Foreword

For the last twenty-five years, HP OpenVMS™, a proven enterprise-class operating system, has been supporting customers' most demanding business-critical requirements for reliability, availability, scalability, and security.

In the last five years, Linux and the open source movement have moved from a small-system operating system on the technology fringe to a serious contender in corporate computing.

Both operating systems have much in common with UNIX and Windows servers. OpenVMS is a general-purpose operating system that has many standard interfaces and APIs. Linux is a customizable operating system that shares its APIs with UNIX. APIs common to both operating systems allow OpenVMS and Linux to interoperate easily.

OpenVMS and Linux operating in the same enterprise environment let you combine the low per-unit cost structure of Linux and its huge open source libraries of software with the proven high availability, security, and scalability and exceptional robustness of OpenVMS.

I've found this book to be a practical, hands-on manual that explains how OpenVMS system managers can integrate their systems with Linux, allowing the two platforms to work cooperatively and extend the processing capability of a computing environment. John clearly describes the features and benefits of both operating systems, so that you can evaluate which one is best for any given task. The hints, tips, and techniques you'll find here will help you create a heterogeneous OpenVMS and Linux computing environment to meet your needs today and tomorrow.

Martin R. Fink
Vice-President and Chief Technology Officer
Business Critical Systems
Hewlett-Packard

Preface

This book began many years ago when Linux was only a dream of Mr. Torvald's. Its germs were in the DECWindows Cookbook, DFWCUG Longwords Newsletter articles, and hints and techniques shared at a hundred DECUS sessions that connected OpenVMS and Unix systems together in some way over the years.

From all of these ideas, plus Linux's evolution with the number of times Linux and OpenVMS were being used in the same Datacenters, I felt a need to better understand the things that Linux and OpenVMS could do together. I also felt that if I needed to understand this type of interoperability, many other people might need the same understanding too. Today Linux and OpenVMS using Open Source programs have an identical look and feel for many of their applications and interfaces . Despite the similarity of look and feel, configurations, installations, and scripts vary between operating systems (OSs)!

This book identifies eight areas of current interoperability, and more importantly provides a step-by-step process to those areas. Products, configurations, scripts, are all in this book with one goal: To make something happen between Linux and OpenVMS. No matter if you're a VMS admin or a Linux admin, this book will allow you to enable interoperablity on both Linux and OpenVMS.

I can only add that you should learn to use the local editors, which are not covered in this book. Learn the terminal or Command Line Interface (CLI) editor like "vi" for Linux and "edt" for VMS; they are essential to any system manager's tool bag. X Windows–based editors are also available, but if you learn the CLI editor, you'll be able to work on these systems even without a windowing system (a distinct possiblity if you are doing installs or changing startup proceedures). After you've learned the editors for Linux and OpenVMS, the rest of the procedures in this book should be easy to accomplish.

You may also want to load clean versions of each OS before trying to configure the tools in this book. I used OpenVMS version 7.3-1 and RedHat 7.3 for all the procedures shown in this book. You may want to visit Chapter 2 first, to learn how to break into Linux and OpenVMS if you've inherited your systems, and if you need to reuse the existing installaitons !

This book is an adventure in Open Source, TCP/IP, X Windows, security, and much much more. I hope it broadens your understanding of both OpenVMS and Linux and helps you to understand what two great OSs can do together.

All my best,
John Robert Wisniewski
March 2003

Publisher's Note: There is considerable variation in the usage of the phrase Open Source; the majority view seems to be that both words are capitalized. However, others consistently use lower case. We have chosen to follow the more common usage and capitalize both words throughout the body of the book.

Chapter 1 — Assessing OpenVMS and Linux: The Right Tool for the Right Job

Good, Better, Best

Welcome to the book that answers the questions why OpenVMS? Linux? and just how do you get these two computer operating systems (OSs) to work together?

Why in the world would you want to have them work together? Let me try to answer that by briefly saying that there are customers who adore OpenVMS—its cluster technology and full-service support. Linux is into Open Source, with tens of thousands of coders in its court. What a wild duet! But still, they're cousins—open systems cousins.

Now, without singing the rest of the Patty Duke theme music; the two operating systems have much in common: their network interface protocols; Open Source tools; industry standards; commercial database products; X Windows; and industry-standard security tools for encryption, tunneling, and secure communications. If general tools are on both Linux and OpenVMS, why continue to use two different OSs? Simply because OpenVMS has features and abilities that Linux or any other OS would be hard pressed to deliver.

What does OpenVMS have? OpenVMS provides something that you only get with 25 years of testing and use—enterprise-class stability and reliability in almost every configuration it's used in.

What does Linux bring to the table? Today Linux is Open Source, with thousands of programmers working on new and exciting applications worldwide, sharing base source codes, and delivering low-cost tools and software that improve the entire software industry.

Together, OpenVMS and Linux provide the best of Open Source and the best of commerical applications, giving users a choice about how to deploy various computing styles: client/server, multitiered database servers, or large transaction processing environments.

Between the high end and the low end of computing, various demployment strategies are needed today and will be needed in the future. The line between OpenVMS and Linux deployment decisions should be drawn wherever it needs to be.

Solution architects and system analysts need to understand the best features of both operating systems, as well as their realistic limits, and then measure the actual cost of deployment of Linux and OpenVMS systems, utilizing the best tools for the system or

work at hand. OpenVMS can run on workstations to mainframes. Linux runs on mainframes to workstations, drawing a hard line in the sand that limits what will be deployed and limits the types of deployment an organization can deliver. Why not use the best of both worlds to solve computing problems? I know I do!

Why OpenVMS?

After 25 years in a constantly changing computer industry—spanning the PC revolution, DEC becoming Digital, VMS becoming OpenVMS, Compaq buying out Digital, HP buying out Compaq, and OpenVMS migrating from VAX 32-bit CPUs to Alpha 64-bit CPUs, as well as HP's migrating from OpenVMS to IA64 Intel's 64-bit CPU, and the advent of Microsoft products, Open Source and Linux products, UNIX products, the IBM pantheon of operating systems, and fault-tolerant products such as NSK—an industry watcher and career participant has to ask the following question: Why should companies use OpenVMS over the next 5 to 10 years? Considering that OpenVMS has the lowest cost of deployment and maintenance in the industry today, you might ask, why wouldn't you use OpenVMS? Lowest cost is one factor, but technology decisions seem to drive many deployment decisions. Let's review OpenVMS's current abilities.

- *Clustering*: OpenVMS has had the acknowledged best clusters and scaling in the industry since 1983. Its features include single system disk, single unified file system view across all cluster members (even with multiple file systems), shared tapes, shared disks, and up to 96 clustered member systems of desktop to mainframe-size systems, with as many as 32 SMP CPUs each and as much as 10 Tbytes of main memory (RAM) in the entire cluster. All systems are managed as a single system and a single work domain. OpenVMS clusters work out of the box across Ethernet, SANs, and high-speed memory channel connections, with minimal configuration and setup. You can literally add a new member system to an OpenVMS cluster in as little as five minutes (once the hardware has been plugged in).

- *Disaster-Tolerant Clusters*: Data centers can be completely duplicated (all disks, all resources, all transactions) in an active-active cluster at distances of up to 540 miles apart. (Note: Everyone else just offers a hot/warm or hot/cold site technology; OpenVMS uses both sites in an active-active configuration, which uses much less hardware then other active-passive cluster configurations—up to 50 percent less!)

- *Oracle 8*: This deploys larger, runs better, and runs faster on OpenVMS clusters. Oracle RdB continues to be one of the fastest databases in the world, fully

integrated into OpenVMS and OpenVMS clusters, with new customers every year.

- *Timeshare*: As the Internet continues to grow and server consolidation continues, systems will be measured by their ability to handle many, many small jobs in a predictable time slot while sharing resources evenly. OpenVMS and its scheduler have been providing and polishing predictable, even real-time, performance since the 1970s. With today's Web server, transaction-servers, and application-server requirements, capacity planning and predictable response for every job is already becoming a prerequisite for deploying even a prototype application.

- *Security*: OpenVMS/VMS has had less then 45 CERT security advisories in the last 13 years. (Windows 2000, 484; Linux, 546; Solaris, 490; AIX, 377 as of June 2002; http://www.cert.org/).

  Deploying other servers in your production environment, you will spend considerably more system-management dollars securing your servers and making sure all the CERT advisories are all plugged than you would if you just deployed OpenVMS out of the box! If you checked all the CERT advisories, it could take hundreds of hours just to review various security holes in other operating systems.

  Out-of-the-box OpenVMS is virtually unhackable (or so say the goons/judges from the DEFCON 9 Hacker Convention, July 2001 http://www.dfwcug.org/dfwcug_newsletters/20107.PDF).

  Kevin Mitnick, celebrity hacker, just testified before Congress that he was defeated for the first time in his life when he recently tried to break into an OpenVMS system in England (http://www.zdnet.com/zdnn/stories/news/0,4586,2454737,00.html).

- *Diicoe*: Defense Information Infrastructure Common Operating Environment certification means that Compaq/HP has signed (in 2001) an agreement with the U.S. government to support OpenVMS for the next 20 years. This allows HP to continue selling OpenVMS to the U.S. government (one of OpenVMS's largest user bases), and it must maintain support for the next 20 years on products sold to the U.S. government.

- *Shared File Systems*: OpenVMS offers integration with Windows 2000 and NT 4.0 SMB file systems via Advanced Server (code from Microsoft). From the

Open Systems (UNIX) side of the house, OpenVMS integrates with NFS (V3.0). OpenVMS can also offer a single directory, which is viewed, coordinated, and accessed by both Windows systems and UNIX systems at the same time!

- *E-business Infrastructure*: Attunity's XML and database gateways are included with the OpenVMS license. Apache, SSH, Java, Microsoft's COM (object-calling standard), CORBA (Open Systems object-calling standard), DCE, and X Windows are all included with OpenVMS with the base license.

- *Service and Support*: For a business that deploys an OS for years at a time, support for past products and previous versions of the OS is critical. HP and OpenVMS continue to support VAX systems (even though they haven't shipped a new VAX in almost 2 years!), and they support previous stable versions of the OpenVMS operating systems as far back as 10 years for customers. It's hard to imagine a 10-year deployment of our latest PC of the hour. Businesses don't enjoy changing applications that are working, and OpenVMS applications, once up and running, tend not to get taken off line, which is one of OpenVMS's most famous hallmarks.

- *High Availablity*: Many companies choose OpenVMS because it's one of the few operating systems that can deliver on its claim of 99.99999 percent (about three minutes downtime per year). If OpenVMS had just started claiming this feat, it might be suspect; but customers have been getting this much service from their machines since the early 1990s (in a properly configured and maintained VMS cluster). Rolling upgrades of software and the OS, and continuous processing for years at a time, are other milestones that OpenVMS pioneered.

After 25 years as a computer, application, and database server, very few of the other operating systems can even begin to match OpenVMS as a deployment platform. While OpenVMS will never be the best games desktop, it has already proven itself the finest, most robust, most secure, and most clusterable OS in our industry today. But wait: Don't touch that dial, there's more! If OpenVMS were just another OS with the most features in our industry, that might be good enough; but what would you pay for such a fantastic tool? Two times or four times the cost of an average UNIX server? How about if OpenVMS gave you all this capability and had the lowest cost of ownership in the industry compared with all other UNIX systems and mainframes? Well, don't believe me—check out the study Techwise Research did on OpenVMS in 2001, where they found that OpenVMS had the lowest cost of ownership of any server deployment over a five year period (http://www.openvms.compaq.com/openvms/whitepapers/techwise.html).

Tested and supported features, lowest cost of ownership and deployment in the industry, highest availability and scalability, and a 25-year track record with interoperability with all the latest Internet and e-business software and tools—that's why you will use OpenVMS for the next 10 years.

Why Linux?

When Linus Torval started his project, he dreamed of an OS unencumbered by source-code restrictions and copyrights. In just a few short years, Linux workstations and servers began to emerge as a powerful Open Source alternative to Windows desktops and many UNIX servers. Productivity tools such as Star Office, WordPerfect, and many other application ports were available. Games were also available on these strange new Open Source boxes. As the desktop functionality grew, many saw computer server functions as Linux's destiny. Linux is a workstation and a server with a graphical user interface for management (Gnome and KDE are the two interfaces of choice). With HP and others beginning to offer certification as Linux Accredited Systems Engineers, support and services are beginning to mature.

So where does Linux fit in today's IT strategy? Let's review some of its features.

- *Laptop*: With Sun's Star Office or Corel's WordPerfect 2000 suite, a Linux installation offers the users less disk space consumption than a comparable Windows OS and Office suite. Linux also offers the ability to reuse two-to-three- year-old laptops that are now too underpowered to run the latest versions of Windows products, but it can also run the latest versions of Linux and many personal productivity tools that run on the platform. For Microsoft interoperability Bynari Systems also offers bug-for-bug compatibility with Windows Exchange Clients (Outlook) and Windows-compatible Exchange Servers for Linux and UNIX systems to integrate directly into Exchange E-mail Server infrastructures. Alternatively, Linux and other TCP/IP users can use Netscape mail, if they have POP3 or IMAP servers already in place (http://www.bynari.com/, http://www.sun.com/staroffice/, http://linux.corel.com/).

- *Workstation*: What goes for laptops goes even more so for Linux desktops. A few-year-old 300–500MHz Pentium workstation or even Alpha workstation can run Linux and its applications blazingly fast. Instead of buying new workstations with 50 percent more memory and disk space, recycling a used desktop saves money and upgrades easily to Linux with as much ease of use as Windows desktops. And while there have been some demands for enhanced Linux security, it's nothing like the defense that must be mounted against the

65,000+ virus definitions that Windows programs must be defended against every day when on a public network.

- *X Window Terminal*: Linux workstations and laptops make excellent X Windows display terminals. Linux, UNIX, and OpenVMS all use X Windows and various X Windows managers and programs to redirect output to remote machines across Ethernet or other network connections. Today, Microsoft Windows has programs to perform X Server displays, but Linux has built-in X Windows security and the same X Windows features as its larger UNIX and OpenVMS cousins. Therefore, bringing all your OpenVMS and UNIX application displays down from a headless server to a single, low-cost, high-quality desktop with a great choice of graphics and sound cards is a wonderful alternative to purchasing expensive workstations. (Although, just like driving a Ferrari, once you've worked on a full-blown 64-bit UNIX or VMS workstation, it's hard to go back—no matter how "sporty" the 32-bit Chevy becomes.)

- *Server*: As a bottom-tier server (in the standard three-tier client/server model), Linux is as good or better then any single NT, UNIX or OpenVMS server. Linux servers support SAMBA for disk shares with Windows servers and clients, NFS for disk mount points for UNIX and OpenVMS machines, Java, Apache, and all the major computer languages and scripting tools. The problem is that Linux servers no matter how functional, live within mean time between failures of their hardware. Sooner or later a deployed single server will have a failure that takes out critical data or programs, and then the system will have to be restored or rebuilt. Commercial clustering for Linux servers is in its infancy, as are automatic failover environments for Linux. As support, cluster technology, and shared storage (SAN, networked storage) continue to mature, so will the use of Linux servers for data-center applications.

The Bottom Line

OpenVMS systems and Linux systems have much in common with UNIX and Windows servers, with some distinct advantages for each. Given our heterogeneous computing environments of today and looking toward the future, both OpenVMS and Linux will find a use in our data centers and professional information services (IS) deployments for many years to come—OpenVMS for its high availability and scalability and Linux for its low per-unit cost structure and its huge Open Source libraries of software.

As we move forward in this millennium, the battle cry should be for the lowest-cost, right tool for every style of computing. Between OpenVMS and Linux operating systems, a very large part of the computing spectrum is being delivered, and in a most cost-effective way. Now let's explore the types of things both Linux and OpenVMS can do together!

The rest of the chapters in this book are a series of how-to processes designed to utilize both your OpenVMS and Linux box in different ways (TCP/IP, NFS, E-mail, X Windows, and more). The procedures are easy and to the point. They should also be a good starting point for you to explore these functions for more advanced use and deployment! You won't need to read this book cover to cover; the chapters should be self-contained and, I hope, a good reference for some of your specific Linux and OpenVMS interoperability issues.

The rest of this book assumes that you have a running OpenVMS and Linux system configured with TCP/IP and a working knowledge of both OSs as a user and operator. (You'll also need system privileges on both systems; don't worry, Chapter 2 shows you how to get System and root on your boxes. But don't experiment on your production machines; two small workstations in the same network would be a great test bed as you learn to connect OpenVMS and Linux together.

Chapter 2 — Breaking into OpenVMS and Linux

Equipment for Those Who Have None

Once upon a time there were these three computer collectors—stop me if you've heard this one.

Oh sure, for a few hundred dollars you can get older Alpha 64-bit systems or Intel-based 32-bit systems that will run OpenVMS or Linux, but consider that many companies (and even some users) are trading up to the latest hardware every 18 months or less. Rather than just throwing out this older equipment, owners would rather give it to a good home, knowing someone could use older hardware for learning purposes or for user groups. IT professionals have a lot of guilt about getting rid of perfectly good "old" equipment, so you should make it known at local computer user group meetings that you need/collect/will work on older equipment. Seek and you will find. Knock and someone might open the back door of a loading dock for you!

You have gone to a Saturday sale, gotten a call from an IT friend, or had someone at the back door of a corporate office offering you used equipment. After you negotiate your final deal for the gear, you find yourself driving home, pleased that you now have in your personal collection equipment that someone once paid thousands or hundreds of thousands of dollars for, and you have now obtained this equipment for the cost of a tank of gas (or less).

Reality Sets in on the Way Home

Of course, as you're driving home you realize that if it's an OpenVMS machine, you need license product authorization keys (PAKs) to make the software work. There may be licenses already on the machine (if the system admin didn't erase the system disk), but there is really nothing like having your own licenses to make you feel like you really own the box. But wait, don't call HP and buy new licenses. If your machine is destined for home use and personal education, you can get an OpenVMS hobbyist license with base VMS, clustering, and over 105 layered product PAKs for free! Just go to the OpenVMS hobbyist Web page at http://www.montagar.com/~hobbyist.

Register on the Web, and you will have all the PAKs you need to run an OpenVMS system sent to you via e-mail in a matter of minutes. You should note that these licenses are for personal or hobbyist use; you would have to call HP and get commercial licenses for your new system if you were going to use it for your or someone else's business!

Back at the Treasure Hunt

Arriving home with the new treasure for your collection you check out the hardware and supply the missing pieces for your new system—maybe a monitor or ASCII terminal, power cord, keyboard, or mouse; perhaps you need to tighten a few loose screws or cables or install a disk. The power cords are in place, all the cards have been reset in their slots, the network card is plugged in, the monitor is connected to what looks like a VGA port, and the moment of truth arrives. You apply power to the system! The monitor makes a sound like a loose strung guitar and symbols appear on your screen:

- 16, 32, 64, 96, 128 Mbytes of memory count down (the system is looking better and better).
- The floppy groans, the CD-ROM blinks, and the BIOS firmware shows it's loaded with options.
- The hard drive starts to boot Linux, or OpenVMS—quick, what are you going to do?

By the time you get to ask the questions the boot messages have flashed by, the X Windows banner login comes up and you are stuck with a multiuser system without a user name or login.

An X Windows login banner hides such secrets! Was this a system that managed a nuclear plant, hospital, bank, or ISP? Maybe part of the deal with the person who gave you the system was that you would scratch the disk to remove all of that valuable, proprietary data; but here blinks the cursor—mocking you from a small login screen.

So you resolve to break in, peek under the covers, and then scratch the drives—as any good system administrator worth his or her weekly salt ration might do. Ethics aside, there are many legitimate reasons for breaking in and taking control of Linux or OpenVMS systems, even if you're not the system owner!

Legitimate Reasons for Breaking into Linux or OpenVMS Computers

- The system admin has left the company with no transition to the new system administrator.
- The system has been compromised and you have to take control and reset/rebuild security. No one knows what is on the system anymore. No one knows what the current system is doing! No one remembers the System (OpenVMS SysAdmin) or root (Linux SysAdmin) password.

I have seen the need for breaking into systems so often in professional IT situations that whenever I tackle a new operating system or computer, learning how to break into it is right up there with learning the local text editor!

As for breaking into OpenVMS, there are few known cracks for this purpose, and even Linux is resistant to many remote attacks as well (when properly configured). For the professional system administrator, hacks and cracks are not the way to open a system. The zero-damage, professional method of breaking into a Linux or OpenVMS system is the only acceptable way to take over a system if you have a legitimate (legal) need to do so. Let's look at the standard methods for reseting root or System passwords for Linux or OpenVMS.

Ten-Step Process for Changing a Linux System's Root Password

Software required: V7.2 or 7. 3 RedHat CD-ROM #1

1. Have physical access to the Linux system.

2. Boot RedHat 7.2/7.3 linux CD-ROM distribution disk 1.

3. Type in "linux rescue" as the boot choice.

4. Choose English (or your local language).
   Choose US Keyboard (or your local variant).

   ```
   running anaconda -- please wait ...
   ```

5. Choose "Continue" when asked to mount your Linux installation (don't choose "read-only").

   ```
   Searching for RedHat Installation...
   Your image has been mounted under /mnt/sysimage/
   ```

6. Enter a return to get a shell "ok."

   ```
   sh-2.05a#
   ```

7. Change root to the mounted file system.

   ```
   sh-2.05a# chroot /mnt/sysimage
   ```

8. Change root password.

   ```
   sh-2.05a# passwd root
   New Password:Welcome1 (not echoed)
   Retype new password:Welcome1 (not echoed)
   ```

```
passwd: all authentication tokens updated successfully
```

9. Exit the shell and shutdown the system.

```
sh-2.05a# exit
```

10. Remove the RedHat Install CD-ROM and reboot from the hard drive. When the Linux system reboots, login as root. The root password will now be Welcome1.

Congratulations, you've just broken into a Linux system!

Ten-Step Process for changing an OpenVMS System Password

Software Required: None

1. Have physical access to the OpenVMS system.

2. Reboot/halt the boot process (find the reset button on your Alpha) and find yourself at a three chevron prompt ">>>".

3. For most Alphas, boot with the "Conversational Boot" flag set (0 refers to the cluster "system software" directory you want to boot from).

```
>>>boot -flag 0,1
```

4. At the "SYSBOOT>" prompt you can change any VMS sysgen parameters before the system runs!

```
sysboot> set uafalternate 1
sysboot> continue
```

System continues to boot up.

5. Because you've set the uafalternate to 1, the user and password files are wide open.

```
Username: <Enter>
Password: <Enter>
Password: <Enter>
```

System logon occurs and you're given a dollar sign prompt.

```
$
```

6. Prepare to modify the old VMS password file.

```
$ define/system/executive sysuaf sys$system:sysuaf.dat
$ set def sys$system
```

7. Run the authorize utility to change the actual system password.

```
$ run authorize
Authorize> Modifiy system/password=welcome1/nopwdexp
Authorize> exit
```

8. Invoke sysman and change the uafalternate system parameter to "0", so when the system reboots it will use the original password file.

```
$ run sys$system:sysman
sysman> parameters use current
sysman> parameters set uafalternate 0
sysman> parameters write current
sysman> exit
```

9. Invoke shutdown.

```
$@sys$system:shutdown
```

10. Reboot and log in to your OpenVMS system.
    The user System password will now be Welcome1

Congratulations, you've just taken over an OpenVMS system!

First the Rush, Then the Letdown

You log in as root or System and look around the box. Data is missing, and shards of pointers hint at what the system's duties once were. There are disk commands that point at nonexistent volumes, and tools that no one has used in five years stand ready to do your bidding. A gem from a past job emerges and reminds you of something you once did. Then you realize it's time to scratch the disks and put together a new configuration that better suits your purposes, rather than trying to salvage the existing system disk and its dated software.

You are tempted to do a full backup of your new system and save the work from an elder system, but you stop and consider privacy and those who left résumés, personal budgets, and e-mail on the system you now have full dominion of. Nothing appeared illegal, immoral, or fattening, so there is no reason to keep a copy of another person's data as far as you can tell.

During the course of work a professional system administrator will likely find a variety of personal user data on many different computer systems. There is no written

code on handling such data (short of a corporate HR directive about these types of files), but there is an ethical code. Just as Congress wrote the law such that radio scanners can listen to cellular and various other communications but not reveal what they've heard, system administrators have a similar responsibility. Corporate rules regarding privacy and personal files complicate the issue, but unless there is a very good reason (such as suspecting illegal activity was conducted with the system) a system administrator should never reveal or copy a user's personal files or information or reveal what was found to anyone else.

You weren't asked specifically, but professional courtesy to the previous owner or administrator demands that you erase the data (not necessarily the system programs) from the older system you acquired. If you believe in karma, then perhaps someone will do the same for you one day. When you're running late and just want to get rid of a machine without throwing it away and give it a good home, perhaps you could give it to an up-and-coming young professional who needs a good used system to practice with.

Of course, it's never going to happen that you give away systems without remembering what's on them, and you'll always delete all the data from all the disks you scrap, erase CD-ROMs in the microwave, bulk erase magnetic tapes, shred personal data and procedures, and destroy all ID cards you find in the boxes of surplus documentation. We're just talking about a hypothetical situation here and not anything I've ever found in/on/with systems I've seen.

There are many reasons to break into a system; in your tool bag for OpenVMS and Alpha you should always have an OpenVMS boot disk (the CD-ROM that loads OpenVMS) at the latest version (7.3, 7.3-1) you can find, and a RedHat Linux 7.2 or 7.3 boot disk. These tools will help you when opening up systems when you need to do so.

From treasure hunt to salvage, to corporate rescue, to recovered systems, breaking into systems is something all admin folks should be able to do.

May you find interesting things in the systems you overpower, and remember with great power come great responsibilities. Ethically, morally, and professionally you don't break into systems just because they are there or you have a way in. That way leads to the dark side of the profession.

I grant you this knowledge to help and not to hinder. I also warn that, as I was taught and then enjoined by my teachers—use these powers for good and not evil. I know you will know the difference when you are confronted by systems you need to take control of!

Chapter 3 — TCP/IP for OpenVMS and Linux

It seems like a lifetime ago, but I remember having an ongoing argument in 1989 about which was better: IBM's SNA protocols, DEC's DECnet and LAT protocols, or the up-and-coming TCP/IP protocols for commercial systems.

The discussion extended to many customers and many in the local DFWLUG user group joined the discussion with barbs and angry words sniping the other protocols. Async transmission is better then sync transmission—it recovers better and let's you use lower-cost lines, peer-to-peer networking is the only way to go, standards are the way of the future, mine has more features, your protocol is lame, and so on.

In the end, all the arguments missed the point. We were all arguing the technical merits of peer-to-peer networking, counting the number of terminals that could be supported, or how many angels danced on the end of a 50-ohm cable. That wasn't the right focus at all, and many people missed the real reason for this change in networking technology in such a short period of time.

IBM's SNA could host tens of thousands of terminals in a single network with subsecond response time. DECnet provided peer-to-peer networking with over 100,000 nodes in a single network (when the Internet had almost 15,000 nodes), Novell was fastest for PC disk sharing, and the much-touted OSI protocol would unite all networks into a single, standards-based world that would correct all the shortcomings of past protocols. But all of the technological greatness of these and other protocols paled against TCP/IP.

TCP/IP would, by 1993, sweep away all the competition in the network protocol world for any new applications or networks. TCP/IP crushed OSI's government mandate as a protocol of choice for all new network acquisitions. DEC had believed it could skip over the TCP/IP revolution if it just had the best OSI protocol stack, because TCP/IP was so wanting (at the time) technologically and the marketplace would have to use OSI to solve its networking problems.

Instead, TCP/IP spawned network companies such as Cisco and built million dollar software companies such as TGV (Two Guys and a VAX) just to write and support a TCP/IP stack for OpenVMS. Other companies emerged for TCP/IP to SNA gateways, TCP/IP terminal servers, and the whole pantheon of TCP/IP routers, bridges, terminals, and printing solutions.

Why did this technological transition occur? Better software applications? Better, more efficient networking communications? A technological masterstroke delivered from the University of California at Berkeley for the rest of the world's betterment? TCP/IP wasn't better than other protocols; its addressing, applications, routing, and name

space handling were as kludged as it got (remember Yellow Pages and the first DNS programs?).

Nothing quite so grand as the best technology catapulted TCP/IP to its preeminent spot; it was simply capitalism driving the network marketplace. What TCP/IP had in the late 1980s that propelled it into the commercial IT departments and governments worldwide was that while other networking protocols were owned and licensed by various vendors with expensive fees and licenses, TCP/IP was practically free.

By the mid-1980s for $500 a development software vendor could develop and use TCP/IP for its operating system, router, bridge, or tool of choice (licensed from AT&T). Other network protocols and vendors didn't want to license their network secrets to other vendors, at least not cheaply. SNA and DECnet folks saw their network technology as added value and a distinct solution instead of just part of the packaging. Berkeley's Berkley Standard Distribution (BSD) of UNIX and System V helped drive UNIX and TCP/IP into colleges and universities in the late 1980s and early 1990s, but it was (in my opinion) the low development cost, with free relicensing and the ability to freely (at no cost to the vendor) redistribute TCP/IP protocol stacks for commercial devices, that won the place TCP/IP now enjoys.

And, as the story goes, by 1993 IBM had SNA as a legacy networking protocol, DEC had DECnet/OSI, and all other commercial vendors of networking hardware and software were using TCP/IP not because it was better, but because it was free ($500 for a development license was as free as it got in the 1980s!). This market transition event taught me one immutable fact for our industry: Free beats something that costs any day of the week in the market place.

Why remind everyone of TCP/IP's golden age in a book about OpenVMS and Linux? The forces at work with Open Source are doing to today's marketplace exactly what TCP/IP did to SNA and DECnet over the last decade. Certainly, Linux has a distribution cost, but after that it's all but free.

Tools that we think are essential and are willing to pay for are next year's open standards and are included with Open Source distributions. To grow in the marketplace OpenVMS must continue to innovate and provide added value with new features such as Galaxy (multiple instances of OpenVMS within an SMP single system) and disaster-tolerant clustering. To continue to take market share, Open Source distributions such as Linux must continue to include new features and improve and evolve from the features customers once spent their IT budgets on with high-end operating systems. Progress in Open Source and commercial systems is a two-way exchange. Open Source tools become part of OpenVMS, and, over time, new features such as clustering and large-scale SMP will become part of the Open Source initiatives. This type of exchange increases all system functionality and lowers the cost of computing for the industry over time.

Today's Networking Standards

Today (10 plus years after the network protocol wars) TCP/IP is robust and well understood by commercial IT groups and individuals. Now the industry is evolving into IPV6 (currently IPV4 is used for most implementations). The protocol is expanding and improving to incorporate even more features in routing and secure applications—all with a standards-based networking protocol.

Networking for OpenVMS and Linux (or any two or more computing machines) has to be based on the same standards. Today's networking standards of choice are TCP/IP software protocols and Ethernet hardware protocols. In previous years it was also important to be able to exchange across terminal lines, modems, and direct synchronous and asynchronous data connections (such as RS-232 cables); the need for the older types of connections is waning but not quite obsolete. Today's workstations continue to use modems for dial-up with point-to-point protocols that encapsulate TCP/IP for transmission between two computers. With more and more workstations and servers tied to DSL or cable modems, routers, and even firewalls, Ethernet networks (even Ethernet wireless networks) are the choice of interconnection whenever possible.

Notice that all the IEEE 802.X standards refer to the bottom two layers (all hardware) of the OSI model (http://standars.ieee.org/) (See Table 3-1).

Table 3-1: Standard OSI Model with TCP/IP Positions

OSI Network Model	TCP/IP Internet Model	IEEE Standards	
Layer 7—Application	Netscape, Xterm, E-mail Readers, FTP-based tools, and other application programs for displaying data		
Layer 6—Presentation	Telnet, FTP, SMTP, other TCP/IP protocols		
Layer 5—Session	Telnet, FTP, SMTP, other TCP/IP protocols		
Layer 4—Transport	Transmission Control Protocol (TCP) Unacknowledged Datagram Protocol (UDP)		
Layer 3—Network	IP Internet protocol		

Table 3-1: Standard OSI Model with TCP/IP Positions (continued)

OSI Network Model	TCP/IP Internet Model	IEEE Standards	
Layer 2—Data Link	Network interface cards: Ethernet twisted pair, coax, Big Orange Cable (AUI), Token Ring, FDDI, ATM, ODI (Open Datalink Interface), NDIS (Network Independent Interface Specification)	802.1 802.2 802.3 802.4 802.5 802.6	Internetworking LLC Log Link C Ethernet CSMA /CD Token Bus Token Ring Metropolitan Networks
Layer 1—Physical	Network transmission media: twisted pair, coax, fiber, wireless	802.7 802.8 802.9 802.10 802.11 802.12	Broadband Advisory Fiber Tech Advisory Integrated Voice /Data Network Security Wireless Networks Demand Priority

Hardware and software standards make computer communications possible, and while looking at another OSI chart (Happy Mnemonic: All People Seem to Need Data Processing!) may not seem useful, you will need to know where TCP/IP applications and protocols operate when you are troubleshooting your network applications! If you can't tell the network manager why the problem is in Layer 3 or below, he or she will always blame it on your box or application—so you better get used to it.

The Importance of a Network Manager

Always remember that the network manager is your best friend in the network and can get you TCP/IP addresses, cables, and even Ethernet cards and software! The network manager knows where the routers are and the magic numbers to get to ***(domain) Digitial Name Service (DNS), as well as the gateway numbers to other networks. You should remember and mark well that your new best friend will become a snarling raging beast, who'll insult your lineage, choice of OS, or computer knowledge, if you ever have occasion to question the network's current status or ask why your system can't seem to communicate with the other network.

So let's go over some questions that you or your network manager will need to answer for both machines before you can even start to connect a VMS and a Linux system to a TCP/IP network.

Questions for Network Setup (Questions You Must Ask to Set up a TCP/IP Network)

Q1: What is my TCP/IP address and how do I get one?

An IP address is a unique numeric representation of a host system number for either your Linux or VMS box.

TCP/IP uses the IP address to communicate with other hosts across a TCP/IP network. An IP number works like a street address. If I am given the address 666 Halloween Lane, I will be able to find the house represented by that address. If you give your TCP/IP networked Linux or VMS box an IP address, other hosts will be able to find it and your box will be able to find other hosts across the network.

Your network administrator or Internet Service Provider (ISP) will issue you an IP address that looks something like this: 10.0.0.1. This number is a decimal representation of a binary number.

If you are over 40 years old, you may remember being taught other base systems. People use a base 10 numbering system for obvious reasons (count your fingers to figure out why!). Is any of this returning from your dank, dark memories of yesteryear? No? Okay, then let's explain in slightly more detail. There are several base systems used in the computer field.

Binary (base 2), hexidecimal (base 16), and decimal (base 10) are used most often. While the IP address you see is in base 10, the hosts running TCP/IP see it in base 2. So an IP address of 10.0.0.1 in base 10 is 00001010000000000000000000000001 in base 2.

Now, which number is easier to remember?

Certainly, if you're a programmer 00001010000000000000000000000001 is much easier, just remember all the 0s and 1s. With the end user and future Microsoft Certified Professionals in mind, the standards committees came to use the base 10 representation of a binary (base 2) address. What is represented here is a simplification of TCP/IP and computer technologies.

Just one last thing about IP addresses before moving on to the next question. IP addresses are divided into classes. Think of classes as large groups of IP addresses, associated in numeric order. Class A IP addresses are grouped from 0–126, Class B is grouped from 128–191, and Class C is grouped from 193–233. We can determine what Class an IP address belongs to by examining the first set of numbers on the far left side. An IP address of 205.165.160.1 is Class C, because 205 is between 193 and 233.

Okay, really, this will be the very last thing about IP addresses. Due to the great explosion of the Internet, with servers, switches, routers, handheld telephones,

refrigerators, and cars' hubcaps needing an IP address, IP addresses have become a scarce commodity.

To alleviate the scarcity of routable IP addresses (routable means able to transverse the Internet), network administrators are using a technology called NAT.

So, most likely, you will be assigned a nonroutable IP address if you are at work or setting up a private local area network at home behind a cable/DSL connection. The nonroutable (private) IP addresses are 10.0.0.0, 172.16.0.0, and 192.168.0.0, depending on how many hosts are on the network.

NAT technology allows you to have many nonroutable IP addresses behind a single routable address, which connects all the nonroutable addresses to the entire Internet.

Q2: What is my subnet mask for the network I'm currently on?

Subnet masks divide TCP/IP class addresses into multiple networks. This allows networks to use more addresses and limits the number of TCP/IP nodes on these subnetworks.

Usually your network adminstrator will provide you with a subnet mask, and there are exact calculations for dividing different TCP/IP classes into subnets.

Q3: What are my host and domain names?

Remember in Q1, where I asked which was easier to remember, the base 10 or the base 2 IP address? Well, it's even easier for your to remember names.

VMSONE.com is easier to remember than 65.65.113.10. Domain names are used to group hosts on specific subnets of an IP address. It's much easy to identify a group of hosts that are related if they all have the same Internet domain.

Host names have to be unique within their domain. Domain and host names with their corresponding IP addresses are recorded in domain name service (DNS/BIND) servers on your network. Both OpenVMS and Linux can be DNS servers, but if you are just connecting to a TCP/IP network, DNS setup and service are best left to your network administrator or ISP. DNS servers are not for the novice network hack and require the cooperation of other DNS servers in a network to work properly.

Q4: What is my DNS server address?

The network administrator or ISP should provide the TCP/IP addresses of a primary DNS server and its backup (known as secondary). As mentioned previously, host and domain names are stored in a DNS server. A phone book and a DNS server are used in a similar manner.

If I know a person's name (host, domain), by using a telephone book, I can locate his or her telephone number (IP address). The same process is used with DNS: If you want to find VMSONE.com, but don't know the IP address, the underlying TCP/IP system will resolve the issue.

So, if I point my Web browser to VMSONE.com (a name that I know and can remember), the underlying TCP/IP system will query the local DNS server, resolving that name into the appropriate IP address, which it then uses to connect to VMSONE.com.

Okay, okay, everything I just said is a vast simplification of the entire process. I realize this, but you don't have to have a Ph.D. in TCP/IP to set up your nodes and get them on the Internet or your local intranet!

Q5: What is my default gateway/router address?

The default gateway number is the address that connects to the other networks from your local network. It could be the address to your NAT (there's that word again) firewall, proxy server, or the router that connects your LAN to the rest of the Internet.

The simplest way to understand this is to note that you will not be able to communicate with any other LAN or network unless you have this number defined. For initial setup you don't want routing enabled from your machine (this would mean your system had multiple TCP/IP paths enabled) and you don't export routing information to the entire network unless your current system connects two separate TCP/IP networks (instead of a router).

Your host TCP/IP address?	Example: 10.0.0.25
Subnet mask address?	Example: 255.255.255.0
Your host name and domain?	Example: lnxone.vmsone.com
Your remote DNS server address?	Example: 10.0.0.8
Your LAN gateway address?	Example: 10.0.0.1

With the answer to these five questions (from your network administraitor or ISP), you should be able to configure TCP/IP on a single network and connect both your OpenVMS and Linux systems to the network. Let's look at the process for configuring TCP/IP systems.

Configuring the Systems

OpenVMS and Linux both use utilities to help configure your TCP/IP networks. On Linux (from the root account) it's called network configuration (under RedHat root menus), and on OpenVMS (from the System account) it's called tcpip$configure.com (or ucx$configure.com on older versions of TCP/IP services for OpenVMS).

Configuring TCP/IP on Linux

From the root account on your Linux box you can invoke Linuxconf by typing Linuxconfig from a terminal window or use the system menu to call network configuration from the GNOME footprint. Figures 3-1 through 3-5 illustrate how TCP/IP can be configured on Linux.

Figure 3-1 Fill in the basic host information and then the adapter information.

Figure 3-2 Choose basic host information

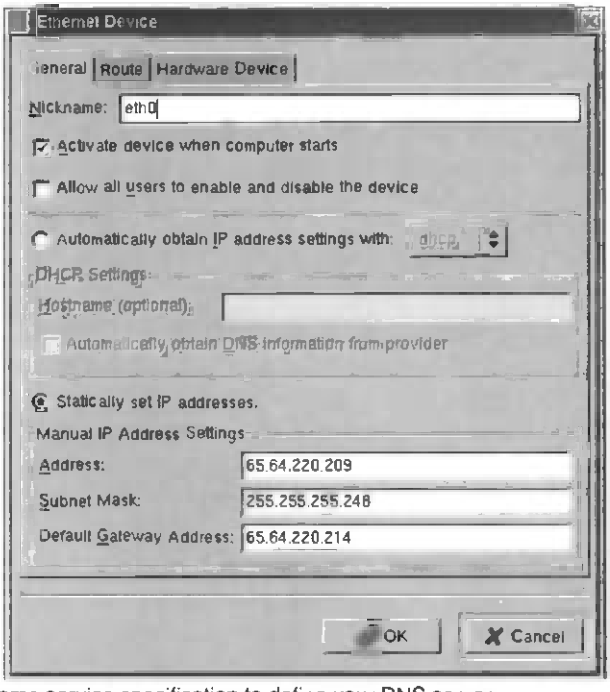


Figure 3-3 Use name service specification to define your DNS server.

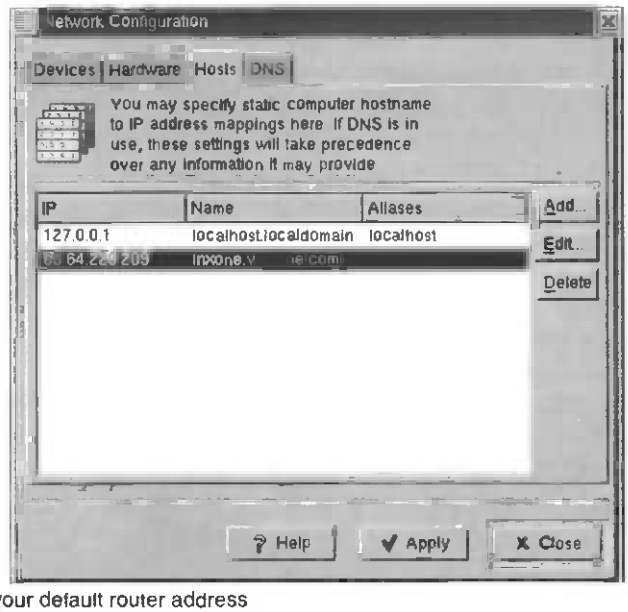


Figure 3-4 Set your default router address

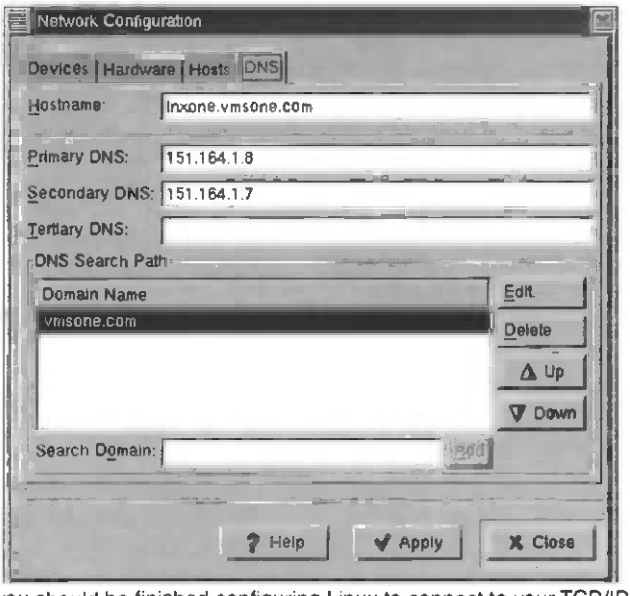


Figure 3-5 Now you should be finished configuring Linux to connect to your TCP/IP network!

Configuring TCP/IP on OpenVMS

Log in to the system account and type from the $ prompt

```
$ @sys$manager:tcpip$config.com
```

Figures 3-6 through 3-9 illustrate how TCP/IP can be configured on OpenVMS.

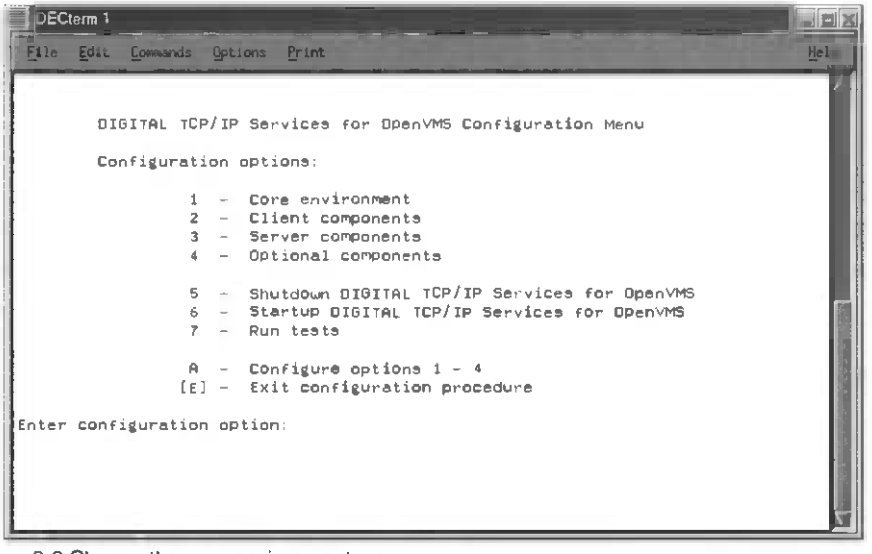


Figure 3-6 Choose the core environment menu.

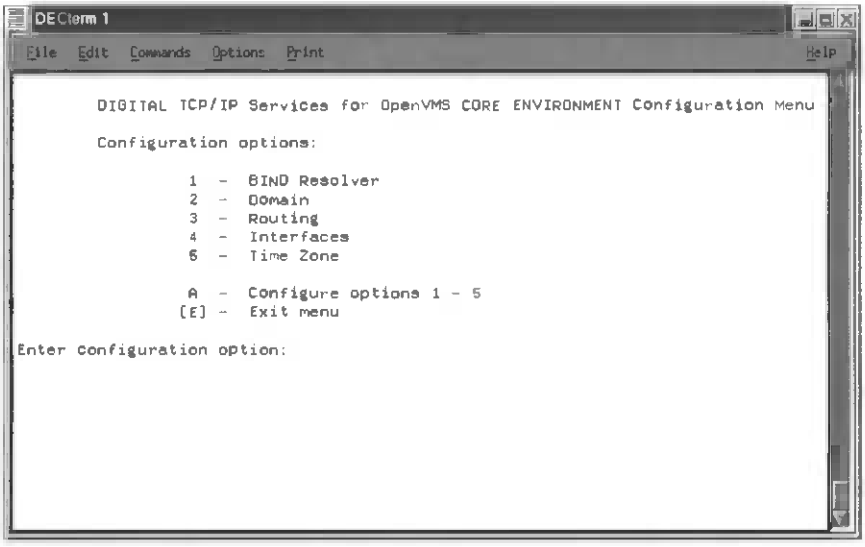


Figure 3-7 Enter the data on the OpenVMS menus. Note that BIND and DNS are the same animal!

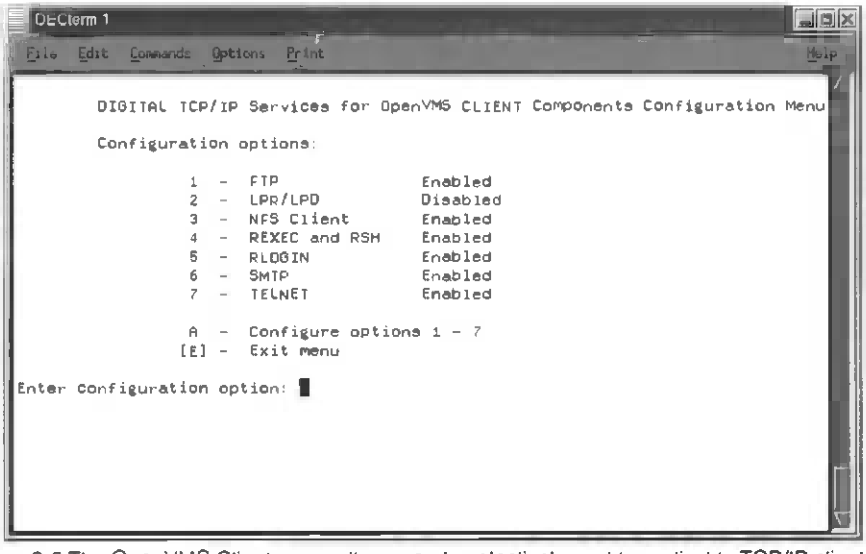


Figure 3-8 The OpenVMS Client menu allows you to selectively enable or disable TCP/IP client applications from your system.

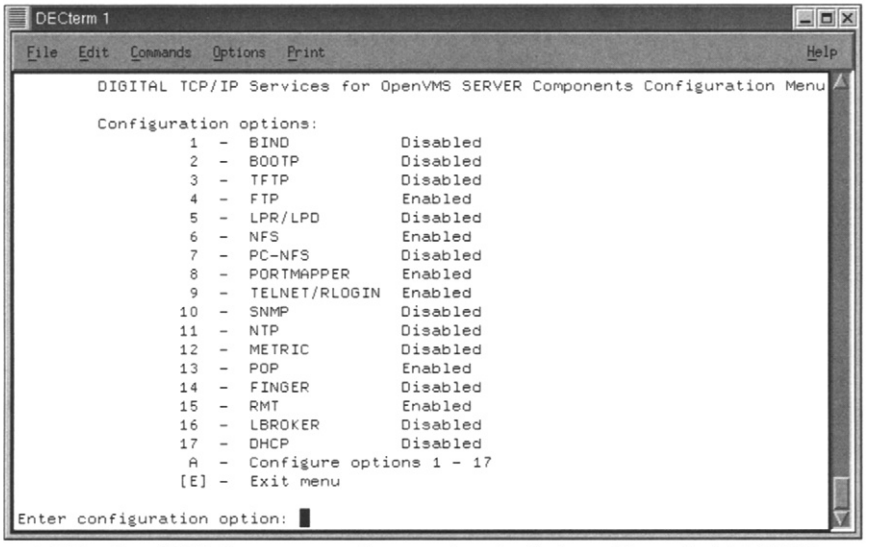


Figure 3-9 The OpenVMS Server menu allows you to selectively enable or disable TCP/IP server applications for your system. Remember that if you want NFS, POP3, IMAP mail, or another feature, it must be enabled to work!

TCP/IP Applications

OpenVMS and Linux network together by using similar applications that work across the TCP/IP protocols connecting both machines. Table 3-2 is a list of some of the more common applications with a brief description of what functions they perform.

Table 3-2: TCP/IP Applications Between OpenVMS and Linux

TCP/IP Application	Description	When Configured
BIND/DNS	Internet name services	Usually with your ISP
FTP	File Transfer Protocol—network copies files between systems	Initial system installation
LPD	Line Printer Daemon—used to print to network printers using TCP/IP	Whenever you add a network printer
SMB	System Message Block—disk sharing between Windows UNIX, VMS, and Linux systems	See Chapter 7 of this book

Table 3-2: TCP/IP Applications Between OpenVMS and Linux (continued)

TCP/IP Application	Description	When Configured
Telnet	Terminal protocol application determines what terminal emulation is used and how well it works; TCP/IP just provides the connection	Initial system installation
SMTP, POP3, IMAP	E-mail, Post Office Protocol, Simple Mail Transport Protocol	See Chapter 6 of this book
SSL, Stunnel	Secure tunnel using secure sockets layer encryption	See Chapter 4 of this book
X Windows	Display X Windows between networked systems	See Chapter 5 of this book

OpenVMS and Linux TCP/IP applications work together because both applications adhere to industry-standard definitions for their functions. It might be a little too easy to say that it just works! But most of the time that's exactly the way it does work! Want to copy a file? FTP works on both systems and will determine if you are sending a binary file (compiled, zipped, or object file) or a text file and will transfer/copy the file correctly between machines. The same is true for X Windows, e-mail clients, Finger, Telnet and all the other standard TCP/IP applications.

URL Tricks and More

TCP/IP tools are built in as part of the Linux and OpenVMS operating system. You can copy files, send e-mail, run a Web site, and much more. Interprocess communications made the client/server revolution possible, and the World Wide Web and known IP ports keep applications working while new applications can be tested and run on the same server before moving a new application into production.

The future of all of this is, of course, new, and we should see more applications with enhancements and continuing increases in encryption and the use of encryption tools.

One more TCP/IP trick before we move on to security in the next chapter: A good friend of mine, Steve Smiley from the DFWLUG here in Dallas, showed me this little trick and did an article for the DFWLUG *Quadwords* newsletter based on this conversion.

If you do an "NSLOOKUP" from a Linux or OpenVMS system and plug in a valid DNS name, you get an address such as 209.39.152.33. If you convert the TCP/IPs four octets to their binary representations (with the help of a scientific calculator), you get

```
209 = 110100201
 39 = 00100111
```

Be sure to add the padded 0s in front for an 8-byte/character binary!

```
152 = 10011000
 33 = 00100001
```

Add the binary (base 2) strings together

```
209.39.152.133 = 11010001001001111001100000100001(base2)
= 3509032993 (base10)
```

From Netscape Internet Explorer or your favorite Web browser, try these numbers:

```
http://3509032993
http://1094769874
```

Which is easier to remember a 10 digit number or a URL name on the Web? You do the math.

Chapter 4 — Security Tools for OpenVMS and Linux

Ever Been Hacked?

It was summer's end, 1999. I was a seasoned, tough, and experienced SE preparing my customers for the Y2K nonevent with firmware, software, and professional advice. She was a tough dame from the wrong side of the pews from the local church group but could she sling COBOL during those months. The computer world felt like it was self-destructing before our very fingers, and then I decided to build a Linux box.

A Linux box. Oh, I had loaded Linux several times before, but it was time. Time to have a Linux box on my home network configured the way I liked it, time to have a Linux box on the Internet as a symbol of my computer prowess. Besides, all of you know how women dig guys who run Linux, right?

Well, that's how it all started. I built a 450MHz workstation/server from junk I had laying around, installed a 40GB disk, 256MB of memory, and 10/100 Ethernet. I built Linux from a "red" distribution I bought for $3 per disk from cheapbytes.com. Then I loaded everything. Not a few tools, but everything. I loaded compilers (FORTRAN, COBOL, C, C++, CplusC, and more) graphic tools such as gimp, hypersnap, and ASCII art; games and more games, X Windows, and SAMBA, MYSQL, YOURSQL, NFS, NSA, UBU. Anything I could find in RPM format, I loaded.

Oh, it was a fine box. A system built by a professional for his own amusement. It would be my personal Linux playground. Then I sat in front of my creation and booted it for the first time connected to the Internet. I had had my OpenVMS server on the Internet for almost 10 years without incident (longer if you count being tied to mail via UUCP), so I felt safe and secure that a Linux OS would be safe enough for just me against the ravages of the Internet. First things first: I configured Apache and created a SAMBA SMB pointer to my home directory and an MP3 file share. I loaded 10 gigabytes of Napster files on that file share and began to explore the tools and things that were now available to my home network.

Four Days Later

The newness of the Linux box had ebbed somewhat, and I was back on my OpenVMS system reading my mail in Netscape. A strange e-mail address appeared in my inbox that looked as if it were from "Down Under." I opened the letter and realized immediately from the headers it was from an OZ domain and began to read. "Greetings," I could almost hear the Aussie accent through the wires "Why are you probing and scanning my entire Class C address space?"

I was stunned. I wasn't running any port scans on Pacific Rim Class Cs, was I? Then I saw the logs he sent me. It wasn't me, it couldn't be me! Remember all the phases—denial, anger, bargining, depression, acceptance. I went through them all that first 24-hour period after being hacked.

It looked like these pings and scans were from my new Linux box! It was late and I was in total denial and sent a mail message back to this brute, "Couldn't be me" Then I went to bed.

The next morning my new friend from Australia had sent me more logs, proving my system's complicity in his probing: "Greetings Mate, I believe the script kiddies have control of your box and you really should find out what they're doing."

Script kiddies—the name struck terror into my four-day-old Linux install. How could they even find me in the huge expanse of the Internet. I was only up for four days! Feeling like a stranger in a strange land, I did what any OpenVMS system manager would do when confronted by a problem with a Linux box. I called another OpenVMS system manager! Not just any system administrator, but my good friend David Cathey, chairman of the DECUS/VMS SIG and president of the Dallas–Ft. Worth local user group. David had more Linux experience then I and quickly pointed out that we should look for strange files on the system.

Dave came over and we looked for temporary files. Seek and ye shall find; I had all the horror of parents finding their child infested with lice! The Linux box was not only hacked, but there were log files of port scans, port scan executables, and logs with entire text file conversations between two miscreants about cracking boxes and using them to find new boxes to attack. David and I watched and waited and captured the session information as superuser—one was looped through AOL so it wouldn't get caught, and the other was looped through the University of Korea.

I did what any red-blooded system administrator would do: took the logs, copied the files, called AOL (it at least was in my jurisdiction), and gave AOL the user ID of the perp who had ravaged my machine for fiendish purposes.

Of course, AOL wanted nothing to do with my evidence unless I had involved the FBI or at least local law enforcement! I thought about the $42 million in IT charges I could bill for my production Linux box being attacked and compromised by an AOL script kiddie, but thought better about wasting the authorities' time.

My postmortum on the Linux box consisted of taking it off the Net and identifying how the crackers had just walked right in and had my machine doing grunt work like the mule that it was. The answer was in the password file—I had loaded everything, every RPM I could! A database loaded and made a privileged user account by default that could create other accounts—WITH A DEFAULT PASSWORD!

Humble, not quite so full of confidence, I scratched my Linux system and loaded only the RPMs I was going to use and understand immediately. From this I learned a great lesson in system security:

> Don't load any software on your system if you don't know what it does and change any default settings to more secure settings.

Of course, I also learned something about trusting Australians more and AOL and University of Korea computer accounts less.

Preparing for the Storm

Steven Smiley and I were taking a break at a well-known computer security convention watching the evening's film-festival double feature (*Colossus the Forbin Project* and *Dark Star*). From the back of the room came cackles of laughter that ment someone was doing something cool. Steve wandered off to see a late night Q&A session, but I went to the back and sat with the laughing crowd of security consultants.

As I listened, the audience was arrayed around a young man who was sitting with his laptop wirelessly connected to the network as he sat on the floor.

"Sniffed his instant messenger password and now I'm talking to his mother"—roars of laughter from the folks sitting around.

"But Mom, I'm going to stay in Vegas and work at a casino. I've found my true calling!" said the keyboardist out loud as he typed to the mark's mother.

"You're not going to destroy that poor guy's mother's opinion of him are you?" I asked.

"Nah" laughed the keyboardist, "I'm just going to make sure he learns the value of encryption in network communications!"

Everyone laughed along at the joke.

The simple truth is that if you are on an open network, anyone on the network can potentially sniff packets and scan for plain-text passwords, scan all your TCP/IP ports, and look for known back doors or worse. What do you do? Firewalls? Intrusion detection tools? Harden the operating system? Unplug from the network completely?

It's not likely we can just retreat from network communications, since they are a prerequisite for computers and portable devices these days. So one of the main

solutions today and increasing in importance in the future is to use an encrypted link between your client and server when making a connection—ANY CONNECTION!

Now remember that standard TCP/IP tools such as Telnet and FTP aren't encrypted at all, so any data you send across those tools can be examined for passwords, PINS, codes, credit card numbers, social security numbers, and even the data that resides inside a transferred file. Now ask yourself: Who would take the time to scan for these things on your network? You'd be surprised. Most computer crime is an inside job by 60/40 percent. This is down from last year when the norm was 70 percent inside jobs. The Internet and other network paths are the fastest growing access method for computer crime!

Consider that each server that runs standard server applications uses the same ports, so that not only do you know what the server is running, but you might be able to use these ports as back doors to these servers.

TCP/IP Known Ports

Because they are standard applications, TCP/IP defines for servers "known ports," which stand ready to communicate with client connections that try to communicate with them.

Perhaps the best known port is port 80 or the port that the World Wide Web protocol communicates on. Connecting from a client with a Web browser, the WWW protocol connects to port 80 on a Web server using TCP/IP across this link—all hypertext, pictures, graphics, movies, Java, and other Web information.

Known ports also tell us what services a server is currently offering. Many applications have their own TCP/IP port number, which they always use for a default installation. These applications can sometimes change port numbers but rarely do. A server administrator can change these ports, but then other client applications can't communicate with them.

The good news is that these ports now give us a way to look up all services being offered on a TCP/IP node. The bad news is that these ports now give us a way to look up all services being offered on a TCP/IP node! Security can be opened up on these nodes by simply probing for known passwords and known back doors for these applications.

The tool of choice for network server watchers is a piece of software called a port scanner. Port scanners are available on the Inernet; costs vary from free to thousands of dollars. Many of these tools will check a server for every port from 1 to 80,000+. Many known software ports have banners that will tell you a great deal about the

port's application, the server's operating system, and even who owns the system! A good or even a mediocre port scanner will save all this for search or later review.

The simplest and easiest way to use the port scanner is, of course, the TCP/IP application Telnet! You can use Telnet to check and even interact with any known port or application port on a server! Usually you use your Telnet client to connect to a server port (number 23 by default), but most Telnet applications let you change your port to connect with any port you wish. Simply type from a VMS or Linux prompt

```
telnet/port=25 yourhost.domain.com
```

from a terminal window. Your interactive terminal session will be connected to port 25 (the e-mail port on that server), if it's active! Type

```
helo
help
```

and you're on your way to an interactive e-mail system that will allow you to even send SMTP mail in an age-old sport call e-mail spoofing. You can Telnet to known ports and then talk directly with the applications that control those ports.

Many applications have security. Many do not. This is not something you use to attack other people's systems, but is a geat way to confirm that a server's TCP/IP applications are configured, installed, and are running when your client application isn't working or talking to the server.

When you are working on your Linux or OpenVMS system, walking known ports is experimental and a learning tool. But when you are walking all the known ports of Australia and New Zealand from someone else's Linux system and using it as a grunt box (doing your dirty work) while looped through AOL and five offshore university accounts, I believe that falls into the criminal-activity realm.

I've included a list of some known ports; there are others and each network application may include its own known port. These are the ports I look for to see if applications are running; other, more thorough, port scans could identify unknown back doors, security holes, and applications you didn't install. You should regularly scan ports 1–80,000 on your servers just to make sure.

TCP/IP known ports (some not all) for Linux and OpenVMS systems include the following:

```
7,echo,Echo
11,systat,Active Users
21,ftp,File Transfer
22,ssh,SSH Remote Login Protocol
23,telnet,Telnet
```

```
25,smtp,Simple Mail Transfer
42,nameserver,WINS Host Name Server
43,nicname,Who Is
53,domain,Domain Name Server
66,sql*net,Oracle SQL*NET
79,finger,Finger
80,http,World Wide Web HTTP
88,kerberos,Kerberos
109,pop2,Post Office Protocol—Version 2
110,pop3,Post Office Protocol—Version 3
111,sunrpc,SUN Remote Procedure Call
118,sqlserv,SQL Services
119,nntp,Network News Transfer Protocol
135,ntrpc-or-dce,DCE endpoint resolution
139,netbios-ssn,NETBIOS Session Service,
143,imap,Internet Message Access Protocol
389,ldap,Lightweight Directory Access Protocol
396,netware-ip,Novell Netware over IP
443,https,https
513,login,remote login a la telnet;
514,rshell,cmd
515,printer,spooler
1080,socks,Socks
1313,bmc_patroldb,BMC_PATROLDB
1352,lotusnote,Lotus Notes
1433,ms-sql-s,Microsoft-SQL-Server
1494,citrix,Citrix
1524,ingreslock,ingres
1525,orasrv,Oracle Server
1527,tlisrv,Oracle Server
1723,pptp,PPTP
1745,winsock-proxy,Winsock Proxy
2000,remotely-anywhere,Remotely Anywhere
2001,cisco-mgt,Cisco router management
2049,nfs,NFS
2301,CIM,Compaq Insight Manager
2447,openview,OpenView
2998,realsecure,RealSecure
3000,hbci,HBCI
3300,bmc-agnt,BMC Patrol agent
3306,mysql,mysql
4001,cisco-mgmt,Cisco router management
4045,mfs-lockd,NFS lock
5222,jabcast ,,
5631,pcanywheredata,pcANYWHEREdata,
5800,VNC,Virtual Network Computing server,
6000,xwindows,6000-6063 X Windows System
6001,cisco-mgmt,Cisco router management
6667,irc,Internet Relay Chat server
8000,web-shoutcast,Web/Shoutcast Server
8001,web,Web
8002,web,Web
8080,WWW-Proxy,Standard HTTP Proxy
```

```
9001,cisco-xremote,Cisco xremote
26000,quake,quake
32771,rpc-solaris,Solaris RPC
65301,pcanywhere-def,pcAnywhere
```

Encryption—Not Just for Breakfast Anymore

You need to use encryption of the tools/commands when communicating sensitive (or not so sensitive) information, or you put yourself at risk of having your passwords, files, and anything that moves across the network read, copied, and redistributed. Let's talk about the application encryption protocols that are generally available on OpenVMS and Linux systems.

SSH (V1 and V2)

SSH, or secure shell, features are well known and have been available for many years between many different hosts in the TCP/IP world. SSH has the ability to create an encrypted session between two hosts. You can open an encrypted SSH client terminal connection to an SSH server. All the terminal communications are then encrypted between the client and server. Both Linux and OpenVMS can be clients and servers in this arrangement, allowing secure terminal sessions to and from both systems. MultiNet offers this protocol and a file transfer application, SCP, for OpenVMS systems using this protocol. (VMS TCP/IP engineering has been requested for SSH for some time, but they have had their attention distracted by other TCP/IP directions.)

In the rest of the Open Source and Linux world, SSH is usually a terminal or shell command tool that comes with the operating system, but does not do file transfer (FTP) by itself. Additional applications have been written to use the SSH protocol for file transfer, such as MultiNet's SCP product for OpenVMS and SFTP (Secure FTP) for Linux and UNIX users.

HP's TCP/IP services for OpenVMS does not have SSH capabilities, but with the addition of the MultiNet SSH layered product, SSH protocols work with TCP/IP services (5.3 TCP/IP EAK from HP), making SSH work between Linux and OpenVMS. You would use SSH if you needed secure terminal access between Linux or OpenVMS systems in a public or private network that had to support interactive terminal or shell command logins and not be able view password exchange and data exchange in plain text.

Software for this type of encryption application can be found at the following

- *For Linux*: OpenSource available in the Linux standard distributions or at http://www.openssh.org/

- *For OpenVMS*: Process software (MultiNet) at http://www.process.com/
- *For HP*: TCP/IP services for OpenVMS V5.3 at http://www.openvms.compaq.com/

SSL

SSL, or secure sockets layer, has been around since before 1996, and it still isn't at Version 1.0. That's not to say that it's not functional—just that it's taking its time becoming a V1.0 release. If you ever doubt SSL's functionality, just check your Web browser's lock icon on the bottom of your favorite browser when you connect to your bank account or try to buy something on the Web. SSL is invoked anytime your lock icon is closed!

