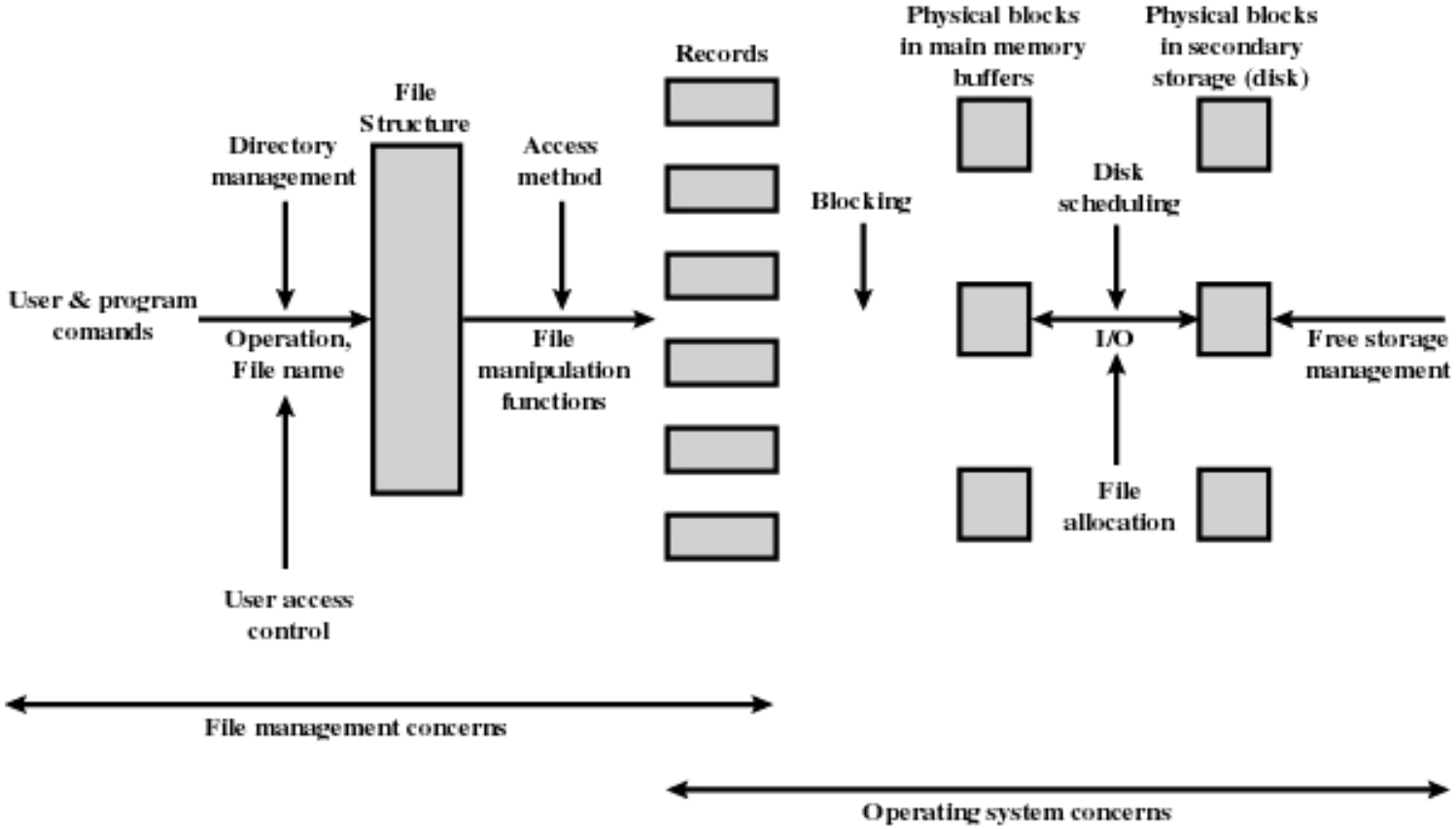
Sisteme de Operare

- Gestiunea sistemului de fişiere
  - Acţiunile SGF la nivel de fişier
    - Conceptul de fişier
    - Categorii de fişiere
    - Descriptorul de fişier
    - Moduri de organizare ale fişierelor
  - Acţiunile SGF la nivel de suport disc
  - Sisteme de directoare (cataloage)
  - Alocarea spaţiului pentru fişiere pe disc
  - Evidenţa spaţiului liber de pe disc

Gestiunea sistemului de fişiere

- Se face in mod transparent pentru utilizator prin intermediul sistemului de gestiune a fişierelor (SGF)



Acțiunile SGF la nivel de fişier

- Una din principalele funcţii ale sistemului de operare este implementarea unei "maşini virtuale", concretizate în cazul fişierelor prin asigurarea unor primitive de manipulare a fişierelor la un nivel cât mai abstract, care să nu implice cunoştinţe legate de modul de stocare
- Utilizatorul manipulează fişierele ca pe o secvenţă contiguă de octeţi în cadrul căreia se poate poziţiona oriunde, poate citi sau scrie, chiar dacă alocarea spaţiului pe disc se face la nivel de blocuri (de regulă, de dimensiune fixă cuprinsă între 512 octeţi şi 4k octeţi).

Acțiunile SGF la nivel de fişier

- Operaţiile realizate de un sistem de fişiere se referă la:
  - denumirea şi manipularea fişierelor, accesibile utilizatorilor ca entităţi cu nume şi asupra cărora se pot efectua operaţii (citire, scriere, execuţie);
  - asigurarea persistenţei datelor care presupune o independenţă faţă de crearea sau distrugerea proceselor din sistem (regăsirea după intervale mari de timp), posibilitatea refacerii structurii şi conţinutului în caz de accidente;
  - asigurarea mecanismelor de acces concurent al proceselor din sistem la informaţiile stocate.

Conceptul de fişier

- Informaţiile sunt grupate de către SO în ansambluri distincte numite fişiere.
- Prin intermediul programelor, un utilizator are acces la un moment dat la o mică entitate din cadrul unui fişier, numită şi articol sau înregistrare
  - este dificilă definirea exactă a unui articol în cadrul unui fişier

Categorii de fişiere din punct de vedere al structurii

- secvenţa de octeţi:
  - fişierul se prezintă ca o secvenţă de octeţi a cărei interpretare este lăsată la latitudinea programelor utilizator; oferă un grad maxim de flexibilitate în reprezentarea datelor (UNIX, Windows etc.);
- secvenţa de înregistrări:
  - fişierul este organizat ca o succesiune de înregistrări de lungime fixă;
  - operaţiile de citire şi scriere se fac la nivel de înregistrare (a apărut datorită cartelelor perforate şi nu mai sunt folosite în prezent);
- fişiere cu structură arborescentă:
  - fişierul este format din înregistrări, care conţin atât informaţia propriu-zisă, cât şi un câmp cheie (situat pe o poziţie prestabilită în cadrul fiecărei înregistrări).
  - Valoarea câmpului cheie este folosită de către sistemul de operare pentru gestionarea unei structuri logice interne de tip arbore a fişierelor.

