CHAPTER 16

System Resources

IN THIS CHAPTER

- System-Monitoring Tools
- Reference

System-Monitoring Tools

To keep your system in optimum shape, you need to be able to monitor it closely. Such monitoring is imperative in a corporate environment where uptime is vital and any system failures can cost real money. Whether it is checking processes for any errant daemons, or keeping a close eye on CPU and memory usage, Ubuntu provides a wealth of utilities designed to give you as little or as much feedback as you want. In this chapter, we look at some of the basic monitoring tools, along with some tactics designed to keep your system up longer. Some of the monitoring tools cover network connectivity, memory, and hard drive usage, but all should find a place in your sysadmin toolkit. Finally you will learn how to manipulate active system processes using a mixture of graphical and command-line tools.

Console-Based Monitoring

Those familiar with UNIX system administration already know about the `ps` or process display command commonly found on most flavors of UNIX. Because Linux is closely related to UNIX, it also benefits from this command and enables you to quickly see the current running processes on the system as well as who owns them and how resource-hungry they are.

Although the Linux kernel has its own distinct architecture and memory management, it also benefits from enhanced use of the `/proc` file system, the virtual file system found on many UNIX flavors. Through the `/proc` file system, you can directly communicate with the kernel to get a deep view of what is currently happening. Developers tend to use the `/proc` file system as a way of getting information

out from the kernel and for their programs to manipulate it into more human-readable formats. The `/proc` file system is beyond the scope of this book; but if you want to get a better idea of what it contains, head on over to http://en.tldp.org/LDP/Linux-Filesystem-Hierarchy/html/proc.html for an excellent and in-depth guide.

Processes can also be controlled at the command line, which is important because you might sometimes have only a command-line interface. Whenever an application or command is launched, either from the command line or by clicking on an icon, the process that comes from the kernel is assigned an identification number called a *process ID*, or *PID* for short. This number is shown in the shell if the program is launched via the command line:

```
$ xosview &
[1] 5918
```

In this example, the `xosview` client has been launched in the background, and the (`bash`) shell reported a shell job number (`[1]` in this case). A job number or job control is a shell-specific feature that allows a different form of process control, such as sending or suspending programs to the background and retrieving background jobs to the foreground. (See your shell's man pages for more information if you are not using `bash`.)

The second number displayed (`5918` in this example) represents the process ID. You can get a quick list of your processes by using the `ps` command like this:

```
$ ps
      PID     TTY     TIME          CMD
      5510    pts/0   00:00:00      bash
      5918    pts/0   00:00:00      xosview
      5932    pts/0   00:00:00      ps
```

Note that not all output from the display is shown here. As you can see, however, the output includes the process ID, abbreviated as `PID`, along with other information, such as the name of the running program. As with any UNIX command, many options are available; the `proc` man page has a full list. A most useful option is `aux`, which provides a friendly list of all the processes. You should also know that `ps` works not by polling memory, but through the interrogation of the Linux `/proc` or process file system. (`ps` is one of the interfaces mentioned at the beginning of this section.)

The `/proc` directory contains quite a few files—some of which include constantly updated hardware information (such as battery power levels and so on). Linux administrators often pipe the output of `ps` through a member of the `grep` family of commands to display information about a specific program, perhaps like this:

```
$ ps aux ¦ grep xosview
USER   PID    %CPU %MEM  VSZ   RSS  TTY    STAT START  TIME COMMAND
andrew 5918  0.3  1.1   2940  1412 pts/0  S    14:04  0:00 xosview
```

This example returns the owner (the user who launched the program) and the PID, along with other information, such as the percentage of CPU and memory usage, size of the command (code, data, and stack), time (or date) the command was launched, and the name of the command. Processes can also be queried by PID like this:

```
$ ps 5918
  PID TTY      STAT   TIME COMMAND
5918  pts/0    S      0:00 xosview
```

You can use the PID to stop a running process by using the shell's built-in `kill` command. This command asks the kernel to stop a running process and reclaim system memory. For example, to stop the `xosview` client in the example, use the `kill` command like this:

```
$ kill 5918
```

After you press Enter (or perhaps press Enter again), the shell might report

```
[1]+  Terminated              xosview
```

Note that users can `kill` only their own processes, but `root` can kill them all. Controlling any other running process requires `root` permission, which should be used judiciously (especially when forcing a `kill` by using the `-9` option); by inadvertently killing the wrong process through a typo in the command, you could bring down an active system.

