Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

Lab Report 1

[Code No: COMP 307]

Submitted by:

Nischal Subedi (53)

Group: CS60 (III/I)

Submitted to:

Ms. Rabina Shrestha

Department of Computer Science and Engineering

Submission Date: 06/12/2025

Q1. What is Linux?

Linux is an open-source operating system built on Unix principles. Developed by Linus Torvalds in 1991, it has grown into one of the most widely used OS platforms worldwide. It’s valued for its reliability, security, and adaptability. Linux runs on devices ranging from Android phones to supercomputers, servers, and personal computers. Unlike closed-source systems, Linux lets anyone access, change, and share its source code without restrictions.

Q2. Explain the Linux Hierarchical File System.

The Linux file system is organized in a hierarchical tree structure that begins at the root directory, represented by /. Every file and folder extends from this main root. Some important directories are:

- **/home** – Stores personal folders for regular users
- **/root** – The home directory for the system’s administrator
- **/bin** – Contains essential command-line programs
- **/etc** – Holds system configuration files
- **/var** – Includes files that change frequently, such as logs
- **/tmp** – Used for temporary files
- **/usr** – Contains user applications and related data

Q3. Explain the importance of Linux commands in Operating Systems..

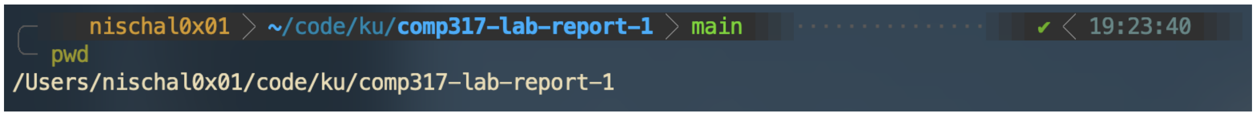
Linux commands are essential tools for working with the operating system. They allow users to:

- Move around and organize the file system
- Handle administrative tasks and adjust system settings
- Automate routine actions using scripts
- Check system performance and solve problems
- Manage users, permissions, and security settings

Some widely used linux commands, their description and use cases

1. pwd (Print Working Directory)

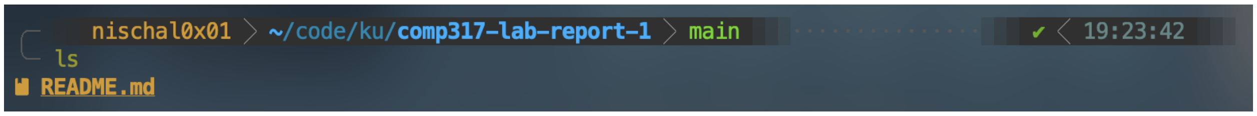
Explanation: The `pwd` command displays the absolute path of your current directory location in the file system. When you open the terminal, you start in your home directory, and this command helps you identify where you are in the directory hierarchy.



```
nischal0x01 > ~/code/ku/comp317-lab-report-1 > main          ✔ < 19:23:40
pwd
/Users/nischal0x01/code/ku/comp317-lab-report-1
```

2. ls (List Directory Contents)

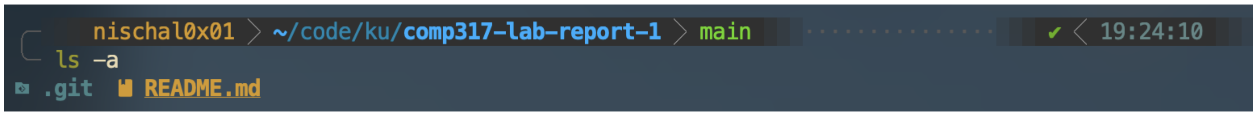
Explanation: The `ls` command lists all files and directories in the current directory. It provides a quick overview of contents without needing a file manager, displaying items alphabetically by default.



```
nischal0x01 > ~/code/ku/comp317-lab-report-1 > main          ✔ < 19:23:42
ls
README.md
```

3. ls -a (List All Files Including Hidden)

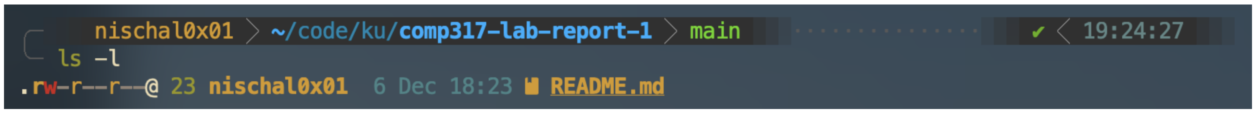
Explanation: The `ls -a` command shows all files including hidden files that start with a dot (.). In Linux, configuration files are often hidden, and this flag reveals them along with `.` (current directory) and `..` (parent directory).



```
nischal0x01 > ~/code/ku/comp317-lab-report-1 > main          ✔ < 19:24:10
ls -a
.git  README.md
```

4. ls -l (Long Listing Format)

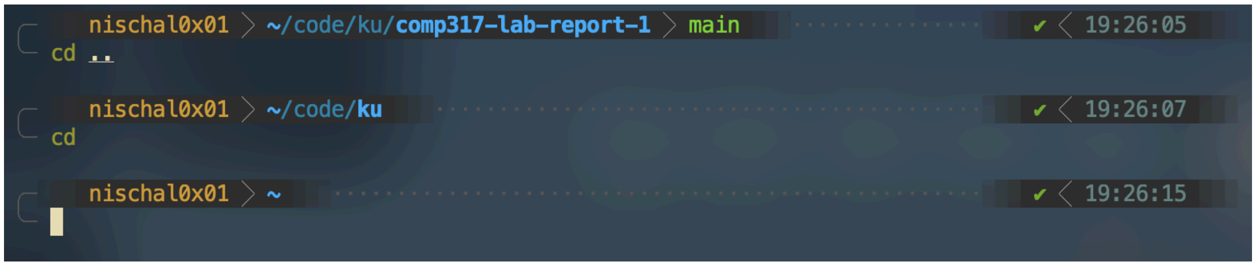
Explanation: The `ls -l` command displays detailed information about files including permissions, owner, group, size, and modification date. The first character indicates file type (- for file, d for directory).



```
  nischal0x01 > ~/code/ku/comp317-lab-report-1 > main ················ ✔ < 19:24:27
  ls -l
.rw-r--r--@ 23 nischal0x01  6 Dec 18:23 README.md
```

5. cd (Change Directory)

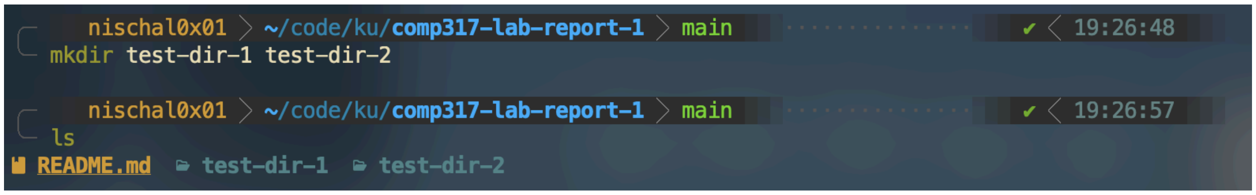
Explanation: The `cd` command navigates between directories in the file system. Use `cd ..` to move to parent directory, `cd ~` or just `cd` to go home, and `cd -` to return to previous directory.



```
  nischal0x01 > ~/code/ku/comp317-lab-report-1 > main ················ ✔ < 19:26:05
  cd ..

  nischal0x01 > ~/code/ku ············································ ✔ < 19:26:07
  cd

  nischal0x01 > ~ ···················································· ✔ < 19:26:15
  ▌
```

