Directory

Paul Manias

COLLABORATORS			
	TITLE : Directory		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Paul Manias	May 28, 2025	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Directory

1.1 Object: Directory

```
OBJECT DOCUMENTATION
Name:      DIRECTORY
Version:   0.9 Beta
Date:      December 1997
Author:    Paul Manias
Copyright: DreamWorld Productions, 1996-1997.  All rights reserved.
Notes:     This document is in the process of being written.

CHANGES VERSION 0.9B
Added:   Directory->ChildDir
         Directory->ChildFile
         Directory->Next
         Directory->Prev

Edited:  Directory->Comment
         Directory->Date

Renamed: Directory->UserFlags to Directory->Permissions
```

1.2 Object: Directory

```
OBJECT
Name:    Directory
Module:  Files
Version: 1
Type:    Complex

DESCRIPTION
This object is used for file and directory management purposes.

FUNCTIONS
Some functions supporting the Directory object are:

     WriteComment()
     WriteDate()
```

```
ACTIONS
The Directory object supports the following actions:

     Activate() Get a directory listing.
     Free()     Free a Directory object.
     Get()      Get a new Directory structure.
     Init()     Initialise a Directory.

STRUCTURE
The Directory structure consists of the following public fields:

     ChildDir    Pointer to first sub-directory in list.
     ChildFile   Pointer to first sub-file list.
     Comment     Comment for the directory, if any.
     Date        Pointer to Time of last date stamping.
     DirList     List of directories under this one.
     FileList    List of files in this directory.
     Flags       Directory flags and options.
     Next        Next directory in list.
     Prev        Previous directory in list.
     Source      Pointer to Directory source.
     Permissions Permission flags, eg RWED.
```

1.3 Object: Directory

```
FIELD
Name:        ChildDir
Type:        struct Directory *
Inheritance: Activate()
Status:      Read Only.

DESCRIPTION
This field points to the first structure of a linked-list of
sub-directories for your master Directory. This list is only built if you
call the Activate() function. The child structures will be Query()'d, so
they will contain Date, Comment information and so on. You may treat them
as normal objects and call Init() and Activate() on each child object to
start building complex tree structures if you wish.

Note that when you free your Directory object, all child directories on the
linked-list will be freed. If you want to keep a child directory, you may
use the Detach() action to do this. After you have extracted your object
you will be responsible for Free()ing it.

If there are no sub-directories then this field will be set to NULL after
an Activate() or Query().

SEE ALSO
Fields: ChildFile
        Next
        Prev
```

1.4 Object: Directory

```
FIELD
Name:        ChildFile
Type:        struct File *
Inheritance: Activate()
Status:      Read/Write

DESCRIPTION
This  field points to the first structure of a linked-list of sub-files for
your  master Directory.  This list is only built if you call the Activate()
function.  The  child  structures will be Query()'d, so will contain Date,
Comment  information  and  so on.  You may treat them as normal objects and
use the various action functions for files.

Note  that  when  you  free  your  Directory object, all child files on the
linked-list will be freed.  If you want to keep a child File, you may
use the Detach() action to do this.  After you have extracted your object
you will be responsible for Free()ing it.

If  there  are  no  sub-files  then this field will be set to NULL after an
Activate() or Query().

SEE ALSO
Fields: ChildDir
        Next
        Prev
```

1.5 Object: Directory

```
FIELD
Name:        Comment
Type:        BYTE *
Inheritance: Obtained from the existing directory, if possible.
To Change:   SetFileComment()
Status:      Read/IWrite

DESCRIPTION
This  field  points  to  a  string that specifies the user commment for the
Directory.  Although the Directory object supports comments of an unlimited
length,  the  file-system  itself  will have a cut-off point on the comment
length.  It would be unwise to assume support for anything longer than 128
bytes.  If when setting a new Comment you go over the limit, the extraneous
characters will be ignored.

NOTE
If  you  initialise  an  existing  Directory and specify a Comment, the old
comment (in the file system) will be over-written with your new one.
```

