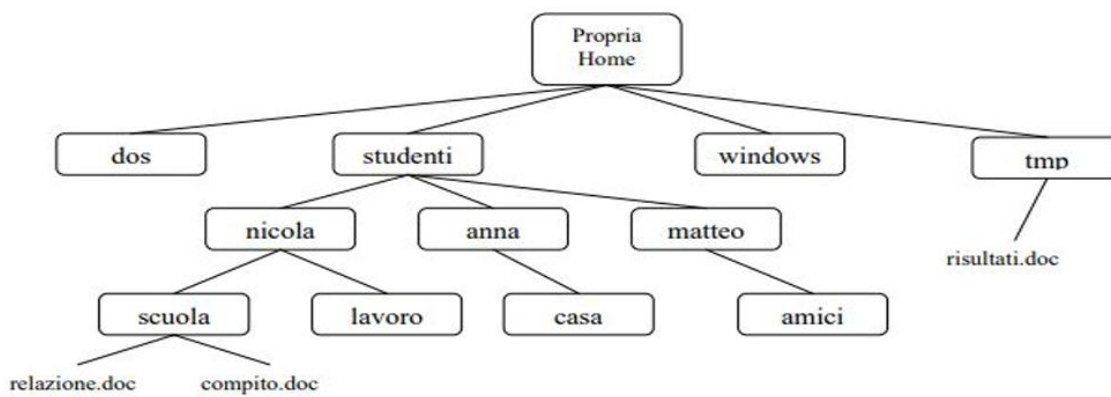




## Esercizi Shell

### ESERCIZIO 1

Generazione cartelle e sottocartelle  
Gerarchia



Creo la cartella **Esercizio** dentro la cartella **Home**

```
(kali㉿kali)-[~]  
$ mkdir Esercizio
```

Ho creato le 4 sottocartelle con il comando **mkdir {dos,studenti,windows,tmp}** e poi le ho visualizzate con **ls** e anche con **tree**

```
(kali㉿kali)-[~/Esercizio]
$ mkdir {dos,studenti,windows,tmp}

(kali㉿kali)-[~/Esercizio]
$ ls
dos  studenti  tmp  windows

(kali㉿kali)-[~/Esercizio]
$ tree
.
├── dos
├── studenti
├── tmp
└── windows
```

Sono entrato dentro studenti con il comando **cd** e poi ho creato, al suo interno, le tre sottocartelle nicola, anna e matteo

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

(kali㉿kali)-[~/Esercizio/studenti]
$ mkdir {nicola,anna,matteo}

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

Stessa cosa per le cartelle dentro **nicola**, ma ho riscontrato un errore dovuto al fatto che tra la virgola e i due nomi delle cartelle ci fosse uno spazio, rimuovendolo l'errore è stato risolto.

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

(kali㉿kali)-[~/Esercizio/studenti/nicola]
$ mkdir {scuola, lavoro}
zsh: parse error near `}'

(kali㉿kali)-[~/Esercizio/studenti/nicola]
$ mkdir {scuola,lavoro}

(kali㉿kali)-[~/Esercizio/studenti/nicola]
$ tree
.
├── lavoro
└── scuola
```

Sono entrato dentro **scuola** e ho creato i due file **compito.doc** e **relazione.doc** con il comando **touch**

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

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

(kali㉿kali)-[~/Esercizio/studenti/nicola/scuola]
$ tree
.
├── compito.doc
└── relazione.doc

1 directory, 2 files
```

Sono risalito di tre livelli prima con **cd.. / cd..** (due livelli) e poi **cd..** (un ulteriore livello) e poi ho visualizzato la gerarchia per vedere cosa manca

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

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

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

Una volta dentro **esercizio** ho creato la cartella **casa** e amici, rispettivamente dentro **anna** e **matteo**

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

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

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

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

Dentro **Esercizio** ho creato il file **risultati.doc**, all'interno della cartella **tmp**

