

GNUShell für den Atari

Version 2.2

Roland Schäuble
czsro@zcvx00.decnnet.ascom.ch

7. August 1994

Inhaltsverzeichnis

1	Einleitung	2
2	GNUShell für Schnelleinsteiger	2
3	Der Menuebalken	2
3.1	Das Menue <i>GNUShell</i>	3
3.2	Das Menue <i>File</i>	3
3.3	Das Menue <i>Compile</i>	3
3.4	Das Menue <i>Projekt</i>	4
3.5	Das Menue <i>Setup</i>	8
3.6	Das Menue <i>Utility</i>	10
4	Das Info-File	10
5	Logfiles	11
6	Fehlermeldungen	11
A	Sourcefiles	13
B	Vertriebsbedingungen	14
C	Registrierung	15

1 Einleitung

Der GNU C-Compiler von der Free Software Foundation erfreut sich zunehmender Beliebtheit, sowohl im professionellen, wie auch im privaten Bereich. Er unterstützt praktisch alle C-Dialekte von K&R C [K&R] über ANSI-C [STEELE] bis hin zu C++ [STROU]. GNU C ist wohl auf alle möglichen Plattformen portiert worden. Gerade aber auch dieser Grad von Universalität macht diesen Compiler nicht leicht im Gebrauch und Installation [POHLERS]. Es müssen einige Environment-Variablen gesetzt werden und der Aufruf erfolgt mit einer Kommandozeile. Sind komplexere Projekte zu bearbeiten, ist man auf das GNU Make [GNUMAKE] angewiesen. Auf dem Atari ist man sich aber an eine komfortablere Benutzeroberfläche gewohnt. Aus diesem Grund wurde die GNUShell entwickelt. GNUShell bietet eine graphische Oberfläche zur Bedienung des Compilers sowie die Möglichkeit, in Projektfiles Abhängigkeiten zwischen den einzelnen Teilen eines Projektes zu definieren.

2 GNUShell für Schnelleinsteiger

Starten Sie GNUShell auf. Stellen Sie unter **Setup** die Environment-Variablen auf die zu Ihrem System passenden Werte ein. Stellen Sie unter **Editor...** Ihren bevorzugten Editor ein. Können dem Editor mehrere Files beim Aufruf übergeben werden, so stellen Sie den Eintrag **parameters to edit sourcefiles** auf **%s %l** ein, sonst nur auf **%s**. Suchen Sie unter **Compiler...** das GNU Compiler-Steuerfile. Es ist meistens unter **...\GNU\BIN\GCC.TTP** zu finden. Stellen Sie die **Compile Options...** auf leer ein. Stellen Sie unter **Link Options...** die Zeile auf **-lGNU.OLB** ein. Wählen Sie das mitgelieferte Programm **HELLO.C** unter **Compile & Link...** aus. Das Programm sollte fehlerfrei übersetzt und gelinkt werden. Lassen Sie das Programm **TEST.PRG** mit **Execute...** laufen. Irgendwo auf dem Bildschirm sollte der Text *Hello world* erscheinen. Wenn alles geklappt hat, speichern Sie die aktuellen Einstellungen unter **Save GNUSHELL.INF** ab.

3 Der Menuebalken

Im Menuebalken finden wir die Einträge **GNUShell**, **File**, **Compile**, **Project**, **Setup** und **Utility**, die in den nachfolgenden Abschnitten erläutert werden.

3.1 Das Menue *GNUShell*

Hier finden sich wie gewohnt die Einträge **About GNUShell** sowie alle installierten Accessories. Im **About GNUShell** finden sich Informationen über die Versionsnummer, das Datum der Compilation und mit welchem Compiler die vorliegende Version übersetzt wurde, sowie meine Adresse und eMail-Adresse..

3.2 Das Menue *File*

Zunächst einmal gibt es hier Einträge zum Editieren verschiedener Files. Der oberste Eintrag **Edit XXXXXXXX.XXX** enthält den aktuellen Filenamen, d.h. der Name des zuletzt editierten Files. Wenn kein Name bekannt ist (z.B. nach dem ersten Aufstarten ohne ein Info-File), ist dieser Eintrag inaktiv. Die Einträge **Edit .C** und **Edit .H** haben die Wildcards ***.C*** bzw. ***.H***, damit sowohl normale C- und H-Files (Extention **.c** bzw. **.h**), wie auch C++ und H++ Files (Extention **.cxx** oder **.cc** bzw. **.h** oder **.hxx**) gefunden werden.

