# Practice M3: Editors. Files. Streams

For the purpose of the current practice we will need one virtual machine. It could be with either **CentOS 8.x**, **openSUSE Leap 15.2**, or **Ubuntu 20.04**. Of course, another version of the listed or another distribution can be used, but there can be some differences.

All commands, that we will use until the end of this practice, will be accompanied by the appropriate prompt. This way it will be easier for us to know which user, in which folder, and on which machine is executing the command.

Next steps will be executed on **CentOS** machine. If there is a big difference in the way we issue the command, or in its result between distributions, it will be stated clearly.

## Part 1

Start the virtual machine.

On the login prompt enter the user and password you specified during the installation process. Most of the time we will be working with the non-privileged user, as working with **root** all the time is not considered a good practice.

Let’s create a local copy of the **/etc/passwd** file:

[lsauser@centos ~]$ **cp /etc/passwd users.txt**

[lsauser@centos ~]$

Usually the standard input is linked to the keyboard. In some circumstances we can use a file as a source of input information. Very often we just supply the file name as argument to the command in question:

[lsauser@centos ~]$ **cat users.txt**

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

bin:x:1:1:bin:/bin:/sbin/nologin

…

lsauser:x:1000:1000:LSA User:/home/lsauser:/bin/bash

newuser:x:1001:1001:New User:/home/newuser:/bin/bash

[lsauser@centos ~]$

We can also use the stdin redirection construction:

[lsauser@centos ~]$ **cat < users.txt**

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

bin:x:1:1:bin:/bin:/sbin/nologin

…

lsauser:x:1000:1000:LSA User:/home/lsauser:/bin/bash

newuser:x:1001:1001:New User:/home/newuser:/bin/bash

[lsauser@centos ~]$

Most often, we want to redirect the standard out, for example to a file:

[lsauser@centos ~]$ **ls -l > dir-list.txt**

[lsauser@centos ~]$

As a result of the above we will end up with **dir-list.txt** file containing list of files in the current directory:

[lsauser@centos ~]$ **cat dir-list.txt**

общо 4

-rw-rw-r--. 1 lsauser lsauser 0 5 дек 10,01 dir-list.txt

-rw-r--r--. 1 lsauser lsauser 904 5 дек 8,25 users.txt

[lsauser@centos ~]$

We can see that our target file is also part of the list, but with zero bytes in size. The explanation is simple – before the command is executed, the redirection is processed.

If we execute the above command several times, we will notice that each time the file is being recreated and the existing content is lost. This is not always what we want. We can easily modify the command to:

[lsauser@centos ~]$ **ls -l >> dir-list.txt**

[lsauser@centos ~]$ **cat dir-list.txt**

общо 4

-rw-rw-r--. 1 lsauser lsauser 0 5 дек 10,01 dir-list.txt

-rw-r--r--. 1 lsauser lsauser 904 5 дек 8,25 users.txt

общо 8

-rw-rw-r--. 1 lsauser lsauser 134 5 дек 10,01 dir-list.txt

-rw-r--r--. 1 lsauser lsauser 904 5 дек 8,25 users.txt

[lsauser@centos ~]$

There is another way of dealing with the constant target file recreation. For this, we must turn on an option in the shell:

[lsauser@centos ~]$ **set -o noclobber**

[lsauser@centos ~]$ **ls -l > dir-list.txt**

-bash: dir-list.txt: не може да се презапише съществуващ файл

[lsauser@centos ~]$

We can restore the default mode with:

[lsauser@centos ~]$ **set +o noclobber**

We can redirect the error stream as well:

[lsauser@centos ~]$ **ls non-existing-file.txt**

ls: няма достъп до non-existing-file.txt: Няма такъв файл или директория

[lsauser@centos ~]$ **ls non-existing-file.txt 2> ls-error.txt**

[lsauser@centos ~]$ **cat ls-error.txt**

ls: няма достъп до non-existing-file.txt: Няма такъв файл или директория

[lsauser@centos ~]$

Of course, we can redirect both the **stdout** and **stderr** to different files:

[lsauser@centos ~]$ **ls -l ~ /missing > ls-stdout.txt 2> ls-stderr.txt**

[lsauser@centos ~]$ **cat ls-stdout.txt**

/home/lsauser:

