information technology & management

viabiliiiit;

INTRO TO OPEN SOURCE OPERATING SYSTEMS

ILLINOIS INSTITUTE OF TECHNOLOGY

ITMO456

Managing Linux Processes

Sean Hughes-Durkin

ITMO/IT-O 456 Fall 2017

Information Technology & Management Programs

School of Applied Technology

Objectives

At the end of this lesson students should be able to:

- Categorize the different types of processes on a Linux system
- View processes using standard Linux utilities
- Explain the difference between common kill signals
- Describe how binary programs and shell scripts are executed

Objectives

At the end of this lesson students should be able to:

- Use standard Linux utilities to modify the priority of a process
- Schedule commands to execute in the future using the at daemon
- Schedule commands to execute repetitively using the cron daemon

Linux Processes

- Program
  - Structured set of commands stored in an executable file on a filesystem
  - Executed to create a process
- Process
  - Program running in memory and on the CPU

Linux Processes

- User process
  - Process begun by a user that runs on a terminal (tty)
- Daemon process
  - System process
  - Not associated with a terminal
- Process ID (PID)
  - Unique identifier assigned to every process as it begins

Linux Processes

- Child processes
  - Refers to a process that was started by another process (parent process)
- Parent processes
  - Process that has started other processes (child processes)
- Parent Process ID (PPID)
  - The PID of the parent process that created the current process
  - The init daemon has a PID of 1 and a PPID of 0

Process Environment

- Each process has its own environment:

process environment

Program name	User and Group ID
Internal data	Process ID (PID)
Open Files	Parent PID (PPID)
Current Directory	Program variables
additional parameters	

  - To see the PID of your current shell process, type: **$ echo $$**

Linux Processes

- The init daemon starts most other daemons during the system initialization process
  - Including those that allow for user logins
- The login program starts a BASH shell
  - BASH shell then interprets user commands and starts all user processes
- Each process on the Linux system can be traced back to the init daemon by examining PPIDs

Process Management

- Viewing process memory use
- Viewing process CPU use
- Finding runaway processes
- Killing processes
- Changing process priority

Linux Processes

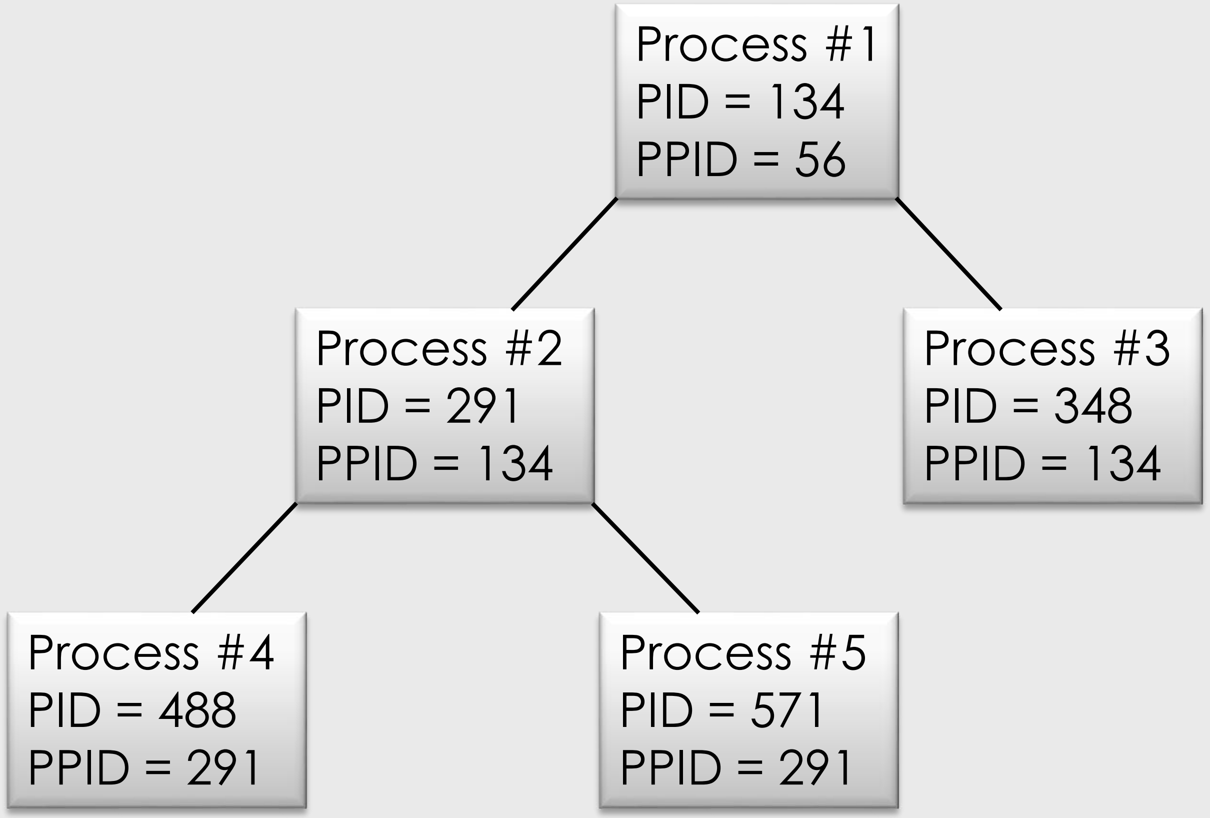


Figure 9-1: Parent and child processes

Linux Processes

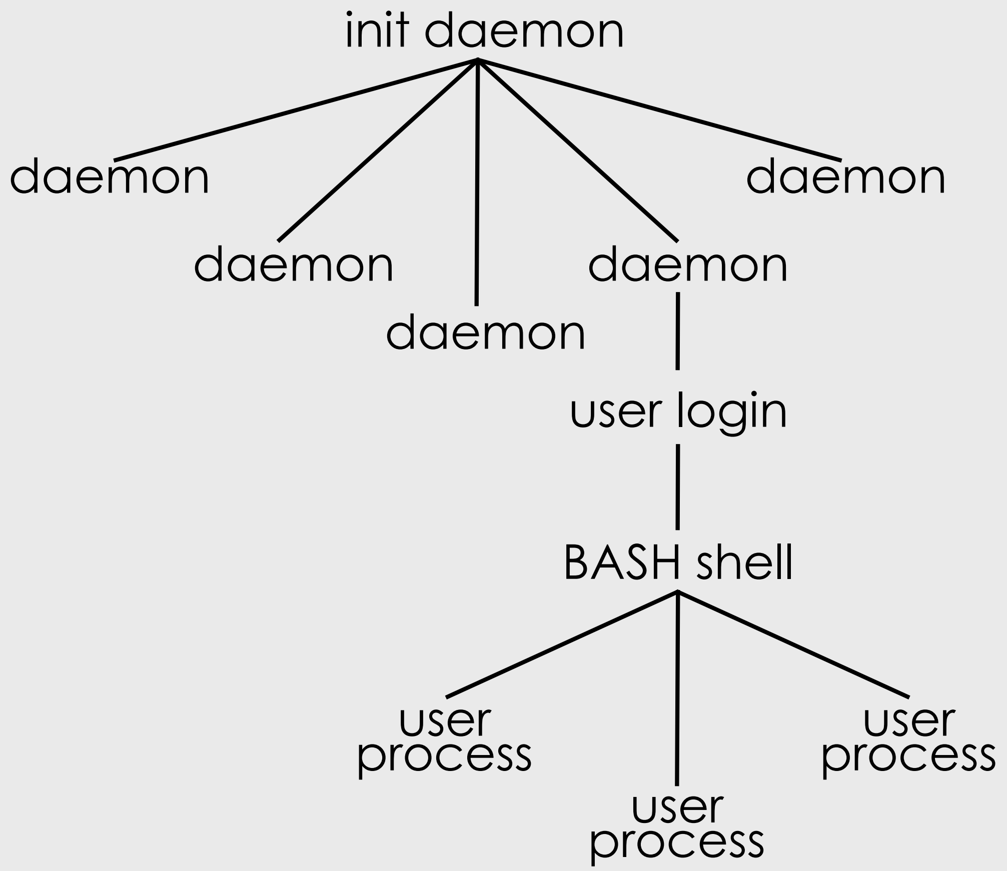
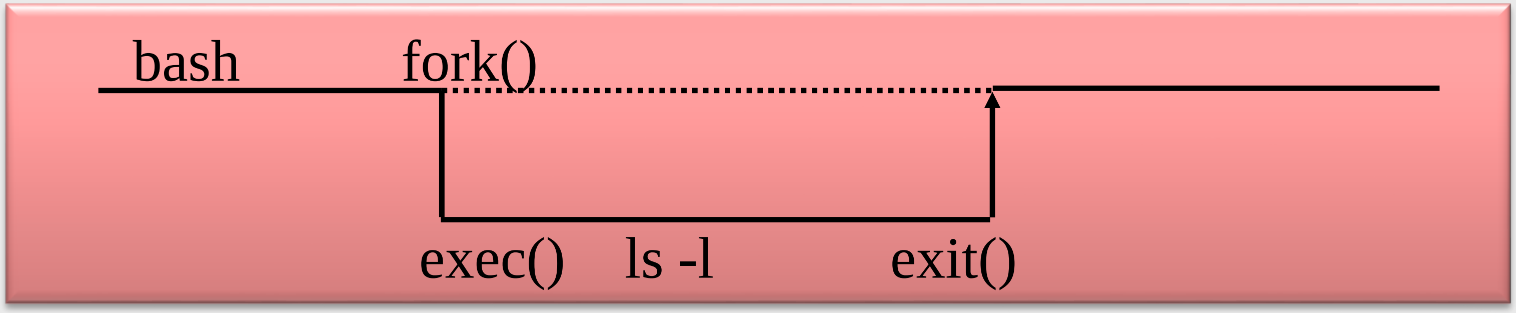


Figure 9-2: Processes genealogy

Starting and Stopping a Process

- All processes are started by other processes
  - Parent/Child relationship

```
$ ls -l
```



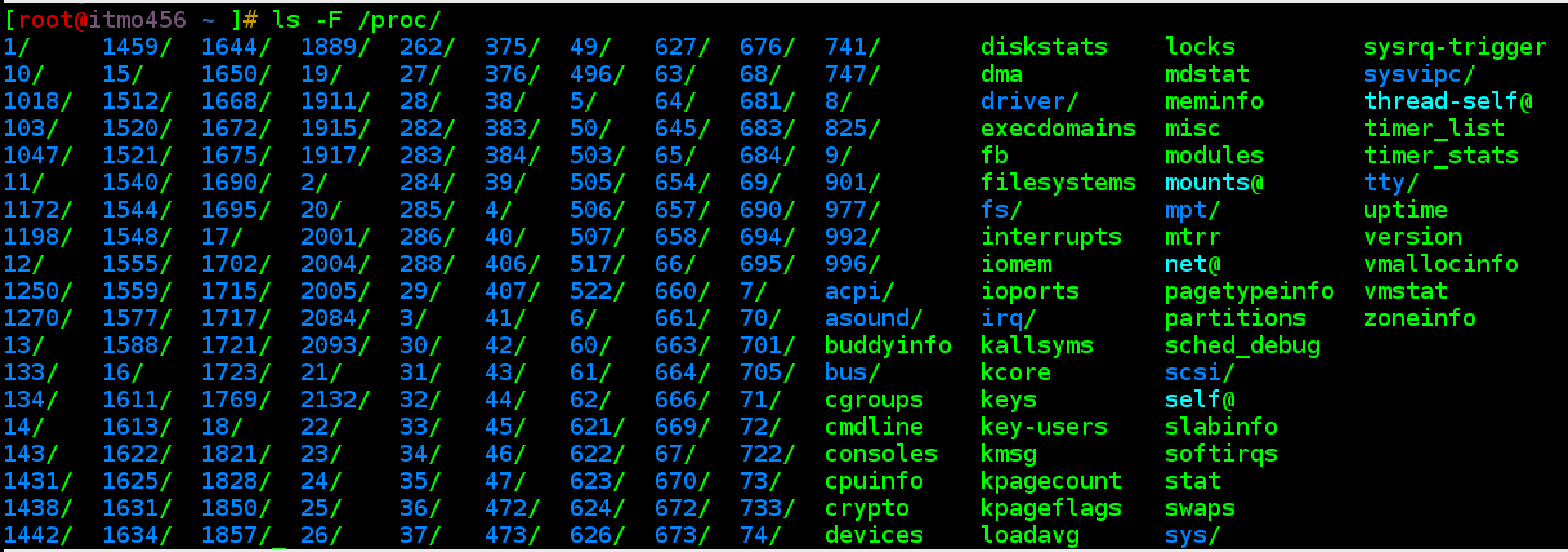
Starting and Stopping a Process

- A process can be terminated for two reasons:
  - The process terminates itself when done
  - The process is terminated by a signal from another process