1.6 Object: Directory

```
FIELD
Name:        Date
Type:        struct Time *
Inheritance: Obtained from the existing directory, if possible.
To Change:   SetFileDate()
Status:      Read/IWrite

DESCRIPTION
This  field points to a Time object, which specifies the date on which this
Directory was last stamped.  This is usually considered to be the last date
on which the Directory was changed, but this does not always have to be the
case.

NOTE
If  you  initialise an existing directory with a Date pointer, the old Date
will be over-written with your new one.
```

1.7 Object: Directory

```
FIELD
Name:      Flags
Type:      LONG
On Change: Cannot change dynamically.
Status:    Read/IWrite.

DESCRIPTION
Specifies  the  Flags  to use when opening the Directory.  Here are the
flags you can specify for OpenFile() and Init():

FL_LOCK
Setting  this  will  lock  the  directory  for  exclusive access - no other
process will be able to open the directory while you are using it.  If this
flag  is  not  set  then  the  directory  will  be  open for shared access.
Attempting  to  set a lock on a directory that is already in use by another
Task  results  in  failure.  Keep  in  mind  that the effects of locking a
Directory will not flow onto any child directories.

FL_FIND
This  flag  allows  OpenFile() to use some intelligence and try to find the
directory  if  it  is  not  immediately located at the given FileName.  The
process  involves  a  simple  but  effective  tree search.  This feature is
limited to localised searching, no attempt will be made to do a tree search
of all directories in an assignment (a lengthy process).

FL_NOBUFFER
Setting  this flag prevents the directory data from being put in the cache.
You  may  want to do this if it is imperative that the directory physically
reflects its data at all points in time.
```

1.8 Object: Directory

```
FIELD
Name:        Next
Type:        struct Directory *
Inheritance: Activate()
Status:      Read Only.

DESCRIPTION
This field only applies when a Directory object forms part of a linked-list
of directories.  It points directly to the next Directory in the chain.  If
the value is NULL, then this is the last link.

SEE ALSO
Fields: ChildDir
        Prev
```

1.9 Object: Directory

```
FIELD
Name:        Permissions
Type:        LONG
Inheritance: Init()
Default:     FPF_READ|FPF_WRITE|FPF_DELETE
Status:      Read/IWrite

DESCRIPTION
The permissions field gives information on the user rights for a particular
file or directory.  Flags currently available are:

FPF_HIDDEN
If  the  directory  should  be hidden from this user then this flag will be
set.

FPF_DELETE
If  the  user  has  delete rights for this directory then this flag will be
set.   Note  that if there are sub files/directories that the user does not
have  permissions  to delete, then the user will be prevented from deleting
the entire directory structure.

FPF_PASSWORD
This  is a special flag that can lock a user out from a directory until the
correct  password  is  given.  If  a  directory  contains  sub  files  and
directories, access to these will also be affected by the password setting.
Note  that  support  for  this  flag  is still under-way and currently most
file-systems are not able to support it directly.

FPF_READ
This flag does not apply to directories.

FPF_SCRIPT
This flag does not apply to directories.

FPF_WRITE
This flag does not apply to directories.
```

1.10 Object: Directory

```
FIELD
Name:        Prev
Type:        struct Directory *
Inheritance: Activate()
Status:      Read/Write

DESCRIPTION
This field only applies when a directory object forms part of a linked-list
of directories.  It points directly to the previous directory in the chain.
If the value is NULL, then this is the first link.

SEE ALSO
Fields: ChildDir
        Next
```

1.11 Object: Directory

```
FIELD
Name:      Source
Type:      struct FileName *
On Change: Cannot change dynamically.
Status:    Read/IWrite.

DESCRIPTION
Points  to  a  source  object, which may be either a FileName or recognised
system object.  The Source will be used to read data from or write data to.

Note that only the FileName source type is fully supported at the moment.
```