The Minimum You Need to Know

to Be an OpenVMS Application Developer

By Roland Hughes

Logikal Solutions

Printed and bound in the United States of America

ISBN 0-9770866-3-1
ISBN-13 978-0-9770866-3-4

This book was published by Logikal Solutions for the author. Neither Logikal Solutions nor the author shall be held responsible for any damage, claim, or expense incurred by a user of this book and its accompanying CD-ROM as a result of its use or reliance upon the contents contained in either the book or the CD-ROM.

Acknowledgments

A book of this nature comes about as a result of a nearly 20-year journey through the career of software development. During the course of that journey your knowledge and skill are honed by the countless individuals you come in contact with either personally or electronically. Some of them you don't remember as time goes on, yet you do remember the things they taught you.

One definitely must thank their family for allowing them to pursue a passion and dream even though they didn't really understand the passion or the dream. At some point it is possible that I didn't fully understand it myself.

In particular I would like to thank Mr. Ken Olsen, the founder of Digital Equipment Corporation. Your company put out the greatest operating system known to mankind. Thanks for the memories.

Source Code Licence

Any person owning a copy of this book may use the source code from this book and accompanying CD-ROM freely when developing software for their personal use, their company's use, or their client's use. Such persons may include the source code either modified or unmodified provided that the source delivered makes reference to the original author and is delivered as part of a fully functional application. It is expressly forbidden for anyone to post this software on any bulletin board system, internet Web site, or other electronic distribution medium without the express written consent of the author. It is also expressly forbidden to sell this source as part of a library or shareware distribution of source.

Users of the source code contained within this book and on the accompanying CD-ROM agree to hold harmless both the author and the publisher for any errors, omissions, losses, or other financial consequences which result from the use of said source. This software is provided "as is" with no warranty of any kind expressed or implied.

Table of Contents

Introduction

I.1 Purpose of This Book

The purpose of this book is to assist developers switching platforms to OpenVMS. It is also designed to be useful to college level students who have had at least one of the programming languages covered in this book as course work. Anyone who is currently successful at writing applications on the lesser platforms, namely Windows and UNIX, should be able to use this book, and in a relatively short period of time, become productive on the much more robust and stable platform of OpenVMS.

There are quite a few third-party books that have been written over the years on/about the OpenVMS platform. Those books go into great detail about very specific areas of this platform. What you should glean from this book is a very broad understanding of development tools and techniques you will encounter when working on the OpenVMS platform. While we will delve into some of the obscurities a developer needs to be aware of as we cover each topic, we will not cover all of the minutiae.

This book is not intended to be the last and final word on OpenVMS development, nor is it intended to replace the dozens upon dozens of manuals written for OpenVMS by its owners. (Currently HP at the time of this writing.) You should know enough when you finish with this book to both not be afraid of development and also know where to look for further information. Upon completion of this book you will have more than enough knowledge and skills to become a maintenance programmer at many shops running OpenVMS.

I.2 What You Need to Know to Read This Book

This book is intended to be used by applications developers, consultants and to some extent, systems analysts. You should have had at least one course in logic and be comfortable with at least one of the programming languages covered in this book or a language very similar to it. Managers will find much of the information useful when making technical direction decisions. Those developers whose only language is “Visual something or other” didn’t learn a language and will flounder miserably with this book.

I.3 Who Should Read This Book

Anyone who plans to become employed or a consultant using the OpenVMS platform should both own and read this book. No matter how seasoned you are on a platform, even if OpenVMS is your current platform, it is nice to have a cheat-sheet to flip through when the mind gets a little foggy. As I get older, I find I cheat more when having to go back and maintain some of the really old stuff at client sites. Hence, it was part of the incentive to write this book.

I.4 How to Read This Book

This book is meant to be read from front to back initially, then serve as a reference manual on your desk. Each chapter builds upon the previous chapter. Once we talk about DCL in Chapter 2, we use it and talk about it throughout much of the book. Likewise, once we talk about BASIC, we will continue talking about it most of the way through the book.

