SYSTEMS SOFTWARE: An Introduction to Language Processors and Operating Systems

FRANK MADDIX

GARETH MORGAN

SYSTEMS SOFTWARE: An Introduction to Language Processors and Operating Systems

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

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

Contents

Note on trademarks

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.

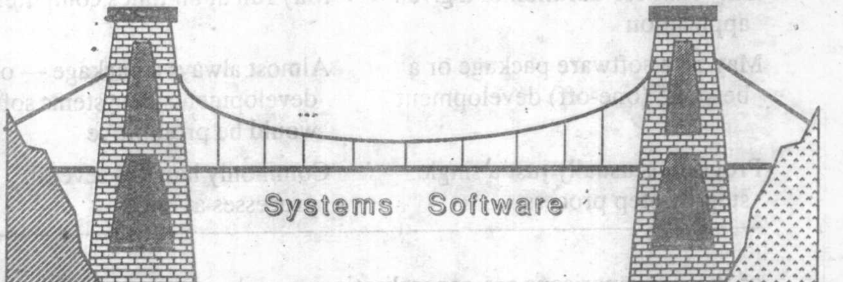

Fig. 1.1 — The bridge view of systems software.

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