

Shell di Linux

Parte 1



Yilei Wu

23 Luglio 2024

Sommario

Consegna Esercizio 1	3
Legenda comandi terminale sulla gestione file & cartelle	4
Cartelle	5
Gerarchia e ramificazione delle cartelle	5
Creazione delle cartelle	6
Percorso relativo e percorso assoluto	9
a) Copia il file compito.doc (nella directory scuola) nella directory corrente (casa)	10
b) Sposta il file relazione.doc (nella directory scuola) nella directory corrente (casa)	10
c) Cancella la cartella \tmp	11
d) creare il file pippo.txt nella cartella lavoro	11
e) Cambiare gli attributi del file pippo.txt e renderlo scrivibile e leggibile solo per il proprietario, mentre per tutti gli altri solo leggibile... ..	12
f) Nascondere il contenuto della cartella anna	12
g) Spostarsi nella cartella lavoro e visualizzare il contenuto del file pippo.txt	13
h) Rimuovere la cartella amici	14
i) Rimuovere tutte le cartelle precedentemente create.....	14
Conclusione esercizio 1	14
Consegna esercizio 2	15
Legenda del terminale sulla gestione dei processi.....	16
Provare i comandi w / who / who am i	16
1. Aprire un terminale	17
2. Leggere il manuale del comando jobs, ps e kill.....	17
3. Lanciare il comando vi pippo	18
4. Aprire un nuovo terminale e visualizzare tutti i processi... ..	18
5. Cercare di terminare (killare) il processo vi per sbloccare il terminale precedente	19
6. Lanciare il comando firefox in background	20
7. portarlo in background	20
8. cercare di terminare il processo firefox	20
9. verificare quanto spazio si sta occupando su disco.....	21
Conclusione esercizio 2	21

Consegna Esercizio 1

ESERCIZI SHELL

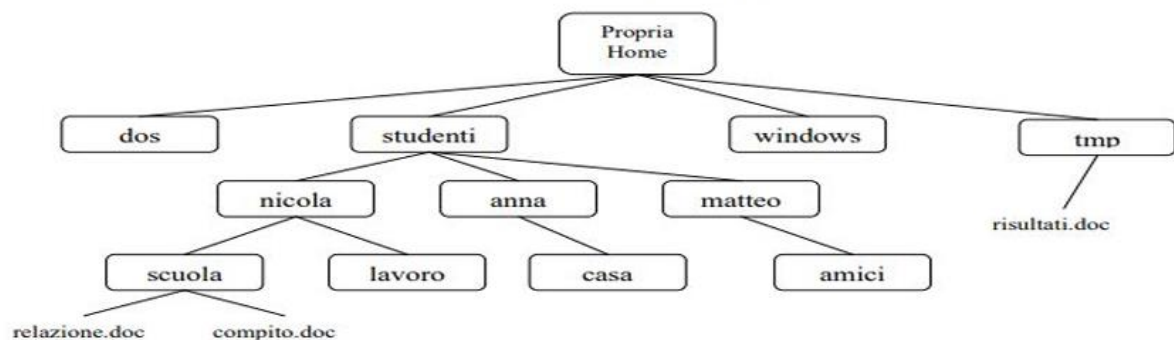
cd	Cambia Directory
Mkdir	Make Directory (fa una directory)
rmdir	rimuove una directory (se vuota)
mv	sposta un file - directory
cp	Copia un file (directory)
rm	Cancella un file
ls	visualizza il contenuto della cartella
pwd	print working directory (stampa il percorso assoluto dove mi trovo)
man argomento	Visualizza il manuale di un comando

Collegarsi al sistema con utenza e password

Esercizio 1

Come prima cosa creare le seguenti cartelle e sottocartelle (usando i comandi "terminale" mkdir cd rmdir ... a partire dalla propria HOME e visualizzarle a video:

(Per "Propria home" si intende il posto dove vi posiziona quanto aprite il terminale!)



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

- Copia il file compito.doc (dalla directory scuola) nella directory corrente (casa).
- Sposta il file relazione.doc nella directory corrente (casa).
- Cancella la cartella **tmp**
- Creare il file pippo.txt nella cartella lavoro
- Cambiare gli attributi del file pippo.txt e renderlo scrivibile e leggibile solo per il proprietario, mentre per tutti gli altri solo leggibile...
- Nascondere il contenuto della cartella anna
- Spostarsi nella cartella lavoro e visualizzare il contenuto del file pippo.txt
- Rimuovere la cartella amici
- Rimuovere tutte le cartelle precedentemente create

Legenda comandi terminale sulla gestione file & cartelle

Lista comandi utili per l'esercizio:

- ✓ **sudo + comando** il comando viene avviato con privilegi di superuser
- ✓ **cd** cambia directory (cd + percorso di destinazione)
 1. **cd-** riporta alla directory da cui sei arrivato immediatamente prima;
 2. **cd ..** porta alla directory padre di quella corrente;
 3. **cd ../..** porta a due livelli più in alto nella struttura delle directory.
- 2. **ls** visualizza il contenuto della cartella
 1. **ls -a** elenca tutto i file e cartelle, incluse quelle nascoste;
 2. **ls -l nome_file** verifica gli attuali permessi del file.
- 3. **pwd** stampa il percorso assoluto di dove mi trovo
- 4. **chmod [opzioni] permessi file**
 1. **Simbolico:**
 - r** (read) Lettura, **w** (write) Scrittura, **x** (execute) Esecuzione;
 - u** (user) Proprietario, **g** (group) Gruppo, **o** (others) Altri utenti.
 2. **Esempi:**
 - Aggiungere permesso di esecuzione per il proprietario: **chmod u+x file.txt**
 - Rimuovere il permesso di scrittura per il gruppo: **chmod g-w file.txt**
 - Impostare i permessi di lettura e scrittura per il proprietario e solo lettura per gli altri: **chmod u=rw,go=r file.txt**
- ✓ **mkdir** creare una cartella
- ✓ **rmdir** rimuovere una cartella vuota
- ✓ **rm** remove
- ✓ **mv** sposta
- ✓ **cp** copia
- ✓ **man argomento** visualizza manuale di un comando
- ✓ editor di testo
 1. nano: **nano nomefile.txt**
Per salvare e uscire **CTRL+O + invio** e **CTRL+X**
 2. vi o vim: **vi nomefile.txt** o **vim nomefile.txt**
Per salvare e uscire, premi **CTRL+C**, poi digita **:wq** e premi Invio. Questo comando scrive le modifiche (w) e chiude l'editor (q).
Per uscire senza salvare, digita **:q!** e premi Invio.
- ✓ **cat nome_file.txt** per leggere l'intero contenuto nel terminale
- ✓ **less nome_file.txt** per scorrere il file avanti e indietro, per uscire premere **q**.

Nascondere i file singolarmente

Rinominando ogni file e cartella all'interno di una directory per aggiungere un punto (.) all'inizio del nome.

Esempio:

```
mv file1.txt .file1.txt
```

```
mv file2.txt .file2.txt
```

```
mv sottocartella .sottocartella
```

Cartelle

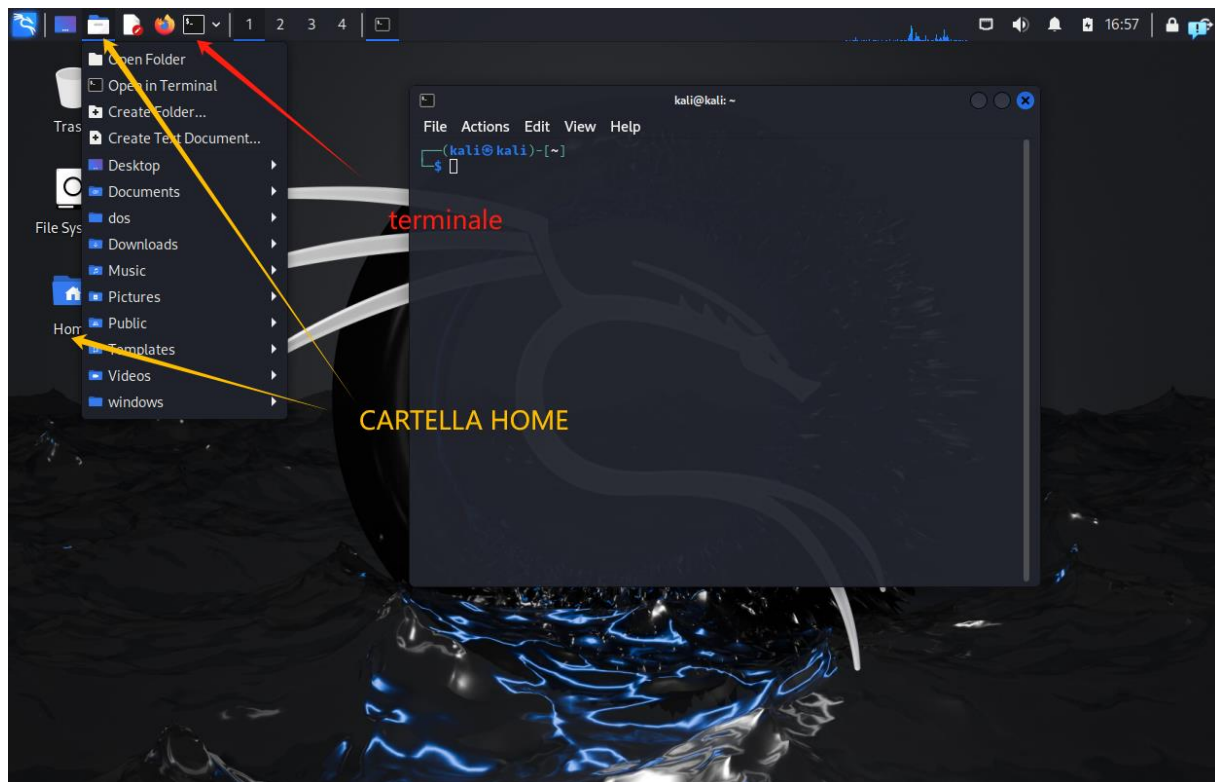
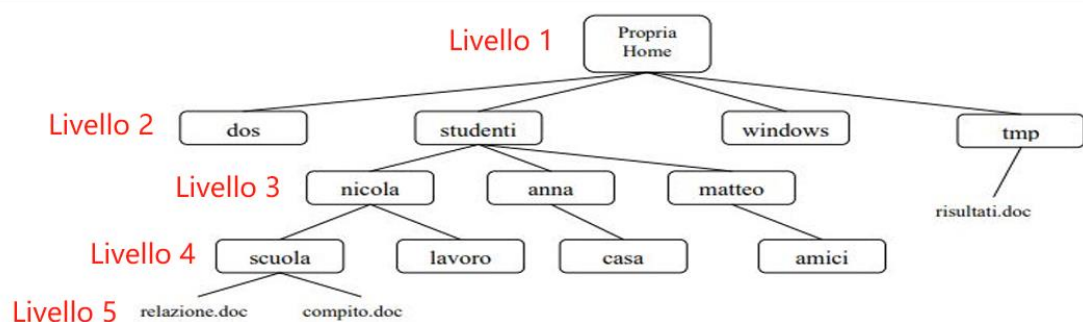


