

# Work

by Jeffreys Copeland and Haemer



ERIC MUELLER

*"I am waiting for you, Vizzini. You told me to go back to the beginning."*  
– "The Princess Bride"

*"Indeed, I believe that virtually every important aspect of programming arises somewhere in the context of sorting or searching!"*  
– Donald E. Knuth, *The Art of Computer Programming*

## Back to Basic(s): sort

A few months ago, we wrote a column entitled "Back to Basic(s)" (February, Page 42, <http://sw.expert.com/C9/SE.C9.FEB.00.pdf>), in which we discussed some of the reasons development on a UNIX platform is different from development on other platforms, and why we think that makes UNIX development easier. There are other reasons we like working on UNIX, too, like the rich variety of tools that are part of the base system. Even though we were tempted to give our editor her third chance in as many months to plaster "Sex" on the cover of the magazine, we'll talk about one of those UNIX tools in detail instead.

(In fact, we keep thinking about under-used and little-known corners of UNIX lore. Don't be surprised if you see the occasional installment of "Back to Basic(s)" interspersed with our normal columnar fare.)

Sorting is one of those processes that gets used a lot in the commercial world, but we tend not to see it day-to-day in

our development work. This means that often we have to go look up the flags to sort when we want to do something other than simply alphabetize the lines in a file. We had such a situation recently. But first...

### A Correction

We normally leave corrections for the end of the column, but this time our goof was so egregious we want to get it off our chests right now. In our January column ("Calendar Pages," Page 43, <http://sw.expert.com/C9/SE.C9.JAN.00.pdf>), we talked about calendars and developed some code to calculate and draw the current phase of the moon. Even though we know better, we described the picture of the moon as "...two arcs, the regular outer edge of the moon and the curve made by the Earth's shadow." Unfortunately, it took University of Maryland astronomer Patrick Shopbell to remind us of the facts.

Remember that the moon always keeps the same face toward us, which is

why nobody had seen the back side of it until the Soviet and American unmanned lunar missions of the mid-1960s. The part of the moon that's unlit is not in the Earth's shadow, but the side facing away from the sun. If you picture the Earth-moon-sun system in your mind, it should be obvious that the edge of the moon toward the sun is a semicircle of the full lunar radius. The other curve is the edge of the moon at right-angles to the sun projected to our angle of view, and is hence a foreshortened circle also with the moon's radius. If the moon were translucent, the view from your backyard of the separator between lunar night and day (the terminator) would be of an ellipse. (This suggests some improvements in our moon-drawing code to make the pictures more verisimilitudinous, but we leave that as an exercise for the reader.)

Again, we should have known better, and now, if you were taken in by our sloppiness, you know better, too.

## Songs in the Last Millennium

As part of the pseudo-millennial madness that gripped the world at the end of December, one of our local radio stations took a listener poll and played the 1,000 most popular hit songs. Because it's a rock 'n' roll station, we weren't surprised that no early 20th century Appalachian folk music appeared among the 1,000 songs. Be that as it may, we started asking the obvious questions. For example, if "Lawyers in Love" is number 1,000 and "Stairway to Heaven" is number one, what songs in between are by Jackson Browne and Led Zeppelin, respectively? How many songs by The Beatles made this list? The Byrds? The Troggs? [Jeff, perhaps we're dating ourselves with those examples?—Jeff]

Though our normal anal-retentive approach would be to sit up for three days straight and madly scribble down song titles during each back-announcement, the radio station was kind enough to supply us with a full list on its Web site. The bulk of the HTML file consists of lines such as:

```
834 SHATTERED, ROLLING STONES<BR>
835 OHIO, "CROSBY, STILLS NASH & YOUNG"<BR>
```

These lines give us three pieces of data: the rank, the title and the group/artist. The rank is separated from the title by a space, and the title from the group by a comma. If the title or the group has a comma, that field is enclosed in double quotes.

First, we want to strip the list out of the HTML file. We could write a Perl program to parse the HTML and do that for us, but it's more trouble than it's worth when we have a text editor like `vi`. While we're at it, we can remove the `<BR>` tag from the end of each line. We'll write the whole list to a file called `hits`.

