W8D4 – FRANCESCO MONTALTO

TASK 1

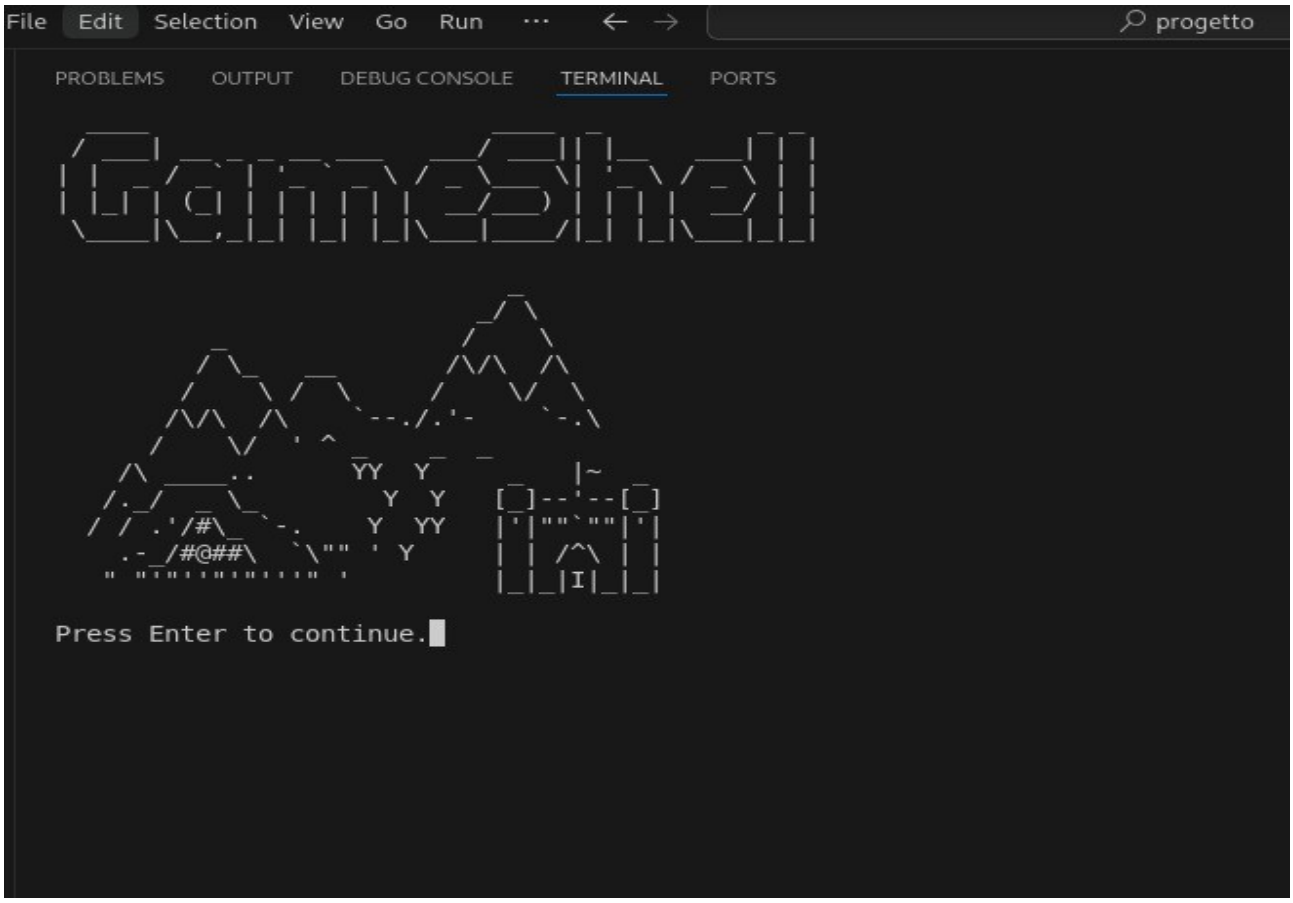
INTRO: Ho aggiornato Kali e i pacchetti disponibili, per poi scaricare tutti i pacchetti richiesti dal gioco.

```
┌──(kali㉿kali)-[~/progetto]
└─$ sudo apt install gettext man-db procps psmisc nano tree bsdmainutils x11-apps wget
gettext is already the newest version (0.23.1-2+b1).
man-db is already the newest version (2.13.1-1).
procps is already the newest version (2:4.0.4-9).
psmisc is already the newest version (23.7-2).
nano is already the newest version (8.6-1).
tree is already the newest version (2.2.1-1).
bsdmainutils is already the newest version (12.1.8).
x11-apps is already the newest version (7.7+11+b1).
wget is already the newest version (1.25.0-2).
The following packages were automatically installed and are no longer required:
  python3-packaging-whl  python3-pyinstaller-hooks-contrib  python3-wheel-whl
Use 'sudo apt autoremove' to remove them.

Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 808

┌──(kali㉿kali)-[~/progetto]
└─$
0 ⚠ 0
```

Ho scaricato l’ultima versione dello script di installazione direttamente da GitHub.

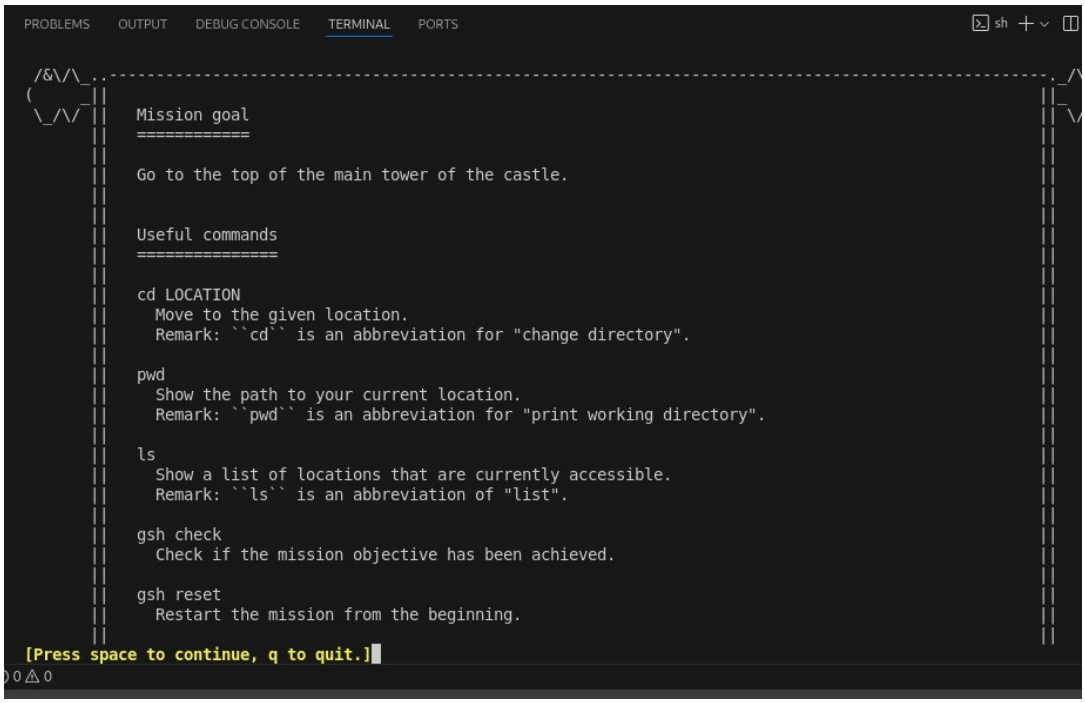
```
┌──(kali㉿kali)-[~/progetto]
└─$ wget https://github.com/phyver/GameShell/releases/download/latest/gameshell.sh
--2025-09-06 18:00:11--  https://github.com/phyver/GameShell/releases/download/latest/gameshell.sh
Resolving github.com (github.com)... 140.82.121.4
Connecting to github.com (github.com)|140.82.121.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://release-assets.githubusercontent.com/github-production-release-asset/94422621/59962d4f-cbe9-4183-a63a-8d45
13f145d4?sp=r&sv=2018-11-09&sr=b&spr=https&se=2025-09-06T22%3A52%3A38Z&rscd=attachment%3B+filename%3Dgameshell.sh&rsct=appli
cation%2Foctet-stream&skoid=96c2d410-5711-43a1-aedd-ab1947aa7ab0&sktid=398a6654-997b-47e9-b12b-9515b896b4de&skt=2025-09-06T2
1%3A51%3A50Z&ske=2025-09-06T22%3A52%3A38Z&sks=b&skv=2018-11-09&sig=H3Ua58HYo%2BwHsv3ugLhjzdbnzIhemS8TN8BFpNv3BFs%3D&jwt=eyJ0
eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJnaXRodWIuY29tIiwiYXVkIjoicmVsZWFzZS1hc3NldHMuZ2l0aHVidXNlcmNvbnRlbnQuY29tIiwia2V
5Ijoia2V5MSIsImV4cCI6MTc1NzE5NjMwOCwibmJmIjoxNzU3MTk2MDA4LCJwYXRoIjoicmVsZWFzZWFzZc2V0cHJvZHVjdGlvbi5ibG9iLmNvcmUud2luZG93cy5
uZXQifQ.dfCzfXmqRsl_GKQKfMaPKIYvJZmIGMUpCQdHVqSgeqs&response-content-disposition=attachment%3B%20filename%3Dgameshell.sh&res
ponse-content-type=application%2Foctet-stream [following]
--2025-09-06 18:00:12--  https://release-assets.githubusercontent.com/github-production-release-asset/94422621/59962d4f-cbe9
-4183-a63a-8d4513f145d4?sp=r&sv=2018-11-09&sr=b&spr=https&se=2025-09-06T22%3A52%3A38Z&rscd=attachment%3B+filename%3Dgameshel
l.sh&rsct=application%2Foctet-stream&skoid=96c2d410-5711-43a1-aedd-ab1947aa7ab0&sktid=398a6654-997b-47e9-b12b-9515b896b4de&s
kt=2025-09-06T21%3A51%3A50Z&ske=2025-09-06T22%3A52%3A38Z&sks=b&skv=2018-11-09&sig=H3Ua58HYo%2BwHsv3ugLhjzdbnzIhemS8TN8BFpNv3
BFs%3D&jwt=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJnaXRodWIuY29tIiwiYXVkIjoicmVsZWFzZS1hc3NldHMuZ2l0aHVidXNlcmNvbnRl
bnQuY29tIiwia2V5Ijoia2V5MSIsImV4cCI6MTc1NzE5NjMwOCwibmJmIjoxNzU3MTk2MDA4LCJwYXRoIjoicmVsZWFzZWFzZc2V0cHJvZHVjdGlvbi5ibG9iLmNv
cmUud2luZG93cy5uZXQifQ.dfCzfXmqRsl_GKQKfMaPKIYvJZmIGMUpCQdHVqSgeqs&response-content-disposition=attachment%3B%20filename%3Dg
ameshell.sh&response-content-type=application%2Foctet-stream
Resolving release-assets.githubusercontent.com (release-assets.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 1
85.199.111.133, ...
Connecting to release-assets.githubusercontent.com (release-assets.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
0 ⚠ 0
```

Ho eseguito e lanciato il gioco.

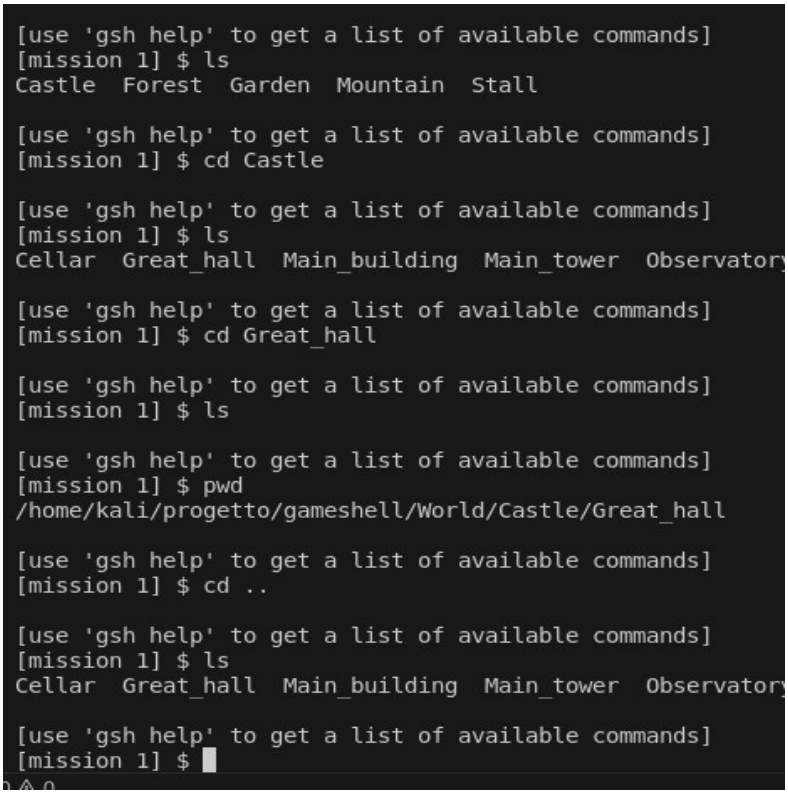


Seguirà un report delle missioni da 1 a 17.

MISSIONE 1



L’obbiettivo è arrivare alla cima della Main Tower.



