Maximum Security -- Ch 21 -- Plan 9 from Bell Labs

38-48 minutes

A Hacker's Guide to Protecting Your Internet Site and Network

Previous chapter Next chapter Contents

21

Plan 9 from Bell Labs

Almost thirty years ago, the team at Bell Labs (now Lucent Technologies) changed the world by developing what would later become the most popular networked operating system in history. From then until now, UNIX has ruled the Internet. Even if that were the only contribution ever made by Bell Labs personnel, it would have been sufficient. They would been have been held in high regard as having achieved something truly useful and important. As any programmer will tell you, however, the contributions from Bell Labs kept coming.

In the early 1990s, the folks at Bell Labs were still busy. This time, however, they had more than 25 years of experience under their belts. With that experience, they challenged themselves to create the *ultimate* networked operating system. Did they succeed? You bet. It is called *Plan 9 from Bell Labs*.

The Basics

The team at Bell Labs (which includes such heavy-duty names as Ken Thompson and Dennis Ritchie) were reportedly dissatisfied with the then-current trends in computer technology. They realized that hardware considerations made networking a difficult proposition, one that didn't always work out in terms of cost effectiveness or performance. Hardware dependencies and proprietary design make networking more of a forced environment than a truly fluid, easy one. By *forced environment*, I mean that dozens of often disparate and incompatible protocols, drivers, and software are patched together to provide a shaky and sometimes unreliable integration of networks.

Alas, although the Internet may sometimes be referred to as a miracle of distributed computing, it isn't. The current system works only because we have forced the TCP/IP stack upon a handful of architectures (many that were never designed to run these protocols). Thus, the Internet has the *appearance* of being an amalgamated, united network. On closer examination, however, it is clear that the Internet is exploiting only a very meager portion of the networking power at its disposal.

Consider this: FTP is one of the most commonly used techniques to move information from one place to another. When a user transfers a file via FTP, he is a *remote* user, accessing some resource on a server in the void. The word *remote* is the key feature here. It denotes a condition wherein the user is isolated. To access the resources at the other end, the user must perform several actions (these may include initiating the FTP session, unzipping the file, placing it in the proper directory, and so on). FTP therefore places the user at arm's length. The use of the resource does not occur in a fluid environment.

Similarly, and to an even greater extent, HTTP isolates the user. True, it appears to the user as though he is working interactively with a Web site, but he isn't. In fact, HTTP may isolate the user more than any other network protocol. For example, you are not logged in as you are with Telnet or FTP. In fact, you are connected only for the brief periods--seconds, actually--necessary for your client to relate which resources it needs. This is the farthest thing from a traditional shared network environment.

In contrast, suppose that instead of retrieving a file and placing it in your physical location, you simply want to use the file momentarily. This is sometimes achieved through file sharing in proprietary network environments (environments where a directory or a file can be attached to the local machine). In such cases, the resource appears and behaves as though it is on the local machine. This technique is more akin to true, networked computing. It is a one-step process.

Now imagine an operating system that was designed to interface in this manner with many different types of systems and hardware, an operating system that could provide this real networking to hundreds (or even thousands) of workstations, irrespective of hardware constraints. Imagine an operating system that makes FTP directories of remote machines appear as local directories (regardless of where the target server may be located). If you can imagine this, you are well on your way to understanding the basic characteristics of Plan 9.

What Plan 9 Is Not

Plan 9 is not UNIX, or any variant thereof. But if you install the demo distribution, you may initially be confused on this point. At first glance, it looks a lot like UNIX (particularly when you make a directory listing). Make no mistake, though. Plan 9 is an entirely new operating system. As explained in the Plan 9 from AT&T Bell Laboratories FAQ:

> Plan 9 is itself an operating system; it doesn't run as an application under another system. It was written from the ground up and doesn't include other people's code. Although the OS's interface to applications is strongly influenced by the approach of UNIX, it's not a replacement for UNIX; it is a new design.

NOTE: Visit the Plan 9 FAQ http://www.ecf.toronto.edu/plan9/plan9faq.html.