Viewing Processes

- Information on each running process is stored in the **/proc** directory of the hard drive
- Contains subdirectories by process number (PID)
- Several utilities commonly used to view processes

Viewing Processes



```
[root@itmo456 ~ ]# ls -F /proc/
1/     1459/  1644/  1889/  262/  375/  49/   627/  676/  741/       diskstats    locks         sysrq-trigger
10/    15/    1650/  19/    27/   376/  496/  63/   68/   747/       dma          mdstat        sysvipc/
1018/  1512/  1668/  1911/  28/   38/   5/    64/   681/  8/         driver/      meminfo       thread-self@
103/   1520/  1672/  1915/  282/  383/  50/   645/  683/  825/       execdomains  misc          timer_list
1047/  1521/  1675/  1917/  283/  384/  503/  65/   684/  9/         fb           modules       timer_stats
11/    1540/  1690/  2/     284/  39/   505/  654/  69/   901/       filesystems  mounts@       tty/
1172/  1544/  1695/  20/    285/  4/    506/  657/  690/  977/       fs/          mpt/          uptime
1198/  1548/  17/    2001/  286/  40/   507/  658/  694/  992/       interrupts   mtrr          version
12/    1555/  1702/  2004/  288/  406/  517/  66/   695/  996/       iomem        net@          vmallocinfo
1250/  1559/  1715/  2005/  29/   407/  522/  660/  7/    acpi/      ioports      pagetypeinfo  vmstat
1270/  1577/  1717/  2084/  3/    41/   6/    661/  70/   asound/    irq/         partitions    zoneinfo
13/    1588/  1721/  2093/  30/   42/   60/   663/  701/  buddyinfo  kallsyms     sched_debug
133/   16/    1723/  21/    31/   43/   61/   664/  705/  bus/       kcore        scsi/
134/   1611/  1769/  2132/  32/   44/   62/   666/  71/   cgroups    keys         self@
14/    1613/  18/    22/    33/   45/   621/  669/  72/   cmdline    key-users    slabinfo
143/   1622/  1821/  23/    34/   46/   622/  67/   722/  consoles   kmsg         softirqs
1431/  1625/  1828/  24/    35/   47/   623/  670/  73/   cpuinfo    kpagecount   stat
1438/  1631/  1850/  25/    36/   472/  624/  672/  733/  crypto     kpageflags   swaps
1442/  1634/  1857/  26/    37/   473/  626/  673/  74/   devices    loadavg      sys/
```

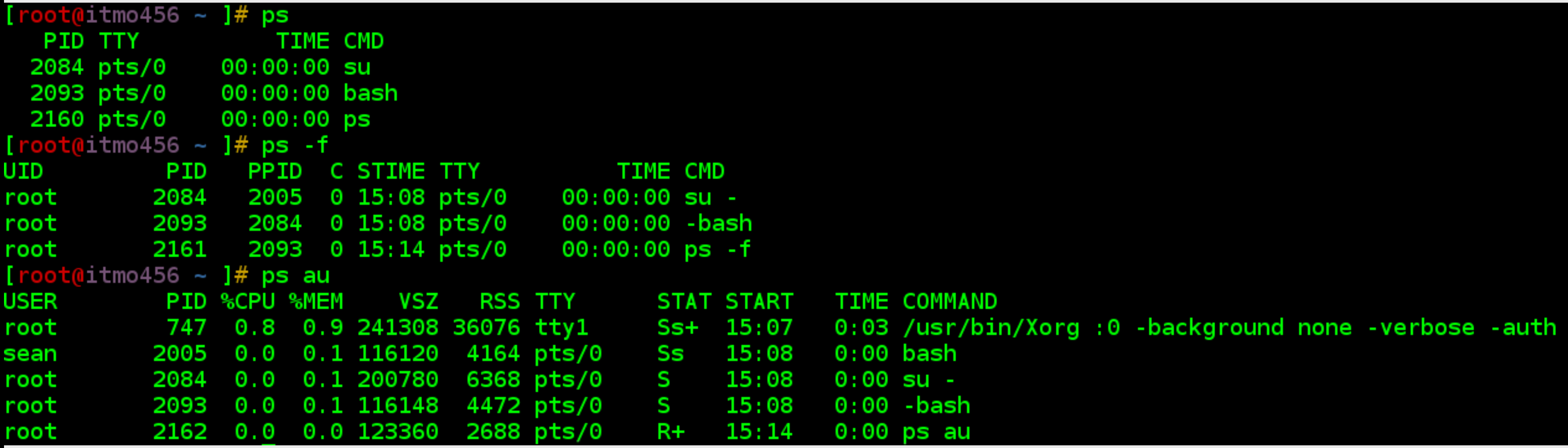
Viewing Processes

- **ps** command
  - Most versatile and common process viewing utility
  - No arguments required
    - Lists all processes running in current shell
    - PID, terminal, command that started process, CPU time
  - Accepts options in several styles
    - UNIX: may be grouped; must be preceded by a dash
    - BSD: may be grouped; must not be used with a dash
    - GNU; may not be grouped, preceded by two dashes

Viewing Processes

- `ps -f` (full) option
  - More complete information
  - User identifier (UID), PPID, start time
- `ps -e --forest`
  - Displays the entire list of processes across all terminals including daemons
  - Helpful in learning about your system

Viewing Processes



```
[root@itmo456 ~ ]# ps
  PID TTY          TIME CMD
 2084 pts/0    00:00:00 su
 2093 pts/0    00:00:00 bash
 2160 pts/0    00:00:00 ps
[root@itmo456 ~ ]# ps -f
UID         PID   PPID  C STIME TTY          TIME CMD
root       2084   2005  0 15:08 pts/0    00:00:00 su -
root       2093   2084  0 15:08 pts/0    00:00:00 -bash
root       2161   2093  0 15:14 pts/0    00:00:00 ps -f
[root@itmo456 ~ ]# ps au
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root        747  0.8  0.9 241308 36076 tty1     Ss+  15:07   0:03 /usr/bin/Xorg :0 -background none -verbose -auth
sean       2005  0.0  0.1 116120  4164 pts/0    Ss   15:08   0:00 bash
root       2084  0.0  0.1 200780  6368 pts/0    S    15:08   0:00 su -
root       2093  0.0  0.1 116148  4472 pts/0    S    15:08   0:00 -bash
root       2162  0.0  0.0 123360  2688 pts/0    R+   15:14   0:00 ps au
```

Viewing Processes

- **ps u** (full) option
  - Username (USER)
  - Process ID (PID)
  - Time process began to run (START)
  - Cumulative system time used (TIME)
  - Associated command (COMMAND)
- **ps ux** option
  - All processes running on the Linux system for the current user

Viewing Processes

- `ps aux`
  - Shows all processes for all users in a user-friendly format
  - Uses BSD-format options
- `ps -o ‘item1,item2,…itemN’`
  - Show the listed items in the display
- `ps --sort=item`
  - Show the list sorted by item

Viewing Processes

- **pstree** shows process hierarchy

```
$ pstree
init-+-apmd
     |-atd
     |-crond
     |-gpm
     |-httpd---10*[httpd]
     |-inetd
     |-kattraction.kss
     |-kdm-+-X
     |     `-kdm---kwm-+-kbgndwm
     |                 |-kfm
     |                 |-kpanel
     |                 |-krootwm
     |                 |-kvt---bash---man---sh-+-gunzip
     |                 |                       `-less
     |                 `-startkde---autorun
     |-kflushd
```

Viewing Processes

- Process state
  - Current state of the process on the processor
  - Most processes sleeping (S) or running (R)
- Zombie process
  - Process finished, but parent has not released PID
  - Defunct process
  - Process state is Z

Viewing Processes

- Process state can be seen when using the **`ps l`** or the **`top`** command
  - **R** = running, **S** = sleeping, **D** = uninterruptible sleep, **T** = stopped or being traced, **Z** = zombie
  - Modifying trailing values:
    < = negative nice value (high priority)
    N = positive nice value (low priority)
    W = swapped out process
    s = session leader
    + = in the foreground process group

Viewing Processes

- Process priority (PRI)
  - Determines how many processor time slices process will receive
  - Higher value means lower priority
  - 0 (high priority) to 127 (low priority)
- Nice value (NI)
  - Can be used to affect process priority indirectly
  - Higher value means lower priority
  - Measured between -20 (a greater chance of a high priority) and 19 (a greater chance of a lower priority)

Viewing Processes

- **ps -efl** command

```
F S UID        PID  PPID  C PRI  NI ADDR    SZ WCHAN  STIME TTY      TIME CMD
4 S root         1     0  0  76   0    -   702 -      Jul09 ?    00:00:06 init [5]
1 S root         2     1  0  75   0    -     0 -      Jul09 ?    00:00:00 [keventd]
1 S root         3     1  0  75   0    -     0 -      Jul09 ?    00:00:00 [kapmd]
1 S root         4     1  0  94  19    -     0 -      Jul09 ?    00:00:00 [ksoftirqd/0]
1 S root         6     1  0  85   0    -     0 -      Jul09 ?    00:00:00 [bdflush]
1 S root         5     1  0  76   0    -     0 -      Jul09 ?    00:00:01 [kswapd]
1 S root         7     1  0  75   0    -     0 -      Jul09 ?    00:00:00 [kupdated]
1 S root         8     1  0  80   0    -     0 -      Jul09 ?    00:00:00 [mdrecoveryd]
1 S root        12     1  0  75   0    -     0 -      Jul09 ?    00:00:00 [kjournald]
1 S root        39     2  0  80   0    -     0 worker 08:37 ?    00:00:00 [ksnapd]
1 S root        81     1  0  75   0    -     0 -      Jul09 ?    00:00:00 [khubd]
1 S root       240     2  0  80   0    -     0 worker 08:37 ?    00:00:00 [kdmflush]
1 S root       247     2  0  80   0    -     0 worker 08:37 ?    00:00:00 [kdmflush]
1 S root       256     2  0  80   0    -     0 kjourn 08:37 ?    00:00:00 [jbd2/dm-0-8]
5 S root       340     1  0  76  -4    -   708 poll_s 08:37 ?    00:00:00 /sbin/udevd -d
1 S root      2871     1  0  76   0    -   823 -      Jul09 ?    00:00:00 syslogd -m 0
5 S root      2875     1  0  76   0    -   463 -      Jul09 ?    00:00:00 klogd -x
5 S rpc       2896     1  0  76   0    -   737 -      Jul09 ?    00:00:00 portmap
5 S rpcuser   2916     1  0  77   0    -   796 -      Jul09 ?    00:00:00 rpc.statd
5 S root      2972     1  0  75   0    -   587 -      Jul09 ?    00:00:00 /usr/sbin/apmd
5 S root      3074     1  0  76   0    -  1114 -      Jul09 ?    00:00:01 /usr/sbin/sshd
```

Viewing Processes

- `ps -A --forest` command

```
PID  TTY      TIME     COMMAND
1309 ?        00:00:00 hald
1310 ?        00:00:00  \_ hald-runner
1339 ?        00:00:00      \_ hald-addon-inpu
1349 ?        00:00:01      \_ hald-addon-stor
1350 ?        00:00:00      \_ hald-addon-acpi
1371 ?        00:00:00 pcscd
1430 ?        00:00:00 VBoxService
1450 ?        00:00:00 sshd
1458 ?        00:00:00 ntpd
1474 ?        00:00:00 sendmail
1483 ?        00:00:00 sendmail
1493 ?        00:00:00 abrtd
1501 ?        00:00:00 crond
1512 ?        00:00:00 atd
1520 ?        00:00:00 gdm-binary
1543 ?        00:00:00  \_ gdm-simple-slav
1546 tty1     00:00:14      \_ Xorg
1705 ?        00:00:00      \_ gdm-session-wor
1728 ?        00:00:01          \_ gnome-session
1862 ?        00:00:00              \_ metacity
1875 ?        00:00:01              \_ gnome-panel
1881 ?        00:00:01              \_ nautilus
```

Viewing Processes

- **`ps aux`** command (also **`ps -e u`**)

