Probleme abordate

Administratorul proceselor sub Windows

Apel de procedură locală

Sistemul de fişiere

Subsistemele de mediu

Administrarea proceselor sub Windows

- Furnizează serviciile necesare pentru creare, ştergere şi utilizare a thread-urilor şi proceselor. Relaţiile dintre procesele părinte şi cele fiu sunt rezolvate de către subsistemele de mediu.
- De **exemplu**, când o aplicaţie care este executată sub Win32 cere crearea unui proces, se executa urm. pasi:

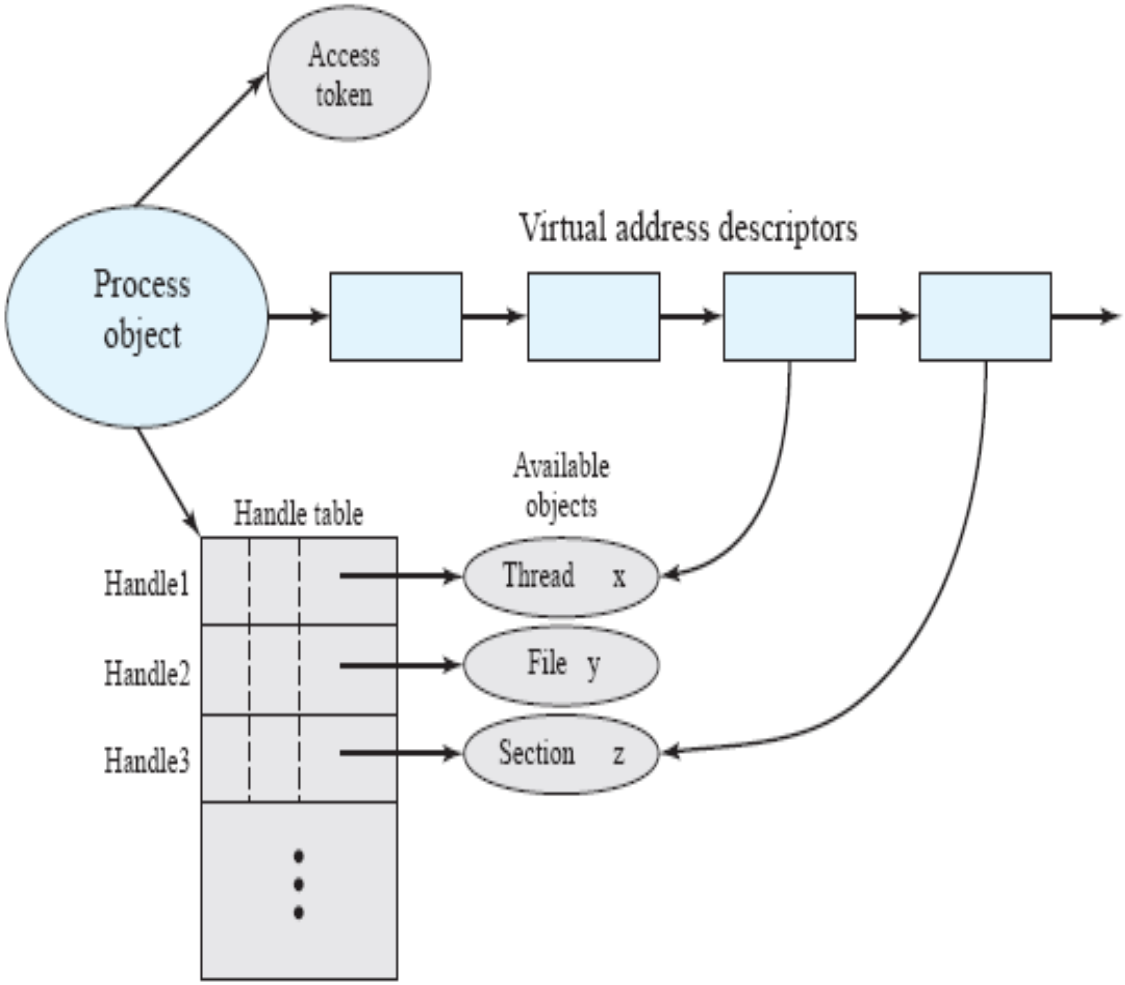
  - se apelează o metodă `CreateProcess`, prin care este transmis un mesaj subsistemului Win32, care cere administratorului de procese să creeze acel proces.

  - Administratorul de procese cere administratorului de obiecte să creeze un obiect proces şi să returneze un manipulator de obiecte lui Win32.

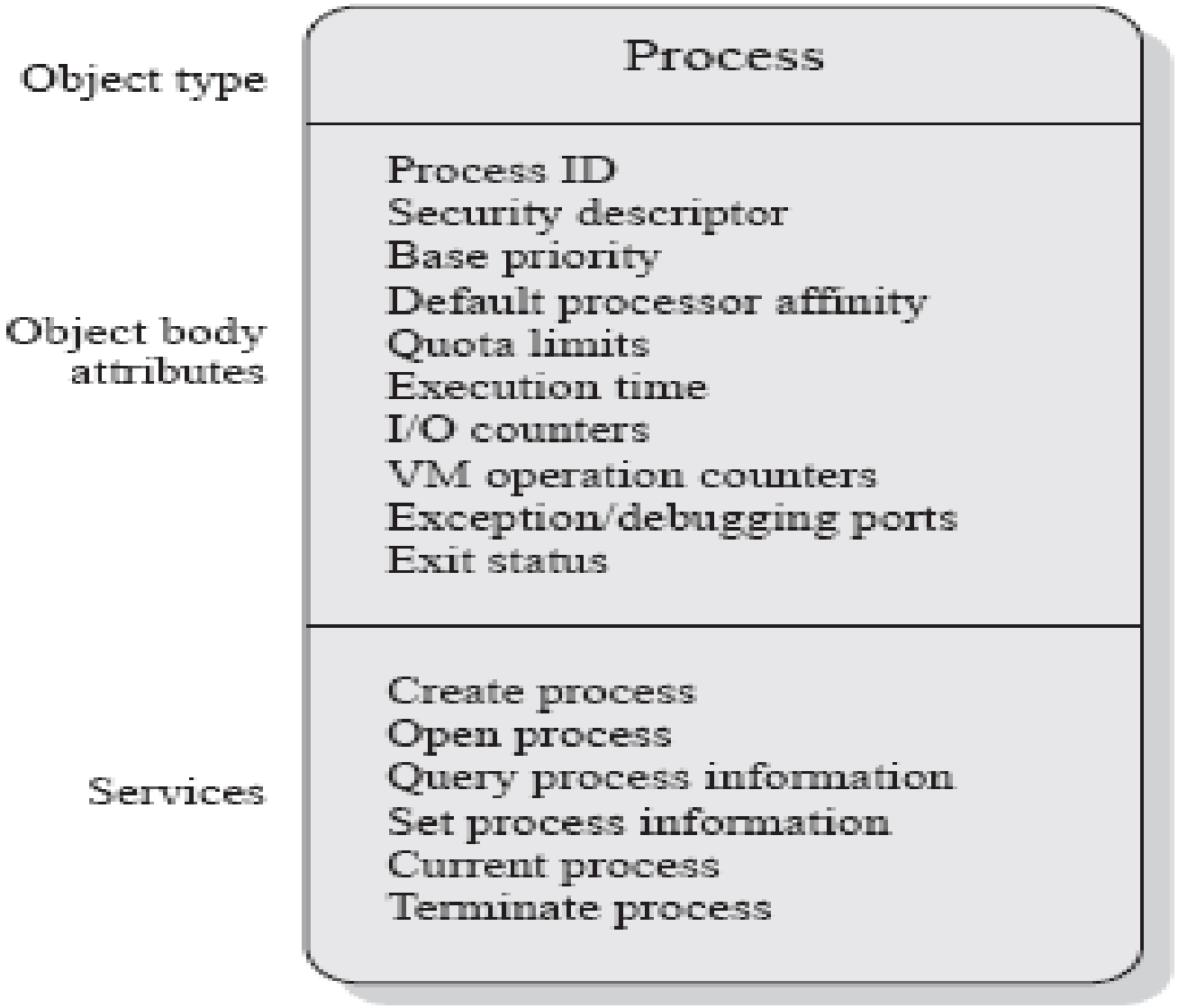
  - Win32 va apela din nou administratorul de procese, pentru a crea threadul respectiv

