

# ***C2110 Operační systém UNIX a základy programování***

**3. lekce / modul 1**

**PS/2021 Prezenční forma výuky: Rev2**

**Petr Kulhánek**

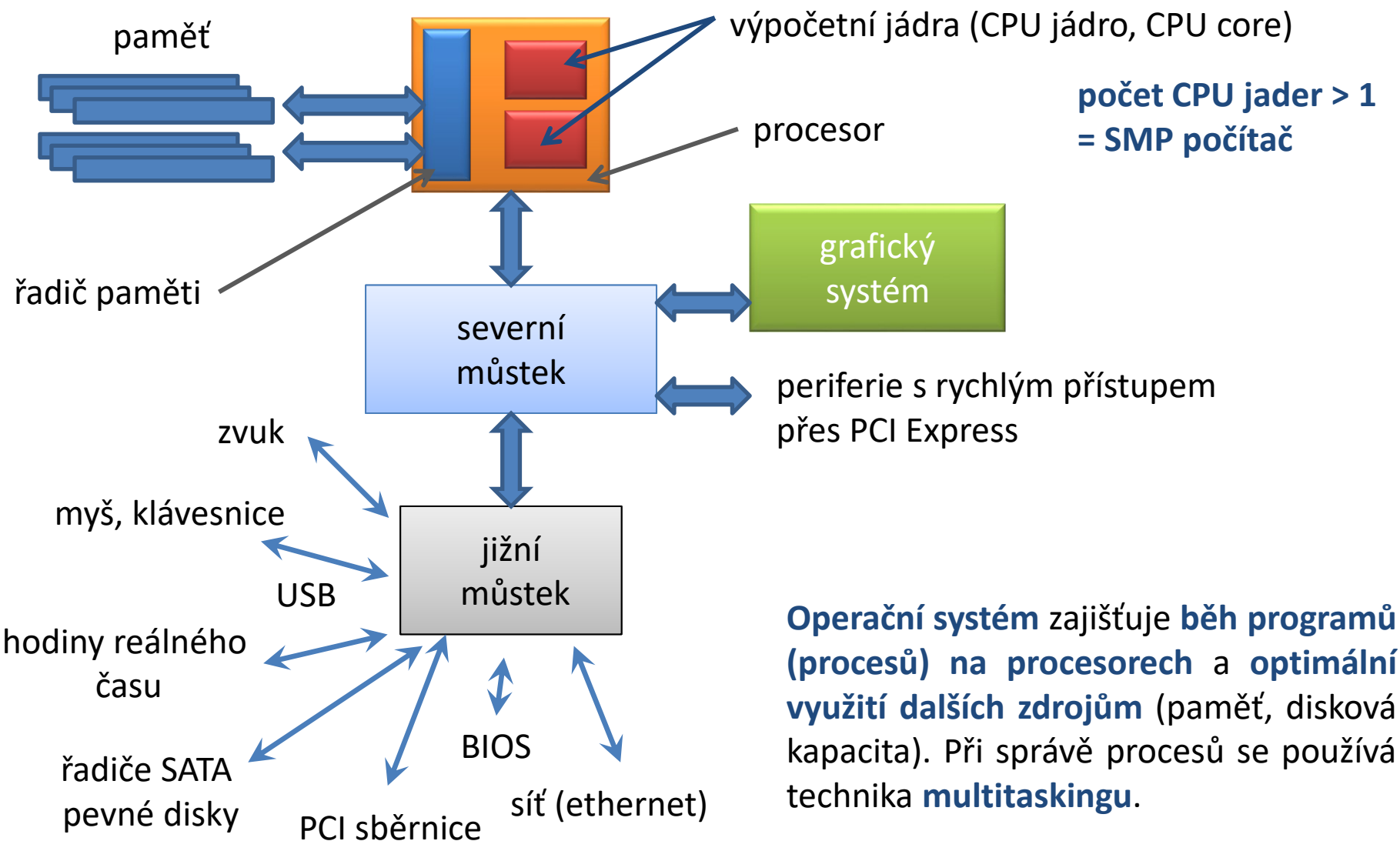
[kulhanek@chemi.muni.cz](mailto:kulhanek@chemi.muni.cz)