общо 16

-rw-rw-r--. 1 lsauser lsauser 268 5 дек 10,06 dir-list.txt

-rw-rw-r--. 1 lsauser lsauser 111 5 дек 10,14 ls-error.txt

-rw-rw-r--. 1 lsauser lsauser 98 5 дек 10,17 ls-stderr.txt

-rw-rw-r--. 1 lsauser lsauser 0 5 дек 10,17 ls-stdout.txt

-rw-r--r--. 1 lsauser lsauser 904 5 дек 8,25 users.txt

[lsauser@centos ~]$ **cat ls-stderr.txt**

ls: няма достъп до /missing: Няма такъв файл или директория

[lsauser@centos ~]$

The same way, we can redirect them to one and the same file:

[lsauser@centos ~]$ **ls -l ~ /absent > ls-mixed.txt 2>&1**

[lsauser@centos ~]$ **cat ls-mixed.txt**

ls: няма достъп до /absent: Няма такъв файл или директория

/home/lsauser:

общо 24

-rw-rw-r--. 1 lsauser lsauser 268 5 дек 10,06 dir-list.txt

-rw-rw-r--. 1 lsauser lsauser 111 5 дек 10,14 ls-error.txt

-rw-rw-r--. 1 lsauser lsauser 97 5 дек 10,21 ls-mixed.txt

-rw-rw-r--. 1 lsauser lsauser 98 5 дек 10,17 ls-stderr.txt

-rw-rw-r--. 1 lsauser lsauser 341 5 дек 10,17 ls-stdout.txt

-rw-r--r--. 1 lsauser lsauser 904 5 дек 8,25 users.txt

[lsauser@centos ~]$

We must be aware that the redirection is processed from left to right by the order they are specified. Because of this the result from this command:

[lsauser@centos ~]$ **ls -l /absent > ls-out.txt 2>&1**

[lsauser@centos ~]$ **cat ls-out.txt**

ls: няма достъп до /absent: Няма такъв файл или директория

[lsauser@centos ~]$

Differs from this:

[lsauser@centos ~]$ **ls -l /absent 2>&1 > ls-out.txt**

ls: няма достъп до /absent: Няма такъв файл или директория

[lsauser@centos ~]$ **cat ls-out.txt**

[lsauser@centos ~]$

New files or documents can be created by utilizing the redirection technique and the echo command:

[lsauser@centos ~]$ **echo 'Hello World!' > hello.txt**

[lsauser@centos ~]$ **cat hello.txt**

Hello World!

[lsauser@centos ~]$

Let’s imagine for a moment that we want to store the current date and time in a file. We could do it with:

[lsauser@centos ~]$ **echo "Current date is" date > date.txt**

[lsauser@centos ~]$ **cat date.txt**

Current date is date

[lsauser@centos ~]$

The result is disappointing. It does not contain what we expected. We must rework our command to:

[lsauser@centos ~]$ **echo "Current date is" $(date) > date.txt**

[lsauser@centos ~]$ **cat date.txt**

Current date is вт дек 5 10:34:39 EET 2017

[lsauser@centos ~]$

Alternatively, we can use:

[lsauser@centos ~]$ **echo "Current date is" `date` > date.txt**

[lsauser@centos ~]$ **cat date.txt**

Current date is вт дек 5 10:35:06 EET 2017

[lsauser@centos ~]$

The substitution can be nested in a text. We must be careful, because there is a difference between double quotes (") and single quotes ('). For example, this will work as expected:

[lsauser@centos ~]$ **echo "Current date is `date`" > date.txt**

[lsauser@centos ~]$ **cat date.txt**

Current date is вт дек 5 10:37:14 EET 2017

[lsauser@centos ~]$

But this one won’t:

[lsauser@centos ~]$ **echo 'Current date is `date`' > date.txt**

[lsauser@centos ~]$ **cat date.txt**

Current date is `date`

[lsauser@centos ~]$

There will be situations in which we would need to create documents on the fly. For this we can use the **heredoc** technique:

[lsauser@centos ~]$ **cat << EOF**

> Here comes the first line

> ... and then two empty lines

>

>

> ... this is our final line.

