GNU C/C++

Christian Felsch

COLLABORATORS			
	TITLE : GNU C/C++		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Christian Felsch	January 16, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

GNU C/C++

1.1 titel

```
                Titel                                                    ↩
                   GNU C/C++



                          GNU C/C++ auf dem Atari

                                   von

                             Christian Felsch


   Inhalt:

      Vorwort

      Installation


                Tips & Tricks zu GNU C auf dem Atari
                      nderungen
```

1.2 vorwort

Vorwort GNU C/C++

Da ich im MausNet immer mal wieder Anfragen dem GNU-Compiler und zu dessen Installation lese, ist es wohl Zeit, mal ein wenig darber aufzuschreiben.

Dieser Hypertext soll bei der Installation des GNU Compilers auf dem Atari helfen. Er ist keine generelle Anleitung zum Thema GNU C, es wird nur auf die Atari-spezifischen Dinge eingegangen. Wer mehr ber die Benutzung von GNU C erfahren will, den verweise ich auf die entsprechenden Dokumentationen.

Durch die Herkunft des GNU Compilers aus der UNIX-Welt sind fr dessen Installation einige Schritte ntig, die man sonst vom Atari her nicht kennt. Unter MiNT mit einem MinixFS ist die Installation vergleichbar mit der unter UNIX.
Immer mehr User wollen den Compiler aber unter anderen Systemen (z.B. MagiC) benutzen wollen und da dort vieles der UNIX-Umgebung fehlt, sind bei der Installation des Compilers einige Dinge zu beachten.

Im Vorfeld zu dieser Dokumentation habe ich den Copiler in der aktuellen Version 2.8.1 auf meinem TT selbst bersetzt. Wichtigster Grund dafr war folgender:
Die letzten Versionen (ab 2.6) wurden von verschiedenen Leuten fr den Atari portiert, allerdings immer mit der Einschrnkung, da MiNT mit einem MinixFS vorausgesetzt wurde.
Die Version 2.5.8 war die letzte, die unter allen Systemen lief und sich auch auf dem normalen TOS-Dateisystem installieren lie.

Ich habe die wesentlichen Portierungen (2.5.8 und 2.7.2) genommen und so in die aktuellen Quellen integriert, da am Ende wieder ein Compiler heraus kam, der wieder auf allen Systemen benutzbar ist.

Alle in diesem Dokument benutzen Dateinamen (der Programme, Archive usw.) beziehen sich auf das von mir zusammengestellte Paket, da im InterNet zu finden ist:

ftp://ftp.rz.tu-harburg.de/pub/software/systems/atari/gnu/gcc

1.3 installation

Die Installation ↩
GNU C/C++

Bevor man sich an die Installation macht, mu sichergestellt sein, da alle bentigten Komponenten des Compilers verfgbar sind:

Compiler

Tools

Bibliotheken

Dokumentation

bersicht der Archive
Die Installation lt sich in zwei Teile aufteilen:

Installation auf einem Minix-FS

Installation auf einem VFAT-FS

Installation auf einem TOS-FS

berprfen der Installation

1.4 compiler

Komponenten: der Compiler GNU C/C++

Der eigentliche Compiler besteht aus fnf einzelnen Programmen:

gcc.ttp	Treiberprogramm, welches vom Anwender gestartet wird und nach einander alle ntigen Programme aufruft. Mit ihm werden C Quelltexte bersetzt.
cpp.ttp	Der C/C++ Prprozessor. Wird von gcc.ttp gestartet, so da der Anweder im Normalfall nichts davon sieht.
cc1.ttp	Der eigentlche C-Compiler. Wird ebenfalls von gcc.ttp gestartet.
g++.ttp	Treiberprogramm fr C++ Quelltexte. Es startet gcc.ttp mit einigen C++ spezifischen Parametern.
cc1plus.ttp	Der eigentliche C++ Compiler.

1.5 tools

Komponenten: die Tools GNU C/C++

Neben dem Compiler bentigt man noch einige andere Tools. Zwei sind unverzichtbar:

as.ttp	Der Assembler, der aus den Assemblerdateien die Objektdateien erzeugt.
ld.ttp	Der Linker, der aus Objektdateien und Bibiliotheken das ausfhrbare Programm erzeugt.

