
BASIS

Выпуск 27/12/2021

GNU Linux Pro

мар. 03, 2025

Оглавление

1 Вступление	1
1.1 Содержимое	2
2 Курс	23
2.1 01. Операционные системы и GNU-Linux	23
2.2 02. Виртуализация	31
2.3 03. Установка AlmaLinux	35
2.4 04. О файловых системах	80
2.5 05. Текстовый интерфейс пользователя	84
2.6 06. Пути и директории.md	92
2.7 07. Создание и копирование файлов	100
2.8 08. Перемещение, переименование, удаление. Жёсткие и символьические ссылки	108
2.9 09. Чтение текстовых файлов	113
2.10 10. Текстовые редакторы nano и vi	121
2.11 11. Стандартные потоки	128
2.12 12. bash №1: bash-completion, alias, type	134
2.13 13. bash №2: переменные	141
2.14 14. Процессы №1: Информация о процессах №1	151
2.15 15. Процессы №2: Информация о процессах №2	158
2.16 16. Процессы №3: Работа с процессами	169
2.17 17. su	174
2.18 18. sudo	180
2.19 19. Пользователи	190
2.20 20. Права на файлы	203
2.21 21. Ядро Linux	222
2.22 22. Работа с дисками	230
2.23 23. Основы файловых систем	243
2.24 24. Работа с файловыми системами	249
2.25 25. Управление логическими томами - LVM	260
2.26 26. Программный RAID - MD	277
2.27 27. bash скрипты №1	290
2.28 28. bash скрипты №2	300
2.29 29. bash скрипты №3	312
2.30 30. bash скрипты №4	325
2.31 31. bash скрипты №5	341
2.32 32. bash скрипты №6	350

2.33	33. Загрузчик GRUB	358
2.34	34. Система инициализации - systemd	369
2.35	35. Системный менеджер systemd	377
2.36	36. Логирование	382
2.37	37. Планировщики задач	395
2.38	38. Создание backup скрипта	409
2.39	39. Инкрементальные бэкапы с tar	426
2.40	40. Дедупликация с VDO	436
2.41	41. Создание systemd юнитов	445
2.42	42. Основы сетей	463
2.43	43. Работа с сетью	478
2.44	44. Удалённый доступ - SSH	495
2.45	45. Принудительный контроль доступа - SELinux	508
2.46	46. Межсетевой экран - firewalld	525
2.47	47. Пакетный менеджер - dnf	547
2.48	48. Восстановление доступа	573
2.49	49. Виртуальная память, swap	580
2.50	50. Планировщик процессов	593
2.51	51. Оптимизация производительности - tuned	601
2.52	52. Управление многоуровневым хранилищем - stratis	613
2.53	53. Установка RHEL	622
2.54	54. Настройка времени	661
2.55	55. Работа с IPv6	675
2.56	56. Передача файлов по сети	684
2.57	57. Сетевые файловые системы - NFS	694
2.58	58. Сетевые файловые системы - SMB	707
2.59	59. Автоматическое монтирование - Autofs	728
2.60	60. Веб-интерфейс - Cockpit	741
2.61	61. Глоббинг и регулярные выражения	759
2.62	62. Основы контейнеризации	771
2.63	63. Работа с podman	784
2.64	64. Про сертификацию RHCSA	804

Вступление

Основы GNU/Linux



и подготовка к RHCSA



Всем привет!

Это начало курса по изучению операционных систем семейства Linux и подготовки к экзамену для сертификации системного администратора по RedHat. Однако, основная цель, которую я ставлю – научить вас работать с линуксами. Я постараюсь сделать курс понятным для начинающих и по началу буду подавать информацию максимально простым языком.

Редхатовский экзамен – практический. Он охватывает ряд тем, необходимых администратору, но далеко не все темы. В названия тем, которые необходимо знать на экзамене, я буду добавлять надпись «RHCSA», но весь курс будет охватывать гораздо больше тем.

Так как я буду объяснять с азов, я поставил еще одну цель – сделать этот курс также для тех, кто абсолютно не знаком с ИТ. Для меня знакомство с операционными системами – это первый и очень важный шаг становления работником ИТ сферы. Поэтому, для кого-то ряд тем может показаться довольно простыми, но это позволит любому желающему разбираться в ИТ.

1.1 Содержимое

1.1.1 Ссылки

1.1.2 Команды

Команды

Некоторые команды и ключи разбираются в разных темах.

05. Текстовый интерфейс пользователя

```
ls  
ls -i  
ls --help  
man  
man -k  
info  
history  
!!
```

06. Пути и директории

```
pwd  
cd  
mkdir  
mkdir -p  
rmdir  
rm -r  
ls -R
```

07. Создание и копирование файлов

```
touch  
ls -a  
cp  
cp -v  
cp -i  
cp -n  
cp -u  
cp -l  
cp -r  
cp -a
```

08. Перемещение, переименование, удаление. Жёсткие и символические ссылки

```
mv
mv -v
rm
rm -v
rm -r
rm -f
ln
ln -s
ln -v
ls -l
```

09. Чтение текстовых файлов

```
cat
cat -n
tac
less
head
head -
tail
tail -
tail -n
tail -f
grep
grep -n
grep -r
grep -l
grep -v
```

10. Текстовые редакторы nano и vi

```
nano
vi
vim
```

11. Стандартные потоки

```
>
>>
2>
&>
|
tee
```

12. bash №1: bash-completion, alias, type

```
alias
ls -d
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
type  
type -a
```

13. bash №2: переменные

```
echo  
env  
export  
$()
```

14. Процессы №1: Информация о процессах №1

```
ps  
ps -e  
ps -f  
less -S  
watch
```

15. Процессы №2: Информация о процессах №2

```
top  
w  
ulimit  
ulimit -u  
ulimit -v  
nice  
nice -n  
renice  
renice -n  
htop
```

16. Процессы №3: Работа с процессами

```
kill  
kill -1  
kill -9  
kill -SIGKILL  
pkill  
pkill -19  
pkill -18
```

17. su

```
wc -l  
exit  
su  
su -  
su -c
```

18. sudo

```
sudo  
sudo -s  
sudoedit  
visudo  
visudo -f  
hostname  
groups  
which
```

19. Пользователи

```
id  
newgrp  
chage  
chage -l  
useradd  
useradd -D  
useradd -b  
useradd -d  
useradd -c  
useradd -g  
useradd -G  
useradd -u  
passwd  
usermod  
usermod -m  
usermod -aG  
userdel  
userdel -r  
groupadd  
groupmod  
groupdel  
gpasswd  
gpasswd -A  
gpasswd -M  
gpasswd -a  
gpasswd -d  
lid  
lid -g
```

20. Права на файлы

```
stat  
ls -l  
chown  
chown -R  
chown -v  
chgrp  
chmod  
chmod -v
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
chmod -R  
chmod [+-=] [rwxts]  
umask  
umask -S  
getfacl  
setfacl  
setfacl -m  
setfacl -b
```

21. Ядро Linux

```
uname  
uname -r  
du  
du -h  
modinfo  
lscpu  
lspci  
lsusb  
lshw  
hardinfo  
dmesg  
dmesg -w  
dmesg -H  
modprobe  
modprobe -r  
lsmod
```

22. Работа с дисками

```
lsscsi  
lsscsi -s  
fdisk  
fdisk -l  
cfdisk  
lsblk
```

23. Основы файловых систем

```
iostat
```

24. Работа с файловыми системами

```
mkfs  
mkfs.ext4  
tune2fs  
tune2fs -l  
lsof  
lsof +D
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
mount  
df  
df -h  
blkid  
cut  
reboot  
fsck
```

25. Управление логическими томами - LVM

```
pvccreate  
pvs  
pvsdisplay  
vgcreate  
vgs  
vgdisplay  
lvcreate  
lvs  
lvdisplay  
lsblk -f  
wipefs  
wipefs -a  
vgextend  
lvextend  
lvextend -r  
resize2fs  
lvremove  
vgremove  
pvremove  
lvcreate -s  
lvconvert  
lvconvert --merge
```

26. Программный RAID - MD

```
mdadm  
mdadm -D
```

27. bash скрипты №1

```
read  
read -p
```

28. bash скрипты №2

```
if  
[]  
test
```

29. bash скрипты №3

```
id -u  
exit 1  
&&  
||  
[ -o ]  
[ -z ]  
[ -f ]
```

30. bash скрипты №4

```
for  
select  
case
```

31. bash скрипты №5

```
tr  
grep -w
```

32. bash скрипты №6

```
while  
++  
--  
sleep  
until
```

33. Загрузчик GRUB

```
grep -e  
grub2-mkconfig  
grub2-mkconfig -o  
lsinitrd  
dracut -f
```

34. Система инициализации - systemd

```
systemctl  
systemctl get-default  
systemctl list-dependencies  
systemctl set-default  
systemctl cat  
systemctl isolate  
systemctl enable  
systemctl disable  
systemctl is-enabled
```

35. Системный менеджер systemd

```
systemctl stop
systemctl start
systemctl restart
systemctl reload
systemctl mask
systemctl unmask
systemctl status
systemctl --all
systemctl show
```

36. Логирование

```
journalctl
journalctl -e
journalctl -u
journalctl -f
journalctl -b
logger
logger -p
```

37. Планировщики задач

```
at
atq
at -l
at -c
at -f
at -r
atrm
crontab -e
crontab -l
systemd-run
```

38. Создание backup скрипта

```
du -s
sort
ls -h
ls -S
find
find -type
find -name
find -user
find -perm
find -exec
find -ls
find -ok
find -mtime
find -delete
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
tar
tar -c
tar -f
tar -t
tar -x
tar -C
tar -u
tar -z
tar -v
gzip
gzip -k
bzip2
bzip2 -k
date
touch -t
```

39. Инкрементальные бэкапы с tar

```
tar -g
tar -vv
```

40. Дедупликация с VDO

```
vdo
vdo create
vdo status
vdostats
```

41. Создание systemd юнитов

```
systemctl daemon-reload
systemctl list-units
systemctl enable --now
systemctl disable --now
```

43. Работа с сетью

```
ip
ip address
ip route
ip link
ping
traceroute
nslookup
nmtui
```

44. Удалённый доступ - SSH

```
ssh
ssh -X
ssh-copy-id
```

45. Принудительный контроль доступа - SELinux

```
sestatus
getenforce
setenforce
semanage
semanage login
semanage user
semanage port
semanage fcontext
semanage boolean
semanage export
id -Z
ps -Z
ls -Z
chcon
restorecon
getsebool
setsebool
setsebool -P
```

46. Межсетевой экран - firewalld

```
firewall-cmd
firewall-cmd --permanent
firewall-cmd --reload
firewall-cmd --add-port
firewall-cmd --remove-port
firewall-cmd --list-all
firewall-cmd --info-service
firewall-cmd --list-services
firewall-cmd --list-ports
firewall-cmd --remove-service
firewall-cmd --get-icmptypes
firewall-cmd --add-icmp-block-inversion
firewall-cmd --remove-icmp-block-inversion
firewall-cmd --set-target
firewall-cmd --get-zones
firewall-cmd --get-default-zone
firewall-cmd --change-interface
firewall-cmd --list-interfaces
firewall-cmd --add-source
firewall-cmd --add-masquerade
firewall-cmd --runtime-to-permanent
firewall-cmd --list-all-zones
firewall-cmd --panic-on
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
firewall-cmd --panic-off
firewall-cmd --add-rich-rule
firewall-cmd --remove-rich-rule
ss
ss -n
ss -l
ss -t
ss -a
nc
nc -z
nc -v
nc -u
nc -l
```

47. Пакетный менеджер - dnf

```
dnf
dnf install
dnf download
dnf deplist
dnf repolist
dnf info
dnf makecache
dnf search
dnf remove
dnf check-upgrade
dnf upgrade
dnf needs-restarting
dnf provides
dnf grouplist
dnf groupinfo
dnf groupinstall
dnf module
dnf module list
dnf module info
dnf module install
dnf history
dnf history info
dnf history undo
dnf history redo
dnf help
rpm --scripts
rpm -p
rpm -i
rpm -q
rpm -l
rpm -a
rpm --import
ldd
cpio
rpm2cpio
```

48. Восстановление доступа

```
load_policy  
chroot
```

49. Виртуальная память, swap

```
free -m  
dd  
mkswap  
swapon  
swapoff  
sysctl
```

50. Планировщик процессов

```
chrt  
chrt -p  
chrt -m  
chrt -f  
chrt -r  
chrt -o
```

51. Оптимизация производительности - tuned

```
tuned-adm  
tuned-adm list  
tuned-adm active  
tuned-adm recommend  
tuned-adm profile  
tuned-adm verify  
tuned-adm off
```

52. Управление многоуровневым хранилищем - stratis

```
stratis  
stratis pool  
stratis pool create  
stratis pool list  
stratis pool init-cache  
stratis pool add-data  
stratis blockdev list  
stratis filesystem create  
stratis filesystem list  
stratis filesystem destroy  
stratis filesystem rename
```

54. Настройка времени

```
hwclock  
hwclock -s  
timedatectl  
timedatectl set-local-rtc  
timedatectl list-timezones  
timedatectl set-timezone  
chronyc sources  
firewall-cmd --add-service
```

55. Работа с IPv6

```
firewall-cmd --add-protocol  
ip -6  
traceroute -6
```

56. Передача файлов по сети

```
scp  
scp -r  
scp -C  
rsync  
rsync -a  
rsync -v  
rsync -z  
rsync -P
```

57. Сетевые файловые системы - NFS

```
exportfs  
exportfs -a  
exportfs -v  
exportfs -s  
showmount  
showmount -e  
umount -f  
umount -l
```

58. Сетевые файловые системы - SMB

```
testparm  
smbpasswd  
smbpasswd -a  
smbclient -L
```

59. Автоматическое монтирование - Autofs

```
hostnamectl  
ipa-client-install
```

63. Работа с podman

```
podman  
podman search  
podman pull  
podman images  
podman run  
podman ps  
podman logs  
podman exec  
podman tag  
podman push  
podman image  
skopeo copy  
podman stop  
podman start  
podman login  
podman generate  
systemctl --user  
logindctl enable-linger  
firewall-cmd --add-forward-port
```

64. Про сертификацию RHCSA

```
shutdown now  
poweroff  
systemctl poweroff  
reboot  
star  
tar --selinux
```

1.1.3 RHCSA

RHCSA

Официальный сайт

Understand and use essential tools

Access a shell prompt and issue commands with correct syntax	05
Use input-output redirection (>, >>, , 2>, etc.)	11
Use grep and regular expressions to analyze text	09, 31, 61
Access remote systems using SSH	44
Log in and switch users in multiuser targets	17
Archive, compress, unpack, and uncompress files using tar, star, gzip, and bzip2	38, 64
Create and edit text files	10
Create, delete, copy, and move files and directories	06 - 08
Create hard and soft links	08
List, set, and change standard ugo/rwx permissions	20
Locate, read, and use system documentation including man, info, and files in /usr/share/doc	05

Create simple shell scripts

Conditionally execute code (use of: if, test, [], etc.)	28, 29
Use Looping constructs (for, etc.) to process file, command line input	30, 32
Process script inputs (\$1, \$2, etc.)	27
Processing output of shell commands within a script	29 - 32
Processing shell command exit codes	27 - 32

Operate running systems

Boot, reboot, and shut down a system normally	64
Boot systems into different targets manually	64
Interrupt the boot process in order to gain access to a system	48
Identify CPU/memory intensive processes and kill processes	15, 16
Adjust process scheduling	50
Manage tuning profiles	51
Locate and interpret system log files and journals	36
Preserve system journals	36
Start, stop, and check the status of network services	35
Securely transfer files between systems	56

Configure local storage

List, create, delete partitions on MBR and GPT disks	22
Create and remove physical volumes	25
Assign physical volumes to volume groups	25
Create and delete logical volumes	25
Configure systems to mount file systems at boot by universally unique ID (UUID) or label	24
Add new partitions and logical volumes, and swap to a system non-destructively	22, 25, 49

Create and configure file systems
--

Create, mount, unmount, and use vfat, ext4, and xfs file systems	24
Mount and unmount network file systems using NFS	57, 59
Extend existing logical volumes	25
Create and configure set-GID directories for collaboration	20
Configure disk compression	40
Manage layered storage	52
Diagnose and correct file permission problems	20

Deploy, configure, and maintain systems
--

Schedule tasks using at and cron	37
Start and stop services and configure services to start automatically at boot	34, 35
Configure systems to boot into a specific target automatically	34
Configure time service clients	54
Install and update software packages from Red Hat Network, a remote repository, or from the local file system	47
Work with package module streams	47
Modify the system bootloader	33

Manage basic networking

Configure IPv4 and IPv6 addresses	43, 55
Configure hostname resolution	43
Configure network services to start automatically at boot	34
Restrict network access using firewall-cmd/firewall	46

Manage users and groups

Create, delete, and modify local user accounts	19
Change passwords and adjust password aging for local user accounts	19
Create, delete, and modify local groups and group memberships	19
Configure superuser access	18

Manage security

Configure firewall settings using firewall-cmd/firewalld	46
Create and use file access control lists	20, 58
Configure key-based authentication for SSH	44
Set enforcing and permissive modes for SELinux	45
List and identify SELinux file and process context	45
Restore default file contexts	45
Use boolean settings to modify system SELinux settings	45
Diagnose and address routine SELinux policy violations	45, 61

Manage containers

Find and retrieve container images from a remote registry	63
Inspect container images	63
Perform container management using commands such as podman and skopeo	63
Perform basic container management such as running, starting, stopping, and listing running containers	63
Run a service inside a container	63
Configure a container to start automatically as a systemd service	63
Attach persistent storage to a container	63

1.1.4 Вопросы с интервью

Вопросы с собеседований**Linux**

- Опишите процесс загрузки ПК
- Что такое ООМ. Как ООМ-killer решает - какие процессы убить?
- Что такое inode?
- Load Average - что это такое, как высчитывается?
- Что такое Linux signal? Для чего используются? Какие сигналы можно перехватить? Отличие SIGKILL от SIGTERM?
- Какие бывают состояния у процессов? Что такое зомби процесс? Как возникают зомби процессы?
- В чем разница между системными вызовами exec и fork?
- При попытке отмонтировать каталог получаем ошибку что каталог занят, как найти PID ?
- Опишите сценарий, когда вы можете получить ошибку «filesystem is full», но „df“ показывает наличие свободного места.
- Расскажите о плюсах systemd
- Что такое LVM?
- В чём разница между yum и apt?
- У вас ext4, место на диске есть, но записать на него не выходит, в чём проблема?
- Как восстановить файл, который сейчас открыт приложением, но который удалили?

Network

- Для чего нужны и используются vlan? Сколько vlan может быть?
- Что происходит, когда вы в браузере набираете <https://www.google.com>?
- Что такое TCP keep-alive?
- Расскажите о TCP handshake
- Расскажите о RR в DNS
- Расскажите, чем отличается proxy от nat?
- Расскажите, что описывает модель OSI?

DevOps

- DevOps и Agile. Что это такое и в чем разница?
- Что такое статическая и динамическая линковка файлов?
- В чем разница между виртуализацией и контейнеризацией?
- CI/CD - опишите pipeline для приложения x
- Расскажите о архитектуре K8s?
- Что такое IaC? в чем разница между Chef, Ansible, Terraform?
- Какие плюсы и минусы есть у Ansible?
- В чем отличие роли от playbook?
- Что такое идемпотентность?
- Что такое state-full и stateless?
- (Дан Dockerfile с кучей слоев) Как бы вы уменьшили размер образа?
- Что такое cgroups и namespaces?
- Что такое Jobs, runner, stages?
- Что такое система контроля версий?
- Что такое Terraform provider?
- Какие механизмы позволяют изолировать процессы внутри докер контейнера?
- Представим задачу, вам нужно прочистить все камины в Бруклине, с чего вы начнёте и сколько возьмёте за работу? (Правильного ответа нет, оценивается сам подход к ответу)

Database

- В чем отличие между SQL и No-SQL базами данных?
- Как правильно делать бекапы SQL баз данных?
- Как можно ускорить работу Postgres?
- Что такое нормализация?
- Расскажите что такое primary key и foreign key

1.1.5 Полезные ссылки

Полезные ссылки

Лабораторные от RedHat

<https://lab.redhat.com/>

GitHub Семаева

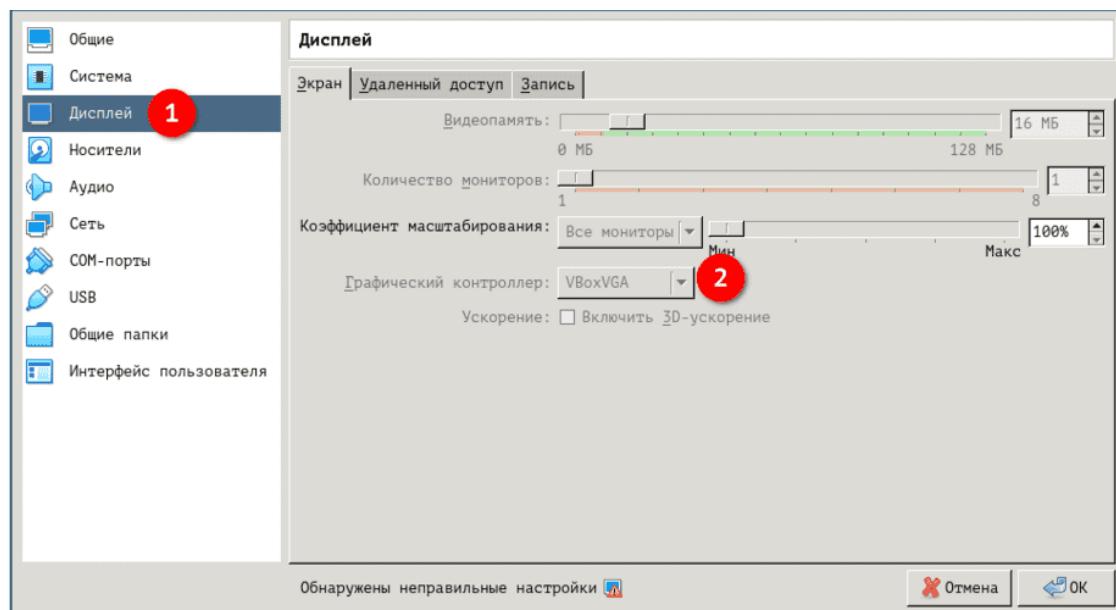
Здесь есть задачи, домашние задания, мануалы и прочая полезная информация:

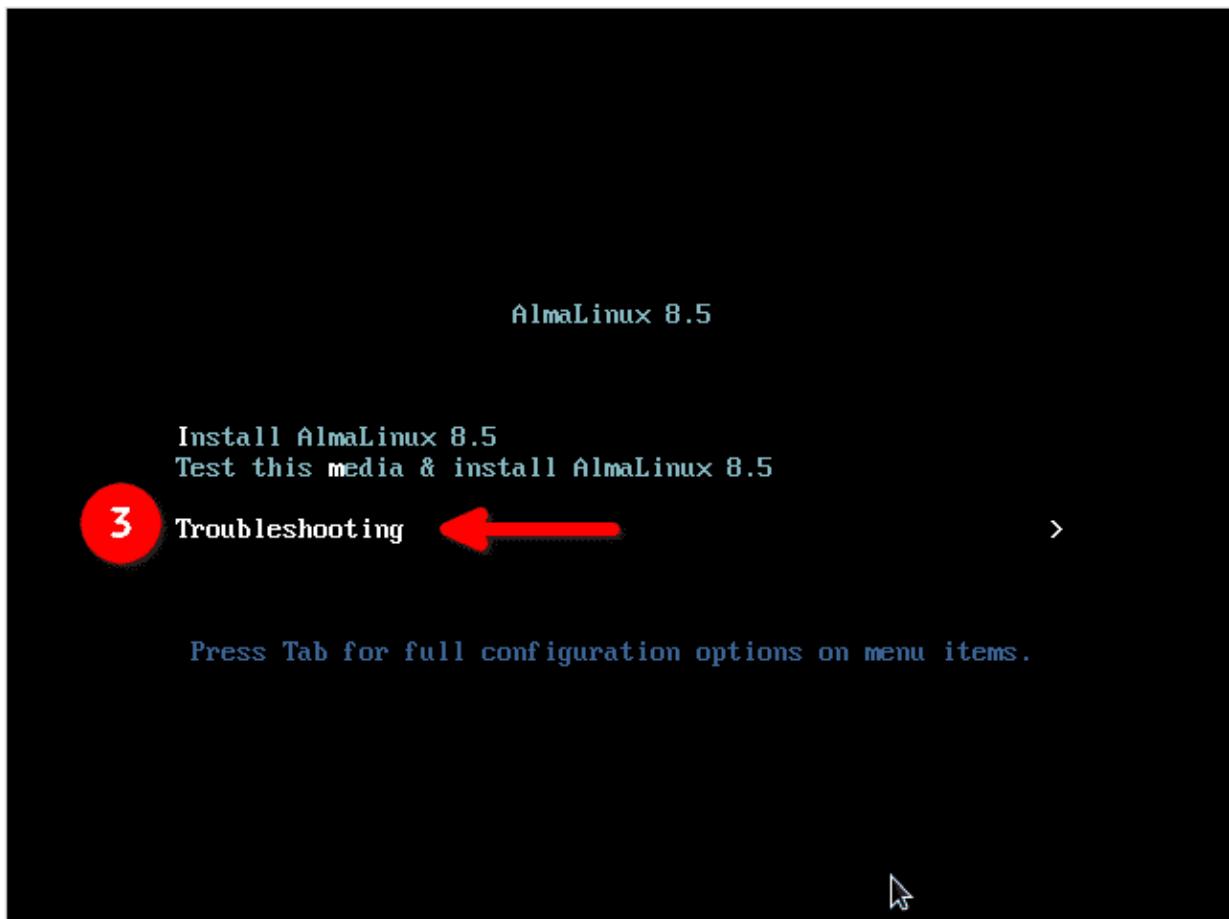
<https://github.com/ksemaev/>

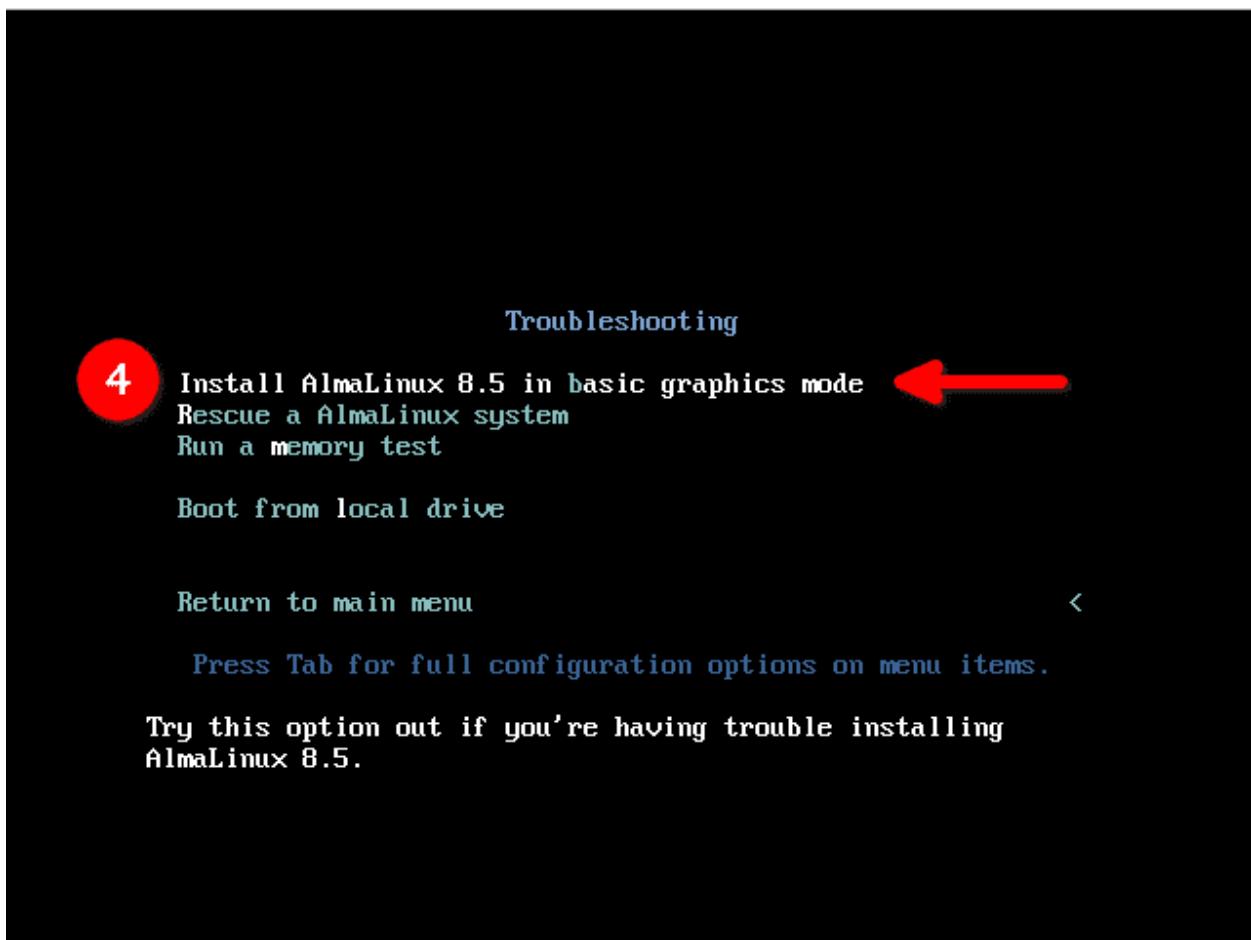
1.1.6 Проблемы и решения

Проблемы и решения

1. Разрешение экрана при установке в VirtualBox







1.1.7 Changelog

Changelog

После публикации курса, чтобы он не устаревал, будут вноситься изменения по мере необходимости. В этом файле будут описываться только изменения существующих уроков, чтобы можно было понять, какой из уроков был обновлён и что было добавлено. Изменять видео на ютубе сложно, поэтому видеокурс в какой-то момент может отставать от текстового и этот файл подскажет вам какие именно темы поменялись.

03

- 16.01.2022 - в теме установки ОС Centos был заменён на AlmaLinux, так как проект Centos прекратили поддерживать.

Глава 2

Курс

2.1 01. Операционные системы и GNU-Linux

2.1.1 01. Операционные системы и GNU-Linux



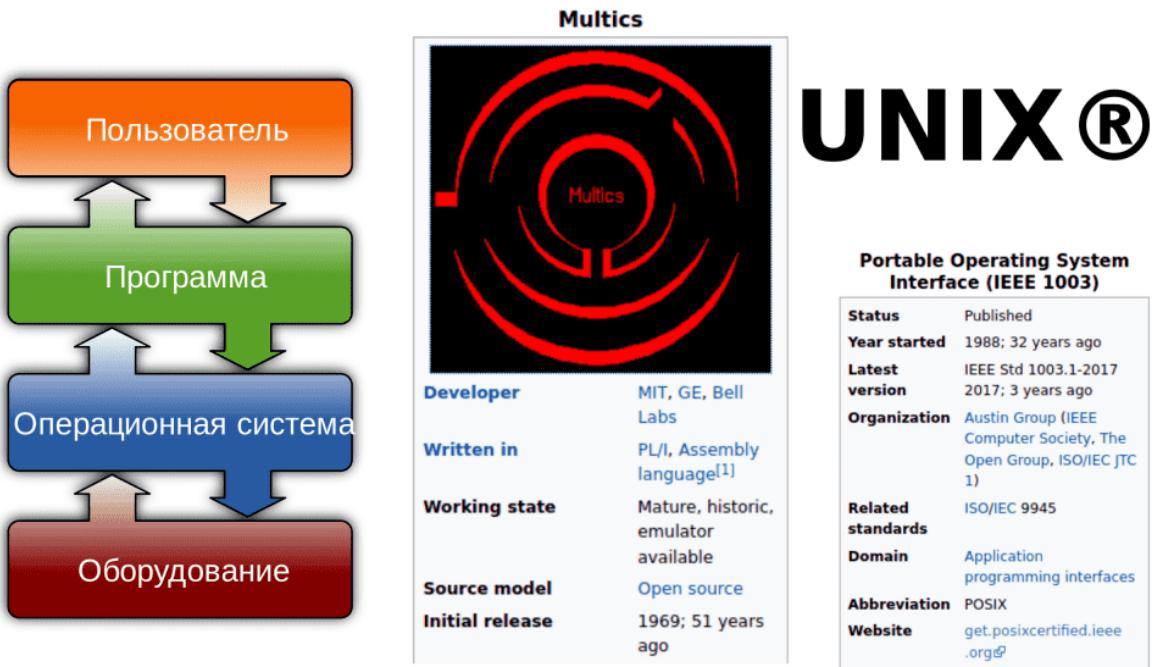
Появление операционных систем

Во времена моего детства на вопрос «какая у тебя операционная система?» люди отвечали «Pentium 4». С появлением мобильных операционных систем Android и iOS и развитием публичного противостояния между Apple и Samsung все больше людей узнало понятие «операционная система». Но для полного понимания этого термина нужно ознакомиться с историей программного обеспечения и понять, как и почему оно появилось и развивалось. Поэтому давайте заглянем в историю программного обеспечения.



Раньше компьютеры были настолько большими и дорогими, что их могли позволить себе только крупные организации и учреждения, такие как университеты и научно-исследовательские центры. При этом компьютеры выполняли только одну задачу в одно время. Под задачей я подразумеваю прикладную программу – программу, с которой работает пользователь. Допустим, ваш браузер, почтовый клиент, текстовой редактор или игра – всё это прикладные программы. Хотя в те годы это были программы для научных и инженерных исследований.

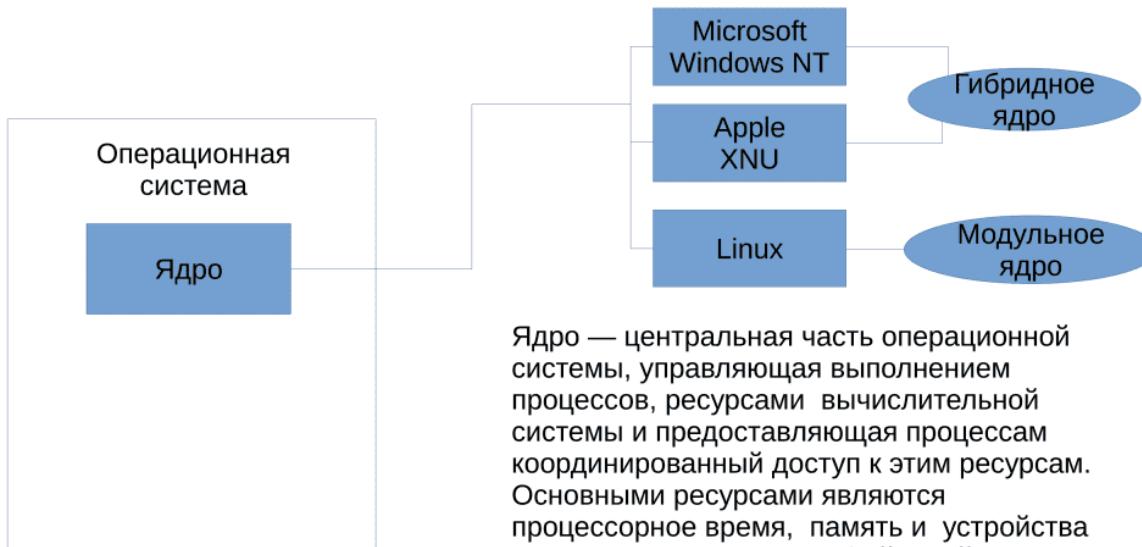
Так вот, как правило, компьютеры стояли в институтах и работники могли долго ждать, пока до них дойдёт очередь поработать с компьютером, как в семье где 10 детей и 1 компьютер. Со временем, мощности компьютеров росли и была необходимость выполнения нескольких задач последовательно или параллельно, а также возможность работать нескольким пользователям одновременно. Была разработана концепция разделения времени, так называемый «time-sharing», на основе которой создали служебные программы, которые решали вопросы многозадачности.



С развитием компьютерной техники, такие служебные программы стали приобретать всё больше функций. Если раньше программы взаимодействовали с оборудованием напрямую, то теперь часть задач брали на себя служебные программы. Они стали эдакой прослойкой между прикладными программами и оборудованием. Набор этих служебных программ начал называться операционной системой, одна из первых реализаций которых называлась Multics. На её идеях создали UNIX, который задал стандарты для современных операционных систем.

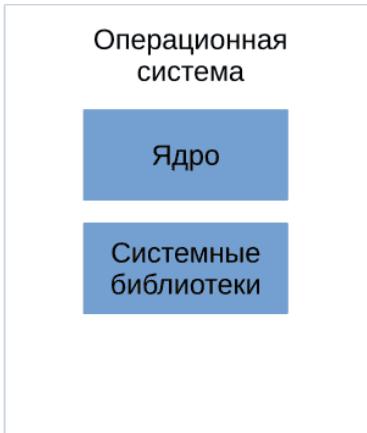
Из чего состоит?

Операционная система – это прослойка между прикладным ПО и оборудованием. Но и ОС можно разделить на 3 составляющие:



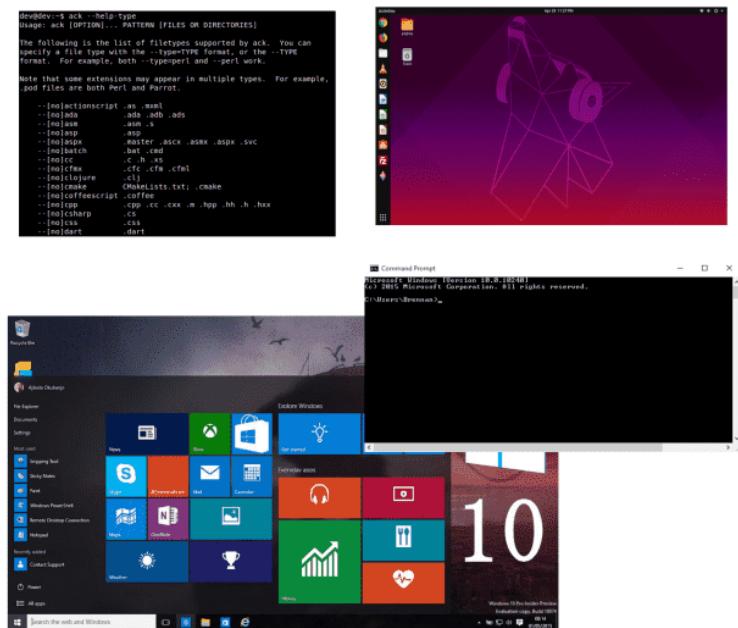
Ядро — центральная часть операционной системы, управляющая выполнением процессов, ресурсами вычислительной системы и предоставляющая процессам координированный доступ к этим ресурсам. Основными ресурсами являются процессорное время, память и устройства ввода-вывода. Доступ к файловой системе и сетевое взаимодействие также могут быть реализованы на уровне ядра.

- Ядро - это программа, отвечающая сразу за несколько важных функций. Одной из ключевых функций ядра является планирование задач, то есть определение того, какие программы и в каком порядке будут выполняться процессором для максимальной производительности и эффективности работы, тот самый «time-sharing». Еще одной важной функцией ядра является управление оперативной памятью – ядро решает, когда и что загружать или выгружать из оперативной памяти. Также ядро отвечает за непосредственную работу с оборудованием за счёт специальных модулей, называемых драйверами. Когда прикладное ПО хочет поработать с оборудованием, допустим, игра хочет обработать какие-то данные и вывести на экран изображение, она обращается к ядру, а ядро пересыпает запрос через драйвер на видеокарту. У ядра есть и другие функции, но на пока этого достаточно. Следует отметить, что существуют различные типы архитектур ядер, и в данном случае мы рассмотрели модульный вариант, который используется в операционной системе Linux.



Библиотéка (от англ. *library*) в программировании — сборник подпрограмм или объектов, используемых для разработки программного обеспечения (ПО).

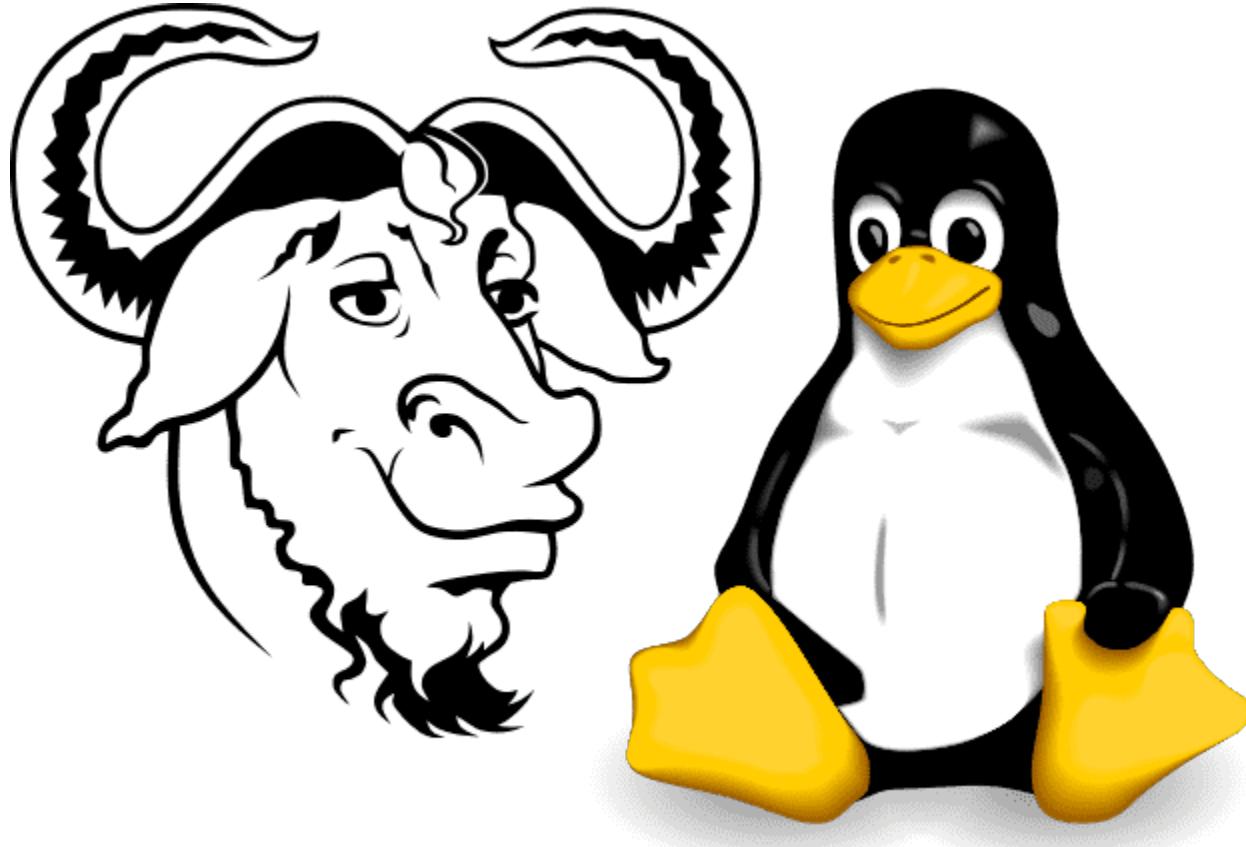
- Системные библиотеки - это важная часть операционной системы, хранящая код, функции и данные, которые используются при запуске и работе прикладных программ. Хотя администраторы редко взаимодействуют с библиотеками напрямую, знание о них может быть полезно при устранении проблем с прикладными программами.



- Оболочка и утилиты. Одна из важных функций операционной системы – дать пользователю интерфейс взаимодействия с компьютером. Интерфейс может быть как графическим, так и текстовым. Не стоит думать, что текстовый интерфейс – это какое-то окно в скрытый мир компьютера, через которое вы можете делать с компьютером всё что угодно. Да, текстовый интерфейс, как

правило, несколько функциональнее графического, но его писали люди для людей и функции у него как у графического интерфейса – дать возможность запускать программы, работать с файлами и т.п. Современные операционные системы содержат сотни небольших программ, называемых утилитами, которые могут служить как для самой системы для каких-то внутренних задач по обслуживанию, так и для пользователей для какого-то базового функционала, а также для диагностики и решения проблем.

GNU/Linux и дистрибутивы



Коммерческие компании, занимающиеся разработкой операционных систем, дают название своим продуктам Windows, MacOS, Android или iOS. Но в случае с GNU/Linux всё сложилось несколько иначе. Ядро, называемое Linux, разрабатывают одни люди, точнее даже сказать тысячи людей и компаний, а библиотеки и утилиты сотни других людей и компаний. Что-то остается ещё с 80-ых, а что-то появляется и исчезает каждый год. Как правило, какие-то базовые утилиты разрабатывает организация GNU, а большинство остальных утилит и оболочек выпускается под лицензией GNU GPL (в том числе ядро Linux).

Существуют люди и компании, которые берут эти компоненты, соединяют и получают готовую операционную систему. Но у разных людей свои видения и свои цели, в итоге получается много разных вариаций этой операционной системы, которые называют дистрибутивами. Ubuntu, Debian, Centos, RedHat Enterprise Linux – всё это дистрибутивы, которые используют программы GNU и ядро Linux. Есть дистрибутивы, которые отличаются только набором предустановленных программ и настройками графического интерфейса, а есть дистрибутивы, в которых абсолютно разный подход к обновлениям, поддержке и даже наличие каких-то специфичных программ. Но так как все эти дистрибутивы в основе имеют программы GNU и ядро Linux - их можно условно объединить под одним названием GNU/Linux.

Распространение ОС

Современные операционные системы для персональных компьютеров, как правило, распространяются в виде специальных файлов с расширением ISO. Этот файл – так называемый «образ диска» – содержит программу-установщик операционной системы и для установки его следует записать на диск или флешку и загрузить компьютер с этого устройства. Несмотря на то, что возможно установить несколько операционных систем на один компьютер, ошибка при установке может привести к потере данных, поэтому к процессу установки следует отнестись с особой ответственностью. Мы рассмотрим основные шаги по установки операционной системы в отдельной части.

Как правило, дистрибутивы GNU/Linux можно скачать с официальных сайтов дистрибутива бесплатно и без всяких регистраций, а коммерческие операционные системы предоставляют доступ к этому файлу только после покупки лицензии – специального документа, разрешающего использование копии программного обеспечения. Некоторые операционные системы жёстко привязаны к определённому железу – как например, MacOS, но большинство ставится на различное оборудование при наличии драйверов.

В этой теме я предоставил минимально необходимую информацию об операционных системах и GNU/Linux. Советую обратить внимание на «Полезные ссылки», указанные в практике к данной главе.

2.1.2 Практика

Полезные ссылки

[Джим Землин - Чему сфера технологий научилась у Линуса Торвальдса](#)

Небольшое выступление, рассказывающее о роли Linux в современном мире. Хотя само видео довольно старое, общая картина та же.

[Revolution OS](#)

Интересный фильм, рассказывающий историю появления и развития свободного программного обеспечения.

[Вкратце о лицензиях Open Source](#)

Лицензии – важная часть программного обеспечения, определяющая, что можно делать с ПО, а что нельзя. Важно иметь представление, какие проблемы возникают из-за закрытого ПО, и почему так важны лицензии «Free Software» и «Open Source».

[Зачем переходить на GPLv3](#)

Написано несколько специфично, но раскрываются проблемы – тивоизация, программные патенты и т.п.

Вопросы

1. Для чего пользователям компьютер?
2. Как операционная система помогает пользователям?
3. Что такое операционная система?
4. Для чего создали операционную систему?
5. На какие составляющие делится операционная система?
6. Какие задачи стоят перед ядром операционной системы?
7. Могут ли программы работать без операционной системы?
8. Что такое исходный код и компилятор?
9. Что такое GNU?

10. Что такое «свободное программное обеспечение»(Free Software) и чем оно отличается от бесплатного (Freeware)?
11. Различия и сходства свободного программного обеспечения (Free Software) и открытого ПО (Open Source Software).
12. Как называется не свободное программное обеспечение?
13. Для чего нужны лицензии Creative Commons?
14. Что такое тивоизация?
15. Что такое Linux?
16. Почему операционная система называется GNU/Linux?
17. Что такое дистрибутивы и почему их много?
18. Какой дистрибутив выбрать и почему?
19. Какая связь между GNU/Linux и UNIX?
20. Можно ли назвать Linux UNIX-ом? Какое условие нужно выполнить, чтобы ОС можно было назвать UNIX-ом?
21. Можно ли зарабатывать на Свободном Программном Обеспечении и как?
22. В чём отличие коммерческих дистрибутивов от пользовательских?
23. Есть дистрибутив Centos, который является почти полной копией RHEL, но при этом распространяется бесплатно. Зачем компании покупают RHEL, если можно использовать CentOS?
24. У некоторых дистрибутивов есть различные релизы - стабильный, тестировочный и нестабильный. В чём могут быть отличия, из-за чего и для чего?

2.2 02. Виртуализация

2.2.1 02. Виртуализация

Виртуализация



Для чего нужна?

Основа современной ИТ инфраструктуры состоит из 3 элементов – безопасность, отказоустойчивость и производительность.

- Безопасность требует, чтобы на одной операционной системе выполнялась одна задача – допустим веб-сервер или почтовый сервер. Это позволит легче контролировать уязвимые места и смягчить потери в случае какой-либо угрозы. Запомните, хорошая безопасность строится не только на превентивных мерах, но и на мерах смягчения. Когда безопасность дала сбой и вам нужно уберечь как можно больше, лучше потерять одну систему, чем несколько.
- Отказоустойчивость требует наличия нескольких копий одного сервиса на различных компьютерах, что позволит сервису работать без перебоев при проблемах с одним компьютером, электричеством в здании или даже стихийным бедствием в городе.
- Производительность требует оптимального использования ресурсов и возможность быстрого масштабирования, чтобы сервис не использовал лишние ресурсы во время простоя и не падал во время пиковых нагрузок.

Всё это звучит страшно как для администратора, так и для кармана начальника. Но есть технологии, которые позволяют упростить и удешевить всё это в разы. Одна из таких технологий – виртуализация, которой пользуются практически все. Она позволяет запускать несколько виртуальных компьютеров с полноценными операционными системами внутри одного реального компьютера.

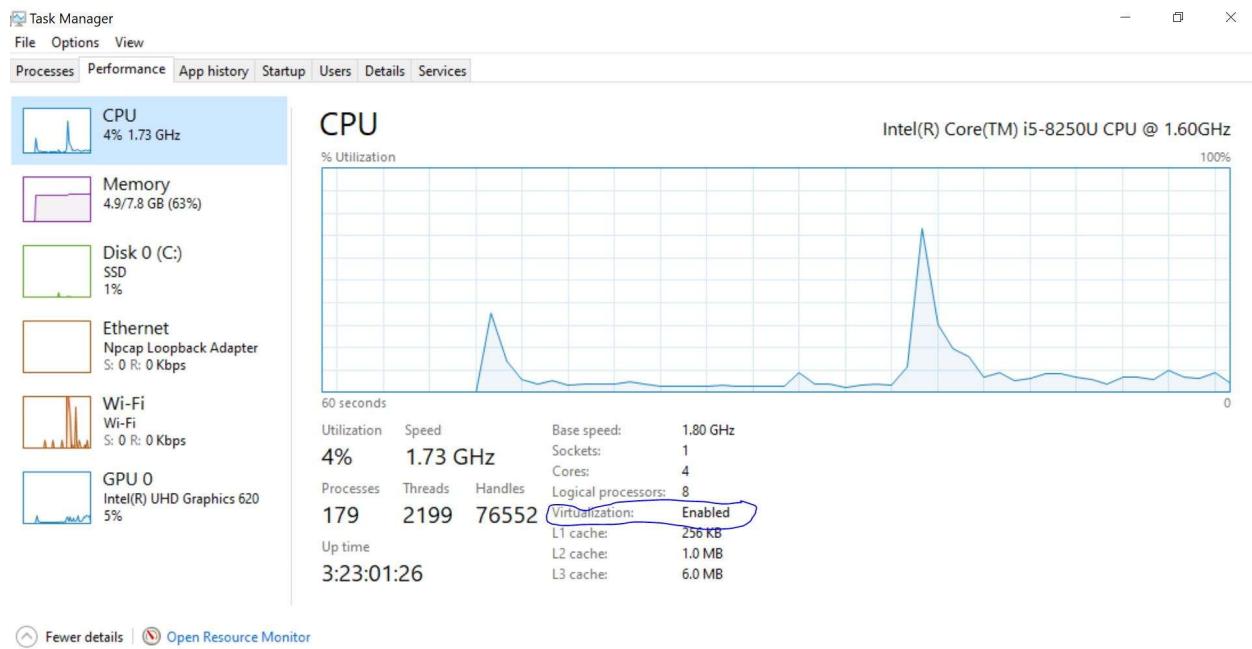


Гипервизор

Для виртуализации используются специальные программы, называемые гипервизорами. Для её работы требуется операционная система и процессор, поддерживающий виртуализацию. Большинство современных процессоров поддерживает эту технологию. Операционная система, на которой запускается гипервизор, называется хост-системой или хостом, а «виртуальные компьютеры», запущенные на гипервизоре – виртуальными машинами, виртуалками или гостевыми машинами.

Есть множество реализаций гипервизоров, которые отличаются функционалом и ценой. Нам, для обучения линуксам, вполне подойдут и бесплатные варианты – к примеру, VirtualBox. Но, прежде чем установить гипервизор, стоит убедиться, поддерживается ли на вашем компьютере виртуализация и включена ли она.

Поддерживает ли ваш компьютер?



Если на вашем компьютере Windows, запустите диспетчер задач, во вкладке производительность выберите CPU и ищите строчку Виртуализация. При значение «Enabled» всё нормально, ваш компьютер поддерживает виртуализацию и она включена. Если «Disabled» – то нужно зайти в BIOS и включить виртуализацию. Я дам [ссылку](#), как это сделать, так как этот процесс может различаться в зависимости от компьютера. Если у вас на компьютере GNU/Linux, то я все же рекомендую использовать виртуальные машины для обучения. Чтобы убедиться, поддерживается ли виртуализация вашим компьютером на GNU/Linux, выполните команду (`lscpu | grep Virtualization`). Если команда выдала ответ – то всё хорошо, если нет – попробуйте проверить в BIOS, по ссылке выше.

```
lscpu | grep Virtualization
Virtualization: VT-x
```

Установка VirtualBox

После этого идём на сайт virtualbox.org → [Downloads](#) – выбираем «Windows hosts» и ждём, пока всё скачается, а затем устанавливаем гипервизор, следя настройкам по умолчанию. После установки гипервизора, скачиваем с сайта «VirtualBox Extensions Pack» и устанавливаем.

VirtualBox 6.1.14 platform packages

- [Windows hosts](#) **Если на компе Windows**
- [OS X hosts](#)
- [Linux distributions](#) **Если на компе GNU/Linux**
- [Solaris hosts](#)

The binaries are released under the terms of the GPL version 2.

See the [changelog](#) for what has changed.

You might want to compare the checksums to verify the integrity of downloaded packages. *The SHA256 checksums should be favored as the MD5 algorithm must be treated as insecure!*

- [SHA256 checksums, MD5 checksums](#)

Note: After upgrading VirtualBox it is recommended to upgrade the guest additions as well.

VirtualBox 6.1.14 Oracle VM VirtualBox Extension Pack

- [All supported platforms](#)

После всех манипуляций перезагружаем компьютер. Если у вас GNU/Linux, VirtualBox можно скачать и установить как из репозитория, так и с сайта. Дополнительный шаг на Linux – добавить вашего пользователя в группу vboxusers (sudo usermod -aG vboxusers username). Здесь username – логин вашего пользователя.

```
sudo usermod -aG vboxusers username
```

2.2.2 Практика

Дополнительно

Традиционная эра развертывания: Ранее организации запускали приложения на физических серверах. Не было никакого способа определить границы ресурсов для приложений на физическом сервере, и это вызвало проблемы с распределением ресурсов. Например, если несколько приложений выполняются на физическом сервере, могут быть случаи, когда одно приложение будет занимать большую часть ресурсов, и в результате чего другие приложения будут работать хуже. Решением этого было запустить каждое приложение на другом физическом сервере. Но это не масштабировалось, поскольку ресурсы использовались не полностью, из-за чего организациям было накладно поддерживать множество физических серверов. Эра виртуального развертывания: В качестве решения была представлена виртуализация. Она позволила запускать несколько виртуальных машин (ВМ) на одном физическом сервере. Виртуализация изолирует приложения между виртуальными машинами и обеспечивает определенный уровень безопасности, поскольку информация одного приложения не может быть свободно доступна другому приложению. Виртуализация позволяет лучше использовать ресурсы на физическом сервере и обеспечивает лучшую масштабируемость, поскольку приложение можно легко добавить или обновить, кроме этого снижаются затраты на оборудование и многое другое. С помощью виртуализации можно превратить набор физических ресурсов в кластер одноразовых виртуальных машин. Каждая виртуальная машина представляет собой полноценную машину, на которой выполняются все компоненты, включая собственную операционную систему, поверх виртуализированного оборудования.

<https://kubernetes.io/ru/docs/concepts/overview/what-is-kubernetes/>

Вопросы

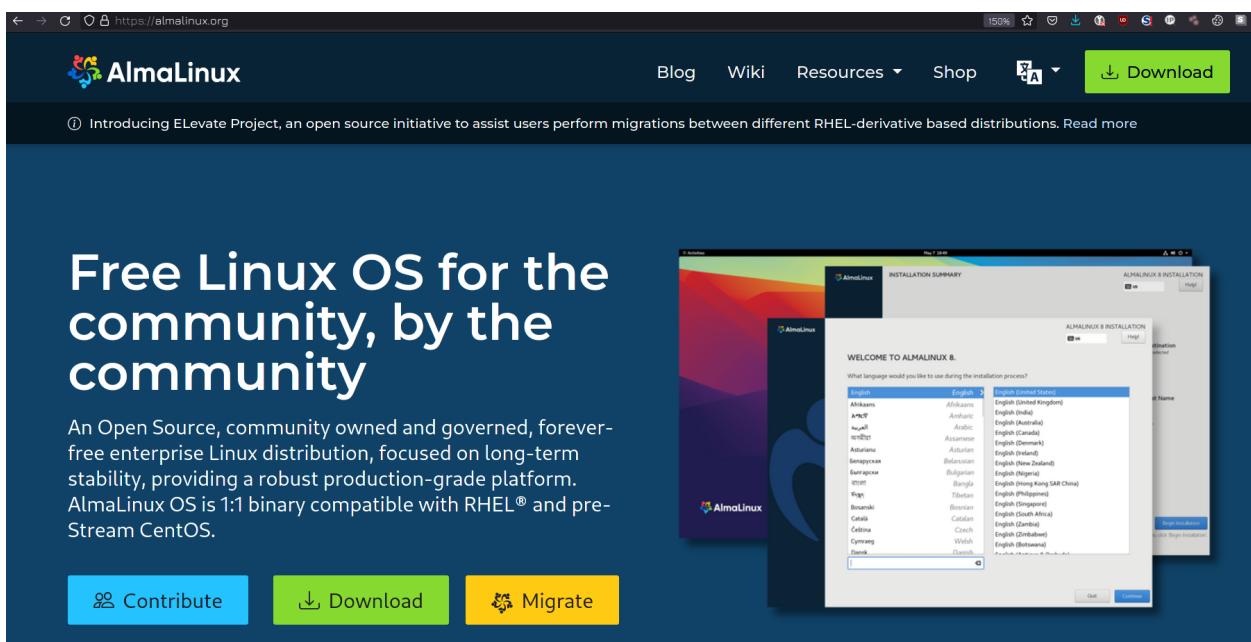
1. Для чего нужна виртуализация?
2. Для чего нужен гипервизор?
3. Что называется «хост-системой» (host, host)?

4. Что называется «виртуальной машиной» (виртуалка, virtual machine, vm)?
 5. Что называется «гостевой ОС» (guest os, guest)?
 6. Что такое и для чего нужны снэпшоты?

2.3 03. Установка AlmaLinux

2.3.1 03. Установка AlmaLinux

Как скачать?



На основе исходного кода Red Hat Enterprise Linux (RHEL) создается полностью совместимый, бесплатный дистрибутив Alma Linux. Именно его мы будем использовать для подготовки к сертификации от Red Hat.

AlmaLinux ISOs links

You can find the list of available AlmaLinux architectures and versions below.

Also you can use a BitTorrent file for downloading ISOs. It might be faster than direct downloading from the mirrors.

A torrent file can be found on all mirrors in the folder with ISO files.

Architecture	Versions
x86_64	8.5-beta 8.5
aarch64	8.5-beta 8.5
ppc64le	8.5-beta 8.5

AlmaLinux ISOs links

You can find the list of available AlmaLinux architectures and versions below.

Also you can use a BitTorrent file for downloading ISOs. It might be faster than direct downloading from the mirrors.

A .torrent file can be found on all mirrors in the folder with ISO files.

The following mirrors are nearest to you:

- mirror.bardia.tech ↗ ← Первая ссылка
- mirror.hosting.com.tr ↗
- mirror.vargonen.com ↗
- mirrors.ereznet.co.il ↗
- alma.intercloud.co.il ↗
- mirror.domainhizmetleri.com ↗
- almalinux.turhost.com ↗
- almalinux.netforce.hosting ↗
- mirror.vsys.host ↗
- KACST - MAEEN ↗

© 2021 AlmaLinux OS Foundation

AlmaLinux можно скачать с сайта almalinux.org, как с помощью прямой [ссылки](#), так и при помощи торрента. Весь данный курс был сделан на дистрибутиве CentOS Linux, который также являлся копией RHEL. Но с недавних пор CentOS прекратили поддерживать и для актуальности курса я выбрал AlmaLinux. Можно сказать, что единственными отличиями являются название, логотип и сайт, поэтому весь курс переделывать смысла нет. Просто имейте ввиду, что там где я буду говорить CentOS - для вас это Alma.

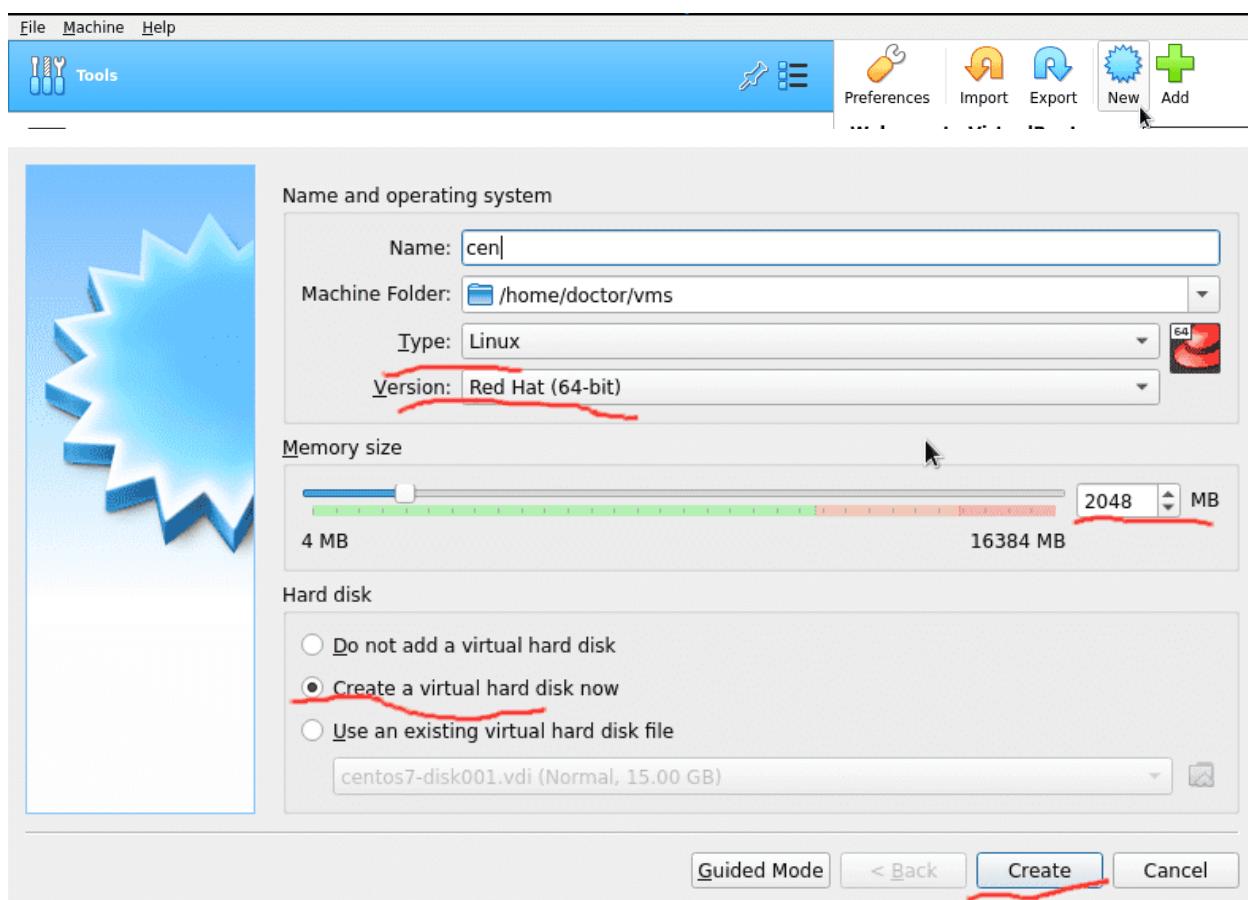


Index of /almalinux/8.5/isos/x86_64

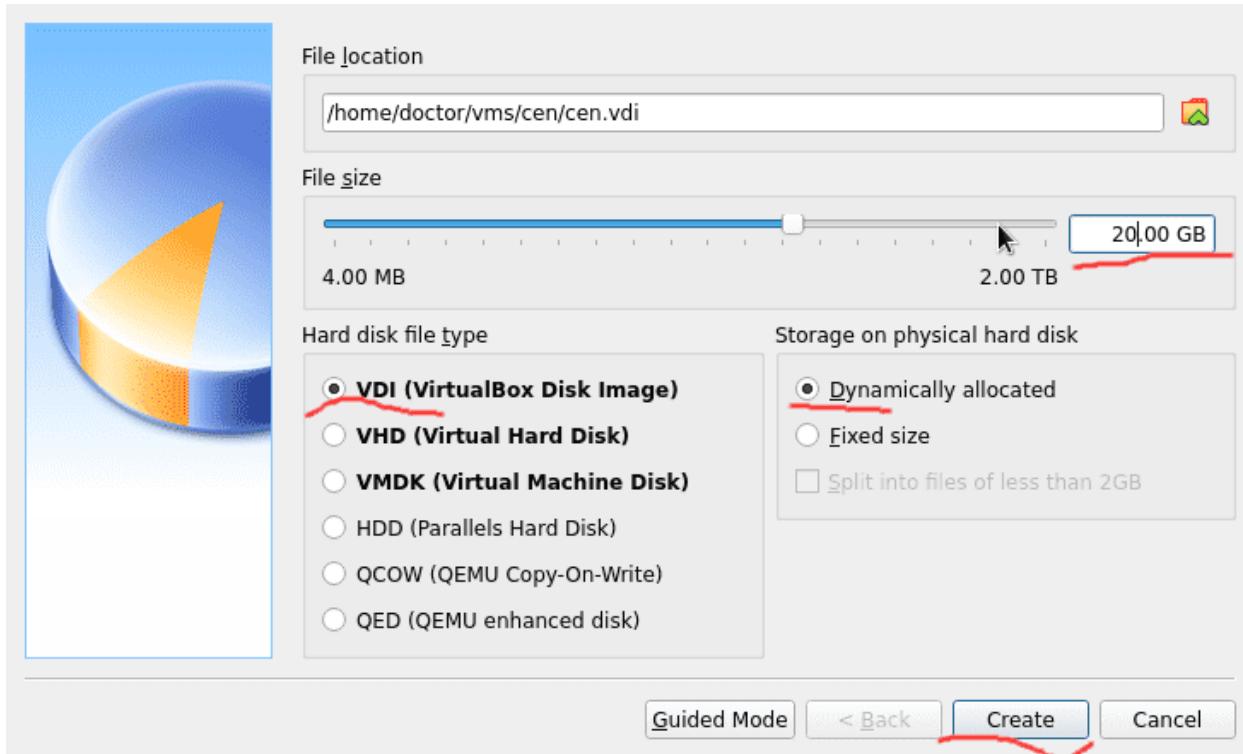
Name	Last modified	Size	Description
Parent Directory		-	
AlmaLinux-8.5-x86_64-boot.iso	2021-11-11 20:05	738M	
AlmaLinux-8.5-x86_64-boot.iso.manifest	2021-11-11 20:07	626	
AlmaLinux-8.5-x86_64-dvd.iso	2021-11-11 20:12	9.9G	
AlmaLinux-8.5-x86_64-dvd.iso.manifest	2021-11-11 20:12	501K	
AlmaLinux-8.5-x86_64-minimal.iso	2021-11-11 19:46	1.9G	
AlmaLinux-8.5-x86_64-minimal.iso.manifest	2021-11-11 19:46	94K	
AlmaLinux-8.5-x86_64.torrent	2021-11-12 12:51	64K	
CHECKSUM	2021-11-11 20:58	1.2K	

Мы можем скачать образ DVD диска на 9 гигабайт, в котором есть весь необходимый для установки софт, либо минимальный образ на 700 мегабайт, который, при установке, будет докачивать необходимое из интернета. Так как я буду устанавливать систему несколько раз - с графическим интерфейсом, без графического интерфейса и т.д. - я скачиваю DVD образ на 9 гигабайт.

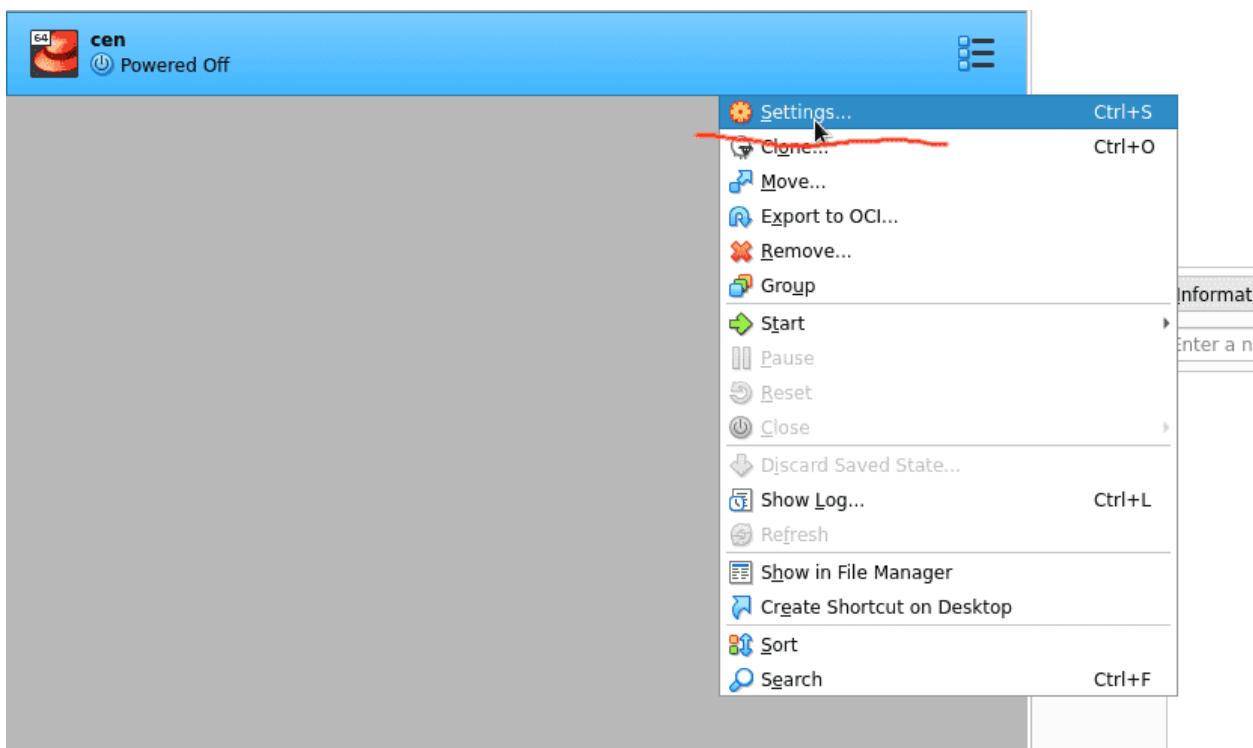
Подготовка VirtualBox



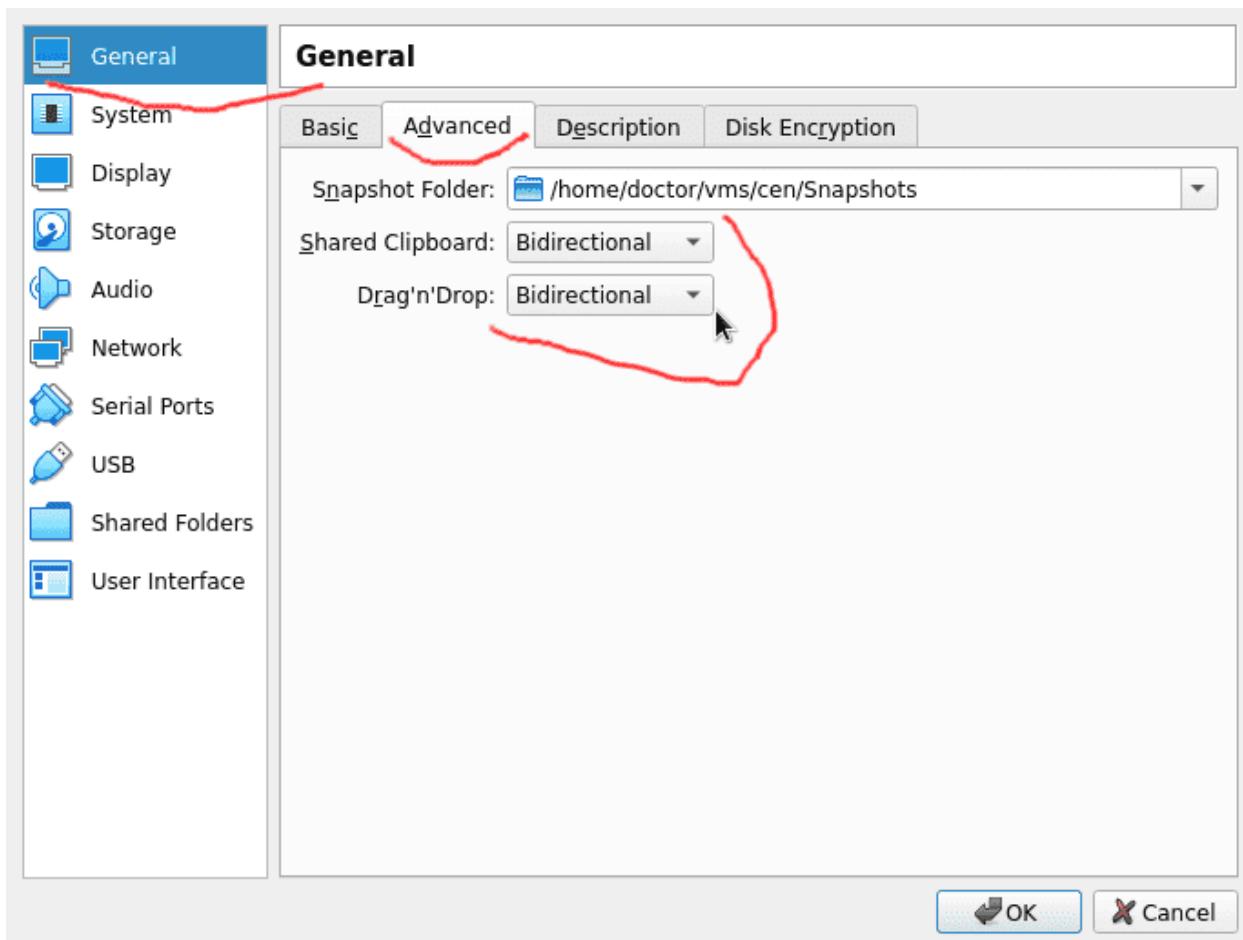
После того как всё скачается, запускаем VirtualBox и создаём новую виртуальную машину. Если в названии указать «Сен», то программа сама предложит тип операционной системы (Linux) и версию (Red Hat (64 bit)). Дальше указываем количество оперативной памяти для виртуальной машины. Минимальное значение – 2 гигабайта, но, в дальнейшем, без графического интерфейса, можно обойтись и меньшими параметрами.



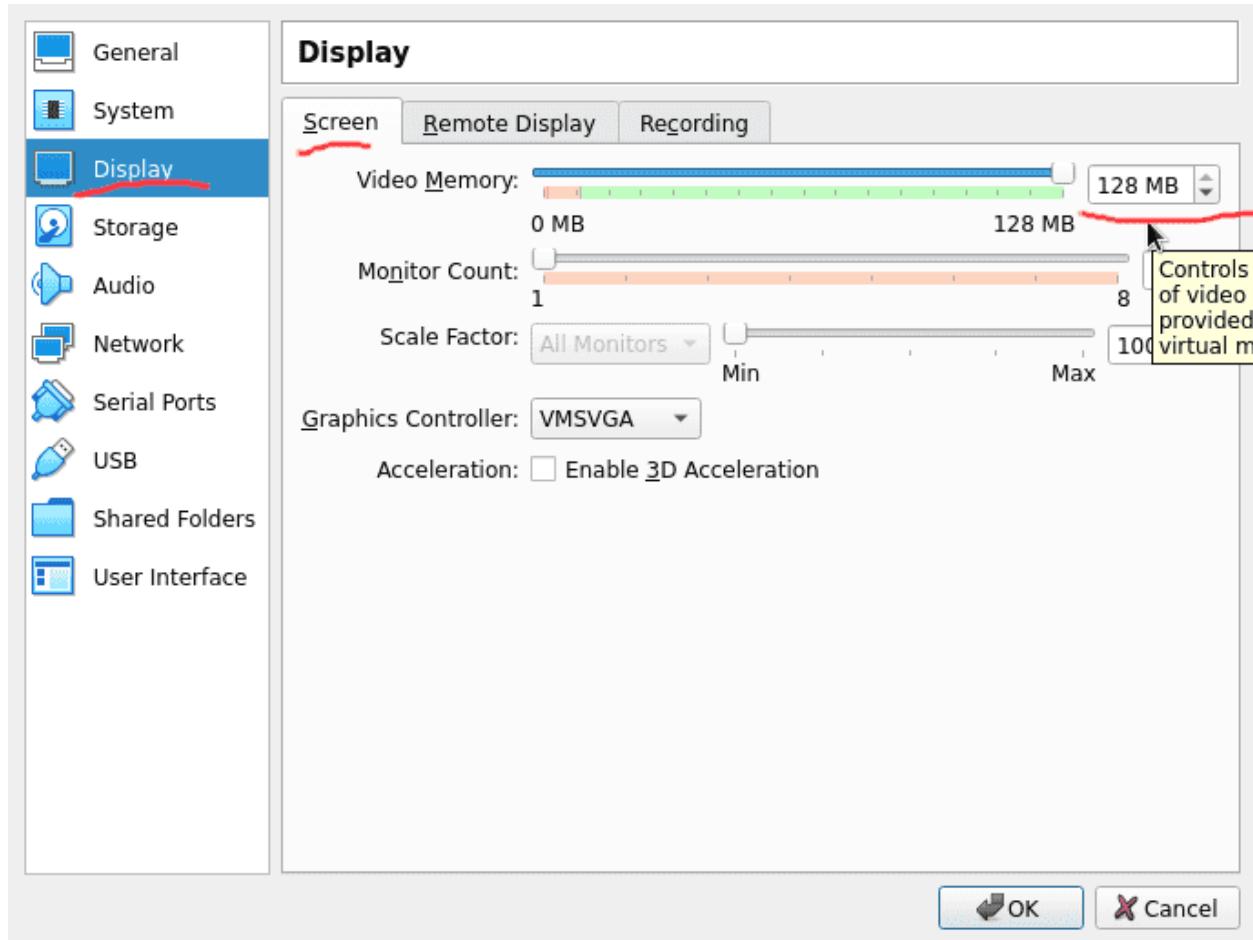
Дальше выбираем «Создать виртуальный диск», так как он нам нужен для операционной системы, а готового диска у нас нет. После этого указываем тип диска, в зависимости от того, будет ли нам нужно этот виртуальный диск переносить на другие платформы или нет. Так как мы этого не планируем, выбираем VDI – родной формат для VirtualBox. Дальше выбираем, как наш виртуальный диск займёт пространство на нашем реальном диске. Тут два варианта – либо мы даём виртуальной машине диск с каким-то виртуальным размером, реальный размер которого будет изменяться в зависимости от действительно занятого пространства, либо мы сразу выделяем место для виртуального диска, после чего это место станет недоступно для хоста, независимо от того, есть там данные или нет. Есть небольшая разница в производительности в пользу фиксированного размера, но динамический метод оставляет больше места на компьютере. Часто такие подходы к выделению пространства называют thin и thick provisioning соответственно. После этого выбираем, где будет храниться виртуальный диск и его объём. Объём зависит от количества устанавливаемых программ и хранимых файлов. При минимальной установке желательно давать объём не меньше 8 гигабайт, а в нашем случае я укажу 20 гигабайт, которых хватит с запасом.



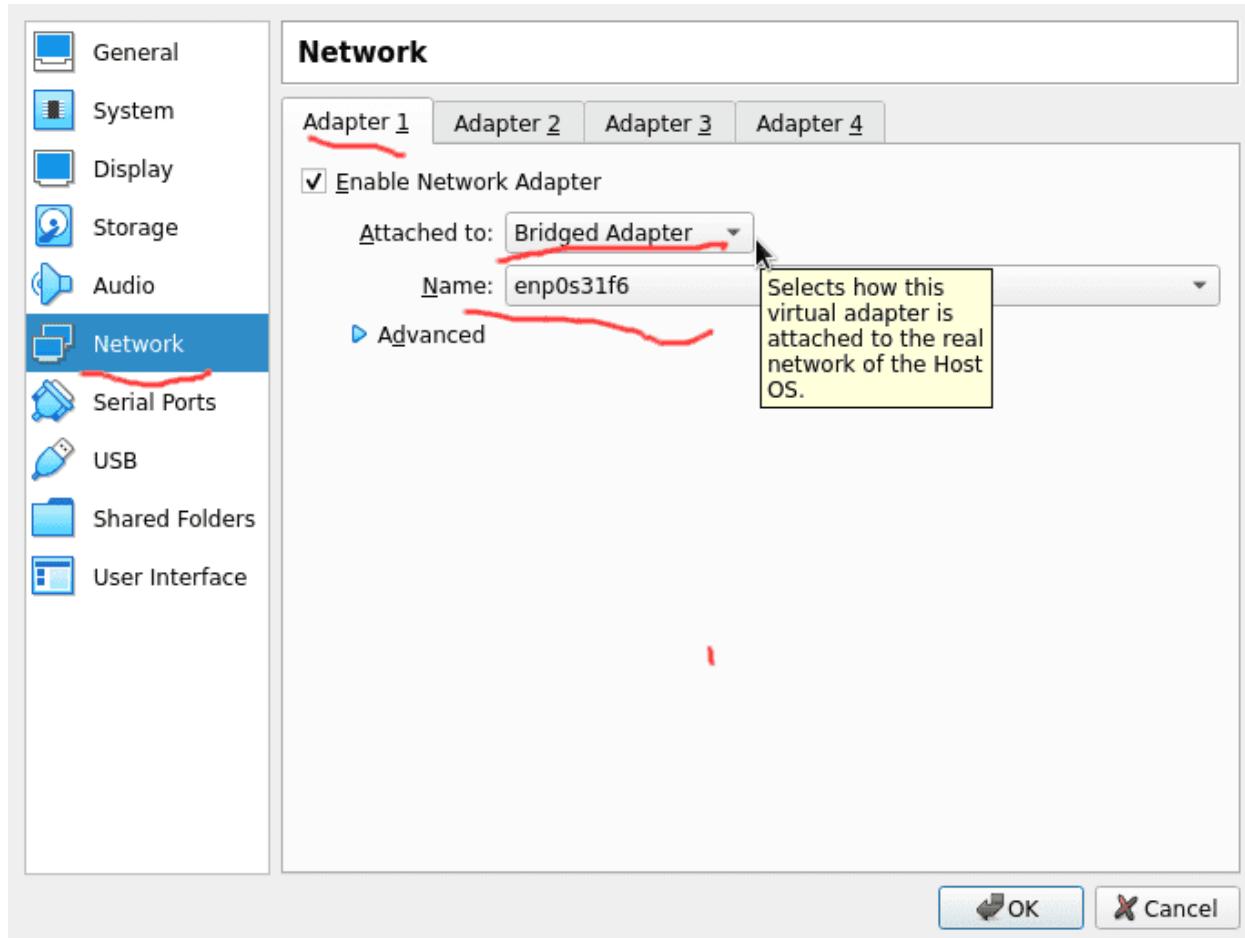
После нажатия Create кликаем правой кнопкой мыши на появившейся виртуальной машине и заходим в настройки.