> EOF

Here comes the first line

... and then two empty lines

... this is our final line.

[lsauser@centos ~]$

During this exercise we experienced the multi-line prompt defined by the **PS2** variable. Let’s ask for its definition:

[lsauser@centos ~]$ **echo $PS2**

>

[lsauser@centos ~]$

Of course, we can save **heredoc** to a file:

[lsauser@centos ~]$ **cat > simple-file.txt << EOF**

> Just one line.

> EOF

[lsauser@centos ~]$ **cat simple-file.txt**

Just one line.

[lsauser@centos ~]$

Let’s create one more file with the help of **echo**, but with some tabs. Write this:

[lsauser@centos ~]$ **echo -e 'File with one\tand\t\ttwo tabs.' > file-w-tabs.txt**

[lsauser@centos ~]$

Now, let’s use extended and then the plain **cat** syntax to examine the file:

[lsauser@centos ~]$ **cat -vET file-w-tabs.txt**

File with one^Iand^I^Itwo tabs.$

[lsauser@centos ~]$ **cat file-w-tabs.txt**

File with one and two tabs.

[lsauser@centos ~]$

We can compare the two outputs and see that should we need to check the control characters, there is a way.

As we already know, we can execute series of commands in one iteration. One possible way to accomplish this task is to separate them with semi-colon. This way every command will be executed, no matter how the preceding one ended – successfully or with an error:

[lsauser@centos ~]$ **ls > ls-seq.txt ; cat missing-file.txt ; cat ls-seq.txt**

cat: missing-file.txt: Няма такъв файл или директория

date.txt

dir-list.txt

hello.txt

ls-error.txt

ls-mixed.txt

ls-out.txt

ls-seq.txt

ls-stderr.txt

ls-stdout.txt

users.txt

[lsauser@centos ~]$

Of course, we can rework the command sequence in such a way that every other command will get executed only if the previous completed successfully:

[lsauser@centos ~]$ **ls ls\* && ls users.txt && ls missing.txt && echo "Not executed"**

ls-error.txt ls-mixed.txt ls-out.txt ls-seq.txt ls-stderr.txt ls-stdout.txt

users.txt

ls: няма достъп до missing.txt: Няма такъв файл или директория

[lsauser@centos ~]$

Alternatively, we can change the behavior so every other command to be executed only if the previous resulted in error:

[lsauser@centos ~]$ **ls ls\* || ls users.txt || ls missing.txt || echo "Not executed"**

ls-error.txt ls-mixed.txt ls-out.txt ls-seq.txt ls-stderr.txt ls-stdout.txt

[lsauser@centos ~]$

In fact, only the first command got executed. Let’s shorten the sequence:

[lsauser@centos ~]$ **ls missing.txt || echo "Executed"**

ls: няма достъп до missing.txt: Няма такъв файл или директория

Executed

[lsauser@centos ~]$

Viable option is to mix the execution modes – in success and on failure:

[lsauser@centos ~]$ **ls ls\* && ls missing.txt || echo "Executed"**

ls-error.txt ls-mixed.txt ls-out.txt ls-seq.txt ls-stderr.txt ls-stdout.txt

ls: няма достъп до missing.txt: Няма такъв файл или директория

Executed

[lsauser@centos ~]$

We can pipe the output of one command to another. We can test it with the **tee** command:

[lsauser@centos ~]$ **ls -l | tee tee-out.txt**

общо 36

-rw-rw-r--. 1 lsauser lsauser 23 5 дек 10,37 date.txt

...

-rw-r--r--. 1 lsauser lsauser 904 5 дек 8,25 users.txt

[lsauser@centos ~]$

Now, let’s do the following to create a small file:

[lsauser@centos ~]$ **echo dir1 > new-dirs.txt**

[lsauser@centos ~]$ **echo dir2 >> new-dirs.txt**

[lsauser@centos ~]$ **echo dir3 >> new-dirs.txt**

[lsauser@centos ~]$ **cat new-dirs.txt**

dir1

dir2

dir3

[lsauser@centos ~]$

If we combine the learned so far with the **xargs** command, we can achieve the following:

