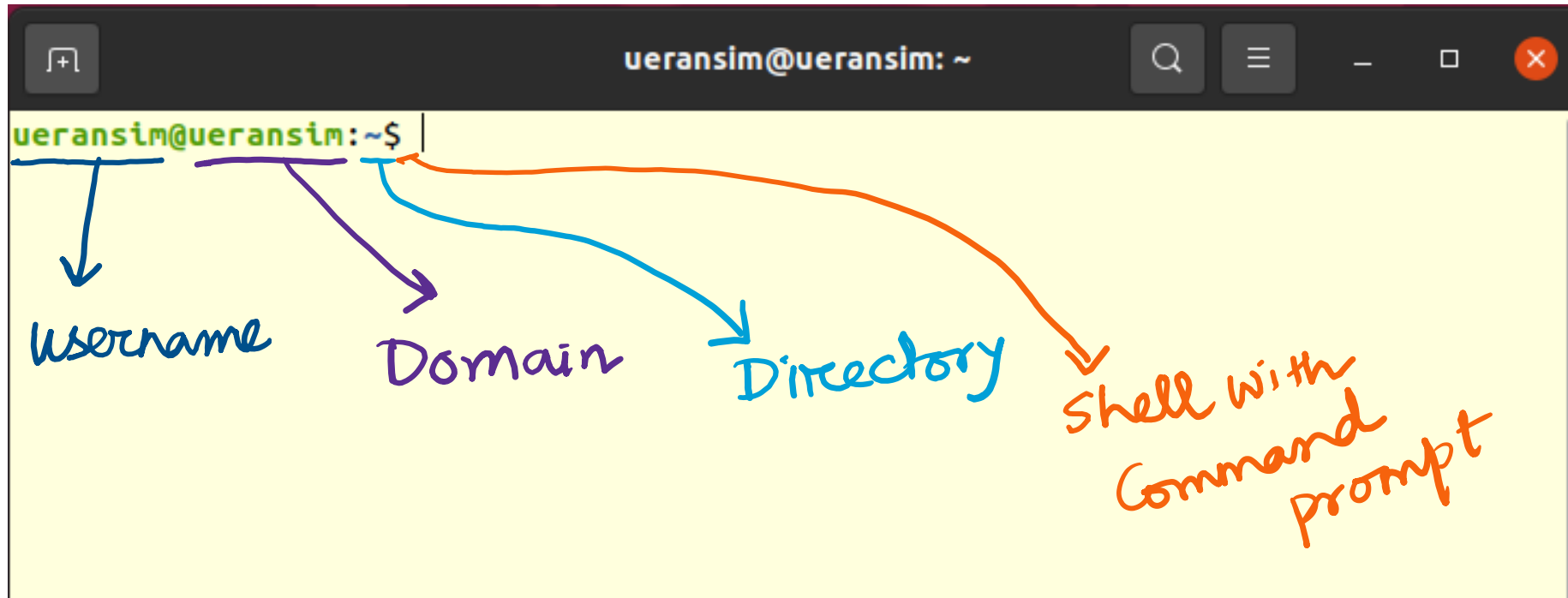


Linux Commands

Samaresh Bera

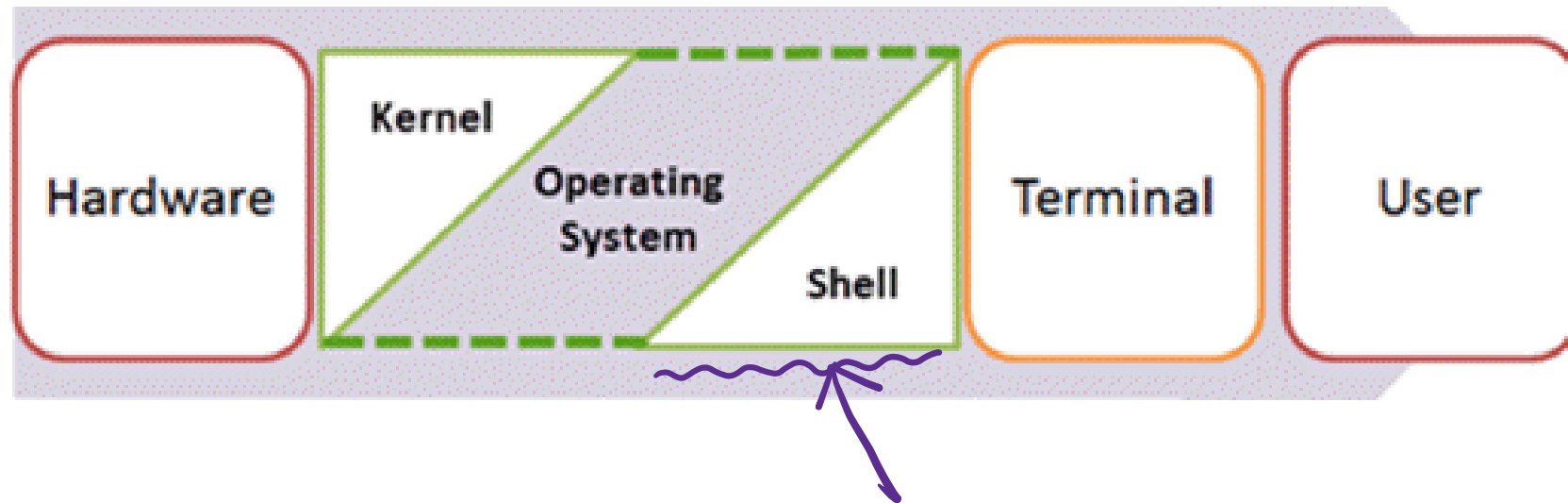


Opening a terminal



What happens when you open a terminal

- Linux/Unix starts another program called **shell**
- Where do we have the shell in systems?



Some facts: Kernel and Shell

- **Kernel**
 - Nucleus of a computer
 - Makes the communication between the hardware and software possible.
 - **Innermost part** of an operating system
- **Shell**
 - **Outermost part** of an operating system
 - Takes input from you in the form of commands, processes it, and then gives an output
 - Interface through which a user works on the programs, commands, and scripts
 - Accessed by a terminal that runs it
- When you run the terminal
 - The Shell issues **a command prompt (usually \$)**
- The Shell wraps around the delicate interior of an Operating system protecting it from accidental damage. Hence the name **Shell**.

What does a shell do?

- Interprets user-commands
- Manages execution of the commands
- Shell commands are case-sensitive

If I don't know exact syntax of command

- Type `man` and the command

```
ueransim@ueransim:~$ man man
```

```
ueransim@ueransim: ~
MAN(1) Manual pager utils MAN(1)

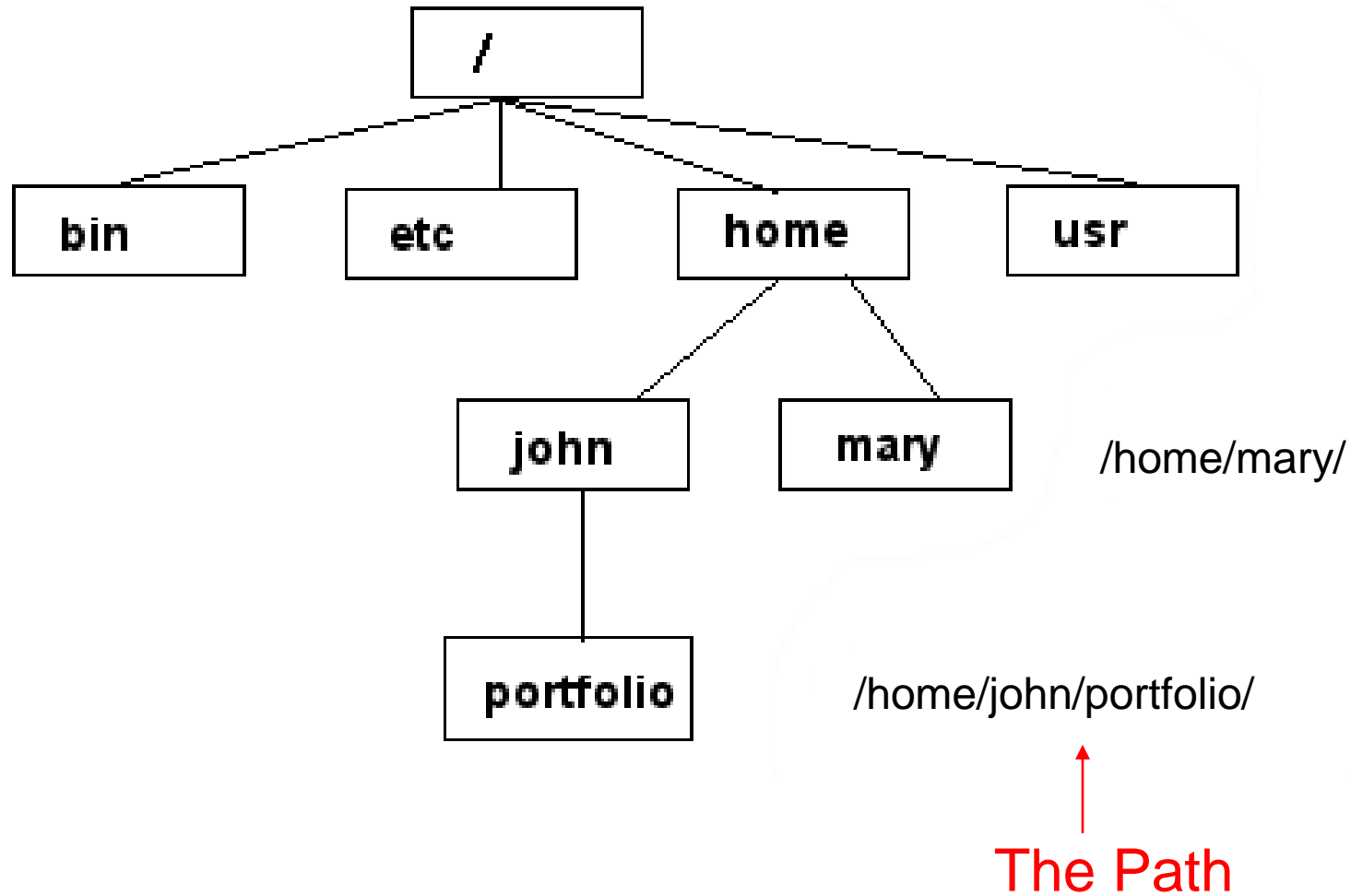
NAME
    man - an interface to the system reference manuals

SYNOPSIS
    man [man options] [[section] page ...] ...
    man -k [apropos options] regexp ...
    man -K [man options] [section] term ...
    man -f [whatis options] page ...
    man -l [man options] file ...
    man -w|-W [man options] page ...

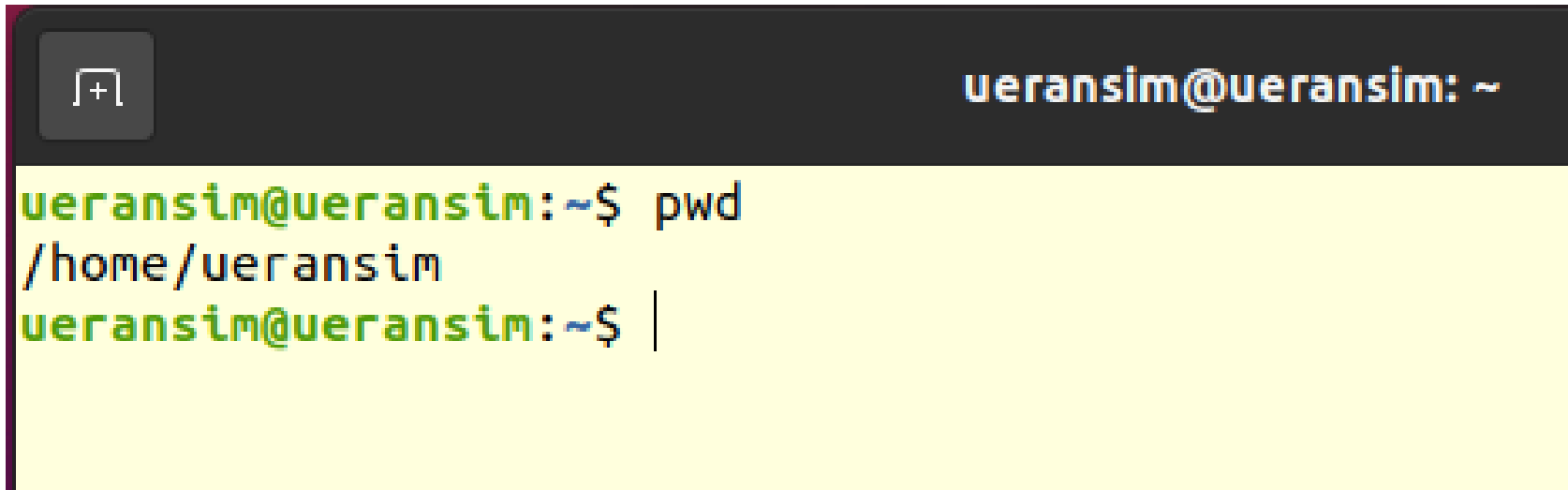
DESCRIPTION
    man is the system's manual pager. Each page argument given to man is normally the name of a program, utility or function. The manual page associated with each of these arguments is then found and displayed. A section, if provided, will direct man to look only in that section of the manual. The default action is to search in all of the available sections following a pre-defined order (see DEFAULTS), and to show only the first page found, even if page exists in several sec-tions.

Manual page man(1) line 1 (press h for help or q to quit)
```