SSL's roll as an encryption protocol is expanding, with new and more sophisticated tools and applications tied into its authentication mechanism and communications style. Today, SSL can be used for Web, terminal sessions, file transfer, POP and IMAP, and virtually any TCP/IP application that uses a known port style of host addressing and also needs encryption for its communications. Of course, SSL needs a little help to do some of these tasks and help is now available for OpenVMS and Linux!

OpenSSL consists only of a toolkit with programs and programming interfaces. Either you have to modify programs to use SSL (as your favorite Web browser has) or you need to have another tool that goes between standard TCP/IP application programs and SSL programming interfaces. That tool, for Linux and OpenVMS, is called a "Stunnel," or secure tunnel.

Before we can use the Stunnel to encrypt port-to-port communications, we must have SSL libraries installed on both systems to do encryption and authentication between nodes.

SSL Distributions

OpenSSL for Linux is Open Source and usually comes with a standard Linux distribution or from http://www.openssl.org/.

OpenSSL for OpenVMS is available from HP and is included in the standard OpenVMS distribution, or it can be pulled from the Web at http://www.openvms.compaq.com/.

Note that OpenSSL for OpenVMS will work on OpenVMS versions V7.2-2 through 7.3-1 as of this writing.

Stunnel

SSL requires an application to map and remap local host ports, and that remapping tool is called Stunnel.

Stunnel is an application that allows you to encrypt TCP connections within SSL. This is a very new and useful approach to encryption applications, because instead of reprograming standard nonencrypted application protocols (such as Telnet, POP, IMAP, LDAP), you can just use Stunnel to provide the encryption.

Just how useful is Stunnel? With the introduction of OpenSS and Stunnel on OpenVMS, a system administrator has another tool that can now securely manage the OpenVMS systems from a Linux workstation or even a Windows workstation! And, of course, it works for the OpenVMS system administrator who wants to securely manage his or her Linux servers too!

Stunnel distributions include the following:

- Open Source Stunnel sources for Linux and Windows may be found at http://www.stunnel.org/.
- OpenVMS Stunnel kits are available with the OpenVMS 7.3-1 distribution from HP and should be on the Web at http://www.openvms.compaq.com

Stunnel Server Installation on OpenVMS

I had just built my first OpenVMS 7.3-1 system in my garage and looked at all the new Open Source tools that were beginning to be integrated into OpenVMS. Wanting to install Stunnel between a Linux system and an OpenVMS system, I called on another OpenVMS expert, Steve Smiley. Now Steve has been a strong advocate for SSH for years and continually asked for SSH on OpenVMS. Asking him for some help doing Stunnel was more than an acid test for encrypted Telnet and how it was going to work across the Stunnel. Figures 4-1 through 4.26 illustrate a Stunnel server installation on OpenVMS.

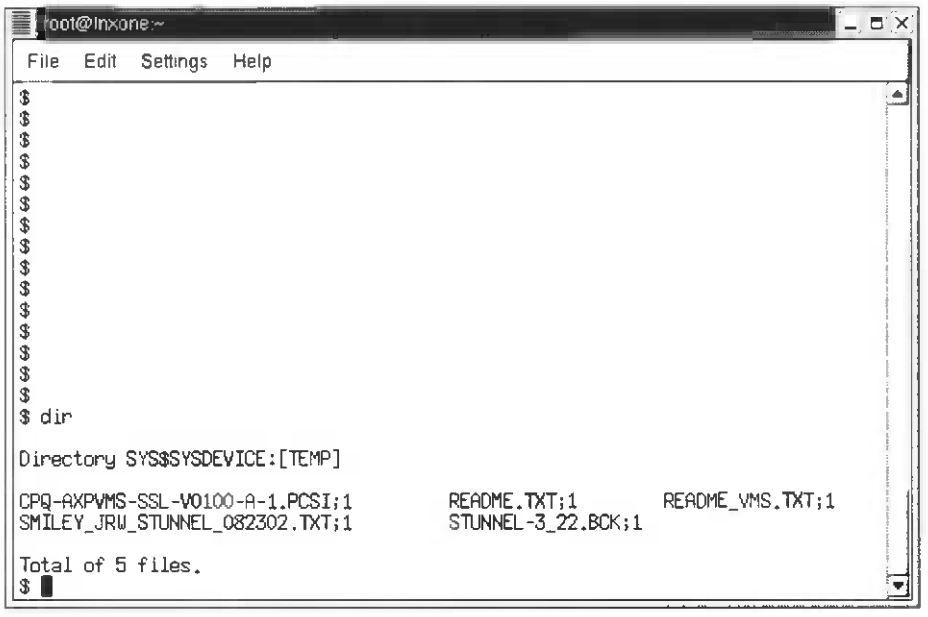


Figure 4-1 First things first. Log in to the OpenVMS system as "SYSTEM." You will need the SSL PCSI install kit and the Stunnel backup saveset.

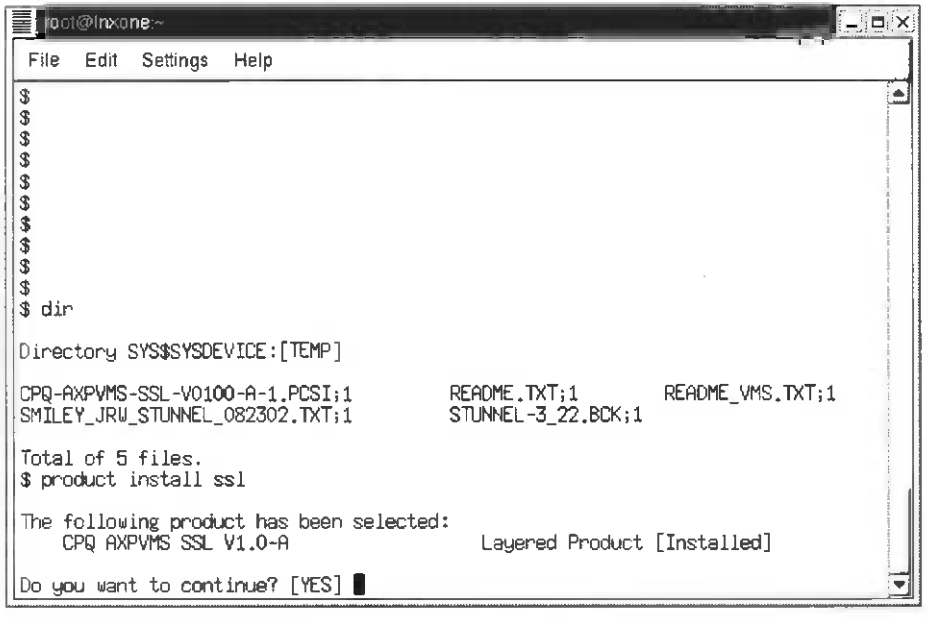


Figure 4-2 Run Product Install on the kit.

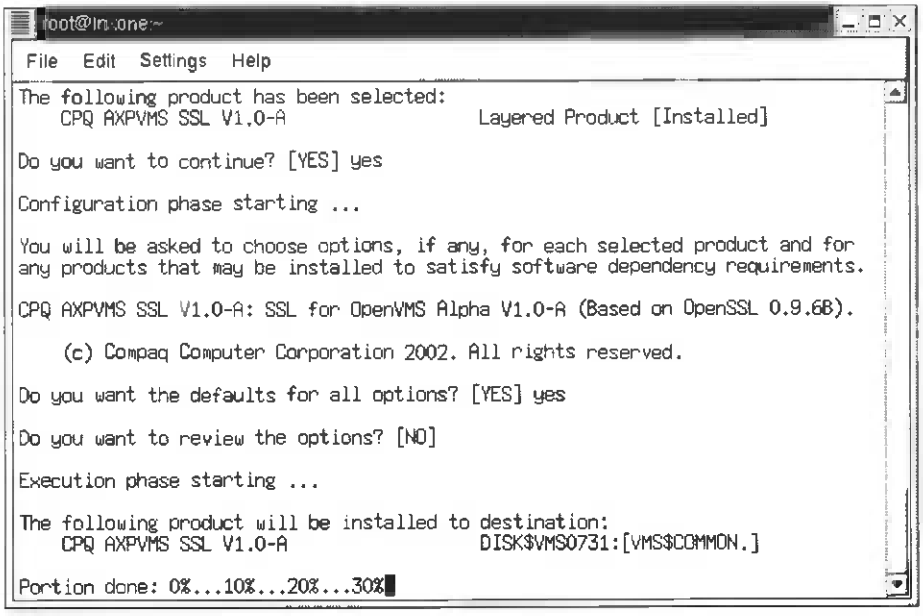


Figure 4-3 The installation will continue counting out the percentage complete.

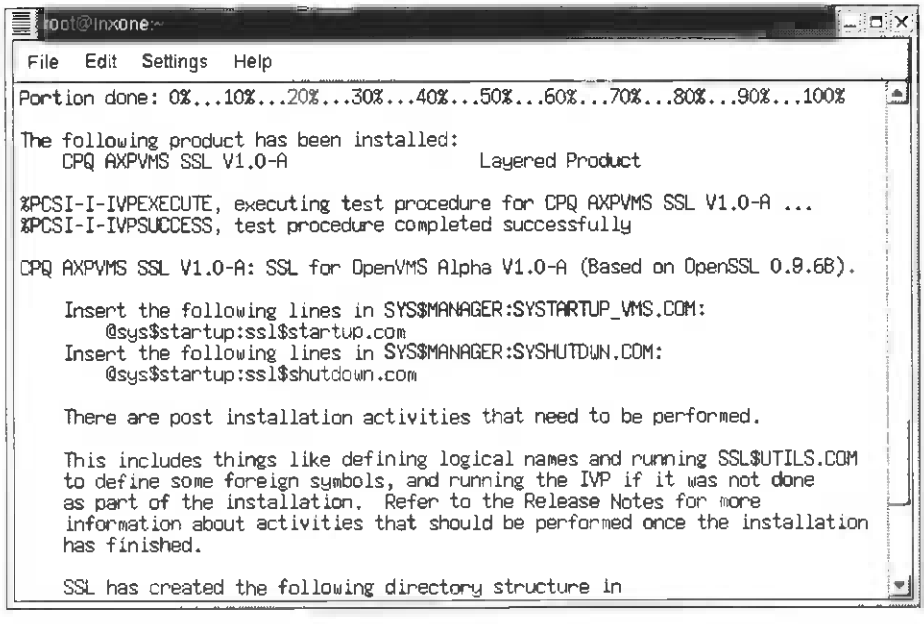


Figure 4-4 Vital postprocessing information is delivered from VMS installations. Capture this information.

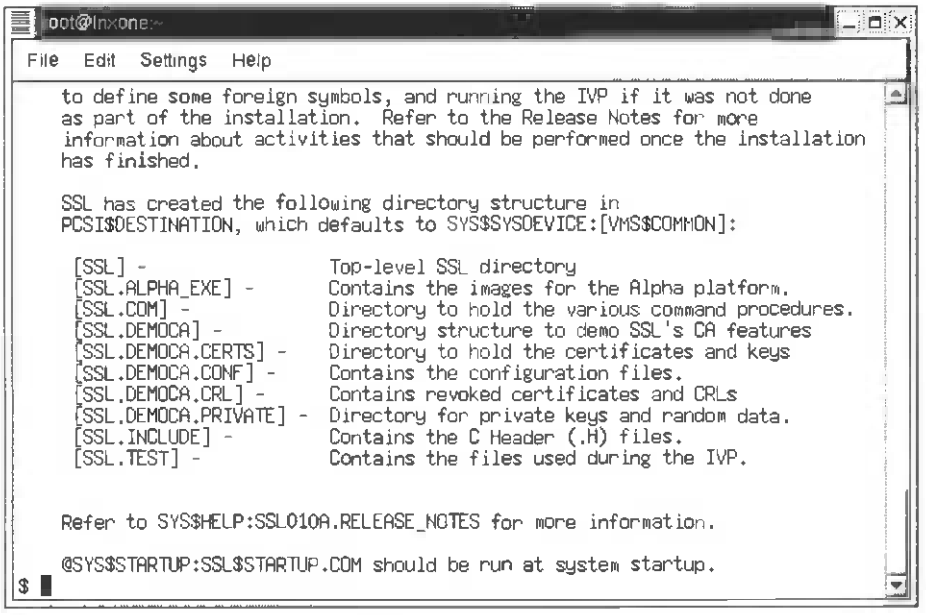


Figure 4-5 Information about where the files are located should also be saved.

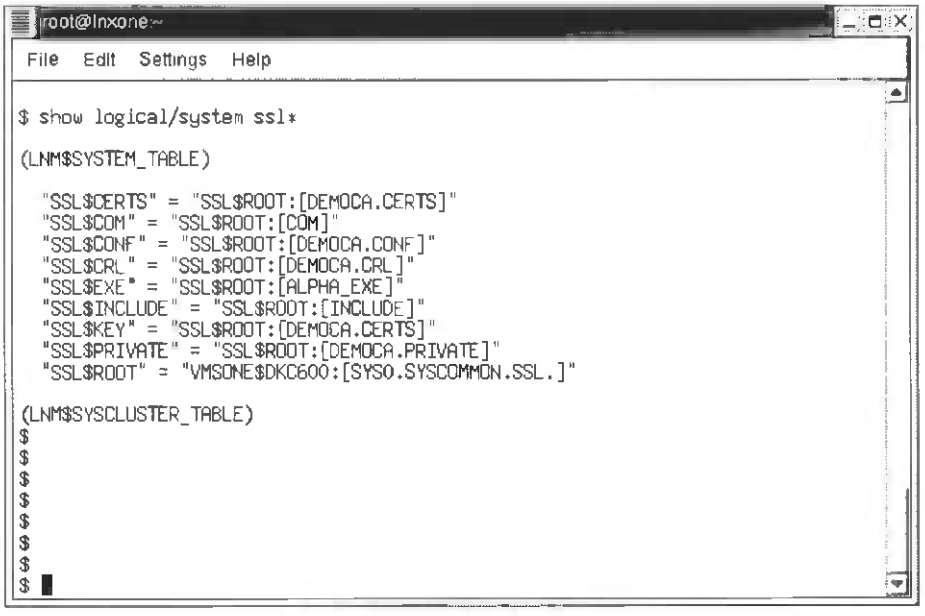


Figure 4-6 Check to see if the OpenVMS logicals are defined.

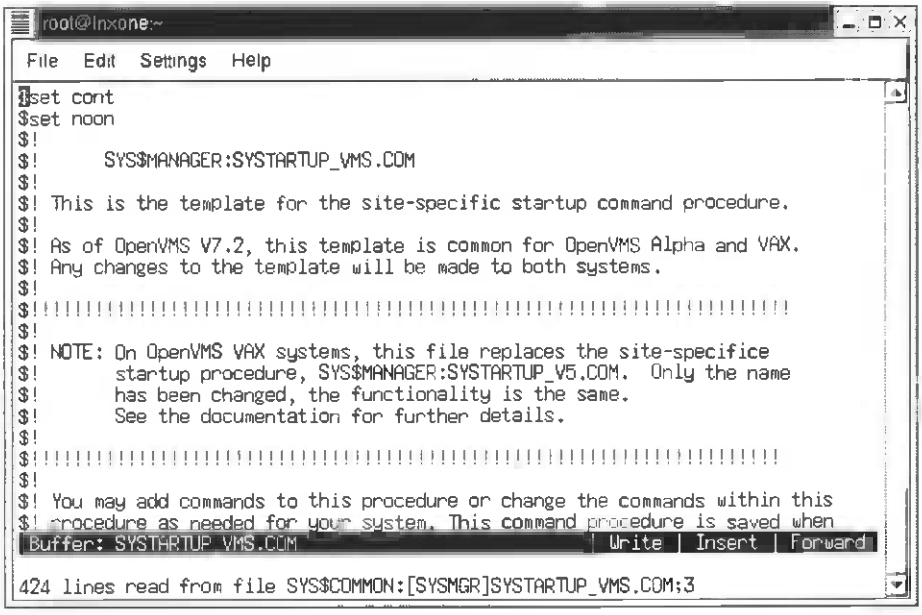


Figure 4-7 Edit the OpenVMS system startup script/command file to start up SSL upon reboot.

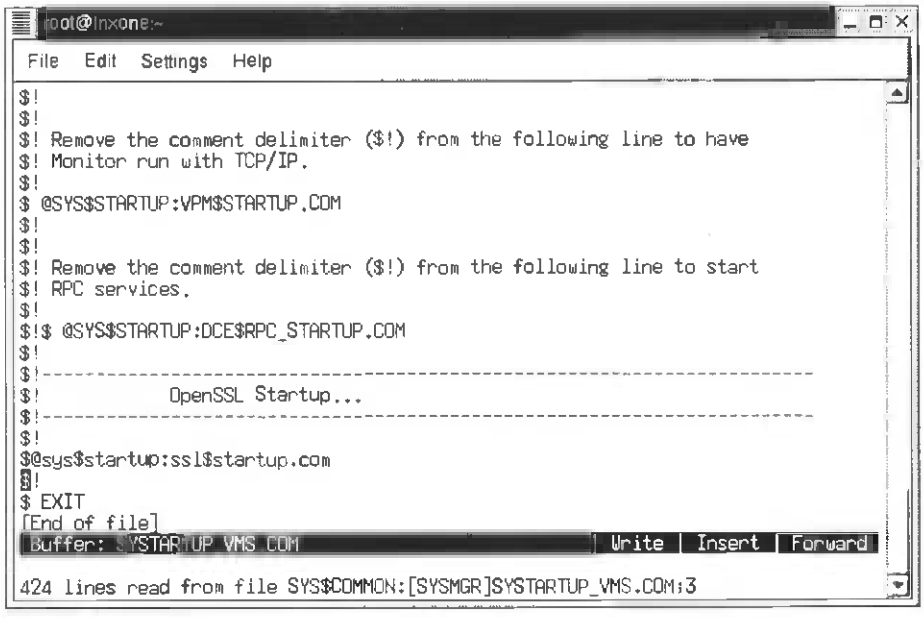


Figure 4-8 Add the line for SSL at the bottom of the startup file (after TCP/IP startup).

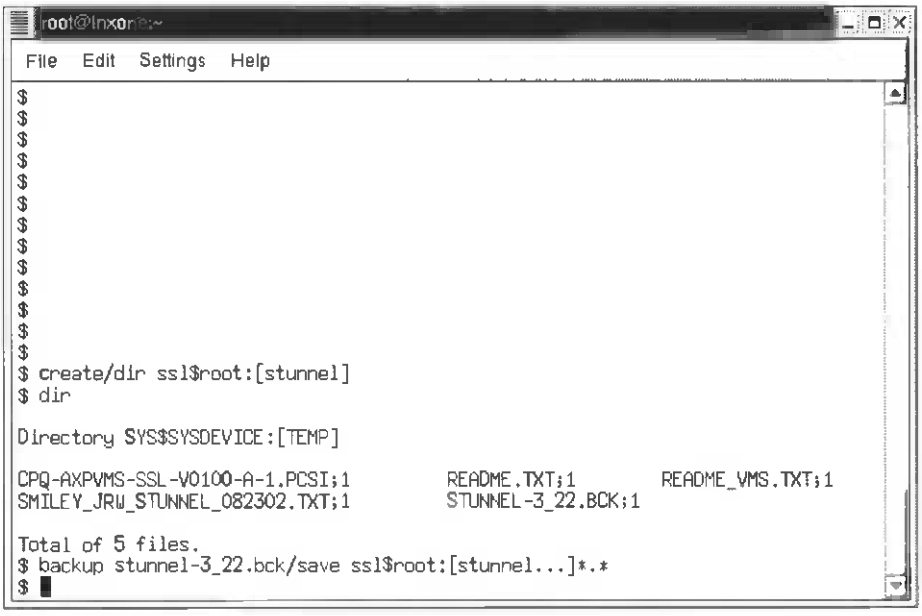


Figure 4-9 Restore the Stunnel backup saveset kit from your temporary directory to the main SSL directory.

Figure 4-10 Run the SSL certificate tool.

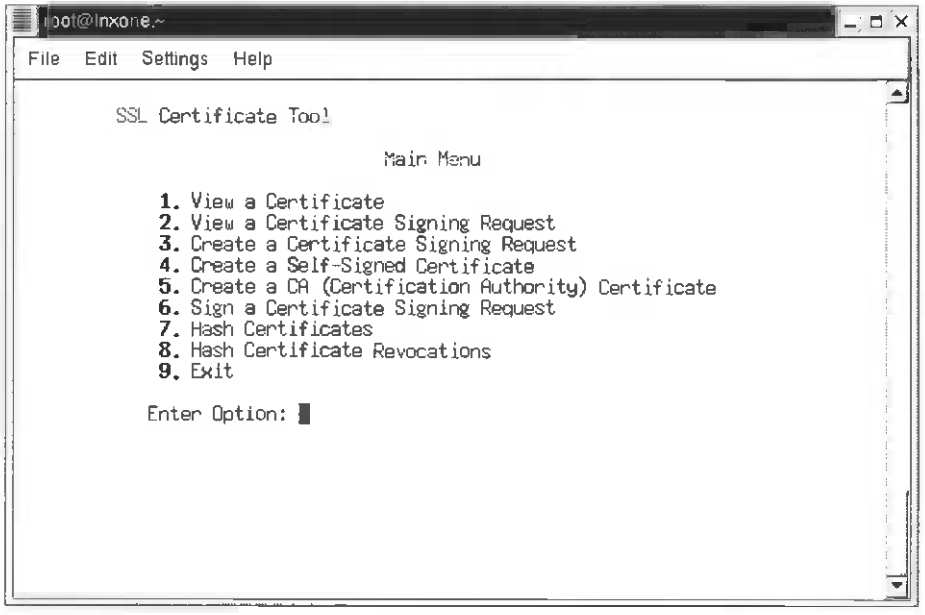


Figure 4-11 The SSL Certificate menu—choose choice 5 to creat a certificate of authority.

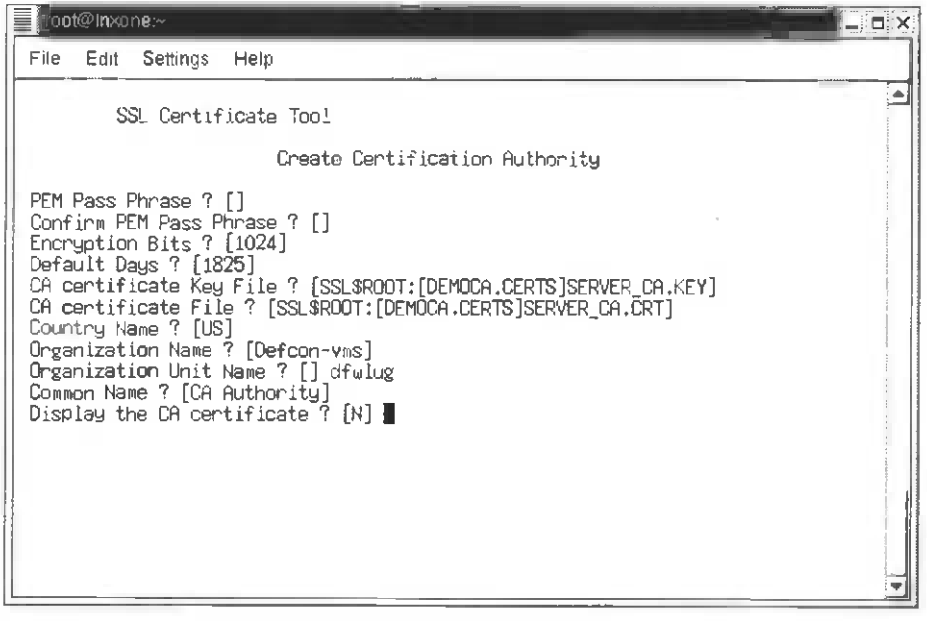


Figure 4-12 Choice 5 will take you through several questions. Use the defaults

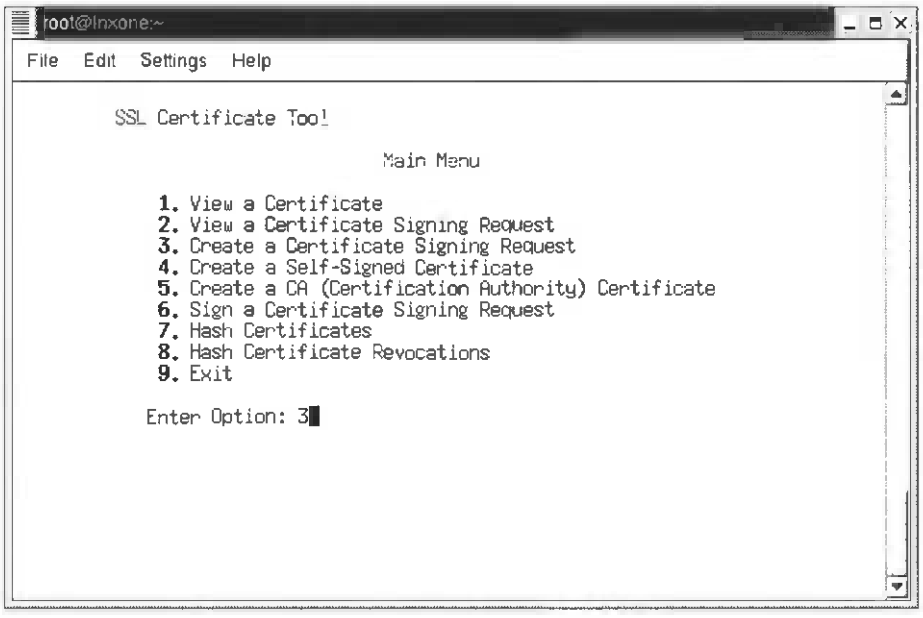


Figure 4-13 Continue and choose choice 3 to create a certificate signing request.

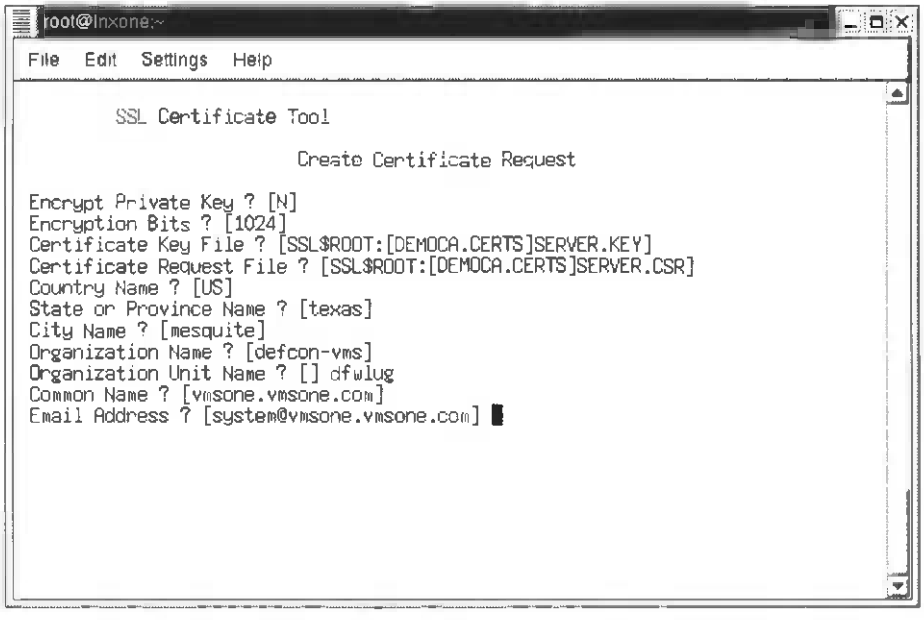


Figure 4-14 Use the defaults and your local information.

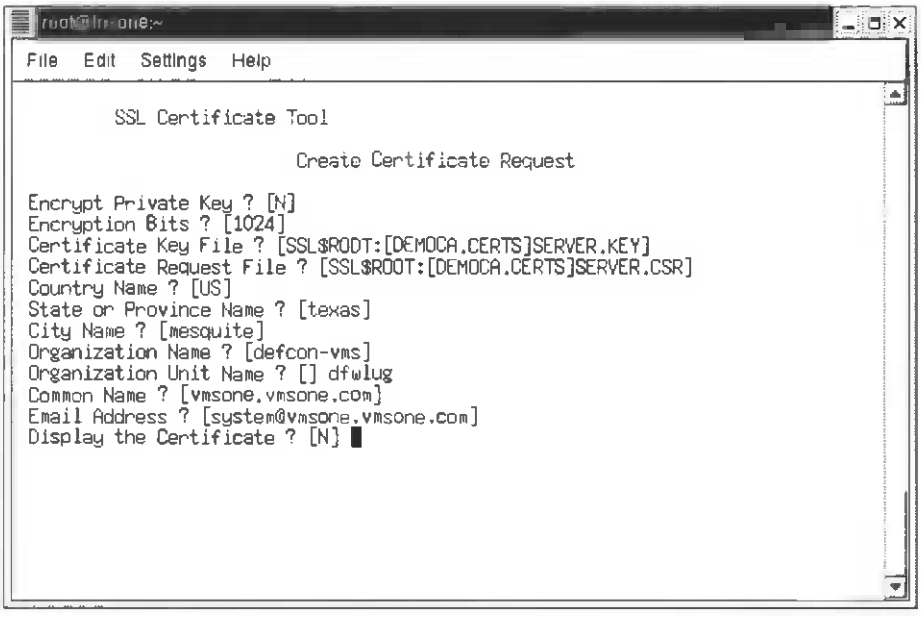


Figure 4-15 Choose choice 5 and create a local certificate request.

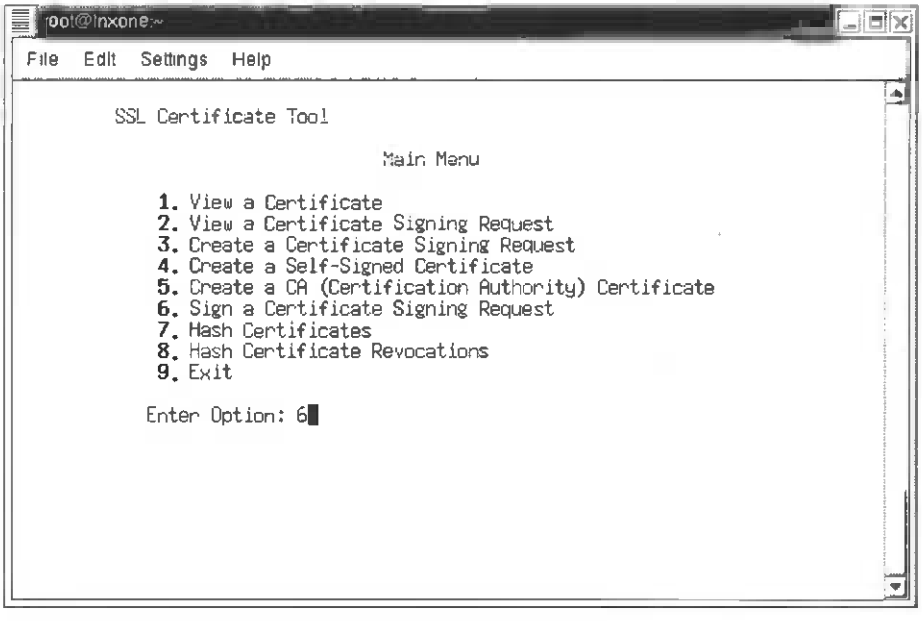


Figure 4-16 Now choose choice 6 and sign a local certificate request.

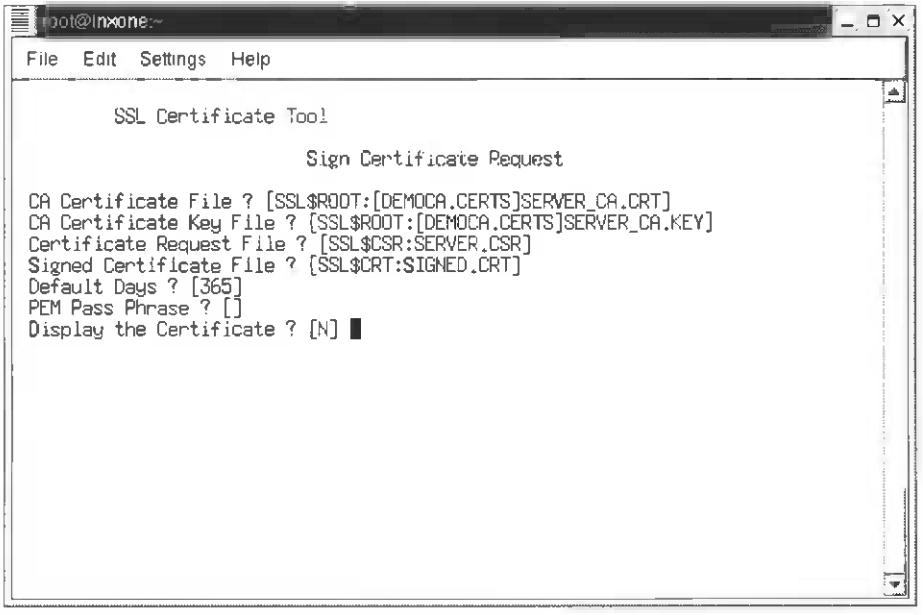


Figure 4-17 Again, use the defaults. Save the PEM passphrase and note the case you used—you'll need it shortly.

Figure 4-18 Exit out of the SSL Certificate menu.

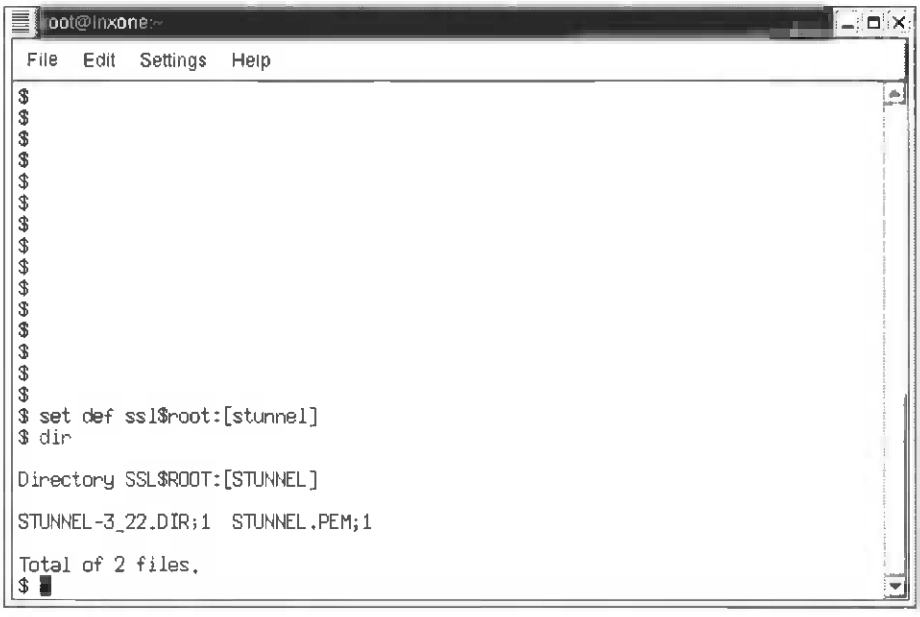


Figure 4-19 Set the directory to the Stunnel location and you'll find a STUNNEL.PEM file.

Figure 4-20 The .PEM file is the local certificate, but of course it's encrypted!

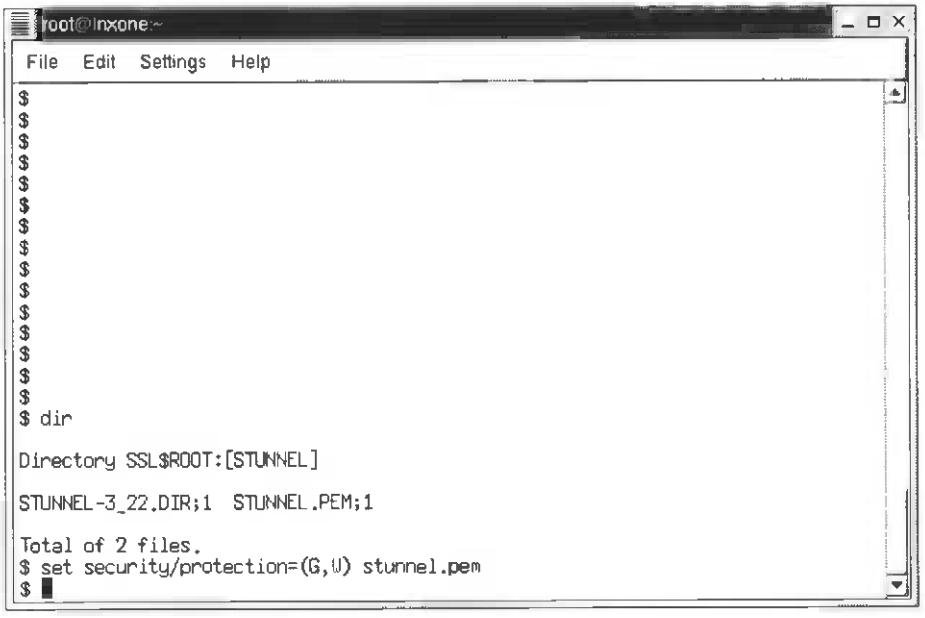


Figure 4-21 Set the protection on the Stunnel.PEM file.

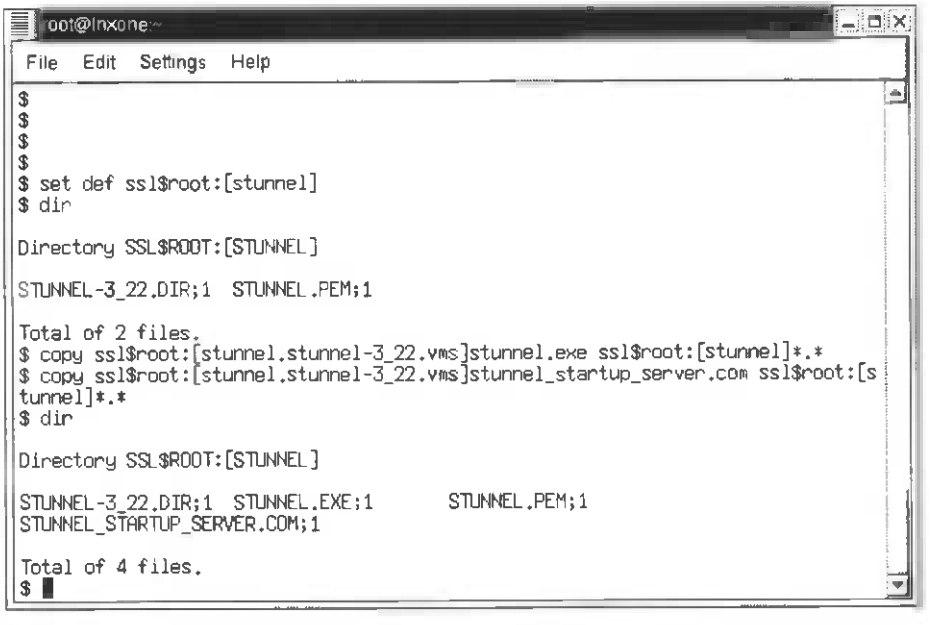


Figure 4-22 Copy the Stunnel.EXE file and the server startup script file to the Stunnel directory.

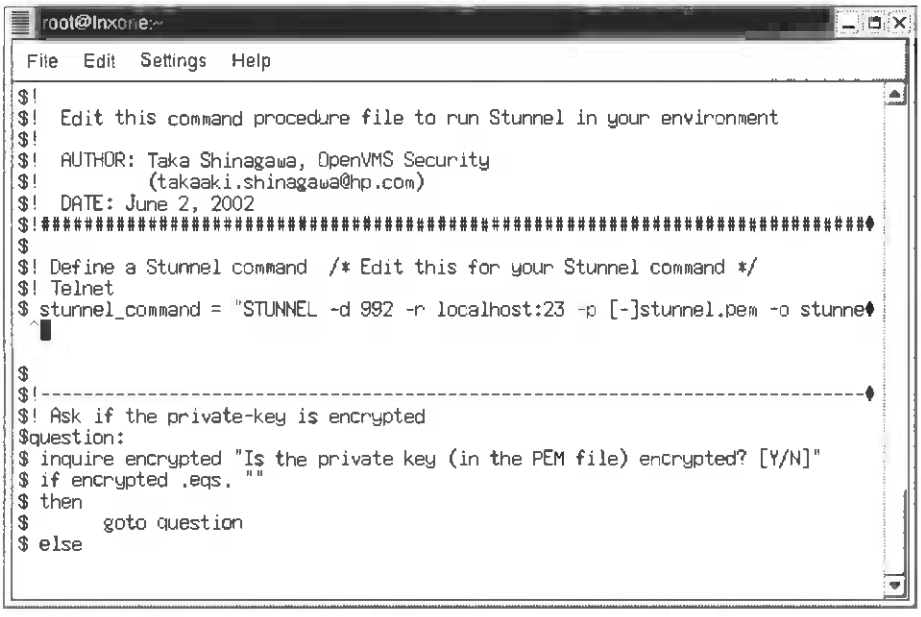


Figure 4-23 Edit the stunnel_startup_server.com file and locate the "stunnel =" line ; comment it out with an "!"

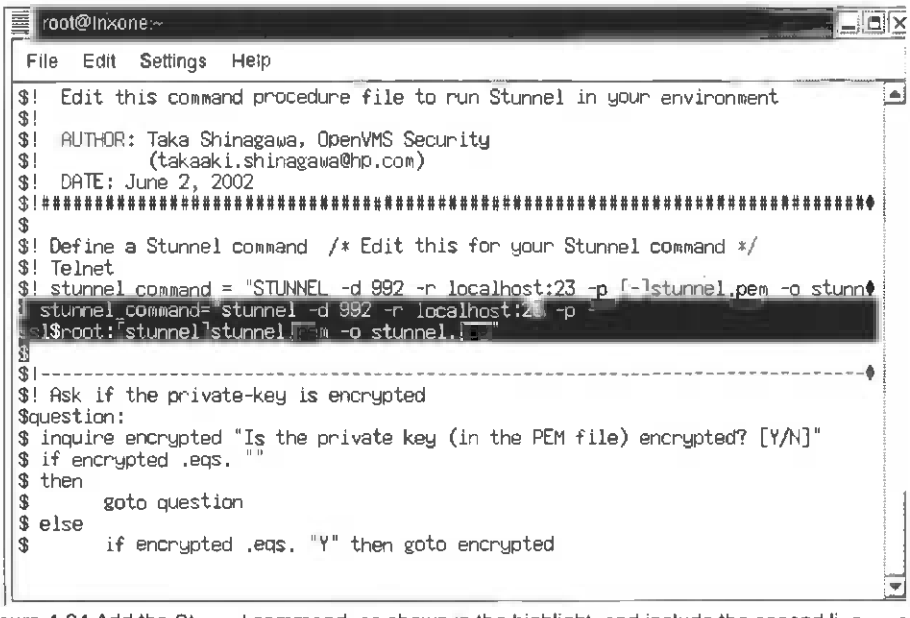


Figure 4-24 Add the Stunnel command, as shown in the highlight, and include the second line as a single command line one space away from the –p (remove the extra "-"). Save as a new version of the DCL script.

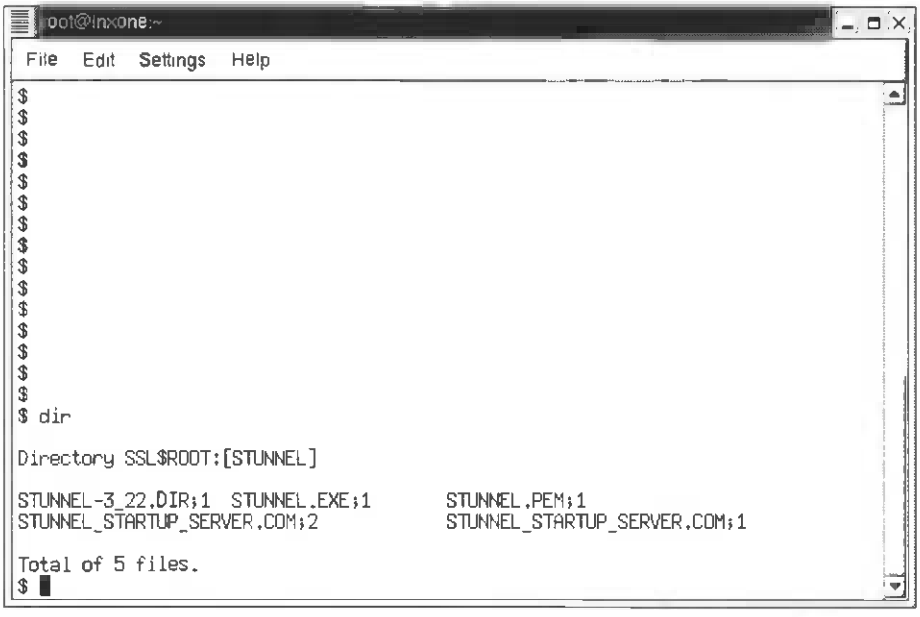


Figure 4-25 Verify that there are now two versions of the stunnel_server_startup.com file.

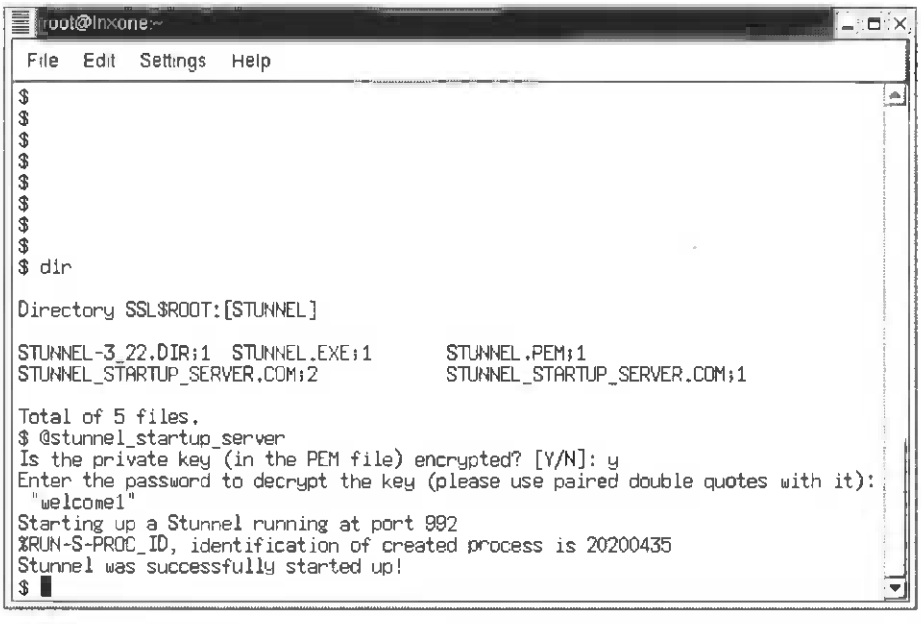


Figure 4-26 Run the stunnel_server_startup.com script. Use the same password you encrypted for the PEM certificate in uppper- and lower-case (remember to use double quotes around this password—always!).

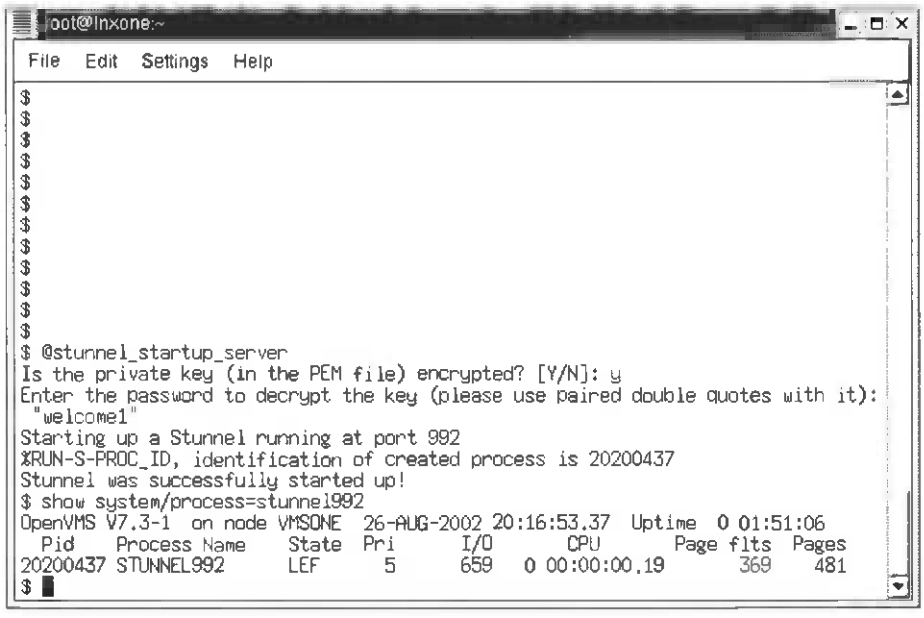


Figure 4-27 Perform a show process to make sure the Stunnel process is running.

Stunnel client installation on Linux

Figures 4-27 through 4-34 illustrate a Stunnel client installation on Linux.

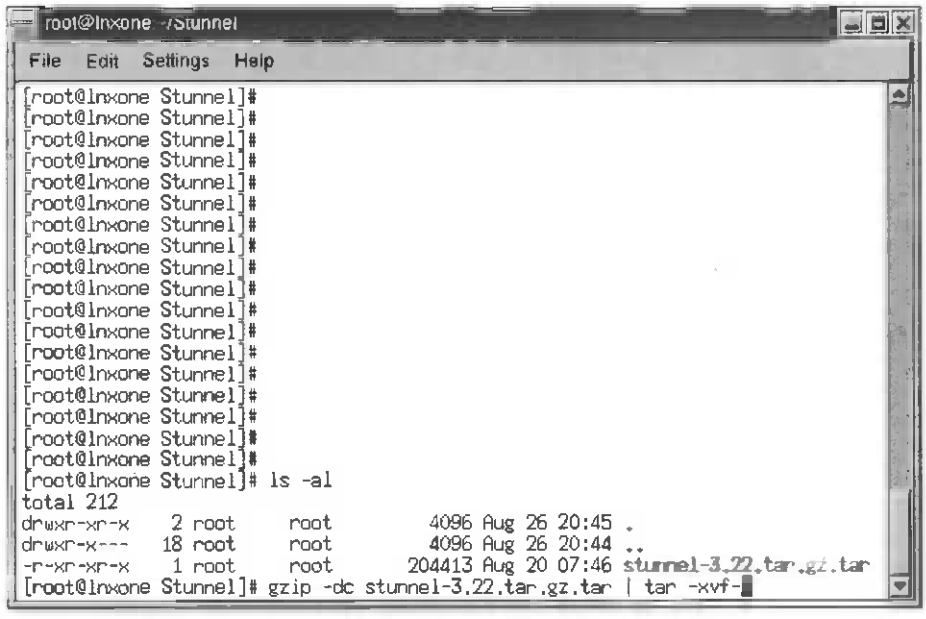


Figure 4-28 Begin in the /root directory with your file. Unzip the Linux Stunnel kit and PIPE it through an Untar command.

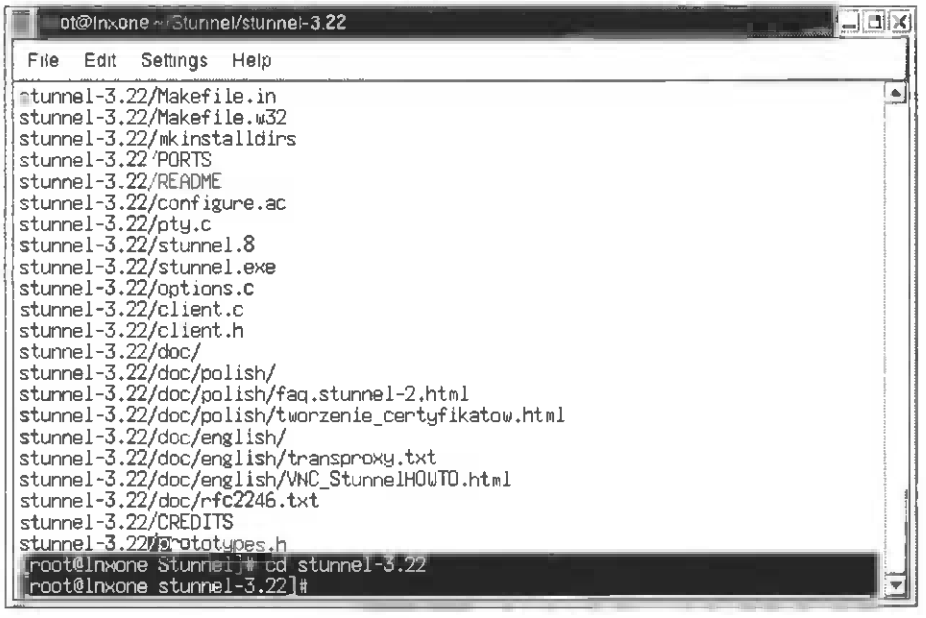


Figure 4-29 The unzip and untar will create a subdirectory structure for you to build Stunnel. After the command finishes, change the directory to the stunnel-3.22 directory.

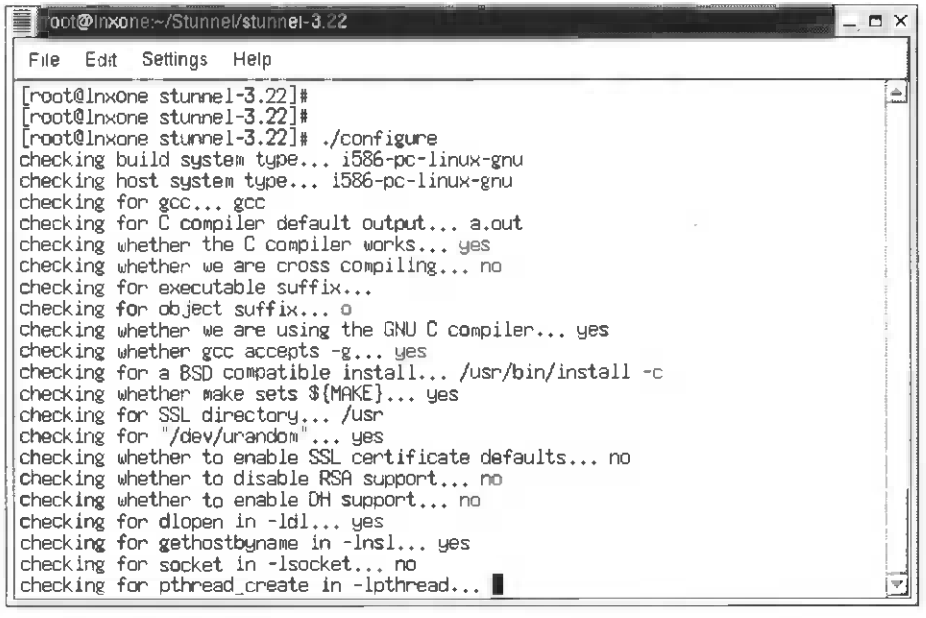


Figure 4-30 Execute the ./configure command to create a make script for the Stunnel.

Figure 4-31 Wait until the make command has been made and the ./configure completes.

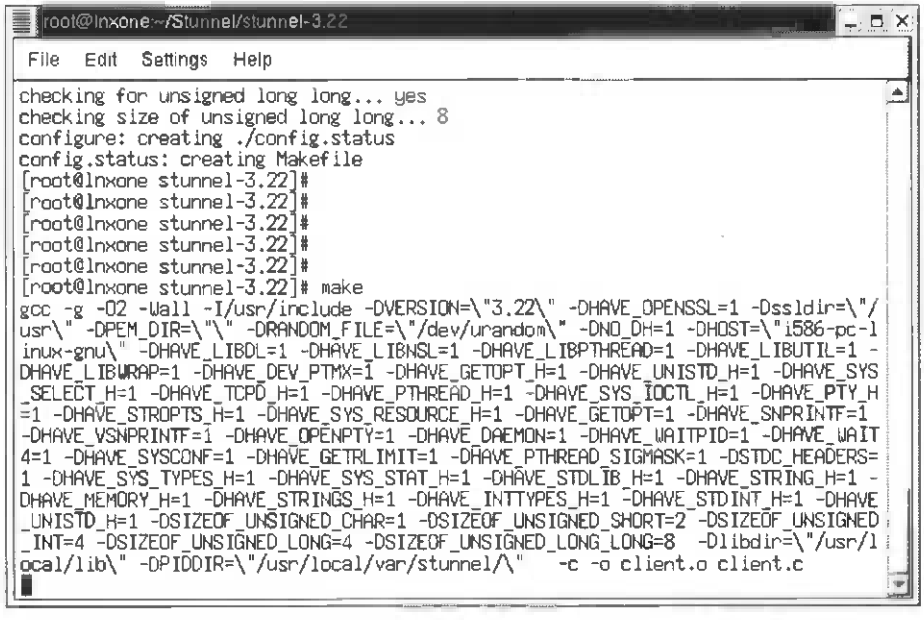


Figure 4-32 Execute make to build the Stunnel program (you must have cCinstalled on the Linux system).

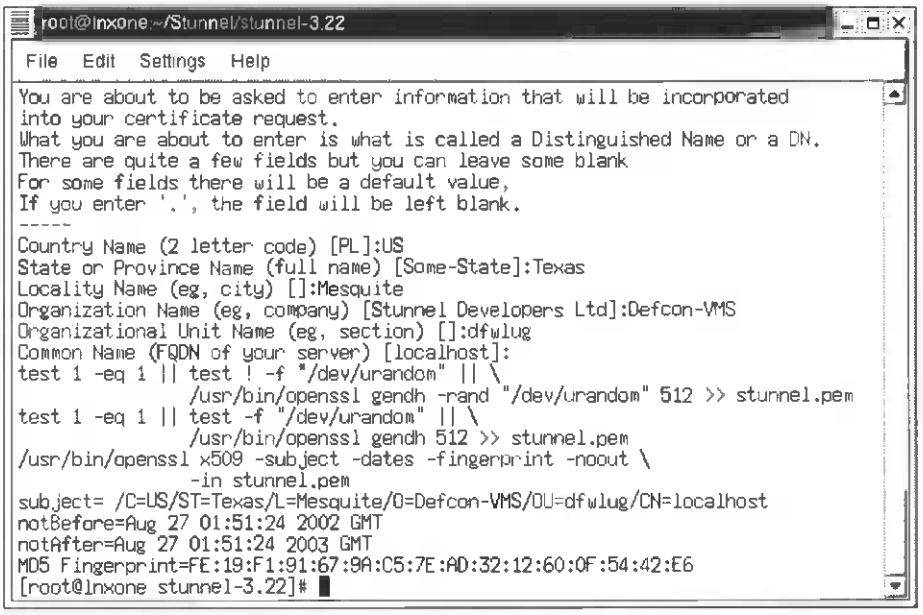


Figure 4-33 Answer the questions from the make script so it can create a stunnel.pem file for the Linux system.

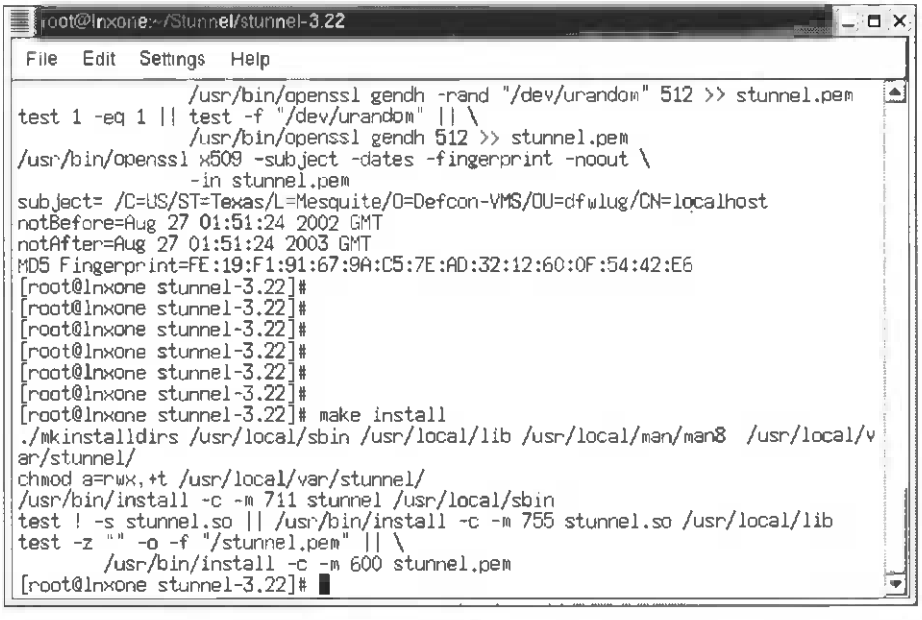


Figure 4-34 Execute make install to complete the installation of the Stunnel programs/libraries into the production system.

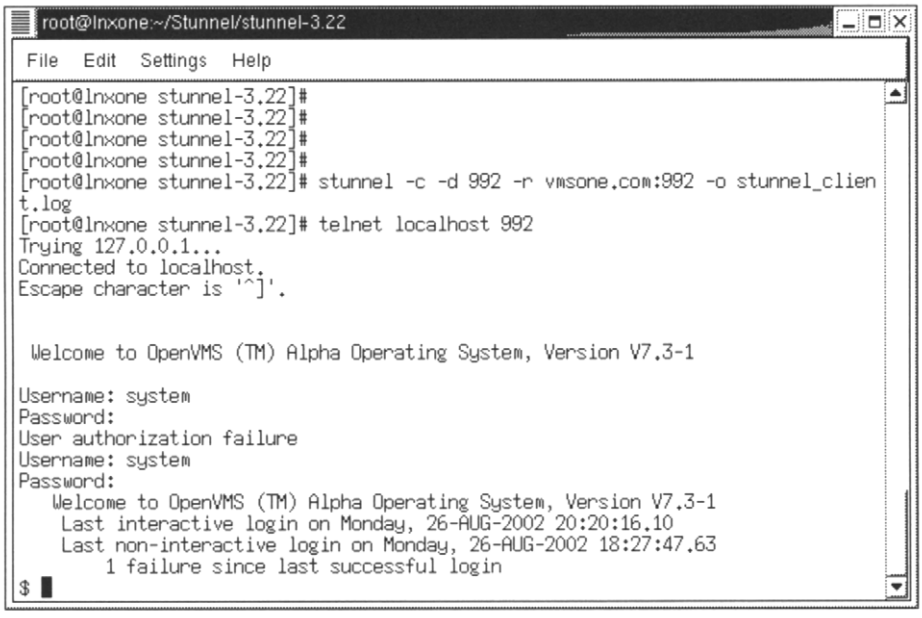


Figure 4-35 Execute the Stunnel client command. Notice it looks as if you are Telneting to a local host but are being redirected to an encrypted tunnel to OpenVMS on port 992!

Congratulations! You just got Stunnel working between a Linux and an OpenVMS system!

Security in an Insecure World

All the OpenVMS engineers I know take VMS security very seriously. I believe that as Open Source software becomes more and more utilized by all systems, the winners (from a security standpoint) will be the companies that pay attention to the details and look for security holes and how to fix them in Open Source software before it gets distributed to customers and systems. Commercial SSL implementation seems to be one of the main directions for security and tools today—for commercial systems and Open Source systems such as Linux.

In the OpenVMS 7.3-1 documentation set there are three volumes available about system security. One is the updated *Guide to Systems Security,* which has kept OpenVMS protected for the last 25 years. The other two volumes are new. They are *Open Source Security for OpenVMS Alpha Volume 1—Common Data Security Architecture* and *Volume 2—Compaq SSL (Secure Sockets Layer) for OpenVMS Alpha.* That's how important Open Source is going to become over the next few years regarding security and authentication. OpenVMS already has two volumes in its doc set devoted to the use and security of Open Source tools. It's my recommendation that OpenVMS

systems people read all three of these books; they represent the future needs of OpenVMS customers.

Other sources for OpenVMS security information include http://vmsone.com/, and Point Secure's OpenVMS product, "System Detective," at http://www.pointsecure.com/. There is information on these Web pages about how to battle harden even the most well-protected OpenVMS system!

As for Linux systems, look on the OpenSSL pages at, http://www.openssl.org/ and http://www.linuxsecurity.com/. These should get even an experienced UNIX administrator into a Linux fortress.

For security in general, the future is encryption, intrusion detection on the host, and network intrusion detection between systems. Today encryption is represented by SSL, intrusion detection by products such as Point Secure's System Detective, and network intrusion detection with products such as Checkpoint's firewall products. Keeping and using all three of these methods in your system security arsenal will ensure that your systems are protected, and you will be able to continue to use your systems even in the face of a security attack.

Open Source software will represent a risk because of security holes, but patches, updates, and all those new Open Source applications will help keep the risks of Open Source at bay. Commercial vendors will also step in and keep standard releases clean, patched, and available for their customers (remember: Open Source doesn't mean free; the implementation will cost money to maintain for commercial systems).

Risks and benefits will chart the course computer security will take over the next few years. With the right tools and some up-front planning Linux and OpenVMS can deliver solid security and functionality for any data-center environment.

Chapter 5 — X Windows Applications: X Windows Interoperability with VMS and Linux

X Windows, VMS, Linux, and a Bit of History

In 1988, a 10-year-old VMS operating system was migrated from a display list window manager, "VWS," to the X Windows–based "DECwindows" for all of its OpenVMS graphical workstations. Then, as X Windows continued to evolve, VMS added the Motif Window Manager and by 1997 even had the UNIX CDE (Common Desktop Environment). Linux evolved in the 1990s, building to these already established by standards: X Windows for interoperability, Motif for standard application-style management of X Windows processes and sessions, and CDE for a common look and feel in the UNIX and X Windows workstation community.

Today, we take these standards for granted in the workstation and Open Systems world, and multiplatform, multimedia Internet browsers may, in fact, be the final word on the merits of X Windows versus local Windows Graphical User Interfaces (GUIs). If we look back, it might be worth noting why OpenVMS changed from a display-list-technology-based GUI to an X Windows–based GUI for its workstation displays.

During the 1980s, display-list window managers had enjoyed their flower. By the mid-1980s graphical heads were still standalone with display computers connected to a computer system that required graphical/Windows output. These two specialized computers communicated (usually) across serial or parallel cables, just as if the primary computational system were talking to a printer or plotter. Plotter is the key to understanding this relationship, because the same commands that drove plotter pens to draw lines and automatically fill areas on paper became the basis for commands to plot graphics on the display-list computer heads of the 1980s. As the first workstation emerged, these computer/display workstations were little more then Pen Plotters driving raster displays. Then, as standalone graphics-display computers were reduced to so many chips with a display screen, the commands that drove their graphics continued to be the same local plotting commands and languages that ran the previous generation of displays. Each company had its own standards, chip sets, and commands and had little regard for any hardware or network abstraction for graphical commands. Graphical programs needed to be different for each workstation vendor's machines. Graphical libraries for developers (such as GKS and PHIGS) were designed to help in developing programs that could run on a variety of hardware, but that wasn't enough. There needed to be a standard way of displaying process graphics across networks and different computers.

