Gestiunea fisierelor

Modalități de acces

Accesul secvențial-indexat

Fișierele care permit accesul secvențial-indexat sunt cele cu format fix.

Ca fapt divers, Cobol are implementat accesul secvențial-indexat.

Un exemplu de bibliotecă de funcții ce permite accesul secvențial-indexat a fost dezvoltată de Informix (IBM Informix acum) este C-ISAM.

NOTA: Urmatoarele apeluri nu trebuiesc știute, ele au fost doar discutate superficial. Pentru mai multe informații: http://50001.com/db/informixside/data/cisam.pdf

- **isbuild**

- printre parametrii numele fișierului și lungimea înregistrării
- creează fișierele xxx.dat și xxx.idx

- **isaddindex**

- adaugă un index în fișier
- câmpurile $c_1,...,c_n$ specificate nu trebuie să fie în ordine; creează indecși pentru fiecare înregistrare
- creează un arbore B (de căutare, dar nu binar)

- **isdelindex**

- ștergere index

- **isopen**

- deschidere fișier

- **isread**

- se specifică indexul și valoarea. (valoarea poate fi ISNEXT, caz în care se referă la următoarea înregistrare)

- **iswrite**

- scriere la sfârșit

- **isrewrite**

- modificare înregistrare curentă

- **isdelete**

- ștergere fișier

Ideea este de a avea cât mai mulți indecși posibil pentru a putea căuta rapid după orice criteriu. Pentru n câmpuri putem aveam maxim n! Indecși. Numărul mare (maxim) de indecși dezavantajează operațiile de scriere și ștergere (care provoacă actualizarea arborilor B). Suprascrierile nu sunt penalizante dacă câmpurile respective nu fac parte din nicio cheie.

Noțiunea de director (catalog)

Fișierele sunt grupate în directoare/cataloage. În majoritatea sistemelor de operare structura fișierelor/directoarelor este arborescentă.

Organizarea discului

În Windows:

- FAT (File Allocation System) (cu tabele de legături)
- NTFS (NT File System)

O diferență între UNIX și Windows ar fi că în UNIX există o singură arborescență de directoare, pe când în Windows fiecare partiție are propria ei structură arborescentă.

În UNIX există un disc0, de dimensiune mică, unde se află root.
Cu comanda **mount** se creează un director nou frunză și se declară că toate subdirectoarele sale vor fi pe discul nou creat, de ex. disc1. Cu **mount** se poate declara ca toată arborescența unui director să fie (de) pe altă mașină. (NFS – Network File System)

Fiecare disc posedă o tabelă de inoduri. La crearea unui fișier, informația se pune pe disc și apare o nouă intrare în tabela de inoduri.

Informatiile conținute de inoduri

- **pointer către zona de date**, adică, către adresa fizică de pe disc unde se află informația (de fapt este vorba de o tabelă de 8 pointeri, primul către începutul zonei de date, apoi în funcție de lungimea informației sunt folosiți si următorii 6; dacă mai este nevoie de încă un pointer atunci al 8-lea pointer din tabel reprezintă adresa unui alt astfel de tabel; etc.)
- **număr de inod**, acesta este dat la creare și este egal cu numărul de disc + numărul de inod. (cum fiecare disc are propria tabelă)
- **tipul fișierului**
  - obișnuit (regular file)
  - directoare
  - periferice de tip caracter
  - periferice de tip bloc
  - link simbolic
  - pipeuri
  - socketuri (fișiere de comunicare bidirecțională)
- **numărul de legături fizice** (în câte directoare e declarat)
  Ex.: ln x y - în director apare o intrare y corespunzătoare aceluiaşi număre de inod al lui x (are două legături fizice din momentul acesta); pot fi declarate cu acelaşi nume, dar atunci în directoare diferite; practic comanda **ln** incrementează numărul de legături fizice, la polul opus se află comanda **rm** care îl decrementează atunci când şterge o intrare dintr-un director. Observăm că un fişier se consideră şters atunci când numărul de legături fizice este 0 şi niciun proces nu-l mai are deschis.

Fişierele speciale de tip director

- fiecare înregistrare conține două câmpuri:
  - numele intrării pe director (unic)
  - numărul de disc + numărul de inod
- se creează prin comanda **mkdir**
- orice director conține la creare două intrări :
  - **.** – numărul de inod propriu
  - **..** – numărul de inod al părintelui
- aceste două intrări sunt obligătorii şi nu pot fi şterse
- intrările cu **.** sunt ascunse la comanda **ls** ; pentru a le vedea folosim **ls –a**
- ştergerea se face cu comanda **rmdir** ; aceasta nu reuşeşte decât dacă directorul este gol (directorul se consideră gol dacă tabela nu conține decât intrările **.** şi **..**)
- comanda **rm –r** şterge recursiv directorul chiar dacă nu este gol
- **proprietarul** (un utilizator, care nu trebuie să aparțină neapărat grupului proprietar)
- **grupul proprietar** (observăm că grupurile nu sunt neapărat disjuncte)
- **drepturi de acces**
  - pot fi modificate de proprietar
  - 9 biți împărțiți in trei secvențe de forma rwx (read,write,execute) pentru proprietar, grupul proprietar şi ceilalți utilizatori.
    Ex. : rwxrwxrwx – toată lumea are toate drepturile
  - absența unuia dintre drepturi se marchează prin **–**
    Ex. : rwxr-xrwx – grupul proprietar nu are drept de scriere
  - mai există 3 biți speciali : **set-uid, set-gid, sticky bit**; primele două pot fi **s** sau **S** (cu sau fără drept de execuție) pe bitul 3, respectiv bitul 6; sticky bit aplicat fişierelor de tip executabil le menține în zona de memorie chiar şi după terminarea programului.
    Ex. (pentru set-uid) : Ştim că parolele sunt reținute în /etc/passwd , director al cărui proprietar este root şi care în drepturile de acces are sigur setate : r-s--x--x ; atunci când un utilizator doreşte să-şi schimbe parola, proprietarul real nu mai coincide cu proprietarul efectiv, cum proprietarul procesului este root şi proprietarul real devine proprietar al procesului, ceea ce îi permite să-şi schimbe propria parolă.
    Remarcăm de asemenea, că **x** are efect asupra executabilelor, fişierelor text de tip scripturi shell, pentru directoare semnifică dreptul de a face change directory.

S-a mai amintit de comenzile :

df
chown
rm –r /*
rm –rf /*

Întrebare : Cum se şterge **rm** ?