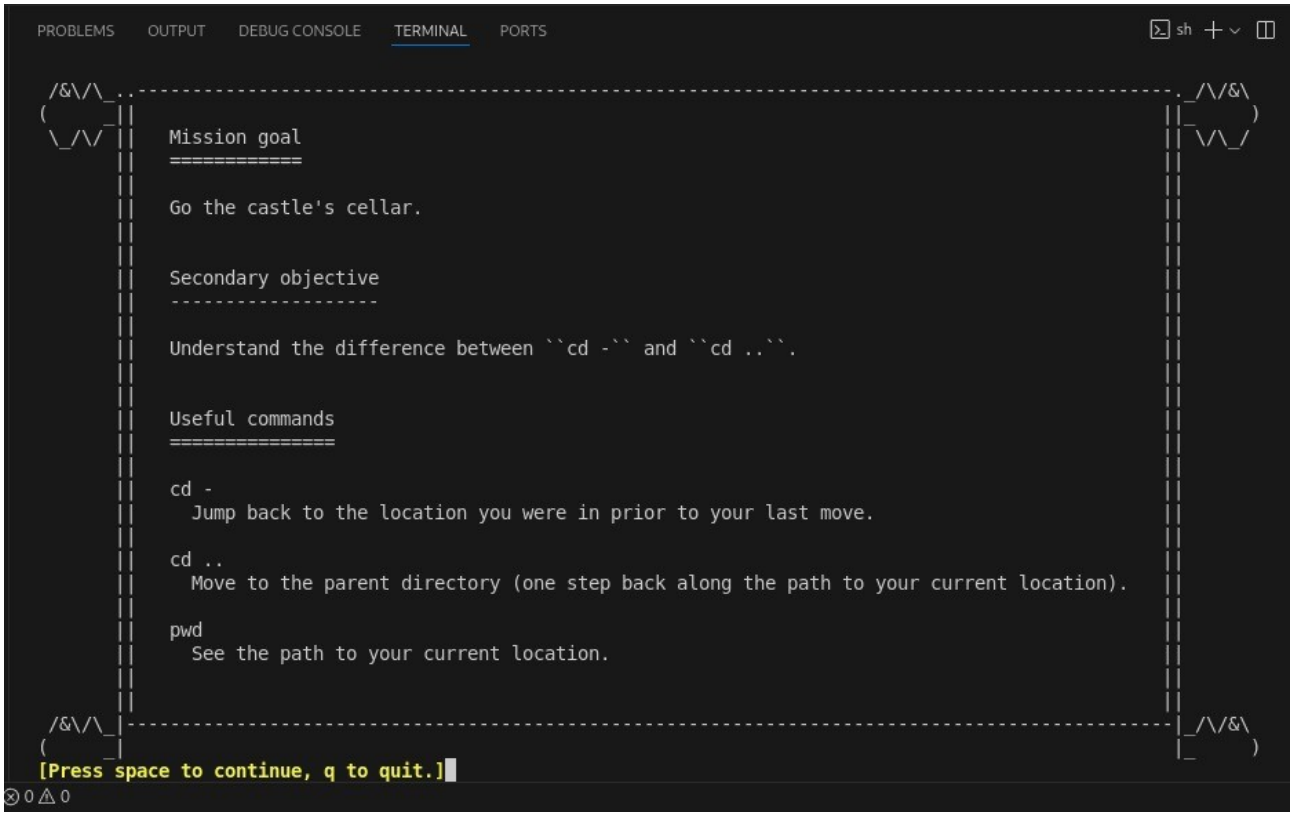
Ho navigato nelle sottodirectory esplorando fino alla cima usando **cd** (e varianti) e controllato il completamento con “gsh check”.

```
[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] $
```

0 ⚠ 0

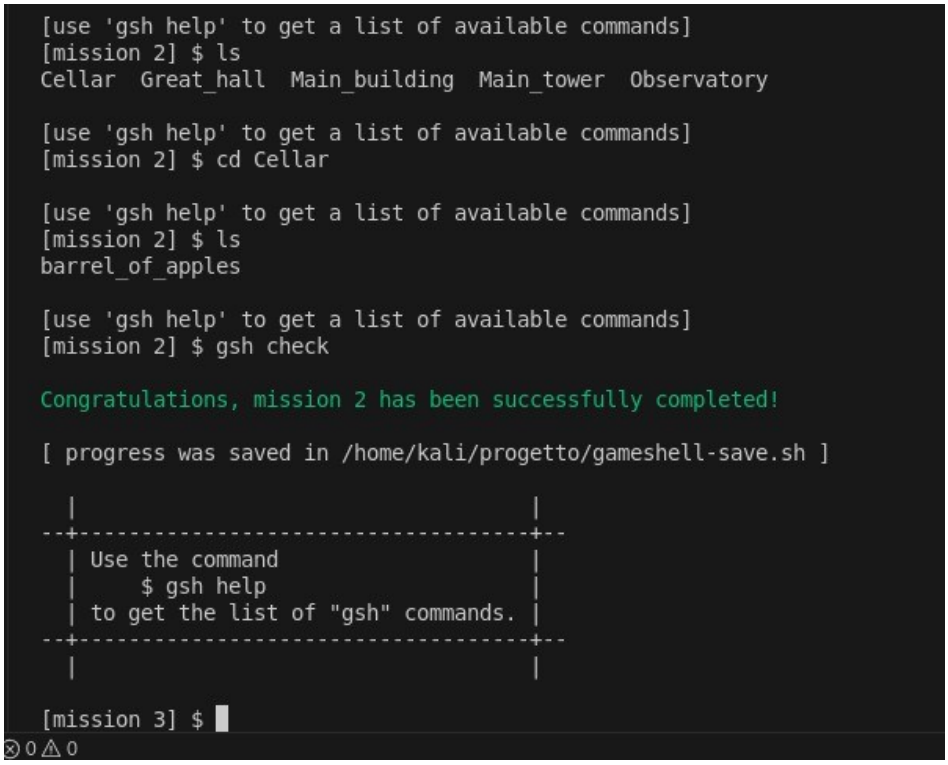
```
[use 'gsh help' to get a list of available commands]
[mission 1] $ gsh check

Congratulations, mission 1 has been successfully completed!

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

MISSIONE 2

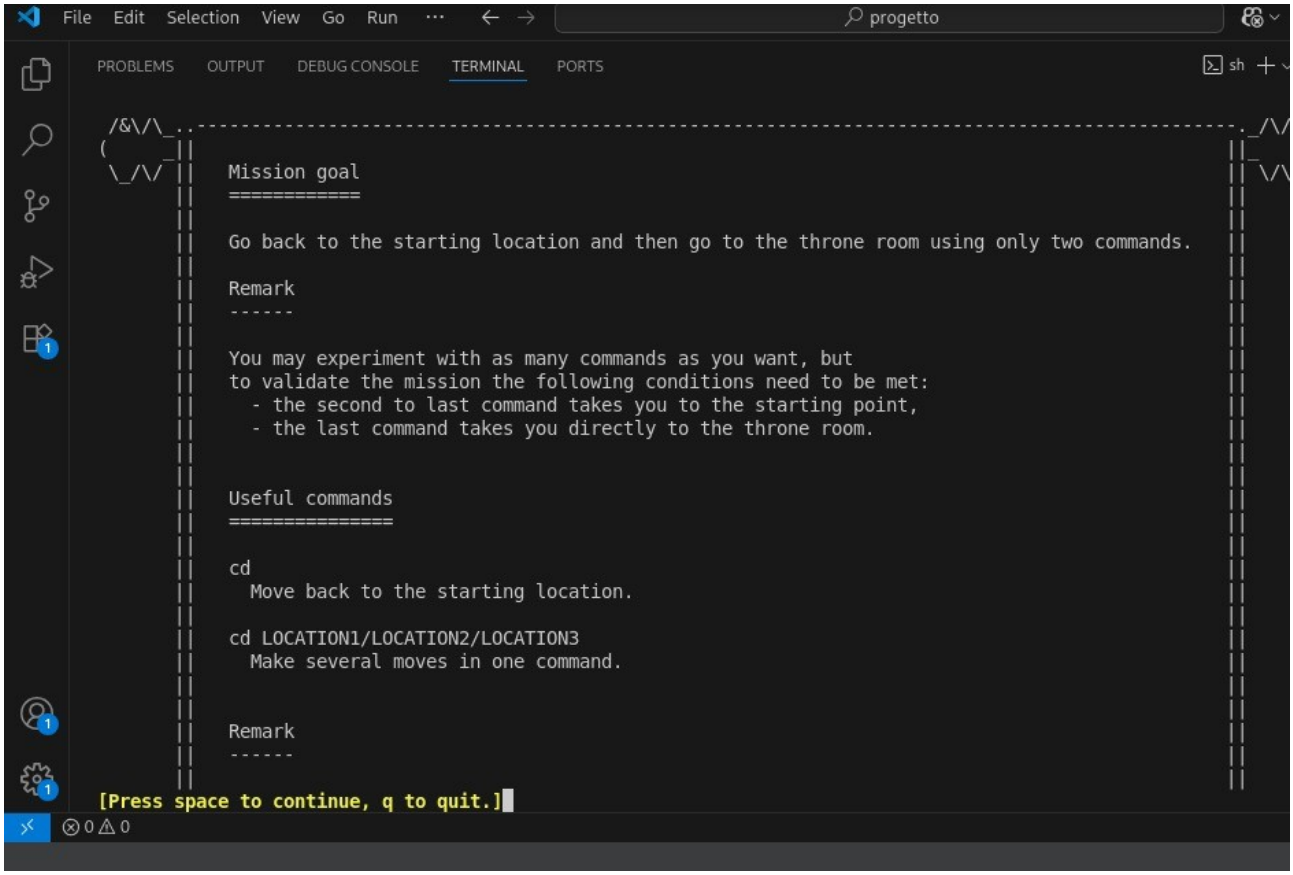


L’obbiettivo è andare nel cellar, capire cd e varianti, ed utilizzare correttamente **pwd** (che mostra il percorso attuale).

```
[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] $ ls
Top_of_the_tower

[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] $ ls
Second_floor

[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] $ ls
First_floor

[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] $ ls
Cellar  Great_hall  Main_building  Main_tower  Observatory

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

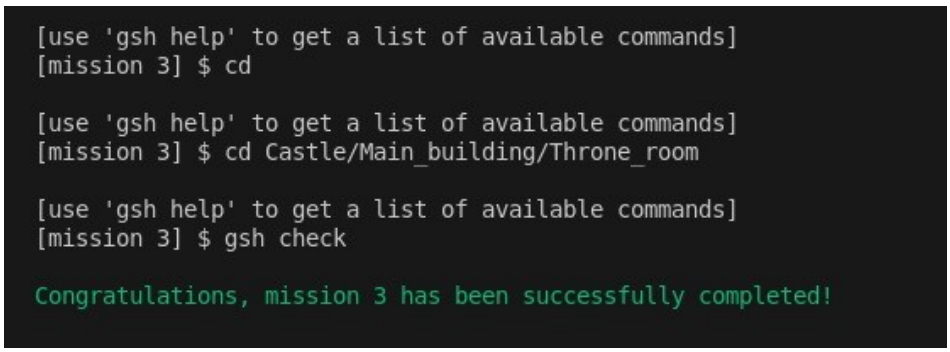

[use 'gsh help' to get a list of available commands]
[mission 2] $ ls
Cellar Great_hall Main_building Main_tower Observatory
[use 'gsh help' to get a list of available commands]
[mission 2] $ cd Cellar
[use 'gsh help' to get a list of available commands]
[mission 2] $ ls
barrel_of_apples
[use 'gsh help' to get a list of available commands]
[mission 2] $ gsh check
Congratulations, mission 2 has been successfully completed!
[progress was saved in /home/kali/progetto/gameshell-save.sh]
| Use the command $ gsh help to get the list of "gsh" commands.
[mission 3] $
0 0

MISSIONE 3



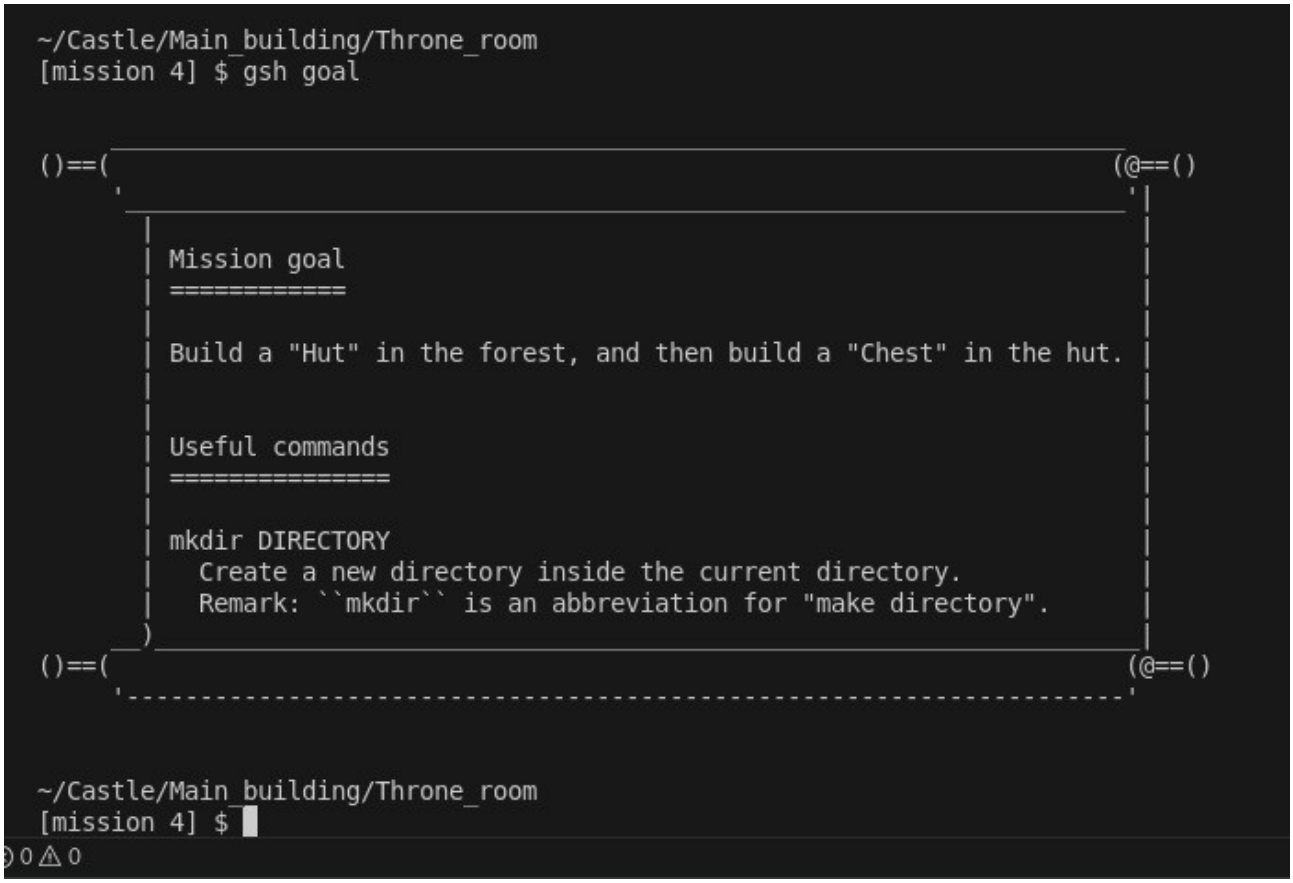
L’obbiettivo è tornare al punto di partenza e poi andare alla throne room, in due comandi.

Intanto ho capito il percorso fino alla throne room, come da immagine che segue.