Despite the fact that Plan 9 is an entirely different operating system, it does retain some of the look

and feel of UNIX. There is still a shell (called rc) and that shell appears much like the popular shells available in most distributions of UNIX. Files, for example, can still be displayed in a UNIX-like long format, along with their attending permissions. Moreover, one can still differentiate between files and directories using the standard -F switch (in fact, many of the stock UNIX commands are available and most of these behave pretty much as they do on a UNIX box). However, the resemblance to UNIX is largely superficial. The underlying operating system works very differently.

One of the chief differences is the way that Plan 9 treats objects (objects in this case being directories, files, processes, and so forth). Under Plan 9, all objects are treated as files. This technique has been implemented in UNIX as well (for example, UNIX treats many devices as files), but not to the extent that it has in Plan 9.

Machines That Run Plan 9

The reported architectures include

- MIPS
- SPARC
- 68020 (NeXT)
- IBM compatibles

It is reported in the Plan 9 from AT&T Bell Laboratories FAQ that various ports are also underway for the following systems:

- SGI Indy
- DEC Alpha
- PowerPC
- DECstation 2100 and 3100

My experience with installing the Plan 9 distribution has been on the IBM compatible platform. As you will see, I went through several generations of hardware before landing on the right combination.

Cross Reference: If you intend to install Plan 9 as a hacking project, you would do well to visit http://www.ecf.toronto.edu/plan9/clone.html. This page describes the hardware that was used at Bell Labs, and will provide you with a nice guideline of hardware that is known to work with Plan 9.

Some Concepts

Plan 9 was designed from the beginning as a networked operating system. As such, the concepts behind it relate more to networking than to the needs of the individual user. Its defining characteristics are the ways in which it handles networking. As noted in the press release for the product:

> The Plan 9 system is based on the concept of distributed computing in a networked, client-server environment. The set of resources available to applications is transparently made accessible everywhere in the distributed system, so that it is irrelevant where the applications are actually running.

> **Cross Reference:** Find the press release from which the preceding paragraph is excerpted at http://www.lucent.com/press/0795/950718.bla.html.

To understand how the Plan 9 system differs from other networked operating systems, examine Figure 21.1.

FIGURE 21.1.

The typical network configuration (without Plan 9).

The typical network configuration (the one most often seen in offices) uses a file server and a series of workstations. Each of the workstations is outfitted with a host of hardware and software (hard disk drives, adequate memory, a windowing system, and so forth) that provides it with the necessary power and networking to connect. System administrators and other, administrative personnel will recognize this setup to be an expensive one.

Because the computer industry has enjoyed tremendous growth (particularly in the last few years), network designs like the one shown in Figure 21.1 are common. Nodes on such networks are usually Pentiums or PowerPCs. You may own such a network yourself. If you do, consider this: Is it necessary that you have such a powerful machine at each node? Or, could it be that this type of configuration is a profligate waste of enormous CPU power? For example, how much CPU power does the accounting department actually require? It depends on what operating system you are running. If you are running DOS-based applications over NetWare, the accounting department doesn't need much power at all. However, if you are running Windows 95, it will need speed and memory.

That speed and memory, by the way, is being eaten purely by features that make your platform prettier and more user friendly. In practice, the average accounting task done in a DOS-based application would be barely noticeable to a Pentium processor. Contrast that with accounting done in Microsoft Excel on the Windows 95 platform. In reality, processor-intensive tasks requiring real power might include tasks like compiling large programs in C++. These tasks, even in a DOS environment, can tax a processor.

So the first point is this: Modern network design wastes processor power by dispersing it, often where it is not most needed. But there are other key disadvantages to this typical network implementation. One is that files are strewn throughout the network, many of them deposited on this or that hard disk drive. How many times have you encountered the following situation:

1. A machine along the network fails.

2. The machine that failed has a file vital to office operations.

3. You recover that file (usually by depositing the hard disk drive from the failed machine into another, operable one, or by performing a restore).

If you have never encountered this situation, consider yourself lucky. I have seen it happen many times. Also, because users often store files locally (on their own workstation), employees must file share and therefore, their machines must always trust each other.

Plan 9 takes a totally different approach. In Plan 9, the jobs of processing and file storage and separated, as are the machines that perform these tasks (see Figure 21.2).

FIGURE 21.2.

The Plan 9 networking concept.

Note the CPU server in Figure 21.2. This would typically be a very powerful machine (probably multiprocessor) that would provide CPU services to remote workstations or terminals. This is complemented by a file server.