Using the `kill` Command to Control Processes

The `kill` command is a basic UNIX system command. We can communicate with a running process by entering a command into its interface, such as when we type into a text editor. But some processes (usually system processes rather than application processes) run without such an interface, and we need a way to communicate with them, too, so we use a system of signals. The `kill` system accomplishes that by sending a signal to a process, and we can use it to communicate with any process. The general format of the `kill` command is as follows:

```
# kill option PID
```

A number of signal options can be sent as words or numbers, but most are of interest only to programmers. One of the most common ones you will use is this:

```
$ sudo kill PID
```

This tells the process with PID to stop; you supply the actual PID.

```
$ sudo kill -9 PID
```

is the signal for `kill` (`9` is the number of the `SIGKILL` signal); use this combination when the plain `kill` shown previously does not work:

```
$ sudo kill -SIGHUP PID
```

is the signal to "hang up"—stop—and then clean up all associated processes, too. (Its number is `-1`.)

As you become proficient at process control and job control, you will learn the utility of a number of `kill` options. You can find a full list of signal options in the `signal` man page.

Using Priority Scheduling and Control

Every process cannot make use of the system's resources (CPU, memory, disk access, and so on) as it pleases. After all, the kernel's primary function is to manage the system resources equitably. It does this by assigning a priority to each process so that some processes get better access to system resources and some processes might have to wait longer until their turn arrives. Priority scheduling can be an important tool in managing a system supporting critical applications or in a situation in which CPU and RAM usage must be reserved or allocated for a specific task. Two legacy applications included with Ubuntu are the `nice` and `renice` commands. (`nice` is part of the GNU `sh-utils` package, whereas `renice` is inherited from BSD UNIX.)

The `nice` command is used with its `-n` option, along with an argument in the range of `-20` to `19`, in order from highest to lowest priority (the lower the number, the higher the priority). For example, to run the `xosview` client with a low priority, use the `nice` command like this:

```
$ nice -n 12 xosview &
```

The `nice` command is typically used for disk- or CPU-intensive tasks that might be obtrusive or cause system slowdown. The `renice` command can be used to reset the priority of running processes or control the priority and scheduling of all processes owned by a user. Regular users can only numerically increase process priorities (that is, make tasks less important) using this command, but the root operator can use the full `nice` range of scheduling (`-20` to `19`).

System administrators can also use the `time` command to get an idea about how long and how much of a system's resources will be required for a task, such as a shell script. (Here, `time` is used to measure the duration of elapsed time; the command that deals with civil and sidereal time is the `date` command.) This command is used with the name of another command (or script) as an argument, as follows:

```
$ sudo time -p find / -name core -print
/dev/core
/proc/sys/net/core
```

```
real 1.20
user 0.14
sys 0.71
```

Output of the command displays the time from start to finish, along with the user and system time required. Other factors you can query include memory, CPU usage, and file system input/output (I/O) statistics. See the `time` command's man page for more details.

Nearly all graphical process-monitoring tools include some form of process control or management. Many of the early tools ported to Linux were clones of legacy UNIX utilities. One familiar monitoring (and control) program is `top`. Based on the `ps` command, the `top` command provides a text-based display of constantly updated console-based output showing the most CPU-intensive processes currently running. It can be started like this:

```
$ sudo top
```

After you press Enter, you will see a display as shown in Figure 16.1. The `top` command has a few interactive commands: pressing **h** displays the help screen; pressing **k** prompts you to enter the PID of a process to kill; pressing **n** prompts you to enter the PID of a process to change its nice value. The `top` man page describes other commands and includes a detailed description of what all the columns of information `top` can display actually represent; have a look at top's well-written man page.

