stormamiga_lib

Matthias Henze

COLLABORATORS			
	TITLE : stormamiga_lib		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Matthias Henze	August 23, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

stormamiga_lib

1.1 Inhalt

```
                    stormamiga.lib Version 41.035 ( 17.08.1996 )



  geschrieben von Matthias Henze

   F R E E W A R E
```

1.2 Einleitung

Einleitung:
~~~~~~~~~~~

Da die Funktionen der "storm.lib" in C geschrieben sind,
werden die damit gelinkten Programme sehr groß und langsam.
Die Funtionen der "amiga.lib" sind auch nicht gerade klein
und schnell.
Aus diesem Grund habe ich mich am 18.03.1996 entschlossen
die "stormamiga.lib" zu schreiben.

Die "stormamiga.lib" ist komplett in Assembler geschrieben.
Dadurch werden die damit gelinkten Programme auch sehr klein
und schnell.
Mein Ziel ist es, alle Funtionen der "amiga.lib", die nicht
in den Pragmadateien enthalten sind, und alle Funktionen der
"storm.lib" durch kurze und schnelle Assemblerroutinen zu
ersetzen. Außerdem will ich einige Spezialbefehle von anderen
Compilern (zur Zeit nur vom GCC) und einige Routinen, die das
Programmieren erleichtern, in die "stormamiga.lib" integrieren.

Wichtig
~~~~~~~~~~~

In der jetzigen Version der "stormamiga.lib" sind noch nicht
alle Funktionen der "storm.lib" und der "amiga.lib" enthalten.
Außerdem wird zur Zeit nur das große Code- und Datenmodell
unterstützt.
Bis auf den Startupcode "stormamiga_startups.o" sollte die
"stormamiga.lib" auch mit C++ funktionieren. Allerdings wurde
das noch nicht genügend getestet.

1.3 Registrierung

Registrierung:
~~~~~~~~~~~~~~

Obwohl die "stormamiga.lib" Freeware ist, was auch so bleiben
wird, können Sie sich bei
mir
registrieren lassen. Ich möchte
vor allem wissen, ob sich die Weiterentwicklung überhaupt lohnt.

Sie können mir natürlich auch Geschenke, die Sie für angemessen
halten, zuschicken. Es währe auch sehr schön, wenn Sie mir Ihr
Programm, bei dem Sie die "stormamiga.lib" verwendet haben,
zuschicken würden. Außerdem würde ich gerne Ihre Meinung zur
"stormamiga.lib" erfahren.

Für Fehlerberichte (möglichst mit dem entsprechenden Quelltext
und einer genauen Beschreibung) und Verbesserungsvorschläge bin
ich jederzeit dankbar.
Wenn Sie Fragen zur "stormamiga.lib" haben, können Sie mich gerne
anrufen oder mir schreiben. Ich werde die Antworten zu Ihren
Fragen in der Anleitung der nächsten Version veröffentlichen.
Wenn ich Ihnen die Antworten zu Ihren Fragen persönlich zuschicken
soll, dann müssen Sie mir natürlich einen frankierten Briefumschlag
mitschicken. Wenn ich Ihnen die neueste Version der "stormamiga.lib"
zuschicken soll, dann müssen Sie neben dem frankierten Briefumschlag
auch eine Diskette (HD oder DD) mitschicken.

1.4 Systemanforderungen

Systemanforderungen:
~~~~~~~~~~~~~~~~~~~~

- Ein Amiga
- AmigaOS 2.0 oder höher
- MC68EC020 oder höher
- StormC Version 1.05 oder höher

1.5 Installation

```
Installation:
~~~~~~~~~~~~~

Starten Sie das Installationsprogramm "stormamiga.lib HD-
Install" und führen Sie die Installation nach Ihren Wünschen
und Anforderungen durch.

Fügen Sie die "stormamiga.lib" an erster Stelle in das
Projekt ein. Da in dieser Version der "stormamiga.lib"
noch nicht alle Funktionen enthalten sind, müssen Sie
noch die "storm.lib" in das Projekt einfügen.
Wenn Sie die Spezialfunktionen der "stormamiga.lib"
nutzen wollen, müssen Sie noch die Includedatei "storm-
amiga.h", mit "#include <stormamiga.h>", in Ihren Quelltext
einbinden. Die Includedatei "stormamiga.h" sollten Sie als
letzte einbinden.
Um auch den neuen Startupcode "stormamiga_startups.o" zu
nutzen, müssen Sie, bei den Linker Optionen der Projekt-
einstellungen, die Option "Eigener Startup-Code" einschal-
ten und die Datei "stormamiga_startups.o" auswählen.


Wichtig

Der Startupcode "stormamiga_startups.o" kann nur für Ansi-C
verwendet werden!
```

1.6 Funktionen

```
                Funktionen der "stormamiga.lib"
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Da die normalen Ansi-C Funktionen bereits bei StormC
beschrieben werden, erkläre ich nur die Spezialfunktionen
und -befehle.

Hinweis
Wenn Sie sich die Map-Datei, eines mit der "stormamiga.lib"
gelinkten Programmes ansehen, werden Sie feststellen, daß
einige Funktionen den Vorsatz "intern__" (z.B. intern__form_in)
besitzen. Diese Funktionen werden in der "stormamiga.lib" nur
intern verwendet und können nicht als Befehle benutzt werden.



                 Der Startupcode
                AmigaDOS stdio Funktionen

                 SPRINTF
                 VSPRINTF
                stdio Funktionen

                 printf_
                 printf__
```

```
 fprintf_
 fprintf__

 sprintf_
 sprintf__
 vprintf_
 vprintf__

 vfprintf_
 vfprintf__
 vsprintf_
 vsprintf__

 scanf_
 scanf__
 fscanf_
 fscanf__

 sscanf_
 sscanf__
 vscanf
 vscanf_

 vscanf__
 vfscanf
 vfscanf_
 vfscanf__

 vsscanf
 vsscanf_
 vsscanf__
GCC string Funktionen

 strcoll
 strxfrm
 bcmp
 bcopy

 bzero
 ffs
 index
 rindex

 memccpy
 strsep
 swab
Spezial Funktionen

 muls
 mulu
 divsl
 divul

 muls64
 mulu64
```

1.7 Der Startupcode

```
Der Startupcode "stormamiga_startups.o"
Der Startupcode "stormamiga_startups.o" ist speziell für
Ansi-C gedacht.
Der einzige Unterschied, zum Startupcode "startup.o" des
StormC, ist der Aufruf der Funktionen "main" und "wbmain".

Der Startupcode "stormamiga_startups.o" ruft die Funtion
"_main__", das entspricht der Funtion "main__()" in Ansi-C,
auf. Wenn es diese Funtion in Ihrem Programm, nicht gibt,
wird sie aus der "stormamiga.lib" dazugelinkt. Die Funktion
"_main__" ruft die Funktion "_main", das entspricht der Funtion
"main()" in Ansi-C, auf. Wenn es diese Funktion nicht gibt,
meldet der Linker einen Fehler.

Bei einem Start von der Workbench ruft der Startupcode
"stormamiga_startups.o" die Funtion "_wbmain", das entspricht
der Funtion "wbmain()" in Ansi-C, auf. Wenn es diese Funktion
nicht gibt, wird sie aus der "stormamiga.lib" dazugelinkt.


Die Funktion "main__()"

Wenn Sie für Ihr Programm keine Auswertung von Argumenten
benötigen, können Sie die Funktion "main()"in "main__()"
umbenennen. Dadurch wird Ihr Programm etwas kleiner.

Wichtig

Da der Compiler die Funktion "main__()" nicht als normale
main-Funktion erkennt, setzt er auch nicht den Returncode
auf 0. Deshalb müssen Sie den Returncode, mit "return 0",
selber auf 0 setzen.
Die Funktion "main__()" benötigt den Startupcode "stormamiga
_startups.o" und funktioniert nur in Ansi-C.
```

1.8 SPRINTF

```
                SPRINTF
Formatierte Ausgabe in den Stringpuffer "s".

