File Manager Caches How the three File System caches work

There are three **File Manager caches** on the Macintosh. Caches are used to create a buffer of data in RAM to minimize disk hits and improve application performance. The **type 1** and **type 2** caches are done automatically by the File System. The **type 3** cache is user-controlled and can be used to enhance application performance. The **type 3** cache is not available on 64K ROM machines.

The first type of cache has been around since the days of the 64K ROM machines. Under **type 1** caching, each volume has a one-block buffer for all file/volume data. This prevents a read of two bytes followed by a read (at the next file position) of 4 bytes from causing actual disk I/O. This cache exists in the system heap, unless HFS is using the new File System caching mechanism (see **type 3** caching described below). The volume allocation map also gets saved in the system heap but is not considered part of the cache.

The second type of caching is only available with the enhanced Sony driver, which will cache the current track (up to twelve blocks) so that subsequent reads to that track may use the cache. The **type 2** track cache is "write through"; all writes go to both the cache and the Sony disk so flushing is never required. Note that track caching is only done for synchronous I/O calls; when an application makes an asynchronous I/O call, it expects to use the time while the disk is seeking, etc., to execute other code.

The **type 3** cache is only available under the 128K and greater ROMs. This user-controlled cache is *NOT* "write-through". This cache is configured via the control panel by the user and it always makes sure that there is room for 128K of application heap and 32K of cache. If the user-requested amount would reduce the heap/cache below these values, then the cache space is readjusted accordingly.

The **type 3** cache allocates storage in the application heap in the space above BufPtr. This is really the space above the jump table and the "A5 world", not technically part of the application heap. However, moving BufPtr down will cause a corresponding reduction in the space available to the application heap. The **type 3** cache will also use the space previously allocated for use by the **type 1** cache since all types of disk blocks can be accommodated by the **type 3** cache.

The bulk of the caching code used for this RAM cache is also loaded above BufPtr at application launch time. This is accomplished by the 'INIT' 35 resource which is installed in the system heap and initialized at boot time. At application launch time, 'INIT' 35 checks the amount of cache allocated via the control panel and moves BufPtr down accordingly before bringing in the balance of the caching code. The RAM caching code is in the 'CACH' 1 resource in the System File.

Up to 36 separate files may be buffered by the **type 3** cache. Information is kept about what blocks are cached for each file. The cache uses the "least recently used" algorithm to determine which blocks are released when new blocks become eligible for caching. The files themselves are queued in such a way as to allow for efficient flushing of the blocks to disk, i.e. the file order approximates disk order.

If the user has enabled the **type 3** cache via the control panel, all files read from or written to by the File System (HFS) calls are subject to caching under the current implementation. The cache is not "write through" like the track cache (**type 2** described above). When a File System write (PBWrite, WriteResource, etc.) is done, the block is buffered until it is released (age discrimination), until a volume flush is done, or until the application terminates.

It may be useful for an application to prevent this process of reading and writing "in place". The Finder disables caching of newly read/written blocks while doing file copies since it would serve no purpose to cache files that the Finder was reading into memory anyway. Copy protection schemes may also need this capability. Disabling reading and writing in place is accomplished by setting a bit in a low memory flag byte, CacheCom. When you set this flag, no new candidates for caching will be accepted, but blocks already saved may be still read from the cache. CacheCom is at low memory location 0x39C. Bit 7 is used to disable subsequent caching. Bit 6 can be set to force all I/O to go to the disk. If Bit 6 is set, the disk will be accessed every time even one byte is requested from the File System.

Apple recommends that you don't change the CacheCom flags. Instead, to avoid the problem of cached data not being saved to disk, call FlushVol after each time you write a file to disk. This ensures that the cache is written, in case a crash occurs soon thereafter.

One final note on BufPtr. The RAM-resident caching software arbitrates BufPtr in the friendliest manner possible. It saves the old value away before changing it, and then when it is time to release its space it looks at it again. If BufPtr has been moved again, it knows that it can't restore the old value it saved until BufPtr is put back to where it left it. In this manner any subsequent code or data put up under BufPtr is assured of not being obliterated by the caching routines.