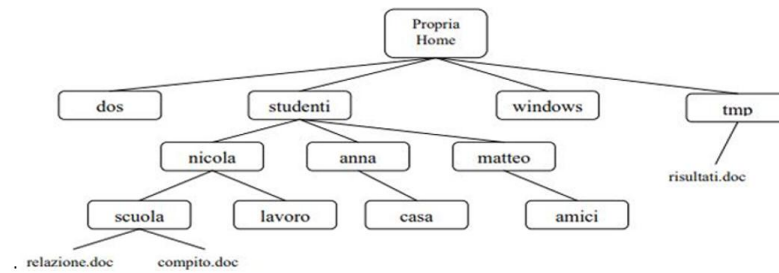
Come si può vedere tutte le carte e i file sono stati creati seguendo la gerarchia richiesta.

```

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

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

```



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

Mi sposto dentro **lavoro**

```

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

```

### Path assoluto

Dopo vari tentativi mi sono reso conto che non conoscevo il path assoluto

```

(kali@kali)-[~/Esercizio/studenti/nicola/lavoro]
$ cd Esercizio/studenti/anna/casa
cd: no such file or directory: Esercizio/studenti/anna/casa

(kali@kali)-[~/Esercizio/studenti/nicola/lavoro]
$ cd home/Esercizio/studenti/anna/casa
cd: no such file or directory: home/Esercizio/studenti/anna/casa

(kali@kali)-[~/Esercizio/studenti/nicola/lavoro]
$ cd Esercizio/studenti/anna/casa
cd: no such file or directory: Esercizio/studenti/anna/casa

(kali@kali)-[~/Esercizio/studenti/nicola/lavoro]
$ cd /Esercizio/studenti/anna/casa
cd: no such file or directory: /Esercizio/studenti/anna/casa

```

Quindi sono entrato nella cartella e l'ho visualizzato con il comando **pwd**

```

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

(kali@kali)-[~/Esercizio/studenti/anna/casa]
$ pwd
/home/kali/Esercizio/studenti/anna/casa

```



A quel punto l'ho inserito dopo **cd** e sono riuscito a spostarmi di cartella

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

Con **CD** – ritorno nel percorso precedente

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

### Path relativo

Accedo inserendo il path relativo

Con **../..** sono risalito di due cartelle e poi ho inserito il percorso per arrivare a **casa**

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

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

Copio, dentro la cartella in cui mi trovo, il file **compito.doc** che sta nella cartella **casa**

```
(kali㉿kali)-[~/Esercizio/studenti/anna/casa]
$ cp ../../nicola/scuola/compito.doc compito.doc
(kali㉿kali)-[~/Esercizio/studenti/anna/casa]
$ tree
.
├── compito.doc
```

Avrei potuto farlo anche con questa sintassi **cp ../../nicola/scuola/compito .**

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

Sposto il file relazione con il comando **mv** (move), questa volta usando la sintassi con il punto

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

(kali㉿kali)-[~/Esercizio/studenti/anna/casa]
$ tree
.
├── compito.doc
└── relazione.doc
```

### c) Cancella la cartella tmp

Elimino con il comando **rm** ricorsivamente (-rf) la cartella **tmp**

```
(kali㉿kali)-[~/Esercizio]
$ rm -rf tmp
```

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

11 directories, 3 files
```

### d) Creare il file pippo.txt nella cartella lavoro

Creo il file **pippo.txt** dentro la cartella **lavoro**

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

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

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

## AUTORIZZAZIONI

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

Con **ls -l** visualizzo le autorizzazioni.

