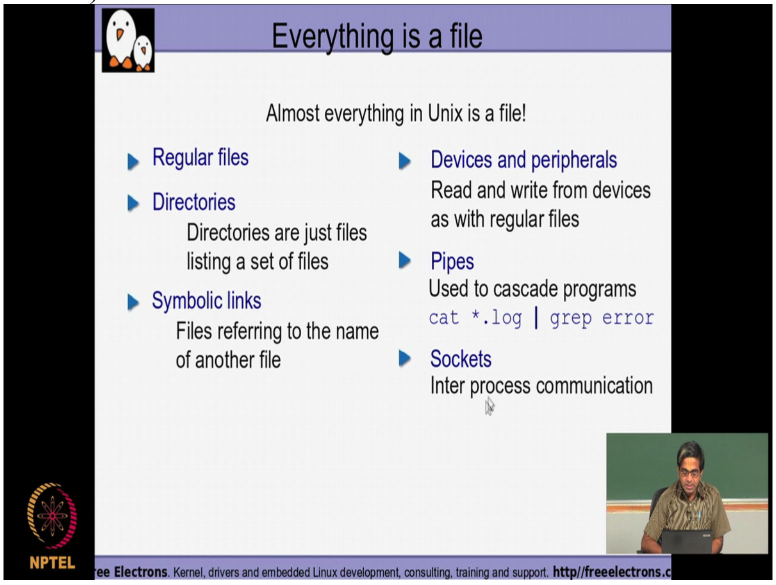
Information Security-3

Sri. Vasan V S, principle consultant
Department of Computer science and Engineering
Indian Institute Of Technology Madras

Basics of Unix and Network Administration
Operating Systems
Mod02 Lecture 13
Module 13: UNIX File system

(Refer Slide Time: 0:21)



Everything is a file

Almost everything in Unix is a file!

- Regular files
- Directories
  Directories are just files listing a set of files
- Symbolic links
  Files referring to the name of another file
- Devices and peripherals
  Read and write from devices as with regular files
- Pipes
  Used to cascade programs
  `cat *.log | grep error`
- Sockets
  Inter process communication

ee Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. http//freeelectrons.c

So starting from this module we are going to be basically looking at, the basics of the Unix file system and start looking at some of the commands that you would (())(0:22) require to work on a Linux base system. So first let us try to define whenever we talk off file, what exactly is a file, from the context of any kind of Linux platform. So in Unix we typically refer to everything as a file, basically because of the fact that file is a handle that is provided to access any kind of data or any kind of peripheral device in the Linux system, right. So what could be the different types of files, there could be regular files, regular files here typically mean , the files in which I store the data and whenever I to retrieve the data, I operate on those files either in terms of simply viewing the data or for copying the data or whatever, right.

Then next comes directories, so as far as unix system in concerned, the directories is also typically files, but they are referred to as one sort of a special file, right. So what exactly is a directory in unix, so for those of us who actually come from the windows world, a directory is nothing but what we use referred to as folder in windows system, right? So directories are typically files and if you try to see the contents of the directory, it will be the list of files that is actually a present and which has been created inside that particular directory.

