Základy operačních systémů

Struktura OS
- modul pro správu procesů
- modul pro správu paměti
- modul pro správu vstupu / výstupu
- modul pro správu souborů
- síťování
- bezpečnost

- použití os – v serverech, desktopech, mobilních zařízení, tabletech, routerech, embedded zařízeních

OS
- softwarová vrstva, která umožňuje spravovat hardware a poskytuje k němu jednotné rozhraní
- zprostředkovává přístup aplikací k hardwaru
- koordinuje a poskytuje služby aplikacím

Privilegovaný režim
- běží v něm jádro
- všechny instrukce CPU jsou povoleny

Uživatelský režim
- aplikace nemá přímý přístup k HW
- některé instrukce jsou zakázány (například přímý přístup k disku)
- aplikace musí požádat OS o přístup k souboru a ten rozhodne
- operační systém může zasahovat do běhu aplikací
- aplikace může žádat os o službu

- přepínání mezi režimy – softwarovým přerušením, speciální instrukcí (novější, např.: sysenter)

Holý počítač – počítač bez hardwarových ovladačů, pouze strojové instrukce

OS - jako rozšířený stroj
- vysokoúrovňové služby (systémová volání)
- z pohledu programátora: pojmenované soubory, transparentní I/O, neomezená paměť

OS - jako správce souborů
- os poskytuje a spravuje zdroje
- zdroje: čas CPU, paměť, I/O zařízení
- operační systém zajišťuje správnou a řízenou alokaci zdrojů procesům, které jej požadují
- řeší konfliktní požadavky na zdroje (ve frontě?, rozděleny efektivně? …)

Historický vývoj PC
1. **generace**
   - elektronky, propojovací desky
   - stejní lidé je navrhují, stavějí a programují
   - OS neexistují

2. **generace**
   - tranzistory, dávkové OS
   - oddělení návrhářů, výroby, operátorů, programátorů a údržby
   - děrovací štítky
3. **generace**
   - multiprogramování, integrované obvody
   - nutnost čekat na vstup/výstup
   - každá úloha ve vlastní oblasti paměti, více úloh v celkové
   - dávkové systémy
   - systémy se sdílením času (CPU střídavě vykonává úlohy)
4. **generace**
   - mikroprocesory, MS DOS, Unix, Windows NT…

DĚLENÍ OS

a) **podle úrovně sdílení CPU**
   a. jednoprocesový – MS DOS, v daném čase je aktivní jeden program
   b. multiprocesový

b) **podle typu interakce**
   a. dávkový systém – sekvenční dávky, není interakce
   b. interaktivní – interakce uživatel-úloha (Windows, Linux)

OS REÁLNÉHO ČASU

- výsledek má smysl, pouze pokud je zaslán v nějakém omezeném čase
- **hard**
  - zaručena odezva v ohraničeném čase
  - není virtuální paměť, sdílení času, často není souborový systém
  - použití v robotice, telekomunikaci, řízení výroby
- **soft**
  - real task (reálné úlohy) – mají prioritu před ostatními
  - nezaručuje odezvu v daném čase, lze sdílet čas

DALŠÍ DĚLENÍ OS

- podle počtu uživatelů (jedno x více), funkcí (univerzální x specializované), velikosti HW
- podle míry distribuovanosti
  - klasické
  - paralelní
  - síťové
  - distribuované

ZÁKLADNÍ FUNKCE OS

- správa procesů, paměti, souborů, zařízení
- síťování
- ochrana a bezpečnost
- uživatelské rozhraní

PROGRAM
- spustitelný kód v binární podobě, který je uložen na disku

PROCES
- instance běžícího programu
- má přidělený čas CPU, potřebuje paměťový prostor, vstupy, výstupy
- PID – ID procesu = základní identifikátor

SPRÁVA PAMĚTI
- přidělování paměti procesům (alokace/dealokace paměti)
- správa informace o volné a obsazené paměti
- odebírání paměti
- ochrana paměti (přístup pouze pro procesy s oprávněním)

SOUBORY
- vytváření a rušení souborů
- vytváření a rušení adresářů
- správa volného prostoru vnější paměti
- mapování souborů na vnější paměť

I/O SUBSYSTÉM
- správa paměti pro buffer…
- společné rozhraní ovladačů zařízení
- ovladače pro zařízení

VOLÁNÍ SLUŽBY SYSTÉMU
- parametry uložíme na určené místo (registr, zásobník)
- provedeme speciální instrukci – přepne do privilegovaného režimu
- OS převezme parametry, rozpozná službu a vykoná ji
- přepnutí do uživatelského režimu

PŘERUŠENÍ
- přerušení je základní mechanismus v OS
- procesor přeruší vykonávání sledu instrukcí a vykoná obsluhu přerušení, pak pokračuje dál
- víceúrovňová – některé přerušení přeruší jiná

