Ask the Core Team

Microsoft Enterprise Platforms Support: Windows Server Core Team

The Four Stages of NTFS File Growth

Jeff Hughes (CORE) 16 Oct 2009 11:09 AM 5

In my quest to better understand the interworking of how NTFS stores information on disk, I have been researching what happens to a file as it grows in size and complexity. The reason I'm after this knowledge is so I can better troubleshoot certain storage issues.

Recently, I realized that I'd stuffed my head with enough information to make a pretty good blog. Read along as I explain what I call 'the four stages of file growth'.

Before we can address file growth, we need to first look at how NTFS works under the covers.

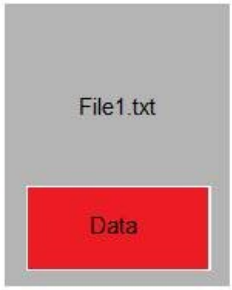
Let's start out with some basics.

When NTFS stores a file, it starts by creating a small 1KB file record segment that we will call the base record. Every file starts like this, including the special hidden files such as $MFT, $LOGFILE, $VOLUME and so on. In fact when we refer to the MFT (master file table), what we are talking about is the entire list of base record segments and child record segments (explained later) for all files in the volume.

For today, we are just going to talk about some simple text files. You will see it getting complex enough without us doing anything fancy. Here are three base records for three text files.

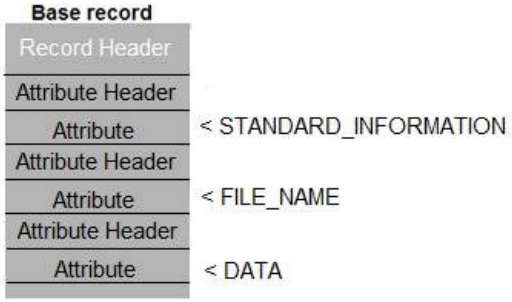
Before going any farther, it is important to clear up a common misconception on what a file really is. We tend to think of the data in our file as the file itself.

The truth is that data is just one *attribute* of a file.



Every file record starts with a header, and then has various attributes, each attribute having its own header. For small files, it is common to find the data attribute last.

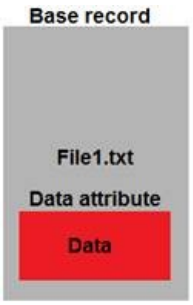
Do not confuse these attributes with file attributes like Read-only, Hidden, or System (which are actually just flags). Think of attributes as structures within the file that define things about the file. Common attributes are $STANDARD_INFORMATION, $FILE_NAME, and of course $DATA.



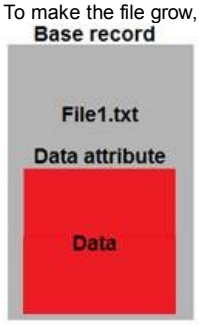
Any space left over in the 1KB record is unused until one of the attributes needs it or a new attribute is added.
Now let's watch our file grow….

Stage one – Completely resident

I created a small text file with just one line of text in it. This file was so small that it was able to fit all parts of the file into its base record. We call this being *resident*, as the data for the file *resides* in the base record segment. This also means that the entire file exists in the MFT. No need to look elsewhere. Everything we need is in that 1KB record.
The diagram shows our 1KB base record segment for the file File1.txt. Inside you can see the data attribute and the file data within it. The file data, also known as the *stream* for this attribute, is what we as computer users tend to think of when we think about a file. We don't think about all the structures involved in storing the stream.



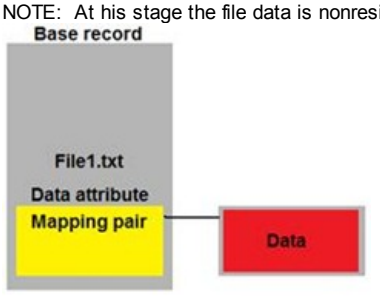
Along with the data that we put in the file, you can also see that we have lots of room still left in the 1KB base record segment.
To make the file grow, I just pasted the same line of text into it a few more times. Soon I had the file looking like this….



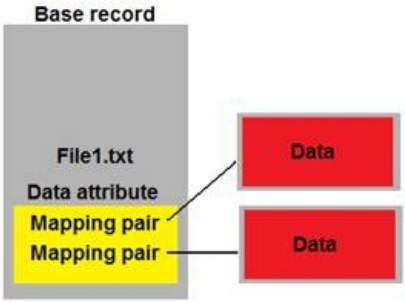
This was about as big as I could get the file before it was too big to fit into the 1k range of the base record segment. Any bigger and we go to stage two.

