# C2115
# Practical introduction to supercomputing

**Lesson 6**

Petr Kulhánek

kulhanek@chemi.muni.cz

National Centre for Biomolecular Research, Faculty of Science
Masaryk University, Kamenice 5, CZ-62500 Brno

# Content

## ➢ Computer architecture

**CPU, memory, graphics system, disks, network, peripherals**

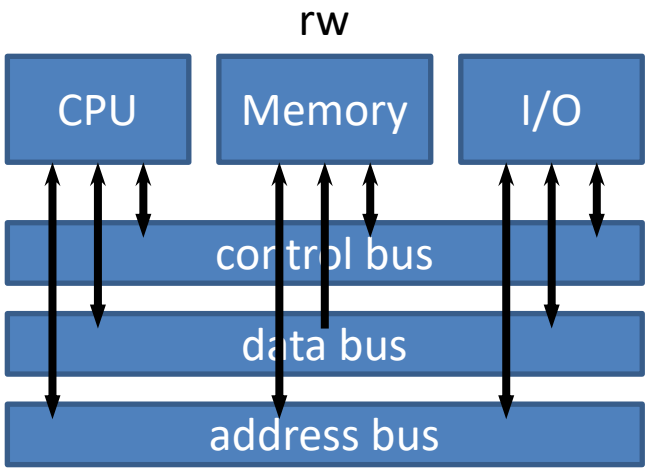
## ➢ Submitting jobs

**scripts vs programs, processes, nohup, screen, VNC**

# Computer architecture

# Overview

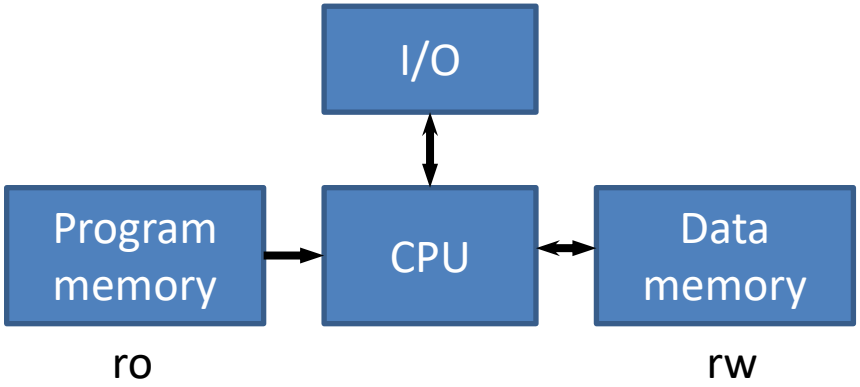
1945 von Neumannova architecture




- the program can modify itself
- program and data cannot be loaded at the same time

**John von Neumann**, originally a Hungarian mathematician working in the United States

1944 Hardwardskand architecture

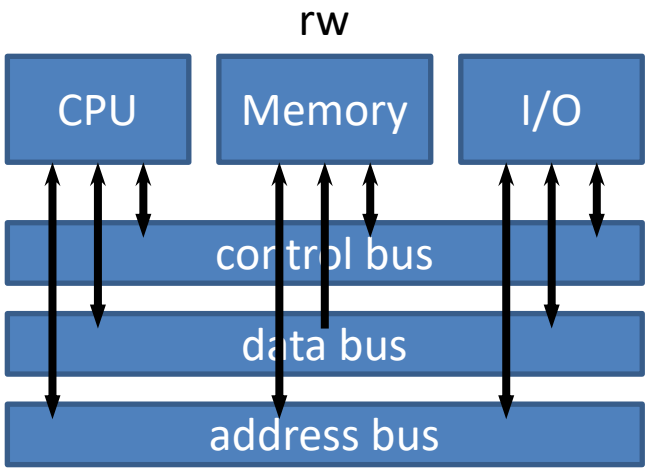



- the program cannot be modified
- the program and the data can be read at the same time

