Operating Systems INF333

TP01

Linux Fundamentals

Eda Bahar 14/02/2025 💗

ebahar@gsu.edu.tr

edaabahar@gmail.com

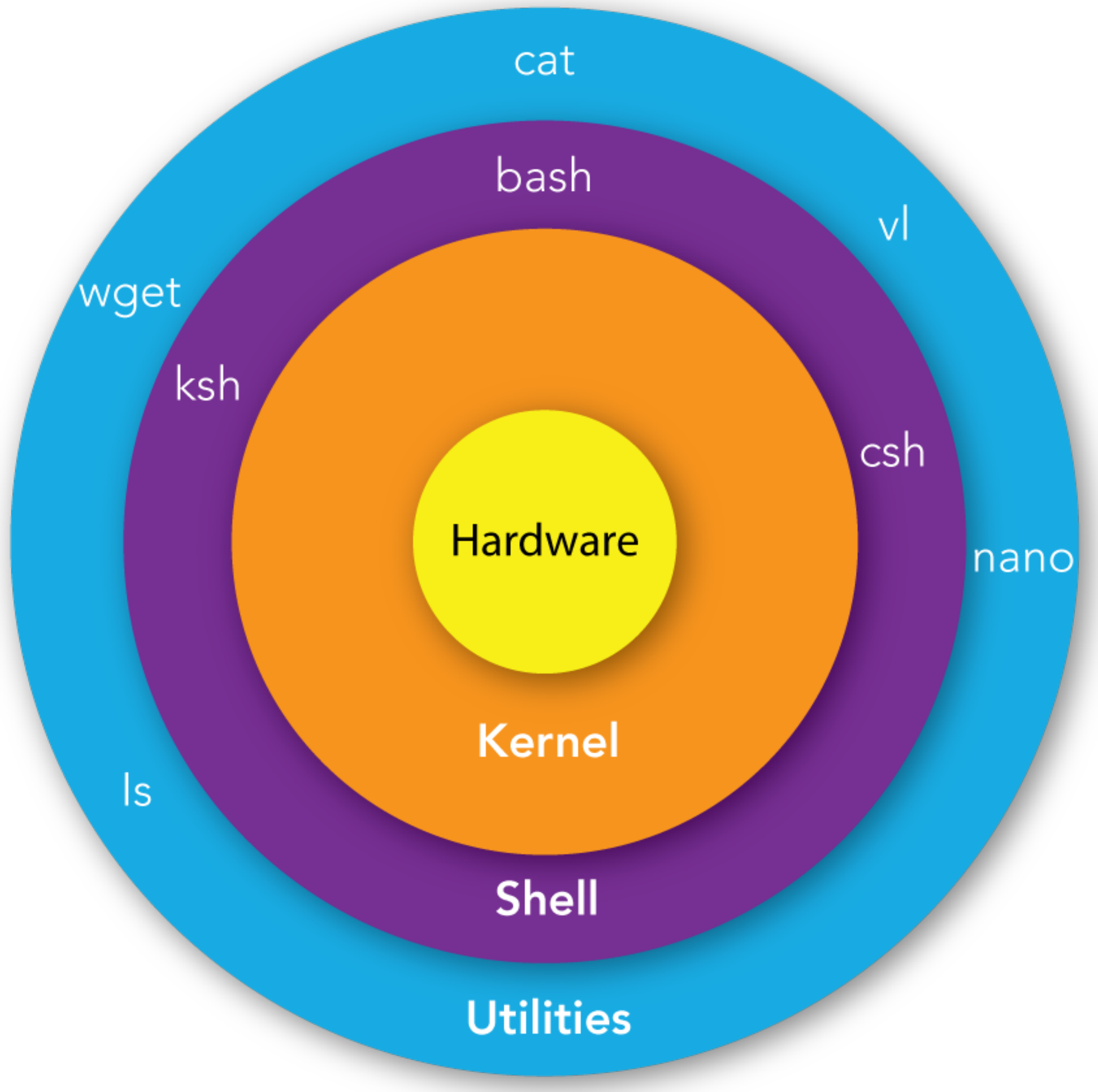
About this TP

- In this TP, you will be learning:
  - bash(the standard Linux shell),
  - Standard Linux commands like **ls**, **cp**, **mv**, etc.
  - Common advanced commands like **grep**, **ps**, etc.
  - The real-time system monitoring command: **top.**

Introducing bash

Kernel

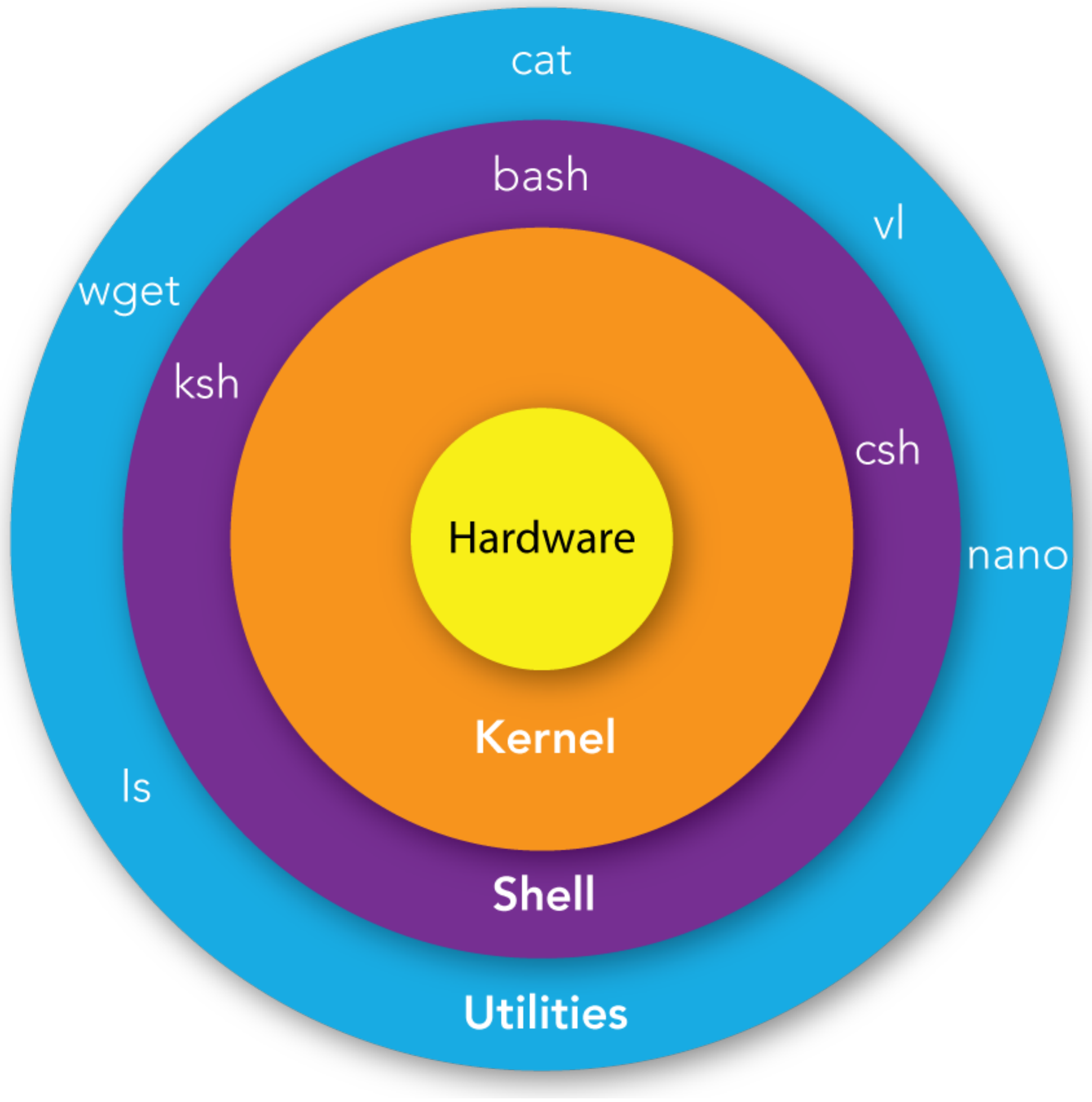
- The kernel is a computer program that forms the core of a computer's operating system, having complete control over all aspects of the system. It manages:
  - File management
  - Process management
  - I/O management
  - Memory Management
  - Device Management
  - …



Introducing bash

Shell

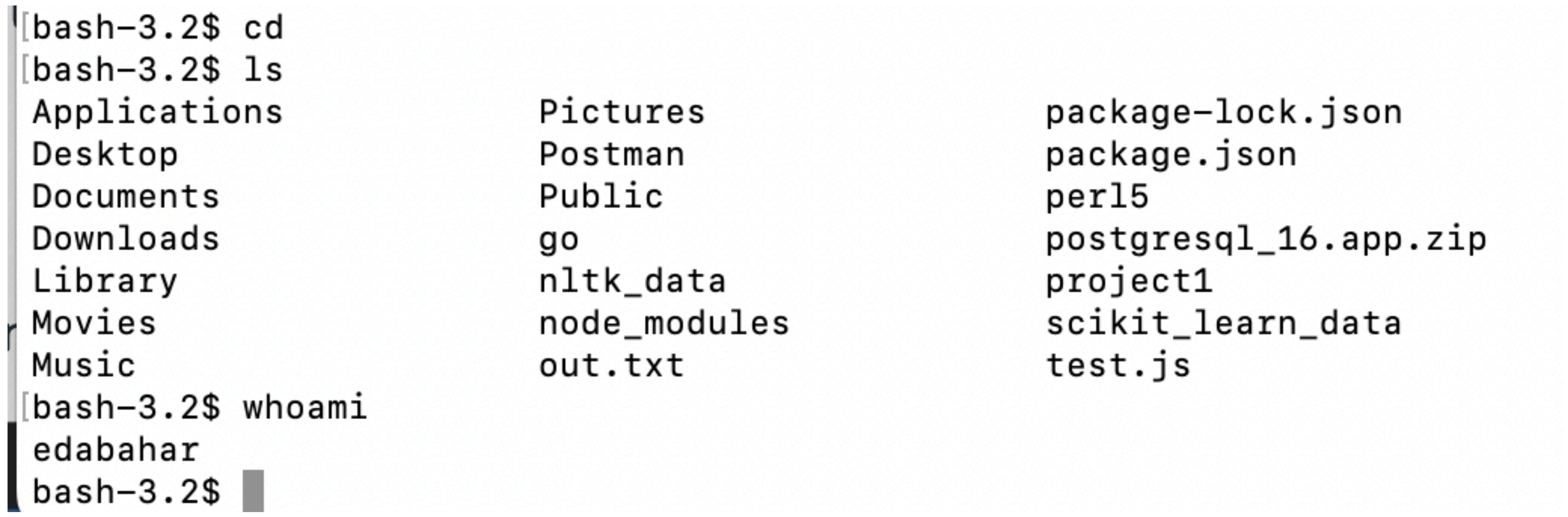
- A shell is a special user program that provides an interface for users to use operating system services. Shell accepts human-readable user commands and converts them into something the kernel can understand.
  - Command Line Shell: can be accessed using a command line interface (aka. Terminal)
  - Graphical Shell: GUI



