

## Practical 4 :

Adam is working in an IT company. He has been given a task to reduce the load of a system by killing 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
- Kill a process based on the process name
- Kill a single process at a time with the given process ID

```
De11@DESKTOP-36N7A9E UCRT64 ~
$ sleep 800 & sleep 500 &
[5] 168
[6] 169

De11@DESKTOP-36N7A9E UCRT64 ~
$ ps -ef | grep sleep
De11      149      119 pty0      18:27:26 sleep 900
De11      166      119 pty0      18:35:58 sleep 400
De11      137      119 pty0      18:18:28 sleep 600
De11      169      119 pty0      18:37:22 sleep 500
De11      168      119 pty0      18:37:22 sleep 800

De11@DESKTOP-36N7A9E UCRT64 ~
$ kill 168 169

De11@DESKTOP-36N7A9E UCRT64 ~
$ ps -ef | grep sleep
De11      149      119 pty0      18:27:26 sleep 900
De11      166      119 pty0      18:35:58 sleep 400
De11      137      119 pty0      18:18:28 sleep 600
[5] Terminated sleep 800
[6]- Terminated sleep 500

De11@DESKTOP-36N7A9E UCRT64 ~
$ |
```

```
e11@DESKTOP-36N7A9E UCRT64 ~
$ sleep 900 &
[4] 160

e11@DESKTOP-36N7A9E UCRT64 ~
$ ps -ef | grep sleep
De11      149      119 pty0      18:27:26 sleep 900
De11      160      119 pty0      18:33:23 sleep 900
De11      137      119 pty0      18:18:28 sleep 600

e11@DESKTOP-36N7A9E UCRT64 ~
$ kill 160

e11@DESKTOP-36N7A9E UCRT64 ~
$ ps -ef | grep sleep
De11      149      119 pty0      18:27:26 sleep 900
De11      137      119 pty0      18:18:28 sleep 600
[4]- Terminated sleep 900

e11@DESKTOP-36N7A9E UCRT64 ~
$ |
```

```

De11@DESKTOP-36N7A9E UCRT64 ~
$ sleep 600 &
[3] 143

De11@DESKTOP-36N7A9E UCRT64 ~
$ kill 143

De11@DESKTOP-36N7A9E UCRT64 ~
$ ps
  PID     PPID     PGID     WINPID    TTY          UID         ST
    144       119       144      14516   pty0        197609    18:23
    139       119       139      1212   pty0        197609    18:20
    119       118       119      16360   pty0        197609    18:18
s   137       119       137      16828   pty0        197609    18:18
    118         1       118      16848    ?          197609    18:18
[3]-  Terminated                  sleep 600

De11@DESKTOP-36N7A9E UCRT64 ~
$ |

```

## 2. Write a program for process creation using C

- Orphan Process

```

GNU nano 8.7 orphan.c
#include <stdio.h>
#include <unistd.h>

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

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

[ Read 17 line
^G Help      ^O Write Out  ^F Where Is   ^K Cut
^X Exit      ^R Read File  ^\ Replace    ^U Paste

```

```
De1l@DESKTOP-36N7A9E MSYS ~  
$ nano orphan.c  
  
De1l@DESKTOP-36N7A9E MSYS ~  
$ gcc orphan.c -o orphan  
  
De1l@DESKTOP-36N7A9E MSYS ~  
$ ./orphan  
Parent exiting  
  
De1l@DESKTOP-36N7A9E MSYS ~  
$ Child Process  
PID : 699  
PPID : 1
```

- Zombie Process

```
GNU nano 8.7  
#include <stdio.h>  
#include <unistd.h>  
  
int main() {  
    pid_t pid = fork();  
  
    if (pid == 0) {  
        printf("Child exiting\n");  
    }  
    else {  
        sleep(10);  
        printf("Parent running\n");  
    }  
    return 0;  
}
```

```

De11@DESKTOP-36N7A9E MSYS ~
$ nano zombie.c

De11@DESKTOP-36N7A9E MSYS ~
$ gcc zombie.c -o zombie

De11@DESKTOP-36N7A9E MSYS ~
$ ./zombie
Child exiting
Parent running

De11@DESKTOP-36N7A9E MSYS ~
$ ps -el

```

PID	PPID	PGID	WINPID	TTY	UID	STIME	COMMAND
544	1	544	4272	?	197609	14:41:22	/usr/bin/mintty
709	546	709	8848	pty1	197609	14:53:45	/usr/bin/ps
546	544	546	12784	pty1	197609	14:41:22	/usr/bin/bash

```

De11@DESKTOP-36N7A9E MSYS ~
$

```

3. Create the process using fork () system call.

- Child Process creation
- Parent process creation
- PPID and PID

```

GNU nano 8.7
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main() {
    pid_t pid;

    pid = fork();    // create process

    if (pid < 0) {
        // fork failed
        printf("Fork failed\n");
    }
    else if (pid == 0) {
        // child process
        printf("Child Process\n");
        printf("Child PID   : %d\n", getpid());
        printf("Parent PID  : %d\n", getppid());
    }
    else {
        // Parent process
        printf("Parent Process\n");
        printf("Parent PID  : %d\n", getpid());
        printf("Child PID   : %d\n", pid);
    }

    return 0;
}

```

```
De11@aryaaa MSYS ~  
$ nano fork_process.c  
  
De11@aryaaa MSYS ~  
$ gcc fork_process.c -o fork_process  
  
De11@aryaaa MSYS ~  
$ ./fork_process  
  
Parent Process  
Parent PID : 772  
Child PID : 773  
Child Process  
Child PID : 773  
Parent PID : 772  
  
De11@aryaaa MSYS ~  
$ |
```