Unter **Edit .RSC** lässt sich der im Menu *Setup* installierte Resource-Editor (Resource Construction Set) aufrufen. Es erscheint eine Fileselektorbox, in der das zu editierende Resourcefile (Extention **.RSC** ausgewählt werden kann.

Weiter befindet sich hier der Eintrag **Execute...**, mit dem ein ausführbares File gestartet werden kann. Hat das File die Extention **.TTP**, so können in einer Dialogbox die zu übergebenden Parameter eingegeben werden. Der aktuelle Pfad wird vor dem Programmstart auf den Pfad des Programmes selbst gesetzt. Damit kann ein GEM-Programm sein Resourcefile finden. Weitergehende Massnahmen (z.B. Umleiten des Outputs auf ein Fenster etc.) finden jedoch nicht statt.

Mit Hilfe des Eintrags **Delete...** kann ein nicht mehr benötigtes File gelöscht werden. Vor dem Löschen findet eine Sicherheitsabfrage statt.

Quit dient zum verlassen der GNUShell. Die aktuellen Einstellungen werden dabei *nicht* gesichert.

3.3 Das Menue *Compile*

Dieser Menüpunkt dient zum compilieren und Linken *einzelner* Files. Der erste Eintrag **Compile XXXXXXXX.XXX** compiliert das aktuelle (d.h. das zuletzt compilierte) File. Ist kein Filename bekannt, so ist dieser Eintrag inaktiv. Mit dem Eintrag **Compile...** kann über eine Fileselektorbox das zu compilierende File ausgewählt werden. Gleichzei-

tig wird es zum aktuellen File für die Compilation gesetzt. Bei der Compilation werden die unter dem Eintrag **Compile Options...** gesetzten Optionen verwendet. Der Eintrag **Compile & Link...** macht die Compilation und linkt gemäss den Angaben, die unter **Link Options..** gesetzt wurden.

Die Angaben, die unter **Compile Options...** bzw **Link Options..** gesetzt wurden, sind nur innerhalb des Menus **Compile** wirksam, nicht jedoch beim Zusammenbau eines Projektes. Noch eine Bemerkung zu den Filenamen: Der GNU C-Compiler interpretiert den Filenamen **xxxxx.c** als C-Sourcefile, **xxxxx.C** hingegen als C++ Sourcefile (wegen des gross geschriebenen „C“). Für die GNUShell gilt folgende Regelung: Filenamen von Sorcefiles werden grundsätzlich kleingeschrieben übergeben. Für die Wahl der Extention von C++ Files empfehle ich **.CXX** bzw. **.HXX**.

3.4 Das Menue *Projekt*

GNUShell kann Projektfiles verwalten, die denjenigen von Pure C sehr ähnlich sind. Aus diesem Grunde wurde für die Projektfiles von GNUShell auch die gleiche Extention nämlich **.PRJ** gewählt. Die Projektfiles beschreiben, aus welchen Sourcefiles ein Projekt zusammengesetzt ist und welche Abhängigkeiten zwischen den einzelnen Files bestehen. Ein Projektfile ist wie folgt aufgebaut:

```

Projektfile ::= Target +
              Optionen +
              '=' +
              { Source }

Target ::= 'Targetfilename'

Optionen ::= (CompilerOption) +
             (AssemblerOption) +
             (LinkerOption)

CompilerOption ::= '.C' + '[' + { GnuOption } + ']'

AssemblerOption ::= '.S' + '[' + { GnuOption } + ']'

LinkerOption ::= '.L' + '[' + { GnuOption } + ']'

GNUOption ::= '-' + TEXT

Source ::= SourceFilename + (Abhängigkeit) + SourceOption

SourceFilename ::= 'Filename'

Abhängigkeit ::= '(' + 'Filename' + { ', ' Filename } + ')'

SourceOption ::= '[' + { GNUOption } + ']'

```

Dies war jetzt eine sehr formale Beschreibung eines Projektfiles. Vielleicht kann man es besser an einem einfachen Beispiel erläutern. Schauen wir uns einmal das folgende Projektfile an:

```

#=====
#Test-Projektfile
#=====
TEST.TTP
.C [-c -ansi -O2]
=
TEST.C
GRAPHIK.C [-Wall]
INOUT.C (INOUT.H)
DRIVER.S
GNU.OLB

