

# 2019 oplossingen labo 3 - Lars Lemmens

Met dank aan de [Github van Martijn](#) en natuurlijk Lars Lemmens

## Labo 3

Match one or more d characters: d  
dd ddd dddd "

```
'user@:~$' • grep -P 'd+' file
```

```
'user@:~$' • cat file | grep -P 'd+'
```

- d matches the character d literally (case sensitive)
- The grep command prints lines matching a pattern
- The -P argument interprets PATTERN as a Perl regular expression

Color and colour are different spellings of the same word.

Color is the preferred spelling in American English, and colour is preferred in all other main varieties of English. The following regex matches both. "

```
'user@:~$' • grep -P 'colou?r' file
```

- colo matches the characters colo literally (case sensitive)
- u matches the character u literally (case sensitive)
- ? matches the previous token between zero and one times, as many times as possible, giving back as needed (greedy)
- r matches the characters r literally (case sensitive)

## Match whe following by one, two or three e character: 'whe whee wheee'

```
'user@:~$' • grep -P 'whe{1,3}' file
```

- The grep command prints lines matching a pattern
- The -P argument interprets PATTERN as a Perl regular expression
- whe matches the characters whe literally (case sensitive)
- {1,3} matches the previous token between 1 and 3 times, as many times as possible, giving back as needed (greedy)

## Match either gray or grey

```
'user@:~$' • grep -P '{gray|grey}' file
```

- The grep command prints lines matching a pattern
- The -P argument interprets PATTERN as a Perl regular expression
- '{gray|grey}' must match gray or gray

Let's say you want to match a text like !abc! or !123!. Only these two are possible, and you want to capture the abc or 123.

```
'user@:~$' • grep -P '!(abc|123)!' file
```

- The grep command prints lines matching a pattern
- The -P argument interprets PATTERN as a Perl regular expression

- '{gray|grey}' must match !abc! or !123!

Search /usr/share/dict/dutch for all words which end in 'vuur', without matching the word 'vuur' itself.

```
'user@:~$' • grep -P '^.+vuur$' /usr/share/dict/dutch
```

- ^ asserts position at start of a line
- . matches any character (except for line terminators)
- matches the previous token between one and unlimited times, as many times as possible, giving back as needed (greedy)
- vuur matches the characters vuur literally

List all words which contain 32 or more letters.

```
user@:~$' • grep -P '^.{32}$' /usr/share/dict/dutch
```

- ^ asserts position at start of a line
- . matches any character (except for line terminators)
- {32} matches the previous token exactly 32 times
- \$ asserts position at the end of a line

List all words starting with the letter 'b', ending with the letter 'l' which are exactly 4 letters long. Capitalize the output

```
'user@:~$' • grep -E '^b.{2}l$' /usr/share/dict/dutch | tr [:lower:] [:upper:]
```

```
'user@:~$' • perl -ne 'print uc if /^b.{2}l$/' /usr/share/dict/dutch
```

- ^ asserts position at start of a line
- b matches the character b literally (case sensitive)
- . matches any character (except for line terminators)

- {2} matches the previous token exactly 2 times
- l matches the character l literally (case sensitive)
- \$ asserts position at the end of a line
- The perl command is how to execute the Perl interpreter
- The -n argument causes Perl to assume the following loop around your program, which makes it iterate over filename arguments somewhat like sed -n or awk:
- The -e argument may be used to enter one line of program.

## List all palindromes with exactly 4 characters

```
'''
'user@:~$' • grep -P '^([A-Za-z])([A-Za-z])\2\1$' /usr/share/dict/dutch

'user@:~$' • perl -lne 'print if $_ eq reverse && length($_) eq 4' /usr/share/dict/dutch
'''
```

- A-Z matches a single character in the range between A and Z (case sensitive)
- a-z matches a single character in the range between a and z (case sensitive)
- \2 matches the same text as most recently matched by the 2nd capturing group
- \1 matches the same text as most recently matched by the 1st capturing group
- \$ asserts position at the end of a line

