

C2115

Praktický úvod do superpočítání

6. lekce

Petr Kulhánek

kulhanek@chemi.muni.cz

Národní centrum pro výzkum biomolekul, Přírodovědecká fakulta
Masarykova univerzita, Kamenice 5, CZ-62500 Brno

➤ **Architektura počítače**

CPU, paměť, grafický systém, disky, síť, periferie

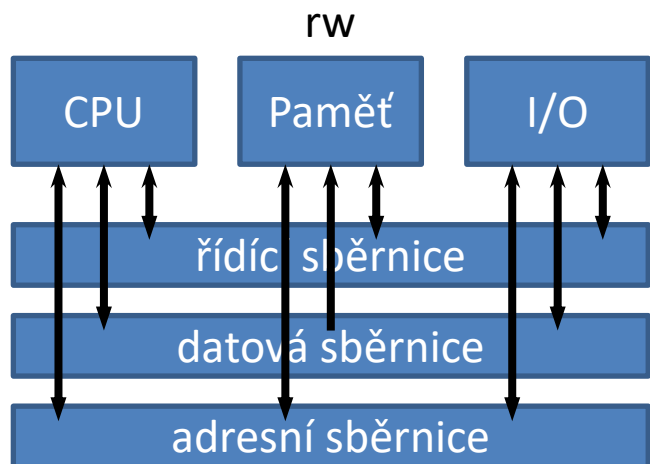
➤ **Spouštíme úlohy**

skripty vs programy, procesy, nohup, screen, VNC

Architektura počítače

Přehled

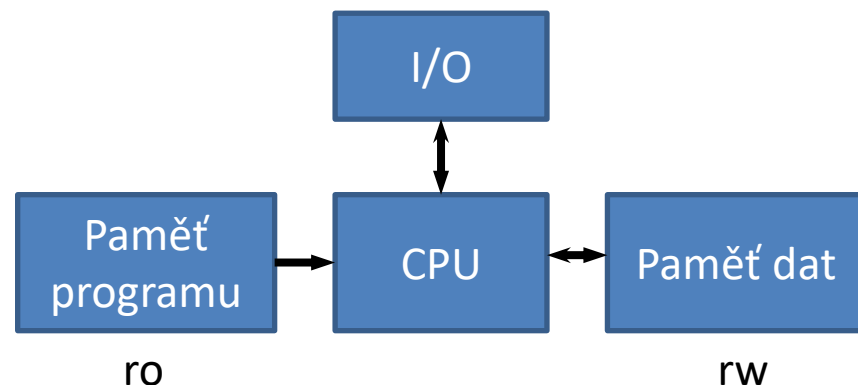
1945 von Neumannova architektura



- program může sebe modifikovat
- program a data nelze načítat současně