**Harvard Mark I** - computer made from relays, 24 bit instructions

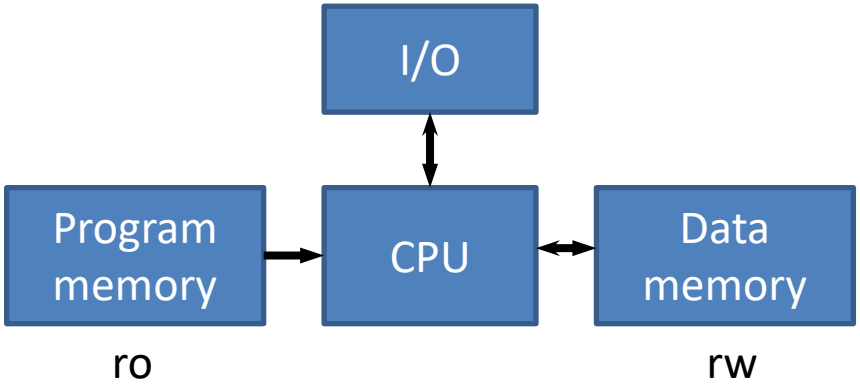
# Overview

1945 von Neumannova architecture




- the program can modify itself
- program and data cannot be loaded at the same time

1944 Hardwardskand architecture



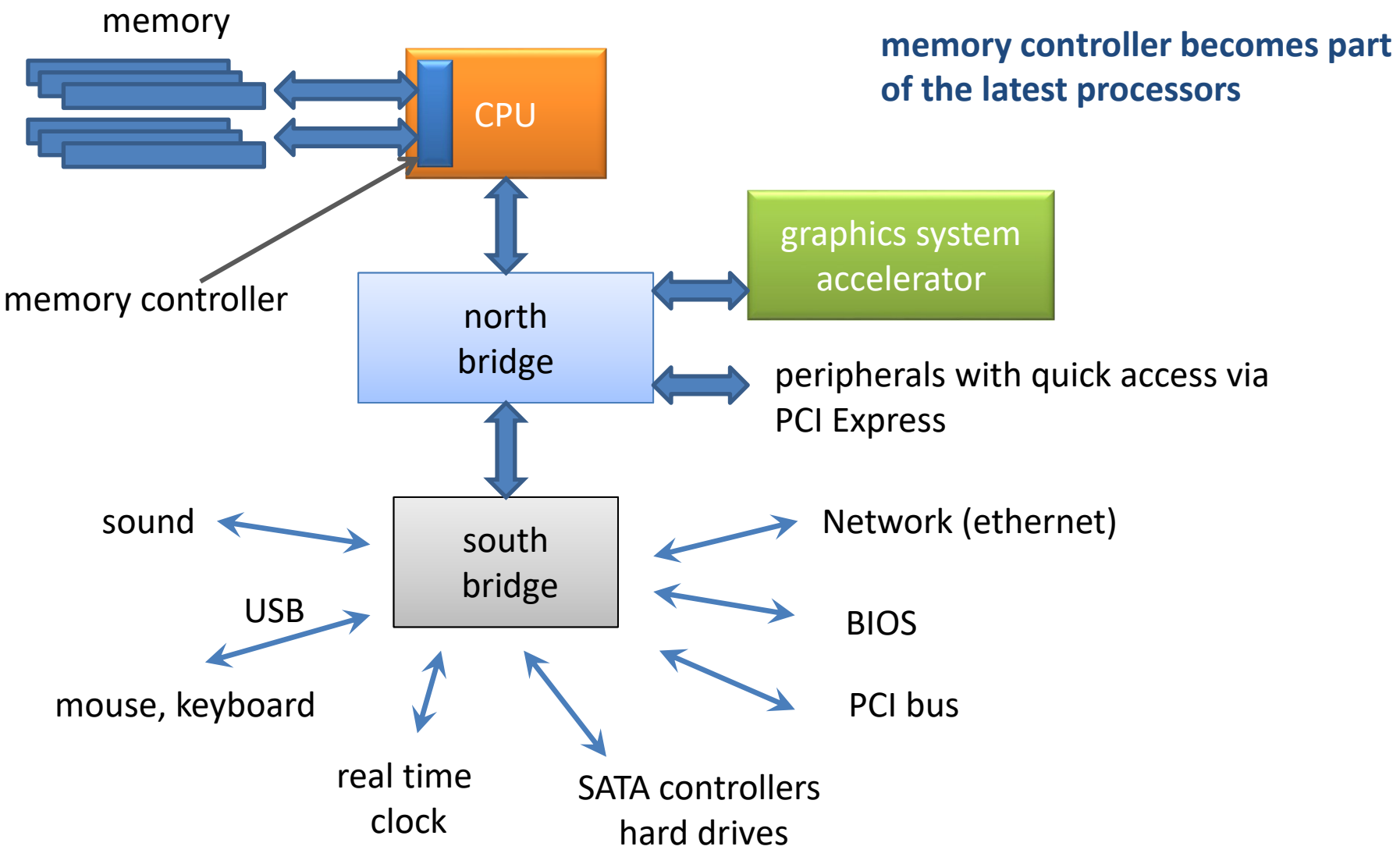

- the program cannot be modified
- the program and the data can be read at the same time

**in today's computers, both architectures are combined**

**John von Neumann**, originally a Hungarian mathematician working in the United States

**Harvard Mark I** - computer made from relays, 24 bit instructions

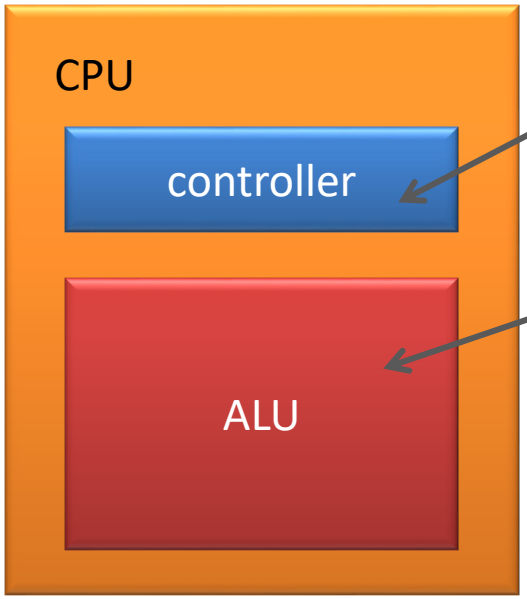
# Typical computer scheme



# CPU

**Processor** also **CPU** (**Central Processing Unit**) is an essential part of a computer; It is a very complex sequential circuit that **executes machine code** stored in the computer's