DRUGI DEO: Upravljanje memorijom

- Fizička memorija
- Virtuelna memorija
- Deljenje podataka i koda u glavnoj memoriji

Fizička memorija

- Brzina operativne memorije je znatno manja od brzine procesora
- Razlika u brzini nadoknađuje se brzim (i skupim) keš memorijama *(cache memory)*
- Performanse celog sistema zavise od efikasnog upravljanja memorijom

Priprema za izvršenje

- Program se piše u izvornom kodu *(source code – source modules)*
- Izvorni kod je nerazumljiv računaru
- Pre izvršenja mora se obaviti:
  - prevođenje (*compilation*)
  - povezivanje (*linking*)
  - punjenje (*loading*)

Prevođenje

- Prevođenje iz izvornog koda vrši programski prevodilac (*compiler*)
- Rezultat prevođenja je objektni kod (*object code – object modules*)
- Objektni kod jeste mašnski kod ali i dalje ne može da se izvršava
- Potrebno ga je povezati sa drugim objektnim modulima

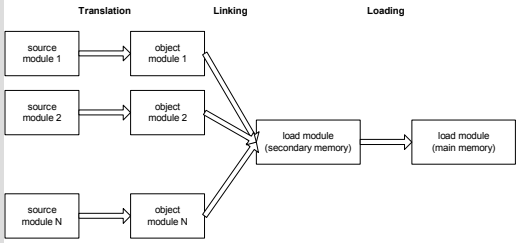
Povezivanje

- Načešće jedan objektni modul predstavlja jednu prevedenu datoteku sa izvornim kodom
- Povezivanje objektnih modula vrši povezivač (*linker*)
- Povezivanje može biti:
  - statičko (svi objektni moduli povezani u jednu datoteku)
  - dinamičko (povezivanje pojedinih modula se vrši u trenutku izvršenja)

Punjenje

- Nakon povezivanja program još uvek nije spreman za izvršenje
- Potrebno napunuti ga u memoriju i povezati logičke i fizičke adrese
- Tabela eksternih simbola (*external symbol table*) postoji za svaki objektni modul i povezuje se u vreme punjenja

Priprema za izvršenje – nastavak...



Relokacija

- Logički adresni prostor koristi programer u vreme pisanja programa
- Mapiranje između logičkog i fizičkog adresnog prostora (*binding*)
- Mapiranje može biti:
  - statičko
  - dinamičko
- Relokatibilni programi – u vreme izvršenja mogu se naći u različitim delovima fizičke memorije

Statičko mapiranje

- Povezivanje logičkih i fizičkih adresa pre početka izvršavanja:
  - *compile-time binding*
  - *link-time binding*
  - *link-time binding*
- Retko se koristi (uglavnom moduli jezgra operativnog sistema)

Dinamičko mapiranje

- Povezivanje logičkih i fizičkih adresa u vreme izvršavanja
- Relokacioni registar (RR)

```
physical_address =
   logical_addres + RR
```


Particionisanje memorije

- Kod jednoprocesnih operativnih sistema memorija se deli između operativnog sistema i jednog procesa
- Kod višeprocesnih operativnih sistema memorija se mora particionisati
- Metoda fiksnih particija
- Metoda varijablinih particija

Fiksne particije

- Memorija se deli u fiksne particije u vreme inicijalizacije OS-a
- Nije moguća promena veličine particija
- Generišu se particije različite veličine da bi odgovarale različitim potrebama
- Svaki proces dobija jednu particiju
- Jednostavna dodela odgovarajuće particije (*best-fit* i slični algoritami)

Fiksne particije–nastavak...

