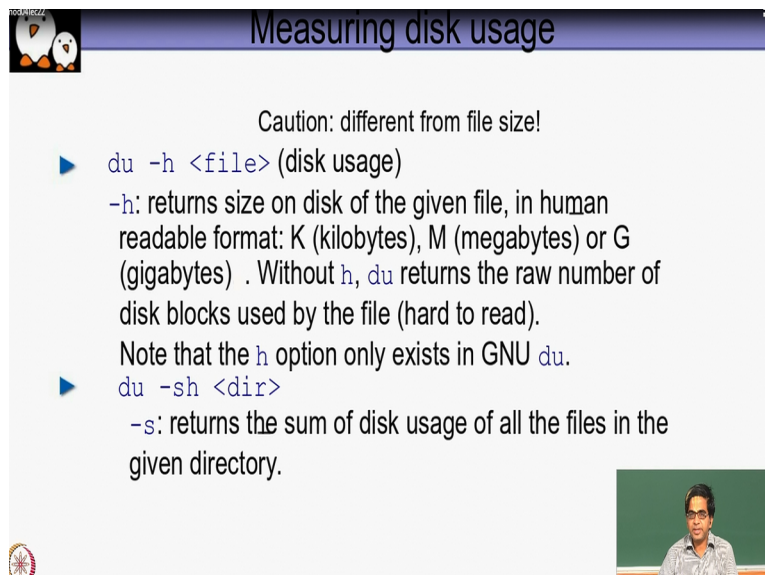


**Information Security**  
**Shri Vasan V S, Principal Consultant**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology Madras**  
**Mod04lec22**  
**LINUX : Compression/Archiving**

In this module we will be specifically looking at the different types of compression or archiving tools that are actually available on a Linux System. So what is exactly compression or archiving and why is it so important for us to understand. So when we actually have a data file, if the size of the data file is pretty small, we generally wouldn't be too much concerned about the fact that the size is small, because it is not really be going to be consuming too much of disc space for us.

But when we actually deal with files that are running into, lets say, megabytes or gigabytes of data, then there will be situations where, unless and until we sort of compress the data and get a compressed file, we wouldn't be very comfortable with actually having that particular huge file occupying so much of our disc space. So for that simple reason, you have different kinds of compression and archiving mechanism possible in the Linux system and we will see in this particular module what the different kinds of tools are, how do they really work and what are the benefits associated with them.

(Refer Slide Time: 01:25)



Caution: different from file size!

- ▶ `du -h <file>` (disk usage)  
-h: returns size on disk of the given file, in human readable format: K (kilobytes), M (megabytes) or G (gigabytes) . Without h, du returns the raw number of disk blocks used by the file (hard to read).  
Note that the h option only exists in GNU du.
- ▶ `du -sh <dir>`  
-s: returns the sum of disk usage of all the files in the given directory.

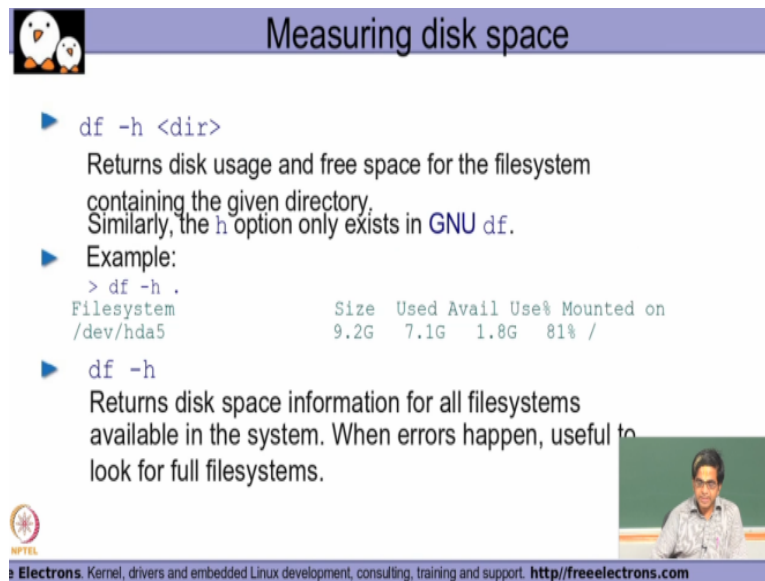
So before we actually go ahead and try to run any of this compression tool, one of the first things we need to understand we will have to first get an idea of how much of disc space is actually used by certain files and once the the the file size cross a certain threshold, then we would possibly be deciding that we need to go ahead and apply the compression tool on that file so as to make it occupy much lesser space.

