Red Hat System Administration I

UNIT 7

Monitoring and Managing Linux Processes

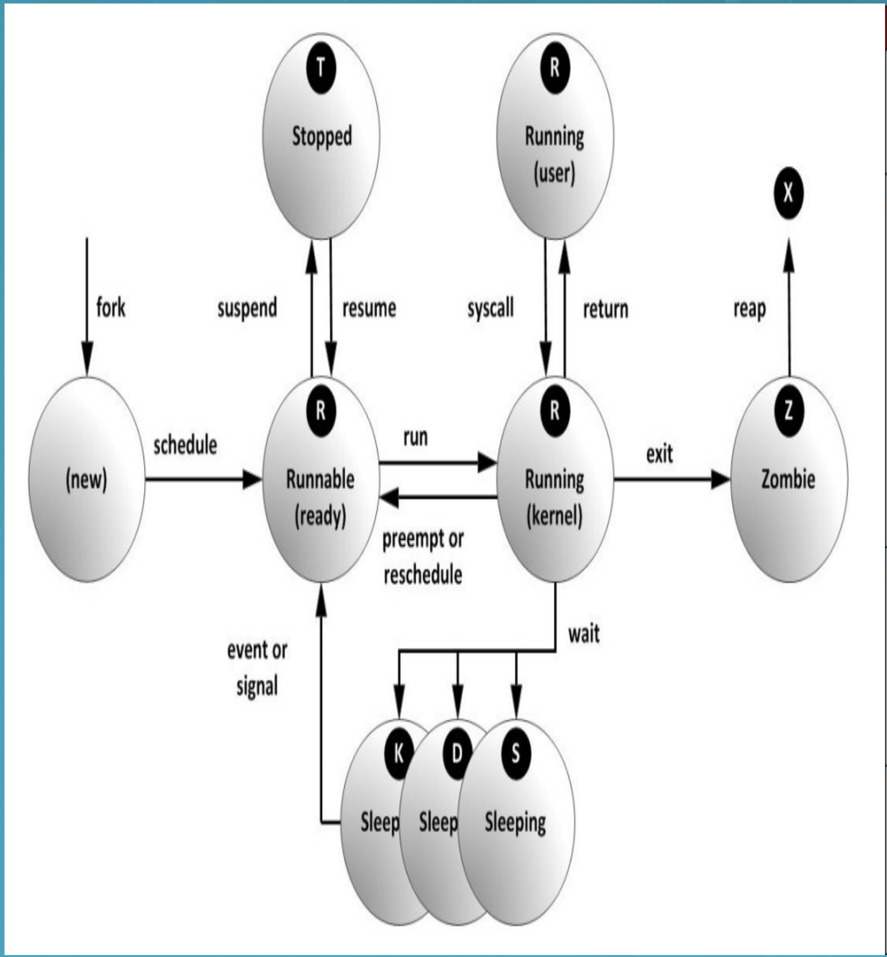
Objectives

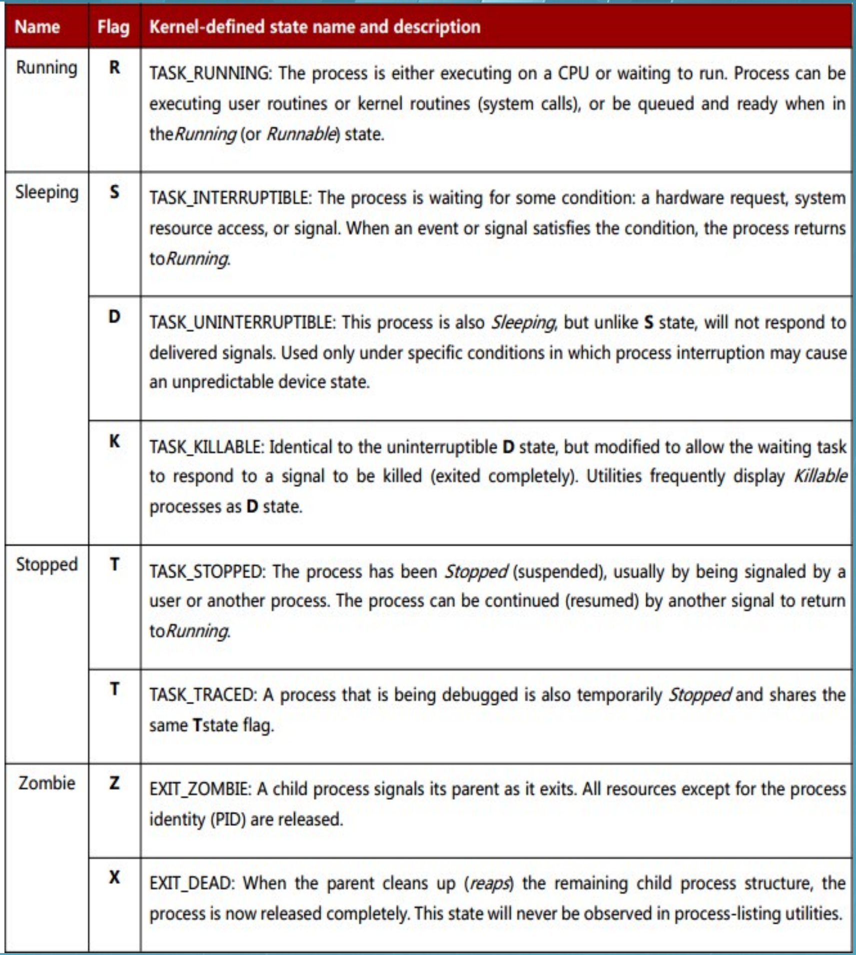
- List and interpret basic information about processes running on the syste
- Control processes in the shell's session using bash job control.
- Terminate and control processes using signals.
- Monitor resource usage and system load due to process activity.

What is a process?

- A processis a running instance of a launched, executable program. A process 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 state.

Process states





Name	Flag	Kernel-defined state name and description
Running	R	TASK_RUNNING: The process is either executing on a CPU or waiting to 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	TASK_INTERRUPTIBLE: 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	TASK_UNINTERRUPTIBLE: 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 unpredictable device state.
	K	TASK_KILLABLE: 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	TASK_STOPPED: The process has been *Stopped* (suspended), usually by being signaled by a user or another process. The process can be continued (resumed) by another signal to return to*Running*.
	T	TASK_TRACED: A process that is being debugged is also temporarily *Stopped* and shares the same **T**state flag.
Zombie	Z	EXIT_ZOMBIE: A child process signals its parent as it exits. All resources except for the process identity (PID) are released.
	X	EXIT_DEAD: 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.

