Assignment 3. Installing and modifying software

Laboratory: Installing a small change to a big package

Keep a log in the file `lab3.txt` of what you do in the lab so that you can reproduce the results later. This should not merely be a transcript of what you typed: it should be more like a true lab notebook, in which you briefly note down what you did and what happened.

You're helping to build an application containing a shell script that invokes the `ls` command to get file status. Your application is running atop the Maroon Chapeau Enterprise Linux 5 distribution, which uses the `ls` implementation supplied by Coreutils 7.6. You've been running into the problem that for some users `ls` generates output that looks like this:

```
$ ls -l /bin/bash
-rwxr-xr-x 1 root root 729040 2009-03-02 06:22 /bin/bash
```

The users want the traditional Unix format, which looks like this:

```
$ ls -l /bin/bash
-rwxr-xr-x 1 root root 729040 Mar  2  2009 /bin/bash
```

You've been asked to look into the problem and fix it.

You discover that the problem is that in some cases users set their locale to a value like `en_US.UTF-8`, for example, by setting the `LC_ALL` environment variable to that value:

```
$ export LC_ALL='en_US.UTF-8'
```

Users who have done this get the YYYY-MM-DD date instead of the traditional Unix date.

You nose around on the net, and discover that the problem is that the locale files for Coreutils are not generated properly (see Jim Meyering's message of 2009-09-29). Getting these files generated and distributed to all your clients seems like a bit of a hassle, so instead, you decide to patch the `ls` program instead, using a temporary workaround patch published by Pádraig Brady.

Try Brady's workaround, as follows:

1. Grab Coreutils 7.6.
2. Compile and install your copy of Coreutils into a temporary directory of your own. Note any problems you run into.
3. Reproduce the bug on your machine with the unmodified version of coreutils. You may need to use the `locale-gen` program to generate the `en_US.UTF-8` locale.
4. Use Emacs or Vim to apply Brady's patch.
5. Type the command `make` at the top level of your source tree, so that you build (but do not install) the fixed version. For each command that gets executed, explain why it needed to be executed (or say that it wasn't neeeded).
6. Make sure your change fixes the bug, by testing that the modified `ls` works on your test case and that the installed `ls` doesn't. Test on a file that has been recently modified, and on a file that is at least a year old. You can use the `touch` command to artficially mark a file as being a year old.

Q1. Why did Brady's patch remove the line "`case_long_iso_time_style:`"? Was it necessary to remove that line? Explain.

Q2. If your company adopts this patched version of Coreutils instead of the default one, what else should you watch out for? Might this new version of Coreutils introduce other problems with your application, perhaps in countries where users don't speak English and don't understand English-format dates?

Homework: Generating random lines from a file

Consider the Python script `randline.py`.

Q3. What happens when this script is invoked on an empty file like `/dev/null`, and why?

Modify `randline.py` so that it takes an arbitrary number of input file arguments, not just one. If no input files are specified, it should read from standard input. If an input file ends in a non-newline character, your code should silently act as if a newline were appended to the file.

Currently `randline.py` outputs lines with replacement; for example, it's possible that it will output the first input line two or more times. Modify `randline.py` so that it accepts a new option `-w` (long version `--without-replacement`) which causes it to output lines without replacement: for example, the first input line is copied to the output at most once. If the input contains duplicate lines, the output can contain the same duplicates, but no more duplicates than what appeared in the input.

With the new option it is an error if the input file contains fewer than `NUMLINE` lines, just as it is already an error without the new option to invoke `randline.py` on an empty file and ask it for one or more lines of output. You need not worry about the exact diagnostic you generate for this new error, so long as you generate some diagnostic.

Your modified version of `randline.py` should use only the `string` module and the modules that `randline.py` already uses (it should not import any other modules). Don't forget to change its usage message to accurately describe the modified behavior.

Submit

Submit the following files.

- The file `lab3.txt` as described in the lab.
- A file `hw3.txt` containing the answer to questions Q1 through Q3 noted above.
- The modified file `randline.py` as described in the homework.

All files should be ASCII text files, with no carriage returns, and with no more than 80 columns per line. The shell command:

```
expand lab3.txt hw3.txt randline.py | awk '/\r/ || 80 < length'
```

should output nothing.