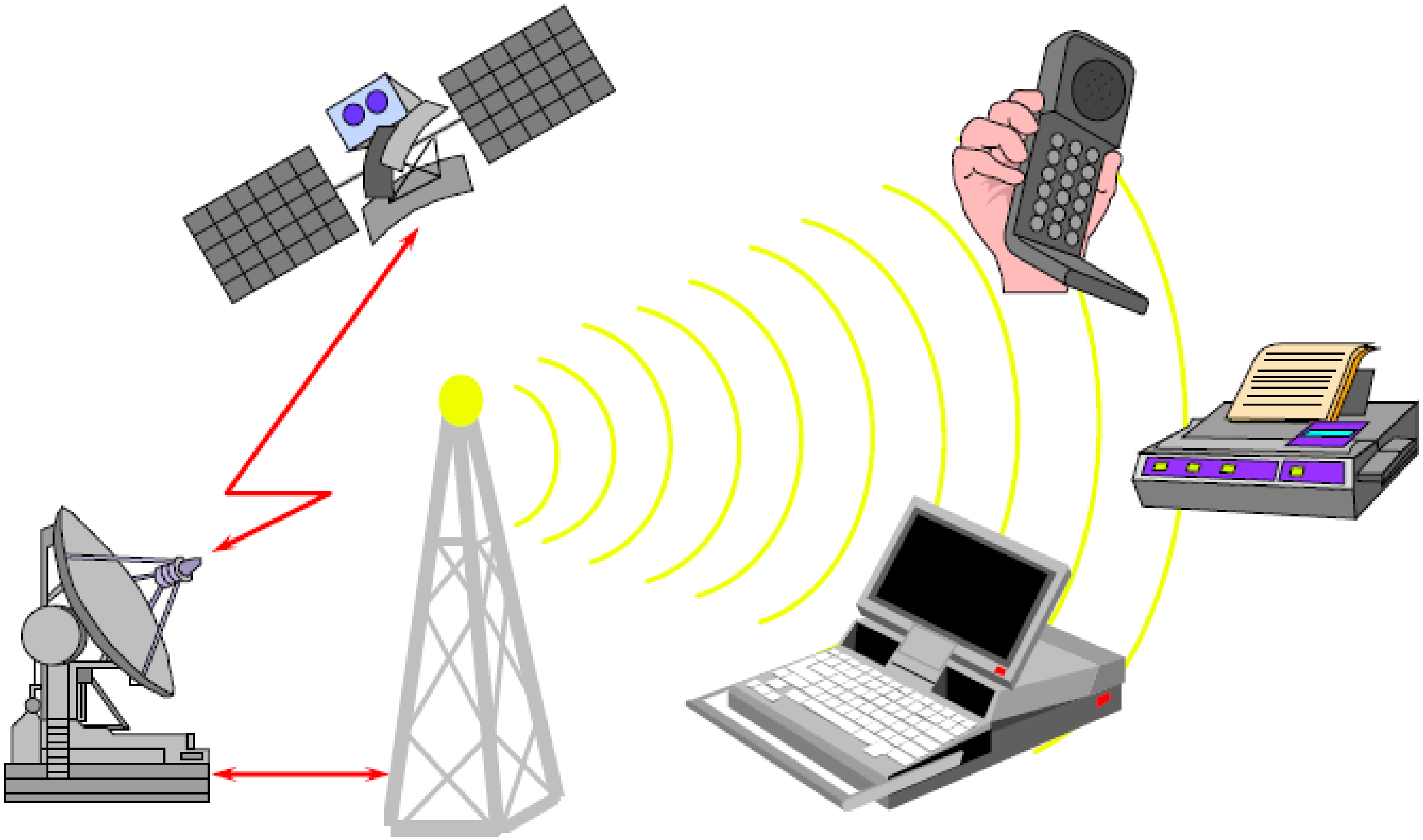
Robotics and Machinery

Logistics & Communication

- Applications of embedded/cyber-physical system technology to logistics:
  - Radio frequency identification (RFID) technology provides easy identification of each and every object, worldwide.
  - Mobile communication allows unprecedented interaction.
  - The need of meeting real-time constraints and scheduling are linking embedded systems and logistics.
  - The same is true of energy minimization issues