operating memory. The machine code consists of individual machine instructions of computer programs loaded into the operating memory.

www.wikipedia.org



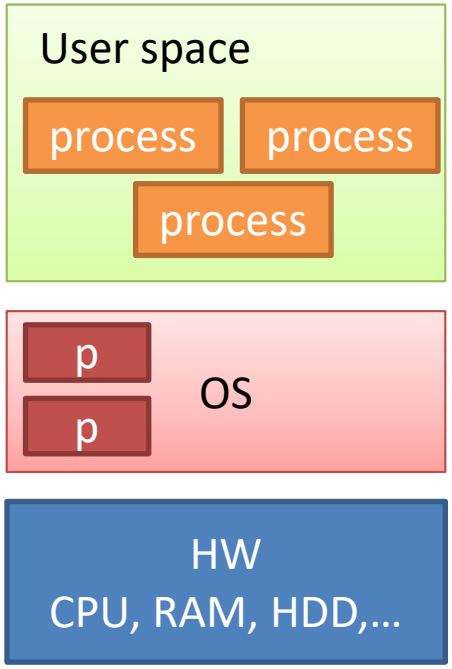

reads machine instructions and data and prepares their processing in ALU

ALU (arithmetic and logic unit), performs arithmetic operations, evaluates conditions

sequential processing of machine instructions is controlled by an internal clock cycle

# Computational tasks

# Jobs vs Processes




**Process,** in informatics, is the name for a **running computer program**. The process is **located in the computer operating memory** as **sequence of machine instructions executed by processor**. **Process management is performed by operating system**, which ensures their separate running, allocates them computer system resources, and allows the user to manage processes (start, end, etc.).