Then, in 1985, DEC funded MIT to research "Project Athena." Out of that research came X Windows, Kerberos (the security model now being adopted by Windows

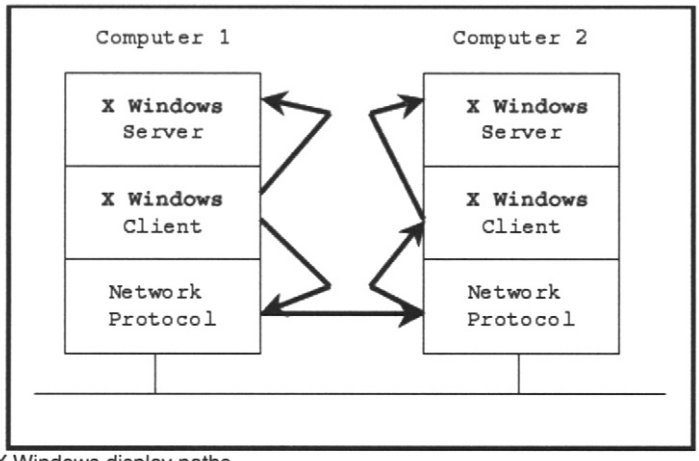


Figure 5-1 X Windows display paths.

2000), and the beginnings of DCE. OpenVMS and many UNIXes adopted X Windows, because, unlike display-list managers or other local GUIs that could only display graphics and control to the local system, X Windows was designed and optimized to have its graphical interface networked. X Windows was designed for local or remote application display across many standard network protocols (TCP/IP, LAT, DECnet, even SNA). In X Windows (unlike window managers that controlled only local hardware, such as a local plotter), graphical output was architected and optimized for transmission across networks to other X Windows displays on any X Windows computer in the network! This made applications easy to display between multiple systems or made it easier still to have many different graphics applications brought to a single graphics display regardless of operating system of origin! That level of Open Systems interoperability is why DEC chose X Windows as its GUI OpenVMS workstations and servers in 1988 and why HP continues to support X Windows to this day!

Fast forward 17 years and 6 major X Windows software releases, and add to this time-tested networked graphical-display system today's security, new graphical cards, Alpha system computational performance, and faster and faster network connections. It's easy to see why a low-cost Linux system would make a perfect, inexpensive integrated workstation to display graphics and applications from an OpenVMS server using standard X Windows tools!

Displaying OpenVMS X Windows Applications on a Linux Workstation

There are two parts to displaying OpenVMS applications on a Linux workstation. The first part is to log in to the Linux workstation and set security to accept remote display from the OpenVMS applications. The second part is to Telnet to the OpenVMS workstation and redirect X Windows output to the Linux machine and then run the

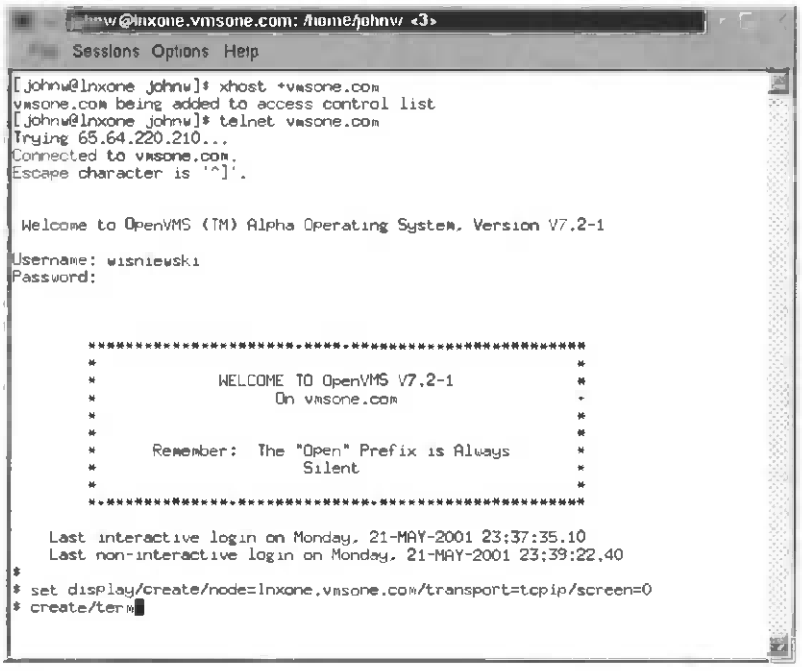


Figure 5-2 You should have redirected your OpenVMS create/terminal command to display its results on the Linux workstation screen!

application. In order to perform this function you must have two accounts, one on the Linux workstation and one on the OpenVMS server.

Let's go through the command process: Log in to an X Windows session on your Linux box. Open an interactive terminal window and type (see Figures 5-2 and 5-3).

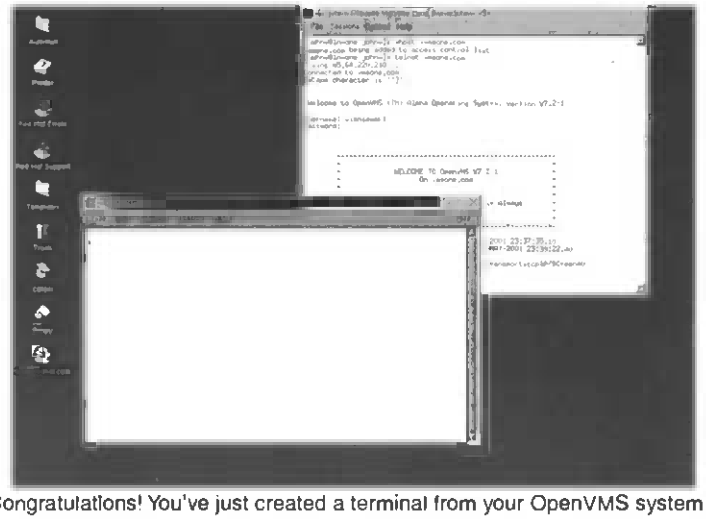


Figure 5-3 Congratulations! You've just created a terminal from your OpenVMS system to your Linux box!

Command Controls for Your X Windows Sessions

Linux Command: xhost

The xhost command on your Linux box controls who can write to your X Windows server display, as follows:

- xhost + allows every host in your network to send X Windows displays to your screen.
- xhost – allows no one in your network to send X Windows displays to your screen.
- xhost +nodename.com allows just "nodename.com" to send X Windows displays to your screen.
- xhost with no arguments tells you who is authorized to send X Windows displays to your screen.

Remember: You can always block all networked X Windows displays by typing "xhost –" in a local Linux terminal window. Be mindful of the security implications of this command for your workstation, especially in a public network! If in doubt be sure to turn off access with an "xhost –". (A practical joker could even overlay your Linux window manager with his or her window manager if you aren't careful!)

OpenVMS Commands: SHOW-SET/DISPLAY and CREATE/TERMINAL

In your new DECterm on your Linux box, type the information shown in Figure 5-4.

- *Device*: WSA4: [super] means that this OpenVMS terminal (process) will redirect its X Windows output to a graphical display device to accept this input.
  - [super] means that all X Windows applications run from this process will be displayed to WSA4.
  - [user] would mean that only the next X Windows application run from this process would be displayed to WSA4.
- *Node*: This is the system where the X Windows server display is located.
- *Transport*: This is the networking protocol that is being used.
- *Server*: Some systems can have multiple instances of X Windows servers running. Zero is the default.

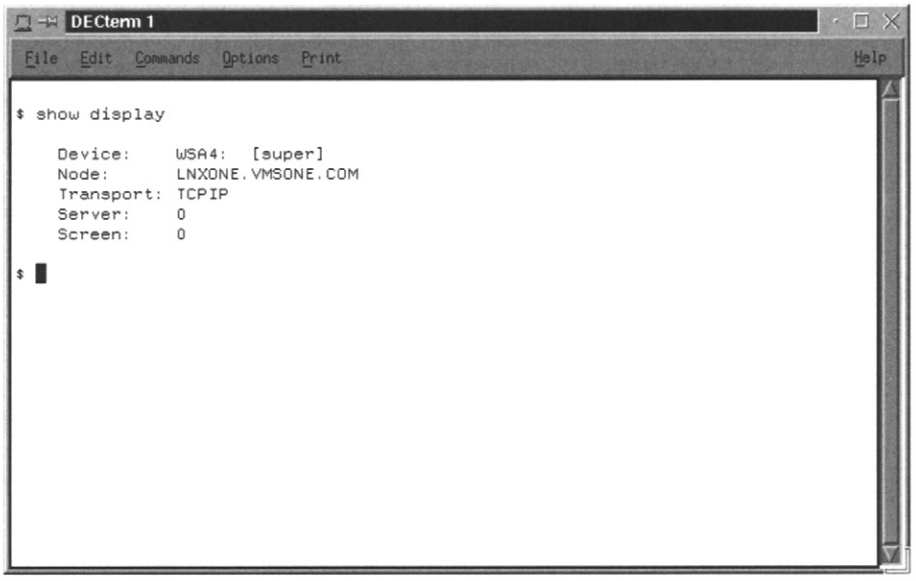


Figure 5-4 The SHOW DISPLAY command tells where X Windows applications run from this window (on the OpenVMS server) will try to display.

- *Screen*: X Windows server displays can have more than one display screen either in software or connected to two or more graphical cards with two or more monitors attached. Screens are numbered 0 + to as many graphical heads as the server contains.

All of these parameters can be changed/redirected by issuing

```
SET DISPLAY/CREATE/NODE=node.com/transport=tcpip/screen=0
```

- /TRANSPORT= (use TCP/IP if you've installed LAT or DECnet in your Linux kernel).
- /EXECUTIVE_MODE (optional) creates an executive mode WSA*x*: device for all processes on your system to use (must have privileges to use this).
- /SUPERVISOR_MODE (optional, default) creates a supervisor mode WSA*x*: device for your current processes (and subprocesses) to use.
- /USER_MODE (optional) restricts the next command.

This command will not change running X Windows display from an OpenVMS server, but it will change where the output is for all new programs that are run!

The CREATE/TERMINAL command on VMS can be added to for additional functions.

- /DETACH will create a detached background process for the terminal similar to the UNIX/Linux command xterm&.
- /NOPROCESS will make the terminal ask you to log in as a VMS user. It is useful if you want to log in as a different user while you're still logged in with your other X Windows terminals.
- /BIG_FONT / LITTLE_FONT /CLI /WINDOW ATTRIBUTES and many other details about your VMS terminal can be modified as the terminal is created. Be sure to check out the OpenVMS help utility HELP CREATE/TERMINAL and create a terminal that is just right for you.

OpenVMS X Windows Applications

Let's look at the X Windows applications that come with OpenVMS when you install X Windows (see Figure 5-5).

All of these standard applications have the prefix DECW$ are installed when (if) DECwindows is installed on an OpenVMS server. DECW$CALC, DECW$PAINT, DECW$CALENDAR, DECW$CLOCK, DECW$MAIL, DECW$CARDFILER, DECW$MESSAGEPANEL, DECW$NOTEPAD, and DECW$PUZZLE and are similar

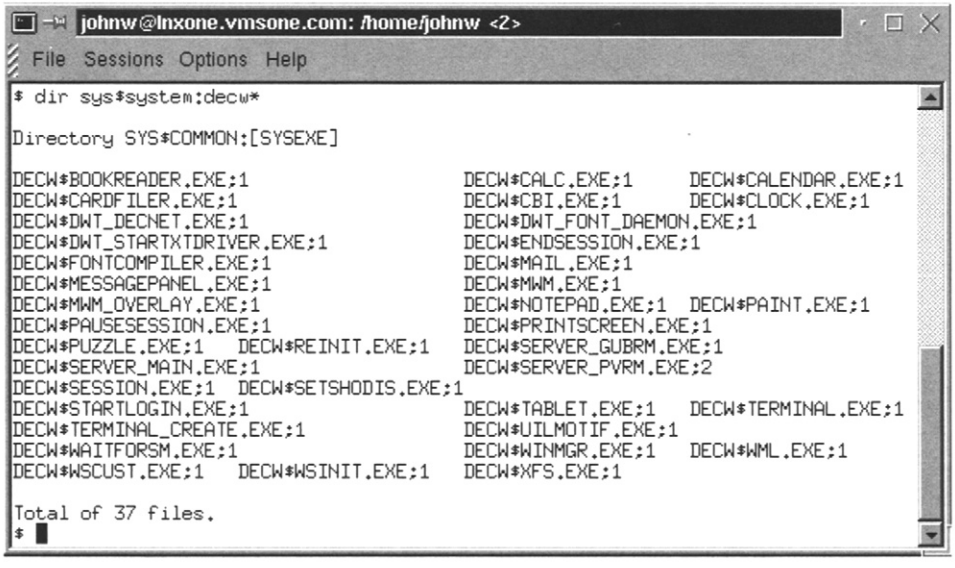


Figure 5-5 The standard VMS X Windows applications are to be found in the directory SYS$SYSTEM:.

to their Linux X Windows application counterparts. These programs can be run after defining a display by typing

```
RUN SYS$SYSTEM:DECW$CLOCK.EXE
```

RUN/DETACH SYS$SYSTEM:DECW$CLOCK.EXE creates a detached process for your program to run in and returns you to a $ prompt in your terminal window for more commands!

Regarding other DECwindow applications: Some of the applications are straightforward (such as DECW$CLOCK), but others are not so clear. DECW$TERMINAL, DECW$WINMGR, and others look like they could start window managers or create a new terminal (and process). In fact, that is what these programs are for, but they are accessed by the VMS startup programs to run X Windows on a VMS server, while others are used by the VMS Command Line Interpreter to create terminals or begin X Windows sessions; this is similar to xinit or xf86config on Linux.

Note: Do not run window manager programs directly, since they could mess up your X Windows sessions on both machines!

One way to gain easy access to the OpenVMS X Windows applications is to use either DECW$SESSION.EXE or DTSESSION.EXE session managers. DECW$SESSION is the older X Windows session manager, and DTSESSION is the CDE X Windows session manager—either one can be on a remote X Windows display. Let's construct a DCL command file that will invoke one of these programs (see Figures 5-6 through 5-13).

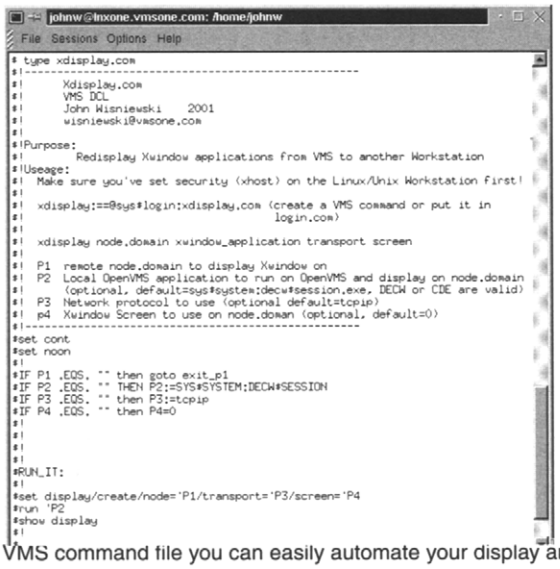


Figure 5-6 With this VMS command file you can easily automate your display and X Windows commands. The default brings up the very useful DECwindows session manager.

Figure 5-7 So how is this useful? Let's look at the pulldowns.

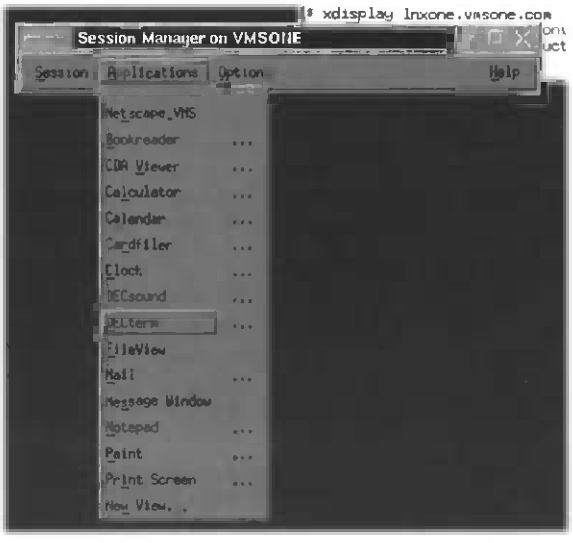


Figure 5-8 Now you can create as many VMS terminals as you need, check the status on your X Windows jobs in progress, or even change your X Windows security or startup options—all by the click of a mouse and all from the display on your Linux box!

Figure 5-9

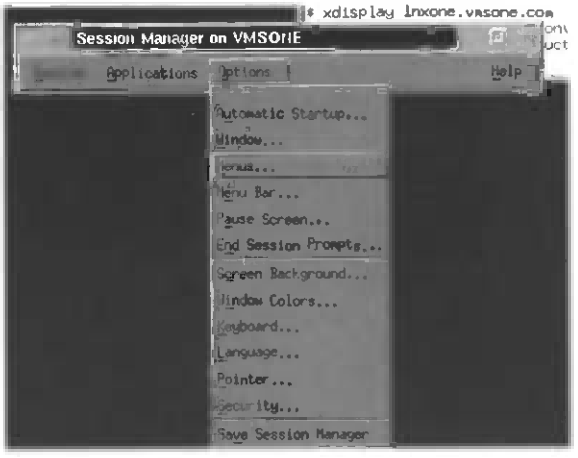


Figure 5-10 But what if you wanted to add your own commands or X Windows applications to launch from this menu? Use the Menus button on the Options pulldown for the Menu Control window.

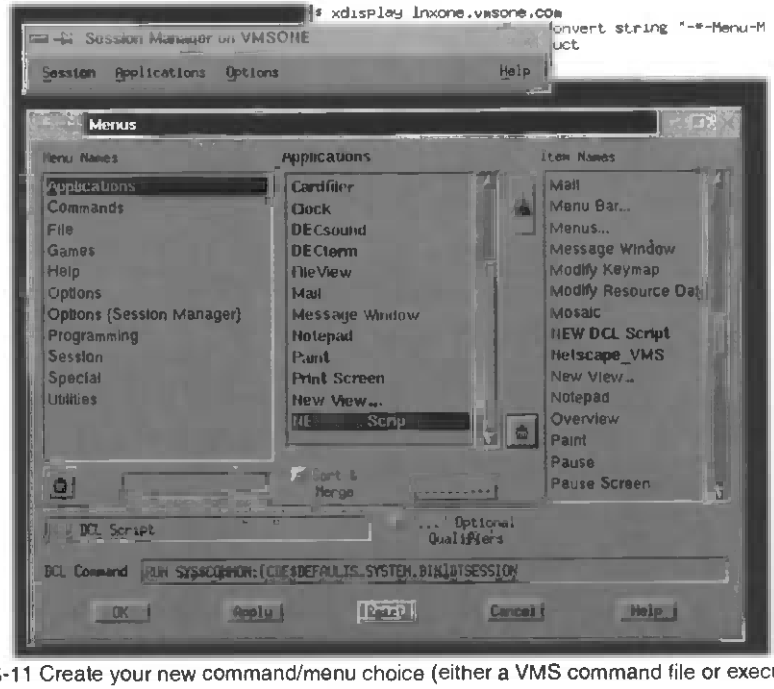


Figure 5-11 Create your new command/menu choice (either a VMS command file or executable) and add it to the appropriate pulldown menu.

Figure 5-12 Then just click on your new pull down!

Figure 5-13 Hey! Where did all these VMS applications and menus come from! I thought this was a Linux Box!

Notice that all these applications came from the VMS CDE/DTSESSION manager program at the same time the DECW$SESSION manager program and the Linux KDE session manager are running! Be careful, because any of the session managers can now log you out.

Displaying Your Linux X Windows Applications on OpenVMS

Back in the mid-1980s, when VMS workstations had a state-of-the-art graphical chip set, you were likely to have a VMS workstation on your desk as your primary workstation. Today, Alpha workstations are just as likely to use Linux or Tru-64 UNIX as OpenVMS. PCs are as likely to run Linux as Windows for the operations staff, and today's $30 PCI or AGP graphics card under Linux or Windows is quite possibly a better display engine than that three-year-old graphics card running in your Alpha workstation. The nice thing about X Windows is that you are not tied to your computer engine or tired old graphics cards; you can just redirect the graphics to enjoy the application with new eyes!

But let's assume you are lucky enough to have the latest state-of-the-art Alpha workstation running VMS on your desk (instead of a GS320 cluster in your computer room) and your Linux box's graphical card is aging a little. You can simply redirect your Linux application to your new OpenVMS display.

The procedure is exactly like the VMS to Linux redirection except that Linux commands must be issued to redirect the output to the VMS workstation; instead of the xhost command, OpenVMS uses a Security menu under the session manager.

First you must authorize VMS to use TCP/IP as a transport for X Windows (inbound to VMS), because it's not turned on by default.

Either you or your VMS systems manager should add this one little command line (see Figure 5-14):

```
SYS$MANAGER:DECW$STARTSERVER.COM
```

If you are uncomfortable changing this file, you can put the logical definition line in your SYS$MANGER:SYSTARTUP_VMS.COM. I put it in DECW$STARTSERVER.COM because TCP/IP should be turned on (with security) by default!

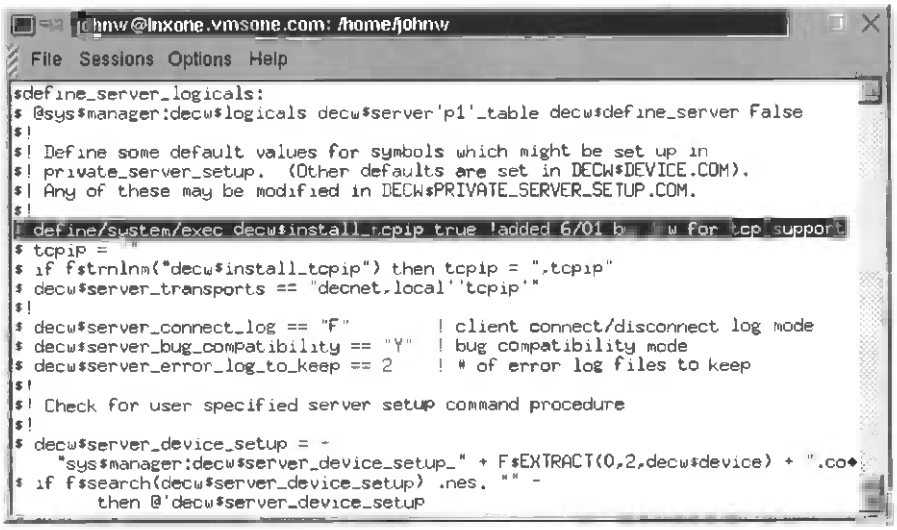


Figure 5-14 Do not modify this file in any other way or you may make DECwindows unstartable.

Now let's set security using the VMS session manager on the VMS workstation (see Figure 5-15).

Figure 5-15 Click on the Security button and you'll get into the Authorization box.

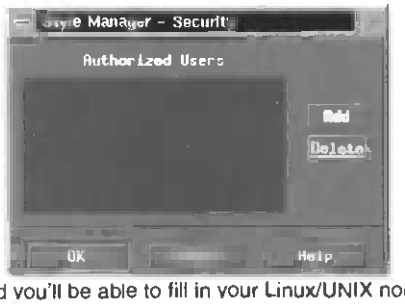


Figure 5-16 Click on Add and you'll be able to fill in your Linux/UNIX node information

For TCP/IP as a transport, always use a "*" for the user name or it won't work (see Figures 5-17 through 5-19)!

Figure 5-17 You are now ready to telnet to your Linux system and send back X Windows applications.

Figure 5-18

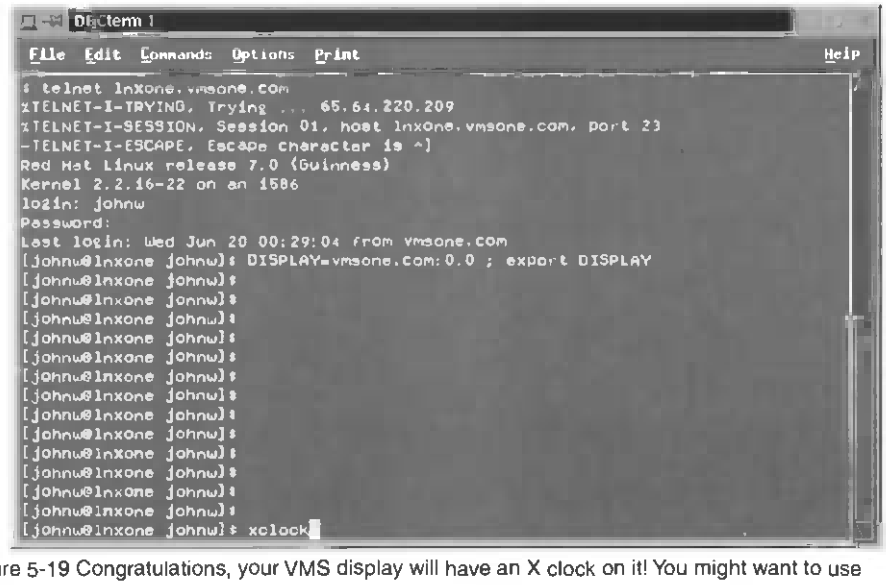


Figure 5-19 Congratulations, your VMS display will have an X clock on it! You might want to use xterm or xterm& (background) or any other X applications from your Linux box too!

And for you bash-shell impaired, you may need to change your UNIX/Linux dialects a little:

- csh:

```
# setenv DISPLAY vmsone.com:0.0
```

- sh and ksh:

```
DISPLAY=vmsone.com:0.0 ; export DISPLAY
```

Summary

X Windows is an expanding and growing part of workstations and high-end servers even 15 years after its introduction. Standard support, and tools that work between multiple operating systems simplify and reduce the complexity of displaying graphics and running applications. This gives all of us a choice for each task that faces us: Whether to use a point-and-click GUI application or a command-driven CLI. Graphics make the job intuitive and easy to visualize; command lines make scripting and repetition easy for many of the tasks server managers are being asked to automate.

The arguments about which is better and what each style of computing should be for will rage on in discussion groups and whereever a few good computer people compare notes. One thing is clear: All of these styles of computing are here to stay, and interoperability between multiple operating systems will be the hallmark of all future workstations and servers.

Chapter 6 — Using E-mail with OpenVMS and Linux

Back in the Old Days

Way back in the early 1980s computer users could send e-mail to each other through electronic BBSs with ASCII graphics and unencoded binary objects across 1,200-baud modems. And (with apologies to SNL's Old Man) we liked it that way—no matter how painful it was or how long it took, and we were grateful for the technology!

By the time I finally switched from the ASCII communications I had used in my younger days to the e-mail client/server graphic's revolution e-mail of PCs and workstations, I could send embedded graphics, attachments, and binary transfer music and even video through e-mail. I also found (while transferring my ASCII e-mail to TK50 tape) that it took me 10 years of e-mail correspondence and daily use to have created a scant 1MB of data (using English as a communication protocol).

As I filled and filled the two-thirds loaded floppy disk (after changing my mind about the 95MB TK50) with 10 years worth of e-mail history, I received my first real attachment from someone at DEC via the new client/server graphics interface to my shiny new DEC 320p laptop computer.

I marveled that the 12 PowerPoint slides that had been attached to the message (the text of which was written in MS Word) were 1.5MB of data—they wouldn't even fit on the $10 1.44MB floppy disk I had just filled with the last ten years of history.

How had I ever managed without the ability to send graphics to everyone I knew? How much more disk space would I need in the brave new world? The horizons seemed endless to me, but I knew that I was glad that my company was bringing out something called StorageWorks in the near future.

We Used to Copy E-mail Across 2,400-Baud Modems?

Even with my corporate e-mail connection to the Net, by the mid-1990s I had been using UUCP at home to send and receive Internet e-mail for over five years. UUCP would dial up between my OpenVMS server and UNIX to copy batches of e-mail to and from Linux, UNIX, and other BBS servers until the messages found themselves on a machine they were destined for or were connected to the real Internet! While real Internet gateways were expensive, UUCP systems were free. The catch was that UUCP nodes might have 20 or more hops to find a gateway to the Internet. My worst time sending mail was a little over eight hours from home to my DEC corporate mailbox; my best time was little over 15 minutes before I switched over to a permanent, instantaneous e-mail connection in 1997.

Today, with e-mail accounts free on the Net and dial-up access costing less then $10 per month for people with a PC, most folks consider this type of e-mail quaint and old-fashioned. While the usefulness of UUCP mail has ebbed, it's interesting to note that the protocol still is used in ham radio (packet/system-to-system) e-mail and other places around the world where Internet dial-up (POTS—plain old telephone service) is still not easily obtained.

From the Dawn of the Network Era to Today

In *The Victorian Internet* Tom Standage describes how the nineteenth-century world was shrunken and delivered by a technological communications marvel: the telegraph. Tom's book quotes Samuel Morse's 1872 poem, "The Victory":

> And science proclaimed, from shore to shore
>
> That time and space ruled man no more.

If in the last 150 years the telegraph, telephone, and satellites shrank the distance across which people could communicate, then the Internet and e-mail in the last 15 years have dwarfed those tools by another factor of 10—simply with compression and near universal access. Why make a phone call when a two-line e-mail explains what needs to be done? Why have conversations when a single picture will do? Why send a picture when a URL points to the common reality of the Web for reference. Does anyone send multiple copies of any paper to anyone anymore unless by fax?

This is great for productivity, but what about that human-to-human contact that has kept us learning and improving our depth of understanding with mentorship and apprenticeship? The jury is still out on how electronic communications will impact how we learn and do that OJT. Our generation has home offices, digital cell phones, telecommuting, the World Wide Web, and cable, so we'll either be the generation most attached, the most plugged-in to our fellow man, or we'll live inside our Xboxes—only venturing out to earn enough to pay for the electricity, protein drinks, and vitamins to keep us going.

Terse, focused, and efficient, for good or bad, these are the qualities e-mail has delivered to us as a business tool.

Of course, if you still want to have a conversation online, you can now use a variety of instant messenger products and gossip your time away, just like the telegraph operators did! Telegraphers were found to spend as much as 50 percent of their free time talking, chatting, and gossiping with other operators across the nineteenth-century ether (but for those who live online today, 50 percent sounds like a very low percentage). Though the technology has changed dramatically, people have changed very little in the last 150 years. Many still have that quite human desire to simply put

down their labors, lean across a back yard fence or gather around an office watering hole, and just talk with other people about the events of the day.

E-mail Programs Come and E-mail Programs Go

In 1987, when I joined Digital, its mail ALL-IN-1 had about eight million users worldwide. ALL-IN-1 had full-text word processing and shared calendar (for resources such as conference rooms, demo rooms, etc.) that would also make people's time easier to schedule and keep track of. File exchanges for all the word-processing documents of the age, attachments, e-mail lists, and gateways to various other mail systems (IBM and the standards of the day).

Alas, ALL-IN-1 ran on OpenVMS clusters (who would want a redundant e-mail system anyway) and its client/server frontend teamlinks are still around. Despite all its technological prowess and PC integration, however, Digital chose a new mail system in 1996. Abandoning its ALL-IN-1 roots, DEC chose a hearty mail system—a mail system that could send word processing documents, pictures, video, and other binaries in e-mail, just like every other 20-year-old e-mail system.

The new e-mail system had the added advantage of being able to run on a Windows NT operating system, which, in 1996, everyone knew was the only system platform that would survive after the Y2K.

Microsoft Exchange was a lot more expensive than using Open Source, standards-based e-mail, but then again, Microsoft had much better marketing. The rest was history until people began to realize that Exchange costs lots of money to install and maintain and Open Source products tend to be free!

Open Source Open Standards Internet Mail

Open standards all revolve around TCP/IP standards and the Request for Comments (RFC) process that controls them. The important thing to note is that current e-mail protocols are evolving and continue to add functionality as they improve. An end user won't be worried about these standards, but someone who is doing the implementation for an operating system such as VMS or Linux will. What end users will need to know is what standards-based e-mail clients they can use from their various client platforms and what e-mail servers they can set up to store/retrieve their mail from OpenVMS and Linux servers on their network.

Internet client e-mail applications all do basically the same things, but some have more features then others. What features should you look for in a client e-mail system? Character cell or graphic e-mail applications should be the first consideration. Programs such as PINE are character-cell based, but can send attachments, binaries,

and run on both OpenVMS and Linux as clients. Other programs, such as Mozilla (now Open Source from its Netscape roots), also run on both OpenVMS and Linux, but require an X Windows graphic display to work. Graphical displays give you a What You See Is What You Get (WYSIWYG) e-mail window, where the character cell clients allow you to telnet in from anywhere in your network and read your mail—at least the text portions of it!

Today's important standards to check for when you configure your e-mail server include the following;

- *POP3—Post Office Protocol V3*: If your mail server supports this protocol, your remote mail clients can be configured to download all your mail/messages from the server to your client system. POP3 deletes the mail from the message/e-mail server once the messages have been downloaded to the client.

- *IMAP—Internet Message Access Protocol*: IMAP stores messages on the e-mail server for use by users from many different locations from many different clients! You can use other users' systems as e-mail clients and never move or delete read or unread messages from the e-mail/message server! This means you can read your mail from anyone's system you have access to, and when you log off, the messages will be secured and remain stored on the server system.

When do you use each protocol? Some users like their servers just to pass messages to them and erase them from the server—for them POP3 is the answer. Other users like to use everyone else's e-mail client applications and prefer to store their messages on their e-mail server for archiving and later retrieval and review—IMAP is best for them. The only real warning I can offer you for using these e-mail protocols is that for each e-mail user account you create, use only POP3 or IMAP as your message/e-mail access method for each mail account you have on a server. Your client could become a little confused (especially if you don't create multiple e-mail profiles on the client) if you use both POP3 and IMAP to access a single account! (Multiple inboxes aren't the best way to begin a new morning of e-mail excellence.)

The good news is that both OpenVMS and Linux are able to let e-mail clients have POP3 or IMAP access to their messages in the standard user accounts. All you have to do is activate e-mail services on the systems, and the user accounts are integrated into e-mail. Remote e-mail clients can then be used to access these standard user accounts via POP3 or IMAP to access e-mail on both OpenVMS and Linux.

E-mail Server Configuration for OpenVMS

Let's set OpenVMS up as an Internet e-mail server (see Figures 6-1 through 6-36). We'll assume that TCP/IP is already set, as descussed in Chapter 3.

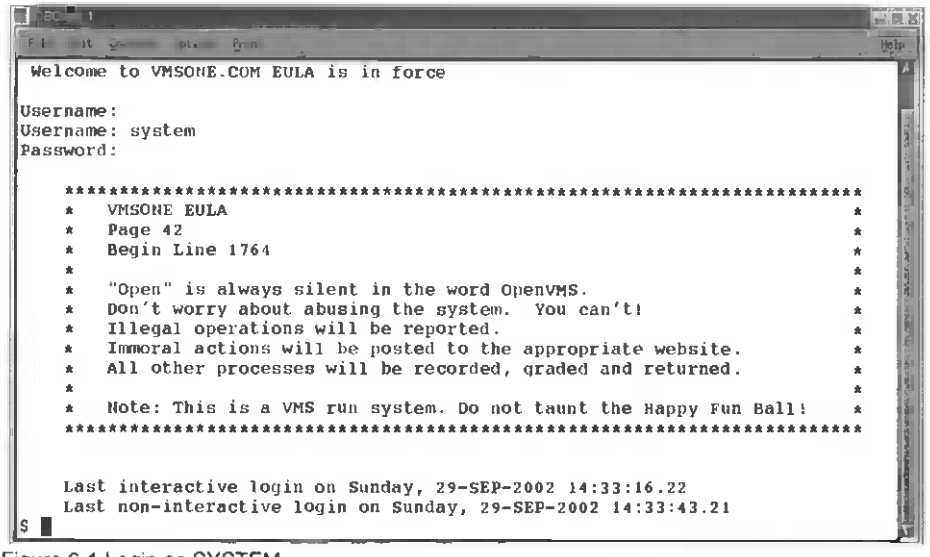


Figure 6-1 Login as SYSTEM.

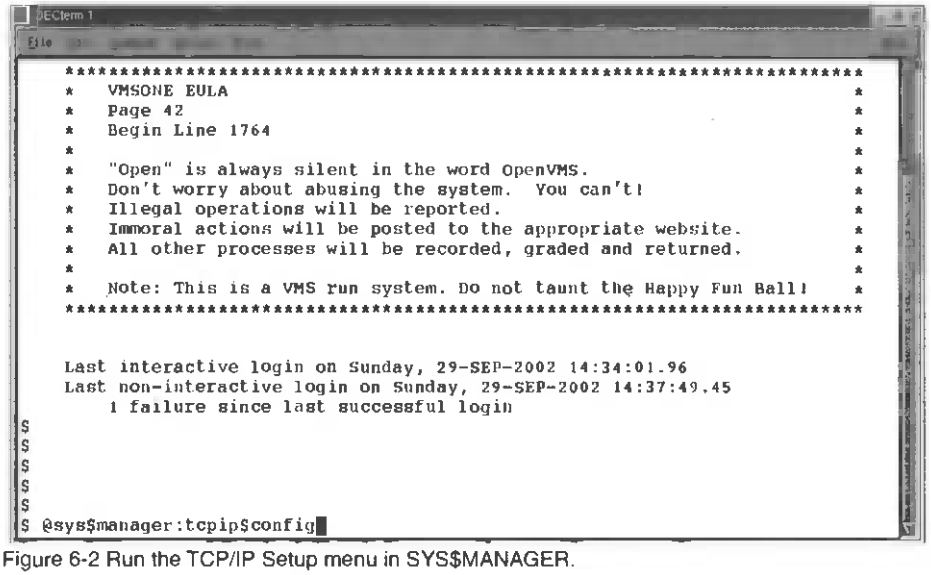


Figure 6-2 Run the TCP/IP Setup menu in SYS$MANAGER.

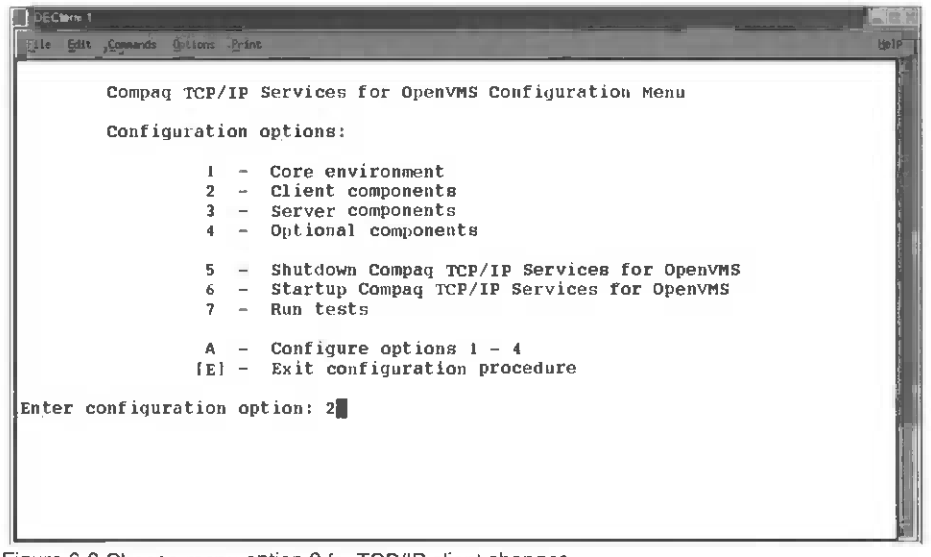


Figure 6-3 Choose menu option 2 for TCP/IP client changes.

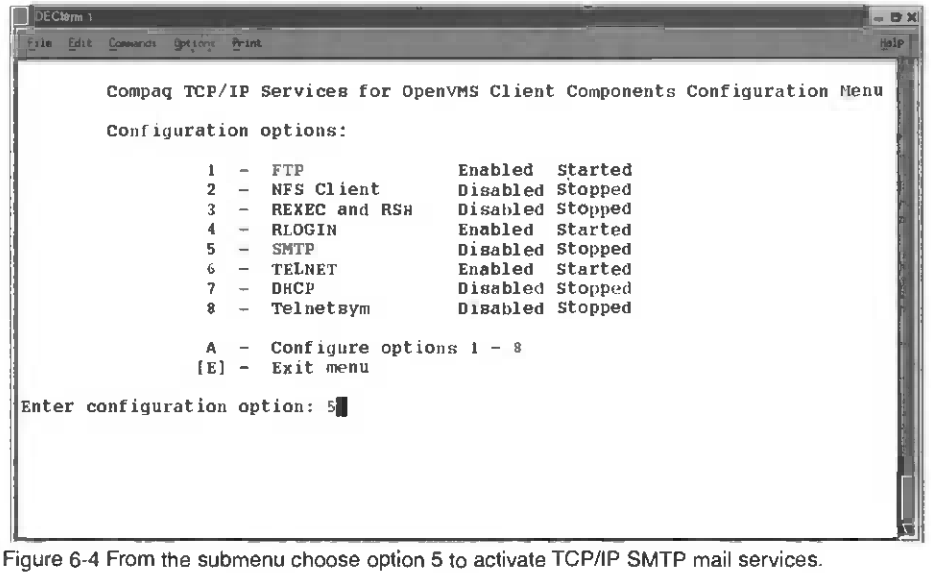


Figure 6-4 From the submenu choose option 5 to activate TCP/IP SMTP mail services.

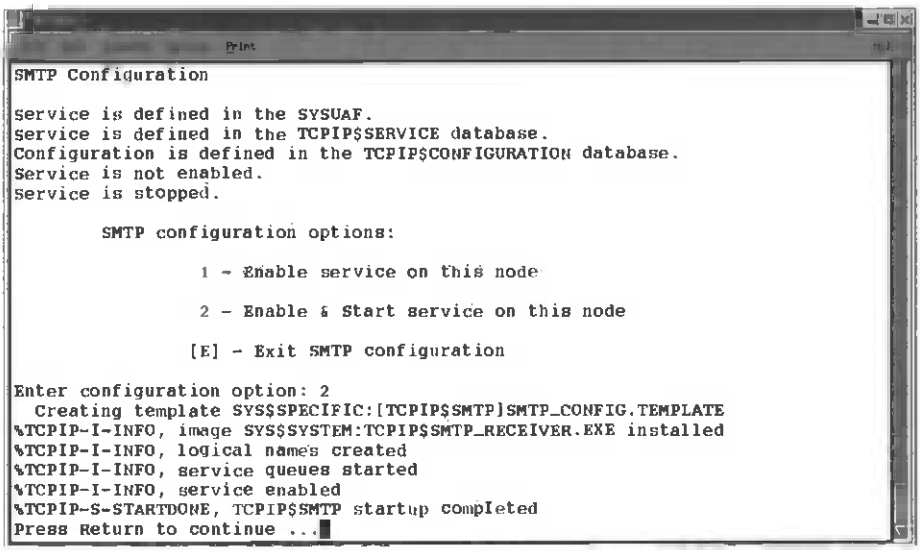


Figure 6-5 Menu choice 2 activates SMTP for this OpenVMS system.

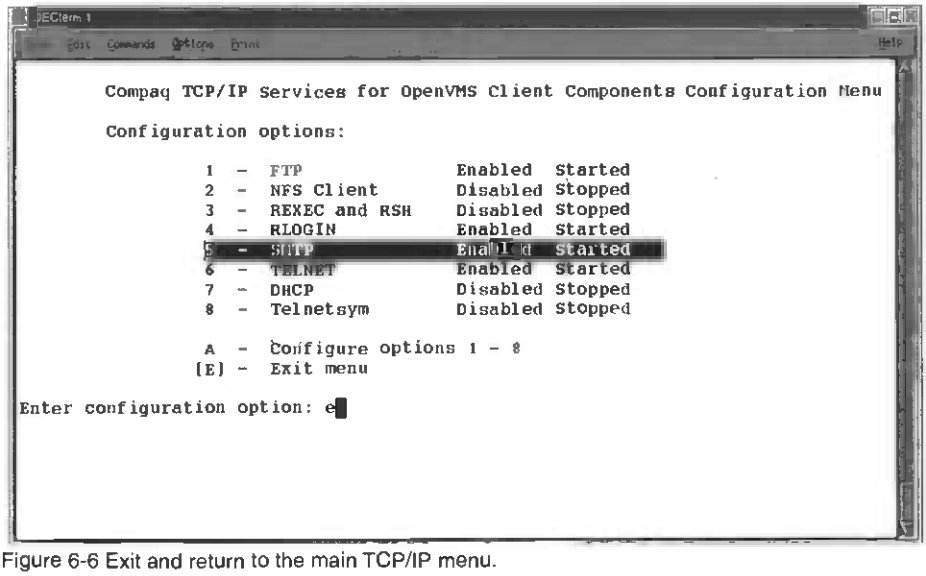


Figure 6-6 Exit and return to the main TCP/IP menu.

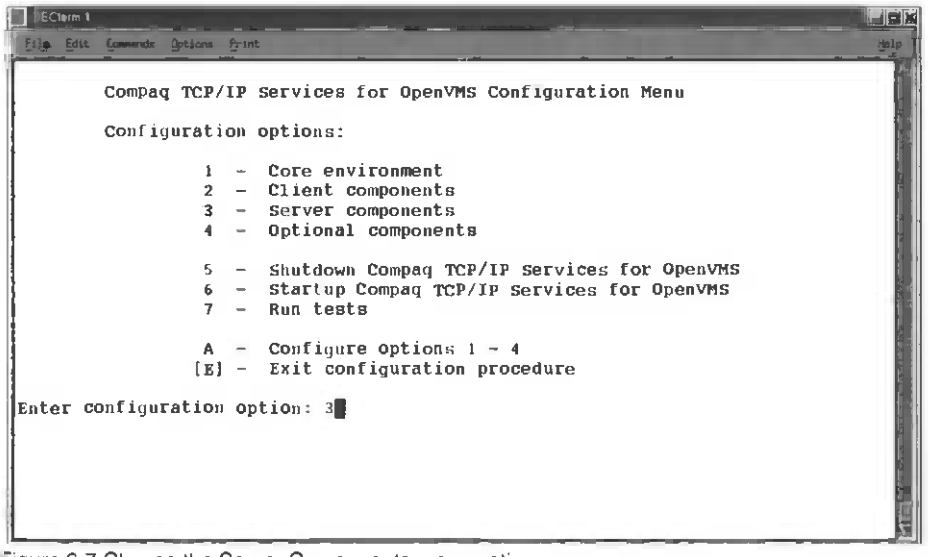


Figure 6-7 Choose the Server Components menu option.

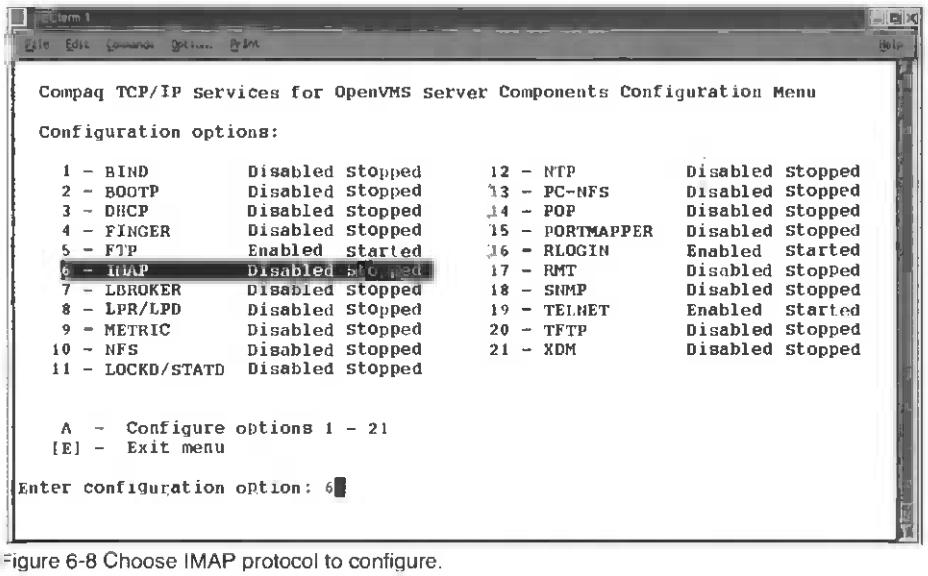


Figure 6-8 Choose IMAP protocol to configure.

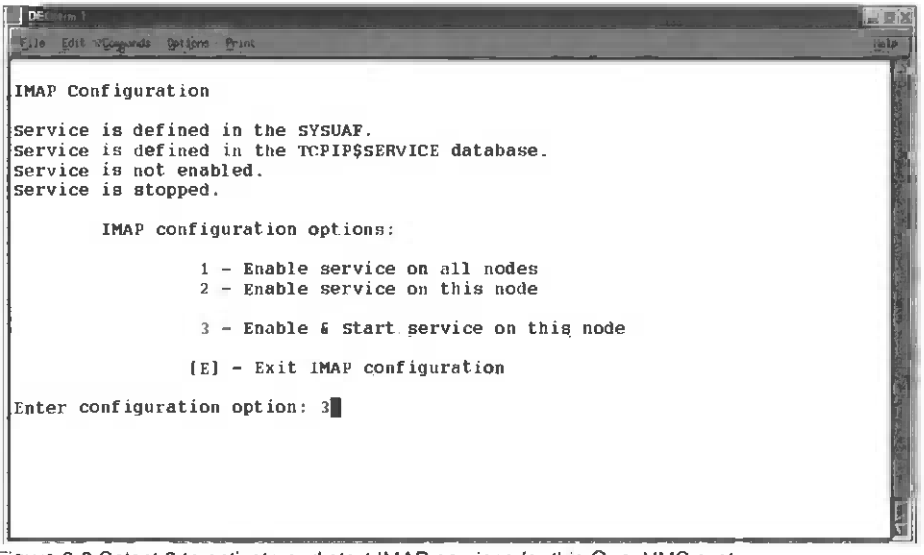


Figure 6-9 Select 3 to activate and start IMAP services for this OpenVMS system.

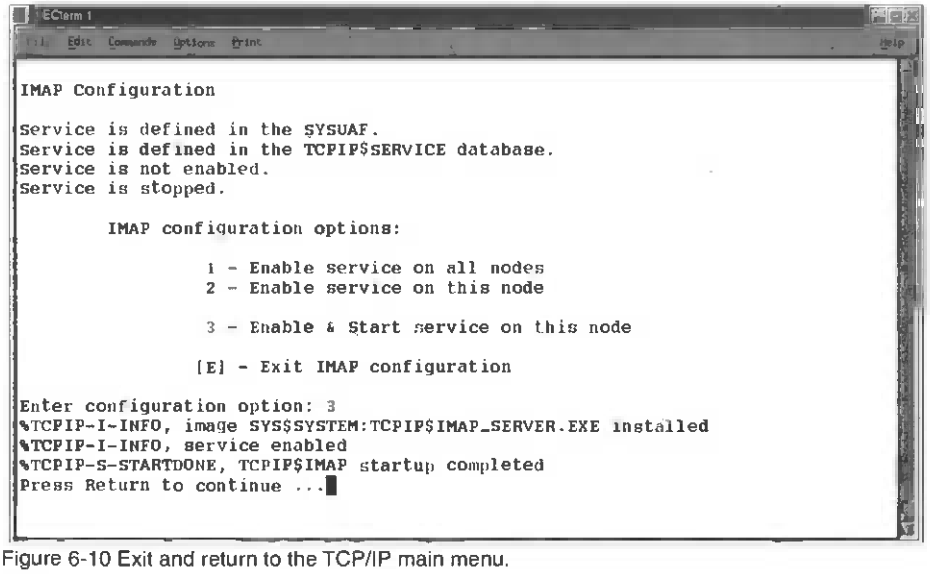


Figure 6-10 Exit and return to the TCP/IP main menu.

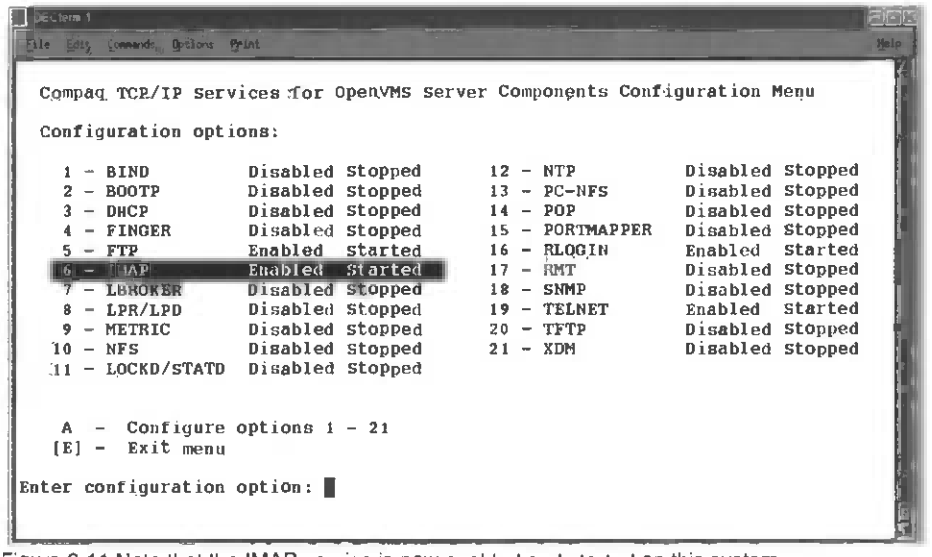


Figure 6-11 Note that the IMAP service is now enabled and started on this system.

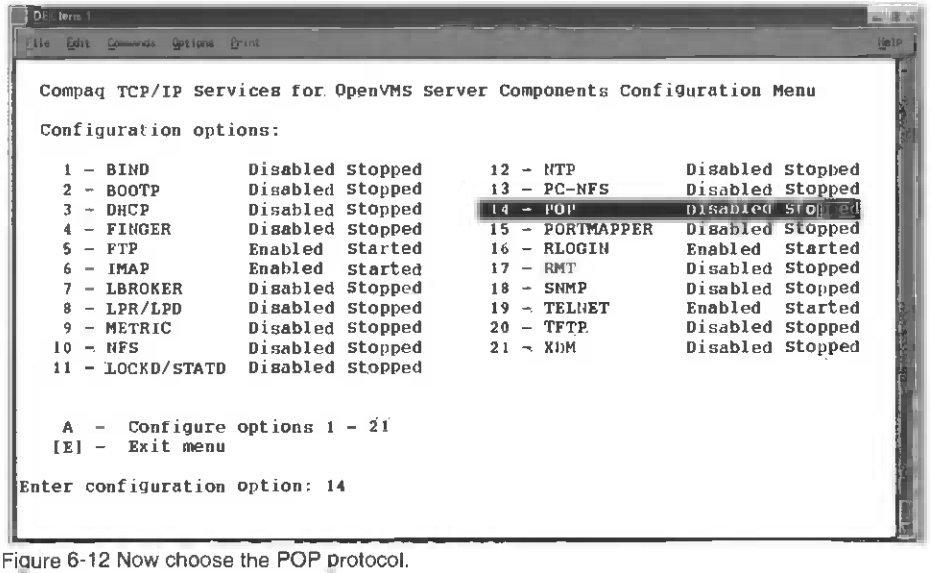


Figure 6-12 Now choose the POP protocol.

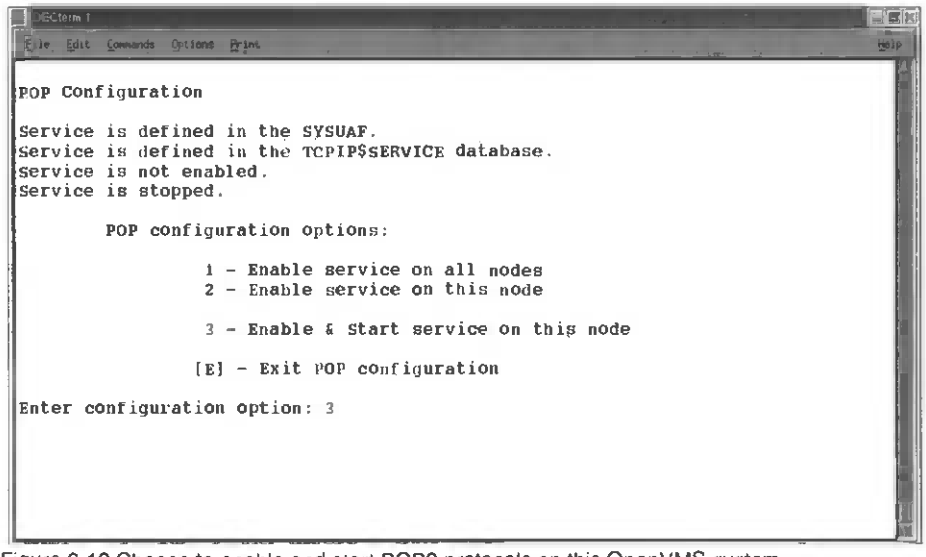


Figure 6-13 Choose to enable and start POP3 protocols on this OpenVMS system.

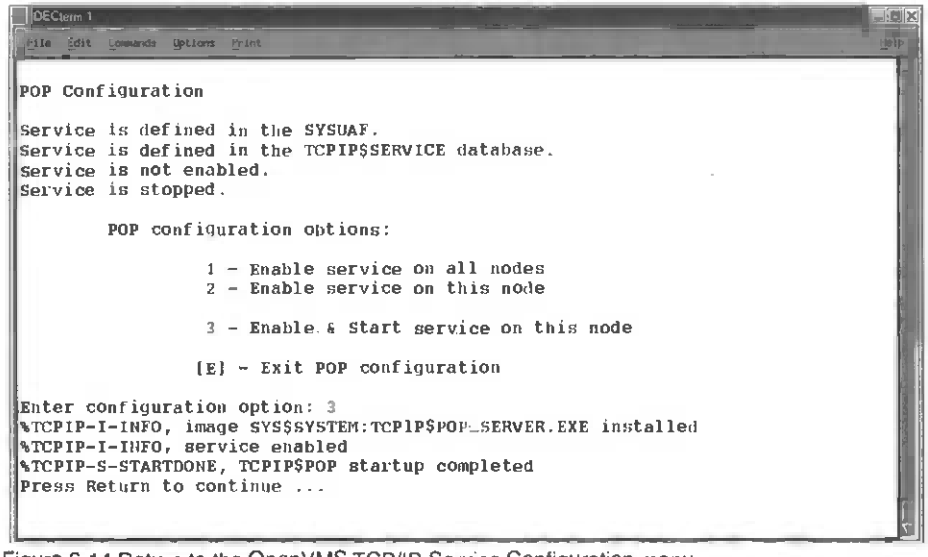


Figure 6-14 Return to the OpenVMS TCP/IP Service Configuration menu.

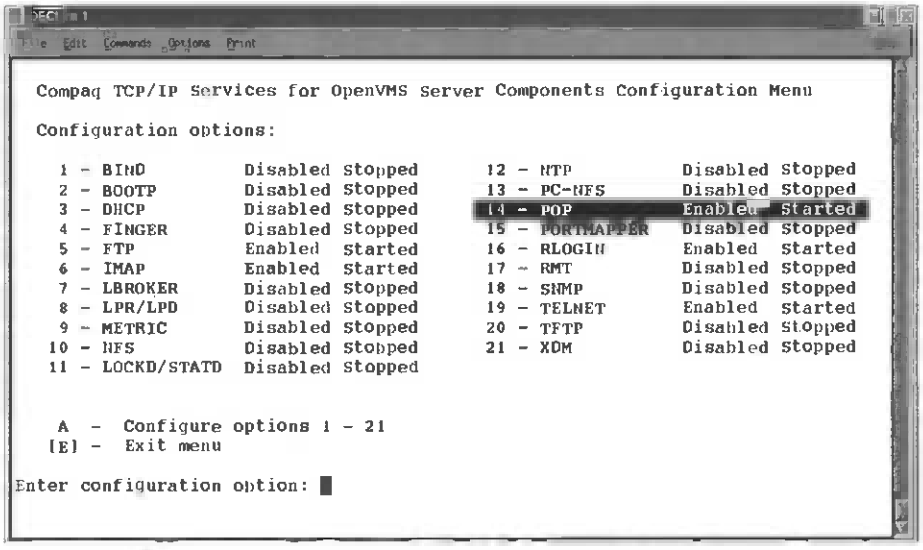


Figure 6-15 Verify that the POP3 protocol is enabled and running on this system.

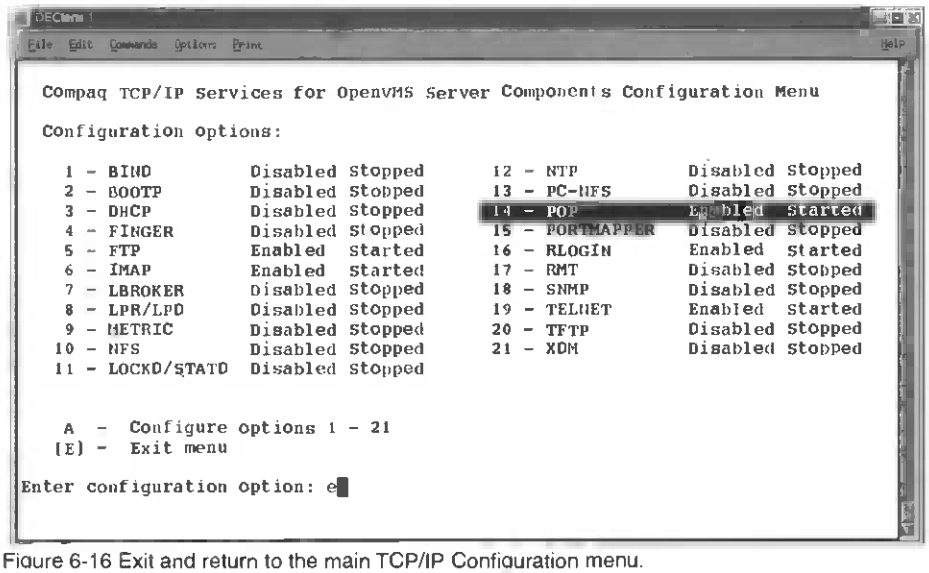


Figure 6-16 Exit and return to the main TCP/IP Configuration menu.

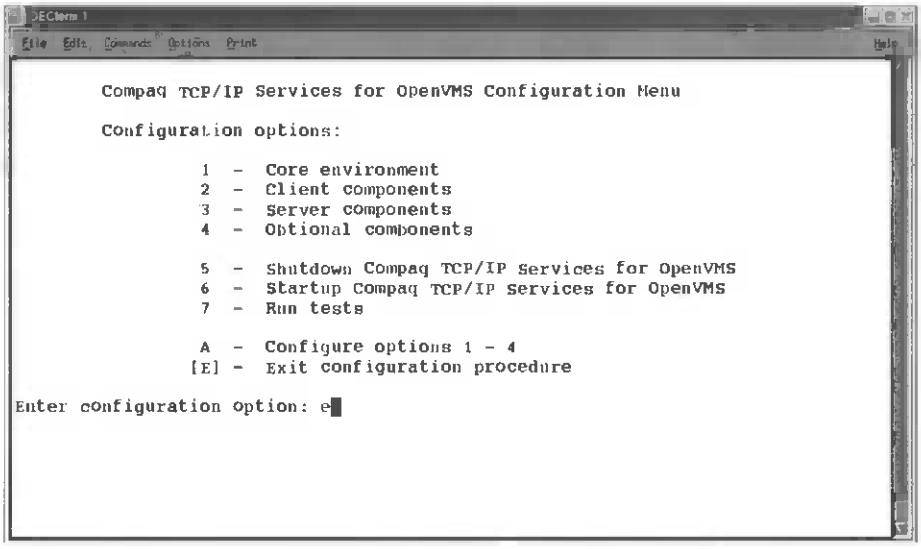


Figure 6-17 Exit the TCP/IP Configuration menu and return to a $ DCL prompt.

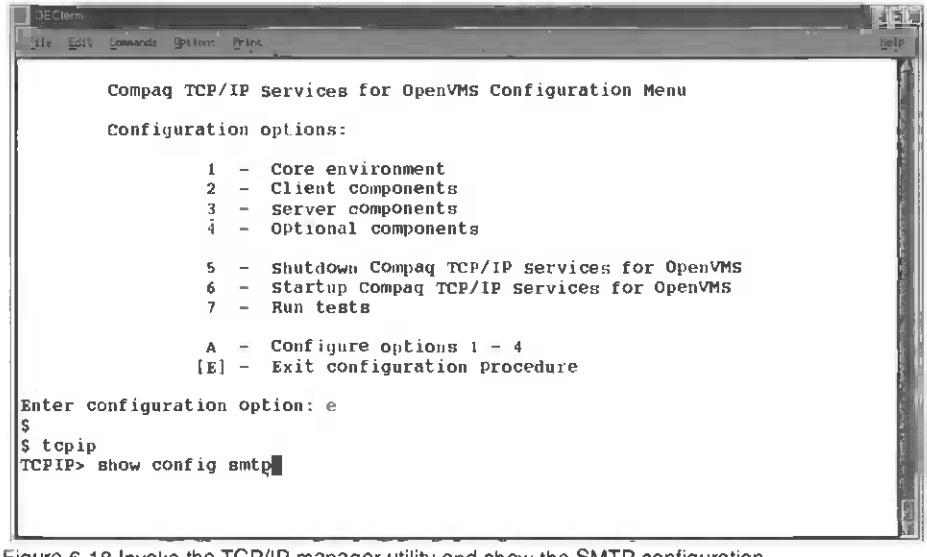


Figure 6-18 Invoke the TCP/IP manager utility and show the SMTP configuration.

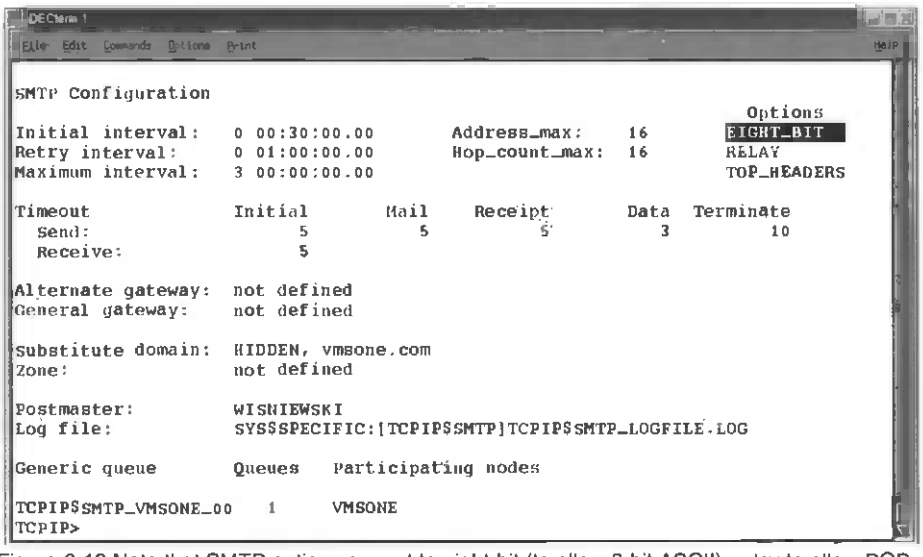


Figure 6-19 Note that SMTP options are set to eight-bit (to allow 8-bit ASCII), relay to allow POP and IMAP clients to send mail through the OpenVMS server, and top headers to display e-mail headers at the top of the mail messages.