Introducing bash

Bash (Bourne Again SHell)

- It is the most widely used shell in Linux systems and the default login shell in macOS (nowadays, macOS comes with zsh: another shell), and it can also be installed on Windows OS.



```
bash-3.2$ cd
bash-3.2$ ls
Applications            Pictures                package-lock.json
Desktop                 Postman                 package.json
Documents               Public                  perl5
Downloads               go                      postgresql_16.app.zip
Library                 nltk_data               project1
Movies                  node_modules            scikit_learn_data
Music                   out.txt                 test.js
bash-3.2$ whoami
edabahar
bash-3.2$ 
```

Introducing bash

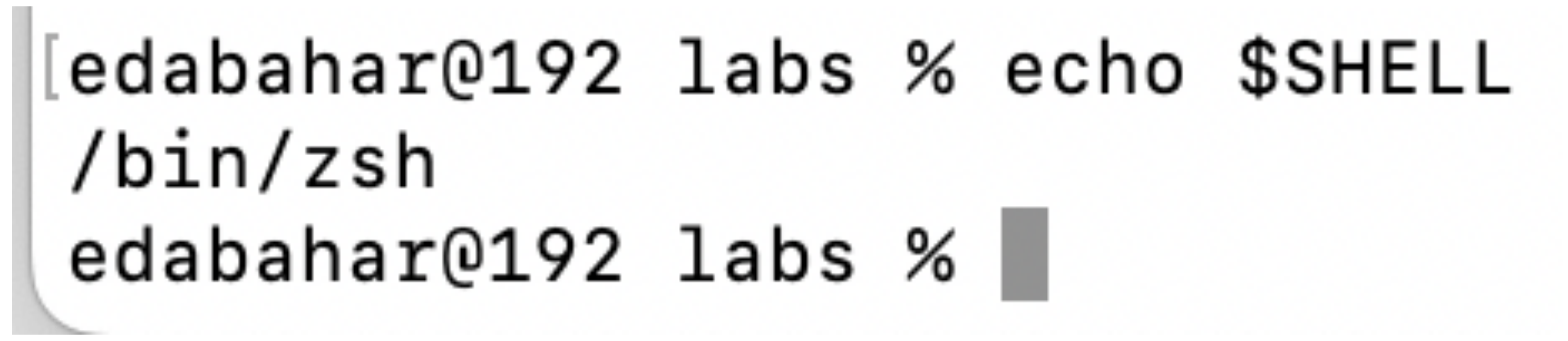
Are you running bash?

- You can check to see if you are running bash by typing:

  $ echo $SHELL

  /bin/bash

- If the above line gave you an error or didn't respond similarly to our example, you may be running a shell other than bash.

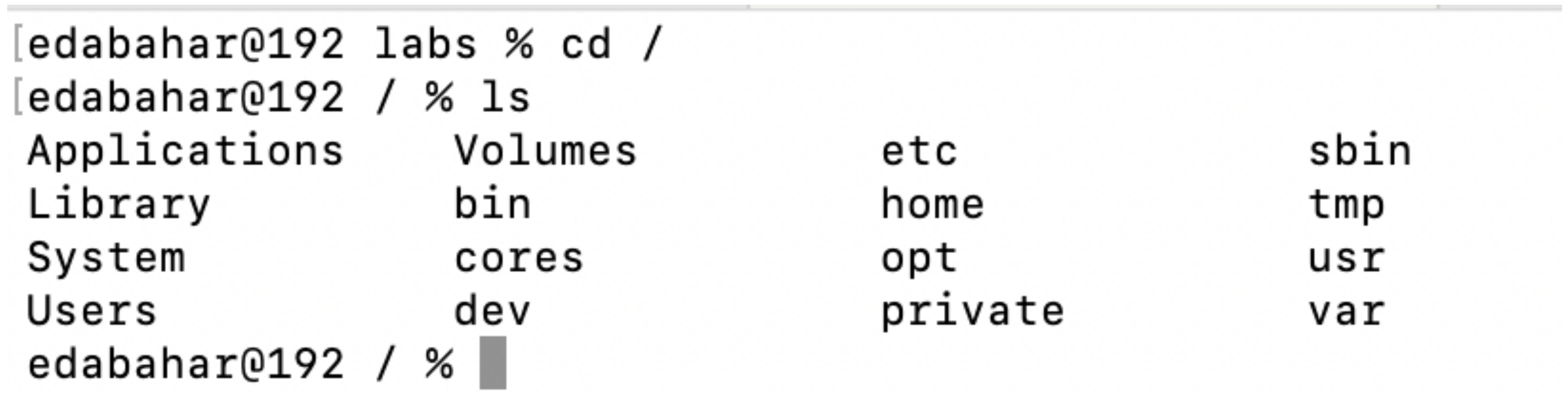


```
edabahar@192 labs % echo $SHELL
/bin/zsh
edabahar@192 labs % 
```

Linux commands

cd

- Let's start using **bash** to navigate around our filesystem.
  $ cd /



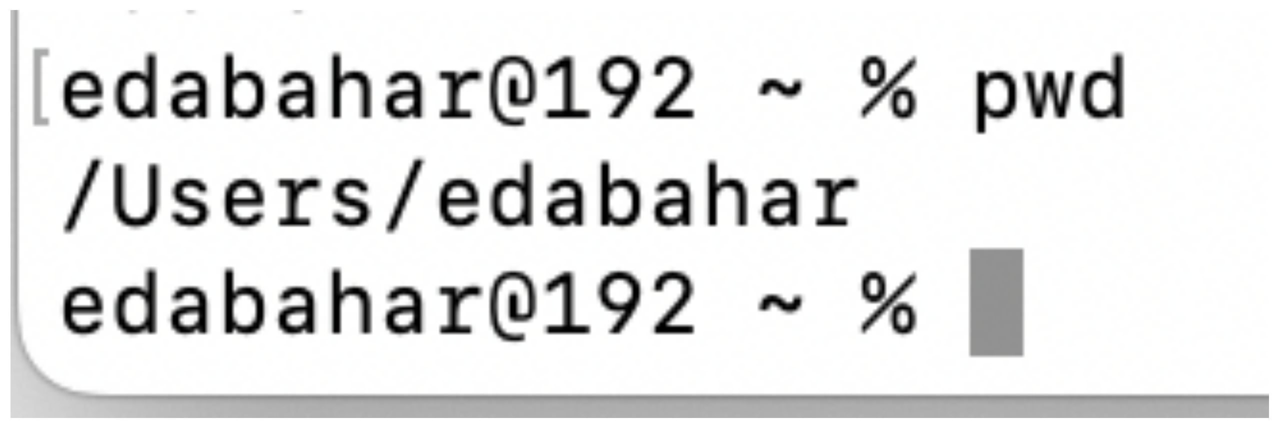
```
[edabahar@192 labs % cd /
[edabahar@192 / % ls
Applications     Volumes          etc              sbin
Library          bin              home             tmp
System           cores            opt              usr
Users            dev              private          var
edabahar@192 / %
```

- We have just navigated to **/**, also known as the **root directory**; all the directories on the system form a tree, and **/ is considered the top of this tree, or the root**. **cd** sets the directory where you are currently working, also known as the **current working directory**.

Linux commands

Paths - Absolute Paths

- An **absolute path** is a full path that specifies the location of a file or directory from the root directory (‘/’).
- To see bash’s current working directory, you can type:
  $ pwd



```
[edabahar@192 ~ % pwd
/Users/edabahar
edabahar@192 ~ %
```

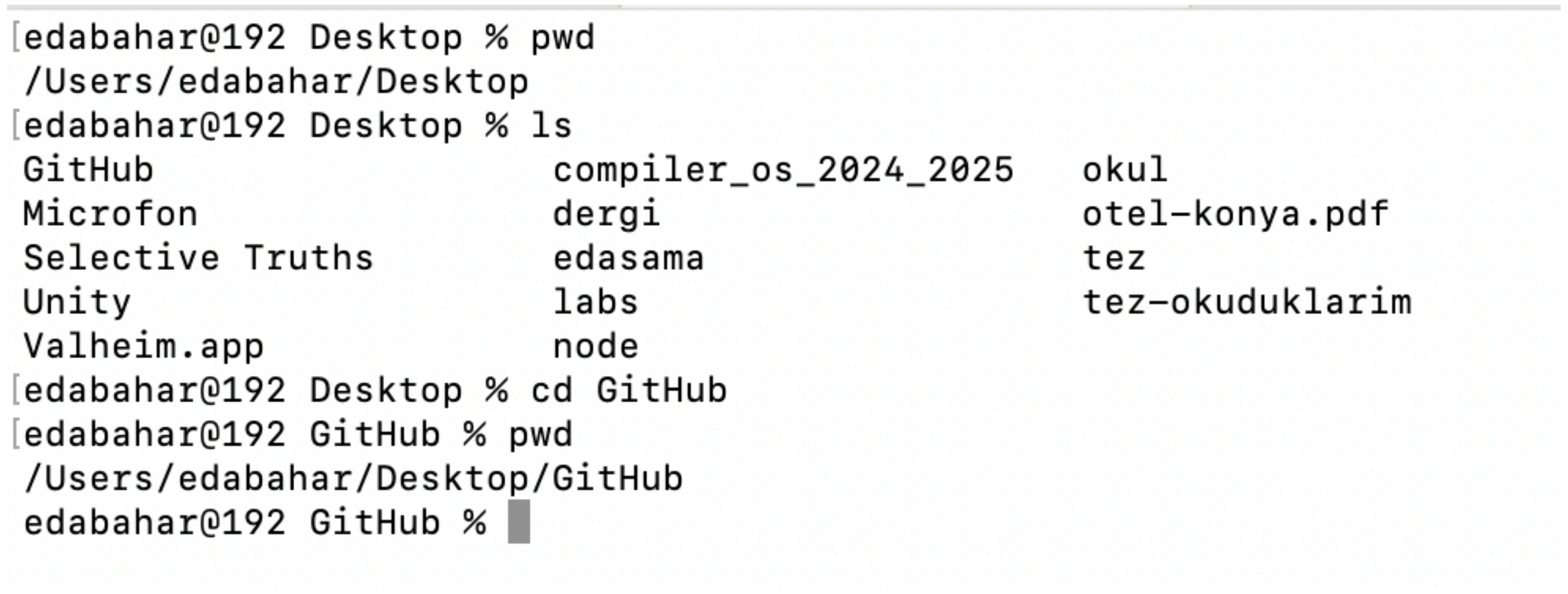
Here are some absolute paths:
/dev
/usr
/usr/bin
/home

As you can see, the one thing that all absolute paths have in common is that they **begin with /**.

