The OS/2 Programming Environment

David A. Schmitt

73.8729

The OS/2 Programming Environment

David A. Schmitt

915-0022

Prentice Hall
Englewood Cliffs, New Jersey 07632

9150022

Library of Congress Cataloging-in-Publication Data

Schmitt, David A.
The OS/2 programming environment / by David A. Schmitt.
p. cm.
Includes index.
ISBN 0-13-642927-0
1. OS/2 (Computer operating system) 2. MS-DOS (Computer operating system) I. Title.
QA76.76.063S38 1989
005.265--dc19 88-29759
CIP

Editorial/production supervision: Jacqueline A. Jeglinski
Cover design: Photo Plus Art
Manufacturing buyer: Mary Ann Gloriande

To Karen--

For the inspiration, the time and constant encouragement.

EC81/31

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-642927-0

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

Foreword

The unprecedented explosion in Personal Computers in the early 1980s was due in large part to the thousands of innovative application programs which were developed for the DOS operating system. At first, it seemed that the capabilities of the PC were limitless. But as the environment matured, PC programmers soon started running into the inherent limitations of the DOS architecture. The 640 KB physical address space, its single tasking structure, and the lack of reasonable protection mechanisms make life very difficult for application developers.

In many ways, the PC software industry of 1988 is being throttled by the capabilities of DOS. The increasingly sophisticated features of today's personal computer hardware cannot be harnessed using an operating system which was never intended to support these new machines. To further complicate things, as the hardware has evolved, so has the PC user community. Users are demanding more and more sophisticated application programs which better exploit their hardware investment. The PC application developers are stuck in the middle of the tempest. Truly supporting the new hardware systems means that application developers must spend more time and effort making DOS do "unnatural acts" and less time making their applications state-of-the-art programs.

The DOS engine which so successfully fueled the PC revolution is rapidly running out of steam. But this is not an indictment of DOS as much as it is an acknowledgement that the underlying hardware is developing in ways which were inconceivable in 1981. DOS is rooted in older hardware technologies which have little support for the large memory or protection that is required by responsive, predictable, multitasking environments.

In April, 1987, the IBM and Microsoft Corporations announced *Operating System/2*, the fruit of a joint software development venture. OS/2 is a new system which was built with programmers in mind. While designing the system we tried to listen to the common complaints of DOS programmers. In fact, the design team which defined the OS/2 architecture was composed of some of the best PC minds in the industry today. This joint Microsoft/IBM group brought their collective experience to the project to attempt what many called a "mission impossible" – building a system which addresses the weaknesses of the DOS environment while striving to maintain overall responsiveness and flexibility. Did the team succeed? Well, the ultimate judge will not be any of us who worked on the project. The ultimate judge is you, the programmers who will use the system facilities to build a new generation of exciting application programs and system extensions. We have done our job, but our appraisals will come from you.

As the president of the Lattice corporation, Dave Schmitt has been involved in the OS/2 project for quite some time. His company was one of the first to acknowledge the possibilities of the system by porting their products onto the new platform. I first met Dave at the OS/2 announcement in 1987 where he enthusiastically gave me a first hand report on what we were doing right, and more importantly, what we were doing wrong. His commentary was incisive. It was clear to me then that Dave really understood the PC business and why OS/2 was needed. Since then, I have spoken to him on many occasions. He has always provided me with clear feedback, which was invaluable input to difficult design trade-offs.

Dave puts this understanding to work in this book. As I read through the text, I could see OS/2 through different eyes. What you will read here is the OS/2 system from the perspective of an experienced programmer who has gone through the conversion from DOS to OS/2. His commentary is sometimes kind, sometimes not, but always honest. He does a good job putting the system's capabilities into perspective. I enjoyed reading this book, as I am sure you will.

Ed Iacobucci

Preface

I never met an operating system I didn't like, going all the way back to the early 1960s when I worked with the primitive I/O Control System (IOCS) on the IBM 1401 computer. During subsequent years I was a designer and programmer for four operating systems (which we called central control programs) used in electronic telephone switching systems, culminating with a three-year stint in the early 1980s on the fault-tolerant version of UNIX known as DMERT.

If I've learned anything during twenty-five years studying operating systems, it is how to distinguish the good ones from the stinkers. The good ones are a pleasure to use because they are efficient, understandable, and logically consistent. The stinkers are a pain to use because they're sluggish, arcane, and constantly surprise you with their inconsistencies.

If you smell anything around OS/2, it's the sweet smell of success. This is probably the most complex operating system ever crammed into a microprocessor, but for all of its features, it is remarkably efficient and is quite approachable. You don't need to be an operating system guru to use most of the OS/2 features.