Linux filesystem



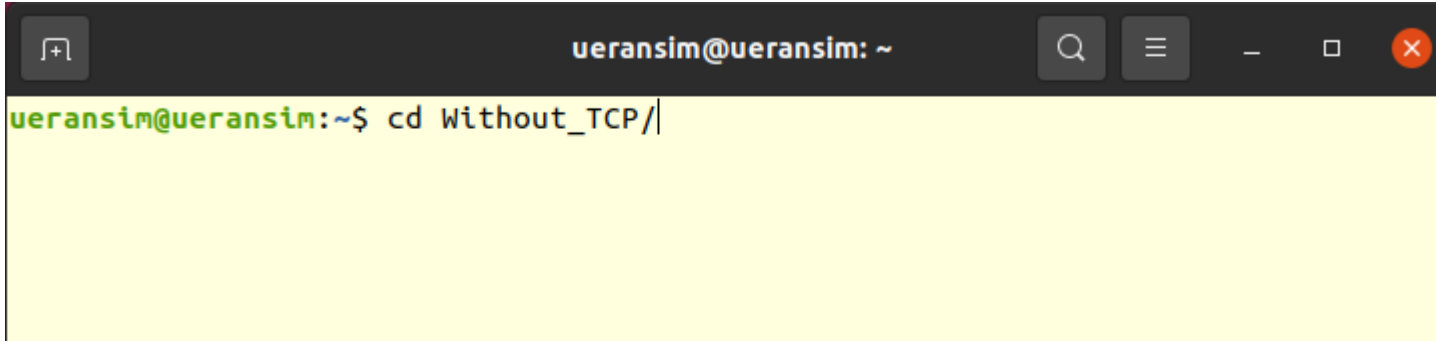
If I don't know the path: Command *pwd*

A terminal window with a dark background. The title bar shows a window icon and the text "ueransim@ueransim: ~". The terminal content shows the command "pwd" being executed, resulting in the output "/home/ueransim". The prompt "ueransim@ueransim:~\$" is visible before and after the command.

```
ueransim@ueransim:~$ pwd
/home/ueransim
ueransim@ueransim:~$ |
```

pwd: Present working directory

Command: *cd*

A terminal window with a dark title bar. The title bar contains a window icon on the left, the text 'ueransim@ueransim: ~' in the center, and search, menu, and window control icons on the right. The terminal area has a yellow background. The prompt 'ueransim@ueransim:~\$' is followed by the command 'cd Without_TCP/' and a cursor.

```
ueransim@ueransim:~$ cd Without_TCP/|
```

cd: change directory

Can I jump to another directory without going to the home directory?

Way 1

```
ueransim@ueransim: ~/Without_UDP
ueransim@ueransim:~/Without_TCP$ cd /home/ueransim/Without_UDP/
ueransim@ueransim:~/Without_UDP$
```

Need to give the absolute path

Way 2

```
ueransim@ueransim: ~/Without_UDP
ueransim@ueransim:~/Without_TCP$ cd ~/Without_UDP/
ueransim@ueransim:~/Without_UDP$
```

~ : location of the home directory

How to list contents of a directory

A terminal window titled 'ueransim@ueransim: ~' with standard window controls. The command 'ls' has been executed, displaying a list of files and directories in a color-coded format. The output is as follows:

```
ueransim@ueransim:~$ ls
Desktop  IDS_NAT  packet_size  snap  Videos
Documents  IPS_NAT  Pictures      Templates  Without_TCP
Downloads  Music    Public        UERANSIM  Without_UDP
ueransim@ueransim:~$
```

`ls`: lists contents of the directory

Can I see the contents of another directory?

```
ueransim@ueransim: ~  
ueransim@ueransim:~$ ls  
Desktop  IDS_NAT  packet_size  snap  Videos  
Documents  IPS_NAT  Pictures  Templates  Without_TCP  
Downloads  Music  Public  UERANSIM  Without_UDP  
ueransim@ueransim:~$
```

Contents in
current
directory

```
ueransim@ueransim: ~  
ueransim@ueransim:~$ ls Without_TCP/  
output.txt  script_ditg_decode.py  Without_NAT_TCP.csv  
ueransim@ueransim:~$
```

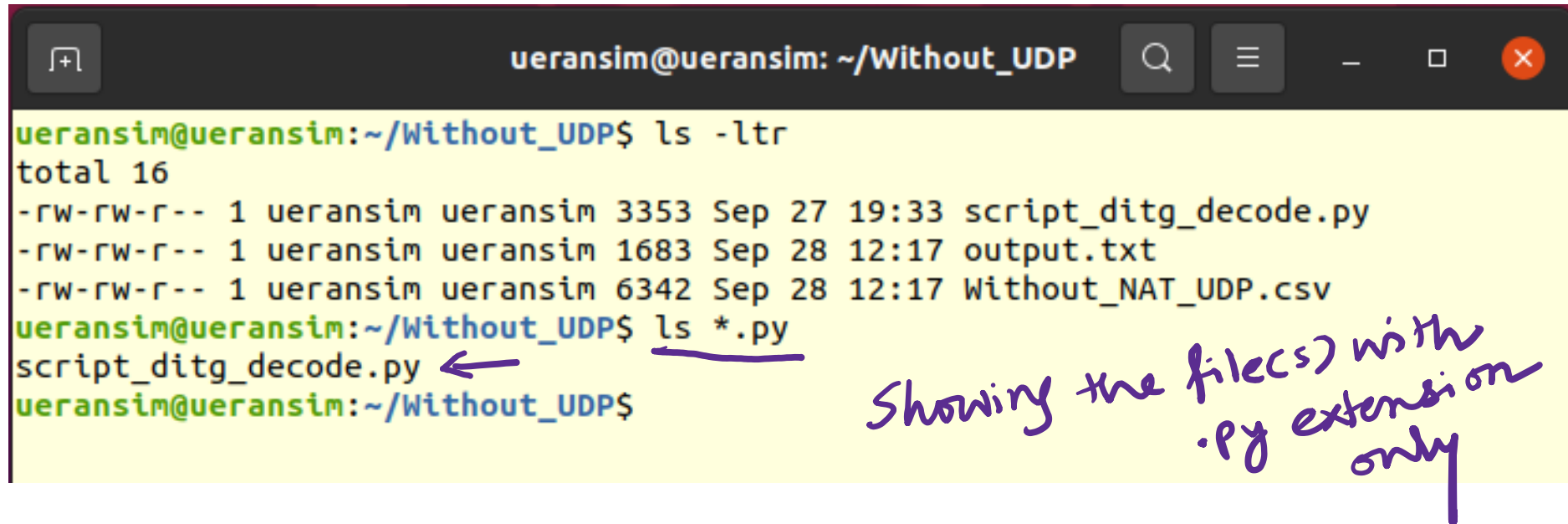
Contents in
another
directory

Can I see all files with `ls` command?

