Úvod do operačních systémů

Milan Radojčić

SSPŠaG – Operační systémy

8. 11. 2025

Motivace

- V tomto předmětu se budeme zabývat tím, co uvnitř dělají operační systémy.[1]
- Proč se takovými věcmi vůbec zabývat?
- **Virtuální paměť** (mimo jiné) je důležitá pro pochopení útoků Spectre a Meltdown.
- Metadata **souborů systémů** jsou důležité např. ve forenzní analýze.
- **Synchronizace vláken** je důležitá pro psaní korektních paralelních programů.
- Zvědavost!

[1] Pro nás bude prakticky vždy OS znamenat kernel.

Úvodní kvíz

Operační systém[2] běží:

A) od začátku spuštění počítače, aktivně monitoruje procesy.
B) kdykoliv o to některé z vláken požádá.
C) při spuštění počítače a poté jen někdy, reaktivně (při tzv. přerušení).
D) vždy na pozadí na jednom z jader. Monitoruje chování procesů, aby mohl reagovat správným způsobem[3].
E) jen při vzniku a zániku vláken.

[2] Myslí se kernel.
[3] Např. ukončit proces, když provede něco nekalého.

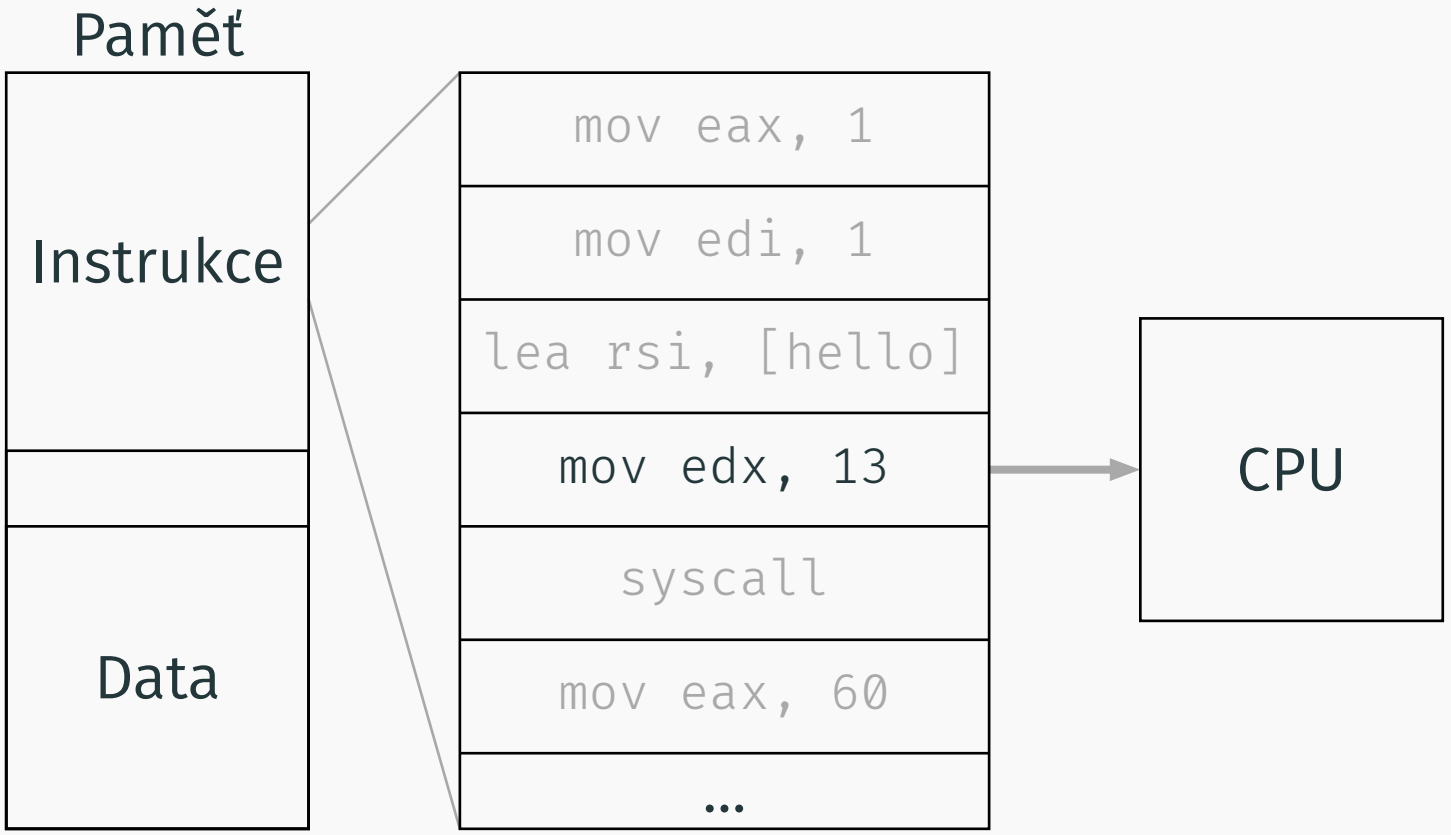
Úvodní kvíz - správné odpovědi

Operační systém běží:

A) od začátku spuštění počítače, aktivně monitoruje procesy.
B) **kdykoliv o to některé z vláken požádá.**
C) **při spuštění počítače a poté jen někdy, reaktivně (při tzv. přerušení).**
D) vždy na pozadí na jednom z jader. Monitoruje chování procesů, aby mohl reagovat správným způsobem.
E) jen při vzniku a zániku vláken.