Übersicht
#include <stormamiga.h>

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

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

Standard
(noch) keiner (Eigenentwicklung)
```

```
Erklärung
Formatierte Ausgabe in den
                 Stringpuffer
                 "s". Der Formatstring
"format" beschreibt das Ausgabeformat. Danach folgen zusätzlich
angegebene Parameter.

"SPRINTF" verwendet den Befehl  RawDoFmt  der "exec.library"
und ist dadurch auch sehr klein.

Rückgabe
Die Anzahl der ausgegebenen Zeichen.
```

1.9 VSPRINTF

```
               VSPRINTF
Formatierte Ausgabe in den Stringpuffer "s".

Übersicht
#include <stormamiga.h>

r = VSPRINTF(s,format,vl);

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

Standard
(noch) keiner (Eigenentwicklung)

Erklärung
Formatierte Ausgabe in den
                 Stringpuffer
                 "s". Der Formatstring
"format" beschreibt das Ausgabeformat. Danach folgt eine

                 Parameterliste
                 "vl".

"VSPRINTF" verwendet den Befehl  RawDoFmt  der "exec.library"
und ist dadurch auch sehr klein.

Rückgabe
Die Anzahl der ausgegebenen Zeichen.
```

1.10 printf_

```
               printf_
Formatierte Ausgabe in die Standardausgabe "stdout".
```

```
Funktionsreduzierte Version von "printf".

Übersicht
#include <stormamiga.h>

r = printf_(format,...);

int r;
const char *format;

Standard
(noch) keiner (Eigenentwicklung)

Erklärung
Formatierte Ausgabe in die Standardausgabe "stdout". Der

                Ausgabeformatstring_
                "format" beschreibt das Ausgabe-
format. Danach folgen zusätzlich angegebene Parameter.

Rückgabe
Die Anzahl der ausgegebenen Zeichen oder im Fehlerfall
eine negative Zahl.
```

1.11 printf__

```
               printf__
Formatierte Ausgabe in die Standardausgabe "stdout".
Funktionsreduzierte Version von "printf_".

Übersicht
#include <stormamiga.h>

r = printf__(format,...);

int r;
const char *format;

Standard
(noch) keiner (Eigenentwicklung)

Erklärung
Formatierte Ausgabe in die Standardausgabe "stdout". Der

                Ausgabeformatstring__
                "format" beschreibt das Ausgabe-
format. Danach folgen zusätzlich angegebene Parameter.

Rückgabe
Die Anzahl der ausgegebenen Zeichen oder im Fehlerfall
eine negative Zahl.
```

1.12 fprintf_

```
                fprintf_
Formatierte Ausgabe in die Datei "f".
Funktionsreduzierte Version von "fprintf".

Übersicht
#include <stormamiga.h>

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

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

Standard
(noch) keiner (Eigenentwicklung)

Erklärung
Formatierte Ausgabe in die Datei "f". Der
                 Ausgabeformatstring_
                "format" beschreibt das Ausgabeformat. Danach folgen zusätzlich
angegebene Parameter.

Rückgabe
Die Anzahl der ausgegebenen Zeichen oder im Fehlerfall
eine negative Zahl.
```

1.13 fprintf__

```
                fprintf__
Formatierte Ausgabe in die Datei "f".
Funktionsreduzierte Version von "fprintf_".

Übersicht
#include <stormamiga.h>

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

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

Standard
(noch) keiner (Eigenentwicklung)

Erklärung
Formatierte Ausgabe in die Datei "f". Der
                 Ausgabeformatstring__
                "format" beschreibt das Ausgabeformat. Danach folgen zusätzlich
angegebene Parameter.

Rückgabe
```

```
Die Anzahl der ausgegebenen Zeichen oder im Fehlerfall
eine negative Zahl.
```

1.14 sprintf_

```
                sprintf_
Formatierte Ausgabe in den Stringpuffer "s".
Funktionsreduzierte Version von "sprintf".

Übersicht
#include <stormamiga.h>

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

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

Standard
(noch) keiner (Eigenentwicklung)

Erklärung
Formatierte Ausgabe in den
                 Stringpuffer
                 "s". Der

                 Ausgabeformatstring_
                 "format" beschreibt das Ausgabe-
format. Danach folgen zusätzlich angegebene Parameter.

Rückgabe
Die Anzahl der ausgegebenen Zeichen oder im Fehlerfall
eine negative Zahl.
```

1.15 sprintf__

```
                sprintf__
Formatierte Ausgabe in den Stringpuffer "s".
Funktionsreduzierte Version von "sprintf_".

Übersicht
#include <stormamiga.h>

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

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

Standard
```

```
(noch) keiner (Eigenentwicklung)

Erklärung
Formatierte Ausgabe in den
                 Stringpuffer
                 "s". Der

                 Ausgabeformatstring__
                 "format" beschreibt das Ausgabe-
format. Danach folgen zusätzlich angegebene Parameter.

Rückgabe
Die Anzahl der ausgegebenen Zeichen oder im Fehlerfall
eine negative Zahl.
```

1.16 vprintf_

```
               vprintf_
Formatierte Ausgabe in die Standardausgabe "stdout".
Funktionsreduzierte Version von "vprintf".

Übersicht
#include <stormamiga.h>

r = vprintf_(format,vl);

int r;
const char *format;
va_list vl;

Standard
(noch) keiner (Eigenentwicklung)

Erklärung
Formatierte Ausgabe in die Standardausgabe "stdout". Der

                 Ausgabeformatstring_
                 "format" beschreibt das Ausgabe-
format. Danach folgt eine
                 Parameterliste
                 "vl".

Rückgabe
Die Anzahl der ausgegebenen Zeichen oder im Fehlerfall
eine negative Zahl.
```

1.17 vprintf__

```
               vprintf__
Formatierte Ausgabe in die Standardausgabe "stdout".
```

```
Funktionsreduzierte Version von "vprintf_".

Übersicht
#include <stormamiga.h>

r = vprintf__(format,vl);

int r;
const char *format;
va_list vl;

Standard
(noch) keiner (Eigenentwicklung)

Erklärung
Formatierte Ausgabe in die Standardausgabe "stdout". Der

                Ausgabeformatstring__
                "format" beschreibt das Ausgabe-
format. Danach folgt eine
                Parameterliste
                "vl".

Rückgabe
Die Anzahl der ausgegebenen Zeichen oder im Fehlerfall
eine negative Zahl.
```

1.18 vfprintf_

```
               vfprintf_
Formatierte Ausgabe in die Datei "f".
Funktionsreduzierte Version von "vfprintf".

Übersicht
#include <stormamiga.h>

r = vfprintf_(f,format,vl);

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

Standard
(noch) keiner (Eigenentwicklung)

Erklärung
Formatierte Ausgabe in die Datei "f". Der

                Ausgabeformatstring_
                "format" beschreibt
das Ausgabeformat. Danach folgt eine

                Parameterliste
```

```
                "vl".

Rückgabe
Die Anzahl der ausgegebenen Zeichen oder im Fehlerfall
eine negative Zahl.
```

1.19 vfprintf__

```
               vfprintf__
Formatierte Ausgabe in die Datei "f".
Funktionsreduzierte Version von "vfprintf_".

Übersicht
#include <stormamiga.h>

r = vfprintf__(f,format,vl);

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

Standard
(noch) keiner (Eigenentwicklung)

Erklärung
Formatierte Ausgabe in die Datei "f". Der

                Ausgabeformatstring__
                "format" beschreibt
das Ausgabeformat. Danach folgt eine

                Parameterliste
                "vl".

Rückgabe
Die Anzahl der ausgegebenen Zeichen oder im Fehlerfall
eine negative Zahl.
```

1.20 vsprintf_

```
               vsprintf_
Formatierte Ausgabe in den Stringpuffer "s".
Funktionsreduzierte Version von "vsprintf".

