



Computer & Literatur Verlag GmbH

DIE GNU AUTOTOOLS

Florian Stöhr

Deutsche Nationalbibliothek – CIP-Einheitsaufnahme
Bibliografische Information der Deutschen Nationalbibliothek

Ein Titeldatensatz für diese Publikation ist bei
der Deutschen Nationalbibliothek erhältlich und im Internet über
<http://dnb.ddb.de> abrufbar.

Alle Rechte vorbehalten. Ohne ausdrückliche, schriftliche Genehmigung des
Herausgebers ist es nicht gestattet, das Buch oder Teile daraus in irgendeiner
Form durch Fotokopie, Mikrofilm oder ein anderes Verfahren zu vervielfältigen
oder zu verbreiten. Dasselbe gilt für das Recht der öffentlichen Wiedergabe.

Der Verlag macht darauf aufmerksam, daß die genannten Firmen- und
Markenzeichen sowie Produktbezeichnungen in der Regel marken-, patent-,
oder warenzeichenrechtlichem Schutz unterliegen.

Die Herausgeber übernehmen keine Gewähr für die Funktions-
fähigkeit beschriebener Verfahren, Programme oder Schaltungen.

1. Auflage 2007

© 2007 by C&L Computer und Literaturverlag
Zavelsteiner Straße 20, 71034 Böblingen
E-Mail: info@CuL.de
WWW: <http://www.CuL.de>

Coverdesign: Hawa & Nöh, Neu-Eichenberg
Satz: C&L-Verlag
Druck und Bindung: Kössinger AG, Schierling
Printed in Bavaria

Dieses Buch wurde auf chlorfrei gebleichtem Papier gedruckt

ISBN: 978-3-936546-48-4

INHALT

Vorwort
Seite 11

Kapitel 1
Der GNU-Dreisatz
Seite 13

1.1	Motive	13
1.2	Arbeiten mit ./configure, make und make install.....	20
1.2.1	Verzeichnisse und Variablen	22
1.2.2	Parameter und Umgebungsvariablen	24
	Standardoptionen	25
	Optionen für Installationsverzeichnisse	26
	Konfiguration mit parallelen Build-Trees	28
	Die Ergänzungsdatei config.site	29
1.2.3	Zieldefinitionen.....	31
	Crosskompilieren	31
	Programme zum Installationszeitpunkt umbenennen	34
	Zieländerung zum Installationszeitpunkt	35

Kapitel 2
Autoconf und Automake
Seite 37

2.1 Grundlagen	37
2.2 Projektaufbau.....	40

Kapitel 3
Die Basiskomponenten
Seite 53

3.1 autoscan	54
3.2 autoheader	55
3.3 autoupdate	57
3.4 ifnames.....	58
3.5 autoconf	60
3.6 automake.....	62
3.7 autoreconf	68

Kapitel 4
Die Steuerdatei configure.ac
Seite 71

4.1 Makros einsetzen	71
4.2 Die Makrosprache	73
4.3 Headertemplates.....	77
4.4 Cachevariablen	82
4.5 Vorgefertigte Autoconf-Makros	84
4.5.1 Prüfmakros.....	86
4.5.2 Definitionen.....	90

INHALT

4.5.3 Ausgabevariablen.....	91
4.5.4 Nachrichten ausgeben	92
4.5.5 Auf Dateien prüfen	94
4.5.6 Auf Bibliotheken prüfen.....	95
4.5.7 Auf Funktionen prüfen.....	97
4.5.8 Auf Header prüfen.....	107
4.5.9 Auf Deklarationen prüfen	112
4.5.10 Auf Strukturmember prüfen	114
4.5.11 Auf Datentypen prüfen	116
4.5.12 Auf System-Services prüfen	118
4.5.13 Auf optionale Features prüfen.....	119
4.5.14 Eigene Tests	122
Beispiel: pthread_t umwandeln	122
Tests in Makros auslagern	132
4.5.15 Statische Fehlermeldungen	134
4.5.16 Veraltete Makros kennzeichnen.....	136
Endgültige Version des Beispiels.....	137
4.6 Beispiel: Socketfunktionalität	139

Kapitel 5 Die Steuerdatei Makefile.am Seite 163

5.1 Traditionelle Make-Dateien	163
5.1.1 Erweiterungen von GNU make	173
5.2 Automatisch erzeugte Makefiles.....	174
5.2.1 Automake steuern: Variablen.....	175
5.2.2 Primaries.....	177
_PROGRAMS, _LIBRARIES und _LTLIBRARIES	180
_HEADERS	187
_SCRIPTS	188
_DATA.....	189
_TEXINFO.....	189
_MANS.....	190
5.2.3 Generierte Quelldateien.....	192

INHALT

5.2.4 Zusätzliche Variablen	192
5.2.5 Optionen für Automake	193
5.2.6 Bedingtes Einbinden von Quellmodulen	197
5.2.7 Conditionals.....	199
5.2.8 Einklinken in Makefile-Regeln	202

