

Mc
Graw
Hill

OSBORNE

The Complete Reference

Build all open source software
—including Linux, GNOME, KDE,
StarOffice, and the Apache Web Server—
and the applications that run on them

GCC

Develop and deploy
software on any UNIX
platform—including
Linux and BSD

Install and use your
own compiler for C,
C++, Objective C,
Fortran, Java, and Ada

Generate native
executable code for
dozens of different
platforms

Arthur Griffith

Author, computer consultant

GCC: The Complete Reference

For Mary



Professional

Want to learn more?

We hope you enjoy this McGraw-Hill eBook! If you'd like more information about this book, its author, or related books and websites, please [click here](#).

McGraw-Hill/Osborne



A Division of The McGraw-Hill Companies

Copyright © 2002 by The McGraw-Hill Companies, Inc. All rights reserved. Manufactured in the United States of America. Except as permitted under the United States Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the publisher.

0-07-222405-3

All trademarks are trademarks of their respective owners. Rather than put a trademark symbol after every occurrence of a trademarked name, we use names in an editorial fashion only, and to the benefit of the trademark owner, with no intention of infringement of the trademark. Where such designations appear in this book, they have been printed with initial caps.

McGraw-Hill eBooks are available at special quantity discounts to use as premiums and sales promotions, or for use in corporate training programs. For more information, please contact George Hoare, Special Sales, at george_hoare@mcgraw-hill.com or (212) 904-4069.

TERMS OF USE

This is a copyrighted work and The McGraw-Hill Companies, Inc. (“McGraw-Hill”) and its licensors reserve all rights in and to the work. Use of this work is subject to these terms. Except as permitted under the Copyright Act of 1976 and the right to store and retrieve one copy of the work, you may not decompile, disassemble, reverse engineer, reproduce, modify, create derivative works based upon, transmit, distribute, disseminate, sell, publish or sublicense the work or any part of it without McGraw-Hill’s prior consent. You may use the work for your own noncommercial and personal use; any other use of the work is strictly prohibited. Your right to use the work may be terminated if you fail to comply with these terms.

THE WORK IS PROVIDED “AS IS”. MCGRAW-HILL AND ITS LICENSORS MAKE NO GUARANTEES OR WARRANTIES AS TO THE ACCURACY, ADEQUACY OR COMPLETENESS OF OR RESULTS TO BE OBTAINED FROM USING THE WORK, INCLUDING ANY INFORMATION THAT CAN BE ACCESSED THROUGH THE WORK VIA HYPERLINK OR OTHERWISE, AND EXPRESSLY DISCLAIM ANY WARRANTY, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. McGraw-Hill and its licensors do not warrant or guarantee that the functions contained in the work will meet your requirements or that its operation will be uninterrupted or error free. Neither McGraw-Hill nor its licensors shall be liable to you or anyone else for any inaccuracy, error or omission, regardless of cause, in the work or for any damages resulting therefrom. McGraw-Hill has no responsibility for the content of any information accessed through the work. Under no circumstances shall McGraw-Hill and/or its licensors be liable for any indirect, incidental, special, punitive, consequential or similar damages that result from the use of or inability to use the work, even if any of them has been advised of the possibility of such damages. This limitation of liability shall apply to any claim or cause whatsoever whether such claim or cause arises in contract, tort or otherwise.

DOI: 10.1036/0072228164

GCC: The Complete Reference

Arthur Griffith

McGraw-Hill/Osborne

New York Chicago San Francisco
Lisbon London Madrid Mexico City
Milan New Delhi San Juan
Seoul Singapore Sydney Toronto

About the Author

Arthur Griffith has been involved with the development of compilers, interpreters, linkers, and assemblers since his first programming job in 1977, where he worked as a team member developing an assembler and linker for special-purpose computers. He then joined the maintenance group for a compiler of the PL/EXUS language, which had an underlying structure very similar to GCC. The next project was to write an interactive interpreter and compiler for a language named SATS.

The projects that followed these included the development of a Forth interpreter, extensions to a COBOL compiler, and the development of some special-purpose interpretive languages for machine control. One of these was an interactive command language providing multistation ground-based control of industrial satellite communications systems.

For the past few years, Arthur Griffith has turned to writing computer books, teaching programming online, and developing some software in Java. The programming books he has written range from *Java*, *XML*, and *Jaxp* to *COBOL for Dummies*. He has used GCC for many software-development projects, and with the inclusion of Java as one of the GCC languages, writing this book became his project of choice.