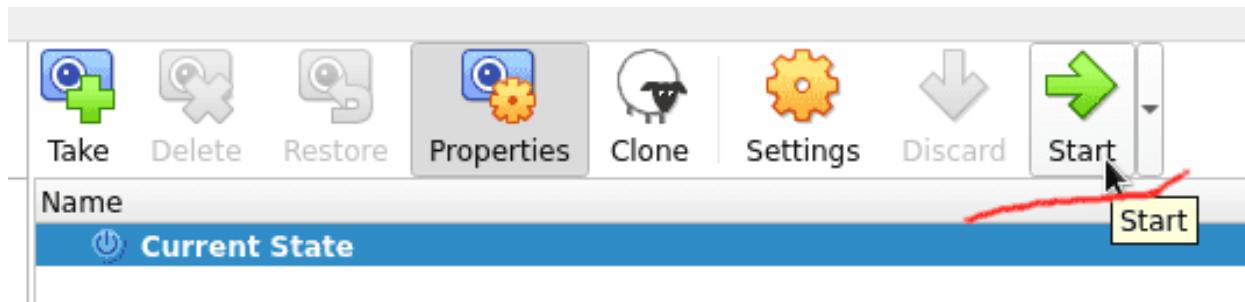
Во вкладке General – Advanced – Shared Clipboard выставляем значение Bidirectional, что позволит копировать текст между хостом и виртуалкой, то есть создаст общий буфер обмена.



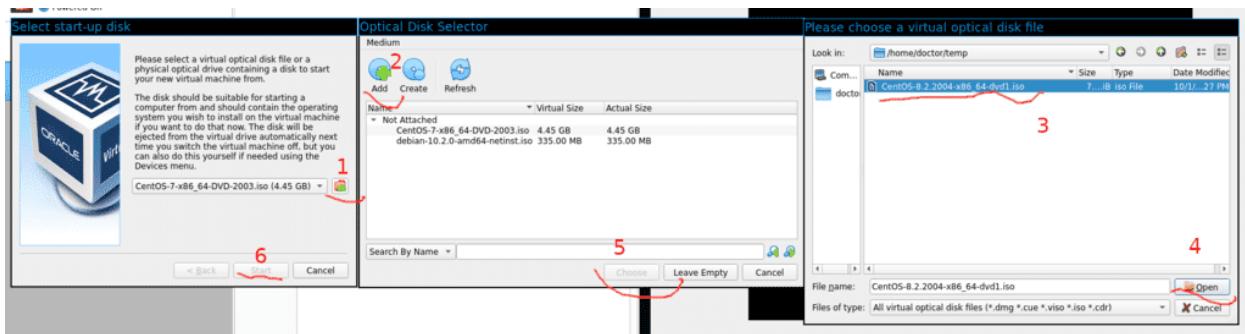
Во вкладке Display – Screen - Video Memory повышаем значение до 64 Мб или выше, так как при малых значениях графический интерфейс виртуальной машины может подвисать.



Во вкладке Network – Adapter 1 меняем Attached to: NAT на Bridged Adapter и указываем сетевой адаптер, к которому подсоединенна сеть на нашем компьютере, что позволит виртуальной машине быть в той же сети, что и хост.

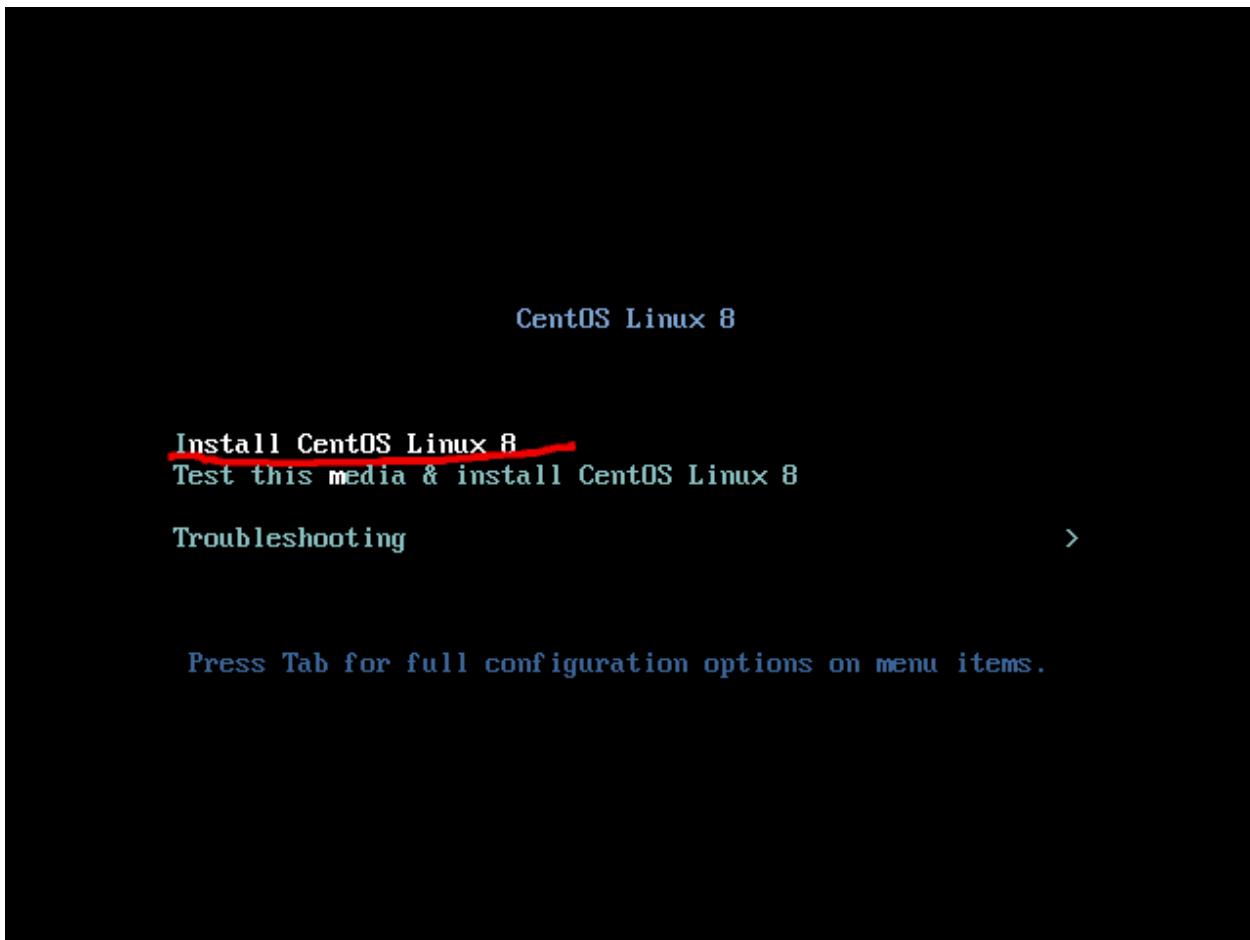


После этого нажимаем OK и Start, что запускает виртуальную машину.

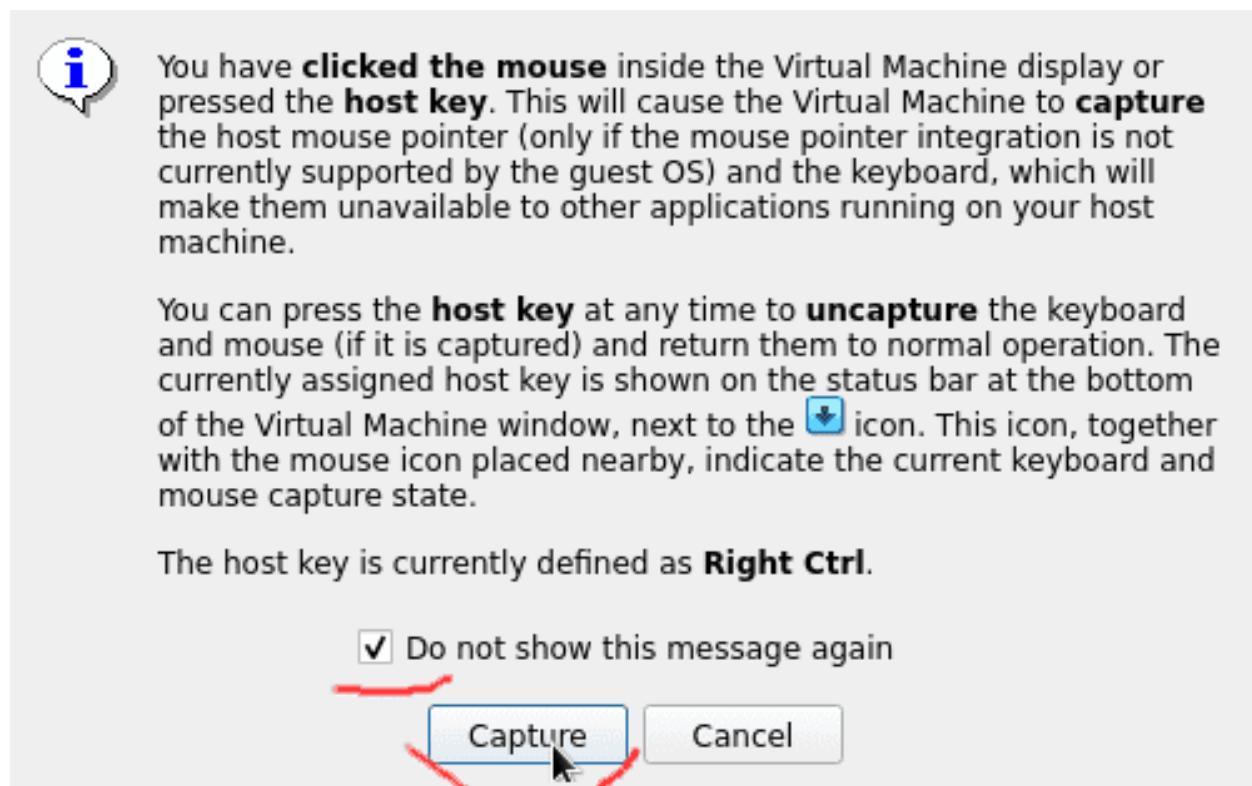


VirtualBox предложит указать образ, с которого будет установлена операционная система. В появившемся окне выбираем Add и указываем ISO файл CentOS-а, который мы скачали. Дальше выбираем этот образ и нажимаем Choose.

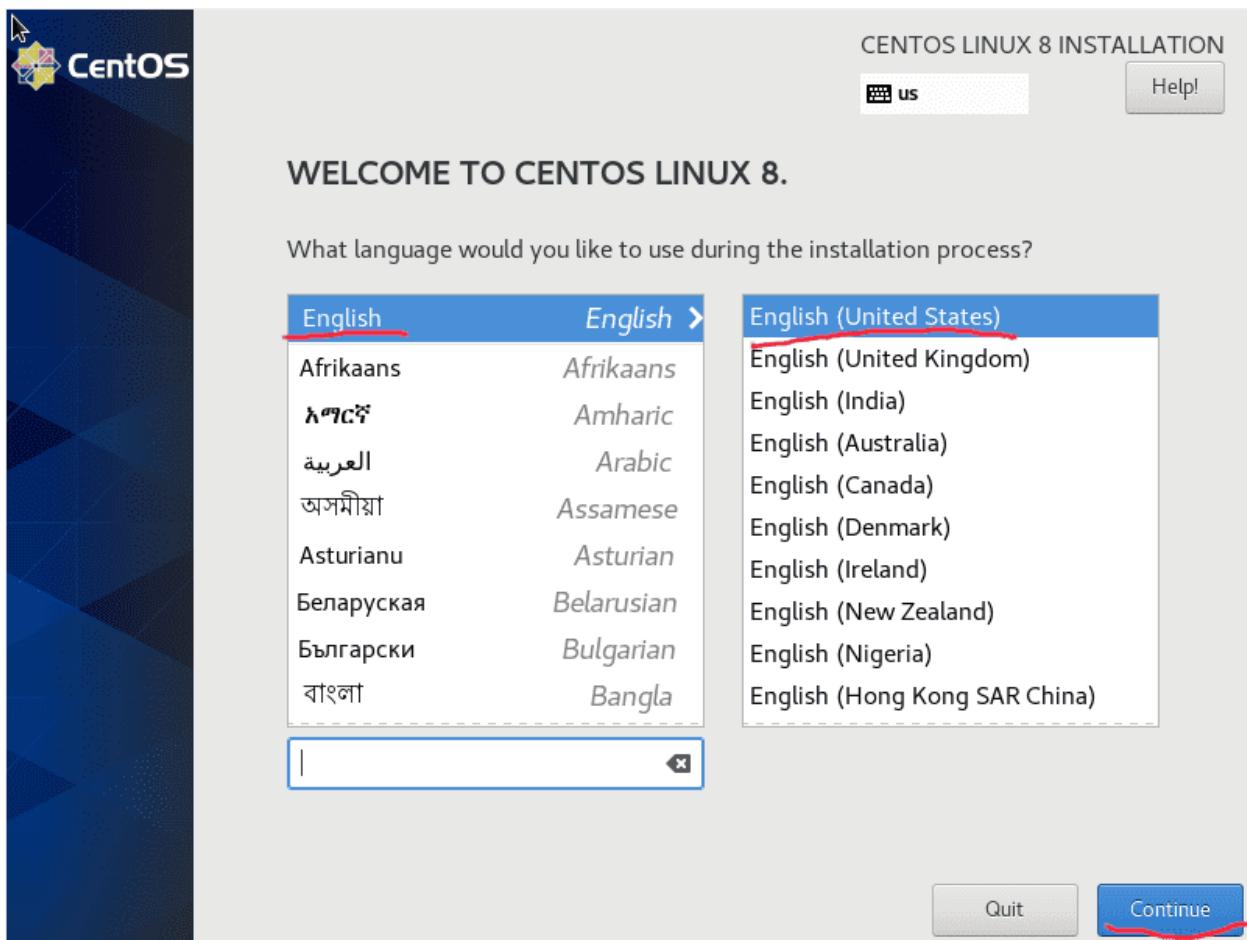
Установка CentOS



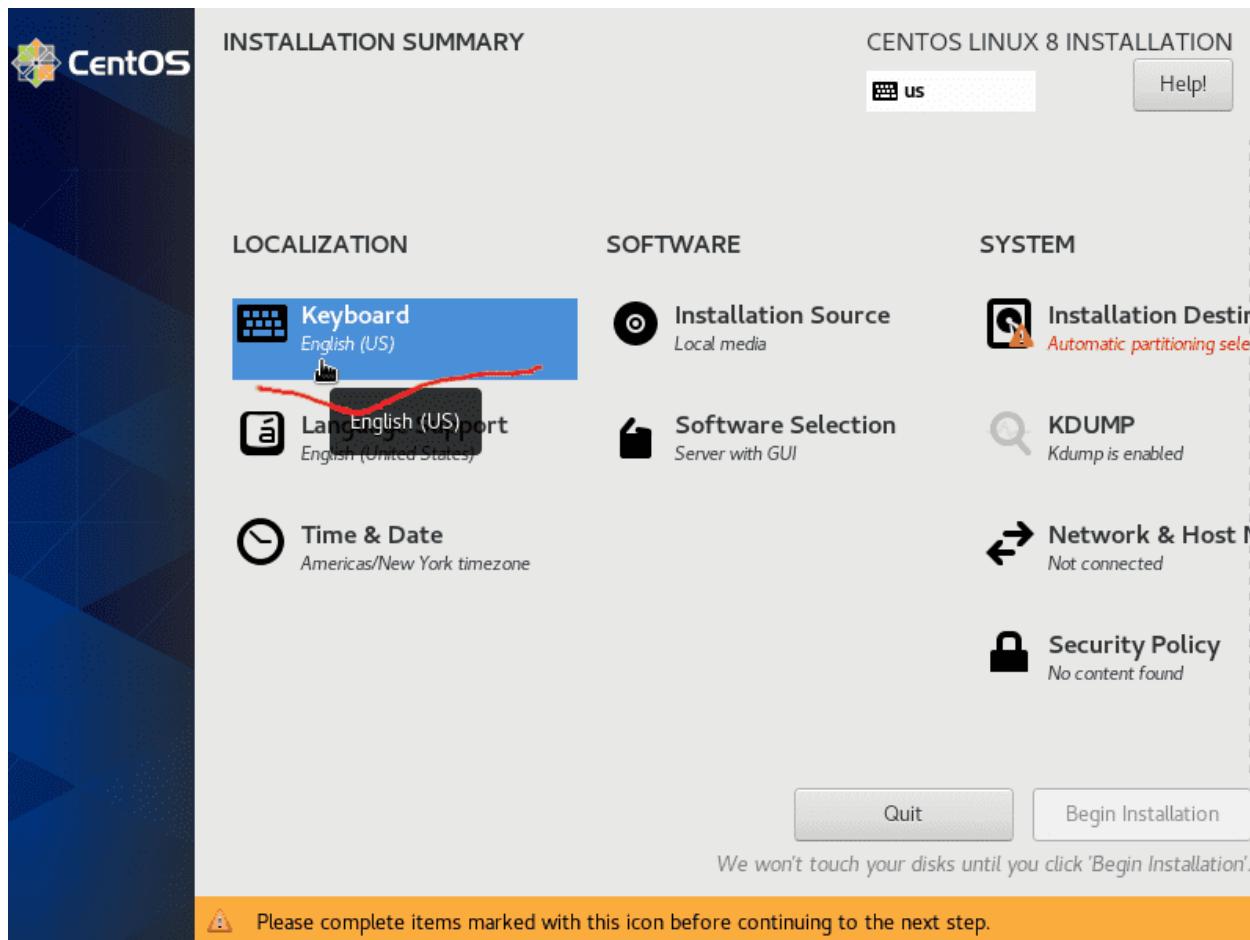
В новом окне появится меню, где по умолчанию выделено «Test this media». Эта опция начнёт проверять установщик на целостность, что занимает время. Выберите опцию «Install CentOS Linux 8» (стрелка вверх) и нажмите Enter. Затем загрузится графический установщик, в котором можно работать мышкой.

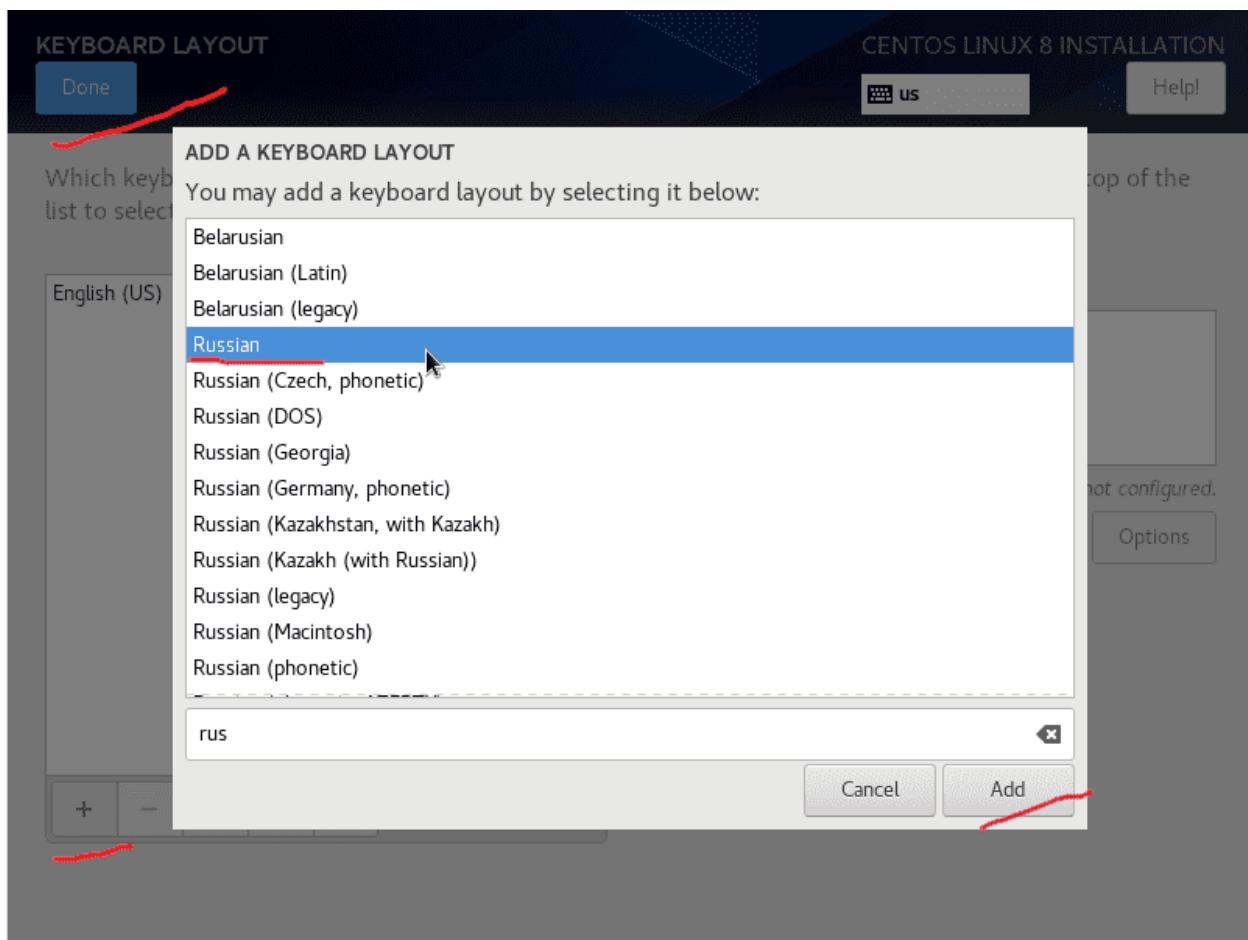


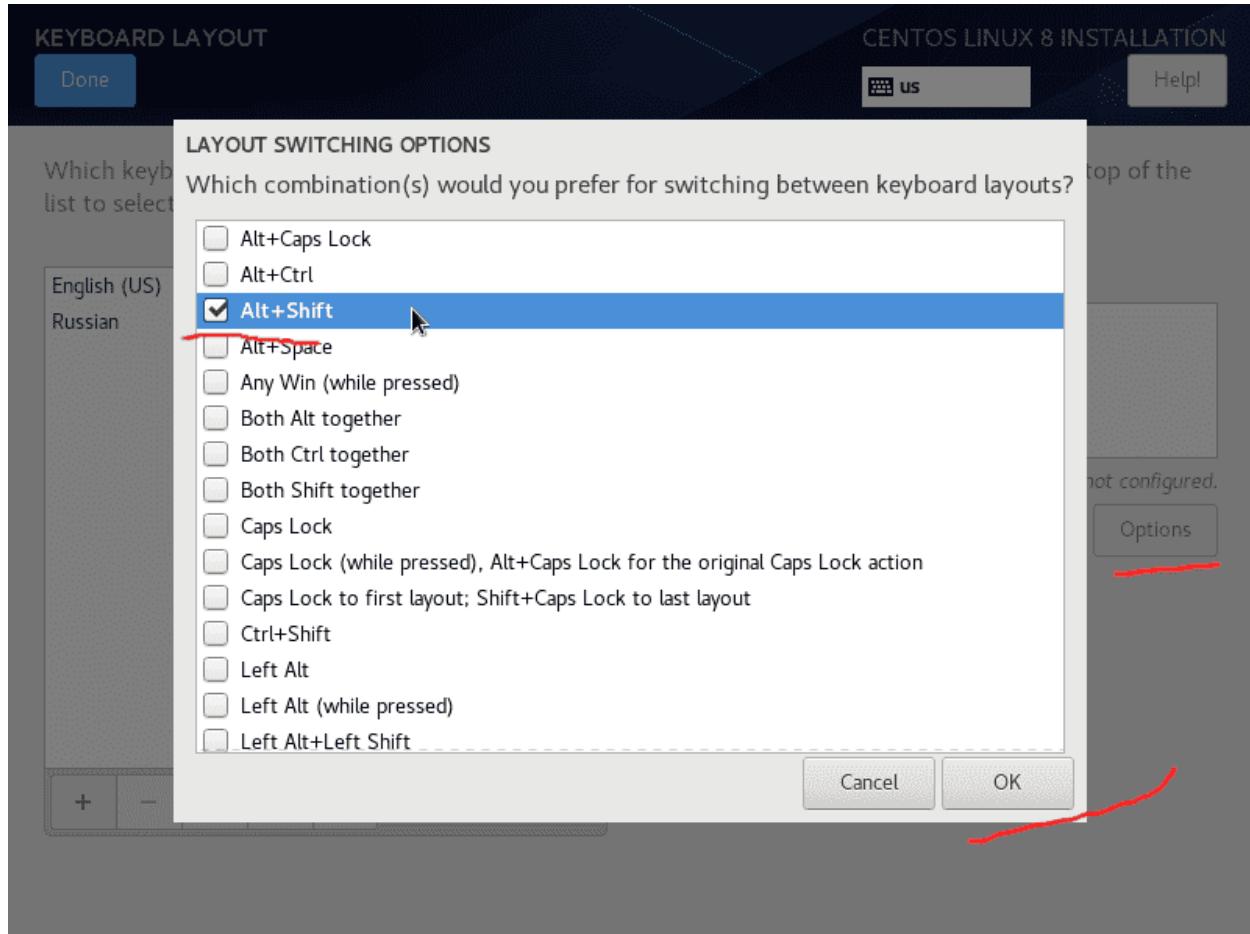
Если вы попытаетесь нажать на экран, VirtualBox предупредит вас, что виртуальная машина перехватит вашу мышку, из-за чего вы не сможете использовать её на хосте. Это не проблема, потому что когда вам понадобится, вы можете нажать правый Ctrl и освободить мышку, поэтому нажимаем Capture.



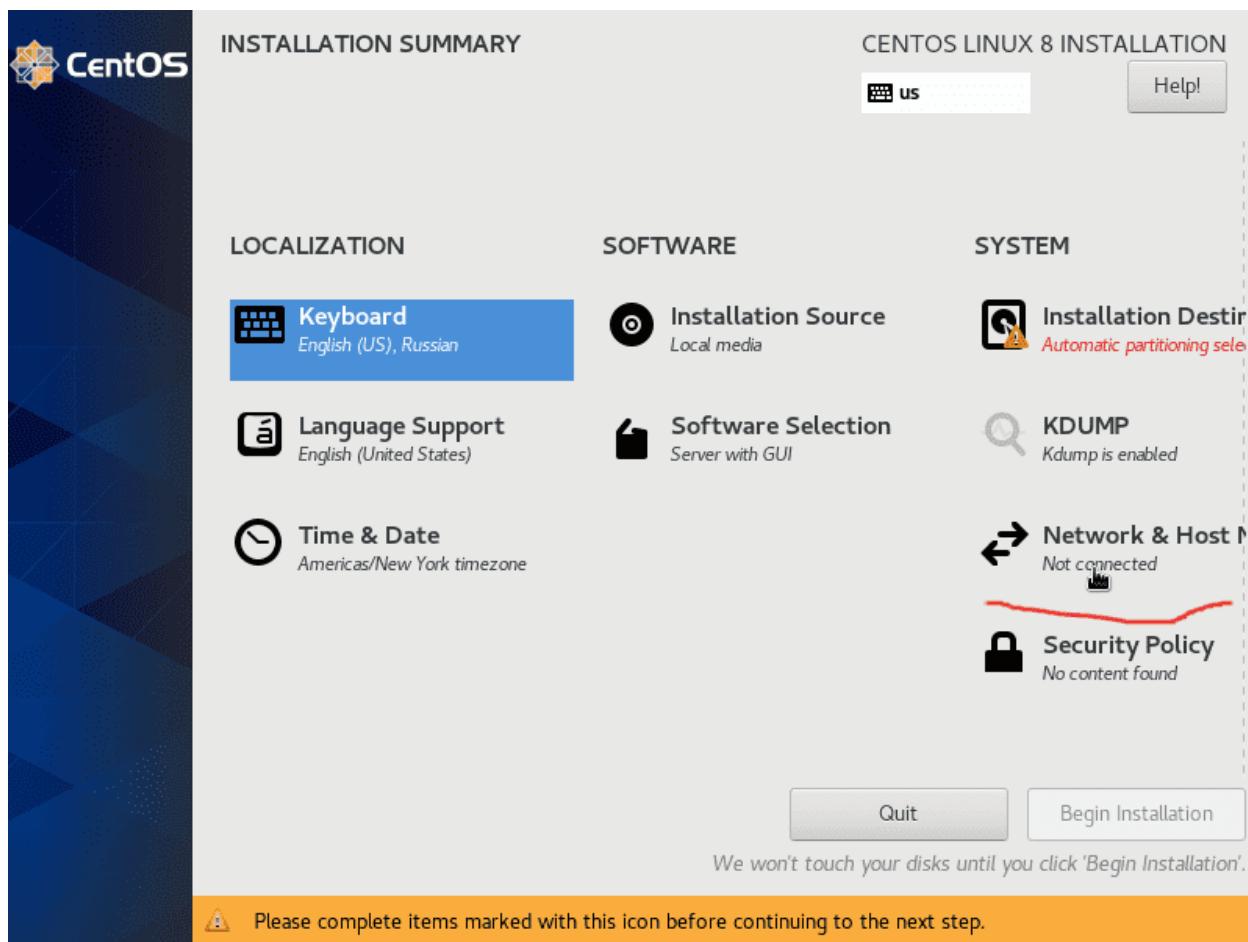
Когда дело касается языка установщика или операционной системы, рекомендую использовать Английский (US). Очень важно знать IT терминологию на английском языке, а также это позволит вам избежать дальнейших недопониманий. Теперь давайте пройдёмся по настройкам.

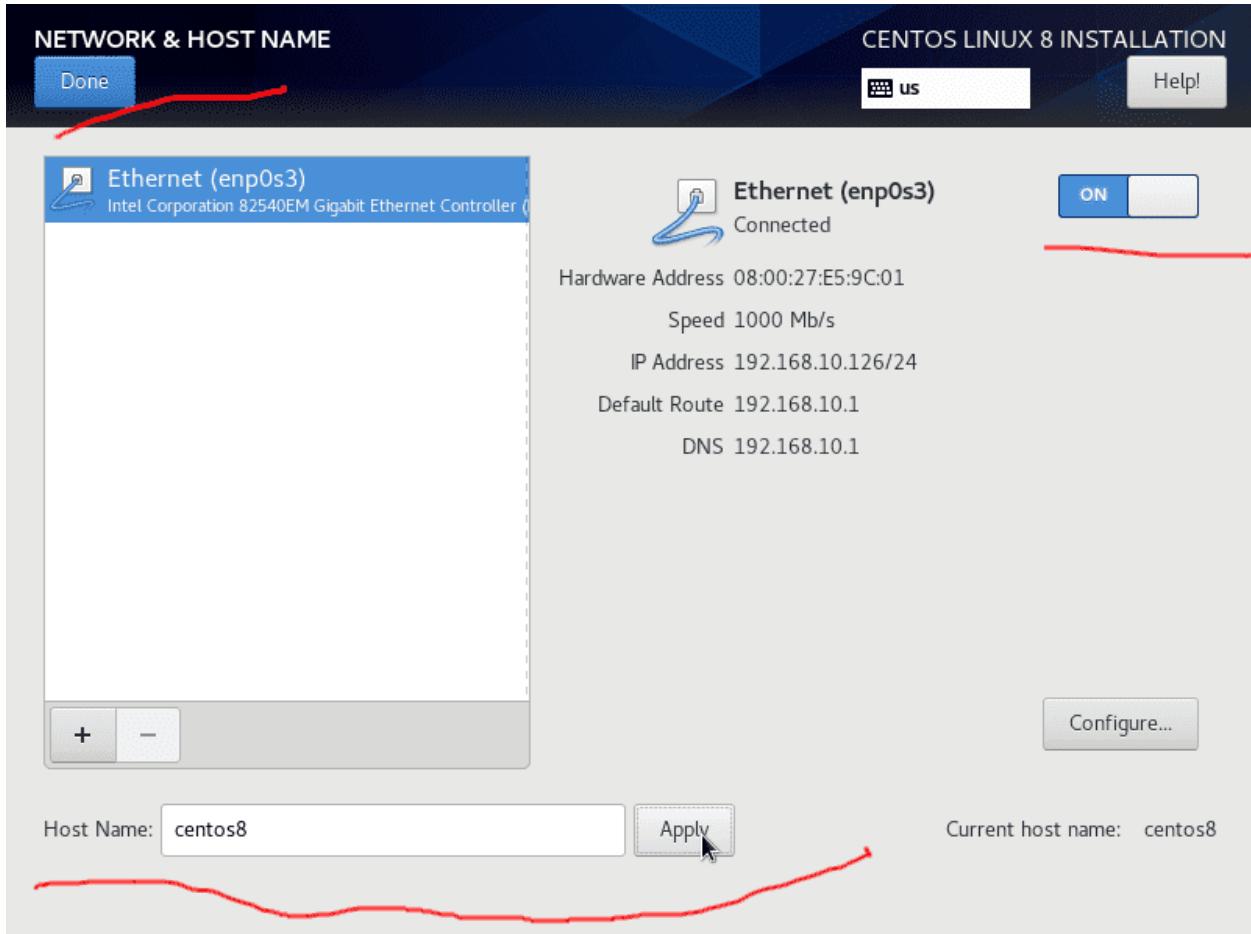




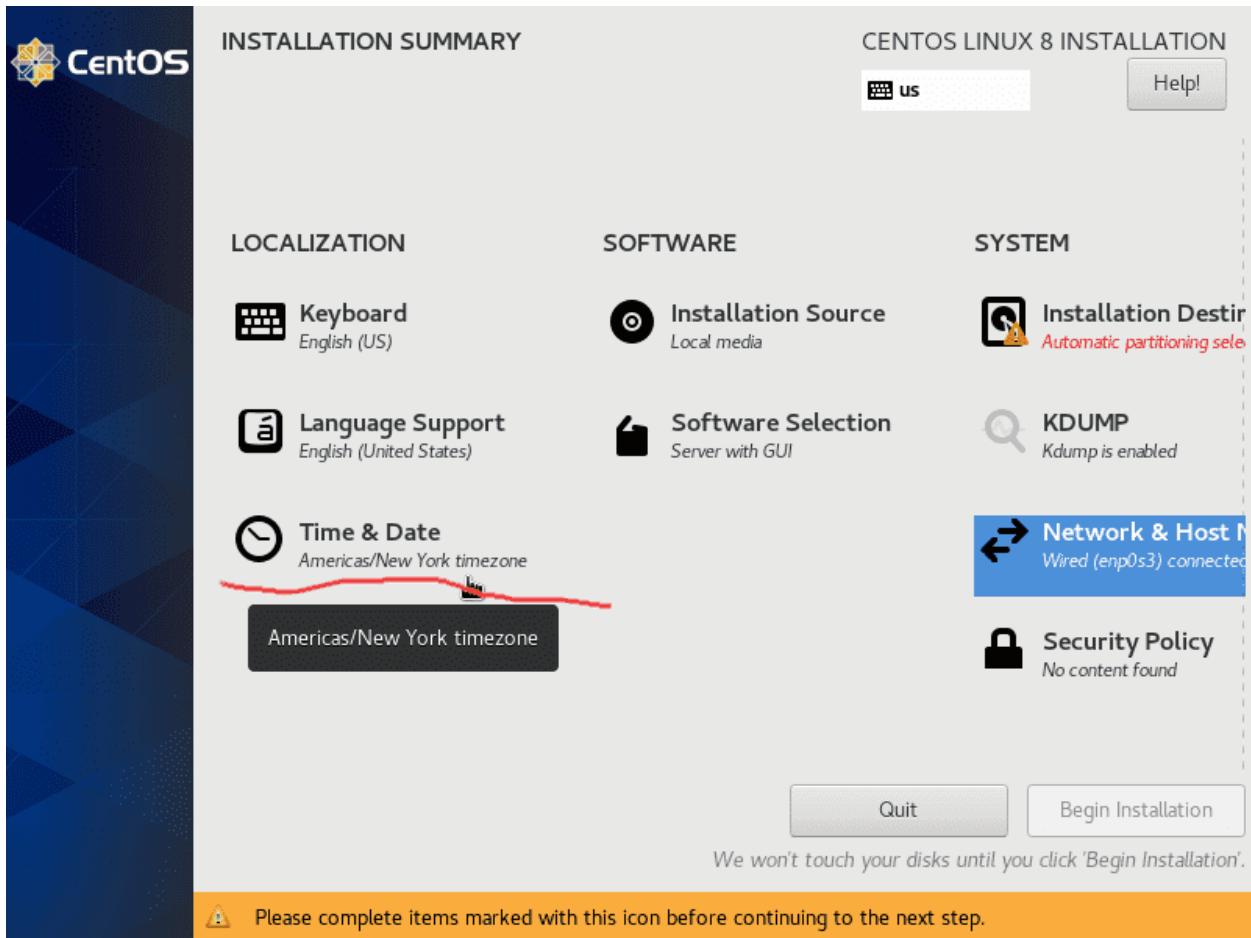


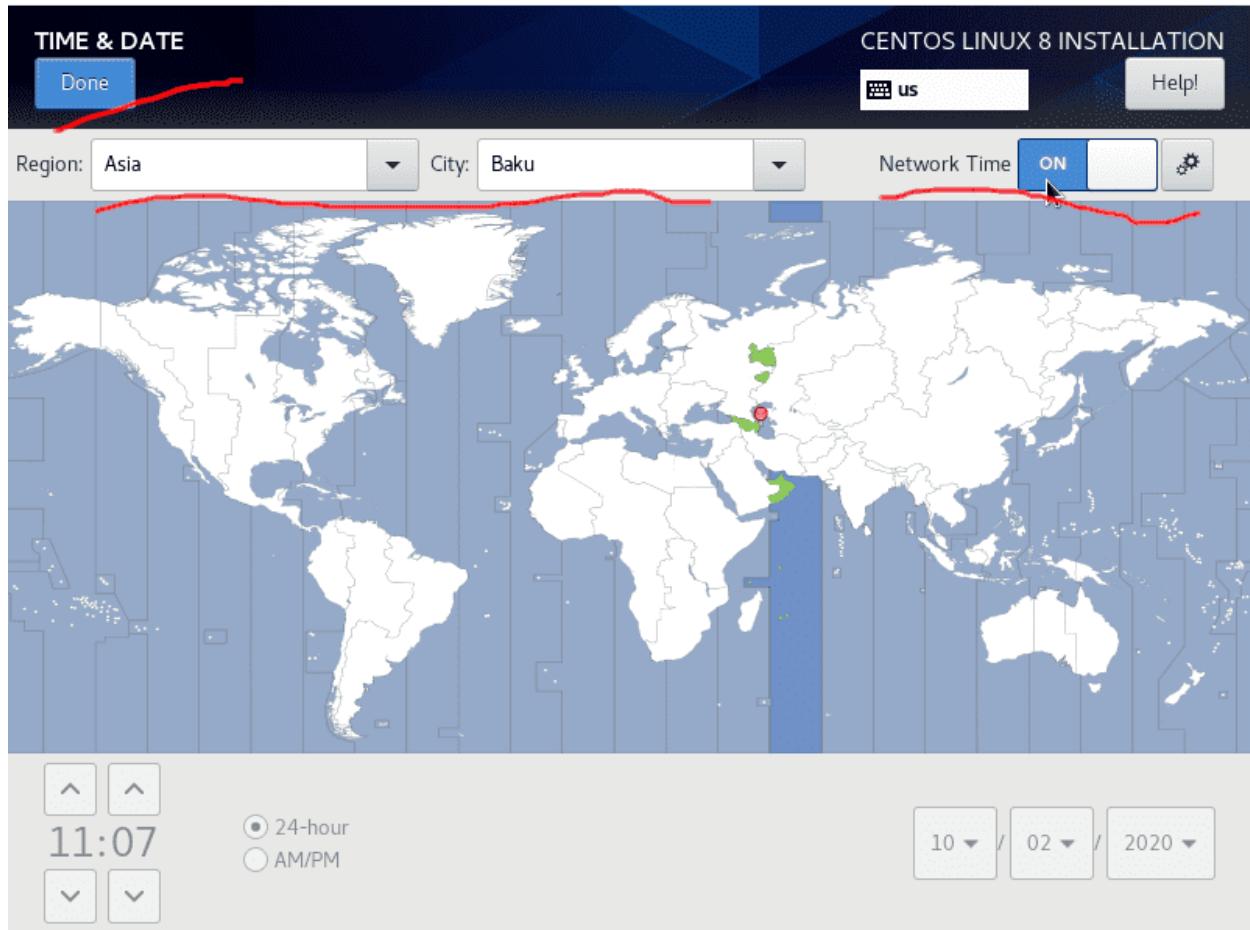
В меню Keyboard добавляем нужную раскладку и выбираем горячие клавиши для переключения раскладки.



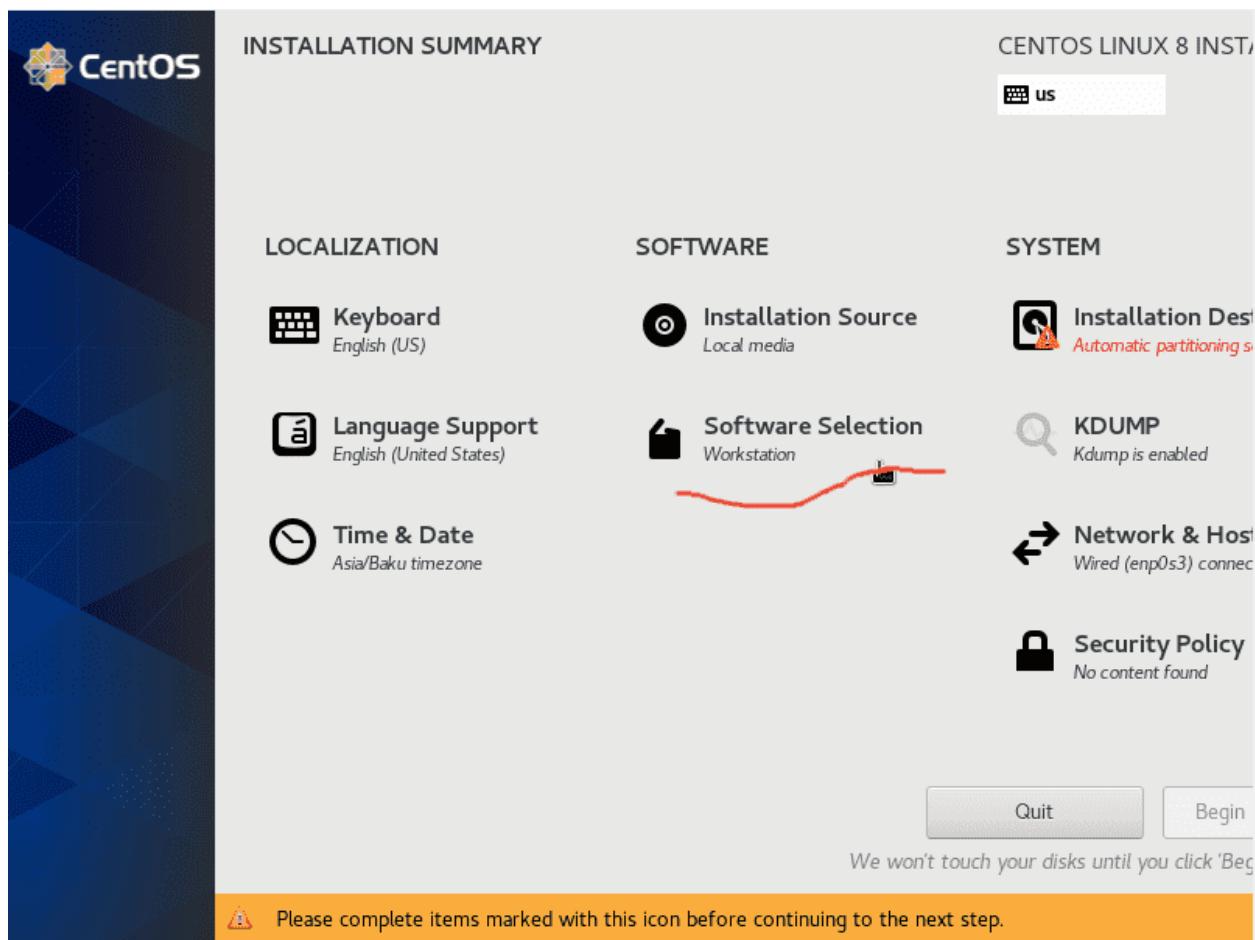


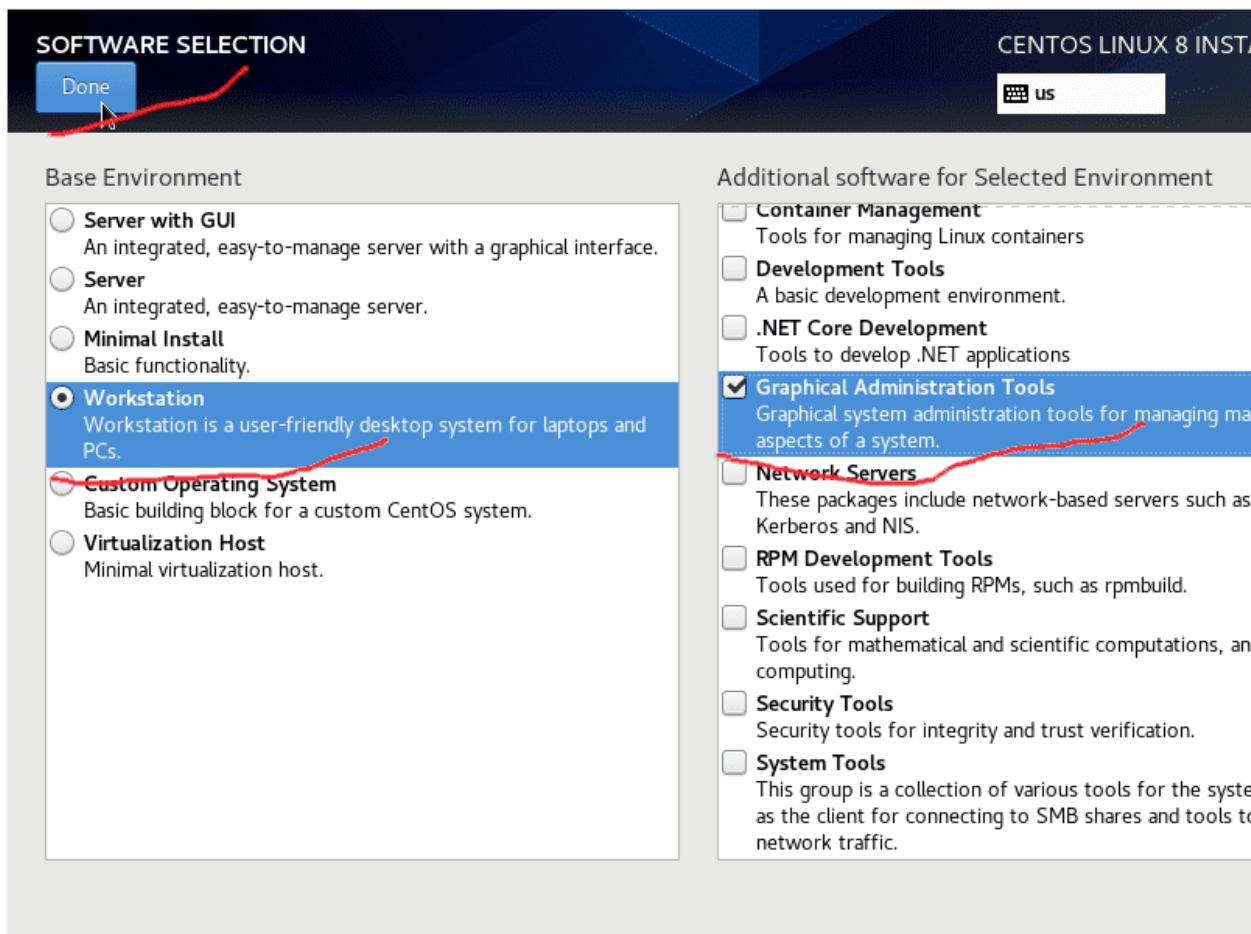
В меню Network & Host Name включаем сетевой адаптер, чтобы появилась сеть и указываем имя компьютеру.



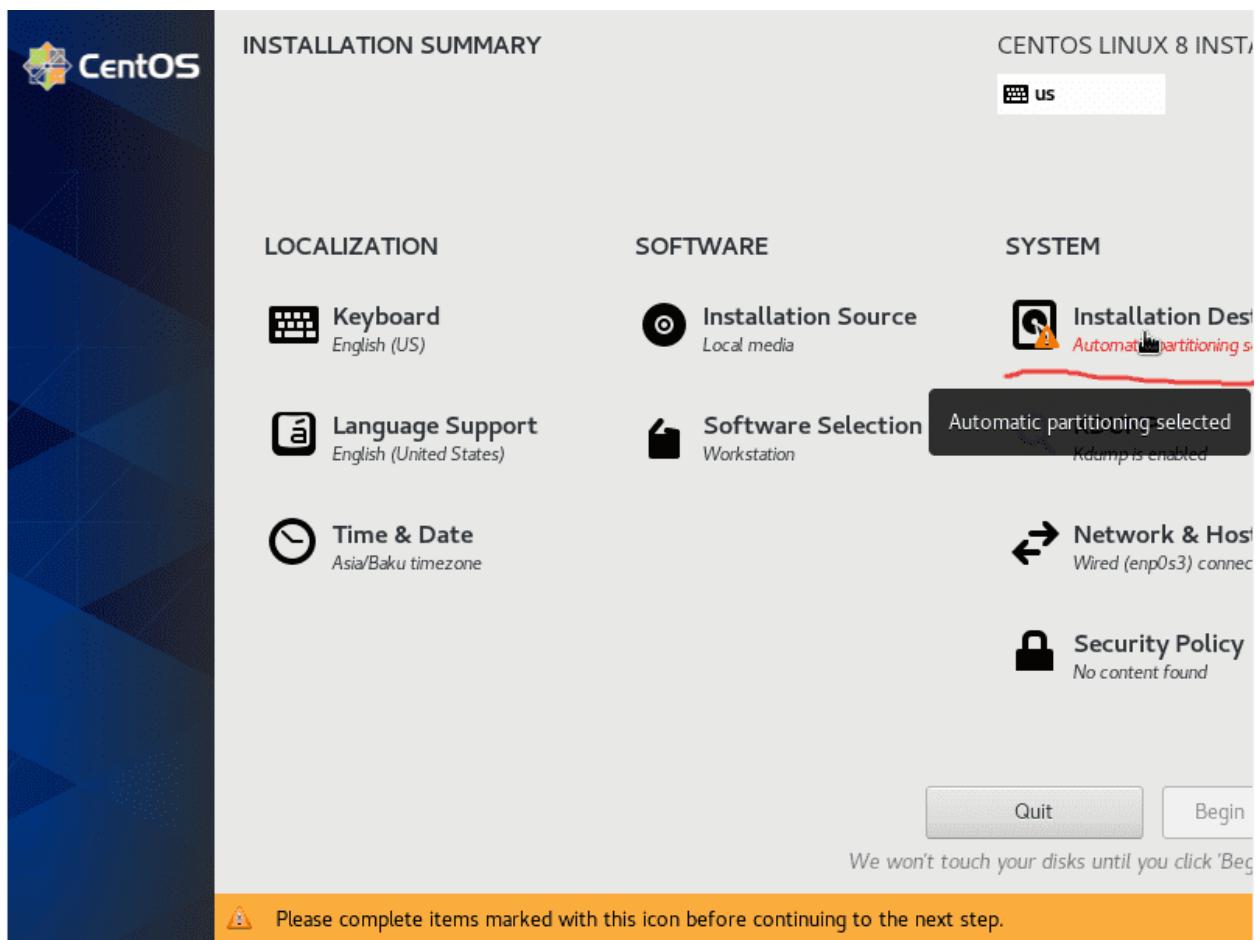


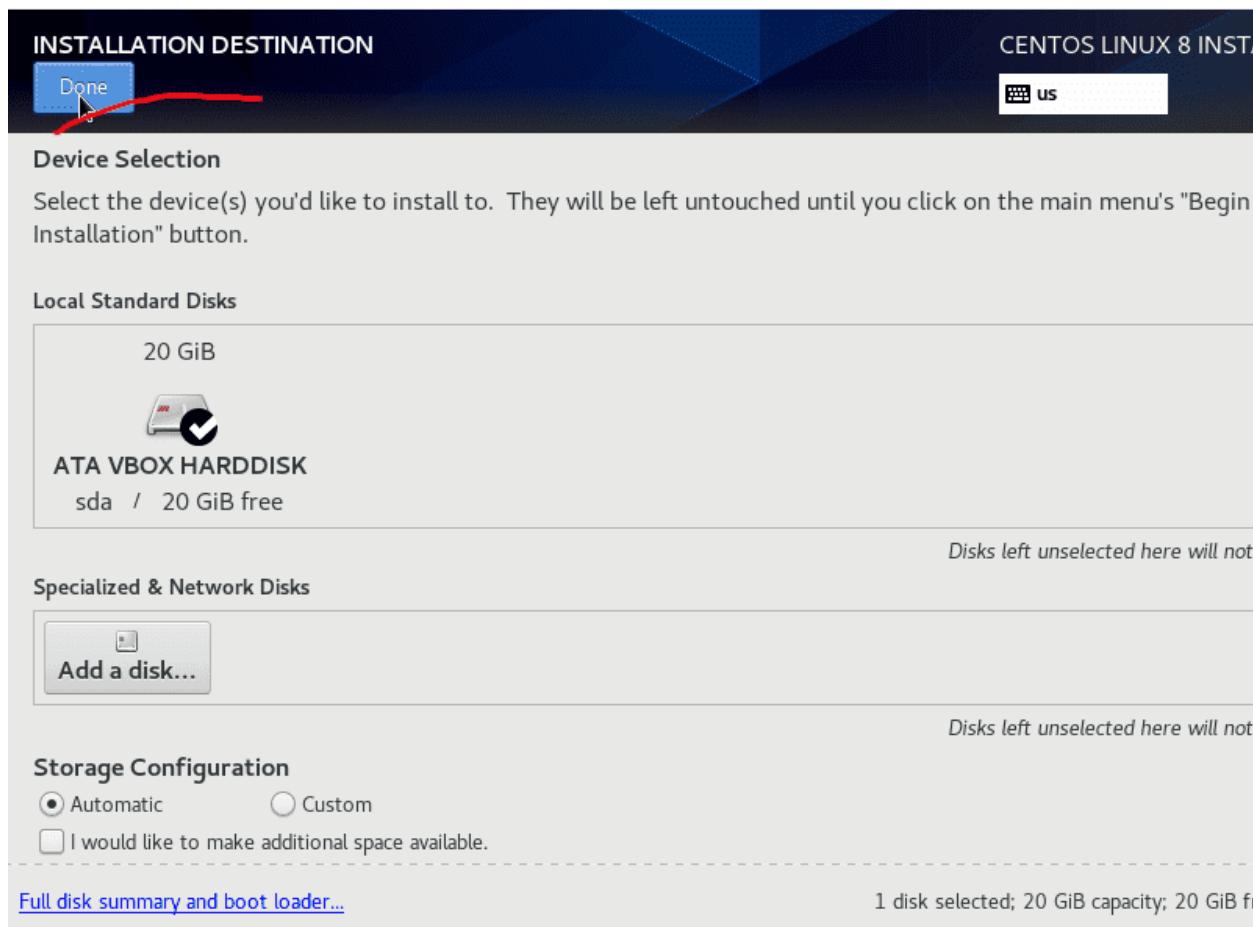
Во вкладке Time & Date убеждаемся, что включён Network Time и указываем правильную временную зону.



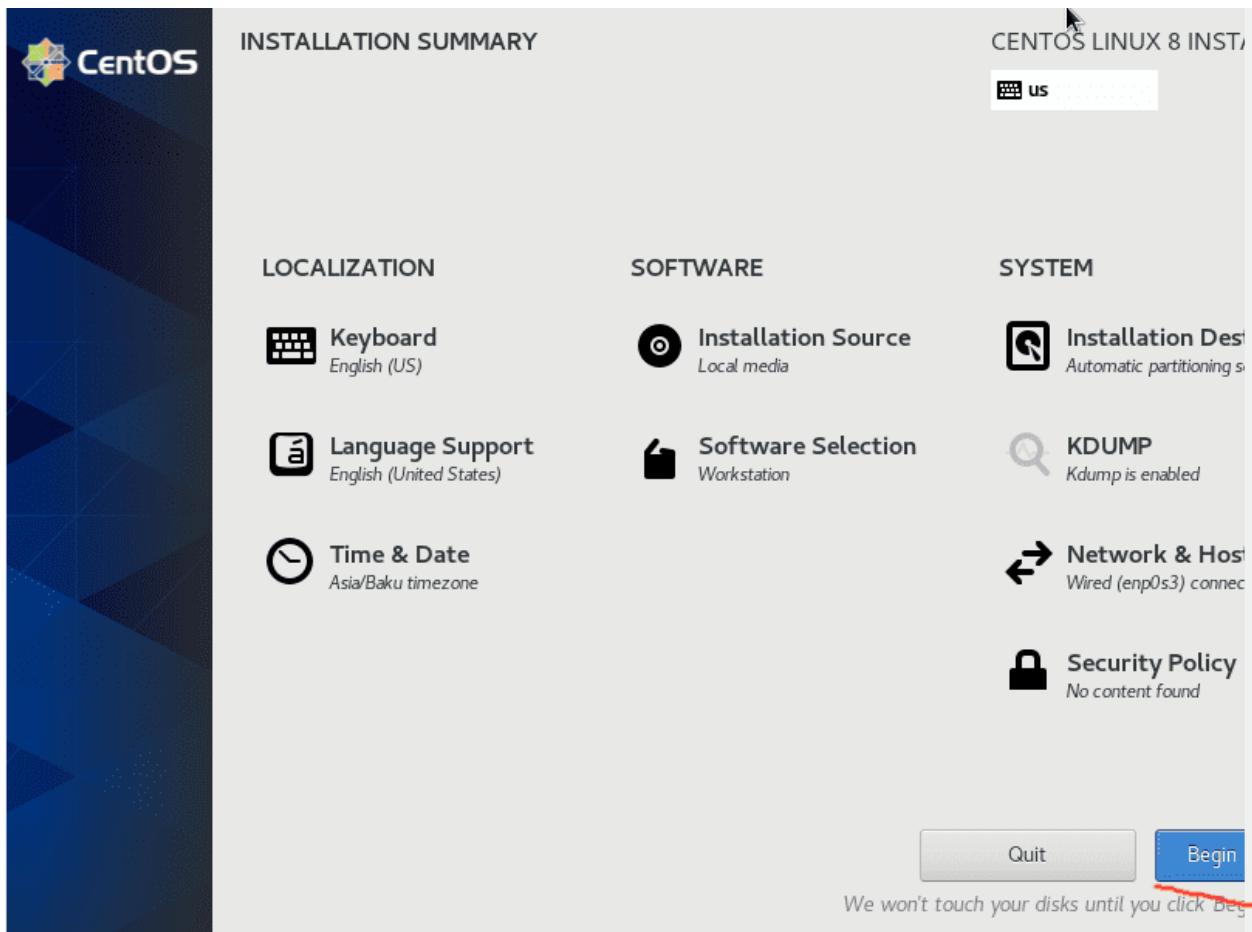


Во вкладке Software Selection выбираем Workstation и ставим галочки перед теми пакетами программ, которые мы хотели бы установить.



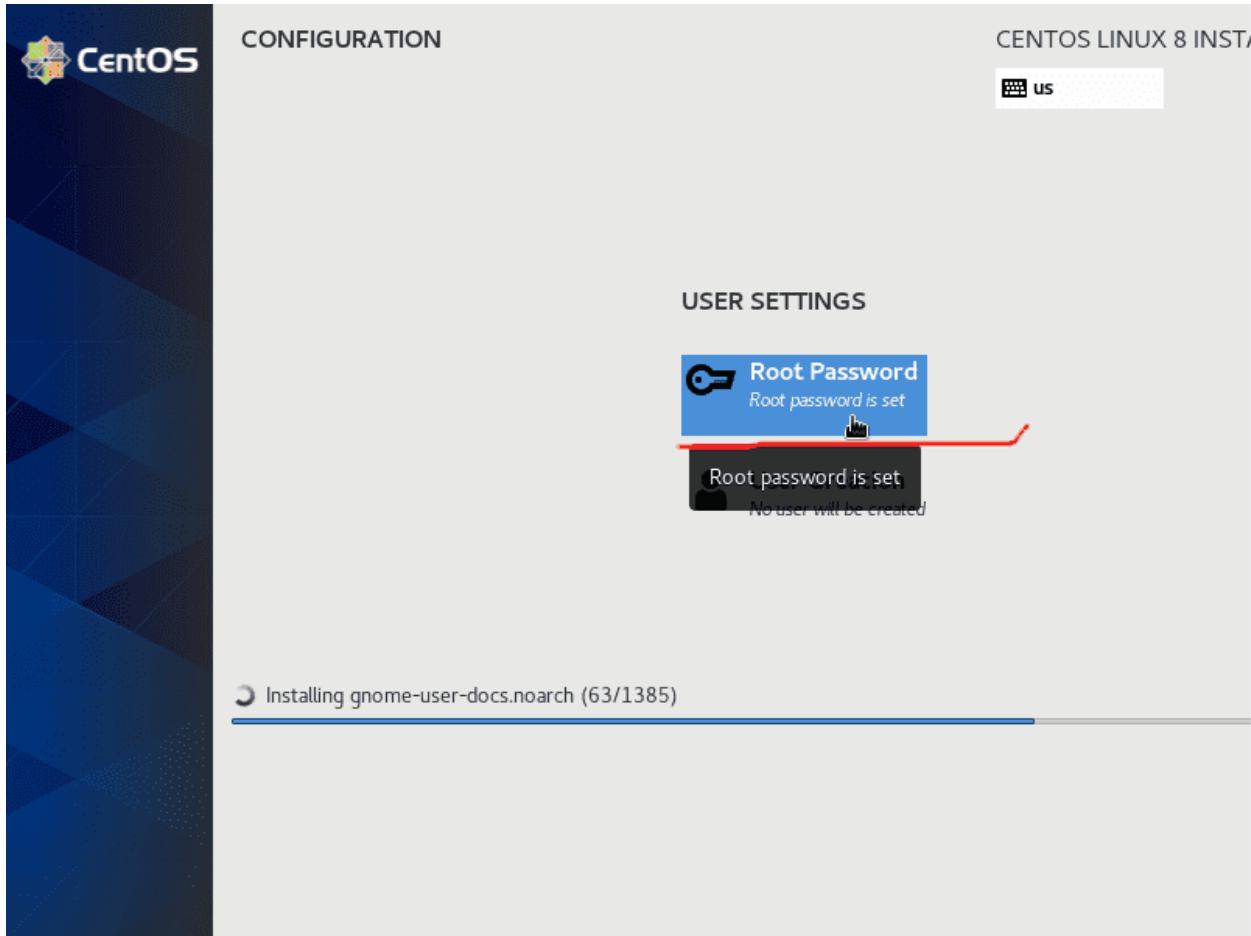


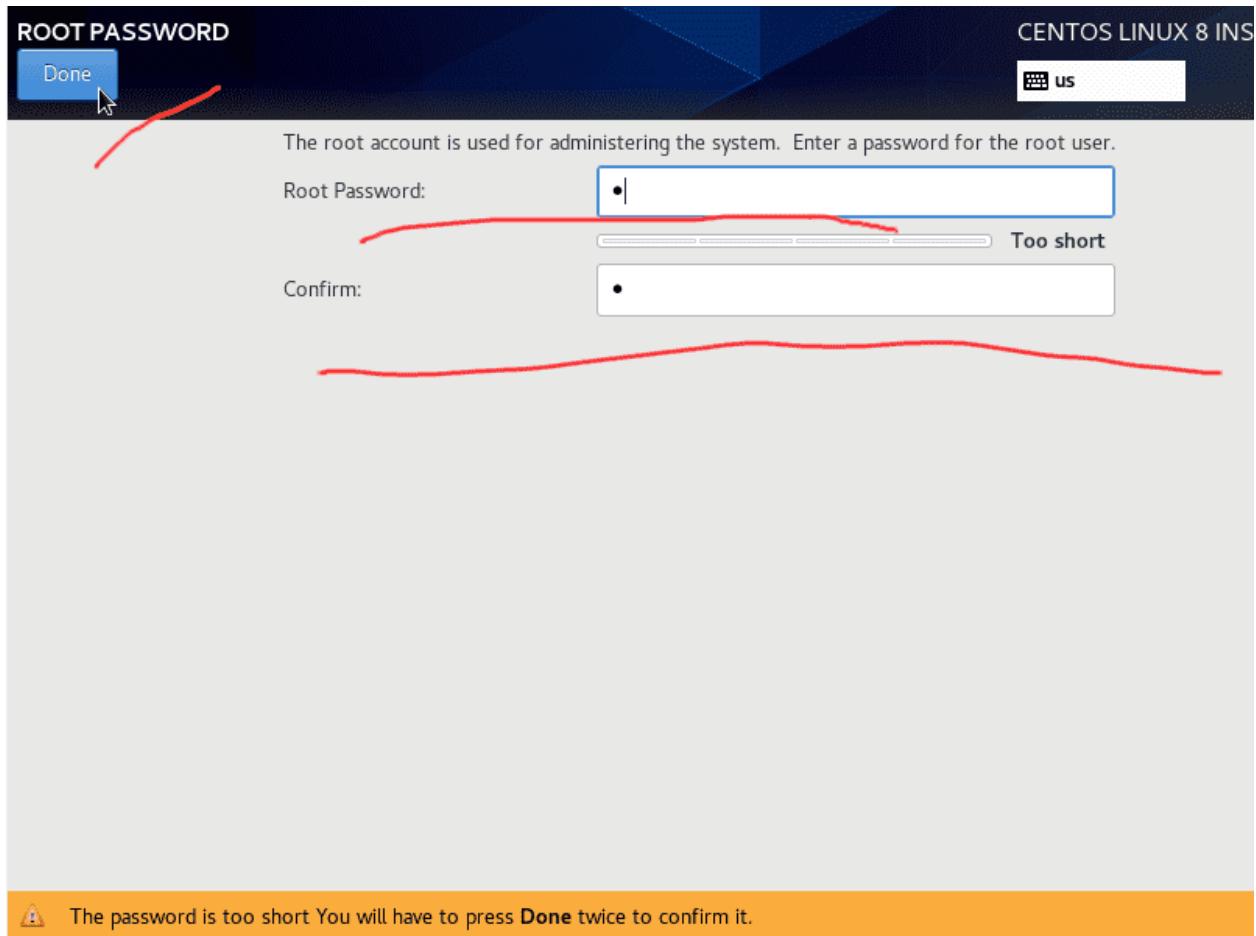
Заходим во вкладку Installation Destination и просто нажимаем Done, установщик сам разметит диск. Я не буду рассматривать вопрос разметки диска сейчас, так как это требует понимания многих тем, которые мы еще не прошли.



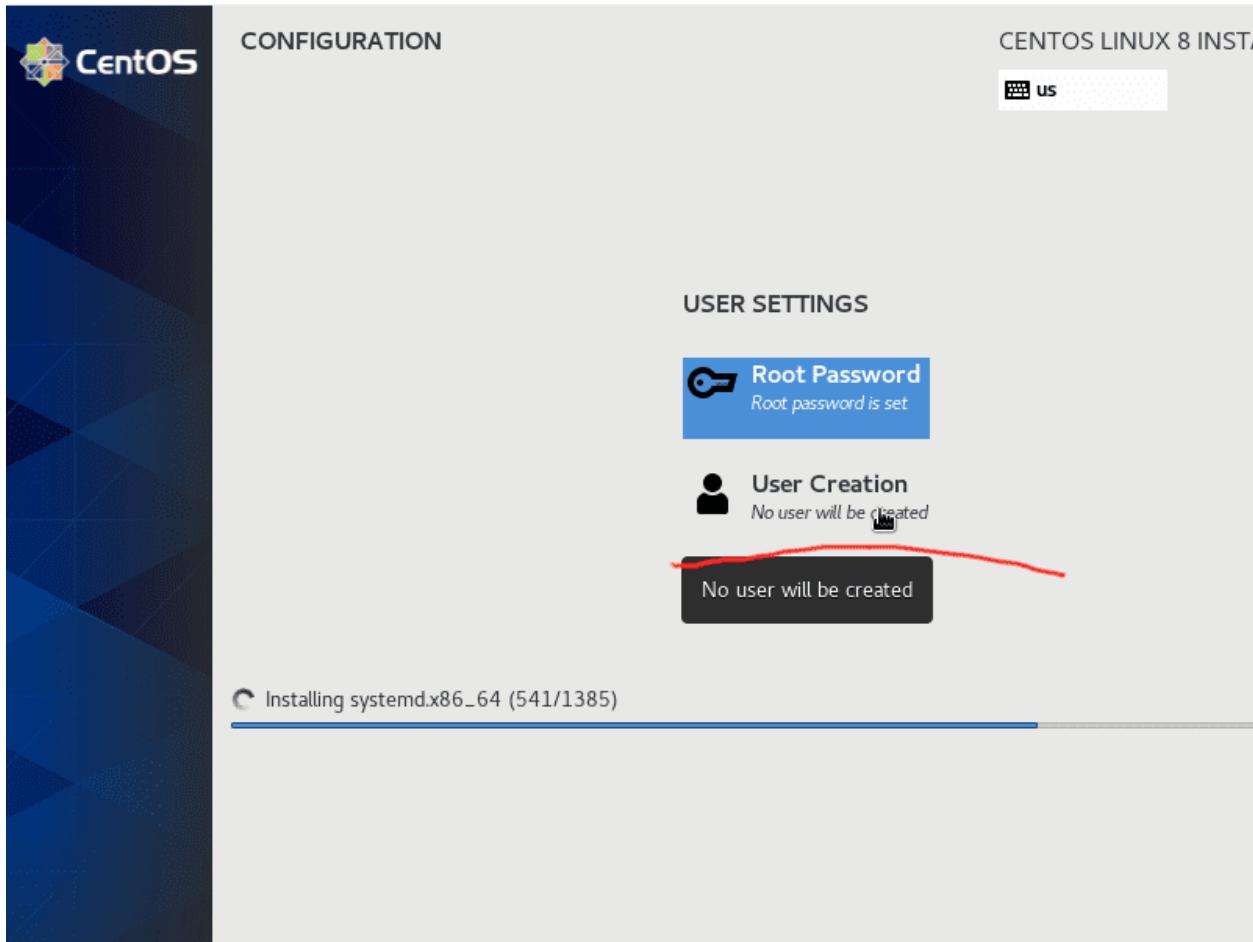
Нажимаем Begin Installation.

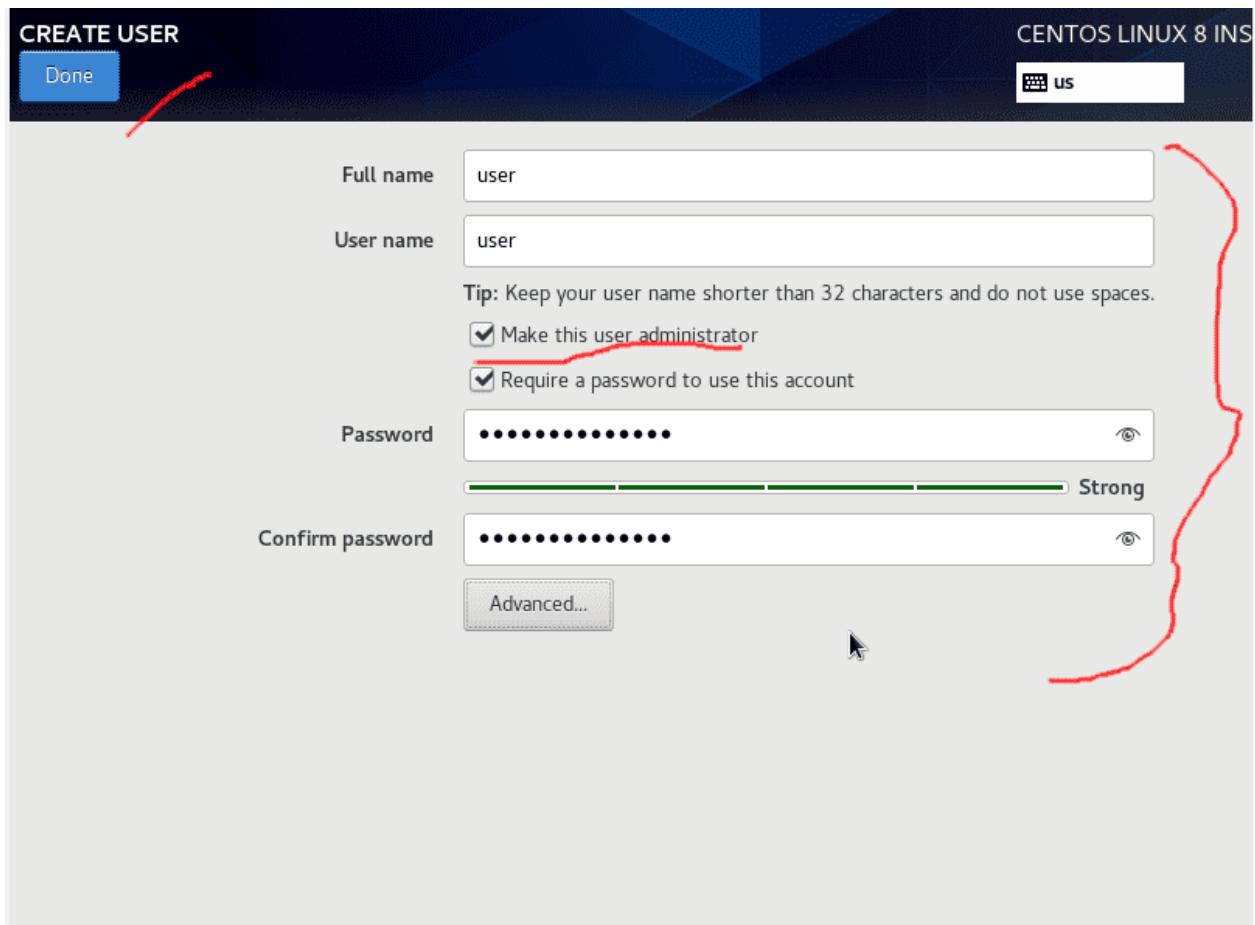
Пока идёт установка, мы должны задать пароль пользователю root и создать пользователя.



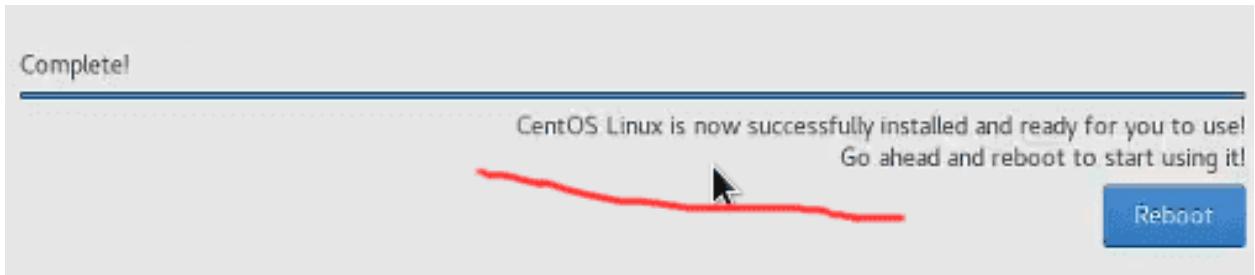


root – это пользователь, у которого есть все права на систему, он может делать с операционной системой практически всё.

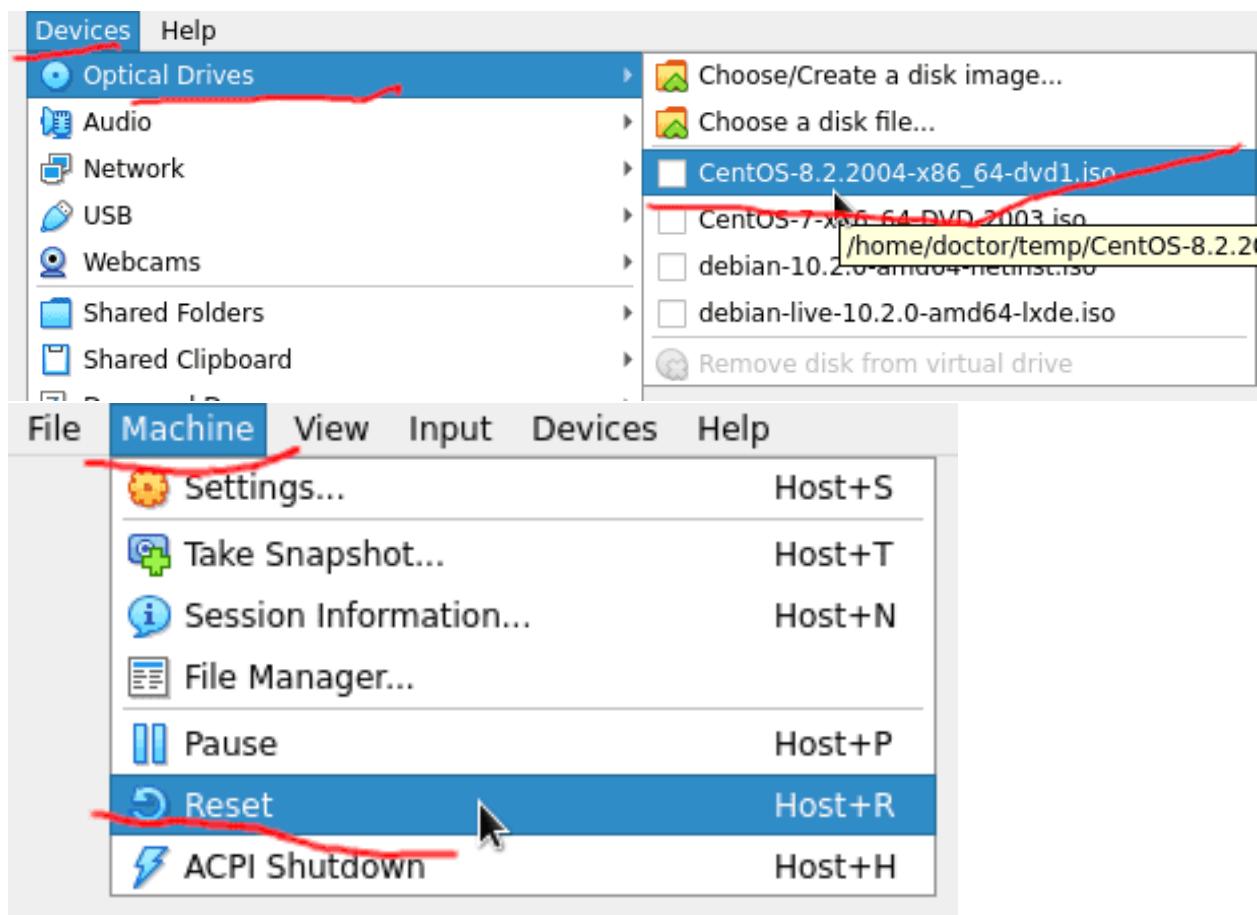




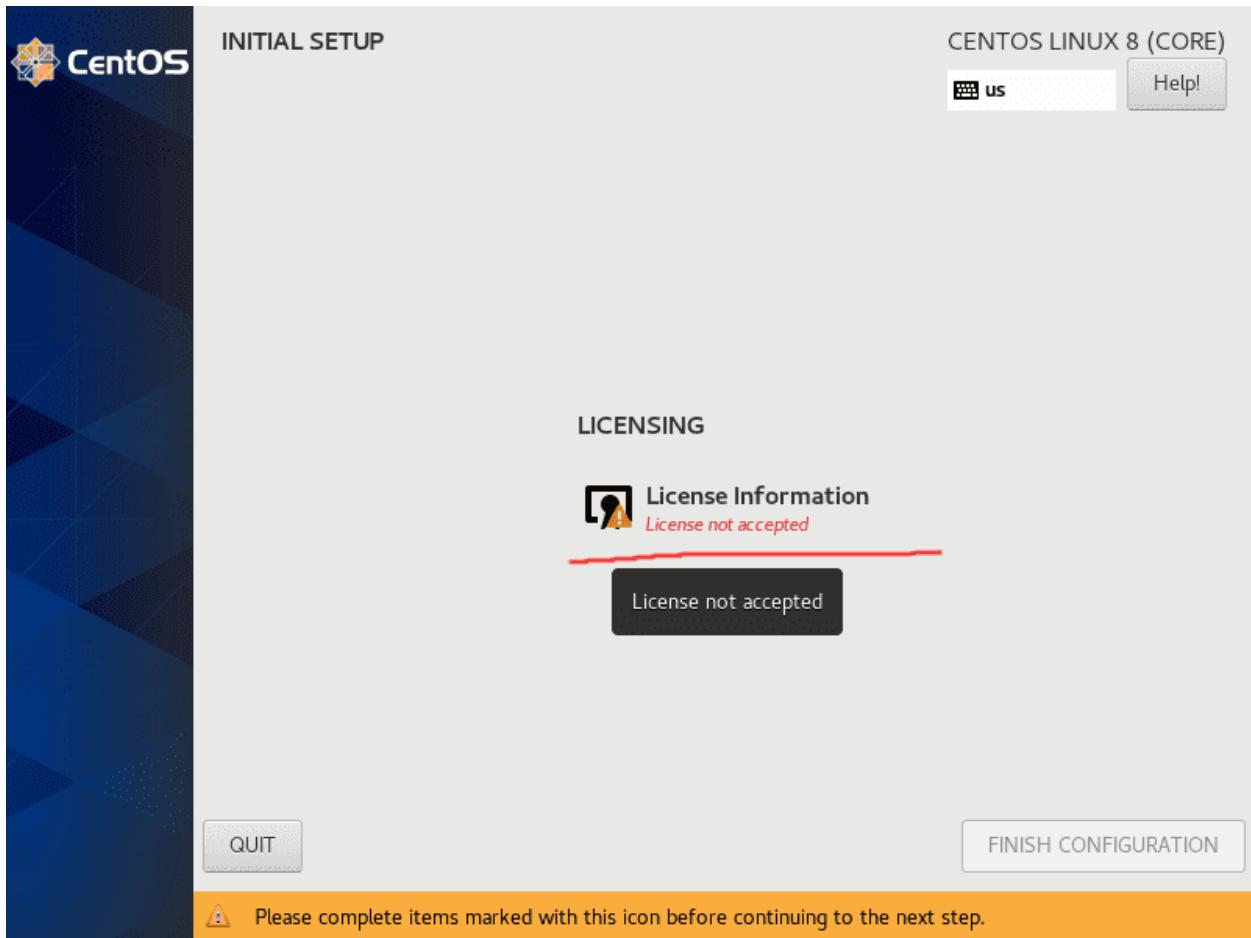
Дальше создаём пользователя и ставим галочку «Сделать этого пользователя администратором». И хотя у нас есть пользователь root с полными правами на систему, безопаснее использовать пользователя с административными правами.

