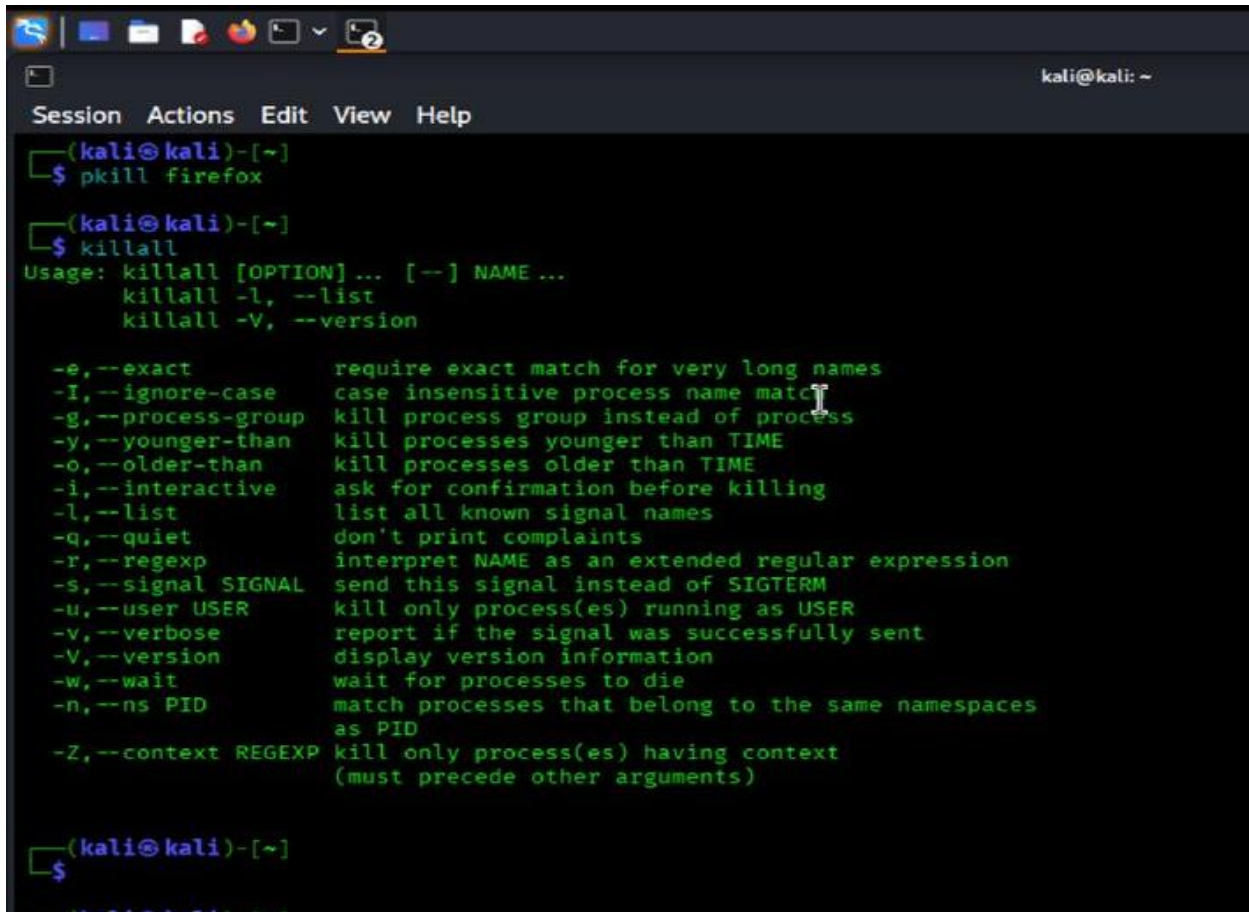


- Kill processes by name



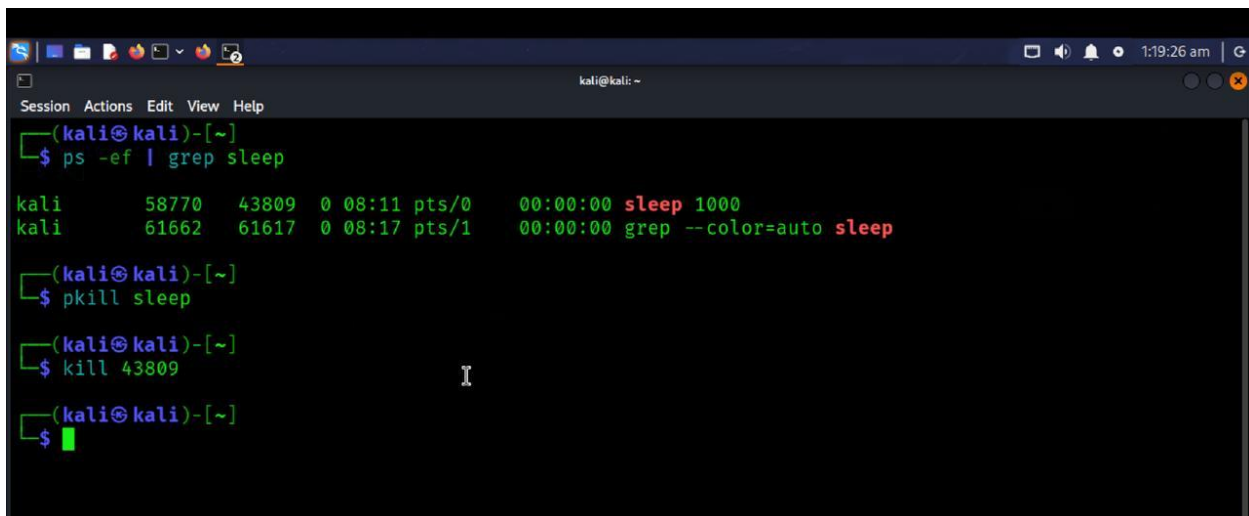
```
(kali@kali)-[~]
$ pkill firefox

(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



```