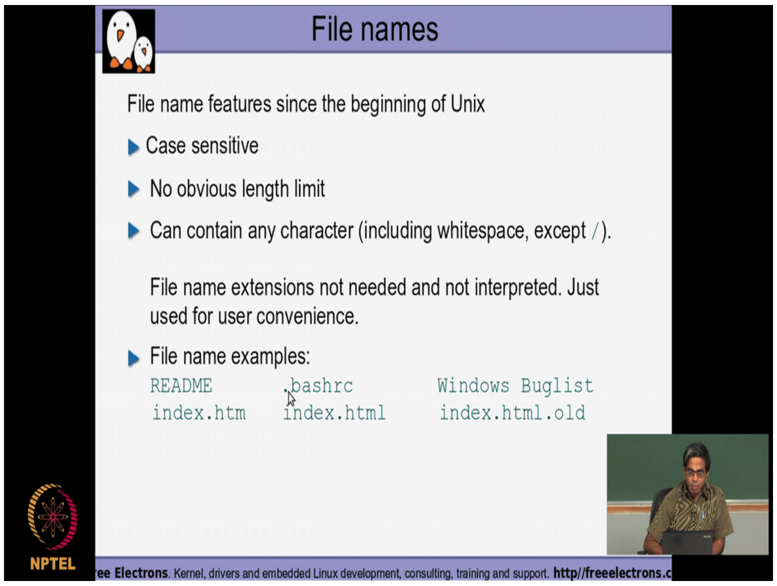
So symbolic links is again another type of a file where it is used to refer to the contents of another file. So again taking example of windows operating system, because most of us would be possibly coming from the windows world, we would have heard of something called as shortcuts, right. so symbolic link is a typically the mechanism by which a shortcuts are really implemented on the unix operating system and we will be basically seeing how the symbolic link typically works, in our subsequent modules and devices and peripherals again are another special type of a file as far as unix is concerned, wherein all the devices that I have in my system, so be it my keyboard, be it my monitor, my pen drive, my mouse, right, my printer, all these are actually considered as a special type of a file and depending on the type of the device, so we will be typically classifying devices as either a character based device or as a block based device, depending on whether we do the input and output byte by byte or as block by block, right.

So depending on the characteristic of the device, I would be appropriately reading and writing, the data from the device. Then pipes is actually another example of a device of a file in unix wherein, I typically take the contents of the outputs of one command in unix and then use the output of this command as an input into the next command, so wherein I will be able to sort of cascade, the execution of the different commands, one after the other and make them related in such a way that the output of the second command is going to be taken as a input for my third command in the command line, right.

So for determining the pipe as we will see later in the subsequent modules we will be actually using this symbol, right. So in this particular example, the output of this particular command is going to be fed in as an input to the second command that on the right hand side of this pipe symbol, right. So just like we have, we are very commonly use water pipes wherein, the water when it is actually getting inside, one end of the pipe is actually go into come out to the other end. Similarly, the output of the first command which is actually getting inside, the pipe from the left hand side of the p6ipe symbol will typically be fed in as the input to the command that is actually on the right hand side of the pipe symbol, right.

Now the next type of a typical file that we have us something called as a sockets. Now what exactly is a socket? We all would have actually heard about client server programming from the context of internet, right. So if the client to (())(5:04) server processes have to communicate across the internet, they should have a mechanism of inter process communication that is possible between these two processes that are typically running on two different systems, right. So what is a mechanism that we use we use the sockets mechanism for that and socket is also, as far as unix is concerned is actually treated as one of the special type of the files. So this again we will actually be seeing in the subsequent module in detail.

(Refer Slide Time: 5:40)



File names

File name features since the beginning of Unix

- Case sensitive
- No obvious length limit
- Can contain any character (including whitespace, except /).

  File name extensions not needed and not interpreted. Just used for user convenience.
- File name examples:

```
README        .bashrc       Windows Buglist
index.htm     index.html    index.html.old
```

ee Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. http//freeelectrons.c

Now coming down to the file names, since we have going to name the file irrespective of whether it is a regular file or a special type of the files that we saw right now. there are certain criteria that needs to be employed , which is again slightly different as compared to what we are used in the windows world, in terms of the file name, contents and the rules for having those file names.