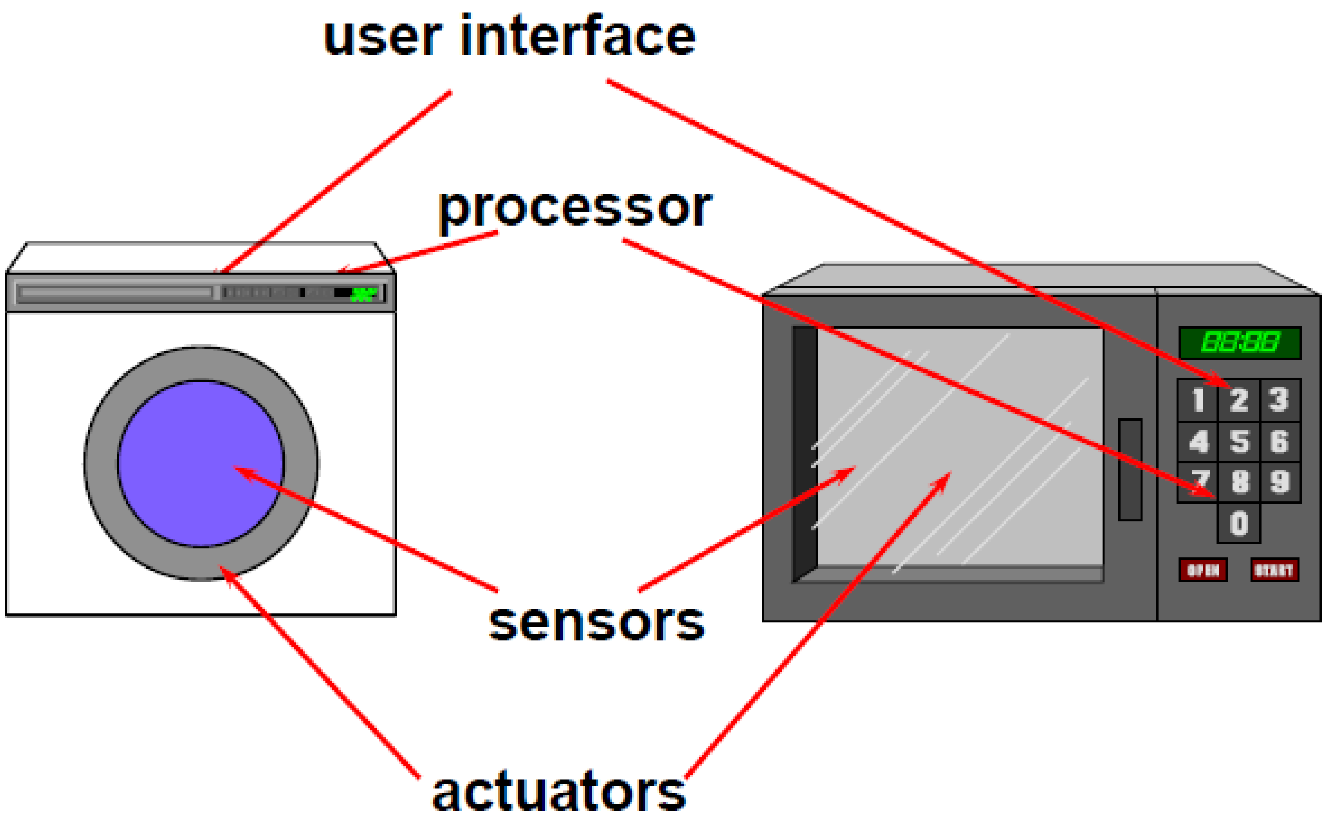
Communication

- Information systems, for example wireless communication (mobile phone, Wireless LAN, ...), end-user equipment, router, ...



Home Appliances

Consumer electronics, for example MP3 Audio, digital camera, home electronics, … .



Embedded products are found in:

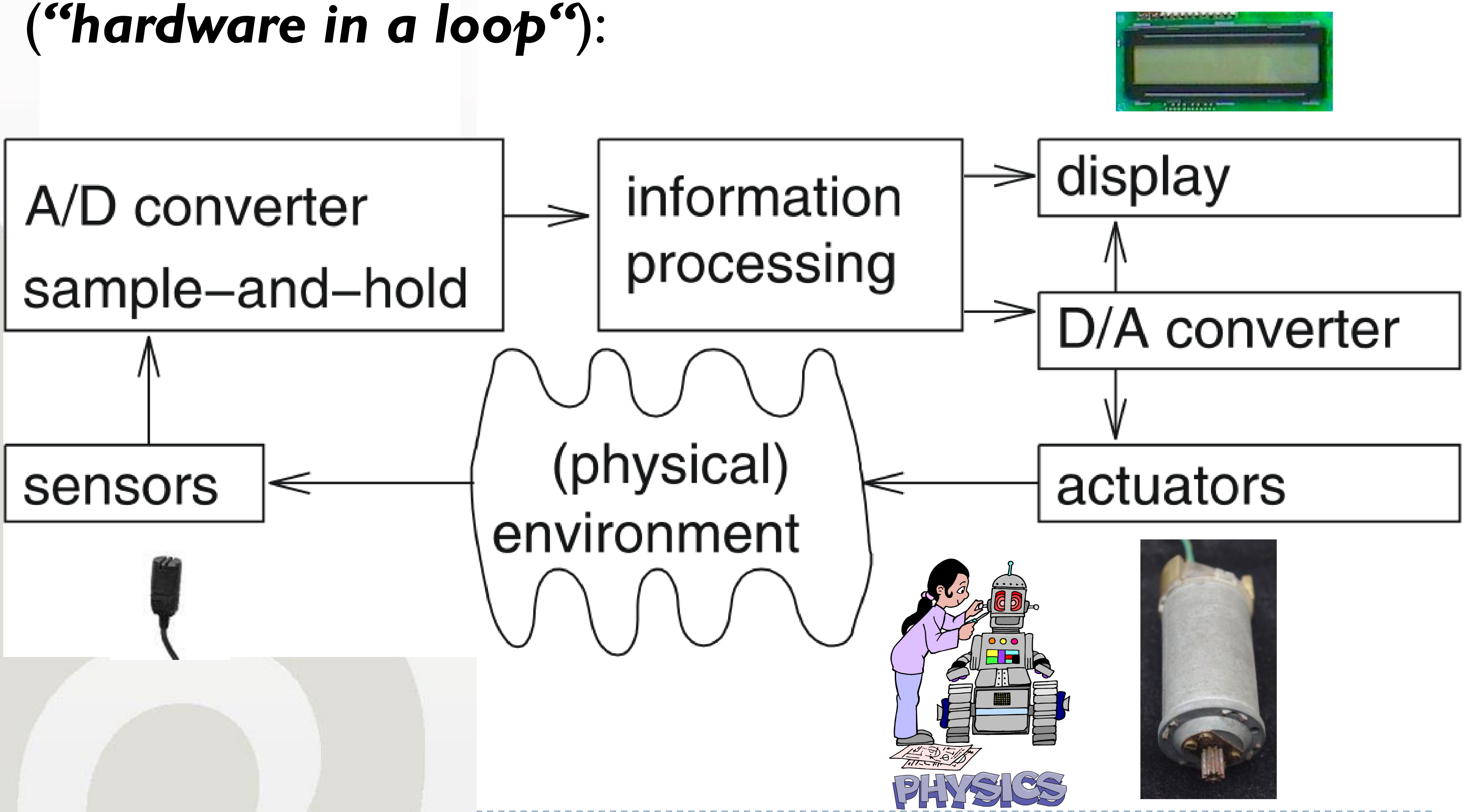
- Robotics
- Industry
- Automotive
- Aerospace
- Medical systems
- Mobile systems
- Communication
- Networking
- Household products (dishwasher, etc)
- Media products – broadcasting
- Cameras
- ---- in other words, everythere ----

