A Gentle Introduction to Plan 9

By James Choate, January 07, 2004

Post a Comment

The current superiority of Unix and its derivatives such as Linux has a history that goes back to the 1960s. The original intent of Unix was to move use away from many people per machine to a few people per machine. As usage of computers has increased, however, we're now seeing many machines per person. Consequently, a variety of issues arise that are related to scaling: access, data, applications, security, upgrades, etc. These issues are problematic in the sense that trying to keep a user's work synchronized across many machines requires more and more work. One example is the current suggestion that users should patch their systems, including all applications, on a weekly basis. This process invariably takes some level of human intervention and, for several machines, may take several hours of user effort. This time could be better spent actually making product or providing a service. Thus, the question arises: "Is there a better way?"

A Gentle Introduction to Plan 9

James Choate and Joe Wessels

The current superiority of Unix and its derivatives such as Linux has a history that goes back to the 1960s. The original intent of Unix was to move use away from many people per machine to a few people per machine. As usage of computers has increased, however, we're now seeing many machines per person. Consequently, a variety of issues arise that are related to scaling: access, data, applications, security, upgrades, etc. These issues are problematic in the sense that trying to keep a user's work synchronized across many machines requires more and more work. One example is the current suggestion that users should patch their systems, including all applications, on a weekly basis. This process invariably takes some level of human intervention and, for several machines, may take several hours of user effort. This time could be better spent actually making product or providing a service. Thus, the question arises: "Is there a better way?"

Background

The original authors of Unix began to address this question in the early 1980s. They began looking at what was wrong with the Unix model, and the result was two operating systems -- Plan 9 and Inferno. In this article, we will describe Plan 9.

Three aspects of Plan 9 make it stand out: per process namespaces, creating pools of processes and file servers, and differentiating the I/O and process aspects of the user's experience. A typical Plan 9 configuration would consist of terminal server, a group of namespace servers, and a cloud of process servers (some of which run authorization services).

A terminal server is a machine that users can use to access their Plan 9 resources from anywhere they can get a network connection

to their authorization server. Think of it as a laptop with good display characteristics but limited CPU resources. This setup provides a level of "global sign-on" that other operating systems strive for, but Plan 9 users get out of the box.

Although Unix derivatives try to make many things appear like a filesystem, there are still many issues that break this model within the Unix approach (e.g., ioctl). Plan 9 has tried to address this so that all resources appear in a file tree and can be shared across many users. This system makes the directory tree, or namespace, seen by each process potentially unique. When a process is started, it inherits the default namespace unless it is assigned an alternate. When this ability is coupled with the sharing of foreign resources as namespaces, you begin to see what is possible. For more information about customizing namespaces on a per process basis, please refer to the **attach()** and **bind()** man pages. Another aspect of this is the "install once, run everywhere" capability, which promises to significantly reduce maintenance costs.

VoIP is a common application with great commercial potential. A lot of time and resources have been spent developing it, yet we still don't have a clean solution. Under Plan 9, it is possible to share a sound card and microphone with many other users and do the equivalent of cat'ing a data stream to many devices. The same sort of thing can be done with /dev/display such that a user can output to a variety of displays with nothing more than a simple list of devices. Another example could be musicians sharing MIDI streams for truly distributed composition and performance. With Plan 9, if you can talk to one namespace, you can talk to every namespace -- for which you have authorization, that is.

With respect to clouds of process servers, Plan 9 is way ahead of the pack in doing parallel distributed processing. The mechanism for this is the authorization server, which is a regular CPU server that has an authorization process and database running. If users have the appropriate credentials, they can use any machine under that authorization server. This authorization server also controls who can access file servers.

An interesting thing about Plan 9 with respect to process execution on hardware is that it's possible to assign a program to a specific machine or create a pool and allow the process to select from that pool without user intervention. This allows administrators to address "trusted computing" in a potentially lucrative way. Consider a small business that has several different business processes going on. Some of these processes are not critical or sensitive to public release, and others might have a variety of levels. You could create a small pool of processors internally for the most sensitive applications, lease process and namespace resources from another party with the proviso that what gets run doesn't go any farther than the user and service provider, as well as create completely public resource pools that might be provided by user groups or other organizations.

Installation

The functionality of Plan 9 is available to you even if you don't have a group of machines with which to experiment. You can put all this functionality on a single machine. The server we'll describe in this article does not use the normal Plan 9 filesystems but rather the kfs filesystem that is the default for bringing up I/O and CPU servers. The kfs filesystem is a very light filesystem and should not be used for production work; however, it is fine for learning.

