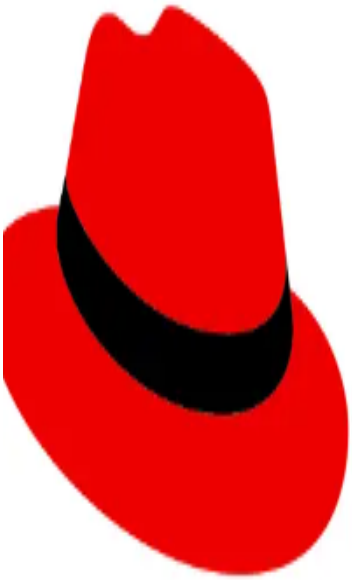


Red Hat Enterprise Linux 9

RHCSA-SA1 LAB-Book

Chapter 8 – Monitoring and Managing Linux Processes

What is a process?

A process is a running instance of a launched, executable program.

Usually, it is used to get the more and detailed information about a specific process or all processes.

For example, it is used to know whether a particular process is running or not, who is running what process in system, which process is using higher memory or CPU, how long a process is running, etc.

A program consists of:

-An address space of allocated memory.

-Security properties including ownership credentials and privileges. -One or more execution threads of program code, and -The process states.

The environment of a process includes:

-Local and Global variables,

-A current scheduling context, and

-Allocated system resources, such as file descriptors and the network ports.

Diagram

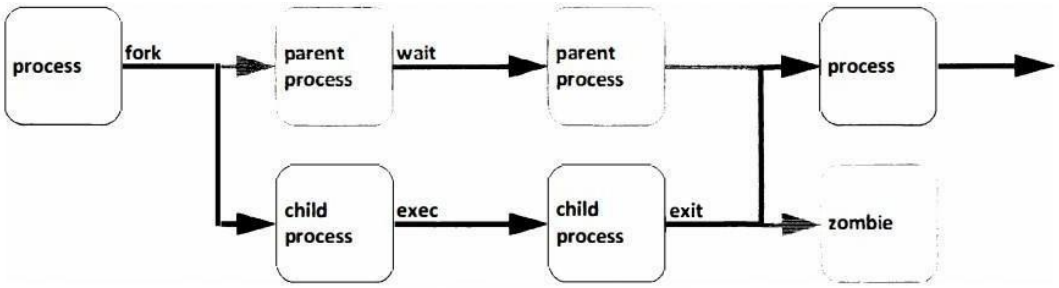


Figure 7.1: Process life cycle

An existing (parent) process duplicates its own address space (fork) to create a new (child) process structure.

Every new process is assigned a unique process ID (PID) for tracking and security.

The PID and the parent's process ID (PPID) are elements of the new process environment.

Any process may create a child process.

All processes are descendants of the first system process, which is systemd () on a Red Hat Enterprise Linux 7 system.

Through the fork routine, a child process inherits security identifies, previous and current file descriptors, port and resource privileges, environment variables, and program code.

A child process may then exec its own program code.

Normally, a parent process sleeps while the child process runs, setting a request (wait) to be signalled when then child completes.

Upon exit, the child process has already closed or discarded its resources and environment the remainder is referred to as a Zombie.

The parent, signalled awake when the child exited, cleans the remaining structure, then continues with its own program code execution.

Process States:

In a multitasking operating system, each CPU can be working on one process at a single point in time.

Diagram

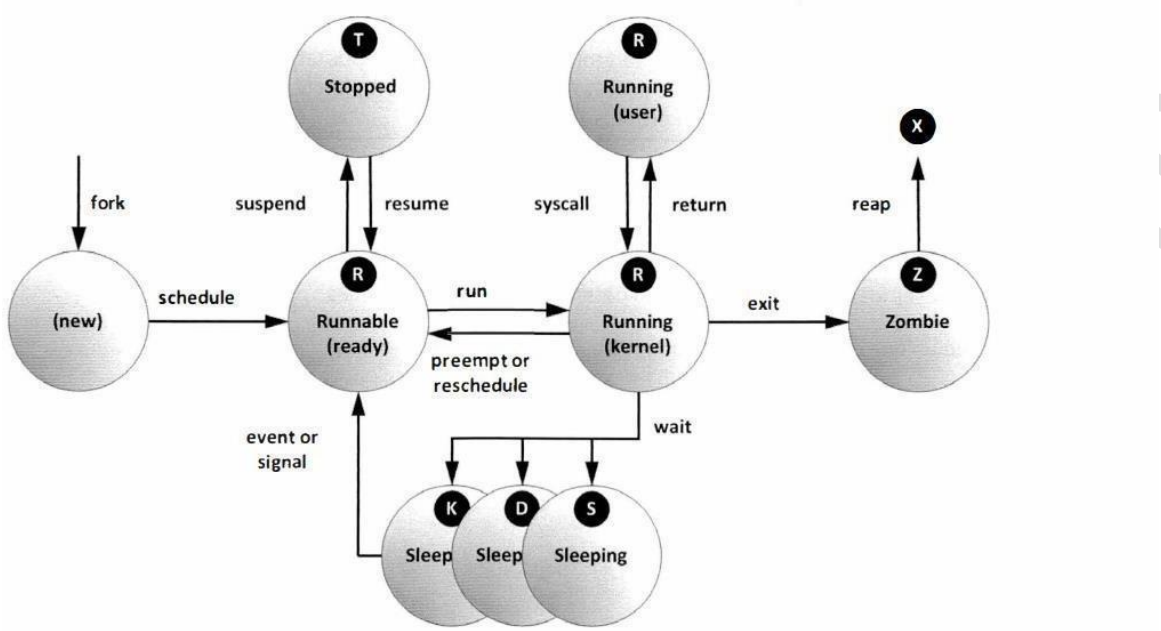


Figure 7.2: Linux process states

Linux process states:

Name	Flag	Kernel -Defined State Name and Description.
New	**N**	The process is being created.
Running	**R**	The process is either executing on a CPU or waiting for run. Process can be executing user routines or kernel routines (system calls,) or be queued and ready when in the Running (or Runnable) State.