Neben Assembler und Linker gibt es noch diverse andere Tools, die nicht unbedingt erforderlich sind, aber zustzliche Funktionen zur Verfgung stellen:

ar.ttp	Der Archivator. Er wird bentigt, wenn man mehrere Objektdateien zu einer Bibliothek zusammenfassen mchte.
cnm.ttp	Zeigt die Symbol-Namen an, die in einem ausfhrbaren Programm enthalten sind.
fixstk.ttp	Stellt die Stacksize eines Programms ein. Nheres dazu unter Tips+Tricks.

nm.ttp	Zeigt die Symbol-Name in Objektdateien oder Bibliotheken an. Damit kann man z.B. nach bestimmten Funktionen in einer Bibiliothek suchen.
printstk.ttp	Zeigt die eingestellte Stacksize an. Nheres dazu unter Tips+Tricks.
size.ttp	Zeigt die Gre der Programmbereiche (Text, Data, Bss) in Objektdateien oder Bibiliotheken an.
size68.ttp	Zeigt Informationen ber ein aufrhrbares Programm an.
strings.ttp	Zeigt alle 'lesbaren' Zeichenketten (also ASCII-Werte zwischen 32 und 127) an. Damit kann man z.B. fest in ein programm eingebaute Pfade finden.
strip.ttp	Entfernt die Symboltabelle aus einem Programm.
sym-ld.ttp	Symbolischer Linker, wird zum bentigt.
toglclr.ttp	Dient zum Anzeigen bzw. Einstellen der 'Programm-Header-Bits'.

1.6 fixstk.ttp

fixstk - einstellen GNU C/C++

Mit dem Programm fixstk.ttp kann die eines Programms verndert werden.

Aufruf: fixstk size file ...

Der Wert von <size> gibt dabei die Anzahl der Bytes an. Einige Werte haben eine besondere Funktion, nheres dazu unter Stacksize.
Bei der Angabe der Gre kann eine Abkrzung angegeben werden. Dabei steht ein 'k' hinter der Zahl fr Kilobyte und ein 'm' fr Megabyte.
Gltige Werte fr <size> sind also z.B. 4096, 4k oder 2m.
Das Programm zeigt nach dem Aufruf die vorher eingestellte und die neue an:

```
/tmp > fixstk 4k test.app
test.app: _stksize was 0 (0K)
test.app: _stksize now is 4096 (4K)
```

Sollte fixstk einen Fehler der Art 'no symbol table' melden, kann es das Symbol _stksize nicht mehr finden (es wurde wegge'stripped!) und man kann nichts mehr verndern.

1.7 nm.ttp

nm.ttp - Symbole anzeigen GNU C/C++

Das Programm listet die in einer Objektdatei oder Bibliothek enthaltenen Symbole auf.

Aufruf: nm [option] file

Folgende Optionen stehen zur Verfgung:

-a Wenn die Objektdatei mit -g erzeugt wurde, werden spezielle Debug-Infos angezeigt.

-g Nur globale Symbole listen.

-n Nach Adressen (erste Spalte) sortiert, sonst alphabetisch.

-o Dateiname der Objektdatei mit ausgeben. Sinnvoll z.B. bei Bibliotheken.

-r Ausgabe umgekehrt sortiert.

-u Nur undefinierte Symbole, also solche die nicht in der Lib deklariert sind, anzeigen.

Die Ausgabe von nm sieht so aus:

```
00000000 t ___gnu_compiled_c
         U ___mint
         U ___open_stat
00000684 d ___umask
         U __enoent
00000000 t __get_umask
         U __unx2dos
00000630 T _creat
         U _errno
         U _fstat
         U _isatty
00000046 T _open
         U _strncmp
         U _strtoul
0000064a T _umask
00000000 t gcc2_compiled.
```

In der ersten Spalte wird die relative Adresse ausgegeben. Der einzelne Buchstabe klassifiziert den Typ des Symbols. Folgende Typen gibt es:

C Eine Variable, die definert aber nicht initialisert ist.

b Eine lokal (static) definierte Variable.

D Eine Variable, die bei ihrer Definition initialisiert wird.

d Eine lokal (static) definierte Variable mit Initialisierung.

t,
T Funktionsnamen. ’t’ bei static sonst ’T’.

U Undefinierte (externe) Symbole.

1.8 strip.ttp

```
strip.ttp - Symboletabelle lschen                                GNU C/C++

Mit dem Programm kann die Symboltabelle eines Programms entfernt werden.
Das Programm belegt dadurch weniger Platz auf der Platte.

Aufruf:  strip [option] file ...

Folgende Optionen stehen zur Verfgung:

   -a          Symboltabelle komplet lschen. Sollte man nicht verwenden,
               da dann die Stacksize nicht mehr einstellbar ist.

   -g          Alle Symbole bis auf die globalen lschen.

   -k          Erhlt das Symbol fr die Stacksize (_stksize).

   -l names    Alles Symbole bis auf die angegebenen lschen.

   -t          TurboC/PureC-Programm strippen.
```

1.9 toglclr.ttp

```
toglclr.ttp - Programheader manipulieren                         GNU C/C++

Mit dem Programm knnen die Bits im Programheader manipuliert werden.

Aufruf:  toglcr [option] file ...

Folgende Optionen stehen zur Verfgung:

   -fload      Fast-Load Bit.

   -frun       Programm in TT-RAM legen.

   -fram       Programm bekommt TT-RAM.

   -fshare     Shared-Text Bit.

   -private    MiNT MemoryProtection: Speicher privat.

   -global     MiNT MemoryProtection: Speicher frei fr alle.

   -super      MiNT MemoryProtection: Speicher frei fr alle im Supermode.

   -readable   MiNT MemoryProtection: Speicher lesbar fr alle.
```