This system has some important advantages. First, there is centralized control of files. This has obvious security advantages. Centralized file control also allows easier management of files. Moreover, it provides an environment in which permissions may be easily viewed and alteration of files may be more readily detected.

Also (though this has little to do with security), as mentioned in the Plan 9 documentation, this centralized file management is of benefit to a programming team. Project management is more easily accomplished and the system offers a sense of community for the programming team.

Moreover, the Plan 9 system performs without a root operator. Users must be authenticated to gain access to privileged files or processes, and this authentication has been described as similar to authentication using MIT's Kerberos. Kerberos is a method of authenticating network connections and requests. To perform this authentication, Kerberos examines secret, ciphered keys belonging to the user. Passwords in Plan 9 are therefore never passed across the network. This greatly enhances the security of the operating system. Moreover, user programs are reportedly never run as processes on the file server, and processes that are run belong to the individual user. Root does not exist.

Cross Reference: To examine the internal workings of Kerberos (and the procedural execution of authentication), visit http://www.pdc.kth.se/kth-krb/doc/kth-krb_2.html.

Discarding the concept of root was an excellent idea. The majority of serious cracking techniques rely on exploiting programming weaknesses in processes that run as root. In Plan 9, there is no root, and therefore, no such processes.

NOTE: To my knowledge, there has not yet been extensive focus on Plan 9 security outside the confines of Lucent Technologies (previously AT&T). Therefore, it is not known

whether there are security flaws inherent in Plan 9's design. The only thing that qualifies as a known bug at this point (those who use it have thus far been pretty quiet) is that users can sometimes log in as user none from a remote connection. I suspect that in the future, as Plan 9 becomes more well known, various attacks will be instituted against the system and bugs will likely surface.

In short, Plan 9 security is an area yet to be explored. Nonetheless, in terms of its network implementation and its basic design, Plan 9 already presents significant roadblocks to the cracker. Certainly, typical advanced techniques of attacking UNIX servers will probably fail when implemented against Plan 9. The future, however, remains to be seen.

NOTE: I cannot stress the importance of the concept of life without root. Nearly all operating systems evaluated as secure maintain some concept of a root operator. The concept of root concentrates all the power in a single place on the system. If a cracker can gain root, the struggle of penetrating security is over. However, I should quickly point out that the absence of root on a Plan 9 system does not mean that an administrator is not needed. In fact, in certain respects, Plan 9 transforms the job of system administrator into one of architect. In short, Plan 9 is designed for vast--if not massive--management of network resources. Although still in the experimental stages, Plan 9 could change the architecture of the Internet and with it, the concepts surrounding acceptable security policies and implementations.

Applications Available for Plan 9

Admittedly, there are few native Plan 9 applications, but the list is growing. Remember: Plan 9 is an entirely new operating system, so the number of applications depends on how many individuals actually use the system.

NOTE: A caveat: The licensing restrictions set forth by Bell Labs makes it very difficult to create commercial applications for Plan 9. The licensing scheme has cast Plan 9 into a position of being available only from Bell Labs at a high price and without hope in the near future of complete commercialization. For now, therefore, Plan 9 remains largely under the purview of researchers and hobbyists who are willing to shell out $300 for the system and documentation. Many freelance programmers protest this situation, arguing that Plan 9 ought to have licensing restrictions similar to those that apply to Linux. After all, why would someone develop on a platform that may ultimately be barred from commercialization? The answer is this: People would undertake such development for the pure pleasure of discovering and hacking a new system. However, many hobbyists are unwilling to pay the stiff licensing fee for the entire system.

Despite some licensing restrictions with Plan 9, some important applications have already been written:

- An HTTPD server
- Text editors
- A version of MIT's X Window system

Moreover, the basic distribution of Plan 9 comes with a wide range of utilities and even a native Web browser called *Mothra*. Mothra may not be as colorful and user friendly as Netscape Navigator or Microsoft Internet Explorer, but it works quickly and efficiently. In fact, Plan 9 possesses few user-friendly features. It is a largely text-based environment. It is more practical than attractive, and many elements of its design are of significant interest to programmers.

