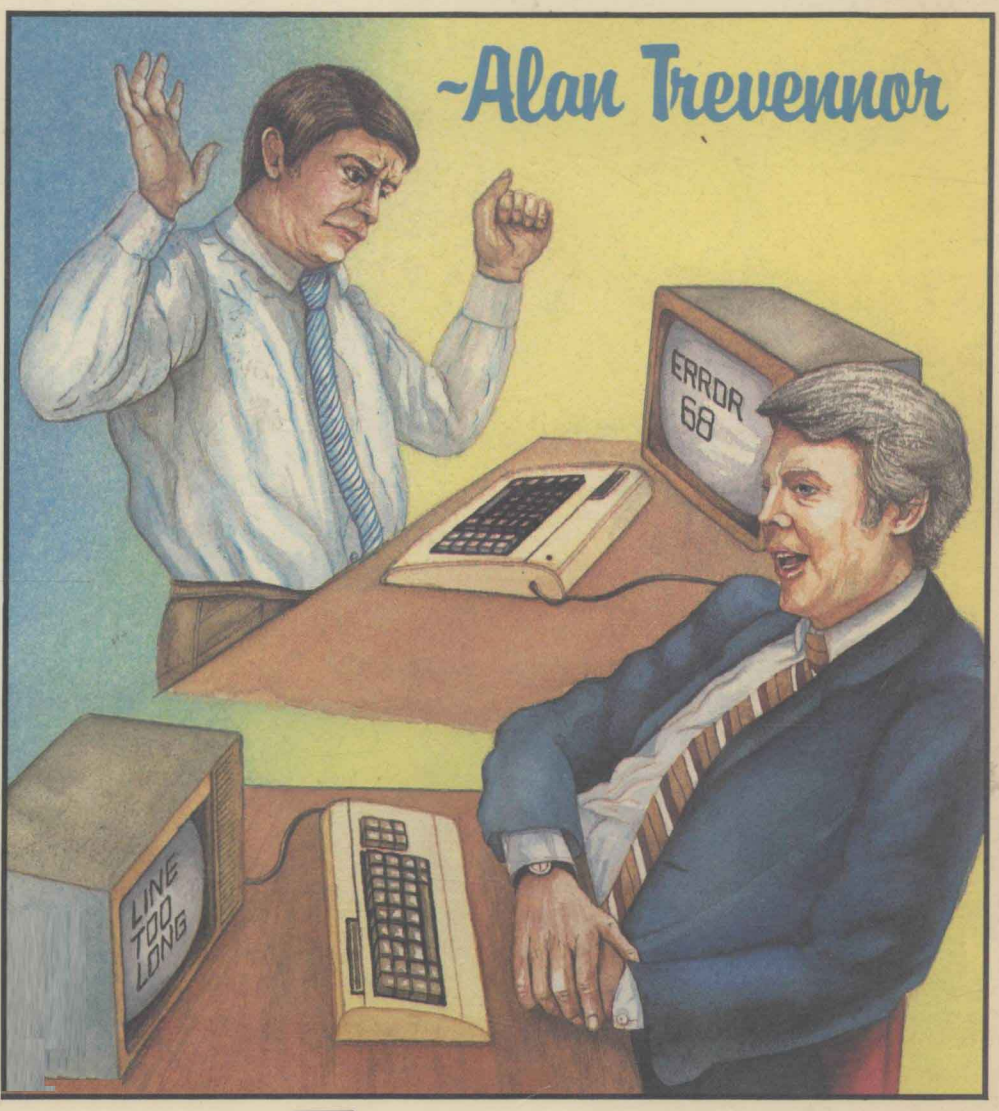
Operating Systems

-a user friendly guide



-Alan Trevennor

Sigma Technical Press

Operating Systems: a user-friendly guide

Alan Trevennor

Sigma Technical Press

Notice: Some material contained in this book has been adapted in part from publications of Digital Equipment Corporation. The material so published herein is the sole responsibility of the author of this book.