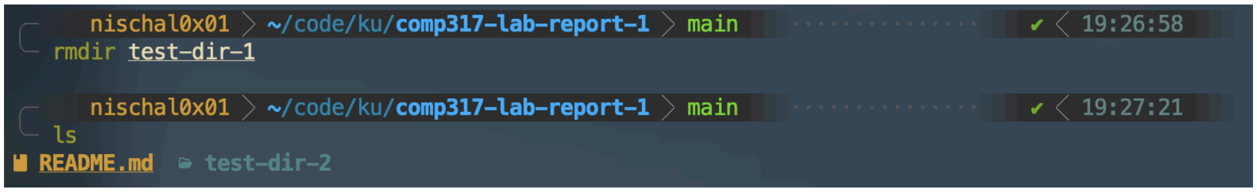
6. mkdir (Make Directory)

Explanation: The `mkdir` command creates new directories in the current location or specified path. You can create multiple directories at once or use `-p` flag to create nested directory structures like `mkdir -p parent/child/grandchild`.


```
nischal0x01 > ~/code/ku/comp317-lab-report-1 > main ··············· ✔ < 19:26:48
mkdir test-dir-1 test-dir-2

nischal0x01 > ~/code/ku/comp317-lab-report-1 > main ··············· ✔ < 19:26:57
ls
README.md  test-dir-1  test-dir-2
```

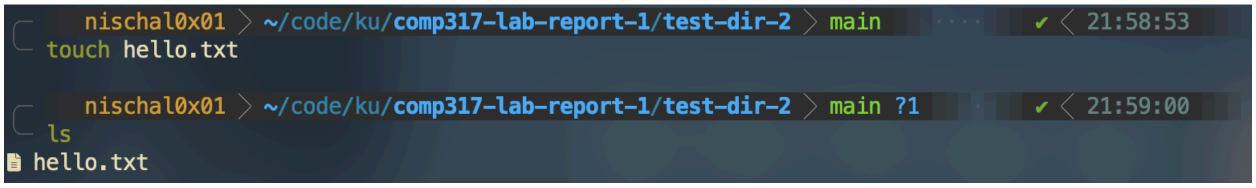
7. rmdir (Remove Empty Directory)

Explanation: The `rmdir` command removes empty directories only. If a directory contains files, it will fail with an error, providing a safety mechanism against accidental deletion of important data.


```
nischal0x01 > ~/code/ku/comp317-lab-report-1 > main ··············· ✔ < 19:26:58
rmdir test-dir-1

nischal0x01 > ~/code/ku/comp317-lab-report-1 > main ··············· ✔ < 19:27:21
ls
README.md  test-dir-2
```

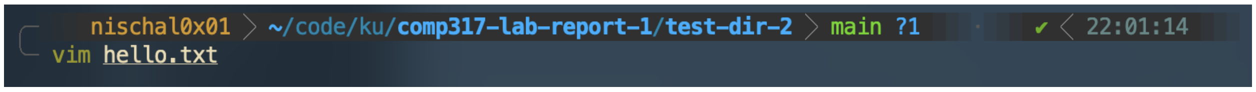
8. touch (Create Empty File)

Explanation: The `touch` command creates new empty files or updates timestamps of existing files. It's commonly used to quickly create placeholder files that will be edited later.

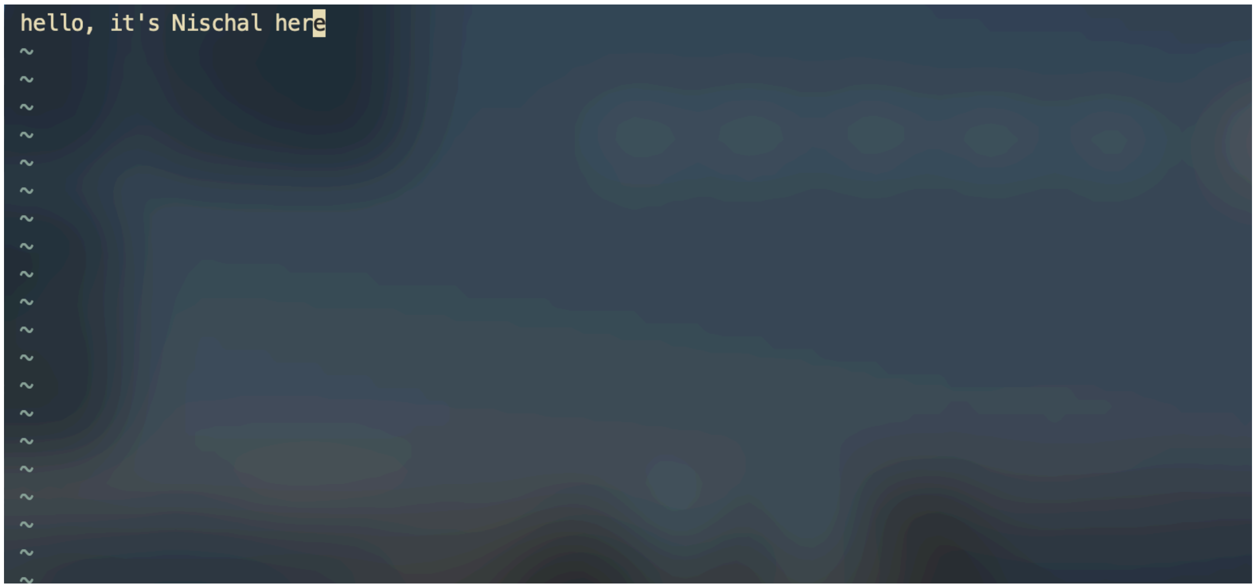

```
nischal0x01 > ~/code/ku/comp317-lab-report-1/test-dir-2 > main ···· ✔ < 21:58:53
touch hello.txt

nischal0x01 > ~/code/ku/comp317-lab-report-1/test-dir-2 > main ?1 · ✔ < 21:59:00
ls
hello.txt
```

9. vim (Text Editor)

Explanation:The `vim` editor is a simple, beginner-friendly command-line text editor.



