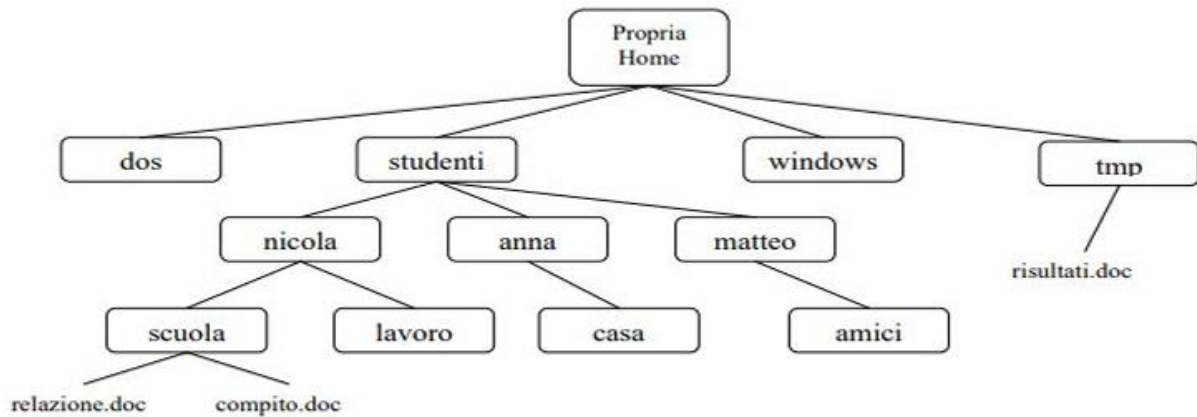


## Esercizio Shell

### Esercizio 1:

Come prima cosa creare le seguenti cartelle e sottocartelle:



Iniziamo aprendo il terminale e creando una nuova cartella in desktop con il nome esercizio dove andremo ad inserire tutte queste cartelle e sottocartelle:

```
(kali㉿kali)-[~]  
└─$ cd Desktop  
  
(kali㉿kali)-[~/Desktop]  
└─$ mkdir Esercizio  
  
(kali㉿kali)-[~/Desktop]  
└─$ ls  
Esercizio  
  
(kali㉿kali)-[~/Desktop]  
└─$ cd Esercizio
```

Creata la cartella iniziamo ad inserire tutte le cartelle che ci occorrono compresi i vari file:

```
(kali㉿kali)-[~/Desktop/Esercizio]  
└─$ mkdir dos  
  
(kali㉿kali)-[~/Desktop/Esercizio]  
└─$ mkdir studenti  
  
(kali㉿kali)-[~/Desktop/Esercizio]  
└─$ mkdir windows  
  
(kali㉿kali)-[~/Desktop/Esercizio]  
└─$ mkdir tmp  
  
(kali㉿kali)-[~/Desktop/Esercizio]  
└─$ ls  
dos studenti tmp windows
```

Abbiamo appena creato le 4 “cartelle principali” andiamo a creare le varie sottocartelle:

Con Cd ci spostiamo nella cartella studenti dove andiamo successivamente a creare le 3 cartelle Anna Matteo Nicola (mi scuso per gli errori di percorso ma dovrebbe essere chiaro lo stesso spero)

```
(kali㉿kali)-[~/Desktop/Esercizio]
$ cd studenti

(kali㉿kali)-[~/Desktop/Esercizio/studenti]
$ mkdir nicola

(kali㉿kali)-[~/Desktop/Esercizio/studenti]
$ mkdir (anna,matte)
zsh: number expected

(kali㉿kali)-[~/Desktop/Esercizio/studenti]
$ mkdir (anna,matteo)
zsh: number expected

(kali㉿kali)-[~/Desktop/Esercizio/studenti]
$ ls
nicola

(kali㉿kali)-[~/Desktop/Esercizio/studenti]
$ mkdir anna

(kali㉿kali)-[~/Desktop/Esercizio/studenti]
$ mkdir matteo

(kali㉿kali)-[~/Desktop/Esercizio/studenti]
$ mkdir matteo
mkdir: cannot create directory 'matteo': File exists

(kali㉿kali)-[~/Desktop/Esercizio/studenti]
$ ls
anna matteo nicola
```

Ci spostiamo nella cartella nicola e continuiamo creando le cartelle scuola e lavoro e verifichiamo con ls la riuscita dell'operazione,

```
(kali㉿kali)-[~/Desktop/Esercizio/studenti]
$ cd nicola

(kali㉿kali)-[~/Desktop/Esercizio/studenti/nicola]
$ mkdir scuola

(kali㉿kali)-[~/Desktop/Esercizio/studenti/nicola]
$ mkdir lavoro

(kali㉿kali)-[~/Desktop/Esercizio/studenti/nicola]
$ ls
lavoro  scuola
```

