stormamiga_lib_e

Matthias Henze Fleischer

COLLABORATORS

	TITLE : stormamiga_lib_e	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Matthias Henze Fleischer	April 15, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

stormamiga_lib_e

1.1 Contents

1.2 introduction

Introduction: ~~~~~~~~~~~~~~

The "stormamiga.lib" is a linkerlibrary for StormC. It replaces the linkerlibraries "amiga.lib", "storm020.lib", "math020.lib", "math881.lib", "math040.lib", "math060.lib" and the Startupcode "startup.o". For StormC version 3.x the following linkerlibraries are replaced "stormclibstartup020.lib", "stormcstartup020.lib" and "stormcsupport020.lib" too. The "stormamiga.lib" is completely written in Assembler. Programs linked with it become really fast and small.

The goal is to replace all functions of "amiga.lib" that are not part of the pragma files and all functions of "storm.lib" without the a6 small datamodell with small and fast assembler functions. Furthermore I'm going to implement some special functions (GNU C, SAS C, DICE, Maxon C++) that allow easier programming. So on March the 18th 1996 I decided to write "stormamiga.lib".

1.3 special features

Special features of "stormamiga.lib": ~~~~~~

1.) written in Assembler; programs linked with "stormamiga.lib" become very small and fast

2.) special versions of startupcode for far and near code for "Ansi C" and "C++"; programs become even shorter

3.) special versions of "stormamiga.lib" for MC68EC020+, MC68881+, MC68040+ and MC68060; "stormamiga_040.lib" is up to 22 times faster than "math040.lib" and shorter

4.) support for near code model; you can optimize your program 'til the last byte

5.) the "stormamiga.lib" contains many functions (e.g.: "access", "chdir", "sleep", "strdup", "isnan" etc.) of Unix and Posix, some special functions (e.g.: "strbpl", "strins" etc.) of GNU C, SAS C, DICE and Maxon C++ compilers and supports paths in Unix format; changing from one compiler to another becomes rather simple.

6.) the definition of STORMAMIGA_INLINE generates inlinefunctions for many systemcalls (e.g.: "putc", "GetAPen" etc.)

7.) the definition of STORMAMIGA_STACK xxxx (xxxx amount of stackmemory) sets the stacksize your program wants to use

8.) programs linked with "stormamiga.lib" are automatically startable from Workbench if STORMAMIGA_NOWB is not defined

9.) "stormamiga.lib" has got some reduced functions (without math support) (e.g.: "printf_", "scanf_" etc.); programs become shorter and faster

10.) the definition of STORMAMIGA_OS3 generates AmigaOS 3.x optimized versions of many functions (e.g.: "malloc", "free" etc.)

11.) special versions of string functions that are aware of the german language (e.g.: "islower_d", "toupper_d" etc.) and special output functions for german (e.g.: "strftime_d", "asctime_d" etc.); have a look at german functions

12.) "stormamiga.lib" contains some special functions (e.g.: "button", "max_Width" etc.) that simplifies control of mouse or window handling dramatically; so even unskilled Amiga programmers can use such things easiely; have look at functions

13.) "stormamiga.lib" is low priced

1.4 requirements

Requirements: ~~~~~~~~~~~~~~

- An Amiga - AmigaOS 2.0 (V37) or higher - MC68EC020 or higher - StormC V.2.0 or higher

1.5 installation

Installation: ~~~~~~~~~~~~~~

Allthough there are different versions of "stormamiga.lib" (named "stormamiga.lib", "stormamiga_nc.lib", etc.) this guide calls them "stormamiga.lib". The different startupcodes ("stormamiga_startups.o", "stormamiga_nc_startups.o", etc.) are usually called "stormamiga_startups.o".

The "stormamiga.lib" is installed by the Installer. StormC has to be installed before you can install "stormamiga.lib". Run "HD-Install_xxx" (xxx is your prefered language) from Workbench and follow the instructions. After the succesfull installation start StormC in order to change your configuration. Open a new or an existing project and insert the "stormamiga.lib" in first place.If you want to create shared libraries the "stormamiga_library.lib" has to be inserted before "stormamiga.lib". If you want to use the small codemodell you should include the nc-version of "stormamiga.lib" (e.g.: "stormamiga_nc.lib") that is optimized for the small codemodell. Then choose a startupcode. Choose the "Linkeroptions" from the "Preferences" menu. Select "Own startupcode" and the startupcode you want to use (e.g.: "stormamiga_startups+.o" for C or "stormamiga_C++_startups+.o" for C++). In order to use the special functions of "stormamiga.lib" you have to include "stormamiga.h" in your source code (#include <stormamiga.h>). Last but not least you have to define STORMAMIGA .

1.6 function overview

Overview of supported functions: ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

functions of "stormamiga.lib", of "stormamiga_nc.lib",of "stormamiga_881.lib", of "stormamiga_nc_881.lib",of "stormamiga_040.lib", of "stormamiga_nc_040.lib"of "stormamiga_060.lib" and of "stormamiga_nc_060.lib" amiga.lib functions assert functions ctype functions

dirent functions fcntl functions internal functions class filebuf class fstream class ifstream

class ios class istream class ofstream class ostream class streambuf math functions

new functions resource functions setjmp functions signal functions other functions special functions

stat functions stdio functions stdlib functions string functions time functions unistd functions

functions of "stormamiga-library.lib" and of "stormamiga-library_nc.lib" internal functions new functions other functions

stdlib functions time functions unistd functions

1.7 amiga.lib functions

amiga.lib functions

AddTOF() ArgArrayDone() ArgArrayInit() ArgInt() ArgString() BeginIO() CallHook() CallHookA() CoerceMethod() CoerceMethodA(CreatePort() CreateTask() CreateExtIO() CreateStdIO() DeleteExtIO() DeletePort() DeleteStdIO() DeleteTask() DoMethod() DoMethodA() DoSuperMethod() DoSuperMethodA() FastRand() FreeIEvents() HookEntry() HotKey() InvertString() LibAllocPooled() LibCreatePool() LibDeletePool() LibFreePooled() NewList() RangeRand() RemTOF() SetSuperAttrs() SPRINTF() TimeDelay() VSPRINTF() waitbeam()

1.8 assert functions

assert functions

assert() assert_() do_assert() do_assert_()

1.9 ctype functions

ctype functions

isalnum() isalnum_d() isalpha() isalpha_d() iscntrl() isdigit() isgraph() islower() islower_d() isprint() isprint_d() ispunct() ispunct_d() isspace() isupper() isupper_d() isxdigit() tolower() tolower_d() toupper() toupper_d() which_xdigit()

1.10 dirent functions

dirent functions

closedir() opendir() readdir() rewinddir()

1.11 internal functions

internal functions

Add64() blocksize_a2_d2() Cmp64() dotimer() exitNearData() EXIT_0_Main() EXIT_4_free() EXIT_4_free_3() EXIT_9_AtExitFuncti EXIT_9_Stack() geta4() GetBaseReg() initNearData() INIT_0_InitExceptionHandling INIT_5_clock() INIT_5_timer INIT_9_Stack() lib_64bit_shl() lib_64bit_shr() lib_catch() lib_catchclass() lib_destruct() lib_destruct_a0() lib_rethrow() lib_throw() main() (Ansi C) main() (C++) main__() Neg64() SDiv64() SDivMod64() set_terminate() set_unexpected() SMod64() SMult64() Sub64() terminate() UDiv64() UDivMod64() UMod64() UMult64() unexpected() wbmain() (Ansi C) wbmain() (C++) wbmain_() (Ansi C) wbmain_() (C++)

Autolib functions EXIT_3_AmigaGuideBase() EXIT_2_AslBase() EXIT_3_BulletBase() EXIT_3_ColorWheelBase() EXIT_2_CxBase EXIT_3_DataTypesBase() EXIT_2_DiskfontBase() EXIT_1_DOSBase() EXIT_2_ExpansionBase() EXIT_2_GadToolsBase() EXIT_3_GradientSliderBase() EXIT_2_GfxBase() EXIT_2_IconBase() EXIT_2_IFFParseBase() EXIT_2_IntuitionBase() EXIT_2_Ke EXIT_2_LayersBase() EXIT_3_LocaleBase() EXIT_3_LowLevelBase() EXIT_2_MathBase() EXIT_2_MathIeeeDoubBasBase() EXIT_2_MathIeeeDoubTransBase() EXIT_2_MathIeeeSingBasBase() EXIT_2_MathIeeeSingTransBase() EXIT_2_MathTransBase() EXIT_2_MUIMasterBase() EXIT_3_NVBase() EXIT_3_RealTimeBase() EXIT_2_ReqToolsBase() EXIT_2_RexxSysBase() EXIT_3_ EXIT_1_UtilityBase() EXIT_2_VersionBase() EXIT_2_WizardBase() EXIT_2_WorkbenchBase() INIT_3_AmigaGuideBase() INIT_2_AslBase() INIT_3_BulletBase() INIT_3_ColorWheelBase() INIT_2_CxBase() INIT_3_DataTypesBase() INIT_2_DiskfontBas INIT_1_DOSBase() INIT_2_ExpansionBase() INIT_2_GadToolsBase() INIT_3_GradientSliderBase() INIT_2_GfxBase() INIT_2_Icon INIT_2_IFFParseBase() INIT_2_IntuitionBase() INIT_2_KeymapBase() INIT_2_LayersBase() INIT_3_LocaleBase() INIT_3_LowLev INIT_2_MathBase() INIT_2_MathIeeeDoubBasBase() INIT_2_MathIeeeDoubTransBase() INIT_2_MathIeeeSingBasBase() INIT_2_M INIT_2_MathTransBase() INIT_2_MUIMasterBase() INIT_3_NVBase() INIT_3_RealTimeBase() INIT_2_ReqToolsBase() INIT_2_Re INIT_3_TranslatorBase() INIT_1_UtilityBase() INIT_2_VersionBase() INIT_2_WizardBase() INIT_2_WorkbenchBase()

IO functions amigaclose() amigaeof() amigaflush() amigagetc() amigagetcunget() amigaopen() amigaputc() amigaread() amigareadunget() amigaseek() amigaungetc() amigawrite() double_in() double_out() EXIT_5_fstream() EXIT_5_InitFiles() EXIT_5_InitSt EXIT_6_InitPOSIXIOFiles() EXIT_9_chdir() form_in() form_in64() form_in_() form_in64_() form_out() form_out64() form_out_() form_out64_() getch() INIT_0_InitFiles() INIT_0_NEAR_CODE_StdioFiles() INIT_5_fstream() INIT_5_InitStdIOFiles() INIT_6_Init INIT_9_chdir() putch() udiv_64() ungetch() UnixToAmigaPath() un_signed_out()

math functions (only in the version for MC68EC020+) lib_double2float() lib_double2int() lib_float2double() lib_float2int() lib_float_add() lib_float_cmp() lib_float_div() lib_float_mult() lib_float_neg() lib_float_sub() lib_float_tst() lib_int2double() lib_int2floa

internal data AmigaGuideBase AslBase BulletBase ColorWheelBase CxBase DataTypesBase DiskfontBase DOSBase errno ExpansionBase fileList GadToolsBase GradientSliderBase GfxBase IconBase IFFParseBase IntuitionBase KeymapBase LayersBase LocaleBase LowLevelBase MathBase MathIeeeDoubBasBase MathIeeeDoubTransBase MathIeeeSingBasBase MathIeeeSingTransBase MathTransBase MUIMasterBase NVBase RealTimeBase ReqToolsBase RexxSysBase SaveSP signal_dat std__err std__in std__out tmpnamList tmpnamNext TranslatorBase UtilityBase VersionBase WBMsg WizardBase WorkbenchBase __ctypetable

1.12 class filebuf

class filebuf

Constructors filebuf::filebuf()

Destructors filebuf::~filebuf()

Methods filebuf *open(const char *, int) filebuf *filebuf::close() filebuf::seekoff(long, enum seek_dir__ios, int) filebuf::seekpos(streampo int) filebuf::setbuf(char *, uint) filebuf::sync() filebuf::doallocate() filebuf::overflow(int) filebuf::underflow() filebuf::xsputn(cchar *, int) filebuf::xsgetn(char *, int) filebuf::pbackfail(int)

1.13 class fstream

class fstream

Constructors fstream::fstream() fstream::fstream(cchar *, int)

Methods fstream::open(const char *, int) fstream::close() fstream::setbuf(char *, uint)

1.14 class ifstream

class ifstream

Constructors ifstream::ifstream() ifstream::ifstream(cchar *, int)

Methods ifstream::open(const char *, int) ifstream::close() ifstream::setbuf(char *, uint)

1.15 unnamed.1

class ifstream

Constructors ifstream::ifstream() ifstream::ifstream(cchar *, int)

Methods ifstream::open(const char *, int) ifstream::close() ifstream::setbuf(char *, uint)

1.16 class ios

class ios

Methods ios::bitalloc() ios::init(streambuf *) ios::xalloc() &ios::userword(int)

Flags dec(ios &) hex(ios &) oct(ios &)

1.17 class istream

class istream

Constructors istream::istream(streambuf *)

Methods istream::ipfx(int) istream::get() istream::peek() istream::operator >>(char &) istream::operator >>(uchar &) istream::operator >>(schar &) istream::operator >>(uchar *) istream::operator >>(schar *) istream::operator >>(char *) istream::operator >>(short &) istream::operator >>(ushort &) istream::operator >>(int &) istream::operator >>(uint &) istream::operator >>(long &) istream::operat >>(ulong &) istream::operator >>(float &) istream::operator >>(double &) istream::operator >>(long double &) istream::operator >>(istream &(*f)(istream &)) istream::operator >>(ios &(*f)(ios &)) istream::get(char *, int, char) istream::get(uchar *, int, char) istream::get(schar *, int, char) istream::get(schar &) istream::get(uchar &) istream::get(char &) istream::getline(char *, int, char) istream::getline(uchar *, int, char) istream::getline(schar *, int, char) istream::ignore(int, int) istream::read(uchar *, int) istream::read(schar *, int) istream::read(char *, int) istream::seekg(streampos) istream::seekg(streamoff, enum seek_dir__ios) isteam &ws(istream &)

1.18 class ofstream

class ofstream

Constructors ofstream::ofstream() ofstream::ofstream(const char *, int)

Methods ofstream::open(const char *, int) ofstream::close() ofstream::setbuf(char *, uint)

1.19 class ostream

class ostream

Constructors ostream::ostream(streambuf *)

Methods ostream::opfx() ostream::osfx() ostream::operator <<(schar) ostream::operator <<(uchar) ostream::operator <<(char) ostream::operator <<(cuchar *) ostream::operator <<(cschar *) ostream::operator <<(cchar *) ostream::operator <<(short) ostream::operator <<(ushort) ostream::operator <<(int) ostream::operator <<(uint) ostream::operator <<(long) ostream::operator <<(ulong) ostream::operator <<(float) ostream::operator <<(double) ostream::operator <<(void *) ostream::flush(); ostream::seekp(strea enum seek_dir__ios) ostream::seekp(streamoff, enum seek_dir__ios) ends(ostream &) endl(ostream &)

1.20 class streambuf

class streambuf

Constructors streambuf::streambuf() streambuf::streambuf(char *, int)

Destructors streambuf::~streambuf()

Methods streambuf *streambuf::setbuf(char *, ulong) streambuf::setbuffer(char *, ulong, int) streambuf::doallocate() streambuf::sgetn(char *, int) streambuf::sputn(cchar *, int) streambuf::overflow(int) streambuf::underflow() streambuf::xsgetn(char *, int) streambuf::xsputn(cchar *, int)

1.21 math functions

math functions

acos() asin() atan() atan2() ceil() cos() cosh() exp() fabs() floor() fmod() frexp() isinf() isnan() ldexp() log() log10() modf() pow() sin() sinh() sqrt() tan() tanh()

1.22 new functions

new functions

set_new_handler()

1.23 resource functions

resource functions

getrusage()

1.24 setjmp functions

setjmp functions

longjmp() setjmp()

1.25 signal functions

signal functions

raise() signal()

1.26 special functions

special functions

button() button_al() button_ar() button_bl() button_br() divs64() divsl() divu64() divul() max_Height() max_Width() muls() muls64() mulu() mulu64() waitbutton() waitbutton_al() waitbutton_ar() waitbutton_bl() waitbutton_br()

1.27 stat functions

stat functions

chmod() fstat() lstat() mkdir() stat()

1.28 other functions

other functions

cinfilebuf::cinfilebuf() cinfilebuf::~cinfilebuf() coutfilebuf::coutfilebuf() coutfilebuf::~coutfilebuf() operator new(size_t) operator delete(void, size_t)

1.29 stdio functions

stdio functions

clearerr() fclose() feof() ferror() fflush() fgetc() fgetpos() fgets() fopen() fprintf() fprintf64() fprintf_() fprintf64_() fputc() fputs() fputstr() fread() freopen() fscanf() fscanf64() fscanf_() fscanf64_() fseek() fsetpos() ftell() fwrite() getc() getchar() gets() getw() perror() printf() printf64() printf_() printf64_() putc() putchar() puts() putw() remove() rename() rewind() scanf() scanf64() scanf_() scanf64_() setbuf() setbuffer() setlinebuf() setvbuf() snprintf() snprintf64() snprintf_() snprintf64_() sprintf() sprintf64() sprintf_() sprintf64_() sscanf() sscanf64() sscanf_() sscanf64_() tmpfile() tmpnam() ungetc() vfprintf() vfprintf64() vfprintf_() vfprintf64_() vfscanf() vfscanf64() vfscanf_() vfscanf64_() vprintf() vprintf64() vprintf_() vprintf64_() vscanf() vscanf64() vscanf_() vscanf64_() vsnprintf() vsnprintf64() vsnprintf_() vsnprintf64_() vsprintf() vsprintf64() vsprintf_() vsprintf64_() vsscanf() vsscanf64() vsscanf_() vsscanf64_()