FIGURA 1

Le cartelle e file verranno create nella cartella "home" di Kali Linux come dalle indicazioni in figura di colore giallo.

Gerarchia e ramificazione delle cartelle

Dividiamo la gerarchia delle cartelle e file per una migliore comprensione della ramificazione.



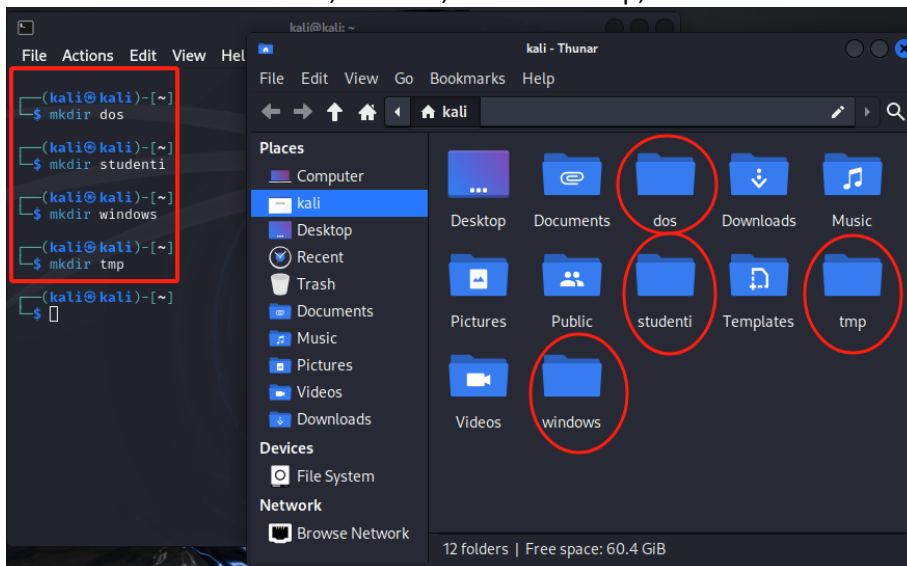
I livelli sono

1. Home
2. Dos – Studenti – Windows – tmp
3. nicola – anna – matteo – risultati.doc
4. scuola – lavoro – casa – amici
5. relazione.doc – compito.doc

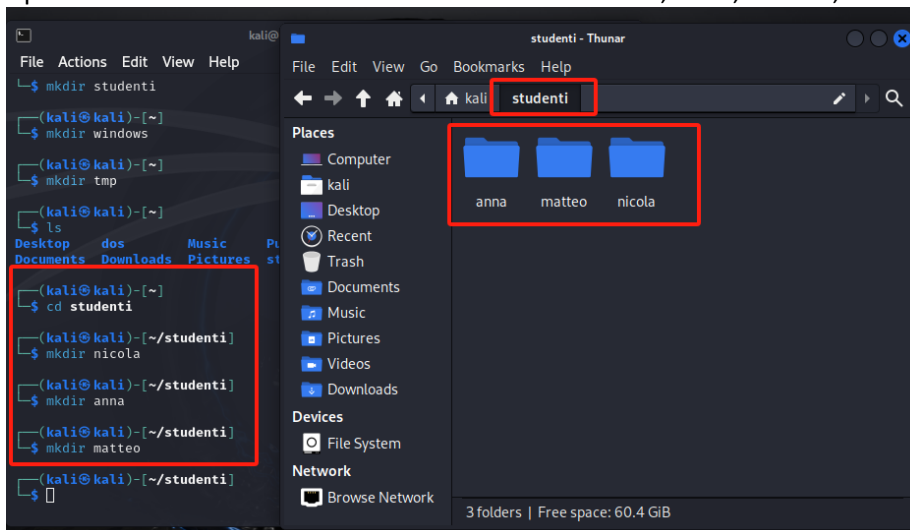
Creazione delle cartelle

Seguiamo il seguente ordine di creazione per essere il più efficienti possibile.

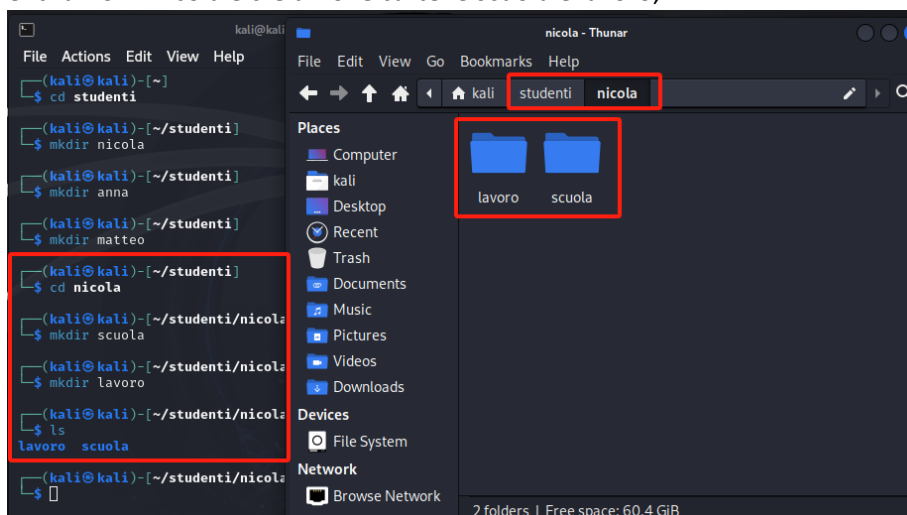
- apriamo il terminale vedi indicazioni in rosso nella figura 1;
- creazione delle cartelle dos, studenti, windows e tmp;



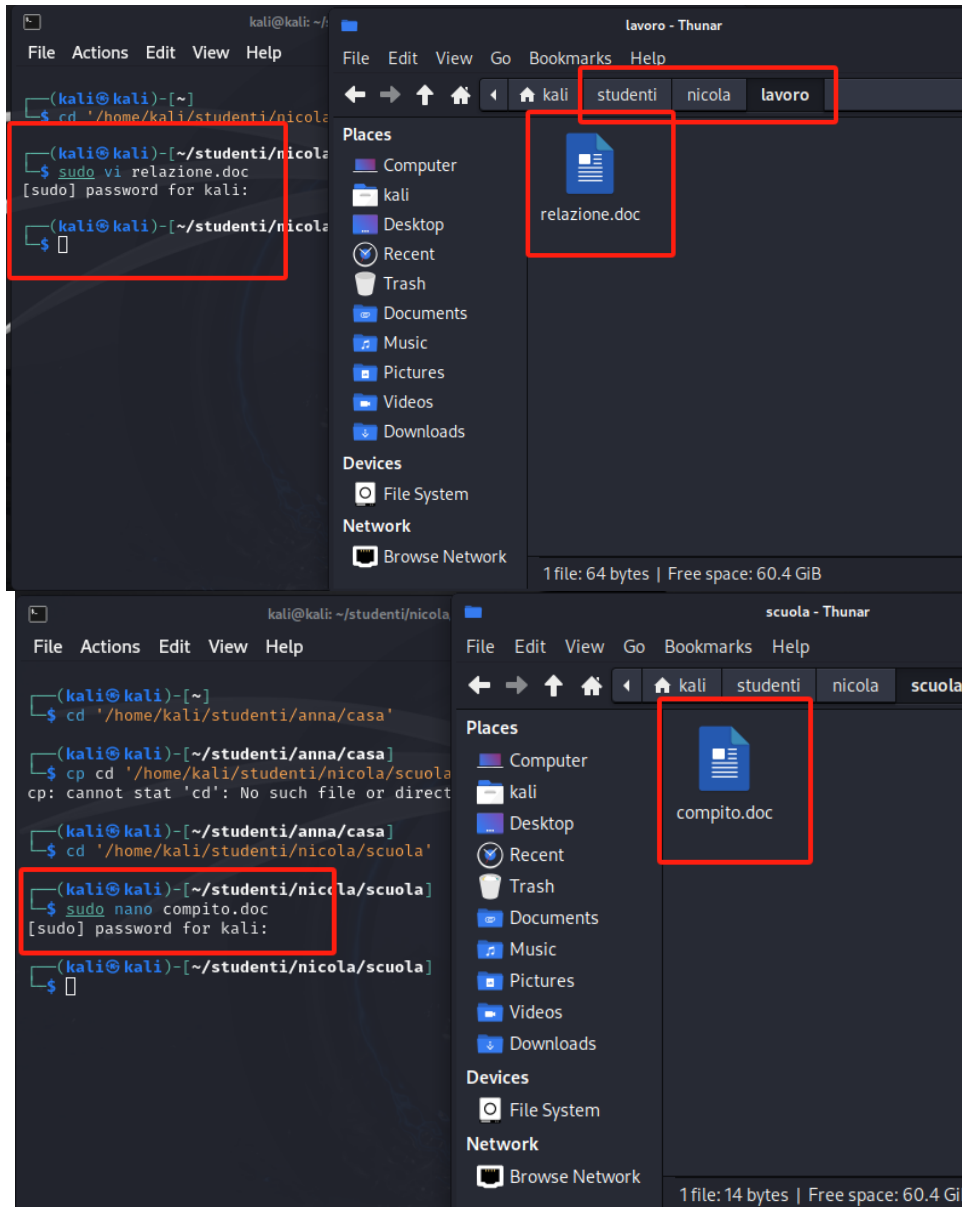
- apriamo la cartella studenti e creiamo le cartelle nicola, anna, matteo;



