Pojem operační systém (OS)

- OS jako rozšíření počítače
  - Skrývá komplikované detaily hardware
  - Poskytuje uživateli „virtuální stroj“, který se snáze ovládá a programuje
- OS jako správce systémových prostředků
  - Každý program dostává prostředky v čase
  - Každý program dostává potřebný prostor na potřebných prostředcích
- Program, který řídí vykonávání aplikačních programů
- Styčná plocha (interface) mezi aplikačními programy a hardware
- Cíle OS:
  - Uživatelské „pohodlí“
  - Účinnost
    - Umožnit, aby systémové zdroje počítače byly využívány efektivně
  - Schopnost vývoje
    - Umožnit vývoj, testování a tvorbu nových systémových funkcí, aniž by se narušila činnost existujícího OS

Typy operačních systémů

- OS „velkých“ (mainframe) počítačů
- OS datových a síťových serverů
- OS multiprocesorových počítačů
- OS osobních počítačů a pracovních stanic
- OS reálného času (Real-time OS)
- Vestavěné OS (tiskárna, pračka, ...)
- OS čipových karet (smart card OS)
- ... a mnoho dalších specializovaných systémů

Požadavky aplikací v reálném čase a Real-time OS

- Správná funkce systému nezávisí jen na logickém výsledku ale i na čase, kdy bude výsledek získán
- Úlohy a procesy reagují a řídí události vně systému, které nastávají v „reálném čase“ a potřebná reakce musí být včasná
- Charakteristiky OS RT
  - Determinismus
    - Operace jsou prováděny ve fixovaných, předem určených časech nebo časových intervalech
    - Reakce na přerušení musí proběhnout tak, aby systém byl schopen obsluhy všech požadavků v požadovaném čase (včetně vnořených přerušení)
  - Uživatelské řízení
    - Uživatel specifikuje:
      - Priority
      - Práva procesů
      - Co musí vždy zůstat v paměti
  - Spolehlivost
    - Degradace chování může mít katastrofální důsledky
  - Zabezpečení
    - Schopnost systému zachovat v případě chyby aspoň částečnou funkcionalitu a maximální množství dat

Jádro OS (JOS)

Jádro operačního systému je základním prvkem OS a poskytuje nejelementárnější služby při práci se systémem. V současnosti nejpoužívanější architekturou je tzv. monolitické JOS. Jde o systém hierarchicky organizovaných vrstev, z nichž každá se stará o příslušnou skupinu

systémových funkcí. Daná vrstva využívá vždy funkcí nižších vrstev, které implementují její funkční primitiva. Následuje příklad jednotlivých vrstev:

1. Vrstva přidělování procesoru na nižší úrovni (obsluha přerušení a MMU), nově tzv. *Hardware Abstraction Layer* = HAL)
2. Řízení vstupu a výstupu (ovladače periferií)
3. Systém ovládání souborů (SOS = FMS)
4. Virtualizace paměti
5. Přidělování procesoru na vyšší úrovni (plánování a přidělování procesoru výpočetním procesům)
6. Synchronizace procesů a meziprocesní komunikace

...

n. Dispečer služeb

U moderních OS se vyskytuje také architektura tzv. mikrojádra (mikrokernel), ve které je jádro zodpovědné jen za nejzákladnější systémové funkce(adresní prostory meziprocesní komunikace pomocí komunikačních schránek, základní plánování procesů/vláken) a vše ostatní je ponecháno v aplikačním adresním prostoru.

Služby JOS

- Několik (zdánlivě nezávislých) skupin služeb:
  - správa výpočetních procesů
  - přístup k datům v souborech a na periferiích
  - správa souborů a souborových systémů
  - různé další služby
- Základní služby

Správa procesů

Služba	Popis
pid = fork()	Vytvoří potomka identického s rodičem
pid = waitpid(pid, &stat, options)	Čeká až zadaný potomek skončí
s = execve(name, argv, environp)	Nahradí „obraz“ procesu jiným „obrazem“
exit(status)	Ukončí běh procesu a vrátí status

Práce se soubory

Služba	Popis
fd = open(filename, how, ...)	Otevře soubor pro čtení, zápis či modif.
s = close(fd)	Zavře otevřený soubor (uvolní paměť)
n = read(fd, buff, nbytes)	Přečte data ze souboru do paměti *buff*
n = write(fd, buff, nbytes)	Zapíše data z paměti *buff* do souboru
pos = lseek(fd, offset, whence)	Posouvá *ukazatel aktuální pozice* souboru
s = stat(filename, &statbuffer)	Dodá stavové informace o souboru

