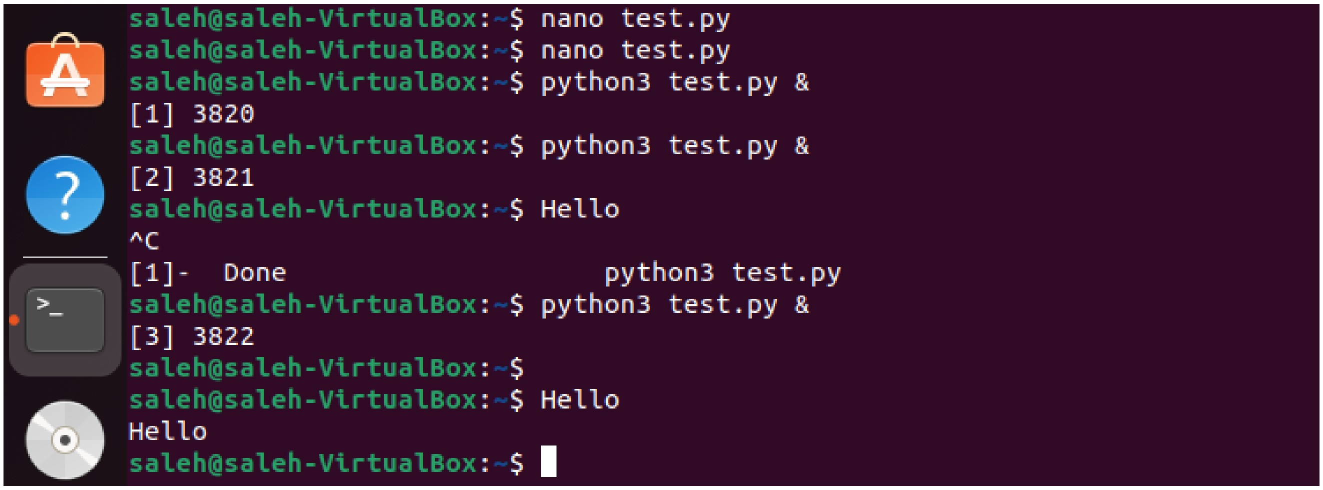
Day 4 – Phase 4 Process and Network Monitoring

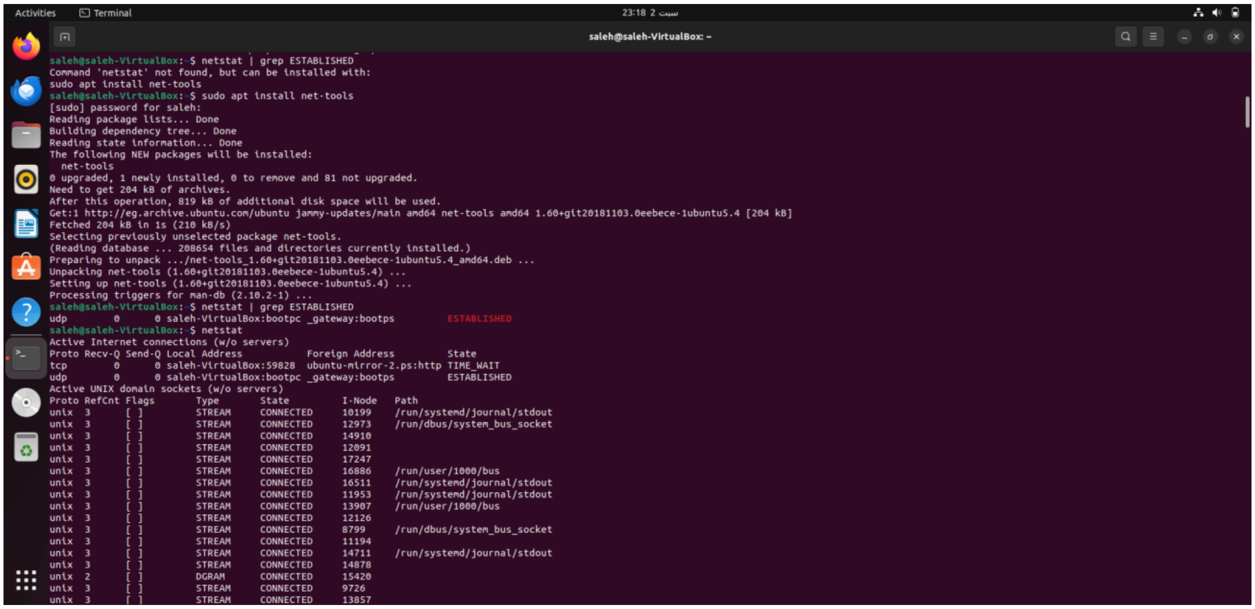
Tasks:

1.