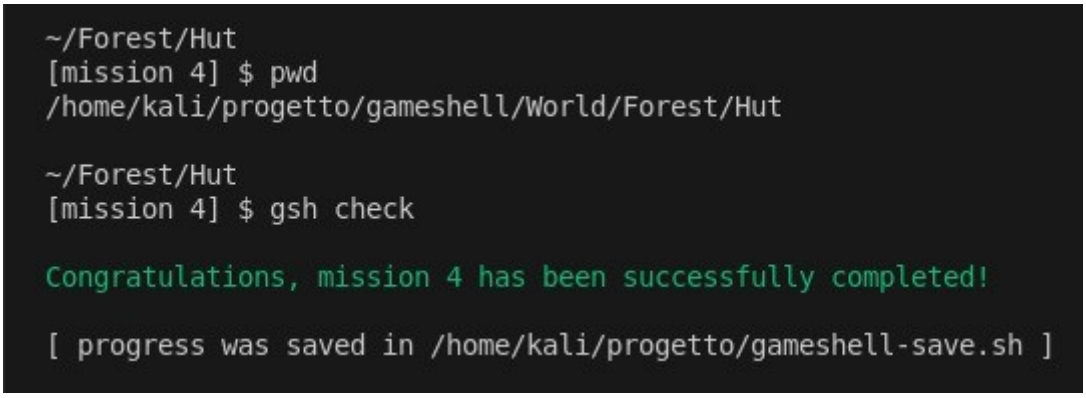
Ho utilizzato cd per tornare alla directory iniziale; e poi cd x/x/x/, completando il percorso per la Throne Room.

MISSIONE 4



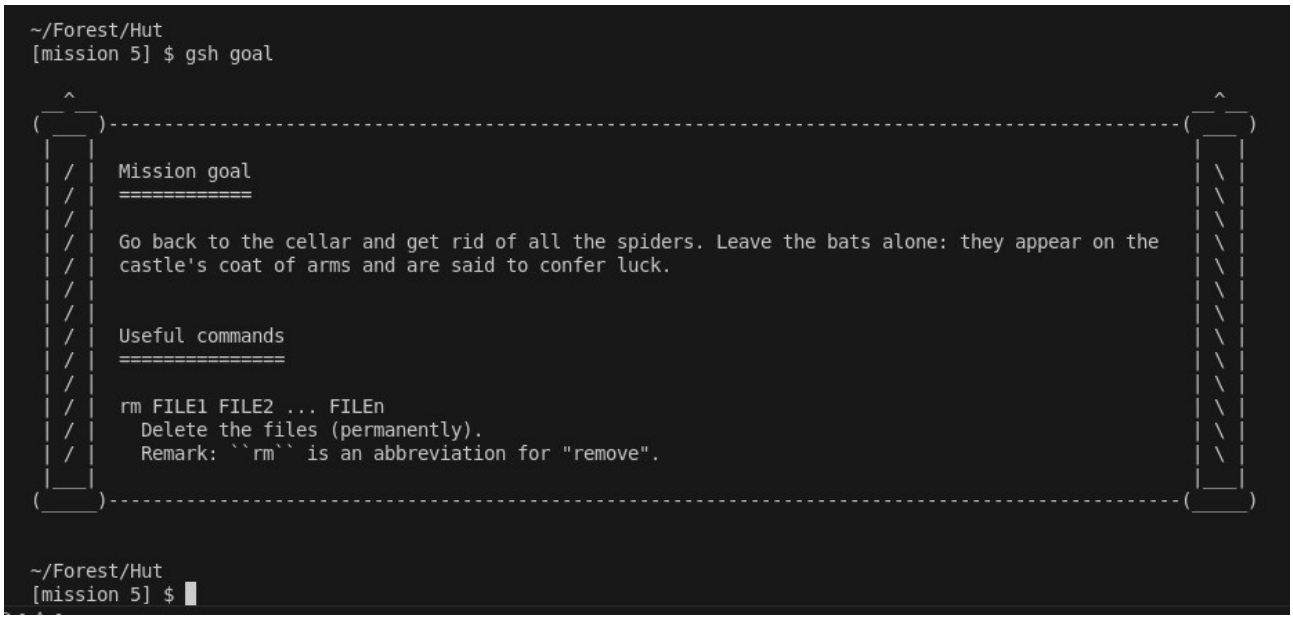
L’obbiettivo è creare una “Hut” nella Foresta e una “Chest” dentro la Hut.

Sono andato in Forest, ho creato una cartella Hut; sono andato in Hut e ho creato una cartella Chest.
“mkdir” crea directory, prima la Hut, poi la Chest dentro.



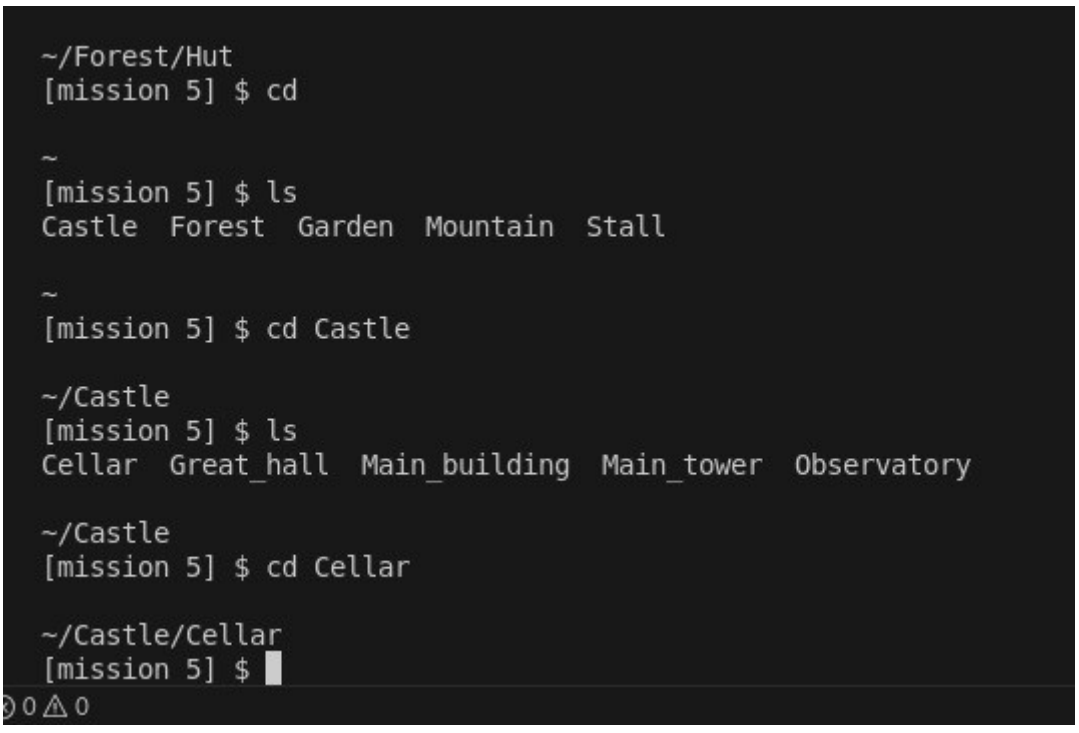
Anche qui ho eseguito, per sicurezza, la verifica (pwd) e il check.

MISSIONE 5

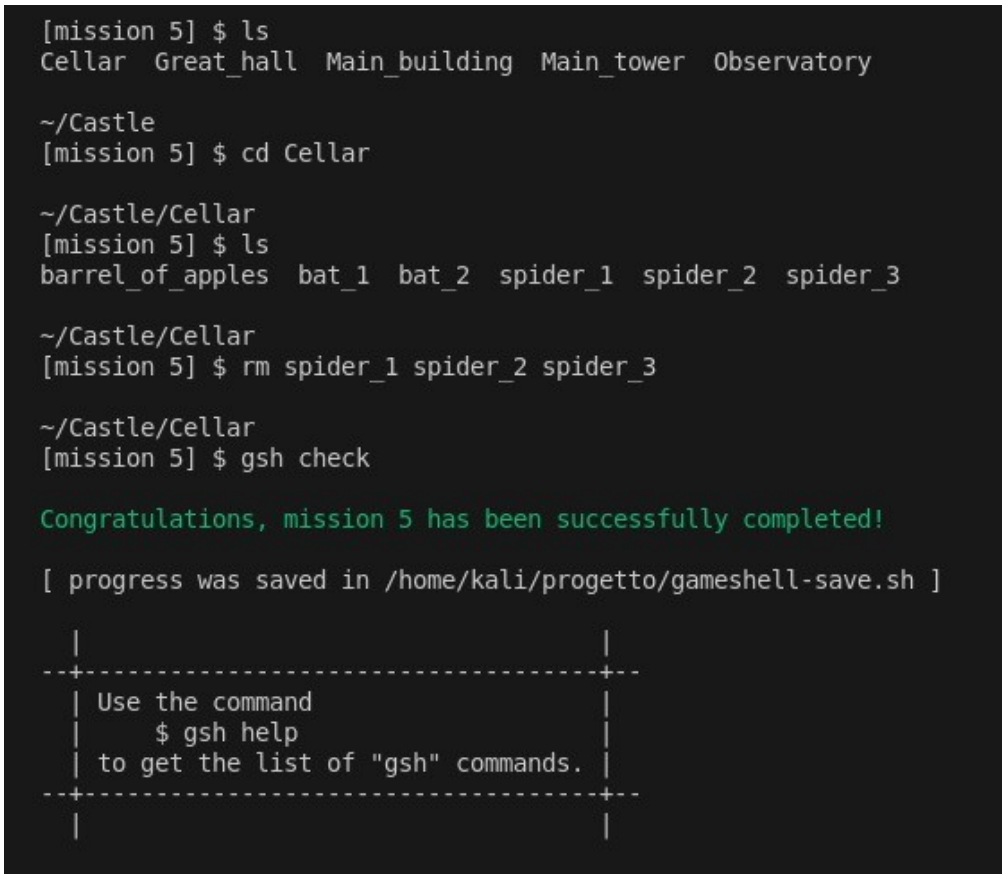


L’obbiettivo è tornare indietro nel cellar ed eliminare ragni senza disturbare i pipistrelli.

Innanzitutto mi sono recato al cellar



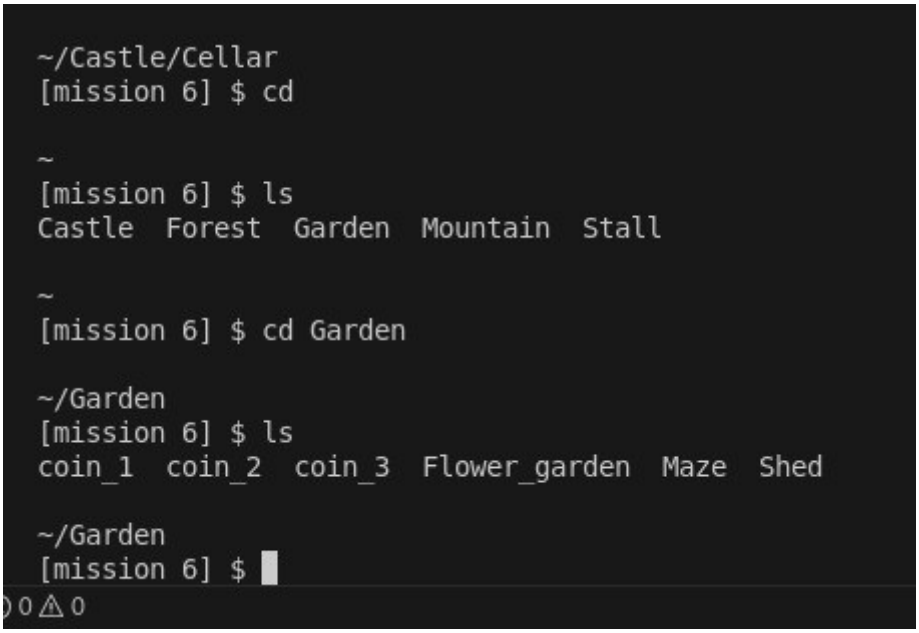
Ho rimosso i ragni tramite il comando “rm”, dopo averli visionati con “ls”.
Il check ha confermato la riuscita del comando.



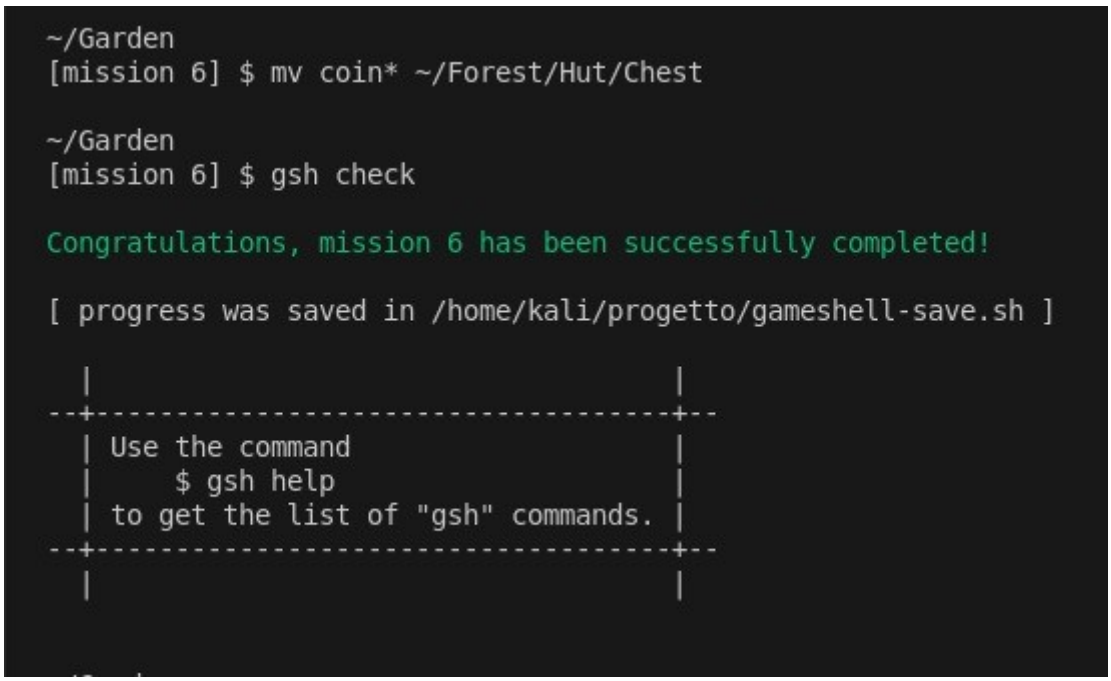
MISSIONE 6

L’obbiettivo è prendere tutte le monete nel garden e metterle nella chest.

