

## **Lab 5: Processes and signals**

---

**Ahmed Baha Eddine Alimi**

**Assignment Report**

**SD-01**

# I. Questions to Answer :

## 1. What are zombie processes? How can you find and kill them?

Zombie processes are completed processes whose exit status hasn't been read by their parent. They appear in the process table but consume no resources.

Finding Zombie process: Using **ps** or **top**: (we need to look for **Z** in the **STAT** column.)

```
ps aux
```

OR

```
top
```

Killing Zombie process:

We need to find the PID of the zombie process:

```
ps -o ppid= -p <zombie_pid>
```

And Then we kill the parent:

```
kill -9 <parent_pid>
```

## 2. What are the differences between **kill**, **killall**, and **pkill**?

- **kill**: Targets a process by PID, it's precise and requires the exact process ID.
- **killall**: Kills all processes by name; useful for terminating multiple instances of a program.
- **pkill**: Kills processes by name, user, or regex; flexible and supports advanced filtering.

## 3. Run the **top** command on your system and annotate the data in the Tasks and %Cpu(s) lines of your output. Provide single sentence explanations for each of the data presented in these two lines.

```
top - 19:12:25 up 42 min,  1 user,  load average: 0,12, 0,03, 0,05
Tasks: 209 total,  1 running, 208 sleeping,  0 stopped,  0 zombie
%Cpu(s):  1,0 us,  1,0 sy,  0,0 ni, 95,7 id,  0,1 wa,  0,0 hi,  2,3 si,  0,0 st
MiB Mem :  3899,8 total,  2202,7 free,   808,2 used,   889,0 buff/cache
MiB Swap:  2680,0 total,  2680,0 free,    0,0 used.  2833,3 avail Mem
```

#### Tasks Line:

- **209 total:** There are 209 processes running or managed by the system.
- **1 running:** Only 1 process is actively using the CPU.
- **208 sleeping:** 208 processes are idle or waiting for resources.
- **0 stopped:** No processes are currently stopped
- **0 zombie:** No zombie processes (completed processes waiting for parent to read their exit status).

#### %Cpu(s) Line:

- **1,0 us:** 1.0% of CPU is used by user processes (non-kernel code).
- **1,0 sy:** 1.0% of CPU is used by system processes (kernel code).
- **0,0 ni:** 0.0% of CPU is used by nice processes (user processes with adjusted priority).
- **95,7 id:** 95.7% of CPU is idle (not in use).
- **0,1 wa:** 0.1% of CPU is waiting for I/O operations to complete.
- **0,0 hi:** 0.0% of CPU is handling hardware interrupts.
- **2,3 st:** 2.3% of CPU is stolen by a virtual machine (time lost to a hypervisor).

## 4. Execute the following bash command:

```
$ bash -c "exec -a fun${RANDOM}process sleep infinity" &
```

- Assume that there are multiple of such processes. To simulate this, you can run the command more than once.
- Write a bash script that will locate and kill all the processes created by this command.
- Display status messages when one of such processes is found, and when the process is killed. Additionally, display a message when the process is not found.
- Your script should work on any machine it is executed on.
- Be extremely careful and be as accurate as possible when finding this process. You don't want to kill the wrong process.