Per completare la fila ci spostiamo in scuola e creiamo i due file: relazione e compito e verifichiamo con ls la riuscita dell'operazione

```
(kali㉿kali)-[~/.../Esercizio/studenti/nicola/scuola]
$ touch compito.doc

(kali㉿kali)-[~/.../Esercizio/studenti/nicola/scuola]
$ touch relazione.doc

(kali㉿kali)-[~/.../Esercizio/studenti/nicola/scuola]
$ ls
compito.doc  relazione.doc
```

Con i due punti .. torniamo indietro nelle cartelle (ovvero andiamo a studenti) per poi muoverci in anna per andare a creare la cartella casa (stessa cosa di prima la dimenticanza di ls ma ho verificato la riuscita)

```
(kali㉿kali)-[~/Desktop/Esercizio/studenti/nicola]
$ cd ..

(kali㉿kali)-[~/Desktop/Esercizio/studenti]
$ cd anna

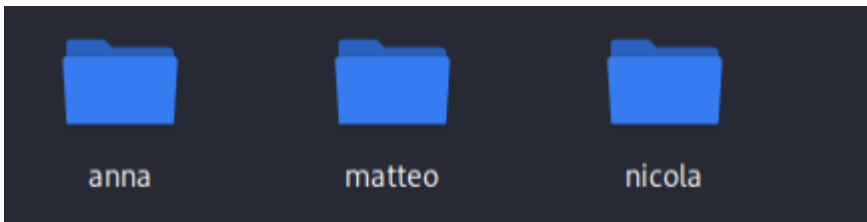
(kali㉿kali)-[~/Desktop/Esercizio/studenti/anna]
$ mkdir casa
```

Per finire questa sezione andiamo a creare quest'ultima sottocartella nella cartella Matteo.

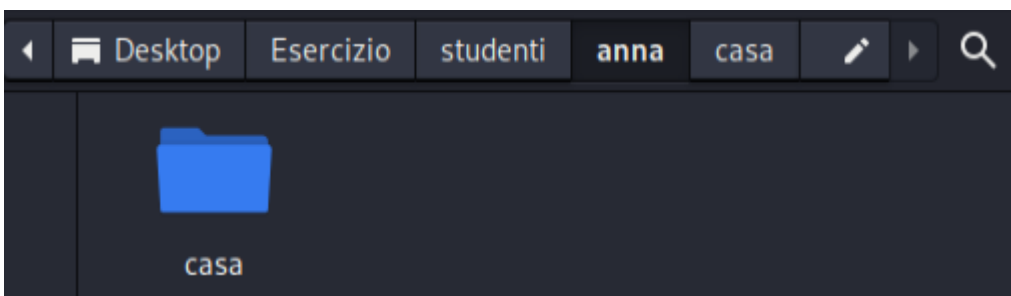
```
(kali㉿kali)-[~/Desktop/Esercizio/studenti]
$ cd matteo

(kali㉿kali)-[~/Desktop/Esercizio/studenti/matteo]
$ mkdir amici
```

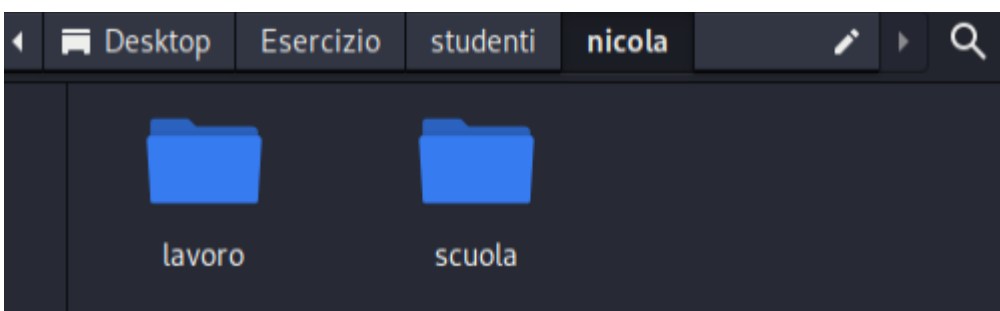
In Conclusione, per quanto riguarda gli studenti avremo queste cartelle:



