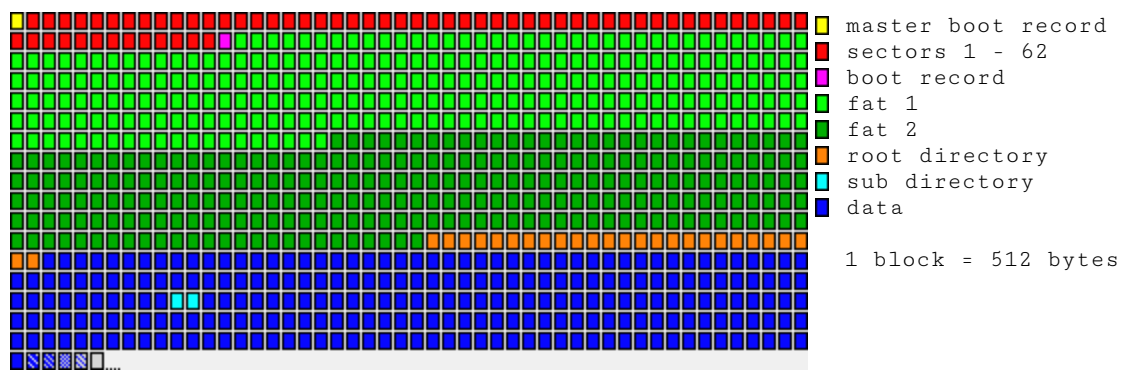


forewords

this article is about fat16 with additional comments regarding windows 98. certain things will appear/handle differently in other oses.

special thanks go to jon fox who helped me out on calculating the volume serial number, the lfn checksum and making some worthy comments while in draft form, i really appreciate it.

overview



this is a scale diagram of the start of the disk with a small fat16 partition at the beginning. this partition is just over 20mb in size, to save space not all the data/subdir/surplus area is shown - there would be 44,815 blocks or ~896 lines if this part was shown completely.

master boot record diagram

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
33	C0	8E	D0	BC	00	7C	FB	50	07	50	1F	FC	BE	1B	7C	3A+D4. uP.P.u4.
BF	1B	06	50	57	B9	E5	01	F3	A4	CB	BE	BE	07	B1	04	z. .PW^a. o^E44. ±.
38	2C	7C	09	75	15	83	C6	10	E2	F5	CD	18	8B	14	8B	8. . u. fE. aóÍ. <. <
EE	83	C6	10	49	74	16	38	2C	74	F6	BE	10	07	4E	AC	ifE. It. 8. tó4. .N-
3C	00	74	FA	BB	07	00	B4	0E	CD	10	EB	F2	89	46	25	<. tú>. . . Í. eóZF%
96	8A	46	04	B4	06	3C	0E	74	11	B4	0B	3C	0C	74	05	-SF. . <. t. . <. t.
3A	C4	75	2B	40	C6	46	25	06	75	24	BB	AA	55	50	B4	. Äu+@EF%. u\$>>^UP
41	CD	13	58	72	16	81	FB	55	AA	75	10	F6	C1	01	74	Aí.Xr. +úU^u. óÁ.t
0B	8A	E0	88	56	24	C7	06	A1	06	EB	1E	88	66	04	BF	. Sã^V\$Ç. i. é. ^f. é
0A	00	B8	01	02	8B	DC	33	C9	83	FF	05	7F	03	8B	4E <U3Éfy. +. <N
25	03	4E	02	CD	13	72	29	BE	46	07	81	3E	FE	7D	55	% . N. Í. r)4F. +> }U
AA	74	5A	83	EF	05	7F	DA	85	F6	75	83	BE	27	07	EB	ãtZfi. +Ü. ouf%'. é
8A	98	91	52	99	03	46	08	13	56	0A	E8	12	00	5A	EB	S^R^M. F. . V. è. . Ze
D5	4F	74	E4	33	C0	CD	13	EB	B8	00	00	00	00	00	00	ÖOta3Aí. é.
56	33	F6	56	56	52	50	06	53	51	BE	10	00	56	8B	F4	V3óVVRP.SQ4. . V<ó
50	52	B8	00	42	8A	56	24	CD	13	5A	58	8D	64	10	72	PR. . B\$V\$Í.ZX+d.r
0A	40	75	01	42	80	C7	02	E2	F7	F8	5E	C3	EB	74	49	. @u. B€Ç. á-ó^ÄetI
6E	76	61	6C	69	64	20	70	61	72	74	69	74	69	6F	6E	nvalid partition
20	74	61	62	6C	65	00	45	72	72	6F	72	20	6C	6F	61	table>Error loa
64	69	6E	67	20	6F	70	65	72	61	74	69	6E	67	20	73	ding operating s
79	73	74	65	6D	00	4D	69	73	73	69	6E	67	20	6F	70	ystem.Missing op
65	72	61	74	69	6E	67	20	73	79	73	74	65	6D	00	00	erating system..
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	8B	FC	1E	57	8B	F5	CB	00	00	00	00	00	00	. . . <ú.W<óE.
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
01	00	06	EF	3F	2F	3F	00	00	00	C1	12	0B	00	80	01 e.
FF	FF	0C	EF	FF	FF	40	03	07	01	F0	AE	24	00	00	00 i
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00 U^

- -> executable code
- -> boot indicator
- -> starting head
- -> starting sector & cylinder
- -> partiton type
- -> ending head
- -> ending sector & cylinder
- -> the starting sector
- -> no. of sectors in partition
- -> executable signature

