

Lab Report 3

Regular Expressions

Submitted By:	Hassan R. Al-Saffar
Student ID:	10190069
Section:	L02
Instructor:	Prof. Jeremy Dalby
Submission Date:	2017-Oct-01

INTRODUCTION

Learning about Perl regular expressions and command line parameters.

ACTIVITIES

ACTIVITY 1

The first step of this lab is to download a log file. This file contains logs from a gateway computer, showing packets intercepted from a firewall.



Figure 1

This file will be used for the exploration programs I will create, to find specific logs in this file.

ACTIVITY 2

The second step was to create a file processing loop program. As shown below in figure 2, the program opens the log file, and the reads through the file one record at a time. Then prints the file count at the end.

```

C:\> Command Prompt Portable
Record 2894: Jan 29 15:45:05 myth kernel: SFW2-FWDint-ACC-FORW IN=eth0 OUT=ppp0 SRC=192.168.17.2
4 DST=192.168.9.51 LEN=235 TOS=0x00 PREC=0x00 TTL=63 ID=0 DF PROTO=UDP SPT=138 DPT=138 LEN=215
Record 2895: Jan 29 15:45:05 myth kernel: SFW2-FWDint-ACC-FORW IN=eth0 OUT=ppp0 SRC=192.168.17.2
4 DST=192.168.10.60 LEN=235 TOS=0x00 PREC=0x00 TTL=63 ID=0 DF PROTO=UDP SPT=138 DPT=138 LEN=215
Record 2896: Jan 29 15:45:05 myth kernel: SFW2-FWDint-ACC-FORW IN=eth0 OUT=ppp0 SRC=192.168.17.2
4 DST=192.168.9.51 LEN=204 TOS=0x00 PREC=0x00 TTL=63 ID=0 DF PROTO=UDP SPT=138 DPT=138 LEN=184
Record 2897: Jan 29 15:45:05 myth kernel: SFW2-FWDint-ACC-FORW IN=eth0 OUT=ppp0 SRC=192.168.17.2
4 DST=192.168.10.60 LEN=204 TOS=0x00 PREC=0x00 TTL=63 ID=0 DF PROTO=UDP SPT=138 DPT=138 LEN=184
This file contains 2897 records
  
```

Figure 2

This program tells me that this file contains 2897 total records.

ACTIVITY 3

In this version of the program I introduced a regular expression into the source code. The purpose of this is to find a specific set of strings that we want. I accomplished this by putting an "If statement" into the loop. We also move the counter into the statement because, we only want matching strings to be incremented.

```

C:\> Command Prompt Portable
Record 331: Jan 29 13:26:33 myth sshd[12490]: Invalid user gruiz from 220.195.35.40
Record 332: Jan 29 13:26:37 myth sshd[12492]: Invalid user josed from 220.195.35.40
Record 333: Jan 29 13:26:42 myth sshd[12494]: Invalid user kop from 220.195.35.40
Record 334: Jan 29 13:26:46 myth sshd[12496]: Invalid user lady from 220.195.35.40
Record 335: Jan 29 13:26:50 myth sshd[12498]: Invalid user mabad from 220.195.35.40
Record 336: Jan 29 13:26:54 myth sshd[12500]: Invalid user osilvera from 220.195.35.40
Record 337: Jan 29 13:26:58 myth sshd[12502]: Invalid user patriciar from 220.195.35.40
Record 338: Jan 29 13:27:02 myth sshd[12504]: Invalid user porteria from 220.195.35.40
This file contains 338 records

```

Figure 3

As a result, we are shown with the text: “This file contains 338 records”. This is because only the records shown in figure 3 matched our regular expression in our if statement.

ACTIVITY 4

In this exercise we want to be able to explore the file without editing the source code every time we change our search. To do this we changed our regular expression from `/sshd/` to `/$ARGV[0]/`.

```

C:\> Command Prompt Portable
D:\Documents\lab3>perl lab3_v3.pl sshd
This file contains 338 records

```

Figure 4

This works because the scalar `$ARGV[0]` is a reference to the first element of an array. `@ARGV` contains any strings inputted from the command line. This in turn passes as a regular expression in the command line. This is shown in figure 4. This is also called a command line parameter.

```

C:\> Command Prompt Portable
Record 331: Jan 29 13:26:33 myth sshd[12490]: Invalid user gruiz from 220.195.35.40
Record 332: Jan 29 13:26:37 myth sshd[12492]: Invalid user josed from 220.195.35.40
Record 333: Jan 29 13:26:42 myth sshd[12494]: Invalid user kop from 220.195.35.40
Record 334: Jan 29 13:26:46 myth sshd[12496]: Invalid user lady from 220.195.35.40
Record 335: Jan 29 13:26:50 myth sshd[12498]: Invalid user mabad from 220.195.35.40
Record 336: Jan 29 13:26:54 myth sshd[12500]: Invalid user osilvera from 220.195.35.40
Record 337: Jan 29 13:26:58 myth sshd[12502]: Invalid user patriciar from 220.195.35.40
Record 338: Jan 29 13:27:02 myth sshd[12504]: Invalid user porteria from 220.195.35.40
This file contains 338 records

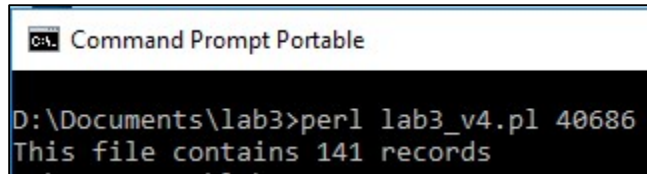
```