Stage two – Nonresident Data

Once the data starts to push out toward the end of the 1KB base record segment, the data will be shipped outside and stored elsewhere on the disk. To keep track of where it is, we maintain a mapping pair that tells us the location and length of the now *nonresident* data. The new location is outside the MFT and is simply an allocated range of clusters.
NOTE: At his stage the file data is nonresident, but the attribute record is in the base record segment.



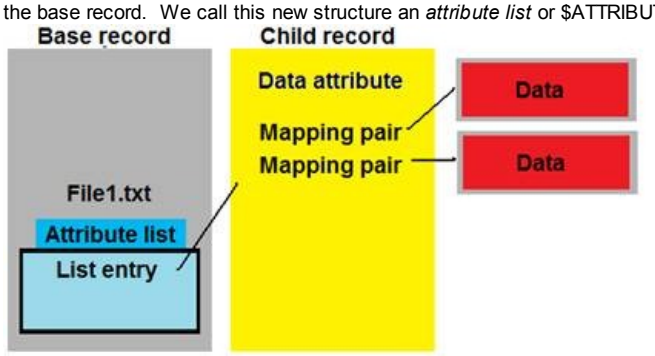
As the file continues to grow we will either increase the length defined by the mapping pair, or if we can't store the data contiguously, we create more mapping pairs.
Eventually, the file starts to look like this….



Stage three – Nonresident Attribute

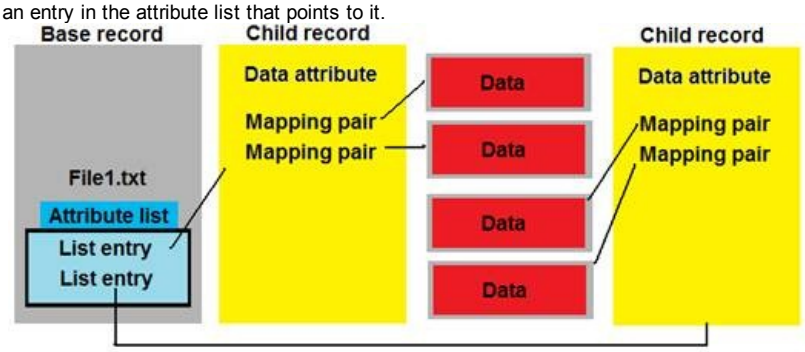
When an attribute grows to the point that the list of mapping pairs no longer fits into the base record segment, it too is shipped out but this time it is housed in a new *child record*. To keep track of this child record a new structure is created in

the base record. We call this new structure an *attribute list* or $ATTRIBUTE_LIST.



Each entry in the attribute list points to the file record where each attribute instance can be found. There will be an attribute list entry for nearly every attribute that the file has. The exception being that there isn't a list entry for the attribute list. For the attributes still resident (like the $FILE_NAME attribute), their respective list entry will simply point back to the base record segment. The diagram above shows only the one entry that corresponds to the $DATA attribute. The other entries are left out of the diagram to keep it readable.

After even more data is stuffed in the file it branches out and creates more child records as needed. Each child record has an entry in the attribute list that points to it.



This is somewhat different than what we did when we moved file data outside the base record segment. When the file data was moved, the new location on disk contained no attribute information. It just had data. If viewed in a sector editor, it would just show lines and lines of file data.

