Gioco completato come da screen, in allegato ci sono anche tutti i comandi che ho utilizzato per completare le missioni.

SPOSTARS TRA LE CARTELLE Con il 'cd -' andiamo ella directory precedente con il 'cd ..' torniamo indietro di una directory con 'cd' torniamo all directory sorgente con il '~/' evitiamo di scrivere /home/kali/etc fino alla directory principale

CALENDARIO 'cal' calendario di questo anno 'cal ANNO' per vedere il calendario di uno specifico anno

VISUALIZZARE I FILE NELLA SHELL

'cat FILE' fa visualizzare tutto il contenuto all'interno della shell 'cat FILE1 FILE2 FILE3' permette la visualizzazione di 3 file o piu 'head' restituisce le prime righe di un file di testo. Di default, mostra le prime 10 righe

'head -n FILE.TXT' mostrerà le prime 20 righe del file file.txt 'tail' restituisce le ultime righe di un file di testo. Di default, mostra le ultime 10 righe 'tail -n 15 file.txt' mostrerà le ultime 15 righe del file file.txt.

‘less’ permette di visualizzare il file direttamente nella console: **q** per uscire. **/** per ricercare parole. **space** per andare avanti di pagina

SCORCIATOIE 'alias nome_scorciatoia='nano /home/kali/windows/castle etc etc'' '~/.gshrc' serve per visualizzare la lista degli alias ed inserirsli

ESEGUIRE UN PROGRAMMA 'programma' 'programma&' e commerciale alla fine per avviarlo in background

‘./FILE’ aprire un file come un programma

VEDERE CONTENUTO DI TUTTE LE DIRECTORY 'ls -R' fa visualizzare tutte le cartelle e sotto cartelle 'tree' visualizza come albero genealogico

CERCARE QUALCOSA ALL'INTERNO DELLE DIRECTORY 'find’ funziona con la seguente stringa: ‘find /percorso -opzioni criteri’ (’find . -opzioni criteri’ il punto sostituisce il percorso corrente 'find /percorso -type f -iname *parola’ con il -type f specifichiamo la ricerca di un file e con -iname il nome del file

‘find /percorso -type f -size +1M’ Trova tutti i file più grandi di 1 MB nella directory scelta

‘find . -type d -name nome_directory’ trova il percorso della directory, -typer d indica il cercare una directory

IL PIPE '|' utilizzato per collegare l'output di un comando all'input di un altro comando 'head -n 20 FILE.txt | tail -n +10' vogliamo visualizzare solo le righe comprese tra la riga 10 e la riga 20.

In questo comando, head -n 20 file.txt mostra le prime 20 righe del file file.txt, e quindi la pipe | invia queste righe a tail -n +10, che visualizza tutte le righe a partire dalla decima fino alla ventesima. 'tail -n 50 FILE.txt | head -n 10' vogliamo visualizzare le ultime 50 righe, quindi le prime 10 righe di queste. In questo comando, tail -n 50 logfile.txt mostra le ultime

50 righe del file di log logfile.txt, e quindi la pipe | invia queste righe a head -n 10, che visualizza solo le prime 10 di queste righe.

VISUALIZZARE I PROCESSI 'ps' crea lista di tutti i processi associati all' utente corrente nella shell corrente. 'ps -e' è possibile visualizzare una lista di tutti i processi di tutti gli utenti. 'ps -F' è possibile visualizzare informazioni più dettagliate, inclusi i dettagli dell'ambiente, dei percorsi dei comandi, dei tempi di avvio, ecc. 'ps' -l' alcuni dettagli

'ps PID' permette di visualizzare informazione dettaglite su numero del processo inserito 'ps -s' permette di vedere lo stato dei processi PENDING,BLOCKED,IGNORED,CAUGHT

KILLARE UN PROCESSI 'kill PID' permette di arrestare il processo corrispondente al PID inserito 'kill -s KILL [PID]' o con la sua forma abbreviata 'kill -9 PID' permette di arrestare un processo immediatamente senza che esse avvia la procedura di uscita, può portare alla perdita di dati 'kill -s TERM [PID]' serve per terminare nella maniera corretta un processo, è CONSIGLIATO USARE QUESTO

