



Konfigurace jádra

- **System V** konfigurace jádra (/etc/system, /etc/conf/).
- **BSD** konfigurace jádra (/sbin/config, konfigurační soubory, adresáře pro kompilaci). 
- **Linux** – jako jiné programy (používá make). .config nebo /proc/config.gz. 

Monolitické jádro

- Jeden soubor na disku.
- Všechny používané ovladače jsou uvnitř jádra.
- Často bez autodetekce zařízení.
- Paměť dostupná všem částem jádra stejně.
- Jedna část jádra (ovladač) může rozbít druhou.

Mikrojaderné systémy

- CMU Mach, OSF Mach, L4, minix, Windows NT HAL, QNX, VxWorks, ...
- Co nemusí být v jádře, dát mimo něj.
- Procesy (**servery**) pro správu virtuální paměti, ovládání zařízení, disků a podobně.
- Dobře definovatelné podmínky činnosti (jen teoreticky, protože: DMA, SMI, IOMMU a další problémy).
- Předávání zpráv – malá propustnost, velká latence.

Exkurze do historie

Linux is obsolete (1992)

[http://oreilly.com/catalog/opensources/
/book/appa.html](http://oreilly.com/catalog/opensources/book/appa.html)

Mikrojaderné systémy

- CMU Mach, OSF Mach, L4, minix, Windows NT HAL, QNX, VxWorks, ...
- Co nemusí být v jádře, dát mimo něj.
- Procesy (**servery**) pro správu virtuální paměti, ovládání zařízení, disků a podobně.
- Dobře definovatelné podmínky činnosti (jen teoreticky, protože: DMA, SMI, IOMMU a další problémy).
- Předávání zpráv – malá propustnost, velká latence.



Exkurze do historie

Linux is obsolete (1992)

[http://oreilly.com/catalog/opensources/
/book/appa.html](http://oreilly.com/catalog/opensources/book/appa.html)

Modulární jádro

- Části (moduly), přidávané do jádra za běhu (odpovídá dynamicky linkovaným knihovnám v uživatelském prostoru).
- Ovladače, souborové systémy, protokoly, ...
- Přidávání ovladačů pouze při startu systému – AIX, Solaris < 10.
- Definovaná rozhraní, nikoliv adresní prostor.

Modulární jádro v Linuxu



- Dynamické přidávání ovladačů podle potřeby.
- Závislosti mezi moduly (`depmod(8)`).
- Dynamická registrace ovladačů:
`register_chrdev()`, `register_blkdev()`,
`register_netdev()`, `register_fs()`,
`register_binfmt()` a podobně.
- Dohledávání pomocí identifikátorů sběrnice (např. PCI ID).


Procesy v jádře

- Při startu – kontext procesu číslo 0 – později idle task.
- Idle task **nemůže být zablokován** uvnitř čekací rutiny.



Definice: Kontext

Stav systému, příslušný běhu jednoho procesu/vlákná.

- Přepnutí kontextu – výměna právě běžícího procesu za jiný.
- Linux – struct task_struct, current .

Procesy uvnitř jádra

? Otázka:

Pod jakým kontextem mají běžet služby jádra?

- UNIX – použije se kontext volajícího procesu.
- Dva režimy činnosti procesu – user-space a kernel-space.
- Mikrokernl – předá se řízení jinému procesu (serveru).
- Nutno vyřešit přístup do user-space (např. pro `write(2)`).

Procesy uvnitř jádra

? Otázka:

Pod jakým kontextem mají běžet služby jádra?

- **UNIX** – použije se kontext volajícího procesu.
- **Dva režimy činnosti procesu** – user-space a kernel-space.
- **Mikrokernel** – předá se řízení jinému procesu (serveru).
- Nutno vyřešit přístup do user-space (např. pro `write(2)`).

Přerušení

- Žádost o pozornost hardwaru
- Obsluha – nepřerušitelná nebo priority.
- Horní polovina – co nejkratší, nepřerušitelná. Např. přijetí packetu ze sítě, nastavení vyslání dalšího packetu. Interrupt time.
- Spodní polovina – náročnější úkoly, přerušitelné. Obvykle se spouští před/místo předání řízení do uživatelského prostoru. Například: směrování, výběr dalšího packetu k odvysílání. Softirq time.
- Preemptivní/nepreemptivní jádro – může dojít k přepnutí kontextu kdekoli v jádře?

Zpracování přerušení v jádře



Otázka:

Pod jakým kontextem lze provádět přerušení?