1 partition entry is as follows:

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
														80	01 e.
01	00	06	EF	3F	2F	3F	00	00	00	C1	12	0B	00		 i

and reads as:

80	01	01	00	06	EF	3F	2F	3F	00	00	00	C1	12	0B	00
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	-----------

mbr in general (=512 bytes)

the master boot record (mbr) is located in the first data sector on the hard disk. the mbr can have upto 4 primary partitions, each entry occupiees 16 bytes. if the slot is unused, the space is filled with 00. each entry sets out the addressing parameters for the partition. hard disks have grown considerably in size since they were first invented, this has caused some strange addressing techniques to be used to ensure compatibility. hard drives used to be addressed by cylinder, head and sector (chs). as cylinders increased, instead of addressing it literally, the addressing values became virtual.

example: 16 heads are specified on the drive and in bios, but when opened up there were only 6 physical heads. this is converted by the hard disks controller card to real values. however virtual chs addressing is also limited, when these values are exceeded, logical block addressing (lba) is used by partitions. lba is achieved by addressing each sector on the disk by number, the first sector is 0.

executable code (=446 bytes) ■

this is the first piece of software that runs when a computer loads up. it can be less than 446 bytes; the remaining bytes will be filled with 00. the bios checks for the executable signature, if absent a hardware error is displayed, else the code is run. it looks for and loads partition parameters in the four areas and checks that only one is bootable. it then checks the active partition's boot record for the executable signature before running it. if this is absent the error: missing operating system will appear.

boot indicator (=1 byte) ■

determines which partition to boot from. 00 = inactive and 80 = active
there must only be one partition marked as 80. if more than one are marked the system will halt with the error: invalid partition table. if none are marked the system will display a hardware error like: operating system not found; this will vary from machine.

starting head (=1 byte) ■

states the starting head for the partition in hex.
eg, 01 = head 1; and another example: 3f = head 63
if these values are incorrect the system will halt with: error loading operating system. the maximum number of heads is 256. if the partition is in lba mode the head will be one less than the maximum.

starting sector/cylinder (=2 bytes) ■

states the sector and cylinder on which the partition begins. though it looks like the first byte states the sector and the second the cylinder in hex, this is not the case.
example: 62 sectors and 690 cylinders; from the disk read: be,b2
first convert to binary: be,b2 -> 1011 1110,1011 0010

the last 6 bits of the first byte: 11 1110 = 3e -> decimal = 62 sectors
6 bits gives a potential maximum of 63 sectors, 1 - 63 inclusively.

join the first 2 bits of the first byte with the whole second byte:
10 1011 0010 = 2b2 -> decimal = 690 cylinders
10 bits gives a potential maximum of 1024 cylinders, 0-1023 inclusively.
if these values are incorrect the system will halt with: error loading operating system.


partition type (=1 byte) ■

states the type of partition, there are others, but for different systems.
the maximum number of heads = 256, the maximum number of sectors is 63 and the maximum number of cylinder is 1024. hard disk are formatted with 512 byte sectors. therefore the maximum partition is $1024 * 256 * 63 * 512 = 8,455,716,864$. anything addressed over this will be a lba partition.


	fat16	fat32	extended	fat16lba	fat32lba	extended lba
visible	06	0b	05	0e	0c	0f

hidden 16 1b 15 1e 1c 1f


if a partition is marked as active and has a operating system installed and the partition is marked as hidden the system will load the io.sys and then prompt: type the name of the command interpreter (eg., c:\windows\command.com) a>_

ending head (=1 byte) 


as the starting head, but ending.

ending sector/cylinder (=2 bytes) 


as the starting sector/cylinder, but ending.

the starting sector (=4 bytes) 

this is the logical sector jump to the partition. this sector contains the partition's boot record. from the disk:
40,03,07,01 flip -> 01,07,03,40 convert to decimal = sector
17,236,800

sectors in the partition (=4 bytes) 

states the number of sectors in the partition, the last sector will be one less than the total as the first sector is sector 0. from the disk: f0,ae,24,00 flip -> 00,24,ae,fo convert to decimal = 2,404,080 sectors

executable signature (=2 bytes) 

these two bytes indicate that the sector is executable, the system will check for them. if incorrect a hardware error will be displayed. example; on a toshiba satellite:
insert system disk in drive.
press any key when ready....
and on a hi-grade ultinote: operating system not found; if they are present the code will run.

sectors 1 - 62 (=31,744 bytes)

sectors 1 - 62 inclusively are normally left empty. applications that do use it include: multi boot loaders like ranish advanced boot manager. security programs such as reflex-magnetics disknet. viruses that copy themselves to the master boot record so that they can load every time, sometimes move the real mbr into this area, plus any more virus code. full disk encryption programs and disk translation software for very large hard disks may also reside here.

