Server System Management - Linux

Lab03 – File mamangement

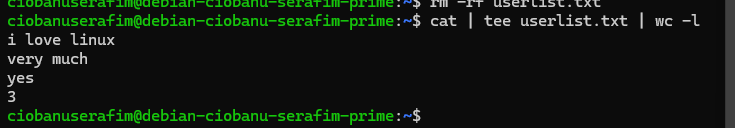
# 

# Lab03: File management

1. (its ok to do this lab entirely as root) Write a command that uses pipes and the commands cat, tee and wc IN THAT ORDER, which lets you input some data that must be written to userlist.txt. Input is terminated on a final empty line with Ctrl-D and immediately, the number of lines must be shown. (QUIZ)

cat | tee userlist.txt | wc -l

cat - | tee userlist.txt | wc -l



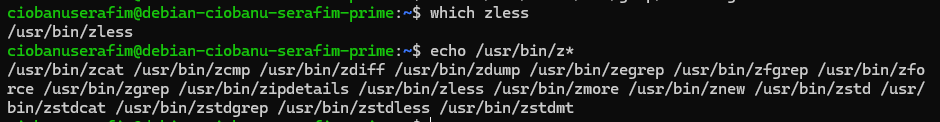
1. Write a command that uses gunzip (use the -c option to send the output to stdout rather than remove the .gz extension from the original file), together with a pipe and less, to view the contents of the compressed file /usr/share/doc/gzip/README.gz.

gunzip -c /usr/share/doc/gzip/README.gz | less

1. Read man zless and use it to accomplish the same thing.

zless /usr/share/doc/gzip/README.gz

Use which to find where zless is on your system. Now use a echo command to use shell expansion (bash), to show you all filenames starting with the letter z, in the same folder as where zless is on your system.



which zless - /usr/bin/zless

echo /usr/bin/z\*

**Important**

Try the same echo command on a Windows system to realize that in linux, the shell does the expansion;

in Windows, the expansion is done by the actual command.

1. Go to the folder /tmp and write a command that uses a pipe of wget, g(un)zip and tar to download https://www.dropbox.com/download/?plat=lnx.x86\_64 and immediately gunzip it (also using -c), and to - in turn - pipe that output to tar to unpack the contents of the file.

**Note**

tar expects a filename to unpack. Often, commands allow you to specify - as the filename, which will make it read its stdin input rather than a file. This is what you need.

wget -qO- "https://www.dropbox.com/download/?plat=lnx.x86\_64" | gunzip -c | tar xf -

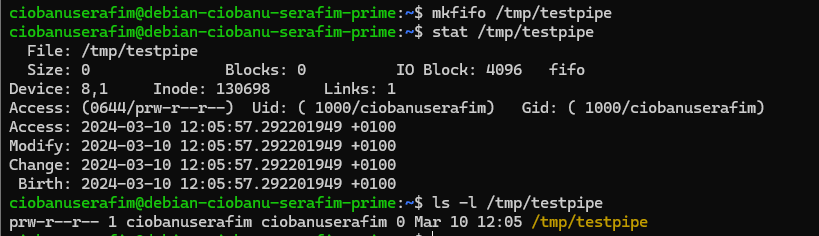
1. Use ls -l /dev to see what is in /dev. Which file types are these? Use man ls or man stat to find the necessary info.

There are present C (Character special files), B (Block special file), L (Links), D (Directory).

1. Create a named pipe /tmp/testpipe using mkfifo. Investigate this special file using ls -l or stat, and compare with what you saw in /dev. What would these kinds of file be useful for ? (QUIZ)

mkfifo /tmp/testpipe

ls –l /tmp/testpipe

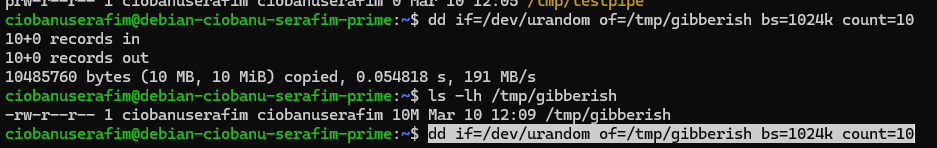


Named pipes are useful for scenarios where you want to establish communication between different processes without having to write data to disk.