Categorii de fişiere din punct de vedere al tipului

- fişiere normale (regular files):
  - conţin informaţii utilizator;
- directoare (directories):
  - fişiere sistem destinate gestionării structurii sistemului de fişiere;
- fişiere speciale de tip caracter/bloc (character/block special files):
  - destinate utilizării în conjuncţie cu dispozitivele periferice;
- legături simbolice (symbolic links):
  - furnizează posibilitatea accesării unui acelaşi fişier fizic pe căi logice multiple.

Fişierele normale

- ASCII:
  - fişierul este format din linii de text terminate cu caracterele de control CR (cariage return) şi/sau LF (line feed);
- binare:
  - fişierul este organizat ca secvenţă de octeţi, dar căruia sistemul de operare îi asociază o anumită structură internă (fişierele executabile şi arhivele).

Descriptorul de fişier

- Informaţiile de descriere a unui fişier sunt conţinute într-un articol special numit **descriptor de fişier**.
- De regulă, acesta nu este memorat la un loc cu articolele de informaţie ale fişierului, ci în director.

Descriptorul de fişier

- conţine patru grupe de informaţii:
  - identificatorul fişierului
  - informaţiile privind adresele fizice disc pe care le ocupă
  - informaţii de control al accesului la fişier
  - informaţii de organizare şi calendaristice

Descriptorul de fişier

- identificatorul fişierului:
  - este compus dintr-o pereche **(N, i)**
    - **N** este numele simbolic al fişierului
    - **i** este un număr prin care descriptorul este reperat pe disc în mod direct
      - este numit în diverse moduri: **i-node** (UNIX), file handle (MSDOS).
  - Prin **N**, SGF realizează legătura cu utilizatorul şi prin **i** cu celelalte componente ale SO.
  - Mai este folosit si de utilitarele de refacere a unei structuri de fişiere compromise.

Descriptorul de fişier

- informaţiile privind adresele fizice disc pe care le ocupă:
  - aici vor fi memorate informaţiile necesare regăsirii articolelor fişierului pe disc.
- informaţii de control al accesului la fişier:
  - aici sunt precizate informaţiile legate de cine şi ce drepturi are pentru a accesa fişierul.
  - tot aici sunt trecute informaţii legate de reglementarea accesului simultan la fişier.

Descriptorul de fişier

- informaţii de organizare şi calendaristice:
  - Aici se află informaţii legate de:
    - modul de organizare al fişierului (secvenţial, secvenţial-indexat, înlănţuit, etc);
    - formatul articolelor (fix, variabil, fişier text şi standardul de codificare);
    - data şi ora creării (ultimei actualizări, ultimei citiri),
    - numărul total de consultări a fişierului;
    - dispoziţii de păstrare a fişierului (permanent, temporar, păstrabil un număr de zile etc…)

Operațiile asigurate de SGF

- **prin functii de bibliotecă:**
  - **create**: crearea unui nou fişier;
  - **delete**: ştergerea fişierului;
  - **open**: indică SO intenţia unui proces de a accesa date conţinute într-un fişier existent;
  - **read**: citirea din fişier a unui număr specificat de octeţi, începând de la poziţia curentă, în spaţiul de adresare al procesului care a iniţiat operaţia;
  - **write**: înscrierea în fişier a unui număr specificat de octeţi începând de la poziţia curentă, cu date conţinute în spaţiul de adresare al procesului care a iniţiat operaţia;
  - **append**: adăugarea de date la sfârşitul unui fişier existent;
  - **close**: eliberarea structurilor de date sistem, ca urmare a terminării operaţiilor pe care un proces le-a executat asupra unui fişier;
  - **seek**: setează poziţia curentă în fişier;
  - **rename**: redenumeşte un fişier existent;
  - **setattributes**, **getattributes**: operaţii de setare şi citire a atributelor asociate unui fişier.
- **asociate unor comenzi**:
  - copierea, concatenarea, compararea, listarea, sortarea (ordonarea).

Operațiile asigurate de SGF

- **open**
  - informează SO că un anume fişier va deveni activ.
  - Prima sarcină a funcţiei de deschidere a unui fişier este de a face legătura între informaţiile de identificare a fişierului şi variabila din program prin care utilizatorul se referă la fişier.

Operațiile asigurate de SGF

- **open** - în funcţie de condiţiile de deschidere
  - A **Pentru un fişier care există deja pe disc**:
    - Realizează legătura dintre variabila din program ce indică fişierul, suportul pe care se află fişierul şi descriptorul de fişier aflat pe disc.
    - Verifică, pe baza informaţiilor de acces, dacă utilizatorul are sau nu drept de acces la fişier.
  - B **Pentru un fişier nou, ce urmează a fi creat**:
    - Alocă spaţiul pe disc pentru memorarea viitoarelor articole ale fişierului. În funcţie de tehnica de alocare folosită, se poate aloca începând de la o singură unitate de alocare (sector, bloc, cluster) şi terminând cu cantitatea maximală de spaţiu estimată de utilizator pentru fişierul în cauză.
  - C **Atât la fişier nou, cât şi la fişier deja existent**:
    - Alocă memorie internă pentru zonele tampon necesare accesului la fişier.
    - Încarcă rutinele de acces la articolele fişierului şi execută instrucţiunile lor de iniţializare.
    - Generează o formă cadru a instrucţiunilor de comandă a canalului I/O.
  - D **Memorează unele dintre datele obţinute la A, B, C** şi cele din descriptorul fişierului într-o **structură de date în memoria internă cunoscută sub numele de bloc de control al fişierului** (File Control Bloc sau FCB). De regulă fiecare utilizator are pentru fiecare fişier, propriul sau FCB (la unele SO este în zonele gestionate de sistem sau în monitorul sistemului)

Operațiile asigurate de SGF

- **close**
  - informează SO că un fişier încetează a mai fi activ.
  - Unele SO cer anunţarea explicită a închiderii, altele efectuează automat închiderea la sfârşitul programului.
    - La închidere, se reactualizează informaţiile generale despre fişier (lungime, adrese de început şi sfârşit, data modificării etc).