  - În final, Win32 returnează manipulatorul către noul thread.
- Caracteristicile cele mai importante ale proceselor sub Windows sunt:
  - sunt implementate ca obiecte.
  - un proces in executie poate sa contina unul sau mai multe thread-uri.
  - ambele au posibilitati de sincronizare.
- Figura urm., ilustreaza modul in care un proces controleaza sau utilizeaza resursele.

- Fiecarui proces ii este asignat un **jeton de acces**, jetonul primar al procesului.
- Cind un utilizator se loghineaza ptr. prima data, Windows creaza un jeton de acces care include **identificatorul de securitate** al utilizatorului. Fiecare proces care este creat sau executat in contextul acestui utilizator are o copie a acestui jeton de aces.
- Windows foloseste jetonul pt. a valida abilitatile unui anumit proces, din clasa legata de utiliz. resp., de a accesa obiecte securizate sau de a executa anumite functii cu caracter restrictiv pe obiectele securizate.
- De as. jetonul de acces controleaza daca procesul isi poate schimba propriile atribute; in acest caz, procesul nu are deschis un handler ptr. a-si accesa jetonul. Daca procesul incearca sa deschida un astfel de handler, sistemul de securitate determina daca acest lucru este permis, adica procesul isi poate schimba atributele proprii.
- De as., legat de procese exista o serie de blocuri ce definesc spatiul de adrese virtuale asignat in mod curent unui anumit proces. Procesul nu poate modifica direct aceste structuri, dar poate cere acest lucru adm. memoriei virtuale, care poate furniza un serviciu de alocare a memoriei pt. procese.
- Procesele includ un **obiect tabela** (fig. urm), cu manipulatori pt. alte procese (thread-uri) vizibile acestui proces. In plus, procesele au acces la un **obiect fisier** si la unul **sectiune** , care definesc o sectiune a memoriei partajate.


Access token
Process object
Virtual address descriptors
Handle table
Available objects
Handle1
Handle2
Handle3
Thread x
File y
Section z

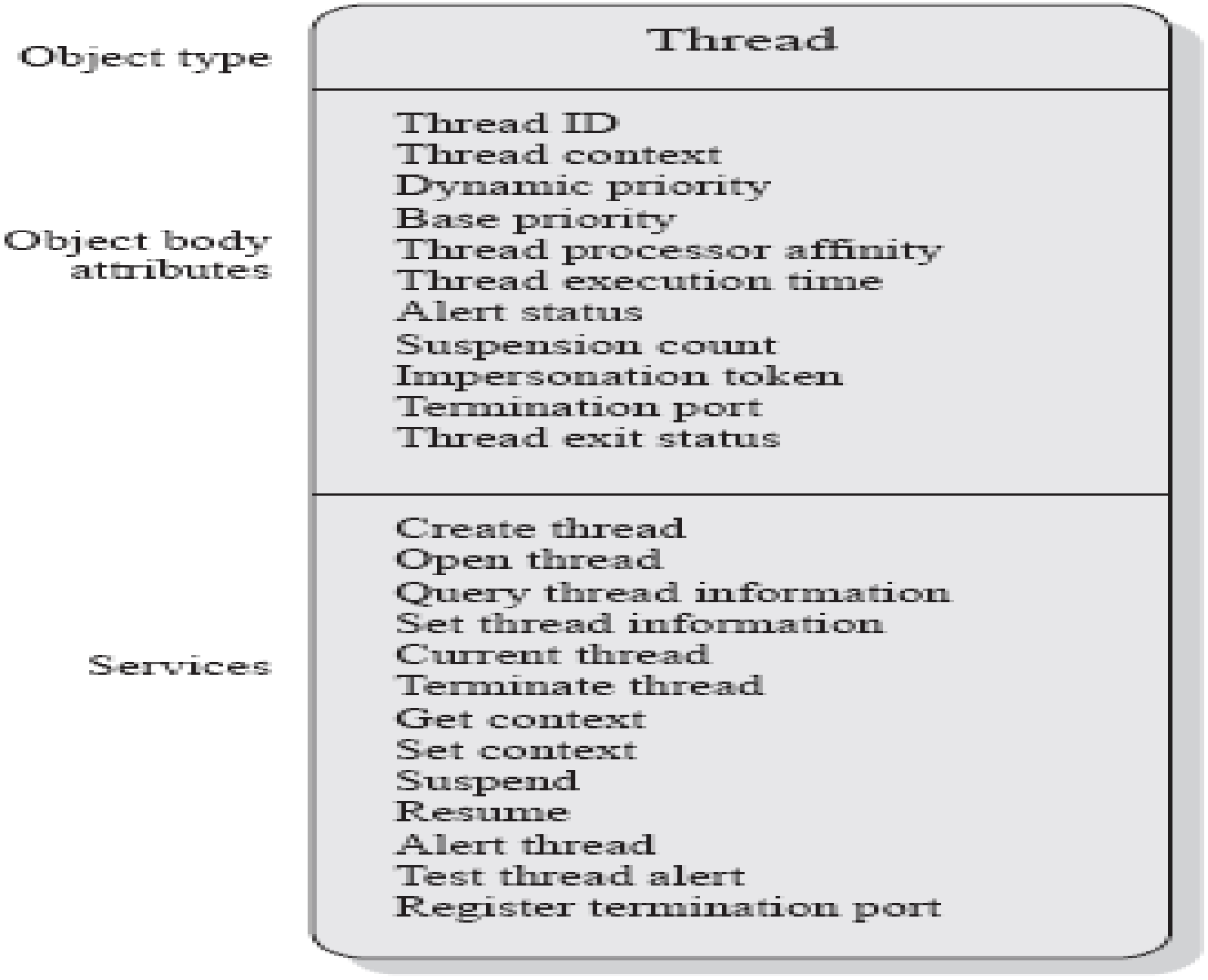
- **Obiectele proces si thread**
- Windows face utilizabile doua tipuri de obiecte legate de procese: procese si thread-uri.
- Procesul este o entitate care corespunde unui job al unui utilizator sau unei aplicatii ce detine anumite resurse, cum ar fi memorie si fisiere deschise.
- Thread-ul este o componenta a procesului ce are un singur fir de executie, care poate fi intrerupt, astfel incit procesorul poate comuta la alt thread.
- Fiecare proces Windows este reprezentat ca un obiect, a carui structura generala este aratata in fig. urm.; de as., el este definit printr-un numar de atribute si incapsuleaza un numar de actiuni sau servicii, pe care le poate executa. Un proces va executa un serviciu cind apeleaza o multime de metode ale interfetei.
- Cind Windows creaza un proces nou, el foloseste un sablon pentru a genera noua instanta obiect. La momentul crearii, sunt asignate valori ale atributelor.
- Tabelul urm.prezinta pe scurt atributele unui proces obiect.



