Information Security

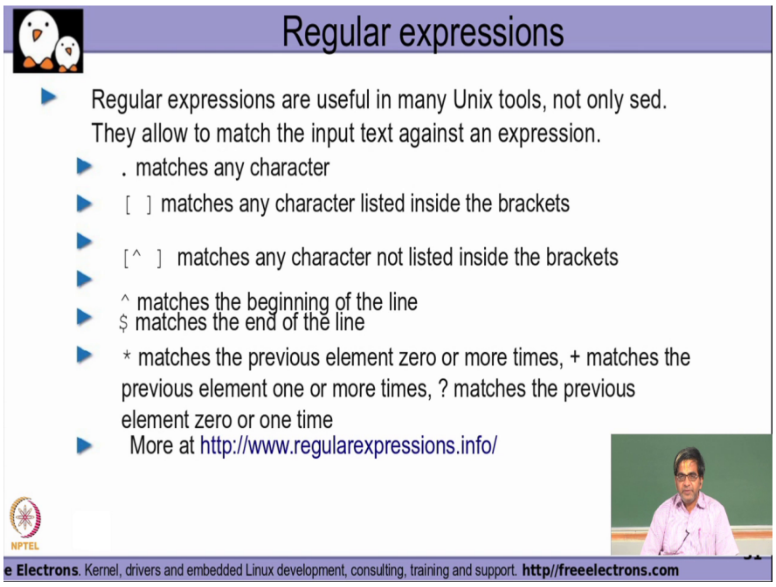
Shri Vasan V S,
Principal Consultant
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Module 16

LINUX Basic Commands (Continued)

So in this module we will continue our discussion on the basic commands after looking at what are the different options available for us to do the regular expression and use the different regular expression characters for making very very powerful searches are very very powerful a replacement.

(Refer Slide Time: 00:33)



Regular expressions

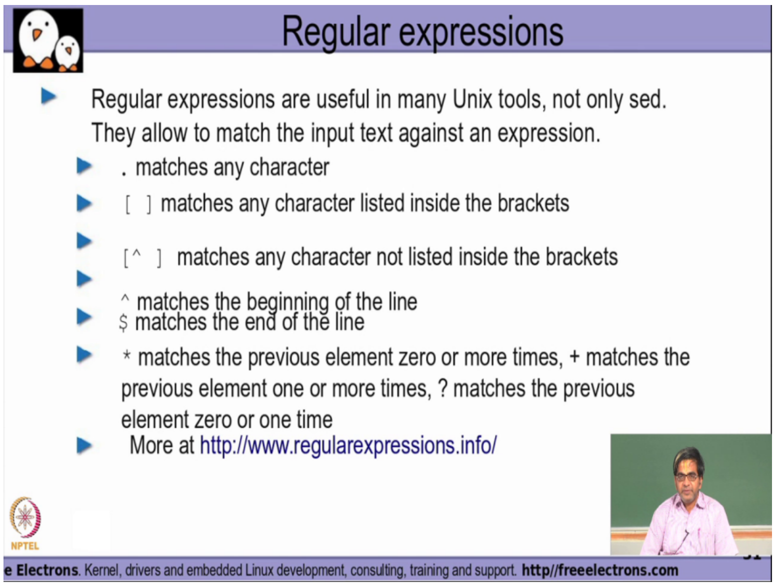
- Regular expressions are useful in many Unix tools, not only sed. They allow to match the input text against an expression.
  - . matches any character
  - [ ] matches any character listed inside the brackets
  - [^] matches any character not listed inside the brackets
  - ^ matches the beginning of the line
  - $ matches the end of the line
  - * matches the previous element zero or more times, + matches the previous element one or more times, ? matches the previous element zero or one time
  - More at http://www.regularexpressions.info/

e Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. http//freeelectrons.com

So regular expressions are used typically in various UNIX tools and commands and we just saw an example of it being used in the set command and LS command in our previous models. So what is a regular expression actually do they really try to allow us to match different kinds of input text that is possible where in the input text will actually be getting dynamically expanded at runtime irrespective of whatever are the contents of that particular location in which that expansion is trying to happen.

So for example if I try to say that I want to use a regular expression star as we have seen some of the examples in a previous modules, so if I say star txt I wouldn't know at any point in time whether there are two files ending with txt or 5 files ending with t x t or any number of n number of files does exist ending with t x t, right? But on the other hand whenever I am trying to use that regular expression in my script or in the command line whatever it is, the star is getting expanded and automatically getting replaced with the contents as it is standing at that instant of time. So my script or my command when I use regular expression does not need to change depending on the contents of my directory or the pattern where ever I am actually trying to use regular expression for, right?