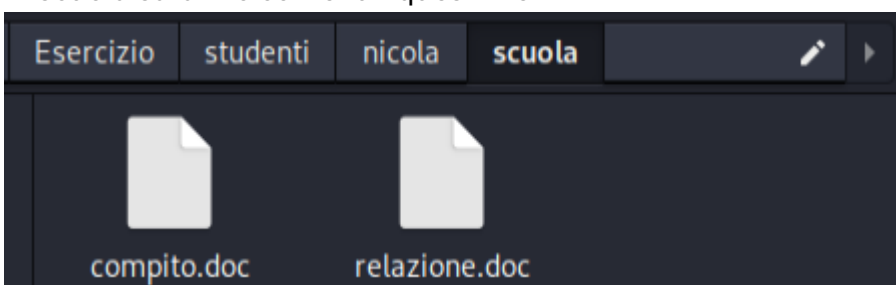
La cartella casa sotto anna,



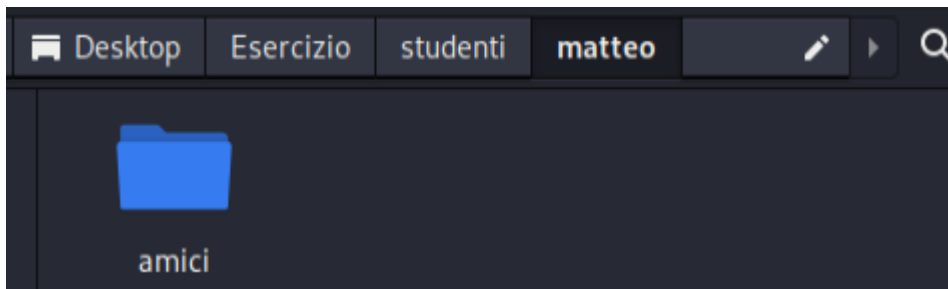
La cartella lavoro e scuola sotto nicola,



In scuola saranno contenuti questi file:



La cartella amici sotto Matteo:



In fine per completare la struttura dell'esercizio inseriamo nella cartella temp risultati.doc:

```
(kali㉿kali)-[~/Desktop/Esercizio]
$ cd tmp/

(kali㉿kali)-[~/Desktop/Esercizio/tmp]
$ touch risultati.doc

(kali㉿kali)-[~/Desktop/Esercizio/tmp]
$ ls
risultati.doc
```

Tramite il comando tree da CLI possiamo vedere la struttura di tutta la cartella “principale” Esercizio:

```
(kali㉿kali)-[~/Desktop/Esercizio]
$ tree
.
├── dos
├── studenti
│   ├── anna
│   │   └── casa
│   ├── matteo
│   │   ├── amici
│   │   ├── nicola
│   │   │   ├── lavoro
│   │   │   └── scuola
│   │       ├── compito.doc
│   │       └── relazione.doc
│   └── tmp
│       └── risultati.doc
└── windows

12 directories, 3 files
```



Ti trovi nella directory lavoro (sotto nicola), scrivere il comando per passare alla directory casa (sotto anna) con percorso relativo e percorso assoluto:

percorso assoluto:

```
(kali@kali)-[~/../Esercizio/studenti/nicola/lavoro]
$ cd /home/kali/Desktop/Esercizio/studenti/anna/casa
(kali@kali)-[~/../Esercizio/studenti/anna/casa]
$
```

Percorso relativo

```
(kali@kali)-[~/../Esercizio/studenti/nicola/lavoro]
$ cd ../ ../anna/casa
(kali@kali)-[~/../Esercizio/studenti/anna/casa]
$
```

Iniziamo con l'esercitazione:

- a) Copia il file compito.doc (dalla directory scuola) alla directory corrente (casa):

```
(kali@kali)-[~/../Esercizio/studenti/anna/casa]
$ cp ../ ../nicola/scuola/compito.doc compito.doc
(kali@kali)-[~/../Esercizio/studenti/anna/casa]
$ ls
compito.doc
```

- b) Sposta il file relazione.doc nella directory corrente (casa):

```
(kali@kali)-[~/../Esercizio/studenti/anna/casa]
$ mv ../ ../nicola/scuola/relazione.doc .
(kali@kali)-[~/../Esercizio/studenti/anna/casa]
$ ls
compito.doc  relazione.doc
```

- c) Cancella la cartella tmp:

```
(kali@kali)-[~/Desktop/Esercizio]
$ rm tmp/risultati.doc
(kali@kali)-[~/Desktop/Esercizio]
$ rmdir tmp
(kali@kali)-[~/Desktop/Esercizio]
$ ls
dos  studenti  windows
```