```

Mit diesem Projektfile wird folgendes festgelegt: Das Zielfile heisst `TEST.TTP` und wird aus den Sourcefiles `TEST.C`, `GRAPHIK.C`, `INOUT.C` sowie `DRIVER.S` zusammengebaut. Alle C-Sourcefiles werden mit den Optionen `-c` (nur compilieren) `-ansi` (ANSI Standard) und `-O2` (volle Optimierung) compiliert. Zusätzlich wird das File `GRAPHIK.C` mit der Option `-Wall` (alle Warnungen) übersetzt. Das File `GRAPHIK.C` ist vom Headerfile `INOUT.H` abhängig. Beim Linken wird das Libraryfile `GNU.OLB` dazugelinkt. Sind noch keine Objektfiles vorhanden, so werden durch das vorliegende Beispiel folgende Kommandos erzeugt:

```

GCC -c -ansi -O2 test.c
GCC -c -ansi -O2 -W graphik.c
GCC -c -ansi -O2 inout.c
GCC driver.s
GCC test.o graphik.o inout.o driver.o -lgnu.olb

```

Kommentare in Projektfiles können mit folgenden Zeichen eingeleitet werden: `,#'` und `,;'`. Ein Kommentar geht immer bis ans Ende der Zeile.

Wird mit Projektfiles gearbeitet, so haben die Einstellungen unter `Compile Options...` bzw. `Link Options...` *keinen* Einfluss auf den Compilations- bzw. Linkvorgang. Beim Arbeiten mit Projektfiles sollen deshalb alle Optionen im Projektfile festgelegt werden. Damit ist auch garantiert, dass ein Projekt immer unter den gleichen Voraussetzungen zusammengebaut wird.

Mit `Select...` kann das Projektfile gewählt werden, mit dem gearbeitet werden soll. Die Auswahl erfolgt über eine Fileselektorbox.

`Make xxxxx.PRJ` startet die Übersetzung. Dabei gelten folgende Regeln:

- C-Sourcefiles sind Files mit der Extention `.C`, `.CC` und `.CXX`.

- Assembler-Sourcefiles sind Files mit der Extension `.S`.
- Objektfiles sind Files, mit der Extension `.O`.
- Alle anderen Files werden als Libraryfiles angesehen.
- Ein Sourcefile wird übersetzt, wenn es kein Objektfile gleichen Namens mit der Extension `.O` gibt.
- Ein Sourcefile wird übersetzt, wenn es ein neueres Datum aufweist als das zugehörige Objektfile.
- Ein Sourcefile wird übersetzt, wenn mindestens ein File aus der Abhängigkeitsliste des Sourcefiles ein neueres Datum aufweist, als das Objektfile zum Sourcefile.¹
- Der Linkvorgang findet nur statt, wenn bei der Compilation keine Fehler aufgetreten sind.
- Der Linkvorgang findet statt, wenn das Zielfile nicht vorhanden ist.
- Der Linkvorgang findet statt, wenn mindestens ein benötigtes Objektfile ein neueres Datum aufweist als das Zielfile.

Aus obigen Regeln wird klar, dass das korrekte Funktionieren des GNUShell Make von der richtig eingestellten Uhrzeit abhängig ist. Es empfiehlt sich daher die Verwendung einer Hardware-Uhr, da das Einstellen der Uhrzeit nach dem Systemstart doch oft vergessen wird.

In folgenden Fällen werden Fehlermeldungen ausgegeben:

- Syntaxfehler im Projektfile.
- Ein Sourcefile wird nicht gefunden.
- Ein File aus der Abhängigkeitsliste wird nicht gefunden.
- Ein Sourcefile oder ein File aus der Abhängigkeitsliste hat eine neuere Uhrzeit/Datum als die aktuell eingestellte Systemzeit/Datum.

Falls von GNUShell Make keine Aktionen unternommen wurden, erscheint die Meldung *filename.ext already up to date*. Wenn diese Meldung trotz Modifikationen an einem oder

¹Dies funktioniert erst ab GNUShell Version 1.7 korrekt

mehreren Files kommt, soll die Liste der Abhängigkeiten im Projektfile überprüft und gegebenenfalls korrigiert werden.

