ELSEWEDY UNIVERSITY OF TECHNOLOGY
- POLYTECHNIC OF EGYPT -

Operating System

Lab 6: process management system in linux

© Fall 2023 - **Eng. Taghreed Salem – Eng.Nahla Hafez**
Building, office No.
Taghreed.salem@sut.edu.eg

Table of Contents

- What is Process Management in Linux?
- Types of process
- Process States in Linux

What is process :

- A **process** in an operating system (OS) is a running instance of a program. It represents the program in action, using system resources like the CPU, memory, and I/O devices to perform its tasks.
- **Key Characteristics of a Process:**

1.Dynamic Nature:A process is active and changes states as it executes (e.g., running, waiting, terminated).
2.System Resources:It uses the CPU for computation, memory for data storage, and I/O devices for interaction.
3.Unique Identifier (PID):Every process has a Process ID (PID), a unique number for tracking and management.
4.Lifecycle:Processes go through a lifecycle with these stages:

- Created: When initiated.
- Running: Actively executed by the CPU.
- Waiting: Paused, waiting for resources or user input.
- Terminated: Completed or stopped.

5.Parent-Child Relationship:A process can create other processes (child processes), making it the parent process.

What is process management in linux?

- Process management in Linux is an essential skill for Linux administrators and developers.
- It involves controlling and monitoring the processes running on a Linux system, including managing process resources, scheduling processes to run on the CPU, and terminating processes when necessary. Understanding the different types of processes, their states, and the available commands for process management, such as ps, top, kill, nice, and renice, are important for managing processes effectively.

Types of process

In Linux, there are two main types of processes based on how they interact with the user:

1. **Foreground Processes** :

- These are the processes you interact with directly.
- They need your input to work, and you can see them running on the screen.
- Example: When you open a text editor or run a command like nano, you're working with a foreground process.

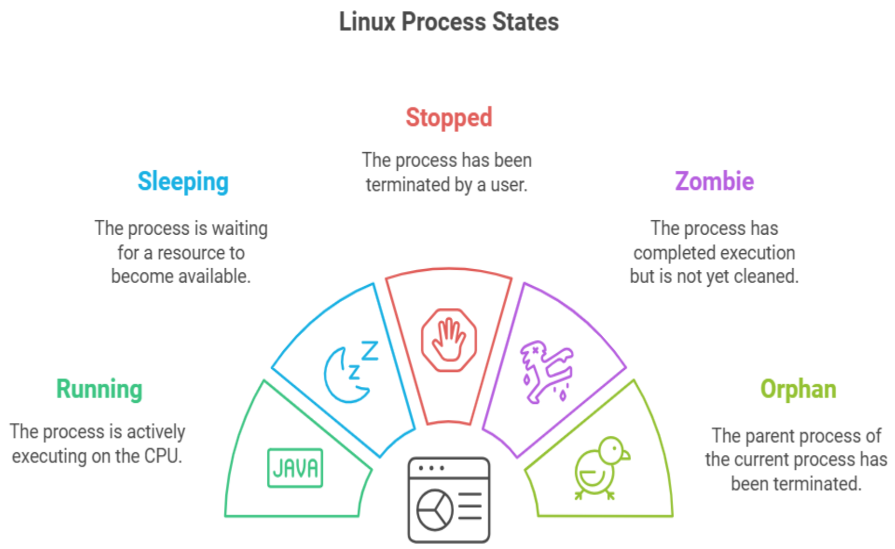
2. **Background Processes** :

- These processes run quietly in the background and don't need your attention or input.
- They keep working even if you don't interact with them.
- Example: An antivirus program that runs automatically in the background to scan files.

Process States in Linux

In Linux, a process can be in one of five states

- **Running:**
  The process is currently executing on the CPU.
- **Sleeping:**
  The process is waiting for a resource to become available.
- **Stopped:**
  The process has been terminated by a user
- **Zombie:**
  The process has completed execution but has not yet been cleaned by the system.
- **Orphan:**
  The parent process of the current process has been terminated.


Linux Process States
Stopped
The process has been terminated by a user.
Sleeping
The process is waiting for a resource to become available.
Zombie
The process has completed execution but is not yet cleaned.
Running
The process is actively executing on the CPU.
Orphan
The parent process of the current process has been terminated.
JAVA

Command For Process Management in Linux

Linux provides several commands for managing processes, which include:

Commands	Description
ps	Displays information about the processes running currently.
top	Provides real-time information about system processes and their resource usage.
kill	Terminates a process by sending a signal to it.
nice	Adjusts the priority of a process.
renice	Changes the priority of a running process.