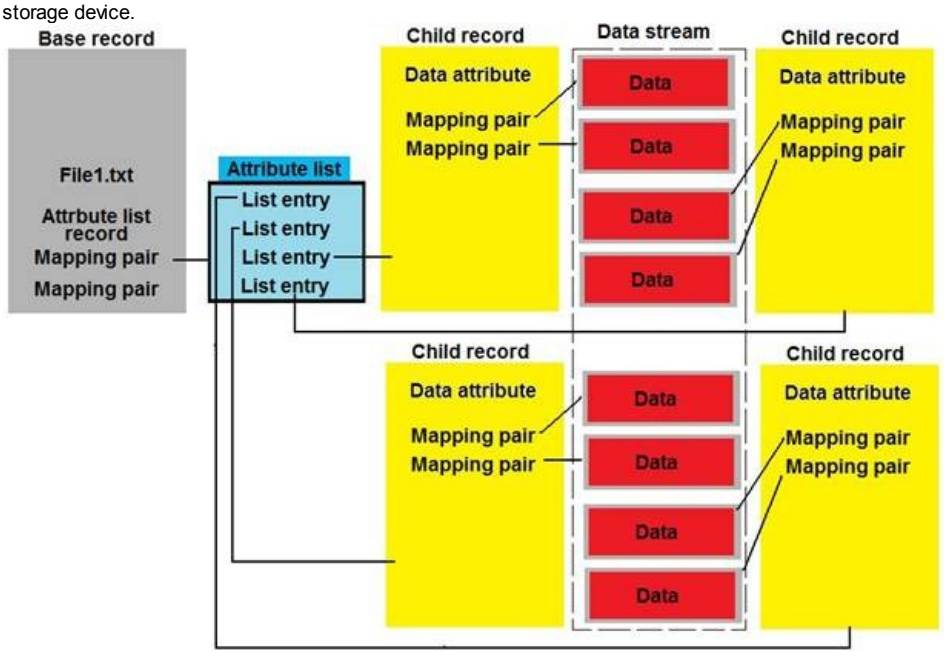
The child records are just that, records. They contain elements common to those found in a base record segment. It will have a MULTI_SECTOR_HEADER and one or more attribute records….along with some mapping pairs. The pairs themselves will point to the allocated clusters that contain actual file data.

The information dumped out of the file gets more complex at each stage. But fear not, it's almost over.

Stage four – Nonresident Attribute List

The final stage of file growth occurs when attribute list contains so many entries that the list itself no longer fits in the base record segment. When we reach that point, the attribute list is shipped outside the record into an allocated cluster range and an attribute list record is left behind to track the location of said cluster range. The new location of the attribute list is outside the MFT and is similar to how we are storing the chunks of data that make up the $DATA:"" stream(shown in the red boxes) in that it is not an actual child record.

The dotted line shows the entire stream as it would be virtually. Logically these chunks of data will be found all over the storage device.



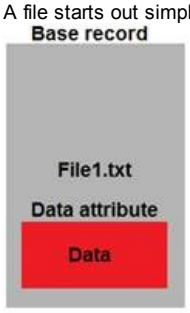
Unlike the child records and the data instances, a file can only have one attribute list and the $ATTRIBUTE_LIST record must

reside in the base record even though the list is nonresident.

In review

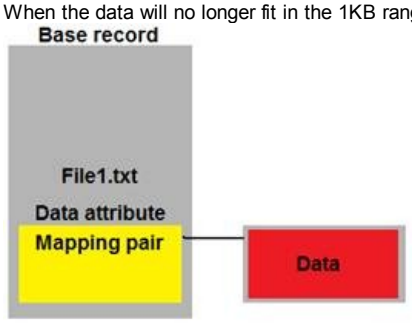
Stage one- Completely resident

A file starts out simple, storing file data locally.

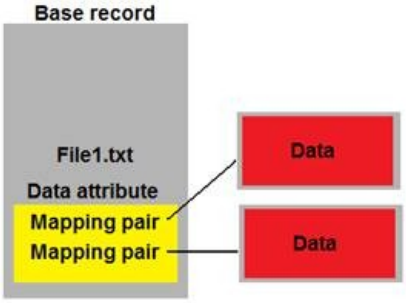


Stage two- Nonresident data

When the data will no longer fit in the 1KB range, it is moved to another part of the disk.

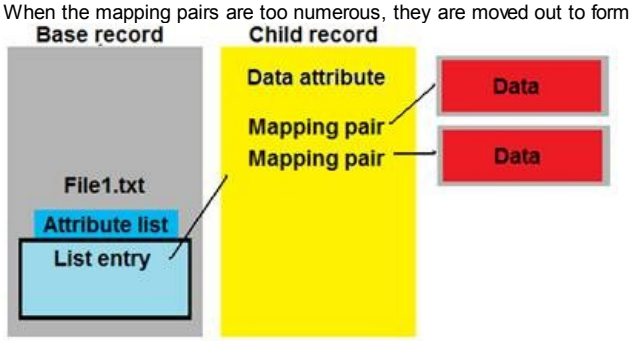


This process can result in multiple mapping pairs.

