Introduzione

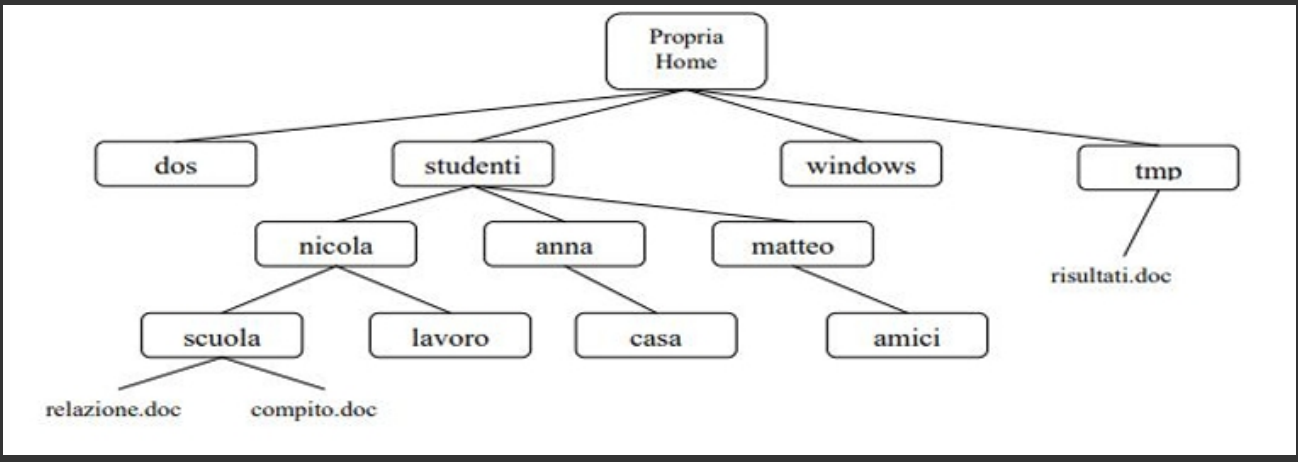
In questo esercizio andremo a fare pratica con alcuni comandi dello Shell di Linux, imparando a muoverci bene all'interno delle directory del nostro sistema operativo.

Creare la struttura

Per prima cosa si andrà a creare la struttura composta da più directory e file nella nostra home.
Ho creato il tutto seguendo questi comandi:

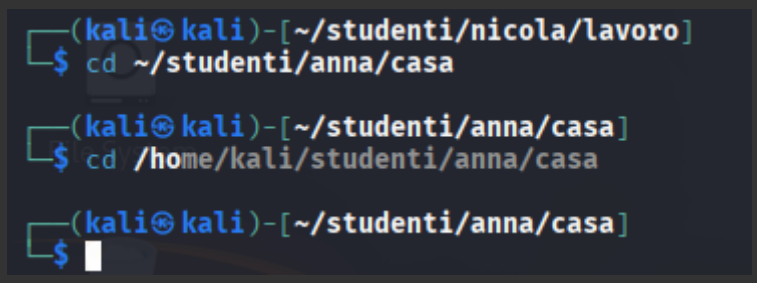
mkdir dos studenti windows tmp	creo nella Home le 4 cartelle
cd ~/tmp	entro nella cartella tmp (situata in home)
touch risultati.doc	creo risultati.doc
cd ~/studenti	entro nella cartella studenti
mkdir nicola anna matteo	all'interno di studenti creo altre 3 cartelle
cd ~/studenti/matteo	entro nella cartella matteo
mkdir amici	creo un altra cartella al suo interno
cd ~/studenti/anna	entro nella cartella anna
mkdir casa	creo la cartella casa al suo interno
cd ~/studenti/nicola	entro nella cartella nicola
mkdir scuola lavoro	creo le cartelle scuola e lavoro
cd ~/studenti/nicola/scuola	entro nella cartella scuola
touch relazione.doc compito.doc	creo i due file
ls	utilizzo ls per vedere cosa contiene la dir.
Cd ~	ritorno alla home

Fatto questo, avremo la seguente struttura:

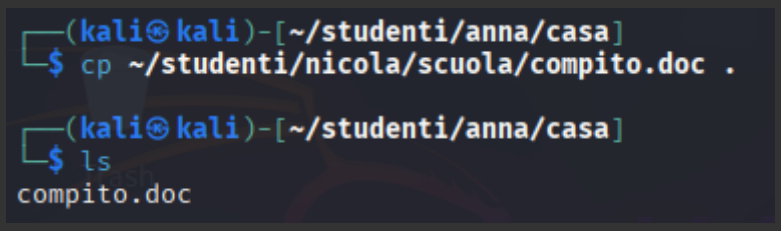


Esercizio

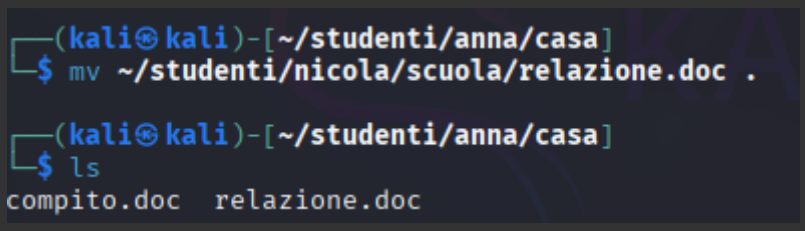
Trovandomi nella directory lavoro mi sono spostato nella directory casa



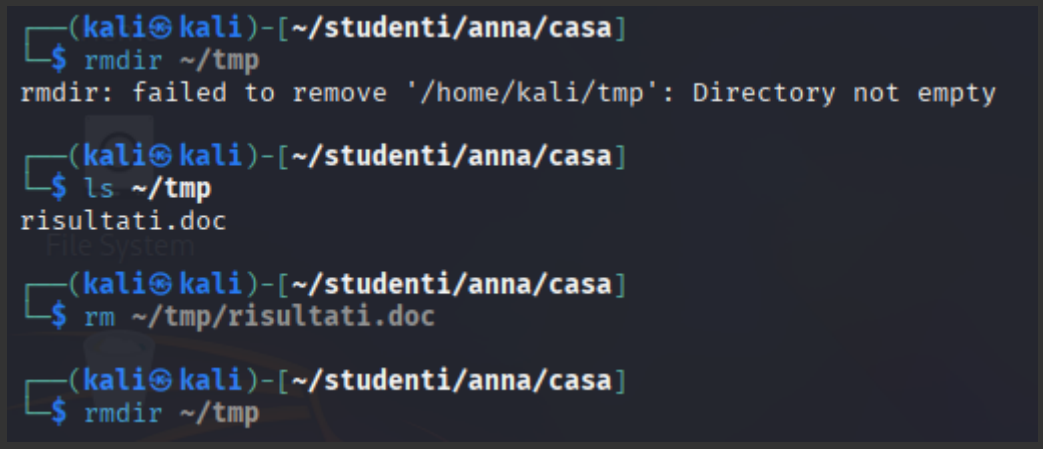
Ho copiato il file compito.doc nella dir in cui mi trovato, ovvero casa, aggiungendo un . alla fine del comando



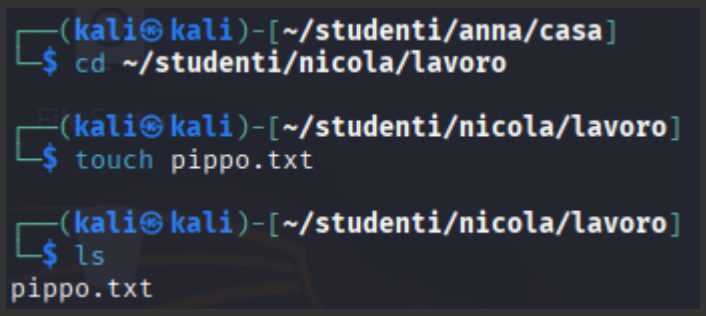
Ho spostato il file relazione.doc nella dir corrente



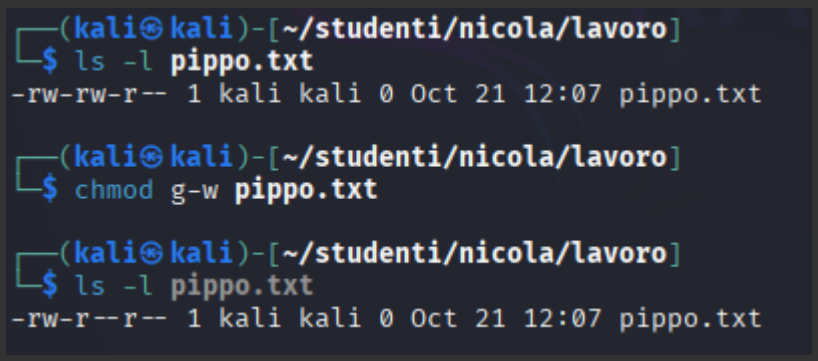
Ho cercato di eliminare la cartella tmp, ma essendoci un file all'interno non me lo ha permesso. In questo caso o possiamo prima eliminare i file al suo interno come ho fatto ora, oppure eseguire il comando rm -r ~ /tmp



