Week 12 Lab

Perl

Task 1. Regular expressions and console I/O

Devise Perl regular expressions to match the following items:

a) Any Melbourne phone number (8 digits with a space in the middle)

b) The days of the week

c) Any IP address (IP addresses look like 130.194.9.1, where each number is in the range 0 to 255)

d) An absolute web URL

e) Text that looks like a sentence of English text (begins with a capital letter, ends in a full stop or ? or !, spaces after commas/semicolons, etc.)

Write a Perl script to test your regular expressions by attempting to match each one against a line of input from the keyboard.

Task 2. Parsing text

The `split` function splits lines based on matching a regular expression. Write a script called `mycut` that implements a simplified version of the Unix `cut` command. It does not need to support any command-line flags, only has to work on comma-separated fields, and reads from standard input. The first command-line argument specifies the number of fields in each line of the input. Fields should be specified on the command line using the `cut` syntax as specified in the manual page, eg

1 specifies the first field

2-4 specifies fields 2, 3, and 4

-3 specifies fields 1, 2, and 3.

2- specifies every field but the first field.

To simplify parsing field specifications, you may assume that there is a maximum of nine fields in the input. For example, given the following input

```
Fred,Nurk,12345,60
Jane,Bloggs,54321,70
```

`mycut 4 1 3 2-4 -2 3-` should produce the following output:

```
Fred,12345,Nurk,12345,60,Fred,Nurk,12345,60
Jane,54321,Bloggs,54321,70,Jane,Bloggs,54321,70
```

Task 3. File I/O

Starting with your script for `mycut`, modify it so that the last command-line argument is used as the input file. You will need to use the open function to open a file handle and use the angle brackets with that file handle instead of `STDIN`.

Task 4. Strings and files

Write a Perl script to remove anything that looks like an XML tag from a file. An XML tag consists of angle brackets (<>) surrounding a name and, optionally, one or more attributes of the form *name=value*. The name may be preceded by a slash.

You do not have to worry about whether opening and closing tags match; just remove anything that looks like a tag.

Test your script – if you're feeling uninspired, you'll find a good source of XML-like data in web page source files.

Task 5. Manipulating files and directories, part I

A *file extension* is, by convention, the part of a file name after the first dot "." character in a filename. In Unix, file extensions are purely a convention and are not enforced by the operating system. For the purposes of this question, the extension includes the leading dot. For instance, the file extension of the file `my_file.tar` is `.tar`, and the extension of the file `my_file.tar.gz` is `.tar.gz`.

Write a Perl script called `fext-counter` whose single command-line argument is a directory. For all files in that directory and any subdirectories of that directory (and so on, recursively), print a list of all of the file extensions, and how many files have that extension. If a file has no extension, ignore it. The output should be sorted into alphabetical order. Use a hash to keep the count.

For example, if you've got files called `fred.c`, `fred.h` and `mainfred.c` in a directory called `/home/xubuntu/FIT3042/example`, and no subdirectories under that directory, then running `fext-counter /home/xubuntu/FIT3042/example` should produce the following output:

```
.c: 2
.h: 1
```

Task 6. Manipulating files and directories, part II

Based on your code for `fext-counter`, write a Perl script called `fext-lister` and modify it so that rather than a count of how many files particular extensions have, print the *base name* of each file, without the directory prefix and extension. They should appear in alphabetic order, regardless of the directory in which they appear. For instance, the output from running `fext-lister /home/xubuntu/FIT3042/example`, with the contents as described in the previous Task, should be:

```
.c: fred mainfred
.h: fred
```