

Il primo esercizio richiede la creazione di una serie di cartelle e sottocartelle.

Per fare ciò abbiamo utilizzato i comandi `mkdir` per la creazione delle singole cartelle, e il comando `touch` per la creazione dei file all'interno delle cartelle stesse (come da consegna).

Allego screenshot dell'avvenuta creazione delle cartelle come sopra.

```
(kali@kali)-[~/home/studenti/nicola]
$ mkdir scuola

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

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

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

(kali@kali)-[~/home/studenti/nicola/scuola]
$ touch compito.doc

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

Per spostarsi da una cartella all'altra, tramite percorso relativo, utilizzeremo il comando `cd ..`

Mentre per spostarci tramite percorso assoluto dalla directory Nicola alla directory Anna utilizzeremo il comando `cd /percorso della directory`.

Il punto a) dell'esercizio richiedeva di copiare il file `compito.doc` dalla cartella `scuola` alla cartella `casa`.

Per fare questa operazione abbiamo utilizzato il comando `cp /percorso della cartella di origine/nome del file /percorso della directory di destinazione`.

```
kali@kali: ~/home/studenti/anna/casa
File Actions Edit View Help
(kali@kali)-[~/home/studenti]
$ cd nicola

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

(kali@kali)-[~/home/studenti/nicola/scuola]
$ ls
compito.doc  relazione.doc

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

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

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

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

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

(kali@kali)-[~/home/studenti/anna/casa]
$ ls
compito.doc

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

A questo punto l'esercizio chiede di spostare nella directory corrente Casa il file relazione.doc, che si trovava all'interno della directory scuola. Con il comando mv /percorso cartella di origine/nome file /percorso della cartella di destinazione.

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

(kali㉿kali)-[~/home/studenti/anna/casa]
$ ls
compito.doc  relazione.doc

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

Al punto c) viene richiesta la cancellazione della directory tmp, contenente il file "risultati.doc". Per fare ciò dobbiamo utilizzare il comando rm -r nome directory. Mettendo la r davanti abbiamo avuto il privilegio amministratori per l'eliminazione della directory contenente il file "risultati.doc"

Al punto d) viene chiesta la creazione del file "pippo.txt" all'interno della directory "lavoro". Per effettuare questa operazione abbiamo utilizzato il comando touch nome del file.

Al punto e) viene chiesto di cambiare gli attributi del file "pippo.txt" in modo tale da renderlo leggibile, scrivibile ed eseguibile solo da noi, e solo leggibile dagli altri utenti. Per fare ciò abbiamo utilizzato il comando chmod 744 pippo.txt, dato che seguendo i permessi numerici, il numero 7 si configura come il permesso per tutte e tre le operazioni, mentre il numero 4 si configura come il permesso di sola lettura.

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

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

(kali㉿kali)-[~/home/studenti/nicola/lavoro]
$ chmod 744 pippo.txt

(kali㉿kali)-[~/home/studenti/nicola/lavoro]
$ ls -l pippo.txt
-rwxr--r-- 1 kali kali 0 Nov 24 12:21 pippo.txt

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

Al punto f) veniva chiesto di nascondere il contenuto all'interno della cartella Anna. Per effettuare questa operazione abbiamo utilizzato il comando mv /percorso della directory /percorso della directory/.nome directory; il punto messo davanti al nome della directory la rende invisibile.

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

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

(kali㉿kali)-[~/home/studenti]
$
```

Al punto g) viene richiesto di aprire il file pippo.txt presente all'interno della directory lavoro. Per farlo abbiamo utilizzato il comando cat nome file

Al punto h) viene chiesta la cancellazione della directory "amici", presente all'interno della directory "matteo". Per effettuare questa operazione utilizziamo il comando rmdir nome directory, come si vede nello screenshot allegato

```
(kali㉿kali)-[~/home/studenti/matteo]
$ ls
amici

(kali㉿kali)-[~/home/studenti/matteo]
$ rmdir amici

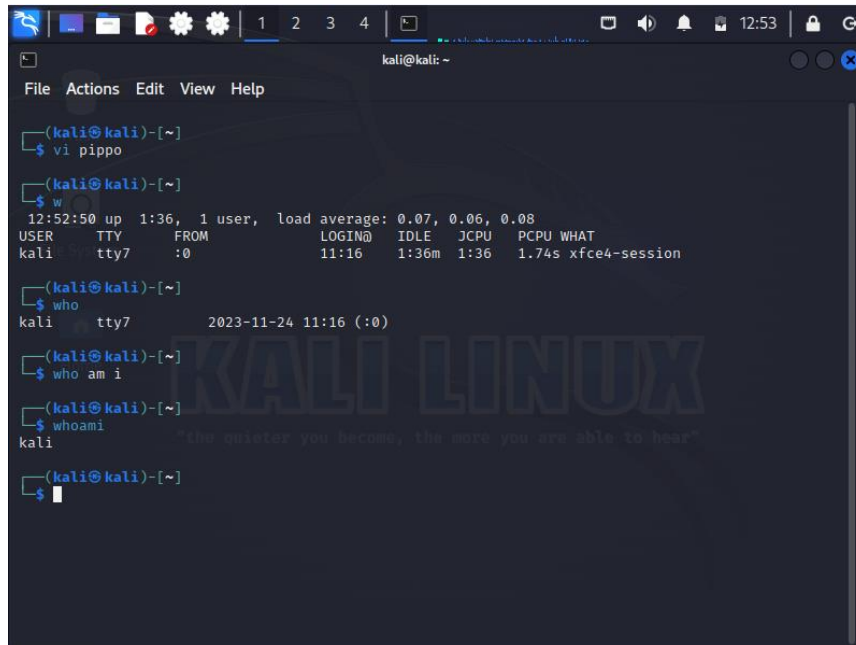
(kali㉿kali)-[~/home/studenti/matteo]
$ ls

