

## PRACTICAL 4

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?

Theory:

- Terminating Processes via Terminal  
To reduce system load, Adam can use these specific commands:
  - \* kill a single process by PID
    - Command : kill <PID>
    - Example : kill 1234
  - \* kill a process by name  
If Adam knows the name but not the ID, he can use pkill.
    - Command : pkill <process-name>
    - Example : pkill chrome.

UID	PID	PPID	C	S TIME	TTY	TIME	CMD
root	1	0	0	15:52 ?		00:00:06	/sbin/init splash
root	2	0	0	15:52 ?		00:00:00	[kthreadd]
root	3	2	0	15:52 ?		00:00:00	[pool_workqueue_release]
root	4	2	0	15:52 ?		00:00:00	[kworker/R-rcu_g]
root	5	2	0	15:52 ?		00:00:00	[kworker/R-rcu_p]
root	6	2	0	15:52 ?		00:00:00	[kworker/R-slub_]
root	7	2	0	15:52 ?		00:00:00	[kworker/R-netns]
root	9	2	0	15:52 ?		00:00:00	[kworker/0:0H-events_highpri]
root	10	2	0	15:52 ?		00:00:00	[kworker/0:1-events]
root	12	2	0	15:52 ?		00:00:00	[kworker/R-mm_pe]
root	13	2	0	15:52 ?		00:00:00	[rcu_tasks_kthread]
root	14	2	0	15:52 ?		00:00:00	[rcu_tasks_rude_kthread]
root	15	2	0	15:52 ?		00:00:00	[rcu_tasks_trace_kthread]
root	16	2	0	15:52 ?		00:00:00	[ksoftirqd/0]
root	17	2	0	15:52 ?		00:00:00	[rcu_preempt]
root	18	2	0	15:52 ?		00:00:00	[migration/0]
root	19	2	0	15:52 ?		00:00:00	[idle_inject/0]
root	20	2	0	15:52 ?		00:00:00	[cpuhp/0]
root	21	2	0	15:52 ?		00:00:00	[cpuhp/1]
root	22	2	0	15:52 ?		00:00:00	[idle_inject/1]
root	23	2	0	15:52 ?		00:00:00	[migration/1]
root	24	2	0	15:52 ?		00:00:00	[ksoftirqd/1]
root	26	2	0	15:52 ?		00:00:00	[kworker/1:0H-events_highpri]
root	27	2	0	15:52 ?		00:00:00	[cpuhp/2]
root	28	2	0	15:52 ?		00:00:00	[idle_inject/2]
root	29	2	0	15:52 ?		00:00:00	[migration/2]
	30	2	0	15:52 ?		00:00:00	[kworker/1:0]

- Kill processes by name

```
m309@m309-BY-OEM:~$ kill 9367
bash: kill: (9367) - No such process
m309@m309-BY-OEM:~$ pkill firefox
m309@m309-BY-OEM:~$ nano fork_demo.c
m309@m309-BY-OEM:~$ gcc fork_demo.c -o fork_demo
m309@m309-BY-OEM:~$ ./fork_demo
Parent Process
PID: 9442
Child PID: 9443
Child Process
PID: 9443
PPID: 9442
m309@m309-BY-OEM:~$ █
```

## Kill a process based on the process name

```
m309      9354  8892  1 17:01 :  88188188 /usr/libexec/gnome-terminal-server
m309      9361  9354  0 17:01 pts/0   00:00:00 bash
m309      9374  8413  0 17:02 ?       00:00:00 /snap/chromium/3293/usr/lib/chromium-browser/chrome --typ
root      9388      2  0 17:02 ?       00:00:00 [kworker/3:2]
root      9389      2  0 17:02 ?       00:00:00 [kworker/1:0-cgroup_destroy]
m309      9391  9361  0 17:02 pts/0   00:00:00 ps -ef
m309@m309-BY-OEM:~$ ps -ef |grep firefox
m309      9393  9361  0 17:02 pts/0   00:00:00 grep --color=auto firefox
m309@m309-BY-OEM:~$ ps -ef | grep firefox
m309      9407  9361  0 17:03 pts/0   00:00:00 grep --color=auto firefox
m309@m309-BY-OEM:~$ kill 9407
bash: kill: (9407) - No such process
m309@m309-BY-OEM:~$ █
```

- Kill a single process at a time with the given process ID

```
m309@m309-BY-OEM:~$ kill 9407
bash: kill: (9407) - No such process
m309@m309-BY-OEM:~$ pkill firefox
m309@m309-BY-OEM:~$ █
```