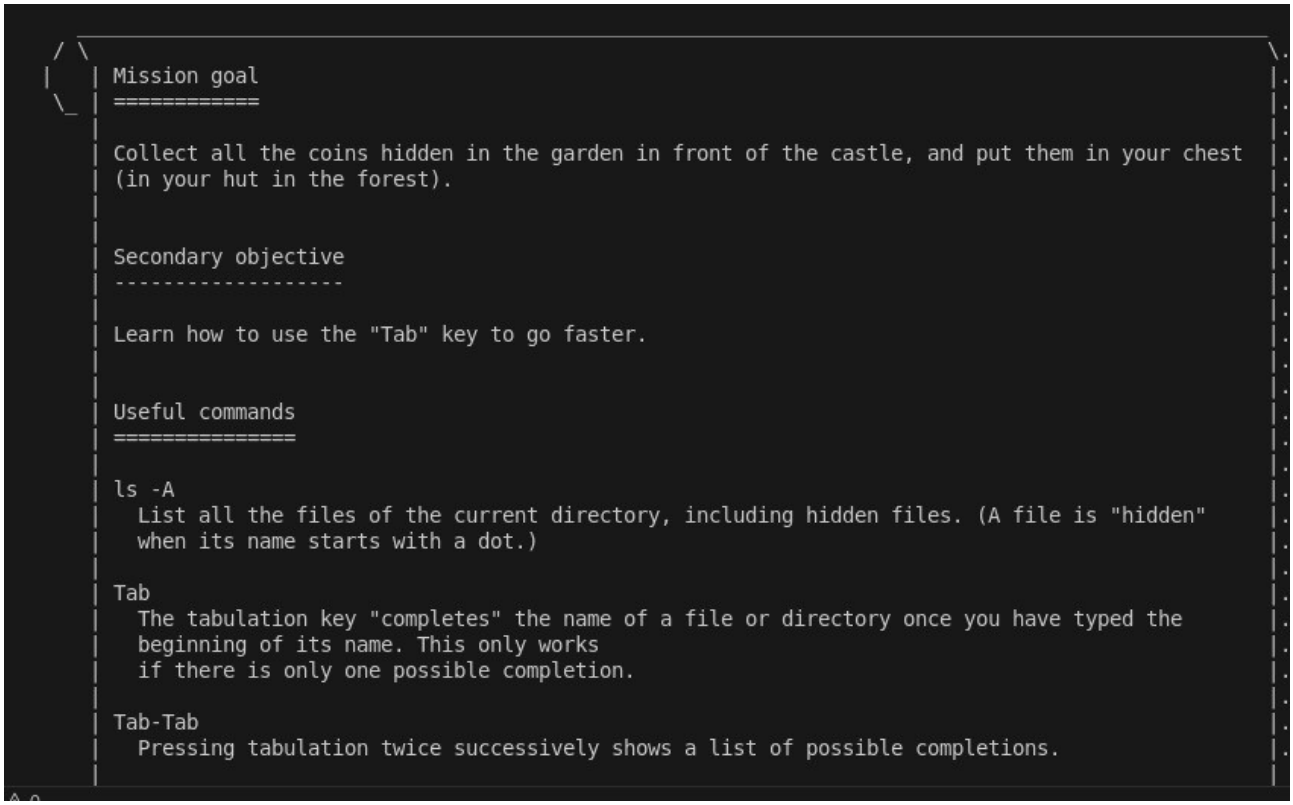
Innanzitutto, sono andato in giardino e ho visionato le monete tramite ls, insieme ad altri elementi.



“mv” sposta i file nella directory Chest. “ * “ raccoglie tutti i file con quel nome.
Così ho spostato le monete, tramite il percorso (~/Forest/Hut/Chest), alla Chest.



MISSIONE 7

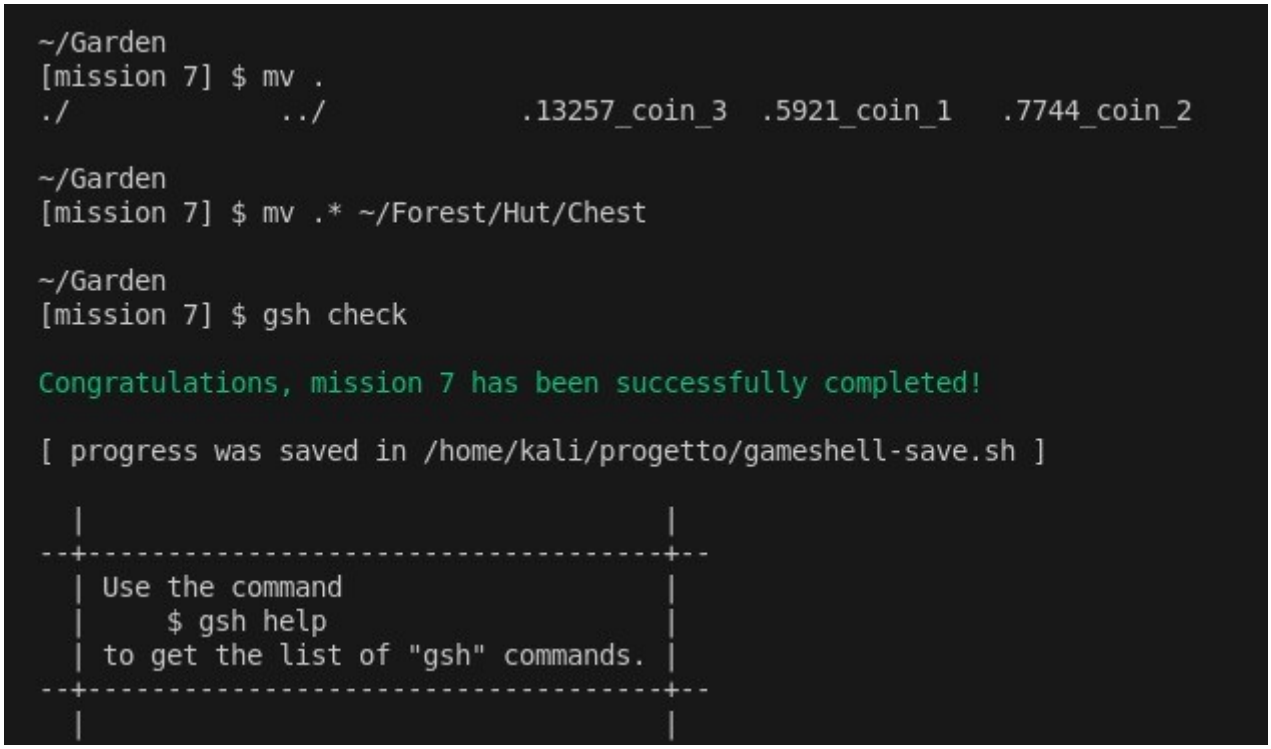


L’obbiettivo è collezionare tutte le monete nascoste in giardino e spostarle nella Chest.

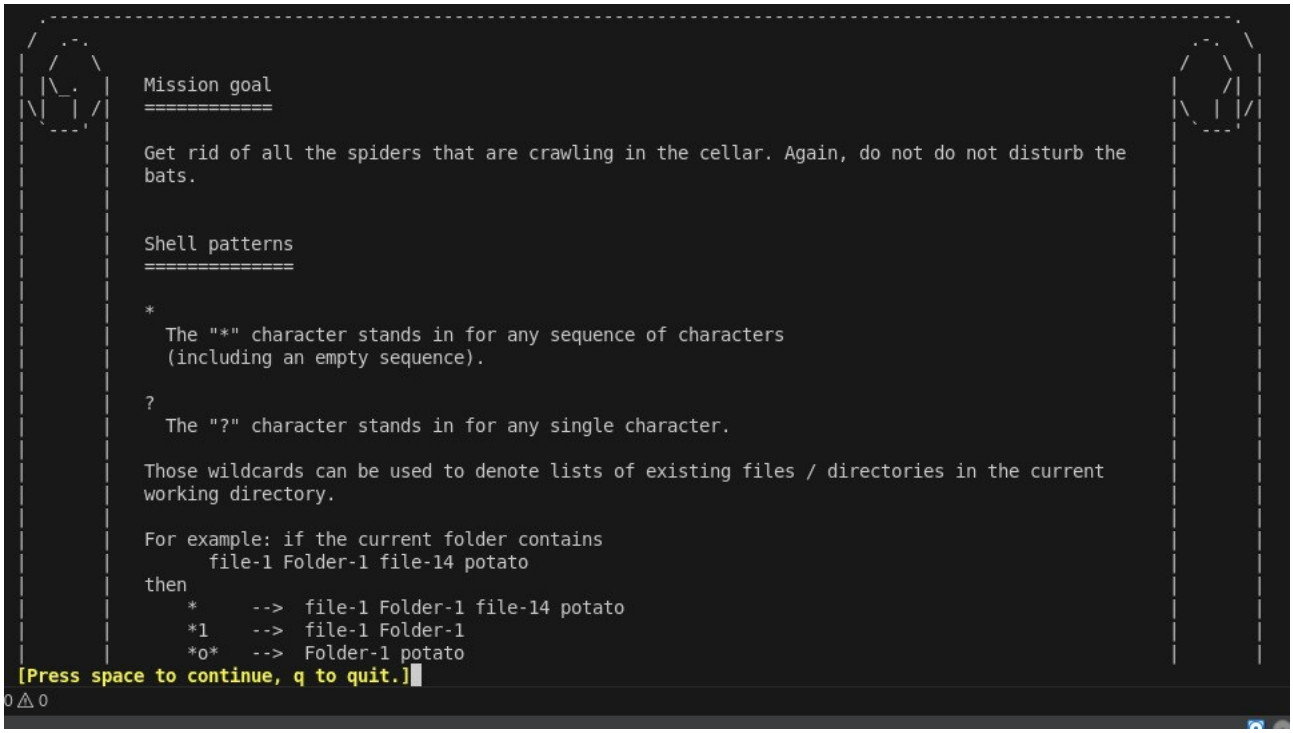
Innanzitutto sono andato in giardino e ho visionato i file nascosti tramite il comando **“ls -A”**.
Le monete sono precedute da “.nnnn_”

Il comando che ho eseguito è stato il seguente:
“mv” + “ . “ (perché le monete iniziano con “ . “) + tab. Quest’ultimo ha messo in elenco i file necessari.

Successivamente, “ **mv .*** ” , che li raggruppa tutti senza doverli selezionare uno alla volta, mi ha permesso di passare direttamente a digitare il percorso verso la Chest.

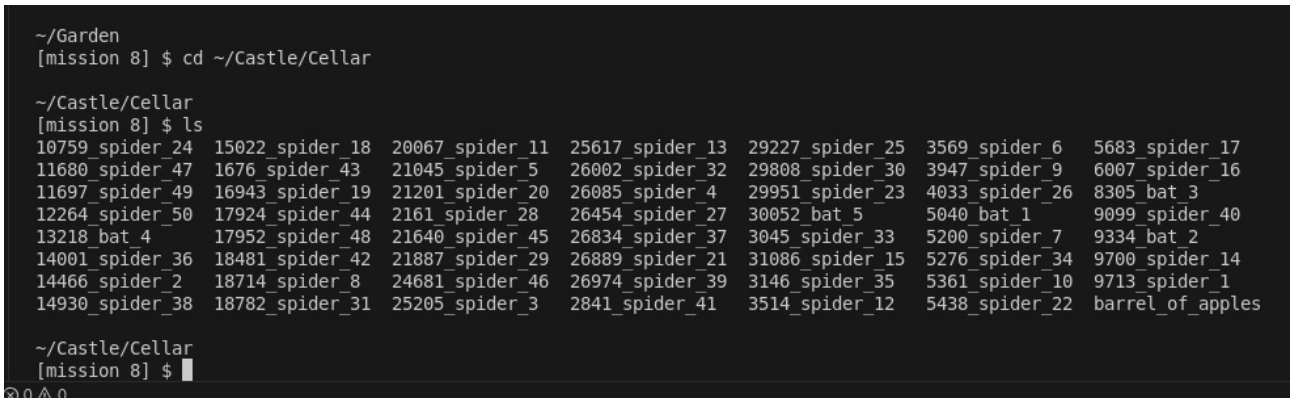


MISSIONE 8



L’obbiettivo è di disfarsi di tutti i ragni senza toccare, anche qui, i pipistrelli.

Innanzitutto ho visionato l’interno della cellar (dopo essermici recato) con “ls”. Come si vede, ogni file è preceduto da cinque numeri, seguito dal proprio nominativo.



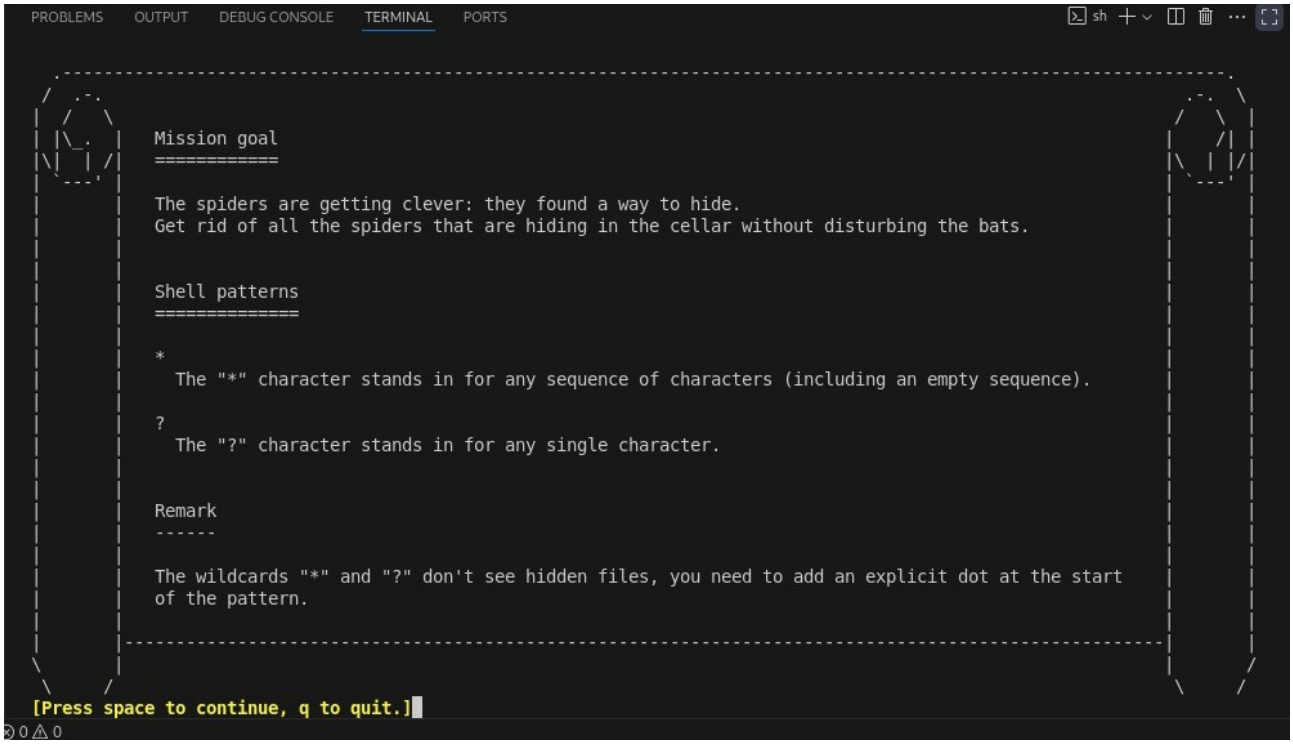
Per rimuovere tutti i ragni, il comando corretto è stato il seguente:
“ **rm *_spider_*”**

Questo perché:
Rm= remove
* = Universale per tutto ciò che precede “_”
spider = rimane il centro del nome
* = Universale per tutto ciò che segue “_”.