ISBN: 0 905104 66 8

Published by: Sigma Technical Press
5 Alton Road
Wilmslow
Cheshire SK9 5DY, UK.

Distributed by: John Wiley and Sons Ltd
Baffins Lane
Chichester
Sussex PO19 1UD

Printed and bound in Great Britain by
J. W. Arrowsmith Ltd., Bristol

This book is dedicated with love to my wife Wendy, who suspended her incomprehension long enough to translate and type the text.

PREFACE

I wrote this book because, three years ago when I looked for it, I could not find it.

This book is all about computer operating systems, from the non-software person's viewpoint. The really unusual thing about it, however, is that I have used the three Digital Equipment Corporation operating systems, RSTS, RSX and VMS as sources for examples. Every book I could lay my hands on three years ago used IBM or ICL or some other company's operating systems in examples. Since that point three years ago, I have conducted a lot of practical, hands-on research into these particular DEC operating systems; not at a MACRO decoding level, but by daily usage of them and by probing into the way they implement the facilities they provide. All this time I have been (and am) a working hardware engineer and an enthusiastic – through struggling – BASIC programmer.

I have often heard it said that DEC RSTS (pronounced Ristus and short for Resource Sharing Time Sharing) is the most widely installed minicomputer operating system in the world. How true this is I cannot say, but it must surely be close to it. There will, therefore, be many thousands of people using it every day in many parts of the world. In addition to those systems running RSTS, the number of machines running RSX (Resource Sharing eXecutive) is also large, especially in the scientific and educational fields. The newer VMS operating system, running on DEC's range of VAX processors, will surely overhaul RSTS in the popularity stakes at some stage.

The number of people involved with these three operating systems is therefore large. It is to these people that this book is offered as a crystallisation of the things they already know, or suspect, about operating systems.

This is what I meant when I said I could not find this book. I make no claim to being the ultimate expert on the subject, but merely pass on what I have learned as an appetite whetter for the reader to use as a springboard to more academic and detailed works, after reading what is, in effect, my written-up notes.

A. Trevennor

CONTENTS

INTRODUCTION 1

Sources for examples 1
Assumptions about the reader 1
Reading Order 2

CHAPTER 1
MEET THE SYSTEM

1.1 Major Operating System Components 3
1.2 Hardware Interface and Management 4
1.3 Operating System Users
– the major groups and their needs 5
1.4 Operating System 'Musts' 6

CHAPTER 2
MAJOR OPERATING SYSTEM COMPONENTS DESCRIBED

2.1 The File Processor 8
2.2 I/O Drivers 12
2.3 The Memory Manager 22
2.4 Language Support 27
2.5 The Job Scheduler 30
2.6 Command Interpreter 32

CHAPTER 3
BREATHING LIFE INTO THE BEAST

3.1 A Tug at the Bootstraps 35
3.2 How VMS is bootstrapped 37
3.3 How RSX is bootstrapped 39
3.4 How RTS is bootstrapped 40
3.5 Some general points about bootstrap
or initialisation failure 42
3.6 Methods for isolating boot failures 48

CHAPTER 4
INTRODUCTION TO AND REASONS FOR HAVING FILE STRUCTURE

4.1 RSTS File Structure in Detail 52
4.2 Other Methods of File Structuring 68
4.3 File Structure Maintenance 69
4.4 Clusters 76

CHAPTER 5
THE ERROR OF OUR WAYS

5.1 Error Logging – How it functions 80
5.2 Error Log Examples 83
5.3 Analysing system crashes
determining their causes 88

CHAPTER 6
"ON MY LEFT, THE SOFTWARE...ON MY RIGHT, THE HARDWARE"

6.1 Buffering of Information 99
6.2 Cacheing – Its effect on System Throughput
and Hardware Usage 103
6.3 Spool Programs 106
6.4 Setting Up the Job Priorities 108
6.5 Language Support 110