Practical Embedded Systems

- Aerospace
  - Flight control
  - Navigation
  - Pilot interface
- Automotive
  - Airbag deployment
  - Antilock braking
  - Fuel injection
- Household
  - Microwave oven
  - Rice cooker
  - Washing machine
- Industrial
  - Crane
  - Paper machine
  - Welding robot
- Multimedia
  - Console game
  - Home theater
  - Simulator
- Medical
  - Intensive care monitor
  - Magnetic resonance imaging
  - Remote surgery

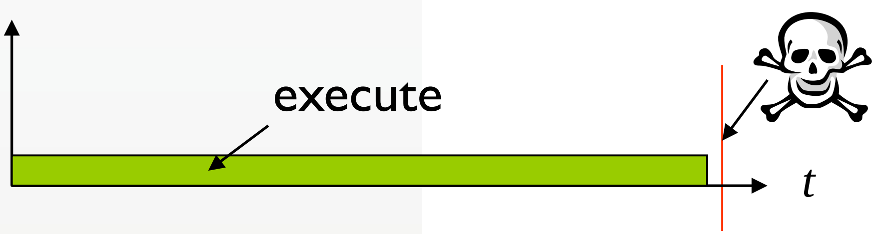
ES Hardware

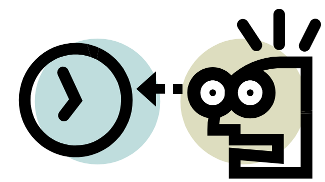
- ES hardware is frequently used in a loop (*“hardware in a loop“*):



Real-time constraints

- CPS must meet **real-time constraints**
  - A real-time system must react to stimuli from the controlled object (or the operator) within the time interval **dictated** by the environment.



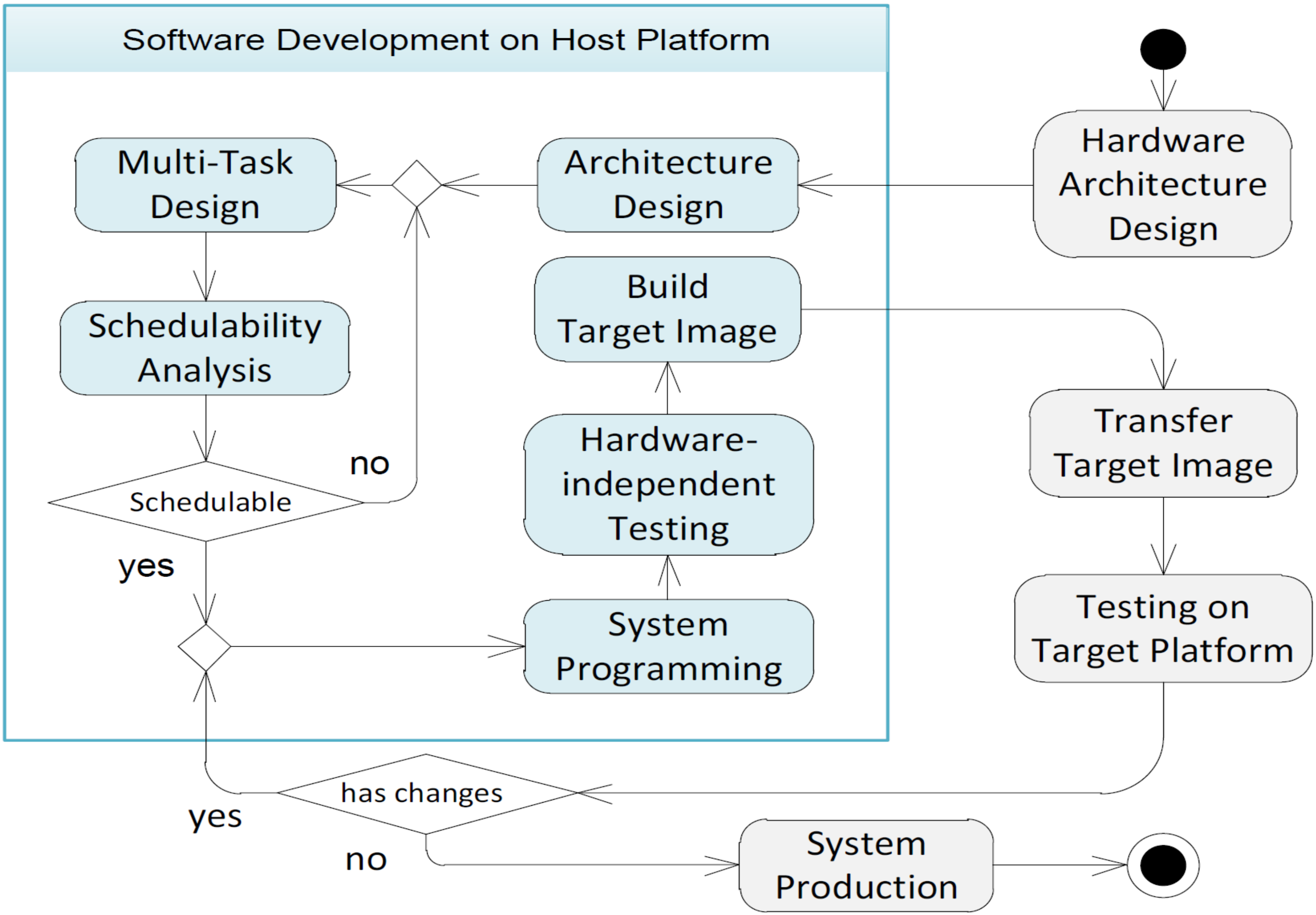


- **"A real-time constraint is called hard, if not meeting that constraint could result in a catastrophe"**
- All other time-constraints are called **soft**.
- A guaranteed system response has to be explained without statistical arguments

Real-Time Systems & CPS

- ES and Real-Time Systems synonymous?
  - For some embedded systems, real-time behavior is less important (smart phones)
  - If real-time behavior is essential than Real-time ES

