

1. 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

```
F0IN@LAPTOP-MC73RDEH MSYS ~
$ gcc --version
gcc (GCC) 11.2.0
Copyright (C) 2021 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

F0IN@LAPTOP-MC73RDEH MSYS ~
$ sleep 1000 &
[1] 1238

F0IN@LAPTOP-MC73RDEH MSYS ~
$ ps
  PID  PPID  PGID  WINPID  TTY      UID  STIME COMMAND
1238  1215  1238     4936 pty0    197610 20:20:37 /usr/bin/sleep
1215  1214  1215    10920 pty0    197610 20:18:13 /usr/bin/bash
1239  1215  1239    21388 pty0    197610 20:20:42 /usr/bin/ps
1214      1  1214    12528 ?      197610 20:18:13 /usr/bin/mintty

F0IN@LAPTOP-MC73RDEH MSYS ~
$ pkill sleep
-bash: pkill: command not found

F0IN@LAPTOP-MC73RDEH MSYS ~
$ ps
  PID  PPID  PGID  WINPID  TTY      UID  STIME COMMAND
1238  1215  1238     4936 pty0    197610 20:20:37 /usr/bin/sleep
1215  1214  1215    10920 pty0    197610 20:18:13 /usr/bin/bash
1214      1  1214    12528 ?      197610 20:18:13 /usr/bin/mintty

F0IN@LAPTOP-MC73RDEH MSYS ~
$ kill 1238

F0IN@LAPTOP-MC73RDEH MSYS ~
$ ps | grep sleep
[1]+  Terminated                 sleep 1000

F0IN@LAPTOP-MC73RDEH MSYS ~
$ 

F0IN@LAPTOP-MC73RDEH MSYS ~
$ sleep 1000 &
[1] 1245

F0IN@LAPTOP-MC73RDEH MSYS ~
$ ps -e | grep sleep
  1245   1215   1245     15576  pty0    197610 20:24:40 /usr/bin/sleep

F0IN@LAPTOP-MC73RDEH MSYS ~
$ kill 1245

F0IN@LAPTOP-MC73RDEH MSYS ~
$ ps -e | grep sleep
[1]+  Terminated                 sleep 1000

F0IN@LAPTOP-MC73RDEH MSYS ~
$ 
```

```
FOIN@LAPTOP-MC73RDEH MSYS ~
$ sleep 1000 &
[1] 1251

FOIN@LAPTOP-MC73RDEH MSYS ~
$ ps -e
  PID  PPID  PGID   WINPID   TTY      UID    STIME COMMAND
1251  1215  1251    19980  pty0    197610 20:26:57 /usr/bin/sleep
1252  1215  1252    14556  pty0    197610 20:27:03 /usr/bin/ps
1215  1214  1215    10920  pty0    197610 20:18:13 /usr/bin/bash
1214          1  1214    12528  ?      197610 20:18:13 /usr/bin/mintty

FOIN@LAPTOP-MC73RDEH MSYS ~
$ kill 1251

FOIN@LAPTOP-MC73RDEH MSYS ~
$ ps -e | grep sleep
[1]+  Terminated                  sleep 1000

FOIN@LAPTOP-MC73RDEH MSYS ~
$ |
```

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() {
    int pid = fork();

    if (pid > 0) {
        printf("Parent process exiting\n");
    } else {
        sleep(5);
        printf("Child process running (Orphan)\n");
    }
    return 0;
}
```

```
M ~
FOIN@LAPTOP-MC73RDEH MSYS ~
$ nano orphan.c

FOIN@LAPTOP-MC73RDEH MSYS ~
$ gcc orphan.c -o orphan

FOIN@LAPTOP-MC73RDEH MSYS ~
$ ./orphan
Parent process exiting

FOIN@LAPTOP-MC73RDEH MSYS ~
$ Child process running (Orphan)
$ |
```



- Zombie Process

```
M ~
GNU nano 8.7                                     zombie.c                                Modified
#include <stdio.h>
#include <unistd.h>

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

    if (pid == 0) {
        // Child process
        printf("Child process finished\n");
    } else {
        // Parent process
        sleep(10); // parent sleeps, child becomes zombie
        printf("Parent process running\n");
    }
    return 0;
}|
```



```
M ~
FOIN@LAPTOP-MC73RDEH MSYS ~
$ nano zombie.c

FOIN@LAPTOP-MC73RDEH MSYS ~
$ gcc zombie.c -o zombie

FOIN@LAPTOP-MC73RDEH MSYS ~
$ ./zombie
Child process finished
Parent process running

FOIN@LAPTOP-MC73RDEH MSYS ~
$
```



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

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

```
M ~
GNU nano 8.7                               fork.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("PPID = %d\n", getppid());
    }
    return 0;
}
```



```
M ~
FOIN@LAPTOP-MC73RDEH MSYS ~
$ nano fork.c

FOIN@LAPTOP-MC73RDEH MSYS ~
$ gcc fork.c -o fork

FOIN@LAPTOP-MC73RDEH MSYS ~
$ ./fork
Parent Process
PID = 1272
PPID = 1215
Child Process
PID = 1273
PPID = 1272

FOIN@LAPTOP-MC73RDEH MSYS ~
$ |
```