(a) Process object

Process ID	A unique value that identifies the process to the operating system.
Security Descriptor	Describes who created an object, who can gain access to or use the object, and who is denied access to the object.
Base priority	A baseline execution priority for the process's threads.
Default processor affinity	The default set of processors on which the process's threads can run.
Quota limits	The maximum amount of paged and nonpaged system memory, paging file space, and processor time a user's processes can use.
Execution time	The total amount of time all threads in the process have executed.
I/O counters	Variables that record the number and type of I/O operations that the process's threads have performed.
VM operation counters	Variables that record the number and types of virtual memory operations that the process's threads have performed.
Exception/debugging ports	Interprocess communication channels to which the process manager sends a message when one of the process's threads causes an exception. Normally these are connected to environment subsystem and debugger processes, respectively.
Exit status	The reason for a process's termination.

- Un proces Windows trebuie sa contina cel putin un thread. Acel thread, poate crea alte thread-uri.
- Intr-un sistem multiprocessor, mai multe thread-uri ale aceluiasi proces pot fi executate in paralel.
- Fig. urm. descrie structura unui obiect thread si tabeul urm. descrie atributele obiectului thread.
- Observam ca anumite atribute ale threadului le inlocuiesc pe cele ale procesului. In acele cazuri, valoarea atributului thread-ului este dedusa (mostenita) din cea a procesului.(De exemplu, *thread processor affinity).*
- Obs. ca unul dintre atributele unui obiect thread este **context**. Aceasta informatie permite thread-urilor sa poata fi suspendate si apoi reluate.
- Prin modificarea contextului unui thread, atunci cind el este suspendat, se poate modifica comportamentul acestuia.



(b) Thread object

Thread ID	A unique value that identifies a thread when it calls a server.
Thread context	The set of register values and other volatile data that defines the execution state of a thread.
Dynamic priority	The thread's execution priority at any given moment.
Base priority	The lower limit of the thread's dynamic priority.
Thread processor affinity	The set of processors on which the thread can run, which is a subset or all of the processor affinity of the thread's process.
Thread execution time	The cumulative amount of time a thread has executed in user mode and in kernel mode.
Alert status	A flag that indicates whether a waiting thread may execute an asynchronous procedure call.
Suspension count	The number of times the thread's execution has been suspended without being resumed.
Impersonation token	A temporary access token allowing a thread to perform operations on behalf of another process (used by subsystems).
Termination port	An interprocess communication channel to which the process manager sends a message when the thread terminates (used by subsystems).
Thread exit status	The reason for a thread's termination.

- **Planificarea firelor de execuţie.**
- Fiecare „thread” se poate afla într-una dintre stările: **ready, standby, run, wait, transition** şi **terminate**.
- Firul de execuţie de prioritate cea mai înaltă trece în starea **standby,** adică el va fi următorul care va primi serviciile unui procesor.
- În sistemele multi-procesor, pot exista mai multe astfel de thread-uri, corespunzător numărului de procesoare ale sistemului.
- Daca prioritatea procesului aflat in starea **standby** este suficient de inalta, thread-ul aflat in executie pe acel procesor poate fi fortat de catre cel aflat in **standby**. In caz contrar, thread-ul aflat in standby asteapta pina cind thread-ul aflat in executie se blocheaza sau isi epuizeaza cuanta de timp.
- Cind nucleul executa o schimbare de context, thread-ul aflat in **standby** intra in starea **run.**
- Un thread aflat în starea **run**, va sta în această stare până când se întâmplă una dintre următoarele situaţii:
  - va fi forţat de către un alt fir de execuţie de o prioritate mai înaltă;
  - cuanta lui de timp s-a epuizat;
  - se termină execuţia lui;
  - va cere execuţia unui apel de sistem (se blocheaza).

- Un thread intra in starea **Wait** intr-una dintre urm. situatii: (1) este blocat pe un eveniment (de ex. o I/O);

  (2) asteapta in mod voluntar, in scopuri de sncronizare;

  (3) un subsistem de mediu directioneza suspendarea thread-ul.

  Cind conditia pt. care a intrat in starea de asteptare este satisfacuta, thread-ul trece in starea **Ready**, daca toate resursele de care are nevoie sunt disponibile.
- Un thread intra in starea **Transition** cind, dupa o asteptare nu are disponibile toate resursele cerute. De **examplu**, stiva thread-ului nu mai are suficient spatiu de memorie. Cind are toate resursele disponibile, el trece in starea **Ready**.
- **Terminated:**
- - Thread-ul poate fi terminat de catre el insusi, de catre un alt thread, sau cind se termina procesul parinte.
- - El poate fi sters definitiv din sistem sau poate fi retinut de catre executiv pt. o reinitializare viitoare.