Kapitel 6 Testsuites mit Autotest Seite 213

6.1 Grundlagen.....	213
6.2 Einbetten von Autotest in ein Projekt	227
6.3 Standard-Kommandozeilen-Parameter	242
6.4 Referenz der Autotest-Makros	243

Kapitel 7 Bibliotheken mit Libtool Seite 247

7.1 Konzepte und Strategien	247
7.2 Statisches Linken gegen Libtool-Bibliotheken	258
7.3 Versionierung von Libtool-Bibliotheken	261
7.4 Schnittstellendesign	267
7.5 Kombiniertes Bibliotheks-/Programm-Projekt	270
7.6 Bibliothek und Programm als getrennte Projekte	284
7.7 Sprachabhängiges.....	285
7.7.1 C++-Bibliotheken und das Name Mangling	285
Compilergrenzen überwinden	288
Fallstricke mit Datentypen und Puffern	289
7.8 Dynamische Bibliotheken unter Windows	295

Kapitel 8
Dynamische Module mit LTDL
Seite 307

8.1 Bibliotheken explizit laden	307
8.2 Libtool und dlopen-fähige Module	311
8.2.1 Module finden	313
8.3 Die libltdl-Bibliothek (LTDL)	315
8.3.1 LTDL-Typen und -Funktionen	317
8.3.2 LTDL einbinden	322
8.3.3 LTDL-Module schreiben	333
8.3.4 Buildsystem für LTDL-Module	334
8.4 Komplexes LTDL-Beispiel	349
8.4.1 Der Aufbau der Schnittstellenbibliothek	350
8.4.2 Erstes Modul, ein Treiber für PostgreSQL	360
8.4.3 Zweites Modul, ein Treiber für MySQL	363
8.4.4 Intelligentes Buildsystem (nicht alle Treiber erzeugen)	374
8.4.5 Intelligenteres Buildsystem (Manuelle Treiberwahl)	378
8.4.6 Das Hauptprogramm zu den Treibern	383

Stichwortverzeichnis
Seite 389

Die Listings zum Buch:

Sie finden die Listings zum Buch auf der Verlagswebseite
<http://www.CuL.de> zum Download

INHALT

VORWORT

Je komplexer eine Software ist, desto dringender wird ein gutes Installationsprogramm benötigt. Was unter Windows Programme wie Installshield, Wise Installation Studio, NSIS oder InnoSetup sind, stellen unter Unix und Linux die GNU Autotools dar. Sie wurden in den neunziger Jahren vom GNU-Projekt für seine eigene Softwarekollektion als einheitliches Build- und Installationssystem entworfen und verbreiteten sich schnell auch unter Projekten außerhalb von GNU bis hin zu großen kommerziellen Anwendungen. Sie sind heute der De-facto-Standard für den Build- und Installationsprozeß von Softwarepaketen unter Unix, wobei sie natürlich auch unter Windows verwendet werden können.

Die GNU Autotools bestehen aus einer Reihe von Programmen, die in die drei Gruppen *Autoconf* (Abfragen der Systemumgebung), *Automake* (Steuern des Buildprozesses) und *Libtool* (Unterstützung für Programm-bibliotheken) zusammengefaßt sind.

VORWORT

Dieses Buch für Softwareentwickler zeigt, wie ein Autotools-basiertes Buildsystem richtig aufgesetzt wird und wie eigener Programmcode geschrieben werden muß, der optimal mit den Autotools zusammenarbeiten kann. Obgleich die Autotools einige weitere Programmiersprachen unterstützen, sind die Beispiele in C und C++ gehalten, weil diese Sprachen bei Autotools-Projekten unter Unix traditionell am verbreitetsten sind.

Das Buch beginnt mit einer Beschreibung autotools-basierter Buildsysteme aus Sicht des Anwenders, dem Leser wird der grundsätzliche Aufbau eines Autotools-Buildsystems vorgestellt. Danach werden die einzelnen Programmkomponenten der Autotools-Suite erläutert, in den Folgekapiteln werden die Autoconf- und Automake-Steuerdateien detailliert besprochen und ihr Befehlssatz vorgestellt. Danach wird gezeigt, wie eine Testsuite in Autotools-Projekte integriert wird. Ein weiteres Kapitel widmet sich dynamischen Bibliotheken mit Libtool, darin geht es um dynamisch ladbare Treiberbibliotheken und Module. Neben einfacheren Beispielen werden auch zwei komplexe Beispiele einer Socketbibliothek und Datenbankschnittstelle vorgestellt, die den Einsatz der Autotools in realen Projekten verdeutlichen und einsatzbereite Implementierungsvorschläge zeigen beziehungsweise als Basis für eigene Projekte dienen.

Ich wünsche Ihnen viel Spaß mit den GNU Autotools und natürlich mit diesem Buch!

Florian Stöhr
(*autotools@florian-stoehr.de*)