Sisteme de Operare 1 - Curs 12

Curs tinut in 2012-2013 *de catre* lector dr. Sanda-Maria Dragos

cap 10 carte

pp. 271 - 296

Gestionarea memoriei interne

Structura ierarhica de organizare a memoriei

Memoria cache

slide 2

Contine informatiile cele mai recent utilizate de catre CPU. Are capacitate relativ mica, dar timp de acces foarte rapid.

La fiecare acces, CPU verifica daca data invocata se afla in memoria cache si abia apoi solicita memoria operativa. Daca data este gasita atunci are loc schimbul intre CPU si ea. Daca nu e gasita atunci cautarea se face la nivele superioare de memorie (e.g. memoria operativa, memoria secundara, memoria de arhivare) si se aplica *principiul vecinatatii* (la aducerea unei date, se duce cu ea si un numar de locatii vecine ei, astfel incat sa umple memoria cache).

Memory bus - magistrala de date si adrese.

Memoria operativa

Contine programele si datele pentru toate procesele existente in sistem. In momentul in care un proces s-a terminat, spatiul de memorie operativa pe care l-a ocupat este eliberat si va fi ocupat de catre alte procese. Viteza de acces este foarte mare, insa mai mica ca a memoriei cache.

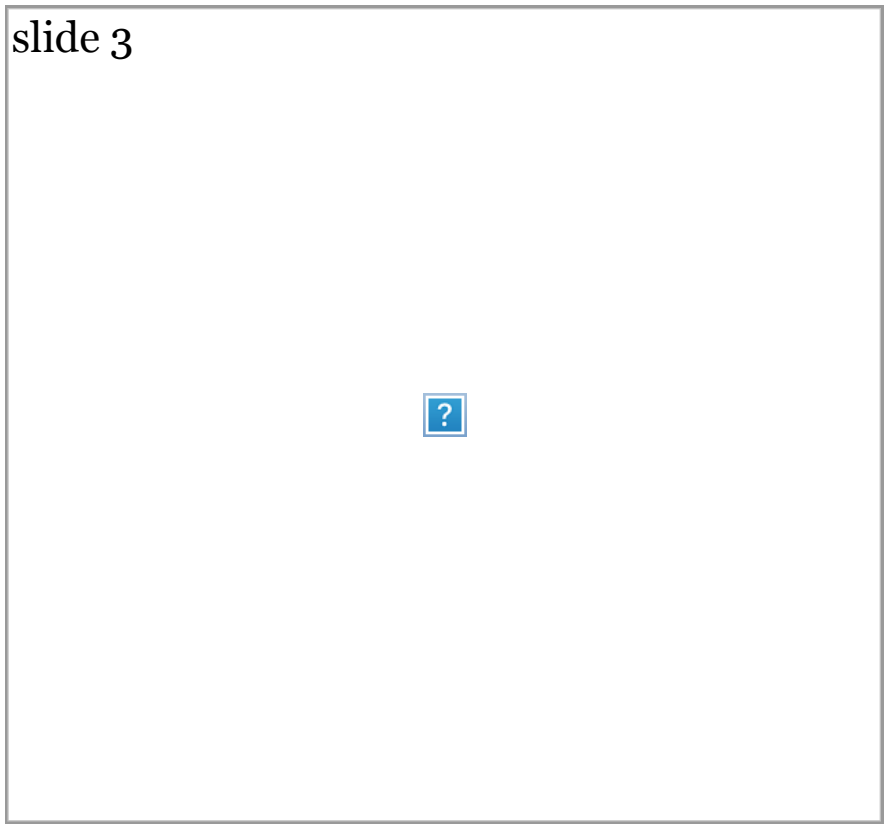
Memoria secundara

Apare la SO care detin mecanisme de memorie virtuala. Tot in cadrul memoriei secundare poate fi inclus spatiul disc de swap. E privita ca o extensie a memoriei operative. Nu e direct accesibila procesorului. Se folosesc canale de intrare/iesire pentru a trasfera datele de pe memoria secundara in memoria interna. Nu e volatila, nu se sterg datele odata cu oprirea sistemului.