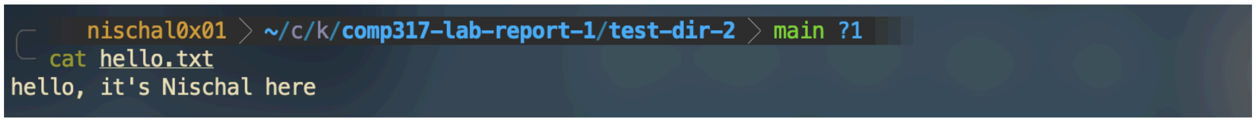
```
╭─   nischal0x01 > ~/code/ku/comp317-lab-report-1/test-dir-2 > main ?1          ✔ < 22:01:14
╰─ vim hello.txt
```



```
hello, it's Nischal here
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

10. cat (Concatenate and Display File Contents)

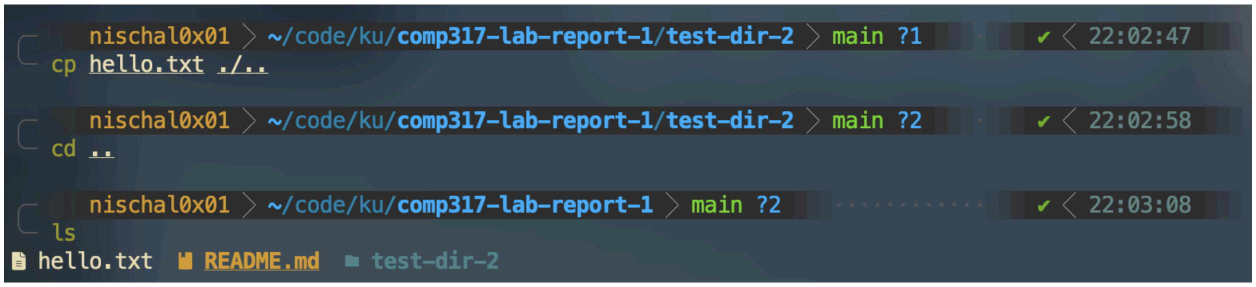
Explanation: The `cat` command displays entire file contents in the terminal. It can also concatenate multiple files and display them together. For large files, use `less` or `more` for better navigation.



```
╭─   nischal0x01 > ~/c/k/comp317-lab-report-1/test-dir-2 > main ?1
╰─ cat hello.txt
hello, it's Nischal here
```

11. cp

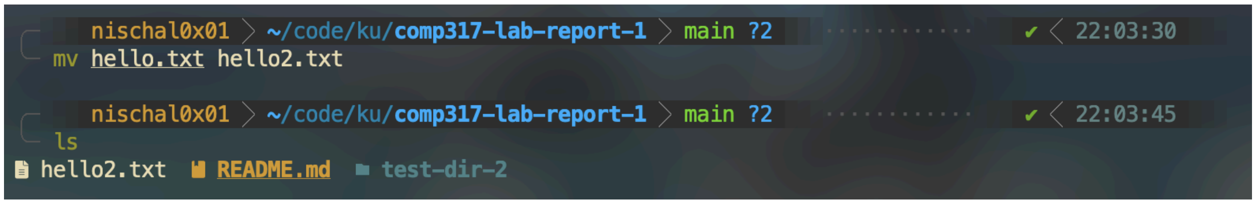
Explanation: The cp command copies files or directories while keeping the original intact. Use -r flag for directories, -i for confirmation prompts, and -v for verbose output showing what's being copied.



```
nischal0x01 > ~/code/ku/comp317-lab-report-1/test-dir-2 > main ?1          ✔ < 22:02:47
cp hello.txt ./..

nischal0x01 > ~/code/ku/comp317-lab-report-1/test-dir-2 > main ?2          ✔ < 22:02:58
cd ..

nischal0x01 > ~/code/ku/comp317-lab-report-1 > main ?2 ·············       ✔ < 22:03:08
ls
hello.txt   README.md   test-dir-2
```

12. mv

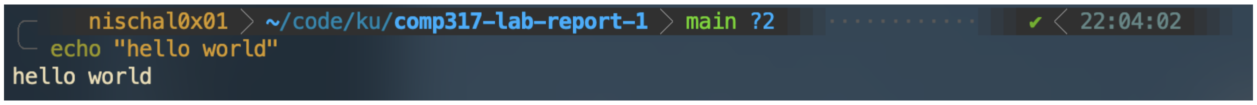
Explanation: The mv command moves files to new locations or renames them. Unlike cp, it removes the original from source location. Use -i flag to prevent accidental overwrites of existing files.



```
nischal0x01 > ~/code/ku/comp317-lab-report-1 > main ?2 ·············       ✔ < 22:03:30
mv hello.txt hello2.txt

nischal0x01 > ~/code/ku/comp317-lab-report-1 > main ?2 ·············       ✔ < 22:03:45
ls
hello2.txt   README.md   test-dir-2
```

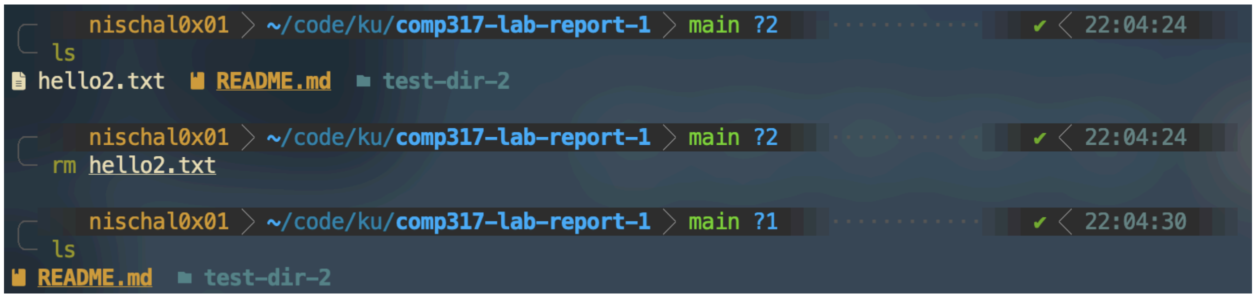
13. echo

Explanation: The echo command prints text or variable values to the terminal. Use > to write to files or >> to append. It's useful in scripts for displaying messages and creating simple text files.


```
nischal0x01 > ~/code/ku/comp317-lab-report-1 > main ?2 ············ ✔ < 22:04:02
echo "hello world"
hello world
```

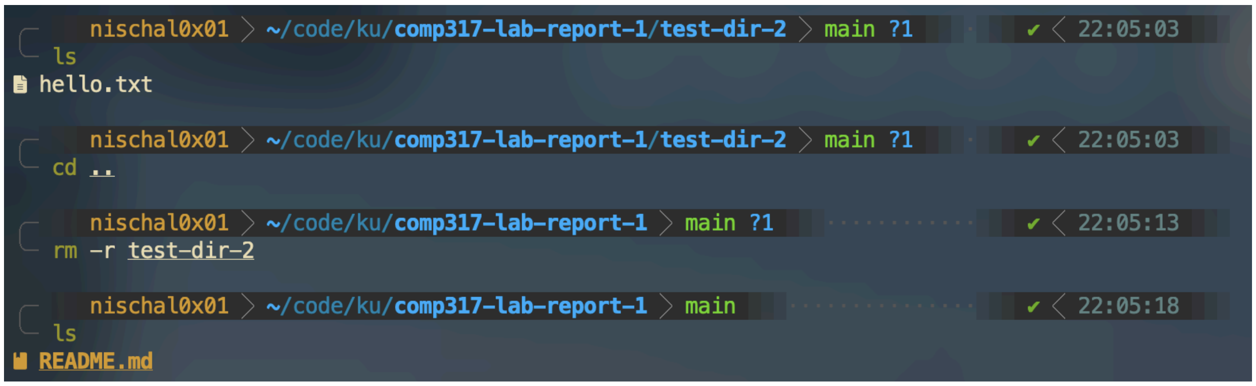
14. rm (Remove Files)

Explanation: The `rm` command permanently deletes files from the file system. Unlike moving to trash, this cannot be easily undone, so use with caution and consider using `-i` flag for confirmation prompts.