Sleeping	**S**	The process is waiting for some condition: a hardware request, system resource access, or signal. When an event or signal satisfies the condition, the process returns to Running.
	D	This process is also Sleeping, but unlike S state, will not respond to delivered signals.
		Used only under specific conditions in which process interruption may cause an un Predictable device state.
	K	Identical to the uninterruptible D state, but modified to allow the waiting task to respond to a signal to be killed (exited completely). Utilities frequently display Killable processes as D state.
Stopped	**T**	The process has been Stopped (suspended), usually by being signalled by a user or another process. The process can be continued (resumed) by another signal to Return to Running.
	T	A process that is begin debugged is also Temporarily Stopped and shares the same T state flag.
Zombie	**Z**	A child process signals its parent as it exits. All resources except for the process identity (PID) are released.
	X	When the parent cleans up (reaps) the remaining child process structure, the process is now released completely. This state will never be observed in process-Listing utilities.

Name	Flag	Kernel -Defined State Name and Description.
Running	**R**	Runnable
Sleeping	**S**	Interruptible
	D	Un - Interruptible
	K	Update Version
Stopped	**T**	Task Stopped
	T	Task Traced- Debugging Process.
Zombie	**Z**	Exit Process
	X	Dead

There are several different types of options:

1. UNIX Style: Preceded by a single

dash.

2. BSD Style: Used without dash.

3. GNU Style: Preceded by two dashes.

A] To see every process on the system using standard syntax:

1) ps: The ps command is used for listing current processes. It includes below:

PID	process ID
TTY	terminal type

TIME	total time the process has been running
CMD	name of the command that launches the process

Important:

Linux version of ps supports three options formats, such as:

- ✓ UNIX (POSIX) options, which may be grouped and must be preceded by a dash.
- ✓ BSD options, which may be grouped and must not be used with a dash, and
- ✓ GNU long options, which are preceded by two dashes.
- ✓ For e.g., ps -aux is not the same as ps aux.

2)ps -A or ps -e: To print all running processes in system:

The options A and e provide summarized overview of running processes It includes below:

PID	Process ID
TTY	Terminal type
TIME	It is used to determine the duration of execution of a particular command
CMD	Name of the command that launches the process.

3) ps -ef: It **display** all processes. Whereas "f" for full format. It includes below:

UID	Unique ID
PID	Process ID
PPID	Parent Process ID
C	Integer value of the processor utilisation percentage. For. e.g., %CPU.
STIME	Set Time

TTYP	Terminal type
TIME	It is used to determine the duration of execution of a particular command
CMD	Name of the command that launches the process.

4) ps -eF: It display all processes. Whereas “F” for Extra full format.

It includes below:

UID	Unique ID
PID	Process ID
PPID	Parent Process ID
C	Integer value of the processor utilisation percentage. For. e.g., %CPU.
SZ	Memory size of the process in KiB. Size in physical pages of the core image of the process. This includes text, data, and stack space.
RSS	Resident Set Size, the non-swappable physical memory used by this process (in KiB)
PSR	Processor that process is currently assigned to.
STIME	Set Time
TTYP	Terminal type
TIME	It is used to determine the duration of execution of a particular command
CMD	Name of the command that launches the process.

5) ps -ely: It display all processes. Whereas “F” for Extra full format. It includes below:

S or STATE	Process status code.

UID	Unique ID
PID	Process ID
PPID	Parent Process ID
C	Integer value of the processor utilisation percentage. For. e.g., %CPU.
PRI	It is the processes actual priority.
NI	It is a nice value, which is a user -space concept.
RSS	Resident Set Size, the non-swappable physical memory used by this process (in KiB)
SZ	Memory size of the process in KiB. Size in physical pages of the core image of the process. This includes text, data, and stack space.
WCHAN	Waiting channel. Address of the kernel function where the process is sleeping (use when if you want the kernel function name). Running task will display a dash (“- “) in this column.
TTY	Terminal type
TIME	It is used to determine the duration of execution of a particular command
CMD	Name of the command that launches the process.
NWCHAN	Name of the kernel function in which the process is sleeping, a “-“if the process is running, OR a “*” if the process is multi-threaded and ps is not displaying threads. -n name list: Set name list file. Identical to N. The name list file is needed for a proper WCHAN display, and must match the current Linux kernel exactly for correct output.

B] To see every process on the system using standard syntax:

1) ps aux: OR ps -aux: Collectively the options "aux" prints all the running process in system regardless from where they have been executed.

It includes below:

a - This option prints the running processes from all users.

u - This option shows user or owner column in output.

x- This option prints the processes those have not been executed from the terminal.

