shortcut

Your Short Cut to Knowledge

CLI for Noobies

A Primer on the Linux Command Line

Joe Barr

SOURCEFORGE®

Community Press

www.prenhallprofessional.com

PRENTICE HALL
PEARSON EDUCATION

Visit us on the Web: www.prenhallprofessional.com

ISBN-13: 978-0-13-2361705
ISBN-10: 0-13-236170-1

Second release, August, 2007

CHAPTER 21
Environmental Issues

One of the things the CLI provides that you just can't get with the GUI is the ability to pop the hood open and take a look underneath. Maybe even tweak a thing or two, if need be.

In this chapter, we're going to take a look at the Bash shell environment, not only to see what's there, but also to learn how to change it.

Taking Charge of Our Environment!

The shell in which your CLI resides has an environment of its own. It includes a lot of information used by the shell and by applications you run in it.

We're talking about variables containing program names, arguments, and paths. To get a better idea of what sort of information the environment contains, type ***env*** at the command prompt and press Enter.

Here's what the first few lines of my environment looks like:

```
LESSKEY=/etc/lesskey.bin
NNTPSERVER=news
INFODIR=/usr/local/info:/usr/share/info:/usr/info
MANPATH=/usr/local/man:/usr/share/man:/usr/X11R6/man:/opt/gnome/share/man
SSH_AGENT_PID=6004
HOSTNAME=linux
DM_CONTROL=/var/run/xdmctl
GNOME2_PATH=/usr/local:/opt/gnome:/usr
XKEYSYMDB=/usr/X11R6/lib/X11/XKeysymDB
```

```
DESKTOP_STARTUP_ID=
HOST=linux
```

The main purpose of the ***env*** command is not to tell us everything in the environment, that's just a side benefit. It's really to allow us to modify the environment for the sake of a single process, and only that process.

Everything else sees the normal environment. You can set, modify, or unset a variable name with the env command.

Browsing through the output of the naked ***env*** command, you can spot all sorts of important items: various paths are defined, the locations of man pages provided, the system host name is given, terminal program is specified, the name of the shell to be run, and much more.

Remember PS1? The CLI prompt definition we mod'ed and colorized in the previous chapter? It's there.

So is your USER name, and your EDITOR, and the HISTSIZE, which controls how many lines of command history will be available to you to scroll back through. My environment contains 79 different variables.

The ***env*** command also allows you to define, set the value for, or remove variables from the environment, prior to execution of a command. *But it's only a temporary addition, change, or removal.*

Let's play with that. We'll create a new variable called FUNNYNAME and set it to mister.hawg, then stack a second command on the first using the magic semicolon, and grep for funny. Here's how that looks on my system:

```
linux:~> env FUNNYNAME=mister.hawg;env | grep FUNNY FUNNYNAME=mister.hawg
```

If I immediately repeat the naked ***env*** with the ***pipe*** to ***grep***, FUNNYNAME has disappeared. Like a butterfly, its time was short but sweet.

Set Down a Moment

The ***set*** command is a better way to get a list of shell variables. Entered naked—without any arguments—***set*** will produce a list of all existing shell variables *sorted by variable name,* which can make finding a particular variable a much easier chore.

What if you want to make the change “stick” for longer than a single moment? Easy. Use the ***export*** command. Let’s change the value of FUNNYNAME to something else, like this:

```
linux~>export FUNNYNAME=w.hawg
```

Now when I feed grep the output of env looking for something FUNNY, I get:

```
FUNNYNAME=w.hawg
```

$talks

Of course, the real reason for having these variables in the first place is to make the information available to processes running on your system. A script or command can retrieve the value of a variable simply by putting a $ character in front of its name. Let’s replace the grep above with something a little more elegant. How about this?

```
echo $FUNNYNAME
w.hawg
```

The change in the environment of the export command entered at the CLI above will remain in effect until it is changed again or the session ends, whichever comes first.

