

Introduzione: il *file system RAW*

Non è purtroppo così raro trovarsi a infilare nella porta USB il proprio disco esterno preferito (rigido o *flash*) e notare una minacciosa finestra di Windows che chiede di formattare l'unità in quanto risulta non formattata.

Talvolta si è fortunati, e basta declinare gentilmente l'offerta per ritrovarsi a lavorare su un disco ancora perfettamente funzionante.

Qualche altra volta, invece, la doverosa esecuzione dell'utilità di riparazione dei dischi, magari dopo un opportuno "sfila e reinfila" della periferica o, al limite, un riavvio del sistema operativo (misure spesso sufficienti a porre fine a simili anomalie), sbatte in faccia una dura realtà: lo strumento CHKDSK di Windows non è capace di riparare le unità con file system "RAW" (=grezzo, non formattato).

In effetti, ciò che Windows lamenta in questi casi è proprio l'assenza di un file system.

Se questi termini sembrano per ora oscuri al lettore, una cosa è ben chiara: tutti i file che ci si aspettava di trovare sul disco danneggiato non risultano più accessibili.

Si può pensare allora di ricorrere a utilità non Microsoft: 7-Zip File Manager, testdisk, O&O Disk Recovery, e tanti altri strumenti, sia *open source* che commerciali, possono fare il miracolo.

Quando non ne sono capaci, tutto è, allora, perduto?

Al contrario, un esame più accurato, condotto con strumenti diversi, rivelerà quasi certamente che tutto è ancora lì, perfettamente intatto, che aspetta di essere recuperato da noi.

Lo scopo di questo articolo è di capire come.

Premessa: l'archiviazione dei dati

Prima di affrontare nel dettaglio le tecniche di recupero manuale o semi-automatico dei file system danneggiati, è opportuno premettere delle nozioni di base sul funzionamento di dischi e file system.

Un disco è un c.d. dispositivo a blocchi, ossia il contenitore di una sequenza di blocchi di eguale dimensione (**settori**) accessibili in ordine lineare o casuale.

E' interessante notare che tale sequenza può essere tranquillamente riversata in un file, così da eseguire qualsiasi operazione su una copia dei dati anziché sul disco originale.

Nelle unità di tipo MBR (Master Boot Record), le sole largamente diffuse fino a poco fa, il settore aveva una dimensione standard di 512 byte (mezzo Kilobyte).

Naturalmente, un proficuo sfruttamento dei settori richiede una qualche forma di organizzazione: questa si chiama file system, e ci consente di isolare e catalogare i dati nei file e cartelle (*folder*, *directory*, *drawer* o indirizzari) che tutti conosciamo.

Spesso il file system non è applicato direttamente all'intero disco, poiché questo viene preventivamente diviso in partizioni: la posizione e la dimensione di eventuali partizioni aggiuntive è segnalata nel MBR, ossia il primissimo settore del disco.

Un disco rigido esterno ha, di regola, almeno una partizione; le chiavette USB, invece, non sono di regola né divise né divisibili in partizioni da Windows (una seconda partizione creata con Linux non viene automaticamente riconosciuta da Windows).

In Windows, i dischi di uso comune per la lettura e scrittura dei dati possono impiegare uno fra questi due diversi file system: **FAT** (File Allocation Table), derivato dai tempi di MS-DOS, i remoti Anni '80; oppure il più moderno **NTFS** (New Technology File System), frutto dell'evoluzione iniziata intorno al 1995 con l'avvento del sistema operativo Windows NT, che sta alla base dei più recenti sistemi operativi (Windows XP, Server 2003, Vista, Server 2008, Seven, 8).

Se l'unità atomica in cui sono ripartiti i dati sul disco al grezzo è il settore (tipicamente di 512 byte, lo si ripete), l'unità minima di allocazione dei file con i predetti file system è il **cluster**, un gruppo di settori consecutivi (1, 2, 4, 8, 16, 32, 64, 128 settori, ossia *cluster* da ½, 1, 2, 4, 8, 16, 32 o 64 Kilobyte).

La dimensione del cluster è determinata, dall'utente, a piacere, o dal sistema, in base alla dimensione del file system, al momento della formattazione.

Il contenuto di un file occupa di regola un numero di *cluster* interi, a prescindere dalla sua effettiva dimensione (quindi, una porzione dell'ultimo *cluster* risulterà il più delle volte inutilizzata, con uno spreco¹ di spazio tanto maggiore quanto più grande è il *cluster*).

Gli altri dati di un file (nome, dimensione, date e ore, permessi, proprietario, posizione sul disco...) sono conservati in apposite strutture del file system, che risiedono in altri settori del disco, normalmente non accessibili per l'utente comune.

Il settore zero

Poiché in informatica la numerazione delle sequenze comincia spesso e volentieri da zero, si allude al primo settore di un disco, di una partizione o di un file system.

Il settore zero contiene informazioni indispensabili per il sistema operativo, quali il settore iniziale della partizione e il numero di settori componenti, oppure la dimensione del *cluster* e la posizione di certe strutture portanti del file system.

Quando riceviamo i messaggi di errore citati nell'introduzione, quasi certamente ciò accade perché uno di questi settori è danneggiato: il sistema, quindi, non è più in grado di capire cosa si trovi dentro al disco o al file system, e nemmeno di indovinarlo da solo.

Molte volte il danno in questione non è fisico: è solo sparito il contenuto del settore zero, ed esso può essere semplicemente riscritto, magari applicando una copia di riserva (le ultime versioni di NTFS scrivono una copia del settore zero nell'ultimo settore del volume, e possono "ripararsi" da sole; ma potremmo creare noi stessi un backup manuale) o, nella peggiore delle ipotesi, rigenerandolo manualmente.

¹ Tale spazio ridondante potrebbe anche essere usato per occultare informazioni.

Comunque sia, è altamente probabile che tutte le altre strutture del file system siano ancora integre e al loro posto: mentre il sistema operativo o le stesse utilità di riparazione dei dischi possono avere problemi a riconoscerle, noi siamo osservatori infinitamente più acuti dei nostri computer.

Ovviamente, prima di riconoscere, bisogna conoscere; e, inoltre, disporre degli strumenti adeguati per esaminare i dischi.

Mentre gli strumenti non mancano su Internet (si consiglia di scaricare il linguaggio di programmazione Python 2.7 e un editore esadecimale capace di accedere ai dischi, come wxHexEditor o HxD), questa guida si propone di colmare le conoscenze sui file system FAT e NTFS.

Numeri esadecimali e CPU

Quanto si andrà a esporre presuppone la conoscenza di come i numeri interi sono registrati e riconosciuti dalle CPU sulle quali gira Windows (Intel x86 e ia64 e compatibili) e sulla notazione esadecimale dei numeri.

Notoriamente, il computer registra qualsiasi cosa come sequenze di **bit** (BInary digiT, cifra binaria, 0 o 1, carico o scarico, acceso o spento).

La rappresentazione di un numero richiede, evidentemente, l'impiego di più bit: ad esempio, con 4 bit possiamo codificare 2^4 diverse combinazioni, ossia 16 numeri).

Ciò ha indotto a sviluppare una **notazione** numerica più compatta, detta **esadecimale** poiché impiega sei simboli (lettere da A a F) oltre ai consueti dieci (numeri da 0 a 9).

La seguente tabella illustra i primi 256 numeri esadecimali, da 0 a 255.

Notiamo che ogni decina esadecimale corrisponde in decimale, a un multiplo di 16; mentre i primi dieci simboli (0-9) sono comuni con il sistema decimale, i numeri decimali da 10 a 15 sono rappresentati in esadecimale con le lettere da A a F.

	16x0	16x1	16x2	16x3	16x4	16x5	16x6	16x7	16x8	16x9	16x10	16x11	16x12	16x13	16x14	16x15
	0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
0	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
+1	01	11	21	31	41	51	61	71	81	91	A1	B1	C1	D1	E1	F1
+2	02	12	22	32	42	52	62	72	82	92	A2	B2	C2	D2	E2	F2
+3	03	13	23	33	43	53	63	73	83	93	A3	B3	C3	D3	E3	F3
+4	04	14	24	34	44	54	64	74	84	94	A4	B4	C4	D4	E4	F4
+5	05	15	25	35	45	55	65	75	85	95	A5	B5	C5	D5	E5	F5
+6	06	16	26	36	46	56	66	76	86	96	A6	B6	C6	D6	E6	F6
+7	07	17	27	37	47	57	67	77	87	97	A7	B7	C7	D7	E7	F7
+8	08	18	28	38	48	58	68	78	88	98	A8	B8	C8	D8	E8	F8
+9	09	19	29	39	49	59	69	79	89	99	A9	B9	C9	D9	E9	F9
+10	0A	1A	2A	3A	4A	5A	6A	7A	8A	9A	AA	BA	CA	DA	EA	FA
+11	0B	1B	2B	3B	4B	5B	6B	7B	8B	9B	AB	BB	CB	DB	EB	FB
+12	0C	1C	2C	3C	4C	5C	6C	7C	8C	9C	AC	BC	CC	DC	EC	FC
+13	0D	1D	2D	3D	4D	5D	6D	7D	8D	9D	AD	BD	CD	DD	ED	FD
+14	0E	1E	2E	3E	4E	5E	6E	7E	8E	9E	AE	BE	CE	DE	EE	FE
+15	0F	1F	2F	3F	4F	5F	6F	7F	8F	9F	AF	BF	CF	DF	EF	FF

In letteratura, un numero esadecimale appare usualmente con uno *0x* prefisso o con un *h* suffisso (ad esempio: 0xFF o FFh).

Mentre i primissimi calcolatori usavano effettivamente un gruppo di 4 bit (c.d. *nibble*) per rappresentare una cifra decimale da 0 a 9, le CPU attuali adottano come unità minima la sequenza di 8 bit conosciuta come **byte**.

Un singolo byte può rappresentare al massimo 2^8 (=256) numeri interi: poco, per i nostri scopi.

