

SYSTEMS SOFTWARE:

An Introduction to Language Processors and Operating Systems

FRANK MADDIX

GARETH MORGAN



SYSTEMS SOFTWARE:
An Introduction to Language Processors
and Operating Systems



ELLIS HORWOOD BOOKS IN INFORMATION TECHNOLOGY

General Editor: Dr JOHN M. M. PINKERTON, Principal, J. & H. Pinkerton Associates, Surrey (Consultants in Information Technology), and formerly Manager of Strategic Requirements, ICL

EXPERT SYSTEMS IN BUSINESS: A Practical Approach

M. BARRETT, Expertech Limited, Slough, and A. C. BEEREL, Lysia Limited, London

ELECTRONIC DATA PROCESSING IN PRACTICE, Vols. 1 and 2

M. BECKER, R. HABERFELLNER and G. LIEBETRAU, Zurich, Switzerland

EXPERT SYSTEMS: Strategic Implications and Applications

A. C. BEEREL, Lysia Limited, London

APPLICATIONS OF OPTICAL STORAGE*

A. BRADLEY, Congleton, Cheshire

SOFTWARE ENGINEERING ENVIRONMENTS

P. BRERETON, Department of Computer Science, University of Keele

SMART CARDS: Principles, Practice and Applications

R. BRIGHT, Information Technology Strategies International Limited, Orpington, Kent

PRACTICAL MACHINE TRANSLATION*

D. CLARKE and U. MAGNUSSON-MURRAY, Department of Applied Computing and Mathematics, Cranfield Institute of Technology, Bedford

ENCYCLOPAEDIA OF IT*

V. CLAUS and A. SCHWILL, FRG

KNOWLEDGE-BASED SYSTEMS: Implications for Human-Computer Interfaces

D. CLEAL, PA Computers and Telecommunications, London, and N. HEATON, Central Computer and Telecommunications Agency, London

THE CASSANDRA ARCHITECTURE: Distributed Control in a Blackboard System

I. CRAIG, Dept. of Computer Science, University of Warwick, Coventry

KNOWLEDGE ELICITATION: Principles, Techniques and Applications

D. DIAPER, Section of Psychology, Liverpool Polytechnic

KNOWLEDGE-BASED MANAGEMENT SUPPORT SYSTEMS

G. I. DOUKIDIS, F. LANU and G. MILLER, Department of Information Management, London Business School, London

KNOWLEDGE-BASED SYSTEMS: Applications in Administrative Government

P. DUFFIN, CCTA, Riverwalk, London

MULTI-LINGUAL SPEECH ASSESSMENT*

A. FOURCIN, Dept. of Phonetics and Linguistics, University College London

KNOWLEDGE ENGINEERING FOR EXPERT SYSTEMS

M. GREENWELL, Expert Systems International, Oxford

EXPERT SYSTEMS: Safety and Risks*

E. HOLLNAGEL, Computer Resources International, Denmark

KNOWLEDGE-BASED EXPERT SYSTEMS IN INDUSTRY

J. KRIZ, Head of AI Group, Brown Boveri Research Systems, Switzerland

ARTIFICIAL INTELLIGENCE: Current Applications*

A. MATTHEWS and J. RODDY, Aragon International Ltd, London

INFORMATION TECHNOLOGY: An Overview*

J. M. M. PINKERTON, J. & H. Pinkerton Associates, Esher, Surrey

PRACTICAL TOOLS IN HUMAN-COMPUTER INTERFACES*

S. RAVDEN and G. JOHNSON, MRC ESRC Social and Applied Psychology Unit, University of Sheffield

EXPERT SYSTEMS IN THE ORGANIZATION: An Introduction for Decision-makers

S. SAVORY, Nixdorf Computer AG, FRG

ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEMS

S. SAVORY, Nixdorf Computer AG, FRG

BUILDING EXPERT SYSTEMS: Cognitive Emulation

P. E. SLATTER, Telecomputing plc, Oxford

SOFTWARE ENGINEERING MANAGEMENT

H. M. SNEED, FRG

SPEECH AND LANGUAGE-BASED COMMUNICATION WITH MACHINES:

Towards the Conversational Computer

J. A. WATERWORTH and M. TALBOT, Human Factors Division, British Telecom Research Laboratories, Ipswich

THE IMPLEMENTATION OF COMPUTER SYSTEMS

R. J. WHIDDETT, Dept. of Computing, University of Lancaster

* In preparation

SYSTEMS SOFTWARE:

An Introduction to Language Processors and Operating Systems

FRANK MADDIX, B.Sc.(Hons)

and

GARETH MORGAN, M.A., B.A.(Hons), MBCS

Department of Computer Studies and Mathematics
Bristol Polytechnic



ELLIS HORWOOD LIMITED

Publishers · Chichester

Halsted Press: a division of

JOHN WILEY & SONS

New York · Chichester · Brisbane · Toronto

First published in 1989 by
ELLIS HORWOOD LIMITED

Market Cross House, Cooper Street,
Chichester, West Sussex, PO19 1EB, England

The publisher's colophon is reproduced from James Gillison's drawing of the ancient Market Cross, Chichester.

Distributors:

Australia and New Zealand:

JACARANDA WILEY LIMITED

GPO Box 859, Brisbane, Queensland 4001, Australia

Canada:

JOHN WILEY & SONS CANADA LIMITED

22 Worcester Road, Rexdale, Ontario, Canada

Europe and Africa:

JOHN WILEY & SONS LIMITED

Baffins Lane, Chichester, West Sussex, England

North and South America and the rest of the world:

Halsted Press: a division of

JOHN WILEY & SONS

605 Third Avenue, New York, NY 10158, USA

South-East Asia

JOHN WILEY & SONS (SEA) PTE LIMITED

37 Jalan Pemimpin # 05-04

Block B, Union Industrial Building, Singapore 2057

Indian Subcontinent

WILEY EASTERN LIMITED

4835/24 Ansari Road

Daryaganj, New Delhi 110002, India

© 1989 F. Maddix and G. Morgan/Ellis Horwood Limited

British Library Cataloguing in Publication Data

Maddix, Frank

Systems software: an introduction to language processors and operating systems. —
(Ellis Horwood series in computers and their applications).

1. Computer systems. Operating systems.

I. Title II. Morgan, Gareth. III. Series

005.4'3

Library of Congress data available

ISBN 0-7458-0638-4 (Ellis Horwood Limited — Library Edn.)

ISBN 0-7458-0639-2 (Ellis Horwood Limited — Student Edn.)

ISBN 0-470-21422-8 (Halsted Press)

Typeset in Times by Ellis Horwood Limited

Printed in Great Britain by Unwin Bros. Woking

COPYRIGHT NOTICE

All Rights Reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the permission of Ellis Horwood Limited, Market Cross House, Cooper Street, Chichester, West Sussex, England.

Contents

Note on trademarks	9
Preface	11
INTRODUCTION	
1 Why study systems software?	
1.1 What is systems software?	15
1.2 Systems and applications software	16
1.3 Types of systems software	17
1.4 Why use systems software?	21
1.5 The study of systems software	22
PART I — LANGUAGE PROCESSORS	
2 Principles of programming languages	
2.1 Introduction	29
2.2 Generations of programming languages	31
2.3 Procedural and non-procedural languages	34
2.4 Language elements	35
2.5 Formal language definitions	39
2.6 Desirable features of programming languages	42
3 Assembly language and assemblers	
3.1 Assembler principles	46
3.2 Assembly language	47
3.3 The GOAT machine	53
3.4 Functions of an assembler	56
3.5 Operation of an assembler	59
3.6 Practical issues	64
4 Compilers: function and purpose	
4.1 The compilation process	66
4.2 Translation of different constructs	68
4.3 Outputs of a compiler	76
5 Compilers: structure and operation	
5.1 Compiler structure	78
5.2 Lexical analysis	81
5.3 Syntax analysis	83

