FYD500 - Homework VI

The first number of assignments for FYD500, an introduction to linux.

Exercise 1 - Vim/vi

Run *vimtutor* and do lessons 1 to 4. Save the result as a new file with file name vimtutorresult.

Exercise 2 - I/0 redirection

1) Create a file linuxcourselog and send text to the file with looking at it with *tail -f linuxcourselog.* Open a new terminal and send the command to the parent with *echo "sending text to linuxcouselog" >> linuxcourselog*

The text will show up in the other terminal.

2) Run the following commands to determine stadard output and standard error.

grep root /etc/passwd /etc/nofiles So if i run *grep root /etc/passwd 1 > /dev/pts/0* I will receive the result. So it's not an error. The output is:

```
root:x:0:0:root:/root:/bin/bash
```

(In the second terminal)

Since there are no *nofiles* in /etc it will generate an error. To view it in the other terminal one should use 2 instead of 1. Ie.

```
grep root /etc/nofiles 2 > /dev/pts/0
grep: /etc/nofiles: No such file or directory
```

The command: *cat nonexistingfile* will generate two errors so I redirect it with 2.

Last command *file /sbin/ifconfig 1> /dev/pts/0* will send the output to the psuedo terminal while a two will print the stdout in the normal terminal since there aren't any errors.

3) Predict what will happen if you type *>time ; date >> time ; cat <time*

It will create the file time, append the output from date and then cat will display the stream output from the file time.

4) To find all processes that *backefel* is the owner fore, use *ps -aef | grep backefel.* All other processes can be found with

```
*ps -aef | grep -v backefel*
*ps -eo user,pid | grep backefel*
```

for a cleaner list

5) What happens when you type *exec 9>somefile 1>&9* then run ls -la? Basically I create a descriptor 9. The commands that are then redirected to descriptor 9 are written to "somefile". All outputs from stdout are then redirected to descriptor 9.

Exercise 3 - Regular expressions

Display a list of all the users that have logged in with bash as default. *w | grep bash | awk '{print $1}'* will print all users and terminal etc. I will then grep for bash and use awk to print out only the first column.

Display all lines of /etc/group starting with "daemon".

```
*cat /etc/group | grep --regexp='^daemon'*
```

will print the whole file and with grep search for lines starting with daemon.

```
cat /etc/group | grep -v --regexp='^daemon'
```

and the following,

```
cat /etc/group | grep -v --regexp='^daemon'
```

will print all lines except the one starting with daemon.

To find the number of lines of /etc/hosts that contain *localhost*, one could use cat

```
/etc/hosts | grep -nc localhost
```

To find all directories under /usr/share/doc, one can use,

```
ls -Rl /usr/share/doc/ 2>/dev/null | grep --regexp='^d' | awk '{print $9}' > directories_us
```

That is, list all files recursively in the directory. Send errors to /dev/null. Grep for files starting with a 'd' ie. directories. Cut out column 9 and redirect the output to directories_usr_share.txt.

Count the number of occurences of *README* in /usr/share/doc.

```
ls -R /usr/share/doc/ | grep -io -nc --regexp='README'
```

Where -R is to make the command recursive. For the grep part of the string,

- i, case insensitive
- o, display only matches
- nc, count the results (202 matches)

List all files in your directory that has been modified today,

```
ls -lR | grep "$(date +"%b %d")" | grep -v drwxr
-rw-r--r-- 1 backefel backefel   3402 Apr 19 21:24 hw4.md
-rw-r--r-- 1 backefel backefel 112229 Apr 19 21:24 hw4.pdf
-rw-r--r-- 1 backefel backefel      0 Apr 19 21:34 test
-rw-r--r-- 1 backefel backefel      0 Apr 19 21:34 test2
```

```
-rw-r--r-- 1 backefel backefel      0 Apr 19 21:34 test23
```

It will grep Apr 19 in this case and remove all search results with drwxr (directories).

```
cat loremipsum | grep -nE -nc '.|^$'
```

Where, * n - displays line number * E - extended regular expressions * nc - counts number occurences

Display the configuration files in /etc that contain numbers

```
ls -R /etc 2\>/dev/null | grep -oE '(\\w[0-9]+.+|\\w+[0-9]+)\.(conf|config)\>' | grep -v /
```

Where: * 2>/dev/null sends stderror to null * grep -oE displays only the matched error with extended regular expression * Regex string is a bit trickier. | means OR or either.