Práce s adresáři souborů a správa souborů

Služba	Popis
s = mkdir(name, mode)	Vytvoří nový adresář s danými právy
s = rmdir(name)	Odstraní adresář
s = link(name1, name2)	Vytvoří položku *name2* odkazující na *name1*
s = unlink(name)	Zruší adresářovou položku
s = mount(spec, name, opt)	„Namontuje“ souborový systém
s = umount(spec)	„Odmontuje“ souborový systém

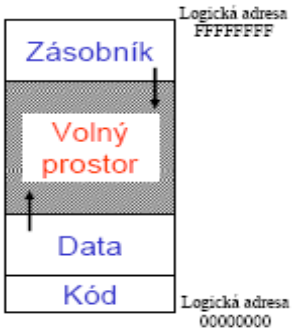
Další služby

Služba	Popis
s = chdir(dirname)	Změní „pracovní adresář“
s = chmod(fname, mode)	Změní „ochranné příznaky“ souboru
s = kill(pid, signal)	Zašle *signál* danému procesu
a mnoho dalších služeb	

Výpočetní proces

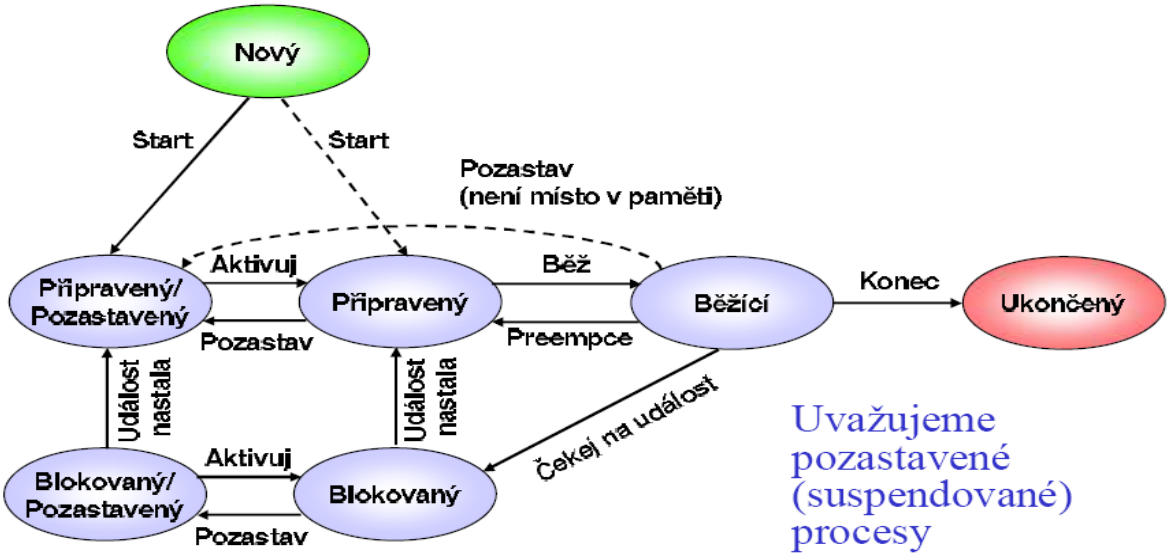
Jde o vykonávaný program (instanci programu běžící v počítači), tedy entitu které se přiděluje procesor. Každý proces je popsán svým deskriptorem, který slouží JOS jako zdroj informací o procesu pro plánování jeho spouštění.

- Skládá se ze tří komponent
  - běžícího programu (kódu)
  - přidružených dat potřebných pro běh programu
  - „*kontextu*“ programu (tzv. systémový segment procesu)
    - Veškeré informace, které potřebuje JOS, aby mohlo spravovat proces a jemu přidělené systémové prostředky
    - Datové struktury, které používá proces pro styk s OS a svým okolím (prostředí procesu – *environment*)



- Každý proces má 3 segmenty:
  - kód (text)
  - data
  - zásobník
- Hranice segmentů „hlídány“ ochranou paměti

- stav procesu v čase lze vždy popsat jedním ze sedmi stavů uvedených na obrázku



Správa a plánování procesů

Úkolem správy a plánování procesů je

- Prokládat vykonávání jednotlivých procesů s cílem maximálního využití procesoru
- Přidělovat procesům požadované systémové prostředky
- Umožňovat procesům vytváření a spouštění dalších procesů
- Vytvářet podporu pro vzájemnou komunikaci mezi procesy
- Poskytovat aplikačním procesům funkčně bohaté, bezpečné a konzistentní rozhraní, včetně uniformní abstraktní prezentace systémových prostředků (např. souborů)

