Chapter 12 Process Monitoring - Notes

12.2 Introduction

Keeping track running (and sleeping) processes -> essential system administration task. **ps** program -> main tool for doing so in UNIX-based operating systems for decades.

HOw ever, because utility has long + complicated history of being used differently in more than one operating system variety, has large assortment of options that can be applied w ith often confusing combinations. Another trust tool provided by **top**, w hich interactively monitors the system's state.

12.3 Learning Objectives:

- Use **ps** to view characteristics and statistics associated w ith processes.
- Identify different **ps** output fields and customize the **ps** output.
- Use **pstree** to get a visual description of the process ancestry and multi-threaded applications.
- Use **top** to view system loads interactively.

12.4 Monitoring Tools

In this section, w ill concentrate on process monitoring. Linux administrators make use of many utilities for process monitoring, eg. **ps**, **pstree**, **top**. All have long histories in UNIX-like operating systems.

Review of some of the main tools for process monitoring:

Process and Load Monitoring Utilities

Utility	Purpose	Package
top	Process activity, dynamically updated	**procps**
uptime	How long system is running and average load	**procps**
ps	Detailed information about processes	**procps**
pstree	Tree of processes and their connections	**psmisc** (or **pstree**)
mpstat	Multiple processor usage	**sysstat**
iostat	CPU utilization and I/O statistics	**sysstat**
sar	Display and collect information about system activity	**sysstat**
numstat	Information about **NUMA** (Non-Uniform Memory Architecture)	**numactl**
strace	Information about all system calls a process makes	**strace**

12.5 Viewing Process States with ps

ps -> workhorse for displaying characteristics and statistics associated with proceeses, all of which garnered from `/proc` directory associated with process.

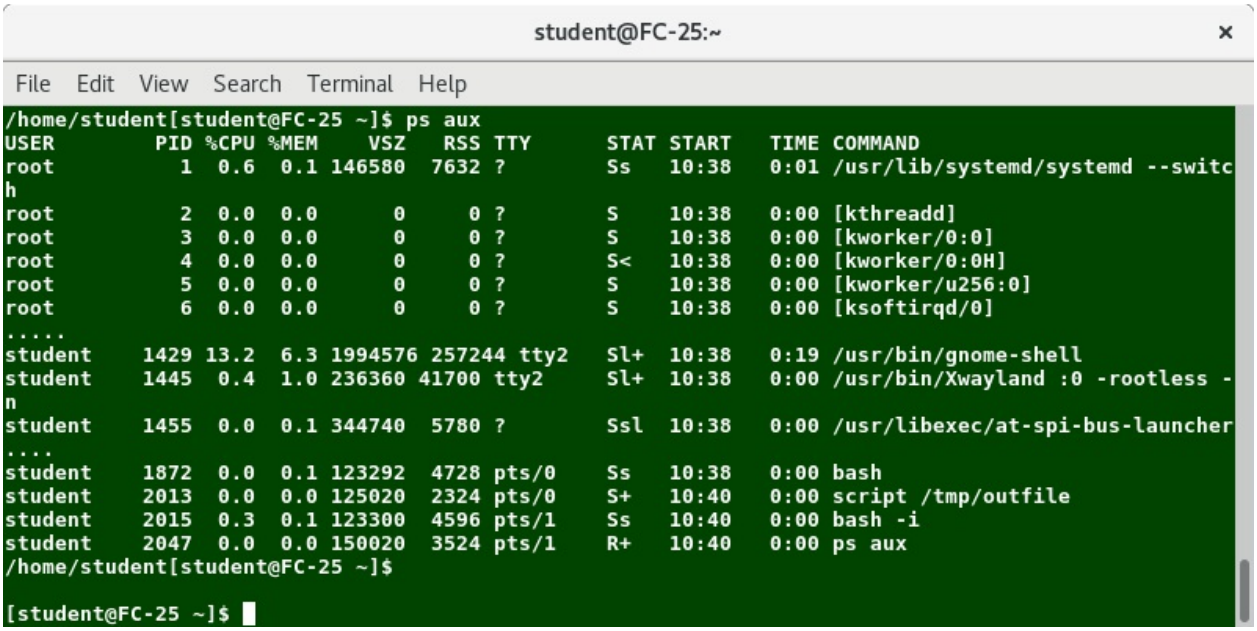
Command utility existed in all UNIX-like operating system variant. Diversity reflected in complicated potpourri of options that Linux version of **ps** accepts. Falls into three categories:

1. UNIX options, which must be preceded by -, and what may be grouped.
2. BSD options, which must *not* be preceded by -, and which may be grouped.
3. GNU long options, each of which must be preceded by --.

Having all these possible options can make life rather confusing. Most system administrators tend to use one or two standard combinations for daily use.