Figure 5

In our program we also added `$ARGV[1]`, this makes it so the record will print if anything is contained in the second parameter. The result shown in figure 5 prints the records just like before.

ACTIVITY 5

In this activity we print substrings using the new version of code we crated last activity.



```

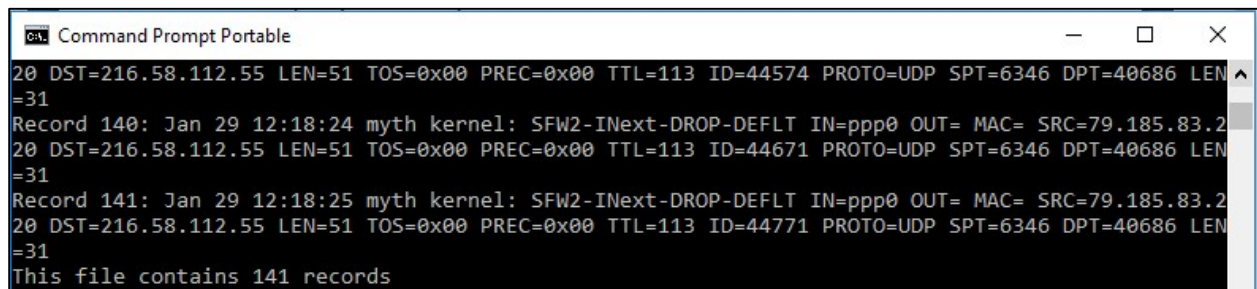
C:\> Command Prompt Portable

D:\Documents\lab3>perl lab3_v4.pl 40686
This file contains 141 records
  
```

Figure 6

In figure 6 we use our program to search for the string 40686. This results in the retrieval of 141 records.

In figure 7 we try printing the retrieved records. We find that the records are too long to fit. So next we created a length variable in the second line parameter.



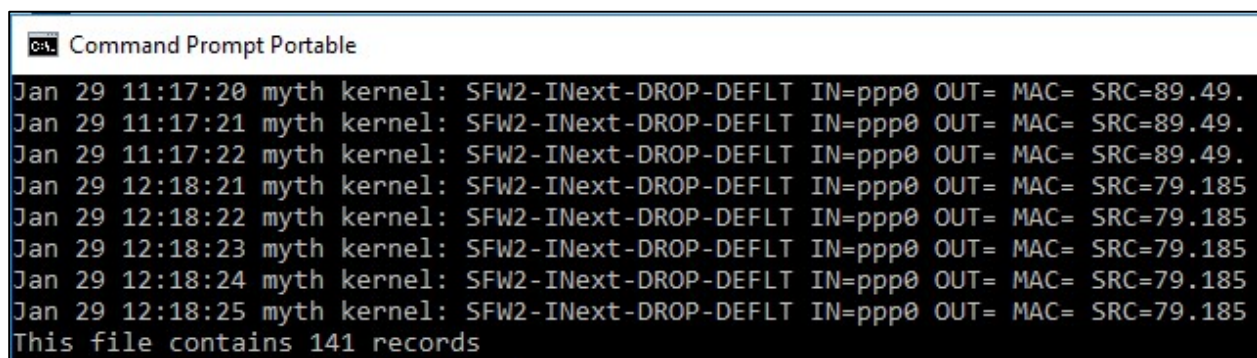
```

C:\> Command Prompt Portable

20 DST=216.58.112.55 LEN=51 TOS=0x00 PREC=0x00 TTL=113 ID=44574 PROTO=UDP SPT=6346 DPT=40686 LEN=31
Record 140: Jan 29 12:18:24 myth kernel: SFW2-INext-DROP-DEFLT IN=ppp0 OUT= MAC= SRC=79.185.83.2
20 DST=216.58.112.55 LEN=51 TOS=0x00 PREC=0x00 TTL=113 ID=44671 PROTO=UDP SPT=6346 DPT=40686 LEN=31
Record 141: Jan 29 12:18:25 myth kernel: SFW2-INext-DROP-DEFLT IN=ppp0 OUT= MAC= SRC=79.185.83.2
20 DST=216.58.112.55 LEN=51 TOS=0x00 PREC=0x00 TTL=113 ID=44771 PROTO=UDP SPT=6346 DPT=40686 LEN=31
This file contains 141 records
  
