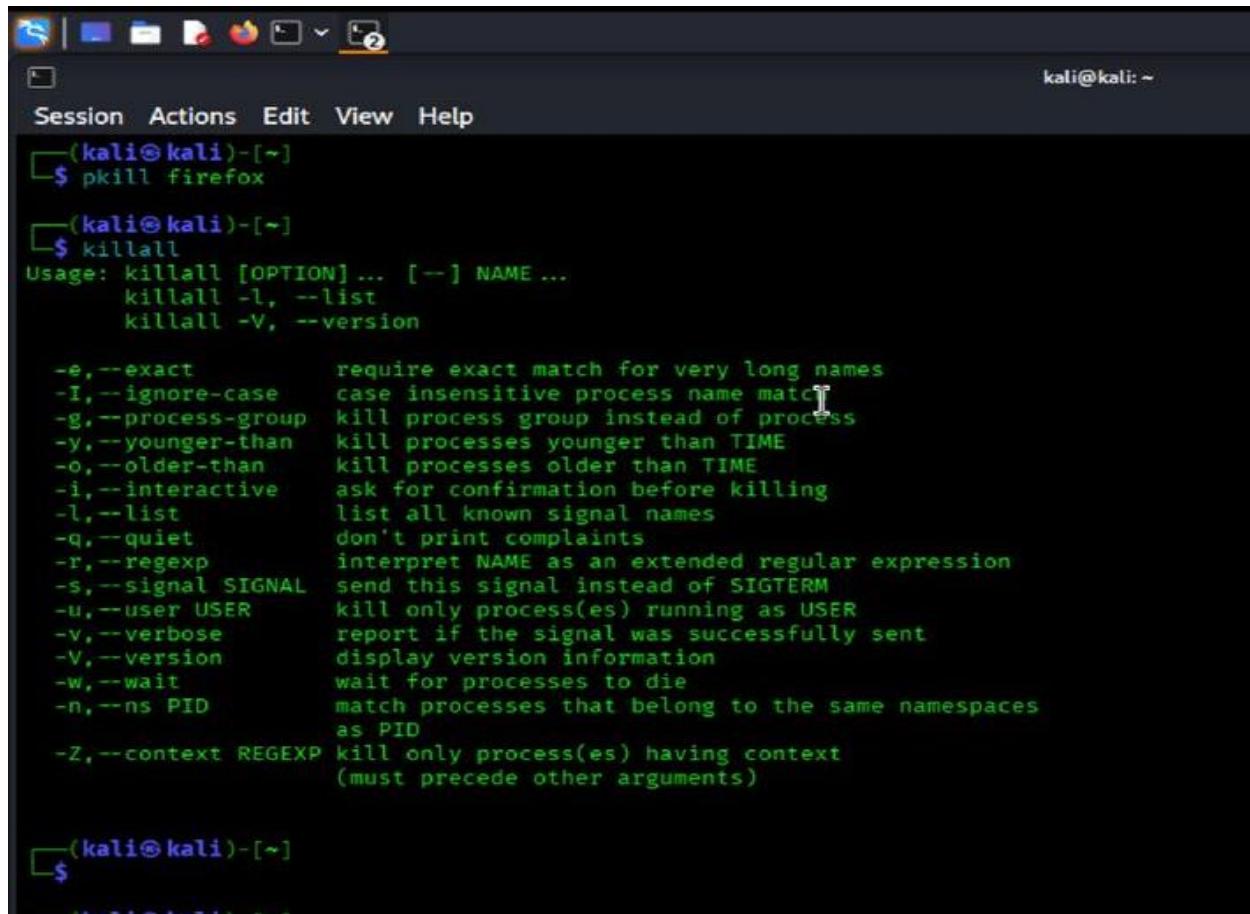


- Kill processes by name



A screenshot of a terminal window titled "Session Actions Edit View Help". The window shows a command-line interface with the following session history:

```

(kali㉿kali)-[~]
$ pkill firefox

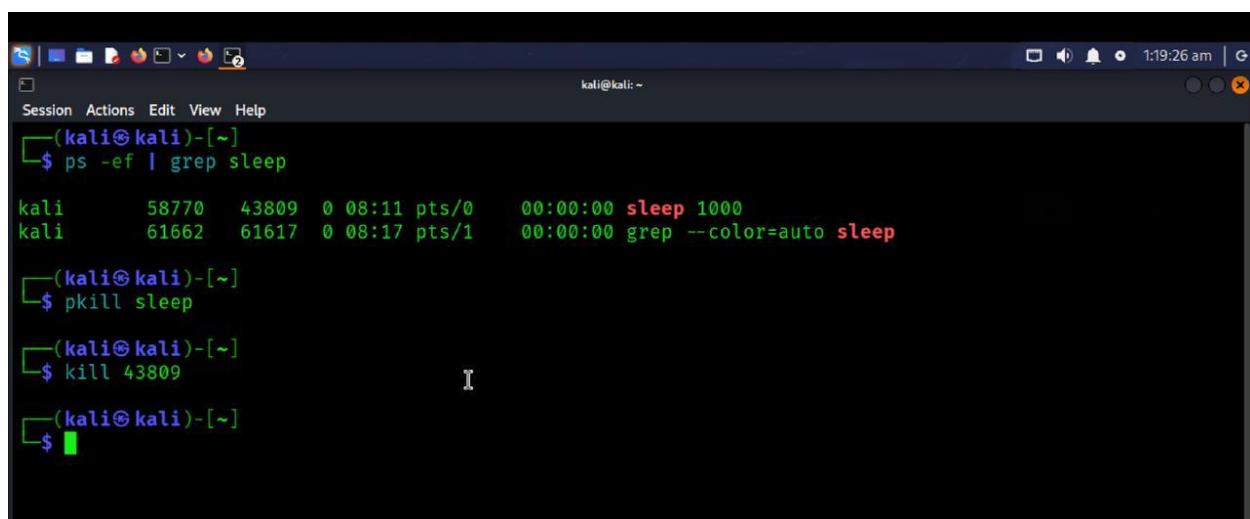
(kali㉿kali)-[~]
$ killall
usage: killall [OPTION]... [-] NAME...
    killall -l, --list
    killall -V, --version

-e,--exact      require exact match for very long names
-I,--ignore-case case insensitive process name match
-g,--process-group kill process group instead of process
-y,--younger-than kill processes younger than TIME
-o,--older-than  kill processes older than TIME
-i,--interactive ask for confirmation before killing
-l,--list        list all known signal names
-q,--quiet       don't print complaints
-r,--regexp      interpret NAME as an extended regular expression
-s,--signal SIGNAL send this signal instead of SIGTERM
-u,--user USER   kill only process(es) running as USER
-v,--verbose     report if the signal was successfully sent
-V,--version     display version information
-w,--wait        wait for processes to die
-n,--ns PID      match processes that belong to the same namespaces
                 as PID
-Z,--context REGEXP kill only process(es) having context
                 (must precede other arguments)

(kali㉿kali)-[~]
$ 

```

- Kill processes by ID



A screenshot of a terminal window titled "Session Actions Edit View Help". The window shows a command-line interface with the following session history:

```

(kali㉿kali)-[~]
$ ps -ef | grep sleep
kali      58770  43809  0 08:11 pts/0    00:00:00 sleep 1000
kali      61662  61617  0 08:17 pts/1    00:00:00 grep --color=auto sleep