```
nischal0x01 > ~/code/ku/comp317-lab-report-1 > main ?2 ············ ✔ < 22:04:24
ls
hello2.txt   README.md   test-dir-2

nischal0x01 > ~/code/ku/comp317-lab-report-1 > main ?2 ············ ✔ < 22:04:24
rm hello2.txt

nischal0x01 > ~/code/ku/comp317-lab-report-1 > main ?1 ············ ✔ < 22:04:30
ls
README.md   test-dir-2
```

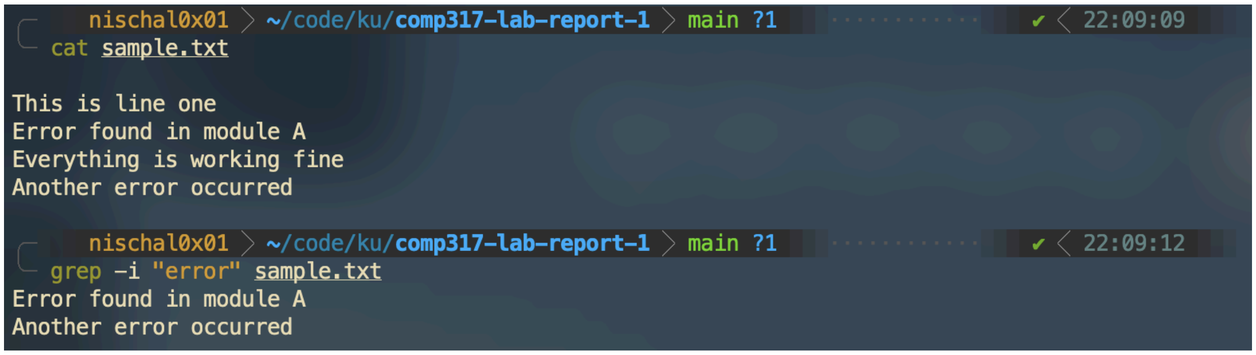
15. rm -r (Remove Directories Recursively)

Explanation: The `rm -r` command deletes directories and all their contents recursively. This is powerful but dangerous; always verify the path before executing. Adding `-f` forces deletion without prompts.


```
nischal0x01 > ~/code/ku/comp317-lab-report-1/test-dir-2 > main ?1 · ✔ < 22:05:03
ls
hello.txt

nischal0x01 > ~/code/ku/comp317-lab-report-1/test-dir-2 > main ?1 · ✔ < 22:05:03
cd ..

nischal0x01 > ~/code/ku/comp317-lab-report-1 > main ?1 ············ ✔ < 22:05:13
rm -r test-dir-2

nischal0x01 > ~/code/ku/comp317-lab-report-1 > main ·············· ✔ < 22:05:18
ls
README.md
```

16. grep

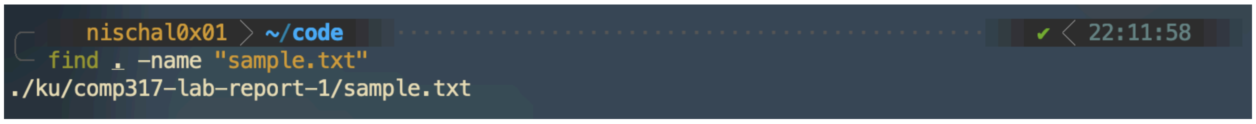
Explanation: The `grep` command searches for text patterns in files and displays matching lines. Use `-i` for case-insensitive search, `-n` to show line numbers, and `-r` for recursive directory searches.



```
nischal0x01 > ~/code/ku/comp317-lab-report-1 > main ?1 ·············  ✔ < 22:09:09
cat sample.txt

This is line one
Error found in module A
Everything is working fine
Another error occurred

nischal0x01 > ~/code/ku/comp317-lab-report-1 > main ?1 ·············  ✔ < 22:09:12
grep -i "error" sample.txt
Error found in module A
Another error occurred
```

17. find

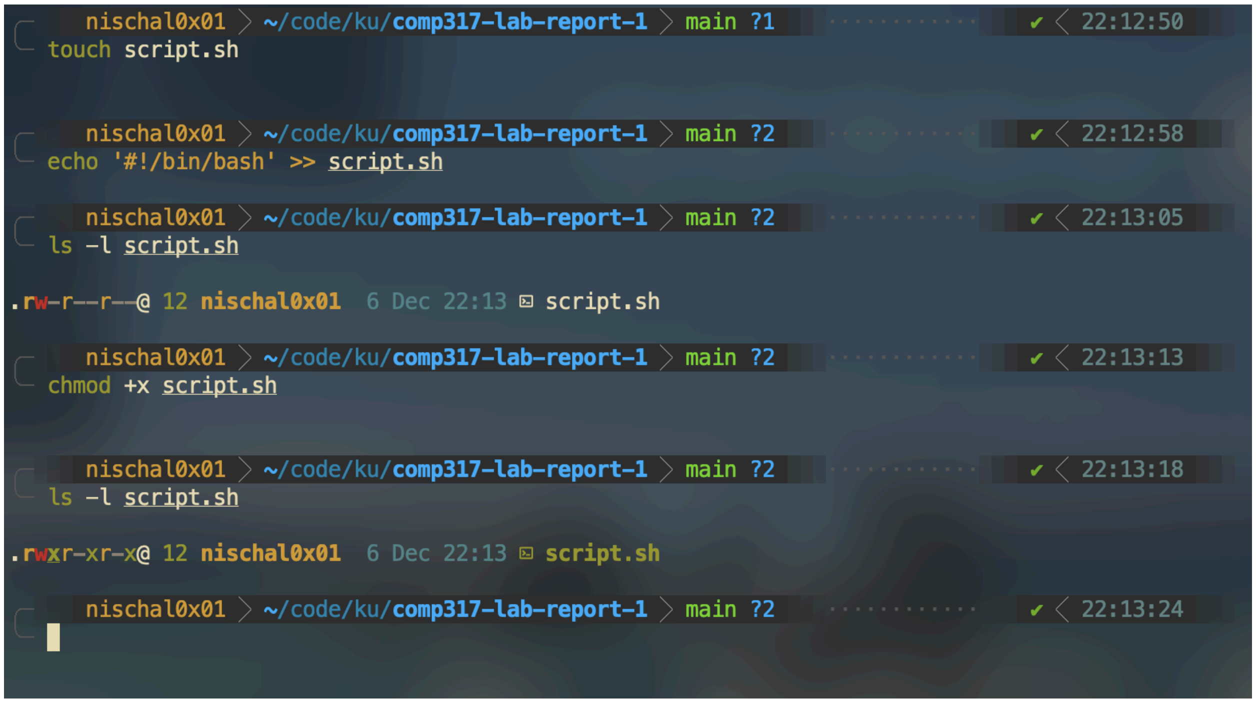
Explanation: The `find` command searches for files based on criteria like name, size, or modification time. It searches in real-time through directories, making it more flexible but slower than database-based commands like `locate`.