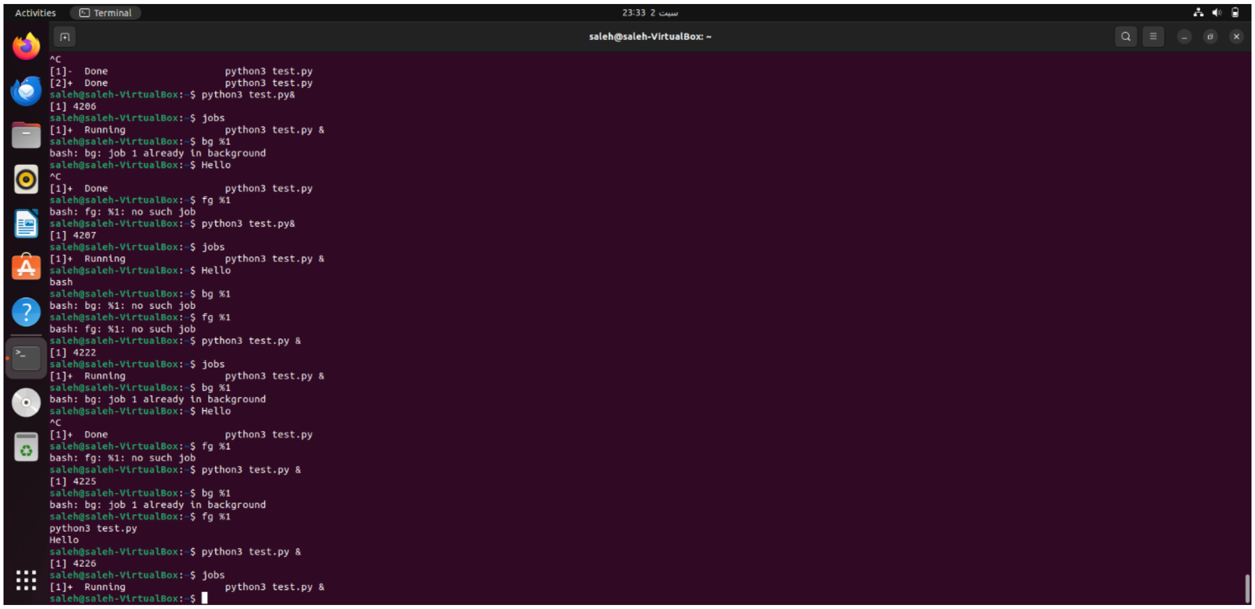
```
saleh@saleh-VirtualBox:~$ nano test.py
saleh@saleh-VirtualBox:~$ nano test.py
saleh@saleh-VirtualBox:~$ python3 test.py &
[1] 3820
saleh@saleh-VirtualBox:~$ python3 test.py &
[2] 3821
saleh@saleh-VirtualBox:~$ Hello
^C
[1]-  Done                    python3 test.py
saleh@saleh-VirtualBox:~$ python3 test.py &
[3] 3822
saleh@saleh-VirtualBox:~$
saleh@saleh-VirtualBox:~$ Hello
Hello
saleh@saleh-VirtualBox:~$ 
```

2,3.