12.6 BSD Option Format for ps

Can see typical usage with BSD option format below, where `aux` option shows all processes. Commands surrounded by square brackets (as in `[ksoftirqd/0]`) -> threads that exist totally within kernel. If there is one for each CPU, command followed by `integer` specifying CPU it is running on.



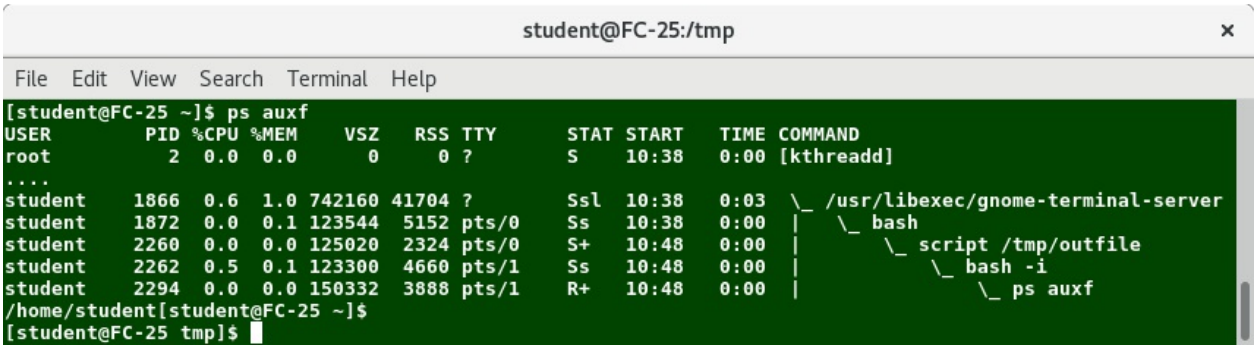
12.7 ps Output Fields

Most fields in preceding example self-explanatory. Of others:

- `VSZ`: process' virtual memory size in KB
- `RSS`: resident set size; non-swapped physical memory task is using in KB.
- `STAT`: describes state of process. in example, only see **S** for sleeping, or **R** for running. Additional character in state (where it exists):
  - `<`: high priority (not *nice*)
  - `N`: low priority (*nice*)
  - `L`: having pages locked in memory
  - `s`: session leader
  - `l`: multi-threaded
  - `+`: being in foreground process group

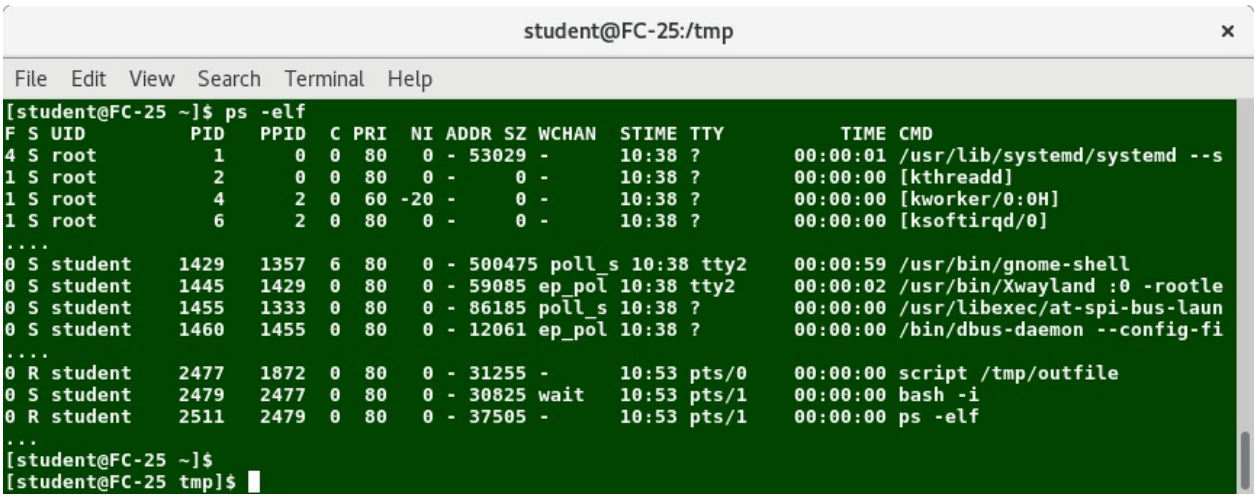
Adding `f` option will show how processes connect by ancestry:

```
$ ps auxf
```



12.8 UNIX Option Format for ps

Can see typical usage with UNIX option format below. Note: now showing **Parent Process ID** (`PPID`) and the niceness (`NI`). May observe that many processes show `PPID=2`, in this screenshot (taken from RHEL 7, using systemd) an internal kernel process `kthreadd` (designed to adopt children when parent process dies). In older kernels/systems, would see `PPID=1` for **sbin/init**, but really same thing going on.