- Nedostaci:
  - pojedine particije ostaju prazne ako nema nikoga da ih iskoristi
  - iako ima dosta praznih particija proces koji zahteva puno memorije ne može da ih iskoristi
  - interna fragmentacija – unutar svake particije ostaje neiskorišćen prostora

Varijabilne particije

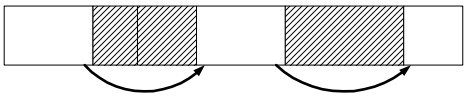
- U vreme izvršavanja procesa OS dodeljuje particiju potrebne veličine
- Nakon završetka procesa memorija se oslobađa i pojavljuju se "rupe" u memoriji
- Susedne "rupe" moraju se spajati u veće blokove inače dolazi do stalnog smanjivanja (eksterna fragmentacija)

Varijabilne particije-nastavak

- Uvek se mora znati gde se nalaze slobodni delovi memorije
- Metoda povezanih listi
- Metoda bitmapa

Povezane liste

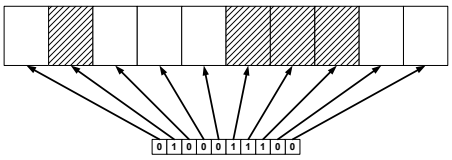
- Svaki prazan blok ima veličinu i pokazivač na sledeći prazan blok
- Praćenjem pokazivača mogu se pretražiati svi prazni blokovi



Bitmape

- Bitmapa je niz nula i jedinica
- Nula označava prazan blok a jedinica zauzet blok
- Blok je fiksne veličine, ali dovoljno mali da ne dolazi do interne fragmentacije
- Broj uzastopnih nula prikazuje veličinu slobodnog bloka

Bitmape – nastavak...



Strategije alokacije

- Kako izabrati optimalnu slobodnu particiju?
- Kada se pronađe slobodna particija odvaja se potreban deo, a preostali deo ostaje slobodan
- Kriterijumi alokacije:
  - izbeći eksternu fragmentaciju
  - minimizovati vreme potrage za adekvatnom particijom

First-fit

- Pretraživanje uvek počinje od početka liste slobodnih particija
- Prva particija koja je dovoljna
- Dodeljuje se iako nije optimalna
- Preostali deo ostaje slobodan
- Minimalno vreme pretraživanja

Next-fit

- Ako se uvek počne pretraživanje od prve slobodne particije dolazi do fragmentacije u prvom delu liste slobodnih i sa vremenom postaje sporije pronaći odgovarajuću particiju
- Problem nestaje ako nastavimo sa mesta gde smo predhodno stali

Best-fit

- Bira se particija koja po veličini nabolje odgovara potrebama
- Pretraživanje je sporo
- Pojavljuje se veliki broj malih particija koje su beskorisne (velika eksterna fragmentacija)

Worst-fit

- Bira se particija koja po veličini najgore odgovara potrebama
- Eliminiše se pojava malih beskorisnih particija
- Pretraživanje je sporo
- Brzo se troše velike particije (problem za zahtevne procese)

Koja je najbolja strategija?

- Ako se uzmu u obzir kriterijumi:
  - vreme pretraživanja
  - iskorišćenje memorije

  simulacije kažu da je nabolji

FIRST-FIT

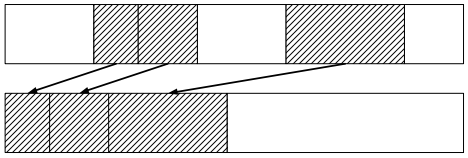
- Razlog: brz algoritam i fragmentacija samo na početku memorije

Nedostatak memorije

- Šta se dešava kada nije moguće dodeliti procesu potrebnu memoriju?
- Kompakcija (*memory compaction*)
- Zamena segmenata (*swapping*)
- Preklapanje segmenata (*overlaying*)

Kompakcija memorije

- Eliminisanje "rupa"
- Dodeljeni segmenti se preraspoređuju tako da se pojave veliki slobodni segmenti
- Troši se procesorsko vreme