- `ls` has many options
 - `-l` long list (displays lots of info)
 - `-t` sort by modification time
 - `-S` sort by size
 - `-h` list file sizes in human-readable format
 - `-r` reverses the order
- Options can be combined: `ls -ltr`

```
ueransim@ueransim: ~/Without_UDP
ueransim@ueransim:~/Without_UDP$ ls -ltr
total 16
-rw-rw-r-- 1 ueransim ueransim 3353 Sep 27 19:33 script_ditg_decode.py
-rw-rw-r-- 1 ueransim ueransim 1683 Sep 28 12:17 output.txt
-rw-rw-r-- 1 ueransim ueransim 6342 Sep 28 12:17 Without_NAT_UDP.csv
ueransim@ueransim:~/Without_UDP$ |
```


If I only want to see files with a specific extension



```
ueransim@ueransim: ~/Without_UDP
ueransim@ueransim:~/Without_UDP$ ls -ltr
total 16
-rw-rw-r-- 1 ueransim ueransim 3353 Sep 27 19:33 script_ditg_decode.py
-rw-rw-r-- 1 ueransim ueransim 1683 Sep 28 12:17 output.txt
-rw-rw-r-- 1 ueransim ueransim 6342 Sep 28 12:17 Without_NAT_UDP.csv
ueransim@ueransim:~/Without_UDP$ ls *.py
script_ditg_decode.py ←
ueransim@ueransim:~/Without_UDP$
```

Showing the file(s) with .py extension only

How to create a new directory



The image shows two terminal windows. The top window shows the command `mkdir Linux_commands` being entered. The bottom window shows the same command followed by `ls`, which lists the contents of the home directory. The newly created directory `Linux_commands` is visible in the list and is underlined with a purple line. A purple arrow points to this underlined text.

```
ueransim@ueransim: ~  
ueransim@ueransim:~$ mkdir Linux_commands  
  
ueransim@ueransim:~$ mkdir Linux_commands  
ueransim@ueransim:~$ ls  
Desktop    IDS_NAT      Music        Public       UERANSIM     Without_UDP  
Documents  IPS_NAT      packet_size  snap         Videos  
Downloads  Linux_commands Pictures      Templates    Without_TCP  
ueransim@ueransim:~$
```

mkdir: make directory

How to delete a directory?

Way 1

```
ueransim@ueransim: ~  
ueransim@ueransim:~$ rmdir Linux_commands/
```

```
ueransim@ueransim:~$ rmdir Linux_commands/  
ueransim@ueransim:~$ ls  
Desktop  IDS_NAT  packet_size  snap  Videos  
Documents  IPS_NAT  Pictures  Templates  Without_TCP  
Downloads  Music  Public  UERANSIM  Without_UDP  
ueransim@ueransim:~$
```

-r or -R:

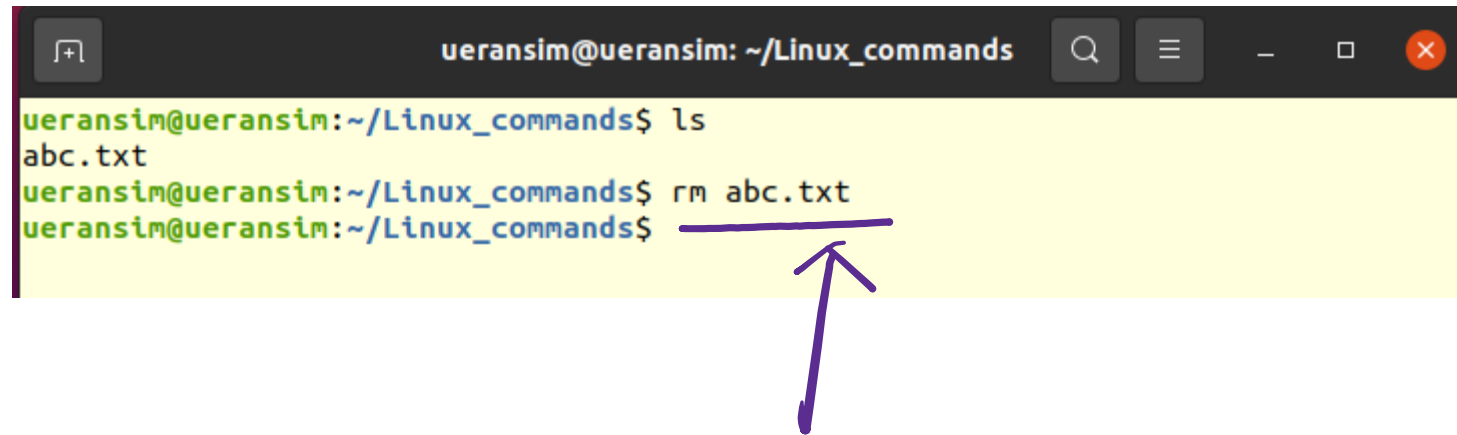
Recursively
deletes any matching
directories,
Subdirectories
and files they
contain

Way 2

```
ueransim@ueransim: ~  
ueransim@ueransim:~$ rm -r Linux_commands/  
ueransim@ueransim:~$
```

rm: remove

How to delete a file?

A terminal window titled 'ueransim@ueransim: ~/Linux_commands' with standard window controls. The terminal shows a sequence of commands: 'ls' followed by 'abc.txt', then 'rm abc.txt'. A purple arrow points to the 'rm abc.txt' command line.

```
ueransim@ueransim:~/Linux_commands$ ls
abc.txt
ueransim@ueransim:~/Linux_commands$ rm abc.txt
ueransim@ueransim:~/Linux_commands$
```

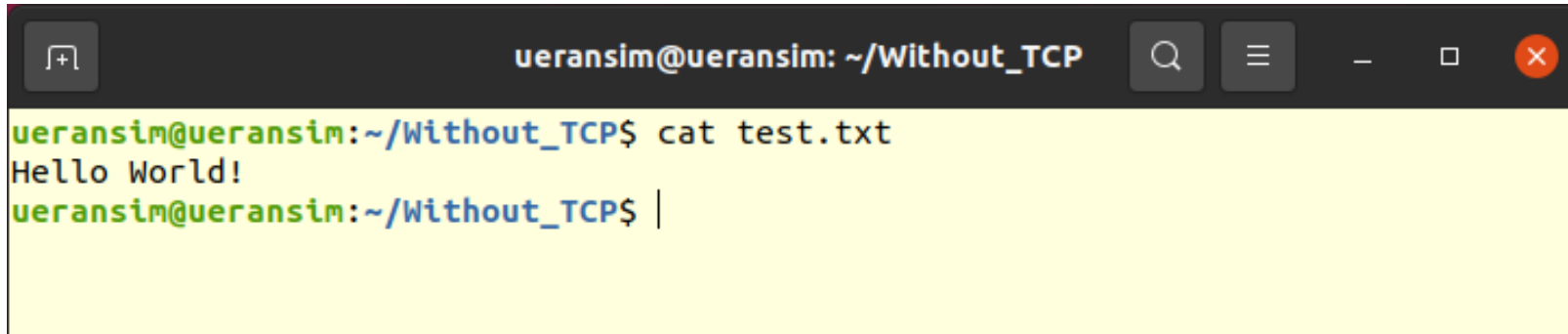
How to delete all files with a specific extension?

How to display contents of a file?

- cat
- less
- head
- tail

cat

- Dumps an entire file to standard output
- Good for displaying short, simple files

A terminal window with a dark title bar. The title bar contains a window icon, the text 'ueransim@ueransim: ~/Without_TCP', and standard window controls (search, menu, zoom, close). The terminal area has a yellow background. It shows the command 'ueransim@ueransim:~/Without_TCP\$ cat test.txt' being executed, followed by the output 'Hello World!'. A new prompt 'ueransim@ueransim:~/Without_TCP\$ |' is shown on the next line.

```
ueransim@ueransim: ~/Without_TCP
ueransim@ueransim:~/Without_TCP$ cat test.txt
Hello World!
ueransim@ueransim:~/Without_TCP$ |
```