Übersicht
#include <stormamiga.h>

r = vsprintf_(s,format,vl);
```

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

Standard
(noch) keiner (Eigenentwicklung)

Erklärung
Formatierte Ausgabe in den
                Stringpuffer
                "s". Der

                Ausgabeformatstring_
                "format" beschreibt das
Ausgabeformat. Danach folgt eine
                Parameterliste
                "vl".

Rückgabe
Die Anzahl der ausgegebenen Zeichen oder im Fehlerfall
eine negative Zahl.
```

1.21 vsprintf__

```
               vsprintf__
Formatierte Ausgabe in den Stringpuffer "s".
Funktionsreduzierte Version von "vsprintf_".

Übersicht
#include <stormamiga.h>

r = vsprintf__(s,format,vl);

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

Standard
(noch) keiner (Eigenentwicklung)

Erklärung
Formatierte Ausgabe in den
                Stringpuffer
                "s". Der

                Ausgabeformatstring__
                "format" beschreibt das
Ausgabeformat. Danach folgt eine
                Parameterliste
                "vl".

Rückgabe
```

```
Die Anzahl der ausgegebenen Zeichen oder im Fehlerfall
eine negative Zahl.
```

1.22 scanf_

```
                scanf_
Einlesen einer formatierten Eingabe aus der Standardeingabe "stdin".
Funktionsreduzierte Version von "scanf".

Übersicht
#include <stormamiga.h>

r = scanf_(format,...);

int r;
const char *format;

Standard
(noch) keiner (Eigenentwicklung)

Erklärung
Einlesen einer formatierten Eingabe aus der Standardeingabe "stdin".
Der
                Eingabeformatstring_
                "format" beschreibt das Eingabeformat.
Danach folgen zusätzlich angegebene Parameter.

Rückgabe
Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rück-
gabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.
```

1.23 scanf__

```
                scanf__
Einlesen einer formatierten Eingabe aus der Standardeingabe "stdin".
Funktionsreduzierte Version von "scanf_".

Übersicht
#include <stormamiga.h>

r = scanf__(format,...);

int r;
const char *format;

Standard
(noch) keiner (Eigenentwicklung)

Erklärung
Einlesen einer formatierten Eingabe aus der Standardeingabe "stdin".
```

```
Der
                 Eingabeformatstring__
                 "format" beschreibt das Eingabeformat.
Danach folgen zusätzlich angegebene Parameter.

Rückgabe
Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rück-
gabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.
```

1.24 fscanf_

```
                fscanf_
Einlesen einer formatierten Eingabe aus der Datei "f".
Funktionsreduzierte Version von "fscanf".

Übersicht
#include <stormamiga.h>

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

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

Standard
(noch) keiner (Eigenentwicklung)

Erklärung
Einlesen einer formatierten Eingabe aus der Datei "f". Der

                 Eingabeformatstring_
                 "format" beschreibt das Eingabeformat.
Danach folgen zusätzlich angegebene Parameter.

Rückgabe
Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rück-
gabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.
```

1.25 fscanf__

```
                fscanf__
Einlesen einer formatierten Eingabe aus der Datei "f".
Funktionsreduzierte Version von "fscanf_".

Übersicht
#include <stormamiga.h>

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

int r;
```

```
FILE *f;
const char *format;

Standard
(noch) keiner (Eigenentwicklung)

Erklärung
Einlesen einer formatierten Eingabe aus der Datei "f". Der

                Eingabeformatstring__
                "format" beschreibt das Eingabeformat.
Danach folgen zusätzlich angegebene Parameter.

Rückgabe
Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rück-
gabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.
```

1.26 sscanf_

```
               sscanf_
Einlesen einer formatierten Eingabe aus dem Stringpuffer "s".
Funktionsreduzierte Version von "sscanf".

Übersicht
#include <stormamiga.h>

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

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

Standard
(noch) keiner (Eigenentwicklung)

Erklärung
Einlesen einer formatierten Eingabe aus dem Stringpuffer "s". Der

                Eingabeformatstring_
                "format" beschreibt das Eingabeformat.
Danach folgen zusätzlich angegebene Parameter.

Rückgabe
Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rück-
gabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.
```

1.27 sscanf__

```
               sscanf__
Einlesen einer formatierten Eingabe aus dem Stringpuffer "s".
```

```
Funktionsreduzierte Version von "sscanf_".

Übersicht
#include <stormamiga.h>

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

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

Standard
(noch) keiner (Eigenentwicklung)

Erklärung
Einlesen einer formatierten Eingabe aus dem Stringpuffer "s". Der

                Eingabeformatstring__
                "format" beschreibt das Eingabeformat.
Danach folgen zusätzlich angegebene Parameter.

Rückgabe
Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rück-
gabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.
```

1.28 vscanf

```
               vscanf
Einlesen einer formatierten Eingabe aus der Standardeingabe "stdin".

Übersicht
#include <stormamiga.h>

r = vscanf(format,vl);

int r;
const char *format;
va_list vl;

Standard
(noch) keiner (Eigenentwicklung)

Erklärung
Einlesen einer formatierten Eingabe aus der Standardeingabe "stdin".
Der
                Eingabeformatstring
                "format" beschreibt das Eingabeformat.
Danach folgt eine
                Parameterliste
                "vl".

Rückgabe
Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rück-
gabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.
```

1.29 vscanf_

```
                vscanf_
Einlesen einer formatierten Eingabe aus der Standardeingabe "stdin".
Funktionsreduzierte Version von "scanf".

Übersicht
#include <stormamiga.h>

r = vscanf_(format,vl);

int r;
const char *format;
va_list vl;

Standard
(noch) keiner (Eigenentwicklung)

Erklärung
Einlesen einer formatierten Eingabe aus der Standardeingabe "stdin".
Der
                 Eingabeformatstring_
                 "format" beschreibt das Eingabeformat.
Danach folgt eine
                 Parameterliste
                 "vl".

Rückgabe
Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rück-
gabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.
```

1.30 vscanf__

```
                vscanf__
Einlesen einer formatierten Eingabe aus der Standardeingabe "stdin".
Funktionsreduzierte Version von "scanf_".

Übersicht
#include <stormamiga.h>

r = vscanf__(format,vl);

int r;
const char *format;
va_list vl;

Standard
(noch) keiner (Eigenentwicklung)
```

```
Erklärung
Einlesen einer formatierten Eingabe aus der Standardeingabe "stdin".
Der
                 Eingabeformatstring__
                 "format" beschreibt das Eingabeformat.
Danach folgt eine
                 Parameterliste
                 "vl".

Rückgabe
Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rück-
gabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.
```

1.31 vfscanf

```
                vfscanf
Einlesen einer formatierten Eingabe aus der Datei "f".

Übersicht
#include <stormamiga.h>

r = vfscanf(f,format,vl);

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

Standard
(noch) keiner (Eigenentwicklung)

Erklärung
Einlesen einer formatierten Eingabe aus der Datei "f". Der

                 Eingabeformatstring
                 "format" beschreibt das Eingabeformat.
Danach folgt eine
                 Parameterliste
                 "vl".

Rückgabe
Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rück-
gabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.
```

1.32 vfscanf_

```
                vfscanf_
Einlesen einer formatierten Eingabe aus der Datei "f".
Funktionsreduzierte Version von "vfscanf".
```

```
Übersicht
#include <stormamiga.h>

r = vfscanf_(f,format,vl);

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

Standard
(noch) keiner (Eigenentwicklung)

Erklärung
Einlesen einer formatierten Eingabe aus der Datei "f". Der

                Eingabeformatstring_
                "format" beschreibt das Eingabeformat.
Danach folgt eine
                Parameterliste
                "vl".

Rückgabe
Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rück-
gabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.
```