boot record diagram

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
EB	3E	90	4D	53	57	49	4E	34	2E	31	00	02	04	01	00	è>+MSWIN4.1...
02	00	02	00	00	F8	00	01	3F	00	F0	00	60	75	0C	00	...z.?è.'u...
10	EC	03	00	80	00	29	1E	3C	D9	19	4E	4F	20	4E	41	.i..è.).<Û.NO NA
4D	45	20	20	20	20	46	41	54	31	36	20	20	20	F1	7D	ME FAT16
FA	33	C9	8E	D1	BC	FC	7B	16	07	BD	78	00	C5	76	00	ú3É+N&u{...&x.Áv.
1E	56	16	55	BF	22	05	89	7E	00	89	4E	02	B1	0B	FC	.V.U¿".Z~.ZN.±.ü
F3	A4	06	1F	BD	00	7C	C6	45	FE	0F	8B	46	18	88	45	ó&...& .ÆEþ.<F.ÊE
F9	FB	38	66	24	7C	04	CD	13	72	3C	8A	46	10	98	F7	úú8f\$.í.r<SF."-
66	16	03	46	1C	13	56	1E	03	46	0E	13	D1	50	52	89	f..F..V..F..NPR%
46	FC	89	56	FE	B8	20	00	8B	76	11	F7	E6	8B	5E	0B	Fü&Vp, ..<v.±æ<^.
03	C3	48	F7	F3	01	46	FC	11	4E	FE	5A	58	BB	00	07	.ÄH-ó.Fü.NpZX>...
8B	FB	B1	01	E8	94	00	72	47	38	2D	74	19	B1	0B	56	<ú±.è".rG8-t.±.V
8B	76	3E	F3	A6	5E	74	4A	4E	74	0B	03	F9	83	C7	15	<v>ó ^tJNt...ùfÇ.
3B	FB	72	E5	EB	D7	2B	C9	B8	D8	7D	87	46	3E	3C	D8	;úráèx+E.Ø}±F><Ø
75	99	BE	80	7D	AC	98	03	F0	AC	84	C0	74	17	3C	FF	u"m&€}-".è~.Ät.<ý
74	09	B4	0E	BB	07	00	CD	10	EB	EE	BE	83	7D	EB	E5	t.'...>..í.èi&f}èâ
BE	81	7D	EB	E0	33	C0	CD	16	5E	1F	8F	04	8F	44	02	&+}èâ3Áí.^.+.+D.
CD	19	BE	82	7D	8B	7D	0F	83	FF	02	72	C8	8B	C7	48	í.& .}<}.fý.rÈ<ÇH
48	8A	4E	0D	F7	E1	03	46	FC	13	56	FE	BB	00	07	53	H&N..+á.Fü.Vp>...S
B1	04	E8	16	00	5B	72	C8	81	3F	4D	5A	75	A7	81	BF	±.è..[rÈ+?MZU\$+¿
00	02	42	4A	75	9F	EA	00	02	70	00	50	52	51	91	92	..BJuÿè...p.PRQ''
33	D2	F7	76	18	91	F7	76	18	42	87	CA	F7	76	1A	8A	30±v.'±v.B±E±v.Š
F2	8A	56	24	8A	E8	D0	CC	D0	CC	0A	CC	B8	01	02	CD	óŠV\$ŠèDìDì.ì...í
13	59	5A	58	72	09	40	75	01	42	03	5E	0B	E2	CC	C3	.YZXr.@u.B.^.âíÄ
03	18	01	27	0D	0A	49	6E	76	61	6C	69	64	20	73	79	...'.Invalid sy
73	74	65	6D	20	64	69	73	6B	FF	0D	0A	44	69	73	6B	stem diský...Disk
20	49	2F	4F	20	65	72	72	6F	72	FF	0D	0A	52	65	70	I/O errorý...Rep
6C	61	63	65	20	74	68	65	20	64	69	73	6B	2C	20	61	lace the disk, a
6E	64	20	74	68	65	6E	20	70	72	65	73	73	20	61	6E	nd then press an
79	20	6B	65	79	0D	0A	00	49	4F	20	20	20	20	20	20	y key...IO
53	59	53	4D	53	44	4F	53	20	20	20	53	59	53	80	01	SYSMSDOS SYSE.
00	57	49	4E	42	4F	4F	54	20	53	59	53	00	00	55	AA	WINBOOT SYS. U²

- -> jumpcode
- -> oem/id name
- -> bytes per sector
- -> sectors per allocation
- -> reserved sectors
- -> no. of fats
- -> root entries
- -> total sectors(16)
- -> media type
- -> sectors per fat
- -> sectors per track
- -> no. of heads
- -> hidden sectors
- -> total sectors(32)
- -> drive id
- -> nt reserved
- -> extended boot signature
- -> volume serial number
- -> volume/partition name
- -> fat type
- -> executable boot(strap) code
- -> executable signature


jumpcode (=3 bytes) ■

the offset jump to the boot(strap) executable code plus a nop. from the disk: eb,3e,90 -> translates to: |jumpshort(to)|offset 3e|no operation|


oem/id name (=8 bytes) ■

some indication to what system formatted the partition, not checked,

but set for compatibility. mswin4.0 and mswin4.1 as formatted by windows 95 and 98 respectfully.

bytes per sector (=2 bytes) 


normally set to 512 bytes; from the disk: 00,02 flip -> 02,00 convert to decimal = 512 bytes. 1024, 2048 and 4096 are also valid, but are not generally used.

sectors per cluster (=1 byte) 

states the number of sectors per cluster. this will vary depending on the size of the partition. example: 10 convert to decimal = 16 sectors
for hard disks:

size ranges (mb)	fat type	no. sectors	cluster size
0 - 15	12	8	4,096 bytes
16 - 127	16	4	2,048 bytes
128 - 255	16	8	4,096 bytes
256 - 511	16	16	8,192 bytes
512 - 1023	16	32	16,384 bytes
1024 - 2047	16	64	32,768 bytes

note that in this table the cluster size is calculated with a sector size of 512, which is most common. cluster sizes should not exceed 32,768 bytes

resevered sectors (=2 bytes) 


states the number of reserved sectors. on fat12/16: 01,00 flip -> 00,01 convert to decimal = 1 sector - the boot record is this.

