Portable Programming

Macmillan Computer Science Series

Consulting Editor

Professor F. H. Sumner, University of Manchester

S. T. Allworth, *Introduction to Real-time Software Design*

Ian O. Angell, *A Practical Introduction to Computer Graphics*

G. M. Birtwistle, *Discrete Event Modelling on Simula*

T. B. Boffey, *Graph Theory in Operations Research*

Richard Bornat, *Understanding and Writing Compilers*

J. K. Buckle, *The ICL 2900 Series*

J. K. Buckle, *Software Configuration Management*

Robert Cole, *Computer Communications*

Derek Coleman, *A Structured Programming Approach to Data**

Andrew J. T. Colin, *Fundamentals of Computer Science*

Andrew J. T. Colin, *Programming and Problem-solving in Algol 68**

S. M. Deen, *Fundamentals of Data Base Systems**

J. B. Gosling, *Design of Arithmetic Units for Digital Computers*

David Hopkin and Barbara Moss, *Automata**

Roger Hutton, *Fortran for Students*

Roger Hutton, *Z80 Assembly Language Programming for Students*

H. Kopetz, *Software Reliability*

Graham Lee, *From Hardware to Software: an introduction to computers*

A. M. Lister, *Fundamentals of Operating Systems, second edition**

G. P. McKeown and V. J. Rayward-Smith, *Mathematics for Computing*

Brian Meek, *Fortran, PL/I and the Algols*

Derrick Morris and Roland N. Ibbett, *The MUS Computer System*

John Race, *Case Studies in Systems Analysis*

B. S. Walker, *Understanding Microprocessors*

Peter J. L. Wallis, *Portable Programming*

I. R. Wilson and A. M. Addyman, *A Practical Introduction to Pascal*

* The titles marked with an asterisk were prepared during the Consulting Editorship of Professor J. S. Rohl, University of Western Australia.

Portable Programming

Peter J. L. Wallis

*School of Mathematics,
University of Bath*

M

© Peter J. L. Wallis 1982

All rights reserved. No part of this publication may be reproduced or transmitted, in any form or by any means, without permission.

First published 1982 by
THE MACMILLAN PRESS LTD
London and Basingstoke
Companies and representatives
throughout the world

Typeset in 10/12 IBM (Press Roman) by
RDL., 26 Mulgrave Road, Sutton, Surrey

ISBN 978-0-333-31036-6   ISBN 978-1-349-05925-6 (eBook)
DOI 10.1007/978-1-349-05925-6

The paperback edition of the book is sold subject to the condition that it shall not, by way of trade or otherwise, be lent, resold, hired out, or otherwise circulated without the publisher's prior consent in any form of binding or cover other than that in which it is published and without a similar condition including this condition being imposed on the subsequent purchaser.

For Edith, Susan and Albert

Contents

<i>Foreword</i> by Peter Brown	xi
<i>Preface</i>	xiii
1 General Introduction	1
Advantages of Portability	2
Portable Programming and Industry Standards	3
Portable Programming and Industry Attitudes	5
Portable Programming in Practice	8
Reading this Book	9
PART I BACKGROUND TO PORTABLE PROGRAMMING	11
2 Legal Protection of Software	13
Patent Protection	14
Copyright and Trademarks	15
Confidentiality and Contract	16
Current Practice and Future Developments	17
References	18
3 Planning for Portability	19
Donor and Installer	19
Documentation of Portable Software	20
Testing of Portable Programs	23
References	25
4 Maintaining Portable Programs	26
An Installer's Text Management Scheme	27
Centralised Text Management within NAG	29
Other Centralised Text Management Schemes	30

Program Versions and Abstraction	31
Summary of Maintenance Issues	33
References	
5 Physical Distribution Issues	35
Character Sets	36
Use of Punched Cards	38
Use of Paper Tape	38
Discs and Magnetic Tapes for Mainframe Computers	39
Magnetic Tapes and Discs for Small Computers	40
Use of Data Communications Lines and Interfaces	43
References	44
PART II PORTABLE PROGRAMMING TECHNIQUES	45
6 Designing Portable Programs	47
Portable Programming and Machine Dependencies	48
Portable Programming and Modular Structure	49
Character Manipulation	51
Arithmetic Range and Accuracy	53
Run-time Environment	54
Data Portability	55
Exotic Computer Architectures	56
Performance Variations	57
References	58
7 Arithmetic Portability	60
Simulation of Standard Arithmetic	60
Types of Arithmetic Hardware	62
Note on Fixed-point Portability	62
Integer Arithmetic Portability	62
Introduction to Floating-point Arithmetic Portability	64
Models for Floating-point Arithmetic	65
Brown's Model for Floating-point Arithmetic	65
Experimental Determination of Parameters	67
Use of the Parameters within Portable Programs	67
References	70
8 Portable Programming in High-level Languages	72
Machine Dependencies	72
Language Dialects	73
Portability and Language Standards	74

	ix
Verifiers, Filters and Preprocessors	75
Guidelines for Portability	75
References	76
9 Portable Programming in Fortran and other Scientific Languages	77
Fortran Dialects	77
Available Software Tools	78
Typical Problems with Fortran Portability	80
Fortran 77 Portability	81
Use of Other Scientific Languages	82
References	83
10 Portable Programming in Cobol	85
Portability and Cobol Standards	86
Production of Portable Cobol Programs	87
References	88
11 Ada and Portable Programming	89
Non-arithmetic Portability Problems	90
Arithmetic Portability in Ada	90
References	93
12 Portable Programming Using Macro Processors	94
Macro Processors	95
Bootstrapping Methods	96
Languages and Abstract Machines	97
Low-level Capabilities	98
High-level Capabilities	99
Comparison of Language Levels	99
Hierarchies of Abstract Machines	100
Relationship to Later Work	102
References	103
13 Portable Compilers	104
Compiler Actions	105
Compiler Bootstrapping	106
Use of Intermediate Languages	107
Design of Intermediate Languages	108
Low-level Intermediate Languages	108
High-level Intermediate Languages	112
Recent Developments	115
References	116