1.10 bibliotheken

```
Komponenten: die Bibliotheken                                    GNU C/C++
```

Zu jedem C-Compiler gehren verschiedene Bibliotheken. Auf jedenfall bentigt wird die Standard C Bibiliothek. Fr den Atari gibt es zwei verschiedene Versionen dieser Standardbibliothek.
Zum einen ist das die GNU-Lib in der Version PL 98, die schon relativ alt ist und nicht mehr weiterentwickelt wird.
Zum anderen ist da die sogenannte MiNT-Lib, die stmliche MiNT-Funktionen untersttzt und inzwischen auch wieder weiterentwickelt wird.
Ich werde mich hier auf die Beschreibung der MiNT-Lib als Standard-bibiliothek beschrnken.

Einige Bibliotheken liegen in verschiedenen Ausfhrungen vor. Wenn ein Bibliotheksname auf '16' endet, bedeutet das, da der verwendete Integertyp nicht wie blich bei GNU 32bit sondern nur 16bit gro ist. Es gibt einige Programme, die sich nur mit 16bit Integer bersetzen lassen (bei PureC ist 16bit Standard), die dann diese Bibliotheken bentigen. Eine Mischung von 16bit und 32bit Objektdateien und Bibliotheken frht zu nicht funktionstchtigen Programmen!

Auerdem gibt es noch Bibliotheken, deren Name mit einem 'b' beginnen. Dies sind Bibliotheken, die das Text-Sharing (ein mehrfach gestartetes Programm belegt mit seinem Textsegment nur einmal RAM) von MiNT untersttzen.

Ab GCC 2.8.1.c und Utils 41 werden die Libs anders benannt:
- die Namen enden nicht mehr auf .olb sondern auf .a
- die C Bibliothek wurde von gnu.* auf c.* umgenannt
- die Mathe Bibliothek wurde von pml.* nach m.* umgenannt

Folgende Bibiliotheken sind verfgbar:

crt.o	Der Startupcode, der vom Linker zu jedem Programm hinzugelinkt wird.
c.a c16.a	Die Standard C Bibiliothek (MiNT-Lib).
gem.a gem16.a	Die Schnittstelle zum AES/VDI
iio.a iio16.a	Enthlt Funktionen fr Integer-IO.
m.a m16.a	Mathematische Standardfunktionen.
stdc++.a g++.a	Die beiden Standard-Bibliotheken fr C++.

Neben den genannten gibt es noch eine Reihe weiterer Bibiliotheken, die nicht zu meinem GNU-Paket gehren aber wenigstens genannt werden sollen:
- curses
- portlib
- socket
- gem++

1.11 dokumentation

```
Dokumentation zu GNU C                                          GNU C/C++

Zum GNU Compiler gibt es jede Menge Dokumentation, die in den Quellen
des Compiler zu finden ist.
In meinem Doku-Archiv sind neben diesem Hypertext noch ein paar weitere
Dokumentationen, die sich speziell mit GNU C auf dem Atari beschftigen
(alle auf Englisch):

   gcc-st.tex     Die ursprngliche Doku von Frank Riddebusch. Zu
                  groen Teilen ist sie noch gltig.

   lib-card.tex   Eine Doku zur MiNT-Lib, als RefCard zusammengefat.
                  Sie bietet einen guten berblick ber die Lib, ist
                  aber leider nicht vollstndig.

   var-card.tex   Eine bersicht der Betriebssystem-Struturen, die in
                  der MiNT-Lib benutzt und definiert sind.

Fr alle, die kein TeX haben, sind die Dokus auch als Postscript
verfgbar. Und wer auch kein Ghostscript zum Anschauen/Drucken hat,
hat selber schuld ;-)
```