no. of fats (=1 byte) 


states the number fats used. normally set to 2 in case of bad sectors, which could lead to data errors. however >=1 is also valid - unsure of maximum.

root entries (=2 bytes) 

states the maximum number of 32byte entries in the root directory; unused for fat32 and set to 00,00; however for fat16: 00,02 flip -> 02,00 convert to decimal = 512; 512 * 32(bytes) = 16384 bytes - data is stored after this point.

total sectors(16) (=2 bytes) 


set if the partiton is less than 33,554,432 bytes (32mb) in size - mainly for a floppy disk: 40,0b flip -> 0b,40 convert to decimal = 2880 sectors; fat16 partitions may also use this entry. if number of sectors is above this will be set to 00,00

media type (=1 byte) 


states the physical media type; f8 = hard drive; f0 = (standard) floppy drive

sectors per fat (=2 bytes) 


number of sectors in one fat; for a floppy: 09,00 flip -> 00,09 covert to decimal = 9 * 512(bytes per sector) = 4608; there are two copies of the fat thus 4608 * 2 = 9,216 bytes - the jump to the root directory.

sectors per track (=2 bytes) 


states the number of sectors per track.

no. of heads (=2 bytes) 


number of disk drive heads; eg on a floppy disk: 02,00 flip -> 02,00
convert to decimal = 2 heads. on a hard disk this will be much more;
eg f0,00 flip -> 00,f0 convert to decimal = 240 heads

hidden sectors (=4 bytes) 


this will matchup with the starting sector stated in the partitions'
entry in the mbr. it states the number of hidden sectors from the
beginning of the drive to the boot record of the partition. normally
set to 3f,00,00,00 flip -> 00,00,00,3f convert to decimal = 63, for a
primary partition. note that sector 63 will contain the boot record,
as sector numbers start at zero. another example from disk:
40,03,07,01 flip -> 01,07,03,40 convert to decimal = 17,236,800

total sectors(32) (=4 bytes) 


no of sectors in the partition eg, 10,ec,03,00 flip -> 00,03,ec,10
convert to decimal = 257040 sectors (normally 512 bytes) this entry
is used if total sectors(16) is set to 00,00

drive id (=1 byte) 


set to 00 for floppy disks and 80 for hard disks. also refered to as
the logical drive number.

nt reserved (=1 byte) 

set at format to 00 and not checked thereafter.

extended boot signature (=1 byte) 

set to 29 indicating that the serial, label and type data is present.

volume serial number (=4 bytes) 

when a partition is formatted; quick or full, it will display the
newly assigned serial such as: 15e7-2a35. this is written in reverse
on the disk as: 35,2a,e7,15. calculated by combining the date and
time at the point of format, it is an unique identifier to keep track
of drives in use. it is not possible to retrieve the date and time
from the serial number.

time: 2:22:32.50p | date: (oct)10-03-2001 | serial on disk:
35,2a,e7,15

first byte is calculated as follows:
milliseconds + days -> convert to hex
50 + 3 = 53 -> = 35


the second byte is calculated as follows:
months + seconds -> convert to hex
10 + 32 = 42 -> = 2a

last two bytes are calculated as follows:
(hours [if pm + 12] * 256) + minutes + years -> convert to hex & flip
(2 + 12 = 14 * 256 = 3584) + 22 + 2001 = 5607 -> 15,e7 -> e7,15

volume/partition name (=11 bytes) 

the volume label can be up to 11 characters. it also has to be
referenced in the root directory as a 32 byte entry with the volume








storage errors when disks were less reliable. the smallest size fat16 seems to be 256*512bytes

media type (=2 bytes) 

the first 2 bytes of fat16 state the media type; f8 = hard disk; floppy disks are f0; this value must match up with the media type stated in the boot record.

partition state (=2 bytes) 


















set to ff,ff at format and at shut down; it means that the partition is "clean" not mounted or hasn't been written to. browsing the directories, getting properties and opening files will not mount the disk; even if the last accessed date is updated, the disk will not be mounted. however if data or an entry is rename or saved these two bytes will change to ff,f7 indicating that the partition is "dirty" mounted or in use. if a system starting up finds that the partition is still mounted, it will know that the system did not shut down properly and may run scandisk. this only applies to hard disks. jon fox informs me that this section is also used to indicate if an hardware error occurred - i was unable to verify this ;-)

fat attributes (varies in size)       

data clusters starts at cluster 2 right after the root directory. on fat16 one cluster represented by 2 bytes in the fat. 2bytes->16bits->fat16 - fat32 uses 4 bytes. each 2 bytes in the fat mark a position in the data area of the partition. if the cluster is part of chain of clusters; occupying more than one cluster, they will be numbered in hex order. the numbers range from 03,00 to 99,ff; adding from the left. if the cluster is the last in the chain or if the entry only takes up one cluster it will be marked as ff,ff. if a file is resaved and it exceeds its allocated cluster, the file or directory list will be split onto two, and the extra clusters located elsewhere on the drive. the fat will be changed so that the last cluster in the first chain will point to the start of the next cluster chain. in the diagram above the first cluster points to cluster 6, (though labeled as 07,00 it is cluster 6). this is known as fragmenting, and can slow the system down a little as these pieces need to be reassembled before use. defragmenting will move the data around so that all the chains of clusters are grouped together. if there is no data stored or data has been deleted the fat will be marked as 00,00. sometimes the system cannot keep up with this; if a 20 cluster entry is deleted and then a new 5 cluster one created the data maybe located after the blank entries of the deleted 20 cluster one. a reboot maybe required for the area to be acknowledged. if the fat entries do not completely fill the last sector of the fat the remaining space will be left empty and scandisk will not check it. bad clusters are marked with f7,ff. data will not be written to these areas and a thorough scandisk will not recover these clusters, only report that they are there. data can be hidden in phoney bad clusters, such program needs to keep track of where the data is stored and replace certain fat entries and the data "disappears"