It includes below:

USER	The user account under which this process is running.
PID	Process ID of this process
%CPU	CPU time used by this process (in percentage).
%MEM	Physical memory used by this process (in percentage).
VSZ	Virtual memory used by this process (in bytes).
RSS	Resident Set Size, the non-swappable physical memory used by this process (in KiB)
TTY	Terminal from which this process is started. Question mark (?) sign represents that this process is not started from a terminal.
STAT	Process state. Explained in next table.
START	Starting time and date of this process
TIME	Total CPU time used by this process

Whereas,

D = Un-Interruptible sleep (usually IO)

R = Running or runnable (on run queue)

S = Interruptible sleep (Waiting for an event to complete)

T = Stopped by job control signal

t = Stopped by debugger during the tracking

w = Paging (No valid since the 2.6.xx kernel)

x = Dead (Should never be seen)

Z = Defunct (“zombie”) process, terminated but not reaped by its parent.

< = High – Priority (not nice to others users)

N = Low – Priority (nice to others users)

L = Low – Priority locked into memory (for real – time and custom IO)

s = Is a session leader

I = Is multi-threaded (using CLONE_THREAD, like NPTLP threads do)

+ = Is in the foreground process group.

2)ps ax:

a - This option prints the running processes from all users.

x- This option prints the processes those have not been executed from the terminal.

PID	Process ID of this process
TTY	Terminal from which this process is started. Question mark (?) sign represents that this process is not started from a terminal.
STAT	Process state. Explained in next table.
TIME	Total CPU time used by this process
COMMAND	The command with all its arguments which started this process.

3)ps lax: Listing all process. It includes below:

F	
UID	Effective User ID.

PID	A number of representing a process ID
PPID	Parent process ID.
PRI	Priority of the process. Higher number means lower priority.
NI	Nice value. This ranges from 19 (nicest) to -20 (not nice to others).
VSZ	Virtual memory size of the process in KiB (1024-byte units). Device mappings are currently excluded, this a subject to change.
RSS	Resident set size, the non-swapped physical memory that a task has used (in kilobytes). (alias rs size, esz).
WCHAN	Name of the kernel function in which the process is sleeping, a "- "if the process is running, or a "*" if the process is multi-threaded and ps is not displaying threads.
STAT	Multi-character process state.
TTY	Controlling tty (Terminal).
TIME	Cumulative CPU Time.
COMMAND	Modification to the command name will not be shown. Command name.

C] To print a process tree:

1. ps -ejH: Whereas, e: To print all running

processes in system.

j: BSD job control format.

H: Show process hierarchy (forest).

It includes below:

PID	Process ID of this process

PGID	Process Group Identifier (Process Group Leader)
SID	Session Identifier (Session Leader)
TTY	Terminal from which this process is started. Question mark (?) sign represents that this process is not started from a terminal.
TIME	Total CPU time used by this process
CMD	The command with all its arguments which started this process

2.ps -axjf:

a - This option prints the running processes from all users.
x- This option prints the processes those have not been executed from the terminal. j - BSD job control format.

Cumulative user time.

f - full format.

It includes below:

PPID	Parent Process ID
PID	Process ID of this process
PGID	Process Group Identifier (Process Group Leader)
SID	Session Identifier (Session Leader)
TTY	Terminal from which this process is started. Question mark (?) sign represents that this process is not started from a terminal.
TPGID	Terminal Process Group ID
STAT	Process state. Explained in next table.
UID	Unique ID
TIME	Total CPU time used by this process

COMMAND	The command with all its arguments which started this process

D] To get info about threads:

1. ps -eLf: Whereas, e: To print all running processes in system.

L: Show threads, possibly with LWP and **NLWP columns.**

f - full format.

It includes below:

UID	Unique ID
PID	Process ID of this process
PPID	Parent Process ID
LWP	Light Weight Process. It's means of achieving multitasking.
C	Integer value of the processor utilisation percentage. For. e.g., %CPU.
NLWP	**NLWP** is the number of threads in the system for the underlying process.
STIME	Set Time
TTY	Terminal from which this process is started. Question mark (?) sign represents that this process is not started from a terminal.
TIME	Total CPU time used by this process
CMD	The command with all its arguments which started this process

2.ps axms: Whereas, a - This option prints the running processes from all users.

x- This option prints the processes those have not been executed from the terminal**.**

m - Show threads after processes. s – It is a session leader.

It includes below:

UID	Unique ID
PID	Process ID of this process
PENDING	Signals pending on the process are distinct from signals pending on individual threads. Use the **m** option or the **-m** option to see both. According to the width of the field, a 32- or 64-bits mask in hexadecimal format is displayed.
BLOCKED	According to the width of the field, a 32 or 64-bit mask in hexadecimal format is displayed.
IGNORED	According to the width of the field, a 32- or 64-bits mask in hexadecimal format is displayed.
CAUGHT	According to the width of the field, a 32- or 64-bits mask in hexadecimal format is displayed.
STAT	Process state. Explained in next table.
TTY	Terminal from which this process is started. Question mark (?) sign represents that this process is not started from a terminal.
TIME	Total CPU time used by this process
COMMAND	The command with all its arguments which started this process

E] To get security info:

1. ps -eo: Whereas, e -To print all running

processes in system**.**

o - The option allows you to specify which columns are displayed w**"o"** hen running the ps command.

It includes below:

PID	Process ID of this process

2. ps -axo euser OR ps -axo user:

a – This option prints the running processes from all users.

x- This option prints the processes those have not been executed from the terminal.

To display also processes without a controlling terminal (kernel processes and daemons) and the default behaviour is not to display them.

o – The "o" option allows you to specify which columns are displayed when running the ps command.

euser / user - Effective user name. This will be the textual user

ID, if it can be permits, or a decimal representation otherwise.