Real-Time Embedded Systems Development Process

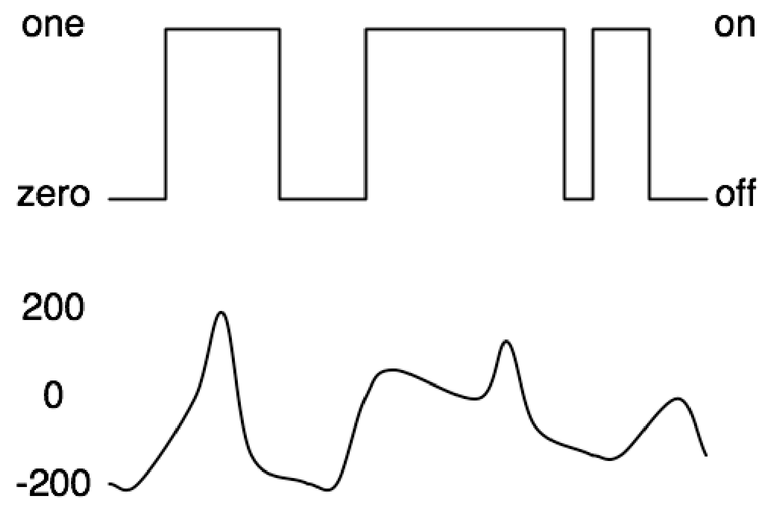


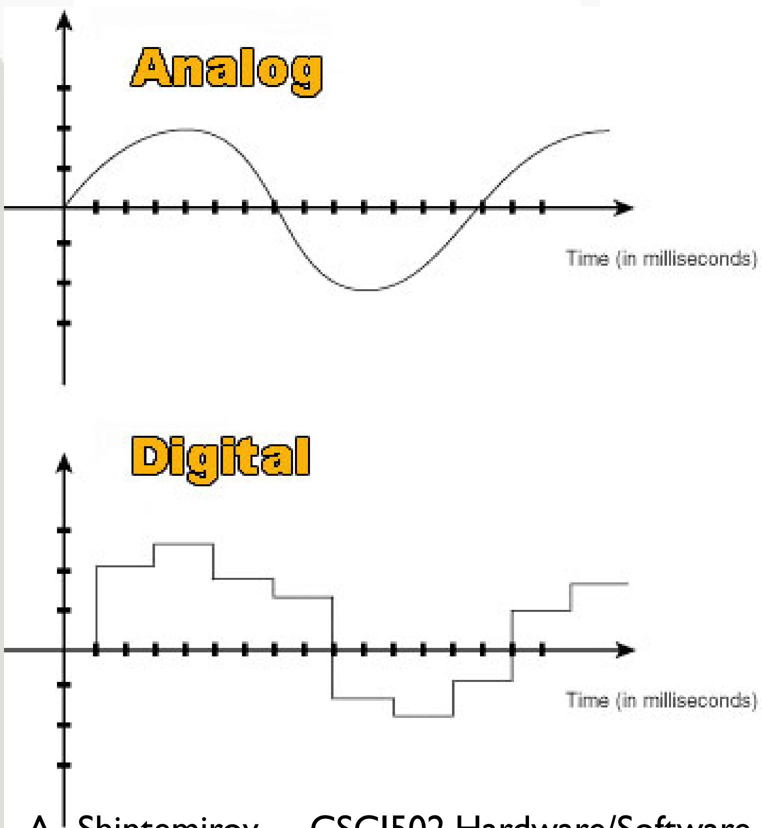
Digital Abstraction

- Most physical variables are **continuous**
  - Voltage on a wire
  - Frequency of an oscillation
  - Position of a mass
- Digital abstraction considers **discrete subset** of values
- **Two discrete values:**
  - 1's and 0's
  - 1, TRUE, HIGH
  - 0, FALSE, LOW
- **1 and 0:** voltage levels, rotating gears, fluid levels, etc.
- Digital circuits use **voltage** levels to represent 1 and 0
- ***Bit***: *Binary digit*

Digital and Analog Input/Output

- ❑ Digital I/O – only two values: on/off
- ❑ Analog I/O – many values
- ❑ Computers don't really do analog
- ❑ So they fake it, with *quantization*



