Assignment 2: Perl Scripts

Due: Monday, Feb 12, 8pm.

1 Introduction

Perl’s strength as a scripting language lies in its ability to process text. It has become popular in the Web programming world for that reason. Many forms you fill out on the web have a Perl program behind them reading and processing the data you enter. Perl is also heavily used by system administrators and others who need to examine and summarize large log files.

In this assignment you will write a filter that reads a Usenet archive and produces summary information at the end. In some ways you will be producing a more sophisticated form of the mailstats program from Assignment 1b. You will get practice using pattern matching, arrays, hashes, and file I/O.

2 Tasks

Write a Perl script that takes as an argument a file containing a Usenet archive of messages from one newsgroup. The script, called newsinfo.pl will read through the file, one line at a time and produce the following output:

- A file called MessagesPerDay that contains in chronological order, a date and the number of messages that arrived that day. If no messages arrived on a given day, no line for that day will appear in the file. The format of a line is “YYYY:MM:DD <num>” where YYYY is the year, MM is the month, DD is the day, and <num> is the number of messages for that day. For example:

  2000:11:30 4
  2000:12:11 3
  2000:12:14 1
  2001:01:13 6

- A file called MessagesPerUser that contains the number of messages sent by each user. Each line should have the format <number> <email address>. For example:

  3 reid@cs.utoronto.ca
  100 ta@eeg.utoronto.ca
  1 zoot@sympatico.ca

  The file should be ordered in alphabetically by email address, or for bonus marks, order the file in ascending order by number of messages sent.

- Print to standard output a summary message with the format as illustrated below. The underlined words are values filled in by your script. Only one decimal place should trail the average number of lines per message.
Summary information for the news archive 209archive:

There were 409 messages from 104 different email addresses

Number of Lines per message:
\[\text{max} = 251, \; \text{min} = 2, \; \text{avg} = 18.6\]

Reply depths:
- 213 messages were original subjects
- 163 were at reply depth 1
- 22 were at reply depth 2
- 11 were at reply depth 3

A sample archive file is found on the web page. It may not produce the same results shown in the examples on this handout.

3 Details

The archive file contains news messages concatenated into one file. A new message begins when a new header begins. The headers follow the standard Usenet format described in the Appendix.

Your script must work under the -w command line argument and must “use strict”. Your script will read each line of the archive file once. This means that you must build up Perl data structures as you read the file. It is recommended that you break down the problem into the following steps, testing each one before you move onto the next:

1. If the user tries to run your script with the wrong number of arguments, your script should print out the message:

   Usage: newsinfo.pl <archive file name>

2. Write the code to read in each line of the file pattern match the headers you need. Test this to be sure you are matching all the headers.

3. Extract the specific information you need for each header:

   (a) The “From” line must match on 3 possible formats, and you want to extract the email address. If the “From” line is incorrectly formatted, set the email address for that message to “someone@do.not.know”. Do not print out an error message.

   (b) The “Date” line has two possible formats, and you want to extract the year, month and day in a format that makes it easier to sort (see above for exact format). You will want to declare a hash to convert the month abbreviation into a 2-digit string (include the leading 0).

   (c) The “References” may contain any number of message-IDs, so think about how to match it and how to store the message-IDs. The reply depth is determined by the number of message-IDs in this line. An original message has no “References” line, a message with a reply depth of k has k message-IDs in this line.
4. Now that the important headers have been matched, what other lines need to be matched specifically? How do we tell when a message begins or ends?

5. Determine how to count the number of messages in the archive.

6. Determine how to count the number of lines in each message. The number of lines should **not** include the headers but will include any blank lines. (If you have figured out how to tell when a message begins, how do you know when the headers are finished?) This will take some thought.

7. Now add the data structure to count the messages per day as each date is matched. Add the loop to print the number of messages per day to a file. Remember to close the file when you are finished.

8. Next work on counting the number of messages per email address.

9. The next task is counting the number of messages at each reply depth.

10. Finally, make sure that all the correct summary lines are printed to standard output.

**What to submit**

You will submit electronically the file `newsinfo.pl`. The file must contain as its first line the “#!” line giving the absolute path the the Perl executable on CDF. (Use “which perl” to find the path.) Immediately below this line should be commented lines containing your name and student number.

```
submit -N A2 csc209h newsinfo.pl
```

This assignment will be marked both by hand and electronically. Part of the marks will be for comments and style. The comments should make it easier to read your program. You will not be using subroutines so it is important to have good comments to separate each logical step. We will be looking for properly indented code and good choice of variable names. Good Perl style in this course means not trying to do everything on one line. Using temporary variables is usually a better choice than trying to combine several function calls on the same line.
4 Appendix: Excerpts from “RFC850: Standard for Interchange of USENET Messages”

All of the information you need about the format of Usenet headers is excerpted below from RFC850. The full RFC may be found at http://www.landfield.com/rfcs/rfc850.html.

4.1 Required Headers

4.1.1 From

The From line contains the electronic mailing address of the person who sent the message, in the ARPA internet syntax. It may optionally also contain the full name of the person, in parentheses, after the electronic address. The three permissible forms are:

- From: mark@cbosgd.UUCP
- From: mark@cbosgd.UUCP (Mark Horton)
- From: Mark Horton <mark@cbosgd.UUCP>

4.1.2 Date

The Date line is the date that the article was originally posted to the network. (Many formats of the data are permissible. The formats of the Date line that we care about are below.)

- <Wdy, DD Mon YYYY HH:MM:SS TIMEZONE> e.g., Wed, 24 Jan 2001 13:50:56 GMT
- <DD Mon YYYY HH:MM:SS TIMEZONE> e.g., 25 Jan 2001 21:06:52 GMT

4.1.3 Subject

The Subject line tells what the article is about. It should be suggestive enough of the contents of the article to enable a reader to make a decision whether to read the article based on the subject alone. If the article is submitted in response to another article (e.g., is a “followup”) the default subject should begin with the four characters “Re: ” and the References line is required.

4.1.4 Message-ID

The Message-ID line gives the article a unique identifier. In order to conform to RFC 822, the Message-ID must have the format

“<” “unique” “@” “full domain name” “>”

where “full domain name” is the full name of the host at which the article entered the network, including a domain that host is in, and unique is any string of printing ASCII characters, not including “<”, “>”, or “@”.

4.2 Optional Headers

4.2.1 References

This field lists the message ID’s of any articles prompting the submission of this article. It is required for all follow-up articles, and forbidden when a new subject is raised.