1. Read man dd. Now use it to read data from the special file /dev/urandom and write it to a file /tmp/gibberish. Specify block size 1024k and 10 blocks. How large is the resulting file ?

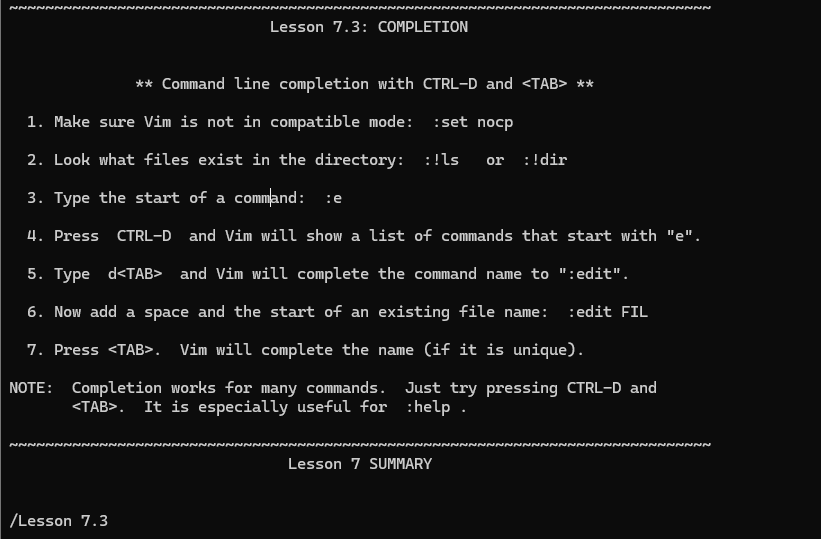
Use ls -lh (and read the man page if you do not know what -h does).

dd if=/dev/urandom of=/tmp/gibberish bs=1024k count=10



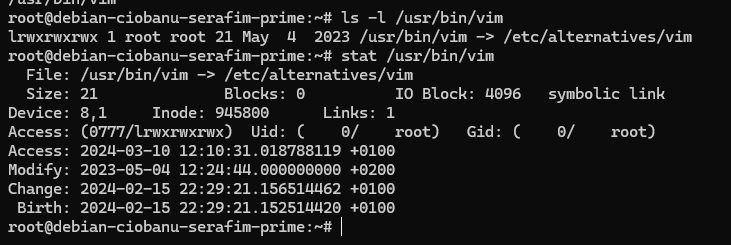
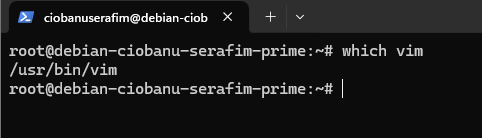
10MB, obviously because we allocated 10 blocks of 1024 KB

8. Use apt update to update your package repository. Now type **apt install vim** to install a powerful text editor. This also installs vimtutor ! **Launch vimtutor** and complete this short introduction to familiarize yourself with some of the more frequently used functions of vim. What is the 3rd step of lesson 7.3 in vimtutor ? (QUIZ)

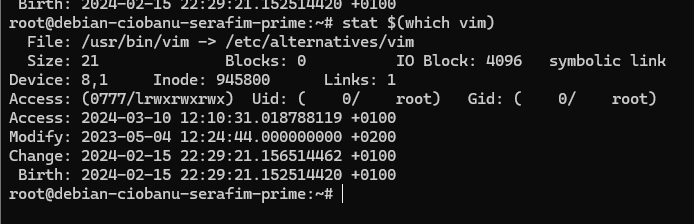


Type the start of a command: :e

09. Exit vimtutor (you’ll know how by now). Use which to find where vim (the binary, executable file) was installed and show the file attributes and type using ls -l or stat.