Per rappresentare numeri più grandi, occorre dunque usare più byte: 2 per una WORD di 16 bit (2^{16} = 65536 numeri), 4 per una DWORD (double word) di 32 bit (2^{32} = 4.294.967.296) od 8 per una QWORD (quad word) di 64 bit (2^{64} = 18.446.744.073.709.551.616).

Naturalmente, possiamo rappresentare anche gli interi negativi, scegliendo di utilizzare un bit per il **segno**: una medesima sequenza di bit può rappresentare dunque sia un numero senza segno, sia un diverso numero con segno.

La seguente tabella illustra le possibili rappresentazioni, a seconda che il numero sia senza (*unsigned*) o con (*signed*) segno.

	<i>Unsigned</i>	Intervallo	<i>Signed</i>	Max
8 bit	da 0 a 2^7-1	0-7F	da 0 a 2^7-1	1 B
	da 2^7 a 2^8-1	80-FF	da -2^7 a -1	
16 bit	da 0 a $2^{15}-1$	0-7FFF	da 0 a $2^{15}-1$	64 KiB
	da 2^{15} a $2^{16}-1$	8000-FFFF	da -2^{15} a -1	
32 bit	da 0 a $2^{31}-1$	0-7FFFFFFF	da 0 a $2^{31}-1$	4 GiB
	da 2^{31} a $2^{32}-1$	80000000-FFFFFFFF	da -2^{31} a -1	
64 bit	da 0 a $2^{63}-1$	0-7FFFFFFFFFFFFFFF	da 0 a $2^{63}-1$	16 EiB
	da 2^{63} a $2^{64}-1$	8000000000000000-FFFFFFFFFFFFFFFF	da -2^{63} a -1	

Notiamo come un numero a 32 bit permette di accedere a un intervallo di 4 Gigabyte (quantità inferiore a quella di molti file e dischi di oggi), mentre uno di 64 addirittura a 16 Esabyte.

Va ora sottolineato che le CPU che ci interessano registrano le sequenze numeriche composte da più byte cominciando da quello contenente le unità e proseguendo con quelli che esprimono gli ordini di grandezza maggiori (c.d. **formato Little Endian**): in altre parole, con ordinamento inverso a quello in cui noi lo scriviamo.

Ad esempio, il numero 0x7FFF apparirà in memoria come FF 7F (WORD) o FF 7F 00 00 (DWORD) o FF 7F 00 00 00 00 00 00 (QWORD).

Un'ultima precisazione, per comprendere meglio le quantità informatiche descritte nella presente trattazione.

Prefisso	Lettura	Valore	Prefisso	Valore
Kb	Kilo-	10^3	KiB	2^{10}
Mb	Mega-	10^6	MiB	2^{20}
Gb	Giga-	10^9	GiB	2^{30}
Tb	Tera-	10^{12}	TiB	2^{40}

Pb	Peta-	10^{15}	PiB	2^{50}
Eb	Esa-	10^{18}	EiB	2^{60}
Zb	Zeta-	10^{21}	ZiB	2^{70}
Yb	Yota-	10^{24}	YiB	2^{80}

In passato, si erano sempre usati i prefissi della prima colonna (la “b” di byte può essere indifferentemente maiuscola) per esprimere le potenze di due nell’ultima, in coerenza con la base binaria che domina tutti i numeri trattati in informatica.

I produttori di memorie e di dischi, tuttavia, hanno sempre voluto intendere i prefissi in parola nel loro abituale valore decimale, non senza generare, a loro vantaggio, equivoci nei consumatori: comprare un disco rigido USB da 400 GB significava, per l’utente, credere di comprare 400×2^{30} byte e ritrovarsi in realtà con 400×10^9 soltanto (circa 372,53 GiB).

Di recente è stata dunque standardizzata la notazione della terza colonna, con una “i” nel mezzo, che esprime proprio le potenze di 2 delle quali tratteremo nel prosieguo.

Un esempio di Master Boot Record

Forti delle nozioni di cui sopra, possiamo finalmente dare uno sguardo al settore zero (MBR) di un disco rigido USB da 400 GB (la trascrizione è stata generata da wxHexEditor):

```

0000000000h  33 C0 8E D0 BC 00 7C FB 50 07 50 1F FC BE 1B 7C  3.....|.P.P....|
0000000010h  BF 1B 06 50 57 B9 E5 01 F3 A4 CB BE BE 07 B1 04  ...PW.....
0000000020h  38 2C 7C 09 75 15 83 C6 10 E2 F5 CD 18 8B 14 8B  8,|.u.....
0000000030h  EE 83 C6 10 49 74 16 38 2C 74 F6 BE 10 07 4E AC  ....It.8,t....N.
0000000040h  3C 00 74 FA BB 07 00 B4 0E CD 10 EB F2 89 46 25  <.t.....F%
0000000050h  96 8A 46 04 B4 06 3C 0E 74 11 B4 0B 3C 0C 74 05  ..F...<.t...<.t.
0000000060h  3A C4 75 2B 40 C6 46 25 06 75 24 BB AA 55 50 B4  :.u+@.F%.u$.UP.
0000000070h  41 CD 13 58 72 16 81 FB 55 AA 75 10 F6 C1 01 74  A..Xr...U.u....t
0000000080h  0B 8A E0 88 56 24 C7 06 A1 06 EB 1E 88 66 04 BF  ....V$......f..
0000000090h  0A 00 B8 01 02 8B DC 33 C9 83 FF 05 7F 03 8B 4E  ....3.....N
00000000A0h  25 03 4E 02 CD 13 72 29 BE 46 07 81 3E FE 7D 55  %.N...r).F...>.)U
00000000B0h  AA 74 5A 83 EF 05 7F DA 85 F6 75 83 BE 27 07 EB  .tZ.....u..'.
00000000C0h  8A 98 91 52 99 03 46 08 13 56 0A E8 12 00 5A EB  ...R..F..V....Z.
00000000D0h  D5 4F 74 E4 33 C0 CD 13 EB B8 00 00 81 07 33 10  .Ot.3.....3.
00000000E0h  56 33 F6 56 56 52 50 06 53 51 BE 10 00 56 8B F4  V3.VVRP.SQ...V..
00000000F0h  50 52 B8 00 42 8A 56 24 CD 13 5A 58 8D 64 10 72  PR..B.V$.ZX.d.r
0000000100h  0A 40 75 01 42 80 C7 02 E2 F7 F8 5E C3 EB 74 49  .@u.B.....^..tI
0000000110h  6E 76 61 6C 69 64 20 70 61 72 74 69 74 69 6F 6E  nvalid partition
0000000120h  20 74 61 62 6C 65 00 45 72 72 6F 72 20 6C 6F 61  table.Error loa
0000000130h  64 69 6E 67 20 6F 70 65 72 61 74 69 6E 67 20 73  ding operating s
0000000140h  79 73 74 65 6D 00 4D 69 73 73 69 6E 67 20 6F 70  ystem.Missing op
0000000150h  65 72 61 74 69 6E 67 20 73 79 73 74 65 6D 00 00  erating system..
0000000160h  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....
0000000170h  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....
0000000180h  00 00 00 8B FC 1E 57 8B F5 CB 00 00 00 00 00 00 00  ....W.....
0000000190h  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....
00000001A0h  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....
00000001B0h  00 00 00 00 00 00 00 00 2C 11 3A E9 00 00 00 20  ....,.....
00000001C0h  21 00 07 FE FF FF 00 08 00 00 00 80 93 2E 00 00  !.....
00000001D0h  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....
00000001E0h  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....
00000001F0h  00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA  ....U.

```

Notiamo che un MBR valido termina sempre con una WORD 0xAA55 (in rosso).

La gran parte del settore contiene il c.d. **codice di avvio**, ossia un piccolo programma che, avviato dal BIOS, legge il settore zero della prima partizione e, tramite il programma ivi contenuto, inizia il caricamento di un sistema operativo; o stampa un messaggio di errore, che forse sarà capitato di leggere, se non trova un settore di avvio valido nella partizione.

E' questo codice che rende il disco "avviabile", come suol dirsi.

In realtà, ciò che è veramente necessario per accedere al contenuto del disco sono i gruppi di 16 byte costituenti la **tabella delle partizioni primarie**, a partire dalla posizione 0x1BE (in azzurro).

Di seguito, il significato dei valori più interessanti (evidenziati in giallo):

Posizione	Tipo	Significato
00	BYTE	Avviabile? (0x80=sì, 0x00=no, altro=invalido)
04	BYTE	Tipo di partizione (0x07=NTFS)
08	DWORD	N° del primo settore della partizione
0C	DWORD	Totale settori della partizione

La partizione (e, quindi, il file system NTFS che ospita), inizia al byte 100000h (800h x 512) e si estende asseritamente per 0x2E9380 settori.

Da notare che quest'ultimo valore (probabilmente, impostato da testdisk dopo una riparazione) è sicuramente errato, dato che il disco ha 400 GB: non di meno, il sistema ha realmente bisogno di conoscere solo la posizione iniziale della partizione, che difatti è corretta.

Un esempio di settore di avvio FAT

Una chiavetta USB, normalmente non divisa né divisibile in partizioni sotto Windows, comincia con il settore zero del file system (c.d. **boot sector**, settore di avvio).