I PROCESSI 'pstree' crea un albero genealogico con tutti i processi avviati dall'utente 'pstree -p [PID]' permette di vedere la strutta genealogica del processo indicando il numero di PID 'pstree -s PID permette di vedere tutti i processi associati a quel PID sin dall'origine 'pstree -p $$' viene utilizzato per visualizzare l'albero dei processi a partire dal processo corrente, '$$' è un valore che sostituisce il processo corrente

COPIARE IN MODO ALTERNATIVO

'comando < FILE_INPUT' stampa l'input del comando eseguito all'interno del file su schermo. ES: 'awk '{sum += $1} END {print sum}' < numeri.txt' = legge i numeri da numeri.txt, li somma utilizzando awk e stampa il risultato.

'comando > FILE_OUTPUT' invece di far comparire a schermo l'output richiesto lo invia all'interno del FILE_OUTPUT

'comando 2> FILE_OUTPUT' invia gli **errori standard** dentro il file FILE_OUTPUT', utile questo è utile quando si desidera registrare gli errori per l'analisi o il debug successivo.

'ls FILE' elencherà le informazioni sul FILE nella directory corrente.

'ls FILE > percorso/FILE.txt' invece di visualizzare a schermo permette di copiare il contenuto dentro il percorso scelto sovrascrivendo il contenuto

'ls FILE >> percorso/FILE.txt' con il doppio segno maiuscolo non sovrascrive il file

`/dev/null`, i dati vengono semplicemente eliminati in un buco nero **la sintassi precisa è di sotto**

'comando > /dev/null/' crea un buco nero che scarta l'output standard.

'comando > /dev/null/' crea un buco nero che scarta gli errori standard.

CERCARE PAROLE DENTRO I FILE

‘grep PAROLA FILE’ cerca le stringhe dei file che contengono quella parola

‘grep -i PAROLA FILE’ cercherà la parola indifferentemente se è scritta maiuscola o minuscola

‘grep -l PAROLA FILE’ Questo comando restituirà solo i nomi dei file che contengono la parola specificata, anziché le stringhe effettive in cui è presente il modello

CERCARE PAROLE DENTRO LE DIRECTORY E LE SUB DIRECTORY

‘find . -type f -exec grep -H parola_da_cercare '{}' \;’

- `find .`: Avvia la ricerca partendo dalla directory corrente e tutte le sue sotto-directory.
- `type f`: Limita la ricerca ai soli file.
- `exec grep -H parola_da_cercare '{}' \;`: Per ciascun file trovato, esegue il comando `grep -H parola_da_cercare '{}'`, dove:
  - `grep`: È un comando utilizzato per cercare stringhe all'interno di file.
  - `H`: Mostra il nome del file insieme alle corrispondenze trovate.
  - `parola_da_cercare`: È la parola che `grep` cerca all'interno di ciascun file.
  - `'{}'`: Viene sostituito con il nome del file corrente trovato da `find`.
  - `\;`: Indica la fine del comando `exec`.

‘find . -type f -exec grep -l parola_da_cercare '{}' \; | xargs mv -v -t ~/’

1. `find . -type f -exec grep -l parola_da_cercare '{}' \;`:
   - Trova tutti i file (non directory) nella directory corrente (`.`) e nelle sue sotto-directory.
   - Per ciascun file trovato, esegue `grep -l parola_da_cercare '{}'`, dove:
     - `grep -l parola_da_cercare '{}'` cerca la parola all'interno di ciascun file.
     - `l` fa sì che `grep` restituisca solo il nome del file anziché le righe che corrispondono.
   - L'output di questo comando sarà una lista di file che contengono la parola
2. `| xargs mv -v -t ~/`:
   - L'operatore `|` (pipe) collega l'output del primo comando all'input del secondo.
   - `xargs` è un comando utilizzato per trasformare l'output di un comando in argomenti per un altro comando.
   - `mv -v -t ~/` sposta (rinomina) i file specificati nella directory specificata (`~/` è la tua directory home) -tcon il flag `v` che indica a `mv` di essere verboso, cioè di mostrare quali file vengono spostati.
   - `-t` permette di specificare la directory di destinazione prima dei file da spostare.