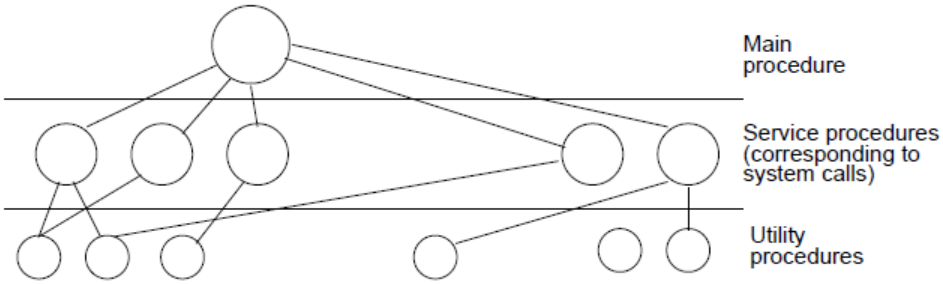
DRUHY PŘERUŠENÍ
- hardwarové přerušení – přichází z I/O, doručuje je řadič přerušení
- vnitřní přerušení – vyvolá je procesor, např.: výpadek stránky paměti, dělení nulou
- softwarové přerušení – speciální strojová instrukce, např.: volání služeb OS z procesu

VEKTOR PŘERUŠENÍ
- index do pole, které obsahuje adresu obslužné rutiny

ARCHITEKTURY OS
- monolitické jádro – jádro jako jeden funkční celek
- mikrojádro – malé jádro, oddělitelné části pracují jako samostatné procesy v user space
- hybridní jádro – kombinace

MONOLITICKÉ JÁDRO

- jeden spustitelný soubor, uvnitř moduly pro jednotlivé funkce
- jeden program, řízení se předává voláním podprogramů
- součást jádra – např.: souborový systém



VRSTVENÉ JÁDRO

- výstavba systému od nejnižších vrstev
- vyšší vrstvy využívají primitiv poskytovaných nižšími vrstvami
- hierarchie procesů

MIKROJÁDRO

- model klient – server
- většinu činností vykonávají samostatné procesy mimo jádro (servery)
- mikrojádro poskytuje pouze nejdůležitější nízko úrovňové funkce, adresní prostor
- pouze mikrojádro běží v privilegovaném režimu – méně pádů
- vyšší režie, složitější návrh systému
- QNX, Hurd, MINIX, Amoeaba

HYBRIDNÍ JÁDRO

- Kombinuje vlastnosti monolitického a mikrojádra
- Windows

IRQ – interrupt request, signál, kterým zařízení žádá procesor o přerušení (IRQL – jeho priorita)
NMI – nemaskovatelné přerušení – např. nezotavitelná HW chyba

PROCESY

- adresní prostor procesu = kód, data, zásobník
- s procesem sdruženy stavové informace (například registry)

ZÁKLADNÍ SLUŽBY OS PRO PRÁCI S PROCESY

- **vytvoření nového procesu**
  - fork (Unix)
  - createProcess (Win32)
- **ukončení procesu**
  - exit (Unix)
  - exitProcess(Win32)
- **čekání na dokončení potomka**
  - wait (waitpid) (Unix)
  - WaitForSingleObject (Win32)

Další služby
- Alokace a uvolnění paměti procesu
- Komunikace mezi procesy (IPC)
- Identifikace na víceuživatelských systémech (UID - user, GID - group)

Soubory
- zakrytí podrobností o discích a I/O zařízeních
- poskytují abstrakci
- systémová volání: vytvoření, zrušení, čtení, zápis, otevření, uzavření
- sekvenční nebo náhodný přístup k datům
- logické sdružování souborů do adresářů a jejich stromových struktur
- ochrana souborů přístupovými právy

Pseudoparalelní běh
- v jednu chvíli je aktivní pouze jeden proces, který je po určité době pozastaven

Rychlost procesů
- není konstantní
- není reprodukovatelná
- procesy nesmějí mít vestavěné předpoklady pro časování
- procesy neběží stejně rychle

Základní stavy procesů
- **běžící** – využívá CPU, koná instrukce
- **připravený** – dočasně pozastaven, aby mohl jiný proces pokračovat
- **blokovaný** – neschopný běhu, dokud nenastane externí událost

Přechody stavů procesu
- plánovač vybere nějaký proces
- proces je pozastaven (vypršelo časové kvantum), plánovač vybere jiný
- proces se zablokuje, protože čeká na událost
- nastala událost, proces běží

Tabulka procesů
- OS si vede evidenci o procesech, které v něm existují
- do tabulky se uloží záznam PCB – proces control block
- PCB obsahuje všechny informace potřebné pro opětovné spuštění procesu
- PCB – pole správy procesů, správy pamětí, správy souborů

Správa procesů
- PID, UID, registry, PC a SP, stav CPU
- stav procesu
- plánovací parametry
- odkazy na rodiče a potomky
- účtovací informace
- nastavení komunikace

Správa paměti
- popis paměti – ukazatel, velikost, přístupová práva
- data – halda, new release, malloc, free…
- zásobník – návratové adresy, parametry funkcí, procedur, lokální proměnné

SPRÁVA SOUBORŮ

- nastavení prostředí – aktuální adresář
- otevřené soubory – způsob otevření, pozice

UKONČENÍ PROCESU

- úspěšně se vykonal
- skončil jeho rodič a OS nedovolí pokračovat (záleží na OS)
- proces překročí limit nějakého zdroje

PŘEPNUTÍ PROCESU

- systém nastaví časovač
- po příchodu přerušení CPU – uloží čítač instrukcí PC do zásobníku, načte adresu obslužného programu přerušení, přepne se do režimu jádra… metoda obsluhy uloží obsah registrů do zásobníku, nastaví nový zásobník, plánovač nastaví proces jako připraven a vybere nový pro spuštění… přepnutí kontextu