1.30 stdlib functions

stdlib functions

abort() abort__STANDARD() abs() atexit() atof() atoi() atol() atoll() bsearch() calloc() div() exit() free() free_3() getenv() inttostr() labs() ldiv() llabs() llongtostr() malloc() malloc_3() qsort() rand() realloc() srand() strtod() strtol() strtoll() strtoul() strtoull() system() uinttostr() ullongtostr()

1.31 string functions

string functions

bcmp() bcopy() bzero() clrmem() cmpmem() ffs() index() memccpy() memchr() memcmp() memcpy() memmove() memset() movmem() rindex() setmem() stccpy() stpchr() stpcpy() strbpl() strcasecmp() strcasecmp_d() strcat() strchr() strcmp() strcpy() strcspn() strdup() strerror() stricmp() stricmp_d() strins() strlen() strlower() strlower_d() strlwr() strlwr_d() strncasecmp() strncasecmp_d() strncat() strncmp() strncpy() strncpyn() strnicmp() strnicmp_d() strpbrk() strrchr() strsep() strspn() strstr() strtok() strupper() strupper_d() strupr() strupr_d() strxfrm() swab()

1.32 time functions

time functions

asctime() asctime_d() clock() ctime() ctime_d() difftime() gmtime() mktime() strftime() strftime_d() time() times() utime() utimes()

1.33 unistd functions

unistd functions

access() chdir() close() getcwd() getwd() lseek() mktemp() read() rmdir() sleep() unlink() usleep() write

1.34 fcntl functions

fcntl functions

creat() open()

1.35 internal functions

internal functions

EXIT_4_free() EXIT_4_free_3() EXIT_9_Stack() INIT_5_clock() INIT_9_Stack()

shared-librarie functions abortLibInit() LibClose() LibExpunge() LibInit() LibInitError() LibNull() LibOpen() __LibClose() __LibOpen()

Autolib functions EXIT_3_AmigaGuideBase() EXIT_2_AslBase() EXIT_3_BulletBase() EXIT_3_ColorWheelBase() EXIT_2_CxBase EXIT_3_DataTypesBase() EXIT_2_DiskfontBase() EXIT_1_DOSBase() EXIT_2_ExpansionBase() EXIT_2_GadToolsBase() EXIT_3_GradientSliderBase() EXIT_2_GfxBase() EXIT_2_IconBase() EXIT_2_IFFParseBase() EXIT_2_IntuitionBase() EXIT_2_Ke EXIT_2_LayersBase() EXIT_3_LocaleBase() EXIT_3_LowLevelBase() EXIT_2_MathBase() EXIT_2_MathIeeeDoubBasBase() EXIT_2_MathIeeeDoubTransBase() EXIT_2_MathIeeeSingBasBase() EXIT_2_MathIeeeSingTransBase() EXIT_2_MathTransBase() EXIT_2_MUIMasterBase() EXIT_3_NVBase() EXIT_3_RealTimeBase() EXIT_2_ReqToolsBase() EXIT_2_RexxSysBase() EXIT_3_ EXIT_1_UtilityBase() EXIT_2_VersionBase() EXIT_2_WizardBase() EXIT_2_WorkbenchBase() INIT_3_AmigaGuideBase() INIT_2_AslBase() INIT_3_BulletBase() INIT_3_ColorWheelBase() INIT_2_CxBase() INIT_3_DataTypesBase() INIT_2_DiskfontBas INIT_1_DOSBase() INIT_2_ExpansionBase() INIT_2_GadToolsBase() INIT_3_GradientSliderBase() INIT_2_GfxBase() INIT_2_Icon INIT_2_IFFParseBase() INIT_2_IntuitionBase() INIT_2_KeymapBase() INIT_2_LayersBase() INIT_3_LocaleBase() INIT_3_LowLev INIT_2_MathBase() INIT_2_MathIeeeDoubBasBase() INIT_2_MathIeeeDoubTransBase() INIT_2_MathIeeeSingBasBase() INIT_2_M INIT_2_MathTransBase() INIT_2_MUIMasterBase() INIT_3_NVBase() INIT_3_RealTimeBase() INIT_2_ReqToolsBase() INIT_2_Re INIT_3_TranslatorBase() INIT_1_UtilityBase() INIT_2_VersionBase() INIT_2_WizardBase() INIT_2_WorkbenchBase()

IO functions EXIT_5_fstream() EXIT_5_InitFiles() EXIT_5_InitStdIOFiles() EXIT_6_InitPOSIXIOFiles() EXIT_9_chdir() INIT_0_In INIT_0_NEAR_CODE_StdioFiles() INIT_5_fstream() INIT_5_InitStdIOFiles() INIT_6_InitPOSIXIOFiles() INIT_9_chdir()

internal data AmigaGuideBase AslBase BulletBase ColorWheelBase CxBase DataTypesBase DiskfontBase DOSBase ExpansionBase fileList GadToolsBase GradientSliderBase GfxBase IconBase IFFParseBase IntuitionBase KeymapBase LayersBase LocaleBase LowLevelBase MathBase MathIeeeDoubBasBase MathIeeeDoubTransBase MathIeeeSingBasBase MathIeeeSingTransBase MathTransBase MUIMasterBase NVBase RealTimeBase ReqToolsBase RexxSysBase std__err std__in std__out tmpnamList tmpnamNext TranslatorBase UtilityBase VersionBase WizardBase WorkbenchBase

1.36 new functions

new functions

set_new_handler()

1.37 other functions

other functions

operator new(size_t) operator delete(void, size_t)

1.38 stdlib functions

stdlib functions

free() free_3() malloc() malloc_3()

1.39 time functions

time functions

clock()

1.40 unistd functions

unistd functions

chdir()

1.41 functions

Functions of "stormamiga.lib": ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Function Overview

Because standard Ansi-C and C++ functions are described in the StormC guide, I am going to explain only the special functions.

main__() Startupcode wbmain()

AmigaDOS stdio functions SPRINTF VSPRINTF

stdio functions fprintf64 fprintf_ fprintf64_ fscanf64

fscanf_ fscanf64_ getw printf64 printf_ printf64_ putw scanf64

scanf_ scanf64_ setbuffer setlinebuf snprintf snprintf64 snprintf_ snprintf64_

sprintf64 sprintf_ sprintf64_ sscanf64 sscanf_ sscanf64_ vfprintf64 vfprintf_

vfprintf64_ vfscanf vfscanf64 vfscanf_ vfscanf64_ vprintf64 vprintf_ vprintf64_

vscanf vscanf64 vscanf_ vscanf64_ vsnprintf vsnprintf64 vsnprintf_ vsnprintf64_

vsprintf64 vsprintf_ vsprintf64_ vsscanf vsscanf64 vsscanf_ vsscanf64_

unistd functions access chdir close getcwd

getwd lseek mktemp read rmdir sleep unlink usleep

write

fcntl functions creat open

dirent functions closedir opendir readdir rewinddir

stat functions chmod fstat lstat mkdir

stat

string functions bcmp bcopy bzero clrmem

cmpmem ffs index memccpy movmem rindex setmem stccpy

stpchr stpcpy strbpl strcasecmp strcasecmp_d strcoll strdup stricmp_d

strins strlower strlower_d strlwr_d strncasecmp strncasecmp_d strncpyn strnicmp

strnicmp_d strsep strupper strupper_d strupr_d strxfrm swab

ctype functions isalnum_d isalpha_d islower_d isprint_d

ispunct_d isupper_d tolower_d toupper_d

assert functions assert_

time functions asctime_d ctime_d strftime strftime_d

times utime utimes

resource functions getrusage

math functions isinf isnan

special functions button button_al button_ar button_bl

button_br divsl divul divs64 divu64 max_Height max_Width muls

mulu muls64 mulu64 waitbutton waitbutton_al waitbutton_ar waitbutton_bl waitbutton_br

1.42 The Startupcodes

The Startupcodes ~~~~~~~~~~~~~~~~~

The startupcodes of "stormamiga.lib" offer more functions than the original StormC "startup.o". Furthermore they are more flexible and smaller than "startup.o".

The Startupcodes for Ansi-C

The startupcodes "stormamiga_startups.o", "stormamiga_nc_startups.o", "stormamiga_startups+.o" and "stormamiga_nc_startups+.o" call, if started from CLI, the function "_main__". This one is equal to the function main__() in Ansi-C. If this function does not exist in your program, it will be linked from "stormamiga.lib". The function "_main__" calls "_main", that is equal to "main()" in Ansi-C. If this function does not exist, the linker will report an error. A program started from WorkBench calls _wbmain_ . If "STORMAMIGA_NOWB" is defined, an empty function will be linked. Otherwise it will be linked from "stormamiga.lib". The function "_wbmain_" calls "_wbmain" that is equal to wbmain() in Ansi-C. If this function does not exist, it will be linked from "stormamiga.lib".

The Startupcodes for C++ The startupcodes "stormamiga_C++_startups.o", "stormamiga_nc_C++_startups.o", "stormamiga_C++_startu and "stormamiga_nc_C++_startups+.o" call, if started from CLI, the function "main_". This one is equal to the function main() in C++. If this function does not exist in your program, it will be linked from "stormamiga.lib". The function "main_" calls "main__iPPc", that is equal to "main(int, char **)" in C++. If this function does not exist, the linker will report an error. A program started from WorkBench calls wbmain__ . If "STORMAMIGA_NOWB" is defined, an empty function will be linked. Otherwise it will be linked from "stormamiga.lib". The function "wbmain__" calls "wbmain__P09WBStartup" that is equal to wbmain(struct WBStartup *) in C++. If this function does not exist, it will be linked from "stormamiga.lib".

Functional Overview of Startupcodes

Startupcode	Ansi C	C++	Codemodell	Datamodell				FAR	NEAR	FAR	NEAR A4	NEAR A6	
stormamiga_startups.o	yes	no	yes	(yes)	yes	(yes)[1]	no						
stormamiga_startups+.o	yes	no	yes	(yes)	yes	yes	no						
stormamiga_nc_startups.o	yes	no	(yes)	yes	yes	(yes)[1]	no						
stormamiga_nc_startups+.o	yes	no	(yes)	yes	yes	yes	no						
stormamiga_C++_startups.o	no	yes	yes	(yes)	yes	(yes)[1]	no						
stormamiga_C++_startups+.o	no	yes	yes	(yes)	yes	yes	no						
stormamiga_nc_C++_startups.o	no	yes	(yes)	yes									

yes | (yes)[1] | no | |------------------------------|--------|-----|-------|-------|-----|---------|---------| | stormamiga_nc_C++_startups+.o | no | yes | (yes) | yes | yes | yes | no | ‾‾ (yes) = Startupcodes marked this way will work with this Codemodell, but your programms become bigger.

(yes)[1] = Startupcodes marked this way will create resident programs. If your programs do not have to be resident, this Startupcodes have no further advantage. Your programs become only bigger.

1.43 main__()

The "main__()" function. ~~~~~~~~~~~~~~~~~~~~~~~~

If your program does not need any arguments or you want to write this procedure on your own you can rename "main()" into "main__()". Your program becomes smaller.

Important

"main__()" is not recognized as a normal main function by the compiler, so the returncode is not set to 0. You have do do this by calling "return 0" at the end of your program.

The function "main__()" is only supported by the following startupcodes "stormamiga_startups.o", "stormamiga_nc_startups.o", "stormamiga_startups+.o" and "stormamiga_nc_startups+.o" "main__()" needs Ansi C to work.

1.44 _wbmain_

The functions "_wbmain_" and "_wbmain__". ~~~~~~

The functions "_wbmain_" and "_wbmain__" are necessary to use sources from different compilers (GNU C, SAS C, DICE, Maxon C++, StormC) with "stormamiga.lib" and its Startupcodes. Furthermore they contend all functions to quit a program, started from WorkBench, correctly. The functions "_wbmain_" and "_wbmain__" copy the Workbenchmessage into the global variable "WBMsg". Please look at Workbenchexamples .

1.45 wbmain()

The function "wbmain()". ~~~~~~~~~~~~~~~~~~~~~~~~

The function "wbmain()" is part of "stormamiga.lib". It changes to the actual directory and, if STORMAMIGA_NO_IO_WB is not defined, redirects standard output "stdout" and standard input "stdin" to a CLI window. The parameters of this window can be defined in the variable "__stdiowin". The defaultvalue is "char __stdiowin[] = "CON://///AUTO/CLOSE/WAIT"". If you write your own "wbmain()", your function will be used and not the one from "stormamiga.lib". Please look at Workbenchexamples .

1.46 workbenchexamples

Some examples for a Workbench start. ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

"stormamiga.lib" contains several ways to react on a start from WorkBench. The following examples may help to show the usage of those feautures.

Example 1 void main (int argc, char **argv) { if (argc == 0) { // Place all function necessary for a Workbench start here } else { // Place all function necessary for a CLI start here }

Example 2 void main__ (void) { extern struct WBStartup *WBMsg; if (WBMsg != 0) { // Place all function necessary for a Workbench start here } else { // Place all function necessary for a CLI start here }

Example 3 void main (int argc, char **argv) { // Place all function necessary for a CLI start here }

void wbmain (struct WBStartup *wbs) { // Place all function necessary for a Workbench start here }

1.47 sprintf

SPRINTF Formated output into stringbuffer "s".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = SPRINTF (s, format, ...);

long r; char *s; const char *format;

Standard not (yet) (own development)

Function Formated output into stringbuffer "s". The formatstring describes the ouputformat followed by different parameters.

"SPRINTF" makes use of the RawDoFmt function of "exec.library" and is a very small function.

Return No. of printed characters.

1.48 vsprintf

VSPRINTF Formated output into stringbuffer "s".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = VSPRINTF (s, format, vl);

long r; char *s; const char *format; va_list vl;

Standard not (yet) (own development)

Function Formated output into stringbuffer "s". The formatstring "format" describes the outputformat, followed by a parameterlist "vl".

"VSPRINTF" makes use of the RawDoFmt function of "exec.library" and is a very small function.

Return No. of printed characters.

1.49 printf64

printf64 Formated output to standard-output "stdout". Extended version of "printf".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = printf64 (format, ...);

int r; const char *format;

Standard not (yet) (own development)

Function Formated output to standard-output "stdout". The outputformatstring "format" defines the outputformat followed by different parameters. The function "printf64" supports the specifier "L" for long long int or unsigned long long int.

Return No. of printed characters or a negative No. if out failed.

1.50 printf_

printf_ Formated output to standard-output "stdout". Functionreduced version of "printf".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = printf_ (format, ...);

int r; const char *format;

Standard not (yet) (own development)

Function Formated output to standard-output "stdout". The outputformatstring_ "format" defines the outputformat followed by different parameters.

Return No. of printed characters or a negative No. if out failed.

1.51 printf64_

printf64_ Formated output to standard-output "stdout". Extended version of "printf_".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = printf64_ (format, ...);

int r; const char *format;

Standard not (yet) (own development)

Function Formated output to standard-output "stdout". The outputformatstring_ "format" defines the outputformat followed by different parameters. The function "printf64_" supports the specifier "L" for long long int or unsigned long long int.

Return No. of printed characters or a negative No. if out failed.

1.52 fprintf64

fprintf64 Formated output into file "f". Extended version of "fprintf".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = fprintf64 (f, format, ...);

int r; FILE *f; const char *format;

Standard not (yet) (own development)

Function Formated output into file "f". The outputformatstring "format" defines the outputformat followed by different parameters. The function "fprintf64" supports the specifier "L" for long long int or unsigned long long int.

Return No. of printed characters or a negative No. if out failed.

1.53 fprintf_

fprintf_ Formated output into file "f". Functionreduced version of "fprintf".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = fprintf_ (f, format, ...);

int r; FILE *f; const char *format;

Standard not (yet) (own development)

Function Formated output into file "f". The outputformatstring_ "format" defines the outputformat followed by different parameters.

Return No. of printed characters or a negative No. if out failed.

1.54 fprintf64_

fprintf64_ Formated output into file "f". Extended version of "fprintf_".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = fprintf64_ (f, format, ...);

int r; FILE *f; const char *format;

Standard not (yet) (own development)

Function Formated output into file "f". The outputformatstring_ "format" defines the outputformat followed by different parameters. The function "fprintf64_" supports the specifier "L" for long long int or unsigned long long int.

Return No. of printed characters or a negative No. if out failed.

1.55 sprintf64

sprintf64 Formated Output into stringbuffer "s". Extended version of "sprintf".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = sprintf64 (s, format, ...);

int r; char *s; const char *format;

Standard not (yet) (own development)

Function Formated output into stringbuffer "s". The outputformatstring "format" defines the outputformat followed by different parameters. The function "sprintf64" supports the specifier "L" for long long int or unsigned long long int.

