Disk Structures Overview

The base element of the BTOS file system is a 512 byte sector which is defined either by logical file address (lfa), or by a cylinder, head and sector number. One or more contiguous sectors comprise a disk extent, which is definable by its starting lfa and its length. Files consist of one or more disk extents.

A lfa is used to locate a particular sector of a file. It specifies the byte position within a file; that is, it is the number (the offset) that would be assigned to a byte in a file if all the bytes were numbered consecutively starting with zero. A lfa is 32 bits in length. Bits 0-29 of the lfa define a disk address, bit 30 can be set to suppress retries, and bit 31 can be set to override normal system checks to access defective disks.

Files are grouped into user-defined sets called directories, such that a file may belong to only one directory. A disk or volume contains at least one directory (sys), which minimally contains the files, which describe the disk. The number of directories per volume, and the number of files per directory are finite numbers set at volume or directory creation time.

Hierarchy

The root structure of the file system is the Volume Home Block (VHB), which contains pointers to the other structures of the file system. Two VHB's are allocated at initialization time, one at lfa zero and one at mid-point on the volume. The VHB contains the lfa of the Master File Directory (MFD), which defines the volume's directories. Each active entry in the MFD contains the lfa of its directory, which is a hashed table of file names and their pointers into the File Headers.

Each file is allotted at least one entry in the File Headers. This entry defines up to 32 disk extents of which the file is comprised. In the rare case that a file requires more than 32 extents, file headers are chained. The base address of the File Headers is also found in the VHB.

There exists the option to write secondary, or backup file headers, to be used in the event that the primary is unreadable. (Secondary file headers are the default option in the standard volume initialization). The secondary headers are placed after every N primary headers, where N is defined in the VHB as AlternateFileHdrPageOffset. The VHB also keeps track of the next free file header, and the total number of free file headers.

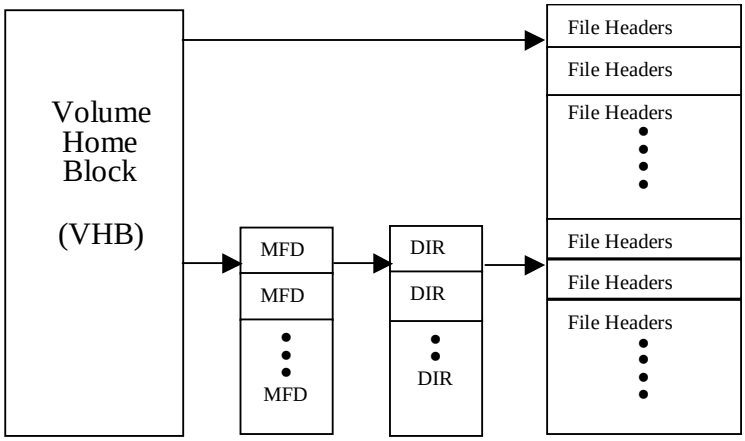

Figure 1

Figure 1 illustrates that the Volume Home Block has a pointer to the first sector of the File Headers, as well as to the first sector of the Master File Directory. Each Directory has entries for all the files in the directory and a pointer to the File Header, which describes the file. Each File Header has pointers to the extents of the file.

The other structures that comprise the file system are the Bad Block File (BadBlk.Sys), which keeps a count of bad spots by cylinder/head/sector address, and the Allocation Bit Map, which contains a bit for every disk sector. The bit is set if the sector is available.

The Volume Home Block

The VHB contains the locations and sizes of the other structures which comprise the BTOS file system as well as pointers to other system files such as the operating system (sysImage.sys), crash dump file (crashDump.sys), log file (log.sys), etc. BTOS initialization writes two Volume Home Blocks per disk, one at logical file address zero and one at a mid-point on the media.

The VHB is accessible by reading lfa zero, by calling the BTOS function GetVHB, or by accessing the pointer to the memory resident VHB found in the Device Control Block (DCB). The VHB itself has no entry in the File Headers.

Offset	Size	Field
0	2	Checksum See Appendix.
2	4	LfaSysImagebase Address of the first sector of the operating system.
6	2	CPagesSysImage Size in sectors of the operating system.
8	4	LfaBadBlkbase Address of the first sector of the bad sector file (badBlk.sys).
12	2	CPagesBadBlk Size in sectors of the bad block file .
14	4	LfaCrashDumpbase Address of the first sector of the crash dump file (crashDump.sys).
18	2	Size in sectors of the crash dump file.CPagesCrashDump
20	13	VolName Volume Name; first byte contains the count of bytes in the volume name.
33	13	VolPasssword Volume Password; first byte contains the count of bytes in the Password
46	4	LfaVHB Address of the first sector of the second (active) VHB.
50	4	LfaInitialVHB Address of the first sector of the first (backup) VHB.
54	4	CreationDT The date of initialization in System Date/Time format.
58	4	ModificationDT The date of last modification in System Date/Time format.
62	4	LfaMFDbase Address of the first sector of the Master File Directory.
66	2	CPagedMFD Size in sectors of the MFD.
68	2	LfaLogbase