Next, we need to separate the fields with a character that doesn't appear elsewhere in the list. Our first choice would be a slash, `/`, but we quickly discover there are titles like "Sgt. Pepper/A Little Help from My Friends," so we fall back to a semicolon. The Perl script to do this is pretty simple:

```
#!/usr/local/bin/perl -p
s/([0-9]+) "([^"]+)"*, "([^"]+)"*/$1; $2; $3/;
```

The `-p` flag to Perl, you may recall, wraps the `s/.../.../` line with

```
while( <> ) {
    ...
    print;
}
```

If you prefer, the `sed` version is almost identical:

```
#!/bin/sed -f
s#\([0-9]*\) "#\([^"]*\)"#, "#\([^"]*\)"#\1;\2;\3#
```

(An aside: The radio station originally displayed this list as an undifferentiated lump of text on its Web page ["Lump

of Text" was the name of a failed online electronic book venture last year] thus:

```
1 STAIRWAY TO HEAVEN, LED ZEPPELIN 2 HOTEL
CALIFORNIA, EAGLES 3 AMERICAN PIE, DON McLEAN
4 HEY JUDE, THE BEATLES 5 CHINA GROVE, DOOBIE
BROTHERS
```

This version of the list would only be a little harder to parse, but fortunately we don't have to. However, we leave the modified scripts as an exercise for the reader.)

## Sorting the Results

Now that we've got a list separated into fields, it is merely a matter of referring to the `sort` man page and setting up some more shell script.

To begin with, we would like the list in order numerically. This is exactly what `sort` is set up for, so we say:

```
sort HITS | head
```

We're surprised with the results:

```
1000; LAWYERS IN LOVE; JACKSON BROWNE
100; I WANT YOU TO WANT ME; CHEAP TRICK
101; SGT. PEPPER/A LITTLE HELP FROM MY
    FRIENDS; THE BEATLES
102; BABY I LOVE YOUR WAY; PETER FRAMPTON
103; WALK THIS WAY; AEROSMITH
```

What happened? `sort` is operating on the lexical values of the strings it sees. In other words, "2" is larger than "10" when sorted alphabetically, and "100;" is less than "1;." In other words, `sort` is saying to us, "well, if you wanted me to sort the list by the number, why didn't say so?" We will:

```
$ sort -n HITS | head -5
1; STAIRWAY TO HEAVEN; LED ZEPPELIN
2; HOTEL CALIFORNIA; EAGLES
3; AMERICAN PIE; DON McLEAN
4; HEY JUDE; THE BEATLES
5; CHINA GROVE; DOOBIE BROTHERS
```

What next? It would be interesting to have the list by title, so we could look up our personal favorites to see where they appeared. For that, we need to look at the second field, which contains the title. We need to tell `sort` what delimiter separates fields, so we use the `-t';'` flag—remember that we need the quotes around the semicolon because it's got special significance to the shell. There are two ways to get `sort` to modify which order it looks at fields to compose its sorting keys. The traditional form uses field numbering beginning at zero, and a plus sign: `sort -t';' +1`. The newer form, built into the POSIX.2 standard, uses the `-k` flag, and begins numbering fields at a more natural 1. Using this flag, we'd say:

```
$ sort -t';' -k2 HITS | head -5
```

```
755; #9 DREAM; JOHN LENNON
170; (Just Like) STARTING OVER; JOHN LENNON
403; (We Ain't Got) NOTHIN' YET; BLUES MAGOOS
182; 19th NERVOUS BREAKDOWN; ROLLING STONES
386; 2 OUT OF 3 AIN'T BAD; MEATLOAF
```

Again, not quite what we wanted. If we add the `-d` and `-f` flags, we get sorting in dictionary order (ignoring nonalphanumeric characters) and sorting that ignores case differences:

```
$ sort -t';' -fd -k2 HITS | head
182; 19th NERVOUS BREAKDOWN; ROLLING STONES
386; 2 OUT OF 3 AIN'T BAD; MEATLOAF
84; 25 OR 6 TO 4; CHICAGO
693; 50 WAYS TO LEAVE YOUR LOVER; PAUL SIMON
544; 867-5309/JENNY; TOMMY TUTONE
755; #9 DREAM; JOHN LENNON
236; A DAY IN THE LIFE; THE BEATLES
91; A HARD DAY'S NIGHT; THE BEATLES
619; A HAZY SHADE OF WINTER; SIMON & GAFUNKEL
472; A HORSE WITH NO NAME; AMERICA
```

So far, so good, but we'd really prefer "A Day in the Life" to sort as though it were "Day in the Life," and "The Logical Song" to sort alongside the other "L" entries. This suggests either adding an `l` flag (for librarian sort) to `sort`, or providing a script wrapper for `sort` to do the same thing. We'll leave that one as an exercise for the reader and ignore the problem for now—but send us your solutions.