- Pentru a determina ordinea de execuţie, dispecerul sistemului foloseşte o **schemă de prioritate** pe 32 de niveluri.
- Priorităţile sunt împărţite în două clase:

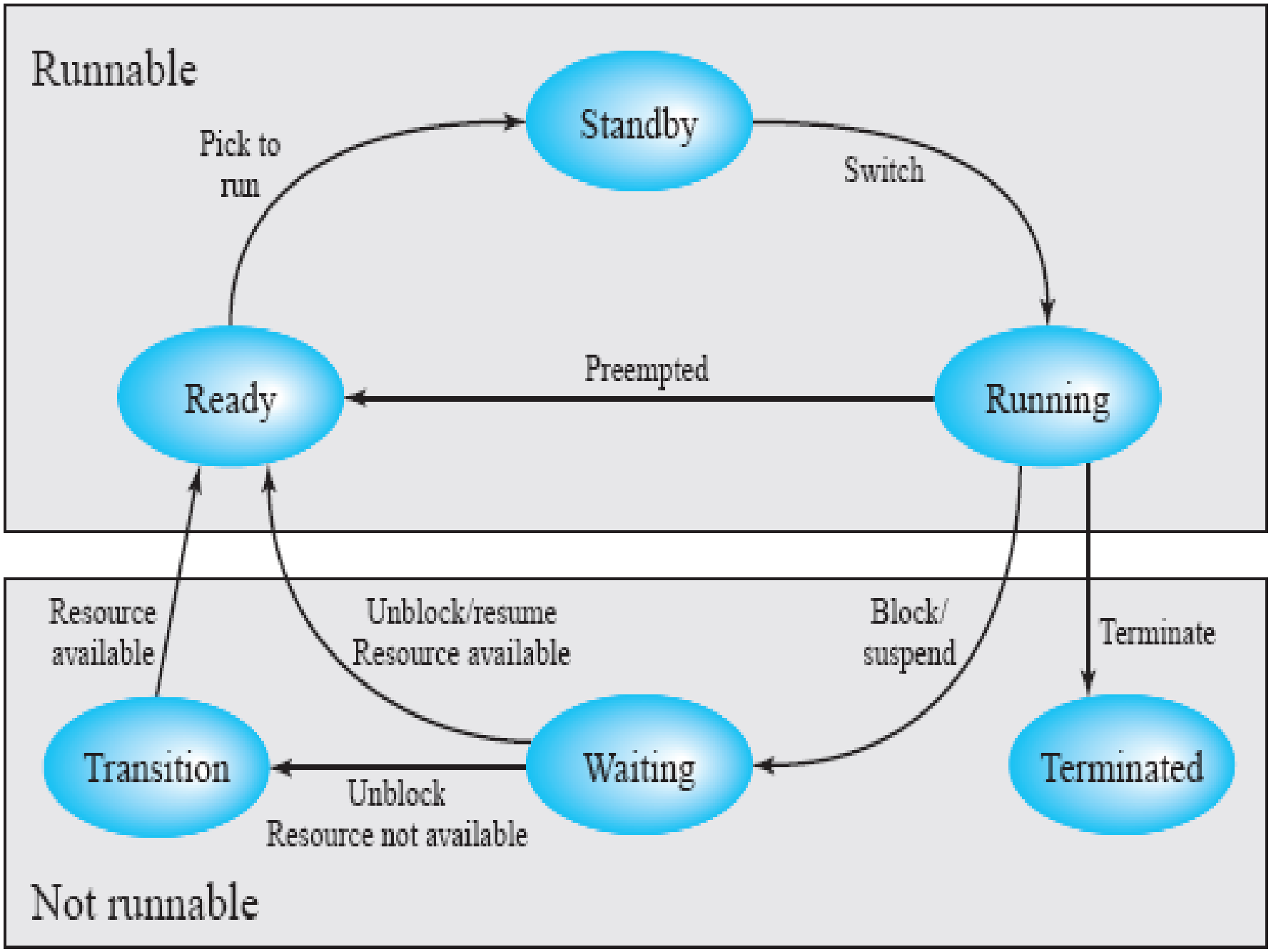
  - **clasa aplicaţiilor în timp real:** firele de execuţie care au o prioritate de clasă cea mai înaltă, cu numere cuprinse între 16 şi 31;

  - **„thread”-uri de clasă variabilă**, au numere de clasă cuprinse între 0 şi 15.
- Dispecerul foloseşte câte o coadă pentru fiecare nivel de prioritate şi atunci când se pune problema selectării unui nou fir de execuţie, acesta va traversa sistemul de cozi, de la cea de nivel cel mai înalt, către cea de nivel cel mai scăzut, până când găseşte un thread gata de execuţie (aflat în starea **standby**).
- Dacă nu este găsit nici un astfel de „thread”, dipecerul va executa un „thread” special, care îl va trece în stare de aşteptare(idle).
- De asemenea, când un thread gata de execuţie are o preferinţă pentru un anumit procesor, altul decât cel liber, dispecerul va trece la un alt thread, aflat în starea **standby.**

- Când un „thread” în execuţie, din clasa celor de prioritate variabilă îşi termină cuanta de timp alocată lui, prioritatea lui este scăzută, până la nivelul imediat, al „thread-ului” aflat în aşteptare cu nivelul cu prioritatea cea mai slabă.
- Această metodă, realizează o servire echitabilă a proceselor care utilizează mult CPU(orientate spre calcule).
- De asemenea, când un astfel de „thread” intră în starea **wait**, dispecerul recalculează prioritatea acestuia în funcţie de tipul operaţiei de I/O efectuate.
- Prin acest mod de lucru, în cazul thread-urilor interactive, se urmăreşte să se obţină un timp de răspuns rezonabil.
- **Obs.**

  1. Planificarea poate să apară atunci când un thread intră într-una dintre stările **ready, wait** sau **finish** sau când trebuie schimbată prioritatea sau preferinţa către un anumit procesor ale acestuia.

  2. Prin modul de organizare a planificării, se urmăreşte servirea prioritară a aplicaţiilor în timp real. Deoarece acestea au priorităţi mai înalte, atunci când un astfel de thread este în starea **ready** şi toate procesoarele sunt ocupate, unul dintre threadurile de clasă variabilă va fi forţat să elibereze procesorul, care va servi aplicaţia în timp real.
- Fig. urm. prezinta tranzitiile efectuate de thread-uri.


Runnable
Standby
Pick to run
Switch
Ready
Preempted
Running
Resource available
Unblock/resume
Resource available
Block/ suspend
Terminate
Transition
Waiting
Terminated
Unblock
Resource not available
Not runnable

Facilitatea de apel de procedură locală

- Apelul de procedura locala(LPC-**L**ocal **P**rocedure **C**all).
- Implementarea sistemului Windows foloseşte modelul client-server pentru implementarea subsistemelor de mediu.
- Modelul client-server este utilizat pentru a se implementa o multitudine de servicii pe care se spijină subsistemele de mediu, printre care: administrarea securităţii, utilizarea „spooling-ului” pentru tipărire, servicii de web, sistemul de fişiere partajate în reţea etc.
- LPC este utilizată de sistemul de operare pentru

  - a transmite mesaje între un proces client şi un proces server, în interiorul aceluiaşi calculator.