Some common selection options in UNIX format:

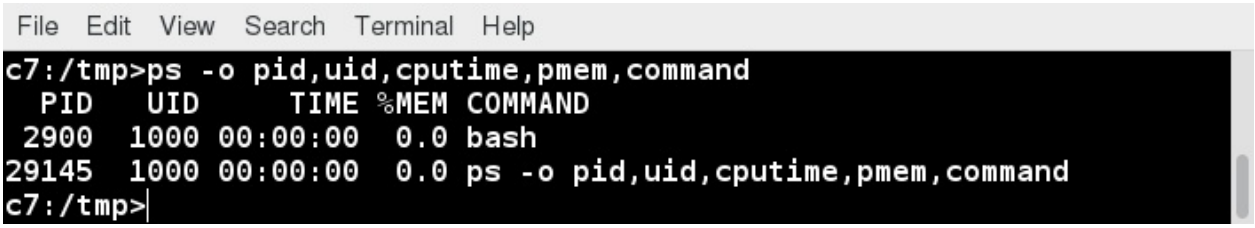
- `-A` or `-e`: Select all processes
- `N`: Negate selection (means do the opposite)
- `C`: Select by command name
- `G`: Select by real group ID (also supports names)
- `U`: Select by real user ID (also supports names)

12.9 Customizing the ps Output

If you use `-o` option, followed by comma-separated list of field identifiers, can print out customized list of **ps** fields:

- `pid`: Process ID number
- `uid`: User ID number
- `cmd`: Command with all arguments
- `cputime`: Cumulative CPU time
- `pmem`: Ratio of process's resident set size to physical memory on machine, expressed as percentage

Can see example below. Can consult **ps man** page for many other output options.



12.10 Using pstree

pstree gives visual description of process ancestry and multi-threaded applications:

```
$ pstree -aAp 2408

bash,2408
|-emacs,24998 pmonitor.tex
|  |-{emacs},25002
|  '-{emacs},25002
|-evince,18036 LFS201-SLIDES.pdf
|  |-{evince},18040
|  |-{evince},18046
|  '-{evince},18047
```

Consult **man** page for **pstree** for explanation of many options. In above, have chosen just to show information for `pid-2408`.

Note: one of its child processes (**evince**, `pid=18036`) has three children of its own. Another way to see that:

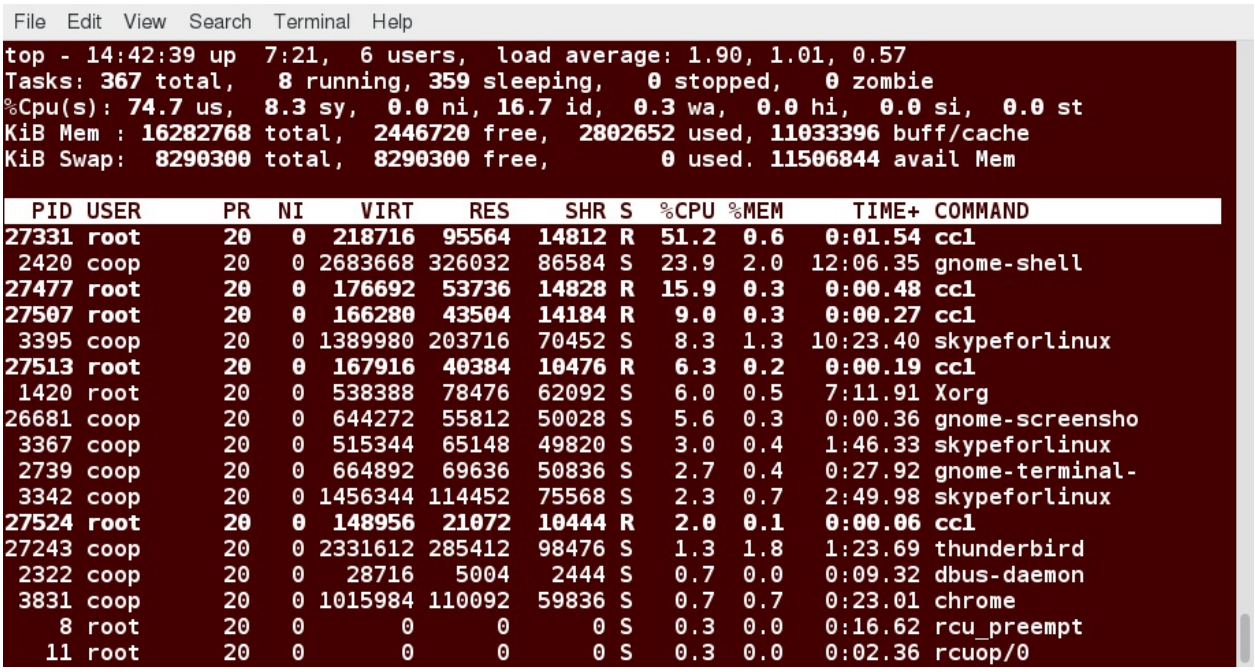
```
$ ls -l /proc/18036/task

total 0
dr-xr-xr-x 5 coop coop 0 Sep 11 07:15 18036
dr-xr-xr-x 5 coop coop 0 Sep 11 07:15 18040
dr-xr-xr-x 5 coop coop 0 Sep 11 07:15 18046
dr-xr-xr-x 5 coop coop 0 Sep 11 07:15 18047
```