Now there is a command called du which basically helps us to understand how much of disc size is being used by a single file or a set of files put together. So if you really try to run minus h option with the du command and give the file name also for which you are interested to know how much size is being occupied it will basically report to you what is the size of the file, either in kilobytes, or megabytes, or gigabytes.

So that it basically becomes very easier for us to quickly understand how much is the size of the file rather than printing it in a normal bytes format. So if you don't use minus h option, the du command will return the raw number of disc blocks this file is using and because of the fact that it is giving the number for the size in terms of the disc block, we need to also know what is a disc block size for us to understand; exactly how much of bytes that file is occupying.

So in that respect, it is a little bit hard to read so typically you will find the administrators or the users who want to use the du command always use this command with the minus s option so that the file size or the directory size whatever is being looked at basically becomes easier and human readable very quickly. So if you also use a minus -s option it displays the sum of the disc usage of all the files in a given directory so that also helps us to get an idea instead of a file, a particular directory how much size is being occupied by that particular directory contents on my disc partition.



(Refer Slide Time: 03:37)



### Measuring disk space

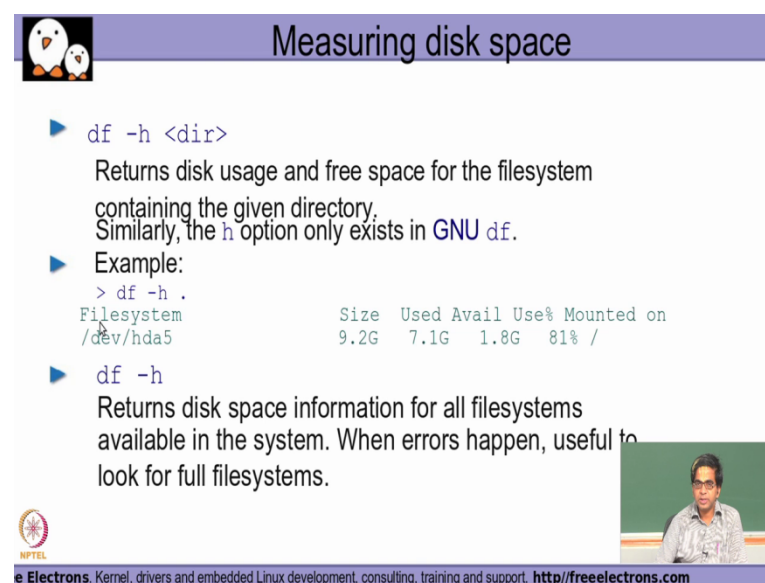
- ▶ `df -h <dir>`  
Returns disk usage and free space for the filesystem containing the given directory. Similarly, the `h` option only exists in GNU `df`.
- ▶ Example:  

```
> df -h .  
Filesystem      Size  Used Avail Use% Mounted on  
/dev/hda5       9.2G  7.1G  1.8G  81% /
```
- ▶ `df -h`  
Returns disk space information for all filesystems available in the system. When errors happen, useful to look for full filesystems.

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

So the other way of looking at the disc space is by running a `df` command, so it basically tells me not only the amount of space that has been occupied, but it also gives me how much of free space is available for the file system. The `du` command will only give the space that has been occupied but the `df` will also additionally give me details on a per file system basis how much is being occupied and how much is actually free space that is available. So `df` is again you will find and also experience is one of the very powerful commands whenever we need to know the current status of our disc partition utilisation.



(Refer Slide Time: 04:19)



### Measuring disk space

- ▶ `df -h <dir>`  
Returns disk usage and free space for the filesystem containing the given directory. Similarly, the `h` option only exists in GNU `df`.
- ▶ Example:  

```
> df -h .  
Filesystem      Size  Used Avail Use% Mounted on  
/dev/hda5       9.2G  7.1G  1.8G  81% /
```
- ▶ `df -h`  
Returns disk space information for all filesystems available in the system. When errors happen, useful to look for full filesystems.

