Sourcecode: Example4.c

COLLABORATORS

	TITLE : Sourcecode: Example4.c		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		February 12, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Sourcecode: Example4.c

1.1 Example4.c

```
/***********************************************************/
/*                                                         */
/* Amiga C Encyclopedia (ACE)           Amiga C Club (ACC) */
/* --------------------------           ------------------ */
/*                                                         */
/* Manual:  AmigaDOS                    Amiga C Club       */
/* Chapter: Files                       Tulevagen 22       */
/* File:    Example4.c                  181 41  LIDINGO    */
/* Author:  Anders Bjerin               SWEDEN             */
/* Date:    93-03-15                                       */
/* 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 program will open an already existing file and update  */
/* the values in it (we simply add 50 to each value). Since    */
/* we do not want any other program to destroy our updated     */
/* values we will lock the file exclusively while we are using */
/* it.                                                         */
/*                                                             */
/* Since we want to put an exclusive lock on an already        */
/* existing file we have to use the new "OpenFromLock()"       */
/* function to open the file once we have successfully locked  */
/* it. This example needs dos library V36 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()...]   */



/* Number of values we want to update: */
#define NUMBER_VALUES 10



/* Set name and version number: */
UBYTE *version = "$VER: AmigaDOS/InputOutput/Example4 1.0";



/* Declare an external global library pointer to the Dos library:  */
/* (Since the Dos library is always open we do not have to open it */
/* ourself. We simply declare the pointer as an external pointer   */
/* and it will automatically be initialized for us. Very handy.    */
/* We need a pointer to the Dos library so we can check which      */
/* version the user has.)                                          */
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[] )
{
  /* A "BCPL" pointer to our lock: */
  BPTR my_lock;

  /* A "BCPL" pointer to our file: */
  BPTR my_file;

  /* Store the collected numbers here: */
  int my_values[ NUMBER_VALUES ];

  /* Store here the number of bytes actually read: */
  long bytes_read;

  /* Store here the number of bytes actually written: */
  long bytes_written;

  /* A simple loop variable: */
  int loop;



  /* Check which version of the dos library the user has: (Since */
  /* this program is using the new "OpenFromLock()" function     */
  /* which was introduced in Release 2 we have to make sure that */
  /* the user really has the new dos library V36 or higher.)     */
  if( DOSBase->dl_lib.lib_Version < 36 )
  {
    /* The user has a dos library which is too old! Inform the */
    /* user and quit immediately:                               */
    printf( "Your Dos Library is too old!\n");
    printf( "This program needs V36 or higher!\n" );

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



  /* Put an exclusive lock on the file: */
  my_lock = Lock( "RAM:HighScore.dat", EXCLUSIVE_LOCK );

  /* Could we lock the file successfully? */
  if( !my_lock )
  {
    /* Problems! Inform the user: */
    printf( "Could not put an exclusive lock on the file!\n" );
    printf( "The file does not exist or is used by some one else!\n" );

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

  /* The file has now been locked: */
  printf( "The file has now an exclusive lock on it!\n" );



  /* We will now try to open the file with help */
  /* of the lock we already have:               */
  my_file = OpenFromLock( my_lock );

  /* Have we opened the file successfully? */
  if( !my_file )
  {
    /* Problems! Inform the user: */
    printf( "Error! Could not open the file!\n" );

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

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

  /* The file has now been opened: */
  printf( "File open!\n" );



  /* Load the values: */
```

```
printf( "Loading values...\n" );

/* Collect the 10 values: */
bytes_read = Read( my_file, my_values, sizeof( my_values ) );

/* Did we get all data? */
if( bytes_read != sizeof( my_values ) )
{
  /* No! We could not read all values! */
  printf( "Error! Could read all values!\n" );

  /* Close the file: */
  Close( my_file );

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

  /* Exit with an error code: */
  exit( 23 );
}
else
{
  /* OK! */
  printf( "All values were successfully collected!\n" );
}



/* We will now "update" the values: */
printf( "Updating the file...\n" );

/* We simply add 50 to each value: */
for( loop = 0; loop < NUMBER_VALUES; loop++ )
{
  printf( "Value[ %2d ]: %5d", loop, my_values[ loop ] );
  my_values[ loop ] += 50;
  printf( " -> %5d\n", my_values[ loop ] );
}

/* All value have been updated and should now be saved! */
printf( "All values have been updated!\n" );



/* We will now save the values again. To do this we have to */
/* move the file cursor to the beginning of the file so we  */
/* can overwrite the old vlues:                             */
Seek( my_file, 0, OFFSET_BEGINNING );

/* Overwrite the old values: */
bytes_written = Write( my_file, my_values, sizeof( my_values ) );

/* Did we write all data? */
if( bytes_written != sizeof( my_values ) )
{
  /* No! The numbers actually written was less */
  /* than we wanted to write!                  */
```

```
    printf( "Error! Could not save all values!\n" );

    /* Well, in this example we do not do much more about the error. */
  }
  else
  {
    /* Yes, all numbers have been written to the file! */
    printf( "All values were saved successfully!\n" );
  }



  /* Close the file: */
  Close( my_file );

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



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