To obtain all the initial software, go to the Plan 9 homepage:

```
http://plan9.bell-labs.com
```

To begin, we recommend reviewing the supported hardware. Plan 9 is very sensitive to differences in video and network, with video being problematic, so you might want to have several different video and network cards just in case. The online documentation is several years out of date and contains numerous errors. However, Hangar 18 (a Plan 9 user group) has winnowed through the various documents and produced a how-to that describes the installation process in much more detail. Besides getting a copy of the Plan 9 boot floppy and the current iso.bz2 image, you should also get all the other packages related to Plan 9. Also, you will want to bunzip the compressed .iso. Put the files into a single directory and burn the CD. We will not cover the additional material in this article, but you'll want it if you stick with Plan 9. The minimal set of files that we suggest are the .iso.bz2, the .iso, and the .flp files. Once you have all the

software on the CD, copy the 9disk.flp to a floppy. In Linux, you will use **dd** for this; on Windows, you will use rawrite.exe. Refer to the documentation for the relevant utility for command-line parameters. Once you have a floppy and a CD, you are ready to begin your installation. Reboot the machine and follow the prompts on the display. If all goes as expected and your video card is a recognized type, you should see a graphical desktop with three windows (see Figure 1). Plan 9 is fundamentally a graphical system; it doesn't even have the concept of a terminal as one would expect. While this may at first seem strange, it turns out to be very useful. Next, prepare the hard drive MBR to ensure support for large-sized drives (e.g., 80 GB). We suggest always using this MBR rather than the default one, which limits the size of drives to only a few gigabytes. To open a new command window, push the right mouse button and select "new". Your point will turn to a cross or plus sign. Move the cursor to the upper left of the screen and hold the right mouse button down. Don't click it; hold it down. Sweep or drag the pointer into lower right of the display. When you release, you'll have a new command window (Figure 2). Enter the following:

```
disk/mbr -m mbr /dev/sdC0/data
```

Note that the drive identifier may be different for your machine if the drive is on a controller other than the first one. You'll then reboot the machine in one of two ways.

- Ctl-Alt-Del (i.e., 3-finger salute)
- disk/kfscmd halt

You should find the installation engine at a prompt that is asking to partition the drive. Select the hard drive you want to use for the installation; it should be the same one you put the MBR on previously. The next stage will be to actually create the partitions. You should have a screen with a suggested partition layout. Accept it by first saving the data to the drive. You should then find yourself at a prompt for mounting the filesystems. You will be asked to select the appropriate partition in which Plan 9 will be installed and whether you want to keep the existing filesystem or wipe it. We suggest wiping it (Figure 3); ignore the error messages. After this, you must tell the installation engine where to get its images -- either the network or local. We assume off your CD drive. The display should update to a prompt that is concerned with mounting disks. Now the process scans all the devices on the system for potential images. This will result in another prompt that allows you to browse the selected device for the specific image from which you want to install (Figure 4). Upon completion, the next screen will be related to copying your distribution image. If all went well copying the image to the drive, you should see a bootsetup prompt. Next will be the selection on how you want the machine to boot. Select "plan9" on your new Plan 9 installation. Assuming no problems, your next prompt will be "finish". Assuming there are no floppies in the machine and that it's configured to boot from the local drive, the first thing you'll see after normal POST is "PBS...Plan 9 from Bell Labs". This indicates you've begun to boot Plan 9. The first prompt is related to where you want the machine to boot from and will appear on every boot. It should be set to some string starting with "local" and specify a particular device to use (e.g., local!#S/sdC0/fs). It will ask for a user to boot the machine as -- use "glenda". "glenda" is the Plan 9 equivalent to a root user. There are significant differences in the way that Plan 9 deals with these sorts of users as compared to other operating systems. We refer you to the Plan 9 documentation for further clarification on these issues. As the machine

2014
Dr. Dobb's Journal
November - **Mobile Development**
August - **Web Development**
May - **Testing**
February - **Languages**

Dr. Dobb's Tech Digest
DevOps
Open Source
Windows and .NET programming
The Design of Messaging Middleware and 10 Tips from Tech Writers
Parallel Array Operations in Java 8 and Android on x86: Java Native Interface and the Android Native Development Kit

2013
January - **Mobile Development**
February - **Parallel Programming**
March - **Windows Programming**
April - **Programming Languages**
May - **Web Development**
June - **Database Development**
July - **Testing**
August - **Debugging and Defect Management**
September - **Version Control**
October - **DevOps**
November- **Really Big Data**
December - **Design**

