Správa hlavní paměti, metody přidělování paměti, virtuální paměť

Z FAV wiki

Ideál programátora

- Pameť nekonecně velká, rychlá, levná
- Zároveň persistentní (uchovává obsah po vypnutí)
- Bohužel neexistuje

Reálný počítač - hierarchie paměti ("pyramida")

1. Registry CPU
2. Malé množství rychlé cache paměti
3. Stovky MB až gigabajty RAM paměti
4. GB na pomalých, levných, persistentních discích

Modul pro spravu pameti

- informace o prideleni pameti
- ktera cast je volna pridelena (a kteremu procesu)
- pridelovani pameti na zadost
- uvolneni pameti, zarazeni k volne pameti
- odebira pamet' procesum
- ochrana pameti
  - pristup k pameti jineho procesu
  - pristup k pameti OS

Funkce MMU (memory managment unit)

- Dostava adresu od CPU, prevadi na adresu do fyzicke pameti
- Nejprve zkontroluje, zda adresa neni vetsi nez limit Ano - vyjimka, Ne - k adrese pricte bazi
- Pokud baze 1000, limit 60
  - Adresa 55 - ok, vysledek 1055
  - Adresa 66 - není ok, vyjimka

Tri varianty rozdeleni pameti

- OS ve spodni casti adresniho prostoru v RAM (minipocitace)
- OS v horni casti adresniho prostoru v ROM (zapouzdrene 2. systemy)
- OS v RAM, ovladace v ROM (PC { MS DOS v RAM, BIOS v 3. ROM)

Část OS, která spravuje paměť, se nazývá správce paměti

- Udržuje informaci, které části paměti se používají a které jsou volné
- Alokuje paměť procesům podle potřeby
- Zařazuje paměť do volné paměti po uvolnění procesem

- Jednoprogramové systémy bez odkládání a stránkování
  - Nejjednodušší - spouštíme pouze jeden program v jednom čase
  - Dovoluje procesu použít veškerou paměť, kterou nepotřebuje OS
  - Po skončení procesu je možné spustit další proces
- Multiprogramování s pevným přidělením paměti
- Nejjednodušší schéma = rozdělit paměť na n oblastí (mohou být i různé velikosti)
  - V historických systémech se provádělo ručně při startu zdroje
  - Po načtení úlohy je obvykle část oblasti nevyužitá
  - Snaha umístit úlohu do nejmenší oblasti, do které se vejde
- Multiprogramování s proměnnou velikostí oblasti
- Každé úloze je přidělena paměť podle požadavku
- Obsazení paměti se postupně mění, jak úlohy přicházejí a končí
- Zlepšuje využití paměti
- Postupem času může vzniknout mnoho malých volných oblastí
- OS musí vědět, která paměť je volná a která alokovaná

Obsah

Nejpoužívanější způsoby správy paměti

níže

Správa paměti pomocí bitových map

- Paměť rozdělena do alokačních jednotek stejné délky
- S každou alokační jednotkou sdružen jeden bit (0 = volno, 1 = obsazeno), tyto bity jsou pohromadě v bitové mapě
  - menší alokační jednotky = větší bitmapa
  - Větší alokační jednotky = více nevyužité paměti, protože velikost procesu nebude přesně násobek alokační jednotky
- Výhoda - konstantní velikost bitmapy
- Nevýhoda - pokud požadujeme úsek paměti velikosti N alokačních jednotek, musí se v bitmapě vyhledat N po sobě následujících nulových bitů (drahá operace)

Správa paměti pomocí seznamů

- Myšlenka udržovat seznam alokovaných a volných oblastí
- Každá položka seznamu obsahuje:
  - Informaci, zda se jedná o proces nebo díru (P nebo H)
  - Počáteční adresu oblasti
  - Délku oblasti
- Práce se seznamem:
  - Pokud proces skončí, nahradí se dírou
  - Pokud jsou vedle sebe dvě díry, sloučí se
- Seznam je dobré mít seřazený podle počáteční adresy oblasti
- zároveň je seznam obousměrný, takže se snadno vrací

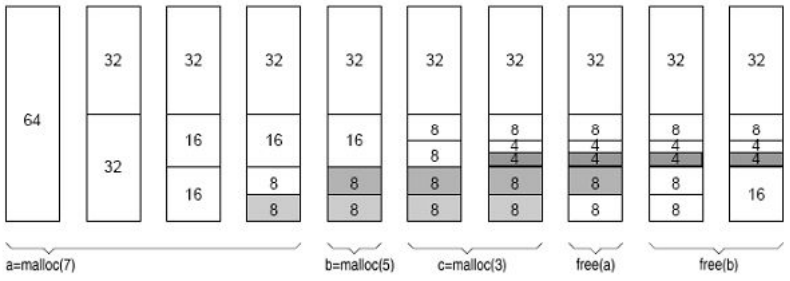
Možnosti alokace:

- Algoritmus first fit
  - Prohledávání seznamu, dokud se nenajde dostatečně velká díra
  - Rychlý
  - díra se rozdělí na proces a zbytek díry
- Algoritmus next fit
  - Prohledávání začne tam, kde skončilo předchozí
  - o trochu pomalejší než first-fit
- Algoritmus best fit
  - Prohledá celý seznam, vezme nejmenší díru, do které se proces vejde
  - Pomalejší, protože prohledává celý seznam
  - zase ale najde nejlepší místo (byť na druhou stranu pak zbyte spousta malých nepoužitelných děr)

Možná vylepšení

- oddělěné seznamy děr a perocesů (rychlejší alokace, pomalejší dealokace
- seznam děr seřazený podle velikosti (rychlé best fit, opět náročná dealokace)
- místo samostatného seznamu děr lze využít samotné díry, které na sebe odkazují - úspora paměti

Mechanismus „buddy system“



- Mějme seznamy volných bloků velikosti 1,2,4,8,16 … alokačních jednotek až do seznamu bloků velikosti celé paměti
- Na začátku veškerá paměť volná, všechny seznamy jsou prázdné kromě seznamu obsahující 1 položku velikosti paměti
- Přijde-li požadavek, zaokrouhlí se nahoru na mocninu 2
- Blok se rozdělí na 2 bloky, pokud ještě moc velké, jeden z nich se zase rozdělí na 2 bloky atd.
- Uvolnění paměti: pokud jsou volné oba sousední bloky stejné velikosti, spojí se do jednoho
- Neefektivní ve využití paměti, ale rychlý

Příklady použití:

- Buddy system - jádro Linuxu, běží ve fyzické paměti
- First fit, Next fit - malloc v C

- Bitová mapa - SWAP v Linuxu

Virtuální paměť

- Problém, že programy jsou větší než dostupná fyzická paměť
- Chceme, aby ve skutečné paměti byla realizovaná pouze část adresového prostoru, zbytek může být odložen na disku
- Procesor používá tzv. virtuální adresy
- Pokud je požadovaná část virtuálního paměťového prostoru ve fyzické paměti, MMU převede virtuální adresu na fyzickou
- Pokud není ve fyzické paměti, OS jí musí přečíst z disku
- Většina systémů virtuální paměti používá techniku nazývanou stránkování