In some ways, OS/2 is a compendium of the good operating system ideas that have evolved during the past twenty-five years. While writing this book about the new technology of OS/2, I've had a lot of fun discovering those old nuggets and seeing how they've been shined up for use in the personal computing environment.

Many thanks to the fine people at IBM who gave me the opportunity to play with OS/2 while it was under development: Joe Carusillo, Steve Hancock, Ed Iacobucci, Tom Peters, Emmet Wainwright, and

many others. I also owe a lot to the programming staff at Lattice who provided valuable insight as OS/2 users, especially Glenn Musial, Tom Prodehl, and John Pruitt. And finally, thanks to Ed Moura, Jackie Jeglinski, and the others at Prentice-Hall who sweated through this book with me, and to Ed Yourdon who delivered the kick in the pants that got me started.

David A. Schmitt

Introduction

Remember the early days of the IBM PC? Programming this little beast wasn't much fun. The original PC was just an anemic 16-bit replacement for the 8-bit CP/M computers then in vogue, and DOS (i.e., Microsoft's MS-DOS and IBM's PC-DOS) didn't do much to correct the deficiencies of CP/M. Furthermore, the Intel 8088 was a nasty chip to program in assembly language, and there weren't many tools available to support programming in higher-level languages.

But within a few years, the PC hardware had improved markedly, and a large number of excellent software development tools were available at reasonable prices. PC programming became a pleasant experience, because these high-level tools provided a smooth ride over the many 8088 potholes. Furthermore, DOS had evolved well beyond its CP/M roots, but it remained an unobtrusive little operating system that could be pushed aside whenever you wanted to get "down and dirty" with the hardware. PC programmers had found something that mainframe and minicomputer programmers dream about: *an environment in which the programmer had total control.*

Unfortunately, the people who pay us programmers to indulge our passion wouldn't let us rest comfortably in this pleasant environment. They saw what we could do with the PC and wanted more: more color, more graphics, more interaction, bigger spreadsheets, better help systems, and on and on. The hardware folks thought this was great. After all, they get their kicks by building systems with more memory, bigger disks, faster processors, and more exotic peripherals. IBM and the other computer manufacturers enjoyed this trend too, since they make most of their money selling hardware.

But those of us who develop PC application software for a living were getting more and more uncomfortable. The software solutions that our customers and managers wanted could only be provided with huge and complex packages that banged up solidly against some DOS limitations that were not so easy to overcome. We found ourselves wasting a lot of time bending, folding, and mutilating our programs so they would fit within the 640-kilobyte memory limit. Many programmers spent long midnight hours devising bizarre schemes to thwart the DOS single-tasking architecture, so we could then develop those essential pop-up utilities, background communication handlers, and other multitasking features.

This constant tussle with DOS took a lot of the fun out of PC programming. Of course, as an old and wise manager once told me, nobody said that programmers are supposed to have fun. But in this case, the pleasure disappeared because PC programming was becoming a slow and tedious process, with constant delays as we figured out how to get around yet another DOS limit. This decrease in efficiency was certainly of great concern to managers as well as programmers. By late 1987, it became apparent that many PC programming projects were in states of crisis, experiencing the delays and setbacks that have often plagued software development projects on larger systems.

But fortunately, December of that year saw the marriage of IBM and Microsoft produce OS/2, an offspring destined to lead us all into a brave new world of PC programming, free of the many problems and constraints in the DOS world. This fancy new operating system combines the best features of DOS and UNIX to give us a programming environment that (at least today) appears to be limitless. We can now wallow in a one-gigabyte address space. We can create entire armies of cooperating processes busily going about their assigned tasks. And with the Presentation Manager, we can finally write sophisticated interactive graphical applications that don't bring the system to its knees.

Well.....maybe OS/2 doesn't really warrant this kind of purple prose; after all, it's really just another in a long string of operating systems, and like all the others, it will probably cure some old problems and create some new ones. So, as an application programmer, you'll be better off if you don't completely convert to the *OS/2 religion*. Learn as much as you can about it, develop an understanding of its strong and weak points, and understand how it relates to its predecessors, especially DOS and UNIX. But remain a free thinker, because OS/2 will not make these other systems obsolete. Furthermore, you'll probably see an even better operating system during your career. Perhaps it will be called OS/3.

This is not the first OS/2 programming book, and it certainly won't be the last. Are these books necessary, or should you just buy the *OS/2 Technical Reference Manual* from IBM or Microsoft? Well, there are several texts which, for the most part, duplicate information that can be found in the reference manual. I suppose these would appeal to someone who doesn't want to lay out $200 for the manual.

