Xmetalfanx Quick guide to make your own Linux Kernel

OPENING

Assumptions

This is assuming you have all the needed packages install first, too .. Please see the tarball's documentation for that information.

What this guide is not for

ALSO this is not for Linux Mint, Ubuntu (or any variant of …), or any (I think) “Debian” based version of Linux. … Well to be honest, this may work … Though I have not tried it, myself … The package manager in Ubuntu, for instance, makes it a piece of cake to upgrade your Kernel, though it may not be the most up-to-date kernel listed on Kernel.org.

Even after adjusting parts of this guide that “changes” things to make the offical tarball Kernels work with Ubuntu.... it may be too much work, considering how easy the package manager is … though if you need a more up-to-date kernel than the Ubuntu Package Manger offers, then go ahead and try it.

“Why would I want to upgrade my kernel?”

[Note: this is REALLY summing this up]

Upgrading your kernel can allow for changes (bugfixes in prior kernels, new features such as hardware being detected, and other changes)

> *Real Example*
>
> In Slackware 12.2, I had things working correctly, though audio kept coming threw my speakers, even when I had headphones plugged in. This may not seem bad, though if in a public place, this could be an issue (if not anything else, then Common Curtacy)
>
> A simple kernel upgrade 100% fixed this issue.

WARNING

Like many things I have learned through messing around, sometimes you can really mess things up. If your even curious about this (and you didn't download this by accident), then your smart enough to back up your data. (Even if you never want to upgrade your kernel, its a great idea to back your needed data up)

“Hey... why do I have to upgrade my kernel to fix an issue?... Wouldn't simply upgrading

Compiling a 2.x Kernel | Compiling a 3.c Kernel

Steps to to making a new 2.x tree kernel in Linux.

This is not geared to Ubuntu-type OS's where you download and "auto-install" new kernels, but this is more for distro's such as Slackware

1. Go to http://kernels.org and download the appropriate kernel in a tarball format (I can not say which is the correct one for your system.. I just downloaded the newest FULL kernel that was available at the time
2. Extract the tarball ..
    - I want to be clear that this needs to be done in a Linux partition
        - THAT MAY SEEM obvious, but I had more room on my NTFS partition and I extracted the tarball on that partition .. later on in the process, I ran into trouble ... after some fussing with it, I decided it was a good idea to move the extracted tarball folder to my Linux partition and the problems were solved
3. Go to /Boot and copy your .config file to the folder where you extracted the tarball
4. **Configure the Kernel**
    - # `make menuconfig`
    - #`make -j`(number of cores+1) `all`" for multi core processor --- WITHOUT the quotes
    - this can vary, but here are two examples
        - #`make menuconfig`
        - `#make xconfig`
    - However you have to do it (command line and xconfig's GUI are different), load the .config file
        - (you may need to rename (***THE COPIED file in your new tarball main directory***) `.config` to `config`
    - Configure and save the settings you need
        - I can not say what all the settings you need are since systems and user needs are so different
        - Just remember .. clicking enable all (features) may seem like a good idea, but not only does it increase the size of the kernel, but it takes longer for the Linux Linux to start.
            - This may also cause un-desired results that are unexpected

**** IMPORTANT STEP HERE ** This is where you compile the kernel and adjust it to what you need.**

5. **Compiling the Kernel**
    - [In the extracted tarball folder]
    - Do the following commands, IN ORDER .. the # i type represents the command prompt and IS NOT part of the command itself
    - #`make dep`
        - * May not be needed; however, some tutorials have this listed, so I do it... if its not needed, this doesn't do any harm.
    - #`make clean`
        - * May not be needed; however, some tutorials have this listed, so I do it... if its not needed, this doesn't do any harm.

- #`make bzImage`
    - This actually makes the kernel, The location of the kernel is located in (/arch/i386/boot/ AND the file name is bzImage) ... (That is when this step is completed)
    - Give this step time as even on my faster laptop it took awhile
- #`make modules`
    - This complies the modules in the kernel, that may be needed by various systems.
    - For a "user to user" basis, you have to choose which ones for your system. It would take way too much space here to go into and even I admit I am not expert on all the different options. Go research options your curious about .. Many Linux users will be happy to help you if you have any questions
- #`make modules_install`
    - This moves the made modules (step 4) into the current directory
- #`make mrproper`
    - This is IF you want to re-configure or remove the .config file

6. NOW YOU MOVE the following three files to your /boot directory

.config

- You want to move it to a file called (in otherwards your re-naming the file) "Config.kernelNumber"
- (where kernelNumber reflects the kernel number your installing/comp