2012
January - **C & C++**
February - **Parallel Programming**
March - **Microsoft Technologies**
April - **Mobile Development**
May - **Database Programming**
June - **Web Development**
July - **Security**
August - **ALM & Development Tools**
September - **Cloud & Web Development**
October - **JVM Languages**
November - **Testing**
December - **DevOps**

boots, you should see a graphical interface come up (see Figure 5). We expect to see a system monitor window in the upper left. To its right is the mail program. There should be a small image of a bunny; this is glenda. You should then see two windows on top of each other, with pieces of each window visible. The topmost window is a Plan 9 shell window, called "rc". You should take the time to read this window; it is a quick introduction to using the windowing system, "rio". Left-click on the window at the bottom of the display; this is "acme", a very cool program that combines various features of shells, text editors, and directory browser. **Configuring Your System** The next step is configuring your system, followed by a kernel recompile so that it has the authorization services. This makes it safe to put on the network. You need to know explicitly the following:

- Ethernet MAC
- IP and Netmask (e.g., 192.68.23.1 & 255.255.255.0)
- FQDN Hostname (e.g., saucer.ssz.com)
- Default Route by IP (e.g., 192.68.23.9)
- DNS Nameserver by IP (e.g., 192.68.23.25)

Contact your network admin or ISP technical support if you don't have this information or don't know how to get it. There is a condition related to command output (e.g., ping and man) where a process will halt sending output to the display window until the "down" cursor key is struck; this function is called "scroll". This forces an update and the command output should continue normally until it either finishes or fills up the screen again. This can cause unexpected delays for the unwary. You gain access to this through the middle mouse button; it is at the bottom of the menu. We normally leave ours set to "unscroll". Now, you need to create a user, "tor". (Hint: Replace "tor" with your userid): **disk/kfscmd allow disk/kfscmd 'newuser tor'** -- Creates a new user named "tor" in adm/users **cat /adm/users** -- Verify the new account is added **disk/kfscmd user** -- Makes KFS re-read adm/usrs **disk/kfscmd 'create /mail/box/tor tor upas 557 d'** -- Creates tor's inbound mailbox dir **disk/kfscmd 'create /mail/box/tor/mbox tor upas 622 al'** -- Creates tor's inbound mailbox file **disk/kfscmd disallow** -- Disables making changes to the filesystem One of the first things to learn is how Plan 9 combines the concepts of user and group. Read **user**(6) for an explanation. Next, you must configure the basic login for each user. The basic process is: 1. Boot machine. 2. Log in to the local filesystem. 3. Log in as a new user. You'll get an rc prompt (i.e., "%"); note that you can run **acme** and other commands at this point. 4. Execute **/sys/lib/newuser**, which will configure your new desktop. A basic desktop should come up -- a gray screen and mouse pointer and *nothing* else. 5. Log out; you can either use ctl-alt-del or open a new window and use **disk/kfdcmd halt**. You'll need to be familiar with the basic tools to edit system files as well as the basic syntax of the system files. These documents are available on the Plan 9 homepage. Read the papers and appropriate man pages for: **sam**, **acme**, and **rc**. After playing around a bit, you may find that the standard 640x480x8 desktop can be a little crowded. The key file is 9fan.ini, so read **9fan.ini**(8) for additional information. Here's how to change the resolution for a system booting from a disk:

```
disk/kfscmd allow
9fat:
cd /n/9fat
acme plan9.ini
```

Scroll to the bottom where you will see:

```
vga=640x480x8
```

Change it to:

```
1024x768x8
```

or to this:

```
vga=ask
```

and it will ask you to set resolution each boot. This is very useful if you're testing drivers or new hardware. Next, we'll configure the network, so reboot and log in as "glenda". At this point, you must decide whether you are going to use DHCP or local configuration. If you're using DHCP:

```
disk/kfscmd allow
cp /bin/termrc $home/termrc.org
cp /bin/cpurc $home/cpurc.org
cp /lib/ndb/local $home/lib.ndb.local.org
cp /lib/ndb/local.complicated $home/lib.ndb.comp.org
cp /lib/ndb/local.complicated /lib/ndb/local
acme /bin/termrc
```

Look for a comment about DHCP (comments start with #):

```
#If you're system has DHCP
```

Just below it are two lines that must be uncommented (i.e., remove the #). The result should be something like:

```
if(! test -e /net/ipifc/0/ctl)
     ip/ipconfig > /dev/null >[2=1]
```