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
rpc       1140  0.0  0.0   2568   800 ?        Ss   08:38   0:00 rpcbind
root      1151  0.0  0.0   2440   224 ?        Ss   08:38   0:00 mdadm --monitor --scan -f --pid-file=/var/run/mdadm/mdadm.pid
dbus      1160  0.0  0.0  14888  1832 ?        Ssl  08:38   0:01 dbus-daemon --system
root      1171  0.0  0.2  21020  4688 ?        Ssl  08:38   0:00 NetworkManager --pid-file=/var/run/NetworkManager/NetworkManager.pid
root      1176  0.0  0.1   4720  2292 ?        S    08:38   0:00 /usr/sbin/modem-manager
root      1180  0.0  0.0   2828  1220 ?        S    08:38   0:00 /sbin/dhclient -d -4 -sf /usr/libexec/nm-dhcp-client.action -pf /var/run/dhclient-
root      1186  0.0  0.0   6252   776 ?        Ss   08:38   0:00 /usr/sbin/wpa_supplicant -c /etc/wpa_supplicant/wpa_supplicant.conf -B -u -f /var/
avahi     1188  0.0  0.0   3116  1312 ?        S    08:38   0:00 avahi-daemon: registering [itm456fedora.local]
avahi     1189  0.0  0.0   3116   164 ?        Ss   08:38   0:00 avahi-daemon: chroot helper
rpcuser   1203  0.0  0.0   2636  1072 ?        Ss   08:38   0:00 rpc.statd
root      1238  0.0  0.0      0     0 ?        S    08:38   0:00 [rpciod/0]
root      1251  0.0  0.0   2800   376 ?        Ss   08:38   0:00 rpc.idmapd
root      1273  0.0  0.1  11584  2724 ?        Ss   08:38   0:00 cupsd -C /etc/cups/cupsd.conf
root      1301  0.0  0.0   2024   568 ?        Ss   08:38   0:00 /usr/sbin/acpid
68        1309  0.0  0.2  17052  4204 ?        Ssl  08:38   0:00 hald
root      1310  0.0  0.0   3908  1212 ?        S    08:38   0:00 hald-runner
root      1339  0.0  0.0   3980   836 ?        S    08:38   0:00 hald-addon-input: Listening on /dev/input/event2 /dev/input/event1 /dev/input/even
root      1349  0.0  0.0   3980  1160 ?        S    08:38   0:02 hald-addon-storage: polling /dev/sr0 (every 2 sec)
68        1350  0.0  0.0   3592  1052 ?        S    08:38   0:00 /usr/libexec/hald-addon-acpi
root      1371  0.0  0.0  14500  1412 ?        Ssl  08:38   0:00 pcscd
root      1430  0.0  0.0   9644   456 ?        Sl   08:38   0:00 /usr/sbin/VBoxService
root      1450  0.0  0.0   6568  1080 ?        Ss   08:38   0:00 /usr/sbin/sshd
ntp       1458  0.0  0.0   5308  1908 ?        Ss   08:38   0:00 ntpd -u ntp:ntp -p /var/run/ntpd.pid -g
root      1474  0.0  0.0  11556  1648 ?        Ss   08:38   0:00 sendmail: accepting connections
smmsp     1483  0.0  0.0   9672  1488 ?        Ss   08:38   0:00 sendmail: Queue runner@01:00:00 for /var/spool/clientmqueue
root      1493  0.0  0.2  21564  4184 ?        Ss   08:38   0:00 /usr/sbin/abrtd
root      1501  0.0  0.0   5840  1288 ?        Ss   08:38   0:00 crond
root      1512  0.0  0.0   2876   348 ?        Ss   08:38   0:00 /usr/sbin/atd
root      1520  0.0  0.0  16468  2060 ?        Ssl  08:38   0:00 /usr/sbin/gdm-binary -nodaemon
root      1525  0.0  0.0   2012   452 tty2     Ss+  08:38   0:00 /sbin/mingetty /dev/tty2
root      1529  0.0  0.0   2012   448 tty3     Ss+  08:38   0:00 /sbin/mingetty /dev/tty3
root      1533  0.0  0.0   2012   456 tty4     Ss+  08:38   0:00 /sbin/mingetty /dev/tty4
root      1536  0.0  0.0   2012   452 tty5     Ss+  08:38   0:00 /sbin/mingetty /dev/tty5
root      1540  0.0  0.0   2012   456 tty6     Ss+  08:38   0:00 /sbin/mingetty /dev/tty6
```

Viewing Processes

- `ps aux --forest` command

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
68        1309  0.0  0.2  17052  4204 ?        Ssl  08:38   0:00 hald
root      1310  0.0  0.0   3908  1212 ?        S    08:38   0:00  \_ hald-runner
root      1339  0.0  0.0   3980   836 ?        S    08:38   0:00      \_ hald-addon-input: Listening on /dev/input/event2 /dev/input/event1 /dev/in
root      1349  0.0  0.0   3980  1160 ?        S    08:38   0:01      \_ hald-addon-storage: polling /dev/sr0 (every 2 sec)
68        1350  0.0  0.0   3592  1052 ?        S    08:38   0:00      \_ /usr/libexec/hald-addon-acpi
root      1371  0.0  0.0  14500  1412 ?        Ssl  08:38   0:00 pcscd
root      1430  0.0  0.0   9644   456 ?        Sl   08:38   0:00 /usr/sbin/VBoxService
root      1450  0.0  0.0   6568  1080 ?        Ss   08:38   0:00 /usr/sbin/sshd
ntp       1458  0.0  0.0   5308  1908 ?        Ss   08:38   0:00 ntpd -u ntp:ntp -p /var/run/ntpd.pid -g
root      1474  0.0  0.0  11556  1648 ?        Ss   08:38   0:00 sendmail: accepting connections
smmsp     1483  0.0  0.0   9672  1488 ?        Ss   08:38   0:00 sendmail: Queue runner@01:00:00 for /var/spool/clientmqueue
root      1493  0.0  0.2  21564  4184 ?        Ss   08:38   0:00 /usr/sbin/abrtd
root      1501  0.0  0.0   5840  1288 ?        Ss   08:38   0:00 crond
root      1512  0.0  0.0   2876   348 ?        Ss   08:38   0:00 /usr/sbin/atd
root      1520  0.0  0.0  16468  2060 ?        Ssl  08:38   0:00 /usr/sbin/gdm-binary -nodaemon
root      1543  0.0  0.1  20320  3224 ?        Sl   08:38   0:00  \_ /usr/libexec/gdm-simple-slave --display-id /org/gnome/DisplayManager/Display1
root      1546  0.5  1.7  52524 35740 tty1     Ss+  08:38   0:16      \_ /usr/bin/Xorg :0 -nr -verbose -auth /var/run/gdm/auth-for-gdm-HPObsO/datab
root      1705  0.0  0.1  18980  3284 ?        Sl   08:38   0:00      \_ pam: gdm-password
itm456    1728  0.0  0.2  29728  5888 ?        Ssl  08:42   0:01          \_ gnome-session
itm456    1862  0.0  0.4  21632  9136 ?        S    08:42   0:00              \_ metacity
itm456    1875  0.0  0.6  96416 14420 ?        S    08:42   0:01              \_ gnome-panel
itm456    1881  0.0  0.7 123056 15472 ?        S    08:42   0:01              \_ nautilus
itm456    1897  0.0  0.1   5576  3092 ?        S    08:42   0:00              \_ /usr/sbin/restorecond -u
itm456    1909  0.0  0.4  25624  8932 ?        S    08:42   0:00              \_ gpk-update-icon
itm456    1913  0.0  0.2  19508  5444 ?        S    08:42   0:00              \_ abrt-applet
itm456    1915  0.0  0.1  17180  4060 ?        S    08:42   0:00              \_ deja-dup-monitor
itm456    1918  0.0  0.4 155556  9604 ?        S    08:42   0:00              \_ nm-applet --sm-disable
itm456    1923  0.0  0.2  19136  5940 ?        S    08:42   0:00              \_ bluetooth-applet
itm456    1932  0.0  0.3  20716  6788 ?        S    08:42   0:00              \_ /usr/bin/seapplet
itm456    1935  0.0  0.3  86660  8188 ?        S    08:42   0:00              \_ gnome-power-manager
itm456    1937  0.0  0.4 156424  9208 ?        S    08:42   0:00              \_ gnome-volume-control-applet
itm456    1941  0.0  0.2  18668  4824 ?        S    08:42   0:00              \_ /usr/libexec/polkit-gnome-authentication-agent-1
itm456    1952  0.0  0.3  20124  6380 ?        S    08:42   0:00              \_ /usr/libexec/gdu-notification-daemon
root      1525  0.0  0.0   2012   452 tty2     Ss+  08:38   0:00 /sbin/mingetty /dev/tty2
```

Viewing Processes

Option	Description
-e, -A	Displays all processes running on terminals as well as processes that do not run on a terminal (daemons)
-f	Displays a full list of information about each process including the UID, PID, PPID, CPU utilization, start time, terminal, processor time, and command name
-l, l	Displays a long list of information about each process including the flag, state, UID, PID, PPID, CPU utilization, priority, nice value, address, size, WCHAN, terminal, and command name
-H	Displays processes indented to show process hierarchy
a	Displays all processes running on terminals
x	Displays all processes that do not run on terminals
u	Displays processes in a user-oriented format

Table 9-1: Common options to the ps command

Viewing Processes

- **ps -h** command (quick reference)

```
 ********* simple selection *********   ********* selection by list *********
-A all processes                        -C by command name
-N negate selection                     -G by real group ID (supports names)
-a all w/ tty except session leaders    -U by real user ID (supports names)
-d all except session leaders           -g by session OR by effective group name
-e all processes                        -p by process ID
T  all processes on this terminal       -s processes in the sessions given
a  all w/ tty, including other users    -t by tty
g  OBSOLETE -- DO NOT USE               -u by effective user ID (supports names)
r  only running processes               U  processes for specified users
x  processes w/o controlling ttys       t  by tty
*********** output format **********    *********** long options ***********
-o,o user-defined   -f full             --Group  --User  --pid  --cols  --ppid
-j,j job control    s  signal           --group  --user  --sid  --rows  --info
-O,O preloaded -o   v  virtual memory   --cumulative  --format  --deselect
-l,l long           u  user-oriented    --sort  --tty  --forest  --version
-F   extra full     X  registers        --heading  --no-heading  --context
                      ********* misc options *********
-V,V  show version         L  list format codes      f  ASCII art forest
-m,m,-L,-T,H  threads      S  children in sum        -y change -l format
-M,Z  security data        c  true command name      -c scheduling class
-w,w  wide output          n  numeric WCHAN,UID      -H process hierarchy
```

ITMO456

Viewing Processes

- **top** command
  - Most common command used to display processes aside from **ps**
  - Displays its interactive screen listing processes
    - Organized by processor time
    - Processes using most processor time listed first
- **top** command can be used to change the priority of processes or kill them
  - Commands issued interactively using **top** command line

Viewing Processes

- Process display with **ps aux**

```
USER       PID %CPU %MEM   VSZ  RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1  2808  428 ?        S    Jul09   0:06 init [5]
root         2  0.0  0.0     0    0 ?        SW   Jul09   0:00 [keventd]
```

- Process display with **ps axl**

```
F   UID   PID  PPID PRI  NI   VSZ  RSS WCHAN  STAT TTY        TIME COMMAND
4     0     1     0  16   0  2808  428 -      S    ?          0:06 init [5]
1     0     2     1  15   0     0    0 -      SW   ?          0:00 [keventd]
```

- Process display with **top** (exit with **q**)

```
top - 00:06:29  up 1 day,  8:29,  1 user,  load average: 0.00, 0.04, 0.28
Tasks: 54 total, 1 running, 53 sleeping, 0 stopped, 0 zombie
CPU(s):  0.0%us   0.0%sy   0.1%ni   0.0%id   0.0%wa   0.0%hi   99.8%si   0.0%st
Mem:    255532k total,   249084k used,    6448k free,    60712k buffers
Swap:   524152k total,        0k used,  524152k free     31532k cached
█ ← interactive command line
  PID USER      PR  NI VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 4867 sean   16    0 1092 1092  896 R  0.1  0.4   0:00.65 top
    1 root      16   0  428  428  372 S  0.0  0.1   0:06.04 init
   16 root      15  -5    0    0    0 S  1.3  0.0   0:42.34 ata/0
```

