This book provides an elementary introduction to classical recursive descent compiling and is aimed at the undergraduate. Its style is clear and unambiguous. The prerequisites of the reader are that they are familiar with elementary MODULA-2, basic data structures and a simple assembly language. MODULA-2 is used both as a source language and an implementation language to illustrate one compiling technique. The reader is not required to have a mathematical background and no knowledge of automata theory is required.

The author takes an evolutionary approach rather than top-down approach. This works well, introducing the reader to simple concepts before leading through to more complex ideas. The reader is introduced to the overall concept of compiling by a description of the main elements of a compiler, which sets the scene for the rest of the text. The book then continues with assembly language before introducing parsing.

The assembly language SAL—Simplified Assembly Language—(a simplified version of Intel 8086 assembly language) is used throughout the book to provide a practical illustration of the concepts of compiling. The assembler uses recursive descent and it is the use of the assembler rather than any abstract ideas which leads the student to understand recursive descent compiling. Readers need to understand basic SAL and its explanation in Chapter 2 is relatively straightforward.

Compiling is tackled by describing its components and then introducing the methods of compiling expressions before continuing on to compiling control structures through to compiling procedures and modules. Final chapters include error recovery and a brief introduction to compiler optimization. Each chapter concludes with a discussion (useful for student revision) and exercises.

The code used to illustrate concepts throughout the book is very readable. The absence of overflow checks, etc., avoids unnecessary distraction for the student trying to grasp the basic concepts. The software is available on disk from the publisher.

The strength of the text is its clear concise style with an undaunting approach. It is unashamedly aimed at the non-expert and assumes very little knowledge of the subject. Some may consider its weakness to be its lack of depth but the aim of the book is to provide an elementary grounding in the subject. Other methods of compiling are not referred to in the text and this may be considered a disadvantage. Certainly, a section containing a discussion of other types of compiler may be useful.

Overall, the book offers a straightforward explanation and illustration of recursive descent compiling for the undergraduate.

JUDITH JONES
De Montfort University

GEORGE COULOURIS, JEAN DOLLIMORE AND TIM KINDBERG
Distributed Systems—Concepts and Design. 2nd edn. Addison-Wesley. 1994. ISBN 0-201-62433-8. £23.95, 644 pp. hardbound.

The second edition of *Distributed Systems—Concepts and Design* is very different from the first: it has an additional author and it is more than twice as thick.

The book briefly puts distributed systems in a historical perspective and then covers networking, IPC and RPC, structure of distributed operating systems, file servers and name servers, time and coordination, replication, centralized and distributed transactions, concurrency control, recovery and fault tolerance, security, and distributed shared memory. The book ends with the presentation of several case studies.

The book reflects a preference of the authors for breadth rather than depth. It is very complete in its coverage of important experimental distributed-systems research, but the price for this is that theory of distributed systems is only marginally covered.

The preface suggests that the book can be used for undergraduate as well as for postgraduate teaching. Although the volume of the material in the book certainly justifies this (covering the whole book would take roughly 50 h of lecturing), I find the level of presentation basically that of undergraduate teaching. The book explains principles but not algorithms, it explains what but not how.

Chapter 3 ('Networking and Internetworking'), for example, presents the OSI reference model and explains what each layer is supposed to achieve, but does not discuss how this is done. I believe treatment of fault models and fault tolerance is essential in a book on distributed systems and I find the treatment of this subject too shallow. What I find especially missing is a discussion of the fundamental possibilities and impossibilities for masking faults under various fault models.

In spite of this defect, I find the book very useful as a first introduction to distributed systems. All the material on distributed systems I could ever hope to teach in an undergraduate course is present and up to date. The descriptions of the systems discussed are balanced and clear. Questions, at the end of each chapter, are useful for students to test their understanding of the material.

SAPE J. MULLENDER
University of Twente

ANDREAS PAEPCKE (ed.)
Object Oriented Programming: The CLOS Perspective. Cambridge University Press. 1993. ISBN 0 13 092990 5. £27.95. 202 pp. hardbound.

This is not a book about the methodology of object-oriented programming nor is it a book that teaches you how to program using the Common Lisp Object System (CLOS). Rather, it is a book that is intended to demonstrate the influence that the features of a particular object-oriented programming language, i.e. CLOS, can have on your approach to object-oriented programming. In other words, a book that provides a

Downloaded from https://academic.oup.com/comjnl/article/37/8/727/398540 by guest on 20 April 2024