

# 2019 oplossingen labo 0 - Lars Lemmens

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

## LABO 0

### Exercise 24

Create 2 files with random text.  
Then create a third file containing  
the contents of the two other files

```
1) echo 'some random text' > file1  
2) pwgen -n1 > file2  
3) cat file1 file2 > file3
```

- The pwgen command generates passwords which are designed to be easily memorized by humans, while being as secure as possible.
- -n stands for minimum 1 number AND -1 stands for per line 1 password.
- '>' means overwrite a file

### Exercise 25

# List all home directories which are open for other users (group or others should have r,w or x rights)

- `ls -l /home/LDAP/ | grep -v 'drwx-----' | grep -v '^total' | tr -s ' ' | cut -d ' ' -f9`
- `ls -l /home/LDAP/ | grep -v 'drwx-----' | grep -v '^total' | awk '{print $9}'`
- `ls -l /home/LDAP/ | grep -vP '^(drwx-----|total)' | awk '{print $9}'`
- `ls -l /home/LDAP/ | awk '$1 !~ /drwx-----|total/{print $9}'`

- `grep` prints lines matching a pattern.
- `-v` stands for inverted match.
- `-P` stands for perl expression
- `^` matches position just before the first character of the string.
- The `tr` command replaces all occurrences of a character in a file, and print the result.
- `-s` replaces each input sequence of a repeated character that is listed in SET1 with a single occurrence of that character
- The command `cut` removes sections from each line of files
- `-d` use DELIM instead of TAB for field delimiter
- `-f(number)` select only these fields
- `awk '{print $(number)}'` prints the field with the matching number

## Exercise 26

# List all lines in the file `/usr/share/dict/dutch` containing the string 'rare'

- `cat /usr/share/dict/dutch | grep rare`
- `grep rare /usr/share/dict/dutch`

# Exercise 27

Only show the columns with filenames and permissions of the files in your homedirectory. (tip: use ls and cut)

```
• ls -l ~ | grep -v '^total' | tr -s ' ' | cut -d ' ' -f1,9
```

- grep prints lines matching a pattern.
- -v stands for inverted match.
- The tr command replaces all occurrences of a character in a file, and print the result.
- -s replaces each input sequence of a repeated character that is listed in SET1 with a single occurrence of that character
- The command cut removes sections from each line of files
- -d use DELIM instead of TAB for field delimiter
- -f(number) select only these fields

# Exercise 28

Sort the lines from 27 in reverse alphabetical order

```
• ls -l ~ | grep -v '^total' | tr -s ' ' | cut -d ' ' -f1,9 | sort -r -k2
```

- grep prints lines matching a pattern.
- -v stands for inverted match.
- The tr command replaces all occurrences of a character in a file, and print the result.

- -s replaces each input sequence of a repeated character that is listed in SET1 with a single occurrence of that character
- The command cut removes sections from each line of files
- -d use DELIM instead of TAB for field delimiter
- -f(number) select only these fields
- The sort command sort lines of text files
- -r reverse the result of comparisons
- -k start a key at POS1 (origin 1), end it at POS2 (default end of line).

## Exercise 29

Use the command cal to find out on which day of the week your birthday is in 2050 and write your output to a newly created file

```
• ncal -m 6 2050 | grep ' 9 ' | cut -d ' ' -f1 > file  
• cal -N -m 6 2050 | grep ' 9 ' | cut -d ' ' -f1 > file
```

- The command ncal displays a calendar and the date of Easter
- The command cal displays a calendar and the date of Easter
- -m displays the specified month
- The command cut removes sections from each line of files
- -d use DELIM instead of TAB for field delimiter
- -f(number) select only these fields
- '>' means overwrite a file

## Exercise 30

Append the sentence 'THE END' to the file you created in the previous exercise without opening this file in an editor (hence: use a one-liner)

- `echo 'THE END' >> file`

# `>>` makes a file and saves it

## Exercise 31

Create a subdirectory TEST in your homedir. Now create some files in it using this command line (and make sure you understand what happens!)

```
for foo in `seq 1 9`; do touch "file $RANDOM"; done && touch 'file keep me'
```