Operațiile asigurate de SGF

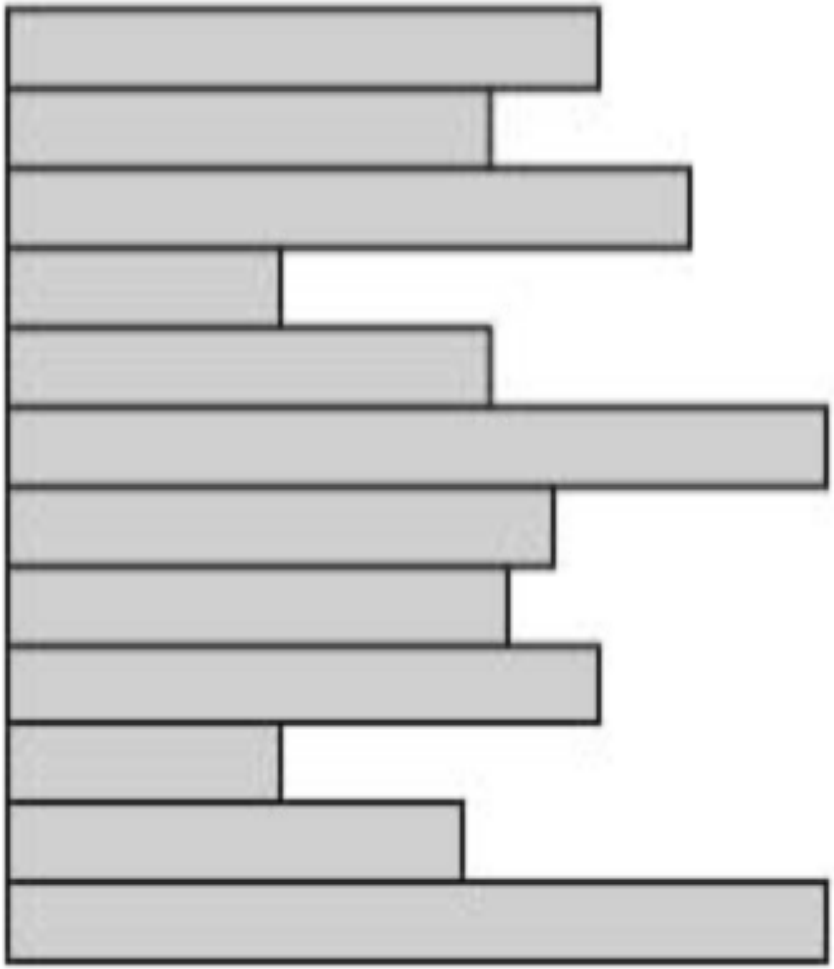
- **close** - în funcţie de condiţiile de deschidere
  - **Pentru fişiere temporare, create în programul curent şi de care nu mai este nevoie în continuare**:
    - Şterge fişierul şi eliberează spaţiul disc ocupat.
  - **Pentru fişiere nou create şi care trebuie reţinute**:
    - Creează o nouă intrare în directorul discului
  - **Pentru toate fişierele care trebuie reţinute**:
    - Inserează, dacă este cazul, marcajul EOF după ultimul articol al fişierului.
    - Goleşte (pune pe suport), dacă este cazul, ultimele informaţii existente în zonele tampon.
    - Pune la zi grupele de informaţii din descriptorul de fişier cu valorile curente
    - Dacă fişierul a fost modificat, marchează acest lucru şi eventual pune numele fişierului într-o listă a sistemului. Acest lucru trebuie făcut în eventualitatea unei salvări automate de către SO a tuturor fişierelor modificate.
  - **pentru toate fişierele, permanente sau temporare**:
    - Aduce capul de citire al discului în poziţia zero
    - Eliberează spaţiul de memorie ocupat de FCB.
    - Eliberează spaţiile ocupate de zonele tampon în memoria operativă.
    - Eliberează (eventual) perifericul sau perifericele pe care a fost montat suportul fişierului.

Moduri de organizare ale fişierelor

- Organizarea fişierelor este modalitatea în care sunt aranjate articolele sau unele părţi din ele astfel încât să permită modurile de acces dorite.

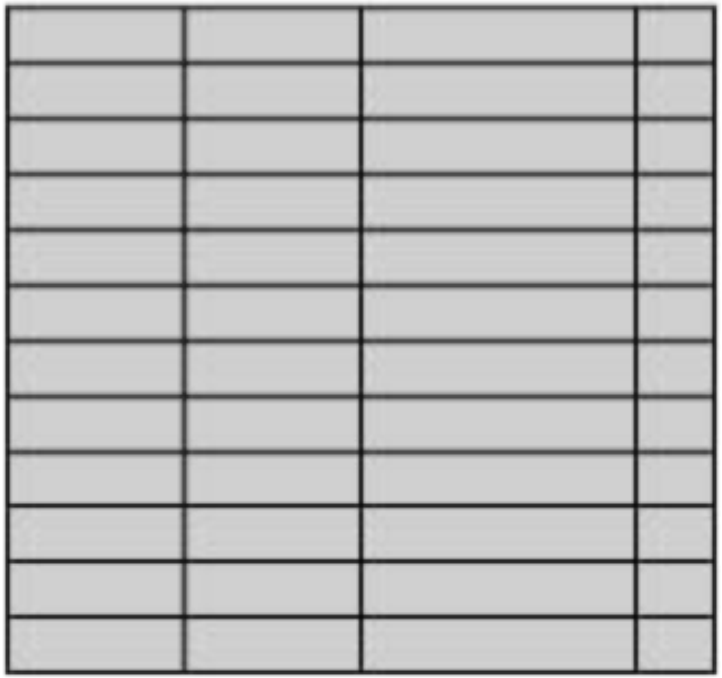
Moduri de organizare ale fişierelor

- Fişiere de tip "pile"(grămadă)
  - datele sunt colectate în ordinea în care apar;
  - scopul este acumularea şi salvarea unei cantităţi de date;
  - înregistrările pot avea câmpuri diferite;
  - nu au o structură bine definită;
  - accesul la o înregistrare poate fi foarte costisitor din punct de vedere al timpului.



Moduri de organizare ale fişierelor

- **Fişiere cu organizarea secvenţială**
  - înregistrările au un format fix (dimensiunea si ordinea articolelor este fixă, înregistrările sunt stocate în ordine pe baza unei chei – este un câmp particular în cazul unei înregistrări);
    - căutarea într-un astfel de fişier implică procesarea tuturor înregistrărilor;
  - este dificilă inserarea unor articole noi, de obicei fiind păstrate într-un fişier separat (log file sau transaction file);
  - o soluţie ar fi folosirea de liste înlănţuite pentru organizarea structurii fişierului.



Moduri de organizare ale fişierelor

- Fişiere cu acces direct prin poziţie
  - un astfel de fişier are, de regulă, articole cu format fix, care sunt plasate în sectoare vecine. Există posibilitatea de a avea acces direct la orice sector de disc alocat fişierului în cauză.
  - soluţia se poate extinde cu mici modificări şi la fişierele cu format variabil (de regulă, se reţine fie lista adreselor disc la care începe fiecare articol, fie lista lungimilor acestor articole).
  - majoritatea implementărilor de limbaje de programare de nivel înalt au mecanisme pentru acces direct prin poziţie pentru articole cu format fix (limbajul C oferă funcţia fseek).

Moduri de organizare ale fişierelor

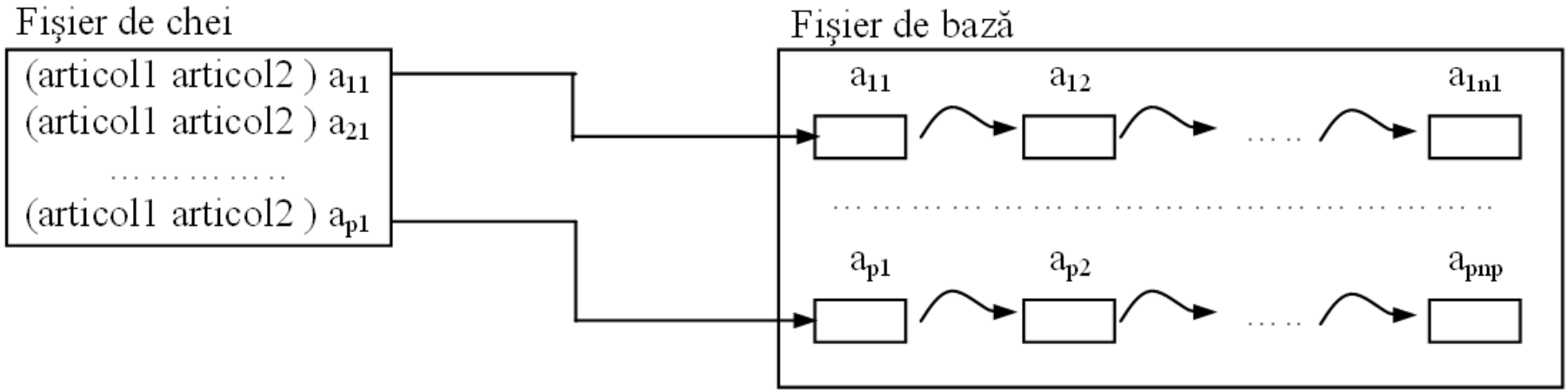
- Fişiere inverse
  - **Un fişier de bază**
  - se creează automat un **fişier invers**,
    - conţine pentru fiecare cheie specificată de utilizator, adresele disc la care se află articolele care conţin cheia respectivă.
  - Atât fişierul de bază, cât şi cel invers se creează şi se actualizează simultan.
  - Un astfel de fişier permite un mod foarte natural de organizare pentru a se realiza **acces direct prin conţinut** şi este folosit la unele SGBD-uri.