SAM

The most prominent native application for Plan 9 is the SAM editor, which is a straight ASCII text editor with a twist. It has a command language that can be used to process repetitive tasks (much like a macro language, I suppose, but a bit more defined). On the surface, SAM operates in much the same way UNIX-based text editors. File names (single or multiple) can be specified on the command line. This loads the file(s) into a windowed area. There, the text can be clipped, pasted, cut, altered, edited, and saved.

Like most UNIX-based text editors, SAM does not support multiple fonts, style sheets, or other amenities common to modern word-processing environments. In fact, Plan 9 would be a poor choice for anyone who relies on such amenities, for they do not exist within the system at this time.

The SAM command language operates mainly on regular expressions and is suitable for inserting, deleting, finding, replacing, and so on. These functions are generally called by a single character, followed by your intended text. In other words, the text to be found, replaced and so forth.

In short, SAM appears very bare bones, but really isn't. Learning the SAM command language takes a day or so, although you might need several weeks to become proficient.

Plan 9's Window System

Plan 9 has a window system called 8 1/2. After the system boots, it asks whether you want to load the window system. If you choose this option, 8 1/2 appears. On first examination, 8 1/2 looks extremely rudimentary (far more so than X, even). The opening screen presents one term window and a clock. Navigation is done largely with the mouse.

TIP: To fully utilize the 8 1/2 windowing system, you need a three-button mouse. A two-button mouse will work, but you will lack at least one menu and some serious functionality.

To size a window, click any portion of the blank screen. This invokes a menu with options including Size, Move, Delete, and Hide. After you choose an option, click the target window. For both the hide

and delete functions, the window behaves as it would in X; it disappears or is deleted. However, for the move and size functions, you must work a little differently. After choosing the menu option, click the window once. Then, instead of directly sizing or moving the window, click the black screen again (this time with the right mouse button) and redraw your window. This may initially seem awkward, but you'll get used to it.

8 1/2 is extremely lightweight. Even on a machine with 8MB of RAM, 8 1/2 responds quickly. On a Pentium 133 with 64MB of RAM, 8 1/2 is incredibly fast.

8 1/2 is more dynamic than most other windowing systems. You can grab any text anywhere and use it as a command. In this regard, 8 1/2 could be called the ultimate cut-and-paste system. Text identifying objects (which is often read only on other platforms) can be grabbed at any point and dropped into any other part of 8 1/2. In fact, this feature is so prominent that new users may find themselves grabbing things without even knowing it. In addition, as part of this functionality, the cursor can be placed anywhere within a window. This is a significant change. Users of X and Microsoft Windows alike will find this feature to be fascinating. For example, although you can cut and paste from an XTERM or a MS-DOS windowed prompt, you cannot arbitrarily drop the cursor in any area and pick up typing again. In 8 1/2, you can.

I suppose 8 1/2 can best be described as a window system optimized for programmers. Code and other data can be moved at any time from any position. But perhaps the most fascinating thing about 8 1/2 is that you can recursively run an instance of 8 1/2 within an 8 1/2 window. To my knowledge, this functionality is indigenous only to 8 1/2. No other windowing system can perform such a task. (Funny. Although this is an extraordinary feature, I have not yet encountered a reason to use it.)

The learning curve on 8 1/2 amounts to a day or two at most. If you are familiar with any implementation of X (or more directly, if you have ever used SunView), learning 8 1/2 will be simple. To some extent, 8 1/2 reminds me of SunView.

NOTE: SunView, a windowing system introduced in early versions of SunOS (the operating system created by Sun Microsystems), is extremely lightweight and, even on SunOS 4.1.3, is faster than Sun's later windowing system, OpenWindows. OpenWindows is enormously popular among Sun users, although it is perhaps slower--and not as visually stunning--as the Common Desktop Environment (CDE), a new windowing system jointly developed by many UNIX vendors.

Programming in Plan 9

Ultimately, I would recommend Plan 9. If you are a programmer and are looking for a new and exciting operating system to develop on, Plan 9 is for you. It is exciting primarily because of its unusual design. And, although it is not UNIX, it has enough UNIXisms that UNIX users can hit the

ground running. Moreover, the unique networking capabilities of Plan 9 present new opportunities for programmers.

