Locking Within An MFS

An MFS may require record locking as part of its own processing. This will be the case, for instance, if an MFS such as an audit trail MFS must update files. As with any program, if executing in a network environment, the MFS should preserve file integrity by locking all records that are to be updated.

If an MFS must lock records, it should never do so with direct calls to the filing system. Instead, the MFS should execute BASIC+ LOCK calls. Moreover, when executing the BASIC+ LOCK calls, the MFS should always first fetch a proper file handle using a BASIC+ OPEN statement.

(Note that this is in contrast to the method used to access data in the file -- see the topic "Access to a File from Within an MFS" earlier in this chapter. Access to the same file should not be done via BASIC+ statements, whereas locking always must be done via BASIC+ OPEN and LOCK.)

By executing an BASIC+ LOCK statement, the local lock table will be updated properly. A direct call to the filing system will not pass through the BASIC+ interpreter, and the local lock table will not be aware of locks made in this way; coordination with locally-held semaphores will not be possible.

In order to pass the proper file handle to the LOCK statement, the MFS should first fetch the handle using an BASIC+ OPEN statement. Though it might seem possible to use the **HANDLE** argument if locking files in the current file, the **HANDLE** argument does not contain the same handle information as returned by an BASIC+ OPEN statement. Since application programs use the full BASIC+ OPEN file handle when locking, an MFS should do so as well in order to guarantee coordination with these higher-level processes.