The environment variables are loaded when the shell is opened, so if you want the change to be permanent, you need to take another approach.

I'm using SUSE as I write this, so I can place any environment modes I want to make in the *.bashrc* file in my home directory. That's because SUSE sources your *$HOME/.bashrc* in */etc/profile*.

Your milage may vary. The distribution you're running might not source the *.bashrc* file, or it may favor your *.profile* file (also on the home directory) instead.

Naturally, the files in you /home directory apply only to you. If you want to make an environment change for all users on the system—like admins often do—you would make them in /etc/profile instead of in either file in the home directory.

Wherever you decide to place the commands, you can do it as shown above with the value being provided as an argument to the export command, or you can split it up, assigning the value on one line and exporting it on a following line, like this:

```
FUNNYNAME=mister.hawg
export FUNNYNAME
```

Unset That Variable

One last item—the ***unset*** command. Using it, you can make the variable simply disappear from the environment:

```
unset GLOBALWARMING
```

OK, there you have it. Refer to the ***man*** pages for the ***env, set, unset***, and ***export*** commands to go deeper into each.

Be sure to bring up your adventures in the wilds of the Bash shell environment the next time you're in the company of Windows users of the opposite gender. They're impressed by that kind of talk.

CHAPTER 22

The Lowdown on Top

Top has lots of secrets to tell. So many, in fact, that I have to squeeze the font size of its output as small as I can make it in order for it to display legibly on the pages of this book. My apologies to all the sighting-impaired.

Let's start with the basics. Just enter ***top*** at a command line and see what it has to say. Here's the first 12 lines of output on my desktop box:

```
top - 22:28:20 up 1 day,  3:11,  2 users,  load average: 0.21, 0.11, 0.09
Tasks: 103 total,   1 running, 102 sleeping,   0 stopped,   0 zombie
Cpu(s):  4.9% us,  1.0% sy,  0.0% ni, 93.8% id,  0.0% wa,  0.3% hi,  0.0% si
Mem:   1034760k total,  1014380k used,    20380k free,    50104k buffers
Swap:   554232k total,     4200k used,   550032k free,   442440k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
18583 warthawg  15   0 23660  13m  18m S  3.3  1.3   0:01.24 gnome-terminal
 5094 root      15   0  113m  45m  80m S  2.0  4.5  38:42.16 X
    6 root       5 -10     0    0    0 S  0.3  0.0   1:57.01 kacpid
 6028 warthawg  16   0 31428  17m  19m S  0.3  1.8   0:12.91 nautilus
18592 warthawg  17   0  1968  956 1764 R  0.3  0.1   0:00.03 top
```

As you can see, the first five lines provide a detailed status of the system, including time since the last boot, number of tasks, number of zombies, CPU, and memory utilization.

Let me get this plug in early. The man pages for top are exceptionally well written. Refer to them early and often as you experiment with the utility.

Those five lines make up the summary section. The real fun comes in the command line (not shown above) and the task section, which includes the column titles and the data displayed beneath them. If you're a tweaker, you're going to love ***top***. You can control almost everything about it.

Here are a few of the command-line options available to you with ***top***.

Option	Action
-b	Turns batch mode on. In batch mode, no keyboard input is accepted.
-c	Flip-flops command line/program name column. Whatever was shown the last time is not shown this time.
-d ss.tt	Sets the delay time for updating the screen in seconds and tenths of seconds.
-n x	Sets the number of times top should update the screen before shutting itself off.
-u user	Shows only the tasks run by selected user. A capital -U does the same thing, only for real. No tasks run by somebody else setting the effective user name to the selected user are shown.
-p PIDs	Show only the PIDs (up to 20 of them) specified.

The command-line options are fine, but top doesn't really get to be fun until you get familiar with the dynamic command set, entered at the keyboard while it's running.

Top Topics

There are 26 different columns of information about a task that top can display. Let's start by listing what's being displayed at a given point in time. Enter an "f" (or an "o") in the terminal window where top is