The n option can be representation. (Alias uname, user).

It includes below:

EUSER	List of users.

3. ps -axo ruser:

a – This option prints the running processes from all users.

x- This option prints the processes those have not been executed from the terminal.

To display also processes without a controlling terminal (kernel processes and daemons) and the default behaviour is not to display them.

o – The "o" option allows you to specify which columns are displayed when running the ps command.

ruser - Real User ID. This will be the textual user id, if it can be

obtained and the field width permits, or decimal representation

otherwise**.**

It includes below:

RUSER	List of users.

4.ps -axo suser:

i – This option prints the running processes from all users.
x-This option prints the processes those have not been executed from the terminal.

To display also processes without a controlling terminal (kernel processes and daemons) and the default behaviour is not to display them.

o – The "o" option allows you to specify which columns are displayed when running the ps command.

suser – Saved username. This will be the textual user ID, if it can be obtained and the filed width permits, or a decimal representation otherwise. (Alias suser).

It includes below:

SUSER	List of users.

5.ps -axo fuser:

i – This option prints the running processes from all users.

This option prints the processes those have not been executed from the terminal.

To display also processes without a controlling terminal (kernel **x:** processes and daemons) and the default behaviour is not to display them. o – The “o” option allows you to specify which columns are displayed when running the ps command. fuser – Filesystem access user ID. This will be the textual user ID, if it can be obtained and the filed width permits, or a decimal representation otherwise. (Alias suser).

It includes below:

FUSER	List of users.

6.ps -axo fuser:

a– This option prints the running processes from all users.

x -This option prints the processes those have not been executed from the terminal.

To display also processes without a controlling terminal (kernel processes and daemons) and the default behaviour is not to display them. o – The “o” option allows you to specify which columns are displayed when running the ps command. fuser – Filesystem access user ID. This will be the textual user ID, if it can be obtained and the filed width permits, or a decimal representation otherwise. (Alias suser).

It includes below:

FUSER	List of users.

7. ps -axo f:

a– This option prints the running processes from all users.

x -This option prints the processes those have not been executed from the terminal.

To display also processes without a controlling terminal (kernel processes and daemons) and the default behaviour is not to display them.

The “o” option allows you to specify which columns are displayed **o –** when running the ps command.

f – Flag associated with the process, see the PROCESS FLAGS section. (Alias flag, flags).

It includes below:

F	List of flag number associated with the process.

8.ps -axo comm:

a– This option prints the running processes from all users.

x -This option prints the processes those have not been executed from the terminal.

To display also processes without a controlling terminal (kernel processes and daemons) and the default behaviour is not to display them.

o – The “o” option allows you to specify which columns are displayed when running the ps command.

c- Command name (only the executable name).

Modifications to the command name will not be shown.

A process marked <default> is partly dead, waiting to be fully destroyed by its parent.

The output in this column may contain spaces. (alias ucmd, ucomm).

See also the args format keyboard, the -f option, and the c option. When specified last, this column will extend to the edge of the display. If ps cannot determine display width, as when output is redirected (piped) into a file or another command, the output width is undefined. (It may be 80, unlimited, determined by the TERM variable, and so on). The COLUMNS environment variable or - - cols option may be used to exactly determine the width in this case.

The w or -w option may be also used to adjust width.

It includes below:

COMMAND	List of command name.

9.ps – axo label:

a – This option prints the running processes from all users.
x- This option prints the processes those have not been executed from the terminal.

To display also processes without a controlling terminal (kernel processes and daemons) and the default behaviour is not to display them.

o – The “o” option allows you to specify which columns are displayed when running the ps command.

l- Security label, most commonly used for SE Linux context data. This is for the Mandatory Access Control (“MAC”) found on high-security systems.

It includes below:

LABEL	List of security label.

10.ps -axZ:

It includes below:

LABEL	Security label, most commonly used for SE Linux context data. This is for the Mandatory Access Control (“MAC”) found on high-security systems.
PID	Process ID of this process
TTY	Terminal from which this process is started. Question mark (?) sign represents that this process is not started from a terminal.
STAT	Process state. Explained in next table.
TIME	Total CPU time used by this process
COMMAND	The command with all its arguments which started this process

11.ps -eM:

It includes below:

LABEL	Security label, most commonly used for SE Linux context data. This is for the Mandatory Access Control (“MAC”) found on high-security systems.
PID	Process ID of this process
TTY	Terminal from which this process is started. Question mark (?) sign represents that this process is not started from a terminal.
TIME	Total CPU time used by this process
COMMAND	The command with all its arguments which started this process

F] To see every process running as root (real & effective ID) in user format:

1. ps -U root: User list.

Select by real user ID (RUID) or name.

It selects the processes whose real user name or ID is in the user list.

The real user ID identities the user who created the process.

It includes below:

PID	Process ID of this process
TTYP	Terminal from which this process is started. Question mark (?) sign represents that this process is not started from a terminal.
TIME	Total CPU time used by this process
CMD	The command with all its arguments which started this process

2. ps -u root: User list.

Select by effective user ID (EUID) or name.

This selects the processes whose effective user name or ID is in user list.

The effective user ID describes the user whose file access permissions are used by the process.

Identical to U and --user.

It includes below:

PID	Process ID of this process
TTYP	Terminal from which this process is started. Question mark (?) sign represents that this process is not started from a terminal.
TIME	Total CPU time used by this process

CMD	The command with all its arguments which started this process

3.ps - - User root: User list.

Select by real user ID (RUID) or name. Identical to -U.