(Refer Slide Time: 02:13)



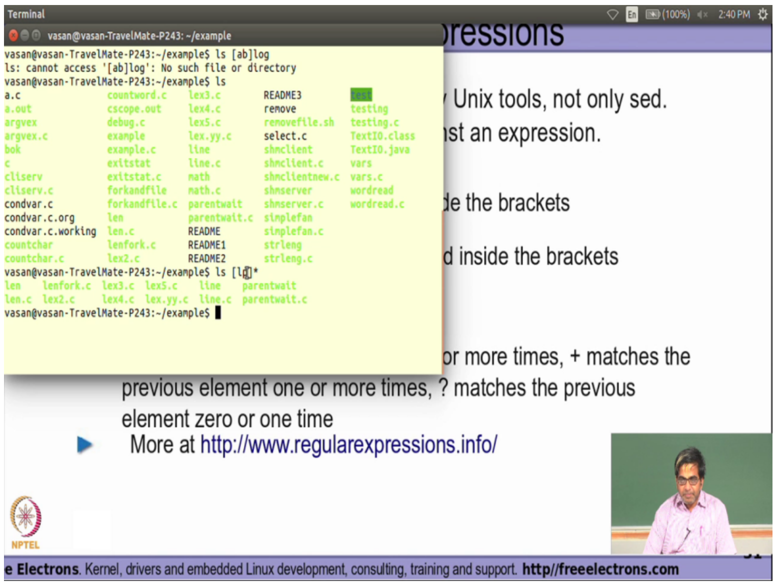
Regular expressions

- Regular expressions are useful in many Unix tools, not only sed. They allow to match the input text against an expression.
  - . matches any character
  - [ ] matches any character listed inside the brackets
  - [^] matches any character not listed inside the brackets
  - ^ matches the beginning of the line
  - $ matches the end of the line
  - * matches the previous element zero or more times, + matches the previous element one or more times, ? matches the previous element zero or one time
  - More at http://www.regularexpressions.info/

e Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. http//freeelectrons.com

So the very common regular expressions are typically used is basically a dot, a dot will match any character so I could have any character in that particular location where I have specified dot in my entire regular expression. So the open square bracket and the closed square brackets will typically match any one character listed inside the square brackets, right?

(Refer Slide Time: 02:50)



```
vasan@vasan-TravelMate-P243:~/example$ ls [ab]log
ls: cannot access '[ab]log': No such file or directory
vasan@vasan-TravelMate-P243:~/example$ ls
a.c               countword.c    lex3.c        README3         test
a.out             cscope.out     lex4.c        remove          testing
argvex            debug.c        lex5.c        removefile.sh   testing.c
argvex.c          example        lex.yy.c      select.c        TextIO.class
bok               example.c      line          shmclient       TextIO.java
c                 exitstat       line.c        shmclient.c     vars
cliserv           exitstat.c     math          shmclientnew.c  vars.c
cliserv.c         forkandfile    math.c        shmserver       wordread
condvar.c         forkandfile.c  parentwait    shmserver.c     wordread.c
condvar.c.org     len            parentwait.c  simplefan
condvar.c.working len.c          README        simplefan.c
countchar         lenfork.c      README1       strleng
countchar.c       lex2.c         README2       strleng.c
vasan@vasan-TravelMate-P243:~/example$ ls [lp]*
len    lenfork.c  lex3.c  lex5.c    line    parentwait
len.c  lex2.c     lex4.c  lex.yy.c  line.c  parentwait.c
vasan@vasan-TravelMate-P243:~/example$ 
```

ressions

Unix tools, not only sed.

ist an expression.

de the brackets

d inside the brackets

or more times, + matches the previous element one or more times, ? matches the previous element zero or one time

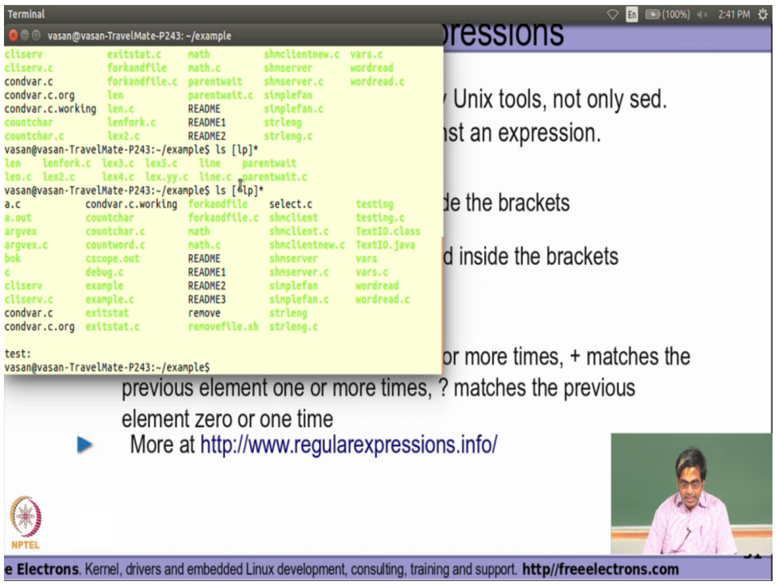
- More at http://www.regularexpressions.info/

e Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. http//freeelectrons.com

So for example if I say that I want to look at a file name starting with a or b I will specify that the filename could be like here I don't have any file that is present here. So if I for example say l p star what I am basically trying to tell here is that I want to list all the files with beginning with either l or beginning with either p followed by any number of characters because a star is used, right? So it will display to me all the files in my current directory which is either beginning with L or beginning with p.