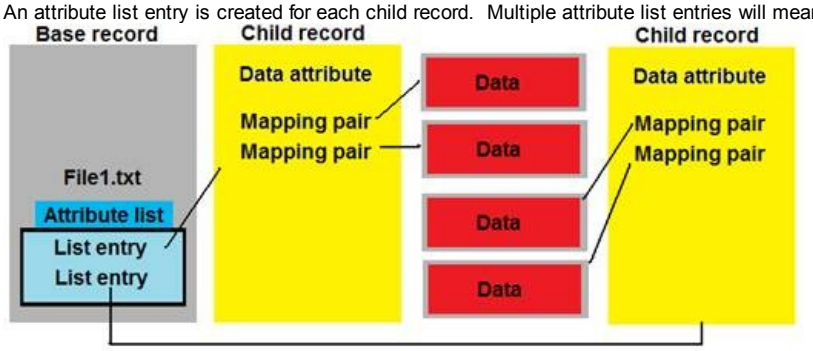


Stage 3-Nonresident attribute

When the mapping pairs are too numerous, they are moved out to form their own child record.

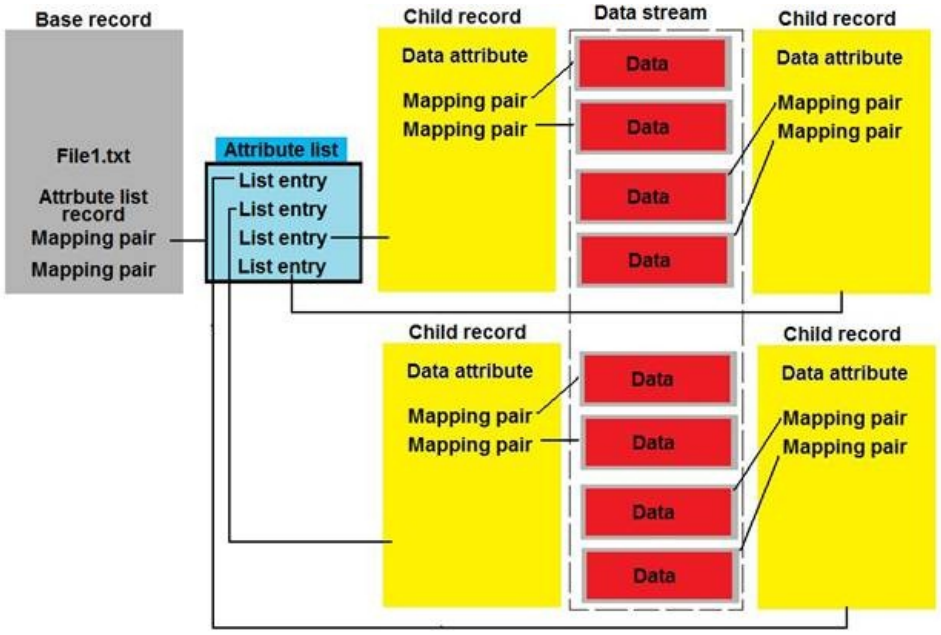


An attribute list entry is created for each child record. Multiple attribute list entries will mean multiple child records.



Stage 4-Nonresident attribute list

Lastly, when the list of attribute entries is too large to be stored inside the base record segment, the attribute list itself becomes nonresident and moves outside the MFT.



The greater the complexity used to store the file, the greater the performance hit will be to your computer when retrieving and storing the file. Things like compression, file size, number of files, and fragmentation all can greatly affect this complexity and therefore affect your computer experience.

Robert Mitchell
Senior Support Escalation Engineer
Microsoft Enterprise Platforms Support

Comments

ichbinpete 18 Feb 2011 3:11 PM

Great blog! Very informative.

3 Aug 2012 5:39 PM

Vladimir

Hi Jeff, sorry for making a comment on a post so old, but as I could not find the information by myself, I had to ask. Will the base record always be 1KB? When the data inside then increase, it does not matter, it will keep its size, correct? What about the atribute list and child records, what is their size? The size of the disk cluster (4kb, for example)? thanks, Vladimir

Robert Mitchell [MSFT] 28 Nov 2012 10:01 PM

Vlad, File records were 1KB for the longest time. When 4KB sector drives started to hit the market, this changed somewhat. For 4KB sector drives that are 'Native 4K', the file records will be 4KB. In Win8/Server2012, the FORMAT command was also given a new switch (/L) that allowed you to format with a 4KB file record size regardless of the physical sector size. I wrote an article for Windows IT Pro about 4K drives. www.windowsitpro.com/.../support-advanced-format-hard-drives-141584

9 Sep 2013 3:54 PM

DaBert

A very interesting article. But could you please update links to the images. There are none of them displayed. Thank you.

3 Apr 2014 2:06 PM

Moulzy

Really great interesting post (even 5 years after!) and nice from your part to share your knowledge here.
THANKS!