Programming in Plan 9 is not limited to C, though the real Plan 9 distribution does come with a native C compiler. This C compiler is designed to accommodate code for all the supported architectures, including (but probably not limited to)

- IBM (Intel X86)
- SPARC
- 68020
- MIPS

The compiler accepts straight ANSI C, but be forewarned: If you have avoided learning some of the newer conventions, you may encounter difficulties. Rob Pike has written a paper describing Plan 9 C compiler usage. I highly recommend reading that paper in its entirety before attempting to code any serious application on the Plan 9 platform.

Cross Reference: Rob Pike's paper, titled "How to Use the Plan 9 C Compiler," can be found online at http://kbs.cs.tu-berlin.de/~jutta/c/plan9c.html.

If you plan to concentrate on porting applications to or from Plan 9, check out the ANSI-POSIX Environment (APE). The APE features a wide range of POSIX tools.

Cross Reference: An excellent technical overview of APE written by Howard Trickey ("APE--The ANSI/POSIX Environment") can be found online at http://plan9.bell-labs.com /plan9/doc/ape.html.

NOTE: POSIX stands for Portable Operating System Interface, a standard that has been in the works for many years. This standard is an effort on the part of developers to establish a unified UNIX. In other words, if a program is fully POSIX compliant, it should run on any fully POSIX-compliant platform, be it SunOS, Linux, AIX, HP-UX, or other versions of UNIX. For many years (and even now) there have been both sharp and subtle differences between various UNIX platforms that prevent easy porting of applications. The POSIX standard will likely change that. To learn more about POSIX, visit http://csrc.ncsl.nist.gov/nistbul/csl91-10.txt.

Garden-variety throwaway programming can also be done in rc, the shell environment of Plan 9.

The complete distribution of Plan 9 also comes with extensive libraries, including one for the development of windowed applications intended to run within 8 1/2 (windows are actually referred to as *panels*). Tom Duff has written quite a good paper on the development of panels in Plan 9. He likens the panel library to the popular development packages Tcl and Tk. Linux users will be familiar

with Tcl and Tk. Both are languages (and development libraries) for use in generating X Window System applications. One of the most popular features about Tk is that you can build a windowed application using a language very similar to a macro language. It is possible to quickly develop applications using these tools because objects within the window environment are placed by the use of direct statements. For example, the command within a Tk script to create a button is:

```
button .name_of_button <options>
pack .name_of_button <options>
```

Similarly, the development syntax for windowed applications intended for use in Plan 9 is reduced to direct statements (although this is still done using basic C).

Cross Reference: If you are interested in developing windowed applications in Plan 9, Tom Duff's paper, "A Quick Introduction to the Panel Library," can be found online at ftp://plan9.bell-labs.com/plan9/doc/panel.html.

Another interesting aspect of Plan 9 is its inclusion of the Alef programming language. Alef is a relatively new language. Unfortunately, a discussion about the Alef language is beyond the scope of this book (that is a delicate way of saying that I know too little about Alef to provide you with quality information).

Cross Reference: To find more information about Alef, check out the language reference manual. It can be found online at http://plan9.bell-labs.com/plan9/doc/ref.html.

Another important resource (probably even more valuable than the language reference manual, especially for the newcomer to Alef) is the Alef mailing list. It can be viewed online at http://plan9.wtf.nyc.ny.us/1996/0001.html.

In all, Plan 9 is a very rich development environment. One thing that makes it especially exciting is that the field is wide open. It is a whole new operating system just waiting for new and useful applications to be written.

Installing the PC Distribution

You might want to try out a working demo of Plan 9; Bell Labs has generously provided such a distribution free of charge on the Internet. However, before you install Plan 9, there are some things you should consider. One is whether you should even begin. As with any new operating system, you should expect bad things to happen. These may range from simple installation failures to data or disk corruption.

NOTE: If such events occur, it is most likely because you are using unsupported hardware. When the right hardware is used, Plan 9 works beautifully. Of course, not everyone has the money to go out and acquire the exact hardware used at Bell Labs. If

your cash is limited, expect a certain amount of trouble along the way.

Moreover, you'll experience a high level of frustration. If you have ever installed an operating system that is less than user friendly, you know the terrain. Cryptic errors may appear, and you will undoubtedly be forced to hack here and there. These obstacles will be particularly prominent if you are installing the four disk, free PC distribution on dubious hardware.