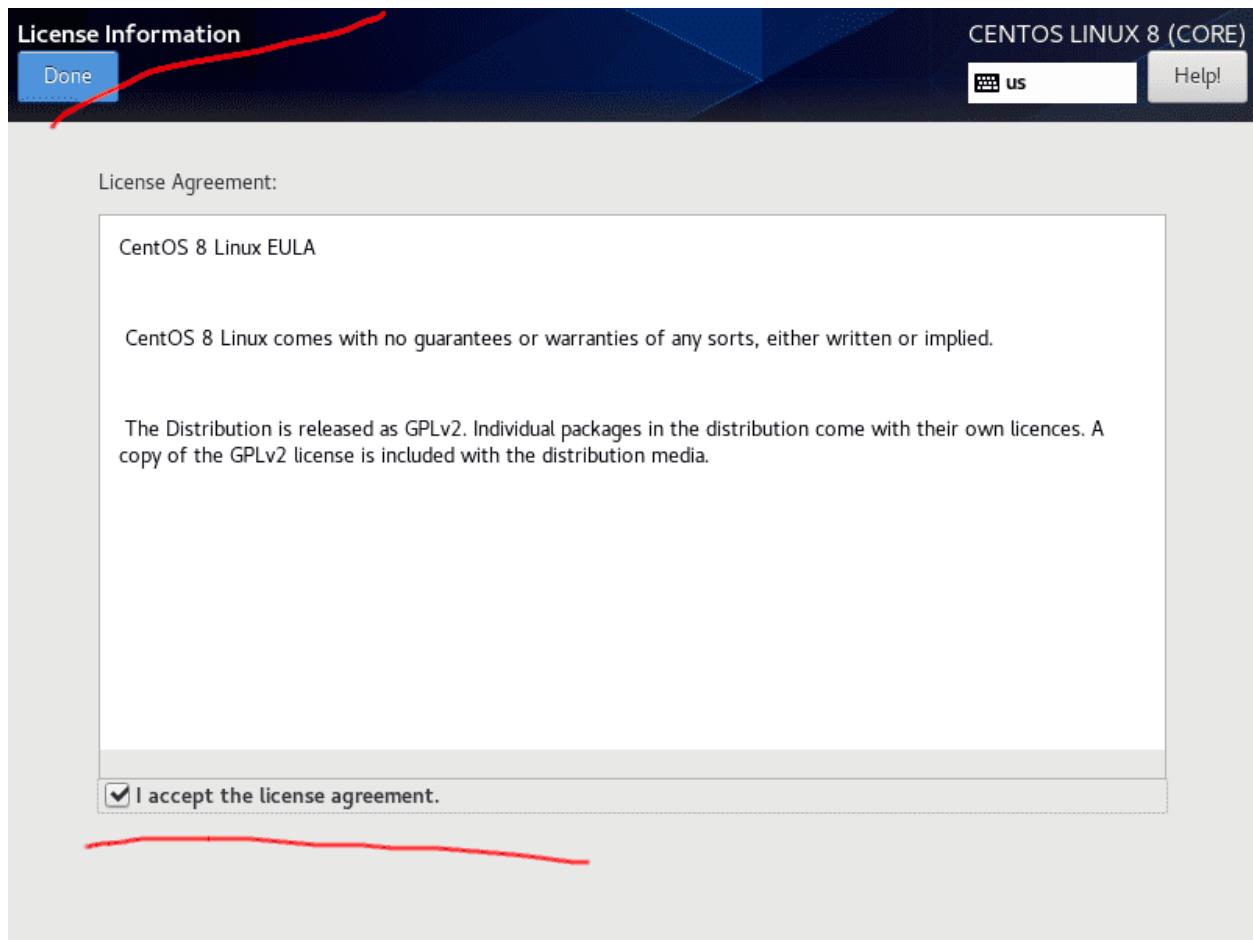


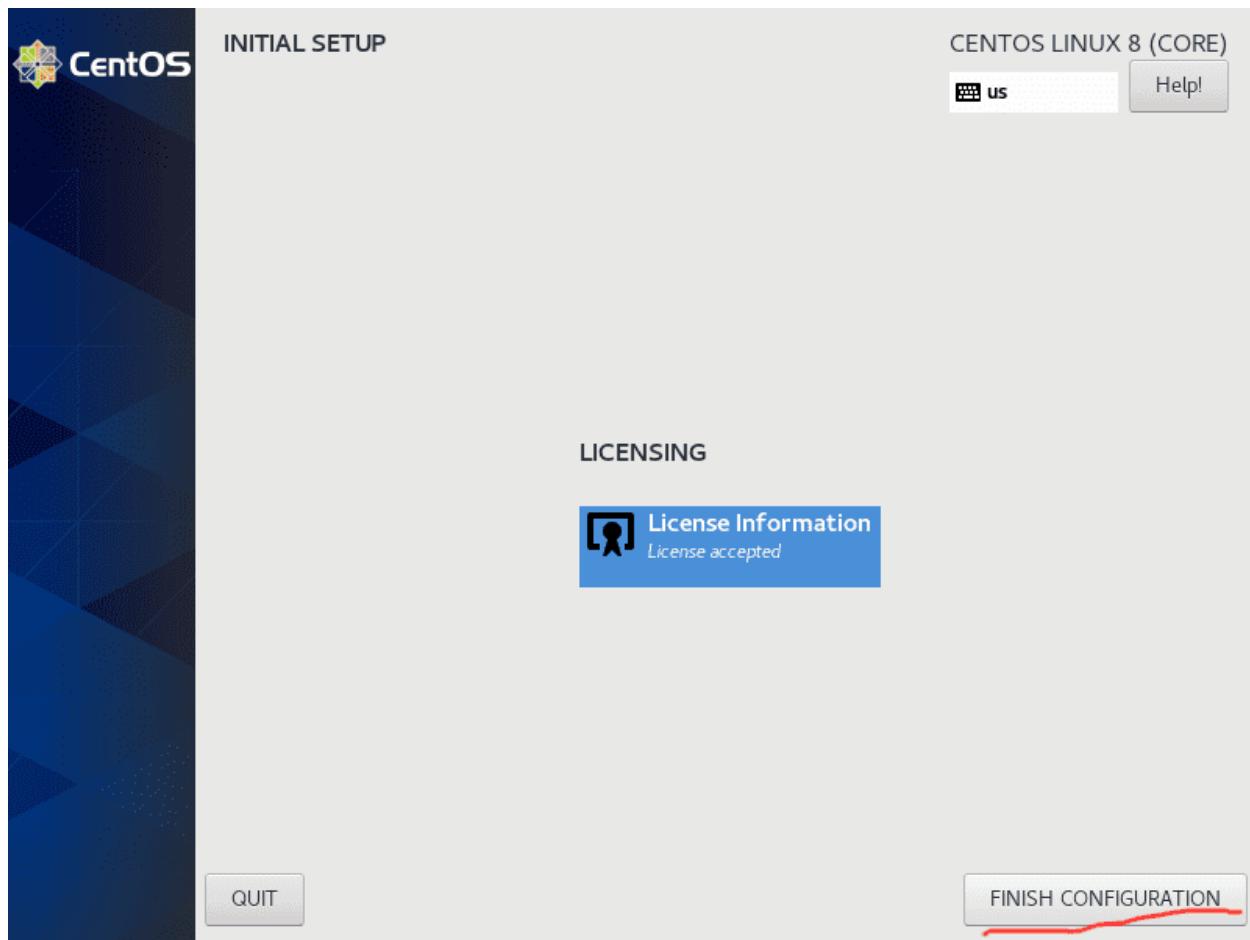
Затем дожидаемся конца установки системы.



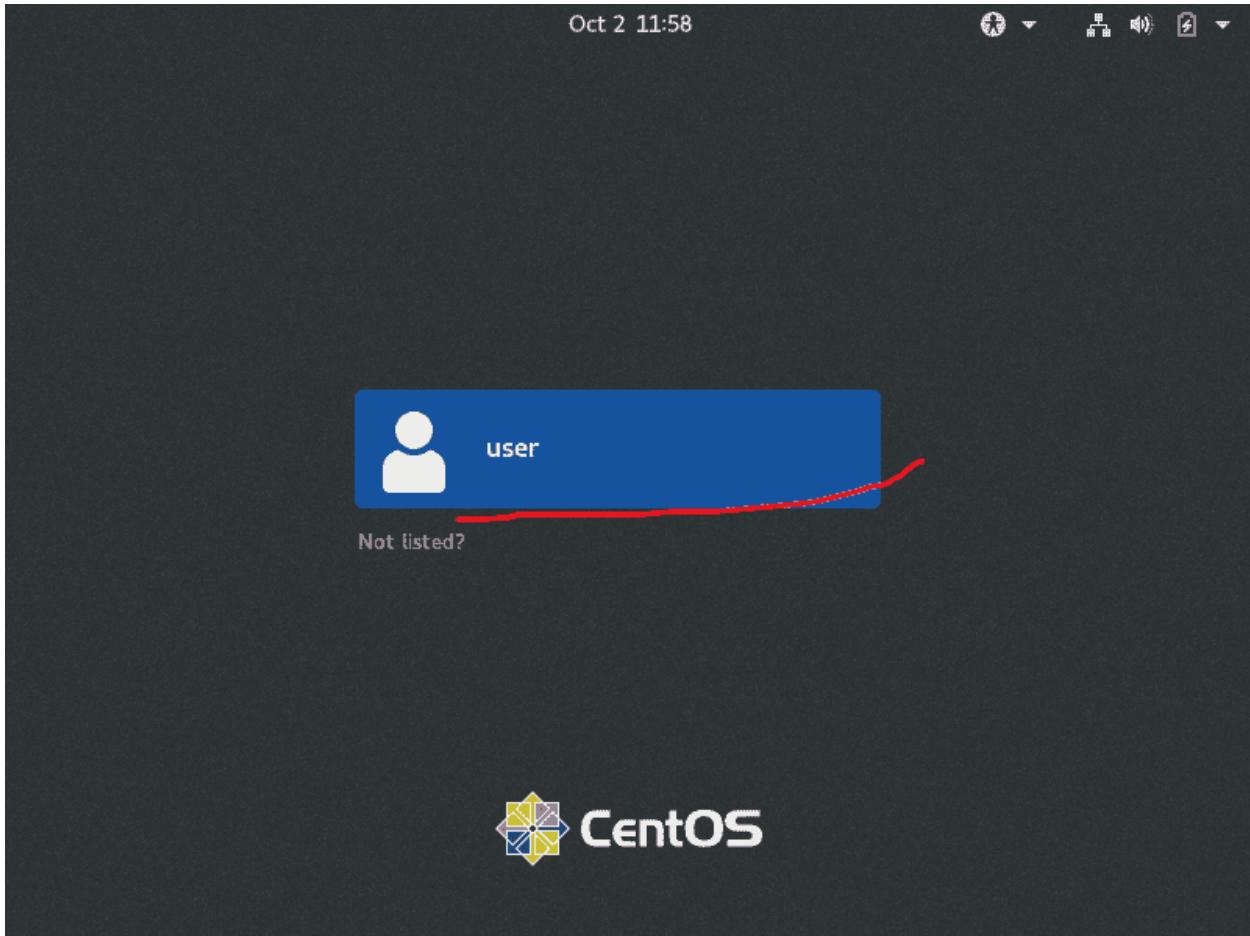
После установки следует отмонтировать ISO образ и перезагрузить виртуалку (Devices – Optical Drive – убираем галочку с установочного образа, затем Machine – Reset для перезагрузки).

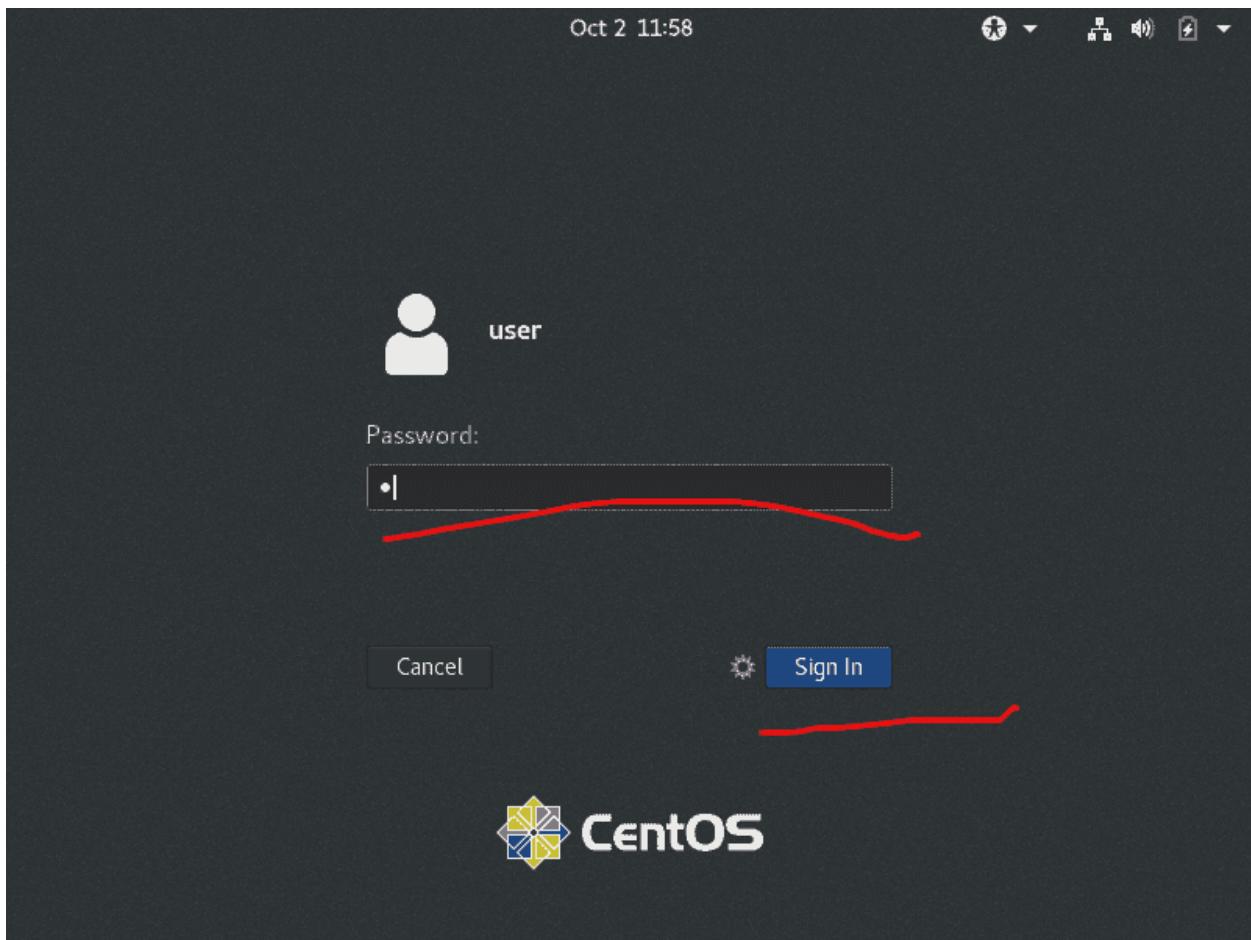


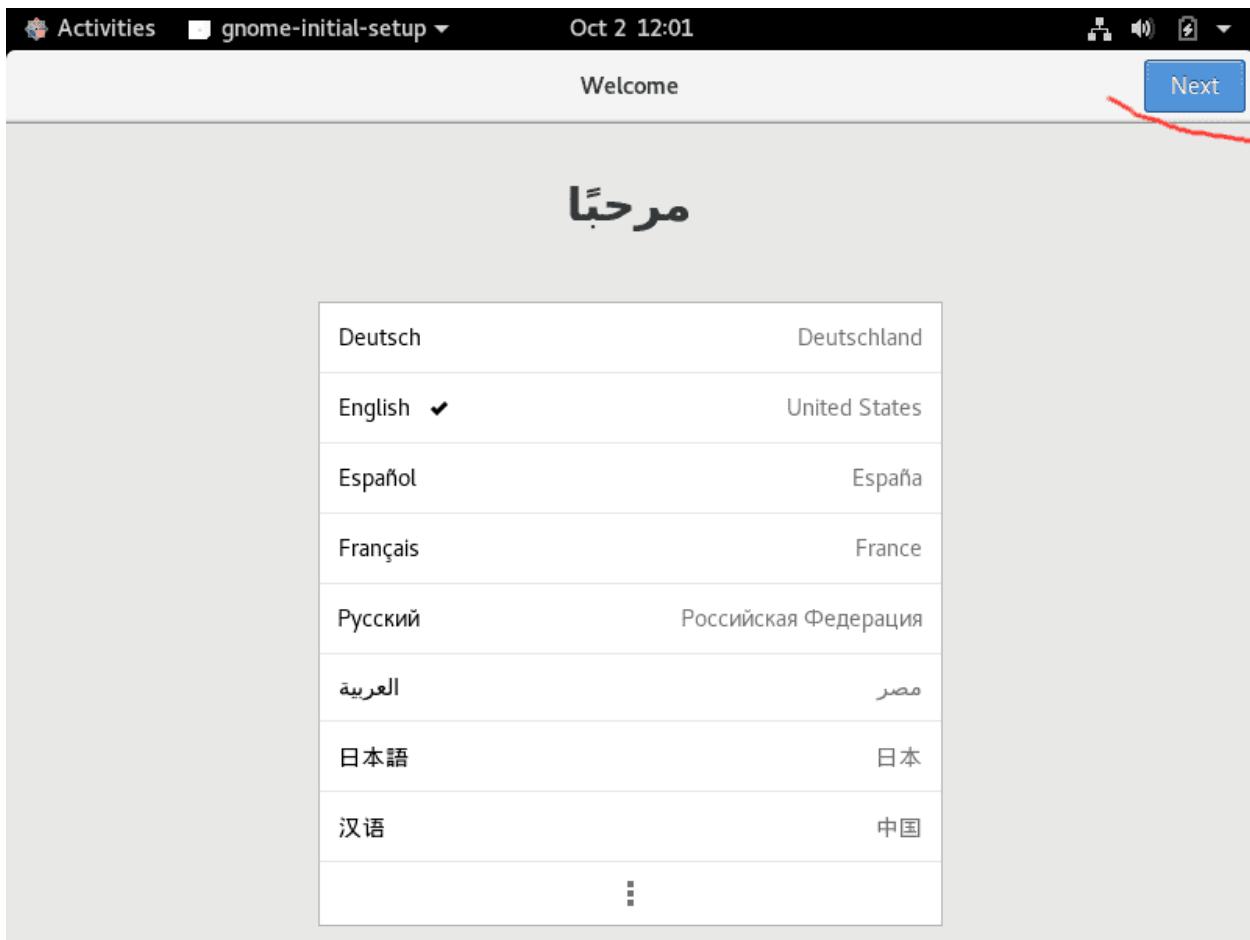


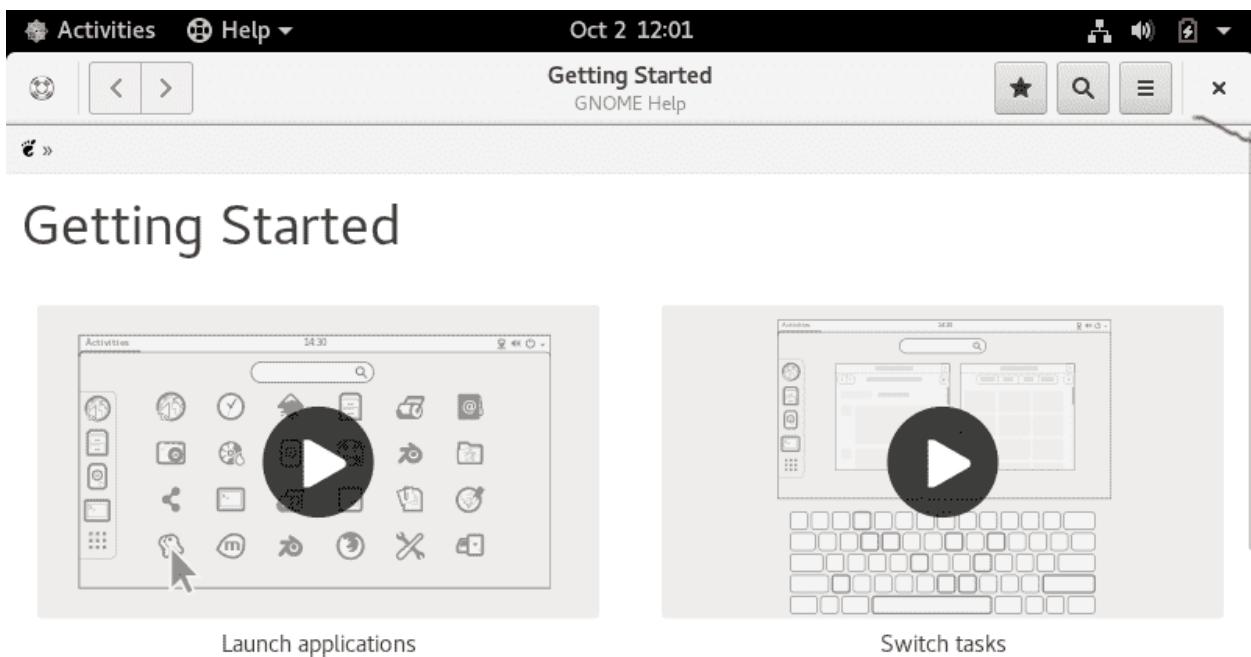


При включении принимаем лицензионное соглашение.









Затем логинимся пользователем user, после чего рабочее окружение, называемое Gnome, спросит у нас пару настроек и покажет окно с возможностями. Советую ознакомиться с ними.

Установка гостевых утилит



```
user@centos8:~$ sudo dnf update -y
We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:
#1) Respect the privacy of others.
#2) Think before you type.
#3) With great power comes great responsibility.

[sudo] password for user:
Last metadata expiration check: 0:09:17 ago on Fri 02 Oct 2020 11:55:42 AM +04.
Dependencies resolved.
=====
Package           Arch   Version        Repo      Size
=====
Installing:
kernel          x86_64  4.18.0-193.19.1.el8_2  BaseOS    2.8 M
kernel-core      x86_64  4.18.0-193.19.1.el8_2  BaseOS    28 M
kernel-modules   x86_64  4.18.0-193.19.1.el8_2  BaseOS   23 M
Upgrading:
NetworkManager   x86_64  1:1.22.8-5.el8_2   BaseOS    2.3 M
NetworkManager-adsl x86_64  1:1.22.8-5.el8_2   BaseOS   133 k
NetworkManager-bluetooth x86_64  1:1.22.8-5.el8_2   BaseOS   158 k
NetworkManager-libnm x86_64  1:1.22.8-5.el8_2   BaseOS    1.7 M
```

The terminal window shows the command `sudo dnf update -y` being run. The output includes a warning about the usual lecture from the local System Administrator, followed by a list of packages being installed or upgraded, and finally a table of package details.

Дальше стоит установить обновления – для этого нажимаем Activities – Terminal и в терминале вводим команду (`sudo dnf update -y`), после чего терминал спросит пароль пользователя. Введённый пароль не будет отображаться даже звёздочками, поэтому просто введите пароль и нажмите Enter. Эта команда обновит нашу систему.

```
sudo dnf update -y
```

Activities Terminal Oct 2 12:28

```
user@centos8:~  
File Edit View Search Terminal Help  
selinux-policy-3.14.3-41.el8_2.6.noarch  
selinux-policy-targeted-3.14.3-41.el8_2.6.noarch  
sos-3.8-7.el8_2.noarch  
subscription-manager-1.26.20-1.el8_2.x86_64  
subscription-manager-rhsm-certificates-1.26.20-1.el8_2.x86_64  
systemd-239-31.el8_2.2.x86_64  
systemd-container-239-31.el8_2.2.x86_64  
systemd-libs-239-31.el8_2.2.x86_64  
systemd-pam-239-31.el8_2.2.x86_64  
systemd-udev-239-31.el8_2.2.x86_64  
teamd-1.29-1.el8_2.2.x86_64  
tzdata-2020a-1.el8.noarch  
unbound-libs-1.7.3-11.el8_2.x86_64  
yum-4.2.17-7.el8_2.noarch  
zlib-1.2.11-16.el8_2.x86_64  
  
Installed:  
grub2-tools-efi-1:2.02-87.el8_2.x86_64  
kernel-4.18.0-193.19.1.el8_2.x86_64  
kernel-core-4.18.0-193.19.1.el8_2.x86_64  
kernel-modules-4.18.0-193.19.1.el8_2.x86_64  
  
Complete!  
[user@centos8 ~]$
```

Иногда, при сбоях с сетью, команда может выдать ошибку, поэтому стоит прочитать последние строчки. Если вы видите там error или что-то в этом духе, то попробуйте заново запустить предыдущую команду. Если Complete – значит всё нормально.

The screenshot shows a terminal window titled "user@centos8:~". The terminal displays the following output:

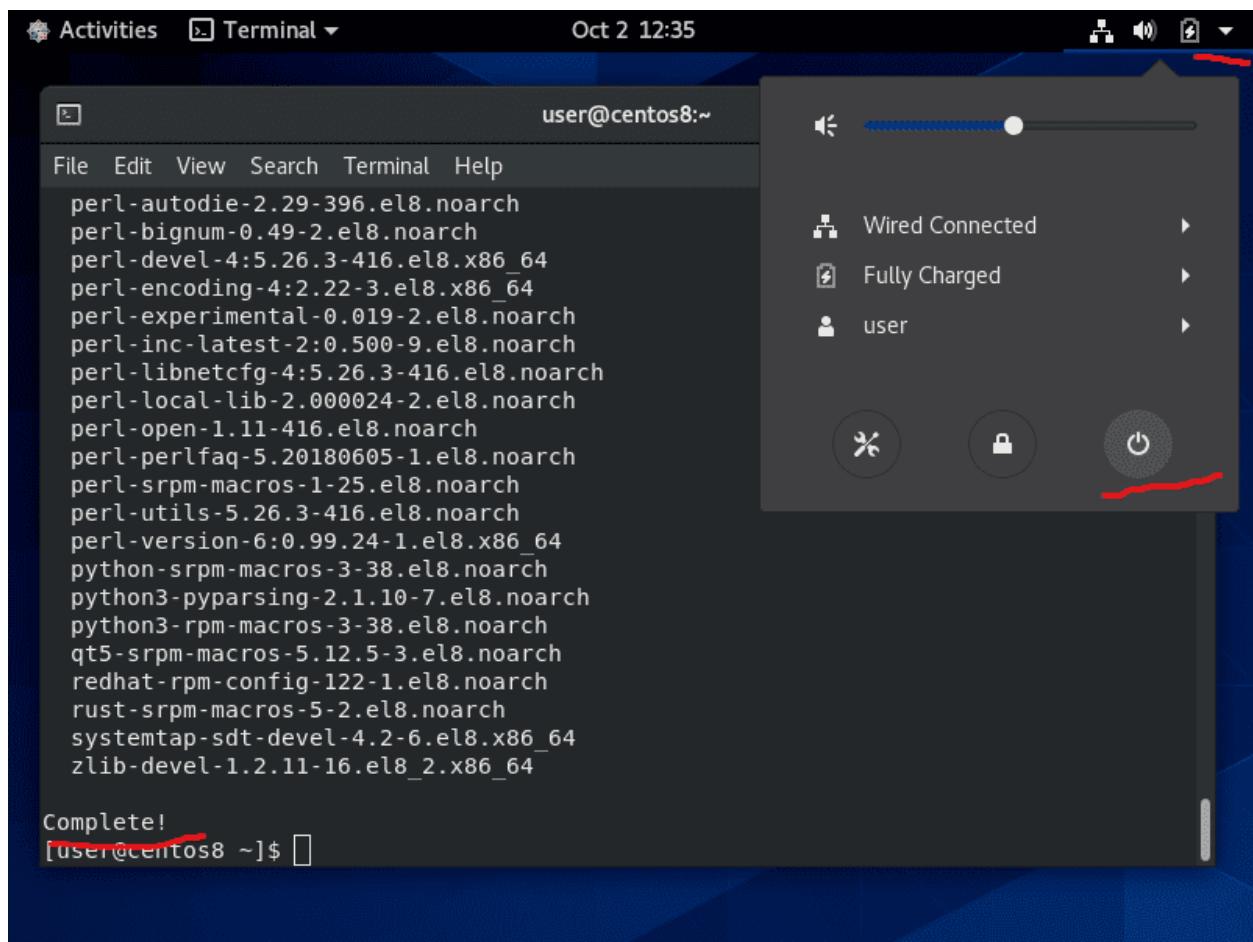
```
sos-3.8-7.el8_2.noarch
subscription-manager-1.26.20-1.el8_2.x86_64
subscription-manager-rhsm-certificates-1.26.20-1.el8_2.x86_64
systemd-239-31.el8_2.2.x86_64
systemd-container-239-31.el8_2.2.x86_64
systemd-libs-239-31.el8_2.2.x86_64
systemd-pam-239-31.el8_2.2.x86_64
systemd-udev-239-31.el8_2.2.x86_64
teamd-1.29-1.el8_2.2.x86_64
tzdata-2020a-1.el8.noarch
unbound-libs-1.7.3-11.el8_2.x86_64
yum-4.2.17-7.el8_2.noarch
zlib-1.2.11-16.el8_2.x86_64

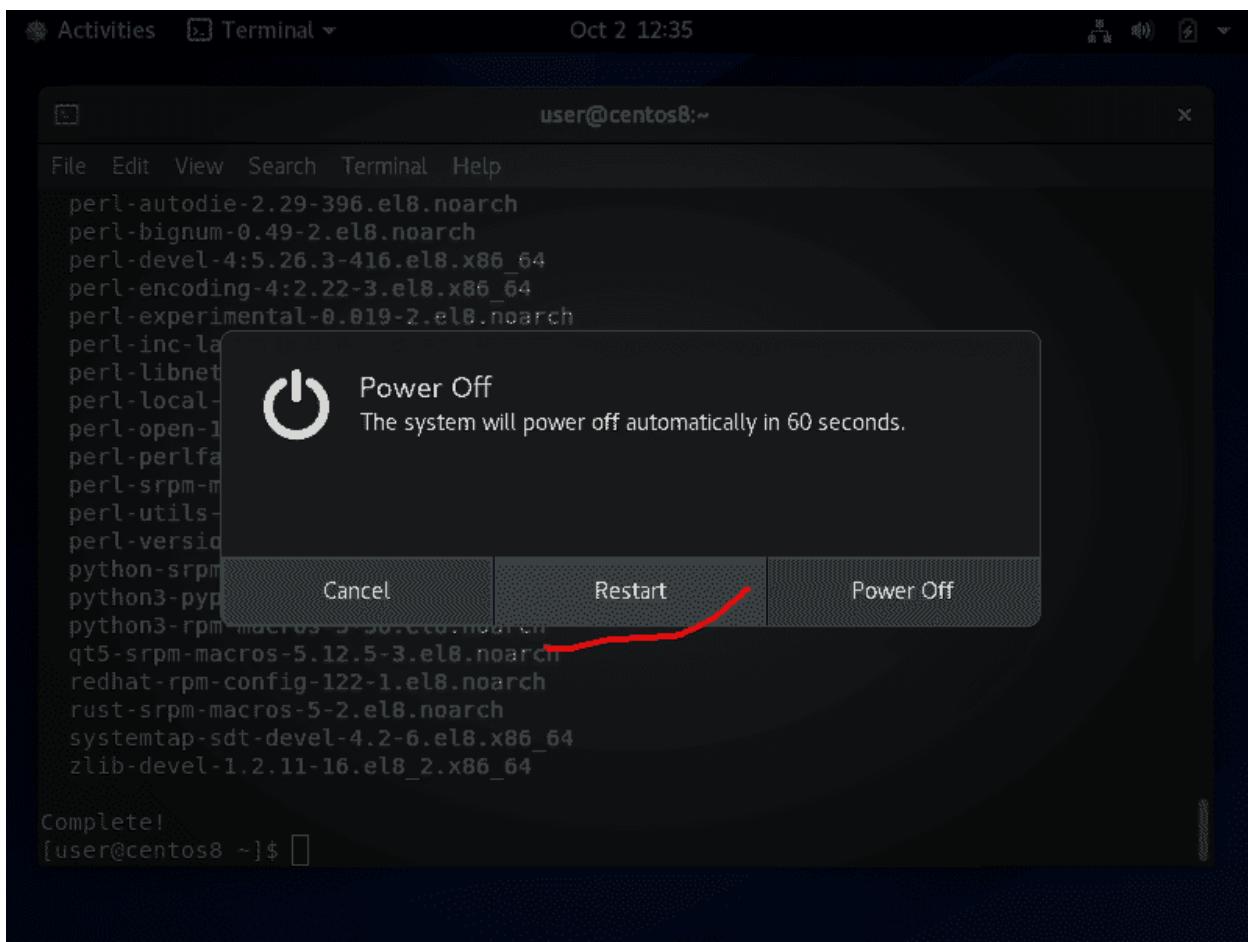
Installed:
grub2-tools-efi-1:2.02-87.el8_2.x86_64
kernel-4.18.0-193.19.1.el8_2.x86_64
kernel-core-4.18.0-193.19.1.el8_2.x86_64
kernel-modules-4.18.0-193.19.1.el8_2.x86_64

Complete!
[user@centos8 ~]$ sudo dnf install kernel-devel kernel-headers elfutils-libelf-devel gcc make perl -y
[sudo] password for user:
```

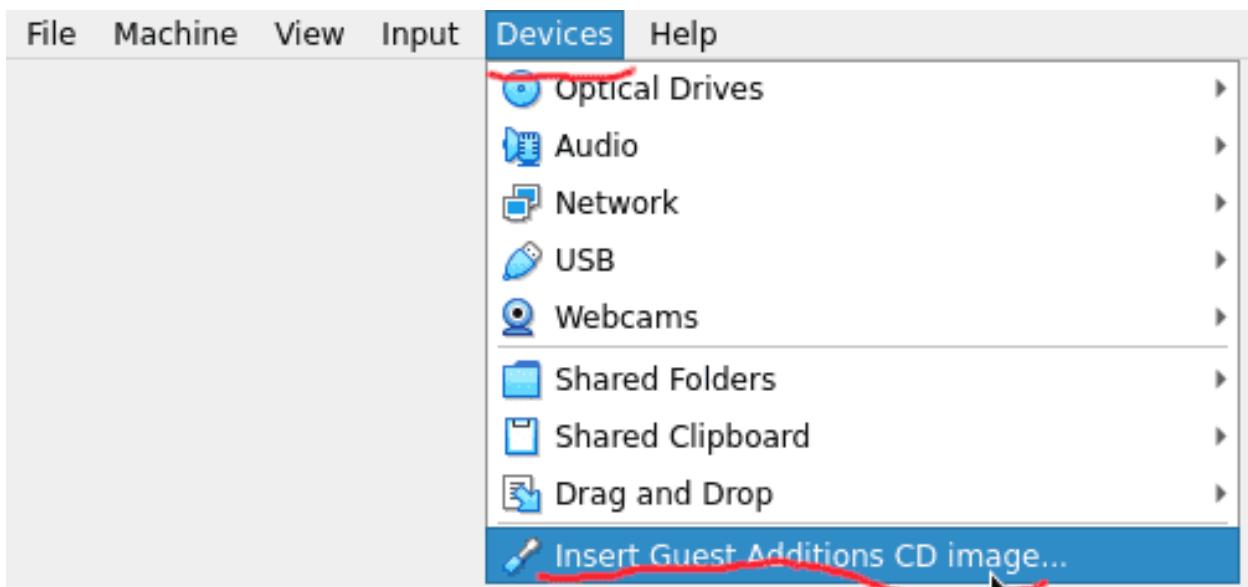
После обновления нужно установить пару программ (`sudo dnf install kernel-devel kernel-headers elfutils-libelf-devel gcc make perl -y`). Это нужно для дальнейшей установки гостевых утилит - драйверов и утилит, необходимых виртуальной операционной системе для работы с гипервизором. Без них виртуалка тоже работает, но некоторые функции будут работать неполноценно - не будет определяться разрешение экрана, не будет общего буфера обмена и т.п. На продвинутых гипервизорах такие дополнения могут показывать на хосте детальную информацию об операционной системе виртуалки и даже управлять ей.

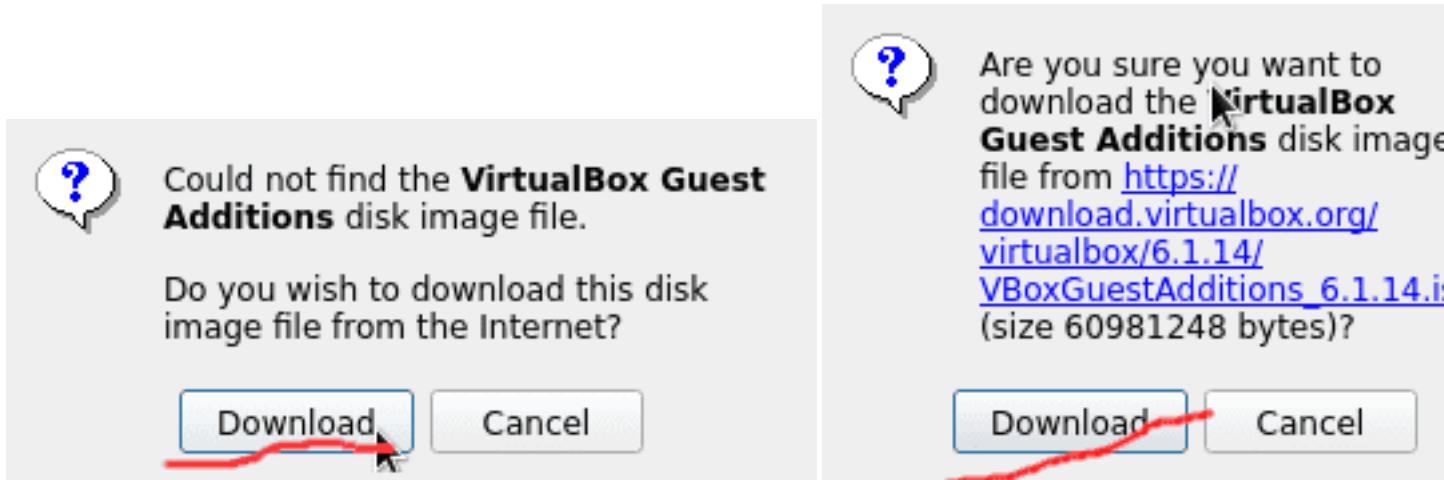
```
sudo dnf install kernel-devel kernel-headers elfutils-libelf-devel gcc make perl -y
```



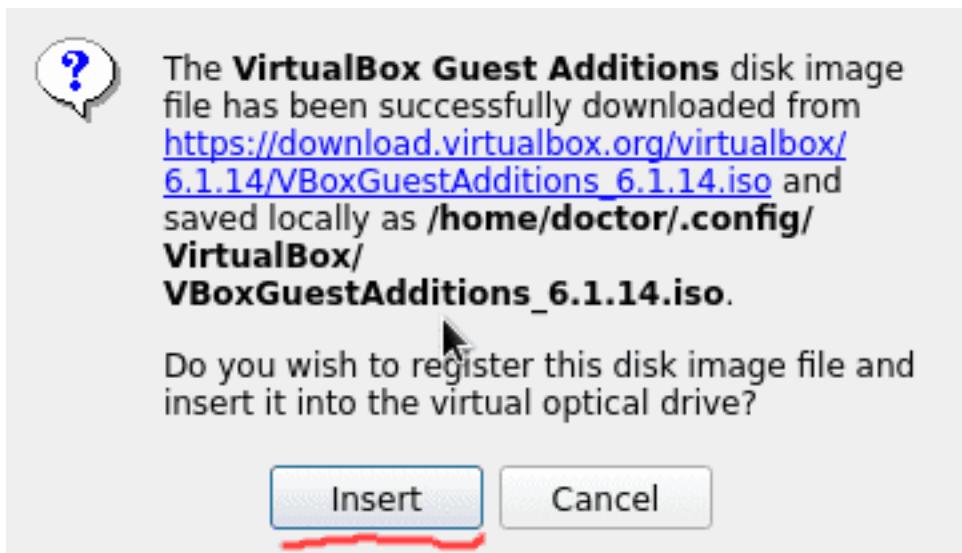


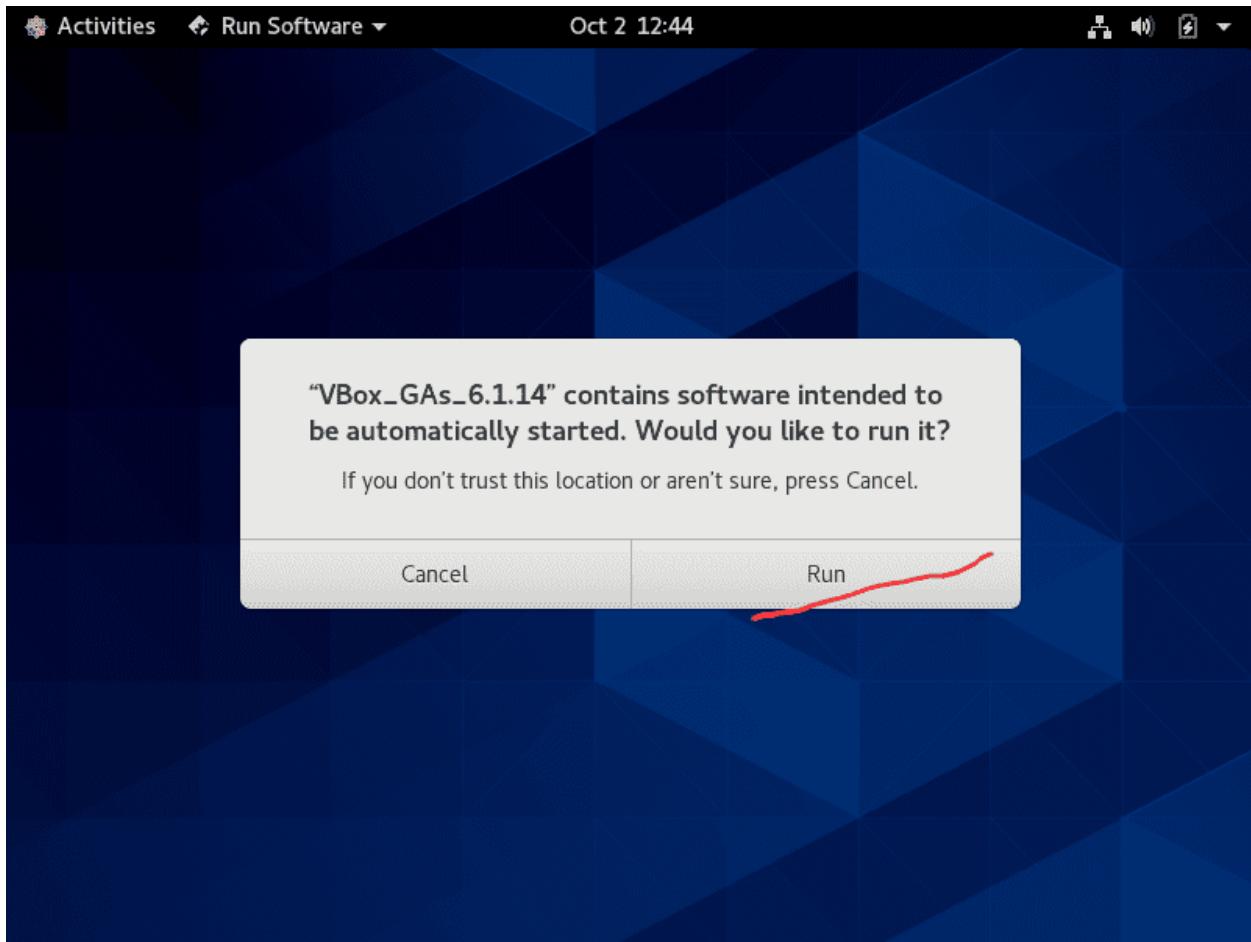
После того как успешно выполнилась предыдущая команда, перезагружаем виртуальную машину, затем снова логинимся нашим пользователем.

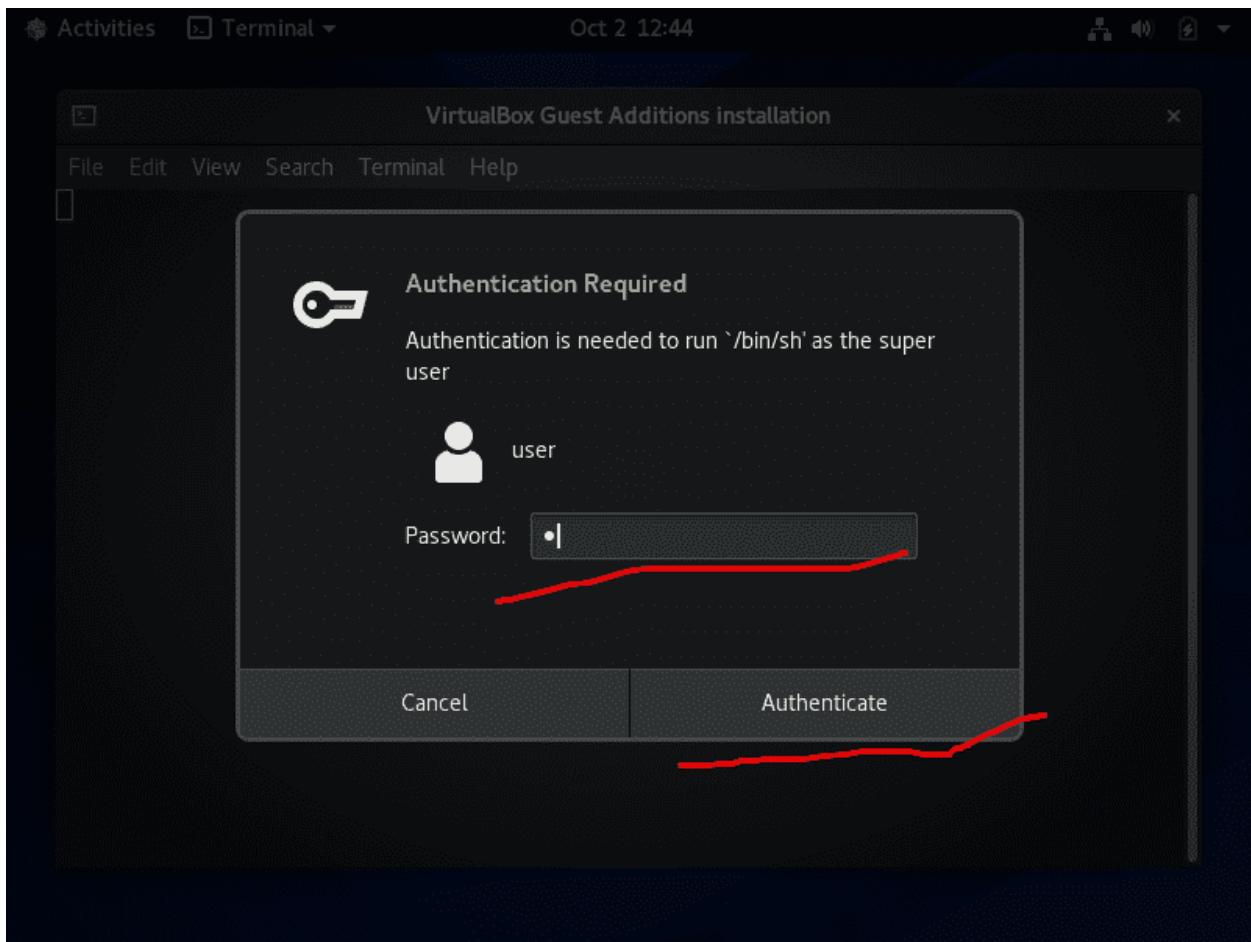




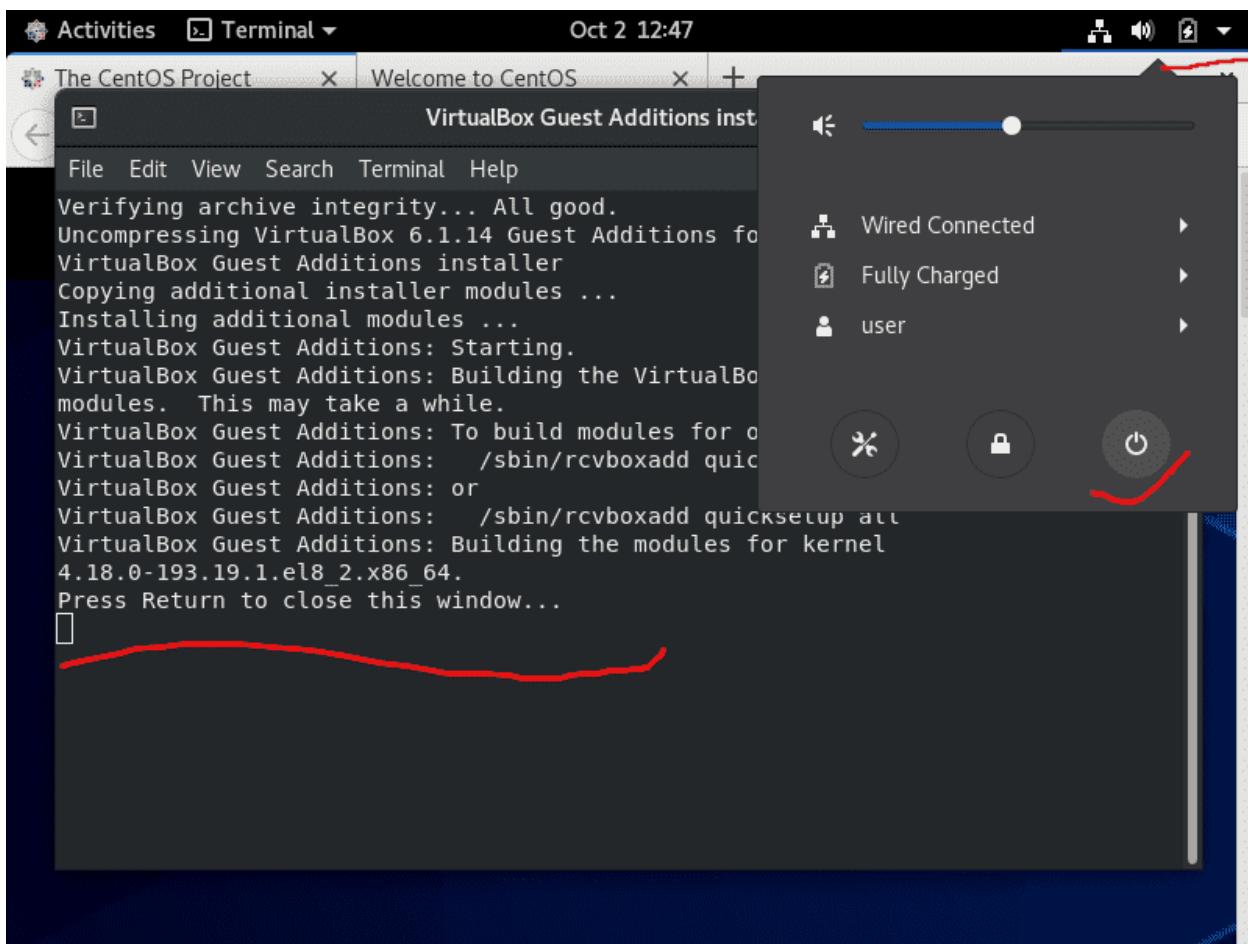
Дальше, в окне VirtualBox-а в меню Devices выбираем Insert Guest Additions CD Image, скачиваем необходимый образ диска.



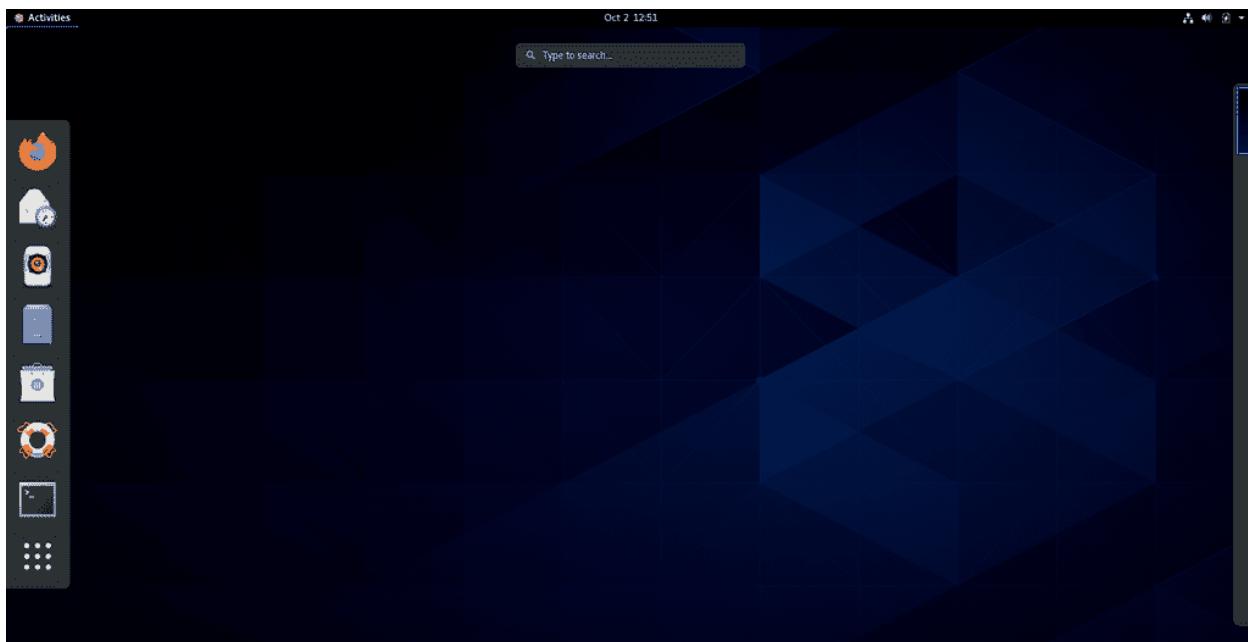




После этого диск подключится к виртуалке, где мы сможем нажать Run и программа установит гостевые утилиты.

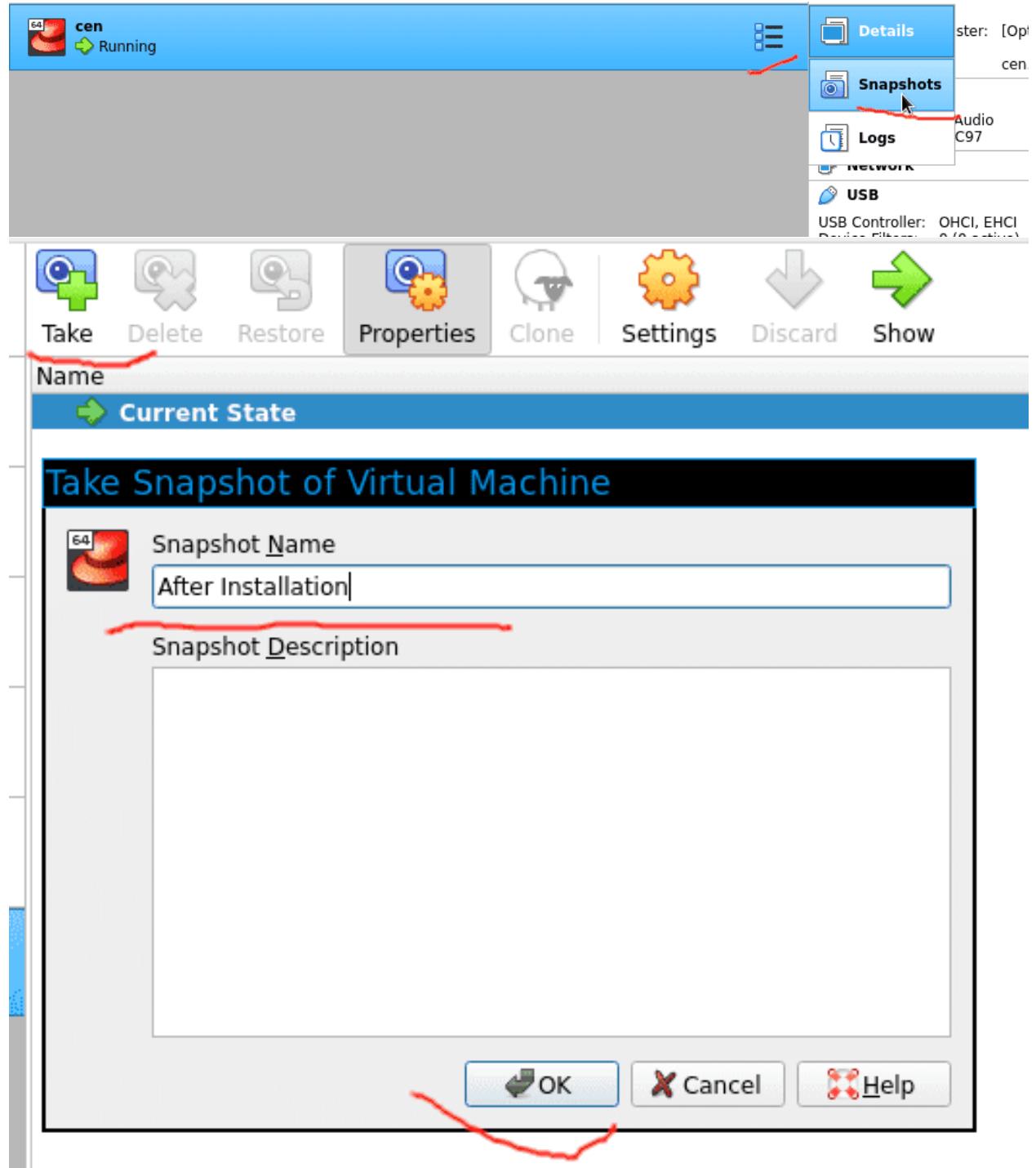


Когда в новом терминале появится надпись «Press Return to close this window», нужно ещё раз перезагрузить компьютер. На этот раз операционная система правильно определит размер экрана и подстроится под размер окна VirtualBox.

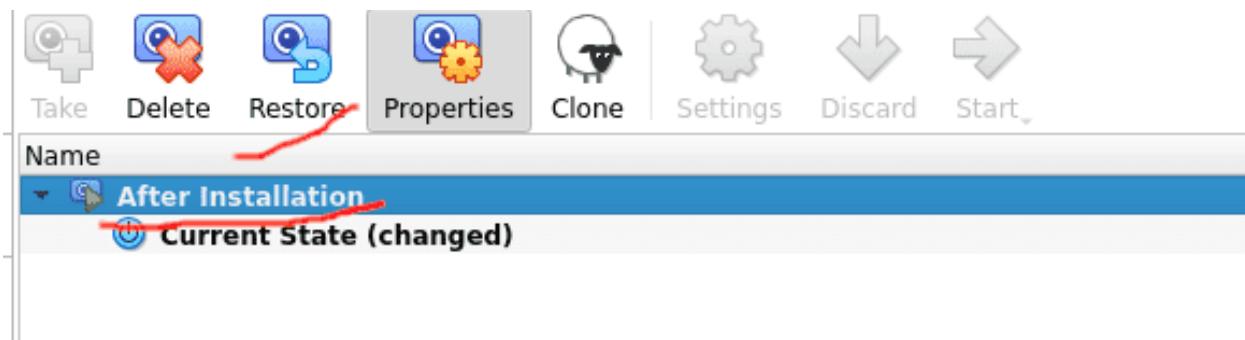


Снапшоты

Советую вам потыкать настройки и программы, чтобы ближе познакомиться с графическим интерфейсом. А чтобы не пришлось переустанавливать систему после того, как вы случайно что-то сломаете, давайте сделаем снимок. Это функция гипервизора позволяет сохранить текущее состояние виртуальной машины, что позволит в дальнейшем откатиться в к этому состоянию.



Для этого нажимаем на значок рядом с виртуальной машиной (Tools), выбираем Snapshots, нажимаем Take и указываем название и описание.



Снапшотов может быть несколько, чтобы откатиться к какому-то состоянию – выключаем виртуалку, выбираем снапшот и нажимаем restore.

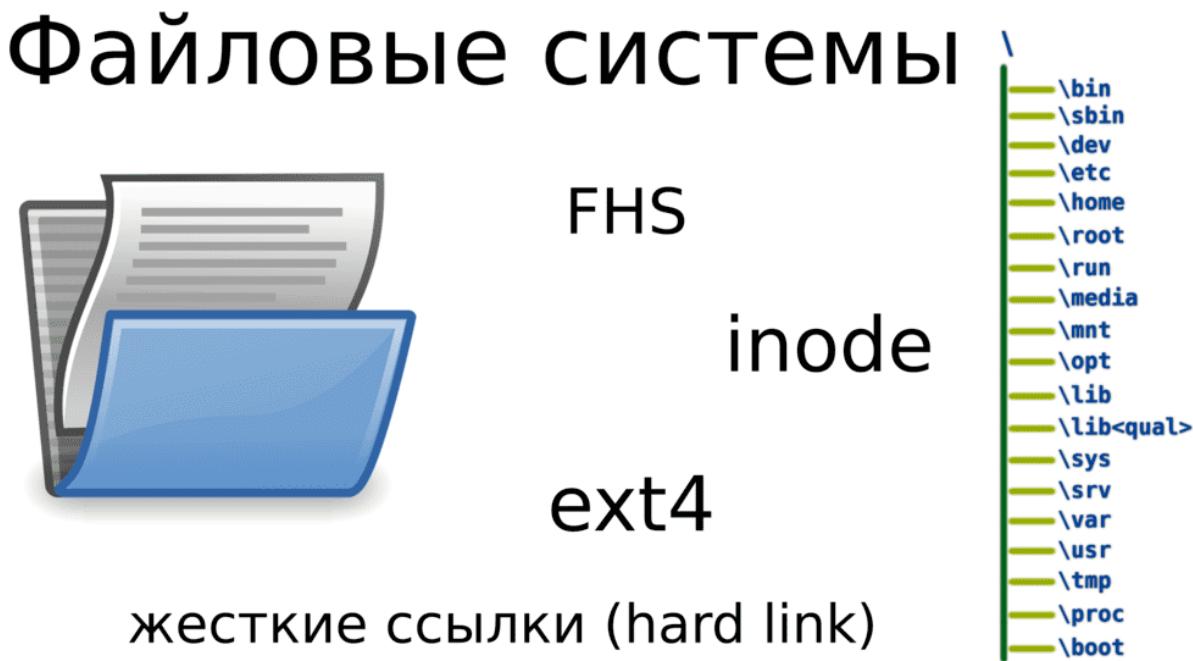
2.3.2 Практика

Задания

1. Установите AlmaLinux

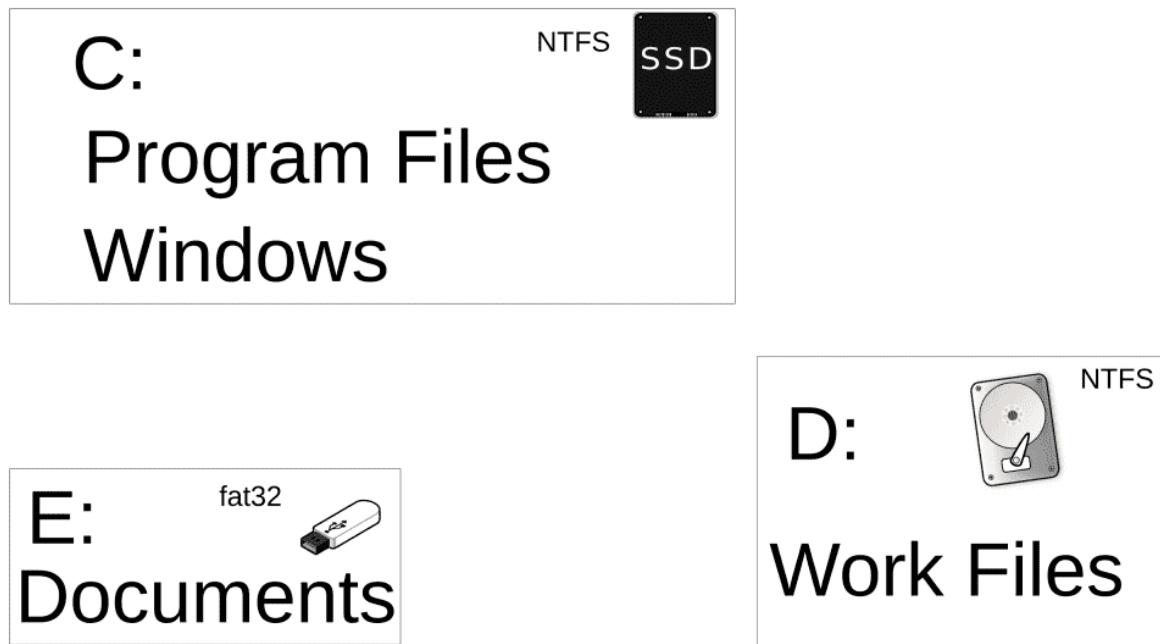
2.4 04. О файловых системах

2.4.1 04. О файловых системах

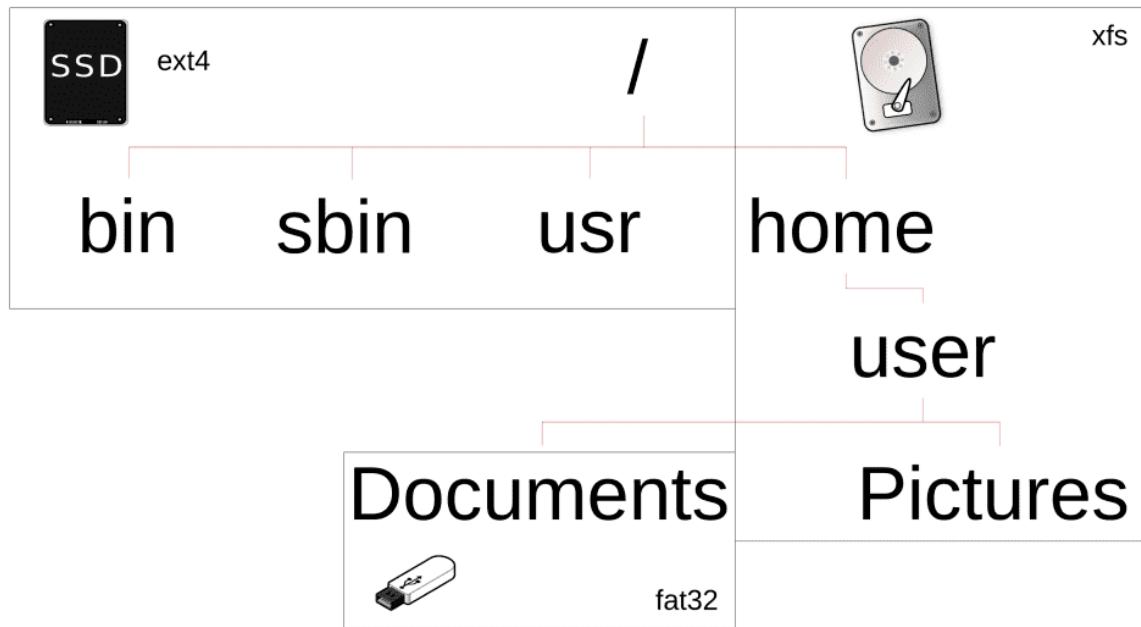


Компьютер – это инструмент для работы с информацией, которая хранится в виде файлов. UNIX-подобные операционные системы придерживаются идеи «Всё есть файл», поэтому в качестве файлов также выступают устройства, процессы, директории, сокеты, пайпы – но об этом пока рано говорить. Современная операционная система состоит из десятков и сотен тысяч файлов, которые нужно правильно организовать и предоставить пользователям и программам удобный доступ к ним. Для этого используется файловая система. Но это довольно широкий термин, для понимания которого нужно несколько точек зрения.

С точки зрения пользователей или прикладных программ, файловая система – это место для записи и чтения файлов. Пользователи видят иерархическую структуру организации файлов – внутри директорий хранятся файлы и другие директории, которые также могут хранить файлы и директории. На Windows системах директории обычно называют папками, но папка – это термин, относящийся к графической оболочке операционной системы, а директория – к файловой системе. У папки можно поменять иконку, цвет или добавить описание, чего не сделаешь с директорией. И хотя на линуксах тоже может стоять графический интерфейс, но при общении люди обычно подразумевают не цвет и не иконку какой-нибудь папки, а содержимое и атрибуты файловой системы, поэтому закрепилось выражение директория.



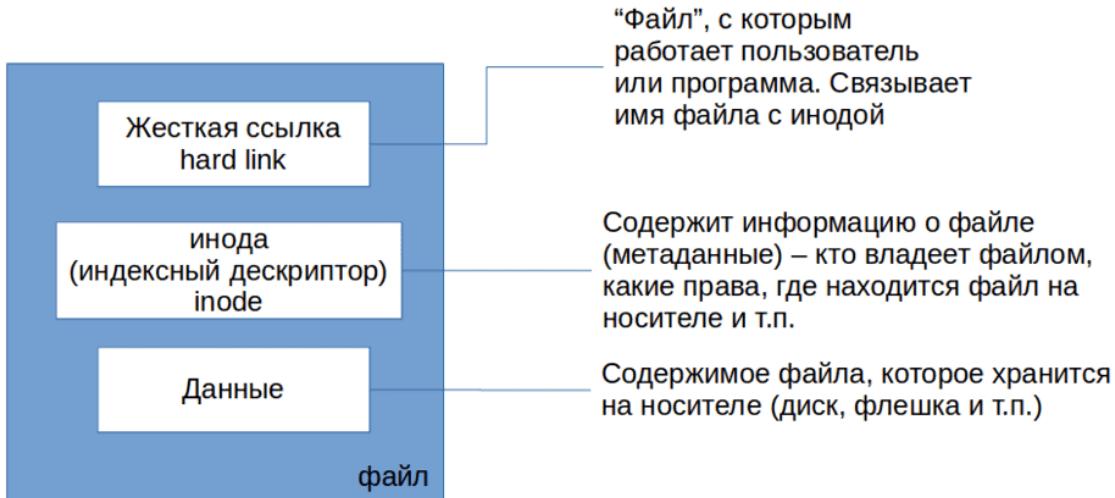
Структура организации файлов на Windows системах и UNIX-подобных системах несколько отличается. На Windows у вас есть некая файловая система, которой вы назначаете букву С. Внутри этой файловой системы у вас несколько директорий – где-то хранятся файлы пользователей, где-то программы, а где-то файлы операционной системы. Когда вы подключаете флешку или любое другое устройство со своей файловой системой, ей назначается другая буква и через неё вы можете увидеть содержимое этой файловой системы. Структуры этих файловых систем независимы и не пересекаются.



На UNIX подобных системах используется другой подход, называемый FHS – стандартом иерархии файловой системы. По этому стандарту у многих файлов и директорий есть специальные пути, где они должны храниться. Допустим, если на Windows системах вы устанавливаете программу, то все её файлы попадают в директорию C:\Program Files\ИмяПрограммы. А на UNIX-подобных системах большинство программ «размазывается по системе» – запускаемая часть программ попадает в директорию /usr/bin/, ярлыки программ попадают в /usr/share/applications/, настройки в /etc/ и т.д. Кроме удобства, это также помогает с безопасностью при правильных настройках.

Это следует из следующего отличия – корень на UNIX-подобных системах один. Корень, с точки зрения структуры, это начало файловой системы, внутри которого хранятся первые директории. На Windows у каждой файловой системы свой корень – С:, D: и т.п.. А на UNIX-подобных системах каждая файловая система «прикрепляется», правильнее сказать монтируется, в какую-нибудь директорию внутри единого корня. Допустим, у вас может быть корневая файловая система на SSD, файлы пользователя хранятся на отдельном диске, при этом доступ к ним в директории /home, ваши документы, которые вы держите на флешке, могут быть доступны в директории /home/user/Documents. Три разных устройства, три разные файловые системы, но внутри одного корня.

Возвращаясь к теме безопасности, различным файловым системам можно задать различные опции монтирования. Допустим, если у вас /home, где обычно хранятся файлы пользователя, на отдельной файловой системе, вы можете запретить запуск программ на этой файловой системе. Программы обычно лежат в директории /usr/bin/, куда у обычного пользователя нет прав для доступа. В итоге, если какой-то пользователь скачает какой-то вирус к себе в домашнюю директорию (/home/user), он просто не сможет его запустить. Или, допустим, мы знаем, что в /usr/ лежат файлы программ. Если от分离 /usr/ на другую файловую систему, то после установки всех программ мы можем убрать право записи в эту файловую систему. В итоге вирус не сможет воспользовавшись уязвимостью программы изменить её запускаемый файл, чтобы добавить в него вредоносный код. А в момент обновления мы просто возвращаем файловой системе право на запись, обновляем программы и опять возвращаем как было.



Как вы, возможно, заметили, я упомянул файловые системы ещё с двух точек зрения – когда я говорил о файловых системах на устройствах и когда говорил про опции на запуск или запись. С точки зрения устройства, файловая система - это способ записи и чтения файлов на устройстве. Недостаточно просто записать файл на диск, нужно ещё определить, куда писать файл, как с ним работать и т.п. Это зависит от типа файловых систем – NTFS, exFAT, EXT4, XFS и т.д.. Файл условно можно разделить на 3 части – сами данные внутри файла, жёсткая ссылка и информация об этом файле – где хранятся данные файла на устройстве, кто владелец, какие права доступа и т.п. Информация о файле называется метаданными и на UNIX-подобных системах хранится на файловых системах в структурах данных, называемых инодами(inode). У каждой иноды есть свой уникальный номер, а чтобы операционная система могла найти иноду по имени файла, она использует жёсткую ссылку (hard link). Подводя итоги, файлы, которые мы обычно видим на компьютере – это жёсткие ссылки, в которых содержится номер иноды, а в иноде содержится информация о файле и его местоположение на устройстве. Этих жёстких ссылок, которые ссылаются на одну и ту же иноду, может быть несколько, по сути это способ обращения к одному и тому же файлу по разным именам и с разных директорий, но внутри одной файловой системы, потому что жёсткая ссылка – часть файловой системы. Когда вы удаляете последнюю жёсткую ссылку на файл, файловая система очищает inode запись и помечает местоположение данных файла как свободное, хоть там и лежат данные. Когда появится необходимость записать новый файл, файловая система запишет туда новые данные, но пока этого не произошло, есть вероятность восстановить эти данные.

С точки зрения операционной системы, файловая система – это драйвер, модуль ядра. Этот драйвер предоставляет интерфейс, через который программы могут взаимодействовать с файлами. И в момент монтирования файловой системы вы можете указать специальные опции монтирования, например, чтобы файловая система была только для чтения. На GNU/Linux системах есть возможность установить модули для работы с множеством различных типов файловых систем, включая файловые системы других операционных систем, допустим NTFS. Но на NTFS нет некоторых функций, необходимых для работы линукса – допустим, прав, как в UNIX-подобных системах и т.п., вследствие чего невозможно установить Linux на NTFS, хотя и можно использовать такую файловую систему для хранения пользовательских файлов. Windows системы же не содержат нужных драйверов для работы с файловыми системами EXT4 или XFS, на которых обычно устанавливается Linux. Из-за чего, если у вас 2 операционные системы на компьютере, на Линуксе видны файлы Windows, а чтобы увидеть содержимое линуксовых файловых систем, на Windows надо установить специальные утилиты.

Есть множество различных типов файловых систем, различающихся функционалом и возможностями, но самые используемые на GNU/Linux – ext4 и xfs. Сейчас я рассказал только о том, что такое файловая система, чтобы мы могли рассмотреть работу с файлами в следующих видео. А к теме «Работа с файловыми системами» мы ещё вернёмся.

2.4.2 Практика

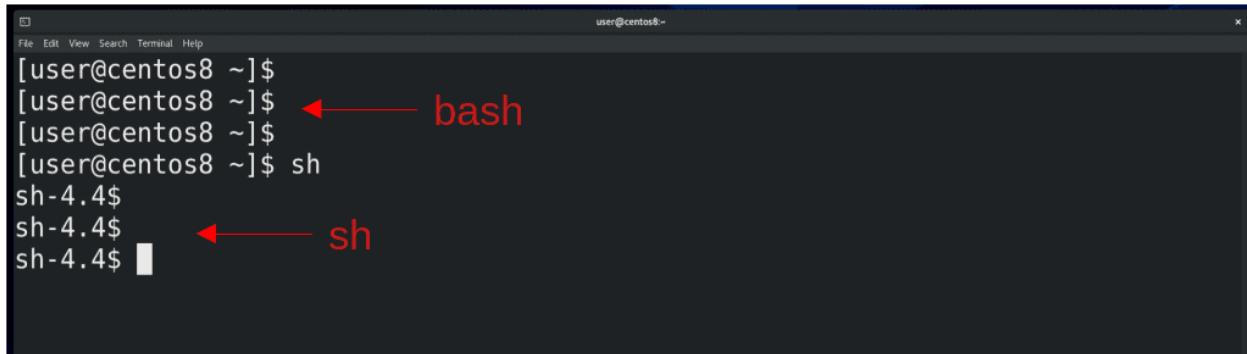
Вопросы

1. Для чего нужна файловая система?
2. Что такое файловая система?
3. Чем отличается папка от директории?
4. Как структурно отличается хранение и доступ к файлам на Windows и GNU/Linux?
5. Что из себя представляет файл на GNU/Linux?
6. Что такое инода?
7. Что означает «жёсткая ссылка»?

2.5 05. Текстовый интерфейс пользователя

2.5.1 05. Текстовый интерфейс пользователя

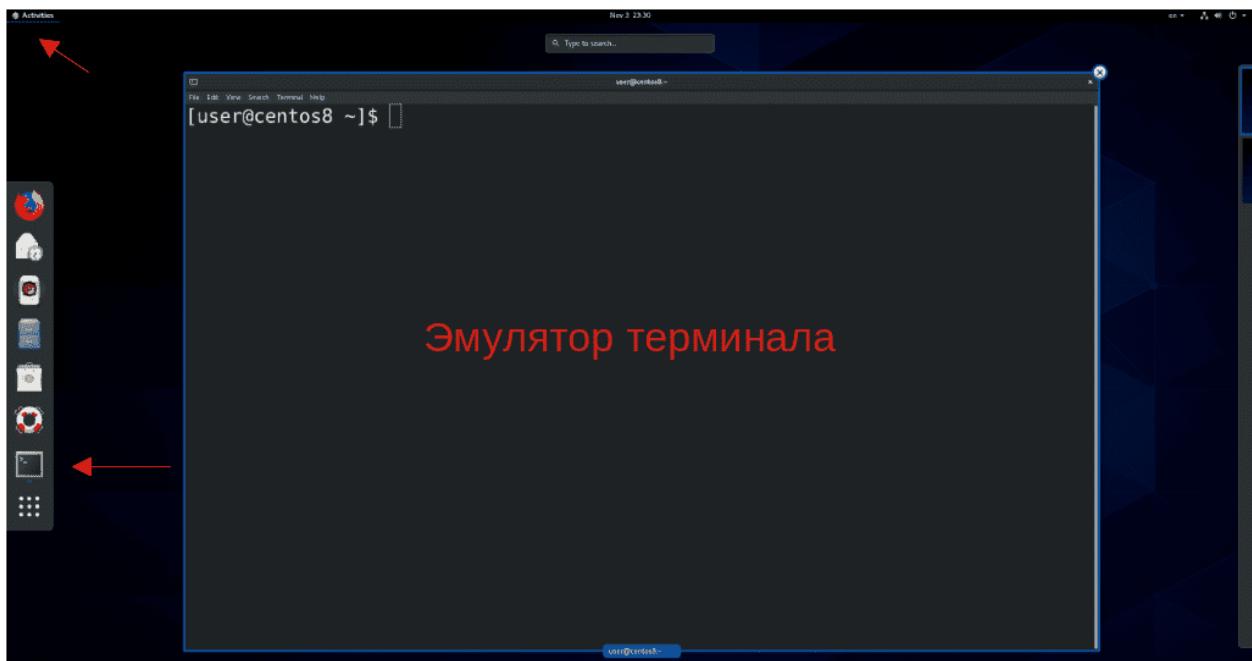
Чтобы пользователь мог взаимодействовать с компьютером, операционная система предоставляет два интерфейса – графический и текстовой. Есть заблуждение, что текстовый интерфейс нужен исключительно для задач администрирования, но, на самом деле, в текстовом интерфейсе можно слушать музыку, читать новости, общаться и даже сёрфить интернет. Графический или текстовый интерфейс – это понятие, а различные реализации называются оболочками.



The screenshot shows a terminal window with a dark background and white text. At the top, there's a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The title bar says 'user@centos8 ~ \$'. The terminal displays several command-line inputs:

```
[user@centos8 ~]$  
[user@centos8 ~]$  
[user@centos8 ~]$ bash ← arrow pointing to the word 'bash'  
[user@centos8 ~]$ sh  
sh-4.4$  
sh-4.4$ sh ← arrow pointing to the word 'sh'  
sh-4.4$ █
```

Благодаря тому, что дистрибутивы GNU/Linux разрабатывает не одна компания, а любой желающий, существует множество различных графических и текстовых оболочек, на любой вкус и цвет. Текстовые оболочки называют интерпретаторами командной строки или shell. По сути, это программы, которые могут отличаться функционалом, но стандартной оболочкой для UNIX-подобных операционных систем является sh, также называемая шелл. А одна из самых популярных называется bash. С помощью интерпретаторов пользователи могут делать практически всё, что может быть связано с запуском программ – ставить условия запуска, использовать переменные, повторять с разными значениями, автоматизировать и т.д. Одно из преимуществ bash над sh - автодополнение. Когда вы что-то печатаете на bash, часто вы можете нажать tab, который допишет команду до конца, а если есть несколько вариантов, то двойное нажатие по tab покажет их.



Есть 3 основных способа получить доступ к текстовому интерфейсу. При наличии графического интерфейса, чаще всего используется эмулятор терминала. Есть множество различных эмуляторов терминала, в основном они отличаются внешним видом и горячими клавишами.

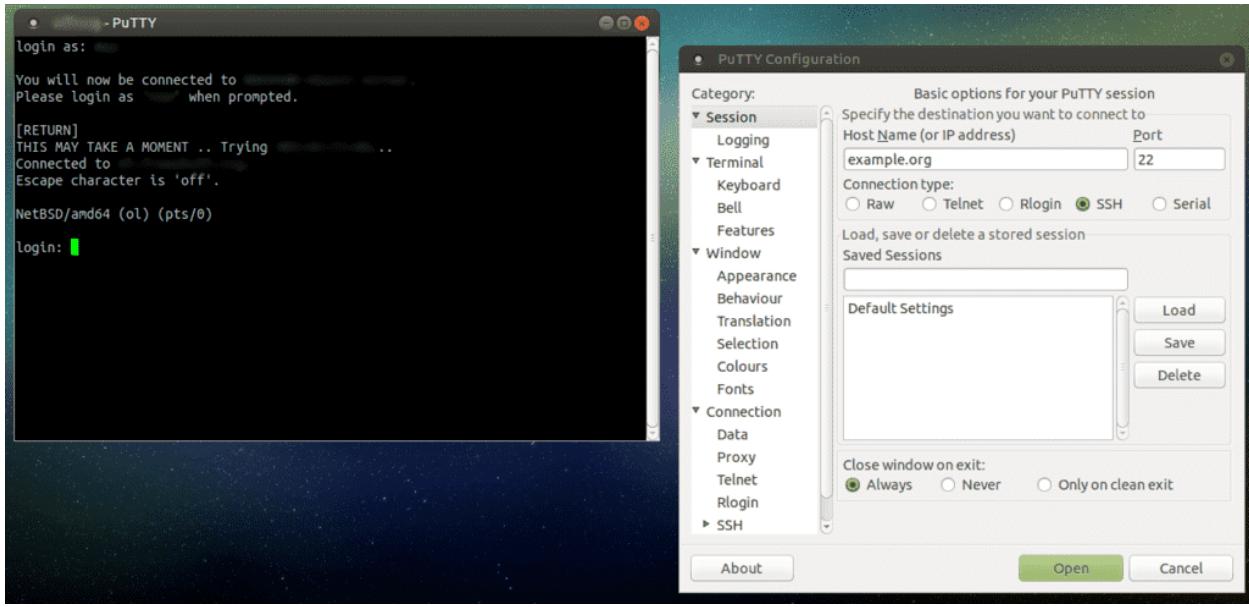
```
CentOS Linux 8 (Core)
Kernel 4.18.0-147.8.1.el8_1.x86_64 on an x86_64

Activate the web console with: systemctl enable --now cockpit.socket

centos8 login: user ←
Password:
Last login: Tue Nov  3 23:45:49 on tty3
[user@centos8 ~]$ _
```

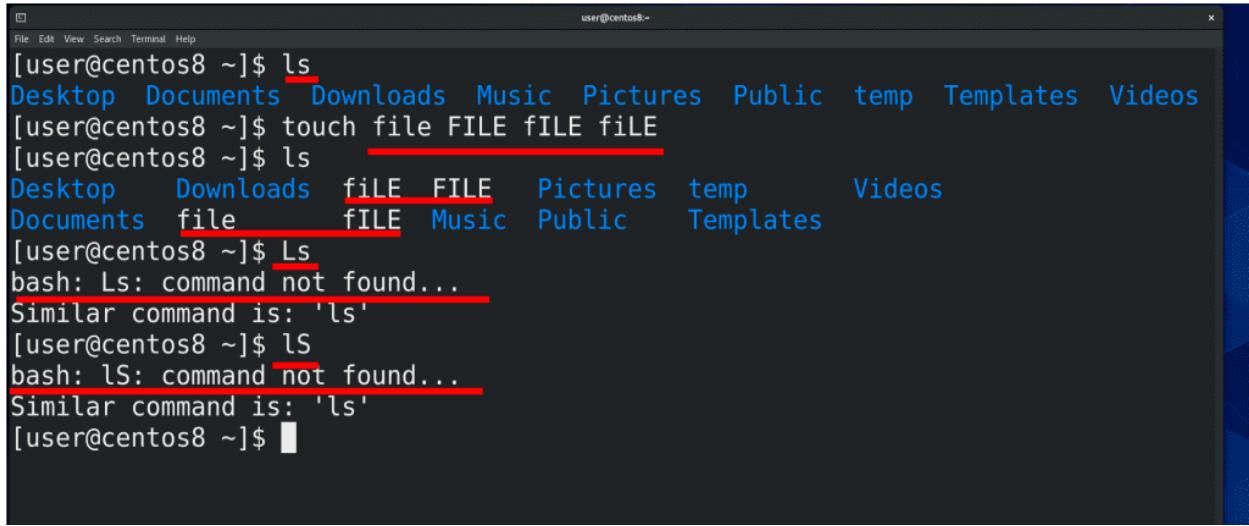
Виртуальный терминал

При отсутствии графического интерфейса, либо при каких-либо проблемах с ним, используется виртуальная консоль, также называемая виртуальным терминалом. Их обычно много и к ним можно подключиться даже имея графический интерфейс, обычно для этого используется комбинация клавиш Alt + Ctrl + F1, F2 и т.д, обычно до F6. Так как VirtualBox перехватывает клавиши, на нём это можно сделать с помощью «правый Ctrl»+F1 и т.п. Графический интерфейс запускается на одной из виртуальных консолей.



Но, чаще всего, администраторы работают с компьютерами не напрямую, а удалённо. Для этого они используют программы для удалённого доступа, работающие с протоколом SSH. Одна из самых популярных на Windows называется Putty, но в Windows 10 это можно делать и без дополнительных программ, просто написав в терминале Windows команду ssh пользователь@айпи.адрес и введя пароль пользователя.

Программы, работающие в текстовом интерфейсе, часто называют командами. Этих команд огромное количество и знать всё наизусть не нужно. Большинство команд, с которыми вы будете работать, вы без труда запомните, так как их создавали для людей - они просты и логичны. Сложные команды всегда можно где-нибудь записать или найти. Чтобы знать, как работать с командой, вы должны знать её синтаксис.