Moduri de organizare ale fişierelor

- **Fişiere multilistă**
  - Dacă fişierul de bază este foarte mare, atunci folosirea fişierelor inverse este ineficientă, deoarece articolele fişierului invers devin foarte lungi şi greu de manipulat
  - Acest lucru este înlăturat prin **fişierele multilistă**, dar accesul este secvenţial.
  - Pentru a putea realiza astfel de fişiere, utilizatorul defineşte o serie de atribute şi o serie de valori ale acestora care vor reprezenta **cheile fişierului**

Moduri de organizare ale fişierelor

- Fişiere multilistă
  - Fiecărei chei dintr-un articol îi este ataşată un pointer către articolul următor care conţine aceeaşi cheie.
  - Articolele fişierului de bază sunt legate prin atâtea liste câte chei sunt în fişier.
  - Acest mod de organizare se pretează pentru **acces direct prin conţinut**.



Moduri de organizare ale fişierelor

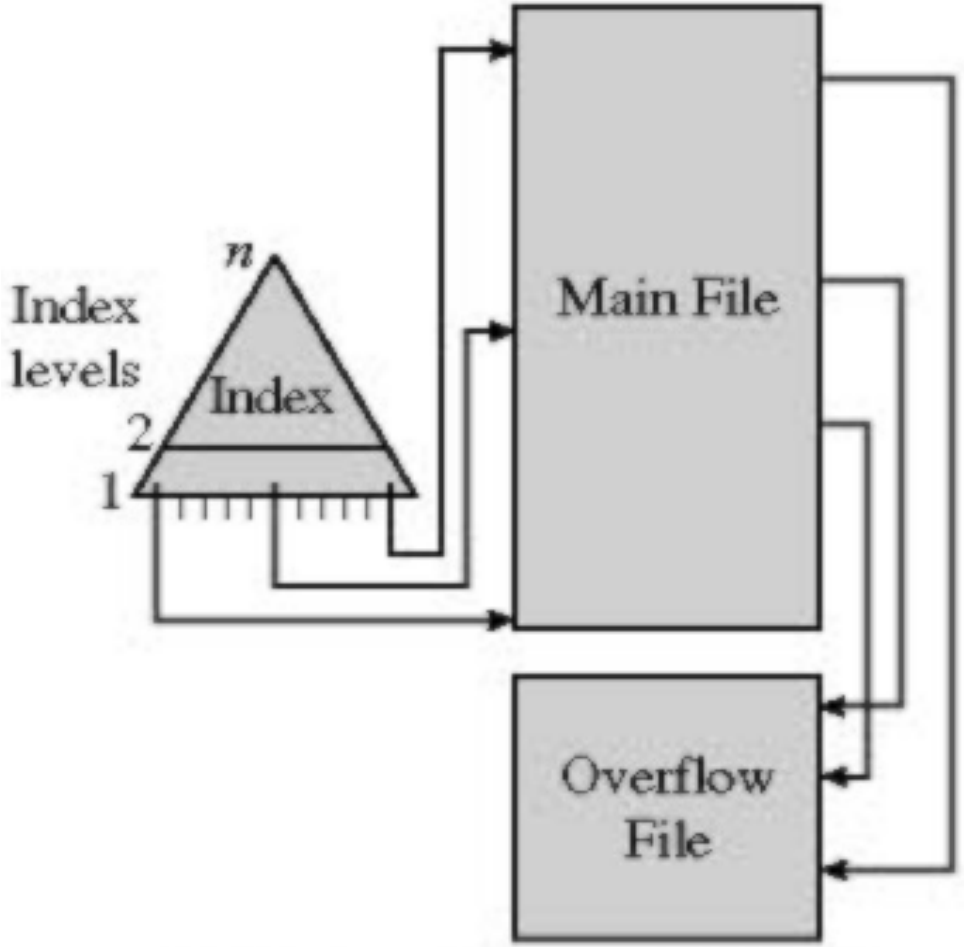
- **Fişiere secvenţial – indexate**
  - Acest tip de organizare a fişierelor este cel mai des folosit pentru **accesul direct prin conţinut**.
  - Articolele vor fi scrise pe suport în acces secvenţial şi plasate în ordinea crescătoare a indexului (unic pentru fiecare articol).
  - Articolele sunt grupate în blocuri de informaţii numite pagini
    - rezultă că toate valorile indexului dintr-o pagină sunt mai mari sau mai mici decât valorile indexului din altă pagină.
  - Împreună cu fişierul se creează şi o tabela de indecşi, în care se va trece pentru fiecare pagină, adresa disc a paginii şi valoarea celui mai mare index din pagină.
  - În funcţie de dimensiunea fişierului, dacă tabela de indecşi este prea mare se poate reorganiza sub forma unui arbore.
  - Acest tip de fişiere se regăsesc la sistemele de operare RSX şi CP/M.

Moduri de organizare ale fişierelor

- **Fişiere secvenţial – indexate**
  - Actualizarea unui astfel de fişier se face astfel:
    - articolele fişierului se leagă între ele printr-o listă simplu înlănţuită.
    - Articolul inserat este memorat pe disc într-o altă zonă numită parte de depăşire (overflow file) şi va face parte împreună cu precedentul şi cu succesorul din aceeaşi listă simplu înlănţuită.
    - Dacă se fac mai multe inserări între două chei vecine din partea principală, atunci lista va lega mai multe articole în partea de depăşire scăzând randamentul accesului la disc.
    - Ştergerile contribuie şi ele la scăderea randamentului, deoarece rămâne loc nefolosit în partea principală.
    - Din aceste motive este necesară reorganizarea frecventă a unui astfel de fişier.

Moduri de organizare ale fişierelor

- Fişiere secvenţial – indexate



Moduri de organizare ale fişierelor

- **Fişiere selective**
  - a apărut aproximativ în acelaşi timp cu cele secvenţial-indexate ca o replică a acestora în următorul sens:
    - dacă fişierele secvenţial-indexate au funcţia de regăsire materializată în tabelele de index, fişierele selective materializează funcţia de regăsire printr-un simplu calcul efectuat de CPU.
    - Se elimină astfel accesele la disc de căutare în tabele, dar funcţia de regăsire este dependentă de datele ce vor fi înmagazinate în fişierul respectiv.
  - Este folosit tot pentru acces direct prin conţinut.

