

Exercises: Post FreeBSD Install: Part II: IP Services Workshop

SANOG 9

January 14, 2007

Your instructor may indicate some updates or changes to these exercises. The initial exercises are to finalize our machine configurations for use later today, and throughout the week, as well as to highlight some areas that are different from Linux - particularly methods by which you can install software.

Exercises

Locking Down and Updating your Box

1. [Keep Your Box off the Net](#)
2. [Edit /etc/rc.conf](#)
3. [pkg_add rsync, ssh, portsnap as needed](#)
4. [Creating a Custom Kernel](#)
5. [Use portsnap to Upgrade your Ports Source Tree](#)

Notes (CRITICAL)

1. The "#" and "\$" characters before commands represents your system prompt and is not part of the command itself. "#" indicates a command issued as root while "\$" indicates a command issued as a normal user.
2. **rehash:** If you install software, update your environment as root and the change is not immediately available try typing `rehash` at the root shell prompt. This is only necessary when running a C shell (e.g., like `/bin/csh`).
3. **italics:** Items that are in *italics* are to be replaced with something of your choice. For instance, *username* means choose your own username, don't literally choose the word "username".
4. **noc:** References to "noc" refer to the classroom server that is used during the workshop to store software, act as a local web server, provide DNS services, etc.

1.) Keep Your Box off the Net [[Top](#)]

OK, so you've probably already booted your box and it may already be live and on the network. If so, there are a few ways to take it off the network. Either you can just pull the network cord out, but more properly let's do (as root):

```
# /etc/rc.d/netif stop
```

You use the `ifconfig` command to control specific interfaces. You could type:

```
# ifconfig fxp0 down
```

(*fxp0* is an example only. Your network interface may be different). This would bring down this specific network interface, but it does not guarantee that the interface won't receive packets, and the default route associated with the interface will still be up for other interfaces.

You may have noted that FreeBSD uses the name of the driver for the network card to describe the network interface while Linux uses the more generic `ETH0`, `ETH1`, etc. names to identify network interfaces.

2.) Edit `/etc/rc.conf` [\[Top\]](#)

At this point the `joe` editor should now be on your machine. You are free to use either `joe` or `vi` for this exercise. The file `/etc/rc.conf` overrides settings in `/etc/defaults/rc.conf`. You should never make changes to `/etc/defaults/rc.conf` as this file may get overridden by a system upgrade.

You can read through `/etc/defaults.rc.conf` to see the hundreds of options you can set. In general, if you wish a service to run at system start up you must enable it in `/etc/rc.conf`. Anything in `/etc/rc.d` can be enabled by using the form:

```
name_of_service_enable="YES"
```

For third party services you should read their initialization scripts in `/usr/local/etc/rc.d` to see what to add to `/etc/rc.d` for the service to start at system boot.

In addition multiple system configuration options are set in this file.

Here is a sample configuration file. At this time take your `/etc/rc.conf` file and try to merge it properly with this example. Your instructor will discuss this file a bit as you start the exercise.

Note, the file has been placed here:

```
ftp://noc/pub/FreeBSD/configs/rc-sanog9.conf
```

Feel free to grab this and copy and paste to your `/etc/rc.conf` file if you want. If you have questions ask one of your instructors. When you are done you should reboot your system.

Sample `/etc/rc.conf` file:

```
#!/bin/sh

# This file now contains just the overrides from /etc/defaults/rc.conf.
# Please make all changes to this file, not to /etc/defaults/rc.conf.

#tmpmfs="YES"                # Place /tmp in memory
#tmpsize="128m"              # How large /tmp will be
                             # We don't have enough RAM to do this
                             # on our machines.
varmfs="NO"                  # Never place /var in memory
background_fsck="NO"         # Don't let fsck run in the background

#####
```

```

### Network configuration sub-section #####
#####

### Basic network and firewall/security options: ###

hostname="xxx.xxx.xx.xx" # NOTE: Given in class.

# For the following option you need to have TCP_DROP_SYNFIN set in your
# kernel. Please refer to LINT and NOTES for details.
#tcp_drop_synfin="YES" # Set to YES to drop TCP packets with SYN+FIN
# NOTE: this violates the TCP specification
#
ifconfig_lo0="inet 127.0.0.1" # default loopback device configuration.

### The values given in class ###
ifconfig_fxp0="inet nnn.nnn.nnn.nnn netmask nnn.nnn.nnn.nnn"
defaultrouter="nnn.nnn.nnn.nnn"

inetd_enable="YES" # Run the network daemon dispatcher (YES/NO).

named_enable="YES" # Run named, the DNS server (or NO).

sshd_enable="YES"

### Network Time Services options: ###
ntpd_enable="YES" # Run ntpd Network Time Protocol (or NO).
ntpd_flags="-g -p /var/run/ntpd.pid -f /var/db/ntpd.drift"
# Flags to ntpd (if enabled).

#####
### System console options #####
#####

keymap="us.emacs" # keymap in /usr/share/syscons/keymaps/* (or NO).
keyrate="fast" # keyboard rate to: slow, normal, fast (or NO).
blanktime="NO" # blank time (in seconds) or "NO" to turn it off.

#####
### Miscellaneous administrative options #####
#####

clear_tmp_enable="YES" # Clear /tmp at startup.

kern_securelevel_enable="NO" # kernel security level (see init(8)),
kern_securelevel="-1" # range: -1..3 ; '-1' is the most insecure

check_quotas="NO"

# end

```

Don't forget to reboot:

```
# shutdown -r now
```

3.) pkg_add rsync, ssh, portsnap as needed [\[Top\]](#)

SSH is already installed on your boxes so you do not need to do this. You will, however, need both rsync and portsnap on a properly run system. If this was a brand new box that had just been brought up and your were trying to update your ports collection immediately as well as bring data across from another server, then you could get these by using pkg_add. Let's do that by doing this (you must be root):

```
# pkg_add ftp://noc/pub/FreeBSD/6.1-RELEASE/i386/packages/All/rsync*
```

Note the trick of using "*" to avoid having to specify the specific filename.