6 TABLE OF CONTENTS

5.4	Code generation	89
5.5	Additional issues of compiler design	95
6	Compilers: run-time environments	
6.1	Introduction	100
6.2	Error and exception handling	101
6.3	Standard routines and file handling	103
6.4	Storage management	105
7	Linkers and loaders	
7.1	Introduction	110
7.2	Modular programming and linkage editing	110
7.3	Translator output	114
7.4	Task of the linkage editor	115
7.5	Simple loaders	116
7.6	Relocating loaders	117
7.7	Multi-phase linking and dynamic loading	118
8	Interpreters and fourth-generation languages	
8.1	Interpretive execution	120
8.2	Interpretive languages	121
8.3	Structure of interpreters	125
8.4	Comparison of compilation and interpretation	126
8.5	Fourth-generation languages and mixed compiler/interpreters	128
PART II — OPERATING SYSTEMS		
9	Introduction to operating systems	
9.1	Introduction	135
9.2	Operating systems — an overview	135
9.3	Conclusion	142
10	Files and directories	
10.1	Introduction	143
10.2	Files	143
10.3	File access mechanisms	148
10.4	Directories	150
10.5	Conclusion	153
11	Input and output	
11.1	Introduction	155
11.2	Hardware independence	155
11.3	User layer	155
11.4	Kernel level I/O: handlers and drivers	157
11.5	An example: time of day clock	158
11.6	Real drivers	160
11.7	Multitasking	161
12	Controlling the machine	
12.1	Introduction	164

12.2	Multitasking	164
12.3	Memory management	176
12.4	An advanced microprocessor — the Intel 80386	181
12.5	Conclusion	183
13	Some real systems	
13.1	Introduction	184
13.2	OS/2	184
13.3	VMS	188
13.4	Unix	189
13.5	Other systems	195
13.6	Conclusion	196
14	System services	
14.1	Introduction	198
14.2	Types of system service	198
14.3	Calling system services	199
14.4	MS-DOS system services	199
14.5	VAX/VMS system services	200
14.6	Unix system services	202
14.7	Conclusion	203
15	System utilities	
15.1	Introduction	204
15.2	The software development process	204
15.3	Software development tools	204
15.4	Project management	208
15.5	Project management software	208
15.6	Conclusion	211
CONCLUSION		
16	Current and future developments	
16.1	Introduction	215
16.2	Hardware	215
16.3	Software	218
16.4	Operating systems	222
16.5	Conclusion	225
Index		227

Note on trademarks

All names of real products mentioned in this book should be taken to be trademarks of the manufacturer or supplier concerned. In particular:

Unix is a trademark of AT&T Bell Laboratories.

MS-DOS and OS/2 are trademarks of Microsoft.

VMS is a trademark of Digital Equipment Corporation.

IBM and PC-DOS are trademarks of IBM Corporation.

Intel is a trademark of Intel Corporation.

Ada is a trademark of the United States Department of Defense.

The name 'GOAT' is a hypothetical creation for the purpose of this book, and any similarity to the names of any real products is purely coincidental.

Preface

An appreciation of the functions of compilers and operating systems is an essential part of most information technology and computer science courses, yet students beginning in this area are faced with a shortage of material at a suitable level. Particularly in the field of language processors, there is often nothing between the one chapter introductions in general computer studies books, and the advanced texts on compiler theory which are too technical for those who are new to the subject.

This book is primarily intended to accompany an introductory course in systems software, either at undergraduate level, or on post-graduate courses for graduates converting from other disciplines. It should also be suitable for professional examinations such as British Computer Society Part I, and as supplementary reading for those taking A-level computer studies. On elementary courses it will serve as a text in its own right; on more advanced courses it will provide an introduction from which students can tackle more detailed books such as those listed in the further reading at the ends of the chapters.

After an introductory chapter, Part I of the book deals with language processors: programming language principles, assemblers, compilers, and interpreters. Part II covers the field of operating systems, and a concluding chapter then discusses current and future implications for systems software. We believe students will find it convenient to have both the main areas of systems software in a single volume, but Parts I and II can be used independently for distinct courses. Part I has been prepared by Gareth Morgan and Part II by Frank Maddix, but there are extensive cross references between the chapters, particularly on topics such as linkers and software development environments, which could be treated under either of the main headings.

No background knowledge is assumed other than an introduction to computer concepts and machine architecture, and some experience of programming.