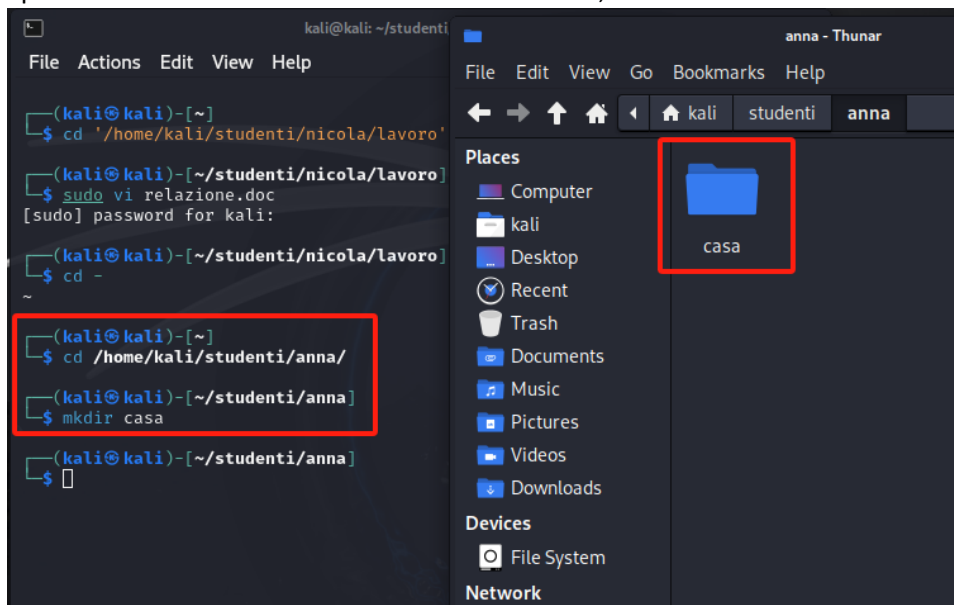
- entriamo in nicola e creiamo le cartelle scuola e lavoro;



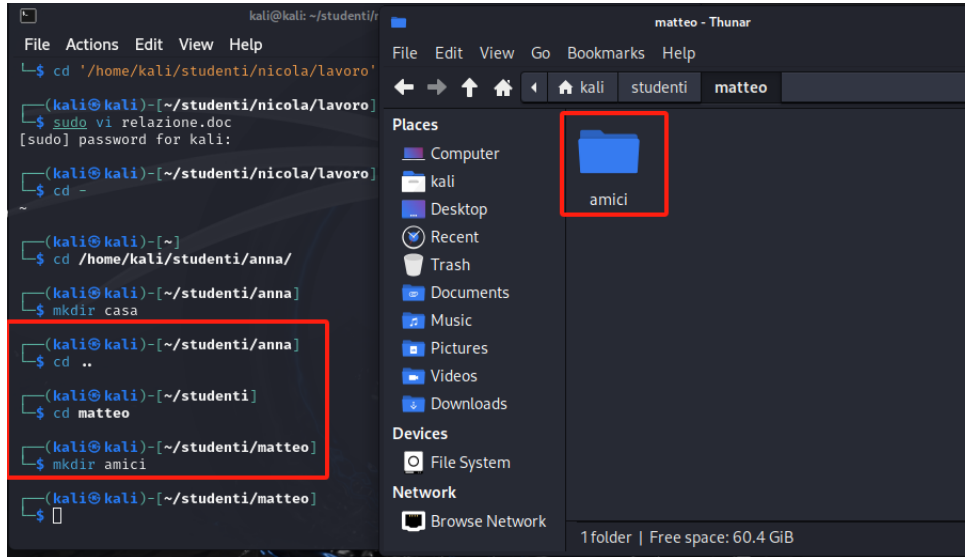
- entriamo in scuola e creiamo la cartella relazione.doc e compito.doc;



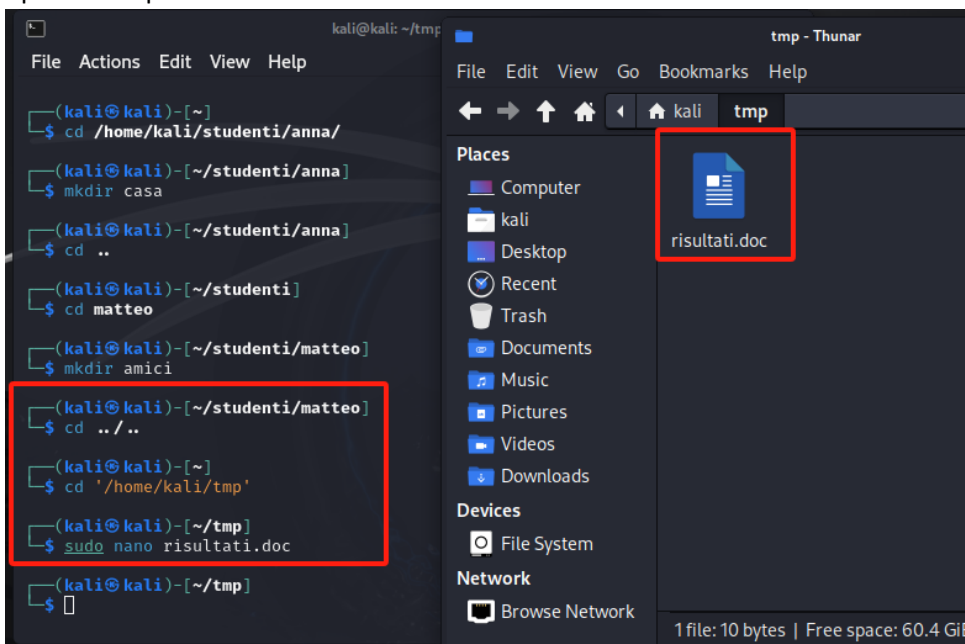
- apriamo la cartella anna e creiamo la cartella casa;



- apriamo la cartella matteo e creiamo la cartella amici;



- apriamo tmp e creiamo il file risultati.doc.



Durante questi passaggi abbiamo utilizzato i vari metodi disponibili per muoverci tra i vari livelli e ramificazione delle cartelle con i comandi `cd` + percorso completo, `cd -`, `cd ..` o `cd ../../` e utilizzato sia `vi` / `vim` e `nano` (vedi legenda comandi). L'utilizzo delle varie metodologie è a discrezione dell'utente. Inoltre si premette che se non avessimo utilizzato l'interfaccia grafica, si può verificare la presenza dei file con il comando **ls** dentro ogni cartella desiderata.