Viewing Processes

- **top** interactive commands

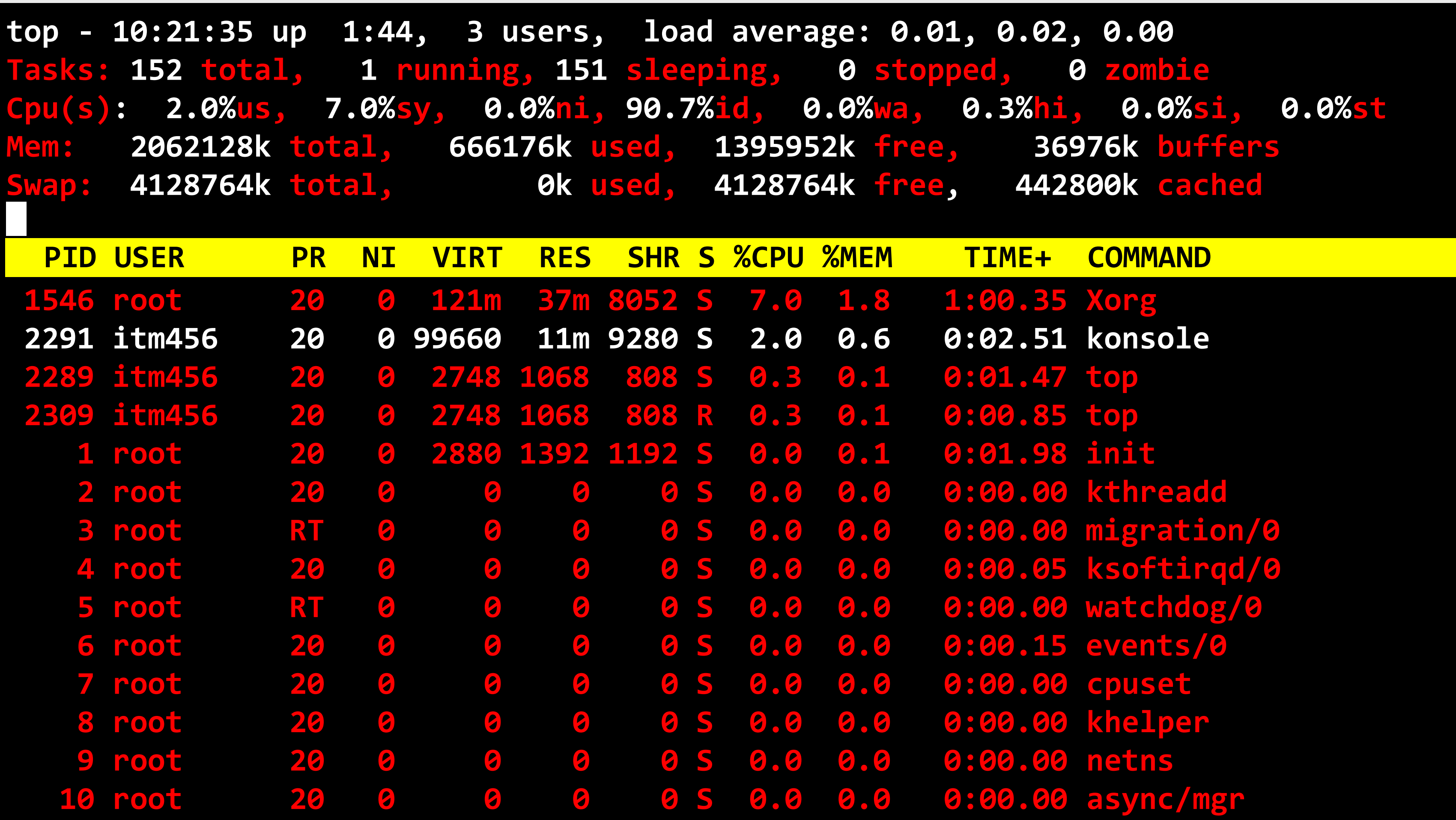
```
Help for Interactive Commands - procps version 3.2.8
Window 1:Def: Cumulative mode Off.  System: Delay 3.0 secs; Secure mode Off.

  Z,B       Global: 'Z' change color mappings; 'B' disable/enable bold
  l,t,m     Toggle Summaries: 'l' load avg; 't' task/cpu stats; 'm' mem info
  1,I       Toggle SMP view: '1' single/separate states; 'I' Irix/Solaris mode

  f,o     . Fields/Columns: 'f' add or remove; 'o' change display order
  F or O  . Select sort field
  <,>     . Move sort field: '<' next col left; '>' next col right
  R,H     . Toggle: 'R' normal/reverse sort; 'H' show threads
  c,i,S   . Toggle: 'c' cmd name/line; 'i' idle tasks; 'S' cumulative time
  x,y     . Toggle highlights: 'x' sort field; 'y' running tasks
  z,b     . Toggle: 'z' color/mono; 'b' bold/reverse (only if 'x' or 'y')
  u       . Show specific user only
  n or #  . Set maximum tasks displayed

  k,r       Manipulate tasks: 'k' kill; 'r' renice
  d or s    Set update interval
  W         Write configuration file
  q         Quit
          ( commands shown with '.' require a visible task display window )
Press 'h' or '?' for help with Windows, any other key to continue
```

Viewing Processes

- **top** command actions
  - **q** to quit top
  - **h** to see help options
  - **M** to sort display by memory usage
  - **P** to sort display by CPU usage
  - **R** to reverse the sort order
  - **u** to enter a username for displaying their processes
  - **1** to toggle between CPU usage for all system CPUs

Viewing Processes

- **top** in color (**z** option @ **top** command line)



```
top - 10:21:35 up  1:44,  3 users,  load average: 0.01, 0.02, 0.00
Tasks: 152 total,   1 running, 151 sleeping,   0 stopped,   0 zombie
Cpu(s):  2.0%us,  7.0%sy,  0.0%ni, 90.7%id,  0.0%wa,  0.3%hi,  0.0%si,  0.0%st
Mem:   2062128k total,   666176k used,  1395952k free,    36976k buffers
Swap:  4128764k total,        0k used,  4128764k free,   442800k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 1546 root      20   0  121m  37m 8052 S  7.0  1.8   1:00.35 Xorg
 2291 itm456    20   0 99660  11m 9280 S  2.0  0.6   0:02.51 konsole
 2289 itm456    20   0  2748 1068  808 S  0.3  0.1   0:01.47 top
 2309 itm456    20   0  2748 1068  808 R  0.3  0.1   0:00.85 top
    1 root      20   0  2880 1392 1192 S  0.0  0.1   0:01.98 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kthreadd
    3 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
    4 root      20   0     0    0    0 S  0.0  0.0   0:00.05 ksoftirqd/0
    5 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 watchdog/0
    6 root      20   0     0    0    0 S  0.0  0.0   0:00.15 events/0
    7 root      20   0     0    0    0 S  0.0  0.0   0:00.00 cpuset
    8 root      20   0     0    0    0 S  0.0  0.0   0:00.00 khelper
    9 root      20   0     0    0    0 S  0.0  0.0   0:00.00 netns
   10 root      20   0     0    0    0 S  0.0  0.0   0:00.00 async/mgr
```

Viewing Processes

Option	Description
-d	*delay* Specifies delay between updates, normally 5 seconds.
-p	*pid* Allows monitoring of specific processes listed by PID; use ps to obtain PIDs. You can specify up to 20 PIDs by using this option multiple times, once for each PID
-n	*iter* Display a certain number of updates (iterations), then quit (Normally top continues updating until terminated)
-b	*batch* Specifies batch mode, in which top doesn't use normal screen-update commands. Could be used to log CPU use of targeted programs to a file

Common options to the top command

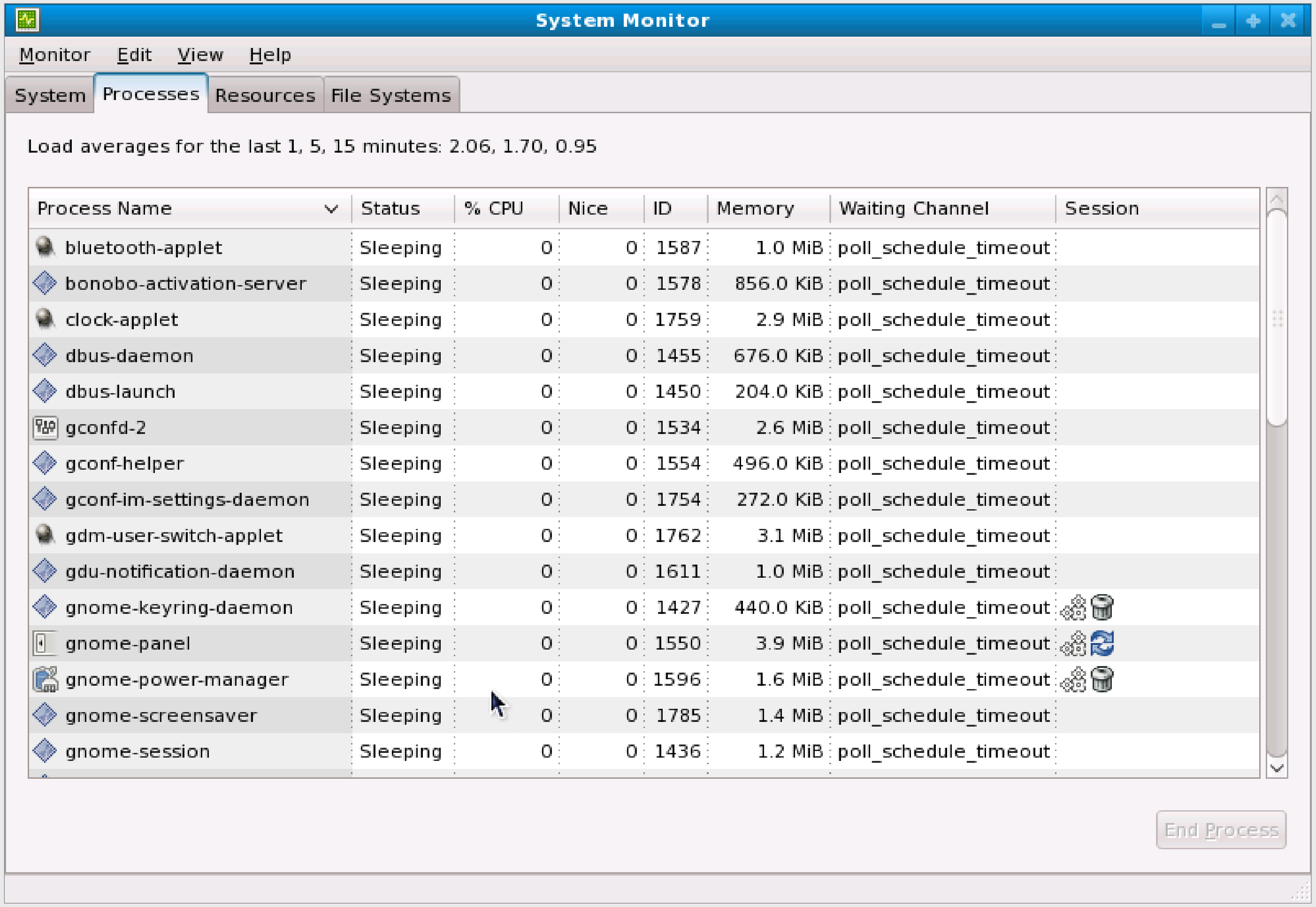
Viewing Processes

- Graphical displays for process viewing
  - GNOME System Monitor (“System Monitor” in Fedora)
  - KDE Process Manager (kpm); aliased to “System Monitor” in Fedora
- Allow sorting by clicking buttons at top of the column

Viewing Processes

- System Monitor provides a more graphical way of displaying processes
  - Sort processes by clicking on the columns
  - Right-click on processes to stop, kill, or renice them
  - By default, only processes associate with your user account are displayed
- Start System Monitor from GNOME
  - **`Applications → System Tools → System Monitor`**

