LINFO

# PID Definition

A PID (i.e., *process identification number*) is an identification number that is automatically assigned to each *process* when it is created on a Unix-like operating system.

A process is an *executing* (i.e., running) instance of a program. Each process is guaranteed a unique PID, which is always a non-negative integer.

The process *init* is the only process that will always have the same PID on any session and on any system, and that PID is 1. This is because init is always the first process on the system and is the ancestor of all other processes.

A very large PID does not necessarily mean that there are anywhere near that many processes on a system. This is because such numbers are often a result of the fact that PIDs are not immediately reused, in order to prevent possible errors.

The default maximum value of PIDs is 32,767. This maximum is important because it is essentially the maximum number of processes that can exist simultaneously on a system. Although this will almost always be sufficient for a small system, large servers may require many more processes. The lower the maximum value, the sooner the values will wrap around, meaning that lower values do not necessarily indicate processes that started to run earlier.

The file *pid_max*, which was introduced with the Linux 2.5 kernel, specifies the value at which PIDs wrap around (i.e., the value in this file is one greater than the maximum PID). It has a default of 32768, but it can be set to any number up to approximately four million. The maximum number of processes on a system is only limited by the amount of physical memory (i.e., RAM) available.

The PIDs for the processes currently on the system can be found by using the *ps* command or the *pstree* command (which shows the process names and PIDs in a tree diagram). The *top* command also shows the PIDs of currently running processes along with other information about them, but it differs in that it continuously updates the information. The *pidof* command provides the PID of a program whose name is passed to it as an *argument* (i.e., input).

The PID is needed in order to terminate a frozen or otherwise misbehaving program with the *kill* command. This command makes it possible to end a program that cannot otherwise be stopped except by rebooting (i.e., restarting) the system, and it is thus an important element in the stability and robustness of Unix-like operating systems.

Information on current processes is stored in the */proc* filesystem. This filesystem consists of kernel data that changes in *real time* (i.e., sensing and responding to external events nearly simultaneously). It is easy to extract the information contained therein using commands such as cat.

Listing the contents of /proc with the *ls* command as follows will show numerous directories whose names consist only of numbers:

```
ls /proc | less
```

It is convenient to *pipe* (i.e., transfer) the output from *ls /proc* to the *less* command because such output can be fairly long and less allows it to be read one screenful at a time.

There is a numbered directory in /proc corresponding to each PID currently on the system. Each of the directories contains several standard files containing information about the process. For example, the file *cmdline* contains the name of the command (along with any options and arguments) that the process was started with, and it can be easily read with the cat or head command. For instance, the cmdline file for PID 1 can be read as follows:

```
cat /proc/1/cmdline
```

On most systems it will be necessary to use the *root* (i.e., administrative) account, which can be accessed by using the *su* (i.e., *substitute user*) command, to read files in the /proc directory.

More extensive information can be found about any PID and the process it represents by reading the PID's *status* file, which is also found in its directory in /proc. For example, the following would display the contents of the status file for PID 1:

```
cat /proc/1/status
```