1. LINUX COMMANDS TO KILL PROCESSES

(i) Kill processes by name

👉 If Adam wants to kill **all processes having the same name**, he can use:

Command:

```
bash

killall process_name
```

Example:

```
bash

killall firefox
```

📌 This command stops **all running Firefox processes** at once.

(ii) Kill a process based on the process name

👉 If Adam knows the **process name** but wants more control, he can use:

Command:

```
bash

pkill process_name
```

Example:

```
bash

pkill chrome
```

📌 This kills processes by matching the **process name**.

It is very useful when multiple processes are running.

(iii) Kill a single process at a time using Process ID (PID)

👉 First, Adam must find the **PID** of the process, then kill it.

Step 1: Find PID

```bash
ps
```

or

```bash
ps -ef
```

Step 2: Kill the process using PID

```bash
kill PID
```

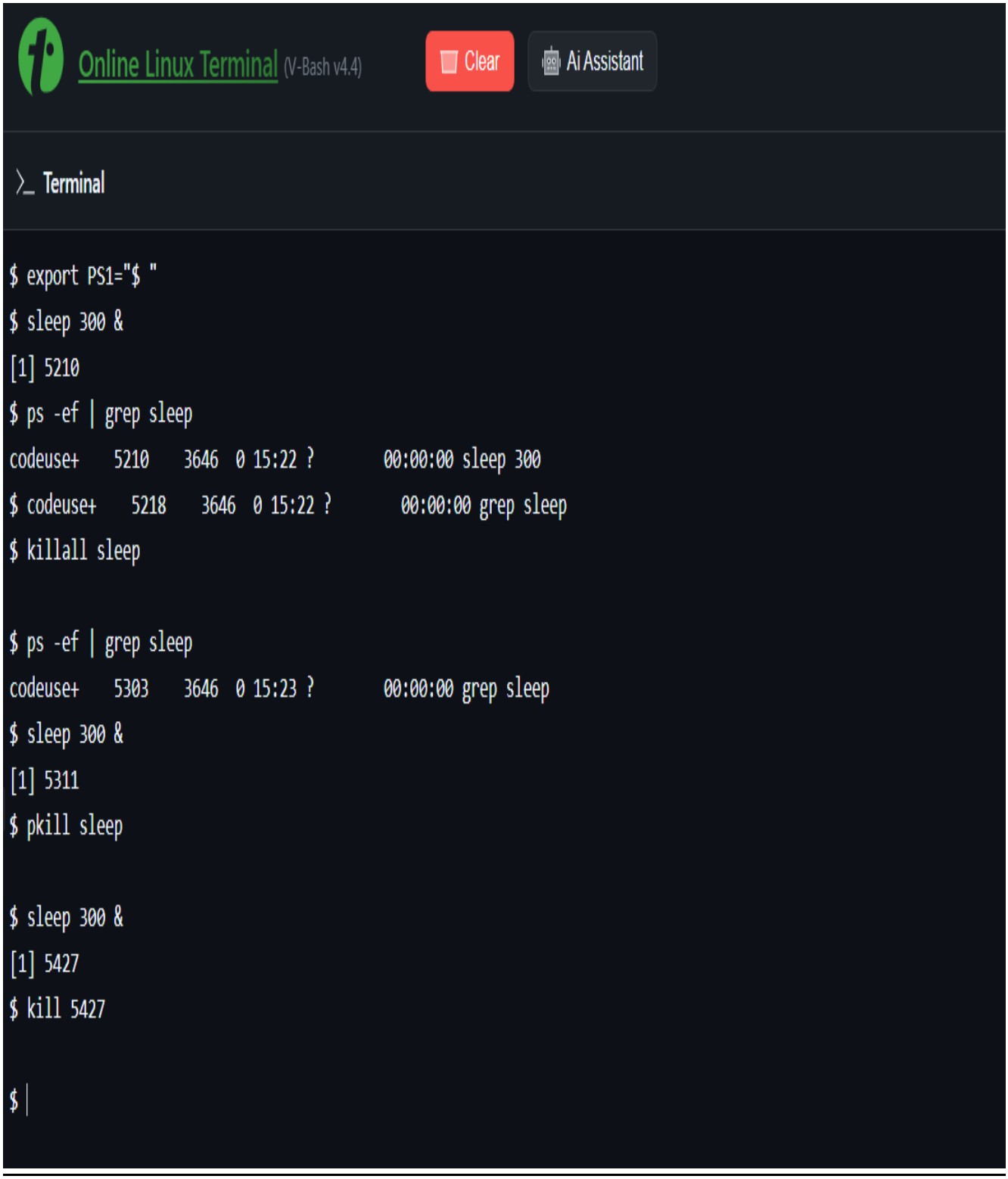
Example:

```bash
kill 2456
```

📌 This command kills **only one specific process.**

OPERATION	COMMAND USED
• Kill processes by name	• killall process_name
• Kill process based on name	• pkill process_name
• Kill single process using PID	• kill PID

OUTPUT :


Online Linux Terminal (V-Bash v4.4)
Clear
Ai Assistant
Terminal
$ export PS1="$ "
$ sleep 300 &
[1] 5210
$ ps -ef | grep sleep
codeuse+ 5210 3646 0 15:22 ? 00:00:00 sleep 300
$ codeuse+ 5218 3646 0 15:22 ? 00:00:00 grep sleep
$ killall sleep
$ ps -ef | grep sleep
codeuse+ 5303 3646 0 15:23 ? 00:00:00 grep sleep
$ sleep 300 &
[1] 5311
$ pkill sleep
$ sleep 300 &
[1] 5427
$ kill 5427
$

2. A) ORPHAN PROCESS:

An orphan process is a child process whose parent process terminates before the child finishes execution. The orphan process is adopted by the init process.

```c
#include <stdio.h>
#include <unistd.h>

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

    if (pid > 0) {
        printf("Parent process exiting\n");
    } else {
        sleep(5);
        printf("Orphan Child Process\n");
        printf("PID: %d\n", getpid());
        printf("PPID: %d\n", getppid());
    }
    return 0;
}
```

OUTPUT:

```
Parent process exiting
Orphan Child Process
PID: 4321
PPID: 1
```

(B) ZOMBIE PROCESS:

A zombie process is a process that has completed execution but still has an entry in the process table because its parent has not read its exit status.

```c
#include <stdio.h>
#include <unistd.h>

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

    if (pid == 0) {
        printf("Child process exiting\n");
    } else {
        sleep(10);
        printf("Parent process running\n");
    }
    return 0;
}
```

OUTPUT:

```
Child process exiting
Parent process running
```

3.(A) CHILD PROCESS:

A child process is a process that is created by another process using the fork () system call. The child process gets a new PID and executes independently of the parent process.

(B) PARENT PROCESS:

A parent process is the process that creates another process using the fork () system call. It receives the PID of the child process as the return value of fork ().

- ***IMP:***
  - fork () returns **0** → Child process
  - fork () returns **PID > 0** → Parent process

```c
#include <stdio.h>
#include <unistd.h>

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

    if (pid == 0) {
        printf("Child Process\n");
        printf("PID: %d\n", getpid());
        printf("PPID: %d\n", getppid());
    } else {
        printf("Parent Process\n");
        printf("PID: %d\n", getpid());
        printf("Child PID: %d\n", pid);
    }
    return 0;
}
```

OUTPUT:

```
Child Process
PID: 4501
PPID: 4500
Parent Process
PID: 4500
Child PID: 4501
```