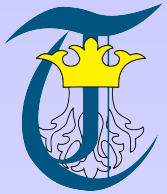


Universitatea *Transilvania* Brașov
Facultatea de Inginerie Electrică și Știința Calculatoarelor
Departamentul de Electronică și Calculatoare

Sisteme de operare

– curs 12 –

2012.05.29

ș.l. dr. ing. Kertész Csaba-Zoltán

4. Sisteme de fișiere

- stocarea informației pe suporturi externe rezolvă 3 probleme importante:
  - posibilitatea stocării unei mari cantități de informație
  - informația trebuie să se păstreze chiar dacă procesele care o utilizează s-au terminat
  - posibilitatea proceselor de a accesa concurent informația
- informația va fi stocată pe suporturile externe în unități numite **fișiere**
- managementul fișierelor este realizat de SO
  - sistemul de fișiere (implementează fișierele și realizează organizarea, utilizarea, protecția lor)

4.1. Fișiere

- analiza fișierelor din punctul de vedere al utilizatorului
  - numele fișierelor
  - structura fișierelor
  - tipuri de fișiere
  - accesul și operații asupra fișierelor
  - organizarea fișierelor

4.1.1. Numele fișierelor

- fișierele asigură o modalitate de stocare a informației pe disc
- utilizatorul nu va trebui să cunoască detaliile de implementarea pentru a lucra cu fișiere
- el va referi un fișier printr-un nume
- numele de fișier va trebui să se supună unor reguli specifice fiecărui SO
- de obicei numele fișierelor conțin o extensie (folosită pentru a indica tipul sau utilizarea fișierului)

4.1.2. Structura fișierelor

- fișierele se pot stoca în 3 posibilități
  - secvență de octeți (UNIX, DOS)
  - secvență de înregistrări (sisteme cu cartele, CP/M)
  - arbore de înregistrări (mainframe-uri pentru baze de date comerciale)
    - se scrie o înregistrare specificând o cheie

4.1.3. Tipuri de fișiere

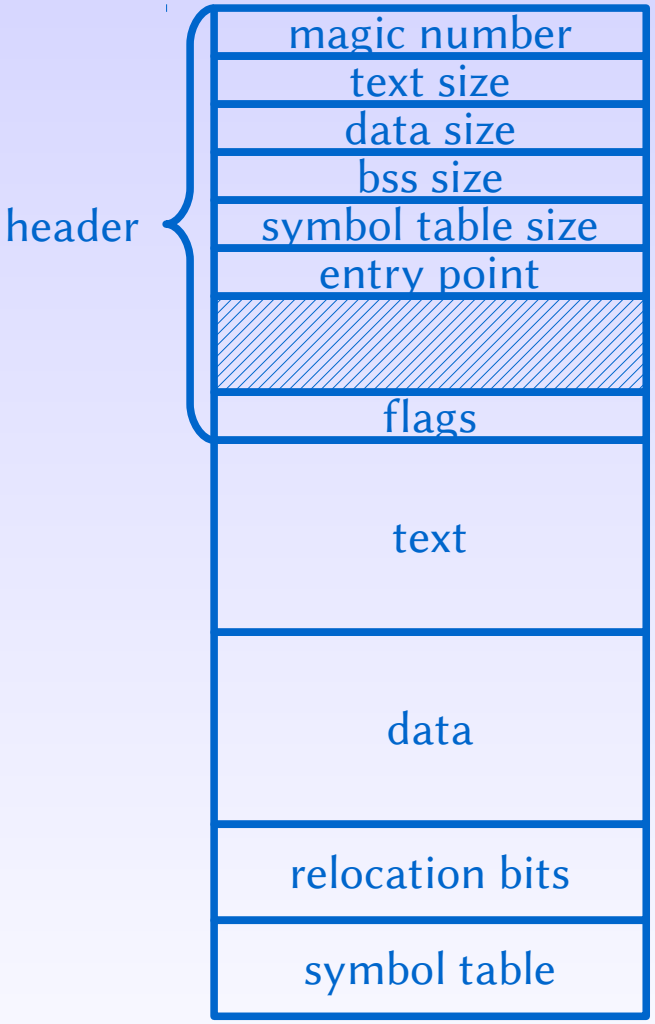
- în UNIX există 7 tipuri de fișiere
  - fișier normal
    - conține informații ale utilizatorilor
  - director
    - fișier care conține numele altor fișiere și pointeri la informațiile din respectivele fișiere
  - fișiere speciale de tip caracter
    - sunt asociate unor dispozitiv I/O de tip caracter (terminale, imprimante)
  - fișiere speciale de tip bloc
    - sunt asociate unor dispozitive I/O de tip bloc (disc, tape,...)