1.33 vfscanf__

```
               vfscanf__
Einlesen einer formatierten Eingabe aus der Datei "f".
Funktionsreduzierte Version von "vfscanf_".

Übersicht
#include <stormamiga.h>

r = vfscanf__(f,format,vl);

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

Standard
(noch) keiner (Eigenentwicklung)

Erklärung
Einlesen einer formatierten Eingabe aus der Datei "f". Der

                Eingabeformatstring__
                "format" beschreibt das Eingabeformat.
Danach folgt eine
                Parameterliste
                "vl".
```

```
Rückgabe
Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rück-
gabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.
```

1.34 vsscanf

```
                vsscanf
Einlesen einer formatierten Eingabe aus dem Stringpuffer "s".

Übersicht
#include <stormamiga.h>

r = vsscanf(s,format,vl);

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

Standard
(noch) keiner (Eigenentwicklung)

Erklärung
Einlesen einer formatierten Eingabe aus dem Stringpuffer "s". Der

                 Eingabeformatstring
                 "format" beschreibt das Eingabeformat.
Danach folgt eine
                 Parameterliste
                 "vl".

Rückgabe
Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rück-
gabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.
```

1.35 vsscanf_

```
                vsscanf_
Einlesen einer formatierten Eingabe aus dem Stringpuffer "s".
Funktionsreduzierte Version von "vsscanf".

Übersicht
#include <stormamiga.h>

r = vsscanf_(s,format,vl);

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

```
Standard
(noch) keiner (Eigenentwicklung)

Erklärung
Einlesen einer formatierten Eingabe aus dem Stringpuffer "s". Der

                Eingabeformatstring_
                "format" beschreibt das Eingabeformat.
Danach folgt eine
                Parameterliste
                "vl".

Rückgabe
Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rück-
gabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.
```

1.36 vsscanf__

```
               vsscanf__
Einlesen einer formatierten Eingabe aus dem Stringpuffer "s".
Funktionsreduzierte Version von "vsscanf_".

Übersicht
#include <stormamiga.h>

r = vsscanf__(s,format,vl);

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

Standard
(noch) keiner (Eigenentwicklung)

Erklärung
Einlesen einer formatierten Eingabe aus dem Stringpuffer "s". Der

                Eingabeformatstring__
                "format" beschreibt das Eingabeformat.
Danach folgt eine
                Parameterliste
                "vl".

Rückgabe
Die Anzahl der eingelesenen Zeichen oder im Fehlerfall einen Rück-
gabewert der kleiner als die Anzahl der eingelesenen Zeichen ist.
```

1.37 strcoll

```
strcoll
vergleichen zweier Strings unter Beachtung der aktuellen Sprache

Übersicht
#include <stormamiga.h>

r = strcoll(s1,s2);

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

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

Erklärung
Vergleicht die Strings "s1" und "s2" Zeichen für Zeichen. Wenn die
"locale.library" geöffnet ist, wird der Befehl "StrnCmp" ausgeführt.
Wenn sie nicht geöffnet ist, wird der Befehl "strcmp" ausgeführt.
Dabei wird die aktuelle Sprache nicht berücksichtigt.

Rückgabe
<0 wenn s1 < s2
=0 wenn s1 = s2
>0 wenn s1 > s2
```

1.38 strxfrm

```
strxfrm
transformieren eines Strings unter Beachtung der aktuellen Sprache

Übersicht
#include <stormamiga.h>

r = strxfrm(dest,source,n);

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

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

Erklärung
Transformiert (kopiert) maximal "n" Bytes vom String "source" nach
"dest". Wenn die "locale.library" geöffnet ist, wird der Befehl
"StrConvert" ausgeführt. Wenn sie nicht geöffnet ist, wird der
Befehl "strncpy" ausgeführt. Dabei wird der String "source", ohne
Berücksichtigung der aktuellen Sprache, nur kopiert .

Rückgabe
Die Länge des transformierten Strings "source" ohne Nullzeichen.
```

1.39 bcmp

```
bcmp
vergleichen zweier Speicherbereiche

Übersicht
#include <stormamiga.h>

r = bcmp(b1,b2,n);

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

Standard
(noch) keiner (4.2BSD)

Erklärung
Vergleicht die Speicherbereiche "b1" und "b2" Byte für Byte auf
maximal "n" Bytes Länge. Die Speicherbereiche dürfen sich über-
schneiden.

Rückgabe
Wenn beide Byte-Strings gleich sind, ist der Rückgabewert 0, sonst
ist er ungleich 0.
```

1.40 bcopy

```
bcopy
Speicher kopieren

Übersicht
#include <stormamiga.h>

r = bcopy(source,dest,n);

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

Standard
(noch) keiner (4.2BSD)

Erklärung
Kopiert "n" Bytes vom Speicherbereich "source" nach "dest". Die
Speicherbereiche dürfen sich überschneiden. Wenn "n" 0 ist, wird
nichts kopiert.

Rückgabe
```

1.41 bzero

```
bzero
schreibt NULL-Bytes in einen Speicherbereich

Übersicht
#include <stormamiga.h>

r = bzero(b,n);

void r;
void b;
size_t n;

Standard
(noch) keiner (4.3BSD)

Erklärung
Schreibt "n" NULL-Bytes in den Speicherbereich "b".

Rückgabe
```

1.42 ffs

```
ffs
Findet das erste gesetzte Bit in einem Bit-String

Übersicht
#include <stormamiga.h>

r = ffs(value);

int r;
int value;

Standard
(noch) keiner (4.3BSD)

Erklärung
Findet das erste gesetzte Bit in dem Bit-String "value" und gibt
den Index davon zurück.

Rückgabe
Index des Bit-String "value"
```

1.43 index

```
index
Sucht das erste Vorkommen eines Zeichens in einem String

Übersicht
```

```
#include <stormamiga.h>

r = index(s,c);

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

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

Erklärung
Sucht das erste Vorkommen des Zeichens "c" in dem String "s"
und gibt einen Zeiger auf das erste gefundene Zeichen "c"
zurück. Wenn das Zeichen "c" nicht gefunden wird, wird 0
zurückgegeben.

Rückgabe
Ein Zeiger auf das erste gefundene Zeichen "c" oder 0.
```

1.44 rindex

```
rindex
Sucht das letzte Vorkommen eines Zeichens in einem String

Übersicht
#include <stormamiga.h>

r = rindex(s,c);

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

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

Erklärung
Sucht das letzte Vorkommen des Zeichens "c" in dem String "s"
und gibt einen Zeiger auf das letzte gefundene Zeichen "c"
zurück. Wenn das Zeichen "c" nicht gefunden wird, wird 0
zurückgegeben.

Rückgabe
Ein Zeiger auf das letzte gefundene Zeichen "c" oder 0.
```

1.45 memccpy

```
memccpy
Speicher kopieren

Übersicht
```

```
#include <stormamiga.h>

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

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

Standard
(noch) keiner (4.3BSD)

Erklärung
Die Funktion kopiert den Speicherbereich "source" in den Speicher-
bereich "dest". Wenn das Zeichen "c" im Speicherbereich "source"
vorkommt, wird der Kopiervorgang an dieser Stelle gestoppt und ein
Zeiger auf das Byte hinter der Kopie des Zeichens "c" im Speicher-
bereich "dest" zurückgegeben. Ansonsten werden "n" Bytes kopiert
und 0 zurückgegeben.

Rückgabe
Ein Zeiger auf das Byte hinter der Kopie des Zeichens "c" im
Speicherbereich "dest" oder 0.
```

1.46 strsep

```
strsep
trennt Strings

Übersicht
#include <stormamiga.h>

r = strsep(s1,s2);

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

Standard
(noch) keiner

Erklärung
Die Funktion findet im String "*s1" (mit abschließendem Nullzeichen)
das erste Vorkommen eines Zeichens aus dem String "s2", ersetzt dieses
durch ein Nullzeichen und verzeichnet die Stelle des nächsten Zeichens
im String "*s1". Es wird der Originalwert des Strings "*s1" zurückgege-
ben. Wenn kein Zeichens aus dem String "s2" gefunden wird, wird 0 zurück-
gegeben.

