

INDICE

INDICE

VERIFICA UTENTE E SESSIONE

CREAZIONE DI UN PROCESSO FOREGROUND

ESECUZIONE PROCESSO IN BACKGROUND /GESTIONE DEL JOB

VERIFICA DEI PROCESSI ATTIVI

TERMINE DI UN PROCESSO ATTIVO

VERIFICA DELLO SPAZIO OCCUPATO SUL DISCO

VERIFICA UTENTE E SESSIONE

```
(kali㉿kali)-[~]
$ w
10:03:33 up 1:58, 1 user, load average: 0.09, 0.27, 0.26
USER      TTY      FROM          LOGIN@   IDLE   JCPU   PCPU   WHAT
kali      -                08:05        0.00s   0.04s  lightdm --session-child 13 24

(kali㉿kali)-[~]
$ who
kali      seat0      2025-10-21 08:05 (:0)

(kali㉿kali)-[~]
$ whoami
kali
```

All'avvio dell'attività viene effettuata una ricognizione del contesto operativo per accertare identità dell'utente e stato della sessione.

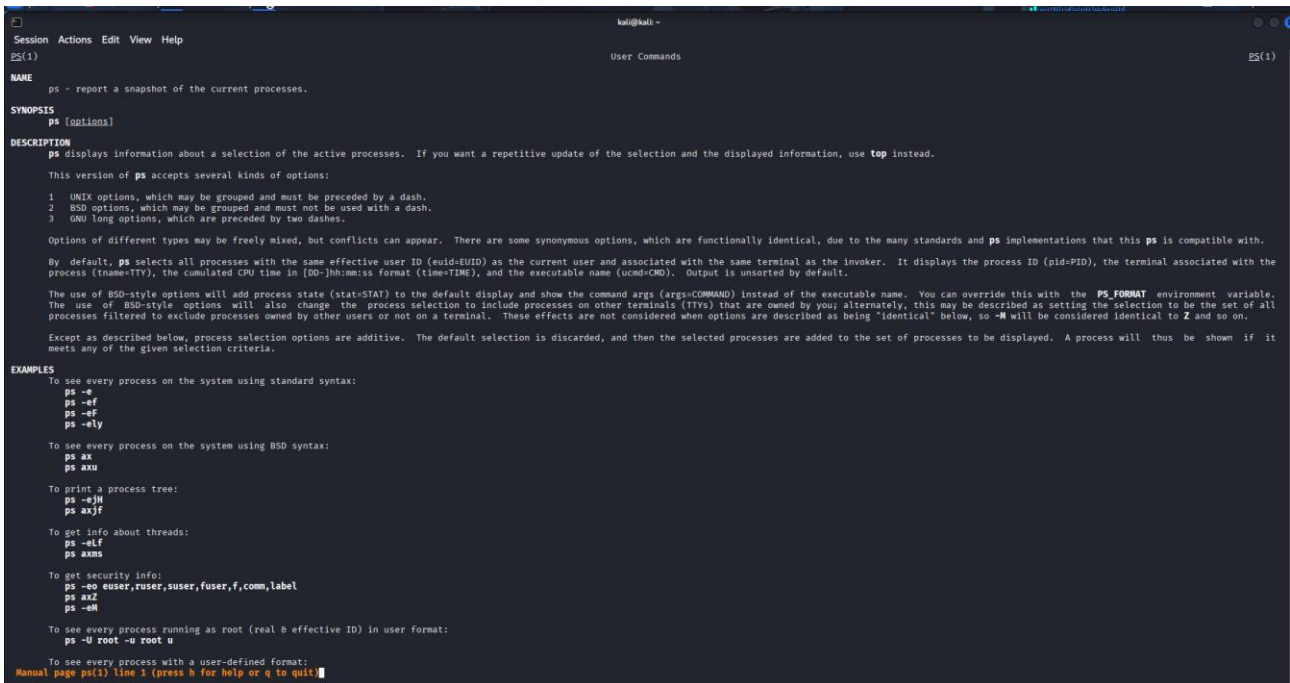
Il comando '**w**' fornisce una panoramica sintetica del sistema e l'elenco degli utenti collegati con le relative informazioni di sessione

. A seguire, **who** conferma le sessioni attive indicando utente, terminale/seat e display grafico, utile a distinguere sessioni locali da eventuali accessi remoti.

Infine, **whoami** restituisce l'utente effettivo con cui verranno eseguiti i comandi.

Questo triplice controllo assicura che le operazioni sui processi vengano condotte nel contesto previsto e con le corrette credenziali operative.