That gives us two interesting forms for the data, but we'd really like a couple more. What songs on the list were done by what performers? In other words, let's sort the list by group. You can guess what the approach for that is:

```
$ sort -t';' -df -k3 HITS
...
724; ANGEL; AEROSMITH
787; LOVE IN AN ELEVATOR; AEROSMITH
808; LET'S STAY TOGETHER; AL GREEN
649; YEAR OF THE CAT; AL STEWART
321; SCHOOL'S OUT; ALICE COOPER
626; ONLY WOMEN; ALICE COOPER
702; NO MORE MR. NICE GUY; ALICE COOPER
294; RAMBLIN' MAN; ALLMAN BROTHERS BAND
352; SISTER GOLDEN HAIR; AMERICA
...
```

This leaves one question unanswered from our original set: How many songs by The Troggs appear on the list? The solution for that requires a little manipulation. To begin, we'll need to extract the third field from each line, using the helpful `cut` utility:

```
$ cut -d';' -f3 HITS | head -5
LED ZEPPELIN
EAGLES
DON McLEAN
```

```
THE BEATLES
DOOBIE BROTHERS
```

(Note that `cut` and `sort` use different flags for their field-delimiter characters. This is an annoyance, but it pervades the complete set of UNIX utilities: different option flags are used for the same data all through the 300-odd basic utilities, mostly for historical reasons.)

Next, we need to sort these so the groups are together, and count each set. Fortunately, we can use `uniq`. Normally, `uniq` compresses like lines down to one instance, but it features a `-c` flag to count the number of identical lines it finds:

```
$ cut -d';' -f3 HITS | sort | uniq -c | head -5
2 10 CC
2 38 SPECIAL
2 AC/DC
1 ACE
7 AEROSMITH
```

So far, so good, but that gives us a count of songs by group. What about a list ordered by count? We could sort the list numerically:

```
$ cut -d';' -f3 HITS | sort | uniq -c |
sort -n | head -5
1 ACE
1 AL GREEN
1 AL STEWART
1 ALLMAN BROTHERS BAND
1 APOLLO 100
```

Not nearly as interesting as we'd like. What about inputting them with the most popular performers first? For that, we need `sort`'s `-r` flag to reverse the order of the sort:

```
$ cut -d';' -f3 HITS | sort | uniq -c |
sort -rn | head -5
78 THE BEATLES
35 ROLLING STONES
27 ELTON JOHN
19 EAGLES
18 CHICAGO
```

That's good, but a little further down the list, something funny happens:

```
11 STYX
11 ELECTRIC LIGHT ORCHESTRA
11 DOOBIE BROTHERS
10 JOHN COUGAR MELLENCAMP
9 JOURNEY
9 ERIC CLAPTON
```

How come those groups are no longer sorted alphabetically? `sort` is just doing what we told it to: it's sorting in reverse order. We can fix this simply by using multiple sort fields.

# Work

First, we have `sort` use only the first column for the reverse numeric sort and stop that sort key at the beginning of the second field by saying, `-k1,2nr`. This should be enough, but not all implementations of `sort` are guaranteed to be stable; that is, we can't be sure they'll leave input lines that would otherwise sort in the same order as they were in the input file. We fix that by adding a second sort key, `-k2bdf`, which begins sorting at the second field, ignores leading blanks (`b`) and uses the `df` combination from earlier. (For reference, some versions of `sort` have an `-s` flag to force stable sorts.)

```
$ cut -d';' -f3 HITS | sort | uniq -c |
sort -k1,2nr -k2bdf
78 THE BEATLES
35 ROLLING STONES
27 ELTON JOHN
19 EAGLES
18 CHICAGO
...
11 DOOBIE BROTHERS
11 ELECTRIC LIGHT ORCHESTRA
11 STYX
10 JOHN COUGAR MELLENCAMP
9 ERIC CLAPTON
9 JOURNEY
```

Which is, of course, what we expected in the first place.

## Wrapping Up

In answer to our original question:

```
$ cut -d';' -f3 HITS | sort | uniq -c |
egrep "(TROGGS|BYRDS)"
4 THE BYRDS
1 THE TROGGS
```

We've seen how useful the UNIX `sort` utility can be for sorting song titles. We can probably generalize it for other uses. There's an entirely different sort you would use inside your programs: the `qsort()` interface. We'll discuss it, and tricks like using it to sort a linked list, some other time.

Next month, we'll talk about the magical kites and darts invented by British mathematician Roger Penrose, and how to have our computers draw them. Until then, happy trails. ✍

---

*Jeffrey Copeland* ([copeland@alumni.caltech.edu](mailto:copeland@alumni.caltech.edu)) is currently living in the Pacific Northwest, where he spends his time writing UNIX software in a large development organization and fighting damp rot.

*Jeffrey S. Haemer* ([jsh@usenix.org](mailto:jsh@usenix.org)) works at QMS Inc. in Boulder, CO, building laser printer firmware. Before he worked for QMS, he operated his own consulting firm and did a lot of other things, like everyone else in the software industry.

Note: The software from this and past Work columns is available at <http://alumni.caltech.edu/~copeland/work> or alternately at <ftp://ftp.expert.com/pub/Work>.