The image shows a terminal window with a dark background. The title bar at the top reads 'kali@kali: ~/studenti/anna/casa'. The menu bar includes 'File', 'Actions', 'Edit', 'View', and 'Help'. The terminal output is as follows:

```
(kali@kali)-[~/tmp]
$ sudo vim risultato.doc

(kali@kali)-[~/tmp]
$ cd '/home/kali/studenti/nicola/lavoro'

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

(kali@kali)-[~/studenti/anna/casa]
$ cd '/home/kali/studenti/nicola/lavoro'

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

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

(kali@kali)-[~/studenti]
$ cd anna

(kali@kali)-[~/studenti/anna]
$ cd casa

(kali@kali)-[~/studenti/anna/casa]
$
```

Annotations on the right side of the terminal:

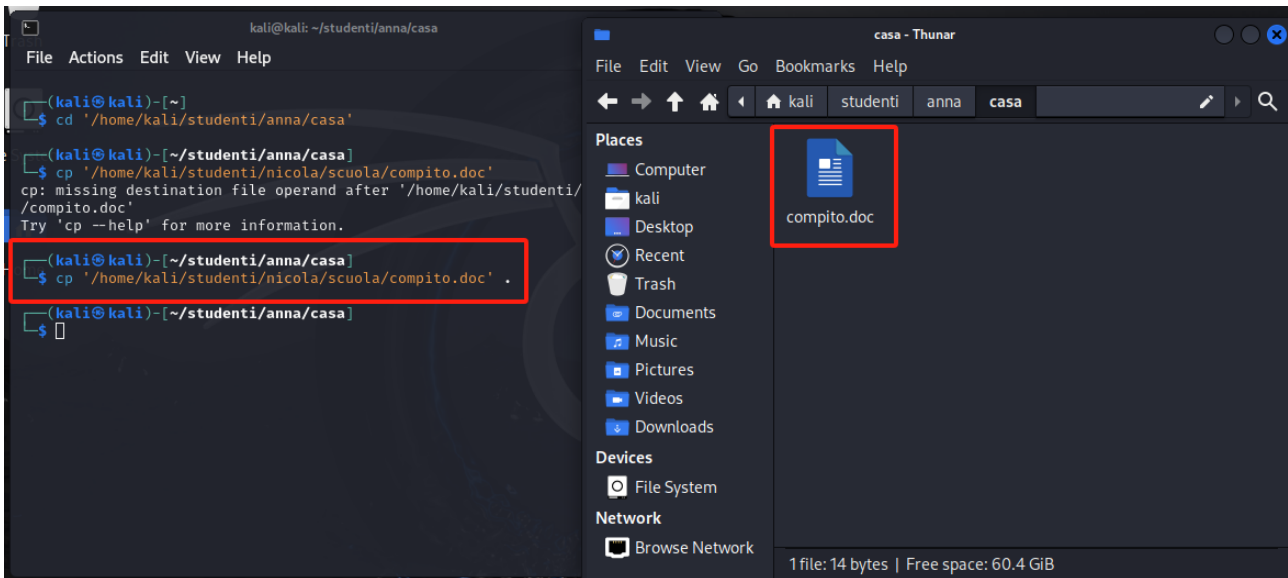
- percorso assoluto** (absolute path): Points to the first two commands, which are enclosed in a red box.
- percorso relativo** (relative path): Points to the subsequent commands, which are enclosed in a green box.

Nel percorso assoluto, in rosso, si inserisce direttamente il percorso completo della directory di destinazione.

Nel percorso relativo, in verde, ci si muove con i comandi `cd ..` e/o `cd ../..` per muoversi tra i vari livelli e ramificazioni.

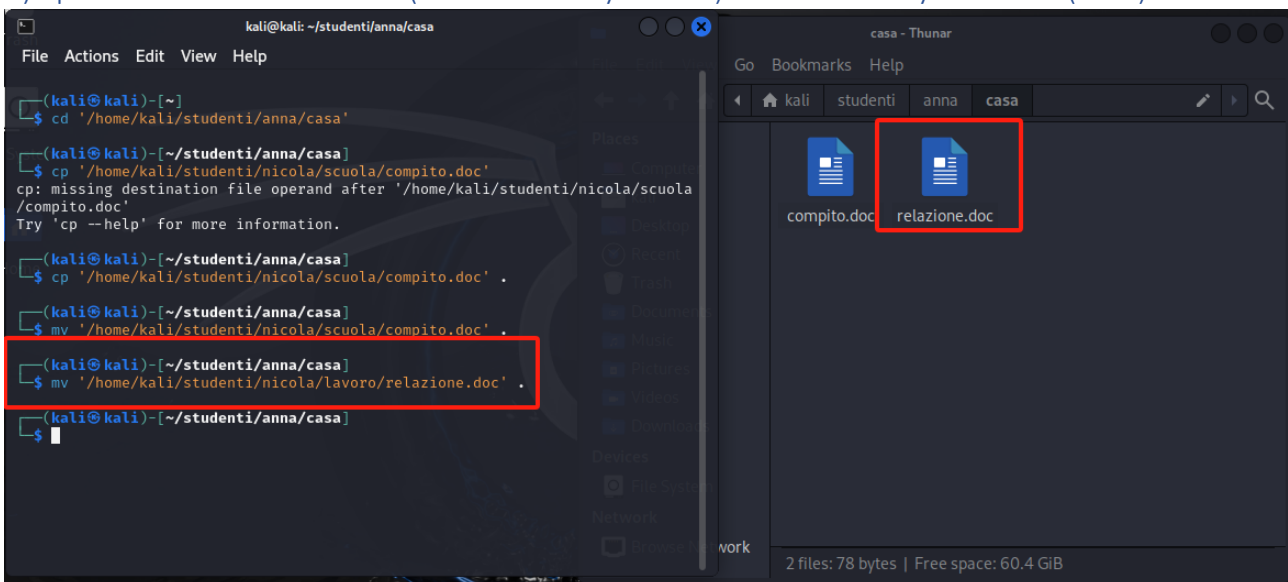
Se si conosce già il percorso completo, per efficienza, si consiglia di eseguire il percorso assoluto.

a) Copia il file compito.doc (nella directory scuola) nella directory corrente (casa)



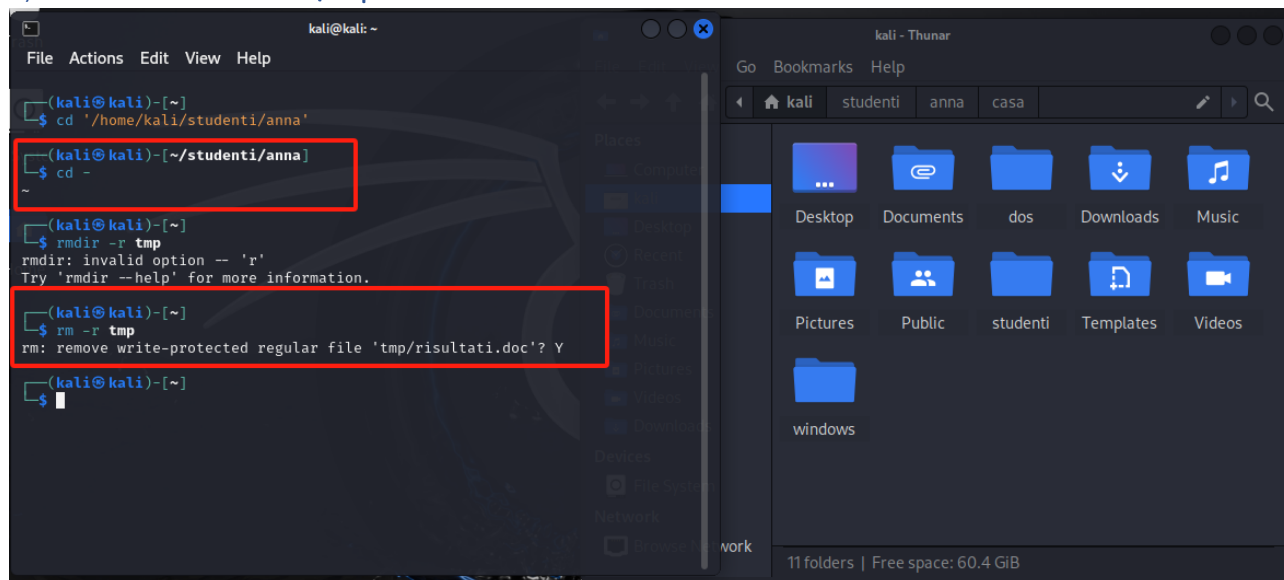
1. Navigare nella cartella studenti/anna/casa;
2. utilizzare il comando per la copia file **`cp [percorso completo con nome file].`** Il punto finale serve per copiare il file 'compito.doc' nella directory in cui ci troviamo attualmente (vedi punto 1);
3. una strada alternativa non proposta dall'esercizio è recarsi nella cartella di partenza quindi **`sudo cp [nome_file] [percorso completo di destinazione]`**

