Jádro operačního systému

Petr Holášek
holasekp@gmail.com

Univerzita Mikuláše Koperníka

27. října 2013

umk

Popis předmětu IJOS

Přednášky:

- Jedna přednáška 26.10.2013; 15:00

Zkouška:

- První a poslední termín zkoušky 28.10.2013
- Full-text + sada otázek na ABCD

Hodnocení:

- 0-49 b. = F
- 50-59 b. = E, 60-69 b. = D, 70-79 b. = C, 80-89 b. = B, 90-100 b. = A

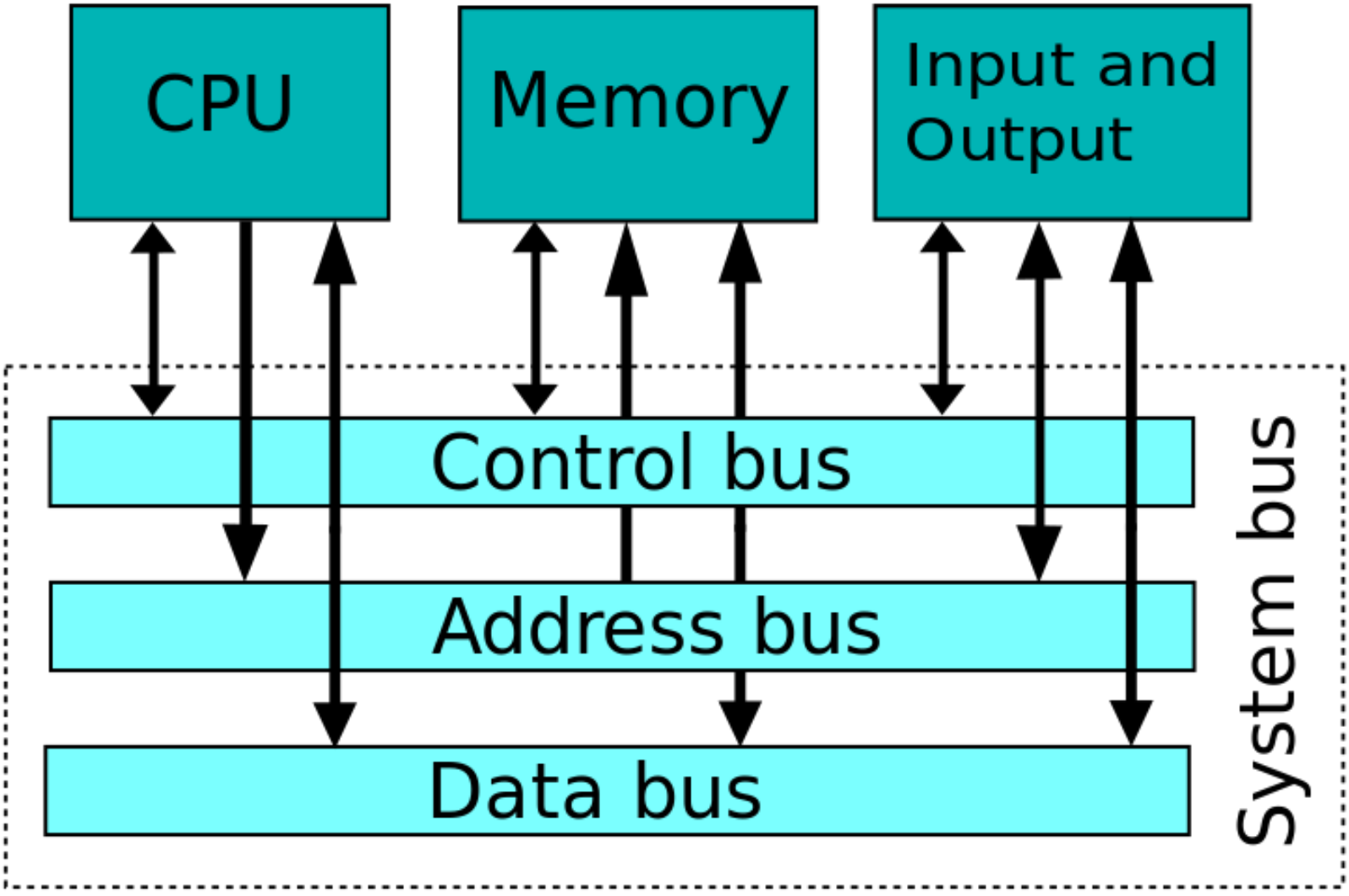
Osnova

- HW a SW
- Správa procesů
- Správa paměti
- Typy jader
- Jak se vyvíjí linuxové jádro (kernel)?