Contents at a Glance

Part I

The Free Software Compiler

- | | | | |
|---|----------|---|----|
| ■ | 1 | Introduction to GCC | 3 |
| ■ | 2 | Acquiring and Installing the Compiler | 17 |

Part II

Using the Compiler Collection

- | | | | |
|---|----------|-----------------------------|-----|
| ■ | 3 | The Preprocessor | 45 |
| ■ | 4 | Compiling C | 67 |
| ■ | 5 | Compiling C++ | 103 |
| ■ | 6 | Compiling Objective-C | 125 |
| ■ | 7 | Compiling Fortran | 137 |
| ■ | 8 | Compiling Java | 157 |

■ ■ ■	9	Compiling Ada	183
■ ■ ■	10	Mixing Languages	215
■ ■ ■	11	Internationalization	243

Part III

Peripherals and Internals

■ ■ ■	12	Linking and Libraries	259
■ ■ ■	13	Using the GNU Debugger	281
■ ■ ■	14	Make and Autoconf	299
■ ■ ■	15	The GNU Assembler	317
■ ■ ■	16	Cross Compiling and the Windows Ports	337
■ ■ ■	17	Embedded Systems	347
■ ■ ■	18	Output from the Compiler	357
■ ■ ■	19	Implementing a Language	371
■ ■ ■	20	Register Transfer Language	387
■ ■ ■	21	Machine-Specific Compiler Options	419

Part IV

Appendices

■ ■ ■	A	GNU General Public License	493
■ ■ ■	B	Environment Variables	501
■ ■ ■	C	Command-Line Cross Reference	505
■ ■ ■	D	Command-Line Options	515
■ ■ ■	E	Glossary	599

Contents

Acknowledgments	xix
Introduction	xxi

Part I

The Free Software Compiler

1	Introduction to GCC	3
	GNU	4
	Measuring a Compiler	4
	Command-Line Options	5
	Platforms	6
	What the Compiler Does	7
	The Languages	8
	C Is the Fundamental Language	9
	C++ Was the First Addition	9
	Objective-C	9
	Fortran	9
	Java	10



Ada	10
The Chill Is Gone	10
Parts List	11
Contact	15



2 Acquiring and Installing the Compiler	17
Binary Download	18
FTP Source Download	20
CVS Source Download	21
Previous Releases	23
The Experimental Version	23
Compiling and Installing GCC	24
Installation Procedure	24
Configuration Options	26
The binutils	36
Win32 Binary Installation	38
Cygwin	38
Installation	39
Running the Test Suite	40

Part II

Using the Compiler Collection

3 The Preprocessor	45
Directives	46
#define	46
#error and #warning	50
#if, #elif, #else, and #endif	51
#ifdef, #else, and #endif	52
#include	53
#include_next	54
#line	55
#pragma and _Pragma	56
#undef	57
##	57
Predefined Macros	58
Including a Header File Only Once	62
Including Location Information in Error Messages	62
Removing Source Code in Place	63
Producing Makefiles	63
Command-Line Options and Environment Variables	64

 4	Compiling C	67
	Fundamental Compiling	68
	Single Source to Executable	69
	Source File to Object File	70
	Multiple Source Files to Executable	70
	Preprocessing	71
	Generating Assembly Language	71
	Creating a Static Library	71
	Creating a Shared Library	73
	Overriding the Naming Convention	75
	Standards	75
	C Language Extensions	76
	Alignment	76
	Anonymous Unions	77
	Arrays of Variable Length	78
	Arrays of Zero Length	78
	Attributes	80
	Compound Statements Returning a Value	86
	Conditional Operand Omission	88
	Enum Incomplete Types	88
	Function Argument Construction	88
	Function Inlining	90
	Function Name	91
	Function Nesting	91
	Function Prototypes	93
	Function Return Addresses and Stack Frames	93
	Identifiers	94
	Integers	94
	Keyword Alternates	94
	Label Addresses	95
	Labels Declared Locally	96
	Lvalue Expressions	96
	Macros with Variable Arguments	97
	Strings	98
	Pointer Arithmetic	98
	Switch/Case	99
	Typedef Name Creation	99
	Typeof References	100
	Union Casting	101
 5	Compiling C++	103
	Fundamental Compiling	104
	Single Source File to Executable	104