- d) Creare il file pippo.txt nella cartella lavoro:

```
(kali@kali)-[~/Desktop/Esercizio]
$ cd /home/kali/Desktop/Esercizio/studenti/nicola/lavoro

(kali@kali)-[~/../Esercizio/studenti/nicola/lavoro]
$ touch pippo.txt

(kali@kali)-[~/../Esercizio/studenti/nicola/lavoro]
$ ls
pippo.txt
```

- e) Cambiare gli attributi del file pippo.txt e renderlo scrivibile e leggibile solo per il proprietario, mentre tutti gli altri solo leggibile:

```
(kali@kali)-[~/../Esercizio/studenti/nicola/lavoro]
$ ls -l pippo.txt
-rw-rw-r-- 1 kali kali 0 Jul 25 00:34 pippo.txt

(kali@kali)-[~/../Esercizio/studenti/nicola/lavoro]
$ chmod 644 pippo.txt

(kali@kali)-[~/../Esercizio/studenti/nicola/lavoro]
$ ls -l pippo.txt
-rw-r--r-- 1 kali kali 0 Jul 25 00:34 pippo.txt
```

- f) Nascondere il contenuto della cartella anna:

```
(kali@kali)-[~/Desktop/Esercizio/studenti]
$ mv anna/ .anna

(kali@kali)-[~/Desktop/Esercizio/studenti]
$ ls
matteo nicola
```

- g) Spostarsi nella cartella lavoro e visualizzare il contenuto del file pippo.txt

```
(kali@kali)-[~/Desktop/Esercizio/studenti/nicola]
$ cd lavoro

(kali@kali)-[~/../Esercizio/studenti/nicola/lavoro]
$ ls
pippo.txt

(kali@kali)-[~/../Esercizio/studenti/nicola/lavoro]
$ echo "ciaoooo" > pippo.txt

(kali@kali)-[~/../Esercizio/studenti/nicola/lavoro]
$ cat pippo.txt
ciaoooo
```

h) Rimuovere la cartella amici:

```
(kali@kali)-[~/Esercizio/studenti/nicola/lavoro]
$ cd ..

(kali@kali)-[~/Desktop/Esercizio/studenti/nicola]
$ cd ..

(kali@kali)-[~/Desktop/Esercizio/studenti]
$ cd matteo

(kali@kali)-[~/Desktop/Esercizio/studenti/matteo]
$ ls
amici

(kali@kali)-[~/Desktop/Esercizio/studenti/matteo]
$ rmdir amici

(kali@kali)-[~/Desktop/Esercizio/studenti/matteo]
$ ls
```

i) Rimuovere tutte le cartelle precedentemente create:

```
(kali@kali)-[~/Desktop/Esercizio]
$ rm -rf studenti windows dos

(kali@kali)-[~/Desktop/Esercizio]
$ ls

(kali@kali)-[~/Desktop/Esercizio]
$
```

Fine esercizio.

Inizio Esercizio facoltativo

2. leggere il manuale del comando job, ps, kill.

Valerio il manuale Job seguendo i passaggi effettuati a lezione non sono riuscito

Aprirlo.

Ps:

```
(kali@kali)-[~/Desktop/Esercizio]
$ man ps
```



```
PS(1) User Commands PS(1)
NAME
  ps - report a snapshot of the current processes.
SYNOPSIS
  ps [options]
DESCRIPTION
  ps displays information about a selection of the active processes. If you want a repetitive update of the selection and the displayed information, use top instead.
  This version of ps accepts several kinds of options:
  1. UNIX options, which may be grouped and must be preceded by a dash.
  2. BSD options, which may be grouped and must not be used with a dash.
  3. GNU long options, which are preceded by two dashes.
  Options of different types may be freely mixed, but conflicts can appear. There are some synonymous options, which are functionally identical, due to the many standards and ps implementations that this ps is compatible with.
  By default, ps selects all processes with the same effective user ID (euid=EUID) as the current user and associated with the same terminal as the invoker. It displays the process ID (pid=PID), the terminal associated with the process (tname=TTY), the cumulated CPU time in [DD-]hh:mm:ss format (time=TIME), and the executable name (ucmd=CMD). Output is unsorted by default.
  The use of BSD-style options will add process state (stat=STAT) to the default display and show the command args (args=COMMAND) instead of the executable name. You can override this with the PS_FORMAT environment variable. The use of BSD-style options will also change the process selection to include processes on other terminals (TTVs) that are owned by you; alternately, this may be described as setting the selection to be the set of all processes filtered to exclude processes owned by other users or not on a terminal. These effects are not considered when options are described as being "identical" below, so -M will be considered identical to -Z and so on.
  Except as described below, process selection options are additive. The default selection is discarded, and then the selected processes are added to the set of processes to be displayed. A process will thus be shown if it meets any of the given selection criteria.
EXAMPLES
  To see every process on the system using standard syntax:
  ps -ef
  ps -ef
  ps -ef
  ps -ely
  To see every process on the system using BSD syntax:
  ps ax
  ps axu
  To print a process tree:
  ps -ejH
  ps axjf
  To get info about threads:
  ps -elf
  ps axms
  To get security info:
  ps -eo user,ruser,suser,fuser,f,comm,label
  ps aZ
  ps -eR
Manual page ps(1) line 1 (press h for help or q to quit)
```

