TỔNG QUAN VỀ HỆ THỐNG FILE NTFS

Hệ thống file NTFS cung cấp sự phối hợp thực hiện, độ tin cậy, khả năng tương thích mà không tìm được ở trong hệ thống file FAT. NTFS được thiết kế để thực hiện nhanh các thao tác chuẩn trên file như đọc, ghi, và tìm kiếm. Thậm chí các hoạt động tiên tiến như khả năng khôi phục hệ thống file, hỗ trợ các đĩa cứng có dung lượng lớn.

Hệ thống file NTFS bao gồm nhiều tính năng an toàn cần đến cho các file server và các máy tính cá nhân high-end trong môi trường tích hợp. NTFS cũng hỗ trợ điều khiển truy cập dữ liệu và các đặc quyền sở hữu, đây là đặc điểu quan trọng cho toàn vẹn dữ liệu.

Dưới đây là một số đặc trưng chỉ được cung cấp bởi hệ thống file NTFS:

 Ta có thể gán các quyền cho các file và thư mục, vì vậy ta có thể chỉ định ai là người được phép quyền truy cập đến file hoặc thư mục đó, và có thể giới hạn khả năng truy cập thông qua các kiểu truy truy cập đã được định nghĩa. Hệ thống file NTFS đưa ra nhiều quyền hơn so với hệ thống file FAT, và ta có thể đặt các quyền cho mỗi người sử dụng hoặc nhóm người sử dụng riêng lẻ. Đặc trưng này thể hiện khả năng bảo mật của NTFS.

 Khả năng khôi phục đã thiết kế cho hệ thống file NTFS là để người sử dụng ít khi phải chạy bất kỳ chương trình sửa chữa đĩa nào trên ổ đĩa NTFS. Trong trường hợp đổ vỡ (crash) hệ thống, thì hệ thống file NTFS dùng file sổ ghi (log file) của nó và thông tin điểm kiểm tra (checkpoint) để tự động khôi phục tính nhất quán của hệ thống file.

 Tổ chức lưu trữ các thư mục theo cấu trúc cây nhị phân làm cho việc truy cập tới các file trên một thư mục lớn nhanh hơn việc truy cập tới các file trên một thư mục cùng kích thước trên hệ thống file FAT.

 Ta có thể nén các file và thư mục riêng lẻ trên đĩa NTFS. Quá trình giải nén sẽ tự động thực hiên khi mở file, và tự động nén lại khi cất hoặc đống file.

 Hỗ trợ đa luồng dữ liệu cho file, tên luồng dữ liệu nhận biết một thuộc tính dữ liệu mới trên file. Một thẻ file có thể mở cùng với mỗi một luồng dữ liệu.

Ngoài các đặc trưng trên thì NTFS5 (phiên bản 5.0) còn cung cấp các đặc trưng sau:

 Khả năng mã hoá.

 Giới hạn không gian đĩa cho từng người sử dụng.

 Các điểm reparse.

 Các điểm gắn đĩa.

 Các file thưa.

 Theo dõi liên kết phân tán.

 Khả năng tạo chỉ mục chung.

 Tên dựa trên Unicode.

 Đánh dấu các cluster hỏng.

 Ghi chép các thay đổi.

 Giải phân mảnh.

 Liên kết cứng.

Tiếp theo em sẽ trình bầy cách tổ chức lưu trữ dữ liệu trên đĩa của hệ thống file NTFS và một số đặc trưng quan trọng của hệ thống file này. Phần này được chia làm hai chương.

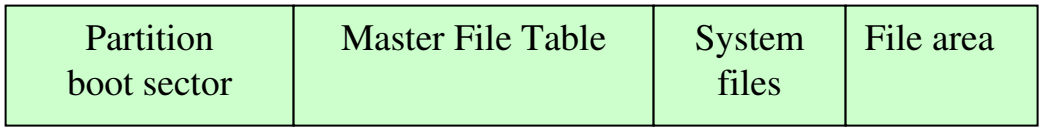
Chương 1 sẽ trình bầy cách tổ chức lưu trữ dữ liệu của NTFS.

Chương 2 trình bầy một số đặc trưng quan trọng trong hệ thống file NTFS. Có thể đưa ra các ví dụ và hình minh hoạ các bước thao tác trên Windows 2000, các hàm API cho khả năng lập trình trên nền Win NT/2K.

Chương 1.

Tổ chức lưu trữ dữ liệu trên hệ thống file NTFS.

I. Cấu trúc một volume sau khi được định dạng bởi NTFS.



Hình minh hoạ một ổ đĩa được định dạng bởi NTFS

- Partition boot sector: là thông tin đầu tiên trên ổ đĩa NTFS được sử dụng khởi động partition. Thông tin này bắt đầu ở sector 0 và có thể dài tới 16 sector, nó bao gồm lệnh nhẩy, ID của hãng chế tạo thiết bị gốc (OEM), bảng tham số chính và bảng tham số mở rộng, chương trình mồi, và cuối cùng là chữ ký của hệ điều hành (luôn là 55AA).

- Master File Table: là bảng file chủ (MFT), là file đầu tiên trên đĩa NTFS. MFT bao gồm các bản ghi chứa thông tin về các file và thư mục trên đĩa. Trong đó có 16 bản ghi đầu tiên dành để lưu trữ các file thông tin đặc biệt (sẽ được đề cập chi tiết ở phần dưới).

- System files: có nhiều loại file hệ thống khác nhau, chúng lưu thông tin đặc biệt để thực thi hệ thống file, các file này không thể nhìn thấy.

- File area: là vùng lưu dữ liệu của file hoặc thư mục khi kích thước dữ liệu (có thể là cả thuộc tính) của file hoặc thư mục quá lớn để điền hết vào một bản ghi MFT.

II. Partition boot sector.

Bảng sau mô tả sector khởi động của đĩa được định dạng bởi NTFS.

Byte Offset	Field Length	Field Name
0x00	3 bytes	Jump Instruction
0x03	8 bytes	OEM ID
0x0B	25 bytes	BPB
0x24	48 bytes	Extended BPB
0x54	426 bytes	Bootstrap Code
0x01FE	WORD	End of Sector Marker

Khi ta định dạng đĩa NTFS thì chương trình định dạng phân 16 sector đầu tiên cho sector khởi động, nó gồm hai cấu trúc:

Khối tham số BIOS (BPB). Khối này chứa thông tin về sự xếp đặt (layout) của đĩa và các cấu trúc file hệ thống.

Chương trình mồi (bootstrap code) mô tả cách tìm và nạp các file khởi động như thế nào.

Trên đĩa NTFS các trường dữ liệu theo sau BPB (BIOS Parameter Block) hình thành BPB mở rộng. Dữ liệu trong các trường này cho phép Ntldr (NT loader program) tìm bảng file chủ (MFT) trong lúc khởi động. MFT không được định vi trước ở sector nào đó (không giống như là FAT16 và FAT 32). Bởi vì MFT có thể được di chuyển nếu ở vị trí thông thường của nó có một sector bị hỏng. Tuy nhiên, nếu dữ liệu của BPB mở rộng bị sai lệch thì không thể xác định được MFT, và Windows NT/2K cho rằng ổ đĩa này chưa được định dạng.

Ví dụ sau minh hoạ sector khởi động của đĩa NTFS được định dạng khi chạy Windows 2000. Ta có thể chia số liệu này thành ba phần:

Các byte 0x00 0x0A là lệnh nhẩy và OEM ID

Các byte 0x0B 0x53 là BPB và BPB mở rộng

Phần còn lại là chương trình mồi và chữ kí của hệ điêu

```
Physical Sector:Cyl 0, Side 1, Sector 1
00000000:EB 52 90 4E 54 46 53 20 -20 20 20 00 02 08 00 00 .R.NTFS ........
00000010:00 00 00 00 00 F8 00 00 -3F 00 FF 00 3F 00 00 00 ........?...?...
00000020:00 00 00 00 80 00 80 00 -4A F5 7F 00 00 00 00 00 ........J.......
00000030:04 00 00 00 00 00 00 00 -54 FF 07 00 00 00 00 00 ........T.......
00000040:F6 00 00 00 01 00 00 00 -14 A5 1B 74 C9 1B 74 1C ...........t..t.
00000050:00 00 00 00 FA 33 C0 8E -D0 BC 00 7C FB B8 C0 07 .....3.....|....
00000060:8E D8 E8 16 00 B8 00 0D -8E C0 33 DB C6 06 0E 00 ..........3.....
00000070:10 E8 53 00 68 00 0D 68 -6A 02 CB 8A 16 24 00 B4 ..S.h..hj....$..
00000080:08 CD 13 73 05 B9 FF FF -8A F1 66 0F B6 C6 40 66 ...s......f...@f
00000090:0F B6 D1 80 E2 3F F7 E2 -86 CD C0 ED 06 41 66 0F .....?.......Af.
000000A0:B7 C9 66 F7 E1 66 A3 20 -00 C3 B4 41 BB AA 55 8A ..f..f....A..U.
000000B0:16 24 00 CD 13 72 0F 81 -FB 55 AA 75 09 F6 C1 01 .$...r...U.u....
000000C0:74 04 FE 06 14 00 C3 66 -60 1E 06 66 A1 10 00 66 t......f`..f...f
000000D0:03 06 1C 00 66 3B 06 20 -00 0F 82 3A 00 1E 66 6A ....f;....:..fj
000000E0:00 66 50 06 53 66 68 10 -00 01 00 80 3E 14 00 00 .fP.Sfh.....>...
000000F0:0F 85 0C 00 E8 B3 FF 80 -3E 14 00 00 0F 84 61 00 ........>.....a.
00000100:B4 42 8A 16 24 00 16 1F -8B F4 CD 13 66 58 5B 07 .B..$......fX [.
00000110:66 58 66 58 1F EB 2D 66 -33 D2 66 0F B7 0E 18 00 fXfX.-f3.f......
00000120:66 F7 F1 FE C2 8A CA 66 -8B D0 66 C1 EA 10 F7 36 f......f..f....6
00000130:1A 00 86 D6 8A 16 24 00 -8A E8 C0 E4 06 0A CC B8 ......$.........
00000140:01 02 CD 13 0F 82 19 00 -8C C0 05 20 00 8E C0 66 ..............f
00000150:FF 06 10 00 FF 0E 0E 00 -0F 85 6F FF 07 1F 66 61 ..........o...fa
00000160:C3 A0 F8 01 E8 09 00 A0 -FB 01 E8 03 00 FB EB FE ................
00000170:B4 01 8B F0 AC 3C 00 74 -09 B4 0E BB 07 00 CD 10 .....<.t........
00000180:EB F2 C3 0D 0A 41 20 64 -69 73 6B 20 72 65 61 64 .....A disk read
00000190:20 65 72 72 6F 72 20 6F -63 63 75 72 72 65 64 00 error occurred.
000001A0:0D 0A 4E 54 4C 44 52 20 -69 73 20 6D 69 73 73 69 ..NTLDR is missi
000001B0:6E 67 00 0D 0A 4E 54 4C -44 52 20 69 73 20 63 6F ng...NTLDR is co
000001C0:6D 70 72 65 73 73 65 64 -00 0D 0A 50 72 65 73 73 mpressed...Press
000001D0:20 43 74 72 6C 2B 41 6C -74 2B 44 65 6C 20 74 6F Ctrl+Alt+Del to
000001E0:20 72 65 73 74 61 72 74 -0D 0A 00 00 00 00 00 00 restart........
000001F0:00 00 00 00 00 00 00 00 -83 A0 B3 C9 00 00 55 AA ..............U.
```