Úvod

- Z čeho se skládá počítač?
- Hardware
  - Procesor(y)
  - Paměť
  - Zařízení - disky, grafické karty, síťové adaptéry, ...
  - Sběrnice
- Software
  - Operační systém (obsahuje jádro)
  - Uživatelské programy

Von Neuman


CPU
Memory
Input and Output
Control bus
Address bus
Data bus
System bus

Co znamená "počítačový program"?

- Pro (většinu) lidí jde o zdrojový kód

```
...
pokus\_file = open\_file("pokus.txt");
content = read\_file(pokus\_file);
close\_file(pokus\_file);
```

- Pro počítač jde o strojový kód

```
...
0010 1101 0100 0010
0110 1000 1111 0001
```

- Mezistupněm je jazyk symbolických adres (asembler)

```
...
mov eax, 0
cmp eax, 4
jne eax\_neni\_rovno
```

Běh software na počítači

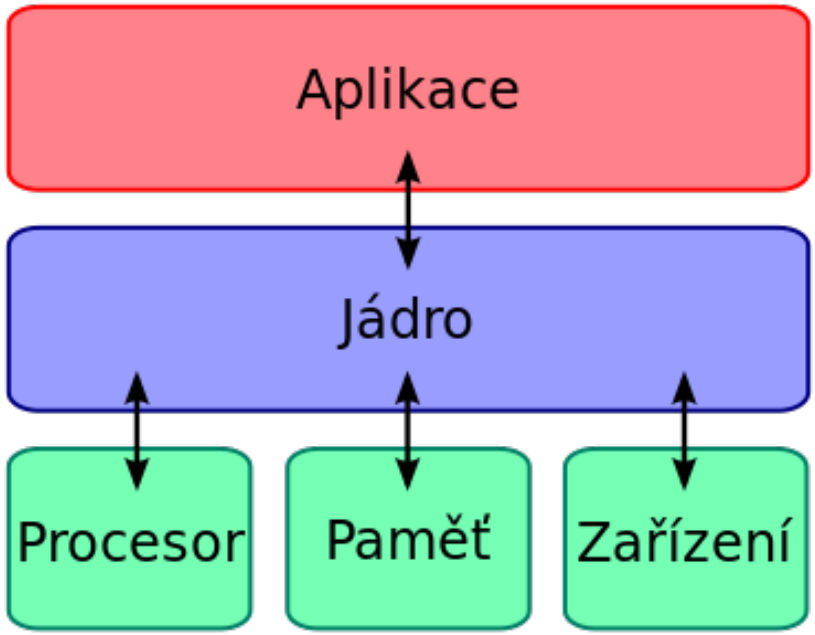
Von Neumannova architektura

- Procesor(y) zpracovává(-ají) instrukce
- Na jejich základě přistupuje procesor do paměti a I/O zařízení
- Všechny části jsou spojeny sběrnicemi

Co je jádro OS?

- Vrstva SW mezi HW a uživatelskými programy
- Vytváří prostředí pro běh uživatelských programů
- Vytváří abstrakci neustále dostupného CPU
- Vytváří abstrakci dostupné lineární paměti (viz. virtuální paměť dále)
- Zpřístupňuje I/O zařízení uživatelských programům

Co je jádro OS?



Obrázek : Kernel

Rozhraní jádra

- systémové volání (read(), iotctl(), ...)
- socket (netlink, udp)
- procfs, sysfs, debugfs, ...
- ...

userspace x kernelspace

- kernelspace - vše, co se děje v prostoru jádra operačního systému
- userspace - vše, co se NEděje v prostoru jádra operačního systému

Režimy činnosti procesoru

- HW ochrana systému
- Dva režimy - uživatelský (user mode) a privilegovaný (kernel mode)

- Uživatelský - vykonává kód aplikace
- Privilegovaný - vykonává přístup k HW, do paměti

```
buffer = read(openedfile);
  ^---- volani C knihovny
     ^---- volani systemoveho volani read()
----------> prechod do privilegovaneho rezime
        ^---- cteni z disku(cache)
```

Reprezentace procesu

- Proces program (!)
- Proces má přiřazené process ID (PID)
- Může být tvořen více vlákny (rozdíl proces vs. vlákno)
- Jádro si musí stále pamatovat jeho
  - stav
  - otevřené soubory/zařízení
  - úsek paměti, který používá
  - otce/potomky (fork())
  - statistiky o běhu
  - a mnoho dalšího..

