Sistemul de fisiere

- Conceptul de fisier

Conceptul de fişier

- Din punctul de vedere al sistemului există două modalităţi de a gestiona spaţiile de memorie secundară: utilizarea **memoriei virtuale** şi **sistemul de fişiere**.
- Dacă memoria virtuală este utilizată de către un proces în timpul execuţiei, fişierele sunt folosite pentru stocarea informaţiilor pentru o perioadă mai lungă de timp.
- Intuitiv, un fişier este un ansamblu de elemente de date, grupate împreună, având ca scop controlul accesului, regăsirea şi modificarea elementelor componente. Aceste elemente de date pot fi **octeţi** sau **articole** (înregistrări).
- La nivelul unui periferic, datele sunt stocate ca secvenţe de blocuri de informaţie. Pentru a fi utilizate de către aplicaţii, informaţiile sunt structurate sub formă de fişiere; componenta SO care se ocupă de gestionarea lor se numeşte sistemul de fişiere.
- Când sunt folosite de aplicaţii, se presupune că fişierele sunt structurate sub forma unei colecţii de articole (inregistrari).
- Informaţiile dintr-un articol sunt de regulă grupate în subdiviziuni, numite **câmpuri** sau **atribute**. Unele sisteme de operare furnizează numai facilităţi de translatare a blocurilor în fluxuri de octeţi, lăsând în seama aplicaţiilor structurarea fişierelor ca flux de articole. Astfel de sisteme de operare conţin un **sistem de fişiere de nivel scăzut**. Aplicatiile sunt cele care structureaza articolele ca o colectie de campuri.

- Sistemele de operare Windows, Linux şi Unix fac parte din această clasă. Sistemele de operare care privesc fişierele ca flux de înregistrări conţin un **sistem de fişiere de nivel înalt**.
- Fiecare câmp(atribut) al unui articol, are o anumită **valoare** în cadrul fişierului. O pereche **(atribut, valoare)** o vom numi cheie.
- Numim **index de articol,** un atribut cu proprietatea că pentru oricare două articole diferite ale fişierului, valorile atributului sunt diferite. Indexul de articol se mai numeşte şi **cheie de articol**.
- **Exemplul 1.** Considerăm un fişier care conţine informaţii despre studenţii unei universităţi(numele, prenumele, numărul matricol, data naşterii, locul naşterii, seria si numărul buletinului de identitate, sexul, facultatea din care face parte, notele obţinute la examene etc.).
- Inf. despre un student sunt inregistrate intr-un articol.
- Perechile (nume, ”Ionescu”), (sex, B), (matricol, 1089) sunt chei.
- Se observă că în acest fişier pot exista mai multe articole care conţin aceeaşi cheie (de exemplu mai mulţi studenţi cu acelaşi nume) şi de asemenea, pot fi chei care să nu existe în fişier.
- Atributele **număr matricol** precum şi atributul obţinut prin concatenarea câmpurilor **seria si numărul buletinului de identitate** reprezintă indexi**.**

- De cele mai multe ori, numărul de câmpuri dintr-un articol este acelaşi pentru toate articolele fişierului. Numărul de octeţi pe care se reprezintă un atribut al lui, este constant sau variabil. De exemplu, un atribut care este un numar întreg se reprezintă pe doi octeţi, indiferent de articol, dar un atribut care este un şir de caractere poate avea lungimi diferite de reprezentare.
- **Lungimea de reprezentare** a unui articol poate fi constantă pentru toate articolele fişierului sau poate să varieze de la un articol la altul, deci avem articole de **format fix** şi articole de **format variabil**. Pentru cele cu format variabil, trebuie să existe fie un câmp în care se trece lungimea efectivă a articolului, fie o valoare specială prin care să se marcheze terminarea articolului.