Typy plánování procesů

- Dlouhodobé plánování
  - Při vzniku nového procesu rozhoduje, zda proces může být zařazen mezi připravené
  - Řídí stupeň multiprogramování - paralelismu
- Střednědobé plánování
  - Souvisí s odkládáním procesů
  - Je proces zcela nebo částečně v paměti?
- Krátkodobé plánování
  - Který připravený proces dostane procesor

- Různá kriteria
  - Uživatelsky orientovaná
    - čas odezvy
      - doba od vzniku požadavku do reakce na něj
    - doba obrátky
      - doba od vzniku procesu do jeho dokončení
    - konečná lhůta (*deadline*)
      - požadavek dodržení stanoveného času dokončení
    - předvídatelnost
      - Úloha by měla být dokončena za zhruba stejnou dobu bez ohledu na celkovou zátěž systému
      - Je-li systém vytížen, prodloužení odezvy by mělo být rovnoměrně rozděleno mezi procesy
  - Systémově orientovaná
    - průchodnost
      - počet procesů dokončených za jednotku času
    - využití procesoru
      - relativní čas procesoru věnovaný aplikačním procesům
    - spravedlivost
      - každý proces by měl dostat svůj čas (ne„hladovění“ či „stárnutí“)
    - vyvažování zátěže systémových prostředků
      - systémové prostředky (periferie, hlavní paměť) by měly být zatěžovány v čase rovnoměrně

Plánovač procesů

Rozhoduje kterému z připravených procesů bude přidělen procesor.

- Aktivace plánovače
  - Obslužná rutina přerušení na svém konci může ohlásit *významnou událost* v systému (např. dokončení přenosu dat, vyčerpání časového kvanta)
  - Významná událost aktivuje plánovač, který rozhodne, co dále
  - Plánovač může přepnout kontext →přepnutí od jednoho procesu k jinému JEN v důsledku nějakého PŘERUŠENÍ
- obecně má plánovač 3 komponenty
  1. Rozhodování-kterému procesu přidělit procesor
     - Pracuje nad frontou připravených
     - Rozhoduje se když:
       1. Nový proces se stane připraveným
       2. Běžící proces skončí
       3. Blokovaný proces změní svůj stav na připravený
       4. Běžící proces se zablokuje
       5. Běžící proces vyčerpá časové kvantum
       6. Připravenému procesu vzroste priorita nad prioritu procesu běžícího
     - Typy rozhodování při plánování
       - Bez preempce
         - Nechá běžet proces až do jeho zablokování nebo ukončení
         - Kooperativní plánování
           - Všechny procesy povinně (programově) volají službu jádra a blokují se – tak dávají možnost přepnutí k jinému procesu
       - Preemptivní plánování
         - Univerzální, avšak preempce může nastat v nevhodný okamžik – kritická sekce přistupující ke sdílenému prostředku
         - Nutné zamykání prostředků

2. Prioritní funkce-všem připraveným procesům určit efektivní priority
   - Určuje efektivní prioritu připravených procesů
   - Může záviset na vlastnostech procesů
   - Základní prioritní úroveň
     - Vysoká pro interaktivní procesy
     - Nízká pro dávkové zpracování „na pozadí“
   - Nároky na paměť (velká režie odkládání):
     - „Malý“ proces – málo odkládání, více malých procesů lze obsluhovat
   - Časové vlastnosti procesu
     - Celkový spotřebovaný čas
     - Relativní spotřeba časových kvant
     - Vyšší priorita krátkých procesů

3. Arbitrážní pravidlo-co činit, jsou-li dva procesy rovnocenné(se shodnou prioritou)
   - Aplikuje se na připravené procesy se stejnou efektivní prioritou
   - Náhodná volba
     - Používá se zřídka
   - Chronologické řazení
     - Nejčastější – klasická fronta (FIFO)
     - Proces s danou prioritou se řadí na konec fronty
   - Cyklické řazení (*round robin*)

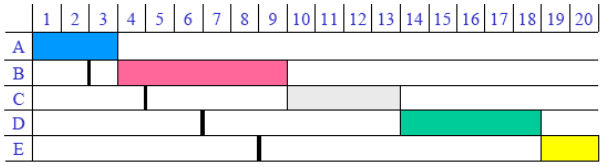
Plánovací algoritmy
- FCFS (*First Come First Served*)
- cyklické (*Round-Robin*)
- SPN (*Shortest Process Next*)
- SRT (*Shortest Remaining Time*)
- HRRN (*Highest Response Ratio Next*)
- zpětnovazební (*Feedback*)