The Machine Targeted for Installation

The machine that you use should be free and clear. That is, under no circumstances should you install the Plan 9 distribution on a machine that you rely on to make a living (even if you are installing Plan 9 on a separate disk). There have been instances in which Plan 9 installations have permanently disabled the boot capabilities of other operating systems or partitions. The only reason you should make such an installation is if your job requires it (or if you are a programmer who loves adversity). Otherwise, use that old DX66 you have lying around in the closet.

If you only have one machine and still want to experiment, get a rack-mount disk changer. This device allows you to switch hard disk drives quickly and easily. It works in exactly the same fashion as a slide-out car stereo. Your hard disk drive is secured in an enclosure that slides into your PC tower. In this manner, you can switch from your regular disk drive (containing the operating system you use for work) to the Plan 9 drive.

NOTE: To my knowledge, there have been no known instances of Plan 9 installations damaging hardware, so there is no reason why you should fear temporarily switching disks.

The machine need not be particularly fast, though I would recommend 66mHz or better. I have installed the distribution on a DX33mHz, a DX266mHz, a DX4120mHz, and a 133 Pentium. To be honest, I did not find an incredible increase in speed between the 120 and the 133, nor did I find the difference between the 66 and the 120 unbearable. However, the DX33 was admittedly quite slow.

Memory is important. You will need at least 8MB. Some documents on the Internet suggest that there are individuals running Plan 9 with 4MB of RAM, and I believe it. However, of the seven times that I installed the PC distribution, I was twice confronted with an Out of Physical Memory message. This was, as it happens, a fatal error. Immediately following this message, the installation failed. On both occasions, I was using only 8MB of RAM. On an identical machine, after installing additional RAM, I managed to complete the installation successfully.

The Hard Disk Drive

What hard disk drive you use depends on what you are installing. If you are simply installing the PC distribution, you can successfully use a 40MB hard disk drive. However, if you intend to install the entire Plan 9 distribution from CD-ROM, you need a hard disk drive equal to or greater than 540MB.

TIP: It has been reported in documentation that a 500MB disk will suffice. As far as I can tell, this is not entirely accurate. If you make a direct installation from the CD-ROM, you will require a 540MB disk (or approximately 532MB). The only way to get around this is to first install the basic four-disk PC distribution, and then more incisively install the remainder of the distribution from CD-ROM. This eliminates many items that are intended for use on other platforms. Be advised, however, that this is a more difficult path and may result in problems getting your CD-ROM to catch. (Sometimes, the CD-ROM driver does not properly initialize the CD-ROM drive.) It is a much better idea to make the full install and later delete what you do not need.

I recommend a 540 or 600MB drive. I should state that my installations were performed entirely with IDE drives and therefore, I cannot give background on SCSI-based installations. I obtained suitable results with the following drives:

- MiniScribe 8051A (41MB) (Don't laugh. It worked flawlessly.)
- Conner CFS540A (EZD03) (540MB)
- Quantum Maverick ProDrive #MV54AO11 (514MB)
- Maxtor 7245AT (243MB)

The Installation Process

I will assume that you are installing the four-disk PC distribution. This set of four diskette images is located online at ftp://plan9.bell-labs.com/plan9/pcdist/README.html. Download these into a temporary directory on your current hard disk drive.

NOTE: The term *diskette images* refers to four files located at the Plan 9 site. These files are exact copies of four diskettes required to make an installation set for Plan 9. These files or images must be downloaded to your local machine and written to floppy diskettes.

TIP: It is important that you obtain the disks from this site. Earlier versions of these boot disks may be available at other locations, but you should not use them. On installation, earlier versions have a tendency to damage other partitions on the disk drive. A typical example would be where Plan 9 disabled a Linux partition during the installation process. Again, I strongly advise against installing Plan 9 on any hard disk drive that contains vital or irreplaceable information.

The diskette images at that location are

- disk1
- disk2.vd
- disk3.vd

- disk4.vd

After you download these diskette images, write them directly to a floppy.

> **NOTE:** Copying the diskette images to a floppy involves a process that is different from copying files to a floppy. Copying the diskette images directly to floppy will not suffice and will result in an installation failure.