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



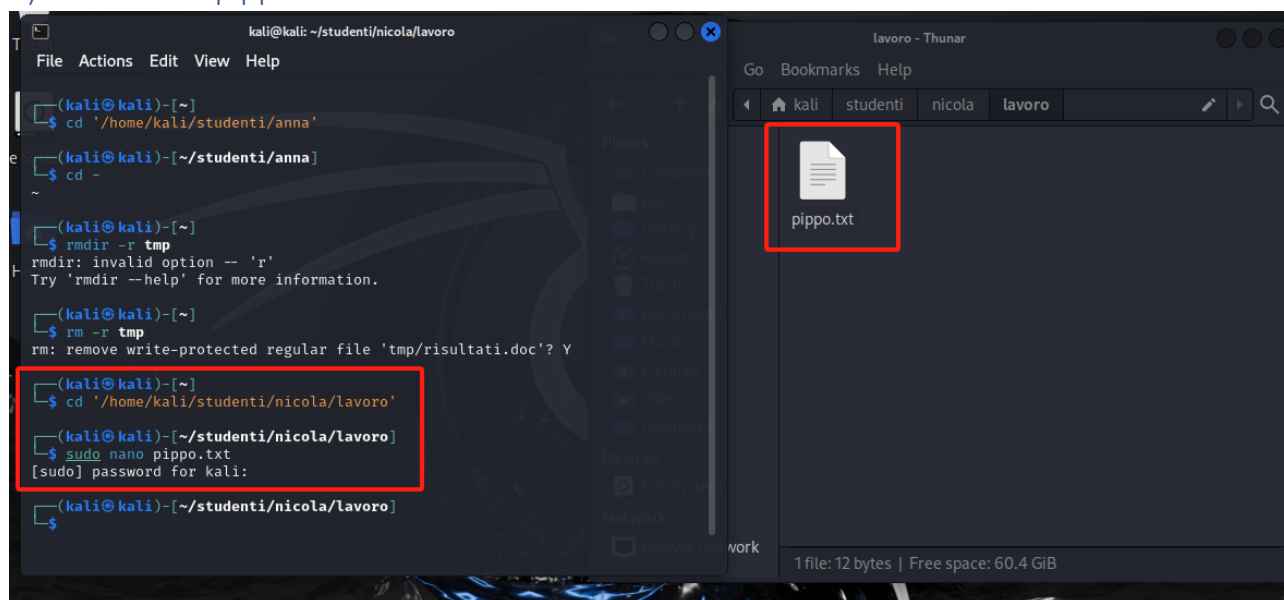
1. Navigare nella cartella studenti/anna/casa;
2. utilizzare il comando per spostare i file **`mv [percorso completo con nome file].`** Il punto finale serve per spostare il file 'relazione.doc' nella directory in cui ci troviamo attualmente (vedi punto 1);
3. una strada alternativa non proposta dall'esercizio è recarsi nella cartella di partenza quindi **`sudo mv [nome_file] [percorso completo di destinazione]`**

c) Cancella la cartella `\tmp`



1. Torniamo nella home con il comando `cd -`;
2. Nella presente versione **2024.2 Kali Linux**, il comando `rmdir` funziona solo ed esclusivamente se la cartella da cancellare sia vuota, altrimenti con il comando **`rmdir -r` non funziona**. Il metodo alternativo per cancellare una cartella non vuota è il comando `rm -r [nome cartella]`, il terminale chiederà se si vuole cancellare il file all'interno, **Y** per confermare, **N** per negare.

d) creare il file `pippo.txt` nella cartella lavoro



1. Recarsi nella cartella lavoro utilizzando il comando del percorso assoluto di `cd`;
2. creare il file con il comando `sudo nano pippo.txt` compilarlo e salvarlo con CTRL+O invio e CTRL+X.

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

```

(kali@kali)-[~]
$ cd '/home/kali/studenti/nicola/lavoro'

(kali@kali)-[~/studenti/nicola/lavoro]
$ sudo nano pippo.txt
[sudo] password for kali:

(kali@kali)-[~/studenti/nicola/lavoro]
$

(kali@kali)-[~/studenti/nicola/lavoro]
$ ls -l pippo.txt
-rw-r--r-- 1 root root 12 Jul 23 18:33 pippo.txt

(kali@kali)-[~/studenti/nicola/lavoro]
$ chmod u=rw,go=r pippo.txt
chmod: changing permissions of 'pippo.txt': Operation not permitted

(kali@kali)-[~/studenti/nicola/lavoro]
$ sudo chmod u=rw,go=r pippo.txt

(kali@kali)-[~/studenti/nicola/lavoro]
$ ls -l pippo.txt
-rw-r--r-- 1 root root 12 Jul 23 18:33 pippo.txt

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

1. Ci rechiamo nella cartella ove è presente il file con il comando `cd`;
2. Verifichiamo i permessi correnti del file con il comando `ls -l pippo.txt`;
3. `-rw-r--r--` questo vuol dire che il file è già in lettura e scrittura per l'utente (proprietario) e per gli altri è `r` e `r` che vuol dire solo lettura, pertanto non sarebbe necessario apportare nessuna modifica;
4. ai fini didattici, per impostarlo come da richiesta il comando sarebbe (da avviare come superuser) `sudo chmod u=rw,go=r pippo.txt` che tradotto sarebbe utente = lettura e scrittura mentre gruppo e altri = lettura per il file pippo.txt.

f) Nascondere il contenuto della cartella anna

```

(kali@kali)-[~/studenti]
$ cd '/home/kali/studenti'

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

(kali@kali)-[~/studenti]
$ ls -a
.  ..  .anna  matteo  nicola

(kali@kali)-[~/studenti]
$
  
```

1. Ci rechiamo nella cartella studenti;
2. utilizziamo il comando `mv anna .anna` per nascondere la cartella anna, nella pratica stiamo spostando la cartella anna nella sezione delle cartelle nascoste;
3. verifichiamo che anche se non visibile, sia lo stesso presente con il comando `ls -a`.

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

```

kali@kali: ~/studenti/nicola/lavoro
File Actions Edit View Help

(kali@kali)-[~]
$ cd '/home/kali/studenti/nicola/lavoro'

(kali@kali)-[~/studenti/nicola/lavoro]
$ cat pippo.txt
Ciao sono Pippo e sono un cane antropomorfo.
Ogni giorno porto a spasso il mio cane di nome Pluto.
Quando diventa grande si chiama Plutone.
Vieni a trovarci nel Mickey Mouse Club House.

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

(kali@kali)-[~/studenti/nicola/lavoro]
$ sudo nano pippo.txt
[sudo] password for kali:

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

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

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

spostarsi nella cartella lavoro

vari metodi possibili

1. utilizzare il comando **cat**;

```

kali@kali: ~/studenti/nicola/lavoro
File Actions Edit View Help

(kali@kali)-[~]
$ cd '/home/kali/studenti/nicola/lavoro'

(kali@kali)-[~/studenti/nicola/lavoro]
$ cat pippo.txt
Ciao sono Pippo e sono un cane antropomorfo.
Ogni giorno porto a spasso il mio cane di nome Pluto.
Quando diventa grande si chiama Plutone.
Vieni a trovarci nel Mickey Mouse Club House.

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

2. utilizzare il comando **less** e per uscire premere **q**;

```

Ciao sono Pippo e sono un cane antropomorfo.
Ogni giorno porto a spasso il mio cane di nome Pluto.
Quando diventa grande si chiama Plutone.
Vieni a trovarci nel Mickey Mouse Club House.
pippo.txt (END)
  
```

3. utilizzare gli editor di testo come da legenda nano, vi o vim.

```

kali@kali: ~/studenti/nicola/lavoro
File Actions Edit View Help

GNU nano 8.0 pippo.txt
Ciao sono Pippo e sono un cane antropomorfo.
Ogni giorno porto a spasso il mio cane di nome Pluto.
Quando diventa grande si chiama Plutone.
Vieni a trovarci nel Mickey Mouse Club House.

~
~
~
  
```

```

kali@kali: ~/studenti/nicola/lavoro
File Actions Edit View Help

Ciao sono Pippo e sono un cane antropomorfo.
Ogni giorno porto a spasso il mio cane di nome Pluto.
Quando diventa grande si chiama Plutone.
Vieni a trovarci nel Mickey Mouse Club House.

~
~
~
  
```

h) Rimuovere la cartella amici

```

kali@kali: ~/students/nicola/lavoro
File Actions Edit View Help
Ciao sono Pippo e sono un cane antropomorfo.
Ogni giorno porto a spasso il mio cane di nome Pluto.
Quando diventa grande si chiama Plutone.
Vieni a trovarci nel Mickey Mouse Club House.
(kali@kali)~/students/nicola/lavoro
$ less pippo.txt
(kali@kali)~/students/nicola/lavoro
$ sudo nano pippo.txt
[sudo] password for kali:
(kali@kali)~/students/nicola/lavoro
$ vim pippo.txt
(kali@kali)~/students/nicola/lavoro
$ vi pippo.txt
(kali@kali)~/students/nicola/lavoro
$ rm /home/kali/students/matteo/amici
rm: cannot remove '/home/kali/students/matteo/amici': Is a directory
(kali@kali)~/students/nicola/lavoro
$ rmdir '/home/kali/students/matteo/amici'
(kali@kali)~/students/nicola/lavoro
$

```

La cartella da cancellare è una cartella vuota. Possiamo cancellarlo con il percorso assoluto come in figura, oppure recarci nel livello antecedente alla cartella da eliminare, livello 3, e digitare **rmdir amici**

i) Rimuovere tutte le cartelle precedentemente create