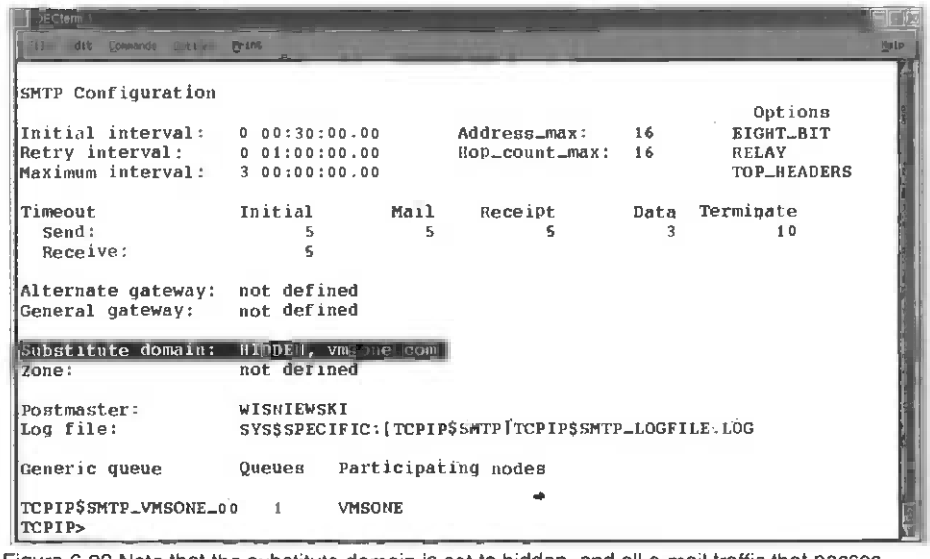


Figure 6-20 Note that the substitute domain is set to hidden, and all e-mail traffic that passes through this server will receive a return e-mail address of VMSONE.com.

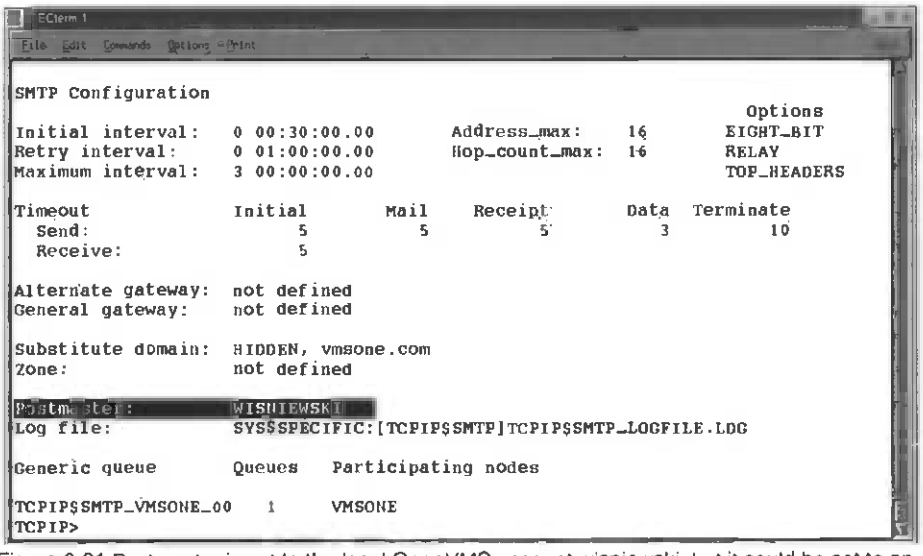


Figure 6-21 Postmaster is set to the local OpenVMS account wisniewski, but it could be set to an Internet e-mail address just as easily (e.g., wisniewski@VMSONE.com). The postmaster info for this system could even be sent to a remote system with this field.

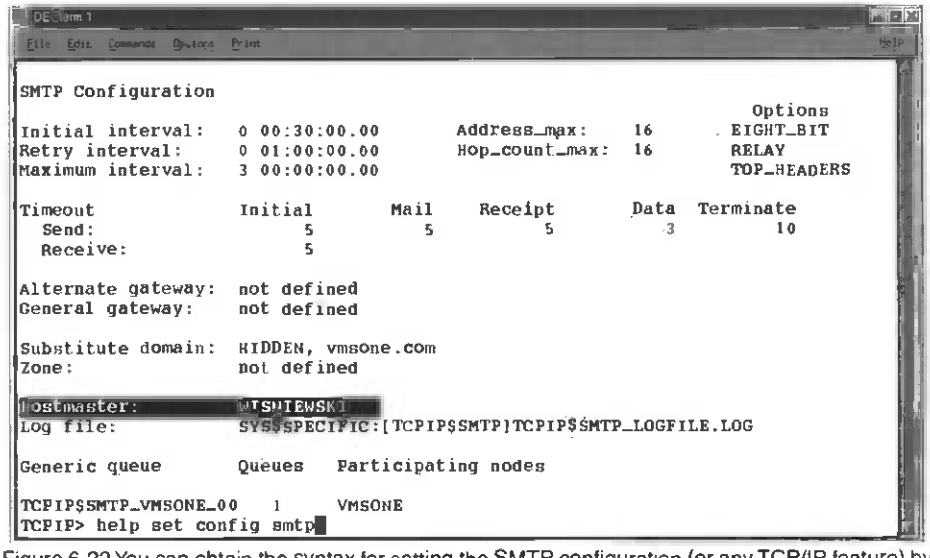


Figure 6-22 You can obtain the syntax for setting the SMTP configuration (or any TCP/IP feature) by typing help.

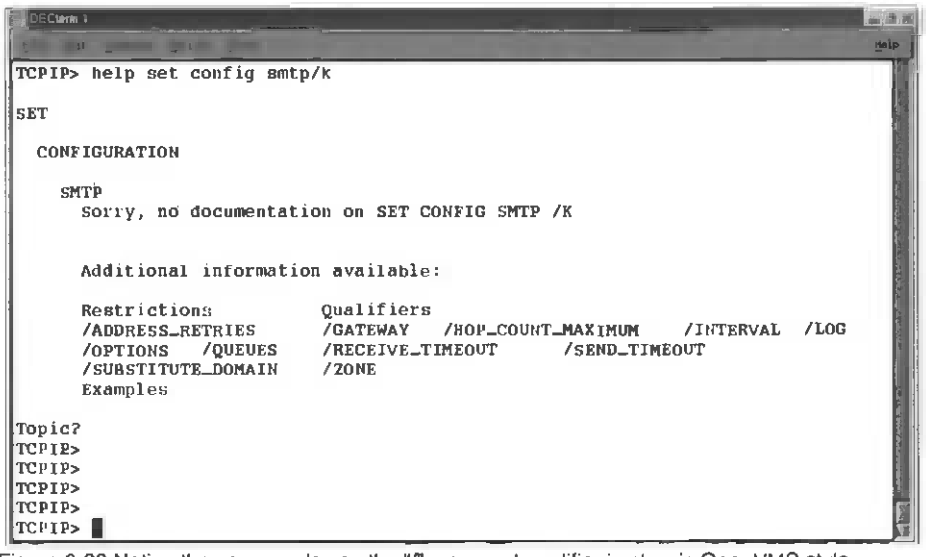


Figure 6-23 Notice the commands use the "/" argument qualifier in classic OpenVMS style.

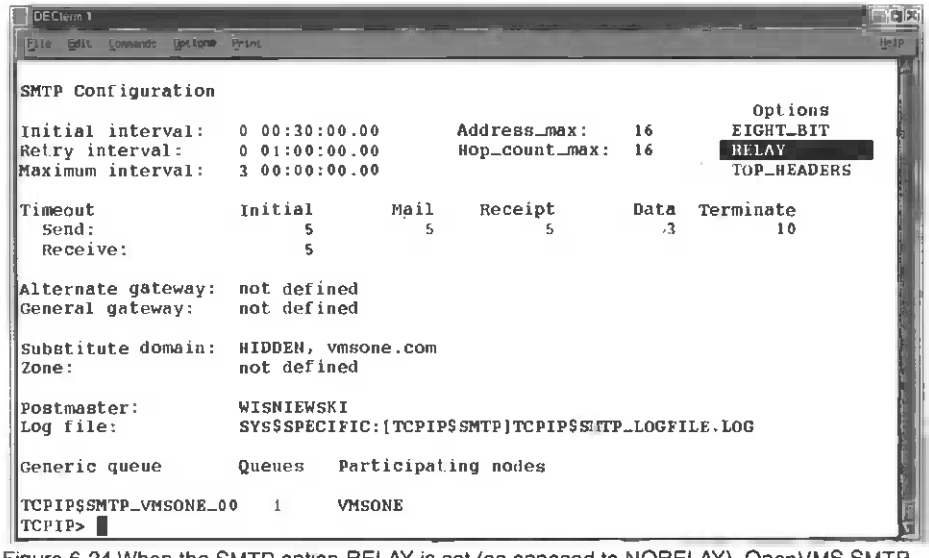


Figure 6-24 When the SMTP option RELAY is set (as opposed to NORELAY), OpenVMS SMTP needs to have its ANTI-SPAM features turned on. SPAM or junk e-mail can be sent though your system to many other systems/users without configuring SPAM filters for your OpenVMS server.

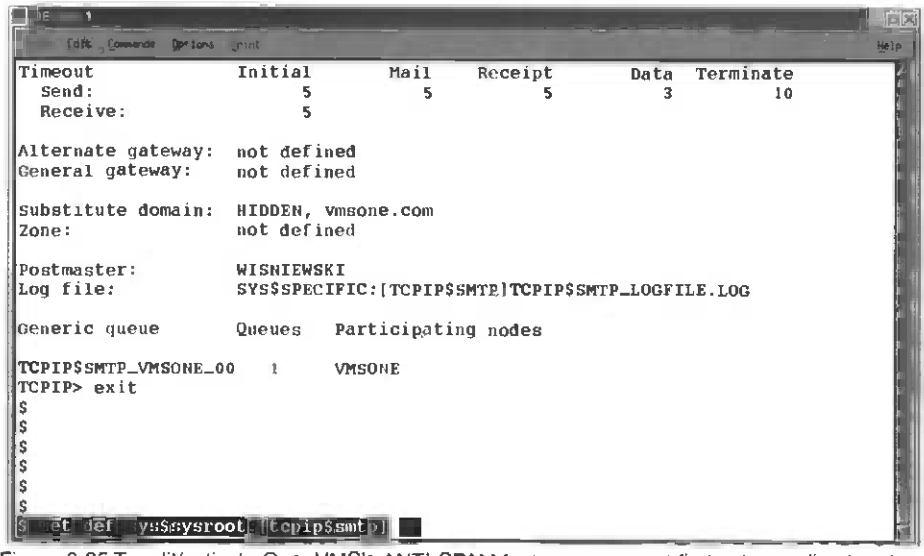


Figure 6-25 To edit/activate OpenVMS's ANTI-SPAM features you must first set your directory to the SMTP handling directory.

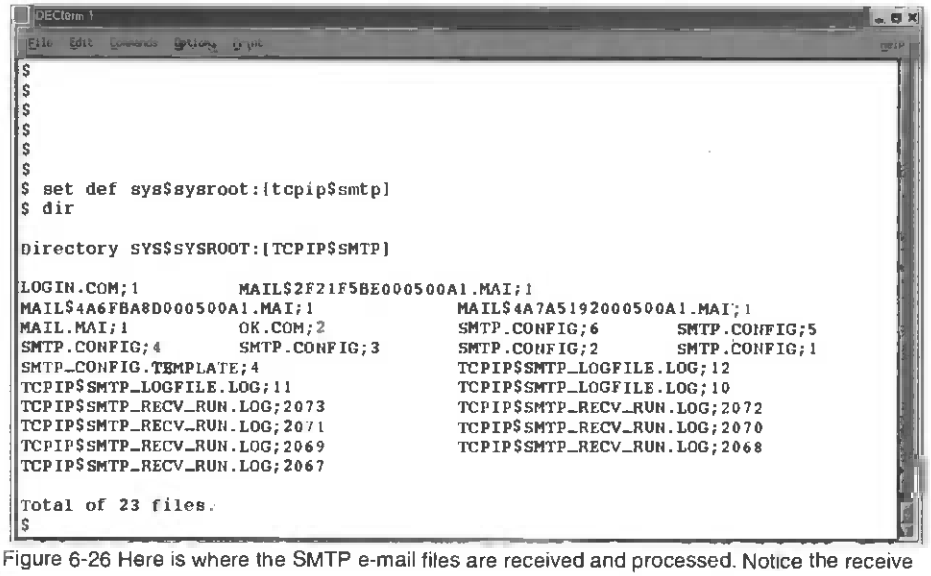


Figure 6-26 Here is where the SMTP e-mail files are received and processed. Notice the receive log files. These are where all SMTP e-mails come for relay or local handling.

Figure 6-27 First copy the configuration template to SMTP.CONFIG and begin to create the ANTI-SPAM rules file.

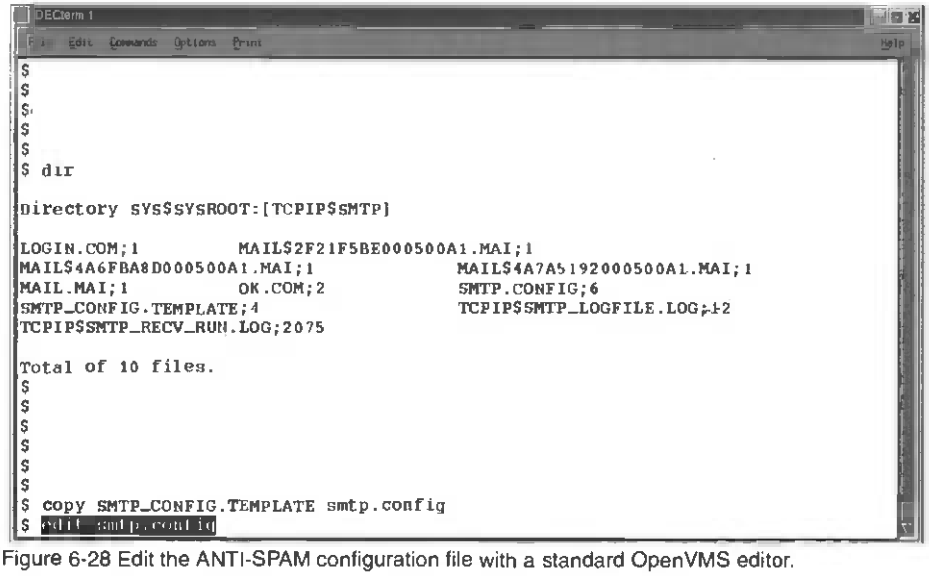


Figure 6-28 Edit the ANTI-SPAM configuration file with a standard OpenVMS editor.

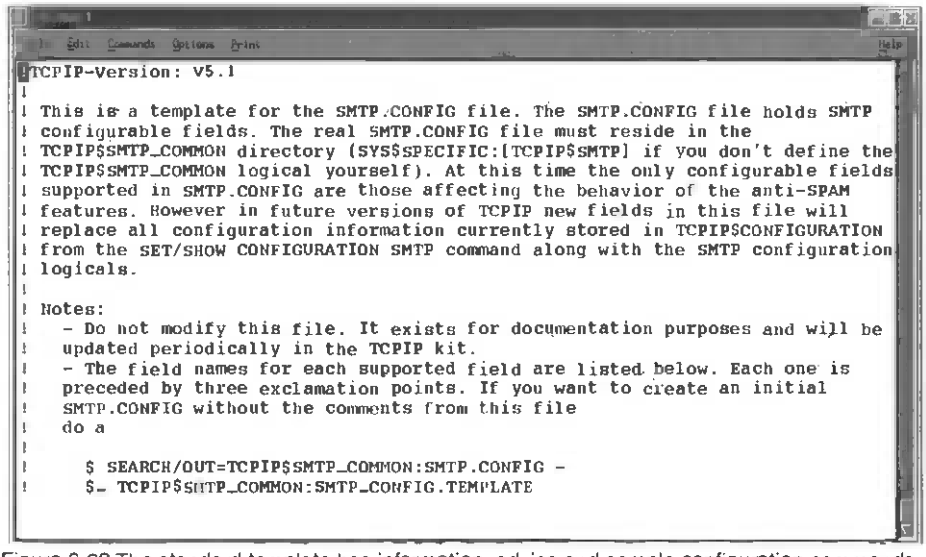


Figure 6-29 The standard template has information, advice and sample configuration commands, but none is active until you edit the configuration commands and remove the three !!! marks before the configuration directives.

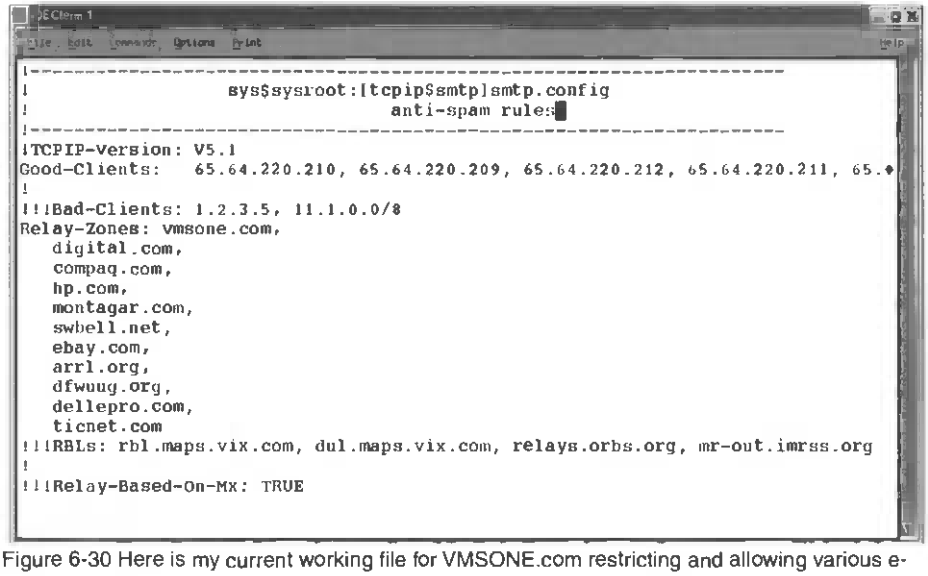


Figure 6-30 Here is my current working file for VMSONE.com restricting and allowing various e-mail systems to use or not use my machine (as an e-mail relay point).

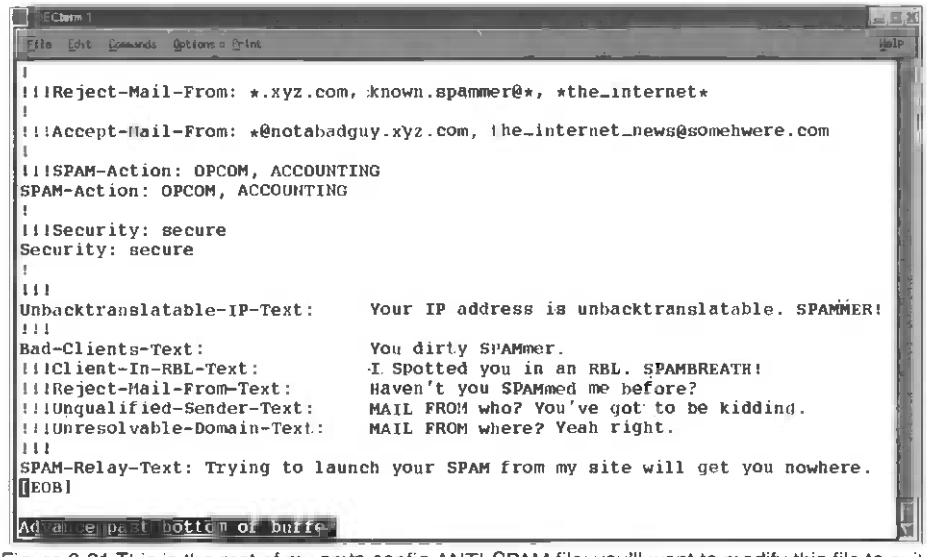


Figure 6-31 This is the rest of my smtp.config ANTI-SPAM file; you'll want to modify this file to suit your own e-mail network or be at risk of being used as a relay launching point or a target for e-mail spam.

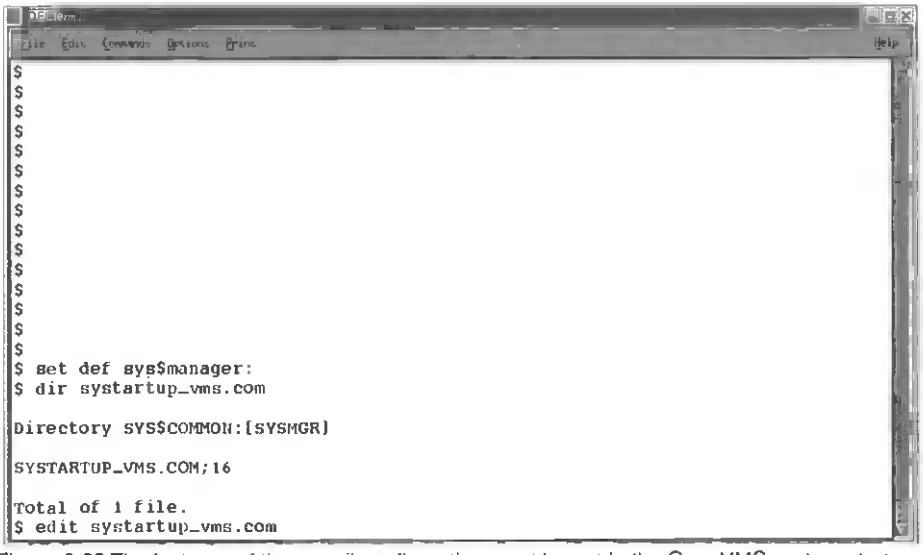


Figure 6-32 The last part of the e-mail configuration must be put in the OpenVMS system startup file, because it needs to be performed before SMTP starts up.

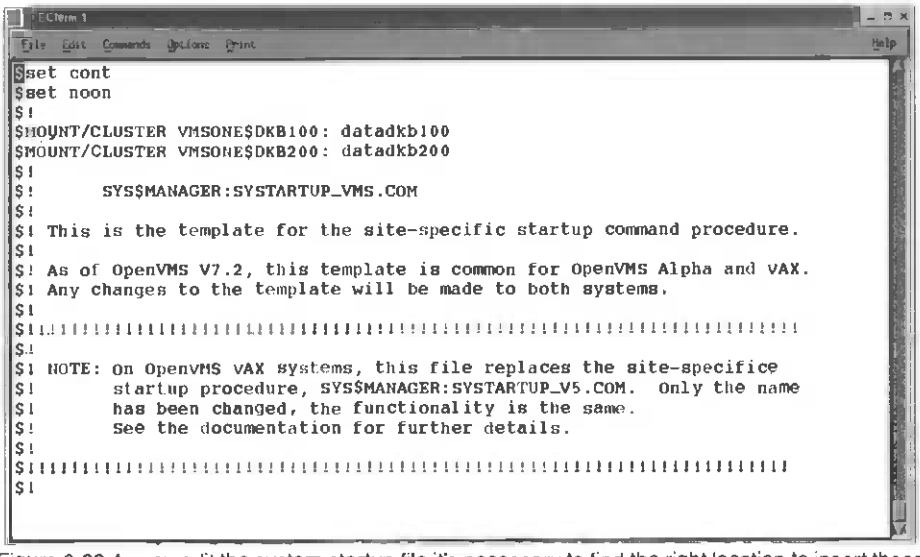


Figure 6-33 As you edit the system startup file it's necessary to find the right location to insert these configuration changes.

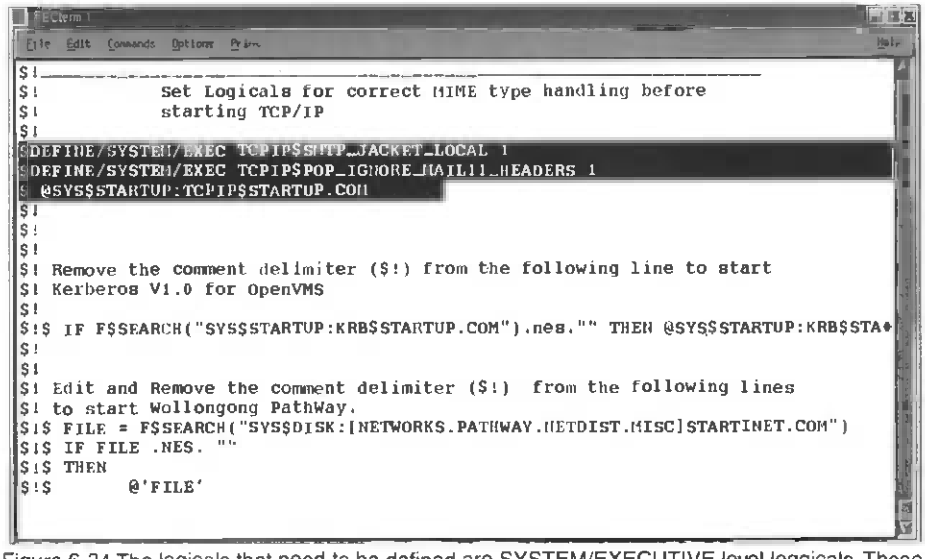


Figure 6-34 The logicals that need to be defined are SYSTEM/EXECUTIVE level loggicals. These logicals are then able to be read by all OpenVMS processes systemwide (similar to the global $PATH variable in Linux). When defined, these logicals tell the SMTP batch process how to correctly handle MIME extensions, e-mail headers, and attachements. Note that these logicals muste be defined before TCP/IP is started!

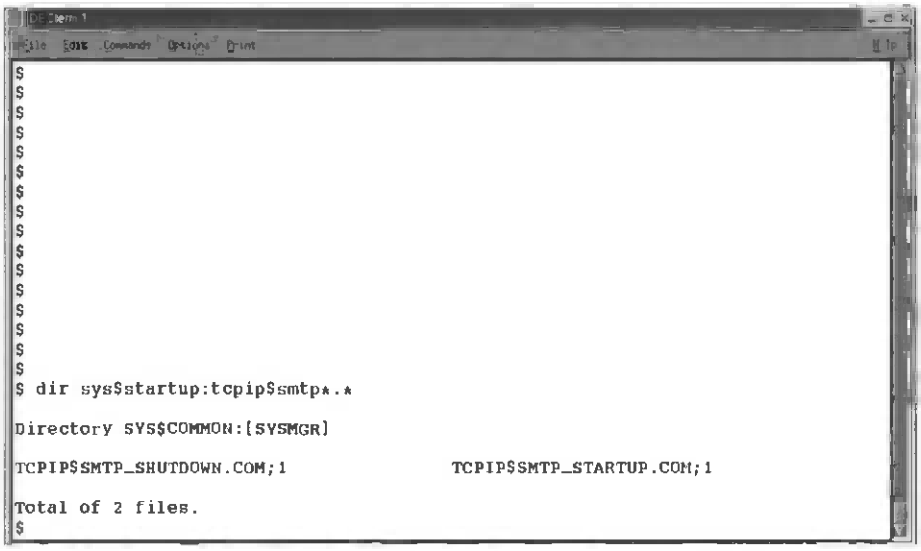


Figure 6-35 Now that we've configured the OpenVMS server for Internet e-mail and set our ANTI-SPAM defenses, we need to stop and restart tthe SMTP processes. The commonad files for both are to be fround in SYS$STARTUP: directory.

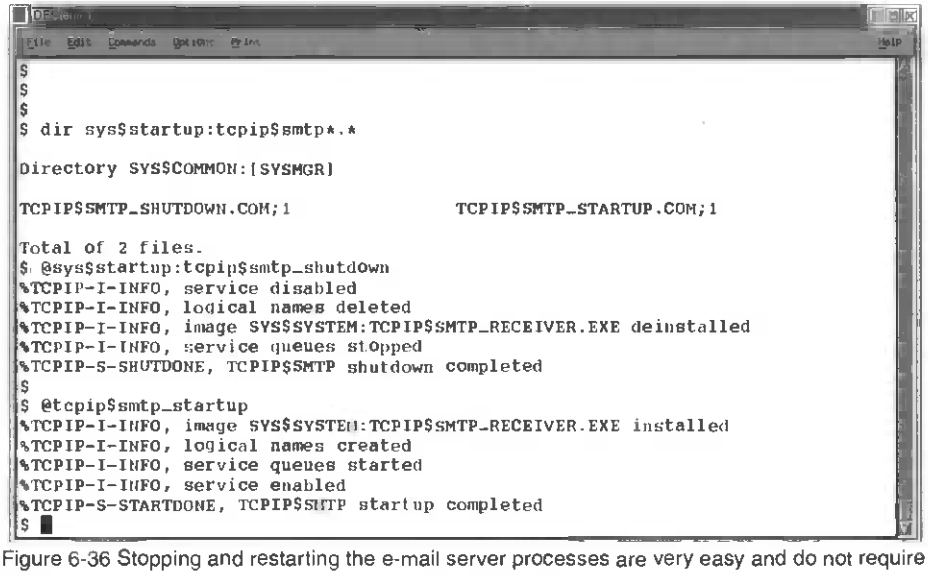


Figure 6-36 Stopping and restarting the e-mail server processes are very easy and do not require that you shut down all TCP/IP functions to accomplish this.

Congratulations! You've just set up an OpenVMS e-mail server with ANTI-SPAM filters!

Setting up E-mail Clients for Our E-mail Server

Both OpenVMS and Linux run Mozilla, now Open Source from its old Netscape roots. This means that for the price of a download, you can have not only a world-class Web browser but a full graphics and network-based e-mail client for your OpenVMS or Linux workstation.

Linux Mozilla Installation

Usually, when you first install RedHat 7.3, Mozilla just gets installed automatically. But what happens when you want to install an update or haven't had Mozilla installed as part of the base installation?

Install Mozilla by first downloading the Mozilla kit from www.mozilla.org (see Figures 6-37 through 6-49).

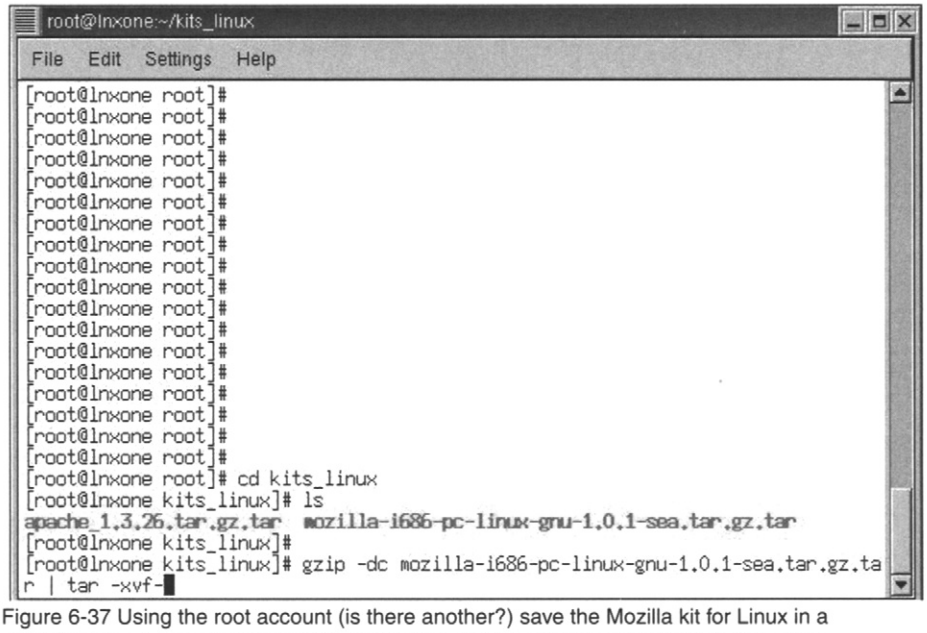


Figure 6-37 Using the root account (is there another?) save the Mozilla kit for Linux in a subdirectory, so you can expand it without filling your default directory with various files.

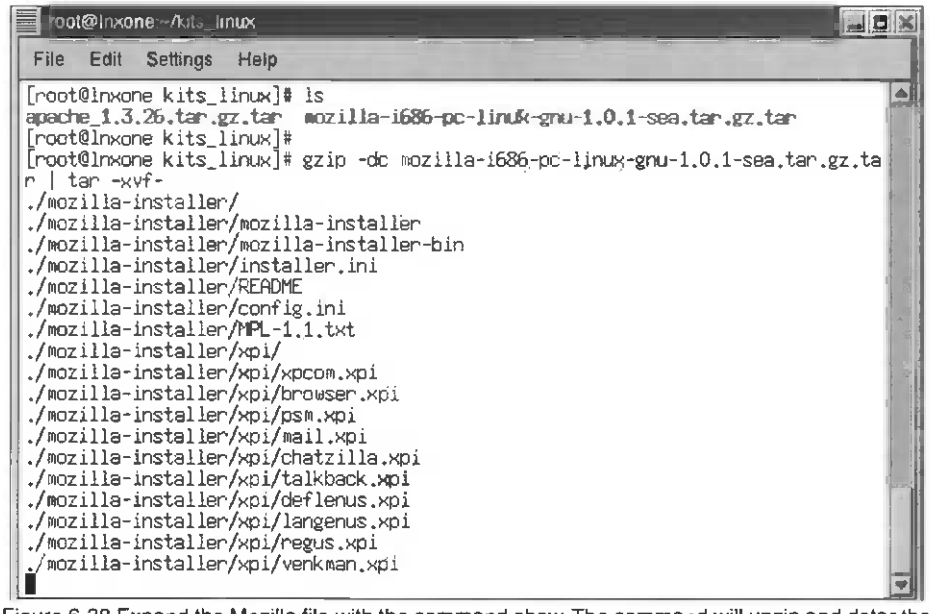


Figure 6-38 Expand the Mozilla file with the command show. The command will unzip and detar the tile into subdirectories.

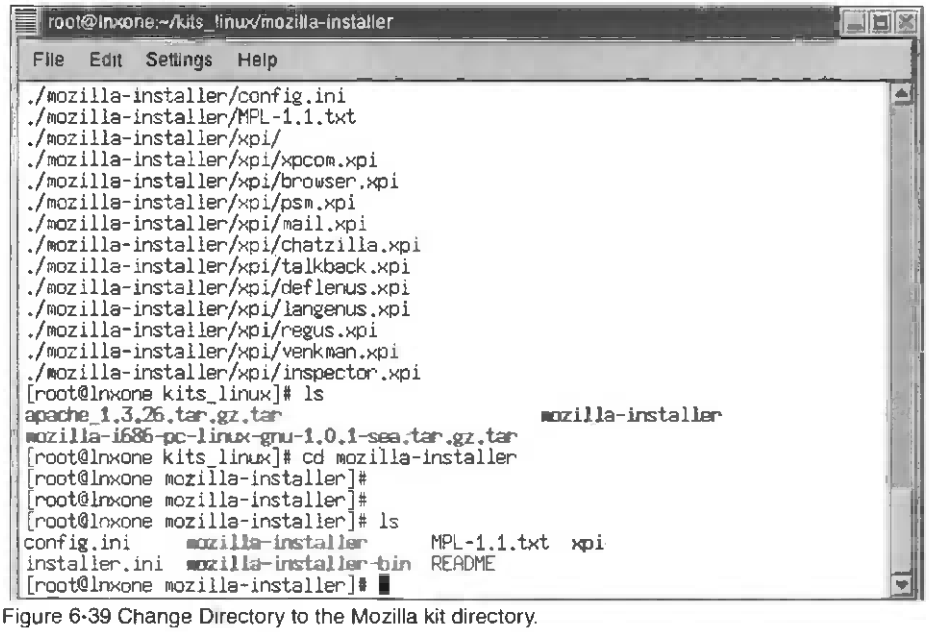


Figure 6-39 Change Directory to the Mozilla kit directory.

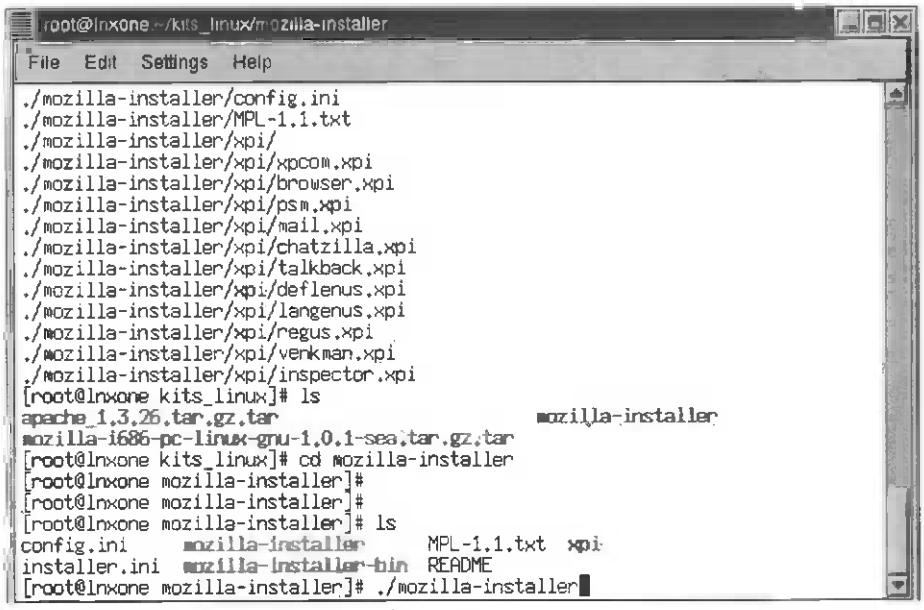


Figure 6-40 Run the ./mozilla-installer script.

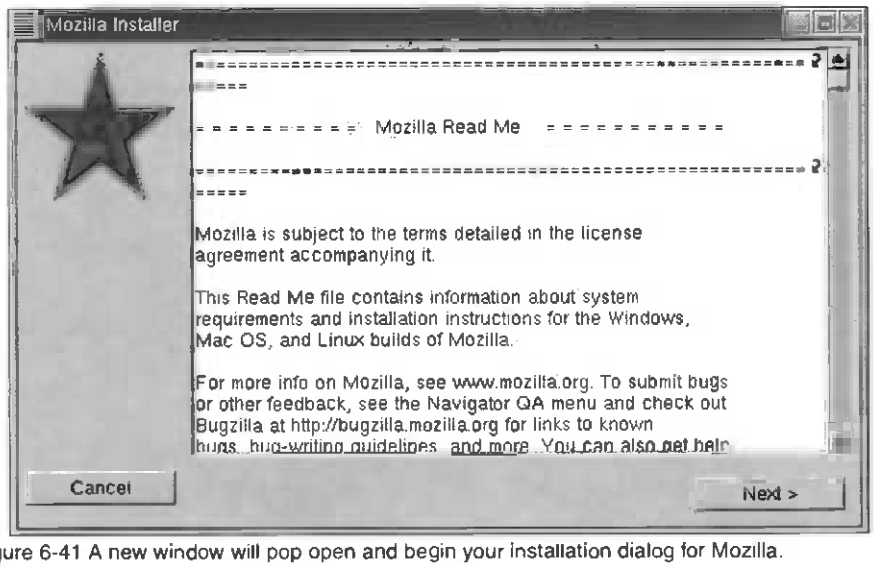


Figure 6-41 A new window will pop open and begin your installation dialog for Mozilla.

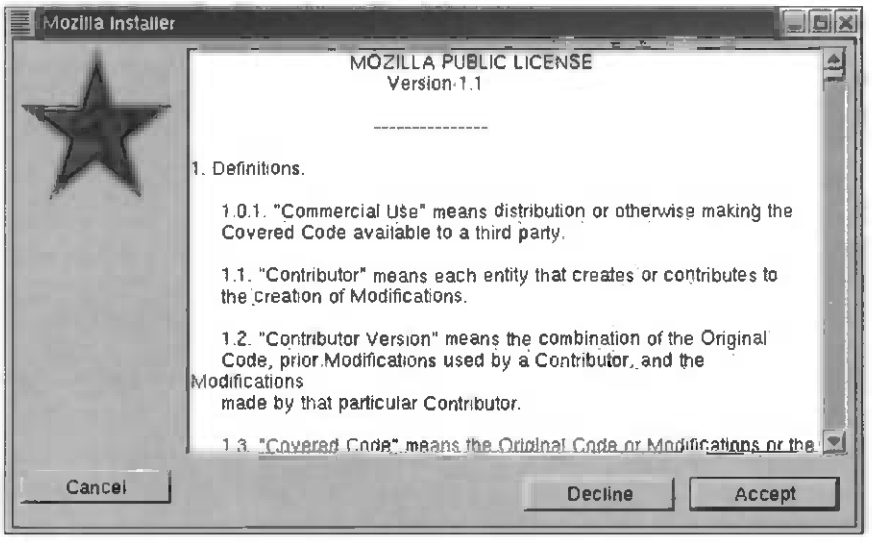


Figure 6-42 Accept the public licenses; Microsoft has made you sign worse.

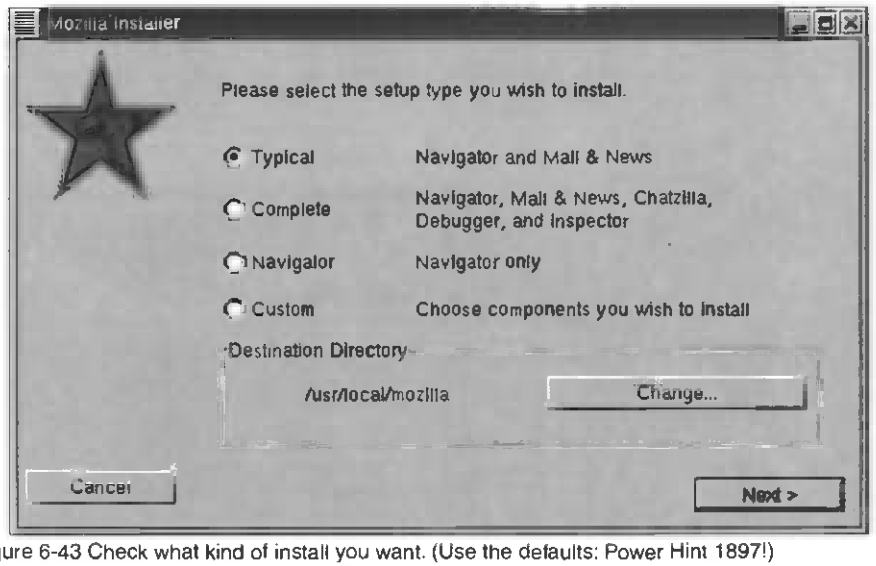


Figure 6-43 Check what kind of install you want. (Use the defaults: Power Hint 1897!)

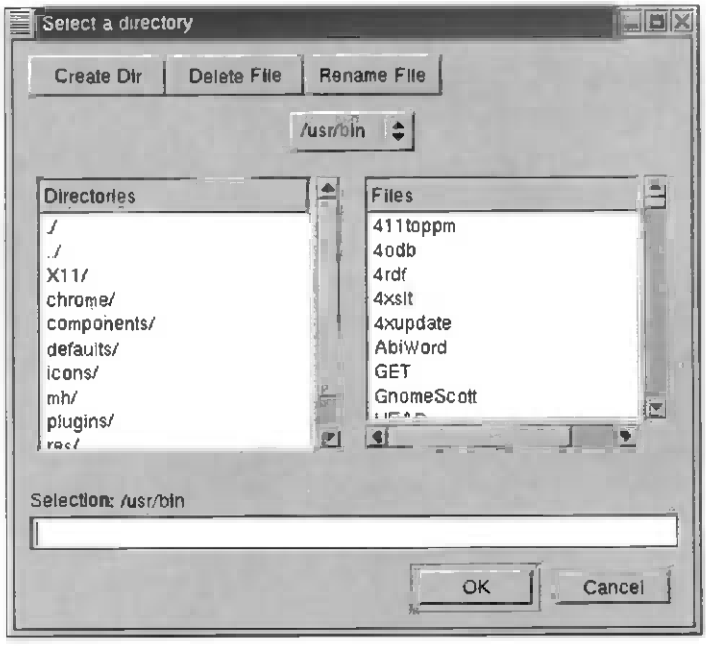


Figure 6-44 Select the location directory to install Mozilla.

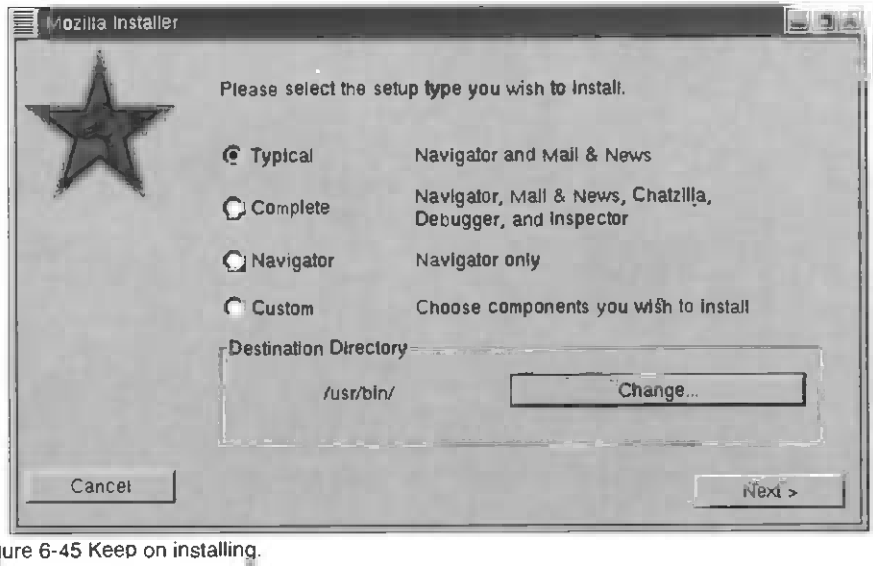


Figure 6-45 Keep on installing.

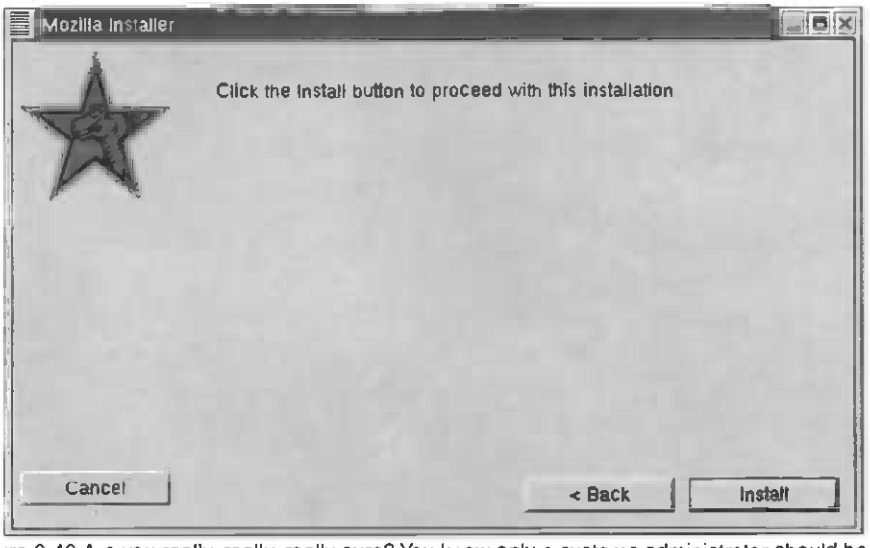


Figure 6-46 Are you really, really, really sure? You know only a systems administrator should be doing this, right?

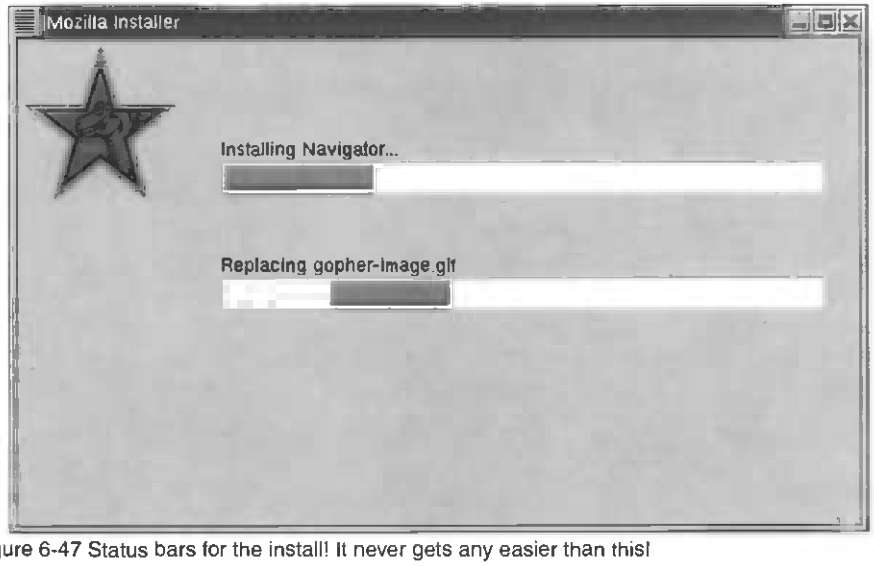


Figure 6-47 Status bars for the install! It never gets any easier than this!

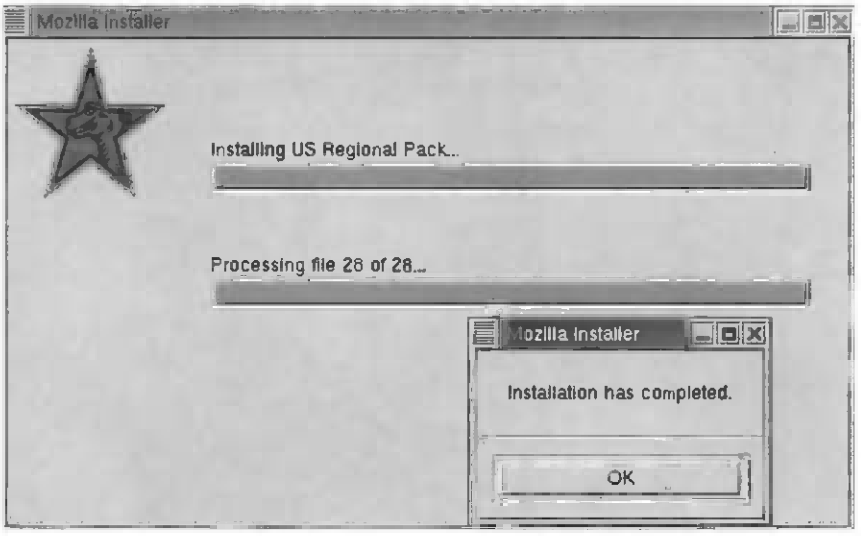


Figure 6-48 Linux Mozilla installation has completed!

OpenVMS Mozilla Installation

```
File  Edit  Commands  Options  Print                                    Help

Directory DKB0:[KITS.TEMP]

2RC92FGX.EXE;1       AUTOGEN_RUN.COM;1    AUTOGEN_SHADOW.COM;1
CPQ-AXPVMS-CSWB-V0100--1.PCSI;1
CPQ-AXPVMS-CSWS-T0103--1.PCSI-DCX-AXPEXE;1
CPQ-AXPVMS-CSWS-V0101--1.PCSI;1
CPQ-AXPVMS-CSWS-V0102--1.PCSI-DCX-AXPEXE;1
CPQ-AXPVMS-CSWS_PERL-V0101--1.PCSI-DCX-AXPEXE;1
CPQ-AXPVMS-CSWS_PHP-T0101--1.PCSI-DCX-AXPEXE;1
CPQ-AXPVMS-PERL-V0506-1-1.PCSI-DCX-AXPEXE;1
CPQ-AXPVMS-SSL-V0100-A-1.PCSI-DCX_AXPEXE;1
CSWB-OPENVMS-ALPHA-V100.SFX_AXPEXE;1     CSWS_MODPERL_RELNOTES.HTML;1
CSWS_PHP_RELNOTES.HTML;1                 DE7SH2NO.EXE;1       DECW_TERM.COM;1
GNUPG1_0_4_VMS.ZIP;1                     KHAZ4H6B.EXE;1
LIBNSPR4_110_ALPHA.SO;1                  LIBNSPR4_CSWB_10.SO;1
LIBNSPR4_CSWB_11B.SO;1                   LIBNSPR4_MOZILLA_100.SO;1
LOGIN.COM;20         LQZ4HY70.EXE;1      MAJORDOMO010_SRC.ZIP;1
O5YJGPQ7.EXE;1       PERL$STARTUP.COM;6  PERL_RELNOTES.TXT;1  SMBCLIENT.ZIP;1
SSL_IGUIDE.TXT;1     TCPDUMP_SMTP.COM;1  TCPDUMP_TELNET.COM;1
UNZIP.ALPHA_EXE;1    VIM.DIR;1           WHQVW1UR.PDF;1

Total of 35 files.
$ product install cswb
```

Figure 6-49 Log in as SYSTEM and put the kit in a directory separate from system files. You will find OpenVMS Mozilla at http://www.openvms.compaq.com (not at http://www.mozilla.org/).

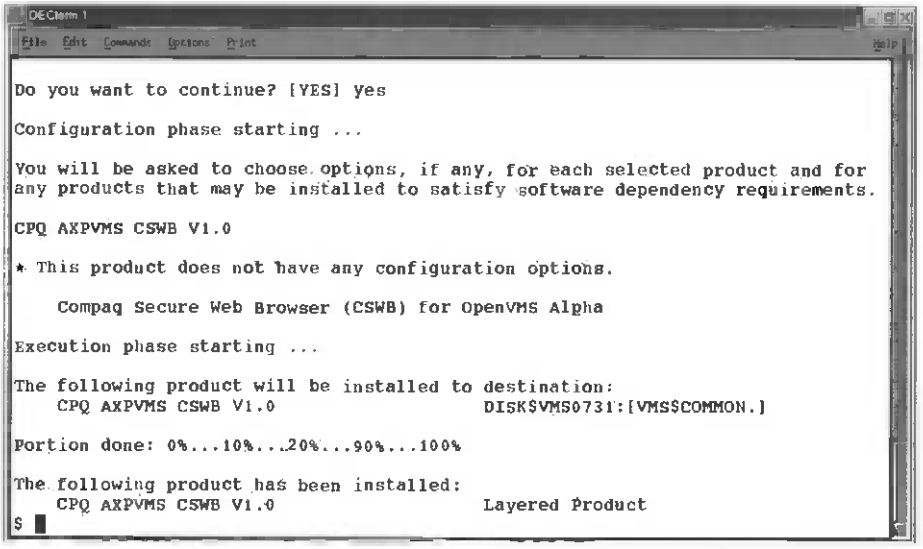


Figure 6-50 Use the standard OpenVMS installation kit (CSWB) and use the $PRODUCT INSTALL command.

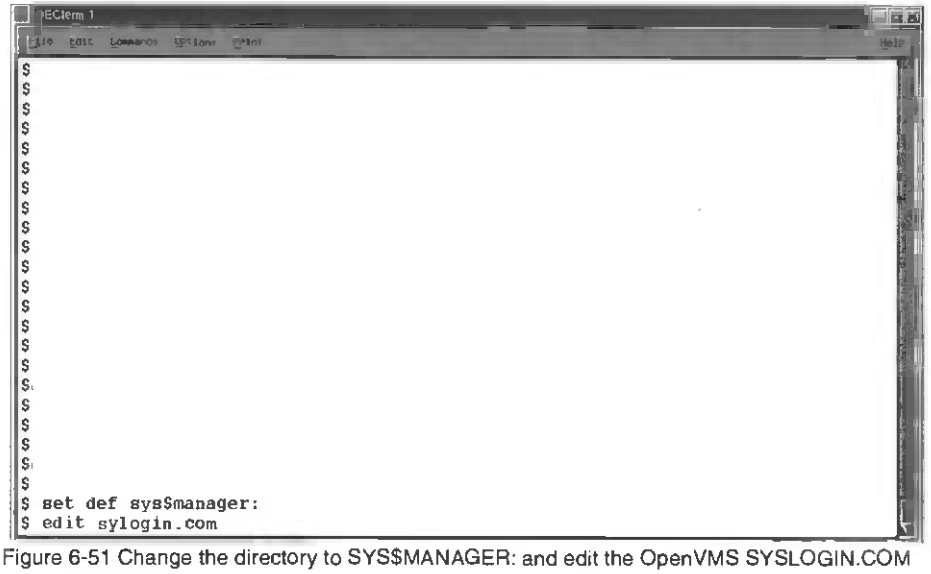
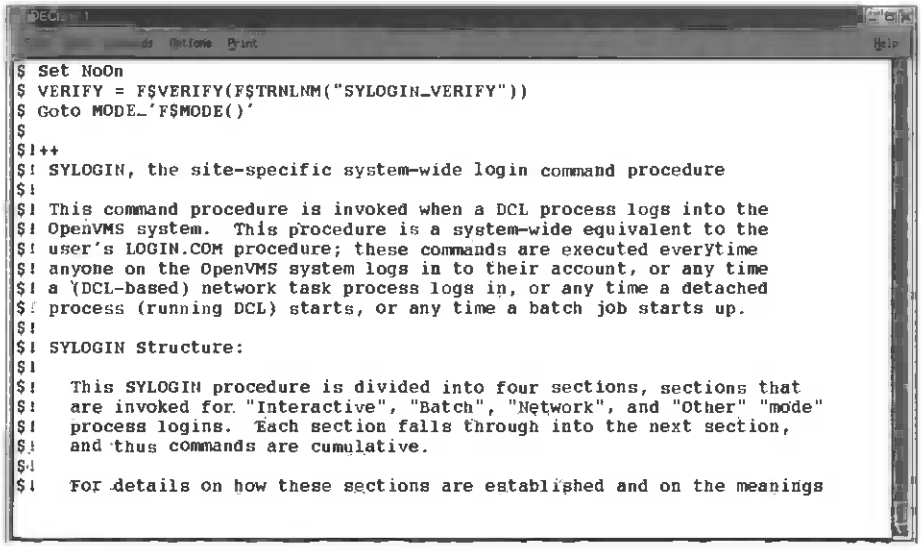
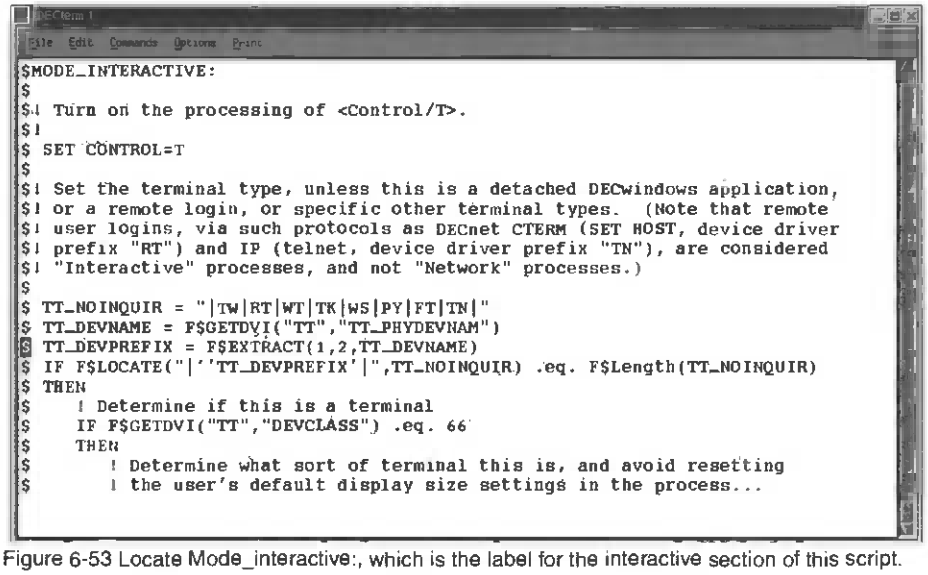


Figure 6-51 Change the directory to SYS$MANAGER: and edit the OpenVMS SYSLOGIN.COM file. This file will define commands and symbols for every user who creates a process on OpenVMS. All users get these definitions.



```
$ Set NoOn
$ VERIFY = F$VERIFY(F$TRNLNM("SYLOGIN_VERIFY"))
$ Goto MODE_'F$MODE()'
$
$!++
$! SYLOGIN, the site-specific system-wide login command procedure
$!
$! This command procedure is invoked when a DCL process logs into the
$! OpenVMS system.  This procedure is a system-wide equivalent to the
$! user's LOGIN.COM procedure; these commands are executed everytime
$! anyone on the OpenVMS system logs in to their account, or any time
$! a (DCL-based) network task process logs in, or any time a detached
$! process (running DCL) starts, or any time a batch job starts up.
$!
$! SYLOGIN Structure:
$!
$!   This SYLOGIN procedure is divided into four sections, sections that
$!   are invoked for "Interactive", "Batch", "Network", and "Other" "mode"
$!   process logins.  Each section falls through into the next section,
$!   and thus commands are cumulative.
$!
$!   For details on how these sections are established and on the meanings
```

Figure 6-52 As you edit the file, you need to note in what area you define your shortcuts. There are places for interactive, batch, network, and other modes; we will place the Mozilla definition in interactive, so when we log in users they can type Mozilla and get the program started.



```
$MODE_INTERACTIVE:
$
$! Turn on the processing of <Control/T>.
$!
$ SET CONTROL=T
$
$! Set the terminal type, unless this is a detached DECwindows application,
$! or a remote login, or specific other terminal types.  (Note that remote
$! user logins, via such protocols as DECnet CTERM (SET HOST, device driver
$! prefix "RT") and IP (telnet, device driver prefix "TN"), are considered
$! "Interactive" processes, and not "Network" processes.)
$
$ TT_NOINQUIR = "|TW|RT|WT|TK|WS|PY|FT|TN|"
$ TT_DEVNAME = F$GETDVI("TT","TT_PHYDEVNAM")
$ TT_DEVPREFIX = F$EXTRACT(1,2,TT_DEVNAME)
$ IF F$LOCATE("|"'TT_DEVPREFIX'|",TT_NOINQUIR) .eq. F$Length(TT_NOINQUIR)
$ THEN
$    ! Determine if this is a terminal
$    IF F$GETDVI("TT","DEVCLASS") .eq. 66
$    THEN
$       ! Determine what sort of terminal this is, and avoid resetting
$       ! the user's default display size settings in the process...
```

Figure 6-53 Locate Mode_interactive:, which is the label for the interactive section of this script.

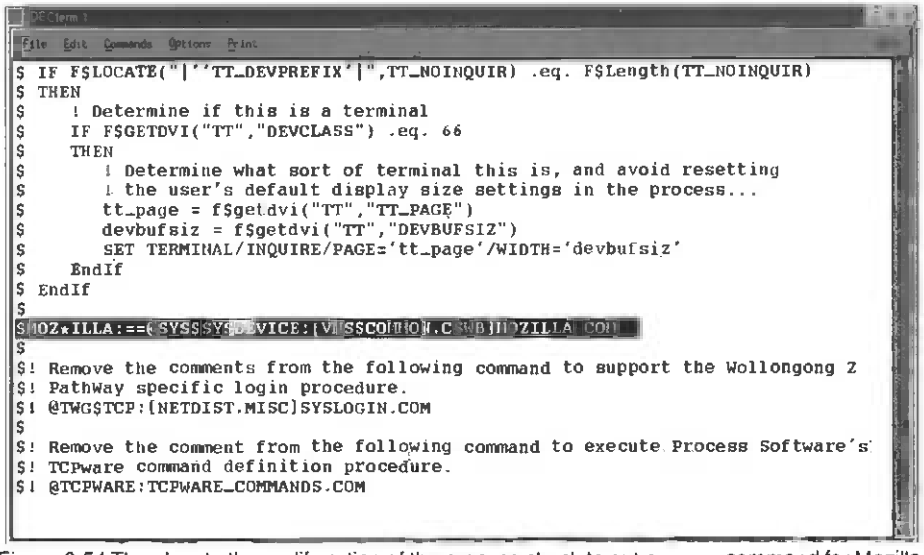


Figure 6-54 Then locate the endif portion of the process check to put our new command for Mozilla. Note that the * in "moz*illa" allows us to use the shorthand "moz" from a dollar sign and have it recognized as if we had typed out the whole string "mozilla."

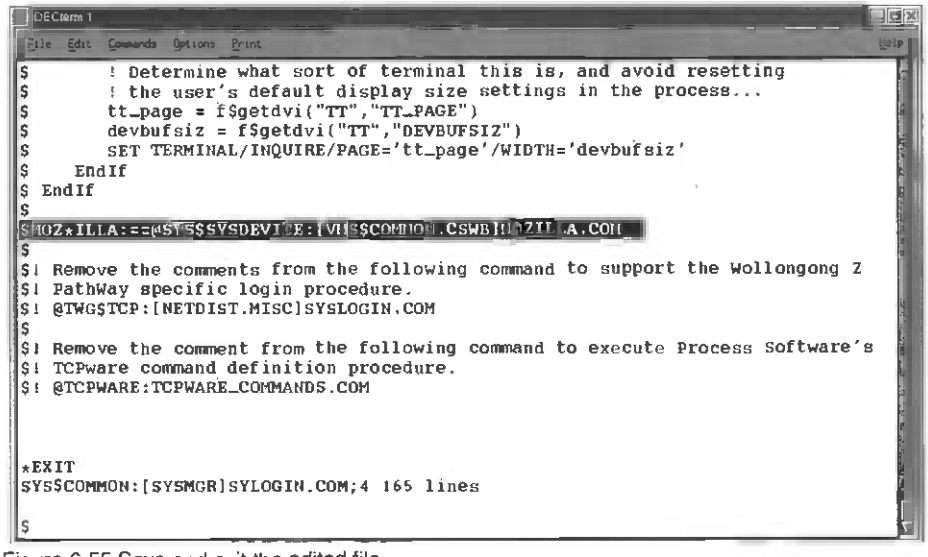


Figure 6-55 Save and exit the edited file.

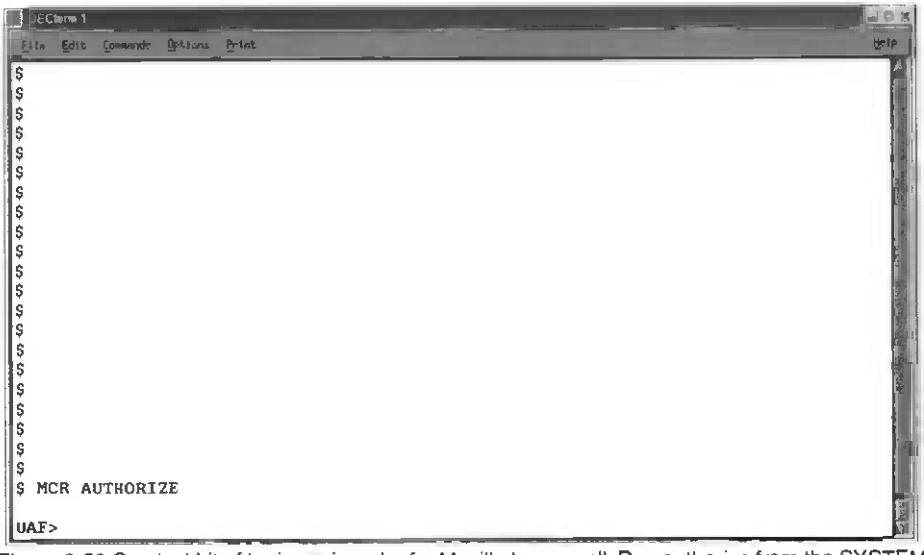


Figure 6-56 One last bit of tuning is in order for Mozilla to run well. Run authorize from the SYSTEM account.