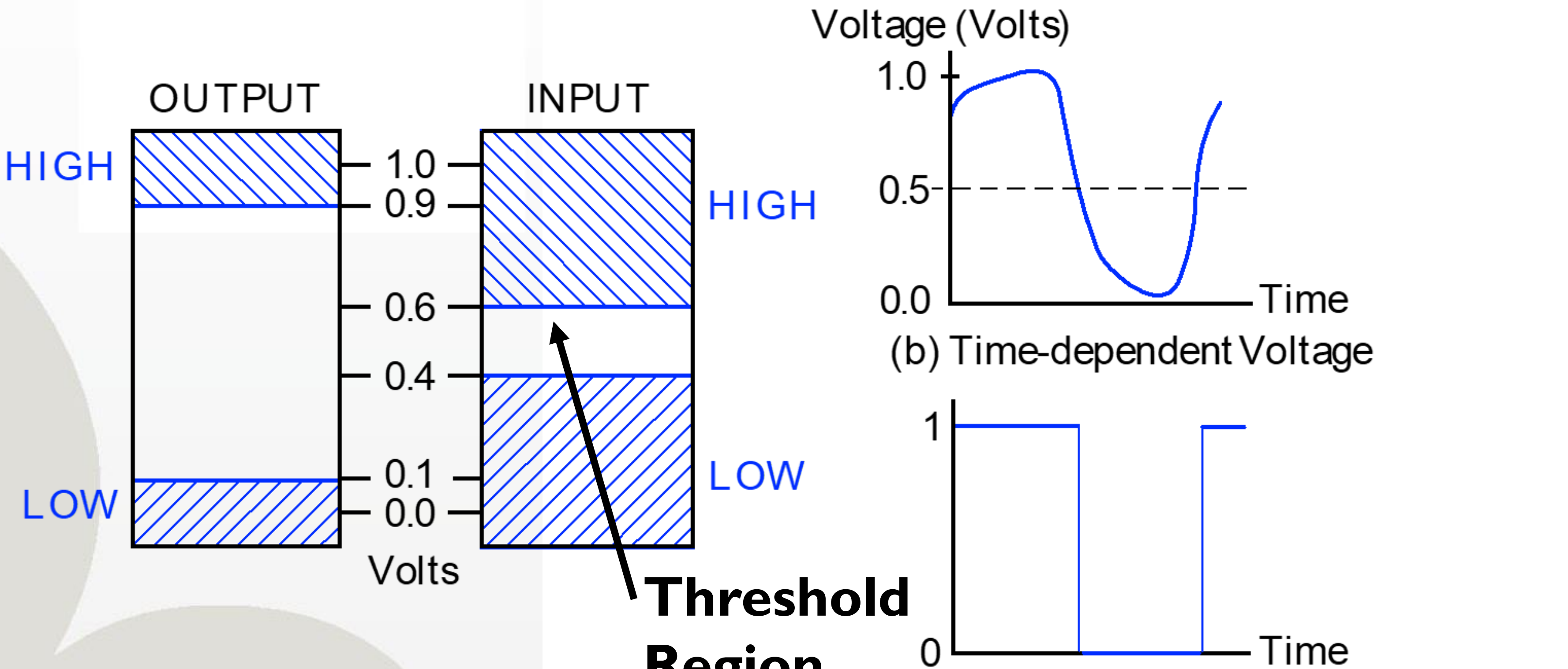


- Quantization = breaking up the analog range into bins.
- The number of bins is the resolution.
- More bins = higher accuracy, but requires more complex circuitry to is implement
- Digital can be thought of as only two bins.

Information Representation - Signals

- Information variables represented by physical quantities.
- For digital systems, the variables take on discrete values.
- Two level, or binary values are the most prevalent values in digital systems.
- Binary values are represented abstractly by:
  - digits 0 and 1
  - words (symbols) False (F) and True (T)
  - words (symbols) Low (L) and High (H)
  - and words On and Off.
- Binary values are represented by values or ranges of values of physical quantities

Signal Example – Physical Quantity: Voltage



(a) Example voltage ranges

(b) Time-dependent Voltage

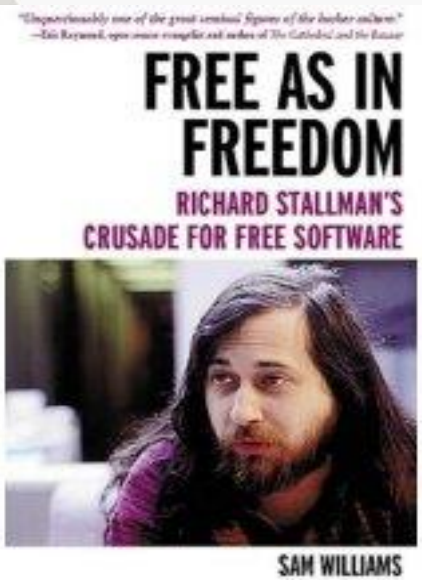
(c) Binary model of time-dependent voltage

History: Before Linux

- In 80's, Microsoft's DOS was the dominated OS for PC
- Apple MAC was better, but expensive
- UNIX was much better, but much, much more expensive. Only for minicomputer for commercial applications
- People was looking for a UNIX based system, which is cheaper and can run on PC
- Both DOS, MAC and UNIX were proprietary, i.e., the source code of their kernel is protected
- No modification is possible without paying high license fees

Linux History: GNU project

- Established in 1984 by Richard Stallman, who believes that software should be free from restrictions against copying or modification in order to make better and efficient computer programs
- Aim to create a free UNIX like OS, consisting of a kernel and all associated software packages



GNU is a recursive acronym for "GNU's Not Unix"

Aim at developing a complete Unix-like operating system which is free for copying and modification

Companies make their money by maintaining and distributing the software, e.g. optimally packaging the software with different tools (Redhat, Slackware, Mandrake, SuSE, etc)

Stallman built the first free GNU C Compiler in 1991. But still, an OS was yet to be developed

Beginning of Linux

- A famous Holland professor from Andrew Tanenbaum developed Minix, a simplified version of UNIX that runs on PC in mid-1980s
- Minix is for class teaching only. No intention for commercial use
- Inspired by Minux in Sept 1991, Linus Torvalds, a second year student of Computer Science at the University of Helsinki, developed the preliminary kernel of Linux, known as Linux version 0.0.1 for running on his Intel 80386 PC

Message from Professor Andrew Tanenbaum

"I still maintain the point that designing a monolithic kernel in 1991 is a fundamental error. Be thankful you are not my student. You would not get a high grade for such a design :-)"

(Andrew Tanenbaum to Linus Torvalds)