Listing processes

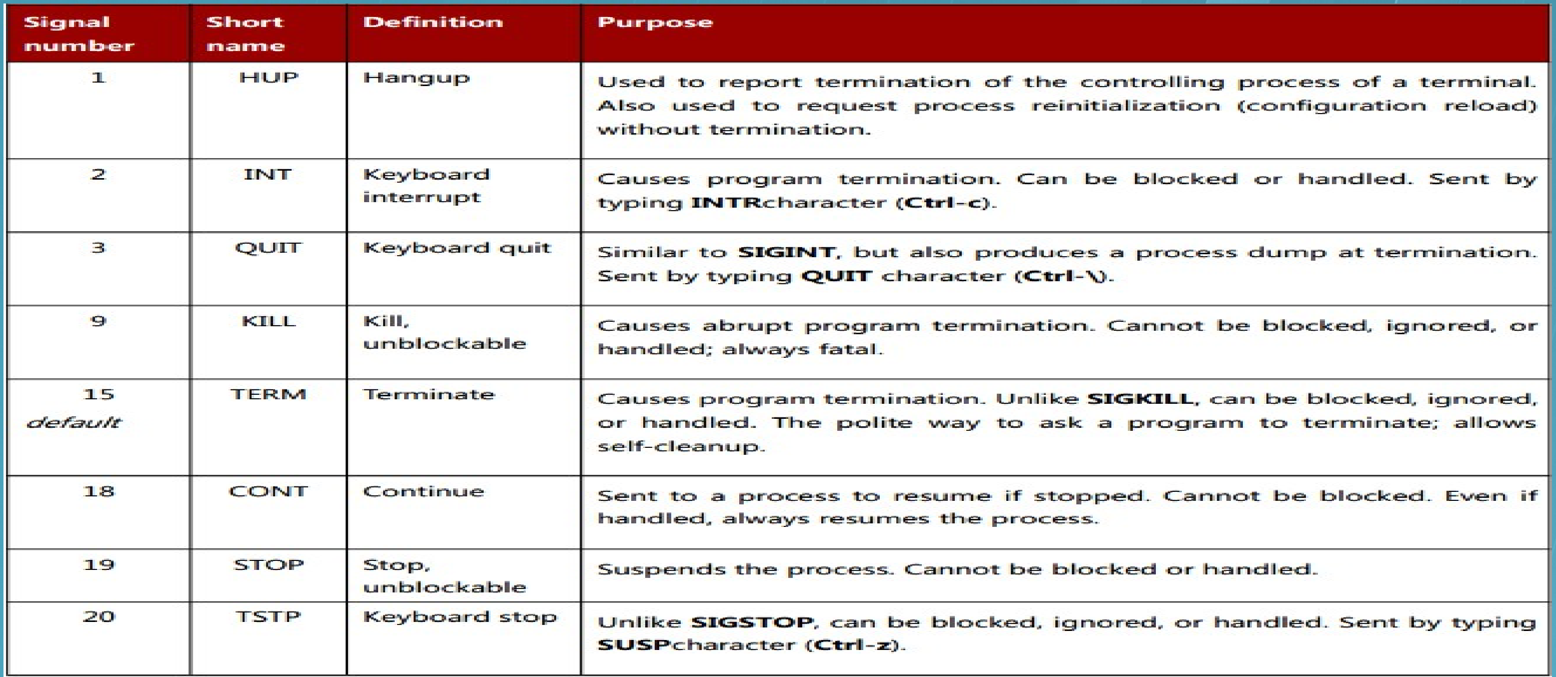
- ps
  - -a
  - -l
  - -e
  - -f
  - -u
  - -x