Viewing Processes

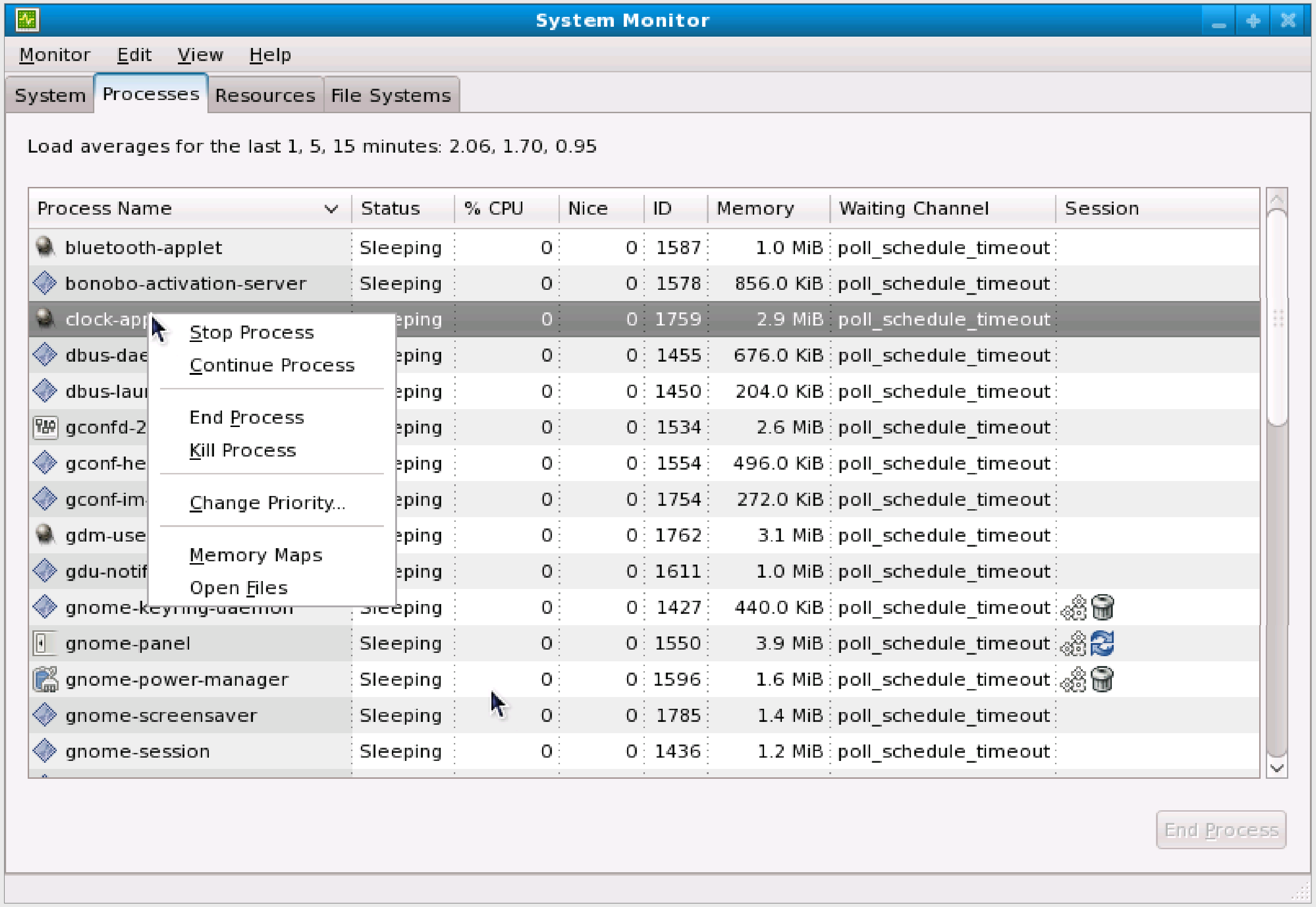


Load averages for the last 1, 5, 15 minutes: 2.06, 1.70, 0.95

Process Name	Status	% CPU	Nice	ID	Memory	Waiting Channel	Session
bluetooth-applet	Sleeping	0	0	1587	1.0 MiB	poll_schedule_timeout	
bonobo-activation-server	Sleeping	0	0	1578	856.0 KiB	poll_schedule_timeout	
clock-applet	Sleeping	0	0	1759	2.9 MiB	poll_schedule_timeout	
dbus-daemon	Sleeping	0	0	1455	676.0 KiB	poll_schedule_timeout	
dbus-launch	Sleeping	0	0	1450	204.0 KiB	poll_schedule_timeout	
gconfd-2	Sleeping	0	0	1534	2.6 MiB	poll_schedule_timeout	
gconf-helper	Sleeping	0	0	1554	496.0 KiB	poll_schedule_timeout	
gconf-im-settings-daemon	Sleeping	0	0	1754	272.0 KiB	poll_schedule_timeout	
gdm-user-switch-applet	Sleeping	0	0	1762	3.1 MiB	poll_schedule_timeout	
gdu-notification-daemon	Sleeping	0	0	1611	1.0 MiB	poll_schedule_timeout	
gnome-keyring-daemon	Sleeping	0	0	1427	440.0 KiB	poll_schedule_timeout	
gnome-panel	Sleeping	0	0	1550	3.9 MiB	poll_schedule_timeout	
gnome-power-manager	Sleeping	0	0	1596	1.6 MiB	poll_schedule_timeout	
gnome-screensaver	Sleeping	0	0	1785	1.4 MiB	poll_schedule_timeout	
gnome-session	Sleeping	0	0	1436	1.2 MiB	poll_schedule_timeout	

Gnome System Monitor: Processes

Viewing Processes

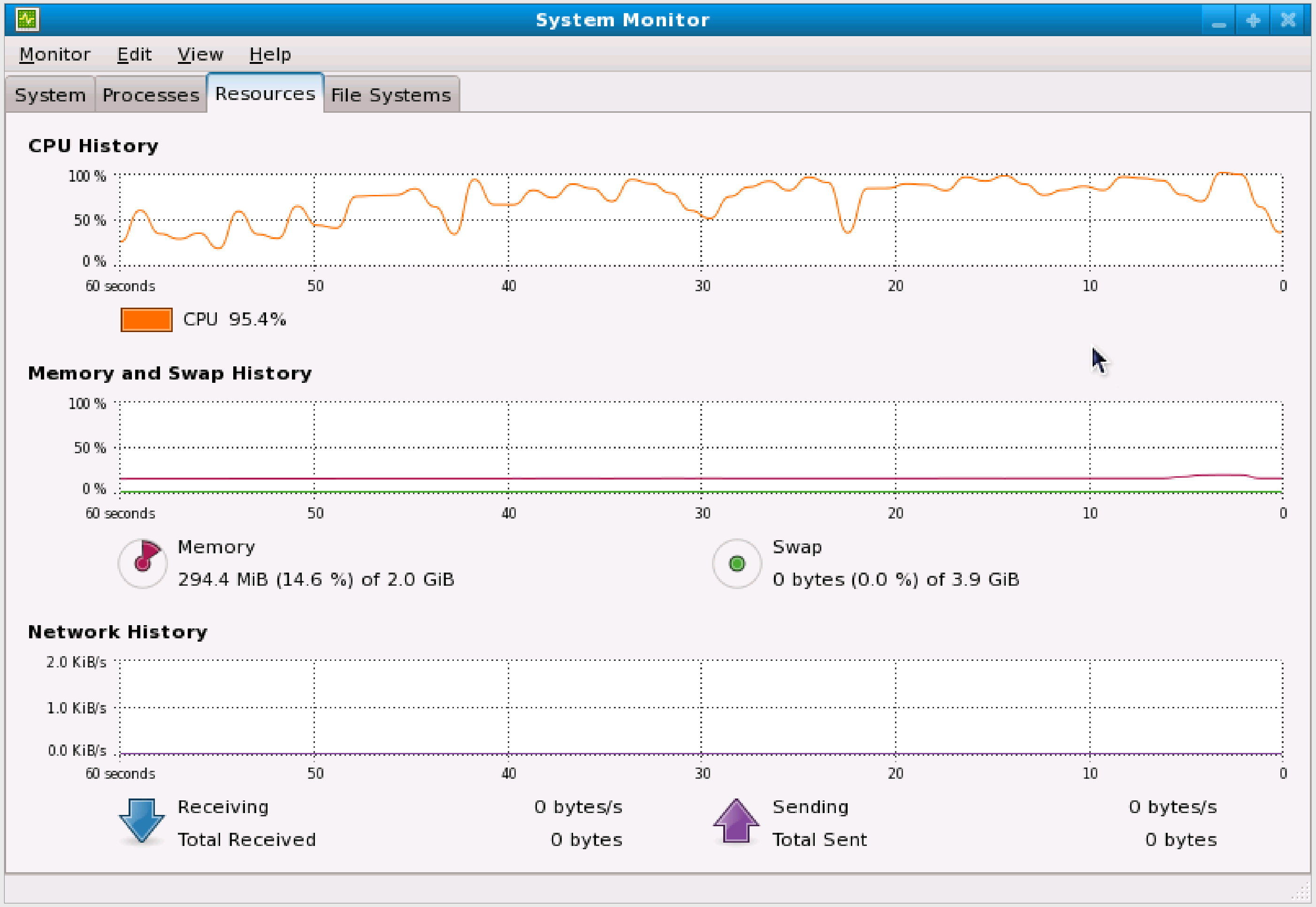


Load averages for the last 1, 5, 15 minutes: 2.06, 1.70, 0.95

Process Name	Status	% CPU	Nice	ID	Memory	Waiting Channel	Session
bluetooth-applet	Sleeping	0	0	1587	1.0 MiB	poll_schedule_timeout	
bonobo-activation-server	Sleeping	0	0	1578	856.0 KiB	poll_schedule_timeout	
clock-app	eping	0	0	1759	2.9 MiB	poll_schedule_timeout	
dbus-dae	eping	0	0	1455	676.0 KiB	poll_schedule_timeout	
dbus-laur	eping	0	0	1450	204.0 KiB	poll_schedule_timeout	
gconfd-2	eping	0	0	1534	2.6 MiB	poll_schedule_timeout	
gconf-he	eping	0	0	1554	496.0 KiB	poll_schedule_timeout	
gconf-im	eping	0	0	1754	272.0 KiB	poll_schedule_timeout	
gdm-use	eping	0	0	1762	3.1 MiB	poll_schedule_timeout	
gdu-notif	eping	0	0	1611	1.0 MiB	poll_schedule_timeout	
gnome-keyring-daemon	Sleeping	0	0	1427	440.0 KiB	poll_schedule_timeout	
gnome-panel	Sleeping	0	0	1550	3.9 MiB	poll_schedule_timeout	
gnome-power-manager	Sleeping	0	0	1596	1.6 MiB	poll_schedule_timeout	
gnome-screensaver	Sleeping	0	0	1785	1.4 MiB	poll_schedule_timeout	
gnome-session	Sleeping	0	0	1436	1.2 MiB	poll_schedule_timeout	