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

So when it says that df output is like given as below the file system that is being marked here is the partition name dev/hda5 that essentially means this the fifth partition of my disc hda that has been detected in my system and slash dev as we had seen in our previous module is basically the interface that my Unix system expose to have all the devices exposed to the user space so my file system on slash dev slash hda5 the total size is 9.2 GB as transfer gigabytes.

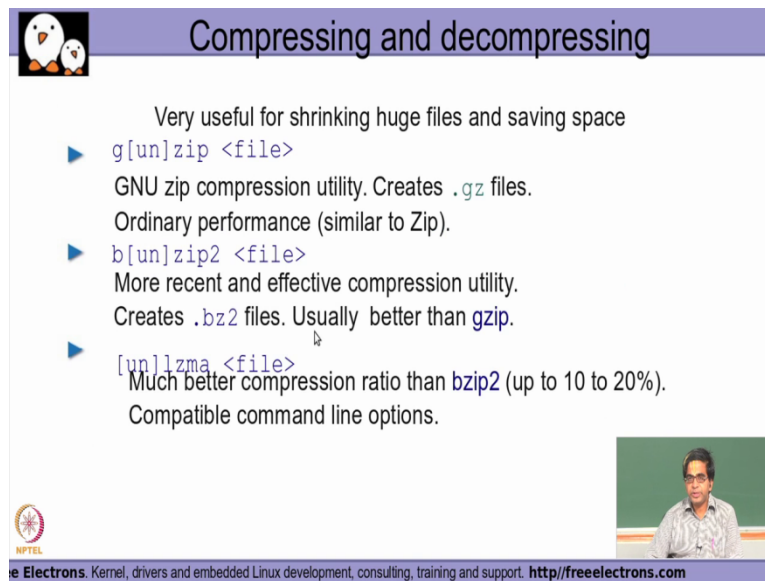
So the total size is 9.2 GB out of which 7.1 GB is currently being used and what is available is 1.8 GB so that is basically the amount of free space that I have and the use percentage is like around 81 percentage is being used for the total capacity and the main point is basically on the root right so this particular partition assistant is there on this partition is mounted as my root file system right now.

So that is basically what this particular line is explaining to the user now if you see the used plus available capacity in comes to rounds. 7.1 plus 1.8 GB in this particular example 8.9 GB whereas the total size of the partition is mentioned as 9.2 GB so we will have to understand as we had seen in the file system modules earlier some part of the file system space will be going for storing the metadata file system related to the file system.

So you will always find that on every system the total size that has been given for the file system will not be the complete usable capacity but usable capacity will be a small percentage lesser than the total available capacity for this reason because the small chunk of my entire file system space is actually occupied for storing the metadata related to that file system and that space that is occupied by the metadata is not available for the end user to store his or her own data in that portion.

So you'll always find that my used plus the available capacity in a file system is slightly lesser as compared to what is a total size of the file system that a command like a DF command reports to us.

(Refer Slide Time: 06:42)



**Compressing and decompressing**

Very useful for shrinking huge files and saving space

- ▶ `g[un]zip <file>`  
GNU zip compression utility. Creates `.gz` files.  
Ordinary performance (similar to Zip).
- ▶ `b[un]zip2 <file>`  
More recent and effective compression utility.  
Creates `.bz2` files. Usually better than `gzip`.
- ▶ `[un]lzma <file>`  
Much better compression ratio than `bzip2` (up to 10 to 20%).  
Compatible command line options.

NPTEL  
Free Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. <http://freeelectrons.com>

Now having looked at how much disc size a file is occupying and also looked at how much of space my file system is having the DF command it will possibly be time for us to do some kind of an operation to reduce the amount of disc space that our files are actually occupying in our partition so how do we do with that is basically why we have compression tools that are there so you have 1 compression tool what is called as gzip.

So if you say gzip followed by the filename that file will be compressed using the gzip compression algorithm which is the standard algorithm that is actually available in a publicly known format. So it will basically compress the file if the compression is successful it will create a dot gz file for every file that you have given in the command line to be compressed once it creates a dot gz file that is a compressed file of the given file successfully the original file will be deleted automatically by this command.