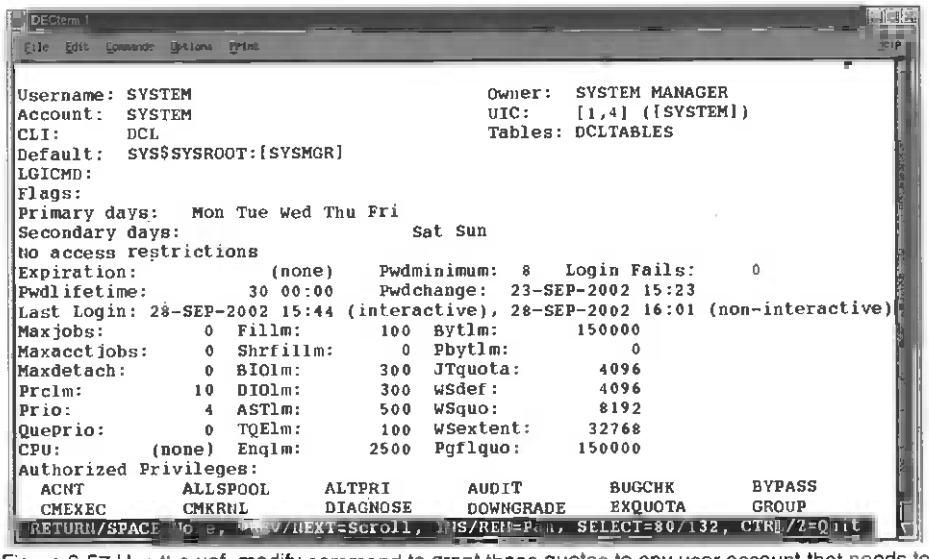


Figure 6-57 Use the uaf>modify command to grant these quotas to any user account that needs to run Mozilla on OpenVMS. Note that these accounts only need tempmbx and netmbx privileges active to run Mozilla, but this is the SYSTEM ACCOUNT shown with all privileges!

Figure 6-58 Exit from the authorize utility and run Mozilla for the first time.

Congratulations! You've just installed the Mozilla Web browser on your OpenVMS and Linux systems.

Mail Client Configuration on OpenVMS or Linux

The nice thing about OpenVMS and Linux Open Source applications is that they all look, feel, and run the same way after installation, no matter the platform! Let's look at how to configure Mozilla for POP3 or IMAP; note that both are exactly the same from an application point of view (see Figures 6-60 through 6-70)!

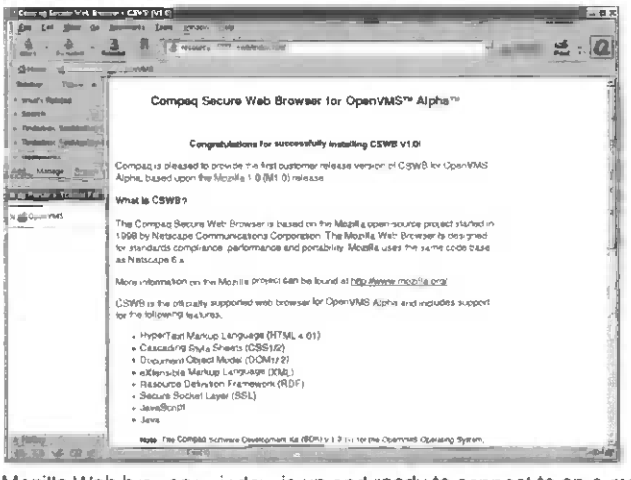


Figure 6-59 The Mozilla Web browser window is up and ready to connect to an e-mail server either remotely or across the local network to the server it's running on.

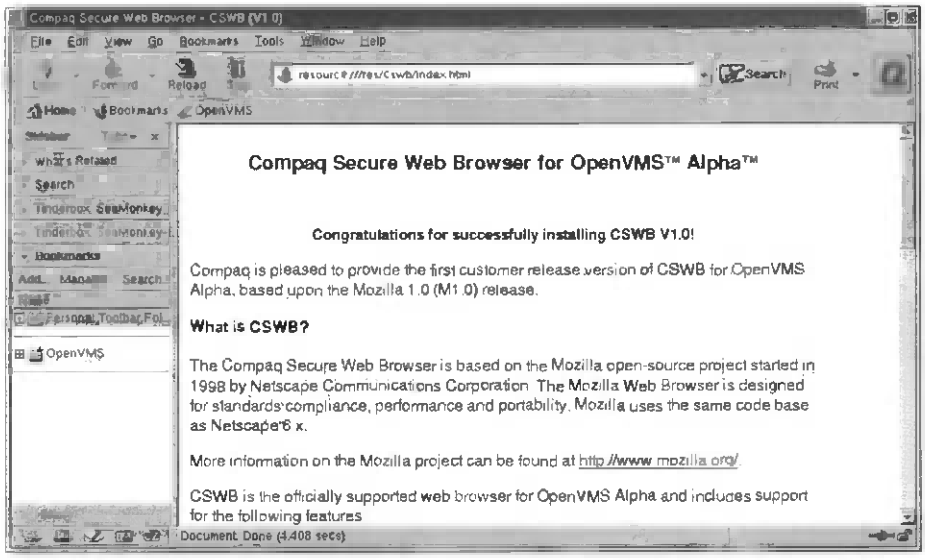


Figure 6-60 Pull down the Edit menu on the tool bar at the top of the window. Chose "Preferences" and release the menu choice.

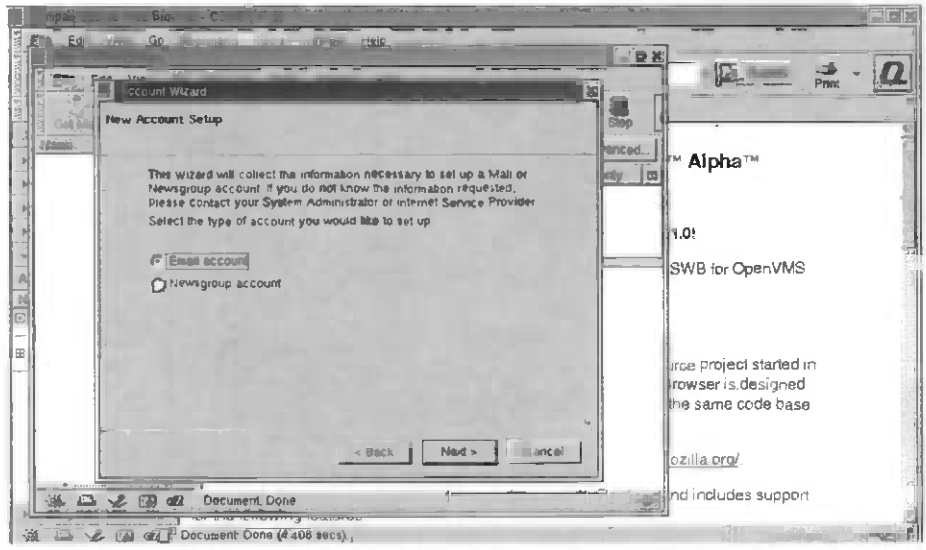


Figure 6-61 The pop-up window asks what kind of account we would like chose e-mail.

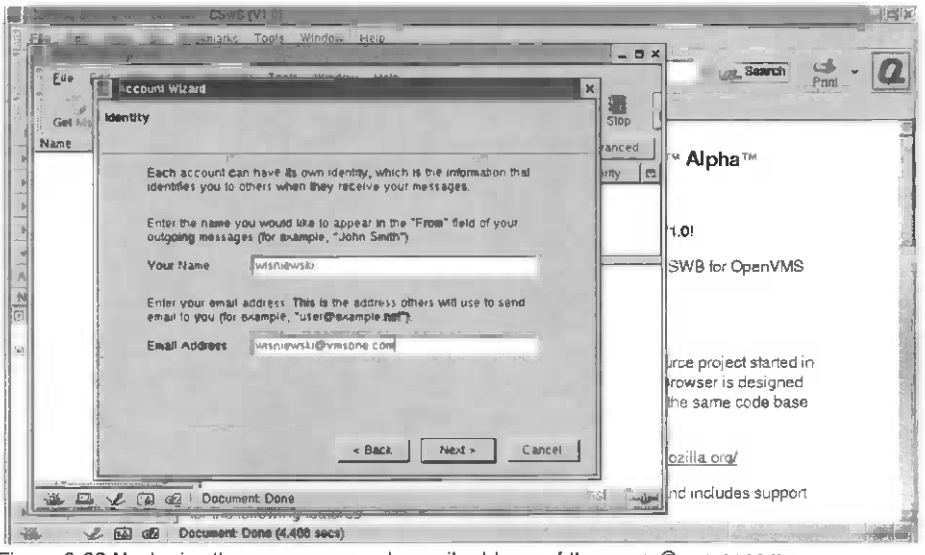


Figure 6-62 Next, give the user name and e-mail address of the users@server.com.

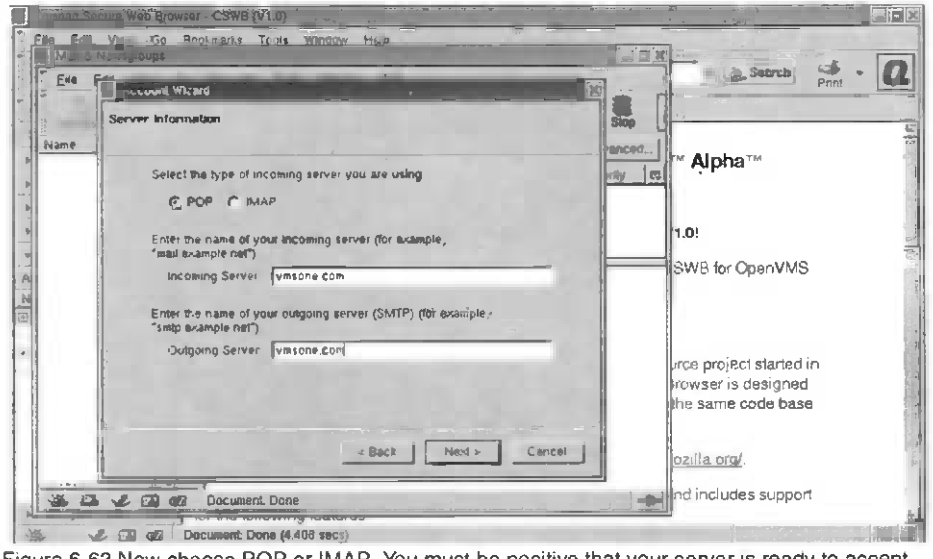


Figure 6-63 Now choose POP or IMAP--You must be positive that your server is ready to accept POP or IMAP connections! Also, you need an outgoing mail server to process your outgoing client e-mail.

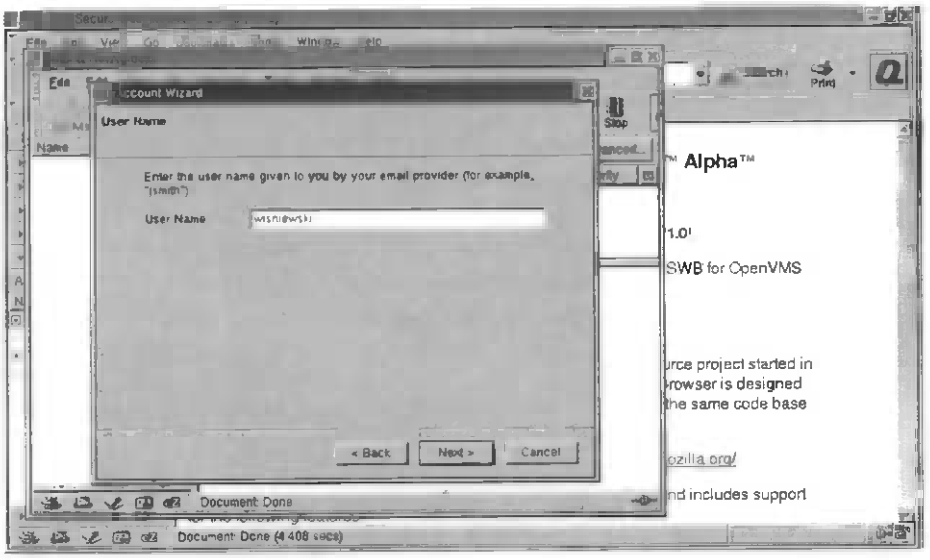


Figure 6-64 Now enter the user name of the account you are going to send and receive e-mail through.

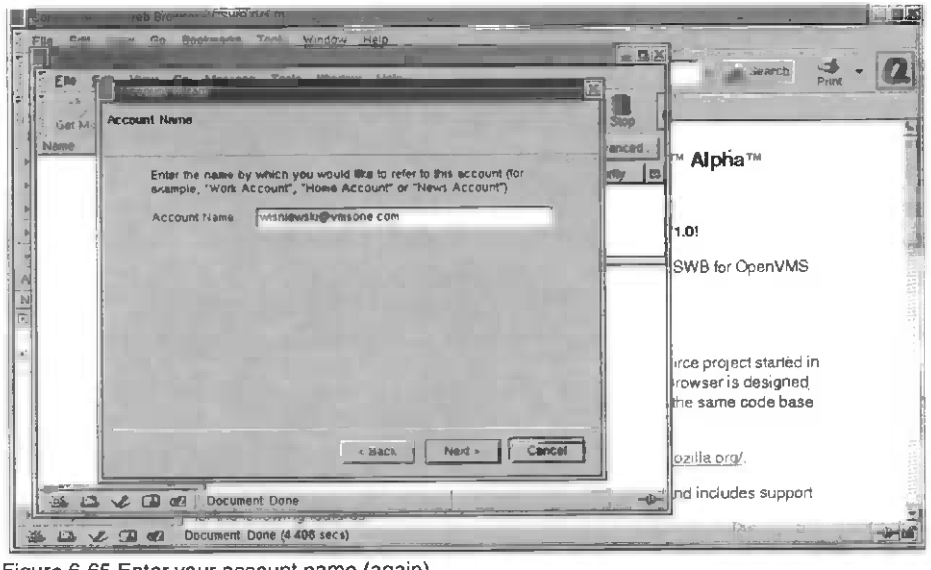


Figure 6-65 Enter your account name (again).

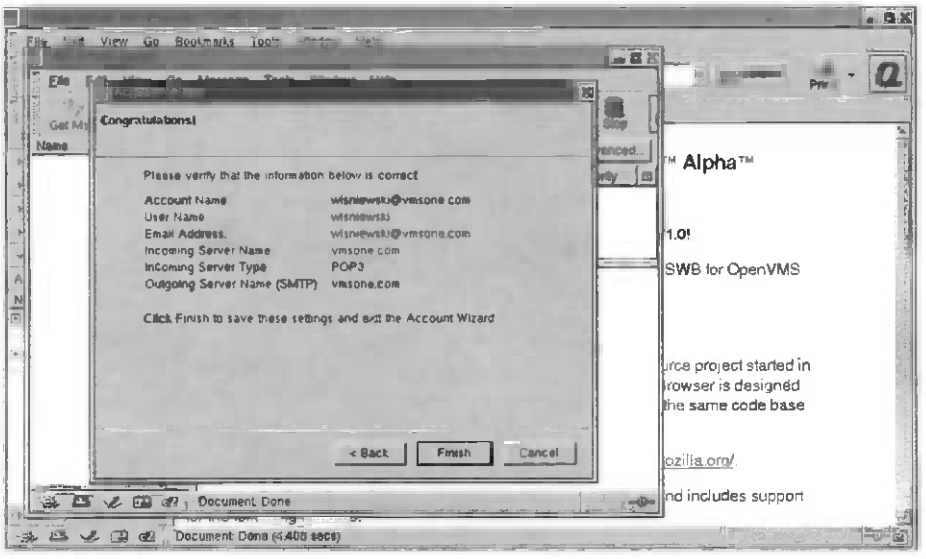


Figure 6-66 Verify your account information (or change).

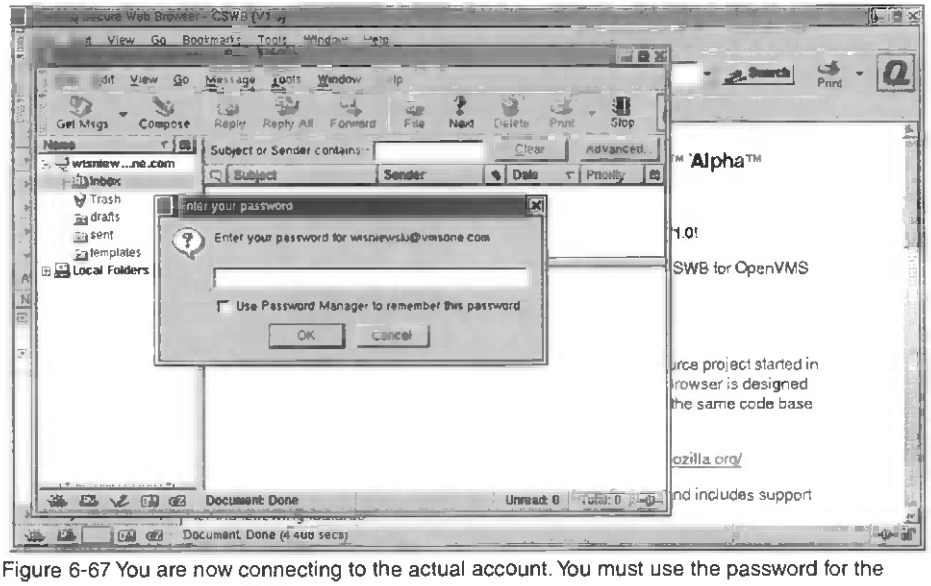


Figure 6-67 You are now connecting to the actual account. You must use the password for the account as if you are logging in interactively.

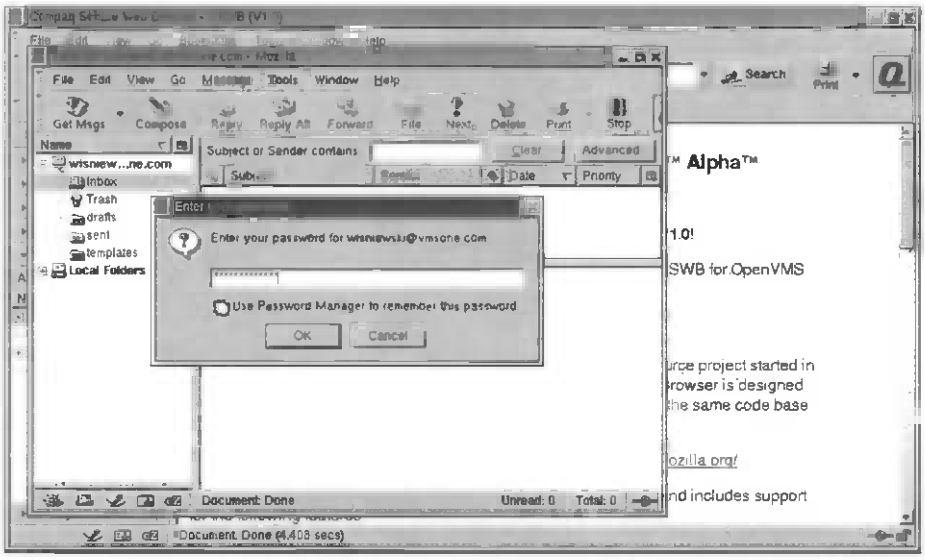


Figure 6-68 Type in the password.

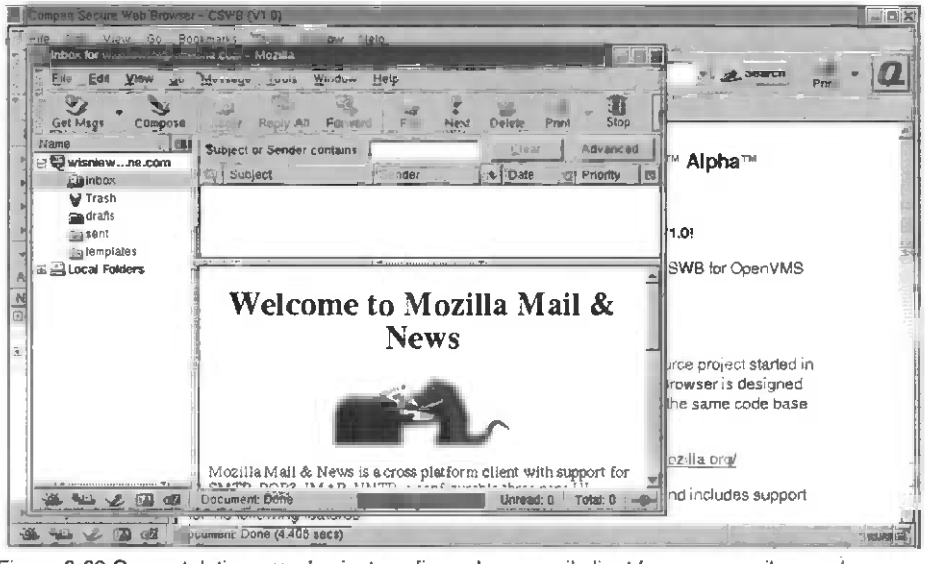


Figure 6-69 Congratulations, you've just configured an e-mail client for your e-mail server!

Figure 6-70 Since this is a POP3 connection, you need to download the e-mail messages this account has to your Mozilla e-mail client.

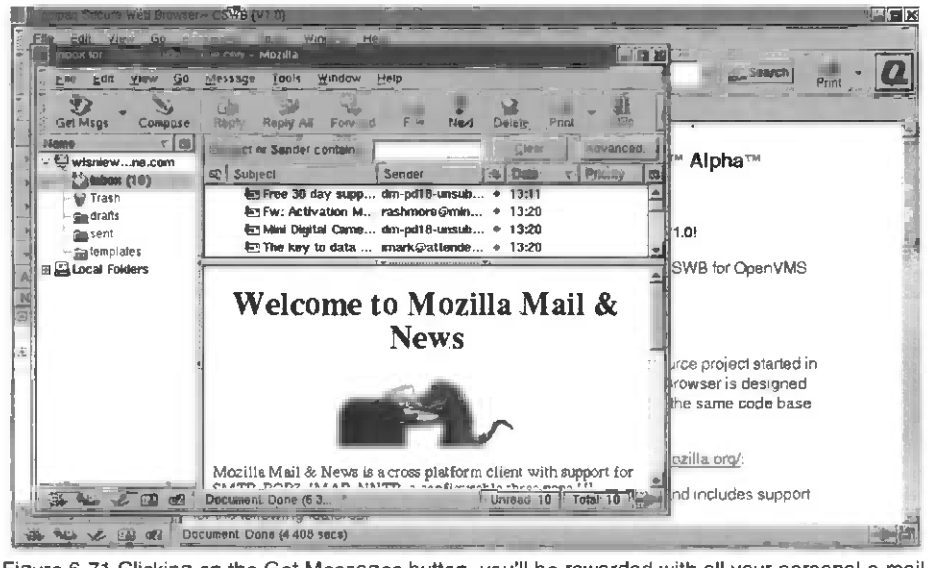


Figure 6-71 Clicking on the Get Messages button, you'll be rewarded with all your personal e-mail and offers you just can't refuse!

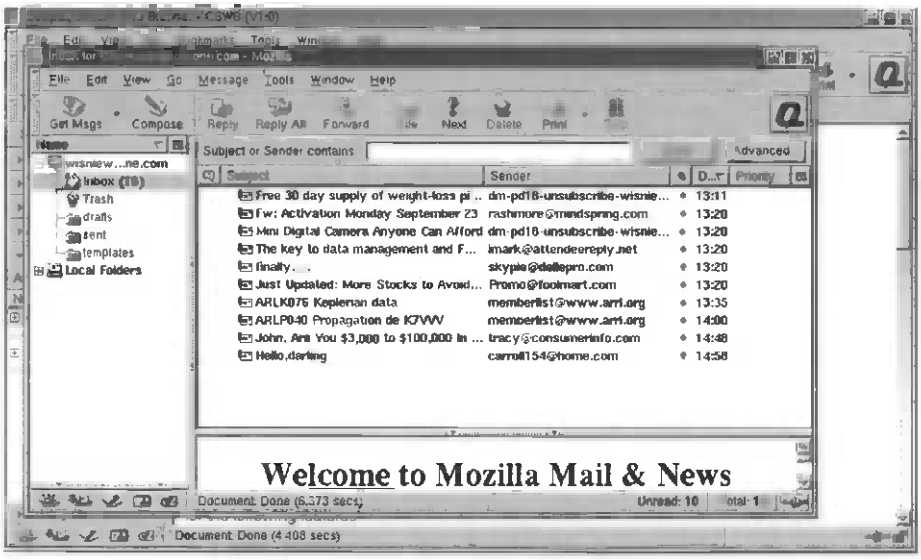


Figure 6-72 Looking over the list there is only one obvious choice to read first.

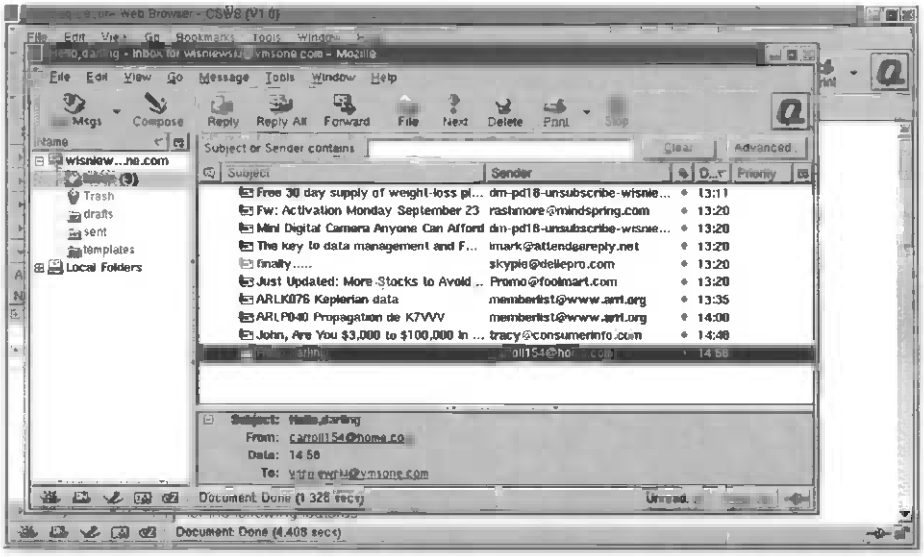


Figure 6-73 Ah, l'amour from the Internet, but a gentleman doesn't spam and tell so let's look at a more typical e-mail message.

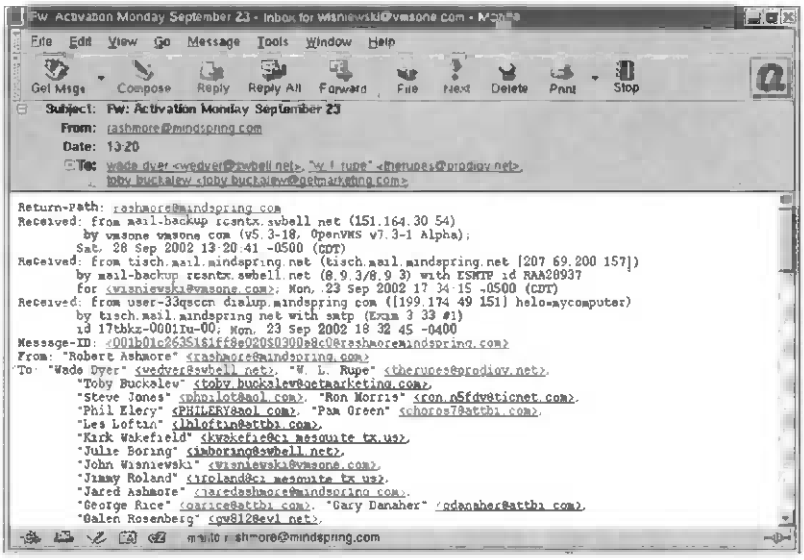


Figure 6-74 It looks like the local ham radio group is being called up to support a storm or hurricane.

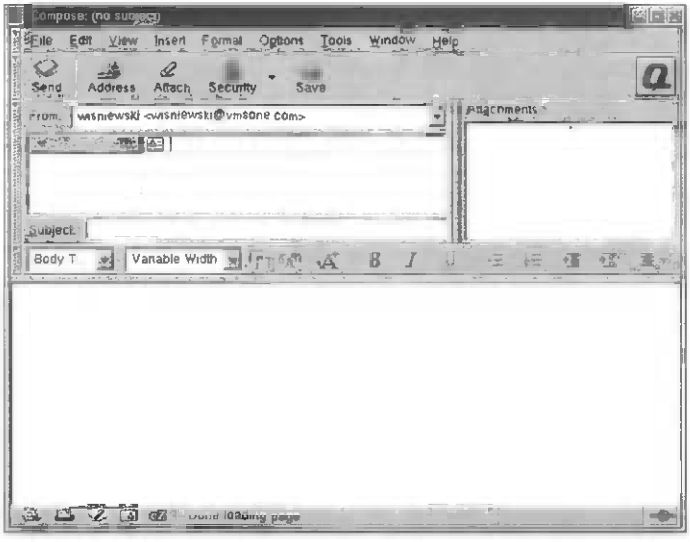


Figure 6-75 Using standard Mozilla you can then write and reply (with attachements) to an e-mail using nothing but Open Source and standard Internet e-mail tools!

E-mail Wants to be Free

E-mail across OpenVMS and RedHat 7.3 is included with the base distribution at no additional charge. OpenVMS adds some additional checking and testing of Open Source e-mail (seeing as it writes the TCP/IP stack and ports Mozilla to OpenVMS). Linux and OpenVMS can use the same standard tools to send e-mail, trade files, share disks, and browse each other's Web sites. E-mail is usually the first application to be established in a network—then, knowing that works, the others are easy. Well, at least we know e-mail works.

Chapter 7 — SMB for Linux and OpenVMS

Using SMB with OpenVMS and Linux for Disk Sharing

Windows, Linux, and OpenVMS systems have been invading each others' turf for the last 10 years. Managers and other "normal" people just expect not only that they will play together nicely, but that they will be able to share files, disks, and printers. This is a simple goal, if an unreasonable request!

There are other disk-sharing protocols, such as NFS in the UNIX world or DFS in the VMS world, but Windows has had the edge by sheer numbers, so its SMB protocol is much more likely to be used between different operating systems.

Windows NT and Windows 2000 server systems, with some client licenses, provide disk and printer sharing plus authentication to use the resources. What was required was Open Source tools on other operating systems to provide integrated file, print, and authentication.

Enter SAMBA

From those humble needs SAMBA was created. SAMBA has become a GNU-licensed, Open Source disk-sharing and printer-sharing network protocol that will let your Linux, OpenVMS, or UNIX systems participate in a Windows workgroup, NT 4.0 domain, or Windows Enterprise domain. The software is distributed for free at http://www.samba.org for OpenVMS and Linux. SAMBA is included as part of the standard RedHat 7.3 distribution.

All of this is managed through a protocol suite currently known as the Common Internet File System (CIFS). This name was introduced by Microsoft and provides some insight into its hopes for the future. At the heart of CIFS is the latest incarnation of the Server Message Block (SMB) protocol, which has a long history. Suffice it to say that SAMBA is an Open Source CIFS implementation.

SMB Options for OpenVMS, Advanced Server, or SAMBA

OpenVMS SAMBA is also available from samba.org, but for OpenVMS there is a commercial option for people who need better integration with their Enterprise Windows environments. HP/OpenVMS engineering has licensed the source code for Advanced Server from Microsoft as a the result of an AT&T lawsuit, which resulted in AT&T having access to source code and an industry relicensing agreement that only the lawyers can understand the true nature of. Still, Microsoft periodically updates

this source code with new features and improvements and delivers it to AT&T, which then updates OpenVMS engineering and engages a team to port, evaluate, and test the product before sending out binaries to OpenVMS Advanced Server customers around the world.

The advantage of using the OpenVMS Advanced Server software instead of the free SAMBA software is commercial support with an 1-800 number from HP. As a commercially supported product from HP, Advanced Server also tends to have Microsoft Server features before SAMBA. (at the time of this writing SAMBA is waiting for the new LDAP, Active DNS, and the new security model in the latest MS products and Advanced Server implementations.) SAMBA will get the latest integration features with MS products, but SAMBA usually lags behind the commercial products in those features, while the Open Source community reengineers the next version of SAMBA.

Support service and testing are what make Advanced Server for OpenVMS the preferred commercial choice for SMB file and printer sharing on OpenVMS with a commercial Windows environment.

That's not to say that SAMBA for OpenVMS isn't making a lot of friends! The OpenVMS hobbyist communities and those who only need file and printer sharing find that SAMBA's Open Source and free cost are most efficacious for sharing with their UNIX, Linux, and Windows clients! With OpenVMS you really do have two choices for your SMB server based on your specific needs and your pocketbook!

Linux and SAMBA

The ability to share file and print services today is assumed for all operating systems. Windows is the largest client population likely to be found on a network. So the easiest way to do this is to add Windows security and sharing to Linux. This not only offers an authentication mechanism common across most desktop platforms but also includes file and printer services for Windows clients that can be stored on a Linux Server.

Installing Linux as an SMB server with RedHat 7.3 allows you to choose an SMB server, which will automatically install SAMBA for you (see Figures 7-1 through 7-13). It installs SAMBA, but it's up to you to configure it!

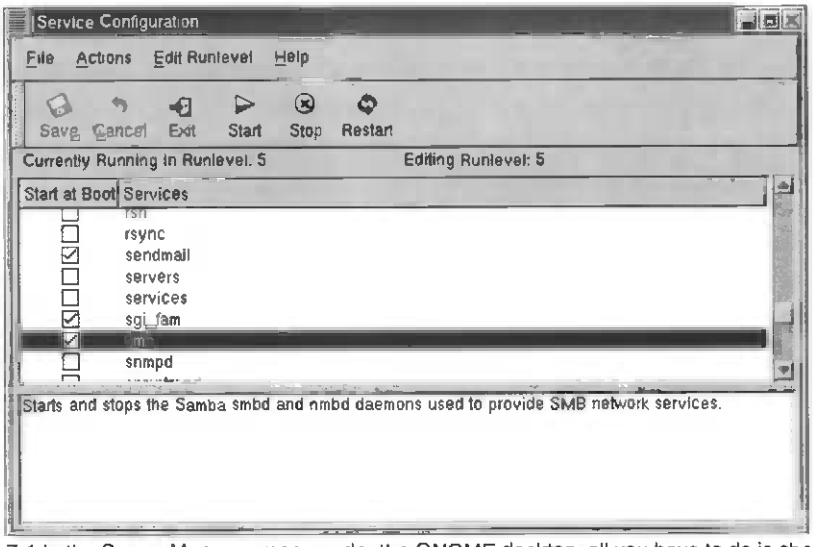


Figure 7-1 In the Server Manager menu under the GNOME desktop, all you have to do is check the box and save your choices.

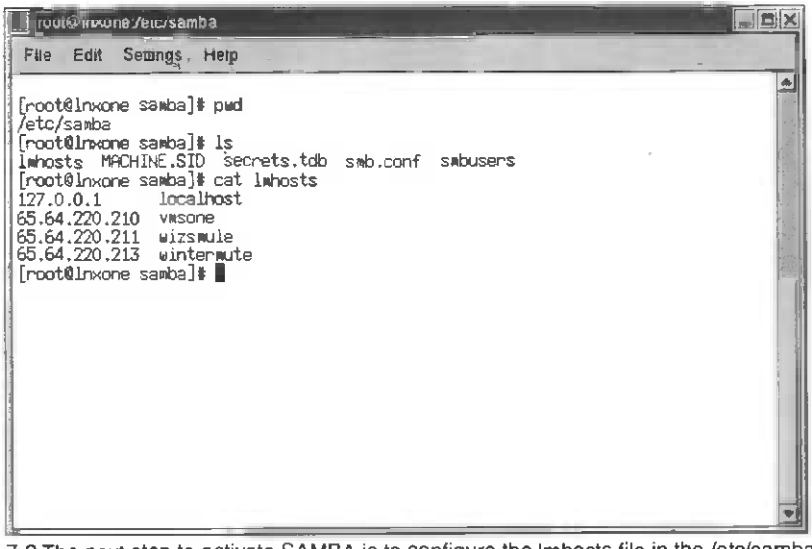


Figure 7-2 The next step to activate SAMBA is to configure the lmhosts file in the /etc/samba directory, use your favorite Linux editor, and add the "LAN Manager" names of the other hosts in your workgroup or domain.

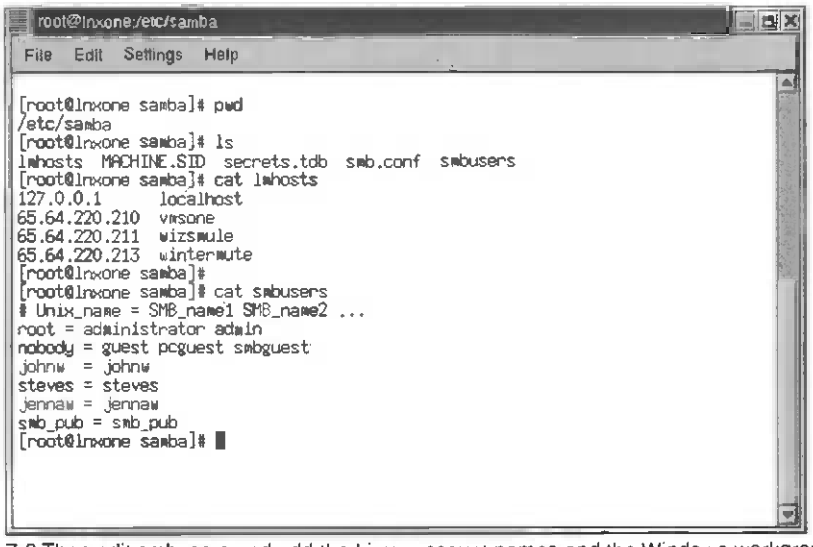


Figure 7-3 Then edit smbusers and add the Linux account names and the Windows workgroup/domain names that you want to map them to. Note that you can map multiple Windows users to a single Linux account!

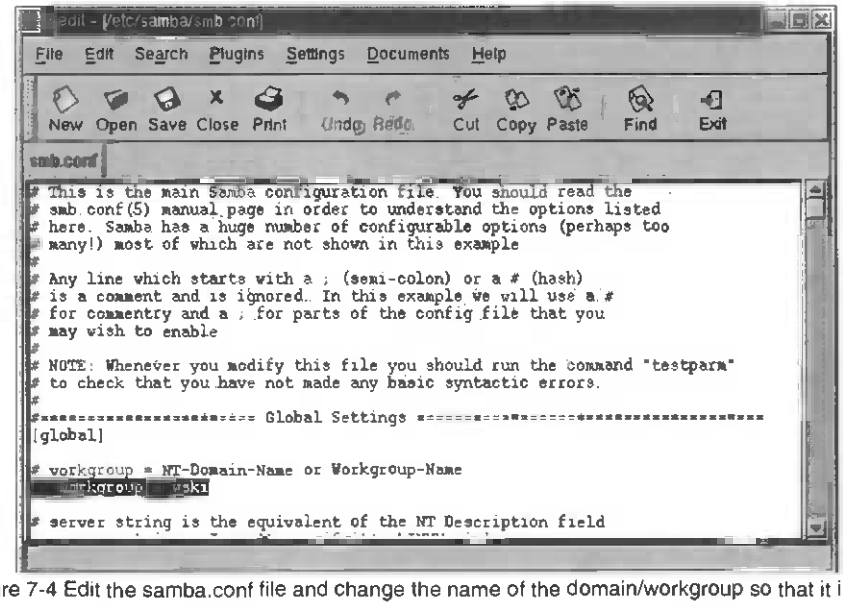


Figure 7-4 Edit the samba.conf file and change the name of the domain/workgroup so that it is the same as the domain/workgroup that your PC (and other Linux) clients are using.

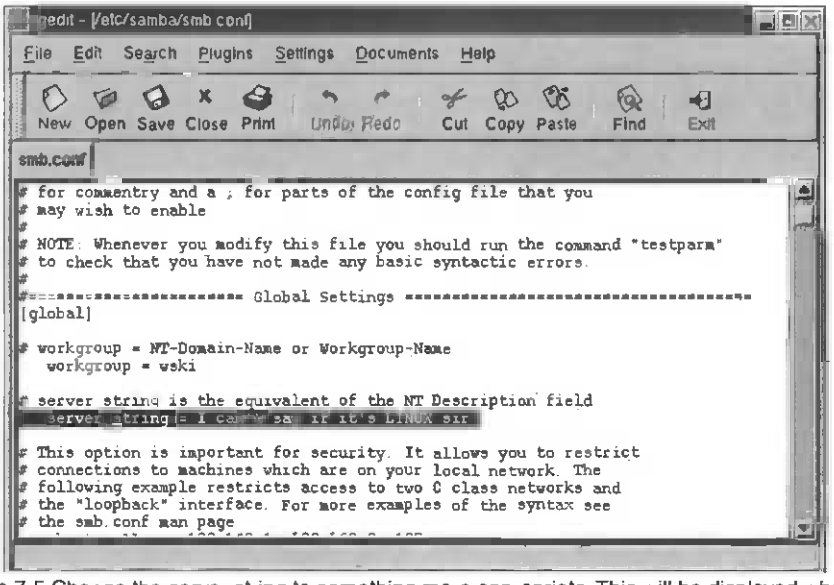


Figure 7-5 Change the server string to something more appropriate. This will be displayed when a client browses your SMB Server.

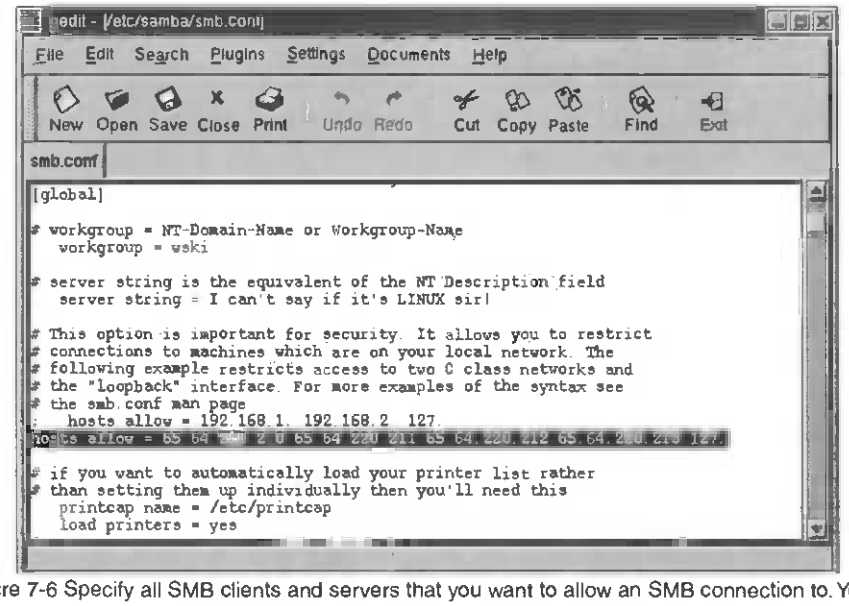


Figure 7-6 Specify all SMB clients and servers that you want to allow an SMB connection to. You can use wildcards!

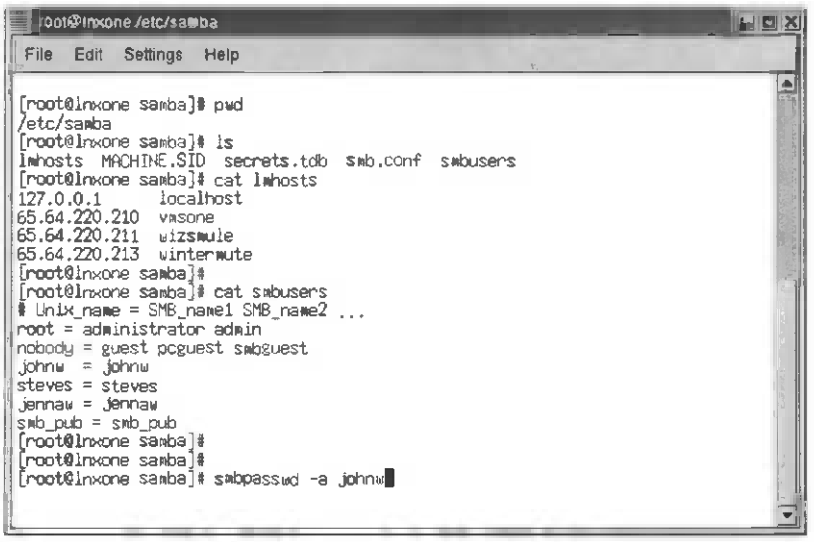


Figure 7-7 After that is done you must run smbpasswd as the root user and create an SMB password file (from the first time it's run). Then add passwords for all the Linux user account names that will be connecting to your SMB Server.

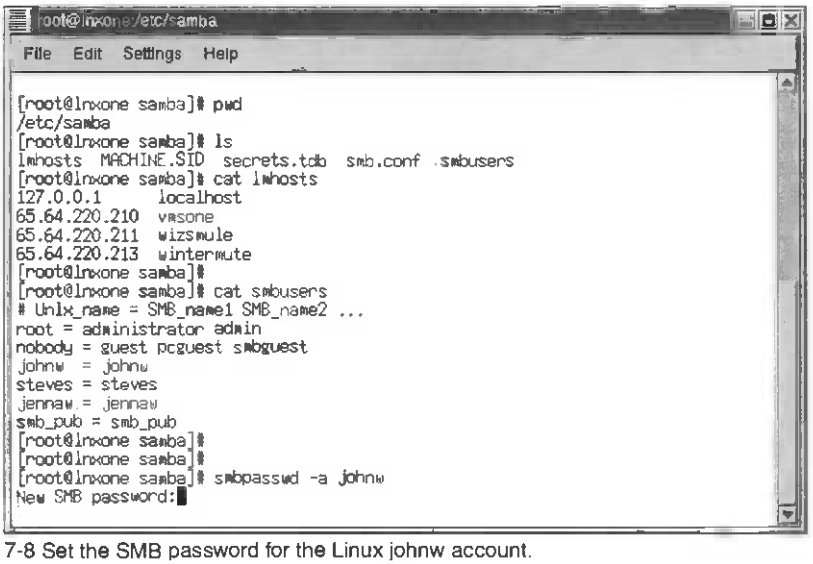


Figure 7-8 Set the SMB password for the Linux johnw account.

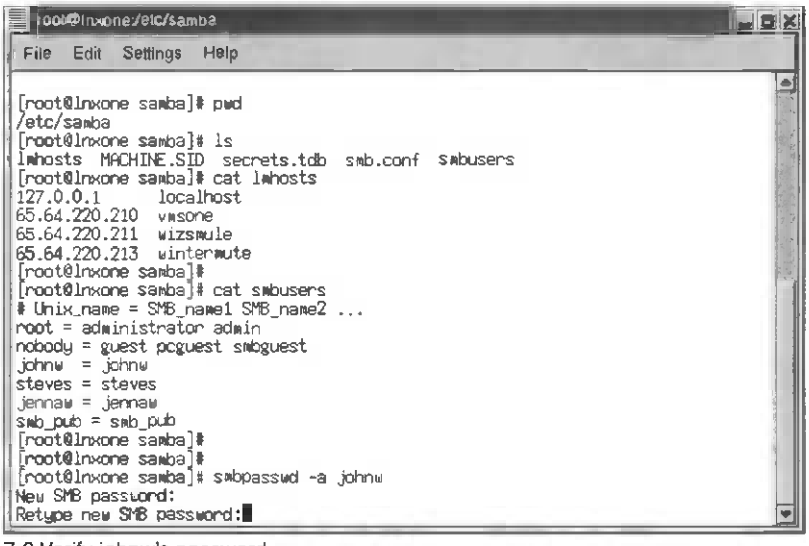


Figure 7-9 Verify johnw's password.

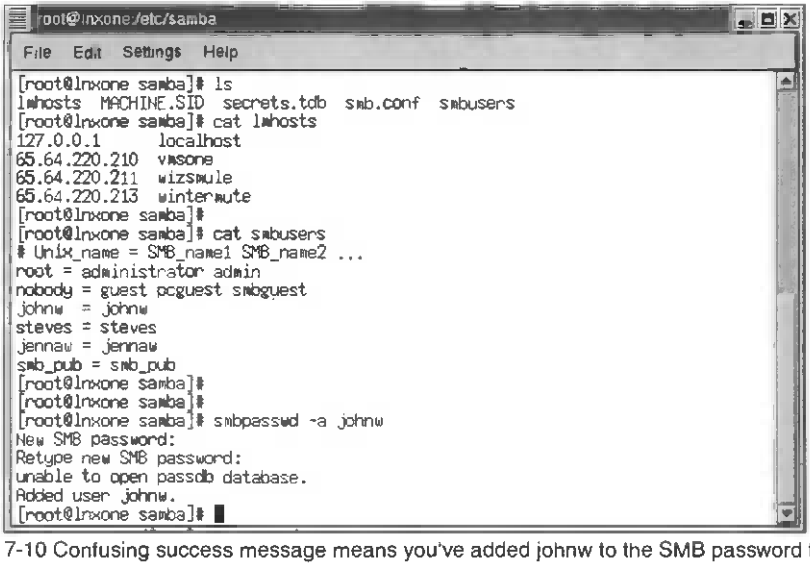


Figure 7-10 Confusing success message means you've added johnw to the SMB password file.

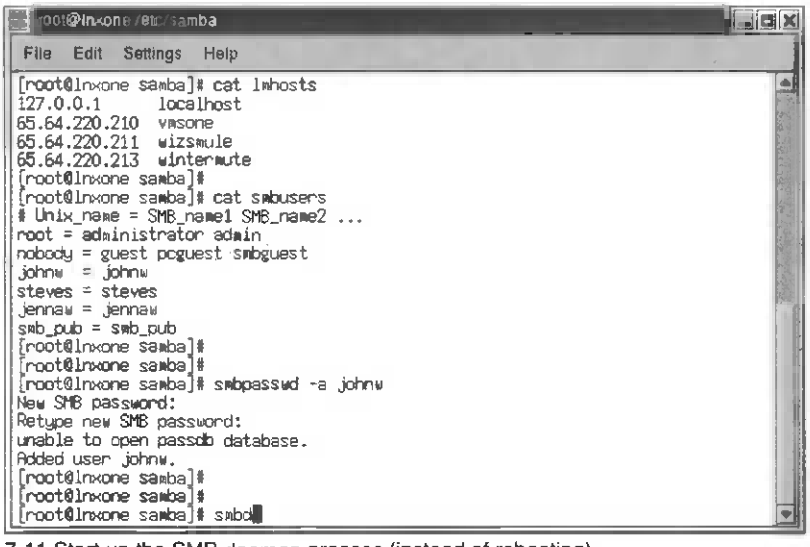


Figure 7-11 Start up the SMB daemon process (instead of rebooting).

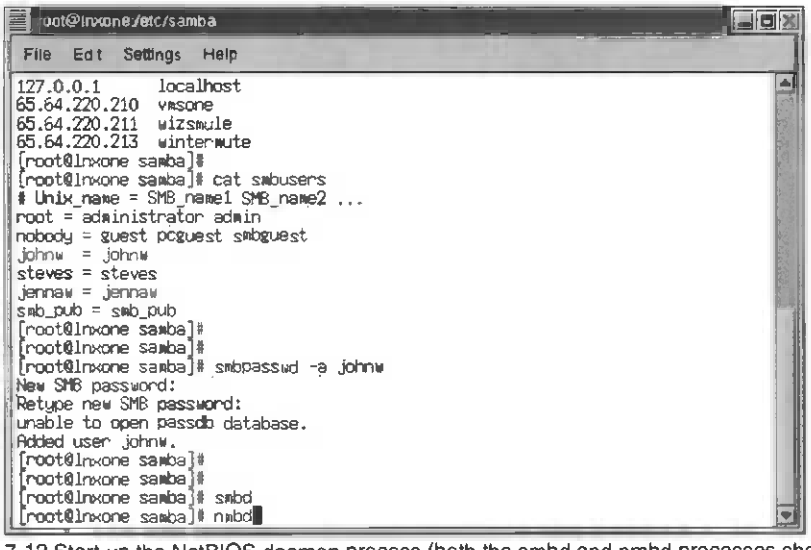


Figure 7-12 Start up the NetBIOS daemon process (both the smbd and nmbd processes should start up at Linux boot).

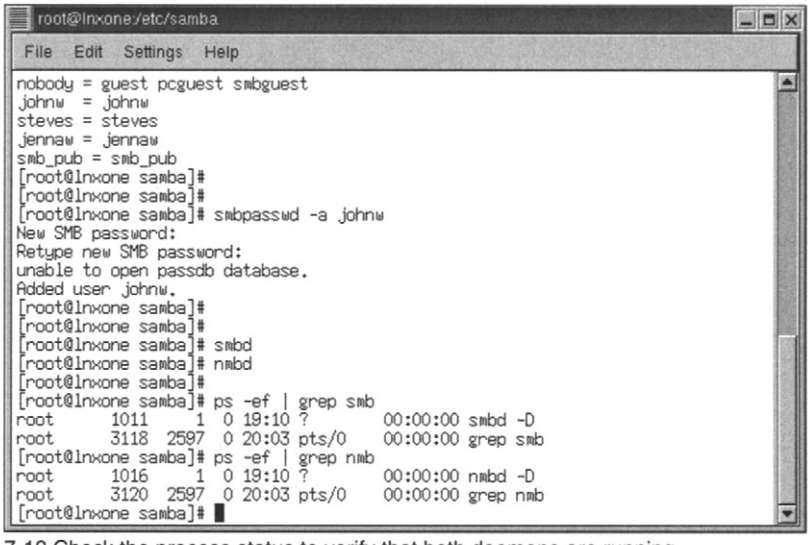


Figure 7-13 Check the process status to verify that both daemons are running.

OpenVMS Advanced Server Installation and Configuration

To install OpenVMS Advanced Server as a primary domain controller (NT 4.0 security model) you must perform five steps as follows:

1. Configure the OpenVMS registry (see Figures 7-14 through 7-26).

2. Configure OpenVMS IP for Advanced Server (see Figures 7-27 through 7-34).

3. Install OpenVMS Advanced Server from the layered products CD-ROM (see Figures 7-35 through 7-45).

4. Configure OpenVMS Advanced Server processes (see Figures 7-46 through 7-65).

5. Configure OpenVMS Advanced Server as a domain controller (see Figures 7-66 through 7-73).

Step 1: Configure the OpenVMS registry

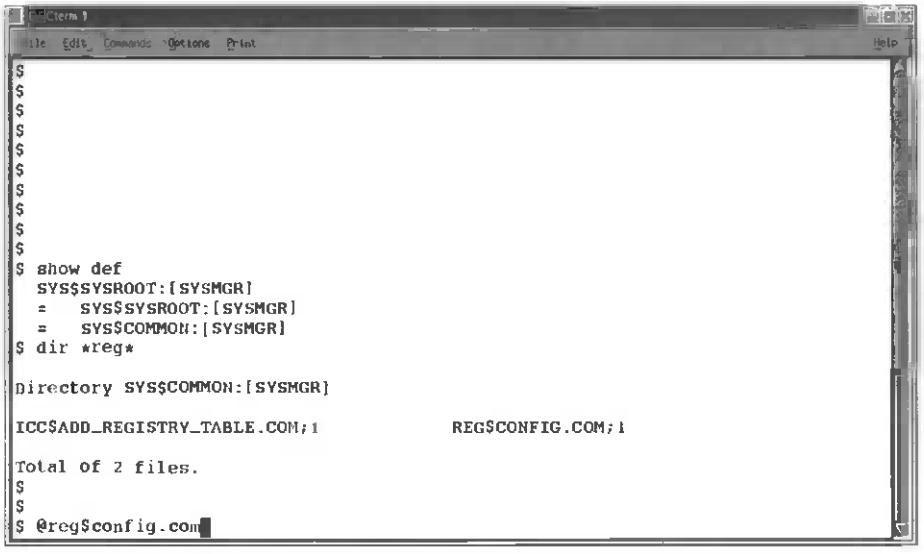


Figure 7-14 To begin configuring the OpenVMS registery, execute script @SYS$MANAGER:REG$CONFIG.COM.

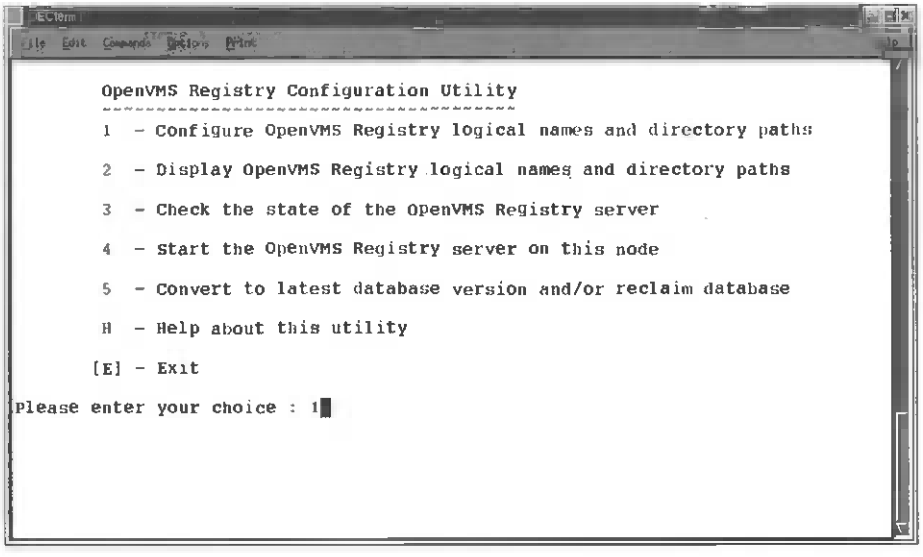


Figure 7-15 Run the menu and chose the first choice.

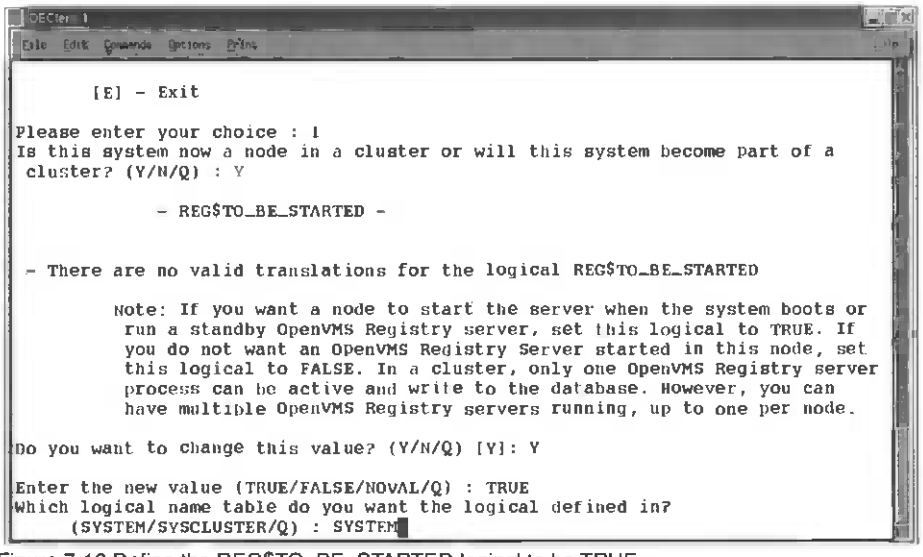


Figure 7-16 Define the REG$TO_BE_STARTED logical to be TRUE.

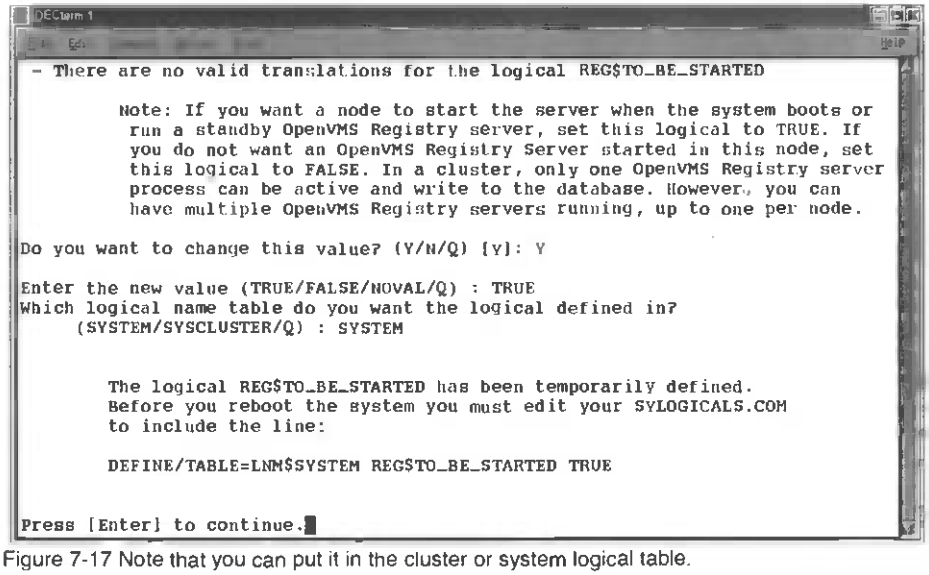


Figure 7-17 Note that you can put it in the cluster or system logical table.

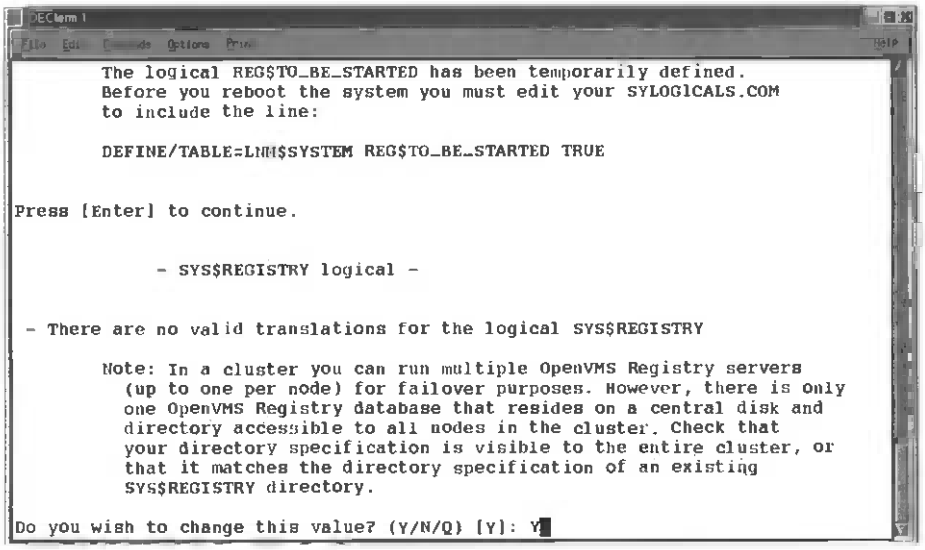


Figure 7-18 Now we need to chose a disk/directory for the VMS registry files. I've used SYS$SYSDEVICE:[VMS_REGISTRY].

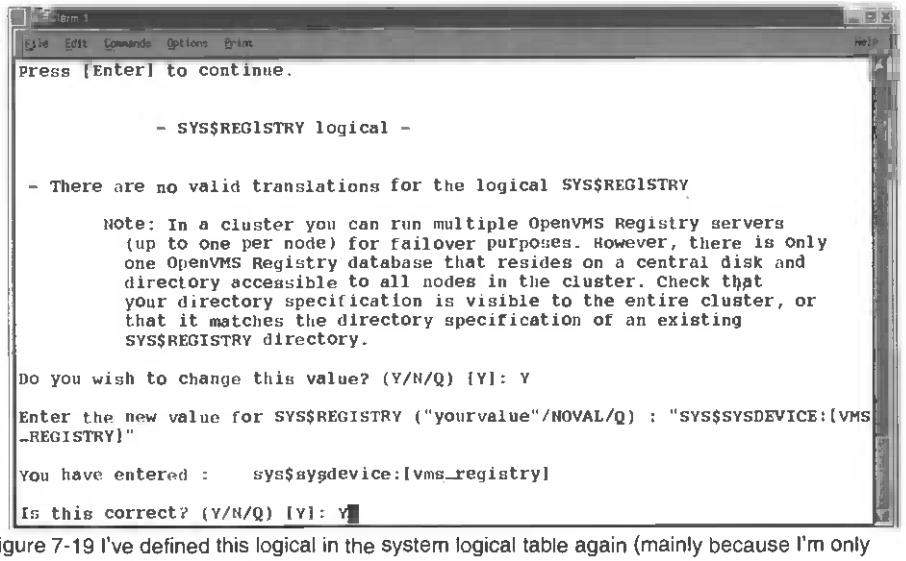


Figure 7-19 I've defined this logical in the system logical table again (mainly because I'm only configuring a single system).

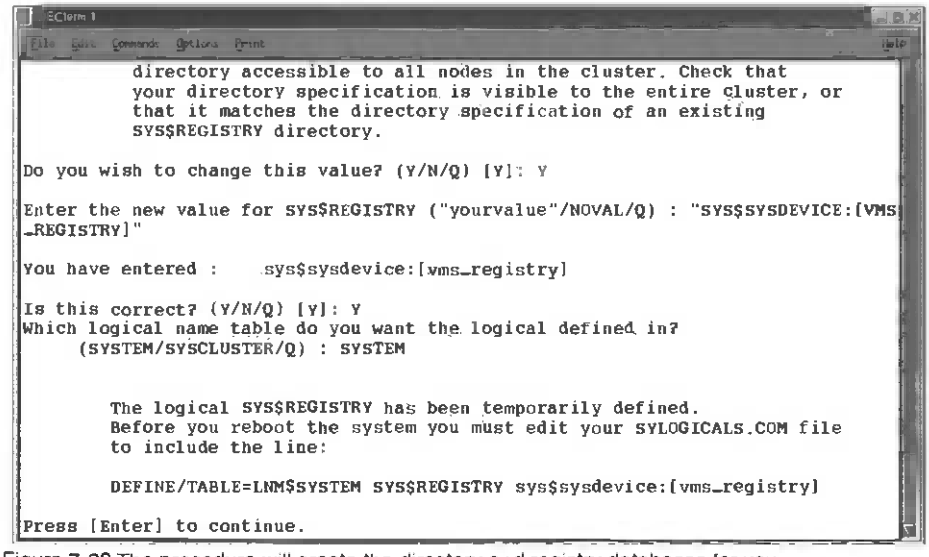


Figure 7-20 The procedure will create the directory and registry databases for you.