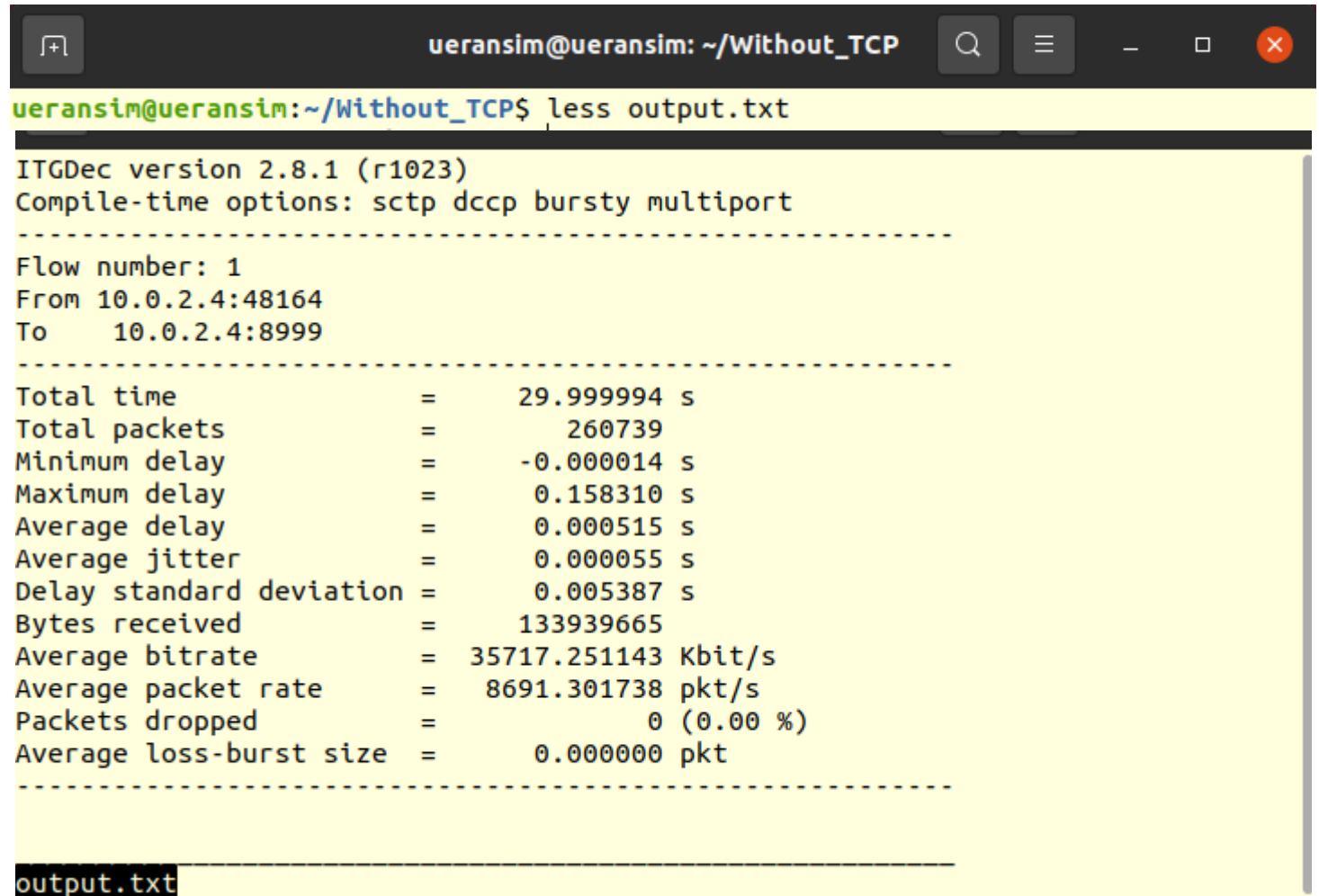
But if we have a file with large contents!

```
ueransim@ueransim: ~/Without_TCP
Packets dropped          =          0 (0.00 %)
Average loss-burst size  =          0.000000 pkt
-----
***** TOTAL RESULTS *****
-----
Number of flows          =           1
Total time               =      29.999994 s
Total packets            =      260739
Minimum delay            =     -0.000014 s
Maximum delay            =       0.158310 s
Average delay            =       0.000515 s
Average jitter           =       0.000055 s
Delay standard deviation =       0.005387 s
Bytes received           =     133939665
Average bitrate          =    35717.251143 Kbit/s
Average packet rate      =     8691.301738 pkt/s
Packets dropped          =          0 (0.00 %)
Average loss-burst size  =          0 pkt
Error lines              =          0
-----
ueransim@ueransim:~/Without_TCP$
```

Cannot see all content!

Command: *less*

- *less* displays a file, allowing forward/backward movement within it
- *return* scrolls forward one line
- *space* one page
- *y* scrolls back one line
- *b* one page
- use */* to search for a string
- Press *q* to quit



```
ueransim@ueransim: ~/Without_TCP
ueransim@ueransim:~/Without_TCP$ less output.txt
ITGDec version 2.8.1 (r1023)
Compile-time options: sctp dccp bursty multiport
-----
Flow number: 1
From 10.0.2.4:48164
To   10.0.2.4:8999
-----
Total time           =      29.999994 s
Total packets        =      260739
Minimum delay        =     -0.000014 s
Maximum delay        =      0.158310 s
Average delay        =      0.000515 s
Average jitter       =      0.000055 s
Delay standard deviation =    0.005387 s
Bytes received       =    133939665
Average bitrate      =   35717.251143 Kbit/s
Average packet rate  =    8691.301738 pkt/s
Packets dropped      =              0 (0.00 %)
Average loss-burst size =    0.000000 pkt
-----
output.txt
```

Command: *head*

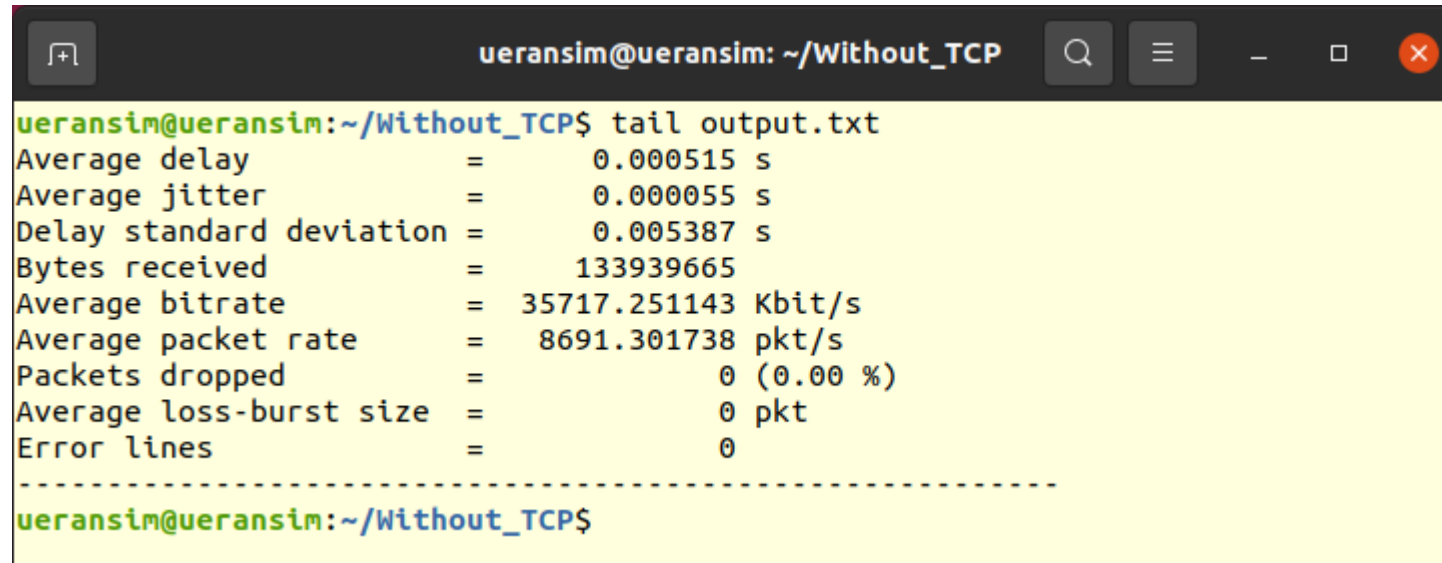
- *head* displays the top part of a file
- By default it shows the first 10 lines
- *-n* option allows you to change the number of lines to display
- Example: *head -n50 file.txt* displays the first 50 lines of file.txt

```
ueransim@ueransim:~/Without_TCP$ head output.txt
ITGDec version 2.8.1 (r1023)
Compile-time options: sctp dccp bursty multiport
-----
Flow number: 1
From 10.0.2.4:48164
To    10.0.2.4:8999
-----
Total time           =      29.999994 s
Total packets        =      260739
Minimum delay        =     -0.000014 s
ueransim@ueransim:~/Without_TCP$
```

```
ueransim@ueransim: ~/Without_TCP
ueransim@ueransim:~/Without_TCP$ head -n15 output.txt
ITGDec version 2.8.1 (r1023)
Compile-time options: sctp dccp bursty multiport
-----
Flow number: 1
From 10.0.2.4:48164
To    10.0.2.4:8999
-----
Total time           =      29.999994 s
Total packets        =      260739
Minimum delay        =     -0.000014 s
Maximum delay        =      0.158310 s
Average delay        =      0.000515 s
Average jitter       =      0.000055 s
Delay standard deviation =    0.005387 s
Bytes received       =    133939665
ueransim@ueransim:~/Without_TCP$ |
```


Command: *tail*

- Same as *head* but shows the *last 10 lines by default*

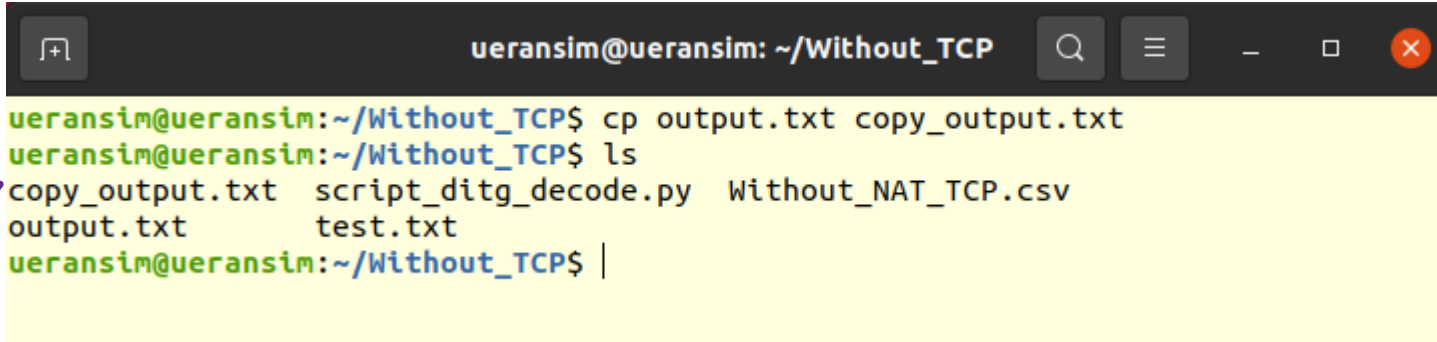
A terminal window titled 'ueransim@ueransim: ~/Without_TCP' with standard window controls. The prompt is 'ueransim@ueransim:~/Without_TCP\$'. The command 'tail output.txt' has been executed, resulting in a block of network statistics. The statistics include average delay, jitter, standard deviation, bytes received, bitrate, packet rate, packets dropped, loss-burst size, and error lines. A dashed line separates the statistics from the next prompt.

```
ueransim@ueransim:~/Without_TCP$ tail output.txt
Average delay          =      0.000515 s
Average jitter         =      0.000055 s
Delay standard deviation =    0.005387 s
Bytes received         =    133939665
Average bitrate        =  35717.251143 Kbit/s
Average packet rate    =   8691.301738 pkt/s
Packets dropped        =              0 (0.00 %)
Average loss-burst size =              0 pkt
Error lines            =              0
-----
ueransim@ueransim:~/Without_TCP$
```

- *tail -n 12 test.txt* shows last 12 lines

File commands: copy, move, rename

- Copying a file: cp



```
ueransim@ueransim: ~/Without_TCP
ueransim@ueransim:~/Without_TCP$ cp output.txt copy_output.txt
ueransim@ueransim:~/Without_TCP$ ls
copy_output.txt  script_ditg_decode.py  Without_NAT_TCP.csv
output.txt       test.txt
```

A purple arrow points from the `copy_output.txt` file in the terminal output to the handwritten text below.

