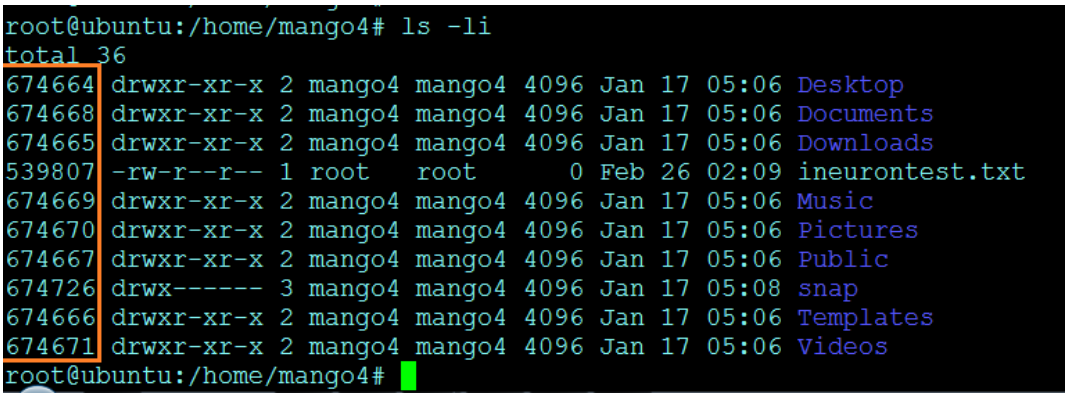
1. What are inode and process id?

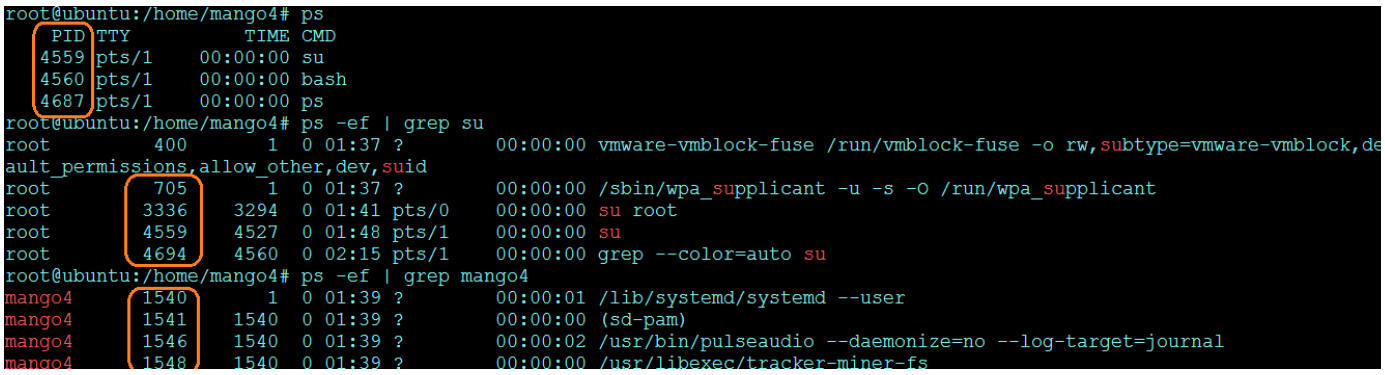
Answer -

- An inode is a information or data structure which will stores various information about a file in Linux
- Each file has an inode containing metadata about the file.
- such as the access mode (read, write, execute permissions), ownership, file type, file size, group, number of links, etc.
- Each file in a filesystem has a unique inode number(integer) and it is assigned whenever the files is created
- we can check that in linux terminal with command for eg. **ls -li** and the left side output in numeric numbers is the inode number representing that file. Below is sample snap



- Process ID (PID) is a unique identification number which is used to identify a particular process in Linux.

- whenever we start any application it will has some PID assigned to it. if we see that PID then it means that process is running
- we can check that for eg. **ps**
- **ps -ef | grep java** - this will show if java app is running with which PID
- Below is the sample screen shot for process id running



2. Which are the Linux Directory Commands?

Answer -

- To Create a new Directory **mkdir directory_name**
- Remove the directory and its contents recursively **rm -r directory_name**
- Forcefully remove directory recursively **rm -rf directory_name**
- Copy source_directory recursively to destination. If destination exists, copy source_directory into destination, otherwise create destination with the contents of source_directory.
  cp -r source_directory_name destination

- to change directory we use **cd full_path_name**
  eg. cd /var/kafka, cd /opt/confluent/kafka
- to come one directory back from current **cd ..**
- to come twice directory back from current we use **cd ../..**

3. What is Virtual Desktop?

Answer -

- Virtual desktops are preconfigured images of operating systems and applications in which the desktop environment is separated from the physical device used to access it.
- Users can access their virtual desktops remotely over a network.
- We can use any client system to connect to virtual desktop
- The virtual desktop provider installs client software on the endpoint device, and the user then interacts with that software on the device..