Some books cover only one topic like a single language. They bombard you with the mass of the syntax in such a general way as to be useless when taken in the context of any single platform. This book is not a dry and sterile treatise on the syntax of languages. This book starts you out with the minimum you need to know just to log on and use the platform. Then it builds one application teaching many different aspects of the platform as you need to know them during the course of the development. Where appropriate, the historical reasons for the way things are get passed along as well.

It is the hope of the author that using a single application, which requires all of the basic skills needed to be a maintenance developer on this platform with each of the languages covered, will teach you more than just those skills. Following the journey through to the end will also give you an idea of the trade offs you make using each tool. There is no one tool that is perfect for all jobs. There are many reasons you will find shops using 4 or more languages. Sometimes they bought a package written in a language they currently weren't using, other times they wrote the package themselves in that language or tool because it was the best tool for the job even if they had to encounter a learning curve to make it work.

At the end of your journey is a reward. The chapter titled "Ruminations and Observations" is a series of essays about IT topics and life in general. This chapter is my reward for writing this book. Some of the sections there may offend you beyond any scope of reason — good. Others may provoke thoughts which keep you lying awake at night trying to figure out a solution — also good. The topics I choose to cover in this chapter are problems and concepts which must be addressed in IT for the good of the industry. How they actually get addressed will be up to you the reader and you the voting shareholder of corporations. There are sections in that chapter covering information gathering, optimal technology, the reason Y2K happened and off shore computing.

I.5 Our Sample Application

We will have a single sample application redeveloped in all languages. I'm not going to bore you to tears with a contrived and hokey inventory or order entry system which seems to be the bane of academia. Let's face it, we have all read books like that and we never spent any time after we finished the book playing with the application because it was boring. I'm also not going to create a useless paint application that you will find touted in almost every GUI platform book on the market.

Our sample application will track the Mega-Zillionare lottery. No, that is not its official name. There are several different multi-state lotteries with similar rules and official Web sites where you can download data. The test data for this application is provided on the CD-ROM and is DRAWING_DATA.TXT. Once you are comfortable with using the various editors, you can download and format your own bulk data file for import.

Initially, we will use RMS indexed files to store the data. One for the drawing data, and two for statistical data. Don't get nervous; we will only be doing some crude stats on the data. Mainly, we will count the number of hits, percentage of hits, sequence of hits and the average and maximum of misses between hits. Calling it statistics is probably being too grandiose.

The input data file is a CSV (comma separated values) formatted file. It contains one record per line in the format: drawing date, number 1, number 2, number 3, number 4, number 5 and the mega number. The primary indexed file will be laid out in much the same way.

We will have the following main components in each application:

- Import program to create a new data file from a bulk import file.
- Stats generation program to create two new indexed statistics files.
- Report programs to print off numbers based upon their statistical ranking.
- A form of data entry to add additional drawing data on a single record basis.
- A browse program to scroll through the drawing data.

As you can see, this little application will encompass all of the core functionality you need to know to begin being productive with each tool set described. By redeveloping the same application with different languages and different tools, you will get a feeling for the design tradeoffs each tool set has. When we get to the relational database chapters, the indexed files will be replaced by tables. Hopefully, you will experiment yourself with each of the tool sets to find improvements to make the application more to your liking. As I said, I've chosen an application you might actually wish to play with if for no other reason than the new data is available every week.

As stated, we will have 3 indexed files for this application. The first will be an indexed file containing the drawing data, the second will contain stats for the main drawing numbers, and the third will contain stats for the mega number. Below are the file layouts. The tag of "k0" to the right of a field is my way of flagging the primary key.

Drawing_Data

```
Draw_dt      char[8]  k0
No_1         integer
No_2         integer
No_3         integer
No_4         integer
No_5         integer
Mega_no      integer
```

Drawing_Stats

```
Elm_no          integer  k0
Hit_count       integer
```

Mega_Stats

```
Elm_no          integer  k0
Hit_count       integer
```

```
Last_draw_no      integer             Last_draw_no      integer
Since_last        integer             Since_last        integer
Curr_seq          integer             Curr_seq          integer
Longest_seq       integer             Longest_seq       integer
Pct_hits          double              Pct_hits          double
Max_btwn          integer             Max_btwn          integer
Ave_btwn          double              Ave_btwn          double
```

I.6 Why OpenVMS?