Co dělá procesor?

- Procesor jednoduše **načítá** instrukce a **spouští** je jednu za druhou.[4]



- Jeden procesor spouští **pouze jeden proud instrukcí**.

[4] Moderní procesory dělají uvnitř různé šílenosti, ale navenek se stejně chovají takhle.

Malá poznámka k přístupu

- Na začátku se pokusíme vyhnout povídání o tom, **co se přesně děje** na úrovni CPU.
- Povíme si o tom až budete probírat procesor na APS.
- Půjdeme tedy z vyšších úrovní pomalu na nižší.

Systémová volání

- Jeden z hlavních způsobů jak interagujeme s OS jsou **systémová volání**.
- Je to vlastně požadavek pro OS, aby něco za nás udělal, např:
  - Spustil proces `fork`, `execve`; ukončil proces `kill`.
  - Otevřel soubor `open`, zapsal do něj data `write`.
- Systémová volání připomínají volání knihovních funkcí.[5]
- Můžeme si o nich přečíst víc pomocí `man`.

[5] Ale nejsou to volání knihovních funkcí, přesné rozdíly si rozebereme později.

strace

`strace` je jednoduchý program, který spustí jiný program a **vypíše systémová volání, které zavolal**.

Např. pro `strace cat /dev/null` získáme:

```
execve("/usr/bin/cat", ["cat", "/dev/null"], 0×7ffc0c033618)
 ...
openat(AT_FDCWD, "/dev/null", O_RDONLY)
 ...
exit_group(0)
```

strace

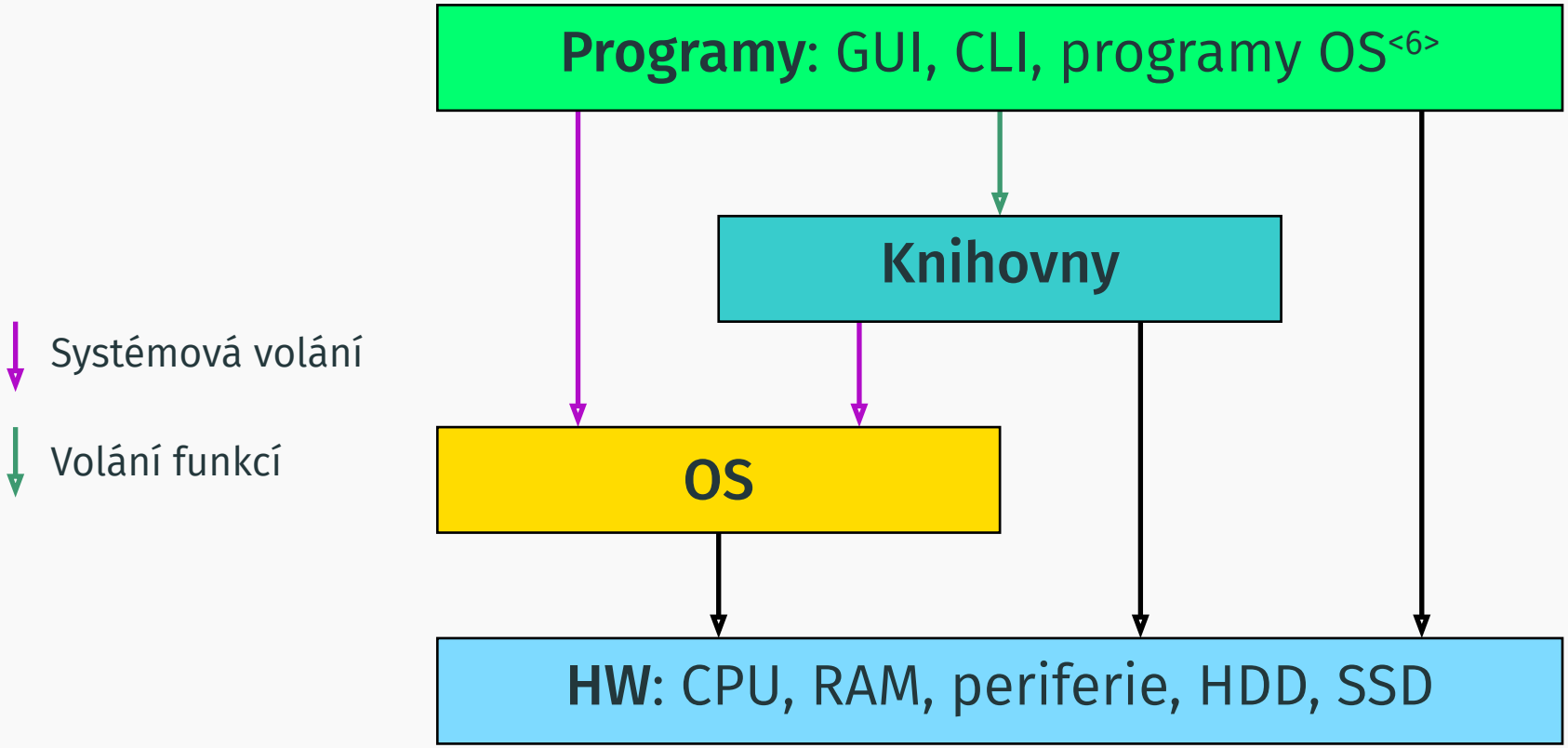
Výstup `strace ps -Flww -p 32923`

```
 ...
openat(AT_FDCWD, "/proc/32923/environ", O_RDONLY) = 3
read(3, "LC_TELEPHONE=cs_CZ.UTF-8\0TERMINA"... , 131072)
read(3, "", 128128)
close(3)
openat(AT_FDCWD, "/proc/32923/cmdline", O_RDONLY) = 3
read(3, "/usr/bin/zsh\0", 131072)
read(3, "", 131059)
close(3)
 ...
```