Các trường bắt đầu từ địa chỉ 0x0B, 0x0D, 0x15, 0x18, 0x1A, và 0x1C có ý nghĩa như hệ thống file của FAT16 và FAT32. Các giá trị mẫu tương ứng số liệu ở trên.

Bảng mô tả các trường trong BPB và BPB mở rộng trên đĩa NTFS.

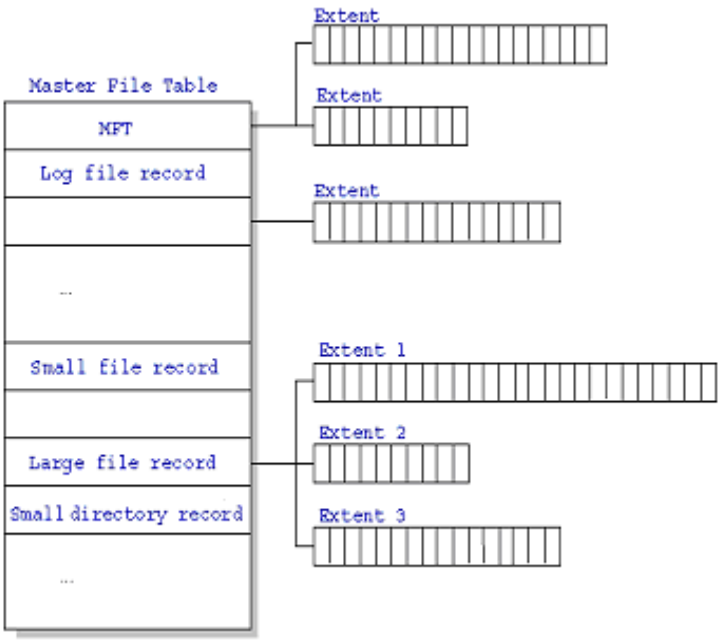
Byte Offset	Field Length	Sample Value	Field Name
0x0B	WORD	0x0002	Bytes Per Sector
0x0D	BYTE	0x08	Sectors Per Cluster
0x0E	WORD	0x0000	Reserved Sectors
0x10	3 BYTES	0x000000	*always 0*
0x13	WORD	0x0000	*not used by NTFS*
0x15	BYTE	0xF8	Media Descriptor
0x16	WORD	0x0000	*always 0*
0x18	WORD	0x3F00	Sectors Per Track
0x1A	WORD	0xFF00	Number Of Heads
0x1C	DWORD	0x3F000000	Hidden Sectors
0x20	DWORD	0x00000000	*not used by NTFS*
0x24	DWORD	0x80008000	*not used by NTFS*
0x28	8 BYTES	0x4AF57F0000000000	Total Sectors
0x30	8 BYTES	0x0400000000000000	Logical Cluster Number for the file $MFT
0x38	8 BYTES	0x54FF070000000000	Logical Cluster Number for the file $MFTMirr
0x40	DWORD	0xF6000000	Clusters Per File Record Segment
0x44	DWORD	0x01000000	Clusters Per Index Block
0x48	8 BYTES	0x14A51B74C91B741C	Volume Serial Number
0x50	DWORD	0x00000000	Checksum

III. Master File Table.

Mỗi một file trên đĩa NTFS tương ứng với một bản ghi (record) trong một file đặc biệt được gọi là bảng file chủ (MFT). NTFS để dành 16 bản ghi đầu tiên của bảng để lưu trữ các file thông tin đặc biệt (metadata files). Bản ghi đầu tiên của MFT mô tả chính MFT, tiếp theo là bản ghi ảnh MFT (MFT mirror record). Nếu bản ghi MFT đầu tiên bị hỏng, thì NTFS đọc bản ghi thứ hai để tìm ra file ảnh MFT. File ảnh của bản ghi đầu tiên giống hệt bản ghi đầu tiên của MFT. Vị trí các đoạn dữ

liệu của MFT và file ảnh MFT được ghi trong sector khởi động. Một bản sao của sector khởi động được định vị ở giữa của ổ đĩa logic. Bản ghi thứ ba của MFT là file sổ ghi (log file), file này được dùng cho công việc khôi phục lại file. Các bản ghi từ thứ 17 và tiếp theo của MFT là mỗi file hoặc thư mục trên đĩa NTFS.

Hình dưới đây là mô tả đơn giản cấu trúc MFT.



Bảng file chủ phân phối không gian lưu trữ thực sự cho mỗi bản ghi file. Các thuộc tính của một file được ghi vào không gian đĩa được phân phối trong MFT. Bên cạnh các thuộc tính file, mỗi bản ghi file còn chứa thông tin về vị trí của nó trong MFT.

Thông thường mỗi một file sử dụng một bản ghi file. Tuy nhiên nếu một file có số lượng thuộc tính lớn hoặc bị phân mảnh, thì file đó có thể cần nhiều hơn một bản ghi file. Trong trường hợp này, bản ghi đầu tiên được gọi là bản ghi file cơ sở, nó lưu trữ vị trí của các bản ghi file

khác mà file yêu cầu. Các file và thư mục nhỏ (điển hình là 1500 bytes hoặc nhỏ hơn) có thể được chứa hoàn toàn trong bản ghi MFT của file.

Ví dụ dưới đây là bản ghi MFT cho một file hoặc thư mục nhỏ.

Thiết kế này làm cho việc truy cập vào file rất nhanh. Đối với hệ thống file FAT thì lại sử dụng bảng phân phối file để đưa ra tên và địa chỉ của mỗi file. Các phần tử thư mục FAT chứa một chỉ mục trong bảng phân phối file. Khi ta muốn xem một file. đầu tiên FAT đọc bảng phân phối file và giả sử rằng file cần tìm là tồn tại, sau đó FAT lấy thông tin file bằng việc tìm chuỗi các khối vị trí đã được gán cho file. Với NTFS thì ngay khi tìm thấy file là ta có thể sử dụng.

Các bản ghi thư mục được lưu lại trong bảng file chủ như là các bản ghi file, chỉ khác là ở đây thông tin chỉ mục được thay cho phần dữ liệu. Các bản ghi thư mục nhỏ thường trú hoàn toàn trong cấu trúc MFT. Các thư mục lớn được tổ chức thành cây nhị phân, và có các bản ghi cùng với các con trỏ trỏ tới các cluster bên ngoài, các cluster này chứa các phần tử thư mục mà không thể chứa cùng với cấu trúc MFT (có thể xem ví dụ minh hoạ trong phần *Tạo chỉ mục chung ở chương 2*).

IV. Các thuộc tính file NTFS.

Hệ thống file NTFS coi mỗi file (hoặc thư mục) như là một tập các thuộc tính file. Mỗi thuộc tính được nhận biết bởi mã kiểu thuộc tính, và

tuỳ chọn một tên thuộc tính. Khi các thuộc tính của một file có thể điền (vừa) vào bản ghi file MFT, thì các thuộc tính này được gọi là các thuộc tính cư trú (resident attributes). Ví dụ các thông tin như tên file, tem thời gian thì luôn luôn nằm trong bản ghi MFT. Khi các thông tin cho một file là quá lớn để điền vào bản ghi file MFT, thì một vài thuộc tính của file sẽ được phân phối một hoặc hơn một cluster trên vùng đĩa khác để phục vụ cho việc lưu trữ các thuộc tính này. Các thuộc tính này được gọi là các thuộc tính không cư trú (nonresident attributes). NTFS tạo danh sách thuộc tính để mô tả vị trí của tất cả các bản ghi thuộc tính.

Bảng dưới đây liệt kê tất cả các thuộc tính hiện tại đã được định nghĩa bởi hệ thống file NTFS. Danh sách thuộc tính này có thể mở rộng ra, có nghĩa là các thuộc tính khác có thể được định nghĩa trong tương lai.