Return No. of printed characters or a negative No. if out failed.

1.56 sprintf_

sprintf_ Formated Output into stringbuffer "s". Functionreduced version of "sprintf".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = sprintf_ (s, format, ...);

int r; char *s; const char *format;

Standard not (yet) (own development)

Function Formated output into stringbuffer "s". The outputformatstring_ "format" defines the outputformat followed by different parameters.

Return No. of printed characters or a negative No. if out failed.

1.57 sprintf64_

sprintf64_ Formated Output into stringbuffer "s". Extended version of "sprintf_".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = sprintf64_ (s, format, ...);

int r; char *s; const char *format;

Standard not (yet) (own development)

Function Formated output into stringbuffer "s". The outputformatstring_ "format" defines the outputformat followed by different parameters. The function "sprintf64_" supports the specifier "L" for long long int or unsigned long long int.

Return No. of printed characters or a negative No. if out failed.

1.58 snprintf

snprintf Formated output of "n" characters into stringbuffer "s".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = snprintf (s, n, format, ...);

int r; char *s; size_t n; const char *format;

Standard not (yet) (BSD)

Function Formated output of "n" characters into stringbuffer "s". A '\0' is appended automatically. Because of this strings longer than "n" are cut down to "n-1" characters. The outputformatstring "format" defines the outputformat followed by different parameters.

Return No. of printed characters or if "n" lower than "format", -1.

Example #define STORMAMIGA #define STORMAMIGA_NOWB

#include <stormamiga.h> #include <stdio.h>

int main__(void) { cchar *format = "This is a test."; char s[8]; int r; size_t n = sizeof s;

r = snprintf(s, n, format); printf("%s\n", s); /* "This is" */ printf("%d\n", r); /* "-1" */ return NULL; }

1.59 snprintf64

snprintf64 Formated output of "n" characters into stringbuffer "s". Extended version of "snprintf".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = snprintf64 (s, n, format, ...);

int r; char *s; size_t n; const char *format;

Standard not (yet) (own development)

Function Formated output of "n" characters into stringbuffer "s". A '\0' is appended automatically. Because of this strings longer than "n" are cut down to "n-1" characters. The outputformatstring "format" defines the outputformat followed by different parameters. The function "snprintf64" supports the specifier "L" for long long int or unsigned long long int.

Return No. of printed characters or if "n" lower than "format" -1.

Example #define STORMAMIGA #define STORMAMIGA_NOWB

#include <stormamiga.h> #include <stdio.h>

int main__(void) { cchar *format = "This is a test."; char s[8]; int r; size_t n = sizeof s;

r = snprintf64(s, n, format); printf64("%s\n", s); /* "This is" */ printf64("%d\n", r); /* "-1" */ return NULL; }

1.60 snprintf_

snprintf_ Formated output of "n" characters into stringbuffer "s". Functionsreduced version of "snprintf".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = snprintf_ (s, n, format, ...);

int r; char *s; size_t n; const char *format;

Standard not (yet) (own development)

Function Formated output of "n" characters into stringbuffer "s". A '\0' is appended automatically. Because of this strings longer than "n" are cut down to "n-1" characters. The outputformatstring_ "format" defines the outputformat followed by different parameters.

Return No. of printed characters or if "n" lower than "format" -1.

Example #define STORMAMIGA #define STORMAMIGA_NOWB

#include <stormamiga.h> #include <stdio.h>

int main__(void) { cchar *format = "This is a test."; char s[8]; int r; size_t n = sizeof s;

r = snprintf_(s, n, format); printf_("%s\n", s); /* "This is" */ printf_("%d\n", r); /* "-1" */ return NULL; }

1.61 snprintf64_

snprintf64_ Formated output of "n" characters into stringbuffer "s". Extended version of "snprintf_".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = snprintf64_ (s, n, format, ...);

int r; char *s; size_t n; const char *format;

Standard not (yet) (own development)

Function Formated output of "n" characters into stringbuffer "s". A '\0' is appended automatically. Because of this strings longer than "n" are cut down to "n-1" characters. The outputformatstring_ "format" defines the outputformat followed by different parameters. The function "snprintf64_" supports the specifier "L" for long long int or unsigned long long int.

Return No. of printed characters or if "n" lower than "format" -1.

Example #define STORMAMIGA #define STORMAMIGA_NOWB

#include <stormamiga.h> #include <stdio.h>

int main__(void) { cchar *format = "This is a test."; char s[8]; int r; size_t n = sizeof s;

r = snprintf64_(s, n, format); printf64_("%s\n", s); /* "This is" */ printf64_("%d\n", r); /* "-1" */ return NULL; }

1.62 vprintf64

vprintf64 Formated output to standard output "stdout". Extended version of "vprintf".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = vprintf64 (format, vl);

int r; const char *format; va_list vl;

Standard not (yet) (own development)

Function Formated output to standard output "stdout". The outputformatstring "format" defines the outputformat followed by a parameterlist "vl". The function "vprintf64" supports the specifier "L" for long long int or unsigned long long int.

Return No. of printed characters or a negative No. if out failed.

1.63 vprintf_

vprintf_ Formated output to standard output "stdout". Functionreduced version of "vprintf".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = vprintf_ (format, vl);

int r; const char *format; va_list vl;

Standard not (yet) (own development)

Function Formated output to standard output "stdout". The outputformatstring_ "format" defines the outputformat followed by a parameterlist "vl".

Return No. of printed characters or a negative No. if out failed.

1.64 vprintf64_

vprintf64_ Formated output to standard output "stdout". Extended version of "vprintf_".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = vprintf64_ (format, vl);

int r; const char *format; va_list vl;

Standard not (yet) (own development)

Function Formated output to standard output "stdout". The outputformatstring_ "format" defines the outputformat followed by a parameterlist "vl". The function "vprintf64_" supports the specifier "L" for long long int or unsigned long long int.

Return No. of printed characters or a negative No. if out failed.

1.65 vfprintf64

vfprintf64 Formated output into file "f". Extended version of "vfprintf".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = vfprintf64 (f, format, vl);

int r; FILE *f; const char *format; va_list vl;

Standard not (yet) (own development)

Function Formated output into file "f". The outputformatstring "format" describes the outputformat followed by a parameterlist "vl". The function "vfprintf64" supports the specifier "L" for long long int or unsigned long long int.

Return No. of printed characters or a negative No. if out failed.

1.66 vfprintf_

vfprintf_ Formated output into file "f". Functionreduced version of "vfprintf".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = vfprintf_ (f, format, vl);

int r; FILE *f; const char *format; va_list vl;

Standard not (yet) (own development)

Function Formated output into file "f". The outputformatstring_ "format" describes the outputformat followed by a parameterlist "vl".

Return No. of printed characters or a negative No. if out failed.

1.67 vfprintf64_

vfprintf64_ Formated output into file "f". Extended version of "vfprintf_".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = vfprintf64_ (f, format, vl);

int r; FILE *f; const char *format; va_list vl;

Standard not (yet) (own development)

Function Formated output into file "f". The outputformatstring_ "format" describes the outputformat followed by a parameterlist "vl". The function "vfprintf64_" supports the specifier "L" for long long int or unsigned long long int.

Return No. of printed characters or a negative No. if out failed.

1.68 vsprintf64

vsprintf64 Formated output into stringbuffer "s". Extended version of "vsprintf".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = vsprintf64 (s, format, vl);

int r; char *s; const char *format; va_list vl;

Standard not (yet) (own development)

Function Formated output into stringbuffer "s". The outputformatstring "format" describes the outputformat followed by a parameterlist "vl". The function "vsprintf64" supports the specifier "L" for long long int or unsigned long long int.

Return No. of printed characters or a negative No. if out failed.

1.69 vsprintf_

vsprintf_ Formated output into stringbuffer "s". Functionreduced version of "vsprintf".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = vsprintf_ (s, format, vl);

int r; char *s; const char *format; va_list vl;

Standard not (yet) (own development)

Function Formated output into stringbuffer "s". The outputformatstring_ "format" describes the outputformat followed by a parameterlist "vl".

Return No. of printed characters or a negative No. if out failed.

1.70 vsprintf64_

vsprintf64_ Formated output into stringbuffer "s". Extended version of "vsprintf_".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = vsprintf64_ (s, format, vl);

int r; char *s; const char *format; va_list vl;

Standard not (yet) (own development)

Function Formated output into stringbuffer "s". The outputformatstring_ "format" describes the outputformat followed by a parameterlist "vl". The function "vsprintf64_" supports the specifier "L" for long long int or unsigned long long int.

Return No. of printed characters or a negative No. if out failed.

1.71 vsnprintf

vsnprintf Formated output of "n" characters into stringbuffer "s".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = vsnprintf (s, n, format, vl);

int r; char *s; size_t n; const char *format; va_list vl;

Standard not (yet) (BSD)

Function Formated output of "n" characters into stringbuffer "s". A '\0' will be appended. Because of this strings longer than "n", are cut down to "n-1" characters. The outputformatstring "format" describes the outputformat followed by a parameterlist "vl".

Return No. of printed characters or if "n" lower than "format" -1.

Example #define STORMAMIGA #define STORMAMIGA_NOWB

#include <stormamiga.h> #include <stdarg.h> #include <stdio.h>

int main__(void) { cchar *format = "This is a test."; char s[8]; int r; size_t n = sizeof s; va_list vl;

va_start (vl, format); r = vsnprintf(s, n, format, vl); va_end (vl); printf("%s\n", s); /* "This is" */ printf("%d\n", r); /* "-1" */ return NULL; }

1.72 vsnprintf64

vsnprintf64 Formated output of "n" characters into stringbuffer "s". Extended version of "vsnprintf".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = vsnprintf64 (s, n, format, vl);

int r; char *s; size_t n; const char *format; va_list vl;

Standard not (yet) (BSD)

Function Formated output of "n" characters into stringbuffer "s". A '\0' will be appended. Because of this strings longer than "n", are cut down to "n-1" characters. The outputformatstring "format" describes the outputformat followed by a parameterlist "vl". The function "vsnprintf64" supports the specifier "L" for long long int or unsigned long long int.

Return No. of printed characters or if "n" lower than "format" -1.

Example #define STORMAMIGA #define STORMAMIGA_NOWB

#include <stormamiga.h> #include <stdarg.h> #include <stdio.h>

int main__(void) { cchar *format = "This is a test."; char s[8]; int r; size_t n = sizeof s; va_list vl;

va_start (vl, format); r = vsnprintf64(s, n, format, vl); va_end (vl); printf64("%s\n", s); /* "This is" */ printf64("%d\n", r); /* "-1" */ return NULL; }

1.73 vsnprintf_

vsnprintf_ Formated output of "n" characters into stringbuffer "s". Functionreduced version of "vsnprintf".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = vsnprintf_ (s, n, format, vl);

int r; char *s; size_t n; const char *format; va_list vl;

Standard not (yet) (own development)

Function Formated output of "n" characters into stringbuffer "s". A "\0" will be appended. Because of this strings longer than "n", are cut down to "n-1" characters. The outputformatstring_ "format" describes the outputformat followed by a parameterlist "vl".

Return No. of printed characters or if "n" lower than "format" -1.

Example #define STORMAMIGA #define STORMAMIGA_NOWB

#include <stormamiga.h> #include <stdarg.h> #include <stdio.h>

int main__(void) { cchar *format = "This is a test."; char s[8]; int r; size_t n = sizeof s; va_list vl;

va_start (vl, format); r = vsnprintf_(s, n, format, vl); va_end (vl); printf_("%s\n", s); /* "This is" */ printf_("%d\n", r); /* "-1" */ return NULL; }

1.74 vsnprintf64_

vsnprintf64_ Formated output of "n" characters into stringbuffer "s". Extended version of "vsnprintf_".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = vsnprintf64_ (s, n, format, vl);

int r; char *s; size_t n; const char *format; va_list vl;

Standard not (yet) (own development)

Function Formated output of "n" characters into stringbuffer "s". A "\0" will be appended. Because of this strings longer than "n", are cut down to "n-1" characters. The outputformatstring_ "format" describes the outputformat followed by a parameterlist "vl". The function "vsnprintf64_" supports the specifier "L" for long long int or unsigned long long int.

Return No. of printed characters or if "n" lower than "format" -1.

Example #define STORMAMIGA #define STORMAMIGA_NOWB

#include <stormamiga.h> #include <stdarg.h> #include <stdio.h>

int main__(void) { cchar *format = "This is a test."; char s[8]; int r; size_t n = sizeof s; va_list vl;

va_start (vl, format); r = vsnprintf64_(s, n, format, vl); va_end (vl); printf64_("%s\n", s); /* "This is" */ printf64_("%d\n", r); /* "-1" */ return NULL; }

1.75 scanf64

scanf64 Read formated Input from standard-input "stdin". Extended version of "scanf".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = scanf64 (format, ...);

int r; const char *format;

Standard not (yet) (own development)

Function Read formated Input from standard-input "stdin". The inputformatstring "format" describes the inputformat followed by parameters. The function "scanf64" supports the specifier "L" for long long int or unsigned long long int.

Return No. of formatvalue or in case of failure a returnvalue lower than the No. of formatvalue.

1.76 scanf_

scanf_ Read formated Input from standard-input "stdin". Functionreduced version of"scanf".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = scanf_ (format, ...);

int r; const char *format;

Standard not (yet) (own development)

Function Read formated Input from standard-input "stdin". The inputformatstring_ "format" describes the inputformat followed by parameters.

Return No. of formatvalue or in case of failure a returnvalue lower than the No. of formatvalue.

1.77 scanf64_

scanf64_ Read formated Input from standard-input "stdin". Extended version of "scanf_".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = scanf64_ (format, ...);

int r; const char *format;

Standard not (yet) (own development)

Function Read formated Input from standard-input "stdin". The inputformatstring_ "format" describes the inputformat followed by parameters. The function "scanf64_" supports the specifier "L" for long long int or unsigned long long int.

Return No. of formatvalue or in case of failure a returnvalue lower than the No. of formatvalue.

1.78 fscanf64

fscanf64 Read formated Input from file "f". Extended version of "fscanf".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = fscanf64 (f, format, ...);

int r; FILE *f; const char *format;

Standard not (yet) (own development)

Function Read formated Input from file "f". The inputformatstring "format" describes the inputformat followed by parameters. The function "fscanf64" supports the specifier "L" for long long int or unsigned long long int.

Return No. of formatvalue or in case of failure a returnvalue lower than the No. of formatvalue.

1.79 fscanf_

fscanf_ Read formated Input from file "f". Functionreduced version of"fscanf".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = fscanf_ (f, format, ...);

int r; FILE *f; const char *format;

Standard not (yet) (own development)

Function Read formated Input from file "f". The inputformatstring_ "format" describes the inputformat followed by parameters.

Return No. of formatvalue or in case of failure a returnvalue lower than the No. of formatvalue.

1.80 fscanf64_

fscanf64_ Read formated Input from file "f". Extended version of "fscanf_".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = fscanf64_ (f, format, ...);

int r; FILE *f; const char *format;

Standard not (yet) (own development)

Function Read formated Input from file "f". The inputformatstring_ "format" describes the inputformat followed by parameters. The function "fscanf64_" supports the specifier "L" for long long int or unsigned long long int.

Return No. of formatvalue or in case of failure a returnvalue lower than the No. of formatvalue.

1.81 sscanf64

sscanf64 Read formated Input from stringbuffer "s". Extended version of "sscanf".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = sscanf64 (s, format, ...);

int r; char *s; const char *format;

Standard not (yet) (own development)

Function Read formated Input from stringbuffer "s". The inputformatstring "format" describes the inputformat followed by parameters. The function "sscanf64" supports the specifier "L" for long long int or unsigned long long int.

Return No. of formatvalue or in case of failure a returnvalue lower than the No. of formatvalue.

1.82 sscanf_

sscanf_ Read formated Input from stringbuffer "s". Functionreduced version of"sscanf".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = sscanf_ (s, format, ...);

int r; char *s; const char *format;

Standard not (yet) (own development)

Function Read formated Input from stringbuffer "s". The inputformatstring_ "format" describes the inputformat followed by parameters.

Return No. of formatvalue or in case of failure a returnvalue lower than the No. of formatvalue.

1.83 sscanf64_

sscanf64_ Read formated Input from stringbuffer "s". Extended version of "sscanf_".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = sscanf64_ (s, format, ...);

int r; char *s; const char *format;

Standard not (yet) (own development)

Function Read formated Input from stringbuffer "s". The inputformatstring_ "format" describes the inputformat followed by parameters. The function "sscanf64_" supports the specifier "L" for long long int or unsigned long long int.

Return No. of formatvalue or in case of failure a returnvalue lower than the No. of formatvalue.

1.84 vscanf

vscanf Read formated Input from standard-input "stdin".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = vscanf (format, vl);

int r; const char *format; va_list vl;

Standard not (yet) (BSD)

Function Read formated Input from standard-input "stdin". The inputformatstring "format" describes the inputformat followed by a parameterlist "vl".

Return No. of formatvalue or in case of failure a returnvalue lower than the No. of formatvalue.

1.85 vscanf64

vscanf64 Read formated Input from standard-input "stdin". Extended version of "scanf".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = vscanf64 (format, vl);

int r; const char *format; va_list vl;

Standard not (yet) (BSD)

Function Read formated Input from standard-input "stdin". The inputformatstring "format" describes the inputformat followed by a parameterlist "vl". The function "vscanf64" supports the specifier "L" for long long int or unsigned long long int.