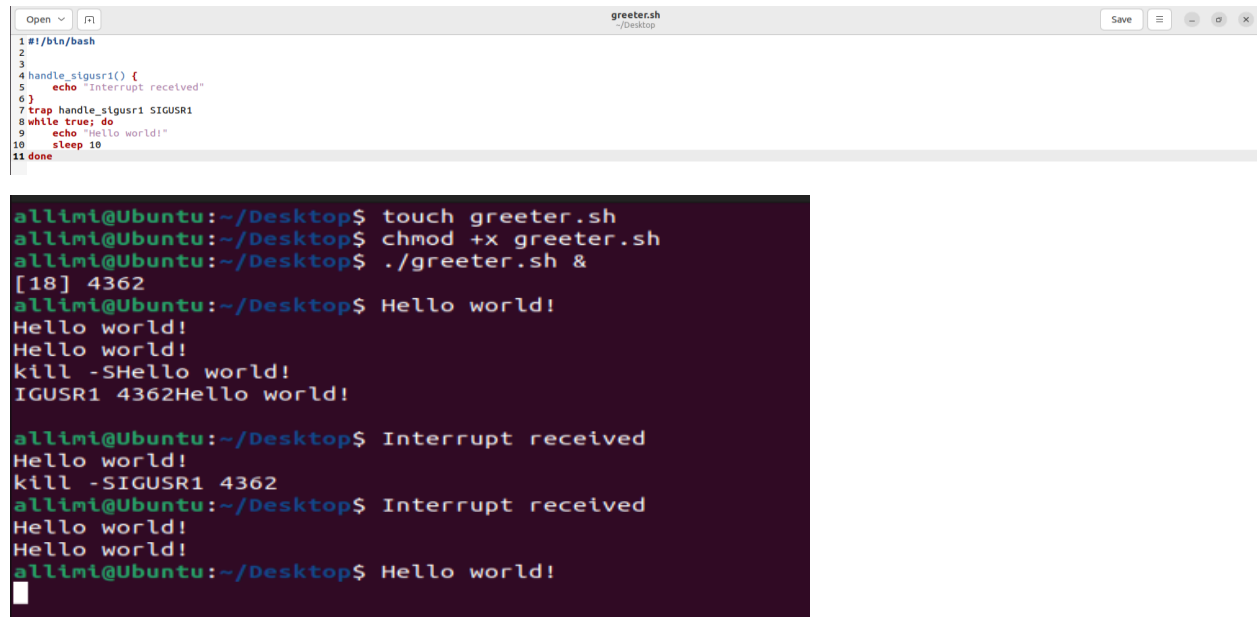
```
Open  hitman.sh
~/Desktop

1 #!/bin/bash
2
3
4 find_and_kill_processes() {
5     # Pattern to match the process names
6     local pattern="fun[0-9]*process"
7
8     # Find all processes matching the pattern
9     local pids=$(pgrep -f "$pattern")
10
11     if [[ -z "$pids" ]]; then
12         echo "No processes matching the pattern '$pattern' were found."
13         return
14     fi
15
16     # Loop through each PID and kill the process
17     for pid in $pids; do
18         local process_name=$(ps -p "$pid" -o comm=)
19         echo "Found process: $process_name (PID: $pid)"
20
21         # Kill the process
22         if kill "$pid"; then
23             echo "Successfully killed process: $process_name (PID: $pid)"
24         else
25             echo "Failed to kill process: $process_name (PID: $pid)"
26         fi
27     done
28 }
29
30 # Execute the function
31 find_and_kill_processes
```

```
allimi@Ubuntu: ~/Desktop
allimi@Ubuntu:~/Desktop$ touch hitman.sh
allimi@Ubuntu:~/Desktop$ bash -c "exec -a fun${RANDOM}process sleep infinity" &
[1] 7017
allimi@Ubuntu:~/Desktop$ bash -c "exec -a fun${RANDOM}process sleep infinity" &
[2] 7018
allimi@Ubuntu:~/Desktop$ bash -c "exec -a fun${RANDOM}process sleep infinity" &
[3] 7019
allimi@Ubuntu:~/Desktop$ bash -c "exec -a fun${RANDOM}process sleep infinity" &
[4] 7020
allimi@Ubuntu:~/Desktop$ bash -c "exec -a fun${RANDOM}process sleep infinity" &
[5] 7021
allimi@Ubuntu:~/Desktop$ bash -c "exec -a fun${RANDOM}process sleep infinity" &
[6] 7022
allimi@Ubuntu:~/Desktop$ bash -c "exec -a fun${RANDOM}process sleep infinity" &
[7] 7023
allimi@Ubuntu:~/Desktop$ chmod +x hitman.sh
allimi@Ubuntu:~/Desktop$ ./hitman.sh
Found process: sleep (PID: 7017)
Successfully killed process: sleep (PID: 7017)
Found process: sleep (PID: 7018)
Successfully killed process: sleep (PID: 7018)
Found process: sleep (PID: 7019)
Successfully killed process: sleep (PID: 7019)
Found process: sleep (PID: 7020)
Successfully killed process: sleep (PID: 7020)
Found process: sleep (PID: 7021)
Successfully killed process: sleep (PID: 7021)
Found process: sleep (PID: 7022)
Successfully killed process: sleep (PID: 7022)
Found process: sleep (PID: 7023)
Successfully killed process: sleep (PID: 7023)
[1] Terminated bash -c "exec -a fun${RANDOM}process sleep infinity"
[2] Terminated bash -c "exec -a fun${RANDOM}process sleep infinity"
[3] Terminated bash -c "exec -a fun${RANDOM}process sleep infinity"
[4] Terminated bash -c "exec -a fun${RANDOM}process sleep infinity"
[5] Terminated bash -c "exec -a fun${RANDOM}process sleep infinity"
[6] Terminated bash -c "exec -a fun${RANDOM}process sleep infinity"
[7]+ Terminated bash -c "exec -a fun${RANDOM}process sleep infinity"
allimi@Ubuntu:~/Desktop$
```

5. Write a bash script that loops infinitely and prints "Hello world!" every ten seconds. It should print "Interrupt received" when it receives **SIGUSR1**.

