Assignment for Laboratory 12
Processes
Computer Science 240

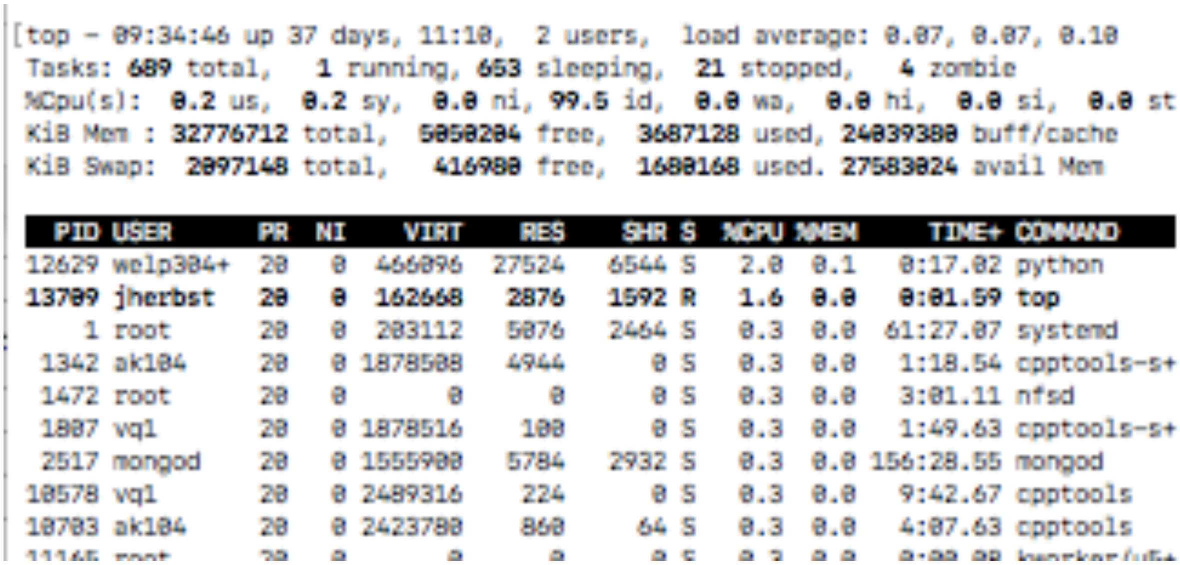
Explore Processes in a Running System

In the terminal, enter the commands shown at the command line.

Run the top command to visualize the currently executing processes and the resources they consume:

```
$ top
```

Examine the output. which looks something like this.



```
[top - 09:34:46 up 37 days, 11:10,  2 users,  load average: 0.07, 0.07, 0.10
 Tasks: 689 total,   1 running, 653 sleeping,  21 stopped,   4 zombie
 %Cpu(s):  0.2 us,  0.2 sy,  0.0 ni, 99.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
 KiB Mem : 32776712 total,  5050204 free,  3687128 used, 24039380 buff/cache
 KiB Swap:  2097148 total,   416980 free,  1680168 used. 27583024 avail Mem

   PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 12629 welp304+  20   0  466096  27524   6544 S   2.0  0.1   0:17.02 python
 13709 jherbst   20   0  162668   2876   1592 R   1.6  0.0   0:01.59 top
     1 root      20   0  203112   5076   2464 S   0.3  0.0  61:27.07 systemd
  1342 ak104     20   0 1878508   4944      0 S   0.3  0.0   1:18.54 cpptools-s+
  1472 root      20   0       0      0      0 S   0.3  0.0   3:01.11 nfsd
  1807 vq1       20   0 1878516    100      0 S   0.3  0.0   1:49.63 cpptools-s+
  2517 mongod    20   0 1555900   5784   2932 S   0.3  0.0 156:28.55 mongod
 10578 vq1       20   0 2489316    224      0 S   0.3  0.0   9:42.67 cpptools
 10703 ak104     20   0 2423780    860     64 S   0.3  0.0   4:07.63 cpptools
 11146 root      20   0       0      0      0 S   0.3  0.0   0:00.00 kworker/u6+
```

This is all the processes running on the server, with the status of each being updated periodically.

Notice that many different users are active, and also notice that your own process, running **top**, is shown.

Read the **Tasks:** line at the beginning of the output, interpreting “Tasks” as “processes”.

1.How many processes are running? Sleeping? Stopped? Zombie?

Read the **%CPU(s)**: line, which shows the percentage of the time that the CPU is spending executing user and operating system kernel code, vs. being idle (and a few other categories we will ignore). These levels probably fluctuate at each sample that *top* displays.

Also displayed are a list of processes ranked by the percentage of CPU time they have used in the most recent time window.

2. Which processes are using the most CPU time? About how much?

Enter <CtrL> C to terminate the **top** command.

Run the **ps** command:

```
$ ps
```

By default, it lists only the processes run under your current login session. (Each terminal window you open actually creates a new login session and runs a shell in it.)

You should see something like this:

```
23314 pts/1     00:00:00 bash
30086 pts/1     00:00:00 ps
```

Run **ps ux** to see the list of all processes belonging to you:

```
$ ps ux
```

3.How many have used at least 1 second of CPU time? (see the TIME column, in minutes:seconds form)

Run **ps aux** to see the list of all processes run by all users on this machine:

```
$ ps aux
```

List the contents of the /proc filesystem:

```
$ ls /proc
```

NOTE: The /proc filesystem is provided by the Linux kernel as an interface to inspect information about process scheduling, individual processes, and other operating system status information

You will see something like this, which is a list of subdirectories:

```
1       12327  171    217    248    29     315    3518    47     8512
10      12328  172    218    24801  290    3152   3519   48     86
100     124    173    219    249    29040  316    352    483    87
101     125    174    22     24946  291    31654  3520   49     88
```

The /proc filesystem has a subdirectory with information about each living process. Each directory is named with the associated PID (Proccess ID) of a process that is currently running.

Examine the interrupts file:

```
$ cat   /proc/interrupts
```

A column is listed for each of the CPUs in the server.

4. How many CPUS's are there?

5.How many interrupts have occurred for scheduling (context-switching)? How many for system calls (traps, labelled "Function Call")?

Find the PID (Process ID) of *python* by using *top*:

```
$ top
```

and finding *python* and its listed PID (there may be more than one occurrence, just choose the first one)..

Change into that directory (for example, if the PID of *python* is xxxx):

$ **cd /proc/xxxx** (you must replace the **xxxx** with the PID for a python process)

Inspect its status information by showing the contents of the *status* file:

```
$ cat status
```

6. How many context switches has *python* experienced? (Look for voluntary and nonvoluntary ctxt switches**)**

7. How many child processes has *python* created? (See the "task" subdirectory or run the pstree -p command to see the hierarchy of process ancestry.)