Sistemul de fişiere

-Optimizarea căutării fişierelor pe disc. Conceptul de director

-Alocarea de spaţiu pentru fişiere pe disc

-Evidenţa spaţiului liber de disc

Conceptul de director(catalog)

- În secţiunea anterioară am abordat problema găsirii unui mecanism cât mai eficient de accesare a informaţiilor dintr-un fişier. În continuare, ne vom ocupa de problema regăsirii cât mai rapide a fişierelor de pe un disc.

  - Orice disc este împărţit în **partiţii logice** sau **discuri virtuale** (una sau mai multe). Reciproc, există posibilitatea de a grupa mai multe discuri fizice într-un disc logic. Împărţirea discului în partiţii, oferă utilizatorilor o serie de avantaje, cum ar fi:

  - posibilitatea de funcţionare a sistemului respectiv sub mai multe sisteme de operare;

  - când se face reinstalarea sistemului de operare, nu se mai pierd prin formatarea discului toate informaţiile salvate pe discul respectiv, ci numai cele din partiţia respectivă;

  - fiecare partiţie conţine o **tabelă** cu informaţii despre fişierele care le conţine, accesului la aceste fişiere; astfel de tabelă poartă numele de **catalog** sau **director** şi fiecare intrare înte-un director conţine un **descriptor de fişier**.

- În cele ce urmează, vom studia diferite **structuri de directoare**, care au apărut odată cu evoluţia sistemelor de operare.
- Orice astfel de structură trebuie să respecte următoarele cerinţe:

  - c**ăutarea unui fişier.** O structură de directori conţine câte o intrare pentru fiecare fişier, în care se află descriptorul de fişier (sau un pointer către el). Operaţia trebuie să permită identificarea unui fişier sau a unui grup de fişiere care fac parte din această familie.

  -inserare. La crearea unui nou fişier, în sistemul de directori se va introduce o nouă intrare, care conţine descriptorul fişierului respectiv.

  - ştergere. La ştergerea unui fişier se şterg din directori informaţiile relative la fişierul respectiv.

  - listare. Fiecare SO dispune de comenzi care afişează conţinutul directorilor.

  - salvarea-restaurarea de directori, împreună cu fişierele subordonate(cele care corespund unor intrări în directorul respectiv).

  - parcurgerea directorului, adică posibilitatea accesării fiecărui fişier din director.

- **Scheme de organizare a unui sistem de directori.**
- Sunt cunoscute patru tipuri de structuri: structuri liniare(un singur nivel); structuri pe două niveluri; structuri arborescente; structură de graf aciclic.
- **Directori pe un singur nivel.**
- Avem un singur director la nivelul discului şi fiecărui fişier îi este asociată o intrare în acest director.
- Această organizare are dezavantajul că limitează spaţiul de nume al fişierelor, deoarece numele fişierelor trebuie să fie diferite, chiar dacă fişierele respective aparţin unor utilizatori diferiţi.
- Deci, numărul de fişiere este limitat şi în cazul redenumirii unui fişier, există pericolul ca noul nume să fie comun pentru două fişiere.
- **Directori pe două niveluri.**
- La primul nivel se află directorul **Master** (MFD - **M**aster **F**ile **D**irectory). Acesta conţine câte o intrare pentru fiecare utilizator al sistemului, intrare care punctează spre un **director utilizator**(UFD - User File Directory).
- Pentru fiecare nou utilizator al sistemului, se creează o intrare în MFD şi când un utilizator se loghinează la sistem, se face o căutare în UFD, pentru a se găsi directorul corespunzător numelui său.
- De fiecare dată când un utilizator se referă la un fişier, căutarea se va face numai în directorul asociat lui. Toate aceste probleme sunt rezolvate de către sistemul de operare, prin intermediul sistemului de fişiere.

