

## **Rebuilding the Linux Kernel**

This paper is intended to be a step-by-step tutorial for rebuilding the linux kernel.

*WARNING: This is a work in progress. I basically sat down and wrote this all out from memory. In the next couple of days I will sit down and do an actual rebuild following my tutorial to verify its completeness and to expand on certain topics. Until then I will leave this warning intact.*

1. Download the new kernel source. <http://www.kernel.org>
2. Save the new source file "linux-2.4.xxx.gz.tar" to the /tmp directory.
3. Extract the source "tar zxvf linux-2.4.xxx.gz.tar"
4. You will now have a sub-directory linux i.e. /tmp/linux which contains the new source.
5. The new source must be moved into /usr/src into its own sub-directory.
6. Make a new directory for the new source "mkdir /usr/src/linux-2.4.xxx/"
7. Move extracted source in /tmp/linux to /usr/src/linux-2.4.xxx
8. Change to /tmp/linux "cd /tmp/linux"
9. Move source "mv \* /usr/src/linux-2.4.xxx/"
10. Change directory to /usr/src. "cd /usr/src"
11. Remove old symbolic link "rm linux"
12. Create a new symbolic link pointing to the new source you want to build.
13. "ln -s /usr/src/linux-2.4.xxx/ linux"
14. Verify the new symbolic link works. "cd linux"
15. You should now be in the directory with the new source.
16. To prepare the source for compilation run 'make mrproper' (2.4+)
17. There are a couple different interface options for selecting the new kernel configuration.
18. If you have X windows installed I recommend using 'make xconfig'
19. If you only have command line access but have a color monitor and capable of ASCII graphics use 'make menuconfig'
20. If you only have barebone command-line access you have to use the command line configuration shell. Just type 'make config'
21. Steps 17 through 19 all accomplish the same task. They are just three different ways of doing it.
22. At this point you will be making selections of what to include in your new kernel. There are 3 options for most selections. Include, Module, or Off. Include means to build as part of the kernel itself. It always loads. Modules on the other hand can be dynamically loaded and unloaded. Off means the code is not compiled.
23. After you have decided on the new kernel configuration, select Save Configuration. Close the window or exit. It might prompt to save. Do it.
24. You will now be back at the console.
25. Now you have to manually compile the kernel in steps.
26. This is as easy as just typing what appears here. First type "make dep"
27. Next "make"
28. Then "make modules"

29. Then “make modules\_install”
30. Then “make install”
31. This process can be simplified by using ‘;’ semi-colons between commands. That way all of the commands can be executed in one command. (make dep ; make modules ; etc. ...)
32. NOTE: By default ‘make install’ will replace the link on your boot menu that points to your current kernel with a link to the newly compiled kernel. If you want to preserve your old kernel you must manually edit “/etc/lilo.conf”
33. NOTE: You must add a new entry for your old kernel in ‘lilo.conf’. Just template the new entry with the other entries changing the kernel file name and the label name. Be careful and get the kernel file names exactly as they are written.
34. NOTE: A good rule of thumb is to edit the ‘Failsafe’ entry of ‘lilo.conf’ and make it point to the exact filename of the kernel you are currently running. This means replace the reference to ‘vmlinuz’ or ‘old’ to the exact filename of the currently running kernel. This ensures that if you have a problem with the new kernel, selecting ‘Failsafe’ from the boot menu will return you to a working linux kernel.
35. NOTE: For info on this type ‘man lilo.conf’ or ‘man lilo’
36. NOTE: Don’t forget to re-run ‘lilo’ every time after editing ‘lilo.conf’ to update the Master Boot Record with your changes.
37. Finally you must run the bootloader updater. Just type “lilo”
38. Lilo updates your master boot record. In this instance it sets the option ‘Linux’ on your boot menu to the kernel you just compiled. (If you manually edited lilo.conf your new setting will take effect)
39. Lilo should come back and say linux (or whatever you changed) was added. The new kernel is now ready to boot.
40. Reboot the machine and watch the boot process. If the kernel config was set up correctly you should see very few or no errors and get to the login prompt.
41. If the machine has many errors and doesn’t get to the login prompt, you’ll need to type “linux single” at the lilo bootloader prompt. This is basically ‘safemode.’ If you can get into the machine in single mode, you can change directory to /usr/src/linux/ and remake the kernel. You probably won’t be able to use ‘make xconfig’ You’ll have to use ‘make menuconfig’ or just ‘make config’
42. If you followed the note in Step 31 and made a backup entry for your old kernel in your bootloader menu you could just reboot and select your old kernel.

That’s it! If everything went smoothly you should have a new stable kernel.

#### Some issues that might come up

##### **‘depmod’ reporting module errors.**

This is normal for modules [outside of the kernel] which were compiled for a different kernel.

SOLUTION: Either recompile that module individually or obtain one built for your new kernel.

There are tons more little issues, I’ll add them as they come up.