Rückgabe
Der Originalwert des Strings "*s1" oder 0.
```

1.47 swab

```
swab
vertauschen angrenzender Bytes

Übersicht
#include <stormamiga.h>

r = swab(source,dest,n);

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

Standard
(noch) keiner (Version 7 AT&T UNIX)

Erklärung
Kopiert "n" Bytes von "source" nach "dest" und vertauscht die
angrenzenden Bytes. "n" muß eine gerade Zahl sein.

Rückgabe
```

1.48 muls

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

Übersicht
#include <stormamiga.h>

r = muls(arg1,arg2);

LONG r;
LONG arg1;
LONG arg2;

Standard
(noch) keiner (Eigenentwicklung)

Erklärung
Der Befehl "muls" multipliziert "arg1" mit "arg2" und ermittelt
das Ergebnis "r".
Der Befehl "muls" ist ein sehr schneller und sehr kleiner (6 Byte)
Ersatz für den Befehl  SMult32  der "utility.library". Da die Funk-
tion und der Aufruf beider Befehle identisch ist, kann "SMult32"
einfach durch "muls" ersetzt werden.

Rückgabe
signed 32 Bit Ergebnis "r"
```

1.49 mulu

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

Übersicht
#include <stormamiga.h>

r = mulu(arg1,arg2);

ULONG r;
ULONG arg1;
ULONG arg2;

Standard
(noch) keiner (Eigenentwicklung)

Erklärung
Der Befehl "mulu" multipliziert "arg1" mit "arg2" und ermittelt
das Ergebnis "r".
Der Befehl "mulu" ist ein sehr schneller und sehr kleiner (6 Byte)
Ersatz für den Befehl  UMult32  der "utility.library". Da die Funk-
tion und der Aufruf beider Befehle identisch ist, kann "UMult32"
einfach durch "mulu" ersetzt werden.

Rückgabe
unsigned 32 Bit Ergebnis "r"
```

1.50 divsl

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

Übersicht
#include <stormamiga.h>

quotient:remainder = divsl(dividend,divisor);

LONG quotient;
LONG remainder
LONG dividend;
LONG divisor;

Standard
(noch) keiner (Eigenentwicklung)

Erklärung
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
(6 Byte) 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.
```

```
Rückgabe
signed 32 Bit Quotient "quotient"
signed 32 Bit Rest "remainder"
```

1.51 divul

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

Übersicht
#include <stormamiga.h>

quotient:remainder = divul(dividend,divisor);

ULONG quotient;
ULONG remainder
ULONG dividend;
ULONG divisor;

Standard
(noch) keiner (Eigenentwicklung)

Erklärung
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
(6 Byte) 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.

Rückgabe
unsigned 32 Bit Quotient "quotient"
unsigned 32 Bit Rest "remainder"
```

1.52 muls64

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

Übersicht
#include <stormamiga.h>

r = muls64(arg1,arg2);

LONG r;
LONG arg1;
LONG arg2;

Standard
(noch) keiner (Eigenentwicklung)
```

```
Erklärung
Der Befehl "muls64" multipliziert "arg1" mit "arg2" und ermittelt
das Ergebnis "r".
Der Befehl "muls64" ist ein sehr schneller und sehr kleiner (6 Byte)
Ersatz für den Befehl  SMult64  der "utility.library". Da die Funk-
tion 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.

Rückgabe
signed 64 Bit Ergebnis "r"
```

1.53 mulu64

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

Übersicht
#include <stormamiga.h>

r = mulu64(arg1,arg2);

ULONG r;
ULONG arg1;
ULONG arg2;

Standard
(noch) keiner (Eigenentwicklung)

Erklärung
Der Befehl "mulu64" multipliziert "arg1" mit "arg2" und ermittelt
das Ergebnis "r".
Der Befehl "mulu64" ist ein sehr schneller und sehr kleiner (6 Byte)
Ersatz für den Befehl  UMult64  der "utility.library". Da die Funk-
tion 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.

Rückgabe
unsigned 64 Bit Ergebnis "r"
```

1.54 Stringpuffer

```
Der Stringpuffer "s"

Der Stringpuffer "s" muß mindestens so groß sein, daß die
Ausgabe mit abschließenden Nullzeichen hineinpaßt.
Die Funktion kann nicht feststellen ob der Stringpuffer groß
genug ist.
```

1.55 Parameterliste

```
Die Parameterliste "vl"

Die Parameterliste "vl" muß vor dem Aufruf mit va_start initia-
lisiert werden und nach dem Aufruf mit va_end abgeschlossen wer-
den.
```

1.56 Ausgabeformatstring_

```
Der Ausgabeformatstring_ "format" zur formatierten Ausgabe
Der Formatstring "format" zur formatierten Ausgabe besteht aus
Formatkommandos und Ausgabezeichen. Die Formatkommandos bestimmen
den Typ der Parameter der Ausgabefunktion und die Art der Konver-
tierung und Ausgabe. Alle Zeichen, die kein Formatkommando sind,
also nicht mit dem Zeichen "%" beginnen sind Ausgabezeichen und
werden unverändert ausgegeben.

Der Aufbau eines Formatkommandos:

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

Die Angaben in den eckigen Klammern können optional angegeben werden.

flags
"-" zur Linksjustierung;
"+" zur Ausgabe auch eines positiven Vorzeichens bei Zahlen;
"0" zur Ausgabe führender Nullen bei Zahlen;
"#" zur Ausgabe von "0x" bei hexadezimalen Zahlen und "0" bei
    oktalen Zahlen

width
Feldbreite als dezimale Ziffernfolge oder "*", in diesem Fall wird
die Feldbreite als nächstes Argument des Typs int übergeben. Die
Feldbreite ist immer ein minimaler Wert, zu lange Ausgaben werden
nicht beschnitten.

limit
Die Genauigkeit als dezimale Ziffernfolge oder "*", in diesem Fall
wird die Genauigkeit als nächstes Argument des Typs int übergeben.
Der Wert beschreibt die maximale Anzahl von Zeichen bei Ausgabe
einer Zeichenkette oder die minimale Anzahl von Ziffern einer ganz-
zahligen Ausgabe.

size
Längenangabe des Arguments:
"h" für ein Argument des Typs short int oder unsigned short int;
"l" für ein Argument des Typs long int oder unsigned long int;

type
Typangabe des Arguments:

"d"
"i" zur Ausgabe einer vorzeichenbehafteten Dezimalzahl, das zuge-
```

```
    hörige Argument ist vom Typ int

"o" zur Ausgabe einer vorzeichenlosen Oktalzahl, das zugehörige
    Argument ist vom Typ int oder unsigned int

"x" zur Ausgabe einer vorzeichenlosen Hexadezimalzahl mit Kleinbuch-
    staben, das zugehörige Argument ist vom Typ int oder unsigned int

"X" zur Ausgabe einer vorzeichenlosen Hexadezimalzahl mit Großbuch-
    staben, das zugehörige Argument ist vom Typ int oder unsigned int

"u" zur Ausgabe einer vorzeichenlosen Dezimalzahl, das zugehörige
    Argument ist vom Typ unsigned int

"c" zur Ausgabe eines Zeichens, das zugehörige Argument ist vom Typ
    int und wird in unsigned char konvertiert

"s" zur Ausgabe einer Zeichenkette, die mit einem Nullzeichen abge-
    schlossen ist, das zugehörige Argument ist vom Typ char *

"p" zur Ausgabe einer hexadezimalen Speicheradresse, das zugehörige
    Argument ist vom Typ void *;

"n" zur Speicherung der Anzahl der bisher von diesem Funktionsaufruf
    ausgegebenen Zeichen in der Variablen, auf die das Argument vom
    Typ int * zeigt, es erfolgt keine Ausgabe

"%" zur Ausgabe eines Prozentzeichens

Alle anderen Typangaben führen zu undefinierten Ausgaben.
```

1.57 Ausgabeformatstring__

```
Der Ausgabeformatstring__ "format" zur formatierten Ausgabe
Der Formatstring "format" zur formatierten Ausgabe besteht aus
Formatkommandos und Ausgabezeichen. Die Formatkommandos bestimmen
den Typ der Parameter der Ausgabefunktion und die Art der Konver-
tierung und Ausgabe. Alle Zeichen, die kein Formatkommando sind,
also nicht mit dem Zeichen "%" beginnen sind Ausgabezeichen und
werden unverändert ausgegeben.