John von Neumann, původem maďarský matematik, působící ve spojených státech

1944 Hardwariská architektura

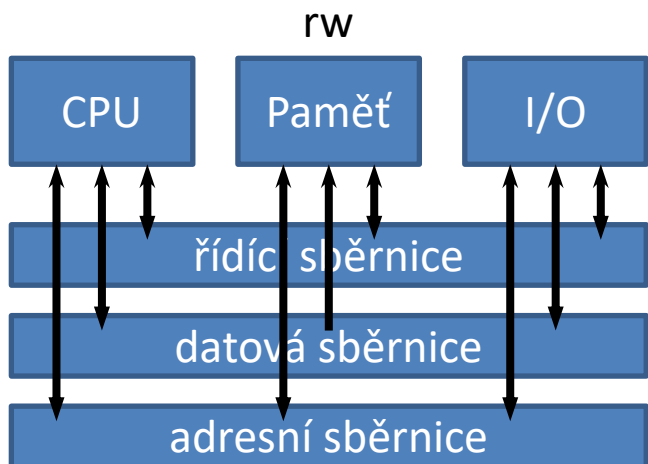


- program se nemůže modifikovat
- program a data se mohou načítat současně

Harvard Mark I - počítač složený z relé, 24 bitové instrukce

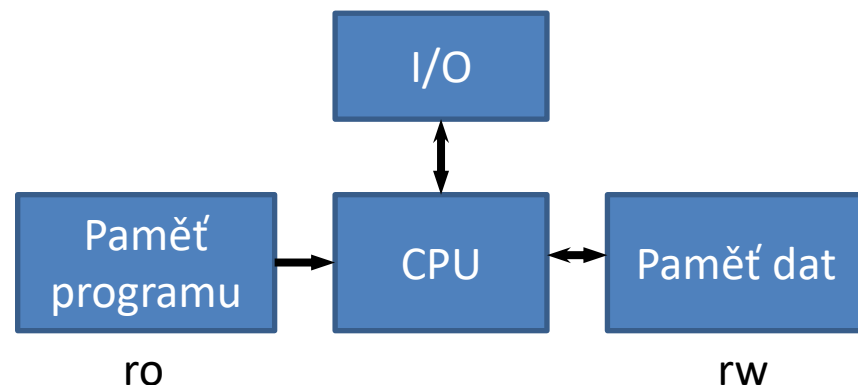
Přehled

1945 von Neumannova architektura



- program může sebe modifikovat
- program a data nelze načítat současně

1944 Hardwariská architektura



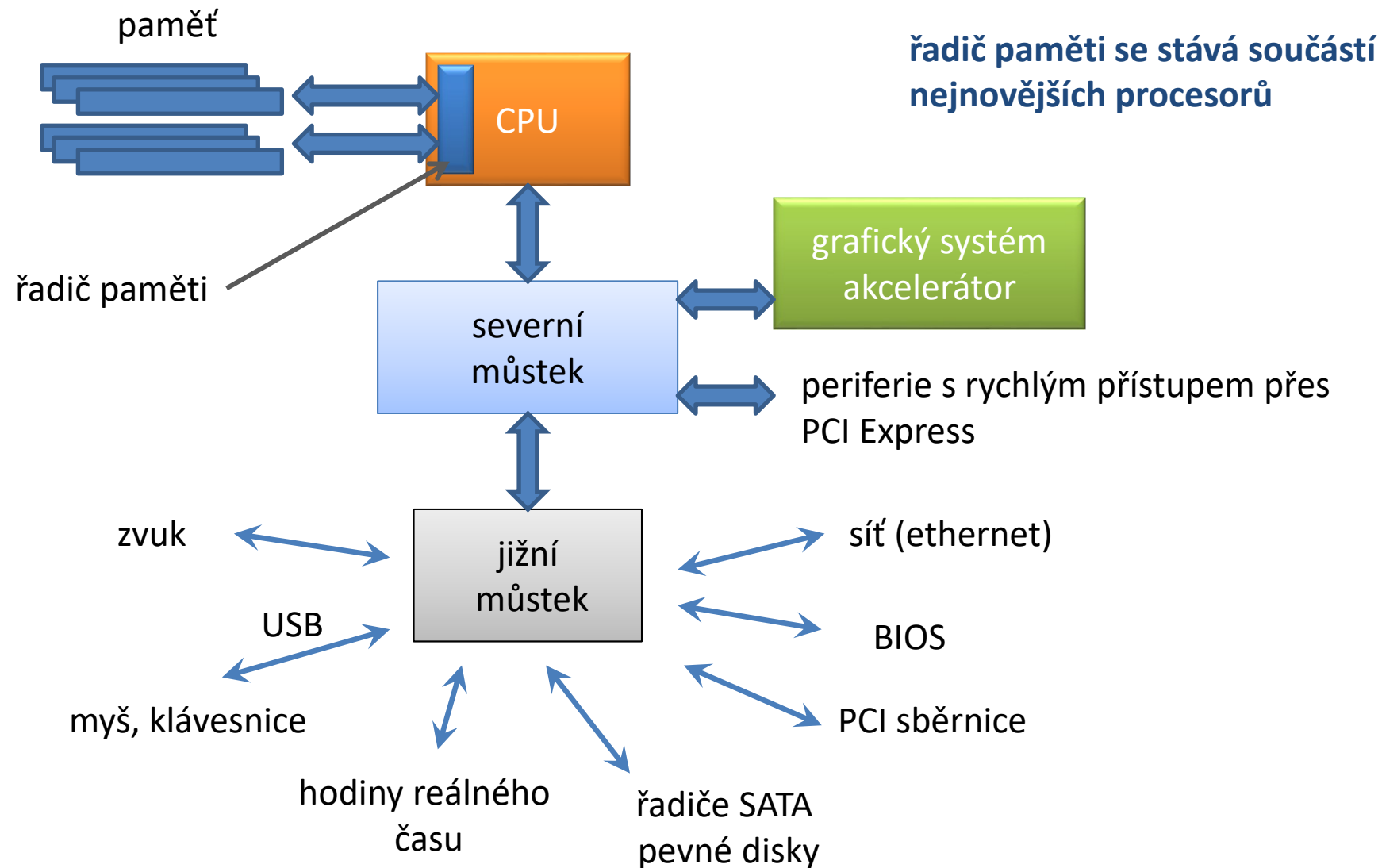
- program se nemůže modifikovat
- program a data se mohou načítat současně

