Sourcecode: Example6.c

COLLABORATORS			
	TITLE : Sourcecode: Example6.c		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		February 12, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Sourcecode: Example6.c

1.1 Example6.c

```
/*********************************************************/
/*                                                       */
/* Amiga C Encyclopedia (ACE)           Amiga C Club (ACC) */
/* --------------------------           ------------------ */
/*                                                       */
/* Manual:  AmigaDOS                    Amiga C Club       */
/* Chapter: Advanced Routines           Tulevagen 22       */
/* File:    Example6.c                  181 41  LIDINGO    */
/* Author:  Anders Bjerin               SWEDEN             */
/* Date:    93-03-17                                     */
/* Version: 1.1                                          */
/*                                                       */
/*   Copyright 1993, Anders Bjerin - Amiga C Club (ACC)  */
/*                                                       */
/* Registered members may use this program freely in their */
/*     own commercial/noncommercial programs/articles.   */
/*                                                       */
/*********************************************************/

/* This example will examine some of the "lowest" parts in  */
/* AmigaDOS. It will look up and print all Assigns, Volumes */
/* and Devices AmigaDOS knows about. Please note that we    */
/* will dig fairly deep down into the system, and only      */
/* experienced programmers are recommended to do this. I    */
/* have added a lot of comments to help you, and if you cut */
/* out parts of this example carefully you should be able   */
/* to use it in your own programs.                          */
/*                                                          */
/* This example can be used with all versions of the dos    */
/* library.                                                 */



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

/* Include memory definitions: (MEMF_ANY...) */
#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()...]   */



/* Set name and version number: */
UBYTE *version = "$VER: AmigaDOS/Advanced Routines/Example6 1.1";



/* 1. Declare an external global library */
/*    pointer to the Dos library:        */
extern struct DosLibrary *DOSBase;



/* Declare our own functions: */

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

/* Prints BCPL strings: */
void PrintBSTR( BSTR string_bstr );



/* Main function: */

int main( int argc, char *argv[] )
{
  /* Temporary BCPL pointer used to convert BPTRs into C pointer: */
  BPTR temp_bptr;

  /* Pointer to the RootNode structure: */
  struct RootNode *rootnode_ptr;

  /* Pointer to a DosInfo structure: */
  struct DosInfo *dos_info_ptr;

  /* Pointer to the first DosList structure: */
  struct DosList *first_doslist_node;

  /* Pointer to the current (the one we are */
  /* working with) DosList structure:       */
  struct DosList *doslist_node;




  /* 2. Get a pointer to the RootNode structure: */
  rootnode_ptr = DOSBase->dl_Root;

  /* 3. Get a BCPL pointer (BPTR) to the DosInfo structure: */
```

```
temp_bptr = rootnode_ptr->rn_Info;

/* 4. Convert the BCPL pointer into a normal C pointer: */
/* (If I say that I hate BCPL with its acquired      */
/* pointers and strings I do not exaggerate...)      */
dos_info_ptr = (struct DosInfo *) BADDR( temp_bptr );



/* Before we may start to examine the DosInfo structure we    */
/* have to turn off the multitasking by calling the Forbid()  */
/* function. As soon as we have finished using the DosInfo    */
/* structure we must of course turn the multitaskin on again, */
/* by calling the Permit() function.                          */
/*                                                            */
/* Note that while the multitasking is OFF we must be very    */
/* careful so we do not try to wait for some external event.  */
/* If we try to wait for something to happen "outside" our    */
/* program we will sit and wait forever since nothing can     */
/* happen outside our program as long as the multitasking is  */
/* off. You must therefore NEVER use the Wait() or similar    */
/* functions after you have forbidden other programs to run.  */
/* As soon as we turn the multitasking on again, by using the */
/* Permit() function, we may of course start to wait for      */
/* external events.                                           */
/*                                                            */
/* A program that turns off the multitasking is interrupting  */
/* other programs. You must therefore try to turn the         */
/* multitaskin on again as soon as possible.                  */
/*                                                            */
/* With the new Release 2 you should actually use the special */
/* LockDosList() and NextDosEntry() functions instead of      */
/* using the Forbid() and Permit() functions. However, since  */
/* this program should run on all Amigas we stick to the old  */
/* methods. (See "Amiga DOS" chapter for more information on  */
/* the new LockDosList() and NextDosEntry() functions.)       */

/* 5. Turn the multitaskin OFF: */
Forbid();



/* 6. Scan the "DosList" nodes... */

/* Get a BCPL pointer (BPTR) to the first "DosList" node: */
temp_bptr = dos_info_ptr->di_DevInfo;

/* Convert the BPTR into a C pointer: */
first_doslist_node = (struct DosList *) BADDR( temp_bptr );

/* Start with the first node: */
doslist_node = first_doslist_node;

/* Check all nodes: */
while( doslist_node )
{
  PrintBSTR( doslist_node->dol_Name );
```

```
    printf ( " - " );

    /* Print type: */
    switch( doslist_node->dol_Type )
    {
      case DLT_DEVICE:     printf( "Device             " ); break;
      case DLT_DIRECTORY:  printf( "Assign             " ); break;
      case DLT_VOLUME:     printf( "Volume             " ); break;
      case DLT_LATE:       printf( "Late-binding Assign" ); break;
      case DLT_NONBINDING: printf( "Non-binding Assign " ); break;
      case DLT_PRIVATE:    printf( "Private node       " ); break;
      default:             printf( "Unknown type!      " );
    }
    printf ( "\n" );

    /* Get a BPTR to the next node: */
    temp_bptr = doslist_node->dol_Next;

    /* Convert the BPTR into a C pointer: */
    doslist_node = (struct DosList *) BADDR( temp_bptr );
  }

  /* 7. Turn the multitaskin ON again: */
  Permit();
}



/* Handly little function which prints BCPL strings (BSTRs): */

void PrintBSTR( BSTR string_bstr )
{
  /* Temporary string pointer */
  UBYTE *string_ptr;

  /* The length of the BCPL string: */
  UBYTE length;

  /* Simple loop variable: */
  int loop;



  /* Conver the BSTR into a normal C pointer to a BCPL string: */
  string_ptr = BADDR( string_bstr );

  /* Get the length of the BCPL string: (A BCPL string does not */
  /* contain a NULL sign in the end, but uses instead the first */
  /* byte to tell how many characters the string contains. A    */
  /* BCPL string (BSTR) can therefore not contain more than 255 */
  /* characters.                                                */
  length = string_ptr[ 0 ];

  /* Print BCPL string: */
  for( loop=1; loop <= length; loop++ )
    putchar( string_ptr[ loop ] );
```

```
}
```