- Zvláštní kontext – nutnost přepnutí kontextu → zvýšení doby odezvy (latence) přerušení. Navíc je nutno případně mít více kontextů pro možná paralelně běžící přerušení.
- UNIX (ve většině implementací): Přerušení se provádí pod kontextem právě běžícího procesu. Obsluha přerušení nesmí zablokovat proces.
- Linux bez samostatného kontextu, uvažuje se o threaded handlers .

Zpracování přerušení v jádře



Otázka:

Pod jakým kontextem lze provádět přerušení?

- **Zvláštní kontext** – nutnost přepnutí kontextu → zvýšení doby odezvy (latence) přerušení. Navíc je nutno případně mít více kontextů pro možná paralelně běžící přerušení.
- **UNIX** (ve většině implementací): Přerušení se provádí pod kontextem právě běžícího procesu. Obsluha přerušení nesmí zablokovat proces.
- **Linux** bez samostatného kontextu, uvažuje se o threaded handlers .

Odložené vykonání kódu

- Funkce, vykonaná později (po návratu z přerušení, při volání scheduleru, atd.)
- Spodní polovina obsluhy přerušení.
- Časově nekritický kód
- Může být přerušen
- Linux – bottom half, tasklety, workqueues, ...

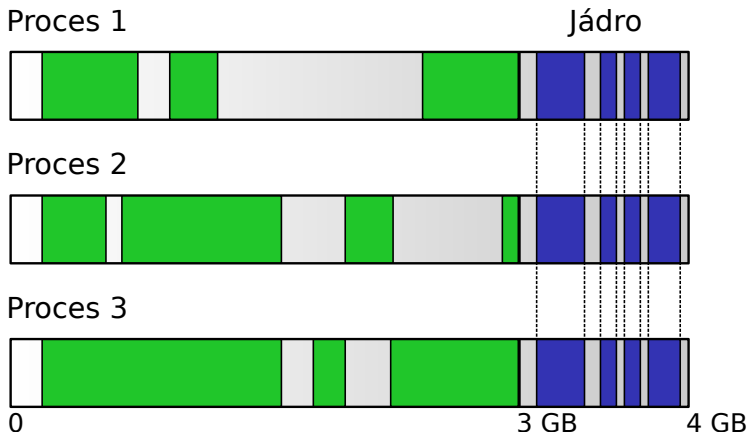
Virtuální paměť

- **Virtuální adresa** – adresa z hlediska instrukcí CPU.
- **Překlad mezi virtuální a fyzickou adresou** – stránková tabulka.
- Každý proces má svoji virtuální paměť: každý proces má svoji stránkovou tabulku.
- **Výpadek stránky** (page fault) – stránka není v paměti, stránkový adresář neexistuje, stránka je jen pro čtení a podobně.
- **Obsluha výpadku stránky** – musí zjistit, jestli jde (například) o copy-on-write, o žádost o natažení stránky z odkládacího prostoru, o naalokování stránky, nebo jestli jde o skutečné porušení ochrany paměti procesem.

Translation Look-aside Buffer

- TLB – asociativní paměť několika posledních použitých párů (*virtuální adresa, fyzická adresa*).
- Přepnutí kontextu – vyžaduje vyprázdnění TLB, v případě virtuálně adresované cache také vyprázdnění cache.
- Přepnutí mezi vlákny je rychlejší.
- Softwarový TLB – OS-specifický formát stránkových tabulek.
- Lazy TLB switch – uvnitř jádra lze ušetřit.

Prostor jádra a uživatelský prostor

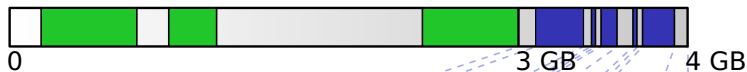


Prostor jádra a uživatelský prostor

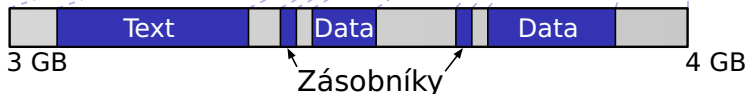
- **Virtuální paměť jádra** – obvykle mapována na nejvyšších adresách.
- **Paměť jádra** – mapována do **všech procesů** stejně.
- **Přepnutí do režimu jádra** – zpřístupnění horních (virtuálních) adres.
- **Alternativa** – jádro má samostatnou VM (ale: TLB flush při volání jádra nebo přerušení); 4:4 GB split.

Virtuální paměť uvnitř jádra

Proces



Prostor jádra



- **Zásobník v jádře** – pro každý thread/kontext.
- **Linux** – 1 stránka/thread, nastavitelné 2 stránky/thread .