Conceptul de fişier abstract

- Un fişier `f` poate fi privit ca o funcţie parţial definită `f:N→T`, unde `N` este mulţimea numerelor naturale, iar `T` este mulţimea valorilor posibile pentru un anumit tip de dată deja definit.
- Un întreg `i` din `N` indică numărul de ordine al unui articol(octet) din fişier; primul articol are numărul de ordine `0`, al doilea are numărul de ordine `1` ş.a.m.d.; prin `f(i)` notăm valoarea(continutul) articolului al `i`-lea, adică ansamblul valorilor câmpurilor acestui articol.
- **Exemple:**

  - `T` poate fi mulţimea valorilor posibile pentru date reprezentate pe un octet şi atunci avem un fişier flux de octeţi,

  - `T` poate fi un tip de date şi atunci avem un fişier structurat ca un flux de articole;

  - dacă tipul de dată este la rândul lui un tip fişier, avem de-a face cu o **bază de date**.

Descriptorul de fişier

- este o structură de date gestionată de SGF în care sunt păstrate informaţii despre un anumit fişier. Această structură diferă de la un sistem de operare la altul, dar sunt unele informaţii conţinute comune:

  • **Numele extern(simbolic)** este un şir de caractere utilizat pentru a identifica un fişier în comenzile utilizatorilor sau în programe. Orice fişier se identifică prin perechea (`N`, `I`)**,** unde **N** reprezintă numele simbolic al fişierului iar **I** este un număr prin care descriptorul este reperat pe disc în mod direct. Prin N, sistemul de fişiere realizează legătura cu utilizatorul, iar prin **I** cu celelalte componente ale SO.

  • **Starea curentă** poate fi: **arhivat(**dacă el nu poate fi deschis fără a se efectua un număr de operaţii asupra sa); **închis(**el se află pe un suport magnetic on-line şi poate fi deschis într-un timp suficient de scurt); **deschis (**pentru citire, scriere, execuţie etc.), dacă ele este alocat unui proces care poate efectua una dintre operaţiile specificate).

  • **Partajare**. Un câmp care specifică faptul că mai multe procese pot utiliza în acelaşi timp fişierul respectiv, putând efectua una dintre operaţiile specificate mai sus.

- **Proprietar**. Conţine identificatorul asociat utilizatorului al cărui proces a creat fişierul. Sistemul permite ca dreptul de proprietate să fie transferat şi altor utilizatori.
- **Utilizator.** Conţine lista proceslor care la momentul respectiv au deschis fişierul.
- **Încuiat.** Un proces care permite sistemului de fişiere să „încuie ” fişierul atunci când acesta este deschis. De exemplu, dacă fişierul este partajat, numai procesul respectiv poate utiliza fişierul atâta timp cât acesta este încuiat.
- **Protecţie.** Conţine nişte fanioane care indică modul în care fişierul respectiv poate fi folosit de către diverse clase de utilizatori.
- **Lungime.** Numărul de octeţi(articole) conţinute în fişierul respectiv.
- **Data creării (data ultimei modificări, data ultimei accesări)**. Conţine datele sistem ale creării, ultimei modificări, respectiv ultimei accesări a fişierului repectiv.
- **Contorul de referenţiere.** Dacă fişierul este partajat, el conţine numărul de procese care au deschis fişierul respectiv simultan.
- **Detalii privind modul de stocare.** Indică modul cum blocurile fişierului pot fi accesate, în funcţie de modul de administrare a unităţii respective.

Clasificarea fişierelor

- Se poate face după mai multe criterii:
- I. **După lungimea unui articol,** se disting:
  - fişiere cu articole de **format fix** (inreg. au lung. constanta);
  - fişiere cu articole de **format variabil** (inreg. au lung. variabila);**.**
- II. **După posibilitatea de afişare sau tipărire a conţinutului**
- **Fişierele text:**conţinutul lor poate fi afişat pe ecran sau poate fi tipărit la imprimantă. El este format dintr-o succesiune de octeţi, fiecare conţinând codul unui caracter tipăribil. **Articolul** unui fisier text este **caracterul**. Într-un fişier text, o linie este un şir de caractere, care se termină cu un caracter special numit **separator de linii**.
- **Fişierele binare:** şiruri de octeţi consecutivi fără nici o semnificaţie pentru afişare. De exemplu, fişierele obiect rezultate în urma compilării şi fişierele executabile rezultate în urma editării de legături sunt fişiere binare.