SLUŽBY PRO PRÁCI S PROCESY

- pid = fork(), vytvoří přesnou kopii rodičovského procesu, návratová hodnota udává, zda se jedná o rodiče nebo potomka, potomek má pid = 0
    - potomka lze ukončit pomocí exit(), rodič může na potomka čekat wait()
    - exec() – potomek může místo sebe spustit jiný program
    - ve Windows CreateProcess()

VLÁKNA

- sdílejí adresní prostor a otevřené soubory (atributy procesu)
- vlákna mají soukromý čítač instrukcí, obsah registrů a zásobník, plánovací vlastnosti, množinu pending a blocked signálů a data specifická pro vlákno
- jsou v jádře, uživatelském prostoru nebo kombinované
- pokud jsou v user space, pak systémové volání vláknem zablokuje celý proces (many to one)
- vlákna v jádře jsou přidána do PCB tabulky a plánována jádrem (one to one); volání clone()
- pthread_create() – vytvoří vlákno v C

STAVY PROCESŮ

- **zombie** – proces dokončil kód, ale má záznam v tabulce procesů, čeká dokud rodič nepřečte exit status
- **sirotek** – jeho kód stále běží, ale rodič skončil… adoptován pomocí init

PLÁNOVÁNÍ PROCESŮ

- liší se frekvencí spouštění plánovače
- **krátkodobé** – CPU scheduling - běžné
- **střednědobé** – swap out – odsun procesu z vnitřní paměti na disk
- **dlouhodobé** – job sheduling – dávkové zpracování úlohy
- **nepreemptivní**
    - běžící na blokovaný
    - každý proces dokončí svůj CPU burst (vykonávání kódu)
- **preemptivní**
    - proces lze přerušit kdykoliv během CPU burstu
    - běžící na připravený po určitém časovém kvantu
    - vyžaduje speciální hardware (timer)

- vyšší režie při přepínání

ČASOVÝ SOUBĚH

- procesy sdílejí společnou paměť (například inkrementování veličiny)
- projevuje se nedeterministicky, většinu času bez problému, hledání chyby je obtížné
- řešení: atomická operace

KRITICKÁ SEKCE

- místo v programu, kde je prováděn přístup ke společným datům)
- vztahuje se vždy ke konkrétním datům, ke kterým se přistupuje
- řešení: vzájemné vyloučení – žádné procesy nesmějí být současně uvnitř kritické sekce
- proces běžící mimo kritickou sekci nesmí blokovat jiné procesy (např.: bránit jim ve vstupu)
- žádný proces nesmí na vstup do své kritické sekce čekat nekonečně dlouho

ZÁKAZ PŘERUŠENÍ (ŘEŠÍ KS)

- vadí přeplánování procesů (přerušení v systémech se sdílením času)
- zakáže přerušení před kritickou sekcí a pak jej povolí
- není dovoleno v uživatelském režimu
- používáno často uvnitř jádra OS, ale není vhodné pro uživatelské procesy

AKTIVNÍ ČEKÁNÍ

- předpoklady: zápis a čtení jsou nedělitelné operace, ks nemohou mít prioritu, relativní rychlost procesů není známa, proces může pozastavit mimo ks
- v nekonečné smyčce testuje proměnnou, dokud nemá očekávanou hodnotu
- plýtvá časem CPU, použití jen tam, kde čekáme krátce (SPIN LOCK)

TSL

- provádí se nedělitelně, proměnná lock
- operace test a set lock – TSL, TS
- jednoduchý zámek s aktivním čekáním

SEMAFORY

- semafor obsahuje nezáporné celé číslo (a frontu)
- lze mu přiřadit proměnnou pouze při deklaraci
- operace P(s) a V(s), které jsou atomické
- **P(s)** – zablokuje proces, snižuje counter (wait)
- **V(s)** – probudí proces, zvýší counter (signal)
- běží v režimu jádra

Vzájemné vyloučení lze řešit mutexy nebo binárními semafory

MUTEX

- jednoduchý zámek, menší režie oproti TSL
- **yield**() – dobrovolně se vzdá CPU ve prospěch někoho jiného
- může jej odemknout jen vlákno/proces, který jej zamkl
- **reentrantní mutex** – stejné vlákno může získat zámek několikrát, ale pak jej musí tolikrát odemknout

MONITOR

- v monitoru může být v jednu chvíli aktivní pouze jeden proces
- proměnné monitoru nejsou viditelné zvenčí
- speciální typ proměnné = podmínka
- **operace wait a signal**
- pokud žádný proces nad podmínkou nespí, pak signál nedělá nic
- **Hoare** – proces volající c.signal se pozastaví
- **Hansen** – signál smí pouze být poslední příkaz v monitoru, po jeho volání musí proces opustit monitor
- **Java** – umožňuje vytvořit si kopii sdílené proměnné, zapíše zpět pouze při vstupu/výstupu z monitoru **acquire**(), **release**()

MEZIPROCESOVÁ KOMUNIKACE

- **přes sdílenou paměť**
  - objekt je umístěn ve sdílené paměti
  - někdy nebezpečné = globální data přístupná bez ohledu na semafor
  - někdy nemožné = procesy na různých strojích
- **zasíláním zpráv**
  - **send**(adresát, zpráva) – obvykle neblokující (asynchronní)
  - **recieve**(odesílatel, zpráva) – obvykle blokující (asynchronní)