Správa a plánování procesů

- Toto přidělování řídí *plánovač*
- Plánování je preemptivní nebo nepreemptivní
- Procesům jsou jádrem přidělovány časová kvanta procesoru (u preemptivního)

preemptivní CPU může být kdykoliv procesu odebrán

nepreemptivní CPU může být procesu odebrán, až mu to proces dovolí

Hierarchie procesů

- Lze zobrazit programem `pstree`
- Počáteční proces je proces s PID 0
- Potomci jsou dále tvoření pomocí volání funkce `fork()` uvnitř rodičovského procesu
- Pokud je dřív ukončen potomek, otcovský proces si přebírá jeho návratovou hodnotu (do té doby je zombie)
- Pokud je dřív ukončen rodič, potomek se stává potomkem procesu s PID 0

Stavy procesu

- běžící - running: běží na procesoru nebo je připraven k běhu ve frontě
- čekající - uninterruptible sleep: čeká typicky na dokončení I/O operace
- uspaný - interruptible sleep: čeká na dokončení události nebo uplynutí časovače
- zombie - ukončený proces, kterého si dosud jeho otec nepřevzal

Lze zobrazit například pomocí programu `htop`

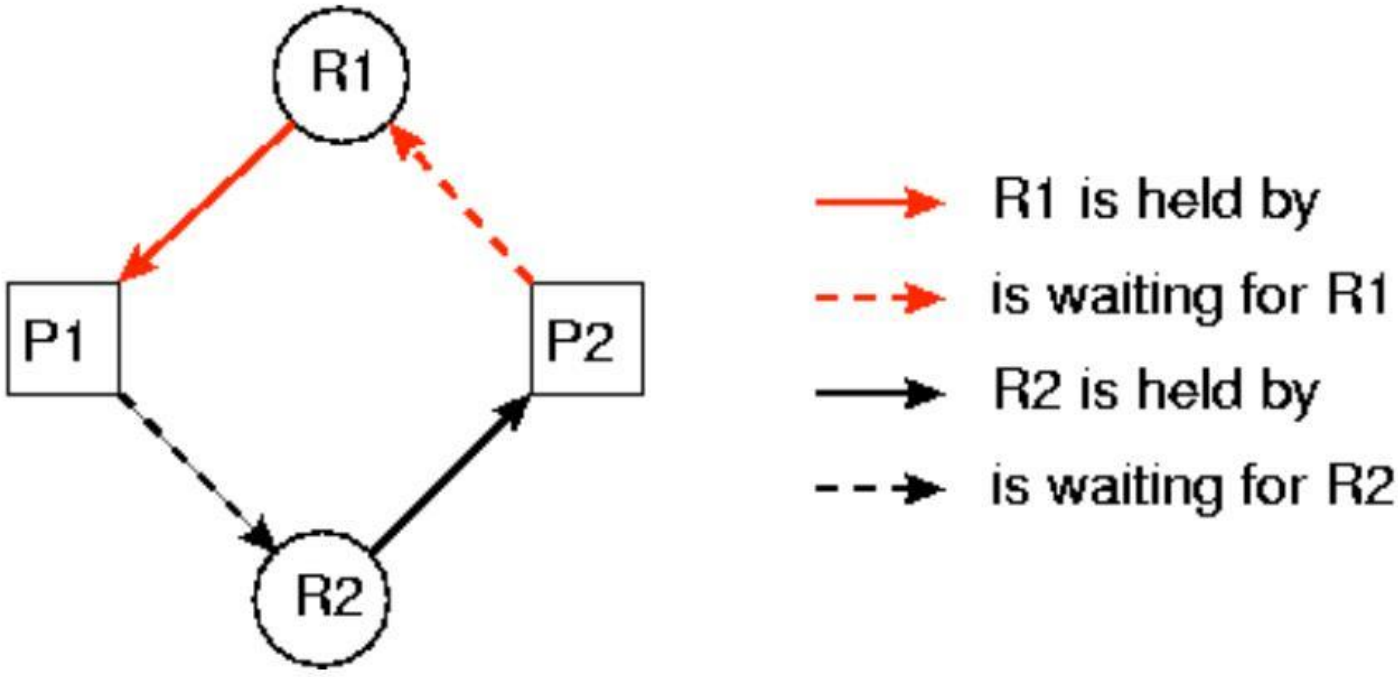
Synchronizace mezi procesy

- Na jednom systému může běžet *mnoho* procesů současně
- Procesy mají k dispozici omezený počet "zdrojů"(paměť, úložný prostor)
- Dochází k "souboji"procesů o zdroje
- Synchronizace pomocí zámků, mutexů nebo semaforů
- Existují následující scénáře, ke kterým by nemělo docházet:

deadlock - Procesy si navzájem blokují závislé zdroje =¿ nikdo nemůže pokračovat

vyhladovění - Proces je stále předbíhán procesy s vyšší prioritou

Deadlock



Obrázek : Deadlock

Vyhladovění

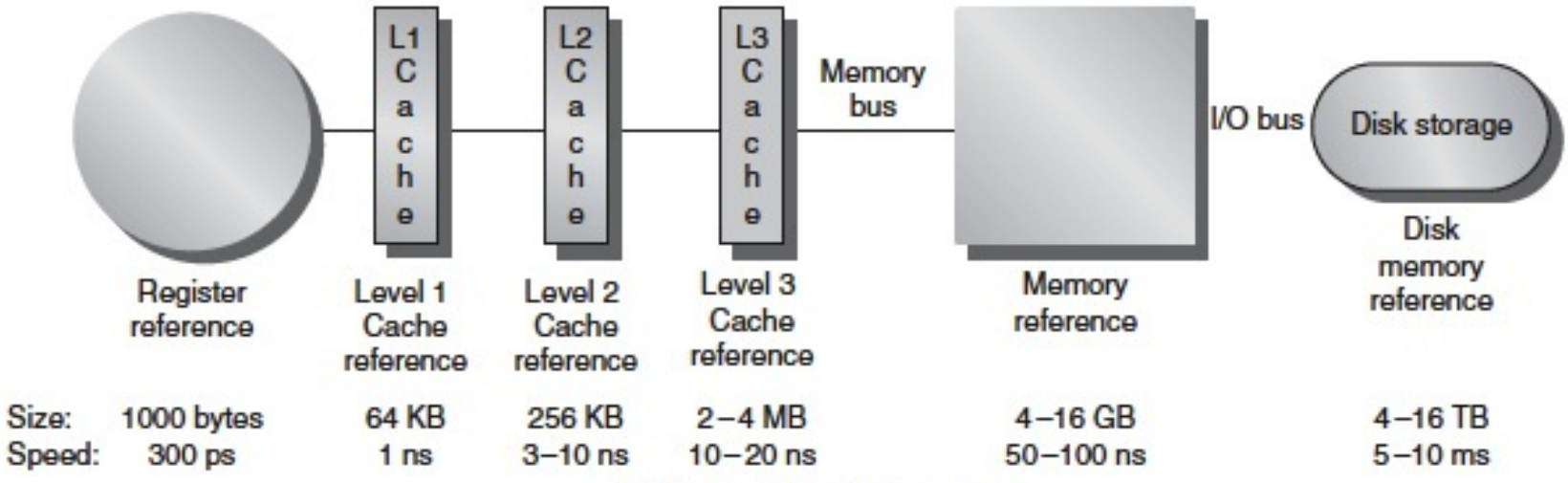
Správa paměti

- Procesy potřebují ke svému běhu vždy paměť
- Jádro operačního systému ji musí jednoduše poskytnout
- Problémy: mnoho procesů, omezená velikost paměti, nutnost izolace paměti mezi procesy
- Jedno z řešení = virtuální paměť

