Command Line: pwd, cd, pushd, popd and dirs

The Right Path

Paths through the directory jungle are dark and mysterious. In this month's "Command Line" we will be discussing how to keep on the right path and investigating some useful navigation tools. BY HEIKE JURZIK

Trees play an important role in Linux: users are continually confronted with hierarchical structures typically represented by a tree with various branches running from the root to the leaves; these could be the relationships between processes, or between windows on a GUI desktop.

A structure originating at the root directory / contains further directories, which in turn can contain subdirectories or files. File managers like Konqueror (see Figure 1) use a tree view to depict relationships of this type.

In principle, tree structures could be constructed arbitrarily, but rules that stipulate what folder can contain what files are useful as they allow programs to quickly discover the files they need. The "*F*ilesystem *H*ierarchy *S*tandard", FHS, (*http://www.pathname.com/fhs/*) summarizes these rules (see Table 1).

Thus, if you are looking for a library required by a specific program, you should look in the */lib* directory first. If an installed program does not react when its name is called, this may be due to the fact that it is not stored below */bin* but instead in the */sbin* directory – a folder that is often omitted from search paths.

The */usr/local* folder is a special case. It is reserved for software that users install without any help from the packet manager provided by their distributions. Upgrading your distribution should not affect this folder. The folder itself has a similar structure to that of the whole filesystem tree with subdirectories like *lib* and *bin*.

All Roads Lead to Rome

You can use an absolute or relative pathname to discover files and directories within the tree structure. An absolute path is the route to the required file starting at / and including any directories the path crosses e.g. */home/huhn/LM/title.png*. Absolute pathnames always start at the root directory.

In contrast to this, a relative path always starts at the current directory (which is abbreviated to a single dot .). The *pwd* ("*p*rint *w*orking *d*irectory") command lets you know your current position in the tree:

```
huhn@asteroid:~$ pwd
/home/huhn
```

The file */home/huhn/LM/title.png* can be accessed relatively to the current working directory:

```
huhn@asteroid:~$ ls LM/title.png
LM/title.png
```

However, if you are currently working in the *LM* directory and want to access a file in your home directory, you need to climb back up the tree to supply an appropriate relative pathname. The superordinate directory is indicated by two dots ..:

```
huhn@asteroid:~/LM$ ls ../file
../file
```

Navigational Skills

The command *cd directory* ("change *d*irectory") allows you to change to a directory; simply typing *cd* returns you

Command Line

Although GUIs such as KDE or GNOME are useful for various tasks, if you intend to get the most out of your Linux machine, you will need to revert to the good old command line from time to time. Apart from that, you will probably be confronted with various scenarios where some working knowledge will be extremely useful in finding your way through the command line jungle.

GLOSSARY

$: *Prepending a dollar sign to a shell variable name displays the contents of the variable.*

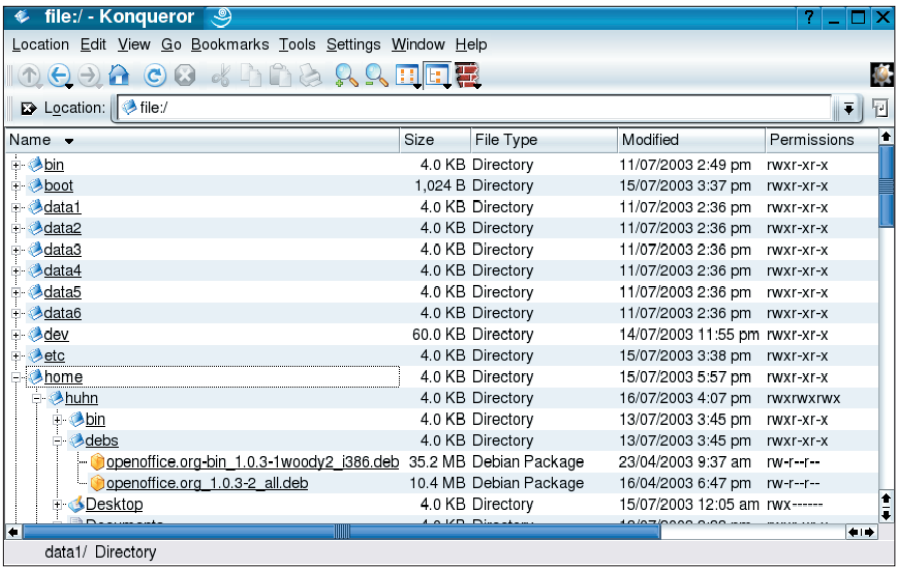

Figure 1: A neat hierarchy – the Linux filesystem tree view in Konqueror

Table 1: Where to find things in the filesystem tree

Directory	Contents
/	The root directory
/home	Private user directories
/root	Home directory for the superuser
/usr	Various programs, the X Window system, documentation and other important data. Contains a sub-hierarchy of its own, with */usr/bin*, */usr/sbin*, */usr/lib*, etc.
/bin and */sbin*	Executable programs ("binaries"); the programs below */sbin* are typically only of interest to admins
/etc	Configuration files
/boot	Files and programs for starting up the system including the kernel
/dev	Device files for hardware components
/lib	Libraries
/opt	Optional programs, often larger software packages that do no distribute their data across the system, but prefer to store it below a common directory

to your own home directory. And again this involves relative and absolute pathnames. Changing directory to */etc* is a lot quicker if you use the absolute pathname (*cd /etc*) rather than typing *cd ../../etc/*, that is climbing up the tree from your home directory.

The standard Linux shell, *bash*, provides a number of practical tools that facilitate command line navigation. For example, it stores information on the last directory accessed by a user in the *OLDPWD* variable. You can type *cd $OLDPWD* to jump back to the previous directory; entering *cd $OLDPWD* a second time returns you to your original position. And using a minus sign instead of *$OLDPWD* with the *cd* command can save you a lot of typing (see Listing 1).

Listing 1: Quick change between two directories

```
huhn@asteroid:~$ cd LM/
huhn@asteroid:~/LM$ cd /etc/
huhn@asteroid:/etc$ cd -
/home/huhn/LM
huhn@asteroid:~/LM$ cd -
/etc
```

As the shell only "remembers" the last place you went to, this function is only useful for jumping back and forth between two directories. But you can create a directory stack to handle multiple directories. Bash provides the *pushd* and *popd* builtins for this purpose.

If you enter *pushd* rather than *cd* to change directory, the shell stores the paths you access on the stack in the order of access. There is no need to memorize the content of the stack, however, as bash lists the directories after each command:

```
huhn@asteroid:~$ pushd LM/
~/linuxmagazine ~
huhn@asteroid:~/LM$ pushd ➥
commandline/
~/LM/commandline ~/LM ~
```

After issuing the second *pushd* command, a total of three directories are now stored on the stack: the tilde (~) on the right representing the home directory, the absolute pathname (*~/LM*) in the center, and the last directory to be accessed on the left (*~/LM/commandline*).

You can use the *popd* command to follow the path back (from left to right). The command removes the last stack entry and calls *cd* for the next list entry. In our example calling *popd* once would change directory to *~/LM*, and a second call would return to the home directory:

```
huhn@asteroid:~/LM/commandline$➥
 popd
~/LM ~
huhn@asteroid:~/LM$ popd
~
```

If you lose track of your directory stack, another bash builtin called *dirs* can be big help:

```
huhn@asteroid:~/LM/commandline$➥
 dirs -p
~/LM/commandline
~/LM
```

The *bash* version 2 command *dirs -p* provides a clearer view of the directories on the stack; the *-c* parameter, which stands for "clear", tells *bash* 2 to remove the directories from the stack. You can also use *pushd +number* to change the order of the directories on the stack: for example *pushd +1* will move the first directory in the list from the left to the start of the list, which is enumerated from 0 upward.

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.