Kiểu thuộc tính	Mô tả
Thông tin chuẩn	Bao gồm các thông tin như tem thời gian và đếm liên kết (link count).
Danh sách thuộc tính	Liệt kê vị trí của tất cả các bản ghi thuộc tính (chỉ cho các thuộc tính không thể điền vào bản ghi MFT).
Tên File	Là một thuộc tính có thể được lặp lại cho cả tên file dài và tên file ngắn. Chiều dài của tên file dài có thể lên tới 255 ký tự Unicode, tên file ngắn có dạng 8.3. Các tên thêm vào là các liên kết cứng, tên được yêu cầu bởi POSIX có thể chứa các thuộc tính tên file thêm vào.
Mô tả bảo mật	Mô tả người chủ sở hữu và người có thể truy cập.
Dữ liệu	Chứa dữ liệu file. NTFS cho phép nhiều thuộc tính dữ liệu trên một file. Thông thường mỗi một file có một thuộc tính dữ liệu không tên. Một file cũng có thể có một hoặc nhiều hơn các thuộc tính dữ liệu có tên, mỗi thuộc tính dữ liệu có tên này dùng một cú pháp riêng.
Định danh đối tượng	Định danh file duy nhất trên đĩa. Được dùng bởi dịch vụ theo dõi liên kết phân tán. Không phải tất cả các file đều có các đinh danh đối tượng.
Logged Tool Stream	Tương tự như một luồng dữ liệu, nhưng các hoạt động được ghi vào file sổ ghi NTFS giống như các thay đổi metadata của NTFS. Thuộc tính này được sử dụng bởi EFS.
Reparse Point	Được sử dụng cho các điểm gắn đĩa. Các điểm reparse cũng được sử dụng bởi các trình điều khiển lọc hệ thống file cài đặt (IFS) để đánh dấu các file thực sự là đặc biệt đối với trình điều khiển đó.
Gốc chỉ mục	Được dùng để thực thi các thư mục và các chỉ mục khác.
Phân phối chỉ mục	Được dùng để thực thi các thư mục và các chỉ mục khác.
Bitmap	Được dùng để thực thi các thư mục và các chỉ mục khác.
Thông tin ổ đĩa	Chỉ được dùng trong file hệ thống. Chứa phiên bản đĩa.
Tên ổ đĩa	Chỉ được dùng trong file hệ thống. Chứa nhãn đĩa.

V. Các file hệ thống NTFS.

NTFS bao gồm các file hệ thống khác nhau, tất cả các file này đều không thể nhìn thấy. Một file hệ thống được sử dụng bởi hệ thống file để lưu trữ các *siêu dữ liệu* (metadata) của nó và để thực thi hệ thống file. Các file hệ thống được đặt lên đĩa bằng một trình tiện ích định dạng (format unility).

Metadata được lưu trong bảng file chủ như sau:

File hệ thống	Tên file	Bản ghi MFT	Mục đích của file
Bảng file chủ	$Mft	0	Chứa một bản ghi file cơ sở cho mỗi file và thư mục trên đĩa NTFS. Nếu thông tin phân phối cho một file hoặc thư mục là quá lớn để điền đầy vào trong một bản ghi, thì bản ghi file khác sẽ được phân phối.
Bảng file chủ 2	$MftMirr	1	Bản sao của bốn bản ghi đầu tiên của MFT. File này đảm bảo việc truy cập được đến MFT trong trường hợp có một sector nào đó bị hỏng.
File sổ ghi (Log file)	$LogFile	2	File này chứa danh sách các bước thực hiện dùng cho khả năng khôi phục NTFS. Kích thước file phụ thuộc vào kích thước của đĩa và có thể là 4MB. File này được dùng bởi Windows NT/2000 để khôi phục tính nhất quán cho NTFS sau một lỗi hệ thống (system failure).
Đĩa (Volume)	$Volume	3	Chứa thông tin về đĩa, ví dụ nhãn đĩa và phiên bản đĩa.
Các định nghĩa thuộc tính	$AttrDef	4	Bảng tên các thuộc tính, số lượng các thuộc tính , và các mô tả thuộc tính
Chỉ mục tên file gốc	$	5	Thư mục gốc.
Cluster bitmap	$Bitmap	6	Một đại diên của ổ đĩa hiên ra mà các cluster đang dùng.
Sector khởi động	$Boot	7	Bao gồm BPB dùng để gắn (mount) đĩa và mã nạp chương trình mồi được dùng nếu đây là đĩa khởi động.
File chứa cluster hỏng	$BadClus	8	Chứa các cluster hỏng của đĩa.
File bảo mật	$Secure	9	Chứa các mô tả bảo mật cho tất cả các file trên một đĩa.
Bảng chữ hoa	$Upcase	10	Chuyển đổi các ký tự thường thành các ký tự Unicode hoa tương ứng.
File mở rộng NTFS	$Extend	11	Được dùng cho các mở rộng lựa chọn khác nhau ví dụ như quotas, reparse point data, and object identifiers.
		12–15	Để dành cho việc sử dụng trong tương lai.

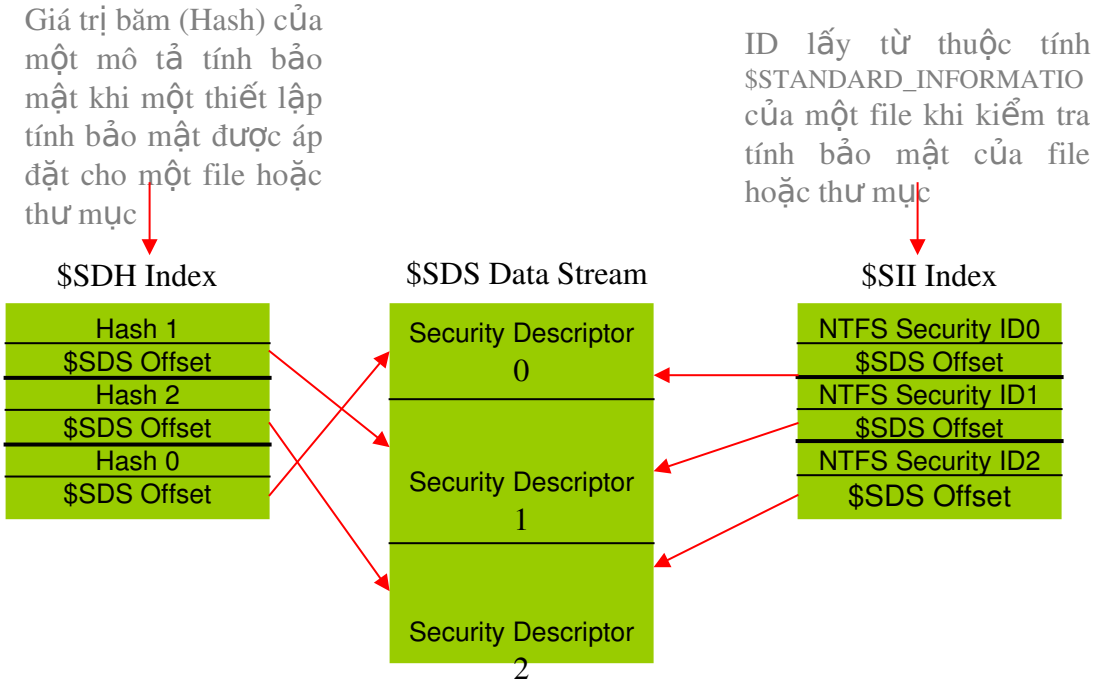
Chương 2.

Các đặc trưng chủ yếu của NTFS/NTFS5.

I. Khả năng bảo mật trong hệ thống file NTFS.

NTFS luôn luôn hỗ trợ tính bảo mật, cho phép người quản trị hệ thống xác định người sử dụng nào có thể và không thể truy cập đến các file và thư mục riêng tư. Trên hệ thống file NTFS mọi file và thư mục đều lưu trữ mô tả tinh bảo mật của nó trong thuộc tính bảo mật của chính nó. Hầu hết trong mọi trường hợp, người quản trị hệ thống có thể áp đặt các thiết lập giống nhau cho toàn bộ một cây thư mục, có nghĩa là nếu việc áp đặt là thư mục gốc thì một bản sao các mô tả tính bảo mật cũng được áp đặt cho các file và thư mục con của thư mục gốc đó. Bản sao này có thể tận dụng không gian đĩa trong môi trường nhiều người sử dụng. NTFS5 tối ưu khả năng tận dụng đĩa cho các mô tả tính bảo mật bằng việc sử dụng một file dữ liệu trung tâm có tên $Secure để lưu chỉ một thể hiện (instance) của mỗi mô tả bảo mật trên một ổ đĩa.

File $Sucure chứa hai thuộc tính chỉ mục là $SDH và $SII, và một luồng dữ liệu có tên $SDS, như chỉ ra ở hình dưới đây.



NTFS5 gán cho mỗi mô tả bảo mật trên đĩa một ID bảo mật nội bộ, và hàm băm (Hash) mô tả tính bảo mật theo một giải thuật băm đơn giản. Các phần tử trong chỉ mục $SDH đặt các giá trị băm (gọi tắt là băm) mô tả tính bảo mật tới vị trí lưu trữ của mô tả bảo mật trong thuộc tính dữ liệu $SDS, và các phần tử chỉ mục $SII đặt các ID bảo mật tới vị trí của mô tả tính bảo mật trong thuộc tính dữ liệu $SDS.

Khi áp đặt một mô tả tính bảo mật cho một file hoặc thư mục, NTFS nhân được một băn của mô tả bảo mật và tìm trong chỉ mục $SDH một tương ứng. NTFS sắp xếp các phần tử chỉ mục theo băm của mô tả bảo mật tương ứng và lưu trữ các phần tử này theo cấu trúc cây nhị phân. Nếu NTFS tìm thấy một tương ứng cho mô tả trong chỉ mục $SDH, thì NTFS xác định vị trí tương đối (offset) của mô tả bảo mật của phần tử từ

