12. Operační systémy a jejich architektury. Systémová volání, vlákna, procesy. Správa virtuální a fyzické paměti, souborové systémy. Bezpečnost, virtualizace.

---

https://cw.fel.cvut.cz/old/courses/b4b35osy/start

Vkládám jen výtažky z prezentací, které mi přijdou důležité.
Bližší vysvětlení najdete tam.

Operační systémy a jejich architektury

Úkoly OS:

- Spouštět a dohlížet uživatelské programy
- Efektivní využití HW
- Usnadnit řešení uživatelských problémů
- Učinit počítač (snáze) použitelný

Multitasking - Zdánlivé spuštění více procesů současně je nejčastěji implementováno metodou sdílení času tzv. Time-Sharing Systems (TSS) - rozšiřuje plánovací pravidla o rychlé (spravedlivé, cyklické) přepínání mezi procesy řešícími zakázky interaktivních uživatelů

Architektura - viz APO (Assembly, cyklus CPU, výjimky a přerušení)

Zdroje přerušení:

- Vnitřní přerušení –
  - problém při zpracování strojové instrukce
  - instrukce nebo data nejsou v paměti - chyba stránky, chyba segmentu instrukci nelze provést - dělení nulou, ochrana paměti, nelegální instrukce
  - nutno reagovat okamžitě, nelze dokončit instrukci, někdy nelze ani načíst instrukci

- Vnější přerušení –
  - vstupně/výstupní zařízení asynchronní s během procesoru
  - signalizace potřeby reagovat na vstup/výstup
  - reakce po dokončení vykonávané instrukce
- Programové přerušení – strojová instrukce proveď přerušení
  - využívá se k ochraně jádra OS
  - obsluha přerušení může používat privilegované instrukce
  - lze spustit pouze kód připravený OS

Systémová volání, vlákna, procesy

Program:

- je soubor (např. na disku) přesně definovaného formátu obsahující instrukce, data, údaje potřebné k zavedení do paměti a inicializaci procesu

Proces:

- je spuštěný program – objekt jádra operačního systému provádějící výpočet podle programu
- je charakterizovaný svým paměťovým prostorem a kontextem (prostor v RAM se přiděluje procesům – nikoli programům!)
- může vlastnit (kontext obsahuje položky pro) otevřené soubory, I/O zařízení a komunikační kanály, které vedou k jiným procesům, ...
- obsahuje jedno či více vláken

Vlákno:

- je sekvence instrukcí vykonávaných procesorem sdílí s ostatními vlákny procesu paměťový prostor a další atributy procesu (soubory, ...)
- má vlastní hodnoty registrů CPU

Proces

= spuštěný program

Proces je identifikovatelný jednoznačným číslem v každém okamžiku své existence - PID (Process IDentifier)

Co tvoří proces:

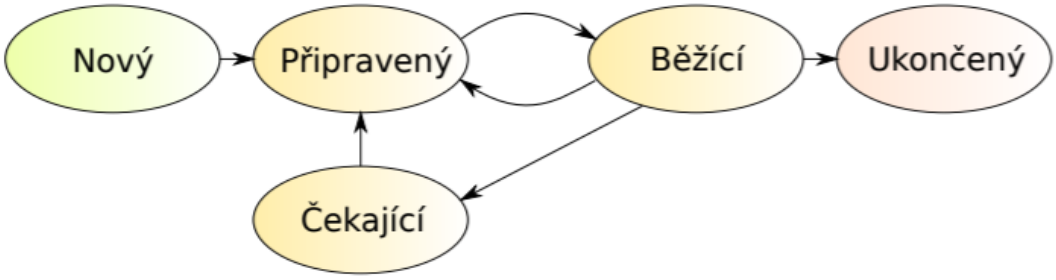
- Obsahy registrů procesoru (čítač instrukcí, ukazatel zásobníku, příznaky FLAGS, uživatelské registry, FPU registry)
- Otevřené soubory
- Použitá paměť: Zásobník – .stack, Data – .data, Program – .text