Linux commands

Paths - Relative Paths

- A **relative path** specifies the location of a file or directory in relation to the current working directory.
- Relative paths never begin with **/**.

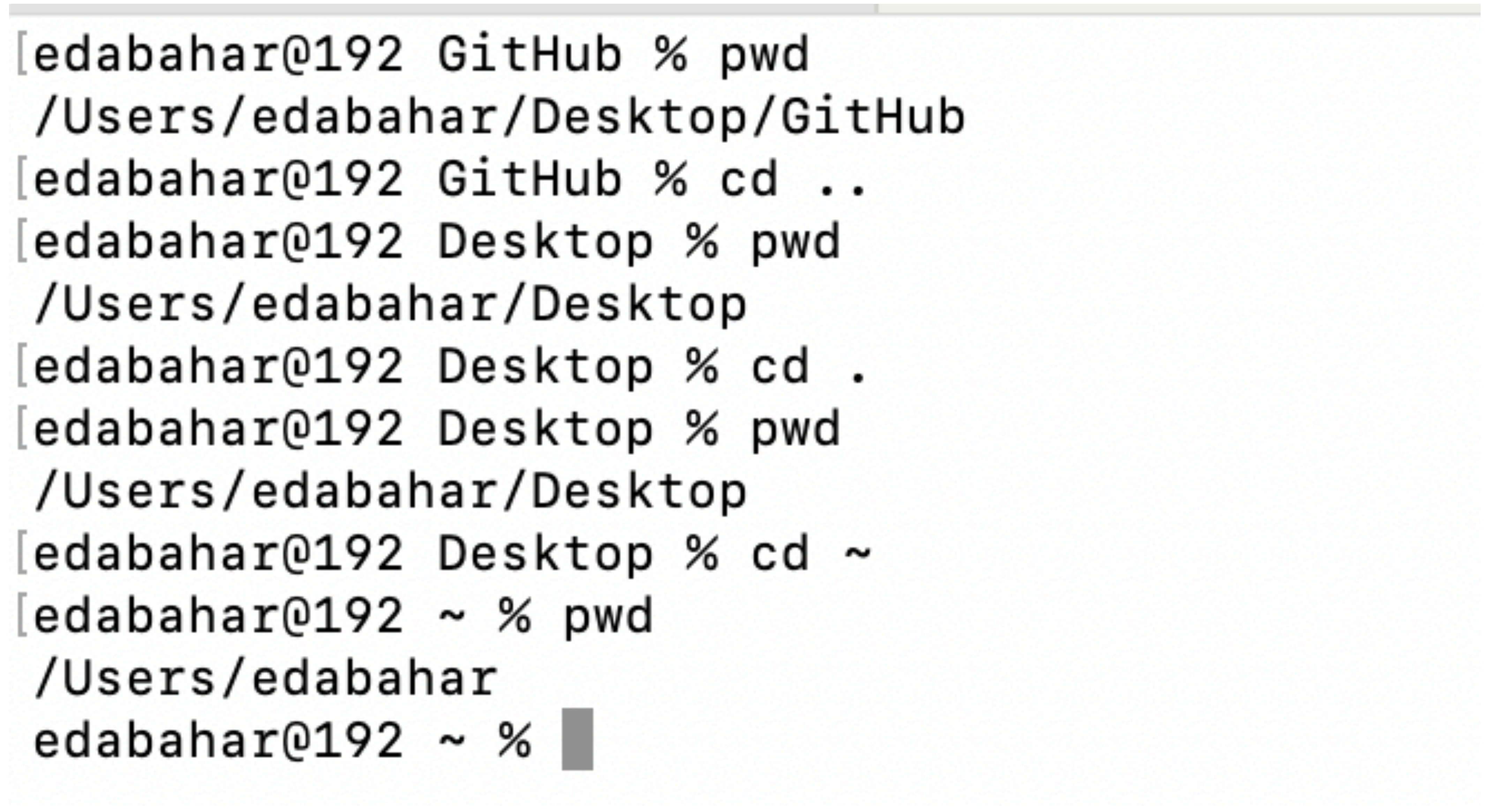


```
[edabahar@192 Desktop % pwd
/Users/edabahar/Desktop
[edabahar@192 Desktop % ls
GitHub                  compiler_os_2024_2025   okul
Microfon                dergi                   otel-konya.pdf
Selective Truths        edasama                 tez
Unity                   labs                    tez-okuduklarim
Valheim.app             node
[edabahar@192 Desktop % cd GitHub
[edabahar@192 GitHub % pwd
/Users/edabahar/Desktop/GitHub
edabahar@192 GitHub % 
```

Linux commands

Paths - . .. ~ -

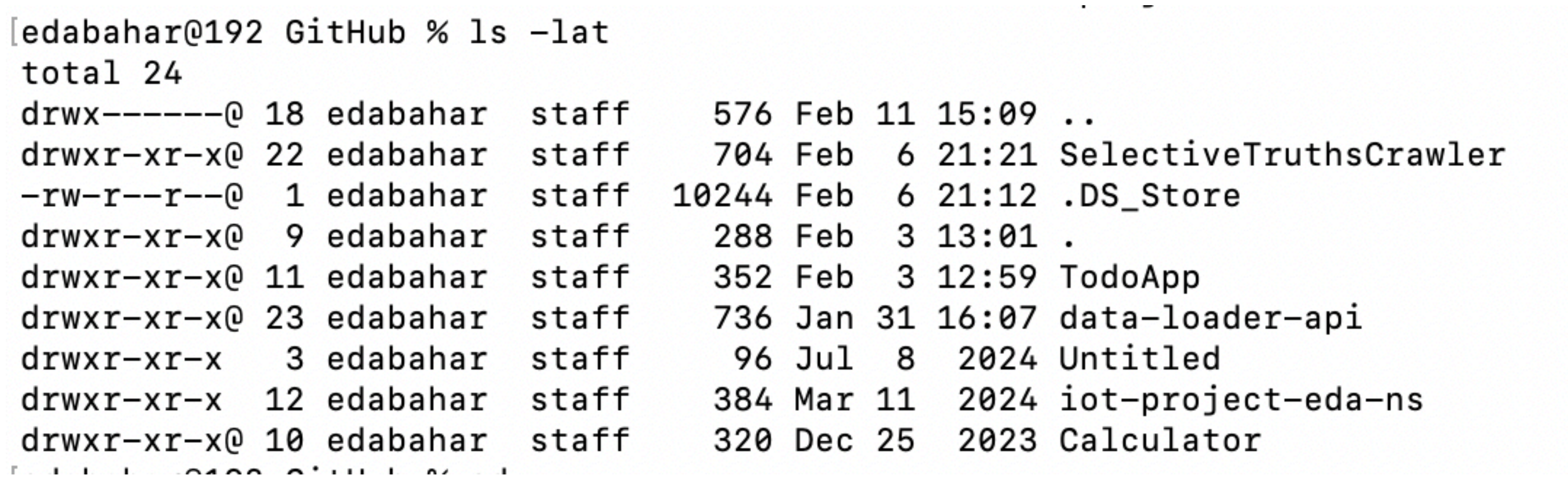
- UNIX offers a shortcut in the **relative pathname:**
  - **. (Dot):** represents the current directory
  - **.. (Double Dots):** represents the parent directory
  - **~ (Tilde)**: represents the home directory
- **Bonus: -**
  $ cd - : it takes you to previous directory



```
[edabahar@192 GitHub % pwd
 /Users/edabahar/Desktop/GitHub
[edabahar@192 GitHub % cd ..
[edabahar@192 Desktop % pwd
 /Users/edabahar/Desktop
[edabahar@192 Desktop % cd .
[edabahar@192 Desktop % pwd
 /Users/edabahar/Desktop
[edabahar@192 Desktop % cd ~
[edabahar@192 ~ % pwd
 /Users/edabahar
 edabahar@192 ~ %
```

Linux commands
ls

- **ls** is a Linux shell command that lists the directory contents of files and directories.
  ls **[option] [file/directory]**
  - **-l:** known as a long format that displays detailed information about files and directories.
  - **-a:** represent all files, including hidden files and directories.
  - **-t:** Sort files and directories by their last modification time, displaying the most recently modified ones first.