Gnome System Monitor right-click options

Viewing Processes



CPU 95.4%

Memory
294.4 MiB (14.6 %) of 2.0 GiB

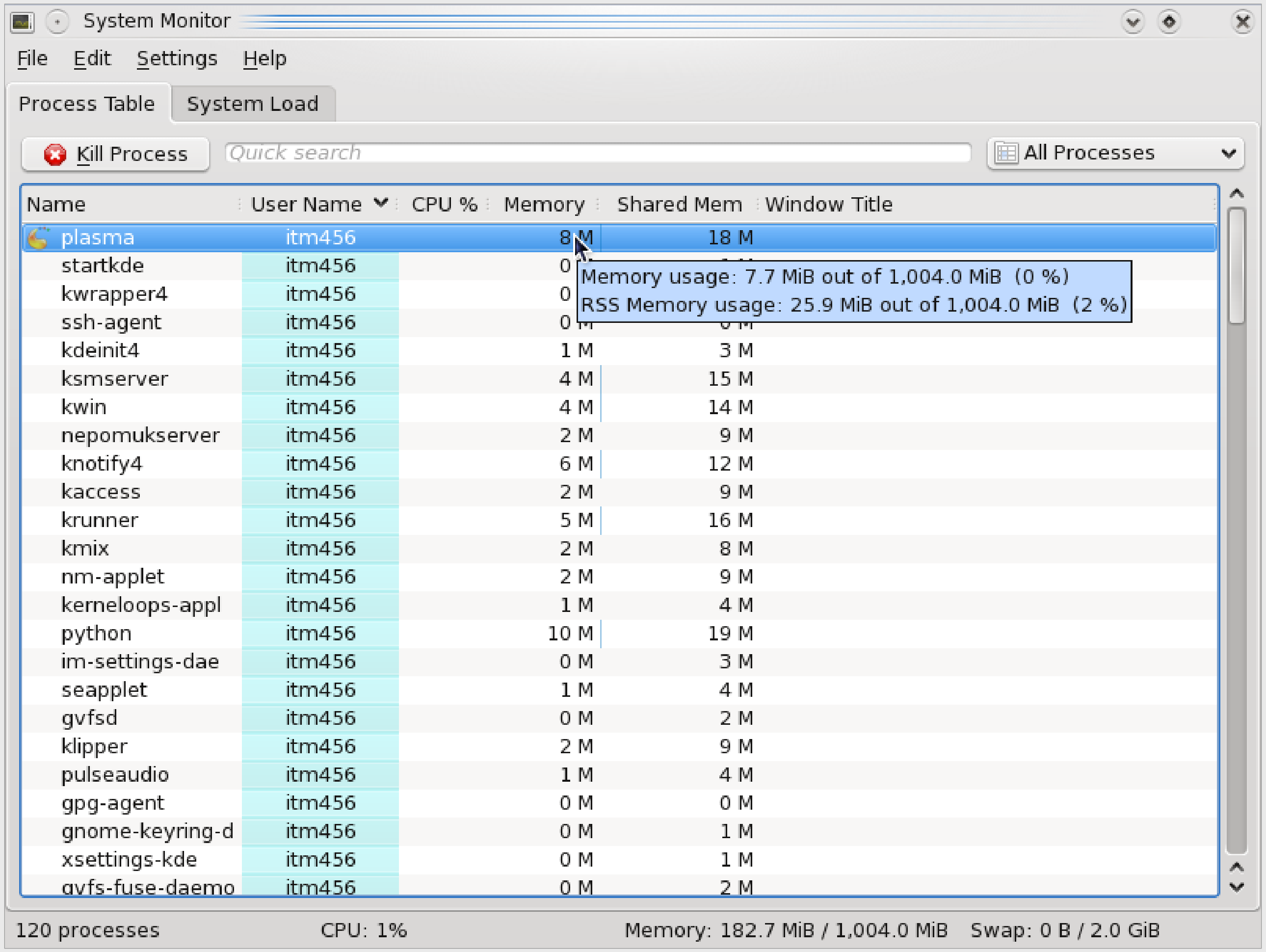
Swap
0 bytes (0.0 %) of 3.9 GiB

Receiving 0 bytes/s
Total Received 0 bytes

Sending 0 bytes/s
Total Sent 0 bytes

Gnome System Monitor: Resources

Viewing Processes



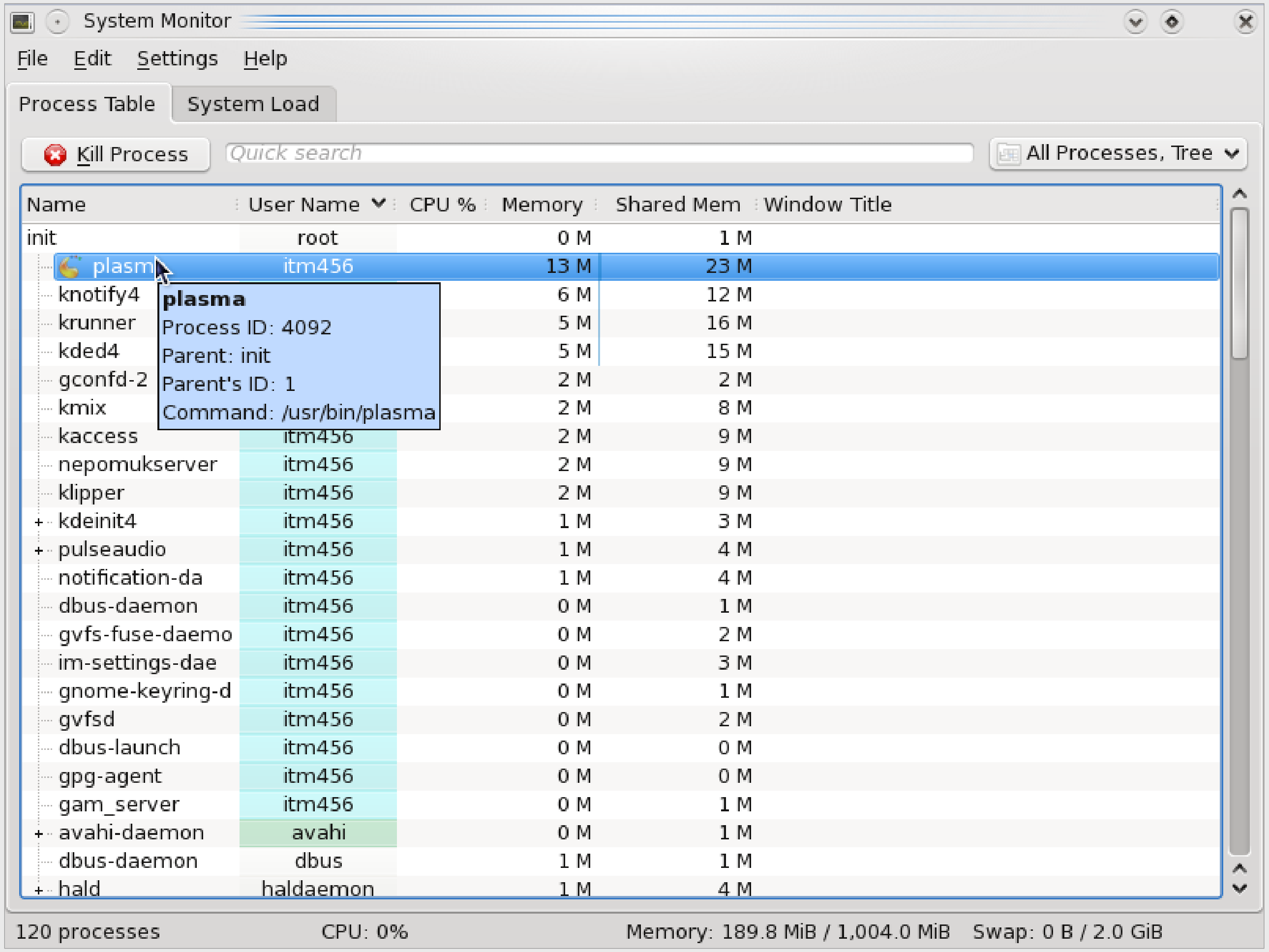
Name	User Name	CPU %	Memory	Shared Mem	Window Title
plasma	itm456		8 M	18 M	
startkde	itm456		0		
kwrapper4	itm456		0		
ssh-agent	itm456		0 M	0 M	
kdeinit4	itm456		1 M	3 M	
ksmserver	itm456		4 M	15 M	
kwin	itm456		4 M	14 M	
nepomukserver	itm456		2 M	9 M	
knotify4	itm456		6 M	12 M	
kaccess	itm456		2 M	9 M	
krunner	itm456		5 M	16 M	
kmix	itm456		2 M	8 M	
nm-applet	itm456		2 M	9 M	
kerneloops-appl	itm456		1 M	4 M	
python	itm456		10 M	19 M	
im-settings-dae	itm456		0 M	3 M	
seapplet	itm456		1 M	4 M	
gvfsd	itm456		0 M	2 M	
klipper	itm456		2 M	9 M	
pulseaudio	itm456		1 M	4 M	
gpg-agent	itm456		0 M	0 M	
gnome-keyring-d	itm456		0 M	1 M	
xsettings-kde	itm456		0 M	1 M	
gvfs-fuse-daemo	itm456		0 M	2 M	

Memory usage: 7.7 MiB out of 1,004.0 MiB (0 %)
RSS Memory usage: 25.9 MiB out of 1,004.0 MiB (2 %)

120 processes CPU: 1% Memory: 182.7 MiB / 1,004.0 MiB Swap: 0 B / 2.0 GiB

KDE Process Manager (kpm)

Viewing Processes



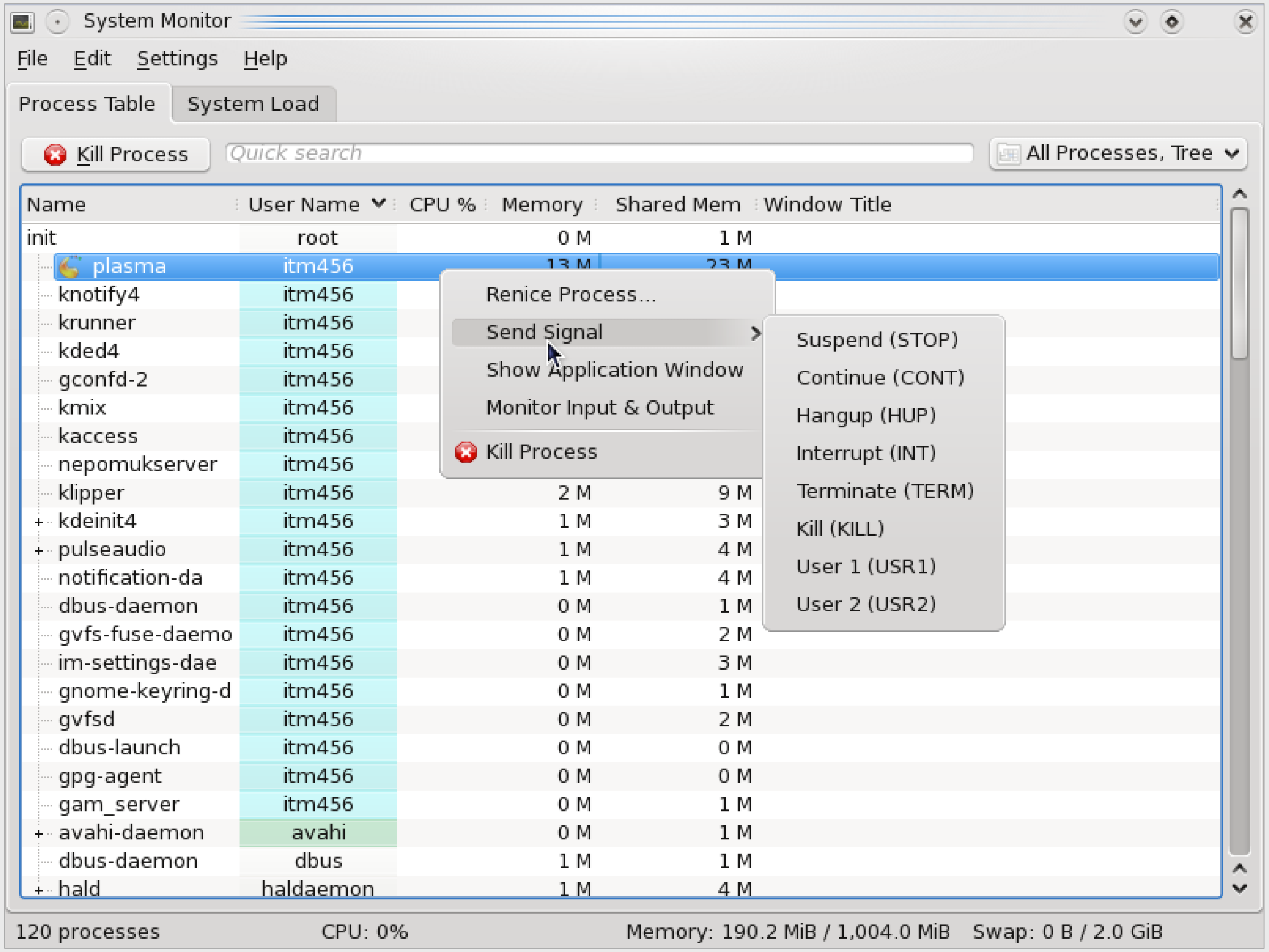
Name	User Name	CPU %	Memory	Shared Mem	Window Title
init	root		0 M	1 M	
plasm	itm456		13 M	23 M	
knotify4			6 M	12 M	
krunner			5 M	16 M	
kded4			5 M	15 M	
gconfd-2			2 M	2 M	
kmix			2 M	8 M	
kaccess	itm456		2 M	9 M	
nepomukserver	itm456		2 M	9 M	
klipper	itm456		2 M	9 M	
+ kdeinit4	itm456		1 M	3 M	
+ pulseaudio	itm456		1 M	4 M	
notification-da	itm456		1 M	4 M	
dbus-daemon	itm456		0 M	1 M	
gvfs-fuse-daemo	itm456		0 M	2 M	
im-settings-dae	itm456		0 M	3 M	
gnome-keyring-d	itm456		0 M	1 M	
gvfsd	itm456		0 M	2 M	
dbus-launch	itm456		0 M	0 M	
gpg-agent	itm456		0 M	0 M	
gam_server	itm456		0 M	1 M	
+ avahi-daemon	avahi		0 M	1 M	
dbus-daemon	dbus		1 M	1 M	
+ hald	haldaemon		1 M	4 M	