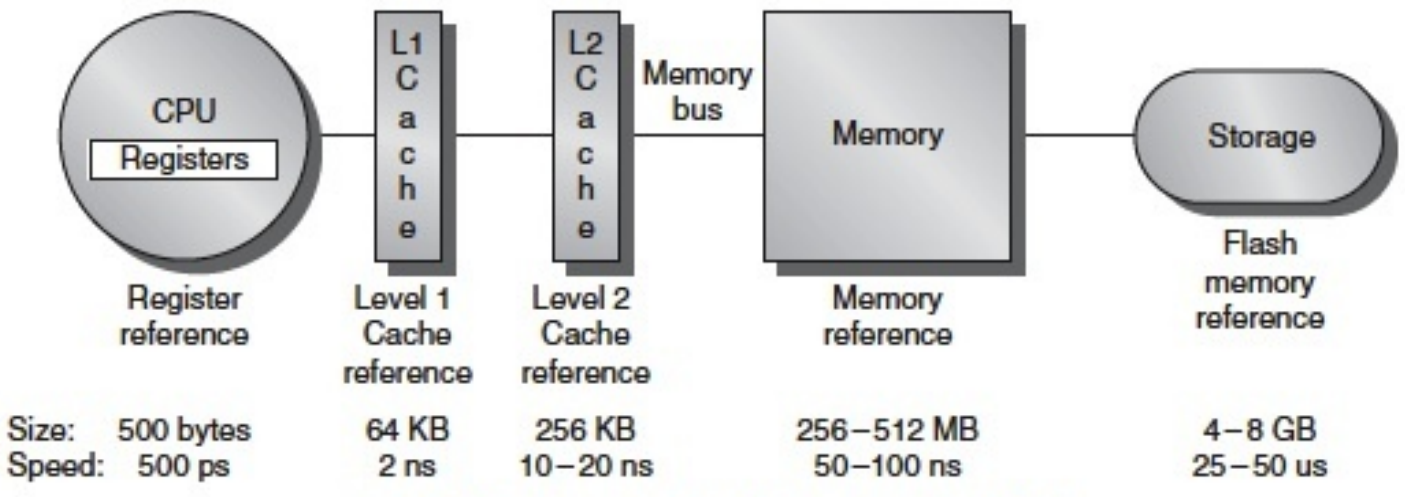
Paměťová hierarchie

- Uspořádání paměťových úložišť:
  - Vzestupně podle kapacity
  - Sestupně podle rychlosti
- Data se pohybují oběma směry
- Stupně:
  - Registry procesoru
  - Cache (L1, L2, L3)
  - Hlavní paměť (RAM - Random Access Memory)
  - Sekundární úložiště (disk, páska)

Paměťová hierarchie



(a) Memory hierarchy for server



(b) Memory hierarchy for a personal mobile device

Stránkování

- Na všech moderních OS je nejmenší použitelné množství paměti 1 stránka
- Typicky 4kB, existují však i větší (viz. hugepages v Linuxu)
- Z pohledu OS rozlišujeme dva typy stránek:
  - Virtuální stránky - viditelné pro běžící procesy a tedy programátory
  - Fyzické rámce - paměť v RAM