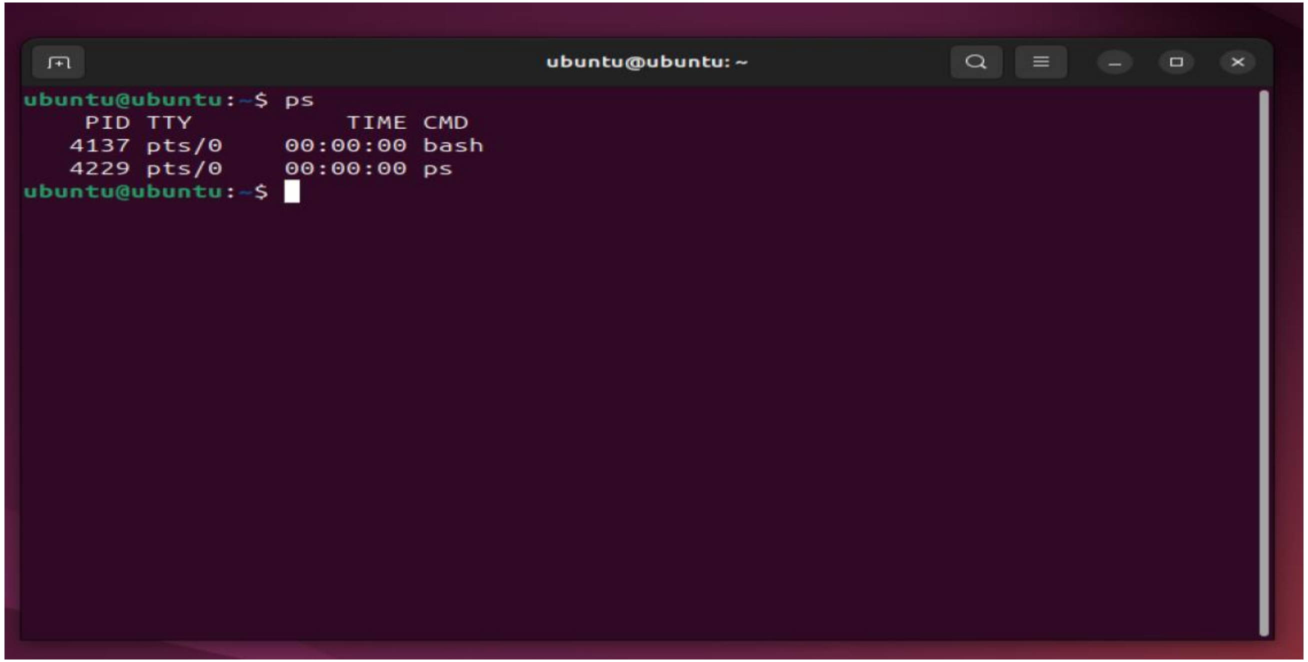
ELSEWEDY UNIVERSITY OF TECHNOLOGY
- POLYTECHNIC OF EGYPT -

Practically Managing the Processes

Identifying and Terminating Processes in Linux.

dentification of processes and termination of processes can be done by using the below-mentioned commands in Linux.

The **ps** command is used to find the process ID (PID) of the process you want to manage. As instance, if it's required to kill a process you need to know the process ID (PID) of that exact process.



```
ubuntu@ubuntu:~$ ps
    PID TTY          TIME CMD
   4137 pts/0    00:00:00 bash
   4229 pts/0    00:00:00 ps
ubuntu@ubuntu:~$ 
```

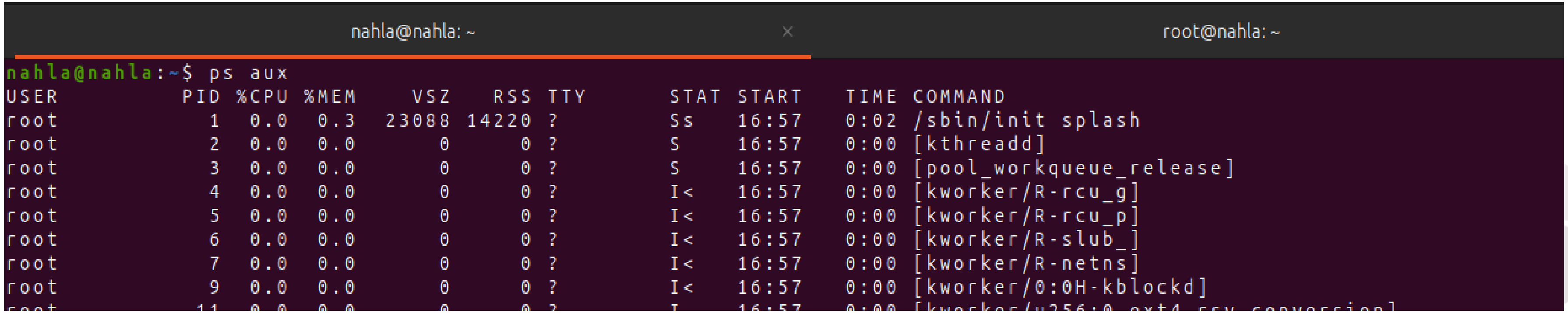
Cont.