Eine andere Möglichkeit ist diese: Im Projektfile werden keine Abhängigkeiten definiert; es besteht praktisch nur aus einer Liste seiner Sourcefiles. Damit ist jedes Objektfile nur von seiner Source abhängig. Bei grösseren Modifikationen kann mit der Option **Make all xxxxx.PRJ** eine Vollcompilation ausgelöst werden. Dabei wird jedes File neu übersetzt und das Projekt neu gelinkt. Diese Technik wird von mir sehr häufig praktiziert, weil das Nachführen der Abhängigkeiten doch häufig nicht mit der dafür notwendigen Konsequenz erfolgt. Will man nach einer Änderung erreichen, dass bestimmte Teile neu übersetzt werden, so kann man die zugehörigen Objektfiles vor dem Aufruf des Make einfach löschen.

Mit der Option **Link** wird nur der Linkvorgang gestartet. Dies kann sinnvoll sein, wenn z.B. ein Libraryfile im Projektfile fehlte, was zu einem Linkerfehler führte.

Der Übersetzungsvorgang kann durch (wiederholten) Tastendruck abgebrochen werden. Allerdings erfolgt der Abbruch nicht während der Compilation eines Files sondern immer danach. Wurden während der Compilation Tasten betätigt, so erscheint eine Alertbox mit der Frage *Abort?*. Die Auswahl *Yes* bricht den Make-Vorgang ab, mit *No* wird er fortgesetzt. **Run** startet das Zielfile aus dem aktuellen Projekt. Dieser Eintrag wird erst aktiv, wenn das Projektfile einmal gelesen wurde, d.h. nach einem **Make**, **Make all** oder **Link**.

3.5 Das Menue *Setup*

Der GNU C-Compiler nimmt Bezug auf einige sogenannte Environment-Variablen. Die wichtigsten dabei sind die folgenden:

- **GCC_EXEC_PREFIX** Bezeichnet den Pfad und den Beginn der Filenamen der einzelnen Teile des C-Compilers.
- **GNUINC** Pfad der System-Includefiles für C-Programme.
- **GXXINC** Pfad der System-Includefiles für C++ Programme.
- **GNULIB** Pfad der System-Libraryfiles
- **TMPDIR** Pfad der Temporärfiles.
- **UNIXMODE** Umsetzung von Filenamen.

Für detailliertere Informationen siehe [STALLMAN]. Diese Environment-Variablen können im Setup unter **Environment...** gesetzt werden. Ausserdem hat es noch Platz

für zwei Benutzerdefinierte Environment-Variablen. Wem dies immer noch nicht reicht, der kann direkt im Info-File weitere Environment-Variablen anfügen.

Unter **Editor...** erscheint eine Dialogbox, in der der verwendete Editor mit Hilfe einer Fileselektorbox eingestellt werden kann. Die meisten heute verfügbaren Editoren verfügen über die Möglichkeit, beim Start mehrere Filenamen zu übergeben, die dann in mehreren Fenstern erscheinen. Bei der Arbeit mit GNUShell wäre es daher nützlich, wenn im einen Fenster der Quelltext erscheinen würde, im anderen die vom Compiler erzeugten Fehlermeldungen. Dies kann bei **Parameters to edit sourcefiles** eingestellt werden. **%s** bezeichnet dabei das Sourcefile, **%l** das Logfile mit den Fehlermeldungen, **%n** die Liniennummer des ersten Fehlers. Dazu einige Beispiele. Nehmen wir an, der Editor wurde als **C:\EDIT\XEDIT.PRG** eingestellt und das Sourcefile hiesse **TEST.C**. Der erste Fehler trat auf Linie 17 auf:

Einstellung	Editoraufruf im Fehlerfall
%s	C:\EDIT\XEDIT.PRG TEST.C
%s %l	C:\EDIT\XEDIT\XEDIT.PRG TEST.C TEST.LOG
%l %s %n	C:\EDIT\XEDIT.PRG TEST.LOG TEST.C 17

Die Reihenfolge der Übergabe der Parameter bestimmt meistens, welches File nach dem Aufstarten des Editors im aktuellen Buffer ist. Hier muss je nach verwendetem Editor ein wenig experimentiert werden. Für den von mir verwendeten Editor *xEdit* lautet die Einstellung **%l %s %n**. Der Editor *7UP* kann meines Wissens keine Zeilennummern in der Parameterliste verarbeiten. Die Einstellung für diesen Editor sollte also **%s** oder **%l %s** lauten.