Return No. of formatvalue or in case of failure a returnvalue lower than the No. of formatvalue.

1.86 vscanf_

vscanf_ Read formated Input from standard-input "stdin". Functionreduced version of "vscanf".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = vscanf_ (format, vl);

int r; const char *format; va_list vl;

Standard not (yet) (own development)

Function Read formated Input from standard-input "stdin". The inputformatstring_ "format" describes the inputformat followed by a parameterlist "vl".

Return No. of formatvalue or in case of failure a returnvalue lower than the No. of formatvalue.

1.87 vscanf64_

vscanf64_ Read formated Input from standard-input "stdin". Extended version of "scanf_".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = vscanf64_ (format, vl);

int r; const char *format; va_list vl;

Standard not (yet) (own development)

Function Read formated Input from standard-input "stdin". The inputformatstring_ "format" describes the inputformat followed by a parameterlist "vl". The function "vscanf64_" supports the specifier "L" for long long int or unsigned long long int.

Return No. of formatvalue or in case of failure a returnvalue lower than the No. of formatvalue.

1.88 vfscanf

vfscanf Read formated Input from file "f".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = vfscanf (f, format, vl);

int r; FILE *f; const char *format; va_list vl;

Standard not (yet) (BSD)

Function Read formated Input from file "f". The inputformatstring "format" describes the inputformat followed by a parameterlist "vl".

Return No. of formatvalue or in case of failure a returnvalue lower than the No. of formatvalue.

1.89 vfscanf64

vfscanf64 Read formated Input from file "f". Extended version of "vfscanf".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = vfscanf64 (f, format, vl);

int r; FILE *f; const char *format; va_list vl;

Standard not (yet) (BSD)

Function Read formated Input from file "f". The inputformatstring "format" describes the inputformat followed by a parameterlist "vl". The function "vfscanf64" supports the specifier "L" for long long int or unsigned long long int.

Return No. of formatvalue or in case of failure a returnvalue lower than the No. of formatvalue.

1.90 vfscanf_

vfscanf_ Read formated Input from file "f". Functionreduced version of"vfscanf".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = vfscanf_ (f, format, vl);

int r; FILE *f; const char *format; va_list vl;

Standard not (yet) (own development)

Function Read formated Input from file "f". The inputformatstring_ "format" describes the inputformat followed by a parameterlist "vl".

Return No. of formatvalue or in case of failure a returnvalue lower than the No. of formatvalue.

1.91 vfscanf64_

vfscanf64_ Read formated Input from file "f". Extended version of "vfscanf_".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = vfscanf64_ (f, format, vl);

int r; FILE *f; const char *format; va_list vl;

Standard not (yet) (own development)

Function Read formated Input from file "f". The inputformatstring_ "format" describes the inputformat followed by a parameterlist "vl". The function "vfscanf64_" supports the specifier "L" for long long int or unsigned long long int.

Return No. of formatvalue or in case of failure a returnvalue lower than the No. of formatvalue.

1.92 vsscanf

vsscanf Read formated Input from stringbuffer "s".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = vsscanf (s, format, vl);

int r; char *s; const char *format; va_list vl;

Standard not (yet) (BSD)

Function Read formated Input from stringbuffer "s". The inputformatstring "format" describes the inputformat followed by a parameterlist "vl".

Return No. of formatvalue or in case of failure a returnvalue lower than the No. of formatvalue.

1.93 vsscanf64

vsscanf64 Read formated Input from stringbuffer "s". Extended version of "vsscanf".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = vsscanf64 (s, format, vl);

int r; char *s; const char *format; va_list vl;

Standard not (yet) (BSD)

Function Read formated Input from stringbuffer "s". The inputformatstring "format" describes the inputformat followed by a parameterlist "vl". The function "vsscanf64" supports the specifier "L" for long long int or unsigned long long int.

Return No. of formatvalue or in case of failure a returnvalue lower than the No. of formatvalue.

1.94 vsscanf_

vsscanf_ Read formated Input from stringbuffer "s". Functionreduced version of"vsscanf".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = vsscanf_ (s, format, vl);

int r; char *s; const char *format; va_list vl;

Standard not (yet) (own development)

Function Read formated Input from stringbuffer "s". The inputformatstring_ "format" describes the inputformat followed by a parameterlist "vl".

Return No. of formatvalue or in case of failure a returnvalue lower than the No. of formatvalue.

1.95 vsscanf64_

vsscanf64_ Read formated Input from stringbuffer "s". Extended version of "vsscanf_".

Usage #include <stdio.h> (stdio_stormamiga.h)

r = vsscanf64_ (s, format, vl);

int r; char *s; const char *format; va_list vl;

Standard not (yet) (own development)

Function Read formated Input from stringbuffer "s". The inputformatstring_ "format" describes the inputformat followed by a parameterlist "vl". The function "vsscanf64_" supports the specifier "L" for long long int or unsigned long long int.

Return No. of formatvalue or in case of failure a returnvalue lower than the No. of formatvalue.

1.96 setbuffer

setbuffer Set a filebuffer

Usage #include <stdio.h> (stdio_stormamiga.h)

r = setbuffer (f, buf, size);

void r; FILE *f; char *buf; int size;

Standard not (yet) (4.3BSD)

Function Set buffer "buf" of length "size" for file "f".

The instruction "setbuffer" is also available as Inlinefunctions .

Return none

1.97 setlinebuf

setlinebuf Set a linebuffer

Usage #include <stdio.h> (stdio_stormamiga.h)

r = setlinebuf (f);

int r; FILE *f;

Standard not (yet) (4.3BSD)

Function Set a linebuffer for file "f".

The instruction "setlinebuf" is also available as Inlinefunctions .

Return Return always 0.

1.98 getw

getw Einlesen eines Wortes

Usage #include <stdio.h> (stdio_stormamiga.h)

r = getw(f);

int r; FILE *f;

Standard not (yet) (BSD)

Function Liest aus der Datei "f" das nächste Wort.

Return Das nächste Wort aus der Datei "f" oder "EOF" falls ein Fehler auftrat oder das Ende der Datei erreicht war.

1.99 putw

putw Ausgabe eines Wortes

Usage #include <stdio.h> (stdio_stormamiga.h)

r = putw(w, f);

int r; int w; FILE *f;

Standard not (yet) (Version 6 AT&T UNIX)

Function Das Wort "w" wird in die Datei "f" ausgegeben.

Return Das ausgegebene Wort oder im Fehlerfall "EOF".

1.100 access

access Überprüft den Zugriffsmodus von Dateien

Usage #include <unistd.h>

r = access(file, mode);

int r; const char *file; int mode;

Standard IEEE Std1003.1-1988 ("POSIX")

Function Es wird überprüft ob der Zugriffsmodus "mode" bei der Datei "name" möglich ist oder nicht.

Bei Definition von STORMAMIGA_UNIXPATH können für diese Funktionen auch Pfadangaben im Unix-Format verwendet werden.

Return Im Fehlerfall -1, sonst 0.

1.101 chdir

chdir Verzeichnis wechseln

Usage #include <unistd.h>

r = chdir(path);

int r; const char *path;

Standard IEEE Std1003.1-1988 ("POSIX")

Function Es wird vom aktuellen Arbeitsverzeichnis zum Verzeichnis "path" gewechselt. Beim Beenden des Programmes wird das Originalverzeichnis wieder hergestellt.

Bei Definition von STORMAMIGA_UNIXPATH können für diese Funktionen auch Pfadangaben im Unix-Format verwendet werden.

Return Im Fehlerfall -1, sonst 0.

1.102 rmdir

rmdir Verzeichnis löschen

Usage #include <unistd.h>

r = rmdir(path);

int r; char *path;

Standard not (yet) (4.3BSD)

Function Das Verzeichnis "path" wird gelöscht.

Bei Definition von STORMAMIGA_UNIXPATH können für diese Funktionen auch Pfadangaben im Unix-Format verwendet werden.

Return Im Fehlerfall -1, sonst 0.

1.103 unlink

unlink Link löschen

Usage #include <unistd.h>

r = unlink(name);

int r; const char *name;

Standard not (yet) (Version 6 AT&T UNIX)

Function Es wird der Link "name" gelöscht.

Bei Definition von STORMAMIGA_UNIXPATH können für diese Funktionen auch Pfadangaben im Unix-Format verwendet werden.

Return Im Fehlerfall -1, sonst 0.

1.104 sleep

sleep stoppt den aktuellen Prozeß

Usage #include <unistd.h>

r = sleep(n);

void r; unsigned int n;

Standard not (yet) (Version 7 AT&T UNIX)

Function Der aktuelle Prozeß wird für "n" Sekunden gestoppt.

Return none

1.105 usleep

usleep stoppt den aktuellen Prozeß

Usage #include <unistd.h>

r = usleep(n);

void r; unsigned int n;

Standard not (yet) (4.3BSD)

Function Der aktuelle Prozeß wird für "n" Mikrosekunden gestoppt.

Return none

1.106 getcwd

getcwd Verzeichnisname ermitteln

Usage #include <unistd.h>

r = getcwd(buf, n);

char *r; char *buf; size_t n;

Standard ANSI C3.159-1989 ("ANSI C")

Function Es wird der Name des aktuellen Arbeitsverzeichnisses nach "buf" kopiert und ein Zeiger ("r") auf "buf" zurückgegeben. Die Größe von "buf" wird mit "n" angegeben.

Return Im Fehlerfall 0, sonst ein Zeiger auf "buf".

1.107 getwd

getwd Verzeichnisname ermitteln

Usage #include <unistd.h>

r = getwd(buf);

char *r; char *buf;

Standard not (yet) (4.0BSD)

Function Es wird der Name des aktuellen Arbeitsverzeichnisses nach "buf" kopiert und ein Zeiger ("r") auf "buf" zurückgegeben.

Return Im Fehlerfall 0, sonst ein Zeiger auf "buf".

1.108 mktemp

mktemp erzeugt einen temporären Dateinamen

Usage #include <unistd.h>

r = mktemp(template);

char *r; char *template;

Standard not (yet) (Version 7 AT&T UNIX)

Function Es wird der temporäre Dateiname "template" erzeugt und ein Zeiger darauf zurückgegeben.

Return Im Fehlerfall 0, sonst ein Zeiger auf "template".

1.109 close

close schließt einen File Descriptor

Usage #include <unistd.h>

r = close(fd);

int r; int fd;

Standard IEEE Std1003.1-1988 ("POSIX")

Function Es wird der File Descriptor "fd" geschlossen.

Return Im Fehlerfall -1, sonst 0.

1.110 lseek

lseek positioniert den Schreib- und Lesezeiger

Usage #include <unistd.h>

r = lseek(fd, offset, whence);

off_t r; int fd; off_t offset; int whence;

Standard IEEE Std1003.1-1988 ("POSIX")

Function Der Schreib- und Lesezeiger von "fd" wird an eine neue Position gesetzt. Der Parameter "offset" gibt die Position relativ zur Startposition, die sich aus dem Modus "whence" ergibt, an.

Als Moduswerte sind erlaubt:

SEEK_CUR: Der Offset wird von der aktuellen Position gezählt und kann daher positiv oder negativ sein. SEEK_SET: Der Offset wird vom Anfang der Datei gezählt und sollte positiv sein. SEEK_END: Der Offset wird vom Ende der Datei gezählt und sollte negativ sein.

Return Im Fehlerfall -1, sonst die aktuelle Position.

1.111 read

read Eingaben lesen

Usage #include <unistd.h>

r = read(fd, buf, n);

ssize_t r; int fd; void *buf; size_t n;

Standard IEEE Std1003.1-1988 ("POSIX")

Function Es werden maximal "n" Zeichen aus "fd" in "buf" gelesen.

Return Im Fehlerfall -1, sonst die Anzahl der gelesenen Zeichen.

1.112 write

write Ausgaben schreiben

Usage #include <unistd.h>

r = write(fd, buf, n);

ssize_t r; int fd; const void *buf; size_t n;

Standard IEEE Std1003.1-1988 ("POSIX")

Function Es werden "n" Zeichen aus "buf" in "fd" geschrieben.

Return Im Fehlerfall -1, sonst die Anzahl der geschriebenen Zeichen.

1.113 creat

creat erzeugt eine neue Datei

Usage #include <fcntl.h>

r = creat(path, mode);

int r; const char *path; mode_t mode;

Standard (noch) keiner (Version 6 AT&T UNIX)

Function Es wird die Datei "path" in dem angegebenem Modus "mode" erzeugt.

Bei Definition von STORMAMIGA_UNIXPATH können für diese Funktionen auch Pfadangaben im Unix-Format verwendet werden.

Return Im Fehlerfall -1, sonst den File Descriptor.

1.114 open

open erzeugt oder öffnet eine Datei

Usage #include <fcntl.h>

r = open(path, flags, ...);

int r; const char *path; int flags;

Standard (noch) keiner (Version 6 AT&T UNIX)

Function Es wird die Datei "path" in dem bei "flags" angegebenem Modus erzeugt oder geöffnet.

Bei Definition von STORMAMIGA_UNIXPATH können für diese Funktionen auch Pfadangaben im Unix-Format verwendet werden.

Return Im Fehlerfall -1, sonst den File Descriptor.

1.115 closedir

closedir Verzeichnis schließen

Usage #include <dirent.h>

r = closedir(dirp);

int r; DIR *dirp;

Standard not (yet) (4.2BSD)

Function Es wird das, in "dirp" benannte, Verzeichnis geschlossen und die zugehörige Struktur freigegeben.

Return Im Fehlerfall -1, sonst 0.

1.116 opendir

opendir Verzeichnis öffnen

Usage #include <dirent.h>

r = opendir(name);

DIR *r; const char *name;

Standard not (yet) (4.2BSD)

Function Es wird das Verzeichnis "name" geöffnet.

Bei Definition von STORMAMIGA_UNIXPATH können für diese Funktionen auch Pfadangaben im Unix-Format verwendet werden.

Return Im Fehlerfall 0, sonst ein Zeiger auf den directory stream.

1.117 readdir

readdir Verzeichnis lesen

Usage #include <dirent.h>

r = readdir(dirp);

struct dirent *r; DIR *dirp;

Standard not (yet) (4.2BSD)

Function Es wird der nächste Verzeichniseintrag, des in "dirp" benannten Verzeichnisses, gelesen.

Return Im Fehlerfall oder am Ende des Verzeichnisses 0, sonst ein Zeiger auf den nächsten Verzeichniseintrag.

1.118 rewinddir

rewinddir Verzeichnisposition zurücksetzen

Usage #include <dirent.h>

r = rewinddir(dirp);

void r; DIR *dirp;

Standard not (yet) (4.2BSD)

Function Es wird die Position, des in "dirp" benannten Verzeichnisses, an den Anfang des Verzeichnisses zurückgesetzt.

Return none

1.119 getrusage

getrusage ermittelt Informationen zum aktuellen Prozess oder dessen Unterprozesse

Usage #include <sys/resource.h>

r = getrusage(who, rusage);

int r; int who; struct rusage *rusage;

Standard not (yet) (4.2BSD)

Function Es werden Informationen zum aktuellen Prozess oder zu dessen Unterprozessen ermittelt und in "rusage" ausgegeben. Mit "who" wird angegeben ob Informationen zum aktuellen Prozess oder zu dessen Unterprozessen ermittelt werden sollen. Für den aktuellen Prozess wird für "who" "RUSAGE_SELF", für dessen Unterprozesse wird "RUSAGE_CHILDREN" angegeben.

Return Im Fehlerfall -1, sonst 0.

1.120 chmod

chmod ändert den Zugriffsmodus von Dateien

Usage #include <sys/stat.h>

r = chmod(path, mode);

int r; const char *path; mode_t mode;

Standard IEEE Std1003.1-1988 ("POSIX")

Function Der Zugriffsmodus von "path" wird in den Modus "mode" geändert.

Bei Definition von STORMAMIGA_UNIXPATH können für diese Funktionen auch Pfadangaben im Unix-Format verwendet werden.

Return Im Fehlerfall -1, sonst 0.

1.121 mkdir

mkdir Verzeichnis anlegen

Usage #include <sys/stat.h>

r = mkdir(path, mode);

int r; const char *path; mode_t mode;

Standard IEEE Std1003.1-1988 ("POSIX")

Function Es wird das Verzeichnis "path" mit dem Zugriffsmodus "mode" angelegt.

Bei Definition von STORMAMIGA_UNIXPATH können für diese Funktionen auch Pfadangaben im Unix-Format verwendet werden.

Return Im Fehlerfall -1, sonst 0.

1.122 stat

stat Status von Dateien ermitteln

Usage #include <sys/stat.h>

r = stat(path, sb);

int r; const char *path; struct stat *sb;

Standard IEEE Std1003.1-1988 ("POSIX")

Function Es werden Informationen der Datei "path" ermittelt. "sb" ist ein Zeiger auf die Struktur stat.

Bei Definition von STORMAMIGA_UNIXPATH können für diese Funktionen auch Pfadangaben im Unix-Format verwendet werden.

Return Im Fehlerfall -1, sonst 0.