- **ps aux:** This is a common option to get a detailed list of processes

a: Lists processes from all users.
u: Displays detailed user information.
x: Includes processes not attached to a terminal.

Common Columns:

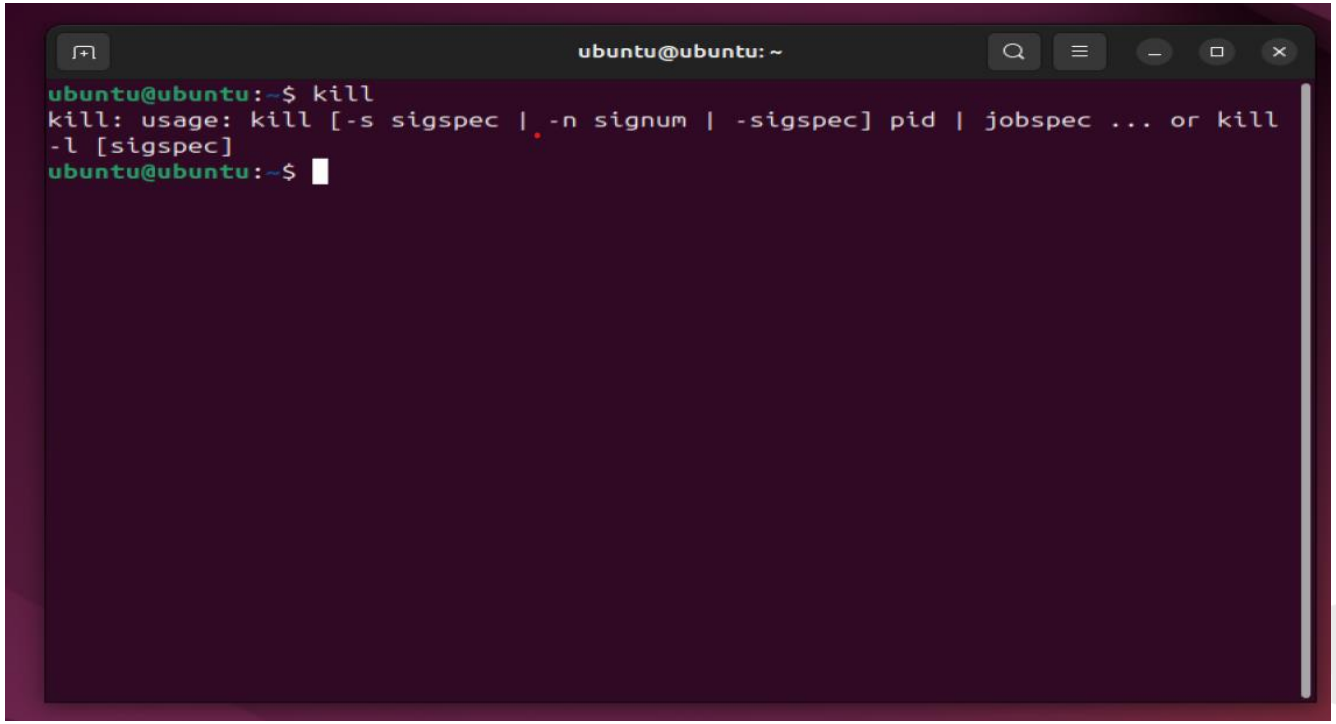
1.PID: The Process ID, a unique number assigned to each process.
2.USER: The user name associated with the process.
3.%CPU: The percentage of CPU time used by the process.
4.%MEM: The percentage of physical memory used by the process.
5.VSZ: The virtual memory size of the process.
6.RSS: The resident set size, or the amount of physical memory used by the process.
7.TTY: The terminal device associated with the process.
8.STAT: The process state, which can be various codes indicating different states like running, sleeping, or waiting.
9.START: The time the process started.
10.TIME: The cumulative CPU time used by the process.
11.COMMAND: The command line used to start the process.



```
nahla@nahla:~$ ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.3  23088 14220 ?        Ss   16:57   0:02 /sbin/init splash
root           2  0.0  0.0      0     0 ?        S    16:57   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        S    16:57   0:00 [pool_workqueue_release]
root           4  0.0  0.0      0     0 ?        I<   16:57   0:00 [kworker/R-rcu_g]
root           5  0.0  0.0      0     0 ?        I<   16:57   0:00 [kworker/R-rcu_p]
root           6  0.0  0.0      0     0 ?        I<   16:57   0:00 [kworker/R-slub_]
root           7  0.0  0.0      0     0 ?        I<   16:57   0:00 [kworker/R-netns]
root           9  0.0  0.0      0     0 ?        I<   16:57   0:00 [kworker/0:0H-kblockd]
```