```
nischal0x01 > ~/code ·············································  ✔ < 22:11:58
find . -name "sample.txt"
./ku/comp317-lab-report-1/sample.txt
```

18. chmod

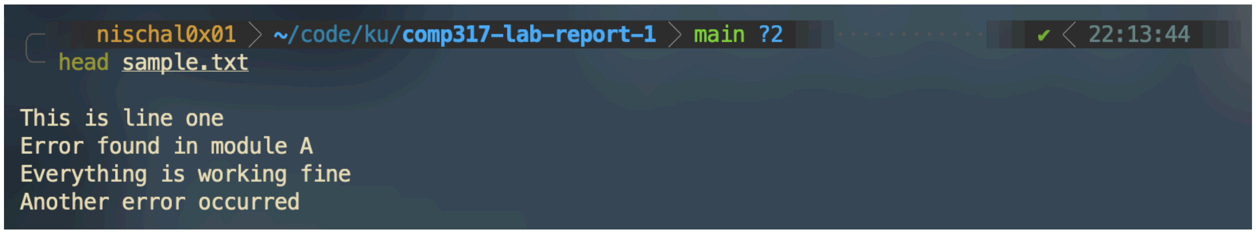
Explanation: The `chmod` command modifies file permissions for owner, group, and others. Use numeric notation (755) or symbolic (u+x) to set read, write, and execute permissions for different user categories.



```
nischal0x01 > ~/code/ku/comp317-lab-report-1 > main ?1 ............ ✔ < 22:12:50
touch script.sh

nischal0x01 > ~/code/ku/comp317-lab-report-1 > main ?2 ............ ✔ < 22:12:58
echo '#!/bin/bash' >> script.sh

nischal0x01 > ~/code/ku/comp317-lab-report-1 > main ?2 ............ ✔ < 22:13:05
ls -l script.sh

.rw-r--r--@ 12 nischal0x01  6 Dec 22:13 script.sh

nischal0x01 > ~/code/ku/comp317-lab-report-1 > main ?2 ............ ✔ < 22:13:13
chmod +x script.sh

nischal0x01 > ~/code/ku/comp317-lab-report-1 > main ?2 ............ ✔ < 22:13:18
ls -l script.sh

.rwxr-xr-x@ 12 nischal0x01  6 Dec 22:13 script.sh

nischal0x01 > ~/code/ku/comp317-lab-report-1 > main ?2 ............ ✔ < 22:13:24
```

19. head (Display Beginning of File)

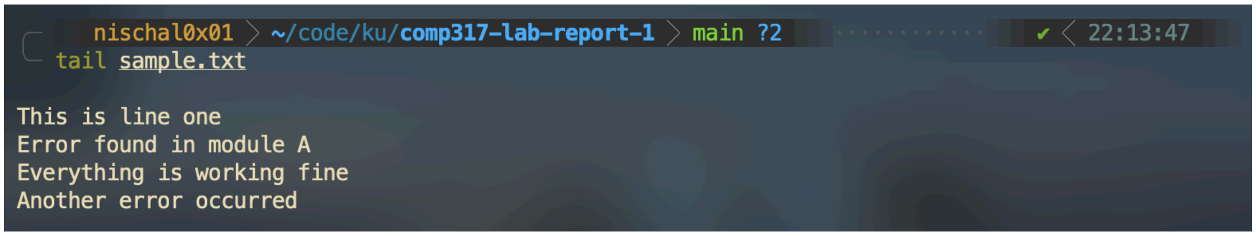
Explanation: The head command shows the first 10 lines of a file by default. Use -n option to specify different number of lines, which is useful for quickly previewing file contents.



```
nischal0x01 > ~/code/ku/comp317-lab-report-1 > main ?2 ............ ✔ < 22:13:44
head sample.txt

This is line one
Error found in module A
Everything is working fine
Another error occurred
```

20. tail (Display End of File)

Explanation: The `tail` command displays the last 10 lines of a file by default. Use `-f` flag to continuously monitor new lines being added, which is particularly useful for watching log files in real-time.

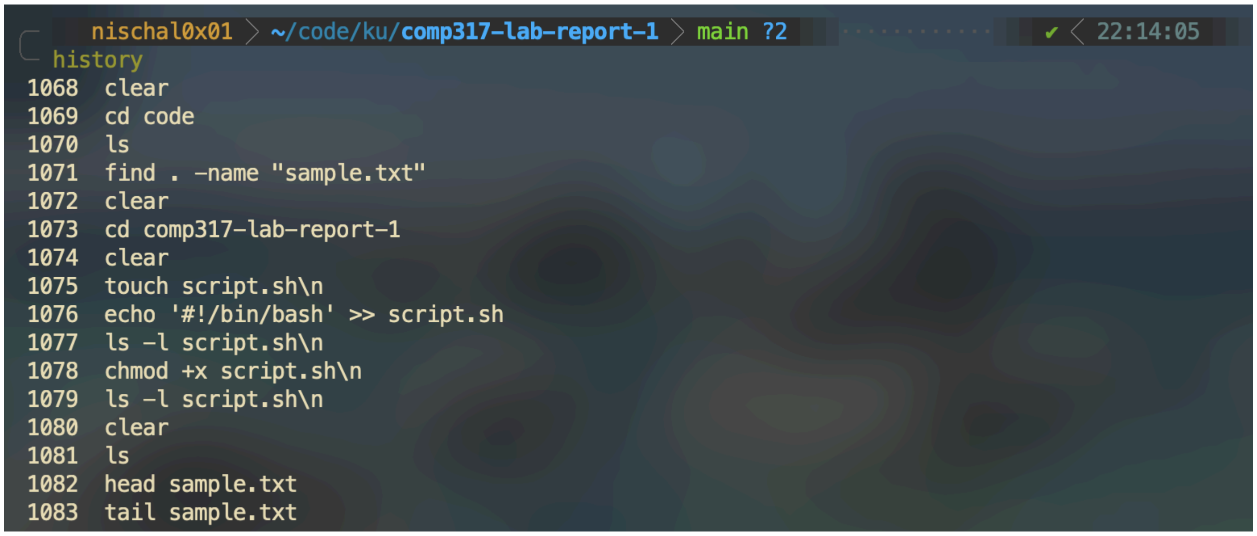


```
nischal0x01 > ~/code/ku/comp317-lab-report-1 > main ?2 ······· ✔ < 22:13:47
tail sample.txt

This is line one
Error found in module A
Everything is working fine
Another error occurred
```

Fewer lines of content in the sample.txt file caused the same thing to display in both the cases

21. history (Command History)

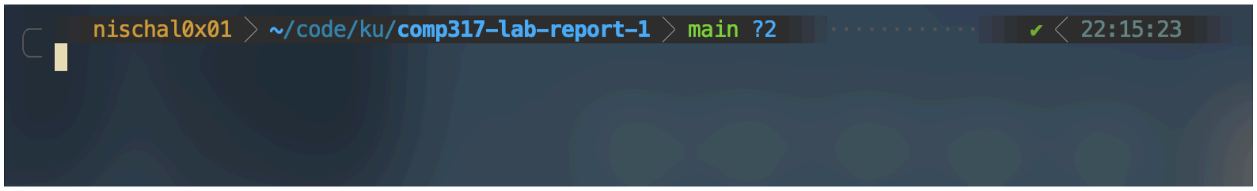
Explanation: The `history` command displays previously executed commands with numbers. Use `!number` to re-execute a specific command, which is helpful for recalling complex commands without retyping them.