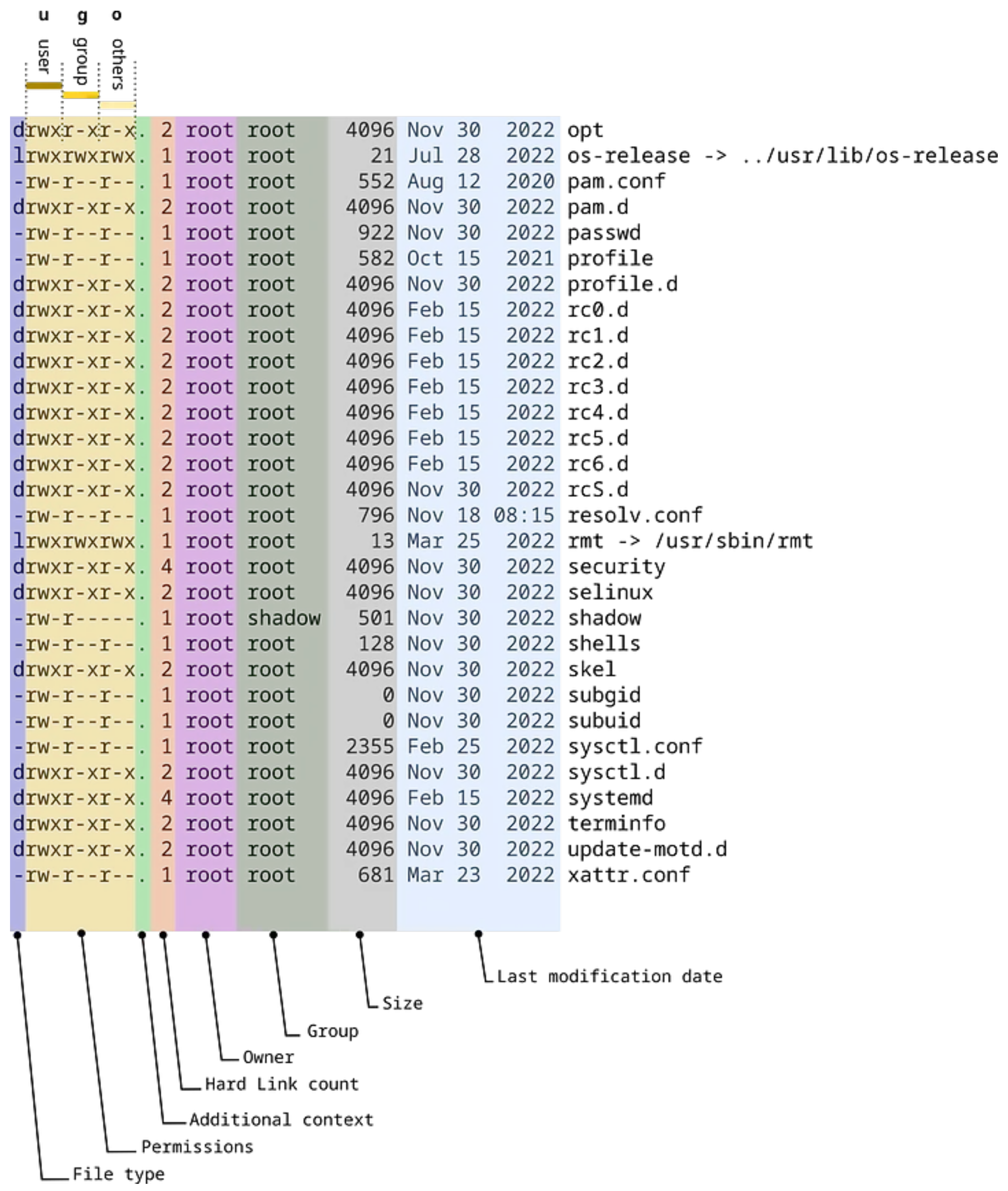


```
edabahar@192 GitHub % ls -lat
total 24
drwx------@ 18 edabahar  staff    576 Feb 11 15:09 ..
drwxr-xr-x@ 22 edabahar  staff    704 Feb  6 21:21 SelectiveTruthsCrawler
-rw-r--r--@  1 edabahar  staff  10244 Feb  6 21:12 .DS_Store
drwxr-xr-x@  9 edabahar  staff    288 Feb  3 13:01 .
drwxr-xr-x@ 11 edabahar  staff    352 Feb  3 12:59 TodoApp
drwxr-xr-x@ 23 edabahar  staff    736 Jan 31 16:07 data-loader-api
drwxr-xr-x   3 edabahar  staff     96 Jul  8  2024 Untitled
drwxr-xr-x  12 edabahar  staff    384 Mar 11  2024 iot-project-eda-ns
drwxr-xr-x@ 10 edabahar  staff    320 Dec 25  2023 Calculator
```

Linux commands

ls -l

- The first character indicates the file type:
  - - : regular file
  - d : directory
  - l : symbolic link
- Next 9 characters represent file permissions:
  - r : read
  - w : write
  - x : execute

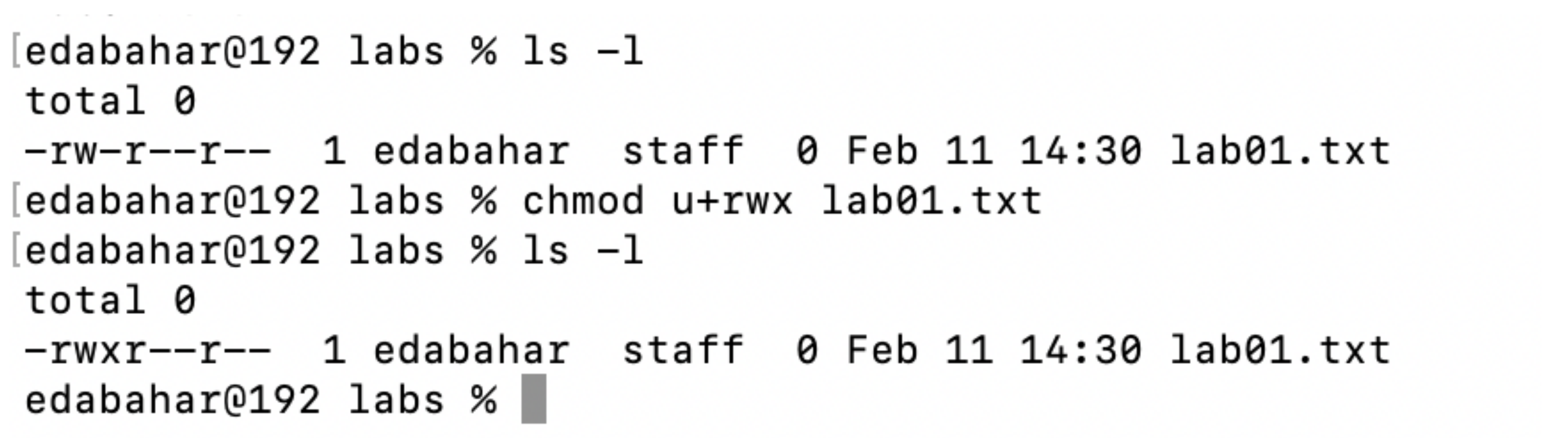


```
drwxr-xr-x. 2 root root    4096 Nov 30  2022 opt
lrwxrwxrwx. 1 root root      21 Jul 28  2022 os-release -> ../usr/lib/os-release
-rw-r--r--. 1 root root     552 Aug 12  2020 pam.conf
drwxr-xr-x. 2 root root    4096 Nov 30  2022 pam.d
-rw-r--r--. 1 root root     922 Nov 30  2022 passwd
-rw-r--r--. 1 root root     582 Oct 15  2021 profile
drwxr-xr-x. 2 root root    4096 Nov 30  2022 profile.d
drwxr-xr-x. 2 root root    4096 Feb 15  2022 rc0.d
drwxr-xr-x. 2 root root    4096 Feb 15  2022 rc1.d
drwxr-xr-x. 2 root root    4096 Feb 15  2022 rc2.d
drwxr-xr-x. 2 root root    4096 Feb 15  2022 rc3.d
drwxr-xr-x. 2 root root    4096 Feb 15  2022 rc4.d
drwxr-xr-x. 2 root root    4096 Feb 15  2022 rc5.d
drwxr-xr-x. 2 root root    4096 Feb 15  2022 rc6.d
drwxr-xr-x. 2 root root    4096 Nov 30  2022 rcS.d
-rw-r--r--. 1 root root     796 Nov 18 08:15 resolv.conf
lrwxrwxrwx. 1 root root      13 Mar 25  2022 rmt -> /usr/sbin/rmt
drwxr-xr-x. 4 root root    4096 Nov 30  2022 security
drwxr-xr-x. 2 root root    4096 Nov 30  2022 selinux
-rw-r-----. 1 root shadow   501 Nov 30  2022 shadow
-rw-r--r--. 1 root root     128 Nov 30  2022 shells
drwxr-xr-x. 2 root root    4096 Nov 30  2022 skel
-rw-r--r--. 1 root root       0 Nov 30  2022 subgid
-rw-r--r--. 1 root root       0 Nov 30  2022 subuid
-rw-r--r--. 1 root root    2355 Feb 25  2022 sysctl.conf
drwxr-xr-x. 2 root root    4096 Nov 30  2022 sysctl.d
drwxr-xr-x. 4 root root    4096 Feb 15  2022 systemd
drwxr-xr-x. 2 root root    4096 Nov 30  2022 terminfo
drwxr-xr-x. 2 root root    4096 Nov 30  2022 update-motd.d
-rw-r--r--. 1 root root     681 Mar 23  2022 xattr.conf
```

Linux commands

Permissions - chmod

- In Unix operating systems, the **chmod** command is used to change the file's access mode. The name is an abbreviation of **change mode**.
  chmod **[options] [mode] [File_name]**
- The "mode" helps set new permissions that must be applied to files or directories. This mode can be specified in several ways; we will discuss the Symbolic and the Octal modes.
  - **Symbolic Mode**: we have to combine **letters** and **operators** to set or tell what to do with permissions.
    chmod u+rwx lab01.txt
    - **+** : add permissions
    - **-** : remove permissions
    - **=** : set the permissions to the specified values
    - **u** : owner
    - **g** : group
    - **o** : others
    - **a** : all (owner, groups, others)

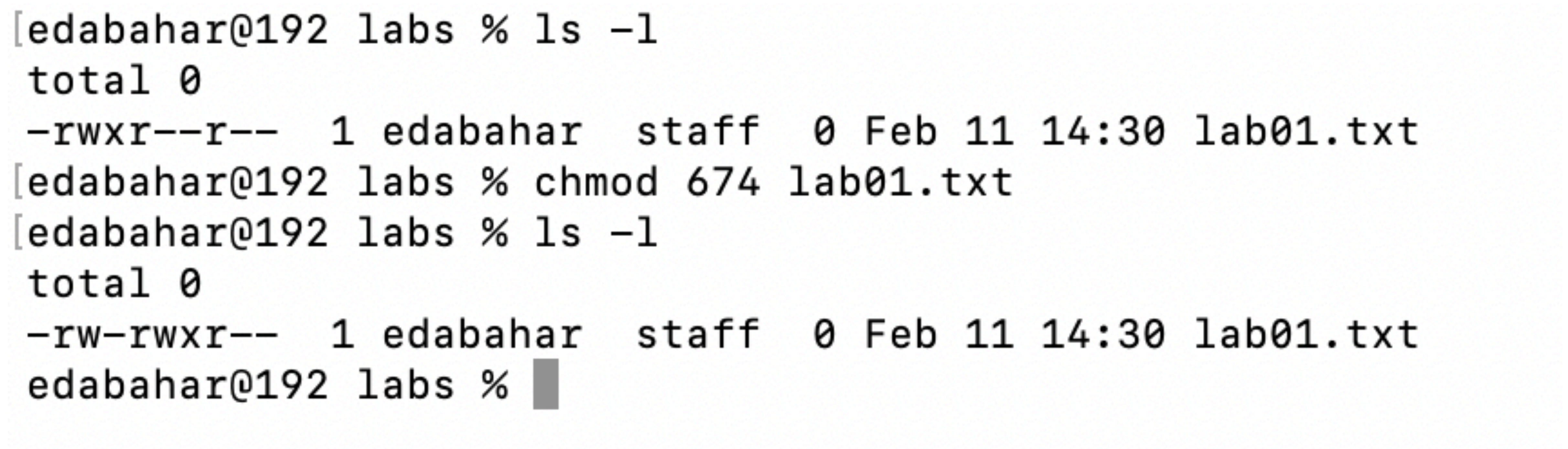


```
edabahar@192 labs % ls -l
total 0
-rw-r--r--  1 edabahar  staff  0 Feb 11 14:30 lab01.txt
edabahar@192 labs % chmod u+rwx lab01.txt
edabahar@192 labs % ls -l
total 0
-rwxr--r--  1 edabahar  staff  0 Feb 11 14:30 lab01.txt
edabahar@192 labs %
```