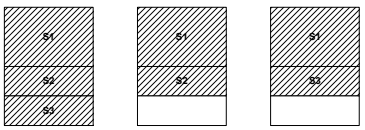


Swapping

- Segment jednog procesa se privremeno sklanja na spoljnu memoriju i segment se oslobađa za drugi proces
- Deo diska predviđen za ovu namenu naziva se *swap space*
- Segment se vraća u glavnu memoriju kada to bude ponovo potrebno
- Dolazi do degradacije performansi

Preklapanje segmenata

- Programer deli program i podatke u više nezavisnih segmenata i specificira koji segmenti istovremeno moraju biti učitani



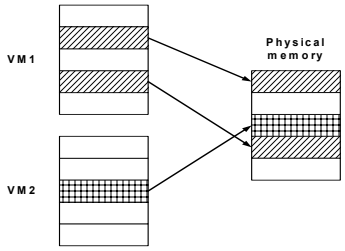
Virtuelna memorija

- Glavna memorija se deli između više procesa i više korisnika
- Veličina memorije je još uvek nedovoljna za dosta proces (multimedija, baze podataka...)
- Virtuelna memorija rešava problem veličine glavne memorije upotrebom eksterne memorije

Principi virtuelne memorije

- Fizička memorije je "nevidljiva"
- Svaki proces ima iluziju posedovanja celokupnog adresnog prostora
- Operativni sistem obezbeđuje mehanizme mapiranja između virtuelne i fizičke memorije
- Prevođenje logičkih u fizičke adrese
- Da bi bili dostupni, podaci moraju biti u fizičkoj memoriji

Principi VM – nastavak...



Principi VM – nastavak...

- Podela VM na delove fiksne dužine:
  - stranična VM
- Podela VM na delove varijabilne dužine:
  - segmentna VM
- Podela VM na delove varijabilne dužine koji se sastoje od delova fiksne dužine:
  - segmentno-stranična VM

Implementacija VM

Moraju se rešiti sledeći problemi:

- Mehanizam mapiranja adresa
- Strategija pozicioniranja
- Strategija zamene
- Kontrola zauzeća
- Deljenje memorije

Mapiranje adresa

- Način na koji se logičke adrese transliraju u fizičke adrese
- Translacija se vrši jednom ili dva puta prilikom izvršenja svake instrukcije
- Način translacije utiče na performanse sistema
- Operativni sistem čuva tabele koje definišu translaciju logičkih u fizičke adrese

Strategija pozicioniranja

- Gde u fizičkoj memoriji pozicionirati delove virtuelne memorije
- Kod segmente organizacije se koristi jedan od algoritama iz sistema sa varijabilnim particijama *(first-fit, best-fit, next-fit, worst-fit)*
- Kod stranične organizacije je pozicioniranje uprošćeno jer se koristi prva slobodna fizička stranica

Strategija zamene

- Šta uraditi kada je potrebno u fizičku memoriju dovesti segment VM a nema dovoljno mesta?
- Kod sistema bez VM, kompletna memorija jednog procesa je izbacivana na disk u *swap* datoteku
- Kod sistema sa VM, može se izbaciti samo jedan deo, stranica ili segment

Kontrola zauzeća

- Koliki deo VM čuvati u fizičkoj memoriji?
- Ako se čuva preveliki deo, dolazi do *swapping*-a jer se memorija različitih procesa stalno premešta između *swap* u fizičke memorije
- Ako se čuva premali deo, dolazi do *swapping*-a jer se delovi memorije procesa stalno dovlače iz *swap* datoteke u fizičke memorije
- Dovlačenje po potrebi (po zahtevu) *demand paging / segmentation*
- Kada stranica nije u memoriji generiše se stranični izuzetak *(page/segment fault trap)*

Deljenje memorije

- Pojedini procesi mogu deliti iste stranice ili segmente u fizičkoj memoriji
- Češće se sreće kod segmentne organizacije VM jer je podela na segmente logička a na stranice fizička

Stranična organizacija VM

- VM se deli u stranice fiksne dužine
- Veličina stranice je eksponent broja 2
- Tipične veličine: od 1 do 16 KB
- Jedna stranica VM se mapira u jednu fizičku stranicu
- Logička adresa se sastoji od rednog broja stranice i pomeraja u okviru stranice *address(page_number, offset)*

Stranična VM – nastavak...