1. Save the modified file to /bin/termrc. 2. Reboot system. 3. Log in as glenda. 4. ip/ping {IP of your DHCP server}, note that ip/ping makes 32 attempts. 5. ip/ping www.whitehouse.gov; this verifies that you're getting off your local network and onto the Internet (it's ok to use another FQDN). If you're using a dedicated IP, the amount of editing is much larger. We've provided a working example of the various files for a machine that doesn't use DHCP. **The bootes Account** Next, we'll recompile the kernel and modify the boot sequence to enable the authorization functions. "bootes" is one of the most important system accounts (besides glenda), because it boots the basic auth server by default. The bootes account is installed during the basic installation, but it is not configured. 1. Log in as glenda. 2. Create bootes' mail; use the same two commands you used to create your own mailbox. 3. Log out and log back in as bootes. 4. Create the homedir /usr/bootes, as follows:

```
disk/kfscmd allow
disk/kfscmd 'create /usr/bootes bootes bootes 775 d'
ls -l /usr
```

5. Configure bootes' desktop:

```
/sys/lib/newuser
```

6. Log out. Edit the authorization configuration file, /bin/cpurc, so that your file looks similar to this example. Save this file to /bin/cpurc and /rc/bin/cpurc. To allow bootes to run as other users:

```
cd /lib/ndb
cp auth $home/lib.ndb.auth.org
```

Open auth in an editor and uncomment:

```
# hostid=bootes
#       uid=!sys uid=!adm uid=*
```

```
Save the file to /lib/ndb/auth.
```

When the authorization services become active, you'll need a disk partition in which to save your keys. The most important aspect here is the distinction between the partition table of the drive and the Plan 9 partition, which has internal partitions. We will insert a new sub-partition inside the Plan 9 partition.

```
disk/prep /dev/sdC0/plan9
```

You should see several partitions, in this particular case:

```
9fat            0 20482        (20482 sectors, 10.00 MB)
fs          20482 6143321      (6122839 sectors, 2.91 GB)
swap      6143321 6297921      (154600 sectors, 75.48 MB)
```

Make a note of your setup; you will need it! To reduce the swap partition by a single sector, remove the existing swap partition and recreate it one sector shorter followed by a new "nvram". Once this is done, reboot the system and log in as glenda. You will see several error messages related to nvram and authentication:

```
disk/kfscmd allow
echo blahblahblah >/dev/sdC0/nvram
```

The nvram partition is ready to be configured *after* we recompile the kernel so that it will support the necessary security features:

```
cd /sys/src/9/pc
mk 'CONF=pcauth'
cp 9pcauth /386
9fat:
cp 9pcauth /n/9fat
```

Edit /n/9fat/plan9.ini:

```
bootfile=sdXX!9fat!9pcauth
```

Reboot the box, and it should come up under the new kernel. If you run into problems at this point, the basic recovery process is to boot from the install floppy and debug from there. You'll want to mount the Plan9 partition by running "9fat:" at the prompt to mount the partition on /n/9fat. Arguably, the best approach might be to copy any missing stanza members in pcdisk to pcauth before the first compile. In this way, the authorization server has all the capabilities of the I/O server. After booting the new kernel, you'll need to configure basic authorization. After selecting the filesystem from which to boot, you should see the following:

```
bad nvram key
bad authentication id
bad authentication domain
authid:
```

This indicates that your NVRAM needs configuring. Do the following:

1. At the "authid:" prompt, enter "bootes".

2. At the "authdom:" prompt, enter your domain.

3. At the "secstore key:", enter a key that is 8 characters. Don't forget it or share it with untrusted parties.

4. At the "password:" prompt, enter the password you would like bootes to use; don't forget it or share it.

You can now reboot the machine and select the kernel you want to use -- a basic I/O-cpu kernel (i.e., glenda) or one with built-in

authorization (i.e., bootes).

Jim Choate has been involved with computers since the late 1970s, with an interest toward automation and the social impact of technology. His current obsessions are developing Open Forge, LLC in Austin, Texas, and completing a Physics degree. When not trying to make a living or working with local activist groups, Jim promotes Plan 9 as an alternative Open Source OS.

Joe Wessels is a self-taught computer geek who started with PETs and the C-64 computer. Joe is currently employed in building/service/support of non-linear editing systems and other broadcast video-related electronics. He can be contacted at: ***joew@open-forge.com****.*

1 2 3 4 5 6 Next