	Multiple Source Files to Executable	106
	Source File to Object File	107
	Preprocessing	107
	Generating Assembly Language	108
	Creating a Static Library	108
	Creating a Shared Library	110
	Extensions to the C++ Language	113
	Attributes	113
	Header Files	114
	Function Name	114
	Interface and Implementation	115
	Operators <? and >?	116
	Restrict	117
	Compiler Operation	118
	Libraries	118
	Mangling Names	119
	Linkage	122
	Compiling Template Instantiations	123
	6 Compiling Objective-C	125
	Fundamental Compiling	126
	Single Source to Executable	126
	Compiling an Object	127
	Creating a Static Library	129
	Creating a Shared Library	132
	General Objective-C Notes	133
	Predefined Types	133
	Creating an Interface Declaration	133
	Naming and Mangling	135
	7 Compiling Fortran	137
	Fundamental Compiling	138
	Single Source to Executable	138
	Multiple Source Files to Executable	140
	Generating Assembly Language	140
	Preprocessing	141
	Creating a Static Library	142
	Creating a Shared Library	144
	Ratfor	144
	GNU Fortran Extensions and Variations	146
	Intrinsics	146
	Source Code Form	146
	Comments	147

	Dollar Signs	147
	Case Sensitivity	147
	Specific Fortran 90 Features	150
8	Compiling Java	157
	Fundamental Compiling	158
	Single Source to Binary Executable	158
	Single Source to Class File	159
	Single Source to Binary Object File	160
	Class File to Native Executable	160
	Multiple Source Files to Binary Executable	161
	Multiple Input Files to Executables	162
	Generating Assembly Language	163
	Creating a Static Library	164
	Creating a Shared Library	165
	Creating a Jar File	166
	The Java Utilities	166
	gij	166
	jar	168
	gcjh	170
	jcf-dump	172
	jv-scan	173
	jv-convert	174
	grepjar	176
	RMI	177
	rmic	177
	rmiregistry	179
	Properties	180
9	Compiling Ada	183
	Installation	184
	Fundamental Compiling	186
	Single Source to Executable	187
	Multiple Source to Executable	189
	Source to Assembly Language	190
	Options	191
	Ada Utilities	197
	gnatbind	197
	gnatlink	200
	gnatmake	201
	gnatchop	205
	gnatxref	205
	gnatfind	207

	gnatkr	208
	gnatprep	209
	gnatls	211
	gnatpsys and gnatpsta	211
10	Mixing Languages	215
	Mixing C++ and C	216
	Calling C from C++	216
	Calling C++ from C	218
	Mixing Objective-C and C	218
	Calling C from Objective-C	219
	Calling Objective-C from C	219
	Mixing Java and C++	221
	Creating a Java String and Calling a Static Method	222
	Loading and Instantiating a Java Class	223
	Exceptions	226
	Data Types of JNI	226
	Mixing Java and C	227
	A Java Class with a Native Method	227
	Passing Arguments to Native Methods	230
	Calling Java Class Methods from C	231
	Mixing Fortran and C	233
	Calling C from Fortran	234
	Calling Fortran from C	235
	Mixing Ada and C	237
	Calling C from Ada	237
	Calling C from Ada with Arguments	239
11	Internationalization	243
	A Translatable Example	244
	Creating a New .po File	246
	Use of the gettext() Functions	250
	Static Strings	250
	Translation from Another Domain	251
	Translation from Another Domain in a Specified Category	251
	Plurality	251
	Plurality from Another Domain	252
	Plurality from Another Domain Within a Category	252
	Merging Two .po Files	252
	Producing a Binary .mo File from a .po File	254

Part III

Peripherals and Internals

12	Linking and Libraries	259
	Object Files and Libraries	260
	Object Files in a Directory	260
	Object Files in a Static Library	261
	Object Files in a Dynamic Library	264
	A Front End for the Linker	264
	Locating the Libraries	265
	Locating Libraries at Link Time	265
	Locating Libraries at Runtime	266
	Loading Functions from a Shared Library	266
	Utility Programs to Use with Object Files and Libraries	269
	Configuring the Search for Shared Libraries	269
	Listing Symbols Names in Object Files	271
	Removing Unused Information from Object Files	274
	Listing Shared Library Dependencies	276
	Displaying the Internals of an Object File	277
13	Using the GNU Debugger	281
	Debugging Information Formats	282
	STABS	282
	DWARF	283
	COFF	283
	XCOFF	284
	Compiling a Program for Debugging	284
	Loading a Program into the Debugger	287
	Performing a Postmortem	291
	Attaching the Debugger to a Running Program	292
	Command Summary	295
14	Make and Autoconf	299
	Make	300
	Internal Definitions	302
	How to Write a Makefile	304
	The Options of Make	305
	Autoconf	310
15	The GNU Assembler	317
	Assembling from the Command Line	318
	Absolute, Relative, and Boundaries	320