Der Aufbau eines Formatkommandos:

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

Die Angaben in den eckigen Klammern können optional angegeben werden.

flags
"-" zur Linksjustierung;
"+" zur Ausgabe auch eines positiven Vorzeichens bei Zahlen;
"0" zur Ausgabe führender Nullen bei Zahlen;
"#" hat keine Auswirkung

width
Feldbreite als dezimale Ziffernfolge oder "*", in diesem Fall wird
```

die Feldbreite als nächstes Argument des Typs int übergeben. Die
Feldbreite ist immer ein minimaler Wert, zu lange Ausgaben werden
nicht beschnitten.

limit
Die Genauigkeit als dezimale Ziffernfolge oder "*", in diesem Fall
wird die Genauigkeit als nächstes Argument des Typs int übergeben.
Der Wert beschreibt die maximale Anzahl von Zeichen bei Ausgabe
einer Zeichenkette oder die minimale Anzahl von Ziffern einer ganz-
zahligen Ausgabe.

size
Längenangabe des Arguments:
"h" für ein Argument des Typs short int oder unsigned short int;
"l" für ein Argument des Typs long int oder unsigned long int;

type
Typangabe des Arguments:

"d"
"i" zur Ausgabe einer vorzeichenbehafteten Dezimalzahl, das zuge-
    hörige Argument ist vom Typ int

"c" zur Ausgabe eines Zeichens, das zugehörige Argument ist vom
    Typ int und wird in unsigned char konvertiert

"s" zur Ausgabe einer Zeichenkette, die mit einem Nullzeichen abge-
    schlossen ist, das zugehörige Argument ist vom Typ char *

"n" zur Speicherung der Anzahl der bisher von diesem Funktionsaufruf
    ausgegebenen Zeichen in der Variablen, auf die das Argument vom
    Typ int * zeigt, es erfolgt keine Ausgabe

"%" zur Ausgabe eines Prozentzeichens

Alle anderen Typangaben führen zu undefinierten Ausgaben.

1.58 Eingabeformatstring

Der Eingabeformatstring "format" zur formatierten Eingabe
Der Formatstring "format" zur formatierten Eingabe besteht aus den
Formatkommandos und Eingabezeichen. Die Formatkommandos bestimmen
den Typ der Parameter der Eingabefunktion und die Art der Konver-
tierung und Eingabe. Alle Zeichen, die kein Formatkommando sind,
also nicht mit dem Zeichen "%" beginnen und keine Trennzeichen (Leer-
zeichen, Tabulatoren und Zeilenvorschübe) sind, sind Eingabezeichen
und werden unverändert in der Eingabe erwartet.

Der Aufbau eines Formatkommandos:

% [width] [size] type

Die Angaben in den eckigen Klammern können optional angegeben werden.

width

```
Die Anzahl der zu lesenden Zeichen als dezimale Ziffernfolge oder "*",
in diesem Fall werden die Zeichen zwar gelesen, aber nicht in das
nächste Argument übertragen.

size
Längenangabe des Arguments:
"h" für ein Argument des Typs short int oder unsigned short int
"l" für ein Argument des Typs long int oder unsigned long int

type
Typangabe des Arguments:

"d" zur Eingabe einer vorzeichenbehafteten dezimalen Ganzzahl, das zuge-
    hörige Argument ist vom Typ int *

"i" zur Eingabe einer vorzeichenbehafteten Dezimalzahl, Oktalzahl (bei
    führender '0') oder Hexadezimalzahl (bei führendem '0x' oder '0X'),
    das zugehörige Argument ist vom Typ int *

"o" zur Eingabe einer Oktalzahl, das zugehörige Argument ist vom Typ int *

"x" zur Eingabe einer Hexadezimalzahl mit oder ohne '0x', das zugehörige
    Argument ist vom Typ int oder unsigned int

"c" zur Eingabe von "width" Zeichen, wobei Leerzeichen und Zeilentrenner
    nicht überlesen werden und kein Nullzeichen angehängt wird, das zuge-
    hörige Argument ist vom Typ char *

"s" zur Eingabe einer Zeichenkette, wobei führende Leerzeichen und Zeilen-
    trenner überlesen werden und ein Nullzeichen angehängt wird, das zuge-
    hörigen Argument ist vom Typ char *

"e"
"f"
"g" zur Eingabe einer Fließkommazahl in beliebiger Darstellung, das zuge-
    hörige Argument ist vom Typ float *

"p" zur Eingabe einer hexadezimalen Speicheradresse, das zugehörige Argument
    ist vom Typ int *

"n" zur Speicherung der Anzahl der bisher von diesem Funktionsaufruf gele-
    senen Zeichen in der Variablen, auf die das Argument vom Typ int * zeigt,
    es wird kein Zeichen gelesen.

"[...]"  zur Eingabe einer Zeichenkette, die nur aus den in den eckigen
   Klammern angegebenen Zeichen besteht, wobei ein Nullzeichen ange-
   hängt wird, das zugehörigen Argument ist vom Typ char *

"[^...]" zur Eingabe einer Zeichenkette, die nur aus Zeichen besteht, die
   nicht in den eckigen Klammern angegebenen sind, wobei ein Null-
   zeichen angehängt wird; das zugehörigen Argument ist vom Typ char *

"%%"     zur Eingabe eines Prozentzeichens

Alle anderen Typangaben führen zu undefinierten Eingaben.
```

1.59 Eingabeformatstring_

```
Der Eingabeformatstring_ "format" zur formatierten Eingabe
Der Formatstring "format" zur formatierten Eingabe besteht aus den
Formatkommandos und Eingabezeichen. Die Formatkommandos bestimmen
den Typ der Parameter der Eingabefunktion und die Art der Konver-
tierung und Eingabe. Alle Zeichen, die kein Formatkommando sind,
also nicht mit dem Zeichen "%" beginnen und keine Trennzeichen (Leer-
zeichen, Tabulatoren und Zeilenvorschübe) sind, sind Eingabezeichen
und werden unverändert in der Eingabe erwartet.

Der Aufbau eines Formatkommandos:

% [width] [size] type

Die Angaben in den eckigen Klammern können optional angegeben werden.

width
Die Anzahl der zu lesenden Zeichen als dezimale Ziffernfolge oder "*",
in diesem Fall werden die Zeichen zwar gelesen, aber nicht in das
nächste Argument übertragen.

size
Längenangabe des Arguments:
"h" für ein Argument des Typs short int oder unsigned short int
"l" für ein Argument des Typs long int oder unsigned long int

type
Typangabe des Arguments:

"d" zur Eingabe einer vorzeichenbehafteten dezimalen Ganzzahl, das zuge-
    hörige Argument ist vom Typ int *

"i" zur Eingabe einer vorzeichenbehafteten Dezimalzahl, Oktalzahl (bei
    führender '0') oder Hexadezimalzahl (bei führendem '0x' oder '0X'),
    das zugehörige Argument ist vom Typ int *

"o" zur Eingabe einer Oktalzahl, das zugehörige Argument ist vom Typ int *

"x" zur Eingabe einer Hexadezimalzahl mit oder ohne '0x', das zugehörige
    Argument ist vom Typ int oder unsigned int

"c" zur Eingabe von "width" Zeichen, wobei Leerzeichen und Zeilentrenner
    nicht überlesen werden und kein Nullzeichen angehängt wird, das zuge-
    hörige Argument ist vom Typ char *