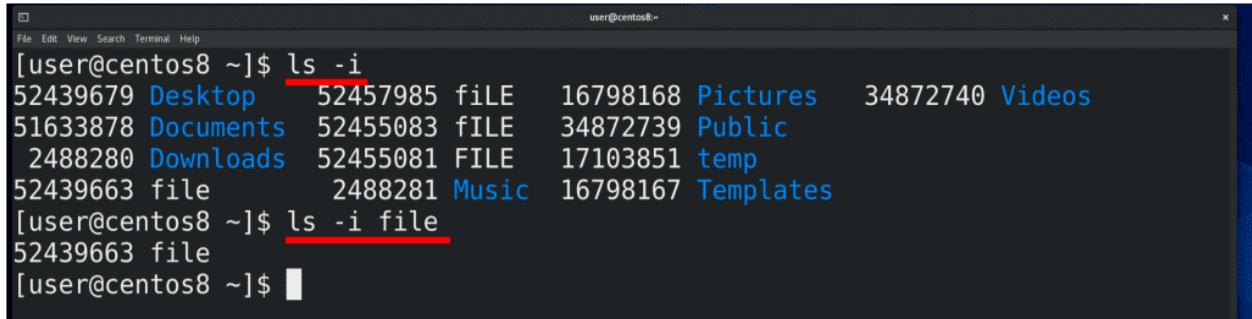
Давайте, для примера, рассмотрим команду ls.

```
ls
```

ls – от слова list – команда, с помощью которой можно просмотреть содержимое директории. Запомните – GNU/Linux – регистрозависимая операционная система. Если вы напишете L большое или S большое – получится уже другое значение, а так как таких команд нет, интерпретатор выдаст ошибку. Тоже

самое касается файлов – файлы file маленькими, FILE большими, ffile, fFILE и т.д. это разные файлы, которые никак друг с другом не конфликтуют. Но это больше относится файловым системам.

Когда у вас в терминале много текста это может напрягать, поэтому терминал можно легко очищать с помощью комбинации Ctrl+L или команды clear. А с помощью стрелок вы можете листать ранее выполненные команды.

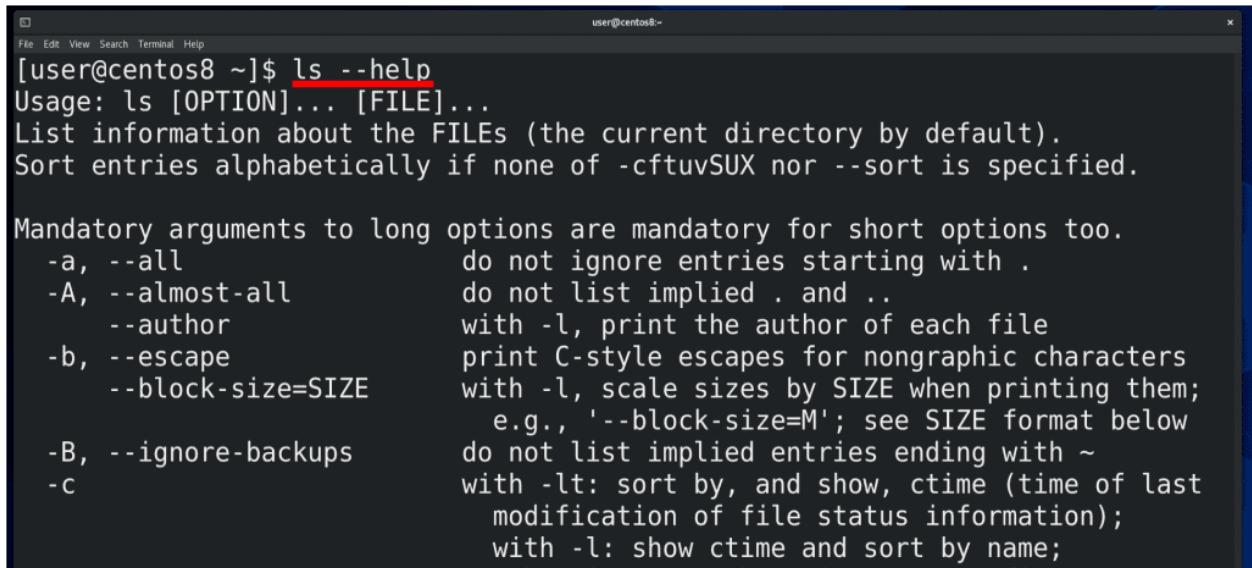


```
[user@centos8 ~]$ ls -i
52439679 Desktop      52457985 file      16798168 Pictures    34872740 Videos
51633878 Documents    52455083 FILE     34872739 Public
2488280 Downloads    52455081 FILE     17103851 temp
52439663 file        2488281 Music    16798167 Templates
[user@centos8 ~]$ ls -i file
52439663 file
[user@centos8 ~]$
```

Возвращаясь к команде ls. У большинства команд есть различные опции и многим командам можно передавать значения. Для примера, команда ls -i покажет содержимое директории с соответствующими инодами. «-i» в данном случае – опция. Опции часто называют ключами команды. Если написать ls -i file, то мы передали команде ключ -i и значение в виде имени файла. Как результат, баш нам показал только информацию по одному файлу.

`ls -i file`

Что делать, если вы забыли или не знаете какой-то ключ или в целом синтаксис? Для этого в системе есть документация, которая также будет доступна во время экзамена по Red Hat. Кстати, об экзамене. Во время экзамена вы вольны пользоваться как графическим интерфейсом, так и текстовым, но, поверьте, на работу с графическим интерфейсом у вас просто не хватит времени. Работать в текстовом интерфейсе гораздо эффективнее. Возвращаясь к документации. Есть 3 основных способа получить доступ к документации – с помощью утилиты man, с помощью утилиты info и при помощи встроенной в большинство команд опции help.



```
[user@centos8 ~]$ ls --help
Usage: ls [OPTION]... [FILE]...
List information about the FILEs (the current directory by default).
Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

Mandatory arguments to long options are mandatory for short options too.
-a, --all                  do not ignore entries starting with .
-A, --almost-all           do not list implied . and ..
--author                 with -l, print the author of each file
-b, --escape                print C-style escapes for nongraphic characters
--block-size=SIZE           with -l, scale sizes by SIZE when printing them;
                           e.g., '--block-size=M'; see SIZE format below
-B, --ignore-backups       do not list implied entries ending with ~
-c                         with -lt: sort by, and show, ctime (time of last
                           modification of file status information);
                           with -l: show ctime and sort by name;
                           with -t: sort by, and show,_mtime (time of last
                           modification of file status information);
```

Самый быстрый и простой способ – опция help.

```
ls --help
```

Как правило, это не документация, а небольшая подсказка по ключам и синтаксису, которая доступна, если писать команду, а затем два дефиса и help.

```
LS(1) User Commands LS(1)

NAME
    ls - list directory contents

SYNOPSIS
    ls [OPTION]... [FILE]...

DESCRIPTION
    List information about the FILEs (the current directory by default).
    Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

    Mandatory arguments to long options are mandatory for short options
    too.
```

man – утилита для чтения документации, доступная на большинстве UNIX-подобных систем. Допустим, меня интересует документация по команде ls. Я пишу man ls:

```
man ls
```

и открывается страница документации. Чтобы её закрыть, нажимаем q. q часто используется различными терминальными программами для выхода. Документацию можно листать с помощью стрелок и клавиш PgUp/PgDn.

```
(standard), -dash (-p), file-type (-T FILE_TYPE), classify (-c)
      -i, --inode
          print the index number of each file

      -I, --ignore=PATTERN
          do not list implied entries matching shell PATTERN

/ -i
```

Если написать / , а потом какой-то текст, допустим ключ -i, и нажать enter программа попытается найти этот текст в документации. Как правило, в начале документации показывается имя программы, синтаксис, описание и ключи, а в конце у некоторых программ можно найти примеры команд в секции EXAMPLES, а также связанные программы.

The terminal window shows the man page for the 'ls' command. The title bar says 'user@centos8:~'. The content of the window is:

```

Next: dir invocation, Up: Directory listing
10.1 'ls': List directory contents
=====
The 'ls' program lists information about files (of any type, including
directories). Options and file arguments can be intermixed arbitrarily,
as usual.

```

Что касается утилиты info:

```
info ls
```

она пришла на замену man, хотя большинство до сих пор пользуются man. info лучше работает с большими документами и поддерживает гиперссылки.

The terminal window shows the command 'man -k password' being run. The output is:

```

[user@centos8 ~]$ man -k password
chage (1)          - change user password expiry information
chgpasswd (8)       - update group passwords in batch mode
chpasswd (8)        - update passwords in batch mode
cracklib-check (8)  - Check passwords using libcrack2
create-cracklib-dict (8) - Check passwords using libcrack2
endpwent (3)        - get password file entry
endspent (3)        - get shadow password file entry
fgetpwent (3)       - get password file entry
fgetspent (3)       - get shadow password file entry
fgetspent_r (3)     - get shadow password file entry

```

А что делать, если вы не знаете команду? В таком случае можно попытаться её найти. Кроме гугла, это можно сделать с помощью команды man с опцией -k, либо команды apropos. Допустим, я не знаю, как поменять пароль пользователю. Я могу написать man -k password:

```
man -k passwd
```

и man выдаст мне все страницы документации, в секции NAME которых есть слово password. Результатов может быть много, но, если поискать, можно найти подходящий вариант.

Кстати, вы заметили циферки рядом с названиями? Давайте посмотрим, что это такое.

The terminal window shows the command 'man -k passwd' with section numbers. The output is:

```

1 Executable programs or shell commands
2 System calls (functions provided by the kernel)
3 Library calls (functions within program libraries)
4 Special files (usually found in /dev)
5 File formats and conventions eg /etc/passwd
6 Games
7 Miscellaneous (including macro packages and conventions), e.g.
   man(7), groff(7)
8 System administration commands (usually only for root)
9 Kernel routines [Non standard]

```

Для этого зайдём в документацию самого man, написав man man:

```
man man
```

Здесь написано, что man состоит из нескольких секций и эти цифры относятся к определённым секциям. Допустим, 1 – это про команды и функции оболочки, 5 – это про формат файлов и т.д.

The screenshot shows a terminal window with the title bar "user@centos8:~". The command entered is "[user@centos8 ~]\$ man -k passwd". The output lists various man pages related to password management, with some entries underlined:

```
[user@centos8 ~]$ man -k passwd
chpasswd (8)          - update group passwords in batch mode
chpasswd (8)          - update passwords in batch mode
fgetpwent_r (3)       - get passwd file entry reentrantly
getpwent_r (3)       - get passwd file entry reentrantly
gpasswd (1)           - administer /etc/group and /etc/gshadow
grub2-mkpasswd-pbkdf2 (1) - Generate a PBKDF2 password hash.
ldappasswd (1)        - change the password of an LDAP entry
lpasswd (1)            - Change group or user password
openssl-passwd (lssl) - compute password hashes
pam_localuser (8)     - require users to be listed in /etc/passwd
passwd (1)             - update user's authentication tokens
passwd (5)             - password file
passwd2des (3)         - RFS password encryption
```

В некоторых случаях, для одного слова есть несколько страниц в документации в разных секциях, как например для passwd. passwd у нас это не только команда для смены пароля, но и специальный файл, в котором хранится информация о пользователях. Чтобы посмотреть документацию по команде, пишем man 1 passwd, а чтобы посмотреть документацию по файлу, пишем man 5 passwd.

```
man 1 passwd
man 5 passwd
```

The screenshot shows a terminal window with the following session:

```

[User@centos8 ~]$ history
47 history
48 ls
49 man man
50 man ls
51 man -k passwd
52 ls file
53 history
[User@centos8 ~]$ ls
Desktop Downloads file FILE Pictures temp Videos
Documents file FILE Music Public Templates
[User@centos8 ~]$ !!
ls
Desktop Downloads file FILE Pictures temp Videos
Documents file FILE Music Public Templates
[User@centos8 ~]$ !48
ls
Desktop Downloads file FILE Pictures temp Videos
Documents file FILE Music Public Templates
[User@centos8 ~]$ 

```

The terminal window has a dark background with light-colored text. The menu bar at the top includes File, Edit, View, Search, Terminal, and Help. The title bar says "user@centos8:~". The command history is displayed in blue, while other text is in white.

Ну и напоследок. Всё что вы вводите в терминале, сохраняется в истории. Для просмотра введённых команд используйте команду history.

history

Для повторения предыдущей команды используйте два восклицательных знака, а для повторения какой-то определённой команды из истории, напишите восклицательный знак и номер команды.

!!
!48

Я постарался рассмотреть самые базовые принципы работы с текстовым интерфейсом, но это очень огромная тема с большим количеством всяких плюшек, которые облегчают работу администратору. По мере того, как мы будем проходить другие темы, я буду рассказывать и о других нюансах, но на пока этого достаточно. Без практики вы быстро забудете все команды, поэтому обязательно уделяйте время практике.

2.5.2 Практика

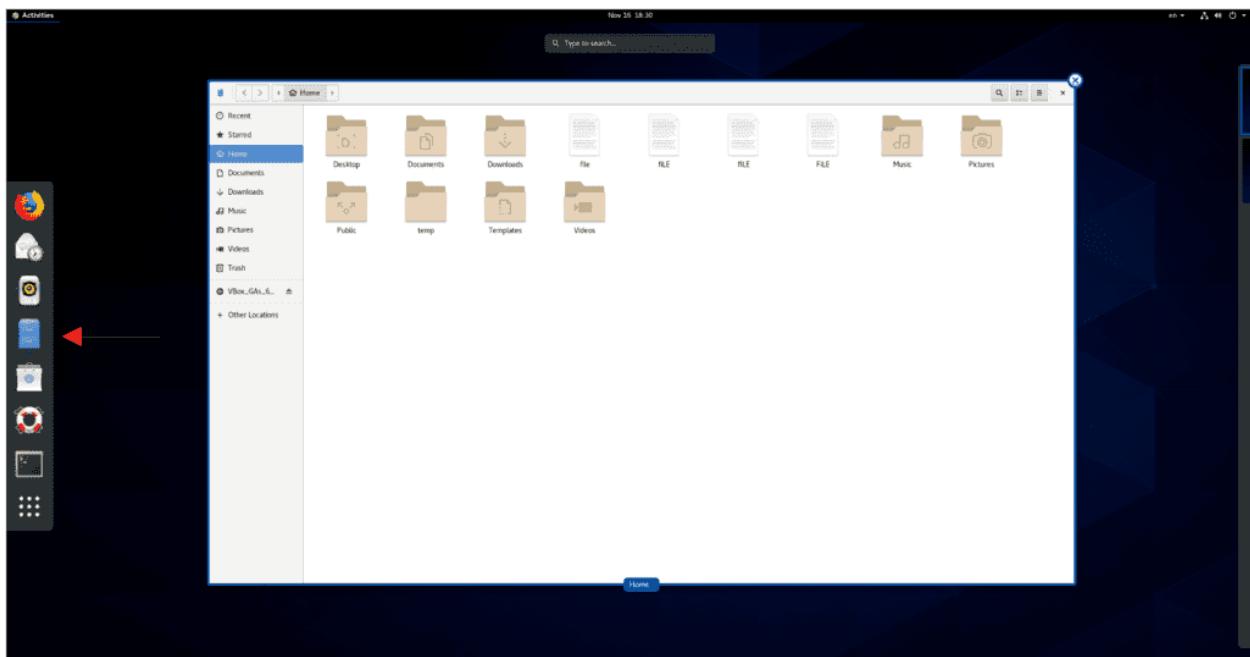
Вопросы

1. Для чего нужна командная строка?
2. Для чего нужен интерпретатор командной строки?
3. Какими способами можно получить доступ к текстовому интерфейсу?
4. С помощью какой команды можно посмотреть содержимое директории?
5. Что такое регистрозависимость?
6. Что такое ключ или опция команды?

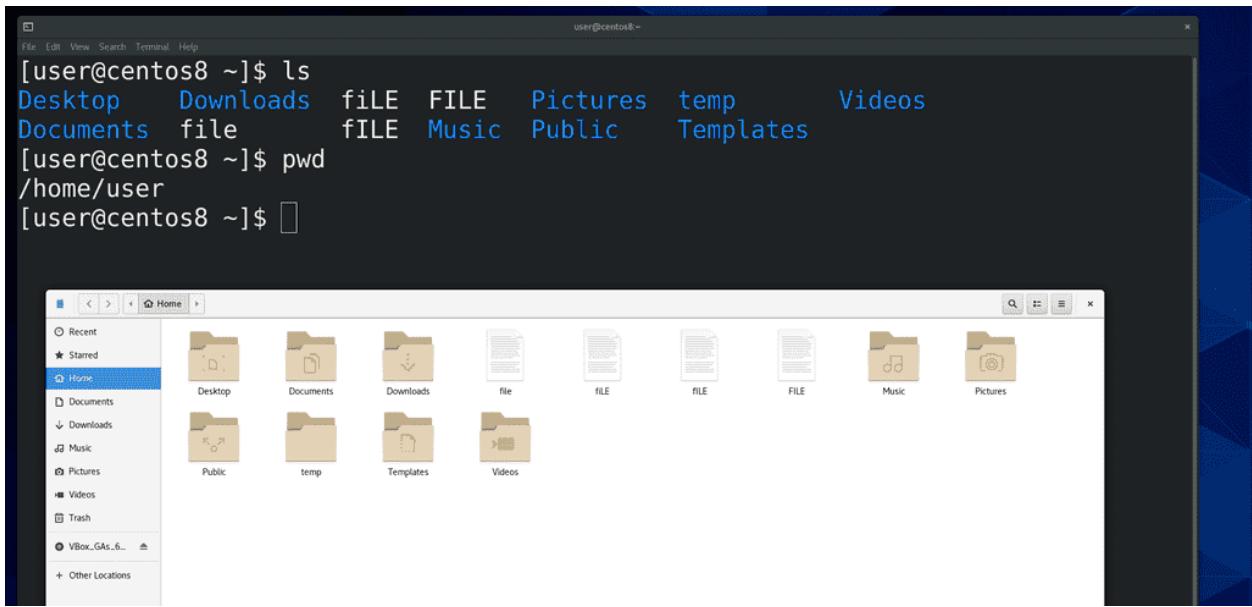
7. Какими способами можно посмотреть информацию о команде и список её ключей?
8. Как узнать номер иноды файла?
9. Как можно попытаться найти нужную команду, если нет интернета?
10. Какие главы документации есть и разница между ними на примере passwd?
11. Как посмотреть список команд, которые вы вводили?
12. Как запустить предыдущую команду?

2.6 06. Пути и директории.md

2.6.1 06. Пути и директории



Для работы с файлами и директориями в графическом интерфейсе есть программы, называемые файловыми менеджерами. У разных графических окружений могут быть разные программы, тот же GNOME, который стоит на CentOS, по умолчанию использует файловый менеджер под названием nautilus. Я думаю все знают, что можно делать с файлами и директориями – смотреть информацию о них, копировать, вырезать, удалять, переименовывать, создавать директории и всё такое. И в командной строке это также просто.

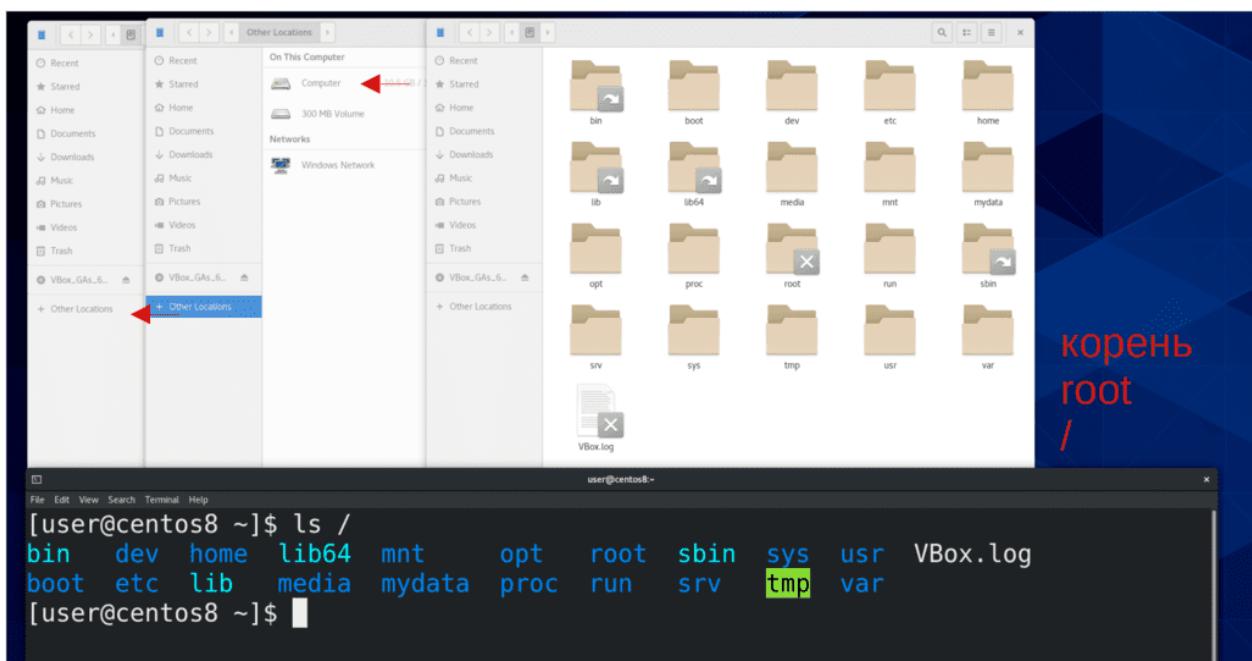


Мы знаем, что посмотреть содержимое директории можно с помощью команды ls. По умолчанию она показывает содержимое текущей директории. Заметьте, что ls показывает директории одним цветом, а файлы другим. Чтобы понять, в какой директории мы сейчас находимся, нужна команда pwd – print working directory.

`pwd`

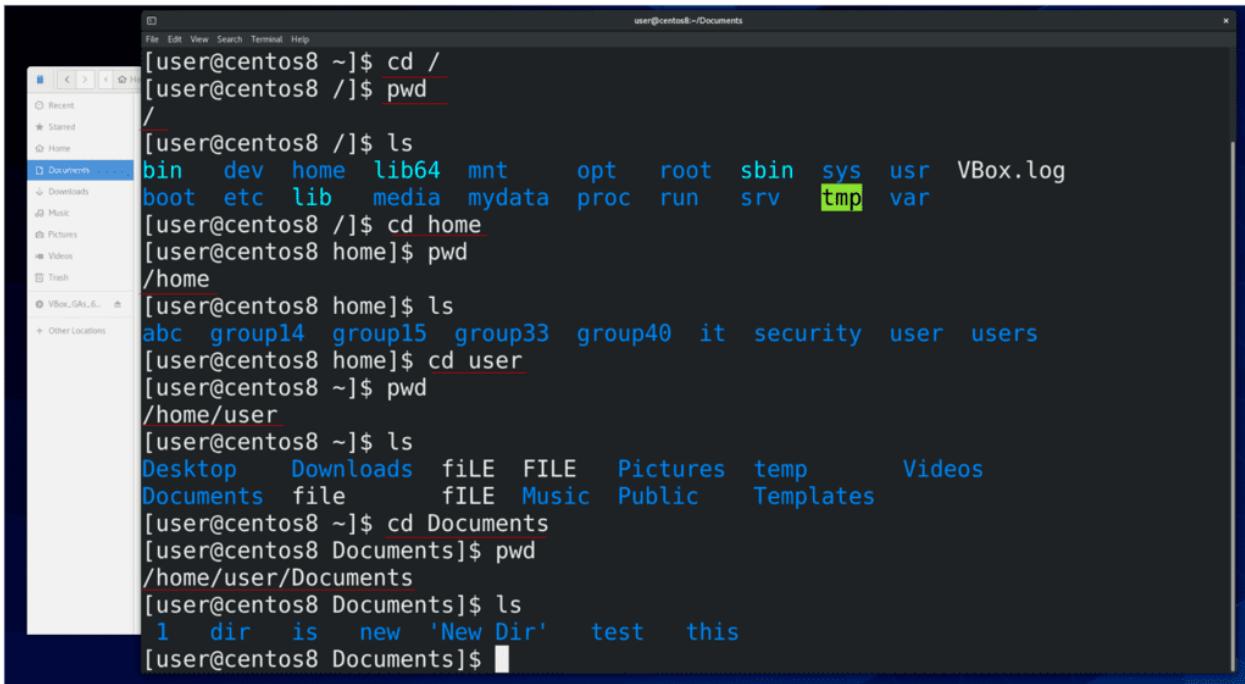
Когда вы открываете терминал, как правило он использует домашнюю директорию пользователя – это личная директория пользователя, где находятся все его файлы. С помощью ls можно смотреть содержимое и других директорий, для этого нужен путь к другой директории.

`ls /`



Кстати о путях. В корне находятся директории, внутри которых тоже есть какие-то директории. На-

пример, в корне у нас есть директория home, где обычно лежат домашние директории пользователей, например, нашего пользователя user. А внутри домашней директории пользователя есть какие-то его личные директории.



The screenshot shows a terminal window with a file browser interface on the left. The terminal window title is "user@centos8: ~/Documents". The terminal content shows a command-line session:

```
[user@centos8 ~]$ cd /
[user@centos8 /]$ pwd
/
[user@centos8 /]$ ls
bin dev home lib64 mnt opt root sbin sys usr VBox.log
boot etc lib media mydata proc run srv tmp var
[user@centos8 /]$ cd home
[user@centos8 home]$ pwd
/home
[user@centos8 home]$ ls
abc group14 group15 group33 group40 it security user users
[user@centos8 home]$ cd user
[user@centos8 ~]$ pwd
/home/user
[user@centos8 ~]$ ls
Desktop Downloads FILE FILE Pictures temp Videos
Documents file FILE Music Public Templates
[user@centos8 ~]$ cd Documents
[user@centos8 Documents]$ pwd
/home/user/Documents
[user@centos8 Documents]$ ls
1 dir is new 'New Dir' test this
[user@centos8 Documents]$
```

То есть, чтобы мне зайти, допустим, в директорию Documents у пользователя user, мне нужно зайти в корень, потом открыть home, user и зайти в директорию Documents. Чтобы переходить по директориям в командной строке, используем команду `cd` – change directory – сменить директорию. И так, пишем `cd /`, то есть заходим в корень (слеш – это путь к корню), проверяем текущую директорию (`pwd`), и смотрим её содержимое(`ls`). Дальше пишем `cd home` и повторяем всё тоже самое. Не стесняйтесь использовать `tab` – допустим, пишете `cd h`, нажимаете `tab` и он автоматом добавляет `ome`, так как никаких других директорий, начинающихся на `h` здесь нет. Точно также для `user` и `Documents`.

```
cd /
pwd
ls
cd home
pwd
ls
cd user
pwd
ls
cd Documents
pwd
ls
```

```
[user@centos8 Documents]$ cd /
[user@centos8 /]$ pwd
/
[user@centos8 /]$ cd /home/user/Documents
[user@centos8 Documents]$ pwd
/home/user/Documents
[user@centos8 Documents]$
```

Чтобы сразу зайти в Documents, мы можем разом написать `cd /home/user/Documents`, разделяя директории знаком слэш.

`cd /home/user/Documents/`

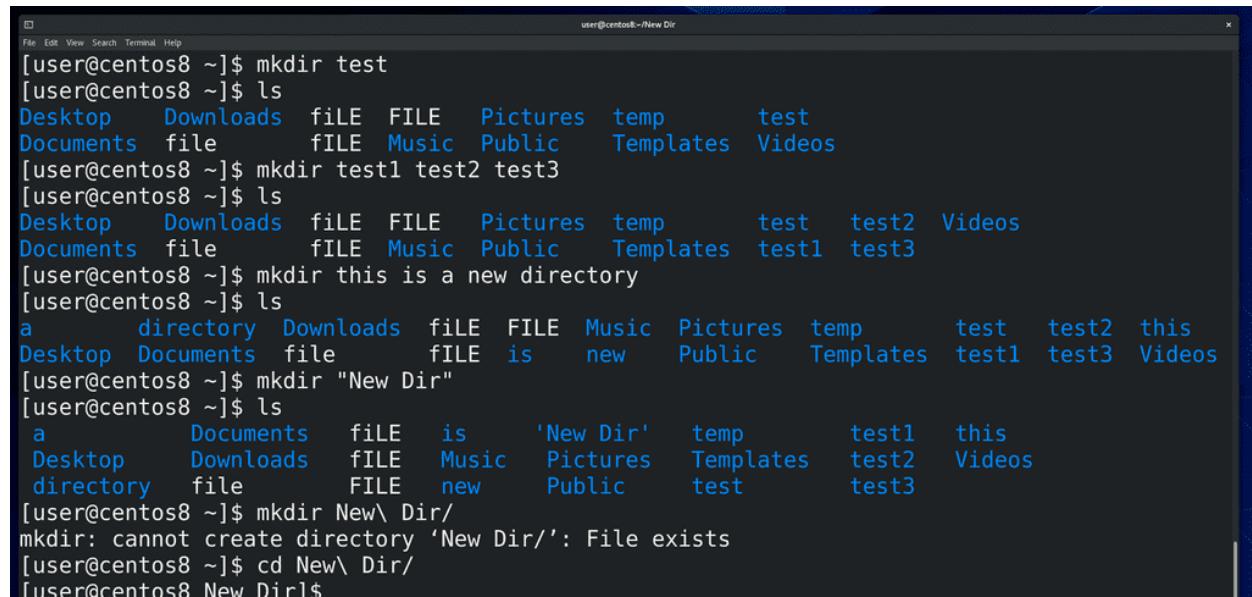
Когда мы в начале указываем корень (/), а потом пишем первую директорию, которая находится в корне, а потом то что внутри и т.д. - это полный путь. `/home/user/Documents` – пример полного пути. Он всегда начинается со знака / - то есть мы возвращаемся в самое начало файловой системы и оттуда начинаем писать путь. Такой путь универсальный - где бы вы сейчас не находились, вы можете указать полный путь и попасть туда куда вам надо.

```
[user@centos8 Documents]$ pwd
/home/user/Documents
[user@centos8 Documents]$ cd
[user@centos8 ~]$ pwd
/home/user
[user@centos8 ~]$ ls
Desktop Downloads file FILE Pictures temp Videos
Documents file FILE Music Public Templates
[user@centos8 ~]$ cd /home
[user@centos8 home]$ cd user/Documents
[user@centos8 Documents]$
[user@centos8 Documents]$ cd ..
[user@centos8 ~]$ cd Documents/
[user@centos8 Documents]$ cd ../Pictures/
[user@centos8 Pictures]$ cd -
/home/user/Documents
[user@centos8 Documents]$ cd .
[user@centos8 Documents]$ cd ~
[user@centos8 ~]$ cd ~/Documents/
[user@centos8 Documents]$ ls ~/Pictures/
'Screenshot from 2020-05-30 15-48-27.png' 'Screenshot from 2020-11-04 00-12-16.png'
'Screenshot from 2020-05-30 15-48-31.png' 'Screenshot from 2020-11-04 00-12-20.png'
'Screenshot from 2020-05-30 17-05-15.png' 'Screenshot from 2020-11-04 00-12-21.png'
```

В домашнюю директорию пользователя можно вернуться просто написав `cd`. Я вижу (`ls`), что здесь есть директория `Documents`. Если я хочу зайти в эту директорию, мне не обязательно писать полный путь – `cd /home/user/Documents`, я могу просто написать `cd Documents`. Или допустим, давайте зайдём в `cd /home/`. Я могу написать `cd user/Documents`. То есть, я строю путь не с корня, а с текущей директории. Такой путь называется относительным. Но что, если я нахожусь в директории `/home/user/Documents` и хочу попасть на директорию выше, то есть в `/home/user`? Для этого в каждой директории есть ссылки на вышестоящую директорию – две точки (...). Если я напишу `cd ..`, я попаду в `/home/user`. Я могу делать всякие комбинации, допустим, находясь в директории `Documents`, я могу написать `cd ../Pictures`. То есть, я разом вернулся в директорию `/home/user` и зашёл в `Pictures`. Чтобы сразу вернуться туда, где

я был раньше, я могу написать `cd -`. Также есть ссылка на текущую директорию – одна точка. Зачем это нужно мы разберём чуть позже. Также есть готовая ссылка на домашнюю директорию нашего пользователя – тильда (`~`). Где бы мы не находились, мы всегда можем написать `cd ~` и попасть в домашнюю директорию, либо использовать это для относительного пути, например `cd ~/Documents`. Ну и возвращаясь к теме `ls`, мы можем посмотреть содержимое любой директории, где бы мы сейчас не находились, используя полный или относительный пути. Допустим, `ls /home/user/Documents` или `ls Documents` или `ls ~/Pictures`.

```
cd  
pwd  
ls  
cd /home  
pwd  
cd user/Documents/  
cd ..  
cd Documents  
cd ../Pictures  
cd -  
cd .  
cd ~  
cd ~/Documents  
ls ~/Pictures
```



The screenshot shows a terminal window titled "user@centos8 ~ /New Dir". The user has run several commands:

```
[user@centos8 ~]$ mkdir test
[user@centos8 ~]$ ls
Desktop Downloads file FILE Pictures temp test
Documents file FILE Music Public Templates Videos
[user@centos8 ~]$ mkdir test1 test2 test3
[user@centos8 ~]$ ls
Desktop Downloads file FILE Pictures temp test test2 Videos
Documents file FILE Music Public Templates test1 test3
[user@centos8 ~]$ mkdir this is a new directory
[user@centos8 ~]$ ls
a directory Downloads file FILE Music Pictures temp test test2 this
Desktop Documents file FILE is new Public Templates test1 test3 Videos
[user@centos8 ~]$ mkdir "New Dir"
[user@centos8 ~]$ ls
a Documents file is 'New Dir' temp test1 this
Desktop Downloads file Music Pictures Templates test2 Videos
directory file FILE new Public test test3
[user@centos8 ~]$ mkdir New\ Dir/
mkdir: cannot create directory 'New Dir/': File exists
[user@centos8 ~]$ cd New\ Dir/
[user@centos8 New Dir]$
```

С путями разобрались. Теперь давайте посмотрим, как работать с директориями. Начнём с создания директории, для этого используем команду `mkdir` – make directory – создать директорию. В качестве аргумента мы должны указать название новой директории – допустим:

```
mkdir test
ls
```

Можно разом создать несколько директорий:

```
mkdir test1 test2 test3
ls
```

Очень не рекомендую использовать пробелы в названиях, так как командная строка воспринимает

пробелы как переход на новый аргумент или опцию. Как я показал выше, если я напишу:

```
mkdir this is a new directory
ls
```

у меня создастся пять директорий. Но если всё же вам необходимо создать директорию с пробелами в названии, вы можете взять название новой директории в кавычки:

```
mkdir "New Dir"
```

Теперь, чтобы зайти в эту директорию, нужно либо использовать кавычки, либо использовать специальный символ - \ (обратный слэш) перед пробелом.

```
cd "New Dir"
cd ..
cd New\ Dir
```

Это называется экранированием символов. В данном случае это позволит консоли игнорировать пробел как переход на новый аргумент.



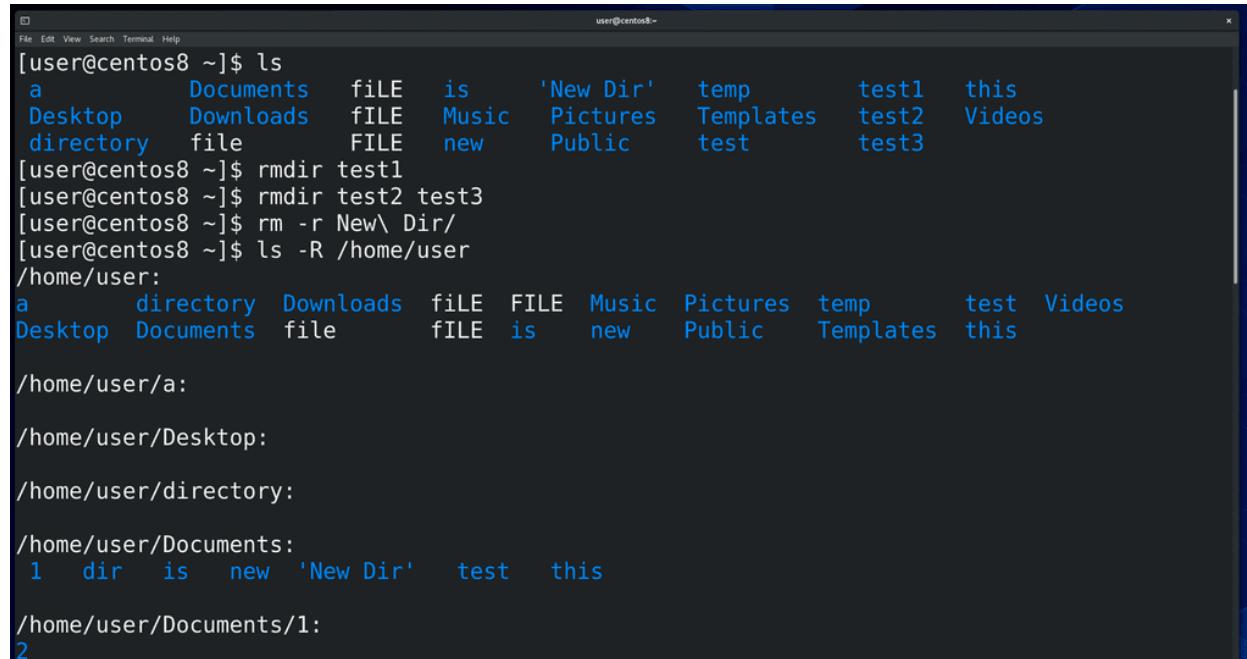
The screenshot shows a terminal window with the following session:

```
[user@centos8 New Dir]$ mkdir -p dir1/dir2/dir3
[user@centos8 New Dir]$ ls
dir1
[user@centos8 New Dir]$ cd dir1/
[user@centos8 dir1]$ ls
dir2
[user@centos8 dir1]$ cd dir2
[user@centos8 dir2]$ ls
dir3
[user@centos8 dir2]$ cd dir3
[user@centos8 dir3]$ ls
[user@centos8 dir3]$
```

Если я хочу создать сразу директорию 1, а внутри неё директорию 2, а внутри неё директорию 3, для этого нужна опция -p – parents – создавать родительские директории. То есть, написав:

```
mkdir -p dir1/dir2/dir3
```

я создам несколько вложенных директорий.



```
[user@centos8 ~]$ ls
a      Documents  file  is  'New Dir'  temp  test1  this
Desktop  Downloads  FILE  Music  Pictures  Templates  test2  Videos
directory  file    FILE  new   Public    test      test3

[user@centos8 ~]$ rmdir test1
[user@centos8 ~]$ rmdir test2 test3
[user@centos8 ~]$ rm -r New\ Dir/
[user@centos8 ~]$ ls -R /home/user
/home/user:
a      directory  Downloads  file  FILE  Music  Pictures  temp  test  Videos
Desktop  Documents  file    FILE  is    new   Public    Templates  this

/home/user/a:

/home/user/Desktop:

/home/user/directory:

/home/user/Documents:
1  dir  is  new  'New Dir'  test  this

/home/user/Documents/1:
2
```

Для удаления пустой директории используется команда `rmdir – remove directory` – удалить директорию. Например:

```
rmdir test1
```

Можно разом удалять несколько директорий:

```
rmdir test2 test3
```

Если директория не пустая, то `rmdir` откажется её удалять. Поэтому, когда вам нужно удалить директорию со всем содержимым, используется команда `rm -r`. Например:

```
rm -r New\ Dir
```

Тогда я удалю как `New Dir`, так и директории `dir1`, `dir2` и `dir3`. Вообще, `rm` используется для удаления файлов, но о файлах мы поговорим в следующий раз. А опцию `-r` вы часто будете встречать – она означает рекурсивно, то есть со всем, что находится внутри. Допустим, та же команда `ls` уже с большим `-R` покажет содержимое директории вместе с содержимым поддиректорий.

```
ls -R /home/user
```

Осталось ещё рассмотреть копирование, перемещение и прочее, но так эти команды совпадают с командами по работе с файлами, а тема уже получилась достаточно большая, оставшееся мы рассмотрим в следующий раз. Чтобы не забывать пройденное, обязательно практикуйтесь. Например, найдите в ролике все команды с опциями, которые я вводил, выпишите, повторите на различных директориях. Добейтесь того, чтобы вы знали без всяких раздумий, для чего нужна каждая из этих команд и ключей.

2.6.2 Практика

Вопросы

- С помощью какой программы можно работать с файлами в графическом интерфейсе?
- Как узнать текущую директорию в командной строке?

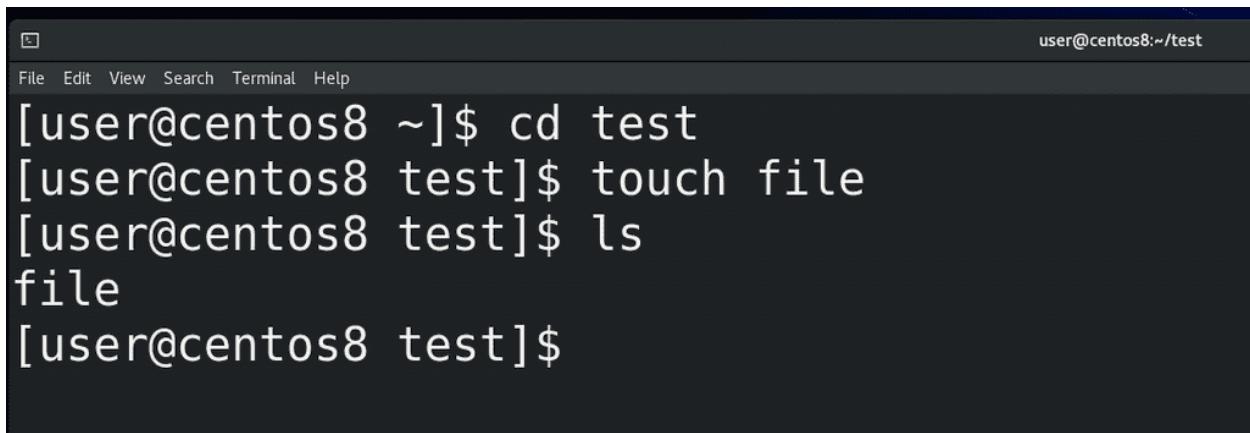
3. Что удобнее - полный или относительный путь?
4. Как сделать так, чтобы не приходилось писать целиком команду или путь?
5. Какими способами можно вернуться в домашнюю директорию пользователя?

Задания

1. Создайте директорию „New Dir“ с тремя пробелами в названии, не используя кавычки.
2. Разом создайте директорию dira со вложенной директорией dirb.
3. Находясь в домашней директории, удалите директорию dirb, при этом не удалив директорию dira.
4. Перечислите и попробуйте все способы зайти в директорию /usr/share/doc/man-pages, учитывая, что вы находитесь в домашней директории пользователя (1) или в директории /etc/ (2).
5. Скопируйте директорию Music в графическом интерфейсе (Files), вставьте в ту же директорию (название копии должно быть Music (copy)) и попробуйте все способы зайти в новую директорию.
6. Переименуйте директорию Music (copy) и добавьте перед скобкой несколько пробелов. Попробуйте все способы зайти в эту директорию.
7. Зайдите в директорию /usr/share/applications . Затем зайдите в директорию /var/log/chrony. Перечислите и попробуйте все способы перемещаться между этими двумя директориями.
8. Находясь в домашней директории своего пользователя, посмотрите содержимое директории /usr/bin/ используя полный и относительный пути.
9. Находясь в домашней директории своего пользователя создайте директорию /tmp/testdir используя полный и относительный пути.
10. Находясь в директории /usr/share/applications создайте одной командой директорию /home/user/testdir1/testdir11/testdir111 и посмотрите разом содержимое всех поддиректорий директории /home/user/testdir1
11. Находясь в директории /var/log создайте одной командой директории /home/user/testdir2/testdir22 и /tmp/testdir2/testdir22, а затем посмотрите одной командой содержимое директорий /home/user/testdir2 и /tmp/testdir2/testdir22
12. Находясь в директории /tmp используйте одну команду и относительные пути (используя знак ~), чтобы удалить директории /home/user/testdir1 и /tmp/testdir2/testdir22 с выводом информации об удалении.

2.7 07. Создание и копирование файлов

2.7.1 07. Создание и копирование файлов

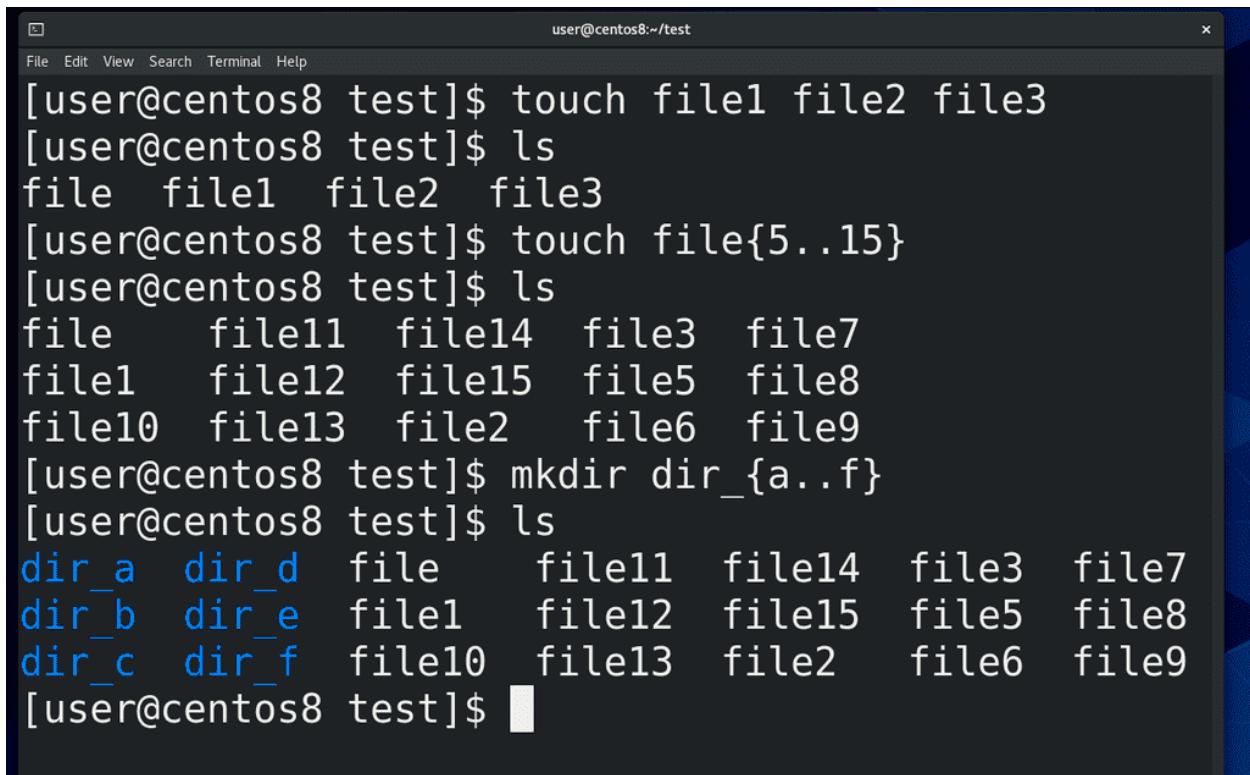


```
[user@centos8 ~]$ cd test
[user@centos8 test]$ touch file
[user@centos8 test]$ ls
file
[user@centos8 test]$
```

Чтобы научиться работать с файлами, нам понадобятся тестовые файлы. Их мы можем создать с помощью команды `touch filename`. Вообще, основная задача `touch` – обновить время доступа к файлу, поэтому и такое название – потрогать. Если файл существует, применив `touch` на него ничего с ним не случится, только обновится информация в inode, а если файла не существует – он создастся.

`touch file`

Это может быть полезно в каких-нибудь скриптах, но и сейчас это помогает нам создать тестовые файлы.



```
[user@centos8 test]$ touch file1 file2 file3
[user@centos8 test]$ ls
file  file1  file2  file3
[user@centos8 test]$ touch file{5..15}
[user@centos8 test]$ ls
file    file11  file14  file3  file7
file1   file12  file15  file5  file8
file10  file13  file2   file6  file9
[user@centos8 test]$ mkdir dir_{a..f}
[user@centos8 test]$ ls
dir_a  dir_d  file    file11  file14  file3  file7
dir_b  dir_e  file1   file12  file15  file5  file8
dir_c  dir_f  file10  file13  file2   file6  file9
[user@centos8 test]$
```

С помощью `touch` можно разом создать несколько файлов:

```
touch file1 file2 file3
```

Одна из фишек командной строки – возможность работать с регулярными выражениями. Например, мы можем написать:

```
touch file{5..15}
```

и баш преобразит это в:

```
touch file5 file6 file7 ...
```

и т.д., в итоге мы получим кучу файлов. Тоже самое работает и с буквами, допустим, тот же:

```
mkdir dir_{a..f}
```

Но о регулярных выражениях мы еще поговорим.

```
[user@centos8 ~]$ touch .file20
[user@centos8 ~]$ mkdir .dir_h
[user@centos8 ~]$ ls
dir_a  dir_d  file   file11  file14  file3  file7
dir_b  dir_e  file1  file12  file15  file5  file8
dir_c  dir_f  file10 file13  file2   file6  file9
[user@centos8 ~]$ ls -a
.      dir_d  file1  file14  file5
..     dir_e  file10 file15  file6
dir_a  dir_f  file11 file2   file7
dir_b  .dir_h file12 .file20 file8
dir_c  file   file13 file3   file9
[user@centos8 ~]$ ls -a ~
.          .local
..         .mozilla
a          Music
.bash_history .nanorc
.bash_logout  new
.bash_profile Pictures
.bashrc       .pki
.cache       Public
```

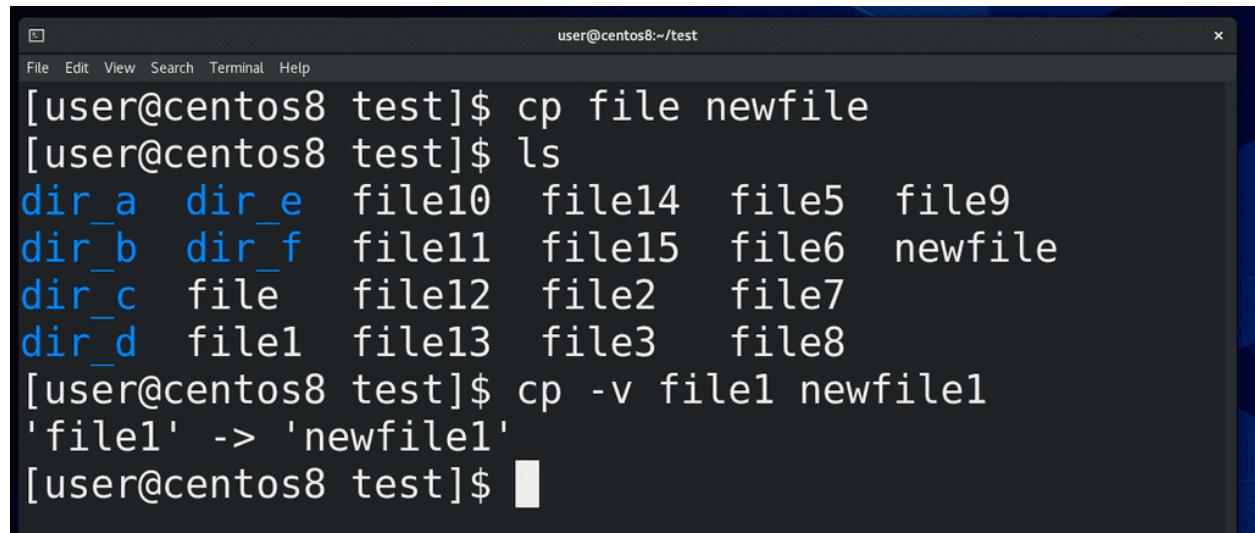
Также можно создавать невидимые файлы и директории. Для этого в начале имени файла или директории следует указывать точку. Например:

```
touch .file20  
mkdir .dir_h
```

Если посмотреть на вывод ls, скрытых файлов и директорий не увидеть. Чтобы их увидеть, для ls нужен ключ -a:

```
ls -a
```

Тогда мы увидим все файлы в директории. Для того, чтобы увидеть скрытые файлы в файловом менеджере, можно нажать Ctrl + h. Тоже самое, чтобы скрыть. Обычно скрытые файлы нужны всяким программам для хранения настроек в домашней директории пользователя, но не только. В целом это позволяет скрыть ненужные пользователю файлы и директории.



A screenshot of a terminal window titled "user@centos8:~/test". The window shows a command-line session. The user runs "cp file newfile", then "ls" to list files, which includes ".dir_a", ".dir_e", "file10", "file14", "file5", "file9", ".dir_b", ".dir_f", "file11", "file15", "file6", "newfile", ".dir_c", "file", "file12", "file2", "file7", ".dir_d", "file1", "file13", "file3", "file8". Finally, the user runs "cp -v file1 newfile1", which outputs "'file1' -> 'newfile1'".

```
[user@centos8 test]$ cp file newfile  
[user@centos8 test]$ ls  
.dir_a .dir_e file10 file14 file5 file9  
.dir_b .dir_f file11 file15 file6 newfile  
.dir_c file file12 file2 file7  
.dir_d file1 file13 file3 file8  
[user@centos8 test]$ cp -v file1 newfile1  
'file1' -> 'newfile1'  
[user@centos8 test]$
```

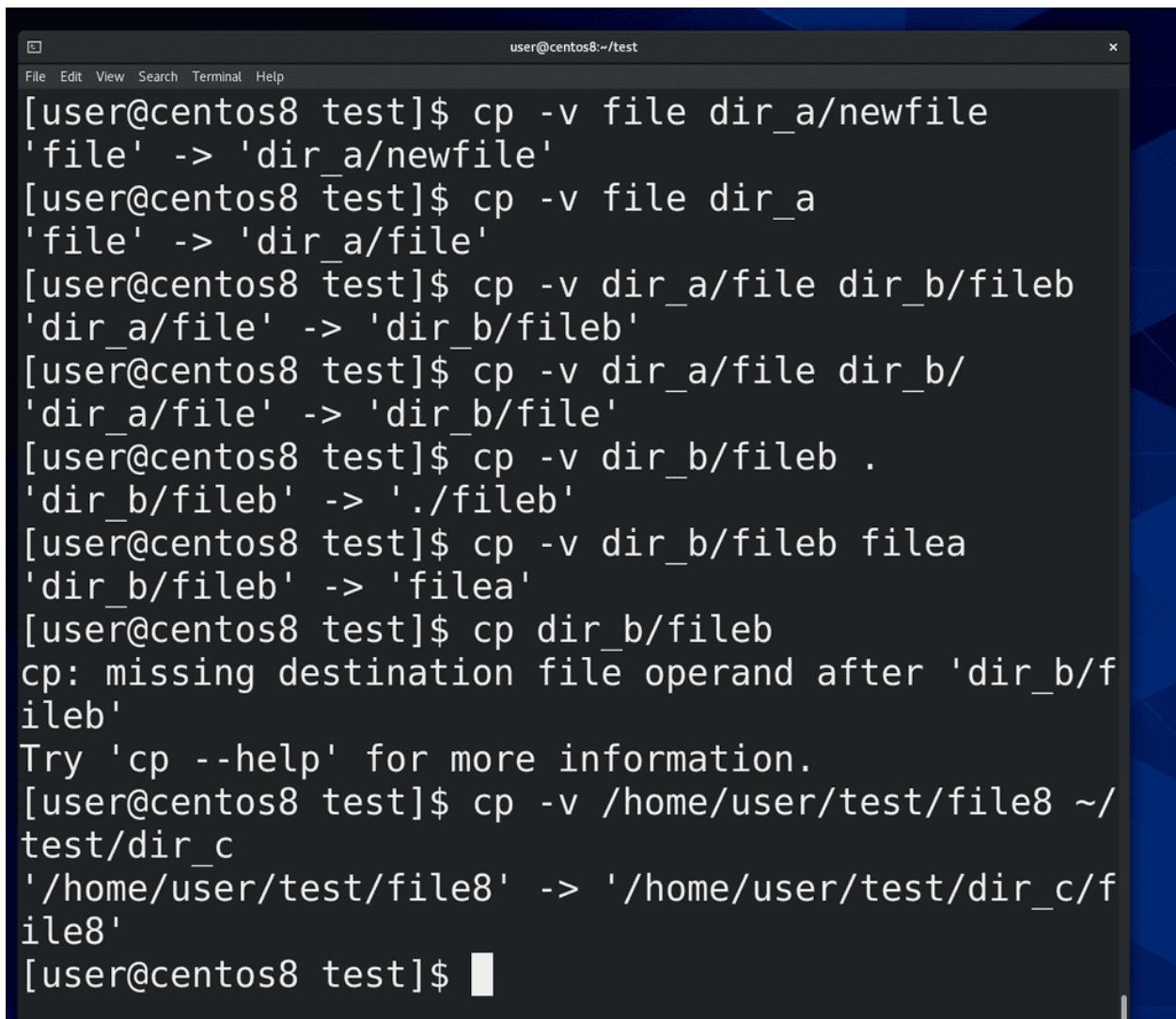
И так, что можно делать с файлами? Начнём с копирования. Для этого используется команда cp – сору. Чтобы скопировать файл в текущую директорию, используем cp имя нужного файла и имя для копии:

```
cp file newfile
```

Команда что-то сделала, а чтобы увидеть, что произошло, нужно посмотреть (ls). Многие команды сами могут показывать, что происходит во время работы. Обычно для этого используется опция verbose (подробно) - вы часто будете натыкаться на эту опцию. Обычно она пишется как -v. То есть:

```
cp -v file1 newfile1
```

Тут мы видим, что файл скопировался.



The screenshot shows a terminal window titled "user@centos8:~/test". The terminal displays the following cp command examples:

```
[user@centos8 test]$ cp -v file dir_a/newfile
'file' -> 'dir_a/newfile'
[user@centos8 test]$ cp -v file dir_a
'file' -> 'dir_a/file'
[user@centos8 test]$ cp -v dir_a/file dir_b/fileb
'dir_a/file' -> 'dir_b/fileb'
[user@centos8 test]$ cp -v dir_a/file dir_b/
'dir_a/file' -> 'dir_b/file'
[user@centos8 test]$ cp -v dir_b/fileb .
'dir_b/fileb' -> './fileb'
[user@centos8 test]$ cp -v dir_b/fileb filea
'dir_b/fileb' -> 'filea'
[user@centos8 test]$ cp dir_b/fileb
cp: missing destination file operand after 'dir_b/f
ileb'
Try 'cp --help' for more information.
[user@centos8 test]$ cp -v /home/user/test/file8 ~/test/dir_c
'/home/user/test/file8' -> '/home/user/test/dir_c/f
ile8'
[user@centos8 test]$
```

Чтобы скопировать файл в директорию, используем:

```
cp -v file dir_a/newfile
```

Но вообще, если копировать в другую директорию, то необязательно указывать новое имя, можно просто написать:

```
cp -v file dir_a
```

Можно копировать из одной директории в другую:

```
cp -v dir_a/file dir_b/fileb
```

либо также не указывая нового имени:

```
cp -v dir_a/file dir_b
```

Ещё можно копировать из другой директории в текущую, используя ссылку для текущей директории – точку, как я говорил в прошлый раз. То есть:

```
cp -v dir_b/fileb .
```

если вы не хотите менять имя, либо, если вы хотите новое имя, то достаточно просто указать его:

```
cp -v dir_b/fileb filea
```

В общем, идея какая – для текущей директории путь не нужен, но нельзя просто написать:

```
cp dir_b/fileb
```

потому что синтаксис требует указать, куда файл нужно копировать. Если это не сделать, команда подумает, что недописана и просто выдаст ошибку. Ну и до этого мы использовали относительные пути, а так мы можем работать и с полными путями, например:

```
cp -v /home/user/test/file8 ~/temp/dir_c
```

```
[user@centos8 test]$ cp -v .file20 dir_c  
'file20' -> 'dir_c/.file20'  
[user@centos8 test]$ cp -v file6 file7 file8 dir_d  
'file6' -> 'dir_d/file6'  
[user@centos8 test]$ cp -v dir_a/file dir_b/fileb dir_d/file8 dir_e  
'dir_a/file' -> 'dir_e/file'  
'dir_b/fileb' -> 'dir_e/fileb'  
'dir_d/file8' -> 'dir_e/file8'  
[user@centos8 test]$ cp -v file{10..13} dir_e  
'file10' -> 'dir_e/file10'  
'file11' -> 'dir_e/file11'  
'file12' -> 'dir_e/file12'  
'file13' -> 'dir_e/file13'  
[user@centos8 test]$ cp -v dir_d/* dir_f  
'dir_d/file6' -> 'dir_f/file6'  
'dir_d/file7' -> 'dir_f/file7'  
'dir_d/file8' -> 'dir_f/file8'  
[user@centos8 test]$ cp -v file1* dir_e  
'file1' -> 'dir_e/file1'  
'file10' -> 'dir_e/file10'  
'file11' -> 'dir_e/file11'  
'file12' -> 'dir_e/file12'  
'file13' -> 'dir_e/file13'  
'file14' -> 'dir_e/file14'
```

Копирование скрытых файлов ничем не отличается от копирования обычных файлов – просто не забывайте указывать точку в начале названия файла, потому что это часть названия файла:

```
cp -v .file20 dir_c
```

Можно копировать несколько файлов разом, но тогда нужно копировать их в другую директорию:

```
cp -v file6 file7 file8 dir_d
```

Ну и по той же логике, указывая пути:

```
cp -v dir_a/file dir_b/fileb dir_d/file8 dir_e
```

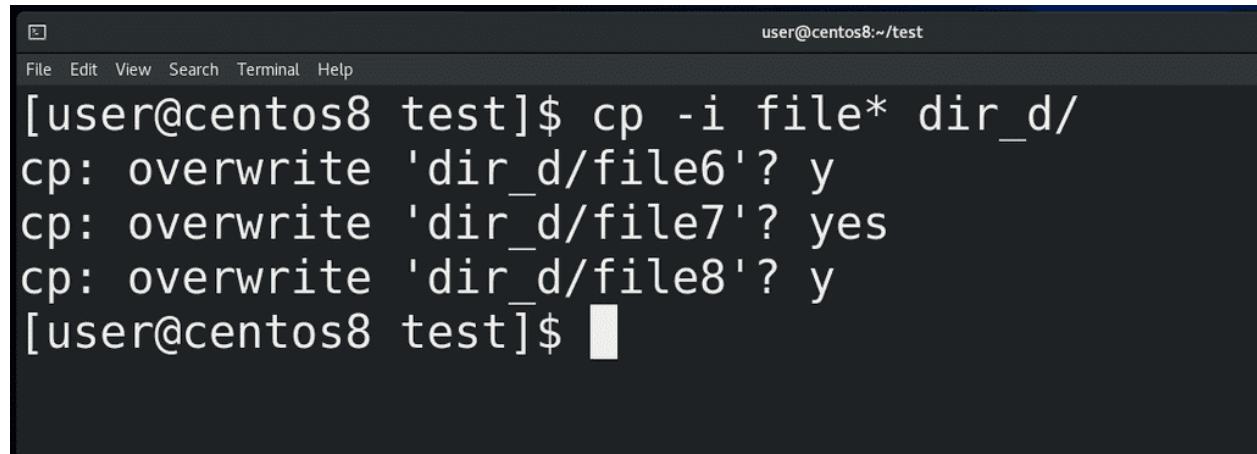
Помните про регулярные выражения? Можем использовать и здесь. Допустим:

```
cp -v file{10..13} dir_e
```

Кроме фигурных скобок, можно использовать и звёздочку – её часто называют wildcard – она означает, что подойдёт любое значение. Допустим, в директории dir_d есть три файла и я хочу скопировать все три:

```
cp -v dir_d/* dir_f
```

Тогда все файлы скопируются. Или, допустим, я хочу скопировать все файлы, названия которых начинаются с file1, а сюда подходят файл file1, file10, file11, file12 и т.д. Я пишу cp -v file1* dir_e и все подходящие файлы скопируются.

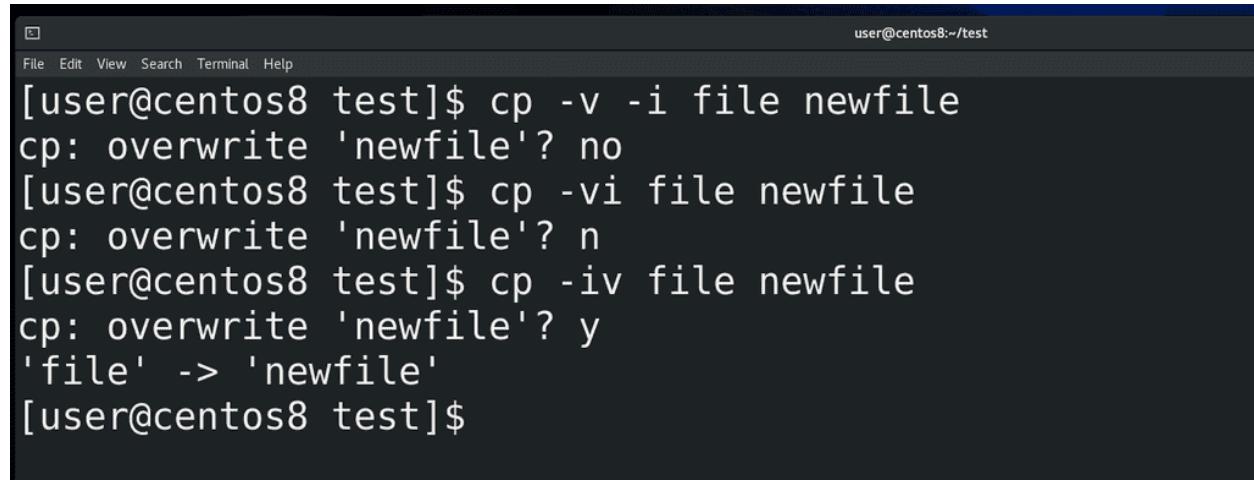


The screenshot shows a terminal window with a dark background and light-colored text. At the top, it says "user@centos8:~/test". The command entered is "[user@centos8 test]\$ cp -i file* dir_d/". The terminal then displays three prompts for overwriting files: "cp: overwrite 'dir_d/file6'? y", "cp: overwrite 'dir_d/file7'? yes", and "cp: overwrite 'dir_d/file8'? y". Finally, the command is completed with "[user@centos8 test]\$".

Давайте рассмотрим ещё пару ключей. Допустим, ключ **-i** – интерактивно. Что это значит? Когда вы копируете файл, если существует файл с таким названием в директории назначения, этот файл просто перезапишется. Это не всегда нас устраивает, иногда мы хотим лично решать, что перезаписывать, а что нет. Поэтому здесь нам поможет ключ **-i**. Допустим, скопируем все файлы в директорию dir_d:

```
cp -i file* dir_d
```

Командная строка начнёт спрашивать, а что же делать с файлами, которые существуют там с тем же именем. Мы можем отказаться от перезаписывания файла, либо согласится. Обычно когда какая-то команда спрашивает вопрос, на который нужно ответить да/нет, то подходят ответы **y**, либо **yes**, **n**, либо **no**. Такой вариант – интерактивный – требует, чтобы вы лично сидели и решали, что делать с каждым совпадающим файлом.

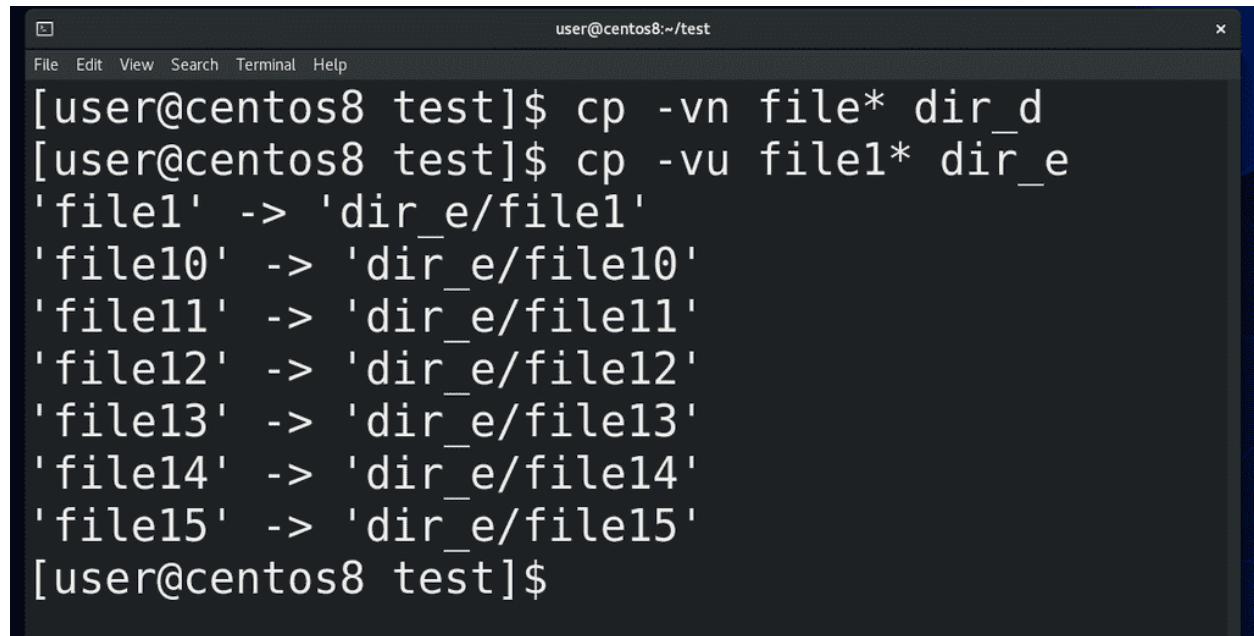


```
[user@centos8 test]$ cp -v -i file newfile
cp: overwrite 'newfile'? no
[user@centos8 test]$ cp -vi file newfile
cp: overwrite 'newfile'? n
[user@centos8 test]$ cp -iv file newfile
cp: overwrite 'newfile'? y
'file' -> 'newfile'
[user@centos8 test]$
```

Кстати, ключи можно использовать вместе. Допустим, мы знаем ключ `-v` и ключ `-i`. Мы можем использовать их вместе:

```
cp -v -i file newfile
cp -iv file newfile
cp -vi file newfile
```

Порядок – какой ключ вначале, какой потом – почти никогда не имеет значения.



```
[user@centos8 test]$ cp -vn file* dir_d
[user@centos8 test]$ cp -vu file1* dir_e
'file1' -> 'dir_e/file1'
'file10' -> 'dir_e/file10'
'file11' -> 'dir_e/file11'
'file12' -> 'dir_e/file12'
'file13' -> 'dir_e/file13'
'file14' -> 'dir_e/file14'
'file15' -> 'dir_e/file15'
[user@centos8 test]$
```

Другой вариант – мы не хотим перезаписывать файлы – тогда сразу используем опцию `-n`. Например:

```
cp -vn file* dir_d
```

Тогда файлы в директории не перезапишутся. А если, допустим, мы хотим пропустить совпадающие файлы, а скопировать только файлы новее или отсутствующие файлы – тогда опция `-u` – update. То есть:

```
cp -vu file1* dir_e.
```

```
[user@centos8 test]$ cp -vl file filelink
'file' -> 'filelink'
[user@centos8 test]$
```

Помните про жёсткие ссылки? Мы можем вместо копирования создать жёсткую ссылку. Например:

```
cp -vl file filelink
```

```
[user@centos8 test]$ cp -vr dir_a dir_b
'dir_a' -> 'dir_b/dir_a'
'dir_a/newfile' -> 'dir_b/dir_a/newfile'
'dir_a/file' -> 'dir_b/dir_a/file'
[user@centos8 test]$ cp -va dir_b dir_c
'dir_b' -> 'dir_c/dir_b'
'dir_b/fileb' -> 'dir_c/dir_b/fileb'
'dir_b/file' -> 'dir_c/dir_b/file'
'dir_b/dir_a' -> 'dir_c/dir_b/dir_a'
'dir_b/dir_a/newfile' -> 'dir_c/dir_b/dir_a/newfile'
'dir_b/dir_a/file' -> 'dir_c/dir_b/dir_a/file'
[user@centos8 test]$
```

Также мы можем копировать директории. Для этого используем опцию **-r** – рекурсивно. Допустим, я хочу скопировать директорию `dir_a` в директорию `dir_b`:

```
cp -vr dir_a dir_b
```

Эту опцию можно использовать не только с директориями, но и с файлами, никому от этого хуже не станет. Забегая вперёд, скажу, что в большинстве случаев правильнее копировать с опцией **-a** вместо **-r**, по сути **-a** это опции **-r** и **-d**. Вкратце, это позволяет сохранить права и владельца файла у копии файла.

```
cp -va dir_a dir_b
```

Как-то получилось, что я очень много времени уделил на копирование и лучше пока не перегружать вас информацией о других командах. Но то что мы сделали сегодня с одной командой, актуально и для большинства других команд – во многом ключи и подходы похожи.

2.7.2 Практика

Вопросы

1. Что делает команда touch?
2. Чем скрытые файлы отличаются от обычных?
3. Для чего нужен ключ -v (verbose)?
4. Какой синтаксис у команды cp, если нужно скопировать несколько файлов и директорий?
5. Зачем нужен ключ -i у команды cp?
6. Как можно использовать несколько ключей одновременно?
7. Как можно использовать * и {} при работе с файлами?

Задания

1. Создайте 5 файлов 5 командами(1), одной командой (2) и одной командой с помощью регулярных выражений (3).
2. Создайте скрытую директорию и скрытый файл внутри этой директории.
3. Одной командой создайте 3 скрытые директории(dira,dirb,dirc), так, чтобы 2 директории(dirb,dirc) были вложены в первую(dira).
4. Создайте скрытые копии файлов в скрытой директории
5. Скопируйте скрытые копии файлов из скрытой директории в текущую директорию так, чтобы они перестали быть скрытыми.
6. Создайте жёсткую ссылку на файл с помощью команды cp.
7. Скопируйте скрытую директорию dira в новую директорию dird.

2.8 08. Перемещение, переименование, удаление. Жёсткие и символические ссылки

2.8.1 08. Перемещение, переименование, удаление. Жёсткие и символические ссылки



The screenshot shows a terminal window titled "user@centos8:~/test". The command entered is "[user@centos8 test]\$ mv -v file dir_a". The output shows the file being renamed: "renamed 'file' -> 'dir_a/file'". The terminal prompt "[user@centos8 test]\$ " is visible at the bottom.

Осталось разобрать ещё пару операций – перемещение, переименование и удаление. В отличии от копирования, при перемещении в рамках одной файловой системы нам не нужно создавать копию данных, нам просто нужно изменить путь к этому файлу. Для этого не нужно затрагивать сами данные, достаточно просто создать жёсткую ссылку в новом месте и удалить в старом. За это отвечает команда move – mv. Синтаксис команды и ключи во многом сходятся с ключами команды copy (cp), та же интерактивность с ключом -i, тот же update и всё такое. Для примера, чтобы переместить файл в директорию, пишем:

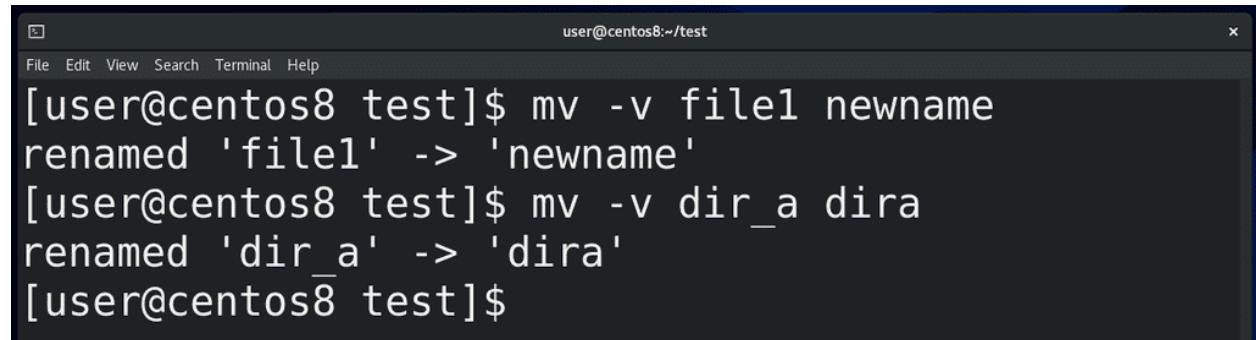
```
mv -v file15 dir_a
```

Но, в отличии от сору, при перемещении директории не нужно указывать опцию рекурсивно (-r). То есть достаточно написать:

```
mv dir_a dira
```

Вы можете спросить, а почему mv для директорий не нужна рекурсия? Всё дело в том, что директория – это специальный файл, который можно представить в виде листа. А файлы внутри директории – это просто строчки на этом листе (жёсткие ссылки), в которых указано имя файла и номер иноды. Внутри директории (листа) есть список жёстких ссылок (файлов) и две специальные жёсткие ссылки – на себя и на директорию выше – точка и две точки соответственно. То есть перемещая директорию вы просто перемещаете лист, при этом вам не нужно ничего делать с файлами, указанными на листе. В отличии от копирования, когда вы создаёте новую директорию(лист), а также создаёте новые файлы и, соответственно, новые жёсткие ссылки.

Но всё вышесказанное работает в рамках одной файловой системы, так как жёсткие ссылки – это часть файловой системы. Они содержат номер иноды, а не данные. Поэтому перемещение из одной файловой системы в другую – это просто копирование данных с удалением того, что было. Хотя это делается той же командой mv.



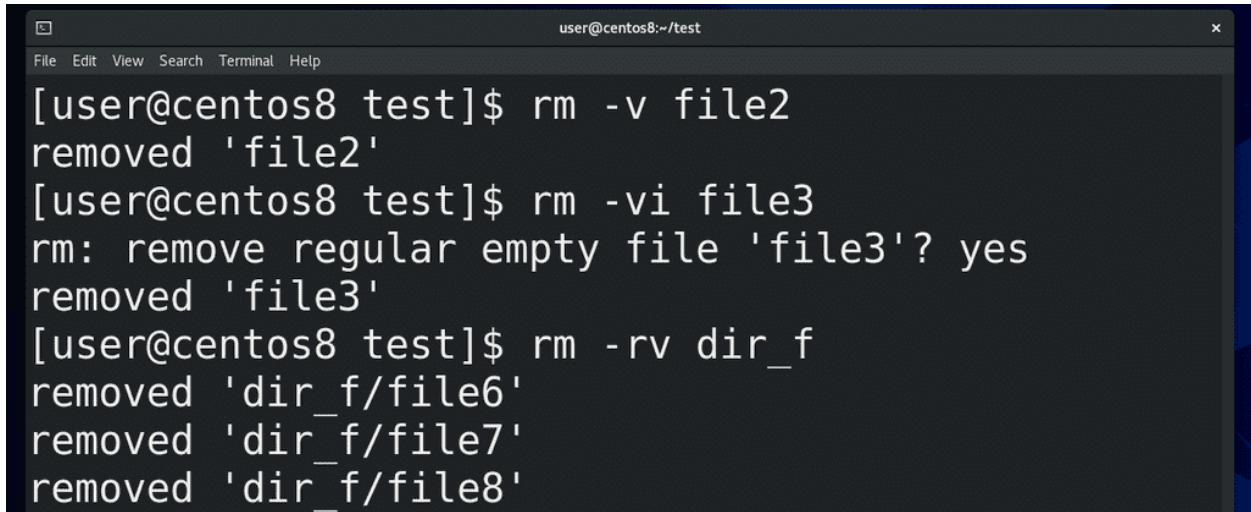
The screenshot shows a terminal window titled "user@centos8:~/test". The user has run two commands:

```
[user@centos8 test]$ mv -v file1 newname
renamed 'file1' -> 'newname'
[user@centos8 test]$ mv -v dir_a dira
renamed 'dir_a' -> 'dира'
```

The terminal window has a dark background with white text. The title bar and menu bar are visible at the top.

Теперь про переименование. Помните, имя файла – это жёсткая ссылка, которая содержит собственно имя и номер иноды? Собственно, поменять имя – это создать жёсткую ссылку с новым именем и удалить со старым. Именно этим у нас занимается команда mv. То есть, если вы хотите переименовать файл или директорию, достаточно написать:

```
mv -v file1 newname
mv -v dir_a dira
```



```
[user@centos8 test]$ rm -v file2
removed 'file2'
[user@centos8 test]$ rm -vi file3
rm: remove regular empty file 'file3'? yes
removed 'file3'
[user@centos8 test]$ rm -rv dir_f
removed 'dir_f/file6'
removed 'dir_f/file7'
removed 'dir_f/file8'
```

Что касается удаления, как думаете, с чем оно работает? Да, опять же, с жёсткими ссылками. Для удаления используется команда remove – rm. Например:

```
rm -v file2
```

Опции во многом совпадают с предыдущими командами – та же интерактивность с ключом -i, та же рекурсия, если мы удаляем целую директорию со всем содержимым.



```
[user@centos8 test]$ rm -rv file10 file11 dira dir_b
removed 'file10'
removed 'file11'
removed 'dira/newfile'
removed 'dira/file'
removed directory 'dira'
removed 'dir_b/fileb'
removed 'dir_b/file'
removed 'dir_b/dir_a/newfile'
```

Можем разом удалить несколько файлов и директорий:

```
rm -rv file1 file2 dir1 dir2
```

В общем, всё как при копировании. Ну или допустим, если хотим удалить всё в текущей директории:

```
rm -r *
```

Кстати, одна из самых «мемных» команд:

```
rm -rf /
```

удаляет всё в корне, включая все поддиректории, то есть все файлы в системе. Поэтому будьте внимательны, когда выполняете команды.

```
[user@centos8 test]$ cp -vl filea filealink
'filea' -> 'filealink'
[user@centos8 test]$ ln -v filea samefile
'samefile' => 'filea'
[user@centos8 test]$ rm -v filealink
removed 'filealink'
```

Помните, я говорил, что файл существует, пока есть хотя бы одна жёсткая ссылка на него? В прошлый раз мы создали жёсткую ссылку с помощью команды cp с ключом -l:

`cp -vl filea filealink`

Но обычно для создания жёстких ссылок используется команда link - ln. Синтаксис простой:

`ln -v filea samefile`

Но жёсткие ссылки работают только для файлов, а не директорий, так как жёсткая ссылка для директории может создать проблему в файловой системе. Жёсткие ссылки можно создавать и удалять, друг на друга они не влияют.

Жёсткие ссылки часто используются при бэкапе данных - когда у вас уже есть копия данных и изменились только некоторые файлы. Вместо того, чтобы копировать в новую директорию абсолютно всё, вы просто кладёте туда новые файлы и жёсткие ссылки на неизмененные файлы. Это позволяет здорово сэкономить место, так как жёсткие ссылки практически ничего не весят, в отличии от копии данных.

```
[user@centos8 test]$ ln -vs filea softlink
'softlink' -> 'filea'
[user@centos8 test]$ ln -vs dir_c dirlink
'dirlink' -> 'dir_c'
```

Кроме жёстких ссылок, существуют символические ссылки. На английском вместо symbolic link чаще говорят soft link, но на русском больше закрепилось выражение символические ссылки. Символические ссылки ссылаются не на иноду, а на жёсткую ссылку и создаются с помощью ln с ключом -s:

`ln -vs filea softlink`

Также символические ссылки работают с директориями:

`ln -vs dir_c dirlink`

Такой ссылке вообще плевать, есть файл или нет, удалили или не создали. Символическая ссылка может ссылаться на файл на других файловых системах.

```
[user@centos8 test]$ ls -l
total 0
lrwxrwxrwx. 1 user user 6 Nov 18 19:16 badlink -> nofile
drwxrwx---. 3 user user 47 Nov 17 18:42 dir_c
drwxrwx---. 2 user user 232 Nov 17 18:30 dir_d
drwxrwx---. 2 user user 141 Nov 17 18:38 dir_e
lrwxrwxrwx. 1 user user 5 Nov 18 19:10 dirlink -> dir_c
-rw-rw----. 3 user user 0 Nov 17 18:18 filea
-rw-rw----. 3 user user 0 Nov 17 18:18 filelink
-rw-rw----. 1 user user 0 Nov 17 18:33 newfile
-rw-rw----. 1 user user 0 Nov 18 18:50 newname
-rw-rw----. 3 user user 0 Nov 17 18:18 samefile
lrwxrwxrwx. 1 user user 5 Nov 18 19:10 softlink -> filea
```

Символические ссылки можно увидеть с помощью команды

```
ls -l
```

Эта команда даёт много информации, но сейчас нас интересует первый символ и последний столбик. Первый символ говорит о типе файла – если дефис, то это обычный файл, а правильнее – жёсткая ссылка, если d – это директория, если l – символьская ссылка. А в последнем столбике видны имена файлов и директорий. Рядом с символьскими ссылками стоит символ, показывающий, на какой файл ссылается символьская ссылка. Если этого файла нет:

```
ln -vs nofile badlink
```

допустим, удалили – то ссылка всё ещё остаётся и становится «битой ссылкой». Ссылки – это тоже файлы, а значит удаляются простой командой:

```
rm -v softlink
```

Многие новички путают символьские ссылки с ярлыками, но это разные вещи. Ярлык относится к графической оболочке, ему можно поставить описание, иконку, а также прописать какую-то команду запуска, чем активно пользуются всякие вирусы. Вы, возможно, сталкивались – вставили флешку в чужой компьютер, а там все содержимое превратилось в ярлыки. И люди ведутся на обман – ёлкают по ярлыкам. А в этих ярлыках, как правило, содержится команда для запуска какого-нибудь скрытого вируса, который прописывает себя в планировщик задач.

```
[user@centos8 test]$ ln -s /usr/share/applications .
[user@centos8 test]$ ls -l
total 0
lrwxrwxrwx. 1 user user 23 Nov 18 19:25 applications -> /usr/share/applications
```

Символические ссылки больше относятся к файловой системе. А путают их, потому что применение примерно одинаковое – допустим, вам нужен простой путь к какой-то директории или файлу, допустим, хотите видеть в домашней директории какую-нибудь другую директорию:

```
ln -s /usr/share/applications .
```

Или, допустим, какая-то программа всегда работает с определённой директорией, а вам это нужно поменять, например, выложить эту директорию на сервер. Это, конечно, зачастую можно сделать и в настройках программ, но ситуации бывают разные.

