

ProcessManagement < Webmin < TWiki

Process Management

This page explains how to manage running processes on your system using Webmin.

Introduction to processes

Every program, process or command running on a Linux system is a *process*. At any time, there are dozens of processes running on your system, some for programs that you are interacting with graphically, some for commands that you have started at a shell prompt, some for servers running in the background and some that perform system tasks. Every time you type a command like `ls` or `vi` at the shell prompt, a new process is created, only to exit as soon as its job is done.

Each process is identified by a unique ID, known as the PID or process ID. Each is owned by a single user and is a member of multiple groups, which determine the privileges that the process has. And each has a priority (also known as the nice level), which controls how much CPU time the process can use up on a busy system. Almost every process has a parent, which is the process that started it, and from which it inherits ownership, priority and other settings.

A process will run until it chooses to exit, or until it is killed by a signal from another process.

The Running Processes module

This module can be used to view, kill, re-prioritize and run processes on your system. When you enter it for the first time from the System category, the main page will display a tree of processes as shown below.

The screenshot shows the 'Running Processes' module in Webmin. The sidebar on the left contains a tree view of system categories, with 'Running Processes' selected. The main content area displays a table of running processes. The table has columns for Process ID, Owner, Started, and Command. The process with PID 459 is highlighted in yellow.

Process ID	Owner	Started	Command
1	root	2006	init [3]
2	root	2006	[keventd]
3	root	2006	[ksoftirqd_CPU0]
4	root	2006	[ksoftirqd_CPU1]
5	root	2006	[kswapd]
6	root	2006	[bdflush]
7	root	2006	[kupdated]
8	root	2006	[mdrecoveryd]
68	root	2006	[khubd]
201	root	2006	[kjournald]
202	root	2006	[kjournald]
204	root	2006	[kjournald]
205	root	2006	[kjournald]
300	root	2006	[kjournald]
459	root	2006	/sbin/dhclient eth1
803	root	May13	syslogd -r
852	root	May13	klogd
913	nobody	2006	rpc.portmap -u nobody -r /var/lib/portmap -f/register
917	root	2006	inetd -D /etc/inet.d
30282	nobody	May08	rpc.rusersd
30283	nobody	May08	rpc.rusersd
30284	nobody	May08	rpc.rusersd
30285	nobody	May08	rpc.rusersd

The Running Processes module tree display

The module has several different ways of viewing all the processes on your system, selectable by the **Display** links at the top of the main page. They are :

- **PID** In this display mode each process is shown indented under its parent, forming a tree of all the processes running on your system. At the top of the tree is the init command, which is started by the kernel at boot time and so has no parent.
- **Memory** In this mode, processes are ordered by the amount of memory they are using up, with those using the most memory shown at the top of the page. A processes memory usage is not always indicative of the amount of memory it is really using, because processes often share memory with each other. In addition, the total and free amount of real and virtual memory on your system is displayed above the process list.
- **CPU** This display mode orders processes by their current CPU usage, with the heaviest user appearing first. Sometimes the Webmin command that generates the page will appear near the top of the list, but it can be safely ignored. The system load averages will be displayed at the top of the page, to give some idea of how busy the system has been over the last 1,5 and 10 minutes. An average of 0 means no activity at all, 1 means the CPU is fully utilized, and anything above 1 means that there are more processes wanting to run than the system has CPU time for.

The **Search** and **Run** options are for searching for processes and running new ones, respectively. See the sections below for more details.

Viewing, killing or re-prioritizing a process

You can see the full details of any running process by clicking on its **Process ID** column entry in any of the displays on the main page. This will take you to the process information page, shown in this screenshot.

The screenshot shows the Webmin interface for the 'Process Information' page. On the left is a navigation tree with categories like 'System', 'Servers', and 'Networking'. The main content area is titled 'Process Information' and shows details for the 'init [3]' process. The details are organized into two columns:

Command	init [3]	Parent process	None
Process ID	1	CPU	0.0 %
Owner	root	Run time	00:01:59
Size	1304 kB	Real user	root
Nice level	0 (Default) <input type="button" value="Change"/>	Started	2006
Group	root	Real group	root
Process group ID	0		
TTY	None		

Below the details are control buttons: 'Send Signal' (with a dropdown menu showing 'HUP'), 'Terminate', 'Kill', 'Suspend', 'Resume', 'Trace Process', and 'Files and Connections'.