Offset	Size	Field
		Address of the first sector of the system log file (log.sys).
72	2	CPagesLog Size in sectors of the system log file.
74	2	CurrentLogpage The sector offset from lfaLogbase where the current log entry is to be written.
76	2	CurrentLogbyte The byte offset within currentLogpage where the current log entry is to be written.
78	4	LfaFileHeadersbase Address of the first sector of the File Headers
82	2	CPagesFileHeaders Size in sectors of the File Headers.
84	2	AltFileHeaderPageOffset The number of file headers that separate a primary file from its secondary file header.
86	2	IFreeFileHeader The offset from lfaFileHeadersbase to the next available file header.
88	2	CFreeFileHeaders The total number of unused file headers.
90	2	ClusterFactor Not used. Contains a 1.
92	2	DefaultExtend Not used. Contains a 1.
94	2	AllocSkipCnt Not used. Contains a 1.
96	4	LfaAllocBitMapbase Address of the the first sector of the allocation bit map.
100	2	CPagesAllocBitMap Size in sectors of the allocation bit map.
102	2	LastAllocBitMapPage When combined with lastAllocWord and lastAllocBit, forms a pointer into the bit map for the next available sector.
104	2	LastAllocWord see lastAllocBitMapPage
106	2	LastAllocBit see lastAllocBitMapPage
108	4	CFreePages Total number of unallocated sectors.
112	2	IDev Offset into the array of device control blocks.
114	105	RgLruDirEntries An array of the three Last Recently Used MFDs (35 bytes each, see MFD structure).
219	2	MagicWd Used to calculate the checksums for the Volume Home Block and the File Headers. Value is 7C39.
221 222 223 225	1 1 2 2	SysImageBaseSector SysImageBaseHead SysImageBaseCylinder SysImageMaxPageCount The above fields describe for the bootstrap ROM the location and file size of the program to be bootstrapped.
227 228	1 1	BadBlkBaseSector BadBlkBaseHead

Offset	Size	Field
229 231	2 2	BadBlkBaseCylinder BadBlkBaseMaxPageCount The above fields describe the location of the Bad Block map used by IVolume when reinitializing a volume.
233 234 235 237	1 1 2 2	DumpBaseSector DumpBaseHead DumpBaseCylinder DumpBaseMaxPageCount The above fields describe the location and file size of the crashdump area to be used by the Bootstrap ROM when a memory dump is performed.
239 241 243 245	2 2 2 2	BytesPerSector SectorsPerTrack TracksPerCylinder CylindersPerDisk The above fields describe the physical characteristics of the disk drive.
247	1	InterleaveFactor see Sector interleaving
248	2	SectorSize
250	1	SpiralFactor see Sector spiraling
251	1	StartingSector The above four fields describe formatting parameters used by IVolume.
252	4	Reserved for expansion.

The Master File Directory.

The master file directory contains hashed entries for each directory on the volume. An entry for the MFD exists in the file headers under the file name "<sys>Mfd.Sys".

The sector address of the MFD is found in the Volume Home Block, as is its length in sectors. Up to fourteen entries can be stored in each sector of the MFD, and each sector has a one byte header before the MFD entries begin.

MFD entry:

Offset	Size	Field
0	13	DirectoryName Name of the directory; first byte contains the count of bytes in the directory name.
13	13	DirPassword Name of the password; first byte contains the count of bytes in the password name.
26	4	LfaDirbase Address of the the first sector of the directory.
30	2	Cpages Size in sectors of the directory.
32	1	DefaultAccessCode Password protection level of the directory.
33	2	LruCnt Last recently used count; the last used directory has a zero lruCnt. The other directories lruCnts are incremented when a file is accessed which does not belong to the directory.

MFD sector:

Offset	Size	Field
0	2	Header
2	490	rgMFDentries(14) an array of fourteen MFD entries described above.

The Directory

The Master File Directory contains the sector address and the sector size of all the directories on the volume. The Directory has no entry for itself in the File Headers and so may be accessed only through the Master File Directory itself.

File names are hashed into the list using the algorithm described in the appendix. All other bytes in the list are set to zero; a sequential search for files within the directory searches for the first non-zero byte, signifying start of entry.

Offset	Size	Field
0	1	CbFileName The count of bytes in the filename.
1	cbFileName	The File name.
1+cbFilename	2	FileHeaderOffset. The offset from the start of the file headers to the entry for this file.