SKUPINOVÉ A VŠESMĚROVÉ ADRESOVÁNÍ

- **skupinové adresování** (multicast) - zprávu pošleme skupině procesů, kde ji obdrží každý proces
- **všesměrové vysílání** (broadcast) – zprávu pošleme všem procesům

DÉLKA FRONTY ZPRÁV (BUFFERING)

- **nulová** – žádná zpráva nemůže čekat, odesílatel se zablokuje
- **omezená kapacita** – blokování při dosažení kapacity
- **neomezená kapacita** – odesílatel se nikdy nezablokuje

PROBLÉM URČENÍ ADRESÁTA

- procesy nejsou trvalé entity nebo existuje více instancí stejného programu = různé PID
- adresujeme frontu zpráv – proces pošle zprávu = vloží se do fronty zpráv… jiný proces přijme zprávu = vyjme zprávu z dané fronty

MAILBOX

- fronta zpráv využívaná více odesílateli a příjemci
- operace recieve je drahá (obzvlášť, pokud jsou procesy na různých strojích)

PORT

- omezená forma mailboxu
- zprávy může vybírat pouze jeden příjemce

Ztráta zprávy – řešení: potvrzení o přijetí nebo pomocí timeoutu

Ztráta potvrzení – řešení: číslování zpráv, duplicitní se ignorují

Problém autentizace - zprávy je možné šifrovat, klíč je známý pouze autorizovaným uživatelům (procesům)

LOKÁLNÍ KOMUNIKACE

- **dvojí kopírování**
  - z procesu odesílatele do fronty v jádře
  - z jádra do procesu příjemce
- **rendezvous**
  - eliminuje frontu zpráv
  - send zavolán dříve než recieve vede k zablokování odesílatele
  - send a recieve = zkopírovat zprávu z odesílatele přímo do příjemce
- **pomocí virtuální paměti**
  - paměť obsahující zprávu je přemapována, tj. nekopíruje se

VZDÁLENÉ VOLÁNÍ PROCEDUR

- dovoluje procesům volat procedury umístěné v jiném procesu
- spojka klienta – reprezentuje vzdálenou proceduru v adresním prostoru klienta
- spojka serveru
- klient zavolá spojku klienta > spojková procedura zabalí argumenty do zprávy a pošle serveru, spojka serveru přijme zprávu a zavolá procedura, procedura se vykoná a návratovou hodnotu pošle spojka serveru zpět ke klientovi> spojka klienta přijme zprávu
- problém: parametry předávané = pouze jednoduchý datový typ a pole
- problém: globální proměnné – použití není možné

SÉMANTIKA VOLÁNÍ RPC

- **právě jednou**
- **alespoň jednou** – opakované volání po timeoutu
- **nejvýše jednou** – klient volání neopakuje

IDEMPOTENTNÍ OPERACE

- operace, kterou lze opakovat se stejným efektem, jaký mělo její první provedení

BARIÉRY

- synchronizační mechanismus pro skupiny procesů; aplikace se skládá z fází
- na konci každé fáze – synchronizace na bariéře (volající pozastaven a čeká na ostatní)
- všechny procesy opustí bariéru současně

Zámky v OS – **poradní** (nejsou vynucené), **mandatorní** (například pro souborový systém)

DISPACHER

- předává řízení procesu vybraného plánovačem (přepne, nevybírá koho)

PLÁNOVAČ

- **rozhodovací mód** – okamžik, kdy jsou vyhodnoceny priority procesu
  - nepreemptivní
  - preemptivní
- **prioritní funkce** – určuje prioritu procesu v systému
  - statická složka – při startu procesu, můžeme určovat prioritu
  - dynamická složka – podle chování procesu (např. jak dlouho již čeká…), proti vyhladovění
- **rozhodovací pravidla** – určí jak se rozhodnout při stejné prioritě
  - náhodně
  - chronologicky (FIFO), cyklickým přidělováním kvanta

DÁVKOVÉ SYSTÉMY

- průchodnost – počet úloh dokončených za časovou jednotku
- doba obrátky – průměrná doba od zadání do dokončení úlohy
- multilevel feedback – více prioritních úrovní, každá má svou frontu úloh, úloha vstupuje s nejvyšší prioritou, proces obsluhuje nejvyšší neprázdnou frontu
- **FCFS** – nepreemptivní, P(r) = r, náhodně (r = celkový čas strávený úlohou)
- **SJF** – nepreemptivní, P(t) = -t, náhodně (t předpokládaná doba běhu úlohy)
- **SRT** – preemptivní, P(a,t) = a-t, FIFO nebo náhodně (a čas strávený během úlohy v systému)
- **MLF** – nepreemptivní, ‚FIFO

INTERAKTIVNÍ SYSTÉMY

- minimalizace doby odpovědi X efektivita

REALTIMOVÉ SYSTÉMY

- dodržení deadlines
- předvídatelnost

PLÁNOVÁNÍ PROCESŮ – V INTERAKTIVNÍCH SYSTÉMECH

- každý proces jedinečný a nepredikovatelný, nelze říct, jak dlouho poběží, než se zablokuje
- vestavěný systémový časovač – provádí pravidelné přerušení (tiky časovače)
- vyvolá se obslužný podprogram v jádře a rozhodne se, zda bude proces pokračovat nebo se spustí jiný => preemptivní plánování