Rodič vytváří nový proces (potomka) voláním služby **fork** - vznikne identická kopie rodičovského procesu až na:

- návratovou hodnotu systémového volání
- hodnotu PID, PPID – číslo rodičovského procesu

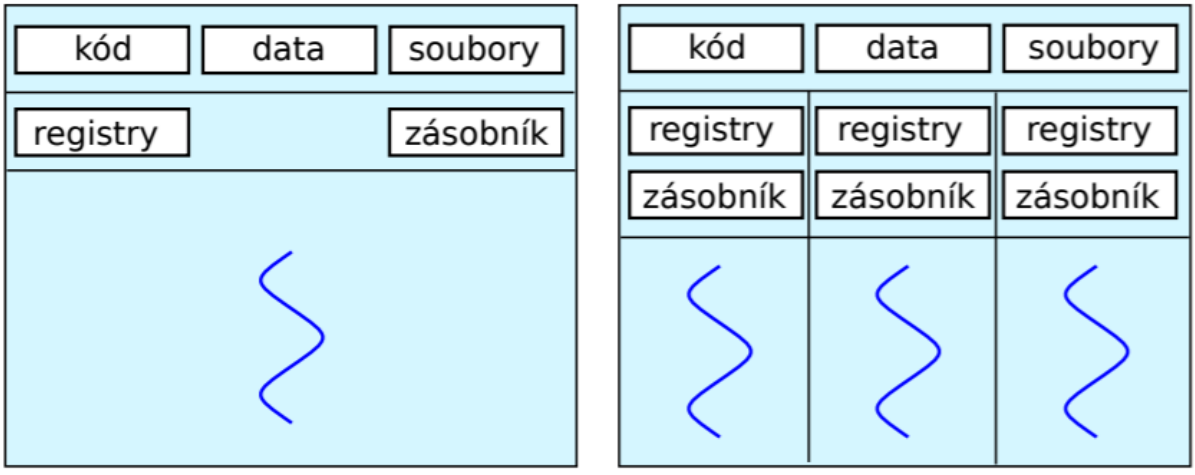
návratová hodnota určuje, kdo je potomek a kdo rodič (0 – jsem potomek, PID – jsem rodič a získávám PID potomka)
potomek může použít volání služby **exec** pro náhradu programu ve svém adresním prostoru jiným programem

Stavy



Přepnutí od jednoho procesu k jinému nastává výhradně v důsledku nějakého přerušení (či výjimky)
Procesy mohou čekat v různých frontách na vykonání (na přidělení procesoru, na dokončení I/O, na přidělení místa v hlavní paměti, na synchronizační událost, na zvětšení adresního prostoru)

Vlákna



Objekt vytvářený v rámci procesu a viditelný uvnitř procesu
Tradiční proces je proces tvořený jediným vláknem
Vlákna podléhají plánování a přiděluje se jim strojový čas i procesory
Vlákno se nachází ve stavech: běží, připravené, čekající, ukončené
Když vlákno neběží, je kontext vlákna uložený v TCB (Thread Control Block) – analogie PCB
Vlákno může přistupovat k globálním proměnným a k ostatním zdrojům svého procesu, data jsou sdílena všemi vlákny stejného procesu:

- Změnu obsahu globálních proměnných procesu vidí všechna ostatní vlákna téhož procesu
- Soubor otevřený jedním vláknem je viditelný pro všechna ostatní vlákna téhož procesu

Co patří komu?

kód programu	proces
lokální proměnné	vlákno
globální proměnné	proces
otevřené soubory	proces
zásobník	vlákno
správa paměti	proces
čítač instrukcí	vlákno
registry CPU	vlákno
plánovací stav	vlákno
uživatelská práva	proces

Přednosti:

- Vlákno se vytvoří i ukončí rychleji než proces
- Přepínání mezi vlákny je rychlejší než mezi procesy
- Dosáhne se lepší strukturalizace programu

Vlákna na uživatelské úrovni:

- OS zná jenom procesy
- Vlákna vytváří uživatelská knihovna, která střídavě mění spuštěná vlákna procesu
- Pokud jedno vlákno na něco čeká, ostatní vlákna nemohou běžet, protože jádro OS označí jako čekající celý proces
- Pouze staré systémy, nebo jednoduché OS, kde nejsou vlákna potřeba

Vlákna na úrovni jádra:

- OS Procesy a vlákna jsou plně podporované v jádře
- Moderní operační systémy (Windows, Linux, OSX, Android)

- Vlákno je jednotka plánování činnosti systému

Realizace: - knihovna PThread, Java - třída Thread

Plánování procesů
preemptivní - CPU může být procesu násilně odebrán
nepreemptivní - nemůže

Typy:
- FCFS (First-Come First-Served)
- SPN (SJF) (Shortest Process Next)
- SRT (Shortest Remaining Time) - preemptivní SPN
- cyklické (Round-Robin) - každý proces dostane CPU na malý úsek a pak je vložen na konec fronty
- zpětnovazební (Feedback) - penalizace za dlouhé využití CPU

Synchronizace
- cílem je zabránit současný přístup více vláken do kritické sekce programu

Nástroje:
- semafor - Semafor je typ zámku založený na čítači. Semafor umožňuje dvě operace:
  - *wait* (acquire, down): Tato operace se volá před vstupem do kritické sekce. Při zavolání operace se vyhodnocuje hodnota čítače: pokud je nenulová, vlákno dekrementuje čítač (typicky o 1) a pokračuje dál ve vykonávání kritické sekce; pokud je nulová, vlákno se zablokuje až do doby, než je čítač opět nenulový.
  - post (release, up): Tato operace se volá ihned po vystoupení z kritické sekce. Při zavolání operace se inkrementuje čítač (typicky o 1). Inkrementování čítače o *n* způsobí probuzení *n* čekajících vláken (viz operace *wait*).

Semafor lze inicializovat na libovolné číslo - toto číslo vyjadřuje, kolik vláken současně může být v kritické sekci. Mutex lze simulovat semaforem s počáteční hodnotou čítače 1.
- mutex - typ zámku, který do kritické sekce vpustí jen jedno vlákno současně. Ostatní vlákna se před vstupem do kritické sekce zablokují, lock() a unlock() metody, lze uzamykat rekurzivně
- monitor - Monitor je typ zámku, který spojuje koncepty vlastníka a množiny čekajících vláken, kde vlákna čekají až do splnění určité podmínky. Vlákna mezi sebou mohou komunikovat a sdělovat si vzájemně, že byla podmínka splněna a mohou čekání přerušit. Vlastník se také může svého vlastnictví vzdát a připojit se

tak k ostatním čekajícím vláknům. Platí, že kritickou sekci může spustit pouze aktuální vlastník monitoru.
- Spin-lock - semafor využívající aktivní čekání vlákna (v některých případech je to rychlejší než režie s procesy)

Deadlock

- nastane v případě, kdy dva procesy čekají na uvolnění zámku tím druhým procesem naráz
- Coffmanovy podmínky - 4 nutné podmínky pro to, aby mohl deadlock nastat
  - Vzájemné vyloučení (Mutual exclusion) - Prostředek může v jednom okamžiku používat jenom jeden proces (aby nedošlo k porušení konzistence dat).
  - Drž a čekej (Hold & wait) - Proces může žádat o další prostředky, i když už má nějaké přiděleny.
  - Neodnímatelnost (No preemption) - Jakmile proces zmíněný prostředek vlastní, nelze mu ho bezpečně odejmout, musí ho sám vrátit.
  - Cyklické čekání (Circular wait) - Je možné uzavřít cyklus z procesů čekající každý na svého předchůdce – respektive k deadlocku dojde, jakmile je tento cyklus uzavřen.