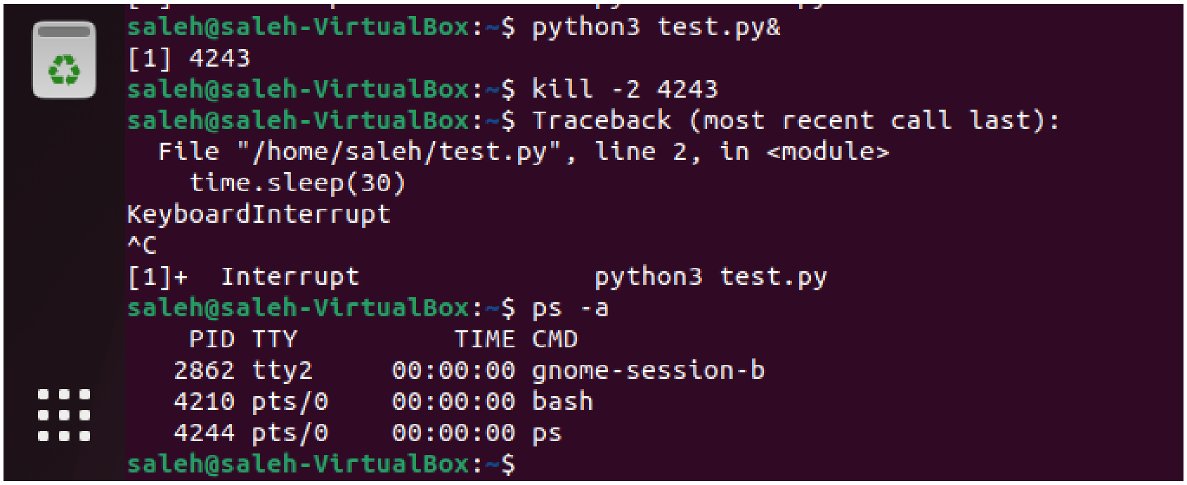
```
saleh@saleh-VirtualBox:~$ netstat | grep ESTABLISHED
Command 'netstat' not found, but can be installed with:
sudo apt install net-tools
saleh@saleh-VirtualBox:~$ sudo apt install net-tools
[sudo] password for saleh:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  net-tools
0 upgraded, 1 newly installed, 0 to remove and 81 not upgraded.
Need to get 204 kB of archives.
After this operation, 819 kB of additional disk space will be used.
Get:1 http://eg.archive.ubuntu.com/ubuntu jammy-updates/main amd64 net-tools amd64 1.60+git20181103.0eebece-1ubuntu5.4 [204 kB]
Fetched 204 kB in 1s (210 kB/s)
Selecting previously unselected package net-tools.
(Reading database ... 208654 files and directories currently installed.)
Preparing to unpack .../net-tools_1.60+git20181103.0eebece-1ubuntu5.4_amd64.deb ...
Unpacking net-tools (1.60+git20181103.0eebece-1ubuntu5.4) ...
Setting up net-tools (1.60+git20181103.0eebece-1ubuntu5.4) ...
Processing triggers for man-db (2.10.2-1) ...
saleh@saleh-VirtualBox:~$ netstat | grep ESTABLISHED
udp        0      0 saleh-VirtualBox:bootpc _gateway:bootps         ESTABLISHED
saleh@saleh-VirtualBox:~$ netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 saleh-VirtualBox:59828  ubuntu-mirror-2.ps:http TIME_WAIT
udp        0      0 saleh-VirtualBox:bootpc _gateway:bootps         ESTABLISHED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  3      [ ]         STREAM     CONNECTED     10199    /run/systemd/journal/stdout
unix  3      [ ]         STREAM     CONNECTED     12973    /run/dbus/system_bus_socket
unix  3      [ ]         STREAM     CONNECTED     14910
unix  3      [ ]         STREAM     CONNECTED     12091
unix  3      [ ]         STREAM     CONNECTED     17247
unix  3      [ ]         STREAM     CONNECTED     16886    /run/user/1000/bus
unix  3      [ ]         STREAM     CONNECTED     16511    /run/systemd/journal/stdout
unix  3      [ ]         STREAM     CONNECTED     11953    /run/systemd/journal/stdout
unix  3      [ ]         STREAM     CONNECTED     13907    /run/user/1000/bus
unix  3      [ ]         STREAM     CONNECTED     12126
unix  3      [ ]         STREAM     CONNECTED     8799     /run/dbus/system_bus_socket
unix  3      [ ]         STREAM     CONNECTED     11194
unix  3      [ ]         STREAM     CONNECTED     14711    /run/systemd/journal/stdout
unix  3      [ ]         STREAM     CONNECTED     14878
unix  2      [ ]         DGRAM      CONNECTED     15420
unix  3      [ ]         STREAM     CONNECTED     9726
unix  3      [ ]         STREAM     CONNECTED     13857
```