The 'Subprocesses' section lists several child processes:

2	[keventd]
3	[ksoftirqd_CPU0]
4	[ksoftirqd_CPU1]
5	[kswapd]
6	[bdflush]
7	[kupdated]
8	[mdrecoveryd]
68	[khubd]
201	[kjournald]
202	[kjournald]
204	[kjournald]

Detailed information on a process

The page displays all available information about the process, including its full command line, parent command and any sub-processes. You can just to the information page for the parent by clicking on its command, or to the page on any of the sub-processes by clicking on its process ID. A list of files that the process has open and network connections that it is currently using can be viewed by clicking the **Files and Connections** button.

The process can be stopping using a TERM signal by clicking the **Terminate Process** button. Because this can be ignored by some commands, the **Kill Process** button can be used to send a KILL signal if the termination fails. Unless the process is hung inside a kernel system call, killing it is guaranteed to succeed.

Other signals can be sent by selecting the type of signal next to the **Send Signal** button before pressing it. Some of the more useful signals are :

The information page can also be used to change the nice level of a running process, giving it a higher or lower priority. To change a processes priority, select a new level from the *Nice level* list, and then click the **Change** button. Lower levels mean higher priorities, so a process with a nice level of 10 will get more CPU time than one with level 5.

On a system with multiple users, long-running processes that take up a lot of CPU time should be given a higher nice level so that they do not slow down processes that are interacting with users. Alternately, you can speed up a process at the expense of others by giving it a lower nice level. You should be careful when setting an extremely low level (such as 20) as all other processes may become starved of CPU time, making the system unresponsive.

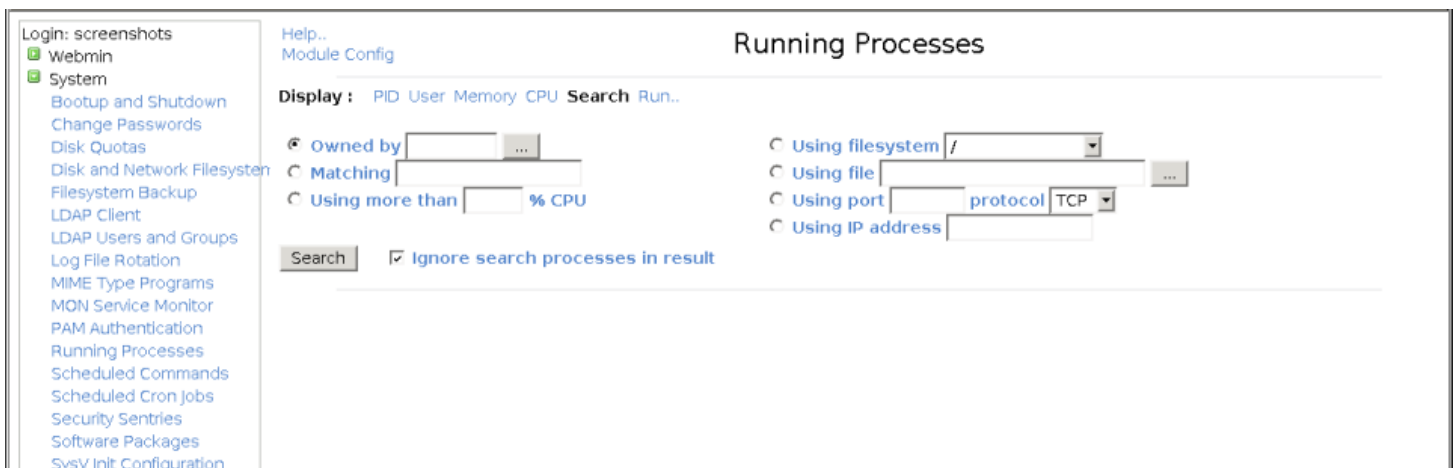
Searching for processes

If you have a large number of processes running on your system and want to find one or more to kill or view, the Running Process module's search feature makes it easy. To find processes, follow these steps :

1. On the main page of the module, click on the **Search** display mode link. This will take you to a search form as shown in the screenshot below.
2. The form shows several different criteria for finding processes, of which you can choose one by selecting the radio button next to it. The criteria are :
 - **Owned by** Processes owned by the user whose name you enter next to this option will be found.
 - **Matching** Finds processes whose command or arguments contains the text that you enter next to this option.
 - **Using more than** Finds processes using more than the specified percentage of CPU time.
 - **Using filesystem** Processes whose current directory is on the chosen filesystem or are accessing any file on it will be found. Useful if you cannot un-mount a filesystem because it is busy. * **Using file** Finds processes that have the entered file open for reading or writing. If you enter a directory, any process that has it as its current directory will be found. * **Using port** Finds processes that are sending, receiving or listening for network traffic on the

entered port using the chosen protocol. Useful if you know the port number a server is listening on, and want to find the server process. * **Using address** Finds processes that have a network connection open to the entered address, or are listening on that address if it is for an interface on your system.