So all compression algorithms will basically delete the original file which is given to it because the intent behind running a compression utility is only to save up on the disc space. So if this compression command is not going to delete the original file what is going to happen is compression command is going to create a new file which is a compressed file and the original file also if it is retained.

So also it is going to sort of not meet our objective of occupying reduce space but because of the new addition of the compressed file in addition to the original file you are going to end up occupying more space so all the compression utility that you find you will find automatically

on a successful completion of compression on the compression of the file original that was given to the compression utility as an argument will be deleted you will only have the compressed file.

In this case if you run the `gzip` command to find and `G` and `UN` in brackets here what it essentially means is `gzip` is the command to compress and `G` `unzip` write a single word without any spaces nothing so `G` answer to the single command that you will use to uncompress or decompress right so you have `gzip` followed by the filename you go ahead take the file read the file and do the compression create the dot `GZ` file and create the file name.

If it is able to successfully create the dot `GZ` file and when you want to uncompress it whenever later on if you want the process the original file you can do on compression of that file by running the command called and `g(un)zip` is again as I was mentioning it is a single word so you just use `g(un)zip` and in short we are commonly called as `gunzip`.

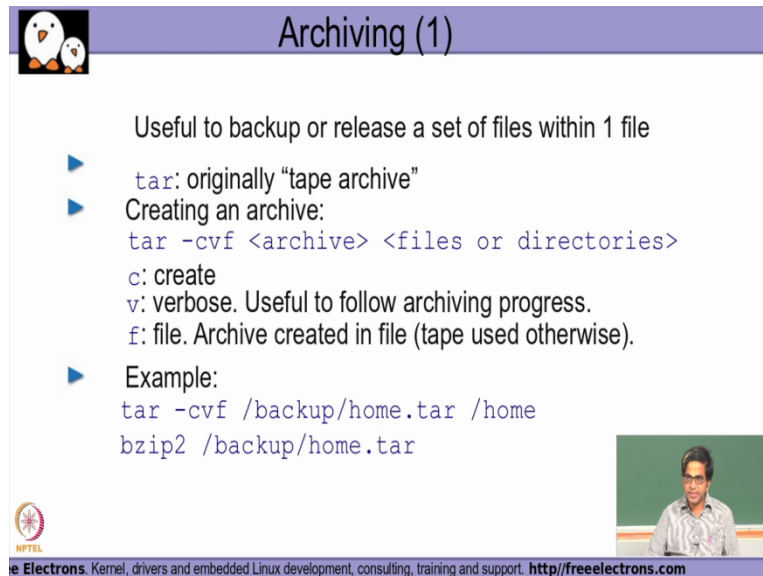
Similarly the another compression utility called `bzip2` which also does compression of the file and the compression percentage factor in `bzip2` is better as compared to the `gzip` algorithm the `gzip` compression algorithm came out first and then more further announcements to the compression algorithm was done which had actually been subsequently got released as a `bzip2` to compression algorithm if you want to have better compression ratio you can also use the `bzip2` to compression utility and `bzip2` compression utility will create a dot `bz2` file now this dot `bz2` file is essentially meaning whatever the filename that is given here.

Suppose if I give `cornal src` as a filename if `bzip2` compression a successful it will now create a file called `cornal src dot bz2` right similarly here if i had given that same file for `gzip` it would create a file `cornal src dot gz` file so that is basically my compressed file so in addition to these two algorithms i also have another algorithm called `lzma` which is possibly not be there by default in most of the distribution.

But this is tool that could optionally be downloaded and installed if we require this particular compression in evolution is little bit better compression ratio as compared to even `bzip2` right, for example you will get approximately 10 to 20% more compression done with this particular `lzma` compression algorithm as even compared to `bzip2` the for the purposes of archiving. When we do when do we do archiving when whenever we come to a decision that a particular directory or a particular project is no longer going to be used and I am going to

sort of only keep referring to wait very rarely I wouldn't want to actually consume my disc space unnecessarily for that.

