Perl Assignments 2013 LP4

Patrik Arlos
Blekinge Institute of Technology,
Karlskrona - Sweden,
`patrik@arlos@bth.se`

October 30, 2013

1 Introduction

These assignments will test that you full fill the requirement for Perl. You need to search and user information found on the Internet to solve the assignments. However, when you find information, you need to understand it, simply copy and paste will not help you (rather the opposite, as copying is not allowed). Plan your time, survey the assignments before you start looking for specific information.

Each assignment should be named as *taskX*, e.g. *task1a*. Sharing files between tasks is ok, in which case they can be named anything. If command-line arguments is required the application must support verify that the correct number of arguments were used, and if not show a description of the arguments and must show a friendly error message if the wrong number of arguments is given. The code should be reasonably documented, i.e. it should be possible to read out how to use/operate the programs from the comments in the file. Once the code has been reviewed you will be called for a demo.

Part I
File I/O

These four assignments test that you can read and write ASCII and binary files, use regular expressions and do simple arithmetic.

2 Assignment 1

Write a program to display the contents of a given ASCII file on the command line. The program should be given the filename as an input argument. After you display contents of the file, print the number of lines and characters in the file (compare your output to that of `wc`).

3 Assignment 2

Write a program to read the contents of an ASCII file (provided as a command line argument), filters each line using a regular expression (provided as command line argument), and if the line matches, writes that line to a file, filename provided via command line. I.e. the program should be called like: `extract sourcefile pattern destination`. You can compare your solution to `grep pattern sourcefile >destination`.

4 Assignment 3

Read the ID3v2 tag from the provided MP3 file, print all information. Write a program that accepts a file as input.

5 Assignment 4

Edit the artist and title of MP3 files. The program should have three inputs; MP3file NewArtist NewTitle. Use A3 to verify that the write operation worked.

Part II
Network Communications

These three assignments will test that you can use Perl to communicate over UDP, TCP and interact with a SNMP device. The TCP and UDP assignments are identical to the C/C++ assignments, while the SNMP is unique for Perl.

6 Assignment 5

Hint: use `netcat`(nc) to test both client and server.

6.1 a) TCP Chat Client

Implement a simple chat-client. Must be able to set IP:PORT and nickname at runtime, e.g. using command-line arguments. See protocol description below. There is a testing-server at 194.47.151.124:4711. For user-input you should read non-blocking without echo.

6.2 a) TCP Chat Server

Implement a chat-server which is compatible with the chat-client. It must be able to handle multiple users simultaneous

6.3 Chat Protocol

Client		Server
Connect()	−− >	
	< −−	Hello version
NICK nick	−− >	
	< −−	OK/ERROR text
MSG text	−− >	
	< −−	MSG nick text/ERROR text

Nicknames is limited to 12 characters (A-Za-z0-9_) and messages is limited to 255 characters (any characters except control characters and newlines, utf-8 encoding). Each message is followed by newline (CR).

7 Assignment 6

7.1 a) UDP SNTP Client

Implement a SNTP (Simple Network Time Protocol) client. Only has to support version 4 in unicast mode.

7.2 a) UDP SNTP Server

Implement a rudimentary SNTP server. Only has to support version 4 in unicast mode.

8 Assignment 7

Implement a program that uses Net::SNMP to query a SNMP version 1 device. The program should display the sysDescr and sysUpTime variables. The program should be given (via command line arguments), the IP and community of the device to probe. If the IP is given in the IP:port format, the program should use the provided port when communicating to the SNMP device. When developing, you can test against 194.47.151.124 on port 1161 using *ups1* as community string.

You can compare your solution to snmpget (use `man` to figure out how it operates).

```
pal:~\$ snmpget -c ups1 -v1 194.47.151.124:1161 \
  SNMPv2-MIB::sysUpTime.0 SNMPv2-MIB::sysDescr.0
SNMPv2-MIB::sysUpTime.0 = Timeticks: (230771181)\
 26 days, 17:01:51.81
SNMPv2-MIB::sysDescr.0 = STRING: APC Web/SNMP \
Management Card (MB:v3.8.6 PF:v3.5.5 PN:apc_hw02_aos_355.bin \
AF1:v3.5.5 AN1:apc_hw02_sumx_355.bin MN:AP9619 \
HR:A10 SN: NA0737015853 MD:09/13/2007) (Embedded \
PowerNet SNMP Agent SW v2.2 compatible)
```