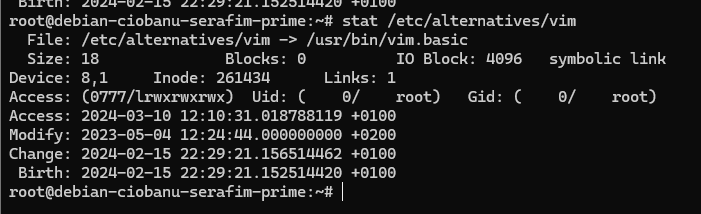
10. Now try to do this in one single command (using which and ls -l) using the $( ) command substitution..



11. You learned that vim is actually a symlink. Where does it point to?

/etc/alternatives/vim

12. Now investigate the file it points to in a similar way.

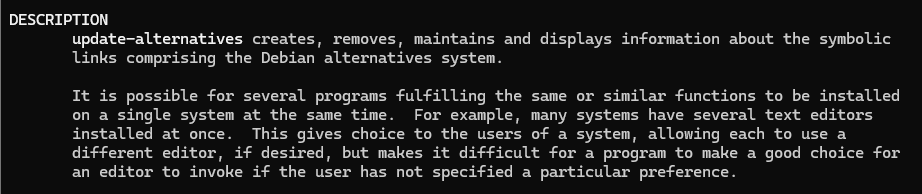


13. Try ls -l in /etc/alternatives. Read man update-alternatives and related manual pages.

This teaches you how the alternatives system works

to provide alternate implementations for various system programs, in different packages, and let you choose which to use,

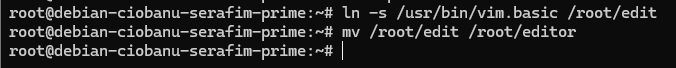
It works using a chain of symbolic links, from canonical file to entry in alternatives, to the name of the actual implementation.



14. Create a symlink to vim.basic in your **/root** homedir, called edit and rename it to editor using the mv command (2 steps).

ln -s /usr/bin/vim.basic /root/edit

mv /root/edit /root/editor



15. Now try and run the “editor” program you just copied. (If you’re not careful, nano may start instead!)

To run a command you simply type its name if it is in the path. Is “editor” in your path? (echo $PATH and your homedirectory is echo ~, try it).

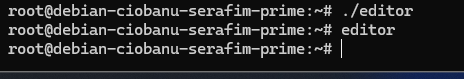
If not you must specify the full path.

Try ~/editor. You should know how to exit vim again by now if you followed vimtutor. Do so and afterwards use cd to go to your homedir and type ./editor. Realize what ~ means. Exit vim again.

Try ls -ld .

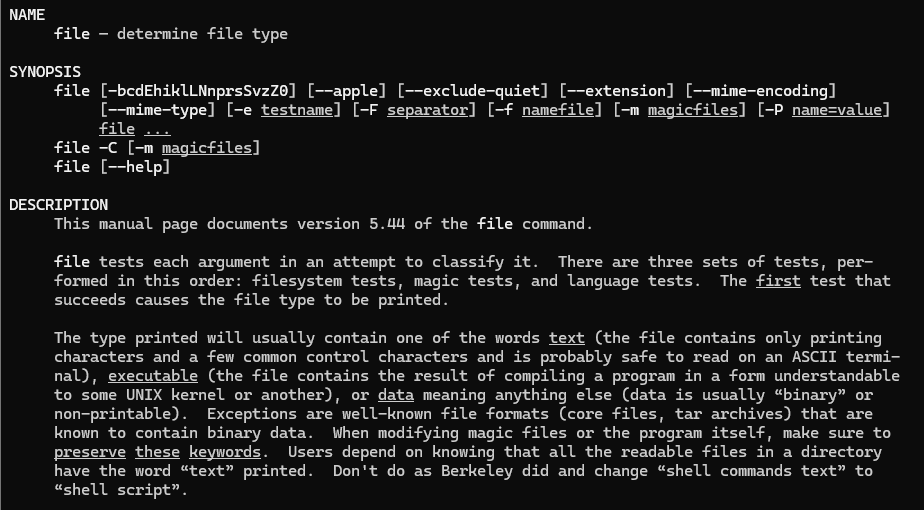
You’ll see that . is actually a hard link to the current folder.