kill command

- The **kill** command in Linux is used to terminate processes by their process IDs (PIDs).



```
ubuntu@ubuntu:~$ kill
kill: usage: kill [-s sigspec | -n signum | -sigspec] pid | jobspec ... or kill
-l [sigspec]
ubuntu@ubuntu:~$ 
```

Cont.

- The **kill** command in Linux is used to send signals to processes, which can be used to terminate them or to instruct them to perform specific actions. Below, I will explain its syntax, usage, common signals, and examples.

Common Signals

The kill command can send various signals to processes. Here are some of the most commonly used signals:

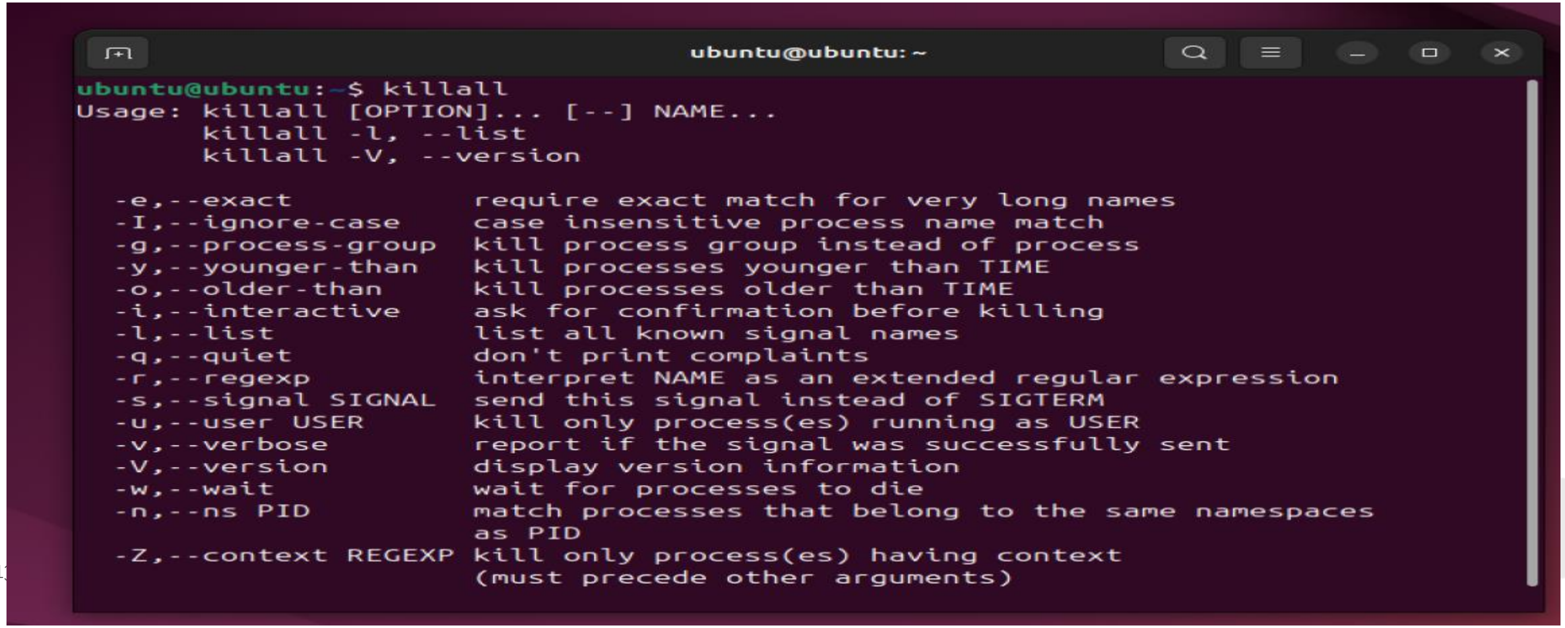
- SIGTERM (15): The default signal to terminate a process gracefully. It allows the process to clean up resources before exiting.(default)
- SIGKILL (9): Forces termination of a process immediately without cleanup. This signal cannot be ignored.
- SIGINT (2): Sent when you press Ctrl + C in the terminal. It interrupts a running process.

