**Chapter 8 - Kernel Upgradation**

**What is kernel**

* The **kernel** is the central module of an operating system (OS). It is the part of the operating system that loads first, and it remains in main memory. Because it stays in memory, it is important for the kernel to be as small as possible while still providing all the essential services required by other parts of the operating system and applications. The the kernel code is usually loaded into a protected area of memory to prevent it from being overwritten by programs or other parts of the operating system.
* Typically, the kernel is responsible for memory management, process and task management, and disk management. The kernel connects the system hardware to the application software. Every operating system has a kernel. For example the Linux kernel is used numerous operating systems including Linux, FreeBSD, Android and others.
* Kernels may be classified mainly in two categories

1. Monolithic
2. Micro Kernel

**1. Monolithic Kernels**

Earlier in this type of kernel architecture, all the basic system services like process and memory management, interrupt handling etc were packaged into a single module in kernel space. This type of architecture led to some serious drawbacks like

1) Size of kernel, which was huge.

2)Poor maintainability, which means bug fixing or addition of new features resulted in recompilation of the whole kernel which could consume hours In a modern day approach to monolithic architecture, the kernel consists of different modules which can be dynamically loaded and un-loaded. This modular approach allows easy extension of OS's capabilities. With this approach, maintainability of kernel became very easy as only the concerned module needs to be loaded and unloaded every time there is a change or bug fix in a particular module. So, there is no need to bring down and recompile the whole kernel for a smallest bit of change. Also, stripping of kernel for various platforms (say for embedded devices etc) became very easy as we can easily unload the module that we do not want.

**2. Microkernels**

This architecture majorly caters to the problem of ever growing size of kernel code which we could not control in the monolithic approach. This architecture allows some basic services like device driver management, protocol stack, file system etc to run in user space. This reduces the kernel code size and also increases the security and stability of OS as we have the bare minimum code running in kernel. So, if suppose a basic service like network service crashes due to buffer overflow, then only the networking service's memory would be corrupted, leaving the rest of the system still functional.In this architecture, all the basic OS services which are made part of user space are made to run as servers which are used by other programs in the system through inter process communication (IPC). eg: we have servers for device drivers, network protocol stacks, file systems, graphics, etc. Microkernel servers are essentially daemon programs like any others, except that the kernel grants some of them privileges to interact with parts of physical memory that are otherwise off limits to most programs. This allows some servers, particularly device drivers, to interact directly with hardware. These servers are started at the system start-up.

**Check the Kernel Version in Linux / Ubuntu**

The following command works with all Linux distributions, such as Red Hat, CentOS, Debian, and Ubuntu. It also works on other UNIX-like operating systems such as HPUX, FreeBSD, OpenBSD, Solaris, etc. Use the following command to check which kernel version your server is currently running:

# uname –r ***or***

# cat /proc/version

You should receive a result similar to the following:

2.6.32-431.11.2.el6.x86\_64

The output can be interpreted with the following key:

2 – Kernel Version  
6 – Major Revision  
32 – Minor Revision  
431.11.2.el6 – Fix/Revision Detail

Per the manual page, uname can also give the following information:

-a, --all                  print all information  
-s, --kernel-name        print the kernel name  
-n, --nodename            print the network node hostname  
-r, --kernel-release     print the kernel release  
-v, --kernel-version     print the kernel version  
-m, --machine             print the machine hardware name  
-p, --processor          print the processor type or "unknown"  
-i, --hardware-platform   print the hardware platform or "unknown"  
-o, --operating-system    print the operating system  
--help                   display this help and exit  
--version                 output version information and exit

**Upgrading Kernel in Ubuntu :-**

**Step 1: Check Installed Kernel Version**

To find the current version of installed kernel on our system we can do:

# uname –sr

**Step 2: Upgrading Kernel in Ubuntu**

To upgrade the kernel in Ubuntu, go to http://kernel.ubuntu.com/~kernel-ppa/mainline/ and choose the desired version (Kernel 4.14 is the latest at the time of writing) from the list by clicking on it.

Next, download the .deb files for your system architecture.

**Download Linux Kernel for 32-Bit System**

# wget <http://kernel.ubuntu.com/~kernel-ppa/mainline/v4.14/linux-headers-4.14.0-041400_4.14.0-041400.201711122031_all.deb>

# wget <http://kernel.ubuntu.com/~kernel-ppa/mainline/v4.14/linux-headers-4.14.0-041400-generic_4.14.0-041400.201711122031_i386.deb>

# wget <http://kernel.ubuntu.com/~kernel-ppa/mainline/v4.14/linux-image-4.14.0-041400-generic_4.14.0-041400.201711122031_i386.deb>

**Download Linux Kernel for 64-Bit System**

# wget <http://kernel.ubuntu.com/~kernel-ppa/mainline/v4.14/linux-headers-4.14.0-041400_4.14.0-041400.201711122031_all.deb>

# wget <http://kernel.ubuntu.com/~kernel-ppa/mainline/v4.14/linux-headers-4.14.0-041400-generic_4.14.0-041400.201711122031_amd64.deb>

# wget <http://kernel.ubuntu.com/~kernel-ppa/mainline/v4.14/linux-image-4.14.0-041400-generic_4.14.0-041400.201711122031_amd64.deb>

Once you’ve downloaded all the above kernel files, now install them as follows:

# sudo dpkg -i \*.deb

Once the installation is complete, reboot your machine and verify that the new kernel version is being used:

# uname -sr

And that’s it. You are now using a much more recent kernel version than the one installed by default with Ubuntu.