Portsnap is, also, already installed. You could, however, have simply typed:

```
# pkg_add -r portsnap
```

to have installed. Read the portsnap man page for some more information [`man portsnap`]

We'll use portsnap a bit later. We might use rsync later in the week.

4.) Creating a Custom Kernel [\[Top\]](#)

Depending on we are doing for time we may or may not do this exercise. Your instructor will let you know.

If you have installed full source (we did), then you will have the directory structure `/usr/src/sys`, which contains the source code for the kernel. In addition, for an i386-based box you can find the configuration file for the kernel you are currently running here:

```
/usr/src/sys/i386/conf
```

The configuration is in the file `GENERIC`. As you might guess this file has a *lot* of options turned on that are likely not relevant to your machine. In addition, it's possible at some point that some option might create a security issue as well. Going through this file, commenting out what you don't need, and then rebuilding your kernel is a bit of an art. To do this properly you should, at least, read:

```
/usr/share/doc/handbook/kernelconfig.html  
/usr/src/sys/i386/conf/NOTES
```

For this exercise we will remove support for one class of devices that your machines do not have - SCSI controllers. This will reduce the complexity and size of your kernel. To do this we'll make a copy of the `GENERIC` kernel configuration file, edit the copy and recompile the kernel with the new options. In order to do this you will need to be root:

```
# cd /usr/src/sys/i386/conf  
# cp GENERIC /root/SANOG9  
# ln -s /root/SANOG9 SANOG9
```

You make a logical link to the `SANOG9` kernel config file using the `ln` command to avoid the file being lost if you upgrade your system and the upgrade process remove files in `/usr/srs/sys`.

Now that you have the `SANOG9` kernel configuration file we are going to edit this and make some changes.

```
# vi SANOG9  
:85 [go to line 85]
```

Now you should comment out all the SCSI controller items. This means all entries between the "SCSI Controllers" heading up to the "SCSI Peripherals" heading. You can do this using a regular expression in VI

with a single command. This is quite powerful. Here is what you would type:

```
:85,104s/^/#/
```

This should insert a "#" (comment) character at the start of each line between line 85 and 104.

Now that you have updated your SANOG9 kernel configuration file you should save and exit:

```
:wq
```

And, now we will make and install a new kernel. Do the following:

```
# cd /usr/src/sys
# make KERNCONF=SANOG9
make install KERNCONF=SANOG9
```

This should complete without errors. If it does, you are ready to reboot your machine and try out your new kernel. If you run in to problems when rebooting you can do the following:

- At the initial boot screen before the 10 second countdown ends press any key but ENTER.
- type "unload" at the prompt.
- Now type "boot /boot/kernel.old/kernel", and this will start your machine with your original, working kernel.

At this point you should reboot your system. You should note that one of the only times you absolutely must reboot a Unix or Linux box is when you install a new kernel. So, now do:

```
# shutdown -r now
```

5.) Use portsnap to Upgrade your Ports Source Tree [\[Top\]](#)

The portsnap tool is a simple and easy method in FreeBSD to keep your entire /usr/ports tree up-to-date. There are other methods to do this, including the use of cvs, and the portupdate tool. The idea behind portsnap is that you down an initial snapshot file that contains all the current updates to the ports collection. This file is somewhat large, around 50MB in size. Thus, on a slow connection, this may take some time. But, the advantage of portsnap is that once you've downloaded this file, remaining updates are small as you only get what's new since your last time updating the file.

To use portsnap is extremely simple. Here are the step, but *please don't* do these right now:

```
# portsnap fetch
# portsnap extract
```

On subsequent updates you do:

```
# portsnap fetch
# portsnap update
```

If you plan on using portsnap consistently and want to use it in a cron process, then please read the portsnap

entry in the handbook:

`/usr/share/docs/handbook/portsnap.html`

For our purposes we don't want everyone download a 50MB file across our outside connection, so we have a copy of the initial portsnap files on our local server and we'll copy these files over to your machines. This is an artificial scenario, but it allows us to update our entire ports collection, and to keep it updated during the week:

```
# mkdir -b /var/db/portsnap
# cd /var/db/portsnap
# scp -r ipserve@noc:/var/db/portsnap/* . <pw = su55ess!>
```

You will see quite a few files scroll by on the screen at this point. This will take some time so we may start our next lecture while it completes, issue the last command below, and continue on.

Once done you need to do one more step before you can extract all the updates that are now available in to your ports collection

Normally you would simply have typed one command to get to this point. We have simply simulated the steps that the portsnap shell script does. This script is `/usr/sbin/portsnap` and you can read it if you wish.

Now let's extract the updates in to your ports collection:

```
# portsnap extract
```

This will take a little bit. Again, once this completes you can always update your ports collection quickly and easily by issuing the command:

```
# portsnap fetch update
```

If you want further information remember to read `/usr/share/docs/handbook/portsnap.html`.

[\[Return to Top\]](#)

Hervey Allen

Last modified: Sat Jan 13 03:14:20 IST 2007