Linux commands
Permissions - chmod

- In Unix operating systems, the **chmod** command is used to change the file's access mode. The name is an abbreviation of **change mode**.
  chmod **[options] [mode] [File_name]**
- The "mode" helps set new permissions that must be applied to files or directories. This mode can be specified in several ways; we will discuss the Symbolic and the Octal modes.
  - **Octal Mode**: we specify permission using a three-digit number.
    chmod 674 lab01.txt
    - **4 :** read permission
    - **2** : write permission
    - **1** : execute permission

- In this example
  chmod 674 lab01.txt
  - **6**: **read** and **write** permission for **owner**
  - **7**: **read, write** and **execute** permission for **group**
  - **4**: **read** permission for **others**



```
edabahar@192 labs % ls -l
total 0
-rwxr--r--  1 edabahar  staff  0 Feb 11 14:30 lab01.txt
edabahar@192 labs % chmod 674 lab01.txt
edabahar@192 labs % ls -l
total 0
-rw-rwxr--  1 edabahar  staff  0 Feb 11 14:30 lab01.txt
edabahar@192 labs %
```

Linux commands

mkdir

- In Linux, the ‘**mkdir**’ command is like a magic wand for easily creating folders. ‘**mkdir**’ stands for “**make directory**”.
  mkdir [options...] [directory_name]
  mkdir lab

  - -help: displays help-related information for the mkdir command and exits.

  - -version: displays the version number and additional information about the license for mkdir.

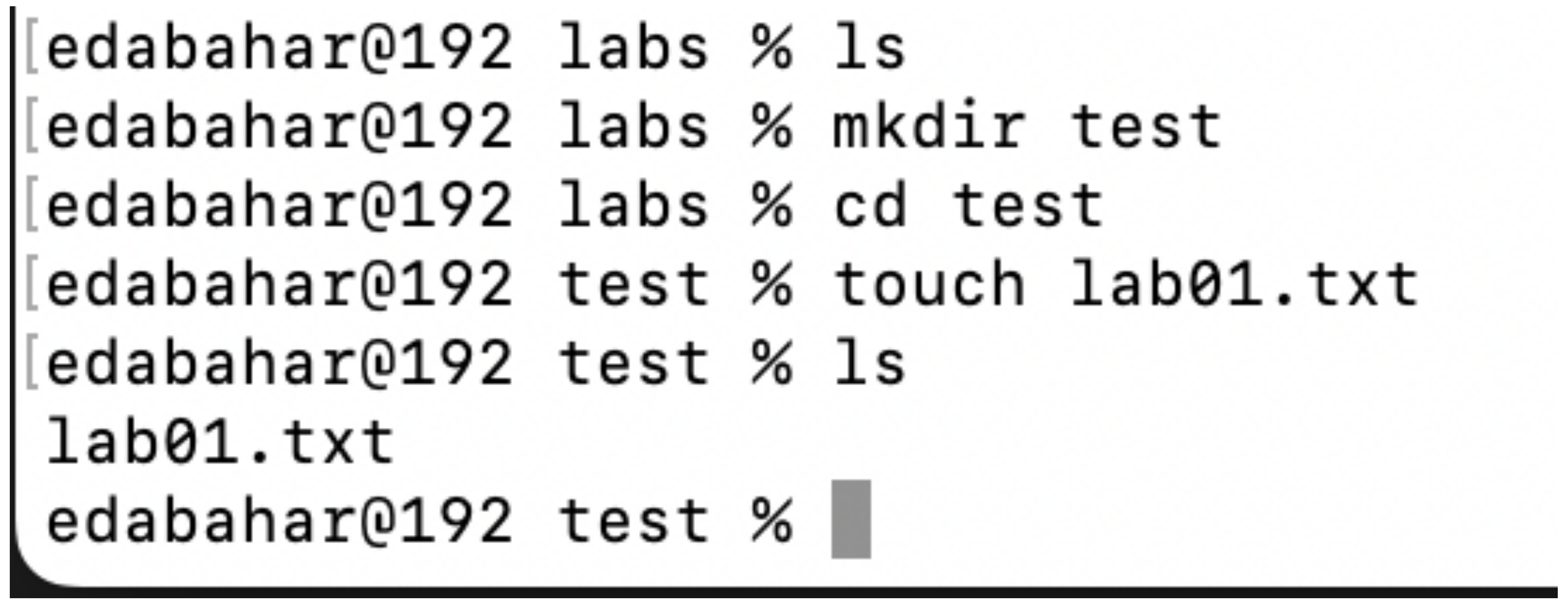
  - -m: sets file modes or permissions for the created directories. The syntax follows that of the chmod command.

Linux commands

touch

- The **touch** command is a fundamental command used in the UNIX/Linux operating system to create, change, and modify the timestamps of a file.
  touch [options] file_name
  touch lab01.txt
  touch lab01.txt lab02.txt

- If the file does not exist, then a new, empty file will be created.

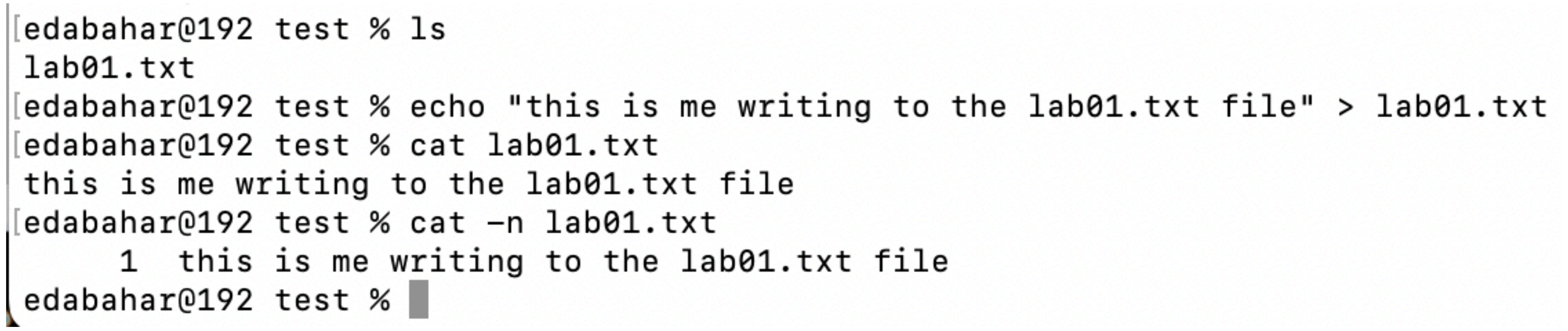


```
[edabahar@192 labs % ls
[edabahar@192 labs % mkdir test
[edabahar@192 labs % cd test
[edabahar@192 test % touch lab01.txt
[edabahar@192 test % ls
 lab01.txt
 edabahar@192 test %
```

Linux commands

echo - cat

- Now that the file exists as **lab01.txt** under the **test** directory, let's add some data to the file using **echo**.
  echo **"this is me writing to the lab01.txt file" > lab01.txt**
- The greater-than sign “>” tells the shell to write **echo**’s output to a file called lab01.txt. This file will be created if it does not exist and will be overwritten if it is exist
- The **cat** command in Linux is more than just a simple tool; it allows **users to view, concatenate, create, copy, merge**, and **manipulate file contents**.
- To display the content of the file in the terminal, use the **cat** command:
  cat **[OPTION] [FILE]**



```
[edabahar@192 test % ls
lab01.txt
[edabahar@192 test % echo "this is me writing to the lab01.txt file" > lab01.txt
[edabahar@192 test % cat lab01.txt
this is me writing to the lab01.txt file
[edabahar@192 test % cat -n lab01.txt
     1  this is me writing to the lab01.txt file
edabahar@192 test % 
```

Linux commands

cp - mv

- A common thing to do in Linux is copying files. The key tool for this task is the “cp” command.
  cp **source_file destination**

- **cp lab01.txt lab01_copy.txt**

  - In this example, if the lab01_copy.txt file does not exist, it is created, and it is a copy of the lab01.txt file. If not, it is overwritten without any warning.

- **cp lab01.txt lab01_copy.txt /new**

  - This command copies lab01.txt and lab01_copy.txt files to **new** directory.

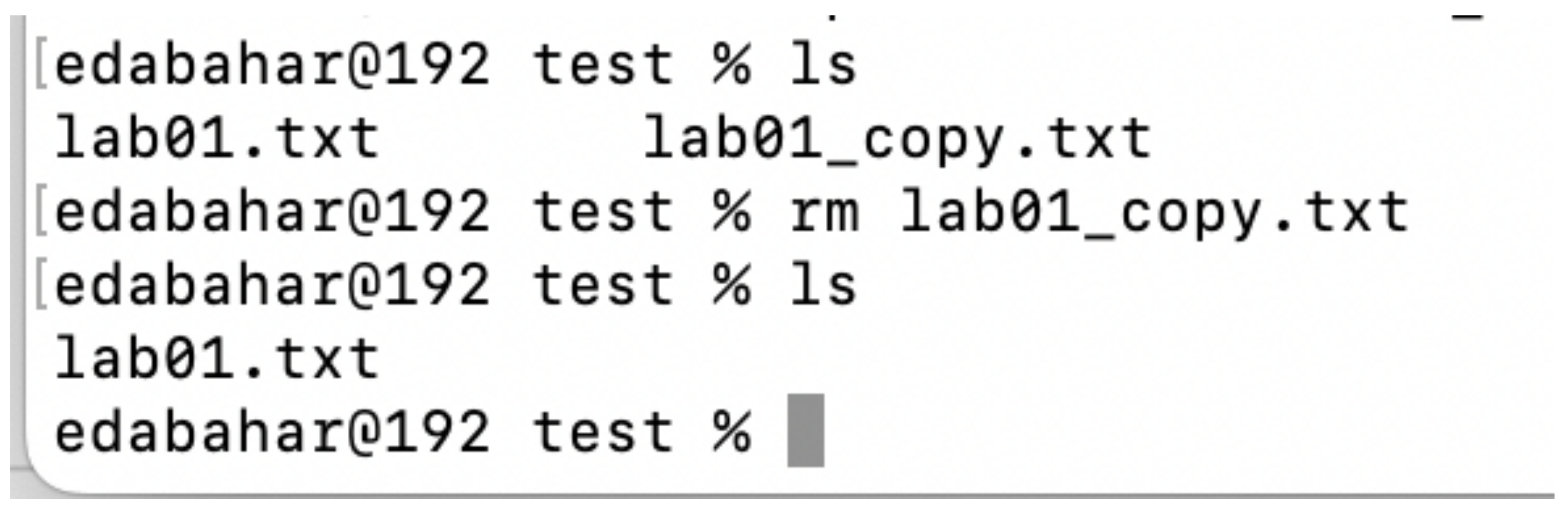
- **cp -R source_directory destination_directory**

  - This command copies all files from **source_directory** into **destination_directory**.