Příklad sady procesů

Proces	Příchod	Potřebný čas
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

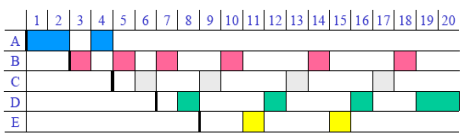
Plánování FCFS

- Nejjednodušší nepreemptivní plánování
- Nový proces se zařadí na konec fronty
- Průměrné čekání může být velmi dlouhé
- Příklad:



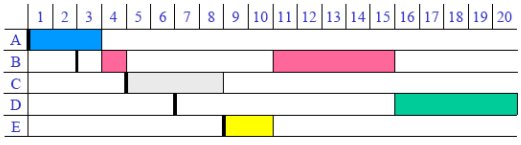
Cyklické plánování (Round-robin)

- Z podstaty preemptivní, na základě časových kvant (10 – 200ms)
- Shodná priorita všech procesů
- Fronta připravených je kruhová – přerušený proces na konec
- Nevyčerpání kvanta (zablokování) – vypadne z fronty připravených
- Žádný proces nedostane 2 kvanta za sebou (pokud není jediný)
- Efektivita silně závisí na volbě kvanta
  - Veliké kvantum => FCFS
  - Malé kvantum => časté přepínání kontextu => značná režie
  - Empirické pravidlo: 80% procesů by nemělo vyčerpat kvantum
- Příklad:



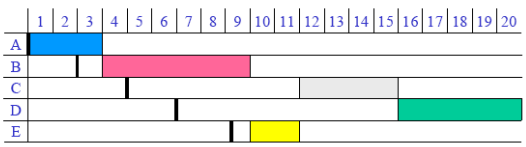
Plánování SRT

- SRT = Shortest Remaining Time (nejkratší zbývající čas)
  - Preemptivní varianta SPN (preempce při příchodu procesu)
  - Proces dostane proces potřebující nejméně času do svého skončení
  - Opět problém s odhadem
- Příklad:



Plánování SPN

- SPN = Shortest Process Next (nejkratší jako příští)
  - Opět nepreemptivní
  - Připravený proces s nejkratší očekávanou dobou běhu dostane procesor
  - Problém s odhadem doby běhu
  - Krátké procesy předbíhají delší, nebezpečí stárnutí dlouhých procesů
- Příklad:

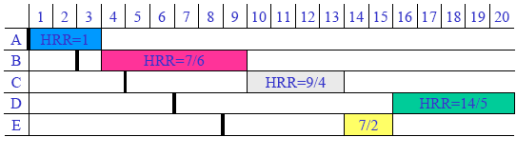


Plánování HRRN

- HRRN = Highest Response Ratio Next (proces s nejvyšší poměrnou odezvou následuje)
  - Zvol jako následující proces s maximálním HRR

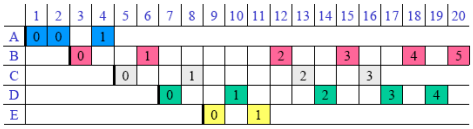
$$HRR = \frac{CasStravenyCekanim + PotrebnyCas}{PotrebnyCas}$$

  - $HRR \geq 1$
  - Bez preempce
  - Kratší procesy favorizovány, dlouho čekajícím procesům roste šance – prevence stárnutí
- Příklad:



Zpětnovazební plánování

- Základní problém: Neznáme předem časy, které budou procesy potřebovat
- Východisko: Penalizace procesů, které běžely dlouho
- Řešení:
  - Dojde-li k preempci přečerpáním časového kvanta, procesu se snižuje priorita (pokud není jediný připravený)
  - Implementace pomocí dělených front
  - Nad každou frontou samostatně běží algoritmus typu cyklického plánování
- Příklad



Meziprocesní komunikace, synchronizace

Souběžný přístup ke sdíleným datům může způsobit jejich nekonzistenci; nutná kooperace procesů

• Synchronizace běhu procesů – čekání na událost od jiného procesu

- Vzájemné vyloučení s aktivním čekáním- proces čeká na událost opakovaným prováděním kontroly zda instrukce nenastala. Tento způsob mrhá strojovým časem a může způsobit také nefunkčnost systému
- Synchronizace bez aktivního čekání – blokování procesu pomocí systémových atomických primitiv
  – sleep() místo aktivního čekání – proces se zablokuje
  – wakeup(process) probuzení spolupracujícího procesu při opouštění kritické sekce
- Semafory – obecný synchronizační nástroj
  • Semafor *S* – systémem spravovaný objekt se základní vlastností typu celočíselná proměnná
  • Dvě standardní atomické operace nad semaforem
    – acquire(*S*) [někdy nazývaná wait() nebo down()]
    – release(*S*) [někdy nazývaná signal() nebo up()]
    Tyto operace jsou implementovány v jádře jako nedělitelné a tak, aby je žádné dva procesy nemohly provést nad týmž semaforem současně.
  • Sémantika těchto operací:

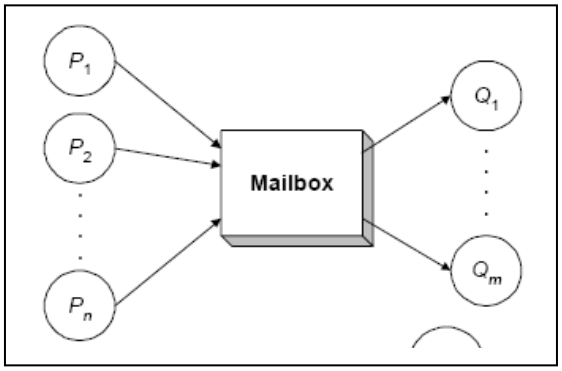
```
acquire(S) {
    while (S <= 0)
        ;   // zablokuj volající proces
    S--;
}
```

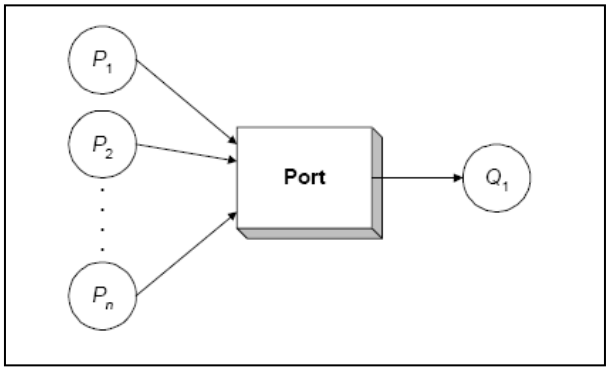
```
release(S) {
    S++;
    // Čeká-li proces před
    // semaforem, probuď ho
}
```

  •Základní typy semaforů
    - Čítající (obecný) semafor – jeho celočíselná hodnota není omezena
    - Binární semafor – hodnota je pouze 0 nebo 1
      • Znám též jako zámek (mutex)
      • Snáze se implementuje
- Monitory – jsou to synchronizační nástroje vysoké úrovně
  • Umožňuje bezpečné sdílení libovolného datového typu
  • Monitor je jazykový konstrukt v jazycích „pro paralelní zpracování“
    – Podporován např. Concurrent Pascal, Java, ...

• Procedury definované jako monitorové procedury se implicitně vzájemně vylučují

- Synchronizace pomocí zasílání zpráv-mechanismus mailboxů a portů
  - • Mailbox je schránka pro předávání zpráv
    - – Může být soukromá pro dvojici komunikujících procesů nebo sdílená více procesy
    - – JOS vytvoří mailbox na žádost procesu a tento proces je pak jeho vlastníkem
    - – Vlastník může mailbox zrušit nebo bude zrušen při skončení vlastníka
  - • Port je schránka vlastněná jedním příjemcem
    - – Zprávy do portu může zasílat více procesů
    - – V modelu klient/server je přijímacím procesem server
    - – Port zaniká ukončením přijímacího procesu





• Komunikace mezi procesy – výměna zpráv
– komunikace – způsob synchronizace, koordinace různých aktivit
– může dojít k „uváznutí" – každý proces v jisté skupině procesů čeká na zprávu od jiného procesu v téže skupině
– může dojít ke „stárnutí" či „hladovění" – dva procesy si opakovaně posílají zprávy zatímco třetí proces čeká na zprávu nekonečně dlouho

Správa paměti

Požadavky na správu paměti

- • Pro běh procesu je nutné, aby program, který ho řídí byl umístěn v operační paměti
  - – Vstupní fronta – kolekce procesů „na disku" čekajících na přidělení místa v hlavní paměti
  - – Program se přetváří do formy schopné interpretace procesorem ve více krocích
    - • Cíl: vazba adres instrukcí a dat na skutečné adresy v operační paměti
- • Logická organizace
  - – Programy jsou sady „modulů" s různými vlastnostmi
    - • Moduly s instrukcemi jsou označovány „execute-only"
    - • Datové moduly jsou „read-only" nebo „read/write"
    - • Některé moduly jsou „soukromé" (private), jiné jsou „veřejné" (public)
    - • OS a HW musí podporovat práci s moduly tak, aby se dosáhlo požadované ochrany a sdílení
  - – Požadavky na sdílení
    - • Více procesů může sdílet společné úseky (sdílené struktury), aniž by tím docházelo k narušení ochrany paměti