"s" zur Eingabe einer Zeichenkette, wobei führende Leerzeichen und Zeilen-
    trenner überlesen werden und ein Nullzeichen angehängt wird, das zuge-
    hörigen Argument ist vom Typ char *

"p" zur Eingabe einer hexadezimalen Speicheradresse, das zugehörige Argument
    ist vom Typ int *

"n" zur Speicherung der Anzahl der bisher von diesem Funktionsaufruf gele-
    senen Zeichen in der Variablen, auf die das Argument vom Typ int * zeigt,
    es wird kein Zeichen gelesen.
```

```
"[...]"  zur Eingabe einer Zeichenkette, die nur aus den in den eckigen
   Klammern angegebenen Zeichen besteht, wobei ein Nullzeichen ange-
   hängt wird, das zugehörigen Argument ist vom Typ char *

"[^...]" zur Eingabe einer Zeichenkette, die nur aus Zeichen besteht, die
   nicht in den eckigen Klammern angegebenen sind, wobei ein Null-
   zeichen angehängt wird; das zugehörigen Argument ist vom Typ char *

"%%"     zur Eingabe eines Prozentzeichens

Alle anderen Typangaben führen zu undefinierten Eingaben.
```

1.60 Eingabeformatstring__

```
Der Eingabeformatstring__ "format" zur formatierten Eingabe
Der Formatstring "format" zur formatierten Eingabe besteht aus den
Formatkommandos und Eingabezeichen. Die Formatkommandos bestimmen
den Typ der Parameter der Eingabefunktion und die Art der Konver-
tierung und Eingabe. Alle Zeichen, die kein Formatkommando sind,
also nicht mit dem Zeichen "%" beginnen und keine Trennzeichen (Leer-
zeichen, Tabulatoren und Zeilenvorschübe) sind, sind Eingabezeichen
und werden unverändert in der Eingabe erwartet.

Der Aufbau eines Formatkommandos:

% [width] [size] type

Die Angaben in den eckigen Klammern können optional angegeben werden.

width
Die Anzahl der zu lesenden Zeichen als dezimale Ziffernfolge oder "*",
in diesem Fall werden die Zeichen zwar gelesen, aber nicht in das
nächste Argument übertragen.

size
Längenangabe des Arguments:
"h" für ein Argument des Typs short int oder unsigned short int
"l" für ein Argument des Typs long int oder unsigned long int

type
Typangabe des Arguments:

"d" zur Eingabe einer vorzeichenbehafteten dezimalen Ganzzahl, das zuge-
    hörige Argument ist vom Typ int *

"i" zur Eingabe einer vorzeichenbehafteten Dezimalzahl, Oktalzahl (bei
    führender '0') oder Hexadezimalzahl (bei führendem '0x' oder '0X'),
    das zugehörige Argument ist vom Typ int *

"c" zur Eingabe von "width" Zeichen, wobei Leerzeichen und Zeilentrenner
    nicht überlesen werden und kein Nullzeichen angehängt wird, das zuge-
    hörige Argument ist vom Typ char *

"s" zur Eingabe einer Zeichenkette, wobei führende Leerzeichen und Zeilen-
```

```
    trenner überlesen werden und ein Nullzeichen angehängt wird, das zuge-
    hörigen Argument ist vom Typ char *

"n" zur Speicherung der Anzahl der bisher von diesem Funktionsaufruf gele-
    senen Zeichen in der Variablen, auf die das Argument vom Typ int * zeigt,
    es wird kein Zeichen gelesen.

"[...]"  zur Eingabe einer Zeichenkette, die nur aus den in den eckigen
   Klammern angegebenen Zeichen besteht, wobei ein Nullzeichen ange-
   hängt wird, das zugehörigen Argument ist vom Typ char *

"[^...]" zur Eingabe einer Zeichenkette, die nur aus Zeichen besteht, die
   nicht in den eckigen Klammern angegebenen sind, wobei ein Null-
   zeichen angehängt wird; das zugehörigen Argument ist vom Typ char *

"%%"     zur Eingabe eines Prozentzeichens

Alle anderen Typangaben führen zu undefinierten Eingaben.
```

1.61 Beispiele

```
Beispiele:
~~~~~~~~~~

Um die Vorteile und Anwendungsmöglichkeiten der "stormamiga.lib"
etwas zu verdeutlichen, habe ich ein paar Beispiele beigelegt.

Die Beispiele mit dem Namen "...-storm" werden mit der "storm.lib"
und dem Startupcode "startup.o" gelinkt.

Die Beispiele mit dem Namen "...-stormamiga" werden mit der
"stormamiga.lib" und dem Startupcode "stormamiga_startups.o" gelinkt.

Die Beispiele mit dem Namen "...-stormamiga-2" werden mit der
"stormamiga.lib" und dem Startupcode "stormamiga_startups.o" gelinkt.
Außerdem wurde der Quelltext für die "stormamiga.lib" optimiert.
```

1.62 Bekannte Fehler

```
Bekannte Fehler:
~~~~~~~~~~~~~~~~

- Der Befehl "printf" hat bei der Ausgabe von Fließkommazahlen einen
  Rundungsfehler.
```

1.63 Updates

```
                Updates:
~~~~~~~~
```

Wenn Sie Zugang zum Internet haben, dann können Sie die Updates auf
der HomePage "http://home.pages.de/~haage" der Haage & Partner
Computer GmbH bekommen.
Sie können die neueste Version natürlich auch direkt von
                mir
                  bekommen.
Allerdings müssen Sie mir dann einen frankierten Briefumschlag und
eine Diskette (HD oder DD) zuschicken.

1.64 Kopierrecht

Kopierrecht:
~~~~~~~~~~~~

Die "stormamiga.lib" ist FREEWARE. Sie darf frei kopiert werden, solange
sie in KEINSTER Weise verändert wird und solange ALLE dazugehörigen
Dateien UNVERÄNDERT mitkopiert werden.

Die "stormamiga.lib" darf auch im Zusammenhang mit anderen Programmen
verwendet und vertrieben werden, solange KLARGESTELLT ist, daß es sich um
FREEWARE handelt UND solange ALLE Dateien UNVERÄNDERT mitkopiert werden.
Mit der Weitergabe der "stormamiga.lib" darf kein Gewinn erzielt werden.
Der Verkaufspreis einer Diskette, die die "stormamiga.lib" enthält, darf
nicht mehr als 5,- DM betragen. Ausgenommen davon sind Disketten, die es
zu Zeitschriften gibt.

Eine Reassemblierung der "stormamiga.lib" ist selbstverständlich NICHT
gestattet.

AM WICHTIGSTEN:

Die Benutzung der "stormamiga.lib" erfolgt AUSSCHLIEßLICH auf eigenes Risiko.

Der Autor kann auf KEINEN FALL für einen Schaden oder Datenverlust der direkt
oder indirekt mit dem Gebrauch der "stormamiga.lib" entstehen sollte verant-
wortlich gemacht werden.

Alle Rechte vorbehalten. Für Fehlermitteilungen oder Verbesserungsvorschläge
bin ich jederzeit dankbar.

1.65 Geschichte

Geschichte:
~~~~~~~~~~~