Removing files

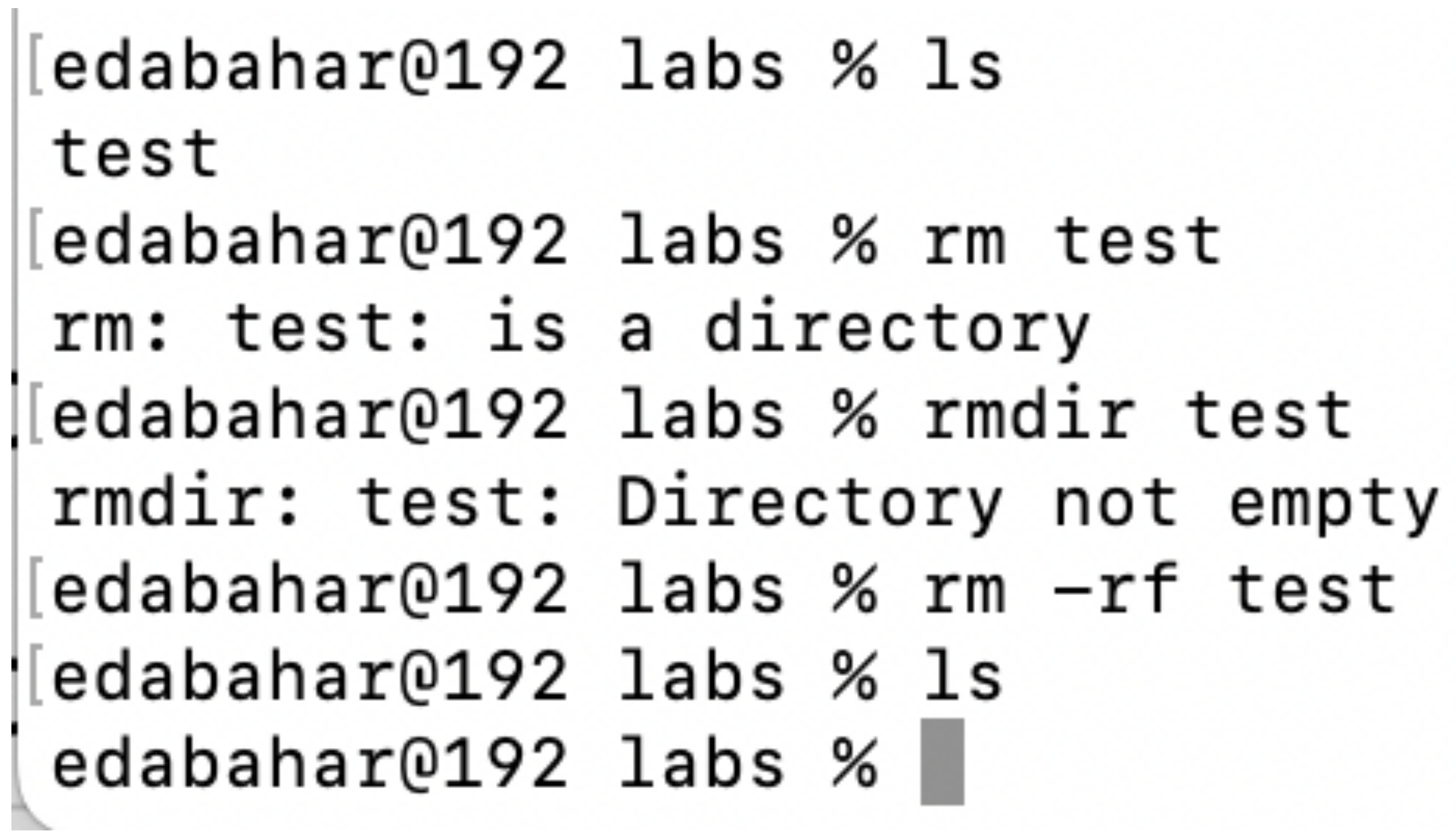
rm - rmdir

- **rm** stands for **remove** here. rm command removes objects such as files, directories, symbolic links, etc.
  rm **[OPTION]... FILE...**
- **rm lab01_copy.txt**
  - It removes the lab01_copy.txt file.



```
[edabahar@192 test % ls
lab01.txt        lab01_copy.txt
[edabahar@192 test % rm lab01_copy.txt
[edabahar@192 test % ls
lab01.txt
edabahar@192 test % 
```

- **rm -rf test**
  - It removes the test directory, which is not an empty directory.
- The **rmdir** command in Linux is specifically designed to remove empty directories.



```
[edabahar@192 labs % ls
test
[edabahar@192 labs % rm test
rm: test: is a directory
[edabahar@192 labs % rmdir test
rmdir: test: Directory not empty
[edabahar@192 labs % rm -rf test
[edabahar@192 labs % ls
edabahar@192 labs % 
```

Linux commands

man

- The **man** command, short for manual, is a powerful tool in the Linux operating system that allows users to access detailed information about various commands, utilities, and system calls. The **man** command is essentially the Linux **manual reader**.
  man **[option] [command]**

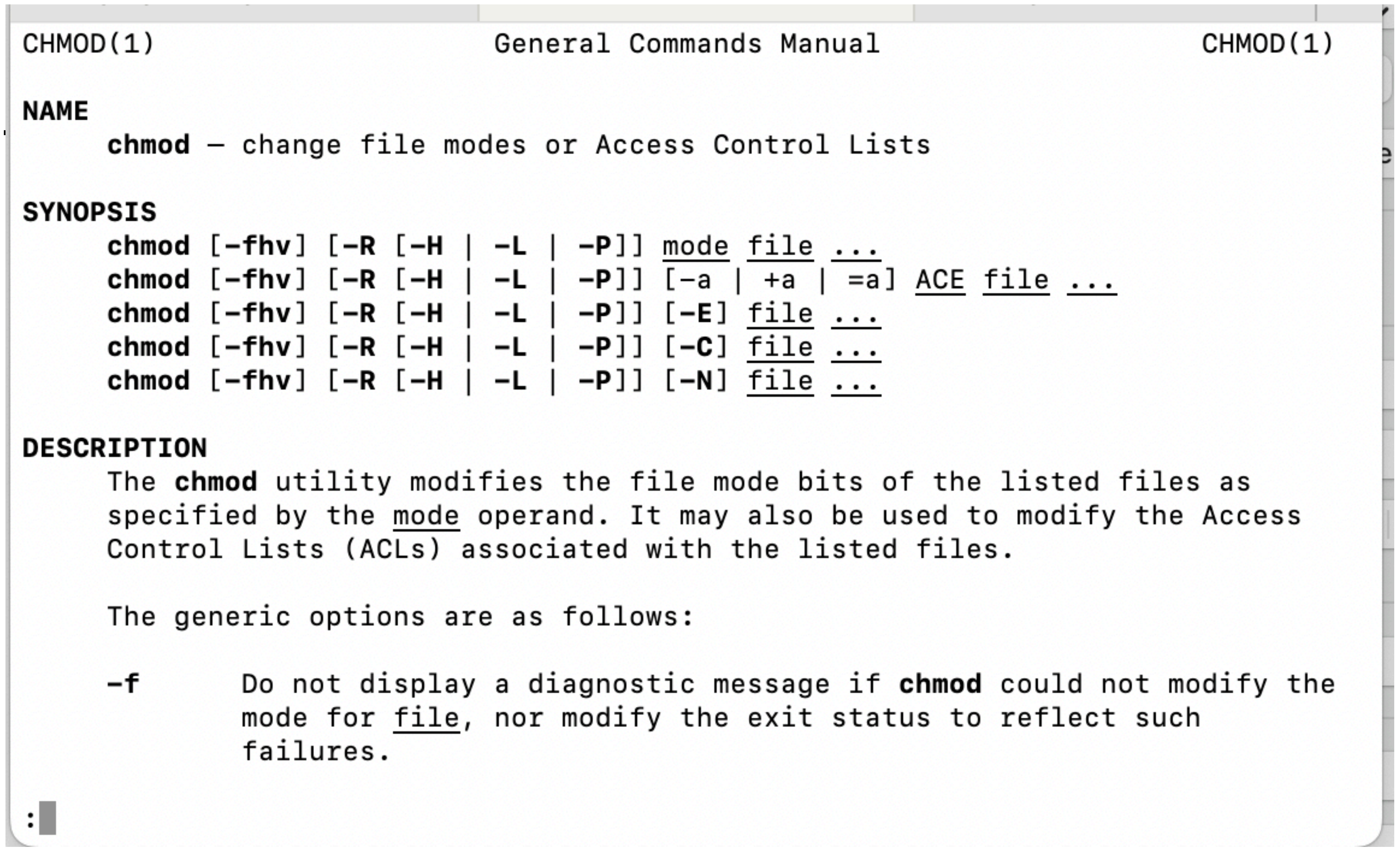
- man command: display the manual page for the specified command.

- -f, -whatis: display a concise one-line description of the command.

- -k, -apropos: search for commands related to a given keyword.

- -a, -all: display all matching manual pages for the specified command.