- III. După **suportul pe care este rezident** fişierul(hard disc; floppy disc; compact disc; USB; imprimantă; tastatură; monitor.)
- Evident, că nu toate aceste tipuri de fişiere acceptă orice operaţii şi moduri de acces. De exemplu, numai suportul disc acceptă accesul direct, celelalte numai cel secvenţial. Fişierele de la tastatură acceptă numai citirea iar fişierele pe imprimantă numai scrierea.
- IV. După modurile de acces permise de către un fişier se disting :

  • **fişiere secvenţiale** care permit numai accesul secvenţial;

  • **fişiere cu acces direct** permit accesul direct cel puţin la citire.

Operaţii asupra fişierelor

- **Deschiderea fişierului** poate fi realizată **explicit**, printr-un apel de sistem(**open)** sau **implicit** la prima referire a fişierului**.**
- Sistemul de operare păstrează o **tabelă** a fişierelor deschise şi fiecărui fişier deschis îi corespunde un indice în această tabelă.
- Când este lansată o operaţie asupra fişierului, acesta este selectat pe baza valorii indicelui din tabelă, evitându-se astfel efectuarea unor operaţii de căutare mult mai complexe.
- Operaţia open returnează un pointer către intrarea corespunzătoare din tabela fişierelor deschise.
- Acest pointer va fi utilizat în toate operaţiile de I/O care se vor efectua asupra fişierului.
- Efectul operaţiei open este diferit în funcţie de natura fişierului. Există acţiuni comune, indiferent de tipul fişierului. **Atât pentru un fişier nou creat, cât şi unul existent**:

  - se face legătura dintre identificatorul logic, utilizat de program şi descriptorul de fişier aflat pe disc;

  - se face alocarea de memorie internă pentru zonele tampon necesare accesului la fişier;

  - se încarcă rutinele de acces la articole;

  -sunt efectuate o serie de controale asupra drepturilor de acces ale utilizatorului la fişier.
- În plus**, pentru un fişier nou creat** se alocă spaţiu pe disc pentru memorarea viitoarelor articole ale fişierului.

- **Închiderea fişierului** se poate realiza **implicit,** la terminarea execuţiei procesului **sau explicit** prin lansare unui apel de sistem **(close)**.
- Şi aici se poate discuta efectul operaţiei de închidere, în funcţie de tipul fisierului.
- Pentru **fişierele temporare** se şterge fişierul şi se eliberează spaţiul pe disc ocupat.
- P**entru toate fişierele care trebuie reţinute,** se efectuează următoarele acţiuni:

  - se actualizează informaţiile generale despre fişier (lungime, adrese de început şi de sfârşit, data modificării etc.);

  - se aduce capul de citire al discului în poziţia zero;

  - se eliberează spaţiile ocupate de zonele tampon în memoria operativă;

  - se eliberează perifericul sau perifericele suport ale fişierului.
- În plus**, pentru fişierele nou create şi care trebuie reţinute**, se efectuează operaţiile:

  - se creează o nouă intrare în directorul discului (concept pe care îl vom aborda mai târziu);

  - se inserează marcajul EOF (sfrsit de fisier) după ultimul articol al fişierului;

  - se goleşte tamponul adică ultimele informaţii existente în zonele tampon, sunt transferate pe periferic.
- Dacă fişierul a fost modificat, se marchează acest lucru şi eventual se pune numele fişierului într-o listă a sistemului, în vederea unei salvări automate de către SO a tuturor fişierelor modificate.