Tipuri de fișiere

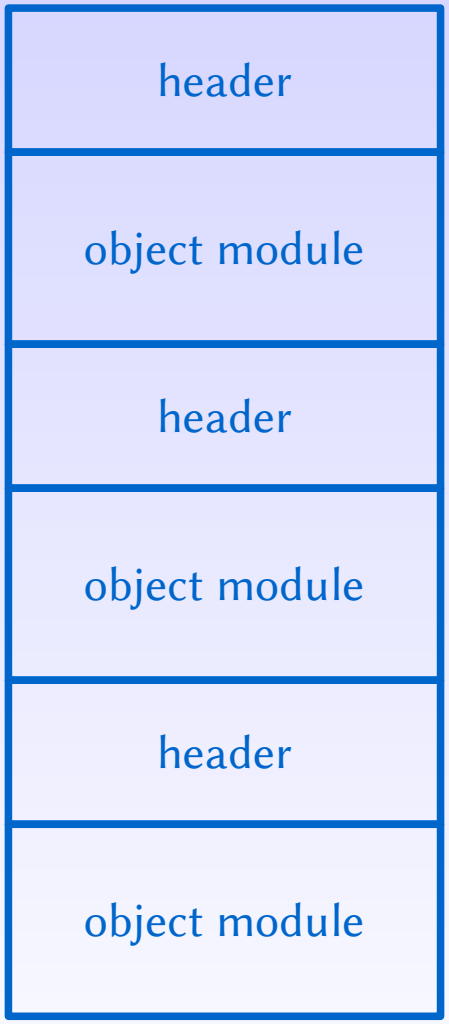
- FIFO (pipe)
  - fișier special utilizat la comunicația între procese
- socket
  - folosit pentru comunicație între procese în rețea
- legături simbolice
  - tip special de fișier care conține numele altui fișier căruia îi este asociat legătura

- fișierele normale pot fi
  - ASCII (linii de text, pot fi afișate de un editor)
  - binare (nu pot fi citite ca și text, conține coduri binare diferite)

Fișier binar executabil



fişier executabil



arhivă / bibliotecă

4.1.4. Accesul la datele dintr-un fișier

- în sistemele actuale toate fișierele sunt cu acces aleator
  - octeții sau înregistrările pot fi citite în orice ordine
- există fișiere cu acces secvențial
  - de exemplu: benzi magnetice

4.1.5. Atributele unui fișier

- SO asociază fiecărui fișier câteva atribute
  - UID, GID, rwx, data (creării, modificării, ultimului acces)

4.1.6. Operații cu fișiere

- apeluri sistem pentru lucrul cu fișiere
  - create
    - se creează un fișier care nu conține date (**creat**)
  - delete
    - fișierul este șters (**remove**, **unlink**)
  - open
    - înaintea utilizării unui fișier acesta trebuie deschis
    - scopul este de a încărca atributele și lista adreselor de pe disc în memorie pentru accesul rapid la următoarele apeluri
  - close
    - când nu se mai utilizează un fișier atunci va trebui închis pentru a elibera memoria

Operații cu fișiere

- read
  - citirea datelor dintr-un fișier de la poziția curentă
- write
  - scrierea unor date într-un fișier la poziția curentă
- append
  - formă restricționată de scriere: datele vor fi scrise numai la sfârșit (**O_APPEND**)
- seek
  - poziționarea deplasamentului într-un fișier (**lseek**)