The emphasis is in looking at systems software very much from the user view, but covering just sufficient detail on internals to enable a general understanding of how a language processor or operating system might function. The authors believe that whilst it is important for all serious computer studies students to understand the principles of systems software,

only a very few will ever be involved in its development, so the book is *not* intended as a manual on compiler or operating system design.

We have aimed to keep the book as broad in application as possible, so examples are taken from many different programming languages and a variety of real operating systems. For reasons of space, the examples are necessarily selective, and many important techniques used in certain systems are either omitted or mentioned only briefly; it is anticipated that a lecture or seminar programme would supplement the text with examples from the systems software actually used.

We would wish to acknowledge the support of many people in making this book possible, particularly our respective partners, Claire and Sharon, who have generously taken on other tasks to free us for this and assisted with checking parts of the text. We are also grateful to our students at Bristol Polytechnic who have impressed upon us the need for an introductory text in this area, and to the staff of Ellis Horwood Limited for their encouragement.

October 1988

Frank Maddix and Gareth Morgan

Introduction

1

Why study systems software?

1.1 WHAT IS SYSTEMS SOFTWARE?

When people start to use computers, they usually have a fairly clear idea of what is meant by the hardware, even though with a big multi-user system they may never get to see the machine room that contains the central processor. Very quickly, they gain an idea of what is meant by applications software in terms of instructing the computer to carry out tasks for their specific applications.

But systems software is often much more of a mystery; the user is frequently unaware of anything other than the application and the hardware. Indeed many application systems are nowadays deliberately designed in such a way that the end user has no contact with the operating system or any other aspect of systems software.

The simplest definition of systems software is that it is software which bridges the gap between application software on the one hand and hardware on the other.

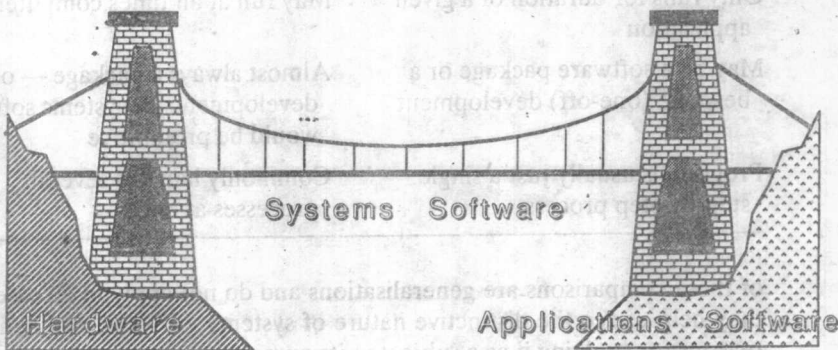


Fig. 1.1 — The bridge view of systems software.

The words **systems** and **software** are both significant. The fact that we are studying a particular kind of *software* is important: language processors and operating systems are still very much programs and they control the

computer by sequences of instructions, in the same way as any other programs. They are quite distinct from the hardware — even though the hardware and systems software may have been designed for use together. But the term *systems software* stresses that these programs are related to the control and management of the computer system as a whole, rather than to a particular application.

For the user who is concerned only with the results of an application, the systems software is usually quite irrelevant. But for the programmer or systems designer, the systems software is vital, as it saves the application developer from having to be concerned with the details of hardware operation.

1.2 SYSTEMS AND APPLICATIONS SOFTWARE

In clarifying the role of systems software it is helpful to assess the differences between applications and systems software, which Table 1.1 sets out. Many

Table 1.1 — Systems versus application software

Applications software	Systems software
Carries out application	Controls system
Manages application data	Manages machine resources
Provides services to end user	Provides services to programmer or to application software
Usually written in a high-level language	Often written in assembler
Only runs for duration of a given application	May run at all times computer is on
May be a software package or a bespoke (one-off) development	Almost always a package — one-off developments of systems software would be prohibitive
Program is usually just a single step-by-step process	Commonly handles several processes at once

of these comparisons are generalisations and do not apply in all cases, but they help us see the distinctive nature of systems software, and hence the reasons for studying it as a subject in its own right.

These distinctions vary over the years — for example, with the emergence of programming languages such as C it is becoming much more common for systems software (or large parts of it) to be written in a high-level language. Also, there are plenty of examples of research projects where a compiler or operating system has been developed purely for one