Tema nr. 10

Sistemul de fisiere

Studenti implicati in redactare :
Brîncoveanu Andrei (cap. 1) – coordonator
Popescu Razvan (cap. 2)
Neacsu Mihaela (cap. 3)
Dinisor Ovidiu (cap.4)
Stoica Oana (cap.5)

- 2008 –

CUPRINS

Capitolul 1
Concepte fundamentale : exemple si comparatii Linux si Windows

1.1Definitii.Concepte

Fisierul este o structura de date ce include un volum mare de informatii procesate omogen. Este o forma standardizata, este una din abstractiile fundamentale în lumea sistemelor de operare, un fisier este folosit pentru a stoca informatiile necesare functionarii sistemului de operare si interactiunii cu utilizatorul.

Un sistem de fisiere este un mod de organizare a fisierelor si prezentare a acestora utilizatorului. Din punct de vedere al utilizatorului un sistem de fisiere are o structura ierarhica de fisiere si directoare, începand cu un director radacina. Localizarea unui fisier în sistemul de fisiere se realizeaza cu ajutorul unei cai în care sunt prezentate toate intrările de pana atunci.

1.2 Reprezentarea partitiilor

Sistemul de fisiere adoptat de SO Linux este diferit fata de SO Windows. In Linux exista o singura structura de directoare. Totul incepe din directorul radacina(root), reprezentat de simbolul "/", dupa care se extinde in sub-directoare. In Windows acest lucru este tratat diferit : utilizatorul poate avea mai multe resurse tip disc , notate diferit A:-Z („drive letter"), A,B pentru FD:, fiecare fiind radacina a unui arbore separat. Linux plaseaza toate partitiile sub directorul radacina (root) in care se „mounteaza" sub denumirea unor directoare. Cel mai aproape de „radacina" in Windows este partitia „c:". Partitia reprezinta o fractiune din HDD-ul unui calculator a carei marime o selecteaza utilizatorul, aceasta fiind cuprinsa intre 1-100% din marimea hard-diskului. Un calculator poate avea una sau mai multe partitii in functie de necesitatile utilizatorului.

Sub Windows, diversele partitii existente sunt detectate la bootare si li se ofera o litera din alabet (C-Z). Sub Linux, daca nu se mounteaza o partitie sau un dispozitiv, sistemul nu stie de existenta acestora.Nu este o metoda usoara de acces pentru incepatori, dar ofera o mare flexibilitate in sensul ca se pot ascunde anumite portiuni din HDD sau diverse componente in functie de necesitatile utilizatorului, e un plus si pentru securitate si ergonomie. Acest mod de abordare gen sistem de fisiere unificat al Linuxului mai are si

alte avantaje. De exemplu directorul „/usr” care contine majoritatea fisierelor executabile. Linux ne permite sa mountam acest director pe alta partitie sau chiar si pe alt calculator din retea. Sistemul nu va detecta nici o incompabilitate, deoarece /usr va aparea ca un director local. In Windows acest lucru nu este posibil, de exemplu mutarea directorului „Program Files” pe alta partitie nu pastreaza o functionalitate totala.

Un lucru minor la prima vedere, dar piesa importanta in functionabilitate este faptul ca Linux foleste slash „/” pentru despartirea cailor catre diverse fisiere spre deosebire de Windows care foloseste back-salsh „\”. Acesta reacomodare poate fi destul de dificila pentru un obisnuit cu Windows, durand ceva vreme pana la acomodare. Mai mult, Linux este un sistem „case sensitive” adica face diferenta intre litere mari si litere mici spre deosebire de Windows la care nu conteaza daca denumire fisierului este introdusa cu litere mici sau litere mari.

Sub Linux fisierele sunt identificate prin i-number(index dintr-un sir de i-noduri). Fiecare fişier are un singur ***i-node*** care conţine :

1. identificatorul utilizatorului ce este proprietarul fişierului;
2. tipul fişierului (obişnuit, director, pipe sau special);
3. drepturile de acces;
4. timpul ultimului acces şi al ultimei modificări, data şi ora ultimei modificări efectuate asupra i-node-ului;
5. numărul de legături (a se vedea comanda unlink);
6. adresele sectoarelor de pe HDD ce conţin datele fişierului;
7. lungimea fişierului în octeţi.

http://www.yolinux.com/TUTORIALS/LinuxClustersAndFileSystems.html

1.3 Sisteme de fisiere Windows (FAT&NTFS)

Sistemul de fisiere FAT (File Allocation Table) a devenit faimos multumita implementarii in sistemele de operare Microsoft. Acest sistem de fisiere avea si are multe defecte. Versiunile initiale (FAT12 si FAT 16) suportau o dimensiune limitata pentru hard-disc-uri(250Mb pentru FAT12 si 4Gb pentru FAT16) si utilizau doar 11 caractere pentru numele fisierelor (opt pentru nume si trei pentru extensie). Versiunea FAT32 a rezolvat problema cu discurile marind limita la 2TB, în timp ce cu introducerea vFAT s-a extins limita pentru numele fisierelor pana la 255 de caractere. In afara de aceasta o partitie FAT, datorita naturii sale, are nevoie din când in când de cate o defragmentare, deoarece blocurile ce compun un fisier pot fi

împrastiate pe disc încetinind semnificativ operatiunile de citire. Avantajul sistemului de fisiere FAT consta in faptul ca este atât de raspândit încât este suportat practic de aproape toate sistemele de operare. Utilizarea Linux-ului în combinatie cu o partitie FAT poate fi luata uneori în considerare deoarece poate ajuta la schimbul de date intre Linux si Windows sau mentinerea unor versiuni mai vechi de Windows. Exista chiar si un sistem de fisiere UMSDOS, care, fiind o varianta de FAT, permite instalarea (pe aceeasi partitie si fara drivere suplimentare) a unui sistem Windows cât si a unuia Linux (chiar daca acesta va avea prestatii ceva mai modeste fata de un sistem de fisiere nativ).

NTFS (NT File System) este un sistem de fisiere dezvoltat pentru Windows NT si apoi imbunatatit pentru Windows 2000,XP si Vista.NTFS4 este folosit la Windows NT, in timp ce pentru Win 2000 si XP se foloseste versiunea NTFS5 putin imbunatatita. Marile diferente dintre NTFS4 si NTFS5 ar fi :

- Encriptarea fisierelor si directoarelor
- Posibilitatea limitarii accesului diversilor useri la partitii impunandu-se o anumita marime a spatiului pe disk disponibil
- Recovery Console
- Dynamic Volume Management

http://arstechnica.com/paedia/n/ntfs/ntfs4-1.html
http://examnotes.net/forums/showthread.php?s=&threadid=59955&highlight=differences

1.4 Caracteristici NTFS

Ca facilitati ale acestui sistem de fisiere pot enumera :

- Foloseste adrese de disc de 64 de biti si poate suporta partitii de pana la 2^{64} bytes;
- Ofera posibilitatea folosirii caracterelor Unicode in numele de fisiere. Unicode este un standard de codificare ai interpretare a datelor binare în format text, incluzand in varianta finala toate caracterele folosite în orice limba. Este proiectat pentru ca oricarui caracter din orice limba, de pe orice platforma, sau program, sa îi corespunda un singur numar.
- Permite folosirea numelor de fisiere de pana la 255 de caractere (inclusiv spatii si puncte);
- Permite indexarea generala a fisierelor;

- Ofera posibilitatea managementului dinamic al sectoarelor;
- Folosindu-se de POSIX (Portable Operating System Interface), permite crearea de „hard-link-uri”, face distinctie intre litere mari si mici in cadrul numelor de fisiere si pastreaza informatii de timp referitoare la fisier. POSIX este destinat managementului API(Application Programming Interface)
- Permite utilizarea fisierelor cu seturi multiple de date.