ANALISI DEL MANUALE DEL COMANDO PS



```
Session Actions Edit View Help
PS(1) User Commands PS(1)

NAME
  ps - report a snapshot of the current processes.

SYNOPSIS
  ps [options]

DESCRIPTION
  ps displays information about a selection of the active processes. If you want a repetitive update of the selection and the displayed information, use top instead.

  This version of ps accepts several kinds of options:

  1. UNIX options, which may be grouped and must be preceded by a dash.
  2. BSD options, which may be grouped and must not be used with a dash.
  3. GNU long options, which are preceded by two dashes.

  Options of different types may be freely mixed, but conflicts can appear. There are some synonymous options, which are functionally identical, due to the many standards and ps implementations that this ps is compatible with.

  By default, ps selects all processes with the same effective user ID (euid=EUID) as the current user and associated with the same terminal as the invoker. It displays the process ID (pid=PID), the terminal associated with the process (tname=TTY), the cumulated CPU time in [DD-]hh:mm:ss format (time=TIME), and the executable name (cmd=CMD). Output is unsorted by default.

  The use of BSD-style options will add process state (stat=STAT) to the default display and show the command args (args=COMMAND) instead of the executable name. You can override this with the PS_FORMAT environment variable. The use of BSD-style options will also change the process selection to include processes on other terminals (TTYS) that are owned by you; alternately, this may be described as setting the selection to be the set of all processes filtered to exclude processes owned by other users or not on a terminal. These effects are not considered when options are described as being "identical" below, so -M will be considered identical to Z and so on.

  Except as described below, process selection options are additive. The default selection is discarded, and then the selected processes are added to the set of processes to be displayed. A process will thus be shown if it meets any of the given selection criteria.

EXAMPLES
  To see every process on the system using standard syntax:
  ps -e
  ps -ef
  ps -xf
  ps -ely

  To see every process on the system using BSD syntax:
  ps ax
  ps aux

  To print a process tree:
  ps -fj
  ps axjf

  To get info about threads:
  ps -tlf
  ps axms

  To get security info:
  ps -eo euser,ruser,suser,fuser,f,comm,label
  ps axz
  ps -lw

  To see every process running as root (real & effective ID) in user format:
  ps -U root -u root u

  To see every process with a user-defined format:
  Manual page ps(1) line 1 (press h for help or q to quit)
```

In questa fase è stata avviata la consultazione del manuale interattivo di Linux mediante il comando **man ps**.

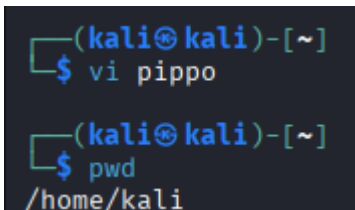
Lo scopo era analizzare la documentazione del comando **ps** per la visualizzazione e il monitoraggio dei processi attivi nel sistema.

Il manuale descrive in dettaglio le opzioni disponibili, suddividendole in categorie e spiega come queste possano essere combinate per personalizzare l'output.

Il comando, infatti, consente di elencare i processi in esecuzione, visualizzando informazioni come l'identificativo del processo (PID), il terminale associato, il tempo di CPU utilizzato e il nome del programma in esecuzione.

La sezione *EXAMPLES* fornisce inoltre esempi pratici (come **ps -ef** o **ps aux**) che mostrano differenti formati di visualizzazione.

CREAZIONE DI UN PROCESSO FOREGROUND



```
(kali㉿kali)-[~]  
$ vi pippo  
  
(kali㉿kali)-[~]  
$ pwd  
/home/kali
```

In questa fase viene avviato un nuovo processo in *foreground* attraverso il comando **vi pippo**.

L'editor di testo **vi** viene aperto per creare o modificare un file denominato *pippo*, ma poiché l'editor opera in modalità interattiva, il terminale rimane occupato finché il processo non viene chiuso.

Questo comportamento è tipico dei processi in foreground, che monopolizzano l'interfaccia utente fino al completamento o all'interruzione manuale.

Successivamente, tramite il comando **pwd**, viene verificata la posizione corrente nel file system, confermando che l'operazione si svolge all'interno della directory */home/kali*.

Questa verifica è utile per mantenere chiara la struttura dei percorsi e garantire che i file creati o modificati siano salvati nella posizione prevista.

ESECUZIONE PROCESSO IN BACKGROUND /GESTIONE DEL JOB

```
(kali㉿kali)-[~]  
$ firefox &  
[1] 66939  
  
(kali㉿kali)-[~]  
$ jobs  
[1] + running  firefox  
  
(kali㉿kali)-[~]  
$ fg %1  
[1] + running  firefox
```

In questo passaggio viene eseguito il browser Firefox come processo in background mediante l'aggiunta del simbolo & al comando di avvio (**firefox &**).

Questo consente al programma di essere eseguito in parallelo, liberando il terminale per l'immissione di altri comandi; Il terminale restituisce un identificativo numerico dove **[1]** rappresenta il numero del job e **66939** è il **PID** ossia il numero univoco assegnato al processo dal sistema operativo.