CHAPTER 7
THE MYSTERIOUS CASE OF THE SLOW SYSTEM

7.1 System Monitoring 112
7.2 The System Monitor Programs 113
7.3 Using the system monitor and statistical programs
to tune systems 121
7.4 Case Study: RSTS QSTATS 125
7.5 Altering systems to make them
run more efficiently 130
7.6 System alterations: other considerations 139

CHAPTER 8
GETTING PERSONAL 142

APPENDIX 1 148

APPENDIX 2 150

APPENDIX 3 152

APPENDIX 4 154

GLOSSARY OF TERMS 157

INTRODUCTION

This text is intended as a guide to the simpler aspects of computer operating systems, and is biased towards the non-software oriented computer person. The book will give an insight into the internal workings of the operating systems – without going into the fine details or getting down to the actual program listings for any one operating system. Many examples are used to illustrate the general information presented, and the actual operating systems from which these examples are drawn are listed in the next paragraph. So why do non-software people need to know about operating systems? Anybody who uses a computer uses an operating system of some kind. As most failures on systems occur when they are running an operating system it is of great value to have a clear idea of how the operating system works. This will enable the reader to talk on a higher level to system managers, who are very often heavily software oriented. The reader will also find it easier to talk about operating systems to software people. And last but certainly not least, the subject is a very interesting one and, at this level, not too difficult to understand.

Sources For Examples

The examples throughout the text are drawn from the following operating systems, all of which are products of DEC (Digital Equipment Corporation):

1. RSTS/E = Resource Sharing Time Sharing / Extended. (Examples use version 7)
2. RSX = Resource Sharing Executive. (Examples use version 4)

3. VMS = Virtual Memory System. (Examples use version 3)

The reader should not infer, except where specific attributed examples are given, that the text describes exactly any or all of these operating systems; nor, in the case of the "average" hardware described, should it be thought to be an average solely of DEC hardware. Rather, an average of many operating systems and many

manufacturers' hardware has been attempted with examples to clarify the information presented, being given from the above sources. These three operating systems were chosen because of the author's familiarity with them, and because of their very widespread use throughout the world.

Assumptions About The Reader

Obviously you should be used to using an operating system, preferably one of the three used in the examples. You should be familiar with computer hardware terminology, although a glossary is included of the general computer terms used in the text. The only other assumption made is that you have a need to learn more about operating systems.

Reading Order

It would probably be best for readers with anything less than a rudimentary knowledge of operating systems to begin at the beginning and work through. Readers who do not fall into this category can probably skip chapters one and two, and start at chapter three, as they may already be familiar with the ideas introduced in the first two chapters, and can in any case refer back to them if, in the rest of the book, they discover some concept which they do not fully understand.

CHAPTER 1
Meet the System

So now we arrive at the inevitable question, "What is an operating system?" It may be defined as a set of software routines for the management of a computer's hardware resources and the orderly and effecient storage of its user's data. But perhaps for our purposes a better definition is as follows: An operating system is something which enables us to type in, "run edit", instead of this kind of thing: "Load disk 0, cylinder 10, sector 0, for 19 blocks, and load it into memory, starting at memory address 27344, unless that would involve overwriting somebody else's data, in which case try location 454435, unless..." A frivolous example, since any operating system which could understand natural English would be very clever indeed! But the example is valid, as it indicates the amount of "behind the scenes" work which an operating system must do, in order that we can use a computer with concise, predefined commands instead of worrying about how they are implemented.

As computer users we use many operating system functions. For example, its I/O handling capabilities, its various management facilities and its statistical gathering capabilities (error counting, and so on).

Let us now do a short list of the things inside the modern operating system. If the names mean little or nothing to you, do not worry. They will all be elaborated upon in chapter two, and the relationship of each to the hardware explained.

1.1 Major Operating System Components

1) I/O Section – the actual routines which make the hardware peripherals do what is required of them.

2) File processor – the body of software which controls and vets

information held on the system, the format in which it is held, and the general flow of information to and from storage.