v dnešních počítačích se kombinují obě architektury

John von Neumann, původem maďarský matematik, působící ve spojených státech

Harvard Mark I - počítač složený z relé, 24 bitové instrukce

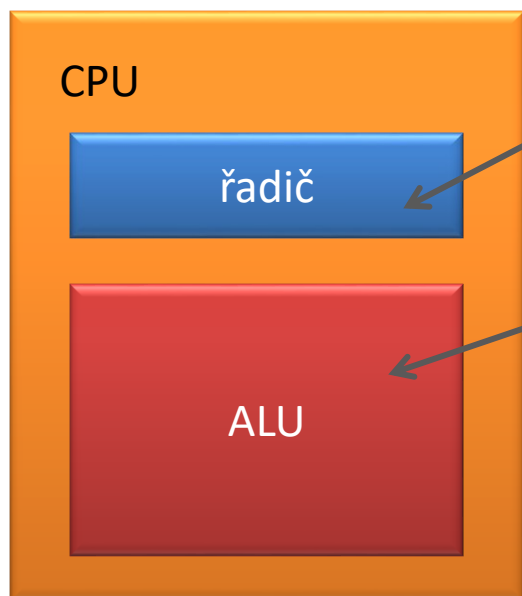
Typické schéma počítače



CPU

Procesor též **CPU** (anglicky **Central Processing Unit**) je základní součástí počítače; jde o velmi složitý sekvenční obvod, který **vykonává strojový kód** uložený v operační paměti počítače. Strojový kód je složen z jednotlivých strojových instrukcí počítačových programů nahraných do operační paměti.

www.wikipedia.org



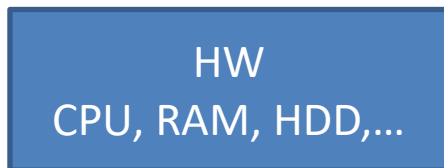
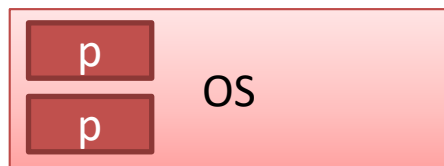
načítá strojové instrukce a data a připravuje jejich zpracování v ALU

ALU (arithmetic and logic unit), vykonává aritmetické operace, vyhodnocuje podmínky

sekvenční zpracování strojových instrukcí je řízeno vnitřním hodinovým taktem

Výpočetní úlohy

Úlohy vs Procesy



Proces (anglicky process) je v informatice název pro **spuštěný počítačový program**. Proces je **umístěn v operační paměti počítače** v podobě **sledu strojových instrukcí vykonávaných procesorem**. **Správu procesů vykonává operační systém**, který zajišťuje jejich oddělený běh, přiděluje jim systémové prostředky počítače a umožňuje uživateli procesy spravovat (spouštět, ukončovat atp.).

Multitasking (z angličtiny, multi = mnoho, task = úloha, používán ve víceúlohovém systému) označuje v informatice **schopnost operačního systému provádět několik procesů současně** (přinejmenším zdánlivě). Jádru operačního systému velmi rychle střídá na procesoru či procesorech běžící procesy (tzv. změna kontextu), takže uživatel počítače má dojem, že běží všechny současně.

upraveno z wikipedia.org

Výpočetní úloha je proces nebo skupina procesů.

Programy vs Skripty

Program je soubor strojových instrukcí zpracovávaných přímo procesorem. Program vzniká **překladem** zdrojového kódu programovacího jazyka.

