BUILDING PARALLEL, EMBEDDED, AND REAL-TIME APPLICATIONS WITH ADA

The arrival and popularity of multi-core processors have sparked a renewed interest in the development of parallel programs. Similarly, the availability of low-cost microprocessors and sensors has generated a great interest in embedded real-time programs. This book provides students and programmers whose backgrounds are in traditional sequential programming with the opportunity to expand their capabilities into parallel, embedded, real-time, and distributed computing. It also addresses the theoretical foundation of real-time scheduling analysis, focusing on theory that is useful for actual applications.

Written by award-winning educators at a level suitable for undergraduates and beginning graduate students, this book is the first truly entry-level textbook in the subject. Complete examples allow readers to understand the context in which a new concept is used, and enable them to build and run the examples, make changes, and observe the results.

JOHN W. MCCORMICK is Professor of Computer Science at the University of Northern Iowa.

FRANK SINGHOFF is Professor of Computer Science at Université de Bretagne Occidentale (University of Brest).

JÉRÔME HUGUES is Associate Professor in the Department of Mathematics, Computer Science, and Control at the Institute for Space and Aeronautics Engineering (ISAE), Toulouse.

BUILDING PARALLEL, EMBEDDED, AND REAL-TIME APPLICATIONS WITH ADA

JOHN W. MCCORMICK
University of Northern Iowa

FRANK SINGHOFF
Université de Bretagne Occidentale

JÉRÔME HUGUES
Institute for Space and Aeronautics Engineering (ISAE), Toulouse

CAMBRIDGE
UNIVERSITY PRESS

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

Cambridge University Press
The Edinburgh Building, Cambridge CB2 8RU, UK

Published in the United States of America by Cambridge University Press, New York

www.cambridge.org
Information on this title: www.cambridge.org/9780521197168

First published 2011

Printed in the United Kingdom at the University Press, Cambridge

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

Library of Congress Cataloging in Publication data
McCormick, John W., 1948–
Building parallel, embedded, and real-time applications with Ada / John W. McCormick,
Frank Singhoff, Jerome Hugues.
p. cm.
Includes bibliographical references and index.
ISBN 978-0-521-19716-8 (hardback)
1. Ada (Computer program language) 2. Parallel programming (Computer science)
3. Embedded computer systems – Programming. 4. Real-time data processing.
5. Multiprocessors – Programming.
I. Singhoff, Frank. II. Hugues, Jerome. III. Title.
QA76.73.A35M375 2011
004′.35 – dc22 2010053214

ISBN 978-0-521-19716-8 Hardback

The photograph on the cover shows astronaut Stephen K. Robinson standing on the end of the International Space Station's robotic manipulator system, Canadarm 2. This arm, built by MacDonald, Dettwiler, and Associates Ltd for the Canadian Space Agency, is 17.6 m long when fully extended. It has seven motorized joints, each a complex embedded real-time system. Given its crucial role on the space station, the reliability of Canadarm 2 must be impeccable. The software for these joints and for the workstation that the astronauts use to control them is written in Ada. This book provides an introduction to the concepts of concurrent programming, embedded systems, and real-time constraints necessary for understanding and developing the software for such systems.

Contents

Illustrations

Tables

Foreword

The task of programming is a difficult one. Over the decades there have been many promises to eliminate this difficulty. It is interesting to recall that Fortran was first promoted on the basis that it would do away with programming, and allow scientists to enter their mathematical equations directly into the computer. Yet, the difficulty of programming has not gone away, and indeed, the task is more difficult now, especially because of the widespread introduction of parallelism into all realms of programming.

In universities today, many professors find that students shy away from difficult problems and challenges. As a result computer science programs have been made easier and "more fun," and many recent textbooks reflect this worrisome trend. What we need is not students who find easy stuff fun, we need students who find difficult challenges fun!

In this climate a textbook that tackles the most difficult area of programming, the creation of complex embedded systems in which parallelism and real-time concerns play a major role, is very welcome. We entrust our lives to such complex systems all the time these days, in cars, planes, trains, medical equipment, and many other circumstances.

This book addresses the task of teaching the complex elements required to create such systems. The choice of Ada, though unfamiliar for say the creation of simple web programs, is a natural one for two reasons. First, it is the only language in common use where tasking and parallelism play an important "first-class" citizen role (C/C++ rely entirely on external libraries for such support, and the support for such concepts in Java is very weak). Second, Ada is indeed a language of choice for building complex systems of this kind. For example, much of the avionics in the new Boeing 787 "Dreamliner" is written in Ada, as is the new air traffic control system in England. The choice of Ada as a vehicle is thus a good one, and actually

makes it easier for students to grasp the critical new concepts, even though they may have been exposed to other languages previously.