Meziprocesní komunikace

Přehled meziprocesní komunikace

Název	Anglicky	Standard
Signál	Signal	POSIX
Roura	Pipe	POSIX
Pojmenovaná roura	Named pipe	POSIX
Soubor mapovaný do paměti	Memory-mapped file	POSIX
Sdílená paměť	Shared memory	System V IPC
Semafor	Semaphore	System V IPC
Zasílání zpráv	Message passing	System V IPC
Soket	Socket	Networking

Správa virtuální a fyzické paměti, souborové systémy

Podobné jako v APO

FAP

- fyzický adresní prostor
- skutečná paměť počítače – RAM

- velikost závisí na možnostech základní desky a na osazených paměťových modulech

LAP

- logický adresní prostor
- někdy také virtuální paměť
- velikost záleží na architektuře CPU

Výhody systému bez správy paměti:

- rychlost přístupu do paměti
- jednoduchost implementace
- lze používat i bez operačního systému – robustnost

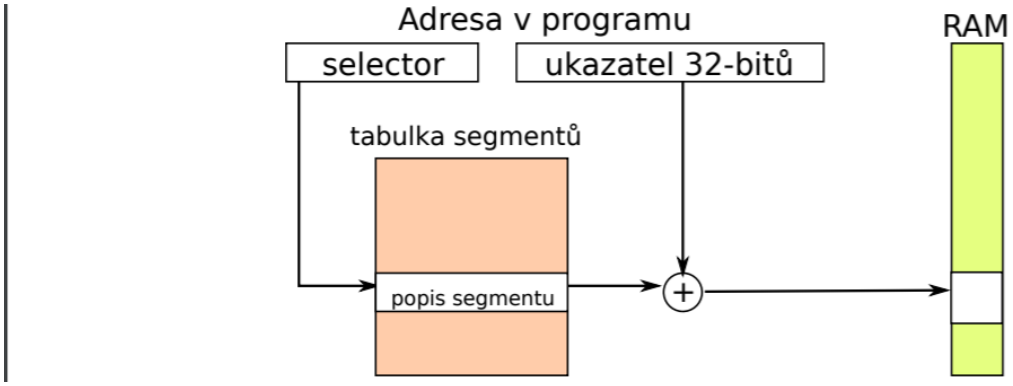
Nevýhody systému bez správy paměti

- Nelze kontrolovat přístup do paměti (kdokoli může cokoli v paměti přepsat)
- Omezení paměti vlastnostmi HW

Segmentace

Program je kolekce segmentů - každý má svůj logický význam (hlavní program, procedura, funkce, objekt a jeho metoda, proměnné, pole, ..)

Základní úkol – převést adresu typu (segment selector, offset) na adresu FAP



Výhody segmentace

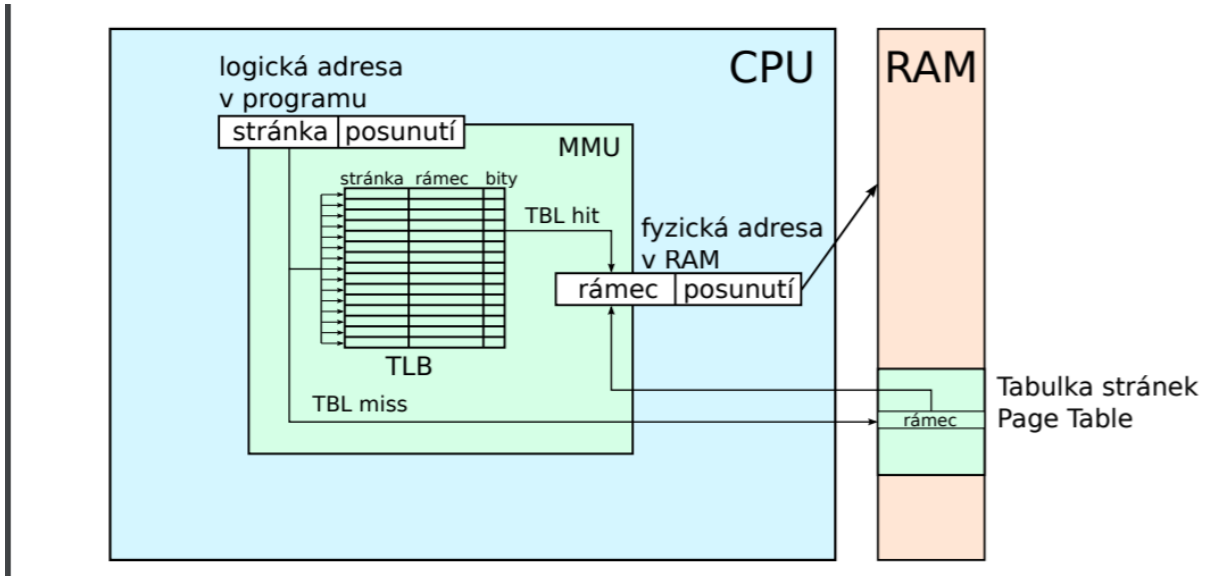
- Segment má délku uzpůsobenou skutečné potřebě
  - minimum vnitřní fragmentace
  - Lze detekovat přístup mimo segment, který způsobí chybu segmentace – výjimku typu „segmentation fault“
- Lze nastavovat práva k přístupu do segmentu
- Lze pohybovat s daty i programem v fyzické paměti - posun počátku segmentu je pro aplikační proces neviditelný a nedetekovatelný