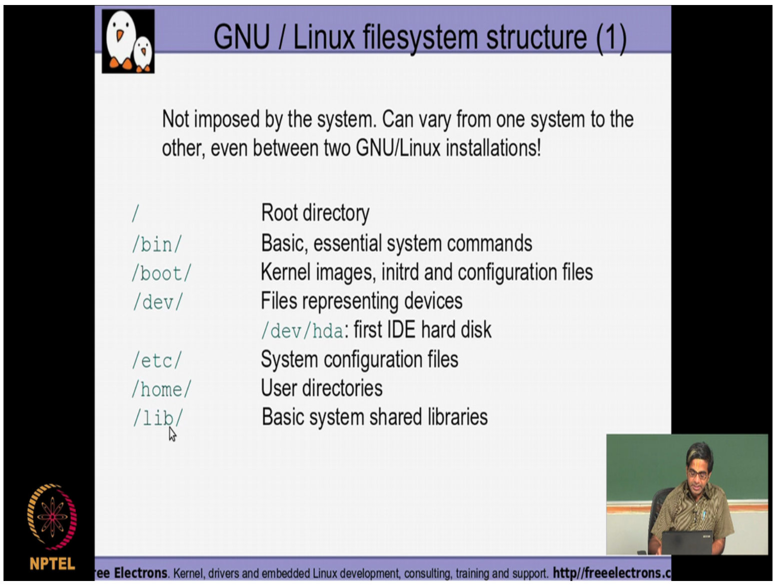
So some of the very important characteristics that we will need to remember, whenever we name a file in the, unix operating system is that that file names are typically case sensitive. So when we say case sensitive, we essentially mean that if you look at this, for example file called README, with the name all in the upper case characters is actually different from the name of the file, README if I have with all the latters in the lower case, so that is essentially one very critical difference in the way we actually name the files and how unix treats a file names, as compared to how it is typically done on the windows platform.

So and secondly, there is no obvious length limit; I could typically have the file names extending to a very large size so that it becomes more readable as compared to having in a restrictions on the name of the file. thirdly file name basically contain any character, so it could really have a white space, it could have the latters in the capital case or in a smaller case in a even in the mix of cases, but one character that it cannot have is basically the slash character, we will see subsequently down below in this module or why this particular character cannot be there, but other than this character you could typically have any character as part of the file name.

So file name extensions are not mandated here, also not interpreted separately and it is just used for some sort of a convenience for sort of easy recollection later on, on what possibly this particular a file is internally containing, right. So the some of the example to the file name are actually mentioned here, so as you see, the first file README, it does not have any extension mentioned, but it is still considered as a valid name as far as unix is concerned and then there is a another file name, Windows Buglist, which actually has a space character, white space character in the middle, but the file name is actually Windows Buglist all put together and it is not to be considered as two different file names, one file name as Windows and another file name separate file name as Buglist, right.

So I could also have two different extensions with the dot character and if you see this example, there of a this is example with the name as dot bashrc. So in Unix, all file names that actually start with dot as a first symbol is actually treated as what is called as a hidden file. So again in windows just for comparison purposes we would have actually come across some files, which are actually marked as hidden, which will not be visible by default, right. So unless and until you enable the view hidden files uh parameter, the files that has been marked as hidden will not be visible in your explorer window. Similarly, any file names that are actually starting with the dot in unix is considered as hidden file, which will not be listed by default unless and until we actually use one very specific option to the unix command.

(Refer Slide Time: 9:22)



GNU / Linux filesystem structure (1)

Not imposed by the system. Can vary from one system to the other, even between two GNU/Linux installations!

/	Root directory
/bin/	Basic, essential system commands
/boot/	Kernel images, initrd and configuration files
/dev/	Files representing devices /dev/hda: first IDE hard disk
/etc/	System configuration files
/home/	User directories
/lib/	Basic system shared libraries