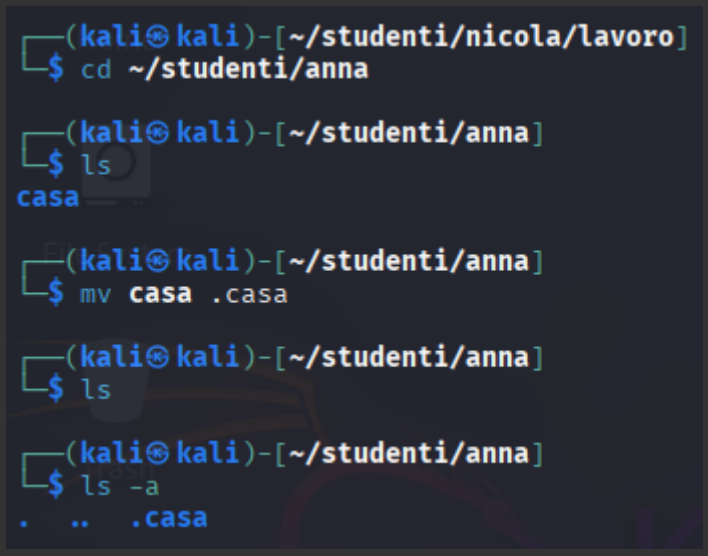
Mi sono spostato nella cartella lavoro, ed ho creato il file pippo.txt



Ho visualizzato gli attributi del file, e li ho cambiati rendendoli scrivibili e leggibili solo dal proprietatio, e per gli altri solo leeggibili