[lsauser@centos ~]$ **cat new-dirs.txt | xargs mkdir**

[lsauser@centos ~]$ **ls -ld dir\*/**

drwxrwxr-x. 2 lsauser lsauser 6 5 дек 11,12 dir1/

drwxrwxr-x. 2 lsauser lsauser 6 5 дек 11,12 dir2/

drwxrwxr-x. 2 lsauser lsauser 6 5 дек 11,12 dir3/

[lsauser@centos ~]$

## Part 2

One quite common operation is to search for a text in either stream of data or set of one or more files. This can be achieved with the help of the **grep** command. Let’s examine how many users in our system are using bash:

[lsauser@centos ~]$ **grep bash users.txt**

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

lsauser:x:1000:1000:LSA User:/home/lsauser:/bin/bash

newuser:x:1001:1001:New User:/home/newuser:/bin/bash

[lsauser@centos ~]$

Same result can be achieved with the help of the pipe mechanism:

[lsauser@centos ~]$ **cat users.txt | grep bash**

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

lsauser:x:1000:1000:LSA User:/home/lsauser:/bin/bash

newuser:x:1001:1001:New User:/home/newuser:/bin/bash

[lsauser@centos ~]$

Both **cat** and **grep** can process multiple files at once. Let’s experiment with this:

[lsauser@centos ~]$ **grep bash users.txt /etc/passwd**

users.txt:root:x:0:0:root:/root:/bin/bash

users.txt:lsauser:x:1000:1000:LSA User:/home/lsauser:/bin/bash

users.txt:newuser:x:1001:1001:New User:/home/newuser:/bin/bash

/etc/passwd:root:x:0:0:root:/root:/bin/bash

/etc/passwd:lsauser:x:1000:1000:LSA User:/home/lsauser:/bin/bash

/etc/passwd:newuser:x:1001:1001:New User:/home/newuser:/bin/bash

[lsauser@centos ~]$

We can use **grep** to extract all users whose name starts with **p**:

[lsauser@centos ~]$ **grep ^p users.txt**

polkitd:x:999:997:User for polkitd:/:/sbin/nologin

postfix:x:89:89::/var/spool/postfix:/sbin/nologin

[lsauser@centos ~]$

Let’s extend our command to return all users whose name starts with either **p** or **s**:

[lsauser@centos ~]$ **grep ^[ps] users.txt**

sync:x:5:0:sync:/sbin:/bin/sync

shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown

systemd-network:x:192:192:systemd Network Management:/:/sbin/nologin

polkitd:x:999:997:User for polkitd:/:/sbin/nologin

postfix:x:89:89::/var/spool/postfix:/sbin/nologin

sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin

[lsauser@centos ~]$

In a similar fashion we can ask for those rows that end with **sh**:

[lsauser@centos ~]$ **grep sh$ users.txt**

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

lsauser:x:1000:1000:LSA User:/home/lsauser:/bin/bash

newuser:x:1001:1001:New User:/home/newuser:/bin/bash

[lsauser@centos ~]$

Let’s continue our experiments in globing direction. We can ask for all files which name is four characters long:

[lsauser@centos ~]$ **ls -al /etc/????.conf**

-rw-r--r--. 1 root root 9 7 юни 2013 /etc/host.conf

-rw-r--r--. 1 root root 590 28 апр 2017 /etc/krb5.conf

-rw-r-----. 1 root root 1786 7 юни 14,33 /etc/sudo.conf

[lsauser@centos ~]$

We can extend this command to see every matching file’s contents:

[lsauser@centos ~]$ **ls /etc/????.conf | xargs cat**

multi on

# Configuration snippets may be placed in this directory as well

includedir /etc/krb5.conf.d/

…

Of course, we can take this expression and use it directly with command like **cat**:

[lsauser@centos ~]$ **cat /etc/????.conf**

multi on

# Configuration snippets may be placed in this directory as well

includedir /etc/krb5.conf.d/

…

We could extend our expression to match all files which name starts with particular letter and has length of six characters:

[lsauser@centos ~]$ **ls -al /etc/[abc]?????.conf**

-rw-r--r--. 1 root root 55 1 мар 2017 /etc/asound.conf

[lsauser@centos ~]$

The condition can be easily reversed – all files with name not starting with the provided letters:

[lsauser@centos ~]$ **ls -al /etc/[^abc]?????.conf**

-rw-r--r--. 1 root root 1285 5 авг 18,02 /etc/dracut.conf

-rw-r--r--. 1 root root 112 16 мар 2017 /etc/e2fsck.conf

-rw-r--r--. 1 root root 19 20 ное 19,49 /etc/locale.conf

-rw-r--r--. 1 root root 5171 10 юни 2014 /etc/man\_db.conf

-rw-r--r--. 1 root root 936 3 авг 6,24 /etc/mke2fs.conf

-rw-r--r--. 1 root root 66 5 дек 9,47 /etc/resolv.conf

-rw-r--r--. 1 root root 449 4 авг 2,57 /etc/sysctl.conf

[lsauser@centos ~]$

Search for files typically is far more complicated. Usually we must consider other attributes beside the name. For this task we can use the **find** command. For example, let’s start with all files named **README** in the **/usr** folder:

[lsauser@centos ~]$ **sudo find /usr -type f -name README**

/usr/lib/firewalld/ipsets/README

/usr/lib/dracut/modules.d/10i18n/README

/usr/lib/dracut/modules.d/98ecryptfs/README

/usr/lib/dracut/modules.d/98syslog/README

/usr/lib/firmware/README

...

We can ask for all files with **conf** extension in the **/etc** folder:

[lsauser@centos ~]$ **sudo find /etc -type f -name "\*.conf"**

/etc/resolv.conf

/etc/pki/ca-trust/ca-legacy.conf

...

Let’s extend the above example by asking for all files with **txt** extension in our system owned by **lsauser**:

[lsauser@centos ~]$ **sudo find / -type f -name "\*.txt" -user lsauser**

/home/lsauser/users.txt

/home/lsauser/dir-list.txt

/home/lsauser/ls-error.txt

/home/lsauser/ls-stdout.txt

/home/lsauser/ls-stderr.txt

...

[lsauser@centos ~]$

We can reverse the result with:

[lsauser@centos ~]$ **sudo find / -type f -name "\*.txt" -not -user lsauser**

Or all configuration files that changed during the last 24 hours:

[lsauser@centos ~]$ **sudo find /etc -type f -mtime 0 -name "\*.conf"**

/etc/resolv.conf

[lsauser@centos ~]$

We can apply actions on the found files as well:

[lsauser@centos ~]$ **sudo find /etc -type f -mtime 0 -name "\*.conf" -exec cat {} \;**

# Generated by NetworkManager

search sulab

nameserver 192.168.1.5

[lsauser@centos ~]$

Now, let’s continue our journey towards examine what is inside the files.

There is a command like **cat**, but showing the data in reverse - **tac**:

[lsauser@centos ~]$ **cat new-dirs.txt**

dir1

dir2

dir3

[lsauser@centos ~]$ **tac new-dirs.txt**

dir3

dir2

dir1

[lsauser@centos ~]$

Should we work with file that does not fit on the screen, there is a tool. In fact, there are two tools – **less** and **more**:

[lsauser@centos ~]$ **less users.txt**

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

bin:x:1:1:bin:/bin:/sbin/nologin

daemon:x:2:2:daemon:/sbin:/sbin/nologin

adm:x:3:4:adm:/var/adm:/sbin/nologin

lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin

sync:x:5:0:sync:/sbin:/bin/sync

shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown

halt:x:7:0:halt:/sbin:/sbin/halt

mail:x:8:12:mail:/var/spool/mail:/sbin/nologin

operator:x:11:0:operator:/root:/sbin/nologin

games:x:12:100:games:/usr/games:/sbin/nologin

ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin

nobody:x:99:99:Nobody:/:/sbin/nologin

systemd-network:x:192:192:systemd Network Management:/:/sbin/nologin

dbus:x:81:81:System message bus:/:/sbin/nologin

polkitd:x:999:997:User for polkitd:/:/sbin/nologin

postfix:x:89:89::/var/spool/postfix:/sbin/nologin

sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin

lsauser:x:1000:1000:LSA User:/home/lsauser:/bin/bash

newuser:x:1001:1001:New User:/home/newuser:/bin/bash

users.txt (END)

We can initiate a search with the **/** key. For example, we can search for **bash**. To go on the next match, press the **n** key. Quit less with the **q** key.