```

Figure 7

Now with our new print statement reflecting our desires, the second command line now tells the program how much to print.



```

C:\> Command Prompt Portable

Jan 29 11:17:20 myth kernel: SFW2-INext-DROP-DEFLT IN=ppp0 OUT= MAC= SRC=89.49.
Jan 29 11:17:21 myth kernel: SFW2-INext-DROP-DEFLT IN=ppp0 OUT= MAC= SRC=89.49.
Jan 29 11:17:22 myth kernel: SFW2-INext-DROP-DEFLT IN=ppp0 OUT= MAC= SRC=89.49.
Jan 29 12:18:21 myth kernel: SFW2-INext-DROP-DEFLT IN=ppp0 OUT= MAC= SRC=79.185
Jan 29 12:18:22 myth kernel: SFW2-INext-DROP-DEFLT IN=ppp0 OUT= MAC= SRC=79.185
Jan 29 12:18:23 myth kernel: SFW2-INext-DROP-DEFLT IN=ppp0 OUT= MAC= SRC=79.185
Jan 29 12:18:24 myth kernel: SFW2-INext-DROP-DEFLT IN=ppp0 OUT= MAC= SRC=79.185
Jan 29 12:18:25 myth kernel: SFW2-INext-DROP-DEFLT IN=ppp0 OUT= MAC= SRC=79.185
This file contains 141 records
  
```

Figure 8

In the above figure we tell the program to look for string 40686 and 79 characters. The result is the information we want as well neat presentation.

ACTIVITY 6

In this activity we select what we want Perl to print, to gather the source IP as well as the source port. Bellow in figure 9 I explain each change to the source code to reflect this new objective.

```

1  use warnings;
2  use strict;
3  open (my $file, 'sample.log') or die $!;
4  my $lineno = 0;
5  while (my $line = <$file>)
6  {
7      if ($line =~ /$ARGV[0]/)
8          #From previous versions of this source code. We have changed our original
9          #regular expression from /sshd/ to /$ARGV[0]/. This expression is a
10         #reference to an array. @ARGV can contain any string inputted into command prompt
11         #to give the program a parameter. This makes it easy to print records
12         #containing the strings we want, without changing the source code.
13         {
14             $lineno++;
15             if ($ARGV[1])
16                 #This if statement tells our program how much to print in the command
17                 #prompt. This means if the value is greater than 0, the number inputted will
18                 #reflect the amount of characters shown in the command prompt. If zero is
19                 #inputted, the if statement will not be true and will skip this command.
20                 {
21                     $line =~ /(SRC=[0-9\.]* )*(SPT=[0-9\.]* )/;
22                     my $src_ip = $1;
23                     my $src_port = $2;
24                     #In lines 22 & 23 we assign "SRC" ($src_ip) to the scalar variable $1.
25                     #We also do the same for "SPT" ($src_port) to $2. By doing this we
26                     #store the source port and source IP information extracted by the regular
27                     #expression into $1 and $2, to variables $src_ip & $src_port.
28                     print $src_ip, $src_port, substr($line, 0, $ARGV[1]), "\n";
29                     #This tell Perl if the following statement is true, what to print.
30                     #This is useful if we want our source program show use the specific
31                     #strings we want, /SRC=[0-9\.]* / matches "SRC=" followed by any number of
32                     #characters followed by a space. This is also true for /SPT=[0-9\.]* /
33                 }
34             }
35     }
36     print "This file contains $lineno records";
37     close $file;

```

Figure 9

After running the program, we get our result in figure 10, we get this because we tell Perl to print SPT and the following numerical up to a space as well as the SRC.


```

C:\> Command Prompt Portable
SRC=89.49.78.88 SPT=61731 Jan 29 11:17:19 myth kernel: SFW2-INext-DROP-
SRC=89.49.78.88 SPT=61731 Jan 29 11:17:20 myth kernel: SFW2-INext-DROP-
SRC=89.49.78.88 SPT=61731 Jan 29 11:17:21 myth kernel: SFW2-INext-DROP-
SRC=89.49.78.88 SPT=61731 Jan 29 11:17:22 myth kernel: SFW2-INext-DROP-
SRC=79.185.83.220 SPT=6346 Jan 29 12:18:21 myth kernel: SFW2-INext-DROP-
SRC=79.185.83.220 SPT=6346 Jan 29 12:18:22 myth kernel: SFW2-INext-DROP-
SRC=79.185.83.220 SPT=6346 Jan 29 12:18:23 myth kernel: SFW2-INext-DROP-
SRC=79.185.83.220 SPT=6346 Jan 29 12:18:24 myth kernel: SFW2-INext-DROP-
SRC=79.185.83.220 SPT=6346 Jan 29 12:18:25 myth kernel: SFW2-INext-DROP-
This file contains 141 records
  
```

Figure 10

We tell Perl to print the SRC and the SPT as well as 45 additional characters to show the dates of each log or in better words when these processes transpired.