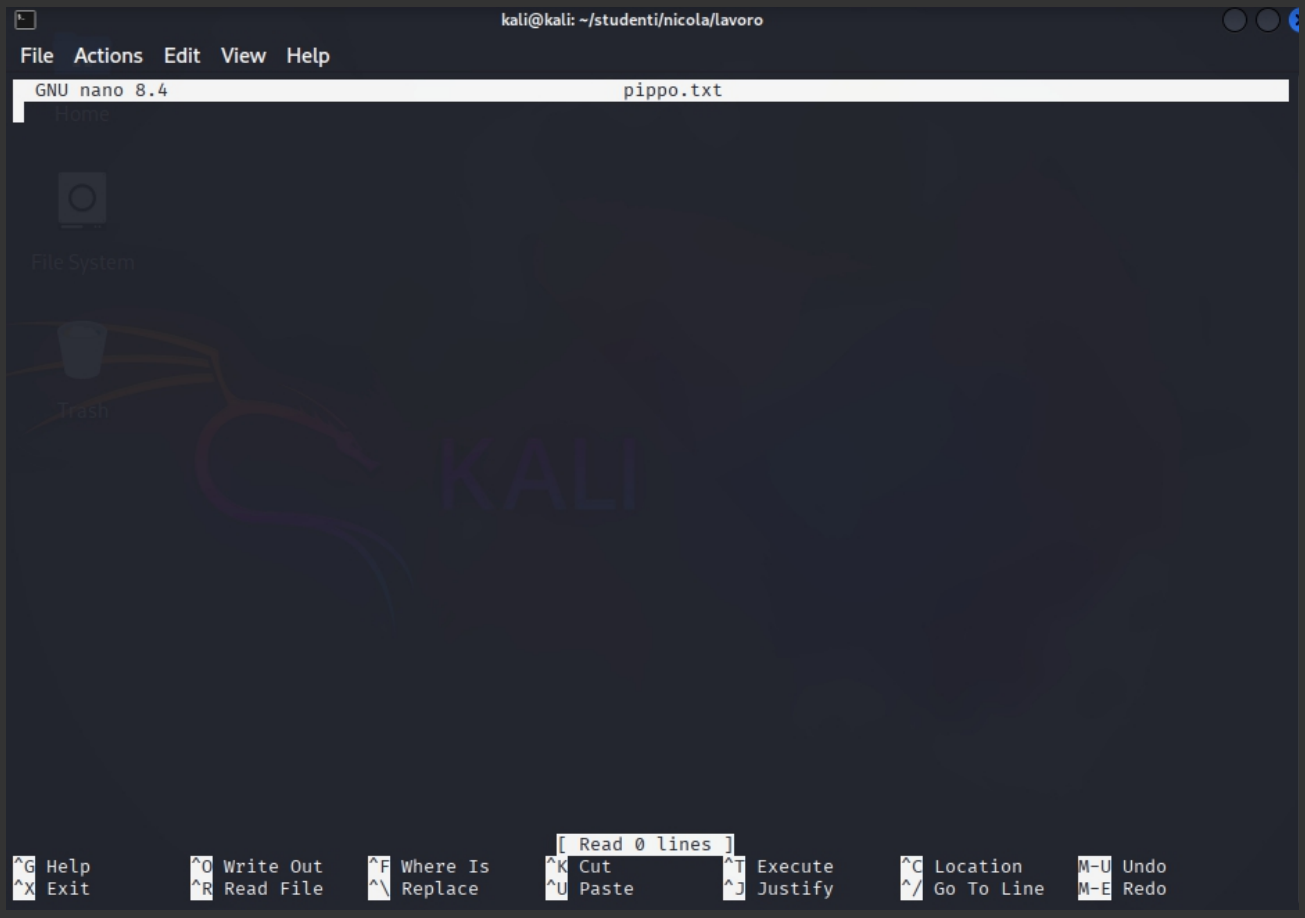


Ho nascosto il contenuto della cartella anna, aggiungendo un . prima del file



Mi sono spostato nella cartella lavoro, e visualizzato il contenuto del file pippo.txt tramite un editor di testo (nano)

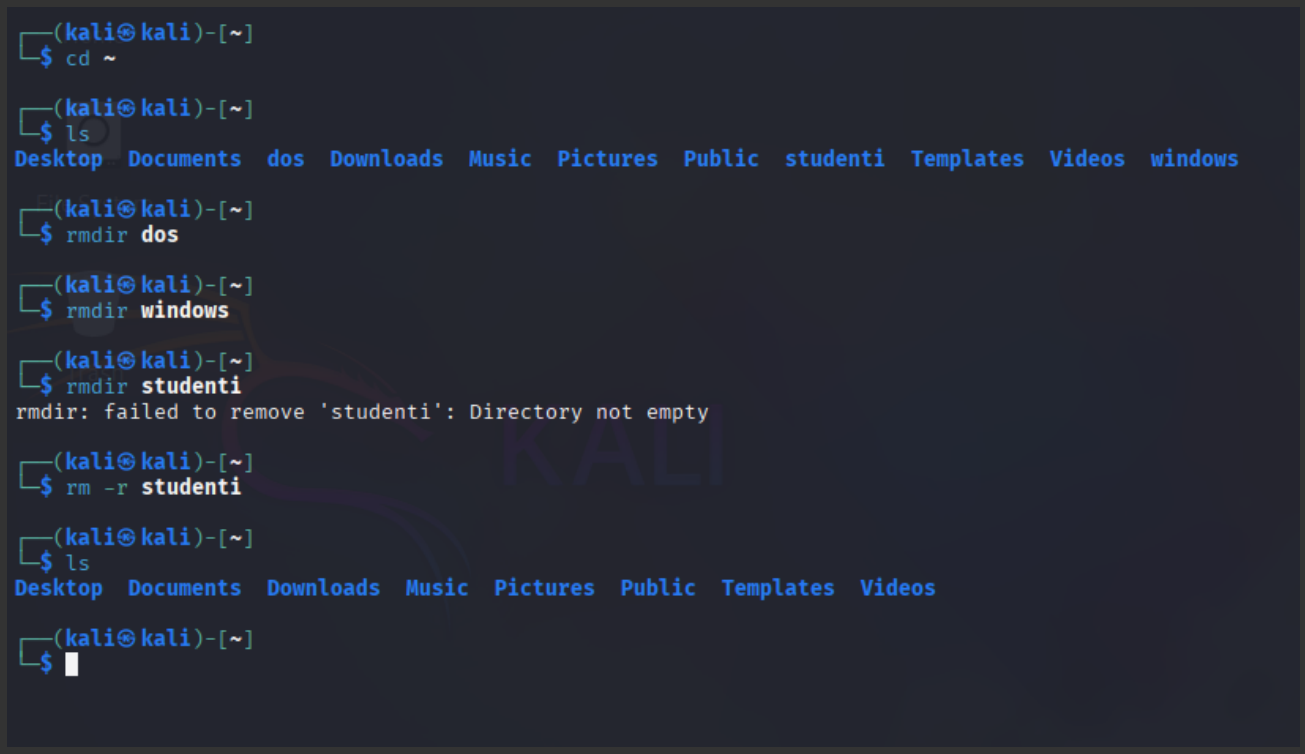
```
┌──(kali㉿kali)-[~/studenti/nicola/lavoro]
└─$ cd ~/studenti/nicola/lavoro

┌──(kali㉿kali)-[~/studenti/nicola/lavoro]
└─$ nano pippo.txt
```