In order to compare and understand the differences between **more** and **less**, we can experiment with a bigger file, for example **/etc/services** file. First open it with **less**, and then with **more**.

Relatively often we will need only the first or the last few lines of a file. For the first, we can use **head**:

[lsauser@centos ~]$ **head -n 3 users.txt**

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

bin:x:1:1:bin:/bin:/sbin/nologin

daemon:x:2:2:daemon:/sbin:/sbin/nologin

[lsauser@centos ~]$

And for the last – **tail**:

[lsauser@centos ~]$ **tail -n 3 users.txt**

sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin

lsauser:x:1000:1000:LSA User:/home/lsauser:/bin/bash

newuser:x:1001:1001:New User:/home/newuser:/bin/bash

[lsauser@centos ~]$

We can use both **head** and **tail** to show all lines but the last or first N. For example, let’s see all lines but the last 2:

[lsauser@centos ~]$ **head -n -2 /etc/passwd**

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

bin:x:1:1:bin:/bin:/sbin/nologin

...

[lsauser@centos ~]$

There are other interesting tools for text manipulation as well. One such tool is **sort**:

[lsauser@centos ~]$ **sort new-dirs.txt**

dir1

dir2

dir3

[lsauser@centos ~]$

Or in reverse order:

[lsauser@centos ~]$ **sort -r new-dirs.txt**

dir3

dir2

dir1

[lsauser@centos ~]$

With the **nl** tool we can prefix every row with a number:

[lsauser@centos ~]$ **nl new-dirs.txt**

1 dir1

2 dir2

3 dir3

[lsauser@centos ~]$

We can count how many users there are in our system by counting the lines in the **/etc/passwd** file with **wc**:

[lsauser@centos ~]$ **wc -l /etc/passwd**

20 /etc/passwd

[lsauser@centos ~]$

Should we need information about the shells used in our system, we can use the **cut** command:

[lsauser@centos ~]$ **cut -d : -f 7 /etc/passwd**

/bin/bash

/sbin/nologin

/sbin/nologin

/sbin/nologin

/sbin/nologin

/bin/sync

/sbin/shutdown

/sbin/halt

/sbin/nologin

/sbin/nologin

/sbin/nologin

/sbin/nologin

/sbin/nologin

/sbin/nologin

/sbin/nologin

/sbin/nologin

/sbin/nologin

/sbin/nologin

/bin/bash

/bin/bash

[lsauser@centos ~]$

This list is not easy to read. We must do some post processing. Let’s store it in file and then use the **uniq** command:

[lsauser@centos ~]$ **cut -d : -f 7 /etc/passwd > shell-list.txt**

[lsauser@centos ~]$ **uniq shell-list.txt**

/bin/bash

/sbin/nologin

/bin/sync

/sbin/shutdown

/sbin/halt

/sbin/nologin

/bin/bash

[lsauser@centos ~]$

Okay, now the list is shorter, but still not what we are looking for. And the reason for this is that it is not sorted, so the uniqueness is not working correctly.

Let’s get rid of the intermediate file and assemble the following command:

[lsauser@centos ~]$ **cut -d : -f 7 /etc/passwd | sort | uniq**

/bin/bash

/bin/sync

/sbin/halt

/sbin/nologin

/sbin/shutdown

[lsauser@centos ~]$

Same result can be achieved with shorter command statement: **cut -d : -f 7 /etc/passwd | sort -u**

## Part 3

Let’s first make a copy of the **/etc/services** file locally and open it with **vi**:

[lsauser@centos ~]$ **cp /etc/services .**

[lsauser@centos ~]$ **vi services**

Now, we can test the navigation and the searching.

Press **G** to go to the end. Now type in **gg** to go to the beginning. Enter **150G** to go to row **150**. Not sure if this indeed is row **150**? Press **:**, type **set number** and press **Enter**.

Let’s search for a string, for example – **https**. While in command mode type **/https** and press **Enter**. Go to the next match by pressing the **n** key.

Close the editor by pressing the **:**, typing **q** and hitting **Enter**.