Come vediamo l'utente ha permessi di lettura e scrittura "**rw**" (readwrite), il gruppo anche ma gli altri utenti possono solo leggere "**r**" (read); se ci fosse stata una **x** ci sarebbe il permesso di eseguire (execute)

**rw-rw-r--**

User Group Others

```
(kali@kali)-[~/Esercizio/studenti/nicola/lavoro]
$ ls -l pippo.txt
-rw-rw-r-- 1 kali kali 0 Jul 22 16:38 pippo.txt
```

Ho tolto con il segno "--" i permessi in scrittura (**W**, write) al Gruppo

```
(kali@kali)-[~/Esercizio/studenti/nicola/lavoro]
$ chmod g-w pippo.txt
(kali@kali)-[~/Esercizio/studenti/nicola/lavoro]
$ ls -l
total 0
-rw-r--r-- 1 kali kali 0 Jul 22 16:38 pippo.txt
```

Per fornire tutte le autorizzazioni a **UGO** è possibile usare il comando **chmod 777**

```
(kali@kali)-[~/Esercizio/studenti/nicola/lavoro]
$ chmod 777 pippo.txt
(kali@kali)-[~/Esercizio/studenti/nicola/lavoro]
$ ls -l
total 0
-rwxrwxrwx 1 kali kali 0 Jul 22 16:38 pippo.txt
```

## f) Nascondere il contenuto della cartella anna

Per nascondere la cartella **anna** dobbiamo rinominarla **.anna** perché il sistema nasconde di default tutte le cartelle che iniziano con il punto.

L'unico modo per visualizzarle è con il comando **ls -a**



```

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

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

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

```

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

Siccome il file **pippo.txt** era vuoto ho dovuto prima scriverci qualcosa dentro con il comando **echo**.

Una volta inserito del testo l'ho visualizzato con il comando **cat**.

```

(kali㉿kali)-[~/Esercizio/studenti/nicola/lavoro]
$ echo "ciao a tutti" > pippo.txt

(kali㉿kali)-[~/Esercizio/studenti/nicola/lavoro]
$ cat pippo.txt
ciao a tutti

```

h) Rimuovere la cartella amici

Per rimuovere la cartella amici uso il comando **rmdir** indicando la cartella da rimuovere

```

(kali㉿kali)-[~/Esercizio]
$ tree
.
├── dos
├── studenti
│   ├── matteo
│   │   └── amici
│   ├── nicola
│   │   ├── lavoro
│   │   │   ├── pippo.txt
│   │   └── scuola
│   │       └── compito.doc
└── windows

```

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

(kali@kali)-[~/Esercizio]
$ tree
.
├── dos
├── studenti
│   ├── matteo
│   └── nicola
│       ├── lavoro
│       ├── pippo.txt
│       └── scuola
│           └── compito.doc
└── windows
```

## i) Rimuovere tutte le cartelle precedentemente create

Per rimuovere le cartelle create uso il comando **rm -rf doc** e poi specifico il nome delle cartelle

```
(kali@kali)-[~/Esercizio]
$ tree
.
├── dos
├── studenti
│   ├── matteo
│   └── nicola
│       ├── lavoro
│       ├── pippo.txt
│       └── scuola
│           └── compito.doc
└── windows

8 directories, 2 files

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

(kali@kali)-[~/Esercizio]
$ tree
.
0 directories, 0 files
```

## ESERCIZIO FACOLTATIVO

Provare i comandi:

W

```
(kali@kali)-[~]
$ w
08:25:57 up 3:57, 1 user, load average: 0.03, 0.05, 0.08
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
kali - 04:29 0.00s 0.01s lightdm --session-child 13 24
```

Who

```
(kali㉿kali)-[~]  
$ who  
kali      seat0      2025-07-23 04:29 (:0)
```

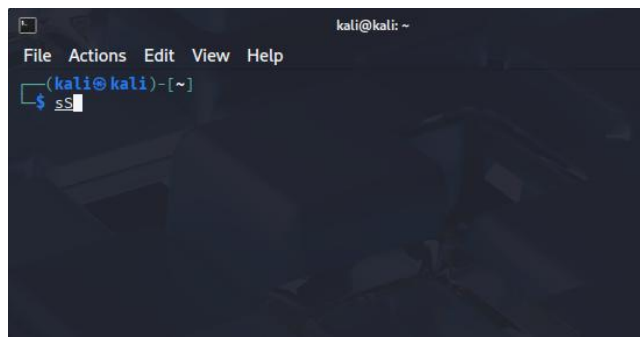
who am i