Im **Setup Compiler...** wird das Compiler-Steuerfile über eine Fileselektorbox ausgewählt. In den meisten Fällen dürfte dies das File **...\BIN\GCC.TTP** sein.

Unter dem Menüpunkt **RCS** kann mit Hilfe einer Fileselektorbox ein Resource Construction Set ausgewählt werden.

Manchmal ist es sinnvoll, mit verschiedenen Umgebungen zu arbeiten. Beispielsweise kann der Compilationsvorgang beschleunigt werden, wenn die Temporärfiles auf einer RAM-Disk abgespeichert werden. Andererseits hat aber besonders die Übersetzung von C++ Programmen einen erheblichen RAM-Bedarf, so dass man in diesem Falle lieber mit der Festplatte arbeiten würde. Für solche Fälle dienen die beiden Einträge **Load...** und **Save...**. Nachdem man seine Einstellungen gemacht hat, kann man sie unter einem beliebigen Filenamen mit **Save...** abspeichern, wo sie später wieder mit **Load...** abgerufen werden können. Die Standardeinstellung wird mit **Save GNUSHELL.INF** und **Load GNUSHELL.INF** abgespeichert und geladen. Das File **GNUSHELL.INF** wird auch beim

Programmstart von der GNUShell geladen, beim Verlassen des Programms jedoch *nicht* automatisch abgespeichert. Falls jedoch Änderungen am Setup vorgenommen wurden, erscheint eine Alert-Box mit der Meldung *Setup not saved*. Mit **Save** können die Änderungen unter dem default-Dateinamen abgespeichert werden. Mit **Quit** wird das Programm ohne Abspeicherung des Setups verlassen. Mit **Return** wird das das Programm normal fortgesetzt.

3.6 Das Menue *Utility*

Mit **Show free memory**, kann der zur Verfügung stehende Speicher ermittelt werden. Für die Übersetzung von C-Programmen empfiehlt es sich, mindestens 1.5MB freien Speicher zu haben, für C++ Programme sollten es schon 2MB oder mehr sein. Bei Speicherplatzmangel bringt der Compiler die Fehlermeldung *virtual memory exhausted*. In diesem Falle können speicherplatzintensive Accessories oder RAM-Disks entfernt werden, um den benötigten Platz zu schaffen.

4 Das Info-File

Im Info-File werden alle wichtigen Einstellungen der GNUShell gespeichert. Das Standard-Info-File heisst `GNUSHELL.INF` und soll im gleichen Directory liegen, wie die GNUShell selbst. Alle Einträge sind mit Namen versehen, sodass es auch leicht mit einem Editor bearbeitet werden kann. Kommentare beginnen mit einem `,%` in Kolonne 1 und gehen bis ans Ende der Zeile. Einträge der GNUShell beginnen mit einem `,#` und haben die Form *Name=Wert*. Einträge von Environmentvariablen beginnen mit einem `,&` und haben ebenfalls die Form *Name=Wert*. Es können so viele Einträge von Environmentvariablen gemacht werden, wie `,&` vorhanden sind.

Folgende Einträge sind nebst den Environment-Variablen im InfoFile vorhanden:

Name	Bedeutung
EDIT_FILE	Pfad und Filename für Edit <code>XXXXXXXXX.XXX</code>
SOURCE_FILE	Pfad und Filename für Compile <code>XXXXXXXXX.XXX</code>
PROJECT_FILE	Pfad und Filename für Make <code>XXXXXXXXX.XXX</code>
EDITOR	Pfad und Filename des Editors
COMPILER	Pfad und Filename des Compiler-Steuerfiles
RUN_FILE	Pfad und Filename für Run <code>XXXXXXXXX.XXX</code>
COMPILE_OPTIONS	Zeile aus Compile Options..
LINK_OPTIONS	Zeile aus Link Options...
EXECUTE_OPTIONS	Parameter beim Starten von .TTP-Files
EDITOR_PARAMETER	Parameter beim Starten des Editors mit einem Sourcefile

Beim Einlesen des Info-Files werden nur rudimentäre Kontrollen gemacht, sodass Falscheinträge durchaus zum berüchtigten „Bombenwurf“ von GNUShell führen können. Hier ist also eine gewisse Vorsicht am Platze. Falls GNUShell immer beim Aufstarten abstürzen sollte, kann das File `GNUSHELL.INF` gelöscht oder umbenannt werden. GNUShell stellt dann seine Defaultwerte ein und muss problemlos anlaufen.