ALGORITMUS CYKLICKÉ OBSLUHY – ROUND ROBIN (RR)

- každý proces má časový interval, ve kterém může běžet
- běží-li na konci kvanta – preempce, naplánován a spuštěn další proces
- proces skončí nebo se zablokuje před uplynutím kvanta = stejná akce jako výše
- krátké kvantum snižuje efektivitu, dlouhé prodlužuje odpověď na požadavky

PRIORITNÍ PLÁNOVÁNÍ

- podobně jako RR, ale některé procesy běží na pozadí
- prioritu lze přidělit staticky, dynamicky

SPOJENÍ CYKLICKÉHO A PRIORITNÍHO PLÁNOVÁNÍ

- prioritní třídy (třída = více procesů se stejnou prioritou)
- prioritní plánování mezi třídami (obsluha třídy s nejvyšší prioritou)
- uvnitř třídy cyklická obsluha

PLÁNOVAČ SPRAVEDLIVÉHO SDÍLENÍ

- spravedlivé sdílení = přiděluje čas každému uživateli proporcionálně = bez ohledu na to, kolik má procesů
- priorita skupiny spravedlivého plánování – má každý uživatel, zohledňuje se vytížení CPU

PLÁNOVÁNÍ POMOCÍ LOTERIE

- procesy obdrží tickety (losy)
- plánovač vybere náhodně jeden ticket, vítěz obdrží jedno kvantum CPU
- důležitější procesy = více ticketů
- spolupracující procesy si mohou předávat tickety

Round Robin	Preemptivní, vyprší kvantum	P() = 1	Cyklicky
Prioritní	Preemptivní, P jiný > P	P = statická + dynamická	Náhodně, cyklicky
Spravedlivé	Preemptivní, P jiný > P	P(p,g) = p-g	Cyklicky
Loterie	Preemptivní, vyprší kvantum	P() = 1	Dle výsledku loterie

PLÁNOVÁNÍ V SYSTÉMECH REÁLNÉHO ČASU

- m = počet periodických událostí, P = perioda, C = délka periody
- zátěž lze zvládnout, pokud platí: suma (C/P) =< 1
- statické plánování = před spuštěním se rozhodne
- dynamické plánování = za běhu

Plánování procesů – vždy součást OS

Plánování vláken – běh plánován OS = kernel-level, plánován uživatelským procesem = OS o nich nic neví, User-level threads

UVÍZNUTÍ (DEADLOCK)

- A zamkne R, B zamkne S,… A požaduje S, B požaduje R
- v množině procesů nastalo uvíznutí, jestliže každý proces množiny čeká na událost, kterou může způsobit jiný proces množiny

PODMÍNKY VZNIKU UVÍZNUTÍ (COFFMANOVY PODMÍNKY)

- **vzájemné vyloučení** – každý zdroj je je buď dostupný nebo přiřazen právě jednomu procesu
- **hold and wait** – proces držící výhradně přiřazení zdroje může požadovat další
- **nemožnost odejmutí** – jestliže přiřazené zdroje nemohou být procesu násilně odejmuty
- **cyklické čekání** – cyklický řetězec 2+ procesů, kde každý z nich čeká na zdroj držený dalším členem
- pro vznik uvíznutí musí být splněny všechny tyto podmínky
- **graficky** – cyklus v grafu je nutnou a postačující podmínkou pro vznik uvíznutí

ŘEŠENÍ UVÍZNUTÍ

- ignorováním – pštrosí algoritmus, tam, kde je vysoká cena za eliminaci
- detekce a zotavení – systém se nesnaží zabránit vzniku, ale pouze uvíznutí detekuje (prohlídne graf alokace zdrojů – detekce kružnic), pokud nastane, provede zotavení (vlastníkovi dočasně odejme zdroj = preempce X zrušení změn (rollback) = pomocí checkpointů X zrušení procesu = jeden nebo více procesů ukončíme)

PREVENCE UVÍZNUTÍ

- alespoň jedna z Coffmanových podmínek nemůže platit
- vzájemné vyloučení => nikdy nepřiřadíme zdroj výhradně, spooling
- **hold&wait** = požadujeme alokaci všech zdrojů před spuštěním procesů
- **cyklické čekání** = očíslovat zdroje a požadavky musejí být v číselném pořadí
- **dynamické zabránění** – systém rozhodne, zda je přiřazení zdroje bezpečné = existuje alespoň jedna posloupnost procesů, ve které nenastane uvíznutí

ZDROJE

- **přeplánovatelné** – lze je odebrat procesu bez negativních efektů
- **nepřeplánovatelné** – proces havaruje, pokud jsou mu odebrány
- **sériově využitelné** – proces alokuje, použije a uvolní zdroj
- **konzumovatelné zdroje** – např. zprávy, které produkuje jiný proces

PRÁCE SE ZDROJEM

- žádost (request) – uspokojena ihned nebo proces čeká, syscall
- použití (use) – například tisk na tiskárně
- uvolnění (release) – proces uvolní zdroj, syscall

VYHLADOVĚNÍ

- proces nikdy neobdrží zdroj

SPRÁVA PAMĚTI

- OS alokuje paměť procesům (malloc) – alokována z haldy
- OS zařazuje paměť do volné paměti (release)
- OS udržuje info o volné paměti
- sbrk – syscall, přidělí další stránky paměti
- pointer ukazuje na virtuální adresu, která se uvnitř procesoru převede na fyzickou