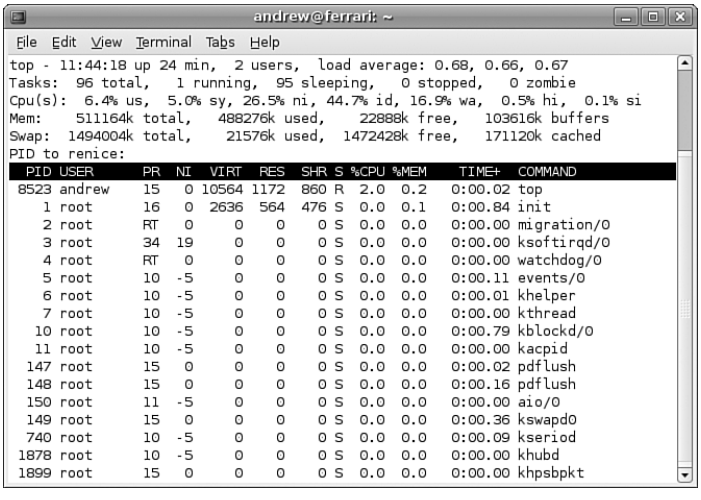

FIGURE 16.1 You can use the `top` command to monitor and control processes. Here, we are prompted to `renice` a process.

The `top` command displays quite a bit of information about your system. Processes can be sorted by PID, age, CPU or memory usage, time, or user. This command also provides process management, and system administrators can use its `k` and `r` keypress commands to kill and reschedule running tasks, respectively.

The `top` command uses a fair amount of memory, so you might want to be judicious in its use and not leave it running all the time. When you've finished with it, simply press **`q`** to quit `top`.

Displaying Free and Used Memory with `free`

Although `top` includes some memory information, the `free` utility displays the amount of free and used memory in the system in kilobytes. (The `-m` switch displays in megabytes.) On one system, the output looks like this:

```
$ sudo free
                    total     used      free     shared    buffers    cached
Mem:                516372    484972    31400    0         19816      317420
-/+ buffers/cache:  147736    368636
Swap:               433712          0   433712
```

This output describes a machine with 512MB of RAM memory and a swap partition of 444MB. Note that some swap is being used although the machine is not heavily loaded. Linux is good at memory management and "grabs" all the memory it can in anticipation of future work.

TIP

A useful trick is to employ the `watch` command; it repeatedly reruns a command every 2 seconds by default. If you use

```
$ sudo  watch free
```

you will see the output of the `free` command updated every 2 seconds.

Another useful system-monitoring tool is `vmstat` (*virtual memory statistics*). This command reports on processes, memory, I/O, and CPU, typically providing an average since the last reboot; or you can make it report usage for a current period of time by telling it the time interval in seconds and the number of iterations you desire, as follows:

```
$ sudo vmstat 5 10
```

The preceding command runs `vmstat` every 5 seconds for 10 iterations.

Use the `uptime` command to see how long it has been since the last reboot and to get an idea of what the load average has been; higher numbers mean higher loads.

Disk Quotas

Disk quotas are a way to restrict the usage of disk space either by user or by groups. Although rarely—if ever—used on a local or standalone workstation, quotas are definitely a way of life at the enterprise level of computing. Usage limits on disk space not only conserve resources, but also provide a measure of operational safety by limiting the amount of disk space any user can consume.

Disk quotas are more fully covered in Chapter 14, "Managing Users."

Graphical Process and System Management Tools

The GNOME and KDE desktop environments offer a rich set of network and system-monitoring tools. Graphical interface elements, such as menus and buttons, and graphical output, including metering and real-time load charts, make these tools easy to use. These clients, which require an active X session and (in some cases) root permission, are included with Ubuntu.

If you view the graphical tools locally while they are being run on a server, you must have X properly installed and configured on your local machine. Although some tools can be used to remotely monitor systems or locally mounted remote file systems, you have to properly configure pertinent X11 environment variables, such as `$DISPLAY`, to use the software or use the `ssh` client's `-X` option when connecting to the remote host.

A handy little application is the `xosview` client, which provides load, CPU, memory and swap usage, disk I/O usage and activity, page swapping information, network activity, I/O activity, I/O rates, serial port status, and if APM is enabled, the battery level (such as for a laptop). You will have to obtain `xosview` using either `synaptic` or `apt-get`.

For example, to see most of these options, start the client like this:

```
$ sudo xosview -geometry 406x488 -font 8x16 +load +cpu +mem +swap \
 +page +disk +int +net &
```

After you press Enter, you will see a display as shown in Figure 16.2.

The display can be customized for a variety of hardware and information, and the `xosview` client (like most well-behaved X clients) obeys geometry settings such as size, placement, or font. If you have similar monitoring requirements, and want to try a similar but different client from `xosview`, try `xcpustate`, which has features that enable it to monitor network CPU statistics foreign to Linux. Neither of these applications is installed with the base set of packages; you have to install them manually if you want to use them.