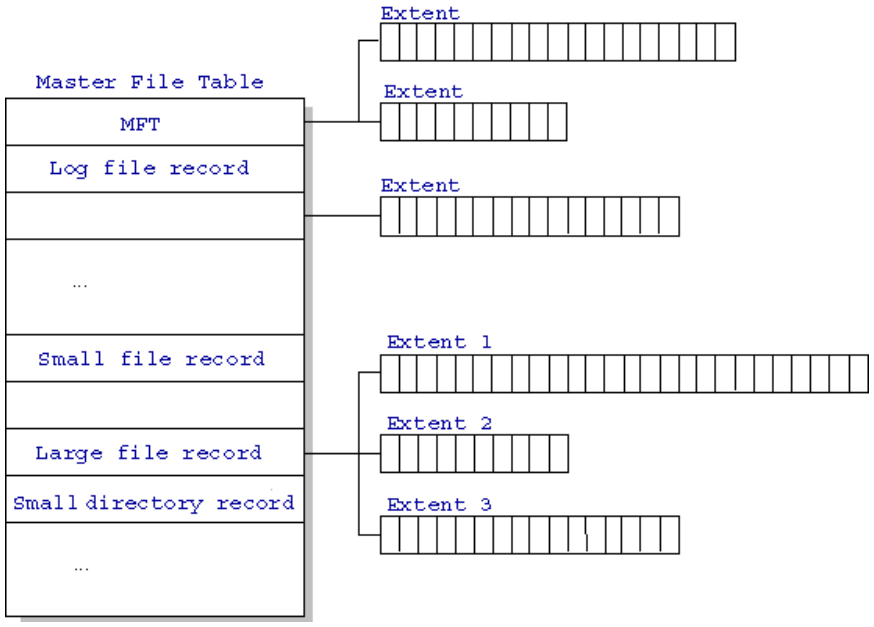

Fig 1.1 Structura MFT

1.5 MBR si MFT

MBR = „***M**aster **B**oot **R**ecord”,*este un program mic care se executa odata cu bootarea sistemului. De obicei, MBR-ul se afla pe primul sector al hard-diskului. Programul demareaza procesul de bootare uitandu-se in in tabelul de partitii pentru a decide ce partitie este folosita pentru bootare. Apoi transfera controlul programului in sectorul de boot al partitiei, care va continua procesul de boot. In DOS si Windows, MBR-ul se creaza folosind comanda FDISK /MBR. MBR-ul ocupa 512 byte.

http://www.ata-atapi.com/hiwmbr.htm

Atunci cand in Windows este formatata o partitie pentru prima oara in sistem NTFS se creaza o serie de fisiere sistem, printre care si Master File Table (MFT) care contine informatii despre toate fisierele si directoarele de

pe volumul in cauza. Prima informatie pe o partitie NTFS este Sectorul de Boot, care este sectorul 0 al partitiei si contine un cod de pornire al sistemului. Alte informatii necesare programului de boot-are pot fi inscrise in sectoarele 1-16. Primul fisier pe un volum NTFS este fisierul MFT. Pentru fiecare fisier de pe un volum NTFSexista cel putin o intrare in MFT. Toate informatiile despre un fisier (nume, dimensiune, data creare,etc) sunt pastrate in MFT.

Informatie standard	Numele fisierului sau directorului	Descriptor de securitate	Date sau Index	

Fig 1.2 MFT record pentru un fisier sau director mic

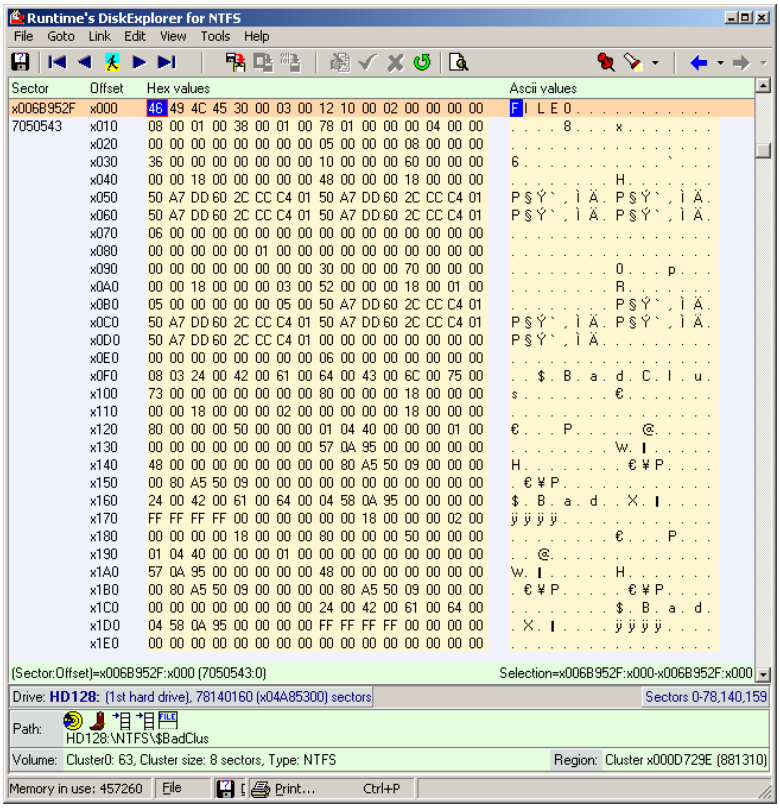

Fig 1.3 Structura grafica MFT

http://www.ntfs.com/ntfs-mft.htm
http://www.pcguide.com/ref/hdd/file/ntfs/archMFT-c.html
http://www.djkaty.com/drupal/files/pictures/BadClus.gif

1.6 Subdirectoare in Linux

Pentru a prezenta complexitatea sistemului de fisiere de sub Linux in continuare voi da cateva exemple de subdirectoare ale root-ului.

- /sbin – Acest director contine toate binarele esentiale functionarii sistemului de operare. Acesta include administrarea sistemului cat si problemele legate de configurarile hardware sau mentenanta. Aceste programe sunt esentiale oricarui utilizator.
- /bin – Spre deosebire de /sbin, acest director contine comenzi utile atat administratorului de sistem cat si utlizatorilor neprivilegiati(obisnuiti-similar Guest sau User in Windows). De obicei aici se gasesc comenzi precum bash, csh, cp, mv, rm, cat si ls.
- /boot – Acest director contine fisierul „system.map" cat si kernelul Linux. Boot-loaderul Lilo plaseaza aici back-up-urile sectorului de boot
- /dev – Acest director este important, el sublinieaza caracteristica principala a sistemului Linux – totul este un director sau un fisiser. O simpla navigare prin el scoate la iveala hda1, hda2, hda3 etc care reprezinta diversele partitii aflate pe drivul master. De exemplu /dev/cdrom sau /dev/fd0 reprezinta unitatea CD-Rom si floppy diskul.
- /home – Deoarece Linuxul este un sistem multi-user (similar Windows), fiecarui user i se asociaza un anumit director care paote fi accesat doar de acel user sau de administrator (gen Make Private Folder in Windows)
- /lib – Aici sunt incluse toate librariile necesate programelor de sistem. In Windows sunt echivalente cu librariile cu extensia DLL.
- /mnt – Acesta este un mount point generic unde se pot mounta device-uri sau filesystems. Mountarea este procedeul prin care un filesystem devine accesibil. Dupa mountare fisierele pot fi accesate din acest mount-point. De obicei aici se gasesc cai cate cd-rom sau floppy disk. Se pot crea si alte mount-points aici dupa necesitatile utilizatorului. Nu exista o limitare a numarului de mount-points.
- /tmp – Acest director contine fisierele de care este nevoie doar pe moment. Multe programe folosesc acest director pentru a pastra informatii temporare care pot fi sterse dupa un restart fara a aafecta functionarea corecta a lor.

- /var – Acest director contine „spooling data” precum e-mailuri si rezultatul printarilor. Logurile de sistem sunt si ele continute aici in calea /var/log/messages.

Linux Filesystem Hierarchy Standard by Mayank Sarup

Pentru a putea ajunge la folosire si configurarea acestor directoare, mai intai este nevoie ca HDD-ul sa fie partitionat corect si pregatit pentru instalare.

Sub Windows acest lucru era foarte facil, ne-existand o gama prea larga de optiuni. sa luam un caz particular : instalarea sistemului Windows XP. Dupa ce sistemul a bootat dupa cd/dvd-ul de instalare si incarcarea unor biblioteci de drivere, se ajunge in momentul partitionarii hdd-ului. Aici se poate alege in cate partitii poate fi impartit, marimea acestora si sistemul de fisiere folosit. Ca sistem de fisiere exista doar optiunea de FAT32(standard vechi Win98) sau NTFS(aparuta odata cu WinNT) care este tehnologie proprie Microsoft.

