Modern Fortran in Practice

From its earliest days, the Fortran programming language has been designed with computing efficiency in mind. The latest standard, Fortran 2008, incorporates a host of modern features, including object-orientation, array operations, user-defined types, and provisions for parallel computing.

This tutorial guide shows Fortran programmers how to apply these features in twenty-first-century style: modular, concise, object-oriented, and resource-efficient, using multiple processors. It offers practical real-world examples of interfacing to C, memory management, graphics and GUIs, and parallel computing using MPI, OpenMP, and coarrays. The author also analyzes several numerical algorithms and their implementations and illustrates the use of several open source libraries. Full source code for the examples is available on the book's website.

Arjen Markus is a senior consultant at Deltares, an institute for applied research in the field of water, subsurface and infrastructure in The Netherlands, where he develops and maintains their numerical modeling programs and the tools that accompany them. He is an active contributor to the ACM newsletter *Fortran Forum* and the comp.lang.fortran newsgroup.

Modern Fortran in Practice

Arjen Markus
with Foreword by Michael Metcalf

CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE UNIVERSIGTY PRESS
Cambridge, New York, Melbourne, Madrid, Cape Town,
Singapore, São Paulo, Delhi, Mexico City

Cambridge University Press
32 Avenue of the Americas, New York, NY 10013-2473, USA

www.cambridge.org
Information on this title: www.cambridge.org/9781107603479

First published 2012

Printed in the United States of America

A catalog record for this publication is available from the British Library.

Library of Congress Cataloging in Publication Data

Markus, Arjen.
Modern Fortran in practice / Arjen Markus, Michael Metcalf.
pages cm
Includes bibliographical references and index.
ISBN 978-1-107-01790-0 (hardback) – ISBN 978-1-107-60347-9 (pbk.)
1. Fortran 2008 (Computer program language) I. Metcalf, Michael. II. Title.
QA76.73.F25M375 2012
005.26′2–dc23 2012000126

ISBN 978-1-107-01790-0 Hardback
ISBN 978-1-107-60347-9 Paperback

Additional resources for this publication at http://flibs.sf.net/examples-modern-fortran.html

"Eadem mutata resurgo"
(Loosely: though changed, I reappear as myself)
Inscription on the grave of Jacob Bernoulli,
referring to the logarithmic spiral.

In memory of my father

My parents taught me to be inquisitive.
My wife and kids teach me still other
important things.

Contents

Foreword

The applications of Fortran span, very nearly, the whole period during which computers have been in general-purpose use. This is quite remarkable and, given the demise so many other high-level languages, it is quite difficult to know why. Possibly the original design concepts of John Backus – ease of use and efficiency of execution – have been two major factors. Another might be the devotion of Fortran's user community, who labor to keep it abreast of developments in programming techniques and to adapt it to ever-more demanding requirements.

Despite all the predictions, over several decades, that Fortran is about to become extinct, the language has shown itself to be remarkably resilient. Furthermore, over the last few years, it has been subject to new rounds of standardization, and the latest standard, Fortran 2008, should again extend the language's life. Against this background, it is very regrettable that old versions of Fortran live on, both in the form of antiquated courses given by incorrigible teachers and also as an outmoded concept in the minds of its detractors. After all, modern Fortran is a procedural, imperative, and compiled language with a syntax well suited to a direct representation of mathematical formulae. Its individual procedures may be compiled separately or grouped into modules, either way allowing the convenient construction of very large programs and procedure libraries. The language contains features for array processing, abstract data types, dynamic data structures, object-oriented programming, and parallel processing. It can readily be interfaced to C. Thus, modern Fortran, as versions from Fortran 95 onwards are called, is a powerful tool. It fully supports structured programming, and the introduction of object-oriented programming into Fortran 2003, which was its most significant new feature, represented a major upgrade. Much of this is explained in this book.

Nevertheless, no Fortran standard up to and including Fortran 2003 contained any significant feature intended directly to facilitate parallel programming. Rather, this has had to be achieved through the intermediary of ad hoc auxiliary standards, in particular HPF, MPI, OpenMP, and Posix Threads. The use of MPI and OpenMP has become widespread, but HPF, ultimately, met with little success. However, now, with the advent of Fortran 2008, a particular strength of modern Fortran is its ability to support parallel programming, as this latest standard introduces a sorely needed facility: coarrays.

HPF directives took the form of Fortran comment lines that were recognized as such only by an HPF processor. An example was the ability to align three

conformable (matching in shape) arrays with a fourth, thus ensuring locality of reference. Further directives allowed, for instance, aligned arrays to be distributed over a set of processors. MPI, on the other hand, is a library specification for message passing, and OpenMP, which supports multiplatform, shared-memory parallel programming, consists of a set of compiler directives, library routines, and environment variables that determine run-time behavior. Posix Threads is a library specification, for multithreading.