  - pentru comunicaţia între subsistemele Windows.

  Obs. LPC este similar RPC(folosit pentru comunicaţia într-o reţea), dar LPC este optimizat pentru a fi folosit în interiorul unui sistem.
- Procesul server face public un obiect port de conectare care este vizibil global.
- Când un proces client doreşte servicii de la un subsistem, el deschide un manipulator la acel port şi transmite o cerere de conexiune la acel port.
- Serverul creează un canal şi returnează un manipulator pentru acel canal: canalul constă dintr-o pereche de porturi de comunicaţie, unul pentru mesaje de la client la server iar celălalt de la server la client.

- Când este creat un canal LPC, trebuie să fie specificată una dintre următoarele tehnici:

  - Transmiterea mesajelor de dimensiune mică se realizează prin punerea mesajelor într-o coadă, de unde sunt luate de către procesele cărora le sunt destinate.

  - Pentru transmiterea mesajelor de dimensiune mare, se utilizează o zonă de memorie partajată; în acest caz, este creat un obiect prin care sunt gestionaţi pointerii către mesaje şi dimensiunea lor. Aceste informaţii sunt utilizate de procesele care primesc mesajele respective să le preia direct.

  - Utilizarea unei API prin care se poate citi/scrie direct din spaţiul de adrese al unui proces.

Sistemul de fişiere

- Sistemul de fişiere sub MS-DOS se bazează pe tabela FAT. În cazul sistemului de fişiere cu FAT pe 16 biţi există o serie de dezavantaje: fragmentarea internă, limitarea dimensiunii la 2 Go etc.
- Odată cu apariţia tabelei FAT pe 32 de biţi, s-au rezolvat o parte din aceste probleme.
- Odată cu Windows XP ,apare sistemul de fişiere NTFS(**N**ew **T**echnology **F**ile **S**ystem); acesta oferă o serie de facilităţi:

  - foloseşte adrese de disc pe 64 de biţi şi poate suporta partiţii de până la 2^64 bytes ;

  - posibilitatea folosirii caracterelor Unicode în numele de fişiere;

  - refacerea volumelor dupa caderea sistemului sau defectarea discurilor;

  - permite folosirea numelor de fişiere de până la 255 de caractere, inclusiv spaţii şi puncte; face distincţie între litere mari şi mici în cadrul numelor de fişiere şi păstrează informaţii de timp referitoare la fişier;
  - oferă posibilitatea managementului dinamic al sectoarelor;

  - protectia si securitatea datelor;

  - permite utilizarea fişierelor cu seturi multiple de date;

  - fişiere de orice dimensiune;

  - fluxuri de date multiple;

  - compresia fişierelor etc.
- Windows XP oferă suport şi pentru alte sisteme de fişiere, cum este de exemplu FAT.

- Entitatea NTFS este **volumul; un v**olum poate ocupa unul sau mai multe discuri logice.
- Sub NTFS, unitatea de alocare este **clusterul;**

  - este format dintr-un număr de sectoare (a carui dim. este o putere a lui 2 si de obicei este 512 octeti).

  - dimensiunea clusterului:

    ►este fixată la formatarea discului;

    ►implicit este dim. unui sector de disc, dacă dimensiunea volumului este mai mică de 512 Mo;

    ►1 Ko pentru volume de până la 1 Go;

    ►2 Ko pentru volume de până la 2 Go;

    ►4 Ko pentru volume mai mari.

  In tabela urm. sunt prez. relatiile dintre dimensiunea volumului, nr. de sectoare/cluster si dimensiunea clusterului.

  - Clusterele alocate unui fisier nu trebuie sa fie contigue; astfel, un fisier se poate fragmenta pe un disc.

  - Un cluster poate avea cel mult 2^{16} octeti.
- In mod curent, dimensiunea maxima permisa a unui fisier sub NTFS este de 2^{32} clustere (2^{48} octeti).
- Deoarece dimensiunea clusterului sub NTFS este mult mai mică decât cea sub FAT, fragmentarea internă a discului este mult mai mică.

Volume Size	Sectors per Cluster	Cluster Size
512 Mbyte	1	512 bytes
512 Mbyte–1 Gbyte	2	1K
1 Gbyte–2 Gbyte	4	2K
2 Gbyte–4 Gbyte	8	4K
4 Gbyte–8 Gbyte	16	8K
8 Gbyte–16 Gbyte	32	16K
16 Gbyte–32 Gbyte	64	32K
> 32 Gbyte	128	64K

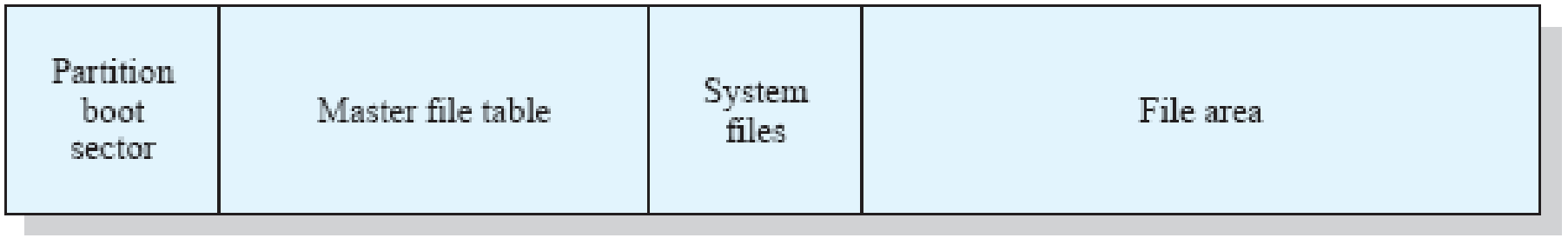
- Pentru adresare pe disc, NTFS foloseşte numărul logic de cluster (LCN-**L**ogical **C**luster **N**umber).
- LCN se atribuie crescator de la începutul spre sfârşitul discului. Folosind acest sistem, SO poate calcula adresa relativa (offset-ul) a unui cluster, prin înmulţirea LCN cu dimensiunea unui cluster.
- Utilizarea clusterelor NTFS, face alocarea spatiului ptr. fisiere independenta de dimensiunea sectorului. Aceasta permite NTFS:

  - sa suporte usor discuri nonstandard, care nu au dim. sect. de 512 octeti;

  - sa suporte eficient discuri si fisiere de dim. mare, folosind clustere mari (Eficienta alocarii este datorata numarului mai mic de clustere si implicit a intrarilor in tabele).
- Sistemul Windows defineşte fişierul ca un flux de octeţi, care nu poate fi accesat decât secvenţial. Pentru fiecare instanţă a unui fişier deschis, există un pointer memorat pe 64 de biţi către următorul octet care urmează să fie accesat. Când fişierul este deschis, pointerul este setat cu 0 şi după fiecare operaţie de scriere/citire a k octeţi, valoarea pointerului este mărită cu k.