It includes below:

PID	Process ID of this process
TTYP	Terminal from which this process is started. Question mark (?) sign represents that this process is not started from a terminal.
TIME	Total CPU time used by this process
CMD	The command with all its arguments which started this process

3.ps - - user root: User list.

Select by effective user ID (EUID) or name. Identical to -u and U.

It includes below:

PID	Process ID of this process
TTYP	Terminal from which this process is started. Question mark (?) sign represents that this process is not started from a terminal.
TIME	Total CPU time used by this process
CMD	The command with all its arguments which started this process

G] To see every process with a user-defined format:

1.ps -eo pid: Display list of Process ID.

It includes below:

PID	A number representing the process ID

2. ps -eo tid: The unique number representing a dispatchable entity (alias lwp, spid). This value may also appear as:

a process ID (pid);

a process group ID (pgrp);

a session ID for the session leader (sid);

a thread group ID for the thread group leader (tgid); and -

a tty process group ID for the process group leader (tpgid).

It includes below:

TID	A number representing the Thread ID

3. ps -eo class or cls or CLS: Scheduling class of the process.

(Alias policy, cls).

Fields possible values are: - not reported.

TS SCHED_OTHER

FF SCHED_FIFO RR

SCHED_RR

ISO SCHED_ISO

IDL SCHED_IDEL

DLN SCHED_DEADLINE

? unknown value

It includes below:

CLS	List of class.

4. ps-eo rtprio: Realtime Priority**.**

It includes below:

RTPRIO	List of real time.

5. ps -eo ni: Display list of nice value. i.e., %n. Represented as NI. This ranges from 19 (nicest) to -20 (not nice to others).

It includes below:

NI	Display list of nice value. i.e., %n.

6. ps -eo pri: Priority of the process. Higher number means lower priority.

It includes below:

PRI	Priority of the process.

6. ps -eo psr: The PSR column shows that init is running on processor 1 and ps is running on processor 0.

It includes below:

PSR	Processor that process is currently assigned to.

7. ps -eo pcpu: Display % CPU**.**

It includes below:

PCPU	Display % CPU list.

8. ps -eo stat: Display Multi -character process state.

It includes below:

STAT

9. ps -eo wchan: Waiting channel.

Name of the kernel function in which the process is sleeping, a "- "if the process is running, or a "*" if the process is multi-threaded and ps is not displaying threads**.**

It includes below:

WCHAN	Waiting channel.

10. ps-axo stat: Multi – character process state.

It includes below:

STAT

11. ps -axo euid: Effective user ID. (Alias uid).

It includes below:

EUID

12. ps -axo ruid: Real user ID.

It includes below:

RUID

13. ps -axo tty: Controlling tty (Terminal). (Alias tname, tt).

It includes below:

TT

14. ps -axo tpgid: ID of the foreground process group on the tty (terminal) that the process is connected to, or -1 if the process is not connected to a tty.

It includes below:

TPGID

15. ps -axo sess or SESS: Session ID or, equivalently, the process ID of the session leader. (Alias session, sid).

It includes below:

SESS

16. **ps -axo pgrp:** Process group ID or, equivalently, the process ID of the process group leader.

It includes below:

PGRP

17. **ps -axo ppid:** Parent process ID.

It includes below:

PPID

18. **ps -axo pid:** A number representing the process ID. (Alias tgid). **It includes below:**

PID

19. **ps -axo pcpu:** It display %CPU.

It includes below:

%CPU

20. **ps -Ao fname:** First 8 bytes of the base name of the process's executable file.
The output in this column may contain spaces.

-A: To print all running processes in system.

-o: The "o" option allows you to specify which columns are displayed when running the ps command.

It includes below:

COMMAND

21. ps -Ao tmout:

-A: To print all running processes in system.

-o: The “o” option allows you to specify which columns are displayed when running the ps command.

It includes below:

TMOUT

22. ps -Ao f: Flags associated with the process.

-A: To print all running processes in system**.**

-o: The “o” option allows you to specify which columns are displayed when running the ps command.

It includes below:

F

H] To print only the process IDs of syslog:

1. ps -C syslogd -o pid = 1

Output: 1

I] To print only the name of PID 42:

ps -q 42 -o comm= Output: COMMAND.

Important Note:

Flag value:

1=> The process forked but didn’t execute.

2=> It is used super user privileged.

Nice value:

It holds value from -20 to 19. Less nice value that much high priority.

Priorities are a below:

NI	Priority
-20	0
-19	1
-18	2
19	39

J] Top Command:

TOP will display a full screen of information about the processes running on the system, as well as some overall information about the system.

This includes load average, number of processes, the CPU status, free memory information, and details about processes including PID, user, priority, CPU and memory usage information, running time, and program name. $ > top: It will display the complete active process, CPU and memory information.

SHIFT + M: List process according to memory usage.

SHIFT + P: List the process according to CPU usage.

Z (Press): List running process in colour.