```
(kali㉿kali)-[~]  
$ whoami  
kali
```

## Esercizi – Processi:

### 1. Aprire un terminale

Apri il terminale



```
kali@kali: ~  
File Actions Edit View Help  
(kali㉿kali)-[~]  
$
```

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

Per leggere il manuale di un comando bisogna usare **man** seguito dal comando

```
(kali㉿kali)-[~]  
$ man ps
```

```
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.  
  
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
```

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

```
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 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_SIGINFO flag to sigaction(2), then it can obtain this  
data via the si_value field of the siginfo_t structure.
```

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

Il comando **man** per **job** non funziona perché non si tratta di un comando esterno ma è parte delle funzionalità della shell

Una volta visualizzato il manuale, premendo **/** e inserendo un testo che vogliamo cercare, verranno evidenziate tutte le occorrenze di tale testo. Comando utile per cercare in modo selettivo del testo.

```
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.  
/number
```



```
File Actions Edit View Help
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_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 num-
ber. 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.
```

### 3. lanciare il comando vi pippo

Lancio **vi pippo**



```
kali@ka
File Actions Edit View Help
(kali@kali)-[~]
$ vi pippo
```

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

Con **ps** viene visualizzato solo il processo corrente **zsh** e il suo sotto processo

```
kali@kal
File Actions Edit View Help
(kali@kali)-[~]
$ ps
  PID TTY          TIME CMD
 48103 pts/1        00:00:00 zsh
 48131 pts/1        00:00:00 ps
```

Co il comando **ps aux** vengono visualizzati tutti i processi dell'utente

```
(kali@kali)-[~]
$ ps aux
USER          PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.7 23928 14208 ?        Ss   04:28   0:01 /sbin/init s
root           2  0.0  0.0      0     0 ?        S    04:28   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        S    04:28   0:00 [pool_workqu
root           4  0.0  0.0      0     0 ?        I<   04:28   0:00 [kworker/R-k
root           5  0.0  0.0      0     0 ?        I<   04:28   0:00 [kworker/R-r
root           6  0.0  0.0      0     0 ?        I<   04:28   0:00 [kworker/R-s
root           7  0.0  0.0      0     0 ?        I<   04:28   0:00 [kworker/R-s
root           8  0.0  0.0      0     0 ?        I<   04:28   0:00 [kworker/R-n
root          11  0.0  0.0      0     0 ?        I<   04:28   0:00 [kworker/0:0
root          12  0.0  0.0      0     0 ?        I    04:28   0:00 [kworker/u8:
root          13  0.0  0.0      0     0 ?        I<   04:28   0:00 [kworker/R-m
root          14  0.0  0.0      0     0 ?        I    04:28   0:00 [rcu_tasks_k
root          15  0.0  0.0      0     0 ?        I    04:28   0:00 [rcu_tasks_r
root          16  0.0  0.0      0     0 ?        I    04:28   0:00 [rcu_tasks_t
root          17  0.0  0.0      0     0 ?        S    04:28   0:00 [ksoftirqd/0
root          18  0.0  0.0      0     0 ?        I    04:28   0:02 [rcu_preempt
root          19  0.0  0.0      0     0 ?        S    04:28   0:00 [rcu_exp_par
root          20  0.0  0.0      0     0 ?        S    04:28   0:00 [rcu_exp_gp_
root          21  0.0  0.0      0     0 ?        S    04:28   0:00 [migration/0
root          22  0.0  0.0      0     0 ?        S    04:28   0:00 [idle_inject
root          23  0.0  0.0      0     0 ?        S    04:28   0:00 [cpuhp/0]
root          24  0.0  0.0      0     0 ?        S    04:28   0:00 [cpuhp/1]
root          25  0.0  0.0      0     0 ?        S    04:28   0:00 [idle_inject
root          26  0.0  0.0      0     0 ?        S    04:28   0:00 [migration/1
root          27  0.0  0.0      0     0 ?        S    04:28   0:01 [ksoftirqd/1
root          30  0.0  0.0      0     0 ?        I    04:28   0:00 [kworker/u9:
root          31  0.0  0.0      0     0 ?        S    04:28   0:00 [kworker/5:1
```

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

Usando una pipe con `|` e selezionando il testo "pippo" con **grep** all'interno dei nostri processi identificheremo il processo interessato.