Výměny, odkládání(swapping)

- • Proces (nebo jeho části) může být vyměňován mezi operační a sekundární pamětí
- • Potřebujeme rychlou sekundární paměť s přímým přístupem
- • Trvání výměn je doba přenosu

– je úměrná objemu vyměňované paměti

• Princip používaný v mnoha OS
– UNIX, Linux, Windows

• Základ virtualizace

Dynamické přidělování více sekcí

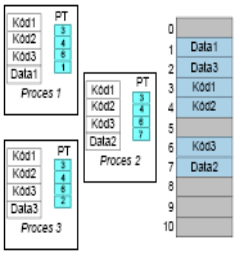
• Díra – blok dostupné paměti
– Bloky jsou roztroušeny po FAP
– Procesu se přiděluje díra, která jeho požadavek uspokojí
– Evidenci o sekcích udržuje OS
– Kde přidělit oblast délky *n*, když je volná paměť rozmístěna ve více souvislých nesousedních sekcích?
• First fit – první volná oblast dostatečné velikosti – rychlé, nejčastější
• Best fit – nejmenší volná oblast dostatečné velikosti – neplýtvá velkými děrami
• Worst fit – největší volná oblast – zanechává velké volné díry

Fragmentace

• Externí Fragmentace – celkové množství volné paměti je dostatečné, aby uspokojilo požadavek procesu, avšak prostor není souvislý, a nelze ho tudíž přidělit
• Interní Fragmentace – přidělená paměť je o málo větší než potřebná, avšak zbytek není využíván
• Redukce externí fragmentace pomocí setřásání
– Přesouvají se obsahy paměti s cílem vytvořit (jeden) velký souvislý volný blok

Stránkování

• LAP procesu není zobrazován jako jediná souvislá sekce FAP, zobrazuje se do po částech do volných sekcí FAP
• FAP se dělí na sekce zvané rámce
– pevná délka, délka v násobcích mocnin 2 (512 až 8 192 B)
• LAP se dělí na sekce zvané stránky
– pevná délka, shodná s délkou rámců
• Udržuje se seznam volných rámců
• Program délky *n* stránek se umístí (zavede) do *n* rámců
• Překlad logická adresa → fyzická adresa
– Tabulkou nastavovanou z OS, interpretovanou MMU
– DAT = Dynamic Address Translation
– PT = Page Table, tabulka stránek- uložena v operační paměti
• Existuje vnitřní fragmentace (stránky nejsou zcela zaplněny)
• Ochrana paměti se implementuje připojením ochranných bitů pro každou stránku v tabulce stránek(r-povoleno čtení, w-povolen zápis, x-povolena interpretace instrukcí)
•Sdílení stránek



- Sdílený kód
– Jediná *read-only* kopie (reentrantního) kódu ve FAP sdílená více procesy (editor, shell, oknový systém)
– Sdílený prostor se musí ve všech procesech nacházet na stejných logických adresách
- Privátní kód a data
– Každý proces si udržuje svoji vlastní kopii kódu a dat
– Stránky s privátním kódem a daty mohou být kdekoliv v LAP

•Stránkování na žádost- stránka se zavádí do FAP jen, když je potřebná
• Výhody

– Méně I/O operací
– Menší požadavky na paměť
– Rychlejší reakce
– Větší stupeň paralelismu

• Kdy je stránka potřebná?

-Byla-li referencována (odkazována)

• Reference

– legální reference

• Stránka se nachází ve FAP a logická adresa se přeložila na fyzickou

– nelegální reference (narušení ochran)

• Výjimka detekovaná mechanismem ochrany paměti – OS zruší proces

– reference stránky, která není ve FAP

• Výjimka typu **výpadek stránky** – OS zavede stránku do FAP

• Kdy je stránku zavádět?

-dvě politiky

• Vlastní stránkování na žádost (*Demand paging*)

- Stránka se zavádí jako důsledek přerušení typu **výpadek stránky**

-Na počátku běhu procesu se tvoří série výpadků stránek

•Předstránkování (*Prepaging*)

- Sousední stránky v LAP obvykle sousedí i na sekundární paměti
- Platí princip lokality – proces bude pravděpodobně brzy odkazovat blízkou stránku v LAP
- Zavádí se proto najednou více stránek
- Výhodné zejména při inicializaci procesu

•Nahrazování stránek

- Politika nahrazování, také politika výběru oběti
- Uplatňuje se, pokud není volný rámec ve FAP

• Typicky v okamžiku zvýšení stupně paralelismu (vzniká nový proces)

- Kterou stránku „obětovat“ a „vyhodit“ z FAP?