How can you remove all files except the file “file keep me” with just one 'oneliner'?

```
1) mkdir ~/TEST; cd ~/TEST
2) for foo in {1..9}; do touch "file $RANDOM"; done && touch 'file keep me'
```

- `rm file\ [0-9]*`
- `ls file\ [0-9]* | while read file; do rm "$file";done`
- `ls -l | grep -v 'file keep me' | xargs -d '\n' rm`
- `find . -type f ! -name 'file keep me' -delete`

- The `for` keyword indicates a for loop ==> SYNTAX : `for WORD [ in WORDLIST ... ] ; do ACTIONS ; done`
- The command `touch` creates a new empty file(s) or change the times for existing file(s) to current time
- The `rm` command removes files or directories
- Remove (unlink) the FILE(s)
- The `while` command continuously executes the list list-2 as long as the last command in the list-1 returns an exit status of zero.
- The command `xargs` builds and executes command lines from standard input
- `-d` stands for delimiter ==> This can be used when the input consists of simply newline-separated items, although it is almost always better to design your program to use `--null` where this is possible.
- The `find` command searches for files in a directory hierarchy
- `-f` stands for regular file
- `-delete` stands for Delete files; true if removal succeeded.

## Exercise 32

List the name and permissions of the first 10 directories in `/etc`.

Sample output:

```
drwxr-xr-x /etc/
drwxr-xr-x /etc/alternatives
drwxr-xr-x /etc/apache2
drwxr-xr-x /etc/apache2/conf.d
drwxr-xr-x /etc/apache2/mods-available
drwxr-xr-x /etc/apache2/mods-enabled
drwxr-xr-x /etc/apache2/sites-available
drwxr-xr-x /etc/apache2/sites-enabled
drwxr-xr-x /etc/apt d
rwxr-xr-x /etc/apt/apt.conf.d"
```

```
• ls -l /etc | head -10 | grep -v '^total' | awk '{print $1 " " $9}'
```

- The head command prints the first 10 lines of each FILE to standard output. With more than one FILE, precede each with a header giving the file name. With no FILE, or when FILE is -, read standard input.
- grep prints lines matching a pattern
- -v stands for inverted match.
- awk '{print \$(number)}' prints the field with the matching number

## Exercise 33

Same question as #32, but now also omit all error messages. "

```
• ls -l /etc 2>/dev/null | head -10 | grep -v '^total' | awk '{print $1 " " $9}'
```

- 2>/dev/null is used to redirect to a file
- The head command prints the first 10 lines of each FILE to standard output. With more than one FILE, precede each with a header giving the file name. With no FILE, or when FILE is -, read standard input.
- grep prints lines matching a pattern.
- -v stands for inverted match.
- awk '{print \$(number)}' prints the field with the matching number

## Exercise 34:

List the name and permissions of all files in the /etc directory containing the word 'host' in their

# name. Do this without the use of the commands grep and/or awk."

- `find /etc/ -type f -name '*host*' 2>/dev/null | xargs -d '\n' ls -l # (recursive search)`
- `find /etc/ -maxdepth 1 -type f -name '*host*' 2>/dev/null | xargs -d '\n' ls -l`
- `ls -ld /etc/*host* | tr -s ' ' | cut -d ' ' -f1,9 # (also show files of type directory)`
- `ls -ld /etc/*host* | sed -e '/^d./d' | tr -s ' ' | cut -d ' ' -f1,9`

- The find command searches for files in a directory hierarchy
- -f stands for regular file
- -name stands for the name
- 2>/dev/null is used to redirect to a file
- The command xargs builds and executes command lines from standard input
- -d stands for delimiter ==> This can be used when the input consists of simply newline-separated items, although it is almost always better to design your program to use --null where this is possible.
- -maxdepth stands for the - levels of directories below the starting point
- The command cut removes sections from each line of files
- -d use DELIM instead of TAB for field delimiter
- -f(number) select only these fields
- The tr command replaces all occurrences of a character in a file, and print the result.
- -s replaces each input sequence of a repeated character that is listed in SET1 with a single occurrence of that character

---

Revision #2

Created 17 June 2021 14:06:22 by Jasper G.

Updated 3 December 2021 22:13:08 by Jasper G.