By contrast, the objective of coarrays is to provide a syntax, designed to make the smallest possible impact on the appearance of a program, that enables the distribution over some number of processors not only of data, as in a Single-Instruction-Multiple-Data (SIMD) model, but also of work, using the Single-Program-Multiple-Data (SPMD) model. The facility requires a programmer to learn only a modest set of new rules. Coarray handling is Fortran 2008's single most important new feature, but the *do concurrent* form of loop control is also introduced as a way of parallelizing loops. We then see how it becomes possible to achieve a significant degree of parallelization without going outside the language itself. These various paradigms are compared and contrasted in this book.

Other major new features in Fortran 2008 include: submodules, enhanced access to data objects, enhancements to I/O and to execution control, and additional new intrinsic procedures, in particular for bit processing. Fortran 2008 was published in 2010 and is the current standard. If Fortran lives on, it will be because it is well suited to applications in high-performance computing and thus, coarrays will be crucial.

But a language cannot survive without a means to learn about it. This implies the availability not only of textbooks on the language's syntax and semantics, but also of books on how to use the language in real-life situations. Somehow, experience in the use and application of a language needs to be passed on to a new generation of programmers and new features require advice on how they are best to be used. At a time when a single language is rarely used in isolation, but more often in conjunction with other languages or with various tools, *Modern Fortran in Practice* fulfills a real need for practical advice in the field.

The author has been a regular contributor to the ACM newsletter *Fortran Forum* and has given valuable and much appreciated advice on the comp.lang.fortran user newsgroup, where he has been very active as a contributor. Here, the community has been able to benefit from his experience in scientific programming in The Netherlands. His papers on topics such as generic programming and design patterns were quite novel to Fortran. Thus, he is more than amply qualified to write this book.

But *Modern Fortran in Practice* is not merely a concatenation of previous contributions. They have been woven into a coherent primer, together with original material on parallel programming in Fortran, using MPI, OpenMP

and, briefly, coarrays, as well as on the use of Fortran for graphics applications and within GUIs. It is supplemented by extended examples that will be invaluable to providing a framework upon which users can build.

This book is a most worthwhile undertaking and I commend it to all Fortran practitioners. After all, as we have seen, Fortran is in for the long haul.

Michael Metcalf
Tokyo, October 2011

Preface

I have been programming in Fortran for more than 25 years, first in FORTRAN IV and somewhat later in FORTRAN 77. In the last decade of the 20th century, I attended, together with a number of colleagues, a course on Fortran 90, given by the late Jan van Oosterwijk at the Technical University of Delft. It was also around this time that I came to know the comp.lang.fortran newsgroup, and I have learned a lot by participating in that friendly community.

In a way, I am a typical Fortran programmer. My background is physics and I learned the task of programming partly during my study, but mostly on the job. In other ways, I am not because I took a fancy to the more esoteric possibilities of programming in general and sought means to apply them in Fortran. I also began writing articles for the *ACM Fortran Forum*. These articles are the groundwork for this book.

This book will not teach you how to program in Fortran. There are plenty of books dedicated to that ([22], [65]). Instead, the purpose of this book is to show how modern Fortran can be used for modern programming problems, such as how techniques made popular in the world of object-oriented languages like C++ and Java fit neatly into Fortran as it exists today. It even shows some techniques for solving certain programming problems that are not easily achieved in these languages.

If you know Fortran mainly from the days before Fortran 90, you may find the first few chapters to be a gentle introduction to array operations, overloaded operations, and other features that were introduced by that standard. You will find that Fortran has also opened the way to completely different styles of programming, normally associated with *functional* programming languages. Most chapters are dedicated to illustrating how all of these language features can be employed in practice.

In this book, I often refer to software I have written myself and published via the *SourceForge* website or to software I am involved with in some other way. This is not to promote that particular software over anything else – it is merely a consequence of knowing that software very well. I have tried to attribute all of the examples that are not my own to the people who have written them. However, as I am only human, I may have forgotten one or two names.

A book like this can hardly come into existence in isolation. Besides Michael Metcalf and Ian Chivers, the editors of the *ACM Fortran Forum*, and many people who participate in the comp.lang.fortran newsgroup, I am indebted to

at least the following people, in no particular order: Bil Kleb, Paul van Delst, Rolf Ade, Henry Gardner, Simon Geard, Richard Suchenwirth, Daniel Kraft, Richard Maine, Steve Lionel, Cameron Laird, and Clif Flynt. I thank them for their discussions, reviewing my contributions, and bringing various aspects of programming, directly related to Fortran or otherwise, to my attention.

The website that accompanies this book, http://flibs.sf.net/examples_modern_fortran.html contains the full source code for the examples. I have run them using the *gfortran* and *Intel Fortran* compilers, mostly on Windows, but also on Linux. As some of these programs use the very latest features of Fortran, you will need a recent version of the compiler of your choice.

Arjen Markus
Rotterdam, November 2011