- Primer:
  - 32-bitne adrese
  - veličina particije: 4 KB
  - adresa: 00ffa012 (heksadecimalno)

  - adresiranje 1 KB – 10 bita
  - adresiranje 4 KB – 12 bita
  - redni broj stranice: 00ffa
  - pomeraj unutar stranice: 012

Stranična VM – nastavak...

- Veličina stranice i fizičke stranice moraju biti jednake
- Redni broj stranice mapira se u redni broj fizičke stranice
- Pomeraj je unutar stranice je isti i u VM i u fizičkoj memoriji
- Za ubrzavanje preslikavanja koriste se asocijativne memorije koje čuvaju najčešće korišćena preslikavanja (u slučaju da se preslikavanje ne nalazi u asocijativnoj memoriji, moraju se pretraživati tabele operativnog sistema)

Segmentna organizacija VM

- VM se deli na logičke segmente promenljive dužine
- Najčešći tipovi segmenata: *code, data, stack*
- Jedan proces ima jedan segment za kod, jedan segment za podatke i po jedan segment za stek za svaku od niti *(threads)*
- Logička adresa se sastoji od adrese početka segmenta i pomeraja u okviru stranice *address(segment_address, offset)*

Segmentna VM - nastavak

- Veličina segmenta i fizičkog segmenta moraju biti jednaki
- Adresa početka segmenta mapira se u adresu početka fizičkog segmenta
- Pomeraj je unutar segmenta je isti i u VM i u fizičkoj memoriji
- Za ubrzavanje preslikavanja koriste se asocijativne memorije koje čuvaju najčešće korišćena preslikavanja (u slučaju da se preslikavanje ne nalazi u asocijativnoj memoriji, moraju se pretraživati tabele operativnog sistema)

Segmentno-stranična VM

- VM se deli u segmente koji se sastoje od stranica
- Kombinacija prethodna dva metoda
- Adresa se sastoji od:
  address(segment_address, page_number, offset)
- Danas se najčešće koristi ovakva organizacija jer ima prednosti logičke podele na segmente sa jednostavnošću stranične
- Zahteva složen hardver zbog performansi

Alokacija memorije

- U segmentnim sistemima alokacija se vrši po istim algoritmima kao i kod sistema sa variajbilnim particijama
- U straničnim sistemima alokacija se vrši prostom dodelom slobodnih stranica
- Kada nestane slobodnih stranica neophodno je izvršiti zamenu

Alokacija – nastavak...

- Zamena se može vršiti:
  - globalno: kandidat za zamenu se traži među svim fizičkim stranicama bez obzira kom procesu pripadaju
  - lokalno: kandidat za zamenu se traži među svim fizičkim stranicama tekućeg procesa

Algoritmi zamene

- Optimalni algoritam zamene

Zamenjuje se stranica koja u budućnosti najduže neće biti korišćena.

- idealan ali neostvariv jer u realnim sistemima ne možemo predvideti koja stranica najduže neće biti korišćena

Algoritmi zamene-nastavak

- Slučajni algoritam zamene

Stranica koja se zamenjuje slučajno se bira.

- loše rešenje jer se obično koriste adrese koje su vremenski ili prostorno povezane

Algoritmi zamene-nastavak

- FIFO (*First In/First Out*)

Zamenjuje se stranica koja se najduže nalazi u memoriji

-loše rešenje jer se može desiti da se upravo stranica koja je najduže u memoriji i nejčešće koristi

Algoritmi zamene-nastavak

- LRU (*Least Recently Used*)

Zamenjuje se stranica koja najduže nije korišćena

- najbolje rešenje, ali zahteva vođenje statistike o korišćenju stranica