```
nischal0x01 > ~/code/ku/comp317-lab-report-1 > main ?2 ······· ✔ < 22:14:05
history
1068  clear
1069  cd code
1070  ls
1071  find . -name "sample.txt"
1072  clear
1073  cd comp317-lab-report-1
1074  clear
1075  touch script.sh\n
1076  echo '#!/bin/bash' >> script.sh
1077  ls -l script.sh\n
1078  chmod +x script.sh\n
1079  ls -l script.sh\n
1080  clear
1081  ls
1082  head sample.txt
1083  tail sample.txt
```

22. clear (Clear Terminal Screen)

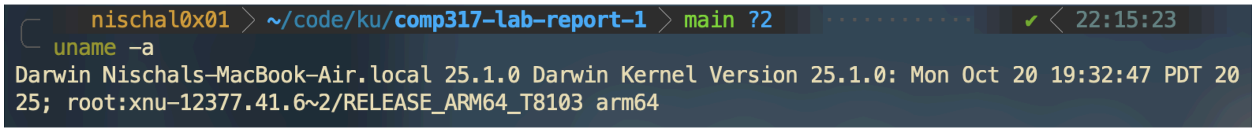
Explanation: The `clear` command removes all text from the terminal screen, providing a clean workspace. You can also use keyboard shortcut `Ctrl+L` for the same result without typing the command.



```
nischal0x01 > ~/code/ku/comp317-lab-report-1 > main ?2 ............ ✔ < 22:15:23
```

23. uname -a

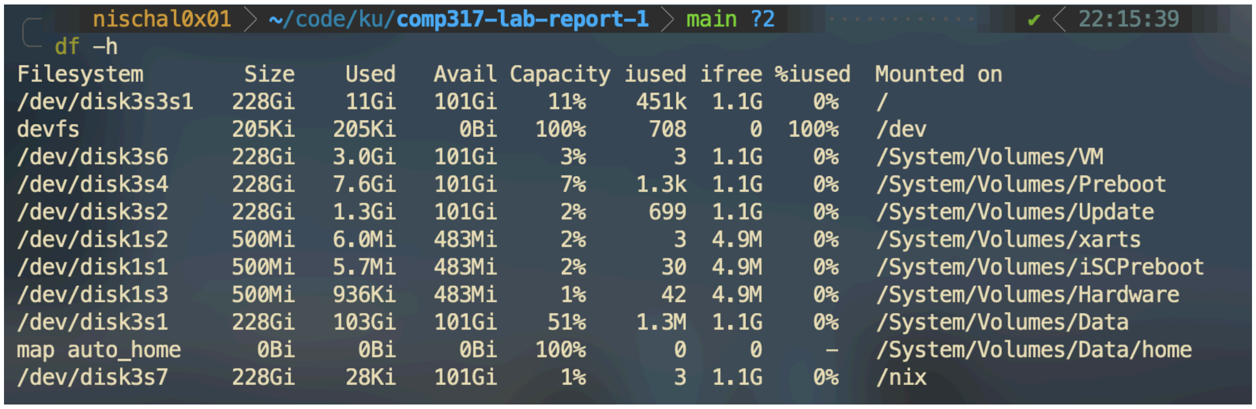
Explanation: The `uname -a` command displays detailed system information including kernel name, version, machine hardware name, processor type, and operating system. This is useful for system diagnostics and checking system specifications.



```
nischal0x01 > ~/code/ku/comp317-lab-report-1 > main ?2 ............ ✔ < 22:15:23
uname -a
Darwin Nischals-MacBook-Air.local 25.1.0 Darwin Kernel Version 25.1.0: Mon Oct 20 19:32:47 PDT 20
25; root:xnu-12377.41.6~2/RELEASE_ARM64_T8103 arm64
```

24. df -h (Disk Space Usage)

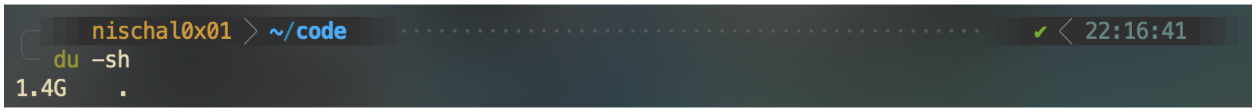
Explanation: The `df -h` command shows disk space usage for all mounted file systems in human-readable format. It displays total size, used space, available space, and usage percentage for each partition.



```
   nischal0x01 > ~/code/ku/comp317-lab-report-1 > main ?2                ✔ < 22:15:39
  df -h
Filesystem         Size    Used   Avail Capacity iused ifree %iused  Mounted on
/dev/disk3s3s1     228Gi    11Gi  101Gi    11%    451k  1.1G    0%   /
devfs              205Ki   205Ki    0Bi   100%     708     0  100%   /dev
/dev/disk3s6       228Gi   3.0Gi  101Gi     3%       3  1.1G    0%   /System/Volumes/VM
/dev/disk3s4       228Gi   7.6Gi  101Gi     7%    1.3k  1.1G    0%   /System/Volumes/Preboot
/dev/disk3s2       228Gi   1.3Gi  101Gi     2%     699  1.1G    0%   /System/Volumes/Update
/dev/disk1s2       500Mi   6.0Mi  483Mi     2%       3  4.9M    0%   /System/Volumes/xarts
/dev/disk1s1       500Mi   5.7Mi  483Mi     2%      30  4.9M    0%   /System/Volumes/iSCPreboot
/dev/disk1s3       500Mi   936Ki  483Mi     1%      42  4.9M    0%   /System/Volumes/Hardware
/dev/disk3s1       228Gi   103Gi  101Gi    51%    1.3M  1.1G    0%   /System/Volumes/Data
map auto_home        0Bi     0Bi    0Bi   100%       0     0     -   /System/Volumes/Data/home
/dev/disk3s7       228Gi    28Ki  101Gi     1%       3  1.1G    0%   /nix
```

25. du -sh (Directory Size)

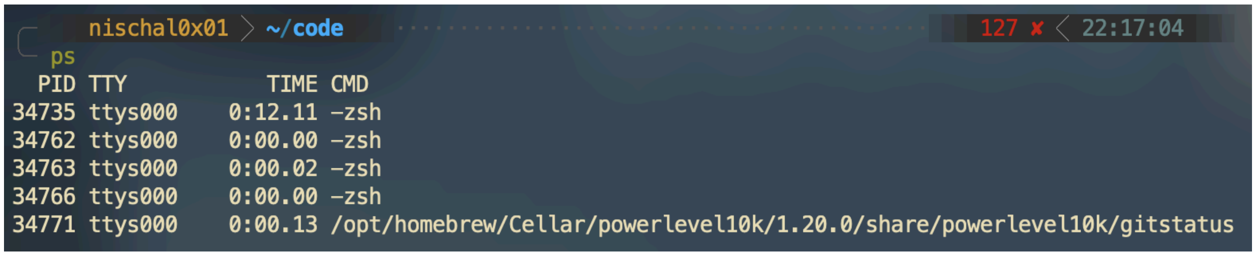
Explanation: The `du -sh` command displays the total size of a directory in human-readable format. The `-s` flag provides summary instead of listing each subdirectory separately, making output cleaner.