1.7 Sisteme de fisiere in Linux

Linuxul suporta 15 sisteme de fisiere : *ext, ext2, xia, minix, umsdos, msdos, vfat, proc, smb, ncp, iso9660, sysv, hpfs, affs si ufs.* Sub instalarea Linux exista insa mai multe sisteme de fisiere ce pot fi alese si confiugurate inca de la inceput.

Sistemul de fisiere ext2 – este un sistem traditional suficient de bun pentru a fi folosit si in viitor. Exista insa o versiune mai noua, ext3, mai rapida,care adauga facilitatea de *„journaling”*. Partitiile ext3 sunt mai putin sensibil la erori datorate resetarilor hardware, prin folosirea optiunii de journaling diminueaza nevoia de a rula „fschk” la fel de des ca pe o partitie ext2 deoarece modificarile aduse SO-ului sunt inregistrate in acelasi timp in care sunt operate. Tranzitia intre cele doua sisteme (ext2 si ext3) realizandu-se foarte rapid.

ReiserFS este un sistem de fisiere cu jornaling dezvoltat de echipa lui Hans Reiser. Acesta lucreaza folosind metadate particulare asociate fisierelor, ceea ce îi permite sa recupereze fisierele, dupa eventualele blocaje de sistem, cu o rapiditate si o fiabilitate superioara altor sisteme. Marele avantaj al acestui sistem de fisiere consta în faptul ca nu este legat de tehnologii anterioare precum Ext3, care este legat de Ext2. Iar unul dintre dezavantajele lui ReiserFS, fata de Ext3, este necesitatea de a face back-up si de a formata o partitie Ext2, în cazul în care se vrea conversia în ReiserFS.

Momentan este în faza de dezvoltare Reiser4, succesorul lui ReiserFS,rescris aproape de la zero; printre caracteristicile sale principale se remarca o viteza majora, un suport mai bun pentru gestionarea directoarelor de dimensiuni mari continând multe fisiere, îmbunatatirea sistemului de journaling, integrarea metadatelor in interiorul spatiului numelor fisierelor, suportul pentru plug-in si optimizarea dinamica a datelor.

Sistemul de fisiere JFS este un sistem de fisiere cu journaling dezvoltat de catre firma IBM utilizat initial pentru OS/2. A fost facuta si o versiune pentru AIX, iar în februarie 2000 a fost donat comunitatii Open Source. Printre caracteristicile specifice acestui sistem de fisiere se poate nota un sistem de journaling bazat pe o tehnica de logare conceputa pentru baze de date. Comparativ cu alte sisteme de fisiere, programele de reparare a partitiilor, nu trebuie sa faca altceva decât sa citeasca cele mai recente log-uri în loc de a citi metadatele tuturor fisierelor. O alta caracteristica interesanta este folosirea de blocuri de dimensiune variabila (de la 512 la 4096 byte), ceea ce are influenta asupra nivelului de fragmentare si asupra vitezei discului. Se introduce alocarea dinamica a inodului, o organizare diferita in functie de marimea directoarelor si o alocare particulara a spatiului fisierelor.

XFS este un sistem de fisiere cu journaling cu prestatii ridicate creat de SGI (Silicon Graphics Inc.) pentru Unix-ul implementat de ei, Irix. S-au facut câteva alegeri de implementare precum „Allocation Group” ce consta în divizarea discului în opt sau mai multe regiuni de dimensiuni egale fiecare autonoma în gestionarea spatiului in asa fel încât se poate lucra pe fiecare regiune simultan. Fata de Ext3 si ReiserFS, XFS utilizeaza un sistem de journaling mult mai sigur si performant, introducând o tehnica denumita „Delayed allocation” pentru alocarea mai rapida si inteligenta a fisierelor pe spatiul liber de pe disc.

http://www.maenad.net/geek/di8k-debian/node29.html

http://en.wikipedia.org/wiki/Ext3

1.8 Partitionare sub Linux

Programele de partitionare a hard-disk-ului sunt numeroase, dar in mare parte ele fac acelasi lucru, adica alocarea spatiului disponibil de pe un HDD in mai multe portiuni de dimensiuni mai mici decat capcitatea hard-disdkului. Aceasta impartire se face pentru o securitate mai buna a datelor,

in cazul virusarii unei portiuni de HDD nu este nevoie formatarea intregului continut, ci doar a partitiei pe care se afla zona afectata. Partitionarea mai are si rolul gestionarii mai usoare a fisierelor detinute, deoarece acestea sunt impartite in zone diferite. De asemenea performantele pot creste cand partitia este de o dimensiune mai mica. De exemplu cu RedHat se poate selecta sa se partitioneze automat hard-disk-ul sau manual. In Mandrake, avem la dispozitie DiskDrake, unul din cele mai bune instrumente de partitionat, care poate si redimensiona o partitie.

Discurile in Linux sunt afisate ca /dev/hdxx. Astfel, daca exista un hard-disk master(HDD-ul ce detine sectorul de boot) pe primul controler, atunci acesta va fi /dev/hda1, unde a1 inseamna prima partitie si primul disc.

Sub Linux este necesara crearea unei partitii de swap, care depinde de cantitatea de RAM disponibila. Astfel, pentru 256 MB o partitie de 200 MB ar trebui sa fie suficienta. Pentru mai putina memorie, o partitie de 500 MB este necesara. Pentru a beneficia la maxim de performatele sistemului, este ideal ca fiecare director important sa fie pus pe o partitie separata.

Bibliografie comuna

http://www.yolinux.com/TUTORIALS/LinuxClustersAndFileSystems.html
http://en.wikipedia.org/wiki/Ext3
www.linuxiso.org
www.reiserfs.org
freshmeat.net/projects/ext3
http://www.ntfs.com/ntfs-mft.htm
Curs Sisteme de Operare - 2008

Capitolul 2
Functiile principale Win32 Api

Acronimul API este o abreviere a Application Programming Interface. Aşadar Windows API (sau Win32 API) este un set de funcţii oferite de sistemul de operare Windows pentru manipularea resurselor calculatorului. Orice sistem de operare oferă (sau exportă) un set de astfel de functii, pentru a fi utilizate de programatori în dezvoltarea de aplicaţii specifice acelui sistem de operare. Denumirea de Win32 API mai este folosită uneori pentru a marca diferenta dintre sistemele de operare Windows pe 16 biti (Windows 3.X) şi sistemele de operare Windows pe 32 de biti (Windows 9X, Windows NT, Windows 2000, Windows XP). Din acest motiv ele sunt construite in mare parte pentru programatori. Programatorilor li s-a oferit multa flexibilitate si putere in dezvoltarea aplicatiilor. In aceasi timp aplicatiilor Windows li s-a impus mare responsabilitate in manipularea nivelelor inferioare.

Pe parcurs au fost facute multe modificari la sistemul de operare Windows si Interfetele API Windows au fost de asemenea schimbate pentru a tine pasul cu sistemul de operare. Interfetele API pentru Windows 1.0 au avut mai putin de 450 de functii, in comparatie cu API-uri moderne care au mii de functii. Avand asta in vedere, Microsoft a pus mare accent pentru compatibilitatea inversa, adica compatibilitatea API-urilor noi cu API-uri din urma. Pentru a oferi compatibilitate, Microsoft a scris, pentru noile versiuni pe 32 biti, o schema complexa de Interfete API care permiteau codului scris pe 32 biti sa apeleze cod scris pe 16 biti (si invers in unele cazuri limitate).

Aproape fiecare noua versiune a sistemului de operare Windows a introdus schimbari in Interfete API Windows. Numele a ramas consistent intre diferite versiuni. Denumirea de Win32 API mai este folosită uneori pentru a marca diferenţa dintre sistemele de operare Windows pe 16 biţi (Windows 3.X) şi sistemele de operare Windows pe 32 de biţi (Windows 9X, Windows NT, Windows 2000, Windows XP).

