Software skills for librarians

Module 1: The Unix shell
Answers

1. The three stages are, probably: collate the sheets, put them in an envelope and finally label the envelope. Each pack is the same, so there's no problem with matching these up to the address label, meaning that labelling the envelope can be done at any stage. What I did was to arrange piles of each sheet, and empty envelopes, then go along the piles taking one of each and put the complete pack to one side. In the next pass I put the packs into the envelopes, and finally affixed the address labels. If another person is available then they could put the packs in the envelopes and stick the labels as the first person collates the sheets.

2. Collate the project, punch it and bind it. Time is an estimate, but possibly total time about 10 minutes. The point here is collating is much quicker than the other two steps. Splitting punching and binding helps slightly if there are many students in the queue, because the second one can start punching while the first is binding. Having two machines approximately doubles throughput.

4. From the home directory: `mkdir libcarp1`

5. To unpack the files use: `unzip libcarp1.zip -d libcarp1`
Windows users may simply do this via the GUI. Most users will probably download the files using their Web browser, but the more confident may try:
`wget http://www.cl.cam.ac.uk/~ncc25/libcarp/libcarp1.zip`

6. There are 6 files, about 610 kbytes. This can be found with `ls -l`

```
-rw-rw-r-- 1 ncc25 ncc25 611864 Mar 15 10:14 gulliver.txt
-rwxrwxr-x 1 ncc25 ncc25    969 Mar 15 10:32 kernel.marc
-rwxrwxr-x 1 ncc25 ncc25    743 Mar 15 10:33 linux.marc
-rwxrwxr-x 1 ncc25 ncc25     34 Mar 15 11:27 marctxt
-rw-r--r-- 1 ncc25 ncc25   2739 Mar 15 10:21 marctxt.pl
-rwxrwxr-x 1 ncc25 ncc25    871 Mar 15 10:32 system.marc
```

Another approach which conveniently displays the totals is to list the contents of the zip file using `unzip -l libcarp1.zip`.

7. `cd ~`, possibly something like `cd /users/ncc25` depending on your system.

8. The following commands, in this order:

```
nano example.txt
cp example.txt ../example.txt
mv example.txt newtext.txt
cat newtext.txt
```

9a. There is too much text to fit on the screen, and it scrolls past too quickly to read.

9b. Use the commands `less` or `more`. You can also pipe the output of another command into these, eg: `cat gulliver.txt | more`

9c. Use the `wc` command, eg:
`wc -w gulliver.txt` or `wc -l gulliver.txt`

10a. For clarity, the loop is best entered over three lines:

```
for filename in *.marc;
  do ./marctxt $filename;
done
```

10b. This is similar, but we need to use output redirection, and we probably want to change the file extension to .txt, so we can't use the wildcard *.marc:

```
for filename in linux system kernel;
  do ./marctxt $filename.marc > $filename.txt;
done
```

11. Most of these require some knowledge of regular expressions, notice also the use of quote marks to prevent the shell from processing the pattern argument.

```
grep -E '^650' {linux,system,kernel}.txt
grep -E '^040.*\$erda' {linux,system,kernel}.txt
grep -E '^33[6-8]' {linux,system,kernel}.txt
grep 'Addison Wesley' {linux,system,kernel}.txt
grep -E '^245.*Linux' {linux,system,kernel}.txt
```

12. The following is the necessary sequence of commands. Some of these could be combined together using a 'pipe':

```
sed '9352,9714d' gulliver.txt >gulliver_nofoot.txt
sed '1,37d' gulliver_nofoot.txt >gulliver_nohead.txt
tr -d [:punct:] <gulliver_nohead.txt | tr [:upper:] [:lower:]
    >gulliver_clean.txt
tr ' ' '\n' <gulliver_clean.txt >gulliver_lines.txt
sort gulliver_lines.txt >gulliver_ordered.txt
uniq -c gulliver_ordered.txt > gullliver_final.txt
rm gulliver_*.txt
```

13a. It matches: France, Franch, Frence and French.
13b. As above but on its own, without any other characters around it.
13c. `/^(France|French)/`
13d. `/Colou?r/`

14. A four letter word at the end of a line: `/\w{4}$/`

15a. Note the escape character is usually needed before a forward slash:
`/\d{2}\/[01]\d\/\d{4}/`
15b. You can make this as complicated as you like. This solution copes with the fact that the day number may have only one digit, or two with the first only between 0 and 3. Similarly the month name will have an initial capital letter:
`/[0-3]?\d [A-Z][a-z]{2} \d{4}/`

16a. Note the final character can be a digit or a letter 'X': `/\d{9}[0-9X]/`
16b. 13 digit ISBNs cannot have an X in the checksum, so: `/978\d{10}/`
16c. There are different ways of hyphenating an ISBN, so no solution is perfect, but one likely one is: `/\d-\d{3}-\d{5}-[0-9X]/`

17a. MARC language code: `/[a-z]{3}/`
17b. MARC language code and subfield: `/\$[a-h][a-z]{3}/`
17c. Complete 041 field: `/041 [_01]_ (\$[a-h][a-z]{3})+/`