Dando un'occhiata a un settore di avvio FAT32, possiamo subito capire perché esso è essenziale al sistema operativo:

```
00000000h EB 58 90 4D 53 44 4F 53 35 2E 30 00 02 08 1E 04 .X.MSDOS5.0....
00000010h 02 00 00 00 00 F8 00 00 3F 00 FF 00 00 00 00 00 .....?.....
00000020h 00 00 78 00 F1 1D 00 00 00 00 00 00 02 00 00 00 ..x.....
00000030h 01 00 06 00 00 1D 00 00 00 00 00 00 00 00 00 00 .....
00000040h 80 00 29 01 18 B2 7E 4E 4F 20 4E 41 4D 45 20 20 ..) ...~NO NAME
00000050h 20 20 46 41 54 33 32 20 20 20 33 C9 8E D1 BC F4 FAT32 3....
00000060h 7B 8E C1 8E D9 BD 00 7C 88 4E 02 8A 56 40 B4 41 {...|.N..V@.A
00000070h BB AA 55 CD 13 72 10 81 FB 55 AA 75 0A F6 C1 01 ..U..r...U.u...
00000080h 74 05 FE 46 02 EB 2D 8A 56 40 B4 08 CD 13 73 05 t..F..-.V@....s.
00000090h B9 FF FF 8A F1 66 0F B6 C6 40 66 0F B6 D1 80 E2 .....f...@f.....
000000A0h 3F F7 E2 86 CD C0 ED 06 41 66 0F B7 C9 66 F7 E1 ?.....Af...f..
000000B0h 66 89 46 F8 83 7E 16 00 75 38 83 7E 2A 00 77 32 f.F..~...u8.~*.w2
000000C0h 66 8B 46 1C 66 83 C0 0C BB 00 80 B9 01 00 E8 2B f.F.f.....+
000000D0h 00 E9 2C 03 A0 FA 7D B4 7D 8B F0 AC 84 C0 74 17 ..,...}.}.....t.
000000E0h 3C FF 74 09 B4 0E BB 07 00 CD 10 EB EE A0 FB 7D <.t.....}
000000F0h EB E5 A0 F9 7D EB E0 98 CD 16 CD 19 66 60 80 7E ....}......f`.~
00000100h 02 00 0F 84 20 00 66 6A 00 66 50 06 53 66 68 10 .... .fj.fP.Sfh.
00000110h 00 01 00 B4 42 8A 56 40 8B F4 CD 13 66 58 66 58 ....B.V@....fXfX
00000120h 66 58 66 58 EB 33 66 3B 46 F8 72 03 F9 EB 2A 66 fXfX.3f;F.r...*f
00000130h 33 D2 66 0F B7 4E 18 66 F7 F1 FE C2 8A CA 66 8B 3.f..N.f.....f.
00000140h D0 66 C1 EA 10 F7 76 1A 86 D6 8A 56 40 8A E8 C0 .f....v....V@...
```

```

00000150h  E4 06 0A CC B8 01 02 CD 13 66 61 0F 82 75 FF 81  ....fa...u..
00000160h  C3 00 02 66 40 49 75 94 C3 42 4F 4F 54 4D 47 52  ...f@Iu..BOOTMGR
00000170h  20 20 20 20 00 00 00 00 00 00 00 00 00 00 00 00  ....
00000180h  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....
00000190h  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....
000001A0h  00 00 00 00 00 00 00 00 00 00 00 00 00 0D 0A 52 69  ....Ri
000001B0h  6D 75 6F 76 65 72 65 20 73 75 70 70 6F 72 74 69  muovere supporti
000001C0h  2E FF 0D 0A 45 72 72 6F 72 65 20 64 69 73 63 6F  ....Errore disco
000001D0h  FF 0D 0A 50 72 65 6D 65 72 65 20 75 6E 20 74 61  ...Premere un ta
000001E0h  73 74 6F 20 70 65 72 20 72 69 61 76 76 69 61 72  sto per riavviar
000001F0h  65 0D 0A 00 00 00 00 00 00 AC C2 D1 00 00 55 AA  e.....U.

```

Come si vede dalla tabella che segue, il *boot sector* contiene informazioni necessarie per stabilire il tipo di file system applicato, la dimensione del settore e del *cluster*, l'estensione del file system (e, quindi, della tabella FAT) e la posizione della tabella con il contenuto della directory *root* (il livello base del disco):

Posizione	Tipo	Significato
0B	WORD	Byte per settore (0x200=512)
0D	BYTE	Settori per cluster (8) → cluster da 4 KiB
0E	WORD	Settori riservati prima della FAT (0x41E)
15	BYTE	Tipo di disco (0xF8=fisso, chiavetta)
20	DWORD	Totale settori
2C	DWORD	N° del cluster con la directory radice (root)
52	CHAR[8]	Tipo di filesystem ² FAT (12, 16 o 32)

Mancando il settore di avvio, o in presenza di valori incoerenti, il sistema operativo non può tirare a indovinare il tipo di contenuto del disco, e perciò si arresta, considerandolo grezzo (RAW).

Per curiosità, notiamo anche i primi due byte del settore (gli ultimi due, come si vede, contengono e devono contenere, come il MBR, il valore 0xAA55): EBh 58h è in effetti un'istruzione della CPU, eseguita allorché il MBR richiama il settore di avvio.

Tale istruzione indica di far proseguire l'esecuzione saltando 0x58 byte (i quali, difatti, contengono le informazioni sul disco di cui abbiamo parlato, e altre ancora).

Un esempio di settore di avvio NTFS

Analoga importanza riveste il settore di avvio NTFS (tecnicamente, il settore di avvio NTFS prosegue nei 15 settori successivi con il codice di avvio: tuttavia, i dati essenziali sul file system si trovano come sempre nel settore zero):

```

0000000000h  EB 52 90 4E 54 46 53 20 20 20 20 00 02 08 00 00  .R.NTFS .....
0000000010h  00 00 00 00 00 00 F8 00 00 3F 00 FF 00 00 08 00 00  ....?.....
0000000020h  00 00 00 00 00 80 00 80 00 FF 7F 93 2E 00 00 00 00  ....
0000000030h  00 00 0C 00 00 00 00 00 FF 37 E9 02 00 00 00 00  ....7.....
0000000040h  F6 00 00 00 01 00 00 00 02 E7 54 AC 2C 55 AC B8  ....T.,U..
0000000050h  00 00 00 00 FA 33 C0 8E D0 BC 00 7C FB 68 C0 07  ....3.....|.h..
0000000060h  1F 1E 68 66 00 CB 88 16 0E 00 66 81 3E 03 00 4E  ..hf.....f.>..N
0000000070h  54 46 53 75 15 B4 41 BB AA 55 CD 13 72 0C 81 FB  TFSu..A..U..r...
0000000080h  55 AA 75 06 F7 C1 01 00 75 03 E9 D2 00 1E 83 EC  U.u.....u.....

```

² Poiché in passato tale identificativo non esisteva, molti programmi si basano in effetti sulla dimensione di settori e cluster per indurre il tipo di FAT in uso.

```

0000000090h 18 68 1A 00 B4 48 8A 16 0E 00 8B F4 16 1F CD 13 .h...H.....
00000000A0h 9F 83 C4 18 9E 58 1F 72 E1 3B 06 0B 00 75 DB A3 .....X.r.;...u..
00000000B0h 0F 00 C1 2E 0F 00 04 1E 5A 33 DB B9 00 20 2B C8 .....Z3...+.
00000000C0h 66 FF 06 11 00 03 16 0F 00 8E C2 FF 06 16 00 E8 f.....
00000000D0h 40 00 2B C8 77 EF B8 00 BB CD 1A 66 23 C0 75 2D @.+..w.....f#.u-
00000000E0h 66 81 FB 54 43 50 41 75 24 81 F9 02 01 72 1E 16 f..TCPAu$....r..
00000000F0h 68 07 BB 16 68 70 0E 16 68 09 00 66 53 66 53 66 h...hp..h..fSfSf
0000000100h 55 16 16 16 68 B8 01 66 61 0E 07 CD 1A E9 6A 01 U...h..fa.....j.
0000000110h 90 90 66 60 1E 06 66 A1 11 00 66 03 06 1C 00 1E ..f`..f...f.....
0000000120h 66 68 00 00 00 00 66 50 06 53 68 01 00 68 10 00 fh....fP.Sh..h..
0000000130h B4 42 8A 16 00 03 16 0F 8B F4 CD 13 66 59 5B 5A .B.....fY[Z
0000000140h 66 59 66 59 1F 0F 82 16 00 66 FF 06 11 00 03 16 fYfY.....f.....
0000000150h 0F 00 8E C2 FF 0E 16 00 75 BC 07 1F 66 61 C3 A0 .....u...fa..
0000000160h F8 01 E8 08 00 A0 FB 01 E8 02 00 EB FE B4 01 8B .....
0000000170h F0 AC 3C 00 74 09 B4 0E BB 07 00 CD 10 EB F2 C3 ..<.t.....
0000000180h 0D 0A 45 72 72 6F 72 65 20 6C 65 74 74 75 72 61 ..Errore lettura
0000000190h 20 64 61 20 64 69 73 63 6F 00 0D 0A 42 4F 4F 54 da disco...BOOT
00000001A0h 4D 47 52 20 6D 61 6E 63 61 6E 74 65 00 0D 0A 42 MGR mancante...B
00000001B0h 4F 4F 54 4D 47 52 20 63 6F 6D 70 72 65 73 73 6F OOTMGR compresso
00000001C0h 00 0D 0A 43 54 52 4C 2B 41 4C 54 2B 43 41 4E 43 ...CTRL+ALT+CANC
00000001D0h 20 70 65 72 20 72 69 61 76 76 69 61 72 65 0D 0A per riavviare..
00000001E0h 00 20 72 65 73 74 61 72 74 0D 0A 00 00 00 00 00 . restart.....
00000001F0h 00 00 00 00 00 00 00 80 9A AD C1 00 00 55 AA .....0.

```

Posizione	Tipo	Significato
03	CHAR[8]	Tipo di filesystem (NTFS)
0B	WORD	Byte per settore (0x200=512)
0D	BYTE	Settori per cluster (8) → cluster da 4 KiB
28	QWORD	Totale settori (0x2E937FFF, pari a 381.551 KiB)
30	QWORD	Cluster iniziale della Master File Table (0xC00000)
38	QWORD	Cluster iniziale della sua replica (\$MFTMirr)

Il settore di avvio NTFS contiene anche il numero del cluster iniziale della Master File Table (MFT), l'importantissimo file di sistema che cataloga l'intero contenuto del disco, e di cui si parlerà ampiamente. Spesso una copia dei primi 16 record della MFT (16 KiB) si trova anche immediatamente dopo il settore di avvio, data la sua cruciale importanza.

Come è registrato un file su FAT

Il contenuto di un file occupa un certo numero di cluster, anche sparsi, nel disco.

Il nome del file, la sua dimensione e il numero del suo primo cluster, sono annotati (insieme ad altri attributi, come data e ora) in una tabella di directory: possiamo immaginarla come un file che rimane invisibile.

