**Linux Kernel :**

Your Linux kernel compiled and always installed in /boot directory:

To list of all installed kernel in your system, enter:  
$ ls -l /boot/  
Outputs:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | -rw-r--r-- 1 root root   106172 Mar 23 03:07 config-2.6.32-5-amd64  -rw-r--r-- 1 root root   130657 Feb 11 15:14 config-3.2.0-0.bpo.1-amd64  drwxr-xr-x 3 root root     4096 Mar 28 19:21 grub  -rw-r--r-- 1 root root 10332764 Mar 28 19:21 initrd.img-2.6.32-5-amd64  -rw-r--r-- 1 root root 11650089 Feb 26 03:50 initrd.img-3.2.0-0.bpo.1-amd64  drwxr-xr-x 2 root root    12288 Jan 11 03:19 lost+found  -rw-r--r-- 1 root root   165084 Oct 21  2010 memtest86+.bin  -rw-r--r-- 1 root root   167264 Oct 21  2010 memtest86+\_multiboot.bin  -rw-r--r-- 1 root root  1665393 Mar 23 03:07 System.map-2.6.32-5-amd64  -rw-r--r-- 1 root root  1992422 Feb 11 15:14 System.map-3.2.0-0.bpo.1-amd64  -rw-r--r-- 1 root root  2424256 Mar 23 02:58 vmlinuz-2.6.32-5-amd64  -rw-r--r-- 1 root root  2813456 Feb 11 15:11 vmlinuz-3.2.0-0.bpo.1-amd64 |

Where,

* config-3.2.0-0.bpo.1-amd64 –> Kernel configuration file generated by make menuconfig/make xconfig/make gconfig.
* System.map-3.2.0-0.bpo.1-amd64 –> This file has a map of positions of symbols in the kernel. Device driver such as USB pen uses hot plug, which depend upon symbols generated by depmod utility.
* vmlinuz-3.2.0-0.bpo.1-amd64 — > Your kernel file.
* initrd.img-3.2.0-0.bpo.1-amd64 –> Contains device drivers which are required to boot and load rest of operating system from disk. Usually SCSI,IDE, software RAID drivers are stored in this file.
* grub –> It is a directory, which stores grub Boot loader configuration file.

You may also find the following files:

* config –> Soft link to current kernel configuration file
* vmlinuz -> Soft link to current running kernel file
* System.map –> Soft link to current running kernel system map file

Please note that 3.2.0-0.bpo.1-amd64 is kernel version.

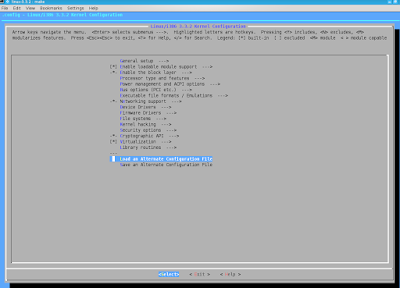
Finding the Running Kernel Information

# uname –r or # cat /proc/version

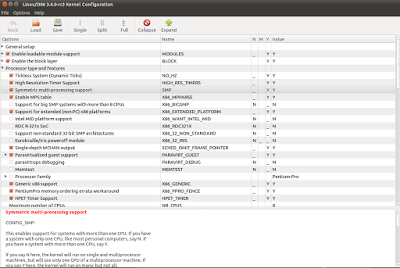
**Check your Linux kernel version**  
        In Linux system use **uname -r** or **cat /proc/version** to find the current kernel version

|  |
| --- |
| [http://4.bp.blogspot.com/-VwizV9ywLPA/T5FL5ibXkmI/AAAAAAAACK0/JJtMFWHudUg/s400/uname.png](http://4.bp.blogspot.com/-VwizV9ywLPA/T5FL5ibXkmI/AAAAAAAACK0/JJtMFWHudUg/s1600/uname.png) |
| Linux Kernel Version |

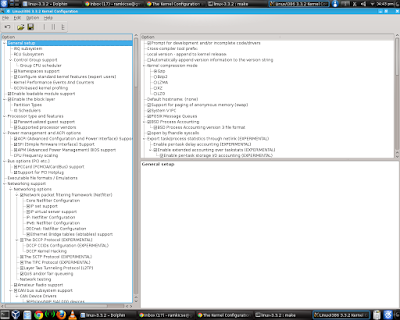
here **2.6.35.22**is current system kernel version  
  
  
**Step 1 : Download the latest kernel**  
 Goto <http://kernel.org/> website and get the latest version of the kernel source. i will get **3.3.3**  
  
**wget http://www.kernel.org/pub/linux/kernel/v3.0/linux-3.3.3.tar.bz2**  
  
and extract the source code.  
**tar xf linux-3.3.3.tar.bz2**  
**cd linux-3.3.3**  
in linux-3.3.3 folder contain lots of files and folders. some folders are use following purpose  
  
**arch** - this folder shows the what are the architecture port the Linux  
**crypto**- contain cryptography implementations like AES, DES  
**drivers**-  all device drivers modules (IDE, SCSI, Ethernet device, Wireless)  
**fs**- all file-system implementation code (ext4, FAT, NTFS)  
**net** - all network protocol implementations (IPv4, IPv6, tcp, 802.11)  
  
 **Step 2 : make the configuration file**  
  While compile the kernel source code, we need configuration file. that configuration file contain lots of variable to help to understand what are the modules we need to compile.  
  