Functionalitatea oferita de Interfete API Windows poate fi grupata in sapte categorii:

- **Base Services** (Servicii de baza)

Ofera acces resurselor fundamentale disponibile in Windows. Sunt incluse sisteme de fisiere, dispozitive, procese, fire de executie, acces la registri Windows si tratarea erorilor. Aceste functii se afla in fisierele kernel32.dll si advapi32.dll pe sisteme de operare pe 32 biti. Kernel nu este responsabil cu functiile de intrare/iesire si de interfata cu utilizatorul, proprii unui sistem de operare. Denumirea de nucleu al sistemului de operare provine din faptul ca el interactioneaza numai cu Windows

- **Graphics Devise Interface** (Interfata Dispozitivelor Graice)

Ofera functionalitate pentru afisarea continutului grafic pe monitoare, imprimante si alte dispozitive de iesire. Se afla in gdi32.dll pe sistemele de operare pe 32 biti. Se administreaza procesul grafic la modul general, independent de dispozitivul folosit pentru afisare. Din punct de vedere al programatorului, interfata GDI este formata din citeva sute de rutine si unele tipuri de date, macroinstructiuni si structuri de date;

Tipuri de apeluri de functii:

-***Functii care deseneaza*** ceva (*TextOut, DrawText*, desenarea liniilor, a

zonelor colorate si a *imaginilor* bitmap...);

-***Functii care obtin (sau creaza) si elibereaza (sau distrug) un context de dispozitiv***; (in API*: BeginPaint* ***...*** *EndPaint, GetDc, ReleaseDC*; in MFC (clase*): CDC, CPaintDC, CClientDC*

De exemplu functia *CDC::GetDeviceCaps,* ajuta sa obtina informatii despre un periferic. Va returna totdeauna valori fizice corecte pt imprimanta(LogpixelsX si logpixelsY).

-Functii care obtin informatii despre contextul de dispozitiv - structura TEXTMETRICS, *GetTextMetrics;*

CMetaFileDC, CWindowDC)

- **User Interface** (Interfata utilizator)

Ofera functionalitati pentru a crea si manipula ferestre si majoritatea controlaelor de baza, cum ar fi butoane si scrollbar-uri, pentru a recepta intrare de la mouse si tastatura, impreuna cu alte functionalitati asociate cu GUI (Graphical User Interface). Aceste functii se afla in user32.dll. De la Windows XP, controlele de baza se afla in comctl.dll, impreuna cu controalele generale (Common

Control Library). Aceasta functie nu se refera la utilizator, ci la comanda tuturor ferestrelor si administrarea acestora: continutul de informatie al ferestrei, structura de baza a lor, toate informatiile din meniuri si submeniuri. In afara ferestrelor modulul USER se ingrijeste si de alte elemente cum sunt casetele de dialog sau structurile de control apelabile prin butoane sau combinatii de taste. In acelasi timp, USER se ocupa cu incarcarea driverelor, perifericelor, supravegherea comunicatiei intre task-urile distincte, a ferestrelor, a iconurilor si aplicatiilor; realizeaza comanda cursorului si iconurilor. In cele din urma acesta gestioneaza intr-o oarecare masura resursele aplicatiilor: iconuri, meniuri, cimpuri de dialog pe care Windows le retine in memoria RAM. Atit pentru modulul USER, cit si pentru cel GDI, exista resurse de memorii speciale; daca ele sunt depasite atunci sistemul se poate bloca.

- **Common Dialog Box Library** (Biblioteca Ferestrerol de Dialog Generale)

Ofera aplicatii si ferestre de dialog standard pentru deschiderea si salvarea fisierelor, alegerea culuorii si fontului, etc. Linraria se afla in fisier numit comdlg32.dll si mai tarziu de la Windows 95 in fisier shlwapi.dll. Este grupat sub categoria *User Interface* a Interfetelor API.

- **Common Control Library** (Libraria de Control Generala)

Ofera acces aplicatiilor la unele controale avansate ale sistemului de operare. Acestea includ status bars, progress bars, toolbars si tabs. Libraria se afla intr-un fisier DLL numit comctl32.dll. Este grupat sub categoria *User Interface* a Interfetelor API.

- **Windows Shell**

Componenta Interfetelor API Windows care care permite aplicatiilor acces la functionalitatile shell ale sistemului de operare, la fel ca si schimbarea lui in vederea inbunatatirii. Acesta componenta se afla in fisierul shell32.dll si mai tarziu incepand cu Windows 95 in shlwapi.dll. Este grupat sub categoria *User Interface* a Interfetelor API.

- **Network Services** (Servicii de retea)

Ofera acces la posibilitatie sistemului de operare legate de retea. Subcomponentele lui includ Net Bios, Winsorck, NetDDE, RPC, etc.

Pe langa aceste interfete, exista si interfeteasociate Web, de exemplu Internet Explorer contine multe Interfete API care sunt la randul sau folosite de alte aplicatii. Aceste Interfete API pot fi considerate ca parte a Interfetelor API Windows. Internet Explorer ofera:

- Controlul browser-ului web incorporat, aflat in shdocvw.dll si mshtml.dll.
- Servicii URL, aflate in urlmon.dll. Aplicatiile pot deasemenea oferi serviciile proprii URL.
- Librarii pentru suport multilingvisit si international (mlang.dll)
- DirectX Transforms, un set de filtre pentru imaigni
- Suport XML (MSXML componente)
- Acces la Windows Adress Book

Interfete API pentru interactionare intre programe.In mare parte Interfetele API Windows se ocupa de interactionare intre Sistemul de Operare si aplicatii care ruleaza pe el. Pentru a face posibila comunicarea intre diferiate aplicatii, Microsoft a dezvoltat o serie de tehnologii incorporate in Interfetele API Windows, cum ar fi Component Object Model (COM) care este o platforma Microsoft introdusa in 1993. Este folosita pentru a permite comunicare intre procese si crearea dinamica de obiecte in orice limbaj care suporta tehnologia respectiva. Implementarea nu depinde de limbaj.

http://hubpages.com/hub/Win-32-Api
http://en.wikipedia.org/wiki/Windows_API
http://msdn2.microsoft.com/en-us/library/default.aspx

Capitolul 3
Apelurile pentru sistemul de fisiere in Linux

3.1 Introducere

Un sistem de fişiere este un mod de organizare a fişierelor şi prezentare a acestora utilizatorului. Din punct de vedere al utilizatorului un sistem de fişiere are o structură ierarhică de fişiere şi directoare, începând cu un director rădăcină. Localizarea unei intrări în sistemul de fişiere (fişier sau director) se realizează cu ajutorul unei căi în care sunt prezentate toate intrările de până atunci. Astfel, calea

```
/usr/local/app/file.txt
```

înseamnă că directorul rădăcină `/` are un subdirector `usr` urmat de subdirectorul `local`. Acesta are la rândul său are un subdirector `app` care conţine un fişier `file.txt`.

Fiecare fişier are asociat, aşadar, un nume cu ajutorul căruia se face identificarea, un set de drepturi de acces şi zone conţinând informaţia utilă.

3.2 Tipuri de fişiere

În Linux se deosebesc patru tipuri de fişiere: ordinare (obişnuite), pipe, speciale şi directoare. Din punct de vedere al sistemului de operare, un fişier este un şir de octeţi de o anumită lungime. Accesul la datele din fişiere se face în mod aleator, adică la orice poziţie (deplasament) la un moment dat.

2.1. ***Un fişier obişnuit*** este creat de un proces. El poate fi text sau binar (de exemplu un fişier executabil). Fiecare fişier are ataşat un număr, care este interpretat ca index într-o listă de structuri (index node),

păstrată într-o zonă rezervată pe partiţia ce conţine sistem de fişiere. Fiecare i-node conţine informaţii importante despre fişier.

2.2. ***Fişierele pipe*** sunt fişiere folosite pentru comunicarea între procese, pe baza principiului FIFO (First-In, First-Out).