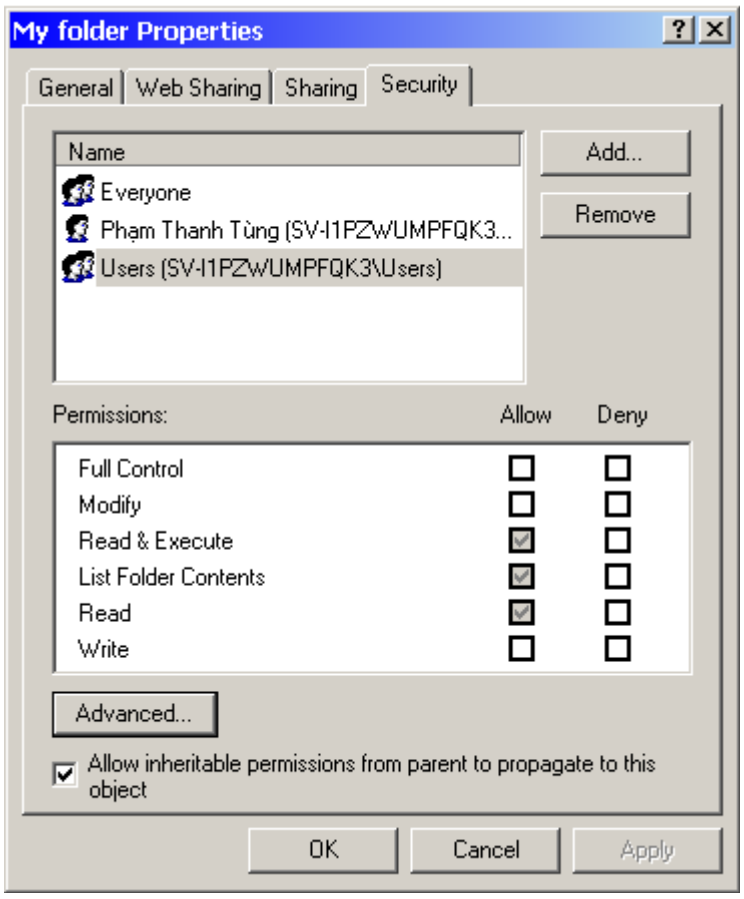
giá trị tương đối của phần tử đó và đọc mô tả bảo mật từ thuộc tính $SDS. Nếu các băn tương ứng tìm thấy, nhưng mô tả bảo mật không tìm thấy, thì NTFS tìm phần tử tương ứng khác trong thuộc tính $SDH. Khi NTFS tìm thấy một tương ứng chính xác, thì file hoặc thư mục mà ta sẽ áp đặt các mô tả bảo mật có thể tham chiếu đến mô tả bảo mật đã tồn tại trong thuộc tính $SDS. NTFS tạo tham chiếu bằng cách đọc định danh bảo mật từ phần tử $SDH và lưu trữ nó trong thuộc tính $standard_information của file hoặc thư mục. Thuộc tính $standard_information là thuộc tính mà tất cả các file và thư mục đều có, nó lưu trữ thông tin cơ bản về một file, bao gồm các thuộc tính của nó và thông tin tem thời gian.

Nếu NTFS không tìm thấy trong chỉ mục $SDH một phần tử có mô tả bảo mật tương ứng với mô tả ta sẽ áp đặt, thì mô tả mà ta sẽ áp đặt là duy nhất đối với ổ đĩa và NTFS gán cho mô tả này một ID bảo mật nội bộ mới. Các ID bảo mật nội bộ có giá trị là 32-bit, trong khi các SID điển hình thì lớn hơn vài lần, do vậy đại diện cho các SID cùng với các ID bảo mật của NTFS cất vào trong không gian thuộc tính $STANDARD_INFORMATION. Sau đó NTFS cộng thêm mô tả bảo mật vào thuộc tính $SDS, là thuộc tính được sắp xếp trong cấu trúc cây nhị phân bằng ID bảo mật của NTFS, và NTFS còn thêm vào chỉ mục $SDS và $SII các phần tử mà nó tham chiếu đến vị trí tương đối của mô tả trong dữ liệu $SDS.

Khi một ứng dụng mở một file hoặc thư mục, NTFS dùng chỉ mục $SII để tìm mô tả bảo mật của file hoặc thư mục đó. NTFS đọc ID bảo mật nội bộ của file hoặc thư mục từ thuộc tính

$STANDARD_INFORMATION của phần tử, sau đó sử dụng chỉ mục $SII của file $Secure để xác định phần tử của ID trong thuộc tính SDS. Vị trí tương đối trong thuộc tính $SDS cho phép NTFS đọc mô tả bảo mật và hoàn thành việc kiểm tra bảo mật. NTFS5 không xoá các phần tử trong file $Secure, thậm chí không có file hay thư mục nào trên đĩa tham chiếu đến các phần tử đó, điều này có nghĩa là không giảm không gian đĩa đã sử dụng cho file $Secure.

Việc sử dụng tạo chỉ mục chung của NTFS5 cho phép các file và thư mục có cùng các thiết lập bảo mật, chia sẻ các mô tả bảo mật một cách hiệu quả. Chỉ mục $SII cho phép NTFS tìm kiếm nhanh một mô tả bảo mật trong file $Secure trong khi thực hiện kiểm tra bảo mật, và chỉ mục $SDH cho phép NTFS quyết định nhanh xem một mô tả bảo mật đã được áp đặt cho một file hay thư mục được lưu trữ trong file $Secure chưa ?, và có thể dùng chung không ?.



*hình minh hoạ thao tác thiết tlập bảo mật cho thư mục **My folder** cho Users trên nền Windows 2000.*

II. Khả năng khôi phục file hệ thống.

NTFS là hệ thống file có khả năng khôi phuc. Nó được thiết kế để khôi phục lại tính nhất quán đối với một đĩa sau một lỗi của CPU, lỗi hệ thống, hoặc lỗi vào/ra. Khả năng này đảm bảo tính nhất quán của đĩa bằng cách sử dụng sổ ghi giao tác chuẩn (standard transaction logging) và các kỹ thuật khôi phục. NTFS khôi phục lại tính nhất quán bằng cách chạy một thủ tục khôi phục. Thủ tục này truy cập thông tin được lưu trong file sổ ghi (log file). Thủ tục khôi phục NTFS là chính xác, đảm bảo ổ đĩa được khôi phục lại trạng thái nhất quán.

NTFS đảm bảo toàn ven ổ đĩa bằng việc thực hiên tự động các hoạt động khôi phục lại đĩa ngay ở lần đầu tiên một chương trình truy cập vào đĩa sau khi máy tính được khởi động lại sau một sự cố.

NTFS cũng sử dụng một kỹ thuật được gọi là sắp đặt lại cluster (cluster remapping) để tối thiểu hiệu ứng của một sector hỏng trên đĩa NTFS.

1. *Khôi phục lại dữ liệu trên đĩa NTFS:*

NTFS xem mỗi hoạt động I/O, hoạt động mà nó sửa đổi một file hệ thống trên đĩa NTFS như là một ***giao tác*** (transaction), và NTFS quản lý mỗi một giao tác như một khối trọn ven. Mỗi lần khởi động thì giao tác có thể là được hoàn thành (làm xong), hoặc trong trường hợp có sự cố đĩa thì quay lại giao tác (ví dụ khi đĩa NTFS trả lại trạng thái ban đầu, trạng thái mà khi giao tác chưa được bắt đầu).

Để đảm bảo rằng một giao tác có thể được hoàn thành hoặc quay trở lại được, NTFS ghi các hoạt động con (suboperations) của một giao tác vào file sổ ghi trước khi nó ghi vào đĩa. Khi một giao tác đã hoàn thành được ghi vào file sổ ghi, NTFS thực hiện các hoạt động con của giao tác trên cache đĩa. Sau khi NTFS cập nhật lại cache, NTFS khẳng định (commit) giao tác đó bằng việc ghi vào sổ ghi toàn bộ giao tác hoàn thành.

Mỗi lần một giao tác được khẳng định, NTFS đảm bảo rằng toàn bộ giao tác đó xuất hiện trên đĩa trong trường hợp lỗi đĩa. Trong lúc khôi phục, NTFS làm lại mỗi giao tác đã khẳng định tìm trong file sổ ghi. Sau đó NTFS xác định các giao tác nào trong file sổ ghi đó mà không được khẳng định khi lỗi hệ thống và xoá bỏ (undo) các hoạt động con giao tác

đã ghi vào file sổ ghi. Các sửa đổi không trọn vẹn đối với ổ đĩa thì bị ngăn cấm.

NTFS cũng sử dụng dịch vụ file sổ ghi để ghi tất cả các thông tin làm lại (redo) và huỷ bỏ (undo) cho một giao tác. NTFS dùng các thông tin làm lại để lặp lại giao tác đó. Thông tin huỷ bỏ cho phép NTFS huỷ bỏ các giao tác chưa hoàn thành hoặc có một lỗi.

2 Sắp đặt lại cluster (cluster remapping).

Trong trường hợp lỗi do một sector bị hỏng. NTFS thực hiện một kỹ thuật khôi phục được gọi là ***sắp đặt lại cluster***. Khi Windows 2000 tìm thấy một sector hỏng, NTFS sắp đặt lại cluster chứa sector hỏng và phân phối một cluster mới cho dữ liệu. Nếu lỗi xẩy ra trong khi đọc, NTFS trả về một lỗi đọc cho chương trình gọi, và trong trường hợp này dữ liệu bị mất. Nếu lỗi xẩy ra trong lúc ghi, NTFS ghi dữ liệu vào một cluster mới, và dữ liêu không bị mất. NTFS đặt địa chỉ của cluster chứa sector hỏng vào file cluster hỏng ($BadClus). Vì vậy sector hỏng sẽ không được sử dụng lại nữa.

III. Khả năng nén của NTFS.

Hệ thống file NTFS cho phép ta nén các file, thư mục một cách riêng rẽ, và toàn bộ đĩa NTFS. Các file được nén trên đĩa NTFS có thể được đọc, ghi bởi bất kỳ ứng dụng nào trên nền Windows mà không cần phải giải nén trước bằng một chương trình chuyên dụng nào đó (Winzip chẳng hạn).

Việc giải nén được thực hiện một cách tự động khi file nén được đọc. File được nén trở lại khi nó được đóng hoặc được cất. (các file và thư mục được nén có thuộc tính **C** khi xem trong Window explorer).

Trên đĩa NTFS có thể đọc dạng dữ liệu nén. Khi một ứng dụng ví dụ như Microsoft Word, hoặc một lệnh của hệ điều hành ví dụ lệnh **copy** yêu cầu truy cập đến file, thì trình điều khiển lọc nén (the compression filter driver) giải nén file trước khi làm cho file này khả dụng. Ví dụ khi ta copy một file nén từ một thư mục nào đó đến một thư mục nén khác trên đĩa NTFS. Thì file được giải nén khi được đọc, copy, và được nén lại khi cất giữ (saved).

Giải thuật nén thì tương tự như giải thuật đã sử dụng trong ứng dụng DriveSpace3 của Windows 98, chỉ khác là ở đây chức năng bị giới hạn. Cụ thể là nó chỉ nén toàn bộ đĩa (đĩa primary hoặc đĩa logic). Với NTFS cho phép nén toàn bộ một ổ đĩa, một hoặc hơn một thư mục trên đĩa, thậm trí một hoặc một vài file trong một thư mục.

Giải thuật nén trên hệ thống file NTFS được thiết kế để hỗ trợ kích thước cluster lên tới 4KB. Khi kích thước cluster lớn hơn 4KB thì chức năng nén của NTFS không còn khả dụng.