```
(kali㉿kali)-[~]  
$ ps aux | grep pippo  
kali      47891  0.0  0.5 16184 10792 pts/0    Sl+  06:03   0:00 vi pippo  
kali      53632  0.0  0.1  6528  2236 pts/1    S+   06:15   0:00 grep --color  
=auto pippo
```

Process id

Inserendo **kill -9 [process ID]**, il relativo processo verrà terminato.

Usiamo il **9** perché si tratta del numero relativo al segnale "SIGKILL" che abbiamo trovato nella documentazione di Kill

3. Any command line shortcuts or commands that are associated with them

Here is the list of signals and what I have so far.

```
0 - ?  
1 - SIGHUP - ?, controlling terminal closed,  
2 - SIGINT - interrupt process stream, ctrl-C  
3 - SIGQUIT - like ctrl-C but with a core dump, interruption by error in code, ctl-/  
4 - SIGILL  
5 - SIGTRAP  
6 - SIGABRT  
7 - SIGBUS  
8 - SIGFPE  
9 - SIGKILL - terminate immediately/hard kill, use when 15 doesn't work or when some  
10 - SIGUSR1
```

```
(kali㉿kali)-[~]  
$ kill -9 47891  
  
(kali㉿kali)-[~]  
$ ps aux | grep pippo  
kali      61637  0.0  0.1  6528  2188 pts/1    S+   06:31   0:00 grep --color  
=auto pippo
```

## 6. lanciare il comando firefox in background

Con il comando **firefox &** lancio Firefox in background

```
[1] ~ done  
(kali㉿kali)-[~]  
$ firefox &  
[1] 72485
```

Si apre Firefox e se lo chiudo, termina (done)

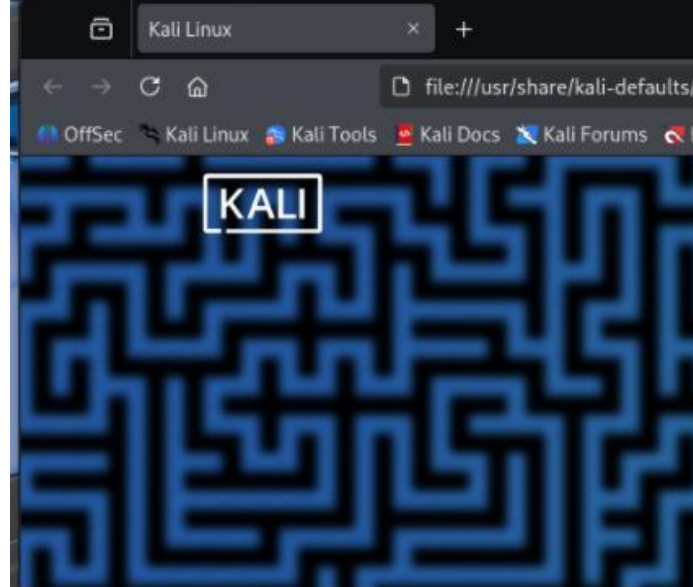
```
(kali@kali)-[~]  
$ [GFX1-]: RenderCompositorSWGL failed mapping default framebuffer, no dt  
[1] + done      firefox
```

## 8. cercare di terminare il processo firefox

Lanciamo Firefox (1), lo sospendiamo con **ctrl+z** (2) e poi lo portiamo in background con **bg** (3). Vedi immagine sotto.

Con **fg** riprendiamo il controllo dell'eseguibile