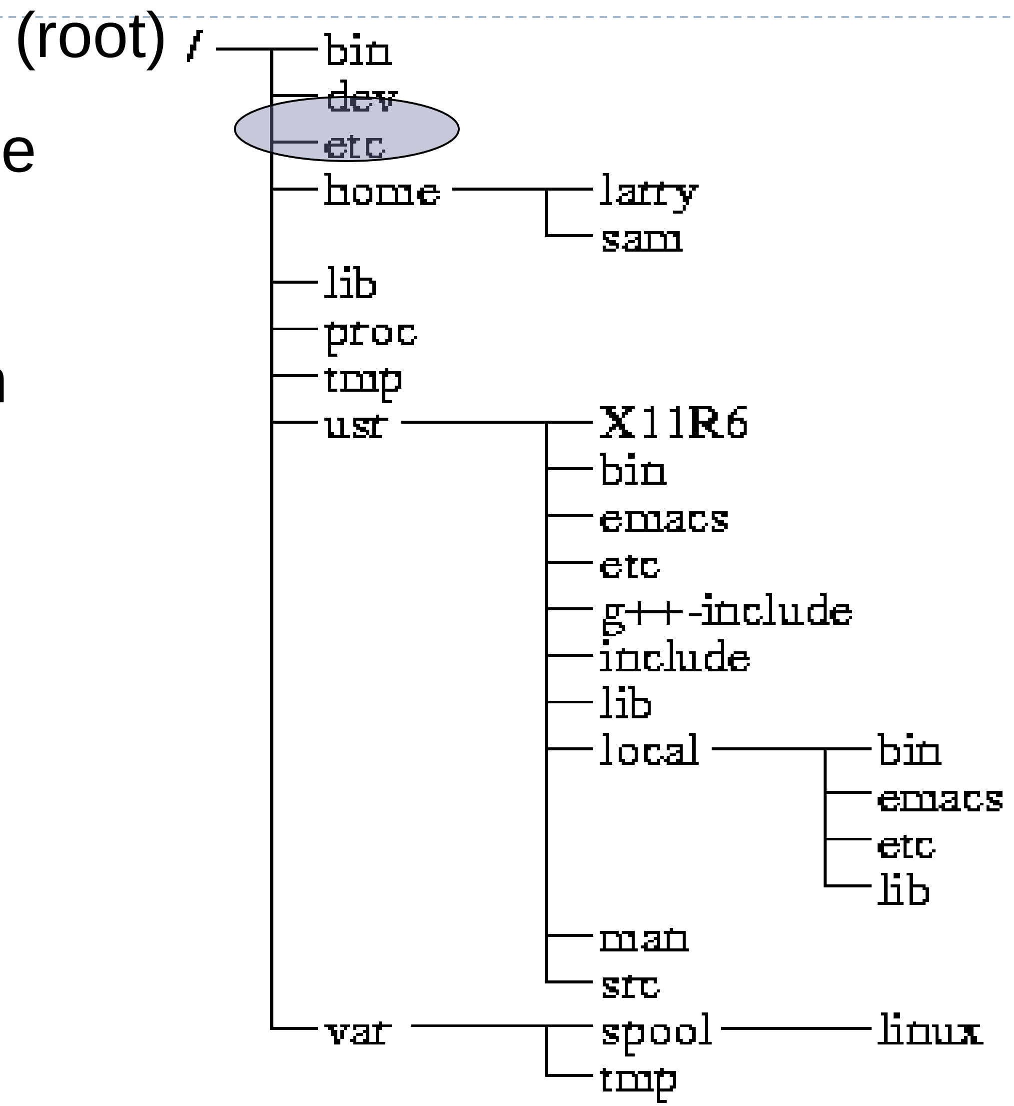
- Soon more than a hundred people joined the Linux camp. Then thousands. Then hundreds of thousands
-
- It was licensed under GNU General Public License, thus ensuring that the source codes will be free for all to copy, study and to change.

Linux Today

- Linux has been used for many computing platforms:
  PCs, supercomputers, smartphones (Android is based on Linux)…
- Not only character user interface but graphical user interface is available
- Commercial vendors moved in Linux itself to provide freely distributed code. They make their money by compiling up various software and gathering them in a distributable format
  - Red Hat, RTLinux, etc

Linux Directory Tree

When you log on the the Linux OS using your username you are automatically located in your home directory.



The most important subdirectories inside the root directory are:

- **/bin** : Important Linux commands available to the average user.
- **/boot** : The files necessary for the system to boot. Not all Linux distributions use this one. Fedora does.
- **/dev** : All device drivers. Device drivers are the files that your Linux system uses to talk to your hardware. For example, there's a file in the /dev directory for your particular make and model of monitor, and all of your Linux computer's communications with the monitor go through that file.
- **/etc** : System configuration files.
- **/home** : Every user except root gets her own folder in here, named for her login account. So, the user who logs in with linda has the directory /home/linda, where all of her personal files are kept.
- **/lib** : System libraries. Libraries are just bunches of programming code that the programs on your system use to get things done.

The most important subdirectories inside the root directory are:

- **/mnt :** Mount points. When you temporarily load the contents of a CD-ROM or USB drive, you typically use a special name under /mnt. For example, many distributions (including Fedora) come, by default, with the directory /mnt/cdrom, which is where your CD-ROM drive's contents are made accessible.
- **/root** : The root user's home directory.
- **/sbin** : Essential commands that are only for the system administrator.
- **/tmp** : Temporary files and storage space. Don't put anything in here that you want to keep. Most Linux distributions (including Fedora) are set up to delete any file that's been in this directory longer than three days.
- **/usr** : Programs and data that can be shared across many systems and don't need to be changed.
- **/var** : Data that changes constantly (log files that contain information about what's happening on your system, data on its way to the printer, and so on).

perry@nugget1:~

File Edit Settings Help

[perry@nugget1 perry]$

Some of the basic commands you should learn are the ones that help you navigate the file system.

Commands:

/ **(root directory)**

/root – home directory of the user root

pwd – you can see your home directory

df – to see disk space available

cd – to change to different directory or to go back to home dir

.. - move to parent directory

ls – list the contents of a directory; Options: -l (more info)