2.3.***Fişiere speciale*** sunt fişiere ataşate dispozitivelor de I/E (de tip bloc sau caracter). În directorul /dev se găsesc toate referinţele la dispozitivele de I/E: discuri, benzi magnetice, terminale, imprimante, mouse etc. Acestea sunt considerate fişiere speciale. De exemplu, pentru fiecare partiţie a unui hard disc se găseşte câte un fişier special. Un fişier special deţine un i-node, care însă nu referă un blocuri de date pe disc. În schimb acest i-node conţine un număr de dispozitiv, care este folosit ca index într-o tabelă internă sistemului de operare de proceduri pentru dispozitive periferice. Pentru identificarea fiecărui dispozitiv se folosesc două numere: major (identifică tipul dispozitivului) şi minor (identifică numărul dispozitivului de tipul dat). Folosirea dispozitivelor în această manieră conferă avantajul tratării uniforme. Din punct de vedere utilizator nu există nici o diferenţă între lucrul cu fişiere ordinare şi cele speciale. De exemplu:

```
cp prg.c /home/mihaelan/ prg1.c #     copiere simplă
```

2.4.***Un director*** face legătura între numele fişierelor şi locul unde acestea sunt memorate pe disc. El nu conţine efectiv fişierele care îi aparţin, ci doar referinţele la acestea, sub forma unei succesiuni de structuri numite intrări în director. O intrare în director are cel puţin două câmpuri, şi anume: numele fişierului şi i-node-ul său. Orice director are în mod implicit două intrări create automat odată cu crearea directorului. Acestea poartă numele '.' şi '..' şi conţin referinţe spre i-nodul directorului curent şi, respectiv spre părintele directorului curent. Un director ce conţine doar cele două intrări amintite se numeşte un director gol, putând fi şters cu ajutorul comenzilor de ştergere.

Ierarhia sistemului de fişiere Unix are un singur director cunoscut sub numele de `root` şi notat `/`, prin care se localizează orice fişier. Notaţia Unix

pentru căile fişierelor este un şir de nume de directoare despărţite prin /, urmat de numele fişierului. Există şi căi relative la directorul curent . sau la directorul părinte ..

În Unix nu se face nici o deosebire între fişierele aflate pe partiţiile discului local, pe CD sau pe o maşină din reţea. Toate aceste fişiere vor face parte din ierarhia unică a directorului root. Acest lucru se realizează prin **montare**: sistemele de fişiere vor fi montate într-unul din directoarele sistemului de fişiere rădăcină.

3.3 Operaţii pe fişiere

Un **descriptor de fişier** în Unix este un întreg care indexează o tabelă cu pointeri spre structuri care descriu fişierele deschise de un proces. Un program care rulează într-un shell Unix îşi deschide 3 fişiere standard cu file descriptori cu valori speciale:

3.4 **standard input** (cu file descriptorul 0) (implicit, citit de la tastatură)

3.5 **standard output** (cu file descriptorul 1) (implicit, afişat pe display)

3.6 **standard error** (cu file descriptorul 2) (implicit, afişat pe display)

3.3.1 *Open*

Funcţia este folosită pentru deschiderea unui fişier; funcţia este declarată în `fcntl.h` iar sintaxa apelului este următoarea:

```
int open(const char *FILENAME, int FLAGS[, mode_t MODE]);
```

unde

- `FILENAME` este numele fişierului care se doreşte deschis
- `FLAGS` controlează modul în care se realizează deschiderea. Se pot specifica mai multe flag-uri făcând sau pe biţi. Cele mai uzuale flag-uri sunt

`O_RDONLY`	`O_EXCL`

O_WRONLY	O_TRUNC
O_RDRW	O_APPEND
O_CREAT	O_NONBLOCK/O_NDELAY

MODE este folosit pentru a stabili drepturile în cazul în care se creează un nou fişier (adică se precizează flag-ul O_CREAT

Dacă operaţia are succes funcţia întoarce file descriptorul alocat (o valoare întreagă pozitivă). În caz de eroare se întoarce -1, iar variabila errno primeşte o valoare care indică motivul eşecului.

Dacă, spre exemplu, dorim să deschidem fişierul alina.txt pentru scriere, cu trunchiere, şi fişierul dan.txt pentru citire şi scriere, cu eventuala creare a acestuia , vom folosi următoarea secvenţă de cod:

Exemplu

```
[...]
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
[...]
   int fd1, fd2;
      fd1 = open ("miha.txt", O_WRONLY | O_TRUNC);
   if (fd1 < 0) {
      perror ("open");
      exit (EXIT_FAILURE);
   }
      fd2 = open ("ana.txt", O_RDWR | O_CREAT, 0644);
   if (fd1 < 0) {
      perror ("open");
      exit (EXIT_FAILURE);
```

```
}
```

3.3.2 *Creat*

Declarată tot în `fcntl.h` este şi funcţia `creat`, care creează un fişier şi are următoarea sintaxă:

```
int creat(const char *FILENAME, mode_t MODE)
```

Funcţia este echivalentă cu:

```
open(FILENAME, O_WRONLY|O_CREAT|O_TRUNC, MODE)
```

3.3.3 *Close*

Funcţia este declarată în `unistd.h` iar sintaxa apelului este următoarea:

```
int close(int FILEDES)
```

unde

- `FILEDES` este filedescriptorul care se doreşte închis.

Dacă dealocarea are loc fără probleme atunci valoarea întoarsă este `0`, altfel `-1`. Ca şi la `open`, mai multe informaţii despre cum s-a terminat operaţia se pot obţine inspectând variabila `errno`.

O greşeală frecventă de programare este neverificarea codului de eroare întors la `close`, pentru că se poate întâmpla ca o eroare la scriere (`EIO`) să fie întoarsă utilizatorului abia la `close`.

3.3.4 Scrierea şi citirea

Scrierea într-un fişier şi citirea dintr-un fişier se realizează cu ajutorul apelurilor `read` şi `write`. Aceste apeluri primesc trei argumente:

- descriptorul fişierului folosit
- buffer-ul ce conţine datele pentru scriere sau unde se stochează datele citite din fişier
- numărul de octeţi care vor fi citiţi/scrişi

Read

Funcţia `read` este declarată în `unistd.h` şi are sintaxa de apel:

```
ssize_t read(int FILEDES, void *BUFFER, size_t SIZE);
```

unde

- `BUFFER` e un pointer spre zona în care vor fi depuse datele citite din fişierul indicat de `FILEDES`
- `SIZE` e cantitatea maxima de date care poate fi primită.

Funcţia întoarce numărul de octeţi citiţi. Valoarea minimă este de un octet, iar când se ajunge la sfârşit se va întoarce `0`. Dacă se face read după ce s-a ajuns la sfârşit se va întoarce în continuare `0`. Dacă apare o eroare se întoarce `-1` şi variabila `errno` este setată corespunzător cauzei care a determinat eroarea.

Write

Funcţia `write` este declarată, de asemenea, în `unistd.h` iar sintaxa este următoarea:

```
ssize_t write(int FILEDES, const void *BUFFER, size_t SIZE);
```

unde parametrii au semnificaţii similare cu cei ai apelului `read`. Valoarea întoarsă este numărul de octeti ce au fost efectiv scrişi. În mod implicit nu se garanteaza că la revenirea din write scrierea în fişier s-a terminat. Pentru a forţa actualizarea se poate folosi `fsync` sau fişierul se poate deschide folosind flagul `O_FSYNC`, caz în care se garanteaza că după fiecare `write` fişierul a fost actualizat.

3.4 Poziţionarea în fişier (lseek)

Funcţia `lseek` permite poziţionarea într-un anumit loc într-un fişier. Similar cu `read` şi `write`, este declarată în `unistd.h`. Sintaxa este următoarea:

```
off_t lseek(int FILEDES, off_t OFFSET, int WHENCE)
```

unde

- `FILEDES` este filedescriptorul fişierului în care dorim repoziţionarea
- `OFFSET` este un deplasament

- `WHENCE` indică modul in care trebuie interpretat `OFFSET`. Poate fi `SEEK_SET` (raportare la începutul fişierului), `SEEK_CUR` (raportare la poziţia curentă în fişier) sau `SEEK_END` (raportare la sfârşitul fişierului).
- Valoarea întoarsă reprezintă offset-ul la care s-a ajuns; un `lseek` pe `SEEK_CUR` cu `OFFSET` zero întoarce poziţia curentă.

Bibliografie comuna

http://en.wikipedia.org

www.linuxiso.org

www.whatis.com

Capitolul 4
EXT 2 si arhitectura NFS la LINUX

4.1 Caracteristici generale

Fiecare sistem de operare se foloseste de propriul sistem de fisiere care este implementat intr-un mod diferit pentru fiecare sistem fisiere in parte.