1.123 lstat

lstat Status von Links ermitteln

Usage #include <sys/stat.h>

r = lstat(path, sb);

int r; const char *path; struct stat *sb;

Standard not (yet) (4.2BSD)

Function Es werden Informationen des Links "path" ermittelt. "sb" ist ein Zeiger auf die Struktur stat.

Bei Definition von STORMAMIGA_UNIXPATH können für diese Funktionen auch Pfadangaben im Unix-Format verwendet werden.

Return Im Fehlerfall -1, sonst 0.

1.124 fstat

fstat Status von Dateien ermitteln

Usage #include <sys/stat.h>

r = fstat(fd, sb);

int r; int fd; struct stat *sb;

Standard IEEE Std1003.1-1988 ("POSIX")

Function Es werden Informationen über "fd" ermittelt. "sb" ist ein Zeiger auf die Struktur stat.

Return Im Fehlerfall -1, sonst 0.

1.125 strcoll

strcoll compare two strings using the actual language

Usage #include <string.h> (string_stormamiga.h)

r = strcoll (s1, s2);

int r; const char *s1; const char *s2;

Standard ANSI C3.159-1989 ("ANSI C")

Function Compare the strings "s1" and "s2" character by character. Because "stormamiga.lib" does not use ANSI-Localization, the actual language is not supported. This function is only for compatibility reasons implemented (e.g.: GNU C).

Return < 0 if s1 < s2 = 0 if s1 = s2 > 0 if s1 > s2

1.126 strxfrm

strxfrm copy a string using the actual language

Usage #include <string.h> (string_stormamiga.h)

r = strxfrm (dest, source, n);

int r; char *dest; const char *source; size_t n;

Standard ANSI C3.159-1989 ("ANSI C")

Function Copies max. "n" bytes of string "source" to "dest". Because "stormamiga.lib" does not use ANSI-Localization, the actual language is not supported. This function is only for compatibility reasons implemented (e.g.: GNU C).

Return the length of the copied string "source" with no '\0'

1.127 bcmp

bcmp compare two memoryblocks using a maximal size

Usage #include <string.h> (string_stormamiga.h)

r = bcmp (b1, b2, n);

int r; const void *b1; const void *b2; size_t n;

Standard not (yet) (4.2BSD)

Function Compare memoryblocks "b1" and "b2" byte by byte with the size of maximal "n". The two blocks may overlap.

Return < 0 if b1 < b2 = 0 if b1 = b2 > 0 if b1 > b2

1.128 cmpmem

cmpmem compare two memoryblocks using a maximal size

Usage #include <string.h> (string_stormamiga.h)

r = cmpmem (b1, b2, n);

int r; const void *b1; const void *b2; size_t n;

Standard not (yet) (UNIX)

Function Compare memoryblocks "b1" and "b2" byte by byte with the size of maximal "n". The two blocks may overlap.

Return < 0 if b1 < b2 = 0 if b1 = b2 > 0 if b1 > b2

1.129 bcopy

bcopy copy memory

Usage #include <string.h> (string_stormamiga.h)

r = bcopy (source, dest, n);

void *r; const void *source; void *dest; size_t n;

Standard not (yet) (4.2BSD)

Function Copy "n" bytes of memoryblock "source" to "dest". The two blocks may overlap. If "n" is 0 nothing will be copied.

Return a pointer to the memoryblock "dest"

1.130 movmem

movmem copy memory

Usage #include <string.h> (string_stormamiga.h)

r = movmem (source, dest, n);

void *r; const void *source; void *dest; size_t n;

Standard not (yet) (UNIX)

Function Copy "n" bytes of memoryblock "source" to "dest". The two blocks may overlap. If "n" is 0 nothing will be copied.

Return a pointer to the memoryblock "dest"

1.131 bzero

bzero overwrite memoryblock with NULL-bytes

Usage #include <string.h> (string_stormamiga.h)

r = bzero (b, n);

void *r; void *b; size_t n;

Standard not (yet) (4.3BSD)

Function Overwrite memoryblock "b" with "n" NULL-bytes.

Return a pointer to the memoryblock "b"

1.132 clrmem

clrmem overwrite memoryblock with NULL-bytes

Usage #include <string.h> (string_stormamiga.h)

r = clrmem (b, n);

void *r; void *b; size_t n;

Standard not (yet) (Specialfunction from "DICE")

Function Overwrite memoryblock "b" with "n" NULL-bytes.

Return a pointer to the memoryblock "b"

1.133 ffs

ffs find first set bit in a bit-string

Usage #include <string.h> (string_stormamiga.h)

r = ffs (value);

int r; int value;

Standard not (yet) (4.3BSD)

Function Find first set bit in bit-string "value" and return its index.

Return index of bit-string "value"

1.134 index

index search for the first appearance of a character in a string

Usage #include <string.h> (string_stormamiga.h)

r = index (s, c);

char *r; const char *s; int c;

Standard not (yet) (Version 6 AT&T UNIX)

Function Search for the first appearance of character "c" in string "s" und return a pointer to the first "c". Return 0 if "c" was not found.

Return a pointer to first appearance of character "c" or 0

1.135 rindex

rindex search for the last appearance of a character in a string

Usage #include <string.h> (string_stormamiga.h)

r = rindex (s, c);

char *r; const char *s; int c;

Standard not (yet) (Version 6 AT&T UNIX)

Function Search for the last appearance of character "c" in string "s" und return a pointer to the last "c". Return 0 if "c" was not found.

Return a pointer to last appearance of character "c" or 0

1.136 memccpy

memccpy copy memory

Usage #include <string.h> (string_stormamiga.h)

r = memccpy (dest, source, c, n);

void *r; void *dest; const void *source; int c; size_t n;

Standard not (yet) (4.3BSD)

Function Copy memory from "source" to "dest". If character "c" appears in "source" memcopy stops and returns a pointer to the byte behind the copy of "c". Otherwise it copies "n" bytes and returns 0.

Return a pointer to the character behind the copy of "c" in memoryblock "dest" or 0

Example #define STORMAMIGA #define STORMAMIGA_NOWB

#include <stormamiga.h> #include <string.h> #include <stdio.h>

int main__(void) { cvoid *source = "This is a test."; void *dest[50]; int c = ’e’; size_t n = sizeof dest;

memccpy(dest, source, c, n); puts((char *)dest); /* "This is a te" */ return NULL; }

1.137 setmem

setmem set a memoryblock to a value

Usage #include <string.h> (string_stormamiga.h>

r = setmem (b, n, c);

void *r; void *b; size_t n; int c;

Standard (not) yet (UNIX)

Function These function fill the memoryblock "b" with the specified character "c". "n" bytes are filled.

Return a pointer to the memoryblock "b"

1.138 strncpyn

strncpyn copy a string with length limitation

Usage #include <string.h> (string_stormamiga.h)

r = strncpyn (dest, source, n);

char *r; char *dest; const char *source; size_t n;

Standard not (yet) (own development)

Function This function copies maximal "n" characters of string "source" to string "dest". A '\0' is always appended. Because of this strings longer than "n" are cut down to "n-1" characters. Strings shorter than "n" are filled up with extra '\0' until "n". A pointer to the "dest" string is returned.

Return a pointer to the destination string "dest"

Example #define STORMAMIGA #define STORMAMIGA_NOWB

#include <stormamiga.h> #include <string.h> #include <stdio.h>

int main__(void) { cchar *source = "This is a test."; char dest[10]; size_t n = 8;

strncpyn(dest, source, n); puts(dest); /* "This is" */ return NULL; }

1.139 stccpy

stccpy copy a string with length limitation

Usage #include <string.h> (string_stormamiga.h)

r = stccpy (dest, source, n);

int r; char *dest; const char *source; int n;

Standard not (yet) (UNIX)

Function This function copies maximal "n" characters of string "source" to string "dest". A '\0' is always appended.

Return actual number of bytes placed in the "dest" area, including the '\0'

Example #define STORMAMIGA #define STORMAMIGA_NOWB

#include <stormamiga.h> #include <string.h> #include <stdio.h>

int main__(void) { char dest[256];

stccpy(dest,"Hello, ",256); stccpy(&dest[strlen(dest)],"my name is ",256-strlen(dest)); stccpy(&dest[strlen(dest)],"Flo.",256-strlen(dest)); puts(dest); /* "Hello, my name is Flo." */ return NULL; }

1.140 strsep

strsep split strings

Usage #include <string.h> (string_stormamiga.h)

r = strsep (s1, s2);

char *r; char **s1; char *s2;

Standard not (yet) (BSD)

Function This function searches string "*s1" (with terminating 0) for the first appearance of any character from string "s2". A matching character will be replaced with '\0' and the next character in "*s1" is marked. The original value of string "*s1" is returned. If no character matches, the function will return 0.

Return original value of string "*s1" or 0

Example #define STORMAMIGA #define STORMAMIGA_NOWB

#include <stormamiga.h> #include <string.h> #include <stdio.h>

int main__(void) { char *s1 = "This is a test."; char *s2 = " "; char *r;

r = strsep(&s1, s2); puts(s1); /* "is a test." */ puts(r); /* "This" */ return NULL; }

1.141 swab

swab swap neighbored bytes

Usage #include <string.h> (string_stormamiga.h)

r = swab (source, dest, n);

void r; const void *source; void *dest; size_t n;

Standard not (yet) (Version 7 AT&T UNIX)

Function Copies "n" bytes from "source" to "dest" and swap the two neighbored bytes. "n" has to be even.

Return none

1.142 strnicmp

strnicmp compare two strings using a maximal length

Usage #include <string.h> (string_stormamiga.h)

r = strnicmp (s1, s2, n);

int r; const char *s1; const char *s2; size_t n;

Standard not (yet) (own development)

Function Compare the strings "s1" and "s2" character by character up to the character with index "n" ignoring the case. Due to portability Umlauts are not supported.

Return < 0 if s1 < s2 = 0 if s1 = s2 > 0 if s1 > s2

1.143 strcasecmp

strcasecmp compare two strings

Usage #include <string.h> (string_stormamiga.h)

r = strcasecmp (s1, s2);

int r; const char *s1; const char *s2;

Standard not (yet) (BSD)

Function Compare the strings "s1" and "s2" character by character ignoring the case. Due to portability Umlauts are not supported.

Return < 0 if s1 < s2 = 0 if s1 = s2 > 0 if s1 > s2

1.144 strncasecmp

strncasecmp compare two strings using a maximal length

Usage #include <string.h> (string_stormamiga.h)

r = strncasecmp (s1, s2, n);

int r; const char *s1; const char *s2; size_t n;

Standard not (yet) (BSD)

Function Compare the strings "s1" and "s2" character by character up to the character with index "n" ignoring the case. Due to portability Umlauts are not supported.

Return < 0 if s1 < s2 = 0 if s1 = s2 > 0 if s1 > s2

1.145 strlower

strlower convert all upper case letters of a string to lower case

Usage #include <string.h> (string_stormamiga.h)

r = strlower (s);

char *r; char *s;

Standard not (yet) (BSD)

Function All upper case characters of string "s" are converted to lower case. Other characters stay untouched. Due to portability Umlauts are not supported.

Return the converted string "s"

1.146 strupper

strupper convert all lower case letters of a string to upper case

Usage #include <string.h> (string_stormamiga.h)

r = strupper (s);

char *r; char *s;

Standard not (yet) (BSD)

Function All lower case characters of string "s" are converted to upper case. Other characters stay untouched. Due to portability Umlauts are not supported.

Return the converted string "s"

1.147 stricmp_d

stricmp_d compare two strings german version of "stricmp"

Usage #include <string.h> (string_stormamiga.h)

r = stricmp_d (s1, s2);

int r; const char *s1; const char *s2;

Standard not (yet) (own development)

Function Compare the two strings "s1" and "s2" character by character ignoring their case. The german Umlauts "ä", "ö", "ü", "Ä", "Ö" and "Ü" are supported. Because "ß" cannot be capitalized, it is not supported.

Look at german functions .

Return < 0 if s1 < s2 = 0 if s1 = s2 > 0 if s1 > s2

1.148 strnicmp_d

strnicmp_d compare two strings using a maximal length german version of "strnicmp"

Usage #include <string.h> (string_stormamiga.h)

r = strnicmp_d (s1, s2, n);

int r; const char *s1; const char *s2; size_t n;

Standard not (yet) (own development)

Function Compare the two strings "s1" and "s2" character by character up to the character with index "n" ignoring their case. The german Umlauts "ä", "ö", "ü", "Ä", "Ö" and "Ü" are supported. Because "ß" cannot be capitalized, it is not supported.

Look at german functions .

Return < 0 if s1 < s2 = 0 if s1 = s2 > 0 if s1 > s2

1.149 strcasecmp_d

strcasecmp_d compare two strings german version of "strcasecmp"

Usage #include <string.h> (string_stormamiga.h)

r = strcasecmp_d (s1, s2);

int r; const char *s1; const char *s2;

Standard not (yet) (own development)

Function Compare the two strings "s1" and "s2" character by character ignoring their case. The german Umlauts "ä", "ö", "ü", "Ä", "Ö" and "Ü" are supported. Because "ß" cannot be capitalized, it is not supported.

Look at german functions .

Return < 0 if s1 < s2 = 0 if s1 = s2 > 0 if s1 > s2

1.150 strncasecmp_d

strncasecmp_d compare two strings using a maximal length german version of "strncasecmp"

Usage #include <string.h> (string_stormamiga.h)

r = strncasecmp_d (s1, s2, n);

int r; const char *s1; const char *s2; size_t n;

Standard not (yet) (own development)

Function Compare the two strings "s1" and "s2" character by character up to the character with index "n" ignoring their case. The german Umlauts "ä", "ö", "ü", "Ä", "Ö" and "Ü" are supported. Because "ß" cannot be capitalized, it is not supported.

Look at german functions .

Return < 0 if s1 < s2 = 0 if s1 = s2 > 0 if s1 > s2

1.151 strlower_d

strlower_d convert all upper case letters of a string to lower case german version of "strlower"

Usage #include <string.h> (string_stormamiga.h)

r = strlower_d (s);

char *r; char *s;

Standard not (yet) (own development)

Function All upper case characters of string "s" are converted to lower case. Other characters stay untouched. The german Umlauts "ä", "ö", "ü", "Ä", "Ö" and "Ü" are supported. Because "ß" cannot be capitalized, it is not supported.

Look at german functions .

Return the converted string "s"

1.152 strupper_d

strupper_d convert all lower case letters of a string to upper case german version of "strupper"

Usage #include <string.h> (string_stormamiga.h)

r = strupper_d (s);

char *r; char *s;

Standard not (yet) (own development)

Function All lower case characters of string "s" are converted to upper case. Other characters stay untouched. The german Umlauts "ä", "ö", "ü", "Ä", "Ö" and "Ü" are supported. Because "ß" cannot be capitalized, it is not supported.

Look at german functions .

Return the converted string "s"

1.153 strlwr_d

strlwr_d convert all upper case letters of a string to lower case german version of "strlwr"

Usage #include <string.h> (string_stormamiga.h)

r = strlwr_d (s);

char *r; char *s;

Standard not (yet) (own development)

Function All upper case characters of string "s" are converted to lower case. Other characters stay untouched. The german Umlauts "ä", "ö", "ü", "Ä", "Ö" and "Ü" are supported. Because "ß" cannot be capitalized, it is not supported.

Look at german functions .

Return the converted string "s"

1.154 strupr_d

strupr_d convert all lower case letters of a string to upper case german version of "strupr"

Usage #include <string.h> (string_stormamiga.h)

r = strupr_d (s);

char *r; char *s;

Standard not (yet) (own development)

Function All lower case characters of string "s" are converted to upper case. Other characters stay untouched. The german Umlauts "ä", "ö", "ü", "Ä", "Ö" and "Ü" are supported. Because "ß" cannot be capitalized, it is not supported.

Look at german functions .

Return the converted string "s"

1.155 strdup

strdup save a copy of a string

Usage #include <string.h> (string_stormamiga.h)

r = strdup (s);

char *r; const char *s;

Standard not (yet) (BSD)

Function "strdup" allocates sufficient memory for a copy of string "s", copies the string and return a pointer to it. This pointer can be used later as an arguement of free .

Return a pointer to a copy of strings "s"

1.156 stpchr

stpchr search for a character in a string

Usage #include <string.h> (string_stormamiga.h)

r = stpchr (s, c);

char *r; const char *s; int c;

Standard not (yet) (UNIX)

Function This searches for the character "c" within the string pointed to by "s". The terminating NULL at the end of "s" is NOT included in the search. A pointer to the first occurance of "c" in s is returned or NULL if "c" could not be found.

Return a pointer to the first occurance of the character "c" or 0

Example #define STORMAMIGA #define STORMAMIGA_NOWB

#include <stormamiga.h> #include <string.h> #include <stdio.h>

int main__(void) { char *s = "This is a test."; char *r; int c = ' ';

r = stpchr(s, c); puts(r); /* " is a test" */ return NULL; }

1.157 stpcpy

stpcpy copy a string

Usage #include <string.h> (string_stormamiga.h)

r = stpcpy (dest, source);

char *r; char *dest; const char *source;

Standard not (yet) (UNIX)

Function This function copies the NULL terminated string pointed to by "s" to the buffer "dest". The NULL is copied. A pointer to the NULL character at the end of the copied string in "dest" is returned.

