Customizing the FreeBSD Kernel

Written by Greg King
Saturday, 04 July 2009 10:24

The kernel is the core of the FreeBSD operating system. It is responsible for managing memory, enforcing security controls, networking, disk access, and much more. While more and more of FreeBSD becomes dynamically configurable it is still occasionally necessary to reconfigure and recompile your kernel.

Because newer FreeBSD versions use dynamic modules, there is not a compelling need to recompile your kernel, unless you just don't like seeing that you are using the GENERIC kernel when you log in or do a uname. If you wish to compile your own kernel, please be careful and avoid commenting drivers out unless you really know what you are doing. If you wish to compile your own kernel, the following are the basic steps.

The following assumes that you selected the 'kernel developer' option when installing your FreeBSD system. If not, the easiest way to do this is by running sysinstall as root, choosing Configure, then Distributions, then src, then sys.

```
# sysinstall
```

Also, you might want to keep your kernel config file elsewhere, and then create a symbolic link to the file in the i386 directory. This is for protection from accidental erasure.

Traditionally, the kernel name is in all CAPITAL LETTERS and, if you are maintaining multiple FreeBSD machines with different hardware, it is a good idea to name it after your machine's hostname. We will call it MYKERNEL for the purpose of this example.

```
 # cd /usr/src/sys/i386/conf
# mkdir /root/kernels
# cp GENERIC /root/kernels/MYKERNEL
# ln -s /root/kernels/MYKERNEL
```

Now, edit MYKERNEL with your favorite text editor.   ***NOTE:*** *Open a new ssh window and type* *dmesg*
 at the root (

#
) prompt to get a list of your particular system's installed devices and use it as a guide during the edit of your config file. It makes life much easier!

```
# nano MYKERNEL
```

***NOTE:** Be sure and leave the line 'machine   i386' alone and intact when defining your architecture.*

You must now compile the source code for the kernel.

```
# /usr/sbin/config MYKERNEL
# cd ../compile/MYKERNEL
# make cleandepend; make depend
# make
# make install
```

The new kernel will be copied to the root directory as /kernel and the old kernel will be moved to /kernel.old. Now, shutdown the system and reboot to use your new kernel.

```
# shutdown -r now
```

When the system comes back up, log in and type

```
> uname -a
```

the output should look something like this...

```
FreeBSD <your system name> 7.1-RELEASE FreeBSD 7.1-RELEASE #0: <date / time of install>   <email address for the system>:/usr/src/sys/i386/compile/          MYKERNEL 
i386
```

which shows us the system booted off the new kernel!

IF SOMETHING GOES WRONG

If for some reason, your system will not boot off of the new kernel, do this...
On boot, when the Boot Menu comes up and during the 10 second countdown period, type number 6 to select the 'Escape to loader prompt' option. At the prompt, type

```
> boot kernel.old
```

to boot from the previous kernel.

If you are having trouble building a kernel, make sure to keep a GENERIC, or some other kernel that is known to work on hand as a different name that will not get erased on the next build. You cannot rely on kernel.old because when installing a new kernel, kernel.old is overwritten with the last installed kernel which may be non-functional. So, copy kernel.old to kernel.safe with:

```
# cd /boot# cp -Rp kernel.old kernel.safe
```