This explains why ./editor is a correct way to invoke this editor symlink (to vim.basic) when you first go to this folder.



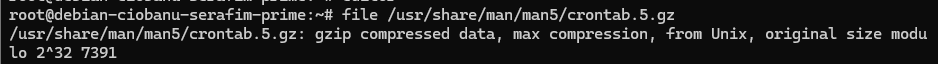
If you do not export it, then it will lauch the nano. If you try to run the command as “./editor”, then you get the vim.

16. Learn what the *file* command does



Try it on the following files:

• /usr/share/man/man5/crontab.5.gz

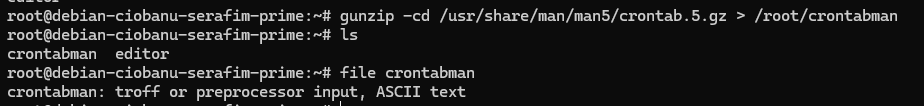


• (unzip it using -c and/or -d option for gzip and redirect to a file crontabman in your homedir)

gunzip -cd /usr/share/man/man5/crontab.5.gz > /root/crontabman

• examine this

file crontabman (in your homedir)



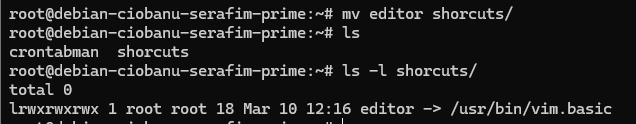
You can also run file on images, movie files, network traffic logs, config files, ... basically any type of file.

It works wonders in scripts that must do an action depending on the type of file a user supplied.(see script lab later on in this series)

17. Create a folder in your homedir: shortcuts using mkdir and move editor (the symlink) to this folder using the mv command.

mkdir storcuts

mkdir storcuts



18. Study your own filesystem using these tools and/or files:

• df

• mount

• /proc/mounts

• /etc/mtab

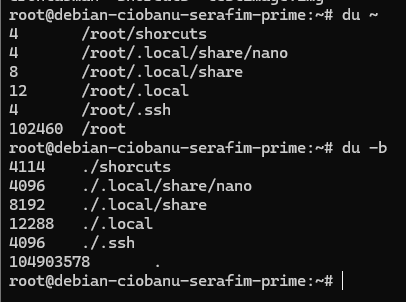
• /etc/fstab

19. Use dd to read 100 megabytes (for example in blocks of 10; see previous part of the lab) from /dev/zero to a file testimage.img in your home folder.

dd if=/dev/zero of=~/testimage.img bs=10M count=10

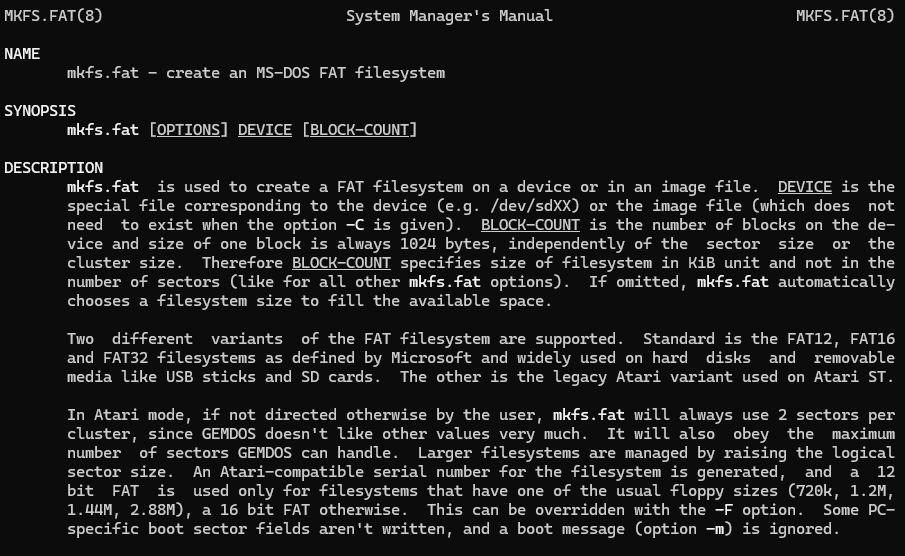
20. Use du ~ (and read its man page) to see the usage on disk of all files and folders under your home folder.