**Multitasking**, in informatics, indicates **ability of the operating system to perform several processes simultaneously** (at least seemingly). The core of the operating system very quickly changes processes running on the processor or processors (so-called context change), so that the computer user has the impression that they are all running at the same time.

edited from wikipedia.org

**A computational job is a process or group of processes.**

# Programs *vs* Scripts

**Program** is a set of machine instructions processed directly by the processor. The program is created by translating source code of the programming language.

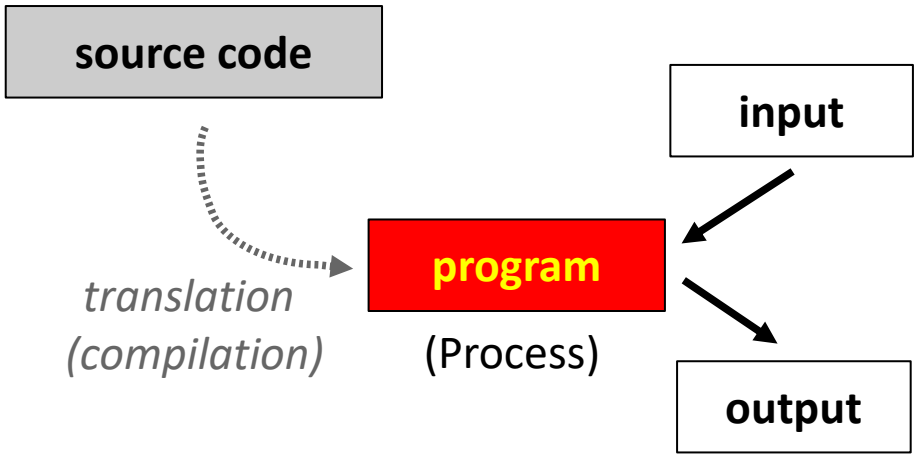
**Translated languages:**

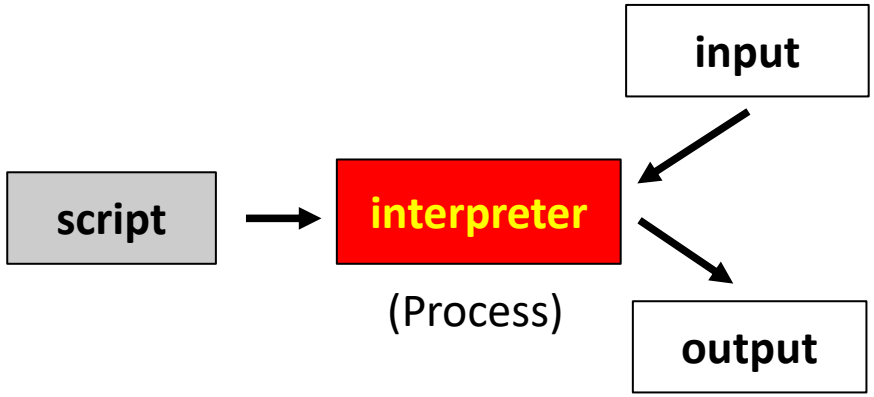
- **C / C ++**
- **Fortran**

**Script** is a text file containing commands and control sequences that are executed by interpreter of used scripting language.

**Scripting languages:**

- **bash**
- **gnuplot**
- **awk**
- JavaScript
- PHP



# Programs *vs* Scripts

Programs designed for **demanding scientific and technical calculations** are always written in **compilable programming languages.** These languages include:

- Fortran
- C/C ++

**Scripting languages** are either not used at all for demanding calculations or are used in **supporting parts of the calculations** which are not computationally intensive.

# Commands

| | |
|---|---|
| **top** | continuously displays processes sorted by CPU load (end with q key) |
| **ps** | lists processes running in given terminal or according to provided specification<br>(`ps -u user_name`) |
| **pstree** | lists processes (tree listing) |
| **type** | lists path to the standard application/command |
| **Kill** | sends a signal to process, can be used to terminate problematic programs |
| **time** | lists length of the process run |
| **ssh** | runs command on remote computer |
| **sleep** | starts a process that waits for specified time |
| **&** | runs process in background |
| **wait** | waits for background processes to complete |
| **nohup** | starts the process without interacting with the terminal |
| **screen** | screen terminal multiplexer |
| **vncserver** | starts VNC server } tigervnc |
| **vncwiever** | connects to VNC server |

# Exercise 1

1. Determine the percentage of programs written in Fortran, C/C++, and others that are listed on the following page:

   **http://en.wikipedia.org/wiki/List_of_quantum_chemistry_and_solid_state_physics_software**

Display the result in the form of a pie chart.