Many of you reading this will be coming from other platforms you are comfortable with and believe they can be used to do anything. Most of these platforms haven't existed as long as OpenVMS. Few, if any, had the number of operating systems and hardware changes that culminated with OpenVMS. Near the dawn of the computer age (you know you are really old if you can remember the dawn of an age), some of the first platforms made by Digital Equipment Corporation (DEC) were the DEC-10 and DEC-20 time-sharing computers. I never had the privilege of working on those platforms but have talked with some who did. They performed exceptionally well for their era running TOPS-10 and TOPS-20 operating systems.

Later, DEC came out with the PDP line of computers. These computers ran various operating systems: RSTS/E, RSX-11 and RT-11 among them. (Those coming from the UNIX side of life may also remember that UNIX was originally developed on the PDP hardware platform.) Each of the previously mentioned OSes still have their followers today. Though the job opening advertisements for those operating systems may be few and far between now, you will still find some PDP hardware running today. There are even some manufacturers making PDP-11 emulators to use inside various other boxes. Each of the operating systems was designed to handle some niche applications incredibly well; so well that some companies have opted not to replace them as long there is some method of keeping them running.

When DEC came out with the VAX platform, they had over a decade of seasoning and think tanking by some incredible minds behind them. Both the platform and the operating system were designed from the ground up to create a seamless network. Today, with the Internet available to the masses for a fee, most would say "so what." OpenVMS, however, wasn't designed today; it was originally laid out in the late 1970s for platforms selling in the early 1980s. With OpenVMS came the original definition of clustering.

Many operating systems today claim clustering capabilities and many application packages claim to be "cluster-aware" on many platforms, but once you become accustomed to the power, stability and flexibility of clustering in an OpenVMS environment, you will see just how hollow those claims really are. We won't cover much on clustering in this book, not because there isn't much to cover, but because once a system manager sets up a cluster, the application developer really doesn't care where the nodes are. When a system manager chooses to mount a disk drive on one node as available to the cluster it looks just like a local disk drive to the application developer no matter what node they are working on. By "nodes" I'm not just talking about servers in the same building. These machines can be anywhere in the world. I have personally consulted for companies that

had nodes scattered across the U.S., England, Ireland, Germany and other countries. From time to time you may see some lag accessing one of the drives due to congestion on the network, but you don't access them any differently.

Batch queues, print queues and ACMS (Application Control and Management System) applications can be defined so that any node in the cluster can use them without the faintest idea of where they are. Admittedly, you might want to know where the printer actually is that is servicing a print queue, but you don't have to. More importantly, that printer could very well be on a different continent from the cluster node servicing it. Indexed, sequential and other file types can be shared by multiple users from any node in the cluster.

The Records Management System (RMS) on OpenVMS provides a feature developers can choose to use called RMS Journaling. This works in unison with a component/product called DTM (Distributed Transaction Manager). DTM keeps everything in sync and rolls partial transactions back. This allows a developer a great amount of freedom and security. You can have a single transaction that encompasses multiple RMS Journalled files and multiple RDB tables and know profoundly that nothing will be updated until you actually commit the DTM transaction. If the power grid drops on your machine while your program is running, when the OS comes back up and starts the DTM portion, DTM cleans itself up. If you develop your application correctly, you can simply restart the application once the system has come up and rejoined the cluster.

OpenVMS doesn't suffer from the e-mail and other viruses which plague other platforms. It has passed numerous DOD security tests. Some hacker conferences have even declared the OS unhackable. While I would never go so far as to make that claim about any OS, coming from groups like that it is quite a statement.

A concept that was pioneered on the PDP platform has become a force of unprecedented power: Logicals. We will cover logicals in Chapter 1. They are a truly amazing creation which provide security, simplicity and complexity all at the same time. A person could write almost an entire book just covering the concept and implementation of logicals. This book will give you enough understanding about them to both use and respect them.

I.7 The Definition of Application

For the purposes of this book and the possible follow-up book, I am defining application as a single hosted program or set of programs running on a single cluster. While many people would call these "systems", I do not. In my view, "systems" development requires a data interface to the outside world.