Now compare with the output of du -b. Is there a difference? (QUIZ)



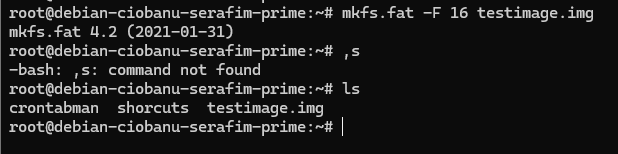
The main difference between du ~ and du -b ~ is that du -b provides the disk usage in bytes for each file and directory, while du ~ provides it in kilobytes by default.

21. Install dosfstools using apt install dosfstools and study mkfs.fat using its man page.



22. Use mkfs.fat to create a FAT16 file system inside the testimage.img file you created before

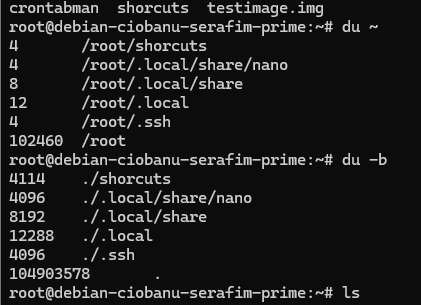
Consider what you are doing here ! You are not writing a filesystem data to a block device (eg: a hard disk partition which is an entry in /dev like we have seen earlier), but to a regular file. This is what UNIX filesystems are all about.



mkfs.fat -F 16 testimage.img

23. Again use du ~ and du -b on your home folder and study the difference with what you saw before

There is really no difference, from what I can see



Maybe there is, because of some kind of modification to the file itself.

24. Use dd to read the first 200 sectors (a sector is 512 bytes) from the file you created, and write it to a new file called simply firstsectors.

dd if=testimage.img of=firstsectors bs=512 count=200

25. use xxd to study firstsectors until the first readable content. What is the first readable text you find ? (QUIZ)

xxd is a hex viewer that simply turns its stdin to a hex dump on stdout. It is included in the vim-common package, that you indirectly installed previously.

Now use xxd by redirecting the input from firstsectors and piping the output to less. That way you can browse through a hex dump of the bytes you just wrote in the file (which now really contains what a partition might contain on your disk had you formatted it with FAT16).

Skim through the output until you find some readable content.

xxd firstsectors | less

This

00000060: 6973 206e 6f74 2061 2062 6f6f 7461 626c is not a bootabl

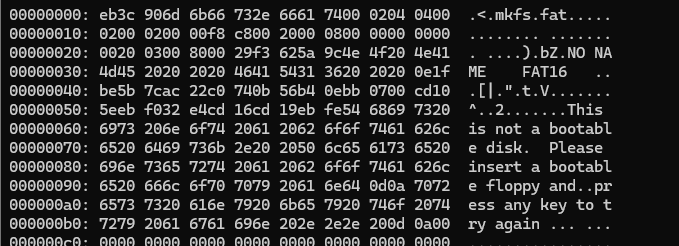
00000070: 6520 6469 736b 2e20 2050 6c65 6173 6520 e disk. Please

