The pipes and redirections concept

The concepts is simple, really. The pipes and redirections are used to send (or retrieve) some information sent from one command or script to another command or script. It works on files too. Let's think about some examples:

* Count number of lines in the file
* Select uniq values from one file and write it in another
* Find occurences of some string in the file or system
* and many more

Before we start, here are some commands which we will use when learning pipes. Obviously, all these commands can be used independently.

grep. This command search for given pattern in the output. Output may be the file or output from other command.

grep 'case' .bashrc This command will search for pattern case in a file .bashrc.

wc is a utility for counting words, newlines, bytes. Commonly we use it for counting lines.

When we execute

wc -l .bashrc

it will count how many lines (-l argument) are in our .bashrc file.

sort will sort the output in alphabetical order

sort numbers.txt will sort the prepared file with generated numbers.

By using uniq we can limit the occurences of the same record to only one. But in order to have this done properly, sort is needed, therefore, we need to pipe these one to another.

Ok, we learned some new commands, lets use the in pipe.

The pipe |

Our first structure here is pipe, which uses |.

The structure looks lie this:

command1 | command2

command1 | command2 | command3

Yes, we can combine as many pipes as we wish.

How it works? The output of command1 is taken over as input to command2. And this process can continue to the next command.

Ok. Lets print the file first.

cat numbers.txt

Ok, the file is quite huge. How many records we have? Let's pipe the output from the command we just used to wc!

cat numbers.txt | wc -l

We have the number of lines in the file! Perfect. Now, let's collect the information how many unique records we have. I believe, you already noticed that this file contains numbers from 1 to 100. And as we have 10000 records... some must occured many times.

cat numbers.txt | uniq |wc -l

Hm... something is not right. Let's think...

Did we piped our data correctly? At this moment, we can say - yes. We printed the file, select unique values and count them. But... Did we, really?

Did we missed something? Yes, we did. In this case we have here, we used uniq on unsorted data. That is why we have close to 9000 values, when we should have max 100 (as the script which generated the file used random values between 1 and 100). It is important to remember, uniq always works best with sort. And sort is first.

Let's try.

cat numbers.txt | sort | uniq | wc -l

Yeah! That works.

Redirect to file

Now we will look on redirecting output to file.

Normally, all output is going to the screen. Here we have streams concept involved, but this will be explained in another lab. For now, let's assume that all output is going to the screen.

On the screen we have listed the files. Lets do it again:

ls -al

>

Now, let's redirect this to the file:

ls -al > directorylist.txt

What we did here?

* ls -al - we know this already
* > redirects all output from the left side of the sign, to the file on the right side of the sign
* directorylist.txt file where the redirected output will be sent.

Ok, let's see what is in the file.

cat directorylist.txt

File contains exact output of ls -al command.

Ok, let's write more stuff to the file. We have small file, .profile. Lets cat it and add the output to the directorylist.txt file.

cat .profile > directorylist.txt

Now it is time to see the file!

cat directorylist.txt

Hm... wait... We should *add* content, not *rewrite* it! What happened?

Well, the > redirection will do as follows:

* If file doesn't exist, create it
* Add content from redirected output
* If file exists and it is not empty, clear the file and write the redirected output in empty file

Now you see what happened? Ok, so how to *append* and not *rewrite* the file? Is it possible?

Yes, it is.

>>

We need to use another syntax for redirection: >>.

Let's try this:

rm directorylist.txt - we removed the file, to have clear start.

cat .profile > directorylist.txt - we did the first redirection, and now we will append something from command line and the file with directory listing:

echo 'this line is a separator!' >> directorylist.txt

ls -al >> directorylist.txt

Let's check

cat directorylist.txt

Summary of redirections

You need to remember, that > will create file (if not exists) and **rewrite** all data as fresh file. >> behaves similarly, however, if there is any content in the fill, this redirection will **add** theoutput of the command on the end of existing content.

# redirect from the file

We did redirection to the file. We can do it in oposite direction too.

We have our numbers.txt file. When we print it, we see:

cat numbers.txt

Ok, let's use this file as input for our commands.

Do you remember how we couted number of lines in the file?

cat numbers.txt | wc -l this is... not perfect way, believe me. Familiarize yourself with this **very old** [webpage](https://porkmail.org/era/unix/award.html). Yeah, it is written more than 20 years ago in HTML 3.2! **How cool is that!** (and... it is 3 years younger than my first HTML page ;) )

So, we can just do

wc -l numbers.txt

Ok, works. we have number of lines, but... We have a filename also. Yeah, we can do different things to remove it. Example? Yeah, why not!

wc -l numbers.txt | awk '{print $1}'

or...

awk 'END{print NR}' numbers.txt

(Now you can go outside and impress everyone around what you know about Bash and awk! ;) )

But we want to use redirection to count the lines.

wc -l < numbers.txt

Ok. Let's write this number to another file

wc -l < numbers.txt > count.txt

cat count.txt - we have it!

We can do crazy (and sometimes useless) things

sort < numbers.txt | uniq for example.

The input redirection, <, is used mostly to redirect content to file. To do some operations. Lets see the very simple example of one-liner:

while read line; do echo "Content of the line: ${line}"; done < numbers.txt

What we just did? We created a loop (we will talk about that later) and print every line from the file, as long as there is nothing more in the file numbers.txt.

Content of the line: 70

Content of the line: 29

Content of the line: 71

Content of the line: 42

Content of the line: 3

Content of the line: 60

Content of the line: 62

Content of the line: 80

Content of the line: 27

Content of the line: 11

Content of the line: 64

Content of the line: 61

Content of the line: 5