[Q]: OS/2 vs. NT: paging subsystem

[A]: Jonathan de Boyne Pollard (2:257/609.3)

It's worth noting some interesting things about Windows NT when compared to OS/2 Warp in this respect. The “portable executable”, PE, format for executable files used in Win32 *does* contain an exact image in the file of the page as it is to be loaded into memory. When Windows NT demand loads a page, it doesn't need to uncompress its contents. Indeed, in most cases it doesn't need to perform relocation fixups either, because of a trick used when creating Win32 import libraries that means that all of the fixups to references imported from other modules are concentrated in a single place. (This trick is actually not specific to the PE executable format, and can be duplicated on OS/2 with the LX executable format as well. I have a replacement OS2386.LIB that does it for fixups to the various system API DLLs, if anyone is interested.)

The disadvantage, of course, is that reading in a page from DASD is more expensive on Windows NT than it usually is on OS/2. In the 32-bit LX executable format used by OS/2, the compression scheme will shrink the size of page images in the file quite noticably. Picking the file \OS2\CMD.EXE at random, we notice that in memory object 1 all of the page sizes are between 3584 and 3072 bytes, a reduction in size by between 12% and 25%. Because of the compression used in the LX executable format, to demand page in a 4KiB page the program loader in the OS/2 kernel often doesn't actually need to read 4KiB of data from disc.

On Windows NT, however, pages are the same size when stored on disc as they are in memory, because the PE executable file format doesn't have compression. So for every 4KiB page to be demand loaded, Windows NT has to read an entire 4KiB of data from disc.

It's worth noting that Windows NT attempts to compensate for this fact by the fact that NTFS has a minimum cluster size of 4KiB. With HPFS on OS/2, the smallest I/O transaction can be as small as a single 512 byte (0.5KiB) sector, since that is the allocation unit size. Reading in a 3072-byte compressed page image thus only need involve reading six or seven sectors, not eight. With NTFS on Windows NT, since the smallest allocation unit size for the filesystem is 4KiB *anyway*, it doesn't make any difference that the program loader needs to read a full eight sectors for each 4KiB page (or even 9 or 10 sectors if the developer hasn't page-aligned the executable properly, which is possible if he has played around with the linker flags). It couldn't read less even if it wanted to.

The most obvious effect of this design is that PE executables are much larger than LX executables. A page containing repeated data (such as an initialised data page that is mostly zeroes, for example) compresses very well in an LX executable. By contrast, a PE executable file contains a whole page's worth of bytes for such a page. Viewing PE and LX executables with a hex file viewer is most instructive. PE executables often have large runs of repeated data, most often large runs of zero bytes. LX executables generally do not. (I say “generally”, because if they use Watcom C/C++ the linker doesn't support compression, alas. This is a deficiency in Watcom's linker, and an unfortunate example of the “jack of all trades, master of none” adage.) This is, of course, visible in the comparative sizes of Win32 and 32-bit OS/2 executables.

One particular irony of the “uncompressing pages during a page-in is expensive, so we don't do it” philosophy embodied in the PE executable file format design is that NTFS can compress file data behind the scenes on the disc. So if an executable file is on an NTFS volume and NTFS has compressed it when storing it on disc, the overhead of uncompressing data each time that there is a page in operation won't have been avoided. All that has changed in reality is the portion of the system that actually performs it. Rather than having the uncompression done by the process loader, it is done by the filesystem driver. It is still done.

Another further irony is that making the executable file format uncompressable, but having compression in the filesystem itself, means that the page data in the in-memory file cache are uncompressed, because of course that is how they are in the file itself. In contrast, when an LX executable file is cached on OS/2 the page data *are* compressed, and less RAM is required to cache the file contents as a result. This is one contributory factor (of many, alas) to the greater physical memory needs that Windows NT has when compared to 32-bit OS/2.