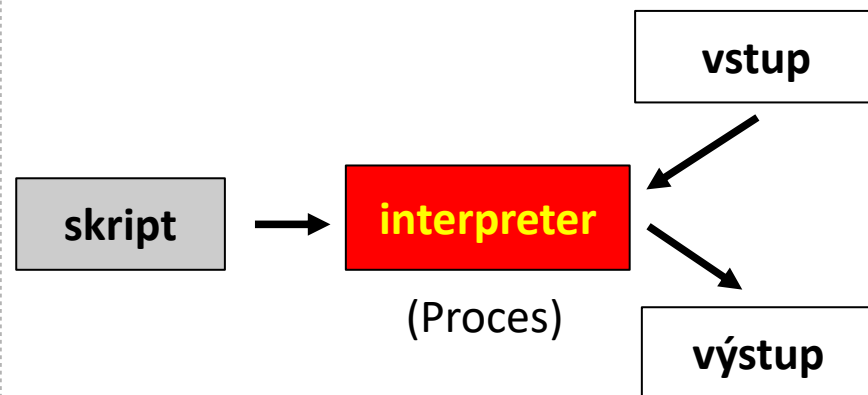
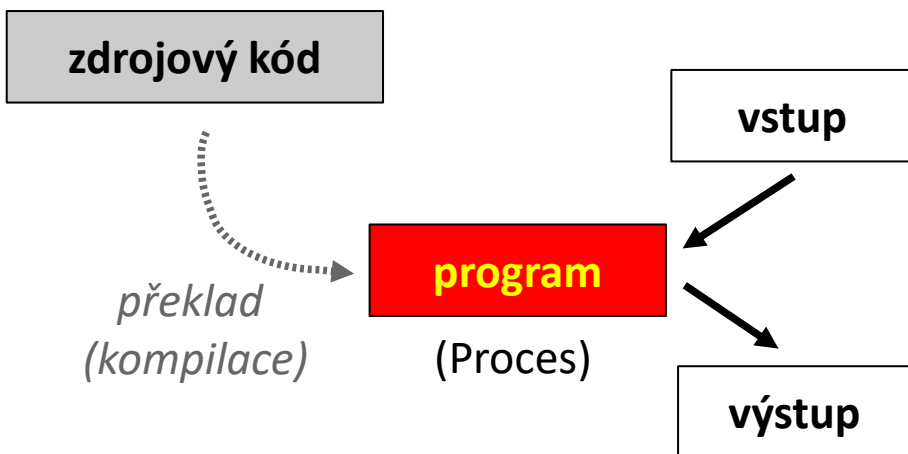
Překládané jazyky:

- C/C++
- Fortran

Skript je textový soubor obsahující příkazy a řídicí sekvence, které jsou vykonávány **interpretem** použitého **skriptovacího jazyka**.

Skriptovací jazyky:

- bash
- gnuplot
- awk
- JavaScript
- PHP



Programy vs Skripty

Programy určené pro **náročné vědeckotechnické výpočty** jsou vždy psané v **kompilovatelných programovacích jazycích**. Mezi tyto jazyky patří:

- Fortran
- C/C++

Skriptovací jazyky se pro náročné výpočty buď vůbec nepoužívají nebo se používají v **podpůrných částech výpočtů**, které nejsou výpočetně náročné.

Příkazy

top	průběžně zobrazuje procesy setříděné podle zátěže procesoru (ukončení klávesou q)
ps	vypíše procesy běžící v daném terminálu nebo podle zadaných specifikací (<code>ps -u user_name</code>)
pstree	vypíše procesy (stromový výpis)
type	vypíše cestu k standardní aplikaci/příkazu
kill	zašle signál procesu, lze použít k ukončení problematických programů
time	vypíše délku běhu procesu
ssh	spustí příkaz na vzdáleném počítači
sleep	spustí proces, který čeká po zadanou dobu
&	spustí proces na pozadí
wait	čeká na dokončení procesů na pozadí
nohup	spustí proces bez interakce s terminálem
screen	screen terminal multiplexer
vncserver	spustí VNC server
vncviewer	připojí se k VNC serveru
	} tigervnc

Cvičení 1

1. Určete procentuální zastoupení programů napsaných v jazyku Fortran, C/C++ a jiný, které jsou uvedeny na následující stránce:

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

Výsledek znázorněte ve formě výsečového grafu.

Poznámka: k řešení použijte příkazy grep a wc.

Cvičení 2

1. Přihlaste se na jiný uzel klastru WOLF ve dvou terminálech. V jednom spusťte příkaz:

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

2. Na tomto uzlu uložte kopii adresáře **/home/kulhanek/C2115/qm-long** do **/scratch/<login_name>**
3. Skript **run_me** spusťte na popředí. Monitorujte výstup a zatížení stroje (příkaz top z příkladu 1).
4. Co se stane, když terminál s běžící úlohou násilně ukončíte?
5. Znovu se přihlaste na uzel. Skript spusťte tentokrát na pozadí.
6. Co se stane, když terminál s běžící úlohou násilně ukončíte
7. Jak úlohy reagují na přerušení síťového spojení? K čemu slouží signál SIGHUP?