(kali㉿kali)-[~]
$ pkill sleep

(kali㉿kali)-[~]
$ kill 43809
I

(kali㉿kali)-[~]
$ 

```

Orphan Process output

A terminal window titled "kali@kali: ~" showing the following command sequence:

```
(kali㉿kali)-[~]
$ nano program1.c
(kali㉿kali)-[~]
$ gcc program1.c
(kali㉿kali)-[~]
$ gcc program1.c -o program1
(kali㉿kali)-[~]
$ ./program1
(kali㉿kali)-[~]
$ Child Process
PID: 52255
(kali㉿kali)-[~]
$ ./program1
(kali㉿kali)-[~]
$ Child Process
PID: 52812
PPID: 1
```

The terminal shows two child processes running simultaneously, each with its own PID and PPID.

Zombie Process output

A terminal window titled "kali@kali: ~" showing the following command sequence:

```
(kali㉿kali)-[~]
$ nano program2.c
(kali㉿kali)-[~]
$ gcc program2.c
(kali㉿kali)-[~]
$ gcc program2.c -o program2
(kali㉿kali)-[~]
$ ./program2
(kali㉿kali)-[~]
$ Child Process
```

The terminal shows one child process running, which is identified as a "Zombie Process".

Fork output

The screenshot shows a terminal window titled 'kali' with a dark theme. The terminal is running on a Kali Linux system, as indicated by the prompt 'kali@kali: ~'. The user has created a file named 'frok.c' using nano, compiled it with gcc, and then run it. The output shows the parent process (PID: 56457) and the child process (PID: 56458) both printing their respective PID and PPID. The terminal window also displays some annotations from a presentation slide.

```
(kali㉿kali)-[~]
$ nano frok.c

(kali㉿kali)-[~]
$ gcc frok.c

(kali㉿kali)-[~]
$ ./frok
Parent Process
PID: 56457
Child Process
PID: 56458
PPID: 1

(kali㉿kali)-[~]
$
```

Annotations from the slide:

- 3. Process Creation using `fork()` System Call
- Child and Parent Process with PID and PPID