plasma
Process ID: 4092
Parent: init
Parent's ID: 1
Command: /usr/bin/plasma

120 processes CPU: 0% Memory: 189.8 MiB / 1,004.0 MiB Swap: 0 B / 2.0 GiB

KDE System Monitor process tree

Viewing Processes

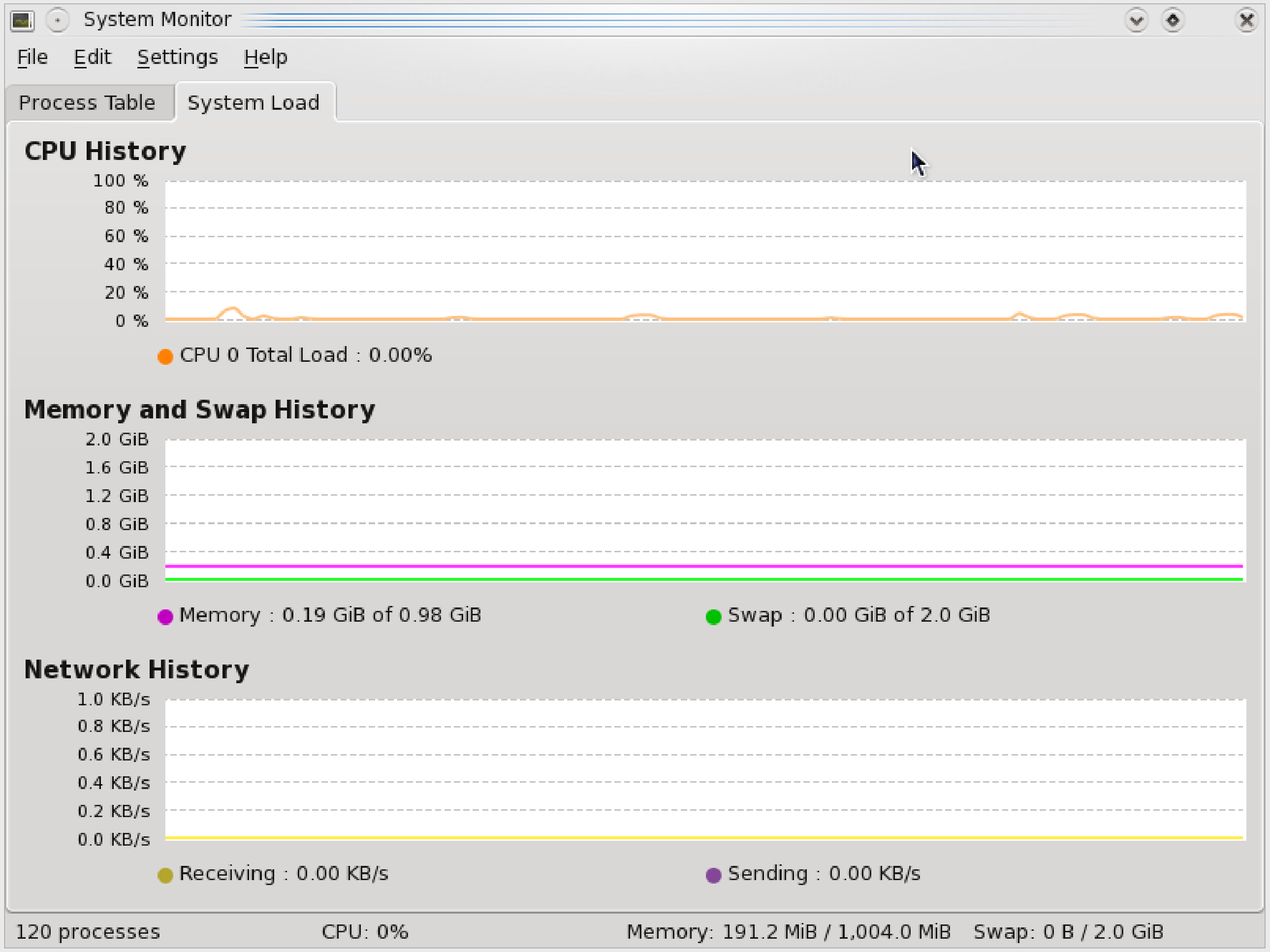


Name	User Name	CPU %	Memory	Shared Mem	Window Title
init	root		0 M	1 M	
plasma	itm456		13 M	23 M	
knotify4	itm456				
krunner	itm456				
kded4	itm456				
gconfd-2	itm456				
kmix	itm456				
kaccess	itm456				
nepomukserver	itm456				
klipper	itm456		2 M	9 M	
+ kdeinit4	itm456		1 M	3 M	
+ pulseaudio	itm456		1 M	4 M	
notification-da	itm456		1 M	4 M	
dbus-daemon	itm456		0 M	1 M	
gvfs-fuse-daemo	itm456		0 M	2 M	
im-settings-dae	itm456		0 M	3 M	
gnome-keyring-d	itm456		0 M	1 M	
gvfsd	itm456		0 M	2 M	
dbus-launch	itm456		0 M	0 M	
gpg-agent	itm456		0 M	0 M	
gam_server	itm456		0 M	1 M	
+ avahi-daemon	avahi		0 M	1 M	
dbus-daemon	dbus		1 M	1 M	
+ hald	haldaemon		1 M	4 M	

120 processes CPU: 0% Memory: 190.2 MiB / 1,004.0 MiB Swap: 0 B / 2.0 GiB

KDE System Monitor right-click options

Viewing Processes



● CPU 0 Total Load : 0.00%

● Memory : 0.19 GiB of 0.98 GiB ● Swap : 0.00 GiB of 2.0 GiB

● Receiving : 0.00 KB/s ● Sending : 0.00 KB/s

120 processes CPU: 0% Memory: 191.2 MiB / 1,004.0 MiB Swap: 0 B / 2.0 GiB

KDE System Monitor performance monitor

Rogue & Zombie Processes

- Rogue process
  - Process that has become faulty
  - Consumes excessive system resources
- Zombie process
  - Process that is completed but has not had the Process ID released
  - May accumulate & prevent new processes from running
  - Eliminated by killing the parent process
  - Also know as defunct processes

Killing Processes

- Kill signal
  - Signal sent to a process by the kill command
  - 64 types of kill signals
  - Different kill signals affect processes in different ways
  - Used to terminate rogue and zombie processes

Kill Signals

- Several signals can be sent to a process:
  - Using keyboard interrupts (if a foreground process)
  - Using the **kill** command
    - Synopsis: **kill -*signal PID***
  - Using the **killall** command to kill all named apps
    - Synopsis: **killall -*signal application***

Kill Signals

- Most important signals:

Signal	Keyboard	Meaning	Default action
01		hangup	end process
02	`Ctrl-C`	interrupt	end process
03	`Ctrl-\`	quit	end process and core dump
09		kill	end process - cannot be redefined - handled by kernel
15		terminate	end process

Killing Processes

- **`kill`** command
  - Kills all instances of a process by command name
  - **`-l`** option: displays list of kill signal names and associated numbers
  - To kill a process, give kill signal and PID
    - If no kill signal given, SIGTERM assumed
  - Often necessary to kill parent process in order to kill zombie processes

Killing Processes

Name	Number	Description
SIGHUP	1	Also known as the hangup signal, it stops a process and then restarts it with the same PID. If you edit the configuration file used by a running daemon, that daemon may be sent a SIGHUP to restart it; when the daemon starts again, it will read the new configuration file.
SIGINT	2	This signal sends an interrupt signal to a process. Although this signal is one of the weakest kill signals, it works most of the time. When you use the Ctrl-c key combination to kill a currently running process, a SIGINT is actually being sent to the process.
SIGQUIT	3	Also known as a core dump, the quit signal terminates a process by taking the process information in memory and saving it to a file called core on the hard disk in the current working directory. You may use the Ctrl-\ key combination to send a SIGQUIT to a process that is currently running.
SIGTERM	15	The software termination signal is the most common kill signal used by processes to kill other processes. It is the default kill signal used by the kill command.
SIGKILL	9	Also known as the absolute kill signal, it forces the Linux kernel to stop executing the process by sending the process's resources to a special device file called /dev/null.

Table 9-2: Common administrative kill signals

Killing Processes

- Trapping
  - Ignoring a kill signal
    - Some processes trap to protect the process
  - SIGKILL signal cannot be trapped by any process
    - Use only as a last resort
- Kill signals sent to processes having children
  - Parent process will terminate all child processes before terminating itself
  - To kill several related processes send signal to parent process

Killing Processes

- **`killall`** command
  - Kills multiple processes of the same name in one command
  - Takes kill signal number as an option
  - Uses process name instead of PID
  - If no kill signal given, default kill signal, SIGTERM, is used
- Can also use **`top`** command to kill processes

Process Execution

- The three main types of Linux commands that you may execute:
  - Binary programs
    - e.g. **`ls, find, grep`**
  - Shell scripts
  - Shell functions
    - e.g. **`cd, exit`**
    - https://www.gnu.org/software/bash/manual/html_node/Shell-Builtin-Commands.html

Process Execution

- Forking
  - Act of creating a new BASH shell child process from a parent
  - Carried out by the fork function in the BASH shell
  - Subshell executes program or shell script using exec function
  - Original shell waits for subshell to complete
  - When done, subshell kills itself
    - Control returns to original shell

Process Execution

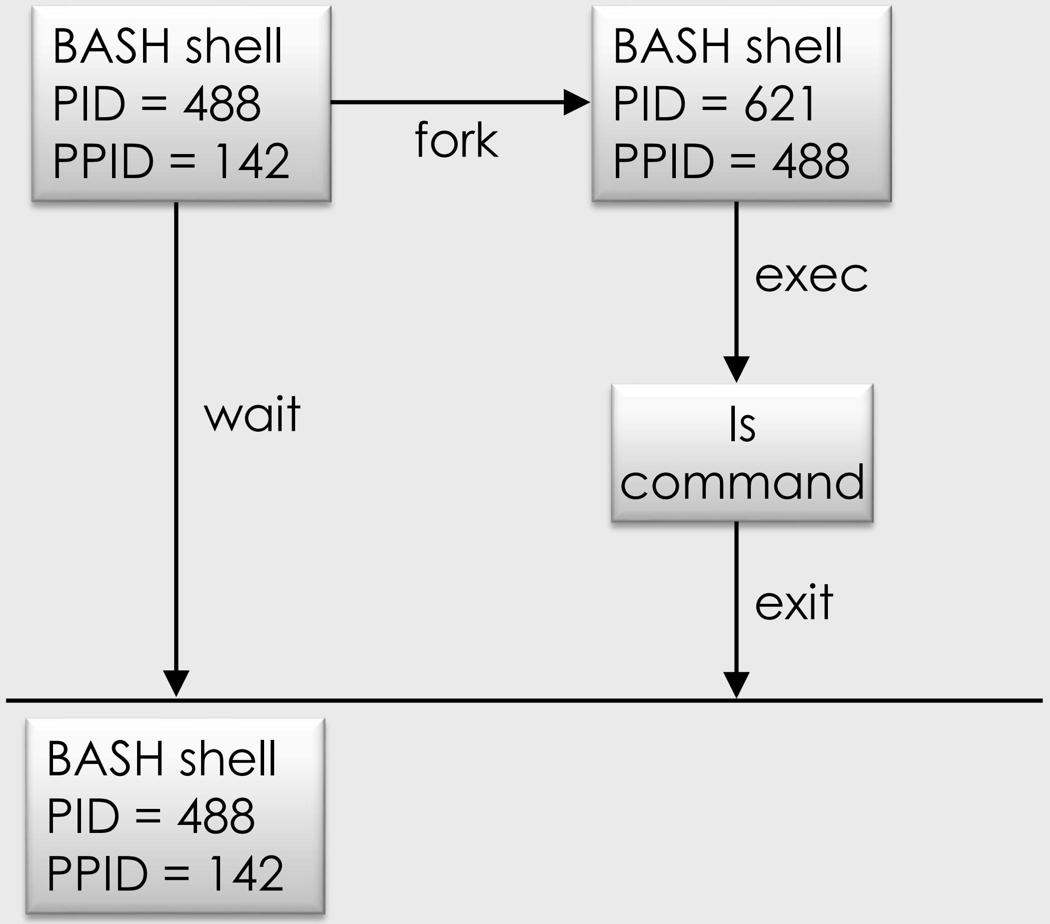


Figure 9-3: Process forking

Process Priorities

- Time slice
  - The amount of time a process is given on a CPU in a multiprocessing operating system
  - More time slices means more execution time on CPU
    - Executes faster
  - Usually measured in milliseconds

Process Priorities

- PRI dictates number of time slices a process gets
  - Cannot change PRI value directly
  - Set NI to indirectly affect priority
    - Negative NI value→more time slices→higher
    - Positive NI value→fewer time slices→lower
- Processes start with NI of 0
- `nice` command
  - Change priority of a process as it starts

