Sourcecode: Example3.c

COLLABORATORS

	TITLE : Sourcecode: Example3.c		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		February 12, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Sourcecode: Example3.c

1.1 Example3.c

```
/***********************************************************/
/*                                                         */
/* Amiga C Encyclopedia (ACE)           Amiga C Club (ACC) */
/* --------------------------           ------------------ */
/*                                                         */
/* Manual:  AmigaDOS                    Amiga C Club       */
/* Chapter: Advanced Routines           Tulevagen 22       */
/* File:    Example3.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 examine all objects in */
/* directory or volume. The program needs a directory or   */
/* volume name as the only argument and it will then list  */
/* all files and directories (subdirectories) in that      */
/* directory or volume. This is a good example on how to   */
/* use the Examine() and ExNext() functions.               */
/*                                                         */
/* 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/Example3 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;



  /* 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 directory or volume name.\n" );
    printf( "Example3 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:  */
  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: */
  if( Examine( my_lock, my_fib ) )
  {
    /* 4. Check if it is a directory or volume: */
    if( my_fib->fib_DirEntryType > 0 )
    {
      /* Print out the directory/device name with underlined characters: */
      /* \033[4m : Underline */
      /* \033[0m : Normal    */
      printf( "\033[4m%s\033[0m\n", my_fib->fib_FileName );



      /* As long as we find objects we stay in the loop: */
      while( ExNext( my_lock, my_fib ) )
      {
        /* If it is a file we print out the name with white characters. */
        /* However, if it is a (sub)directory we use orange:            */
        if( my_fib->fib_DirEntryType < 0 )
          printf( "%s\n", my_fib->fib_FileName ); /* File */
        else
          printf( "\033[33m%s\033[31m\n", my_fib->fib_FileName ); /* Dir */

        /* \033[33m : Orange (Colour 3) */
        /* \033[31m : White  (Colour 1) */
      }


      /* The ExNext() function has failed. It was either an error  */
      /* or there were simply no more objects in the direcotry/    */
      /* volume. We must therefore call IoErr() to see what        */
      /* actually happened. If we get the error code:              */
      /* "ERROR_NO_MORE_ENTRIES" there were simply no more objects */
      /* to examine, else something went wrong.                    */
      if( IoErr() == ERROR_NO_MORE_ENTRIES )
        printf( "No more files!\n" );
      else
        printf("Error while reading!\n");
    }
    else
    {
      /* The user gave us a file name! We can */
      /* not list objects inside a file!      */
      printf( "%s is a file!\n", argv[1] );
      printf( "This program needs a directory or volume name!\n" );
    }
  }
  else
    printf( "Could not examine %s!\n", argv[ 1 ] );



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

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

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