00000080: 696e 7365 7274 2061 2062 6f6f 7461 626c insert a bootabl

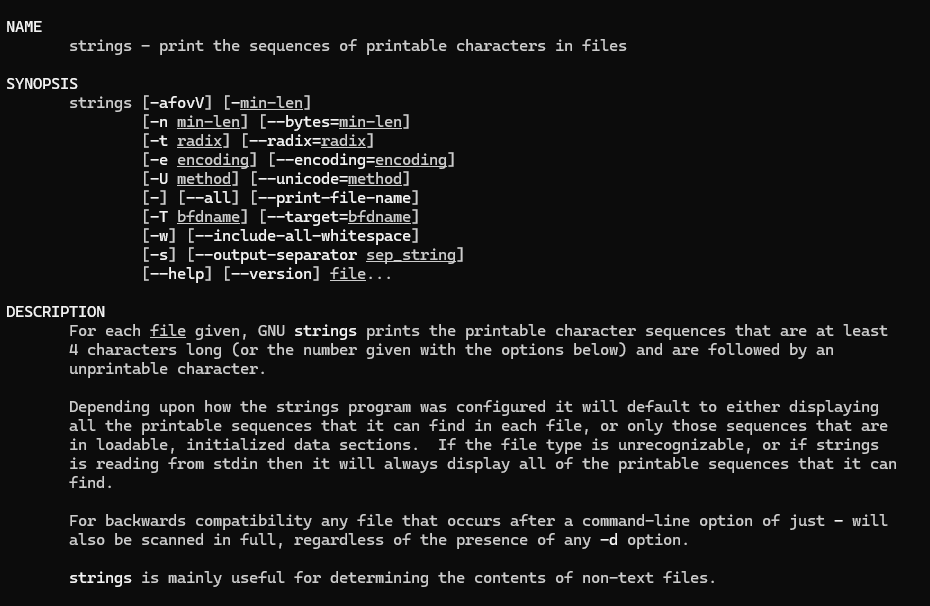
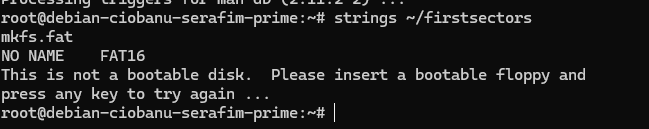
00000090: 6520 666c 6f70 7079 2061 6e64 0d0a 7072 e floppy and..pr

000000a0: 6573 7320 616e 7920 6b65 7920 746f 2074 ess any key to t

000000b0: 7279 2061 6761 696e 202e 2e2e 200d 0a00 ry again

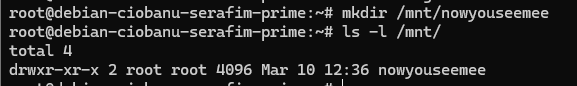


26. Use apt install binutils to install the package binutils and try strings ~/firstsectors | less and guess what strings does. Read its man page for more info.



27. Prepare to mount a new filesystem, under a mount point /mnt which already exists on your system at install time.

First list the contents of /mnt using ls -l /mnt. You will see it doesn’t contain anything at this moment. Create a folder called nowyouseemee in the /mnt folder using mkdir. Check you can see that folder using ls –l /mnt.



*Under /mnt, you will find temporary mount points, intended for a system administrator to mount file systems temporarily under, during setup or maintenance work.*

*During production use, different dedicated mount points will be used, you learned for instance that your root filesystem is always mounted under /.*

28. Now read the man page for mount and find out how to loopback-mount a filesystem.

Try to mount the testimage.img file that you created which contains a FAT16 filesystem. Mount it as a loopback device. Mount it under /mnt.

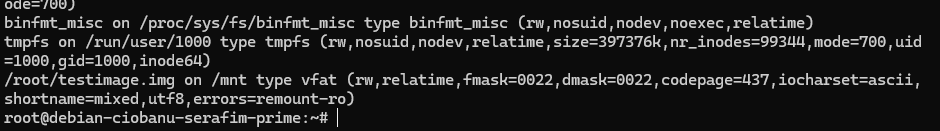
sudo mount -o loop testimage.img /mnt

Try again to list the directory contents of /mnt You will see that the folder nowyouseeme is not visible anymore. It was cascaded by the filesystem that lives in testimage.img and was mounted under /mnt.