Giải thuật nén:

- Dùng tối thiểu 3 bytes tìm kiếm thay cho 2 bytes mà DoubleSpace đã sử dụng cho phép nén và giải nén nhanh hơn (xấp xỉ hai lần) trong khi chỉ lãng phí 2% khả năng nén cho file văn bản.

- Mỗi luồng dữ liệu chứa thông tin chỉ ra phần nào của luồng được nén. Các vùng đệm nén (compressed buffers) riêng lẻ được nhận biết bởi các lỗ trống (holes) theo sau chúng trong thông tin đã lưu trữ cho luồng đó.

Nếu có một lỗ trống, thì NTFS tự động giải nén vùng đệm phía trước để điền vào lỗ trống.

NTFS cung cấp thời gian thực truy cập đến một file nén, giải nén file khi nó được mở và nén khi nó được đóng.

Khả năng lập trình:

IV. Mã hoá các file và thư mục.

Hệ thống file mã hoá (EFS) cung cấp kỹ thuật mã hóa file để lưu các file được mã hoá trên ổ đĩa NTFS. EFS dữ các file an toàn từ những người xâm nhập trái phép hòng đạt được mục đích nào đó có lợi cho họ.

EFS sử dụng mã hoá khoá cân đối kết hợp với kỹ thuật khoá công khai để bảo vệ các file và đảm bảo rằng chỉ có người chủ của file mới có thể truy cập được vào file. Mọi người sử dụng EFS được phát cho một chứng nhận số cùng với một khoá công khai và một cặp khoá riêng. EFS sử dụng tập khoá này và thông tin nhân được khi người sử dụng nạp vào hệ thống để quyết định quyền truy cập vào file đã được mã hoá dành cho người sử dụng đó.

Người sử dụng làm việc với file và thư mục đã được mã hoá giống như họ làm với bất kỳ file hoặc thư mục nào bởi vì việc mã hoá là trong suốt đối với người sử dụng (là người đã mã hoá file). Hệ thống sẽ tự động giải mã file hoặc thư mục đã mã hoá khi người sử dụng truy cập vào nó. Khi file được cất thì quá trình mã hoá lại được áp dụng lại. Những người truy cập trái phép khi truy cập vào file hoặc thư mục đã được mã hoá thì họ sẽ nhận được thông báo “Access denied” nếu họ cố tình mở, copy, di chuyển, hoặc đổi tên file hoặc thư mục đó.

Để mã hoá hoặc giải mã một file hoặc thư mục, ta đặt thuộc tính mã hoá cho file hoặc thư mục đó giống như ta đặt bất kỳ thuộc tính nào. Nếu ta mã hoá một thư mục, thì tất cả các file và các thư mục con được tạo trong thư mục được mã hoá sẽ tự động được mã hoá.

Khả năng lập trình.

Mã hoá dựa vào một cặp hàm API là **EncryptFile()** và **DecryptFile()**. Hai hàm này lấy thuận lợi của thư viện **CryptoAPI** để làm công việc mã hoá và giải mã. Khi làm việc trong thư mục không được mã hoá ta có thể tao file được mã hoá bằng hai cách: hoăc dụng hàm **CreateFile()** và chỉ ra thuộc tính FILE_ATTRIBUTE_ENCRYPTED hoặc tao/copy file như bình thường và sau đó sử dụng **EncryptFile().**

Để kiểm tra xem một file đưa ra có được mã hoa hay không, ta có thể vẫn sử dụng hàm **GetFileAttributes()** như trường hợp các file nén. Chỉ có một điểm khác là ta kiểm tra hằng số có tên FILE_ATTRIBUTES_ENCRYPTED . Tuy nhiên với Windows 2000 đã đưa ra hàm FileEncryptionStatus() thuân tiên hơn.

Tất cả các file đều có thể được mã hoá, ngoai trừ các file hệ thống và các thư mục gốc hệ thống. Tuy nhiên khả năng mã hoá của NTFS có thể được bật hoặc tắt thông qua lời gọi hàm **EncryptionDisable()**. Có nghĩa là ta có thể lập trình dừng kỹ thuật mã hoá/giải mã tự động cho các thư mục đã được đánh dấu là đã mã hoá.

```
BOOL EncryptionDisable(
    LPCWSTR DirPath,
    BOOL Disable
);
```

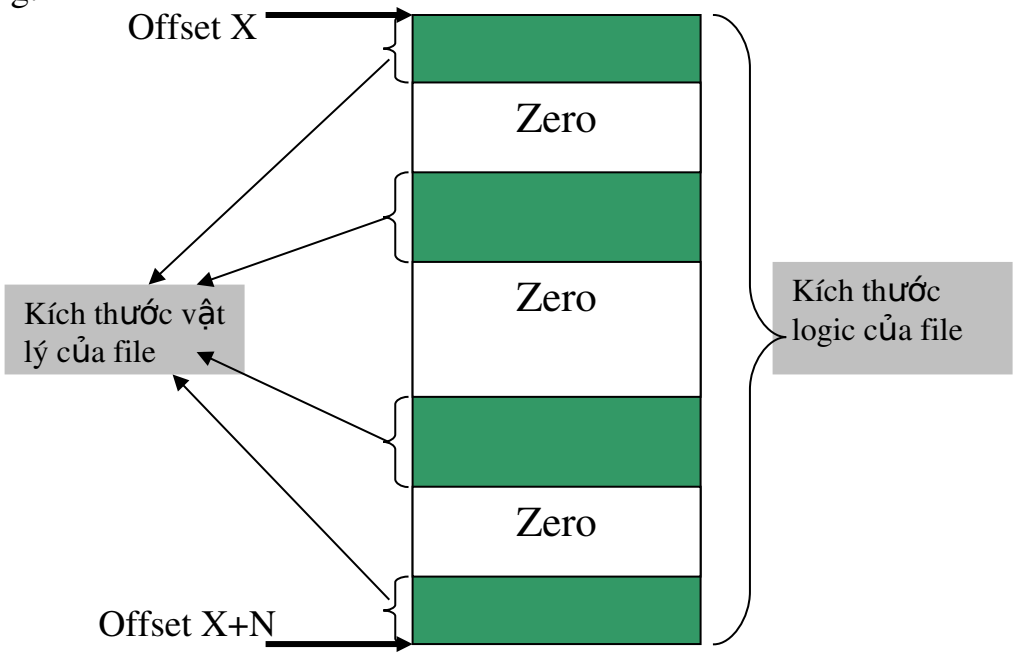
V. File thưa:

là một file (thường là file rất lớn) chứa tập dữ liệu vô nghĩa (zero). File thưa có một thuộc tính làm cho hệ thống con I/O chỉ phân phối đĩa cho những dữ liệu có ích (nonzero), còn các dữ liệu vô ích khác (thường là chiếm đa số) thì không được phân phối đĩa. Khi file thưa được đọc, thì phần dữ liệu đã được phân phối đĩa sẽ được trả về như là nó đã được lưu trữ, còn phần dữ liệu không được phân phối đĩa thì trả lại mặc định là zero.

NTFS hỗ trợ file thưa cho cả file đã nén và file chưa nén. NTFS xử lý các thao tác đọc trên file thưa bằng cách trả lại dữ liệu đã phân phối và dữ liệu thưa, do đó có thể đọc file thưa như là dữ liệu đã phân phối và một vùng dữ liệu không lấy lai được thông tin, mặc dù NTFS trả lại toàn bộ tập dữ liệu này bởi mặc định.

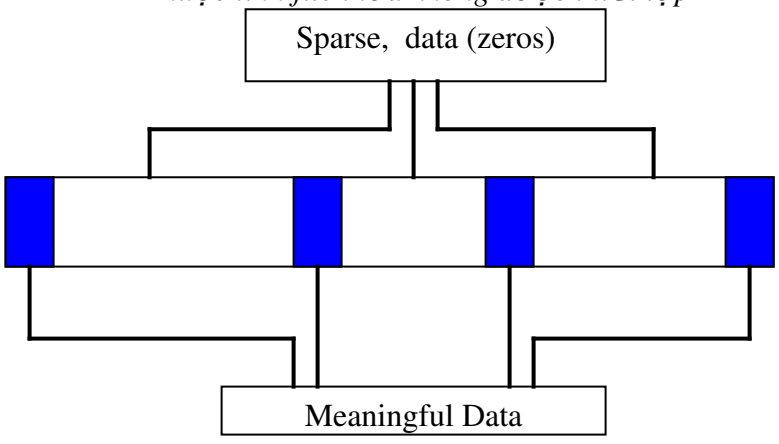
Nếu ta copy hoặc di chuyển một file thưa đến hệ thống file FAT hoặc không phải là hệ thống file NTFS được hỗ trợ bởi Windows 2000, thì file được xây dựng lai theo kích thước xác định ban đầu của nó. Nếu không gian đĩa yêu cầu là không khả dụng, thì thao tác này không thành công.

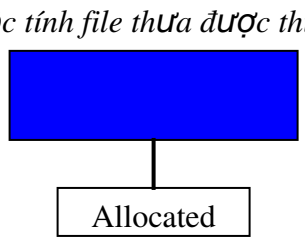
Hình minh hoạ một file thưa.



NTFS đánh dấu (deallocate) các luồng dữ liệu thưa và chỉ giữ lại luồng dữ liệu thực sự như đã phân phối. Khi một chương trình truy cập đến file thưa, hệ thống trả lại phần đữ liệu đã được phân phối như dữ liệu thực sự, và dữ liệu đã đánh dấu trả lại như là zero.

Thuộc tính file thưa không được thiết lập





Hình trên minh hoạ một file được lưu với thuộc tính thưa và không được lưu với thuộc tính thưa.

Các khối mầu xẫm biểu diễn dữ liệu khả dụng, Các khối mầu trắng là các vùng dữ liệu thưa, vùng này được giải phóng bằng *DeviceIoControl()*. Nếu ta yêu cầu hàm *GetFileSize()* để trả về kích thước file, thì ta sẽ nhận được kích thước logic của file. Để nhận được kích thước thực sự (vật lý) của file thưa ta dùng hàm *GetCompressedFileSize().*

Khả năng lập trình.