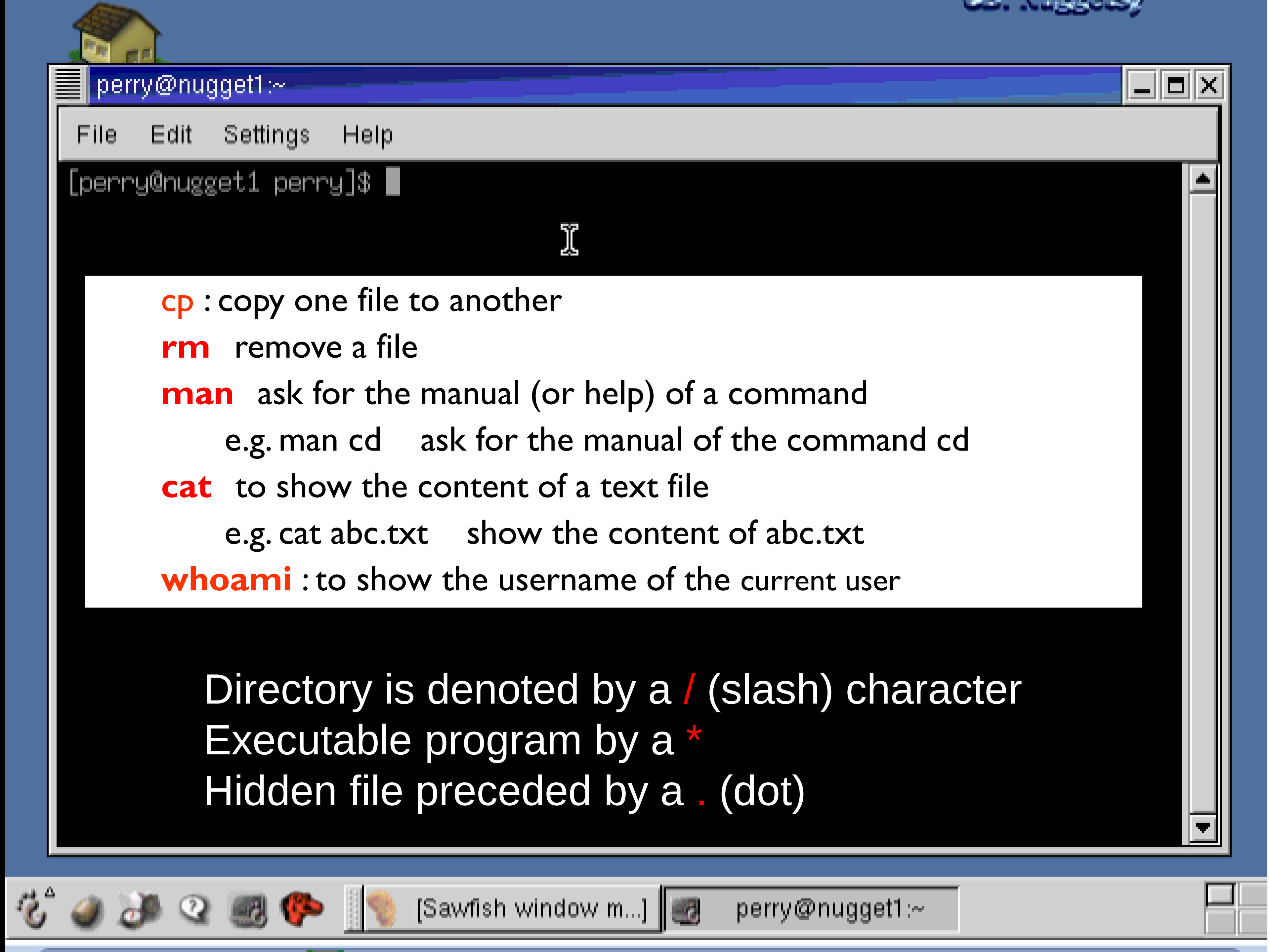
-a (displays hidden files)

-t (sort by time)

-r (oldest first)

Example: ls –ltr : display an long list of files that are sorted by time, display the oldest ones first

[Sawfish window m...] perry@nugget1:~



cp : copy one file to another

rm remove a file

man ask for the manual (or help) of a command

e.g. man cd ask for the manual of the command cd

cat to show the content of a text file

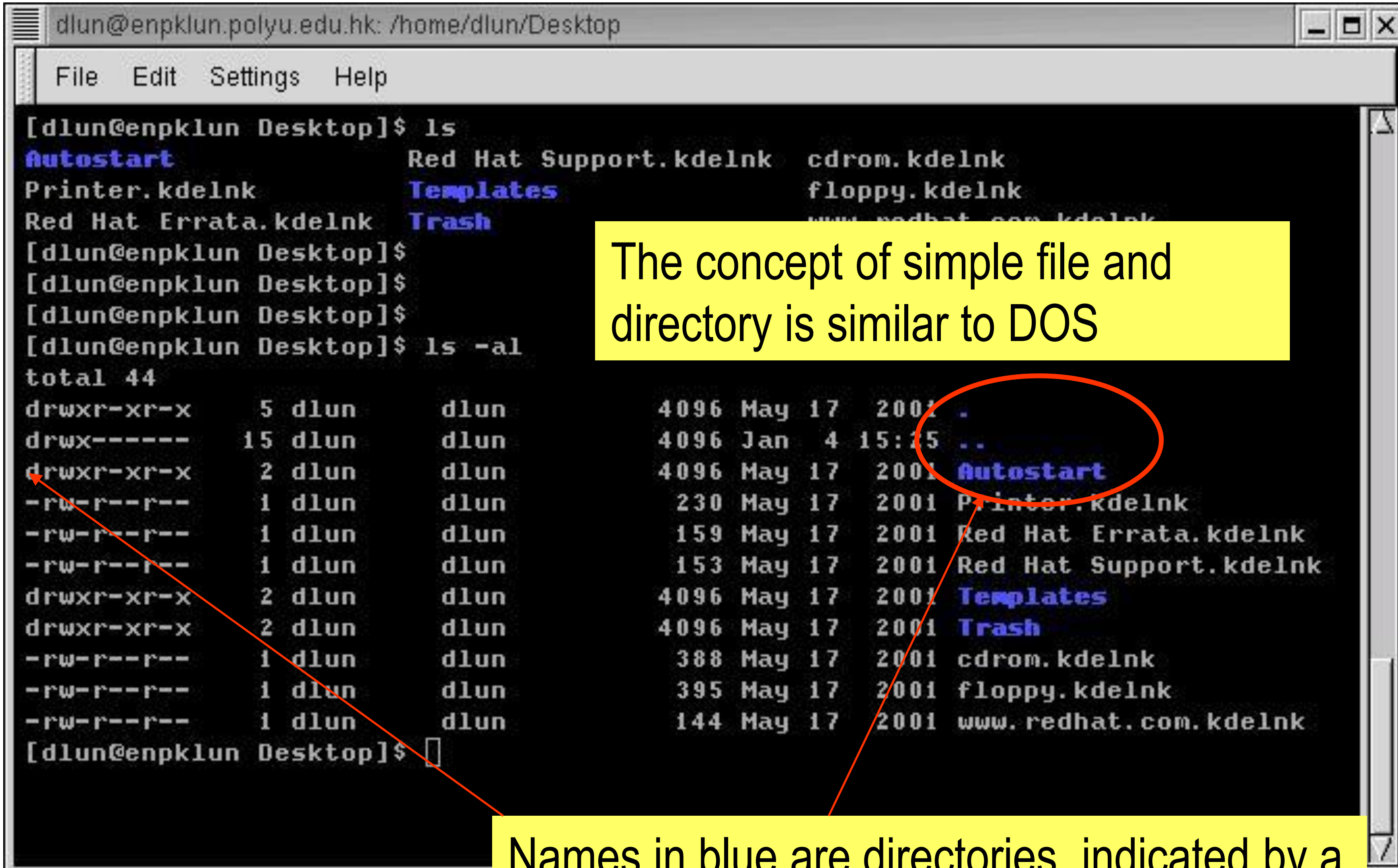
e.g. cat abc.txt show the content of abc.txt