- **Observaţie.** Într-un mediu **multiusers** mai mulţi utilizatori pot deschide acelaşi fişier în acelaşi timp; deci implementarea operaţiilor **open** şi **close** devine mult mai complicată. Pentru rezolvarea acestei probleme, sistemul de operare foloseşte o tabelă pe **două niveluri**: o **tabelă la nivel de proces** şi una la **nivel de sistem**.
- Tabela asociată procesului ţine evidenţa tuturor fişierelor deschise de procesul respectiv şi aici sunt stocate informaţii care sunt utilizate numai de proces în lucrul cu fişierul respectiv (drepturile de acces la fişier, valoarea pointerului care indică următorul articol ce poate fi citit/scris etc.).
- În tabela de sistem, sunt memorate informaţii care privesc partajarea fişierului respectiv de către mai multe procese. Sistemul administrează un **cont de deschidere**, care este iniţializat cu 1, când un proces deschide fişierul, este incrementat de fiecare dată când un alt proces deschide acelaşi fişier, respectiv decrementat, atunci când un proces execută un apel **close** asupra fişierului respectiv. Atunci când contorul devine 0, înseamnă că fişierul nu mai este folosit de nici un proces, deci intrarea corespunzătoare lui va fi ştearsă din tabelă.

- Scrierea(Write) înseamnă adăugarea unui articol la fişier. Daca x este valoarea articolului care trebuie introdus, fişierul f cu n articole se transformă într-un fişier f1 cu n+1 articole, definit astfel:

$$f1(i) = f(i), daca\ 0 \le i \le n$$

$$f1(n+1) = x$$

  Citirea(Read) a articolului k din fişier înseamnă obţinerea valorii f (k); k trebuie să fie valoarea indicatorului de fişier.
- Inserarea(Insert) a unui nou articol cu valoarea x, după articolul cu numărul de ordine k înseamnă obţinerea unui nou fişier f1 cu n+1 articole, definit astfel:

$$f1(i) = f(i), daca\ \ 0 \le i \le k$$

$$f1(k+1) = x$$

$$f1(i+1) = f(i), daca\ \ k \le i \le n$$

- **Obs.** 1.Operaţia de scriere este echivalentă cu operaţia de inserare dupa ultima poziţie.
- 2. Inserarea după poziţia 0, înseamnă plasarea unui articol nou la începutul fişierului.
- 3. **Inserarea** unui articol nu trebuie făcută în aşanu trebuie sa mute fizic celelalte. Noile articolele sunt legate de celelalte printr-o listă înlănţuită. Două art. pot fi vecine logic, chiar dacă ele ocupă spaţii fizice îndepărtate.

- **Ştergerea(Delete)** articolului k înseamnă obţinerea unui nou fişier f1 cu n-1 articole, definit astfel :

$$f1(i) = f(i), \text{daca } 1 \le i \le k-1$$
$$f1(i-1) = f(i), \text{daca } k+1 \le i \le n$$

- **Ştergerea** poate fi **logică** sau **fizică**. Prin ştergerea logică articolul respectiv continuă să ocupe zona fizică. Pentru aceasta, sistemul de fişiere adaugă la fiecare articol câte un octet numit **indicator de ştergere.** Dacă valoarea lui este 1, înseamnă că articolul există pentru sistemul de fişiere, în caz contrar el este ignorat de sistem, considerându-l şters. Pentru ştergerea fizică, trebuie recreat fişierul.
- **Crearea unui fişier**. Pentru fişierele cu organizare mai simplă, această operaţie coincide cu deschiderea unui fişier nou în vederea scrierii în el. Pentru fişierele cu organizare mai complicată crearea înseamnă o formatare, care pregăteşte spaţiul fizic în vederea încărcării lui cu informaţii, cum este cazul bazelor de date.
- **Ştergerea unui fişier.** Se eliberează spaţiul fizic ocupat de articolele fişierului şi se elimină din director intrarea care conţine descriptorul fişierului respectiv.