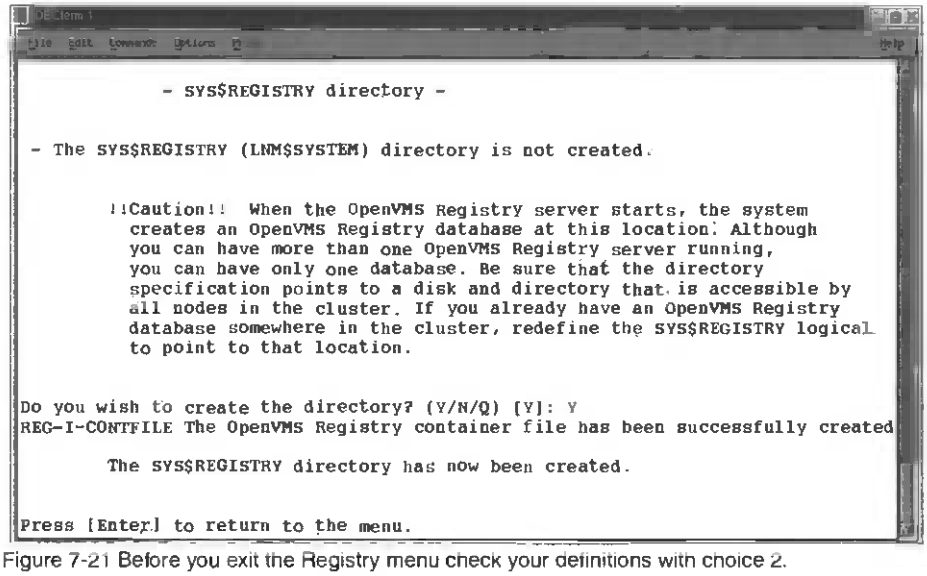


Figure 7-21 Before you exit the Registry menu check your definitions with choice 2.

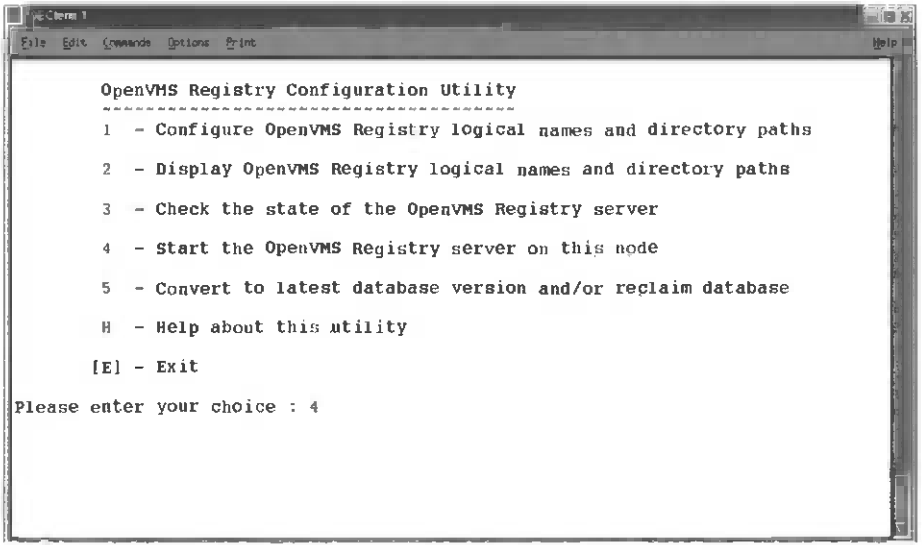


Figure 7-22 Then start up the OpenVMS registry with menu choice 4.

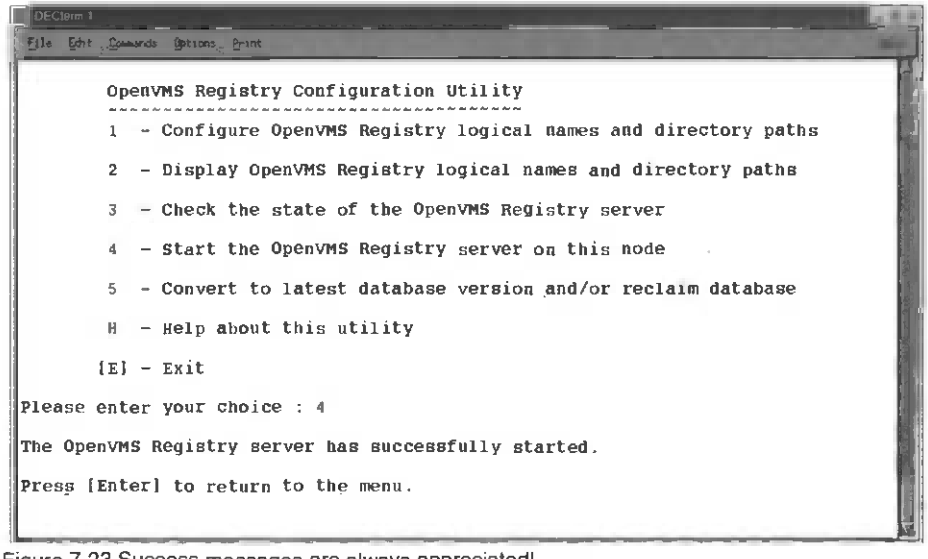


Figure 7-23 Success messages are always appreciated!

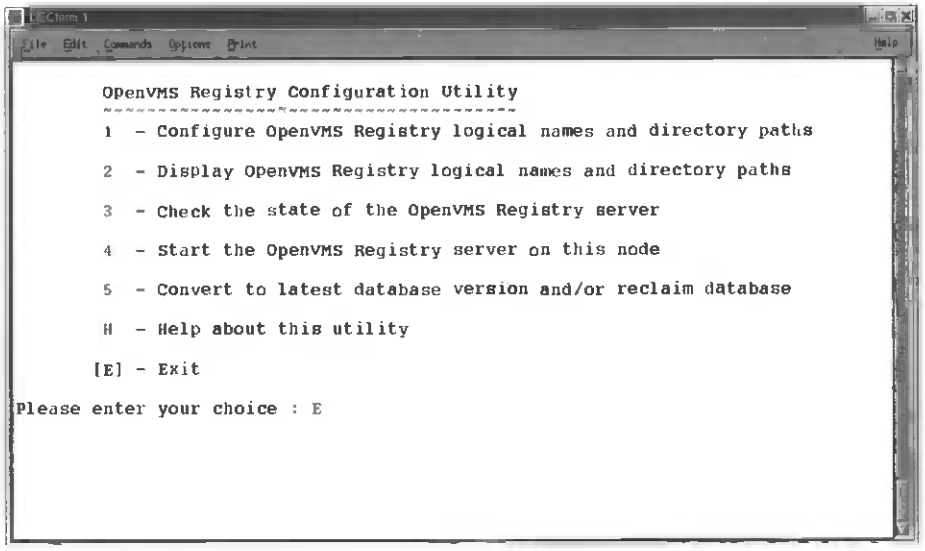


Figure 7-24 Exit the Registry configuration menu, but we're not done yet.

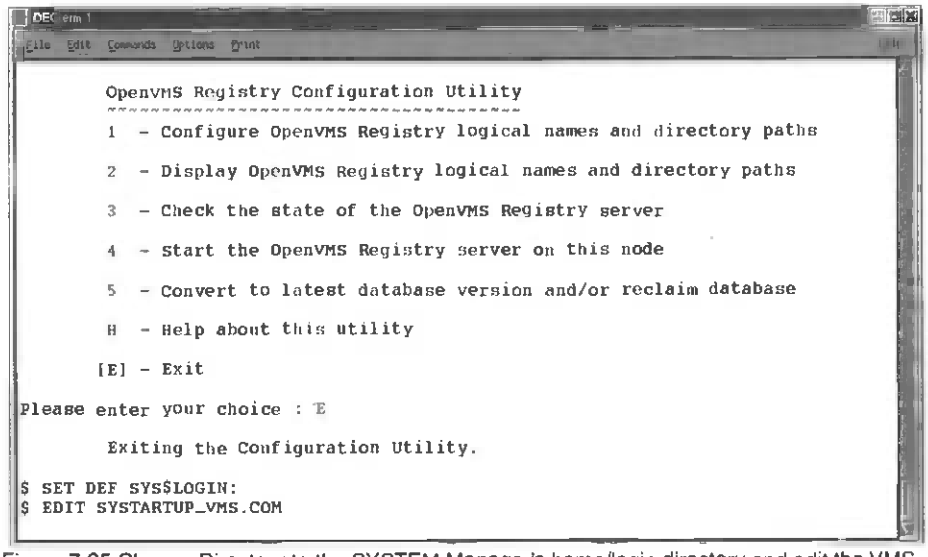


Figure 7-25 Change Directory to the SYSTEM Manager's home/login directory and edit the VMS startup file.

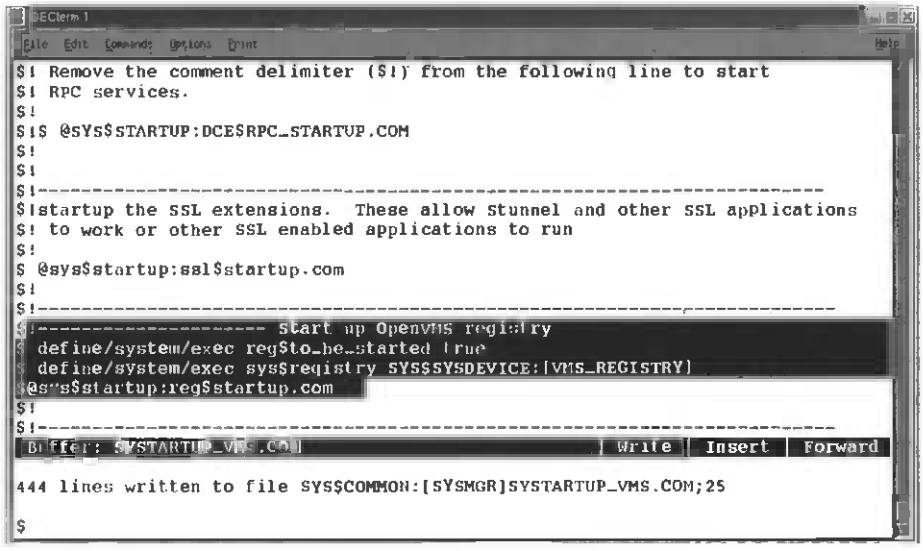


Figure 7-26 Edit the startup file, add the logical definitions, and run the registry startup so that the VMS registry will start up every time the system boots.

Step 2: Configure OpenVMS IP for Advanced Server

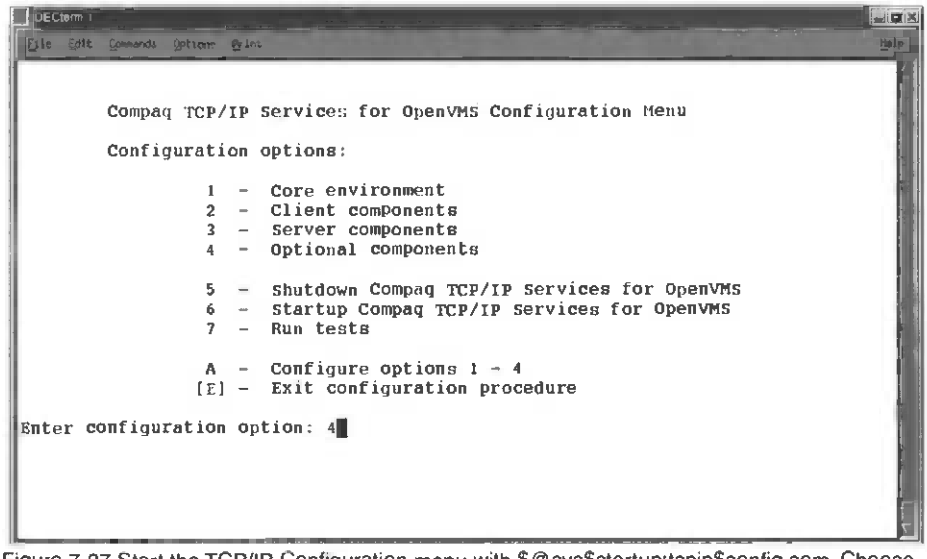


Figure 7-27 Start the TCP/IP Configuration menu with $@sys$startup:tcpip$config.com. Choose menu choice 4.

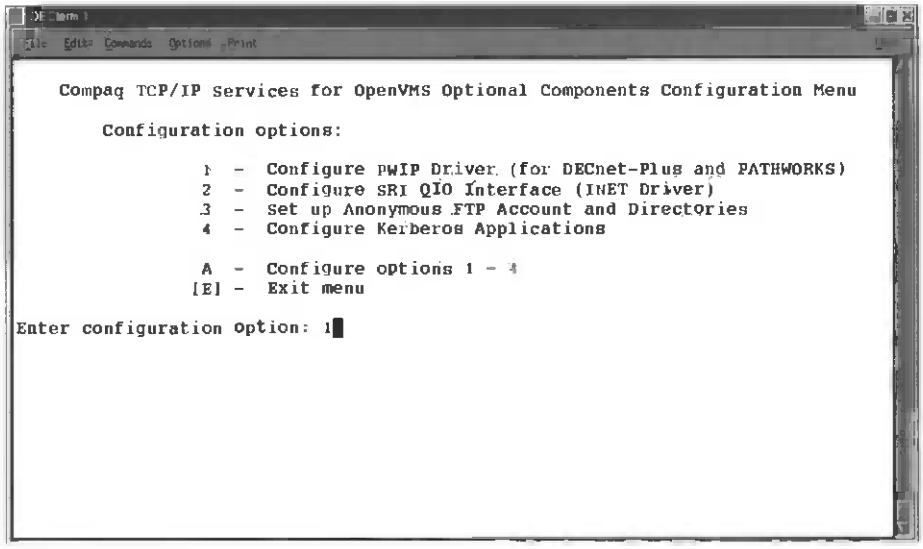


Figure 7-28 Choose menu choice 1 for the PWIP driver.

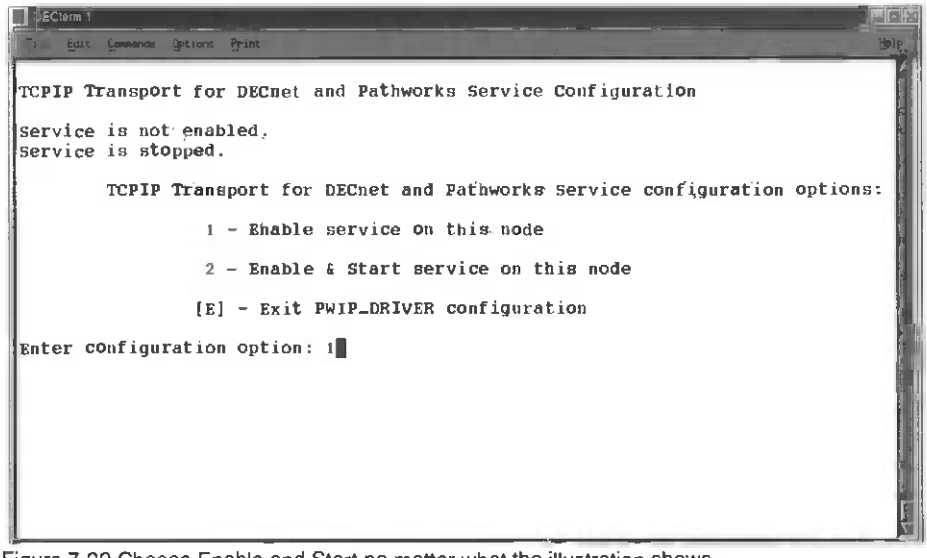


Figure 7-29 Choose Enable and Start no matter what the illustration shows.

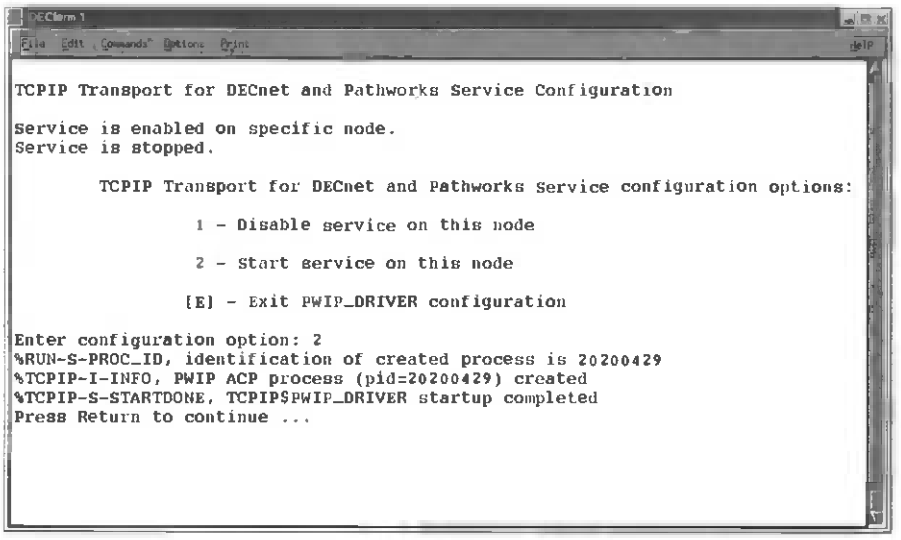


Figure 7-30 You should receive a similar message for successful installation.

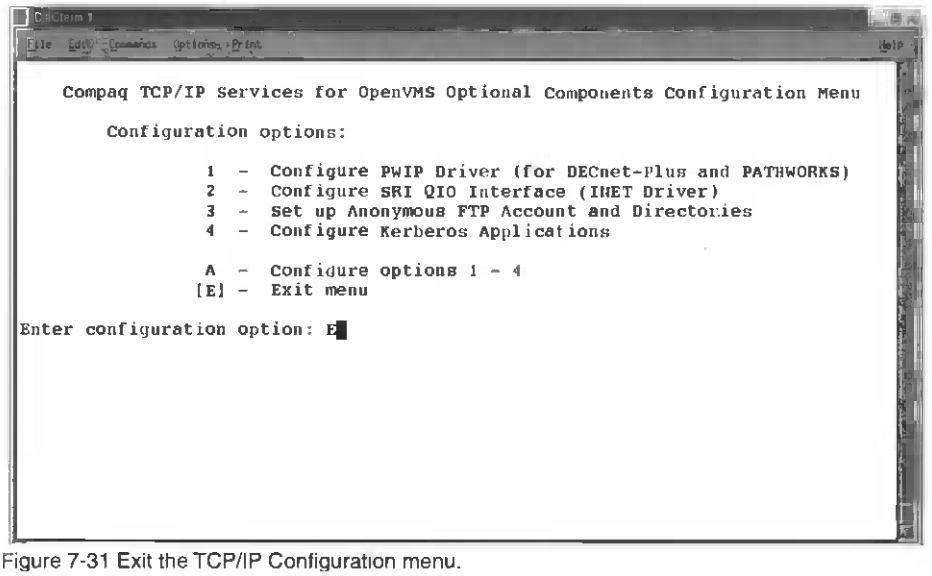


Figure 7-31 Exit the TCP/IP Configuration menu.

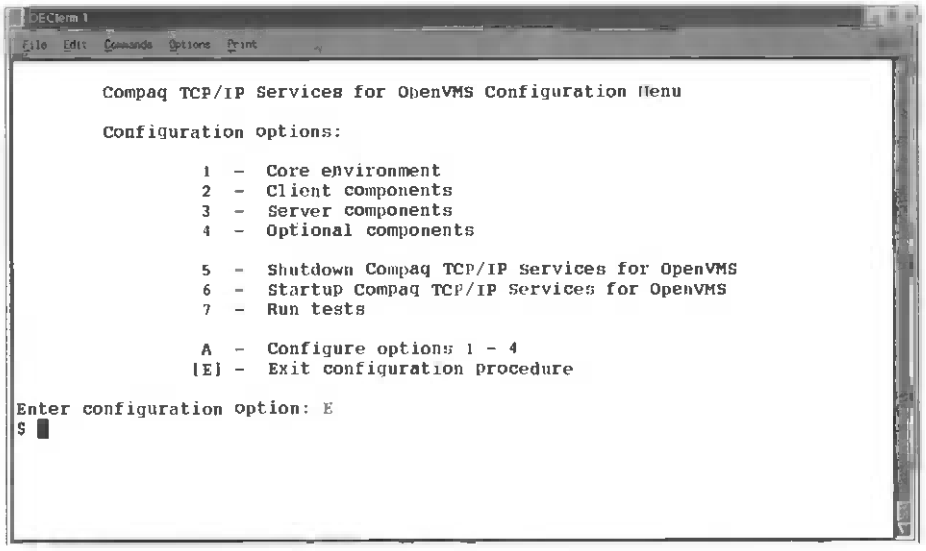


Figure 7-32 Return to the $ prompt.

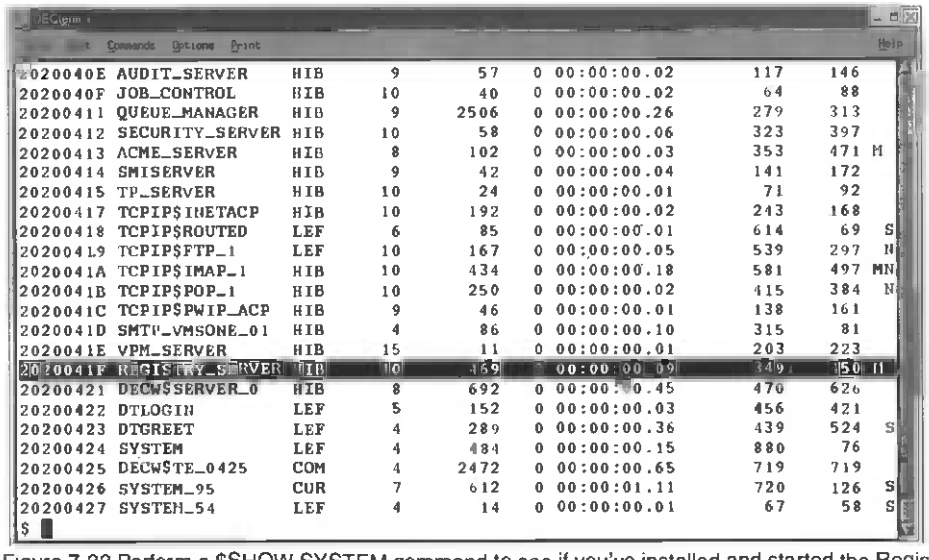


Figure 7-33 Perform a $SHOW SYSTEM command to see if you've installed and started the Registry Server.

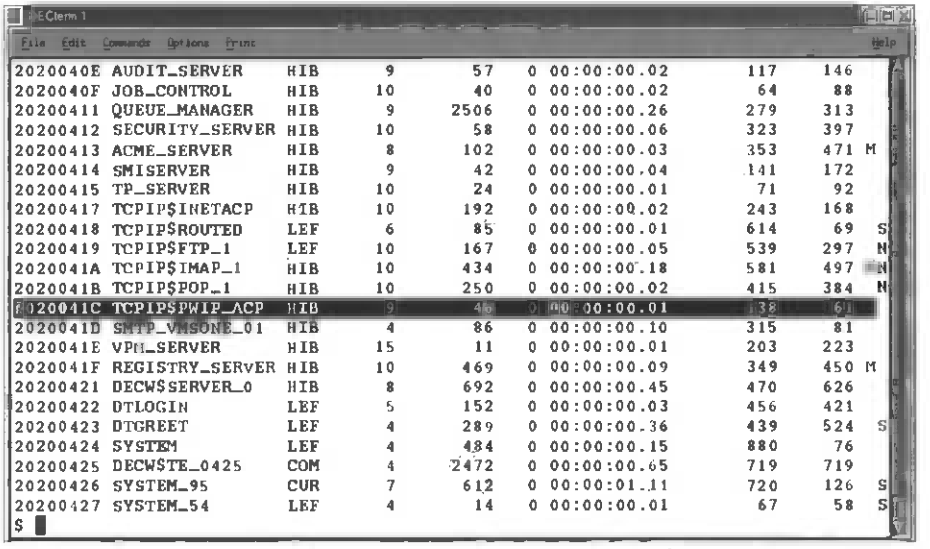


Figure 7-34 Also check to see if you've installed and started the TCPIP$PWIP_ACP process.

I would recommend that you reboot your system before performing the next step of the Advanced Server installation.

Step 3: OpenVMS Advanced Server Install

Figure 7-35 Ok! Now for the main event. Logged in to the SYSTEM ACCOUNT mount the CD-ROM with Advanced Server on it. Then set your current directory to the disk

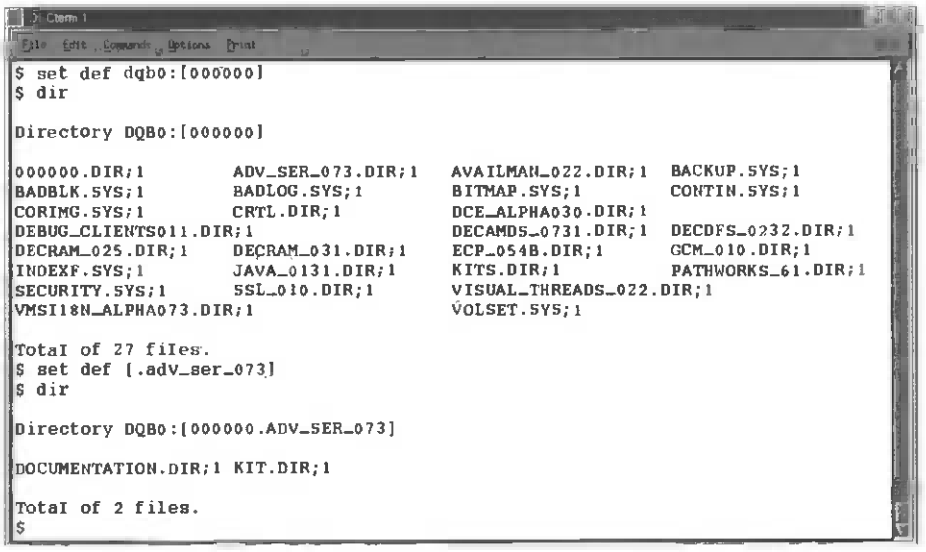


Figure 7-36 Change the directory to the Advanced Server 7.3 kit directory.

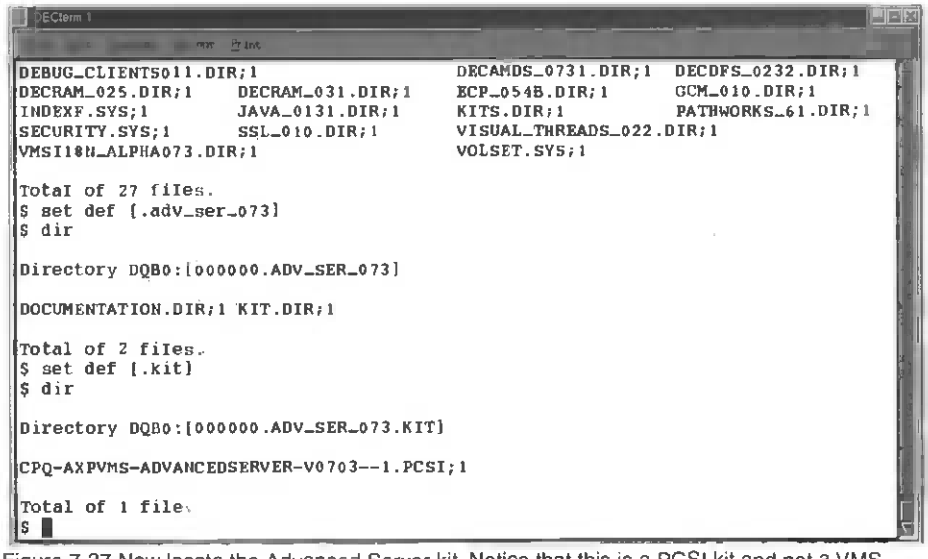


Figure 7-37 Now locate the Advanced Server kit. Notice that this is a PCSI kit and not a VMS install kit.

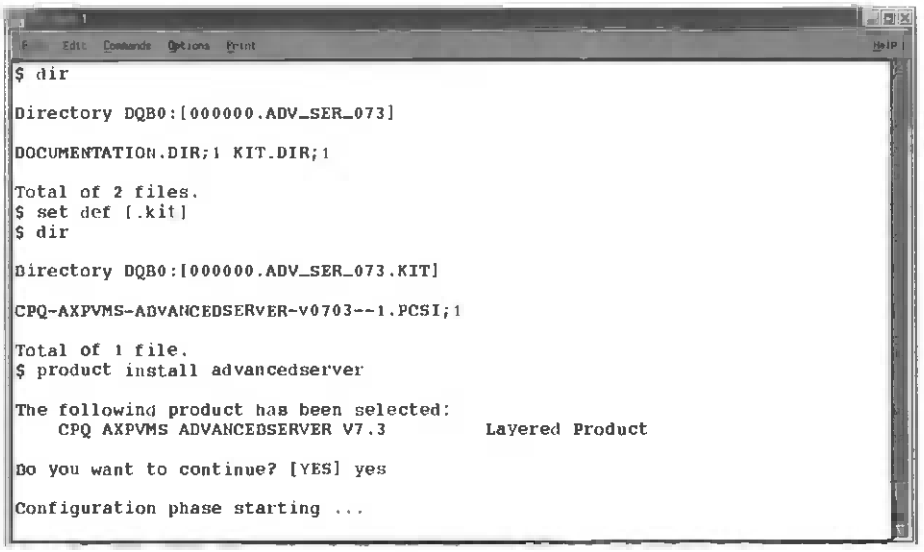


Figure 7-38 Use the product install command to begin the installation.

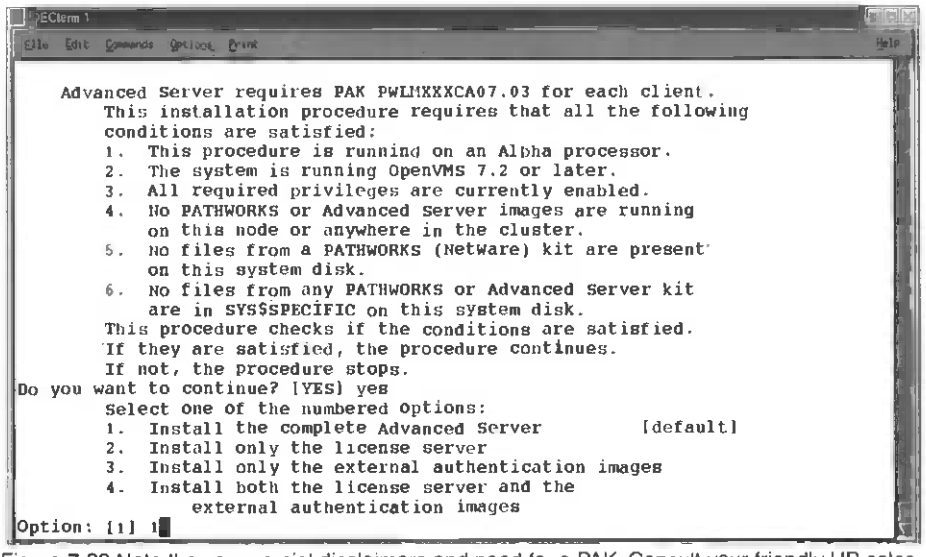


Figure 7-39 Note the commercial disclaimers and need for a PAK. Consult your friendly HP sales rep for more help with this.

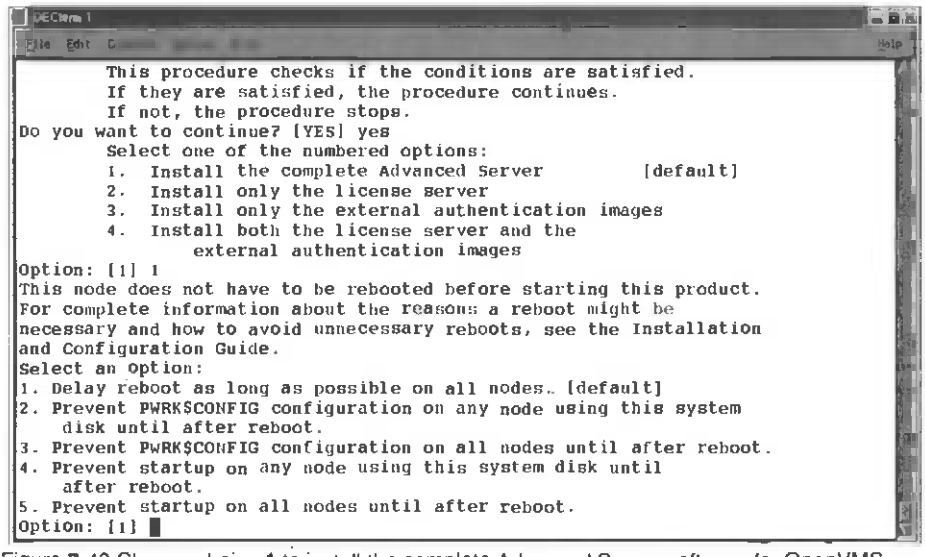


Figure 7-40 Choose choice 1 to install the complete Advanced Server software for OpenVMS.

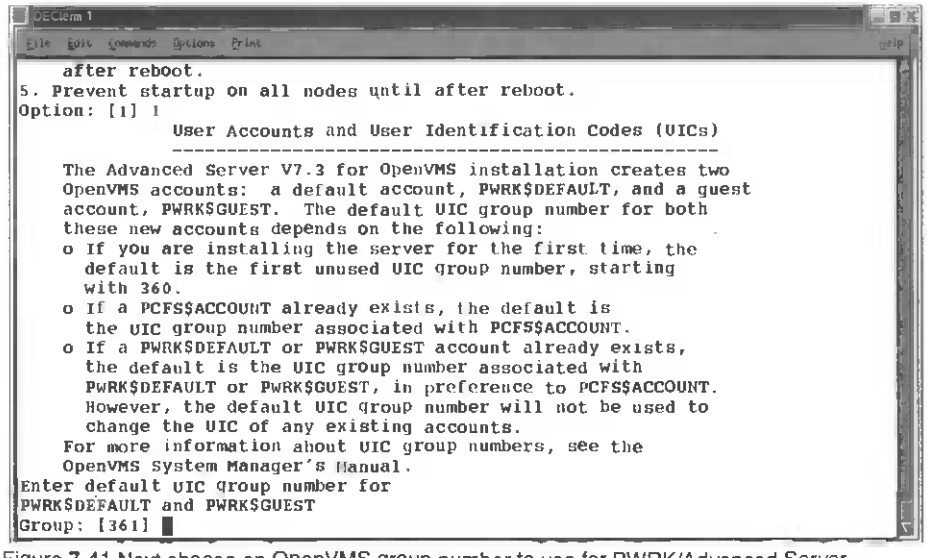


Figure 7-41 Next choose an OpenVMS group number to use for PWRK/Advanced Server accounts! (You might want to use the defaults).

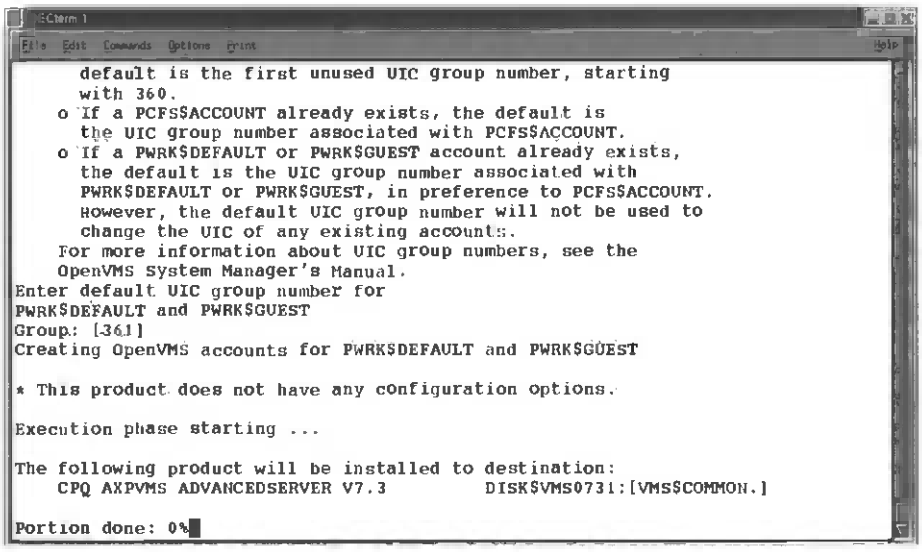


Figure 7-42 Now that all the questions have been asked the actual installation begins!

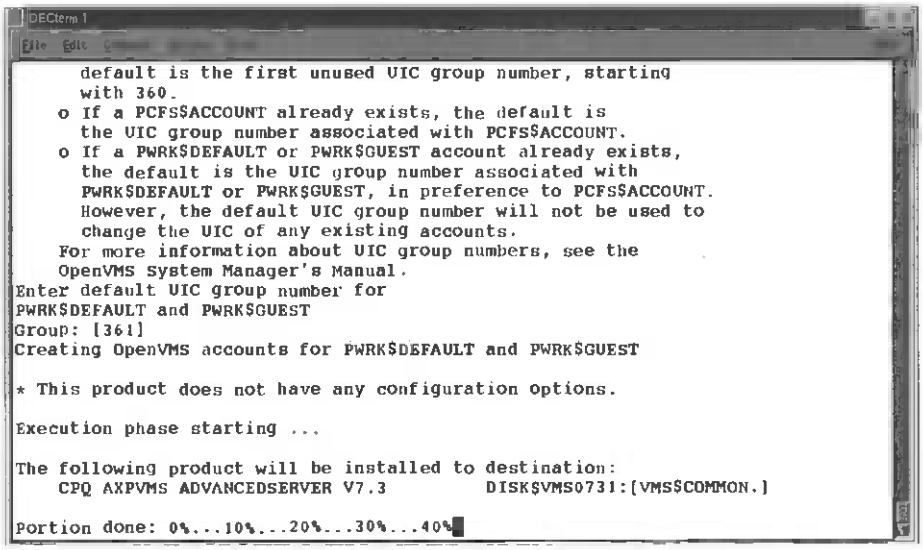


Figure 7-43 Advanced Server installation will proceed until 100 percent of the files have been copied from the install procedure.

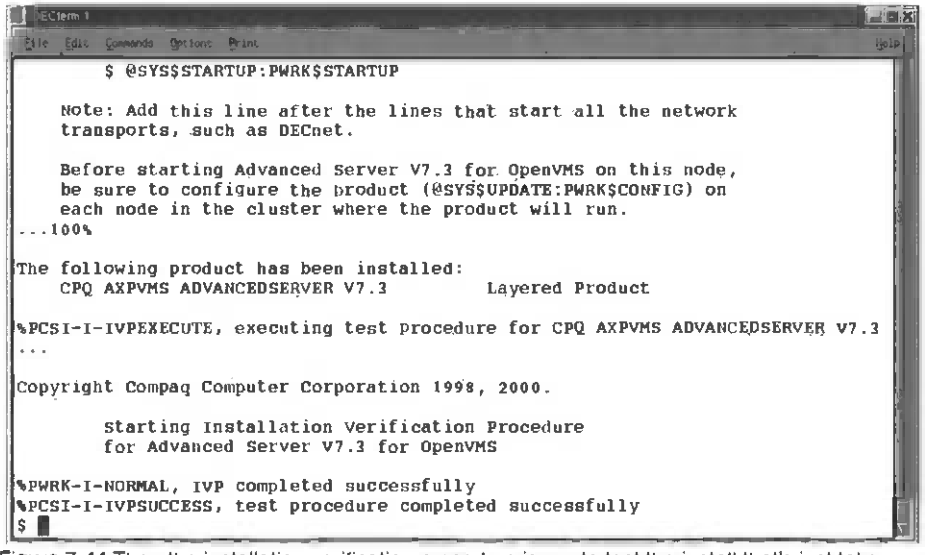


Figure 7-44 Then the installation verification procedure is run to test the install that's just taken place. And you've installed Advanced Server!

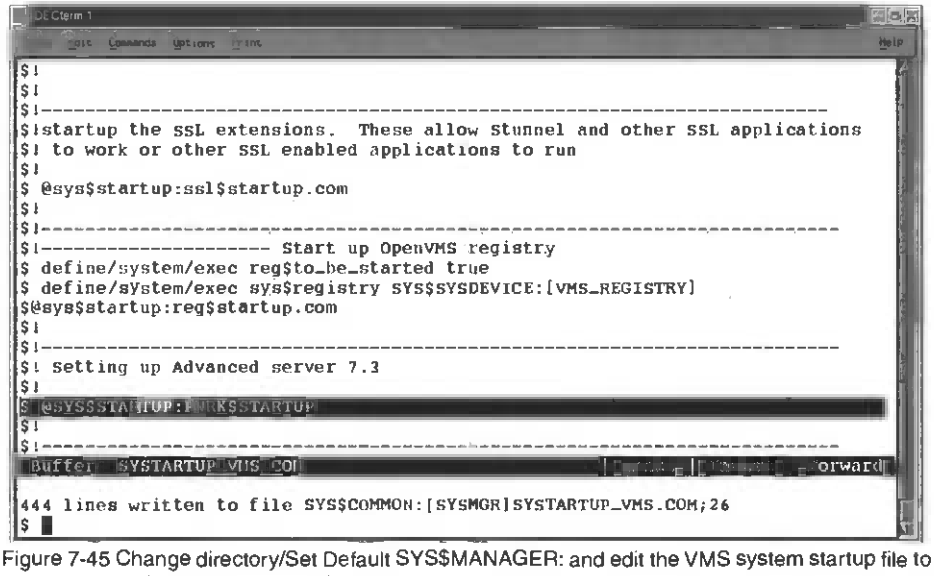


Figure 7-45 Change directory/Set Default SYS$MANAGER: and edit the VMS system startup file to add the @SYS$STARTUP:PWRK$STARUP.COM command so Advanced Server will start up on a reboot.

Now reboot your OpenVMS system and prepare to configure your new Windows Domain Server!

Step 4: Configuring OpenVMS Advanced Server processes

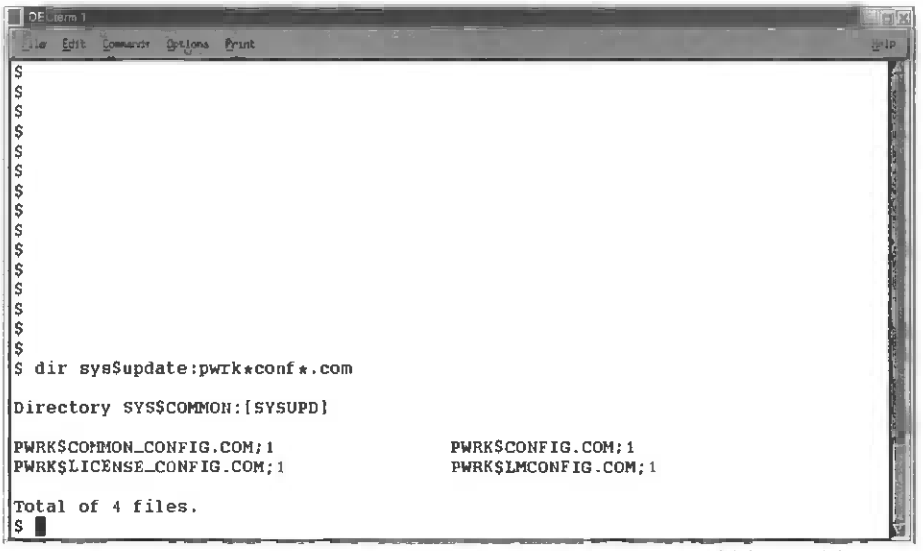


Figure 7-46 After the reboot, log in as SYSTEM again and locate the PWRK$CONFIG.COM file in SYS$UPDATE then run the command file.

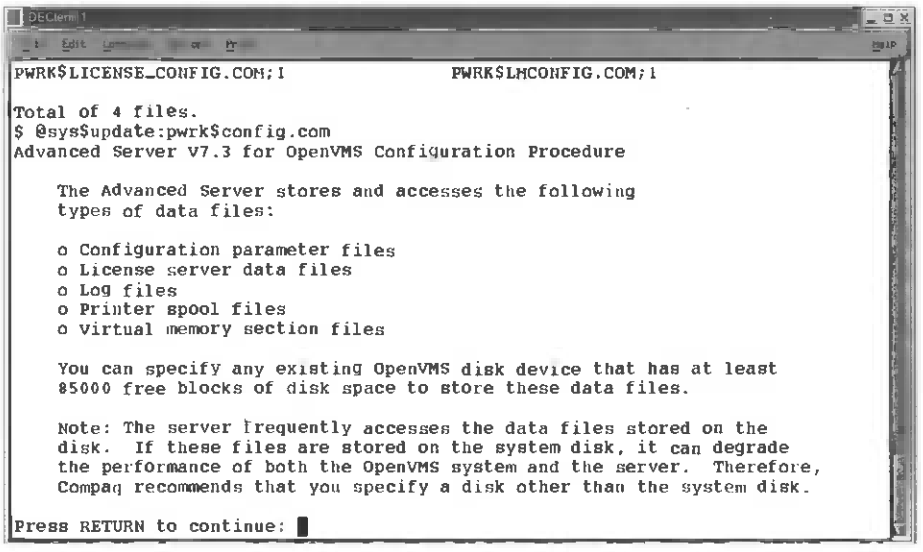


Figure 7-47 Recommendations for free disk space and so on.

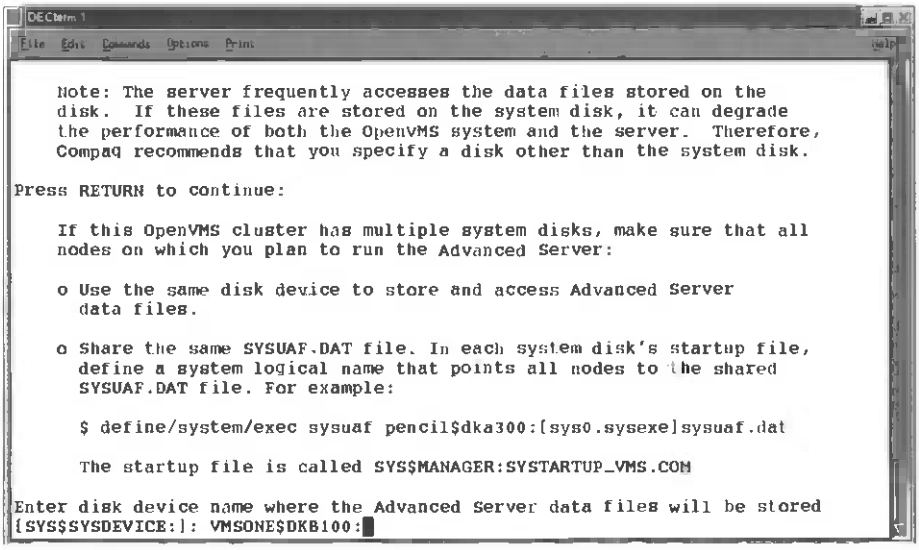


Figure 7-48 For the Advanced Server files, choose a disk that has free space and isn't a system disk but is available clusterwide (to all other cluster members).

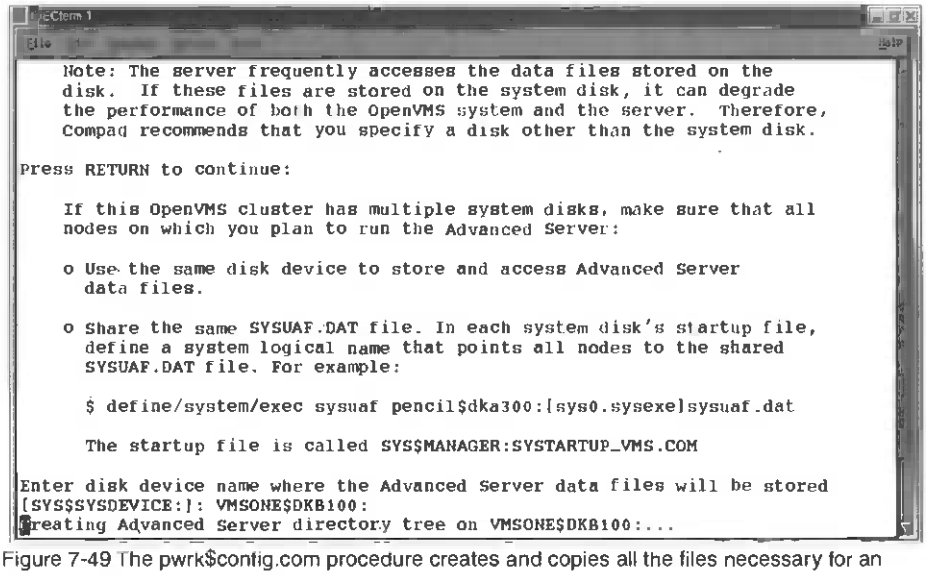


Figure 7-49 The pwrk$config.com procedure creates and copies all the files necessary for an Advance Server instance.

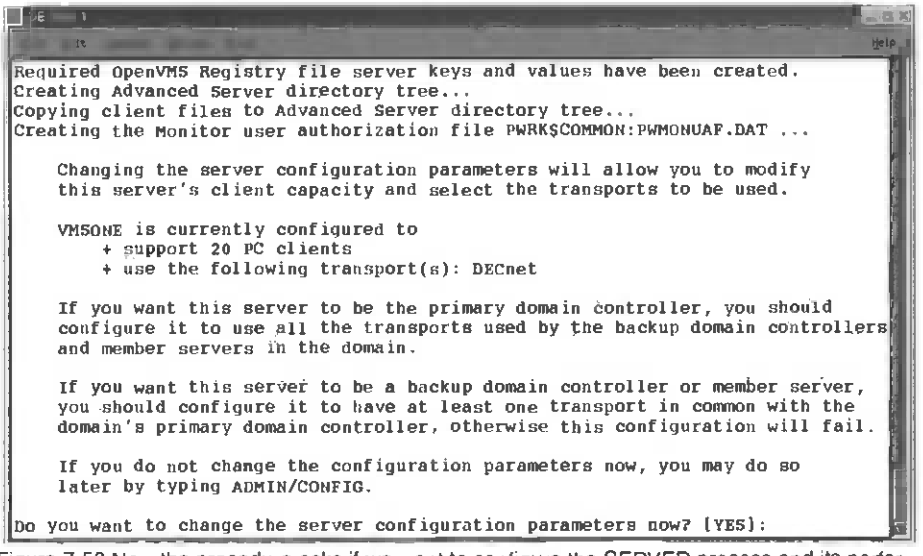


Figure 7-50 Now the procedure asks if we want to configure the SERVER process and its performance characteristics. Answer yes and configure the Advanced Server processes.

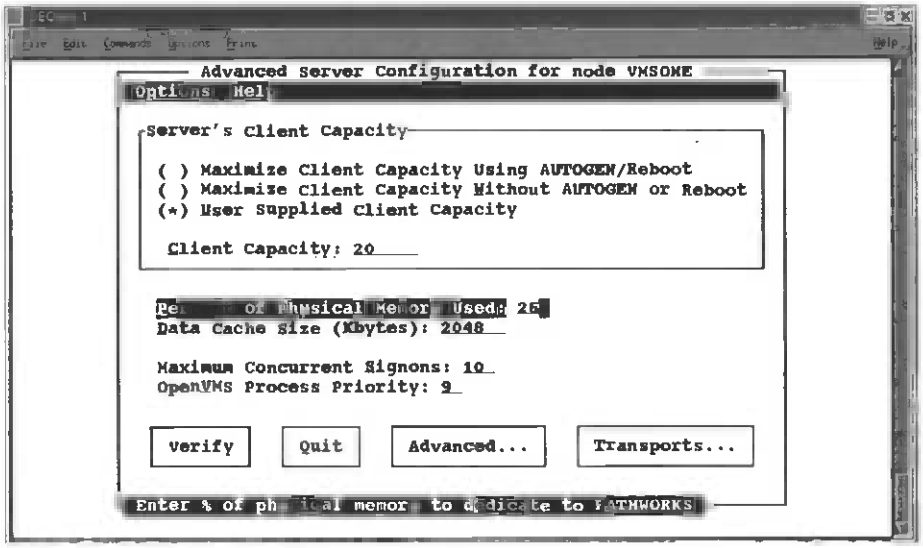


Figure 7-51 You might only want to use 512MB instead of 1.6GB of memory for cache.

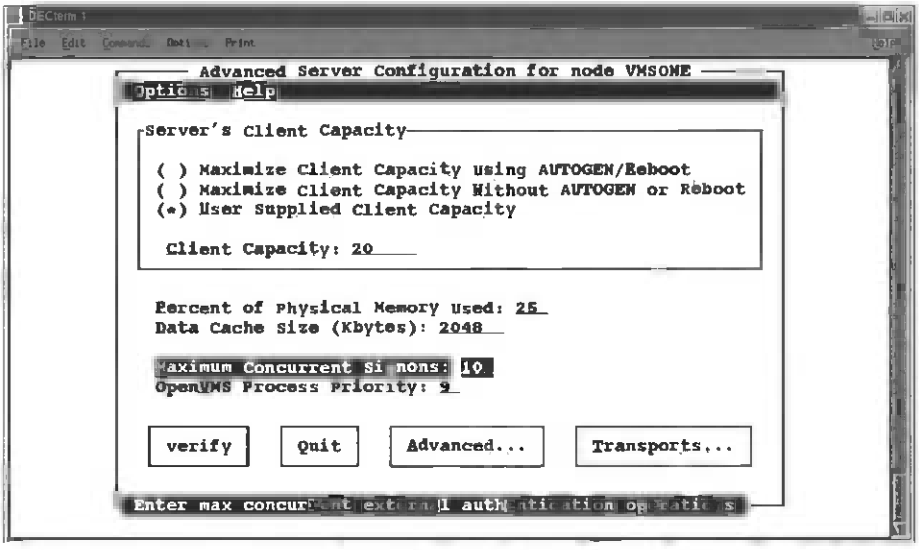


Figure 7-52 For a home network, ten concurrent sign-ons are more than enough. The menu will ask you to confirm the number of users and other choices you should verify first.

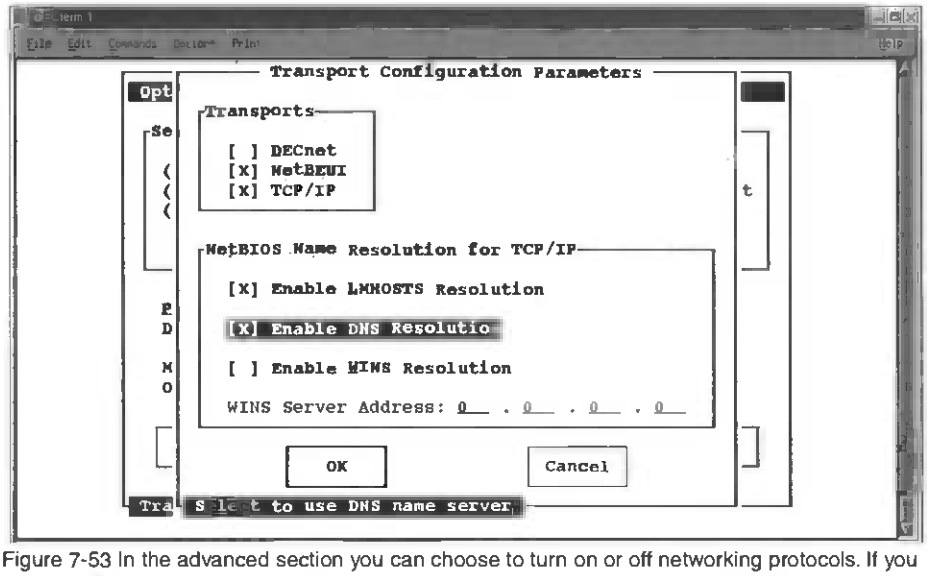


Figure 7-53 In the advanced section you can choose to turn on or off networking protocols. If you are not using a protocol, turn it off! You might also want to enable DNS and LMHOST translations in a TCP/IP network.

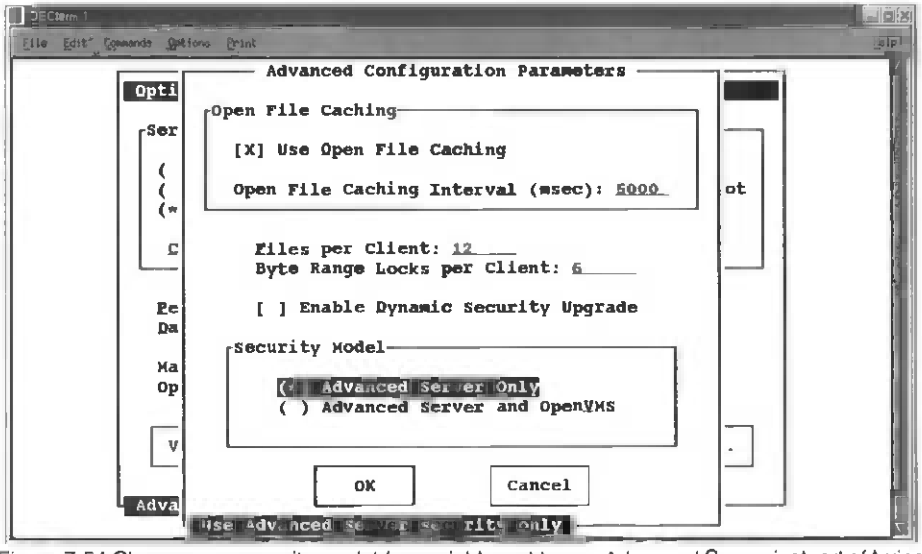


Figure 7-54 Choose your security model (you might want to use Advanced Server instead of trying to map the OpenVMS security to the Windows world.

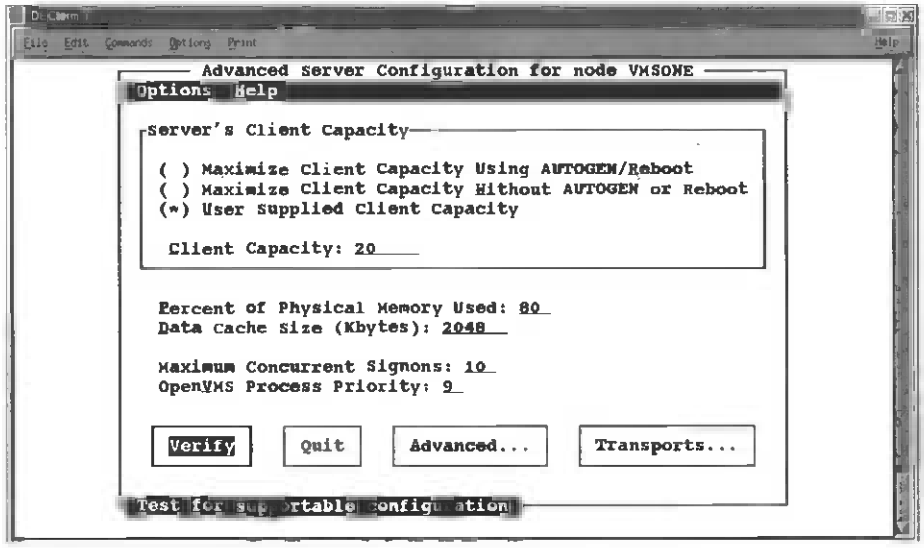


Figure 7-55 Verify the configuration you've just changed.

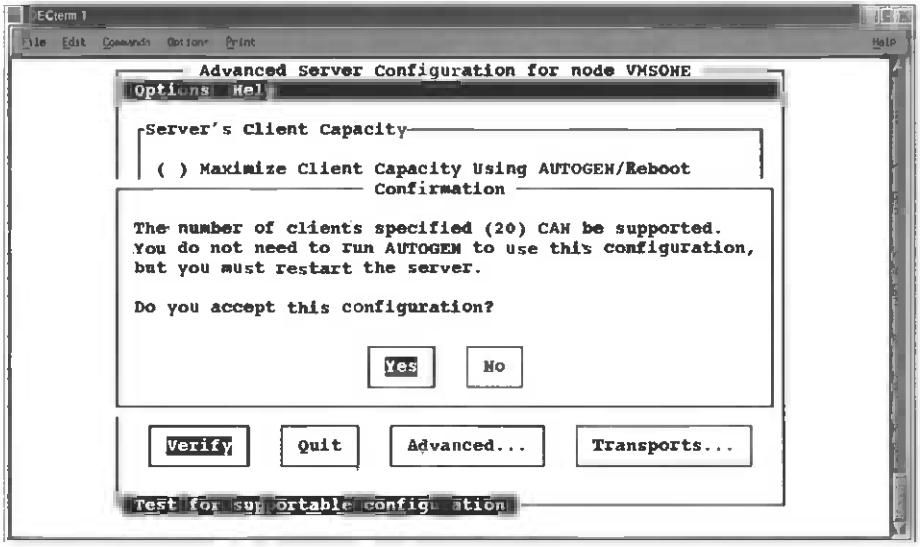


Figure 7-56 Success! The defaults will let you set up a 20-person server (usually), but if you need to go back or tune VMS for more users, you can rerun this portion of the configuration at any time after installation.

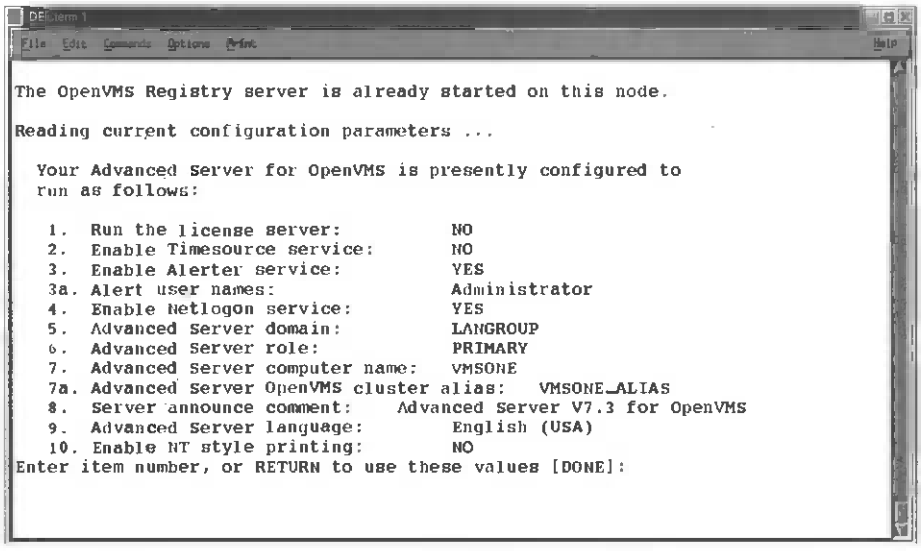


Figure 7-57 This menu shows your definitions for your server and domain. You may need to change some of these for your domain and/or workgroup.

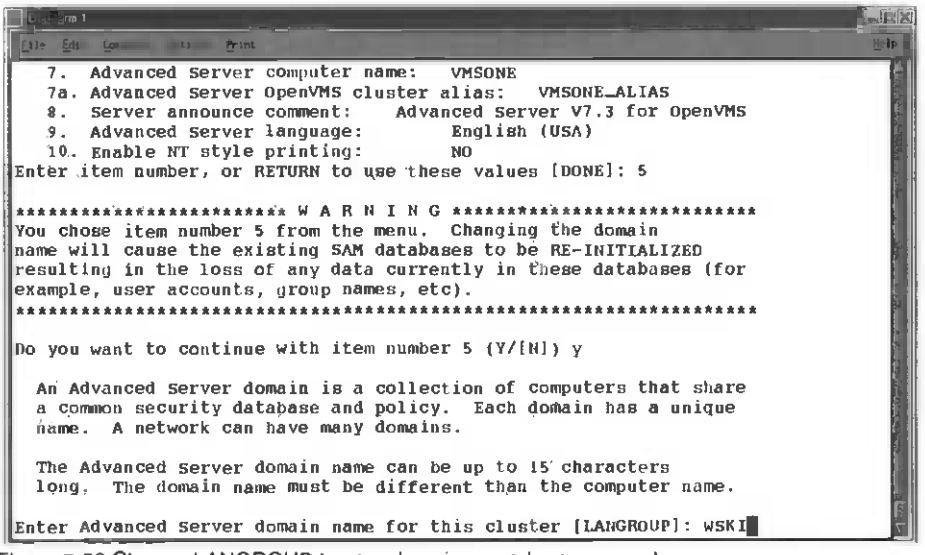


Figure 7-58 Change LANGROUP to your domain or workgroup name!

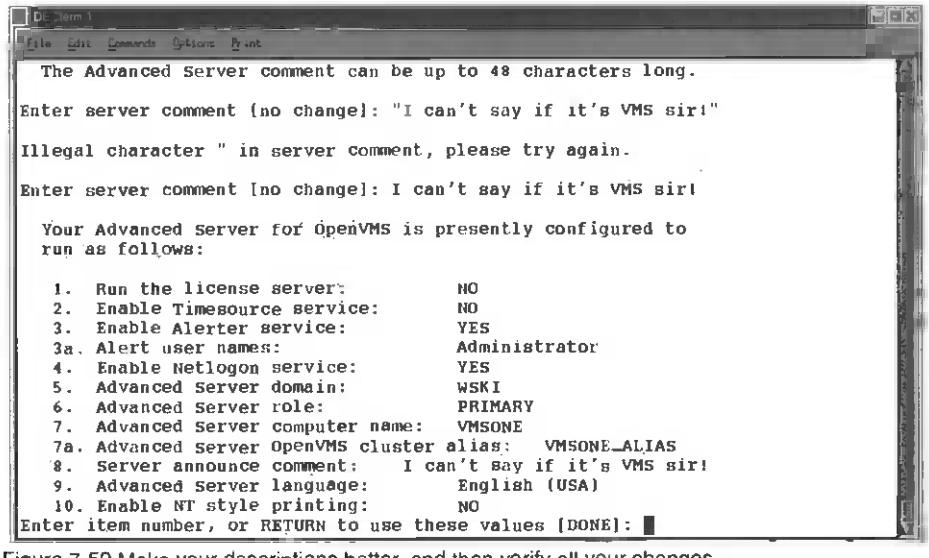


Figure 7-59 Make your descriptions better, and then verify all your changes.