System.map bzImage

- The way I do this is in the main kernel (extracted directory) I do the following two commands
    - # `cp System.map /boot`
    - # `cp .config /boot/configure_kernelNumber` (from above comments)
- In **`/arch/i386/boot/`**, as mentioned in step 3, I do
- # `cp bzImage /boot/vmlinuz-kernelNumber`
    - Example - vmlinuz-3.04

7. Configure your boot loader to use the new kernel

- IF DONE properly, the great thing about upgrading your kernel via this method (and actually Ubuntu, package manager does the same thing) is if your newly compiled kernel does not boot no matter what you do, you DIDN'T replace the .config file or the actual “Original kernel” itself. (the kernel you were booted into when you tried this entire guide)
    - Just Select the old kernel from the grub boot menu, and your good to go in the old kernel again to try again (to see which setting you selected or un-selected that goofed things up) OR .. .just to take a break from kernel upgrading and do whatever you want in Linux, to come back to upgrading the kernel later.
        - [Todo: put URLs to sites like SuperGrubBootCD here, … just for helping users avioid issues]
- (Details here will be added later)

Steps to making a new 3.x tree kernel in Linux

I have to say that what's listed below is from the following URL
http://www.standardcode.eu/blog/linux/compilation-and-installation-of-kernel-3.0.0-in-slackware.html ..

Much of the instructions seems the same; though, any changes or additions to the 2.x tree instructions (despite being on many sites) I give credit to that site for where I got my information.. I do not claim I would have any idea how to compile a 3.x without that site and give full credit to the information additions I make (below) to that site's author/Article's author

1. Go to http://kernels.org and download the appropriate kernel in a tarball format *(I can not say which is the correct one for your system.. I just downloaded the newest FULL kernel that was available at the time)*
2. Extract the tarball of the kernel
3. Go to /Boot and copy your .config file to the folder where you extracted the tarball
4. Configure the Kernel
    1. # `make menuconfig`
    2. #`make -j`(number of cores+1) `all`" for multi core processor --- WITHOUT the quotes
    3. this can vary, but here are two examples
        - #`make menuconfig`
        - #`make xconfig`
        - However you have to do it (command line and xconfig's GUI are different), load the .config file
            - (you may need to rename .config to config)
        - Configure and save the settings you need
            - I can not say what all the settings you need are since systems and user needs are so different
5. # `make all`
6. # `make modules_install`
7. Go to the /etc/rc.d Directory
    1. # `cd /etc/rc.d`
    2. Run the following two commands in that directory
        - copies file (or "copies and renames that newly copied file")
        - # `cp rc.modules rc.modules-3.0`
        - removes the original file
        - # `rm rc.modules`
    3. Create Sym link for the file mentioned above
        - # `ln -s /etc/rc.d/rc.modules-3.0 /etc/rc.d/rc.modules`

8. NOW YOU MOVE the following three files to your /boot directory

.config
- You want to move it to a file called (in otherwards your re-naming the copied file) "Config.kernelNumber"
- (where kernelNumber reflects the kernel number your installing/compiling)

System.map   bzImage

- The way I do this is in the main kernel (extracted directory) I do the following two commands
    - # `cp System.map /boot`

    - # `cp .config /boot/configure_kernelNumber` (from above comments)
- In /arch/i386/boot/, as mentioned in step 3, I do
- # `cp bzImage /boot/vmlinuz-kernelNumber`
    - Example - vmlinuz-3.04

Closing

My Website:

For more tips. programs suggestions, and security information, please visit my sites at http://xmetalfanx.awardspace.us or http://xmetalfanx.x10.mx/ .. When I have fully uploaded my files, both sites will be exactly the same (think of them as mirrors). I have had bad luck with web-hosts, and did not originally intended to “have a mirror setup” for my site, though once the idea came to me, it seems to make sense.

When fully uploaded, both sites will be considered my “homepage” and I do not have any preference as one host as a “homepage” and the other as a “Mirror of my homepage”.

Different web-hosts I have had in the past, have canceled service without notifying me and when signing up for another host, I have had to wait, so my site was offline (off the internet) for in some cases a week or more at a time. I figure even if one of these hosts, stop service, then I still have the other host to refer people to.

This guide was typed in:

- **Libre Office** (Free Office Client for a variety of Operating Systems)
    - (http://www.libreoffice.org/)

Thanks

Thanks to (I promise I will get exact URLs) all the Linux users who have helped me over the years (you know who you are), and for the kind people in Linux forums (say LinuxQuestion.org for example) and the people taking the time to write out information (example – Tutorials), and people who answer on those prior mentioned forums, in their own free time to help others