So that is basically what is square brackets is all about. So replace any one character with specified within the square brackets and expand the given regular expression to see what kind of pattern is matching. Next is basically within a square brackets I specify the caret symbol which which essentially means that whatever characters are following this particular symbol those characters should not be present in the pattern matching, right? So this is basically and negation of what the normal open square brackets closed square brackets that we discussed here. So its basically the negations behaviour of this particular regular expression.

(Refer Slide Time: 04:18)



```
Terminal
vasan@vasan-TravelMate-P243: ~/example
cliserv          exitstat.c     math            shmclientnew.c  vars.c
cliserv.c        forkandfile    math.c          shmserver       wordread
condvar.c        forkandfile.c  parentwait      shmserver.c     wordread.c
condvar.c.org    len            parentwait.c    simplefan
condvar.c.working len.c         README          simplefan.c
countchar        lenfork.c      README1         strleng
countchar.c      lex2.c         README2         strleng.c
vasan@vasan-TravelMate-P243:~/example$ ls [lp]*
len    lenfork.c  lex3.c  lex5.c    line    parentwait
len.c  lex2.c     lex4.c  lex.yy.c  line.c  parentwait.c
vasan@vasan-TravelMate-P243:~/example$ ls [^lp]*
a.c              condvar.c.working  forkandfile    select.c        testing
a.out            countchar          forkandfile.c  shmclient       testing.c
argvex           countchar.c        math           shmclient.c     TextIO.class
argvex.c         countword.c        math.c         shmclientnew.c  TextIO.java
bok              cscope.out         README         shmserver       vars
c                debug.c            README1        shmserver.c     vars.c
cliserv          example            README2        simplefan       wordread
cliserv.c        example.c          README3        simplefan.c     wordread.c
condvar.c        exitstat           remove         strleng
condvar.c.org    exitstat.c         removefile.sh  strleng.c

test:
vasan@vasan-TravelMate-P243:~/example$
```

ressions

Unix tools, not only sed.
st an expression.

de the brackets

d inside the brackets

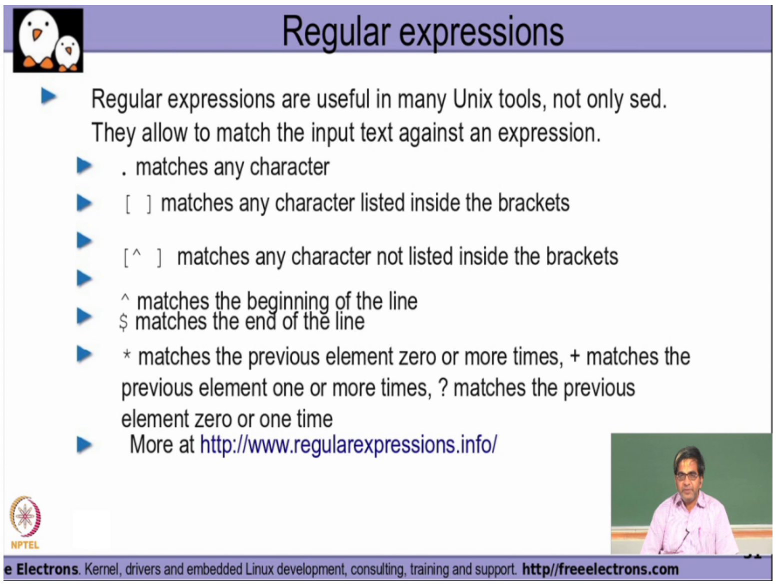
or more times, + matches the
previous element one or more times, ? matches the previous
element zero or one time

More at http://www.regularexpressions.info/

e Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. http//freeelectrons.com

So if you see this here if I specify it like this it will display to me all files and directories which is not beginning with L and P. So whatever you saw here as the contents previously for my the previous in vocational regular expression you will not find the same contents here because the negation is actually been asked for saying that the first character should neither be a l nor be a P in this particular regular expression.