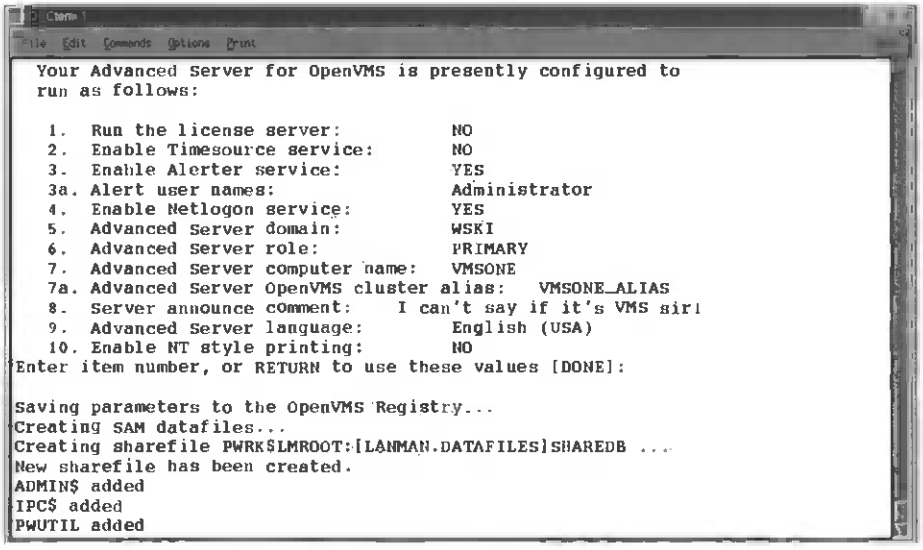


Figure 7-60 When finished, the menu adds common shares and admin access for your new Advanced Server.

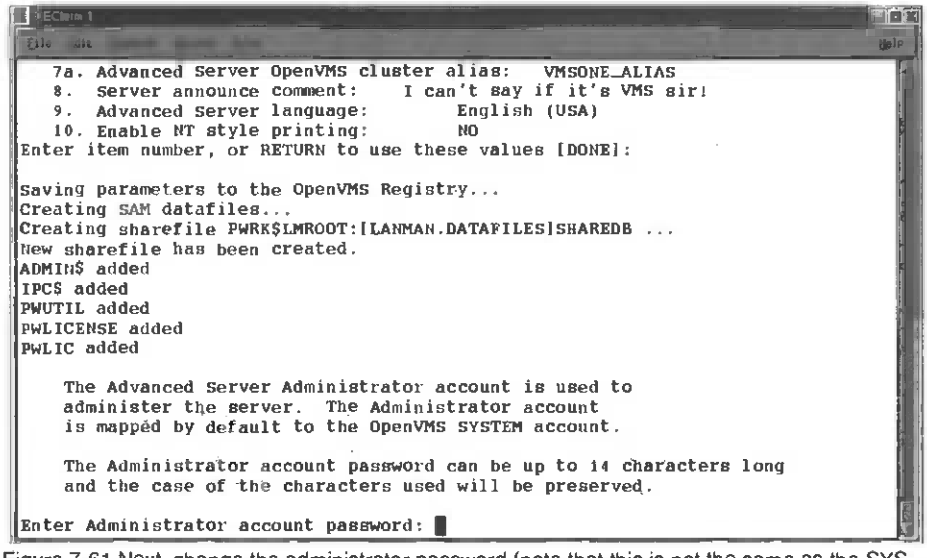


Figure 7-61 Next, change the administrator password (note that this is not the same as the SYSTEM ACCOUNT's password!).

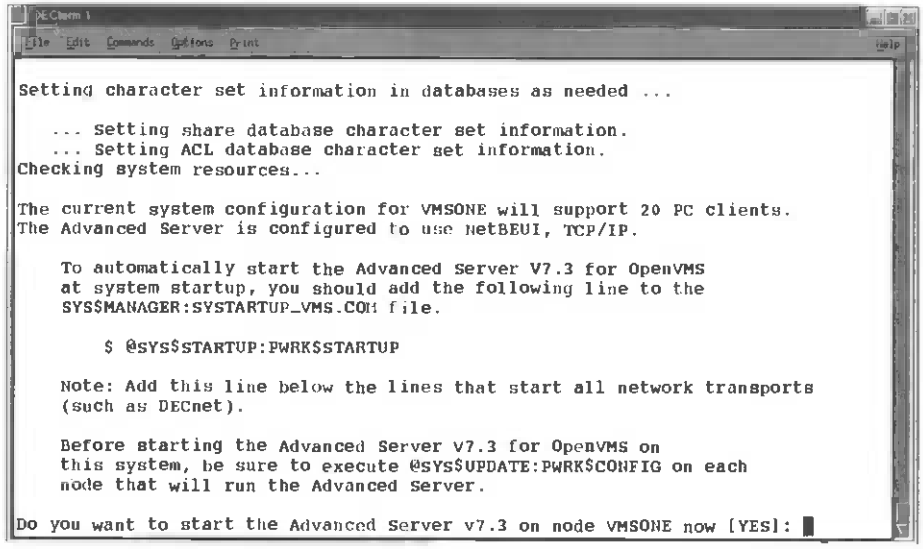


Figure 7-62 Start up all the Advanced Server processes.

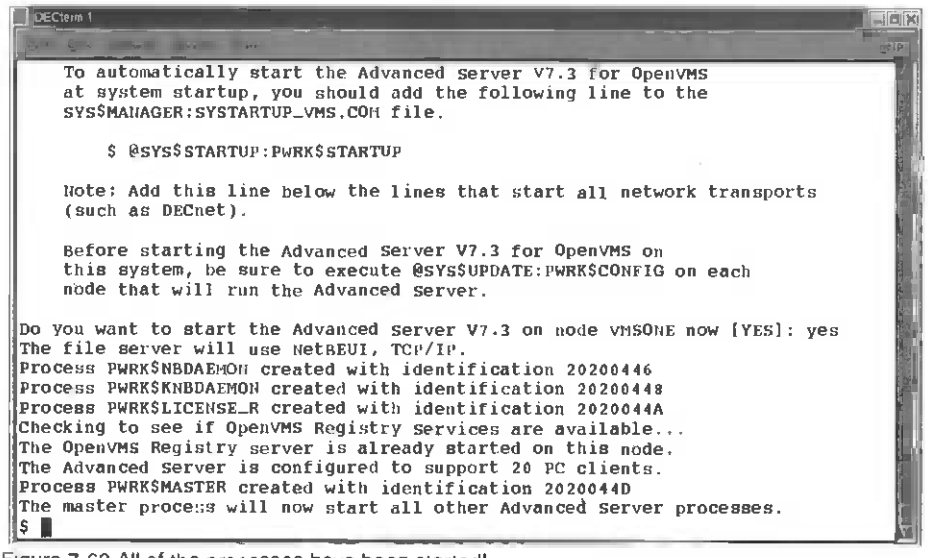
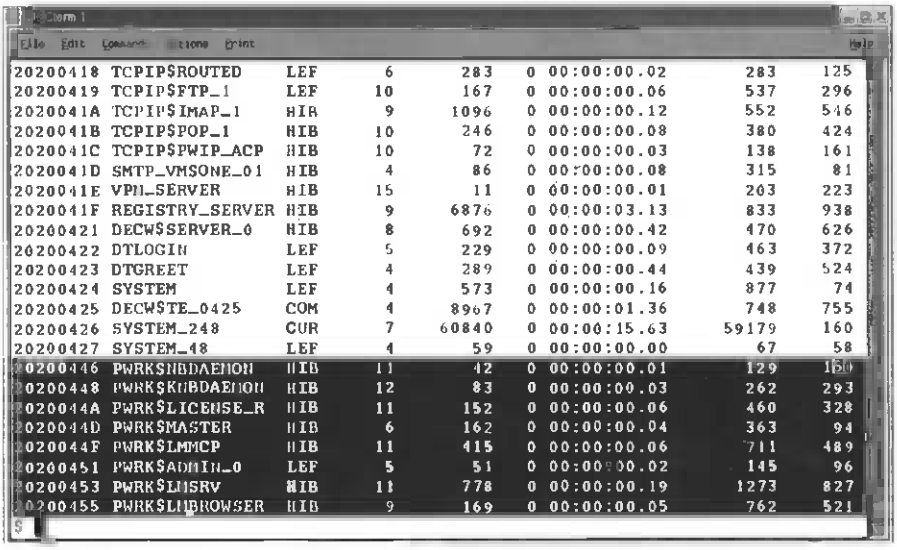


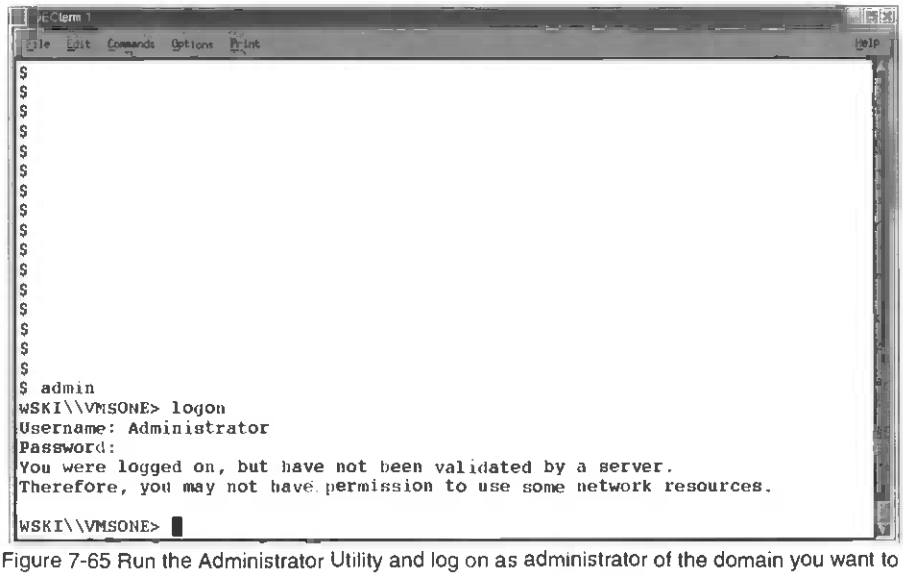
Figure 7-63 All of the processes have been started!



```
20200418 TCPIP$ROUTED          LEF    6      283    0 00:00:00.02      283    125
20200419 TCPIP$FTP_1           LEF   10      167    0 00:00:00.06      537    296
2020041A TCPIP$IMAP_1          HIB    9     1096    0 00:00:00.12      552    546
2020041B TCPIP$POP_1           HIB   10      246    0 00:00:00.08      380    424
2020041C TCPIP$PWIP_ACP        HIB   10       72    0 00:00:00.03      138    161
2020041D SMTP_VMSONE_01        HIB    4       86    0 00:00:00.08      315     81
2020041E VPN_SERVER            HIB   15       11    0 00:00:00.01      203    223
2020041F REGISTRY_SERVER       HIB    9     6876    0 00:00:03.13      833    938
20200421 DECW$SERVER_0         HIB    8      692    0 00:00:00.42      470    626
20200422 DTLOGIN               LEF    5      229    0 00:00:00.09      463    372
20200423 DTGREET               LEF    4      289    0 00:00:00.44      439    524
20200424 SYSTEM                LEF    4      573    0 00:00:00.16      877     74
20200425 DECW$TE_0425          COM    4     8967    0 00:00:01.36      748    755
20200426 SYSTEM_248            CUR    7    60840    0 00:00:15.63    59179    160
20200427 SYSTEM_48             LEF    4       59    0 00:00:00.00       67     58
20200446 PWRK$NBDAEMON         HIB   11       42    0 00:00:00.01      129    160
20200448 PWRK$KNBDAEMON        HIB   12       83    0 00:00:00.03      262    293
2020044A PWRK$LICENSE_R        HIB   11      152    0 00:00:00.06      460    328
2020044D PWRK$MASTER           HIB    6      162    0 00:00:00.04      363     94
2020044F PWRK$LMMCP            HIB   11      415    0 00:00:00.06      711    489
20200451 PWRK$ADMIN_0          LEF    5       51    0 00:00:00.02      145     96
20200453 PWRK$LMSRV            HIB   11      778    0 00:00:00.19     1273    827
20200455 PWRK$LMBROWSER        HIB    9      169    0 00:00:00.05      762    521
$
```

Figure 7-64 Now check with a $SHOW SYSTEM and verify that the Advanced Server processes are running!

Step 5: Configure OpenVMS Advanced Server as a Domain Controller



```
$
$
$
$
$
$
$
$
$
$
$
$
$
$
$
$ admin
WSKI\\VMSONE> logon
Username: Administrator
Password:
You were logged on, but have not been validated by a server.
Therefore, you may not have permission to use some network resources.

WSKI\\VMSONE>
```

Figure 7-65 Run the Administrator Utility and log on as administrator of the domain you want to work on!

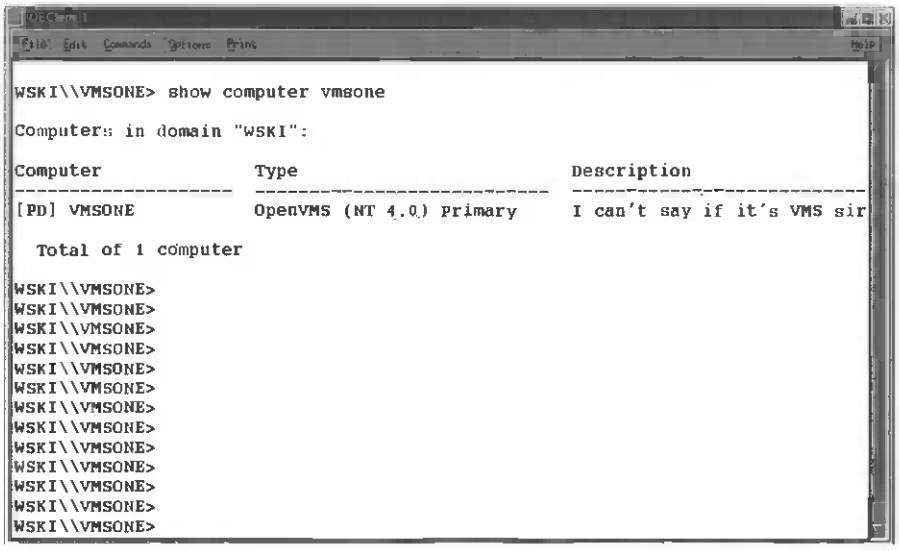


Figure 7-66 Use the show computer command and you'll find only this server; it hasn't been configured yet!

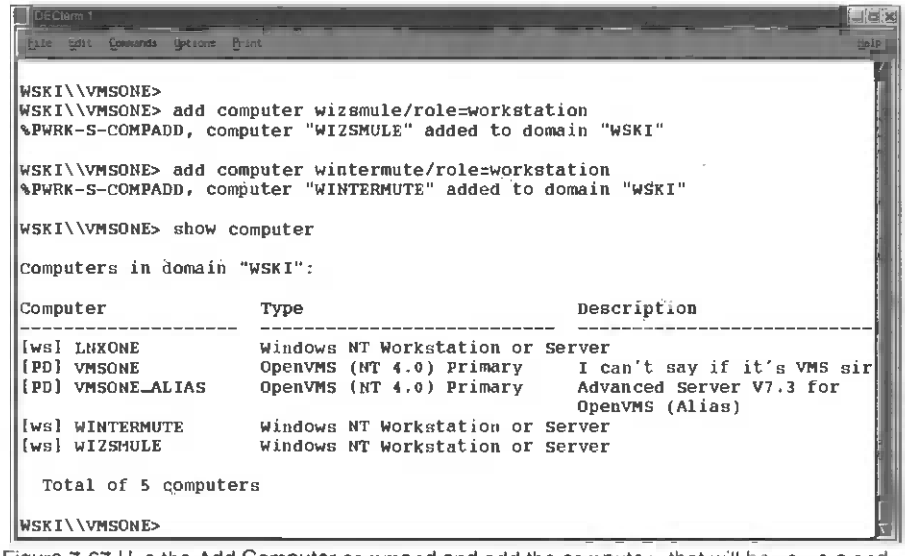


Figure 7-67 Use the Add Computer command and add the computers that will be servers and workstations in your domain.

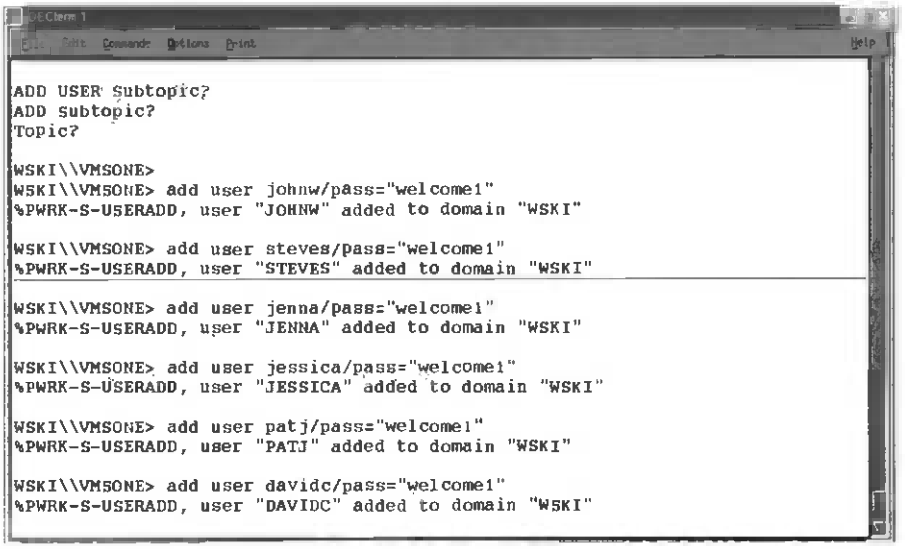


Figure 7-68 Next, add users who can log on to the domain (and its shares) from workstations and other computers!

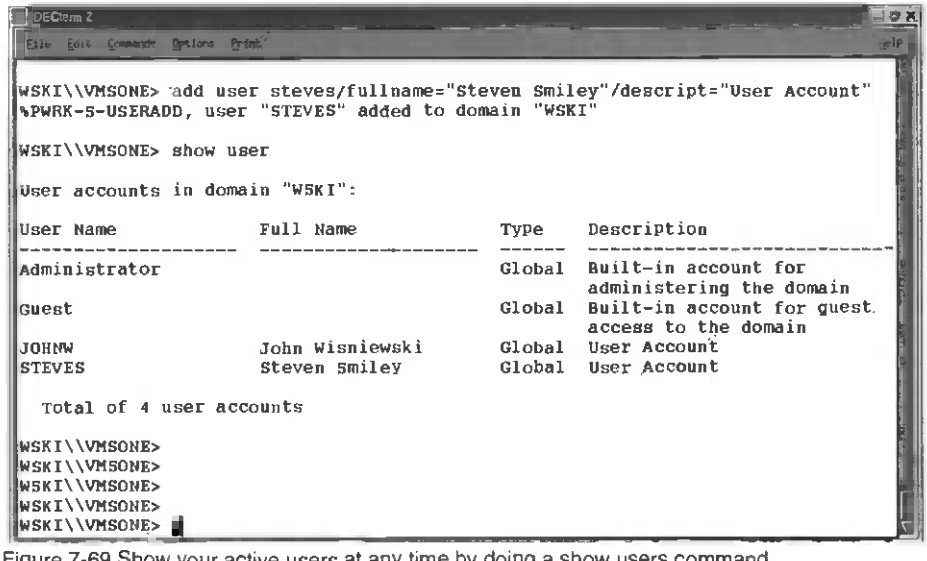


Figure 7-69 Show your active users at any time by doing a show users command.

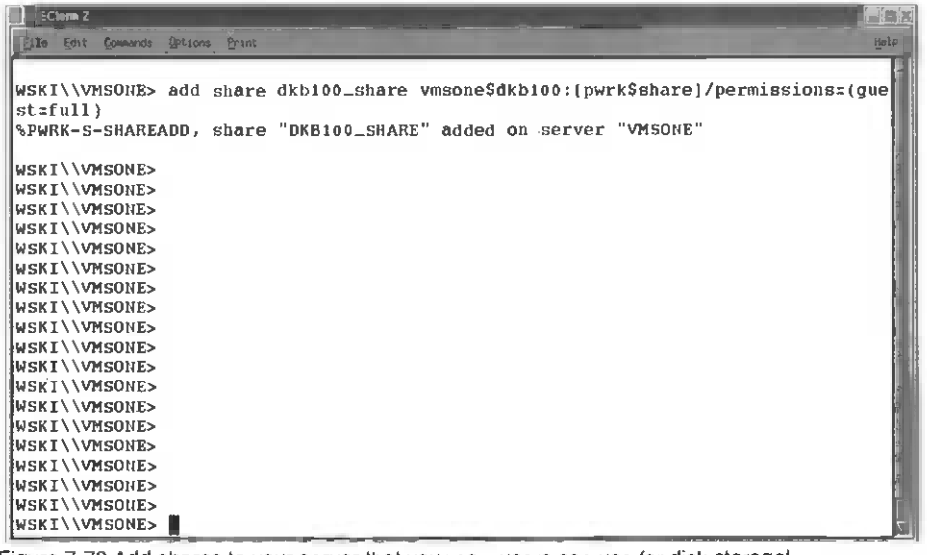


Figure 7-70 Add shares to your server that your new users can use for disk storage!

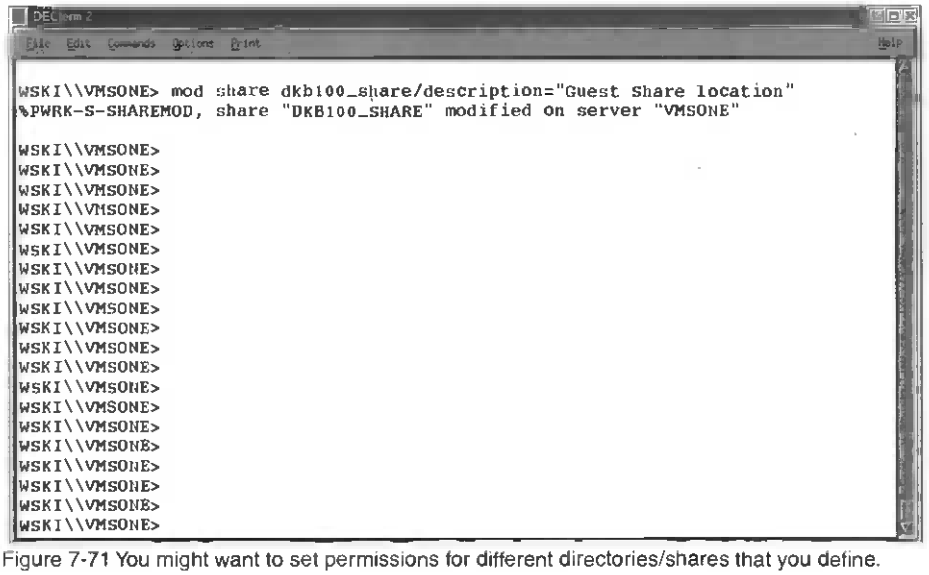


Figure 7-71 You might want to set permissions for different directories/shares that you define.

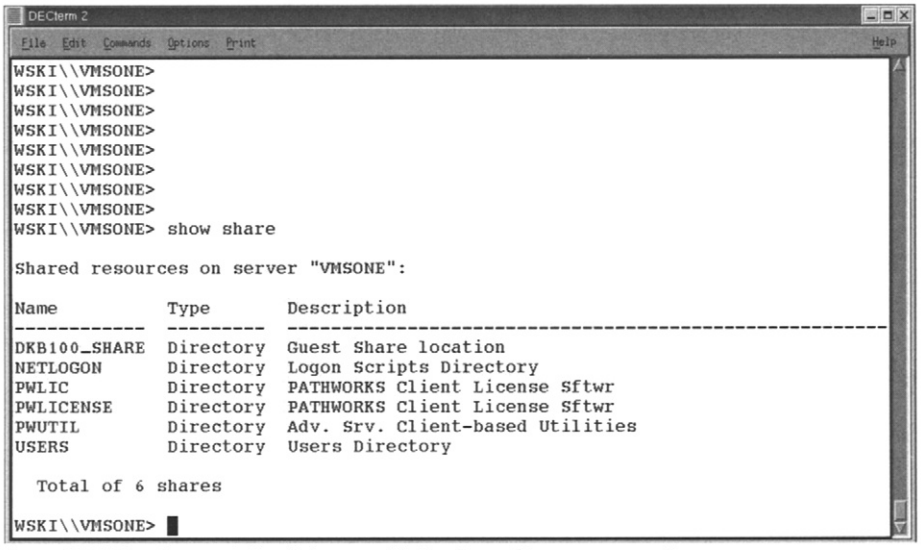


Figure 7-72 Check your defined shares with the Show Share command!

Congratulations! You've installed Advanced Server on OpenVMS and configured computers, users, and shares for network use!

Chapter 8 — Apache VMS and Linux Style

You Never Really Forget your First Web Server

You never really forget your first Web server. It was September, the leaves were just starting to turn in New England, and I was at a class about Web servers, Internet e-business, and Web mastering. The week was much too short and the teachers much too quick with their information, because there was just so much to cover.

In 1995 and the hot tool of choice was, of course, Netscape (some might say there was a rumor of something called Front Page from the Northwest). The browser wars were yet to be fought, but posturing and rumblings of the battles to come were in the air. Every OS worth its salt was laying claim to being a Web server.

I flew home from Boston on Friday with a full laptop disk of VMS software, PC Web tools, and my most precious cargo—the thing that would proudly put my AlphaServer on 2000 and share with the internal network, was Netscape Commerce Server. It was to be my first Web server. SSL, user directory support, and fast well—at least able to do 10 to 20 simultaneous Web requests. I was pumped about the prospects of building a Web server.

I got home and created user/Web accounts for every CD-ROM that was on my Infotowers (I think I had 21 CD-ROM drives back then) and served all the OpenVMS, PC, and then OpenSource CD-ROMs I had to offer to the internal network. It was a glorious few days serving out through HTTP and HTTPS tools, disks files, and even PowerPoint slides. A complete DECUS library of graphics, history files for DEC CD-ROMs going back to the 1970s.

I remember going back to my cubical the morning when all the Web data was up and just walking the pages I had added/built/HTML'd with my own hands. Later I realized that having raw HTML code on my workstation was even better then having a spreadsheet up when my boss Dennis Bashore did a little "management by walking around" in those days!

I was just so proud when the main Web page hit 100 hits later that month—and then quadrupled that number less than a week after that as the DEC file grabbers religiously copied my few paltry gigabytes to their every increasing plunder. Hint number 8972: You know you've been plundering file sites too long when you start finding YOUR directory structures on other sites.

Web master flash forward with Apache

I'll never forget that first Web server, and today we take Web server tools for granted (and for free), such as Apache from Open Source. Some are quick to say that Linux's first killer/breakthrough application was Apache moving Linux into many people's minds as the very economical Web server. At one point almost all the Linux boxes being brought up were built for the primary purpose of sharing World Wide Web pages.

Today 65 percent or more of all Web servers are Apache based. RedHat ships Apache with Linux now (although today you must at least do a minimal configuration, start the Apache Server, and set it for startup for autostartup upon reboot). OpenVMS engineering also ships an Apache-based Web server that's been tested by OpenVMS engineering for commercial support and performance.

Both Web servers are available on the Web at:

```
http://www.apache.org
http://www.openvms.compaq.com
```

Today Apache has a lot to recommend it as the dominant Web server for Linux and OpenVMS systems. It's easy to install, performance is great, it's Open Source (which makes licensing much easier, and all those new Open Source features just seem to show up in the code base! And because they are the same sources, training to use one gives you a feel for the directory structure, configuration files, and other set values that are common with Apache running on a variety of platforms.

While Web masters over the years have had many choices, the hardest part of their jobs remains much as it was ten years ago: finding enough content to satisfy the most hungry user. Which is the better server for Apache Linux or OpenVMS? The answer depends on which applications need to have Apache Web interfaces? Old crusty FORTRAN or COBOL apps that have been running for 20+ years can be given a new polished interface with Apache and the World Wide Web. Databases, user programs, and even DCL can be Webified with Apache on OpenVMS. Linux systems can give every desktop or server an easy, standard, secure Web page serving tool that has cutting-edge features and e-business storefronts.

Where do you start? Just begin by setting up servers on your systems and joining the WWW!

Linux Apache Web server Installation

Figures 8-1 through 8-24 illustrate the Linux Apache Web server installation.

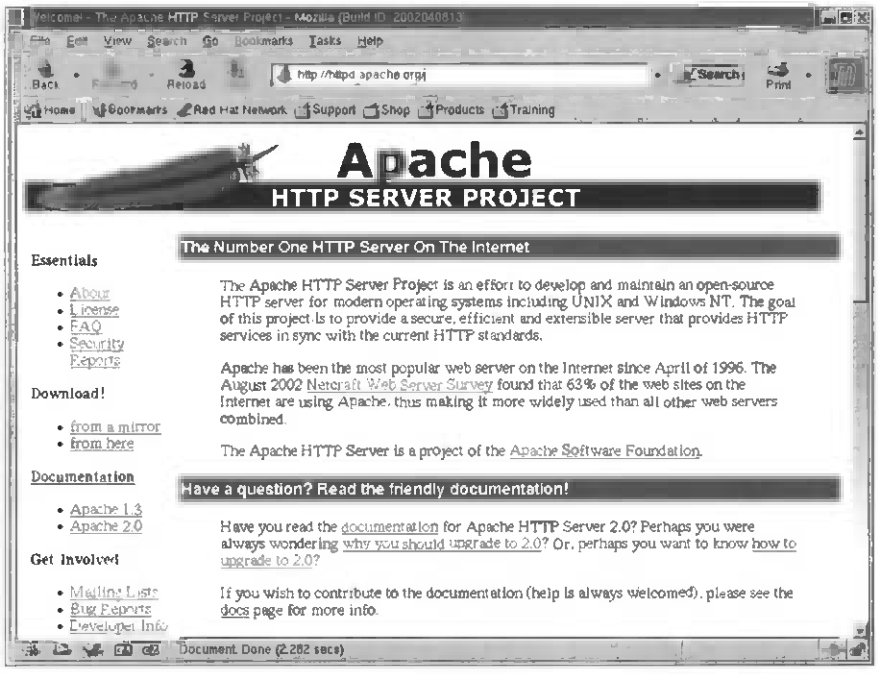


Figure 8-1 First, download Apache binaries from the Apache Web page.

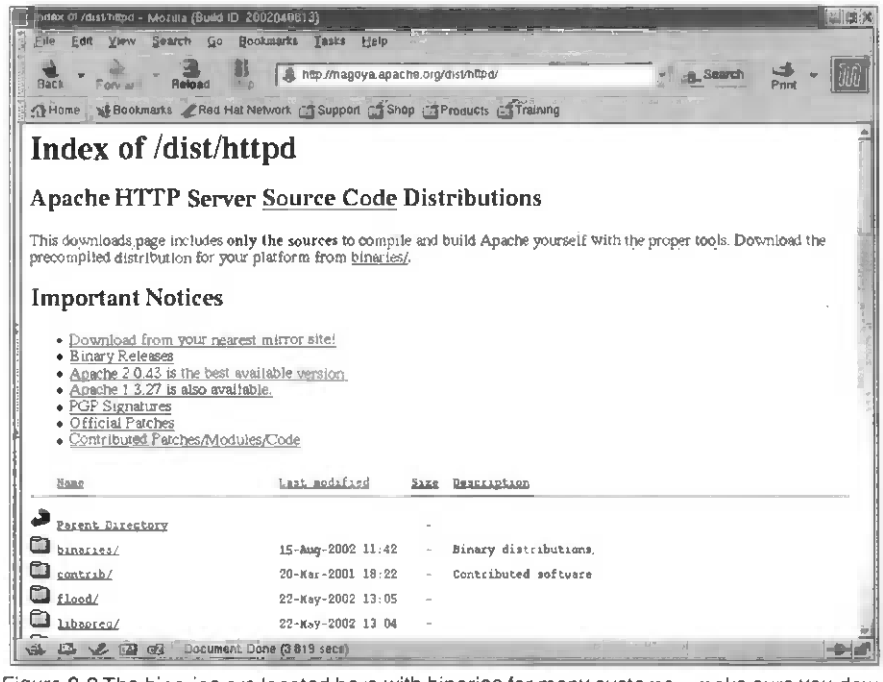


Figure 8-2 The binaries are located here with binaries for many systems—make sure you download the Linux one.

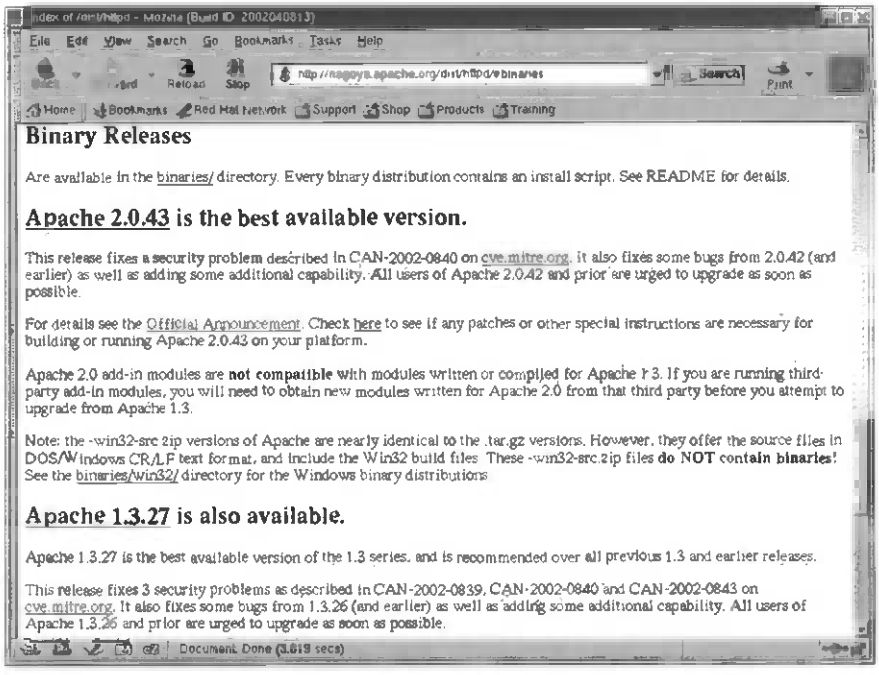


Figure 8-3 The latest version for our installation.

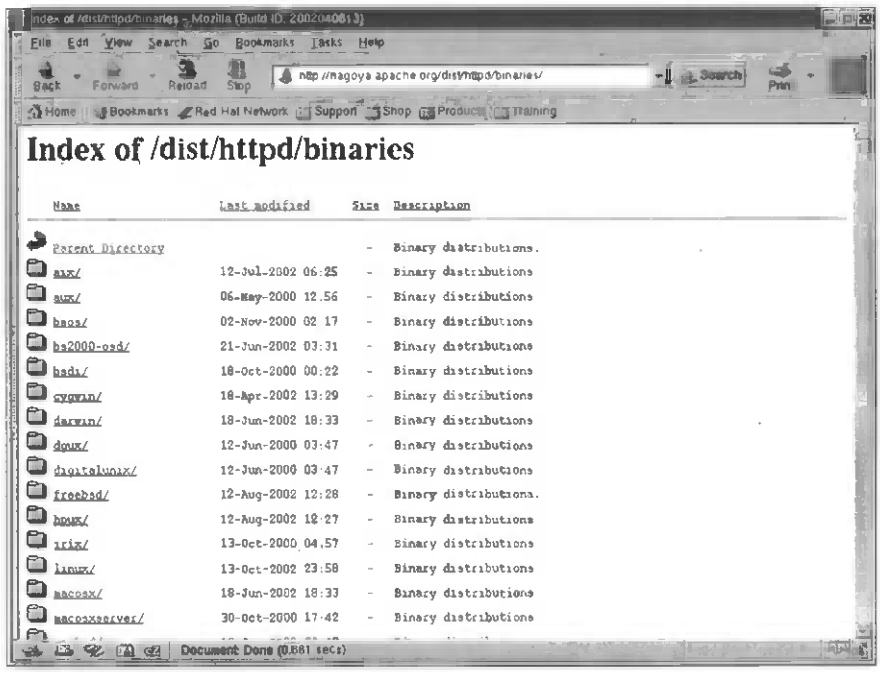


Figure 8-4 Look for the Linux directory.

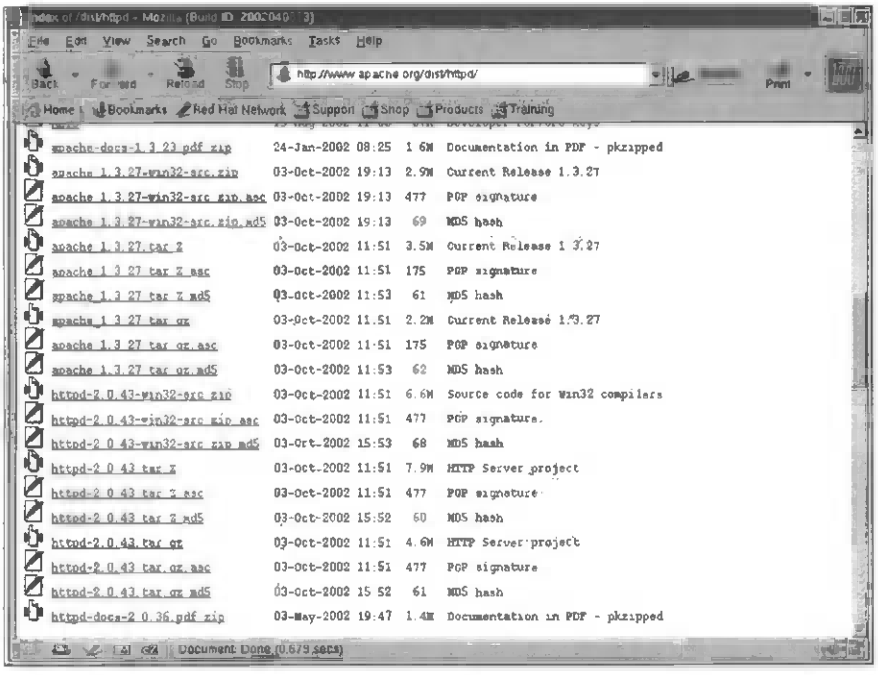


Figure 8-5 Then chose the latest zipped file for download.

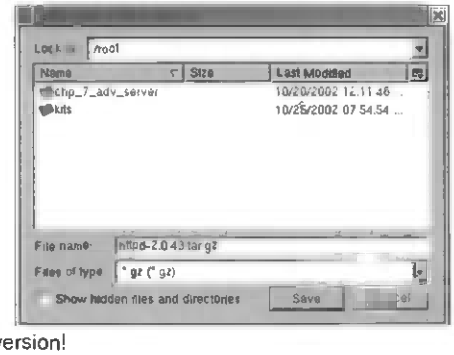


Figure 8-6 Such as this version!

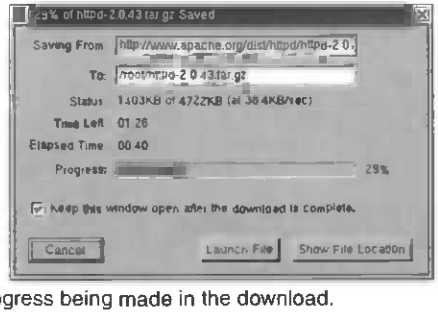


Figure 8-7 Watch the progress being made in the download.

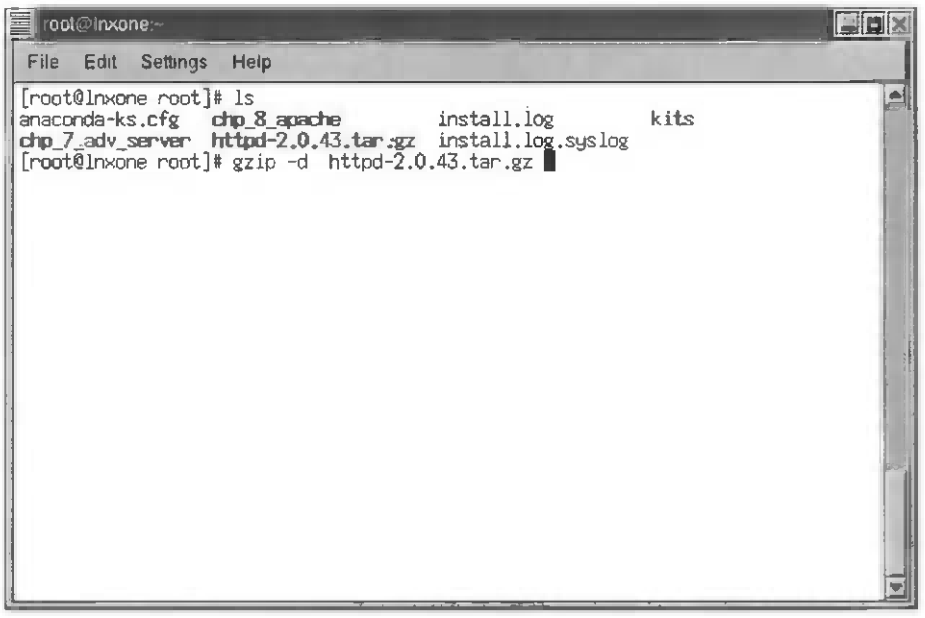


Figure 8-8 Then open a Linux terminal window in the /root directory (which is where you should have downloaded your Apache kit). Unzip the file to it's tar file component.

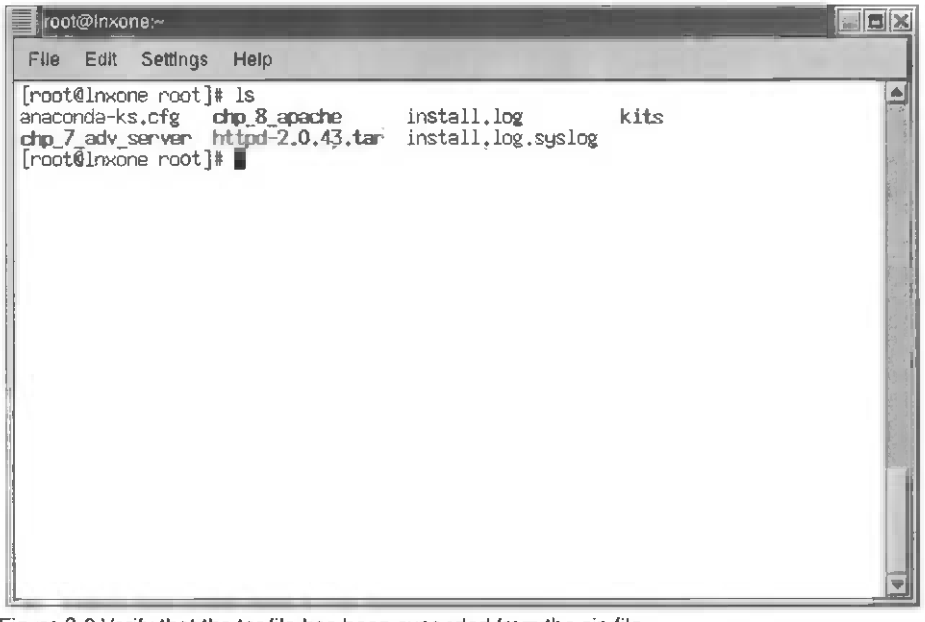


Figure 8-9 Verify that the tar file has been expanded from the zip file.

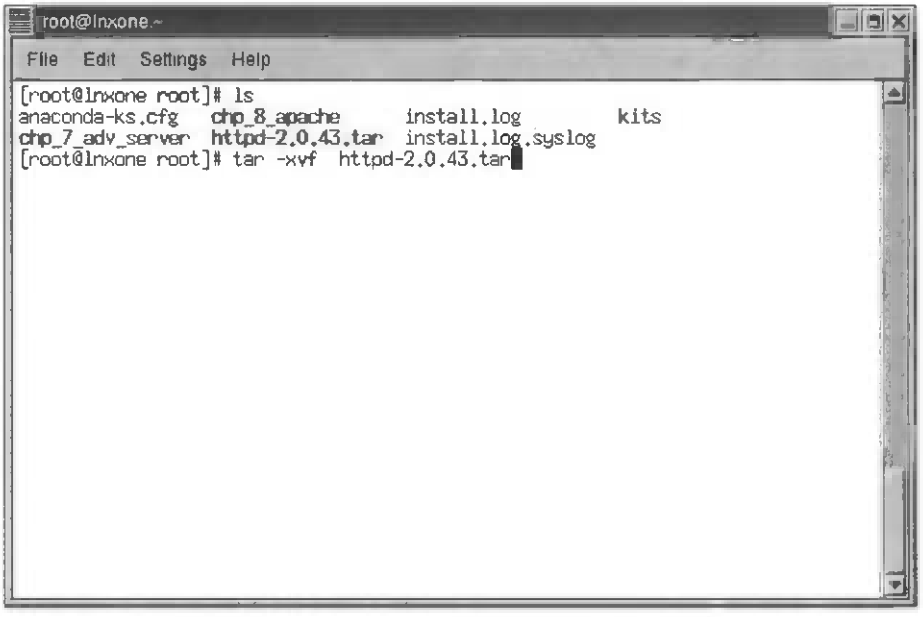


Figure 8-10 Then –xvf the tar archive and expand it under the root directory.

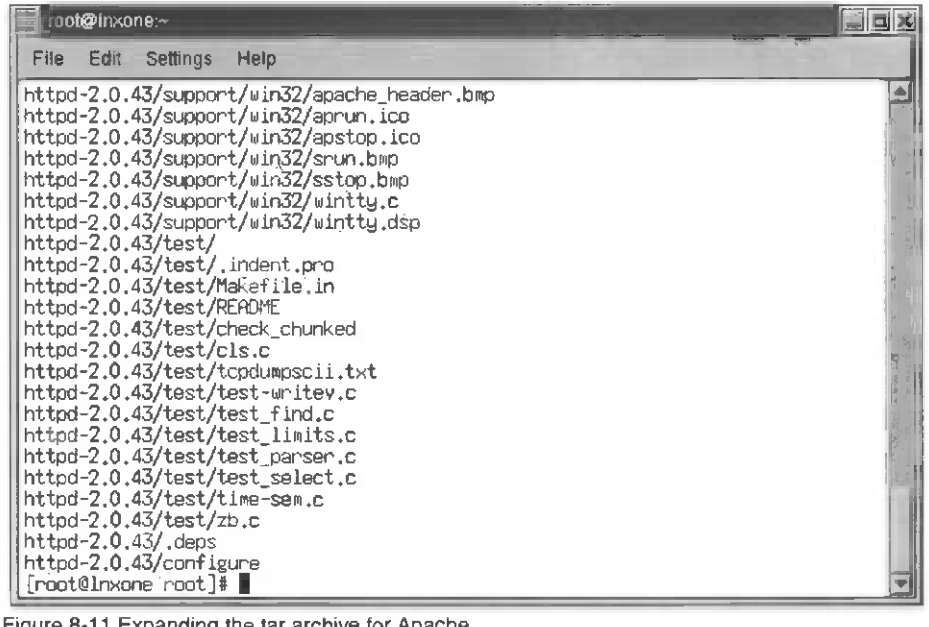


Figure 8-11 Expanding the tar archive for Apache.

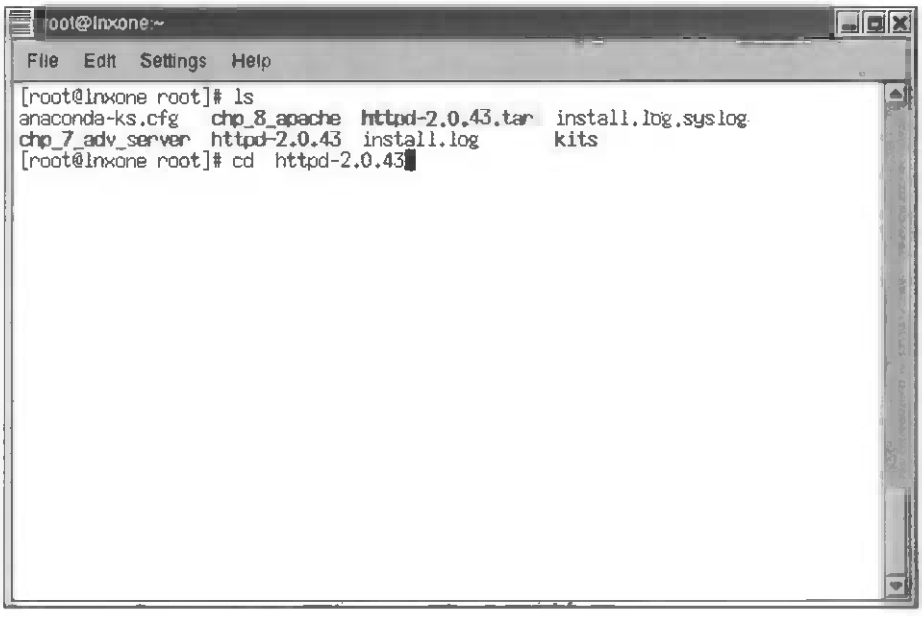


Figure 8-12 Verify that the httpd-2.0.43 directory structure has been created and then change the directory down into that tree.

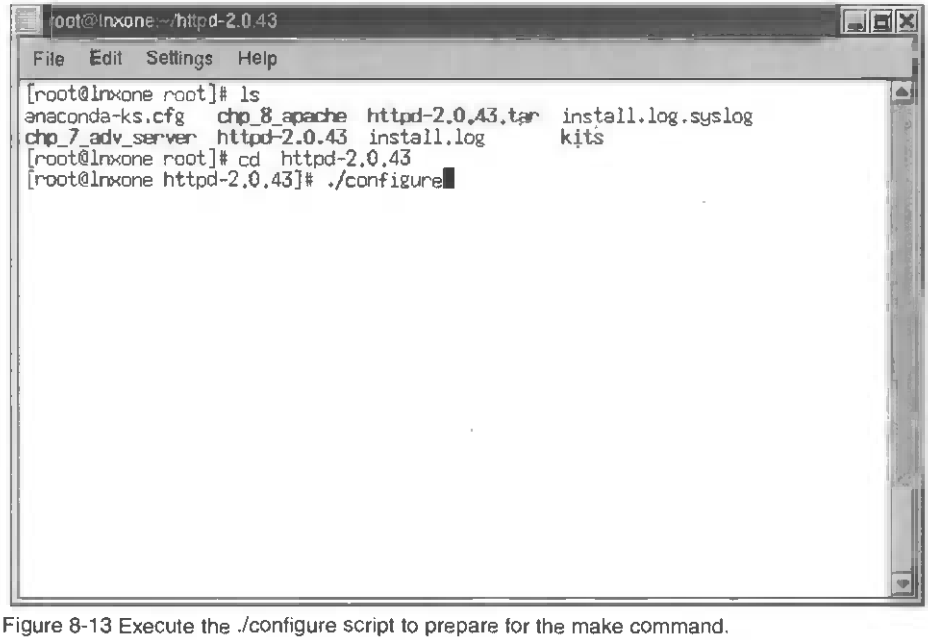


Figure 8-13 Execute the ./configure script to prepare for the make command.

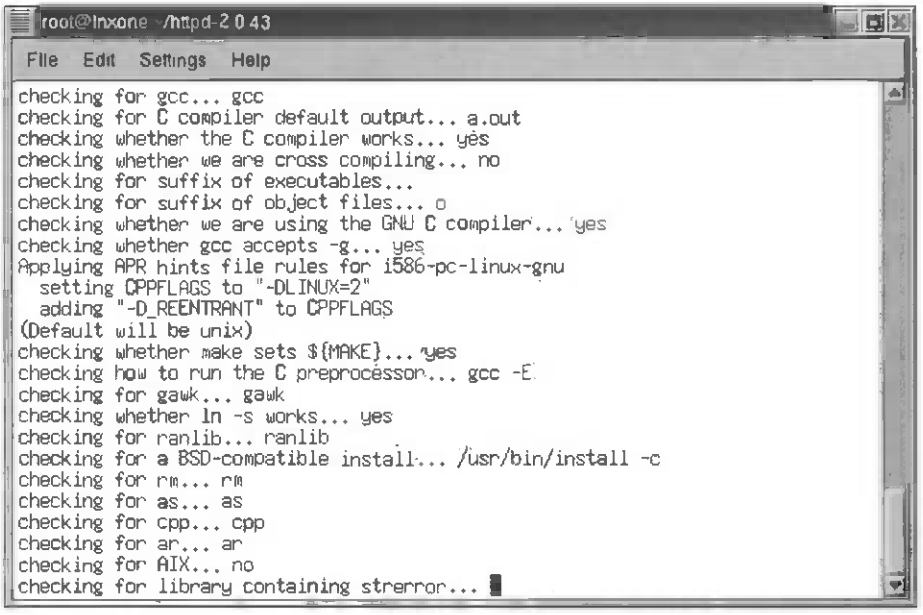


Figure 8-14 The ./configure continues until it has prepared the directory /files for the make commands.

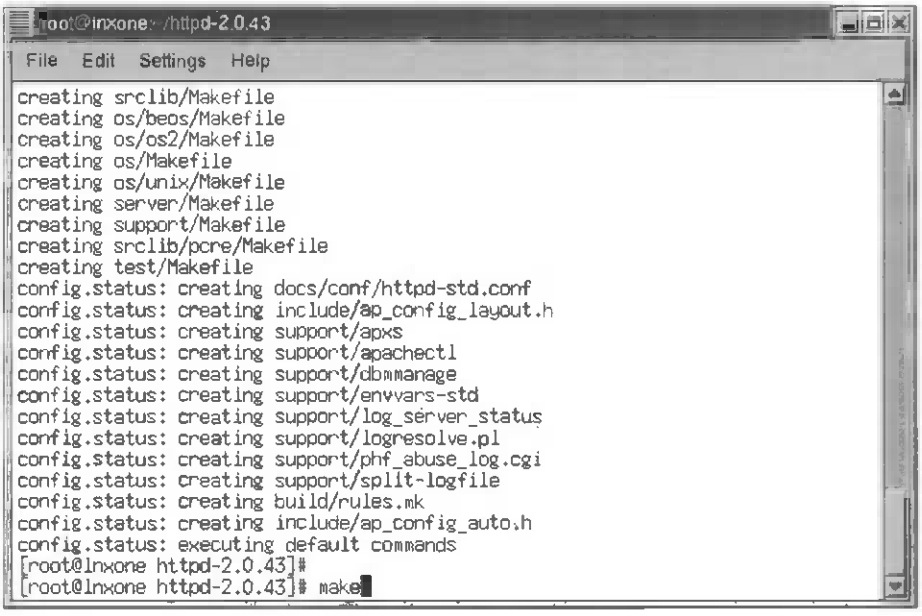


Figure 8-15 After the ./configure script has finished, execute the make command.

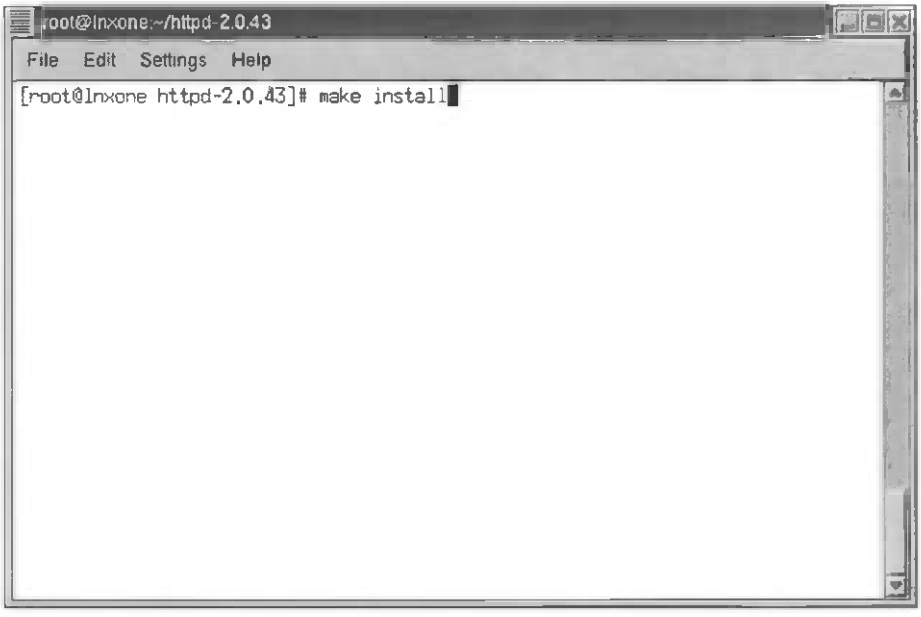


Figure 8-16 Continue after the make command with the make install command.

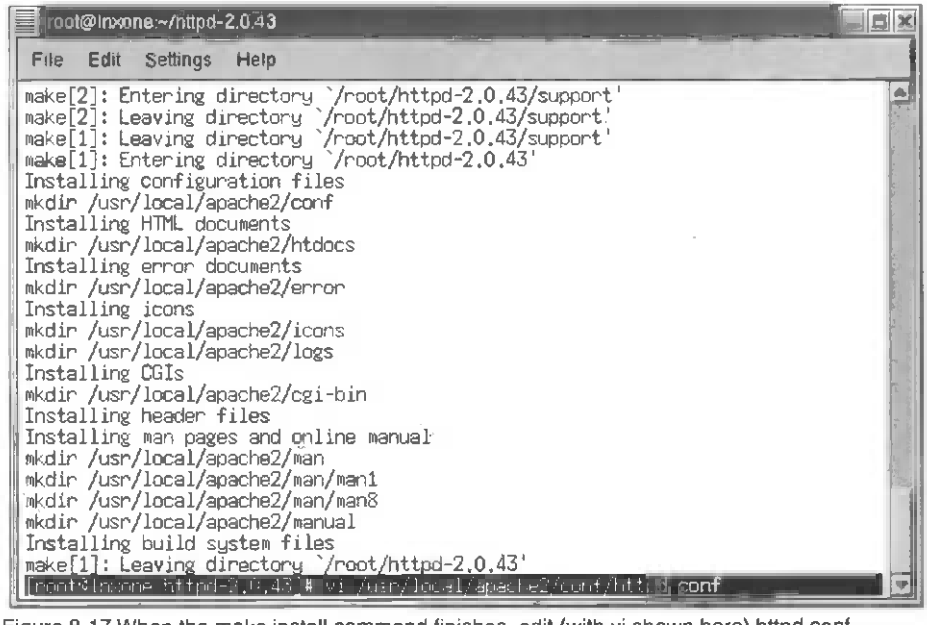


Figure 8-17 When the make install command finishes, edit (with vi shown here) httpd.conf.

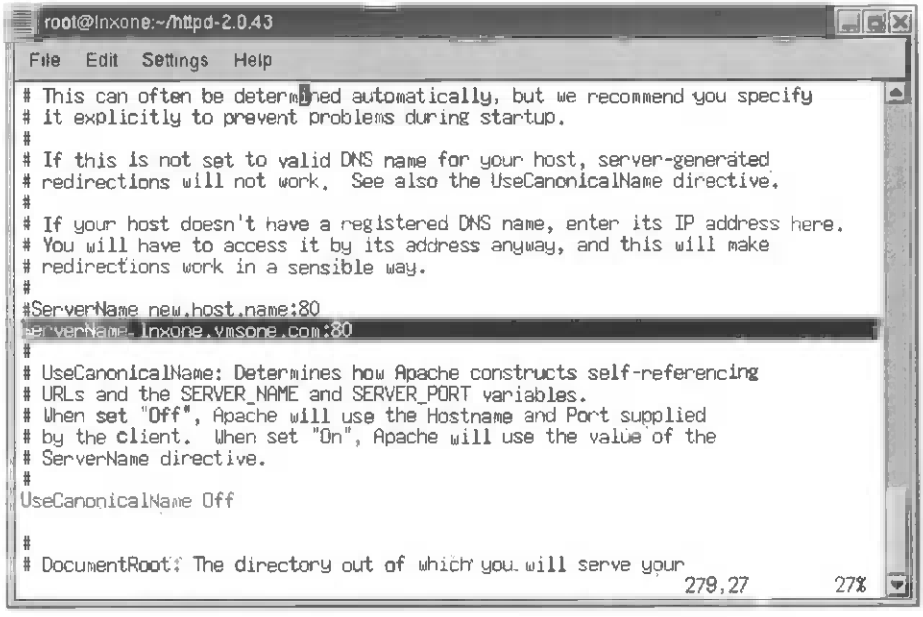


Figure 8-18 Change the configuration file and add (at least your server name).

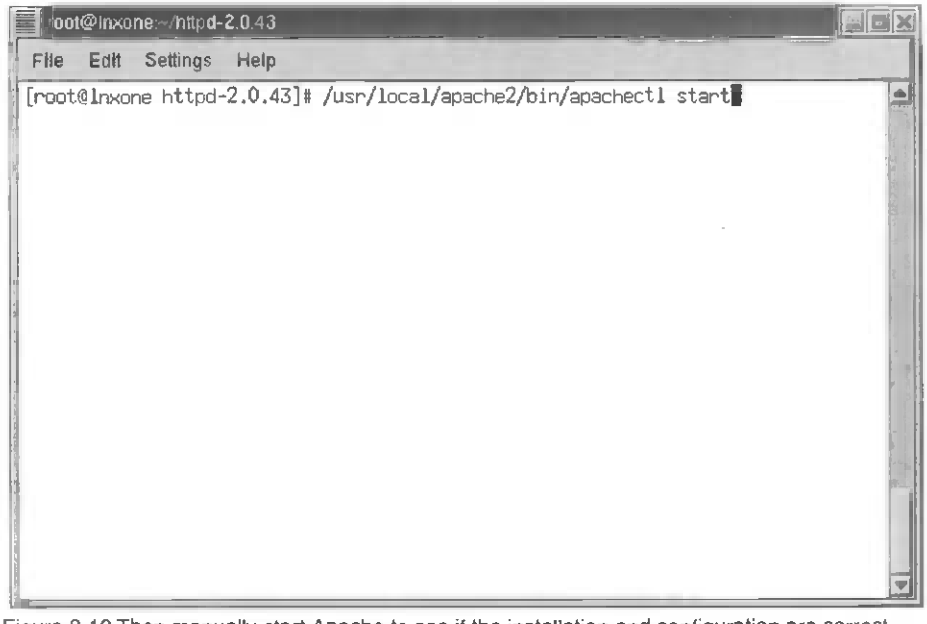


Figure 8-19 Then manually start Apache to see if the installation and configuration are correct. Note that using "stop" with apachectl will stop the Apache server!

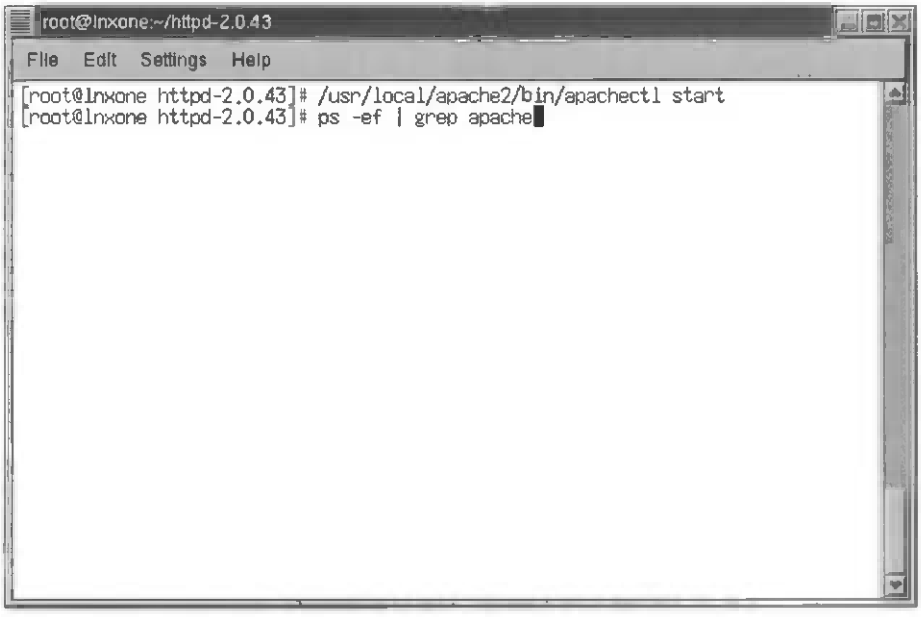


Figure 8-20 Execute a process status command to look for the processes named Apache.

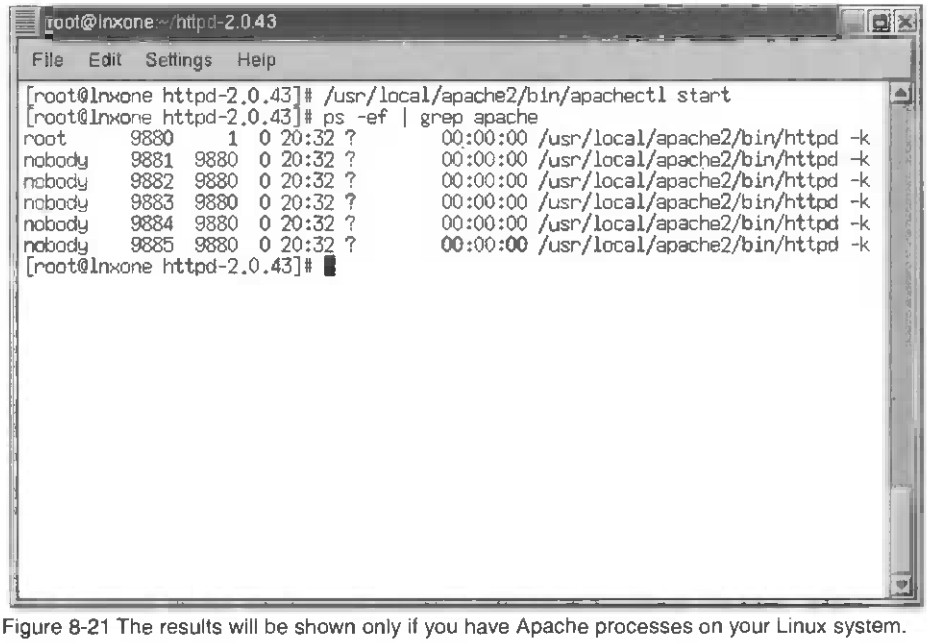


Figure 8-21 The results will be shown only if you have Apache processes on your Linux system.

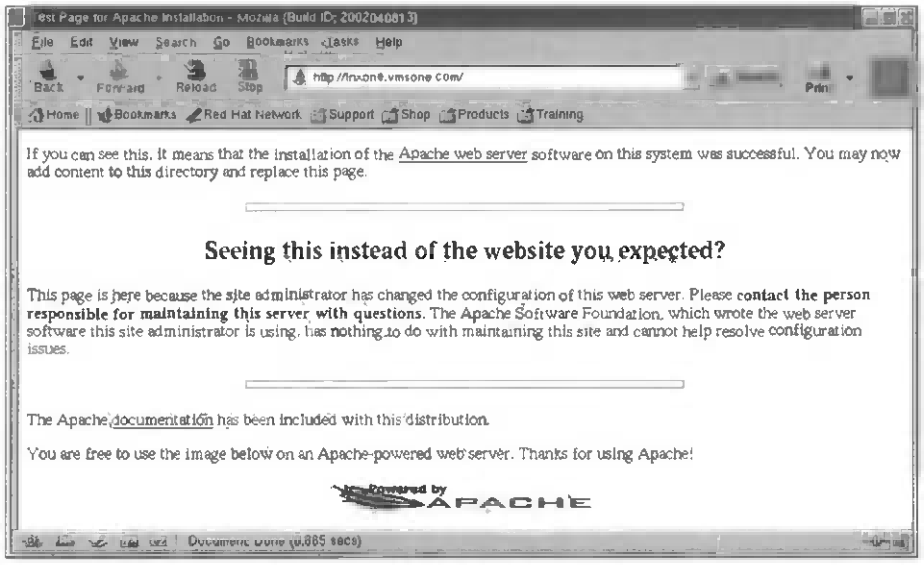


Figure 8-22 Go back to Mozilla or your favorite Web browser and verify that you're serving pages on port 80.