(kali@kali)-[~]
$ ps -ef | grep sleep

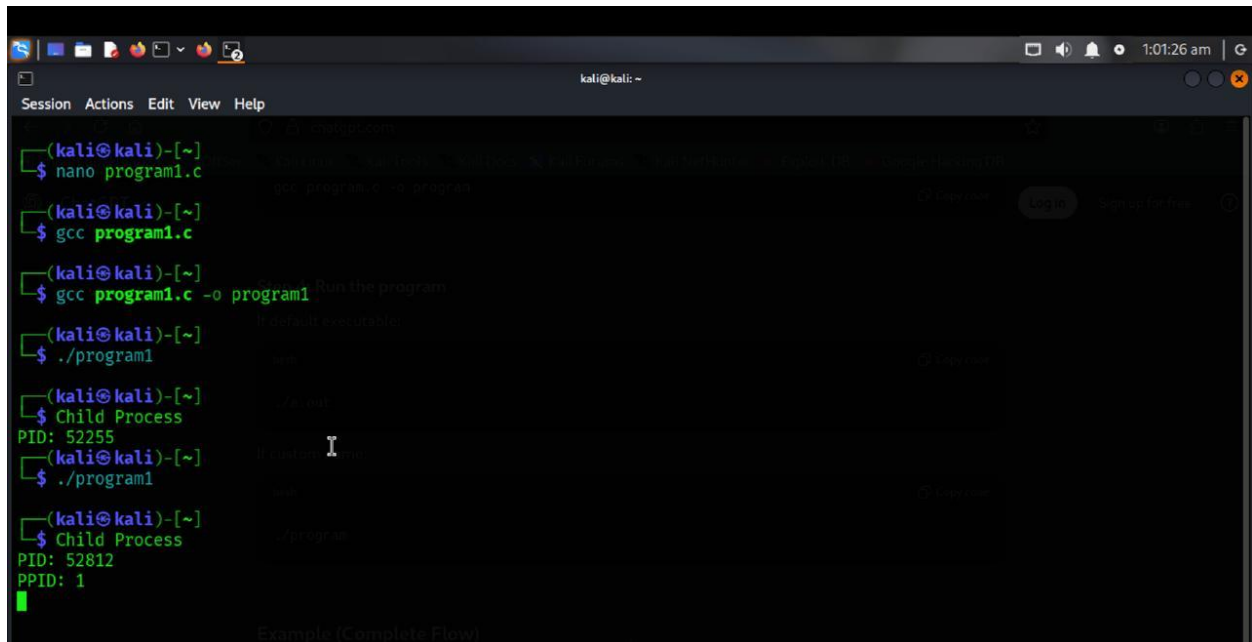
kali      58770   43809  0 08:11 pts/0    00:00:00 sleep 1000
kali      61662   61617  0 08:17 pts/1    00:00:00 grep --color=auto sleep