Ho rimosso la cartella amici

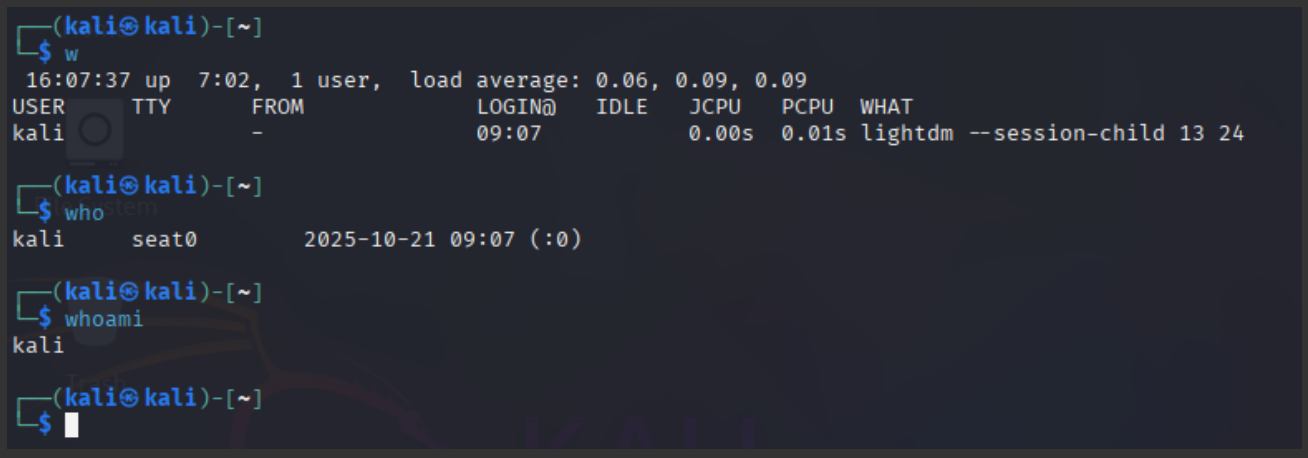
```
┌──(kali㉿kali)-[~/studenti/nicola/lavoro]
└─$ rmdir ~/studenti/matteo/amici

┌──(kali㉿kali)-[~/studenti/nicola/lavoro]
└─$ ls ~/studenti/matteo/amici
ls: cannot access '/home/kali/studenti/matteo/amici': No such file or directory

┌──(kali㉿kali)-[~/studenti/nicola/lavoro]
└─$ ls ~/studenti/matteo/
```

Infine ho eliminato tutte le altre cartelle precedentemente create



Facoltativo

Il comando w mostra quali utenti sono collegati, orario di login e altre informazione, who mostra solo chi è connesso al sistema, mentre whoami indica quale sto effettivamente usando nella sessione attuale.