1.12 installation, minix

```
Installation auf einem MinixFS                                  GNU C/C++

Die Installation unter MiNT auf einem MinxFS stellt sich genau so dar,
wie GNU C es von -Systemen her gewhnt ist.
Da ich nicht zwei verschiedene Archive erstellen wollte, mssen alle,
die den Compiler in dieser Art installieren woll, die Dateien in die
UNIX-Namen umbenennen :-)
Bei der unten angegebenen Verzeichnisstruktur wird der jeweilige Name
im Archiv in () angegeben.

   /usr/lib/gcc-lib/m68k-atari-mint/2.8.1
      Enthlt den eigentlichen Compiler:
         cc1         (bin/cc1.ttp)
         cc1plus     (bin/cc1plus.ttp)
         cpp         (bin/cpp.ttp)
         specs       (lib/specs)

   /usr/m68k-atari-mint/bin
      Enthlt die Programme, die vom gcc gestartet werden:
         as          (bin/as.ttp)
         ld          (bin/ld.ttp)
         sym-ld      (bin/ld.ttp)

   /usr/bin
      Alle Programme, die der User benutzen will, mssen so
      installiert sein, das sie gefunden werden. Das Verzeichnis
      mu in der Environmentvariable PATH enthalten sein.
         gcc         (bin/gcc.ttp)
```

```
         g++          (bin/g++.ttp)
         ar           (bin/ar.ttp)
         ...

   /usr/lib
   /usr/local/lib
   /usr/m68k-atari-mint/lib
   /usr/lib/gcc-lib/m68k-atari-mint/2.8.1/lib
      Hier sucht der Linker nach den Bibliotheken.
      Dabei geht er von den UNIX-Namen aus, die mit 'lib' beginnen
      und auf '.a' enden.
         libgnu.a       (lib/gnu.olb)
         libgem.a       (lib/gem.olb)
         crt0.o         (lib/crt0.o)
         ...
      Achtung: Der Linker entscheidet anhand des Vorhandenseins
      der Environmentvariable GNULIB, ob er die UNIX- oder die TOS-
      Namen verwendet. Wer also die Minix-Installation verwendet,
      sollte sicherstellen, die diese Variable nicht gesetzt ist!

   /usr/local/include
   /usr/m68k-atari-mint/include
   /usr/lib/gcc-lib/m68k-atari-mint/2.8.1/include
   /usr/include
   /usr/lib/g++-include
      Hier sucht der Prprozessor nach den Header-Dateien.
      Der letzte Pfad wird nur bei C++ benutzt.

Fr den Compiler selbst sind keine speziell Environmentvariablen ntig.
Er beachtet aber folgende Standardvariablen:

   TMPDIR
   TEMP
   TMP      Der temporre Bereich, in dem einige Dateien zwischen-
            gelagert werden.

   PATH     Wer eine Kommandoshell benutzt, sollte den Pfad, wo die
            ausfhrbaren Programme (*.ttp) der Compilers liegen,
            darin mit angeben.
```

1.13 installation, vfat

```
Installation auf einem VFAT-FS                                  GNU C/C++

Durch das VFAT-System unter MagiC bzw. MiNT 1.15 ist es mglich, die
UNIX-Struktur nachzubilden.
Der folgende Text stamm von dem Schweizer Daniel Augsburger, der
sich die Mhe gemacht hat, seine MagiC-Installation zu dokumentieren.
Wer Fragen zur VFAT-Installation hat, mge ich bitte an Daniel wenden,
da ich so eine Installation nicht habe.

----

Das GNU C Paket kann auf einer beliebigen VFAT-Partition (lange
Dateinamen!) entsprechend der fr die Installation auf einem
```

Minix-Filesystem unter MiNT vorgegebenen Verzeichnisstruktur
installiert werden.

Folgende Verzeichnisstruktur hat sich bei mir unter MagiC bewhrt,
wobei sich das Verzeichnis 'usr' nicht zwingend im Wurzelverzeichnis
des Laufwerks befinden muss, sondern an beliebiger Stelle auch in
einem Unterverzeichnis stehen sein darf:

```
   /usr
      /bin
            ar.ttp
            cnm.ttp
            fixstk.ttp
            gcc.ttp
            nm.ttp
            printstk.ttp
            size.ttp
            size68.ttp
            strings.ttp
            strip.ttp
            toglclr.ttp

      /include
            *.h

      /lib
            *.a
            /gcc-lib
               /m68k-atari-mint
                  /2.7.2
                     cc1.ttp
                     cc1plus.ttp
                     cpp.ttp
                  /2.8.1
                     cc1.ttp
                     cc1plus.ttp
                     cpp.ttp
                     specs

      /m68k-atari-mint
            /bin
                     as.ttp
                     ld.ttp
                     sym-ld.ttp
```

Nun muss man noch einen Link von 'usr' auf dem Laufwerk U anlegen.
Damit man diesen Link nicht jedesmal nach dem Start von MagiC neu
anlegen muss, habe ich ein kleines Programm fr den Start-Ordner
von MagiC entwickelt, welches beliebige Links automatisch anlegt
(-> link2u, zu beziehen ber meine Homepage).
Anwender von Start Me Up! '98 knnen auch auf die dort implementierte
Funktion zurckgreifen (Kommando '/savelinks').