Moduri de organizare ale fişierelor

- **Fişiere selective** – organizare
  - se presupune că datele care se vor înregistra permit existenţa unui atribut de valori unice pentru fiecare articol (un index);
  - se poate defini o funcţie:
    - f: {valori_posibile_index}→{0, 1, … , n-1} numită funcţie de randomizare sau funcţie **hash**;
    - definirea ei se face de utilizator; n – numărul de clase în care se împart articolele; toate articolele pentru care se obţine aceeaşi valoare a funcţiei se spune că sunt sinonime;
  - se recomandă ca numărul de articole sinonime să fie apropiat de numărul de articole care încap într-un bloc de informaţie (**pagină**);

Moduri de organizare ale fişierelor

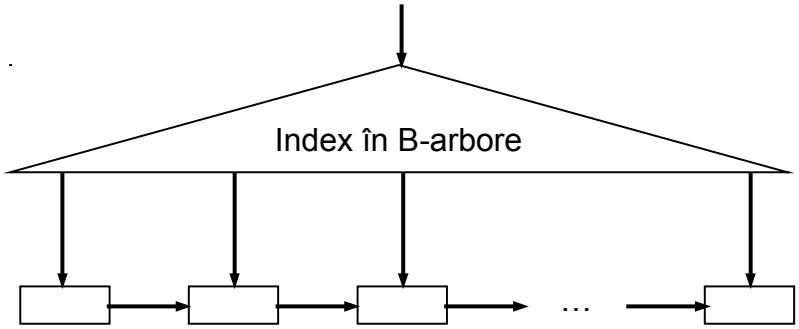
- **Fişiere selective**
  - Scrierea:
    - se aplică funcţia de randomizare valorii indexului şi se obţine numărul clasei în care se află articolul.
    - Partea principală a fişierului este formată din câte o pagină pentru fiecare clasă.
    - Dacă pagina din partea principală nu este încă plină, articolul se pune în pagina respectivă; dacă este plină se pune în partea de depăşire.
    - Articolul este legat printr-o listă simplu înlănţuită de ultimul articol sinonim cu el din partea principală.
  - Citirea:
    - se furnizează indexul articolului dorit căruia i se aplică funcţia de randomizare, determinând clasa de sinonime din care face parte. În cadrul clasei căutarea articolului se face secvenţial.

Moduri de organizare ale fişierelor

- **Fişiere organizate folosind B-arbori**
  - Această tehnică a fost adoptată datorită faptului că tabelele de indecşi pot fi foarte mari, precum şi datorită nevoii de a reduce numărul de accese la disc.
  - Un fişier bazat pe B-arbore este net superior unuia secvenţial indexat din următoarele motive:
    - cheile pot fi furnizate în orice ordine, nu neapărat în ordine crescătoare;
    - dispare zona de depăşire, performanţele de regăsire rămânând aceleaşi indiferent de numărul adăugărilor şi ştergerilor;
    - căutarea se face strict în acces direct, eliminându-se căutarea secvenţială în cadrul unei pagini.
  - B-arborii nu permit şi accesul secvenţial în ordinea crescătoare a cheilor. Din acest motiv au fost introduse două noi variante ce permit şi acest lucru: B+-arborele şi B-arborele extins.

Moduri de organizare ale fişierelor

- Fişiere organizate folosind B+-arbori
  - acest tip de arbore conţine chei (şi pointeri la fişierul de bază) numai în nodurile terminale (frunză) ale arborelui.
  - Nodurile interioare, organizate ca un B-arbore, au doar rolul de a indica nodul terminal ce conţine cheia căutată.
  - Nodurile terminale se leagă între ele printr-o listă înlănţuită, simplă sau dublă, permiţând astfel şi accesul secvenţial într-un mod foarte eficient.



Moduri de organizare ale fişierelor

- Fişiere organizate folosind B-arborele extins
  - Din raţiuni practice a apărut problema realizării accesului relativ într-un B-arbore
    - dacă la un moment dat ne aflăm pe cheia **c** (cheie curentă) în B-arbore, cum s-ar putea ajunge în mod direct la o cheie **k** (cheie nouă) aflată cu **n** chei (în ordine crescătoare) **înainte sau după** cheia **c**?

Moduri de organizare ale fişierelor

- Fişiere organizate folosind B-arborele extins
  - Soluţia constă în adăugarea unei informaţii suplimentare pe lângă fiecare pointer din B-arbore, informaţie ce constă dintr-un număr întreg care reţine câte chei conţine subarborele indicat de pointer
  - Pentru realizarea accesului relativ, trebuie efectuate în plus câteva operaţii de adunare şi scădere.
  - Pentru trecerea de la cheia curentă la o cheie nouă, se parcurg nodurile B-arborelui de la strămoşul comun al celor două chei şi până la cel ce conţine cheia nouă.

Acțiunile SGF la nivel de suport disc

- SGF are trei sarcini principale:
  - implementarea unui sistem de regăsire a fişierelor aflate pe volumul de disc respectiv;
  - evidenţa spaţiului neutilizat pe disc
  - punerea la dispoziţia componentelor care creează sau extind fişiere, a acestui spaţiu atunci când este nevoie (alocarea spaţiului pentru fişiere pe disc).

Sisteme de directoare (cataloage)

- Rezolvă o problemă fundamentală a sistemului de fişiere:
  - realizarea **legăturii** între **numele** atribuit de utilizator unui fişier şi **localizarea fizică** pe disc a informaţiei asociate fişierului respectiv, operaţie numită **mapare.**
- Structura unui director poate fi asimilată cu aceea a unei tabele în care fiecare intrare este asociată unui fişier şi conţine informaţiile necesare accesului:
  - numele fişierului şi adresa unei structuri de date sistem care conţine atributele fişierului
  - informaţii despre localizarea acestuia pe disc

Sisteme de directoare (cataloage)

- Operaţiile asigurate de sistemul de operare pentru lucrul cu directoare sunt:
  - **create**: crearea unui director;
  - **delete**: ştergerea unui director;
  - **open**: "deschiderea" unui director, de exemplu în scopul citirii şi afişării intrărilor;
  - **close**: eliberează structurile de date sistem care au fost alocate unui director ca urmare a unei operaţii open;
  - **readdir**: returnează următoarea intrare într-un director. Conţinutul directoarelor poate fi citit şi utilizând operaţia read (definită pentru fişiere) dar aceasta trebuie să aibă ca rezultat returnarea întregii tabele asociate directorului respectiv, obligând utilizatorul să cunoască structura acesteia în scopul regăsirii informaţiilor căutate;
  - **rename**: dă posibilitatea redenumirii unui director;
  - **link**: dă posibilitatea stabilirii unei legături simbolice care are ca efect apariţia numelui unui acelaşi fişier în mai multe locuri în cadrul sistemului de fişiere;
  - **unlink**: are ca efect ştergerea unei intrări într-un director.