Blocarea şi deblocarea articolelor

- Ptr. prelucrarea inf. dintr-un fişier, trebuie efectuate citiri, prin care acestea sunt aduse în memoria internă.
- De asemenea, pentru transferul informaţiilor într-un fişier trebuie efectuate scrieri.
- Cele două operaţii sunt costisitoare din punctul de vedere al sistemului. Deci, este indicat ca printr-o op.de citire/scriere să fie aduse în mem. internă, resp. transferate pe disc, nu un articol ci mai multe articole.
- Un disc este divizat în blocuri fizice, de dimensiune const.
- Acestea sunt compuse dintr-un număr întreg de sectoare vecine. Mai mult, aceste sectoare trebuie să fie în întregime cuprinse într-o pistă sau într-un cilindru. Fiecare SO are modalităţi proprii de stabilire a dimensiunii blocului.
- Un astfel de bloc fizic, poate conţine un număr de articole ale fişierului. Aceste articole formeaza un **bloc** logic. Un fişier se poate diviza astfel în **blocuri** logice**.**
- Blocul logic este unitatea de schimb între suportul fişierului şi memoria internă.
- Zona de memorie în care este adus un bloc pentru a fi prelucrat, resp. unde sunt păstrate inf. care urmează să fie transf. în fişiere, se numeşte **zonă tampon** (buffer).
-

- **Observaţii.**

  1. Citirile/scrierile economisite sunt înlocuite cu mutări ale unor octeţi dintr-o zonă de memorie în alta.

  2. Prin blocarea articolelor poate apare fenomenul de **fragmentare internă** al discului, datorită cerinţei ca într-un bloc să intre un număr întreg de articole.

  3. În cazul unui fişier în care se scriu articole noi, este obligatorie închiderea acestuia, operaţie care va transfera şi informaţiile transferate în zona tampon, după ultima operaţie de scriere.

Moduri de organizare a fişierelor

- **Tipuri de acces la articole**. Să presupunem că există un fişier `f` şi se efectuează numai operaţii de citire asupra lui.
- **Accesul secvenţial** la un articol `f(i)`, presupune realizarea în prealabil a `i-1` accese la articolele cu numerele de ordine `1,2,…,i-1`.
- **Accesul direct(random access)** presupune existenţa unui mecanism de obţinere a articolului căutat, fără parcurgerea prealabilă a tuturor articolelor care îl preced. Mecanismele de acces direct sunt:

  • Acces direct prin **număr de poziţie (adresă);** are loc atunci când se furnizează sistemului de fişiere o valoare i şi aceasta întoarce o valoare f(i) a articolului reaspectiv.

  • Acces direct prin **conţinut;** are loc atunci când se furnizează SO o cheie `(a,v)` (o valoare `v` pentru un atribut `a`) şi aceasta întoarce acel articol `i` pentru care are loc relaţia `f(i).a=v`, adică atributul **a** al articolului are valoarea căutată v.
- **Organizarea secvenţială** presupune că între articolele fişierului este stabilită o relaţie de ordine: un articol X ”urmează” altui articol Y, dacă X a fost introdus în fişier după introducerea lui Y.
- Formatul articolelor la un fişier secvenţial poate fi **fix** sau **variabil.** Accesul la un articol al unui fişier secvenţial se face, de regulă, tot secvenţial.

- **Fişiere cu acces direct prin poziţie**. Un astfel de fişier are articole de format fix şi el este stocat pe hard-disc.
- Operaţiile de citire-scriere trebuie să conţină, ca parametru numărul blocului. Acest număr este furnizat de către utilizator sistemului de operare şi este un indice relativ la începutul fişierului, primul bloc relativ este `0`, al doilea este `1`, ş.a.m.d.
- De asemenea, presupunem că articolele sunt plasate pe suport unul după celălalt, chiar dacă un articol începe pe un sector şi se termină pe altul.
- **Observaţii.**
- 1. În cazul formatului variabil, această schemă nu mai funcţionează. Există însă diverse metode de acces direct şi în acest caz, bazate fie pe o listă a adreselor de pe disc unde începe fiecare articol, fie pe lista lungimilor acestor articole.
- 2. Majoritatea implementărilor de limbaje de programare de nivel înalt au mecanisme de acces direct prin poziţie pentru articole de format fix (funcţia **seek**).