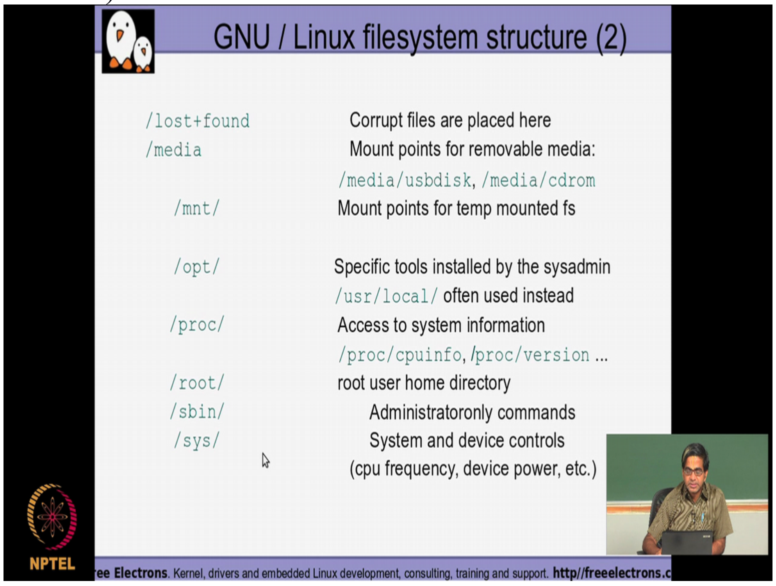
ee Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. http//freeelectrons.c

So how is the entire file system structure going to be and what are that mean components of the file system that is something, which is little important for us to understand, because we would be needing to navigate, my entire file system structure for doing various operations and making use of the OS itself, right. So the slash as we were just discussing in a previous slide is a special character that is actually denoting the root directories. So this is basically the starting point of my entire files system as far as , the different directories and sub-directories in files or concern, so all the files are actually going to be starting from this root directory, which is denoted by this forward slash character, right. There is a directory called bin under root , which essentially contains all the system commands that could really be run by normal users for utilizing the operating system functionalities.

So all common commands are needs to be executed by a typical not-privileged user will be available under this big directory, then there is a next directory called as a boot in which I will have all the kernel images and the configuration files are required for the kernels at the time of boot up of my system. I have a directory called dev, which basically containing all the special files that represent my devices. So as we were discussing in the previous slide in unix, all the devices that I have on my system, whether it be my monitor, whether it be my keyboard, mouse, my printer device, my Wi-Fi , modem, whatever it is my hard disc, all these are actually considered as device special files and all these device special files are actually having an interface under the slash dev directory for them to be accesses, for example, I could have file called had, which is as special device file under this slash dev directory, which could possibly represent my IDE hard discussion, right.

Then there is a another directory called etc under the root, in which I will have all the system configuration files that are required for my system to function properly, I have a directory called home under root, which has the home directories of all the users who have been created on this system. So since unix is basically multiuser operating system on any kind of typical installation, you will find more than one user minimally and each user will have something called as home directory that you will find under the slash home directory in my system, then there is a directory called lib, which basically contains all the sheared libraries that my applications and products that have been installed on a system is going to be actually making use of.

(Refer Slide Time: 12:14)



GNU / Linux filesystem structure (2)

/lost+found	Corrupt files are placed here
/media	Mount points for removable media: /media/usbdisk, /media/cdrom
/mnt/	Mount points for temp mounted fs
/opt/	Specific tools installed by the sysadmin /usr/local/ often used instead
/proc/	Access to system information /proc/cpuinfo, /proc/version ...
/root/	root user home directory
/sbin/	Administratoronly commands
/sys/	System and device controls (cpu frequency, device power, etc.)