- Ca şi celelalte resurse gestionate de sistemul Windows şi gestionarea fişierelor este orientată pe obiecte.
- Un fişier este un obiect care are mai multe atribute. Fiecare atribut este un flux de octeţi, care poate fi creat, şters, citit sau scris.
- Atributele standard(numele fişierului, data creerii, descriptorul de securitate etc.) sunt comune tuturor fişierelor, pe când anumite atribute particulare sunt specifice unei anumite clase de fişiere (director, de **exemplu**).
- Inclusiv datele continute in fisier sunt un atribut.

- **Structura de disc NTFS**
- La formatarea unei partiţii (volum) cu sistemul de fişiere NTFS se creează o serie de fişiere sistem, dintre care cel mai important este fişierul **Master File Table (MFT)**, care conţine informaţii despre toate fişierele şi directoarele de pe volumul NTFS.
- Fiecare fişier sub NTFS este descris prin una sau mai multe înregistrări din (**M**aster **F**ile **T**able). (fig. urm).



- **Organizarea informatiilor pe un volum NTFS**.
- Fiecare element de pe un volum este fisier si fiecare fisier consta dintr-o colectie de atribute (inclusiv datele continute in fisier sunt tratate ca un atribut).
- Din fig. ant. , obs. ca un **volum NTFS este format din 4** regiuni(zone).
- Primele sectoare ale volumului sunt ocupate de catre **partitia de boot** (pina la 16 sectoare lungime); aceasta contine:

  - informatii despre organizarea volumului si structuri ale sistemului de fisiere;

  - o secventa de cod care, incarcata in memorie si lansata in exec. va incarca sist. de operare (fisierul de boot-are);

  - informatii de “startup”.

- **Tabela MFT** (**master file table- MFT)** contine informatii despre toate fisierele si foldere-le (directoare) de pe acest volum NTFS.
- MFT este o lista a tuturor fisierelor si atributelor lor de pe acest volum NTFS, organizata ca o multime de linii, intr-o structura de baza de date relationala.
- Fiecare fisier in NTFS este descris prin una sau mai multe inregistrari din MFT.
- Dimensiunea înregistrării este cuprinsă între 1 Ko şi 4 Ko, fiind fixată la crearea sistemului.
- Atributele de fişier sunt păstrate în MFT atunci când dimensiunea lor permite să fie memorate în intrarea corespunzătoare din MFT, sau în zone auxiliare de pe HDD, exterioare fişierului MFT şi asociate intrării din MFT a fişierului.
- Ptr. un fis. de dim. mici, toate atributele (inclusiv) datele sunt memorate in MFT.
- Atributele de dim. mare (**atribute nerezidente)** sunt stocate intr-una sau mai multe zone contigue de extensie de pe disc; in MFT exista cite un pointer catre fiecare astfel de zona.
- Fiecare inregistrare din MFT consta dintr-o multime de atribute care serveste la definirea caracteristicilor fisierului (sau folder-ului) resp. si a continutului sau. Aceste atribute sunt:

- **Informatii standard,** contine: atribute de acces (read-only, read/write, etc.); marca timpului (data creerii, data ultimei modificari a fisierului) ; contorul de legaturi la fisier.
- **Lista de atribute**: este utilizata ptr. localizarea atributelor fisierului.
- **Numele fisierului**.
- **Descriptorul de securitate**: Specifica proprietarul fis. si cine are acces la el.
- **Date**: Continutul fisierului.
- **Indexul radacinii**: folosit pt. a implementa foldere.
- **Alocare de index**: folosit pt. a implementa foldere.
- **Informatii despre volum**: de ex. versiunea si numele volumului.
- **Harta de biti**: Indica inregistrarile utilizate.

- Fiecare fişier dintr-un volum NTFS are un **identificator unic**, numit **referinţă** la fişier şi este reprezentat pe 64 de biţi, dintre care 48 sunt alocaţi **numărului de fişier**, iar ceilalţi 16 conţin **numărul de secvenţă**.
- Numărul de fişier este numărul înregistrării corespunzătoare fişierului din tabela MFT.
- Numărul de secvenţă este incrementat de fiecare dată când o intrare din MFT este reutilizată. Acest număr permite NTFS să execute anumite verificări, cum ar fi, de **exemplu** referenţierea unei intrări în tabelă, alocată unui alt fişier, după ce a fost utilizată pentru un fişier care a fost şters.

- Dupa MFT este o regiune (de lungime 1 Mb), ce contine **sistemul de fisiere**. Printre fisierele din aceasta regiune sunt urm:

  • **MFT2:** Aflat pe primele 3 linii ale lui MFT, folosit ptr. a garanta accesul la MFT in cazul unei defectiuni aparute pe un singur sector.

  • **Fisierul de jurnalizare** (**Log file):** Contine o lista a tranzactiilor efectuate; este folosit ptr. refacerea NTFS.

  • **Harta de biti a clusterelor** (**Cluster bit map):** Indica ce clustere sunt folosite.

  • Tabela de definire a atributelor (**Attribute definition table):**

  - defineste tipurile de atribute permise pe acest volum;

  - Indica daca ele pot fi indexate si daca pot fi refacute in timpul unei operatii de refacere a sistemului.

  • Fişierul clusterelor defecte conţine adresele de pe disc a zonelor care nu pot fi utilizate.

- **Fiş. jurnal** înreg. toate actualizările de metadate pentru sist. de fişiere.
- **Fişierul volum** conţine numele a volumului, versiunea de NTFS sub care a fost formatat volumul, şi un fanion, care spune dacă volumul a fost corupt şi trebuie să fie verificat pentru consistenţă.
- **Tabela de def. a atributelor** indică tipurile de atribute utilizate în operaţiunile de volum şi ceea ce poate fi efectuat de fiecare dintre ele.
- **Directorul rădăcină** este dir. de nivel cel mai inalt în ierarhia sist. de fişiere
- **Fişierul bitmap** care indică ce clustere de pe un volum sunt alocate pentru fişiere şi care sunt libere.
- **Fişierul de boot** conţine codul de pornire pentru Windows şi trebuie să fie amplasat la o adresă de disc, astfel încât să poată fi găsit cu uşurinţă. Fişierul de boot conţine, de asemenea, adresa fizică a MFT.
- **Fişierul clusterelor defecte** păstrează toate zonele deteriorate de pe volum; NTFS utilizează această înregistrare pentru recuperare in caz de erori.