Organizarea secvenţial indexată a fişierelor

- se bazează pe conceptul **index**; **indexul** este un **atribut** cu valori unice pentru fiecare articol. Deci, această metodă este caracterizată de accesul prin conţinut.
- Articolele sunt scrise pe suport în acces secvenţial şi sunt grupate în blocuri logice de informaţie(**pagini),** astfel încât valorile indexului tuturor articolelor dintr-o pagină sunt mai mari sau mai mici decât valorile indexului articolelor dintr-o altă pagină.
- Odată cu crearea fişierului, se creează şi o **tabelă de indecşi**. Pentru fiecare pagină, în această tabelă, se memorează adresa de pe disc a paginii şi valoarea maximă a indecşilor din pagină.
- Scopul acestui mod de organizare, este de **a înlocui căutarea secvenţială** a articolelor, cu **căutarea binară** sau **arborescentă**.
- Prin aceste metode de căutare se determină pagina în care se găseşte articolul respectiv, după care se efectuează o căutare secvenţială în pagina respectivă.
- In funcţie de dimensiunea fişierului, tabela de indecşi poate fi o listă liniară sau o structură arborescentă.

- Distingem următoarele cazuri:
- i) Fişierul este memorat pe mai multe volume(partiţii) de disc; în acest caz, vom avea o tabelă organizată pe trei niveluri.
- **Nivelul 3(fişier)**, are atâtea intrări câte volume disc ocupă fişierul şi fiecare intrare conţine: **Ultimul index din volum** şi **Numărul volumului**.
- Tabela este plasată pe ultimul volum al fişierului.
- **Nivelul 2(volum)**, are atâtea intrări câţi cilindri sunt alocaţi fişierului în volumul respectiv şi fiecare intrare conţine: **Ultimul index din cilindru** şi **Numărul cilindrului**.
- Tabela este plasată pe ultimul cilindru alocat fişierului în volumul respectiv.
- **Nivelul 1(cilindru),** are atâtea intrări câte pagini din fişier există pe cilindrul respectiv şi fiecare intrare conţine: **Ultimul index din pagină** şi **Adresa paginii**.
- Tabela este plasată pe ultimele sectoare ale cilindrului.
- ii) Fişierul este memorat pe un singur volum de disc; în acest caz, vom avea o tabelă organizată pe două niveluri, care au aceeaşi descriere cu nivelurile 1 şi 2 descrise anterior.
- iii) Fişierul este memorat pe un singur cilindru; în acest caz, vom avea o tabelă ce reprezintă o listă.

- **Obs.** Pentru căutarea unui articol, numărul de accesări ale discului depinde de dimensiunea fişierului şi de locul unde este memorată tabela respectivă. De **exemplu**, dacă fişierul este multivolum şi tabela de nivel 3 este încărcată în memoria internă, atunci pentru găsirea paginii unde se află articolul respectiv sunt necesare 5 accesări ale discului.
- Pentru **actualizarea** unui fişier secvenţial – indexat, spaţiul fizic alocat unei pagini logice este divizat în două părţi: **partea principală** şi **partea de depăşire.**
- Articolele din partea de depăşire sunt organizate ca o listă înlănţuită**.**
- Astfel, pentru **inserarea** unui articol, dacă coresp. cheii respective, articolul nu mai are loc în partea princ., el va fi introdus în partea de depăşire.
- Dacă ar exista un spaţiu fizic de dimensiune fixă, coresp. unei pagini logice, atunci ar exista posibilitatea ca articolul cu cheia respectivă să nu mai aibă loc în pagina fizică unde ar trebui introdus, deci ar trebui reorganizată tabela de indecşi.
- Dacă se fac mai multe înserări între două chei vecine din partea principală, atunci lista leagă mai multe articole în partea de depăşire, iar randamentul accesului la disc scade considerabil.

- Prin **ştergerile de articole** fie că rămâne un loc nefolosit în partea princ., fie trebuie reorganizată o listă simplu înlănţuită în partea de depăşire.
- Rezultă necesitatea **reorganizării periodice** a unui astfel de fişier (rescrierea întregului fişier; eliminarea porţiunilor şterse; articolele din partea de depăşire trec în partea princ.; se reface tabela de indecşi).
- Acest gen de fişiere prezintă două mari dezavantaje:

  - **necesitatea reorganizării** lor destul de frecvente;

  - necesitatea **efectuării de două până la cinci accese la disc** pentru accesarea unui articol.