(kali㉿kali)-[~/home/studenti/matteo]
$
```

All'ultimo punto i) viene richiesta l'eliminazione di tutte le directory create fino ad ora. Effettueremo l'operazione richiesta con il comando rm -r nome della directory principale. In questo modo abbiamo provveduto alla cancellazione di tutte le cartelle e sottocartelle precedentemente create.

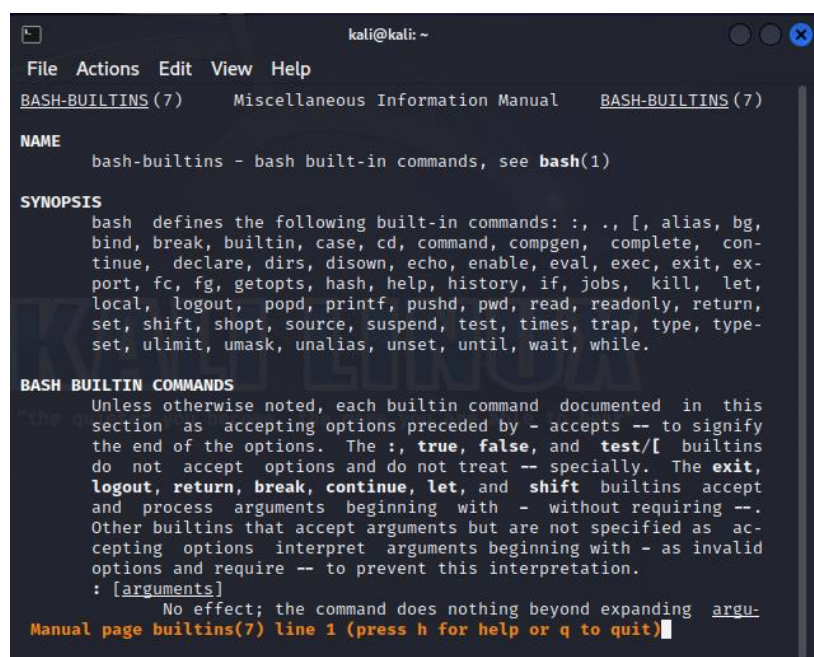
Seconda consegna:

Nel secondo esercizio viene richiesto di provare i comandi `w`, `who` e `whoami` come da screenshot allegato qui sotto:



```
kali@kali: ~  
File Actions Edit View Help  
  
(kali@kali)-[~]  
$ vi pippo  
  
(kali@kali)-[~]  
$ w  
12:52:50 up 1:36, 1 user, load average: 0.07, 0.06, 0.08  
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU WHAT  
kali      tty7      :0               11:16    1:36m  1:36   1.74s xfce4-session  
  
(kali@kali)-[~]  
$ who  
kali      tty7      2023-11-24 11:16 (:0)  
  
(kali@kali)-[~]  
$ who am i  
  
(kali@kali)-[~]  
$ whoami  
kali  
"the quieter you become, the more you are able to hear"  
  
(kali@kali)-[~]  
$
```

2) richiesta la lettura del manuale per i comandi `job`, `kill` e `ps`. Per fare ciò abbiamo utilizzato il comando `man` builtins, e tramite questo abbiamo avuto modo di accedere al manuale di shell per leggere i suddetti comandi.



```
kali@kali: ~  
File Actions Edit View Help  
BASH-BUILTINS(7) Miscellaneous Information Manual BASH-BUILTINS(7)  
  
NAME  
    bash-builtins - bash built-in commands, see bash(1)  
  
SYNOPSIS  
    bash defines the following built-in commands: :, ., [, alias, bg, bind, break, builtin, case, cd, command, compgen, complete, continue, declare, dirs, disown, echo, enable, eval, exec, exit, export, fc, fg, getopts, hash, help, history, if, jobs, kill, let, local, logout, popd, printf, pushd, pwd, read, readonly, return, set, shift, shopt, source, suspend, test, times, trap, type, type-set, ulimit, umask, unalias, unset, until, wait, while.  
  
BASH BUILTIN COMMANDS  
    Unless otherwise noted, each builtin command documented in this section as accepting options preceded by - accepts -- to signify the end of the options. The :, true, false, and test/[ builtins do not accept options and do not treat -- specially. The exit, logout, return, break, continue, let, and shift builtins accept and process arguments beginning with - without requiring --. Other builtins that accept arguments but are not specified as accepting options interpret arguments beginning with - as invalid options and require -- to prevent this interpretation.  
    : [arguments]  
    No effect; the command does nothing beyond expanding argu-  
Manual page builtins(7) line 1 (press h for help or q to quit)
```