Now, open the same file, but go directly to a particular row:

[lsauser@centos ~]$ **vi +4801 services**

Or to a row, containing a string:

[lsauser@centos ~]$ **vi +/VNC services**

Let’s change all occurrences of **tcp** to **TCP**. This can be done by typing **:%s/tcp/TCP/** and pressing **Enter**. Now if we repeat it once more, there should not be any modification. Let’s do it. It appears that there are some. Once more? Again, few modifications. The reason for this is that there are rows with multiple occurrences of the word **tcp**. Now press three times the **u** key to undo the changes. Correct the search and replace command by typing **:%s/tcp/TCP/g** and pressing **Enter**. Now all occurrences of **tcp** are replaced with **TCP**.

We can limit the replace action to a range of rows. Let’s add four spaces in front of lines between **100** and **150**. First go to **100-th** line by typing **100G**. Now type **:100,150s/^/ /** and press **Enter**.

Save the modified file to another file. Type **:w services-modified.txt** and press **Enter**.

Go to line **22** and insert the contents of external file. For example, insert the **/etc/hostname** file. **Type :r /etc/hostname** and press **Enter**.

In a similar fashion we can insert the result from a command’s execution. Type **:r ! uname -a** and press **Enter**.

Now save the first **100** lines in a separate file by typing **:1,100w services-100.txt** and press **Enter**.

It is time to close the file without saving the changes. Type **:q!** and press **Enter**.

Beside the screen-oriented editors, there are also stream editors such as **sed**.

Let’s create a simple stream and pipe it through **sed**:

[lsauser@centos ~]$ **echo "Tram-ta-ra-ra-ram" | sed s/a/A/**

TrAm-ta-ra-ra-ram

[lsauser@centos ~]$

If we want all lower-case a turned into upper case, then we must change the command to:

[lsauser@centos ~]$ **echo "Tram-ta-ra-ra-ram" | sed s/a/A/g**

TrAm-tA-rA-rA-rAm

[lsauser@centos ~]$

We can process files as well:

[lsauser@centos ~]$ **sed s/tcp/TCP/g services**

...

systat 11/udp users

daytime 13/TCP

daytime 13/udp

qotd 17/TCP quote

qotd 17/udp quote

...

[lsauser@centos ~]$

Of course, the result can be piped through another command like head or tail. It can also be stored as file.

We can have more than one processing iteration with:

[lsauser@centos ~]$ **sed s/tcp/TCP/g services | sed s/udp/UDP/g**

Of course, the same could be achieved with either this:

[lsauser@centos ~]$ **sed 's/tcp/TCP/g ; s/udp/UDP/g' services**

Or this:

[lsauser@centos ~]$ **sed -e s/tcp/TCP/g -e s/udp/UDP/g services**

We can return only the changes:

[lsauser@centos ~]$ **sed -n s/dns/DNS/pg services**

DNSix 90/tcp # DNSIX Securit Attribute Token Map

DNSix 90/udp # DNSIX Securit Attribute Token Map

sDNSkmp 558/tcp # SDNSKMP

...

Or change only a sub-set of rows:

[lsauser@centos ~]$ **sed -n '650,680s/dns/DNS/pg' services**

DNSix 90/tcp # DNSIX Securit Attribute Token Map

DNSix 90/udp # DNSIX Securit Attribute Token Map

[lsauser@centos ~]$

We can use **sed** to remove lines matching a pattern as well, for example – remove all comments and empty lines:

[lsauser@centos ~]$ **sed '/^#/d ; /^$/d' services**

And save the result in the same file, while retaining a backup copy:

[lsauser@centos ~]$ **sed -i.bak '/^#/d ; /^$/d' services**

Now, with the help of our favorite editor 😉, we can open both files side-by-side. We can have them visualized in a split window. As the default installed version is not the full one, let’s use the horizontal split:

[lsauser@centos ~]$ **vi -o services.bak services**

We can switch between files with **:n** and **:N** or with (**Ctr+w) + w**

If have a default installation of **openSUSE** and **Ubuntu**, we can test the vertical split as well. In order to do it, we must change the switch to **-O** (capital letter O).

To close both files, assuming no changes made, we can execute **:qa**

***WARNING: The remaining set of tasks could harm your system. Do them with caution.***

Vi is used in few other tools like **vipw** and **vigr**. With those tools we can change in an easier fashion information about the users, groups, and their passwords, covering all four files.