MECHANISMY SPRÁVY PAMĚTI

- **základní** – program je v paměti po celou dobu svého běhu
- **s odkládáním** – programy jsou přesouvány mezi hlavní pamětí a diskem

ZÁKLADNÍ MECHANISMY PRO SPRÁVU PAMĚTI

- **jednoprogramové systémy**
  - spouštíme pouze jeden program v jednom čase – umožní využít všechnu paměť, kterou OS nepotřebuje
  - OS ve spodní části RAM (miniPC)
  - OS v horní části ROM (zapouzdřené systémy)
  - OS v RAM, ovladače v ROM

- **multiprogramování s pevným přidělením paměti**
  - rozdělíme paměť na N oblastí (klidně různé velikosti)
  - více front – každá úloha do nejmenší oblasti, kam se vejde
  - jedna front – po uvolnění oblasti z fronty vybrat největší úlohu, která se vejde
  - využití CPU … u = 1 – p^n … n je stupeň multiprogramování, p čas

- **multiprogramování s proměnnou velikostí**
  - v čase se mění počet oblastí, velikost oblastí, umístění oblastí
  - lepší využití paměti X komplikovanější alokace
  - může vzniknout mnoho volných děr – řešení kompaktace paměti (drahá)

SPRÁVA PAMĚTI BITOVOU MAPOU

- paměť se rozdělí na alokační jednotky stejné délky, menší alokační jednotky = větší mapa
- hledání volného úseku je náročná operace
- výhodou je konstantní velikost bitmapy

SPRÁVA PAMĚTI SEZNAMEM

- seznam alokovaných a volných oblastí
- položka oblasti – typ, počáteční adresa, velikost oblasti
- proces končí, (P) se nahradí dírou (H)
- díry vedle sebe se sloučí
- **first fit**
    - prohledává, dokud se nenajde dostatečně velká díra
    - rychlý
- **next fit**
    - prohledává od místa, kde skončilo poslední hledání
- **best fit**
    - prohlédne celý seznam a vezme nejmenší díru, do které se proces ještě vejde

VYLEPŠENÍ SPRÁVY PAMĚTI

- **oddělené seznamy pro proces a díry**
    - dobré pro rychlou alokaci dat z I/O zařízeních
    - alokace – prohledá seznam děr
    - dealokace – složitá, přesuny mezi seznamy z děr do procesů
- **oddělené seznamy a díry seřazené**
    - rychlé pro first fit
    - dealokace je nákladná
- **quick fit**
    - samostatné seznamy nejpoužívanější délek
    - alokace – rychlá, dealokace náročnější

ASYMETRIE MEZI PROCESY A DÍRAMI

- procesy se nesloučí
- díry se sloučí
- při normálním běhu je počet děr poloviční oproti počtu procesů

BUDDY SYSTEMS

- seznam volných bloků (násobky 2)
- plýtvá místem, ale je rychlý
- jádro Linuxu používá pro správu buddy system

Modul pro správu paměti

- informace o přidělení paměti (volná x přidělená paměť)
- přidělování paměti na žádost
- uvolnění paměti, odebírání paměti procesům
- ochrana paměti

Realokace a ochrana

- programy běží na různých fyzických adresách – realokace (jednou na X, pak na Y)
- paměť je chráněna před zásahy jiných programů – ochrana

Ochrana paměti přístupovým klíčem

- paměť rozdělena do bloků, každý má 4 bitový klíč ochrany, PSW procesu obsahuje klíč; klíč a kód mění pouze OS

Jednotka správy paměti (MMU)

- součástí CPU, obsahuje dva registry
  - báze – počáteční adresa paměti
  - limit – velikost oblasti
- dynamická (za běhu, pomocí báze a limitu) x statická relokace
- pro dávkové systémy není potřeba správy paměti s odkládáním

Správa paměti s odkládáním celých procesů

- pro systémy se sdílením času (více procesů v paměti současně)
- odkládáním celých procesů (swapping) – proces se uloží na disk
- virtuální pamětí

Swapping

- data procesu mohou růst => je alokováno více paměti než je třeba
- pokud je potřeba více paměti, než je alokováno:
  - přesunout proces do větší oblasti
  - překážející proces odložit
  - odložit žadatele o paměť
  - zrušit proces (out of memory)
- stejné algoritmy jako pro přidělení paměti

Překrývání (overlays)

- proces může být větší než dostupná fyzická paměť
- program rozdělen na moduly
- zavádění modulů zařizuje OS
- rozdělení programů i dat na části navrhuje programátor

Virtuální paměť

- proces může být větší než dostupná fyzická paměť
- ve skutečné paměti je pouze část adresového prostoru
- dnes se využívá nejčastěji
- ve fyzické paměti je aktuálně využívaná část, zbytek je na disku
- fyzická paměť slouží jako cache virtuálního adresního prostoru procesů
- pokud je požadovaná část virtuálního prostoru ve fyzické paměti, pak MMU převede její adresu (virtuální adresu) na fyzickou adresu
- pokud požadovaná část není, pak je načtena z disku
- většina systémů používá stránkování pro správu virtuální paměti
- stránky – části virtuální paměti, mají pevnou délku (násobek 2)
- fyzická paměť – rámec, stejná délka

