

Sourcecode: Example2.c

COLLABORATORS

	<i>TITLE :</i> Sourcecode: Example2.c		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		February 12, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	Sourcecode: Example2.c	1
1.1	Example2.c	1

Chapter 1

Sourcecode: Example2.c

1.1 Example2.c

```
/******  
/*  
/* Amiga C Encyclopedia (ACE)           Amiga C Club (ACC) */  
/* -----  
/*  
/* Manual:  AmigaDOS                    Amiga C Club      */  
/* Chapter: Advanced Routines          Tulevagen 22     */  
/* File:    Example2.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 does exactly the same thing as the previous one, */  
/* it simply demonstrates how to use the Examine() function.      */  
/* However, this example uses the new AllocDosObject() and        */  
/* FreeDosObject() functions which were introduced in Release 2.  */  
/* You should use these new functions (if possible) instead of    */  
/* using the older method of allocating a fixed amount of memory  */  
/* for the dos object (the FileInfoBlock structure).              */  
/*  
/* This example can only be used with dos library V37 or higher.  */  
  
/* Include the dos library definitions: */  
#include <dos/dos.h>  
  
/* Now we include the necessary function prototype files:          */  
#include <clib/dos_protos.h> /* General dos 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/Example2 1.0";

/* Declare an external global library pointer to the Dos library: */
extern struct DosLibrary *DOSBase;

/* 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;

    /* We need dos library V37 or higher: */
    if( DOSBase->dl_lib.lib_Version < 37 )
    {
        /* The user has a dos library which is too old! */
        printf( "This program needs dos library V37 or higher!\n" );

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

    /* This program needs one argument: */
    /* (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( "Example2 Name/A\n" ); /* Simple template */
    }
}
```

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

/* 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( 22 );
}

/* 2. Create a FileInfoBlock structure with help of the      */
/* new AllocDosObject() function. Note that this function  */
/* needs dos library V37 or higher! Note also that anything */
/* we allocate with this function must be deallocated with */
/* help of the FreeDosObject() function!                    */
/* (Type FileInfoBlock structure and no Tags.)              */
my_fib = AllocDosObject( DOS_FIB, NULL );

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

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

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

/* 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 FileInfoBlock structure wich we have */
    /* created with help of the AllocDosObject() function: */
    FreeDosObject( DOS_FIB, my_fib );
}
```

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

/* Exit with an error code: */
exit( 24 );
}

/* 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 FileInfoBlock structure wich we have */
/* created with help of the AllocDosObject() function: */
/* (Remember that we must use this function if you */
/* allocated the object with help of the AllocDosObject() */
/* function. The advantage with AllocDosObject() is that */
```

```
/* the size of the object can vary between different */
/* releases of the dos library and your program will */
/* still be able to run. Since the size may vary you */
/* must of course use some function which can deallocate */
/* all memory that was allocated, and not just use a */
/* fixed size as we have to do when we use AllocMem() and */
/* FreeMem().) */
FreeDosObject( DOS_FIB, my_fib );

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

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