(Refer Slide Time: 04:52)



Regular expressions

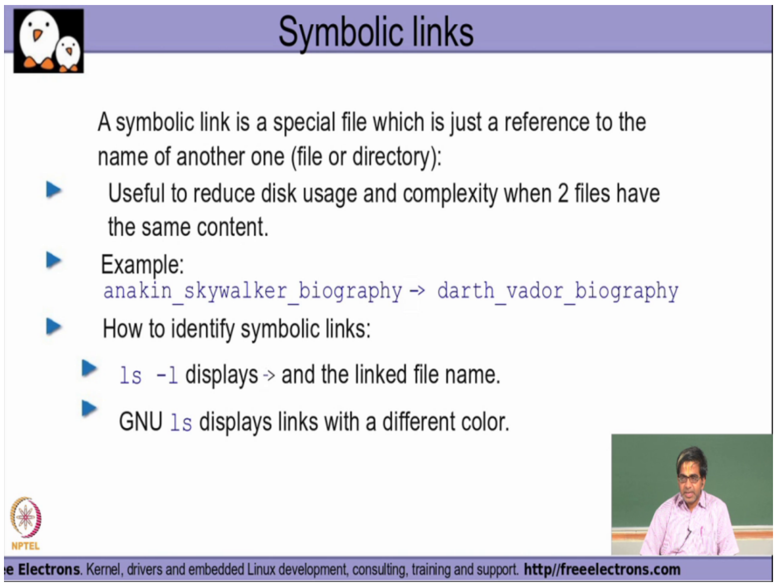
- Regular expressions are useful in many Unix tools, not only sed. They allow to match the input text against an expression.
  - . matches any character
  - [ ] matches any character listed inside the brackets
  - [^] matches any character not listed inside the brackets
  - ^ matches the beginning of the line
  - $ matches the end of the line
  - * matches the previous element zero or more times, + matches the previous element one or more times, ? matches the previous element zero or one time
  - More at http://www.regularexpressions.info/

e Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. http//freeelectrons.com

So on the other hand if I use the caret symbol without putting it within the square brackets then that means the the pattern that is following the correct symbol should be the beginning of the line for it to match the regular expression. Dollar similarly would mean end of the line and star would as we have already seen few examples in our previous modules and also in the previous example that I showed you here. Star would typically mean 0 or more times plus would mean that 1 or more times and Question mark would mean 0 or 1 time.

So question mark Here would essentially mean that there should be only one character that that is present in that particular location. So regular expressions as I was telling you it's a very very powerful very very useful tool for doing lot of very complex task and it also is very handy for us when scripting and more details of the regular expressions can be found in this particular URL site this is Justin sample and there are really lot of other sites which also talks in detail about regular expressions. So we would strongly suggest that you actually go through some of those sites as well and then get very comfortable the regular expressions because you will find it really very very handy going forward.

(Refer Slide Time: 06:18)



Symbolic links

A symbolic link is a special file which is just a reference to the name of another one (file or directory):

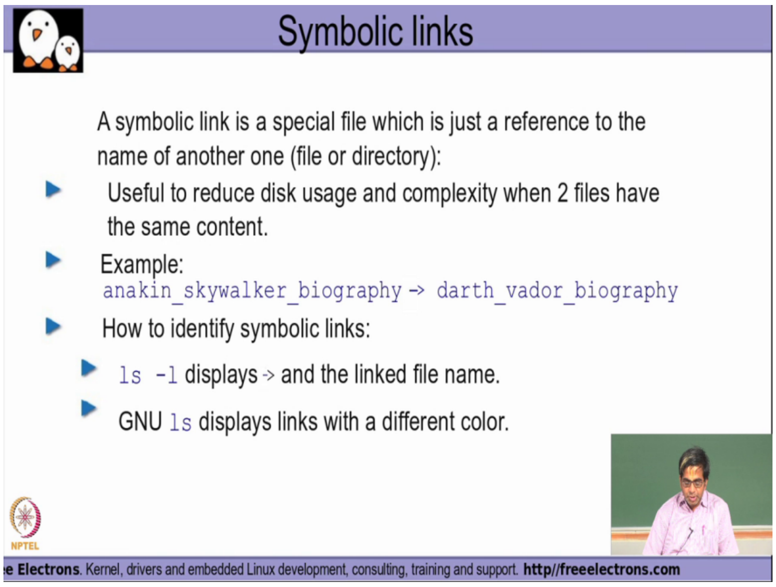
- Useful to reduce disk usage and complexity when 2 files have the same content.
- Example:
  `anakin_skywalker_biography -> darth_vador_biography`
- How to identify symbolic links:
  - `ls -l` displays -> and the linked file name.
  - GNU `ls` displays links with a different color.

e Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. http//freeelectrons.com

Now coming down to the next Basic concept that we will be talking about is what is called as a symbolic link. So as we were talking in the initial module a link in the UNIX parlance is something very similar to what we actually call as a shortcut the desktop and the windows world, right? So are typically for those of us who come from the Windows background we refer to the something called as shortcut, where in you want to really have a shortcut created typically in our desktop screen for a file or directory or some content that is actually available somewhere in our hard disc in a very embedded location right?