VÝPOČET ADRESY U STRÁNKOVÁNÍ

- virtuální adresu tvoří číslo stránky a offset.
    - číslo stránky = virtuální adresa (VA) / velikost stránky
    - offset = VA % velikost stránky
- převedeme číslo stránky na číslo rámce
    - v tabulce, kde číslo stránky je index
- sestavíme fyzickou adresu (FA)
    - FA = rámec * velikost rámce + offset

VÝPADEK STRÁNKY

- stránka není mapovaná
- OS zachytí výjimku a nastane přerušení, přepne na jiný proces, zahájí zavádění stránky
- po zavedení stránky se upraví tabulka stránek
- proces může pokračovat
- pro urychlení se použije kopie části tabulky v MMU

VNĚJŠÍ FRAGMENTACE

- zůstávají nepřidělené úseky paměti
- vzniká například dynamickým přidělováním (malé mezery)
- při stránkování vnější fragmentace nenastává

VNITŘNÍ FRAGMENTACE

- část přidělené oblasti je nevyužita (dostaneme stránku, ale využijeme jen její část)

ČISTÉ STRÁNKOVÁNÍ

- bez odkládací oblasti
- logický adresní prostor procesu je mapován do nesouvislých částí paměti
- OS udržuje 1 tabulku rámců a tabulku stránek pro každý proces

TABULKA RÁMCŮ A TABULKA STRÁNEK PROCESŮ

- rámců
    - pro správu fyzické paměti, informace o obsazenosti rámců

- procesů
    - mapuje číslo stránky na číslo fyzického rámce
    - poskytuje informace o příznacích ochrany
    - řeší realokaci
- přepnutí na jiný proces = MMU přepne na jinou tabulku stránek

Problémy čistého stránkování

- velikost tabulky stránek
    - řeší víceúrovňová struktura
- rychlost převodu VA na FA
    - TLB – HW cache pro MMU
    - zpomalení jen 5 - 10 %

Obsah položky v tabulce stránek

- číslo rámce
- příznak platnosti
- příznak ochrany
- bit modified
    - zápis jej nastaví na 1
- bir referenced
    - zápis/čtení jej nastaví na 1
- móre…

Stránkování na žádost

- s ukládacím prostorem
- při vytvoření procesu se vytvoří tabulka stránek, alokuje místo na disku pro odkládání stránek…
- při běhu – žádná stránka není v paměti… tj. 1. přístup způsobí výpadek stránky
- OS zavede stránku do paměti, čímž postupně vytváří pracovní množinu stránek
- má-li proces svou pracovní množinu stránek v paměti, pak pracuje bez mnoha výpadků, dokud se pracovní množina stránek nezmění

Ošetření výpadku stránky

- Výpadek způsobí přerušení
- OS zjistí, pro kterou stránku nastalo přerušení
- určí umístění stránky na disku
- najde rámec, do kterého bude stránka zavedena
- načte stránku do rámce
- změní položku v tabulce stránek
- návrat k procesu

Algoritmy nahrazování stránek

- pokud jsou všechny stránky obsazené, musí se některé stránky nahradit

Algoritmus FIFO

- udržuji seznam stránek v pořadí, ve kterém byly zavedeny

- vyhazuji nejstarší stránku
- není nejvhodnější, často používané stránky mohou být v paměti dlouho a navíc trpí
- Béládyho anomálie – dojde k většímu počtu výpadků s rostoucí počtem stránek

ALGORITMUS MIN/OPT

- optimální – nejmenší možný výpadek stránek
- vyhodíme ty, které nikdo dlouho nebude požadovat
- není realizovatelný
- pouze pro srovnání v simulátoru

ALGORITMUS LRU (LEAST RECENTLY USED)

- vyhodí se nejdéle používaná
- stránky používané v posledních instrukcích se budou nejspíše používat i v následujících
- obtížná implementace
- SW řešení není použitelné (seznam stránek v pořadí referencí), nutná podpora HW
- HW – MMU obsahuje čítač (64 bitový), který se při každém přístupu do paměti zvětší
- každá položka v tabulce stránek obsahuje místo pro uložení čítače
- při odkazování se zapíše čítač do položky pro odkazovanou stránku
- vyhodí se stránka s nejnižším číslem
- řešení maticí n*n bitů
    - na začátku nulová matice
    - odkaz na stránku k-tého rámce
        - všechny bity k-tého řádku matice se změní na 1
        - všechny bity k-tého sloupce matice se změní na 0
    - řádek s nejnižší binární hodnotou = nejdéle nepoužitá stránka
- nejlepší z realizovatelných řešení
- Béládyho anomálie nemůže nastat