Dopodiché ho eseguito “ls”, per verificare che fossero rimasti solo i bat.

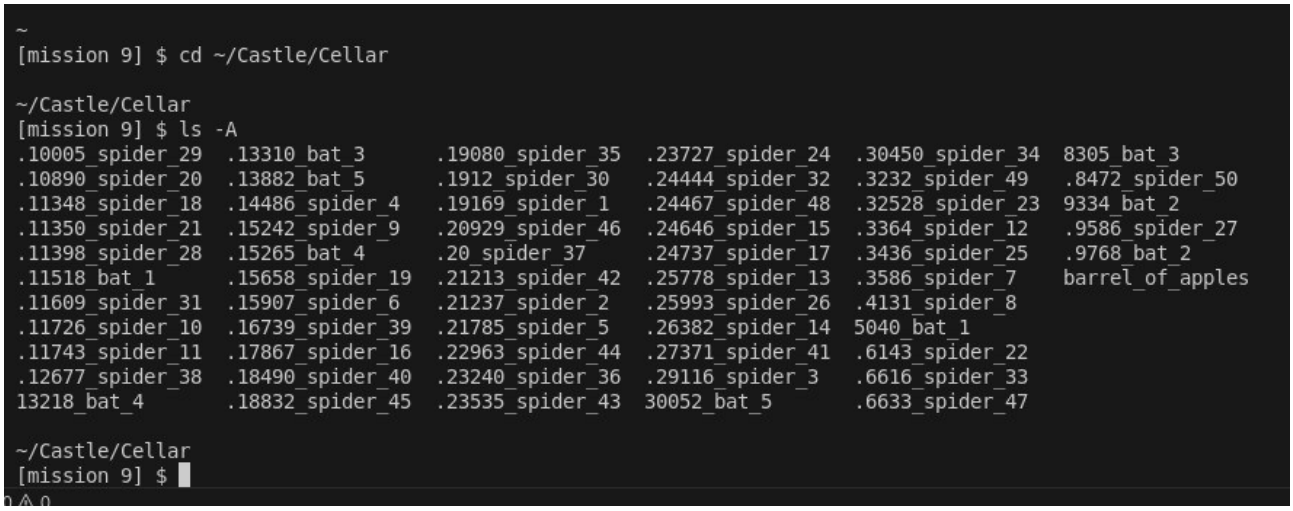
```
~/Castle/Cellar
[mission 8] $ rm *_Spider_*
rm: cannot remove '*_Spider_*': No such file or directory

~/Castle/Cellar
[mission 8] $ rm *_spider_*

~/Castle/Cellar
[mission 8] $ ls
13218_bat_4  30052_bat_5  5040_bat_1  8305_bat_3  9334_bat_2  barrel_of_apples

~/Castle/Cellar
[mission 8] $ gsh check

Congratulations, mission 8 has been successfully completed!

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

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

MISSIONE 9



L’obbiettivo è di eliminare tutti i ragni che si nascondono nella cantina senza far del male ai pipistrelli.

Dopo essermi recato nella cellar ho visionato i file nascosti tramite “ls -A”.
Come visibile, tutti i file dei ragni iniziano con “ . “



Il comando che ho strutturato per superare la missione è stato il seguente:
“**rm .[0-9]*_spider_***”

Questo perché:

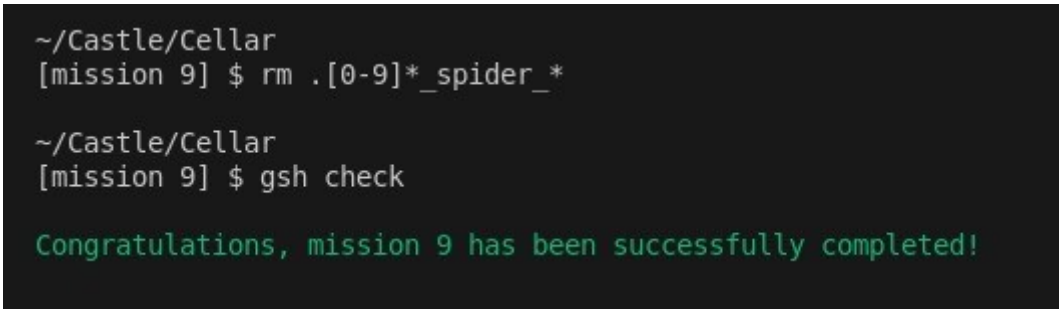
rm = remove

. (iniziale) = indica che il file è nascosto. In Linux qualunque file che inizia con “.” non viene mostrato dai normali comandi ls o dai wildcard “*” senza parametri. Tutti i ragni in questa missione sono nascosti, quindi serve il “.” per includerli.

[0-9] = significa “qualunque cifra tra lo 0 e il 9”, ossia, qualsiasi sequenza di caratteri dopo la cifra iniziale. Sostanzialmente, “ .**[0-9]*** ” cattura tutti i file nascosti che iniziano con un punto seguito da uno o più numeri.

Spider = è ciò che distingue, ovviamente, i ragni dai pipistrelli.

* (finale) = significa “qualsiasi sequenza di caratteri dopo spider”.



MISSIONE 10

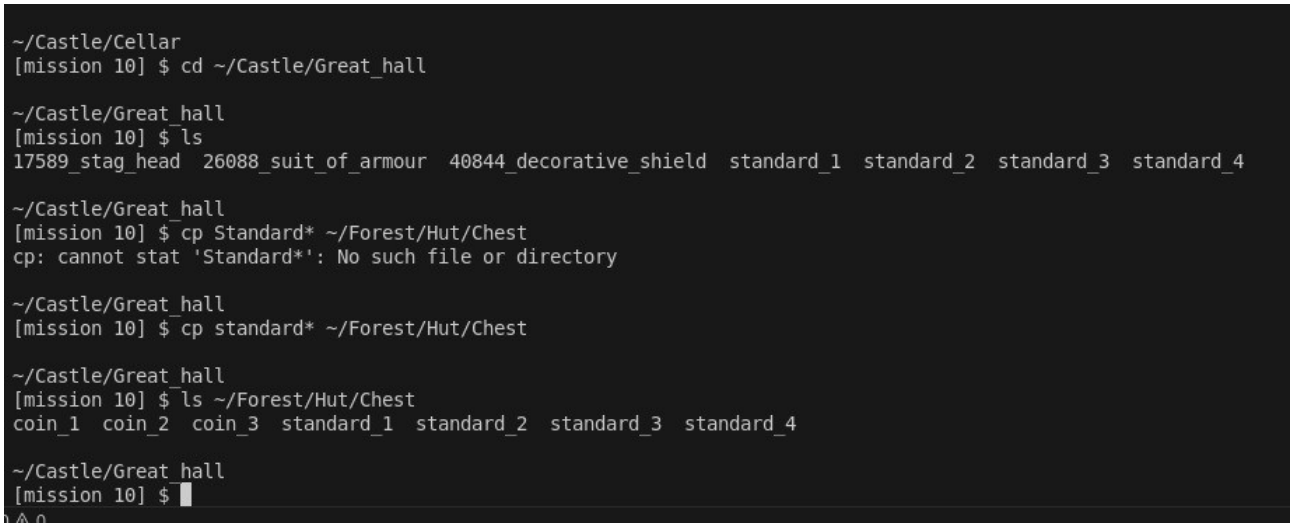
L’obbiettivo è eseguire una copia dei 4 stendardi nella Great Hall, e metterne una copia di ognuno nella Chest.

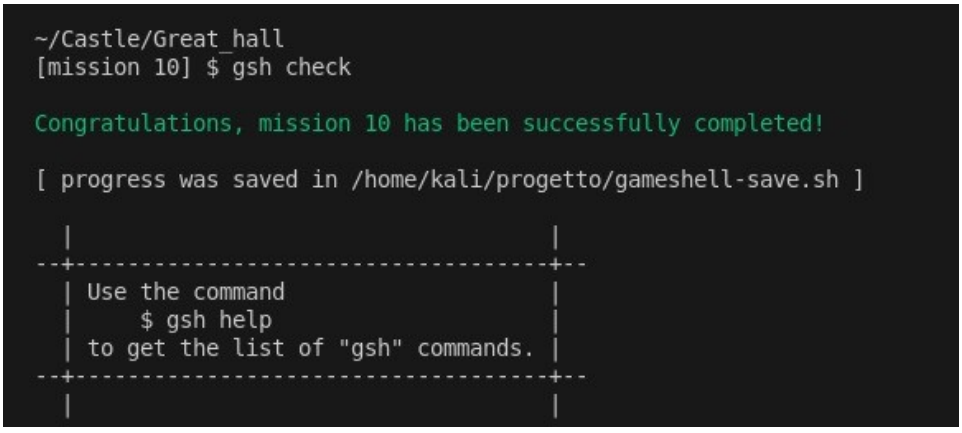
Dopo essermi recato alla Great Hall ho potuto visionare gli stendardi tramite “ls”.

Dal momento che il mio intento è stato di copiarli tutti in simultanea e spostarli verso la Chest, il comando che ho eseguito è stato il seguente:

“**cp Standard* (per copiarli tutti), + il percorso verso Chest tramite ~”**.

Ne ho poi verificato la presenza tramite “ls + percorso ~”

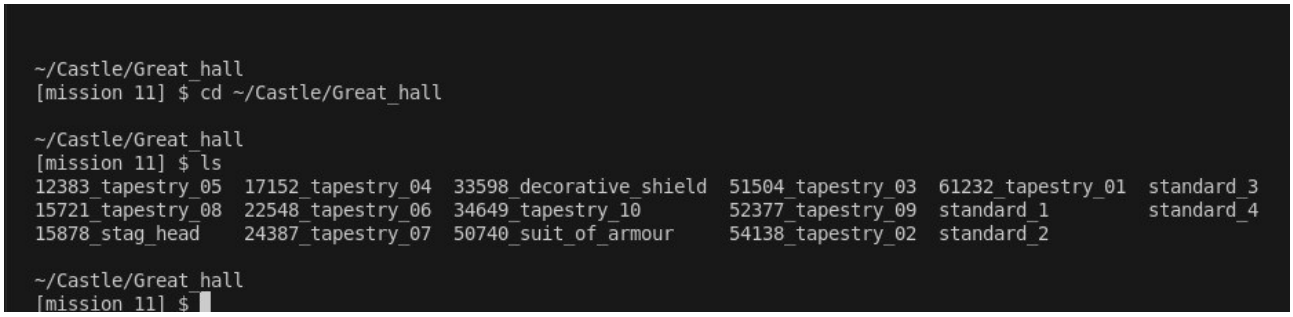




MISSIONE 11

L’obbiettivo è mettere una copia di ogni arazzo dentro la Chest.

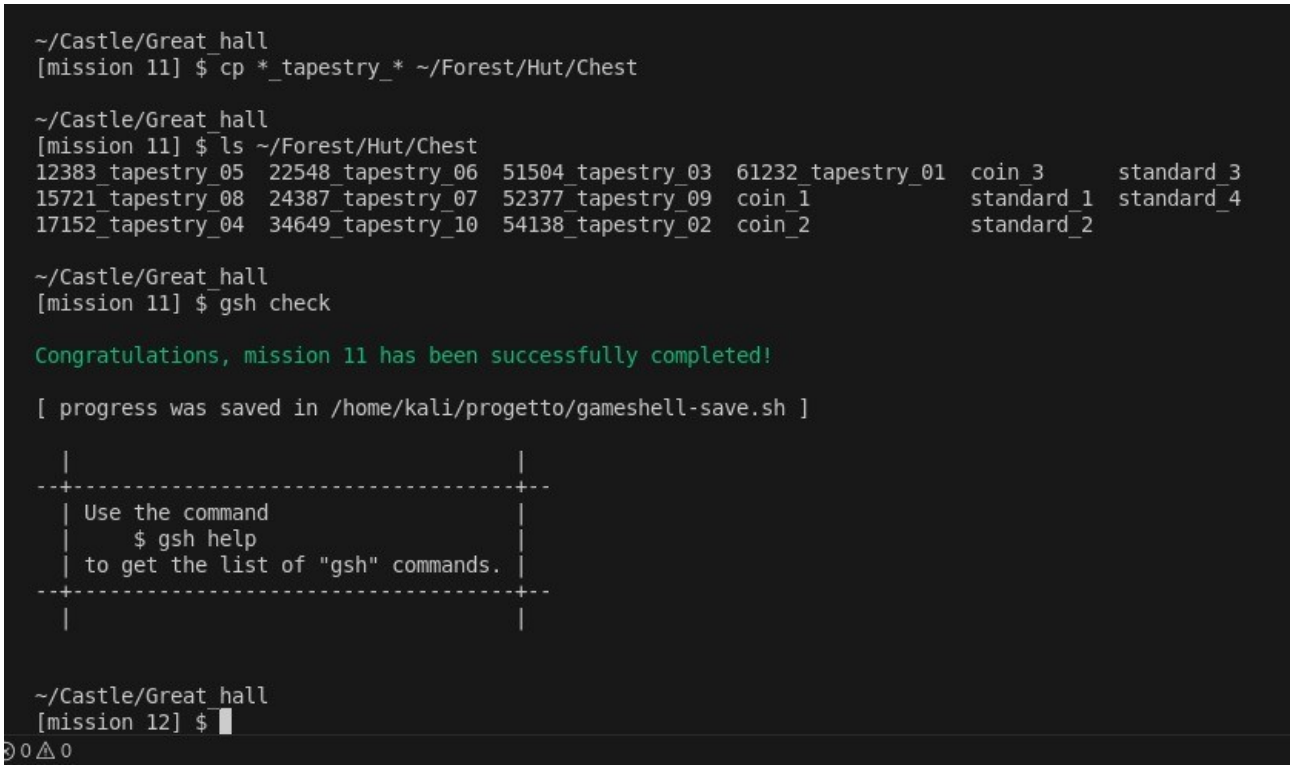
Dopo essermi recato alla Great Hall (dove si trovano gli arazzi) ho visionato questi ultimi con ls.



Il comando per passare la missione è simile a quello della missione precedente:

* = prende qualsiasi sequenza di caratteri prima o dopo “tapestry”.
~/Forest/Hut/Chest = è percorso della Chest.

Il comando è: “**cp *_tapestry_*~/Forest/Hut/Chest**”



MISSIONE 12

L’obbiettivo è aggiungere una copia del più vecchio quadro del First Floor nella Chest.

Mi sono recato al First Floor.
L’opzione **“-l”** mostra: permessi, proprietario, dimensione e data di ultima modifica
Qui il più vecchio è “painting_RCjfHHBb” , datato 1986.