Mit dieser Verzeichnisstruktur ist es sogar mglich mehrere Versionen
von GNU C parallel installiert zu haben und zu verwenden. ltere
Versionen als 2.7.2 scheinen mit dieser Verzeichnisstruktur jedoch

```
nicht klar zu kommen, zumindest 2.5.8 als auch 2.6.1

Shell: Mupfel
=============

Auschnitt aus der Datei PROFILE fr die Mupfel:

#!mupfel
# Liste der Pfade, in denen nach Kommandos gesucht wird. export
PATH="$GEMINIHOME\bin;$GEMINIHOME\scripts;u:\usr\bin;."

# Verzeichnis fr temporre Dateien
export TMPDIR="$GEMINIHOME\tmp"


Betrieb des C-Compilers
=======================

Um mit GNU arbeiten zu knnen, muss auf Laufwerk U ein Link auf das
Verzeichnis 'usr' angelegt werden. Ausserdem muss man sich zum Arbeiten
mit GNU immer auf Laufwertk U befinden, da die GNU-Tools sonst die
bentigten Dateien nicht finden knnen. D.h. wenn man beispielsweise
G:\progs\foo\bar.c compilieren mchte, muss man auf dieses Verzeichnis
via Laufwerk U zugreifen, also ruft man gcc von U:\G\progs\foo\ aus auf.
Die GNU-spezifischen Variablen GNUEXEC, GNULIB, GNUINC brauchen somit
nicht zu existieren.


Autor: Kontakt
==============

Snail:  Daniel Augsburger
        Jasminweg 39
        8050 Zrich
        SCHWEIZ
Fon:    ++41 1 312 2238

Email:  skydiver@jumpgates.com
Inet:   http://www.jumpgates.com/skydiver/
        http://www.jumpgates.com/skydiver/d/neptun/
```

1.14 installation, tos

```
Installation auf einem TosFS                                    GNU C/C++

Wenn der Compiler nicht wie in der UNIX-Welt blich installiert wird,
weil z.B. kein Minix-Dateisystem zur Verfgung steht oder man den
Compiler mit einer anderen Struktur installieren mchte, knnen die
benutzten Suchpfade von auen eingestellt werden.
Diese Mglichkeit ist fr alle diejenigen interessant, die den Compiler
unter normalem TOS oder MagiC (ohne VFAT) benutzen mchten.

Im Gegensatz zur Installation auf einem MinixFS knnen die Dateien
so installiert werden, wie sie in den Archiven enthalten sind, brauchen
```

also nicht umbenannt werden. Auch die in einigen Archiven lngeren Namen, die auf einem TOSFS auf 8.3 verkrzt werden, sollten kein Problem darstellen.
Als zweckmig hat sich die folgende Struktur auf einer TOS-Partition X: erwiesen:

```
X:\gnu\bin
   Hier werden alle ausfhrbaren Programme (*.ttp) abgelegt.

X:\gnu\lib
   Hier liegen alle Bibliotheken (*.olb, *.o) und, ganz wichtig,
   die Datei 'specs'.

X:\gnu\include
X:\gnu\include\g++
   Hier liegen alle Header-Dateien (C und C++).
```

Um den Compiler mitzuteilen, wo er die Dateien findet, gibt es vier Environment-Variablen. Die Pfade, die in diese Variablen eingetragen werden, drfen sowohl '\' als auch '/' als Verzeichnistrenner enthalten. Sollen mehrere Pfade in einer Variablen eingetragen werden, sind diese durch ',' zu trennen.

```
GNUEXEC  ber diese Variable wird dem gcc.ttp mitgeteilt, wo
         Compiler, Assembler und Linker liegen. Der eingestellte
         Pfad mu mit einem '/' bzw. '\' enden, da er den
         einzelnen Programmen einfach nur vorangestellt wird und
         dabei ein gltiger Pfad entstehen mu.
         Im Beispiel:
            GNUEXEC=X:/gnu/bin/

GNULIB   Auf dem in dieser Variablen eingestellten Pfad werden die
         Bibliotheken und die 'specs'-Datei gesucht. Hier ist der
         abschlieende Verzeichnistrenner nicht ntig.
         Im Beispiel:
            GNULIB=X:/gnu/lib

GNUINC   Auf dem in dieser Variablen eingestellten Pfad werden die
         Standard C Header-Dateien gesucht.
         Im Beispiel:
            GNUINC=X:/gnu/include

GXXINC   Wer C++ machen mchte, mu in dieser Variablen den Pfad zu
         den C++ Header eintragen. Auerdem mu zustzlich noch der
         Pfad zu den Standard C Headern eingetragen werden.
         Im Beispiel:
            GXXINC=X:/gnu/include/g++,X:/gnu/include
```

Neben diesen absolut notwendigen Variablen, beachtet der Compiler noch folgende andere Variablen:

```
TMPDIR
TEMP
TMP      Der temporre Bereich, in dem einige Dateien zwischen-
         gelagert werden.

PATH     Wer eine Kommandoshell benutzt, sollte den Pfad, wo die
```

```
            ausfhrbaren Programme (*.ttp) der Compilers liegen,
            darin mit angeben.
```