5 Logfiles

Alle Meldungen des Compilers werden auf ein sogenanntes Logfile umgeleitet. Das Logfile hat im allgemeinen den gleichen Namen wie das entsprechende Sourcefile, aber die Extension `.LOG`. Es liegt auf dem Directory des entsprechenden Sourcefiles. Link-Fehler werden in ein Logfile geschrieben, das den gleichen Namen hat, wie das Zielfile, aber die Extension `.LOG`. Diese Logfiles liegen auf dem Directory des entsprechenden Zielfiles. Sind während einer Compilation oder während des Linkvorgangs keine Meldungen aufgetreten, so wird das Logfile nach Abschluss wieder gelöscht. Im anderen Fall bleibt es bestehen und es öffnet sich ein Fenster, auf dem der Inhalt des Logfiles gezeigt wird.

6 Fehlermeldungen

- *File xxxxxxxx.xxx not found*
Dieser Fehler kann in folgenden Fällen auftreten:
 - Das Projektfile wurde nicht gefunden
 - Ein Sourcefile wurde nicht gefunden

- Ein Logfile wurde nicht gefunden
- Ein File aus der Abhängigkeitsliste wurde nicht gefunden
- *Syntax Error*
Dieser Fehler tritt auf, wenn im Projektfile ein syntaktischer Fehler gefunden wurde. Die Fehlermeldung hat folgende Form:


```
File:      filename.ext
Line:      nnnnn
Found:     xxxxx
Expected:  yyyyy
```

Der Fehler trat im Projektfile `filename.ext` auf der Linie `nnnnn` auf. Es wurde `xxxxx` gelesen, aber `yyyyy` erwartet. Der eigentliche Fehler muss sich nicht unbedingt auf der angegebenen Linie befinden. Er kann durchaus auch eine oder mehrere Linien davor liegen. Abhilfe: Korrigieren des Projektfiles.
- *File xxxxxxxx.xxx is in the future!*
Dieser Fehler tritt auf, wenn ein File gefunden wurde, dessen Datum und Uhrzeit in der Zukunft liegt. In den meisten Fällen dürfte die aktuelle Systemzeit falsch eingestellt sein oder die Systemzeit zum Zeitpunkt der Erzeugung des bezeichneten Files war falsch eingestellt. Hier sollte die Systemzeit korrekt eingestellt werden und mit `Make all` eine Vollcompilation durchgeführt werden.
- *Fatal: Out of memory*
Dieser Fehler tritt auf, wenn von GNUShell kein freier Speicher mehr alloziert werden kann. GNUShell wird nach Bestätigung dieses Fehlers abgebrochen. Dieser Fehler sollte im normalen Betrieb nie auftreten. Wenn er dennoch auftritt, sollte man das System neu aufstarten. Tritt der Fehler wiederholt auf, so kann versucht werden, durch weglassen von speicherintensiven Accessories oder RAM-Disks etc. mehr freien Speicher zu schaffen.
- *Compiler returns xxx*
Dieser Fehler wird dann ausgegeben, wenn der Compiler einen Returnstatus $\neq 0$ ausgegeben hat, aber gleichzeitig das Logfile leer ist. Dieser Fehler sollte eigentlich normalerweise nicht auftreten. Falls er dennoch auftritt: Environment, Compiler-Setup und Memory prüfen.
- GEMDOS Fehlermeldungen
Diese Fehler sind am schwierigsten zu lokalisieren und können auch mehrfach auftreten. Meistens liegt es an einem falschen Setup. Man kontrolliere den zu der gerade

aufgerufenen Funktion gehörenden Setup. Beispiel: Es wurde versucht ein File zu editieren und es trat der Fehler *Diese Anwendung kann das angegebene Objekt nicht finden* auf. Die Ursache liegt vermutlich in einem falsch eingestellten Pfad zum Editor. Abhilfe: Im **Setup Editor...** den richtigen Pfad einstellen.