Vsuvka: magické soubory Linuxu

- Už asi znáte speciální soubory v `/dev` nebo `/proc`.
- Implementace takových souborů je vlastně jednoduchá!
  - OS nezajímá, jestli data má číst **soubor z disku** nebo informace z nějakých **vnitřních datových struktur** (v případě `/proc`).

Počítač z pohledu SW



[6] např. init

Historie OS: dávkové zpracování

- Historicky byly první počítače využívány k specifickým účelům:
  - fyzikální simulace,
  - vojenské výpočty (dráhy raket atd.),
  - finanční výpočty (predikce trhu apod.).
- Počítače **nebyly interaktivní**.
- OS byly v této době spíše knihovny.

IBM 709[7]

[7] https://en.wikipedia.org/wiki/IBM_709#/media/File:IBM_709_front_panel_at_CHM.agr.jpg

Historie OS: systémové volání a virtuální paměť

- Jedním z prvních počítačů co používál modernější techniky byl britský **Atlas**.
- Objevily se zde poprvé systémové volání a **virtuální paměť**.

Atlas[8]

[8] https://en.wikipedia.org/wiki/Atlas_(computer)#/media/File:University_of_Manchester_Atlas,_January_1963.JPG

Historie OS: multiprogramming

- Velkou změnou v tom, jak počítače fungují bylo multiprogramming (nebo multitasking).
- Už není spuštěn jen jeden program a ten běží do konce, ale může být spuštěno **více programů paralelně**.
- Jeden z prvních OS co toto umožňoval byl **Unix**.

PDP-7[9]

[9] https://en.wikipedia.org/wiki/PDP-7#/media/File:Pdp7-oslo-2005.jpeg

OS v současné době

- V 90. letech v reakci na uzavření zdrojového kódu Unixu vznikl **Linux**.
- Znáte taky **Windows**, ale hodně používaných OS jsou deriváty Unixu:
  - **Darwin** (Applovský kernel pro jejich OS)
  - **OpenBSD**, **FreeBSD**, **NetBSD** (open-source varianty BSD Unixu)
    - OpenBSD si zakládá na bezpečnosti.
  - Dále komerční Unix-like systémy, např. **Solaris** od Oraclu[10]
    - Vzniklo v něm mnoho zajímavých fíčur jako např. ZFS.[11]

[10] dříve Sun

[11] Které bylo i později přepsáno jako OpenZFS pro Linux.

Další materiály

- Zájemcům o další obecné informace doporučuju kapitoly Introduction a Process API z Operating Systems: Three Easy Pieces.[12]
  - V kapitole o procesech se hodí znalost C.
- Dále je zajímavá kapitola Operating system interfaces z xv6 knížky.[13]

[12] https://pages.cs.wisc.edu/~remzi/OSTEP/
[13] https://pdos.csail.mit.edu/6.1810/2025/xv6/book-riscv-rev5.pdf

Cvičení/úkol: vlastní sada systémových volání

- vytváření procesů: `exec(path): pid`
- ukončení procesu: `kill(pid): success`
- otevírání souboru: `open(pid, path): file_descriptor`
- zavírání souboru: `close(pid, file_desc): success`
- získání informací o procesu: `stats(pid): proc_status`
- Kód se zadáním je na https://radojcic.cz/osy/handouts/

Cvičení: zamyslete se nad efektivitou!

- Jak rychlý bude váš kód, když bude spuštěno hodně procesů?
  - Do jaké datové struktury dává smysl ukládat procesy?
  - Hint: https://www.youtube.com/watch?v=KyUTuwz_b7Q
- Rychlost OS je důležitá, jelikož ovlivňuje rychlost celého systému.