Contents of both files are the same.

File commands: copy, move, rename

- Move or rename a file: mv

```
ueransim@ueransim: ~/Without_TCP
ueransim@ueransim:~/Without_TCP$ mv copy_output.txt ~/
ueransim@ueransim:~/Without_TCP$ |
```

→ move to another directory

```
ueransim@ueransim: ~/Without_TCP
ueransim@ueransim:~/Without_TCP$ mv output.txt output_new.txt
ueransim@ueransim:~/Without_TCP$ ls
output_new.txt  script_ditg_decode.py  test.txt  Without_NAT_TCP.csv
ueransim@ueransim:~/Without_TCP$
```

→ Rename a file

File permissions

```
ueransim@ueransim: ~/Without_TCP
ueransim@ueransim:~/Without_TCP$ ls -l
total 20
-rw-rw-r-- 1 ueransim ueransim 1683 Sep 28 20:43 output_new.txt
-rw-rw-r-- 1 ueransim ueransim 3353 Sep 27 20:21 script_ditg_decode.py
-rw-rw-r-- 1 ueransim ueransim  13 Oct  5 12:10 test.txt
-rw-rw-r-- 1 ueransim ueransim 6316 Sep 28 20:43 Without_NAT_TCP.csv
ueransim@ueransim:~/Without_TCP$ |
```

r - Read
w - Write

total size in KB

Question: What is the size of an empty directory?



ueransim@ueransim: ~/Without_TCP

```
ueransim@ueransim:~/Without_TCP$ ls -as1
```

```
total 28
```

```
4 .
```

```
4 ..
```

```
4 output_new.txt
```

```
4 script_ditg_decode.py
```

```
4 test.txt
```

```
8 Without_NAT_TCP.csv
```

```
ueransim@ueransim:~/Without_TCP$ |
```



ueransim@ueransim: ~/Without_TCP

```
ueransim@ueransim:~/Without_TCP$ ls -as1
```

```
total 28
```

```
4 .  
4 ..
```

What are these?

```
4 output_new.txt
```

```
4 script_ditg_decode.py
```

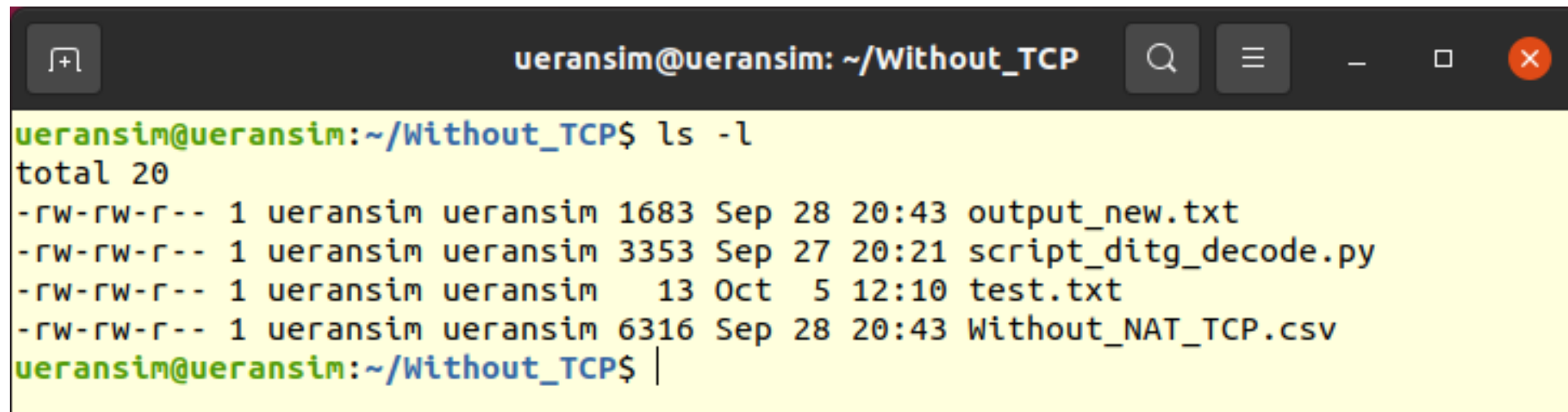
```
4 test.txt
```

```
8 Without_NAT_TCP.csv
```

```
ueransim@ueransim:~/Without_TCP$ |
```

File permissions

- Read: r
- Write: w
- Execute: x
 - In case of directory, **x** grants permission to list directory contents

A terminal window titled 'ueransim@ueransim: ~/Without_TCP' with standard window controls. The terminal shows the command 'ls -l' and its output, which lists four files with their permissions, owner, group, size, date, and name. The permissions for all files are '-rw-rw-r--'.

```
ueransim@ueransim: ~/Without_TCP
ueransim@ueransim:~/Without_TCP$ ls -l
total 20
-rw-rw-r-- 1 ueransim ueransim 1683 Sep 28 20:43 output_new.txt
-rw-rw-r-- 1 ueransim ueransim 3353 Sep 27 20:21 script_ditg_decode.py
-rw-rw-r-- 1 ueransim ueransim  13 Oct  5 12:10 test.txt
-rw-rw-r-- 1 ueransim ueransim 6316 Sep 28 20:43 Without_NAT_TCP.csv
ueransim@ueransim:~/Without_TCP$ |
```

```
ueransim@ueransim: ~  
ueransim@ueransim:~$ ls -l  
total 68  
-rw-rw-r-- 1 ueransim ueransim 1683 Oct  5 13:55 copy_output.txt  
drwxr-xr-x 2 ueransim ueransim 4096 May 29 09:44 Desktop  
drwxr-xr-x 2 ueransim ueransim 4096 May 29 09:44 Documents  
drwxr-xr-x 2 ueransim ueransim 4096 Sep 28 20:44 Downloads  
drwxrwxr-x 2 ueransim ueransim 4096 Sep 27 10:41 IDS_NAT  
drwxrwxr-x 2 ueransim ueransim 4096 Sep 30 21:50 IPS_NAT  
drwxrwxr-x 2 ueransim ueransim 4096 Oct  5 12:05 Linux_commands  
drwxr-xr-x 2 ueransim ueransim 4096 May 29 09:44 Music  
drwxrwxr-x 2 ueransim ueransim 4096 Oct  1 01:03 packet_size  
drwxr-xr-x 2 ueransim ueransim 4096 May 29 09:44 Pictures  
drwxr-xr-x 2 ueransim ueransim 4096 May 29 09:44 Public  
drwx----- 3 ueransim ueransim 4096 May 29 09:54 snap  
drwxr-xr-x 2 ueransim ueransim 4096 May 29 09:44 Templates  
drwxrwxr-x 9 ueransim ueransim 4096 May 29 09:54 UERANSIM  
drwxr-xr-x 2 ueransim ueransim 4096 May 29 09:44 Videos  
drwxrwxr-x 2 ueransim ueransim 4096 Oct  5 13:58 Without_TCP  
drwxrwxr-x 2 ueransim ueransim 4096 Sep 30 20:51 Without_UDP  
ueransim@ueransim:~$
```

The first letter indicates the type of file:

- - means it's a normal file
- d means it's a directory
- l means it's a link

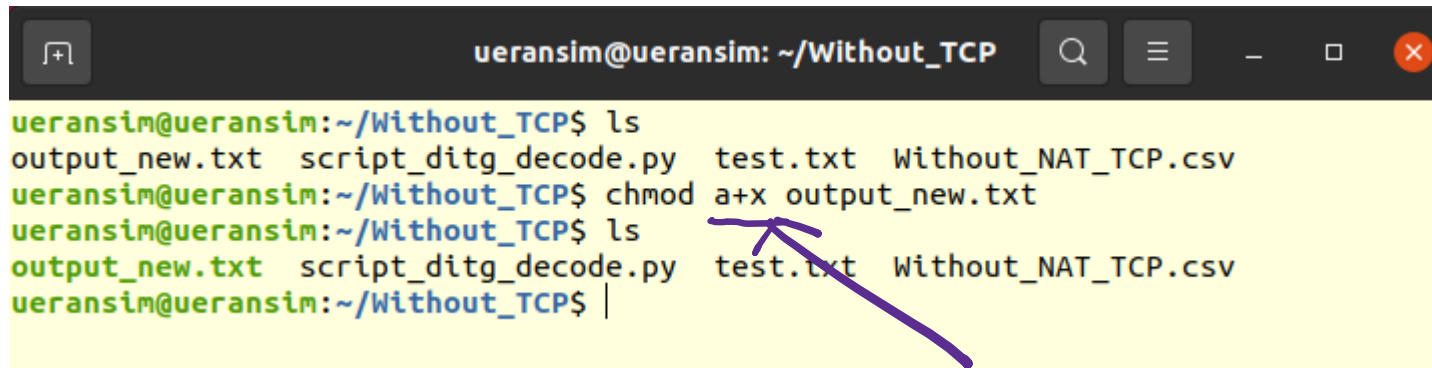

```
ueransim@ueransim:~/Without_TCP$ ls -l
total 20
-rw-rw-r-- 1 ueransim ueransim 1683 Sep 28 20:43 output_new.txt
-rw-rw-r-- 1 ueransim ueransim 3353 Sep 27 20:21 script_ditg_decode.py
-rw-rw-r-- 1 ueransim ueransim  13 Oct  5 12:10 test.txt
-rw-rw-r-- 1 ueransim ueransim 6316 Sep 28 20:43 Without_NAT_TCP.csv
ueransim@ueransim:~/Without_TCP$ |
```

owner group world.

if we have executable file: `rx`

Can we change file permission?