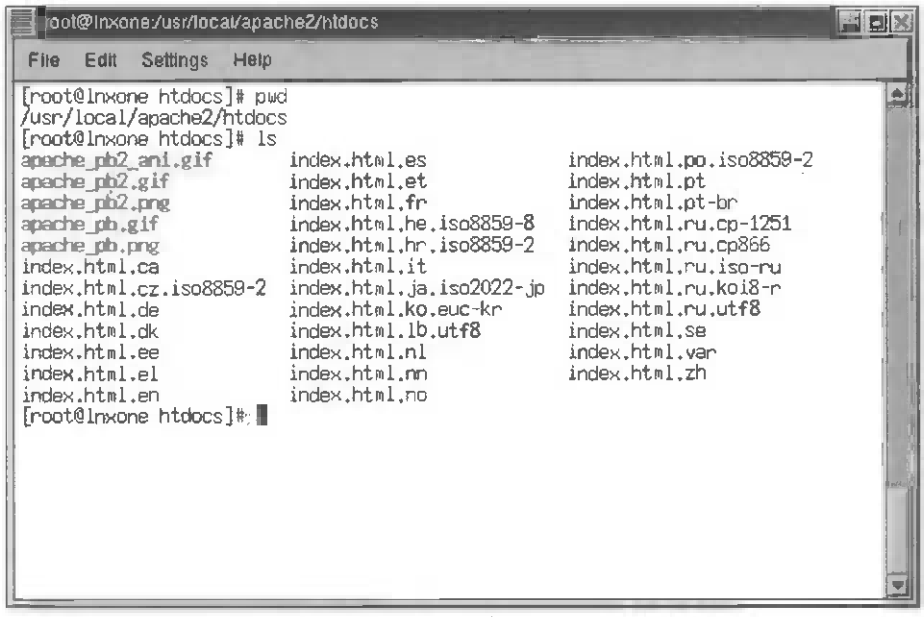


Figure 8-23 Add your HTML code and pages here! Congratulations, you're running Apache on Linux!

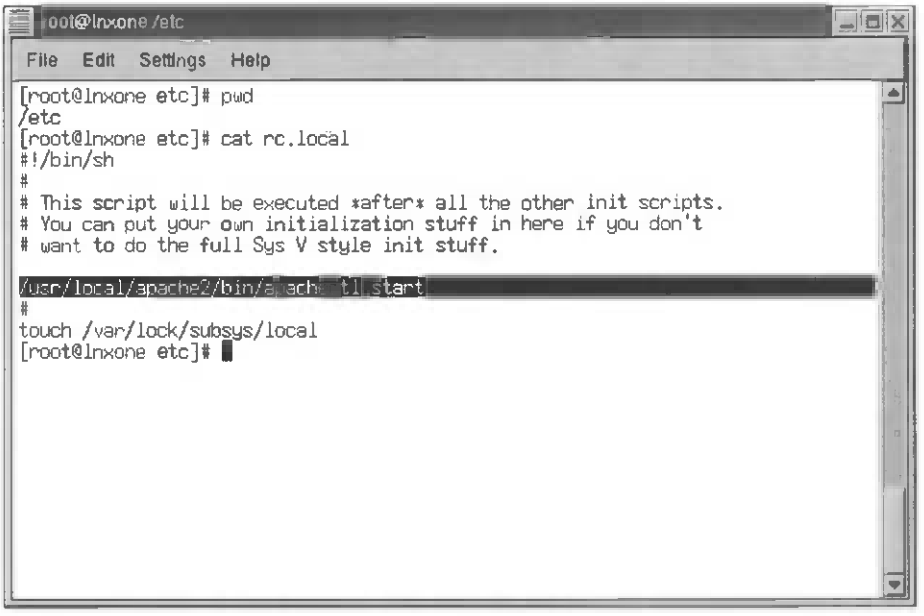


Figure 8-24 And if you want Apache to start/restart at boot time, be sure to edit the rc.local file.

OpenVMS Apache based Secure Web server

Figures 8-25 through 8-58 illustrate the OpenVMS Apache-based secure Web server.

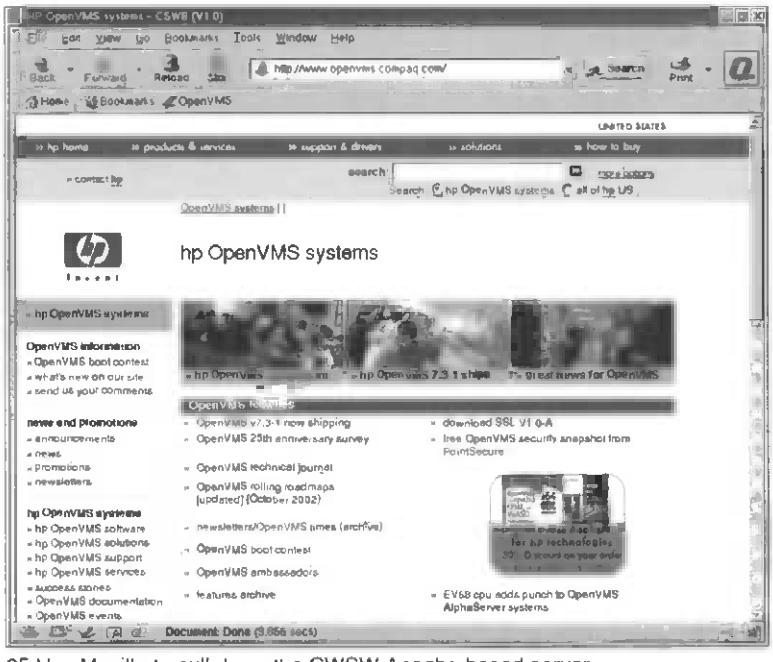


Figure 8-25 Use Mozilla to pull down the CWSW Apache-based server.

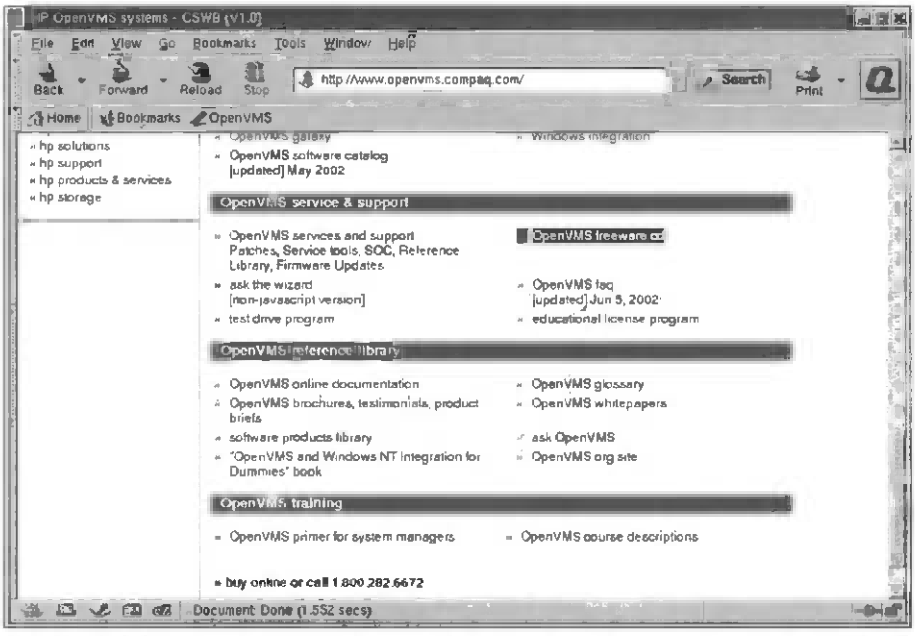


Figure 8-26 Note that it's located under freeware on the OpenVMS home page.

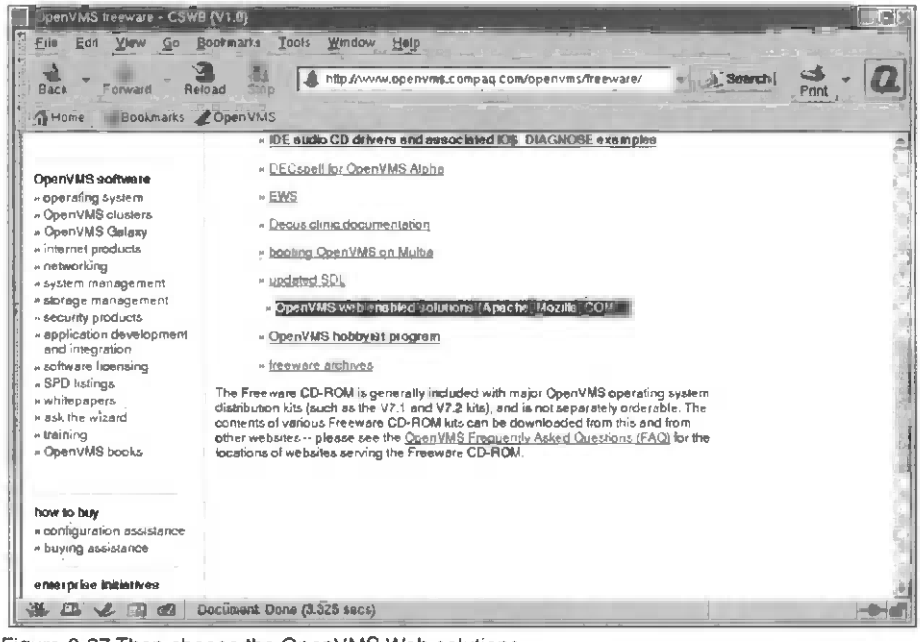


Figure 8-27 Then choose the OpenVMS Web solutions.

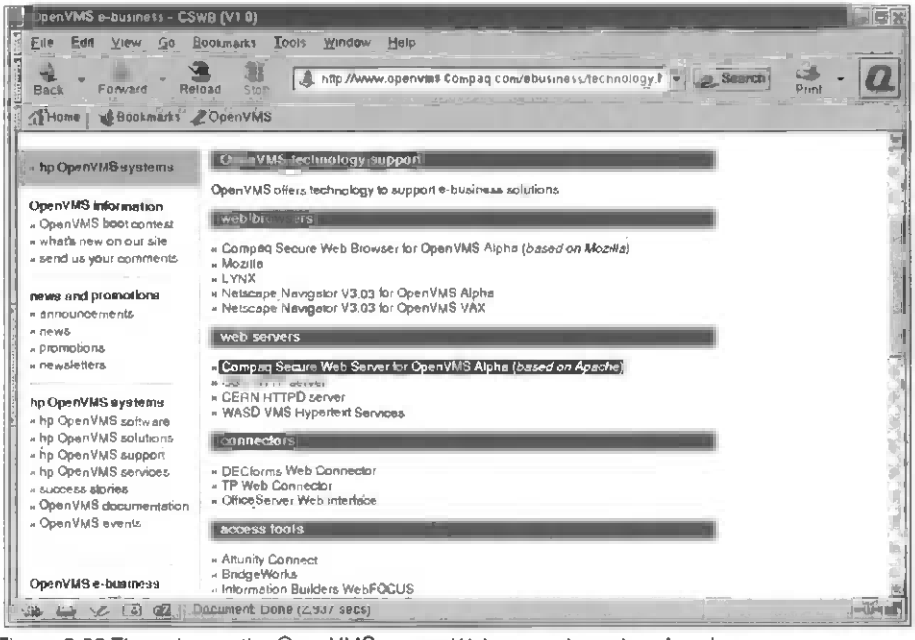


Figure 8-28 Then choose the OpenVMS secure Web server based on Apache.

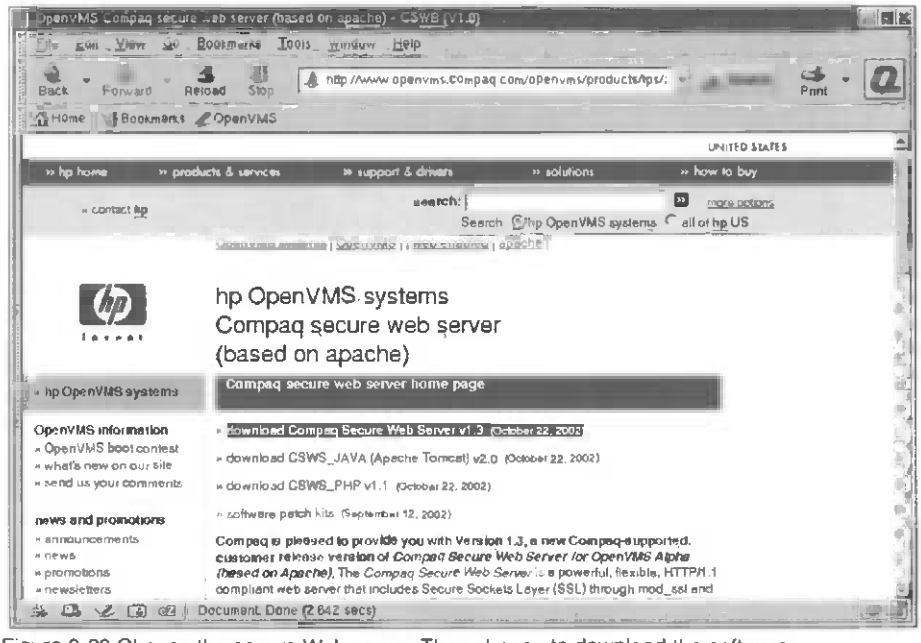


Figure 8-29 Choose the secure Web server. Then choose to download the software.

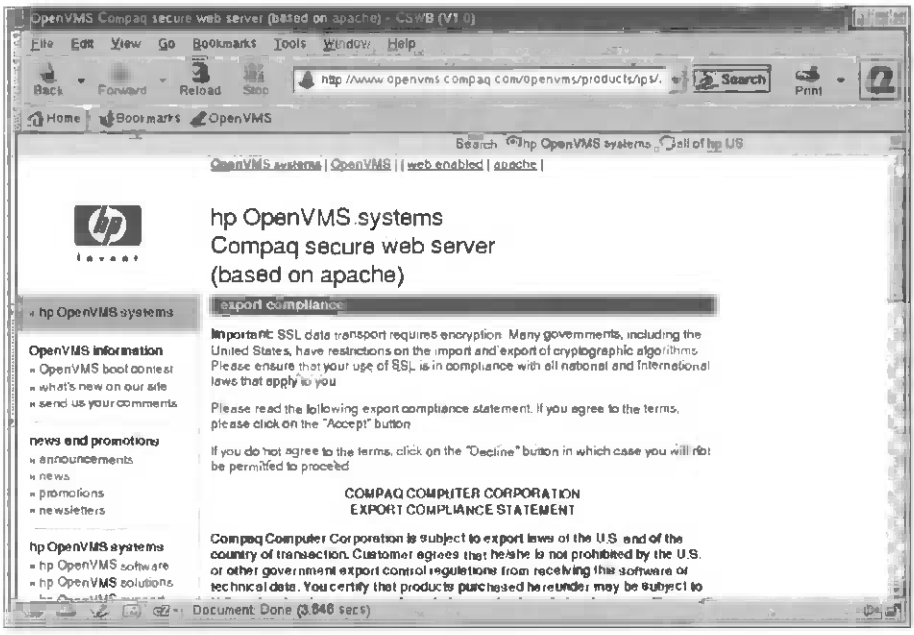


Figure 8-30 Read the export restrictions carefully; we wouldn't want you to violate the laws of the United States.

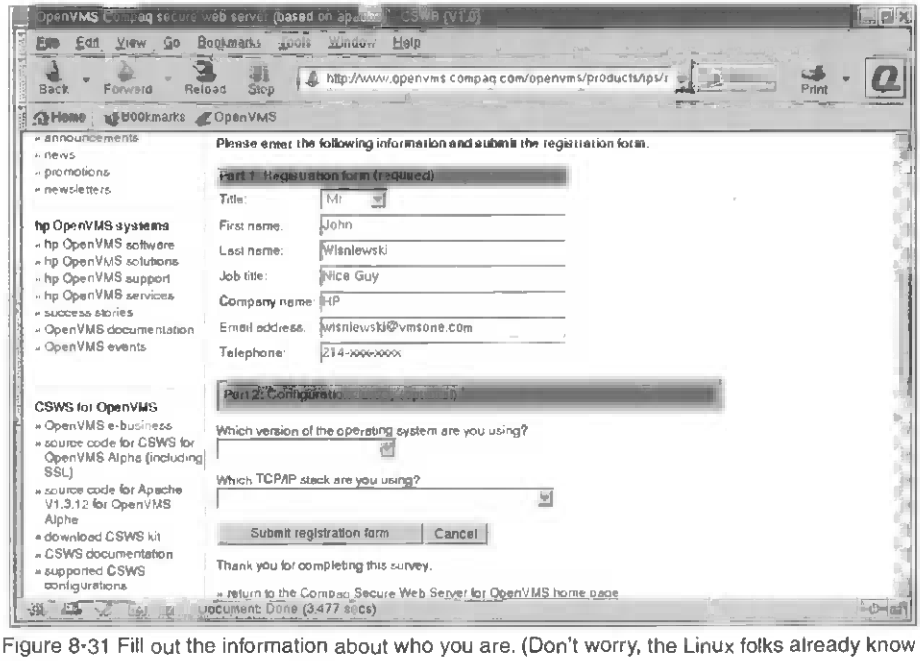


Figure 8-31 Fill out the information about who you are. (Don't worry, the Linux folks already know who you are.)

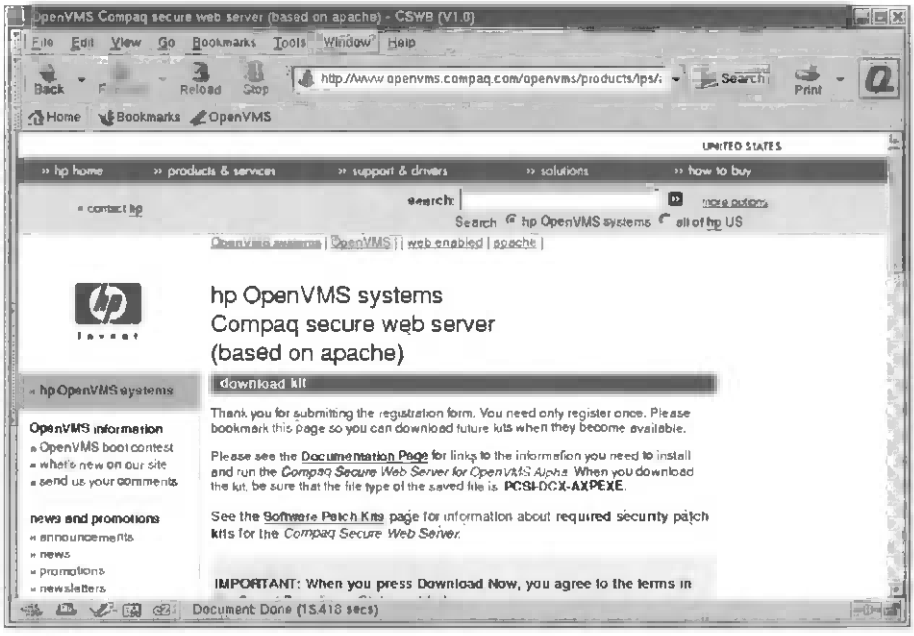


Figure 8-32 Really—only one more page to go! Now download the file!

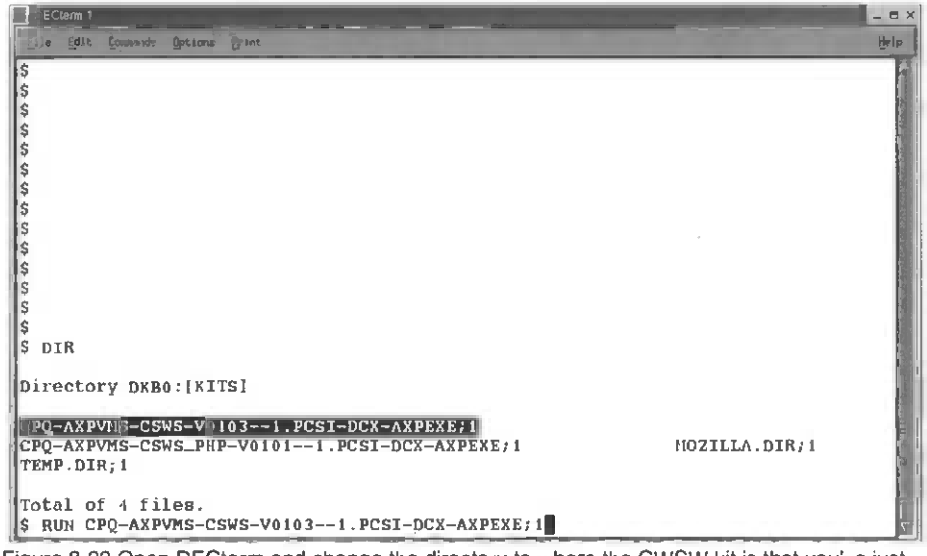


Figure 8-33 Open DECterm and change the directory to where the CWSW kit is that you've just downloaded. You must expand this kit before you use it by running the program (it's self-extracting).

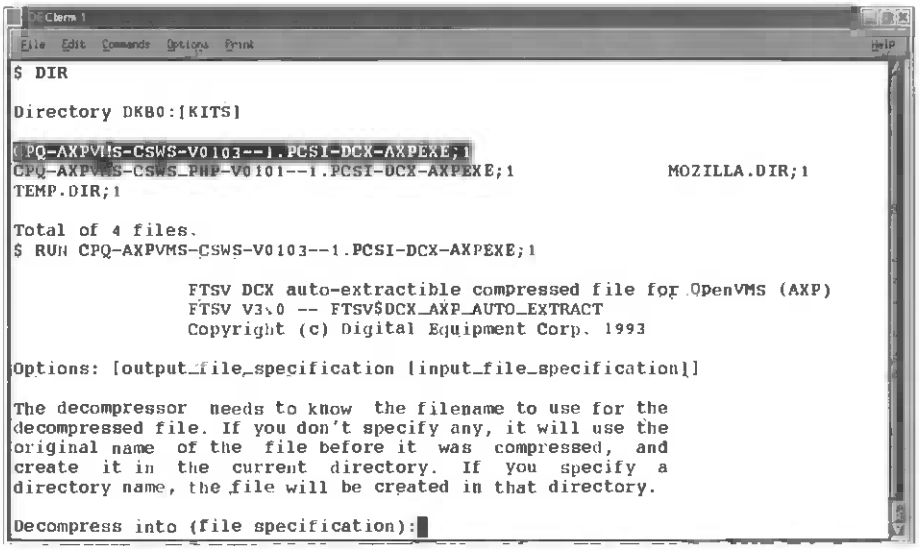


Figure 8-34 Just take the default to the questions and the file will be expanded in the same directory this *.axpexe file is in.

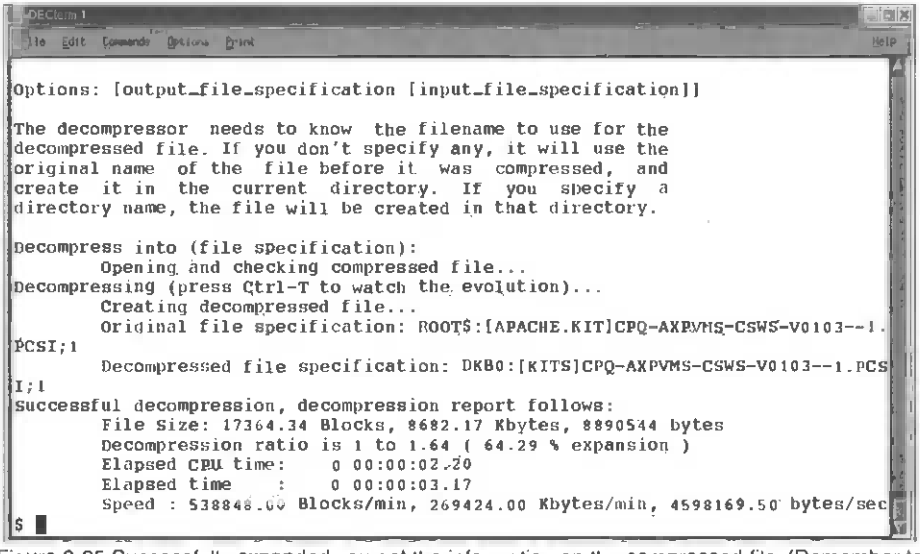


Figure 8-35 Successfully expanded you get the information on the compressed file. (Remember to save the green bar if you are doing the install from an LA120 at this point.)

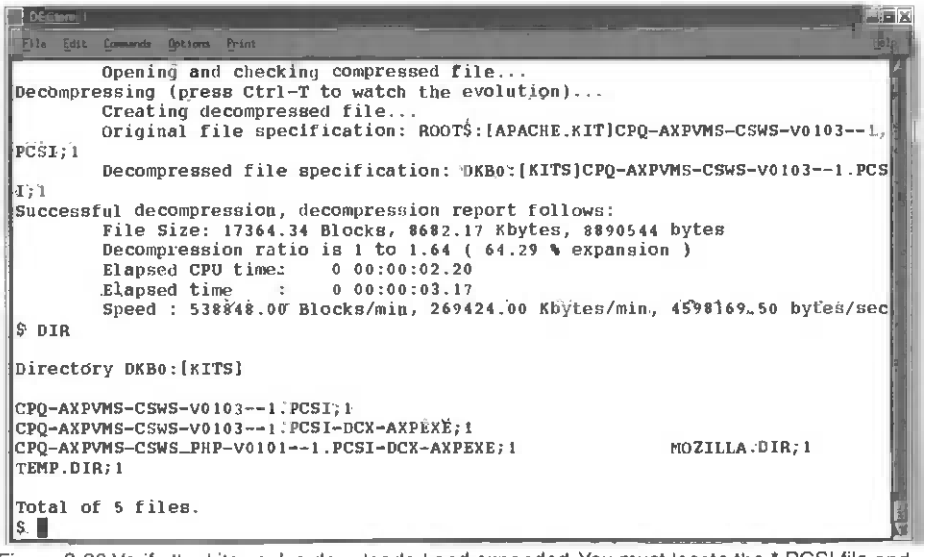


Figure 8-36 Verify the kits you've downloaded and expanded. You must locate the *.PCSI file and run an install with $product install csws.

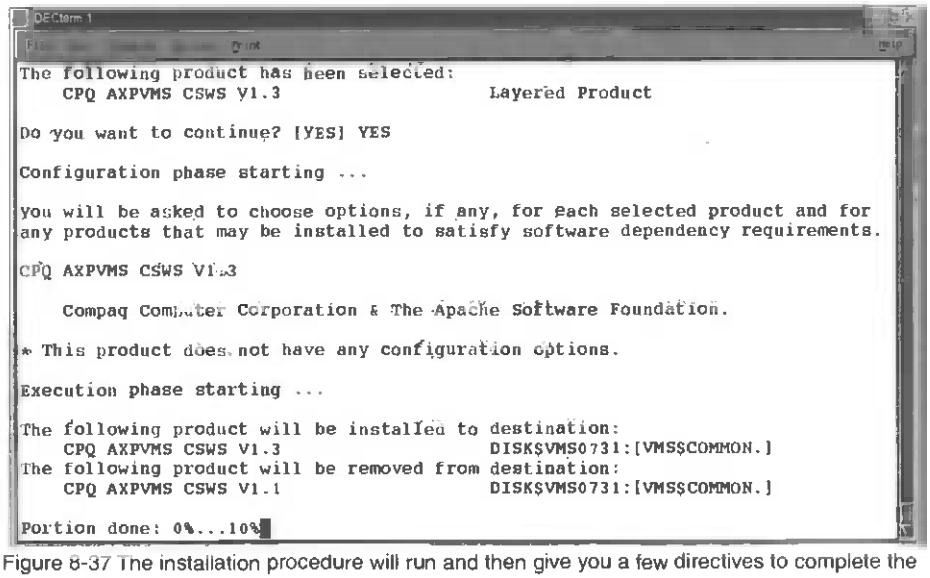


Figure 8-37 The installation procedure will run and then give you a few directives to complete the installation.

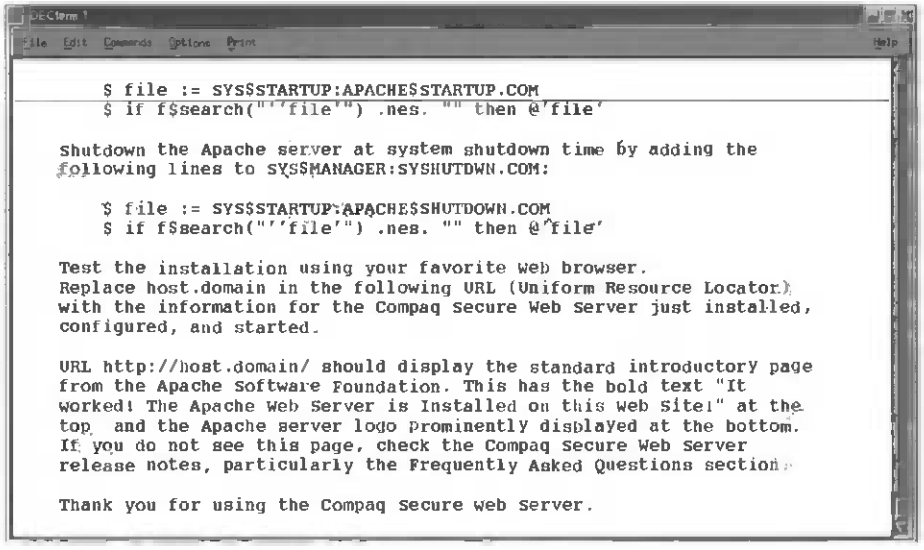


Figure 8-38 Read and record the directives or follow along as we customize the install.

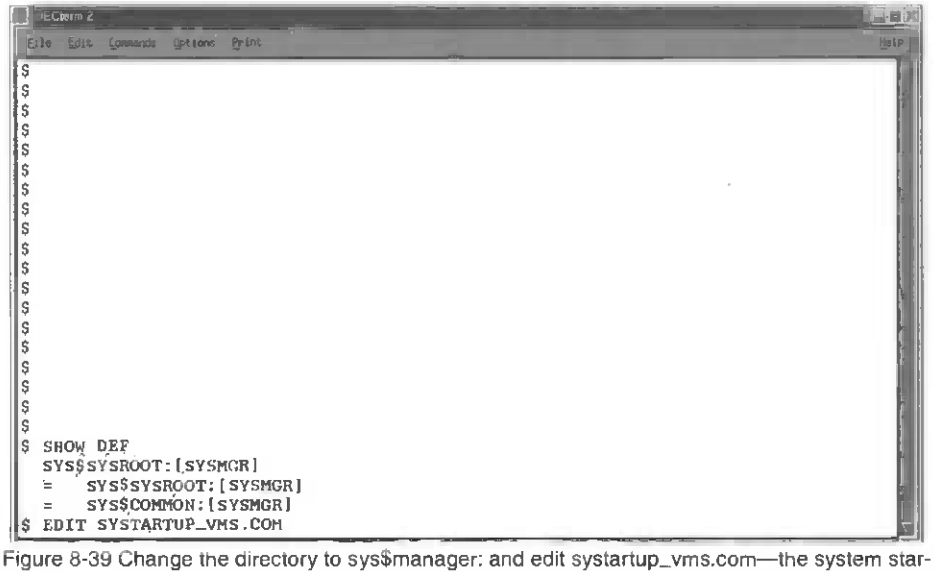


Figure 8-39 Change the directory to sys$manager: and edit systartup_vms.com—the system startup file.

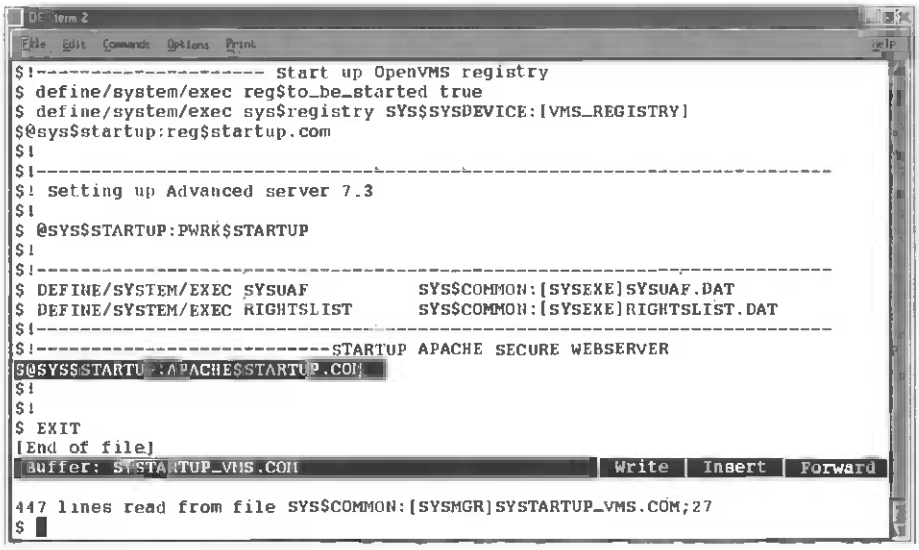


Figure 8-40 Add the line to start up Apache at the bottom of the system startup file and save it.

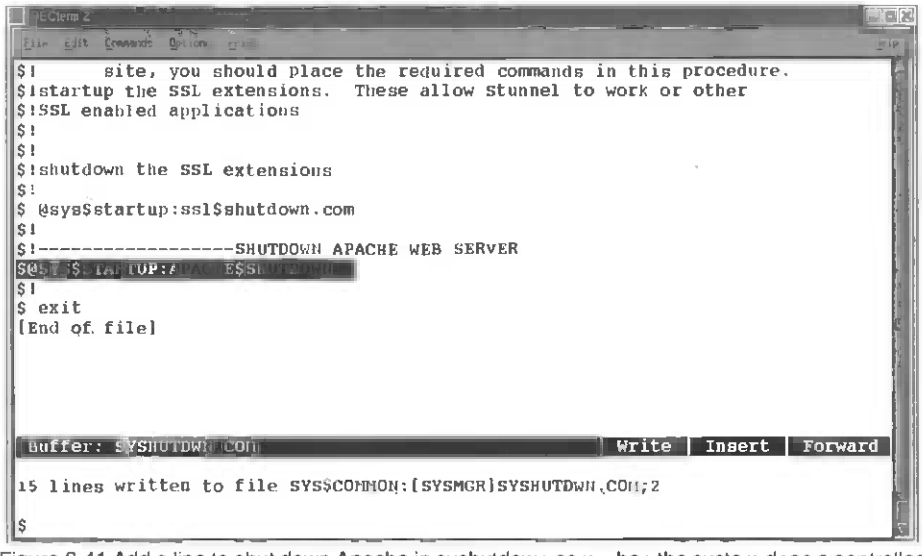


Figure 8-41 Add a line to shut down Apache in syshutdown.com when the system does a controlled shutdown

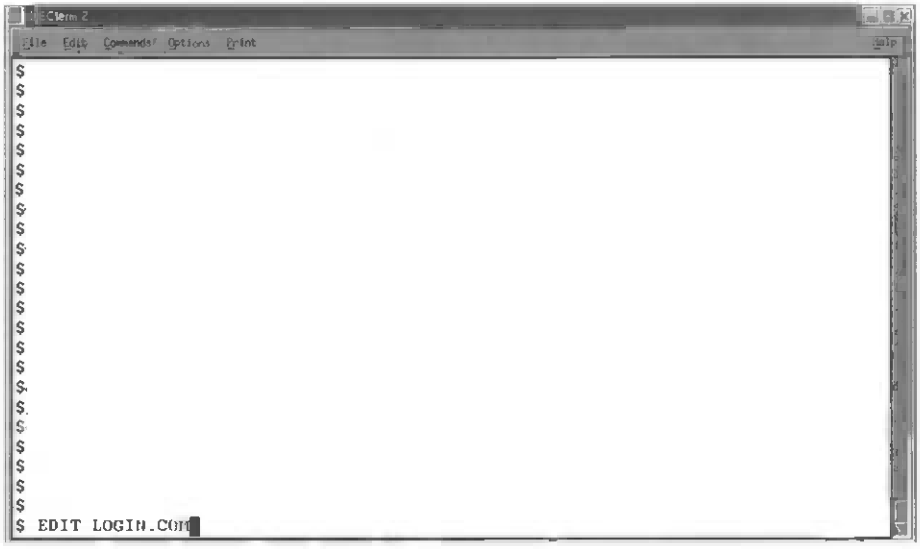


Figure 8-42 Edit the system manager's login.com script to add Apache startup and shutdown commands.

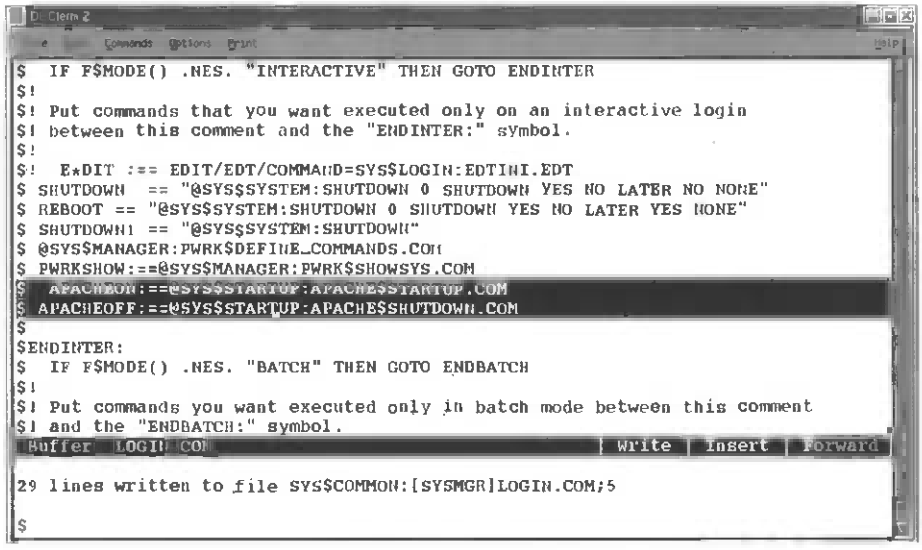


Figure 8-43 Add the lines for APACHEON AND APACHEOFF, so that when the system manager logs in he or she can turn Apache on and off easily!

Figure 8-44 Edit the SYLOGICALS.COM file, which defines system logicals at system boot/startup

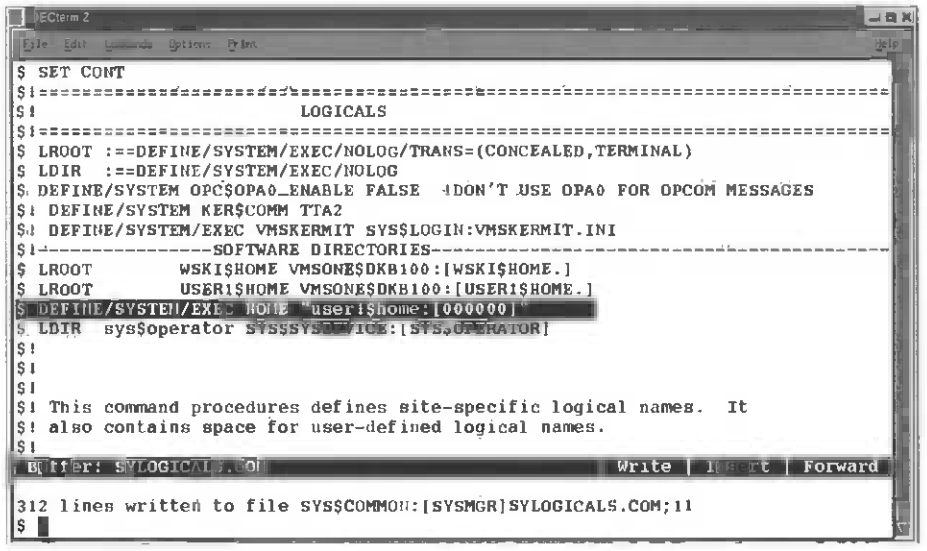


Figure 8-45 Define the "home" logical, so that the VMS users can use Apache from their home/login directories.

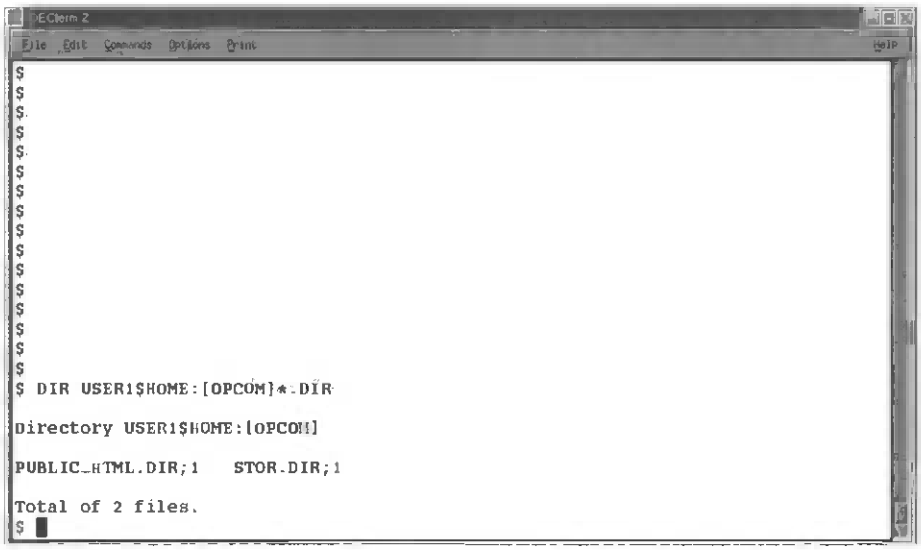


Figure 8-46 Users can create a subdirectory (from their login directory) called [.PUBLIC_HTML] that can contain their personal home page or any Web pages they want to serve up once the home directory is defined for Apache. You'll address personal Web pages from your browser as http://vmsone.com/~opcom to get access to these or other users' Web files.

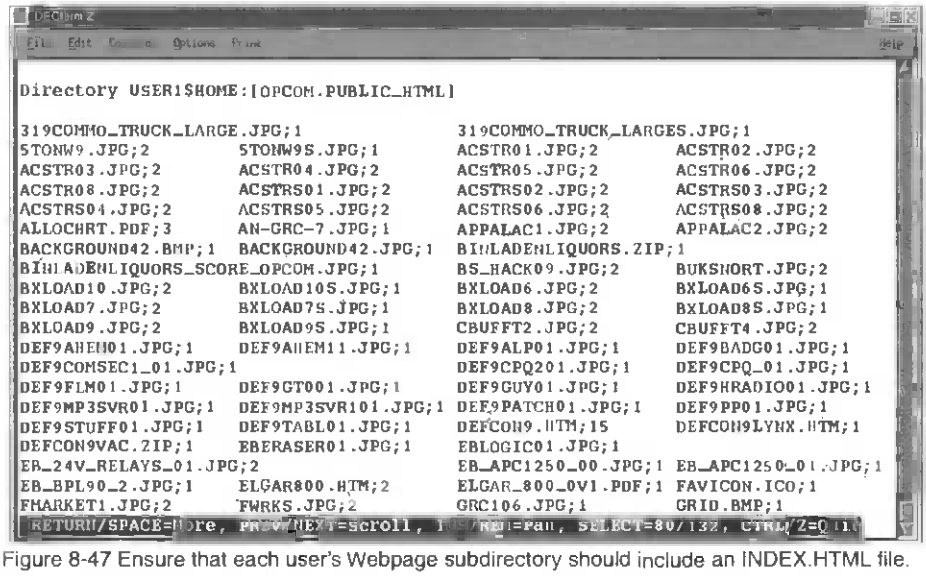


Figure 8-47 Ensure that each user's Webpage subdirectory should include an INDEX.HTML file. Without INDEX.HTML. the files in the subdirectories can be viewed and retrieved by the Web browser.

Figure 8-48 Before starting up the server for the first time, you must run Apache$config.com.

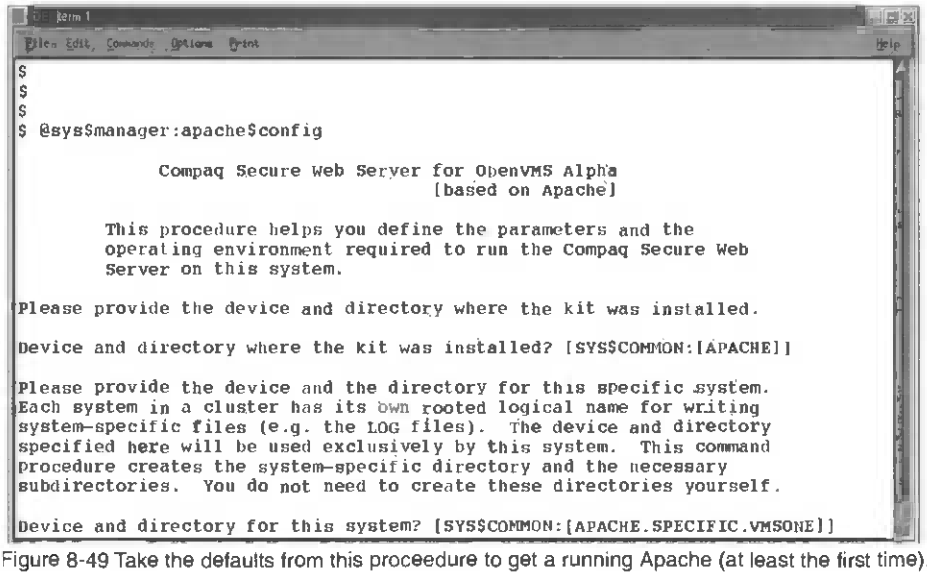


Figure 8-49 Take the defaults from this proceedure to get a running Apache (at least the first time).

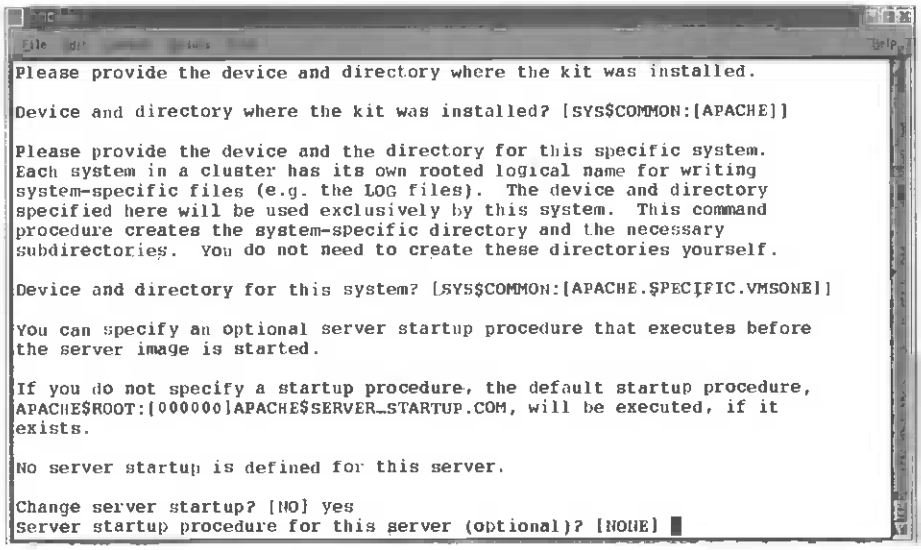


Figure 8-50 Answer yes to starting up the server/service.

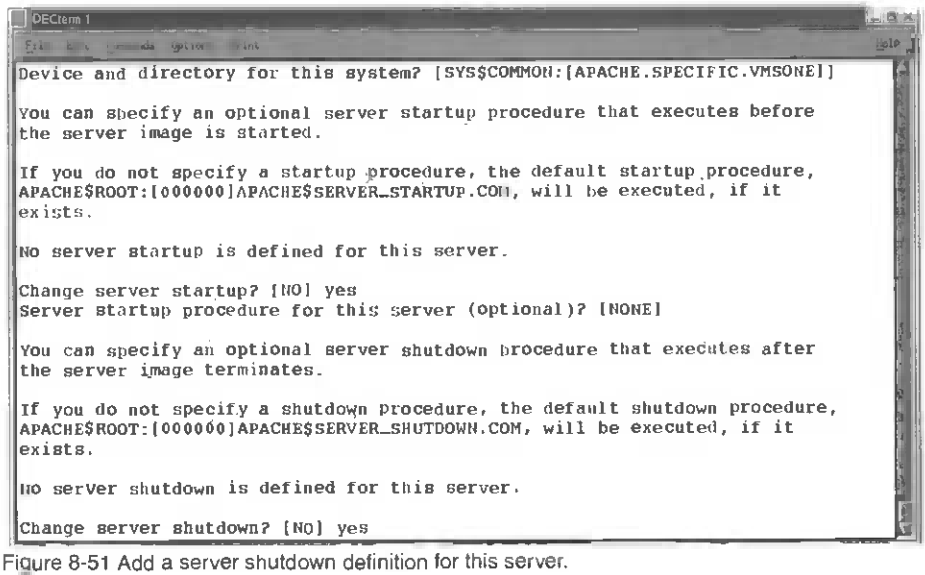


Figure 8-51 Add a server shutdown definition for this server.

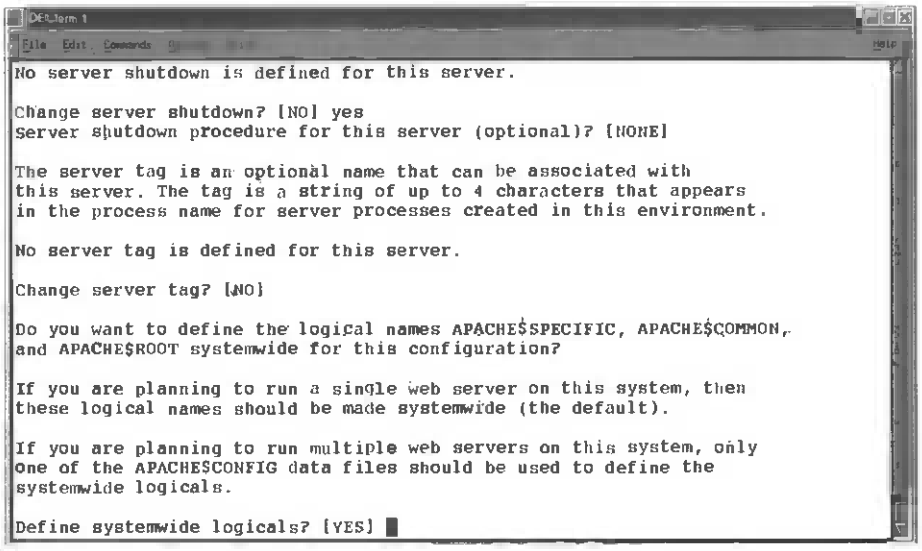


Figure 8-52 Choose yes to define systemwide logicals or No to exit procedure.

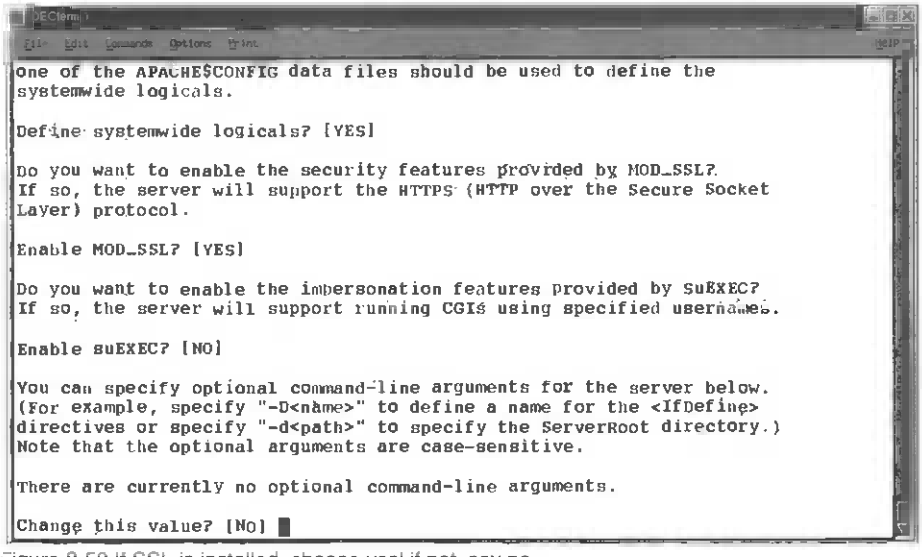


Figure 8-53 If SSL is installed, choose yes! if not, say no.

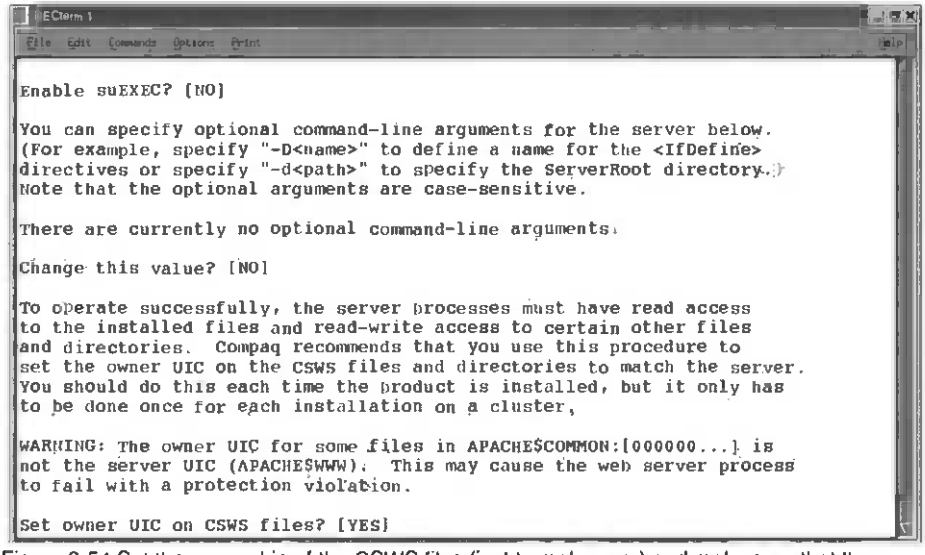


Figure 8-54 Set the ownership of the CSWS files (just to make sure) and make sure that the HTDOCS directory's protections are set correctly.

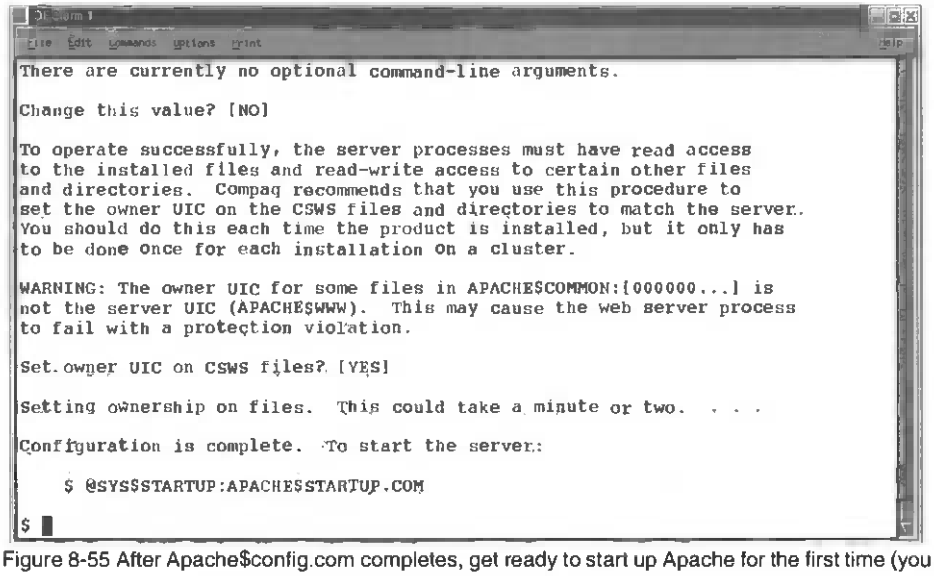


Figure 8-55 After Apache$config.com completes, get ready to start up Apache for the first time (you may even load your HTML files into APACHE$ROOT:[HTDOCS...] before you start up.

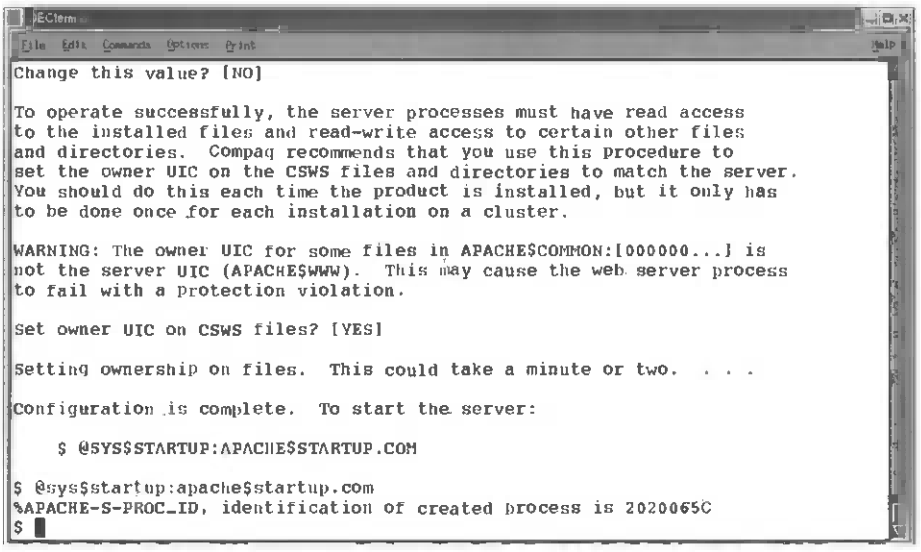


Figure 8-56 Run the Apache startup and look for the process creation message. If you want to see that the Apache processes are running, just do a $ SHOW SYSTEM/PROCESS=APACHE* command.

Figure 8-57 Use your favorite browser and verify that your VMS Apache-based Web server is up and running! Congratulations, you've installed and configured Apache for OpenVMS!

Linux and OpenVMS—The Future

In my day I've seen PDP 16-bit systems, VAX 32-bit systems, Alpha 64-bit systems, and now Itanium 64-bit systems. No doubt there will be a few more hardware platforms and systems in the general-purpose computing world before I'm ready to retire. Already Itanium supports 64-bit Linux, and the plan is to have OpenVMS and its layered products ported to Itanium by early 2004. What platform do we use when Windows, Linux, UNIX, VMS, NSK, and a host of other operating systems all run on the same hardware? As hard as it is to hear for many folks, one operating system isn't the answer for all computing jobs.

As we move forward with our industry, we need to look for the right tool for the right job and not be hung up about every computer being the same. Same interfaces, same applications, same tools? Certainly! Not the same operating environment.

We have needs for desktops, mid-range servers, super servers, clusters, and disaster-tolerant clusters; these aren't deployed just on a single OS—it's interoperability that gives us more choices, less cost, and better solutions for computer needs across our industry.

The features that are hallmarks for Linux simply don't add value at the high end of commercial systems, and without specific funding for that market niche, they may never become part of the Open Source and GNU world. The OpenVMS features that make it the best tool for 24/7/365 operations won't win any friends down on the desktop because that's simply not its forte.

Common ground, common applications, and common server applications are a great start to solving new problems, but learning about all the computing options that are available today lets us solve future problems that we couldn't solve using a single technology.

Linux, OpenVMS, and Open Source will be around for at least the next 20 years. I hope that we can all learn from today and make interoperability between systems and people that much better in the future.

See you in the future. It's where we're all destined to spend the rest of our lives!

Chapter 9 — A Compendium of Open Source Tools for OpenVMS and Linux

Open Source Tools

Look for the latest versions of these products when you want them (it's always best to use the latest!).

BASH bash is the GNU Project's Bourne Again shell, a complete implementation of the POSIX.2 shell specification with interactive command-line editing, job control on architectures that support it, csh-like features such as history substitution and brace expansion, and a slew of other features. Here is a short list of some of the features available in BASH.

OpenVMS distribution.

```
http://www.openvms.compaq.com
http://cnswww.cns.cwru.edu/~chet/bash/bashtop.html
```

BZIP Is another Open Source ZIP. Net says it's better then GZIP.

```
http://www.dlhoffman.com/publiclibrary/RPM/BByName.html
http://www.ourservers.net/openvms_ports/bzip2/bzip2_contents.html
```

CircleMUD is a multiuser dungeon (MUD) game system written by Jeremy Elson, originally at Johns Hopkins University's Department of Computer Science (I graduated in May 1996). CircleMUD is a derivative of DikuMUD Gamma 0.0, which was written in 1990 at DIKU, the Department of Computer Science at the University of Copenhagen, by Katja Nyboe, Tom Madsen, Hans Henrik Staerfeldt, Michael Seifert, and Sebastian Hammer.

```
http://www.circlemud.org/
http://www.ourservers.net/openvms_ports/circlemud
     /circlemud_contents.html
```

Corba The OMG specification suite defines just the environment; CORBA enables natural interoperability regardless of platform, operating system, programming language, or network hardware and software (although CORBA defines a mandatory TCP/IP-based protocol for interoperability over the Internet and most intranets).

```
http://www.corba.org
http://www.openvms.compaq.com/openvms/brochures/appdev/pages
     /PAGE7.HTM
```

Emacs

```
http://www.gnu.org/directory/GNU/emacs.html
http://vms.gnu.org/software/released1/
```

FreeCiv is a free turn-based multiplayer strategy game in which each player becomes the leader of a civilization, fighting to obtain the ultimate goal.

```
http://www.freeciv.org/
http://www.ourservers.net/openvms_ports/index.html
http://www.ourservers.net/openvms_ports/freeciv
     /freeciv_contents.html
```

GIMP is the GNU Image Manipulation Program is a freely distributed piece of software suitable for such tasks as photo retouching, image composition, and image authoring. This site contains information about downloading, installing, using, and enhancing GIMP. This site also serves as a distribution point for the latest releases, patches, plugins, and scripts. We also try to provide as much information about the GIMP community and related projects as possible.

```
http://www.gimp.org
```

GNUCC GCC development is a part of the GNU project, aiming to improve the compiler used in the GNU system, including the GNU/Linux variant. The GCC development effort uses an open development environment and supports many other platforms in order to foster a world-class optimizing compiler, to attract a larger team of developers, to ensure that GCC and the GNU system work on multiple architectures and diverse environments, and to more thoroughly test and extend the features of GCC.

```
http://gcc.gnu.org/
```

GNUPG stands for GNU Privacy Guard and is GNU's tool for secure communication and data storage. It can be used to encrypt data and to create digital signatures. It includes an advanced key management facility and is compliant with the proposed OpenPGP Internet standard described in RFC 2440. As such, it is aimed to be compatible with PGP from NAI, Inc.

```
http://www.gnupg.org/
```

GTK+ is a multiplatform toolkit for creating GUIs. Offering a complete set of widgets, GTK+ is suitable for projects ranging from small one-off projects to complete application suites.

```
http://www.gtk.org
http://www.openvms.compaq.com/openvms/products/ips/gtk.html
http://www.ourservers.net/openvms_ports/gtk/gtk_contents.html
```

OpenSSL The OpenSSL project is a collaborative effort to develop a robust, commercial-grade, full-featured, Open Source toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols, as well as a full-strength, general-purpose cryptography library managed by a worldwide community of volunteers who use the Internet to communicate, plan, and develop the OpenSSL toolkit and its related documentation.

OpenVMS distribution

```
http://www.openvms.compaq.com/
http://www.openssl.org
```

PERL is a popular and powerful scripting language for Linux and other OSs it works well with Apache.

```
http://www.perl.com
http://www.openvms.compaq.com/openvms/products/ips/apache
     /csws_modperl.html
```

PHP is a server-side, cross-platform, HTML-embedded scripting language that lets you create dynamic Web pages. PHP-enabled Web pages are treated the same as regular HTML pages, and you can create and edit them the way you normally create regular HTML pages.

```
http://www.php.net
http://www.openvms.compaq.com/openvms/products/ips/apache
     /csws_php.html
```

POVray, the Persistence of Vision Raytracer, is a high-quality, totally free tool for creating stunning three-dimensional graphics. It is available in official versions for Windows, Mac OS/Mac OS X, and i86 Linux. The source code is available for those wanting to do their own ports.

```
http://www.povray.org/
http://www.software.ethz.ch/OpenVMS_AXP_Distr
     /9601-OpenVMS_Freeware.html
http://www.ourservers.net/openvms_ports/povray/povray_contents.html
http://www.ourservers.net/openvms_ports/megapovray
     /megapovray_contents.html
```

PYTHON is an interpreted, interactive, object-oriented programming language. It is often compared to Tcl, Perl, Scheme, or Java.

```
http://www.python.org
http://www.pi-net.dyndns.org/docs/python_vms/genman/toc_gen.html
```

SAMBA is an Open Source/Free Software suite that provides seamless file and print services to SMB/CIFS clients. SAMBA is freely available under the GNU General Public License.

```
http://www.samba.org/
http://de.samba.org/samba/ftp/Binary_Packages/vms/
```

RADIUS is the Remote Access Dial-In User Service, an authorization, authentication, and accounting client/server protocol. RADIUS is the de facto industry standard for remote access AAA (authentication, authorization, accounting), as well as an IETF standard. In general, it's a network daemon (network process) that performs authentication, authorization, and accounting actions when someone logs in on a network.

```
http://www.freeradius.org/
http://www.radiusvms.com
```

VNC, or Virtual network computing, is a remote display for Windows systems (X and MS).

```
http://www.uk.research.att.com/vnc
http://www.tmk.com/ftp/vms-freeware/fileserv/vnc333r1vms011.zip
```

Vim is an almost compatible version of the UNIX editor Vi.

```
http://packages.debian.org/unstable/editors/vim.html
http://www.thomer.com/vi/vi.html
```

WGET GNU WGET is a free software package for retrieving files using HTTP, HTTPS, and FTP, the most widely used Internet protocols. It is a noninteractive command-line tool, so it may easily be called from scripts, cron jobs, terminals without X support, and so on.

```
http://axp603.gsi.de:8080/www/vms/sw/ger/wget.htmlx
http://wget.sunsite.dk/index.html
```

OpenVMS and Linux Open Source Archives

```
http://vms.gnu.org
http://www.openvms.compaq.com/openvms/freeware
http://vms.process.com/fileserv-software.htmlp9
http://www.ourservers.net/openvms_ports/index.html
http://www.gnu.org/directory/GNU
```

See the VMS Web site for Open Source ports by VMS engineering.

Index

A

M

N