- Această structură rezolvă problema limitării spaţiului de nume al fişierului, dar prezintă şi un mare dezavantaj: nu există posibilitatea comunicării între utilizatori, deci nu există posibilitatea partajării resurselor între utilizatori.
- O altă problemă apare atunci când un utilizator doreşte să utilizeze o componentă software de sistem. Pentru utilizarea acesteia, trebuie realizată o copie, aflată în directorul utilizatorului respectiv.
- **Directori cu structură de arbore**. Structura de directori pe două niveluri, poate fi privită ca un arbore a cărui adâncime este 2. În mod natural, s-a pus problema generalizării acestei structuri, adică de a reprezenta structura de directori ca un arbore de adâncime arbitrară.
- Fiecare disc logic conţine un director iniţial numit **root**(rădăcină).

- Un **fişier director** se deosebeşte de un fişier obişnuit numai prin informaţia conţinută în el. Un director(sau subdirector) conţine lista de nume şi adrese pentru fişierele sau subdirectoare subordonate lui.
- De obicei, fiecare utilizator are un director propriu care punctează la fişierele lui obişnuite sau alţi subdirectori definiţi de el.
- Toate directoarele au acelaşi format intern. O intrare într-un director conţine: Numele fişierului subordonat, Descriptorul fişierului(pointer spre el)
- Pentru a se face distincţia dintre un subdirector şi un fişier obişnuit, se foloseşte un bit care este setat pe 1 pentru un fişier, respectiv 0 pentru un subdirector.

- Fiecare director are două intrări cu nume special şi anume:
- “ . “ (punct) care punctează spre însuşi directorul respectiv;
- “ . . “ (două puncte succesive), care punctează spre directorul părinte.
- Într-o sesiune, la orice moment, utilizatorul se află într-un **director curent,** care conţine majoritatea fişierelor utilizate la un moment dat. De asemenea, la intrarea în sistem, fiecare utilizator se află într-un **director gazdă (home directory**.
- SO permite utilizatorului să-şi schimbe directorul curent, să-şi creeze un nou director, să afişeze calea de acces de la **root** la un director sau fişier etc.
- Orice fişier este identificat prin **calea** către el care, care este un şir de directoare separate printr-un caracter, ce poate porni de la rădăcină(cale absolută) sau de la directorul curent(cale relativă).

- Această generalizare permite utilizatorilor să creeze propriile lor structuri de directori. Sistemele de operare moderne folosesc o astfel de structură. Aceste SO împart fişierele în: **obişnuite** şi **directori. Sistemul Unix** foloseşte şi un alt tip de fişiere, şi anume cel **special.**
- Indiferent de sistemul de operare utilizat, **fişierele obişnuite** sunt privite ca şiruri de înregistrări sau succesiuni de octeţi, accesul putându-se realiza prin unul din mecanismele prezentate.
- Sub Unix, dispozitivele de I/O sunt privite ca fiind fişiere **speciale**. Din punctul de vedere al utilizatorului, nu există deosebiri între lucrul cu un fişier disc obişnuit şi lucrul cu un fişier special.
- Această abordare are o serie de avantaje:

  - simplitatea şi eleganţa comenzilor;

  - numele unui dispozitiv poate fi transmis ca argument în locul unui nume de fişier;

  - fişierele speciale sunt supuse aceluiaşi mecanism de protecţie ca şi celelalte fişiere.

- **Directori cu structură de graf.** Această organizare este utilă, atunci când un fişier trebuie să fie accesibil din mai mulţi directori părinte, ceea ce permite partajarea unui fişier de către mai mulţi utilizatori. Această structură este specifică sistemului Unix.
- **Observaţii.**

  - Directorii cu structură de arbore au fost introduşi pentru a înlocui căutarea secvenţială a fişierelor pe un disc cu căutarea arborescentă.

  - Pe lângă faptul că acest tip de directori nu mai limitează dimensiunea spaţiului de nume al fişierelor de pe un disc, permite utilizatorului să-şi structureze propria ierarhie de directoare în conformitate cu anumite criterii, să păstreze fişierele în anumite directoare, în funcţie de conţinutul lor.