Nevýhody segmentace

- Alokace segmentů v paměti je netriviální úloha
  - Segmenty mají různé délky.
  - Při běhu více procesů se segmenty ruší a vznikají nové. Problém s externí fragmentací
- Režie při přístupu do paměti

Fragmentace
Externí (vnější) fragmentace - Celkové množství volné paměti je sice dostatečné, aby uspokojilo požadavek procesu, avšak prostor není souvislý, takže ho nelze přidělit
Interní (vnitřní) fragmentace - Přidělená díra v paměti je o málo větší než potřebná, avšak zbytek je tak malý, že ho nelze využít

Stránkování
FAP se dělí na úseky zvané rámce - Pevná délka, zpravidla v celistvých mocninách 2
LAP se dělí na úseky zvané stránky - Pevná délka, shodná s délkou rámců
Proces o délce n stránek se umístí do n rámců, rámce ale nemusí v paměti bezprostředně sousedit
Mechanismus překladu logická adresa → fyzická adresa - pomocí tabulky stránek (PT = Page Table)
Může vznikat vnitřní fragmentace - stránky nemusí být zcela zaplněny



Může být i více úrovní tabulek

Virtualizace paměti

Kdy stránku zavádět do FAP? (Fetch policy)

- Stránkování při spuštění
  - Program je celý vložen do paměti při spuštění
  - velmi nákladné a zbytečné, předem nejsou známy nároky na paměť, dříve se nevyužívalo, dnes je využívána
- Stránkování či segmentace na žádost (Demand Paging/Segmentation)
  - Tzv. „líná metoda“, nedělá nic dopředu
  - Řeší problémy s dynamickou alokací proměnných
- Předstránkování (Prepaging)
  - Nahrává stránku, která bude pravděpodobně brzy použita
- Čištění (Pre-cleaning)
  - změněné rámce jsou ukládány na disk v době, kdy systém není vytížen
- Kopírovat při zápisu (copy-on-write)
  - Při tvorbě nového procesu není nutné kopírovat žádné stránky, ani kódové ani datové. U datových stránek se zruší povolení pro zápis.
  - Při modifikaci datové stránky nastane chyba, která vytvoří kopii stránky a umožní modifikace

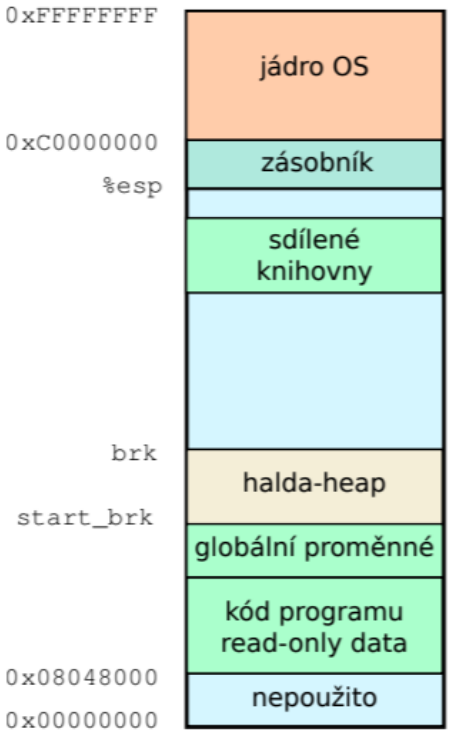
Stránkování – **politika nahrazování** - Musí se vyhledat vhodná stránka pro náhradu, když je FAP plná:

- LRU (Least recently used) - zahodí se nejdéle nepoužitá stránka
- Druhá šance - zahodí se až stránka, ke které se přistoupilo a byla již použita
- NRU (not recently used) - druhá šance s přidáním dirty bitu (stránka byla modifikována), zahodí se primárně stránka, která byla využita i modifikována

Thrashing – "Výprask" – Jestliže proces nemá v paměti dost stránek, dochází k výpadkům stránek velmi často a počítač nedělá nic jiného než výměny stránek

Správa paměti

- Virtuální paměťový prostor procesu je rozdělen na:
  - systémovou část – dostupnou pro proces systémovými voláními (1GiB pro OS, Windows má dokonce 2GiB)
  - uživatelskou část – prostor pro program, zásobník (pro všechna vlákna) a jeho data
    - část program, statická data
    - část halda – heap, dynamická data až do 3GiB
    - část mma – mapovaná paměť, dynamické knihovny
    - část zásobník, limit 8MiB
- paměťová mapa procesu je dostupná v /proc/__pid__/maps



Typy alokace:

- Explicitní – program alokuje a uvolňuje paměť pro dynamické proměnné (např. funkce malloc a free v jazyce C, new/delete v C++)
- Implicitní – program alokuje paměť pro nové proměnné, ale již je neuvolňuje (např. garbage collector v Jave nebo Pythonu)

Hlavní cíle:

- co nejrychlejší provedení funkcí malloc a free, měla by být rychlejší než lineárně k počtu alokovaných bloků
- minimalizovat fragmentaci paměti, co nejlepší využití paměti souvisí s minimální fragmentací (vnitřní i vnější)

Vedlejší cíle:

- Prostorová lokalita:
  - bloky alokované v podobném čase by měly být blízko u sebe
  - bloky podobné velikosti by měly být blízko u sebe
- Implementace by měla být robustní:
  - operace free by měla proběhnout pouze na správně alokovaném objektu
  - alokace by měla umožnit kontrolovat, zda se jedná o odkaz na alokované místo

Různé způsoby:

- alokace a uvolňování
- udržování informace o volných blocích
- výběru volného bloku

viz https://cw.fel.cvut.cz/old/_media/courses/b4b35osy/lekce07.pdf

Souborové systémy

- Trvalé úložiště dat. Rotační nebo flash.
- Posloupnost bloků nebo sektorů
- Oddíly - jeden fyzický disk, vícero logicky → mám jednu plotnu ale disky C,D,E
- Na začátku disku je tabulka
- sektor na disku je boot record

Souborový systém

- Způsob organizace dat na pevném disku
- Data uložená v pojmenovaných souborech
- Soubory v adresářích (složkách)
- Hierarchická struktura adresářů
- Metadata = data o datech
- Organizace pomocí B stromu
-

Bezpečnost, virtualizace

Běžné mechanismy zabezpečení v OS:

- Systémy pro kontrolu přístupu - kontrola, k čemu může daný proces přistupovat
- Autentizační systémy - potvrzení identity toho, jehož „jménem“ proces běží
- Logování - Kvůli auditům, detekci útoků, vyšetřování a obnovu
- Šifrování souborových systémů - HW lze šifrovat celý disk, SW jen souborový systém (nelze šifrovat partition table)
- Správa pověření (credentials)
- Automatické aktualizace

Safety – ochrana okolí před systémem
Security – ochrana systému před okolím