И так, мы рассмотрели, как создавать, копировать, перемещать, удалять файлы и директории, а также разобрали жёсткие и символические ссылки. Многое из этого, хотя и не всё, можно делать и через графический интерфейс, но когда речь идёт о серверах, ставить на них графический интерфейс не стоит – это пустая трата ресурсов и, теоретически, новые уязвимости. Эти команды довольно простые, если вы будете практиковаться, то многие эти операции вам проще будет сделать через командную строку даже при наличии графического интерфейса.

2.8.2 Практика

Вопросы

1. Что происходит с данными при перемещении в рамках одной файловой системы и разных файловых систем?
2. Что такое жёсткая ссылка?
3. Что такое символическая ссылка?
4. Удаляет ли файл команда rm?
5. Создайте жёсткую и символическую ссылки на файл. Удалите файл. Что произойдёт со ссылками и данными?
6. Как отличить в выводе „ls -l“ файл, директорию и символическую ссылку?
7. Как понять, сколько жёстких ссылок у файла?
8. Как понять, является ли два файла жёсткими ссылками на один файл?

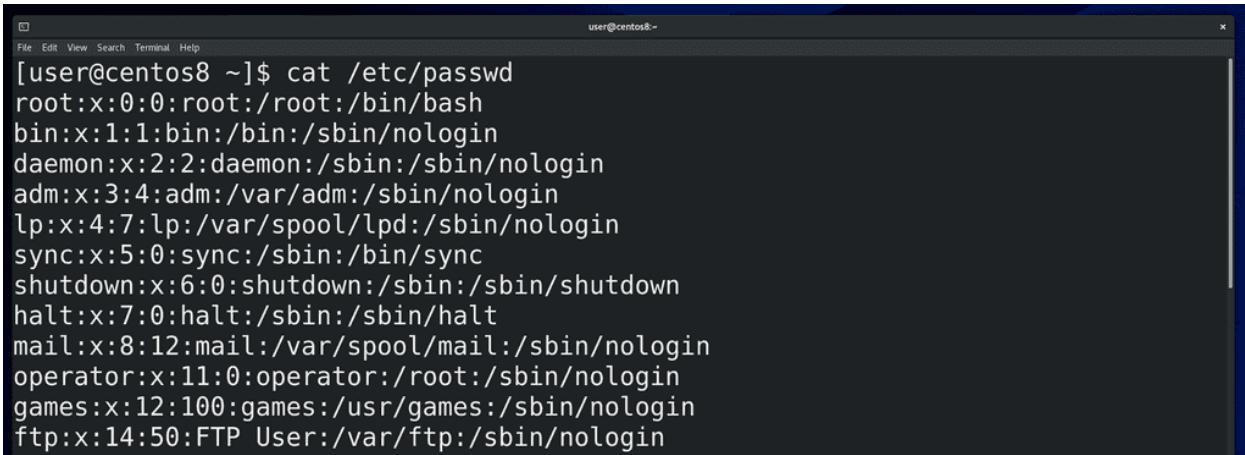
Задания

1. Создайте скрытую директорию и переместите файл в эту директорию.
2. Одной командой создайте файлы file20-29. Одной командой создайте жёсткие ссылки на эти файлы в скрытой директории. Одной командой удалите эти файлы в текущей директории.
3. Создайте символическую ссылку на существующие файл и директорию, а также на несуществующие файл и директорию.
4. Создайте файл file1 в директории dir1. Переименуйте файл в file2, не заходя в директорию с помощью cd.

2.9 09. Чтение текстовых файлов

2.9.1 09. Чтение текстовых файлов

Как вы, возможно, поняли, в текстовом интерфейсе всё есть текст – команды, файлы, устройства – всё что угодно. В том числе это касается настроек и логов большинства программ, которые хранятся в виде текстовых файлов. Поэтому инструментов для работы с текстом на UNIX-подобных системах просто дофига. Какие-то из них очень простые, а о каких-то пишут книги на сотни страниц. Ну и чтобы научиться работать с текстом, нам нужен какой-то подопытный файл – и для этого идеально подойдёт файл /etc/passwd – файл, в котором хранится информация о пользователях в системе.

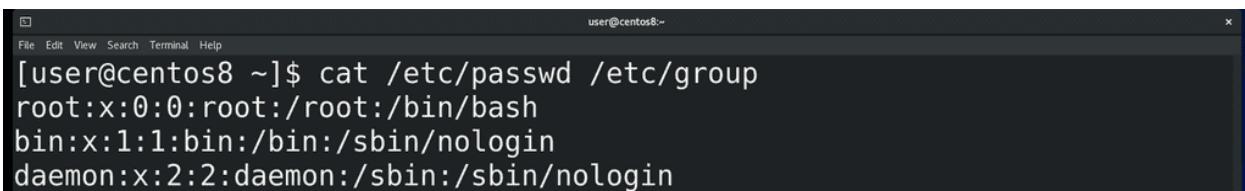


```
[user@centos8 ~]$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
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
```

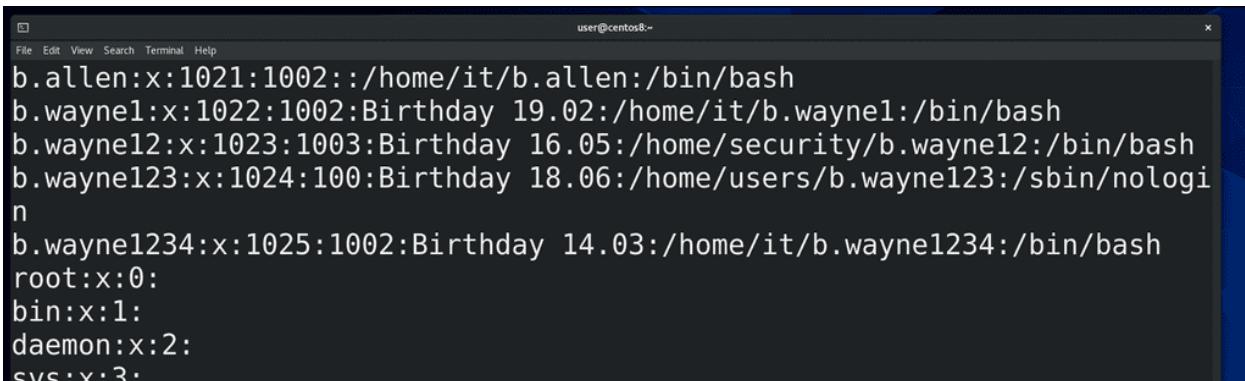
Для начала посмотрим содержимое этого файла. С помощью команды cat мы можем вывести содержимое этого файла в терминал:

```
cat /etc/passwd
```

Как вы видите, в терминале появилось много текста. Мы можем прокрутить его с помощью колёсика мыши, либо с помощью клавиш shift+PgUp, либо shift+PgDn. Запомните эти клавиши, потому что в виртуальном терминале мышка не работает.



```
[user@centos8 ~]$ cat /etc/passwd /etc/group
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
```



```
b.allen:x:1021:1002::/home/it/b.allen:/bin/bash
b.wayne1:x:1022:1002:Birthday 19.02:/home/it/b.wayne1:/bin/bash
b.wayne12:x:1023:1003:Birthday 16.05:/home/security/b.wayne12:/bin/bash
b.wayne123:x:1024:100:Birthday 18.06:/home/users/b.wayne123:/sbin/nologin
b.wayne1234:x:1025:1002:Birthday 14.03:/home/it/b.wayne1234:/bin/bash
root:x:0:
bin:x:1:
daemon:x:2:
sys:x:3:
```

Команда cat подойдёт, когда у вас есть относительно небольшой текстовой файл и вам просто нужно посмотреть его содержимое. Но в целом cat – от слова конкатенация – может объединять вывод содержимого нескольких файлов. Для примера возьмём один файл - /etc/group – и выведем оба файла разом:

```
cat /etc/passwd /etc/group
```

Если покрутим вверх, то увидим – как только закончился один файл, начинается другой.

```
[user@centos8 ~]$ cat -n /etc/passwd /etc/group
 1  root:x:0:0:root:/root:/bin/bash
 2  bin:x:1:1:bin:/bin:/sbin/nologin
 3  daemon:x:2:2:daemon:/sbin:/sbin/nologin
 4  adm:x:3:4:adm:/var/adm:/sbin/nologin
 5  lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
 6  sync:x:5:0:sync:/sbin:/bin/sync

[65] c.kent:x:1020:100::/home/users/c.kent:/sbin/nologin
[66] b.allen:x:1021:1002::/home/it/b.allen:/bin/bash
[67] b.wayne1:x:1022:1002:Birthday 19.02:/home/it/b.wayne1:/bin/bash
[68] b.wayne12:x:1023:1003:Birthday 16.05:/home/security/b.wayne12:/bin/bash
[69] b.wayne123:x:1024:100:Birthday 18.06:/home/users/b.wayne123:/sbin/nologin
[70] b.wayne1234:x:1025:1002:Birthday 14.03:/home/it/b.wayne1234:/bin/bash
 71  root:x:0:
 72  bin:x:1:
 73  daemon:x:2:
 74  sys:x:3:
 75  adm:x:4:
```

Ещё cat может пронумеровать строки с помощью ключа -n:

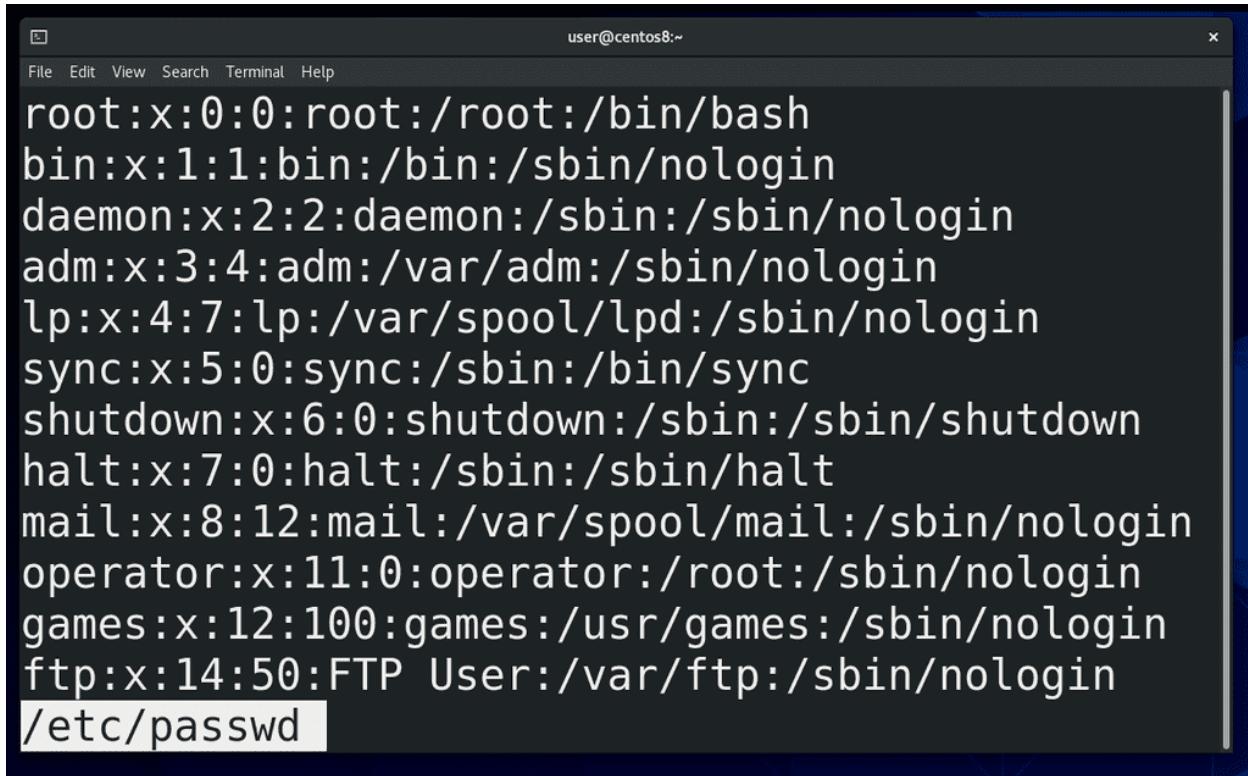
```
cat -n /etc/passwd
```

Ну и обратите внимание, как эта опция работает при выводе двух файлов.

```
[user@centos8 ~]$ tac /etc/passwd
b.wayne1234:x:1025:1002:Birthday 14.03:/home/it/b.wayne1234:/bin/bash
b.wayne123:x:1024:100:Birthday 18.06:/home/users/b.wayne123:/sbin/nologin
b.wayne12:x:1023:1003:Birthday 16.05:/home/security/b.wayne12:/bin/bash
b.wayne1:x:1022:1002:Birthday 19.02:/home/it/b.wayne1:/bin/bash
b.allen:x:1021:1002::/home/it/b.allen:/bin/bash
c.kent:x:1020:100::/home/users/c.kent:/sbin/nologin
o.queen:x:1019:1003::/home/security/o.queen:/bin/bash
b.murphy:x:1018:1002::/home/it/b.murphy:/bin/bash
```

У команды cat есть противоположная команда – tac, которая выводит текст реверсивно, то есть с конца:

```
tac /etc/passwd
```



The screenshot shows a terminal window with the command `cat /etc/passwd` running. The output lists various system users and their details:

```

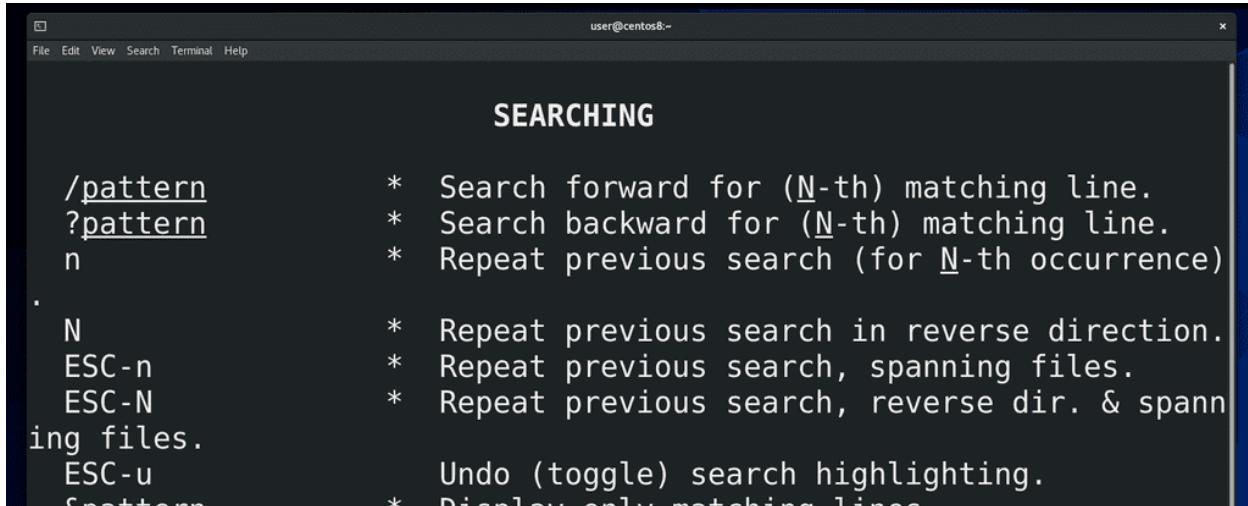
root:x:0:0:root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
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
/etc/passwd

```

Как вы заметили, команда `cat` просто вывела содержимое файла на экран и всё. Если там сотни строк – придётся крутить вверх, вниз. Если у вас задача прочесть какой-то большой файл, то вам больше подойдёт программа-читалка, например, `less`:

```
less /etc/passwd
```

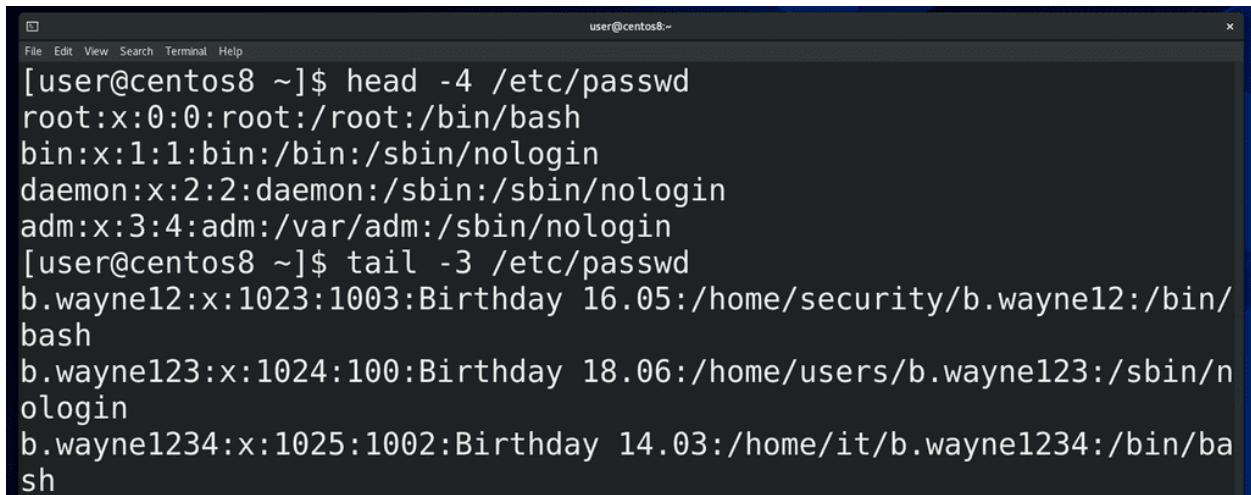
Такие программы часто называют пейджерами. Стрелки, Enter, pgup-pgdn, пробел – с помощью всего этого можно листать.



The screenshot shows the `less` program displaying a search menu. The title "SEARCHING" is at the top. Below it is a table of search commands and their descriptions:

<u>/pattern</u>	* Search forward for (<u>N</u> -th) matching line.
? <u>pattern</u>	* Search backward for (<u>N</u> -th) matching line.
n	* Repeat previous search (for <u>N</u> -th occurrence)
.	
N	* Repeat previous search in reverse direction.
ESC-n	* Repeat previous search, spanning files.
ESC-N	* Repeat previous search, reverse dir. & spann ing files.
ESC-u	Undo (toggle) search highlighting.
<u>s</u> pattern	* Display only matching lines

Если написать слэш (/) и текст, то `less` поищет этот текст в файле, а с помощью `n` или `N` можно перейти на следующее или предыдущее совпадение соответственно. Ну и `q` чтобы выйти. Похоже на `man?` А это потому что `man` использует `less` в качестве читалки. Если в `man` или в `less` нажать `h`, то откроется небольшой гайд по командам управления читалкой.



```
[user@centos8 ~]$ head -4 /etc/passwd
root:x:0:0: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
[user@centos8 ~]$ tail -3 /etc/passwd
b.wayne12:x:1023:1003:Birthday 16.05:/home/security/b.wayne12:/bin/
bash
b.wayne123:x:1024:100:Birthday 18.06:/home/users/b.wayne123:/sbin/n
ologin
b.wayne1234:x:1025:1002:Birthday 14.03:/home/it/b.wayne1234:/bin/ba
sh
```

Но жизнь слишком коротка, чтобы читать большие файлы, поэтому часто пользуются двумя командами – `head` и `tail` – они показывают определённое количество строк с начала и с конца файла соответственно:

```
head /etc/passwd
tail /etc/passwd
```

По умолчанию они показывают 10 строк, но можно указать что-то своё:

```
head -4 /etc/passwd
tail -3 /etc/passwd
```

Ещё `tail` может читать с определённой строки, допустим всё что ниже 35 строки:

```
tail -n +35 /etc/passwd
```

```
[user@centos8 ~]$ tail -f ~/.bash_history
ls -s /usr/share/applications .
ln -s /usr/share/applications .
ls -l
rm applications
ln -s /usr/share/applications .
ls -l
ls
touch
command1
command2

command3
```

Одна из самых используемых опций tail – ключ f – может показывать, что добавляется в файл в реальном времени:

```
tail -f logfile
```

Это часто применяется при решении проблем, когда вы видите кучу логов и вам нужно понять, что именно происходит при выполнении каких-то действий. Для примера посмотрим файл:

```
tail -f ~/.bash_history
```

Сюда записываются выполненные команды. Так вот, я открываю ещё один эмулятор терминала и ввожу какие-то команды. Потом закрываю новое окно и вижу, что у меня в этот файл добавились строчки. Когда текста много, не всегда понятно, что где куда добавилось, поэтому я могу зажать Enter и у меня появляется пустое пространство. Не беспокойтесь, это никак не влияет на сам файл, это просто для удобства чтения. Чтобы выйти, нажмите Ctrl+c. Возможно вам пока это не понятно, но просто запомните – если вам нужно открыть конец файла и ждать появления новых строчек, например в случае чтения логов, то используется tail с ключом f.

Вообще, одна из лучших фишек команд less, head и tail – то что они не пытаются прочесть весь файл целиком. Вы когда-нибудь пытались открыть текстовой файл на 100 мегабайт? А админы иногда сталкиваются с файлами с размером в пару гигабайт. Бывает – утром пришел на работу, а там сервер не работает. Смотришь – нет места на диске. Почему? А там лог файл на десяток или сотню гигабайт забил весь диск за одну ночь. Да, конечно, по хорошему такие ситуации легко предотвратить, но всё же речь о другом. И вот тебе нужно понять, что это там в логах такого на пару гигов. Сервера просто зависают при попытке открыть такие файлы. Но на линуксах есть эти утилиты, с помощью которых можно запросто прочитать файл любого размера. И так как, обычно, такие большие лог файлы забиваются одними и теми же строчками, то достаточно вывести, допустим, последние строки 50 файла, чтобы понять, что там произошло.

```
[user@centos8 ~]$ grep bash /etc/passwd
root:x:0:0:root:/root:/bin/bash
user:x:1000:1000:User:/home/user:/bin/bash
user5:x:1001:1002:/home/it/user5:/bin/bash
user7:x:1003:1002:/home/it/user7:/bin/bash
user8:x:1004:1002:/home/it/user8:/bin/bash
user13:x:1005:1003:/home/security/user13:/bin/bash
```

Ну и давайте напоследок затронем ещё одну команду – grep, хотя о ней мы ещё поговорим более подробно в другой раз. grep позволяет нам искать строки текста по шаблону. Допустим, в нашем файле /etc/passwd есть пользователи, которые пользуются интерпретатором bash. Я могу написать:

```
grep bash /etc/passwd
```

и команда выдаст мне только строки, в которых есть слово bash.

```
[user@centos8 ~]$ grep -n bash /etc/passwd
1:root:x:0:0:root:/root:/bin/bash
44:user:x:1000:1000:User:/home/user:/bin/bash
46:user5:x:1001:1002:/home/it/user5:/bin/bash
48:user7:x:1003:1002:/home/it/user7:/bin/bash
49:user8:x:1004:1002:/home/it/user8:/bin/bash
50:user13:x:1005:1003:/home/security/user13:/bin/bash
```

Я могу добавить опцию n:

```
grep -n bash /etc/passwd
```

тогда я ещё увижу номера строк.

```
[user@centos8 ~]$ grep -rn bash /etc/
grep: /etc/crypttab: Permission denied
/etc/X11/xinit/xinitrc.d/10-qt5-check-opengl2.sh:1:#!/bin/b
ash
/etc/X11/xinit/Xclients:1:#!/bin/bash
/etc/X11/xinit/Xsession:1:#!/bin/bash
grep: /etc/libreport/cert-api.access.redhat.com.pem: Permis
sion denied
/etc/skel/.bash_logout:1:# ~/.bash_logout
/etc/skel/.bash_profile:1:# .bash_profile
```

grep может искать рекурсивно. То есть, я могу указать grep-у, чтобы он нашёл мне все упоминания bash в директории /etc:

```
grep -rn bash /etc/
```

Как вы видите, вывода много, много где ошибки, потому что не хватает прав.

```
[user@centos8 ~]$ grep -rl bash /etc/
grep: /etc/crypttab: Permission denied
/etc/X11/xinit/xinitrc.d/10-qt5-check-opengl2.sh
/etc/X11/xinit/Xclients
/etc/X11/xinit/Xsession
grep: /etc/libreport/cert-api.access.redhat.com.pem: Permission denied
/etc/skel/.bash_logout
/etc/skel/.bash_profile
/etc/skel/.bashrc
```

Я могу использовать ключ l – чтобы просто выводить имена файлов, в которых встречается слово bash:

```
grep -rl bash /etc/
```

```
[user@centos8 ~]$ grep -v bash /etc/passwd
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
```

Ну и я могу найти все строки, в которых не содержится это слово, то есть реверсировать поиск с помощью ключа v:

```
grep -v bash /etc/passwd
```

И вот мы затронули 4 команды для чтения текста – cat, less, head и tail, и даже научились искать текст с помощью команды grep. Осталось научиться писать.

2.9.2 Практика

Вопросы

- С помощью каких команд можно посмотреть содержимое текстового файла?
- Какими способами можно посмотреть 30 строчку файла /etc/passwd?
- Какое преимущество дают head и tail при чтении больших файлов?
- Как найти файл, в котором есть слово «hello»?
- Каким удобным способом можно смотреть изменяющийся файл?

Задания

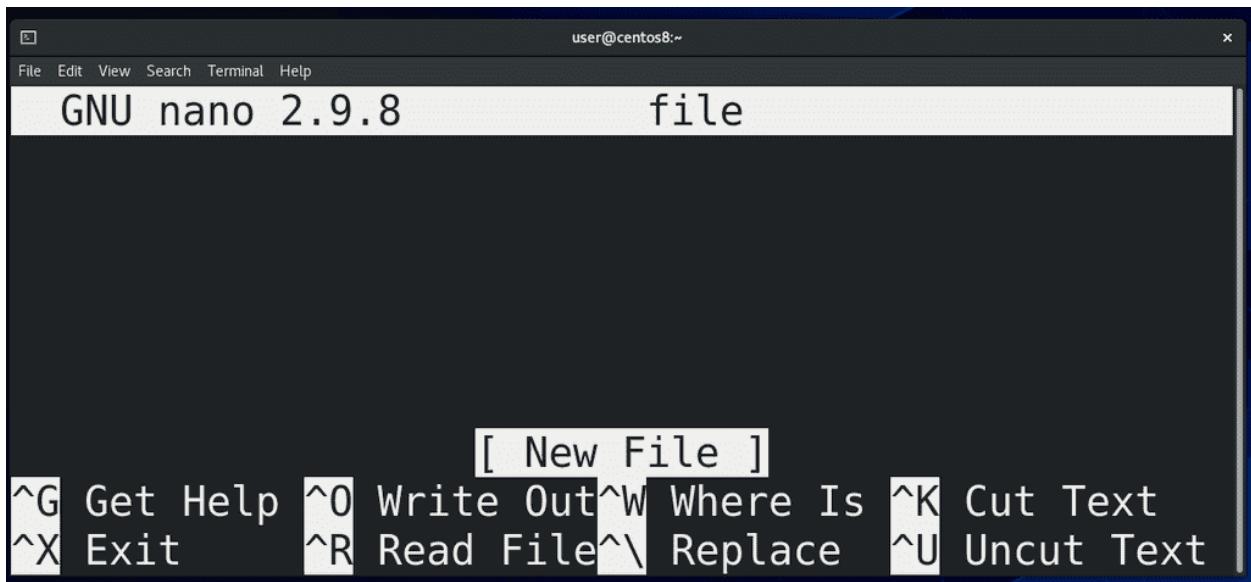
- Выведите первые и последние 7 строк из файла /etc/group.
- Выведите всё строки ниже 14 в файле /etc/group.
- Используя поиск в less, найдите слово games в файле /etc/passwd.

4. Найдите слово games и номер соответствующей строки в файле /etc/passwd с помощью команды grep.
5. Найдите все файлы, в которых встречается текст sync в директории /etc.
6. Найдите все файлы, в которых встречается текст «sync» независимо от регистра букв. *

2.10 10. Текстовые редакторы nano и vi

2.10.1 10. Текстовые редакторы nano и vi

Для редактирования текстовых файлов вам нужен текстовой редактор. Самые известные – vi (и его современная реализация vim), emacs и nano. Объяснением одного nano не обойтись - хоть он и попроще, но очень важно уметь работать с vi и от этого никуда не деться. nano не всегда предустановлен, а без умения работы с vi может сложиться ситуация, что вы и nano не сможете установить. Нет, теоретически, можно и без vi обойтись, но это потребует у вас много времени - легче научиться хотя бы базово работать с этой программой. И не то чтобы vi хуже или сильно сложнее – большинство админов и программистов как раз таки предпочитают vim. Я рассмотрю обе программы, чем пользоваться – решайте сами. Но уметь работать с vi нужно в любом случае.



Начнём с nano. Программа простая – вы пишете nano и имя файла, который вы хотите создать или изменить:

```
nano file
```

Открывается программка и здесь вы можете вводить текст, изменять его. Для перемещения используются стрелки. Чтобы быстро перемещаться между словами можно зажать Ctrl и использовать стрелки, чтобы перейти в начало строки – Ctrl+A, чтобы в конец – Ctrl+E. Внизу есть подсказки по горячим клавишам. Значок рядом с буквами (^ - карет) обозначает Ctrl, например, Ctrl+G для небольшого гайда.

The screenshot shows a terminal window titled "user@centos8:~". The title bar has menu items: File, Edit, View, Search, Terminal, Help. The main area displays the "Main nano help text" in white on a black background. It includes key bindings for Undo (M-U), Redo (M-E), Mark (M-A), Copy (M-6), Refresh (^L), Close (^X), and various search and navigation commands like Where Is (^W), Prev Line (^P), Next Line (^N), Prev Page (^Y), and Next Page (^V). The bottom status bar shows the file name "file" and the status "Modified".

```

Main nano help text

M-U Undo the last operation
M-E Redo the last undone operation

M-A (^6) Mark text starting from the
cursor position
M-6 (^M-^) Copy current line (or marked
region) and store it in cutbuffer

^L Refresh ^W Where Is ^P Prev Line ^Y Prev Page
^X Close M-W WhereIs ^N Next Line ^V Next Page

```

В гайде кроме знака Ctrl также встречается M – это не буква М, а клавиша мета. Скорее всего, у вас на клавиатуре её нет, поэтому её заменяет либо Alt, либо клавиша Win. Допустим, комбинация Alt-U. Как видите, внизу есть подсказка – Ctrl+X - закрыть это окно.

The screenshot shows a terminal window titled "user@centos8:~". The title bar has menu items: File, Edit, View, Search, Terminal, Help. The main area displays the "GNU nano 2.9.8" title and the file name "file" with the status "Modified". Below the file name, there is a list of lowercase letters: a, b, d, c, e. At the bottom, a message "File Name to Write: file" is followed by a series of hotkey descriptions: Get Help (^G), DOS Form (^M), Append (^M-A), Backup File (^M-B), Cancel (^C), Mac Form (^M-M), Prepend (^M-P), and To Files (^T).

```

File Name to Write: file
^G Get Help ^M-D DOS Form ^M-A Append ^M-B Backup File
^C Cancel ^M-M Mac Form ^M-P Prepend ^T To Files

```

Разбирать все горячие клавиши я не буду, но давайте пройдёмся по основным. Начнём с сохранения – написали какой-то текст, хотим сохранить. Нажимаем Ctrl+O – внизу появляется поле, где можно указать новое имя для файла, либо оставить то имя, которое мы указывали, когда запускали nano. Нас это имя устраивает, поэтому нажимаем Enter и видим, что внизу появилась надпись, где сказано, сколько линий у нас в файле. Чтобы выйти – Ctrl+X. Если перед закрытием мы сделали какие-то изменения, то у нас появится вопрос – сохранить изменения или нет – тут пишем Y или N, или Ctrl+C, всё как указано в подсказке.

```

GNU nano 2.9.8      /etc/passwd

root:x:0:0: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:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
[ File '/etc/passwd' is unwritable ]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text
^X Exit      ^R Read File ^\ Replace ^U Uncut Text

```

Давайте обратимся к файлу /etc/passwd:

```
nano /etc/passwd
```

Как видите, внизу надпись нас предупреждает, что этот файл невозможно редактировать – потому что у нас нет на это прав. Поэтому просто скопируем этот файл к себе в домашнюю директорию:

```
cp /etc/passwd ~
```

С копией файла у нас не будет никаких проблем, так как эта копия принадлежит нам. Откроем копию файла:

```
nano ~/passwd
```

и убедимся, что теперь этой ошибки нет.

```

GNU nano 2.9.8      /home/user/passwd      Modified

 9 mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
10 operator:x:11:0:operator:/root:/sbin/nologin
11 games:x:12:100:games:/usr/games:/sbin/nologin
12 ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
13 nobody:x:65534:65534:Kernel Overflow User:/:/s$
14 dbus:x:81:81:System message bus:/:/sbin/nologin
15 systemd-coredump:x:999:997:systemd Core Dumper$
```

line 12/71 (16%), col 1/44 (2%), char 431/3776 (11)

^G Get Help ^O Write Out ^W Where Is ^K Cut Text
^X Exit ^R Read File ^\ Replace ^U Uncut Text

Нередко бывает нужно сориентироваться, в какой строчке мы сейчас находимся. Для этого нужно нажать **Ctrl+C** – появится подсказка – на какой линии вы находитесь, на каком символе этой линии и на каком символе файла в целом. Можно ещё нажать **Alt+#+** (может различаться от дистрибутива, нужно смотреть в подсказке **Ctrl+G**) и слева появится нумерация строк. Но если мы закроем nano и заново откроем, то придётся заново включать нумерацию.

```

set linenumbers
```

^G Get Help ^O Write Out ^W Where Is ^K Cut Text
^X Exit ^R Read File ^\ Replace ^U Uncut Text

А давайте сделаем так, чтобы нумерация была всегда видна – для этого нужно подредактировать файл конфигурации nano – **.nanorc**, который должен находиться в домашней директории пользователя и должен быть скрытым, то есть имя должно начинаться с точки. Пишем:

```
nano ~/.nanorc
```

Я использую тильду слеш (**~/**), потому что не важно, где я нахожусь, тильда слэш всегда ведёт в домашнюю директорию, ну и название файла - **.nanorc**. Файл новый, потому что до этого мы никаких настроек nano не сохраняли. Пишем:

```
set linenumbers
```

сохраняем и выходим. Открываем файл:

```
nano passwd
```

и вот теперь по умолчанию нумерация включена.

The screenshot shows the nano 2.9.8 editor window. The title bar says "GNU nano 2.9.8" and "passwd". The main area displays the contents of the /etc/passwd file:

```
1 root:x:0:0:root:/root:/bin/bash
2 bin:x:1:1:bin:/bin:/sbin/nologin
3 daemon:x:2:2:daemon:/sbin:/sbin/nologin
4 adm:x:3:4:adm:/var/adm:/sbin/nologin
5 lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
```

The word "user" is typed into the search bar at the bottom. The status bar at the bottom shows various keyboard shortcuts for navigation and search.

Теперь попытаемся найти строчку нашего пользователя user – нажимаем Ctrl+W и появляется надпись Search: – пишем user. Вариантов может быть несколько. Чтобы перемещаться между вариантами, нажимаем Alt+W.

The screenshot shows the nano 2.9.8 editor window. The title bar says "GNU nano 2.9.8" and "passwd". The main area displays the contents of the /etc/passwd file:

```
1 root:x:0:0:root:/root:/bin/bash
2 bin:x:1:1:bin:/bin:/sbin/nologin
3 daemon:x:2:2:daemon:/sbin:/sbin/nologin
4 adm:x:3:4:adm:/var/adm:/sbin/nologin
5 lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
```

The word "user" is selected in the search results list. The status bar at the bottom shows various keyboard shortcuts for navigation and search.

Теперь попытаемся перейти сразу на какую-то строчку – пишем Ctrl+W, а потом Ctrl+T – и появляется строка с предложением ввести номер строки и символа. Можно просто написать номер строки и Enter, либо номер строки и символа через пробел – 13 6.

The screenshot shows a terminal window titled "user@centos8:~". The title bar includes the application name "GNU nano 2.9.8" and the file status "passwd Modified". The main area displays several lines of text starting with "74 nobody:x:65534:65534:Kernel Overflow User:/:/s\$". At the bottom, there is a menu bar with options like File, Edit, View, Search, Terminal, Help, and a set of keyboard shortcuts for Get Help (^G), Write Out (^O), Where Is (^W), Cut Text (^K), Exit (^X), Read File (^R), Replace (^V), and Uncut Text (^U).

```

74 nobody:x:65534:65534:Kernel Overflow User:/:/s$
75 dbus:x:81:81:System message bus:/sbin/nologin
76 systemd-coredump:x:999:997:systemd Core Dumper$
77 systemd-coredump:x:999:997:systemd Core Dumper$
78 systemd-coredump:x:999:997:systemd Core Dumper$
```

^G Get Help **^O** Write Out **^W** Where Is **^K** Cut Text
^X Exit **^R** Read File **^V** Replace **^U** Uncut Text

Часто бывает нужно вырезать или скопировать целую строку – нажимаем Ctrl+K, чтобы вырезать и Ctrl+U чтобы вставить. Можно переместиться в другое место и снова нажать Ctrl+U, чтобы вставить. Очень удобно, когда нужно несколько похожих строк. Можно разом вырезать несколько строк – нажимаем Ctrl+K несколько раз, а потом при Ctrl+U вставляются все вырезанные строки. Чтобы отменить последние изменения, нажимаем Alt+U.

Ладно, не будем пере усложнять. Для начала выше сказанных горячих клавиш для работы с nano вполне хватит. Некоторые другие клавиши мы рассмотрим, когда непосредственно начнём работать с файлами. Теперь перейдём к vi. Я не буду также детально рассматривать vi, только пройдусь по самому необходимому. Сам я vi не пользуюсь без необходимости, но для желающих в интернете миллион гайдов и даже [игра](#), объясняющая, как работать с vi.

The screenshot shows a terminal window titled "user@centos8:~". The title bar includes the application name "File Edit View Search Terminal Help". The main area displays a list of system users from the /etc/passwd file. The status bar at the bottom right shows "1,1 Top".

```

n: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
```

1,1 Top

Точно как и с nano, вы можете написать vi file, чтобы создать или изменить файл:

```
vi passwd
```

В vi есть несколько режимов – командный режим, режим ввода и режим последней строки. Когда вы открываете vi, вы оказываетесь в командном режиме – в этом режиме вы не можете писать текст, но

можете выполнять команды, например - x – для удаления символа или два раза d для удаления строки. Backspace в vi не работает. Но стоит учесть, что Centos вместо vi открывает vim, а в нём backspace работает (в режиме ввода).

```
n:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
some text here
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
-- INSERT --           3,15          Top
```

Чтобы перейти в режим ввода, можно использовать несколько клавиш – нажимаем і маленькое – внизу появляется надпись INSERT – и начинаем писать ровно там, где был курсор. Чтобы выйти из режима ввода, нажимаем Esc. I большое, то есть Shift+i – начинаем писать с начала строки. а маленькое – начинаем писать после курсора, A большое – в конце строки. o – добавляем строку ниже и начинаем в ней писать, O – строка сверху. Напомню, чтобы удалить текст, используем backspace, либо переходим в режим команд, то есть нажимаем Esc, наводим курсор куда нужно и нажимаем x. Если хотим сохранить это безобразие, в командном режиме нажимаем Shift+Z+Z.

```
some text herdas
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
E37: No write since last change (add ! to override)
)
:q!
```

Ну и третий режим – режим последней строки. Чтобы перейти в него, нужно в командном режиме написать двоеточие (:) – внизу появится двоеточие. Тут вообще много всяких команд можно ввести, но нас интересуют основные – как сохранить и как выйти. И так, если никаких изменений нет – то пишем q, то есть чтобы стало :q и нажимаем Enter. Если вы сделали какие-то изменения, то есть, нажали i, ввёли какой-то текст, потом нажали Esc и двоеточие, то при попытке написать q и Enter внизу появится ошибка с подсказкой, что нужно добавить восклицательный знак. Пишем :q! и Enter – никакие изменения не сохраняются. Если хотим сохранить изменения – пишем w и enter и изменения сохраняются. Можем разом сохранить изменения и выйти, как при использовании Shift+Z+Z – для этого пишем :wq.

Несмотря на все эти махинации, vi, а точнее vim, очень любят в народе. Для задач администрирования функционала nano вполне хватает, но каждый сам решает, с каким редактором ему работать. И тем не менее, обязательно научитесь работать с vi, хотя бы базово – открыть файл, изменить какой-то текст и сохранить, либо не сохранять.

2.10.2 Практика

Задания

- Создайте директорию myfiles в домашней директории вашего пользователя. Скопируйте файл /etc/passwd в эту директорию.
- Откройте файл с помощью nano, найдите и удалите 16 строчку.
- Найдите слово false и замените на true.
- Сделайте так, чтобы нумерация строк оставалась даже после перезапуска nano.
- С помощью vi добавьте в этом файле новую строчку между 5 и 6. Пропишите там Hello и сохраните файл.
- В конце файла добавьте новую строчку и пропишите World. Выберите не сохраняя файл.
- Проверьте содержимое с помощью cat. Найдите номер строчки с Hello с помощью grep.

2.11 11. Стандартные потоки

2.11.1 11. Стандартные потоки

Если открыть в википедии страничку с названием «Философия Unix», можно заметить 3 пункта:

[Дуг Макилрой](#), изобретатель каналов Unix и один из основателей традиции Unix, обобщил философию следующим образом:

«Философия Unix гласит:

Пишите [программы](#), которые делают что-то одно и делают это хорошо.

Пишите программы, которые бы работали вместе.

Пишите программы, которые бы поддерживали [текстовые потоки](#), поскольку это универсальный интерфейс».

Второй и третий пункты можно связать – если программы будут иметь возможность обмениваться текстом, то, теоретически, возможно сделать так, чтобы они работали вместе.

$$\begin{array}{l} A + B = C \\ C - D = E \end{array}$$

Представьте себе 2 программы – одна умеет только складывать два числа, а другая только отнимать. Если вам нужно решить уравнение, в котором сначала сложить два числа, а потом из суммы отнять третье число, то вам нужно было бы выполнить первую программу, посмотреть сумму, а потом выполнить вторую программу с полученным числом. Но было бы здорово, если первая программа смогла передать свой результат, то есть сумму, второй программе, без вашего участия. Для того, чтобы команды могли работать между собой, в UNIX-подобных системах есть так называемые стандартные потоки – стандартный ввод, стандартный вывод и стандартный вывод ошибок. Вообще, эти потоки существуют независимо от того, объединяете вы команды или нет, и мы с ними уже сталкивались.

Первый поток – это стандартный ввод – называется stdin, то есть то, откуда вводятся данные, например, с клавиатуры. Из того что мы делали – мы вводили какие-то данные с клавиатуры в эмулятор терминала.

```
[user@centos8 ~]$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos
[user@centos8 ~]$ cat
some text
some text
another text
another text
```

Второй поток – стандартный вывод – называется `stdout`, то есть то, что выводит команда. Например, если мы пишем `ls` – мы видим на экране какую-то информацию – вот то что появляется на экране – это и есть стандартный вывод. `man` по `cat` говорит, что если запустить команду `cat`, не указав файл, то по умолчанию он будет копировать стандартный ввод в стандартный вывод. То есть, вы ввели какой-то текст – это `stdin`, а `cat` вывела этот текст – `stdout`.

```
[user@centos8 ~]$ mkdir dir1 dir3
[user@centos8 ~]$ ls dir1 dir2 dir3
ls: cannot access 'dir2': No such file or directory
dir1:

dir3:
[user@centos8 ~]$
```

Ну и третий поток – это стандартный вывод ошибки – `stderr`. Многие программы на Linux заточены под то, чтобы особо реагировать на ошибки, отделять их от стандартного вывода. Я создаю 2 директории – `dir1` и `dir3`, и запускаю `ls` на 3 директории:

```
ls dir1 dir2 dir3
```

Как видите, команда `ls` для второй директории вывела сообщение об ошибке – что такой директории или файла нет, и вывела информацию о существующих директориях. Мы, как читатели, можем не обратить внимания на ошибку, но другие программы, например, тот же `bash`, видят `stdout` и `stderr` по разному.

```
(stdin) A + B = C (stdout)
(stdin) C - D = E (stdout)
```

Вернёмся к нашему уравнению. Теперь мы понимаем, что для отправки результата первой программы, то есть суммы (стандартного вывода), во вторую программу в качестве стандартного ввода, нам нужно скопировать `stdout` первой команды в `stdin` второй команды. А чтобы направлять потоки нам нужны каналы, ну или трубопровод. Возвращаясь на страницу философии Unix, мы увидим – человек, который говорил нам про 3 пункта философии – Макилрой – является изобретателем каналов Unix. Эти каналы, часто называемые конвейером, позволяют управлять стандартными потоками. Мы можем делать это с помощью специальных символов.

Эти символы как направления. Символ стандартного ввода (`stdin`) – знак меньше (`<`). Обычно под

stdin мы подразумеваем то, что вводим с клавиатуры. Но вместо клавиатуры может быть, допустим, содержимое текстового файла. С пройденными командами получится не совсем наглядно. Поэтому я покажу на примере команды mail, которая отправляет письма. Я могу написать:

```
mail user < file
```

Если бы я не использовал перенаправление stdin (< file), после нажатия Enter я написал бы письмо вручную. А так - направил команде mail содержимое файла. Но направление стандартного ввода используется не так часто.

The screenshot shows a terminal window with a dark background and light-colored text. At the top, there's a menu bar with options: File, Edit, View, Search, Terminal, Help. Below the menu, the terminal prompt is [user@centos8 ~]\$. The user runs two commands: 'ls > filelist' and 'cat filelist'. The 'ls' command lists various directory entries like Desktop, dir1, dir3, Documents, Downloads, file, filelist, Music, Pictures, Public, Templates, and Videos. The 'cat' command then displays the same list of entries from the 'filelist' file.

```
[user@centos8 ~]$ ls > filelist
[user@centos8 ~]$ cat filelist
Desktop
dir1
dir3
Documents
Downloads
file
filelist
Music
Pictures
Public
Templates
Videos
[user@centos8 ~]$
```

Для направления стандартного вывода используется знак больше (>). Мы можем направлять вывод команд не на экран, а в какой-нибудь файл. Допустим, давайте отправим вывод команды ls в файл:

```
ls > filelist
cat filelist
```

Как вы заметили, на экране теперь ls ничего не показал, но то что он должен был показать, сохранилось в файле.

```

user@centos8 ~]$ grep user /etc/passwd >> filelist
user@centos8 ~]$ cat filelist
Desktop
dir1
dir3
Documents
Downloads
file
filelist
Music
Pictures
Public
Templates
Videos
tss:x:59:59:Account used by the trousers package to sandbox the tcscd daemon:/dev/null:/sbin/nologin
qemu:x:107:107:qemu user:::/sbin/nologin
usbmuxd:x:113:113:usbmuxd user:::/sbin/nologin
saslauthd:x:990:76:Saslauthd user:/run/saslauthd:/sbin/nologin
radvd:x:75:75:radvd user:::/sbin/nologin
clevis:x:983:982:Clevis Decryption Framework unprivileged user:/var/cache/clevis:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
user:x:1000:1000:user:/home/user:/bin/bash

```

Когда мы используем знак больше:

```
ls > filelist
```

содержимое файла перезаписывается. Но я могу не перезаписывать файл, а добавлять в него текст. Для этого нужно два знака больше (`>>`). Для примера, добавим в существующий файл также содержимое команды:

```
grep user /etc/passwd >> filelist
cat filelist
```

Как видите, файл не перезаписался, просто внизу появился вывод команды grep.

```
grep bash /etc/passwd > filelist
```

– файл перезаписался,

```
grep user /etc/passwd >> filelist
```

– вывод grep добавился в тот же файл.

```
[user@centos8 ~]$ grep -rl bash /etc/ 2> errors
/etc/skel/.bash_logout
/etc/skel/.bash_profile
/etc/skel/.bashrc
/etc/selinux/targeted/contexts/files/file_contexts
/etc/selinux/targeted/contexts/files/file_contexts.bin
/etc/kernel/postinst.d/51-dracut-rescue-postinst.sh
/etc/bashrc
/etc/passwd-
/etc/passwd
/etc/profile
```

Для направления стандартного вывода ошибок используется знак `- 2>`. Помните, я показывал рекурсивный поиск с помощью grep? Тогда у нас была куча ошибок с доступом к файлам. Теперь мы можем сделать так:

```
grep -rl bash /etc 2> errors
cat errors
```

Все ошибки доступа направились в файл errors, а на экране только список нужных файлов, то есть stdout. Если сообщения об ошибках нас не интересуют, мы можем направить их не в файл, а в эдакую чёрную дыру `- /dev/null`. Мы о ней ещё поговорим. А пока пример с тем же grep:

```
grep -rl bash /etc 2> /dev/null
ls dir1 dir2 dir3 2> /dev/null
```

Также мы можем разные потоки направлять в разные места, допустим:

```
grep -rl bash /etc/ > filelist 2> error
```

```
[user@centos8 ~]$ grep -rl bash /etc/ &> output
[user@centos8 ~]$ cat output
grep: /etc/crypttab: Permission denied
grep: /etc/libreport/cert-api.access.redhat.com.pem: Permission denied
/etc/skel/.bash_logout
/etc/skel/.bash_profile
/etc/skel/.bashrc
grep: /etc/pki/rpm-gpg: Permission denied
/etc/selinux/targeted/contexts/files/file_contexts
/etc/selinux/targeted/contexts/files/file_contexts.bin
```

Как вы, возможно, заметили, если направить в файл stdout:

```
ls dir1 dir2 dir3 > file
```

то в командной строке будет stderr. Если направить stderr:

```
ls dir1 dir2 dir3 2> file
```

то мы увидим stdout. Если ничего не направлять, на экране мы увидим оба потока. А что делать, если мы хотим сохранить в файле и stdout, и stderr? Для этого есть символ амперсанд (&):

```
grep -rl bash /etc &> output
```

Если посмотреть, то всё будет ровно так, как было бы в командной строке - cat output.

```
(stdin) A + B = C (stdout)
(stdin) C - D = E (stdout)
```

Возвращаясь к нашему уравнению, всё что мы делали выше – направляли потоки в файлы. А в уравнении нам нужно направить stdout в stdin. Для этого используется другой символ – прямая вертикальная линия - |. Её часто называют пайп, то есть труба.

```
[user@centos8 ~]$ grep user /etc/passwd | tail -2
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
user:x:1000:1000:user:/home/user:/bin/bash
[user@centos8 ~]$
```

Для примера возьмём команду:

```
grep user /etc/passwd
```

которая покажет пару строк, а затем, поставив вертикальную линию и написав tail -2:

```
grep user /etc/passwd | tail -2
```

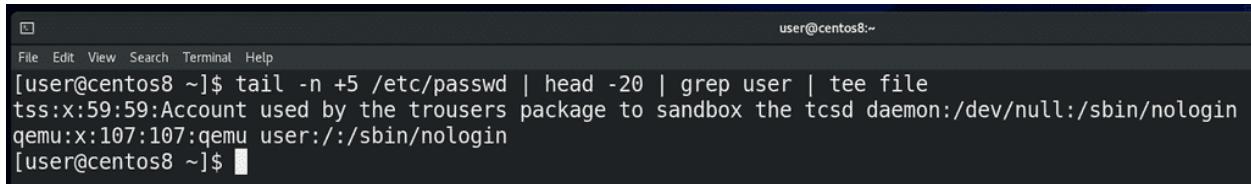
мы получим только последние две строчки из того, что показал бы grep. То есть, сначала мы получили стандартный вывод от команды grep и превратили его в стандартный ввод для команды tail.

```
[user@centos8 ~]$ head -9 /etc/passwd | tail -1
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
[user@centos8 ~]$
```

Или, допустим, я хочу увидеть 9 строчку файла /etc/passwd – я вывожу первые 9 строчек с помощью head -9, а потом, с помощью tail -1:

```
head -9 /etc/passwd | tail -1
```

показываю последнюю строчку от вывода предыдущей команды, то есть 9-ую.



```
[user@centos8 ~]$ tail -n +5 /etc/passwd | head -20 | grep user | tee file
tss:x:59:59:Account used by the trousers package to sandbox the tcscd daemon:/dev/null:/sbin/nologin
qemu:x:107:107:qemu user::/sbin/nologin
[user@centos8 ~]$
```

Ещё один пример – я хочу вывести строчки с 5-ой до 25 и среди них найти слово user и сохранить это в файле – я пишу:

```
tail -n +5 /etc/passwd
```

то есть вывожу всё что ниже 5 строки. Дальше из полученного результата вырезаю первые 20 строк (head -20), а потом в полученном результате ищу слово user (grep user) и направляю в файл:

```
tail -n +5 /etc/passwd | head -20 | grep user > file
```

Если вы хотите, чтобы результат одновременно выводился в командную строку и записывался в файл, то можете использовать команду tee:

```
tail -n +5 /etc/passwd | head -20 | grep user | tee file
```

Как вы заметили, вывод можно направлять от одной команды к другой. И, ставя много таких пайпов, мы можем с помощью простых команд решать сложные задачи. Конвейер вместе с функционалом интерпретатора даёт возможность нам решать такие задачи, для которых нет готовых программ.

2.11.2 Практика

Вопросы

1. Перечислите стандартные потоки.
2. Как и куда можно направлять стандартные потоки?
3. Как направить стандартный вывод в терминал и в файл одновременно?
4. Как направить stdout и stderr в один файл и в разные файлы?

Задания

1. Запишите в файл mykernel последние 10 строк файла /var/log/kdump.log
2. Добавьте (не перезаписывайте) в этот файл с 13 по 21 строку из файла /etc/group
3. Посмотрите файлы mykernel и /etc/ddd, при этом стандартный вывод направьте в файл result.log, а ошибки в файл /tmp/ERRORS
4. Направьте строки с 13 до 20 из файла /etc/group в файл ~/lines, а потом добавьте в этот файл 16 строку из файла /etc/passwd.

2.12 12. bash №1: bash-completion, alias, type

2.12.1 12. bash №1: bash-completion, alias, type

В теме «Текстовый интерфейс пользователя» я рассказал про интерпретатор командной строки bash. Давайте разберём, что же он умеет.

```
[user@centos8 ~]$ t
Display all 115 possibilities? (y or n)
[user@centos8 ~]$ to
toc2cddb          toe          totem-video-thumbnailer
toc2cue           top          touch
toc2mp3           totem
[user@centos8 ~]$ touch
a/                  .nanorcold
.bash_history       new/
.bash_logout        passwd
```

Тогда я упомянул, что bash умеет дополнять команды – допустим, вы пишете часть команды (to, tou, touch file), нажимаете tab и bash дописывает команду или файл за вас. Если вариантов несколько – нажимаете tab два раза и видите все варианты. Но есть два типа дополнения – простое и продвинутое. В случае простого дополнения у вас просто дописывается сама команда или файл. Продвинутое дополнение может добавлять опции и значения.

```
[user@centos8 ~]$ man
Display all 6690 possibilities? (y or n)
[user@centos8 ~]$ tar -
-A -c -d -r -t -u -X
[user@centos8 ~]$ tar -■
```

Мы с вами на виртуалке поставили CentOS с графическим интерфейсом, то есть user-friendly версию, поэтому у нас сейчас стоит продвинутое дополнение. Из пройденных команд это можно увидеть по команде man – если написать man и нажать два раза tab, то bash скажет, что у нас больше 6 тысяч вариантов – всё это страницы документации, то есть, значения для man. Или, например, команда для архивации – tar – о ней мы поговорим в другой раз. Но, с точки зрения дополнения, если написать tar и нажать tab, то сперва появится дефис, а после двух нажатий tab появятся опции этой команды. В отличии от пройденных команд, где всякие ключи не обязательны, у tar ключи обязательны, поэтому bash сразу предлагает их написать.

```
[user@centos8 ~]$ sudo dnf -y remove bash-completion
[sudo] password for user:
Dependencies resolved.
=====
Package           Arch
=====
Removing:
  bash-completion      noarch
```

```
[user@centos8 ~]$ man
a/
.bash_history
.noarc
.bash_logout
.bash_profile
.bashrc
.nanorc
.nanorcold
new/
passwd
```

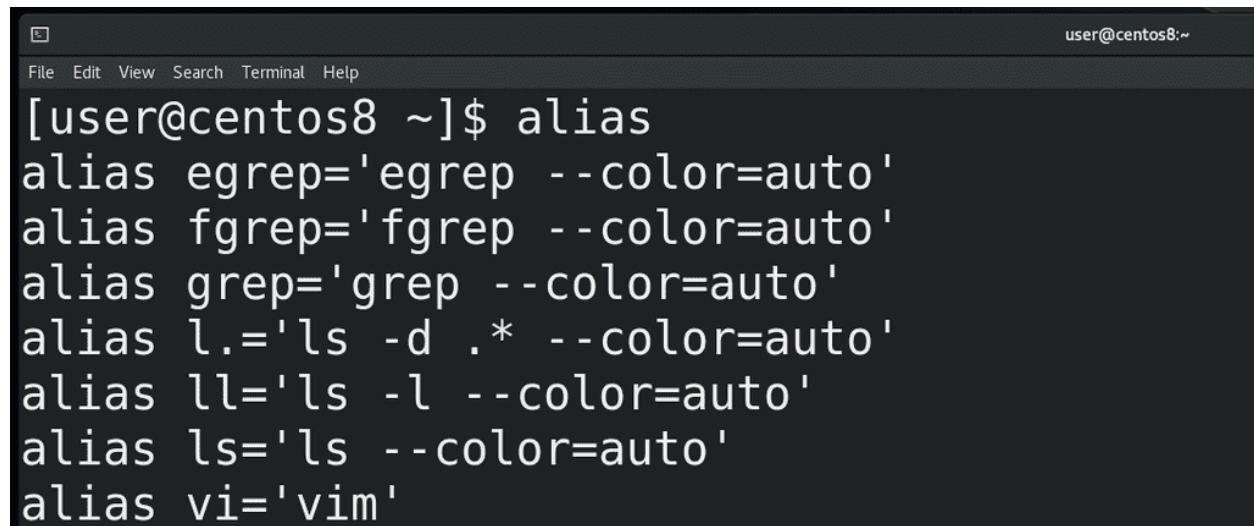
Теперь давайте рассмотрим простой вариант дополнения. Обычно, когда стоит минимальная система

без графического интерфейса, дополнение простое. Функционал продвинутого дополнения ставится с пакетом bash-completion. Давайте я удалю этот пакет и мы посмотрим что будет. Для этого я пишу:

```
sudo dnf -y remove bash-completion
```

и ввожу пароль моего пользователя. В рамках запущенной bash сессии ничего не изменится, но если открыть новое окно терминала – то тут у нас дополнение будет работать в урезанном виде. Например, пишу man, нажимаю tab – а он мне предлагает файлы вместо страниц документации. Или пишу tar, нажимаю tab – опять же файлы. Сами команды и файлы будут дописываться – если написать to и два раза нажать tab, то мы увидим все команды, начинающиеся на эти буквы. Но сразу после команд при нажатии tab будут предлагаться файлы, и не важно, работает команда с файлами или нет. Поэтому установим обратно bash-completion:

```
sudo dnf -y install bash-completion
```



The screenshot shows a terminal window with a dark background and light-colored text. At the top, there's a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The title bar says 'user@centos8:~'. Below the menu, the command [user@centos8 ~]\$ alias is run, followed by a list of aliases defined in the system:

```
[user@centos8 ~]$ alias
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l.='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
alias ls='ls --color=auto'
alias vi='vim'
```

Ещё в bash можно настроить алиасы. Алиасы дают возможность настроить какие-то свои команды на основе других. Если запустить команду

```
alias
```

можно увидеть список заданных алиасов для текущей bash сессии.

Я говорю bash сессия, но что это такое? Когда вы запускаете эмулятор терминала, у вас запускается bash сессия. Если вы запустите другое окно – будет другая сессия. Напишите bash – ещё одна сессия. Для каждой сессии bash запускается «с нуля», то есть он считывает файл настроек. Это значит, что если вы измените файл настроек, то в запущенном bash-е ничего не изменится. Но если написать bash или запустить новое окно, то есть запустить новую сессию bash, то новая сессия эти изменения увидит. Сейчас, на примере алиасов, разберёмся.

```
[user@centos8 ~]$ ls
.ICEauthority .vboxclient-display.pid
..lessht .vboxclient-display-svga-x11.pid
.bash_history .local .vboxclient-draganddrop.pid
.bash_logout .mozilla .vboxclient-seamless.pid
.bash_profile .nanorc .viminfo
.bashrc .nanorcold .zcompdump
.cache .passwd.swp .zshrc
.config .pki
.esd_auth .ssh
[user@centos8 ~]$ alias ls
alias ls='ls --color=auto'
```

Так вот, команда alias показала нам текущие алиасы. Тут мы видим пару знакомых команд, связанных с ls и vi. Итак, здесь написано, что команда l точки (l.) - это на самом деле команда ls -d .* --color=auto. ls показывает список файлов и директорий в текущей директории, ключ -d покажет сами директории, а не их содержимое, .* - все файлы и директории, у которых в начале названия есть точка, то есть скрытые файлы и директории. И --color=auto – раскрасить вывод. Так вот, если написать:

1.

мы увидим все скрытые файлы и директории в текущей директории. Если вы обратили внимание, у нас ls в принципе всегда раскрашивает вывод – а это потому что есть алиас на сам ls с опцией --color=auto. Тут же есть алиас на vi – то есть когда мы запускаем vi у нас запускается vim.

```
[user@centos8 ~]$ alias vi=nano
[user@centos8 ~]$ alias show="tail -5 /etc/passwd"
[user@centos8 ~]$ show
b.allen:x:1021:1002::/home/j
b.wayne1:x:1022:1002:Birthda
b.wayne12:x:1023:1003:Birthc
b.wayne123:x:1024:100:Birthc
b.wayne1234:x:1025:1002:Birt
[user@centos8 ~]$ show
bash: show: command not found...
Failed to search for file: Failed to lo
[user@centos8 ~]$
```

Давайте сами зададим какой-нибудь alias. Например:

alias vi=nano

Теперь у нас при запуске vi будет открываться nano – vi file. Или, допустим:

alias show="tail -5 /etc/passwd"

Обратите внимание, что я взял правую часть в кавычки, а в примере с nano я этого не делал. Сами подумайте, почему, если не знаете – попробуйте ввести без кавычек и проверить. Ну и если вам пока это сложно понять – спрашивайте в комментариях, я отвечу. Так вот, чуть ранее я говорил про bash сессии. И те алиасы, которые мы создали, сохранились только в текущей сессии. Это значит, что если запустить новое окно – то там наших алиасов уже не будет.

```

1 # .bashrc
2
3 # Source global definitions
4 if [ -f /etc/bashrc ]; then
5     . /etc/bashrc
6 fi
7
8 # User specific environment
9 PATH="$HOME/.local/bin:$HOME/bin:$PATH"
10

```

Если мы хотим, чтобы наши алиасы остались навсегда, мы должны их написать в файле настроек bash – `~/.bashrc`, который находится у нас в домашней директории:

```
nano ~/.bashrc
```

Как вы видите, этот файл не пустой, здесь есть какие-то настройки. Некоторые строчки начинаются со знака решётки (`#`). Такие строчки называются закомментированными – как правило, программы такие строчки не считывают, они больше нужны для людей.

```

14 # User specific aliases and functions
15
16 # My custom aliases
17
18 alias show="grep user /etc/passwd"
19

```

[user@centos8 ~]\$ show

tss:x:59:59:Account used by the **trousers** package to sandbox:/null:/sbin/nologin

qemu:x:107:107:qemu **user**:/:sbin/nologin

usbmuxd:x:113:113:usbmuxd **user**:/:sbin/nologin

saslauthd:x:991:76:Saslauthd **user**:/run/saslauthd:/sbin/nologin

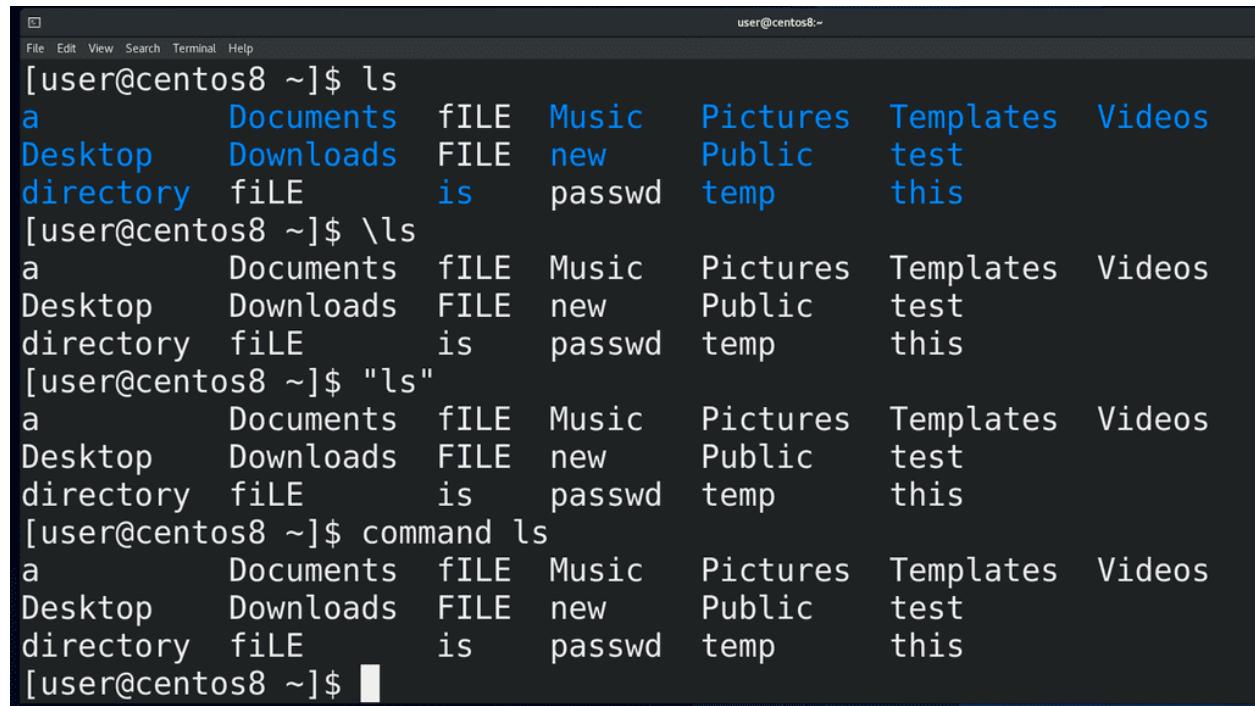
Обычно без разницы, где добавлять новые строчки, но для удобства давайте спустимся в самый низ и добавим там. Напишем комментарий:

```
# My custom aliases
```

Кстати, в nano можно нажать Esc+3 и nano сам добавит или уберёт знак решётки с текущей строки. Дальше я пишу свои алиасы, по одному на каждую строку:

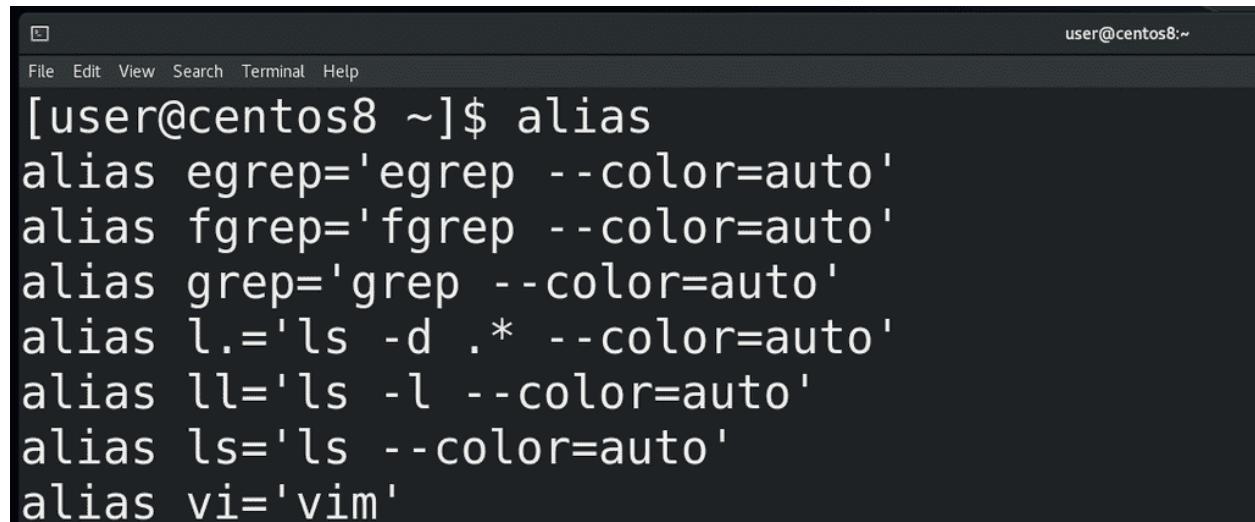
```
alias show="grep user /etc/passwd"
```

и закрываю файл. В текущей баш сессии ничего не изменится, поэтому я запускаю новое окно и уже там эти алиасы видны и работают.



```
[user@centos8 ~]$ ls
a      Documents FILE Music Pictures Templates Videos
Desktop Downloads FILE new Public test
directory file     is passwd temp   this
[user@centos8 ~]$ \ls
a      Documents FILE Music Pictures Templates Videos
Desktop Downloads FILE new Public test
directory file     is passwd temp   this
[user@centos8 ~]$ "ls"
a      Documents FILE Music Pictures Templates Videos
Desktop Downloads FILE new Public test
directory file     is passwd temp   this
[user@centos8 ~]$ command ls
a      Documents FILE Music Pictures Templates Videos
Desktop Downloads FILE new Public test
directory file     is passwd temp   this
[user@centos8 ~]$ [ ]
```

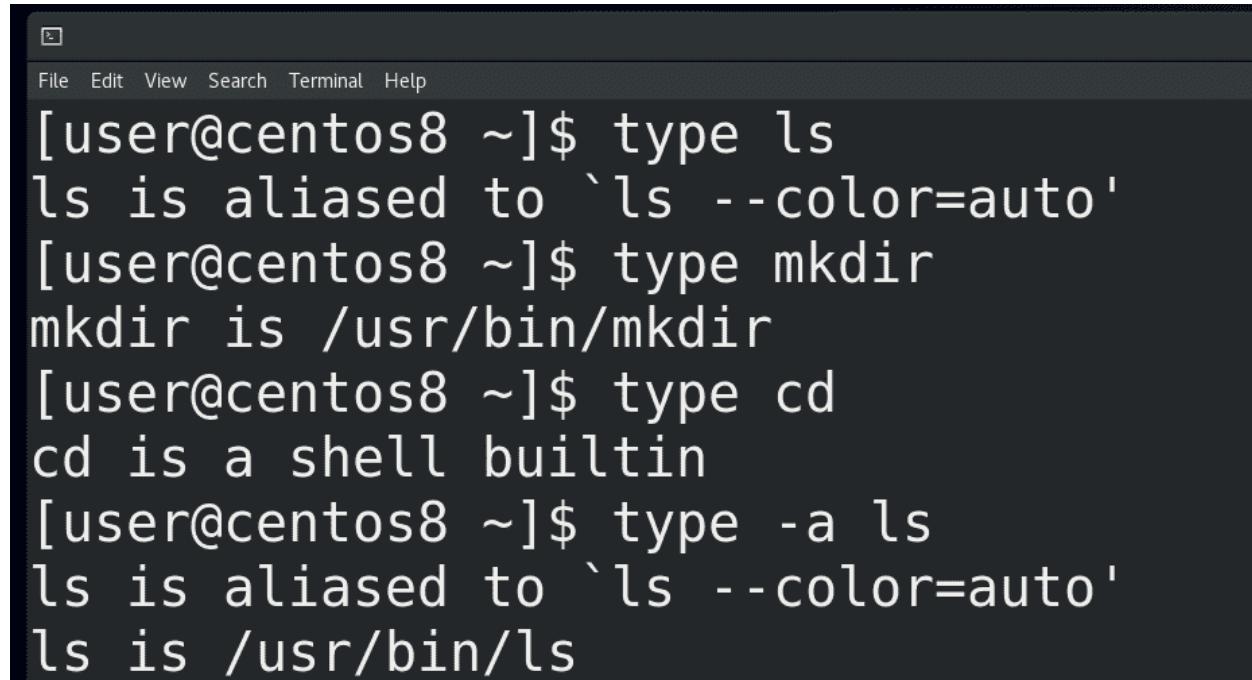
Алиасы нужны пользователям для упрощения работы, бывают сложные и длинные команды, которые вы часто должны вбивать. И чтобы облегчить себе жизнь, вы можете создать для себя алиасы. Но бывают случаи, когда вам нужен не алиас, а сама команда. Допустим, в системе есть алиас вместо vi запускать vim, или окрашивать вывод ls. А что, если мне нужно разок запустить обычный ls? Есть несколько способов это сделать – поставить обратный слэш перед командой - \ls, либо взять команду в кавычки – «ls», либо написать перед ней command - command ls.



```
[user@centos8 ~]$ alias
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l.='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
alias ls='ls --color=auto'
alias vi='vim'
```

Кстати, как вы, возможно, заметили, список алиасов от команды alias не малый, но в файле `~/.bashrc` их не было. Файл `.bashrc` распространяется только на моего пользователя, потому что он в моей домашней директории. Но есть файл `/etc/bashrc`, который распространяется на всех пользователей в системе.

Допустим, если вы хотите создать alias, который бы работал у всех пользователей – создаёте его именно в этом файле. Но, если полистать этот файл, вы всё равно не найдёте те же алиасы на ls. Поэтому ещё одна задачка для вас – постарайтесь найти файл, в котором указаны алиасы, которые у нас есть по умолчанию. Всё что нужно чтобы найти – мы уже проходили.



```
[user@centos8 ~]$ type ls
ls is aliased to `ls --color=auto'
[user@centos8 ~]$ type mkdir
mkdir is /usr/bin/mkdir
[user@centos8 ~]$ type cd
cd is a shell builtin
[user@centos8 ~]$ type -a ls
ls is aliased to `ls --color=auto'
ls is /usr/bin/ls
```

Мы разобрались, что bash может запускать как команды, так и алиасы. Сами команды можно разделить на 2 типа – внешние и внутренние. Какие-то команды встроены в bash, они называются внутренними – например, тот же cd. Но большинство команд лежат на файловой системе в специальных директориях и называются внешними. Чтобы как-то отличать, что это за команда – внутренняя, внешняя или аlias – есть команда type. Допустим:

```
type ls
type mkdir
type cd
```

В случае ls – это аlias, в случае mkdir bash показывает путь к программе, то есть это внешняя программа, а в случае с cd bash говорит, что это встроенная в оболочку команда. Но ведь ls – это аlias сам на себя, как понять, что там стоит за этим алиасом? Для этого можно использовать type с опцией -a:

```
type -a ls
```

Теперь мы видим, что ls – это /usr/bin/ls – то есть внешняя программа.

И так, мы узнали, что у нашего интерпретатора командной строки – bash – есть дополнения – простые и продвинутые. Также мы поговорили про алиасы, внутренние и внешние программы. Но bash может делать гораздо больше, поэтому будет несколько тем, посвящённых bash.

2.12.2 Практика

Вопросы

- Вы пишете команду, нажимаете tab – но вместо опций вы видите список файлов. Нормальное ли это поведение. Если нет, в чём заключается проблема и как её решить?

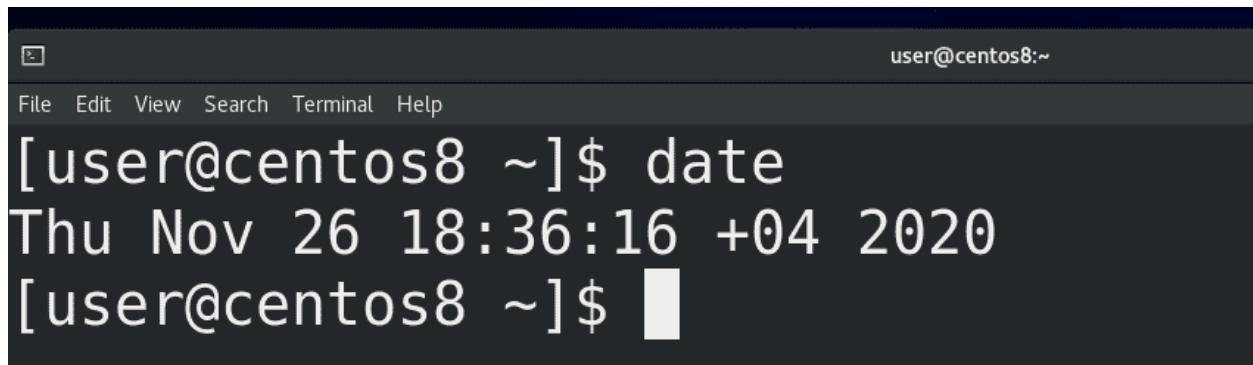
2. Вы запускаете команду ls file, но ничего не происходит. У вас подозрения, что кто-то подменил команду ls. Как это проверить?
3. Вы открыли файл ~/.bashrc, добавили в нём алиас, сохранили и закрыли текстовой редактор. Но алиас не работает. В чём может быть причина?

Задания

1. Создайте «команду» (алиас), которая бы сохраняла список алиасов в файле.
2. Создайте два алиаса с одинаковыми названиями в ~/.bashrc. Какой из них работает?
3. Создайте команду getlog, которая будет показывать последние 5 строк файла /var/log/kdump.log и одновременно добавлять вывод в файл mykernellog.

2.13 13. bash №2: переменные

2.13.1 13. bash №2: переменные

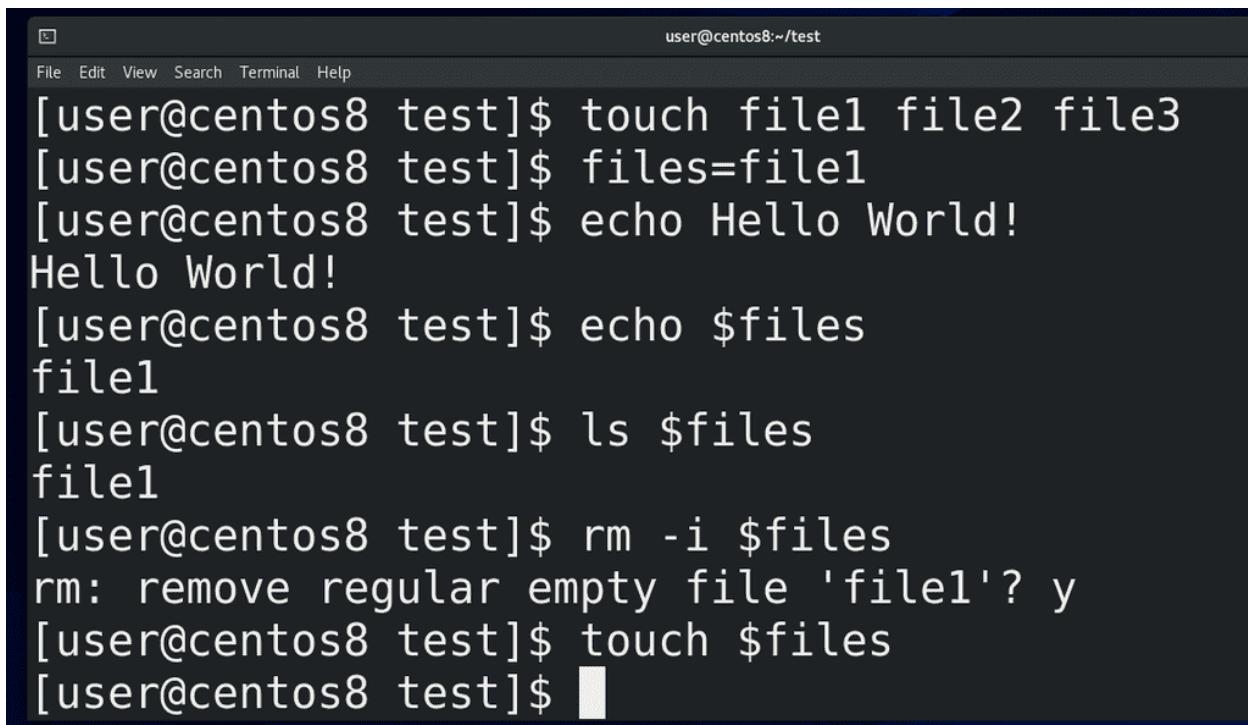


The screenshot shows a terminal window with a dark background. At the top, there is a menu bar with File, Edit, View, Search, Terminal, and Help. To the right of the menu, it says user@centos8:~. Below the menu, the command [user@centos8 ~]\$ date is entered, followed by its output: Thu Nov 26 18:36:16 +04 2020. The cursor is at the end of the date command.

```
[user@centos8 ~]$ date
Thu Nov 26 18:36:16 +04 2020
[user@centos8 ~]$ █
```

Когда мы говорим сегодня, мы подразумеваем 26 ноября. Если вы читаете эту тему завтра, то для вас слово сегодня будет означать 27 ноября. То есть, есть слово сегодня и оно может иметь различные значения. Постоянная часть, то есть слово сегодня – называется именем переменной или просто переменной, а та часть, которая меняется – 26 ноября или 27 ноября – называется значением переменной. Без переменных нельзя представить программирование и в bash-е они часто используются. Например, в эмуляторе терминала вы видите надпись user@centos8. Собственно user – это логин текущего пользователя, а centos8 – это имя системы. Если зайти другим пользователем или поменять имя системы, то будут отображаться другие значения.