Operații cu fișiere

- get attributes
  - preluarea atributelor unui fișier (**stat**, **access**)
- set attributes
  - modificarea atributelor unui fișier (**chown**, **chmod**)
- rename
  - schimbarea numelui unui fișier

4.2. Directoare

- sunt fișiere care conțin numele altor fișiere și pointeri la informațiile din respectivele fișiere
- un sistem de fișiere de obicei are o structură ierarhică
  - **arbore de directoare**
- cale absolută: este formată din numele directoarelor ce trebuie parcurse de la rădăcină până la fișierul ce trebuie accesat
- cale relativă: este dată pornind de la directorul de lucru sau directorul curent

Apeluri de sistem pentru lucrul cu directoare

- create
  - creează un director având subdirectoarele . și .. (**mkdir**)
- delete
  - șterge un director (**rmdir**)
  - se poate șterge numai un director gol
- open
  - pentru a putea citi un director (lista de fișiere conținute) el va trebui să fie deschis (**opendir**)
- close
  - închide un director pentru a elibera memoria (**closedir**)

Apeluri de sistem pentru lucrul cu directoare

- read
  - permite citirea unui director (lista fișierelor) (**readdir**)
- rename
  - redenumește un director
- link
  - creează o legătură simbolică sau hard la un director
- unlink
  - șterge o legătură pentru un director

4.3. Implementarea sistemelor de fișiere

- folosirea fișierelor din punctul de vedere a SO

4.3.1. Implementarea fișierelor

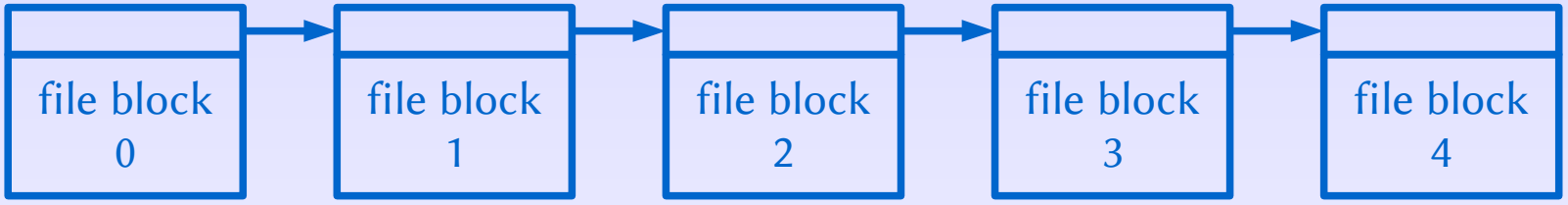
- cea mai importantă problemă pe care trebuie rezolvat de SO: cum ținem evidența blocurilor care sunt asociate unui fișier

Alocarea continuă

- cea mai simplă schemă de alocare este să stocăm fiecare fișier pe disc ca un bloc continuu de date
- avantaje
  - ușor de implementat, evidența blocurilor asociate unui fișier se poate ține foarte ușor (memorarea adresa de început a primului bloc)
  - performanțe excelente deoarece întregul fișier poate fi citit realizând o singură căutare
- dezavantaje
  - nu se poate utiliza dacă nu se specifică o limită a fișierului, SO va trebui să știe cât spațiu pe disc să aloce
  - fragmentarea discului

Alocarea utilizând liste înlănțuite

- un fișier = listă înlănțuită de blocuri
- primul word a fiecărui bloc este utilizat ca pointer la următorul bloc, restul blocului va conține datele



- avantaje
  - nu se pierde spațiu prin fragmentare deoarece orice bloc liber se poate utiliza pentru un fișier
- dezavantaje
  - accesul aleator la datele din fișier este mai lentă
  - pointerul va consuma câțiva octeți deci blocul nu va mai fi de dimensiune putere a lui 2