4. Which are the different modes of vi editor?

Answer -

- In linux VI editor in general is visual editor(it is a command line text editor)
- there are 3 modes

  1. Command Mode (eg. vi file.txt and enter this mode is command mode)

  2. Inset Mode (eg to write something in file.txt press small i then insert mode will be activated, you can verify at bottom)

  3. Last line Mode (press Escape then it will take you to

last line and from there you can type colon **(:wq!** save and exit or **:q!** to exit without save)

5. What are daemons?

Answer -

- A daemon is a service process that runs in the background and supervises the system or provides functionality to other processes.
- It waits for a condition or event to be triggered.
- Daemons are processes that are often started when the system is bootstrapped and terminate only when the system is shut down.
- A daemon process is an application that is designed to run in the background, typically managing some kind of ongoing service.
- A daemon process might listen for an incoming request for access to a service.
- For example httpd daemon listens for requests to view web pages.
- In short Daemon is a background process.

6. What are the process states in Linux?

Answer -

- There are five Linux process states.
- running & runnable, interruptable_sleep, uninterruptable_sleep, stopped, and zombie.
- To check we can use command below commands:
- **ps -l** to see processes associated with the current shell
- **ps -el** to see all processes on the system

7. Explain grep command.

Answer -

- grep command is used to find a pattern in a file
- ef. there is a file with name file123.txt and inside that file there are list of fruits available and we want to find if fruit with name apple is there then we can do that with command : **grep apple file123.txt** -> this will show/print if there is a word with name apple in files123.txt . If that pattern matches multiple times then also it will show the word that many times.
- In above example if we want to know the line number we can add **grep -n apple file123.txt** - > this will show apple pattern with line numbers. it will show if these characters are in other word as well. for eg. if pineapple is in the file123.txt and if we do **grep -n apple** then apple is in pineapple so it will show that pineapple also.
- if we want only the perfect keyword match then in that case we should use **grep -wn apple file123.txt** and it will show only if apple is there as a single word in that file123.txt
- if we want to search for capital case sensitive then we have to use **grep -i apple file123.txt**
- **grep -v mango file123.txt** -> In this it will show all results except mango.
- **grep -A2 apple file123.txt** -> this will first find apple and will show 2 lines after that apple
- **grep -B2 apple file123.txt ->** this will first find apple and will show 2 lines before that apple
- **grep -A2 -B2 apple file123.txt ->** this will first find apple and will show 2 lines before and after that apple

- **grep -nr mango .** -> if we do not know which file we need to find this mango then we can use this command and give fullstop . at end and it will search all files in that directly and will show the file name and the word apple in it.

8. Explain Process Management System Calls in Linux

Answer -

- Process management system calls in Linux are as below
- fork – For creating a new or duplicate process from the parent process.
- wait – Processes are supposed to wait for other processes to complete their work.
- exec – To run a new program or Loads the selected program into the memory.
- exit – Terminates the process.

9. Explain the 'ls' command

Answer -

- The ls command is used to lists the files and directories/subfolders within the file system
- If we use with some parameters then we can get a detailed information for those files and directories/subfolders(eg. permissions, owner, group, last

edit date/time, size etc..) eg. ls, ls -lrt, ls -l, ls -a and so on etc.

- Syntax is $ ls --help -> this will show all options and parameters which we can use based upon our requirements
- Usage: ls [OPTION]... [FILE]...

10. Explain the redirection operator

Answer -

- Linux has some command or special character to redirect these input and output functionalities
- Redirection means the ouput of one command we want to give as input of other other command then we can use these
- there are 3 types of redirection
- standard input(stdin), standard output(stdout), standard error(stderr)
- for eg. we type hostname in terminal and if we get ouput in terminal, but if we want to redirect that output rather than in file instead of terminal then we can do that with redirection operator.

- eg 1. **cat > filename.txt**
- then hit enter and what ever we type and press ctrol+D
- now all above line will be added as content to filename.txt

- eg 2. suppose we want to add to filename.txt at bottom /append then we can do with
- **cat >> filename.txt**
- hit enter and type what ever you want and press control+D
- now this will be appended to filename.txt

- eg 3 . suppose we want to overwrite all content in filename.txt then use below
- **cat > filename.txt**
- hit enter and type what ever you want and cotrol+D will save as a new content and will overwrite all previous content.

