Make yourself at home

EASY SETUP

Environmental variables give users the ability to set up the command line to suit their own needs.

BY HEIKE JURZIK

If you enter a command like *ls* in the shell, the shell knows that you want to call the */bin/ls* program. Bash knows where on the filesystem to look for executables, so there is no need to type in the full path. This path is defined in what is known as an environmental variable. Variables of this kind are also used to modify the appearance of the shell prompt, set the time zone and the user's home directory, and other things.

Note that because bash (the "Bourne Again Shell") has become the default shell on Linux, all of the examples in this article assume bash as the command-line interpreter.

An environmental variable has three parts: the name, an assignment operator (=), and a value:

```
variable=value
```

The name can include upper- and lowercase letters, underlines, and numbers. The only restriction is that a table can't start with a number. If the value of the variable contains blanks or nonstandard characters, you have to double-quote your input:

```
LESS="-X"
```

Every user on a Linux system has a place he or she can call home; bash provides a fully furnished environment for every single account, but the appearance depends on the distribution you use. To discover which variables your home uses, type *env* (for "environment"). Listing 1 shows an example.

Some of the variables in Listing 1 can be set by the user. Other shell variables are predefined but can be modified as needed. The convention is to use uppercase letters. Table 1 gives an overview of common variables for bash.

THE AUTHOR

Heike Jurzik studied German, Computer Science and English at the University of Cologne, Germany. She discovered Linux in 1996 and has been fascinated with the scope of the Linux command line ever since. In her leisure time you might find Heike hanging out at Irish folk sessions or visiting Ireland.

Figure 1: You can change the variable temporarily without worrying about the aftereffects.

The *echo* command allows you to discover the value of a variable. The command outputs everything to its right place at the command line:

```
huhn@asteroid:~$
echo Huhu
Huhu
```

In addition to this, *echo* displays the value of the variable if you type a dollar sign in front of the variable's name:

```
huhn@asteroid:~$
echo $LANG
de_DE@euro
huhn@asteroid:~$
echo $PS1
\u@\h:\w\$
```

TIP

Instead of using two separate commands (e.g., *LESS="-X"; export LESS*), you can define and export in one fell swoop: *export LESS="-X"*.

GLOSSARY

Builtin: An abbreviation for "built-in command." Commands like this are built in to the shell, and you do not need to run a special program to use them.

The shell replaces the variables *$LANG* and *$PS1* with their respective values and then calls *echo* to output the text.

Even if you have everything set up just the way you like it, it is nice to have a change from the daily grind. You might want to refresh your language skills, so you could use the following command line to launch the Firefox browser in French, even though your environment normally speaks English:

```
LANG=FR firefox
```

Figure 1 shows how to temporarily move to a different language environment. Before calling the program, the example shows how to output a list of the locales your machine supports with the *locale -a* command.

Listing 1: env Example

```
01 huhn@asteroid:~$ env
02 TERM=xterm
03 SHELL=/bin/bash
04 USER=huhn
05 LS_COLORS=no=00:fi=00:...
06 PATH=/usr/local/bin:/usr/
   bin:/bin:/usr/bin/X11:/usr/
   games:/home/huhn/bin
07 IRCSERVER=irc.freenode.net
08 LANG=de_DE@euro
09 ...
```

Change

As you have seen, you can easily change a variable by entering its name, the assignment operator, and the value. Variables defined in this way are only valid for the current terminal session. To export the settings, you need the *export* command. This makes the settings accessible to child processes and subshells.

For example, you can set the *IRCNICK*, *IRCNAME*, and *IRCSERVER* variables for the current bash to tell command line-based IRC clients such as Irssi or BitchX which nick and name to use to automatically log on to a specific IRC server. If you then launch a second shell from the first one and try to output the values of the new variables, you won't see a thing. Quit the second shell by typing *exit* (Ctrl + D), and export the new variables. Now open a new shell from the first one, and you will see that it has inherited variables, which are now available in the subshell (Figure 2).

Figure 2: The export command is used to export new variables to subshell environments.

