Operační systémy 2

Správa operační paměti

Petr Krajča

Katedra informatiky
Univerzita Palackého v Olomouci

Organizační informace

- email: petr.krajca@upol.cz
- konzultační hodiny
  - pátek 11:30 – 13:00, úterý 14:15 – 15:15
  - jiný termín po dohodě emailem, konzultace emailem
- www: `http://phoenix.inf.upol.cz/~krajca/`
- slidy
- rozšiřující texty a praktické úkoly (nepovinné, ale užitečné)
- **není možné pořizovat obrazové a zvukové záznamy přednášek bez předchozího souhlasu vyučujícího**

Literatura

- Keprt A. *Operační systémy. 2008*
- Silberschatz A., Galvin P.B., Gagne G. *Operating System Concepts, 7th Edition.* John Wiley & sons, 2005. ISBN 0-471-69466-5.
- Tanenbaum A.S. *Modern Operating Systems, 2nd ed.* Prentice-Hall, 2001. ISBN 0-13-031358-0.
- Stallings, W. *Operating System Internals and Design Principles, Fifth Edition.* Prentice Hall, 2004. ISBN 0-13-127837-1.
- Bovet D., Cesati M. *Understanding Linux Kernel, 3rd ed.* O'Reilly, 2006. ISBN 0596005652.
- Solomon D.A., Russinovich M. E. *Windows Internals: Covering Windows Server 2008 and Windows Vista.* Microsoft Press, 2009. ISBN 0735625301.
- Jelínek L. *Jádro systému Linux: kompletní průvodce programátora.* Brno, Computer Press, 2008.

Operační pameť

- zásadní část počítače
- uložení kódu a dat běžících procesů i operačního systému
- přístup k zařízením (DMA)
- virtuální pamět umožňuje např. přístup k souborovému systému
- z HW pohledu může být operační paměť realizovaná různými způsoby (DRAM, SRAM, (EEP)ROM, ale i HDD, SSD, flash paměti)
- různé rychlosti
- přístup CPU k paměti nemusí být přímočarý (L1, L2 a L3 cache)
- pro jednoduchost budeme HW stránku věci zanedbávat
- Ulrich Drepper: What every programmer should know about memory (http://lwn.net/Articles/250967/)

Požadavky na správu paměti (Memory Managment)

- evidence prostoru volného a přiděleného procesům
- přidělování a uvolňování paměti procesů
- přesunutí (přiděleného prostoru) – program by neměl být závislý na místě, kde se v paměti nachází (nutné k umožnění swapování)
- ochrana (přiděleného prostoru) – jednotlivé procesy by měly být izolovány
- sdílení – pokud je to žádoucí, mělo by být možné sdílet některé části paměti mezi procesy ($2\times$ spuštěný stejný program)
- logická organizace – paměť počítače (spojitý prostor, ,,sekvence bytů") vs. typický program skládající se z modulů (navíc některých jen pro čtení nebo ke spoustění)
- fyzická organizace – paměť může mít více částí/úrovní (RAM, disk); program jako takový se nemusí vlézt do dostupné paměti RAM

Adresní prostory

- v současných OS existuje několik různých nezávislých *adresních prostorů*, způsobů číslování paměťových buněk
- každý proces by měl mít vlastní paměťový prostor (izolace)
- různá zařízení mají odlišné adresní prostory
- ve spolupráci s hardwarem dochází k mapování fyzické paměti do adresního prostoru procesu
- např. grafická paměť může být namapována (např. od adresy 0xD0000000) do adresního prostoru procesu (zápis do paměti procesu $\rightarrow$ zápis do grafické paměti)

Správa paměti v jednoúlohových operačních systémech

- na správu paměti nejsou kladeny výraznější nároky
- v jeden okamžik může běžet jenom jeden proces (MS-DOS, CP/M, jednoúčelová zařízení)
- (časté řešení) operační systém a ovladače zařízení se nachází na začátku nebo na konci paměti, program pak někde ve zbývajícím prostoru