(kali@kali)-[~]
$ pkill sleep

(kali@kali)-[~]
$ kill 43809

(kali@kali)-[~]
$
```

## Orphan Process output

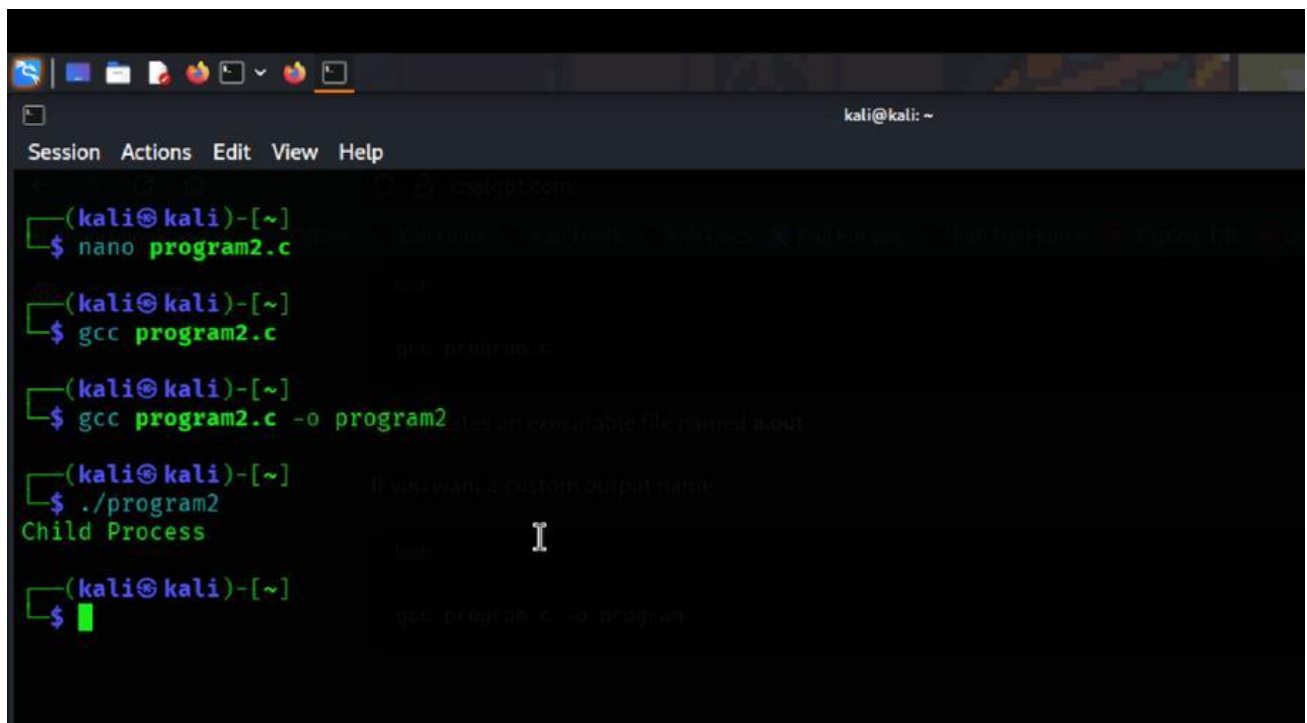


A terminal window on a Kali Linux system showing the following commands and output:

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

The output demonstrates the creation of a child process (PID: 52255) and its subsequent execution (PID: 52812) with PPID: 1, which is characteristic of an orphan process.

## Zombie Process output

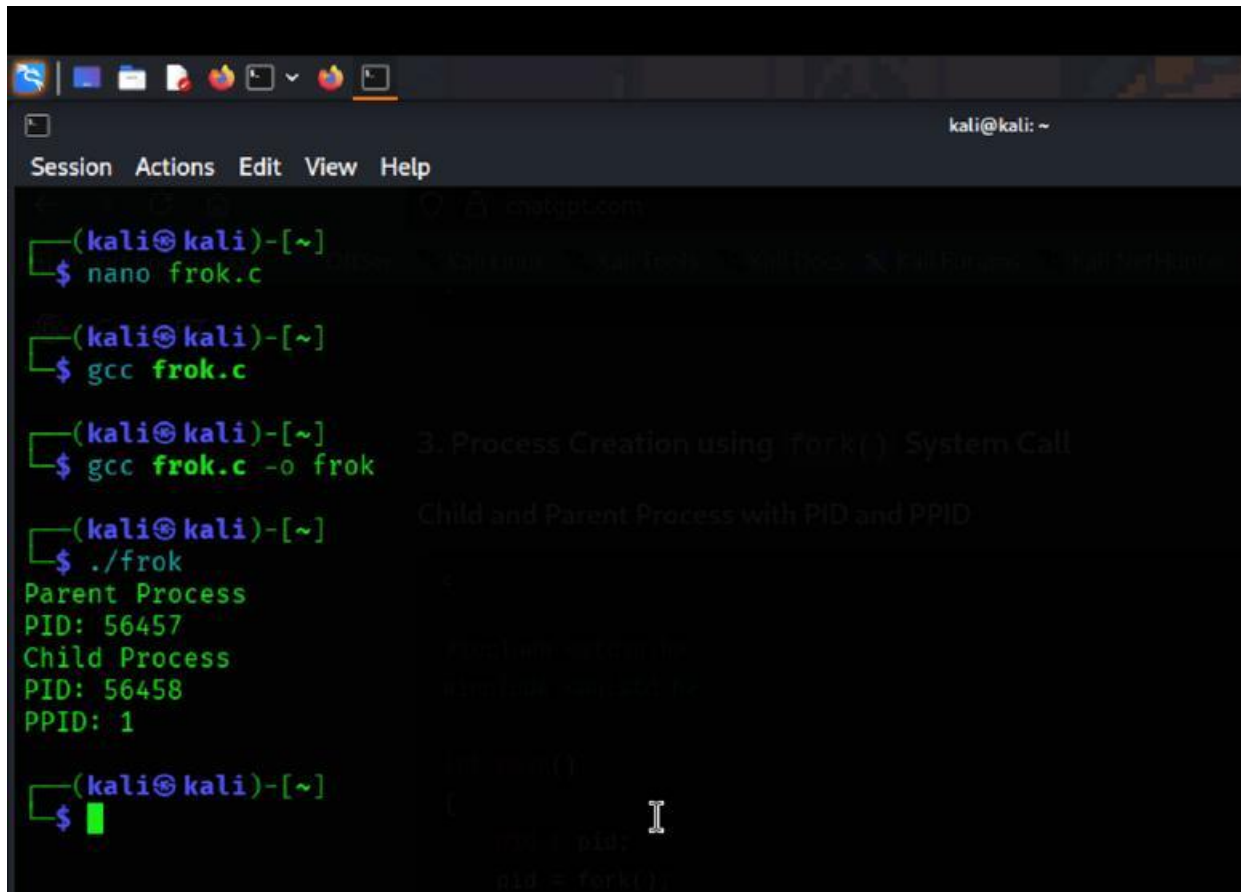


A terminal window on a Kali Linux system showing the following commands and output:

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

The output shows the creation of a child process (Child Process) which is then terminated, leaving a zombie process behind.

## Fork output



```
(kali㉿kali)-[~]
└─$ nano frok.c
└─$ gcc frok.c
└─$ gcc frok.c -o frok
└─$ ./frok
Parent Process
PID: 56457
Child Process
PID: 56458
PPID: 1
└─$
```

3. Process Creation using `fork()` System Call

Child and Parent Process with PID and PPID

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main()
{
    int pid;
    pid = fork();
```