data entry diagram


0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
43	2E	00	64	00	61	00	74	00	00	00	0F	00	98	FF	FF	C . d a t . . . j y y
FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	00	00	FF	FF	FF	FF	y y y y y y y y y y . . y y y y
02	6E	00	6F	00	70	00	71	00	72	00	0F	00	98	73	00	h o p q r . . . s
74	00	75	00	76	00	77	00	78	00	00	00	79	00	7A	00	t u v w x . . . y z .
01	61	00	62	00	63	00	64	00	65	00	0F	00	98	66	00	a b c d e . . . f
67	00	68	00	69	00	6A	00	6B	00	00	00	6C	00	6D	00	g h i j k . . . l m .
41	42	43	44	45	46	7E	31	44	41	54	24	00	64	12	13	A B C D E F ~ 1 D A T \$. . d .
64	27	37	2B	00	00	9D	BA	35	2B	17	00	00	00	10	00	d ' 7 + . + * 5 + . . .

	-> ordinal field		-> create time
	-> unicode long file name		-> create date
	-> longfile name attribute		-> access date
	-> reserved for future use		-> access time
	-> dos name check sum		-> modified time
	-> lfn data location		-> modified date
	-> dos file name		-> data location
	-> entry attributes		-> data length
	-> nt reserved		

entries in general

entries are made up of 32 bytes. if the entry is deleted the first byte is changed to e5; and the sectors are marked as free. basic computer forensics can reveal alot because of this. if this is a risk, check http://www.fortunecity.com/skyscraper/true/882/Comparison_Shredders.htm for prevention programs. defragging the disk will compress the directory entries and clear any redundant ones; it will also rearrange the data so that it is grouped together, but old data will still be there. if the first byte is 00, entries are considered to have ended for that directory.

be aware of that


ordinal fields (=1 byte - every 32 bytes of the lfn) 

the first byte of each 32 byte string of the long file name is called the ordinal field. it tells the system which order the entries are to be read in. this is in hex and read from 01. the first byte of the _entire_ entry however is different, it defines the length of the total lfn entry.

from the root dir:

letter	the long file name's...	no. of 16 byte lines/size
a/41	length is >=001 & <=013	02 lines 032 bytes/020
b/42	length is >=014 & <=026	04 lines 064 bytes/040
c/43	length is >=027 & <=039	06 lines 096 bytes/060
d/44	length is >=040 & <=052	08 lines 128 bytes/080
e/45	length is >=053 & <=065	10 lines 160 bytes/0a0
f/46	length is >=066 & <=078	12 lines 192 bytes/0c0
g/47	length is >=079 & <=091	14 lines 224 bytes/0e0
h/48	length is >=092 & <=104	16 lines 256 bytes/100
i/49	length is >=105 & <=117	18 lines 288 bytes/120
j/4a	length is >=118 & <=130	20 lines 320 bytes/140
k/4b	length is >=131 & <=143	22 lines 352 bytes/160
l/4c	length is >=144 & <=156	24 lines 384 bytes/180
m/4d	length is >=157 & <=169	26 lines 416 bytes/1a0
n/4e	length is >=170 & <=182	28 lines 448 bytes/1c0
o/4f	length is >=183 & <=195	30 lines 480 bytes/1e0
p/50	length is >=196 & <=208	32 lines 512 bytes/200
q/51	length is >=209 & <=221	34 lines 544 bytes/220
r/52	length is >=222 & <=234	36 lines 576 bytes/240
s/53	length is >=235 & <=247	38 lines 608 bytes/260
t/54	length is >=248 & <=250	40 lines 640 bytes/280

the 32 byte blocks are positioned in reverse order with the last section at the beginning of the entire entry.

the long file name 

the name is stored in unicode format. a double byte character system designed to handle all possible foreign and scientific characters. the



original ascii characters are the same except they are two bytes in size, the second byte is a null; 00. check <http://www.unicode.org> for a more in depth explanation. there can be up to 13 unicode characters per 32 byte section. if however the long file name does not fill the slot exactly a unicode null (00,00) will be added to the end, followed by ff,ff's until the section is filled. note that entries do not have to have a lfn.

as you can see from the table above a long file name in the root directory can be between 1 and 250 characters. this is a little odd as there is still space for 10(?) more characters, also stated by microsoft that files could have up to 255 characters. total path length; tpl referred to in this text is the character length from the drive letter to the folder or file, eg: "c:\new folder" has a tpl of 13. "c:\new folder\test.dat" has a tpl of 22. a large tpl can have strange effects on explorer. a single folder in c:\ can have a tpl of 253 (250 + c:\) it can only contain files <= 4 characters in length. however if there are two folders the tpl can be up to 255, with 2 remaining for files. the maximum valid tpl for 98f16 including a file is 258. note that this value can change a little, eg one character folders; c:\a\a\a\...\etc (122 folders; maximum) can have a tpl length of 254, including 7 for a file; and even this can change, if you paste a file in it can be up to 12 characters for a file. why this varies is unclear. explorer checks the tpl on entry creation to see that it does not exceed an illegal value, however it only checks it up to that point. thus two folders could be created, and the first one renamed to a longer length so that when the subfolder is accessed the tpl is greater than allowed. while a tpl of such length will have strange effects on the system eg, inaccessible, unviewable, unmoveable, undeleteable, false properties of files and folders; scandisk will fix them by either renaming the affected files or folders and/or moving them to the root directory. one combination is of note. create a folder in the root directory with a name of 1 character, save a file of 5 characters in length into the folder, rename the folder so it has 250 characters in the name. tpl is 259. properties are correct for the file and it can be opened and resaved to the same location, however it cannot be moved out of that folder nor renamed or deleted, to do so the folder name must be shortened. this doesn't stop the file being opened and resaved to a different location. scandisk doesn't seem to mind this combination.