```
   nischal0x01 > ~/code                                                  ✔ < 22:16:41
  du -sh
1.4G    .
```

26. ps (Process Status)

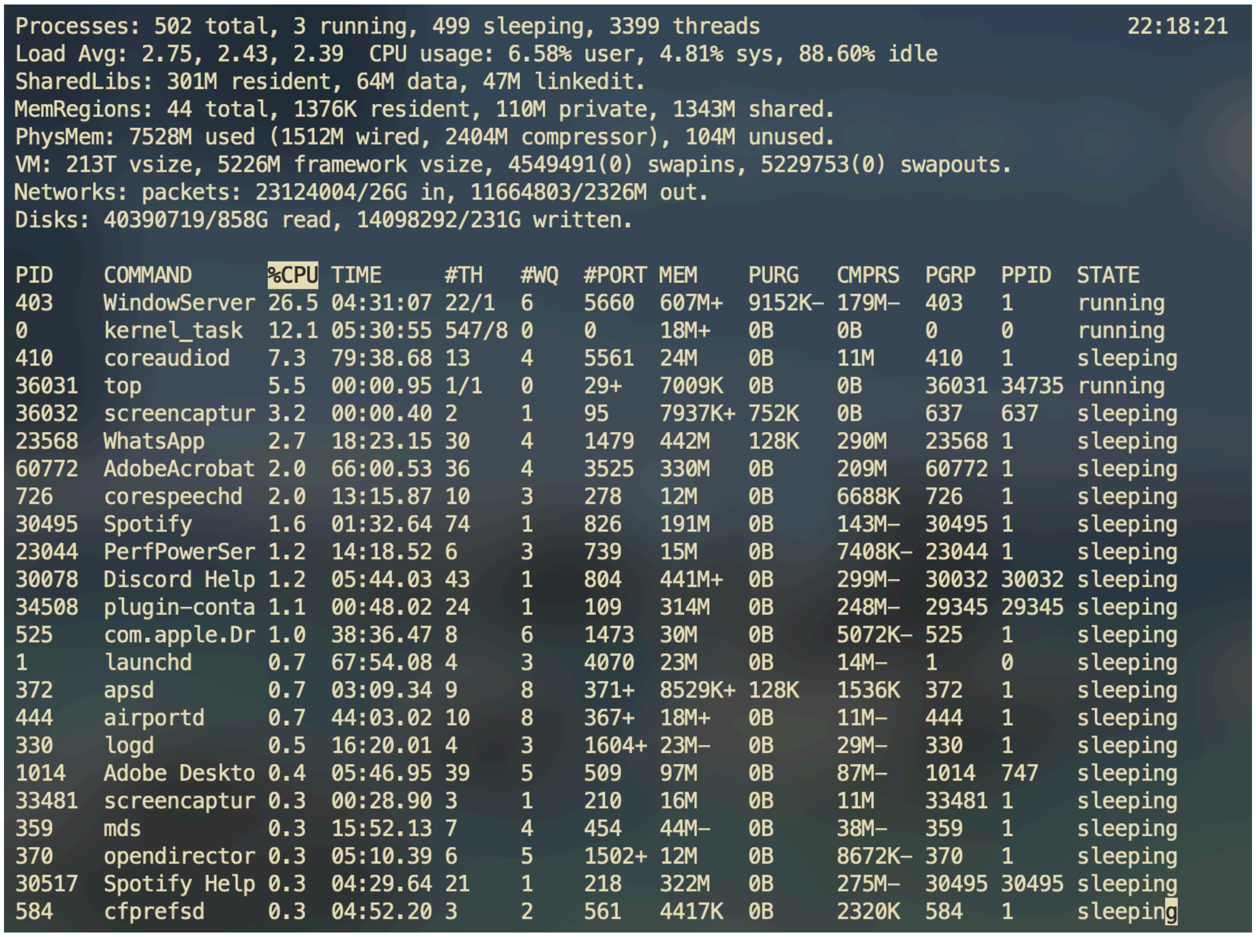
Explanation: The `ps` command displays information about currently running processes including process ID (PID), terminal, CPU time, and command name. Use without options to see basic user processes.



```
   nischal0x01 > ~/code                                            127 ✘ < 22:17:04
  ps
  PID TTY           TIME CMD
34735 ttys000    0:12.11 -zsh
34762 ttys000    0:00.00 -zsh
34763 ttys000    0:00.02 -zsh
34766 ttys000    0:00.00 -zsh
34771 ttys000    0:00.13 /opt/homebrew/Cellar/powerlevel10k/1.20.0/share/powerlevel10k/gitstatus
```

27. top (Real-time Process Monitor)

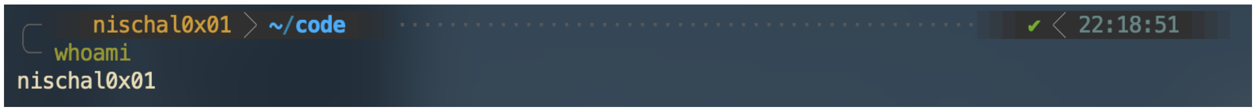
Explanation: The `top` command provides a dynamic real-time view of system processes, CPU usage, and memory consumption. It updates continuously and allows sorting by various metrics. Press `q` to quit the display.



```
Processes: 502 total, 3 running, 499 sleeping, 3399 threads                          22:18:21
Load Avg: 2.75, 2.43, 2.39  CPU usage: 6.58% user, 4.81% sys, 88.60% idle
SharedLibs: 301M resident, 64M data, 47M linkedit.
MemRegions: 44 total, 1376K resident, 110M private, 1343M shared.
PhysMem: 7528M used (1512M wired, 2404M compressor), 104M unused.
VM: 213T vsize, 5226M framework vsize, 4549491(0) swapins, 5229753(0) swapouts.
Networks: packets: 23124004/26G in, 11664803/2326M out.
Disks: 40390719/858G read, 14098292/231G written.

PID    COMMAND      %CPU TIME     #TH    #WQ  #PORT MEM    PURG   CMPRS  PGRP  PPID  STATE
403    WindowServer 26.5 04:31:07 22/1   6    5660  607M+  9152K- 179M-  403   1     running
0      kernel_task  12.1 05:30:55 547/8  0    0     18M+   0B     0B     0     0     running
410    coreaudiod   7.3  79:38.68 13     4    5561  24M    0B     11M    410   1     sleeping
36031  top          5.5  00:00.95 1/1    0    29+   7009K  0B     0B     36031 34735 running
36032  screencaptur 3.2  00:00.40 2      1    95    7937K+ 752K   0B     637   637   sleeping
23568  WhatsApp     2.7  18:23.15 30     4    1479  442M   128K   290M   23568 1     sleeping
60772  AdobeAcrobat 2.0  66:00.53 36     4    3525  330M   0B     209M   60772 1     sleeping
726    corespeechd  2.0  13:15.87 10     3    278   12M    0B     6688K  726   1     sleeping
30495  Spotify      1.6  01:32.64 74     1    826   191M   0B     143M-  30495 1     sleeping
23044  PerfPowerSer 1.2  14:18.52 6      3    739   15M    0B     7408K- 23044 1     sleeping
30078  Discord Help 1.2  05:44.03 43     1    804   441M+  0B     299M-  30032 30032 sleeping
34508  plugin-conta 1.1  00:48.02 24     1    109   314M   0B     248M-  29345 29345 sleeping
525    com.apple.Dr 1.0  38:36.47 8      6    1473  30M    0B     5072K- 525   1     sleeping
1      launchd      0.7  67:54.08 4      3    4070  23M    0B     14M-   1     0     sleeping
372    apsd         0.7  03:09.34 9      8    371+  8529K+ 128K   1536K  372   1     sleeping
444    airportd     0.7  44:03.02 10     8    367+  18M+   0B     11M-   444   1     sleeping
330    logd         0.5  16:20.01 4      3    1604+ 23M-   0B     29M-   330   1     sleeping
1014   Adobe Deskto 0.4  05:46.95 39     5    509   97M    0B     87M-   1014  747   sleeping
33481  screencaptur 0.3  00:28.90 3      1    210   16M    0B     11M    33481 1     sleeping
359    mds          0.3  15:52.13 7      4    454   44M-   0B     38M-   359   1     sleeping
370    opendirector 0.3  05:10.39 6      5    1502+ 12M    0B     8672K- 370   1     sleeping
30517  Spotify Help 0.3  04:29.64 21     1    218   322M   0B     275M-  30495 30495 sleeping
584    cfprefsd     0.3  04:52.20 3      2    561   4417K  0B     2320K  584   1     sleeping
```

28. whoami (Current User)

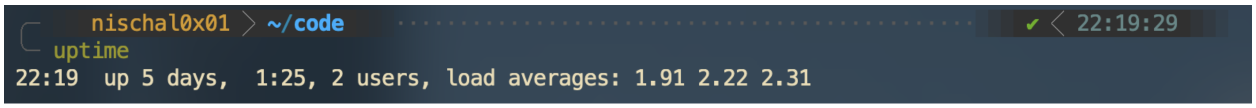
Explanation: The `whoami` command displays the username of the currently logged-in user. It's useful in scripts to verify user identity or when switching between multiple user accounts.