•Používá se celá řada algoritmů

-FIFO(First in first out)

-LRU(Last recently used)-obětuje se stránka která nebyla nejdelší dobu referencována

-Čítací algoritmy- založeny na čítání přístupů do stránky

Principy virtuální paměti

• Virtuální paměť

– Separace LAP a FAP
– Ve FAP se může nacházet pouze potřebné části programů pro bezprostřední řízení procesů
– LAP může být větší než FAP
– Adresní prostory lze sdílet
– Lze efektivně ji vytvářet procesy

• Techniky implementace

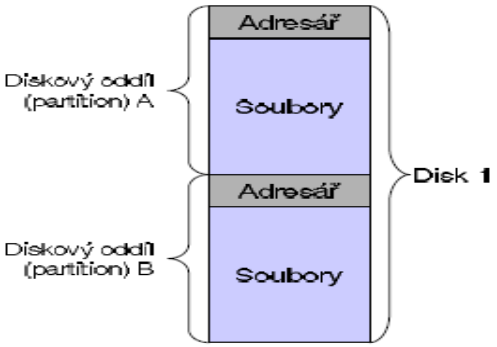
– stránkování na žádost, *Demand Paging*
– segmentace na žádost, *Demand Segmentation*

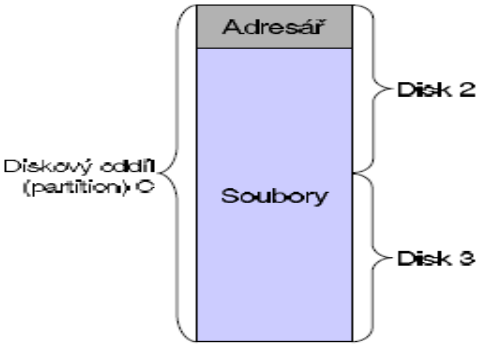
• Častá synonyma

– virtuální paměť – LAP
– reálná paměť – FAP

Systém správy souborů

Organizace systému souborů





- Jeden disk je rozdělen na více logických oddílů a na každém z nich je samostatně organizovaný systém souborů
- Jeden diskový oddíl pokrývá více fyzických disků a systém souborů je vytvořen na tomto logickém oddílu

Adresáře

- Adresář
  - Množina datových položek uchovávajících informace o souborech uložených na diskovém oddílu
  - Dvě pojetí pojmu „adresář“
    1) adresář souborového systému na diskovém oddílu (nemusí obsahovat jména souborů)
    2) uživatelsky dostupná struktura se jmény souborů a odkazy do 1)
  - Položky adresářů obsahují atributy souborů
- Operace s adresáři
  - Vyhledání souboru, poskytnutí seznamu souborů
  - Vytvoření, zrušení či přejmenování souboru
  - Procházení souborovým systémem (hierarchií adresářů)
- Logická organizace adresářů – cíle
  - efektivita – soubor je třeba najít rychle
  - nezávislé pojmenovávání souborů
    - 2 uživatelé mohou dát různým souborům totéž jméno
    - 2 uživatelé mohou pojmenovat týž (sdílený) soubor různými jmény
  - snadné seskupování dle nějaké logické příbuznosti
  - struktury: stromy, acyklické grafy, B-stromy

Ochrana souborů

- Požadavek víceuživatelských systémů
- Volitelné řízení přístupu (Discretionary Access Control) – DAC
  - Vlastník souboru (kdo ho vytvořil) má možnost určovat kdo co smí se souborem dělat
  - Typy přístupu read, write, execute, append, delete, ...
  - UNIXy – bity read, write, execute; user, group, other
    - rwx rwx rwx
      u g o
- Povinné řízení přístupu (Mandatory Access Control) – MAC
  - Možnosti práce se souborem určuje systémová politika řízení přístupu –pravidla (součást bezpečnostní politiky OS)
  - Uživatelé systému nemají zpravidla možnost pravidla měnit
    - může jen správce systému
  - Windows na NTFS