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

# Procesy

---

# Vnitřní schéma počítače



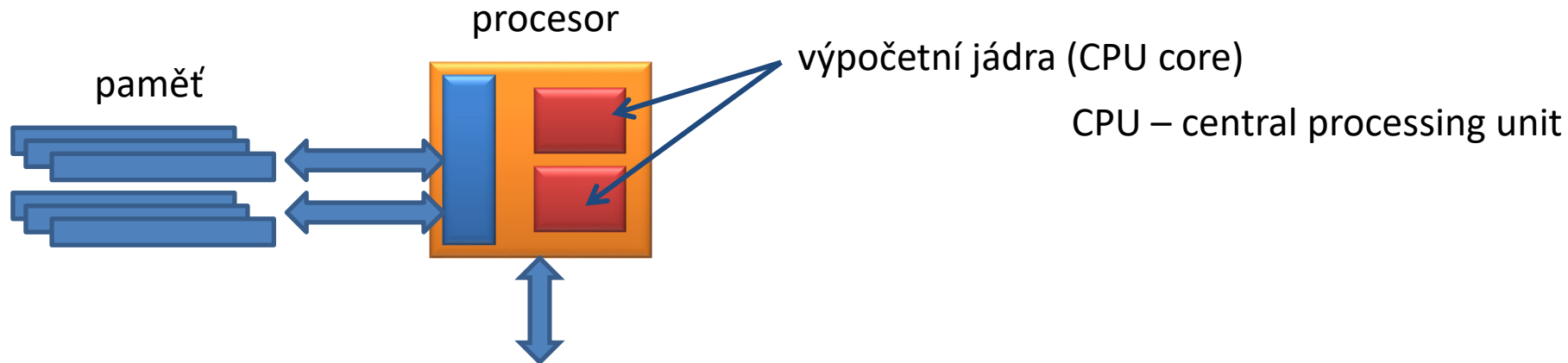
# Proces a multitasking

**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**. Obsahuje nejen kód vykonávaného programu, ale i dynamicky měnící se data, která proces zpracovává. Jeden program může v počítači běžet jako více procesů s různými daty (například vícekrát spuštěný webový prohlížeč zobrazující různé stránky). **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). Jádro operačního systému velmi rychle střídá na procesoru či procesorech běžící procesy (tzv. změna kontextu), takže uživatel počítače má dojem, že běží současně.

upraveno z wikipedia.org

# SMP – Symetrický multiprocessing



V minulosti se rychlost výkonu procesorů zvyšovala kromě lepší architektury i rychlostí zpracovávání instrukcí (frekvence procesoru), což v dnešní době naráží na fyzikální omezení používané technologie (spolehlivost, tepelné ztráty, ...). Dalším směrem bylo tedy uvedení více výpočetních jader (cca od roku 2005 pro x86 architekturu) na jednom fyzickém čipu. **Dnešní počítače jsou tak již běžně víceprocesorové.**

**Symetrický multiprocessing** (SMP, anglicky Symmetric multiprocessing) je v informatice označení pro druh **víceprocesorových systémů**, u kterých jsou všechny procesory v počítači rovnocenné. Zvýšení počtu procesorů, které v počítači sdílí stejnou operační paměť, vede **ke zvýšení výkonu počítače**, i když ne lineárním způsobem, protože část výkonu je spotřebována na režii (zamykání datových struktur, řízení procesorů a jejich vzájemnou komunikaci).

upraveno z wikipedia.org

# Přehled běžících procesů

## Procesy lze vypsat příkazy:

- ps** vypíše procesy běžící v daném terminálu nebo podle zadaných specifikací  
(`ps -u user_name`)
- top** průběžně zobrazuje procesy setříděné podle zátěže procesoru (ukončení klávesou q)
- pstree** výpis znázorňující hierarchii procesů

```
$ ps
  PID TTY          TIME CMD
 8763 pts/5        00:00:00 bash
 8852 pts/5        00:00:00 gimp
 8857 pts/5        00:00:00 ps
```

jméno spuštěného příkazu

číslo procesu

terminál, ve kterém proces běží

spotřebovaný strojový čas

# Přehled běžících procesů - top

Příkazem **top** je možné v pravidelných intervalech monitorovat běžící procesy. Běh příkazu se ukončuje klávesou **q** (quit).