A number of utilities for writing disk images to floppy disks are available. The most popular is a program called DD.EXE. A DOS version (suitable for use under Windows 95) is available on-line at

- http://access1.sun.com/drivers/utilities/dd.exe

Another utility used for this purpose (and one that is more widely available on the Internet) is RAWRITE.EXE. Linux users will be familiar with RAWRITE.EXE because it used to write the boot diskette images for Linux to a floppy. RAWRITE.EXE is available online at

- ftp://sunsite.unc.edu/pub/Linux/distributions/slackware/install/RAWRITE.EXE

After you write these images to a floppy, switch to the target machine (or hard disk drive). On the hard disk drive of the target disk, you must establish a DOS partition. The size of this partition is not particularly important (I use 10MB). It really only needs to be large enough to hold DOS and approximately 1.5MB of information in a directory called C:\PLAN9.

Partitioning the Disk

To partition the disk, use the FDISK.EXE utility (see Figure 21.3).

FIGURE 21.3.
The FDISK utility.

First, delete all partitions (you will be starting over with an entirely clean disk). Allocate whatever space you intend for the DOS area (I recommend at least 11MB). After you reboot your machine, you will format this partition and install DOS onto it.

> **NOTE:** At this stage, you should have a hard disk drive with one DOS partition at least 11MB in size. The rest of your hard disk drive should not contain any other type of partition.

Installing the Basic Plan 9 System

The remaining steps of this portion of the installation are simple. Boot with the Plan 9 boot diskette (this is the diskette to which you wrote the disk1 file). During the boot process, you will see a series of

messages; most are easy to read and interpret. Plan 9 tracks your memory and reports what portions of it are available. It identifies your hardware. Ultimately, it brings you to a nice blue screen, which contains the heading *System Installation & Configuration*.

From this screen, you perform the installation. Pressing the Enter key invokes a menu containing various installation options. Ensure that these are correct for your machine. When you are satisfied that these options are correct, choose the Install option.

A window will appear, and many filenames will scroll past. Do not be alarmed. This is Plan 9's way of telling you which files were installed. When this process is complete, Plan 9 interrogates you about your hardware. Specifically, options must be set for your VGA, mouse, and so forth. Provide the necessary answers and choose Save Configuration from the menu.

NOTE: Plan 9 gives you an opportunity to change the options before they are committed to the disk. A dialog box appears, listing your options. If they are incorrect, go back and change them.

After the options have been saved, you can remove the floppy disk and reboot the machine. Your DOS should boot normally. As mentioned previously, almost all problems with this installation procedure occur at time of booting disk1. To my knowledge, there have been few instances of problems occurring on the reboot to DOS.

Installing the Remaining Diskette Files

After you reboot your machine, change your current directory to C:\PLAN9 and type the letter B to load an installation program. You must define the target disk drive. This tells Plan 9 in which disk you intend to house the Plan 9 file system. I am assuming here that you have a single disk, so this is not an issue. But if you are installing to a machine with multiple disks (and partitions), take extra care to ensure that it is the correct partition.

WARNING: If you choose the incorrect partition, all your data on that partition will be lost. Be *absolutely certain* that you have chosen the correct partition before you commit to the install procedure.

After you choose your partition, Plan 9 begins the file system installation process. A dialog box appears, prompting you to insert the second disk into the floppy disk drive. After you do so, the installation of system files commences. You will again see a pop-up window with filenames scrolling by. Insert disks as the program requests them. When the installation procedure is complete, you will be prompted by a menu. Choose the option that says Make the newly installed Plan 9 the default.

Congratulations. At this stage, the installation procedure has been completed. You can now remove the fourth disk, reboot your machine (choose this menu option), and begin to explore Plan 9.

Starting Plan 9

When you machine reboots, change your current directory to Plan 9. There, type the command

```
B
```

This loads the Plan 9 system. First, you will see many of the same messages you saw when you booted with the boot floppy (Disk1). The system will identify your hardware, count your memory, and describe some resources. When it is finished, a line that looks like the following will be printed to your terminal:

```
root is from (local, 9600, 19200, il)[local!#H/hd0fs]:
```

Press the Enter key. You will then be prompted with a line that looks like this:

```
user[none]:
```

Press the Enter key again. Depending on the speed of your system, you will soon be asked whether Plan 9 should start the window system. Choose Yes.