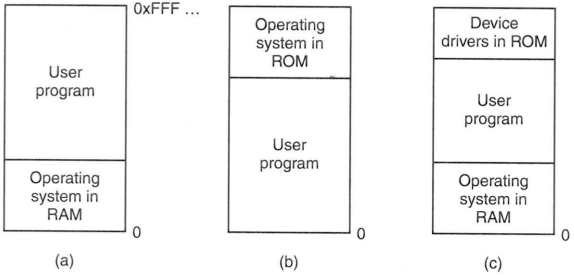


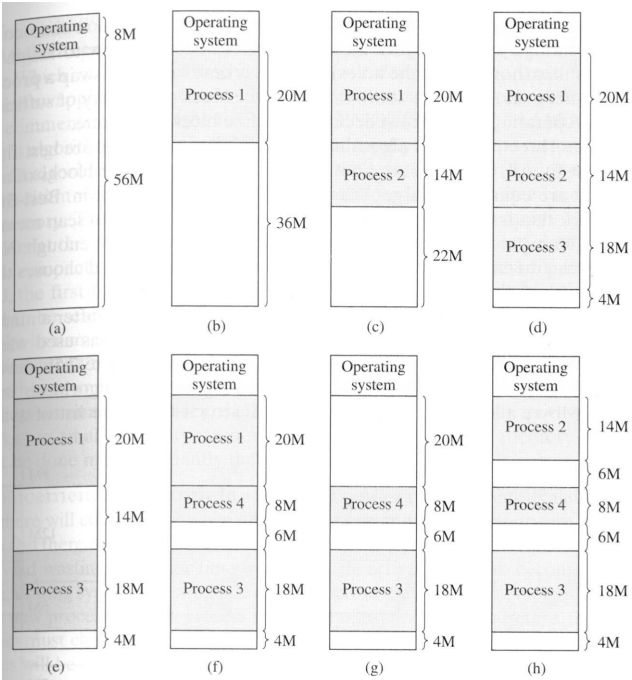
Figure 4-1. Three simple ways of organizing memory with an operating system and one user process. Other possibilities also exist.

Rozdělení paměti na souvislé bloky pevné velikosti

- předpokládejme, že OS je umístěn v nějaké části paměti
- nejjednodušší přístup je rozdělit pamět na souvislé bloky stejné velikosit (např. 8MB)
- pokud proces potřebuje víc paměti, musí se sám postarat o odsun/načtení dat do sekundární paměti
- pokud proces potřebuje míň paměti dochází k neefektivnímu využití paměti – tzv. **vnitřní fragmentaci**
- výhodou je, že lze velice jednoduše vybrat umístění, kam načíst proces
- problém přesunutí $\rightarrow$ relativní adresování
- v IBM OS/360 (historická záležitost)
- vylepšení: rozdělit paměť na bloky různých velikostí (např. 2, 4, 6, 8, 12 a 16 MB)
- jedna vs. více front (každá fronta pro procesy, které se vlezou do dané paměti)

Přidělování paměti po blocích proměnlivé velikosti (1/2)

- každý proces dostane k dispozici tolik paměti, kolik potřebuje
- sníží se tak míra vnitřní fragmentace
- po čase dochází k **vnější fragmentaci** (paměť je volná, ale je rozkouskovaná, i.e., je problém přidělit větší blok)

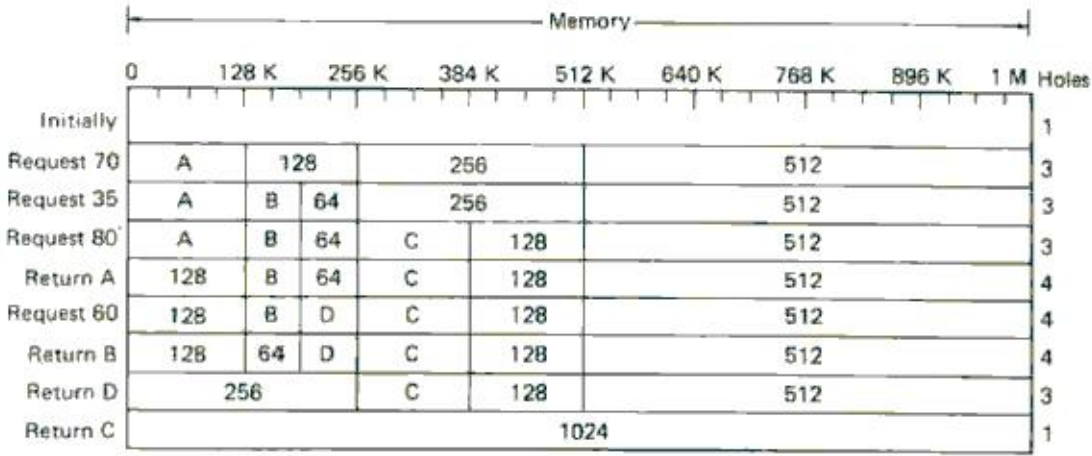

Operating system
8M
56M
(a)
Operating system
Process 1
20M
36M
(b)
Operating system
Process 1
20M
Process 2
14M
22M
(c)
Operating system
Process 1
20M
Process 2
14M
Process 3
18M
4M
(d)
Operating system
Process 1
20M
14M
Process 3
18M
4M
(e)
Operating system
Process 1
20M
Process 4
8M
6M
Process 3
18M
4M
(f)
Operating system
20M
Process 4
8M
6M
Process 3
18M
4M
(g)
Operating system
Process 2
14M
6M
Process 4
8M
6M
Process 3
18M
4M
(h)