```

kali@kali: ~
File Actions Edit View Help
$ cd ~
(kali@kali)~
$ ls
Desktop dos Music Public Templates windows
Documents Downloads Pictures studenti Videos
(kali@kali)~
$ rmdir dos
(kali@kali)~
$ rmdir windows
(kali@kali)~
$ rm -r studenti
rm: remove write-protected regular file 'studenti/nicola/scuola/.relazione.doc.swp'? y
rm: remove write-protected regular file 'studenti/nicola/scuola/.relazione.doc.c.swp'? y
rm: remove write-protected regular file 'studenti/nicola/lavoro/pippo.txt'? y
rm: remove write-protected regular file 'studenti/.anna/casa/relazione.doc'? y
rm: remove write-protected regular file 'studenti/.anna/casa/compito.doc'? y
(kali@kali)~
$

```

1. andare alla home con il comando **cd -**;
2. comando **ls** per ottenere la lista delle cartelle e file esistenti;
3. cancelliamo le directory vuote dos e windows con **rmdir**;
4. cancelliamo la directory non vuota studenti con **rm -r**.

Conclusione esercizio 1

Attraverso questo esercizio si impara a gestire completamente le cartelle e i file, nonché a visualizzare e modificare i file di testo, il tutto tramite un'interfaccia non grafica come il terminale di Kali Linux. È fondamentale comprendere la logica di ogni passaggio e tenere sempre presente la struttura dei percorsi e la ramificazione delle cartelle per orientarsi al meglio.

who	lista utenti collegati
who am i	chi sono io
jobs	elenco lavori sul terminale
&	apre processo in background
fg	metti in foreground
bg	metti in background
ps	elenco processi
kill	termina processo

Provare i comandi:

w
who
who am i

Esercizi - processi:

1. Aprire un terminale
2. leggere il manuale del comando job, ps e kill
3. lanciare il comando vi pippo
4. aprire un nuovo terminale e visualizzare tutti i propri processi...
5. cercare di terminare (killare) il processo vi per sbloccare il terminale precedente
6. lanciare il comando firefox in background
7. portarlo in background
8. cercare di terminare il processo firefox
9. verificare quanto spazio si sta occupando su disco

Legenda del terminale sulla gestione dei processi

Lista comandi utili per l'esercizio:

- ✓ **sudo su + comando** il comando viene avviato con privilegi di superuser assoluto
- ✓ **who** lista utenti collegati
- ✓ **who am i** chi sono io
- ✓ **jobs** elenco lavori sul terminale
- ✓ **&** apre processo in background [**nome_processo &**]
- ✓ **fg nome_processo** metti in foreground
- ✓ **bg nome_processo** metti in background
- ✓ **ps** elenco processi
ps aux questo comando mostra tutti i processi in esecuzione sul sistema con dettagli come l'utente, l'ID del processo (PID), la percentuale di utilizzo della CPU e della memoria, e il comando che ha avviato il processo.
- ✓ **top** è un comando interattivo che fornisce una visualizzazione in tempo reale dei processi in esecuzione, inclusi l'utilizzo della CPU e della memoria
- ✓ **kill + PID** termina processo
 opzione **kill -9 + PID** indica il segnale "SIGKILL" per forzare l'interruzione immediata senza possibilità di pulizia o salvare lo stato
- ✓ **killall nome_processo** serve per killare tutti i processi con quel nome
- ✓ **egrep** extended grep serve per filtrare più file
 esempio **egrep 'error|warning' server.log**: Cerca tutte le righe nel file server.log che contengono "error" o "warning". Il simbolo | è utilizzato come operatore OR nelle espressioni regolari estese.
- ✓ **pgrep + nome processo** Trova gli ID dei processi che corrispondono a "nome_processo".
- ✓ **df**: Mostra l'utilizzo del disco per tutti i file system montati, con i valori in blocchi (in base a 1K).
 - a. **-h**: Mostra l'uso del disco in un formato leggibile dall'uomo (ad es. GB, MB).
 - b. **-T**: Mostra anche il tipo di file system.
- **du** Mostra l'uso del disco per la directory corrente e le sue sottodirectory in blocchi
 - a. **-h**: Mostra l'uso del disco in un formato leggibile dall'uomo.

Provare i comandi **w** / **who** / **who am i**

```

kali@kali: ~
File Actions Edit View Help

(kali@kali)-[~]
$ w
19:55:54 up 4:29, 1 user, load average: 0.16, 0.08, 0.03
USER      TTY      FROM          LOGIN@   IDLE   JCPU   PCPU WHAT
kali      tty7            16:56      2:31m   0.00s   0.04s lightdm --se

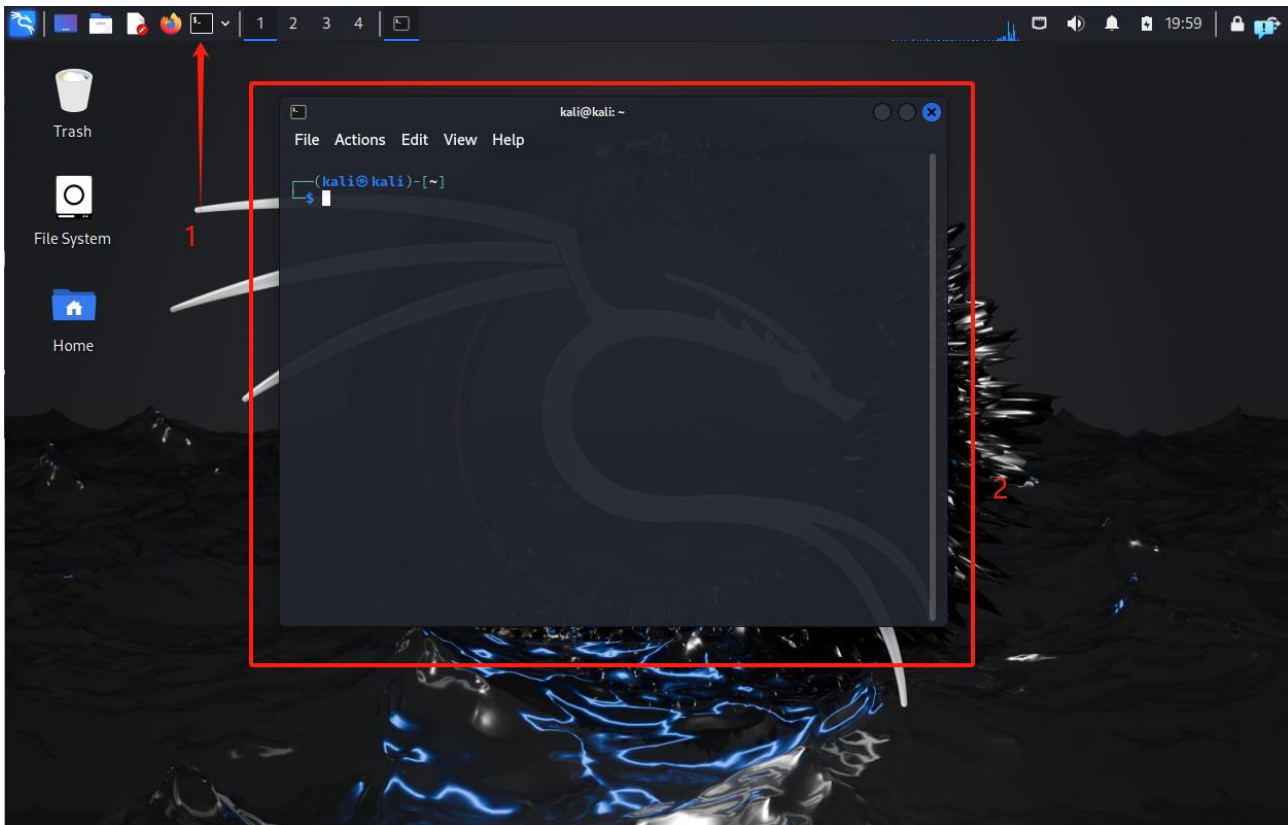
(kali@kali)-[~]
$ who
kali      tty7            2024-07-23 16:56 (:0)

(kali@kali)-[~]
$ whoami
kali

(kali@kali)-[~]
$
  
```

La prima cosa che si nota è che il comando **w** è quella che ritorna più informazioni relativa all'utente, mentre **who** restituisce 3 informazi e **whoami** solo il nome utente.

1. Aprire un terminale



2. Leggere il manuale del comando jobs, ps e kill

Non esiste una pagina di manuale separata per il comando jobs perché jobs è un comando built-in della shell, non un comando esterno. Le pagine di manuale di solito riguardano comandi esterni installati nel sistema, ma i comandi built-in delle shell hanno la loro documentazione integrata nella documentazione della shell stessa.

```
kali@kali: ~  
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 it-  
self 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  
Manual page kill(1) line 1 (press h for help or q to quit)
```

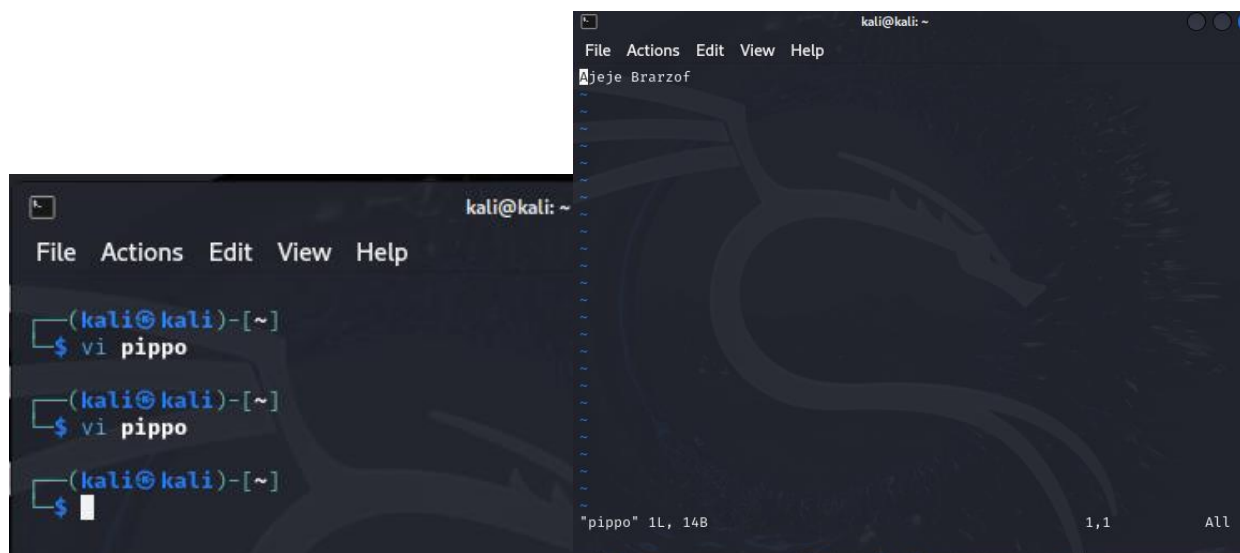
```
kali@kali: ~  
File Actions Edit View Help  
PS(1) User Commands PS(1)  
  
NAME  
ps - report a snapshot of the current processes.  
  