odezva systému může být pomalá,  
pokud je používána swapovací paměť

zatížené CPU ve zlomku (1.0 = 100%)  
v poslední 1, 5 a 15 minutách

```
top - 13:05:58 up 16 days, 2:27, 2 users, load average: 2.95, 3.10, 3.03
Tasks: 150 total, 3 running, 147 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.3 us, 0.1 sy, 10.6 ni, 88.9 id, 0.1 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem: 8138412 total, 8005624 used, 132788 free, 210168 buffers
KiB Swap: 4194300 total, 168 used, 4194132 free. 7239188 cached Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3351	ivo	39	19	46784	29872	772	R	100.0	0.4	24:16.67	sc
30745	root	20	0	51732	1228	400	S	13.0	0.0	8:15.87	systemd-udevd
1	root	20	0	104664	4984	2736	S	6.5	0.1	6:36.74	init
383	root	20	0	19596	948	628	S	6.5	0.0	4:30.06	upstart-udev-br
2	root	20	0	0	0	0	S	0.0	0.0	0:00.70	kthreadd

číslo procesu

vlastník procesu

priorita

paměť

stav: S – sleeping, R – running,  
D – nepřerušitelný spánek (čeká na zařízení)

využití CPU a paměti

spotřebovaný CPU čas

jméno programu

# Spouštění příkazů a aplikací

Aby mohl shell zadaný příkaz spustit, potřebuje **znát cestu** k souboru, který obsahuje binární program nebo skript.

1. Cesta k příkazu se nejdříve hledá v tabulce s již použitými příkazy:

```
$ hash
```

```
hits      command
1         /bin/rm
3         /bin/ls
```

Tabulku lze smazat příkazem:

```
$ hash -r
```

2. Pokud není příkaz nalezen, hledá se v adresářích uvedených v systémové proměnné **PATH**, které jsou odděleny dvojtečkou.

```
$ echo $PATH
```

pořadí prohledávání adresářů

```
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:
/bin:/usr/games:/usr/local/games:/snap/bin:/usr/bin
```

3. Pokud není příkaz nalezen, je indikována chyba. V opačném případě je příkaz spuštěn a cesta uložena do tabulky.

```
$ prt
```

```
bash: prt: command not found
```



# Úprava proměnné PATH

## Manuální změna proměnné PATH

```
$ export PATH=/moje/cesta/k/mym/prikazum:$PATH
```



Cesta k adresáři obsahující příkazy, u kterých chci, aby byly přístupné bez uvádění cesty.

**Cesta se vždy uvádí absolutně!** (uvádění relativních cest je bezpečnostním rizikem)

oddělující znak

Původní hodnota proměnné PATH  
(nutné pro nalezení systémových příkazů)

## Automatizovaná změna proměnné PATH

Automatizovanou změnu proměnné PATH (a případně jiných systémových proměnných) provádí příkaz **module**.

```
$ module add vmd
```

**Budeme probírat v Lekci 4**

# Cesta k příkazům, dokumentace

**Cestu** k příkazu či aplikaci, pokud existuje, lze zjistit příkazem **type** nebo **whereis**

```
$ type ls
```

```
ls is aliased to `ls --color=auto`
```

**ls** je v shellu alias na příkaz **ls** s volbou **color**

```
$ whereis ls
```

```
ls: /bin/ls /usr/share/man/man1/ls.1.gz
```

příkaz **ls** je program uložen v souboru **/bin/ls** (man **ls**)

```
$ type pwd
```

```
pwd is a shell builtin
```

příkaz **pwd** je implementován jako vnitřní příkaz shellu (dokumentace příkazu **pwd** je v man **bash**)

```
$ whereis pwd
```

```
pwd: /bin/pwd /usr/include/pwd.h /usr/share/man/man1/pwd.1.gz
```

některé příkazy mohou mít více implementací (man **pwd**), přednost mají vnitřní příkazy shellu

# Procesy na popředí a pozadí

## Spouštění aplikací na popředí

```
$ gimp
```

procesy běžící **na popředí blokují terminál**, protože používají jeho standardní vstup a výstup

## Spouštění aplikací na pozadí

```
$ gimp &
```

procesy běžící **na pozadí neblokují terminál**

na konec (za argumenty a přesměrování) příkazu uvedeme ampersand

## Terminál (užitečné klávesové zkratky):

**Ctrl+Z** pozastaví běh procesu, další osud procesu lze kontrolovat pomocí příkazů:



**jobs** vypíše procesy, které shell spravuje

**bg** přesune proces do pozadí

**fg** přesune proces do popředí

**disown** zruší vazbu procesu na shell (proces není ukončen při ukončení shellu)

# Spouštění příkazů a aplikací ...

## Uživatelské programy a skripty

```
$ ./muj_script
```

```
$ ~/bin/my_application
```

jméno programu nebo skriptu udáváme včetně cesty (absolutní nebo relativní)

## Zrušení výpisu do terminálu