- **Arborele NTFS B+.**
- Ca şi celelalte sisteme de fişiere, spaţiul de nume NTFS este organizat ierarhic.
- Fiecare director foloseste o structura de date numita arbore B + pentru a stoca indicele de nume de fişier din acel director.
- Structura arborescentă utilizată este arborele B+. Acesta are proprietatea că lungimea oricărui drum de la rădăcină la oricare dintre frunze este aceeaşi.
- Arborele B+ este utilizat deoarece elimină costurile necesare reorganizării structurii de fişiere.
- **Rădăcina index** a directorului, aflată la cel mai înalt nivel al arborelui, conţine pointeri către celelalte elemente ale structurii.
- Fiecare intrare din director, conţine numele şi o serie de informaţii despre fişier(timpul ultimei prelucrări, dimensiunea fişierului etc.), copiate din tabela MFT. Astfel, se poate lista conţinutul oricărui director, fără a se consulta tabela MFT, care, datorită complexităţii acesteia, este o operaţie costisitoare.

- Fişierele *metadata* sunt structurile de date folosite de NTFS pentru accesul şi manag. fişierelor. Acest sistem de fişiere se bazează pe principiul „totul este fişier”. Astfel, descriptorul de volum, informaţia de boot, înregistrări ale sectoarelor defecte etc. sunt toate stocate în fişiere. Fişierele care stochează informaţiile metadata ale NTFS sunt:

  - Primul fişier este MFT.

  - Al doilea fişier este folosit pt. recuperarea inf., în cazul cînd MFT este deteriorat si conţine o copie a primelor 16 intrări ale MFT.

- În NTFS, putem identifica următoarele tipuri de fişiere:
  - *fişiere sistem*: sunt fişierele descrise în tabelul de mai sus şi conţin informaţii (metadata) ce sunt folosite numai de către sistemul de operare.
  - *fişiere cu seturi multiple de date (Alternate Data Streams - ADS)*: sunt fişiere care pe lângă setul de date principal (şi implicit), mai conţin şi alte seturi distincte de date. Toate aceste seturi de date sunt reprezentate prin atribute de tip *Data*.
  - *fişiere arhivate*: NTFS poate arhiva şi dezarhiva fişierele în momentul efectuării operaţiilor de scriere şi, respectiv citire a datelor din ele. Acest mecanism este invizibil aplicaţiilor ce utilizează astfel de fişiere.
  - *fişiere criptate:* EFS (Encrypted File System) oferă suport pentru a stoca fişiere criptate pe un volum NTFS. Criptarea este transparentă faţă de utilizatorii care au încriptat fişierul. Accesul celorlalţi utilizatori nu este permis la aceste fişiere.
  - *fişiere „rare" (sparse files):* sunt fişiere în care informaţia scrisă nu se găseşte într-o singură zonă contiguă, ci zonele în care s-au scris date alternează cu zone mari în care nu s-au scris („găuri"). NTFS permite setarea unui atribut special al acestor fişiere, prin care se indică sistemului de I/E să aloce spaţiu pe disc numai pentru zonele efectiv scrise din fişier.
  - *fişiere de tip „hard-link":* sunt fişiere speciale introduse de NTFS. Aceste fişiere permit ca un fişier să poate fi accesat prin mai multe căi fără ca datele efective să fie duplicate. Dacă ştergem un fişier la care există şi o altă legătură, datele nu vor fi şterse de pe disc, până când nu se şterg toate legăturile.

- În ceea ce priveşte drepturile de acces, în NTFS ele sunt gestionate prin *liste de control al accesului (ACL)*. Aceste ACL-uri conţin informaţii care definesc pentru fiecare utilizator sau grup de utilizatori drepturile pe care le au asupra unui fişier. Drepturile de acces se numesc *permisiuni.*
- NTFS-ul defineşte 6 astfel de permisiuni de bază, numite *permisiuni speciale*.

Permisiune	**Drepturi fişiere**	**Drepturi directoare**
*Read (*R)	Citire conţinut fişier	Citire conţinut director
Write (W)	Modificare conţinut fişier	Modificare conţinut director (creare fiş. sau subdir.)
Execute (X)	Executare program	Traversare structură subdir.
Delete (D)	Stergere fişier	Ştergere director
*Change Permissions(*P)	Schimbare drepturi de acces pt. fişier	Schimbare drepturi de acces pt. director
*Take Ownership(*O)	Schimbare proprietar	Schimbare proprietar

•

- **Recuperarea (refacerea) sistemului**
- În multe sisteme de fişiere simple, o pană de curent, la momentul nepotrivit poate deteriora structurile sistemului de fişiere, a.i. întregul volum este compromis.
- În NTFS, toate actualizările structurilor de date ale sistemul de fişiere sunt efectuate prin intermediul tranzacţiilor.
- Înainte ca o structura de date sa fie modificata, tranzacţia scrie o înregistrare de tip log, care conţine informatii de tip **redo** şi **undo**; după ce structura de date a fost schimbata, tranzacţia scrie o înregistrare în fisierul de log, cu semnificatia că tranzacţia a reuşit.
- După un accident, prin prelucrarea înregistrărilor din jurnal, sistemul poate restabili structurile de date ale sistemului de fişiere pentru a reveni intr-o stare consistenta.
- Mai intii se refac operaţiile ptr. tranzactiile realizate, apoi se anuleaza tranz. care nu au fost realizate cu succes.
- Ptr. aceasta sunt necesare punctele de verificare (checkpoint); periodic o inregistrare care contine un punct de verificare este scrisa in fisierul de log.
- Sistemul nu mai are nevoie de inregistrarile dintr-un fiser de log de dinaintea unui punct de verificare, deci acesta nu poate creste ca dim. peste o anumita limita.
- Prima dată după pornirea sistemului de operare, cind un volum NTFS este accesat, automat se executa operatia de recuperare NTFS.

Administratorul I/O

- este responsabil pentru sistemul de fişiere, administrarea memoriei, driverele de unităţi şi cele de reţea. El urmăreşte care sistem de fişiere este încărcat şi administrează bufferele pentru efectuarea operaţiilor de I/O.
- Administratorul I/O converteşte cererile pe care le primeşte într-o formă standard, numită pachet cerere de I/O(IRP-**I**/O **R**equest **P**acket), pe care le transmite pentru prelucrare driverului corespunzător.
- După ce operaţia este efectuată, administratorul I/O primeşte IRP-ul de la driverul care a efectuat operaţia.
- De asemenea, împreună cu administratorul memoriei virtuale, administratorul I/O realizează proiecţia în spaţiul de adrese virtuale a informaţiilor de pe disc.

