**CSCI 2930: Practical System Administration**

**Lab 5**

**Spring 2014**

FreeBSD is an operating system based on Unix, as most Linux distributions are. The primary benefit of FreeBSD is the method it uses for handling storage. It uses partitions like all traditional operating systems do, but it also uses “slices,” which are essentially logical partitions (or logical volumes, for you Windows users), to manage its disk space and file system(s).

Now that we have FreeBSD installed on our virtual server, let’s explore some of these storage concepts so we can get a solid grasp on what FreeBSD has to offer and how to manage this.

First, let’s note that the text editor that we used in our previous servers (CentOS/RHEL) won’t be as friendly in FreeBSD. Vi isn’t as flexible or easy to use in this operating system. As such, we’ll be switching over to “easy editor” using the command “ee” to make this easier on ourselves.

The first thing we want to look at is our mount points to see how FreeBSD is configured by default. Boot up your FreeBSD virtual machine in the default multi-user mode and login with root access.

Take a look at your fstab file:

# ee /etc/fstab

You should see an output in a table-style format. Each column is aptly labeled “Device, Mountpoint, FStype,” etc. Under device, you’ll see a couple of devices listed that look like this:

# /dev/da0p2

-/dev/ is what notes the mountpoint for the device (all devices by default are mounted into /dev/ using the device file system).  
-da0 is the device in question, meaning “device 0.” This is your first hard disk.  
-p2 is the partition on the device that is mounted. So we’re working with partition 2.  
-For your first entry, the Mountpoint is just “/” which is your root directory. You’ll notice that the file system is UFS (Unix File System) and it has been mounted as read/write.  
-Your second entry (if using default settings) should read a separate partition (/dev/da0p3) with a Mountpoint of “none” with a “swap” file system, mounted as swap/writeable. This is your swap partition or swap space. FreeBSD creates this by default and uses it as “virtual memory” when your RAM fills up.

Hit “Esc” on your keyboard to exit ee.

Next, let’s take a look at our used space. First, let’s explore the command to do this:

# man df

What does this command do for us? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Now let’s use it in practice:

# df –h

What does the “-h” option provide? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Here, you can see our file systems that are currently in use, how much space is allotted to that slice, how much is used, how much is available, its capacity percentage, and its mountpoint.

Why is this useful to us? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Let’s “pretend” that our storage space is filling up quickly. As system administrators, we need to solve this problem before it brings our server to a halt. We’ve scheduled some maintenance time to add some storage to this server so we may correct the problem.

First, let’s shutdown our server:

# shutdown –h now

What does the “-h” option do for us this time? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

When the system has halted, Power Off your virtual machine using VMWare.

Using your VMWare settings, add a second hard drive to your virtual machine with the following configuration:

Type: SCSI  
 Size: 5GB  
 (I’m not going to hold your hand on this one – it’s a GUI – figure it out!)

Power your virtual machine back on into the default multi-user mode.

Once you’re logged in, let’s take a look at our used file systems again:

# df –h

Notice that our new hard drive does not show up yet.

Let’s make sure that FreeBSD can detect our new hard drive. To do so, we’ll need to investigate the dmesg.boot file:

# cat /var/run/dmesg.boot | less

What does the “less” pipe do for us? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Note that there is a lot of information here, but we’re looking for something very specific. If you remember from the beginning of this lab, we found that our disks are labeled as “da#” where the “#” changes based on the number of the disk. These are our physical disks. If we see two different disk numbers listed in this file, we know that FreeBSD is detecting our physical disk. We should see a “da0” and a “da1.” You’ll also be able to get what type of hard disk it is and its size in MB, even its transfer speed.

What is the transfer speed of the disk we just added? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In order to use the hard disk we added, we need to configure it. Our first step is to add a scheme:

# gpart create –s GPT da1

GPT stands for GUID Partition Table. This is the method that FreeBSD uses to manage its slices.

Now, we create our first slice:

#gpart add –t freebsd-ufs da1

This creates da1p1 and assigns it the FreeBSD Unix File System (but note that it doesn’t format it!).

What does “da1p1” mean? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Now we actually create the file system on the slice (this is similar to formatting):

# newfs –U /dev/da1p1

Next, we must create an empty directory in order to mount the drive for use:

# mkdir /mnt/disk2

Let’s mount our new disk manually, just to be sure our configuration is working:

# mount /dev/da1p1 /mnt/disk2

Note here that since we mounted this manually, it will not mount automatically in the future. This is something we have to do in a configuration file (you’ll see this later in the lab).

Next, let’s make sure that FreeBSD is recognizing it as a usable file system:

# df –h

You should now see /dev/da1p1 at the bottom of your list. Note that it is showing that it is mounted at the mountpoint that we created “/mnt/disk2.”

Now that we have successfully added our hard disk, we need to work on migrating data from the currently-used disk to it.

Reboot your server and boot into Single User Mode at the prompt:

# shutdown –r now

When you boot into Single User Mode, you are prompted for a shell that you’re going to use. Hit “Enter” to select the default /bin/sh.

Think of Single User Mode as the equivalent of “Safe Mode” in Windows, except much more powerful. It doesn’t even mount our root directories by default. All it does is load a shell so we can do our thing.

As such, our first step is to mount our file systems:

# mount –a –t ufs

This command says to mount all (-a) file systems of type (-t) UFS.

Next, we need to mount our new drive (since it’s not done automatically):

# mount /dev/da1p1 /mnt/disk2

Now we change our directory to the new disk:

# cd /mnt/disk2

Next, we copy over the data that we want to offload from the current hard drive. In our case, we’re going to move over the /usr directory first:

# cp –RpP /usr/\* /mnt/disk2

Note that most documentation will tell you to use the “dump” command. This was an excellent way of doing this in older versions of FreeBSD, but now the “dump” command can only copy entire file systems, not portions of them. Since we only want to copy over one directory (and its contents), we’re using the “cp” command. Dump is much better for backups.

Using the man page for the cp command, what do the options –RpP do for us here? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

What option could we use to replace the “-RpP” options? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Now, we edit our fstab file to so that the new drive is used as the mount point for the /usr directory:

# ee /etc/fstab

Add a new line with the following information:

/dev/da1p1 /usr ufs rw 2 2

Use tabs for your spacing, as this is a tab-delimited file.

Save and close your fstab file. Now when you boot into multi-user mode, the new drive will be mounted automatically at the /usr location, containing all of the contents that we copied over.

Let’s reboot our machine and take a look:

# reboot now

;)

Let’s take a quick look to make sure that everything is working fine after our reboot:

# df –h

You should see our new drive mounted at the /usr mountpoint if it’s working properly.

Show a quick list of your /usr directory, and you’re free to leave! Remember to upload your answers to the questions throughout the lab to Canvas!