So what we mean by embedded location is the maybe it is actually available in lot of depth in my entire file system and I don't want to really waste my time in trying to click on or trying to migrate to each of those of those sub directory locations before accessing my content. So likewise the same behaviour of a shortcut in Windows is what is called as actually a link in the lakes world.

(Refer Slide Time: 07:40)



Symbolic links

A symbolic link is a special file which is just a reference to the name of another one (file or directory):

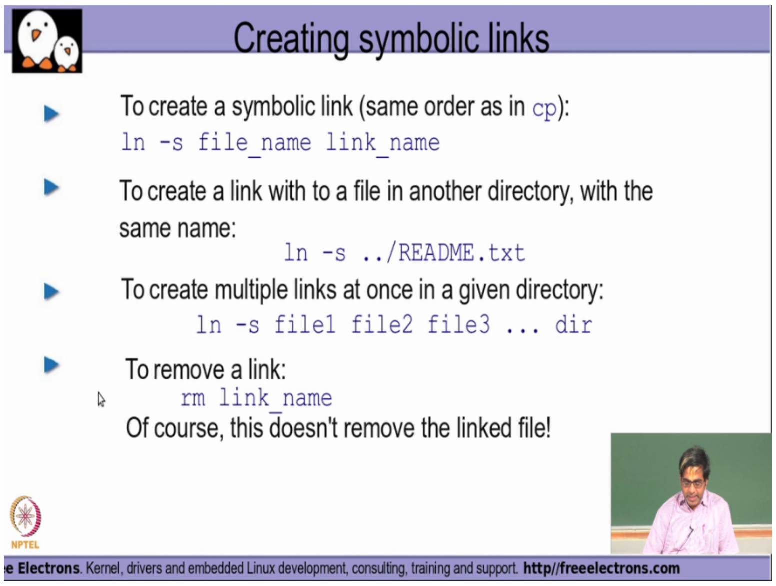
- Useful to reduce disk usage and complexity when 2 files have the same content.
- Example:
  `anakin_skywalker_biography -> darth_vador_biography`
- How to identify symbolic links:
  - `ls -l` displays -> and the linked file name.
  - GNU `ls` displays links with a different color.

e Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. http//freeelectrons.com

A symbolic link is basically special file which is actually just a reference to the name of another file or directory, right? So this is typically used for reduce the disk space occupied and as well as a complexity when two files is actually going to have the same content unnecessarily wasting the disc space if I really try to have it as two different files only one file will be stored and second name which with which I want Access the same content will be created as a link to my original file within which I have my content, right?

So whenever I use symbolic link if those links are typically denoted ls minus l output as hyphen followed by the greater than symbol and then followed by whatever is a linked file name. So if we actually use JNU LS LS version of the command it also will display it in a different colour.

(Refer Slide Time: 08:34)



Creating symbolic links

- To create a symbolic link (same order as in `cp`):
  `ln -s file_name link_name`
- To create a link with to a file in another directory, with the same name:
  `ln -s ../README.txt`
- To create multiple links at once in a given directory:
  `ln -s file1 file2 file3 ... dir`
- To remove a link:
  `rm link_name`
  Of course, this doesn't remove the linked file!

e Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. http//freeelectrons.com

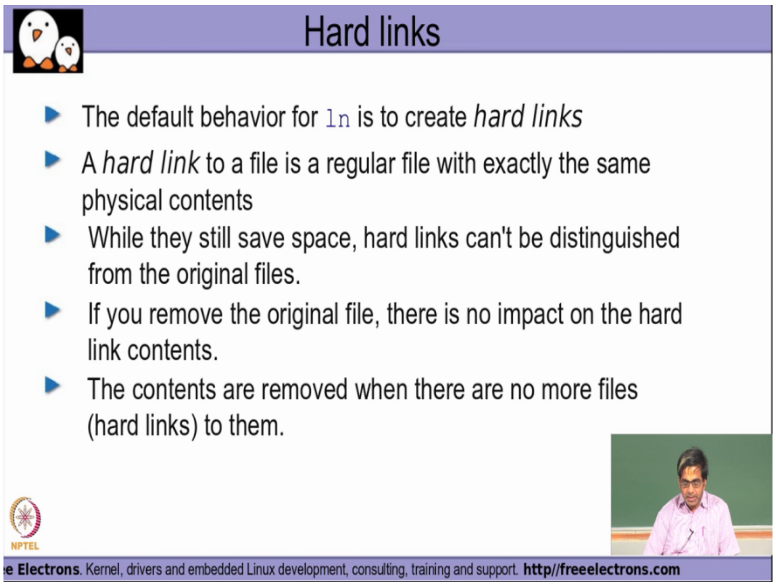
So how do I create a symbolic link I specify use LN command and then specify minus s option for creating a soft link where in I specify first the filename whatever is a filename to which I want the link to be created, and then follow it up the link name? So the link name will be the name which would be created as the pointer to the original file name that is actually created here.