В bash переменные нужны для нескольких задач. Начнём с так называемых локальных переменных, которые также называются shell variables. Это переменные, которые мы сами создаём и нужны они нам для каких-то своих задач.



```
user@centos8:~/test
File Edit View Search Terminal Help
[user@centos8 test]$ touch file1 file2 file3
[user@centos8 test]$ files=file1
[user@centos8 test]$ echo Hello World!
Hello World!
[user@centos8 test]$ echo $files
file1
[user@centos8 test]$ ls $files
file1
[user@centos8 test]$ rm -i $files
rm: remove regular empty file 'file1'? y
[user@centos8 test]$ touch $files
[user@centos8 test]$
```

Для примера создадим переменную files. Я создаю 3 файла:

```
touch file1 file2 file3
```

и пишу:

```
files=file1
```

Обратите внимание, никаких пробелов, чтобы bash не подумал, что я запускаю программу files. Итак, я создал переменную files и дал ей значение file1. Значение переменной я могу увидеть с помощью команды echo. Echo у нас выводит текст в stdout:

```
echo Hello World!
```

и используется во многих задачах, но о нём подробно поговорим в другой раз. Пока что выведем значение переменной с помощью echo:

```
echo $files
```

files – имя переменной, а чтобы взять значение нужно перед именем поставить знак доллара. echo нам показал значение file1. Мы можем использовать эту переменную с другими командами, например, чтобы посмотреть:

```
ls $files
rm -i $files
```

или удалить файл. Но нужно понимать, что значение переменной – просто текст и с файлом он никак не связан. Просто bash превращает переменную в её значение, прежде чем запустить команду. В итоге в терминале получается команда:

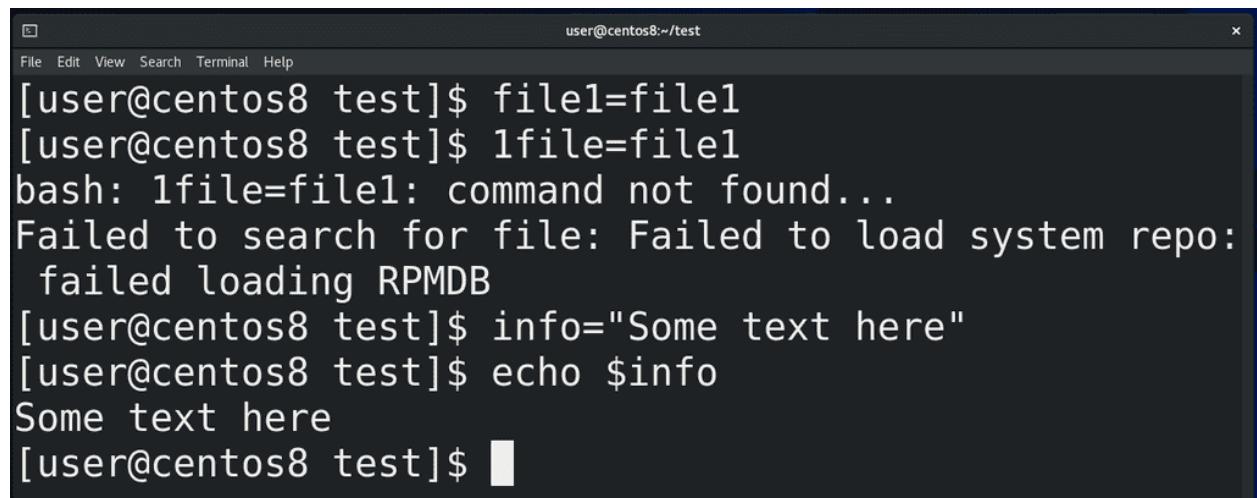
```
rm file1
```

От того, что я удалил файл, значение переменной никуда не делось:

```
echo $files
```

поэтому я могу снова использовать эту переменную:

```
touch $files
```



The screenshot shows a terminal window titled "user@centos8:~/test". The command history and output are as follows:

```
[user@centos8 test]$ file1=file1
[user@centos8 test]$ lfile=file1
bash: lfile=file1: command not found...
Failed to search for file: Failed to load system repo:
failed loading RPMDB
[user@centos8 test]$ info="Some text here"
[user@centos8 test]$ echo $info
Some text here
[user@centos8 test]$
```

Имя переменной может содержать цифры, но не может начинаться с неё:

```
file1=file1
1file=file1 (неправильно)
```

Также в имени переменной нельзя использовать специальные символы, точку, дефис и всё такое:

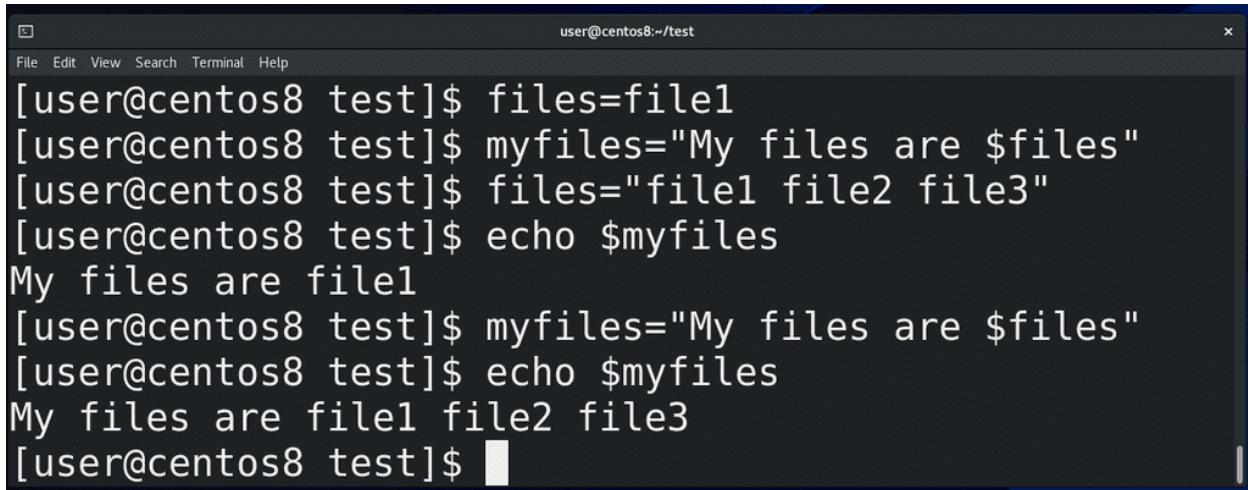
```
file.1=file1 (неправильно)
file-1=file (неправильно)
```

Но имя можно писать строчными, заглавными и их комбинацией:

```
pRiFfKi=2007
echo $pRiFfKi
```

И помните – всё это регистрозависимо. А в значениях переменных можно указывать всё что угодно, но если там есть пробелы и всякие символы – нужно брать значение в кавычки:

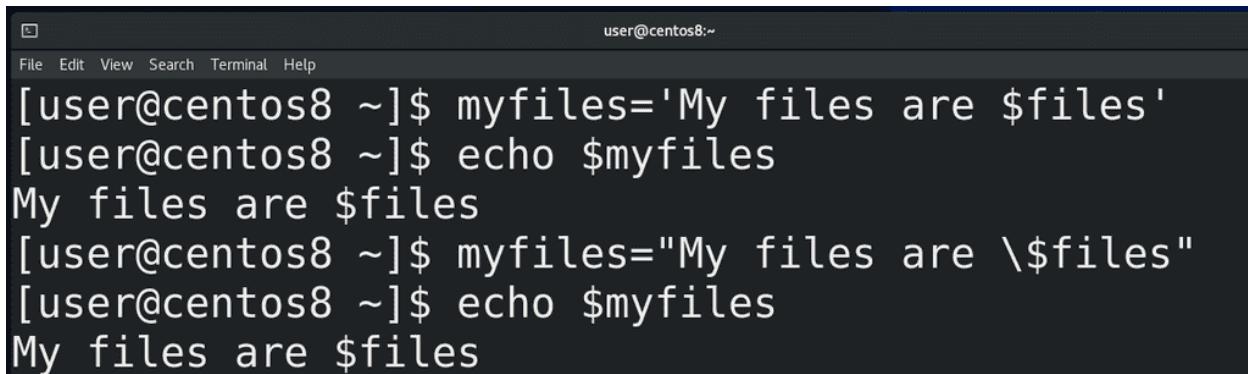
```
info="Some text here"
```



```
[user@centos8 test]$ files=file1
[user@centos8 test]$ myfiles="My files are $files"
[user@centos8 test]$ files="file1 file2 file3"
[user@centos8 test]$ echo $myfiles
My files are file1
[user@centos8 test]$ myfiles="My files are $files"
[user@centos8 test]$ echo $myfiles
My files are file1 file2 file3
[user@centos8 test]$ █
```

Можно при задавании значения переменной использовать значение другой переменной:

```
myfiles="My files are $files"
files="file1 file2 file3"
echo $myfiles
```



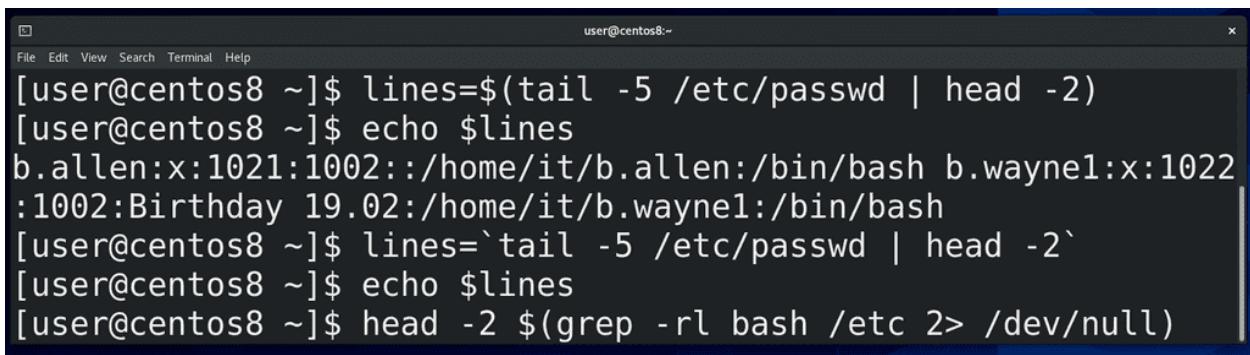
```
[user@centos8 ~]$ myfiles='My files are $files'
[user@centos8 ~]$ echo $myfiles
My files are $files
[user@centos8 ~]$ myfiles="My files are \$files"
[user@centos8 ~]$ echo $myfiles
My files are $files
```

Как тут видно, мы указали знак доллара и это воспринялось как значение переменной. Если же мы не хотим, чтобы наши значения обрабатывались, чтобы они были просто текстом, то берём значения в одинарные кавычки:

```
myfiles='My files are $files'
echo $myfiles
```

либо экранируем знак доллара:

```
myfiles="My files are \$files"
echo $myfiles
```



```
[user@centos8 ~]$ lines=$(tail -5 /etc/passwd | head -2)
[user@centos8 ~]$ echo $lines
b.allen:x:1021:1002::/home/it/b.allen:/bin/bash b.wayne1:x:1022
:1002:Birthday 19.02:/home/it/b.wayne1:/bin/bash
[user@centos8 ~]$ lines=`tail -5 /etc/passwd | head -2`
[user@centos8 ~]$ echo $lines
[user@centos8 ~]$ head -2 $(grep -rl bash /etc 2> /dev/null)
```

Бывает удобно указывать в значении переменной какую-то команду. Для этого нужно начинать значение с символа доллара и указать команду в скобках:

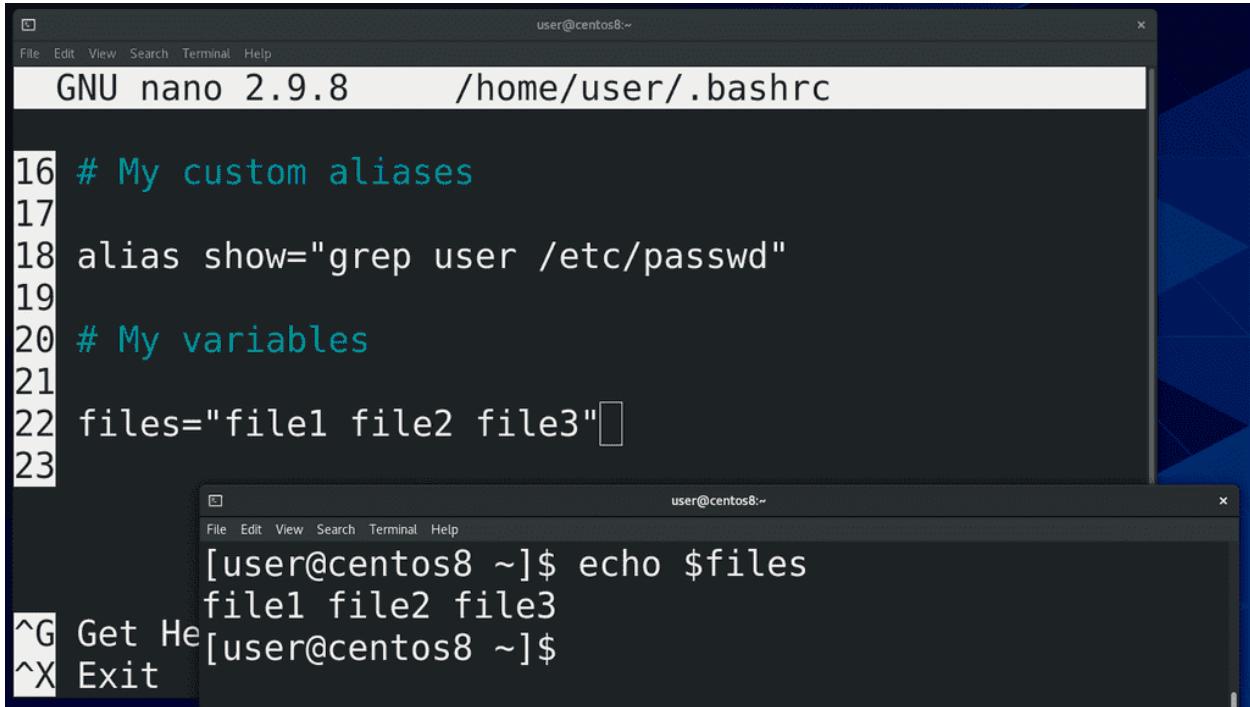
```
lines=$(tail -5 /etc/passwd | head -2)
echo $lines
```

Либо использовать обратные кавычки:

```
lines=`tail -5 /etc/passwd | head -2`
```

но это устаревший метод. Ещё таким образом можно одни команды внедрять в другие, например:

```
head -2 $(grep -rl bash /etc/ 2> /dev/null)
```



The top terminal window shows the contents of the `/home/user/.bashrc` file being edited with `GNU nano 2.9.8`. The file contains:

```
16 # My custom aliases
17
18 alias show="grep user /etc/passwd"
19
20 # My variables
21
22 files="file1 file2 file3"
```

The bottom terminal window shows the output of the command `echo $files`, which prints `file1 file2 file3`.

Кстати, можно использовать `tab` для дополнения переменных. Так вот, те переменные, которые мы задали, существуют только в текущей bash сессии. Обычно локальные переменные используются во всяких скриптах, которые мы рассмотрим в другой раз, и в обычной работе с терминалом их используют не так часто. Но если вам всё же нужно, чтобы переменная работала и в других окнах, вы можете сохранить её значение в файле `~/.bashrc`, как мы это делали для алиасов. Открываем файл:

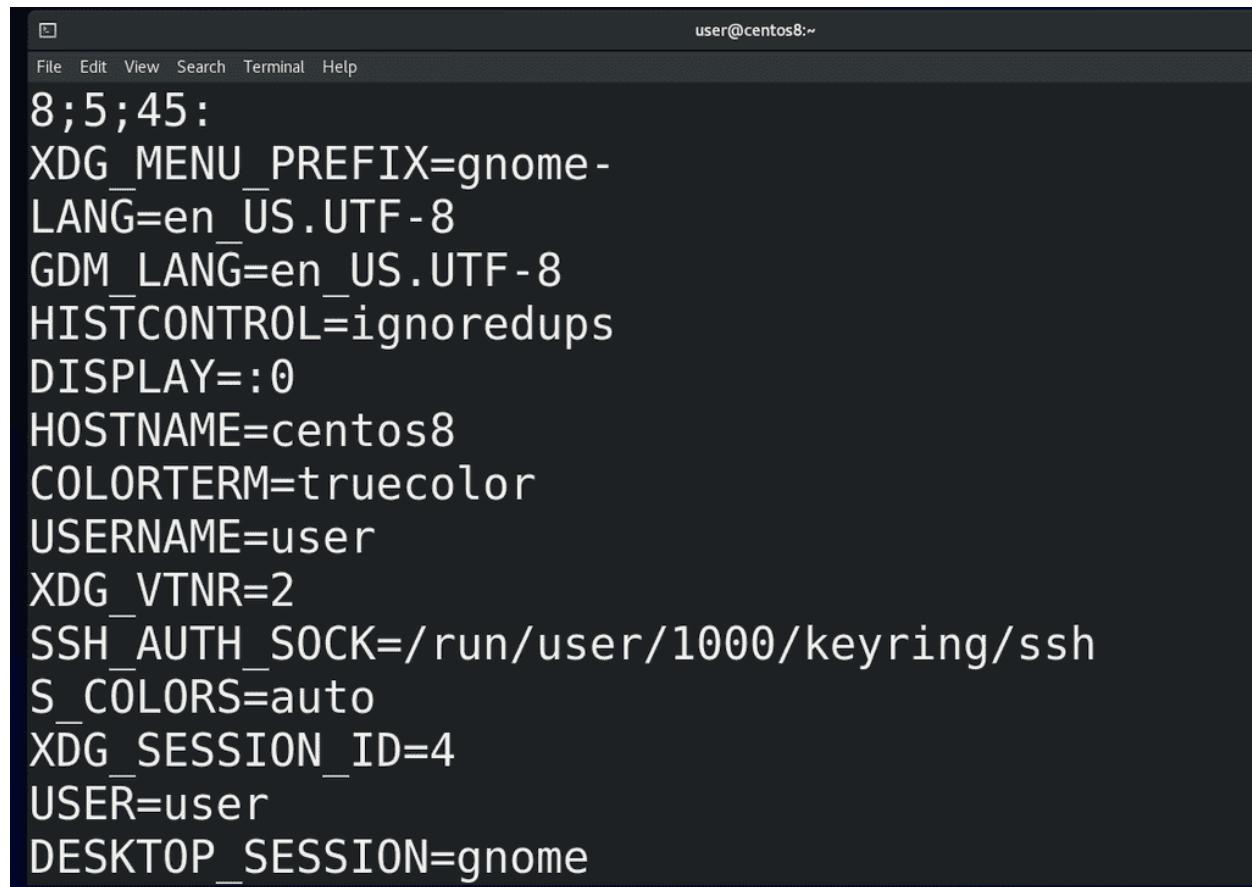
```
nano ~/.bashrc
```

добавляем запись:

```
files="file1 file2 file3"
```

и сохраняем. В новых сессиях у нас теперь будет задана эта переменная:

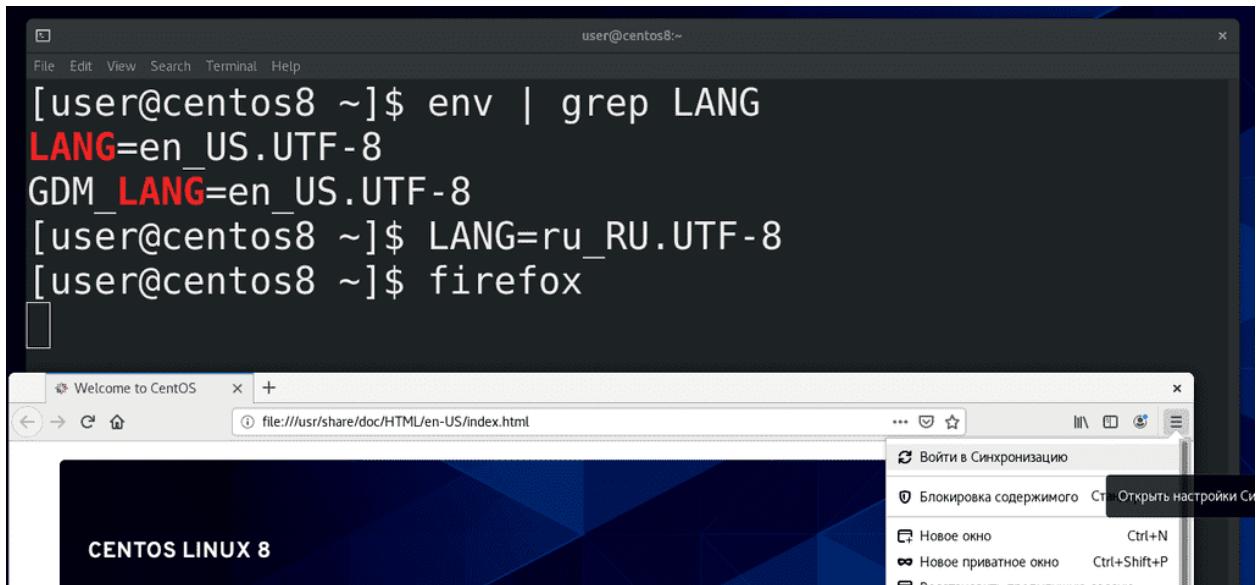
```
echo $files
```



The screenshot shows a terminal window titled 'user@centos8:~'. The window contains a list of environment variables. At the top, it says '8;5;45:'. Below that is a list of variables: XDG_MENU_PREFIX=gnome-, LANG=en_US.UTF-8, GDM_LANG=en_US.UTF-8, HISTCONTROL=ignoredups, DISPLAY=:0, HOSTNAME=centos8, COLORTERM=truecolor, USERNAME=user, XDG_VTNR=2, SSH_AUTH_SOCK=/run/user/1000/keyring/ssh, S_COLORS=auto, XDG_SESSION_ID=4, USER=user, DESKTOP_SESSION=gnome.

```
8;5;45:  
XDG_MENU_PREFIX=gnome-  
LANG=en_US.UTF-8  
GDM_LANG=en_US.UTF-8  
HISTCONTROL=ignoredups  
DISPLAY=:0  
HOSTNAME=centos8  
COLORTERM=truecolor  
USERNAME=user  
XDG_VTNR=2  
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh  
S_COLORS=auto  
XDG_SESSION_ID=4  
USER=user  
DESKTOP_SESSION=gnome
```

В противовес локальным переменным, которые используются в рамках текущей bash сессии или скрипта и не особо-то нужны другим программам, существуют переменные окружения – environment variables. С помощью команды env можно увидеть их список. Как вы заметили, все они указаны заглавными буквами. Такие переменные нужны для передачи каких-то значений различным программам.



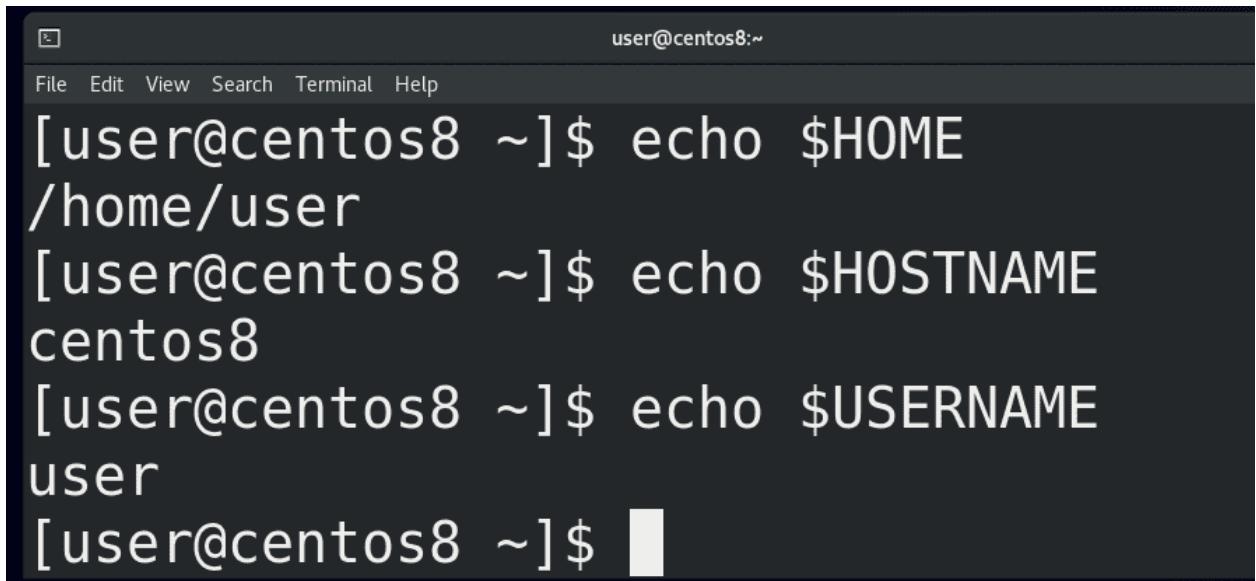
Например, есть переменная LANG:

```
env | grep LANG
```

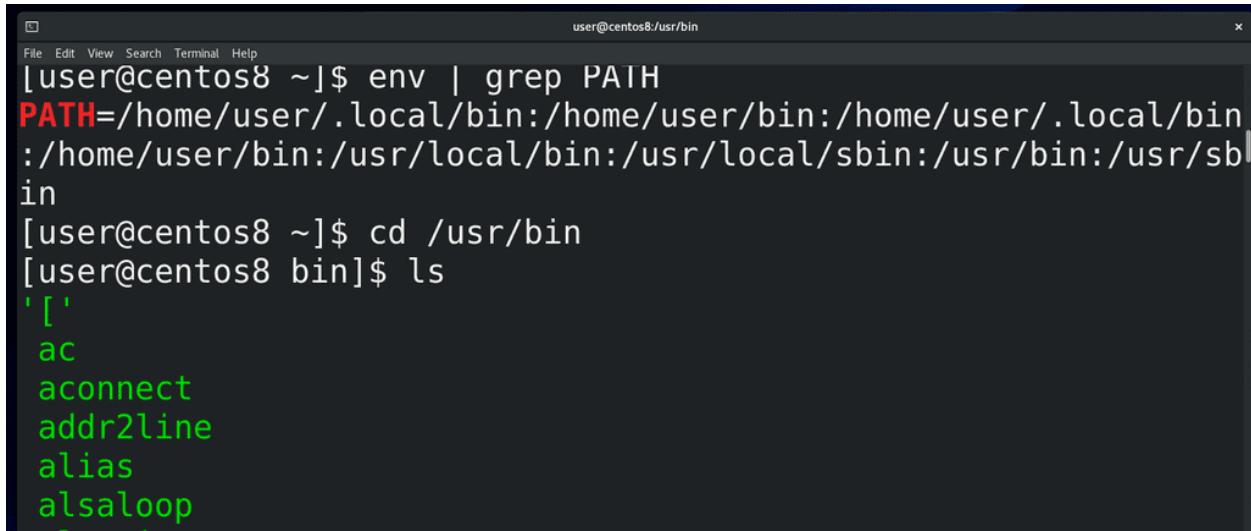
которая указывает язык для запускаемых программ. Сейчас стоит английский и, если я запущу, допустим, firefox, он запустится на английском языке. Если я поменяю значение этой переменной на русский:

```
LANG=ru_RU.UTF-8
```

и запущу firefox через ту же bash сессию – то увижу, что Firefox теперь на русском. То есть, с помощью таких переменных можно настраивать окружение.



Возвращаясь к env, тут есть пара переменных, которые могут нам пригодится в дальнейшем. Например, HOME – указывает на домашнюю директорию текущего пользователя, HOSTNAME на имя системы, USERNAME на имя текущего пользователя и т.п.



```
user@centos8 ~]$ env | grep PATH
PATH=/home/user/.local/bin:/home/user/bin:/home/user/.local/bin
:/home/user/bin:/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin
in
[user@centos8 ~]$ cd /usr/bin
[user@centos8 bin]$ ls
'['
ac
aconnect
addr2line
alias
alsaloop
...
```

Ещё одна примечательная переменная – PATH:

```
env | grep PATH
```

Обратите внимание, что в этой переменной перечислены директории. Опустим директории /home/user/bin и /home/user/.local/bin , так как их у нас пока нет, и зайдём в /usr/bin:

```
cd /usr/bin
```

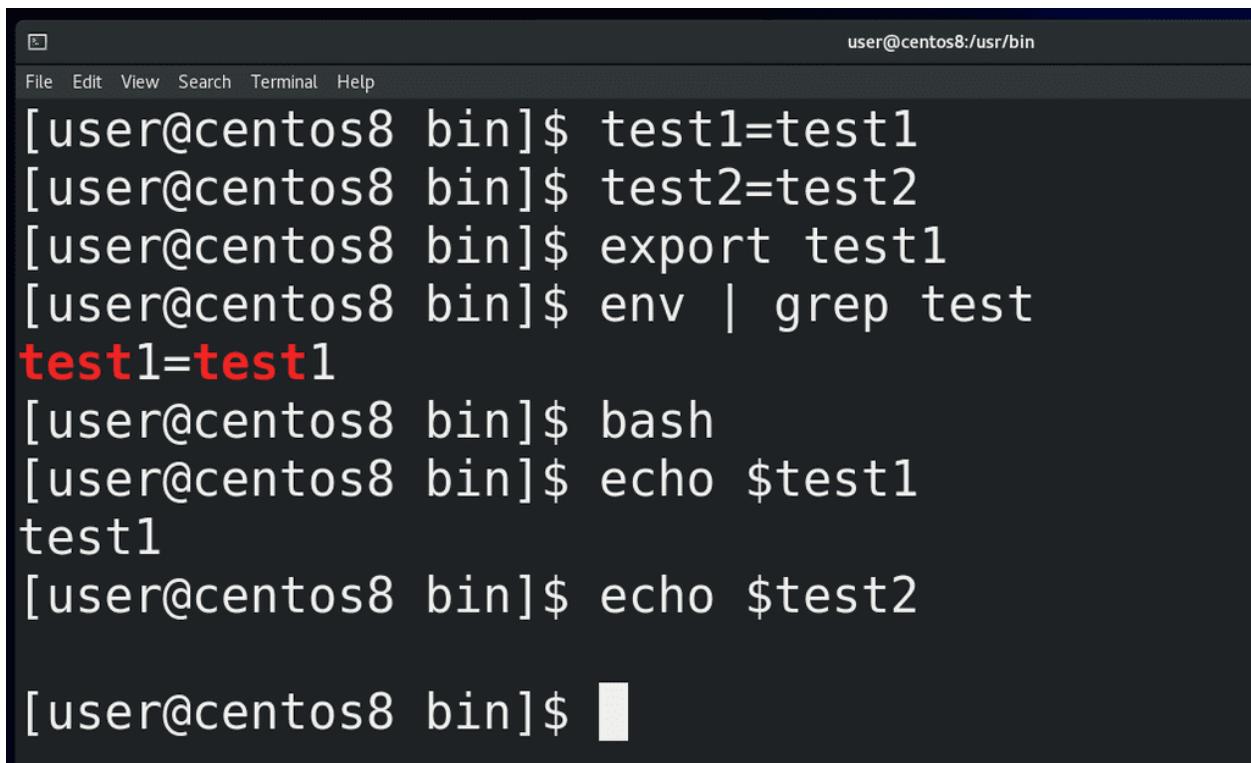
Здесь у нас огромное количество файлов – всё это исполняемые файлы, то есть программы. В прошлый раз я говорил, что команды, которые запускает bash – это либо алиасы, либо внутренние команды bash, либо внешние команды. Так вот, в переменной PATH перечислены директории, где искать эти внешние программы. То есть, когда я пишу mkdir, bash ищет во всех директориях PATH наличие такого файла и, если находит, запускает.

Так вот, в отличии от локальных переменных, переменные среды передаются дочерним процессам. Например, до этого мы поменяли значение переменной LANG и firefox при запуске прочитал значение этой переменной, потому что она переменная окружения. Мы можем превратить локальную переменную в переменную окружения, чтобы использовать её в дочерних процессах, с помощью команды export. Например:

```
files=file1
export files
```

либо разом:

```
export files=file1
```



The screenshot shows a terminal window with the title bar "user@centos8:/usr/bin". The menu bar includes "File", "Edit", "View", "Search", "Terminal", and "Help". The terminal content is as follows:

```
[user@centos8 bin]$ test1=test1
[user@centos8 bin]$ test2=test2
[user@centos8 bin]$ export test1
[user@centos8 bin]$ env | grep test
test1=test1
[user@centos8 bin]$ bash
[user@centos8 bin]$ echo $test1
test1
[user@centos8 bin]$ echo $test2

[user@centos8 bin]$ █
```

Это хорошо видно на примере дочерних bash сессий. Например, создадим локальные переменные test1 и test2:

```
test1=test1
test2=test2
```

Одну из них экспортнём:

```
export test1
```

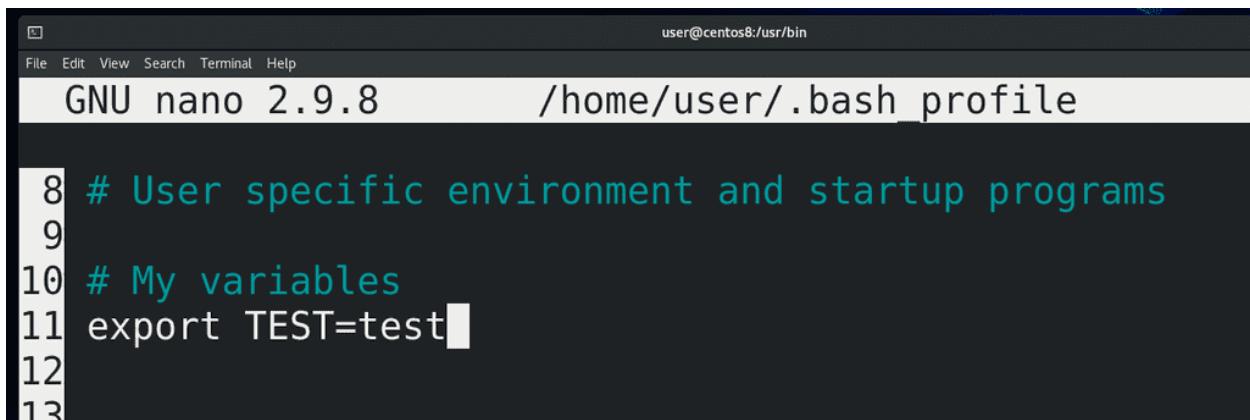
а другую нет. Тут же можем запустить env и увидеть, что здесь появился test1. Запустив другое окно мы test1 там не найдём, но если запустить дочернюю сессию bash:

```
bash
echo $test1
echo $test2
```

мы увидим, что у test1 здесь есть значение, а у test2 нет. Потому что дочерние сессии получают только значения переменных окружения. И чтобы задать переменную окружения на постоянно, опять же, нужно редактировать файл.

Мы с вами уже работали с файлом `~/.bashrc`, и там можно задать переменную. Но основное предназначение `bashrc` – настройка алиасов и всяких функций bash для эмулятора терминала. Правильнее говоря, в `bashrc` задаются настройки bash для интерактивной оболочки, в которой не нужно логиниться – то есть для эмулятора терминала. При запуске он у нас не требует логина и при этом нам нужно с ним вручную работать, то есть интерактивно. То есть, интерактивная оболочка без логина. А, скажем, firefox обычно мы запускаем не через эмулятор терминала, а через лаунчер. И для случаев, когда нам нужно использовать какие-то переменные независимо от эмулятора терминала, то есть независимо от интерактивной оболочки, нам нужен другой файл – `~/.bash_profile`. Но, на самом деле, во многих дистрибутивах этот файл при запуске также считывает настройки с `~/.bashrc`, из-за чего технически без разницы, где добавлять переменные. Также в каких-то дистрибутивах этот файл обычно называет

.profile. Так вот, переменную мы можем создать как в `~/.bashrc`, так и в `~/.bash_profile`, или вообще создать свой файл со всеми своими алиасами и переменными.



```
user@centos8:/usr/bin
File Edit View Search Terminal Help
GNU nano 2.9.8      /home/user/.bash_profile

8 # User specific environment and startup programs
9
10 # My variables
11 export TEST=test
12
13
```

Но я этого делать не буду, просто добавлю свою переменную в `~/.bash_profile`:

```
nano ~/.bash_profile
```

```
#My variables
export TEST=test
```

Единственное что, этот файл считывается в момент моего логина, а значит недостаточно просто открыть новую bash сессию, нужно перезайти в систему. В этом плане `~/.bashrc` удобнее. Ну и если мы хотим, чтобы наши переменные работали не только для нашего пользователя, но и для всех других пользователей, то настраивать эти переменные нужно в файлах `/etc/profile` и `/etc/bashrc`. А если мы не хотим зависеть от bash-а, а использовать любую другую оболочку, то лучше указывать в `/etc/environment`.

Так вот, мы с вами разобрались, что bash умеет работать с переменными, что эти переменные бывают локальными и переменными окружения и разобрали, где и как их задавать. Там где есть переменные, там же будут и условия, циклы и прочее, что, действительно, поддерживает bash. Всё это мы будем разбирать в другие разы.

2.13.2 Практика

Вопросы

1. Чем отличаются локальные переменные от переменных окружения?
2. Я задал переменную в `~/.bashrc` (`files=file1`) и она не является переменной окружения. Но при этом она работала в дочерних bash сессиях. Почему?

Задания

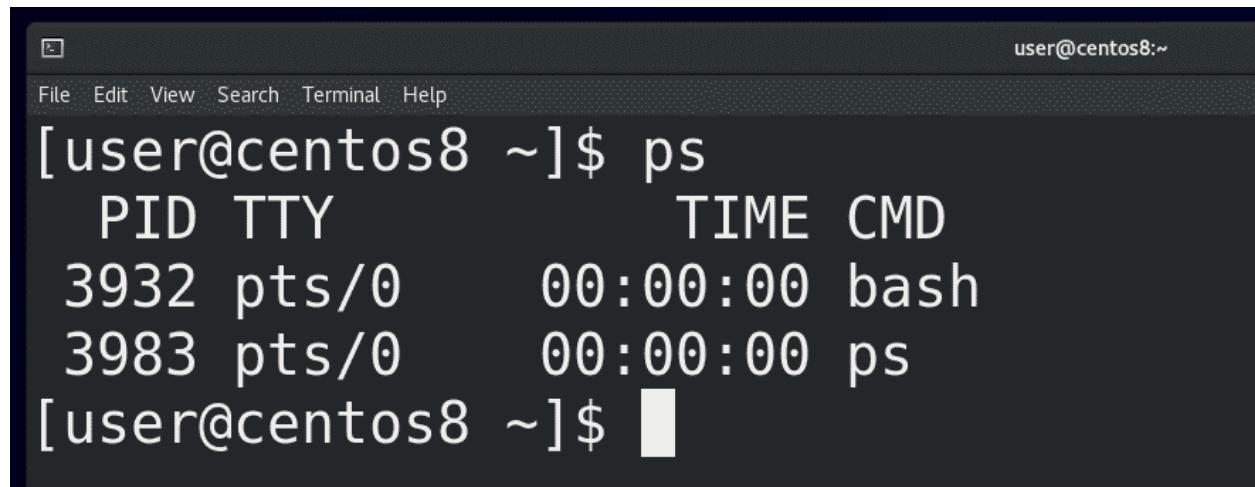
1. Создайте переменную `curtime`, которая будет брать своё значение от команды `date`. Запишите переменную в `~/.bashrc`. Выведите значение переменной. Показывает ли она актуальное время и, если нет, почему?
2. Создайте переменную, значение которой показывает имя пользователя и имя системы, как в терминале - `user@centos8`.

2.14 14. Процессы №1: Информация о процессах №1

2.14.1 14. Процессы №1: Информация о процессах №1

Как мы уже выяснили, программы хранятся в файловой системе на накопителе – т.е. жёстком диске или ssd. Когда мы запускаем программу, она загружается в оперативную память, так как скорость чтения с жёсткого диска или даже ssd относительно низкая, а процессор работает на больших скоростях. Как правило, большие программы загружаются в оперативную память не полностью, а по мере необходимости. При этом, для каждой программы создаётся иллюзия, что она – единственная в оперативной памяти, то есть для неё создаётся так называемая «виртуальная память». Также программы при запуске загружают какие-то файлы, будь то файлы настроек или пользовательские файлы – как например, если мы запускаем nano file, то в память загружаются как сам /usr/bin/nano, его настройки /etc/nanorc и ~/.nanorc, всякие библиотеки, необходимые для работы nano и сам файл, который мы открываем. Кроме этого также запускаемой программе передаются переменные окружения и ещё много всего. Ну и находясь в оперативке, эта программа делает какие-то вычисления с помощью центрального процессора, обрабатывает данные и сохраняет на диске. И совокупность всего этого называется процессом.

Иногда одной программе нужно бывает выполнить несколько операций параллельно. Представьте себе сложное математическое уравнение – есть всякие скобки, умножения и прочее. Такое уравнение можно разделить на составляющие и компьютер может разом выполнить все составляющие, а потом, используя результаты, получить простое уравнение и выполнить его. Или, допустим, веб сервер – к нему обращаются много клиентов, и каждого из них он должен обслужить и желательно параллельно. Для этого один процесс может разделяться на так называемые потоки – все они используют общую виртуальную память. У каждого процесса есть как минимум один поток.



```
[user@centos8 ~]$ ps
 PID TTY          TIME CMD
 3932 pts/0        00:00:00 bash
 3983 pts/0        00:00:00 ps
 [user@centos8 ~]$
```

И так, выполняемая программа – это процесс. Начнём с того, что администратору важно видеть список процессов. Для этого есть несколько способов, начнём с утилиты ps. Если просто запустить:

```
ps
```

мы увидим список процессов, запущенных в этом терминале. Как вы заметили, вывелось 2 строчки – bash и ps. При том, что ps у нас выполнился за какие-то доли секунды, он у нас всё равно виден в выводе – потому что он делает эдакий скриншот процессов именно в момент выполнения, поэтому и видит сам себя.

DESCRIPTION

ps displays information about a selection of the active processes. If you want a repetitive update of the selection and the displayed information, use **top(1)** instead.

This version of **ps** accepts several kinds of options:

- 1 UNIX options, which may be grouped and must be preceded by a dash.
- 2 BSD options, which may be grouped and must not be used with a dash.
- 3 GNU long options, which are preceded by two dashes.

Manual page ps(1) line 9 (press h for help or q to quit)

Вообще, ps работает с 3 видами ключей: юниксовыми – они обычно начинаются на один дефис (-), BSD-шные – вовсе без дефиса и GNU-шные – как правило это слова, поэтому, чтобы не счесть их за комбинацию букв, используется два дефиса. Если посмотреть документацию:

```
man ps
```

можно заметить очень много дублирующихся ключей. Но документация по ps огромная, да и все ключи знать не нужно. Достаточно выучить какую-то одну комбинацию, которая подойдёт в большинстве случаев:

```
ps -ef
```

а если вам понадобится что-то конкретное, то всегда можно погуглить или найти в мане.

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	1	0	0	18:31	?	00:00:01	/usr/lib/systemd/systemd >
root	2	0	0	18:31	?	00:00:00	[kthreadd]
root	3	2	0	18:31	?	00:00:00	[rcu_gp]
root	4	2	0	18:31	?	00:00:00	[rcu_par_gp]
root	5	2	0	18:31	?	00:00:00	[kworker/0:0-events]
root	6	2	0	18:31	?	00:00:00	[kworker/0:0H-kblockd]
root	8	2	0	18:31	?	00:00:00	[mm_percpu_wq]
root	9	2	0	18:31	?	00:00:00	[ksoftirqd/0]

Как видите, вывод у ps довольно большой и не помещается на экране, поэтому урезается сбоку. Чтобы мы могли нормально прочесть, мы можем передать вывод ps команде less. Правда по умолчанию less переносит текст на новые строки, из-за чего сбиваются столбцы, поэтому less лучше использовать с ключом -S, который не переносит строки. В итоге:

```
ps -ef | less -S
```

Давайте разберём, что означают ключи и как читать вывод. Ключ -e выводит все процессы всех пользователей:

```
ps -e
```

Да, процессы запускаются от имени пользователей. От этого зависит какие права будут у процесса. Допустим, если я запускаю программу nano от пользователя user, то программа сможет работать с моими файлами. А ключ f:

```
ps -f
```

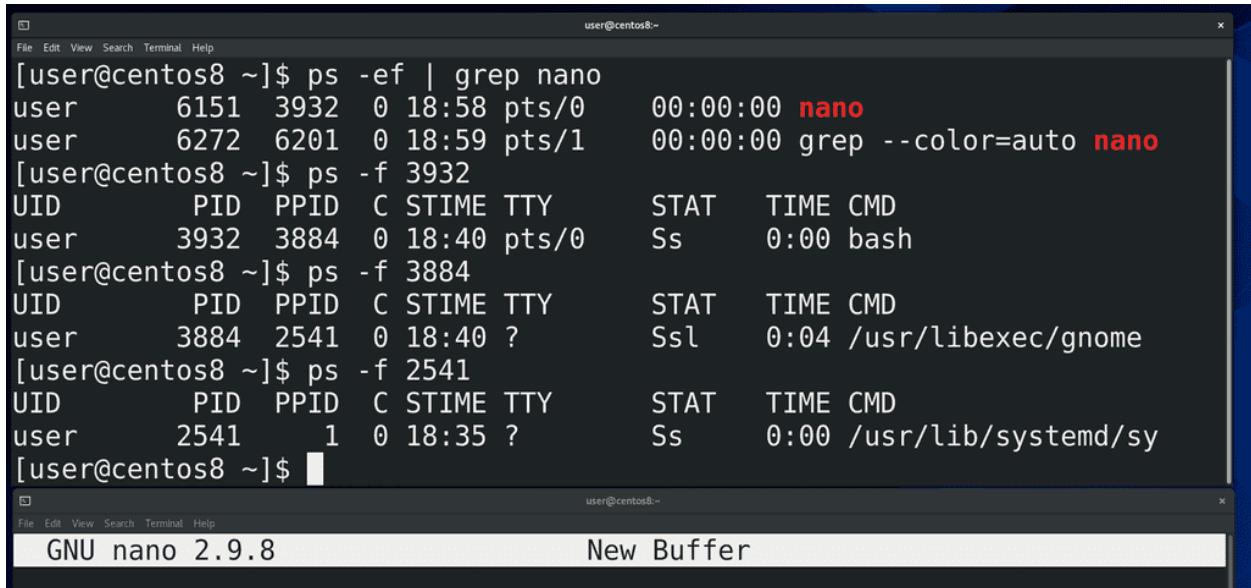
показывает чуть больше информации о процессе. Давайте пройдёмся по столбикам:

```
ps -ef | less -S
```

gdm	2393	2294	0	18:32	tty1	00:00:00	/usr/libexec/ibus-engine
gdm	2399	2239	0	18:32	tty1	00:00:00	/usr/libexec/gsd-share
gdm	2406	2239	0	18:32	tty1	00:00:00	/usr/libexec/gsd-smart
gdm	2410	2239	0	18:32	tty1	00:00:00	/usr/libexec/gsd-sound
gdm	2414	2239	0	18:32	tty1	00:00:00	/usr/libexec/gsd-wacom
colord	2437	1	0	18:32	?	00:00:00	/usr/libexec/colord
root	2531	2171	0	18:35	?	00:00:00	gdm-session-worker [pan
user	2541	1	0	18:35	?	00:00:00	/usr/lib/systemd/system
user	2547	2541	0	18:35	?	00:00:00	(sd-pam)
user	2557	2541	0	18:35	?	00:00:00	/usr/bin/pulseaudio --o
user	2562	1	0	18:35	?	00:00:00	/usr/bin/gnome-keyring-
user	2570	2541	0	18:35	?	00:00:00	/usr/bin/dbus-daemon --
user	2578	2531	0	18:35	tty2	00:00:00	/usr/libexec/gdm-waylari
:							

Первое – UID – user id - пользователь, который запустил процесс. Большинство процессов в системе запущены от пользователя root – его также называют суперпользователем – это юзер, у которого есть все права на систему. По возможности, люди стараются не использовать рута везде. Если у программы будет какой-то баг или уязвимость и если она запущена от рута, то программа может сильно навредить системе. Поэтому для программ, которые не требуют особых прав, обычно создают сервисных пользователей. Как правило при установке программы она сама всё это настраивает. Ну и наконец у нас тут есть программы, запущенные от нашего пользователя. Как видите, я вроде ничего кроме эмулятора терминала не запускал, а в системе уже пару сотен процессов.

Второй столбик – PID – process id – идентификатор процесса. Он уникальный для каждого процесса, но совпадает для потоков одного процесса. Когда программа завершается, она освобождает номер и через какое-то время другая программа может использовать этот номер. С помощью этих номеров мы можем управлять процессами.



```
[user@centos8 ~]$ ps -ef | grep nano
user      6151  3932  0 18:58 pts/0    00:00:00 nano
user      6272  6201  0 18:59 pts/1    00:00:00 grep --color=auto nano
[user@centos8 ~]$ ps -f 3932
UID      PID  PPID  C STIME TTY      STAT   TIME CMD
user     3932  3884  0 18:40 pts/0    Ss      0:00 bash
[user@centos8 ~]$ ps -f 3884
UID      PID  PPID  C STIME TTY      STAT   TIME CMD
user     3884  2541  0 18:40 ?        Ssl     0:04 /usr/libexec/gnome
[user@centos8 ~]$ ps -f 2541
UID      PID  PPID  C STIME TTY      STAT   TIME CMD
user     2541      1  0 18:35 ?        Ss      0:00 /usr/lib/systemd/sy
[user@centos8 ~]$ █
```

GNU nano 2.9.8 New Buffer

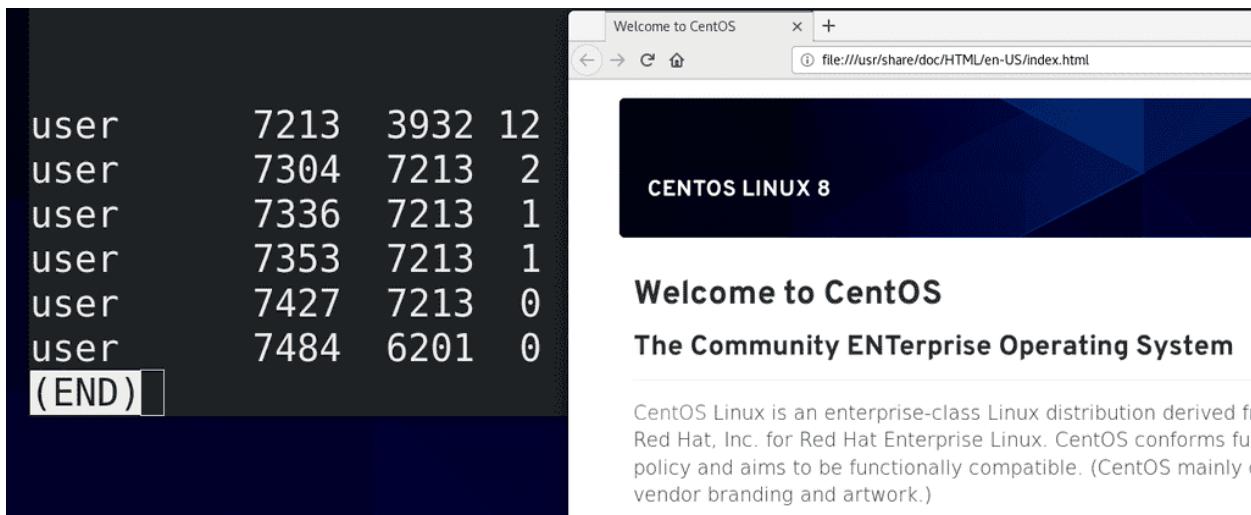
Третий столбик – PPID – parent process id – идентификатор родительского процесса. Почти все процессы в системе были запущены каким-то другим процессом. Допустим, когда мы запускаем эмулятор терминала, а в нём nano – то родительским процессом для nano является bash, который запущен в этом эмуляторе терминала:

```
ps -ef | grep nano
```

Родительским процессом для этого bash:

```
ps -f ppid
```

является gnome - процесс рабочего окружения. Родительским процессом для него является systemd - первый процесс. О systemd мы ещё поговорим.



```
Welcome to CentOS
file:///usr/share/doc/HTML/en-US/index.html
```

user	7213	3932	12
user	7304	7213	2
user	7336	7213	1
user	7353	7213	1
user	7427	7213	0
user	7484	6201	0
(END)			

CENTOS LINUX 8

Welcome to CentOS
The Community ENTerprise Operating System

CentOS Linux is an enterprise-class Linux distribution derived from Red Hat, Inc. for Red Hat Enterprise Linux. CentOS conforms fully to the Red Hat Enterprise Linux compatibility policy and aims to be functionally compatible. (CentOS mainly changes the vendor branding and artwork.)

Четвёртый столбик – С – использование процессора данным процессом. Много где у нас нули, но давайте запустим какое-нибудь тяжёлое приложение, допустим, firefox, найдём этот процесс:

```
ps -ef | grep firefox | less -S
```

и увидим, что для него это значение отличается от нуля.

```

File Edit View Search Terminal Help
user@centos8:~
gdm      2399  2239  0 18:32  tty1      00:00:00 /usr/libexec/gsd-sharing
gdm      2406  2239  0 18:32  tty1      00:00:00 /usr/libexec/gsd-smartcard
gdm      2410  2239  0 18:32  tty1      00:00:00 /usr/libexec/gsd-sound
gdm      2414  2239  0 18:32  tty1      00:00:00 /usr/libexec/gsd-wacom
colord   2437      1  0 18:32 ?        00:00:00 /usr/libexec/colord
root    2531  2171  0 18:35 ?        00:00:00 gdm-session-worker [pam/g>
user    2541      1  0 18:35 ?        00:00:00 /usr/lib/systemd/systemd >
user    2547  2541  0 18:35 ?        00:00:00 (sd-pam)
user    2557  2541  0 18:35 ?        00:00:00 /usr/bin/pulseaudio --dae>
user    2562      1  0 18:35 ?        00:00:00 /usr/bin/gnome-keyring-da>
user    2570  2541  0 18:35 ?        00:00:00 /usr/bin/dbus-daemon --se>
user    2578  2531  0 18:35  tty2     00:00:00 /usr/libexec/gdm-wayland->
user    2581  2578  0 18:35  tty2     00:00:00 /usr/libexec/gnome-sessio>
:

```

Дальше - TTY – от слова телетайп. На хабре есть неплохая [статья](#), объясняющая разницу между телетайпом, консолью, терминалом и т.п. А ps в этом столбике говорит, с каким терминалом ассоциируется данный процесс. Обычно, процесс, запущенный системой и не требующий графики, вывода информации на экран, не связан ни с каким терминалом. Процессы, требующие графики, завязаны на каком-нибудь виртуальном терминале – о них мы говорили ранее. Можно заметить, что тут указано tty1 и tty2 – если нажать правый ctrl + f1 или ctrl+f2, можно увидеть, что именно здесь у нас запущен графический интерфейс. При переходе на ctrl+f3 и далее открывается виртуальный терминал. А для эмуляторов терминала здесь могут быть значения pts/0, pts/1, pts/2 и т.п.

```

user      8443  3932  8 19:14 pts/0      00:00:06 /usr/lib64/firefox/firefox
user      8522  8443  0 19:14 pts/0      00:00:00 /usr/lib64/firefox/firefox
user      8556  8443  0 19:14 pts/0      00:00:00 /usr/lib64/firefox/firefox
user      8642  8443 11 19:14 pts/0      00:00:08 /usr/lib64/firefox/firefox
user      8690  8443  0 19:15 pts/0      00:00:00 /usr/lib64/firefox/firefox
user      8928  8779  0 19:16 pts/1      00:00:00 grep --color=auto firefox

```

Ещё одно поле – TIME – это сколько времени процессор потратил на работу с данным процессом. Вы можете заметить, что здесь сплошные нули – потому что большинство этих процессов не требуют и секунды процессорного времени. Но если немного поработать с тем же браузером, то это время будет расти:

```
ps -ef | grep firefox
```

Кстати, чтобы мне не приходилось постоянно запускать эту команду, я могу использовать команду watch:

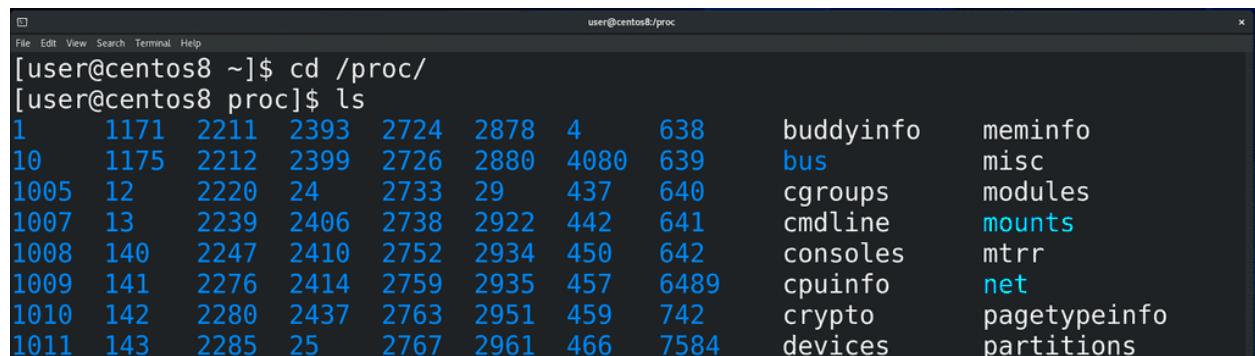
```
watch "ps -ef | grep firefox"
```

Эта команда будет каждые 2 секунды запускать указанную команду. И так мы видим, что параметр TIME для нашего браузера постоянно увеличивается.

rpc	972	1	0	18:32	?	00:00:00	/usr/bin/rpcbind -w -t
root	976	2	0	18:32	?	00:00:00	[rpciod]
root	977	2	0	18:32	?	00:00:00	[kworker/u3:0]
root	978	2	0	18:32	?	00:00:00	[xprtiod]
root	981	1	0	18:32	?	00:00:00	/sbin/auditd
root	983	981	0	18:32	?	00:00:00	/usr/sbin/sedispatch
rtkit	1005	1	0	18:32	?	00:00:00	/usr/libexec/rtkit-daemon
dbus	1007	1	0	18:32	?	00:00:01	/usr/bin/dbus-daemon --system ->
root	1008	1	0	18:32	?	00:00:00	/usr/sbin/alsactl -s -n 19 -c ->
libstor+	1009	1	0	18:32	?	00:00:00	/usr/bin/lsmd -d
root	1010	1	0	18:32	?	00:00:00	/usr/lib/systemd/systemd-machineid
root	1011	1	0	18:32	?	00:00:00	/sbin/rngd -f
	1012	1	0	18:32	?	00:00:00	/usr/lib/systemd/systemd-random

Ну и последнее – CMD – это команда, которая запустила процесс. Некоторые значения в квадратных скобках – для таких процессов ps не смог найти аргументов – обычно это процессы самого ядра.

Ладно, с выводом ps разобрались. Теперь мы знаем, где найти информацию о процессах. Но, помните, я говорил, что в Unix подобных системах придерживаются идеи «Всё есть файл»? И даже процессы у нас представлены в виде файлов. Но хранить информацию о процессах на жёстком диске нецелесообразно – какие-то процессы существуют доли секунд, какие-то появляются и удаляются сотнями – жёсткий диск не подходит для такого. А вот в оперативной памяти информацию о процессах можно спокойно хранить и представлять в виде файлов. Но раз уж речь идёт о файлах, то нам нужна файловая система. И вот ядро действительно создаёт так называемую виртуальную файловую систему, которая существует только в оперативной памяти.

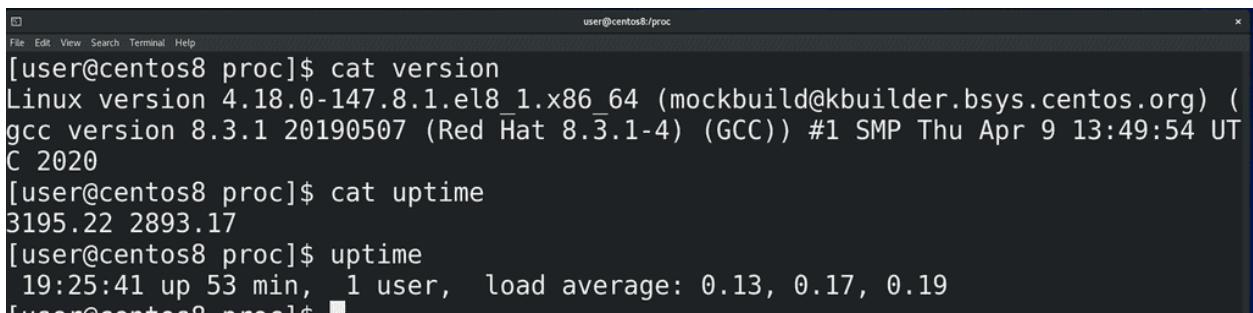


```
user@centos8 ~]$ cd /proc/
[user@centos8 proc]$ ls
1      1171  2211  2393  2724  2878  4      638      buddyinfo    meminfo
10     1175  2212  2399  2726  2880  4080  639      bus        misc
1005   12    2220  24    2733  29    437    640      cgroups   modules
1007   13    2239  2406  2738  2922  442    641      cmdline   mounts
1008   140   2247  2410  2752  2934  450    642      consoles  mtrr
1009   141   2276  2414  2759  2935  457    6489    cpuinfo   net
1010   142   2280  2437  2763  2951  459    742      crypto    pagetypeinfo
1011   143   2285  25    2767  2961  466    7584    devices   partitions
```

Вообще, этих виртуальных файловых систем несколько, они используются для разных задач, мы о них поговорим в другой раз. Сейчас нас интересует файловая система procfs. Она примонтирована в директорию /proc:

```
cd /proc
ls
```

Если посмотреть содержимое этой директории, мы увидим кучу директорий и файлов. Директории вам ничего не напоминают? Именно, это номера процессов, т.е. pid-ы. Ядро операционной системы генерирует эту информацию налету, стоит нам посмотреть – мы увидим актуальную информацию.



```
[user@centos8 proc]$ cat version
Linux version 4.18.0-147.8.1.el8_1.x86_64 (mockbuild@kbuilder.bsys.centos.org) (gcc
version 8.3.1 20190507 (Red Hat 8.3.1-4) (GCC)) #1 SMP Thu Apr 9 13:49:54 UT
C 2020
[user@centos8 proc]$ cat uptime
3195.22 2893.17
[user@centos8 proc]$ uptime
19:25:41 up 53 min, 1 user, load average: 0.13, 0.17, 0.19
```

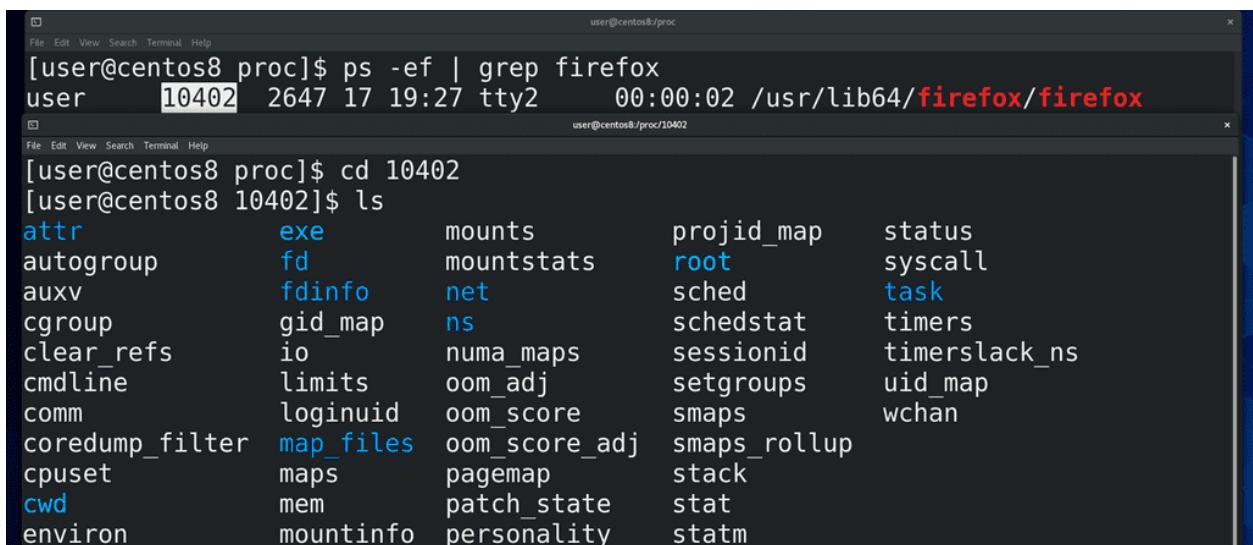
В этой директории кроме директорий процессов есть много других файлов – допустим, version:

```
cat version
```

показывает нам информацию о версии ядра или uptime:

```
cat uptime
```

информацию о том, сколько секунд включена система. Ну в секундах непонятно, поэтому легче использовать утилиту uptime. Кстати, постарайтесь самостоятельно найти, что означает второе значение в файле /proc/uptime и напишите в комментариях так, чтобы было понятно всем.



```
[user@centos8 proc]$ ps -ef | grep firefox
user    10402  2647 17 19:27 tty2    00:00:02 /usr/lib64/firefox/firefox
user@centos8$ cd 10402
[user@centos8 10402]$ ls
attr      exe      mounts      projid_map      status
autogroup  fd      mountstats  root          syscall
auxv      fdinfo   net          sched          task
cgroup     gid_map  ns          schedstat      timers
clear_refs io      numa_maps   sessionid      timerslack_ns
cmdline    limits   oom_adj     setgroups      uid_map
comm       loginuid  oom_score   smaps         wchan
coredump_filter map_files  oom_score_adj smaps_rollup
cpuset     maps     pagemap    stack
cwd        mem      patch_state stat
environ    mountinfo personality statm
```

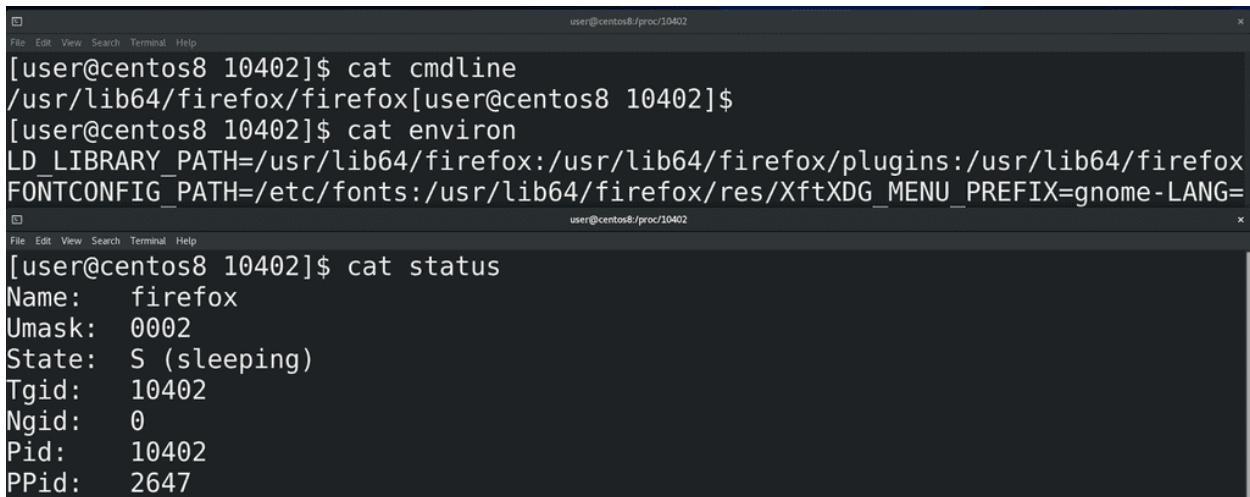
Ну и давайте посмотрим, что же такого в директориях процессов. Найдём pid процесса, допустим того-же firefox:

```
ps -ef | grep firefox
```

и зайдём в эту директорию:

```
cd pid
ls
```

Тут у нас тоже куча файлов, которые относятся к нашему процессу. Эти файлы нужны не столько для людей, сколько для программ.



The screenshot shows two terminal windows side-by-side. The left window has a title bar 'File Edit View Search Terminal Help' and a command prompt '[user@centos8 10402]\$. cat cmdline'. Below it, the output shows the command '/usr/lib64/firefox/firefox' followed by another command '[user@centos8 10402]\$'. The right window also has a title bar 'File Edit View Search Terminal Help' and a command prompt 'user@centos8:/proc/10402\$. cat environ'. Below it, the output shows environment variables like 'LD_LIBRARY_PATH=/usr/lib64/firefox:/usr/lib64/firefox/plugins:/usr/lib64/firefox/FONTCONFIG_PATH=/etc/fonts:/usr/lib64/firefox/res/XftXDG_MENU_PREFIX=gnome-LANG='.

Какие-то из этих файлов и мы можем прочесть. Например, cmdline:

```
cat cmdline
```

Тут отображена команда, которая запустила процесс. Или environ:

```
cat environ
```

те переменные, которые передались процессу при запуске. Или status:

```
cat status
```

Какие-то из этих строчек понятны, а какие-то без гугла не разберёшь. Знать всё это не нужно, но, со временем, углубляясь в теорию или сталкиваясь с какими-то проблемами, вы начнёте разбираться в этих файлах.

2.14.2 Практика

Вопросы

1. Запустите ps -ef | head и объясните значение каждого столбика.

Задания

1. Выведите список процессов вашего пользователя.
2. Выведите список процессов, запущенных в эмуляторах терминала.
3. Запустите gedit, найдите его pid, найдите pid его родительского процесса и так по цепочке все родительские pid-ы.

2.15 15. Процессы №2: Информация о процессах №2

2.15.1 15. Процессы №2: Информация о процессах №2

Обычно процессы появляются, делают свою работу и завершаются не требуя никакого внимания. Но бывают случаи, когда администратору все же следует вмешаться. Давайте разберём пару таких ситуаций. Самая популярная – зависание процесса или системы. Я думаю все сталкивались с такими ситуациями, когда какой-то процесс начинает использовать слишком много оперативной памяти или

ресурсов процессора. Это приводит к тому, что каким-то другим процессам, той же оболочке, перестаёт хватать ресурсов и всё начинает зависать. Чтобы решить проблему, сначала нужно понять, какой процесс во всём виноват. Для этого нужно увидеть список процессов, которые используют большую часть ресурсов. Это нам покажет утилита top:

```
top
```

```
user@centos8:~$ top - 18:59:13 up 1 min, 1 user, load average: 1.34, 0.43, 0.15
Tasks: 250 total, 3 running, 246 sleeping, 0 stopped, 1 zombie
%Cpu(s): 0.4 us, 0.4 sy, 0.0 ni, 98.6 id, 0.4 wa, 0.4 hi, 0.0 si, 0.0 st
MiB Mem : 1829.6 total, 170.5 free, 1144.1 used, 515.0 buff/cache
MiB Swap: 2048.0 total, 2032.0 free, 16.0 used. 512.1 avail Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
3336 root 29 9 187536 8264 3700 R 2.0 0.4 0:00.06 (coredump)
2588 user 20 0 2875352 301744 111764 S 1.3 16.1 0:03.80 gnome-she+
  1 root 20 0 187532 13344 8804 S 0.7 0.7 0:00.95 systemd
  973 root 16 -4 143088 2712 2100 S 0.3 0.1 0:00.01 auditd
1024 polkitd 20 0 1635812 25924 18528 S 0.3 1.4 0:02.24 polkitd
1032 dbus 20 0 72328 7060 4148 S 0.3 0.4 0:00.46 dbus-daem+
3206 user 20 0 736524 46976 36552 S 0.3 2.5 0:00.23 gnome-ter+
3304 user 20 0 273768 4888 4036 R 0.3 0.3 0:00.09 top
```

Наверху отображается общая информация по системе. Давайте пройдёмся по каждому из значений. Первая строчка – вывод утилиты uptime:

1. Текущее время в системе
2. Сколько времени система включена
3. Хоть тут и написано user, подразумевается количество залогиненных сессий. Это могут быть как сессии одного пользователя, так и других.

```
user@centos8:~$ top - 19:04:38 up 6 min, 2 users, load average: 0.11, 0.21, 0.13
Tasks: 242 total, 1 running, 241 sleeping, 0 stopped, 0 zombie
user@centos8:~$ w
 19:04:24 up 6 min, 2 users, load average: 0.14, 0.22, 0.13
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
user tty2 tty2 18:58 6:13 18.26s 0.77s /usr/bin/gno
user tty3 - 19:03 32.00s 0.24s 0.23s top
user@centos8 ~]$
```

Эмулятор терминала даёт нам оболочку без входа, т.е. в ней мы не логинимся, а вот виртуальный терминал – оболочку со входом. Давайте, для примера, я залогинюсь тем же пользователем в виртуальной терминале (ctrl+alt+f3). Теперь здесь отображается 2 пользователя. Кстати, информацию о залогиненных пользователях можно увидеть с помощью утилиты w:

```
w
```

4. Среднее значение загрузки системы за минуту, 5 минут и 15 минут. Есть свои нюансы подсчёта этого значения, рекомендую почитать [статью](#) на википедии.

```

top - 18:59:13 up 1 min, 1 user, load average: 1.34, 0.43, 0.15
Tasks: 250 total, 3 running, 246 sleeping, 0 stopped, 1 zombie
%Cpu(s): 0.4 us, 0.4 sy, 0.0 ni, 98.6 id, 0.4 wa, 0.4 hi, 0.0 si, 0.0 st
MiB Mem : 1829.6 total, 170.5 free, 1144.1 used, 515.0 buff/cache
MiB Swap: 2048.0 total, 2032.0 free, 16.0 used. 512.1 avail Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
3336 root 29 9 187536 8264 3700 R 2.0 0.4 0:00.06 (coredump)
2588 user 20 0 2875352 301744 111764 S 1.3 16.1 0:03.80 gnome-she+
1 root 20 0 187532 13344 8804 S 0.7 0.7 0:00.95 systemd
973 root 16 -4 143088 2712 2100 S 0.3 0.1 0:00.01 auditd
1024 polkitd 20 0 1635812 25924 18528 S 0.3 1.4 0:02.24 polkitd
1032 dbus 20 0 72328 7060 4148 S 0.3 0.4 0:00.46 dbus-daem+
3206 user 20 0 736524 46976 36552 S 0.3 2.5 0:00.23 gnome-ter+
3304 user 20 0 273768 4888 4036 R 0.3 0.3 0:00.09 top

```

На второй строчке информация о количестве процессов:

1. Сколько всего запущенных процессов
2. Сколько процессов выполняется в данный момент
3. Сколько процессов просто спят, например, в ожидании каких-то данных
4. Сколько процессов остановлены. Вообще, процессы можно останавливать. В этот момент они всё ещё находятся в оперативке, но перестают посылать код на процессор – просто ничего не делают. Допустим, когда несколько программ пишут данные на диск, то у диска перестаёт хватать скорости. Вы можете временно остановить какие-то из программ, чтобы другие ускорили свою работу. Но нужно понимать, что это очень специфично, и разные программы могут по разному на такое реагировать. Допустим, если программа работает с сетью и её остановить, то скорее всего сетевое соединение прервётся. Как останавливать программы мы разберём чуть позже.

```

[user@centos8 ~]$ cat /proc/sys/kernel/pid_max
32768
[user@centos8 ~]$ ulimit -u
7171
[user@centos8 ~]$ 

```

5. Сколько зомби процессов. Когда один процесс запускает другой, то он является для него родительским процессом, а запущенный – дочерним. Так вот, когда дочерний процесс выполнит свою задачу, он умирает, держа в руках табличку со статусом завершения, мол, всё ок или не ок. То есть передаёт родительскому процессу статус выхода. Такой процесс называется зомби. При этом родительский процесс должен прочитать эту табличку, тем самым отпустить зомби процесс с миром. Зомби процессы не используют никаких ресурсов, просто пока родитель не прочтёт статус, они всё ещё числятся с таблице процессов. Но если родительская программа плохо написана, постоянно создаёт дочерние процессы, с которыми не прощается, то со временем таблица будет заполняться зомби процессами. А количество процессов в таблице ограничено. Для 32-битных систем максимально может быть примерно 32 тысячи записей, а для 64 битных – 4 миллиона. Но это значение можно поставить и поменьше. Увидеть текущий максимум можно взглянув на файл:

```
cat /proc/sys/kernel/pid_max
```

Обычно, для пользователей ставят своё ограничение на количество процессов – это можно увидеть с помощью команды:

```
ulimit -u
```

Ограничение у пользователей ставится, чтобы какой-то один пользователь не забил всю таблицу, либо не запустил слишком много процессов. Ведь когда таблица забита, например, теми же зомби процессами – то ничего сделать с системой не получится, потому что не получится создать новый процесс. Остаётся разве что по жёсткому перезагрузить систему.

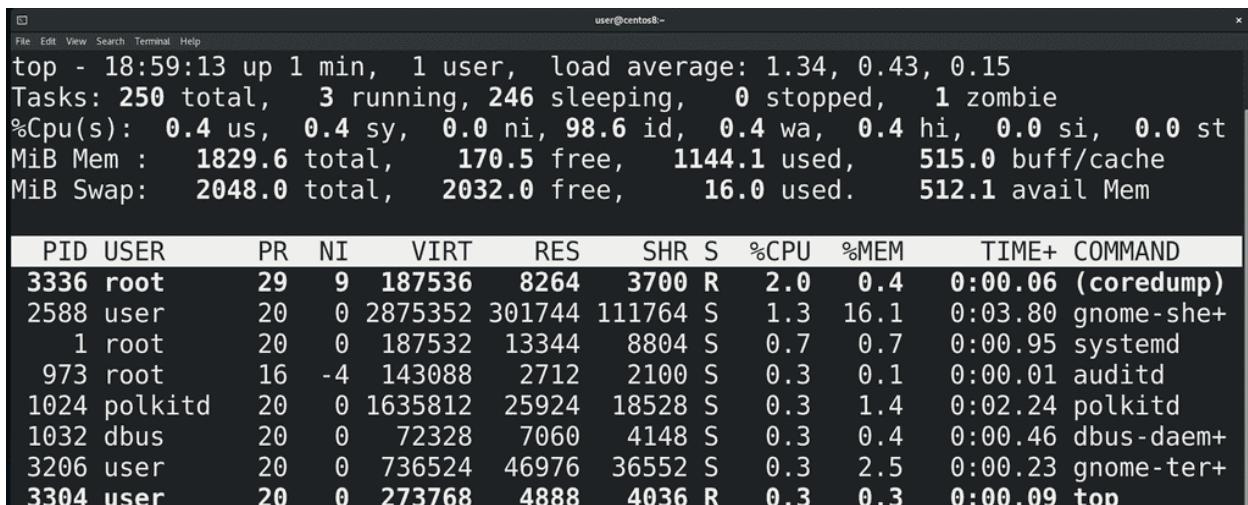
```
GNU nano 2.9.8          /etc/security/limits.conf
#*      soft  core      0
#*      hard  rss      10000
#@student  hard  nproc     20
#@faculty   soft  nproc     20
#@faculty   hard  nproc     50
#ftp      hard  nproc     0
#@student  -    maxlogins  4

GNU nano 2.9.8          /home/user/.bash_profile
14 # Limits
15 ulimit -u 1024
```

Чтобы ограничить количество процессов у пользователя, нужно в домашней директории этого пользователя в файле `~/.bash_profile` выставить значение, например:

```
ulimit -u 1024
```

Но обычно этот файл может редактировать пользователь, и, если у вас много пользователей или вы не доверяете ему, то, как правило, ограничения устанавливаются в файле `/etc/security/limits.conf`, либо внутри директории `/etc/security/limits.d/` создаётся файл, заканчивающийся на `.conf`. Пример есть в файле `/etc/security/limits.conf`. Нас интересует параметр `nproc – number of processes` – количество процессов. Это всё превентивные меры, которые защищают работу самой системы. Но если всё же у какого-то пользователя забилась таблица процессов и он не может запускать новые, то нужно избавляться от родительского процесса. Мы это рассмотрим чуть позже.



The screenshot shows the output of the 'top' command on a Linux system. The title bar indicates 'user@centos8 ~'. The output provides system statistics and a process list.

```

top - 18:59:13 up 1 min, 1 user, load average: 1.34, 0.43, 0.15
Tasks: 250 total, 3 running, 246 sleeping, 0 stopped, 1 zombie
%Cpu(s): 0.4 us, 0.4 sy, 0.0 ni, 98.6 id, 0.4 wa, 0.4 hi, 0.0 si, 0.0 st
MiB Mem : 1829.6 total, 170.5 free, 1144.1 used, 515.0 buff/cache
MiB Swap: 2048.0 total, 2032.0 free, 16.0 used. 512.1 avail Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
3336 root 29 9 187536 8264 3700 R 2.0 0.4 0:00.06 (coredump)
2588 user 20 0 2875352 301744 111764 S 1.3 16.1 0:03.80 gnome-she+
1 root 20 0 187532 13344 8804 S 0.7 0.7 0:00.95 systemd
973 root 16 -4 143088 2712 2100 S 0.3 0.1 0:00.01 auditd
1024 polkitd 20 0 1635812 25924 18528 S 0.3 1.4 0:02.24 polkitd
1032 dbus 20 0 72328 7060 4148 S 0.3 0.4 0:00.46 dbus-daem+
3206 user 20 0 736524 46976 36552 S 0.3 2.5 0:00.23 gnome-ter+
3304 user 20 0 273768 4888 4036 R 0.3 0.3 0:00.09 top

```

На третьей строчке немного информации про использование процессора. Она отображается в процентах. По сути это процент времени, которое процессор потратил на те или иные задачи в промежуток времени обновления информации в top, по-умолчанию это 3 секунды. Немного запутано, но на первом примере, надеюсь, станет понятнее.

1. us – это user cpu time, т.е. время, потраченное на user space. Под user space подразумевается пользовательское пространство. И пусть вас не путает слово user, речь идет о всех программах, кроме ядра. Для ядра есть пространство ядра – kernel space. Допустим, если взять веб сервер, то сама программа веб сервера, браузер, всякие утилиты – все это user space. А ядро при этом отвечает за работу с сетью – это kernel space. Так вот, первый параметр – время, выделенное процессором на работу user space программ. Допустим, из 3 секунд полторы секунды процессор потратил на выполнение кода веб сервера – то тут значение будет 50%. Но нужно понимать, что процессор не уделяет все эти полторы секунды чисто одному процессу, какие-то доли секунд на один процесс, какие-то доли секунд на другой, и вот суммарно на user-space программы тратится сколько-то процентов времени.
2. sy – system cpu time – собственно, время, потраченное на kernel space.
3. ni – nice cpu time – время, потраченное на процессы с низким приоритетом. Процессов в системе много, каждый из них выполняет свою задачу – какая-то из них важная и нужна срочно, какая-то нет. Например, у меня рендерится видео и тут мне по работе присылают кучу документов в сжатом архиве. Мне нужно всё это разархивировать, поработать и заархивировать. Но оба этих процесса – рендеринг и архивация/сжатие – требуют много процессорного времени, поэтому мне придется либо остановить рендеринг, либо долго ждать, пока разархивируется файл. Но, используя приоритеты, я могу сказать, что пусть рендеринг займёт чуть больше времени, мне срочно нужно выполнить новый процесс. Для выставления приоритетов используется утилита nice и числа от -20 до 19. nice в переводе – вежливый, поэтому чем больше число, тем «вежливее» программа. Т.е. программа с приоритетом 19 будет уступать процессорное время другим процессам. Таким образом, -20 это наивысший приоритет, а 19 это низший приоритет.

```
[user@centos8 ~]$ nice
0
[user@centos8 ~]$ nice -n -1 firefox
nice: cannot set niceness: Permission denied
[user@centos8 ~]$ nice -n 5 firefox
[user@centos8 ~]$ ps -l | head -1
F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY          TIME CMD
[user@centos8 ~]$ ps -el | grep firefox
4 S 1000 11724 4145 17 85 5 - 778157 x64_sy pts/1  00:00:02 firefox
[user@centos8 ~]$ renice -n 10 11724
11724 (process ID) old priority 5, new priority 10
[user@centos8 ~]$
```

Приоритет по умолчанию – 0. Это можно увидеть просто выполнив команду nice:

nice

Причём, более высокий приоритет, например, -1, -2, обычный пользователь не может задать, а вот приоритет пониже можно. Например, чтобы запустить firefox с приоритетом 5, я пишу:

nice -n 5 firefox

Чтобы посмотреть текущий приоритет, я использую ps -el. Допустим:

ps -el | grep firefox

Обратите внимание, что 5 указано в столбике NI – это niceness – вежливость. А слева от него написано PRI – приоритет – и он отличается от того, что мы указывали в утилите nice. Это потому, что в nice мы указываем желаемый приоритет программы, но во время работы ядро распределяет приоритеты по своей шкале. Поэтому параметр priority показывает текущий приоритет в ядре, а nice – заданный приоритет. Если же программа уже запущена, её вежливость можно изменить с помощью renice:

renice

Причём, всегда можно повысить вежливость, тем самым уменьшив приоритет, если, конечно, процесс запущен от вашего пользователя. Но для повышения приоритета нужны права суперпользователя. Пример утилиты renice:

renice -n 10 pid

Обратите внимание, что здесь я использую идентификатор процесса.

4. id – от слова idle – время, проведённое в ожидании. Как вы видите, большую часть времени процессор простояивает в ожидании программ.
5. wa – input output wait – время, потраченное процессором на ожидание чтения или записи на диск. Например, какая-то программа работает и в какой-то момент ей нужно записать данные на диск. Это занимает время, и, пока это происходит, процессор простояивает. На самом деле не совсем простояивает, он может и другими процессами заняться, но, в любом случае, для данного процесса он простояивает.
6. hi – hardware interrupts – аппаратные прерывания. Когда какое-то оборудование, допустим, жёсткий диск или сетевая карта, пытаются сообщить процессору о каком-то событии, оно посыпает

процессору сигнал, называемый аппаратным прерыванием. Собственно этот параметр в top – время процессора, потраченное для аппаратных прерываний.

7. si – software interrupts – это уже программные прерывания.
8. st – steal time – это параметр, относящийся к виртуальным машинам – и говорит он о том, какое время реальный процессор был недоступен виртуальной машине, потому что был занят гипервизором или другими виртуальными машинами.

```

top - 18:59:13 up 1 min, 1 user, load average: 1.34, 0.43, 0.15
Tasks: 250 total, 3 running, 246 sleeping, 0 stopped, 1 zombie
%Cpu(s): 0.4 us, 0.4 sy, 0.0 ni, 98.6 id, 0.4 wa, 0.4 hi, 0.0 si, 0.0 st
MiB Mem : 1829.6 total, 170.5 free, 1144.1 used, 515.0 buff/cache
MiB Swap: 2048.0 total, 2032.0 free, 16.0 used. 512.1 avail Mem

PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM TIME+ COMMAND
3336 root      29   9  187536  8264  3700 R  2.0   0.4  0:00.06 (coredump)
2588 user      20   0 2875352 301744 111764 S  1.3  16.1  0:03.80 gnome-she+
  1 root      20   0  187532 13344  8804 S  0.7   0.7  0:00.95 systemd
  973 root     16  -4  143088  2712  2100 S  0.3   0.1  0:00.01 auditd
1024 polkitd   20   0 1635812 25924 18528 S  0.3   1.4  0:02.24 polkitd
1032 dbus      20   0  72328  7060  4148 S  0.3   0.4  0:00.46 dbus-daem+
3206 user      20   0  736524 46976 36552 S  0.3   2.5  0:00.23 gnome-ter+
3304 user      20   0 273768  4888  4036 R  0.3   0.3  0:00.09 top

```

На четвёртой строчке у нас немного про оперативную память. Как видите, здесь написано MiB Mem – это мебибайты. Если вы раньше считали, что в одном килобайте 1024 байт, а в одном мегабайте – 1024 килобайт – это упрощение информатики. [На самом деле](#), в одном килобайте 1000 байт, а в одном мегабайте – 1000 килобайт. А вот 1024 относится к кибибайтам и мебибайтам. Но эти значения не сильно отличаются и можно догадаться, что 1800 с лишним мебибайт – это 2 гигабайта оперативки.

1. total – сколько всего оперативки
2. free – сколько оперативки свободно. Вас может смутить, что тут очень мало, но, не всё так просто и нужно смотреть третье значение
3. used – сколько оперативки используется. Казалось бы, математика простая – от общего числа отними используемое – и получится свободное – но тут не получается. Есть даже специальный [сайт](#), который говорит – не паникуйте, всё норм. Нужно понимать, что оперативка создана, чтобы её использовали, поэтому Linux старается выжать из неё максимум. Если она свободна, линукс использует её для кэша диска – то есть хранит часто запрашиваемые файлы с жёсткого диска в оперативке, тем самым всё работает быстрее. Если же какой-то программе понадобится оперативка, то она запросто отожмёт это места у кэша.
4. buff/cache – память, используемое для кэша.

The screenshot shows three terminal windows. The top window displays the command `ulimit -v` with the output `unlimited`. The middle window shows the contents of `/home/user/.bash_profile`, which contains the following lines:

```
14 # Limits
15 ulimit -u 1024
16 ulimit -v 1000000
```

The bottom window shows the contents of `/etc/security/limits.conf`, which contains the following lines:

```
40 #           - as - address space limit (KB)
41 #           - maxlogins - max number of logins for this user
42 #           - maxsyslogins - max number of logins on the system
```

Таким образом, если хотите знать, сколько оперативки использует ваш Linux – смотрите значение used. Вся остальная оперативка свободна для других программ. Кстати, как и с количеством процессов, вы можете ограничить количество виртуальной памяти, выделяемое каждому процессу. Для этого можете использовать `ulimit` с опцией `-v`:

`ulimit -v`

указав количество памяти в килобайтах, и записав в `~/.bash_profile`, чтобы всегда работало, либо в `/etc/security/limits.conf` прописать свое значение с параметром `as` – address space limit. Но виртуальная память состоит не только из оперативки, мы о ней ещё поговорим.