```
(kali@kali)-[~]  
$ firefox (1)  
^Z (2)  
zsh: suspended firefox  
$ bg (3)  
[1] + continued firefox  
$  
$
```



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

Con il comando **jobs** vediamo i processi in corso

```
(kali㉿kali)-[~]  
$ jobs  
[1] + running    firefox  
[2] - running    sleep 500
```

Con il comando **Kill %numero processo** chiudiamo il processo relativo al numero inserito

```
(kali㉿kali)-[~]  
$ kill %1  
  
(kali㉿kali)-[~]  
$  
[1] + terminated  firefox  
(kali㉿kali)-[~]  
$ jobs  
[2] + running    sleep 500
```

## 9. verificare quanto spazio si sta occupando su disco

Il comando **df -h** consente di visualizzare lo spazio occupato su disco

**Df** (disk free) serve per visualizzare lo spazio e **-h** serve a rendere i valori leggibili all'essere umano, ovvero convertendo i numeri in formati come MG, GB, etc.

```
(kali㉿kali)-[~]  
$ df -h  
Filesystem      Size  Used Avail Use% Mounted on  
udev            921M   0    921M   0% /dev  
tmpfs           198M  972K   197M   1% /run  
/dev/sda1       79G   16G   59G   21% /  
tmpfs           987M   4.0K   987M   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           987M  36K   987M   1% /tmp  
tmpfs           1.0M   0    1.0M   0% /run/credentials/getty@tty1.service  
tmpfs           198M  124K   198M   1% /run/user/1000
```

### ESERCIZIO SUEPER FACOLTATIVO

#### GameShell

Sono arrivato alla missione 10 ma continuerò!

```
~/Castle/Cellar
[mission 9] $ ls -a .*spider*
.12466_spider_17 .19468_spider_12 .26153_spider_46 .3815_spider_26
.12610_spider_23 .19516_spider_33 .26553_spider_19 .4092_spider_21
.12637_spider_49 .20326_spider_34 .26721_spider_48 .4229_spider_11
.13450_spider_27 .21899_spider_18 .27443_spider_31 .4567_spider_25
.13512_spider_38 .22938_spider_42 .2949_spider_3 .5717_spider_8
.14373_spider_10 .23300_spider_37 .29582_spider_13 .7088_spider_32
.15115_spider_2 .23425_spider_30 .29597_spider_4 .7327_spider_7
.15149_spider_16 .24361_spider_28 .2969_spider_29 .8155_spider_1
.16006_spider_35 .24961_spider_36 .29997_spider_24 .8386_spider_22
.16345_spider_14 .25108_spider_45 .3024_spider_15 .9293_spider_39
.1751_spider_9 .25109_spider_41 .30410_spider_5 .9593_spider_6
.18130_spider_40 .25180_spider_43 .31396_spider_20
.18648_spider_44 .25513_spider_47 .363_spider_50
```

```
~/Castle/Cellar
[mission 9] $ rm .*spider*
```

```
~/Castle/Cellar
[mission 9] $ gsh check
```

Congratulations, mission 9 has been successfully completed!

```
+-----+
| Congratulations !
|
| From now on, the ``ls`` command will
| automatically show a "/" character at the
| end of directories.
|
+-----+
```

[ progress was saved in /home/kali/gameshell-save.sh ]

```
--+-----+
| Use the command
| $ gsh help
| to get the list of "gsh" commands.
|
--+-----+
```

```
~/Castle/Cellar
[mission 10] $
```