Show the script in your report, and show how you're sending the signal to it.



```
1 #!/bin/bash
2
3
4 handle_sigusr1() {
5     echo "Interrupt received"
6 }
7 trap handle_sigusr1 SIGUSR1
8 while true; do
9     echo "Hello world!"
10    sleep 10
11 done
```

```
allimi@Ubuntu:~/Desktop$ touch greeter.sh
allimi@Ubuntu:~/Desktop$ chmod +x greeter.sh
allimi@Ubuntu:~/Desktop$ ./greeter.sh &
[18] 4362
allimi@Ubuntu:~/Desktop$ Hello world!
Hello world!
Hello world!
kill -SHello world!
IGUSR1 4362Hello world!

allimi@Ubuntu:~/Desktop$ Interrupt received
Hello world!
kill -SIGUSR1 4362
allimi@Ubuntu:~/Desktop$ Interrupt received
Hello world!
Hello world!
allimi@Ubuntu:~/Desktop$ Hello world!
```

6. Write a bash script to monitor CPU usage, memory usage, and disk space usage.

- For testing purposes, the check should execute every 15 seconds.
- The usage statistics should be saved to a log file **/var/log/system\_utilization.log**.
- One line of log should contain the timestamp, the % of CPU in use, the % of memory in use, and the % of disk space used.
- The log should contain descriptive information that will make it easy to understand.



```
1 #!/bin/bash
2
3 LOG_FILE="/var/log/system_utilization.log"
4
5 touch "$LOG_FILE"
6
7 get_cpu_usage() {
8     cpu_usage=$(top -bn1 | grep "Cpu(s)" | sed "s/./, " | awk '{print 100 - $1}')
9     echo "$cpu_usage"
10 }
11
12 get_memory_usage() {
13     memory_usage=$(free | grep Mem | awk '{print $3/$2 * 100.0}')
14     echo "$memory_usage"
15 }
16
17 get_disk_usage() {
18     disk_usage=$(df -h / | grep / | awk '{print $5}' | sed 's/%//g')
19     echo "$disk_usage"
20 }
21
22 while true; do
23     timestamp=$(date "+%Y-%m-%d %H:%M:%S")
24
25     cpu_usage=$(get_cpu_usage)
26     memory_usage=$(get_memory_usage)
27     disk_usage=$(get_disk_usage)
28
29     echo "Timestamp: $timestamp | CPU Usage: $cpu_usage% | Memory Usage: $memory_usage% | Disk Usage: $disk_usage%" >> "$LOG_FILE"
30
31     sleep 15
32 done
```

```
system_utilization.log
Timestamp: 2023-09-20 10:10:10 | CPU Usage: 1.0% | Memory Usage: 1.0% | Disk Usage: 1.0%
Timestamp: 2023-09-20 10:10:25 | CPU Usage: 1.0% | Memory Usage: 1.0% | Disk Usage: 1.0%
Timestamp: 2023-09-20 10:10:40 | CPU Usage: 1.0% | Memory Usage: 1.0% | Disk Usage: 1.0%
Timestamp: 2023-09-20 10:10:55 | CPU Usage: 1.0% | Memory Usage: 1.0% | Disk Usage: 1.0%
Timestamp: 2023-09-20 10:11:10 | CPU Usage: 1.0% | Memory Usage: 1.0% | Disk Usage: 1.0%
```

monitor.sh				system_utilization.log			
1	Timestamp:	2025-02-24 20:24:05	CPU Usage: 98%	Memory Usage: 34,0311%	Disk Usage: 59%		
2	Timestamp:	2025-02-24 20:24:20	CPU Usage: 96%	Memory Usage: 33,9674%	Disk Usage: 59%		
3	Timestamp:	2025-02-24 20:24:36	CPU Usage: 94%	Memory Usage: 34,3846%	Disk Usage: 59%		
4	Timestamp:	2025-02-24 20:24:51	CPU Usage: 95%	Memory Usage: 34,8783%	Disk Usage: 59%		
5	Timestamp:	2025-02-24 20:25:06	CPU Usage: 100%	Memory Usage: 34,9074%	Disk Usage: 59%		
6	Timestamp:	2025-02-24 20:25:21	CPU Usage: 98%	Memory Usage: 34,9189%	Disk Usage: 59%		
7	Timestamp:	2025-02-24 20:25:36	CPU Usage: 100%	Memory Usage: 34,9096%	Disk Usage: 59%		
8	Timestamp:	2025-02-24 20:25:52	CPU Usage: 92%	Memory Usage: 34,9089%	Disk Usage: 59%		
9	Timestamp:	2025-02-24 20:26:07	CPU Usage: 100%	Memory Usage: 34,9067%	Disk Usage: 59%		
10	Timestamp:	2025-02-24 20:27:28	CPU Usage: 93%	Memory Usage: 34,8614%	Disk Usage: 59%		
11	Timestamp:	2025-02-24 20:27:43	CPU Usage: 97%	Memory Usage: 34,6095%	Disk Usage: 59%		
12	Timestamp:	2025-02-24 20:27:58	CPU Usage: 95%	Memory Usage: 34,713%	Disk Usage: 59%		
13	Timestamp:	2025-02-24 20:28:14	CPU Usage: 97%	Memory Usage: 34,7625%	Disk Usage: 59%		
14	Timestamp:	2025-02-24 20:28:29	CPU Usage: 97%	Memory Usage: 34,8081%	Disk Usage: 59%		