Week 2 study diary

Exercises:

1. Two At Once ! Redirecting stdout to one file and stderr to another:

```
 fuksi@dhcp-asv-103:~$ ls /foObarR 2> ~/LinuxFundam
entals2016/Week2/ls-unsuccessful.txt ~ > ~/LinuxFun
damentals2016/Week2/ls-home-successful.txt.
```

   Hey! What about STDIN?

   From man “cat - concatenate files and print on the standard output”. Cat is copying my stdin feed to my stdout.

2. Pipelines. First count the files from folder:

```
fuksi@dhcp-asv-103:~$ /bin/ls ~ | wc -l
30
```

   Then, redirecting the output to a file and counting the lines of that file:

```
fuksi@dhcp-asv-103:~$ /bin/ls > ~/LinuxFundamentals
```

```
2016/Week2/wcfile.txt | wc -l < ~/LinuxFundamentals
2016/Week2/wcfile.txt
30
```

I’m not sure if i was expected to use cat in that, but it seemed to work, so i went with command above. Command modified to count just the folders in my home directory:

```
fuksi@dhcp-asv-103:~$ ls -d */ > ~/LinuxFundamental
s2016/Week2/wcfile.txt | cat ~/LinuxFundamentals201
6/Week2/wcfile.txt | wc -l
21
```

or

```
    fuksi@dhcp-asv-103:~$ ls -d .*/ */ > ~/LinuxFun
damentals2016/Week2/wcfile.txt | cat ~/LinuxFundame
ntals2016/Week2/wcfile.txt | wc -l
21
```

if one wants to count the hidden directories too.

Filters. Solution to count files matching the filter:

```
fuksi@dhcp-asv-103:~$ ls | grep 'e' | wc -l
12
```

Interlude: Bash:

count-homedir.sh content

```
#!/bin/bash

# Counts the number of files in home directory.

ls ~/ | wc -l
```

And running the script on terminal works:

```
fuksi@dhcp-asv-103:~$ count-homedir.sh
31
```

3. Bash script that lists Exactum-kamera's files and subdirectories from november 2011:

```
 #!/bin/bash

 # Exactum-kamera saves one image every hour. The p
ictures are archived as files in subdirectories. Th
is script lists all files and subdirectories from N
ovember 2011.

 ls -R /cs/home/tkt_cam/public_html/2011/11/*
```

Another script that does the same without listing directories:

```
 #!/bin/bash

 # Exactum-kamera saves one image every hour. The p
ictures are archived as files in subdirectories. Th
is script lists all files, but no directories from
November 2011.

 ls -R /cs/home/tkt_cam/public_html/2011/11/* | gre
p .jpg
```

Script that counts the files that previous script listed:

```
 #!/bin/bash

 # Exactum-kamera saves one image every hour. The p
ictures are archived as files in subdirectories. Th
is script counts picture files (.jpg extension) fro
m subdirectories

 ls -R /cs/home/tkt_cam/public_html/2011/11/* | gre
p .jpg | wc -l
```

Script counting picture files of Exactum-kamera from current month:

```
 #!/bin/bash

 # Exactum-kamera saves one image every hour. The p
ictures are archived as files in subdirectories. Th
is script counts picture files (.jpg extension) fro
m subdirectories on current month.

 ls -R /cs/home/tkt_cam/public_html/$(date +%Y/%m)
| grep .jpg | wc -l
```

4. The Big Brother of ls. Script counting images from current month using find:

```
 #!/bin/bash

 # Exactum-kamera saves one image every hour. The p
ictures are archived as files in subdirectories. Th
is script counts picture files (.jpg extension) fro
m subdirectories on current month using find, date
and wc -l.

 find /cs/home/tkt_cam/public_html/$(date +%Y/%m) -
iname '*.jpg' | wc -l
```

5. Three variables from my shell enviroment listed:

    - BASH : Expands to the full filename used to invoke this

instance of bash.

- PWD : Current working directory.
- RANDOM : Generates a random number between 0 and 32767.

Script echo.sh that echoes back command-line arguments:

```
#!/bin/bash

# This script echos back command line arguments ba
ck to user.

echo "$*"
```

The difference between bash and bash: Script that illustrates variable visibility:

```
#!/bin/bash

# This script illustrates variable visibility by p
rinting process identifiers (PIDs) of the shells.

echo $$
```

Remote invocation: I’m not sure if i shoul’ve used export somehow in this script, but i didn’t because the script is so straight forward.

```
 #!/bin/bash

 # This script takes two command line arguments as its parameters: a hostname and a command. The script will execute command at hostname, and print out the output returned by the host.

 ssh $1 $2
```