- eg 4. we type hostname and what that to be redirected to filename.txt then use
- hostname > filename.txt -> this will overwrite all contents and save only the hostname output in filename.txt
- hostname >> filename.txt -> this will add at bottom/append to end of filename.
- similarly we can do multiple things with this >, >> operator. Below is some example which i tried and want to show:
- Please find snap in next page as an example.

```
total 0
localhost:/home/ineuronexcersise# touch filename.txt
localhost:/home/ineuronexcersise# cat filename.txt
localhost:/home/ineuronexcersise# cat > filename.txt
line1 test
line2 test
line3 test
localhost:/home/ineuronexcersise# cat filename.txt
line1 test
line2 test
line3 test
localhost:/home/ineuronexcersise# cat >> filename.txt
line 4 test
line 5 test
localhost:/home/ineuronexcersise# cat filename.txt
line1 test
line2 test
line3 test
line 4 test
line 5 test
localhost:/home/ineuronexcersise# cat > filename.txt
all above will be overwritten, New line 1
New line2
localhost:/home/ineuronexcersise# cat filename.txt
all above will be overwritten, New line 1
New line2
localhost:/home/ineuronexcersise# hostname >> filename.txt
localhost:/home/ineuronexcersise# cat filename.txt
all above will be overwritten, New line 1
New line2
localhost
localhost:/home/ineuronexcersise#
```

- similarly we can do multiple things with this >, >> operator. Below is some example which i tried and want to show:

- Command '2>' redirects the error of an output
- This command is useful to handle the errors. below is the example.
- Also if we want to append then we have to use 2>> as show in below example.

```
localhost:~# ls -l
total 16
-rw-r--r--    1 root     root           114 Jul  5  2020 bench.py
-rw-r--r--    1 root     root            76 Jul  3  2020 hello.c
-rw-r--r--    1 root     root            22 Jun 26  2020 hello.js
-rw-r--r--    1 root     root           151 Jul  5  2020 readme.txt
localhost:~# ls
bench.py    hello.c     hello.js    readme.txt
localhost:~# cat file.txt
cat: can't open 'file.txt': No such file or directory
localhost:~# cat file.txt 2> error.txt
localhost:~# cat error.txt
cat: can't open 'file.txt': No such file or directory
localhost:~# cat file.txt 2>> error.txt
localhost:~# cat error.txt
cat: can't open 'file.txt': No such file or directory
cat: can't open 'file.txt': No such file or directory
localhost:~#
```

- Suppose we want to work with two commands and then redirect the output we can do as below in example
- we have listed only .txt files and that all filenames ending with .txt want to save in a new file called alltext.txt the we can do with below command
- **ls -l *.txt | cat > alltext.txt**
- same way if the file alltext.txt already exist and we want to append at end of file then use >>
- eg. **ls -l *.txt| cat >> alltext.txt**
- Please find snap in next page as an example.

```
localhost:~# ls -l
total 24
-rw-r--r--    1 root     root           337 Feb 26 18:40 all.txt
-rw-r--r--    1 root     root           114 Jul  5  2020 bench.py
-rw-r--r--    1 root     root           108 Feb 26 18:35 error.txt
-rw-r--r--    1 root     root            76 Jul  3  2020 hello.c
-rw-r--r--    1 root     root            22 Jun 26  2020 hello.js
-rw-r--r--    1 root     root             0 Feb 26 18:39 hello.txt
-rw-r--r--    1 root     root           151 Jul  5  2020 readme.txt
-rw-r--r--    1 root     root             0 Feb 26 18:39 test.txt
localhost:~# ls -l *.txt
-rw-r--r--    1 root     root           337 Feb 26 18:40 all.txt
-rw-r--r--    1 root     root           108 Feb 26 18:35 error.txt
-rw-r--r--    1 root     root             0 Feb 26 18:39 hello.txt
-rw-r--r--    1 root     root           151 Jul  5  2020 readme.txt
-rw-r--r--    1 root     root             0 Feb 26 18:39 test.txt
localhost:~# ls -l *.txt | cat > alltext.txt
localhost:~# cat alltext.txt
-rw-r--r--    1 root     root           337 Feb 26 18:40 all.txt
-rw-r--r--    1 root     root             0 Feb 26 18:40 alltext.txt
-rw-r--r--    1 root     root           108 Feb 26 18:35 error.txt
-rw-r--r--    1 root     root             0 Feb 26 18:39 hello.txt
-rw-r--r--    1 root     root           151 Jul  5  2020 readme.txt
-rw-r--r--    1 root     root             0 Feb 26 18:39 test.txt
localhost:~#
```