```
$ kwrite &> /dev/null
```

↙ přesměrování výstupu uvádíme na konec příkazu (za argumenty)

## Spouštění aplikací na pozadí

```
$ gimp &> /dev/null &
```

↙ na konec (za argumenty a přesměrování) příkazu uvedeme ampersand

# Signály a procesy

## Terminál (užitečné klávesové zkratky):



**Ctrl+C** běžícímu procesu zašle signál SIGINT (Interrupt), proces je ve většině případů násilně ukončen

## Příkaz kill:

```
$ kill [-signal] PID
```

číslo procesu, kterému se má signál zaslat (lze zjistit příkazem **ps**, **top**, **pstree**)

specifikace signálu: -N (číslo signálu), -NAME (jméno signálu), -SIGNAME (SIG+jméno signálu)

## Užitečné signály:

<b>TERM</b>	<b>15</b>	žádost o ukončení (proces na signál může reagovat)
<b>INT</b>	<b>2</b>	žádost o přerušení (ekvivalent <b>Ctrl+C</b> ) (proces na signál může reagovat)
<b>KILL</b>	<b>9</b>	ukončení ( <b>proces nemůže signál ignorovat, je násilně ukončen</b> )
<b>STOP</b>		pozastaví proces (proces nemůže signál ignorovat) (ekvivalent <b>Ctrl+Z</b> )
<b>CONT</b>		obnoví běh pozastaveného procesu (proces nemůže signál ignorovat)

# Přehled příkazů

<b>top</b>	průběžně zobrazuje procesy setříděné podle zátěže procesoru (ukončení klávesou q)
<b>ps</b>	vypíše procesy běžící v daném terminálu nebo podle zadaných specifikací ( <code>ps -u user_name</code> )
<b>pstree</b>	vypíše procesy (stromový výpis)
<b>type</b>	vypíše cestu k standardní aplikaci/příkazu (včetně vnitřních příkazů shellu)
<b>whereis</b>	vypíše cestu k standardní aplikaci/příkazu
<b>time</b>	vypíše délku běhu procesu
<b>sleep</b>	čeká po zadanou dobu
<b>kill</b>	zašle signál procesu, lze použít k ukončení problematických programů
<b>ssh</b>	spustí příkaz na vzdáleném počítači
<b>jobs</b>	vypíše procesy na pozadí
<b>fg</b>	převede proces do popředí
<b>bg</b>	převede proces do pozadí
<b>nohup</b>	spustí proces bez interakce s terminálem (C2115)
<b>wait</b>	čeká na dokončení procesů na pozadí (C2115)



# Cvičení 1

1. (Prezenčně) Otevřete nový terminál.
2. (Distančně) Otevřete nový terminál na pracovní stanici `wolf02.ncbr.muni.cz`
3. Vypište tabulku s již použitými příkazy (Výpis by měl být prázdný).
4. Spusťte příkaz **ls** a opět vypište tabulku s již použitými příkazy.
5. Kde se nalézá soubor obsahující program k příkazu **ls**. Použijte příkaz **type** a **whereis**. Jaký je mezi oběma příkazy rozdíl?
6. Jakou velikost a přístupová práva má soubor, který obsahuje program **ls**.
7. Vypište obsah proměnné `PATH` (echo `$PATH`).
8. Je ve výpisu cesta k adresáři, ve kterém je příkaz **ls**?
9. Vytvořte kopii souboru s programem **ls** do vašeho domovského adresáře pod názvem **my\_ls**.
10. Spusťte program **my\_ls** a porovnejte jeho výstup s příkazem **ls**. V čem se výstupy liší?
11. Soubor **my\_ls** smažte.

# Cvičení 2

1. (Prezenčně) Otevřete nový terminál.
2. (Distančně) Otevřete nový terminál na pracovní stanici `wolf03.ncbr.muni.cz`
3. Spustíte příkaz `sleep 60`. Co udává číslo 60?
4. Spustíte příkaz `sleep 300`.
5. Jeho běh ukončete pomocí `Ctrl+C`
6. Spustíte příkaz `sleep 300` a nechte jej běžet.
7. (Prezenčně) Otevřete nový terminál.
8. (Distančně) Otevřete nový terminál na pracovní stanici `wolf03.ncbr.muni.cz`
9. Vypište vaše běžící procesy (`ps -u username`)
10. Proces `sleep` ukončete pomocí příkazu `kill`
11. Body 6, 9, 10 opakujte pro různé signály (`SIGTERM`, `SIGINT` a `SIGKILL`)