File Headers

The File Headers contain primary, and optionally, secondary file headers. Secondary file headers reside N sectors past the primary file header, where N is "AltFileHeaderPageOffset" as described in the VHB. Non-active file headers contain a zero length in the file name field. Since there is no notion of the "last" active file header, the file headers are rarely read sequentially - rather the Directory entries are searched for a pointer to the file headers.

The file headers reside in a file describing itself called "<sys>FileHeaders.Sys".

Offset	Size	Field
0	2	Checksum see appendix
2	2	FileHeaderPageNumber. The sector offset from the start of the file headers of the primary file header.
4	51	sbFileName. File name, the first byte contains the length of the file name. If the first byte is zero the file header is inactive.
55	13	sbFileNamePassword. File password, first byte contains the length.
68	13	sbDirectoryName. Directory name the file is located in. The first byte contains the length of the name.
81	2	FileHeaderNumber. The sector offset from the start of the file headers of the primary file header.

Offset	Size	Field
83	2	ExtensionFileHeaderNumber. The sector offset from the start of the file headers of the extension file header in the case of a file with more than 32 extents.
85	1	bHeaderSequenceNumber. Sequential number assigned to extension file headers.
86	1	bFileClass. Not Implemented.
87	1	bAccessProtection. File protection level.
88	4	lfaDirPage. Sector address of the master file directory entry for the file's directory.
92	4	CreationDate.
96	4	ModificatioDate.
100	4	AccessDate.
104	4	ExpirationDate. Not implemented.
108	1	fNoSave. Used by the BackUp Volume utility to determine whether to backup the file. Set to TRUE for FileHeaders.Sys, Mfd.Sys, etc.
109	1	fNoDirPrint. Not Implemented
110	1	fNoDelete Set to TRUE for system files which should not be deleted.
111	4	cbFile Size of the file in bytes.
115	4	defaultExpansion The sector size by which to expand the file if a new extend is to be created.
119	2	iFreeRun Index into the rgLfaExtent and rgoExtent tables below of the next available empty extent.
121	128	rgLfaExtents. Array of 32 sector addresses of the file extents.
249	128	rgcbExtents Array of 32 byte lengths of extents, where the length is a multiple of the sector size.
377	71	Reserved
448	64	application specific field

Bad Block File

The Bad Block file records up to 128 bad sectors on the volume. Its lfa is contained in the VHB, and its length is always one sector. For information on converting cylinder, head, and sector addresses into a lfa, see appendix.

Offset	Size	Field
0	128	RgbBadSector Array of 128 bytes defining the sector number of the first 128 bad spots.
128	128	RgbBadHead Array of 128 bytes defining the head number of the first 128 ad spots.
256	256	RgwBadCylinder Array of 128 words defining the track number of the first 128 bad spots.

Allocation Bit Map

The lfa and size of the Allocation Bit Map is described in the VHB. Each bit in the map describes the availability of one sector on the volume; the bit is set to on if the sector is available. Since the bit map has no entry in the file headers, and hence no end of file description, it can be read successfully by determining the total number of sectors on the volume (this information can be extrapolated from the Device Control Block), and reading only the first nSectors/8 bytes of the bit map. (The DCB is described in the "Memory Structures", below).

Memory Structures Overview

Information about the disk devices available to the system is found in the Device Control Blocks, which describe the characteristics of the device (eg, tracks, sectors, sector size, number of retries), as well as the name and password. The DCB also contains an offset to the memory resident VHB for the device, and offsets to the first and active IO Blocks for it. The pointer to the offset to an array of offsets to the system DCBs is located at 27Ch (in the System Common Address Table, described in the BTOS manual). The number of DCBs is a sysgen parameter.

When a request to open a file is issued, the OS allocates a File Control Block, does a search of the file header block for the filename, and loads the file header number of the file into the FCB (a new file header number is returned if the file did not previously exist). The lfa's and sizes of the file's disk extents are read from the file header and loaded into a linked list of File Access Blocks; the FCB remembering the first FAB. Also a File User Block is allocated which contains a pointer to the FCB and contains the user number. A file handle is assigned to the file and returned to the caller for subsequent operations as an offset to the correct FUB. The User Control Block associated with the task contains a pointer to an array of offsets, which point to the FCB's of the task.

Pointers to the offset of an array of offsets to FCBs and UCBs are stored in the System Common Address Table (SCAT) at addresses 280h and 284h respectively. (Note: these offsets must be combined with the segment address of DGroup of the Operating System, which is found at 242h of the SCAT).

The number of FCBs and FABs are sysgen parameters: the number of FCBs should be equal to the number of files to be open at one time; the number of FABs is a factor of the number of FCBs and how fragmented the volume becomes (i.e., disk extents per open file).

Device Control Block

The DCB can be accessed via the BTOS call "QueryDCB", or through the pointer in the SCAT described above.