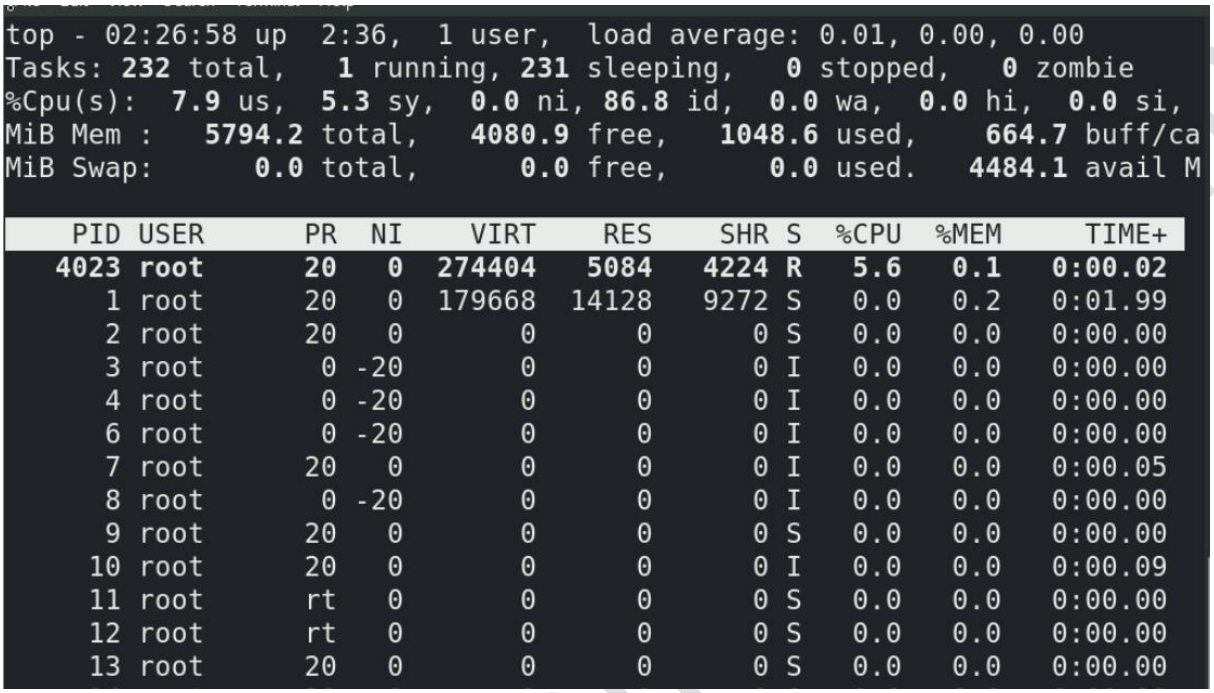
C (press): List running process with absolute path.

K <pid>: For killing the process.

SHIFT + W: For saving the top output in a file (/root/. toprc.)
$>topu: Research – wing.

The -u option allows to specify a username or UID and monitor only those processes owned by that UID.

%>q: Quit.



Jobs and Sessions:

Job control is a feature of the shell which allows a single shell instance to run and manage multiple commands.

A job associated with each pipeline entered at a shell prompt.

All processes in that pipeline are part of the job and are members of the same process group. (If only one command is entered at a shell prompt, that can be considered to be a minimal “pipeline” of one command. That command would be the only member of that job).

Only one job can read input and keyboard – generated signals from a particular terminal, window at a time.

A job is a part of exactly one session, the one belonging to its controlling terminal.

Processes that are part of that job are foreground and background.

1. **Foreground process:** Refer to applications you are running that you are currently interacting with, and which applies equally to graphical user interfaces as it does to the command line.

2. **Background process:** Refer to applications that are running but not being interacted with by the user.

Background processes of a terminal cannot read input or receive keyboard – generated interrupts from the terminal, but may be able to write to the terminal.

A job in the background may be stopped (suspended) or it may be running, if a running background job tries to read from the terminal, it will be automatically suspended.

Note:

bg <n>: command is used sending a process to background. fg <n>: command is used bringing a process to foreground.

Commands are as below:

- **jobs:** Command is used viewing current jobs + indicates that fg will foreground this specific process.

- **command &:** To run a command in the background, add the ampersand symbol (&) at the end of the command.

Output: [1] 2950

Whereas,

[1]- The shell job ID (surrounded with brackets) and 2950 – Process ID.

•The bash shell tracks jobs, per session, in a table displayed with the jobs command.

➔jobs

•A background job can be brought to the foreground by using the fg command with its job ID (%job number).

➔fg %1

•To send a foreground process to the background, first press the keyboard-generated suspend request (Ctrl+z) on the terminal.

➔sleep 1000

➔Press Ctrl+z

➔[1] + Stopped sleep 10000

•**ps -j:** Command will display information relating to jobs. Whereas, SID: Session leader.

•To start the suspended process running in the background, use the bg command with the same job ID.

➔bg %1

Killing Processes:

Process control using signals:

A signal is a software interrupt delivered to a process.

Signals report events to an executing program.

Events that generate a signal can be an error, external event, (e.g., I/O request or expired timer) or explicit request (e.g., use of a signal-sending command or by keyboard sequence).

SIGNAL NUMBER	SHORT NAME	DEFINITION	PURPOSE
1	HUP	Hangup	Used to report termination of the controlling process of a terminal. Also used to request process reinitialization (configuration reload) without termination.
2	INT	Keyboard interrupt	Causes program termination. Can be blocked or handled. Sent by pressing INTR key sequence (**Ctrl+c**).
3	QUIT	Keyboard quit	Similar to SIGINT; adds a process dump at termination. Sent by pressing QUIT key sequence (**Ctrl+**).
9	KILL	Kill, unblockable	Causes abrupt program termination. Cannot be blocked, ignored, or handled; always fatal.
15 default	TERM	Terminate	Causes program termination. Unlike SIGKILL, can be blocked, ignored, or handled. The "polite" way to ask a program to terminate; allows self-cleanup.
18	CONT	Continue	Sent to a process to resume, if stopped. Cannot be blocked. Even if handled, always resumes the process.
19	STOP	Stop, unblockable	Suspends the process. Cannot be blocked or handled.
20	TSTP	Keyboard stop	Unlike SIGSTOP, can be blocked, ignored, or handled. Sent by pressing SUSP key sequence (**Ctrl+z**).