(Refer Slide Time: 12:00)



**Archiving (1)**

Useful to backup or release a set of files within 1 file

- ▶ `tar`: originally "tape archive"
- ▶ Creating an archive:  
`tar -cvf <archive> <files or directories>`  
c: create  
v: verbose. Useful to follow archiving progress.  
f: file. Archive created in file (tape used otherwise).
- ▶ Example:  
`tar -cvf /backup/home.tar /home`  
`bzip2 /backup/home.tar`

NPTEL

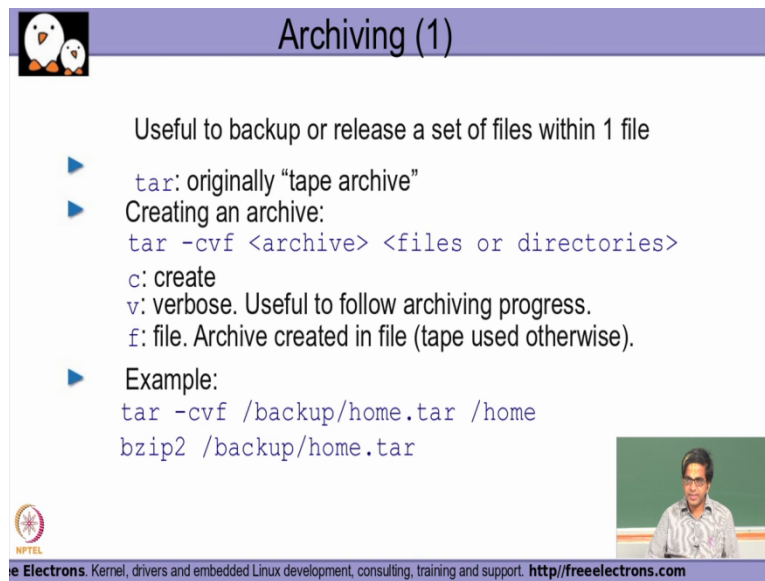
Free Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. <http://freeelectrons.com>

So what we do what we do and archiving of the entire directory folder which is no more going to be used and most common command it is actually used for doing this archiving on Linux is what is called as the tar command. So the tar command is actually originally stood for tape archive. So in the olden days whenever archiving was done because they didn't have too much of a disc space in our primary storage medium.

Then the entire archiving would be done on the data would be actually moved into an offline tape right but since nowadays even the archive data is actually available as part of our primary storage medium we will find only very very rarely the tape medium being used but historically unconventionally the tar retain its original acronym for standing for tape archive.

And we basically make use of this tar command to do the archiving I run the command with options of minus cvf and give the archived name followed by whatever are the file or directory is the needs to be archived so I hear the minus c in this c the option C option basically stands for create instruction to the tar command to create an archive file the v option basically stands for verbose with this verbose option the tar command keeps giving me details of what exactly is trying to do whenever that are archiving process is on.

(Refer Slide Time: 13:00)



**Archiving (1)**

Useful to backup or release a set of files within 1 file

- ▶ `tar`: originally "tape archive"
- ▶ Creating an archive:  
`tar -cvf <archive> <files or directories>`  
c: create  
v: verbose. Useful to follow archiving progress.  
f: file. Archive created in file (tape used otherwise).
- ▶ Example:  
`tar -cvf /backup/home.tar /home`  
`bzip2 /backup/home.tar`

Free Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. <http://freeelectrons.com>

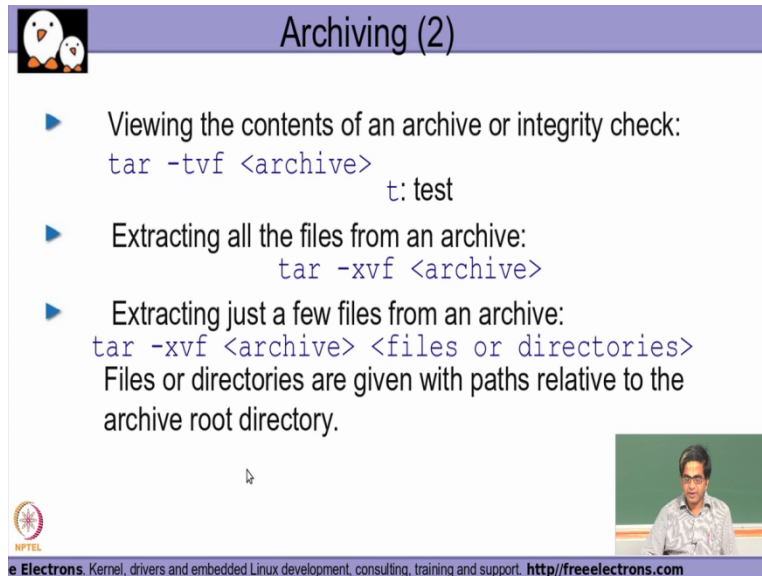
So it will keep telling me that I have now taken this file and archive in next file and I have time and so on and so forth and my output will be continuously getting refreshed right then the f option is basically to specify what is a archive filename that is going to be created and the second argument that is actually mentioned here what are the setup files are directories that needs to be archive into this archive file name so if u see this example we set tar minus cvf and we specify what is the filename directory name that we want to have finally and then secondly we specify what is a directory that we want to archive.