- You can exit with “**q**”.



```
CHMOD(1)                     General Commands Manual                    CHMOD(1)

NAME
     chmod – change file modes or Access Control Lists

SYNOPSIS
     chmod [-fhv] [-R [-H | -L | -P]] mode file ...
     chmod [-fhv] [-R [-H | -L | -P]] [-a | +a | =a] ACE file ...
     chmod [-fhv] [-R [-H | -L | -P]] [-E] file ...
     chmod [-fhv] [-R [-H | -L | -P]] [-C] file ...
     chmod [-fhv] [-R [-H | -L | -P]] [-N] file ...

DESCRIPTION
     The chmod utility modifies the file mode bits of the listed files as
     specified by the mode operand. It may also be used to modify the Access
     Control Lists (ACLs) associated with the listed files.

     The generic options are as follows:

     -f      Do not display a diagnostic message if chmod could not modify the
             mode for file, nor modify the exit status to reflect such
             failures.

:
```

Linux commands
grep

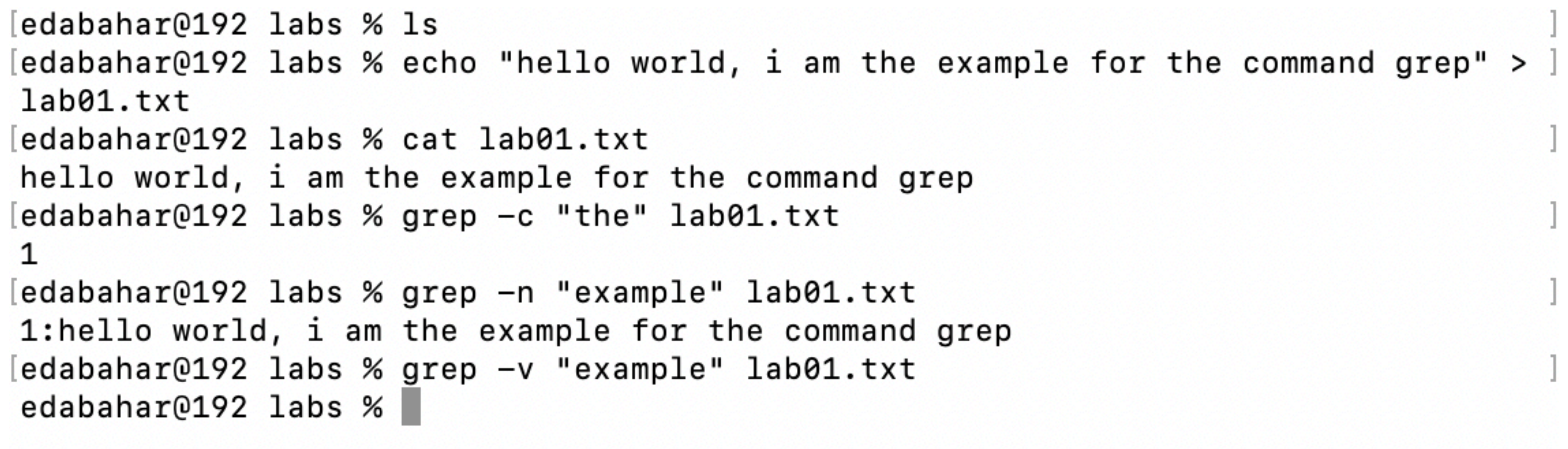
- It is used to search for a specific pattern within files. It stands for "global regular expression print."
  grep **[options] pattern [files]**

- -c : this prints only a count of the lines that match a pattern.

- -l : displays list of a filenames only.

- -n : displays the matched lines and their line numbers.

- -v : prints out all the lines that do not match the pattern.

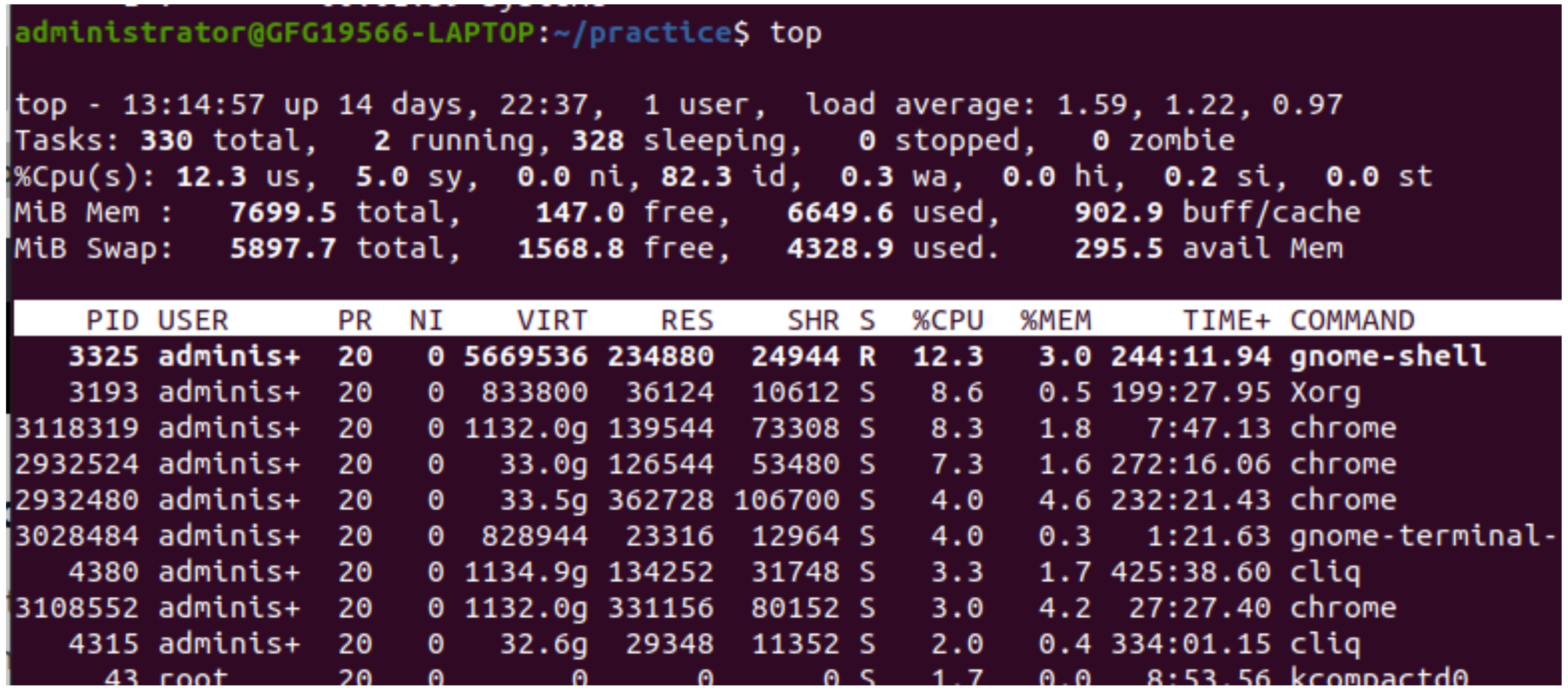


```
edabahar@192 labs % ls
edabahar@192 labs % echo "hello world, i am the example for the command grep" >
lab01.txt
edabahar@192 labs % cat lab01.txt
hello world, i am the example for the command grep
edabahar@192 labs % grep -c "the" lab01.txt
1
edabahar@192 labs % grep -n "example" lab01.txt
1:hello world, i am the example for the command grep
edabahar@192 labs % grep -v "example" lab01.txt
edabahar@192 labs %
```

Linux commands

top

- In Linux, the **top** command is a dynamic and interactive tool that provides real-time information about system processes. It offers a comprehensive view of running processes, system resource utilization, and other critical system metrics.
    - PID: process id
    - USER: owner of the process
    - PR: priority
    - VIRT: virtual memory usage
    - COMMAND: command or process name
    - %CPU: percentage of CPU usage
    - %MEM: percentage of memory usage
    - TIME+: total CPU time
    - S: process status (S: Sleeping, R: Running, I: Idle)
    - …

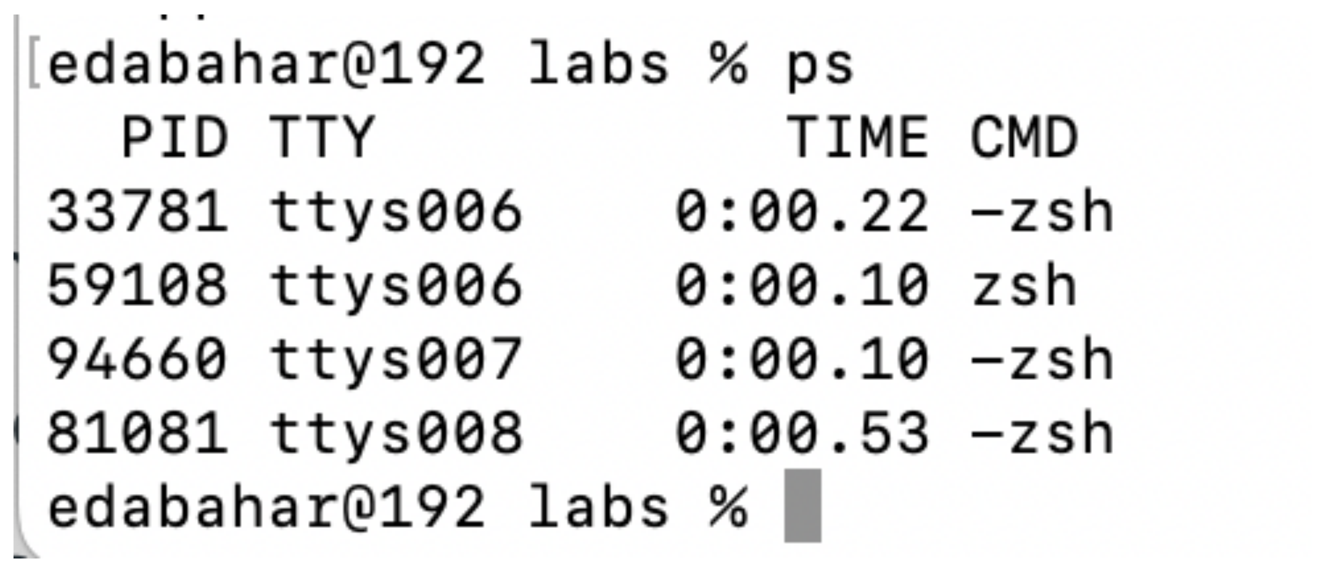


```
administrator@GFG19566-LAPTOP:~/practice$ top

top - 13:14:57 up 14 days, 22:37,  1 user,  load average: 1.59, 1.22, 0.97
Tasks: 330 total,   2 running, 328 sleeping,   0 stopped,   0 zombie
%Cpu(s): 12.3 us,  5.0 sy,  0.0 ni, 82.3 id,  0.3 wa,  0.0 hi,  0.2 si,  0.0 st
MiB Mem :   7699.5 total,    147.0 free,   6649.6 used,    902.9 buff/cache
MiB Swap:   5897.7 total,   1568.8 free,   4328.9 used.    295.5 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   3325 adminis+  20   0 5669536 234880  24944 R  12.3   3.0 244:11.94 gnome-shell
   3193 adminis+  20   0  833800  36124  10612 S   8.6   0.5 199:27.95 Xorg
3118319 adminis+  20   0 1132.0g 139544  73308 S   8.3   1.8   7:47.13 chrome
2932524 adminis+  20   0   33.0g 126544  53480 S   7.3   1.6 272:16.06 chrome
2932480 adminis+  20   0   33.5g 362728 106700 S   4.0   4.6 232:21.43 chrome
3028484 adminis+  20   0  828944  23316  12964 S   4.0   0.3   1:21.63 gnome-terminal-
   4380 adminis+  20   0 1134.9g 134252  31748 S   3.3   1.7 425:38.60 cliq
3108552 adminis+  20   0 1132.0g 331156  80152 S   3.0   4.2  27:27.40 chrome
   4315 adminis+  20   0   32.6g  29348  11352 S   2.0   0.4 334:01.15 cliq
     43 root      20   0       0      0      0 S   1.7   0.0   8:53.56 kcompactd0
```

