Partition Table

From OSDev Wiki

The following information about Partition Tables only applies to hard disk drives. Most other storage devices use other partitioning techniques, if they even use partitioning at all.

Contents

MBR

The Master Boot Record is the traditional way of storing partition information about a hard disk, along with some boot code. That is, the Partition Table is contained inside the MBR, which is stored in the first sector (cylinder 0, head 0, sector 1 -- or, alternately, LBA 0) of the hard drive. (See the MBR article for the overall structure and contents of the MBR.)

Almost all PCs still use an MBR for booting hard disks, and for storing partition information on hard disks. Traditional MBRs are nearly obsolete at this time, because the 32 bit design of the LBA fields in the Partition Table begins to "overflow" when dealing with disks larger than 2Tb. One possible replacement for the MBR system is GPT (see below). It might also be possible to agree on a new standard for the MBR, with 48 bit LBA fields for the partitions (see below).

Information about primary partitions and an extended partition is contained in a 64-byte data structure located in the MBR. This Partition Table conforms to a standard layout that is independent of the operating system. Each Partition Table entry is 16 bytes long, making a maximum of four entries available. Each entry starts at a predetermined offset from the beginning of the sector, as follows:

Partition number	Offset
Partition 1	0x01BE (446)
Partition 2	0x01CE (462)
Partition 3	0x01DE (478)
Partition 4	0x01EE (494)

Note: Naming the partition table entries as "1" through "4" is for convenience only. The partition table entries are not required to be in any kind of order.

Each of the four Partition Table entries contains the following elements, in the following structure:

Element (offset)	Size	Description
0	byte	Boot indicator bit flag: 0 = no, 0x80 = bootable (or "active")
1	byte	Starting head
2	6 bits	Starting sector (Bits 6-7 are the upper two bits for the Starting Cylinder field.)
3	10 bits	Starting Cylinder
4	byte	System ID
5	byte	Ending Head
6	6 bits	Ending Sector (Bits 6-7 are the upper two bits for the ending cylinder field)
7	10 bits	Ending Cylinder
8	uint32_t	Relative Sector (to start of partition -- also equals the partition's starting LBA value)
12	uint32_t	Total Sectors in partition

Notes:

- Any one of the partitions may be "active" (ie. bootable).
- At most one partition should be active.
- The Partition Table entries are **not** aligned on 4 byte boundaries (if the MBR is itself loaded into memory on a 4 byte boundary).
- Therefore, neither are the two uint32_t LBA entry values -- so the LBA values cannot be copied directly into a register.
- The Cylinder, Head, Sector fields (taken together) are only 3 bytes (24 bits) long.
- Sector values (in the CHS fields) of 0 are illegal.
- CHS fields "max out" on a drive that is approximately 8GB in size -- and are therefore useless on almost all current drives.
- For drives smaller than 8GB, the LBA fields and the CHS fields must "match" when the values are converted into the other format.
- For drives bigger than 8GB, generally the CHS fields are set to Cylinder = 1023, Head = 254 or 255, Sector = 63 -- which is considered an invalid setting.
- If a Partition Table entry is unused, then it should be set to all 0.
- A System ID byte value of 0 is the definitive indicator for an unused entry.
- Any other illegal value (CHS Sector = 0 or Total Sectors = 0) may also indicate an unused entry.

The System ID byte

The System ID byte is supposed to indicate what filesystem is contained on the partition (ie. Ext2, ReiserFS, FAT32, NTFS, ...). There was never any standard created for the System ID byte -- which means that Microsoft went and tried to hog almost all of the possible values. See the links below for tables of values of the System ID byte, for filesystems that have been lucky enough to acquire their own value by common consensus.

If you create your own custom filesystem, then you can simply pick a System ID value for your filesystem that seems to be unused. There is also an attempt to standardize the use of System ID value = 0x7f (by the Alt-OS gang), to cover all custom filesystems that follow the standard -- however, their effort seems to be losing steam.

"Unofficial" 48 bit LBA Proposed MBR Format

The two CHS fields are unused in all current drives, leaving only the two 32 bit LBA fields to "do all the work". But there never was any 32 bit LBA addressing mode -- the current "standard" LBA addressing mode is 48 bits. So it is reasonable to try to redefine the Partition Table to eliminate the unused CHS fields, and use the extra space to extend the two LBA fields to a full 48 bit size. This would eliminate the impending obsolescence of the entire MBR scheme.

It seems reasonable to try to preserve (as much as possible) the current Partition Table structure. Therefore, the following alternate structure for 48 bit LBA Partition Table entries is proposed:

Element (offset)	Size	Description
0	byte	Bitflags field: 1 = not bootable, 0x81 = bootable (or "active")
1	byte	Signature-1 (0x14)
2	uint16_t	Partition Start LBA (high 16-bit of 48 bit value)
4	byte	System ID
5	byte	Signature-2 (0xeb)
6	uint16_t	Partition Length (high 16-bit of 48 bit value)
8	uint32_t	Partition Start LBA (low uint32_t)
12	uint32_t	Partition Length (low uint32_t)

Note: The basic intent is to use bit #0 (value = 1) in the bitflags field as a "48 bit LBA" indicator, preserve the offsets and functions of the bitflags and System ID fields, and use the two available aligned 16-bit values to extend the LBA fields. When detecting a valid partition entry, a little extra verification turns out to be useful, so the "extra" bytes in the table entry should be loaded with signature values.

Extended Partitions

Extended partitions are a way of adding more than 4 partitions to a partition table.

The partition table may have one and only one entry that has the SystemID 0x5 (or 0xF). This describes an extended partition.

An extended partition is one physical partition that is subpartitioned into "logical" partitions. So, the partition table entry describing it has a StartLBA and NumSectors that describe the space inside which any number of logical partitions may sit.

At the start of an extended partition is an extended partition table. This takes exactly the same form as a normal partition table, apart from most of the fields are unused. Only two of the partition entries are used - the first one describes the desired logical partition, and the second one is a link (much like a linked list) that points at another extended partition table. The size should officially include the logical partition that follows with it. The remaining two entries are not used and should be left zeroed out.

Note that the StartLBA fields in these extended partition table entries are relative to the start of the extended partition itself.

GPT

GPT is an updated Partition Table standard, that has been adopted as the recommended partition mechanism under UEFI. It does not contain the artificial 24 bit or 32 bit limitations of the MBR Partition Tables. It also contains enhancements to the concept of partition tables, in general, and is significantly more complex than the MBR scheme.

See Also

GPT

External Links

- System ID byte values ("Partition Types") for known filesystems (http://www.win.tue.nl/~aeb/partitions/partition_types-1.html)
- Microsoft's GPT info (http://www.microsoft.com/whdc/device/storage/GPT-on-x64.mspx)
- A list of System IDs/Partition Types from osdever (http://www.osdever.net/documents/partitiontypes.php)
- Another list of System IDs/Partition Types from osdever (pdf) (http://www.osdever.net/documents/pdf/partitiontypes.pdf)
- The wikidpedia article on the Extended Boot Record (AKA Extended Partition) has lots of pictures and many more words (http://en.wikipedia.org/wiki/Extended_partition)
- GPT at Wikipedia (http://en.wikipedia.org/wiki/GUID_Partition_Table)