not all ascii (unicode) characters can be used in long file names. names cannot contain the following characters: \/:*?"<>| if the name begins with a space it will be removed, if a fullstop - prompt for a valid name. if the name ends with a fullstop or spaces they will be removed. letters from the extended character set can cause some confusion of explorer in 9x/me systems. the following ascii characters inclusively are of note in 9x: 158, 169, 176->224, 226->229, 231->240, 242->245, 247, 249, 251->252 and 254->255; 75 characters. in me: 176->180, 185->188, 191->197, 200->206, 213, 217->220, 223, 242 and 254; 31 characters. if a folder contains any of these characters you cannot access it via explorer; it errors with "the folder 'path' doesn't exist" twice. properties are displayed incorrectly. copying and deletion attempts result in "cannot delete file file system error (1026)." explorer will prevent the entering of these characters by turning them in the underscore. they can be entered in a dosbox. check numlock is on; normally by default, then press alt+type the number on the keypad - laptops normally alt+fn+no. if a file has these characters in, when opened it will prompt "cannot find the 'path\file' do you want to create a new file?" clicking no creates an untitled blank document. clicking yes creates a new blank file, but with the illegal character an underscore in the name - resave it. now, and this is quite interesting; when opening or deleting the illegal file, it will open/delete the one with the underscore in. this means that explorer, on finding an entry with a character in that it cannot handle, scans the

current directory for one of the same name, but with an underscore in. folders can also be "redirected" this way and the underscore named entry will be opened with internet explorer. scandisk doesn't seem to mind this.


some whole words are also a problem. you cannot create directories or files of the following names: aux, con, nul, prn, com1, com2, com3, com4, lpt1, lpt2, lpt3, lpt4, lpt5, lpt6, lpt7, lpt8, lpt9, clock\$ and config\$ in total, 19 words. it will error with either: unable to create directory or cannot rename the filename: access is denied. it seems that these values, reserved hardware device/port names, may already be in use by the operating system - they are mentioned in the io.sys. attempts to access a folder with a subfolder of both the same reserved name will crash/freeze the system for: aux\aux, con\con, nul\nul, clock\$\clock\$ and config\$\config\$. the others will result in the error: invalid directory. you cannot create files or folders in windows of these reserved words, but you can change them with a hexeditor on the disk, most result in either errors, instant freezing/crashing of the system and nondeletable, noncopiable or inaccessible entries in explorer. a lot of these entries' properties are reported as 112bytes. scandisk will rename most reserved names with a hyphen on the end. however scandisk does seem to mind the following names: lpt4 to lpt9. the names have to be in uppercase or scandisk will error and fix to uppercase. the folder 'path' does not exist error appears twice on attempted access; files cannot be opened. cannot delete file: file system error (1026). properties report the folder as a file of zero bytes. cannot copy file: file system error (1026) note that these may vary as hardware configurations may be different.

lfn attributes and location  
(= 1 byte and 2 bytes respectfully; every 32 byte section of the lfn)

entire entries are read in as 32 byte blocks. to ensure compatibility with older systems, when they introduced long file names, each section of a long file name had to "pretend" to be an entire entry. to do this the entry attribute and data location are set to the "impossible values" of 0f and 00,00 respectfully. each 32 byte section of the long file name is required to have this slotted in.

lfn reserved (=1 byte - every 32 bytes of the lfn) 

set to 00 at entry creation. it is reserved for future use; however copying or renaming a file/folder resets it to 00. resaving a file retains any changed values.

dos name checksum (=1 byte - every 32 bytes of the lfn) 

this is for checking that the dos name and the long file name match up correctly. if this value is incorrect, the lfn will be discarded and the dos name substituted.

the steps are:

1. take the ascii value of the first character. this is your first sum
2. rotate all the bits of the sum rightward by one bit
3. add the ascii value of the next character with the value from above. this is
your next sum
4. repeat steps 2 through 3 until you are all through with the 11 characters in
the 8.3 filename

note that spaces (ascii value 20) are also included.

to calculate manually use calc.exe; view scientific, switch to hex mode

and select the byte radiobutton under the display.

dosname: UNTITLEDTXT - lfn checksum: 44

in this section "->" means rotate all the bits, one bit to the right.