While one could make the argument that an application that receives tapes loaded via "sneaker net" is a system using a tape interface, in today's world those types of systems don't usually exist. (Sneaker net refers to operators wearing sneakers running around a computer room gathering printouts and handling tapes.) Systems in today's world interface via a communications library (MQ Series, Tibco, DECNET, Tuxedo and, if desperate, TCP/IP) to non-homogenous platforms. They feed each other in real time or batch mode. Their function and failure is visible outside the company. Application failure is only visible inside the company.

Chapter 1

Fundamentals of OpenVMS

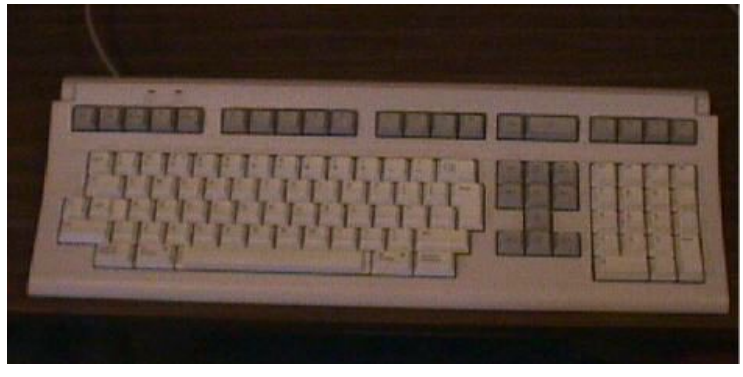
1.1 Hardware

Covering all the hardware produced by DEC and third-party companies over the years would be a book of massive size and provide you with little information needed by an application developer. What an application developer really needs to be aware of is the terminals they will log in with. Occasionally, you will need to know how to hit the OFF-LINE button on a system printer, hit FORM FEED a few times, then hit OFF-LINE again to put the printer back on-line, but those buttons are clearly marked on all printers, so I think you can figure it out.

Many types of terminals exist. The DEC series of terminals are referred to as VT terminals for Video Terminal. (Some say it stands for Video Tube as all such devices were called “tubes”; others say it stands for Video Text but many of these terminals handle graphics.) Today, all standard terminal models have 3-digit model numbers. The 4-digit numbers are generally VXT terminals for multi-session X window type interfaces. The granddaddy of them all is the VT-52. This was a massive black and white (monochrome) terminal normally communicating at a whopping 150 or 300 baud working in uppercase all of its life. Yes, they could communicate faster, but the cable length had to be very short. There were older terminals and paper consoles (yes, a printer with a keyboard built into it), but the VT-52 seems to be when things were finally catching on. The VT-52 definition for the most part is the ANSI TTY definition with added formatting controls for cursor movement. It's replacement, the VT-100, became the ANSI standard terminal and is supported by a plethora of non-DEC operating systems. My first exposure to it was on a PDP-11/70 running RSTS/E. This was also my first exposure to DEC equipment.

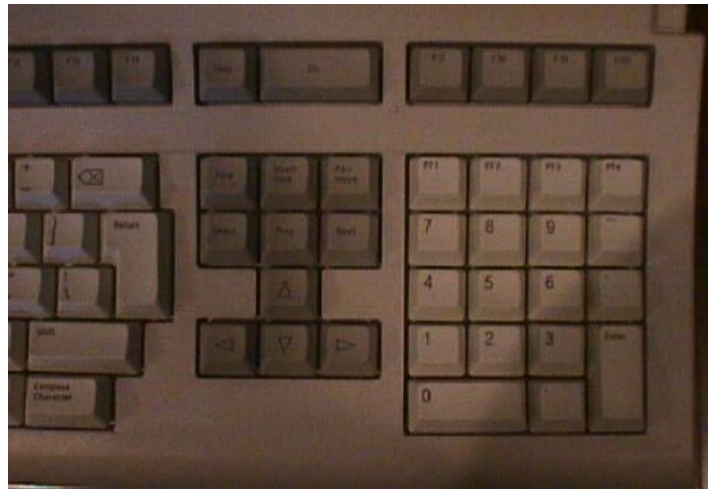
Today, regardless of the VT model, it will generally support VT-100 escape sequences and functionality. Some newer models only go back as far as VT-220, which was one of the first models supporting the enhanced keyboard used by later DEC terminals. Below is a picture of the DEC style keyboard attached to my Alpha machine.