Sisteme de directoare (cataloage)

- Tipuri de structuri logice
  - directoare cu un singur nivel:
    - este cea mai simplă structură.
    - toate fişierele de pe disc sunt în acelaşi director, fapt ce impune anumite restricţii asupra numelui şi numărului fişierelor.
      - numărul de intrări în tabelă este limitat de dimensiunea tabelei director şi nu pot exista două fişiere cu acelaşi nume chiar dacă aparţin la utilizatori diferiţi.
    - acest sistem era folosit la sistemele de operare CP/M şi SIRIS.

Sisteme de directoare (cataloage)

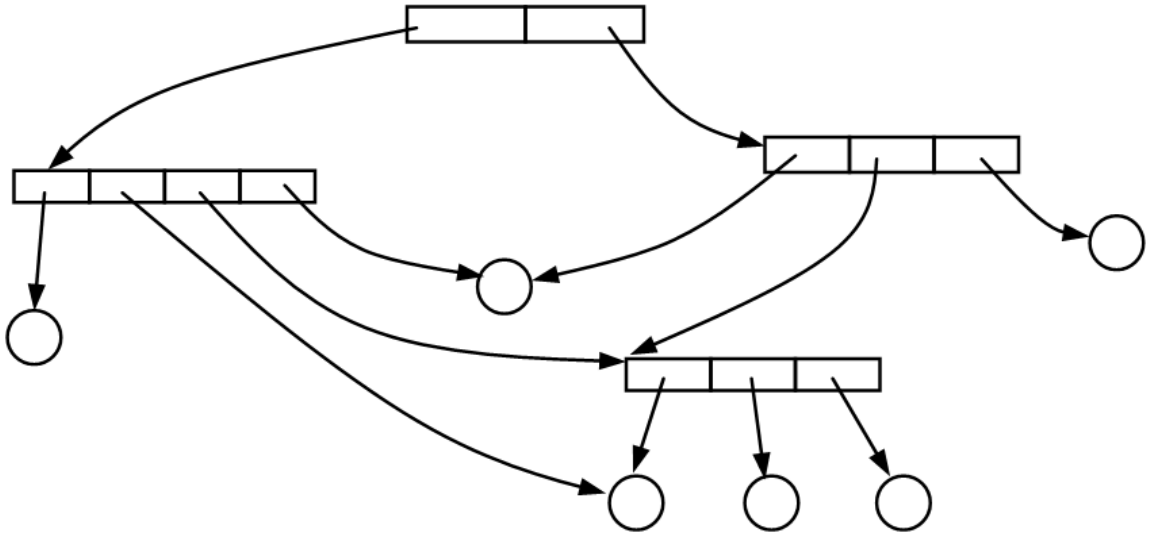
- Tipuri de structuri logice (2)
  - directoare cu două niveluri:
    - O astfel de organizare elimină neajunsul unicităţii numelui de fişier.
    - O astfel de structură conţine la nivel superior directorul Master (MFD – Master File Directory).
    - Acesta conţine câte o intrare pentru fiecare utilizator al sistemului, intrare care pointează către un director Utilizator (UFD – User File Directory).
    - Din fiecare UFD se pointează către fişierele utilizatorului respectiv. Acest sistem a fost folosit la sistemele de operare SIRIS şi RSX.

Sisteme de directoare (cataloage)

- Tipuri de structuri logice (3)
  - directoare cu structură de arbore
    - reprezintă extinderea cea mai firească a structurii cu două niveluri.
    - A început să fie folosită la SO UNIX şi la MS-DOS de la versiunea 2, singura deosebire între cele două fiind caracterul de separare între numele de director care este “/” pentru UNIX şi “\” pentru MS-DOS.

Sisteme de directoare (cataloage)

- Tipuri de structuri logice (4)
  - directoare cu structură de graf aciclic
    - acest tip de structură este util deoarece directoarele cu structură de arbore nu permit ca anumite fişiere sau directoare să fie accesibile din mai multe directoare părinte
    - util când se doreşte partajarea unei porţiuni a structurii de fişiere între mai mulţi utilizatori.



Alocarea spațiului pentru fişiere pe disc

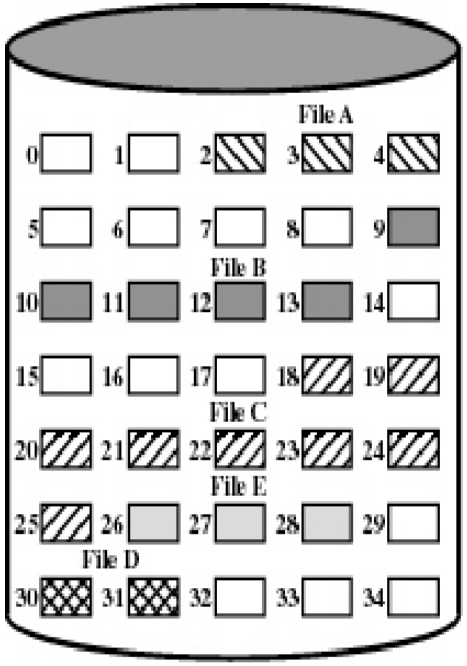
- **alocare statică (prealocare)**: în care trebuie specificată dimensiunea spaţiului ce va fi ocupat de fişier, lucru destul de greu de estimat, dar care va conduce la o alocare contiguă pe disc a fişierului;
- **alocare dinamică**: în care fişierul va ocupa atât spaţiu cât este necesar, dar pe disc nu vom mai avea locaţii contigue.
  - Este necesară folosirea unei tabele de alocare a fişierelor (File Allocation Table) care va păstra informaţiile legate de spaţiul alocat fiecărui fişier.
- Cele mai cunoscute metode de alocare a spaţiului pentru fişiere sunt:
  - alocarea contiguă, alocarea înlănţuită şi alocarea indexată.

Alocarea spațiului pentru fişiere pe disc

- **Alocarea contiguă**
  - **un fişier pe disc trebuie să ocupe un set de adrese consecutive pe disc. În descriptorul de fişier se pun adresa de început şi lungimea zonei alocate.**
  - Problemele ridicate de acest mod de alocare coincid cu cele care apar la alocarea memoriei cu partiţii variabile: apare fenomenul de fragmentare şi se efectuează compactarea.
  - Acest tip de alocare a fost folosită la SIRIS (alocă un număr întreg de cilindri pentru fiecare fişier) şi parţial la RSX.

