QuickDOS

COLLABORATORS			
	TITLE : QuickDOS		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		November 28, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

QuickDOS

1.1 QuickDOS

TABLE OF CONTENTS

1.2 QuickDOS (QuickDOS)

QuickDOS V1.00 -- Written by Alexis WILKE (c) 1994

The memory increasement of every system enables a disk speed-up by caching files in that memory. Then no time is spent to retreive those data from the disk, but a simple copy or better a returned pointer on the bufferized file.

QuickDOS© is build for programmers who would like to make their programs a little bit quicker not modifing their code too much. QuickDOS supports most of important AmigaDOS commands for file access (Thoses are listed below.) Even some calls might be a little bit different, it can be used exactly the same way. This means you might copy the AmigaDOS base pointer to

replace the missing QuickDOS one.

QuickDOS© must be called from a process and each process will have its own OpenLibrary() call. If you do not follow those two rules, QuickDOS© will fail.

Like the DOS functions, when a call successed -1 is returned else 0, except when specified as different. Anyway the value -1 should not be checked while some functions might return a pointer rather than -1 (Then check zero!)
To check a function result you should use the following code:

```
if(Examine(file, 0L) != 0) {
  /* Hourra! It works! */
  ...
}
else {
  /* Ho ch... it fails! */
  ...
}
```

A null file handle will usuly generates the error:

```
ERROR_INVALID_LOCK
```

It is similar to AmigaDOS for the following calls:

```
QDClose
QDExamine
QDIoErr
QDOpen
QDRead
QDSeek
QDWrite
```

Those functions actually return D0 and D1 equal but this may change in the future. The library pointer must be in A6 in order to work properly.

An internal device called 'QuickDOS:' is build into QuickDOS library to permit any one to read/write files in memory not using any physical device (Also no AmigaDOS device.) This may be used to share large amount of information between several tasks.

QuickDOS creates a secondary process to execute the low DOS accesses (Open/Close/Read/Write with AmigaDOS.) That process has a priority of 5. If you create a program with a higher priority, you will not see effective faster loading operations. There is actually no way to change the priority of that process through QuickDOS.library. Because some messages are sent to that process, a free signal is needed when a call is done to any of the QuickDOS© function. When no signal is available the ERROR_NO_FREE_STORE error occurs.

```
  Note: a task which did not open (or which already closed the
  library,) has no right to use it and any call will fail. This
  is the same for an handle. Handles can not be shared between
  tasks (but this does not matter because it is so quick to
  open a file...)

  QuickDOS© is a copyright of Alexis WILKE March 1994.
  All rights reserved.
```

1.3 QuickDOS (QDClose)

```
quickdos.library_QDClose                                quickdos.library/QDClose

   NAME
  QDClose -- Closes a QuickDOS file

   SYNOPSIS
  success = QDClose(file)
  D0      D1

  long QDClose(struct QDFileHandle *);

   FUNCTION
  This function will close a QuickDOS file. This will free the
  handle associated with the file but not the file information.
  To be able to retreive the file information again you will
  have to reopen the file.
  No more access can be done with this file handle.

  If you want the file to be completly discarded from memory
  you must use QDDiscard after the closure.

  A file opened with MODE_NEWFILE or MODE_READWRITE will be
  written on the disk before to be closed if necessary.
  The AmigaDOS file is closed when no more access is granted
  to a file. Note that the closure may happen at a latter
  time, until the flush take place.

  You may call this function with a null handle.

  Note: the success of this function does not means that the
  file is successfully written to the disk.

   INPUTS
  file - the file handle returned by QDOpen

   RETURN
  success - DOSTRUE if the file has been closed. In any case
    you cannot use the file handle any more and
    QuickDOS will take care of the closure at a later
    time.
```

```
   BUGS

   SEE ALSO
   QDCloseAll ,  QDDiscard ,  QDForget
```