3) Memory manager – on more recent systems some of this task is delegated to hardware which is external, and thus transparent to the operating system, but traditionally the memory manager allocates parts of memory to various parts of the operating system on an as-needed basis.

4) Language support – provides support from the operating system to high level programming languages like Pascal, BASIC, FORTRAN, C, etc.

5) The job scheduler – this component varies from system to system, but basically decides which one of all the users currently connected to the system is going to have the next brief period on the processor.

6) Command interpreter – interprets commands from user.

Everybody you ask will give you a different list, but the above cover the major areas of the typical operating system.

1.2 Hardware Interface and Management

The section of the operating system which deals with hardware resource management will obviously vary greatly from machine to machine, partly because of the difference between the machines they run on, and partly because of the degree of control required. An example of the hardware interface and management section of any operating system running on DEC's 11/34 processor, such as RSX or RSTS, is the software to run the memory segmentation hardware which is used to increase the memory size above that which could normally be addressed with a 16 bit address bus. The hardware interface and management section depends upon the methods used to run the hardware. An example of this is hardware-interrupts; RSX being a non-interrupts driven system, would require less of this kind of software than RSTS, which is interrupt-driven. If the definition of the hardware management section of the operating systems is extended to include things like the hardware clock or the power fail detect mechanisms, then there is a fairly wide spectrum of operations covered by it. Broadly speaking, controllable bits of hardware which are not ruled by an I/O driver will probably fall under the control of the hardware management and interface software.

Facilities Provided

A modern multi-user operating system takes a huge amount of time, money and effort to produce. It costs more of the same to maintain, extend and update to new versions. Accordingly, at the early stages in planning a system, there must be a fairly good idea of what type of computer users are likely to buy it. There are many categories of users, all with different, often conflicting, requirements. Some of the major groups are listed below, along with the sorts of things they are likely to need from the operating system they buy.

1.3 Operating System Users – The Major Groups And Their Needs

1) Commercial users – fast access to large bodies of simple statistical and textual information, plain, easy to understand system commands and procedures and error messages, easy modification of programs written in high level languages, as little involvement as possible in machine specific details, maximum assistance to hardware maintenance engineers to obtain fast repair times, information security between users and outsiders.

2) Scientific users – easy attachment to the operating system of specialist devices, easy interchange of information between users, great mathematical precision, good handling capability of large array types of data, easy modification of the operating system code, mostly: speed of processing secondary to absolute accuracy.

3) Educational users – good small file handling (10 classes of 20 students, all with a ten-line program!), on-line help messages, error messages must be clear and concise, good support of the more popular high level languages, must be good at terminal I/O and provide fast response times under a heavy terminal load, must have good security between users to prevent student experiments or mishaps seriously jeopardising other users' data or the operating system itself.

It should be obvious from the list that no one operating system can hope to satisfy all these requirements; for example, RSTS may satisfy 1 and 3, RSX may fulfill 1 and 2, but in general no one system will be totally satisfactory for ANY application. Listed below are the kind of things which will decide the customer to whom it will be attractive.

1.4 Operating System 'Musts':

Terminal handling.

Hard copy print-out capability (i.e. must provide an ability to drive printer).

A set of error messages.

A set of predefined commands.

Some kind of documentation for use in training – the fuller the better.

A set of minimum utility support programs to implement these functions.

Most important to the vast majority of users is the ability to store and retrieve information from attached peripheral storage devices.

Market deciding factors: operating system additions.

Statistics gathering (a great help in obtaining and maintaining system efficiency – see chapter seven).

Ease of expansion of hardware (a lot of people discover to their cost that the operating system they bought a year ago is only able to run five terminals, and that they now need seven).

The degree to which the system may be tailored to specific installations requirements.

Support for very large file sizes (data base users would be attracted by this feature).

A foolproof data protection system between users, or between users and the outside world.

Command style – an operating system where the command syntax is variable may not be a good selling point to the inexperienced or infrequent user.