Esaminando la FAT in corrispondenza del primo cluster del file, vi si troverà il numero del cluster successivo, oppure un indicatore di ultimo cluster.

Come si vede, ritrovare il contenuto di un file è semplice, quando (grazie al *boot sector*) si conosce la posizione della FAT e la sua estensione, nonché la posizione della tabella della *directory root* (a partire dalla quale si possono recuperare tutte le *directory* che discendono da essa).

Tali strutture, d'altro canto, sono facilmente riconoscibili a vista.

La File Allocation Table – FAT

La FAT non è altro che una tabella di numeri interi di eguale dimensione (12, 16 o 32 bit³ a seconda del tipo di FAT indicato nel settore zero), ognuno dei quali indica lo stato di un *cluster*.

I possibili stati del *cluster* sono rappresentati nella seguente tabella:

FAT12	FAT16	FAT32	EXFAT	Significato
0	0	0	0	Cluster libero
1	1	1	1	Valore riservato (non usato)
FF0-FF6	FFF0-FFF6	0FFFFFF0-0FFFFFF6	FFFFFFFF0-FFFFFFFF6	Valori riservati (non usati)
FF7	FFF7	0FFFFFF7	FFFFFFFF7	Cluster danneggiato
FF8-FFF	FFF8-FFFF	0FFFFFF8-0FFFFFFF	FFFFFFFF8-FFFFFFFFF	Cluster finale

Ogni altro valore presente in una posizione della FAT indica il numero del successivo *cluster* del file (o della tabella di *directory*).

Il primo cluster del file system è sempre il numero 2 (le prime due voci della tabella sono infatti riempite con valori di sistema), e si trova subito dopo l'ultima copia della FAT (di regola, la seconda copia).

Le copie della FAT si trovano al principio del disco, poco dopo il settore di avvio, esattamente dopo i settori riservati indicati nel *boot sector*; iniziano con un byte che identifica il tipo di disco (lo stesso byte trovato in posizione 0x15 del *boot sector*) seguito da alcuni byte a 0xFF; si compongono di sequenze numeriche, usualmente consecutive per vasti tratti, concluse da un marcatore di cluster finale.

L'estensione di ciascuna copia di FAT dipende, ovviamente, dal numero di *cluster* che compongono il file system, determinabile dai numeri di settori e di settori per cluster rilevabili nel settore di avvio.

Notiamo nell'esempio, in giallo, il byte 0xF8 che indica il tipo di disco e, in azzurro, la catena che inizia al cluster numero 3 e finisce al cluster numero 8.

```
00083C00h  F8 FF FF 0F FF FF FF FF 30 11 00 00 04 00 00 00 .....0.....
00083C10h  05 00 00 00 06 00 00 00 07 00 00 00 08 00 00 00 .....
00083C20h  FF FF FF 0F 0A 00 00 00 0B 00 00 00 0C 00 00 00 .....
00083C30h  0D 00 00 00 0E 00 00 00 0F 00 00 00 10 00 00 00 .....
00083C40h  11 00 00 00 12 00 00 00 13 00 00 00 14 00 00 00 .....
00083C50h  15 00 00 00 16 00 00 00 17 00 00 00 18 00 00 00 .....
00083C60h  19 00 00 00 1A 00 00 00 1B 00 00 00 1C 00 00 00 .....
00083C70h  FF FF FF 0F 1E 00 00 00 FF FF FF 0F FF FF FF 0F .....
00083C80h  D2 06 00 00 22 00 00 00 23 00 00 00 24 00 00 00 ...."....#...$...
00083C90h  25 00 00 00 26 00 00 00 6C 05 00 00 FF FF FF 0F %...&...1.....
00083CA0h  29 00 00 00 2A 00 00 00 2B 00 00 00 FF FF FF 0F )...*...+.....
00083CB0h  FF FF FF 0F FF FF FF 0F 2F 00 00 00 30 00 00 00 ...../...0...
00083CC0h  31 00 00 00 32 00 00 00 33 00 00 00 34 00 00 00 1...2...3...4...
00083CD0h  FF FF FF 0F FF FF FF 0F 37 00 00 00 38 00 00 00 .....7...8...
00083CE0h  39 00 00 00 3A 00 00 00 3B 00 00 00 3C 00 00 00 9...:...;...<...
00083CF0h  3D 00 00 00 3E 00 00 00 3F 00 00 00 40 00 00 00 =...>...?...@...
00083D00h  41 00 00 00 42 00 00 00 43 00 00 00 44 00 00 00 A...B...C...D...
00083D10h  45 00 00 00 46 00 00 00 47 00 00 00 48 00 00 00 E...F...G...H...
```

³ Gli unici file system ancora adatti ai dischi di oggi sono FAT32 (ed exFAT, al quale è riservato il capitolo successivo): FAT12 era il formato dei floppy, oggi praticamente scomparsi, e dei primi dischi rigidi di pochi MiB dei primi Anni '80; FAT16 con cluster da 32KiB andava bene per i dischi da non più di 2 GiB diffusi fino ai primi Anni 2000. Oggi, con chiavette USB e schede di memoria di svariati gigabyte, non si può prescindere da FAT32/exFAT.

00083D20h	49 00 00 00 4A 00 00 00 4B 00 00 00 4C 00 00 00	I...J...K...L...
00083D30h	4D 00 00 00 4E 00 00 00 4F 00 00 00 50 00 00 00	M...N...O...P...
00083D40h	51 00 00 00 52 00 00 00 53 00 00 00 54 00 00 00	Q...R...S...T...
00083D50h	55 00 00 00 56 00 00 00 57 00 00 00 58 00 00 00	U...V...W...X...
00083D60h	59 00 00 00 5A 00 00 00 5B 00 00 00 5C 00 00 00	Y...Z...[\...^...
00083D70h	5D 00 00 00 5E 00 00 00 5F 00 00 00 60 00 00 00]...^..._...`...
00083D80h	61 00 00 00 62 00 00 00 63 00 00 00 64 00 00 00	a...b...c...d...
00083D90h	65 00 00 00 66 00 00 00 67 00 00 00 68 00 00 00	e...f...g...h...
00083DA0h	69 00 00 00 6A 00 00 00 6B 00 00 00 6C 00 00 00	i...j...k...l...
00083DB0h	6D 00 00 00 6E 00 00 00 6F 00 00 00 70 00 00 00	m...n...o...p...
00083DC0h	71 00 00 00 72 00 00 00 73 00 00 00 74 00 00 00	q...r...s...t...
00083DD0h	75 00 00 00 76 00 00 00 77 00 00 00 78 00 00 00	u...v...w...x...
00083DE0h	79 00 00 00 7A 00 00 00 7B 00 00 00 7C 00 00 00	y...z...{... ...
00083DF0h	7D 00 00 00 7E 00 00 00 7F 00 00 00 80 00 00 00	}...~.....

La tabella di directory

Anche le tabelle di directory sono facilmente riconoscibili, poiché ognuna (tranne la *root*, che però si trova in una posizione fissa, subito dopo la seconda FAT 12 o 16; o nelle immediate vicinanze, se si tratta di FAT32) contiene in principio due pseudo file facilmente riconoscibili, “.” e “..” (rispettivamente, un riferimento alla directory corrente e a quella superiore).

Una tabella di directory è composta da una sequenza di voci (slot) di 32 byte, ciascuna contenente le informazioni relative a un file (o directory).

Da Windows 95 in poi, uno slot principale può essere preceduto da uno o più slot “speciali” che contengono il nome “lungo” (fino a 255 caratteri) del file.

Prima di Windows 95⁴, infatti, MS-DOS permetteva solo nomi “corti”, non più lunghi di 11 caratteri (8 di radice e 3 di estensione): il nome corto è quello che si trova nello slot di base di un file (o directory).

00800000h	56 4F 59 41 47 45 52 20 20 20 08 00 00 00 00 00	VOYAGER
00800010h	00 00 00 00 00 00 CC 8E 87 40 00 00 00 00 00 00@.....
00800020h	E5 57 52 44 32 31 35 33 54 4D 50 20 10 6E 59 4B	.WRD2153TMP .nYK
00800030h	9E 40 9E 40 00 00 E9 80 9E 40 09 78 8B A2 00 00	.@.@.....@.x...
00800040h	E5 57 52 4C 32 32 32 33 54 4D 50 02 10 6E 59 4B	.WRL2223TMP..nYK
00800050h	9E 40 9E 40 00 00 87 7A 9E 40 FF 77 14 9C 00 00	.@.@...z.@.w....
00800060h	E5 57 52 4C 33 37 33 39 54 4D 50 02 10 6E 59 4B	.WRL3739TMP..nYK
00800070h	9E 40 9E 40 00 00 2E 6D 9E 40 DC 77 CB 6C 00 00	.@.@...m.@.w.l..
00800080h	41 31 00 37 00 20 00 47 00 69 00 0F 00 41 75 00	A1.7. .G.i...Au.
00800090h	67 00 6E 00 6F 00 2E 00 64 00 00 00 6F 00 63 00	g.n.o...d...o.c.
008000A0h	31 37 47 49 55 47 7E 31 44 4F 43 20 00 91 E4 8E	17GIUG~1DOC
008000B0h	87 40 89 40 00 00 09 6A D8 3E 03 00 00 54 00 00	.@.@...j.>...T..
008000C0h	41 50 00 79 00 74 00 68 00 6F 00 0F 00 92 6E 00	AP.y.t.h.o....n.
008000D0h	00 00 FF FF FF FF FF FF FF FF 00 00 FF FF FF FF
008000E0h	50 59 54 48 4F 4E 20 20 20 20 20 10 00 29 E4 49	PYTHON ..).I
008000F0h	98 40 98 40 00 00 E5 49 98 40 1F 00 00 00 00 00	.@.@...I.@.....
00800100h	E5 24 46 4F 47 4C 7E 31 58 4C 53 22 00 99 70 49	.\$FOGL~1XLS"..pI
00800110h	98 40 98 40 00 00 74 49 98 40 C7 04 A5 00 00 00	.@.@...tI.@.....
00800120h	33 39 30 20 20 20 20 20 48 54 4D 20 00 98 E4 8E	390 HTM
00800130h	87 40 98 40 00 00 28 80 A2 3E 1D 00 F0 1A 00 00	.@.@..(>.....
00800140h	44 66 00 00 00 FF FF FF FF FF FF 0F 00 4D FF FF	Df.....M..
00800150h	FF FF FF FF FF FF FF FF FF FF 00 00 FF FF FF FF
00800160h	03 75 00 63 00 63 00 65 00 73 00 0F 00 4D 73 00	.u.c.c.e.s...Ms.
00800170h	69 00 6F 00 6E 00 65 00 2E 00 00 00 70 00 64 00	i.o.n.e....p.d.
00800180h	02 20 00 69 00 6E 00 20 00 63 00 0F 00 4D 61 00	. .i.n. .c...Ma.
00800190h	73 00 6F 00 20 00 64 00 69 00 00 00 20 00 73 00	s.o. .d.i... .s.
008001A0h	01 51 00 75 00 61 00 6C 00 69 00 0F 00 4D 20 00	.Q.u.a.l.i...M .
008001B0h	69 00 6D 00 70 00 6F 00 73 00 00 00 74 00 65 00	i.m.p.o.s...t.e.
008001C0h	51 55 41 4C 49 49 7E 31 50 44 46 20 00 15 D7 61	QUALII~1PDF ...a
008001D0h	9A 40 9A 40 00 00 DA 61 9A 40 20 00 CC 9F 0F 00	.@.@...a.@
008001E0h	41 37 00 49 00 6E 00 66 00 73 00 0F 00 9C 2E 00	A7.I.n.f.s.....
008001F0h	37 00 7A 00 00 00 FF FF FF FF 00 00 FF FF FF FF	7.z.....