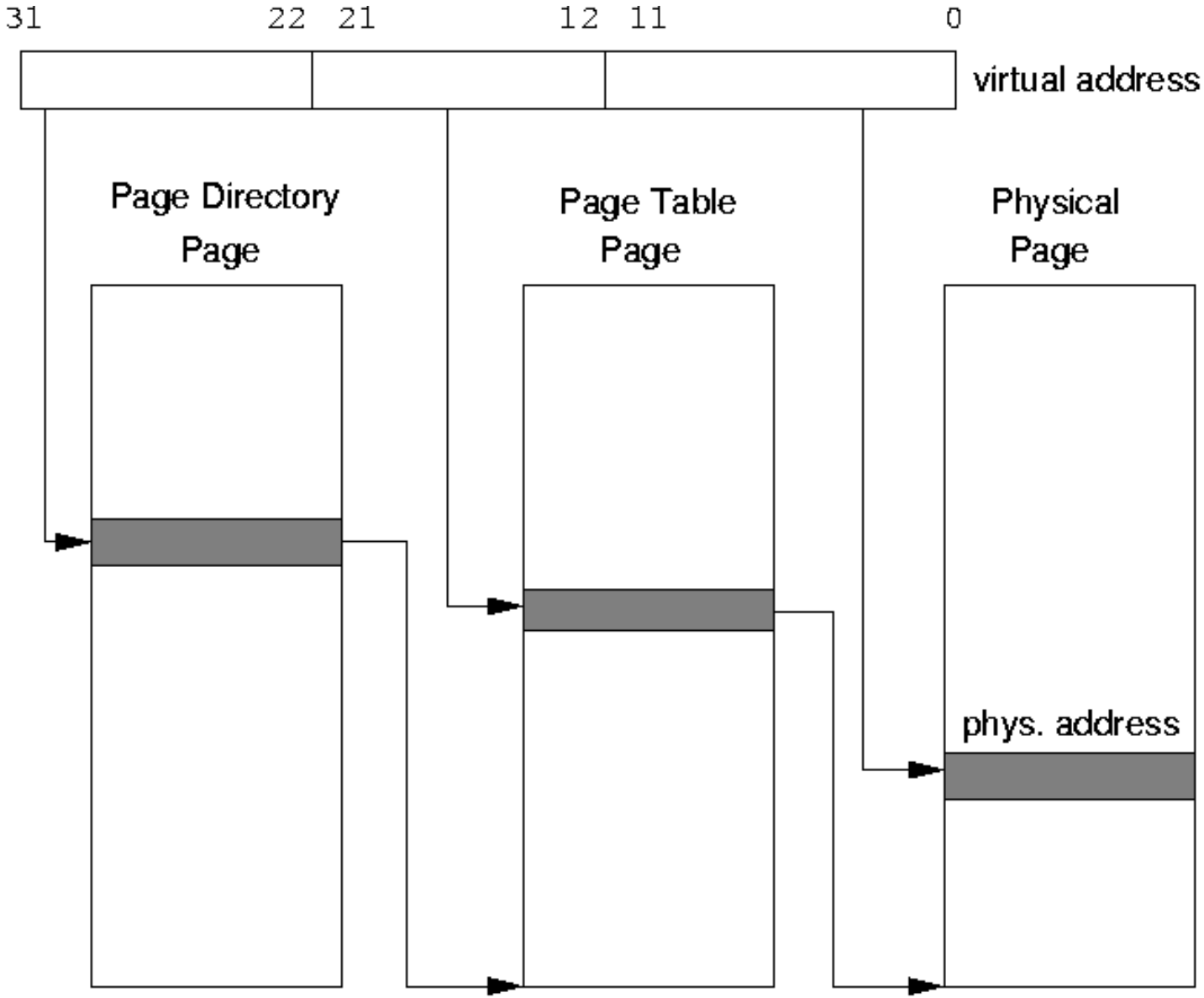
Virtuální paměť

- Koncept řešící současný běh více procesů na jednom systému
- Každý běžící proces má k dispozici 4GB adresového prostoru rozděleného na jednotlivé *stránky*
- Virtuální paměť řeší přiřazení fyzických rámců ke stránkám virtuálních (ne nutně 1:1 - u sdílených knihoven)
- Adresa `0x89f92ad2` u procesu 1 ukazuje do jiné buňky paměti než stejná adresa u procesu 2