12.11 Viewing System Loads with top

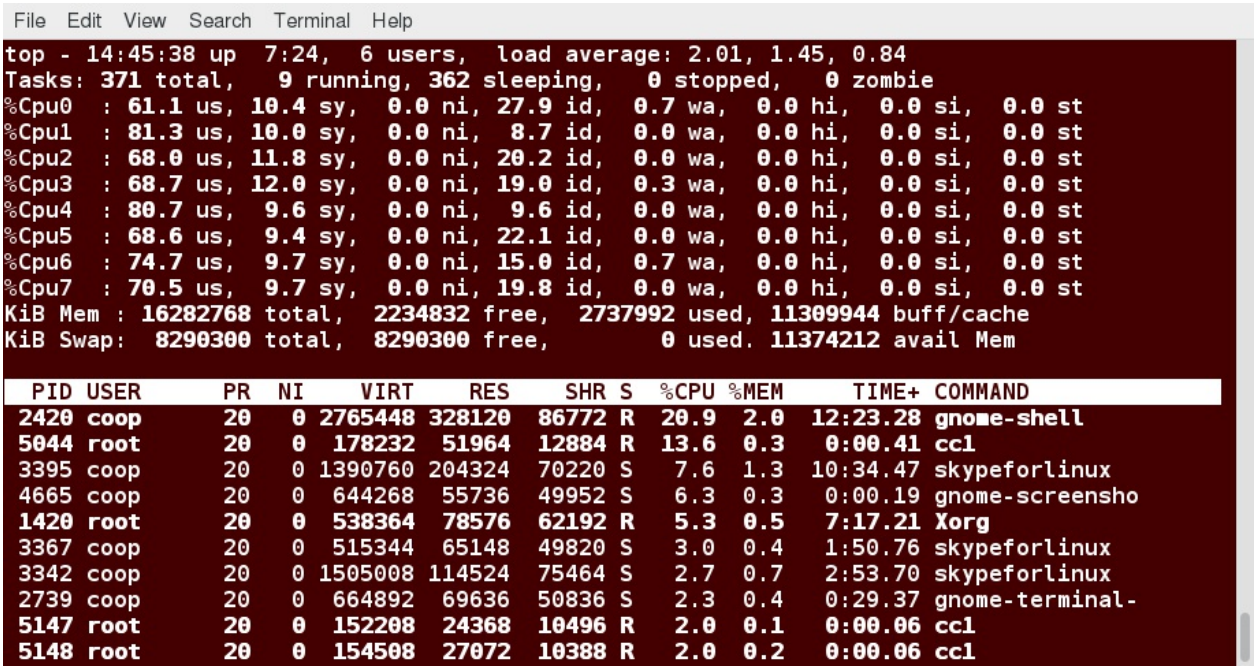
When want to know what system spending time on, first tool often used is **top**. Below shows what can be seen when using **top** without arguments.

BY default, **top** refreshes every 3.0 seconds.



12.12 top Options

top: ancient utility and has tons of options, as w ell as interactive commands triggered w hen certain keys pressed. Eg. if hit `1`, each CPU show n separately, if hit `i` only active processes show n. Can see w hat doing both gives you below.



Have lot of control over how processes sorted, w hich fields displayed. Many others besides defaults. Eg. hitting `h` or `?` gives brief list of interactive commands, `q` quits.

Can **kill** task by hitting `k`, or **renice** (change its priority) w ith `r`.

man top w ill give extensive documentation on configuration possibilities, options, interactive possibilities.

Note: there are popular alternatives to standard **top** program. Some have more visual interfaces and/or additional information, such as **htop**, **ntop**, **atop**. Most Linux distributions have graphical system monitor (eg. **gnome-system-monitor** or

ksysguard), which has **top**-like display window that can be shown.

##

Back to top

Previous Chapter - Table of Contents - Next Chapter