Jádro a fyzická paměť

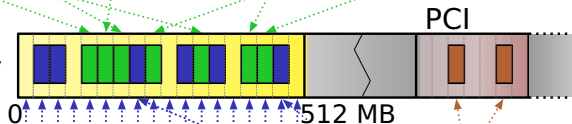
- Fyzická paměť – mapována také 1:1 do paměťové oblasti jádra (Linux bez CONFIG_HIGHMEM).
- Použití víc než 4 GB paměti na 32-bitových systémech – Intel PAE, 36-bitová fyzická adresa.
- Virtuální alokace – dočasné zpřístupnění fyzické paměti uvnitř jádra.

Jádro a fyzická paměť

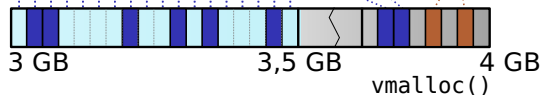
Procesy



Fyzické adresy



Jádro



Fyzická paměť 32-bitového Linuxu



Úkol:

Kolik fyzické paměti může obsloužit 32-bitový Linux bez CONFIG_HIGHMEM, má-li 128 MB vyhrazeno pro virtuální alokace?

Paměť z hlediska hardwaru

- **Fyzická adresa** – adresa na paměťové sběrnici, vycházející z CPU (0 je to, co CPU dostane, vystaví-li nuly na všechny bity adresové sběrnice).
- **Virtuální adresa** – interní v CPU. Instrukce adresují paměť touto adresou.
- **Sběrniceová adresa** – adresa místa v paměti tak, jak je vidí ostatní zařízení.
- **IOMMU** – překlad adres mezi sběrnici a operační pamětí. Příklad: AGP GART, AMD Opteron IOMMU.

Úkol:

K čemu může sloužit IOMMU? Proč mít odlišné fyzické a sběrniceové adresy?

Paměť z hlediska hardwaru

- **Fyzická adresa** – adresa na paměťové sběrnici, vycházející z CPU (0 je to, co CPU dostane, vystaví-li nuly na všechny bity adresové sběrnice).
- **Virtuální adresa** – interní v CPU. Instrukce adresují paměť touto adresou.
- **Sběrniceová adresa** – adresa místa v paměti tak, jak je vidí ostatní zařízení.
- **IOMMU** – překlad adres mezi sběrnici a operační pamětí. Příklad: AGP GART, AMD Opteron IOMMU.



Úkol:

K čemu může sloužit IOMMU? Proč mít odlišné fyzické a sběrniceové adresy?

Přístup do uživatelského prostoru

- **Přístup do user-space:** proces předá jádru ukazatel (např. buffer pro `read(2)`).
- **Robustnost** – user-space nesmí způsobit pád jádra.
- **Validace před použitím?** Problémy ve vícevláknových programech (přístup versus změna mapování v jiném vlákně).

Úkol:

Přístup do uživatelského prostoru není možný uvnitř ovladače přerušení (proč?).

Přístup do uživatelského prostoru

- **Přístup do user-space:** proces předá jádru ukazatel (např. buffer pro `read(2)`).
- **Robustnost** – user-space nesmí způsobit pád jádra.
- **Validace před použitím?** Problémy ve vícevláknových programech (přístup versus změna mapování v jiném vlákně).

? Úkol:

Přístup do uživatelského prostoru není možný uvnitř ovladače přerušení (proč?).

Přístup do user-space v Linuxu



```
status = get_user(result, pointer);  
status = put_user(result, pointer);  
get_user_ret(result, pointer, retval);  
put_user_ret(result, pointer, retval);  
copy_user(to, from, size);  
copy_to_user(to, from, size);  
copy_from_user(to, from, size);  
...
```

Implementace v Linuxu



- **Využití hardwaru CPU** – kontrola přístupu do paměti. Přidání kontroly do `do_page_fault()`.
- **Tabulka výjimek** – adresa instrukce, která může způsobit chybu, opravný kód.
- **Normální běh** – cca 10 instrukcí bez skoku.
- **ELF sekce** – pro generování druhého toku instrukcí.
- Viz též `linux/arch/x86/include/asm/uaccess.h`, např. `__put_user_asm_u64()`.

Použití uživatelského ukazatele



- Porovnání s `PAGE_OFFSET` (3 GB na 32-bitovém systému). Je-li větší, chyba.
- Použití ukazatele - není-li platný, výjimka CPU.
- Obsluha výjimky - je adresa instrukce v tabulce výjimek? Ano: zavolat opravný kód.
- Jinak: interní chyba jádra (kernel oops).