## Exercise 1:

List all words out of the file /usr/share/dict/dutch which contain 4 or 5 consecutive vowels.

```
'''
'user@:~$' • cat /usr/share/dict/dutch | grep -P '[aeiou]{4,5}'
'''
```

- aeiou matches a single character in the list aeiou (case sensitive)

- {4,5} matches the previous token between 4 and 5 times, as many times as possible, giving back as needed (greedy)

## Exercise 2:

Count the number of words out of the file `/usr/share/dict/dutch` which start with 'ver', and end with 'en'. ('verplaatsen' is ok, 'overkomen' and 'vertrek' are not ok)

```
...  
'user@:~$' • grep -P '^ver.*en$' /usr/share/dict/dutch | wc -l  
...
```

- ^ asserts position at start of a line
- ver matches the characters ver literally (case sensitive)
- . matches any character (except for line terminators)
- '\*' matches the previous token between zero and unlimited times, as many times as possible, giving back as needed (greedy)
- \$ asserts position at the end of a line
- The wc command prints newline, word, and byte counts for each file
- The -l argument prints the newline counts

## Exercise 3:

In those annoying night television games people must guess words. Given are all the letters the word consist of and a list of

dots, one for every letter.

E.g. We are looking for a 23 letter word with a 'v' in position 8: '.....v.....'. Use the letters 'enrtiscau' for the other positions."

```
'''
'user@:~$' • cat /usr/share/dict/dutch | grep -P '[enrtiscau]{7}v[enrtiscau]{15}'
'''
```

- enrtiscau matches a single character in the list enrtiscau (case sensitive)
- {7} matches the previous token exactly 7 times
- {15} matches the previous token exactly 15 times

## Exercise 4:

Show 'System load = X Y Z' replacing X, Y and Z by the values displayed by the uptime command.

```
'''
'user@:~$' • echo "System load =" $(uptime | awk -F: ' '{print $2}' | tr -d ',')
'''
```

## Exercise 5:

# List the usernames of the students who logged in from outside of the UCLL network.

```
...  
'user@:~$' • who | grep -Pv '(.*)\s+pts.*\s+(10\.|tmux' | awk '/^r[0-9]/{print $1}' | sort -u  
...
```

- . matches any character (except for line terminators)
- '\*' matches the previous token between zero and unlimited times, as many times as possible, giving back as needed (greedy)
- \s matches any whitespace character (equivalent to [\r\n\t\f\v ])
- r matches the character r literally (case sensitive)
- ◦ matches the previous token between one and unlimited times, as many times as possible, giving back as needed (greedy)
- tmux matches the characters tmux literally (case sensitive)
- ^ asserts position at start of a line
- 0-9 matches a single character in the range between 0 and 9 (case sensitive)

## Exercise 6:

# How many times have students logged into leia from outside the UCLL network?

```
...  
'user@:~$' • last | grep -Pv '.*?pts.*?\s+10\.' | awk '/^r[0-9]/{print $1}' | wc -l  
...
```

- . matches any character (except for line terminators)
- .\*? matches the previous token between zero and unlimited times, as few times as possible, expanding as needed (lazy)
- pts matches the characters pts literally (case sensitive)
- r matches the character r literally (case sensitive)
- \s matches any whitespace character (equivalent to [\r\n\t\f\v ])

- '+' matches the previous token between one and unlimited times, as many times as possible, giving back as needed (greedy)
- '^' asserts position at start of a line

## Exercise 7:

Show the file `/etc/debconf.conf` on screen without comment lines (i.e. lines starting with a `#`)

```
'''  
'user@:~$' • cat /etc/debconf.conf | grep -vP '^#'  
'''
```

- '^' asserts position at start of a line
- '#' matches the character `#` literally (case sensitive)

## Exercise 8:

List all unique IP addresses that contacted your Apache web server.

```
'''  
'user@:~$' • cat /var/log/apache2/wiki-ssl-access.log | awk '{print $1}' | sort -u  
'''
```