whoami : to show the username of the current user

Directory is denoted by a / (slash) character

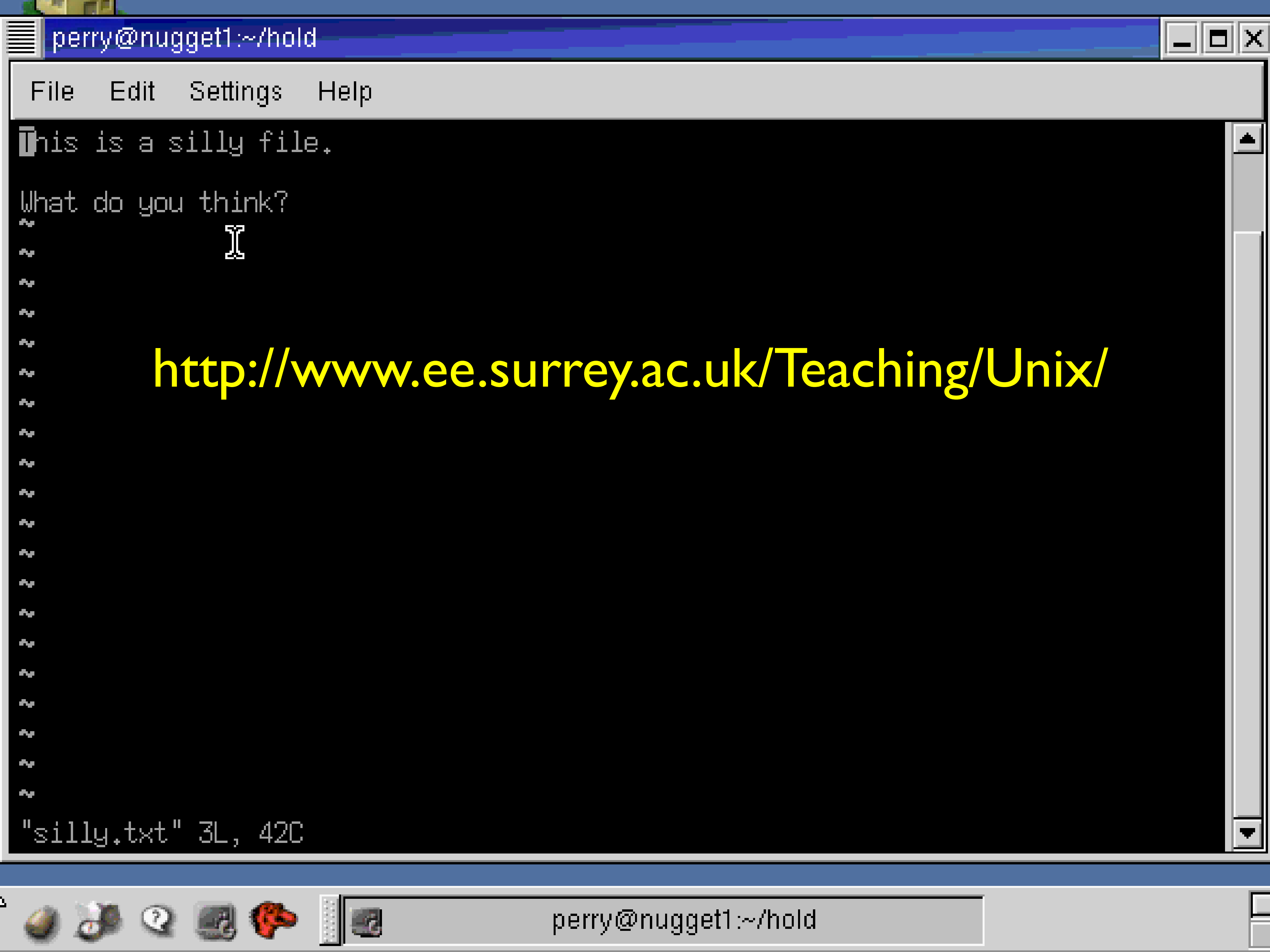
Executable program by a *

Hidden file preceded by a . (dot)



The concept of simple file and directory is similar to DOS

Names in blue are directories, indicated by a letter d at the beginning of the line


perry@nugget1:~/hold
File
Edit
Settings
Help
This is a silly file.
What do you think?
http://www.ee.surrey.ac.uk/Teaching/Unix/
"silly.txt" 3L, 42C
perry@nugget1:~/hold

Course Programming Environment Homework Assignment

- Install **Ubuntu 16.04 LTS** on your laptop/use lab PC
- **Learn Linux command line basics**
- **Learn/practice C/C++ programming in Linux**
- Project 0 in Moodle

Reference Reading:

- **Exploring BeagleBone. 2nd edition: Chapter 3**

7.322 PCs Login: student

Password: robot7322

Any Questions?