V41.000 alpha - V41.002 alpha (18.03. - 21.03.1996):
----

  interne Entwicklungsphase

- Funktionen der "amiga.lib" für MC68000 oder höher geschrieben

```
V41.003 alpha - V41.021 alpha (22.03. - 16.05.1996):
----------------------------------------------------

  interne Entwicklungsphase

- Funktionen der "amiga.lib" für MC68EC020 oder höher umgeschrieben (ab
  V41.003 alpha wird ein MC68EC020 oder höher benötigt)
- Ein- und Ausgaberoutinen geschrieben (GCC-kompatibel)
- Funktionen zum automatischen Öffnen und Schließen der Libraries ge-
  schrieben
- die meisten ctype-Funktionen geschrieben
- einige stdio-, string- und stdlib-Funktionen geschrieben
- Startupcode für die "stormamiga.lib" optimiert
- einige andere Funktionen geschrieben
- einige Optimierungen und Fehlerkorrekturen
- Includedatei "stormamiga.h" geschrieben


V41.022 alpha - V41.029 alpha (20.05. - 11.06.1996):
----------------------------------------------------

  interne Entwicklungsphase

- Ein- und Ausgaberoutinen komplett neu geschrieben (StormC-kompatibel)
- einige stdio-Funktionen geschrieben
- 64Bit Befehle der "storm.lib" für MC68EC020+ optimiert
- einige andere Funktionen geschrieben
- einige Optimierungen und Fehlerkorrekturen
- Includedatei "stormamiga.h" erweitert
- Anleitung geschrieben


Hinweis:

Durch ein Versehen wurde die Version 41.028 alpha, einige uralte Beispiel-
programme und Teile der Anleitung, mit der Version 1.1 von StormC, veröffent-
licht. Diese Version hatt aber noch einige größere Fehler und funktioniert
nicht mit den alten Beispielprogrammen.
Die erste, zur Veröffentlichung gedachte Version, ist die Version 41.032 beta.


V41.030 beta - V41.031 beta (13.06. - 15.06.1996):
--------------------------------------------------

  interne Entwicklungs- und Testphase

- Optimierung der Ein- und Ausgaberoutinen
- Fehlerkorrekturen
- Anleitung überarbeitet


V41.032 beta (16.06.1996):
--------------------------

  Erste öffentliche Version


V41.033 beta (17.06. - 15.07.1996):
-----------------------------------
```

```
- umfangreiches Betatesting
- Fehlerkorrektur der string-Funktion "memcpy" (Die Benutzung von "memcpy"
  führte zu einem Fehler oder zum Systemabsturz. Die Ursache war ein Tipp-
  fehler. Ich hatte "a2" statt "a1" geschrieben.)
- die mathematischen Funktionen "acos", "asin", "atan", "ceil", "cos",
  "cosh", "exp", "fabs", "floor", "log", "log10", "pow", "sin", "sinh",
  "sqrt", "tan" und "tanh" geschrieben
- die string-Funktionen "memccpy", "strcoll", "strlwr", "strsep", "strupr",
  "strxfrm" und "swab" geschrieben
- die stdlib-Funktionen "abs", "labs", "atoi" und "atol" geschrieben
- die stdlib-Funktion "rand" optimiert
- die string-Funktionen "memchr", "memmove", "memset", "strcspn", "strpbrk",
  "strspn" und "strtok" optimiert
- Includedatei "stormamiga.h" überarbeitet und erweitert


V41.034 (16.07. - 31.07.1996):
------------------------------
- Fehlerkorrektur der internen Funktion "amigawrite" (Der auszugebende Text wurde
  erst nach einigen Sekunden angezeigt. Bei dem Programm "GadTools" wurde der Text
  erst ausgegeben, wenn 5 Schalter gedrückt wurden.)
- die string-Funktionen "strerror" und "stricmp" geschrieben
- die string-Funktionen "memcmp", "strcat", "strcmp", "strcpy", "strncat",
  "strncmp", "strncpy" und "strstr" optimiert
- die stdlib-Funktionen "malloc" und "free" optimiert
- die stdio-Funktionen "vfprintf", "vfprintf_", "vfprintf__", "vfscanf",
  "vfscanf_", "vfscanf__", "fflush" und "setvbuf" optimiert
- die internen Funktionen "amigaread", "amigareadunget", "amigawrite", "amiga-
  eof", "amigaseek", "amigagetc", "amigagetcunget", "amigaungetc", "amigaputc",
  "amigaflush", "amigaclose", "SMult64", "UMult64", "SDiv64", "INIT_0_InitFiles",
  "EXIT_5_InitFiles", "INIT_5_InitStdIOFiles", "EXIT_5_InitStdIOFiles" und
  "EXIT_4_free" optimiert
- "aufräumen" der "stormamiga.lib" (dadurch wird das Linken beschleunigt,
  die Programme etwas schneller und die Map-Datei übersichtlicher)
- Anleitung überarbeitet


V41.035 (02.08. - 17.08.1996):
------------------------------
- die string-Funktionen "strcmp", "strcoll" und "strxfrm" neu geschrieben
- die stdlib-Funktionen "llabs",  "atoll", "strtol", "strtoll", "strtoul",
  "strtoull", "inttostr", "llongtostr", "uinttostr" und "ullongtostr"
  geschrieben
- die stdio-Funktion "puts" geschrieben
- die Spezial-Funktionen "muls64" und "mulu64" geschrieben
- Includedatei "stormamiga.h" erweitert
- Benutzer-Lexikon mit allen Sonder-Funktionen der "stormamiga.lib" geschrieben
- Installerskript geschrieben
- Anleitung überarbeitet (Beschreibung der Funktionen neu geschrieben, Index
  hinzugefügt und an AmigaOS 3.0 angepaßt); Anleitung als ASCII-Text beigelegt
```

1.66 In Zukunft

```
In Zukunft:
~~~~~~~~~~~
```

Die folgenden Punkte habe ich mir für die nächsten Versionen der
"stormamiga.lib" vorgenommen.

- alle, noch fehlenden, Funtionen der "storm.lib" und der "amiga.lib"
  in die "stormamiga.lib" integrieren

- Unterstützung des kleinen Datenmodelles a4 (eventuell auch a6)

- spezielle Version der "stormamiga.lib" für die Koprozessoren
  MC68881 und MC68882 (mit "68040.library" bzw. "68060.library"
  auch für den MC68040 und MC68060)

- spezielle Version der "stormamiga.lib" für den MC68040 (mit
  "68060.library" auch für den MC68060)

1.67 Danksagungen

Danksagungen:
~~~~~~~~~~~~~

Als erstes möchte ich mich bei der Haage & Partner Computer GmbH
bedanken, weil sie mir ihre Entwicklerunterlagen und Quelltexte
kostenlos überlassen haben. Ohne diese Unterlagen währe ich wahr-
scheinlich an den Ein- und Ausgaberoutinen für die Befehle printf
und scanf verzweifelt.
Besonderen Dank an Jochen Becher, der auch an Sonntagen zu später
Stunde Zeit für meine Probleme hatte.

Ohne die Unterstützung der Haage & Partner Computer GmbH würde
es die "stormamiga.lib", in dieser Form, nicht geben.

Außerdem möchte ich mich bei folgenden Leuten bedanken:

- Uwe Schienbein, für Betatesting, Bugreports und neue Ideen
- Carsten Bornholz, für Betatesting
- Dietmar Heidrich, für seinen "OMA"
- Frank Wille, für seinen "FreePhxAss"

1.68 Autor

Autor:
~~~~~~

Matthias Henze
Gorkistraße 119
04347 Leipzig
Deutschland

Telefon: 0341/2326414

1.69 Index

```
                Index:
~~~~~~

 A


                 Ausgabeformatstring_

                 Ausgabeformatstring__

                 Autor
                 B


                 bcmp

                 bcopy

                 Beispiele

                 Bekannte Fehler

                 bzero
                 D


                 Danksagungen

                 Der Startupcode

                 divsl

                 divul
                 E


                 Eingabeformatstring

                 Eingabeformatstring_

                 Eingabeformatstring__

                 Einleitung
                 F


                 ffs

                 fprintf_

                 fprintf__

                 fscanf_
```

sscanf_

sscanf__

strcoll

Stringpuffer

strsep

strxfrm

swab

Systemanforderungen
U

Updates
V

vfprintf_

vfprintf__

vfscanf

vfscanf_

vfscanf__

vprintf_

vprintf__

vscanf

vscanf_

vscanf__

VSPRINTF

vsprintf_

vsprintf__

vsscanf

vsscanf_

vsscanf__