⁴ Va notato che Windows NT 3.1 (destinato però al mercato aziendale) metteva già da qualche anno a disposizione il file system NTFS, che permette nomi di 255 caratteri e percorsi fino a 32768 caratteri complessivi.

La *root*, d'altro canto, si differenzia proprio perché non inizia con i file "." e ".." (contiene spesso l'etichetta del disco, "Voyager", in questo caso).

Le voci impilate che compongono il nome lungo (dall'ultima alla prima) contengono caratteri Unicode (riconoscibili dai byte a zero inframmezzati), e precedono il corrispondente nome di file corto.

Il secondo slot, ad esempio, appartiene a un file cancellato (il primo byte del nome è stato impostato a 0xE5) di tipo temporaneo (estensione TMP nel nome corto MS-DOS), lungo 0xA28B (41611) byte e iniziante al cluster 0x7809 (ricavabile dalle 2 WORD, in 0x14 e 0x1A marcate in giallo e verde: in FAT 12 e 16, il cluster iniziale è rappresentato dalla sola WORD in verde).

In posizione 8000E0h notiamo che il byte 0xB dello slot (in viola) contiene il valore 0x10, il quale indica che l'oggetto è una directory: la sua dimensione è, infatti, impostata a zero byte (ultima DWORD), mentre il cluster iniziale della tabella di directory è a 0x1F.

Posizione	Tipo	Significato (slot ordinario)
00	CHAR[8]	Nome (MS-DOS) del file
08	CHAR[3]	Estensione del file
0B	BYTE	Permessi MS-DOS (0x10=directory)
14	WORD	16-bit alti del n° del primo cluster (solo FAT32)
1A	WORD	16-bit bassi del n° del primo cluster
1C	DWORD	Dimensione del file

Posizione	Tipo	Significato (slot di LFN)
00	BYTE	Numero sequenziale dello slot (+0x40 se ultimo)
0B	BYTE	Impostato sempre a 0x0F
0C	BYTE	Impostato sempre a 0x00
1A	WORD	Impostato sempre a 0x0000

Considerazioni finali

Da quanto detto, risulta semplice ricostruire manualmente il contenuto di un file, o progettare un programma che lo faccia automaticamente, previo inserimento manuale di alcuni dati (posizione della FAT, della *root*, dimensione del cluster e della voce di FAT), ricavabili dall'esame del file system tramite un editore esadecimale.

Anche se non si dispone di settore di avvio, guardando la FAT è facile capire se si tratta di FAT 12, 16 o 32; e, dal termine della prima copia, indurre il numero di cluster e, dunque, di settori.

D'altra parte, la dimensione del disco è conoscibile dal sistema operativo semplicemente interrogando l'elettronica di esso; se troviamo qualche tabella di directory, esaminando il numero di cluster iniziale di un file di contenuto ben noto o tipico, possiamo, egualmente, ricostruire la dimensione del cluster.

Tale semplicità importa però anche due evidenti limitazioni.

In primo luogo, si è visto che la dimensione di un file è espressa da una DWORD: quindi, non è possibile registrare file maggiori di 4 GiB⁵. Inoltre, poiché anche il numero totale di settori è registrato con una DWORD, il file system non può eccedere i 2 TiB con settori standard da 512 byte⁶.

⁵ Tale limitazione è rimossa in exFAT. Si noti pure che la FAT32 impiega in effetti solo 28 bit per indicizzare i cluster in tabella: ne consegue che la massima capacità di un file system FAT32 con cluster da 32 KiB è di 8 TiB teorici.

In secondo luogo, le strutture indispensabili per recuperare un file (slot della tabella di directory⁷ e catena di cluster nella FAT) sono molto piccole e concentrate, e perciò molto esposte al rischio di danneggiamento. Se ciò era particolarmente vero con i floppy (nei quali era facile perdere 5-10 settori consecutivi per qualche anomalia di rotazione o di allineamento delle testine di scrittura), non bisogna credere che il guasto di un settore di disco rigido sia un evento così improbabile: l'accesso in lettura e scrittura ai medesimi settori della FAT accade in media migliaia e migliaia di volte anche in una sola sessione; inoltre, la posizione della FAT è fissa, non può essere rilocata in diverse aree del disco⁸, ad esempio tramite programmi di deframmentazione.

Come è registrato un file su exFAT

Il file system exFAT (disponibile dal 2008 per Windows XP e superiori) si discosta dalla famiglia FAT per introdurre alcuni miglioramenti prestazionali e raggiungere nuovi limiti dimensionali, pur senza aumentare enormemente la complessità del codice.

Le differenze di maggior rilievo⁹ rispetto a FAT32 sono:

- applicabile anche a unità inferiori a 512 MiB;
- cluster maggiorato fino a 32 MiB;
- indici di cluster a 32 bit anziché a 28;
- limite di grandezza del singolo file a 2^{64} byte anziché 2^{32} ;
- impiego della FAT (in singola copia) per i soli file non contigui (e la *root* directory);
- impiego di una bitmap per l'allocazione dei cluster;
- razionalizzazione delle tabelle di directory, mediante diverse tipologie di slot (eliminazione dei nomi corti 8+3; orari in formato UTC con risoluzione a 1/100 di secondo; *checksum* sui nomi per accelerare le ricerche).

Un evidente svantaggio di exFAT è che, al momento, non esiste un codice di avvio: esso può essere usato solo su dischi di dati.

Inoltre, la compatibilità con altri sistemi operativi o hardware¹⁰ non è ancora vastissima quanto FAT32.

Esempio di settore di avvio exFAT

```
00000000h  EB 76 90 45 58 46 41 54 20 20 20 00 00 00 00 00 00  .v.EXFAT  ....
00000010h  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
```

⁶ Che difatti è il limite indicato da Microsoft nella propria specifica.

⁷ In effetti, mancando lo slot (che determina cluster iniziale e lunghezza esatta), si può almeno recuperare la catena "orfana" di cluster e cercare di indovinare la dimensione del file guardando l'ultimo cluster. Mancando la catena di FAT, invece, è pressoché impossibile ricostruire il contenuto del file, a meno che: 1) esso risiede nel solo primo cluster; oppure 2) si tratta di un file in più cluster consecutivi.

⁸ Probabilmente sarebbe possibile solo far "avanzare" la FAT aumentando il numero di settori riservati dopo la *boot sector*: a patto, naturalmente, che lo spazio immediatamente successivo a essa fosse libero. Ciò permetterebbe di confinare nell'area riservata eventuali settori guasti.

⁹ Implementate in concreto nella versione 1.0. Altre caratteristiche, quali il meccanismo transazionale di scrittura mediante una seconda copia di FAT e le Access Control Lists (ACL), sono state solo pianificate.

¹⁰ FAT32 è, di fatto, anche formato di interscambio tra piattaforme diverse (Mac OS, Linux) e macchine elettroniche che usano dischi rigidi o schede di memoria, come lettori DVD e DVD recorder, macchine fotografiche, lettori MP3 portatili e da tavolo, videocamere, telefonini, ecc. ecc. Proprio per questi ultimi dispositivi Microsoft ha, in un primo tempo, sviluppato exFAT.

```

00000020h  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00000030h  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00000040h  80 00 00 00 00 00 00 00 00 00 60 77 00 00 00 00  .....`w.....
00000050h  80 00 00 00 C0 03 00 00 80 04 00 00 6E DD 01 00  .....n....
00000060h  04 00 00 00 64 2E B3 5E 00 01 00 00 09 06 01 80  ....d..^.....
00000070h  47 00 00 00 00 00 00 00 33 C9 8E D1 BC F0 7B 8E  G.....3.....{.
00000080h  D9 A0 FB 7D B4 7D 8B F0 AC 98 40 74 0C 48 74 0E  ...}.)....@t.Ht.
00000090h  B4 0E BB 07 00 CD 10 EB EF A0 FD 7D EB E6 CD 16  .....}.....
000000A0h  CD 19 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
000000B0h  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
000000C0h  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
000000D0h  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
000000E0h  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
000000F0h  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00000100h  0D 0A 52 69 6D 75 6F 76 65 72 65 20 73 75 70 70  ..Rimuovere supp
00000110h  6F 72 74 69 2E FF 0D 0A 45 72 72 6F 72 65 20 64  orti....Errore d
00000120h  69 73 63 6F FF 0D 0A 50 72 65 6D 65 72 65 20 75  isco...Premere u
00000130h  6E 20 74 61 73 74 6F 20 70 65 72 20 72 69 61 76  n tasto per riav
00000140h  76 69 61 72 65 0D 0A 00 00 00 00 00 00 00 00 00  viare.....
00000150h  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00000160h  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00000170h  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00000180h  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00000190h  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
000001A0h  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
000001B0h  00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF FF  .....
000001C0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  .....
000001D0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  .....
000001E0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  .....
000001F0h  FF FF FF FF FF FF FF FF FF FF FF FF 00 16 25 55 AA  .....%U.

