Lab 1 – DV1357 (*DV1305*) Programming in the UNIX environment (*Stefan Axelsson*)

1. Background

In this lab you will write a script that processes a log file and extracts useful information summaries from it, a task that is very common in security and system administration work. This is also a task that scripting languages shine at (Indeed, the Perl language is named after this task "Practical Extraction and Report Language).

The log file in question is a web server log file from the small, portable and secure webserver thttpd ***(http://www.acme.com/software/thttpd***). It follows a standard format that is very common in the UNIX world, that is, it's a flat text file with one record per line and fields delimited by whitespace or other characters.

Your task is to write a script that answers the following questions, the idea is that this script will be used by sys admins to extract and summarise relevant information about the operation of the webserver so that they can check for anomalies, i.e. things that stand out from the ordinary, and investigate further if necessary. This is in fact a very simple and common form of intrusion detection (just reading the logs won't work, there's just too much).

The **logfile** consists of lines where each line represents one access request. It pretty much follows the unified apache combined log format. The fields, describe the following: IP number the request originated from; the ident answer from the originator (always '-'); the username of the requester as determined by http authentication; date and time the request was processed; the request line as it was received from the client in double quotes; http status code that was sent to the client; number of bytes that was transferred to the client; referrer page (from the client); user agent string (from the client). See e.g. the Apache documentation for more details ***(http://httpd.apache.org/docs/1.3/logs.html).*** Note that the fields are not well delimited (i.e. there is no reserved character to separate fields), this is unfortunately common when it comes to log files and a problem you have to contend with.

2. The tasks

You will write a shell script (and a Python program if you so choose) that reads a log file and answers the following questions:

1. Which IP addresses makes the most number of connection attempts?
2. Which IP addresses makes the most number of successful connection attempts?
3. What are the most common result codes and where do they come from (IP number)?
4. What are the most common result codes that indicate failure (no auth, not found etc) and where do they come from?
5. Which IP number gets the most bytes sent to them?
6. Which IP number sends the most bytes to the server?

For all the above the answer should be in the form of a sorted list with the largest number first (i.e. topmost). The user should be able to cap all answers, i.e. by asking for only the 'x' topmost results. With no cap all available answers should be presented.

Also all questions should be able to be qualified by time range, i.e. in last number of hours or days. You are allowed to assume that hours will be less than 24 and that the day changes at midnight (***so "the last 24 hours" and "the last day" doesn't necessarily refer to the same period of time, the hours count from now, and the day counts back until the last midnight***). All times are relative to the last time in the **logfile**, not the time when the log analysing program is run. Queries without a range specified are assumed to refer to time span represented in the whole log file.
The arguments to the shell script should be coded as such:

- `log_sum(.sh|.py) [-n N] [-h H|-d D] [-c|-2|-r|-F|-t|-f] <filename>`
- `log_sum is the name of the script; .sh means a bourne shell script, .py means a python script`
- **`-n`**`: Limit the number of results to N`
- `-h: Limit the query to the last number of hours (< 24)`
- `-d: Limit the query to the last number of days (counting from midnight)`
- `-c: Which IP address makes the most number of connection attempts?`
- `-2: Which address makes the most number of successful attempts?`
- `-r: What are the most common results codes and where do they come from?`
- `-F: What are the most common result codes that indicate failure (no auth, not found etc) and where do they come from?`
- `-t: Which IP number get the most bytes sent to them?`
- `-f: Which IP number sends the most bytes to the server`
- `<filename> refers to the logfile. If '-' is given as a filename, or no filename is given, then standard input should be read. This enables the script to be used in a pipeline.`

In the above '|' denotes choice, one of. Furthermore, '[]' denotes that all within the square brackets is optional. So e.g. "(.sh|.py)" means that the two allowed forms of program name are log_sum.sh and log_sum.py. "[-h H|-d D]" means that either '-h' or '-d' or neither (choice within optional brackets) can be given, but not both.

The output format should be in the form:
-c: xxx.xxx.xxx.xxx yyy where yyy is the number of connection attempts
-2: xxx.xxx.xxx.xxx yyy where yyy is the number of successful attempts
-r: yyy xxx.xxx.xxx.xxx where yyy is the result code, one ip per line
-F: yyy xxx.xxx.xxx.xxx where yyy is the result code indicating failure, one ip per line
-t: xxx.xxx.xxx.xxx.xxx yyy where yyy is the number of bytes sent from the server
-f: xxx.xxx.xxx.xxx.xxx yyy where yyy is the number of bytes sent to the server

xxx.xxx.xxx.xxx is the IP address in dotted quad form. Yyy is an integer (with between 1 and the required number of digits, i.e. not just three digits). You are allowed to insert tabs and or whitespace between the elements of the output according to taste. (But not within the elements, i.e. within an IP address). For -F and -r you are allowed to output multiple lines with the same result code or count, but the groups must still be sorted. There should still only be one IP address per line.

3. Example usage:

```
$log_sum.sh -n 10 -c thttpd.log
213.64.237.230    2438
213.64.225.123    1202
213.64.141.89      731
213.64.64.53       591
213.64.214.124     480
213.64.55.182      429
213.64.246.37      400
213.64.153.92      336
213.64.100.52      336
213.64.19.224      272
```

That is to say, we're asking for a sorted list of the top ten most connexion originating IP addresses, most prolific first.. The above example usage can be used as a test case as it is correct given the thttpd.log file you have been supplied with.

4. Presentation and limitations

The assignment should be presented in person with both group members in attendance on a lab slot. You will be required to show the code, let me run it with the test file you have been given and be prepared to discuss your work.

Note:

- You're allowed to use all tools available to the shell script programmer.
- You're allowed to use the full capabilities of the Python language, together with the official standard libraries of the C-python distribution.

5. Grading

You should demonstrate a clear understanding of the problem and detail you strategy of how to solve it, including drawbacks and advantages of the solution chosen. Your code should be clear, concise and to the point.

You are allowed to discuss problems and solutions with other groups, but the code that you write must be your own and cannot be copied, gleaned, downloaded from the internet etc. (You may of course copy idioms and snippets, such as “awk '{print $1}' “verbatim).

Table 1: Marking of the assignment

Grade	**Program implemented in:**
A	Bourne Shell + Awk + Python
C	Bourne Shell + Awk
E	Bourne Shell
For final grade calculation, see the course descriptor	

Created on: 2009-09-16 **Updated on:** 2013-09-03