Successivamente viene utilizzato il comando `jobs` per visualizzare l'elenco dei processi attivi nella sessione corrente, mostrando che Firefox è in stato **running**.

Infine, tramite `fg %1`, il processo viene riportato in foreground, ovvero torna a occupare il terminale, permettendo l'interazione diretta con il programma o la sua terminazione manuale.

Questo passaggio dimostra la gestione dinamica dei processi tra modalità *background* e *foreground*.

VERIFICA DEI PROCESSI ATTIVI

```
kali@kali:~$ jobs
[1] + suspended firefox

kali@kali:~$ ps aux | grep firefox
kali 66939 4.2 16.4 2862088 331668 pts/1 TNL 18:18 0:00 /usr/lib/firefox-esr/firefox-esr -contentproc -parentBuildID 2025081113756 -prefIslen 27528 -prefMapSize 248953 -appDir /usr/lib/firefox-esr/browser {46f1bbb3-9648-4e12-a990-2a1e7233c35a} 66939 true socket
kali 67852 0.2 4.7 2417828 95852 pts/1 TNL 18:18 0:00 /usr/lib/firefox-esr/firefox-esr -contentproc -childID 1 -isForBrowser -prefIslen 27869 -prefMapSize 248953 -jsInitlen 234912 -parentBuildID 2025081113756 -greomni /usr/lib/firefox-esr/omni.ja -appomni /usr/lib/firefox-esr/browser/omni.ja -appDir /usr/lib/firefox-esr/browser {f0e2a33-8c06-470-923e-ec218c64f7e3} 66939 true tab
kali 67890 0.0 5.7 2436676 115620 pts/1 TNL 18:18 0:01 /usr/lib/firefox-esr/firefox-esr -contentproc -childID 2 -isForBrowser -prefIslen 33078 -prefMapSize 248953 -jsInitlen 234912 -parentBuildID 2025081113756 -greomni /usr/lib/firefox-esr/omni.ja -appomni /usr/lib/firefox-esr/browser/omni.ja -appDir /usr/lib/firefox-esr/browser {35d8dd39-d582-4bae-a8f8-7ad7ccdb2f6} 66939 true tab
kali 67234 0.4 4.7 2427448 95644 pts/1 TNL 18:18 0:01 /usr/lib/firefox-esr/firefox-esr -contentproc -childID 3 -isForBrowser -prefIslen 33078 -prefMapSize 248953 -jsInitlen 234912 -parentBuildID 2025081113756 -greomni /usr/lib/firefox-esr/omni.ja -appomni /usr/lib/firefox-esr/browser/omni.ja -appDir /usr/lib/firefox-esr/browser {7ace646-0b8e-4a7f-b0b9-d632be53f1eb} 66939 true tab
kali 67195 0.0 2.0 211388 40444 pts/1 TNL 18:18 0:00 /usr/lib/firefox-esr/firefox-esr -contentproc -parentBuildID 2025081113756 -sandboxingKind 0 -prefIslen 33078 -prefMapSize 248953 -appDir /usr/lib/firefox-esr/browser {b0c6a5-c9f2-4c4a-af3a-e4e0d057c818} 66939 true utility
kali 67202 0.1 3.6 2398588 73932 pts/1 TNL 18:18 0:00 /usr/lib/firefox-esr/firefox-esr -contentproc -childID 4 -isForBrowser -prefIslen 38501 -prefMapSize 248953 -jsInitlen 234912 -parentBuildID 2025081113756 -greomni /usr/lib/firefox-esr/omni.ja -appomni /usr/lib/firefox-esr/browser/omni.ja -appDir /usr/lib/firefox-esr/browser {074d8284-e1d9-435d-90af-ba71c91c287} 66939 true tab
kali 67200 0.1 3.6 2398588 74112 pts/1 TNL 18:18 0:00 /usr/lib/firefox-esr/firefox-esr -contentproc -childID 5 -isForBrowser -prefIslen 38501 -prefMapSize 248953 -jsInitlen 234912 -parentBuildID 2025081113756 -greomni /usr/lib/firefox-esr/omni.ja -appomni /usr/lib/firefox-esr/browser/omni.ja -appDir /usr/lib/firefox-esr/browser {8935a693-7fa5-400e-98eb-dd4ab23279ab} 66939 true tab
kali 67220 0.1 3.6 2398588 73724 pts/1 TNL 18:18 0:00 /usr/lib/firefox-esr/firefox-esr -contentproc -childID 6 -isForBrowser -prefIslen 38501 -prefMapSize 248953 -jsInitlen 234912 -parentBuildID 2025081113756 -greomni /usr/lib/firefox-esr/omni.ja -appomni /usr/lib/firefox-esr/browser/omni.ja -appDir /usr/lib/firefox-esr/browser {c3d01f9b-3841-430b-b125-c347af3978e7} 66939 true tab
kali 68946 0.0 0.1 6528 2252 pts/1 S+ 18:22 0:00 grep --color=auto firefox
```

In questa fase viene eseguito un controllo dei processi in esecuzione per monitorare lo stato di Firefox e comprendere la gestione delle attività in background.