Each signal has a default action, usually one of the following:

Term: Cause a program to terminate (exit) at once.

Core: Cause a program to save a memory image (core dump), then terminate.

Stop: Cause a program to stop executing (suspend) and wait to continue (resume).

Commands for sending signals by explicit request:

Ctrl + z: To send a foreground process to the background, first press the keyboard-generated suspend request (Ctrl+z) on the terminal.

Ctrl + c: Kill the process.

Ctrl + \: Core dump the process.

Signals can be specified either by name (e.g. -HUP or -SIGHUP) or by number (e.g., -1).

Users may kill their own processes, but root privilege is required to kill processes owned by others.

The kill command sends a signal to a process by ID.

killall: To send a signal to one or more processes matching selection criteria, such as a command name, processes owned by a specific user, or all system-wide processes. Signals a multiple process.

pkill: Command uses advanced selection criteria, which can include combination of:
Command: Processes with a pattern-matched command name.
UID: Processes owned by a Linux user account, effective or real.
GID: Processes owned by a Linux group account, effective or real.
Parent: Child processes of a specific parent process.

Terminal: Processes running on a specific controlling terminal.

To terminate all processes for one user, use the pkill command.

Logging users out administratively:

w: Command views users currently logged into the system and their cumulative activities. Use the TTY and FROM columns to determine the user's location.

Output:

```
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
student  :0       :0               08:36   ?xdm?   1:45   0.80s /usr/libexec/gnome-session-binary --session gnome-classic
student  pts/0    :0               08:37    3.00s  0.38s  0.16s w
[student@rhel7 ~]$
```

Whereas,

JCPU column: consist of current jobs, including background tasks and child processes.

PCPU column: consist of current foreground process.

-f: Remote user's display their connecting system name in the FROM column when using the -f option.

ptsree: Command to view a process tree for the system or a single user.

Output:

```
systemd─┬─ModemManager───2*[{ModemManager}]
        ├─NetworkManager─┬─dhclient
        │                └─2*[{NetworkManager}]
        ├─VGAuthService
        ├─2*[abrt-watch-log]
        ├─abrtd
        ├─accounts-daemon───2*[{accounts-daemon}]
        ├─alsactl
        ├─at-spi-bus-laun─┬─dbus-daemon───{dbus-daemon}
        │                 └─3*[{at-spi-bus-laun}]
        ├─at-spi2-registr───2*[{at-spi2-registr}]
        ├─atd
        ├─auditd─┬─audispd─┬─sedispatch
        │        │         └─{audispd}
        │        └─{auditd}
        ├─avahi-daemon───avahi-daemon
        ├─colord───2*[{colord}]
        ├─crond
        ├─cupsd
        ├─2*[dbus-daemon───{dbus-daemon}]
        ├─dbus-launch
```

Monitoring process activity:

The Linux kernel calculates a load average metric as an exponential moving average of the load number, a cumulative CPU count of active system resource requests.

•Active requests: are counted from per-CPU queues for running threads and threads waiting for I/O, as the kernel t racks process resource activity and corresponding process state changes.

•Load number: is a calculation routine run every five seconds by default which accumulates and averages the active requests into a single number for all CPUs.

•Exponential moving average: is a mathematical formula to smooth out trending data highs and lows, increase current activity significance, and decrease aging data quality.

•Load average: is the load number calculation routine res u lt.

Collectively, it refers to the three displayed values of system activity data averaged for the last 1, 5, and 15 minutes.

Understanding the Linux load average calculation:

Linux implements the load average calculation as a representation of expected service wait times, not only for CPU but a l so for disk and network I/O.

•Linux counts not only processes, but threads individually, as separate tasks. CPU request queues for running threads (nr_running) and threads waiting for I/O resources (nr_iowait) reasonably correspond to process states R (Running) and D (Uninterruptable Sleeping). Waiting for I/O includes tasks sleeping for expected disk and network responses.

•The load number is a g lobal counter calculation, which is sum-totaled for all CPUs. Since tasks returning from sleep may

reschedule to different CPUs, accurate per-CPU counts a re-difficult. but an accurate cumulative count is assured. Displayed load averages represent all CPUs.

•Linux counts each physical CPU core and microprocessor hyper-thread as separate execution units, logically represented and referred to as individual C PUs. Each CPU has independent request queues. View /proc/cpuinfo for the kernel representation of system CPUs.

Real – Time Process Monitoring:

The top program is a dynamic view of the system's processes, displaying a summary header followed by a process or thread list similar to ps information.

Unlike the static ps output. top continuously refreshes at a configurable interval, and provides capabilities for column reordering, sorting, and highlighting.

User configurations can be saved and made persistent.

Default output columns are recognizable from other resource tools:

➢ The process ID (PID).

➢ User name (USER) is the process owner.

➢ Virtual memory (VIRT) is all memory the process is using, including the resident set. shared libraries, and any mapped or swapped memory pages. (Labeled VSZ in the ps command.)

➢ Resident memory (RES) is the physical memory used by the process, including any resident sha red objects. (Labeled RSS in the ps command.)

