

```

/*****
/*
/*----- A R G S . C -----*/
/* Task : Functions for editing command line parameters */
/*-----*/
/* Author : Michael Tischer / Bruno Jennrich */
/* Developed on : 03/20/1994 */
/* Last update : 04/14/1995 */
/*****
#ifndef __ARGS_C
#define __ARGS_C

/*- Link Include Files -----*/
#include <string.h>
#include <stdlib.h>
#include "types.h"
#include "args.h"

/*****
/* GetArg : Determine command line parameters */
/*-----*/
/* Input : argc - Argument Count (s. main(INT argc, CHAR *argv[])) */
/* argv - Argument Values (s. main(INT argc, CHAR *argv[])) */
/* pPrefix - Parameter Prefix (e.g: "-o" fOr Output) */
/* iType - Type of parameter (_char, _int, _long, _string, */
/* _none = tests whether parameters exist) */
/* pVar - Address of variables which are to receive */
/* command line parameter value. */
/* iNumElements - In case several values are separated by */
/* ', ' up to iNumElements parameters */
/* are stored in the array specified in */
/* pVar. */
/* Output : Number of determined values, or 0 if there is no */
/* command parameter available. */
/*****
INT GetArg( INT argc, PCHAR argv[], PCHAR pPrefix, INT iType,
PVOID pVar, INT iNumElements )
{
    INT i, j;
    INT iLen;

    PCHAR cPtr;
    PINT iPtr;
    PLONG Ptr;
    PCHAR *sPtr;

    iLen = _fstrlen( pPrefix );

    for( i = 1; i < argc; i++ )
        if( _fstrnicmp( pPrefix, ( LPCHAR )argv[ i ], iLen ) == 0 )
            switch( iType )
            {
                case _int:
                    iPtr = (PINT)pVar;
                    for( j = 0; ( j < iNumElements ) && argv[ i ][ iLen ]; j++ )
                    {
                        *iPtr = atoi( ( PCHAR )argv[ i ][ iLen ] );
                        while( argv[ i ][ iLen ] && ( argv[ i ][ iLen ] != ',' ) )
                            iLen++;
                        if( argv[ i ][ iLen ] == ',' ) iLen++;
                        iPtr++;
                    }
                    return j;

                case _char:
                    cPtr = (PCHAR)pVar;
                    for( j = 0; ( j < iNumElements ) && argv[ i ][ iLen ]; j++ )
                    {
                        *cPtr = argv[ i ][ iLen ];
                        while( argv[ i ][ iLen ] && ( argv[ i ][ iLen ] != ',' ) )
                            iLen++;
                        if( argv[ i ][ iLen ] == ',' ) iLen++;
                        cPtr++;
                    }
                    return j;
            }
}