1.4 QuickDOS (QDCloseAll)

```
quickdos.library_QDCloseAll                    quickdos.library/QDCloseAll

   NAME
  QDCloseAll -- Closes all QuickDOS file of your task

   SYNOPSIS
  success = QDCloseAll()
  D0

  long QDCloseAll(void);

   FUNCTION
  QuickDOS keep a track of all the opened files for each task.
  Then a task can close all its files at a time. This function
  invalidates all your file handles.
  This function is called by the system when you close the
  library. You also can avoid a call to it or a call to QDClose()
  for each of the opened files.
  Files remains in memory, only file handles are freed.

  Note: the success of this function does not means that the
  files are successfully written to the disk.

   INPUTS
  None

   RETURN
  success - DOSTRUE when nothing went wrong
    In any case you cannot use any of the file handle.
    QuickDOS will take care of the closure at a later
    time.
   BUGS

   SEE ALSO
   QDClose ,  QDDiscard ,  QDForget
```

1.5 QuickDOS (QDDiscard)

```
quickdos.library_QDDiscard                      quickdos.library/QDDiscard

   NAME
  QDDiscard -- Try to discard a file from memory
```

```
   SYNOPSIS
  success = QDDiscard(name)
  D0        D1

  long QDDiscard(char *);

   FUNCTION
  Try to discard a file. The given name must be a valid file
  name. This function will be valid only if absolutly no task
  is currently accessing the file.

  The file will be freed from memory and also the memory
  will be available to you.

  A call to the discard function right after a file closure
  usuly wont be successful.

   INPUTS
  name - the file name (C terminated string)

   RETURN
  success - DOSTRUE when the file is discarded

   BUGS

   SEE ALSO
   QDClose ,  QDCloseAll
```

1.6 QuickDOS (QDExamine)

```
quickdos.library_QDExamine                          quickdos.library/QDExamine

   NAME
  QDExamine -- Examines a file

   SYNOPSIS
  fib = QDExamine(file, buffer)
  D0/A0   D1    D2

  struct FileInfoBlock * QDExamine(struct QDFileHandle *,
            struct FileInfoBlock *);

   FUNCTION
  This examine is a little bit different from the one of
  AmigaDOS. AmigaDOS will each time read the disk to return
  the file informations, this function make a copy of
  those informations which are in memory. This means you
  may receive old informations if you used QDOF_SHARED
  flag calling QDOpen().

  QuickDOS has a copy of the FileInfoBlock in memory. Then,
  if you do not need to modify it, you can ask for the
  QuickDOS buffer pointer giving a null pointer in the
  parameter list. This buffer pointer is no more valid when
```

```
  the file is closed.
  Otherwise you must pass a long word aligned pointer.
  If 'buffer' is not long word aligned ERROR_INVALID_LOCK
  occur or null is returned.

  The buffer will be actualized if another task open the
  file.

   INPUTS
  file - the file handle returned by QDOpen
  buffer - a pointer to a FileInfoBlock structure
    or null to retreive the QuickDOS pointer

   RETURN
  fib - a 'like AmigaDOS' file info block
    or null when an error occur

   BUGS
  The buffer must be long word aligned. This is keept
  to make sure you will have an AmigaDOS compatible
  function (BPCL oblige.)

   SEE ALSO
   QDInfo ,  QDExNext
```

1.7 QuickDOS (QDExNext)

```
quickdos.library_QDExNext                              quickdos.library/QDExNext

   NAME
  QDExNext -- Search next file

   SYNOPSIS
  qdfile = QDExNext(file)
  D0      D1

  struct QDFile * QDExNext(struct QDFile *);

   FUNCTION
  This function searchs for the next available file which
  stands in memory. To look for the first existing file,
  use null for the file handle.

  This is not similar to the AmigaDOS function.

  The structure information should be used only for the file name.
  If more information is needed, a call to the proper functions
  should be done (QDOpen() and QDInfo() or QDExamine().)

  The end of list is known because this function will return
  zero and an error of type 'ERROR_NO_MORE_ENTRIES' (The error
  code has to be checked with QDIoErr() to ensure the proper
  action.)
```

```
   INPUTS
  file - the file structure returned by ExFirst() or ExInfo()

   RETURN
  qdfile - a QuickDOS file structure
    or null when an error occur
    (reach the end is considered as an error)

   BUGS
  No protection against multi-tasking being is provided, then a
  call to this function should be between a Forbid() and Permit()
  calls, without access only to Exec low level functions (Like
  memory allocation or so.)

   SEE ALSO
   QDExamine ,  QDInfo ,  QDOpen
```