```

Posizione	Tipo	Significato
03	CHAR[8]	Tipo di filesystem (EXFAT)
48	QWORD	Numero di settori del filesystem
50	DWORD	Posizione della FAT (in settori)
54	DWORD	Lunghezza della FAT (in settori)
58	DWORD	Posizione del primo cluster (in settori)
5C	DWORD	Dimensione dell'area dati (in cluster)
60	DWORD	Cluster iniziale della root directory
6C	BYTE	Byte per settore (esponente di 2)
6D	BYTE	Settori per cluster (esponente di 2)
6E	BYTE	Numero di copie di FAT

La tabella di directory

Le tabelle di directory exFAT sono meno riconoscibili, poiché non contengono i riferimenti “punto” alla directory corrente o superiore né gli slot di nome corto (la *root* è riconoscibile dalla presenza dello slot di tipo 0x3 per l’etichetta del volume); d’altro canto, risultano strutturati in modo più razionale.

Ogni file (o directory) è sempre definito da un minimo di 3, a un massimo di 19, slot consecutivi: il primo, di tipo 0x05, contiene date, attributi e numero di slot secondari che seguono; il secondo, di tipo 0x40, contiene informazioni sulla lunghezza del file, sulla posizione del primo cluster e sulla validità della catena FAT (se un file non ha catena, ciò significa che i suoi cluster sono contigui); il terzo, ed eventuali slot successivi, di tipo 0x41, contengono i caratteri Unicode componenti il nome dell’oggetto.

```

000A0000h  83 08 56 00 65 00 72 00 62 00 61 00 74 00 69 00  ..V.e.r.b.a.t.i.
000A0010h  6D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  m.....
000A0020h  81 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
000A0030h  00 00 00 00 02 00 00 00 AE 3B 00 00 00 00 00 00  .....;.....
000A0040h  82 00 00 00 0D D3 19 E6 00 00 00 00 00 00 00 00  .....
000A0050h  00 00 00 00 03 00 00 00 CC 16 00 00 00 00 00 00  .....
000A0060h  85 02 7F 77 20 00 00 00 36 77 A7 40 3A 8C 39 40  ...w ...6w.@:.9@
000A0070h  36 77 A7 40 AF 00 88 88 88 00 00 00 00 00 00 00 00 6w.@.....
000A0080h  C0 03 00 0A A1 2D 00 00 FC B6 00 00 00 00 00 00 00  ....-.....
000A0090h  00 00 00 00 05 00 00 00 FC B6 00 00 00 00 00 00 00  .....
000A00A0h  81 00 31 00 31 00 36 00 31 00 43 00 4E 00 2E 00  ..1.1.6.1.C.N...
000A00B0h  70 00 64 00 66 00 00 00 00 00 00 00 00 00 00 00 00  p.d.f.....
000A00C0h  85 03 BB 1E 20 00 00 00 36 77 A7 40 2E 4B 3B 40  .... ...6w.@.K;@
000A00D0h  36 77 A7 40 C4 00 88 88 88 00 00 00 00 00 00 00 00 6w.@.....
000A00E0h  C0 03 00 17 E8 89 00 00 39 40 09 00 00 00 00 00 00  ....9@.....
000A00F0h  00 00 00 00 07 00 00 00 39 40 09 00 00 00 00 00 00  ....9@.....
000A0100h  C1 00 37 00 33 00 30 00 5F 00 32 00 30 00 31 00  ..7.3.0._.2.0.1.
000A0110h  32 00 5F 00 69 00 73 00 74 00 72 00 75 00 7A 00  2._.i.s.t.r.u.z.
000A0120h  C1 00 69 00 6F 00 6E 00 69 00 2E 00 70 00 64 00  ..i.o.n.i...p.d.
000A0130h  66 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  f.....
000A0140h  85 03 7D FC 20 00 00 00 37 77 A7 40 2B 4B 3B 40  ..}. ...7w.@+K;@
000A0150h  37 77 A7 40 11 00 88 88 88 00 00 00 00 00 00 00 00 7w.@.....
000A0160h  C0 03 00 14 83 3A 00 00 3C CE 08 00 00 00 00 00 00  ....:..<.....
000A0170h  00 00 00 00 1A 00 00 00 3C CE 08 00 00 00 00 00 00  ....<.....
000A0180h  C1 00 37 00 33 00 30 00 5F 00 32 00 30 00 31 00  ..7.3.0._.2.0.1.
000A0190h  32 00 5F 00 6D 00 6F 00 64 00 65 00 6C 00 6C 00  2._.m.o.d.e.1.1.
000A01A0h  C1 00 69 00 2E 00 70 00 64 00 66 00 00 00 00 00 00  ..i...p.d.f.....
000A01B0h  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
000A01C0h  85 03 45 DC 20 00 00 00 37 77 A7 40 17 4B 3B 40  ..E. ...7w.@.K;@
000A01D0h  37 77 A7 40 4E 00 88 88 88 00 00 00 00 00 00 00 00 7w.@N.....
000A01E0h  C0 03 00 1D 60 91 00 00 87 4F 0D 00 00 00 00 00 00  ....`....O.....
000A01F0h  00 00 00 00 2C 00 00 00 87 4F 0D 00 00 00 00 00 00  ....,....O.....

```

Come si nota dai byte evidenziati in rosso, ogni slot inizia con un byte che ne identifica il tipo (i primi 5 bit) e lo status: se il settimo bit non è impostato, lo slot non è in uso e può essere sovrascritto. I tipi di slot sono:

In uso	Tipo di slot
81	Bitmap
82	Tabella UpCase
83	Etichetta del disco
85	Voce principale di file (o directory)
A0	GUID
A1	Padding
C0	Contenuto del file (o directory)
C1	Nome del file (o directory) (da 1 a 17 slot)

Lo slot di tipo 0x05 (0x85 se attivo) segna l'inizio di un nuovo gruppo definitorio di un oggetto (file o directory):

Posizione	Tipo	Significato (slot tipo 5)
00	BYTE	Tipo (0x85=in uso, 0x05=cancellato)
01	BYTE	Numero di slot ulteriori nel gruppo
04	WORD	Permessi MS-DOS
08,0C,10	DWORD	Date e ore di creazione, modifica, ultimo accesso
14,15	BYTE	Millisecondi per creazione e modifica
16-18	BYTE	Fusi orari per creazione, modifica, ultimo accesso

Il successivo slot di *stream extension* (tipo 0xC0 o 0x40) definisce i dati necessari per ritrovare il contenuto:

Posizione	Tipo	Significato (slot tipo 0x41)
00	BYTE	Tipo (0xC0=in uso, 0x40=cancellato)
01	BYTE	Flags (0x02=file contiguo, non usa catena di FAT)
03	BYTE	Lunghezza del nome del file (max. 255)
14	DWORD	Numero del primo cluster
18	QWORD	Dimensione del file

Segue, come detto, almeno uno slot 0xC1 recante il nome del file (o directory).

Considerazioni finali

Paradossalmente, la recuperabilità dei file da un file system exFAT corrotto potrebbe essere anche più problematica di uno FAT32.

Quando un file è contiguo, occupa, cioè, solo cluster consecutivi, la catena di FAT non è usata: dunque, qualora si corrompesse lo slot del file nella tabella di directory, sarebbe pressoché impossibile stabilire il cluster iniziale del contenuto.

Inoltre, mancando i riferimenti al cluster della directory superiore (..), tipici di FAT12/16/32, potrebbe rivelarsi impossibile ricostruire la gerarchia fra le directory.

D'altro canto, tale struttura permette di ridurre il numero di scritture necessarie a registrare un file (la modifica della *bitmap* è più breve di quella delle catene di FAT) e, conseguentemente, l'usura del disco¹¹.

Come è registrato un file su NTFS

Il file system NTFS è estremamente più complesso, essendo disegnato per rispondere con efficienza a tutta una serie di problemi posti da FAT, non da ultimo i limiti dimensionali di dischi e file.

Non è questa la sede per trattare tutte le caratteristiche di NTFS e la sua evoluzione, dall'introduzione in Windows NT 3.1¹² negli Anni '90 a oggi.

Per quanto concerne il profilo della recuperabilità dei dati, che qui ci interessa, basterà dire che ogni file (o indice di directory) ha un proprio **record** di 1024 byte nella Master File Table o MFT del file system, il quale riporta la dislocazione dei cluster di quel file (o indice di directory).

In altre parole, il singolo record contiene di regola¹³ tutti i dati necessari, e sufficienti, per recuperare il file.

¹¹ Non va dimenticato che le stesse celle di memoria di chiavette USB o schede SD sono soggette a usura più o meno rapida (e protette perciò da un meccanismo interno di ricircolo dei settori, invisibile all'utente): per tale motivo, il file system NTFS, pur essendo applicabile anche a tali tipi di dischi, non è consigliabile, a causa dell'elevato numero di scritture aggiuntive richieste per ogni modifica dalle infrastrutture di *journaling*.

¹² Le versioni di NTFS sono, di solito (e con l'eccezione di Windows NT 3.1), compatibili non solo in avanti, ma anche all'indietro: un s/o più vecchio riesce a leggere un file system più nuovo. Uno spiacevole comportamento di Windows 7 (ma, probabilmente, anche dei precedenti s/o) è che, al primo accesso a un file system NTFS di vecchia versione, Windows lo aggiorna a quella più recente, determinando con ciò possibili problemi di lettura da parte del sistema che ha formattato quel disco.