➢ Process state (S) displays as:

D = Uninterruptable Sleeping

R = Running or Runnable · S = Sleeping

T = Stopped or Traced

Z = Zombie

➢CPU time (TIME) is the total processing time since the process started. May be tog g l ed to include cumulative time of all previous children. s

➢The process command name (COMMAND).

Fundamental keystrokes in top:

Key	Purpose
? *or* **h**	Help for interactive keystrokes.
l, **t**, **m**	Toggles for load, threads, and memory header lines.
1	Toggle showing individual CPUs or a summary for all CPUs in header.
s (1)	Change the refresh (screen) rate, in decimal seconds (e.g., 0.5, 1, 5).
b	Toggle reverse highlighting for *Running* processes; default is bold only.
B	Enables use of bold in display, in the header, and for *Running* processes.
H	Toggle threads; show process summary or individual threads.
u, **U**	Filter for any user name (effective, real).
M	Sorts process listing by memory usage, in descending order.
P	Sorts process listing by processor utilization, in descending order.
k (1)	Kill a process. When prompted, enter **PID**, then **signal**.
r (1)	Renice a process. When prompted, enter **PID**, then **nice_value**.
W	Write (save) the current display configuration for use at the next **top** restart.
q	Quit.
Note:	(1) Not available if top started in secure mode. See **top**(1).

Practical:

1. In a terminal window, run the top utility. Size the window to be as tall as possible. **Output:**

```
[student@serverX ~]$ top
top - 12:47:46 up  2:02,  3 users,  load average: 1.67, 1.25, 0.73
Tasks: 361 total,   6 running, 355 sleeping,   0 stopped,   0 zombie
%Cpu(s): 98.5 us,  1.4 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.1 si,  0.0 st
KiB Mem:   2043424 total,   897112 used,  1146312 free,     1740 buffers
KiB Swap:  4079612 total,        0 used,  4079612 free.   296276 cached Me

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
 4019 root      20   0    4156     76      0 R 57.5  0.0   2:54.15 hippo
 2492 student   20   0 1359500 168420  37492 S 16.8  8.2   3:55.58 gnome-shell
 1938 root      20   0  189648  35972   7568 R  1.9  1.8   0:29.66 Xorg
 2761 student   20   0  620192  19688  12296 S  0.4  1.0   0:04.48 gnome-termi+
output truncated
```

2. Observe the top display. The default display sorts by CPU utilization, highest first. What are the processes using the most CPU time? In addition to the default GNOME shell, find the process named hippo.

3. Change the display to sort by the amount of memory in use by each process. Press M. Output:

```
top - 12:57:38 up  2:11,  3 users,  load average: 2.09, 1.70, 1.19
Tasks: 360 total,   5 running, 355 sleeping,   0 stopped,   0 zombie
%Cpu(s): 99.8 us,  0.2 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   2043424 total,   896952 used,  1146472 free,     1740 buffers
KiB Swap:  4079612 total,        0 used,  4079612 free.   296280 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
 2492 student   20   0 1359500 168420  37492 S  0.5  8.2   4:01.04 gnome-shell
 4013 root      20   0   55360  51208    152 S  0.0  2.5   0:00.43 elephant
 1938 root      20   0  189648  35972   7568 R  0.2  1.8   0:30.49 Xorg
 2576 student   20   0  533752  33684  27784 S  0.1  1.6   0:09.29 vmtoolsd
 2420 student   20   0  916268  25616  14404 S  0.0  1.3   0:00.61 gnome-setti+
 2550 student   20   0 1048204  23136  16060 S  0.0  1.1   0:00.46 nautilus
output truncated
```

4. What are the processes with the largest memory allocations? In addition to the default GNOME shell and **Xorg**, find a process named elephant.

5. Turn off the use of bold in the display. Save this configuration for reuse when top is restarted.
   Press the single uppercase keystroke B to toggle bold use off. Press the single uppercase keystroke W to save this configuration. The default config u ration file is. toprc in the current user's home directory.

6. Exit top, then restart it again. Confirm that the new display uses the saved configuration; i.e., the display starts sorted by memory utilization and bold is turned off.
   Press q to quit the current display, then run top again. Output:

```
[student@serverX ~]$ top
```

7. Modify the display to again sort by CPU utilization. Turn on the use of bold. Observe that only Running or Runnable (state R) process entries are bold. Save this configuration.
   Press the sing le uppercase keystroke P to sort by CPU utilization.

   Press the single uppercase keystroke B to toggle bold use on.

   Press the single uppercase keystroke W to save this configuration.

8. Open a not her terminal window if necessary. As root, suspend the hippo process. In top, observe that the process state is now T. Output:

```
[student@serverX ~]$ su -
Password: redhat
[root@serverX ~]# pkill -SIGSTOP hippo
```

9. The hippo process quickly disappears from the display, since it is no longer actively using CPU resources. List the process information from the command line to confirm the process state. Output:

```
[root@serverX ~]# ps -f $(pgrep hippo)
```

10. Resume execution of the hippo processes. Output:

```
[root@serverX ~]# pkill -SIGCONT hippo
```

11. When finished observing the display, terminate the extra processes using the command line. Confirm that the processes no longer display in top. Output:

```
[root@serverX ~]# pkill elephant
[root@serverX ~]# pkill hippo
```

12. Check that the clean-up is successful l by running the grading script. If necessary, find and terminate processes listed by the grading script, and repeat grading. Output:

```
[root@serverX ~]# lab processes grade
```

13. Exit the top display. Close extra terminal windows. Press q to quit.

END