**Important**

Don't worry! The nowyouseemee folder is still there, just not visible until we unmount the filesystem again.

29. Revisit the mount list using the mount command. Note which device is new in the mount list and at what mount point it is mounted. As what type can you see it is mounted? (QUIZ)

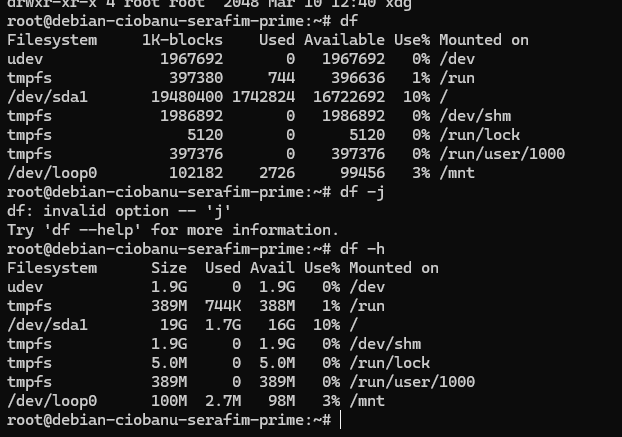


vfat

30. Copy the contents of /etc, including its subfolders (called a recursive copy), to the mounted filesystem. Read the man page for cp to know how to do this.

sudo cp -r /etc/\* /mnt

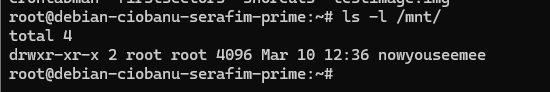
31. Confirm using cd, ls, ls -R and find that the files were copied to /mnt. Also check used and available disk space again using df and df -h (human-readable output). Take note of the diskspace used compared to when you checked the disc space before.



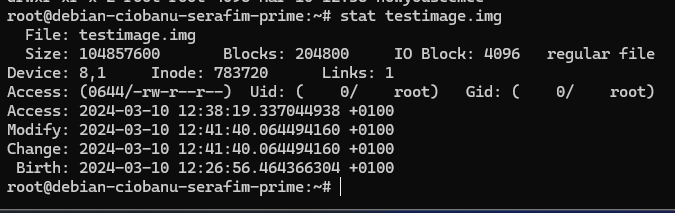
32. Use umount to unmount the filesystem you mounted om /mnt again. Note that you only need to specify either the mountpoint or the device, Unmount using the mount point.

umount /mnt

33. Confirm that you can again see the original contents of the /mnt folder. Use find or ls.



34. Check the modification timestamp of testimage.img using ls -l or stat. Check the current date/time using date. You should see that the file was only just modified. That's because we just wrote a lot of files to it, and it was just unmounted.

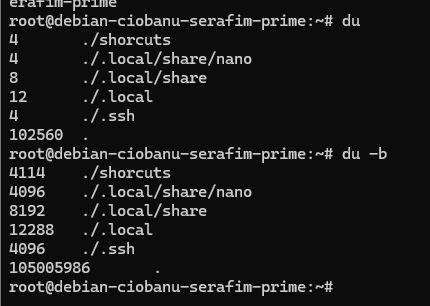


Indeed

35. Use xxd to browse through testimage.img, and try to observe the changes in the raw bytes of your FAT16 filesystem. Try to find the names of some files from /etc and their contents.

36. Use strings from before and try to find data from /etc in your image file by piping the output of strings to grep and look for your vm’s $HOSTNAME. (SCREENSHOT).

37. Like before, check the output of du and du -b on your home folder and compare. Read up on **sparse files** to understand why the reported size on disk is smaller (but gradually less so) than the total number of bytes it claims.



Because you have slack space