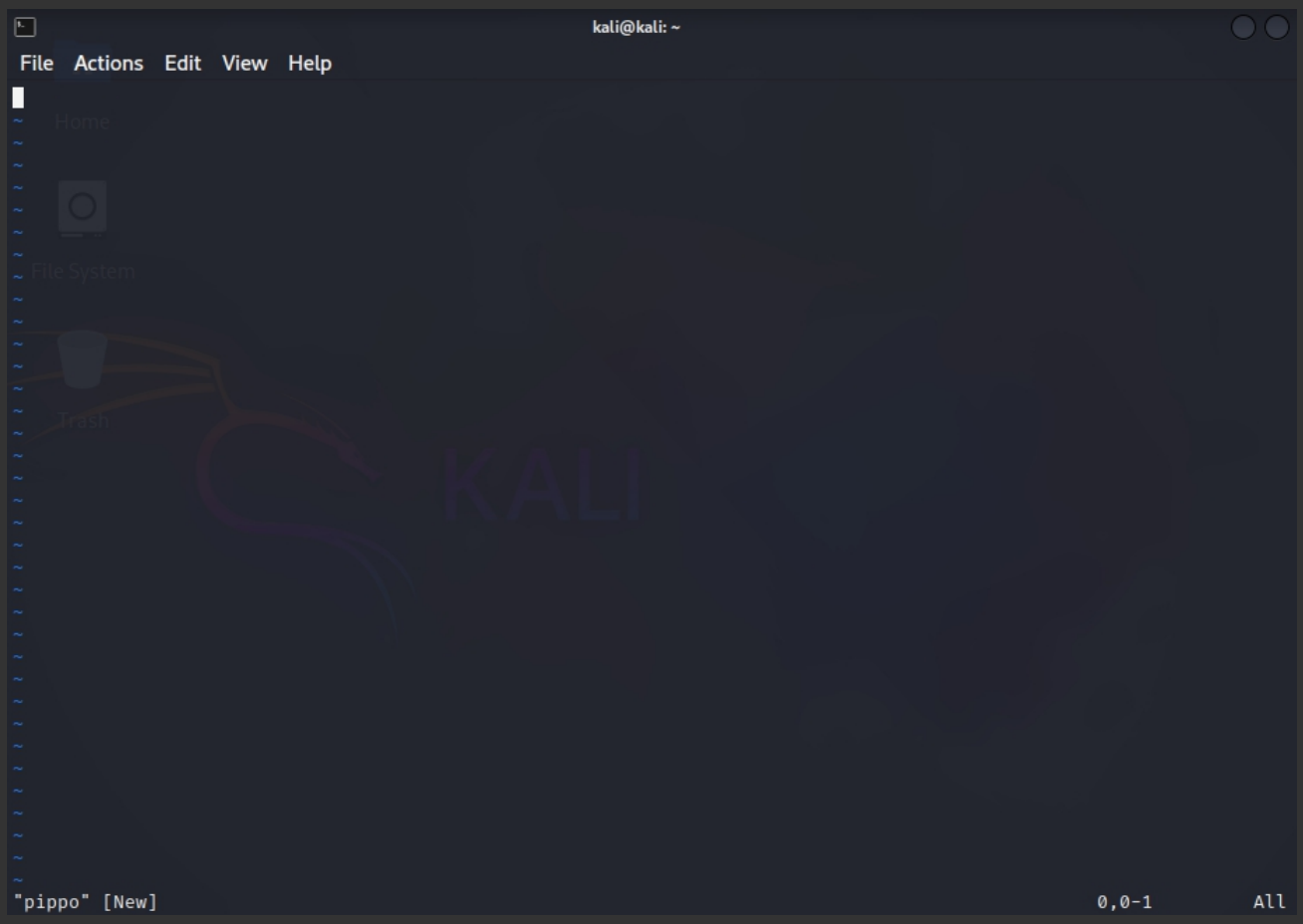


Qui tramite il comando man possiamo vedere il manuale dei comandi, come in questo caso del comando ps e kill

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

```
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 com-
       mand 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  sigqueue(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_SIG-
              INFO  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.
Manual page kill(1) line 1 (press h for help or q to quit)
```