This book is an important addition to the arsenal of teaching materials for a well-educated computer science graduate. It is also eminently readable, and, perhaps I can even say it is fun to read. Despite the potentially dry nature of this subject matter, it is presented in an entertaining and accessible manner that reflects the remarkable teaching skills of its authors.

Robert Dewar
AdaCore

Preface

The arrival and popularity of multi-core processors have sparked a renewed interest in the development of parallel programs. Similarly, the availability of low cost microprocessors and sensors has generated a great interest in embedded real-time programs. This book provides students and programmers with traditional backgrounds in sequential programming the opportunity to expand their capabilities into these important emerging paradigms. It also addresses the theoretical foundations of real-time scheduling analysis, focusing on theory that is useful for real applications.

Two excellent books by Burns and Wellings (2007; 2009) provide a *complete*, *in depth* presentation of Ada's concurrent and real-time features. They make use of Ada's powerful object-oriented programming features to create high-level concurrent patterns. These books are "required reading" for software engineers working with Ada on real-time projects. However, we found that their coverage of *all* of Ada's concurrent and real-time features and the additional level of abstraction provided by their clever use of the object-oriented paradigm made it difficult for our undergraduate students to grasp the fundamental concepts of parallel, embedded, and real-time programming. We believe that the subset of Ada presented in this book provides the simplest model for understanding the fundamental concepts. With this basic knowledge, our readers can more easily learn the more detailed aspects of the Ada language and the more widely applicable patterns presented by Burns and Wellings. Readers can also apply the lessons learned here with Ada to the creation of systems written in more complex languages such as C, C++, and real-time Java.

The first chapter gives an overview of and motivation for studying parallel, embedded, and real-time programming. The fundamental terminology of parallel, concurrent, distributed, real-time, and embedded systems is presented in the context of two cooks sharing resources to prepare food.

The first six sections of Chapter 2 provide a brief introduction to sequential programming with Ada. Only those Ada constructs relevant to the remainder of the book are presented. In conjunction with the on-line learning references provided, our students whose backgrounds did not include Ada were able to come quickly up to speed with their peers who had some previous Ada experience. Students with some Ada background found this chapter an excellent review of the language. The final section of this chapter provides detailed coverage of Ada's low-level programming constructs. These features allow the program to interact with hardware — almost always necessary in an embedded system. These features are illustrated with a complete example of a device driver for a sophisticated analog to digital converter.

Chapter 3 introduces the notion of the task, Ada's fundamental construct for parallel execution. We show how tasks are created, activated, run, and completed. We use state diagrams to illustrate the life cycle of a task. We present the parent-child and master-dependent task hierarchies. We conclude this chapter with a brief discussion of some less frequently used tasking features: abortion, identification, and programmer-defined attributes.

Chapter 4 introduces the most widely used construct for communication and synchronization between tasks — the protected object. The need for mutual exclusion, introduced with cooking examples in Chapter 1, is revisited in the context of an Ada program with multiple tasks incrementing a shared variable. After demonstrating this classic problem, we introduce encapsulation of shared data within a protected object. We illustrate the locking mechanisms provided by protected functions and procedures with several figures. The need for synchronization, again introduced with cooking examples in Chapter 1, is revisited and the solution based on the protected entry is discussed and illustrated. We provide additional practice with protected objects by developing a number of useful concurrent patterns — semaphores, barriers, and broadcasts. The use of requeue is motivated with a problem of allocating a limited number of pencils. The chapter concludes with a discussion of interrupts. We return to the analog to digital converter introduced in Chapter 2 and replace the polling logic with an interrupt handler.

Chapter 5 looks at the use of direct task interaction through the rendezvous. We begin by illustrating the behavior of simple entry calls and accepts. Then we discuss the variations of the selective accept statement that provides the server with more control over its interactions with clients. We introduce the use of both relative and absolute delays. Next we discuss the entry call options which give a client more control over its interaction

with a server. We conclude this chapter with a discussion on the implementation of state machines for both passive and active objects.