Přidělování paměti po blocích proměnlivé velikosti (2/2)

- strategie přiřazování paměti
  - **first fit** – začne prohledávat paměť od začátku a vezme první vhodný blok
  - **next fit** – začne prohledávat paměť od místa kam se podařilo umístit blok naposledy
  - **best fit** – přiřadí nejmenší vyhovující blok (v paměti jsou nevyužité bloky, ale jsou co nejmenší)
  - **worst fit** – vezme největší blok (v paměti nejsou malé nevyužité bloky, ale nelze spustit větší program)
- informace o volném místu lze ukládat do *bitmap* nebo *seznamů volného* místa
- reálné uplatnění při správě paměti (malloc/free)
- vylepšení: zahuštění (compaction) – procesy jsou v paměti přesunuty tak, že vznikne souvislý blok volného místa (časově náročná operace)

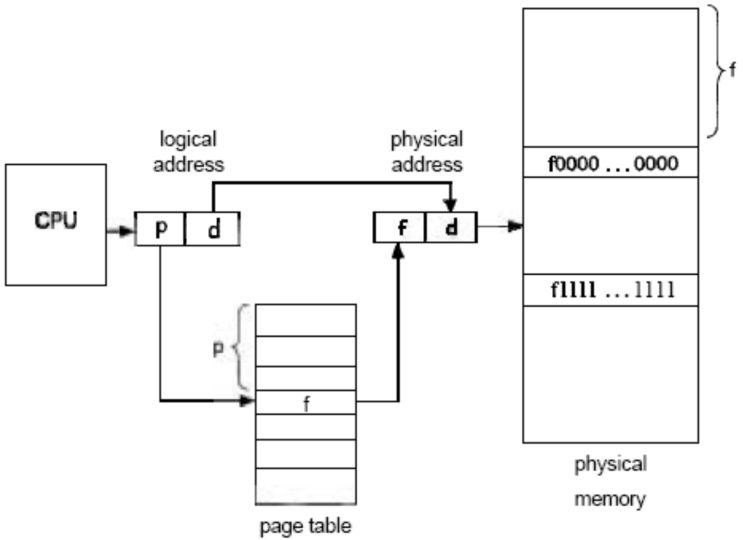
Buddy alokace paměti

- kompromisní řešení
- přiděluje pamět po blocích velikost 2^K, kde $L \leq K \leq U$, kde 2^L je nejmenší blok, který chceme přiřadit a 2^U je největší možný blok (celá paměť)
- algoritmus alokace paměti velikosti s
  1. najdi blok velikosti $2^{k-1} < s \leq 2^k$, příp. pokud je $s < 2^L$, blok velikosti 2^L
  2. pokud takový není, rozděl nejmenší blok větší než 2^k na půl
  3. přejdi na krok jedna
- uvolnění paměti
  1. uvolni blok paměti
  2. je jeho soused volný?
  3. pokud ano, sluč je dohromady, pokračuj krokem 2
- snižuje míru vnější fragmentace, má střední míru vnitřní fragmentace
- spojení volný bloků je rychlé


Memory
0
128 K
256 K
384 K
512 K
640 K
768 K
896 K
1 M
Holes
Initially
1
Request 70
A
128
256
512
3
Request 35
A
B
64
256
512
3
Request 80
A
B
64
C
128
512
3
Return A
128
B
64
C
128
512
4
Request 60
128
B
D
C
128
512
4
Return B
128
64
D
C
128
512
4
Return D
256
C
128
512
3
Return C
1024
1