So slash home is a directory that you want Archives all the contents recursively under slash home we want to tar command to archive and finally place it under this particular file home.tar in slash backup location. So all these absolute path in everything by and actually seen our previous modules so this home dot tar file.

If the tar command successful is going to be getting created under the slash backup directory and since I expect the tar like to be consuming lot of disc space I want want to compress it for example with bzip2 command that we saw in the previous slide and so I run bzip2 command and then give the tar filename as a file to be compressed right now at the end of it is bzip2 is a successful without any errors that it is encountering what I will end up getting is a file called home dot tar dot bz2 in slash backup directory so as before the the compressed file when you run the bzip2 command will be created with a dot bzip2 to extension so in this case is you are giving the filename as home dot tar will basically create a file called home dot dot bz2 in the same location of slash backup.



(Refer Slide Time: 15:43)



**Archiving (2)**

- ▶ Viewing the contents of an archive or integrity check:  
`tar -tvf <archive>`  
t: test
- ▶ Extracting all the files from an archive:  
`tar -xvf <archive>`
- ▶ Extracting just a few files from an archive:  
`tar -xvf <archive> <files or directories>`  
Files or directories are given with paths relative to the archive root directory.

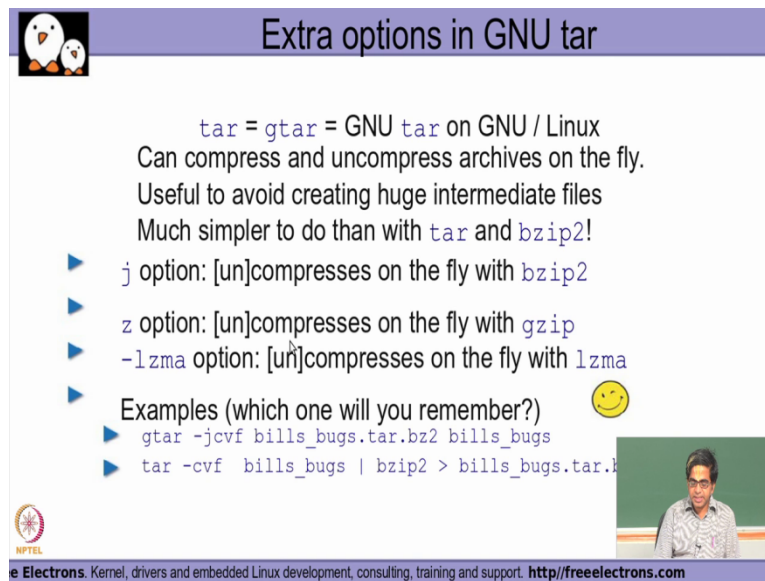
NPTEL  
Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. <http://freeelectrons.com>

So now after having compressed to if I want to verify whether all the content that I really wanted to be archived into this file is available or not I will be using the t option to the tar command so v and f these two options are the same behaviour as we want this on a previous slide but the t option is basically it stands for either test or table of contents now whenever we run the tar minus TVF and give the archive file name it will list down whatever are the files that have been put inside that particular archive file that has been given as an argument.