Để cho phép khả năng thưa trên một file hoặc trên một luồng, ta có thể sử dụng hàm ***DeviceIoControl()***, như được chỉ ra dưới đây:

```
DWORD dwReturnedBytes=0;
DeviceIoControl(hFile,  FSCTL_SET_SPARSE, NULL, 0,
NULL, 0, &dwReturnedBytes,  NULL);
```

và chắc chắn lệnh sau đã xuất hiện ở đầu chương trình.

```
#include winioctl.h
```

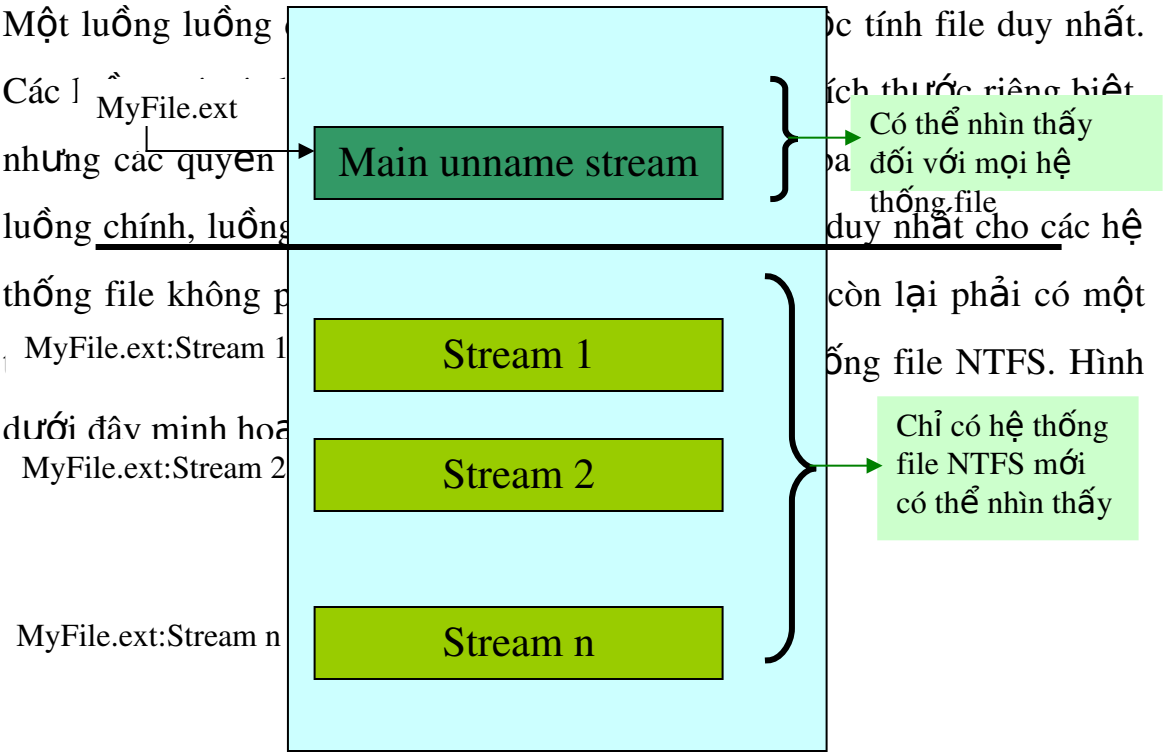
File tiêu đề *winioctl.h* chứa các định nghĩa cho tất cả các cở áp dụng cho ***DeviceIoControl()***. Bao gồm FSCTL_SET_SPARSE chỉ nếu là Win 2000, số hiệu phiên bản của Windows 2000 là 5.0.

Để kiểm tra một file hoặc luồng có phải là thưa hay không ta có thể dùng hàm GetFileAttributes() và kiểm tra cờ FILE_ATTRIBUTE_SPARSE_FILE

```
DWORD dwAttrib = GetFileAttributes(m_szFile);
return (dwAttrib & FILE_ATTRIBUTE_SPARSE_FILE);
```

VI. File đa luồng dữ liệu (Multiple File Streams).

. Dưới hệ thống file NTFS thì mỗi một file có thể có nhiều luồng dữ liêu. Một file đa luồng là việc kết hợp một tập các file đơn luồng được nhúng vào trong cùng một mục nhập (entry) hệ thống. Chúng giống như là một đơn vị nguyên tử và duy nhất, nhưng vẫn bao gồm các đơn vị con độc lập nhau, ta có thể tao, xoá, sửa đổi một cách tách biệt từng đơn vị con này. Mỗi thẻ file (handle) có thể được mở với mỗi luồng dữ liệu, Một luồng luồng
c tính file duy nhất.
Các l
ích thước riêng biệt
nhưng các quyền
a
luồng chính, luồng
duy nhất cho các hệ
thống file không p
còn lại phải có một
ống file NTFS. Hình
dưới đây minh hoạ



Cấu trúc của một file đa luồng

Khi copy một file đa luồng trên đĩa NTFS tới một đĩa không phải là NTFS (hệ thống file FAT chẳng nạy) thì chỉ một luồng dữ liệu chính không có tên được copy, các luồng thêm vào sẽ bị mất. Thậm chí nếu ta copy file đó ngược lại thì cũng không thể khôi phục lại được các luồng đã bị mất đó.

Để xác định một luồng có tên trong một file ta dùng quy ước sau:

tên file + ":" + tên luồng. Ví dụ, để truy cập vào luồng có tên là *StreamName1* trong file có tên là *MyFile.txt* ta sử dụng tên file như sau:

MyFile.txt:StreamName1

Sử dụng quy ước trên với bất kỳ hàm API nào của Win32 để thao tác trên các file. Để truy cập đến nội dung của luồng *StreamName1*, ta chuyển tên luồng này cho hàm ***CreatFile()*** và sau đó sử dụng hàm ***ReadFile()*** và ***WriteFile()*** để đọc và ghi như bình thường. Nếu ta muốn

kiểm tra một luồng thực sự đã tồn tại cùng với một file hay chưa ta có thể dùng mẫu sau:

```
HANDLE hFile = CreateFile(szFileStreamName,
        GENERIC_READ,  0,  NULL, OPEN_EXISTING, 0, 0);
CloseHandle(hFile);
if (hFile == NULL)
MessageBox(hWnd, "Error", NULL, MB_OK);
```

VII. Nhật ký thay đổi (change journal):

Là một đặc trưng mới đối với Windows 2000. Nó theo dõi các thay đổi trên đĩa NTFS, bao gồm việc thêm, xoá, sửa đổi. Nó tồn tại trên đĩa như một file thưa. Mỗi một đĩa NTFS sử dụng một nhật ký thay đổi để theo dõi các thông tin về các file, thư mục, và các đối tượng khác đã thêm, xoá, sửa đổi, NTFS nhập các bản ghi vào nhật ký thay đổi trong các luồng. Mối bản ghi chỉ ra kiểu thay đổi và đối tượng đã thay đổi. Độ lệch (offset) từ chỗ bắt đầu của luồng cho tới một bản ghi khác được gọi là số trình tự cập nhật (the Update Sequence Number-USN) của bản ghi đó. Bản ghi mới được nối vào cuối của luồng. NTFS có thể xoá bản ghi cũ để giữ không gian đĩa. Nếu các bản ghi cần thiết đã bị xoá, thì dịch vụ chú dẫn (indexing service) khôi phục lại bằng cách chú dẫn lại ổ đĩa như là nó làm khi nhật ký thay đổi không tồn tại. Nhật ký thay đổi chỉ ghi những thay đổi thực sự tới file và nguyên nhân (reason) cho sự thay đổi đó. Nó không ghi đầy đủ các thông tin để cho phép đảo ngược các thay đổi. Hơn nữa, nhiều thay đổi với cùng một file có thể cho ra kết quả chỉ trong một cờ nguyên nhân được thêm vào bản ghi hiện tại. Nếu thay đổi

xẩy ra hơn một lần, NTFS không ghi vào một bản ghi mới cho sự thay đổi đó sau bản ghi đầu tiên. Ví dụ các hoạt động ghi khác nhau không trèn vào giữa hoạt động đóng và mở lại file, mà chúng chỉ tồn tại trong một bản ghi duy nhất cùng với nguyên nhân USN_REASON_DATA_OVERWRITE.

Để minh hoạ nhật ký thay đổi làm việc như thế nào, giả sử người sử dụng truy cập một file trong các kiểu sau:

1. Ghi vào một file.
2. Đặt tem thời gian cho file.
3. Ghi vào một file.
4. Cắt một file.
5. Ghi vào một file.
6. Đóng file.

Trong trường hợp này, NTFS giữ lại các hoạt động sau trong nhật ký thay đổi (ký tự "l" thể hiện toán tử **or**):

