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: Introduction                Tulevagen 22       */
/* File:    Example4.c                  181 41  LIDINGO    */
/* Author:  Anders Bjerin               SWEDEN             */
/* Date:    93-09-24                                       */
/* 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 contains a useful function which converts hard to */
/* use BSTR (BCPL stings) into normal easy to use C strings. This */
/* example is not directly runnable and must instead be linked    */
/* together with some other program.                              */



/* Include the dos library definitions: */
#include <dos/dosextens.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()]    */



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

```
/* Declare the function: */
void BSTRtoC
(
  BSTR string_bstr,
  UBYTE * string_c,
  int length_c
);



/* Converts a BCPL string (BSTR) into a normal C string: */
void BSTRtoC
(
  BSTR string_bstr, /* The BSTR (BCPL string)         */
  UBYTE *string_c,  /* Pointer to a normal C string   */
  int length_c      /* Maximum length of the C string */
)
{
  /* Temporary string pointer: */
  UBYTE *string_ptr;

  /* The length of the BSTR string: (A BSTR can not be   */
  /* longer than 255 characters so we can use a unsigned */
  /* byte to store the length in.)                       */
  UBYTE length_bstr;

  /* The number of characters that will be copied: */
  int length;

  /* Simple loop variable: */
  int loop;


  /* Since we have to put a NULL sign at the end of the  */
  /* C string we can only store "length_c" - 1 number of */
  /* characters. Therefore we have to reduce the length  */
  /* by one:                                             */
  length_c--;

  /* Convert the BSTR into a normal C pointer */
  /* to a BCPL string: (Are you with me?)     */
  string_ptr = (UBYTE *) BADDR( string_bstr );

  /* Get the length of the BCPL string: (A BCPL string  */
  /* does not contain a NULL sign at 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     */
  /* (0 - 255 = one byte).                              */
  length_bstr = string_ptr[ 0 ];

  /* Get the smallest value: (If the C string is smallest  */
  /* we should of course not copy more than can be fitted  */
  /* in the C string. On the other hand, if the BCPL       */
  /* string is smaller we should of course not copy more   */
  /* characters than there actually exist in the BCPL      */
  /* string. Consequently we should only use the smallest  */
  /* value: (If you have included the header file "math.h" */
  /* you can equally well use the macro "min()".)          */
  length = length_c <= length_bstr ? length_c : length_bstr;

  /* Convert the BCPL string into a C string: */
  for( loop = 1; loop <= length; loop++ )
    string_c[ loop - 1] = string_ptr[ loop ];

  /* Note that the loop starts with 1 and not 0 as normal! */
  /* The first byte in a BCPL string contains the lenght   */
  /* of the string, and since we don’t want to copy that   */
  /* value into the C string we start one byte later. We   */
  /* must then also change the "<" sign into a "<=" since  */
  /* we want to copy all characters including the last     */
  /* one since that also contains a character. Normal C    */
  /* strings ends with a NULL, but BSTRs do not.           */


  /* Finally we have to put a NULL sign at the */
  /* end of the C string:                      */
  string_c[ loop - 1] = NULL;
}
```