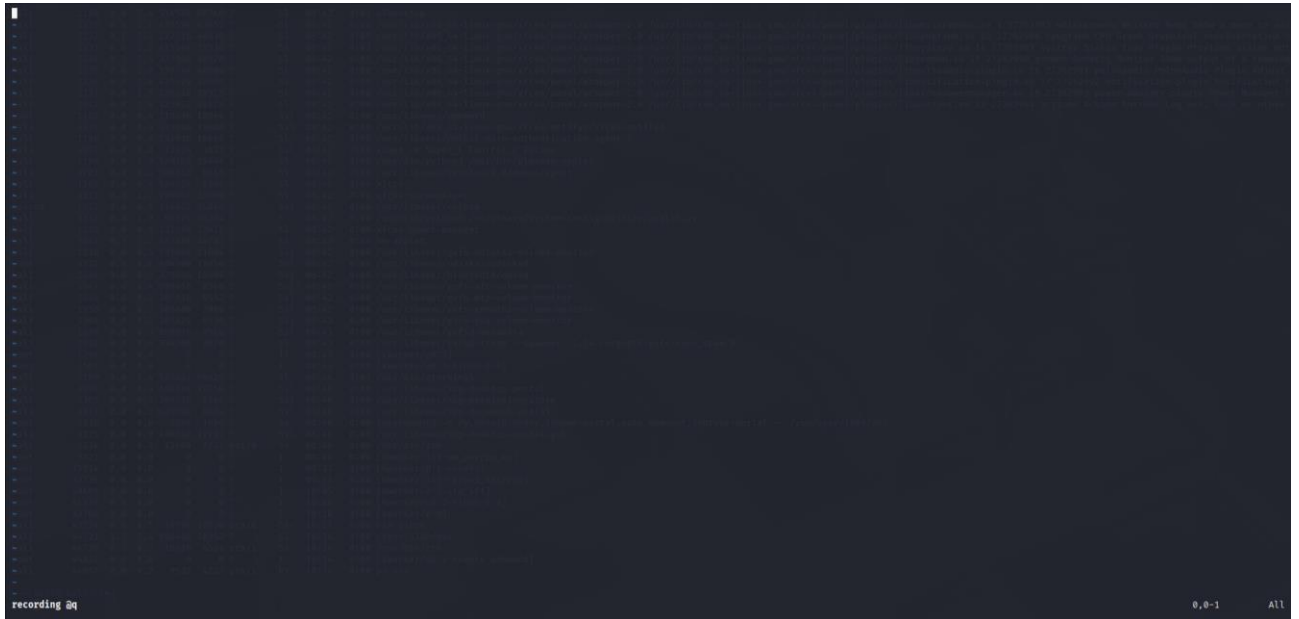
Kill:

```
(kali㉿kali)-[~/Desktop/Esercizio]
$ man kill
```

```
File Actions Edit View Help
kill(1) User Commands kill(1)
NAME
  kill - send a signal to a process
SYNOPSIS
  kill [options] <pid> [...]
DESCRIPTION
  The default signal for kill is TERM. Use -l or -L to list available signals. Particularly useful signals include HUP, INT, KILL, STOP, CONT, and 0. Alternate signals may be specified in three ways: -9, -SIGKILL or -KILL. Negative PID values may be used to choose whole process groups; see the PGID column in ps command output. A PID of -1 is special; it indicates all processes except the kill process itself and init.
OPTIONS
  <pid> [...]
  Send signal to every <pid> listed.
  -[signal]
  -s <signal>
  --signal <signal>
  Specify the signal to be sent. The signal can be specified by using name or number. The behavior of signals is explained in signal(7) manual page.
  -q, --queue <value>
  Use <value>(3) rather than kill(2) and the value argument is used to specify an integer to be sent with the signal. If the receiving process has installed a handler for this signal using the SA_SIGINFO flag to sigaction(2), then it can obtain this data via the si_value field of the siginfo_t structure.
  -l, --list [<signal>]
  List signal names. This option has optional argument, which will convert signal number to signal name, or other way round.
  -L, --table
  List signal names in a nice table.
NOTES
  Your shell (command line interpreter) may have a built-in kill command. You may need to run the command described here as /bin/kill to solve the conflict.
  If you use negative PID values, you will need to specify a signal as well so that kill knows if the option is for the PID or the signal number. For example, Issuing the command with the single option -9 it is not clear if you mean signal 9 (SIGKILL) or process group 9.
EXAMPLES
  kill -9 -1
  Kill all processes you can kill.
  kill -l 11
  Translate number 11 into a signal name.
  kill -L
  List the available signal choices in a nice table.
  kill 123 543 2341 3453
  Send the default signal, SIGTERM, to all those processes.
  kill -SIGTERM -123
  Send the signal SIGTERM to process group 123. The signal name or number is required if specifying process groups with a negative PID.
Manual page kill(1) line 1 (press h for help or q to quit)
```