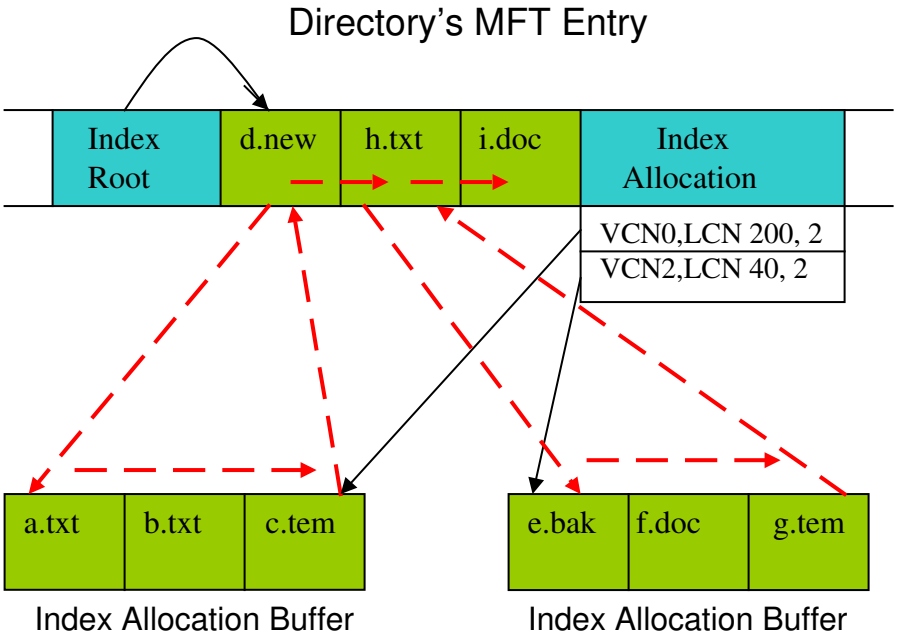
Sự kiện	Hành động của NTFS
Hoạt động ghi đầu tiên	NTFS ghi vào một bản ghi USN mới cùng với việc đặt cờ USN_REASON_DATA_OVERWRITE
Đặt tem thời gian cho file	NTFS ghi vào một bản ghi USN mới cùng với đặt cờ USN_REASON_DATA_OVERWRITE | USN_REASON_BASIC_INFO_CHANGE
Hoạt động ghi thứ hai	NTFS không ghi vào một bản ghi USN mới. Bởi vì cờ nguyên nhân USN_REASON_DATA_OVERWRITE đã được đặt cho bản ghi đang tồn tại, và không có sự thay đổi nào được tao ra cho bản ghi.
Cắt file	NTFS ghi vào một bản ghi USN mới cùng với đặt cờ USN_REASON_DATA_OVERWRITE | USN_REASON_BASIC_INFO_CHANGE | USN_REASON_DATA_TRUNCATION
Hoạt động ghi thứ ba	NTFS không ghi vào một bản ghi USN mới. Bởi vì cờ nguyên nhân USN_REASON_DATA_OVERWRITE đã được đặt cho bản ghi đang tồn tại, và không có sự thay đổi nào được tao ra cho bản ghi.
Hoạt động đóng	NTFS ghi vào một bản ghi USN mới cùng với cùng với các cờ sau được đặt: USN_REASON_DATA_OVERWRITE | USN_REASON_BASIC_INFO_CHANGE | USN_REASON_DATA_TRUNCATION | USN_REASON_CLOSE

Nhật ký thay đổi chứa một loạt các bản ghi từ khi mở file (lần đầu tiên) cho tới khi đóng file (lần cuối cùng). Mỗi bản ghi có một bộ cờ

nguyên nhân mới (a new reason flag set). Bộ cờ này chỉ ra rằng đã có một kiểu thay đổi mới xẩy ra. Trình tự của các bản ghi đưa ra một phần lịch sử của file. Bản ghi cuối cùng được tạo khi file được đóng, và thêm vào đó cờ USN_REASON_CLOSE. Bản ghi này mô tả tóm tắt các thay đổi đối với file, nhưng không giống như các bản ghi phía trước, nó không đưa ra thứ tự của các thay đổi.

VIII. Tạo chỉ mục chung (General Indexing).

Một vài đặc trưng mới của NTFS5 dựa vào đặc trưng cơ sở của NTFS được gọi là *tạo chỉ mụ chung*. Tạo chỉ mục thuộc tính bao gồm việc sắp xếp các phần tử (entry) của một kiểu thuộc tính riêng biệt nào đó, sử dụng các kỹ thuật lưu trữ có hiệu quả cho quá trình tìm kiếm nhanh. Các phiên bản Win 2K trước của NTFS chỉ hỗ trợ tạo chỉ dẫn thuộc tính cho $I30, thuộc tính chỉ dẫn chỉ lưu các phần tử thư mục. Quá trình tạo chỉ dẫn thuộc tính sắp xếp các phần tử thư mục bởi tên và lưu các phần tử này theo cấu trúc cây nhị phân. Hình bên dưới minh hoạ bản ghi (phần tử) MFT của một thư mục chứa 9 phần tử lưu trong 3 nút, mỗi nút ứng với 3 phần tử. Thuộc tính Index Root chứa gốc của cây nhị phân. Bởi vì 9 phần tử không thể điền hết vào một phần tử MFT của thư mục. NTFS phải lưu một vài phần tử vào nơi khác. Do đó NTFS phân phối hai vùng đệm phân phối chỉ mục (two Index allocation buffers) để lưu trữ hai phần tử (chỉ mục gốc và các vùng đệm phân phối chỉ mục điển hình có thể chứa nhiều hơn ba file trong một phần tử, phụ thuộc vào chiều dài tên file). Như đã đề cập một phần tử MFT có kích thước là 1KB, và các vùng đệm phân phối chỉ mục là 4KB.



Mũi tên nét đứt chỉ ra cách NTFS lưu trữ các phần tử. Nó là cách lưu trữ theo thứ tự từ điển. Nếu ta chạy một chương trình để mở file e.dak trong thư mục được minh hoạ trong hình vẽ, NTFS sẽ đọc thuộc tính *Index root*, Index root chứa các phần tử *d.new*, *h.txt*, và *i.doc*, và so sánh xâu "*e.dak*" với tên trong phần tử đầu tiên là *d.new*, NTFS kết luận rằng *e.dak* theo thứ tự từ điển là lớn hơn *d.new* và nó dịch lên phần tử kế tiếp là *h.txt*. Sau khi thực hiện quá trình so sánh như trên, NTFS phát hiện ra rằng *e.dak* theo thứ tự từ điển là nhỏ hơn *h.txt*. Sau đó NTFS sẽ tìm bên trong phần tử thư mục của *h.txt* để lấy số hiệu cluster ảo (VCN) của vùng đệm chỉ mục. Vùng này chứa các phần tử thư mục là nhỏ hơn *h.txt* theo thứ tự từ điển (nhưng lớn hơn *d.new*). VCN tương ứng với thứ tự một cluster trong phạm vi một file hoặc một thư mục. NTFS sử dụng

thông tin sắp xếp để dịch một VCN thành số hiệu cluster logic (LCV), LCN là số hiệu cluster tương đối với vị trí bắt đầu của đĩa. Nếu phần tử thư mục cho *h.txt* không lưu một VCN nào cho vùng đệm chỉ mục, thì ngay lập tức NTFS biết rằng thư mục *h.txt* không chưa *e.dak* và sẽ chỉ ra rằng việc tìm kiếm bị thất bại.

Sau khi có số hiệu VCN của cluster suất phát của vùng đệm chỉ mục, NTFS đọc vùng đệm phân phối chỉ mục và quét toàn bộ vùng đệm này để tìm ra một tương ứng. Phần tử đầu tiên của vùng đệm chỉ mục là phần tử mà NTFS đang tìm, do đó NTFS đọc số hiệu phần tử MFT của *e.bak*. Các phần tử thư mục cũng chứa các thông tin khác như tem thời gian của file (ngày, giờ tạo, ngày giờ sửa lần cuối cùng,...), kích thước, và các thuộc tính khác.

Các phần tử thư mục được sắp xếp theo thứ tự từ điển, điều này giải thích tại sao các file NTFS luôn luôn in ra theo thứ tự từ điển. Ngược lại hệ thống file FAT do không sắp xếp các thư mục theo thứ tự từ điển nên các file được in ra không được sắp xếp. Hơn nữa bởi vì NTFS lưu trữ như là cây nhị phân, do đó việc tìm kiếm một file nào đó trong một thư mục lớn rất hiệu quả, NTFS chỉ cần quét một phần nhỏ của thư mục. Ngược lại FAT phải quét toàn bộ thư mục đó.

Trong khi NTFS chỉ thực hiện tạo chỉ mục cho tên file thì NTFS5 thực hiện tạo chỉ mục chung. Đặc trưng này cho phép NTFS5 lưu trữ dữ liệu bất kỳ chỉ mục và sắp xếp các phần tử dữ liệu này bằng một vài tên khác nhau. NTFS sử dụng tạo chỉ mục chung để quản lý các mô tả tinh bảo mật, thông tin hạn ngạch đĩa, các điểm *reparse*, và các định danh đối tượng file.

IX. Các điểm reparse.

Trên hệ thống file NTFS thì một file hoặc thư mục có thể chứa điểm reparse. Điểm reparse là một bộ dữ liệu của người dùng định nghĩa. Định dạng của dữ liệu này được nhận biết bởi chương trình ứng dụng (là chương trình lưu trữ dữ liệu), và *bộ lọc hệ thống file* (the file system filter), là chương trình để dịch dữ liệu và xử lý file. Khi một chương trình ứng dụng đặt một điểm reparse vào một file hoặc thư mục, thì chương trình đó lưu trữ dữ liệu này, cộng với một thẻ reparse, thẻ này sẽ xác định dữ liệu điểm reparse đang lưu trữ. Khi hệ thống file mở một file có chứa điểm reparse, nó sẽ tìm *bộ lọc hệ thống file* tương ứng với dạng dữ liệu đã xác định bởi cờ reparse. Nếu một bộ lọc hệ thống file như vậy được tìm thấy, thì bộ lọc xử lý file như được chỉ dẫn bởi dữ liệu reparse. Nếu không tìm thấy bộ lọc như vậy, thì thao tác mở file là thất bại.

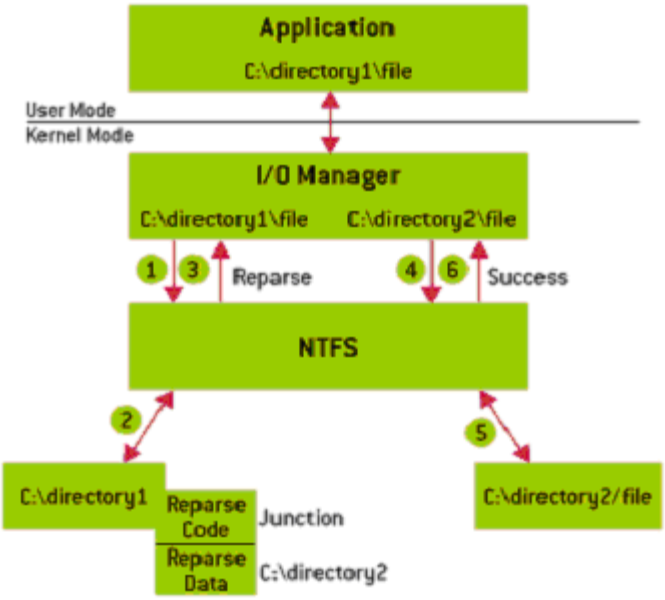
Win2k cung cấp một vài kiểu chức năng như các điểm gắn, các đầu nối NTFS, và quản lý lưu trữ phân cấp (HSM). Em xin trình bầy cách mà mỗi chức năng này làm việc, sau đó sẽ đi sâu vào cách thực thi của các điểm reparse.

