How mformat-3.9.10 and above calculates needed FAT size

Alain Knaff

November 13, 2010

This small document explains the formula used by `mformat.c` to figure out fat size and number of clusters. Due to the way that filesystem parameters are stored in the boot sector, we can afford to have a FAT that is larger than need to be to accomodate the clusters present on disk, but under no circumstances can we have one that is too small.

In this discussion, we use the following variable names:

$fatNybls$	Number of nubbles (4 bit unit per FAT). This is 3 for FAT12, 4 for FAT16, and 8 for FAT16
$numClus$	Number of clusters on the disk
$clusSiz$	Size of a cluster, expressed in sectors
$secSiz$	Size of a sector, in bytes. Size of a sector in nybbles is $secSiz$ * 2
$nfats$	Number of FAT copies, usually two
$remSects$	"Remaining sectors", after number of boot sectors and root directory have been accounted for
$fatLen$	Length of the FAT, in sectors

Taking into account both data and fat, each cluster takes up the following number of nybbles (units of 4 bytes):

$$clusSiz * (2 * secSiz) + nfats * fatNybls$$

This accounts for the data of the cluster ($clusSiz * secSiz$), as well as for the space taken up by its descriptor.

The space taken up by all clusters together, plus the space taken by descriptors for clusters 0 and 1 ($2 * nfats * fatNybls$) should be less than what is available.

Additional sectors may be lost due to slack (you have to use a full FAT sector, you also have to use a full cluster in the data area). Thus, an *upper bound* for the number of clusters is as follows:

$$numClus \leq \frac{2 * remSect * secSiz - 2 * nfats * fatNybls}{2 * clusSiz * secSiz + nfats * fatNybls}$$

$$numClus \leq \frac{(remSect + 2 * clusSiz) * 2 * secSiz}{2 * clusSiz * secSiz + nfats * fatNybls} - 2$$

These will take up at most the following space in one copy of the FAT (we have to round up, because even a half-full fat sector must be taken in its entirety)

$$fatLen \leq \left\lceil \frac{(numClus + 2) * fatNybls}{secSiz * 2} \right\rceil$$

$$fatLen \leq \left\lceil \frac{\left(\frac{2*(remSect+2*clusSiz)*secSiz}{2*clusSiz*secSiz+nfats*fatNybls} \right) * fatNybls}{2 * secSiz} \right\rceil$$

$$fatLen \leq \left\lceil \frac{(remSect + 2 * clusSiz) * fatNybls}{2 * clusSiz * secSiz + nfats * fatNybls} \right\rceil$$

The space left after FAT sector has been deduced is now *less than or equal* to what would be needed for the data area of the clusters (including fractional clusters), which is good, as we may have under no circumstances *more* clusters in the data area than in the FAT. An important point in this calculation is that we based the needed FAT size on a *fractional* number of clusters, rather than a rounded down amount of clusters. Indeed, using a rounded down number could have exposed us to a situation where we had an *almost enough* space for one more cluster (i.e. not enough space for data + FAT, but enough for data alone). This situation, combined with a situation where the last FAT sector was flush full could have lead to a situation where *numClus* would become too large for the FAT to accomodate. I think this is clearer with an example:

- $remSect = 4127$, $clusSiz = 1$, $nfats = 1$
- (Non rounded down) $numClus = \frac{(4127+2)*(1024)}{1032} - 2 = 4094.992$
- Rounded down: 4094 clusters
- These fit into 16 FAT sectors, exactly
- ... leaving us 4095 clusters, which is one to many (because these 4095 clusters would now need 17 FAT clusters)

Keeping the fractional part (0.992) allows us to round *up* the needed number of FAT sectors to 17, nicely solving this problem.

The downside of counting the fractional part however is that we quite often waste a sector in the really common situation where both $nfats$ and *clusSiz* are even, while *remSect* is odd. An easy way to address this is to substract one from *remSect* before application of the formula, if this case is detected. Such operation carries no risk, as the odd final sector cannot be used to make a full cluster.

There is still a case however, where fatLen must be grown manually after application of the formula: If numClus exceeds the maximum number of clusters allowable for FAT12 or FAT16), we need to shrink *numClus* after application of the formula, and manually make the FAT larger in order to take up any excess space.

Also note that as we used upper bounds, we may waste a certain number of sectors, which an exact calculation may not have wasted. However, normally, we should not lose more than one sector per FAT copy.

N.B. In its document at http://www.microsoft.com/hwdev/download/hardware/fatgen103.pdf, Microsoft proposes a much simpler formula. However, this formula is both wrong (i.e. occasionally it produces a smaller FAT than is needed for the clusters on disk), less generic (only works for sector sizes equal to 512), and less efficient (in case of FAT32, it may waste up to 8 sectors!)

The formula is the following (for FAT16):

$$fatLen \leq \left\lceil \frac{remSect}{256 * clusSiz + nfats} \right\rceil$$

Note that it doesn't account for the dummy clusters 0 and 1. Thus, if we have 258 sectors remaining, with a cluster size of 1, and two FAT copies, the Microsoft formula mistakenly assumes fatLen = 1. This leaves 258 - 2 = 256 sectors for clusters, which yields 256 clusters. However, those clusters do not fit into the FAT, as two clusters are lost (0 and 1). However, to Micro$ofts' credit, this is not actually the formula they're using (tested with $remsect = 160056$ and $clusSize = 4$), so this seems to be a documentation issue rather than a genuine bug.

In case of FAT32, Microsoft just halves the denominator. This is wasteful, as only the $256 * clusSiz$ part would need to be halved.

If we assume 16777000, and a cluster size of 8, our formula would give us:

$$fatLen = \left\lceil \frac{(16777000 + 16) * 8}{2 * 8 * 512 + 16} \right\rceil = 16352$$

This would leave $16777000 - 2 * 16352 = 16744296$ sectors for the clusters, i.e. 2093037 clusters. The FAT descriptors for those 2093037 clusters do indeed fit into our 16352 fat sectors.

Microsoft, on the other hand, would get:

$$fatLen = \left\lceil \frac{16777000}{(256*8+2)/2} \right\rceil = 16368$$

This would only leave $16777000 - 2*16368 = 16744264$, i.e. 2093033 clusters, thus wasting 4 clusters. The FAT would be 28 sectors too big, i.e. more than the mere 8 sectors announced by Microsoft! Unfortunately, I currently don't have access to any sufficiently recent Windows to check out whether this really happens in the code, or whether it is only a documentation issue as well.

Oh, and before somebody points it out: the formula in this document occassionnally wastes sectors too, although not as much (Example: With $remsect = 685$, $clusSiz = 1$ and $nfats = 2$ our formula gives $fatLen = 3$, which leaves 679 clusters. However, we could use $fatLen = 2$, leaving 681 clusters.