Return a pointer to end of the NULL character at the end of the copied string in "dest"

Example #define STORMAMIGA #define STORMAMIGA_NOWB

#include <stormamiga.h> #include <string.h> #include <stdio.h>

int main__(void) { cchar *source1 = "This is a "; cchar *source2 = "test."; char dest[50]; char *r;

r = stpcpy(dest, source1); stpcpy(r, source2); puts(dest); /* "This is a test." */ return NULL; }

1.158 strins

strins insert one string within another

Usage #include <string.h> (string_stormamiga.h)

r = strins (dest, source);

void r; char *dest; const char *source;

Standard not (yet) (Specialfunction from "DICE")

Function strins inserts string "source" into "dest" by shifting the string in "dest" over strlen(s) spaces and then copying "source" into the newly made hold (except for the NULL, of course).

Return none

Example #define STORMAMIGA #define STORMAMIGA_NOWB

#include <stormamiga.h> #include <string.h> #include <stdio.h>

int main__(void) { cchar *source1 = "This is a test."; cchar *source2 = "new "; char dest[50];

strcpy(dest, source1); strins(dest + 10, source2); puts(dest); /* "This is a new test." */ return NULL; }

1.159 strbpl

strbpl unpack a string-array buffer into an array of pointers

Usage #include <string.h> (string_stormamiga.h)

r = strbpl(av, n, sary);

int r; char **av; int n; const char *sary;

Standard not (yet) (Specialfunction from "DICE")

Function strbpl unpacks the string-array "sary" into an array of string pointers. The string array "sary" is a series of NULL terminated strings strung together and terminated by a final NULL. A pointer to each string is placed in the arary-of-pointers "av" with a final NULL entry assuming the number of strings does not exceed (max-1).

Return number of pointers loaded into the av array notincluding the final NULL

Example #define STORMAMIGA #define STORMAMIGA_NOWB

#include <stormamiga.h> #include <string.h> #include <stdio.h>

int main__(void) { char *sary = "This\0is\0a\0test.\0\0"; char *av[17]; int n = sizeof av / sizeof av[0];

strbpl(av, n, sary); puts(av[0]); /* "This" */ puts(av[1]); /* "is" */ puts(av[2]); /* "a" */ puts(av[3]); /* "test." */ return NULL; }

1.160 isalnum_d

isalnum_d Tests for a character or number. german version of "isalnum"

Usage #include <ctype.h> (ctype_stormamiga.h)

r = isalnum_d (ch);

int r; int ch;

Standard not (yet) (own development)

Function This function tests, if "ch" is a character or a number. The german Umlauts "ä", "ö", "ü", "ß", "Ä", "Ö" and "Ü" are supported.

Look at german functions .

Return 0 if "ch" is neither a character nor a number, otherwise a value != 0

1.161 isalpha_d

isalpha_d Tests for a character german version of "isalpha"

Usage #include <ctype.h> (ctype_stormamiga.h)

r = isalpha_d (ch);

int r; int ch;

Standard not (yet) (own development)

Function This function tests, if "ch" is a character. The german Umlauts "ä", "ö", "ü", "ß", "Ä", "Ö" and "Ü" are supported.

Look at german functions .

Return 0 if "ch" is not a character, otherwise a value != 0

1.162 islower_d

islower_d Tests for small letters. german version of "islower"

Usage #include <ctype.h> (ctype_stormamiga.h)

r = islower_d (ch);

int r; int ch;

Standard not (yet) (own development)

Function This function tests, if "ch" is a small letter. The german Umlauts "ä", "ö", "ü", "ß", "Ä", "Ö" and "Ü" are supported.

Look at german functions .

Return 0 if "ch" is not a small character, otherwise a value != 0

1.163 isprint_d

isprint_d Tests for printable characters. german version of "isprint"

Usage #include <ctype.h> (ctype_stormamiga.h)

r = isprint_d (ch);

int r; int ch;

Standard not (yet) (own development)

Function This function tests, if "ch" is a printable character. The german Umlauts "ä", "ö", "ü", "ß", "Ä", "Ö" and "Ü" are supported.

Look at german functions .

Return 0 if "ch" is not a printable character, otherwise a value != 0

1.164 ispunct_d

ispunct_d Tests for special characters. german version of "ispunct"

Usage #include <ctype.h> (ctype_stormamiga.h)

r = ispunct_d (ch);

int r; int ch;

Standard not (yet) (own development)

Function This function tests, if "ch" is a special character. The german Umlauts "ä", "ö", "ü", "ß", "Ä", "Ö" and "Ü" belong not to the special characters.

Look at german functions .

Return 0 if "ch" is not a special character, otherwise a value != 0

1.165 isupper_d

isupper_d Tests for a capitalized letter german version of "isupper"

Usage #include <ctype.h> (ctype_stormamiga.h)

r = isupper_d (ch);

int r; int ch;

Standard not (yet) (own development)

Function This function tests, if "ch" is a capitalized character. The german Umlauts "ä", "ö", "ü", "Ä", "Ö" and "Ü" are supported. Because "ß" cannot be capitalized, it is not supported.

Look at german functions .

Return 0 if "ch" is not a capitalized character, otherwise a value != 0

1.166 tolower_d

tolower_d Convert capital letters into small ones. german version of "tolower"

Usage #include <ctype.h> (ctype_stormamiga.h)

r = tolower_d (ch);

int r; int ch;

Standard not (yet) (own development)

Function If "ch" is a capital letter, it will be converted into a small one. Otherwise it changes nothing. The german Umlauts "ä", "ö", "ü", "Ä", "Ö" and "Ü" are supported. Because "ß" cannot be capitalized, it is not supported.

Look at german functions .

Return converted character

1.167 toupper_d

toupper_d Convert small letters into capital ones. german version of "toupper"

Usage #include <ctype.h> (ctype_stormamiga.h)

r = toupper_d (ch);

int r; int ch;

Standard not (yet) (own development)

Function If "ch" is a small letter, it will be converted into a capital one. Otherwise it changes nothing. The german Umlauts "ä", "ö", "ü", "Ä", "Ö" and "Ü" are supported. Because "ß" cannot be capitalized, it is not supported.

Look at german functions .

Return converted character

1.168 assert_

assert Test for a specific condition and interrupt the programm.

Usage #include <assert.h> (assert_stormamiga.h)

assert_ (x);

int x;

Standard not (yet) (own development)

Function Look at assert . The only difference to "assert" is that "assert_" uses "printf_" and not "printf".

1.169 strftime

strftime Formated output of date and time into stringbuffer "s". Extended version of "strftime" of "storm.lib".

Usage #include <time.h>

r = strftime (s, size, format, tp);

size_t r; char *s; size_t size; const char *format; const struct tm *tp;

Standard ANSI C3.159-1989 ("ANSI C")

Function Formated output of date and time into stringbuffer "s". The timeformatstring "format" describes the outputformat.

Return No. of printed characters.

1.170 strftime_d

strftime_d Formated output of date and time into stringbuffer "s". Extended german version of "strftime" of "storm.lib".

Usage #include <time.h> (time_stormamiga.h)

r = strftime_d (s, size, format, tp);

size_t r; char *s; size_t size; const char *format; const struct tm *tp;

Standard not (yet) (own development)

Function Formated output of date and time into stringbuffer "s". The timeformatstring "format" describes the outputformat. Names of months and days are printed in german.

Look at german functions .

Return No. of printed characters.

1.171 asctime_d

asctime_d Generate a string from date and time. german Version of "asctime"

Usage #include <time.h> (time_stormamiga.h)

tp = asctime_d (t);

char *tp; const struct tm *t;

Standard not (yet) (own development)

Function "asctime_d" converts the time from "*t" to a string of the form "Mit Okt 07 01:03:42 1992". This string is internally buffered and the function returns a pointer to this buffer. Names of months and days are printed in german.

Look at german functions .

Return This function returns an ASCII-text with the exact length of 26 characters. The format of this text is: "DDD MMM dd hh:mm:ss YYYY"

DDD = day of week, MMM = month, dd = day of month, hh:mm:ss = hour:minute:second and YYYY = year. For example: "Mit Okt 25 12:05:43 1995"

1.172 ctime_d

ctime_d Converts a time-value to an ASCII-Text. german version of "ctime"

Usage #include <time.h> (time_stormamiga.h)

s = ctime_d (t);

char *s; const time_t *t;

Standard not (yet) (own development)

Function "ctime_d" is identical to "asctime_d (localtime (t))", it converts a "time_t"-value into string. Names of months and days are printed in german.

Look at german functions .

This function is also available as Inlinefunctions .

Return This function returns an ASCII-text with the exact length of 26 characters. The format of this text is: "DDD MMM dd hh:mm:ss YYYY"

DDD = day of week, MMM = month, dd = day of month, hh:mm:ss = hour:minute:second and YYYY = year. For example: "Mit Okt 25 12:05:43 1995"

1.173 utime

utime setzen von Zugriffs- und Bearbeitungszeit

Usage #include <utime.h>

r = utime(file, timep);

int r; const char *file; const struct utimebuf *timep;

Standard IEEE Std1003.1-1988 ("POSIX")

Function Es wird die Zugriffs- und Bearbeitungszeit der Datei "file" gesetzt. Wenn "timep" null ist, dann wird die aktuelle Zeit als Zugriffs- und Bearbeitungszeit gesetzt.

Bei Definition von STORMAMIGA_UNIXPATH können für diese Funktionen auch Pfadangaben im Unix-Format verwendet werden.

Return Im Fehlerfall -1, sonst 0.

1.174 utimes

utimes setzen von Zugriffs- und Bearbeitungszeit

Usage #include <sys/time.h>

r = utimes(file, times);

int r; const char *file; const struct timeval *times;

Standard not (yet) (4.2BSD)

Function Es wird die Zugriffs- und Bearbeitungszeit der Datei "file" gesetzt. Wenn "times" null ist, dann wird die aktuelle Zeit als Zugriffs- und Bearbeitungszeit gesetzt.

Bei Definition von STORMAMIGA_UNIXPATH können für diese Funktionen auch Pfadangaben im Unix-Format verwendet werden.

Return Im Fehlerfall -1, sonst 0.

1.175 times

times

Usage #include <sys/times.h>

r = times(tp);

clock_t r; struct tms *tp;

Standard IEEE Std1003.1-1988 ("POSIX")

Function

Return

1.176 isinf

isinf Test for an infinite number.

Usage #include <math.h> (math_stormamiga.h)

r = isinf (x);

int r; double x;

Standard IEEE Standard for Binary Floating-Point Arithmetic, Std 754-1985, ANSI

Function "isinf" (is infinite) tests, if "x" is infinite or not.

Return 1 if "x" is infinite, otherwise 0.

1.177 isnan

isnan Test for an illegal number.

Usage #include <math.h> (math_stormamiga.h)

r = isnan (x);

int r; double x;

Standard IEEE Standard for Binary Floating-Point Arithmetic, Std 754-1985, ANSI

Function "isnan" (is not a number) tests, if "x" is an illegal number (division by 0, square root of negative number).

Return 1 if "x" is an illegal number, otherwise 0.

1.178 muls

muls signed 32 Bit mal 32 Bit Multiplikation mit 32 Bit Ergebnis

Usage #include <stormamiga.h>

r = muls (arg1, arg2);

long r; long arg1; long arg2;

Standard not (yet) (own development)

Function Der Befehl "muls" multipliziert "arg1" mit "arg2" und ermittelt das Ergebnis "r". Der Befehl "muls" ist ein sehr schneller und sehr kleiner Ersatz für den Befehl SMult32 der "utility.library". Da die Funktion und der Aufruf beider Befehle identisch ist, kann "SMult32" einfach durch "muls" ersetzt werden.

Der Befehl "muls" ist auch als Inlinefunctions verfügbar.

Return signed 32 Bit Ergebnis "r"

1.179 mulu

mulu unsigned 32 Bit mal 32 Bit Multiplikation mit 32 Bit Ergebnis

Usage #include <stormamiga.h>

r = mulu (arg1, arg2);

ulong r; ulong arg1; ulong arg2;

Standard not (yet) (own development)

Function Der Befehl "mulu" multipliziert "arg1" mit "arg2" und ermittelt das Ergebnis "r". Der Befehl "mulu" ist ein sehr schneller und sehr kleiner Ersatz für den Befehl UMult32 der "utility.library". Da die Funktion und der Aufruf beider Befehle identisch ist, kann "UMult32" einfach durch "mulu" ersetzt werden.

Der Befehl "mulu" ist auch als Inlinefunctions verfügbar.

Return unsigned 32 Bit Ergebnis "r"

1.180 divsl

divsl signed 32 Bit durch 32 Bit Division mit 32 Bit Quotient und Rest

Usage #include <stormamiga.h>

quotient : remainder = divsl (dividend, divisor);

long quotient; long remainder; long dividend; long divisor;

Standard not (yet) (own development)

Function Der Befehl "divsl" teilt den Dividenden "dividend" durch den Divisor "divisor" und ermittelt den Quotient "quotient" und den Rest "remainder". Der Befehl "divsl" ist ein sehr schneller und sehr kleiner Ersatz für den Befehl SDivMod32 der "utility.library". Da die Funktion und der Aufruf beider Befehle identisch ist, kann "SDivMod32" einfach durch "divsl" ersetzt werden.

Return signed 32 Bit Quotient "quotient" signed 32 Bit Rest "remainder"

1.181 divul

divul unsigned 32 Bit durch 32 Bit Division mit 32 Bit Quotient und Rest

Usage #include <stormamiga.h>

quotient : remainder = divul (dividend, divisor);

ulong quotient; ulong remainder; ulong dividend; ulong divisor;

Standard not (yet) (own development)

Function Der Befehl "divul" teilt den Dividenden "dividend" durch den Divisor "divisor" und ermittelt den Quotient "quotient" und den Rest "remainder". Der Befehl "divul" ist ein sehr schneller und sehr kleiner Ersatz für den Befehl UDivMod32 der "utility.library". Da die Funktion und der Aufruf beider Befehle identisch ist, kann "UDivMod32" einfach durch "divul" ersetzt werden.

Return unsigned 32 Bit Quotient "quotient" unsigned 32 Bit Rest "remainder"

1.182 muls64

muls64 signed 32 Bit mal 32 Bit Multiplikation mit 64 Bit Ergebnis

Usage #include <stormamiga.h>

r = muls64 (arg1, arg2);

long r; long arg1; long arg2;

Standard not (yet) (own development)

Function Der Befehl "muls64" multipliziert "arg1" mit "arg2" und ermittelt das Ergebnis "r". Der Befehl "muls64" ist ein sehr schneller und sehr kleiner Ersatz für den Befehl SMult64 der "utility.library". Da die Funktion und der Aufruf beider Befehle identisch ist, kann "SMult64" einfach durch "muls64" ersetzt werden. Im Gegensatz zu "SMult64", benötigt "muls64" nicht AmigaOS 3.x, sondern ist bereits ab AmigaOS 2.x lauffähig.

Return signed 64 Bit Ergebnis "r"

1.183 mulu64

mulu64 unsigned 32 Bit mal 32 Bit Multiplikation mit 64 Bit Ergebnis

Usage #include <stormamiga.h>

r = mulu64 (arg1, arg2);

ulong r; ulong arg1; ulong arg2;

Standard not (yet) (own development)

Function Der Befehl "mulu64" multipliziert "arg1" mit "arg2" und ermittelt das Ergebnis "r". Der Befehl "mulu64" ist ein sehr schneller und sehr kleiner Ersatz für den Befehl UMult64 der "utility.library". Da die Funktion und der Aufruf beider Befehle identisch ist, kann "UMult64" einfach durch "mulu64" ersetzt werden. Im Gegensatz zu "UMult64", benötigt "mulu64" nicht AmigaOS 3.x, sondern ist bereits ab AmigaOS 2.x lauffähig.

Return unsigned 64 Bit Ergebnis "r"

1.184 divs64

divs64 signed 64 Bit durch 32 Bit Division mit 32 Bit Quotient und Rest

Usage #include <stormamiga.h>

quotient : remainder = divs64 (dividend, divisor);

long quotient; long remainder; long dividend; long divisor;

Standard not (yet) (own development)

Function Der Befehl "divs64" teilt den Dividenden "dividend" durch den Divisor "divisor" und ermittelt den Quotient "quotient" und den Rest "remainder".

Return signed 32 Bit Quotient "quotient" signed 32 Bit Rest "remainder"

1.185 divu64

divu64 unsigned 64 Bit durch 32 Bit Division mit 32 Bit Quotient und Rest

Usage #include <stormamiga.h>

quotient : remainder = divu64 (dividend, divisor);

ulong quotient; ulong remainder; ulong dividend; ulong divisor;

Standard not (yet) (own development)

Function Der Befehl "divu64" teilt den Dividenden "dividend" durch den Divisor "divisor" und ermittelt den Quotient "quotient" und den Rest "remainder".

Return unsigned 32 Bit Quotient "quotient" unsigned 32 Bit Rest "remainder"

1.186 button_al

button_al Abfrage der linken Maus- oder Joysticktaste an Port A

Usage #include <stormamiga.h>

r = button_al ();

int r;

Standard not (yet) (own development)

Function Der Befehl "button_al" ist zur Abfrage der linken Maus- oder Joysticktaste an Port A. Wenn zum Zeitpunkt der Abfrage die entsprechende Taste betätigt ist, wird 1 zurückgegeben, sonst 0.