Alocarea spațiului pentru fişiere pe disc

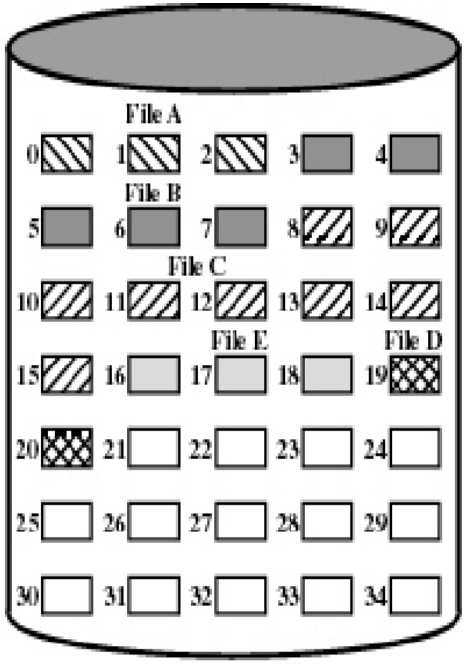
- Alocarea contiguă



File Allocation Table

File Name	Start Block	Length
File A	2	3
File B	9	5
File C	18	8
File D	30	2
File E	26	3

Alocare contiguă



File Allocation Table

File Name	Start Block	Length
File A	0	3
File B	3	5
File C	8	8
File D	19	2
File E	16	3

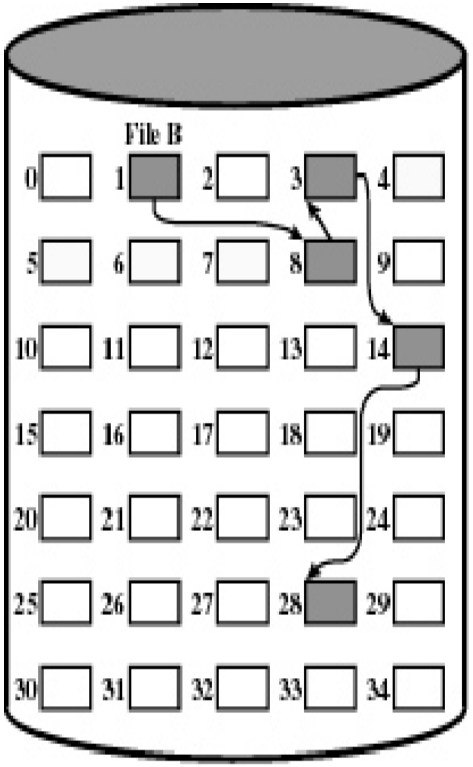
Alocare contiguă (după compactare)

Alocarea spațiului pentru fişiere pe disc

- **Alocarea înlănţuită**
  - **Fiecare fişier este înregistrat într-un şir de blocuri legate între ele printr-o listă înlănţuită.**
  - se foloseşte pentru a se evita fenomenul de fragmentare.
  - Această metodă este foarte bună pentru accesul secvenţial la fişiere.
  - Acest mod de alocare este utilizat la fişierele secvenţial-înlănţuite
  - Sistemul de fişiere de la MSDOS FAT12/16/32 foloseşte o variantă a acestui mod de alocare

Alocarea spațiului pentru fişiere pe disc

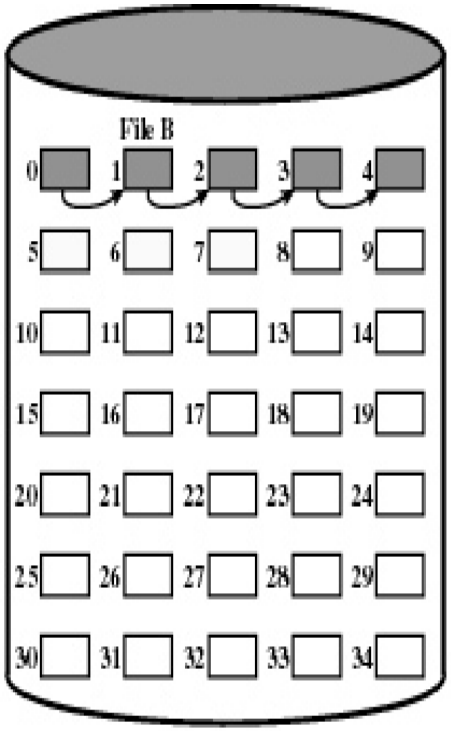
- Alocarea înlănţuită



File Allocation Table

File Name	Start Block	Length
•••	•••	•••
File B	1	5
•••	•••	•••

Alocare înlănțuită



File Allocation Table

File Name	Start Block	Length
•••	•••	•••
File B	0	5
•••	•••	•••

Alocare înlănțuită (după compactare)

Alocarea spațiului pentru fişiere pe disc

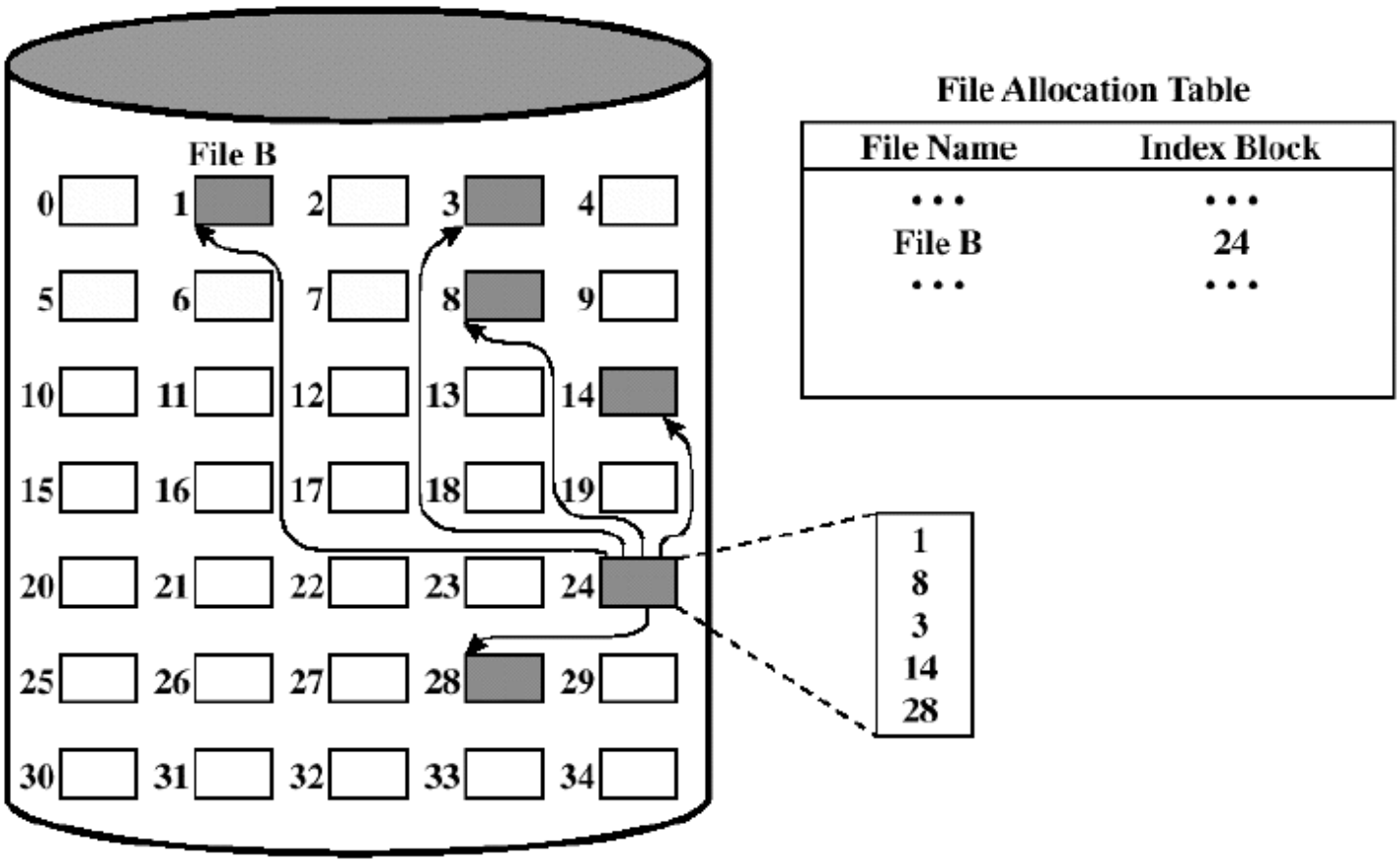
- **Alocarea indexată**
  - deoarece alocarea înlănţuită nu permite accesul direct, această problemă este rezolvată prin alocarea indexată.
  - **La acest tip de alocare, pe lângă blocurile ataşate fişierului, la crearea fişierului respectiv, creează un bloc special, numit bloc de index.**
    - În acest bloc, se trec în ordine adresele tuturor sectoarelor ocupate de fişierul respectiv.