ee Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. http//freeelectrons.c

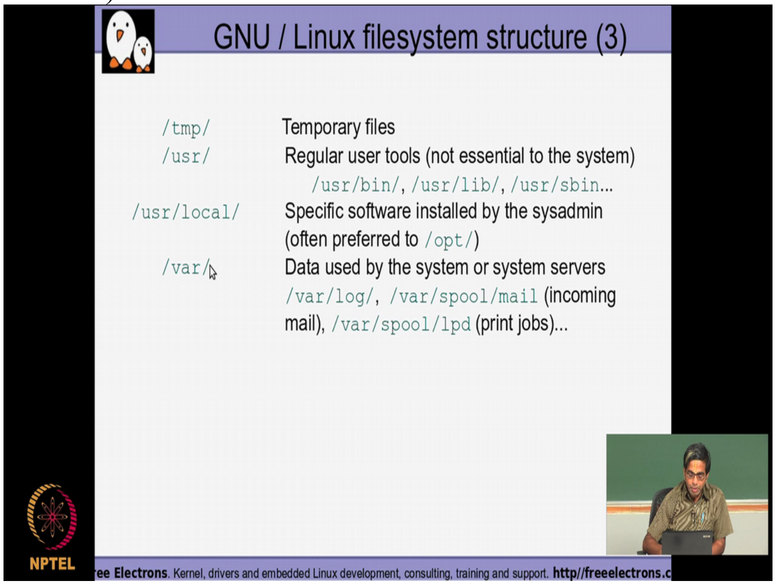
So there is a lost plus found directory, in which any file that has got corrupted will actually be kept for retrieval subsequently. There is a directory called medial under root, which will typically contain the mount points for all the removable media that I would possibly be trying to use on my system, so I could possibly we inserting a CD drive, I could be inserting a pen drive, I could be inserting printer connected on to my USB port. So all these are actually considered as removal media and the slash media directory will typically the containing the mount points of all this removable media before which they can all be made use of.

Then mnt under the slash is basically considered as the mount point for all, the temporary file systems are needs to be mounted. So if I connect for example, a USB hard drive into my USB port and let us say, there are two different file systems available on the hard drive, then under slash mnt, I will have two different directory that is created, each of them being used for mounting each of those partitions in that hard drive. So slash opt is basically used other than slash user slash local wherein all the products that are optionally installed by the system administrator on the system will typically get install by default in this particular location.

So slash proc is basically interface, wherein a any kind of system information like for example, how much of CPU is being used by my system at that instant of time? How much of memory is being used? How much of packets it is actually coming into the system over the network and how much of packets are actually going outer the network? All these kinds of details are exposed by the kernel through the slash proc interface.

So slash root is basically the root users home directory, root is typically the default privileged user on the unix system just like you have the administrator user on your windows system as a privileged user. So sbin is basically the directory where, I have all the commands that are typically used only by the privileged user. So the difference between the bin directory that we saw in the previous slide in sbin is that. Bin directory will contain commands that are used that could be run potentially by even non-privileged normal users, whereas the sbin directory will contain the commands that could be run typically only by the privileged user of the system. So slash sys like my slash proc, we will also be typically used for exposing different kinds of dynamic information depending on the state of my system for the end users reference.

(Refer Slide Time: 15:05)



GNU / Linux filesystem structure (3)

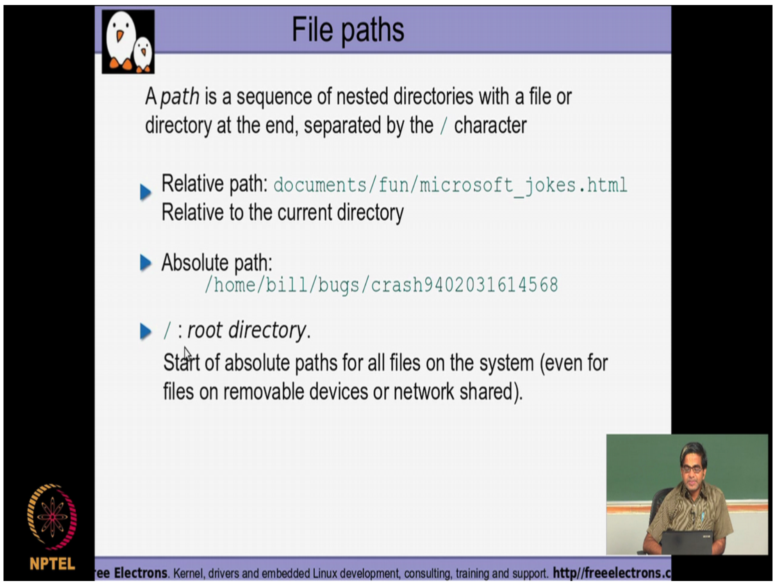
/tmp/	Temporary files
/usr/	Regular user tools (not essential to the system) /usr/bin/, /usr/lib/, /usr/sbin...
/usr/local/	Specific software installed by the sysadmin (often preferred to /opt/)
/var/	Data used by the system or system servers /var/log/, /var/spool/mail (incoming mail), /var/spool/lpd (print jobs)...