```
~/Castle/Great_hall
[mission 12] $ cd ~/Castle/Main_tower/First_floor

~/Castle/Main_tower/First_floor
[mission 12] $ ls -l
total 16
-rw-rw-r-- 1 kali kali 1503 Mar  3  2002 painting_jBfTkGOb
-rw-rw-r-- 1 kali kali 1055 Sep 10  1986 painting_RCjfHHBb
-rw-rw-r-- 1 kali kali 1455 Apr 13  2013 painting_YvsVneID
drwxrwxr-x 3 kali kali 4096 Sep  6 18:01 Second_floor/
```

E’ sufficiente copiare il file in questione nella directory richiesta dalla missione e verificare che si trovi lì.

Il comando utilizzato è stato composto da:

cp = copia

painting_RCjfHHBb = il nome del quadro che ci interessa

~/Forest/Hut/Chest = il percorso fino alla chest

MISSIONE 13

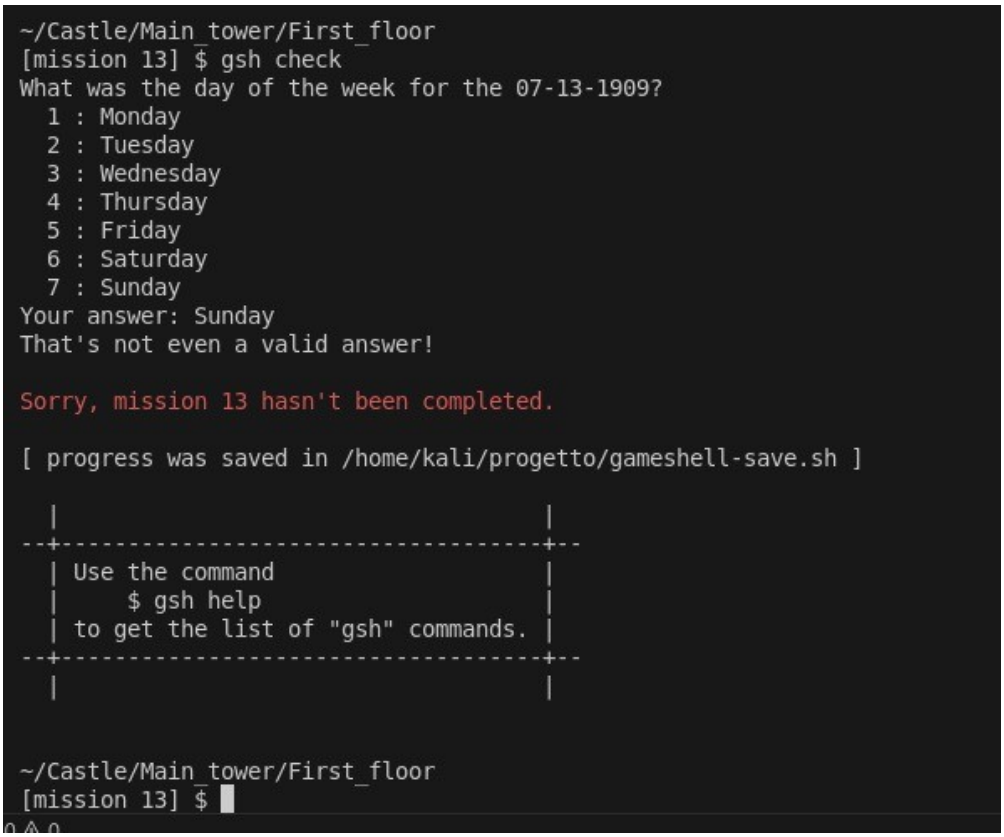
```
Mission goal
============

Nostradamus predicted a spectacular star conjunction on the 01-15-1938.
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.

This mission is optionnal. You can skip it and go to the next one with the
command

  $ gsh skip
```

L’obbiettivo è scoprire il giorno della settimana della data riportata (01-15-1938) predetta da Nostradamus.

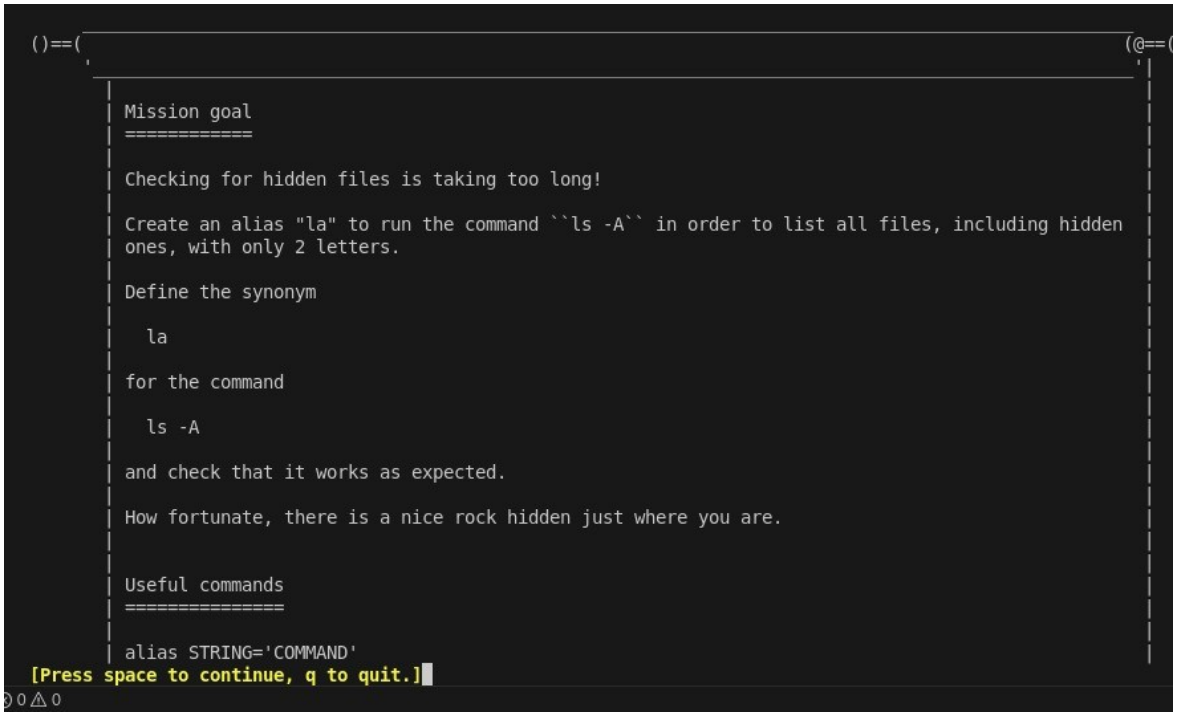
Dopo aver visionato il calendario del 1938 tramite il comando “cal 1938”, è stato possibile decretare il giorno, ossia Sabato 15 Gennaio.