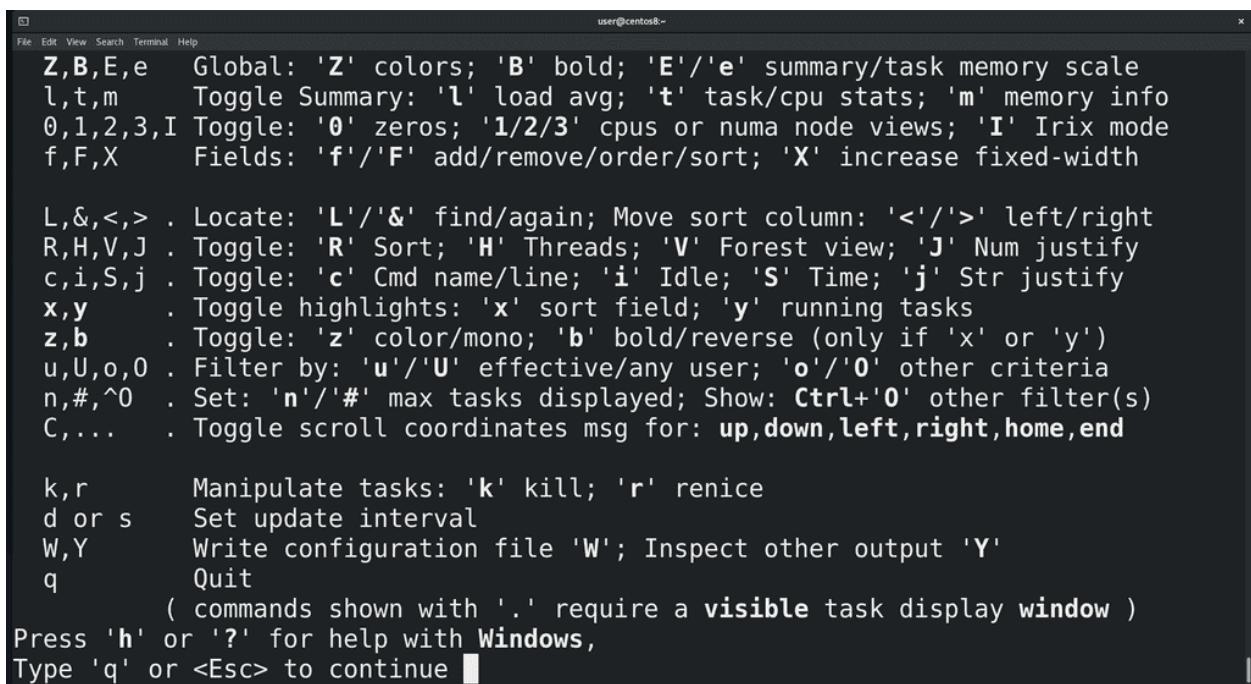
Ну и последняя строчка – swap, также называемый подкачкой. Это такой локальный филиал оперативки на жёстком диске. Когда оперативке перестаёт хватать места или ей требуется освободить пространство для более активных процессов, она может использовать специальное место на жёстком диске или ssd. Это может быть как специальный файл, так и целый раздел. Часто бывает полезно на некоторых серверах. Раньше это было полезно и на домашних компьютерах, когда не хватало оперативной памяти, но сейчас оперативки обычно много и всё это не так актуально. Подкачка ещё используется для спящего режима – но его размер должен быть больше оперативки, а это не всегдаrationально, когда речь идёт про ssd и большое количество оперативки.

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
2730	root	20	0	413976	26520	5996	S	0.3	1.4	0:11.44	sssd_kcm
3206	user	20	0	740720	48104	32500	R	0.3	2.6	0:13.55	gnome-terminal
4052	user	20	0	273788	3748	2872	S	0.3	0.2	0:23.06	top
16346	user	20	0	273756	4944	4104	R	0.3	0.3	0:02.20	top
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_par_gp
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0+
8	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
9	root	20	0	0	0	0	S	0.0	0.0	0:00.80	ksoftirqd/0
10	root	20	0	0	0	0	I	0.0	0.0	0:00.17	rcu_sched
11	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
12	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	watchdog/0
13	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cputop/0

Ниже у нас таблица. Многое мы с вами уже разбирали:

- PID - идентификатор процесса;
- USER - пользователь, запустивший процесс;
- PR - текущий приоритет в ядре. Кстати, обратите внимание, что у некоторых процессов вместо чисел написано rt – это real time – реальное время. Это означает, что такие процессы операционная система будет выполнять моментально, без всяких ожиданий выполнения других процессов. Как правило, это процессы, работающие с оборудованием. Есть даже специальные операционные системы реального времени, которые используются в промышленности, телекоммуникациях и прочих местах, где есть требования к скорости работы операционной системы с железом.
- NI - niceness - вежливость процесса;
- VIRT – виртуальная память. Мы говорили, что при запуске для процесса создаётся иллюзия, что этот процесс единственный в оперативке. Так вот, это осуществляется благодаря тому, что для каждого процесса создаётся адресное пространство. Представьте себе это как программу excel – у каждого процесса есть свой лист с таблицей, где можно заполнять данные. Есть адреса, допустим A1, B5 и т.п. – и у разных листов, т.е. процессов, адреса могут совпадать, но данные в них разные. Эти адреса ядро операционной системы связывает где-то с физическими адресами в оперативке, где-то со свапом на диске, а где-то с большими файлами на диске, которые система не загружает в оперативку. И вот все эти адреса для процесса находятся в одном листе. Лист – просто аналогия, а на деле это называется адресным пространством.
- RES – сколько реальной оперативной памяти использует процесс. Отображается в килобайтах, как и виртуальная память. Столбик MEM показывает это же значение в процентах от всей оперативной памяти.
- SHR – shared – потенциально сколько виртуальной памяти может быть разделено с другими процессами. Чтобы разные процессы не дублировали одни и те же данные, к примеру, библиотеки, процессы могут делиться частью виртуальной памяти.
- S – статус процесса. S – это спит, R – работает, I – специфично для потоков ядра, ожидающих каких-то событий. Тут ещё может быть Z – зомби и пара других статусов.
- CPU – использование процессора в процентах. По умолчанию это процент не от всего процессора, а от одного ядра. Т.е. тут значения могут быть больше 100%, допустим, 150%.
- TIME+ - сколько времени процессор работал с этим процессом. Тут минуты:секунды.сотые секунд.

На самом деле, это далеко не все столбцы, которые может показать top. В принципе можно перенастроить программу так, чтобы она показывала другие столбцы, но разбирать все это займёт много времени. У top достаточно хороший ман, а мы с вами и так достаточно ударились в теорию. Давайте уже к практике.



```

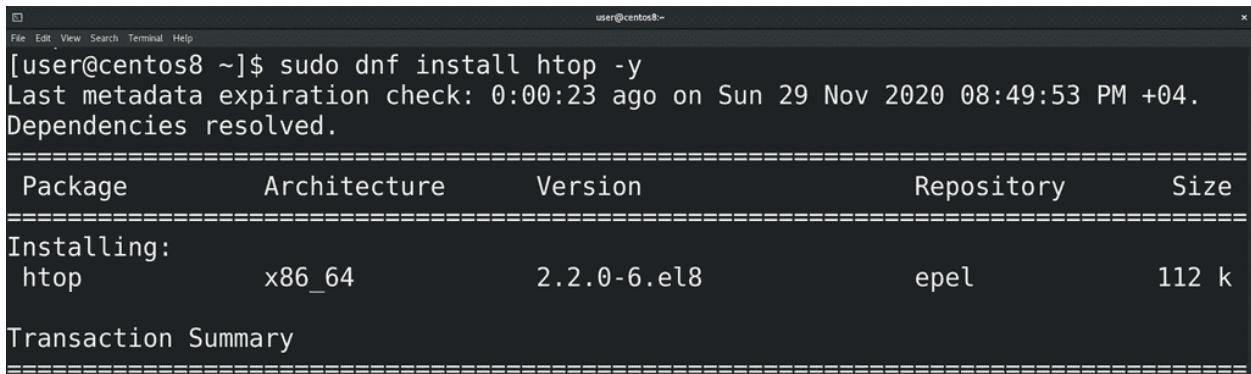
user@centos8:~-
Z,B,E,e  Global: 'Z' colors; 'B' bold; 'E'/'e' summary/task memory scale
l,t,m    Toggle Summary: 'l' load avg; 't' task/cpu stats; 'm' memory info
0,1,2,3,I Toggle: '0' zeros; '1/2/3' cpus or numa node views; 'I' Irix mode
f,F,X    Fields: 'f'/'F' add/remove/order/sort; 'X' increase fixed-width

L,&,<,> . Locate: 'L'/'&' find/again; Move sort column: '<'/'>' left/right
R,H,V,J . Toggle: 'R' Sort; 'H' Threads; 'V' Forest view; 'J' Num justify
c,i,S,j . Toggle: 'c' Cmd name/line; 'i' Idle; 'S' Time; 'j' Str justify
x,y     . Toggle highlights: 'x' sort field; 'y' running tasks
z,b     . Toggle: 'z' color/mono; 'b' bold/reverse (only if 'x' or 'y')
u,U,o,O . Filter by: 'u'/'U' effective/any user; 'o'/'O' other criteria
n,#,^0   . Set: 'n'/'#' max tasks displayed; Show: Ctrl+'0' other filter(s)
C,...   . Toggle scroll coordinates msg for: up,down,left,right,home,end

k,r     Manipulate tasks: 'k' kill; 'r' renice
d or s  Set update interval
W,Y     Write configuration file 'W'; Inspect other output 'Y'
q       Quit
( commands shown with '.' require a visible task display window )
Press 'h' or '?' for help with Windows,
Type 'q' or <Esc> to continue █

```

И так, у нас есть таблица, и больше всего здесь нас интересуют 3 столбца – процент использования процессора, процент использования оперативки и идентификатор процесса. И так, если я хочу отсортировать по использованию процессора, я нажимаю shift+p. Чтобы отсортировать по оперативке, я нажимаю shift+m. Ну и как всегда, можно просто нажать h и будет подсказка по горячим клавишам.



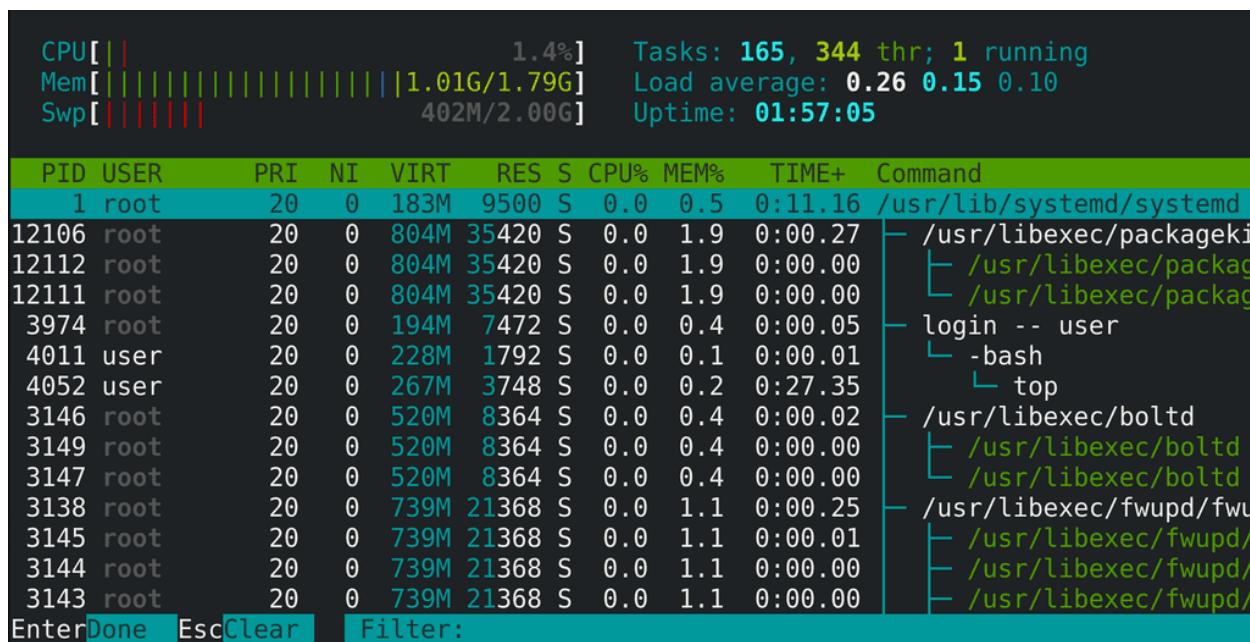
```

user@centos8 ~]$ sudo dnf install htop -y
Last metadata expiration check: 0:00:23 ago on Sun 29 Nov 2020 08:49:53 PM +04.
Dependencies resolved.
=====
Package           Architecture      Version       Repository      Size
=====
Installing:
  htop            x86_64          2.2.0-6.el8   epel           112 k
Transaction Summary
=====
```

И хотя top даёт достаточно информации и предустановлен на многих UNIX-подобных системах, у него есть более простые интуитивно понятные аналоги. Как и в случае с vi, ситуации бывают разные и может случиться, что у вас не будет альтернатив, поэтому уметь хотя бы базово разбираться в top нужно. Но в повседневной жизни вы можете использовать, к примеру, htop. Он у нас не предустановлен, поэтому давайте его установим:

```

sudo dnf install epel-release -y
sudo dnf install htop -y
htop
```



htop более интерактивен. Во первых, мы можем работать мышкой – для сортировки можно просто нажать на CPU или Mem. Снизу также есть кнопки с возможностями. Допустим, F2 – поменять внешний вид программы, поменять цвета, добавить столбцы – допустим уберем shared memory и сохраним. По F3 можем искать процессы по тексту. F4 – можем фильтровать. F5 покажет процессы в древовидной форме, чтобы увидеть родительские и дочерние процессы. Тот же nice ness – обратите внимание, при нажатии F7 ничего не происходит, а при F8 вежливость увеличивается. Почему? F10 это выход, а вот F9 – kill. Если нажать – слева появится панель – послать сигнал и список сигналов. Но kill – это не часть htop, а вполне себе отдельная команда.

Попробуйте поиграться с лимитами – ограничьте у своего пользователя количество процессов или оперативной памяти у процессов, и посмотрите, что из этого выйдет. Не забудьте предварительно снять снэпшот виртуальной машины – как это делать я показывал в части с установкой CentOS. Выставив маленькое значение, возможно, вы не сможете даже зайти в систему. Постарайтесь исправить ситуацию без использования других пользователей или восстановления со снэпшота и расскажите в комментариях, как вы это сделали.

2.15.2 Практика

Вопросы

- Запустите top и объясните значения каждого столбика и строчки.
- Что такое зомби процессы?
- Как запретить пользователю запускать слишком много процессов и зачем это нужно?

Задания

- ВНИМАНИЕ: ТОЛЬКО НА ТЕСТОВОЙ ВИРТУАЛКЕ. Запустите форк-бомбу :

```
:(){ :|:& };:
```

Затем перезагрузите виртуалку, ограничьте количество процессов у своего пользователя до 500 и заново запустите форк-бомбу¹.

¹ Результат будет в том, что в первом случае проблемы возникают у всей системы, а во втором случае только у вашего

2.16 16. Процессы №3: Работа с процессами

2.16.1 16. Процессы №3: Работа с процессами

```
[user@centos8 ~]$ gedit
Terminated
[user@centos8 ~]$ 
[user@centos8 ~]$ ps -ef | grep gedit
user      3306  3219  1 10:25 pts/0    00:00:00 gedit
user      3362  3312  0 10:26 pts/1    00:00:00 grep --color=auto
dit
[user@centos8 ~]$ kill 3306
```

kill позволяет нам посылать сигналы:

```
kill
```

Судя по названию – в основном, чтобы убивать процессы. Для примера, откроем эмулятор терминала и выполним команду gedit – откроется блокнот. Потом найдём идентификатор процесса блокнота с помощью:

```
ps -ef | grep gedit
```

и используем команду kill с нужным pid:

```
kill pid
```

И вот – блокнот закрылся.

```
[user@centos8 ~]$ kill -l
 1) SIGHUP      2) SIGINT      3) SIGQUIT      4) SIGILL      5) SIGTRAP
 6) SIGABRT     7) SIGBUS      8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

пользователя. Но чтобы заметить это изменение, нужно после запуска залогиниться от другого пользователя (а как это сделать разбирается в последующих уроках). Поэтому, желательно, вернуться к этому эксперименту после 20 темы.

Давайте запустим

```
kill -1
```

и посмотрим список сигналов. До этого мы написали просто kill и номер процесса, из-за чего процессу послался сигнал по умолчанию – 15) SIGTERM. Это мягкий сигнал, который даёт процессу время закончить свои дела, прежде чем умереть. За это время процесс успевает попрощаться с дочерними и родительскими процессами. После смерти, дочерние процессы становятся процессами-сиротами, а их родителем становится процесс с номером 1.

The screenshot shows two terminal windows. The top window displays the command `ps -f` output:

UID	PID	PPID	C	STIME	TTY	TIME	CMD
user	7297	6985	0	11:14	pts/1	00:00:00	bash
user	8129	7297	0	11:23	pts/1	00:00:00	ps -f

The bottom window shows the command `ps -ef | grep gedit` output:

user	PID	PPID	C	STIME	TTY	TIME	CMD
user	8149	7297	1	11:23	pts/1	00:00:00	gedit
user	8222	8167	0	11:24	pts/0	00:00:00	grep --color=auto

Для примера, откроем окно эмулятора терминала, там у нас появится новая bash сессия:

```
ps -f
```

В этом окне я запущу блокнот с амперсандом в конце:

```
gedit &
```

Амперсанд нужен, чтобы выполнить команду в фоне, а не быть зависимым от эмулятора терминала. При запуске мы увидели id процесса блокнота. Можно посмотреть:

```
ps -ef | grep gedit
```

где видно, что у процесса gedit родительским процессом является bash сессия в этом эмуляторе терминала.

The screenshot shows a terminal window with the following commands and their outputs:

- `ps -ef | grep gedit` output:

user	PID	PPID	C	STIME	TTY	TIME	CMD
user	8149	7297	1	11:23	pts/1	00:00:00	gedit
user	8222	8167	0	11:24	pts/0	00:00:00	grep --color=auto gedit
- `kill 7297`
- `ps -ef | grep gedit` output after kill:

user	PID	PPID	C	STIME	TTY	TIME	CMD
user	8149	2508	0	11:23	?	00:00:00	gedit
user	8731	8167	0	11:29	pts/0	00:00:00	grep --color=auto gedit
- `ps 2508` output:

PID	TTY	STAT	TIME	COMMAND
2508	?	Ss	0:00	/usr/lib/systemd/systemd --user
- `ps 1` output:

PID	TTY	STAT	TIME	COMMAND
1	?	Ss	0:01	/usr/lib/systemd/systemd --switched-root --system

Теперь давайте избавимся от родительского процесса:

```
kill ppid
```

Блокнот все ещё запущен. Найдём его:

```
ps -ef | grep gedit
```

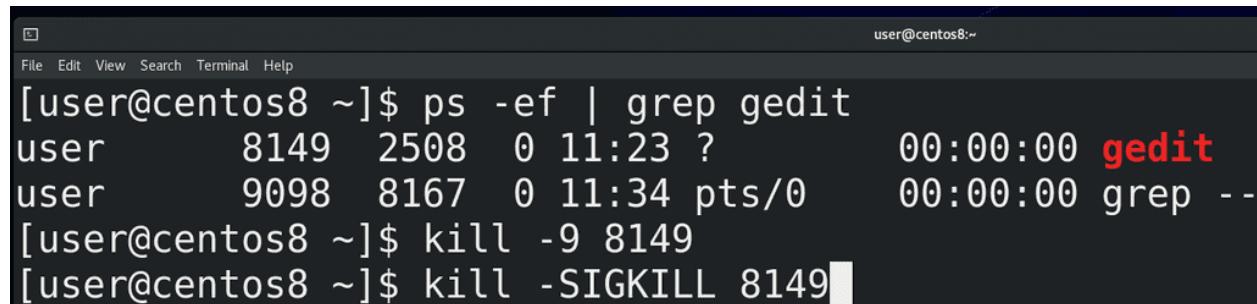
и у блокнота изменился родительский процесс. Но я говорил, что процессы сироты забирает себе процесс с номером 1, а тут другой pid. Если посмотреть этот процесс:

```
ps ppid
```

и первой процесс:

```
ps 1
```

то команды будут похожи – systemd. В большинстве unix-подобных систем сирот на себя берёт первый процесс, но то что мы видим сейчас – связано с изменениями, которые происходят в последние лет 10. Во первых, это пока не затронуло все дистрибутивы, во вторых, это связано именно с пользовательскими сессиями и не так актуально на серверах. Про systemd мы ещё поговорим, я же просто объяснил, что бывает с дочерними процессами.



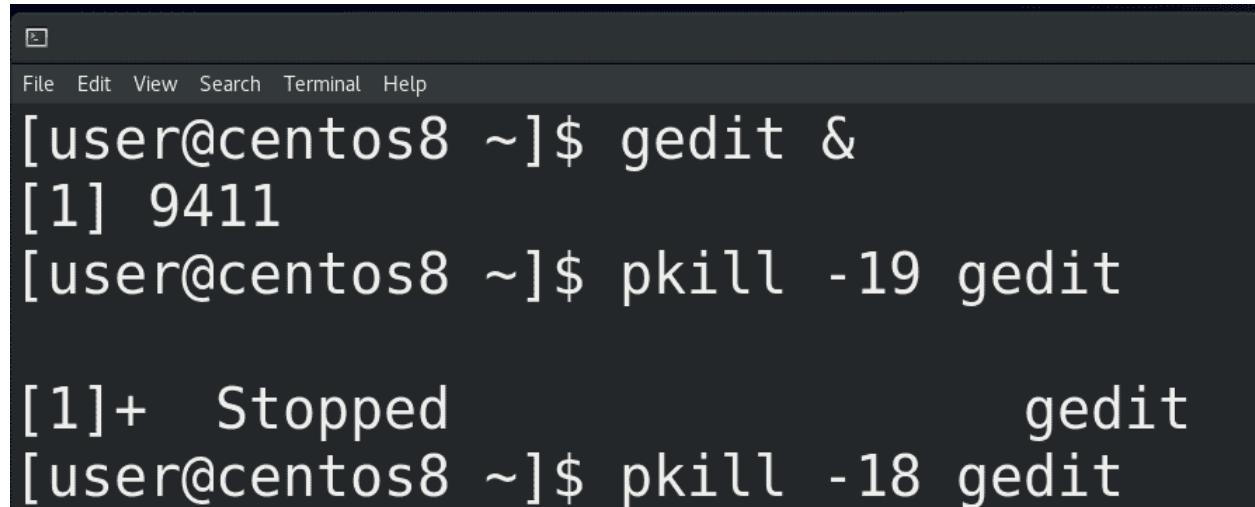
```
[user@centos8 ~]$ ps -ef | grep gedit
user      8149  2508  0 11:23 ?        00:00:00 gedit
user      9098  8167  0 11:34 pts/0      00:00:00 grep --
[user@centos8 ~]$ kill -9 8149
[user@centos8 ~]$ kill -SIGKILL 8149
```

Вернёмся к сигналам. Когда мы посыпаем SIGTERM, процесс сам отвечает за своё завершение. Но что если процесс завис, ни на что не реагирует? На самом деле, в таких случаях лучше разобраться, с чем это связано. Есть способ по жёсткому избавиться от процесса, использовав сигнал – 9) SIGKILL. Но это очень опасный сигнал – резкое убийство процесса, без возможности завершить все дела, может повредить базу данных, файловую систему и т.п. Поэтому нужно быть крайне аккуратным с этим сигналом и лучше постараться найти причину проблемы и попытаться решить её, а не убивать процесс. Но всё же, если нет другого выхода – посыпаем:

```
kill -9 pid
```

Можно ещё вместо номеров использовать сами сигналы:

```
kill -SIGKILL pid
```



```
[user@centos8 ~]$ gedit &
[1] 9411
[user@centos8 ~]$ pkill -19 gedit

[1]+ Stopped gedit
[user@centos8 ~]$ pkill -18 gedit
```

Большинство сигналов нужны не столько администраторам, сколько разработчикам. Но пара занятых сигналов всё же есть, допустим - 19) SIGSTOP - о котором я говорил в прошлый раз. Потренируемся на том же блокноте. Чтобы постоянно не искать id процесса, я могу использовать команду pkill. Она сама ищет процесс по шаблону, но нужно быть осторожным, потому что в некоторых случаях у разных процессов могут быть совпадения и это может привести к плохим последствиям. Но так как блокнот запущен один, я пишу:

```
pkill -19 gedit
```

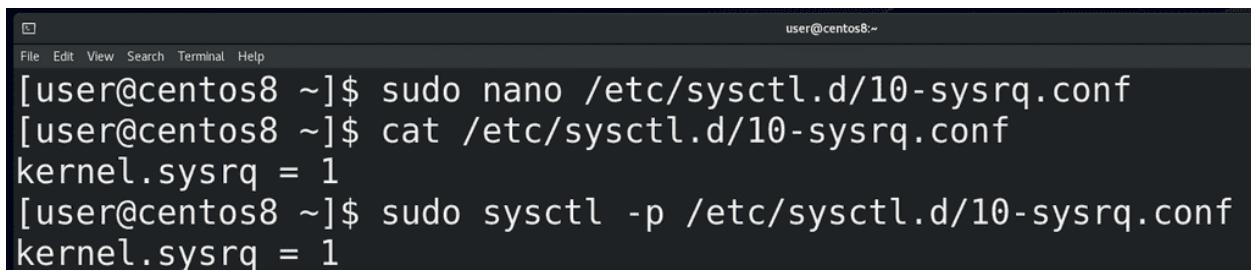
И теперь блокнот ни на что не реагирует, как будто завис. Если я хочу чтобы он продолжил работу, я посылаю сигнал 18) SIGCONT - от слова continue:

```
pkill -18 gedit
```

и он опять продолжает работать.

И так, когда у вас начинает зависать система, смотрим top или htop, сортируем по sru или memory, а потом решаем, что делать с процессом – пытаемся найти причину и решить. Допустим, у вас шёл бэкап базы данных на внешний сервер, а он отвалился от сети и теперь база данных начала грузить процессор. Легче всего будет вернуть в работу бэкап сервер. Это просто пример, возможно не самый хороший, просто объясняющий, что делать. Если процесс не такой важный и это какая-то мелочь, типа блокнота, можно попытаться ему послать сигнал с помощью kill, для начала тот же SIGTERM. Старайтесь избегать использования SIGKILL, это крайняя мера.

Если у вас зависит графический интерфейс, попробуйте открыть виртуальный терминал – ctrl+alt+f3, f4 и т.п. и решайте проблему оттуда. Виртуальный терминал особо ресурсы не расходует, и, в случае тормозов, он работает лучше графического интерфейса. Ещё одна проблема, которая может быть – ваша система намертво зависает и не реагирует абсолютно ни на что. Зачастую, это связано с проблемой нехватки памяти – возможно какой-то багованный софт съел всю оперативную память, отчего вся система зависла. Это называется ООМ – out of memory – и различные компании предлагают свои превентивные меры против этой проблемы, достаточно погуглить oom killer linux. Но это всё превентивные меры, а если вы всё же столкнулись с проблемой, не спешите перезагружать систему. Вам поможет магическая кнопка sysrq. На современных клавиатурах её не всегда пишут, но обычно это та же клавиша PrtSc. Эта клавиша позволяет напрямую посыпать какие-то команды ядру.



```
[user@centos8 ~]$ sudo nano /etc/sysctl.d/10-sysrq.conf
[user@centos8 ~]$ cat /etc/sysctl.d/10-sysrq.conf
kernel.sysrq = 1
[user@centos8 ~]$ sudo sysctl -p /etc/sysctl.d/10-sysrq.conf
kernel.sysrq = 1
```

Например, чтобы решить проблему out of memory, стоит нажать Alt+PrtSc+f, тогда ядро избавится от проблемного процесса, чтобы освободить память. Но это нужно предварительно настроить. Для этого создадим файл внутри директории /etc/sysctl.d с расширением .conf:

```
sudo nano /etc/sysctl.d/10-sysrq.conf
```

Напишем в этом файле:

```
kernel.sysrq = 1
```

Сохраним, затем выполним команду:

```
sudo sysctl -p /etc/sysctl.d/10-sysrq.conf
```

Есть и другие способы, допустим, установка демона oom-killer (earlyoom, systemd-oom и т.п.), который будет решать проблему без ручного вмешательства. Но это отдельная тема.

Таким образом, мы разобрались с тем, где найти информацию по процессам, немного углубились в теорию и разобрали, как управлять процессами с помощью сигналов, какие есть превентивные меры – ulimit, oom killer-ы – и что делать, когда система тормозит или вообще зависла.

2.16.2 Практика

Вопросы

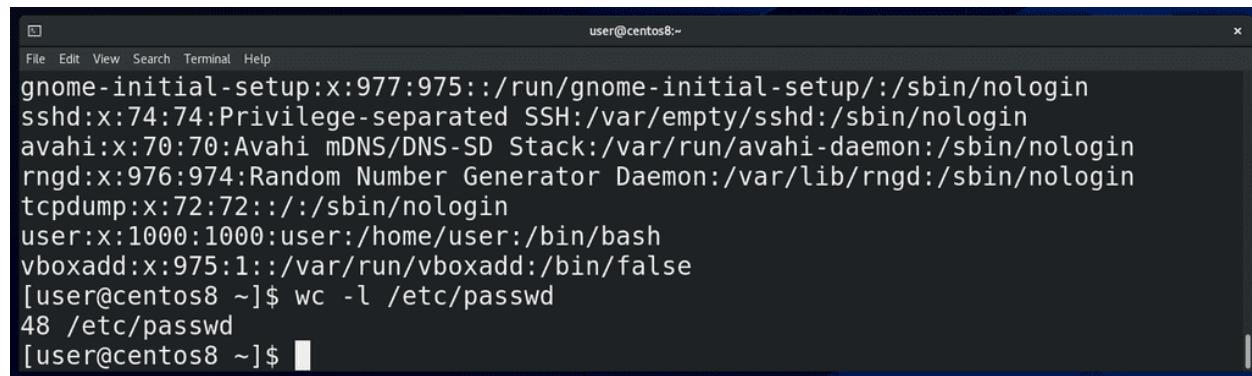
1. Что делает команда kill?
2. Как лучше всего убивать процессы?

Задания

1. Запустите любой процесс в фоновом режиме (к примеру, gedit). Найдите его родительский процесс. Найдите все дочерние процессы, которые принадлежат этому родительскому процессу. Убейте родительский процесс. Проверьте дочерние процессы и найдите новый родительский процесс.

2.17 17. su

2.17.1 17. su

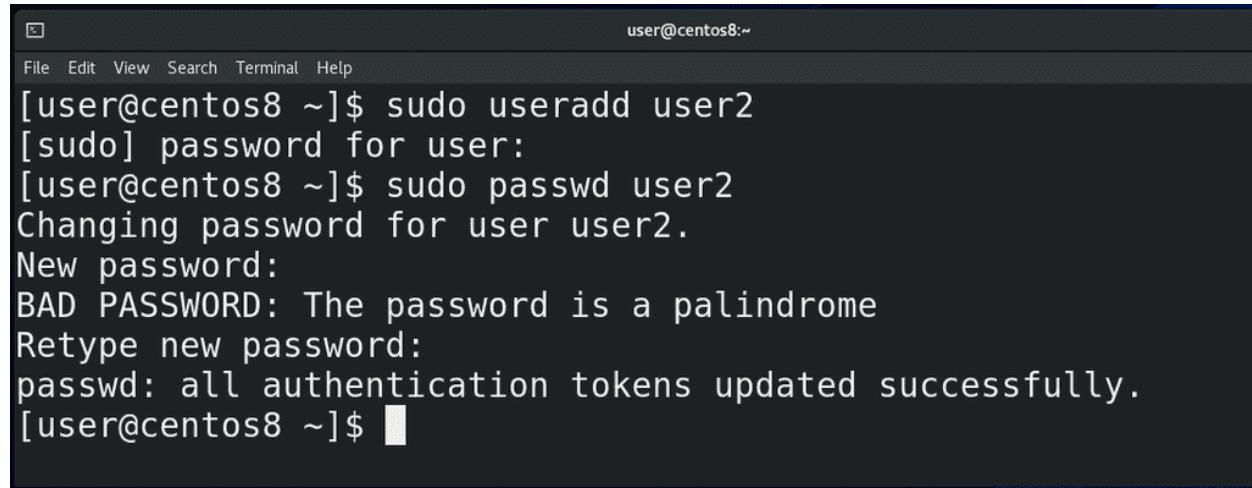


```
user@centos8:~$ gnome-initial-setup:x:977:975::/run/gnome-initial-setup/:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
avahi:x:70:70:Avahi mDNS/DNS-SD Stack:/var/run/avahi-daemon:/sbin/nologin
rngd:x:976:974:Random Number Generator Daemon:/var/lib/rngd:/sbin/nologin
tcpdump:x:72:72::/sbin/nologin
user:x:1000:1000:user:/home/user:/bin/bash
vboxadd:x:975:1::/var/run/vboxadd:/bin/false
[user@centos8 ~]$ wc -l /etc/passwd
48 /etc/passwd
[user@centos8 ~]$
```

Мы с вами уже выяснили, что Linux – система многопользовательская. Если с помощью команды `wc` посчитать количество строк в файле `/etc/passwd`:

```
cat /etc/passwd
wc -l /etc/passwd
```

где перечислены все пользователи, мы увидим, что сейчас в системе 48 пользователей. Среди них есть наш пользователь `user`, суперпользователь `root`, а остальные, которых создавали не мы, а система, считаются сервисными пользователями.



```
[user@centos8 ~]$ sudo useradd user2
[sudo] password for user:
[user@centos8 ~]$ sudo passwd user2
Changing password for user user2.
New password:
BAD PASSWORD: The password is a palindrome
Retype new password:
passwd: all authentication tokens updated successfully.
[user@centos8 ~]$
```

Нам сегодня понадобится ещё один пользователь, поэтому создадим его с помощью команды:

```
sudo useradd user2
```

введём наш пароль, а потом с помощью:

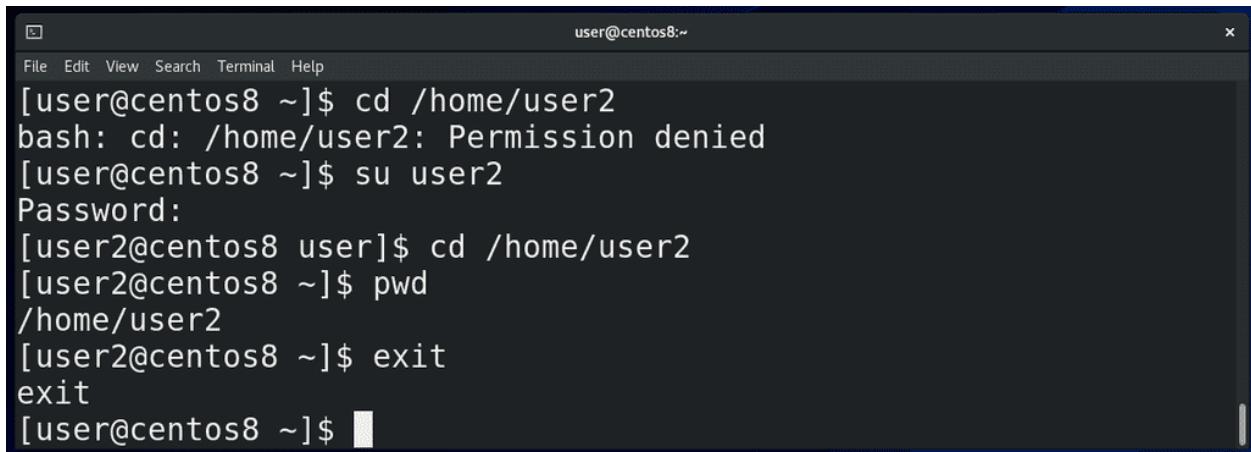
```
sudo passwd user2
```

зададим пароль для второго пользователя.

Начнём с команды `su`:

su

Она позволяет залогиниться каким-то пользователем или запускать команды от имени другого пользователя. При этом нужно знать пароль этого другого пользователя. Это бывает нужно, когда у нашего пользователя нет нужных прав, либо когда нам нужно запустить какой-то процесс от имени другого пользователя, например, в целях безопасности.



```
user@centos8:~$ su user2
Password:
[user2@centos8 ~]$ cd /home/user2
[user2@centos8 ~]$ pwd
/home/user2
[user2@centos8 ~]$ exit
exit
[user@centos8 ~]$
```

Например, сейчас мой пользователь не может зайти в директорию /home/user2:

cd /home/user2

потому что у него недостаточно прав. Я могу написать:

su user2

ввести пароль пользователя user2 и стать этим самым вторым пользователем, как видно в начале строки. А дальше смогу зайти в нужную директорию:

cd /home/user2

Чтобы вернуться к моему пользователю, я могу написать:

exit

либо нажать Ctrl+d. Если написать просто:

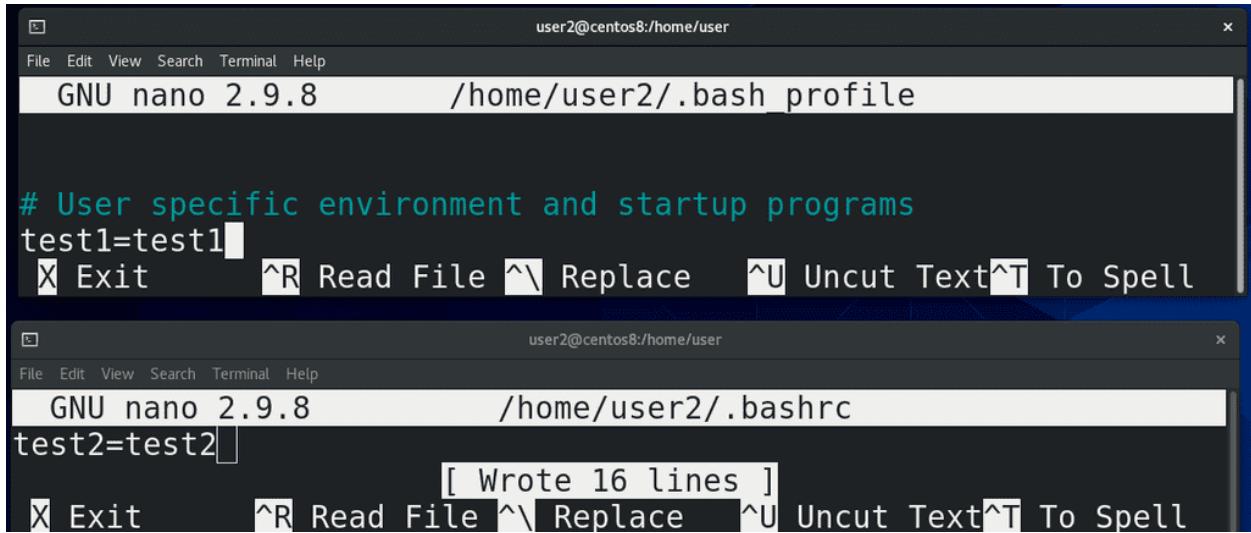
su

либо

su root

и ввести пароль rootа, можно работать от пользователя root.

Но помните мы разбирали файлы `~/.bash_profile` и `~/.bashrc`? Мы писали там переменные и алиасы, и, в случае с `~/.bash_profile`, нам нужно было перезалогиниться, а в случае с `~/.bashrc` нам достаточно было просто запустить новый эмулятор терминала. То есть `~/.bash_profile` это файл настроек для login shell, а `~/.bashrc` для nonlogin shell.



```
user2@centos8:/home/user
File Edit View Search Terminal Help
GNU nano 2.9.8      /home/user2/.bash_profile

# User specific environment and startup programs
test1=test1
X Exit ^R Read File ^\ Replace ^U Uncut Text^T To Spell

user2@centos8:/home/user
File Edit View Search Terminal Help
GNU nano 2.9.8      /home/user2/.bashrc

test2=test2
[ Wrote 16 lines ]
X Exit ^R Read File ^\ Replace ^U Uncut Text^T To Spell
```

Давайте сделаем вот что. У пользователя user2:

```
su user2
```

в файле `~/.bash_profile`:

```
nano /home/user2/.bash_profile
```

создадим переменную `test1` равную `test1`:

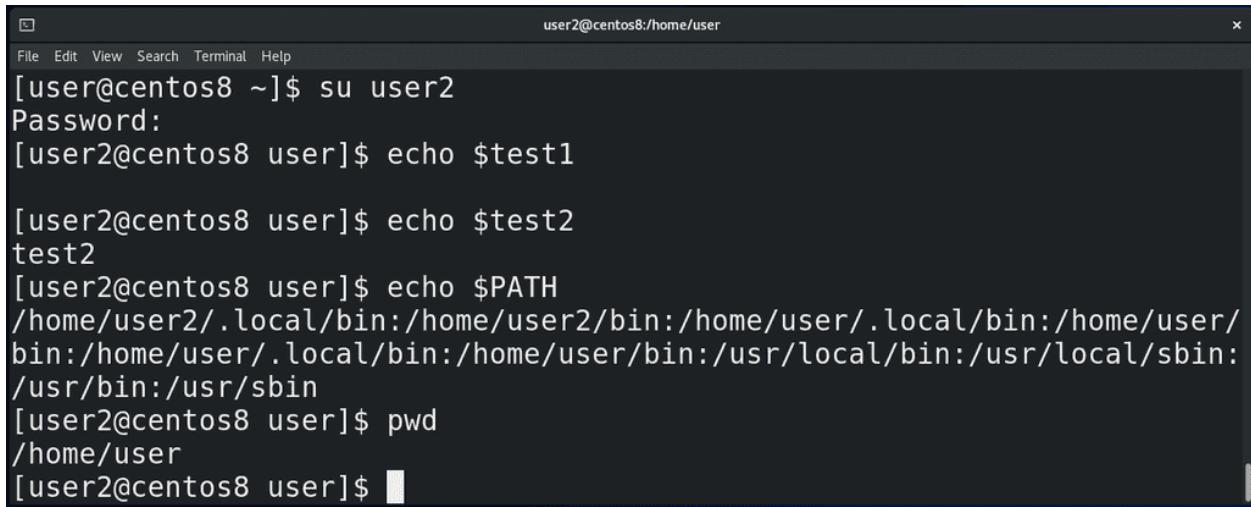
```
test1=test1
```

а в файле `~/.bashrc`:

```
nano /home/user2/.bashrc
```

переменную `test2` равную `test2`:

```
test2=test2
```



```
[user@centos8 ~]$ su user2
Password:
[user2@centos8 user]$ echo $test1
[user2@centos8 user]$ echo $test2
[user2@centos8 user]$ echo $PATH
/home/user2/.local/bin:/home/user2/bin:/home/user/.local/bin:/home/user/
bin:/home/user/.local/bin:/home/user/bin:/usr/local/bin:/usr/local/sbin:
/usr/bin:/usr/sbin
[user2@centos8 user]$ pwd
/home/user
[user2@centos8 user]$
```

Теперь выйдем - `ctrl+d` - заново зайдём:

```
su user2
```

и посмотрим, как обстоят дела с переменными:

```
echo $test1
echo $test2
```

Как видите, сработала настройка только из `~/.bashrc`, то есть non-login shell. Это означает, что переменные окружения не прочитались с `~/.bash_profile` (login shell) пользователя `user2`. На самом деле, все переменные остались от предыдущего пользователя. Допустим, если посмотреть переменную PATH:

```
echo $PATH
```

можно увидеть пути к директориям `/home/user/bin`, а это домашняя директория первого пользователя. Также, если писать `su user2`, можно заметить, что директория, в которой мы находимся, не меняется:

```
pwd
```

The screenshot shows a terminal window titled "user2@centos8:~". The window has a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The terminal prompt is "[user@centos8 ~]\$". The user runs the command "su - user2", followed by a password entry. After switching users, the prompt changes to "[user2@centos8 ~]\$". The user then runs "echo \$test1" and "echo \$test2", both of which output their respective values ("test1" and "test2"). Finally, the user runs "pwd" to show the current working directory, which is "/home/user2".

```
[user@centos8 ~]$ su - user2
Password:
[user2@centos8 ~]$ echo $test1
test1
[user2@centos8 ~]$ echo $test2
test2
[user2@centos8 ~]$ pwd
/home/user2
[user2@centos8 ~]$
```

Зачастую нужно, чтобы при логине за другого пользователя поменялось окружение, то есть, чтобы применились настройки из `~/.bash_profile` нужного пользователя. Для этого после `su` следует писать дефис:

```
su - user2
```

Теперь у меня есть обе переменные:

```
echo $test1
echo $test2
```

то есть считался файл `~/.bash_profile`, а значит это был login shell. Также стоит заметить, что при `su` с дефисом поменялась и директория – раньше мы находились в домашней директории пользователя `user`, а после «`su -»» меняется текущая директория – /home/user2.`

The screenshot shows a terminal window titled 'user2@centos8:~'. The terminal content is as follows:

```
[user@centos8 ~]$ su user2
Password:
[user2@centos8 user]$ echo $0
bash
[user2@centos8 user]$ exit
[user@centos8 ~]$ su - user2
Password:
[user2@centos8 ~]$ echo $0
-bash
[user2@centos8 ~]$ echo $PATH
/home/user2/.local/bin:/home/user2/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin
[user2@centos8 ~]$ █
```

Кстати, чтобы понять, текущий shell – login или non-login, не обязательно выдумывать каждый раз какие-то проверки с переменными, достаточно проверить значение переменной \$0:

```
echo $0
```

При non-login shell значение будет просто bash, а при login shell «-bash». И если проверить ту же переменную PATH:

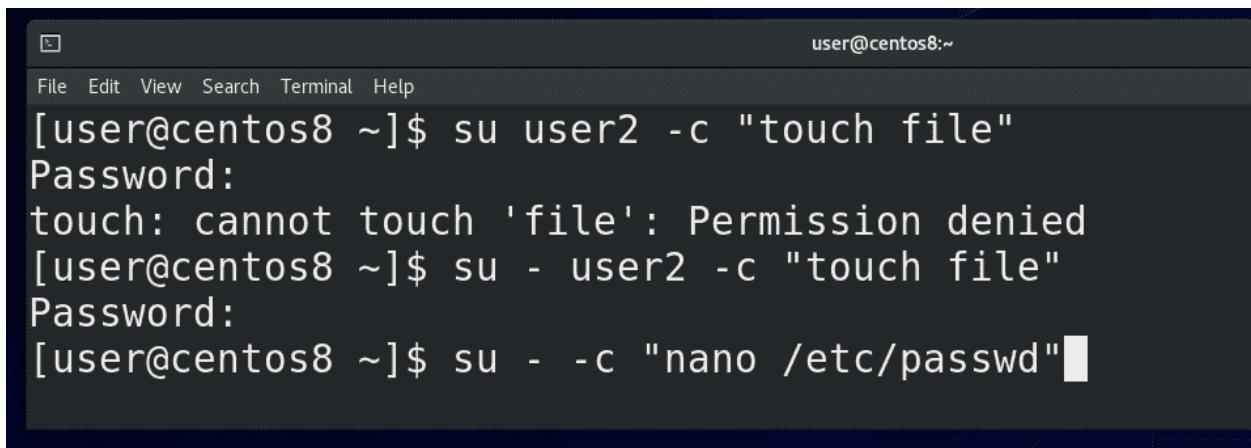
```
echo $PATH
```

можно увидеть, что теперь здесь нет пути /home/user/bin, то есть переменные окружения не передались от предыдущего пользователя, а появились как следует.

Так вот, подводя итоги. Когда вы вводите свой логин и пароль, будь то удалённо с помощью ssh, либо локально, зайдя в виртуальный терминал, либо залогинившись в графической оболочке - запускается оболочка со входом - login shell. Это оболочка с авторизацией, она считывает настройки – те же переменные, алиасы, функции сначала с файла /etc/profile, где написано смотреть на файлы в директории /etc/profile.d/ и в файл ~/.bash_profile в домашней директории пользователя. Там также написано смотреть в файл ~/.bashrc в домашней директории пользователя, в котором также написано смотреть в файл /etc/bashrc.

Когда же вы запускаете программу эмулятор терминала, то там нет логина, вы без логина и пароля можете вводить команды - оболочка без входа - non-login shell. В случае с non-login shell сначала считывается файл ~/.bashrc в домашней директории пользователя, затем считывается файл /etc/bashrc, который в свою очередь ссылается на файлы в /etc/profile.d/. Но приведённая схема может отличаться на других дистрибутивах.

Кроме этого, оболочки делятся на interactive и non-interactive. Если вы логинитесь и запускаете команды - это interactive login shell. Если вы работаете в эмуляторе терминала - это interactive non-login shell. Когда работают скрипты, они обычно запускаются без всякого логина - тут уже non-interactive non-login shell. Да, когда вы запускаете эмулятор терминала, в этом эмуляторе запускается bash и запускает ~/.bashrc. Сами по себе ~/.bashrc и ~/.bash_profile - это просто набор команд в одном файле. То есть это скрипты. Когда же вы логинитесь, не важно каким образом, тоже запускается bash, при этом он запускает ~/.bash_profile.



The screenshot shows a terminal window with the title bar "user@centos8:~". The menu bar includes "File", "Edit", "View", "Search", "Terminal", and "Help". The command history shows:

```
[user@centos8 ~]$ su user2 -c "touch file"
Password:
touch: cannot touch 'file': Permission denied
[user@centos8 ~]$ su - user2 -c "touch file"
Password:
[user@centos8 ~]$ su - -c "nano /etc/passwd"
```

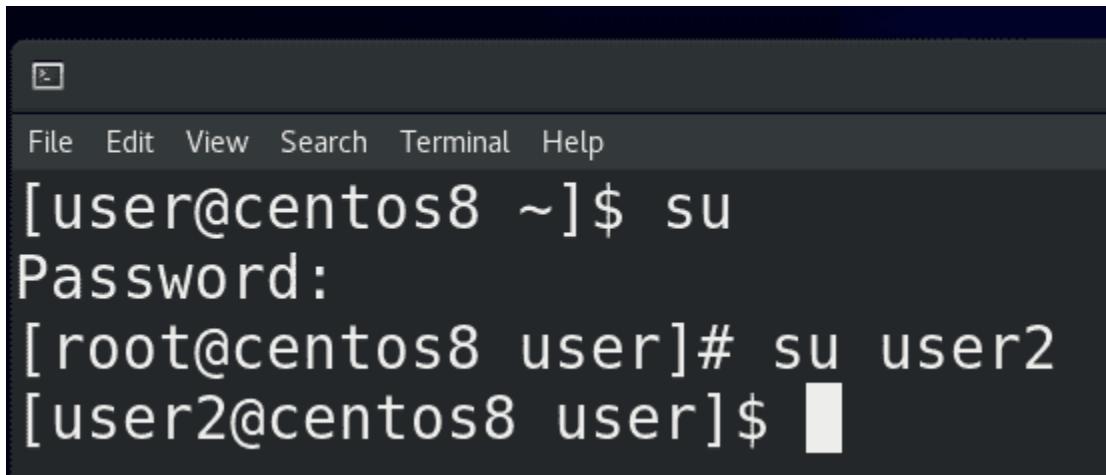
Ладно, со сменой пользователя разобрались. Мы еще говорили, что `su` позволяет запускать команды от имени другого пользователя. Для этого используется ключ `-c`. Например:

```
su user2 -c "touch file"
su - user2 -c "touch file"
```

Подумайте, почему первая команда завершилась с ошибкой, а вторая без? Ну и зачастую, `su` используют чтобы работать от имени `root` пользователя, для примера запустим `nano` от рута:

```
su - -c "nano /etc/passwd"
```

и теперь мы можем редактировать файл `/etc/passwd` и в целом можем делать всё что угодно.



The screenshot shows a terminal window with the title bar "user@centos8:~". The menu bar includes "File", "Edit", "View", "Search", "Terminal", and "Help". The command history shows:

```
[user@centos8 ~]$ su
Password:
[root@centos8 user]# su user2
[user2@centos8 user]$
```

Кстати, если запускать `su` от имени рута:

```
su
su user2
```

то никаких паролей не потребуется, `root` может логиниться кем угодно.

2.17.2 Практика

Вопросы

1. Что такое и чем отличаются login shell, non-login shell, interactive и non-interactive shell?
2. Для чего нужна команда su?
3. Чей пароль нужно вводить при использовании su?
4. Какие файлы задействованы в формировании окружения?

Задания

1. С помощью переменных докажите, что login shell и non-login shell отличаются.

2.18 18. sudo

2.18.1 18. sudo

У su есть один нюанс – нужно знать пароли других пользователей, если, конечно, вы не root. Когда вы один админ в системе – это не проблема. Но представьте, что есть несколько пользователей или администраторов. Давать друг другу пароли плохая идея, потому что безопасность строится на полном недоверии друг к другу, включая других администраторов. Другой администратор может умышленно вас подставить, неопытный хелпдеск может сделать ошибку от вашего имени, завтра могут уволить кого-то, а он от вашего имени может положить все сервера. Ну и если не говорить о недоверии, когда несколько админов работают от имени одного пользователя, то кто-то может что-то неправильно настроить, и потом не вспомнит или не сознается. Таким образом в коллективе формируется недоверие и негативная атмосфера. Учётные записи придумали не просто так, в жизни бывает всякое, поэтому всем будет легче работать от своего имени. Ну и во вторых, когда работает несколько пользователей, допустим если есть хелпдеск и вы хотите дать ему возможность только создавать пользователей, то вроде ему и нужны какие-то права суперпользователя, но и давать пароль от рута не хочется.

Для решения таких проблем есть утилита sudo:

```
sudo
```

Вы уже замечали её пару раз – с помощью неё мы устанавливали какие-то программы и создавали пользователей – потому что у нашего пользователя не было соответствующих прав. А если писать перед командой sudo – например:

```
sudo useradd user2
```

то команда useradd запускается от имени суперпользователя. Но, естественно, не любой пользователь может написать sudo и выполнить любую команду – у нашего user это работает, потому что при установке системы мы поставили галочку «сделать пользователя администратором». Как это работает – давайте разбераться.

Вкратце, sudo позволяет запускать какие-то команды от имени какого-то пользователя. А в файле sudoers пишутся те самые политики – кому, где и что. Из того, что мы уже видели – наш пользователь может запускать команды от имени суперпользователя. Давайте для начала найдём, где это написано.

```
## Sudoers allows particular users to run various commands as
## the root user, without needing the root password.
##
## Examples are provided at the bottom of the file for collections
## of related commands, which can then be delegated out to particular
## users or groups.
##
## This file must be edited with the 'visudo' command.

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell  ^_ Go To Line
```

У sudo есть файл настроек - /etc/sudoers:

```
nano /etc/sudoers
```

Наш пользователь его читать не может, но так как у нас есть права суперпользователя, можем написать:

```
sudo nano /etc/sudoers
```

введём свой пароль и файл откроется. В начале у нас есть строчка – этот файл нужно редактировать с помощью команды visudo:

```
visudo
```

Всё дело в том, что при запуске sudo программа каждый раз считывает этот файл, и, если здесь будет синтаксическая ошибка, sudo просто перестанет работать. А если у нас нет пароля от рута – то придётся повозиться, чтобы восстановить работу.

```
asdasdasd

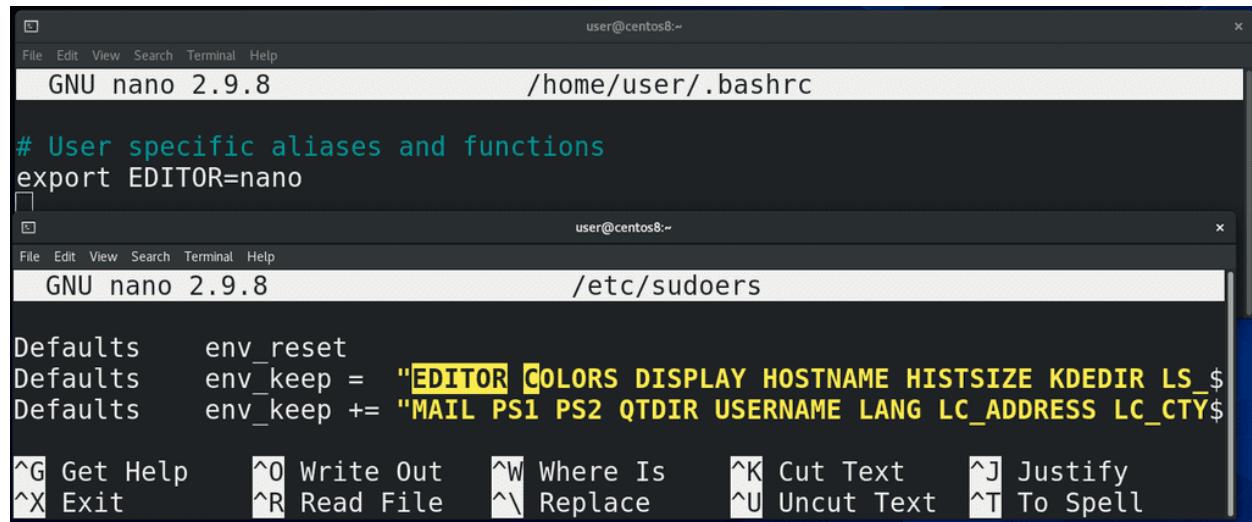
^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell
```

```
[user@centos8 ~]$ sudo nano /etc/sudoers
>>> /etc/sudoers: syntax error near line 123 <<<
sudo: parse error in /etc/sudoers near line 123
sudo: no valid sudoers sources found, quitting
sudo: unable to initialize policy plugin
[user@centos8 ~]$ █
```

Давайте просто покажу – напишу в файле рандомные буквы, сохранию и выйду. Теперь, при попытке заново открыть файл с помощью sudo, у меня ничего не откроется - sudo скажет, что не смог прочесть такую-то строчку и просто откажется работать. К счастью, у меня есть пароль от рута, который яставил при установке системы – я могу просто написать su, ввести пароль рута, а потом просто зайти и удалить лишнюю строчку:

```
nano /etc/sudoers
```

Теперь sudo опять будет работать.



Попытаемся прислушаться к совету и запустить команду visudo:

```
sudo visudo
```

Visudo открывает /etc/sudoers в дефолтном редакторе – vim. Если же я хочу, чтобы всё работало с nano, мне нужно будет кое-что поменять. Как правило, программы по умолчанию задаются с помощью переменных окружения. Как это делать мы уже знаем – зайдём в ~/.bashrc:

```
nano ~/.bashrc
```

и напишем:

```
export EDITOR=nano
```

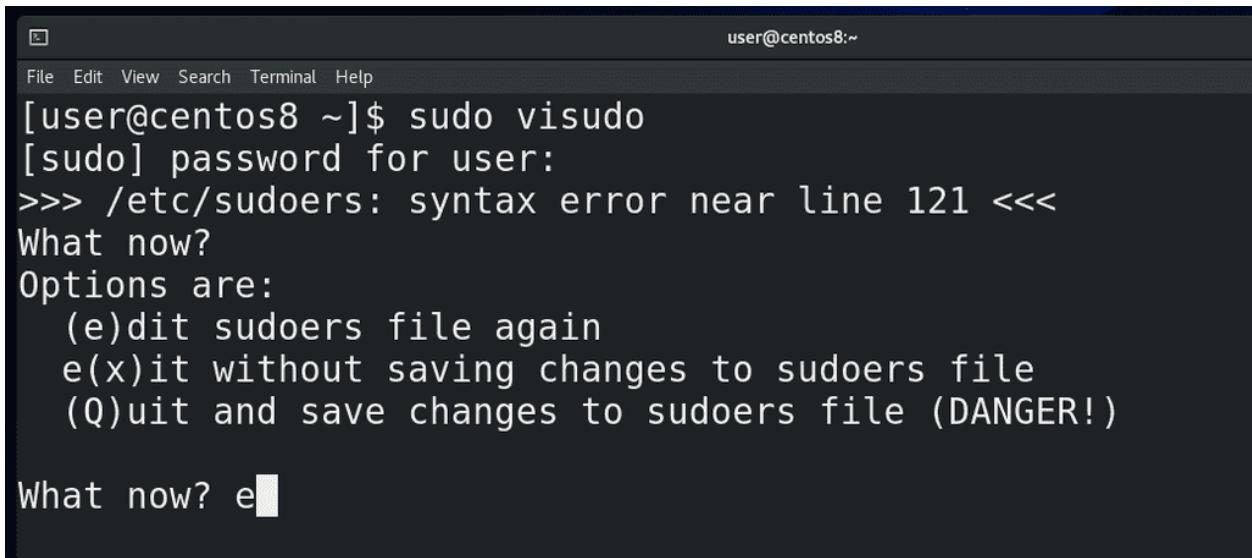
Дальше нам нужно зайти в sudoers:

```
sudo nano /etc/sudoers
```

найти строчку с env_keep. ВЫше можно заметить env_reset – которая сбрасывает все переменные, чтобы сделать окружение с sudo минимальным. Так вот, в env_keep добавляем EDITOR, чтобы sudoers не игнорировал нашу переменную, сохраним, закроем эмулятор и заново запустим, чтобы bash перечитал настройки ~./.bashrc. И теперь, при запуске:

```
sudo visudo
```

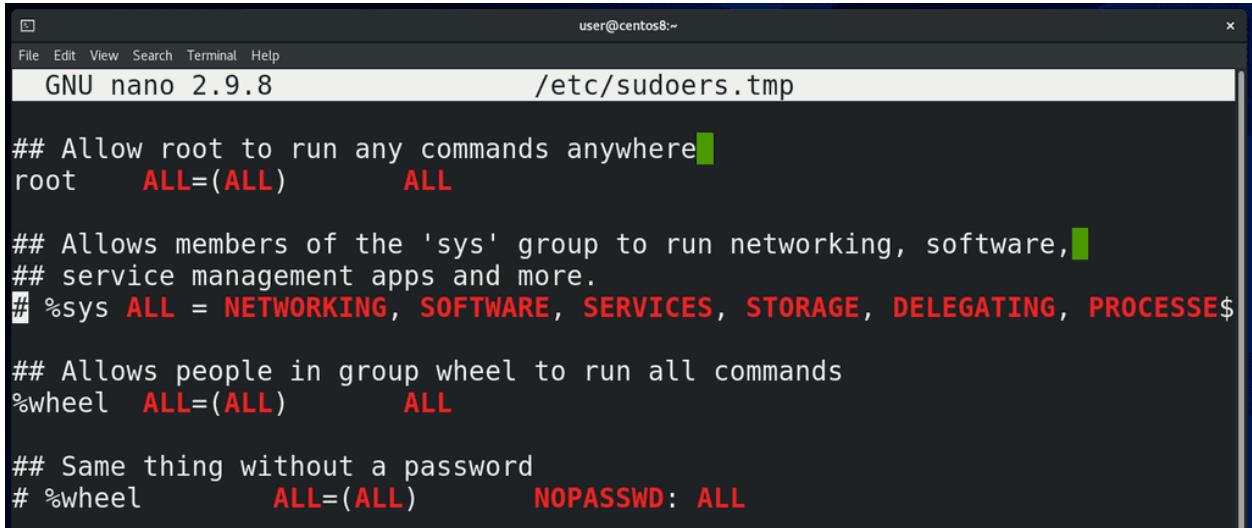
файл откроется в nano.



```
[user@centos8 ~]$ sudo visudo
[sudo] password for user:
>>> /etc/sudoers: syntax error near line 121 <<<
What now?
Options are:
  (e)dit sudoers file again
  e(x)it without saving changes to sudoers file
  (Q)uit and save changes to sudoers file (DANGER!)

What now? e
```

Обратите внимание - наверху указано не /etc/sudoers, а /etc/sudoers.tmp. Это означает, что когда мы запускаем visudo, у нас sudoers копируется во временный файл. Если здесь сделать ошибку, например, написать рандомные буквы, сохранить и попытаться выйти, то visudo сначала проверит, всё ли нормально с файлом, заметит синтаксическую ошибку и спросит нас – что делать? Если нажать enter, можно увидеть варианты – e чтобы вернуться к редактированию, x – чтобы выйти не сохраняя, Q – чтобы сохранить, но мы видим предупреждение, что это опасно. Можно вернуться, исправить ошибку, сохранить и выйти. При сохранении, sudoers.tmp заменит оригинальный файл sudoers. Таким образом visudo защищает нас от синтаксических ошибок.



```
user@centos8:~
```

```
File Edit View Search Terminal Help
```

```
GNU nano 2.9.8          /etc/sudoers.tmp
```

```
## Allow root to run any commands anywhere
root    ALL=(ALL)      ALL

## Allows members of the 'sys' group to run networking, software,
## service management apps and more.
%sys   ALL = NETWORKING, SOFTWARE, SERVICES, STORAGE, DELEGATING, PROCESSES

## Allows people in group wheel to run all commands
%wheel  ALL=(ALL)      ALL

## Same thing without a password
# %wheel      ALL=(ALL)      NOPASSWD: ALL
```

Спускаемся вниз, где у нас начинаются политики. Например:

```
root ALL=(ALL) ALL
```

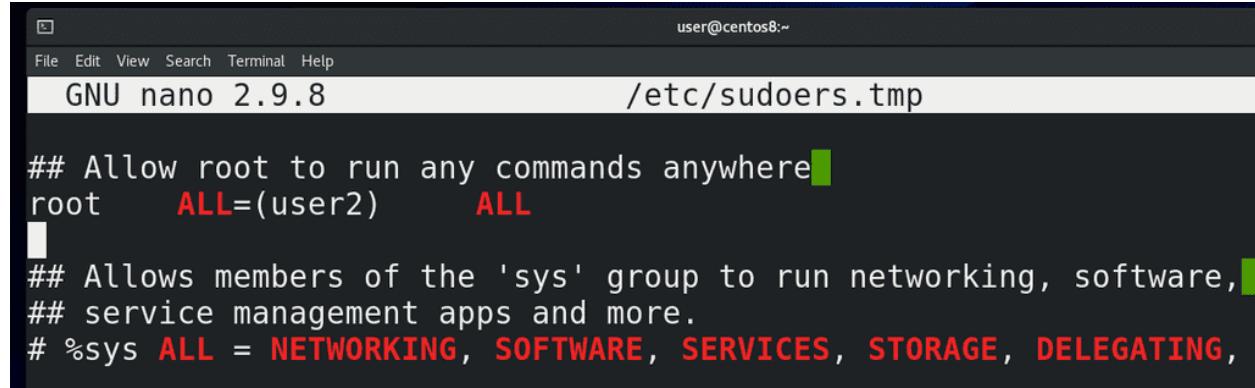
Первое – это пользователь, для которого написана политика. Если начинается со знака процента (%), то речь о группе пользователей. Не то, чтобы руту были нужны sudo права – он и так может делать всё что угодно. Но без этой строчки – если root почему-то запустит sudo, например, если это написано в скрипте или кто-то скопировал команды с интернета – sudo выдаст ошибку и команда не сработает. Поэтому, на всякий случай, есть эта строчка.

Теперь, что означает ALL=(ALL) ALL. Первая ALL – это на каком компьютере. Если первое значение

ALL или совпадает с именем компьютера, которое можно узнать с помощью команды:

```
hostname
```

то sudo будет работать с этой строчкой. Если же тут написано что-то другое – то sudo проигнорирует эту строчку. Если у вас несколько компьютеров, вы можете написать один файл sudoers и закинуть его на все компьютеры. И sudo на каждой машинке будет считывать только строчки, где хост равен ALL или совпадает с хостнеймом машинки.



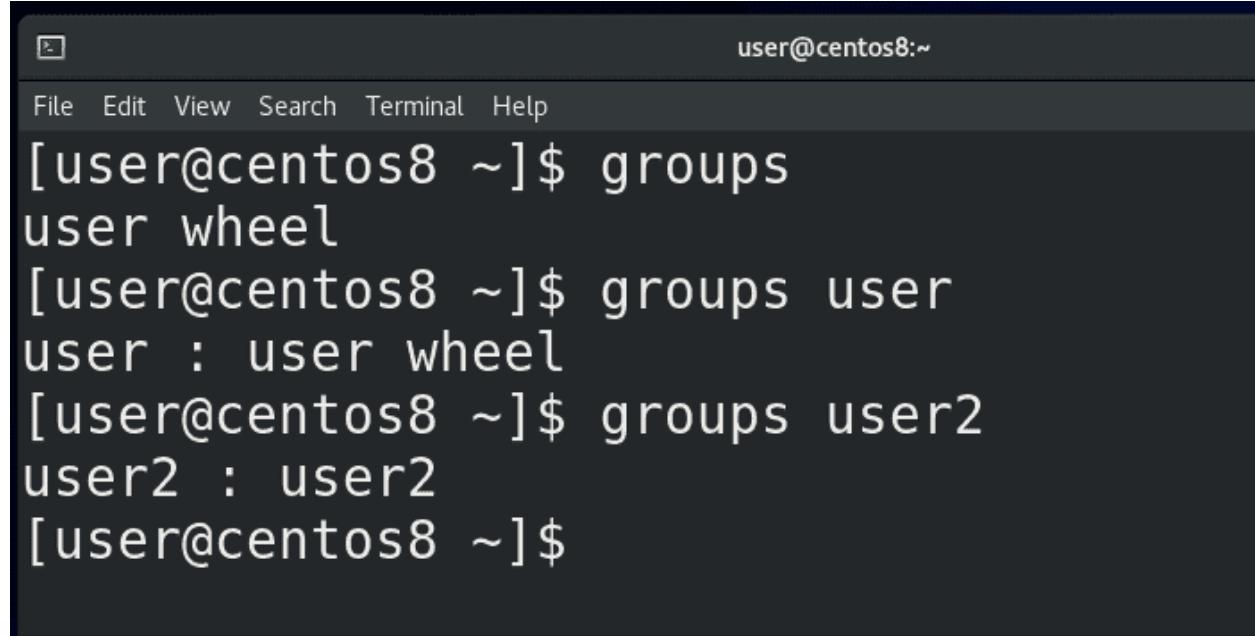
```
GNU nano 2.9.8 /etc/sudoers.tmp

## Allow root to run any commands anywhere
root    ALL=(user2)      ALL

## Allows members of the 'sys' group to run networking, software,
## service management apps and more.
# %sys ALL = NETWORKING, SOFTWARE, SERVICES, STORAGE, DELEGATING,
```

Вторая ALL – от имени какого пользователя, ALL значит от всех, то есть можно запустить команду от имени любого пользователя. По умолчанию, если явно не указывать пользователя, то команда запустится от имени root. А так, здесь можно написать конкретного пользователя, допустим user2. В таком случае пользователь root сможет запускать команды с помощью sudo только от имени user2. Также тут можно указать не только пользователя, а также группу, или просто группу. Чуть дальше будут примеры.

И третья ALL – это команды. В случае ALL можно запустить любую команду, а так, здесь можно прописать только те команды, которые вы хотите разрешить. Или наоборот, если вы хотите какие-то команды разрешить и какие-то запретить. Перед запрещёнными командами ставится восклицательный знак.



```
[user@centos8 ~]$ groups
user wheel
[user@centos8 ~]$ groups user
user : user wheel
[user@centos8 ~]$ groups user2
user2 : user2
[user@centos8 ~]$
```

Так вот, почему наш пользователь user может запускать команды с помощью sudo? Пока что в этом

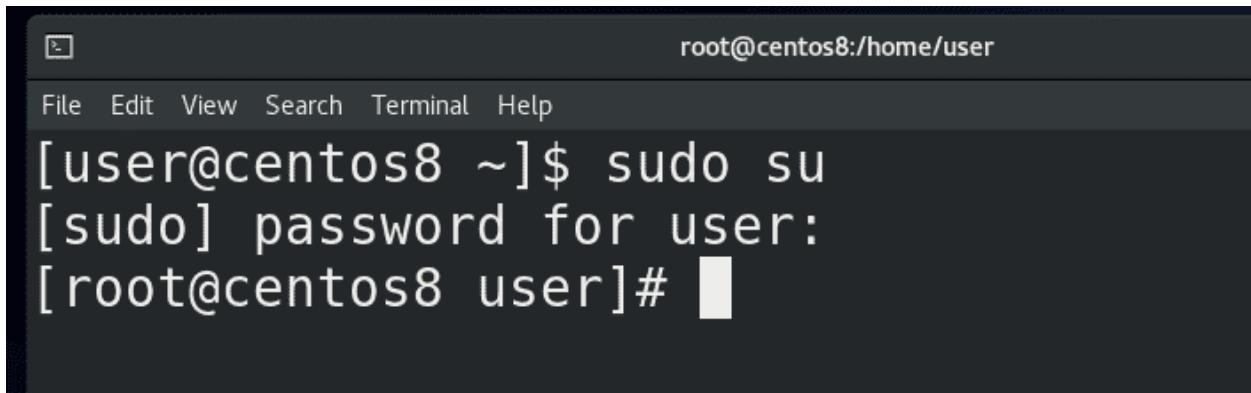
файле всего 2 политики – для пользователя root и для группы wheel, всё остальное – закомментированные примеры. У группы wheel тоже все права – ALL ALL ALL. Чтобы узнать, в каких группах состоит наш пользователь, можно выполнить команду:

```
groups
```

от его имени, либо указать имя пользователя:

```
groups
groups user
groups user2
```

Как видно, наш пользователь user состоит в группе wheel – именно поэтому у него есть все права на систему. В других дистрибутивах вместо wheel может быть группа с именем sudo – но суть одна и та же.

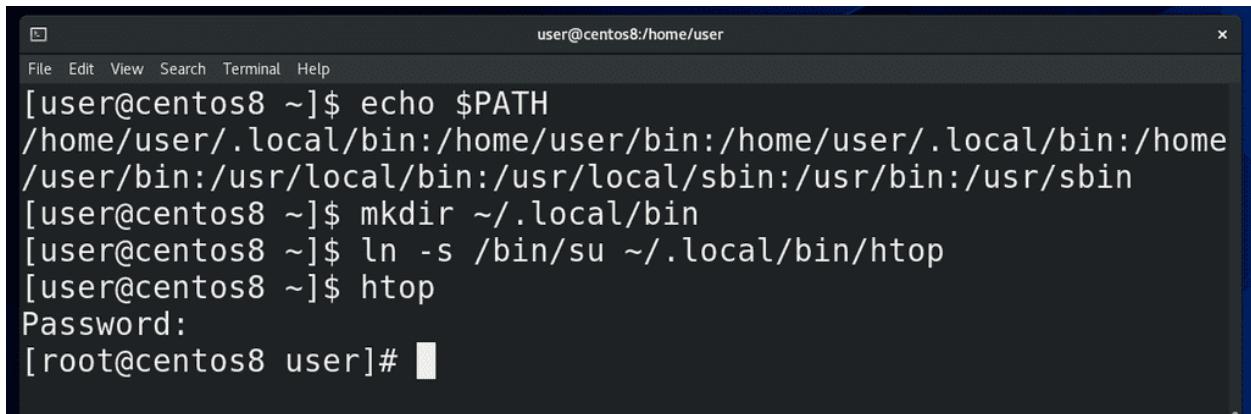


The screenshot shows a terminal window titled 'root@centos8:/home/user'. The window has a dark background and light-colored text. At the top, there's a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. Below the menu, the command '[user@centos8 ~]\$ sudo su' is entered. A password prompt '[sudo] password for user:' follows. After entering the password, the terminal shows '[root@centos8 user]#', indicating successful root login.

Прежде чем пойдём дальше, давайте я покажу частный случай использования sudo:

```
sudo su
```

Если выполнить просто команду su, то мы меняем текущего пользователя на root пользователя, правда для этого нам нужно знать пароль этого рут пользователя. Команда sudo позволяет нам выполнить команду от имени суперпользователя, а мы знаем, что если выполнять команду su от имени root пользователя, то пароль нам указывать не нужно. А значит выполнив sudo su и введя пароль своего юзера, мы просто от имени суперпользователя запустим команду su, а так как ему не нужно вводить пароль, мы просто станем рутом. Таким образом, не зная пароля рута, мы можем стать рут пользователем.



The screenshot shows a terminal window titled 'user@centos8:/home/user'. The window has a dark background and light-colored text. At the top, there's a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. Below the menu, the command '[user@centos8 ~]\$ echo \$PATH' is entered, followed by several directory paths. Then '[user@centos8 ~]\$ mkdir ~/.local/bin', '[user@centos8 ~]\$ ln -s /bin/su ~/.local/bin/htop', and '[user@centos8 ~]\$ htop'. A password prompt 'Password:' appears, followed by '[root@centos8 user]#', indicating successful root login.

Учитывая, что sudo это механизм, который позволяет повысить свои привилегии, этим активно пользуются злоумышленники. Давайте, для примера, посмотрим один из теоретических способов с помощью

переменной PATH. У меня в переменной PATH:

```
echo $PATH
```

есть путь /home/user/.local/bin. Сейчас этой директории нет, но я могу её создать:

```
mkdir ~/.local/bin
```

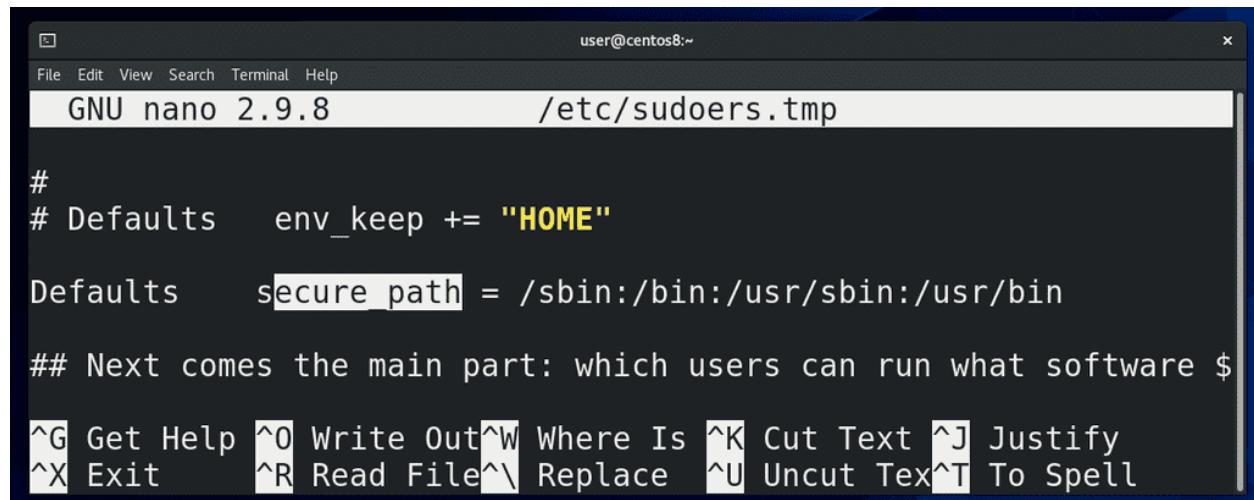
и сделать такую обманку – создать символьическую ссылку:

```
ln -s /bin/su ~/.local/bin/htop
```

Теперь когда я пишу htop, bash смотрит переменную PATH, видит в начале ~/.local/bin, находит там программу htop и запускает, а на самом деле это программа su. И представьте ситуацию, если мы разрешим пользователю запускать только команду htop с правами суперпользователя, допустим, чтобы понижать nice ness, а этот пользователь напишет:

```
sudo htop
```

на деле выполнится sudo su – и он введя свой пароль станет root пользователем.



```
GNU nano 2.9.8          /etc/sudoers.tmp

#
# Defaults    env_keep += "HOME"

Defaults    secure_path = /sbin:/bin:/usr/sbin:/usr/bin

## Next comes the main part: which users can run what software $

^G Get Help ^O Write Out^W Where Is ^K Cut Text ^J Justify
^X Exit      ^R Read File^\\ Replace   ^U Uncut Tex^T To Spell
```

Естественно, этой очень банальный способ и его невозможно применить – во-первых, sudo требует, чтобы в sudoers был полный путь к программам. То есть, если в sudo явно не указан путь /home/user/~/local/bin/htop – то ничего не сработает. Во вторых – в sudoers есть настройка secure_path. Когда пользователь будет писать sudo command, то sudo будет смотреть только в директории, указанные в этой настройке. Кстати, насчёт полного пути – его всегда можно легко найти с помощью команды which – допустим:

```
which htop
which ls
which rm
```

Но нужно понимать, что при ALL ALL ALL у пользователя будет полный доступ, а если вы ограничиваете пользователя определёнными командами, следует быть предельно осторожным с выбором команд, потому что очень часто можно повысить привилегии неочевидным способом. Допустим, давая кому-то право создавать пользователей, он может создать пользователя с группой wheel и через него стать rootом. Поэтому, прежде чем давать какому-то пользователю права на какую-то программу, следует очень внимательно изучить, что эта программа позволяет сделать.

The screenshot shows three terminal windows. The top window displays the contents of /etc/sudoers.tmp:

```
user2  ALL=(user:group)  NOEXEC: /usr/bin/less /home/user/file
user2  ALL=(root)  NOPASSWD: /bin/cat, !/bin/cat /etc/shadow

## Allows members of the 'sys' group to run networking, software,
```

The middle window shows the output of a sudo command:

```
[user2@centos8 user]$ sudo cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
```

The bottom window shows the result of attempting to execute a command without a password:

```
[user2@centos8 user]$ sudo cat /etc/shadow
Sorry, user user2 is not allowed to execute '/bin/cat /etc/shadow' as root on centos8.
[user2@centos8 user]$
```

Допустим, тот же less или vi позволяют запускать команды и для них есть специальный ключ, позволяющий предотвратить выполнение сторонних команд - NOEXEC. Кроме NOEXEC есть ещё пара ключей, один из примечательных - NOPASSWD – позволяет запускать указанные команды без ввода пароля. Также, чтобы постоянно не вводить sudo, можно предварительно написать:

```
sudo -s
```

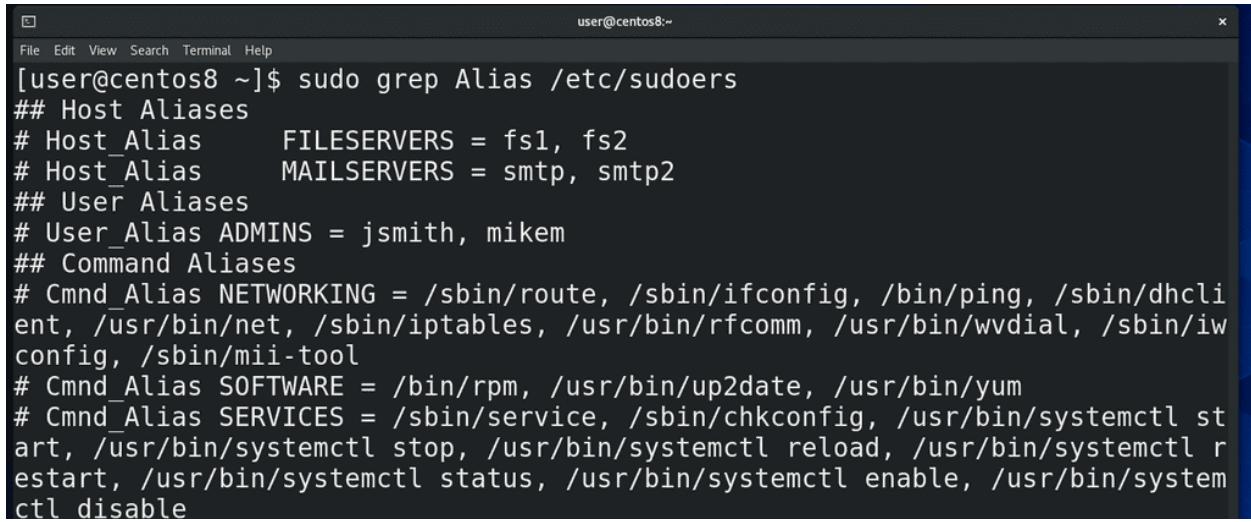
В некоторых случаях люди используют sudo, чтобы предоставить доступ к редактированию каких-то файлов. И хотя лучше в таких случаях использовать права на файлы, ситуации бывают разные, поэтому у sudo есть относительно безопасный способ редактирования файлов с помощью sudoedit.

The screenshot shows a terminal window with the following content:

```
user2  ALL=(user:group)  NOEXEC: /usr/bin/less /home/user/file
user2  ALL=(root)  NOPASSWD: /usr/bin/sudoedit, /etc/passwd

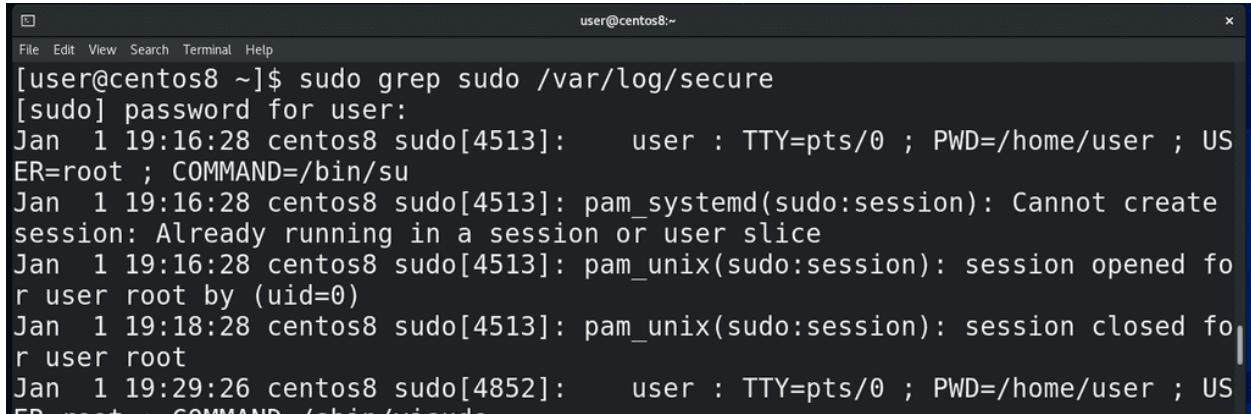
## Allows members of the 'sys' group to run networking, software,
[ Wrote 123 lines ]
^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify
^X Exit      ^R Read File   ^\ Replace   ^U Uncut Text  ^T To Spell
```

Для примера, давайте предоставлю пользователю user2 право редактировать файл /etc/passwd с помощью nano. Он вроде бы и не может открыть командой другой файл, но я могу изнутри nano открыть другой файл с помощью ^R, допустим, тот же sudoers, изменить его и сохранить как /etc/sudoers, тем самым обеспечив себе все права. Если же в sudoers я пропишу sudoedit, то предыдущая схема не будет работать, потому что принцип работы немного другой. sudoedit копирует нужный мне файл во временный файл, я редактирую временный файл, а когда сохраняю – то sudoedit заменяет нужный файл той копией, которую я изменил. Почти как с visudo.



