C2110 *Operační systém UNIX a základy programování*

3. lekce / modul 1

PS/2020 Distanční forma výuky: Rev1

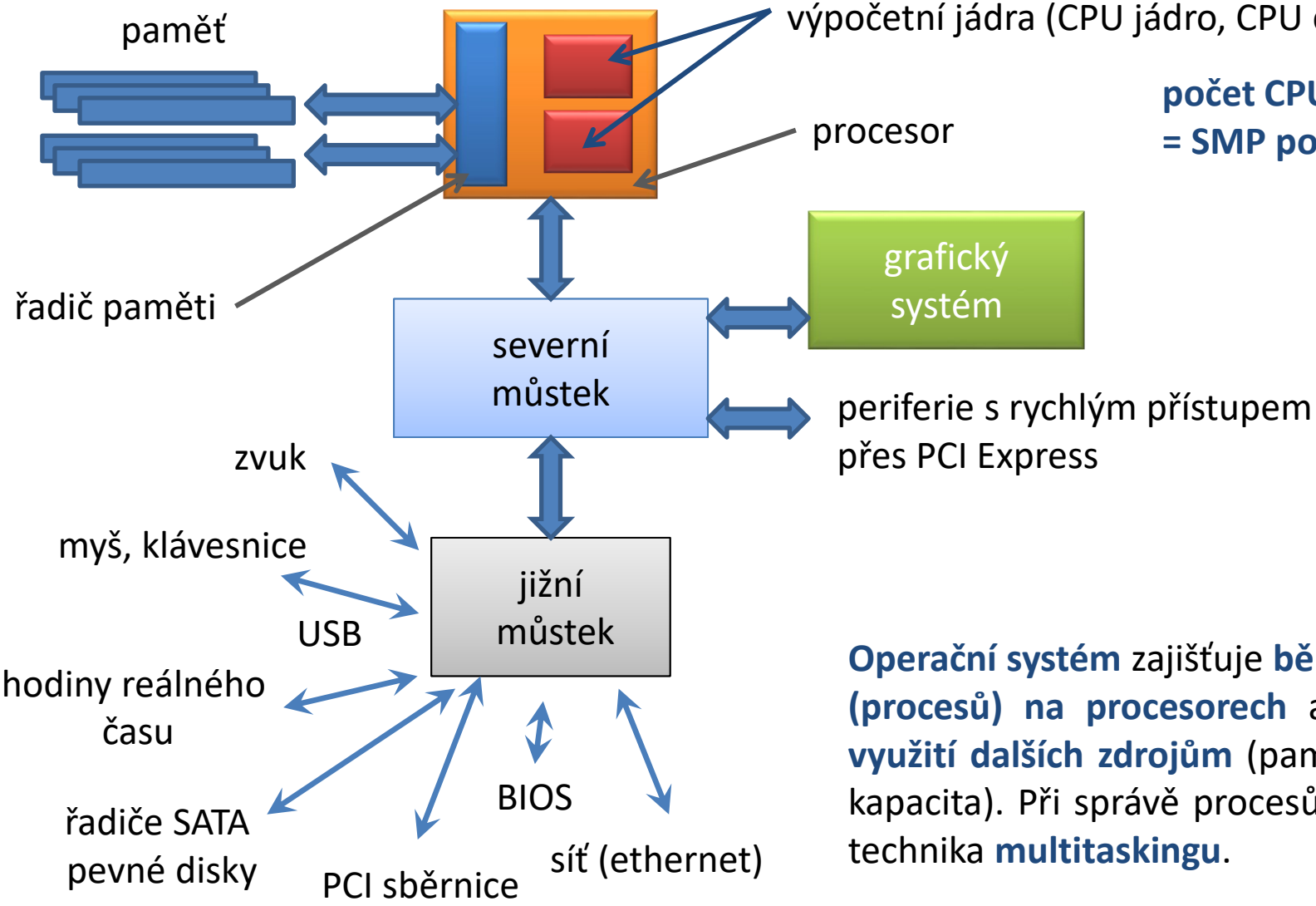
Petr Kulhánek

kulhanek@chemi.muni.cz

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

Procesy

Vnitřní schéma počítače



Operační systém zajišťuje **běh programů (procesů) na procesorech** a **optimální využití dalších zdrojům** (paměť, disková kapacita). Při správě procesů se používá technika **multitaskingu**.

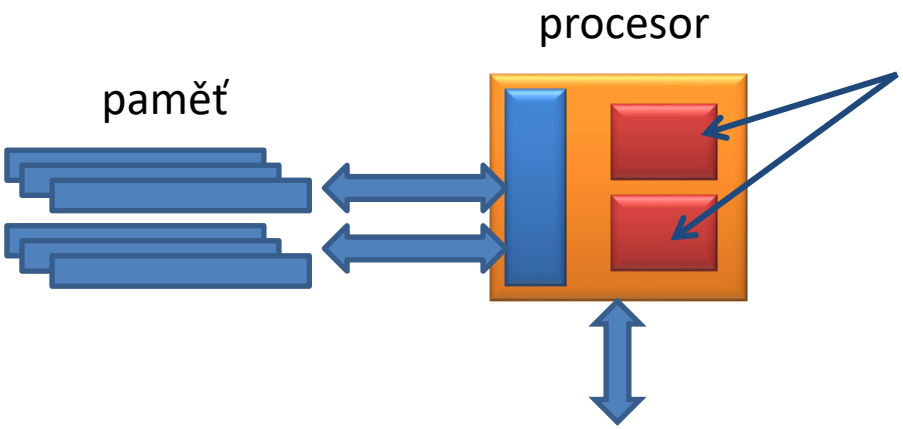
Proces a multitasking

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 multiprocesing



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 multiprocesing (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