In LINUX primul sistem de fisiere era bazat pe sistemul de fisiere *MINIX* dar pe masura ce sistemul de operare LINUX s-a maturizat a aparut *Extended Filesystem* care a adus imbunatiri sistemului de fisiere dar din punct de vedere al performantei era nesatisfacator.Ext 2 sau *Second Exended Filesystem* a aparut in 1994 si odata cu noile modificari acesta a devenit cel mai folosit sistem de fisiere pe LINUX.

O parte din modificarile aduse de Ext 2 LINUX sunt:

- Atunci cand se creaza un sistem de fisiere , administratorulpoate sa aleaga marimea optima a blocului (intre 1024 sau 4096 bytes) depinzand de marimea fisierelor care se asteapta a fi stocate.In acest sens este de preferat sa alegi un bloc de marime 1024 atunci cand majoritatea fisierelor au marime mai mici de 1024 bytes pentru ca asta duce la mai putine fragmentari interne.Pe de alta parte se aleg blocuri mai mari de cateva mii de bytes pentru fisierele care depasesc 1024 de bytes pentru ca in acest caz vor avea loc mai putine transferuri intre discuri;

- De altfel administratorul mai are posibilitatea de a alege cate inoduri sa permita unei partitii de o anumita marime depinzand de dumarul de fisiere care se vor stoca acolo.Acest lucru va duce la marirea eficientei spatiului liber de pe disc;

- Sistemul de fisiere partitioneaza blocurile de disc in grupuri si fiecare grup contine blocuri de date si inoduri stocate in track-uri adiacente.Cu acest tip de structura fisierele stocate intr-un singur grup de bloc-uri pot fi accesate cu un timp de cautare mai mic;

- In sistemul de fisiere Ext 2 sunt suportate symbolic link-urile rapide.Calea unui symbolic link se gaseste in inod daca aceasta are mai putin de 60 de bytes;

- Exista un support pentru verificarea starii sistemului de fisiere in momentul boot-arii.Verificarile sunt facute de /sbin/e2fsck;

- De altfel Ext 2 are un suport pentru fisierele imuabile (care nu se schimba) si pentru fisierele de tip append-only (unde se pot adauga date doar la sfarsitul fisierului).Aceste protectii nu pot fi dezasamblate nici de catre superuser;

Alte aparitii sau imbunatatiri la sistemul de fisiere Ext 2 :

Fragmentarea blocurilor

Dat fiind faptul ca de obicei administratorii aleg blocuri de marime mare pentru a accesa discurile recente rezultatul este ca fisierele mici care sunt stocate in blocuri mari fac o risipa de spatiu.Aceasta problema se poate rezolva daca li se permite fisierelor sa fie stocate in diferite fragmente din acelasi bloc.

Lista de control al accesului

In loc sa clasificam userii unui fisier in cele 3 grupuri(proprietar,grup si altii) – lista de control al accesului(ACL) este asociata fiecarui fisier pentru a putea specifica drepturile pentru oricare dintre useri.

Fiserele comprimate su fisierele criptate

Aceasta noua optiune care trebuie specificata in momentul crearii unui fisier le va permite utilizatorilor sa stocheze si sau sa stocheze fisiere comprimate sau criptate pe disc.

Stergerea logica

Optunea de *undelete* le va permite userilor sa recupereze cu usurinta continutul fiserului anterior sters.

4.2 Strucutura de date a discului

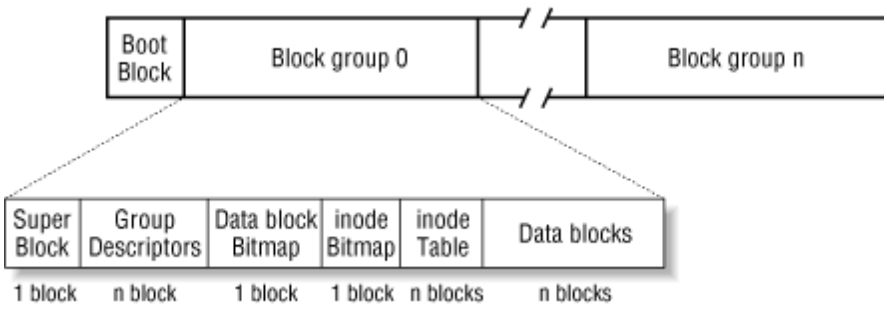

Fig 4.1 Strucutura de date a discului

Primul bloc din orice partitie a Ext 2 nu este manageriat de sitemul de fisiere al Ext 2 ci este rezervat pentru partitia sectorului de boot.Restul partitiei Ext 2 este impartita in grupuri de blocuri.Aceste grupuri de blocuri reduc fragmentarea fisierelor din moment ce kernel-ul incearca pe ct posibil sa tina blocurile de date ca apartinand unui fisier in acelasi grup de blocuri.Fiecare bloc dintr-un grup de blocuri contine urmatoarele informatii:

-o copie a sistemului de fisiere al superblocului
-un grup de inoduri
-un bloc de date al bitmapului

Daca un bloc de date nu contine nicio informatie insemnata se spune ca blocul este liber.Un superbloc al Ext 2 este stocat in structura ext2_super_block.Tipurile de date _u8,_u16, _u32 sunt de fapt numere neasignate de lungime 8, 16, and 32 bits in timp ce _s8, _s16, _s32. Numarul inodurilor se gasesc in campul *s_inodes_count* iar in campul *s_blocks_count* se stocheaza numarul de blocuri din sistemul de fisiere Ext2.Campurile s*_mnt_count, s_max_mnt_count, s_lastcheck, and s_checkinterval* seteaza sistemul de fisiere Ext2 pentru a fi verificat in mod automat in timpul boot-arii.

Fiecare grup de blocuri are propriul descriptor de grup care se gaseste in structura *ext2_group_desc.*Campurile *bg_free_blocks_count, bg_free_inodes_count, si bg_used_dirs_count* se folosesc atunci cand se aloca noi inoduri si blocui de date.Aceste campuri determina care este cel mai potrivit bloc pentru care sa poata aloca o anumita strucutura de date.

Tabelul de inoduri consta intr-o serie de blocuri consecutive, si fiecare sa contina un numar predefinit de inoduri.Numarul primului bloc din tabelul inodurilor este stocat in campul *bg_inode_table* al grupului descriptor.Toate inodurile ocupa la fel adica 128 bytes.

Fisierele regulate au nevoie de blocuri de date doar atunci cand incep sa contina date.Cand sunt create prima data fisierele sunt goale si nu au nevoie de blocuri de date.

Directoarele sunt implementate de Ext 2 ca niste fisiere special ale caror blocuri de date stocheaza numele fisierelor impreuna cu inodul corespunzator.Din motive de eficienta lungimea intrarilor intr-un director este intotdeauna un multiplu de 4, si de aceea caracterul nul (/0) este adaugat pe post de "captuseala" la sfarsitul numelui fisierului daca este necesar.Campul *name_len* stocheaza lungimea numelui fisierului.

Pentru a sterge intrare unui director este suficient sa ii setezi campul inodului si sa incrementezi valoarea campului *rec_len* cu valoare valida a precedent.Un exemplu de director este exemplificat in figura de mai jos:

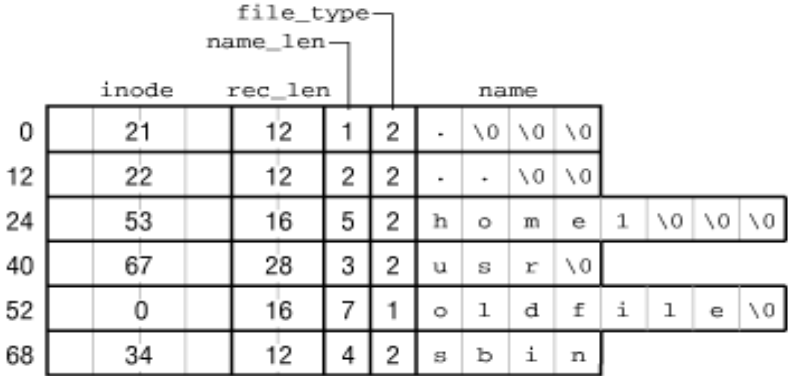

Fig 4.2 Structura director

4.3 Structura de date a memoriei

Pentru a fi cat mai eficient ,o partitie a sistemului de fisiere Ext 2 care stocheaza majoritatea informatiei pe disc o si copiaza pe RAM atunci cand sistemul de fisiere este montat.Pentru a vedea cat de des se schimba structurile de date vom considera cazurile de mai jos:

-Atunci cand se creaza un nou fisier valoare din campul *s_free_inodes_count* din superblocul Ext 2 si campul *bg_free_inodes_count* din grupul corespunzator descriptor trebuie sa fie decrementat.