For x86 specific kernel variable check [this reference](http://kernel.org/doc/menuconfig/x86.html)  
  
using make command we can build the configuration, but this command is interactive, its ask more than 1000 question about enable or disable the particular module, like which file-system r u want and IP-Tables related modules like SIP components (state-full inspection firewall).  
  
so the best way is copy the current Linux config file. its stored in /boot/config-<version>.  
**cp /boot/config-2.6.35.22-generic      .config**  
  
now we got old configuration. now we change/add new configuration settings.  
for this purpose therse is lots of options are avialable  
 **make help**    ==>Provides the help  
  
**There is GUI option is also available**  
 **make   menuconfig          ==> this provides Text based GUI Configuration**

[](http://4.bp.blogspot.com/-YrXAfpUms4g/T5FL4LnKeRI/AAAAAAAACKw/5sUvVfHIJkc/s1600/snapshot1.png)

here load the old config file and add/change the settings then save the config file.  
  
**make   gconfig      ==> this provides GTK 2 based GUI Configuration for GNOME**

[](http://2.bp.blogspot.com/-6e-hTFPXIhM/T5FL2I-da1I/AAAAAAAACKg/QVBD97NEPfU/s1600/gconfig.png)

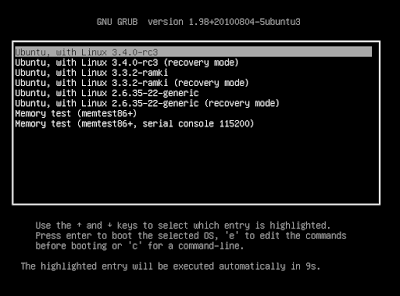
**f**or this we need to install**libgtk2.0-dev, libglad2-dev**through apt-get install  
  
  
**make   xconfig      ==> this provides QT based GUI Configuration for KDE**

[](http://3.bp.blogspot.com/-m6Onxy4ErJU/T5FL6pgEXFI/AAAAAAAACLA/sASX8eU-qFk/s1600/xconfig.png)

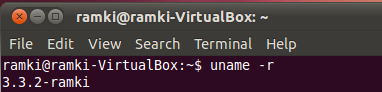
for this we need qt-dev tools  
  
  
Using any one of the above method modify the configuration.  
  
**Step 3 : Compile the Kernel**  
    
        There are two ways to compile the kernel

* Generic way (lots of steps)
* Debian specific way (simple, Use make-kpkg tool)

**Step 3.1 : Generic Way**  
  
    **Step 3.1.1 Compile the Kernel and its modules**  
  
          **make**  
       above code is take 1 to 2 hours depending on system performance. In this step it compile the kernel and store the kernel in binary form to **arch/x86/boot/bzImage**file  
then based on kernel headers it will compile the kernel  modules (device driver, file-system, network,...)  
and generate .ko files. ko means kernel object. These modules also called Loadable Kernel Modules (LKM).  
  
  **Step 3.1.2 Install Kernel modules**  
  
     **make  modules\_install**  
     This step copy the all kernel modules (\*.ko) to **/lib/modules/<version>/kernel/**   folder.  
  
  
      **Step 3.1.3 Install Kernel**  
  
       **make  install**  
      This step copy the the kenel from **arch/x86/boot/bzImage**to**/boot**folder and copy the**.config** file to **/boot/config-<latest-version>** and generate the **System.map**file.  
  
  
     **Step 3.1.4 Create Initramfs file**  
up to now kernel and its modules are compiled and installed. when next boot up time we need to choose latest kernel. so we need to prepare the boot-loader and its support files. When system turns on, after bios and boot loader load the kernel to main memory and mount initial dummy file system as a root file system of system. this initial file system have necessary drivers for disk hardware (SCSI or IDE) and mount the correct file system as a root file system.  
  
so we need to create initramfs file using update-initramfs or mkinitfs tool  
  
    **update-initramfs -c -k 3.3.3**  
here 3.3.3 is new kernel version.  
  
  
   **Step 3.1.5  Update GRUB bootloader**  
  the last step is update the boot loader here i m using GRUB boot-loader.  
  
    **update-grub**  
this command automatically probe the kernels in **/boot** folder and add the entries in its configuration file, **grub.cfg**  
  
  
now restart the system , we will see the new kernel is added in boot loader entries. then choose new kernel in boot loader.

[](http://2.bp.blogspot.com/-RPM3M0gpFeM/T5FL3JLNOCI/AAAAAAAACKk/PnhMQMfWdxk/s1600/grub.png)

now open the terminal and issue**uname -r** command, its shows the  current kernel version.

[](http://3.bp.blogspot.com/-cujdCiZE3no/T5FL05Oe0yI/AAAAAAAACKY/vUGLB2ACtAo/s1600/after.png)