```
~/Castle/Main_tower/First_floor
[mission 13] $ cal 1938
                               1938
      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         1  2  3  4  5         1  2  3  4  5
 2  3  4  5  6  7  8   6  7  8  9 10 11 12   6  7  8  9 10 11 12
 9 10 11 12 13 14 15  13 14 15 16 17 18 19  13 14 15 16 17 18 19
16 17 18 19 20 21 22  20 21 22 23 24 25 26  20 21 22 23 24 25 26
23 24 25 26 27 28 29  27 28                 27 28 29 30 31
30 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   1  2  3  4  5  6  7            1  2  3  4
 3  4  5  6  7  8  9   8  9 10 11 12 13 14   5  6  7  8  9 10 11
10 11 12 13 14 15 16  15 16 17 18 19 20 21  12 13 14 15 16 17 18
17 18 19 20 21 22 23  22 23 24 25 26 27 28  19 20 21 22 23 24 25
24 25 26 27 28 29 30  29 30 31              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      1  2  3  4  5  6                  1  2  3
 3  4  5  6  7  8  9   7  8  9 10 11 12 13   4  5  6  7  8  9 10
10 11 12 13 14 15 16  14 15 16 17 18 19 20  11 12 13 14 15 16 17
17 18 19 20 21 22 23  21 22 23 24 25 26 27  18 19 20 21 22 23 24
24 25 26 27 28 29 30  28 29 30 31           25 26 27 28 29 30
31
```

Nonostante il giorno corretto sia sabato, sia quest'ultimo, sia tutti gli altri giorni della settimana mi sono stati calcolati come errore, forse per un errore di programmazione, quindi ho saltato questa missione, che era facoltativa e skippabile.



MISSIONE 14



L’obbiettivo è creare un alias “la” per poter eseguire il comando “ls -A” così da poter listare tutti i file, inclusi quelli nascosti, con sole due lettere.

Il comando strutturato è stato il seguente:

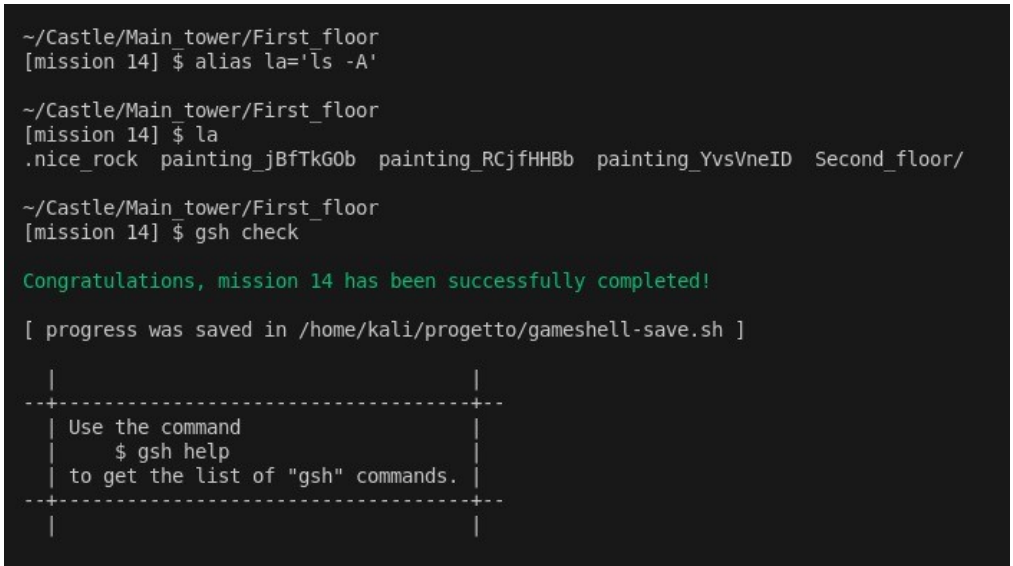
alias = comando per creare un alias.

la= il nuovo nome breve del comando.

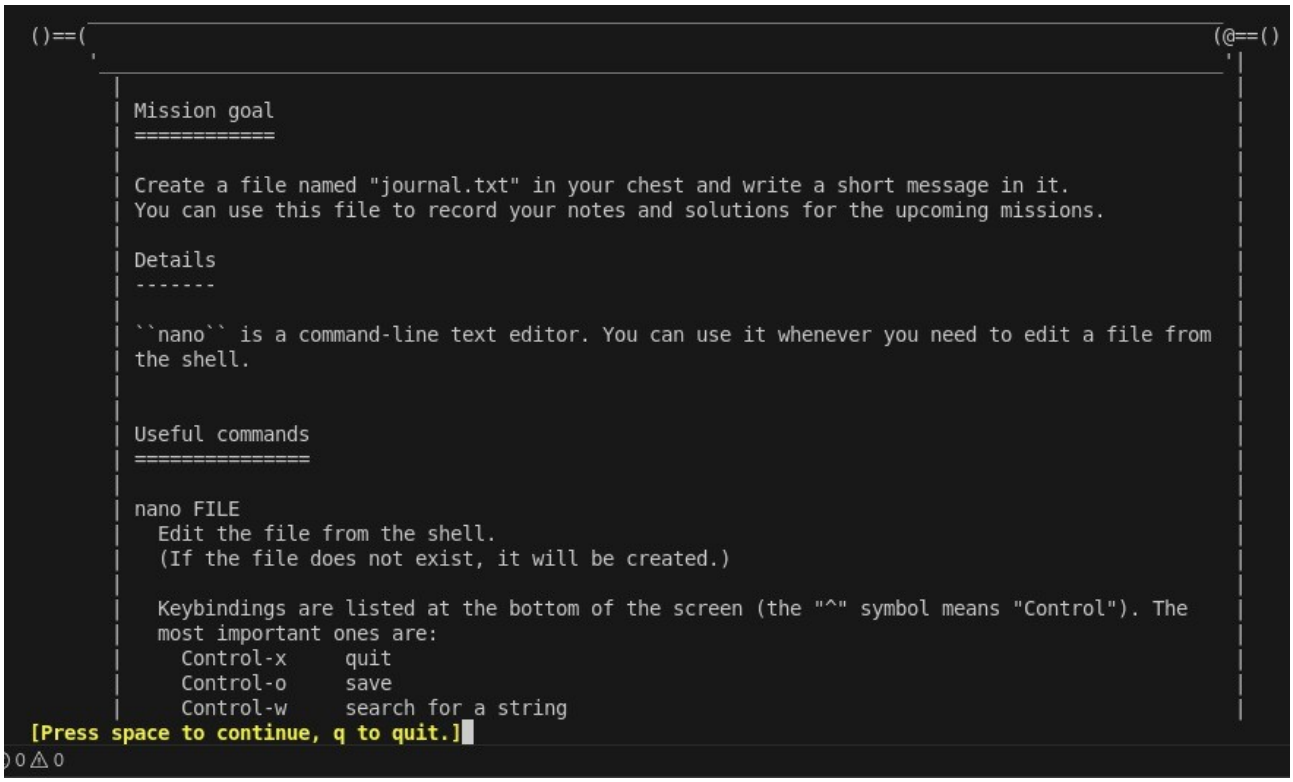
'ls -A' = il comando vero e proprio che verrà eseguito.

Il comando per intero è stato: “ **alias la=’la -A’** “

poi abbiamo testato l'alias con “la”.

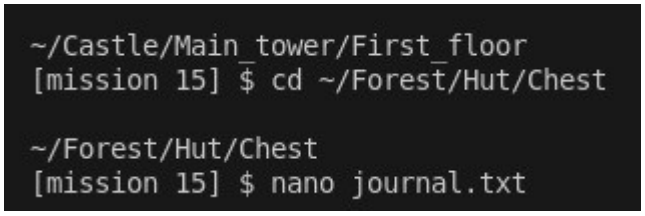


MISSIONE 15

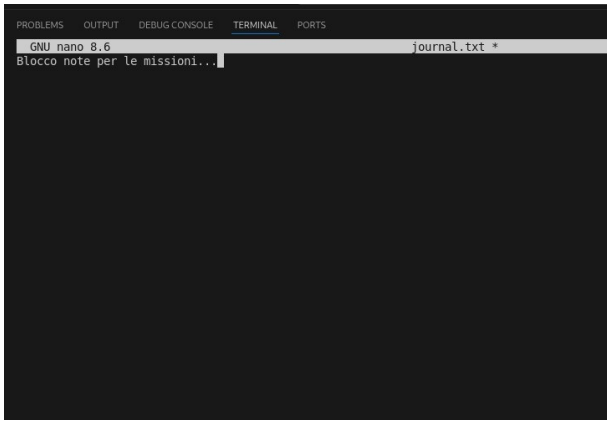


L’obbiettivo è creare un file chiamato “journal.txt” nella Chest e scriverci un messaggio. Questo file potrà essere utilizzato come blocco note per le prossime missioni.

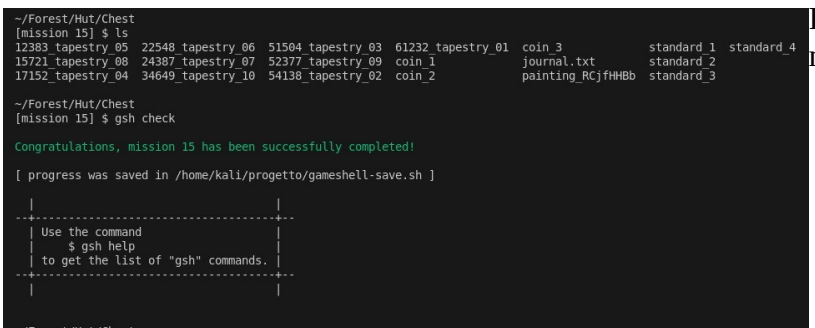
Dopo essermi recato alla Chest, ho scritto il comando “nano journal.txt”



→

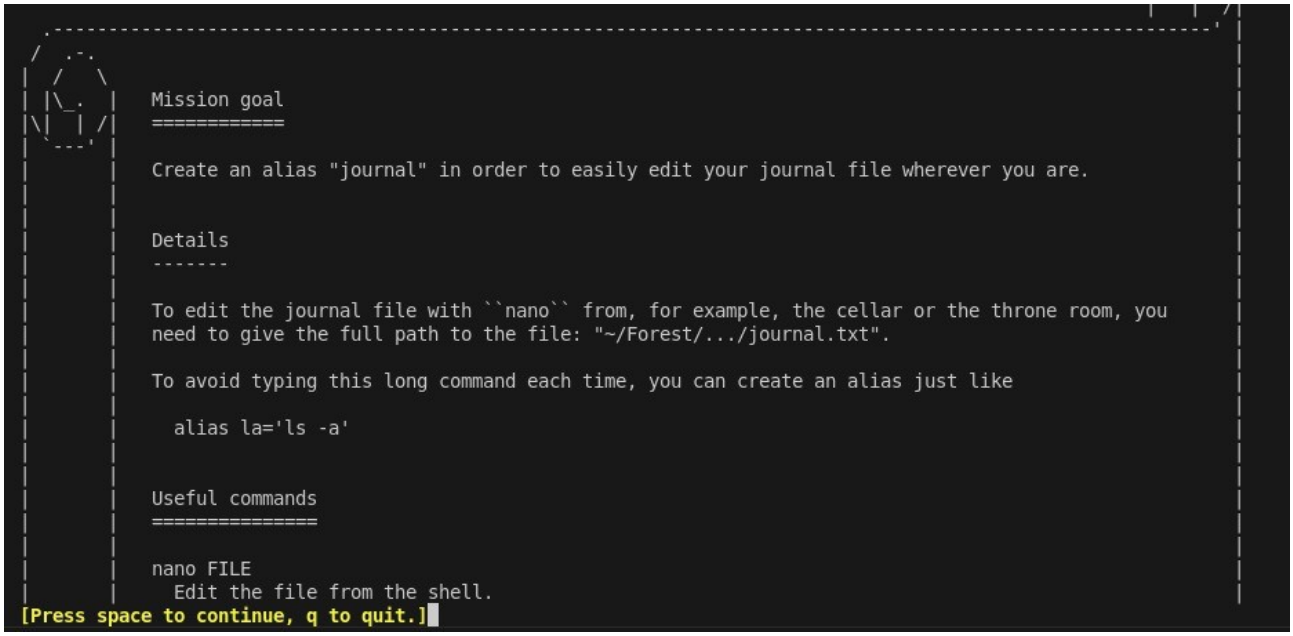


Successivamente aver scritto il messaggio, sono uscito, salvando.
Ne ho verificato la presenza nella Chest ed ho completato la missione.



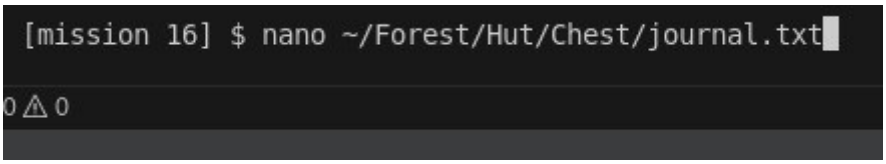
Per leggere il contenuto in qualsiasi momento: **cat journal.txt**

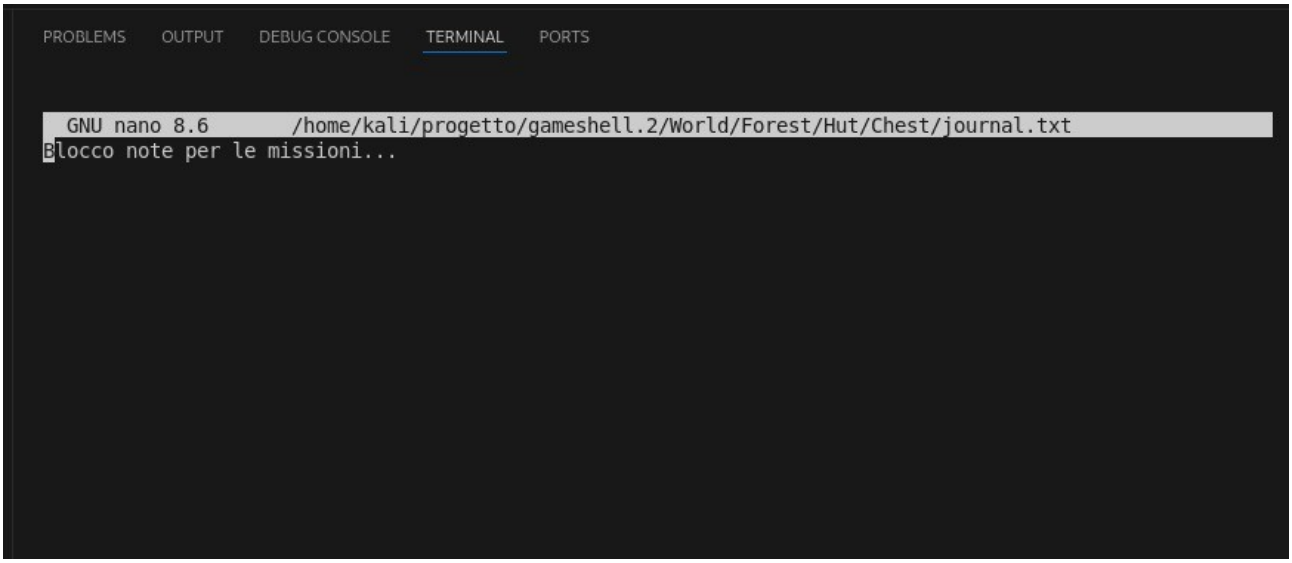
MISSIONE 16



L’obbiettivo è creare un alias chiamato “journal”
L’alias deve permettere di aprire il file journal.txt con nano da qualsiasi posizione.

Partiamo dal presupposto che il comando “nano ~/Forest/Hut/Chest/journal.txt” apre il nostro file di testo, come visibile.



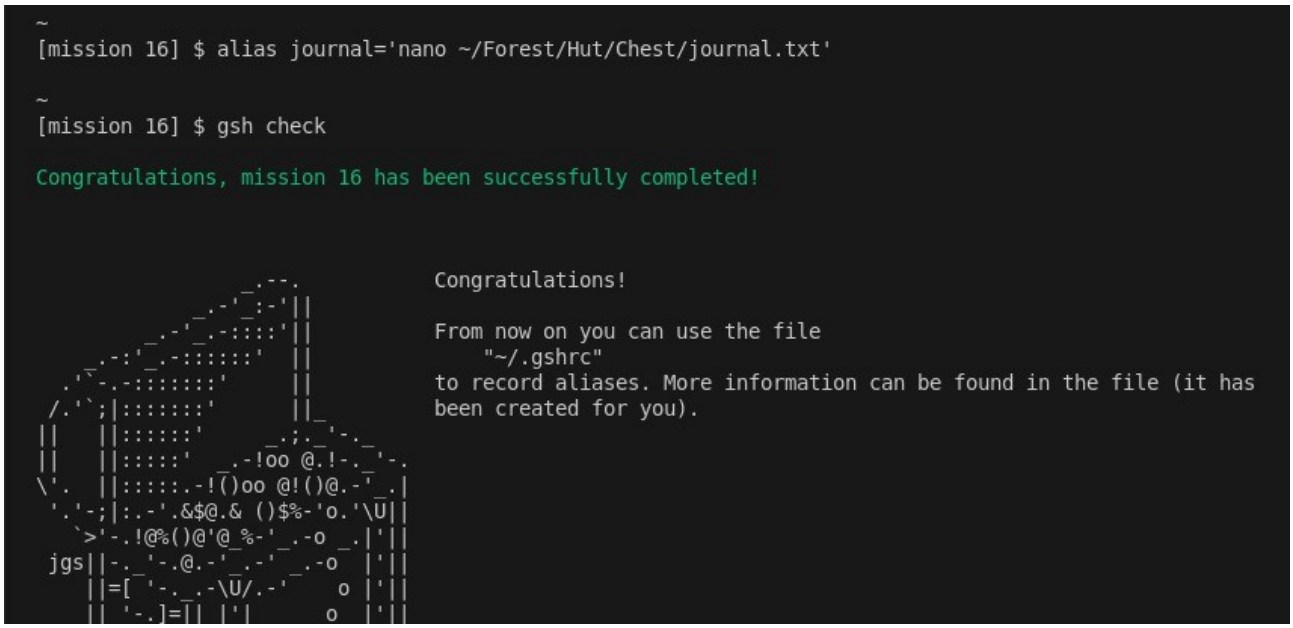


Il comando per risolvere la missione è stato da me composto così:

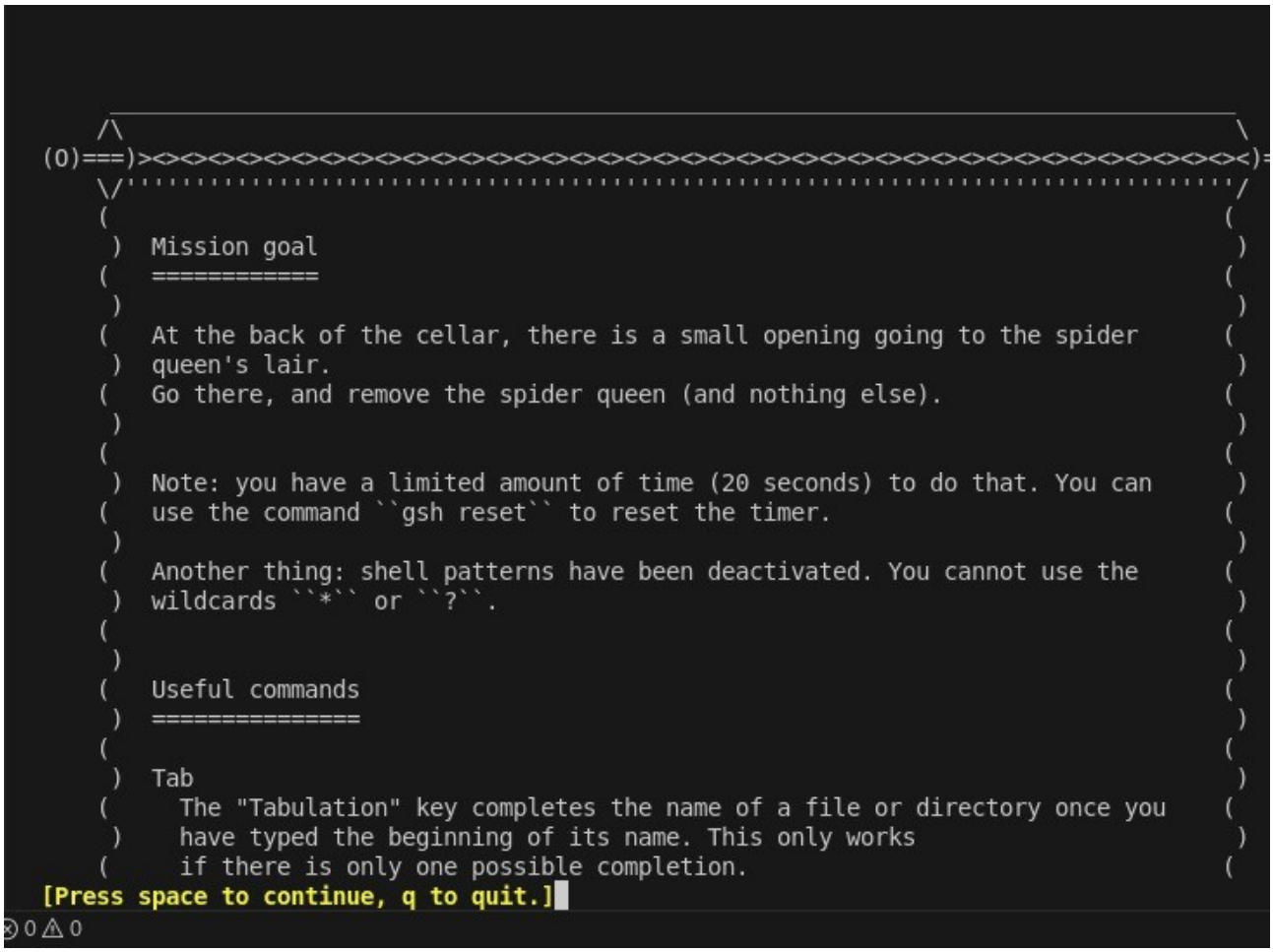
“alias” = il comando della shell Linux che crea un “soprannome” per un comando lungo o complicato.
Serve a risparmiare tempo: invece di scrivere tutto il percorso e il comando ogni volta, si utilizza l’alias.

Journal = questo è il nome dell’alias. Quando scriviamo journal nel terminale, la shell sostituirà automaticamente questo nome con il comando completo che abbiamo definito.

'**nano ~/Forest/Hut/Chest/journal.txt**' = tutto ciò tra le virgolette singole è il comando che verrà eseguito quando digiteremo “journal”.



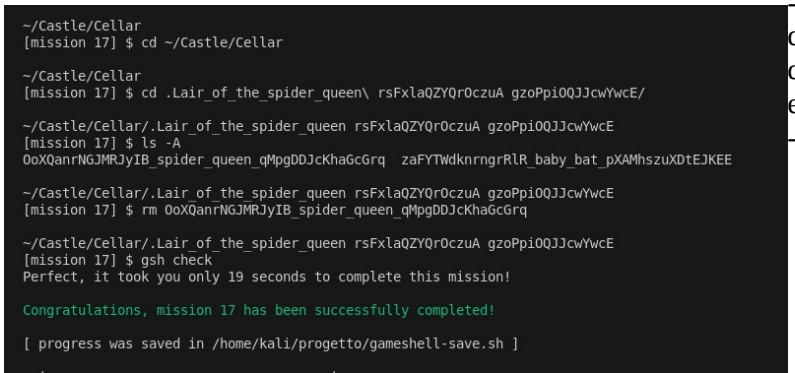
MISSIONE 17



L’obbiettivo è entrare nella tana della regina ragno (spider queen) nel retro della Cellar.
-Bisogna rimuovere solo la spider queen, senza eliminare altri ragni.
-Si hanno solo 20 secondi per completare la missione.
-Non si possono usare i wildcards * o ?, cioè niente rm *.txt o simili.

Si può utilizzare usare il comando gsh reset per resettare il timer in caso di errore.

VELOCEMENTE:
-Mi sono spostato nella Cellar
-Ho visualizzato tutti i file,inclusi quelli nascosti con “ls -A” (**ls** mostra i file presenti nella directory. L’opzione **-A** mostra anche i file nascosti,quelli che iniziano con un punto. In questa lista ci sono i pipistrelli e la directory della spider queen).
-Sono entrato nella directory della spider queen (con “cd .Lair[...] + tab”, per completamento veloce) Questo comando mi ha portato dentro la tana.
-Ho visualizzato i file all’interno della tana (ls -A)

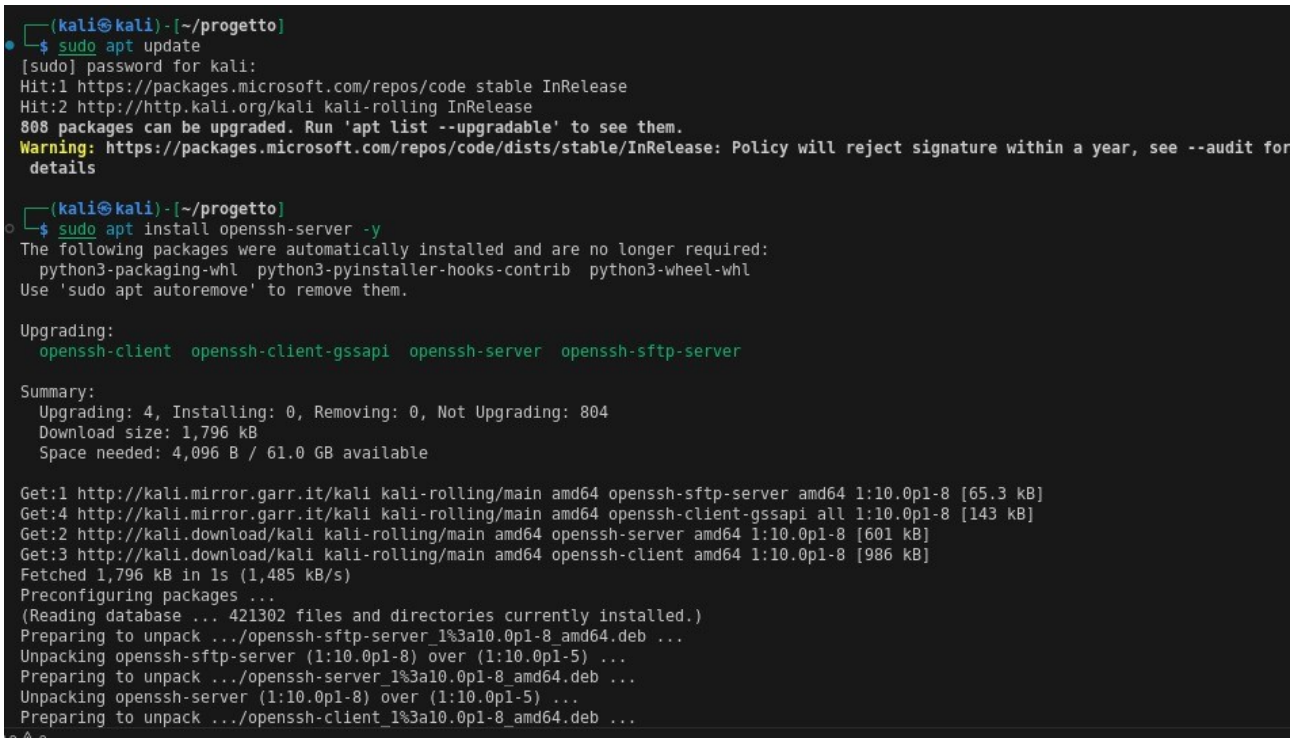


-Ho trovato e rimosso il file della spider queen (**rm .12345_spider_queen**). Dopo questo comando, la spider queen è eliminata.
-Verifica con il check.

TASK 2

L’obiettivo di questo secondo task è creare uno script in Python che esegua un attacco brute-force SSH su una macchina Debian/Ubuntu/Kali controllata dall’utente.

1. Intanto ho verificato che Server SSH fosse installato e attivo sulla macchina locale.