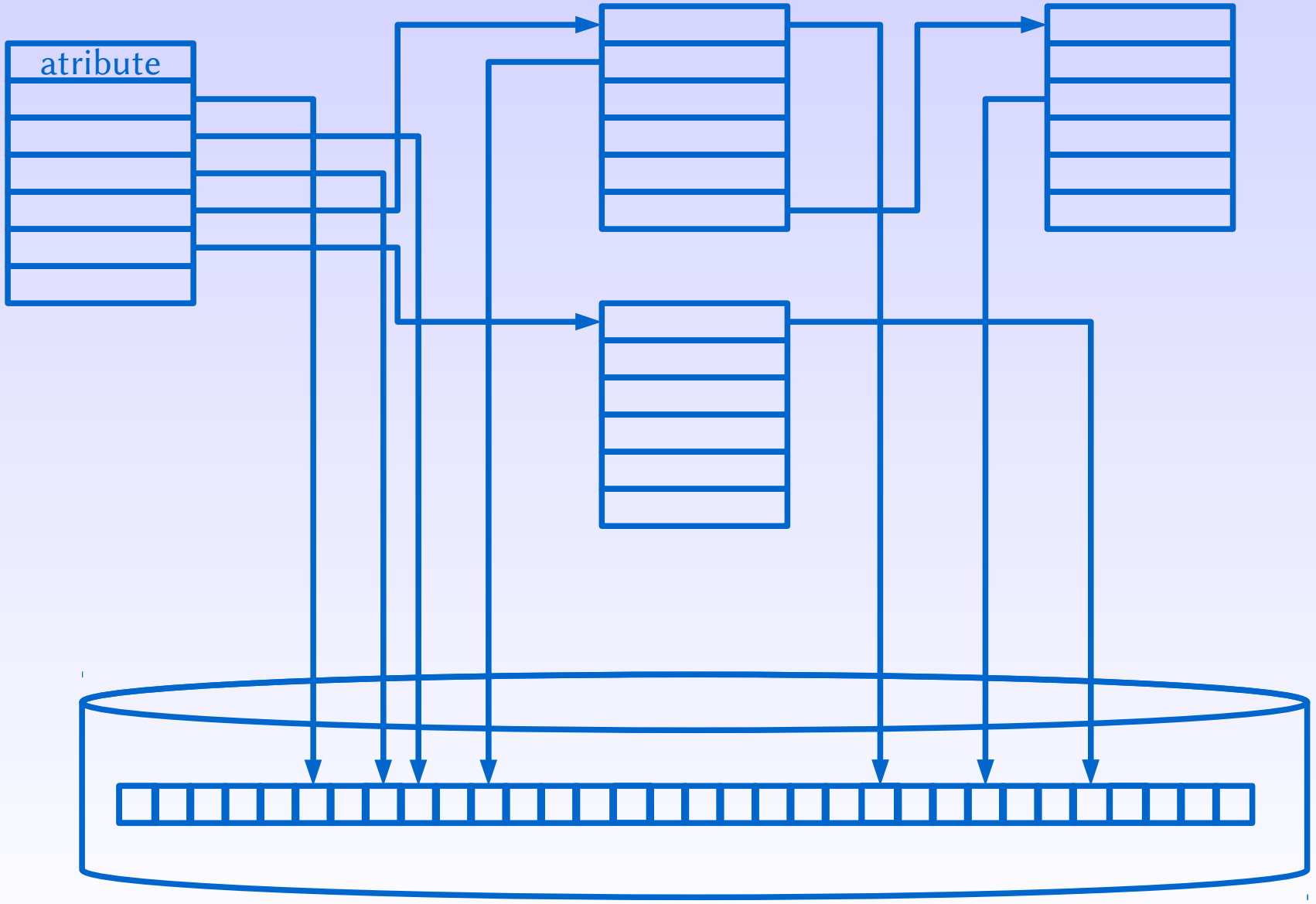
Alocarea cu liste înlănțuite utilizând un index