1.15 installation, prfen

```
berprfung der Installation                                    GNU C/C++

Eine berprfung der Konfiguration kann durch die Option '-v' beim
Aufruf von gcc.tpp erfolgen.

Wenn man ein C-Testprogramm mit 'gcc -v test.c' bersetzt und die
Installation erfolgreich war, gibt es folgende Ausgaben:

installiert auf MinixFS

/tmp > gcc -v test.c
   Reading specs from /usr/lib/gcc-lib/m68k-atari-mint/2.8.1/specs
   gcc version 2.8.1
    /usr/lib/gcc-lib/m68k-atari-mint/2.8.1/cpp -lang-c -v -undef
    -D__GNUC__=2 -D__GNUC_MINOR__=8 -Dmc68000 -Datarist -Dgem -D__mc68000__
    -D__atarist__ -D__gem__ -D__mc68000 -D__atarist -D__gem -Asystem(tos)
    -Asystem(gem) -Acpu(m68k) -Amachine(m68k) test.c /tmp/cc482000.i
   GNU CPP version 2.8.1 (68k, MIT syntax)
   #include "..." search starts here:
   #include <...> search starts here:
    /usr/include
   End of search list.
    /usr/lib/gcc-lib/m68k-atari-mint/2.8.1/cc1 /tmp/cc482000.i -quiet
    -dumpbase test.c -version -o /tmp/cc482000.s
   GNU C version 2.8.1 (m68k-atari-mint) compiled by GNU C version 2.8.1.
    /usr/m68k-atari-mint/bin/as -v -mc68000 -o /tmp/cc4820001.o /tmp/cc482000.s
   GNU assembler version 2.9 (m68k-atari-mint)
    /usr/m68k-atari-mint/bin/ld -v /usr/lib/crt0.o -L/usr/lib/gcc-lib/
      m68k-atari-mint/2.8.1/tmp/cc4820001.o -lgnu
   /usr/m68k-atari-mint/bin/ld Patchlevel 40 (AtariST) for GNU

installiert auf TosFS

i:\ > gcc -v test.c
   Reading specs from g:/gnu/lib/specs
   gcc version 2.8.1
    g:/gnu/bin/cpp.ttp -lang-c -v -iprefix g:/gnu/bin/m68k-atari-mint/2.8.1/
      -undef -D__GNUC__=2 -D__GNUC_MINOR__=8 -Dmc68000 -Datarist -Dgem
      -D__mc68000__ -D__atarist__ -D__gem__ -D__mc68000 -D__atarist -D__gem
      -Asystem(tos) -Asystem(gem) -Acpu(m68k) -Amachine(m68k) test.c
      /dev/p/cc320000.i
   GNU CPP version 2.8.1 (68k, MIT syntax)
   #include "..." search starts here:
   #include <...> search starts here:
    g:/gnu/include
   End of search list.
    g:/gnu/bin/cc1.ttp /dev/p/cc320000.i -quiet -dumpbase test.c -version -o
      /dev/p/cc320000.s
   GNU C version 2.8.1 (m68k-atari-mint) compiled by GNU C version 2.8.1.
    g:/gnu/bin/as.ttp -v -mc68000 -o /dev/p/cc3200001.o /dev/p/cc320000.s
```

```
   GNU assembler version 2.9 (m68k-atari-mint)
    g:/gnu/bin/ld.ttp -v g:/gnu/lib/crt0.o -Lg:/gnu/bin -Lg:/gnu/lib
      /dev/p/cc3200001.o -lgnu
   g:/gnu/bin/ld.ttp Patchlevel 40 (AtariST) for GNU (Oct 21 1997 13:35:50)

In beiden Fllen erkennt man, wie nacheinander Prprozessor, Compiler,
Assembler und Linker gestartet werden.
Dabei entsteht ein Programm mit dem Namen 'a.out'. Dies ist der Default-
name, der vom Linker erzeugt wird, wenn kann anderer angegeben wurde.
Um z.B. `test.tos` zu erzeugen, mu man den Aufruf entsprechend erweitern:

   gcc -v test.c -o test.tos
```

1.16 installation, archive

```
Alle Archive auf einen Blick                                    GNU C/C++

Hier eine bersicht ber die von mir zusammengestellten Archive:

   Name           Inhalt

   gcc-281c.lzh   Treiber, Prprozessor und Compiler fr C.

   g++-281c.lzh   Treiber und Compiler fr C++.

   gas-291.lzh    Der Assembler.

   util-41.lzh    Utilities wie Assembler, Linker usw.

   mintlib49.lzh  C Bibliothek (MiNT-Lib)

   gemlib37.lzh   GEM-Bibiliothek.

   mlib23.lzh     Mathe-Bibliothek.

   lib++272.lzh   C++ Bibliothek (von Frank Naumann bersetzt).

   doku-st.lzh    Dieser Hypertext und einige andere Dokus zum
                  Thema 'GNU C auf dem Atari'.

Weitere ntzliche Programme, die nicht zu diesem Paket gehren:

   gdb36p4b.zoo   GNU Debugger.
   make3761.zoo   GNU Make.
```

1.17 tips+tricks

```
                Tips & Tricks zu GNU C                                  ↩
                   GNU C/C++


                GNU-Programme und ihre Stacksize
```