Exercise 4 - sed

Make a long list with the listings of the directory /usr/share/pixmaps. Now, use sed to filter out the xpm and png files.

The following command will find 36 hits,

```
sed -nE '/(xpm|png)/p'  pixmaps_listings

sed -nE '/(xpm|png)/p'  pixmaps_listings | grep -nc -E '.'
```

The *n*-flag will remove all commands that don't match.

To invert the search, just add an exclamation mark,

```
sed -nE '/(.xpm|.png)/!p'  pixmaps_listings
```

Create another file with the files in usr/bin that have an *v* as a second char. Then do the same with *r*.

```
ls /usr/bin | sed -nE '/^\wv/p'
ls /usr/bin | sed -nE '/^\wr/p'
```

Create a file with "The quick brown fox jumps over the lazy dog". replace *lazy* by *brown* and then make the fox yellow.

```
cat the_quick | sed -E 's/lazy/brown/'
cat the_quick | sed -E 's/brown/yellow/'
```

If you only want to replace the first occurence of a word, one can use,

```
cat the\_quick | sed -E '0,/quick/s/quick/slow/'
```

This will turn,

```
echo "The quick brown fox jumps over the quick dog" > the\_quick
```

into the following,

```
The slow brown fox jumps over the quick dog
```

Replace *sagittis* in the loremipsum file with *saggittis*.

```
cat loremipsum | sed -E 's/sagittis/saggittis/' | grep saggittis

cat loremipsum | sed -E 's/sagittis/saggittis/g' | grep saggittis
```

In the loremipsum file, replace all capital letters with lower case.

```
cat loremipsum | sed -E 's/./\L&/g'
```

The global *g* parameter is necessary here but why is that? EDIT: Added the global to the *saggittis*-assignment and I got more hits. Should be used at all time.

Ok! Now replace all instances of "ut" to "Ut" in the beginning of each sentence.

This can be done by adding another a new instance of sed.

```
cat loremipsum | sed -E 's/./\L&/g' | sed -E 's/\. ut/\. Ut/g' | grep -E 'Ut'
```

Exercise 4 - awk

Assume that you have text with the following form,

```
Username:Firstname:Lastname:Telephone number
```

Make an awk script that converts such a line to an LDAP record format:

```
dn: uid=Username, dc=example, dc=com
cn: Firstname Lastname
sn: Lastname
telephoneNumber: Telephone number
```

To do this, we'll use a seperate file with an awk-script. It's basically an fprintf script for those who are familiar with matlab.

To start off lets create a new file,

```
echo "laback:Lars:Efternamn:666" > pre_ldap
echo "haback:Hans:Efternamn:776" >> pre_ldap

vim awk_script
```

Add the following lines to the file awk_script:

```
BEGIN  {
    FS=":"
}
{
    print "\ndn: uid=" $1 ", dc=example, dx=com\n" \
    "cn: " $2 " " $3 "\n" \
    "sn: " $3 "\n"\
    "telephoneNumber: " $4 "\n"
}
END {}
```

Where FS is the field separator, in this case colon.

To run the script we use:

```
awk -f awk\_script ./pre\_ldap
```

```
dn: uid=laback, dc=example, dx=com
cn: Lars Efternamn
sn: Efternamn
telephoneNumber: 666


dn: uid=haback, dc=example, dx=com
cn: Hans Efternamn
Efternamn: Hmm
telephoneNumber: 776
```

Note that you can turn on syntax highlightning in vim with ':set syntax=awk'

For the next assignment, print processes with their corresponding RSS if it's higher than 1000, ie Resident Set Size.

```
ps -eo pid,rss > pid\_rss
```

And for the awk script,

```
BEGIN{
FS=" "
}
{
if($2>10000)
    print $1 " " $2
}
END{}
```

Ok! Now print the total number of bytes in your home directory. First of I think it's easier to redirect the output from ls into a file. Then I'll use awk.

```
ls -Rl ~/> home\_dir
cat home\_dir | awk '{sum+= $ 5} END {print sum}'
```

Make a list of files in /usr/bin that have the letter 'r' as the second character. *Note! the assignment said nothing of recursive listing but the result should be the same, just more hits.

```
ls -lR /usr/bin | awk '{print $9}' | awk '/^\wr/{print $0}'
```

An if statement should have worked as well.

EDIT: A much cleaner solution would be,

```
ls -lR /usr/bin | awk '$9 ~ /^\wr/ {print $9}'
```