U = 55 = 0101 0101 -> 1010 1010 = AA
N = 4E + AA = F8 = 1111 1000 -> 0111 1100 = 7C
T = 54 + 7C = D0 = 1101 0000 -> 0110 1000 = 68
I = 49 + 68 = B1 = 1011 0001 -> 1101 1000 = D8
T = 54 + D8 = 2C = 0010 1100 -> 0001 0110 = 16
L = 4C + 16 = 62 = 0110 0010 -> 0011 0001 = 31
E = 45 + 31 = 76 = 0111 0110 -> 0011 1011 = 3b
D = 44 + 3b = 7f = 0111 1111 -> 1011 1111 = bf
T = 54 + bf = 13 = 0001 0011 -> 1000 1001 = 89
X = 58 + 89 = e1 = 1110 0001 -> 1111 0000 = f0
T = 54 + f0 = 44

dos file name (=11 bytes)

dos names can be ≥ 1 and ≤ 11 characters. any remaining bytes will be filled with spaces. if the entry also has a long file name attached the dos name is changed slightly. the name is shortened to 6 characters and a tilde~ and number are added to the end, giving a total of 8 characters. if the first 6 letters are different from all other dos names in that directory then the ending is ~1 however if they are the same the ending number will be different. the system scans the dos names in that directory for the lowest available ending number to use, incrementing by 1 (decimal) as needed. when ~10 is reached the dos name is reduced to 5 characters to make room. if a file is copied to a different directory the dos name and long file name checksum may change, as the system checks both dos and long file names already in use. process referred to in this text as tilded. the last three letters of a dos name are considered the file extension and will be divided off from the rest by a space or a fullstop. all letters are converted to uppercase. if an entry has its dos name converted to lowercase it can cause problems. while files handle okay, attempts to open a folders results in the error: c:\folder is not accessible. this folder was moved or removed. in a dosbox the result is path not found and the volume serial number and name are displayed, it can however be deleted, in dos or explorer. scandisk will fix this by converting back to uppercase. if the name contains spaces, they will be removed and the name tilded. if the name contains a fullstop, upto the first three letters thereafter are considered the extension. if the extension is more than three letters, just the first three will be used plus the name is tilded. if the name contains more than one fullstop the last one is used as the divide between the name and extension. all other fullstops are removed and the name is tilded.

lfn names cannot contain the following characters: \/:*?"<>| some can be attempted to be used in dos, but they have strange effects. tested in a dosbox; file creator is "copy con filename" typed text "ctrl + z", "enter"

\ creation: path not found; rename: invalid parameter

/ creation: invalid switch; rename: anything after and including the forward slash will be removed

: creation: to many parameters; rename: file not found

* creation: anything after and including the asterisk; but not the extension will be removed. * is a wildcard

? creation: the question mark is removed; rename: either the question mark will be substituted for an other letter or it will be removed; how it substitutes is unclear. ? like * is a wildcard

" creation: the double quotation mark is removed; rename: as creation

< creation: file not found; rename: as creation

> creation: splits into two different file at the greater-than sign. the first half contains the text, while the second contains the

following text: " 1 file(s) copied" - the confirmation that appears after the text is saved; rename: as creation, but the second half contains nothing as there isn't a confirmation on rename. the > character pushes screen output to "somewhere" as shown in the example above a file. >> appends to "wherever"

| creation: anything after and including the vertical bar will be removed, also on save the message bad command or file name appears; rename: as creation. the | can be used to "pipe" instructions eg: echo |del *.* >nul will quietly delete all files in the current directory (hidden, readonly and system attribute file are not deleted)

entry attributes (=1 byte) 

documented attribute values

none: 00
archive: 20
read-only: 01
system: 04
hidden: 02
directory: 10
volume: 28 (hdd)

if attribute = 00 then its treated as a file.


for archive, hidden and system attributes; add the values together: 26

archive and directory *can* also be unofficially stated as other values. read in the first nybble and convert it to decimal. if its divisible by 2 its an archive, else its a folder.


if the second nybble is >=8 and <=e the entry disappears and becomes the volume name. the volume attribute should only be stated once on the partition, offset 15h. sometimes there will be an entry already in root that will override this. this is valid if the entry has no data attached, else scandisk will delete and recover any clusters. n.b: volumes are archives; if stating it in the root (this will only work in root) so f8 is invalid here. there can only be one valid entry in the root that can have the volume attribute.

if the second nybble is =f the entry disappears completely, however its still a volume. scandisk will error if there is data attached, and delete any data.

volumes can labelled <=11 characters from explorer, however you can increase the length by saving an entry in the root directory and changing its attributes to a volume, this can have some interesting effects. volume names >=12 and <=32 will prevent renaming of the label via explorer even though the text field will be editable. error: access denied. >=33 the label will be greyed out and uneditable. volume names of >=78 will cause illegal operation errors and not enough available memory errors on attempted format in explorer. volume names of >=83 will cause scandisk to crash in a similar way, and reboot maybe needed even if the volume attribute is removed. all values above these have not been tested, but are believed to be the same. partitions can still be formatted in dos/dosbox.


ntreserved (=1 byte) 

set to 00 at entry creation and never modified or checked thereafter. reserved for windows nt. copying the file resets the attribute, but renaming doesn't.

create time (=3 bytes) 


minimum time: 00:00:00 = 00 00 00 if zero, no time in properties

maximum time: 23:59:59 = 64 7D BF
how it calculates it:
the values on the disk are: 64,90,a6 and properties: 8:52:33pm
first flip the bytes: 64,90,a6 -> a6,90,64
convert to binary: a6,90,64 -> 10100110,10010000,01100100
divide up the sections and convert back to decimal: (from left to right)
(5bits;hour) 10100 -> = 20hrs (24hr) or - 12 = 8pm
(6bits;mins) 110100 -> = 52 minutes
(5bits;secs) 10000 -> 16; * 2 = 32 seconds
(8bits;mili) 01100100 -> = 100 milliseconds
if millisecs >=000 and <=099; seconds is correct + no. of milliseconds
if millisecs >=100 and <=199; seconds + 1 = seconds + no. of milliseconds
if millisecs >=200; seconds + 2 = seconds (etc) but its considered invalid
which gives 33 seconds exactly: 8:52:33pm
invalid times such as 2600hours simply roll fowards to become 0200hours in properties, scandisk does not error.