Spouštíme úlohy

- nohup
- screen
- tigervnc

Spouštění úloh

Pro spuštění úlohy, která je imunní vůči ukončení signálem SIGHUP, je možné použít:

- nohup (pro neinteraktivní úlohy)
 - screen nebo tmux (pro interaktivní úlohy)
 - VNC sezení (pro GUI úlohy)
 - **dávkový systém**
-
- Příkazy nohup, screen, tmux používáme pouze pro spuštění úloh na počítačích, kde není dostupný dávkový systém.
 - V superpočítačových centrech je možné tyto příkazy používat pouze v dávkových úlohách (screen, VNC) nebo pro spuštění servisních úloh (které nejsou výpočetně náročné) na čelních uzlech.

nohup

Typické použití:

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

Výstup ze skriptu *my_script* se připojí do souboru *nohup.out*.

Poznámka:

- Procesy běžící na pozadí (spuštěné pomocí nohup) mohou být ukončeny správcem systemd po ukončení posledního sezení uživatele.
- Tomu je možné zabránit změnou konfigurace:

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

screen

Typické použití:

```
$ screen
  win1$ ./my_script
  win1$                                     # Ctrl+a d - odpojí sezení
$ screen -list                             # vypise otevrena sezeni
$ screen -r session.id                     # pripoji se k sezeni
```

Poznámka:

- Procesy běžící na pozadí (screen a procesy běžící v jeho instanci) mohou být ukončeny správcem systemd po ukončení posledního sezení uživatele.
- Tomu je možné zabránit změnou konfigurace:

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

Cvičení 3

1. Přihlaste se na jiný uzel klastru WOLF.
2. Otevřete sezení programem screen. Spusťte v něm skript **run_me** (viz cvičení 2).
3. Odpojte se ze sezení a odhlaste se z uzlu.
4. Opětovně se přihlaste se na uzel a obnovte sezení v programu screen. Co pozorujete?

Virtual Network Computing (VNC) je grafický program, který umožňuje vzdálené připojení ke grafickému uživatelskému rozhraní pomocí počítačové sítě.

VNC pracuje jako klient-server, kde server vytváří grafickou plochu v operační paměti počítače a komunikuje přes síť s klientem, který plochu zobrazuje uživateli (většinou na jiném počítači).

Pro komunikaci se používá protokol RFB (anglicky remote framebuffer), jehož cílem je minimalizovat objem přenášených dat mezi klientem a serverem a umožnit tak komunikaci i přes pomalejší datové linky (např. přes Internet).

VNC ve výchozí konfiguraci nenabízí zabezpečený přenos dat!!!!

Poznámka:

- Procesy běžící (vncserver) na pozadí mohou být ukončeny správcem systemd po ukončení posledního sezení uživatele.
- Tomu je možné zabránit změnou konfigurace:

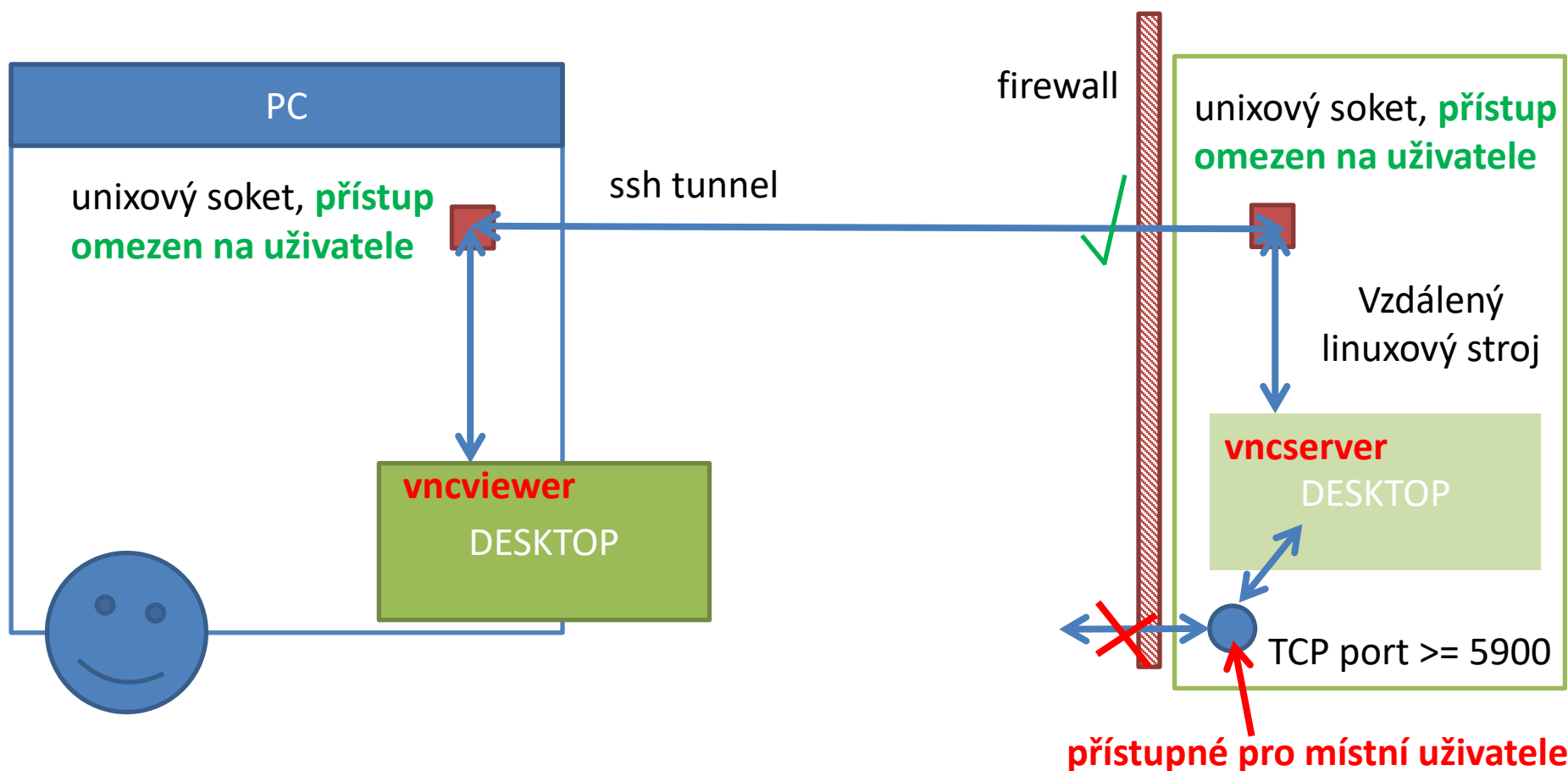
```
/etc/systemd/login.conf
[Login]
KillUserProcesses=no
```

wikipedia.cz

VNC - zabezpečení přenosu

Zabezpečení přenosu je poměrně komplikované. Buď vyžadující vytvoření X.509 certifikátu nebo zajištění zabezpečené linky pomocí ssh tunelování.

Druhou možnost nabízí upravené verze programů **vncviewer** a **vncserver** z modulu **tigervnc**.



tigervnc

1

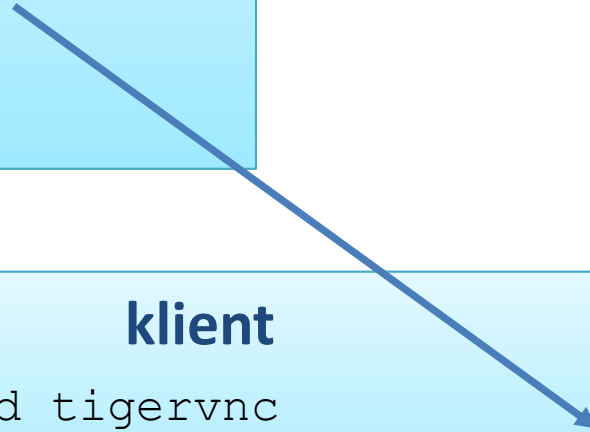
server

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

2

klient

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



Cvičení 4

1. Spusťte na jiném uzlu klastru WOLF vncserver z modulu tigervnc a připojte se k němu z vaší pracovní stanice. Ve VNC sezení spusťte program VMD a v něm otevřete model nanovlákná chitinu. Sezení nechte otevřené.
2. Na stejném uzlu klastru WOLF, který jste použili v předchozím úkolu, spusťte program VMD (za použití exportu displeje) a v něm opět zobrazte model nanovlákná chitinu.
3. Srovnejte rychlost interaktivní práce s programem VMD, který zobrazuje pomocí VNC a přímo. Pozorování diskutujte.