SYNOPSIS  
ps [options]  
  
DESCRIPTION  
ps displays information about a selection of the active processes.  
If you want a repetitive update of the selection 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.  
  
Manual page ps(1) line 1 (press h for help or q to quit)
```

Per ps e kill si inseriscono i rispettivi comandi **man ps** e **man kill**.

```
(kali@kali)-[~]  
$ man ps  
  
(kali@kali)-[~]  
$ man kill
```

3. Lanciare il comando vi pippo

Il comando **vi pippo** non genera un processo separato visibile in ps o top nel senso tradizionale perché vi è un editor che interagisce direttamente con il terminale.



```
kali@kali: ~
File Actions Edit View Help
Mjeje Brarzoj

kali@kali: ~
File Actions Edit View Help

(kali@kali)-[~]
$ vi pippo

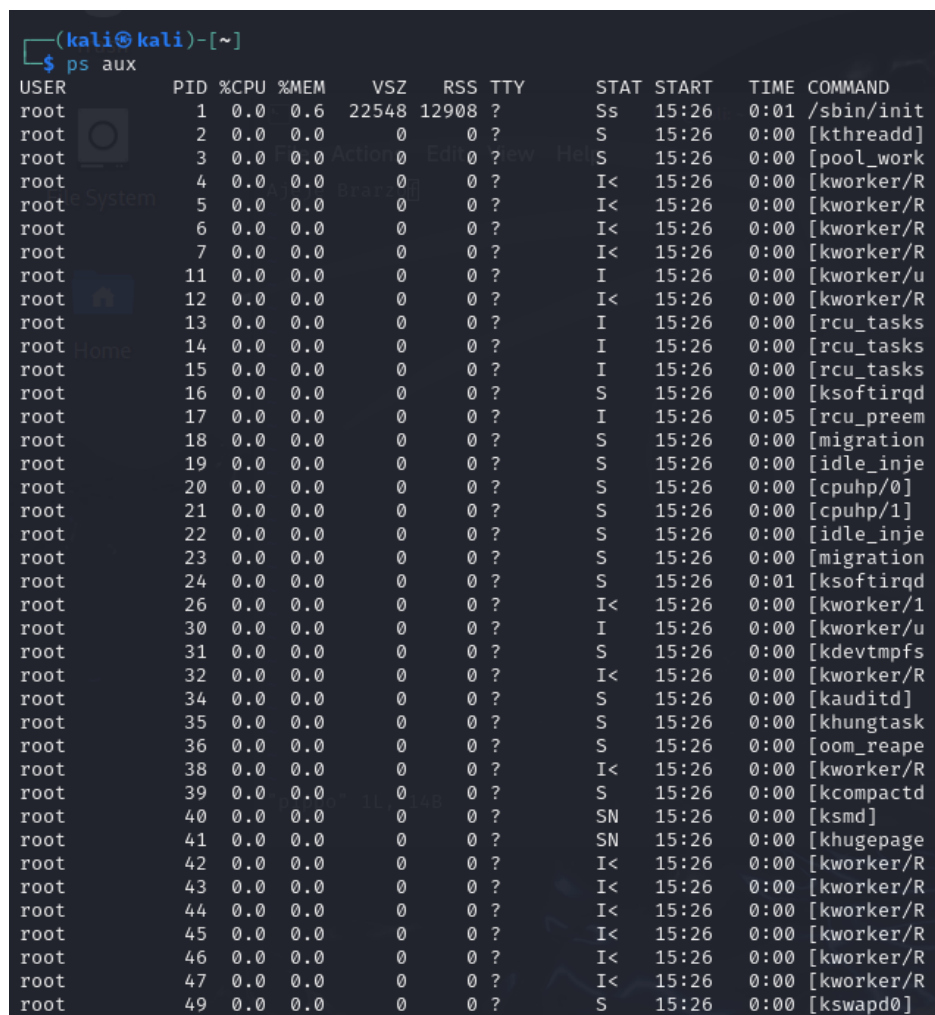
(kali@kali)-[~]
$ vi pippo

(kali@kali)-[~]
$

"pippo" 11, 14B
1,1 All
```

4. Aprire un nuovo terminale e visualizzare tutti i processi...

Comando **ps aux**



```
(kali@kali)-[~]
$ ps aux
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.0	0.6	22548	12908	?	Ss	15:26	0:01	/sbin/init
root	2	0.0	0.0	0	0	?	S	15:26	0:00	[kthreadd]
root	3	0.0	0.0	0	0	?	S	15:26	0:00	[pool_work
root	4	0.0	0.0	0	0	?	I<	15:26	0:00	[kworker/R
root	5	0.0	0.0	0	0	?	I<	15:26	0:00	[kworker/R
root	6	0.0	0.0	0	0	?	I<	15:26	0:00	[kworker/R
root	7	0.0	0.0	0	0	?	I<	15:26	0:00	[kworker/R
root	11	0.0	0.0	0	0	?	I	15:26	0:00	[kworker/u
root	12	0.0	0.0	0	0	?	I<	15:26	0:00	[kworker/R
root	13	0.0	0.0	0	0	?	I	15:26	0:00	[rcu_tasks
root	14	0.0	0.0	0	0	?	I	15:26	0:00	[rcu_tasks
root	15	0.0	0.0	0	0	?	I	15:26	0:00	[rcu_tasks
root	16	0.0	0.0	0	0	?	S	15:26	0:00	[ksoftirqd
root	17	0.0	0.0	0	0	?	I	15:26	0:05	[rcu_prem
root	18	0.0	0.0	0	0	?	S	15:26	0:00	[migration
root	19	0.0	0.0	0	0	?	S	15:26	0:00	[idle_inje
root	20	0.0	0.0	0	0	?	S	15:26	0:00	[cpuhp/0]
root	21	0.0	0.0	0	0	?	S	15:26	0:00	[cpuhp/1]
root	22	0.0	0.0	0	0	?	S	15:26	0:00	[idle_inje
root	23	0.0	0.0	0	0	?	S	15:26	0:00	[migration
root	24	0.0	0.0	0	0	?	S	15:26	0:01	[ksoftirqd
root	26	0.0	0.0	0	0	?	I<	15:26	0:00	[kworker/1
root	30	0.0	0.0	0	0	?	I	15:26	0:00	[kworker/u
root	31	0.0	0.0	0	0	?	S	15:26	0:00	[kdevtmpfs
root	32	0.0	0.0	0	0	?	I<	15:26	0:00	[kworker/R
root	34	0.0	0.0	0	0	?	S	15:26	0:00	[kauditd]
root	35	0.0	0.0	0	0	?	S	15:26	0:00	[khungtask
root	36	0.0	0.0	0	0	?	S	15:26	0:00	[oom_reape
root	38	0.0	0.0	0	0	?	I<	15:26	0:00	[kworker/R
root	39	0.0	0.0	0	0	?	S	15:26	0:00	[kcompactd
root	40	0.0	0.0	0	0	?	SN	15:26	0:00	[ksmd]
root	41	0.0	0.0	0	0	?	SN	15:26	0:00	[khugepage
root	42	0.0	0.0	0	0	?	I<	15:26	0:00	[kworker/R
root	43	0.0	0.0	0	0	?	I<	15:26	0:00	[kworker/R
root	44	0.0	0.0	0	0	?	I<	15:26	0:00	[kworker/R
root	45	0.0	0.0	0	0	?	I<	15:26	0:00	[kworker/R
root	46	0.0	0.0	0	0	?	I<	15:26	0:00	[kworker/R
root	47	0.0	0.0	0	0	?	I<	15:26	0:00	[kworker/R
root	49	0.0	0.0	0	0	?	S	15:26	0:00	[kswapd0]

Comando top

```

kali@kali: ~
File Actions Edit View Help
top - 20:27:52 up 5:01, 1 user, load average: 0.01, 0.04, 0.01
Tasks: 161 total, 1 running, 160 sleeping, 0 stopped, 0 zombie
%Cpu(s): 1.0 us, 0.9 sy, 0.0 ni, 98.1 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 1976.7 total, 777.5 free, 860.3 used, 485.8 buff/cache
MiB Swap: 1024.0 total, 1024.0 free, 0.0 used, 1116.4 avail Mem

  PID USER      PR  NI  VIRT  RES  SHR S %CPU  %MEM    TIME+
664  root        20   0 390468 111760 54156 S  2.0   5.5   1:36.76
1185 kali        20   0 969268 122368 76360 S  0.7   6.0   0:36.61
104852 kali      20   0 453088 97920 82896 S  0.7   4.8   0:00.48
1211 kali        20   0 215552 3712 3200 S  0.3   0.2   0:02.77
1249 kali        20   0 363316 53312 22716 S  0.3   2.6   0:22.13
1300 kali        20   0 264572 26336 17280 S  0.3   1.3   0:00.57
81452 root        20   0 0 0 0 I  0.3   0.0   0:00.99
1  root        20   0 22548 12908 9708 S  0.0   0.6   0:01.92
2  root        20   0 0 0 0 S  0.0   0.0   0:00.04
3  root        20   0 0 0 0 S  0.0   0.0   0:00.00
4  root        0 -20 0 0 0 I  0.0   0.0   0:00.00
5  root        0 -20 0 0 0 I  0.0   0.0   0:00.00
6  root        0 -20 0 0 0 I  0.0   0.0   0:00.00
7  root        0 -20 0 0 0 I  0.0   0.0   0:00.00
11 root        20   0 0 0 0 I  0.0   0.0   0:00.00
12 root        0 -20 0 0 0 I  0.0   0.0   0:00.00
13 root        20   0 0 0 0 I  0.0   0.0   0:00.00
14 root        20   0 0 0 0 I  0.0   0.0   0:00.00
15 root        20   0 0 0 0 I  0.0   0.0   0:00.00
16 root        20   0 0 0 0 S  0.0   0.0   0:00.87

