Organizing Lists with the *sort* Command

GETTING SORTED

sort helps you organize file lists and program output. And if you like, you can even use this small but powerful tool to merge and sort multiple files.

BY HEIKE JURZIK

The *sort* command allows you to sort a file line by line. The command can either process a specified file or read from standard input. You decide the sort criteria – *sort* has parameters that give you granular control.

Simple Sorting Exercises

/etc/passwd is an excellent place to start experimenting with sort. If you look inside the file, you will see that the lines comprise a number of fields, for example:

```
chicken:x:506:100:➭
Heike Jurzik:/home/chicken:➭
/bin/bash
```

The individual fields are colon-separated and include the user name (*chicken*), the password (*x* means that the password is in */etc/shadow*), the User Identification Number (UID; for "normal" users, this is a number above *100*), the Group Identification Number (GID), additional information (name and telephone number of the user, etc.), and the user's home directory and standard shell.

To check if the file is already sorted, you can run *sort* with the *-c* flag:

```
$ sort -c /etc/passwd
sort: /etc/passwd:2: unsorted: ➭
bin:x:1:1:bin:/bin:/bin/bash
```

The output tells us that the file is unsorted – the first unsorted entry is in line two. To sort your password file, run *sort* (without specifying any other parameters at first). Don't forget to redirect the output, however, otherwise it will be sent to your display. You can use the > operator to redirect the output to a file name:

```
sort /etc/passwd > passwd_sort
```

The greater than character will overwrite the content in *passwd_sort* if the file exists. You can prevent this from happening using the > > operator, which appends the output at the end of the file.

As the output from our first sort experiment shows (see Figure 1), *sort* has sorted the file alphabetically (using the first field – the user name – as an index). This puts the "pseudo-user" *at* at the top of the list. The *-r* option reverses the order – this puts *wwwrun* at the top of the list. If you are sorting a file that contains duplicate entries, you can suppress the output of identical lines using the *-u* (for "unique") parameter.

Columns

Unless you tell it not to, *sort* will look at the whole line and start with the first let-

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.

ter – just as in our first example. But you can use a few tricks to get the command to start sorting at a specific position. By default, *sort* will look for spaces between entries and interpret these spaces as column delimiters. Consider a fictitious password file with the following structure:

```
...
peggy x 502 100 Peggy Goose ➥
/home/peggy /bin/bash
chicken x 506 100 Heike Jurzik ➥
/home/chicken /bin/bash
petronella x 507 100 ➥
Petronella chicken ➥
/home/petronella /bin/bash
...
```

Let's assume we want to sort this list based on the UID (that is, column three); we need the *-k* option and the column number as the value:

```
$ sort -k 3 /etc/passwd
root x 0 0 root /root /bin/bash
vdr x 100 33 Video Disk ➥
Recorder ➥
/var/spool/video /bin/false
uucp x 10 14 Unix-to-Unix ➥
CoPy system /etc/uucp /bin/bash
...
```

Unfortunately, *sort* interprets numbers as simple ASCII strings and happily sorts them alphabetically. This behavior means that "534" comes before "55" in the list, because, from an alphabetical point of view, "53" is smaller than "55." In our example, this means that the UID *100* is higher up the list than the UID *2*. To stop this from happening, we need to specify the *-n* option, which helps *sort* get its figures right:

```
$ sort -nk 3 /etc/passwd
root x 0 0 root /root /bin/bash
bin x 1 1 bin /bin /bin/bash
daemon x 2 2 Daemon /sbin ➥
/bin/bash
...
```

As the real password file uses colons, not blanks, as the separating characters between entries, we need to let *sort* know which separating character we are using; in other words, we need the *-t* "separator." Make sure you escape the separator character with double quotes to prevent Bash from mangling it. The following command will sort your real */etc/passwd* file:

```
sort -t ":" -nk 3 /etc/passwd
```

Being Restrictive

You can restrict the columns that sort inspects even further to target specific areas. Let's look at another example. We will be sorting a file called *sortfile* with the following contents:

```
1 Hens are loud.
2 Geese are louder.
3 Ducks are loud.
4 Eggs are good.
5 Wrens are lots of noise.
6 Hens squawk loudly.
7 Geese gobble louder.
8 Ducks quack louder.
```

Our first command will sort the file starting in the third column and working towards the end of the file; in other words, if the third column is identical ("are" is the third word in half of these examples), *sort* inspects the fourth column, then the fifth, etc. The option we need for this is *-k 3*:

```
$ sort -k 3 -k 2,2 sortfile
4 Eggs are good.
5 Wrens are lots of noise.
3 Ducks are loud.
1 Hens are loud.
2 Geese are louder.
7 Geese gobble louder.
8 Ducks quack louder.
6 Hens squawk loudly.
```

But what happens if the third column and the remainder of the line are identical (which is the case in lines 1 and 3?) In this case, *sort* inspects the second field (as specified by the second *-k* parameter) and sorts alphabetically, putting "Ducks" higher up the list than "hens." In other words, the argument *-k 3* means "column 3 and all columns that follow," whereas *-k 2,2* means column 2 through 2 (in other words, just column 2.)

Let's restrict the sort operation to column 3 now:

```
$ sort -k 3,3 -k 2,2 sortfile
3 Ducks are loud.
4 Eggs are good.
2 Geese are louder.
1 Hens are loud.
5 Wrens are lots of noise.
7 Geese gobble louder.
8 Ducks quack louder.
6 Hens squawk loudly.
```

The number that follows the comma indicates the end of the sort area; this is column 3 in our case. This tells the sort tool to ignore everything from column 4 through to the end of the line. After sorting the lines alphabetically (based on the word in column 3), *sort* inspects column 2 and carries on sorting. You may have noticed that the word "are" occurs quite often in column 3, so the lines are arranged in alphabetical order based on the word in column 2; that is, we have "ducks" before "geese", "hens", and "wrens."