User friendliness – this is the industry term for the quality of the terminal user-to-operating system interface; where this is good an operating system is said to be "user friendly".

Simplicity (or otherwise) of system procedures.

Good mechanisms for passing messages between user terminals (e.g. the MAIL utility of VMS or the TALK program of RSTS).

Comprehensive logging of hardware errors – this factor decides to a large extent how long a system takes to fix for all but the most obvious faults.

A cacheing capability (see chapter six).

Portability – it is nice (though not common) to have an operating system which can be translated between different manufacturers' machines, and look reasonably similar once this is done. Operating systems like UNIX are the exception to this rule.

This list is not complete, since a whole range of decisions need to be made about the high level language support: to what degree should the language system do its own I/O, what measure of autonomy, as regards access to system facilities, shall the language systems be allowed, and so on. As I have already stated, an operating system is a very complex and expensive thing to produce, and must be aimed carefully at its market. The brief history of the computer industry is littered with the corpses of expensively produced operating systems which aimed for the mass market by pleasing everybody and ended by pleasing nobody, as in the fable. The decisions taken at the design stage are thus very important with regard to its success – along with other factors such as which machines the product is to run on, marketing and publicity efforts, after-sale support, and so forth.

CHAPTER 2
Major Operating System Components Described

2.1 The file processor.

A file processor performs a great many functions, almost all of them highly complex. Its primary purpose is to process the retrieval and the storage of information (files). And indeed this is the component which imposes and maintains the file structure. (File structures are introduced in chapter four.)

The operations of the file processor cover a wide range, but let us start with an example drawn from RSTS. Any user who types a command to run a program, or perhaps load a file from disk for listing or modification, or any other operation which requires transit of information from the file storage to the user's area of memory will have his request formatted into an entry in the file processor's request queue. After all other entries in the queue above it (or with higher priority) have been serviced, it will be serviced itself. First the file processor will decode the operation to be performed, and break its execution up into a number of steps for completing it. For our example: the loading of a file into memory. The steps would look like this:

a) Perform a verification of the file-name supplied. For various reasons most operating systems disallow certain characters in file names. As an example, try typing the command "old FIL?23" to RSTS. The result is, because the question mark character – ? – is not allowable in an RSTS file-name, the error "illegal file-name" is given in response. This

occurs before the file processor even accesses the directory to see if the file exists.

b) The directory belonging to the user (or an alternative one which he may have included in his command) is located and searched for a file whose name matches the one in the command. If the file processor finds the match it is looking for, then the information about the file which is stored in the directory along with its name is passed on to the next stage. If the file does not exist in the specified or default directory the user is informed that the file processor "Can't find file or account". Note that this error also occurs as its text says if the specified account directory cannot be found.

c) Using the information about the file passed on from the previous stage the file processor now sets about converting the information into a parameter block for an I/O operation, subject to the proviso that the user has not tried to load a file which he has no access rights to, or that another user has not already got the file open for writing new information into it. If either of these conditions are met the error "Protection Violation" occurs, the file processor goes on to its next request and our user must remedy the situation, perhaps by logging into another account from which he can access the file or by waiting until the current access to the file is completed, then try again.

The format which I/O drivers require for their parameters is usually quite strict. It has to be, otherwise the I/O driver would be far bigger than it needs to be. The file processor knows this format and converts the information from the directory into a set of or maybe several sets of parameter lists to be put in the I/O driver's parameter area or queue. The descriptor information contained in the RSTS directory is listed and explained in chapter four.

d) When the file has been loaded into memory by the I/O driver the driver signals the file processor that the job is done and the file processor then ceases its involvement. The user's statistics must be updated to show what he is running. The program is now ready to run.

The same kind of sequence will be executed when a user program requests some data to work on, or requests that a file be updated are encountered – though, of course, when a file is to be updated then the directory information must also be changed.

I have grossly simplified the steps in the example above, since it makes no mention of the run time system interaction which occurs in almost every case, nor does it take account of any cacheing schemes