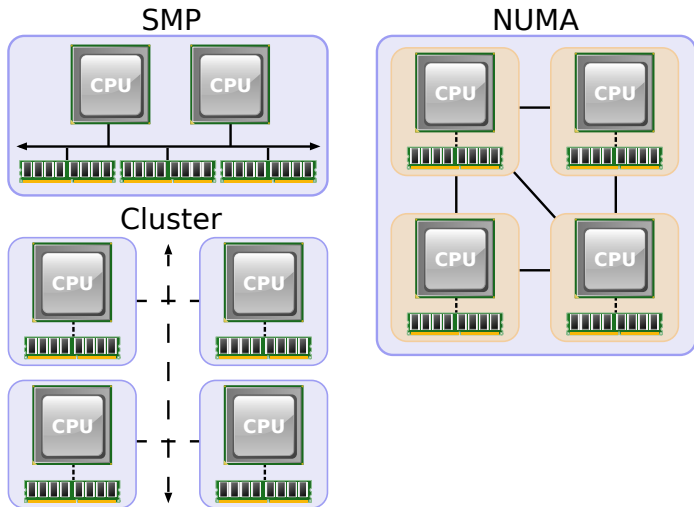
Problém: volání služby jádra zevnitř jádra

- Nutno předem oznámit. Linux: `set_fs(KERNEL_DS)`
- Například: `net/socket.c: kernel_sendmsg()`.

Paralelní stroje

- **SMP** – symetrický multiprocessing. Společný přístup všech CPU k paměti.
- **NUMA** – hierarchická paměť – z určitých CPU rychlejší přístup než z jiných (cc-NUMA – cache coherent).
- **Multipočítače** – na částech systému běží zvláštní kopie jádra (clustery a podobně).
- **Problémy** – cache ping-pong, zamykané přístupy na sběrnici, afinita přerušení.


Paralelní stroje



Zamykání kódu

- **Paralelismus** – v jednom okamžiku mohou tytéž data modifikovat různé procesy (kontexty).
- **Na jednom CPU** – v kterémkoli okamžiku může být proces přerušen a tentýž kód může provádět i jiný proces.
- **Problém** – manipulace s globálními datovými strukturami (alokace paměti, seznam volných i-uzlů, atd.).

Zamykání a jednom CPU

- Postačí ochrana proti přerušení
- Zákaz přerušení na CPU - instrukce cli a sti, v Linuxu funkce cli() a sti() .
- Problém - proměnná doba odezvy systému.

Na paralelním systému

- **Large-grained (hrubozrnný) paralelismus** – jeden zámek kolem celého jádra (Linux: `lock_kernel()`, `unlock_kernel()`). Paralelismus možný pouze v uživatelském prostoru. Jednodušší na implementaci, méně výkonný.
- **Fine-grained paralelismus** – zámky kolem jednotlivých kritických sekcí v jádře. Náročnější na implementaci, možnost vzniku netriviálně detekovatelných chyb. Vyšší výkon (několik IRQ může běžet paralelně, několik procesorů zároveň běžících v kernelu).
- **Zamykání v SMP** – nutnost atomických instrukcí (test-and-set) nebo detekce změny nastavené hodnoty (MIPS). Zamčení sběrnice (prefix lock na i386).

Semaforey

- Exkluzivní přístup ke kritické sekci
- Určeno i pro dlouhodobé čekání
- Lze volat pouze s platným uživatelským kontextem
- Linux – `up()`, `down()`, `down_interruptible()`.

Spinlocky

- Krátkodobé zamykání
- Nezablokuje proces – proces čeká ve smyčce, až se zámek uvolní.
- V Linuxu – `spin_lock_init(lock)`,
`spin_lock_irqsave(lock)`,
`spin_unlock_irqrestore(lock)` a podobně.

R/W zámky

- Paralelní čtení – exkluzivní zápis
- Linux – `struct rwlock`, `struct rwsem`.
- Problémy – priority? upgrade r-zámku na w-zámek (deadlock).

Read-copy-update

- **RCU** – původně Sequent (Dyrix/PTX), později IBM, implementace i v Linuxu.
- **Atomické instrukce** – pomalé (stovky taktů; přístup do hlavní paměti).
- **Obvyklá cesta** (např. čtení) by měla být rychlá.



Princip činnosti RCU

- Vytvoření **kopie** struktury.
- **Publikování** nové verze (změna ukazatele).
- **Uvolnění** původní verze.