Offset	Size	Field
0	1	fMountable
1	1	fNonSharable
2	1	fDoubleDensity
3	1	fNoMultiTrack
4	1	fAttention
5	1	fTimeout
6	13	sbDeviceName
19	13	sbDevicePassword
32	1	controllerNum
33	1	iUnit

Offset	Size	Field
34	1	state
35	1	unitStatus
36	1	deviceClass
37	1	cUsers
38	2	oVHB
40	2	oIOBfirst
42	2	oIOBactive
44	4	lfaMax
48	4	lfaMask
50	2	verifyKey
52	2	ovlyProcOpen
54	2	ovlyProcClose
56	2	cRetryMax
58	2	cSoftSectors
60	2	cHardErrors
62	2	currentCylinder
64	1	sectorSizeCode
65	2	GapLength
67	1	DataLength
68	2	BytesPerSector
70	2	SectorsPerTrack
72	2	TracksPerCylinder
74	2	CylindersPerDisk

File Access Block

Offset	Size	Field
0	2	oChainFAB
2	4	lfaDiskExtent
6	4	sizeDiskExtent

File Control Block

Offset	Size	Field
0	2	oFAB
2	2	devNum
4	2	userCount
6	2	openMode
8	2	FileHeaderNum

File User Block

Offset	Size	Field
0	2	oFCB
2	2	userNum
4	1	fhStatus

Appendix A

Sector Spiraling

Spiraling is a performance mechanism of the floppy device driver on the B20 Series used to reduce the wait time for a sector to come under the head when the head must switch tracks or diskette sides. During the time the head switches or seeks the next track approximately 3 sectors have spun past it; to avoid waiting for the next cSectorsPerTrack-3 to spin past the head, the first sector of the next track/head is assigned the sector number of nSectorPrevTrack+3. The first three tracks of a B20 16 sector diskette are shown below:

```
                Physical Sector Position

                                  1 1 1 1 1 1 1
             1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6

                                  1 1 1 1 1 1 1
Track 1      1 2 3 4 5 6 7 8 9 1 1 2 3 4 5 6

             1 1 1 1                1 1 1 1
Track 2      4 5 6 1 2 3 4 5 6 7 8 9 1 1 2 3

             1 1 1 1 1 1                  1
Track 3      1 2 3 4 5 1 1 2 3 4 5 6 7 8 9 0
```

Sector Interleaving

Interleaving is a sector mapping technique used on B20 hard disks to match the spin speed of the disk to the disk controller's ability to read sectors. Since 3 sectors spin by the head in the time it takes the controller to be ready to read the next sector. Sector numbers are assigned which are three past the last physical position read.

The sectors of a track of an interleaved disk are shown here:

```
                Physical Sector Position

                                  1 1 1 1 1 1 1
             1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6

                   1 1 1        1 1        1 1
Track 1      1 4 7 0 3 6 3 6 9 2 5 2 5 8 1 4
```

Hashing Algorithm

The following hashing algorithm is used to place and locate entries within the Master File Directory and the Directories to determine the sector within which the entry resides.

The arguments the hasher takes are name (directory or file) and the size in sectors of the table (Mfd.cDirectory or Vhb.cMfd).

```
"Name" is an array[1..n] of bytes;
"Divisor" is the size in sectors of the structure;
"n" is the length of Name;
"b" is a byte;
"x" is a word;

x := 0;
For i := 1 to n do
begin
   b := name[i];
   if b >= 'a' and b <= 'z'
         then b := b-#20;
         {make upper case}
   x := 73*x+b;
end;
hashSectorNumber := x Mod Divisor;
```

Checksum Algorithm

The Volume Home Block and the File Headers use this checksum, however the VHB checksums the first 128 words only, while the FHB checksums 256 words. The value of magicWord is 7C39h, and can be found in the VHB.

```
"w" is a word;
"nLastWdSector" is the number of words to checksum;
"wBuffer" is the sector to be checksummed;

w := magicWord;
for i := 1 to nLastWdSector do
      w := w - wBuffer[i];
if w <> 0 then erc := ercBadCheckSum;
```

Appendix B

Computing Lfa's from head/cylinder/sector

The algorithm for computing a logical file address (lfa) given the cylinder, head, and sector number (where sector number is zero based) is :

```
Lfa = ((cylinder * Dcb.TracksPerCylinder + Head) *
      (Dcb.SectorsPerTrack * Dcb.BytesPerSector)) +
      ((iSector-1) * Dcb.BytesPerSector).
```

Terminology

Prefix	Meaning
lfa	logical file address, 4 bytes in length.
sb	character string where the first byte is the length of the string.
p	memory pointer, 4 bytes; most significant word is segment address, least is offset.
o	offset, the register address portion of a pointer
rg	array of.
c	count of, displaces a word.
b	byte.

(note that words are in Intel format, most significant byte last)