```

```

    case _long:
        Ptr = (PLONG)pVar;
        for( j = 0; ( j < iNumElements ) && argv[ i ][ iLen ]; j++ )
        {
            *Ptr = atol( ( PCHAR )&argv[ i ][ iLen ] );
            while( argv[ i ][ iLen ] && ( argv[ i ][ iLen ] != ',' ) )
                iLen++;
            if( argv[ i ][ iLen ] == ',' ) iLen++;
            Ptr++;
        }
        return j;

    case _string:
        sPtr = ( PCHAR *)pVar;
        for( j = 0; ( j < iNumElements ) && argv[ i ][ iLen ]; j++ )
        {
            *sPtr = &argv[ i ][ iLen ];
            while( argv[ i ][ iLen ] && ( argv[ i ][ iLen ] != ',' ) )
                iLen++;
            if( argv[ i ][ iLen ] == ',' ) argv[ i ][ iLen++ ] = '\\0';
            sPtr++;
        }
        return j;

    case _none:
        return TRUE;
}
return FALSE;
}

/*****
/* GetNArg : Determine command line parameters without prefix */
/*-----*/
/* Input      : argc - Argument count (s. main(INT argc, CHAR *argv[])) */
/*              argv - Argument values (s. main(INT argc, CHAR *argv[])) */
/*              pPrefix - Parameter prefix (e.g: "-o" fOr Output), which */
/*                      CANNOT be contained in the parameters being */
/*                      searched for. */
/*              pString - Address of string pointer array, that receives */
/*                      the parameters. */
/*              iNumElements - In case several parameters are contained */
/*                      in the command line, only the maximum */
/*                      parameters will be stored in iNumElements. */
/* Output      : Number of determined parameters, or 0 if no command */
/*              parameters exist. */
*****/
INT GetNArg( INT argc, PCHAR argv[], PCHAR pPrefix,
             PCHAR *pString, INT iNumElements )
{
    INT i, j;
    INT iLen;

    iLen = _fstrlen( pPrefix );

    for( i = 1, j = 0; ( i < argc ) && ( j < iNumElements ); i++ )
        if( _fstrnicmp( pPrefix, ( PCHAR )argv[ i ], iLen ) != 0 )
            pString[ j++ ] = argv[ i ];

    return j;
}

/*****
/* FindString : Find string in a string array and return position of */
/*              string found within the array. */
/*-----*/
/* Input      : pArray - String array to be searched */
/*              pFind - String to be searched for */
/*              iNum - Number of strings in string array */
/* Output      : Position + 1 of found string in array or 0 if search */
/*              string is not in the array. */
*****/
INT FindString( PCHAR pArray[], PCHAR pFind, INT iNum )
{
    INT i;
    for( i = 0; i < iNum; i++ )
        if( _fstricmp( pArray[ i ], pFind ) == 0 ) return i + 1;
}

```

```

    return 0;
}

/*****
/* htoi : Converts passed hex string to numerical value */
**-----**/
/* Input      : pStr - Address of string to be converted */
/*             iDef - Default value */
/* Output     : Value of string or 'iDef' if passed string doesn't */
/*             represent a valid hex number. */
*****/
INT htoi( PCHAR pStr, INT iDef )
{
    INT iNumChars = 0;
    INT iVal = 0;

    if( pStr )
    {
        while( *pStr == ' ' ) pStr++; /* Skip spaces */
        for(;;)
        {
            switch( *pStr )
            {
                case '0': case '1': case '2': /* Convert decimal numbers */
                case '3': case '4': case '5':
                case '6': case '7': case '8':
                case '9':
                    iVal *= 0x10; /* Shift value one place to the left */
                    iVal += *(pStr++) - '0'; /* Add 'Unit's' place */
                    iNumChars++; /* Increase number of converted characters */
                    break;
                case 'a': case 'b': case 'c':
                case 'd': case 'e': case 'f':
                    iVal *= 0x10; /* Shift value one place to the left */
                    iVal += *(pStr++) - 'a' + 10; /* Add 'Unit's' place */
                    iNumChars++; /* Increase number of converted characters */
                    break;
                case 'A': case 'B': case 'C':
                case 'D': case 'E': case 'F':
                    iVal *= 0x10; /* Shift value one place to the left */
                    iVal += *(pStr++) - 'A' + 10; /* Add 'Unit's' place */
                    iNumChars++; /* Increase number of converted characters */
                    break;
                default:
                    return ( iNumChars != 0 ) ? iVal : iDef;
            }
        }
    }
    return iDef;
}

/*****
/* strichr : Find first occurrence of a partial string without */
/*           taking case into account. */
**-----**/
/* Input      : pStr - Address of string to be searched */
/*             pSearch - Partial string to be searched for */
/* Output     : Address of character that follows the first occurrence */
/*             of partial string being searched for. */
*****/
PCHAR strichr( PCHAR pStr, PCHAR pSearch )
{
    if( pStr && pSearch )
    {
        INT iLen; /* Length of search string */

        iLen = _fstrlen( pSearch ); /* Model independent string functions */
        while( *pStr ) /* String to be searched still not empty ? */
            if( _fstrnicmp( pStr, pSearch, iLen ) == 0 )
            {
                /*- Pass address of character after found search string -*/
                pStr += iLen;
                if( *pStr ) return pStr;
            }
            else pStr++;
    }
    return NULL; /* Search string not found */
}

```

```
}
```

```
#endif
```