Alocarea spaţiului pentru fişiere disc

- **Alocarea contiguă** cere ca fiecare fişier să ocupe un set de blocuri consecutive de pe disc.
- Deoarece adresele de pe disc ale blocurilor care compun fişierul sunt în ordine, pentru a accesa blocul de adresă b, după blocul b-1, nu necesită mutarea capului de citire, cu excepţia cazului când se trece de la ultimul sector al unui cilindru, la primul sector al următorului cilindru.
- Astfel, pentru accesarea directă a blocurilor fişierului, numărul de mutări ale capului discului este minimal.
- Dacă presupunem că fişierul conţine n blocuri şi adresa primului bloc este b, atunci fişierul va ocupa blocurile de adrese b, b+1, ...,b+n-1, deci în descriptorul de fişier din director, trebuie să se memoreze adresa de început şi lungimea zonei alocate.
- De asemenea, un astfel de fişier acceptă atât accesul secvenţial, cât şi cel aleator.
- Problemele ridicate de acest fel de alocare, precum şi modalităţile lor de rezolvare, coincid cu cele care apar la alocarea dinamică a memoriei operative.
- Şi aici apare fenomenul de **fragmentare**.
- Opţional şi aici se efectuează **compactarea**, în aceleaşi condiţii şi cu aceleaşi metode ca şi cele întâlnite la gestiunea memoriei.

- **Alocarea înlănţuită.**
- Rezolvă problemele legate de alocare contiguă(fenomenul de fragmentare).
- Fiecare fişier este privit ca o listă înlănţuită de blocuri de pe disc. Directorul conţine câte un pointer către primul, respectiv ultimul bloc al fişierului şi fiecare bloc conţine un pointer către următorul bloc al fişierului. Valoarea pointerului pentru ultimul bloc este nil.
- Toate operaţiile de găsire a blocurilor libere necesare fişierului, precum şi realizarea legăturilor între blocuri sunt realizate de către sistemul de operare.
- **Avantaj**:Această metodă evită fragmentarea discului.

- **Dezavantaje**:
  - nu permite decât **accesul secvenţial** la fişiere;
  - se consumă **spaţiu de pe disc** pentru pointeri:
  - **fiabilitatea-** deoarece pointerii se află pe tot discul, ştergerea unui pointer poate duce la distrugerea fişierului.
- O soluţie pentru evitarea acestui dezavantaj, este gruparea blocurilor într-o nouă unitate de alocare numită **cluster**.
- De **exemplu**, sistemul de fişiere poate defini clusterul ca având 4 blocuri. Această metodă are avantajul că diminuează spaţiul necesar memorării pointerilor, că micşorează numărul acceselor la disc atunci când se citesc/scrie informaţii de pe/pe disc. Totuşi, această metodă măreşte gradul de fragmentare al discului.
- **Exemplu.** Ordinea blocurilor pe care este memorat fişierul este: `9, 12, 1, 2, 11, 22, 25`. În director, în intrarea corespunzătoare fişierului, este trecut blocul 9 ca prim bloc al fişierului şi blocul 25, ca ultim bloc al fişierului. Fiecare bloc va conţine un pointer către următorul.

- Tabela FAT. O variantă a acestei metode, presupune utilizarea unei tabele de alocare a fişierelor**(FAT-F**ile **A**location **T**able**),** aflată la începutul fiecărui disc logic.
- Tabela este indexată după numărul de bloc
- Fiecare intrare va conţine numărul blocului care urmează în fişier blocului al cărui număr este index, dacă acest bloc este ocupat.
- Indexului ultimului bloc din fişier, îi corespunde o valoare specială, de sfârşit de fişier(-1).
- Blocurile neutilizate sunt marcate în tabelă cu valoarea 0.
- Pentru alocarea unui nou bloc unui fişier, se caută prima intrare din tabelă care are valoarea 0, şi se înlocuieşte valoarea anterioară de sfârşit de fişier, cu adresa noului bloc, iar valoarea 0 este înlocuită cu marca de sfârşit de fişier.
- **Exemplu**. Presupunem că avem un disc cu 12 blocuri, în care sunt memorate două fişiere(fis1, fis2). Presupunem că, iniţial fişierul fis1 ocupă blocurile 5, 2, 4 iar fis2 blocurile 6, 3, 9. În figura urm. a) sunt prezentate intrările în director, şi conţinutul **FAT** pentru această situaţie. Dacă fişierul fis2, va mai necesita un bloc, acesta va fi blocul 1, tabela **FAT** fiind cea din figura urm b).