Return 1 wenn die entsprechende Taste betätigt ist, sonst 0

Beispiel #include <stormamiga.h>

int main__(void) { start: if (!button_al ()) goto start; printf_ ("Hello World!\n"); return NULL; }

1.187 button_ar

button_ar Abfrage der rechten Maustaste an Port A

Usage #include <stormamiga.h>

r = button_ar ();

int r;

Standard not (yet) (own development)

Function Der Befehl "button_ar" ist zur Abfrage der rechten Maustaste an Port A. Wenn zum Zeitpunkt der Abfrage die entsprechende Taste betätigt ist, wird 1 zurückgegeben, sonst 0.

Return 1 wenn die entsprechende Taste betätigt ist, sonst 0

Beispiel #include <stormamiga.h>

int main__(void) { start: if (!button_ar ()) goto start; printf_ ("Hello World!\n"); return NULL; }

1.188 button_bl

button_bl Abfrage der linken Maus- oder Joysticktaste an Port B

Usage #include <stormamiga.h>

r = button_bl ();

int r;

Standard not (yet) (own development)

Function Der Befehl "button_bl" ist zur Abfrage der linken Maus- oder Joysticktaste an Port B. Wenn zum Zeitpunkt der Abfrage die entsprechende Taste betätigt ist, wird 1 zurückgegeben, sonst 0.

Return 1 wenn die entsprechende Taste betätigt ist, sonst 0

Beispiel #include <stormamiga.h>

int main__(void) { start: if (!button_bl ()) goto start; printf_ ("Hello World!\n"); return NULL; }

1.189 button_br

button_br Abfrage der rechten Maustaste an Port B

Usage #include <stormamiga.h>

r = button_br ();

int r;

Standard not (yet) (own development)

Function Der Befehl "button_br" ist zur Abfrage der rechten Maustaste an Port B. Wenn zum Zeitpunkt der Abfrage die entsprechende Taste betätigt ist, wird 1 zurückgegeben, sonst 0.

Return 1 wenn die entsprechende Taste betätigt ist, sonst 0

Beispiel #include <stormamiga.h>

int main__(void) { start: if (!button_br ()) goto start; printf_ ("Hello World!\n"); return NULL; }

1.190 button

button Abfrage der Maus- und Joysticktasten

Usage #include <stormamiga.h>

r = button (port, button);

int r; int port; int button;

Standard not (yet) (own development)

Function Der Befehl "button" ist zur Abfrage der Maus- und Joysticktasten. Für "port" kann 0, für Port A, oder 1, für Port B, angegeben werden. Für "button" kann 0, für die linke Maustaste oder die Feuertaste am Joystick, oder 1, für die rechte Maustaste, angegeben werden. Wenn zum Zeitpunkt der Abfrage die entsprechende Taste betätigt ist, wird 1 zurückgegeben, sonst 0.

Return 1 wenn die entsprechende Taste betätigt ist, sonst 0

Beispiel #include <stormamiga.h>

int main__(void) { int p = 0; // Port A int b = 0; // linke Maustaste oder Feuertaste

start: if (!button (p, b)) goto start; printf_ ("Hello World!\n"); return NULL; }

1.191 waitbutton_al

waitbutton_al Abfrage der linken Maus- oder Joysticktaste an Port A

Usage #include <stormamiga.h>

r = waitbutton_al ();

void r;

Standard not (yet) (own development)

Function Der Befehl "waitbutton_al" ist zur Abfrage der linken Maus- oder Joysticktaste an Port A. Das Programm wartet solange, bis die richtige Taste betätigt wird.

Return keine

Beispiel #include <stormamiga.h>

int main__(void) { waitbutton_al (); printf_ ("Hello World!\n"); return NULL; }

1.192 waitbutton_ar

waitbutton_ar Abfrage der rechten Maustaste an Port A

Usage #include <stormamiga.h>

r = waitbutton_ar ();

void r;

Standard not (yet) (own development)

Function Der Befehl "waitbutton_ar" ist zur Abfrage der rechten Maustaste an Port A. Das Programm wartet solange, bis die richtige Taste betätigt wird.

Return keine

Beispiel #include <stormamiga.h>

int main__(void) { waitbutton_ar (); printf_ ("Hello World!\n"); return NULL; }

1.193 waitbutton_bl

waitbutton_bl Abfrage der linken Maus- oder Joysticktaste an Port B

Usage #include <stormamiga.h>

r = waitbutton_bl ();

void r;

Standard not (yet) (own development)

Function Der Befehl "waitbutton_bl" ist zur Abfrage der linken Maus- oder Joysticktaste an Port B. Das Programm wartet solange, bis die richtige Taste betätigt wird.

Return keine

Beispiel #include <stormamiga.h>

int main__(void) { waitbutton_bl (); printf_ ("Hello World!\n"); return NULL; }

1.194 waitbutton_br

waitbutton_br Abfrage der rechten Maustaste an Port B

Usage #include <stormamiga.h>

r = waitbutton_br ();

void r;

Standard not (yet) (own development)

Function Der Befehl "waitbutton_br" ist zur Abfrage der rechten Maustaste an Port B. Das Programm wartet solange, bis die richtige Taste betätigt wird.

Return keine

Beispiel #include <stormamiga.h>

int main__(void) { waitbutton_br (); printf_ ("Hello World!\n"); return NULL; }

1.195 waitbutton

waitbutton Abfrage der Maus- und Joysticktasten

Usage #include <stormamiga.h>

r = waitbutton (port, button);

void r; int port; int button;

Standard not (yet) (own development)

Function Der Befehl "waitbutton" ist zur Abfrage der Maus- und Joysticktasten. Für "port" kann 0, für Port A, oder 1, für Port B, angegeben werden. Für "button" kann 0, für die linke Maustaste oder die Feuertaste am Joystick, oder 1, für die rechte Maustaste, angegeben werden. Das Programm wartet solange, bis die richtige Taste betätigt wird.

Return keine

Beispiel #include <stormamiga.h>

int main__(void) { int p = 0; // Port A int b = 0; // linke Maustaste oder Feuertaste

waitbutton (p, b); printf_ ("Hello World!\n"); return NULL; }

1.196 max_height

max_Height Ermitteln der sichtbaren Fensterhöhe

Usage #include <stormamiga.h>

r = max_Height (window);

int r; struct Window *window;

Standard not (yet) (own development)

Function Der Befehl "max_Height" ermittelt die sichtbare Höhe des Fensters "window".

Der Befehl "max_Height" ist auch als Inlinefunctions verfügbar.

Return Die sichtbare Höhe des Fensters "window".

1.197 max_width

max_Width Ermitteln der sichtbaren Fensterbreite

Usage #include <stormamiga.h>

r = max_Width (window);

int r; struct Window *window;

Standard not (yet) (own development)

Function Der Befehl "max_Width" ermittelt die sichtbare Breite des Fensters "window".

Der Befehl "max_Width" ist auch als Inlinefunctions verfügbar.

Return Die sichtbare Breite des Fensters "window".

1.198 stringbuffer

The stringbuffer "s". ~~~~~~~~~~~~~~~~~~~~~~

The stringbuffer "s" has to be large enough for the outputstring and a terminating zero. The function is not able to determin wether the buffer is large enough.

1.199 parameterlist

The parameterlist "vl". ~~~~~~~~~~~~~~~~~~~~~~~~

The parameterlist "vl" has to be initialized with va_start before its first call and be terminated after the call with va_end.

1.200 outputformatstring

The outputformatstring "format". ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

The formatstring "format" consists of formatspecifiers and outputsymbols. The formatspecifiers describe the type of the output-function's parameters, the type of conversion and the output. All characters that are not beginning with "%" are not formatspecifiers and stay untouched.

The structure of formatspecifiers:

% [flags] [width [.limit]] [size] type

Values in square brackets are optional.

flags "-" leftjustified "+" positive sign for numbers "0" leading zeros for numbers "#" prints a "0x" for hexadecimal numbers and "0" for octal numbers it adds trailing zeros for the types "g" and "G"

width Fieldwidth as decimal numbers or "*", if the width should be the next argument of type int. The fieldwidth is always a minimum value, longer outputs are not adapted.

limit The precision of decimal numbers or "*", in this case the precision is the next argument. It is of type int. Its value is the maximal number of characters printed in case of strings or the minimal number of numbers for fixpoint output or the number of numbers behind the decimal point for floating point values.

size Length of the argument: "h" argument is of type short int or unsigned short int or float "l" argument is of type long int or unsigned long int or double "L" argument is of type long long int or unsigned long long int (Only available in the special 64bit versions.)

type Type of the argument:

"d", "i" Output a signed decimal number, argument is of type int

"o" Output an unsigned octal number, argument is of type int or unsigned int

"x" Output an unsigned hex number in noncapitalized letters, argument is of type int or unsigned int

"X" Output an unsigned hex number in capitalized letters, argument is of type int or unsigned int

"u" Output an unsigned decimal number, argument is of type unsigned int

"c" Output a single character, argument is of type int or is converted to unsigned char

"s" Output a string, terminated with a NULL, argument is of type char *

"f" Output a floating point number in nonexponetial form, argument is of type double

"e" Output a floating point number in exponetial form (with "e" as exp sign), argument is of type double

"E" Output a floating point number in exponetial form (with "E" as exp sign), argument is of type double

"g" Output a floating point number depending on its exponent in nonexponential or in exponential form (with "e" as exp sign), argument is of type double

"G" Output a floating point number depending on its exponent in nonexponential or in exponential form (with "E" as exp sign), argument is of type double

"p" Output a hex memory address, argument is of type void *

"n" Saves the number of charcters printed by this function call into the variable pointed to by the argument of type int *, no output happens

"%" Output a "%" sign

All other types lead to undefined outputs.

1.201 outputformatstring_

The outputformatstring_ "format". ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

The formatstring "format" consists of formatspecifiers and outputsymbols. The formatspecifiers describe the type of the output-function's parameters, the type of conversion and the output. All characters that are not beginning with "%" are not formatspecifiers and stay untouched.

The structure of formatspecifiers:

% [flags] [width [.limit]] [size] type

Values in square brackets are optional.

flags "-" leftjustified "+" positive sign for numbers "0" leading zeros for numbers "#" prints a "0x" for hexadecimal numbers and "0" for octal numbers it adds trailing zeros for the types "g" and "G"

width Fieldwidth as decimal numbers or "*", if the width should be the next argument of type int. The fieldwidth is always a minimum value, longer outputs are not adapted.

limit The precision of decimal numbers or "*", in this case the precision is the next argument. It is of type int. Its value is the maximal number of characters printed in case of strings or the minimal number of numbers for fixpoint output or the number of numbers behind the decimal point for floating point values.

size Length of the argument: "h" argument is of type short int or unsigned short int "l" argument is of type long int or unsigned long int "L" argument is of type long long int or unsigned long long int (Only available in the special 64bit versions.)

type Type of the argument:

"d", "i" Output a signed decimal number, argument is of type int

"o" Output an unsigned octal number, argument is of type int or unsigned int

"x" Output an unsigned hex number in noncapitalized letters, argument is of type int or unsigned int

"X" Output an unsigned hex number in capitalized letters, argument is of type int or unsigned int

"u" Output an unsigned decimal number, argument is of type unsigned int

"c" Output a single character, argument is of type int or is converted to unsigned char

"s" Output a string, terminated with a NULL, argument is of type char *

"p" Output a hex memory address, argument is of type void *

"n" Saves the number of charcters printed by this function call into the variable pointed to by the argument of type int *, no output happens

"%" Output a "%" sign

All other types lead to undefined outputs.

1.202 inputformatstring

The inputputformatstring "format". ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

The formatstring "format" consists of formatspecifiers and inputcharacters. The formatspecifiers describe the type of the input-function's parameters, the type of conversion and the input. All characters that are not beginning with "%" and that are not delimiter (blanks, tabs, linefeeds) are not formatspecifiers and stay untouched.

The structure of formatspecifiers:

% [width] [size] type

Values in square brackets are optional.

width Numbers of bytes to read as decimal number or "*". The later reads the characters, but does not insert them into the next argument.

size Length of the argument: "h" argument is of type short int or unsigned short int or float "l" argument is of type long int or unsigned long int or double "L" argument is of type long long int or unsigned long long int (Only available in the special 64bit versions.)

type Type of the argument:

"d"{par} Input a signed decimal number, argument is of type int *

"i" Input a signed decimal, octal (starts with '0') or hex number (starts with '0x'), argument is of type int

"o" Input an unsigned octal number, argument is of type int or unsigned int

"x" Input an unsigned hex number with or without a leading '0x', argument is of type int or unsigned int

"c" Input "width" characters, blanks and whitespaces are not ignored and no NULLs are appended, argument is of type char *

"s" Input a string, leading blanks and whitespaces are filtered out, a NULL is appended, argument is of type char *

"e", "f", "g" Input a floating point number, argument is of type float *

"p" Input a hex memory address, argument is of type int *

"n" Saves the number of charcters read by this function call into the variable pointed to by the argument of type int *, no input happens

"[...]" Input a string, only consisting of the characters inclosed in the brackets, a NULL is appended, argument is of type char *

"[^...]" Input a string, consisting not of the characters inclosed in the brackets, a NULL is appended, argument is of type char *

"%" Input a "%" sign

All other types are undefined inputs.

1.203 inputformatstring_

The inputputformatstring_ "format". ~~~~~~

The formatstring "format" consists of formatspecifiers and inputcharacters. The formatspecifiers describe the type of the input-function's parameters, the type of conversion and the input. All characters that are not beginning with "%" and that are not delimiter (blanks, tabs, linefeeds) are not formatspecifiers and stay untouched.

The structure of formatspecifiers:

% [width] [size] type

Values in square brackets are optional.

width Numbers of bytes to read as decimal number or "*". The later reads the characters, but does not insert them into the next argument.

size Length of the argument: "h" argument is of type short int or unsigned short int "l" argument is of type long int or unsigned long int "L" argument is of type long long int or unsigned long long int (Only available in the special 64bit versions.)

type Type of the argument:

"d"{par} Input a signed decimal number, argument is of type int *

"i" Input a signed decimal, octal (starts with '0') or hex number (starts with '0x'), argument is of type int

"o" Input an unsigned octal number, argument is of type int or unsigned int

"x" Input an unsigned hex number with or without a leading '0x', argument is of type int or unsigned int

"c" Input "width" characters, blanks and whitespaces are not ignored and no NULLs are appended, argument is of type char *

"s" Input a string, leading blanks and whitespaces are filtered out, a NULL is appended, argument is of type char *

"p" Input a hex memory address, argument is of type int *

"n" Saves the number of charcters read by this function call into the variable pointed to by the argument of type int *, no input happens

"[...]" Input a string, only consisting of the characters inclosed in the brackets, a NULL is appended, argument is of type char *

"[^...]" Input a string, consisting not of the characters inclosed in the brackets, a NULL is appended, argument is of type char *

"%" Input a "%" sign

All other types are undefined inputs.

1.204 timeformatstring

The timeformatstring to convert different forms of time. ~~~~~~

The timeformatstring "format" for the formated output of date and time consists of formatspecifiers and inputcharacters. The formatspecifiers describe the type of the outputfunction's parameters, the type of conversion and the output. All characters that are not beginning with "%" and that are not delimiter (blanks, tabs, linefeeds) are not formatspecifiers and stay untouched.

The structure of formatspecifiers:

% type

type type of the argument:

"a" abbreviated name of the day of the week (Mon, Tue, ...)

"A" full name of the day of the week

"b", "h" abbreviated name of the month (Jan, Feb, ...)

"B" full name of the month

"c" short form of date and time ("%m/%d/%y %I:%M:%S")

"C" display in format "%a %b %e %I:%M:%S %Y"

"d" number of the day of a month (01 bis 31)

"e" number of the day of a month (1 bis 31)

"H" american hour format (01 bis 12)

"I" eropean hour format (00 bis 23)

"j" number of the day of a year (001 bis 366)

"k" european hour format (0 bis 23)

"l" american hour format (1 bis 12)

"m" number of the month (01 bis 12)

"M" number of minutes (00 bis 59)

"n" new line

"p" half of the day ("AM" oder "PM")

"r" display in format "%H:%M:%S %p"

"R" display in format "%I:%M"

"S" number of seconds (00 bis 60)

"t" a tab

"u" number of the day of the week (1 bis 7); monday is the first day of the week

"U" number of the week of a year (00 bis 53); sunday is the first day of the week

"V" number of the week of a year (01 bis 53); monday is the first day of the week

"w" number of the day of the week (0 bis 6); sunday is the first day of the week

"W" number of the week of a year (00 bis 53); monday is the first day of the week

"x", "D" short format of date ("%m/%d/%y")

"X", "T" short format of time ("%I:%M:%S")

"y" number of year without centuries

"Y" full year including century

"Z" name of timezone

"%" a "%" sign

All other types are undefined inputs.

1.205 hints

Hints: ~~~~~~

Although using "stormamiga.lib" is quite easy, I give you some hints.

General notes General usage notes.

Inlinefunctions Description of Inlinefunctions.

64 bit functions Description of 64 bit functions.

OS3 functions Description of OS3 functions.

german functions Description of german functions.

Amiga - functions Description of Amiga - functions.