Ora lanciamo il comando vi pippo, aprendo un file tramite editor di testo

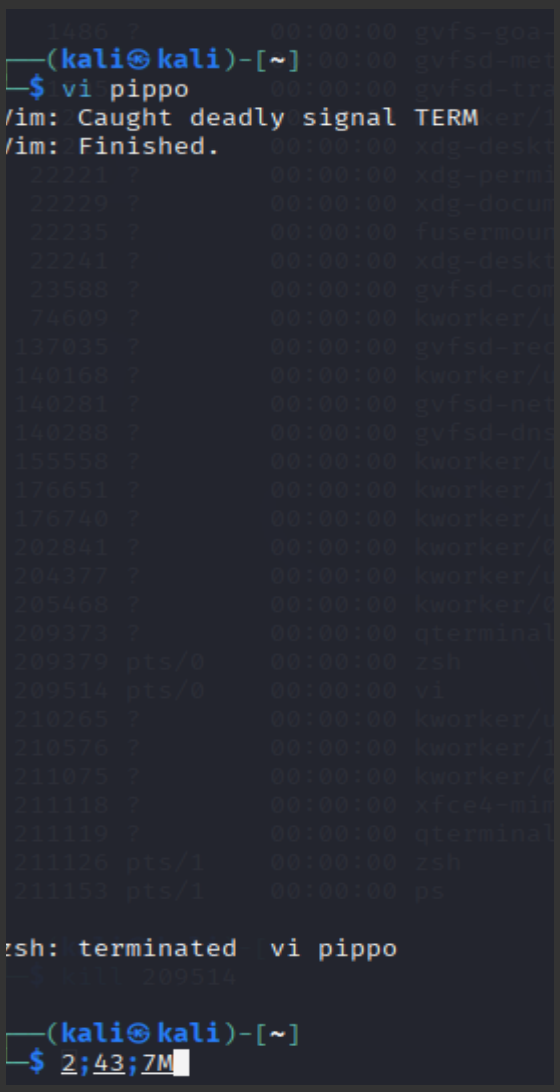


Tramite il comando ps -e possiamo vedere tutti i processi attivi nel sistema, e a quale PID sono associati, tra cui il terminale con cui abbiamo lanciato il comando vi pippo.

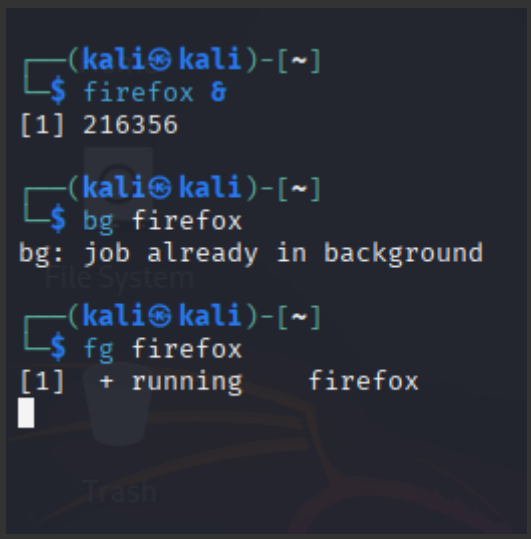
```
┌──(kali㉿kali)-[~]
└─$ ps -e
    PID TTY          TIME CMD
      1 ?        00:00:04 systemd
      2 ?        00:00:00 kthreadd
      3 ?        00:00:00 pool_workqueue_release
      4 ?        00:00:00 kworker/R-kvfree_rcu_reclaim
      5 ?        00:00:00 kworker/R-rcu_gp
      6 ?        00:00:00 kworker/R-sync_wq
      7 ?        00:00:00 kworker/R-slub_flushwq
      8 ?        00:00:00 kworker/R-netns
     11 ?        00:00:01 kworker/0:0H-kblockd
     12 ?        00:00:00 kworker/u8:0-ipv6_addrconf
     13 ?        00:00:00 kworker/R-mm_percpu_wq
     14 ?        00:00:00 rcu_tasks_kthread
     15 ?        00:00:00 rcu_tasks_rude_kthread
     16 ?        00:00:00 rcu_tasks_trace_kthread
     17 ?        00:00:02 ksoftirqd/0
     18 ?        00:00:09 rcu_preempt
     19 ?        00:00:00 rcu_exp_par_gp_kthread_worker/0
     20 ?        00:00:00 rcu_exp_gp_kthread_worker
     21 ?        00:00:00 migration/0
     22 ?        00:00:00 idle_inject/0
     23 ?        00:00:00 cpuhp/0
     24 ?        00:00:00 cpuhp/1
     25 ?        00:00:00 idle_inject/1
     26 ?        00:00:02 migration/1
     27 ?        00:00:02 ksoftirqd/1
     29 ?        00:00:00 kworker/1:0H-kblockd
     31 ?        00:00:00 kworker/u10:0-events_unbound
     35 ?        00:00:00 kdevtmpfs
     36 ?        00:00:00 kworker/R-inet_frag_wq
     37 ?        00:00:00 kauditd
     38 ?        00:00:00 khungtaskd
     39 ?        00:00:00 oom_reaper
     41 ?        00:00:00 kworker/R-writeback
     42 ?        00:00:02 kcompactd0
```

```
   1486 ?        00:00:00 gvfs-goa-volume
   1501 ?        00:00:00 gvfsd-metadata
   1505 ?        00:00:00 gvfsd-trash
  16205 ?        00:00:03 kworker/1:2-events
  22215 ?        00:00:00 xdg-desktop-por
  22221 ?        00:00:00 xdg-permission-
  22229 ?        00:00:00 xdg-document-po
  22235 ?        00:00:00 fusermount3
  22241 ?        00:00:00 xdg-desktop-por
  23588 ?        00:00:00 gvfsd-computer
  74609 ?        00:00:00 kworker/u9:3-events_unbound
 137035 ?        00:00:00 gvfsd-recent
 140168 ?        00:00:00 kworker/u10:4-events_unbound
 140281 ?        00:00:00 gvfsd-network
 140288 ?        00:00:00 gvfsd-dnssd
 155558 ?        00:00:00 kworker/u9:0-events_unbound
 176651 ?        00:00:00 kworker/1:1-events
 176740 ?        00:00:00 kworker/u9:1-events_unbound
 202841 ?        00:00:00 kworker/0:1-ata_sff
 204377 ?        00:00:00 kworker/u10:1-flush-8:0
 205468 ?        00:00:00 kworker/0:0-ata_sff
 209373 ?        00:00:00 qterminal
 209379 pts/0    00:00:00 zsh
 209514 pts/0    00:00:00 vi
 210265 ?        00:00:00 kworker/u9:2-flush-8:0
 210576 ?        00:00:00 kworker/1:0-events
 211075 ?        00:00:00 kworker/0:2-events
 211118 ?        00:00:00 xfce4-mime-help
 211119 ?        00:00:00 qterminal
 211126 pts/1    00:00:00 zsh
 211153 pts/1    00:00:00 ps

┌──(kali㉿kali)-[~]
└─$ kill 209514
```