## Exercise 9:



List all unique IP addresses that contacted the ssh daemon of your Apache web server.

```
'''
'user@:~$' • cat /var/log/auth.log | grep -oP 'sshd.*?\K[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}' | sort -u
'''
```

- sshd matches the characters sshd literally
- . matches any character (except for line terminators)
- \*? matches the previous token between zero and unlimited times, as few times as possible, expanding as needed (lazy)
- \K resets the starting point of the reported match. Any previously consumed characters are no longer included in the final match
- [0-9]{1,3} matches the previous token between 1 and 3 times, as many times as possible, giving back as needed (greedy)
- 0-9 matches a single character in the range between 0 and 9 (case sensitive)
- . matches the character . literally (case sensitive)

## Exercise 10:

Create a regular expression to match all words in the dictionary /usr/share/dict/dutch which with 10 unique letters.

```
'''
'user@:~$' • cat /usr/share/dict/dutch | grep -P '^.{10}$' | grep -vP '(.)*\1'
'''
```

- ^ asserts position at start of a line
- . matches any character (except for line terminators)

- {10} matches the previous token exactly 10 times
- '\*' matches the previous token between zero and unlimited times, as many times as possible, giving back as needed (greedy)
- \1 matches the same text as most recently matched by the 1st capturing group

## Exercise 11:

To combat spam, a mail server administrator wants to reject mail coming from home users. The IP addresses home users always seem to connect from have hostnames like this: ip.domain. Create a regex which matches all these host names.

```
'''  
'user@:~$' • cat mail.log | grep -oP 'NOQUEUE: reject: RCPT from \K[0-9]{1,3}.*?\[' | tr -d '['  
'''
```

- [0-9]{1,3} matches the previous token between 1 and 3 times, as many times as possible, giving back as needed (greedy)
- . matches any character (except for line terminators)
- \*? matches the previous token between zero and unlimited times, as few times as possible, expanding as needed (lazy)

## Exercise 12:

List all unique firefox 2.0.0.x versions used to connect to [www.khleuven.be](http://www.khleuven.be). Use log file /home/logs/apache\_google.log. "

```
'''
'user@:~$' • cat apache_google.log | grep -oP 'Firefox\2\0\0\.[0-9]+' | sort -n -t '.' -k4 -u
'''
```

- Firefox matches the characters Firefox literally (case sensitive)
- / matches the character / literally (case sensitive)
- 2 matches the character 2 literally (case sensitive)
- . matches the character . literally (case sensitive)
- 0 matches the character 0 literally (case sensitive)
- [0-9]+ matches the previous token between one and unlimited times, as many times as possible, giving back as needed (greedy)

## Exercise 13:

List all words with 14, 15 or 16 unique letters.

```
'''
'user@:~$' • for foo in 14 15 16; do echo "Words with $foo letters:" $(echo "grep -vP '(.)*\1' /usr/share/dict/dutch | grep -P '^.{ $foo }$'" | sh);done

'user@:~$' • for foo in 14 15 16; do echo "Words with $foo letters:" $(grep -vP '(.)*\1' /usr/share/dict/dutch | grep -P '^.{ $foo }$");done

'user@:~$' • for foo in 14 15 16; do echo "Words with $foo letters:" && echo "grep -vP '(.)*\1' /usr/share/dict/dutch | grep -P '^.{ $foo }$'" | sh;done

'user@:~$' • for foo in 14 15 16; do echo "Words with $foo letters:" && grep -vP '(.)*\1' /usr/share/dict/dutch | grep -P '^.{ $foo }$";done
```

- . matches any character (except for line terminators)
- '\*' matches the previous token between zero and unlimited times, as many times as possible, giving back as needed (greedy)
- \1 matches the same text as most recently matched by the 1st capturing group
- ^ asserts position at start of a line
- \$ asserts position at the end of a line
- foo} matches the characters foo} literally (case sensitive) \$ asserts position at the end of a line

Revision #1

Created 17 June 2021 14:13:47 by Jasper G.

Updated 3 December 2021 22:13:09 by Jasper G.