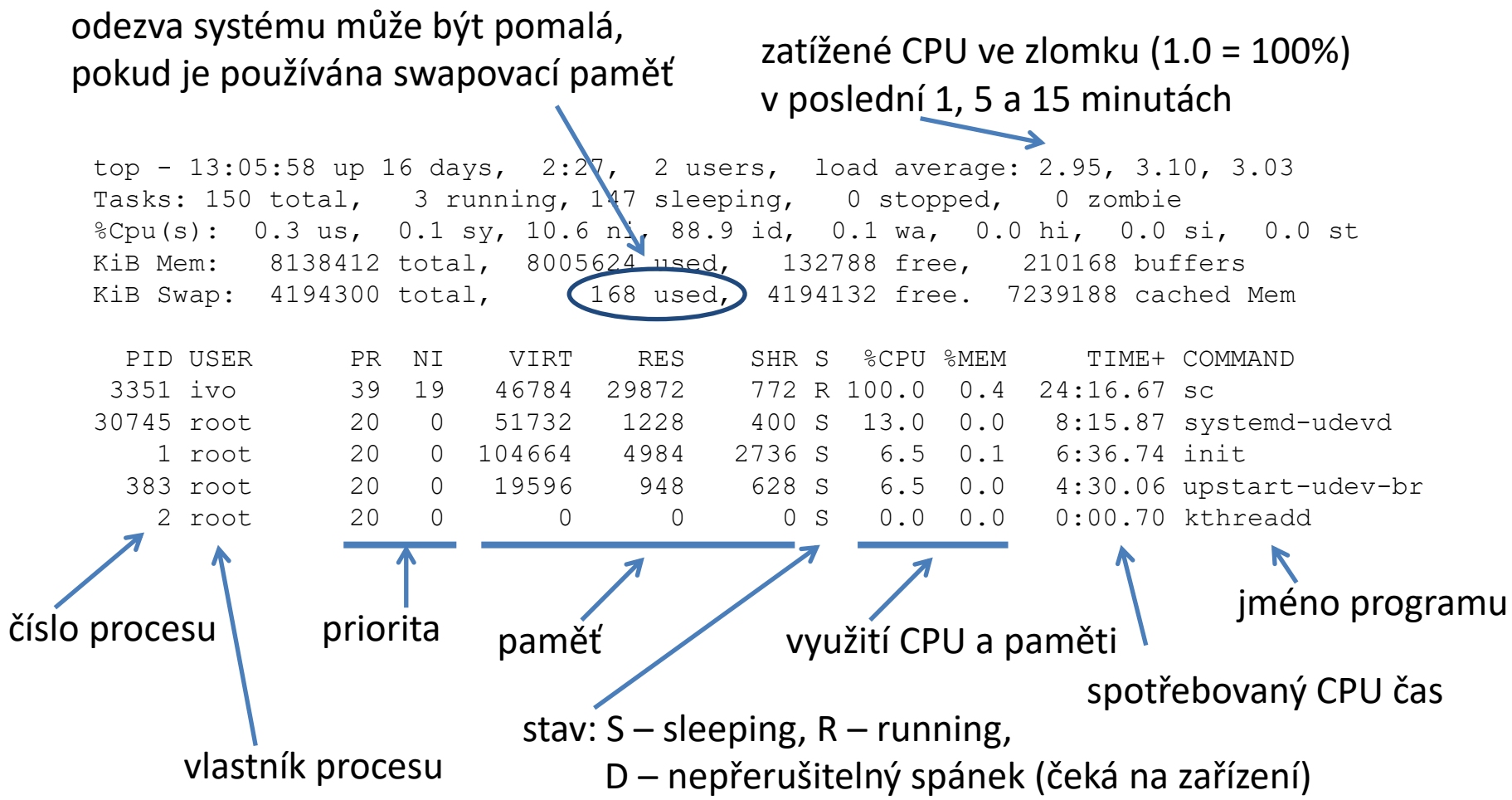
spotřebovaný strojový čas

číslo procesu

terminál, ve kterém proces běží

Přehled běžících procesů - top

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



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

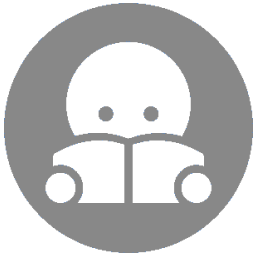
```
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:
/bin:/usr/games:/usr/local/games:/snap/bin:/usr/bin
```

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

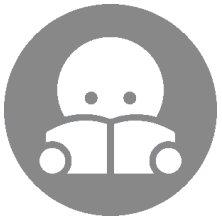
```
$ gimp &
```

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 pomoci 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:

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

Přehled příkazů

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

Cvičení 1

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

Cvičení 2

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