Stránkování

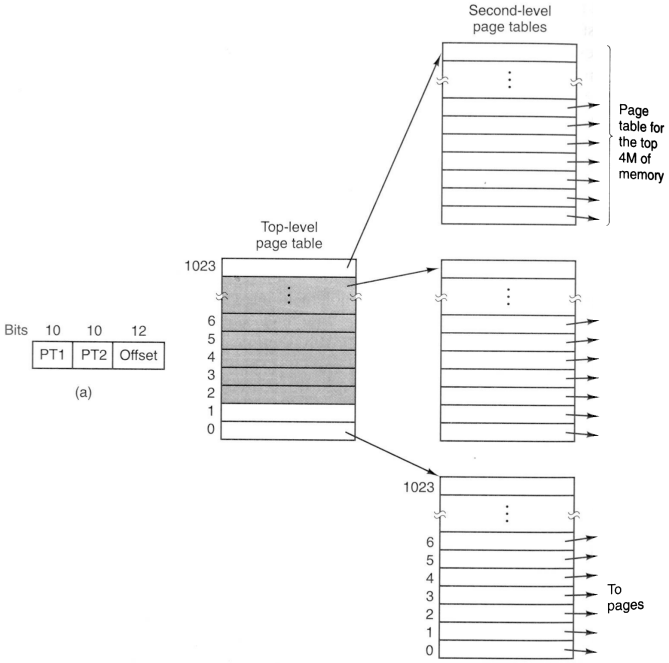
- adresní (**logický**) prostor procesu je rozdělen na menší úseky – **stránky** (pages)
- **fyzická paměť** je rozdělena na úseky stejné delky – **rámce** (frames, page frames)
- provádí se mapování **logických adres** $\rightarrow$ **na fyzické**
- procesy už nemusí být umístěny v souvislých blocích paměti
- výpočet fyzické adresy musí být implementovaný hardwarově (efektivita)
- CPU společně s OS si udržuje *stránkovací tabulku*
- logická adresa má dvě části: pd, kde p je číslo stránky a d je offset
- fyzicka adresa vznikne jako fd, kde f je číslo rámce v tabulce stránek příslušného stránce p



Obrázek 14: Základní model stránkování. [SGG05]

Adresář stránek

- velikost stránek je mocnina 2 (typicky 4 kB)
- jednoduchý přesun na disk
- uvažujeme-li velikost stránky 4 kB a velikost adresního prostoru 32 bitů $\Longrightarrow$ velikost adresní tabulky je 1 milion záznamů
- nepraktické udržovat tak velkou tabulku (obzvlášť pro každý proces)
- používá se víceúrovňová tabulka
- část logické adresy udává tabulku, další část index v tabulce a další část offset
- např. pro 4 kB stránky a 32bitové adresy může být toto rozložení 10-10-12 bitů
- tzn. systém k adresaci používá 1024 tabulek po 1024 záznamech
- nevyužité tabulky mohou být prázdné
- v praxi se používají i tří a čtyřúrovňové tabulky
- ke stránkám jsou asociovány dodatečné informace (NX bit, modified bit)


Second-level
page tables
Page
table for
the top
4M of
memory
Top-level
page table
1023
6
5
4
3
2
1
0
Bits 10 10 12
PT1 PT2 Offset
(a)
1023
6
5
4
3
2
1
0
To
pages

TLB: Translation Lookaside Buffer

- cache procesoru obsahující hodně používané části stránkovacích tabulek
- pro danou stránku uchovává adresu rámce
- pokud je adresa v cache (cache hit), je hodnota vrácena velice rychle (cca 1 hodinový cyklus)
- pokud hodnota není v cache (cache miss) načtení adresy trvá delší dobu (10-100 cyklů), typický miss rate $< 1\% \rightarrow$ průměrný přístup k zjištění rámce je ~ 1.5 hodinového cyklu
- princip lokality – je vhodné programovat tak, aby data ke kterým se přistupuje, byly na jednom místě (lokální proměnné na zásobníku)

Sdílená paměť

- může být užitečné sdílet paměť (komunikace, úspora místa)
- dva stejné programy/knihovny v paměti
- **stránky více procesů** jsou navázány na **jeden rámec**

Copy-on-Write (COW)

- speciální případ
- daná stránka má nastavený příznak CoW
- dojde-li k pokusu o zápis, vznikne výjimka a procesu je vytvořena kopie rámce
- bude-li na rámec s příznakem CoW odkazovat jenom jedna stránka, příznak se odstraní
- `fork()` – vytvoří identickou kopii procesu; data jsou sice izolovaná, ale je možné sdílet kód
- úspora místa; úspora strojového času (není potřeba vytvářet kopie stránek/rámců); nízká penalizace, pokud stránky přepíšeme
- virtuální paměť umožňuje další optimalizace (mapování souborů do paměti, etc.)

Segmentace

- paměť je rozdělena na několik segmentů (např. kód, data, zásobník)
- speciální případ přidělování souvislých bloků paměti
- stránkování je obvykle pro programátora transparentní (nerozeznatelné)
- naproti tomu segmentace umožňuje rozdělit program do logických celků (kód, data)
- při použití segmentace a stránkování programy nepracují přímo s lineární adresou
- používají adresu ve tvaru segment + offset a ta se až poté převádí na fyzickou adresu pomocí stránkování
- ochrana paměti přes záčatek a délku segmentu + oprávnění