Die Option -pipe des gcc.ttp

Debuggen mit dem gdb

1.18 stacksize

GNU-Programme und die Stacksize GNU C/C++

Sehr viele GNU-Programme, nicht nur der GNU C Compiler, nutzen intensiv eine besondere Speicherverwaltung. Normalerweise forder man Speicher mit dem C-Befehl malloc() an. Der Nachteil dieser Methode ist der, da man so angeforderten Speicher explizit mit free() wieder freigeben mu. Viel praktischer ist es, wenn man in einer Prozedur lokal Speicher anfordern knnte, der dann, beim Verlassen der Prozedur automatisch wieder freigegeben wird. Im GNU-Compiler gibt es die Funktion alloca(), die genau diese Funktionalitt zur Verfgung stellt.
Das Problem bei dieser methode: der Speicher, der lokal angefordert wird, mu irgendwo herkommen - nmlich vom Stack. Unter UNIX hat jedes Programm einen quasi unendlich groen Stack, da man dort virtuellen Speicher hat. Auf dem Atari mu einem Programm aber der Stack explizit zugewiesen werden. Sollte die als Stack zugewiesene Speichermenge zu klein sein, strzen die meisten Programme einfach mit Buserror ab.
Die MiNT-Lib bietet nun die Mglichkeit, von auen, nachtrglich diese Stacksize einzustellen. Dazu gibt es in den Programmen die globalbe Variable _stksize, deren Wert normalerweise 0 enthlt. Bei Programmen, die von vornherein einen groen Stack bentigen (z.B. der Compiler), kann der Wert entweder direkt im Quelltext gesetzt oder nachtrglich mit dem Programm eingestellt werden.
Wem also mal ein GNU-Programm pltzlich mit Buserror abstrtzt, sollte als aller erstes mal die Stacksize etwas erhhen.
Die aktuell eingestellte Gre kann man sich mit dem Programm printstk.ttp anzeigen lassen.

Da der Speicher, den man als Stacksize anmeldet, beim Start des Programms sofort belegt wird, obwohl das Programm gar nicht soviel bentigt (z.B. Stacksize des cc1.ttp 4MB -> die 4MB sind beim Start von cc1.ttp weg!), haben sich die Programmierer der MiNT-Lib einige Spezialflle ausgedacht. Normalerweise enthlt _stksize einen Wert, der die Gre des Stacks in Bytes angibt. Folgende Werte haben eine besondere Bedeutung:

-1 Benutze smtlichen freien Speicher als Stack. Dieser Schalter ist auf (Multitasking-)Systemen mit viel RAM sehr unpraktisch, da dann stmlicher RAM von einem Programm belegt wird.

0 Minimale Stackgre (8k), Standardeinstellung.

1 Gibt ein viertel des freien RAMs als Stack an das Programm, die restlichen 3/4 bleiben frei.

2 Gibt die Hlfte des freien RAMs als Stack.

3 Gibt drei viertel des RAMs als Stack und lt dem System

noch ein viertel.

Bei dem von mir verteilten Compiler ist fr cpp.ttp, cc1.ttp und cc1plus.ttp der Wert 1 eingestellt, da bei bir 1/4 des RAMs normalerweise ausreicht.
Fr Leute mit wenig RAM (z.B. 4MB insgesammt), sollte fr die Compiler _stksize auf -1 gesetzt werden, damit sie allen verfgbaren Speicher bekommen.

Achtung:
Das zur Einstellung der Stacksize bentigte Symbol _stksize kann durch das Programm strip.ttp gelscht werden, so da danach die nderung der Stacksize nicht mehr mglich ist!
Wenn also selbstbersetzte Programme mit strip.ttp verkleinert werden sollen, darf auf keinen Fall vollstndig ge'stripped werden!

1.19 -pipe

Die Option -pipe des gcc.ttp GNU C/C++

Normalerweise startet gcc.ttp Prprozessor, Compiler, Assembler und Linker nacheinander. Die bei jeder Schritt erzeugte Datei wird im Temp-Verzeichnis zwischengelagert und dem nchsten Programm bergeben.
Wird die Option '-pipe' beim Aufruf des gcc.ttp gesetzt, werden alle ntigen Programme gleichzeitig gestartet und das Ergebnis des einen wird ber eine Pipe zum nchsten gereicht.
Die Option ist auch auf dem Atari verfgbar, es gibt aber ein paar Dinge zu beachten:

1. Sie funktioniert nur unter MiNT, da den Pipes in MagiC dafr scheinbar Funktionalitt fehlt. Es mag auch sein, da es an der Mupfel liegt, mit der ich es unter MagiC probiert habe.

2. Wird die Option unter MiNT benutzt, mu man daran denken, da alle Programme gleichzeitig im Speicher liegen und damit auch gleichzeitig ihre fordern. Sollte eines der Programme smtlichen Speicher belegen (Stacksize -1), bleibt natrlich nichts mehr fr die anderen brig.

Wenn man diese beiden Dinge beachtet, kann man die Option unter MiNT einsetzen. Unter MagiC konnte ich die Option noch nicht erfolgreich einsetzen. Entwerder lieg das an der als Kommandoshell verwendeten Mupfel oder aber an nicht korrekt funktionierenden Pipes in MagiC.