3. lanciare il comando vi pippo

```
(kali㉿kali)-[~/Desktop/Esercizio]
$ vim pippo
```



4. aprire un nuovo terminale e visualizzare tutti i propri processi:

tramite il comando `ps` vediamo soltanto il processo corrente ed il suo sottoprocesso mentre con `ps aux` vediamo tutti i processi: che appartengono anche ad altri utenti, l'output in formato di utilizzo delle risorse, ed in fine i processi che vengono lanciati senza che nessun terminale li abbia lanciati

```

zsh: corrupt history file /home/kali/.zsh_history
(kali@kali)-[~]
$ ps
  PID TTY          TIME CMD
 44728 pts/1    00:00:00 zsh
 44780 pts/1    00:00:00 ps

(kali@kali)-[~]
$ ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1   0.0  0.7 23820 14516 ?        Ss   08:42   0:01 /sbin/init splash
root         2   0.0  0.0      0     0 ?        S    08:42   0:00 [kthreadd]
root         3   0.0  0.0      0     0 ?        S    08:42   0:00 [pool_workqueue_release]
root         4   0.0  0.0      0     0 ?        I<   08:42   0:00 [kworker/R-kvfree_rcu_reclaim]
root         5   0.0  0.0      0     0 ?        I<   08:42   0:00 [kworker/R-rcu_gp]
root         6   0.0  0.0      0     0 ?        I<   08:42   0:00 [kworker/R-sync_wq]
root         7   0.0  0.0      0     0 ?        I<   08:42   0:00 [kworker/R-slub_flushwq]
root         8   0.0  0.0      0     0 ?        I<   08:42   0:00 [kworker/R-netns]
root        12   0.0  0.0      0     0 ?        I    08:42   0:00 [kworker/u8:0-flush-8:0]
root        13   0.0  0.0      0     0 ?        I<   08:42   0:00 [kworker/R-mm_percpu_wq]
root        14   0.0  0.0      0     0 ?        I    08:42   0:00 [rcu_tasks_kthread]
root        15   0.0  0.0      0     0 ?        I    08:42   0:00 [rcu_tasks_rude_kthread]
root        16   0.0  0.0      0     0 ?        I    08:42   0:00 [rcu_tasks_trace_kthread]
root        17   0.0  0.0      0     0 ?        S    08:42   0:00 [ksoftirqd/0]
root        18   0.0  0.0      0     0 ?        I    08:42   0:00 [rcu_preempt]
root        19   0.0  0.0      0     0 ?        S    08:42   0:00 [rcu_exp_par_gp_kthread_worker/0]
root        20   0.0  0.0      0     0 ?        S    08:42   0:00 [rcu_exp_gp_kthread_worker]
root        21   0.0  0.0      0     0 ?        S    08:42   0:00 [migration/0]
root        22   0.0  0.0      0     0 ?        S    08:42   0:00 [idle_inject/0]
root        23   0.0  0.0      0     0 ?        S    08:42   0:00 [cpuhp/0]
root        24   0.0  0.0      0     0 ?        S    08:42   0:00 [cpuhp/1]
root        25   0.0  0.0      0     0 ?        S    08:42   0:00 [idle_inject/1]
root        26   0.0  0.0      0     0 ?        S    08:42   0:00 [migration/1]
root        27   0.0  0.0      0     0 ?        S    08:42   0:00 [ksoftirqd/1]
root        29   0.0  0.0      0     0 ?        I<   08:42   0:00 [kworker/1:0H-events_highpri]
root        32   0.0  0.0      0     0 ?        S    08:42   0:00 [kdevtmpfs]
root        33   0.0  0.0      0     0 ?        I<   08:42   0:00 [kworker/R-inet_frag_wq]
root        34   0.0  0.0      0     0 ?        S    08:42   0:00 [kauditd]
root        35   0.0  0.0      0     0 ?        S    08:42   0:00 [khungtaskd]
root        36   0.0  0.0      0     0 ?        S    08:42   0:00 [oom_reaper]
root        38   0.0  0.0      0     0 ?        I<   08:42   0:00 [kworker/R-writeback]
root        39   0.0  0.0      0     0 ?        S    08:42   0:00 [kcompactd0]
root        40   0.0  0.0      0     0 ?        SN   08:42   0:00 [ksmd]
root        41   0.0  0.0      0     0 ?        SN   08:42   0:00 [khugepaged]
root        42   0.0  0.0      0     0 ?        I<   08:42   0:00 [kworker/R-kintegrityd]
root        43   0.0  0.0      0     0 ?        I<   08:42   0:00 [kworker/R-kblockd]
root        44   0.0  0.0      0     0 ?        I<   08:42   0:00 [kworker/R-blkcg_punt_bio]
root        45   0.0  0.0      0     0 ?        S    08:42   0:00 [irq/9-acpi]
root        47   0.0  0.0      0     0 ?        I<   08:42   0:00 [kworker/R-tpm_dev_wq]
root        48   0.0  0.0      0     0 ?        I<   08:42   0:00 [kworker/R-edac-poller]
root        49   0.0  0.0      0     0 ?        I<   08:42   0:00 [kworker/R-devfreq_wq]
root        50   0.0  0.0      0     0 ?        I<   08:42   0:00 [kworker/1:1H-kblockd]
root        51   0.0  0.0      0     0 ?        S    08:42   0:00 [kswapd0]

