Cooperative Record Locking with AmigaDOS

by Ewout Walraven

Cooperative record locking was introduced to the Amiga's ROM file system and RAM-handler (RAM:) in Release 2. The Amiga's cooperative record locking scheme allows an application to lock regions of a file rather than locking an entire file. These regions are known as records.

The Amiga thinks of a record in terms of size and offset. The size refers to the length of the record in bytes. The offset refers to the starting position of the record from the beginning of the file. The size and offset can vary from record to record. The only limit on a record's size and offset are the limits the file system has on the size of the file.

In Release 2, there are two basic types of record lock, exclusive and shared. An exclusive lock gives an application exclusive access to a record. While an application holds an exclusive record lock, no other process can obtain a record lock that overlaps the exclusively locked region. The file system can only grant an exclusive record lock on a region that is not part of an existing record lock.

The other type of record lock, the shared lock, gives an application shared access to a record. While an application holds a shared record lock, other applications can also obtain a shared record lock that overlaps the original shared record lock, but no other process can obtain an exclusive record lock that overlaps the shared record lock.

To lock (or unlock) a record within a file, an application needs a valid file handle on the file. Two DOS Library functions handle individual record locks:

```
BOOL LockRecord(BPTR myFH, ULONG my_offset, ULONG my_length,
                ULONG my_mode, ULONG my_timeout);

BOOL UnLockRecord(BPTR myFH, ULONG my_offset, ULONG my_length);
```

In both functions, the myFH field refers to the valid file handle mentioned above, and the my_offset and my_length fields refer to the record's offset and size.

The LockRecord() function has two additional parameters: my_mode and my_timeout. The my_mode field refers to the type of record. Release 2 supports four record types:

REC_EXCLUSIVE	Create an exclusive record lock. If the record is not immediately available, the file system will make a second record lock attempt when the timeout expires. The timeout (the my_timeout field from the RecordLock() prototype above) is in DOS ticks (1/50 of a second).
REC_EXCLUSIVE_IMMED	This record type is like REC_EXCLUSIVE, but the attempt to lock a record will fail if the record is not immediately available. In this case the file system ignores the timeout.
REC_SHARED	Create a shared record lock. If the record is not immediately available, the file system will make a second record lock attempt when the timeout expires. The timeout (the my_timeout field from the RecordLock() prototype above) is in DOS ticks (1/50th of a second).
REC_SHARED_IMMED	This record type is like REC_SHARED, but the attempt to lock a record will fail if the record is not immediately available. In this case the file system ignores the timeout.

The Amiga record locking scheme is cooperative--the file system does not prevent any process from accessing any part of a data file. The record locking scheme has nothing to do with other actions of a file system. The file system will let other processes read and write a data file regardless of any existing record locks on the data file.

Also, when an application attempts to create a record lock, the file system's record locking mechanism only makes sure the record won't conflict with any existing record locks on the same data file. The file system doesn't check the validity of the locked region. This makes it possible to lock records in an empty data file and also to lock records that are well beyond the end of a data file. This feature may be useful to an application that needs to lock records that don't exist yet.

When releasing a record lock, an application must provide exactly the same parameters it used when it locked the record. Attempting to unlock a record using a different offset or length will fail. If the application does not successfully unlock a record, the record lock will remain in effect. This means further requests to lock that record can fail, depending on the type of record lock. The record lock will remain in effect until the data file is removed or the system restarted. As with most Amiga resources, an application should release a record lock as quickly as possible. This helps out other applications that might be waiting to access the locked record.

Locking Multiple Records

DOS Library offers a function to lock an array of record locks:

```
BOOL LockRecords(struct RecordLock *recordarray, ULONG multi_timeout);
```

LockRecords() locks a group of records using one function call. LockRecords() accepts a pointer to an array of RecordLock structures (as defined in <*dos/record.h*>):

```
struct RecordLock {
     BPTR rec_FH;   /* The file handle of the data file */
     ULONGrec_Offset;/* The record offset in the data file */
     ULONGrec_Length;/* The length of the record */
     ULONGrec_Mode; /* The mode of the record lock */
};
```

The fields in each RecordLock structure correspond to the parameters from the LockRecord() function. The records do not have to be in the same file. The array is terminated by a dummy RecordLock with a `NULL` file handle (rec_FH).

The RecordLock structure does not include a timeout. Instead, LockRecords() applies the same timeout (multi_timeout from the LockRecords() prototype above) to each RecordLock in the array. If LockRecords() fails to lock any of the records in its array, LockRecords() releases any successful record locks from the array, and return `DOSFALSE`.

To unlock records locked by LockRecords(), use the dos.library function UnLockRecords():

```
BOOL UnLockRecords(struct RecordLock *recordarray);
```

This function accepts the same array used to lock the records with LockRecords().

Note that it is possible to use UnLockRecord() on a record locked by LockRecords(). However, if an application uses UnLockRecord() to unlock one record in the RecordLock array, it should use UnLockRecord() to unlock all of the records in the array.

Locking Records Using DOS Packets

To lock a record using the DOS packet interface, send an ACTION_LOCK_RECORD (2008) packet to the file system. This packet uses the following arguments:

```
ARG1: BTPR   The file handle of the data file in which you wish to lock a record.
ARG2: ULONG  The offset (in bytes) in the file of the record.
ARG3: ULONG  The length (in bytes) of the record.
ARG4: ULONG  The mode with which you wish to lock record.
ARG5: ULONG  Time (in ticks) you are will to wait for the record to become available.

RES1: BOOL   Upon return RES1 will contain the success/failure status:
               DOSTRUE if the record lock was successfully locked.
               DOSFALSE if the record lock failed.
RES2: CODE   Failure code if the record lock failed for a reason other than denied
               access (collision for example).
```

To unlock a record using the DOS packet interface, send an ACTION_FREE_RECORD (2009) packet. This packet uses the following arguments:

```
ARG1: BPTR   The file handle of the data file in which you locked the record.
ARG2: ULONG  The offset (in bytes) in the file of the record with which you locked
               the record.
ARG3: ULONG  The length (in bytes) of the record with which you locked the record.

RES1: BOOL   Upon return RES1 will contain the success/failure status:
               DOSTRUE if the record lock was successfully unlocked.
               DOSFALSE if there was no lock to unlock.
RES2: CODE   Possible failure code if the record lock could not be unlocked.
```