¹³ Vedremo che, eccezionalmente, il contenuto di un file può essere descritto da un insieme di 2 o più record.

Come possiamo vedere dall'esempio che segue, la stessa MFT ha un record di file denominato \$MFT (normalmente nascosto all'utente) che descrive il proprio contenuto:

00C0000000h	46 49 4C 45	30 00 03 00	4C 16 5B 46	00 00 00 00	FILE0...L.[F...
00C0000010h	01 00 01 00	38 00 01 00	A8 01 00 00	00 04 00 008.....
00C0000020h	00 00 00 00	00 00 00 00	07 00 00 00	00 00 00 00
00C0000030h	F2 00 00 00	00 00 00 00	10 00 00 00	60 00 00 00`...
00C0000040h	00 00 18 00	00 00 00 00	48 00 00 00	18 00 00 00H.....
00C0000050h	C0 CC A1 4C	61 A2 CA 01	C0 CC A1 4C	61 A2 CA 01	...La.....La...
00C0000060h	C0 CC A1 4C	61 A2 CA 01	C0 CC A1 4C	61 A2 CA 01	...La.....La...
00C0000070h	06 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C0000080h	00 00 00 00	00 01 00 00	00 00 00 00	00 00 00 00
00C0000090h	00 00 00 00	00 00 00 00	30 00 00 00	68 00 00 000...h...
00C00000A0h	00 00 18 00	00 00 03 00	4A 00 00 00	18 00 01 00J.....
00C00000B0h	05 00 00 00	00 00 05 00	C0 CC A1 4C	61 A2 CA 01La...
00C00000C0h	C0 CC A1 4C	61 A2 CA 01	C0 CC A1 4C	61 A2 CA 01	...La.....La...
00C00000D0h	C0 CC A1 4C	61 A2 CA 01	00 00 38 0E	00 00 00 00	...La.....8.....
00C00000E0h	00 00 38 0E	00 00 00 00	06 00 00 00	00 00 00 00	..8.....
00C00000F0h	04 03 24 00	4D 00 46 00	54 00 00 00	00 00 00 00	..\$.M.F.T.....
00C0000100h	80 00 00 00	50 00 00 00	01 00 40 00	00 00 01 00P.....@.....
00C0000110h	00 00 00 00	00 00 00 00	BF 1F 01 00	00 00 00 00
00C0000120h	40 00 00 00	00 00 00 00	00 00 FC 11	00 00 00 00	@.....
00C0000130h	00 00 FC 11	00 00 00 00	00 00 FC 11	00 00 00 00
00C0000140h	33 20 C8 00	00 00 0C 42	A0 57 6F 52	04 03 00 8A	3B.WoR....
00C0000150h	B0 00 00 00	50 00 00 00	01 00 40 00	00 00 06 00P.....@.....
00C0000160h	00 00 00 00	00 00 00 00	09 00 00 00	00 00 00 00
00C0000170h	40 00 00 00	00 00 00 00	00 A0 00 00	00 00 00 00	@.....
00C0000180h	C0 91 00 00	00 00 00 00	C0 91 00 00	00 00 00 00
00C0000190h	31 01 FF FF	0B 31 09 2E	A3 74 00 00	00 10 8E 8A	1...1...t.....
00C00001A0h	FF FF FF FF	00 00 00 00	FF FF FF FF	00 00 00 00
00C00001B0h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C00001C0h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C00001D0h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C00001E0h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C00001F0h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 F2 00
00C0000200h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C0000210h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C0000220h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C0000230h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C0000240h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C0000250h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C0000260h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C0000270h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C0000280h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C0000290h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C00002A0h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C00002B0h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C00002C0h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C00002D0h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C00002E0h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C00002F0h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C0000300h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C0000310h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C0000320h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C0000330h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C0000340h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C0000350h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C0000360h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C0000370h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C0000380h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C0000390h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C00003A0h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C00003B0h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C00003C0h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C00003D0h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C00003E0h	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00C00003F0h	00 00 00 00	00 00 00 00	00 00 00 00	00 F2 00

Notiamo in posizione iniziale la stringa FILE, che designa un record valido, e in posizione 0x16 (azzurro) una WORD impostata a 1 che contrassegna il record come in uso.

I primi 48 byte del record contengono la sua intestazione standard, recante alcuni dati molto importanti:

Posizione	Tipo	Significato
00	CHAR[4]	Contrassegno FILE
14	WORD	Primo attributo del record (38h)
16	WORD	Flags (0x1 = in uso; 0x2 = directory)
20	QWORD	Record MFT cui appartiene (0x00 = file di sistema)
2C	DWORD	Numero di questo record MFT

Il resto del record contiene uno o più **attributi**, di dimensione e contenuto variabile¹⁴.

Ognuno inizia con una DWORD che ne designa il tipo (in verde), e una WORD che specifica la lunghezza dell'attributo (in grigio).

Nel record \$MFT, notiamo alcuni attributi di interesse:

Posizione	Nome e valore
38	\$STANDARD_INFORMATION (10h)
98	\$FILE_NAME (30h)
100	\$DATA (80h)
150	\$BITMAP (B0h)
1A0	Marcatore di fine attributi

I 2 attributi essenziali per il recupero del file sono \$FILE_NAME e \$DATA.

Gli attributi del record MFT

Un attributo inizia con un'intestazione generica (i primi 0x10 byte), il cui contenuto saliente è il seguente:

Posizione	Tipo	Significato
00	DWORD	Tipo di attributo (0xFFFFFFFF = fine attributi)
04	DWORD	Lunghezza dell'attributo
08	BYTE	Flag (0x01 = contenuto non residente nel record)
09	BYTE	Lunghezza del nome dell'attributo
0A	WORD	Posizione del nome dell'attributo

Segue un'intestazione specifica, che è lunga 8 byte, se il contenuto dell'attributo è residente nel record:

Posizione	Tipo	Significato
10	DWORD	Dimensione del contenuto dell'attributo
14	WORD	Posizione del contenuto (dall'inizio attributo)

oppure 0x30 byte, quando il contenuto non è residente:

¹⁴ I cui nomi e caratteristiche salienti sono, fra l'altro, descritti in uno speciale file di sistema, \$AttrDef.

Posizione	Tipo	Significato
20	WORD	Posizione dei Datarun
22	WORD	Blocco di compressione ¹⁵ (0 = non compresso)
28	QWORD	Byte allocati per il contenuto (arrotondato al cluster)
30	QWORD	Dimensione effettiva del contenuto

L'attributo \$FILE_NAME, oltre al nome del file (o directory) in caratteri Unicode, reca anche date, ore e dimensione del file (offset 48h dall'inizio dell'attributo, in viola: E380000h).

\$DATA contiene, sotto forma di *datarun*, le informazioni necessarie per trovare i cluster con il contenuto del file; oppure, direttamente quel contenuto, quando esso è residente.

Va detto che un file può avere più \$FILE_NAME (ad esempio, è comune che un file abbia un nome corto MS-DOS e uno "normale"), ma anche più attributi \$DATA, ossia più contenuti: il contenuto principale è sempre riposto in un attributo \$DATA innominato.

Quando il contenuto del file è sufficientemente piccolo (e ciò vale per il contenuto di qualsiasi altro attributo, ad esempio, un indice di directory o una bitmap), esso può rimanere residente all'interno del record stesso: tale condizione è segnalata dal valore zero del BYTE in posizione 0x08 dell'attributo.

La lista di attributi \$ATTR_LIST

Può capitare, in rare occasioni, che un record solo non sia sufficiente per contenere tutti gli attributi del file: in tal caso, subito dopo \$STANDARD_INFORMATION compare l'attributo \$ATTR_LIST (20h), con un elenco di tutti gli attributi annessi all'oggetto e dei record nei quali ciascuno risiede.

Posizione	Tipo	Significato
00	DWORD	Tipo di attributo in lista
04	WORD	Lunghezza della voce di lista
10	QWORD	N° del record sul quale si trova

I datarun

La WORD in posizione 20h dell'attributo \$DATA (in rosso) specifica la posizione dei datarun: in questo caso, 0x40 byte dopo l'inizio dell'attributo \$DATA, ossia a partire da C0000140h.

Un datarun non è altro che la rappresentazione di un segmento del file, che si estende per un certo numero di cluster a partire da un dato cluster (LCN, Logical Cluster Number, o numero di cluster a partire dall'inizio del file system).

Un file può consistere di un datarun solo, oppure essere frammentato, e avere più datarun.

Nei datarun successivi al primo, la base del segmento, anziché essere espressa con un LCN, è indicata mediante il numero (con segno) dei cluster che seguono o precedono la base del segmento precedente.

¹⁵ Qualora il contenuto sia compresso, non sarà possibile recuperarlo "manualmente", poiché l'espansione dei dati richiede una complessa elaborazione da parte della CPU.

Tutti i numeri sono poi “compressi” con un ingegnoso sistema che evita di registrare byte inutili.

In particolare, ogni datarun inizia con un byte di “legenda” il quale specifica quanti dei byte successivi compongono la lunghezza del segmento (i primi 4 bit, ossia il numero esadecimale più a destra) e quanti la base (gli altri 4 bit)¹⁶.

Nell’esempio, la sequenza: 33 20 C8 00 00 00 0C specifica che la lunghezza del primo segmento è rappresentata dai prossimi 3 byte (20 C8 00), e il LCN dai 3 byte che li seguono (00 00 0C).

Naturalmente, si tratta di numeri in formato Little Endian, nella specie di due DWORD: 0xC820 (la lunghezza in cluster del segmento), e 0xC0000 (il LCN iniziale della \$MFT).

Il datarun successivo, 42 A0 57 6F 52 04 03, impiega invece 2 byte per codificare la lunghezza del secondo segmento (il nibble a “2” in 0x42) e 4 byte per lo scostamento in cluster della base di tale segmento da quella del precedente (il nibble a “4” in 0x42).

Avremo, perciò, lunghezza di 0x57A0 cluster, e offset al LCN (0xC0000 + 0x304526F=) 0x310526F.