```

5. cercare di terminale (killare) il processo vi per sbloccare il terminale precedente:

innanzitutto cerchiamo il file da killare ovvero vim pippo tramite il comando:

ps aux | grep pippo

```

(kali@kali)-[~]
$ ps aux | grep pippo
kali      43724  0.0  0.5 16396 10820 pts/0    Sl+  10:11   0:00 vim pippo
kali      49559  0.0  0.1  6528  2132 pts/1    S+   10:24   0:00 grep --color=auto pippo

```

Infine killiamo il programma:

tramite il comando: kill -9 43724 (nominativo numerale di vim pippo)

```

(kali@kali)-[~]
$ kill -9 43724

```

```
zsh: killed vim pippo
~
(kali@kali)-[~/Desktop/Esercizio]
$ 0;13;5M0;13;5m64;25;5M0;35;7M0;35;7m0;36;8M0;36;8m
```

6. lanciare il comando firefox in background:

tramite il comando `firefox &`

```
(kali@kali)-[~]
$ firefox &
[1] 57676
```

7. portarlo in background

Tramite CTRL Z lo sospendiamo e tramite `bg` lo portiamo in background

```
(kali@kali)-[~]
$ firefox
^Z
zsh: suspended firefox

(kali@kali)-[~]
$ bg
[1] + continued firefox
```

8. killare firefox

Riapriamo firefox, lo mettiamo in background, tramite `jobs` vediamo il suo codice (numero 1) tramite `kill %1` killiamo il codice e quindi il programma firefox stesso.

```
(kali@kali)-[~/Desktop/Esercizio]
$ firefox
^Z
zsh: suspended firefox

(kali@kali)-[~]
$ jobs
[1] + suspended firefox

(kali@kali)-[~/Desktop/Esercizio]
$ kill %1

(kali@kali)-[~/Desktop/Esercizio]
$ 
[1] + terminated firefox
```