Some of the graphical system- and process-monitoring tools included with Ubuntu are as follows:

- `vncviewer`—AT&T's open-source remote session manager (part of the `Xvnc` package), which can be used to view and run a remote desktop session locally. This software (discussed in more detail in Chapter 19, "Remote Access with SSH and Telnet") requires an active, but background, X session on the remote computer.

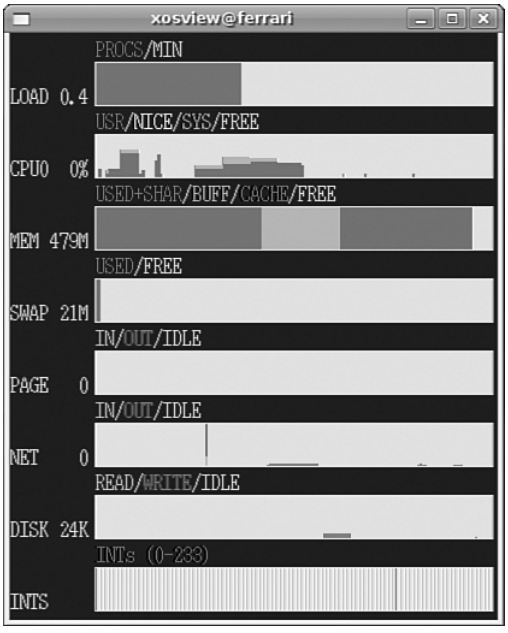

FIGURE 16.2 The `xosview` client displays basic system stats in a small window. You can choose from several options to determine what it will monitor for you.

- `gnome-nettool`—A GNOME-developed tool that enables system administrators to carry out a wide range of diagnostics on network interfaces, including port scanning and route tracing.
- `ethereal`—This graphical network protocol analyzer can be used to save or display packet data in real time and has intelligent filtering to recognize data signatures or patterns from a variety of hardware and data captures from third-party data capture programs, including compressed files. Some protocols include AppleTalk, *Andrew File System (AFS)*, AOL's Instant Messenger, various Cisco protocols, and many more.
- `gnome-system-monitor`—Replacing `gtop`, this tool is a simple process monitor offering two views: a list view and a moving graph. It is accessed via the System Tool menu selection as the System Monitor item (see Figure 16.3).

The System Monitor menu item (shown in Figure 16.3) is found in the System, Administration menu. You can launch it from the command line as follows:

```
$ gksudo gnome-system-monitor
```

From the Process Listing view (chosen via the tab in the upper-left portion of the window), select a process and click on More Info at the bottom left of the screen to display details on that process at the bottom of the display. You can select from three views to filter the display, available in the drop-down View list: All Processes, My Processes (those you alone own), or Active Processes (all processes that are active).

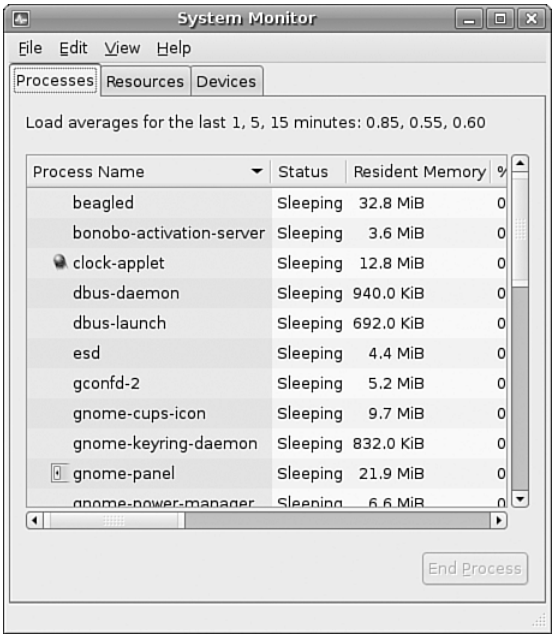

FIGURE 16.3 The Process Listing view of the System Monitor.

Choose Hidden Processes under the Edit command accessible from the top of the display to show any hidden processes (those that the kernel does not enable the normal monitoring tools to see). Select any process and kill it with End Process.

The processes can be `reniced` by selecting Edit, Change Priority. The View selection from the menu bar also provides a memory map. In the Resource Monitor tab, you can view a moving graph representing CPU and memory usage (see Figure 16.4).