3. To filter the Webmin search processes from the results, select the **Ignore search processes in result** option. This can be useful when searching by CPU usage, as the Webmin processes use up a lot of CPU time.
4. After you have select the search criteria, click the **Search** button. Any matching processes will be displayed below the form.
5. If you want to see additional information about a process, change its priority or send it alone a signal, click on its **Process ID** in the results.
6. To kill all matching processes, click the **Terminate Processes** or **Kill Processes** button. You can also send any signal to all processes by selecting it from the list next to the **Send Signal** button. A page will be displayed listing each process ID and whether it was signaled or killed successfully.



The screenshot shows the 'Running Processes' module in Webmin. On the left is a navigation menu with items like 'Login: screenshots', 'Webmin', 'System', 'Bootup and Shutdown', etc. The main area has a title 'Running Processes' and a 'Display:' section with options for 'PID', 'User', 'Memory', 'CPU', 'Search', and 'Run..'. Below this are search criteria: 'Owned by' (with a dropdown), 'Matching' (with a text input), 'Using more than' (with a text input and '% CPU' label), 'Using filesystem' (with a dropdown), 'Using file' (with a text input), 'Using port' (with a text input, 'protocol' dropdown, and 'TCP' dropdown), and 'Using IP address' (with a text input). There is a 'Search' button and a checked checkbox for 'Ignore search processes in result'.

The process search form

Running a process

The module can also be used to run simple commands, either in the foreground so that their output is displayed, or in the background as daemons. This can be useful if you just want to run a command without having to login via telnet or SSH (or if a firewall is preventing a telnet or SSH login). The steps to follow are :

1. On the main page of the module, click on the **Run** link next to the display mode options. This will take you to the form for starting a new process.
2. Enter the command that you want to run into the **Command to run** field. Shell operators and special characters like `;`, `<`, `>` and `&&` can be used.
3. If the command is something that will take a long time to run, you can set the **Run mode** option to **Run in background** to have Webmin automatically put it in the background. However, if you want to see the output from the command, leave the option set to **Wait until complete**.

4. Enter any input that you want fed to the command into the **Input to command** field.
5. Click the **Run** button to run it. If the **Wait until complete** option was selected, any output from the command will be displayed.

Module access control options

By default, any Webmin user with access to this module can manage all processes running on the system, as though he were logged in as root. However, using the Webmin Users module you can limit a user's access so that he can only kill or re-nice processes owned by a particular Unix user. It is also possible to restrict a user to read-only mode, allowing him to only see processes by not change them in any way or start new ones.

You should read chapter 52 first to learn more about module access control and how to grant a user access to the Running Processes module. Once that is done, to edit a Webmin user's access to this module, the steps to follow are :

1. In the Webmin Users module, click on Running Processes next to the name of the user or group that you want to restrict.
2. Change the **Can edit module configuration?** field to **No**.
3. To give the Webmin user access to only those processes owned by a particular Unix user, enter the username into the **Manage processes as user** field. If the Unix and Webmin users have the same name, you can select **Current Webmin user** instead. This can be useful when setting up module access control for a group in which you want each member to be able to manage only his own processes.
4. To put the user into read-only mode, set the **Can kill and renice processes?** and **Can run commands? *fields to *No**. If this is done, it doesn't really matter what username you enter in step 7 because no process management can be done.
5. Click the **Save** button to have your changes activated.

To restrict the processes that a Webmin user can manage, the module code simply switches to run as the Unix user specified in step 4. Because a Unix user cannot kill or re-prioritize any process that he does not own, switching user like this causes the operating system to automatically enforce process access control for Webmin.

Other operating systems

Because processes exist on all versions of Unix with almost identical attributes, this module appears almost exactly the same on all supported operating systems.

When viewing detailed information about a process, different information may be available on other operating systems. The range of nice levels may also be different, but lower levels still mean a higher priority and vice-versa.

When searching for a process, the **Using filesystem**, **Using file** or **Using port** criteria may not be available. These options depend on the `fuser` and `lsof` commands that are not available for or installed by default on all systems.