Implementace souborových systémů

- Systém souborů jako součást operačního systému bývá vrstven
  - I/O Control: drivery, správa přerušení
  - Basic File System: čtení/zápis fyzických bloků z/na disk
  - File Organization Module: správa (volné) paměti na disku
  - Logical File System (LFS): správa metadat (organizace souboru, File Control Block – FCB, adresáře souborů, ochrana, bezpečnost)
- FCB – řídicí struktura pro práci se souborem
  - Vytvoření souboru
    - Aplikace volá LFS, který vytvoří nový FCB, na disku opraví adresář a uloží nový FCB
  - Otevření souboru
    - LFS najde záznam o souboru na disku a jeho FSB zavede do paměti
  - LFS udržuje FCB otevřeného souboru v paměti

Virtualizace souborového systému

- Cíle virtuálního souborového systému (VFS)
  - možnost používat jednotné rozhraní systémových volání (API) i pro odlišné typy souborových systémů
  - API se vytváří spíše jako API k rozhraní VFS než jako rozhraní ke konkrétnímu systémů souborů
- Proč více systémů souborů?
  - jiný pro pevné disky
  - jiný pro diskety
  - jiný pro CD, DVD, ...
  - interoperabilita různých OS

Přidělování diskového prostoru

- Přidělování alokačních bloků souborům
- Přidělování souvislých diskových prostoru
  - každý soubor zabírá množinu sousedních bloků disku
  - varianta – Extent-Based File Systems – souborům se přiděluje vždy několik souvislých úseků, tvořených několika diskovými bloky – *extents*
  - soubor je tvořen jedním nebo více „extenty"
  - Výhody
    - Malé pohyby diskových hlav
      - rychlé
    - Jednoduchá evidence – jen začátek a počet bloků
    - Sekvenční i přímý přístup
  - Nevýhody
    - Špatné využití diskového prostoru
      - hledání volného prostoru (BEST-FIT, FIRST-FIT, ...)
    - Soubory nemohou růst (obtížné připisování)
    - Nutnost „setřásání"
- Vázané přidělování prostoru(mapa disku, File Allocation Table)
  - Soubor je vázaným seznamem diskových bloků
  - Bloky mohou být rozptýleny po disku libovolně
  - Mapa disku – tabulka FAT je umístěna mimo vlastní oblast souborů na disku
  - První blok souboru je odkazován z adresáře
  - Další bloky jsou pak ve formě „rozptýlené tabulky" uvedeny ve FAT
  - Výhody
    - Jednoduché – stačí znát jen počáteční adresu

- Není nutno udávat velikost souboru při jeho vytváření
- Vhodné zejména pro sekvenční přístup– snadné připisování
- Nevzniká „externí“ fragmentace– netřeba setřásat
- Používáno pro správu volné paměťi – řetězený list volných bloků

– Nevýhody
- Problém s velikostí tabulky
– velký disk – mnoho bloků nebo velké bloky s malým využitím

• Indexované přidělování prostoru
– Ukazatelé bloků přidělených souboru jsou seskupeny ve společném (*indexovém*) bloku, v tabulce indexů
– Položka adresáře odkazuje na blok obsahující index – seznam bloků
– Vhodné pro sekvenční i přímý přístup
– Indexní blok se při otevření souboru nahraje do operační paměti
– Indexy možno organizovat hierarchicky (Unix FS – UFS)

Systém souborů Windows NTFS

• Základní strukturou je svazek (*volume*)
– Analogie *partition*
– Na discích jsou svazky formátovány pomocí *disk administrator utility*
– Svazek může být vytvořen na části disku, na celém disku nebo se může prostírat přes více disků

• Vše je popsáno jako tzv. metadata
– všechna metadata, vč. např. informace o svazku, jsou ukládána na disku jako soubory

• Struktura disku
– ID sektor – Boot sektor
– tabulka MFT
- Hlavní tabulka souborů, definice obsahu svazku
– Relační databáze
• řádky (záznamy) – soubory, sloupce – atributy
– záznamy MFT – definice souborů na NTFS svazku
– komponenty záznamu MFT:
• časová značka, čítač násobných vazeb, jméno souboru /adresáře, seznam externích alokačních bloků, bezpečnostní deskriptor (vlastník, kdo smí sdílet), data nebo index na data, bitová mapa použitých záznamů v MFT nebo v adresáři, ...
– v MFT jsou záznamy s ukazateli na alokační bloky, které se nevešly do MFT struktury

– ostatní systémové soubory
- Systémové soubory
– MFT a jeho záložní kopie
– protokol: seznam akcí pro obnovu (*recovery*), změn adresářů, vytvoření souboru, ...
– soubor se jménem svazku
– soubor s definiční tabulkou atributů
– soubor s indexem na kořenový adresář
– soubor s bitovou mapou volných a přidělených alokačních bloků
– soubor s definicí vadných sektorů

– oblast uživatelských adresářů a dat