- elimină dezavantajele metodei anterioare
- se formează o tabelă sau index în memoria principală
- această tabelă va conține pointerii la blocuri
- avantaje
    - blocurile pot avea dimensiuni puteri a lui 2
    - accesul aleator este mai rapid (se fac doar accesări de memorie, nu și de disc)
    - directorul va menține doar numărul primului bloc
- dezavantaje
    - tabela trebuie să fie în memorie tot timpul

Alocare utilizând i-node-uri

- se asociază fiecărui fișier o tabelă numită **index-node** sau **i-node** care conține atributele fișierului și adresele de pe disc ale blocurilor care aparțin fișierului
- pentru fișiere mici adresele blocurilor sunt conținute în totalitate în i-node, dacă fișierul este mare se pot utiliza 1,2 sau 3 nivele de indirectare
- i-node-ul este încărcat în memorie atunci când este deschis fișierul

I-node-uri



4.3.2. Implementarea directoarelor

- directoarele vor trebui să asigure informațiile necesare pentru regăsirea blocurilor de pe disc
- aceste informații depind de sistem astfel:
  - pentru alocare continuă: adresa fișierului
  - pentru alocare cu liste: numărul primului bloc
  - pentru alocare cu i-node: numărul i-node-ului
- funcția unui director este de a realiza maparea între numele fișierelor și informațiile necesare pentru a localiza datele aparținând fișierelor

Exemple de implementarea a directoarelor

- MS-DOS

- UNIX

Implementarea directoarelor UNIX

- toate informațiile despre tipul, mărimea, timpii, proprietar și blocurile de pe disc sunt conținute în i-node
- când se deschide un fișier sistemul primește numele său și va trebui să localizeze blocurile care îi aparțin
- sistemul localizează directorul „root”, care are o poziție fixată pe disc
- se determină i-node-ul corespunzător directorului și se caută numărul i-node-ului fișierului respectiv
- i-node-urile au o adresă fixă pe disc

4.3.3. Fișiere partajate

- este nevoie de partajarea unor fișiere între mai multi useri
- aceste fișiere partajate trebuie să apară în mai multe directoare
- conexiunea dintre directorul B și fișierul partajat se numește link
- la folosirea linkurilor sistemul de fișiere nu mai are o structură de arbore, ci o structură DAG (Directed Acyclic Graph)

Soluții pentru fișiere partajate

- directorul B pointează către i-node-ul fișierului partajat (**hard link**)
- sistemul creează un fișier de tip link care conține calea și numele fișierului partajat (**symbolic link**)
- legăturile hard prezintă anumite probleme:
  - când proprietarul fișierului șterge fișierul, celălalt director va pointa către un i-node invalid
- se folosește contorul de legături

4.3.4. Managementul spațiului de pe disc

- există 2 strategii pentru a stoca un fișier pe disc
  1. se alocă o zonă continuă pe disc
  2. fișierul este împărțit într-un număr de blocuri
- dezavantajul metodei 1. este că atunci când fișierul crește peste mărimea alocată, el va trebui mutat în altă zonă
- majoritatea sistemelor de fișiere utilizează a doua strategie

Mărimea blocului

- o problemă importantă în proiectarea sistemelor de fișiere este alegerea dimensiunii blocului de pe disc
- candidați pentru unitatea de alocare sunt: un sector, o pistă sau un cilindru
- având o unitate de alocare mare (ex. cilindru) vom avea o fragmentare internă foarte mare
- având o unitate de alocare mică (ex. sector) un fișier va avea foarte multe blocuri, citirea unui fișier se va face foarte încet
- în mod uzual blocurile sunt de 512B–4kB

Evidența blocurilor libere

- se utilizează 2 metode
  - utilizarea unei liste înlănțuite de blocuri care conțin numere de blocuri libere
    - dacă avem blocuri de 1kB, ce pot conține 512 numere de blocuri libere, un disc de 20MB gol are nevoire de 40 de blocuri pentru stocarea numerelor
  - utilizarea unui bitmap: fiecărui bloc de pe disc corespunde un bit din bitmap
    - discul de 20MB are nevoie de 20Kb ~ 3 blocuri pentru bitmap