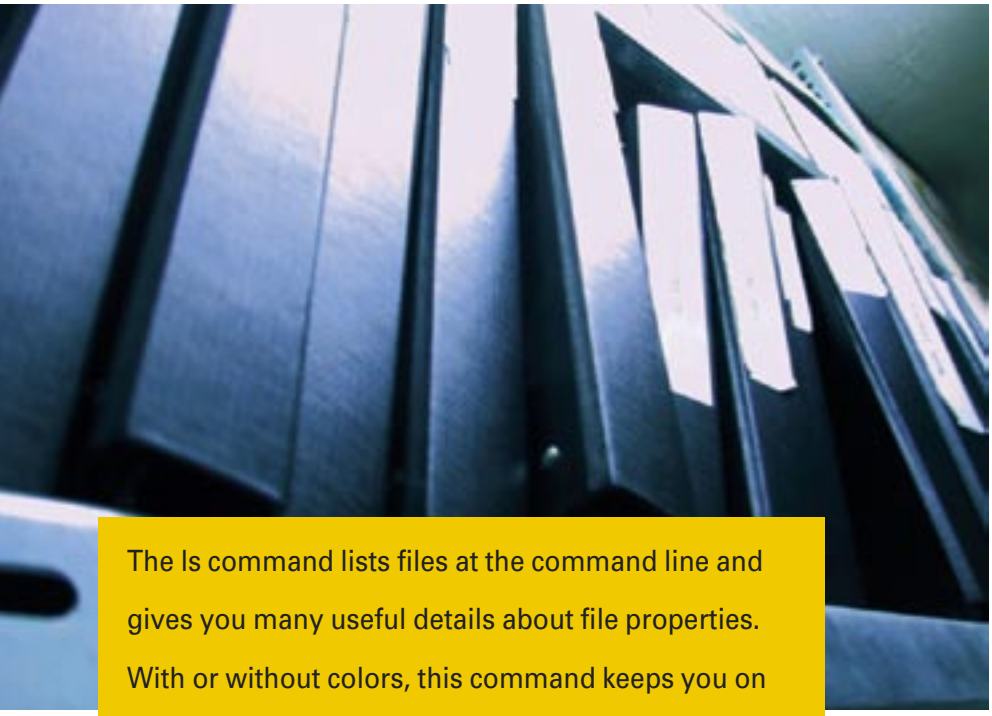
Detailed directory lists with ls

THE LISTING TOOL



The ls command lists files at the command line and gives you many useful details about file properties. With or without colors, this command keeps you on top of your files. **BY HEIKE JURZIK**

The *ls* command is hard to beat when it comes to options and parameters – the manpage reads like a bestseller. As a user, you can select from innumerable options to decide what to show and how to format the display.

What's in There?

Just typing *ls* at the prompt displays the current directory contents, but you can supply a **relative** or **absolute path** to view the content of another directory:

```
ls /etc
```

or

```
ls ../../etc
```

If the output is too colorful, or if you are shown too much information (asterisks for executable files and slashes for directories), you may be working with an *ls* alias. Many modern distributions define a standard abbreviation for their users and pass default parameters to the program. You can type the following, for example, to find out if *ls* is an alias on your system:

```
$ alias ls
alias ls='/bin/ls $LS_OPTIONS'
```

The alias tells the shell to use the full pathname for the program (*/bin/ls*) and the *$LS_OPTIONS* argument whenever a user types *ls*. You can also use the command line to discover the current value of the variable:

```
$ echo $LS_OPTIONS
-N --color=tty -T 0
```

To disable an alias for the current shell session, and experience the original *ls* feeling, type

```
unalias ls
```

If you intend to keep this behavior, you can either add the *unalias* command to your shell configuration file (*.bashrc*) or define your own alias for *ls* (see the section titled "Adaptable") and overwrite the system global setting.

Views

By default, a simple call to *ls* does not display hidden files and directories, that is, the files and directories with names that start with a dot. If you would like to view hidden objects, you can set the *-a* (for "all") option:

```
$ ls -a
./
../
.bash_history
.bash_logout
.bashrc
```

This list also includes the current and parent directories (indicated by a dot or two dots, respectively). To exclude these directories from the list, but still keep any hidden objects, use *-A* instead of *-a*.

Really Informative

The *ls* command gives you a flood of information if you specify the *-l* parameter (see the box titled "Detailed Directory Listing"). The file name is shown on the right – for symbolic links (such as *blubb.ps* in our example) an arrow indicates the original file the link points to. To the left, you can see the last change time and date. (This can be a year if the change took place a while back.) Farther left, you can see the file size in bytes, the owner, and the group.

If you would like to exchange bytes for a different unit of size, simply add the *-h* parameter – *ls* will then round up or down to the nearest full unit.

The second column from the left tells you the number of directory entries for directories (including . and ..) and the number of **hard links** for files. Finally, the ten characters on the far left stand for the file type and permissions. The file type designators are as follows:

- - for a normal file

- *d* a directory
- *l* a symbolic link
- *b/c* **device file** ("block" or "character device")

Sorted by Size

The *ls* command has options that allow you to sort files by size. The *-S* flag gives you a sorted list with the biggest file first. You can use this parameter in combination with *-s* to display the size (again in bytes) in front of the filename:

```
$ ls -sS
11836 user.mpg  4 bla  ➥
4 script.sh  0 blubb.ps
```

Again, you can use the *-h* flag to tell *ls* to use more intuitive units of size:

```
$ ls -sSh
12M user.mpg  4,0K bla  ➥
4,0K script.sh  0 blubb.ps
```

Of course, you can change the sorting order of this output using the *-r* parameter to give you the smallest file first:

```
$ ls -sShr
0 blubb.ps  4,0K script.sh  ➥
4,0K bla  12M user.mpg
```

Age Before Beauty?

If needed, *ls* can sort the directory list by the last change date and time. By default, the "youngest" file heads the list:

```
$ ls -t
script.sh  blubb.ps  ➥
user.mpg  bla
```

Again, *-r* can optionally change the order and output the oldest file first. As a list of this kind is not easy to read, since multiple files are listed in a single line, you can specify the *-1* option to tell *ls* to output only one entry per column:

```
$ ls -tr1
bla
user.mpg
blubb.ps
script.sh
```

Verbosity

If you find this detailed output too verbose, but you would still like to know what kind of files you are dealing with, you can specify the *-F* option with the *ls* command:

```
$ ls -F
bla/  blubb.ps@  blubb.ps~  ➥
user.mpg  script.sh*
```

In this list format, *ls* appends a slash to directories, an at sign (@) to symbolic links, and an asterisk to executables.

You can also tell *ls* to restrict the display, excluding backup copies, which are identifiable by the tilde at the end of the filename, by specifying the *-B* option. If your backups have a suffix such as *.bak*, for example, you can pass the *-I* parameter to *ls* and additionally specify a search pattern for the files to ignore:

```
ls -I *.bak
```

You can also tell the command not to output the contents of subdirectories if you are working with shell wildcards. If you enter the following command

```
ls /etc/cron*
```

to view any files that start with *cron* in the */etc* folder, you will notice that the content of the */etc/cron.daily* subdirectory is output. Bash interprets the asterisk in this case and passes the *ls* command the */etc/cron.daily* directory as an argument, leaving *ls* no option but to output it. You can set the *-d* option to prevent this from happening; in this case you just see an entry for the subdirectory, but not the subdirectory's contents:

```
$ ls -d /etc/cron*
/etc/cron.d  /etc/cron.monthly
/etc/cron.daily  /etc/crontab
/etc/cron.hourly  ➥
/etc/cron.weekly
```

Color Magic

Using the *-F* option to identify file types gives you a good overview, but there is an even simpler way of identifying directory contents at a glance. The *--color* option adds color coding for various file types (Figure 1).

The parameter accepts the definitions *--color = always*, *--color = none*, or *--color = auto* as additional parameters. The last of these variants is the default, and it tells *ls* to use color for direct output to the terminal only. If you redirect the output to another program or file, *ls* drops the make-up and returns to a monochrome display. In contrast to this, *always* will always use color, and *none* will never use color.

To check out the palette that *ls* uses, that is, to discover which color *ls* uses for which file type, you can take a look at the *LS-COLORS* variable:

```
$ echo $LS_COLORS
no=00:fi=00:di=01;➥
34:ln=00;36:pi=40;33:so=01;➥
35:do=01;35:bd=40;33;01:...
```

Since the color codes are difficult to read, you might like to check the color mapping using a command like the following:

```
dircolors -p | less
```

GLOSSARY

Device file: In Linux, devices are represented as files below the */dev* directory. Access to character devices (such as */dev/tty0* the first virtual console) is character-oriented, in contrast to the block-wise read and write operations for block devices (such as */dev/hda* for the first IDE hard disk.)

Detailed Directory Listing

```
01 $ ls -l
02 total 11844
03 drwxrwxr-x  2 hen users
   4096 2005-07-20 10:25 bla
04 lrwxrwxrwx  1 hen users
   10 2005-07-20 10:49 blubb.ps
   -> ../post.ps
05 -rw-r--r--  1 hen users
   12101636 2005-07-20 10:48 hen.
   mpg
06 -rwxr-xr-x  1 hen users
   1325 2005-07-20 10:49 script.
   sh
01 $ ls -lh
02 total 12M
03 drwxrwxr-x  2 hen users 4,0K
   2005-07-20 10:25 bla
04 lrwxrwxrwx  1 hen users   10
   2005-07-20 10:49 blubb.ps ->
   ../post.ps
05 -rw-r--r--  1 hen users  12M
   2005-07-20 10:48 hen.mpg
06 -rwxr-xr-x  1 hen users 1,3K
   2005-07-20 10:49 script.sh
```

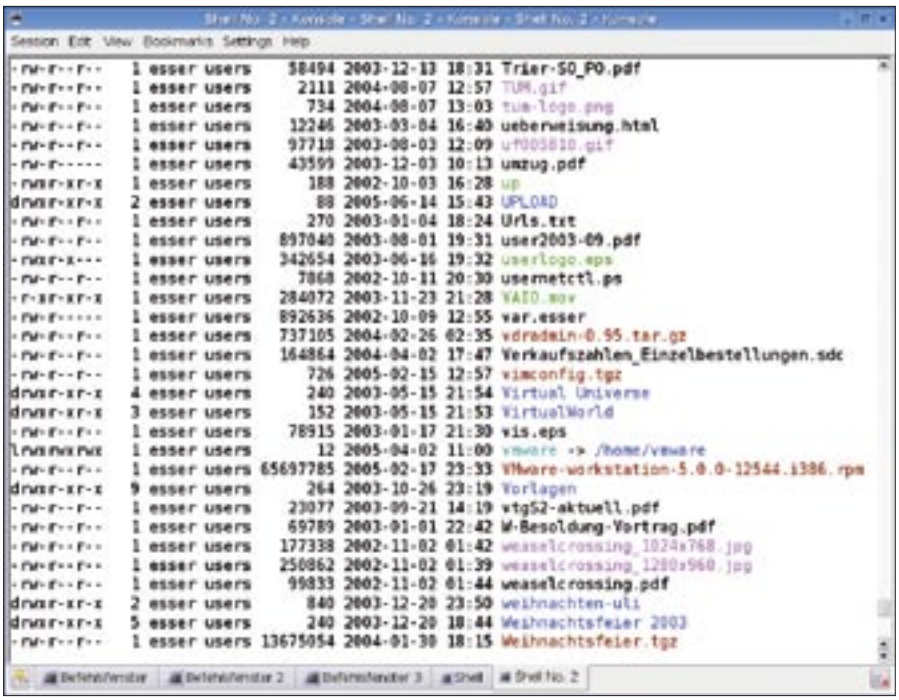


Figure 1: You can color the output from ls.

To prevent the display from dropping off your screen, it makes sense to pipe this output to the *less* utility and scroll through pagewise. The output will tell you that directories are displayed in *bold* type and *blue*:

```
# 00=none 01=bold ➥
04=underscore 05=blink ➥
07=reverse 08=concealed
# 30=black 31=red ➥
32=green 33=yellow ➥
34=blue 35=magenta ➥
36=cyan 37=white
DIR 01;34   # directory
```

Versatile

If you prefer to define a color scheme for the *ls* command, you need to define another varilable in your bash configuration file, *LS_COLORS*. Select the output from the *echo $LS_COLORS* command by pressing the left mouse button and dragging the mouse; then drop the contents of the clipboard into *.bashrc* using the middle mouse button. Before doing so, call *export LS_COLORS =* ; use quotes for any control characters. The file should have a line that looks like this:

```
export LS_COLORS=➥
"no=00:fi=00:di=..."
```

Based on the color codes that you recently identified by entering *dircolors -p | less*, you can now start to color your world. If you don't like the idea of using red for Debian packages (*.deb* extension), you can replace the number *31* with a number of your own, such as *35* for magenta:

```
*.deb=00;35:
```

Then reparse your bash configuration file by giving the following command:

```
source ~/.bashrc
```

and your Debian packages will be displayed in magenta the next time you call *ls --color*.

As I mentioned earlier in this article, most distributors define an alias for the *ls* command, and the alias calls a set of default options. You can customize the parameter set you use in your *.bashrc* file. For example, if you would like *ls* always to use symbols (*-F*) for file types, along with colors (*--color)*, enter the following:

```
alias ls='ls -F --color'
```

It can also make sense to add a separate alias for a long and complex command used to output directories, including hidden directories and files. For example, you could enter:

```
alias ll='ls -laF --color'
```

Don't forget to save your configuration file whenever you make changes (*source ~/.bashrc*).

Too Much

If all of these *ls* options are too much for you to memorize, just add aliases for the commands you use must frequently. That saves typing and human CPU power. ■

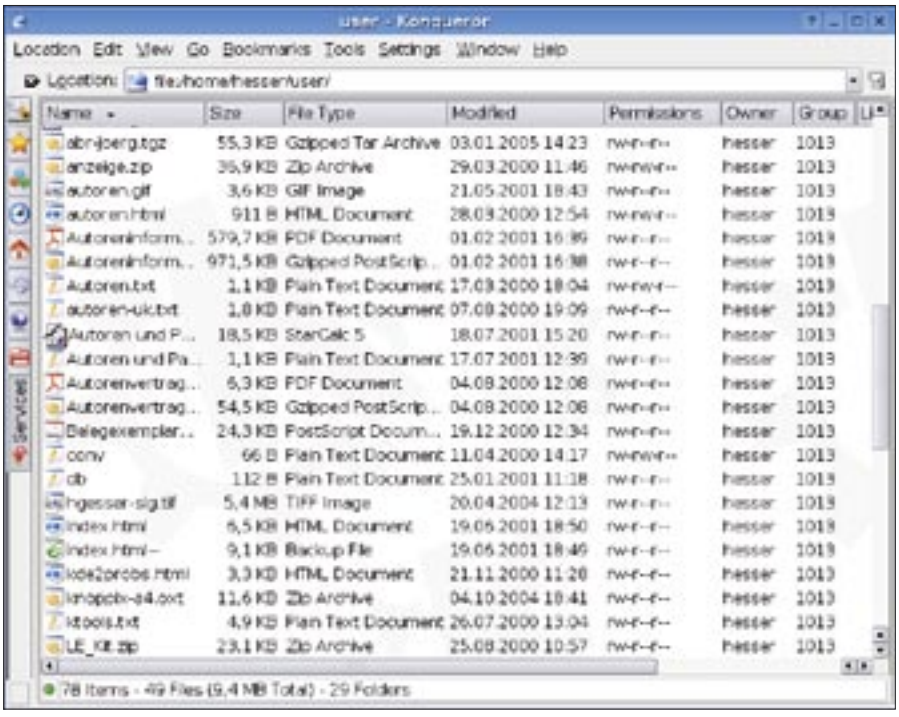


Figure 2: Konqueror will also give you detailed information on files, but it is much slower than ls.