If the columns contain padding characters that you want *sort* to ignore, you can specify where the sorting tool should assume the column starts. To start sorting at character four in column three, add a dot and the character *4* to the column number:

```
sort -k 3.4
```

The following command sorts from the fourth character in column three to the fifth character in column four:

Listing 1: Sorting a directory by file size.

```
$ ls -l /boot/ | sort -k 5g
total 3792
lrwxrwxrwx    1 root   root
   1 2003-11-18 16:53 boot -> ./
lrwxrwxrwx    1 root   root
   23 2003-11-18 17:07 initrd ->
   initrd-2.4.21-99-athlon
lrwxrwxrwx    1 root   root
   24 2003-11-18 16:57 vmlinuz ->
   vmlinuz-2.4.21-99-athlon
drwxr-xr-x    2 root   root
   1024 2003-11-18 16:53 grub/
drwx------    2 root   root
   12288 2003-11-18 16:47
   lost+found/
-rw-r--r--    1 root   root
   52224 2003-09-24 15:32
   config-2.4.21-99-athlon
...
```

```
sort -k 3.4,4.5
```

More Sorting Wizardry

sort has a many additional parameters that you may or may not need for your daily work. The manpage has a useful overview. The ability to enable or disable parameters individually for individual fields is a useful feature. For example, you do not need to apply the *-r* (for reverse direction sorting) and *-n* (for correct numerical output) options globally. If required, you can apply these options to individual fields – although this does mean fairly cryptic looking commands:

```
sort -t ":" -k 4n,4 -k ↩
3nr,3 /etc/passwd
```

This command sorts */etc/passwd*, first specifying the fields are colon-separated (*-t* ":") before going on to sort numerically by group ID (field 4) and then in reverse order by user ID.

File Merger

You can tell *sort* to merge two previously sorted files to create a new (and still sorted) file a lot quicker than with the *cat file1 file2 | sort* command. Just use the merge option, *-m*:

```
sort -m file1 file2 > merged
```

Again you can specify the *-u* parameter to remove duplicate entries.

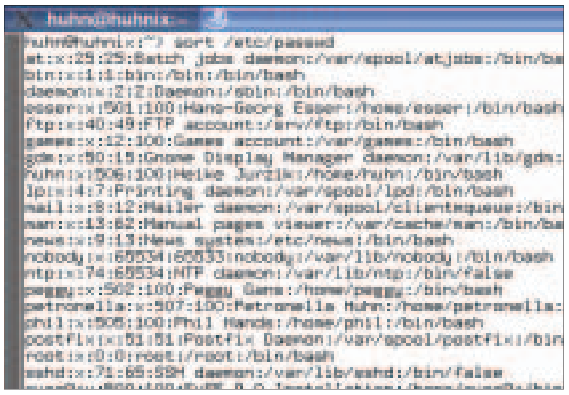
Teamwork with Other Commands

sort really shines in combination with other command-line tools. For example to sort the contents of a long */boot* directory listing by file size, simply pipe the output of the *ls* command to the *sort* command, as shown in Listing 1.

Another common usage, which administrators may be familiar with, is a link between *du* and *sort* to identify "disk hogs." If you suspect a disk hog in your home directories, you can type the following to identify the culprit:

```
$ du -s /home/* | sort -rn
6356156 /home/petronella
47304   /home/peggy
18864   /home/chicken
...
```

As you can see, the *du* command uses the *-s* parameter to output the size of each home directory; the output from the *du* command is then piped to the *sort* command, and it soon becomes apparent that *petronella* is using more than her fair share of space. ■



```
huhn@huhnix:~> sort /etc/passwd
at:x:25:25:Batch jobs daemon:/var/spool/atjobs:/bin/ba
bin:x:1:1:bin:/bin:/bin/bash
daemon:x:2:2:Daemon:/sbin:/bin/bash
esser:x:501:100:Hans-Georg Esser:/home/esser:/bin/bash
ftp:x:40:49:FTP account:/srv/ftp:/bin/bash
games:x:12:100:Games account:/var/games:/bin/bash
gdm:x:50:15:Gnome Display Manager daemon:/var/lib/gdm:
huhn:x:506:100:Heike Jurzik:/home/huhn:/bin/bash
lp:x:4:7:Printing daemon:/var/spool/lpd:/bin/bash
mail:x:8:12:Mailer daemon:/var/spool/clientmqueue:/bin
man:x:13:62:Manual pages viewer:/var/cache/man:/bin/ba
news:x:9:13:News system:/etc/news:/bin/bash
nobody:x:65534:65533:nobody:/var/lib/nobody:/bin/bash
ntp:x:74:65534:NTP daemon:/var/lib/ntp:/bin/false
peggy:x:502:100:Peggy Gans:/home/peggy:/bin/bash
petronella:x:507:100:Petronella Huhn:/home/petronella:
phil:x:505:100:Phil Hands:/home/phil:/bin/bash
postfix:x:51:51:Postfix Daemon:/var/spool/postfix:/bin
root:x:0:0:root:/root:/bin/bash
sshd:x:71:65:SSH daemon:/var/lib/sshd:/bin/false
```

Figure 1: "sort" gives you an alphabetically sorted password file in next to no time.