Technical Note FL26
Lock, Unlock the Range

CONTENTS

This Technical Note discusses the `_PBLockRange` and `_PBUnlockRange` routines; how they act on local and shared volumes and why you should not set the `ioPosMode` field to `fsFromLEOF` in the parameter block for those routines when accessing a file on an AppleShare volume.

[Apr 01 1988]

Introduction

A file can be opened with a shared read and write permission (`ioPermssn` = `fsRdWrShPerm`) to allow several users to share the data in that file. (When a file is opened more than once, each open is considered a different "user".) When you, as a user of a shared file, need to modify a portion of the file (for example, a record in a database), it is usually very desirable to make that portion of the file unavailable to otherusers while you are making your changes. The `_PBLockRange` call is provided to lock a portion or range of bytes of the file. `_PBUnlockRange` can be called later to unlock the portion of a file locked with `_PBLockRange`. If you have read the File Manager chapters of Inside Macintosh IV and V, you know all of that, but there are a few things that it doesn't tell you, and that is the topic of this Note.

Back to top

Local Versus AppleShare

`_PBLockRange` and `_PBUnlockRange` do nothing when used on files that are on a local HFS (or MFS) volume unless local file sharing is in effect under System Software 7.0. The two routines do not return any errors and they do not lock or unlock a range of bytes. Following is a way to verify that `_PBLockRange` works on an open file:

```
    {Start with an open file}
    WITH theParmBlk DO
      BEGIN
        {ioRefNum from open call}
        ioCompletion := NIL;
        ioReqCount := 1;          {lock a single byte}
        ioPosMode := fsFromStart; {at the beginning of the file}
        ioPosOffset := 0;
      END;
    theErr := PBLockRange(@theParmBlk, FALSE);   {lock the byte - ignore error}
    theErr := PBLockRange(@theParmBlk, FALSE);   {lock it again}
    IF theErr = afpRangeOverlap {see if PBLockRange locked it the first time}
      THEN
        BEGIN
          RangeLockPresent := TRUE; {Range was locked}
          theErr := PBUnlockRange(@theParmBlk, FALSE); {unlock the locked byte}
        END;
      ELSE RangeLockPresent := FALSE; {Range was not locked}
```

Luckily, `_PBLockRange` and `_PBUnlockRange` work when used on files that are on a shared volume (i.e., AppleShare) where the possibility of shared access usually comes up much more often. When `_PBLockRange` and `_PBUnlockRange` are used on a file on a shared volume, both calls are translated into an AppleTalk Filing Protocol (AFP) call, `FPByteRangeLock`. The `FPByteRangeLock` call locks or unlocks a specified range of bytes within an open fork.

Note:
With System Software 7.0, file sharing can be started or stopped with the Sharing Setup Control Panel while your application is running. If file sharing is stopped, AppleTalk AFP specific attributes on local volumes, including locked ranges, become invalid. The `bHasPersonalAccessPrivileges` bit in the `vMAttrib` longword returned by `_PBHGetVolParms` can be used to check the current state of file sharing on local volumes.

Back to top

One Way To Lock, Two Ways To Unlock

The File Manager only gives one way to lock a range of bytes of a file, the `_PBLockRange` routine. However, there are two ways to unlock a locked range of bytes of a file. The obvious method of unlocking a range of bytes is the `_PBUnlockRange` routine; the not so obvious method is to close the file. When a file is closed, all locked ranges held by a user of the file are unlocked.

The range of bytes unlocked by `_PBUnlockRange` must be the exact same range locked by a `_PBLockRange` call; you cannot unlock a portion of a locked range nor can you unlock portions of a file that are not locked. If, for some reason, you need to unlock a range you have locked and you do not know where the range started or how long the range is, you must close the file to unlock the range.

Back to top

What Range Did I Just Lock?

A range of bytes may be locked relative to the beginning or the end of the file, however AFP's `FPByteRangeLock` call only lets you unlock a range of bytes relative to the beginning of the file. If a range of bytes is successfully locked with the `FPByteRangeLock` call, then a reply packet containing `RangeStart`, the number of the first byte of the range just locked (i.e., the offset from the beginning of the fork) is returned to `FPByteRangeLock`. That offset allows an application to set a lock when the application does not know the exact end of file (which can often happen when sharing a file) and later know what range to unlock.

This brings us to an interesting fact: `_PBLockRange` does not return the number of the first byte of the range just locked to your application. The reply packet containing `RangeStart` is returned by `FPByteRangeLock`, but `_PBLockRange` does not return it to your application. That would make it impossible to really know what range you just locked if the range locked were relative to the end of file. However, the Macintosh does not use the file server's end of file. Instead, the file's logical end of file is retrieved from the server when the file is opened and that local copy of end of file is used whenever the `ioPosMode` field is set to `fsFromLEOF` (the current logical end of file can be retrieved with `_PBGetEOF`).

Back to top

An Off By EOF Bug

With Macintosh System Software 6.0.7 and earlier, the `_PBLockRange` call's parameters are incorrectly translated to the parameters passed to the AFP `FPByteRangeLock` call. When `ioPosMode` field is set to `fsFromLEOF` the offset parameter sent is equal to the offset supplied to the `_PBLockRange` call plus what your system locally thinks is the current EOF. This would work correctly except that the `Start/EndFlag` in the `FPByteRangeLock` call's parameter list is set to `End` (relative to end of file) which means that the range locked is EOF bytes further out than that for which you asked.

The problem can be avoided under System Software 6.0.7 and earlier by using `fsFromStart`, `fsAtMark`, or `fsFromMark` for the `ioPosMode` field. System Software 7.0 fixes the bug by setting the `Start/EndFlag` in the `FPByteRangeLock` call's parameter list to `Start` (relative to beginning of file) when `ioPosMode` is set to `fsFromLEOF`.

Back to top

Summary

Remember, `_PBLockRange` does not do anything on local unshared volumes, and if you use `_PBLockRange` under Macintosh System Software 6.0.7 or earlier, never set `ioPosMode` to `fsFromLEOF`.

Back to top

References

Inside Macintosh , Volumes IV & VI, The File Manager

Inside Macintosh , Volume V, File Manager Extensions In a Shared Environment

Inside AppleTalk , AppleTalk Filing Protocol

Back to top

Downloadables

	Acrobat version of this Note (K)	Download

Back to top