Controlling Jobs

- bg
- fg
- jobs
- ctrl +z
- &

Killing Processes

- **kill signal pid and killall signal command**



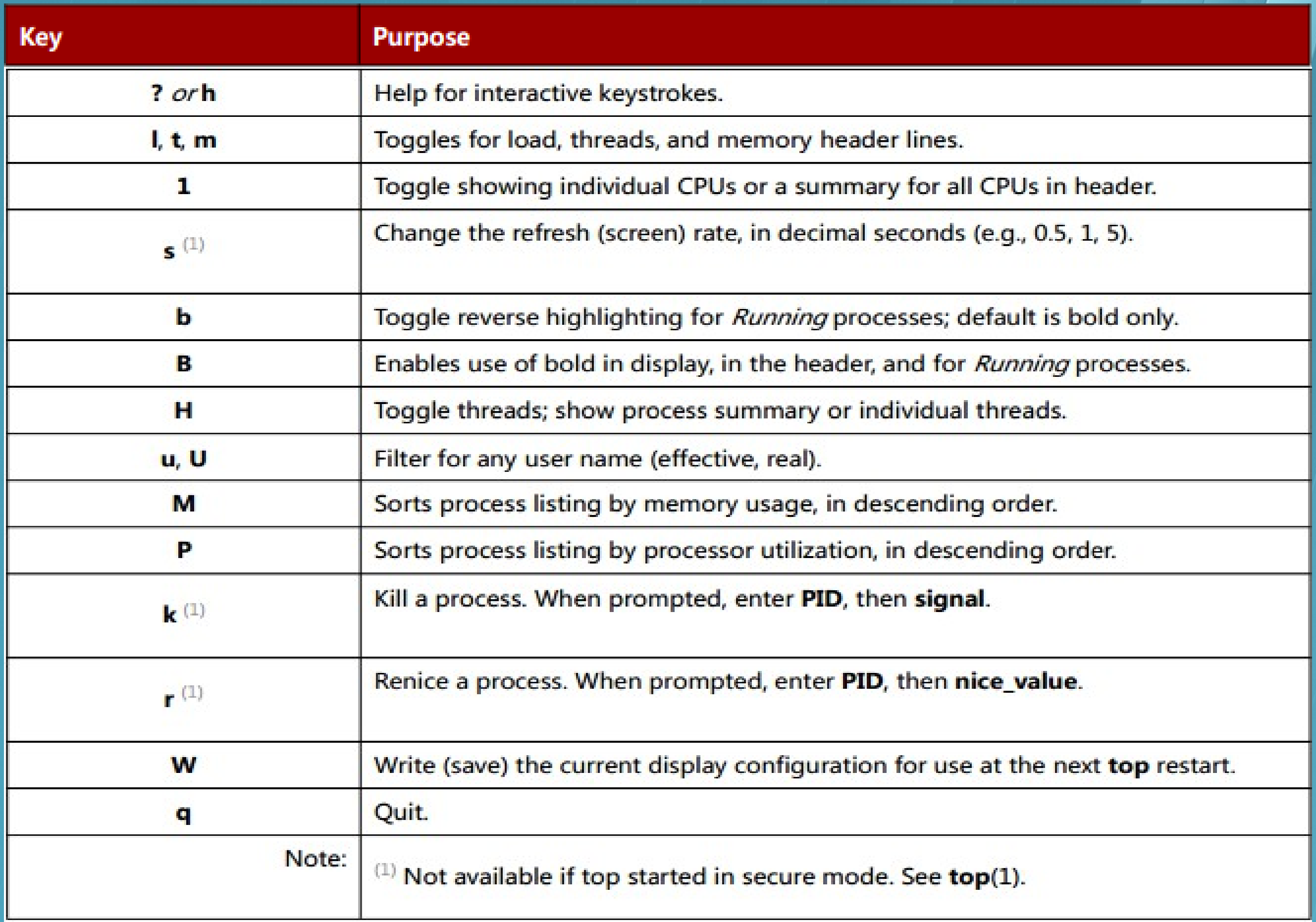
Signal number	Short name	Definition	Purpose
1	HUP	Hangup	Used to report termination of the controlling process of a terminal. Also used to request process reinitialization (configuration reload) without termination.
2	INT	Keyboard interrupt	Causes program termination. Can be blocked or handled. Sent by typing **INTR**character (**Ctrl-c**).
3	QUIT	Keyboard quit	Similar to **SIGINT**, but also produces a process dump at termination. Sent by typing **QUIT** character (**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 typing **SUSP**character (**Ctrl-z**).

Logging users out administratively

- w
- last
- lastb

Real-time process monitoring

- 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).

lab one

<lab 1>
In a terminal window, run the top utility. Size the window to be as tall as possible.
<lab 2>
Observe the top display. The default display sorts by CPU utilization, highest first. What are the processes using the most CPU time?
<lab 3>
Change the display to sort by the amount of memory in use by each process.
<lab 4>
What are the processes with the largest memory allocations?
<lab 5>
Turn off the use of bold in the display. Save this configuration for reuse when top is restarted.
<lab 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.

lab two

<lab 7>

Modify the display to again sort by CPU utilization. Turn on the use of bold. Observe that only RunningorRunnable(state R) process entries are bold. Save this configuration.

<lab 8>

Open another terminal window if necessary. As root, suspend the hippo process. In top, observe that the process state is now T

<lab 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.

<lab 10>

Resume execution of the hippo processes.

<lab 11>

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

<lab 12>

Check that the cleanup is successful by running the grading script. If necessary, find and terminate processes listed by the grading script, and repeat grading.

That's all