Director

fis1	...	5
fis2	...	6

FAT

1	0
2	4
3	9
4	–1
5	2
6	3
7	0
8	0
9	–1
10	0
11	0
12	0

a)

FAT

1	-1
2	4
3	9
4	-1
5	2
6	3
7	0
8	0
9	1
10	0
11	0
12	0

b)

- **Alocarea indexată** .Fiecare fişier are o **tabelă de index,** în care se trec în ordine crescătoare adresele tuturor blocurilor ocupate de fişierul respectiv. Cea de-a `i`-a intrare din tabela de index conţine un pointer către cel de-al `i`-lea bloc al fişierului.
- Această tabelă se păstrează într-un bloc separat. În intrarea corespunzătoare fişierului din director, se păstrează această tabelă.
- Această tabelă poate avea o organizare arborescentă, în funcţie de dimensiunea fişierului, similară tabelei utilizate în paginarea memoriei interne. Dacă fişierul este mare şi ocupă mai multe adrese decât încap într-un bloc, atunci se creează mai multe blocuri de index, legate între ele sub forma unei liste simplu înlănţuite.

Evidenţa spaţiului liber de disc

- Deoarece spaţiul de pe disc este limitat şi se modifică în timp (alocarea de spaţiu pentru fişierele nou create sau pentru fişierele în care se adaugă informaţii şi reutilizarea spaţiului datorat ştergerii unor fişiere), sunt necesare metode prin care să se cunoască blocurile libere de pe discuri.
- **Evidenţa spaţiului printr-un vector de biţi**. Tabela(vectorul de biţi) care ţine evidenţa blocurilor libere, este indexată după numărul de bloc. Dacă un bloc este liber, atunci componenta corespunzătoare a vectorului este setată pe zero, iar în caz contrar pe unu.
- **Exemplu**. Presupunem că avem un disc cu 32 de blocuri, numerotate cu `1,..,32`, în care fişierele ocupă blocurile:

  `2, 3, 6, 7, 14, 15, 16, 19, 20, 21, 22, 28, 29, 30`.

  Atunci vectorul spaţiilor libere va fi:

  `(0,1,1,0,0,1,1,0,0,0,0,0,0,1,1,1,0,0,1,1,1,1,0, 0,0,0,0,1,1,1,0,0)`

- **Evidenţa prin listă înlănţuită**. În directorul volumului există un pointer către primul bloc liber. Acest bloc conţine un pointer către următorul bloc ş.a.m.d. Ultimul bloc liber conţine un pointer nul.
- **Observaţii**.

  1. Ordinea de apariţie a blocurilor libere în listă, este dată de acţiunile de cerere şi eliberare a blocurilor efectuate până la momentul respectiv.

  2. Lista este organizată pe principiul cozii. Astfel, dacă se cere ocuparea unui bloc, se va lua primul bloc din listă şi se va actualiza valoarea pointerului din director, al doilea bloc devenind primul. Eliberarea unui bloc presupune adăugarea unui nou element în listă, la celălalt cap.

  3. Deficienţa acestei metode constă în operaţiile cu pointeri care sunt foarte costisitoare pe disc.
- **Evidenţa înlănţuită şi indexată** constituie o îmbunătăţire a metodei anterioare**.**
- Directorul conţine un pointer către primul bloc liber, acesta conţine n pointeri către alte n blocuri libere, dintre care al n-lea va conţine alţi n pointeri către alte n blocuri libere ş.a.m.d.
- În acest mod, numărul operaţiilor de I/O necesare pentru căutarea de spaţiu liber se micşorează în medie de $n-1$ ori.