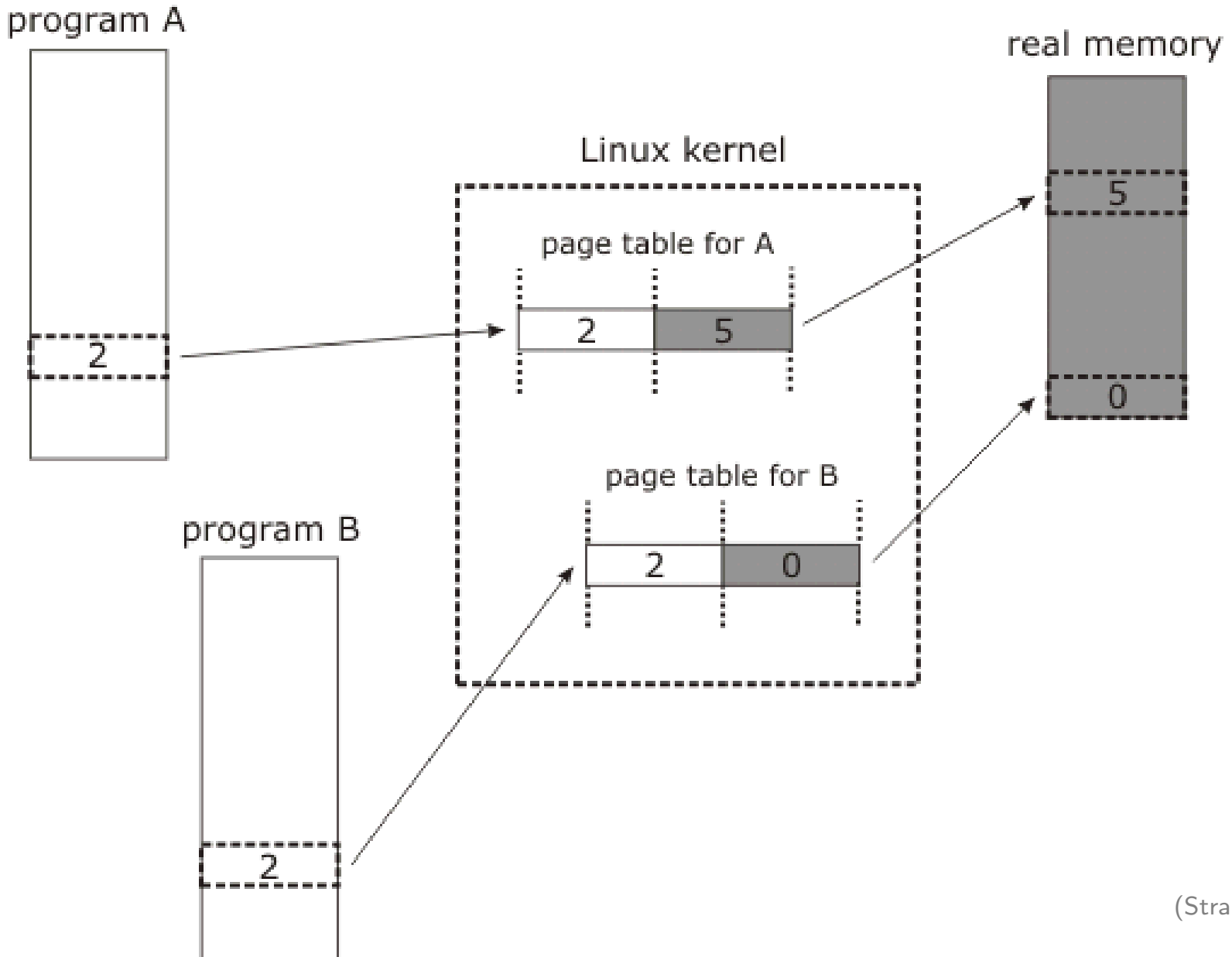
Tabulka stránek

- Udržuje mapování virtuálních stránek na fyzické rámce pro **každý** běžící proces
- Adresa je rozdělena na několik částí podle počtu stupňů TS a offset
- Hledání v tabulce je časově náročné, existuje tedy malá cache TLB (Translation lookaside buffer), která obsahuje poslední úspěšné překlady mapování
- Při přepnutí procesu je TLB vyprázdňována

Tabulka stránek



Překlad adres



Swapování stránek

- Řeší záhadu proč může každý proces používat 4GB paměti i když je fyzicky nainstalováno méně RAM
- Pokud je hlavní paměť RAM plná, jádro OS vybere nejméně potřebnou stránku a uloží ji na swap oddíl, typicky na sekundárním úložišti:
  - v Linuxu obvykle samostatný diskový oddíl typu swap
  - ve Windows stránkovací soubor viditelný přímo na disku
- Poté toto uvolněné místo ve fyzické paměti nahradí novou požadovanou stránkou
- Při přístupu na stránku umístěnou ve swapu je stránka opět nahrána do hlavní paměti, kde opět vystřídá jinou, "nejméně" potřebnou, pokud není stále volné místo.
- V designu paměťového podsystému je vždy velká snaha pomoci použití sofistikovaných algoritmů omezit použití swapu na minimum

Načítání jádra při startu počítače

1. Zapnutí počítače
2. Start BIOSu - kontrola HW, inicializace
3. Načtení bootloaderu z MBR - 512 byte dlouhý úsek na začátku disku
4. Běh bootloaderu, který vybere jádro a načte jej do paměti

Typy jader

mikrojádro - implementuje pouze základní služby (paměť, procesy), zbytek implementován v uživatelském prostoru