KDE Process- and System-Monitoring Tools

KDE provides several process- and system-monitoring clients. The KDE graphical clients are integrated into the desktop taskbar by right-clicking on the taskbar and following the menus.

These KDE monitoring clients include the following:

- `kdf`—A graphical interface to your system's file system table that displays free disk space and enables you to mount and unmount file systems using a pointing device.
- `ksysguard`—Another panel applet that provides CPU load and memory use information in animated graphs. `ksysguard` enables you to monitor a wide range of measurable events, as shown in Figure 16.5

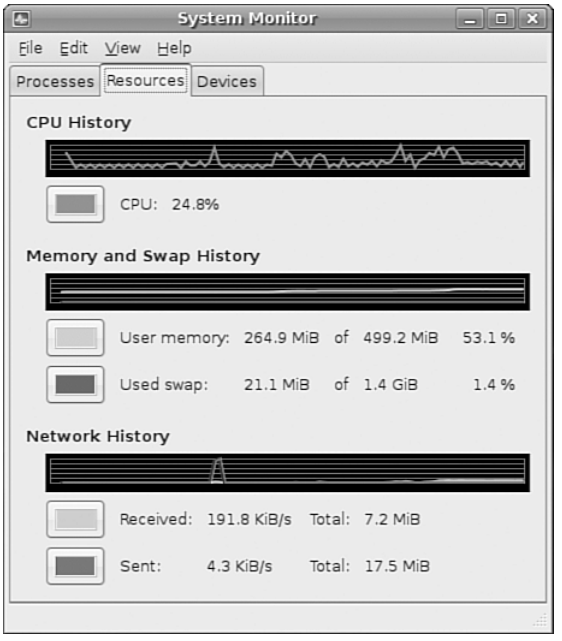

FIGURE 16.4 The Graph view of the System Monitor. It shows CPU usage, memory/swap usage, and disk usage. To get this view, select the Resource Monitor tab.

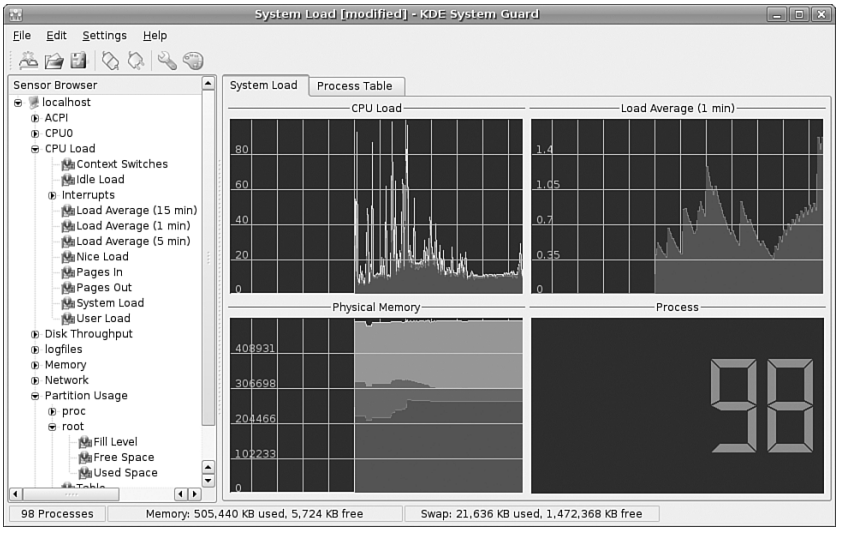

FIGURE 16.5 `ksysguard` can monitor all sorts of events on your system. Here we can see CPU load, memory availability, and number of active processes.

Reference

http://and.sourceforge.net/—Home page of the `and` auto-`nice` daemon, which can be used to prioritize and reschedule processes automatically.

http://sourceforge.net/projects/schedutils/—Home page for various projects offering scheduling utilities for real-time scheduling.

http://freetype.sourceforge.net/patents.html—A discussion of the FreeType bytecode interpreter patents.

http://www.ethereal.com/—Home page for the Ethereal client.

http://www.realvnc.com/—The home page for the Virtual Network Computing remote desktop software, available for a variety of platforms, including Ubuntu. This software has become so popular that it is now included with nearly every Linux distribution.