Il comando `jobs` mostra che il processo Firefox risulta sospeso, indicando che è stato temporaneamente interrotto senza essere terminato.

Per ottenere una visione più dettagliata, viene utilizzato il comando `ps aux | grep firefox` in cui elenca tutti i processi in esecuzione nel sistema e filtra quelli relativi a Firefox.

L'output fornisce informazioni fondamentali:

- **PID (Process ID)**, ovvero l'identificativo numerico univoco di ciascun processo attivo.
- **%CPU e %MEM**, che mostrano rispettivamente l'utilizzo della CPU e della memoria.
- **TTY e STAT**, che indicano il terminale associato e lo stato del processo (ad esempio *R* per running o *T* per stopped).
- **COMMAND**, che riporta il percorso esatto dell'eseguibile in uso.

L'uso combinato di `ps` e `grep` consente di individuare rapidamente tutti i processi collegati a un'applicazione specifica, facilitando il controllo, la diagnostica e la successiva gestione (sospensione o terminazione) dei task nel sistema.

TERMINE DI UN PROCESSO ATTIVO

```
(kali㉿kali)-[~]  
$ kill -9 66939  
  
(kali㉿kali)-[~]  
$  
[1] + killed      firefox  
(kali㉿kali)-[~]  
$ ps aux | grep firefox  
  
kali      70045  0.0  0.1  6528  2348 pts/1    S+   10:24   0:00 grep --color=auto firefox
```

In questo passaggio viene eseguita la **terminazione forzata del processo Firefox** precedentemente individuato.

Il comando utilizzato è `kill -9 66939`, dove:

- **kill** serve per inviare un segnale a un processo;
- l'opzione **-9** corrisponde al segnale **SIGKILL**, che forza l'interruzione immediata del processo senza possibilità di salvataggio o gestione interna;
- **66939** rappresenta il **PID** (Process ID) del processo Firefox da chiudere.

Dopo l'esecuzione del comando, il terminale conferma l'avvenuta terminazione con il messaggio "killed firefox", segnalando che il processo è stato rimosso dalla memoria.

Per verificare l'effettiva chiusura, viene ripetuto il comando ***ps aux | grep firefox***: il risultato mostra soltanto il processo del comando grep stesso, a conferma che *Firefox non è più attivo nel sistema*.

Questo passaggio dimostra un'operazione fondamentale di amministrazione: la capacità di identificare e terminare manualmente processi bloccati o non responsivi, garantendo così il controllo e la stabilità del sistema.

VERIFICA DELLO SPAZIO OCCUPATO SUL DISCO

```
(kali㉿kali)-[~]  
$ df -h
```

Filesystem	Size	Used	Avail	Use%	Mounted on
udev	921M	0	921M	0%	/dev
tmpfs	198M	968K	197M	1%	/run
/dev/sda1	79G	16G	59G	22%	/
tmpfs	987M	4.0K	987M	1%	/dev/shm
tmpfs	5.0M	0	5.0M	0%	/run/lock
tmpfs	1.0M	0	1.0M	0%	/run/credentials/systemd-journald.service
tmpfs	987M	284K	987M	1%	/tmp
tmpfs	1.0M	0	1.0M	0%	/run/credentials/getty@tty1.service
tmpfs	198M	120K	198M	1%	/run/user/1000

L'ultimo passaggio dell'esercitazione riguarda l'analisi dello spazio disponibile e utilizzato sul disco mediante il comando `df -h`.

Il comando **df** (*disk free*) consente di visualizzare lo stato di utilizzo delle partizioni del filesystem, mentre l'opzione **-h** (*human readable*) converte i valori numerici in un formato leggibile.

CONCLUSIONE

L'esercizio facoltativo ha consentito di comprendere in modo pratico la gestione dei processi in ambiente Linux, dall'avvio di comandi in foreground e background fino al loro monitoraggio e terminazione.

L'utilizzo combinato dei comandi `jobs`, `ps`, `grep` e `kill` ha mostrato come identificare e controllare i processi attivi, mentre `df -h` ha permesso di verificare lo stato di utilizzo del disco.

Nel complesso, l'attività ha rafforzato la conoscenza dei meccanismi interni del sistema operativo, fondamentali per un controllo efficiente delle risorse e per la risoluzione di eventuali blocchi o sovraccarichi di sistema.