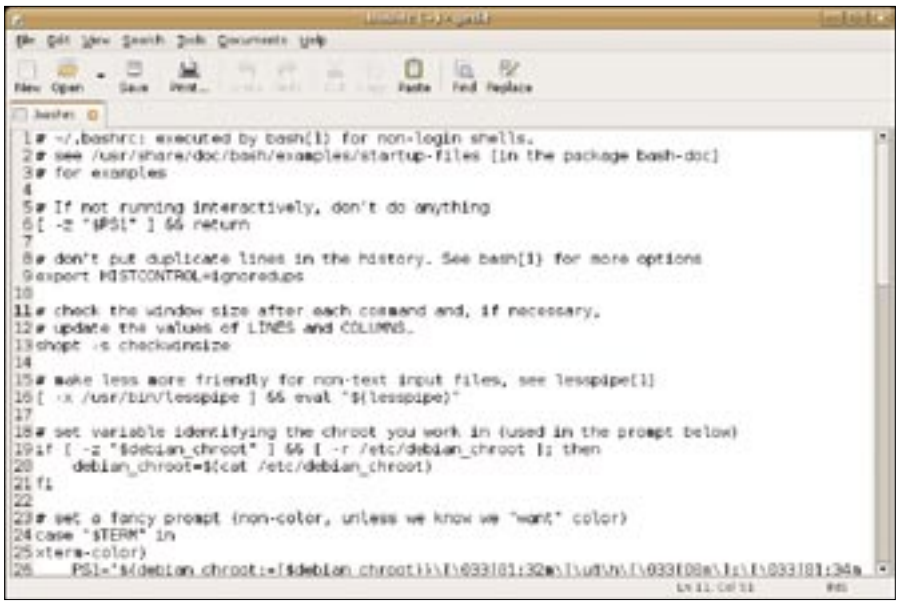

Figure 3: The bashrc file contains configuration settings for the bash environment.

The *export* **builtin** has more tricks up its sleeve. In combination with the -p option, or when called without any parameters, it displays a list of all exported variables. The -n option deletes a variable from the list. Listing 2 gives a couple of examples.

Any variables that you *export* would only be effective in the current shell and its child processes. To set up a variable permanently, you need to modify your bash configuration file and reload your environment. To do so, add your new *export* commands to one of the bash startup files, such as *~/.bashrc (Figure 3)*, and then parse the file once you've finished your home improvements:

```
source ~/.bashrc
```

Listing 3 shows a couple of practical environmental variables, including comments, that should make you feel right at home. ■

Table 1: Standard Bash Environment Variables

Variable	Meaning
CDPATH	Search path for the *cd* command
EDITOR	Standard text editor
HISTFILE	History file (e.g., *~/.bash_history*)
HISTSIZE	Maximum number of commands in history
HOME	Home directory for the user account
HOSTNAME	Host name
LANG	Language for program output, such as the date format, etc., assuming none of the *LC_* variables (see below) have been set. You can output a listing of all defined language variables by giving the *locale* command.
LC_ALL	Country setting, such as *C* or *de*; this overrides *LANG* and any other *LC_* variables.
LC_MESSAGES	Language for program and error messages
LC_TIME	Time format
LOGNAME	Login name of user
MAIL	Path to the user's mailbox (incoming mail)
MANPATH	Search path for man pages
PATH	Search path for executables
PS1	Default appearance of the shell prompt. For non-privileged users, this is typically *\u@\h:\w\$* (*huhn@asteroid:~$*); for the administrator, this is typically *\h:\w\$* (*asteroid:~#*).
PWD	Name of the current directory ("print working directory")
SHELL	Full path name of the current shell (*/bin/bash*)
TERM	Terminal settings, such as *xterm* or *vt100*
TZ	Time zone, such as *CET* or *MET*

Listing 2: export Example

```
01 huhn@asteroid:~$ export
02 declare -x CHARSET="latin1"
03 declare -x LS_COLORS="no=00:
   fi=00:..."
04 declare -x IRCSERVER="irc.
   freenode.net"
05 ...
06 huhn@asteroid:~$ export -n
   IRCSERVER
```

Listing 3: Environmental variables

```
01 # prevents "less" from leaving
   an empty
02 # screen at the end of the
   output:
03
04 export LESS="-X"
05
06 # adds a touch of color: user
   prompt in
07 # green, root prompt in red:
08
09 if [ $(id -u) = 0 ] ; then
10         COLOR1='\[\
   033[00;31m\]'
11 else
12         COLOR1='\[\
   033[01;32m\]'
13 fi
14
15 COLOR2='\[\033[00;33m\]'
16 COLOR3='\[\033[00;37m\]'
17 PS1=$COLOR1'[\u@\h'$COLOR2' \
   W'$COLOR1']'$COLOR3'\$ '
18 PS2=$COLOR1'>'$COLOR3' '
19
20 # I want the CET time zone on
   my server in Canada:
21
22 export TZ=CET
23
24 # I want ispell to use the
   correct character set
25 # and the correct dictionary:
26
27 export DICTIONARY=ngerman
28 export CHARSET=latin1
```