```
nischal0x01 > ~/code                                           ✔ < 22:18:51
whoami
nischal0x01
```

29. uptime (System Uptime)

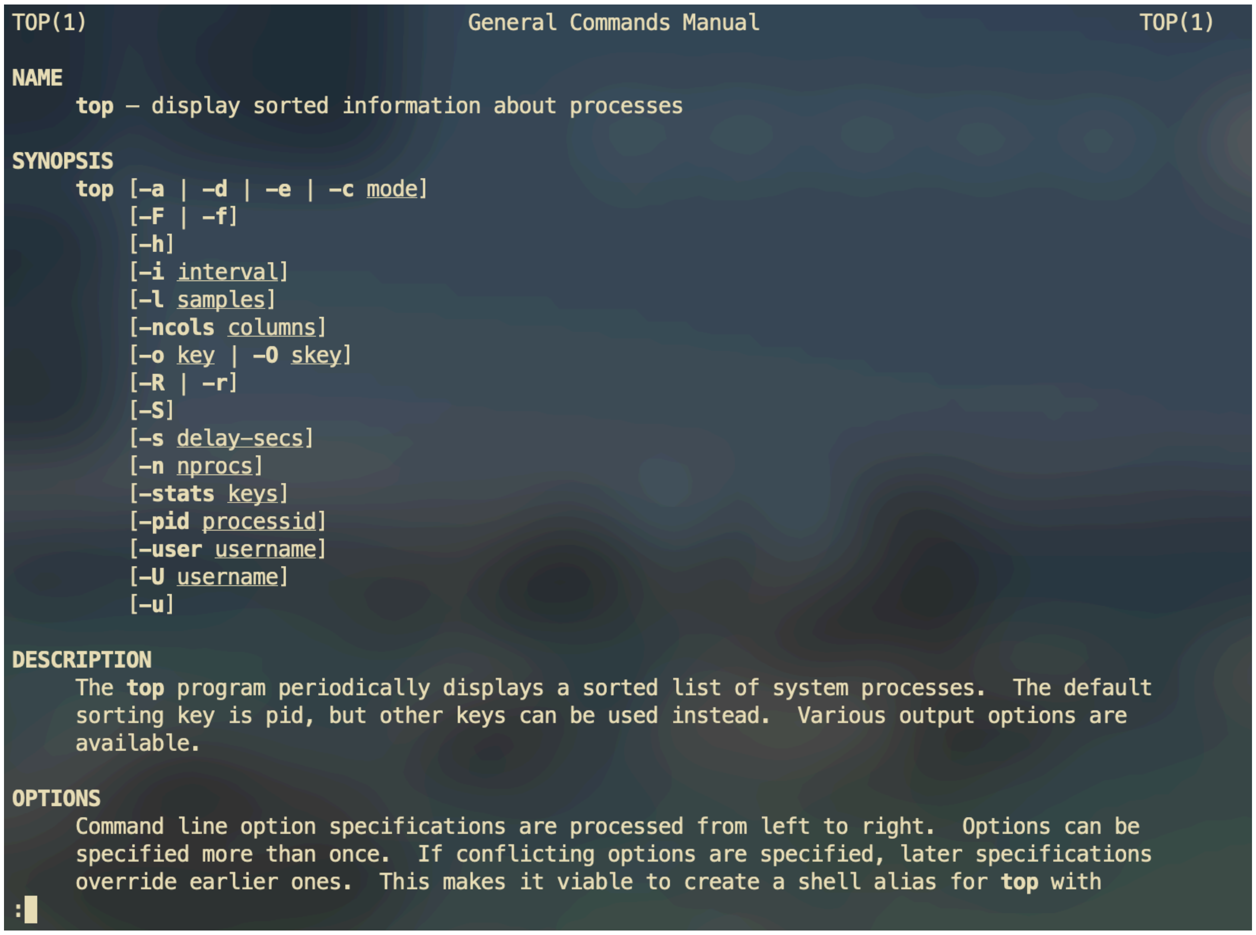
Explanation: The `uptime` command shows how long the system has been running, number of logged-in users, and system load averages for 1, 5, and 15 minutes. This helps assess system stability and performance.



```
nischal0x01 > ~/code                                           ✔ < 22:19:29
uptime
22:19  up 5 days,  1:25, 2 users, load averages: 1.91 2.22 2.31
```

30. man (Manual Pages)

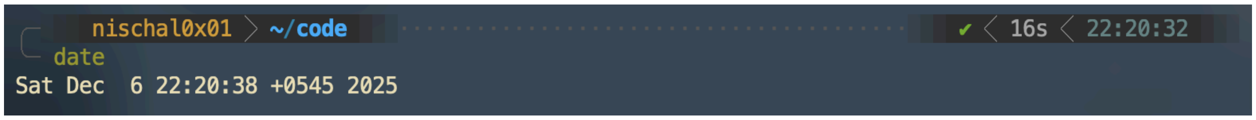
Explanation: The `man` command displays detailed manual pages for other commands, providing usage information, options, and examples. Use arrow keys to scroll and press `q` to quit the manual viewer.



```
TOP(1)                    General Commands Manual                    TOP(1)

NAME
     top – display sorted information about processes

SYNOPSIS
     top [-a | -d | -e | -c mode]
         [-F | -f]
         [-h]
         [-i interval]
         [-l samples]
         [-ncols columns]
         [-o key | -O skey]
         [-R | -r]
         [-S]
         [-s delay-secs]
         [-n nprocs]
         [-stats keys]
         [-pid processid]
         [-user username]
         [-U username]
         [-u]

DESCRIPTION
     The top program periodically displays a sorted list of system processes.  The default
     sorting key is pid, but other keys can be used instead.  Various output options are
     available.

OPTIONS
     Command line option specifications are processed from left to right.  Options can be
     specified more than once.  If conflicting options are specified, later specifications
     override earlier ones.  This makes it viable to create a shell alias for top with
:
```

31. date (Display Date and Time)

Explanation: The `date` command shows the current system date and time in various formats. With appropriate permissions, it can also be used to set the system date and time.



```
╭─   nischal0x01 > ~/code ·································· ✔ < 16s < 22:20:32
╰─ date
Sat Dec  6 22:20:38 +0545 2025
```