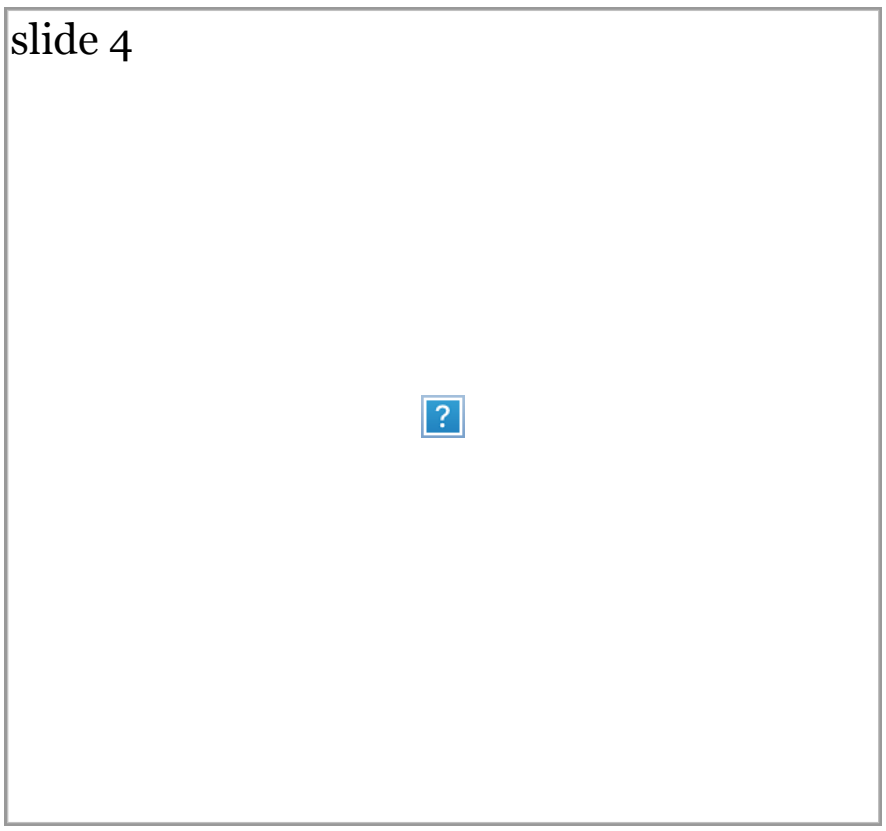
Memoria de arhivare

E gestionata de utilizator si consta din fisiere, baze de date, etc. reziente pe diferite suporturi.

slide 3



slide 4



Problematica gestiunii memoriei

Pentru a fi executat, un program are nevoie de o anumita cantitate de memorie.

Daca se lucreaza in multiprogramare, e necesar ca in memorie sa fie prezente mai multe programe.

In general, pe durata executiei unui program, necesarul de memorie variaza. Cauze:

folosirea variabilelor dinamice

se aloca spatiu intr-o zona de memorie numita heap; alocarea si eliberarea spatiului se realizeaza de catre proces prin new - dispose, malloc - mfree, constructor - destructor

segmentarea

tehnica prin care programul executabil e decupat in entitati distincte numite segmente.

Segmentele pot fi: de cod, de date sau de stiva. Programul specifica incarcarea sau reincarcarea unui segment in memorie. La construirea programului executabil se poate defini *structura de acoperire a segmentelor*.