If you have gotten this far, 8 1/2 should load cleanly (barring some obscure error). If so, you will be presented with a bright screen (its color may depend on your video card; on mine, the screen is white), a clock, and a window with the contents of the README file printed within it. Welcome to Plan 9. (Look at the size of that mouse cursor!)

Navigating the file system works much like it does in UNIX. You can use the CD command to get from directory to directory. Here are some minor notes that may assist you on your first test drive of Plan 9:

- The -a option in the ls utility is not supported. You are not crazy, it is simply not available in Plan 9.
- The -F flag is supported in Plan 9. However, the output appears a bit differently than in UNIX. The character that denotes a directory is found on the extreme left of the table, not the right. It is easy to miss.
- If you are a big fan of the MORE utility, you may be disappointed. Use CAT instead.
- If you want to get an overall look at the system, start ACME. This utility works a similarly to a file manager, revealing files and directories. These can be opened within the ACME environment.
- DF is not supported, but DU is.

Summary

Plan 9 from Bell Labs is a new and unique operating system that offers an entirely new outlook on Internet security. Still, it is far from supplanting modern UNIX. (In fact, many people believe that the next commercially viable operating system will be Inferno from Lucent. To some extent, this is true, because Inferno is now being used in set-top boxes for interactive TV.)

Plan 9 seems best suited for very large organizations. The system appears to have been designed expressly with the Internet in mind and on a grand scale. I would call it some finely crafted mortar with which to seal the cracks in modern networking techniques. Because there is little research available on the security model of Plan 9, we must simply wait and see. However, as a hacking project, I cannot think of a better start than Plan 9 from Bell Labs.

Resources

Following are several resources on Plan 9, including sites on the WWW, papers, and mailing lists (of course, some may change by the time you read this book). It seems that the Plan 9 computing base is a small (but growing) faction, making information on this new operating system scarce.

Plan 9 on the WWW

The Plan 9 Server Document Directory

- ftp://plan9.bell-labs.com/plan9/doc/

The Plan 9 PC Diskette Installation Distribution

- ftp://plan9.bell-labs.com/plan9/pcdist/

The Plan 9 FAQ

- http://plan9.bell-labs.com/plan9/faq.html

Technical Documentation on Plan 9 (Papers)

- http://plan9.bell-labs.com/plan9/vol2.html

Overview of the Plan 9 File System

- http://www.emrtc.nmt.edu/~mikebaz/plan9.html

The Plan 9 Mailing List Archive

- ftp://ftp.cse.psu.edu/pub/plan9-fans/

Online Bibliography on Plan 9

- http://liinwww.ira.uka.de/bibliography/Os/plan9.html

The Plan 9 Usenet Newsgroup

- news:comp.os.plan9

The Unofficial Plan 9 Page (Very Good)

- http://www.ecf.toronto.edu/plan9/

Plan 9 Web Server at the University of York (UK)

- http://www.plan9.cs.york.ac.uk/

Official Plan 9 Home Page

- http://plan9.bell-labs.com/plan9/index.html

Articles and Such

Plan 9: Son of UNIX. Robert Richardson. *LAN Magazine* (Volume 11, Page 41). August 1, 1996.

Plan 9: Feature Film to Feature-Rich OS. Paul Fillinich. *Byte Magazine* (Volume 21, Page 143). March 1, 1996.

Plan 9 from AT&T. David Bailey. *UNIX Review* (Volume 1, Page 27). January 1, 1996.

Plan 9 from Bell Labs. Rob Pike, Dave Presotto, and Phil Winterbottom. *Computing Systems Journal* (Volume 8, Page 221). Summer, 1995.

Plan 9. Sean Dorward, Rob Pike, and Dave Presotto. *UNIX Review* (Volume 10, Page 28). April 1, 1992.

Designing Plan 9. Rob Pike, Dave Presotto, and Ken Thompson. *Dr. Dobb's Journal* (Volume 16, Page 49). January 1, 1991.

Is Plan 9 Sci-Fi or UNIX for the Future? Anke Goos. *UNIX World* (Volume 7, Page 61). October 1, 1990.

Previous chapter Next chapter Contents