ALGORITMUS NRU

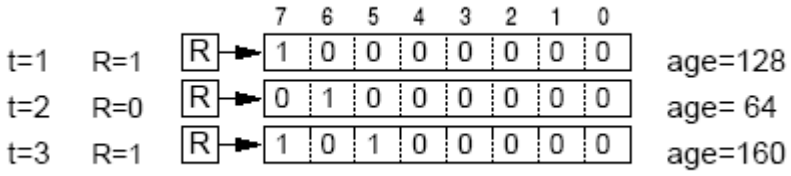
- Snaha vyhazovat nepoužívané stránky
- HW – stavové bity referenced (R) a dirty (M)
- bit R se nastaví při čtení nebo zápisu, M při zápisu
- M zůstává 1, dokud je SW nenastaví zpět
- R je periodicky měněn na 0 OS
- 4 kategorie stránek
    - Třída 0: R = 0, M = 0
    - Třída 1: R = 0, M = 1
    - Třída 2: R = 1, M = 0
    - Třída 3: R = 1, M = 1
- NRU vyhodí stránku z nejnižší neprázdné třídy
- výběr mez stránkami je náhodný
- lepší je vyhodit modifikovanou stránku, která nebyla použita 1 tik, než modifikovanou stránku, která se používá
- jednoduchý, srozumitelný, snadno implementovatelný
- neefektivní

ALGORITMY SECOND CHANCE A CLOCK

- vycházejí z FIFO
- Second Chance – podle bitu R… R= 0 pak je stránka nepoužívaná = vyhodíme, R = 1, pak nastavíme R na 0 a přesuneme na konec seznamu (druhá šance)
- pokud jsou všechny stránky R = 1, pak jako FIFO
- algoritmus končí po (počtu rámcích + 1) krocích
- Clock – jako Second Chance, ale použije se kruhový seznam

AGING – APROXIMACE LRU

- pole o velikosti N bitů (pro každou položku stránek)
- na začátku age = 0
- při každém přerušení časovače se pro každou stránku:
    - posune pole stáří o jeden bit vpravo
    - zleva se přidá hodnota R
    - nastaví se R na 0



- nevýhoda = uchovává pouze omezenou historii, několik stránek může mít stejnou hodnotu age, pak se vybere náhodně

ALOKACE FYZICKÝCH RÁMCŮ

- globální

    - pro vyhození uvažujeme všechny rámce
    - lepší průchodnost systému
    - na běh procesu má vliv běh ostatních procesů
- lokální
    - uvažují se pouze rámce alokované procesem
    - počet stránek alokovaných pro proces se nemění
    - nejjednodušší – všem procesům dát stejně
    - proporciální – každému díl podle jeho velikosti
    - nejlepší – podle frekvence výpadů stránek PFF

ZLODĚJ STRÁNEK (PAGE DAEMON)

- v systému se běžně udržuje určitý počet volných stránek
- když klesne počet volných stránek na určitou mez, pustí se page daemon, který uvolní určité množství stránek, které se hned nepřidělí (lze je vrátit zpátky)

ZAMYKÁNÍ STRÁNEK

- zabrání odložení stránky
    - části jádra
    - stránka, kde probíhá I/O
    - tabulky stránek
    - nastavení uživatelem mlock()

ZAHLCENÍ

- proces pro svůj rozumný běh potřebuje určitou pracovní množinu stránek
- pokud se pracovní množina stránek nevejde do paměti, nastává zahlcení
- systém intenzivně pracuje s diskem a běh programů se zpomalí
- řešení = snížit úroveň multiprogramování

VIRTUÁLNÍ PAMĚŤ

- jednorozměrná
- rozsah u 32 bit (2 GB proces, 2 GB systém nebo 3+1)
- efektivnější využití fyzické paměti – bez vnější fragmentace
- režie při přechodu, režie procesoru, režie I/O, vnitřní fragmentace

SEGMENTACE

- segmenty = nezávislé adresové prostory, logické seskupení informací
- každý segment je lineární posloupností adres od 0
- přístup k souborům – 1 soubor = 1 segment
- sdílené knihovny – vložit knihovnu do segmentu a sdílet mezi více programy
- každý segment – samostatná ochrana (může mít)

ČISTÁ SEGMENTACE

- každý odkaz do paměti – dvojice (selektor, offset)
    - selektor – číslo segmentu, určuje segment
    - offset – relativní adresa v rámci segmentu

- musí být technicky možné přemapovat selektor a offset na lineární adresus
- tabulka segmentů
    - počáteční adresa (báze)
    - rozsah segmentu (limit)
    - příznaky ochrany segmentu

SEGMENTACE NA ŽÁDOST

- segment buď v paměti, nebo odložený na disku
- adresování segmentu, který není v paměti – výpadek
- HW podpora – bity v tabulce segmentů

SEGMENTACE SE STRÁNKOVÁNÍM

- velké segmenty je nepraktické celé udržovat v paměti – řešení: stránkování segmentů
- implementace – např.: každý segment = vlastní tabulka stránek
- virtuální adresa -> lineární adresa -> fyzická adresa
- virtuální – používá proces
- lineární – po segmentaci, pokud není stránkování, pak je zároveň fyzickou
- fyzická – adresa do fyzické paměti RAM

LDT – local descriptor table – segmenty lokální pro proces (kód, data, zásobník)

GDT – global descriptor table – pouze jedna sdílená všemi procesy (systémové segmenty, OS)

SELEKTOR SEGMENTU

- 16 bitový
- 13 bitů – index do GDT nebo LDT
- 1 bit – 0 = GDT, 1= LDT
- 2 bity – úroveň privilegovanosti (0-3)

DESKRIPTOR SEGMENTU

- 64 bitový, 32 bitů = báze, 20 bitů limit, zbytek příznaky

KONVERZE NA FYZICKOU ADRESU

- Proces adresuje paměť podle segmentového registru
- CPU použije popisovač segmentu

(1-10; 2013/14)