Organizarea selectivă a fişierelor

- Se presupune că articolele fişierelui conţin un **index**. Fie T tipul de date al indexului.
- Articolele fişierului se împart în n **clase de sinonimie**. Clasa de sinonimie căreia îi aparţine articolului, se obţine pe baza unei funcţi parţial definite: $f : T \rightarrow \{0, 1, \ldots, n-1\}$ numită **funcţie de regăsire (randomizare).**
- Toate articolele pentru care se obţine aceeaşi valoare a funcţiei de rand., fac parte din aceeaşi clasă de **sinonimie.**
- Deci, organizarea selectivă foloseşte **accesul direct prin conţinut**. O **grupare logică** de articole(clasă de sinonimie) trebuie să coresp. unui bloc fizic de stocare pe disc.
- Fişierele selective sunt o replică la fişierele indexat secvenţiale, în sensul următor: în timp ce pentru fişierele indexat–secvenţiale căutarea paginii în care se află articolul respectiv se face prin tabelele de index, în cazul celor selective, pagina este clasa de sinonimie, care se determină aplicând funcţia de regăsire, ce reprezintă un calcul efectuat de CPU, deci mult mai rapid.
- Dezavantajul acestei metode, constă în faptul că definirea funcţiei de regăsire este foarte puternic dependentă de datele ce vor fi înmagazinate în fişierul respectiv.

- Actualizarea fişierelor selective se realizează într-un mod analog celui al fişierelor indexat-secvenţiale.
- **Scrierea** se realizează astfel: Se aplică funcţia de randomizare valorii indexului şi se obţine numărul clasei(de sinonime) în care se află articolul.
- **Partea principală** a fişierului este formată din câte o pagină pentru fiecare clasă. Dacă pagina clasei respective nu este încă plină, atunci articolul se depune în pagina respectivă pe disc. Dacă pagina din partea principală este plină, atunci articolul este pus în **partea de depăşire**. Ca şi la fişierele secvenţial–indexate, acest articol este legat printr-o listă simplu înlănţuită de ultimul articol sinonim cu el din partea principală.
- La **citire**, utilizatorul furnizează indexul articolului dorit a fi citit. Sistemul de fişiere îi aplică acestuia funcţia de randomizare, depistând clasa de sinonimie din care face parte. În cadrul clasei, căutarea articolului se face secvenţial, mai întâi în partea principală şi apoi în partea de depăşire.

. **Exemple.**.

1.Să presupunem că se doreşte crearea unui fişier selectiv pentru un colectiv de 2000 de persoane. Fiecare persoană are un cod unic de identificare, exprimat printr-un număr între 1 şi 2500. Presupunem că se foloseşte drept suport un disc cu 10 piste pe cilindru, iar pe o pistă încap 20 de articole. Vom alege drept pagină fizică o pistă, deci aproximativ 20 de articole într-o clasă de sinonimie, numărul claselor de sinonimie n=100, iar funcţia de randomizare este f(x)=x (mod 100). Dacă, valorile indexului sunt 100, 200,.., 2000, 2100, 2200, 2300, 2400, 2500, înseamnă că toate aceste articole sunt sinonime(fac parte din clasa 0) şi sunt în număr de 25. Deci această clasă va avea 5 articole plasate în partea de depăşire, număr care este destul de mare mare şi care poate creşte dacă sunt adăugate articole în fişier al căror indice să fie multiplu de 100.

2. Presupunem că fişierul are 10000 de articole, însă indexul se reprezintă printr-un cod de 20 de cifre. S-ar impune o împărţire în 500 de clase, însă de această dată delimitarea superioară a părţii de depăşire pentru o clasă practic nu se poate face. Este posibil ca toate cele 10000 de articole să fie sinonime, deci căutarea s-ar face într-o singură clasă, adică am avea de fapt un fişier secvenţial.

- **Observaţie**. Din cele două exemple prezentate, rezultă că eficienţa căutării unui articol într-un fişier selectiv este dependentă de datele pe care le conţine.