So if I want to create a file create a link with the same name as a filename then I just don't specify the link name explicitly I just specify the filename alone. So in this particular case there is a file called readme.txt in my parent directory with that is the reason why we are actually use dot / readme.txt and with this command since the command has not been given the link name explicitly what is going to happen is that there is going to be a link on readme.txt that is going to be created in the current directory with the same name, right?

So to create multiple links at once in a given directory I could actually use multiple file name arguments followed finally by the directory in which I want to create the link so each of these files will actually be created as a link in the directory that I am actually mentioned finally in my LN command line and for removing a link since the link is also a special type of a file as we saw in of a previous modules I just need to use this standard RM command that we would use for a typical file.

So like I use RM command for typical file I use the RM command for the link name also and then if I use RM followed by the link name that particular command umm that particular file will be getting deleted from my file system hierarchy.

(Refer Slide Time: 10:40)



Hard links

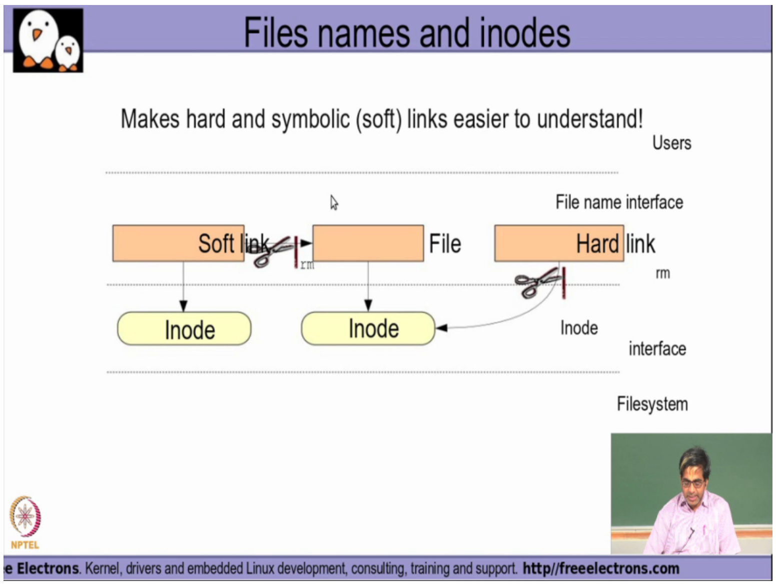
- The default behavior for `ln` is to create *hard links*
- A *hard link* to a file is a regular file with exactly the same physical contents
- While they still save space, hard links can't be distinguished from the original files.
- If you remove the original file, there is no impact on the hard link contents.
- The contents are removed when there are no more files (hard links) to them.

e **Electrons**. Kernel, drivers and embedded Linux development, consulting, training and support. **http//freeelectrons.com**

So another type of a link is basically what is called as hard link where the default behaviour pattern command is actually created as a hard link only. Now the hard link is it is a like a regular file that contains are actually getting pointed to the same physical contents of the original source file, right?

So the difference between a hard link and soft link is if I basically remove the original file as far as I hardly contents are concerned there will be no impact the hard link contents and the contents will be finally removed only if there are no file names that are actually pointing to the contents in when when when they have been created as a hard link, right?

(Refer Slide Time: 11:32)



Files names and inodes

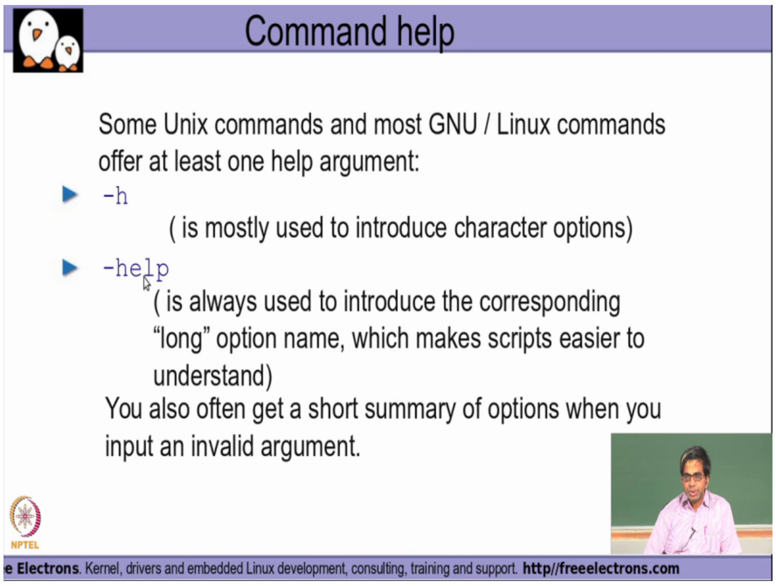
Makes hard and symbolic (soft) links easier to understand!

e Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. http//freeelectrons.com

So in this case what I could really actually try to do here is that when I use this soft link because they are going to be using the i nodes in a file systems. I node is basically the interface that I really have within a file system. So every file that I have created will actually have umm I node Data Structure inside the file system so I node as we would have seen in our file system module as part of the initial OS description stands for index node and there will be a file there will be I node that will be actually available for every file that is created. So this I node will have meta data associated is a file as well as the pointer to the data blocks in which data for a data does blocks in which the data associated with this file is actually stored inside, right?