ee Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. http//freeelectrons.c

So slash tmp is basically the location where my temporary files will be all stored, this particular directory location will be cleaned of when the system is getting restarted. So any kind of data that you want to be stored permanently especially across reboot to the system should not be stored in this particular slash tmp location. Slash user will be used to store any kind of user based tools, so it could have user bin, user lib, user sbin and so on. So all these directories will be containing commands that have been actually required or installed by user based applications, user specific applications that has been installed on a per user basis.

So user local is an alternate directory, where any kind of customize software that is getting installed on that particular system could be getting installed, var is basically a directory, where all kinds of log or any kinds of mail related files for that particular system will be getting stored and just like opt and slash user local, the slash var directory is also typically used by the optional products that are installed on the system by the system administrator.

(Refer Slide Time: 16:21)



File paths

A *path* is a sequence of nested directories with a file or directory at the end, separated by the / character

- Relative path: `documents/fun/microsoft_jokes.html`
  Relative to the current directory
- Absolute path:
  `/home/bill/bugs/crash9402031614568`
- / : *root directory*.
  Start of absolute paths for all files on the system (even for files on removable devices or network shared).

NPTEL
ee Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. http//freeelectrons.c

Now coming down into the file paths. So is very important to understand how can we access a file depending on where the file is located, so for that unix define something called as a file path. So a path is basically a sequence of nested directories with a file or a directory at the end, separated by the slash character. Now why is the slash character chosen as the delimiter or the separator between the different parts of my path is basically, because as we just saw the slash is basically the root of the file system and because of the fact that it is actually used as the root of the file system, the same slash character is also used for differentiating between the different components of my path.

So whenever we refer to a path, there are two different possibilities here, one what is called as relative path and another, what is called as an absolute path, right. Now the relative path is identified whenever the path is actually specified without the leading slash, right. So the relative path for a particular file name if it does not start with the slash character that means it is actually relative to the current directory. So if I am in my particular home directory and if I refer to the path as like this, document slash fun slash Microsoft underscore jokes dot html, I am basically telling the OS that I am having a directory called documents in my current directory, under that I have directory called fun, under that there is a file called Microsoft underscore jokes dot html, which is to be referred by me right now, okay so that is a reason why we are calling, this is relative path again relative path would be in that my path is not actually starting with slash character.

The absolute path is something that irrespective of whatever is my current location, I will be referring to the file name in absolute path starting from the root directory. So if you see in the absolute path, in this particular example, you find that it is starting with the slash character, right. So I am basically saying with this absolute path that under slash, there is a directory called home under this home, there is something called as a bill, there is a directory called bill, under this particular directory called bill, there is a sub-directory called bugs, under that bugs I have a this particular file, right.

So I have given here an absolute path, wherein irrespective of wherever I am currently located, as my current working directory, if I start the file path with the slash, it will only go and start looking at the root of the tree and from the root of the tree, it will expect that whatever has been specified as a path name to be valid path name. So valid path name here essentially means that that particular path is existing and is valid without which it will not be able to access the contents.

So if you try to give the absolute path or relative path and if that particular path is non-existed, okay then it will come back and report an error to you saying that there is no such file or directory. So as we were saying the slash is basically denoting the root directory and that is how I will have all the absolute paths specified for my particular system. So we come to the end of module 13, so we will look at the subsequent modules wherein we will look at the individual commands in greater detail.