Other books, most notably *The OS/2 Programmer's Guide* by Ed Iacobucci and *Inside OS/2* by Gordon Letwin, were written by members of the OS/2 design team in order to explain how and why this new operating system was developed. But while these books explain how OS/2 works, they don't tell you much about writing application programs for the new operating system. Iacobucci's book does contain a thorough treatment of the OS/2 Application Program Interface, but it concentrates on assembly language programming, just like the *OS/2 Technical Reference*.

And that's where this text comes in. Rather than present a history of OS/2 or simply summarize the information contained in the reference manual, I've tried to focus on the transition from DOS to OS/2 for programmers who are working in a high-level language such as C. Most people who will write programs for OS/2, especially in its early years, have some experience with DOS. Indeed, many of you will be converting DOS programs to run under OS/2; you're probably less concerned about using the new OS/2 features and more concerned with minimizing your conversion effort. So, I will present OS/2 in just that way. First we'll examine the features it shares with DOS and explain how to use them. Then we'll see how to improve your DOS programs by making use of new OS/2 features such as multitasking.

Here's a brief summary of the book's contents:

- Chapter 1 explains why OS/2 was developed – basically a lengthier treatment of the preceding paragraphs.
- Chapter 2 explains the different ways that DOS and OS/2 utilize the Intel processors.

- Chapter 3 gives a quick overview of the OS/2 feature set.
- Chapter 4 compares the DOS and OS/2 Application Program Interfaces.
- Chapter 5 gives some guidelines for converting existing DOS programs to run in OS/2 protected mode.
- Chapter 6 explains the OS/2 File System, which is very similar to that of DOS and also has some useful new features.
- Chapters 7, 8, and 9 describe the OS/2 techniques for accessing the screen, keyboard and mouse, which are much different and, in some ways, better than the DOS techniques.
- Chapters 10, 11, and 12 discuss OS/2's multitasking, memory management, and interprocess communication features. Then Chapter 13 covers some advanced I/O topics not discussed in Chapter 6.
- Chapter 14 covers the OS/2 features that support country-independent programming.
- Chapter 15 describes how to use the OS/2 "family-mode" capability to produce software that can run under DOS or OS/2 and utilize the best features of both operating systems.

The Appendix is a quick reference to the functions that constitute the Application Program Interface for IBM OS/2 Standard Edition 1.0, which is essentially identical to Microsoft OS/2 Version 1.0.

You should have no trouble with this book if you are reasonably familiar with DOS and the C language, and if you are at least comfortable with short stretches of Intel assembly-language code. Those who do not know DOS and/or C may have a harder time, but none of the examples are complicated or arcane, and so you should get in the groove after the first few chapters.

One area that gave me some trouble was the representation of hexadecimal numbers. In C, the hexadecimal form of the decimal number 256 is 0x100, while the assembly language form is 100H. I finally decided to use both formats, depending on which one seemed to make the most sense at the time. In many cases, such as when writing addresses in the *segment:offset* or *selector:offset* form, I write the plain hexadecimal numbers with no prefix or suffix, since the meaning is clear.

Incidentally, although neither Microsoft nor IBM like to state this directly, C is the preferred language for programming high-performance OS/2 applications. If you review the OS/2 reference manuals, you'll find a lot of C-related examples and jargon, and the OS/2 Presentation Manager can't really be used effectively from any other language. So if you are not already a C programmer, I hope this book will help you get your OS/2 "C legs."

Table of Contents

Chapter 1

The Need for OS/2

1.1 The Personal Computer Revolution

Odd matings often produce interesting and unexpected results. In 1981 IBM, one of the world's largest and best-managed companies, joined with Microsoft, a group of brash young microcomputer hackers, to produce a simple personal computer and disk operating system. Who could have predicted that this coupling would give rise to a multibillion dollar PC hardware and software industry in less than five years? Or that this computer architecture would be so resilient that it would find use in every facet of domestic and international business? Or that the IBM PC and its clones would almost totally eclipse the then-dominant CP/M, Apple, and Commodore computers?

Those early PCs were small machines by today's standards – an anemic 8088 microprocessor, less than 256 kilobytes of main memory, one or two low-density floppy disks, and only a few I/O adaptors. But the PC and DOS designers at IBM and Microsoft were either extraordinarily foresighted or unusually lucky, because their basic design decisions survived more than five years of rapid change – the so-called *personal computer revolution*.

Today's PC is likely to sport at least 640 kilobytes of memory, a high performance 80286 or 80386 microprocessor, a hard disk storing at least 20 megabytes, and a multitude of I/O devices. In addition to the keyboard,