**Note:** use the grep and wc commands to solve the exercise.

# Submitting jobs

- nohup
- screen
- tigervnc

# Exercise 2

1. Log in to the wolf39 node in two terminals. In one of them, run the command:

```
$ top -u <login_name>
```

2. Run pi program (it is part of the system) in the second terminal. What is it used for?
3. Write a script that calculates the number pi with an accuracy of 10 to 10,000 digits. Save the resulting pi numbers into the pi.txt file.
4. Run the script in the background and log out of the wol39 node.
5. What will happen?
6. Log in to the wolf39 node again. Kill the running job.
7. Modify the script so that the result is printed to standard output. Run the script again, this time in the foreground.
8. What happens if the network connection is lost? What is the purpose of the SIGHUP signal?

# Running jobs

To run a job that is immune to termination by the SIGHUP signal, you can use:

- nohup (for non-interactive tasks)
- screen or tmux (for interactive tasks)
- VNC session (for GUI tasks)
- **batch system**

Commands nohup, screen, tmux are used only to run jobs on computers where a batch system is not available.
In supercomputer centers, these commands can only be used in batch jobs (screen, VNC) or to run service tasks (which are not computationally intensive) on the front nodes.

# nohup

Typical use:

```
$ nohup ./my_script &
```

Script output *my_script* attaches to the file *nohup.out*.

**Note:**

- Processes running in the background (started with nohup) may be terminated by the system administrator after the last user session is closed.
- This can be prevented by changing the configuration:

```
/etc/systemd/login.conf
[Login]
KillUserProcesses=no
```

# screen

Typical use:

```
$ screen
  win1$ ./my_script
  win1$                          # Ctrl+a d – leaves session
$ screen –list                   # lists open sessions
$ screen –r session.id           # connects  into session
```

**Note:**

- Background processes (screen and processes running in its instance) can be terminated by the system administrator after the last user session is closed.
- This can be prevented by changing the configuration:

```
/etc/systemd/login.conf
[Login]
KillUserProcesses=no
```

# Exercise 3

1. Log in to node wolf39.
2. Open screen program session. Run the script to calculate the number pi in it.
3. Disconnect from the session and log out of the wolf39 node.
4. Log in to node wolf39 and resume the session in the program screen. What are you observing?

# VNC

**Virtual Network Computing (VNC)** is a graphical program that allows remote connection to a graphical user interface using a computer network.

VNC works as a client-server, where the server creates a graphical desktop in the computer operating memory and communicates over the network with the client, which displays the desktop to the user (usually on another computer).

Remote Frame Buffer protocol (RFB) is used for communication and the aim is to minimize the volume of data transferred between the client and the server and thus enable communication even over slower data lines (e.g., via the Internet).

**VNC does not offer secure data transfer by default!!!!**

**Note:**

- Processes running (vncserver) in the background can be terminated by the system administrator after the last user session.
- This can be prevented by changing the configuration:

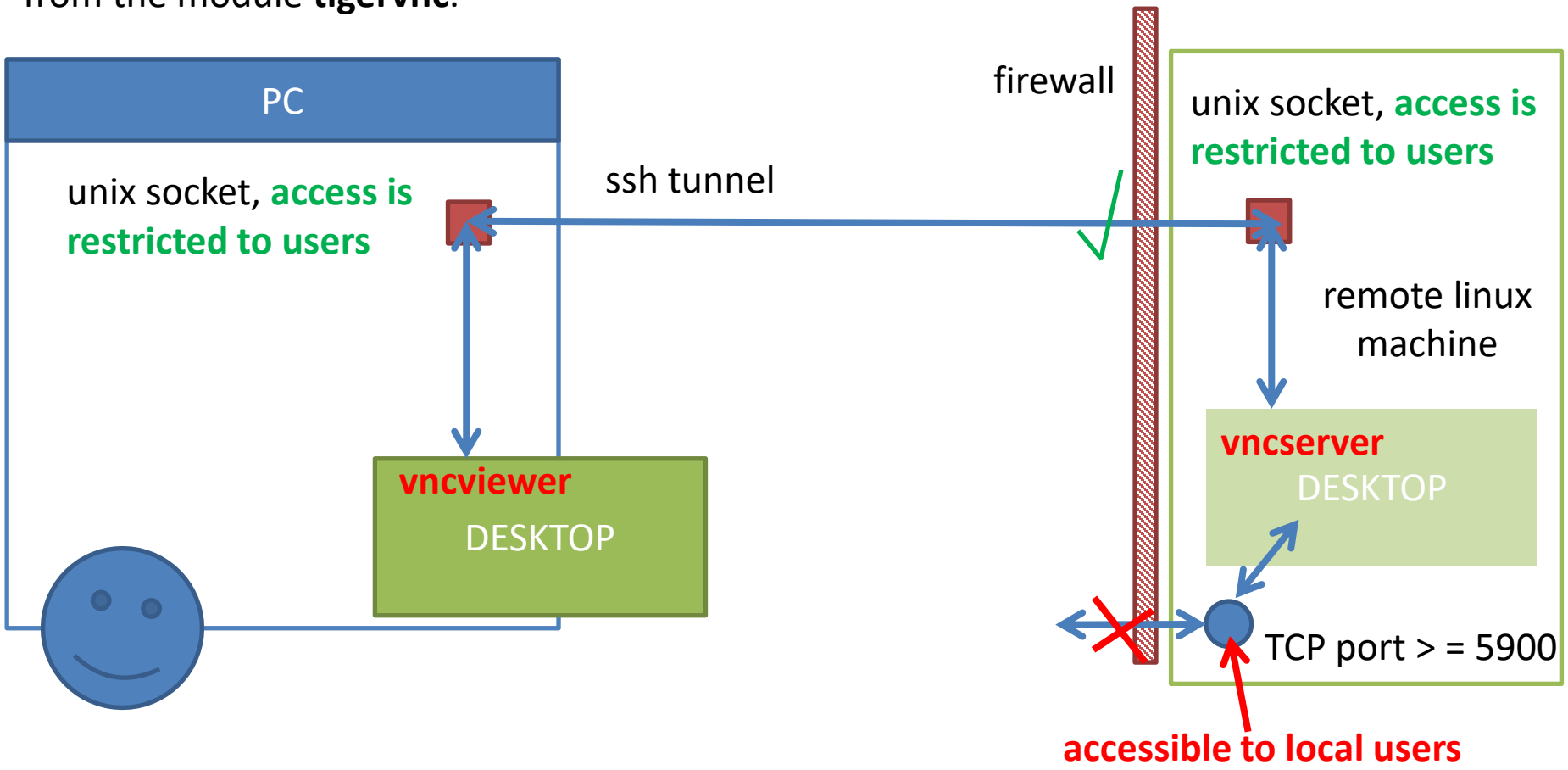
```
/etc/systemd/login.conf
[Login]
KillUserProcesses=no
```

wikipedia.cz

# VNC - transmission security

Securing the transmission is quite complicated. Either requiring the creation of an X.509 certificate or creating a a secure line using ssh tunneling.

The second option is offered by modified versions of **vncviever** and **vncserver** programs from the module **tigervnc**.



# tigervnc

1

**server**

```
wolf39$ module add tigervnc
wolf39$ vncserver
..
VNCID:  kulhanek@wolf39.ncbr.muni.cz:1
..
wolf39$
```

2

**client**

```
wolf01$ module add tigervnc
wolf01$ vncviewer kulhanek@wolf39.ncbr.muni.cz:1
```

# Exercise 4

1. On node wolf39, run vncserver from the module tigervnc and connect to it from your workstation. In the VNC session, run the VMD program and open the chitin nanofiber model. Leave the seating open.
2. Log in to the wolf39 node and run the VMD program on it (using the display export) and display the chitin nanofiber model again.
3. Compare the speed of interactive work with the VMD program, which displays using VNC and directly. Discuss the observations.