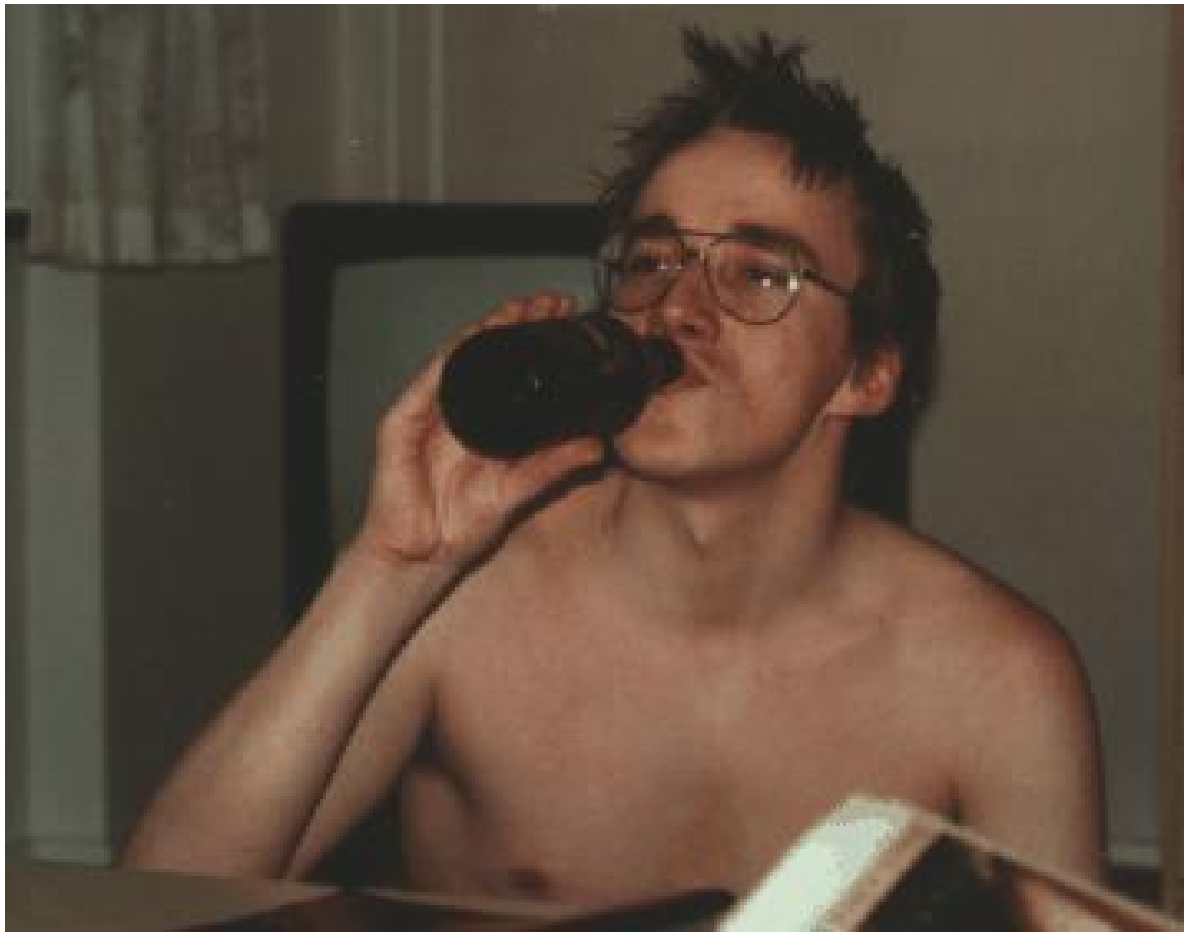
monolitické - vše implementováno uvnitř jádra - Linux

hybridní - kompromis mezi dvěma přístup - Windows

Vývoj linuxového jádra

- Finský programátor Linus Torvalds začal s vývojem v roce 1991
- Dosud hlavním šéfem a diktátorem vývoje
- První verze jádra měla 10 000 řádků
- Linuxový kernel má v současné době přes 15 000 000 řádků

Linus Torvalds



Obrázek : Linus Torvalds

Komunikace mezi vývojáři

- Změny se zasílají formou patchů - rozdílů v kódu
- Téměř výlučně je používán email + systém pro zprávu verzí git
- Linux Kernel Mailing List: https://lkml.org/
- Tisíce mailů denně od vývojářů z celého světa, ne úplně přátelské prostředí
- Až na výjimky velice vysoká úroveň kódu, který je přijat

Patch

- Změna řádků v kódu
- Hlavní způsob komunikace v OSS komunitě
- ukázka

Literatura

- http://www.kernel.org/