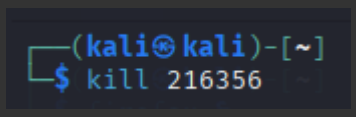
Una volta individuato il PID associato al comando vi possiamo chiuderlo facendo kill vi.

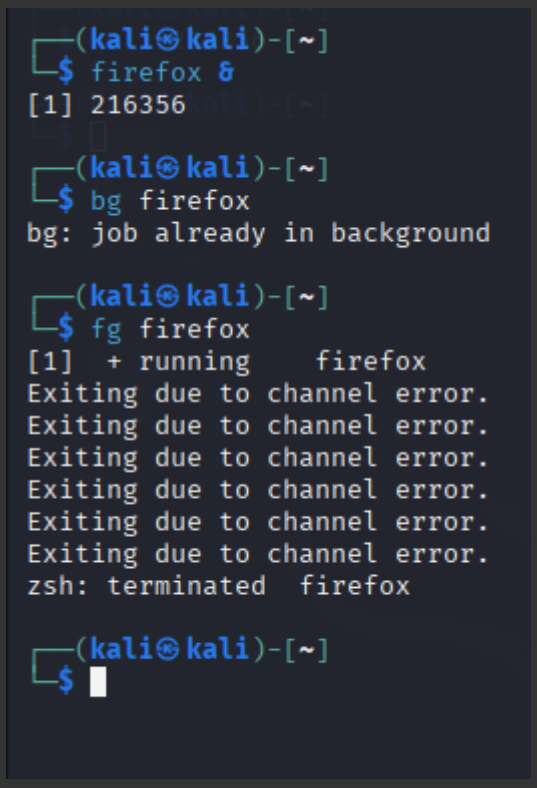


Ora apriamo Firefox in background, eseguendolo con Firefox &. In questo caso è in background, ed anche lanciando il comando bg Firefox questo ci comunica che è già in background. Ora passiamolo in forground lanciando fg Firefox.



Per terminare il processo eseguiamo sempre kill e il PID, questa volta fornito quando abbiamo lanciato il comando in background.





Infine, verifichiamo quanto spazio sul disco si sta occupando lanciando il comando df -h

```
(kali㉿kali)-[~]
$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            1.4G     0  1.4G   0% /dev
tmpfs           298M  972K  297M   1% /run
/dev/sda1        79G   17G   59G  22% /
tmpfs           1.5G  4.0K  1.5G   1% /dev/shm
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           1.0M     0  1.0M   0% /run/credentials/systemd-journald.service
tmpfs           1.5G  8.0K  1.5G   1% /tmp
tmpfs           1.0M     0  1.0M   0% /run/credentials/getty@tty1.service
tmpfs           298M  124K  298M   1% /run/user/1000
```

Visual Studio Code

Qui invece si conferma l'installazione del programma Visual Studio Code sulla macchina Kali.