Alocarea spațiului pentru fişiere pe disc

- **Alocarea indexată**
  - În descriptorul fişierului (din director) există un pointer către blocul de index.
    - Dacă fişierul este mare şi ocupă mai multe adrese decât încap într-un bloc, atunci se adoptă una din soluţiile următoare:
      - se creează mai multe blocuri de index, legate între ele sub forma unei liste simplu înlănţuite;
      - utilizarea unor blocuri index multinivel, numărul de niveluri va depinde de cât de mare se va dori a fi dimensiunea unui fişier;
      - utilizarea unei combinaţii între cele două soluţii.
  - Această schemă de alocare este foarte flexibilă şi este folosită cel mai mult în diverse sisteme de operare (UNIX adoptă o schemă de alocare indexată cu trei niveluri).

Alocarea spațiului pentru fişiere pe disc

- Alocarea indexată



Evidența spațiului liber de pe disc

- Principala problemă care se pune la alocarea spaţiului pe disc este utilizarea eficientă a spaţiului şi informaţiile să fie introduse/obţinute rapid.
- Rezolvarea acestora presupune, pe lângă o alocare eficientă a spaţiului şi o parte de evidenţă a spaţiului liber de pe disc.
- Metodele de evidenţă a spaţiului liber de pe disc sunt bazate pe **fişiere inverse** şi pe **liste înlănţuite**.

Evidența spațiului liber de pe disc

- **fişiere inverse**
  - În directorul volumului, există o zonă rezervată pentru ocuparea volumului, numită şi **tabela de ocupare a volumului** (TOV), în care este rezervat câte un bit pentru fiecare bloc de pe disc.
  - Dacă blocul este liber atunci bitul este zero, în caz contrar este unu.

    Exemplu: 00111000011111000011111111111011000

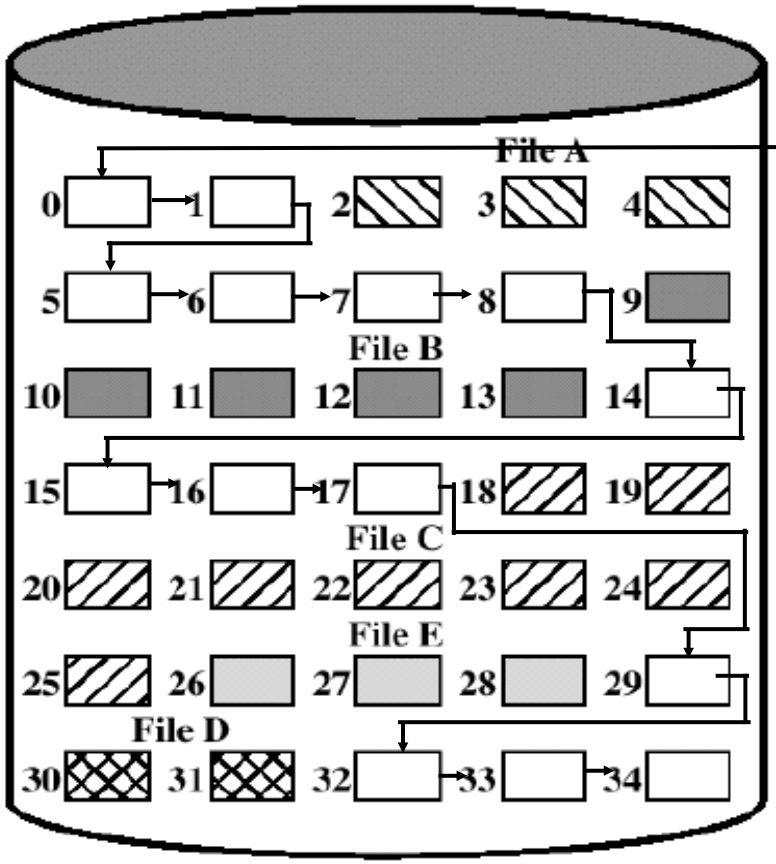
  - La alocarea unui fişier se caută şi se ocupă atâtea blocuri câte sunt necesare fişierului, după care toţi biţii corespunzători primesc valoarea unu.
  - La ştergerea fişierului biţii corespunzători se pun pe zero.
  - Acest mod de evidenţă a fost folosit: la CP/M, unde TOV se află pe pista 2, la RSX, unde este într-un fişier BITMAP.SYS.

Evidența spațiului liber de pe disc

- **listă înlănţuită**
  - În directorul volumului, există un pointer către primul sector liber, care la rândul lui conţine un pointer la următorul sector liber ş.a.m.d. Ultimul sector are pointer de valoare zero.
  - Ordinea apariţiei blocurilor libere în listă este dată de cererea şi eliberarea de blocuri până în momentul respectiv.
  - Astfel, dacă se cere ocuparea a **p** blocuri, se vor ocupa primele **p** poziţii din listă, după care al **p+1** – lea bloc liber din listă va deveni primul liber. Cererea de eliberare se face nominal pentru fiecare bloc, lista actualizându-se conform ştergerii dintr-o listă simplu înlănţuită.
  - Această schemă nu este foarte eficientă, deoarece pentru ocuparea unor sectoare este necesar un consum mare de timp pentru operaţiile de I/O care caută spaţiu liber
  - Începutul listei pointează spre primul bloc

Evidența spațiului liber de pe disc

- listă înlănţuită



File Allocation Table

File Name	Start Block	Length
File A	2	3
File B	9	5
File C	18	8
File D	30	2
File E	26	3

0

Evidența spațiului liber de pe disc

- **listă înlănţuită şi indexată**
  - Această metodă este o îmbunătăţire a metodei anterioare şi constă în următoarele:
    - Presupunem că într-un bloc pot fi înregistrate n adrese de blocuri disc. În primul bloc liber, indicat din director, se memorează adresele a **n** blocuri libere.
    - Primele **n-1** sunt adrese de sectoare efectiv libere, iar ultima adresă pointează la un bloc liber care conţine, iarăşi, adresele a **n** blocuri libere ş.a.m.d.
    - Astfel numărul de operaţii de I/O necesare pentru căutarea de spaţiu liber se micşorează în medie de **n-1** ori.
    - Ca urmare procedurile de eliberare şi alocare de blocuri vor fi complicate, dar acest lucru este compensat de reducerea masivă a numărului de accese la disc.
    - Mai mult, de aici se poate face uşor şi evidenţa spaţiului liber într-o structură arborescentă

Evidența spațiului liber de pe disc

- listă înlănţuită şi indexată