Worin der eigentliche Sinn besteht, alle Programme gleichzeitig laufen zu lassen, wei ich jedenfalls nicht. Subjektiv erscheint einem der Compilerlauf schneller, was aber ja eigentlich nicht sein kann, da zwar alle Programme gleichzeitg laufen, aber jedes ja nur etwa ein viertel der Rechenzeit bekommt.

1.20 debugger

Debugging mit dem gdb

Ja, auch das gibts fr Atari: den GNU Debugger.
Die letzte Version, die portiert wurde, ist zwar schon etwas lter,
lt sich aber trotzdem gut zum Debugging einsetzen.
Den gdb habe ich aber bisher nur unter MiNT zum Laufen bekommen. Dort
luft er aber sehr gut, unter N.AES kann man sogar erfolgreich GEM-
Programme im Single-Step Mode entwanzen.

Um den Debugger benutzen zu knnen, mu vom Compiler zum einen das
Programm, zum anderen eine erweiterte Symbolliste erstellt werden,
damit der Debugger auf Variablen und Quelltext zurckgreifen kann.
Da diese Symbolliste nicht, wie unter UNIX blich innerhalb des
ausfhrbaren Programms abgelegt werden kann, gibt es einen speziellen
Symbol-Linker (sym-ld.ttp), der eine extra Datei mit den Symbolen
erzeugt.
Die Erzeugung einer debug-fhige Programm-Version luft also ber
drei Stufen:

1. gcc -g test.c -o test.o
   Programmdatei mit Debugcode erzeugen

2. gcc test.o -o test.tos
   Ausfhrbares Programm linken

3. gcc test.o -o test.sym -B/X/gnu/bin/sym-
   Symboldatei erzeugen. Dafr mu bei der Option -B der
   vollstndige Pfad zum Symbollinker inklusive der Zeichen
   'sym-' angegeben werden. Der gcc versucht nun, den normalen
   Linker ('ld') zu starten, fgt vor 'ld' aber den angegebenen
   Prfix ein, wobei dann ein gltiger Pfad auf den Symbollinker
   entsteht und gcc startet selbigen.

Die beiden entstandenen Dateien knnen dann mit

   gdb test.tos test.sym

dem Debugger bergeben werden.

Zur Bediehnung des gdb verweise ich auf die beraus informative
Quick Reference Card des gdb.

1.21 nderungen

nderungen

   Neu in Version 2.8.1c

      ù Neues Doku-Kapitel zur Installation auf VFAT/MagiC.

      ù Bibliotheken heien jetzt "xxx.a".
        Falls $GNULIB nicht gesetzt ist (Minix), wird ein 'lib'
        vorangestellt, also 'libxxx.a'.
        Die Namen unterscheiden sich jetzt nur noch durch das 'lib',

die Endung 'olb' wird nicht mehr beachtet!!!
Zwei Libs wurden umbenannt:
gnu.* -> c.*
pml.* -> m.*

- ù Linker bekommt immer nur den Basisnamen der Lib (-lgem) und zustzlich die Flags -H fr mshort und -n fr mbaserel. Er baut dann den richtigen Lib-Namen zusammen (libgem.a, libbgem.a, libgem16.a).
  Fr mshort mssen also keine extra Libs mehr angegeben werden!

- ù Die Versionsnummer des Compilers wurde durch einen Buchstaben ergnzt, damit man die verschiedenen Atari-Versionen des 2.8.1 gcc unterscheiden kann.

Neu in Version 2.8.1b

- ù Option -mint entfernt, da es nur noch eine Lib (MiNT-Lib) gibt

- ù In TMPDIR sollten Laufwerkspfad (z.B. p:\) wieder funktionieren

- ù Alles mit neuer Lib gelinkt, damit die Text-Ausgaben richtig erscheinen (MagiC)

Neu in Version 2.8.1

- ù Bei '-g' wird nicht mehr automatisch der Symbollinker gestartet. Es hat sich doch als unpraktisch erwiesen, da die Symboldateien automatisch erzeugt wurden, da z.B. bei normalen GNU-Makefiles '-g' immer gesetzt ist.
  Wer Debugcode braucht, sollte eigentlich so fit sein, das Makefile um den entsprechenden Aufruf zu erweitern :-)
- ù Ab Version 2.8.0 erzeugt der Compiler Assemblercode, den der gute alte gas 1.38 nicht mehr verarbeiten kann. Durch geschickte 'Manipulation' gelang es, den aktuelle GNU Assembler gas 2.9 fr MiNT nutzbar zu machen. Es wurde aber ledeglich der Assembler portiert, alle anderen Bestandteile der GNU binutils (ld, ar usw.) sind nicht portiert worden, so da die alten Utils PL40 weiterhin verwendet werden.
- ù gcc 2.8.1 sollte 'weak symbols' untersttzten (nicht getestet).

Version 2.7.2.3
Erste ffentliche Version.