Để có thể truy cập, tất cả các đĩa NTFS phải có tên ổ đĩa. Các điểm gắn cho phép ta nối một ổ đĩa vào một thư mục điểm gắn trên đĩa NTFS5 cha mà không cần gán một tên ổ đĩa cho ổ đĩa con đó. Khả năng này cho phép ta hợp nhất nhiều đĩa dưới một tên ổ đĩa. Ví dụ, nếu ta gắn một đĩa có chứa thư mục \articles tới một điểm gắn có tên C:\documents, thì ta có thể sử dụng đường dẫn C:\articles\documents để truy cập các file trong thư mục \documents. Điểm gắn là một điểm reparse mà dữ liệu của

nó có tên nội bộ của một đĩa. Tên nội bộ này có dạng \??\Tên đĩa XX-XX-XX-XX , với các X là số hiệu ID toàn cục duy nhất (GUID) mà Win2K đã gán cho đĩa.

Khi ta mở file C:\articles\documents\column.doc, thì NTFS bắt gặp một điểm gắn liên kết với thư mục \documents. NTFS đọc dữ liệu reparse của điểm gắn (tên ổ đĩa) và trả về trạng thái reparse cho trình quản lý đối tượng (the Object Manager). Trình quản lý I/O dịch trạng thái reparse này, kiểm tra dữ liệu reparse, và quyết định rằng NTFS rằng NTFS đã bắt gặp một điểm gắn. Trình quản lý I/O sửa tên đường dẫn đang tìm và định hướng trình quản lý đối tượng phát lại quá trình tìm kiếm sử dụng đường dẫn đã sửa đổi là \??\tên đĩa *GUID* \documents\column.doc, và việc tìm kiếm sẽ tiến hành trên ổ đĩa được gắn với đường dẫn \documents\column.doc.

Các mối nối NTFS tương tự như các điểm gắn, trình quản lý I/O và trình quản lý đối tượng thực thi các mối nối thư là chúng thực thi các điểm gắn. Tuy nhiên các mối nối thường tham chiếu đến các thư mục hơn là tham chiếu đến các ổ đĩa như điểm gắn. Nếu ta tạo mối nối C:\artiles\documents tham chiếu đến d:\documents, thì ta có thể truy cập đến các file được lưu trữ trong D:\documents bằng đường dẫn C:\articles\documents. Điểm reparse của mối nối lưu trữ thông tin đường dẫn được đổi hướng, và như là điểm gắn, trình quản lý I/O sửa tên đường dẫn và phát lại quá trình tìm kiếm khi NTFS bắt gặp một mối nối. Hình dưới đây minh hoạ các mối nối làm việc như thế nào.



Khi một ứng dụng mở C:\directory1\file, thì NTFS bắt gặp một điểm reparse trong C:\directory1, điểm reparse này trỏ vào C:\directory2. Trình quản lý I/O thay đổi tên đường dẫn thành C:\directory2\file, và cuối cùng chương trình ứng dụng mở C:\directory2\file.

Win2K không có các Tool cho để tạo các mối nối, và Microsoft chính thức không hỗ trợ các Tool này bởi vì một vài ứng dụng không thực hiện đúng đắn khi chúng sử dụng đường dẫn có chứa mối nối. Tuy nhiên ta có thể dùng công cụ Linkd của Microsoft Windows 2000 Recource Kit hoặc công cụ mối nối miễn phí *(http://www.sysinternals.com/misc.htm)* để tạo và liệt kê các mối nối.

Khi một file hoặc thư mục có một điểm reparse kết hợp cùng với nó, NTFS tạo một thuộc tính có tên là $Reparse cho điểm reparse này. Thuộc tính này lưu trữ mã và dữ liệu reparse. Do vậy NTFS có thể dễ dàng xác định được tất cả các điểm reparse trên đĩa, File siêu dữ liệu (metadata file) có tên \$Extend\$Reparse lưu trữ toàn bộ các phần tử, là các phần tử liên kết các số hiệu phần tử file và thư mục của MFT có

chứa điểm reparse với các mã điểm reparse tương ứng. NTFS sắp xếp các phần tử này bằng số hiệu phần tử của MFT trong chỉ mục $R.

X. ***Giới hạn dung lượng cho mỗi người*** *(gọi tắt là hạn ngạch đĩa-Disk quotas).*

Là dung lượng đĩa khả dụng tối đa dành cho mỗi người. Windows 2000 hỗ trợ các hạn ngạch đĩa cho ổ đĩa được định dạng bằng hệ thống file NTFS. Ta có thể sử dụng hạn ngạch đĩa để theo dõi và giới hạn không gian đĩa sử dụng. Hạn ngạch đĩa được theo dõi trên cơ sở mỗi người sử dụng trên một ổ đĩa. Người quản lý hệ thống có thể sử dụng nhãn ***Quota*** của hộp hội thoại Properties để thực hiện các tác vụ sau:

- Cho phép hoặc không cho phép hạn ngạch trên một ổ đĩa.
- Cấm mọi người sử dụng cất dữ liêu (mới) khi họ đã vượt quá hạn nghạch.
- Đặt mức cảnh báo hạn ngạch đĩa mặc định và giới hạn hạn nghạch đĩa cho mọi người mới sử dụng đĩa.

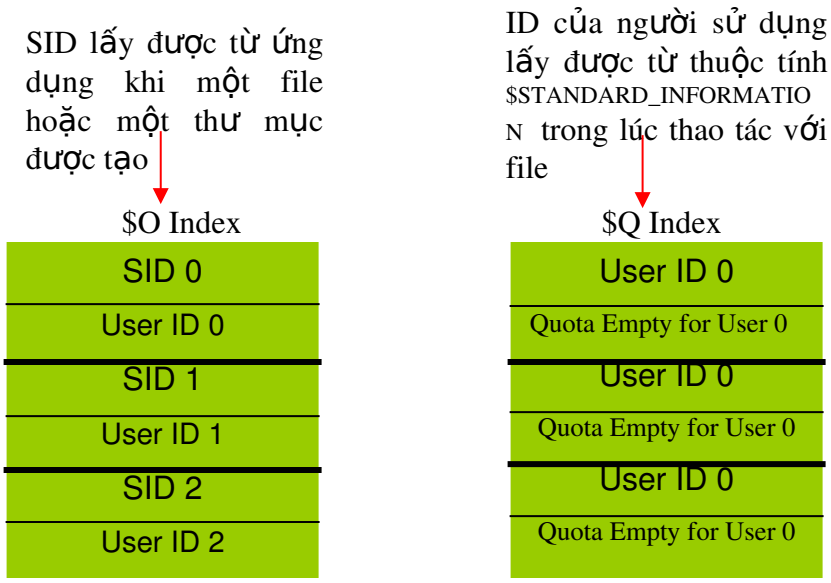
Xem thông tin hạn ngạch đĩa cho mỗi người sử dụng (sử dụng Quota emtries).

Hạn ngạch đĩa theo dõi và kiểm soát việc sử dụng không gian đĩa cho các ổ đĩa. Người quản lý hệ thống có thể cấu hịnh Windows 2000 để thực hiện các tác vụ sau:

- Cấm sử dụng quá không gian đĩa và nạp một sự kiện khi người sử dụng vượt quá không gian đĩa đã chỉ ra.
- Nạp một sự kiện khi người sử dụng vượt quá mức cảnh bào đã chỉ ra.

Khi cho phép hạn ngạch đĩa, ta có thể đặt giới hạn và mức cảnh báo han ngạch đĩa. Giới hạn đĩa chỉ ra dung lượng đĩa được phân cho một người sử dụng, còn mức cảnh báo chỉ ra khi một người sử dụng gần tới giới hạn.

NTFS lưu các thông tin hạn ngạch trong file \$Extend\$Quota, file này gồm các chỉ mục $O và $Q. Hình dưới đây chỉ ra cách tổ chức của các chỉ mục này. Giống như khi NTFS gán cho mỗi mô tả bảo mật một ID nội bộ duy nhất, NTFS cũng gán cho mỗi người sử dụng một ID duy nhất. Khi người quản lý hệ thống chỉ định thông tin hạn ngạch cho một người sử dụng, thì NTFS phân phối một ID cho người sử dụng đó (ID tương ứng với SID của người sử dụng). Trong chỉ mục $O, NTFS tao ra một phần tử để đặt một SID tương ứng với một ID người sử dụng và NTFS sắp sếp chỉ mục này theo ID người sử dụng. Trong chỉ mục $Q, NTFS tạo một phần tử kiểm soát hạn ngạch. Một phần tử kiểm soát hạn ngạch chứa giá trị giới hạn hạn ngạch của người sử dụng ngay khi người sử dụng tiêu dùng không dung lượng đĩa của họ.



Khi một ứng dụng tạo một file hoặc thư mục, NTFS lấy SID của người sử dụng ứng dụng và tra cứu ID người sử dụng tương ứng trong chỉ mục $O. NTFS ghi ID người sử dụng trong thuộc tính $STANDARD_INFORMATION của file hoặc thư mục mới, thuộc tính này tính toàn bộ không gian đĩa đã phân phối cho file hoặc thư mục dựa vào hạn ngạch của người sử dụng đó. Sau đó NTFS tìm phần tử giới hạn trong chỉ mục $Q và quyết định xem việc phân phối mới này có làm cho người sử dụng vượt quá ngưỡng giới hạn hoặc cảnh báo của họ hay không?. Nếu vượt quá thì NTFS hoặc nạp một sự kiện thích hợp hoặc không cho phép người sử dụng tạo file hoặc thư mục đó. Khi một file hoặc thay đổi kích thước, NTFS cập nhật phần tử kiểm soát hạn ngạch tương ứng với ID người sử dụng đã lưu trữ trong thuộc tính $STANDARD_INFORMATION. NTFS dùng *chỉ mục chung* để tạo mối tương quan giữa các ID người sử dụng với các SID (accounts), và với một ID

người sử dụng đã đưa ra thì cho phép tra cứu thông tin kiểm soát hạn ngạch của một người sử dụng một cách hiệu quả.