1.206 general notes

General notes: ~~~~~~~~~~~~~~

Functional Overview of Libraries

Library	required	Codemodell	Datenmodell			Processor	FAR	NEAR	FAR	NEAR A4	NEAR A6	
stormamiga.lib	MC68EC020+	yes	(yes)	yes	yes	no	
stormamiga_nc.lib	MC68EC020+	(yes)	yes	yes	yes	no	
stormamiga_881.lib	MC68881+	yes	(yes)	yes	yes	no	
stormamiga_nc_881.lib	MC68881+	(yes)	yes	yes	yes	no	
stormamiga_040.lib	MC68040+	yes	(yes)	yes	yes	no	
stormamiga_nc_040.lib	MC68040+	(yes)	yes	yes	yes	no	
stormamiga_060.lib	MC68060	yes	(yes)	yes	yes	no	
stormamiga_nc_060.lib	MC68060	(yes)	yes	yes	yes	no	
stormamiga-library.lib	MC68EC020+	yes	(yes)	yes	no	no	
stormamiga-library_nc.lib	MC68EC020+	(yes)	yes	yes	no	no	

(yes) = Libraries marked this way will work with this Codemodell, but your programms become bigger.

The include file "stormamiga.h" contains the declarations for all special functions. In order to avoid long terms like "unsigned long long int", I'v defined short terms like "ullint" for this purpose.

definet shortening

cvoid for const void cchar for const char cschar for const signed char cuchar for const unsigned char schar for signed char uchar for unsigned char ushort for unsigned short lint for long int llint for long long int uint for unsigned int ulint for unsigned long int ullint for unsigned long long int llong for long long ulong for unsigned long ullong for unsigned long long

definition in the file "stormamiga.h"

STORMAMIGA STORMAMIGA_DEUTSCH STORMAMIGA_INLINE STORMAMIGA_NOWB STORMAMIGA_NO_IO_WB STORMAMIGA_OS3

STORMAMIGA_STACK STORMAMIGA_UNIXPATH STORMAMIGA_64BIT

For questions concerning the statupcode look at Startupcode .

All functions ending with "...64" (e.g.: "printf64") are 64 bit functions .

All functions ending with "..._3" (e.g.: "malloc_3") are OS3 functions .

All functions ending with "..._d" (e.g.: "ctime_d") are german functions .

If you want to write portable sources for different compilers, your programs should look like this example.

Example

#ifdef __STORM__ #define STORMAMIGA_INLINE #define STORMAMIGA_DEUTSCH #include <stormamiga.h> #define printf printf_ #define main main__ #endif

#include <stdio.h> #include <time.h>

int main (void) { struct tm *tp; time_t t;

time (&t); tp = localtime (&t); printf ("Die aktuelle Zeit ist %s", asctime (tp)); return NULL; }

1.207 stormamiga

The definition "STORMAMIGA": ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

If you want to use the special functions of "stormamiga.lib", you have to insert the line "#define STORMAMIGA" before you include any include file or you can enter the define into the compileroptons window (preprocessor).

1.208 stormamiga_unixpath

The definition "STORMAMIGA_UNIXPATH": ~~~~~~

The definition of "STORMAMIGA_UNIXPATH" enables the use of paths in unix style. The line "#define STORMAMIGA_UNIXPATH has to be inserted before you include any include file or you can enter the define into the compileroptons window (preprocessor).

All included functions:

stdio functions remove()

unistd functions access() chdir() rmdir() unlink()

fcntl functions creat() open()

stat functions chmod() lstat() mkdir() stat()

dirent functions opendir()

time functions utime() utimes()

1.209 stormamiga_stack

The definition "STORMAMIGA_STACK": ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

You can set the stacksize your program uses with the definition of "STORMAMIGA_STACK". All you have to do is to insert the line "#define STORMAMIGA_STACK xxxx" (xxxx is the stacksize) before you include the includefiles into your program or define it in the window "compileroptions preprocessor" via menu preferences.

Important

If you use this definition, you have to replace the "xxxx" by the real stacksize or you program might crash.

1.210 stormamiga_nowb

The definition "STORMAMIGA_NOWB": ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Programs linked with "stormamiga.lib" and that define "STORMAMIGA_NOWB" are no longer startable from WorkBench, but they become smaller. If you want to use this behaviour you have to insert the line "#define STORMAMIGA_NOWB" into your source before including the includefiles or define it in the window "compileroptions preprocessor" via menu preferences.

Important

If you use this definition, an existing wbmain function will be ignored. If you start a program compiled with this definition from WorkBench it might crash. This is for "real" CLI programs only.

1.211 stormamiga_no_io_wb

The definition "STORMAMIGA_NO_IO_WB": ~~~~~~

The definition of "STORMAMIGA_NO_IO_WB" disables the redirection of standardinput "stdin" and standardouput "stdout" to a CLI window. If you want to use this behaviour you have to insert the line "#define STORMAMIGA_NO_IO_WB" into your source before including the includefiles or define it in the window "compileroptions preprocessor" via menu preferences.

1.212 os3 functions

The OS3 functions: ~~~~~~~~~~~~~~~~~~~

The OS3 functions are especially optimized for AmigaOS 3.x. They make use of the new functions of AmigaOS 3.x and so programs become smaller. If you want to use the OS3 functions you have to insert the line "#define STORMAMIGA_OS3" into your source before including the includefiles or define it in the window "compileroptions preprocessor" via menu preferences.

Important

Please activate the OS3 functions only through the definition of "STORMAMIGA_OS3" and never on its own. Programs compiled this way do not work with AmigaOS 2.x.

All included functions:

stdlib functions free_3() malloc_3()

internal functions EXIT_4_free_3()

1.213 64 bit functions

The 64 Bit functions: ~~~~~~~~~~~~~~~~~~~~~~

The 64bit functions are expanded versions of printf and scanf. They support the specifier "L" for long long int and unsigned long long int. If you want to use these 64 Bit functions, you will have to "#define STORMAMIGA_64BIT" before including the includefiles. This can be done in the compileroptions window in the preprocessor section. If you want to use only a few of these functions, you will have to add a "64" to the name of the function ("printf" becomes "printf64").

All included functions:

stdio functions fprintf64() fprintf64_() fscanf64() fscanf64_() printf64() printf64_() scanf64() scanf64_() snprintf64() snprintf64_() sprintf64() sprintf64_() sscanf64() sscanf64_() vfprintf64() vfprintf64_() vfscanf64() vfscanf64_() vprintf64() vprintf64_() vscanf64() vscanf64_() vsnprintf64() vsnprintf64_() vsprintf64() vsprintf64_() vsscanf64() vsscanf64_()

1.214 inlinefunctions

The inlinefunctions: ~~~~~~~~~~~~~~~~~~~~

The inlinefunctions are placed at the line of the original function call. Programs often become shorter and faster. But if you do not use any optimization, your program will become possibly larger and slower.

If you want to use inline functions, you will have to "#define STORMAMIGA_INLINE" before including the includefiles. This can be done in the compileroptions window in the preprocessor section.

All included functions:

stdio functions clearerr() feof() ferror() fgetc() fputc() getc() perror() putc() putchar() remove() rename() setbuf() setbuffer() setlinebuf() ungetc()

string functions memchr()

stdlib functions abort() atof() atoi() atol() atoll()

time functions ctime() ctime_d() difftime() localtime()

special functions max_Height() max_Width() muls() mulu()

amiga.lib functions CreateExtIO() CreateStdIO() DeleteExtIO() DeleteStdIO() DeleteTask() NewList() RemTOF() waitbeam()

Amiga functions GetAPen() GetBPen() GetDrMd() GetOutlinePen() Move()

1.215 german functions

The german functions: ~~~~~~~~~~~~~~~~~~~~~

Because neither "ANSI C" nor "C++" supports umlaute and all texts are printed in english, I've written some german functions, that output texts in german and support umlaute. If you want to use these german functions, you will have to "#define STORMAMIGA_DEUTSCH" before including the includefiles. This can be done in the compileroptions window in the preprocessor section. If you want to use only a few of these functions, you will have to add a "_d" to the name of the function ("ctime" becomes "ctime_d").

All included functions:

string functions strcasecmp_d() strncasecmp_d() stricmp_d() strnicmp_d() strlower_d() strlwr_d() strupper_d() strupr_d()

ctype functions isalnum_d() isalpha_d() islower_d() isprint_d() ispunct_d() isupper_d() tolower_d() toupper_d()

time functions asctime_d() ctime_d() strftime_d()

1.216 amiga-functions

The Amiga - functions: ~~~~~~~~~~~~~~~~~~~~~~

The Amiga-functions are functions that are already included in the AmigaOS. Because AmigaOS-functions are very often large and slow, I've integrated some of those functions in "stormamiga.lib". The Amiga - functions can be used as inlinefunctions .

1.217 examples

Examples: ~~~~~~~~~

In order to show the advantages and the usage of "stormamiga.lib" in Ansi-C, I included the example-programs "Hello_World", "Pi", "Dhrystone", "SpeedTest" and "TaskDemo".

The examples called "...-storm" are linked with "storm020.lib" and the startupcode "startup.o".

The examples called "...-stormamiga" are linked with "stormamiga_nc.lib" and the startupcode "stormamiga_nc_startups+.o".

The examples called "...-stormamiga-2" are linked with "stormamiga_nc.lib" and the startupcode "stormamiga_nc_startups+.o". Furthermore the sourcecode is optimized for "stormamiga.lib".

In order to show the advantages and the usage of "stormamiga.lib" and "C++.lib" in C++, I included the example-program "Hello_World_C++".

The examples called "...-storm" are linked with "storm020.lib" and the startupcode "startup.o".

The examples called "...-stormamiga" are linked with "stormamiga_nc.lib", "C++.lib" and the startupcode "stormamiga_nc_C++_startups

The examples called "...-stormamiga-2" are linked with "stormamiga_nc.lib", "C++.lib" and the startupcode "stormamiga_nc_C++_startu
Furthermore the sourcecode is optimized for "stormamiga.lib".

Some hints to "SpeedTest":

"SpeedTest" is a simple testprogram that checks the speed and precision of mathematical- and output-functions. The number of testruns should be around 1000000 for MC68060, between 500000 and 1000000 for MC68040+, between 100000 and 500000 for MC68881+ and between 10000 and 50000 without FPU.

On an A1200 with OS 3.0 the "sqrt"-function of "mathieeedoubbas.library" returns an infinite value (inf) instead of an reasonable value (Nan).

1.218 known bugs

Known bugs: ~~~~~~~~~~~

- N O N E

1.219 updates

Updates: ~~~~~~~~

The newest version can be found on the homepage "http://WWW.CyberdyneSystems.de/".

Updates can be obtained directly from the author . If you would like to get the newest version via snail mail just send me a disk (HD or DD) and an adressed envelope including postage or 5,- DM (German Marks) or 3,- $ (US Dollar) or 2,- £ (UK Pound).

1.220 restrictions

Restrictions of the Demo-Version: ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

The demo-version of "stormamiga.lib" lacks the support for the small code modell and optimized versions for MC68881+, MC68040+ und MC68060. Das Erstellen von Shared Librarys ist ebenfalls nicht möglich. Furthermore the demo version shows an information requester whenever you start a program linked with it.

All included files in the demo-version:

Linkerlibraries stormamiga.lib

A special demo version of "stormamiga.lib".

Startupcodes stormamiga_startups.o stormamiga_C++_startups.o

These Startupcodes are special demo versions.

Includefiles sys/dir.h sys/dirent.h sys/fcntl.h sys/resource.h sys/stat.h sys/time.h sys/times.h sys/types.h sys/unistd.h assert.h assert_stormamiga.h ctype.h ctype_stormamiga.h dirent.h fcntl.h limits.h limits_stormamiga.h math.h math_stormamiga.h stdio.h stdio_stormamiga.h stdlib.h stdlib_stormamiga.h stormamiga.h string.h strings.h string_stormamiga.h time.h time_stormamiga.h unistd.h utime.h

Userdictionary User x.dic ("x" is a number between 1 and 3)

Examples Hello_World Hello_World_C++ Dhrystone TaskDemo drops SpeedTest

These examples are adapted to the demo version.

All included files of the registered version:

Linkerlibraries stormamiga.lib stormamiga_nc.lib stormamiga_881.lib stormamiga_nc_881.lib stormamiga_040.lib stormamiga_nc_040 stormamiga_060.lib stormamiga_nc_060.lib stormamiga-library.lib stormamiga-library_nc.lib

Startupcodes stormamiga_startups.o stormamiga_startups+.o stormamiga_nc_startups.o stormamiga_nc_startups+.o stormamiga_C++_s stormamiga_C++_startups+.o stormamiga_nc_C++_startups.o stormamiga_nc_C++_startups+.o

Includefiles sys/dir.h sys/dirent.h sys/fcntl.h sys/resource.h sys/stat.h sys/time.h sys/times.h sys/types.h sys/unistd.h assert.h assert_stormamiga.h ctype.h ctype_stormamiga.h dirent.h fcntl.h limits.h limits_stormamiga.h math.h math_stormamiga.h stdio.h stdio_stormamiga.h stdlib.h stdlib_stormamiga.h stormamiga.h string.h strings.h string_stormamiga.h time.h time_stormamiga.h unistd.h utime.h

Userdictionary User x.dic ("x" is a number between 1 and 3)

Examples Hello_World Hello_World_C++ Dhrystone TaskDemo drops SpeedTest

1.221 registration

Registration: ~~~~~~~~~~~~~

The "stormamiga.lib" is Shareware. If you want to use the unrestricted version, you have to register first and pay the fee. Plaese use the registerform on our homepage "http://WWW.CyberdyneSystems.de/" for registration. If you would like to get the full version via snail mail just send me a disk (HD or DD) and an addressed envelope including postage or 5,- DM (German Marks) or 3,- $ (US Dollar) or 2,- £ (UK Pound) additional. You can find my address at author . Contact me for my Bankaccount. When I have the money then I'll send you your registration number and login to download the full version and all updates from our Homepage.

price version for AmigaOS/68k: 10,- DM (German Marks) or 7,- $ (US Dollar) or 5,- £ (UK Pound)

1.222 copyright

Copyright: ~~~~~~~~~~

The demo-version of "stormamiga.lib" is freely distributable as long as it is NOT CHANGED and ALL files are included UNCHANGED. The registered version of "stormamiga.lib" is ONLY for registered users. The "stormamiga.lib", the password and the registrationnumber must not be published or copied.

Re- or disassembling of "stormamiga.lib" is NOT allowed.

MOST IMPORTANT:

"stormamiga.lib" is PROVIDED AS IS. Use it your OWN RISK.

The author is NOT RESPONSIBLE for any data loss or damage caused directly or indirectly by the use of "stormamiga.lib".

All rights reserved. Please send bugreports and suggestions.

1.223 history

History: ~~~~~~~~

stormamiga.lib V.45.00 (14.08.1998 - 05.01.2000):

stormamiga-library.lib V.45.00 (03.08.1998 - 05.01.2000):

1.224 The

The future: ~~~~~~~~~~~

The following topics are on my wishlist for the next versions of "stormamiga.lib".

- writing all missing functions of "stormamiga.lib" for "PowerPC"

- writing many functions of "UNIX", "POSIX" and other compilers like "DICE" and "SAS C"

- some Amiga functions and new functions

- better support for AmigaOS 3.x

- Your suggestions

1.225 thanks

Thanks: ~~~~~~~

I want to thank the following persons and companies: ––

- by all registered user

- Haage & Partner Computer GmbH; for their support; for the sourcecodes, they gave me for free; especially Jochen Becher and Jürgen Haage, who spend several hours for my probelms

- Uwe Schienbein; for Betatesting, Bugreports, new ideas, his support developing the first generation of the functions "cos" and "sin" for MC68040+ and MC68060, for the exampleprogram "TaskDemo" and the logo for the "stormamiga.lib"

- Thomas Blätte; for the english translation of the installerscripts, the icons for installer, Betatesting, Bugreports and new ideas

- ALeX Kazik; for Bugreports and new ideas

- Kai Fleischer; for the english translation of this guide, Betatesting, Bugreports, new ideas and moral support

- Allan Odgaard; for Betatesting and very much good Bugreports

- Christian Hattemer; for Betatesting, very much good Bugreports and new ideas

- Andreas Mayer-Gürr; for Bugreports and moral support

- Jens Rosenboom; for Bugreports and many, many suggestions

- Onur Pekdimir; for his support publishing "stormamiga.lib"

- Dietmar Heidrich; for "OMA"

- Frank Wille; for "PhxAss"

- by all user which I have forget, sorry @endnode /// /// Author

1.226 author

Author: ~~~~~~~

Matthias Henze Gorkistrasse 127 04347 Leipzig Germany

fon: +49 (0) 341/2326414

email: Matthias_Henze@CyberdyneSystems.de

URL: http://WWW.CyberdyneSystems.de/

English guide: ~~~~~~~~~~~~~~ For bugreports or suggestions concerning the english guide please email:

kai.fleischer@gmd.de

I am always looking for bugreports und suggestions. Please tell me your opinion about "stormamiga.lib".

To all users who work for some kind words

I am searching for users who want to translate the documentation and installerscript into a not (yet) supported language and for betatesters. If you are interested in one of the mentioned topics please mail or call.

Thanks for your support.

1.227 Index

Index: ~~~~~~

A

B

C

D

E

F

G

H

I

K

L

M

O

P

R