- Using *chmod*
 - Syntax: `chmod [user/group/others/all]+[permission] [file(s)]`



A terminal window titled 'ueransim@ueransim: ~/Without_TCP' showing a sequence of commands and their output. The first command is 'ls', which lists files: 'output_new.txt', 'script_ditg_decode.py', 'test.txt', and 'Without_NAT_TCP.csv'. The second command is 'chmod a+x output_new.txt'. The third command is another 'ls', which shows the same files, but 'output_new.txt' is now green, indicating it is executable. A purple arrow points from the 'a+x' in the 'chmod' command to the green 'output_new.txt' in the second 'ls' output.

```
ueransim@ueransim:~/Without_TCP$ ls
output_new.txt  script_ditg_decode.py  test.txt  Without_NAT_TCP.csv
ueransim@ueransim:~/Without_TCP$ chmod a+x output_new.txt
ueransim@ueransim:~/Without_TCP$ ls
output_new.txt  script_ditg_decode.py  test.txt  Without_NAT_TCP.csv
ueransim@ueransim:~/Without_TCP$ |
```

observe: changes in color,

: What did we do with a+x?

```
ueransim@ueransim: ~/Without_TCP
ueransim@ueransim:~/Without_TCP$ ls
output_new.txt  script_ditg_decode.py  test.txt  Without_NAT_TCP.csv
ueransim@ueransim:~/Without_TCP$ chmod a+x output_new.txt
ueransim@ueransim:~/Without_TCP$ ls
output_new.txt  script_ditg_decode.py  test.txt  Without_NAT_TCP.csv
ueransim@ueransim:~/Without_TCP$ |
```

- a stands for *all*
- u stands for *user*
- g stands for *group*
- o stands for *others*

+ : add permission ; - : remove permission

r, w, x : read, write, execute

Can we change the permission in another way?

- Using numbers:

- 1 if has execution permission
- 2 if has write permission
- 4 if has read permission

⇒ So, total number value can be 7

0 no permissions

1 can execute

2 can write

3 can write, execute → (2+1)

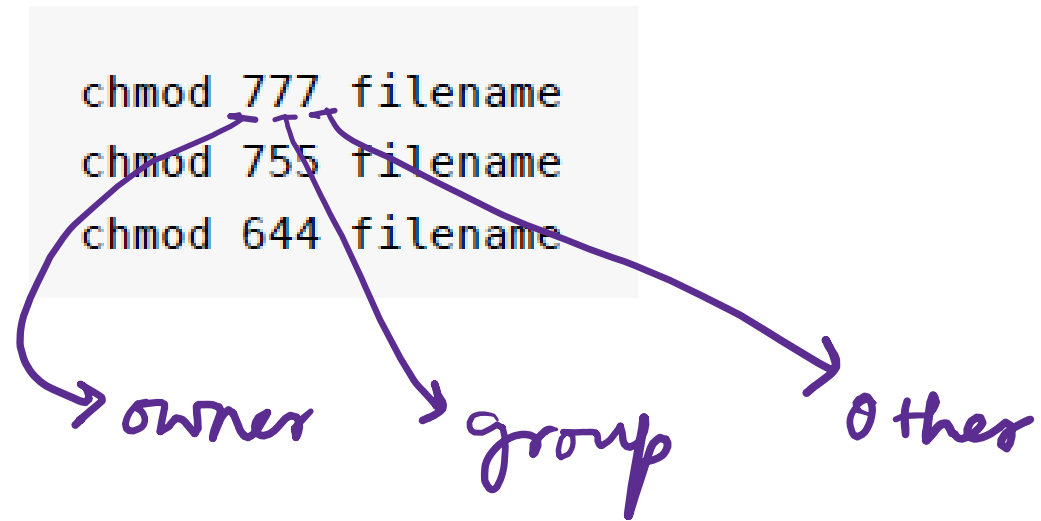
4 can read

5 can read, execute → (4+1)

6 can read, write → (4+2)

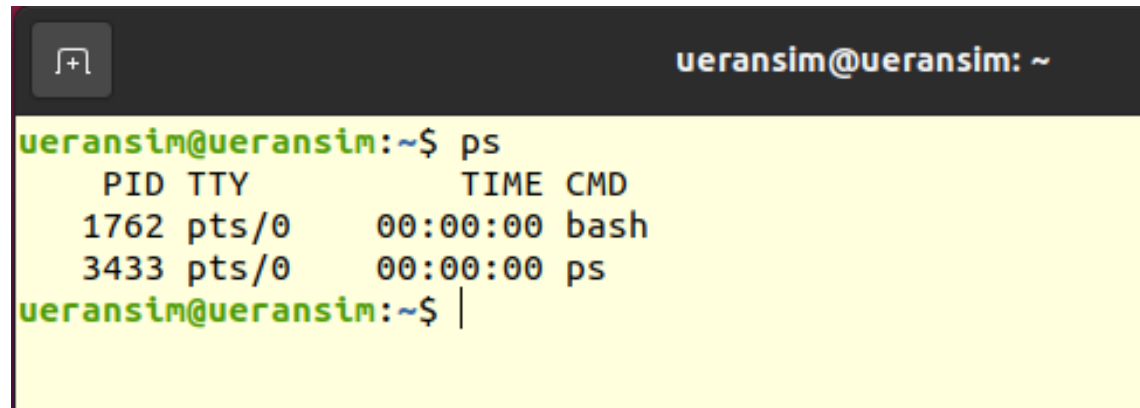
7 can read, write and execute → (4+2+1)

What about the users?



Command: *ps*

- List of user-initiated processes



A terminal window titled 'ueransim@ueransim: ~' showing the output of the 'ps' command. The output is a table with columns: PID, TTY, TIME, and CMD. It lists two processes: 'bash' with PID 1762 and 'ps' with PID 3433. The prompt 'ueransim@ueransim:~\$' is shown before and after the output.

```
ueransim@ueransim:~$ ps
  PID TTY          TIME CMD
 1762 pts/0    00:00:00 bash
 3433 pts/0    00:00:00 ps
ueransim@ueransim:~$ |
```

Command: *ps ax*

- To list all processes

```
ueransim@ueransim: ~$ ps ax
```

PID	TTY	STAT	TIME	COMMAND
1	?	Ss	0:01	/sbin/init splash
2	?	S	0:00	[kthreadd]
3	?	I<	0:00	[rcu_gp]
4	?	I<	0:00	[rcu_par_gp]
5	?	I<	0:00	[slub_flushwq]
6	?	I<	0:00	[netns]
8	?	I<	0:00	[kworker/0:0H-events_highpri]
10	?	I<	0:00	[mm_percpu_wq]
11	?	S	0:00	[rcu_tasks_rude_]
12	?	S	0:00	[rcu_tasks_trace]
13	?	S	0:00	[ksoftirqd/0]
14	?	I	0:00	[rcu_sched]
15	?	S	0:00	[migration/0]
16	?	S	0:00	[idle_inject/0]
18	?	S	0:00	[cpuhp/0]

```

1220 ?      S      0:00 /usr/bin/pulseaudio --daemonize=no --log-target=jo
1225 ?      S<sl  0:02 /usr/bin/pulseaudio --daemonize=no --log-target=jo
1227 ?      SNsl  0:00 /usr/libexec/tracker-miner-fs
1229 ?      Ss    0:00 /usr/bin/dbus-daemon --session --address=systemd:
1246 ?      Ssl  0:00 /usr/libexec/gvfsd
1248 ?      Sl    0:00 /usr/bin/gnome-keyring-daemon --daemonize --login
1255 ?      Sl    0:00 /usr/libexec/gvfsd-fuse /run/user/1000/gvfs -f -o
1257 ?      Ssl  0:00 /usr/libexec/gvfs-udisks2-volume-monitor
1270 tty2    Ssl+  0:00 /usr/lib/gdm3/gdm-x-session --run-script env GNOME
1272 tty2    Sl+   0:21 /usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/use
1273 ?      Ssl  0:00 /usr/libexec/gvfs-afc-volume-monitor
1280 ?      Ssl  0:00 /usr/libexec/gvfs-goa-volume-monitor
1284 ?      Sl    0:00 /usr/libexec/goa-daemon
1292 ?      Sl    0:00 /usr/libexec/goa-identity-service
1299 ?      Ssl  0:00 /usr/libexec/gvfs-mtp-volume-monitor
1303 ?      Ssl  0:00 /usr/libexec/gvfs-gphoto2-volume-monitor
1320 tty2    Sl+   0:00 /usr/libexec/gnome-session-binary --systemd --syst
1390 ?      Ssl  0:00 /usr/libexec/gvfsd-metadata

```

*longer names
are cut*



To show all processes with full name

```
ueransim@ueransim: ~  
ueransim@ueransim:~$ ps axww
```

PPID	PID	USER	PR	NI	PM	PS	TTY	TIME	CMD
1649	?	SL	0	0	0	0:00			/usr/libexec/gsd-disk-utility-notify
1650	?	SL	0	0	0	0:00			/usr/libexec/evolution-data-server/evolution-alarm-notify
1709	?	SL	0	0	0	0:00			/usr/libexec/gsd-printer

← shown in detail (no cut)

`ps axww | grep "Visual Studio Code"` → to search running process with phrase "visual --"

Command: *top*

- Shows the dynamic real-time information about the running processes

```
ueransim@ueransim: ~  
top - 14:49:57 up 3:50, 1 user, load average: 0.00, 0.03, 0.00  
Tasks: 176 total, 1 running, 175 sleeping, 0 stopped, 0 zombie  
%Cpu(s): 0.3 us, 0.7 sy, 0.0 ni, 98.6 id, 0.2 wa, 0.0 hi, 0.2 si, 0.0 st  
MiB Mem : 1963.9 total, 70.9 free, 746.0 used, 1147.0 buff/cache  
MiB Swap: 1162.4 total, 1161.4 free, 1.0 used. 1036.1 avail Mem  
  
  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM    TIME+  COMMAND  
 1272 ueransim  20   0  278156  88672  48916 S   1.7   4.4   0:24.14 Xorg  
 1458 ueransim  20   0 4219396 361444 125872 S   1.7  18.0   0:31.95 gnome-+  
 1229 ueransim  20   0   13688  10452   3776 S   0.3   0.5   0:00.57 dbus-d+  
 3431 root      20   0        0        0        0 I   0.3   0.0   0:00.20 kworke+  
    1 root      20   0  168320  11544   8376 S   0.0   0.6   0:01.17 systemd  
    2 root      20   0        0        0        0 S   0.0   0.0   0:00.00 kthrea+  
    3 root       0 -20        0        0        0 I   0.0   0.0   0:00.00 rcu_gp  
    4 root       0 -20        0        0        0 I   0.0   0.0   0:00.00 rcu_pa+  
    5 root       0 -20        0        0        0 I   0.0   0.0   0:00.00 slub_f+  
    6 root       0 -20        0        0        0 I   0.0   0.0   0:00.00 netns  
    8 root       0 -20        0        0        0 I   0.0   0.0   0:00.00 kworke+  
   10 root       0 -20        0        0        0 I   0.0   0.0   0:00.00 mm_per+
```

How to kill a process?