Spatiul de memorie fiind limitat, SO trebuie sa gestioneze eficient acest spatiu. Principalele obiective ale *gestiunii memoriei* sunt:

» Calculul de translatare al adresei (relocare);
» Protectia memoriei;
» Organizarea si alocarea memoriei operative;
» Gestiunea memoriei secundare;

» Politici de schimb intre proces, memoria operativa si memoria secundara.

slide 5

Mecanismul de traslatare al adresei

Adresarea memoriei consta in realizarea unei legaturi intre un obiect al programului si adresa corespunzatoare din memoria operativa.

Fazele translatarii unui program.

» Faza de *compilare* transforma numele obiectelor program in numere reprezentand adrese relativa, adica adrese in cadrul modulului obiect.
» Faza *editarii de legaturi* (grupeaza toate modulele formand un fisier executabil) trasforma adresele din cadrul modulelor in adrese relocabile.
» Faza de *incarcare si executie* realizeaza traslatarea/relocarea adresei.

slide 6

slide 7

Tehnici de alocare a memoriei

» alocare reala
» la SO monoutilizator
» la SO multiutilizator
» cu partitii fixe (statica)
» absoluta
» relocabila
» cu partitii variabile (dinamica)

» alocare virtuala
» paginata
» segmentata
» segmentata si paginata

slide 8

Alocarea la SO monoutilizator

La SO monoutilizator e disponibil aproape tot spatiul de memorie, iar gestiunea acestuia cade in seama utilizatorului. In cazul in care memoria fizica este insuficienta, iar sistemul nu contin memorie virtuala, se aplica tehnici de *suprapunere (overlay)* pentru incarcarea unor programe mari. Aceasta metoda presupune impartirea programului in mai multe blocuri numite *overlays*. O componenta a SO va incarca *overlay*-ul dorit din memoria externa in regiunea destinatie din memoria interna pentru a fi folosit.

Alocarea cu partitii fixe (statica)

slide 9

Se mai numeste *alocarea statica* sau *Memory Fixed Tasks (MFT)*.

Presupune decuparea memoriei in zone de lungime fixa numite *partitii*. O partitie este alocata unui proces pe toata durata executiei lui, indiferent daca o ocupa complet sau nu.

Alocarea absoluta se face pentru programele pregatite de editorul de legaturi pentru a fi rulate intr-o zona de memorie prestabilita.

Mult mai folosita este **alocarea relocabila**, la care adresarea in partitie se face cu baza si deplasament. La incarcarea programului in memorie se pune in registru de baza adresa de inceput a partitiei.

In cazul sistemelor seriale cu mutiprogramare, daca un proces este plasat spre executie intr-o partitie insuficienta, el este eliminat fara a fi executat.

De obicei partitiile au lungimi diferite. Stabilirea acestor lungimi este dificila: dimensiuni prea mari elimina riscul ca unele procese sa nu poata fi executate, insa reduce numarul proceselor active din sistem.

» *fiecare partitie are coada proprie de asteptare* -
» *o singura coada pentru fiecare partitie* - SO alege prentu procesul care urmeaza a se executa in ce partitie se va incarca.

slide 10

Alocarea cu partitii variabile (dinamica)

Se mai numeste *alocarea dinamica* sau *Memory Variable Tasks (MVT)*.

Reprezita o extensie a alocarii cu paritii fixe care permite o expoatare mai supla si mai economica a memoriei: *in functie de solicitari si de capacitatea de memorie inca disponibila la un moment dat, numarul si dimensiunea partitiilor se modifica automat*.

In momentul in care un proces intra in sistem, el este plasat in partitia in care incape cea mai lunga ramura a sa. Daca sistemul functioneza timp indelungat, numarul spatiilor libere va creste iar dimensiunile lor var scadea, fenomen numit *fragmentarea interna a memoriei*. Solutii:

» Asteptarea eliberarii unei cantitati suficiente de memorie
» Alipirea unor spatii libere vecine: *colationare*
» Compactarea memoriei: *relocare* - deplasarea partitiilor active pentru a se comasa toate fragmentele de memorie neutilizate.