-Daca kernel-ul adauga date la un fisier existent astfel incat numarul de blocuri de date allocate pentru acesta sa creasca, atunci valoarea campului *bg_free_blocks_count* din grupul descriptor va trebui modificat.

-Chiar si atunci cand rescrii o portiune a unui fisier deja existent implica un update al campului *s_wtime* din superblocul sistemului de fisiere Ext 2.
In figura de mai jos este ilustrata structura de date a memoriei a unui sistem de fisiere montat.In figura sunt 3 grupuri de blocuri ale caror decriptori sunt stocati in 3 blocuri pe disc;de altfel campul *s_group_desc* al *ext_sb_info* este indreptat catre un array cu un buffer cu 3 terminatii.

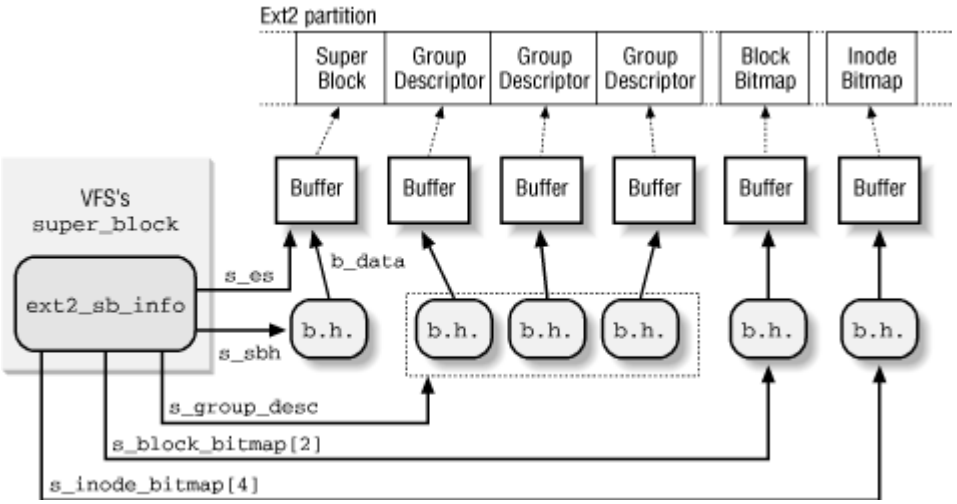

Fig 4.3 Alocare date in memorie

4.4 Crearea sistemului de fisiere

Sistemele de fisiere al Ext 2 sunt create de catre utilitarul */sbin/mke2fs.*
Acest program face urmatoarele actiuni:
-initializeaza superblocul si grupul descriptor;
-verifica daca partitia contine blocuri defective iar daca gaseste acest tip de blocuri creaza o lista cu acestea;
-pentru fiecare grup de blocuri ,se rezerva toate blocurile de disc de care este nevoie pentru stocarea superblocului, grupului descriptor, tabelului de inoduri, si cele 2 bitmapuri;
-initializeaza tabelul de inoduri ale fiecarui grup de blocuri
-creaza directorul /root;
-creaza directorul lost+found care este folosit de catre */sbin/e2fsck* pentru a tine o legatura cu blocurile pierdute sau defective gasite;
-updateaza bitmap-ul inodului si bitmapul blocului de date al grupului de blocuri in care au fost create directoarele anterioare;
-grupeaza blocurile defective (daca sunt) in directorul *lost+found*

Capitolul 5
Compresia si criptarea fisierelor NTFS

5.1 Compresia fisierelor NTFS

Compresia se foloseste pentru a reduce dimensiunile fisierului,fiind importanta pentru a economisi timp,spatiu ,dar si pentru datele redundante.Aceasta se mai foloseste si pentru a elibera un anumit spatiu pe disk,atat pentru fisiere text,imagini,video,audio etc.

Exemple de aplicatii pentru compresia datelor:

- Fisiere:GZIP,BZIP,BOA
- Arhive:PKZIP
- Sistem de fisiere:NTFS
- Imagini:GIF,JPEG
- Sunet:MP3
- Video:MPEG,HDTV
- Baza de date:GOOGLE
- Comunicatii:ITU-T T4 Group 3FAX

Partitiile sistemului de fisiere NTFS suporta compresia fisierelor intr-un fisier de baza individual.Algoritmul de compresie al fisierelor de catre sistemul de fisiere NTFS se numeste compresia Lempel –Ziv.Acesta este un algoritm de compresie « fara pierdere »,prin care se intelege ca datele nu se pierd atunci cand are loc compresia fisierului ,in opozitie cu algoritmii de compresie « cu pierdere » ca in cazul JPEG ,cand de fiecare data cand se face compresia datelor au loc pierderi.

Prin compresia datelor se reduce marimea fisierului minimizand datele.Intr-un fisier text,datele redundante pot fi adesea caractere intamplatoare,de

exemplu caracterul spatiu,sau vocale precum literele e sau a ;pot fi de asemenea si caractere de tip string.

Fiecare algoritm de compresie a datelor minimizeaza date redundante intr-un anumit mod.De exemplu, « Algoritmul de codare Huffman » atribuie un cod caracterelor dintr-un fisier si este bazat aparitia acelor caractere.Un alt algoritm de compresie,numit codarea RUN-LENGTH imparte in doua categorii caracterele care se repeta :prima parte specifica timpul de repetare al caracterului(perioada),iar cea de-a doua parte indica ce caracter s-a repetat.Un alt algoritm de compresie,cunoscut sub numele de algoritmul LEMPEL-ZIV,converteste variabilele de tip string in coduri fixe care ocupa mai putin spatiu decat stringul original.

http://msdn2.microsoft.com/en-us/library/aa364219.aspx

5.2 Compresia fisierelor sistemului de fisiere NTFS

In sistemul de fisiere NTFS, compresia are loc in mod transparent.Acest lucru implica folosirea fara schimb de cereri pentru modificarile aplicatiilor.Compresia bitilor fisierului nu este accesibil pentru aplicatii ;se pot vedea doar datele necomprimate.Prin urmare,aplicatiile care deschid fisierele comprimate pot opera in sistemul NTFS cu datele care nu au fost comprimate.Totusi,aceste fisiere nu pot fi copiate de un alt sistem de fisiere.

Daca dorim sa facem compresia unor date mai mari de 30Gb,aceasta nu se va finaliza cu succes.

http://msdn2.microsoft.com/en-us/library/aa364219.aspx

Atributele compresiei

In sistemul de partitii ale fisierelor NTFS ,fiecare fisier si director au cate un atribut de compresie.Celelalte sisteme de fisiere pot implementa cate un atribut de compresie pentru fisiere individuale.

Se poate determina daca un sistem de fisiere suporta un atribut de compresie pentru fisier si director apeland functia GetVolumeInformation si examinand fanionul (« bit flag ») FS_FILE_COMPRESSION.

Utilizand GetFileAttributes si GetFileAttributesEx se poate determina atributul de compresie al fisierului sau directorului. Daca atributul de compresie al unui fisier este FILE_ATTRIBUTE_COMPRESSED,atunci toate datele din fisier sunt comprimate.Daca atributul de compresie nu are nicio valoare, atunci nicio data din fisier nu a fost comprimata.

Atributul de compresie este un indicator simplu BOOLEAN al starii comprimarii.

Atributul de compresie al unui director de fisiere prezinta attribute de compresie nesigure pentru fisierele si subdirectoarele nou create.Cand se apeleaza **CreateFile** sau **CreateDirectory** pentru a creea un fisier nou sau un director,fisierul cel nou sau directorul mostenesc atributul compresiei din directorul parinte.

http://msdn2.microsoft.com/en-us/library/aa363849(VS.85).aspx

5.3 Starea compresiei

Fiecare fisier sau director dintr-o partitie care suporta compresie au o anumita stare.

Pe cand atributul compresiei unui fisier sau director indica pur si simplu daca un fisier sau director a fost comprimat sau nu,starea compresiei specifica formatul datelor de comprimat.