A Sourcefiles

GNUShell besteht aus folgenden Sourcefiles:

- **GNUALLOC.C, GNUALLOC.H**
Kontrollierte Speicherplatzallozierung.
- **GNUDEF.H**
Allgemein verwendete Definitionen.
- **GNUENV.C, GNUENV.H**
Behandlung des Environments.
- **GNUERR.C, GNUERR.H**
Ausgabe von Fehlermeldungen.
- **GNUFILE.C, GNUFILE.H**
Definitionen und Funktionen zur Behandlung von File- und Pfadnamen.
- **GNUINFO.C, GNUINFO.H**
Verwaltung der Info-Files.
- **GNULEX.C, GNULEX.**
Lexical Scanner zum Lesen eines Projektfiles.
- **GNULOG.C, GNULOG.H**
Verwaltung der Log-Windows.
- **GNUMAKE.C, GNUMAKE.H**
Verarbeitung der Projektfiles.
- **GNUMENU.C, GNUMENU.H**
Verwaltung der Menues.
- **GNUPARSE.C, GNUPARSE.H**
Parser zum Einlesen der Projektfiles in eine interne Datenstruktur.

- `GNUPATH.C`, `GNUPATH.H`
Verwaltung der verschiedenen Pfade und Filenamen.
- `GNUSHELL.C`
Hauptprogramm der GNUShell.
- `GNUSHELL.H`
Definitionen zu `GNUSHELL.RSC`
- `GNUSHOW.C`, `GNUSHOW.H`
Anzeige der aktuellen Aktivität beim compilieren oder linken.
- `GNUTEXT.C`, `GNUTEXT.H`
Verwaltung von Textfenstern.

B Vertriebsbedingungen

Das Programm, d.h. die Files `GNUSHELL.PRG`, `GNUSHELL.RSC`, `GNUSHELL.INF` und `README.TXT` dürfen in *unveränderter Form* in beliebig vielen Kopien weitergegeben werden.

Bei regelmässiger Benutzung ist die Sharewaregebühr an untenstehende Adresse zu entrichten.

Änderungen am Programm oder an der Dokumentation sind nur zu privaten Zwecken zulässig.

Das Programm darf nicht verkauft werden.

Die Weitergabe über PD-Versand ist gestattet, sofern die Gebühr für eine Diskette DM/sFr 15.- nicht überschreitet.

Die Benutzung des Programmes erfolgt auf eigenes Risiko. Jede Haftung wird abgelehnt.

C Registrierung

GNUShell ist Shareware. Bei regelmässiger Benutzung sind die Anwender verpflichtet, die Sharewaregebühr von DM/sFr 20.- an folgende Adresse zu entrichten:

Roland Schäuble
Dattikonstr. 2
CH-8730 **Uznach**
Tel. (055) 72 41 34
eMail: czsro@zcvx00.decnet.ascom.ch

Durch die Registrierung kommt der Anwender in den Genuss folgender Vorteile:

- Er wird von mir in die Liste der registrierten Anwender aufgenommen.
- Ein registrierter Anwender bekommt die zum Zeitpunkt der Registrierung aktuelle Version von GNUShell von mir zugesandt
- Zudem erhält er alle Sourcefiles und das Projektfile, mit dem GNUShell zusammengebaut werden kann (GNUShell kann sowohl mit dem GNU C-Compiler, wie auch mit Pure C übersetzt werden).
- Registrierte Anwender erhalten eine Kopie der vorliegende Dokumentation zugesandt.
- Ein registrierter Anwender wird über (wesentliche) Neuerungen von GNUShell unterrichtet.
- Ein registrierter Anwender kann jederzeit gegen Einsendung einer Leerdiskette die aktuelle Version von GNUShell bei mir beziehen.
- Bei Fragen und Problemen stehe ich im Rahmen meiner Möglichkeiten zur Verfügung.

Literatur

[STALLMAN] Richard M. Stallman *Using and Porting GNU CC*

[GNUMAKE] Richard M. Stallman and Roland McGrath *GNU Make, A Program for Directing Recompilation*

- [POHLERS] Bjarne Pohlars, ST 68000er Magazin, Ausgaben 8/92 und 9/92 *Keine Angst vorm GNU*
- [K&R] Brian W. Kernighan, Dennis M. Ritchie *The C Programming Language*
Prentice Hall Inc.
ISBN 0-13-110163-3
- [STEELE] Samuel P. Harbison / Guy L. Steele Jr. *C, A Reference Manual*
Prentice Hall Inc.
ISBN 0-13-109810-1
- [STROU] Bjarne Stroustrup *Die Programmiersprache C++*
Addison-Wesley
ISBN 3-89319-386-3