Operatie costisitoare

» compactare periodica: la fiecare 10 secunde
» compactarea partiala pentru a asigura spatiul doar procesului care asteapta
» mutarea numai a unora dintre procese cu colationarea spatiilor ramase libere

Intre MFT si MVT nu sunt diferente hard. MVT este realizate prin intemediul unor rutine a SO.

Mecanisme de memorie virtuala

memorie virtuala - capacitatea de a adresa un spatiu de memorie mai mare decat este cel disponibil la memoria operativa; concept aparut in 1960, Univ. Manchester, Anglia.

Toate sistemele de calcul actuale folosesc, intr-o forma sau alta, mecanisme de memorie virtuala.

Alocarea paginata

Pentru a evita fragmentarea excesiva (care apare la MVT), si drept consecinta evitarea relocarii.
Presupune:

slide 11

1. Instructiunile si datele unui program sunt impartite in zone de lungime fixa numite *pagini virtuale* care se pastreaza in memoria secundara.
2. Memoria operativa este impartita in zone de lungime fixa, numite *pagini fizice*. Lungimea unei pagini fizice este fixata prin hard. paginile virtuale si cele fizice au aceeasi lungime (putere a lui 2: ex, 1Ko, 2Ko).
3. Fiecare adresa relocabila este o pereche de forma **(p, d)**, unde **p** este numarul paginii virtuale, iar **d** este adresa in cadrul paginii.
4. Fiecare adreasa fizica este o pereche de forma **(f, d)**, unde **f** este numarul paginii fizice, iar **d** este adresa in cadrul paginii.
5. Calculul de traslatare se face prin hard ca in imaginea de mai jos

Fiecare proces are propria lui tabela de pagini, in care este trecuta adresa fizica a paginii virtuale, daca ea este prezenta in memoria operativa. La incarcarea unei noi pagini virtuale, aceasta este depusa intr-o pagina fizica libera. Paginile fizice sunt distribuie necontiguu, intre mai multe procese. *proiectarea spatiului virtual peste cel real*

slide 12

Avantajul acestui mecanism este ca foloseste mai eficient memoria operativa. Avantajul colateral este folosirea in comun a unei portiuni de cod. EX: editor de text a carui cod ocupa 2 pagini, iar utilizatorii ocupa fiecare cate o pagina pentru date proprii, putem avea urmatorul scenariu in care se consuma doar 5 pagini in loc de 9:

Alocarea segmentata

Textul unui program poate ocupa zone distincte de memorie, numite *segmente*. Deosebirea de alocarea paginata este ca segmentele sunt de lungimi diferite.

In mod analog cu adresarea paginata, o adresa virtuala este o pereche de forma (s, d), unde s este numarul segmentului, iar d este adresa in cadrul segmentului.

slide 13

Fiecare proces activ are o *tabela de de segmente*, in care fiecare intrare contine *adresa de inceput a segmentului*.

Avantaje
» *segmente reentrante* - cod pur care poate fi folosit in comun de catre mai multe procese
» *foarte buna protectie a memoriei* - fiecare segment poate primi alte drepturi de acces, depuse in tabela de segmente.

dejavantaj: e posibil sa apara fenomenul de fragmentare.

Alocarea segmentata si paginata

slide 14

Alocarea spatiului pentru fiecare segment se face paginat. Ficare proces are propria lui *tabela de segmente*. Fiecare segment dintre cele incarcate in memorie are propia *tabela de pagini*.

O adresa virtuala este de forma **(s, p, d)** unde **s** este numarul segmentului, **p** este numarul paginii virtuale in cadrul segmentului, iar **d** este deplasamentul in cadrul paginii. O adresa fizica este de forma **(f, d)**, unde **f** este numarul paginii fizice, iar **d** este deplasamentul in cadrul paginii.

MULTICS a fost primul So care a folosit (a fost cel care a introdus) aceasta modalitate de alocare. Calculatoarele IBM-PC cu microprocesor cel putin 80386 dispun de mecanism hard de gestiune paginata si segmentata a memoriei extinse.