Process Priorities

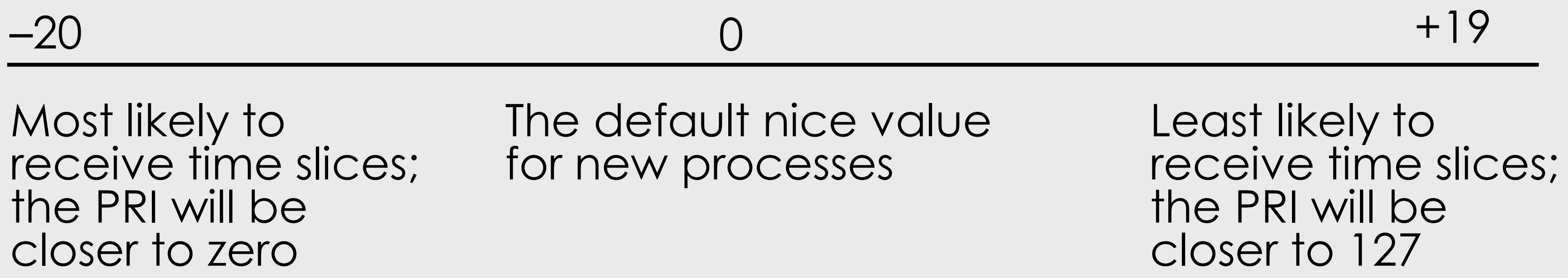

–20
0
+19
Most likely to receive time slices; the PRI will be closer to zero
The default nice value for new processes
Least likely to receive time slices; the PRI will be closer to 127

Figure 9-4: The nice value scale

Process Priorities

- **`renice`** command
  - Alter the nice value of a process after it has been started
  - Only root user may change the nice value to a negative value
  - **`-u`** option changes NI for all processes owned by the specified user or group
- May also change NI of running processes using the **`top`** utility

Scheduling Commands

- at daemon (**`atd`**)
  - System daemon that executes tasks at a future time
  - Configured with the **`at`** command
- cron daemon (**`crond`**)
  - System daemon that executes tasks repetitively in the future
  - Configured using cron tables

Scheduling Commands with **atd**

- **at** command
  - Schedule commands a tasks to run at a preset time in the future
  - Enter **at** with a time parameter, which gives you an **at>** prompt
    - Then type in each command to be executed
    - Type in **Ctrl-d** to end & save

Scheduling Commands with **atd**

- **at** options
  - **-l** option
    - View a list of scheduled jobs
    - Regular users see only their own jobs
  - **-c** option
    - View content of a specified **at** job
  - **-d** option
    - Delete the specified **at** job
  - **-f** option
    - Run scheduled jobs from shell script

Scheduling Commands with **`atd`**

- **`atq`** command
  - Alternative method to view scheduled jobs
- **`at`** daemon uses current shell's environment for execution
  - Shell environment and scheduled commands stored in **`/var/spool/at`**
- If stdout of scheduled command has not been redirected to file, it is mailed to user

Scheduling Commands with **atd**

Command	Description
at 10:15PM	Will schedule commands to run at 10:15PM on the current date
at 10:15PM July 15	Will schedule commands to run at 10:15PM on July 15
at midnight	Will schedule commands to run at midnight on the current date
at noon July 15	Will schedule commands to run at noon on July 15
at teatime	Will schedule commands to run at 4:00PM on the current date (how veddy *British*, eh?)
at tomorrow	Will schedule commands to run at the current time the next day
at now + 5 minutes	Will schedule commands to run in 5 minutes
at now + 10 hours	Will schedule commands to run in 10 hours
at now + 4 days	Will schedule commands to run in 4 days
at now + 2 weeks	Will schedule commands to run in 2 weeks
at now or batch	Will schedule commands to run immediately
at 9:00AM 07/03/2009 or **at 9:00AM 07032009** or **at 9:00AM 07.01.2009**	Will schedule commands to run at 9:00AM on July 3rd 2009

Table 9-3: Common **at** commands

Scheduling Commands with `atd`

- **`/etc/at.allow`**
  - File listing all users who can use **`at`**
- **`/etc/at.deny`**
  - File listing all users who cannot access **`at`**
- If both files exist, only **`/etc/at.allow`** file is processed
- On Fedora Linux, only **`/etc/at.deny`** file exists by default
  - Initially left blank; all users allowed to use at daemon

Scheduling Commands with **`crond`**

- Cron daemon (**crond**)
  - Schedules scripts, applications or shell functions to run on a regular scheduled basis
  - Suitable for scheduling repetitive tasks
  - Should start automatically when the system starts

Scheduling Commands with `crond`

- Configured through cron tables (**`crontab`**)
  - Specify when commands should be executed
  - User and system cron tables
  - Six fields separated by spaces or tabs
    - First 5 specify times to run the command
    - 6th absolute pathname to command to execute
    - Normally use numbers but can use abbreviations for months (*Jan-Dec*) and days (*Mon-Sun*)

Scheduling Commands with `crond`

```
1        2        3        4        5        command
```

1 = minute past the hour (0-59)
2 = hour (0-23)
3 = day of month (1-31)
4 = month of year (1-12)
5 = day of week
    0=Sun (or 7=Sun)
    1=Mon
    2=Tue
    3=Wed
    4=Thu
    5=Fri
    6=Sat

Figure 9-5: User cron table format

Scheduling Commands with `crond`

minute	hour	day_of the_month	month	day_of the_week	command
1	2	3	4	5	command
20,40	17	*	*	1-5	/root/myscript

Figure 9-6: Sample user cron table entry

Scheduling Commands with `crond`

- User cron tables
  - Represent tasks scheduled by individual users
- System cron tables
  - Contains system tasks
- `/var/spool/cron`
  - Stores user cron tables
- `/etc/crontab`
  - Default system cron table
- `/etc/cron.d`
  - Contains system cron tables

User Cron Tables

- **`/etc/cron.allow`**
  - File listing all users who can use **`cron`**
- **`/etc/cron.deny`**
  - File listing all users who cannot access **`cron`**
- If both files exist, only **`/etc/cron.allow`** file is processed
- On Fedora Linux, only **`/etc/cron.deny`** file exists by default
  - Initially left blank, all users allowed to use cron daemon

User Cron Tables

- **`crontab`** command
  - View and edit user cron tables
  - **`-e`** option: Edit cron tables in vi editor
  - **`-l`** option: List a user cron table
  - **`-r`** option: Remove cron table and all scheduled jobs
  - **`-u`** option: used by root user to edit, list, or remove a specified user's cron table

The System Cron Table

- Linux systems are typically scheduled to run many commands during non-business hours
- May perform system maintenance, back up data, or run CPU-intensive programs
- Most of these commands are scheduled by the cron daemon from entries in the system cron table **`/etc/crontab`**

The System Cron Directories

- The system cron table **`/etc/crontab`** normally executes repetitive tasks by using the **`/usr/bin/run-parts`** script, which runs all the scripts & tasks in a specified directory
- Cron has four time-associated directories for this purpose:
  - **`/etc/cron.hourly`**
  - **`/etc/cron.daily`**
  - **`/etc/cron.weekly`**
  - **`/etc/cron.monthly`**

The System Cron Directories

- Items in directories executed by **`run-parts`** are run in alphabetical order
  - Often results in renaming or odd naming of scripts in the cron directories:

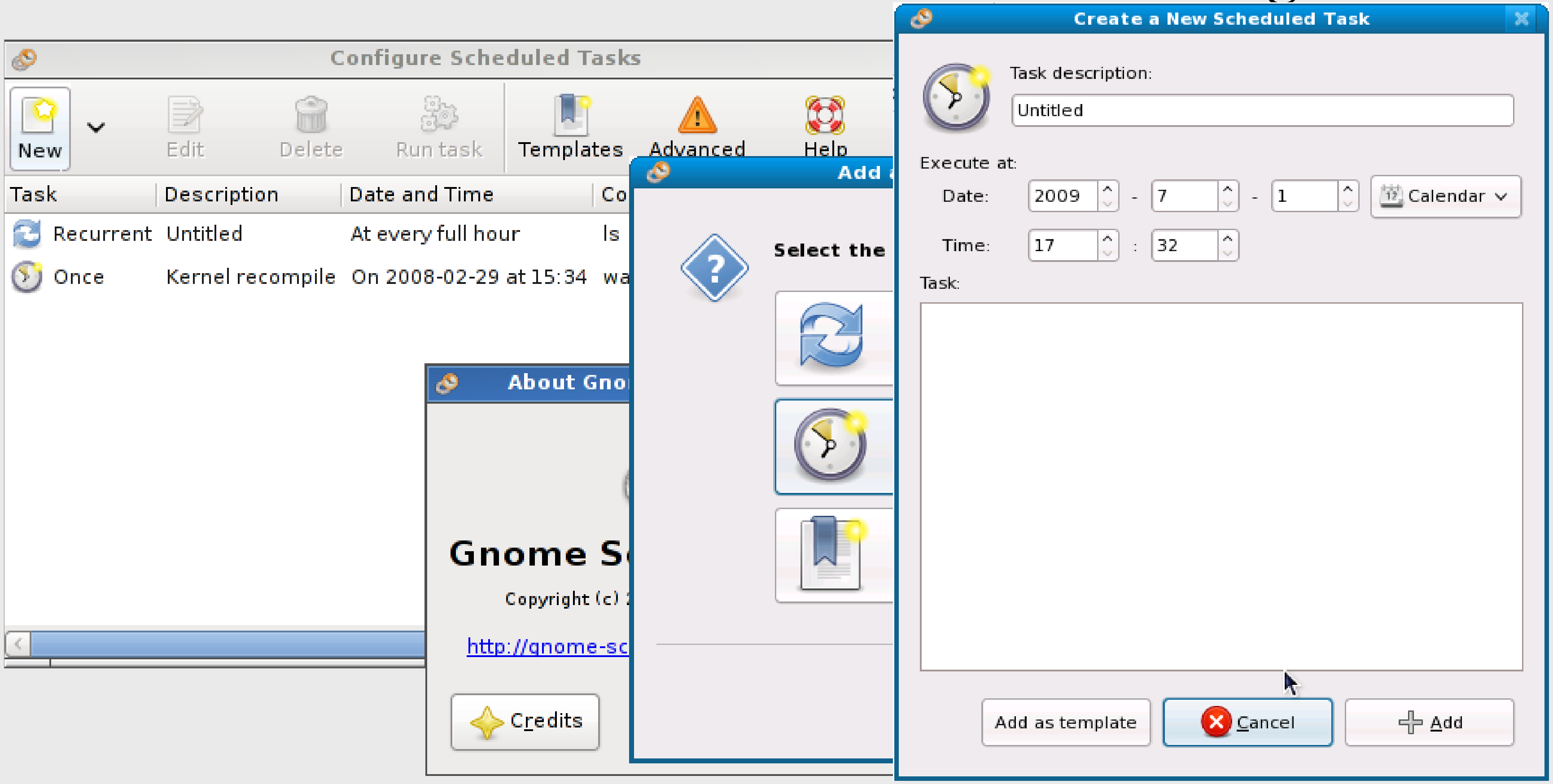
```
00-logwatch   00webalizer       0anacron
logrotate     makewhatis.cron   prelink
```

The System Cron Table

- Initial section of cron table specifies execution environment
- Remainder similar to user cron table entries
- Sixth field specifies who to run command as
- Remaining fields represent command to run
- **`run-parts`** command
  - Execute all files in a directory

cron and **at** Scheduling GUI

- Install Gnome Schedule to provide a GUI interface for **cron/at** task scheduling



Task	Description	Date and Time	Co
Recurrent	Untitled	At every full hour	ls
Once	Kernel recompile	On 2008-02-29 at 15:34	wa

Task description: Untitled

Execute at:

Date: 2009 - 7 - 1

Time: 17 : 32

Task:

Summary

- Processes are programs that are executing on the system
- User processes are run in the same terminal as the user who has executed them, whereas daemon processes are system processes that do not run on a terminal
- Every process has a parent process associated with it and, optionally, several child processes

Summary

- Process information is stored in the **/proc** filesystem
  - **ps**, **pstree** and **top** commands can be used to view this information
- Zombie and rogue processes that exist for long periods of time use up system resources and should be killed to improve system performance
- You may send kill signals using the **kill**, **killall**, and **top** commands

Summary

- The BASH shell forks a subshell to execute most commands
- The priority of a process may be affected indirectly by altering its NI or nice value
- Commands may be scheduled to run at a later time using the **`at`** & **`cron`** daemons

The End…

- Questions?