Va subito sottolineato che, quando l’ultimo byte di un offset è 0x80 o più, il numero codificato è negativo. Perciò, un secondo datarun 31 01 17 FB DA indicherebbe un segmento, lungo un cluster, la cui base si trova a 0xFFDAFB17 (-2.426.089) cluster dal LCN precedente.

Il byte di legenda a zero segnala la fine dei datarun: la \$MFT consiste dunque, in questo caso, di due segmenti, il primo di circa 200 MiB (cluster da 4 KiB), e il secondo di circa 90 MiB.

Gli indici di directory

Diversamente dal file system FAT, in NTFS gli indici di directory sono strutture secondarie, rigenerabili a partire dalla MFT, e conservati su disco solo per una questione di efficienza.

Abbiamo visto che un record FILE della MFT contiene in 0x20 una QWORD con il numero del record padre: quando il record padre è un record di directory, il file è contenuto in quella directory; il record di questa punterà ovviamente a un altro record di directory, e così via, fino ad arrivare a una directory che punta al record n° 5, il cui nome è “.” (la directory radice del file system).

Ovviamente, poiché sarebbe estremamente inefficiente analizzare ogni volta tutti i record per stabilire le gerarchie tra file e cartelle, queste sono archiviate negli indici, che sono tecnicamente indici di attributi \$FILE_NAME (anche altri attributi, infatti, possono avere i loro indici).

```
00C0001400h 46 49 4C 45 30 00 03 00 E8 55 00 74 00 00 00 00 FILE0....U.t...
00C0001410h 05 00 01 00 38 00 03 00 68 03 00 00 00 04 00 00 ....8...h.....
00C0001420h 00 00 00 00 00 00 00 00 1B 00 00 00 05 00 00 00 .....
00C0001430h 67 03 00 05 00 00 00 00 10 00 00 00 48 00 00 00 g.....H...
00C0001440h 00 00 18 00 00 00 00 00 30 00 00 00 18 00 00 00 .....0.....
00C0001450h C0 CC A1 4C 61 A2 CA 01 AC 4A 96 AA 2F 25 CD 01 ...La....J../%..
00C0001460h AC 4A 96 AA 2F 25 CD 01 AC 4A 96 AA 2F 25 CD 01 .J../%...J../%..
00C0001470h 06 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00C0001480h 30 00 00 00 60 00 00 00 00 00 00 00 18 00 00 01 00 0...`.....
00C0001490h 44 00 00 00 18 00 01 00 05 00 00 00 00 00 05 00 D.....
00C00014A0h C0 CC A1 4C 61 A2 CA 01 6C 6B 47 41 0C 3B CB 01 ...La...lkGA.;..
00C00014B0h 6C 6B 47 41 0C 3B CB 01 6C 6B 47 41 0C 3B CB 01 lkGA.;..lkGA.;..
00C00014C0h 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00C00014D0h 06 00 00 10 00 00 00 00 01 03 2E 00 00 00 00 00 .....
00C00014E0h 40 00 00 00 28 00 00 00 00 00 00 00 00 00 0A 00 @... (.....
```

¹⁶ Se la base è lunga zero byte, si intende che il blocco contiene solo cluster pieni di byte zero, e non è scritto fisicamente su disco (c.d. “sparse file”, solo nelle più recenti versioni del file system).

00C00014F0h	10 00 00 00 18 00 00 00 40 A6 59 1C 2A 46 DF 11@.Y.*F..
00C0001500h	B0 9F 00 16 36 F8 A0 3B 50 00 00 00 00 01 00 006...;P.....
00C0001510h	00 00 18 00 00 00 02 00 E4 00 00 00 18 00 00 00
00C0001520h	01 00 04 80 CC 00 00 00 D8 00 00 00 00 00 00 00
00C0001530h	14 00 00 00 02 00 B8 00 08 00 00 00 00 18 00
00C0001540h	FF 01 1F 00 01 02 00 00 00 00 05 20 00 00 00
00C0001550h	20 02 00 00 00 0B 18 00 00 00 10 01 02 00 00
00C0001560h	00 00 00 05 20 00 00 00 20 02 00 00 00 14 00
00C0001570h	FF 01 1F 00 01 01 00 00 00 00 05 12 00 00 00
00C0001580h	00 0B 14 00 00 00 10 01 01 00 00 00 00 00 05
00C0001590h	12 00 00 00 00 14 00 BF 01 13 00 01 01 00 00
00C00015A0h	00 00 00 05 0B 00 00 00 0B 14 00 00 00 01 E0
00C00015B0h	01 01 00 00 00 00 05 0B 00 00 00 00 18 00
00C00015C0h	A9 00 12 00 01 02 00 00 00 00 05 20 00 00 00
00C00015D0h	21 02 00 00 00 0B 18 00 00 00 A0 01 02 00 00	!.....
00C00015E0h	00 00 00 05 20 00 00 00 21 02 00 00 01 01 00!.....
00C00015F0h	00 00 00 05 12 00 00 00 01 01 00 00 00 67 03g.
00C0001600h	12 00 00 00 00 00 00 00 90 00 00 00 58 00 00X...
00C0001610h	00 04 18 00 00 00 18 00 38 00 00 00 20 00 008...
00C0001620h	24 00 49 00 33 00 30 00 30 00 00 00 01 00 00	\$.I.3.0.0.....
00C0001630h	00 10 00 00 01 00 00 00 10 00 00 00 28 00 00(...
00C0001640h	28 00 00 00 01 00 00 00 00 00 00 00 00 00 00	(.....
00C0001650h	18 00 00 00 03 00 00 00 03 00 00 00 00 00 00
00C0001660h	A0 00 00 00 70 00 00 00 01 04 40 00 00 1A 00p.....@.....
00C0001670h	00 00 00 00 00 00 00 00 0E 00 00 00 00 00 00
00C0001680h	48 00 00 00 00 00 00 00 00 F0 00 00 00 00 00	H.....
00C0001690h	00 F0 00 00 00 00 00 00 00 F0 00 00 00 00 00
00C00016A0h	24 00 49 00 33 00 30 00 41 05 96 F2 B2 00 31 01	\$.I.3.0.A.....1.
00C00016B0h	17 FB DA 41 01 61 CD 0D 02 41 01 F1 44 93 00 31	...A.a...A..D..1
00C00016C0h	03 9D 00 A7 31 01 82 04 59 41 03 BE 58 DC FC 001...YA..X...
00C00016D0h	B0 00 00 00 28 00 00 00 04 18 00 00 00 19 00(.....
00C00016E0h	08 00 00 00 20 00 00 00 24 00 49 00 33 00 30\$.I.3.0.
00C00016F0h	7F 40 00 00 00 00 00 00 01 00 00 68 00 00 00	.@.....h...
00C0001700h	00 09 18 00 00 00 09 00 38 00 00 00 30 00 008...0...
00C0001710h	24 00 54 00 58 00 46 00 5F 00 44 00 41 00 54	\$.T.X.F._.D.A.T.
00C0001720h	41 00 00 00 00 00 00 00 05 00 00 00 00 05 00	A.....
00C0001730h	01 00 00 00 01 00 00 00 00 00 00 00 00 00 00
00C0001740h	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00C0001750h	00 00 00 00 00 00 00 00 02 00 00 00 00 00 00
00C0001760h	FF FF FF FF 00 00 00 00 00 00 00 00 00 00 00
00C0001770h	02 00 00 00 00 00 00 00 FF FF FF FF 00 00 00
00C0001780h	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00C0001790h	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00C00017A0h	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00C00017B0h	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00C00017C0h	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00C00017D0h	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00C00017E0h	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00C00017F0h	00 00 00 00 00 00 00 00 00 00 00 00 67 03g.

Notiamo nel record della directory radice “.”, la presenza degli attributi \$INDEX_ROOT (0x90), \$INDEX_ALLOCATION (0xA0) e \$BITMAP (0xB0), segnalati in verde, e nominati “\$I30”: ciò sta a indicare che l’indice risiede all’esterno del record (lo \$INDEX_ALLOCATION specifica i datarun dei cluster che lo compongono), mentre la \$BITMAP segnala quali blocchi dell’indice sono in uso.

Gli indici (tecnicamente implementati mediante strutture ad albero B+Tree) accelerano enormemente la navigazione fra le directory e l’elencazione e l’accesso ai file: tuttavia, se ne tralascia l’esame, non essendo strettamente necessari al recupero dei dati (per il quale basta individuare la \$MFT e analizzare i suoi record).

Eguale, si tralascia l’esame di tutte le altre strutture di NTFS non strettamente necessarie al fine del recupero dei dati e che ne fanno un file system robusto ed evoluto (il *journal* che registra come transazioni

le modifiche alla \$MFT e ad altri file di sistema; il giornale utente; le Access Control Lists o ACL, che limitano l'accesso su singoli oggetti a utenti o gruppi di utenti; la compressione e la crittografia dei contenuti; le quote disco; i punti di montaggio; i collegamenti fra directory; e così via).

Considerazioni

Come si è visto dalla breve panoramica su NTFS, neanche il recupero manuale di un file da questo file system (a patto che il file non sia compresso o cifrato) è cosa impossibile, sebbene molto più laboriosa rispetto al sistema FAT (causa la presenza dei datarun da decodificare).

Una volta trovato il record di un file (meglio se quello della \$MFT...), basta individuare gli attributi 0x30 e 0x80 e, a partire dalla decodifica di quest'ultimo, individuare i segmenti componenti il contenuto.

Un evidente vantaggio concettuale di NTFS è che, in esso, tutto è un file, e tutto in un file è un attributo. Lo spreco di tempo e di spazio necessario a registrare l'intestazione di un file (record MFT da 1 KiB, voce di indice, *journal*) è compensato dall'incremento di sicurezza: per perdere irrimediabilmente decine di file in un file system FAT, basta il guasto di qualche settore nell'area della tabella FAT, guasto che, riportato alla \$MFT, si tradurrebbe in un numero notevolmente minore di record danneggiati¹⁷.

¹⁷ Il *journal* contenuto nel file di sistema \$LogFile permette inoltre di replicare certe operazioni di aggiornamento dei record, aumentando la probabilità di ricostruirli in caso di guasto del disco o di interruzione del sistema.