The Executable Path

In the last section we examined the file /bin/cp, which is Linux's copy program. You can call the program like this:

sudo cp /etc/group /etc/group_backup

This creates a backup of your group file. You need the prefix →*sudo*("superuser do") because only root is allowed to write to the /etc directory where the file group_backup is saved. Now the question is: how does Linux know where to find the *cp* program? The above example works regardless of your current working directory. You can call *cp* from any directory without specifying the full program path /bin/cp (although you can do that if you wish).

Linux knows that calls to *cp* are meant to execute the program /bin/cp because the directory /bin is part of the system's executable path. This is not a path in the regular sense. It's a collection of directories that are searched automatically when Linux looks for executable programs. /bin is part of the executable path, that's why calls to *cp* succeed.

You can view the contents of your executable path by entering:

echo $PATH

PATH is a shell environment variable. The prefixed dollar sign ($) tells *echo* to output the contents of the PATH variable rather than the string "PATH". Let's have a look at a typical value of PATH:

/usr/local/bin:/usr/bin:/bin

This means that when you enter *cp* and press Enter, the directories /usr/local/bin, /usr/bin and /bin are searched for an executable file called *cp*, which is then executed. Linux uses the executable path mechanism to spare you from typing long command names and for security reasons. On a Linux server it would be easy to put a malicious, counterfeit version of *cp*into a user's home directory and wait until he calls *cp*!

If you want to execute a program that resides in a non-path directory, you have to specify its full path, such as:

/home/myuser/myprogram

If you're currently working in /home/myuser, you can also enter:

./myprogram

The dot (.) is a shorthand for the current working directory. To conclude this section, let us note that root has his own executable path. You can display it using this little hack:

echo `echo $PATH' | sudo bash

You'll notice that root's path contains some directories named sbin, which stands for "superuser binaries".