So but it will not really extract the contents from the archive now at some point in time in future if for some reason you want to refer to the files in the archive and also maybe modify any of them we need to extract source file from the archive for that we actually make use of the x. option actually stands for extract similarly we also would be using the v and f option reasons which we have seen with the other tar examples so when I say tar minus xvf X standing for extract of file and also give this archive file name.

All the individual files that were archived into this particular file will now be extracted and then placed in the current directory right but on the other hand if I had an intention only to extract only few files from the archive not the entire set of files that are available i could also specified As given in this particular user scenarios what are the exact set of files and directories that I want to be extracting alone and in this particular scenario only those files from this archive will be extracted and not everything else.

(Refer Slide Time: 17:42)



**Extra options in GNU tar**

`tar = gtar = GNU tar` on GNU / Linux  
Can compress and uncompress archives on the fly.  
Useful to avoid creating huge intermediate files  
Much simpler to do than with `tar` and `bzip2`!

- ▶ `j` option: [un]compresses on the fly with `bzip2`
- ▶ `z` option: [un]compresses on the fly with `gzip`
- ▶ `-lzma` option: [un]compresses on the fly with `lzma`

Examples (which one will you remember?) 😊

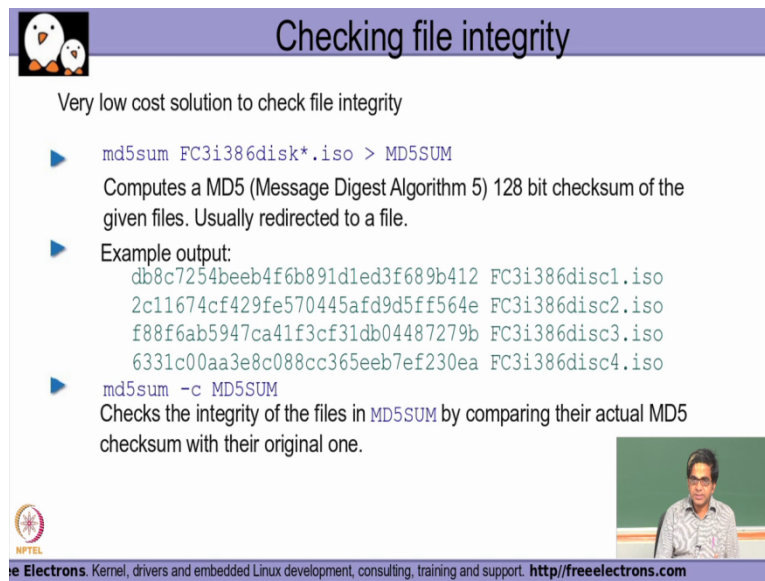
- ▶ `gtar -jcvf bills_bugs.tar.bz2 bills_bugs`
- ▶ `tar -cvf bills_bugs | bzip2 > bills_bugs.tar.bz2`

NPTEL  
Free Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. <http://freeelectrons.com>

So if I use basically the `i` `g` and `u` `tar` command which is the default as it is available on more storage in Linux distributions I could also use the `j` and `z` option as part of my `xvf` to directly enter from `bzip2` compression compressed file or a `gzip` compression file. So for example if I basically do and say minus `cvf` if it will now create a file that I have given here as archive file it will create archive from this particular directory to a `bzip2` to compression on it and then create this file together similarly I could also extract dynamically.

When I use the `X` option here directly from his `bz2` file instead of first doing and compression and then doing and extraction separately in two different commands this kind of an enhancement is actually available if I am actually trying to make use of the GNU version of the `tar` command which is basically what most of the Linux distributions today is containing the wether this option is available with your distribution or not you could actually check up by observing the main page of the `tar` command and we had actually discuss how to use a man page in one of the previous module that we had looked at earlier.

(Refer Slide Time: 19:14)



**Checking file integrity**

Very low cost solution to check file integrity

- ▶ `md5sum FC3i386disk*.iso > MD5SUM`  
Computes a MD5 (Message Digest Algorithm 5) 128 bit checksum of the given files. Usually redirected to a file.
- ▶ Example output:  
`db8c7254beeb4f6b891d1ed3f689b412 FC3i386disc1.iso`  
`2c11674cf429fe570445afd9d5ff564e FC3i386disc2.iso`  
`f88f6ab5947ca41f3cf31db04487279b FC3i386disc3.iso`  
`6331c00aa3e8c088cc365eeb7ef230ea FC3i386disc4.iso`
- ▶ `md5sum -c MD5SUM`  
Checks the integrity of the files in `MD5SUM` by comparing their actual MD5 checksum with their original one.