1.8 QuickDOS (QDFlush)

```
quickdos.library_QDFlush                              quickdos.library/QDFlush

   NAME
  QDFlush -- Flushes the contains of the file buffer

   SYNOPSIS
  success = QDFlush(file)
  D0      D1

  long QDFlush(struct QDFileHandle *);

   FUNCTION
  This function flushes the given file. Then the effective modifications
  will be written back on the disk. The smaller buffer than possible
  will be written.

  Note: the success of this function does not means the file is
  successfully written on the disk.

   INPUTS
  file - the file handle returned by QDOpen

   RETURN
  success - DOSTRUE when nothing goes wrong

   BUGS

   SEE ALSO
```

1.9 QuickDOS (QDForget)

```
quickdos.library_QDForget                          quickdos.library/QDForget

   NAME
  QDForget -- Closes a file but do not save it.

   SYNOPSIS
  success = QDForget(file)
  D0       D1

  long QDForget(struct QDFileHandle *);

   FUNCTION
  This function is similar to QDClose() in that it close and prohibid
  any further access to the given file. But QDForget() will not make
  the written information effective, which means it will not be
  copied to the AmigaDOS disk. If several tasks was using this
  file, you may anyway have the result on the disk because another
  task saved it.

  This function should be used with care, and the usage of 'QuickDOS:'
  device is adviced.

  You may call this function with a null pointer.

  Note: the file may be forgotten but remains empty on the disk.
  This happen when the file was opened without the QDOF_MEMORY flag.

   INPUTS
  file - the file handle returned by QDOpen()

   RETURN
  success - DOSTRUE if the file has been closed in any case
    you cannot use the file handle any more and
    QuickDOS will take care of the closure at a later
    time.

   BUGS

   SEE ALSO
   QDClose ,  QDCloseAll
```

1.10 QuickDOS (QDInfo)

```
quickdos.library_QDInfo                            quickdos.library/QDInfo

   NAME
  QDInfo -- Returns the QuickDOS file structure

   SYNOPSIS
  qdfile = QDInfo(file)
  D0/A0   D1

  struct QDFile * QDInfo(struct QDFileHandle *);
```

```
   FUNCTION
  The function returns the pointer on the QuickDOS file structure.
  That structure can ONLY be READ. This structure will remains
  valid until the file closure.

  There are the most important fields:

  QDF_Data  is this file data pointer
  QDF_Size  is the size of valid data into the buffer

  If the file was opened with QDOF_SHARED flag, it may change
  at anytime.

  This function can be used instead of a QDRead() as a really
  quick read function.

   INPUTS
  file - the file handle returned by QDOpen

   RETURN
  qdfile - the QuickDOS file structure like described in quickdos.i/h
    or null

   BUGS

   SEE ALSO
   QDExamine
```

1.11 QuickDOS (QDIoErr)

```
quickdos.library_QDIoErr                              quickdos.library/QDIoErr

   NAME
  QDIoErr -- Returns the last QuickDOS error for this task

   SYNOPSIS
  error = QDIoErr()
  D0

  long QDIoErr(void);

   FUNCTION
  This function returns the last error which occurs in QuickDOS.
  It can be called several times, the error is not erased after
  the first call.
  A function call, except this one, will always clear the
  current error when no error occur.

   INPUTS
  None

   RETURN
  error - last error number (Like AmigaDOS)
```

```
    or null

   BUGS

   SEE ALSO
```