Planificarea schimburilor cu memoria

CAT?

toata cantitatea ceruta: alocare statica, sau doar de cat e nevoie la un moment dat: alocare dinamica

UNDE?

- la alocare cu partitii variabile. Politici de plasare care decide in care dintre zonele libere va fi plasat programul.

CAND?

- la sistemele cu paginare si la alocare cu partitii variabile. Politici de incarcare (fetch) care decid in ce moment o pagina virtuala este pusa intr-o pagina fizica.

CAND?

- pentru compactare (relocare).

CARE?

- la sistemele cu paginare. Politici de inlocuire (replacement) care alege pagina existenta in memoria operativa care va fi inlocuita cu pagina ce trebuie incarcata. Manevra de inlocuire poarta numele de swapping.

slide 15

Politici de plasare (UNDE?)

Presupunem ca toata cantitatea de memorie solicitata la un moment dat este formata dintr-*un sir de octeti consecutivi*, si exista un *depozit de memorie* (numit *heap*) de unde se poate obtine memorie libera. Rezolvarea cererilor presupune existenta a 2 rutine:

1. are sarcina de a ocupa (aloca) o zona de memorie si de a intoarce adresa ei de inceput.
2. are sarcina de a elibera spatiul alocat anterior, in vederea refolosirii lui.

slide 16

(analogie cu alocarea cu partitii variabile)

» Metoda primei potriviri (First-fit)
» Metoda celei mai bune potriviri (Best-fit)
» Metoda celei mai rele potriviri (Worst-fit)
» Metoda alocarii prin camarazi (Buddy-system)

Data fiind fragmentarea inerenta a memoriei, cea mai convenabila structura de date petru regasirea zonelor libere este *lista inlantuita*. Fiecare nod al listei va descrie o zona de memorie libera specificandu-i adresa de inceput, lungimea si adresa urmatorului nod; stocat la inceputul

zonei de memorie pe care o descrie; nu mai contine adresa de inceput; se numeste *cuvant de control*.

Metoda primei potriviri (First-fit)

Partitia solicitata este alocata in prima zona libera in care incape. Avantaj: simplitate.

Metoda celei mai bune potriviri (Best-fit)

Cautarea acelei zone libere care lasa dupa alocare cel mai mic spatiu liber.

Avantaj: economiseste zonele de memorie mai mari; dezavantaje: timpul suplimentar de caurate, si proliferarea blocurilor libere de lungime mica, *fragmentarea interna excesiva*.

Solutie 1: pastrarea spatiilor libere in ordine crescatoare a lungimii spatiilor. Dezavantaj: comasarea zonelor libere adiacente se complica foarte mult.

Metoda celei mai rele potriviri (Worst-fit)

Cautarea zonei libere care lasa dupa alocare cel mai mult spatiu liber.

Avantaj: fragmentarea interna nu evolueaza asa de rapid. Dezavantaj: cautare este mai mare ca in cazul Best-fit.

Solutie 1: pastrarea spatiilor libere in ordine descrescatoare a lungimii spatiilor. Dezavantaj: comasarea zonelor libere adiacente se complica foarte mult.

Metoda alocarii prin camarazi (Buddy-system)

La sistemele Buddy, dimensiunile spatiilor ocupate si a celor libere sunt de forma $\mathbf{2^k}$, unde **m <= k <= n** (**m** - unitatea de alocare a memoriei, iar **n** - cea mai mare putere a lui 2 prin care se poate exprima dimensiunea memoriei interne).

Idea este de a pastra liste separate de spatii disponibile pentru fiecare dimensiune $\mathbf{2^k}$. Vor exista astfel **n-m+1** liste de spatii disponibile.