- *kill <pid>*

```
kill -HUP <PID>  
kill -INT <PID>  
kill -KILL <PID>  
kill -TERM <PID>  
kill -CONT <PID>  
kill -STOP <PID>
```

with flags

HUP \Rightarrow hung up

INT \Rightarrow interrupt

KILL \Rightarrow to kill, send to os kernel (NOT ^{to} process)

TERM \Rightarrow terminate

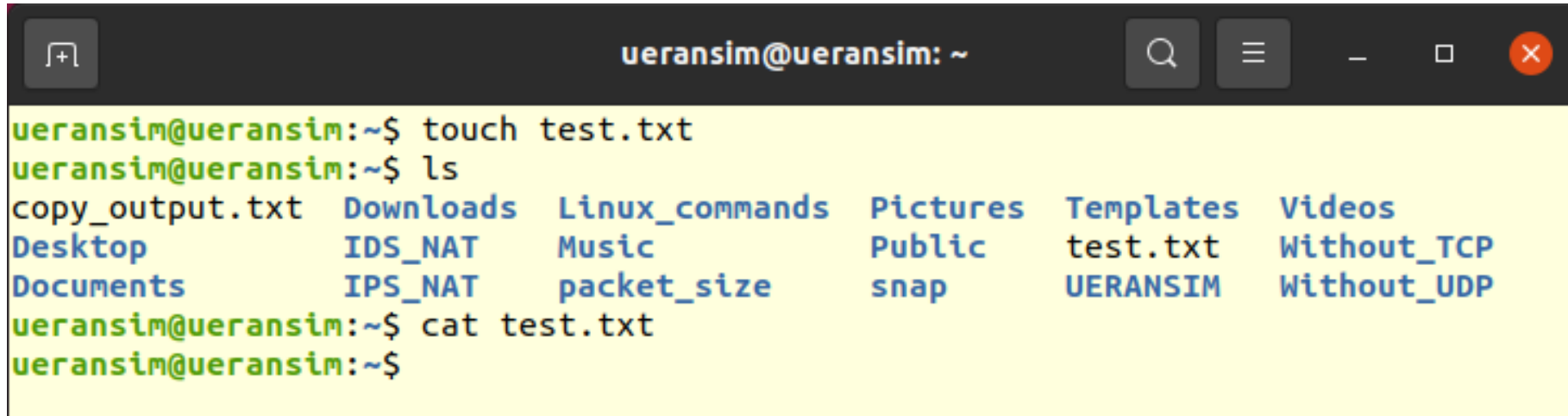
CONT \Rightarrow to resume a stopped process

STOP \Rightarrow stops (but not terminate)
- send to os kernel (NOT to a process)

- Can also use numbers:

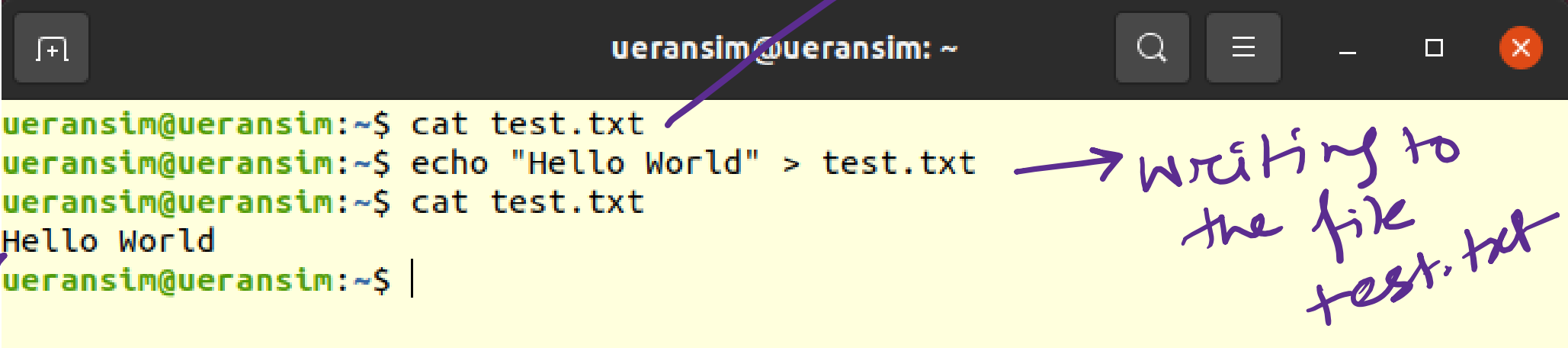
1 corresponds to HUP . 2 corresponds to INT . 9
corresponds to KILL . 15 corresponds to TERM .
18 corresponds to CONT . 15 corresponds to
STOP .

Create a new file using *touch*



```
ueransim@ueransim: ~  
ueransim@ueransim:~$ touch test.txt  
ueransim@ueransim:~$ ls  
copy_output.txt  Downloads  Linux_commands  Pictures  Templates  Videos  
Desktop          IDS_NAT    Music           Public    test.txt   Without_TCP  
Documents        IPS_NAT    packet_size     snap     UERANSIM   Without_UDP  
ueransim@ueransim:~$ cat test.txt  
ueransim@ueransim:~$
```

Write something into the file

A terminal window with a dark title bar containing the text 'ueransim@ueransim: ~' and standard window controls. The terminal has a yellow background and shows the following commands and output:

```
ueransim@ueransim:~$ cat test.txt
ueransim@ueransim:~$ echo "Hello World" > test.txt
ueransim@ueransim:~$ cat test.txt
Hello World
ueransim@ueransim:~$ |
```

Written to
the file

empty

→ writing to
the file
test.txt

Write your own shell script

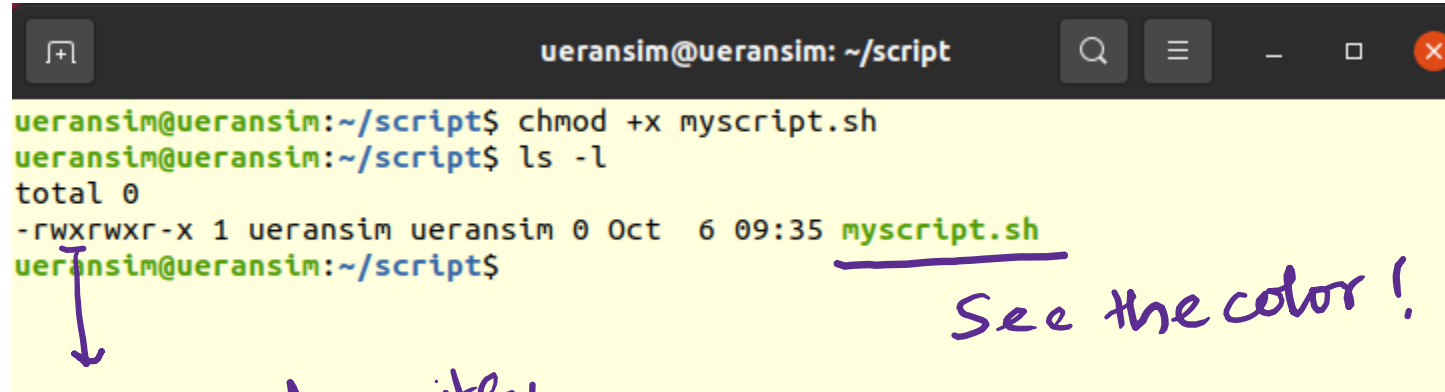
Create a file with *sh* extension

```
ueransim@ueransim: ~/script
ueransim@ueransim:~/script$ ls
ueransim@ueransim:~/script$ touch myscript.sh
ueransim@ueransim:~/script$ ls
myscript.sh
ueransim@ueransim:~/script$ cat myscript.sh
ueransim@ueransim:~/script$ ls -l
total 0
-rw-rw-r-- 1 ueransim ueransim 0 Oct  6 09:35 myscript.sh
ueransim@ueransim:~/script$ |
```

→ empty file

→ only read and write
NOT executable

Make *myscript.sh* executable

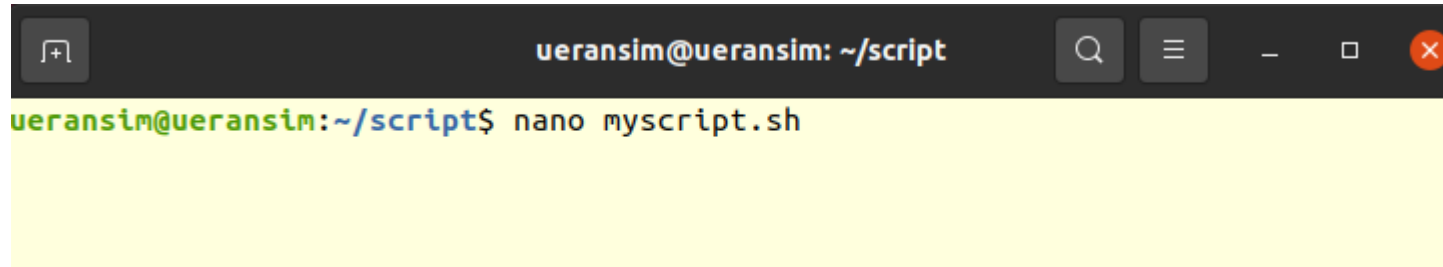


```
ueransim@ueransim: ~/script
ueransim@ueransim:~/script$ chmod +x myscript.sh
ueransim@ueransim:~/script$ ls -l
total 0
-rwxrwxr-x 1 ueransim ueransim 0 Oct  6 09:35 myscript.sh
ueransim@ueransim:~/script$
```

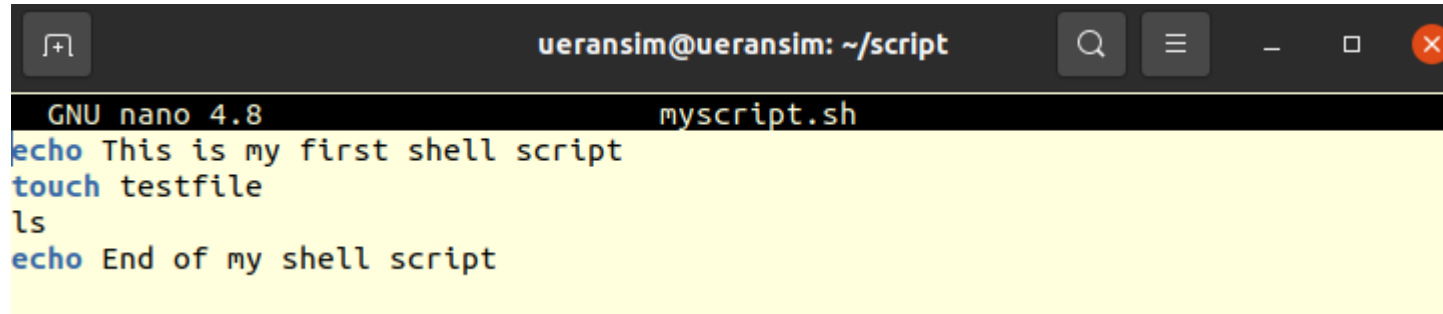
See the color!