So in case of a soft link what is actually happening is that I will have a soft link with the different file name pointing which actually has a different I node created whereas when I actually have a hard link created a hard link is also a link pointing to the same I node where in the number of links is actually incremented which is presented which is one of the information that is actually presented inside the I node and so if I remove the hard link name unless and until the number of references in that I node comes to zero the contents of that file which has been pointed by that i node will never be deleted by the file system, right? So that's basically the core difference between soft link and a hard link.

(Refer Slide Time: 13:29)



Command help

Some Unix commands and most GNU / Linux commands offer at least one help argument:

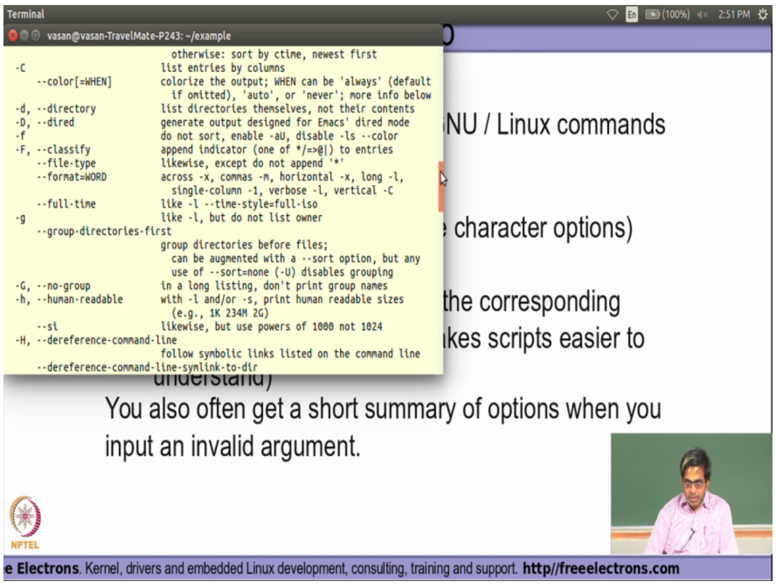
- `-h`
  (is mostly used to introduce character options)
- `-help`
  (is always used to introduce the corresponding "long" option name, which makes scripts easier to understand)

You also often get a short summary of options when you input an invalid argument.

e Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. http//freeelectrons.com

So whenever we need any kind of additional details on how to use a particular command options or there are typical options available either minus H or minus L on almost all the Unix commands, so if you really use minus H option or the minus L option it will give you details about how many options are there what each of those options mean and what kind of usage are required for each of those options all those details will be given to you.

(Refer Slide Time: 14:07)



```
vasan@vasan-TravelMate-P243: ~/example
                              otherwise: sort by ctime, newest first
  -C                        list entries by columns
      --color[=WHEN]        colorize the output; WHEN can be 'always' (default
                              if omitted), 'auto', or 'never'; more info below
  -d, --directory           list directories themselves, not their contents
  -D, --dired               generate output designed for Emacs' dired mode
  -f                        do not sort, enable -aU, disable -ls --color
  -F, --classify            append indicator (one of */=>@|) to entries
      --file-type           likewise, except do not append '*'
      --format=WORD         across -x, commas -m, horizontal -x, long -l,
                              single-column -1, verbose -l, vertical -C
      --full-time           like -l --time-style=full-iso
  -g                        like -l, but do not list owner
      --group-directories-first
                            group directories before files;
                              can be augmented with a --sort option, but any
                              use of --sort=none (-U) disables grouping
  -G, --no-group            in a long listing, don't print group names
  -h, --human-readable      with -l and/or -s, print human readable sizes
                              (e.g., 1K 234M 2G)
      --si                  likewise, but use powers of 1000 not 1024
  -H, --dereference-command-line
                            follow symbolic links listed on the command line
      --dereference-command-line-symlink-to-dir
```