Options

- -l: Lists all available signals.
- -s SIGNAL: Specifies a signal to send. For example, kill -s SIGKILL <PID> .

Cont.

The **killall** command in Linux is used to terminate processes by their names. It sends a signal to all processes with a specified name, effectively killing them.
Ex: killall nano



```
ubuntu@ubuntu:~$ killall
Usage: killall [OPTION]... [--] NAME...
       killall -l, --list
       killall -V, --version

  -e,--exact          require exact match for very long names
  -I,--ignore-case    case insensitive process name match
  -g,--process-group  kill process group instead of process
  -y,--younger-than   kill processes younger than TIME
  -o,--older-than     kill processes older than TIME
  -i,--interactive    ask for confirmation before killing
  -l,--list           list all known signal names
  -q,--quiet          don't print complaints
  -r,--regexp         interpret NAME as an extended regular expression
  -s,--signal SIGNAL  send this signal instead of SIGTERM
  -u,--user USER      kill only process(es) running as USER
  -v,--verbose        report if the signal was successfully sent
  -V,--version        display version information
  -w,--wait           wait for processes to die
  -n,--ns PID         match processes that belong to the same namespaces
                      as PID
  -Z,--context REGEXP kill only process(es) having context
                      (must precede other arguments)
```

Analyzing Resource Usage in Linux.

- The **top** command in Linux is used to display real-time information about processes running on a system, including CPU and memory usage. It provides an interactive interface that allows users to monitor and manage processes. Is used To display a *real-time* view of running processes and their system resource usage, including CPU, memory, and more. It is like task manager in windows.
- **Output of top:**

Header Section (System Overview):

- Tasks: Total processes (running, sleeping, stopped, and zombie).
- CPU Usage: Breakdown of how the CPU is being used (e.g., by user processes, system processes, idle time).
- Memory Usage: Total memory, used, free, and available.
- Swap Usage: Information about swap memory (virtual).

Process List:

- PID: Process ID (unique identifier for each process).
- USER: User who owns the process.
- PR: Priority of the process.
- NI: Nice value (affects priority).
- VIRT: Virtual memory used by the process.
- RES: Resident memory (physical memory) used.
- %CPU: Percentage of CPU usage by the process.
- %MEM: Percentage of memory usage by the process.
- TIME+: Total CPU time the process has used.
- COMMAND: The name of the command or program running.

Linux priority :

- In Linux, **process priority** determines how much CPU time a process gets. The priority is influenced by the nice value, which users can adjust to make a process run faster or slower compared to others. Here's how to work with priority commands:

1.*Priority (PR):*A number assigned to processes by the OS scheduler to decide the order of execution. Lower values mean higher priority.Normal processes have priority values between 0 and 39.

2.*Nice Value (NI):*A user-set value that indirectly affects priority.Ranges from -20 (highest priority) to +19 (lowest priority). Default is 0.

Managing Priority of Processes in Linux

- Managing the priority of a running process is done using “nice” and “renice” commands in Linux.

- The **nice** command in Linux is used to modify the priority of a process. It assigns a lower priority to a process to reduce its resource usage.
- Ex : nice ni process .

- The **renice** command is used to modify the priority of an already running process. It can increase or decrease the priority of a process, depending on the specified value.
- Ex : sudo renice –ni <PID>

Conclusion

- Processes are instances of programs that are currently running on a Linux system.
- Processes can be categorized into foreground, background, and system and user processes.
- Processes can be in one of five states: **running, sleeping, stopped, zombie, or orphan**.
- Process management involves controlling and monitoring the processes running on a Linux system.
- Linux provides several commands for managing processes, including **ps, top, kill, nice, and renice.**
- Practically managing processes involves using these commands to identify and terminate processes, adjust process priorities, and monitor system resource usage.