Folosind codul de control FSCTL GET COMPRESSION se poate determina starea compresiei unui fisier sau director.Starea compresiei este o valoare pe 16 biti.Aceasta operatie seteaza atributul compresiei fisierului sau directorului. COMPRESSION_FORMAT_NONE arata ca fisierul nu este comprimat,iar valoarea COMPRESSION_FORMAT_DEFAULT arata ca fisierul a fost comprimat .

Folosind codul de control FSCTL SET COMPRESSION se seteaza starea compresiei a fisierului sau directorului. Prin aceasta operatie se seteaza si atributul de compresie al fisierului sau directorului.
Setand starea compresiei la o valoare diferita de zero,se face compresia fisierului,daca starea este setata pe zero are loc decompresia fisierului.Acestea sunt operatii sincrone.Fisierul este comprimat sau extins imediat ce se seteaza aceste stari.

Setand starea de compresie a directorului aceasta nu implica imediat compresia sau extinderea imediata(ca in cazul fisierului).

http://msdn2.microsoft.com/en-us/library/aa363849(VS.85).aspx

Obtinerea dimensiunii unui fisier comprimat

Folosind functia **GetCompressedFileSize** se obtine dimensiunea unui fisier comprimat.Daca un fisier a fost comrpimat dimensiunea sa va fi mai mica decat atunci cand era necomprimat.Folosind functia **GetFileSize** se poate determina dimensiunea fisierului necomprimat.

5.4 Efecte ale compresiei prin mutarea sau copierea fisierelor

Mutand sau copiind fisiere intre partitii se schimba starea compresiei.Starea compresiei unui fisier NTFS se controloleaza prin atributul sau.

De exemplu,daca se muta un fisier necomprimat intr-un folder comprimat,fisierul ramane necomprimat dupa mutare.

Daca se copiaza un fisier comprimat intr-un folder necomprimat,fisierul devine automat necomprimat ,deci capata aceeasi stare ca si folderul in care este copiat.

http://msdn2.microsoft.com/en-us/library/aa364223(VS.85).aspx

In cazul compresiei pot aparea de asemenea erori.Un exemplu de eroare poate fi „Sistemul de fisiere nu poate suporta compresia.”Cauza acestei erori este datorita dimensiunii partitiei.Compresia fisierelor NTFS

Nu este suportata pentru o locatie mai mare de 4Kb.Pentru a rezolva aceasta problema,se poate face reformatarea partitiei NTFS utilizand clustere de dimensiuni mai mici sau egale cu 4Kb.

In concluzie,exista doua cai pentru a masura performantele compresiei de date NTFS : dimensiunea si viteza.Se poate spune cat de bine lucreaza compresia comparand dimensiunile fisierelor sau datelor necomprimate cu ale celor comprimate

5.5 Criptarea fisierului

Sistemul de fisiere criptate sau EFS(Encrypting File System) ,a fost introdus in NTFS 5.0 fiind in plus un nivel de securitate pentru fisier si director.El ofera o protectie criptografica fisierelor individuale din partitiile sistemului de fisiere NTFS folosind o cheie publica pentru system.

Controlul accesului la obiectele fisierului sau directorului oferit de modelul de securitate al WINDOWS-ului este suficient pentru a proteja informatia “sensibila” de accesul neautorizat.Totusi,daca un laptop care contine date importante este pierdut sau furat , protectia datelor din punct de vedere al securitatii este compromisa.Criptand fisierele marim securitatea.

Pentru a determina daca un sistem de fisiere suporta criptarea fisierului sau directorului ,putem apela functia **GetVolumeInformation** si examinand fanionul FS_FILE_ENCRYPTION.Urmatorii itemi nu pot fi criptati:

- Fisierele comprimate

- Sistemul de fisiere
- Sistemul de directoare
- Tranzactiile
- Registrele de baza

Putine fisiere pot fi criptate.

TxF nu poate suporta o mare parte din operatiile de criptare cu fisierele EFS .Singura operatie pe care o suporta TxF este citirea operatiilor ,ca de exemplu **ReadEncryptedFileRaw**.

http://msdn2.microsoft.com/en-us/library/aa364223(VS.85).aspx

5.6 Lucrul cu fisierele si directoarele criptate

Un programator sau utilizator poate semnala un fisier ca fiind criptat . Un fisier gasit criptat a fost criptat de sistemul de fisiere NTFS utilizand driverul curent de criptare.Daca mai tarziu fisierul a fost gasit necriptat,acesta a fost decriptat si plasat intr-un fisier text nesecurizat.

Pentru a cripta un nou fisier se foloseste functia **CreateFile** impreuna cu flagul FILE_ATTRIBUTE_ENCRYPTED.Pentru a cripta un fisier existent se foloseste functia **EncryptFile** cu ajutorul careia se cripteaza toate datele.Daca fisierul a fost deja criptat ,**EncryptFile** va intoarce o valoare diferita de zero , care indica succesul operatiei.Daca fisierul a fost deja comprimat , EncryptFile face decomprimarea fisierului inainte de a-l cripta.

Pentru a decripta un fisier criptat se foloseste functia **DecryptFile**.Daca fisierului nu era criptat ,DecryptFile nu face nimic,dar returneaza o valoare diferita de zero.

Functia **EncryptionDisable** activeaza sau dezactiveaza criptarea directorului indicat si a fisierelor din director.Aceasta nu afecteaza subdirectoarele criptate anterior in directorul indicat.

http://www.microsoft.com/technet/security/guidance/cryptographyetc/efs.mspx

Caracteristici importante ale EFS:

- Criptarea EFS poate avea loc la nivel de fisier al sistemului,nu la nivel de aplicatie;criptarea si decriptarea fisierelor sunt transparente pentru utilizator si pentru aplicatie;daca un folder este criptat , fiecare fisier

din acesta care este creat sau mutat intrun alt folder poate fi criptat;daca un utilizator doreste sa acceseze un fisier criptat va introduce o parola;daca parola nu este corecta,li se va afisa un mesaj de roare care contine „ accesul nepermis”.

- Parolele(cheile) pot fi arhivate si tinute intr-un loc sigur pentru a nu fi descoperite.
- Aceste chei sunt protejate de parola utilizatorului;orice utilizator care afla parola poate avea acces la fisierele criptate si sa le decripteze.
- Inainte de criptarea fisierelor trebuie sa ne asiguram ca partitia apartine unui fisier NTFS,fisierul nu a fost comprimat,s-a scris accesul catre fisier.

5.7 Beneficiile securitatii NTFS

NTFS asigura securitatea fisierelor si folderelor cu ajutorul ACL-urilor(LISTE DE CONTROL AL ACCESULUI).ACL-urile sunt descriptori de securitate atasati fisierelor si directoarelor din sistemul de fisiere NTFS.Niciun fisier sau director nu poate avea mai multe nivele de permisiune al accesului.Inainte ca unui proces sa i se permita accesul catre un fisier,sistemul de securitate verifica daca procesul este autorizat sa faca acest lucru.

NTFS suporta directoare active.Directoarele active permit sistemului sa acceseze un domeniu,si utilizand autorizarea din artea serverului de baza stabileste permisiunile fisierelor.Sistemul de fisiere FAT nu implementeaza securitatea,si toti utilizatorii au drepturi egale de acces la fisierele si directoarele din sistem.

Concluzii

Din punct de vedere al utilizatorilor,NTFS organizeaza fisiere in directoare,iar ca si in cazul HPFS,acestea sunt sortate.La NTFS nu exista locatii speciale pe disc,in comparatie cu FAT si HPFS unde exista tabelele FAT si respectiv,Super Blocurile HPFS.

Avantajele folosirii NTFS sunt acelea ca acesta se utilizeaza pe partitii de 400MB sau chiar mai mari.Un utilizator nu trebuie niciodata sa compileze un utilitar reparat al discului pe o partitie NTFS.In NTFS nu exista obiecte special ape disc si exista un numar mare de copii al MFT(Master File Table).

Desi exista destule diferente intre sistemele de fisiere folosite de Linux si de Windows, principiile de baza raman asemanatoare. Fie ca este vorba de un utilizator advanced sau de unul novice, resursele hardware trebuiesc gestionate pentru a oferi versatilitate, fiabilitate si performanta ; sistemul de fisiere folosit de sistemul de operare trebuie sa se indeplineasca aceste conditii.

Idex Figuri