Subsistemele de mediu

- reprezintă procese executate în mod utilizator, care permit execuţia unor aplicaţii dezvoltate sub alte sisteme de operare(MS-DOS, Windows pe 16 biţi şi 32 de biţi).
- fiecare subsistem furnizează o interfaţă utilizatorului, prin care se pot lansa aplicaţii realizate în cadrul sistemelor de operare respective.
- Windows foloseşte subsistemul Win32 ca mediu principal de operare pentru lansarea proceselor. Când este executată o aplicaţie, Win32 apelează administratorul memoriei virtuale pentru a încărca fişierul executabil, corespunzător aplicaţiei respective. Administratorul memoriei returnează un mesaj către Win32, prin care specifică tipul aplicaţiei, pe baza căruia este ales un anumit subsistem de mediu sub care se va executa procesul respectiv.
- Subsistemele de mediu folosesc facilitatea LPC pentru a obţine serviciile nucleului pentru aplicaţiile lansate. Astfel, se asigură robusteţea sistemului, deoarece parametrii transmişi printr-un apel de sistem sunt verificaţi, înainte de a fi cerută o anumită rutină.
- De asemenea, o aplicaţie executată sub un anumit mediu, nu poate apela o rutină corespunzătoare altui mediu.

- **Regiştrii Windows XP** sunt utilizaţi pentru păstrarea unor informaţii de configurare ale sistemului(preferinţe implicite ale utilizatorilor, instalări de soft-uri, securitate etc.). Deoarece informaţia din regiştri este necesară pentru autolansarea în execuţie a sistemului(boot-are), administratorul regiştrilor este implementat ca o componentă a executivului. Boot-area se realizează pe baza conţinutului regiştrilor şi, o dată cu aceasta se face şi o actualizarea a conţinutului acestora. De asemenea, anumiţi regiştri sunt modificaţi odată cu instalarea unor componente software.
- **Bootarea Windows** începe cu exec. BIOS, care este rezident în ROM. BIOS identifică unităţile sistemului şi încarcă şi execută programul de "bootstraping" de pe primul sector al discului primar. Acest program va încărca programul NTLDR, care este utilizat pentru a încărca biblioteca HAL, nucleul şi regiştrii corespunzători operaţiei de bootare. Pe baza conţinutului acestor regiştri, se determină care drivere de unităţi sunt necesare pentru bootare şi acestea sunt încărcate. În final, NTLDR începe execuţia nucleului.
- Nucleul iniţializează sistemul şi crează două procese. Procesul **sistem** conţine toate firele interne de lucru şi el va fi executat totdeauna în mod protejat. Primul proces care poate fi lansat în mod utilizator este SMSS. Acesta face anumite iniţializări ale sistemului: includerea fişierelor de paginare, încărcarea driverelor de unităţi, crearea proceselor WINLOGON şi CSSR. CSSR reprezintă sub-sistemul Win32. WINLOGON crează restul sistemului, inclusiv subsistemul de securitate. Sistemul optimizează procesul de bootare, prin pre-încărcarea unor fişiere de pe disc, care au fost salvate în boot-ările anterioare

- **Rularea aplicaţiilor MS-DOS.** Iniţial, Windows a constituit o extensie a sistemului MS-DOS. Multe programe scrise sub MS-DOS trebuie să fie executate sub Windows, care manipulează aceste aplicaţii prin intermediul unei aplicaţii speciale, **maşina virtuală DOS**(VDM-**V**irtual **D**OS **M**achine). O maşină virtuală este un bloc de memorie configurat să efectueze operaţii DOS, ca şi cum blocul respectiv ar fi un computer care rulează sub vechiul DOS. Astfel, VDM execută instrucţiuni cod maşină ale procesorului Intel 486, furnizează rutine pentru a emula componenta ROM BIOS şi drivere virtuale pentru ecran, tastatură şi porturi de comunicaţie.
- Aplicaţiile DOS se pot accesa printr-un dublu click pe denumirea aplicaţiei sau pe icon, prin introducerea de la tastatură a numelui programului DOS în caseta de dialog **Run** sau prin deschiderea ferestrei **MS-DOS Prompt** şi introducerea comenzii DOS de lansare a programului. Orice program DOS are un fişier de start pe care îl rulează programul respectiv. Numele fişierului de start este în general acelaşi nume cu cel al aplicaţiei, urmat de extensia .exe (fişier executabil) sau .com (fişier de comenzi). În Windows este suficient să se introducă numele fişierului, nu şi extensia.
- Spre deosebire de cele mai multe ferestre, fereastrea **MS-DOS Prompt** nu are o bară de meniu. Ea are totuşi un meniu de Control, care oferă opţiuni corespunzătoare butoanelor de pe bara de instrumente şi butoane de dimensionare a ferestrei. După accesarea MS-DOS Prompt, prompterul de comandă apare pe ecran, adică sistemul se găseşte în mediul DOS, indicând discul implicit; acum se pot introduce comenzile DOS care sunt încorporate în sistemul Windows respectiv.

- Transferul de date între programele DOS şi programele Windows este similar transferului de date între aplicaţiile Windows. Datele copiate sau „ tăiate ” dintr-o aplicaţie se pot lipi la o altă aplicaţie. Pentru aceasta, trebuie mai întâi selectate. În DOS procedura de selectare a datelor cu ajutorul mouse-ului este controlată de facilitatea **QuickEdit** care apare pe pagina de proprietăţi Misc. Programele DOS au cinci pagini de proprietăţi (**Program**, **Font**, **Memory**, **Screen** şi **Misc**(Miscellaneous - diverse)) care se folosesc pentru a controla diferite proprietăţi generale ale aplicaţiei. Accesul la paginile de proprietăţi se face printr-un clic pe butonul **Properties** de pe bara de instrumente. Dacă opţiunea **QuickEdit** din pagina Misc nu este marcată, înainte de a se putea selecta text cu mouse-ul, trebuie apăsat butonul Mark de pe bara de instrumente, înainte de a putea folosi mouse-ul pentru selectarea textului dorit. După ce datele care urmează să fie copiate sau „ tăiate ” au fost selectate, se execută procedurile cunoscute pentru operaţiile de tăiere, copiere şi lipire.
- Programul DOS rulat trebuie să fie închis corect, urmând calea normală de închidere sau comanda de ieşire corespunzătoare. Programul se poate închide prin apăsarea butonului **Close,** se selectarea opţiunii **Close** din meniul **Control** sau efectuarea unui dublu clic pe icon-ul programului de pe bara de titlu.