Sourcecode: Example1.c

COLLABORATORS

	TITLE : Sourcecode: Example1.c	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		February 12, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Sourcecode: Example1.c

1.1 Example1.c

```
/*********************************************************/
/*                                                       */
/* Amiga C Encyclopedia (ACE)           Amiga C Club (ACC) */
/* --------------------------           ------------------ */
/*                                                       */
/* Manual:  AmigaDOS                    Amiga C Club       */
/* Chapter: Advanced Routines           Tulevagen 22       */
/* File:    Example1.c                  181 41  LIDINGO    */
/* Author:  Anders Bjerin               SWEDEN             */
/* Date:    93-03-17                                     */
/* Version: 1.0                                          */
/*                                                       */
/*   Copyright 1993, Anders Bjerin - Amiga C Club (ACC)  */
/*                                                       */
/* Registered members may use this program freely in their */
/*     own commercial/noncommercial programs/articles.   */
/*                                                       */
/*********************************************************/


/* This example demonstrates how to use the Examine() function. */
/* The program needs a file, directory or volume name as the    */
/* only argument and it will print some interesting information */
/* about given the object.                                      */
/*                                                              */
/* This example can be used with all versions of the dos        */
/* library.                                                     */



/* Include the dos library definitions: */
#include <dos/dos.h>

/* Include the memory type definitions: (MEMF_ANY, MEMF_CLEAR...) */
#include <exec/memory.h>

/* Now we include the necessary function prototype files:         */
#include <clib/dos_protos.h>       /* General dos functions...    */
```

```
#include <clib/exec_protos.h>      /* System functions...          */
#include <stdio.h>                 /* Std functions [printf()...] */
#include <stdlib.h>                /* Std functions [exit()...]   */
#include <string.h>                /* Std functions [strlen()...] */



/* Set name and version number: */
UBYTE *version = "$VER: AmigaDOS/Advanced Routines/Example1 1.0";



/* Declared our own function(s): */

/* Our main function: */
int main( int argc, char *argv[] );



/* Main function: */

int main( int argc, char *argv[] )
{
  /* "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;



  /* This program needs one arguement:  */
  /* (a file, directory or volume name) */
  if( argc != 2 )
  {
    /* Wrong number of arguments! */
    printf( "Error! Wrong number of arguments!\n" );
    printf( "You must enter a file, directory or volume name.\n" );
    printf( "Example1 Name/A\n" ); /* Simple template */

    /* Exit with an error code: */
    exit( 20 );
  }



  /* 1. Try to lock the object: (Shared access is enough.) */
  my_lock = Lock( argv[ 1 ], SHARED_LOCK );

  /* Could we lock the object? */
  if( !my_lock )
  {
    /* Problems! Inform the user: */
    printf( "Could not lock the object!\n" );
```

```
  /* Exit with an error code: */
  exit( 21 );
}



/* 2. Allocate enough memory for a FileInfoBlock structure:  */
/* Since we want this example to be compatible with all dos  */
/* library versions we have to use the AllocMem() function.  */
/* If your program only should be used with dos library V37  */
/* or higher you should use the AllocDosObject() function.   */
/* (Any type of memory, preferably fast but if not available */
/* use chip, and clear it. The allocated memory will be long */
/* word aligned.)                                            */
my_fib = (struct FileInfoBlock *)
   AllocMem( sizeof( struct FileInfoBlock ), MEMF_ANY | MEMF_CLEAR );

/* Check if we have allocated the memory successfully: */
if( !my_fib )
{
  /* Problems! Inform the user: */
  printf( "Not enough memory!\n" );

  /* Unlock the object: */
  UnLock( my_lock );

  /* Exit with an error code: */
  exit( 22 );
};



/* 3. Get some information about the object we have locked: */
ok = Examine( my_lock, my_fib );

/* Any problems? */
if( !ok )
{
  /* Problems! Inform the user: */
  printf( "Could not examine the object!\n" );

  /* Deallocate the memory: */
  FreeMem( my_fib, sizeof( struct FileInfoBlock ) );

  /* Unlock the object: */
  UnLock( my_lock );

  /* Exit with an error code: */
  exit( 23 );
}



/* 4. You may now examine the FileInfoBlock structure! */
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[0] ? 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 );
}
```