Pentru dimesiunea de 640Ko= $10*2^{16}$, n=16, daca consideram m=8 (unitatea de alocare 256 octeti), vom avea 9 liste:

1. lista de ordin 8 avand dimensiunea unui spatiu de 256 octeti
2. lista de ordin 9 cu spatii de dimensiune 512 octeti

9. lista de ordin 16 poate avea maximum 10 spatii de cate 65536 (2^{16}) octeti

Prin definitie orice spatiu (liber sau ocupat) de dimensiunea 2^k are adresa de inceput un multiplu de 2^k.

Doua spatii libere de ordinul **k** se numesc *camarazi (Buddy)* de ordin **k**, daca adresele lor **A1** si **A2** verifica:

```
A1 < A2,        A2 = A1 + 2^k    si      A1 mod 2^(k+1) = 0
A2 < A1,        A1 = A1 + 2^k    si      A2 mod 2^(k+1) = 0
```

Atunci cand intr-o lista de ordin k apar doi camarazi, sistemul ii concateneaza intr-u spatiu de dimensiune 2^{k+1}.

Avantaje: manipularea comoda a adreselor de zone si determinarea usoara a camarazilor:

Un numar binar este multiplu de 2^k daca se termina in k zerouri binare. Rezulta deci ca adresele a doi camarazi difera doar prin bitul de pe pozitia k (pozitia incepand cu 0). De ex:

```
A:  1010000000000000
B:  1010010000000000
        _____^
```

Politici de incarcare (CAND?)

slide 17

Cand sa se aduca o anumita pagina in memorie pentru ca cererile de pagini sa se reduca?

Solutie 1: Incarcarea la inceput a tuturor paginilor: DAR dispare efecutul mecanismului de paginare!

Solutie 2: Aducerea la cerere; cand ea e solicitata. - cea mai folosita metoda

Alte metode aduc pagini in avans pe *principiul vecinatatii* (P.J. Denning, 1968) care spune ca spune ca este foarte probabil ca urmatoarele referiri sa fie facute dintre paginile vecine.

Politici de inlocuire (CARE?)

In cazul in care memoria operativa este ocupata in intregime si trebuie sa se aduca o noua pagina, alta pagina existenta deja in memorie trebuie evacuata pentru a crea spatiul necesar.

Solutie: *Se evacueaza acea pagina care va fi solicitata in viitorul cel mai tarziu!?!* - nu se poate prevedea!!!

slide 18

» Metoda NRU (Not Recently Used)
» Metoda FIFO (First In First Out)
» Metoda LRU (Least Recently Used)

Metoda NRU (Not Recently Used)

Fiecare pagina are 2 biti prin care se va decide pagina de evacuat:

» R - bitul de referire - 0 la incarcare, 1 la fiecare referire, periodic (la 20 milisecunde) pus iarasi pe 0
» M - bitul de modificare - 0 la incarcare, 1 la modificare

Paginile se impart in 4 clase. Se cauta mai intai in clasa 0, apoi in 1, etc.

Desi nu optimal este foarte eficient.

Metoda FIFO (First In First Out)

Implementare simpla: lista in ordinea incarcarii lor care se actualizeaza la fiecare noua incarcare. Se inlocuieste prima din lista.

Bitul M indica daca pagina trebuie salvata sau nu inaintea inlocuirii.

O forma imunatatita: NRU si FIFO in cadrul fiecarei clase.

O alta imbunatatire (*metoda celei de-a doua sanse*). Daca bitul R=0 se inlocuieste, daca R=1, se pune pe 0 si se muta ultima in lista

Dezavantaj: anomalia lui Belady - sansa ca o pagina sa fie inlocuita NU scade pe masura ce numarul de pagini fizice creste

Metoda LRU (Least Recently Used)

Se bazeaza tot pe principiul vecinatatii. Nu exista lista, ci un registru contor pe 64 biti care creste la fiecare accesare, memorat in tabela de pagini. Se inlocuieste pagina cu varoare cea mai mica a contorului.