```

Comando ps

```

(kali@kali)-[~]
$ ps
  PID TTY          TIME CMD
 104855 pts/1        00:00:00 zsh
 107245 pts/1        00:00:00 ps

```

5. Cercare di terminare (killare) il processo vi per sbloccare il terminale precedente

Apriamo da terminale col il comando **open pippo** e Kali avvierà il programma di lettura testo “Mousepad”. Di conseguenza cerchiamo il PID utilizzando la funzione **pgrep mousepad** per trovare il PID del processo.

```

~/pippo - Mousepad
File Edit Search View Document
1 Ajeje Brarzo
2
(kali@kali)-[~]
$ pgrep pippo
(kali@kali)-[~]
$ pgrep mousepad
27651
(kali@kali)-[~]
$ pippo
pippo: command not found
(kali@kali)-[~]
$ open pippo
(kali@kali)-[~]
$ open pippo
(kali@kali)-[~]
$ grep mousepad
^C
(kali@kali)-[~]
$ pgrep mousepad
28273
(kali@kali)-[~]
$

```

Lo terminiamo con il comando **kill -9 28273** (il pid di mousepad) che verrà chiuso all'istante.

6. Lanciare il comando firefox in background

```

kali@kali: ~
File Actions Edit View Help

(kali@kali)-[~]
$ firefox &
[1] 34691

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

(kali@kali)-[~]
$ man jobs
No manual entry for jobs

(kali@kali)-[~]
$ bg firefox
bg: job already in background

(kali@kali)-[~]
$

```

Avviamo firefox in background con il comando `&` dopo il nome processo, per verificare dell'avvenuta apertura del processo utilizziamo la funzione **jobs**.

7. portarlo in background

Come da figura nel punto precedente, il comando **bg firefox** per portarlo in background non funziona in quanto è già in background.

8. cercare di terminare il processo firefox

```

kali@kali: ~
File Actions Edit View Help

(kali@kali)-[~]
$ firefox &
[1] 38395

(kali@kali)-[~]
$ pgrep firefox
38395

(kali@kali)-[~]
$ kill 38395

(kali@kali)-[~]
$
Exiting due to channel error.
Exiting due to channel error.
[GTK1-]: CompositorBridgeChild receives IPC close with reason=AbnormalShutdown
Exiting due to channel error.
Exiting due to channel error.
Exiting due to channel error.
Exiting due to channel error.

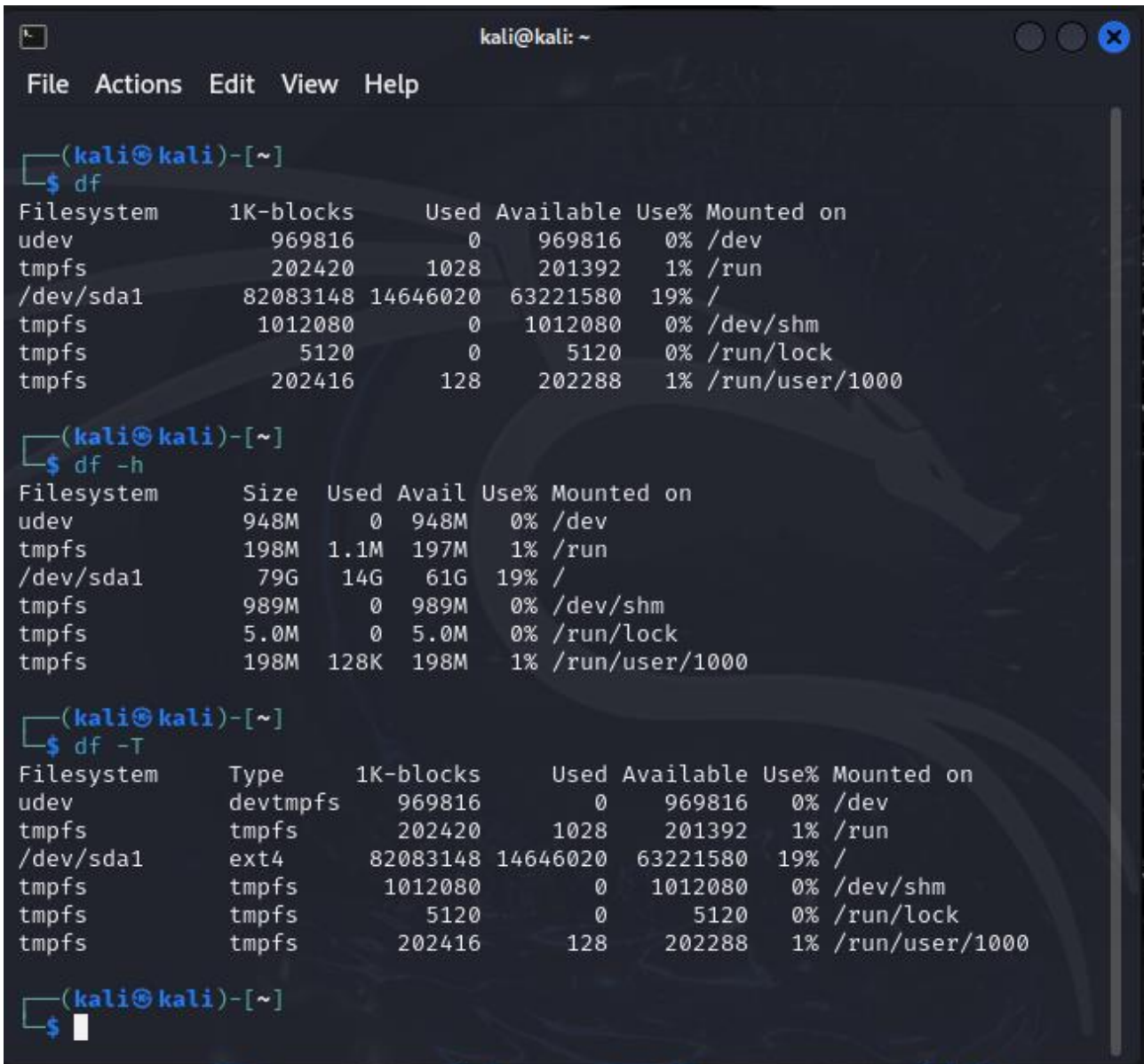
[1] + terminated firefox
(kali@kali)-[~]
$

```

Il PID di firefox è 38395 in questo caso e per terminare è stato usato il comando semplice **kill 38395**. Ignoriamo gli errori, in quanto è un avviso di firefox che è stato terminato in modo anomalo.

9. verificare quanto spazio si sta occupando su disco

21



```
kali@kali: ~  
File Actions Edit View Help  
  
(kali@kali)-[~]  
$ df  
Filesystem      1K-blocks    Used Available Use% Mounted on  
udev            969816         0   969816  0% /dev  
tmpfs           202420      1028   201392  1% /run  
/dev/sda1       82083148 14646020 63221580 19% /  
tmpfs           1012080         0   1012080  0% /dev/shm  
tmpfs            5120         0     5120  0% /run/lock  
tmpfs           202416       128   202288  1% /run/user/1000  
  
(kali@kali)-[~]  
$ df -h  
Filesystem      Size  Used Avail Use% Mounted on  
udev            948M   0    948M   0% /dev  
tmpfs           198M  1.1M  197M   1% /run  
/dev/sda1       79G   14G   61G   19% /  
tmpfs           989M   0    989M   0% /dev/shm  
tmpfs           5.0M   0    5.0M   0% /run/lock  
tmpfs           198M  128K  198M   1% /run/user/1000  
  
(kali@kali)-[~]  
$ df -T  
Filesystem      Type      1K-blocks    Used Available Use% Mounted on  
udev            devtmpfs  969816         0   969816  0% /dev  
tmpfs           tmpfs     202420      1028   201392  1% /run  
/dev/sda1       ext4      82083148 14646020 63221580 19% /  
tmpfs           tmpfs     1012080         0   1012080  0% /dev/shm  
tmpfs           tmpfs      5120         0     5120  0% /run/lock  
tmpfs           tmpfs     202416       128   202288  1% /run/user/1000  
  
(kali@kali)-[~]  
$
```

Utilizziamo i comandi **df**, **df-h** e **df-T**

Conclusione esercizio 2

Attraverso questo esercizio si impara a gestire completamente i processi, anche in tempo reale, tramite un'interfaccia non grafica come il terminale di Kali Linux.