1.12 QuickDOS (QDOpen)

```
quickdos.library_QDOpen                                    quickdos.library/QDOpen

   NAME
  QDOpen -- Opens a file

   SYNOPSIS
  file = QDOpen(name, mode)
  D0        D1    D2

  struct QDFileHandle * QDOpen(char *, long);

   FUNCTION
  This function opens a file like DOS would do. It creates a new
  file handle and reads the specified file when enough time is
  given for that purpose and the file is opened with one of the
  following modes:

    MODE_OLDFILE
    MODE_READWRITE

  The file just stands in memory when opened as a new file with
  the mode:

    MODE_NEWFILE

  This function can also be used like the AmigaDOS function and
  with some flags. The mode include 16 flags in bits 16 to 31.
  If you want your code to remains compatible, no flag can be
  specified, otherwise there is a list:

  With MODE_READWRITE or MODE_NEWFILE:
    QDOF_MEMORY Open the file only in memory.

  With MODE_READWRITE or MODE_NEWFILE:
    QDOF_SHARED Do not keep an AmigaDOS file handle
        then other tasks can access the file
        with a read or a write access.

  With MODE_READWRITE:
    QDOF_APPEND Write only behind the already existing
        data. QuickDOS will forbid any write
        function in existing data.

   INPUTS
  file -     the file handle returned by QDOpen
  buffer -   a pointer to a buffer
```

```
  length -   length of the data to be retreived
    (length of the given buffer or less)

   RETURN
  file - a new file handle (not AmigaDOS compatible at all)
    or null

   BUGS

   SEE ALSO
   QDWrite ,  QDSeek
```

1.13 QuickDOS (QDRead)

```
quickdos.library_QDRead                                quickdos.library/QDRead

   NAME
  QDRead -- Reads the file contains at current position

   SYNOPSIS
  size = QDRead(file, buffer, length)
  D0        D1    D2      D3

  long QDRead(struct QDFileHandle *, void *, long);

   FUNCTION
  This function copies the file contains into the specified
  buffer. This file must have been opened with any mode
  except NEWFILE. The operation starts at the current position
  and will continue at most up to the end of file.

  You may change the current position with the QDSeek()
  function.

  The returned value is the size of the effectivly read
  data.

   INPUTS
  file -     the file handle returned by QDOpen()
  buffer -   a pointer to a buffer
  length -   length of the data to be retreived
    (length of the given buffer or less)

   RETURN
  size - the read size
    null when the end of file has been reached
    or -1 when an error occur (You can check QDIoErr()
    for more informations)

   BUGS

   SEE ALSO
   QDWrite ,  QDSeek
```

1.14 QuickDOS (QDReallocStep)

```
quickdos.library_QDReallocStep                  quickdos.library/QDReallocStep

   NAME
  QDRallocStep -- Modifies the reallocation length

   SYNOPSIS
  oldstep = QDRallocStep(step)
  D0           D1

  long QDRallocStep(long);

   FUNCTION
  The file buffers are enlarged and copied each time the limit of the
  current buffer is reached. This means a lot of wasting time. If you
  are working with large length with QDWrite command, you should
  use a larger value here. The default size is 16Ko.

  The size will always be a multiple of 8 and also 8 at least. Use
  the special value 0 to simply know about the current size.

  This value is only used with files opened with NEWFILE or READWRITE
  modes.

  Note: a too large value will just enable QuickDOS to allocates all
  your memory and generates - not enough memory - errors.

   INPUTS
  file -     the file handle returned by QDOpen

   RETURN
  oldstep - the step in use before this redefinition
    if zero is returned, an error has occured

   BUGS
  This size is shared between all tasks.

   SEE ALSO
   QDWrite
```