Of special interest to us is the numeric keypad on the right hand side. When we get to the section on using the various editors available on OpenVMS, you will see that they are the primary interface.



Terminals are going the way of the abacus though. Most places today purchase a VT terminal emulation package for a personal computer. I use one almost exclusively on my notebook computer when traveling to client sites. The advantages of the emulator are many. Most notably you can communicate over the office network at full network speed. You can have multiple sessions opened on multiple machines and use your mouse to paste between terminal windows. Emulators also give us the ability to paste to/from PC- based applications into the terminal. Above all, you can customize the colors almost anyway you wish to make your computing experience enjoyable.

Notice the keyboard layout. The numeric keypad on the right has 4 PF keys across the top and 3 keys down the right side below them. PC-based keyboards are short one key in this area which can make for some interesting problems. The numeric keypad is how you navigate through the different text editors available on OpenVMS. When we get to the section on using the EDT version of the editor we will cover this in significant detail.



Another point worthy of note are the [HELP] and [DO] keys at the top. In the TPU version of the editor (and many OpenVMS applications), these two keys are used heavily.

The keys [F1]. [F2], [F3] and [F5] are quite significant. [F1] is the Hold key. This stops the terminal scroll. You hit it again to turn scrolling back on. *It does not stop an application that is running or turn off output*. This key is also mimicked by <ctrl><S> (hold) and <ctrl><Q> (end hold) combinations. [F2] is the screen print key. If your terminal has a printer attached to the back of it (or a network printer setup via the SETUP key), when you hit this key it causes the contents of the screen to be dumped to the printer. You can turn on a scrolling print by pressing <ctrl>[F2] and turn it off by hitting the same key combination again. [F3] is known as the SETUP key. When you hit it, one or more menus will pop up that you can navigate through setting all the possible configuration parameters for your terminal. There are too many to go into here and they vary terminal to terminal. Basically, all aspects of how the terminal communicates with the outside world and appears to you are controlled here.

[F5] is a key of special importance. Unless your terminal is hooked up to a LAT or is a multi-session enabled terminal, you should NEVER hit this key. When the terminal is hardwired as the console for many systems (central terminal used by operators on a special port on the machine) hitting the BREAK key can halt the entire system for all users. You will know you have done this when you see the following prompt: >>>. If you do not type RES and hit <return> before an internal time-out occurs on the box, you will have crashed the entire system for all users. Don't play with this key unless you know what you are doing. On old paper terminals hooked up as system consoles, the combination of <ctrl><p> used to duplicate this function. More than one operator went to hit <ctrl><o> to turn output off for something and accidentally hit <ctrl><p>.

The combination of <ctrl><o> is used to turn output off. This is not like "hold," which simply stops the scrolling so you can read it. This combination routes the output to the bit bucket until the command interpreter regains control.

Most of the remaining function keys across the top are available for application usage. Their function will vary based upon what program you are running.

1.2 Logging In

The first terms you need to become familiar with are "Logging On" and its counterpart "Logging Off." (Also called "Logging In" and Logging Out.") Logging In is the process of getting Logged In. (Sorry, I couldn't resist doing a Webster on you.) We will start with the easy scenario assuming you have a single session terminal hooked up via a serial connection. We will also assume that you have been assigned a user ID of FRED with a password of MY_PASSWORD. (Neither user names nor passwords are case sensitive in the OpenVMS environment.) Your password will not be displayed while it is typed.

```
<return>
Username: FRED
Password:

        Welcome to ....yada yada yada
$
```

The "$" is the default prompt on a OpenVMS system. Most system managers will have changed the prompt to be the name of the node you are logged into followed by "->." So you see something like:

```
Kirk->
```

instead of the $ prompt. To change the prompt so it shows your node name instead of the default $, requires the use of a lexical function that we will talk about later on. If you are exploring on your own for a while, you can type HELP LEX at your command prompt to display the lexical functions available to you.

```
$ set prompt='f$getsyi("nodename")->
LGKL1->
```