```
┌──(kali㉿kali)-[~/progetto]
└─$ sudo systemctl enable ssh
Synchronizing state of ssh.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable ssh
Created symlink '/etc/systemd/system/sshd.service' → '/usr/lib/systemd/system/ssh.service'.
Created symlink '/etc/systemd/system/multi-user.target.wants/ssh.service' → '/usr/lib/systemd/system/ssh.service'.

┌──(kali㉿kali)-[~/progetto]
└─$ sudo systemctl start ssh

┌──(kali㉿kali)-[~/progetto]
└─$ ssh localhost

The authenticity of host 'localhost (::1)' can't be established.
ED25519 key fingerprint is SHA256:IwdYJP2r7u+K178OqPgTiGaFU/gX3zbyD7A1DKJmkZQ.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'localhost' (ED25519) to the list of known hosts.
kali@localhost's password:
Linux kali 6.12.33+kali-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.12.33-1kali1 (2025-06-25) x86_64

The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
┌──(kali㉿kali)-[~]
└─$
```

2. Ho proceduto alla creazione della wordlist di password. Dentro la cartella “progetto”, ho eseguito il comando “nano passwords.txt”, scrivendo, di quattro password, tre false e una reale (la mia).
Il contenuto è verificabile in qualunque momento tramite “cat passwords.txt”.

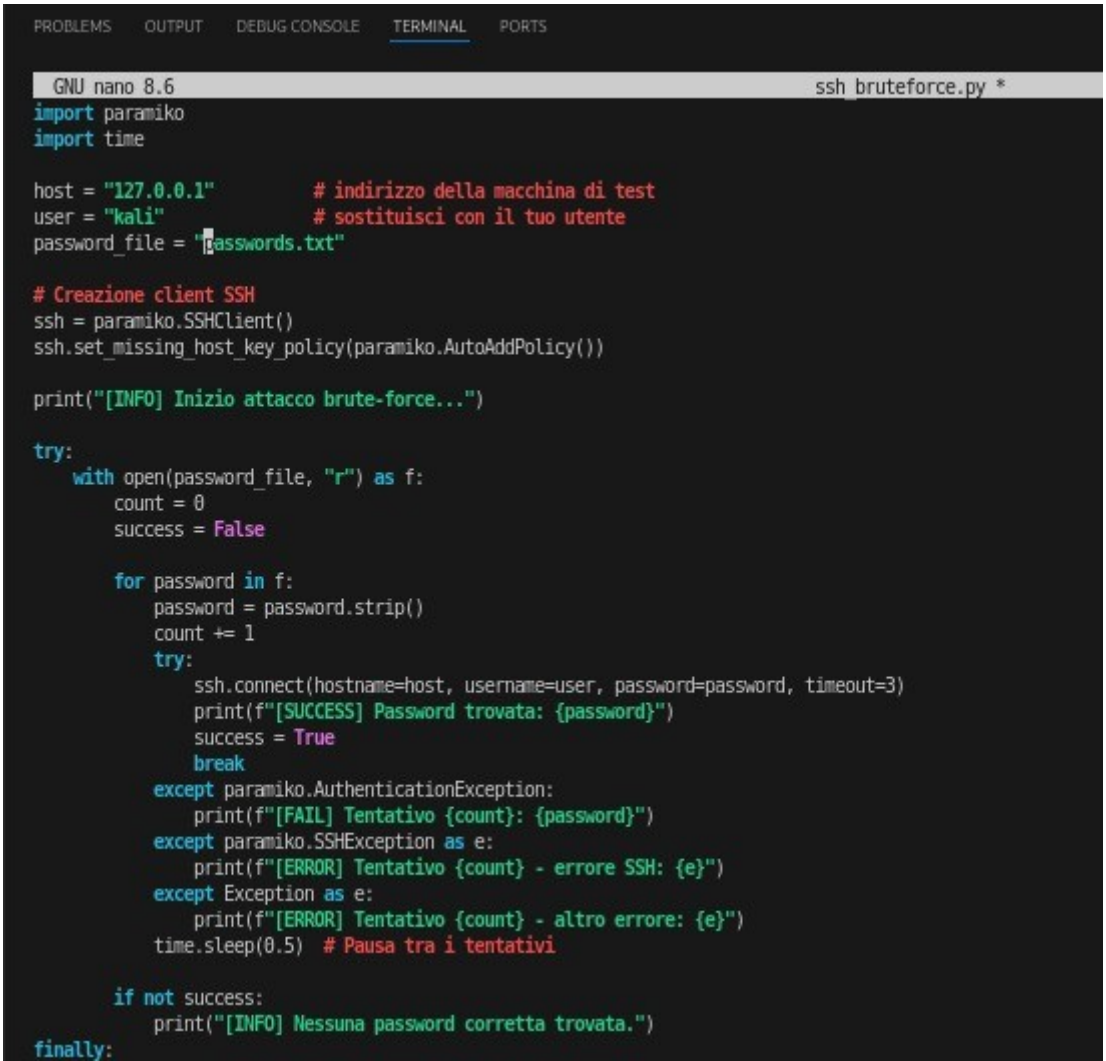
```
┌──(kali㉿kali)-[~]
└─$ cd /home/kali/progetto

┌──(kali㉿kali)-[~/progetto]
└─$ nano passwords.txt
```

```
  GNU nano 8.6
4321
drowssap
ilak321
kali
```

3.Mi sono procurato Paramiko, libreria Python per SSH, tramite “sudo apt install python3-paramiko -y”

```
┌──(kali㉿kali)-[~]
└─$ sudo apt install python3-paramiko -y

[sudo] password for kali:
python3-paramiko is already the newest version (3.5.1-3).
python3-paramiko set to manually installed.
The following packages were automatically installed and are no longer required:
  python3-packaging-whl  python3-pyinstaller-hooks-contrib  python3-wheel-whl
Use 'sudo apt autoremove' to remove them.

Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 801

┌──(kali㉿kali)-[~]
└─$ nano passwords.txt
```

4. Ho proceduto alla scrittura dello script Python su “**nano ssh_bruteforce.py**”



Come ho strutturato lo script:

1.import paramiko = gestisce la connessione SSH in Python.
Import time = permette di inserire pause tra i tentativi, per non sovraccaricare il server.

2. host = indirizzo della macchina di test.
user = utente SSH della VM.
password_file = file che contiene la lista delle password da provare.

3. **SSHClient()** = permette di gestire le connessioni SSH in modo programmato.
set_missing_host_key_policy(AutoAddPolicy()) = consente al client di accettare automaticamente le chiavi degli host sconosciuti, evitando errori di autenticità della macchina di test. In pratica, questa configurazione assicura che lo script possa tentare la connessione anche se non si è mai connessi prima a quella macchina, senza richiedere conferme manuali.

Nello specifico, AutoAddPolicy() è una politica predefinita di Paramiko che dice: “Se la chiave del server non è presente nel database delle chiavi conosciute, aggiungila automaticamente e continua la connessione.”
4. Print del messaggio iniziale, che comunica all’utente l’inizio dell’esecuzione dello script.

5. Lettura del file password:
open(password_file, "r") = apre il file delle password in modalità lettura ("r").
with ... as f = utilizza un context manager, che chiude automaticamente il file alla fine del blocco.
count = tiene traccia di quanti tentativi sono stati effettuati.
Success = variabile che indica se è stata trovata la password corretta.

or password in f: = legge una riga alla volta dal file f. Ogni riga contiene una password.
password.strip() = rimuove eventuali spazi bianchi e caratteri di nuova linea (\n) all’inizio o alla fine della riga. Serve perché spesso nei file di testo ci sono spazi o ritorni a capo che altrimenti rovinerebbero la connessione SSH.
count += 1 = aumenta il contatore dei tentativi, utile per stampare messaggi tipo [FAIL] Tentativo 3: password.

6. Tentativo di connessione SSH
Tenta di connettersi con la password corrente.
Se va a buon fine: stampa [SUCCESS], aggiorna success = True e interrompe il ciclo.

7. Gestione degli errori (except).
AuthenticationException = password sbagliata.
SSHException = problemi generali SSH (es. handshake fallito).
Exception = altri errori imprevisti.

8. Pausa tra tentativi (time.sleep(0.5))

9. Messaggio “if not success”, se nessuna macchina funziona.

10. Chiusura della connessione (finally):
    Chiude il client SSH indipendentemente dall’esito dei tentativi.
    Stampa un messaggio di conclusione.

5. Infine l’ho salvato, l’ho eseguito (python3 ssh_bruteforce.py) e ho visionato i risultati: