Adam is working in an lT company. He has been given a task to reduce the load of a system by kitling some of the processes running in the LINUX operating system. Which commands will he use to complete the given task with the help of the following operation?

- **Kill Processes by name**

```
(base) sakshirakhade@Sakshis-MacBook-Air OS_Practical % sleep 500 &
sleep 500 &

[1] 50663
[2] 50664
(base) sakshirakhade@Sakshis-MacBook-Air OS_Practical % killall sleep

[2]  + terminated  sleep 500
[1]  + terminated  sleep 500
(base) sakshirakhade@Sakshis-MacBook-Air OS_Practical % killall -9 sleep

No matching processes belonging to you were found
(base) sakshirakhade@Sakshis-MacBook-Air OS_Practical % ls

fork_basic      fork_basic.c    orphan          orphan.c        zombie          zombie.c
(base) sakshirakhade@Sakshis-MacBook-Air OS_Practical %
```

- **Kill a process based on the process name**

```
(base) sakshirakhade@Sakshis-MacBook-Air OS_Practical % sleep 500 &
sleep 500 &
sleep 500 &

[1] 50655
[2] 50656
[3] 50657
(base) sakshirakhade@Sakshis-MacBook-Air OS_Practical % pkill sleep

[3]  + terminated  sleep 500
[2]  + terminated  sleep 500
[1]  + terminated  sleep 500
(base) sakshirakhade@Sakshis-MacBook-Air OS_Practical % pgrep sleep
```

- **Kill a single process at a time with the given process lD**

```
(base) sakshirakhade@Sakshis-MacBook-Air OS_Practical % sleep 500 &

[1] 50630
(base) sakshirakhade@Sakshis-MacBook-Air OS_Practical % kill 50630
[1]  + terminated  sleep 500
(base) sakshirakhade@Sakshis-MacBook-Air OS_Practical % ps -ef | grep sleep

  501 50645 50529   0  4:52PM ttys000    0:00.01 grep sleep
(base) sakshirakhade@Sakshis-MacBook-Air OS_Practical %
```

2.Write a program for process creation using C

- **Orphan Process**

```
(base) sakshirakhade@Sakshis-MacBook-Air OS_Practical % nano orphan.c

(base) sakshirakhade@Sakshis-MacBook-Air OS_Practical % gcc orphan.c -o orphan

(base) sakshirakhade@Sakshis-MacBook-Air OS_Practical % ./orphan


[PARENT] Exiting immediately...
Parent PID = 51509

[CHILD] Before sleep
Child PID  = 51510
Child PPID = 51509
(base) sakshirakhade@Sakshis-MacBook-Air OS_Practical %
[CHILD] After sleep (Parent exited)
Child PID  = 51510
New PPID   = 1

(base) sakshirakhade@Sakshis-MacBook-Air OS_Practical %
```

```
  UW PICO 5.09                                                    File: orphan.c

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

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

    if(pid < 0) {
        printf("Fork failed!\n");
        return 1;
    }

    if(pid == 0) {
        printf("\n[CHILD] Before sleep\n");
        printf("Child PID  = %d\n", getpid());
        printf("Child PPID = %d\n", getppid());

        sleep(5);

        printf("\n[CHILD] After sleep (Parent exited)\n");
        printf("Child PID  = %d\n", getpid());
        printf("New PPID   = %d\n", getppid());
    }
    else {
        printf("\n[PARENT] Exiting immediately...\n");
        printf("Parent PID = %d\n", getpid());
    }

    return 0;
}
```

- **Zombie Process**

```
(base) sakshirakhade@Sakshis-MacBook-Air OS_Practical % nano zombie.c

(base) sakshirakhade@Sakshis-MacBook-Air OS_Practical % gcc zombie.c -o zombie

(base) sakshirakhade@Sakshis-MacBook-Air OS_Practical % ./zombie &

[1] 51528
(base) sakshirakhade@Sakshis-MacBook-Air OS_Practical %
[PARENT] Sleeping 20 sec (Child will become Zombie)
Parent PID = 51528

[CHILD] Exiting now...
Child PID = 51529

(base) sakshirakhade@Sakshis-MacBook-Air OS_Practical %
```

```
  UW PICO 5.09                                                    File: zombie.c

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

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

    if(pid < 0) {
        printf("Fork failed!\n");
        return 1;
    }

    if(pid == 0) {
        printf("\n[CHILD] Exiting now...\n");
        printf("Child PID = %d\n", getpid());
        return 0;
    }
    else {
        printf("\n[PARENT] Sleeping 20 sec (Child will become Zombie)\n");
        printf("Parent PID = %d\n", getpid());
        sleep(20);
        printf("\n[PARENT] Done sleeping.\n");
    }

    return 0;
}
```

3. Create the pro.cess using fork () system call.
   - **Child Process creation**
   - **Parent Process creation** using PPID and PID

```
Last login: Mon Feb  2 16:41:33 on ttys000
(base) sakshirakhade@Sakshis-MacBook-Air ~ % cd OS_Practical
(base) sakshirakhade@Sakshis-MacBook-Air OS_Practical % nano fork_basic.c

(base) sakshirakhade@Sakshis-MacBook-Air OS_Practical % gcc fork_basic.c -o fork_basic

(base) sakshirakhade@Sakshis-MacBook-Air OS_Practical % ./fork_basic


[PARENT PROCESS]
Parent PID = 51486
Child PID  = 51487

[CHILD PROCESS]
Child PID  = 51487
Child PPID = 51486
(base) sakshirakhade@Sakshis-MacBook-Air OS_Practical %
```

```
  UW PICO 5.09                                                    File: fork_basic.c

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

int main() {
    pid_t pid;

    pid = fork();

    if(pid < 0) {
        printf("Fork failed!\n");
        return 1;
    }
    else if(pid == 0) {
        printf("\n[CHILD PROCESS]\n");
        printf("Child PID  = %d\n", getpid());
        printf("Child PPID = %d\n", getppid());
    }
    else {
        printf("\n[PARENT PROCESS]\n");
        printf("Parent PID = %d\n", getpid());
        printf("Child PID  = %d\n", pid);
    }

    return 0;
}
```