Distribution is an important issue in software engineering and is the topic of Chapter 6. While it is easy to motivate the advantages of distribution, the details take more effort. We introduce middleware and discuss the major problems with distributed applications. We provide an introduction to two approaches for distributing an Ada program. The Distributed Systems Annex (DSA) provides a solution for a pure Ada application. We develop the software for a very simple distributed student information system through three examples of increasing complexity. For those who want more details, we provide information on more advanced DSA concepts. While CORBA requires a larger amount of code to implement a distributed system than the DSA, it is more commonly used in industry. Not only is there more support for CORBA than the DSA, it provides an easy integration of different programming languages in a single application. We use the same example of a distributed student information system to develop a CORBA-based solution. Again, we provide additional details for those wanting a more advanced understanding of CORBA. We introduce and use the PolyORB tool for our DSA and CORBA examples.

Chapter 7 is an introduction to the theoretical foundations of real-time scheduling analysis, focusing on theory that is useful for real applications. This presentation is independent of any programming language. We begin with an introduction of the characteristics of a task relevant to scheduling analysis. We define the characteristics of schedulers and discuss fixed priority scheduling and earliest deadline first scheduling. Ensuring that all tasks meet their deadlines is paramount to any hard real-time system. We show you how to calculate and use processor utilization factors and worst case response times to verify schedulability. We begin by analyzing examples with independent tasks. Then we add the complexities of resource sharing with different priority inheritance protocols.

In Chapter 8 we explain how to write real-time applications in Ada that are compliant with the scheduling theory discussed in the previous chapter. In particular, we show how to implement periodic tasks, activate priority inheritance protocols, and select an appropriate scheduler. The first six sections discuss how we can use Ada's Real-Time Systems Annex to implement compliant applications. The remainder of the chapter is devoted to implementing such applications with POSIX. We conclude with a comparison of the two approaches.

Our Ada applications do not run alone. They execute within a run-time configuration consisting of the processor and the environment in which the

application operates. Our final chapter introduces the reader to run-time environments. We discuss three variants of the GNAT run-time: ORK+, MaRTE, and RTEMS. We examine the effect of the run-time on the analysis of schedulability and the determination of task characteristics such as worst case execution time. The schedulability tools MAST and Cheddar are also introduced.

Resources

A website with the complete source code for all examples and some of the exercises in the book may be found at `www.cambridge.org/9780521197168.`

Solutions to all of the exercises are available to qualified instructors. Please visit `www.cambridge.org/9780521197168.`

Acknowledgments

We would like to thank the many individuals who have helped us with this project. The comments, corrections, and suggestions made by our technical reviewers, Alexander Mentis and Pat Rogers, have enormously improved and enriched this book. We are grateful to them both. In addition to providing many useful comments and corrections, Dan Eilers compiled many of our examples with the ICC Ada compiler to ensure we did not use any GNAT-specific constructs.

It is sometimes difficult for those of us with years of experience to write for those who are just beginning. The students in the Real-Time Systems class at the University of Northern Iowa did not hesitate to point out those portions of the manuscript, including the exercises, they felt needed further explanation.

Anyone who has written a textbook can appreciate the amount of time and effort involved and anyone related to a textbook author can tell you at whose expense that time is spent. John thanks his wife Naomi for her support and understanding.

Frank would like to thank all his Master's students who have for the past 10 years tested his courses and exercises on real-time systems. In 2002, we began work on Cheddar, a simple tool whose goal is to provide students with a better understanding of real-time scheduling. We were helped by different partners and contributors: special thanks to Pierre Dissaux, Alain Plantec, Jérôme Legrand, Laurent Pautet, Fabrice Kordon, Peter Feiler, Jérôme Hugues, Yvon Kermarrec and all the others that are listed on the Cheddar website. Finally, many thanks Magali for your patience during all my busy days off!

Jérôme would like to thank all his friends and colleagues for the memorable years he spent working on PolyORB at Telecom ParisTech: Laurent Pautet, Fabrice Kordon, and all our Master's students that helped us at that time; Bob Duff and Thomas Quinot from AdaCore; and Vadim Godunko for all his personal contributions to the project. Participating in such a large project, and seeing it now as a project used in the industry, is a great pleasure. Finally, I'd like to thank Alice for letting me write this book near the fireplace over the long winter nights!

John W. McCormick
University of Northern Iowa
mccormick@cs.uni.edu

Frank Singhoff
Université de Bretagne Occidentale
singhoff@univ-brest.fr

Jérôme Hugues
Institute for Space and Aeronautics
Engineering (ISAE), Toulouse
jerome.hugues@isae.fr