Listing processes

ps

- Linux is a **multitasking** and **multi-user operating system**. It allows multiple **processes** to run concurrently without interfering.
- In Linux, a process is a running instance of a program. (We will talk about this later in the semester.)
- The **ps** command, which stands for "process status," is like a computer tool that helps you see what's happening inside your Linux computer.
    - **a** : lists all running processes for all users.
    - **u** : expands the output to include additional information like CPU and memory usage.
    - **x** : includes processes without a TTY, showing background processes not tied to a specific terminal session.
- In the output of **ps**:
    - **PID**: the unique process ID
    - **TTY**: terminal type that the user is logged into
    - **TIME**: amount of CPU in minutes and seconds that the process has been running
    - **CMD**: name of the command that launched the process



```
edabahar@192 labs % ps
  PID TTY           TIME CMD
33781 ttys006    0:00.22 -zsh
59108 ttys006    0:00.10 zsh
94660 ttys007    0:00.10 -zsh
81081 ttys008    0:00.53 -zsh
edabahar@192 labs %
```

Interacting with processes

kill

- **kill** command in Linux is a built-in command that is used to terminate processes manually.
  kill **[signal] PID**

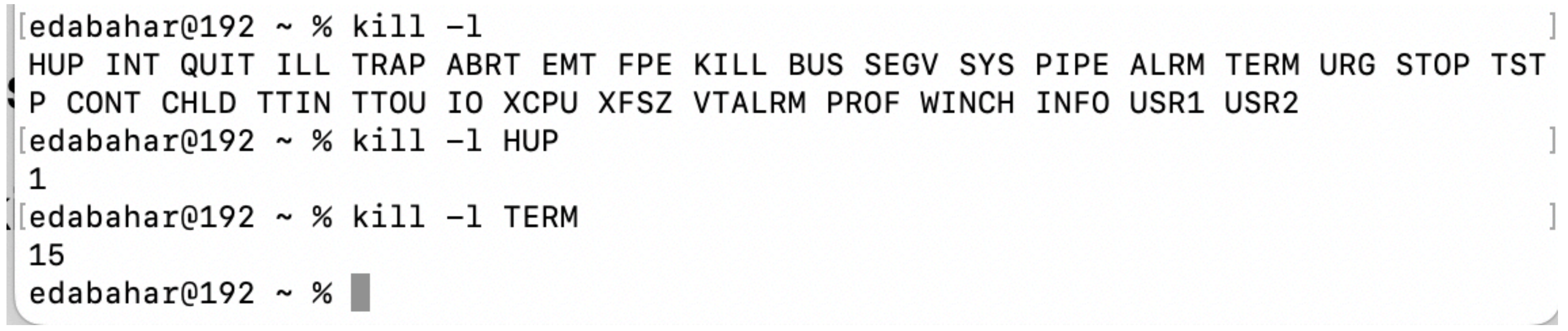
- **[signal] =** We have to specify the signal and if we don't specify the signal, the default signal `**SIGKILL**` is sent to kill the process

  - **-9**: **SIGKILL**: it kills the process

  - **-2**: **SIGINT**: it interrupts from keyboard

  - **-15**: **SIGTERM**: it terminates the signal

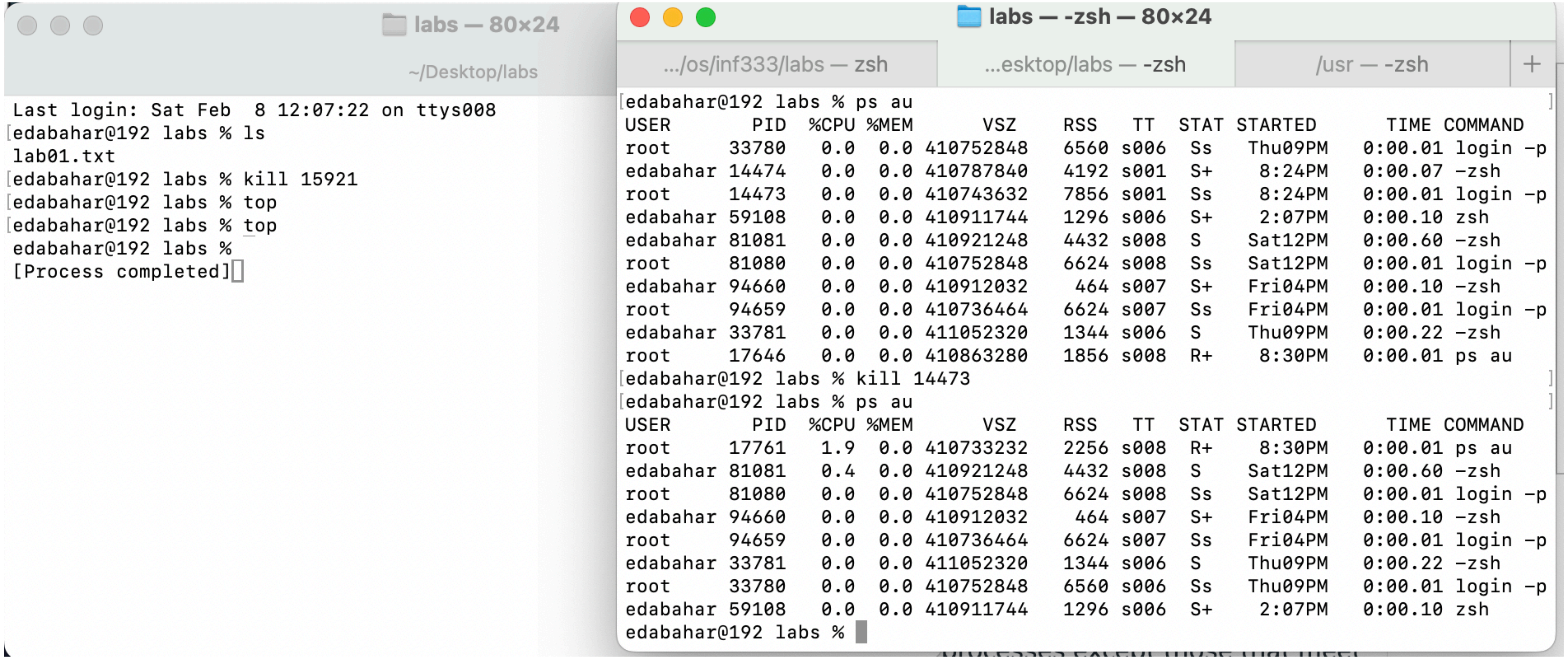
  - **kill -l**: shows the list of options, for example **kill -l TERM** gives the number of the signal.



```
edabahar@192 ~ % kill -l
HUP INT QUIT ILL TRAP ABRT EMT FPE KILL BUS SEGV SYS PIPE ALRM TERM URG STOP TST
P CONT CHLD TTIN TTOU IO XCPU XFSZ VTALRM PROF WINCH INFO USR1 USR2
edabahar@192 ~ % kill -l HUP
1
edabahar@192 ~ % kill -l TERM
15
edabahar@192 ~ %
```

Killing a process

ps - kill



```
Last login: Sat Feb  8 12:07:22 on ttys008
[edabahar@192 labs % ls
lab01.txt
[edabahar@192 labs % kill 15921
[edabahar@192 labs % top
[edabahar@192 labs % top
edabahar@192 labs %
[Process completed]
```

```
[edabahar@192 labs % ps au                                                      ]
USER       PID  %CPU %MEM      VSZ    RSS   TT  STAT STARTED      TIME COMMAND
root     33780   0.0  0.0 410752848   6560 s006  Ss   Thu09PM   0:00.01 login -p
edabahar 14474   0.0  0.0 410787840   4192 s001  S+    8:24PM   0:00.07 -zsh
root     14473   0.0  0.0 410743632   7856 s001  Ss    8:24PM   0:00.01 login -p
edabahar 59108   0.0  0.0 410911744   1296 s006  S+    2:07PM   0:00.10 zsh
edabahar 81081   0.0  0.0 410921248   4432 s008  S     Sat12PM   0:00.60 -zsh
root     81080   0.0  0.0 410752848   6624 s008  Ss    Sat12PM   0:00.01 login -p
edabahar 94660   0.0  0.0 410912032    464 s007  S+    Fri04PM   0:00.10 -zsh
root     94659   0.0  0.0 410736464   6624 s007  Ss    Fri04PM   0:00.01 login -p
edabahar 33781   0.0  0.0 411052320   1344 s006  S     Thu09PM   0:00.22 -zsh
root     17646   0.0  0.0 410863280   1856 s008  R+    8:30PM   0:00.01 ps au
[edabahar@192 labs % kill 14473                                                 ]
[edabahar@192 labs % ps au                                                      ]
USER       PID  %CPU %MEM      VSZ    RSS   TT  STAT STARTED      TIME COMMAND
root     17761   1.9  0.0 410733232   2256 s008  R+    8:30PM   0:00.01 ps au
edabahar 81081   0.4  0.0 410921248   4432 s008  S     Sat12PM   0:00.60 -zsh
root     81080   0.0  0.0 410752848   6624 s008  Ss    Sat12PM   0:00.01 login -p
edabahar 94660   0.0  0.0 410912032    464 s007  S+    Fri04PM   0:00.10 -zsh
root     94659   0.0  0.0 410736464   6624 s007  Ss    Fri04PM   0:00.01 login -p
edabahar 33781   0.0  0.0 411052320   1344 s006  S     Thu09PM   0:00.22 -zsh
root     33780   0.0  0.0 410752848   6560 s006  Ss    Thu09PM   0:00.01 login -p
edabahar 59108   0.0  0.0 410911744   1296 s006  S+    2:07PM   0:00.10 zsh
edabahar@192 labs %
```