Now, read, write,
execute
permissions

Adding some commands to *myscript.sh*



```
ueransim@ueransim: ~/script
ueransim@ueransim:~/script$ nano myscript.sh
```



```
GNU nano 4.8 myscript.sh
echo This is my first shell script
touch testfile
ls
echo End of my shell script
```

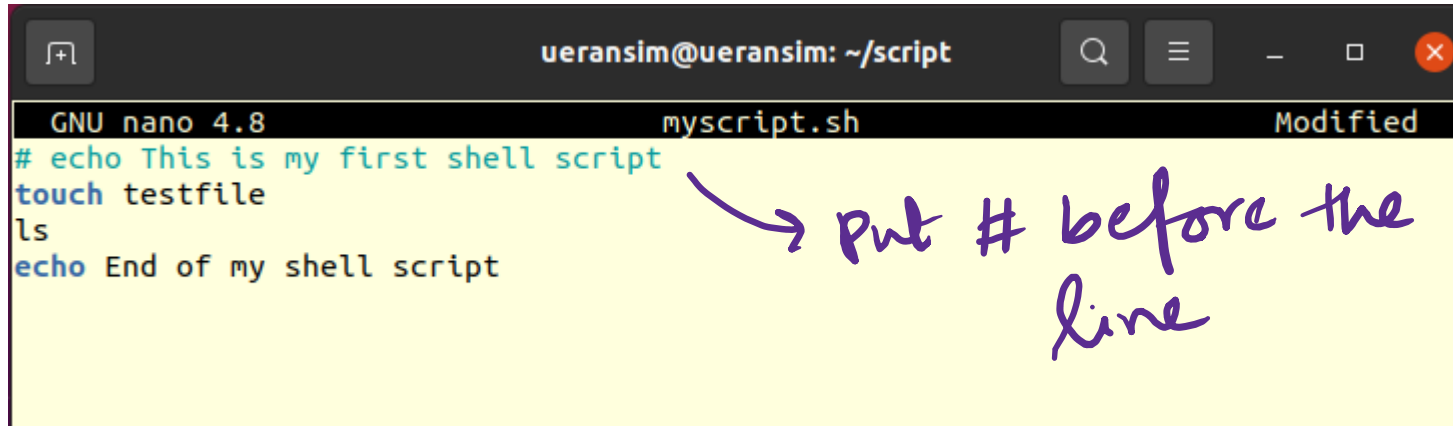
Run *myscript.sh*

```
ueransim@ueransim: ~/script
GNU nano 4.8 myscript.sh
echo This is my first shell script
touch testfile
ls
echo End of my shell script
```

What is done
by this command?

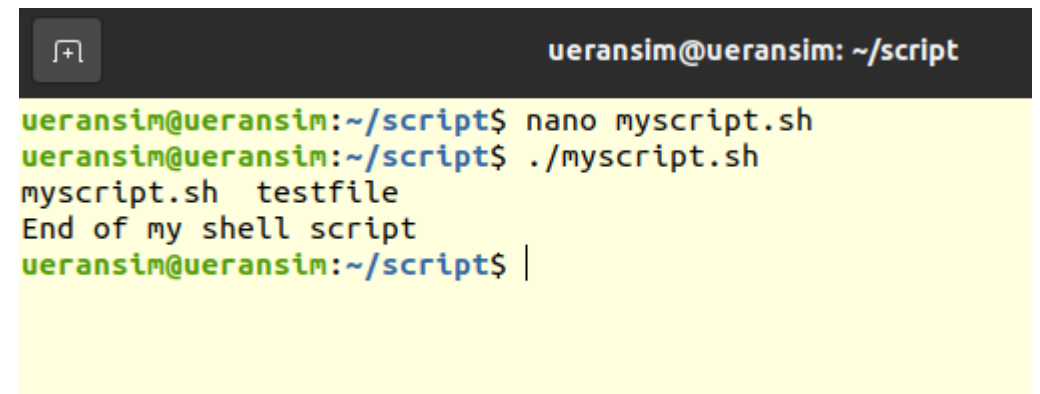
```
ueransim@ueransim: ~/script
ueransim@ueransim:~/script$ nano myscript.sh
ueransim@ueransim:~/script$ ./myscript.sh
This is my first shell script
myscript.sh testfile
End of my shell script
ueransim@ueransim:~/script$ |
```

If I want to comment a line



A screenshot of a terminal window with a nano editor. The window title is "ueransim@ueransim: ~/script". The editor shows a file named "myscript.sh" with the following content: `# echo This is my first shell script`, `touch testfile`, `ls`, and `echo End of my shell script`. A purple handwritten note with an arrow points to the first line, saying "put # before the line".

```
GNU nano 4.8      myscript.sh      Modified
# echo This is my first shell script
touch testfile
ls
echo End of my shell script
```



A screenshot of a terminal window with the title "ueransim@ueransim: ~/script". It shows the execution of the script: `ueransim@ueransim:~/script$ nano myscript.sh`, `ueransim@ueransim:~/script$./myscript.sh`, followed by the output: `myscript.sh testfile`, `End of my shell script`, and the prompt `ueransim@ueransim:~/script$`.

```
ueransim@ueransim:~/script$ nano myscript.sh
ueransim@ueransim:~/script$ ./myscript.sh
myscript.sh testfile
End of my shell script
ueransim@ueransim:~/script$
```

Can I use variables in shell-scripts like programs?

- System-defined variable/ Environment variable
- User-defined variable

```
ueransim@ueransim: ~/script
GNU nano 4.8      myscript.sh
# echo This is my first shell script
touch testfile
ls

# Accesssing Environment variable
echo $USER

#Creating and accessing user-defined variable
my_variable="Hello World!"
echo $my_variable

echo End of my shell script
```

```
ueransim@ueransim: ~/script
ueransim@ueransim:~/script$ ./myscript.sh
myscript.sh testfile
ueransim
Hello World!
End of my shell script
ueransim@ueransim:~/script$
```

Shell-script interpreter

- Many Shells available in Linux
 - The bourne shell(sh)
 - The Korn Shell(ksh)
 - GNU Bourne-Again Shell(bash)
- Scripts written for the *sh shell* are called shell scripts, and they can be interpreted by both, the ksh and bash shells
- *ksh* and *bash* are improved versions of the original sh shell and they have more features than sh
- Bash is generally the default shell in most of the Linux Distributions and scripts written specifically for bash shell are called bash scripts.

Comparison in shell-scripts

- Integer comparison

Operator	Description
-eq	is equal to
-ne	is not equal to
-gt	is greater than
-ge	is greater than or equal to
-lt	is less than
-le	is less than or equal to

- String comparison

Operator	Description
==	is equal to
!=	is not equal to
\<	is less than, in ASCII alphabetical order
\>	is greater than, in ASCII alphabetical order

When do we need this?

Conditional *if* statement in shell-script

Syntax

```
if [ condition ]  
then  
#statements  
fi
```

example:

```
#!/bin/sh  
x=10  
y=11  
if [ $x -ne $y ]  
then  
echo "Not equal"  
fi
```

—————→ *interpreter*

Conditional *if-else* statement in shell-script

Syntax

```
if [ condition ]  
then  
#set of statements if the condition is true  
else  
#set of statements if the condition is false  
fi
```

```
#!/bin/sh  
x=10  
y=10  
if [ $x -ne $y ]  
then  
echo "Not equal"  
else  
echo "They are equal"  
fi
```

While loop statement in shell-script

Syntax

```
while [ condition ]  
do  
#set of statements  
done
```

```
#!/bin/sh  
x=2  
while [ $x -lt 6 ]  
do  
echo $x  
x=`expr $x + 1`  
done
```

expression
- will be
evaluated

For loop in shell-script

Syntax

```
for var in val1 val2 val3
do
#statements
done
```

```
#!/bin/sh
for var in 2 4 5 8
do
echo $var
done
```

Assignment 1: Basics on shell-script

- Write a shell script to display all file information in detail within the working directory
- Write a shell script to display the running processes in the system
- Write a shell script to create a file, write something into the file, and make the file executable

Assignment 2

- Suppose you need the following packages to execute your program:
 - Python3.6
 - Numpy
- Your task: Write a shell script to install the packages

Assignment 2: Steps for python installation

- Check whether python is already available
- If available, check the python version
- Compare the existing version with the required version
- If not available, install using “*sudo apt-get install python3.6*”
- Display the python version
- Note: you need to run your shell script using *sudo*

Assignment 2: Steps for numpy installation

- Check whether numpy is already installed
- If available, update the numpy to the latest version

Assignment 3

- Write a Shell script for reading and saving terminal output into a file
 - Hint: use *`yourCommand 2>&1 | tee outputFile.txt`*
- *Tee*: The tee command is **normally used to split the output of a program so that it can be both displayed and saved in a file.**

More practice sets

- <https://www.emertxe.com/embedded-systems/linux-systems/ls-assignments/>