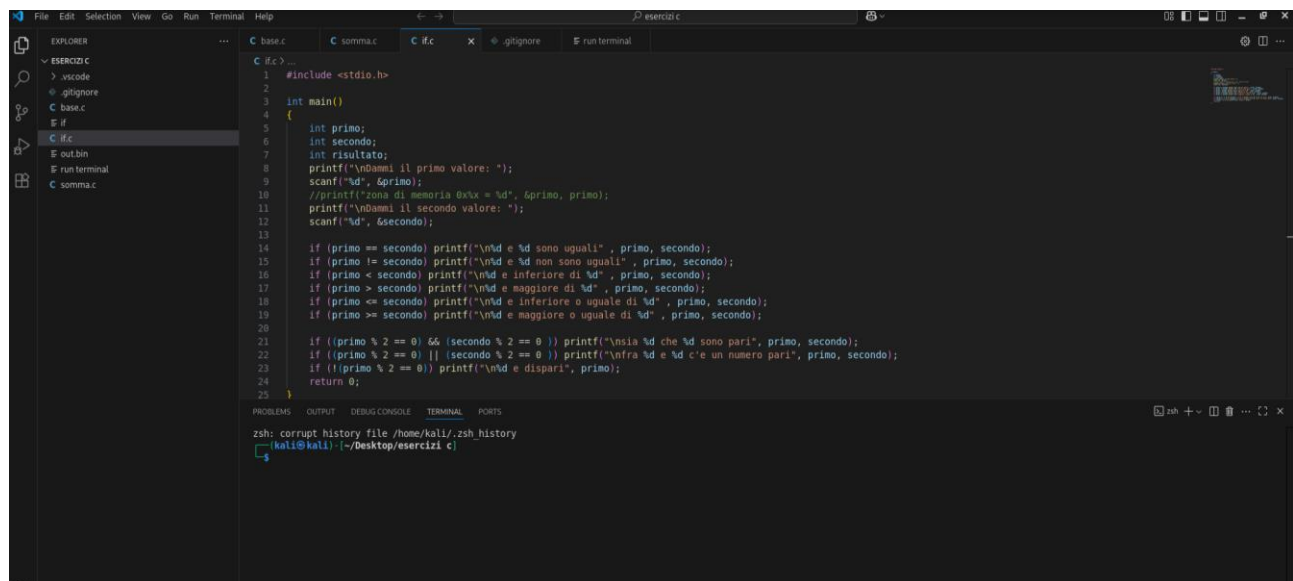
9. verificare quanto spazio si sta occupando su disco:

tramite il comando `df -h` visualizziamo lo spazio disponibile:

```
(kali@kali)-[~]
└─$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            921M   0    921M   0% /dev
tmpfs           198M  992K   197M   1% /run
/dev/sda1       79G   16G   59G   22% /
tmpfs           987M  28M   959M   3% /dev/shm
tmpfs           5.0M   0    5.0M   0% /run/lock
tmpfs           1.0M   0    1.0M   0% /run/credentials/systemd-journald.service
tmpfs           987M  2.3M   985M   1% /tmp
tmpfs           1.0M   0    1.0M   0% /run/credentials/getty@tty1.service
tmpfs           198M  124K   198M   1% /run/user/1000
```

Installazione visual studio:

Ho scaricato ed installato visual studio a lezione e funziona tutto correttamente, di seguito riporto la foto di un esercizio che stavamo facendo a lezione per dimostrare che funziona.



The screenshot shows the Visual Studio Code interface. The Explorer pane on the left shows a project named 'ESERCIZI C' with files 'base.c', 'somma.c', and 'if.c'. The 'if.c' file is open in the editor, displaying a C program that compares two integers and checks for parity. The program includes `<stdio.h>` and defines a `main` function. It prompts the user for two values, reads them with `scanf`, and then uses a series of `if` statements to compare the values and check if they are even or odd. The output of the program is shown in the terminal window at the bottom, which displays the prompt `zsh: corrupt history file /home/kali/.zsh_history` followed by the command `if.c` and the prompt `─$`.

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int primo;
6     int secondo;
7     int risultato;
8     printf("\nDammi il primo valore: ");
9     scanf("%d", &primo);
10    //printf("Zona di memoria 0x%x = %d", &primo, primo);
11    printf("\nDammi il secondo valore: ");
12    scanf("%d", &secondo);
13
14    if (primo == secondo) printf("\nd e %d sono uguali", primo, secondo);
15    if (primo != secondo) printf("\nd e %d non sono uguali", primo, secondo);
16    if (primo < secondo) printf("\nd e inferiore di %d", primo, secondo);
17    if (primo > secondo) printf("\nd e maggiore di %d", primo, secondo);
18    if (primo <= secondo) printf("\nd e inferiore o uguale di %d", primo, secondo);
19    if (primo >= secondo) printf("\nd e maggiore o uguale di %d", primo, secondo);
20
21    if ((primo % 2 == 0) && (secondo % 2 == 0)) printf("\nsia %d che %d sono pari", primo, secondo);
22    if ((primo % 2 == 0) || (secondo % 2 == 0)) printf("\nra %d e %d c'e un numero pari", primo, secondo);
23    if (!(primo % 2 == 0)) printf("\nd e dispari", primo);
24    return 0;
25 }
```

```
zsh: corrupt history file /home/kali/.zsh_history
─$ if.c
─$
```