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

### Orphan Process

	<p><u>Orphan Process</u></p> <p>Creates when the parent process terminates before the child process. the child process continue execution and it adopted by the system.</p>
	<ul style="list-style-type: none"><li>• nano orphan.c → To create and write a c program for orphan process.</li><li>• gcc orphan.c -o orphan → To compile the c program</li><li>• ./orphan → To execute the compiled program.</li></ul>

### Code-

```
GNU nano 7.2                                     orphan.c

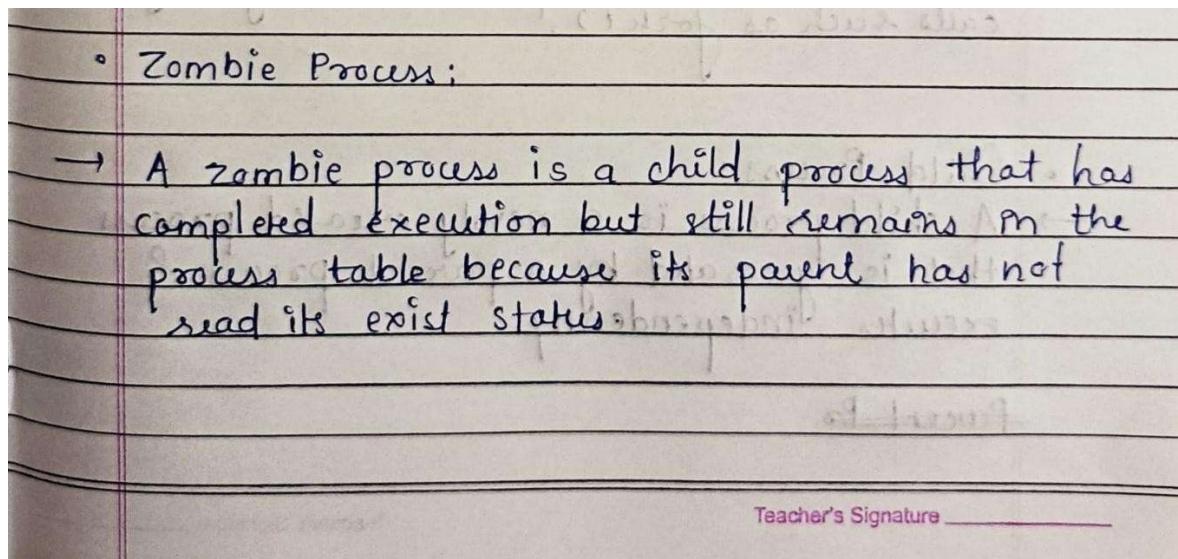
#include <stdio.h>
#include <unistd.h>

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

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

```
m309@m309-BY-OEM:~$ nano orphan.c
m309@m309-BY-OEM:~$ gcc orphan.c -o orphan
m309@m309-BY-OEM:~$ ./orphan
Parent exiting
m309@m309-BY-OEM:~$ Child PID: 9506
New Parent PID: 6892
```

- Zombie Process



```
GNU nano 7.2                                     zombie .c
#include <stdio.h>
#include <unistd.h>

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

    if (pid == 0) {
        // Child process
        printf("Child exiting\n");
        return 0;
    } else {
        // Parent process
        printf("Parent sleeping, child becomes zombie\n");
        sleep(30); // VERY IMPORTANT
    }
    return 0;
}
```

```
nano zombie.c
n309@m309-BY-OEM:~$ gcc zombie.c -o zombie
n309@m309-BY-OEM:~$ ./zombie
Child process exiting
```

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

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

* <u>PID (Process ID) :-</u>	A unique identifier for a specific process.
* <u>PPID (Parent Process ID) :-</u>	The ID of the process that created it .
<u>Commands :</u>	
<u>getpid ()</u> : Returns the PID of the current process	
<u>getppid ()</u> :- Returns the PID of the parent process	
<u>Command</u> →	<u>Purpose</u> .
• <u>ps -ef</u> → lists all running processes with their PID and PPID.	
• <u>pstree</u> → show processes in a visual tree format	
• <u>top</u> → provides a real time view of <sup>running</sup> processes	
• <u>kill -9 &lt; PID &gt;</u> → forcefully terminates a process using its PID.	

```
m309@m309-BY-OEM:~$ nano fork_demo.c
```

```
GNU nano 7.2                                     fork_demo.c

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

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

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

## 4.Infinite Loop

```
m309@m309-BY-OEM:~$ nano loop.c
m309@m309-BY-OEM:~$ gcc loop.c -o loop
m309@m309-BY-OEM:~$ ./loop
Running...
Running...
Running...
Running...
Running...
Running...
Running...
```

```
GNU nano 7.2                                     loop.c *

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

int main() {
    while (1) {
        printf("Running...\n");
        sleep(1);
    }
    return 0;
}
```