create date (=2 bytes) 

minimum date: 1/1/1980 = 21 00
maximum date: 2/7/2106 = 46 FC
how it calculates it:
the values on the disk are: 14,2b and properties: 20th august 2001
first flip the bytes: 14,2b -> 2b,14
convert to binary: 2b,14 -> 00101011,00010100
divide up the sections and convert back to decimal: (from left to right)
(7bits;yr) 0010101 -> 21; + 1980 = 2001
(4bits;mt) 1000 -> = 8 or august
(5bits;dy) 10100 -> = 20th
20th august 2001
invalid dates such as the 31st of september simply roll forwards to the 1st of october in properties, scandisk does not error.

the last possible create date and time is:
sunday, february 07, 2106 7:28:15am anything beyond = (unknown)
the create date and time seem to be handled together though they are separate entries.

access date (=2 bytes) 

minimum date: 1/1/1980 = 21 00
maximum date: 2/7/2106 = 46 FC
this date is highly changeable, just right clicking on it will reset to current date, however it can be correctly queried programatically.
this is calculated the same way as the modified date (see above)

access time (=2 bytes) 

minimum/maximum: 00 00
this is a strange entry, it has to be 00 00, if you change it manually, windows changes it back if you view the properties of the file, or if you resave it. therefore you can only get access properties to the day.

modified time (=2 bytes) 

00:00:00 = 00 00 if zero, no time in properties
23:59:58 = 24 28
this is calculated the same way as the modified time (see above) except that it does not have the same accuracy; one less byte. thus the modified time is to the nearest 2 seconds.

the last possible modified date and time is:

sunday, february 07, 2106 7:28:14am

modified date (=2 bytes)

minimum date: 1/1/1980 = 21 00

maximum date: 2/7/2106 = 46 FC

this is calculated the same way as the modified date (see above)

data location (=2 bytes)

the minimum can be 00 - 0 and the maximum ff - 65535

the correct value will point the os to the starting cluster of the data.

eg, 03 00

flip these values: 00 03

convert to decimal: 3

data starts at the beginning of the third cluster.

if an entry points to the same cluster they are said to be cross-linked, this is of note regarding folders. folder names mentioned are only for reference. create a folder in root called folder1, create a subfolder within called folder2. change the location of folder2 to 00. attempts to access folder2 result in explorer looping back to root. scandisk will error and the fix is to move/recover folder2 and its contents to the root directory. scandisk can get a little muddled if you create a subfolder within folder2 called folder3 and point the location of folder3 to folder2. the default fix is to give each file a separate copy of the shared cluster(s) if this is attempted scandisk will loop at each attempt, and any data in folder3 may not be recovered. there are, however other fix options available which will recover the data, though some directories maybe undeletable due to an exceeded total path length, just shorten the folder name.

data length (=4 bytes)

the minimum can be 00 00 00 00 - 0 and the maximum ff ff ff ff - 4,294,967,295

gb, mb, kb, by

this entry will state the length of data to read in.

eg, AC 3B 96 00

flip these values: 00 96 3B AC

convert to decimal: 9,845,676 bytes or 9.38MB

folders have a value of: 00 00 00 00

the maximum is restricted by the size of the partition. the largest partition creatable by fdisk for fat16 is 2047.31mb (2,146,765,824 bytes) entry stated as 00,80,f0,7f. the largest file creatable in this partition is 1.99gb (2,146,467,840 bytes). not all the data area is filled, 18,944 bytes of surplus sectors were left empty at the end.

subdirectories (=64 bytes)

at the beginning of all directories (not root) there will be two folder entries of total 64 bytes before the list of files or folders within that directory. they are dot and dotdot and are visible in dos. dot is the current folder and dotdot is the parent folder. each entry will have the following properties: a dos name of either dot or dotdot; attributes of a folder; the create, access, modified date and time, though only the modified date and time are required; the cluster location of the folder, dot will be the same as the current folder and dotdot will be the same as the parent folder or 00 if root; and a data length of zero. also these entries have to be at the beginning of the directory list. a three dot entry is not valid even though you can change to two folders up. you can add more entire "dot" entries without scandisk erroring, on the condition that the dos name, the directory

attribute, the cluster location matchup and the data length is zero. these extra entries will also not be visible in explorer and they do not have to be at the beginning of the directory list. folders seem to handle okay without the dot and dotdot entries and the system doesn't check the location values when changing directory.

the root directory in fat16 is a fixed at 16,384 bytes in size. in total: 512 dosname entries of 32bytes or 256, 64byte entries with a lfn of upto 13 characters. if the maximum lfn entry is used, a maximum of 24 entries can be listed in the root directory with 256 bytes remaining. at this point errors will occur if any entries are attempted to be added or lengthen. entrycopy: "cannot copy thefile: the directory or file cannot be created." entryrename: "cannot rename thefile: access is denied. make sure the disk is not full or write-protected and that the file is not currently in use."

subfolders can have many entries. when a subfolder is created, 1 cluster is set aside for entries. when this is exceeded the directory list extends into another cluster, usually one is not available right after the first section, as a result directory lists get fragmented accross the drive.

you must get permission from the respective author before reproduction