Free Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. <http://freeelectrons.com>

So i want to really find out whether has there been any corruptions in the file that I am right now trying to use as compared to the original file are there is a command called md5sum so the md5 basically stands for a very popular a Data integrity algorithm called as a message digest 5 and this md5sum command basically implement md5 algorithm inside which will ensure which sort of help us to understand whether there has been any kind of corruption or modification of the file as compared to the original file right.

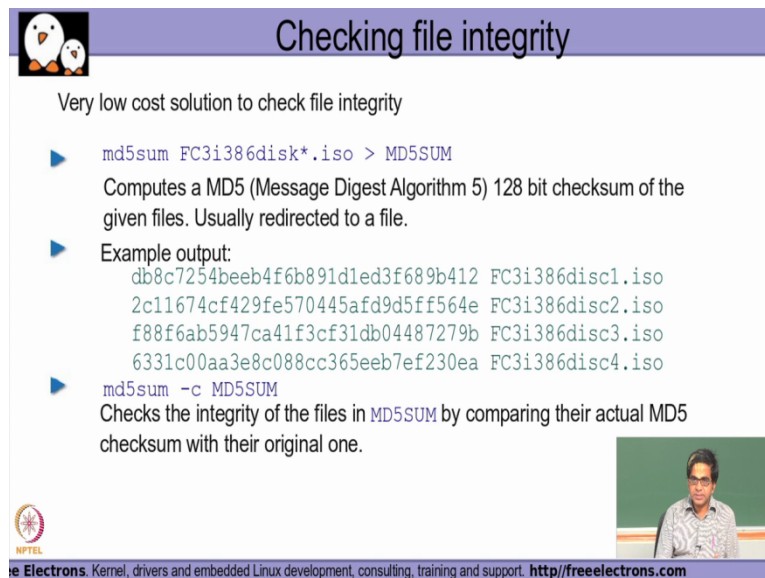
So the way we actually make use of it is if you are really for example downloading a very huge file from the Internet and the original author of the that particular file who has allowed you the download will run this particular md5sum command and then give you a hash value that has actually been generated by this command along along with the file to be downloaded.

Now after this file in order to ensure that as part of the download over the Internet no corruption or modification is actually happened you will also be expected to run the same md5sum at your end locally on the same downloaded file and if there's been no modification or corruption is actually happened the output that you are getting as the output of the md5sum command running locally on your local system should exactly match byte by byte the md5sum value that the provider of that file as actually put.

And these two values match you can very safely assume that there has been no corruption or modification the file as part of your downloaded transfer or if there's been a modification you

are basically discard this downloaded file and sort of try to re download it again with the hope that this time you will get the original file in track.

(Refer Slide Time: 21:20)



### Checking file integrity

Very low cost solution to check file integrity

- ▶ `md5sum FC3i386disk*.iso > MD5SUM`  
Computes a MD5 (Message Digest Algorithm 5) 128 bit checksum of the given files. Usually redirected to a file.
- ▶ Example output:  
`db8c7254beeb4f6b891d1ed3f689b412 FC3i386disc1.iso`  
`2c11674cf429fe570445afd9d5ff564e FC3i386disc2.iso`  
`f88f6ab5947ca41f3cf31db04487279b FC3i386disc3.iso`  
`6331c00aa3e8c088cc365eeb7ef230ea FC3i386disc4.iso`
- ▶ `md5sum -c MD5SUM`  
Checks the integrity of the files in `MD5SUM` by comparing their actual MD5 checksum with their original one.

Free Electrons. Kernel, drivers and embedded Linux development, consulting, training and support. <http://freeelectrons.com>

So this is basically a command that used to ensure that the file that you are actually download it over the Internet especially files that are very very huge in size has not got accidentally corrupted or modified before you really start using that file so that you actually end up saving up on some time wherein you will be able to detect any kind of corruption well before hand before you actually start using the file. So md5sum command is basically used for this particular purpose to check your file or the data integrity.

Thank you!