# Sourcecode: FreeDosObject.c

| COLLABORATORS | | | |
|---|---|---|---|
| | *TITLE* :<br><br>Sourcecode: FreeDosObject.c | | |
| *ACTION* | *NAME* | *DATE* | *SIGNATURE* |
| WRITTEN BY | | February 12, 2023 | |

| REVISION HISTORY | | | |
|---|---|---|---|
| NUMBER | DATE | DESCRIPTION | NAME |
| | | | |

# Contents

# Chapter 1

# Sourcecode: FreeDosObject.c

## 1.1 FreeDosObject.c

```
/* FreeDosObject.c   V1.0   93-09-27                          */
/* ROM library: "dos.library/FreeDosObject", (All versions) */
/* Copyright 1993, Anders Bjerin, Amiga C Club               */

#include <dos/dos.h>

#include <clib/dos_protos.h>
#include <stdio.h>
#include <stdlib.h>

UBYTE *version = "$VER: FreeDosObject 1.0";

int main( int argc, char *argv[] );
int main( int argc, char *argv[] )
{
  /* Pointer to our dos object (FileInfoBlock) which we will allocate: */
  struct FileInfoBlock *my_fib;


  /* Create a dos object (FileInfoBlock structure): */
  my_fib = AllocDosObject( DOS_FIB, NULL );
  if( !my_fib )
  {
    printf( "Could not allocate the FileInfoBlock!\n" );
    exit( 20 );
  };

  /* Use the allocated object... */

  /* Deallocate the dos object: */
  FreeDosObject( DOS_FIB, my_fib );

  exit( 0 );
}


  /* "BCPL" pointer to our lock: */
  BPTR my_lock;
```

```
/* Pointer to our FileInfoBlock which we will allocate: */
struct FileInfoBlock *my_fib;

/* AmigaDOS boolean check variable: */
LONG ok;



/* 1. Lock the object we want to examine: */
my_lock = Lock( "C:Dir", SHARED_LOCK );
if( !my_lock )
{
  printf( "Could not lock the object!\n" );
  exit( 20 );
}

/* 2. Allocate a FileInfoBlock structure:  */
my_fib = (struct FileInfoBlock *)
   AllocMem( sizeof( struct FileInfoBlock ), MEMF_ANY | MEMF_CLEAR );
if( !my_fib )
{
  printf( "Not enough memory!\n" );
  UnLock( my_lock );
  exit( 21 );
};

/* 3. Examine the locked object: */
ok = Examine( my_lock, my_fib );
if( !ok )
{
  printf( "Could not examine the object!\n" );
  FreeMem( my_fib, sizeof( struct FileInfoBlock ) );
  UnLock( my_lock );
  exit( 22 );
}

/* 4. Print some info about the object: */
printf( "Name:      %s\n", my_fib->fib_FileName );

if( my_fib->fib_DirEntryType < 0 )
  printf( "Type:      File\n" );
else
  printf( "Type:      Directory or Volume\n" );

printf( "Size:      %d\n", my_fib->fib_Size );
printf( "Blocks:    %d\n", my_fib->fib_NumBlocks );
printf( "Comment:   %s\n",
  my_fib->fib_Comment[02 ? my_fib->fib_Comment : "No comment" );

printf( "Protection bits:\n" );
printf( "  Delete:  %s\n",
  my_fib->fib_Protection & FIBF_DELETE ? "On" : "Off" );

printf( "  Execute: %s\n",
  my_fib->fib_Protection & FIBF_EXECUTE ? "On" : "Off" );
```

```
  printf( "  Write:   %s\n",
    my_fib->fib_Protection & FIBF_WRITE ? "On" : "Off" );

  printf( "  Read:    %s\n",
    my_fib->fib_Protection & FIBF_READ ? "On" : "Off" );

  printf( "  Archive: %s\n",
    my_fib->fib_Protection & FIBF_ARCHIVE ? "On" : "Off" );

  printf( "  Pure:    %s\n",
    my_fib->fib_Protection & FIBF_PURE ? "On" : "Off" );

  printf( "  Script:  %s\n",
    my_fib->fib_Protection & FIBF_SCRIPT ? "On" : "Off" );

  printf( "Last changed: (Internal datestamp value)\n" );
  printf( "  Days:    %d\n", my_fib->fib_Date.ds_Days );
  printf( "  Minutes: %d\n", my_fib->fib_Date.ds_Minute );
  printf( "  Ticks:   %d\n", my_fib->fib_Date.ds_Tick );



  /* 5. Deallocate the memory we have allocated: */
  FreeMem( my_fib, sizeof( struct FileInfoBlock ) );

  /* 6. Unlock the file: */
  UnLock( my_lock );

  /* The End! */
  exit( 0 );
}
```