1.15 QuickDOS (QDSeek)

```
quickdos.library_QDSeek                         quickdos.library/QDSeek

   NAME
  QDSeek -- Defines the file current position

   SYNOPSIS
```

```
  oldposition = QDSeek(file, position, mode)
  D0          D1    D2      D3

  long QDSeek(struct QDFileHandle *, long, long);

   FUNCTION
  This function tries to reach the specified position. The modes
  are similar to thoses of AmigaDOS:

    OFFSET_BEGINNING
    OFFSET_CURRENT
    OFFSET_END

  The returned value is the current position from the beginning of
  the file before the seek action or -1 when an error occured.

  If the given position is unreachable, the current position is not
  modified.

  You can not write at the beginning of a READWRITE file
  which was opened with the QDOF_APPEND flag.

  Note: to know about a seek error, you must check the
  QDIoErr() after each call to QDSeek() to keep a compatibility
  with any AmigaDOS.

   INPUTS
  file -     the file handle returned by QDOpen()
  position - the new position to reach
  mode -     the seek mode like in AmigaDOS

   RETURN
  oldposition - the previous position
    or -1 when an error occured

   BUGS

   SEE ALSO
   QDRead ,  QDTell ,  QDWrite
```

1.16 QuickDOS (QDSetSize)

```
quickdos.library_QDSetSize                          quickdos.library/QDSetSize

   NAME
  QDSetSize -- Defines a new file length

   SYNOPSIS
  oldsize = QDSetSize(file, length)
  D0        D1    D2

  long SetSize(struct QDFileHandle *, long);

   FUNCTION
```

```
  This function tries to fix the file size to the given value.
  It returns the previous size. If the new file size is larger,
  then the buffer is filled with zeroes. The current position is
  modified as required to never stand outside of the file buffer.

  This function is only valid on files opened with: MODE_NEWFILE
  and MODE_READWRITE.

  Note: when a size of zero is returned you should check if
  an error occured. Note that this function is entirely
  different from the one in AmigaDOS.

   INPUTS
  file -     the file handle returned by QDOpen()
  length -   the new file size

   RETURN
  oldsize - returns the previous size
    or zero when an error occured

   BUGS

   SEE ALSO
   QDSeek ,  QDTell
```

1.17 QuickDOS (QDTell)

```
quickdos.library_QDTell                                quickdos.library/QDTell

   NAME
  QDTell -- Returns the current file length

   SYNOPSIS
  size = QDTell(file)
  D0        D1

  long QDTell(struct QDFileHandle *);

   FUNCTION
  This function returns the size of the specified file. This is
  useful and quicker than a double QDSeek() call.

   INPUTS
  file -     the file handle returned by QDOpen

   RETURN
  size - the current file size
    or -1 when an error occur (You can check QDIoErr()
    for more informations)

   BUGS

   SEE ALSO
   QDSeek ,  QDSetSize
```

1.18 QuickDOS (QDWrite)

```
quickdos.library_QDWrite                                quickdos.library/QDWrite

   NAME
  QDWrite -- Writes a buffer into the file

   SYNOPSIS
  size = QDWrite(file, buffer, length)
  D0          D1    D2      D3

  long QDWrite(struct QDFileHandle *, void *, long);

   FUNCTION
  This function writes the given buffer into the file. The write is
  effective only in memory. To make the write effective on the disk
  a call to QDFlush is needed. QDFlush is automatically called
  when the file is closed.

  The buffer is written at the current position and the file
  might be enlarged if requiered. If the size (the returned value)
  is smaller than the length, then a memory error probably occured.

  Note: write at the beginning of a file opened with the mode
  MODE_READWRITE and the flag QDOF_APPEND will simulate a
  write command, but the write will not be effective.

   INPUTS
  file -      the file handle returned by QDOpen()
  buffer -    a pointer on the buffer to be written
  length -    the size of the data to be written in the file

   RETURN
  size - the written size
    or -1 when an error occur (You can check QDIoErr()
    for more informations)

   BUGS

   SEE ALSO
   QDRead ,  QDSeek
```