Inline Assembly	322
The asm Construct	322
Assembler Directives	325
16 Cross Compiling and the Windows Ports	337
The Target Machines	338
Creating a Cross Compiler	339
Installing a Native Compiler	339
Building binutils for the Target	340
Installing Files from the Target Machine	341
The Configurable Library libgcc1.a	341
Building the Cross Compiler	342
Running the Cross Compiler	343
MinGW	343
Cygwin	344
Compiling a Simple Cygwin Console Program	344
Compiling a Cygwin GUI Program	345
17 Embedded Systems	347
Setting Up the Compiler and Linker	348
Choosing a Language	349
GCC Embedding Facilities	350
Command-Line Options	350
Diagnostics	351
Assembler Code	351
Libraries	352
Trimming the Standard Library	352
A Library Designed for Embedded Systems	353
The GNU Linker Scripting Language	353
Script Example 1	354
Script Example 2	355
Some Other Script Commands	356
18 Output from the Compiler	357
Information about Your Program	358
The Parse Tree	358
Header Files	359
The Memory Required by the Program	360
Time Consumed	361
The C++ Intermediate Tree	362
The C++ Class Hierarchy	363
Information for the Makefile	363

	Information about the Compiler	365
	Time to Compile	365
	Subprocess Switches	366
	Verbose Compiler Debugging Information	366
	Information about Files and Directories	370
19	Implementing a Language	371
	From Front to Back	372
	Lexical Scan	373
	A Simple Lex	374
	Lex with Regular Expressions	374
	Parsing	375
	Creating the Parse Tree	381
	Connecting the Back to the Front	383
20	Register Transfer Language	387
	RTL Insns	388
	The Six Fundamental Expression Codes	388
	The Type and Content of Insns	388
	Modes and Mode Classes	411
	Flags	415
21	Machine-Specific Compiler Options	419
	The Machine List	420
	The GCC Command-Line Options	421
	Alpha Options	421
	Alpha/VMS Options	426
	ARC Options	426
	ARM Options	427
	AVR Options	433
	CRIS Options	433
	D30V Options	437
	H8/300 Options	437
	HPPA Options	438
	IA-64 Options	440
	Intel 386 and AMD x86-64 Options	441
	Intel 960 Options	446
	M32R/D Options	448
	M680x0 Options	449
	M68HC1x Options	452
	M88K Options	452
	MCore Options	456
	MIPS Options	457

MMIX Options	462
MN10200 Options	464
MN10300 Options	464
NS32K Options	464
PDP-11 Options	467
RS/6000 and PowerPC Options	468
RT Options	478
S/390 and zSeries Options	478
SH Options	479
SPARC Options	481
System V Options	486
TMS320C3x/C4x Options	486
V850 Options	489
VAX Options	490
Xstormy16 Options	490

Part IV

Appendices

A GNU General Public License	493
Preamble	494
B Environment Variables	501
C Command-Line Cross Reference	505
Cross Reference	506
D Command-Line Options	515
Option Prefix	516
The Order on the Command Line	517
The File Types	518
Alphabetic List of Options	519
E Glossary	599
Index.....	623

Acknowledgments

I must thank Wendy Rinaldi at McGraw-Hill/Osborne for giving me the opportunity to write this book, and for her patience in the early days when it looked like it was going to take forever.

I want to thank Katie Conley for keeping me on track and heading in the right direction. She has a unique ability for keeping track of the status of the various parts of the book as it moves through the editing process. Bart Reed and I have a completely different take on the English language—his is both readable and correct. I want to thank Paul Garland for checking the technical accuracy of the book and pointing out the places where my imagination overtook the facts.

I must thank Margot Maley at Waterside for keeping my feet on the ground and my hands on the keyboard.

My understanding of how compilers work was a necessity for writing this book. I want to thank Dave Rogers for introducing me to the C language many years ago, and for drafting me to write a compiler for it. I also need to thank Ron Souder and Travis Mitchell for throwing me into some very strange projects that caused me to become immersed in some of the more obscure nooks and crannies of language processing and object code generation.

Perhaps most of all, I owe a great deal of thanks to the late Fred Lewis for introducing me to the fascinating world of compilers, assemblers, and linkers.