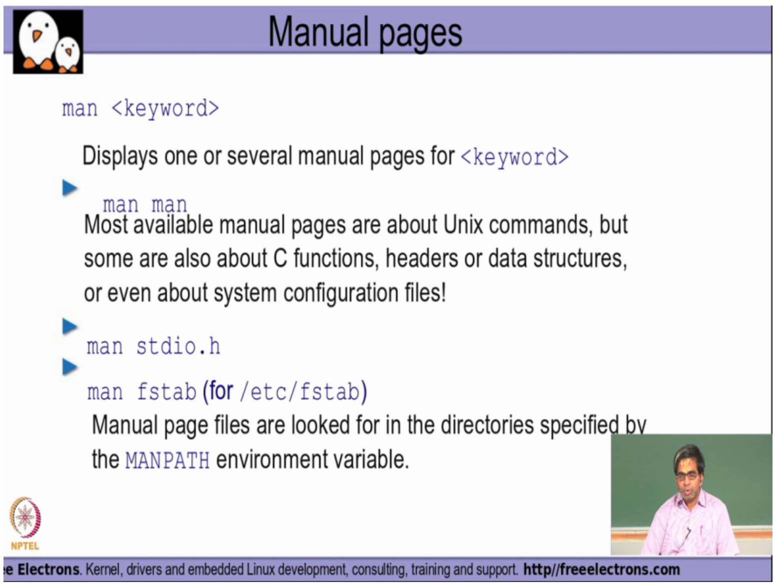
You also often get a short summary of options when you input an invalid argument.

e Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. http//freeelectrons.com

So for example if you really want to find out how to use a command called let us say it displays the different kinds of options that is available here and tells me for each of those options what is the corresponding explanations of what those each of the options really mean.

So in this particular case minus H actually stands for human readable format so likewise there will be some examples some minimum examples of the day of the Unix commands where in the minus H option would be used for doing some other implementation of a specific feature within the command but predominantly if you see the almost all the commands will have the minus H or the minus L option to give you more details about what that particular command is doing and what are the features of the individual options available within the command.

(Refer Slide Time: 16:05)



Manual pages

`man <keyword>`

Displays one or several manual pages for `<keyword>`

- `man man`

Most available manual pages are about Unix commands, but some are also about C functions, headers or data structures, or even about system configuration files!

- `man stdio.h`
- `man fstab` (for `/etc/fstab`)

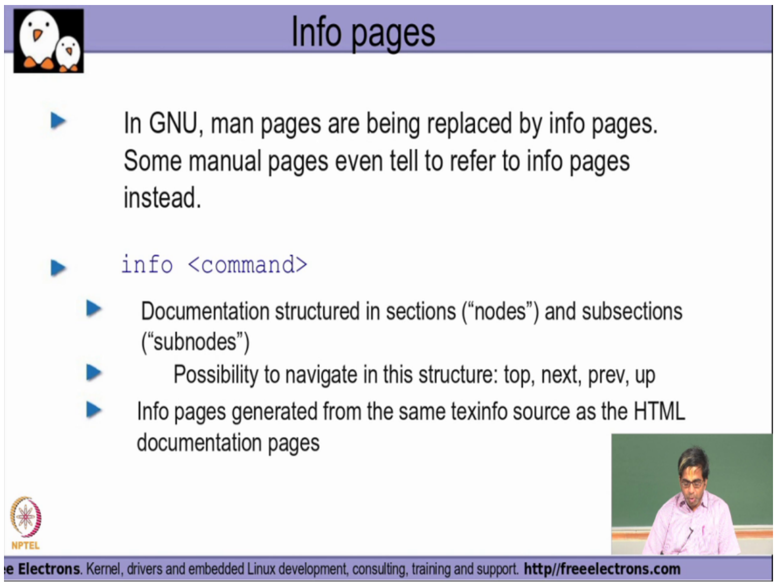
Manual page files are looked for in the directories specified by the `MANPATH` environment variable.

e Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. **http//freeelectrons.com**

So in the other hand if you want to know details about a command the weight has to be used the what is the behaviour of the command? what kind of error messages would possibly come from the command and all that you have what is called the Mac Command so you can just type the man followed by the keyboard whatever Command or the key that I would that you want more details on it will display you the complete set of health related information for that particular argument that you are given to the man Command, right?

So it it tells you what are the different ways of using it how each of those options are arguments changes the behaviour the command what kind of error messages you could typically expect to get from the command and also what are the other related commands to the one that is actually getting displayed right now. So it is actually extremely useful for us as a sort of a cheat sheet to have a quick reference whenever we have to sort of recapsulate on what are the different options available the behaviour of each of those options and so on.

(Refer Slide Time: 16:13)



Info pages

- In GNU, man pages are being replaced by info pages. Some manual pages even tell to refer to info pages instead.
- `info <command>`
  - Documentation structured in sections ("nodes") and subsections ("subnodes")
  - Possibility to navigate in this structure: top, next, prev, up
  - Info pages generated from the same texinfo source as the HTML documentation pages

e Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. http//freeelectrons.com

Similarly you also have the info command which gives you more details as compared to what a man command actually tells you and it also will tell you the associated command that is actually available and any kind of a relationship or dependency of the given command with those associated commands.

So essentially gives you more details about how to use the commands? What is the behaviour? What are the other related commands? And So on and So forth Man and info are typically two useful commands that will give you additional information on how a particular command could be used by you including the options including different kinds of arguments that it might take and behave differently.

Thank You!