Let’s rename our user’s login from **lsauser** to **mylsauser**:

[lsauser@centos ~]$ **sudo vipw**

You have modified /etc/passwd.

You may need to modify /etc/shadow for consistency.

Please use the command 'vipw -s' to do so.

[lsauser@centos ~]$

Now, if we try to edit the groups file, we will see that our password is not being accepted anymore and the reason is that there is no match between our user and a row in the **/etc/shadow** file. Our only option is to switch to **root** and make the necessary changes there:

[lsauser@centos ~]$ **su -**

Парола:

Последно влизане: вт дек 5 01:02:36 EET 2017 на pts/0

[root@centos ~]# **vigr**

You have modified /etc/group.

You may need to modify /etc/gshadow for consistency.

Please use the command 'vigr -s' to do so.

[root@centos ~]# **vipw -s**

You have modified /etc/shadow.

You may need to modify /etc/passwd for consistency.

Please use the command 'vipw' to do so.

[root@centos ~]# **exit**

Then change the password file and exit the **root** session.

Our user is still with the old name. Now if we ask for the list of files in our home directory, we will see that the owner and the group are with their new name:

[lsauser@centos ~]$ **ls -l ls\***

-rw-rw-r--. 1 mylsauser mylsauser 111 5 дек 10,14 ls-error.txt

-rw-rw-r--. 1 mylsauser mylsauser 501 5 дек 10,21 ls-mixed.txt

-rw-rw-r--. 1 mylsauser mylsauser 0 5 дек 10,24 ls-out.txt

-rw-rw-r--. 1 mylsauser mylsauser 118 5 дек 10,42 ls-seq.txt

-rw-rw-r--. 1 mylsauser mylsauser 98 5 дек 10,17 ls-stderr.txt

-rw-rw-r--. 1 mylsauser mylsauser 341 5 дек 10,17 ls-stdout.txt

[lsauser@centos ~]$

The last step would be to change the information in the **/etc/gshadow** file:

[lsauser@centos ~]$ **sudo vigr -s**

You have modified /etc/gshadow.

You may need to modify /etc/group for consistency.

Please use the command 'vigr' to do so.

[lsauser@centos ~]$

When we edit the shadow files, we must use the **w!** or **wq!** command to force the write.

There is similar tool for modifying the **sudo** allowance – **visudo**. Unlike the **vipw** and **vigr**, this tool does syntax check.

We can add one new user named **demouser** with:

[lsauser@centos ~]$ **sudo useradd -m -c "Demo User" demouser**

If we want to allow it to execute commands with **sudo**, we can go either by adding it to a special group. For **CentOS** and **openSUSE** this group is **wheel**:

[lsauser@centos ~]$ **sudo usermod -aG wheel demouser**

And for **Ubuntu** this group is **admin** or **sudo**, depending on the version:

[lsauser@ubuntu ~]$ **sudo usermod -aG sudo demouser**

Another option is to edit a special configuration file **/etc/sudoers**. This can be done with any text editor, or with a special tool – **visudo**. Let’s do it and add the user just after the **root**. Save the changes and log out. Now log in back, but this time with the **demouser** and try to execute command with **sudo**.

Not all distributions are made the same. You can encounter different behavior. For example, with **openSUSE** this functionality is organized a little bit different – all users can act as administrators, but they should use the **root** password when escalating. We can modify this if we edit the **/etc/sudoers** file and comment the following two lines:

**#**Defaults targetpw # ask for the password of the target user i.e. root

**#**ALL ALL=(ALL) ALL # WARNING! Only use this together with 'Defaults targetpw'!

Then we can add a record for the user in question under the one for **root**:

root ALL=(ALL) ALL

**lsauser ALL=(ALL) ALL**

Now, we can save and then log out. Once log in again, we will see that the **sudo** behavior is the same as with the other two distributions.

Besides the group membership, the other supported and recommended option is to create a file with the appropriate instructions and store it in a special place. For example, if we had a **demouser2**, and we want to give him/her **sudo** permissions, we would create a file that contains:

**demouser2 ALL=(ALL) ALL**

And save it to the **/etc/sudoers.d/demouser2** or another name that is meaningful.