14 Portable Operating Systems	117
OS6	118
Tripos	119
Solo	119
UNIX	119
Thoth	120
MUSS	120
Summary of Experiences	122
MicroProducts Software MicroCobol	122
UCSD/Pascal	123
References	123
 <i>Appendix Products and Organisations</i>	 124
 <i>Bibliography</i>	 126
 <i>Author Index</i>	 136
 <i>Subject Index</i>	 138

Foreword

If portability could be achieved by applying a few general principles, our life would be easy. Sadly, this is not so.

Portability requires both hard work and foresight. It needs attention to a wealth of detail covering issues as diverse as the use of programming languages, formats of physical media such as tapes and discs, the structure of documentation, and questions of legal protection. Yet the hard work is worth while. For each hour invested in achieving portability, we might save ten future hours of tedious work converting programs and data to an alien environment. Huge sums of money are wasted every year, and many tears shed, because people are unwilling to make this small investment.

I therefore welcome a book such as this one, which tries to make people aware of portability and to bring together some of the detail that is scattered widely over the literature. Readers can then separate the wood from the trees, and identify what techniques meet their particular needs, and where to find out more.

PETER BROWN

Preface

It is not easy to write programs that are 'portable' in the sense of being usable on several computers. Many different portable programming projects have been attempted, and many different difficulties encountered as a result. The main obstacles to portability are now well understood, and anyone approaching a portable programming project can be greatly helped by accounts of the experiences of those who have had similar problems.

Most of the current literature on portable programming deals with detailed solutions to specialised problems, ranging (for example) from Fortran dialects to data transfer problems and from legal difficulties to the design of intermediate languages. Such contributions are immensely valuable to those with similar problems, but their specialised nature limits their usefulness for those planning portable programming projects with little idea of where to look for the experiences of people who have tried something similar before.

Ideas like these prompted me to write a book on portability which would supply an overview of the whole field. The aim is to give an account of all the work that has been done on various aspects of portable programming, so that you can see how your particular problems fit into the overall picture. To encourage the reader to gain the necessary perspective, I felt it was important to keep the book fairly short and readable; this should enable you to get a picture of the whole field before plunging into details. If while you are reading the book you identify a particular area that seems relevant to your needs, the references will tell you where to look for more information.

The overall structure of the book reflects this aim by progressing from generalities to particular cases. After an introductory chapter, the book is divided into two parts. Part I, on the background to portable programming, deals with such issues as legal considerations and the planning, maintenance and distribution of portable software. Part II, concerned with portable programming techniques, proceeds from general ideas about the design of portable programs to portable programming in high-level languages and the special problems of portable compilers and operating systems. This progression is intended to encourage readers to study the whole book even if they think they know which chapters are the most relevant to them. The book has also been used as the basis for a course of lectures on portable programming: around 12 hours would be needed to give fair coverage to all the material.

I am extremely grateful for help received from many sources while this book was being written and regret that not all the information so freely provided could, in the end, be used. The editors of *Computing* and *Computer Weekly* published letters appealing for information on portable programming which resulted in my receiving over 30 replies on many different topics. Figures 5.1 and 5.2 are reprinted from Waite (1975) by permission of John Wiley & Sons Ltd, and figure 5.4 is taken from Lesea and Zaks (1978) by permission of Sybex Inc.

Chapter 2 could not have been written without the co-operation of Bryan Niblett of University College, Swansea, and Colin Tapper of Magdalen College, Oxford. I am most grateful to Peter Brown of the University of Kent at Canterbury for his foreword, to Cornelia Boldyreff, Adrian Bowyer and John Slape of the University of Bath for reading and commenting on the book in typescript, and to Colin Theaker of the University of Manchester for his conscientious refereeing efforts. Further improvements resulted from the kind invitation of Martti Tienari to lecture on portable programming at the University of Helsinki. Others who have helped me in various ways include: J. G. P. Barnes, P. M. Bell, F. Boeuf, K. L. Bowles, R. K. Brooks, P. Carr, M. Chiu, A. D'Agapeyeff, S. J. Dawes, J. J. DuCroz, M. E. Falla, J. P. Fitch, B. Ford, T. J. Froggatt, S. J. Hague, A. Hall, W. T. Harwood, C. J. B. Hawkins, C. L. S. Henderson, C. J. Higley, P. D. B. Jones, G. A. Kerry, R. Langer, J. Larmouth, D. Martland, R. A. Mason, A. Mills, R. Moore, D. T. Muxworthy, M. Redwood, J. M. Richards, A. Rogers, O. Roubine, R. Sibson, P. Smith, M. Stewart, J. M. Triance, S. Tumbridge, D. M. Vallance, J. A. Wellford, C. J. West, B. A. Wichmann and S. A. Williams. Finally it is a pleasure to thank Adrienne Higgins for her efficient typing of a messy manuscript.

PETER WALLIS