4.



```
^C
[1]-  Done                    python3 test.py
[2]+  Done                    python3 test.py
saleh@saleh-VirtualBox:~$ python3 test.py&
[1] 4206
saleh@saleh-VirtualBox:~$ jobs
[1]+  Running                 python3 test.py &
saleh@saleh-VirtualBox:~$ bg %1
bash: bg: job 1 already in background
saleh@saleh-VirtualBox:~$ Hello
^C
[1]+  Done                    python3 test.py
saleh@saleh-VirtualBox:~$ fg %1
bash: fg: %1: no such job
saleh@saleh-VirtualBox:~$ python3 test.py&
[1] 4207
saleh@saleh-VirtualBox:~$ jobs
[1]+  Running                 python3 test.py &
saleh@saleh-VirtualBox:~$ Hello
bash
saleh@saleh-VirtualBox:~$ bg %1
bash: bg: %1: no such job
saleh@saleh-VirtualBox:~$ fg %1
bash: fg: %1: no such job
saleh@saleh-VirtualBox:~$ python3 test.py &
[1] 4222
saleh@saleh-VirtualBox:~$ jobs
[1]+  Running                 python3 test.py &
saleh@saleh-VirtualBox:~$ bg %1
bash: bg: job 1 already in background
saleh@saleh-VirtualBox:~$ Hello
^C
[1]+  Done                    python3 test.py
saleh@saleh-VirtualBox:~$ fg %1
bash: fg: %1: no such job
saleh@saleh-VirtualBox:~$ python3 test.py &
[1] 4225
saleh@saleh-VirtualBox:~$ bg %1
bash: bg: job 1 already in background
saleh@saleh-VirtualBox:~$ fg %1
python3 test.py
Hello
saleh@saleh-VirtualBox:~$ python3 test.py &
[1] 4226
saleh@saleh-VirtualBox:~$ jobs
[1]+  Running                 python3 test.py &
saleh@saleh-VirtualBox:~$ 
```

5.



```
saleh@saleh-VirtualBox:~$ python3 test.py&
[1] 4243
saleh@saleh-VirtualBox:~$ kill -2 4243
saleh@saleh-VirtualBox:~$ Traceback (most recent call last):
  File "/home/saleh/test.py", line 2, in <module>
    time.sleep(30)
KeyboardInterrupt
^C
[1]+  Interrupt               python3 test.py
saleh@saleh-VirtualBox:~$ ps -a
    PID TTY          TIME CMD
   2862 tty2     00:00:00 gnome-session-b
   4210 pts/0    00:00:00 bash
   4244 pts/0    00:00:00 ps
saleh@saleh-VirtualBox:~$
```

Open-Ended Questions:

Q1. What happens when you type a command in Bash until you see the output?

When you enter any command in Bash. The shell parses your input command while dividing it into the command and the arguments. Fetching commands for its own self built commands is checked. For commands that aren't self built and for instance for 'ls', the shell looks through the directories using the PATH environment variable. After the binary is located, the shell spawns a new process to run the binary. The program requests the kernel for hardware resources with system calls, like reading the filesystem or printing to the screen. The kernel handles these system calls, and the output is returned by the shell which displays it in the terminal.

2. Explain the types of processes in Linux: daemon, zombie, orphan. How can you detect them?

Just like with other operating systems, each process in Linux can be in different states. A daemon is a process running in the background that has no controlling terminal and usually a service to run—as with sshd, which manages SSH connections. A zombie process is a completed process that still has an entry in the process table because its parent process has not yet collected its exit status. It can be detected with ps aux showing a state of Z. An orphan process in Linux is a process whose parent has exited, causing it to run unsupervised. These orphan processes in Linux are taken care of by being adopted by init or systemd. You can check for these process states by issuing the commands ps -el or top; these show all processes and their states, such as Z for zombie, S for sleeping, and others.

3. Why do we need Inter-Process Communication (IPC)? List some IPC mechanisms and real-life examples.

Collaborating, Sharing, or Coordinating Always Requires Inter-Process Communication. Effective Data Sharing Needs Inter Process Communication as Each Process Has Its Own Separate Memory Space. In Linux, IPC is Supported by Several Mechanisms. One such Mechanism is Shared Memory, which Permits Multiple Users to Access Common Memory but Requires Synchronization. Message Queues Allow Processes to Exchange Compact Messages, and “Sockets” Support Communication Over a Network or Between Local Processes. Message Queues Allow Processes to Exchange Compact Messages, and

“Sockets” Support Communication Over a Network or Between Local Processes. For Instance, a Web Server (Apache or Nginx) Uses Sockets to Communicate with its Worker Processes, and the Pipe ls | grep ".txt" Helps One Program (ls) to Streamline Its Output as Another Program's (grep's) Input.