```
[user@centos8 ~]$ sudo grep Alias /etc/sudoers
## Host Aliases
# Host_Alias      FILESERVERS = fs1, fs2
# Host_Alias      MAILSERVERS = smtp, smtp2
## User Aliases
# User_Alias ADMINNS = jsmith, mikem
## Command Aliases
# Cmnd_Alias NETWORKING = /sbin/route, /sbin/ifconfig, /bin/ping, /sbin/dhcclient, /usr/bin/net, /sbin/iptables, /usr/bin/rfcomm, /usr/bin/wvdial, /sbin/iwconfig, /sbin/mii-tool
# Cmnd_Alias SOFTWARE = /bin/rpm, /usr/bin/up2date, /usr/bin/yum
# Cmnd_Alias SERVICES = /sbin/service, /sbin/chkconfig, /usr/bin/systemctl start, /usr/bin/systemctl stop, /usr/bin/systemctl reload, /usr/bin/systemctl restart, /usr/bin/systemctl status, /usr/bin/systemctl enable, /usr/bin/systemctl disable
```

Когда у вас много пользователей, много разных команд и компьютеров, для облегчения прописывания политик можно использовать алиасы, где можно перечислить несколько значений через запятую. В файле уже даны примеры использования. Политика состоит из 4 столбиков: User – это собственно пользователь или группа, к которой применяется политика, поэтому алиас называется User_Alias. Во втором столбике хостнеймы, поэтому Host_Alias. Дальше у нас столбик, где указывается от чьего имени разрешено запускать – называется Runas – собственно, Runas_Alias и последний столбик – команды, поэтому здесь Cmnd_Alias. Ну и тут пример использования политик с алиасами.

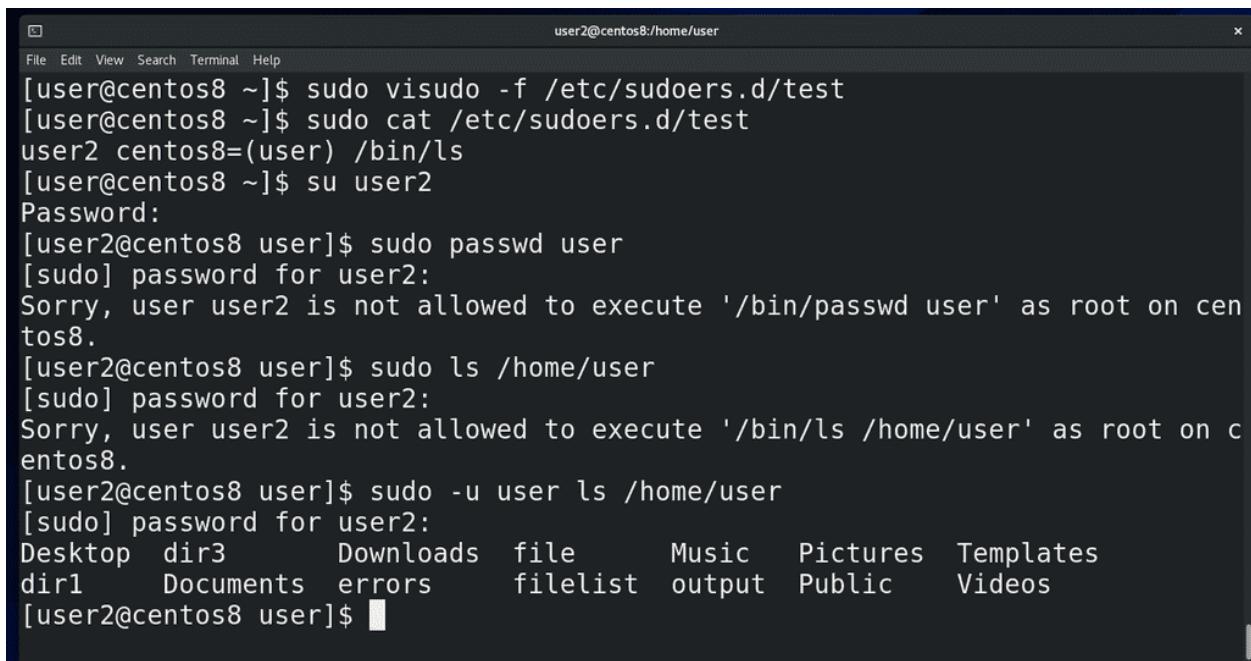


```
[user@centos8 ~]$ sudo grep sudo /var/log/secure
[sudo] password for user:
Jan  1 19:16:28 centos8 sudo[4513]:      user : TTY=pts/0 ; PWD=/home/user ; US
ER=root ; COMMAND=/bin/su
Jan  1 19:16:28 centos8 sudo[4513]: pam_systemd(sudo:session): Cannot create
session: Already running in a session or user slice
Jan  1 19:16:28 centos8 sudo[4513]: pam_unix(sudo:session): session opened fo
r user root by (uid=0)
Jan  1 19:18:28 centos8 sudo[4513]: pam_unix(sudo:session): session closed fo
r user root
Jan  1 19:29:26 centos8 sudo[4852]:      user : TTY=pts/0 ; PWD=/home/user ; US
ER=root ; COMMAND=/bin/su
```

Также бывает полезно узнать, кто какие команды вводил с помощью sudo. Все действия sudo логирует и их можно посмотреть в файле /var/log/secure:

```
sudo grep sudo /var/log/secure
```

Также, в плане логирования, у sudo есть инструмент, называемый sudoreplay. Я его разбирать не буду, это вам такое задание – настроить и проверить его работу. Если будут какие-то сложности – спрашивайте в комментариях.



The screenshot shows a terminal window titled 'user2@centos8:/home/user'. The user runs several commands to test sudo permissions:

```
[user@centos8 ~]$ sudo visudo -f /etc/sudoers.d/test
[user@centos8 ~]$ sudo cat /etc/sudoers.d/test
user2 centos8=(user) /bin/ls
[user@centos8 ~]$ su user2
Password:
[user2@centos8 user]$ sudo passwd user
[sudo] password for user2:
Sorry, user user2 is not allowed to execute '/bin/passwd user' as root on centos8.
[user2@centos8 user]$ sudo ls /home/user
[sudo] password for user2:
Sorry, user user2 is not allowed to execute '/bin/ls /home/user' as root on centos8.
[user2@centos8 user]$ sudo -u user ls /home/user
[sudo] password for user2:
Desktop dir3 Downloads file Music Pictures Templates
dir1 Documents errors filelist output Public Videos
[user2@centos8 user]$
```

Последняя строчка файла говорит нам, что sudoers будет читать настройки из всех файлов, расположенных внутри директории `/etc/sudoers.d`. Для этого нам понадобится указать для visudo файл:

```
sudo visudo -f /etc/sudoers.d/test
```

Давайте пропишем здесь правило для пользователя `user2`, допустим, чтобы он мог от имени пользователя `user` запускать команду `ls`:

```
user2 centos8=(user) /bin/ls
```

Сохраним, выйдем и проверим:

```
su user2
sudo passwd user
ls /home/user/
sudo -u user ls /home/user
```

Подводя итоги, `sudo` – инструмент, который позволяет дать определённым пользователям определённые права. Но это делает `sudo` очень опасным инструментом, которым активно пользуются злоумышленники. Поэтому нужно запомнить – без острой необходимости пользователей в `sudo` прописывать не стоит, если речь идёт о правах суперпользователя. Не стоит строить правила на основе запретов – разрешить всё, а потом запретить опасные команды. Это заведомо проигрышный вариант – есть огромное количество способов обойти запреты. Давайте доступ только на необходимые команды, заранее проанализируйте, посмотрите в интернете, а насколько опасно то, что вы разрешаете. В дальнейшем вы научитесь писать скрипты – так вот, вместо самих команд пишите скрипты, которые выполняют строго заданные функции и указывайте в `sudoers` эти скрипты вместо команд. Ну и у `sudo` много всяких настроек, которые я не рассмотрел – но для основ этого будет достаточно.

2.18.2 Практика

Вопросы

- Чем отличается `su` от `sudo`?

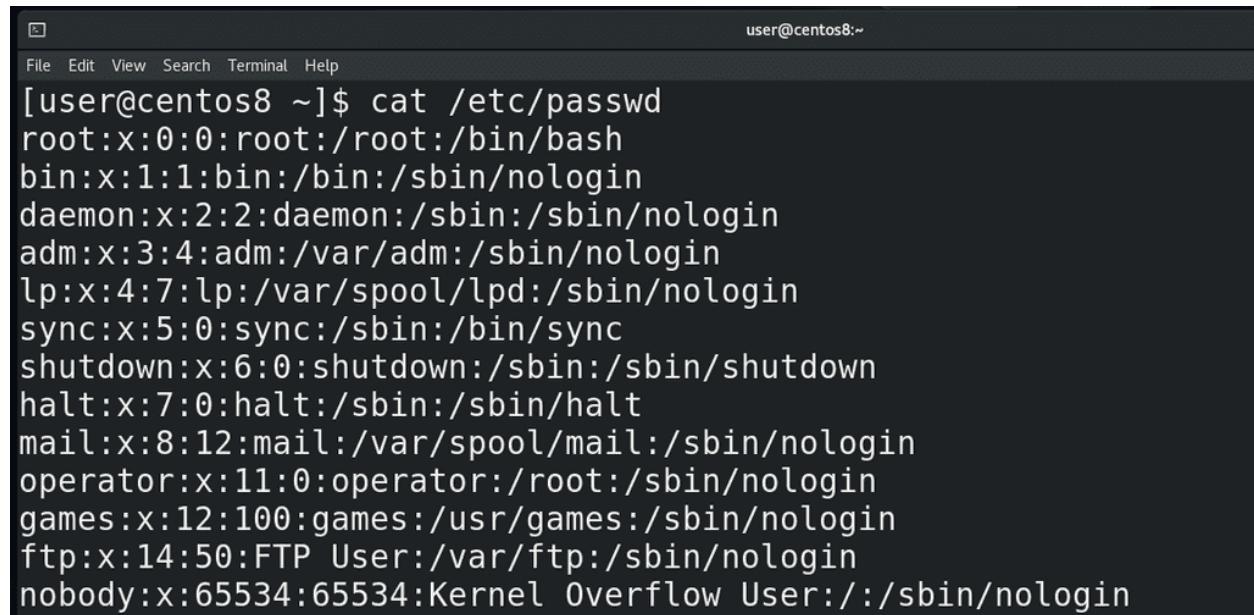
2. Кто может использовать sudo?
3. Чем отличается su user от su - user?
4. Чем отличается su от sudo su ?
5. Объясните эту запись: ALL=(ALL:ALL) ALL
6. Какими способами можно дать пользователю «права администратора», какой способ лучше и почему?

Задания

1. Сделайте пользователя user2 администратором и докажите, что это работает.
2. Разрешите пользователю user1 запускать команду touch от пользователя user2 и докажите, что это работает.

2.19 19. Пользователи

2.19.1 19. Пользователи

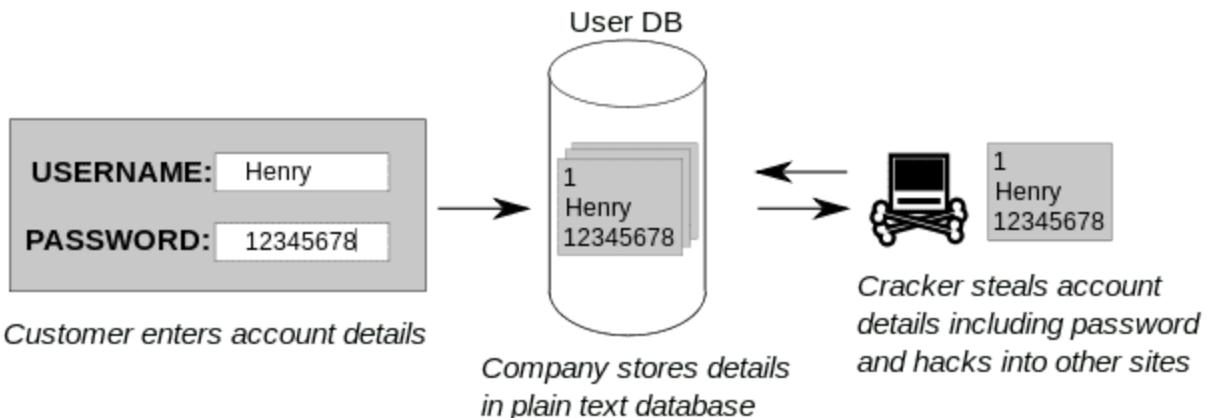


```
[user@centos8 ~]$ cat /etc/passwd
root:x:0:0:root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
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:65534:65534:Kernel Overflow User::/sbin/nologin
```

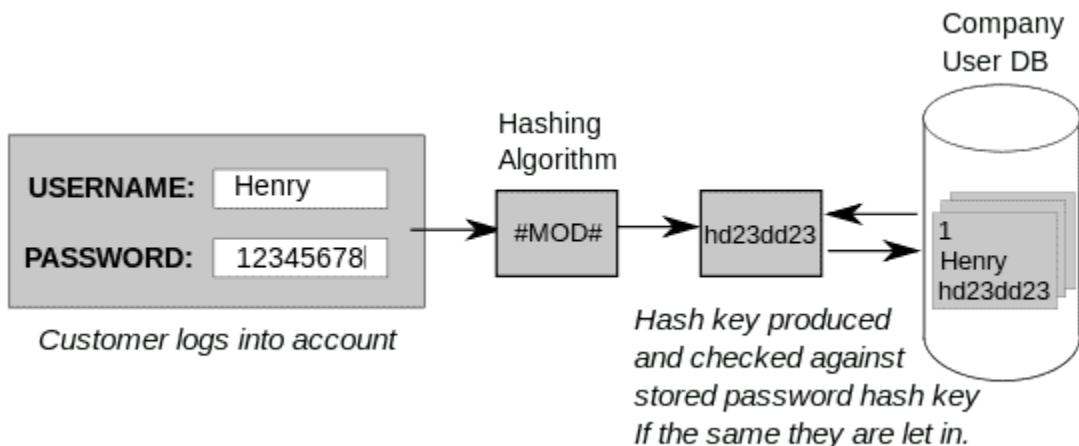
Мы с вами уже не раз обращались к файлу /etc/passwd, наконец-то пришло время разобраться, что же там написано:

```
cat /etc/passwd
```

Каждая строчка содержит информацию о каком-то пользователе. Двоеточие в данном файле выступает в роли разделителя – специального символа, который делит строку на столбцы. В первом столбце у нас логин пользователя, во втором – икс-ы. Очень давно здесь хранились пароли пользователей в хэшированном виде. Вообще, хэширование используется не только для паролей, но сейчас нас интересует именно хэширование паролей.

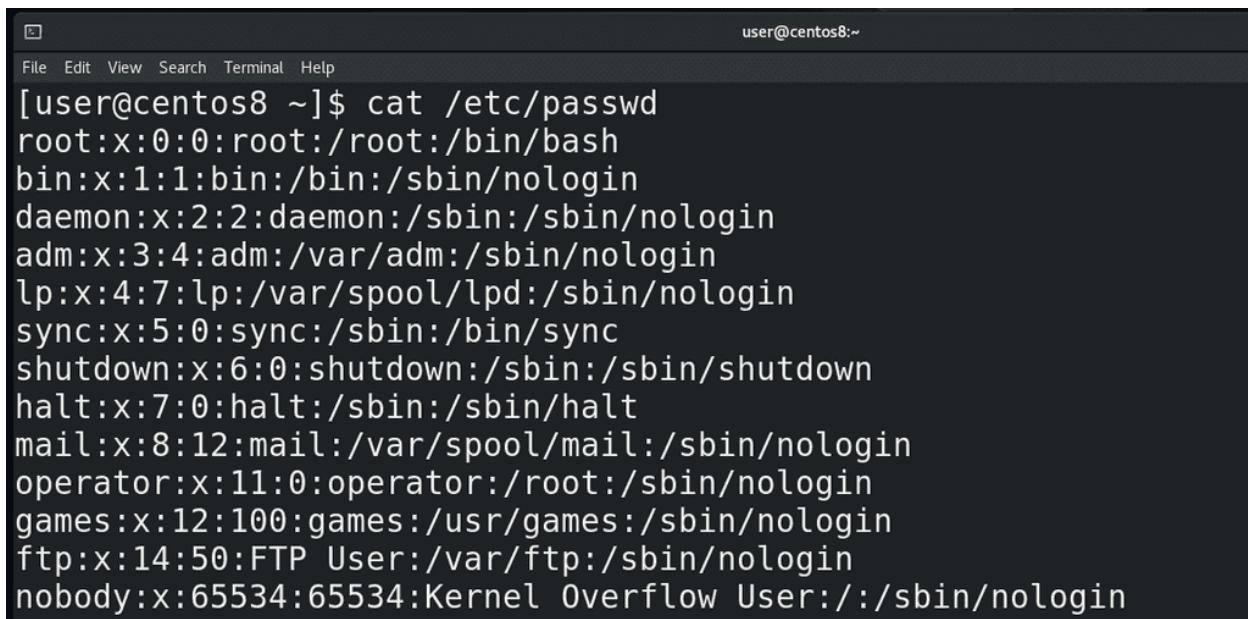


И так, что это вообще такое? Думаю все согласятся, что хранить пароли пользователя в открытом виде нельзя, иначе их с лёгкостью узнает любой пользователь с правами суперпользователя. Но система должна знать пароли, чтобы их подтвердить, когда вы логинитесь – поэтому пароли нужно преобразовать в какой-то нечитаемый вид. Но если это сделать так, чтобы можно было преобразовать обратно в читаемый вид – то опять же любой пользователь с правами рута сможет это сделать и получить первоначальный пароль.



Значит пароль нужно так преобразовать, чтобы никак нельзя было узнать первоначальный вид. И когда пользователь введёт пароль, можно будет преобразовать введённый им пароль тем же способом, получить ровно такое же значение, сравнить эти значения, и, если совпадают - пароль подходит. Но даже такой способ недостаточно безопасен – зачастую люди используют стандартные пароли и легко создать базу, где для каждого стандартного пароля будет его хэшированный вид. А потом можно будет попытаться найти в этой базе совпадающий хэш. Если пароль стандартный – это будет довольно просто. Поэтому при хэшировании к паролю ещё добавляют рандомные символы, называемые солью, благодаря чему даже два одинаковых пароля после хэширования будут выглядеть по-разному. Но это всё криптография. Я объяснил очень поверхностно, опуская много деталей, а если кому интересно, можете почитать по [ссылке](#).

Учитывая, что в passwd есть полезная информация о пользователях, скрывать этот файл от всех пользователей как-то нежелательно, но и хранить тут пароли, пусть даже в хэшированном виде, тоже как-то не правильно. Поэтому пароли из /etc/passwd перенесли в другой файл - /etc/shadow, passwd сделали читаемым для всех, а shadow доступен только для рута пользователя.



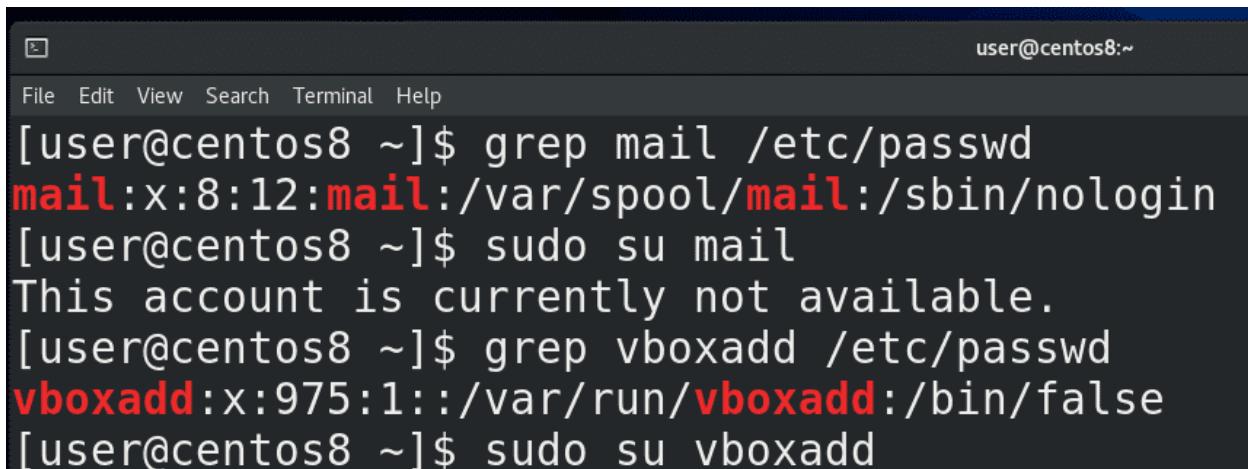
```
[user@centos8 ~]$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
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:65534:65534:Kernel Overflow User:/:/sbin/nologin
```

Пойдём дальше. Третий столбик – уникальный идентификатор пользователя – user id - uid. Очень многое строится именно вокруг uid-а, а не логина пользователя. У root пользователя идентификатор всегда ноль. Обычно для базовых сервисных пользователей uid-ы назначаются ниже 100, для каких-то дополнительных сервисных пользователей – до 999, а с 1000 начинаются uid-ы обычных пользователей.

Дальше идёт идентификатор группы - group id – gid. У каждого пользователя есть одна основная группа. И у каждой группы есть свой уникальный идентификатор. Здесь отображается идентификатор этой основной группы пользователя, для самих групп есть файл /etc/group.

После идентификатора группы идёт небольшой комментарий о пользователе – тут иногда пишут полное имя пользователя, его телефонный номер или какой-то комментарий. Как видите, для некоторых пользователей этот столбец не имеет значения.

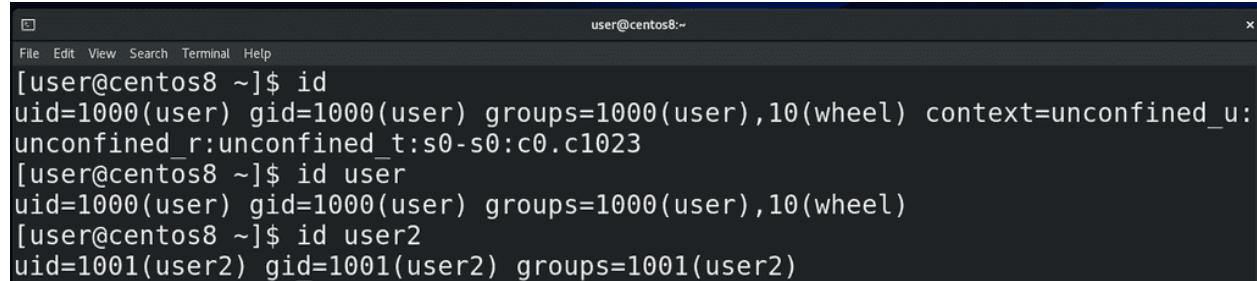
Следующий столбец – домашняя директория пользователя. Мы привыкли, что обычно домашняя директория хранится внутри директории /home , но это не обязательно и те же сервисные пользователи используют в качестве домашней директории абсолютно другие пути. У суперпользователя домашняя директория - /root.



```
[user@centos8 ~]$ grep mail /etc/passwd
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
[user@centos8 ~]$ sudo su mail
This account is currently not available.
[user@centos8 ~]$ grep vboxadd /etc/passwd
vboxadd:x:975:1::/var/run/vboxadd:/bin/false
[user@centos8 ~]$ sudo su vboxadd
```

Ну и последний столбик – shell – оболочка пользователя. Допустим, у нашего юзера или рута оболочкой выступает bash. Есть ещё другие оболочки – zsh, csh и т.п. - у всех свои преимущества. У сервисных пользователей, как правило, вместо оболочки указан nologin – и если кто-то попытается залогиниться

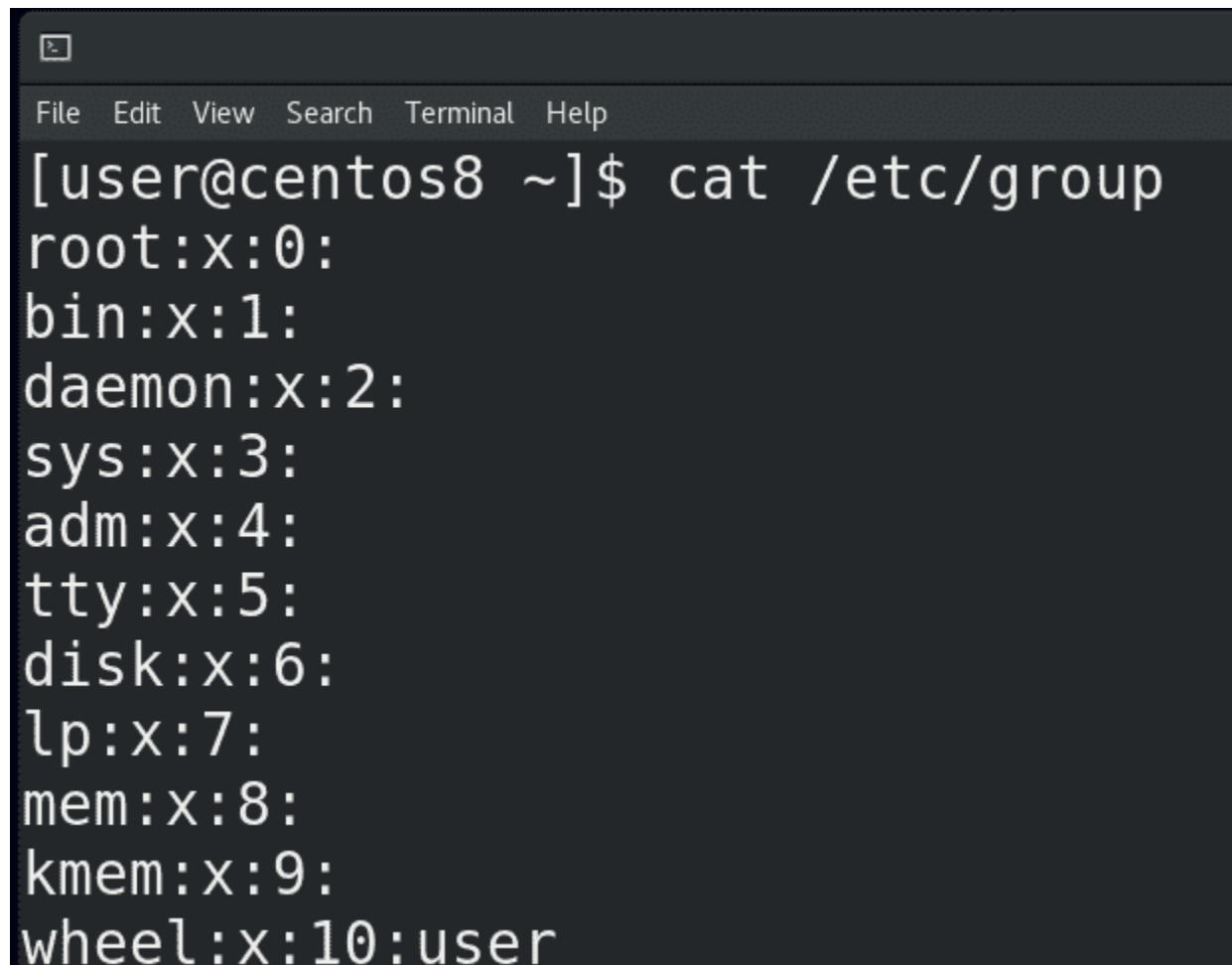
этими пользователями – увидит текст о том, что нельзя. И этот текст можно заранее прописать в файле /etc/nologin.txt. В некоторых случаях вместо оболочки можно встретить /bin/false – тоже не позволяет логинится, но работает немного по другому принципу. /bin/false – программа, которая ничего не делает и просто выдаёт ошибку – обычно нужна для каких-нибудь скриптов. И если это указать в /etc/passwd – при логине пользователя с /bin/false он просто получит ошибку и всё.



```
[user@centos8 ~]$ id
uid=1000(user) gid=1000(user) groups=1000(user),10(wheel) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[user@centos8 ~]$ id user
uid=1000(user) gid=1000(user) groups=1000(user),10(wheel)
[user@centos8 ~]$ id user2
uid=1001(user2) gid=1001(user2) groups=1001(user2)
```

Зачастую при работе может понадобиться узнать uid пользователя или группы, в которых он состоит – и не обязательно для этого искать нужные строки в /etc/passwd – можно использовать утилиту id:

```
id
id user
id user2
```



```
[user@centos8 ~]$ cat /etc/group
root:x:0:
bin:x:1:
daemon:x:2:
sys:x:3:
adm:x:4:
tty:x:5:
disk:x:6:
lp:x:7:
mem:x:8:
kmem:x:9:
wheel:x:10:user
```

Что касается групп – давайте заглянем в /etc/group. Тут синтаксис похож на passwd – тот же раз-

делитель в виде двоеточия, но меньше столбцов. Однако первый столбец – это не имя пользователя, а имя группы. Вы, возможно, заметили, что имена пользователей и групп совпадают. При создании какого-то пользователя по умолчанию создаётся группа с таким же названием – личная группа пользователя (User Private Group – UPG). Это сделано в целях безопасной, но удобной совместной работы с файлами. Станет понятнее, когда пройдём права на файлы.

Дальше у нас x – как и в passwd, речь про пароль, который хранится в хэшированном виде в файле /etc/gshadow. Как и у пользователей, у групп можно поставить пароль с помощью утилиты gpasswd

```
gpasswd
```

И потом, с помощью этого пароля, кто-то из другой группы может временно получить права этой группы, используя утилиту newgrp:

```
newgrp
```

Но это очень специфичная задача и я пока не видел реальных примеров использования.

Потом у нас идентификатор группы – gid. А в конце – список пользователей в этой группе, через запятую. Пока что у нас тут пусто, только в группе wheel есть пользователь user.

```
[user@centos8 ~]$ sudo head /etc/shadow
[sudo] password for user:
root:$6$vZHDPOJgArwLt9ZU$BviUXYv8EzzKeC6BbUvrfWDm058GcNSG.
/gFidjzVQHpf8HrFEcOoYdBZ11vaP31::0:99999:7:::
bin:*:18358:0:99999:7:::
daemon:*:18358:0:99999:7:::
adm:*:18358:0:99999:7:::
lp:*:18358:0:99999:7:::
```

Ну и давайте ещё заглянем в файл /etc/shadow:

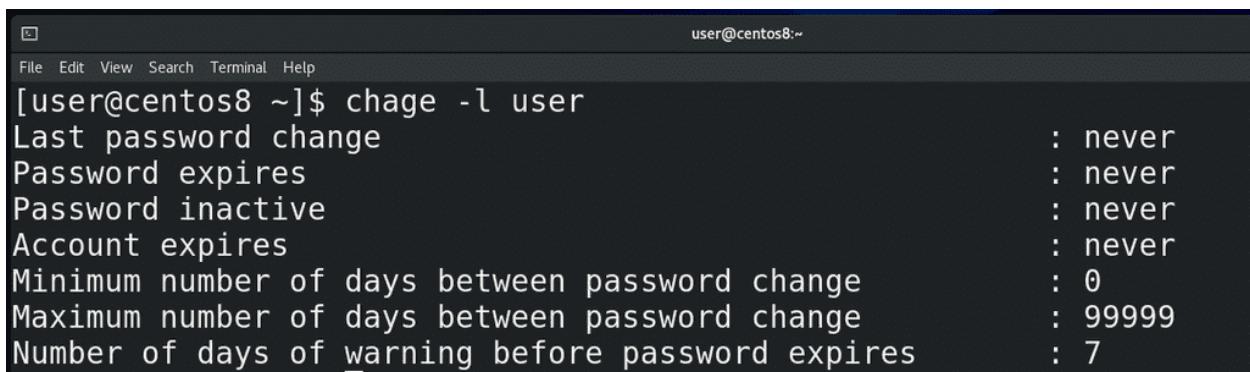
```
sudo cat /etc/shadow
```

Тут у нас хранится информация о пароле пользователя и всё, что относится к паролю:

- пользователь;
- пароль в хэшированном виде, причём в начале указывается хэш функция, например, \$6\$ - это sha512 – то есть каким алгоритмом был хэширован пароль. Также тут вместо пароля может быть * или ! или два восклицательных знака – может зависеть от дистрибутива. Обычно это означает, что аккаунт заблокирован. Как правило, это относится к сервисным или новым аккаунтам, у которых нет паролей.

- Days since epoch of last password change
- Days until change allowed
- Days before change required
- Days warning for expiration
- Days after no logins before account is locked
- Days since epoch when account expires
- Reserved and unused

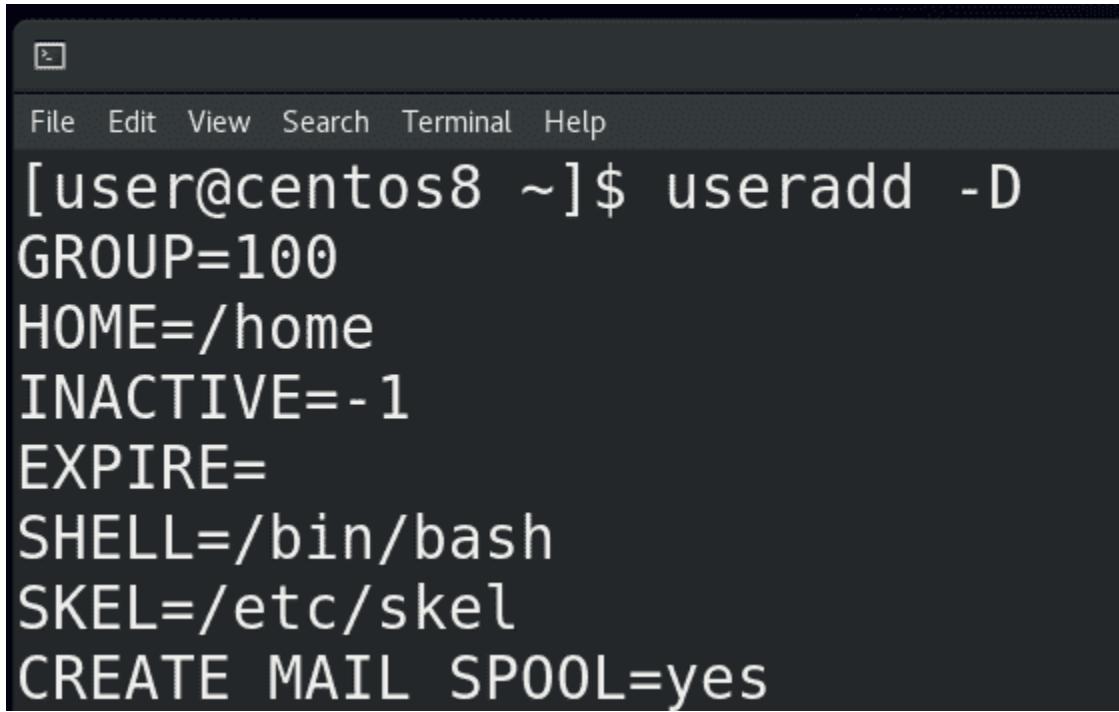
Дальше у нас идёт информация о том, когда менялся пароль, когда заблокируется и всё такое.



```
[user@centos8 ~]$ chage -l user
Last password change : never
Password expires       : never
Password inactive     : never
Account expires        : never
Minimum number of days between password change : 0
Maximum number of days between password change : 99999
Number of days of warning before password expires : 7
```

Эту информацию не очень удобно читать из файла – легче использовать утилиту chage:

```
chage -l user
```



```
[user@centos8 ~]$ useradd -D  
GROUP=100  
HOME=/home  
INACTIVE=-1  
EXPIRE=  
SHELL=/bin/bash  
SKEL=/etc/skel  
CREATE_MAIL_SPOOL=yes
```

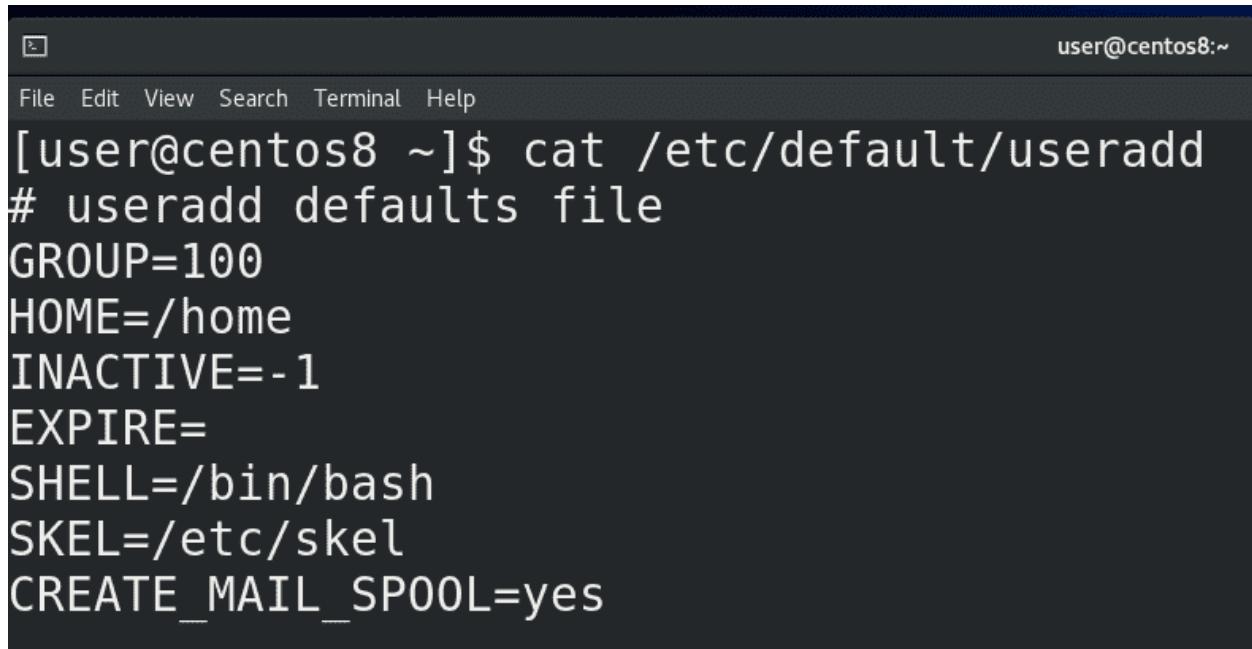
До этого мы уже создавали пользователя с помощью команды:

```
useradd user2
```

Как мы заметили, у пользователей есть много разных настроек, а значит useradd откуда-то взяла настройки по умолчанию. Настройки по умолчанию можно увидеть при помощи ключа -D:

```
useradd -D
```

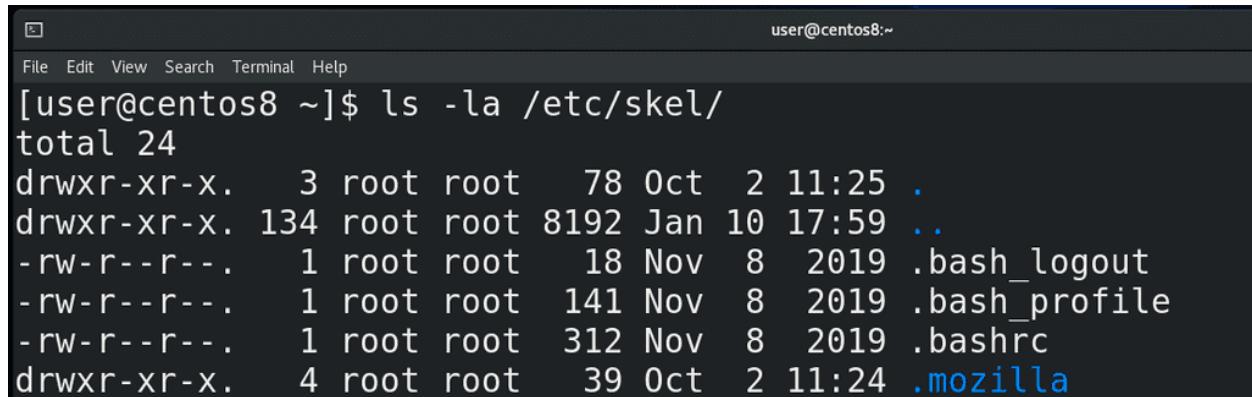
Сразу скажу, что первый параметр – GROUP – будет игнорироваться, для каждого пользователя создаётся его личная группа. Настройки по умолчанию распределены в двух файлах – /etc/default/useradd и /etc/login.defs. Если первый файл – сугубо параметры утилиты useradd, то login.defs содержит параметры для многих утилит, работающих с пользователями и группами.



```
[user@centos8 ~]$ cat /etc/default/useradd
# useradd defaults file
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
CREATE_MAIL_SPOOL=yes
```

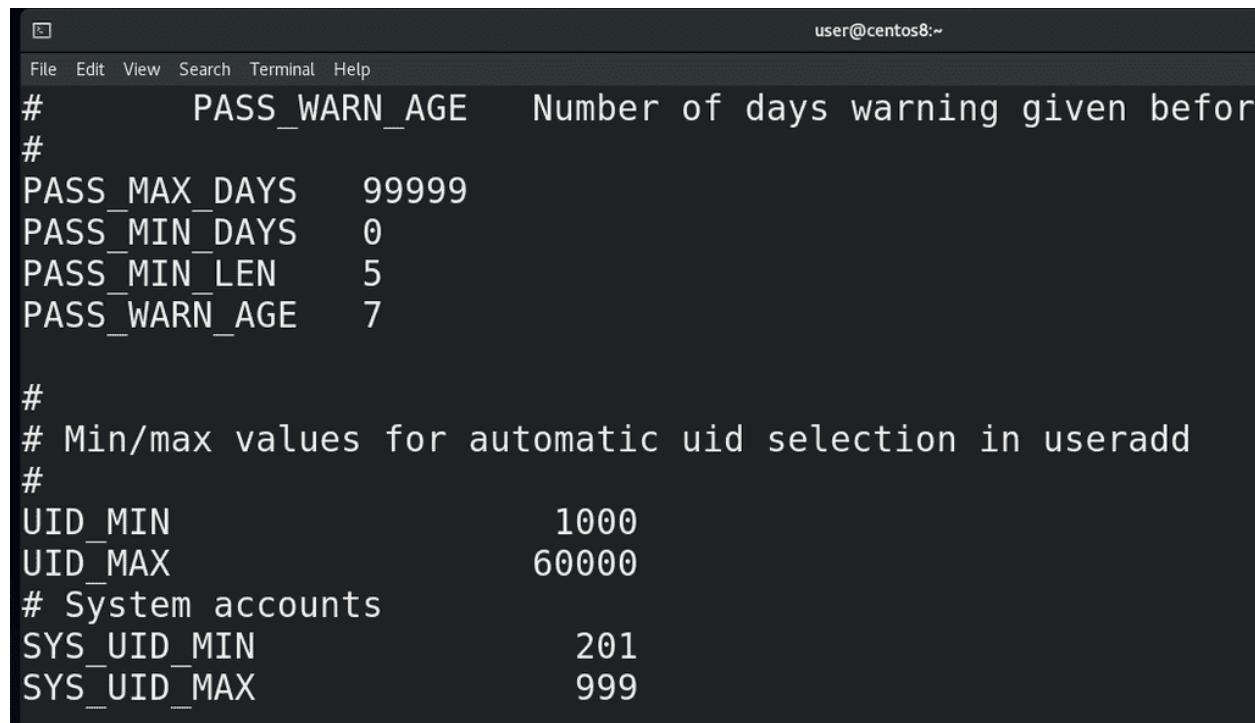
И так, в файле /etc/default/useradd у нас несколько параметров:

- GROUP – если мы не захотим создавать личную группу пользователя, то группа по умолчанию будет группа с gid 100 – это группа users;
- HOME – это внутри какой директории создаётся домашняя директория пользователя. Т.е мы создаём пользователя user2 и для него создаётся директория user2 внутри директории /home.
- INACTIVE – это через сколько дней после устаревания пароля заблокируется аккаунт: -1 – никогда, 0 – сразу же, как устареет пароль, ну или указываете количество дней.
- EXPIRE – когда аккаунт заблокируется. Указывается как год, месяц, день (ГГГГ-ММ-ДД).
- SHELL – какой интерпретатор будет по умолчанию, в данном случае /bin/bash;



```
[user@centos8 ~]$ ls -la /etc/skel/
total 24
drwxr-xr-x.  3 root root 78 Oct  2 11:25 .
drwxr-xr-x. 134 root root 8192 Jan 10 17:59 ..
-rw-r--r--.  1 root root 18 Nov  8 2019 .bash_logout
-rw-r--r--.  1 root root 141 Nov  8 2019 .bash_profile
-rw-r--r--.  1 root root 312 Nov  8 2019 .bashrc
drwxr-xr-x.  4 root root 39 Oct  2 11:24 .mozilla
```

- SKEL – путь к шаблонной директории, которая используется при создании пользователя. Тут у нас есть .bash_profile и .bashrc. Если вы хотите, чтобы у всех новых пользователей в домашней директории были какие-то файлы или директории, достаточно положить их в /etc/skel.
- CREATE_MAIL_SPOOL – создаёт специальный файл, куда будет попадать входящая почта для пользователя.



```

user@centos8:~$ cat /etc/login.defs
#           PASS_WARN_AGE      Number of days warning given before
#                                changing password
#
PASS_MAX_DAYS     99999
PASS_MIN_DAYS     0
PASS_MIN_LEN       5
PASS_WARN_AGE      7

#
# Min/max values for automatic uid selection in useradd
#
UID_MIN             1000
UID_MAX            60000
# System accounts
SYS_UID_MIN         201
SYS_UID_MAX         999

```

Теперь что касается /etc/login.defs:

```
cat /etc/login.defs
```

- MAIL_DIR – директория, где создаётся файл для входящей почты. Вообще тут есть разные варианты, но давайте пока не будем трогать почту.
- PASS_MAX_DAYS – максимум дней, разрешённых на один пароль. Скажем, если поставить 30 – нужно будет менять пароль каждый месяц.
- PASS_MIN_DAYS – минимум дней, необходимых для смены пароля. Допустим, если поставить 7 – то можно будет менять пароль максимум раз в неделю. Это нужно, если вы хотите защититься от того, чтобы ваши пользователи повторно не использовали старый пароль. Допустим, у вас может стоять политика, чтобы у пользователя пароли не совпадали как минимум с 10 предыдущими паролями. Без минимального времени смены пароля он может просто разом 10 раз ввести новые пароли и потом старый. Так что этот параметр защищает от таких любителей одного пароля.
- PASS_MIN_LEN – минимальная длина пароля. Естественно, руку плевать на этот параметр, а вот юзеры должны будут придумать пароль указанной длины.
- PASS_WARN_AGE – за сколько дней до устаревания пароля пользователю выйдет предупреждение о том, что ему стоит сменить пароль.
- UID_MIN и UID_MAX – минимальный и максимальный uid, который будет выдан пользователю, если конечно вручную не указать другой uid. Максимальное значение примерно 65000.
- SYS_UID_MIN и SYS_UID_MAX – uid-ы для сервисных пользователей.
- CREATE_HOME – создавать ли домашнюю директорию при создании пользователя.
- USERGROUPS_ENAB – тот самый параметр, отвечающий за создание приватной группы пользователя. Без этого параметра группа по умолчанию будет та, что указана в файле /etc/default/useradd.

- ENCRYPT_METHOD – SHA512 – алгоритм, по которому хэшируются пароли для /etc/shadow.
На самом деле для login.defs есть много других параметров, но пока что нам этого достаточно.

```
[user@centos8 ~]$ sudo useradd user3
[sudo] password for user:
```

Теперь, зная всё это, давайте рассмотрим утилиту useradd. При простом добавлении:

```
useradd user3
```

всё будет ровно с теми параметрами, которые мы рассматривали в файлах useradd и login.defs. Если же мы хотим сделать как-то по своему, то давайте я возьму пару параметров для примера, а остальное вы сами протестируйте.

```
[user@centos8 ~]$ sudo mkdir -p /home/company/it
[user@centos8 ~]$ sudo useradd user4 -b /home/company/it
[cut]
```

И так, sudo useradd и ключ -b – base dir – это собственно директория, внутри которой создаётся домашняя директория пользователя, как параметр HOME в useradd. Допустим, если я указжу:

```
sudo useradd user4 -b /home/company/it
```

то внутри этой директории /home/company/it создаётся директория user4. Но нужно заранее создать эту директорию /home/company/it:

```
mkdir -p /home/company/it
```

Если у меня уже есть какая-то директория для пользователя и я не хочу её создавать, я могу указать её с ключом -d:

```
sudo useradd -d /home/olduser user4
```

Ключ -c – для комментария. Ключ -g основная группа пользователя. Как мы говорили, если этот ключ не указывать, то создаётся приватная группа пользователя и она станет основной группой этого пользователя. Если же мы хотим существующую группу – то указываем после ключа -g:

```
sudo useradd user4 -g groupname
```

Ключ -G большое – для дополнительных групп. Допустим, если вы хотите, чтобы пользователь кроме основной группы был также в группах wheel и users2:

```
sudo useradd user4 -g users -G wheel,users2
```

Вы можете сами задать uid для будущего пользователя:

```
sudo useradd user4 -u 1111
```

```
[user@centos8 ~]$ grep user4 /etc/passwd /etc/group  
/etc/passwd:user4:x:1111:100:User Userovich:/home/company/it/user4:/bin/bash  
/etc/group:wheel:x:10:user,user4  
/etc/group:user2:x:1001:user4  
[user@centos8 ~]$ id user4  
uid=1111(user4) gid=100(users) groups=100(users),10(wheel),1001(user2)
```

Теперь посмотрим, что у нас получилось:

```
sudo useradd user4 -b /home/user/it -c "User Userovich" -g users -G wheel,user2  
grep user4 /etc/passwd /etc/group  
id user4
```

Как видим, всё так, как мы указывали. Но пока этого не достаточно – пока мы не зададим пароль пользователю, аккаунт будет недоступен.

```
[user@centos8 ~]$ sudo passwd user4  
Changing password for user user4.  
New password:  
BAD PASSWORD: The password is a palindrome  
Retype new password:  
passwd: all authentication tokens updated successfully.
```

Для создания пароля используем команду passwd:

```
sudo passwd user4
```

Можете также использовать утилиту chage, чтобы настроить времена для пароля:

```
chage user4
```

```
[user@centos8 ~]$ sudo usermod user4 -d /var/user4 -m -aG user  
[user@centos8 ~]$ grep user4 /etc/passwd /etc/group  
/etc/passwd:user4:x:1111:100:User Userovich:/var/user4:/bin/bash  
/etc/group:wheel:x:10:user,user4  
/etc/group:user:x:1000:user4  
/etc/group:user2:x:1001:user4
```

Если вы уже создали пользователя, но хотите изменить какие-то параметры – допустим, поменять комментарий, добавить в группу, переместить домашнюю директорию, поменять uid и т.п. – используйте утилиту usermod. Например, я хочу перенести домашнюю директорию пользователя и добавить его в группу:

```
sudo usermod user4 -d /var/user4 -m -aG user
```

Ключ -d указывает на новую домашнюю директорию, но без ключа -m текущая домашняя директория не перенесётся на новое место. Что касается -aG, то G указывает дополнительные группы, но без ключа -a все текущие группы пользователя сбросятся и останется одна группа user.

```
[user@centos8 ~]$ sudo userdel -r user2
userdel: group user2 not removed because it has other members.
```

Чтобы удалить пользователя, используется команда userdel. Но без ключа -r после удаления пользователя останется его домашняя директория, личная группа и почтовый ящик, а с ключом всё это удалится:

```
sudo userdel -r user2
```

```
[user@centos8 ~]$ sudo groupadd group1
[user@centos8 ~]$ sudo gpasswd group1 -A user -M user4,root
[user@centos8 ~]$ gpasswd -a user group1
Adding user user to group group1
[user@centos8 ~]$ gpasswd -d root group1
Removing user root from group group1
```

Что касается групп – всё примерно также – команды:

```
groupadd
groupmod
groupdel
```

Для примера, давайте добавим группу group1:

```
sudo groupadd group1
```

У группы можно назначить администратора и пользователей с помощью команды gpasswd:

```
sudo gpasswd group1 -A user -M user4,root
```

А администратор группы может добавлять и удалять пользователей из группы уже без всяких прав суперпользователя:

```
gpasswd -a user group1
gpasswd -d root group1
```

```
[user@centos8 ~]$ sudo lid user
wheel(gid=10)
user(gid=1000)
group1(gid=1003)
[user@centos8 ~]$ sudo lid -g group1
user4(uid=1111)
user(uid=1000)
```

Чтобы посмотреть, какие группы у пользователя и какие пользователи в группе, можно использовать команду `lid`:

```
sudo lid user
sudo lid -g group1
```

```
Usage: useradd [options] LOGIN
        useradd -D
        useradd -D [options]

Options:
  -b, --base-dir BASE_DIR      base directory for the home directory of the
                                new account
  -c, --comment COMMENT        GECOS field of the new account
  -d, --home-dir HOME_DIR     home directory of the new account
  -D, --defaults               print or change default useradd configuration
  -e, --expiredate EXPIRE_DATE expiration date of the new account
  -f, --inactive INACTIVE      password inactivity period of the new account
  -g, --gid GROUP              name or ID of the primary group of the new
                                account
  -G, --groups GROUPS         list of supplementary groups of the new
                                account
```

Мы много чего разобрали – файлы `/etc/passwd` и `/etc/group`, где хранится информация о пользователях и группах, `/etc/shadow`, где хранится информация о паролях, файлы `/etc/default/useradd` и `/etc/login.defs`, где прописаны параметры для новых пользователей, а также различные утилиты для создания новых пользователей и групп, изменения их параметров, паролей и т.п. И хотя мы не стали задерживаться на различных ключах – для вас это практика – создавайте пользователей и группы с различными параметрами, если что не понятно – откройте маны – там многое объясняется. А если какие-то трудности – обращайтесь, вместе разберёмся.

2.19.2 Практика

Вопросы

1. Возьмите любую строчку из `/etc/passwd` и «расшифруйте», что написано.
2. В каких файлах хранится информация о пользователях и группах?

Задания

- Создайте директорию /home/it. Создайте группу ИТ с идентификатором 10000. Создайте 3 пользователей - sysadmin, helpdesk и netadmin, домашние директории которых будут находиться в /home/it, а также они будут входить в группу ИТ и оболочкой у всех будет /bin/bash. Идентификаторы должны быть 10101, 10201, 10301.
- Предоставьте пользователю sysadmin все права на систему. Пользователь netadmin должен иметь возможность запускать команду nmtui от имени root-a. А пользователь helpdesk должен иметь возможность менять пароли всех пользователей, кроме sysadmin и root. *
- Создайте три группы – marketing, sales и it. Для каждой группы создайте директорию внутри /home, а также по два пользователя - user.marketing, manager.marketing, user.sales, manager.sales, admin и helpdesk. Группы для пользователей должны быть основными. manager-ы должны иметь возможность залогиниться, а юзеры нет. У manager-ов должна быть своя домашняя директория, а для user-ов домашней директорией должна выступать директория группы.

2.20 20. Права на файлы

2.20.1 20. Права на файлы

```
[user@centos8 ~]$ mkdir temp
[user@centos8 ~]$ cd temp/
[user@centos8 temp]$ touch file
[user@centos8 temp]$ stat file
  File: file
  Size: 0          Blocks: 0          IO Block: 4096   regular empty file
Device: fd00h/64768d  Inode: 34699313      Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/    user)    Gid: ( 1000/    user)
Context: unconfined_u:object_r:user_home_t:s0
Access: 2021-01-10 22:08:10.455754106 +0400
Modify: 2021-01-10 22:08:10.455754106 +0400
Change: 2021-01-10 22:08:10.455754106 +0400
 Birth: -
```

В теме «О файловых системах» мы познакомились с таким понятием, как инода – в ней хранится всякая информация о файле. С помощью утилиты stat мы можем увидеть часть этой информации и сейчас нас интересует 4 строки, где указаны стандартные UNIX права на этот файл:

```
mkdir temp
cd temp
touch file
stat file
```

владелец файла – user - u

группа с правами на файл – group - g

остальные – others - o

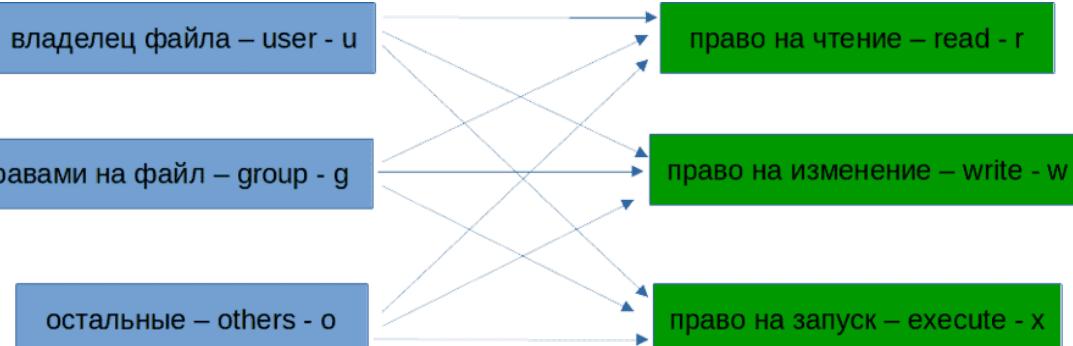
Доступы пользователей к файлам можно разделить на 3 категории: либо пользователь является владельцем этого файла и для него есть обозначение u - user; либо пользователь относится к группе, для которой выделены какие-то права на этот файл – обозначение для них – g - group; либо он относится к остальным – все остальные пользователи, которые не подходят под первую и вторую категорию – для них обозначение o - others. Конечно, есть ещё root – он может делать всё что угодно.

право на чтение – read - r

право на изменение – write - w

право на запуск – execute - x

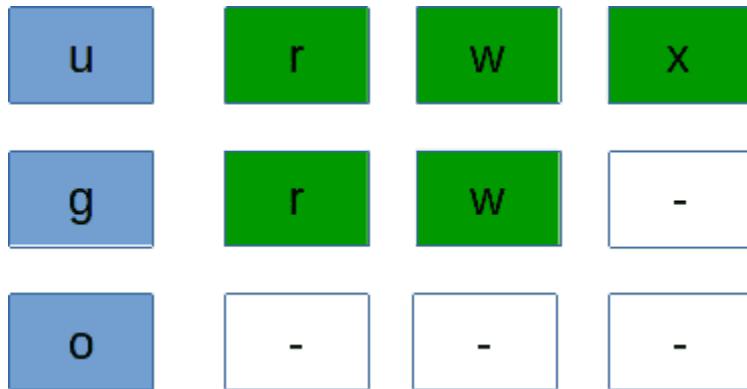
Что касается самих прав, то они делятся на три типа: право на чтение файла – обозначается как r – read; право на изменение файла – обозначается как w – write; право на запуск файла - обозначается как x – execute. Право на запуск обычно относится к программам и скриптам.



Теперь, указывая для владельца, группы и остальных права на файл, мы получаем такое значение:

```
rwxrwxrwx
```

Первая тройка rwx это для владельца, вторая для группы, третья для остальных. Такое обозначение говорит, что мы разрешаем и владельцу, и группе и остальным читать файл, изменять его и запускать.



Если же мы хотим владельцу разрешить всё, группе разрешить только читать и изменять, а остальным ничего, получаем:

```
rwxrw----
```

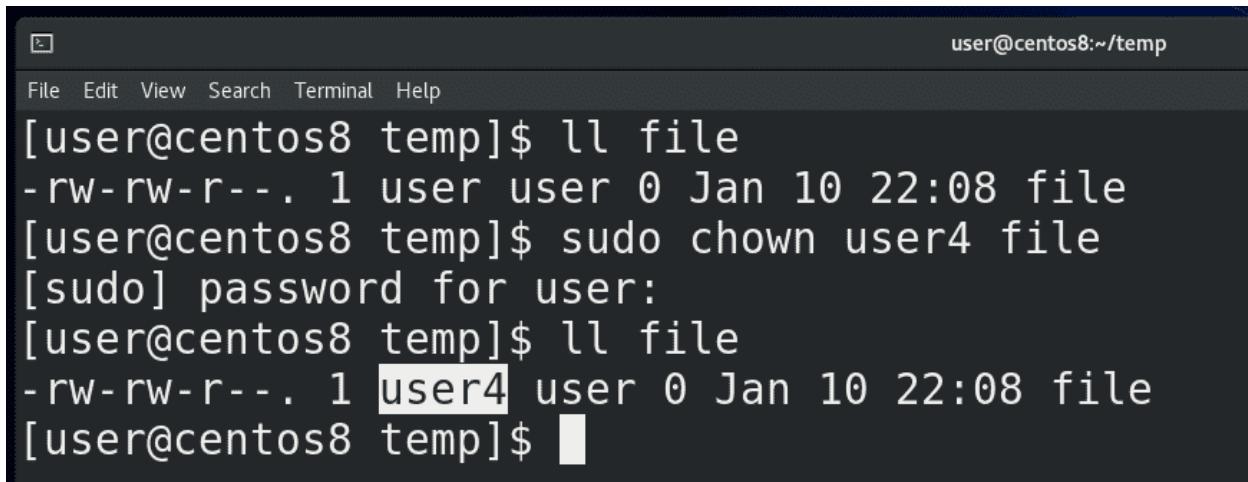
где отсутствующие права заменены на дефисы.

```
[user@centos8 temp]$ ls -l ~
total 20
drwxr-xr-x. 2 user user 6 Oct 2 11:58 Desktop
drwxrwxr-x. 2 user user 6 Nov 23 20:09 dir1
drwxrwxr-x. 2 user user 6 Nov 23 20:09 dir3
drwxr-xr-x. 2 user user 6 Oct 2 11:58 Documents
drwxr-xr-x. 2 user user 6 Oct 2 11:58 Downloads
-rw-rw-r--. 1 user user 2561 Nov 23 20:50 errors
-rw-rw-r--. 1 user user 15 Jan 1 19:45 file
-rw-rw-r--. 1 user user 580 Nov 23 20:43 filelist
drwxr-xr-x. 2 user user 6 Oct 2 11:58 Music
-rw-rw-r--. 1 user user 3490 Nov 23 20:58 output
drwxr-xr-x. 2 user user 4096 Jan 10 22:08 Pictures
```

Кроме stat, можно использовать ls -l или его аlias ll:

```
ll ~
```

где в первом столбце также отображаются права на файлы, в 3 отображается владелец, а в четвёртом группа.

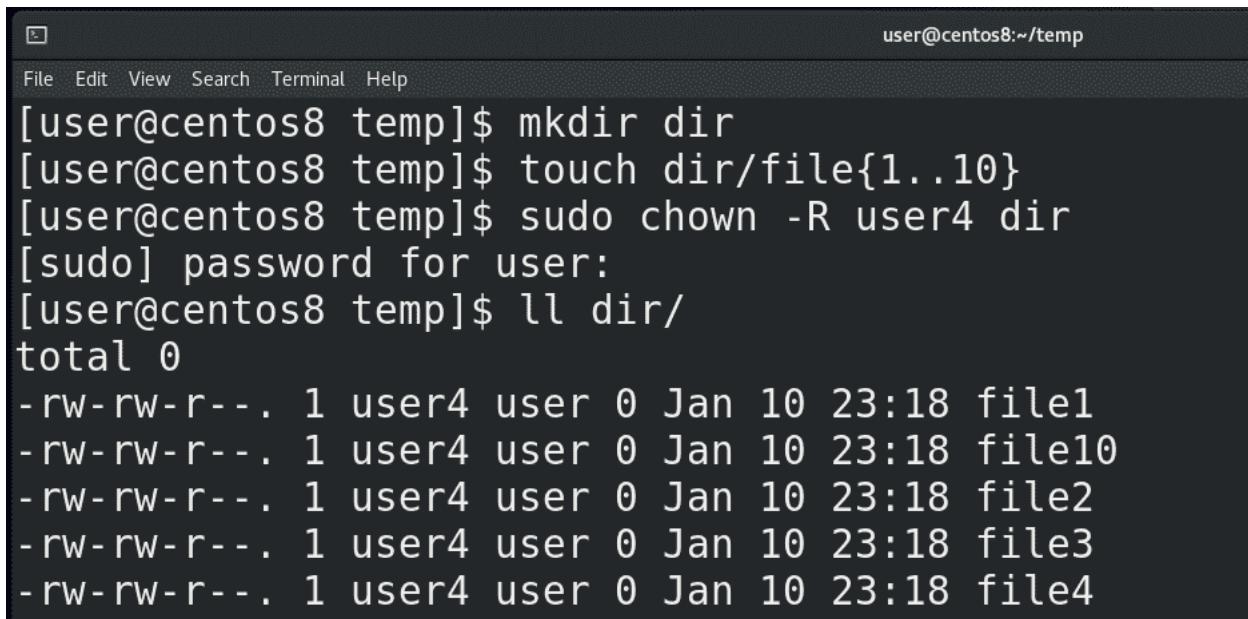


The screenshot shows a terminal window with the title bar "user@centos8:~/temp". The command history is as follows:

```
[user@centos8 temp]$ ll file
-rw-rw-r--. 1 user user 0 Jan 10 22:08 file
[user@centos8 temp]$ sudo chown user4 file
[sudo] password for user:
[user@centos8 temp]$ ll file
-rw-rw-r--. 1 user4 user 0 Jan 10 22:08 file
[user@centos8 temp]$ █
```

Теперь, как это всё менять. Начнём с владельца. Чтобы поменять владельца нужна команда `chown` – `change owner`. Но даже если мы владелец этого файла, без прав суперпользователя мы это сделать не сможем. Потому что если какой-то пользователь создаст файл, а потом укажет, что владельцем файла является другой пользователь, то он сможет подставить другого пользователя. Или, например, использовать выделенное для второго пользователя место в своих целях, когда каждому пользователю выделено сколько-то места на диске. Поэтому, чтобы поменять владельца, нужно использовать `sudo`:

```
ll file
sudo chown user4 file
ll file
```

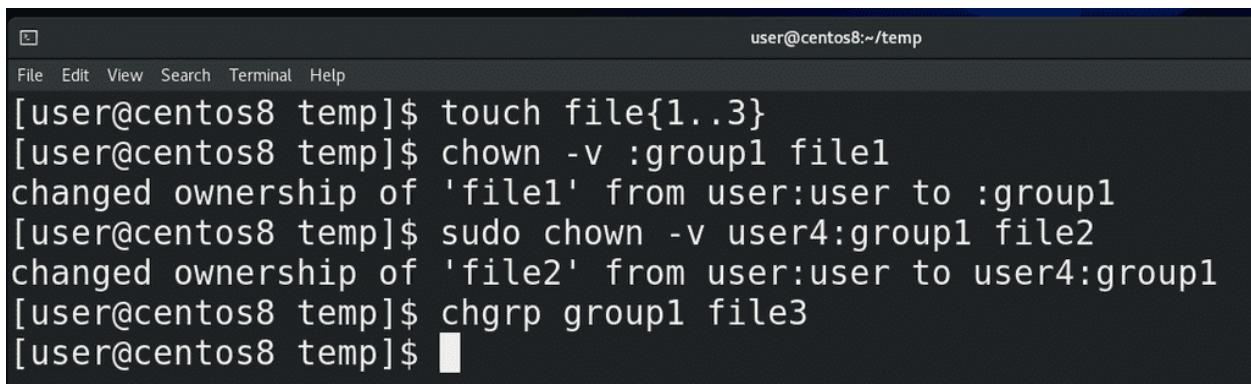


The screenshot shows a terminal window with the title bar "user@centos8:~/temp". The command history is as follows:

```
[user@centos8 temp]$ mkdir dir
[user@centos8 temp]$ touch dir/file{1..10}
[user@centos8 temp]$ sudo chown -R user4 dir
[sudo] password for user:
[user@centos8 temp]$ ll dir/
total 0
-rw-rw-r--. 1 user4 user 0 Jan 10 23:18 file1
-rw-rw-r--. 1 user4 user 0 Jan 10 23:18 file10
-rw-rw-r--. 1 user4 user 0 Jan 10 23:18 file2
-rw-rw-r--. 1 user4 user 0 Jan 10 23:18 file3
-rw-rw-r--. 1 user4 user 0 Jan 10 23:18 file4
```

Если мы хотим это сделать для всех файлов в директории и во всех поддиректориях, вы уже знаете, для этого нужно делать рекурсивно:

```
mkdir dir
touch dir/file{1..10}
sudo chown -R user4 dir
ll dir
```



```
user@centos8:~/temp
File Edit View Search Terminal Help
[user@centos8 temp]$ touch file{1..3}
[user@centos8 temp]$ chown -v :group1 file1
changed ownership of 'file1' from user:user to :group1
[user@centos8 temp]$ sudo chown -v user4:group1 file2
changed ownership of 'file2' from user:user to user4:group1
[user@centos8 temp]$ chgrp group1 file3
[user@centos8 temp]$ █
```

Группу также можно поменять с помощью chown:

```
touch file{1..3}
chown -v :group1 file1
```

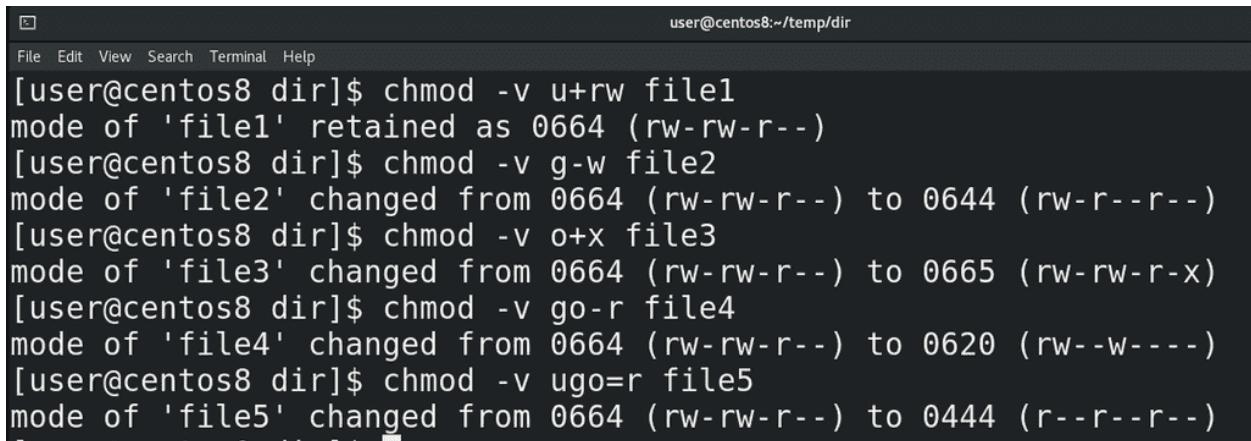
Здесь ключ -v - verbose - только для вывода информации. Можно сменить разом владельца и группу:

```
sudo chown -v user:user file2
```

А вообще, чтобы менять группу, есть отдельная команда:

```
chgrp group1 file3
```

Как видите, чтобы менять группу, не нужны права root-а, конечно, если вы владелец файла.



```
user@centos8:~/temp/dir
File Edit View Search Terminal Help
[user@centos8 dir]$ chmod -v u+rw file1
mode of 'file1' retained as 0664 (rw-rw-r--)
[user@centos8 dir]$ chmod -v g-w file2
mode of 'file2' changed from 0664 (rw-rw-r--) to 0644 (rw-r--r--)
[user@centos8 dir]$ chmod -v o+x file3
mode of 'file3' changed from 0664 (rw-rw-r--) to 0665 (rw-rw-r-x)
[user@centos8 dir]$ chmod -v go-r file4
mode of 'file4' changed from 0664 (rw-rw-r--) to 0620 (rw--w----)
[user@centos8 dir]$ chmod -v ugo=r file5
mode of 'file5' changed from 0664 (rw-rw-r--) to 0444 (r--r--r--)
```

Теперь касательно самих прав. Для начала вернём все файлы нашему пользователю, они нам пригодятся:

```
sudo chown -R user dir
cd dir
```

Для смены прав используется команда chmod. Допустим, если мы хотим владельцу разрешить читать и изменять файл, пишем:

```
chmod -v u+rw file1
```

Хотим группе запретить изменять файл, пишем:

```
chmod -v g-w file2
```

Хотим всем разрешить запускать файл, пишем:

```
chmod +x file3
```

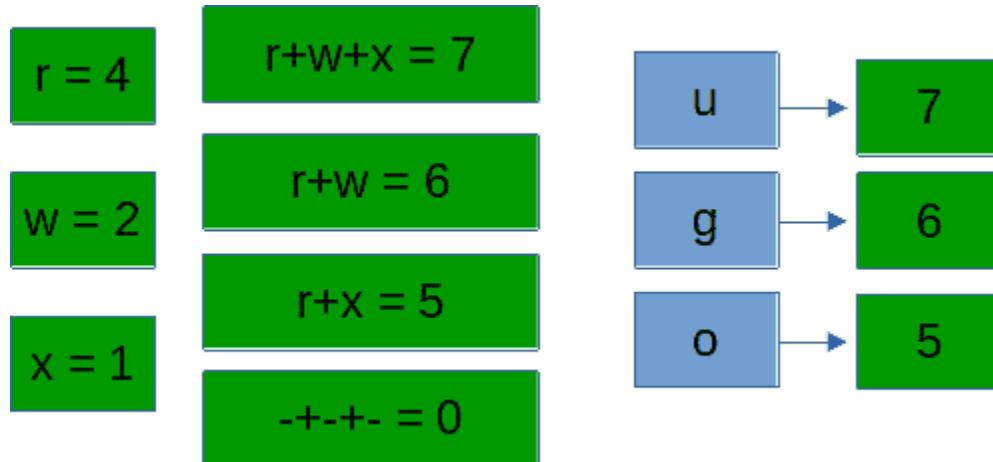
Группе и остальным запретить читать файл:

```
chmod -v go-r file4
```

+ добавляет, - убирает. Также можно использовать =, чтобы выставить определённые права, допустим:

```
chmod -v ugo=r file5
```

тогда останутся только права на чтение. chmod тоже работает рекурсивно с ключом -R. В целом по буквам понятно и это несложный способ, но он больше про изменение – кому-то что-то добавить, у кого-то что-то отнять. И если права для всех отличаются – придётся вводить две-три команды, чтобы выставить нужные права.



Часто легче использовать цифровой способ. У каждого права есть своя цифра – у read это 4, у write это 2, у execute это 1. И используя сумму этих чисел можно задать права разом. Допустим, rwx – это 4+2+1 = 7, rwxrwxrwx – это 777, rwxr-x— - это 750, rwxrw-r- - это 754. Так и пишем:

```
chmod 777 file6  
chmod 750 file6  
chmod 754 file6
```

```

user@centos8:~/temp/dir
File Edit View Search Terminal Help
[user@centos8 dir]$ mkdir dir{1..3}
[user@centos8 dir]$ touch dir{1..3}/file
[user@centos8 dir]$ chmod 400 dir1
[user@centos8 dir]$ chmod 600 dir2
[user@centos8 dir]$ chmod 700 dir3
[user@centos8 dir]$ ll dir*
dir1:
ls: cannot access 'dir1/file': Permission denied
total 0
-????????? ? ? ? ? ? file

dir2:
ls: cannot access 'dir2/file': Permission denied
total 0
-????????? ? ? ? ? ? file

dir3:
total 0
-rw-rw-r--. 1 user user 0 Jan 10 23:57 file

```

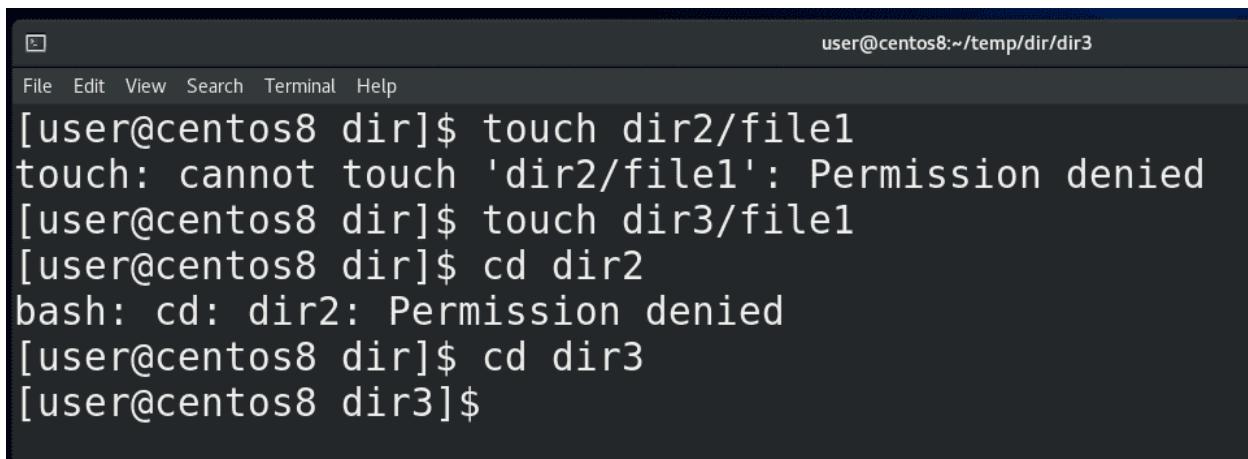
Если же говорить о директориях, вспомните, что директория как лист, в котором указаны имена файлов и их иноды — т.е. жёсткие ссылки:

```

mkdir dir{1..3}
touch dir{1..3}/file
chmod 400 dir1
chmod 600 dir2
chmod 700 dir3
ll dir*

```

read позволяет видеть только имена файлов — список файлов и директорий — как в dir1.



```
[user@centos8 dir]$ touch dir2/file1
touch: cannot touch 'dir2/file1': Permission denied
[user@centos8 dir]$ touch dir3/file1
[user@centos8 dir]$ cd dir2
bash: cd: dir2: Permission denied
[user@centos8 dir]$ cd dir3
[user@centos8 dir3]$
```

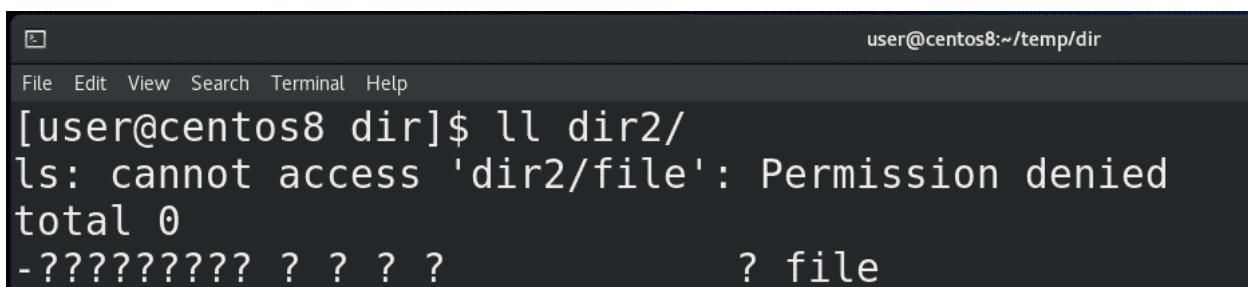
write позволяет изменять её содержимое – добавлять, переименовывать или удалять файлы – опять же, на самом деле это даже не файлы, а ёжёсткие ссылки – т.е. «записи на листе». Это как добавлять какие-то записи в лист или стирать их, при этом, не важно, есть у вас права на сами эти файлы или нет:

```
touch dir2/file1
touch dir3/file1
```

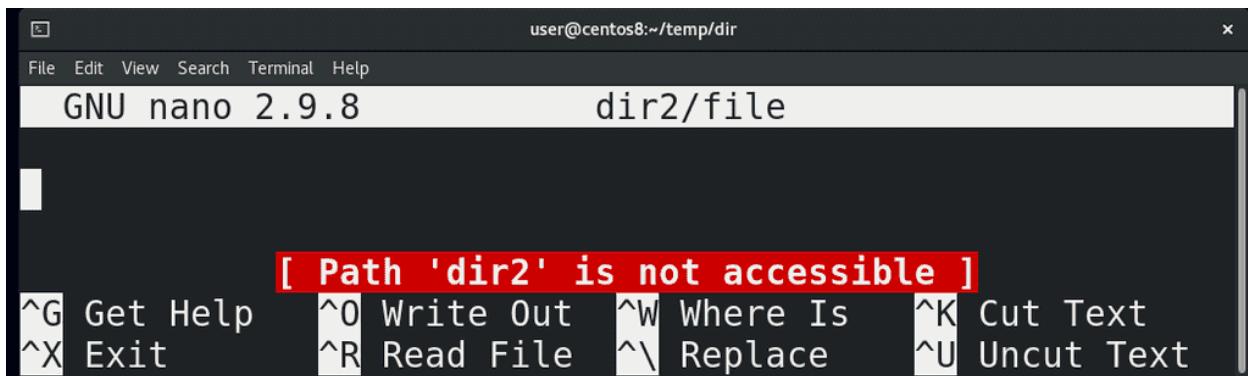
Как вы видите, несмотря на то, что write позволяет изменять содержимое директории, это не работает без execute. А execute позволяет заходить в эту директорию с помощью cd:

```
cd dir2
cd dir3
```

и выполнять операции с содержимым директории. Т.е. без execute write бесполезен.



```
[user@centos8 dir]$ ll dir2/
ls: cannot access 'dir2/file': Permission denied
total 0
-????????? ? ? ? ? ? ? file
```



```
GNU nano 2.9.8          dir2/file

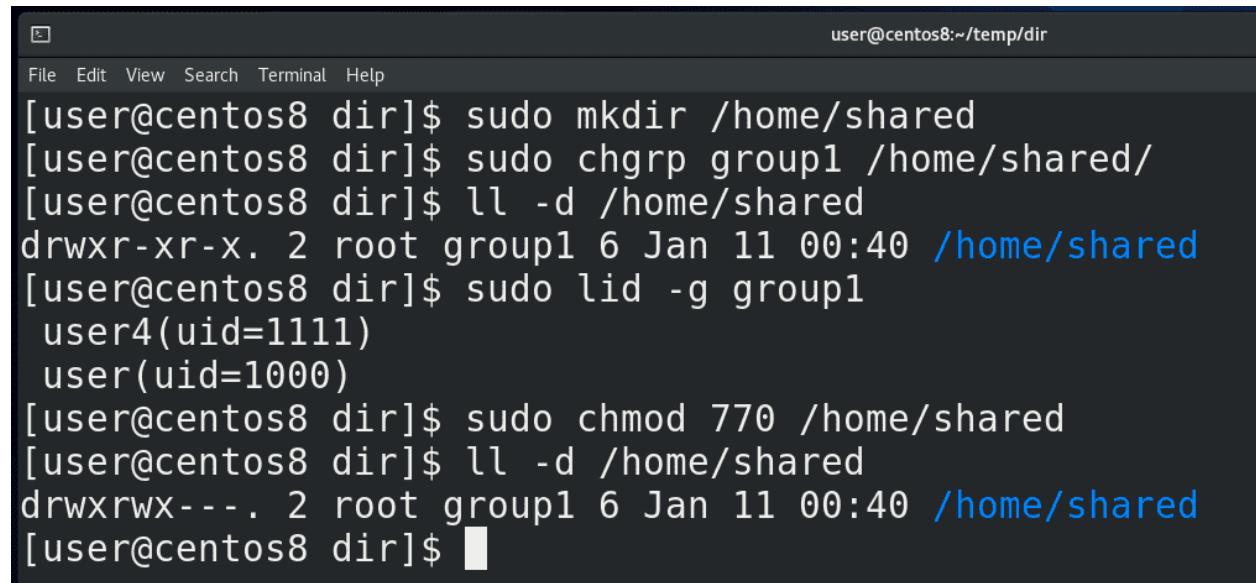
[ Path 'dir2' is not accessible ]

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text
```

При этом без execute вы не сможете работать с файлами внутри этой директории, даже если у вас есть права на них:

```
nano dir2/file
```

обратите внимание на ошибку, которая говорит, что данный путь недоступен.



The screenshot shows a terminal window with the title bar "user@centos8:~/temp/dir". The menu bar includes "File", "Edit", "View", "Search", "Terminal", and "Help". The command history and output are as follows:

```
[user@centos8 dir]$ sudo mkdir /home/shared
[user@centos8 dir]$ sudo chgrp group1 /home/shared/
[user@centos8 dir]$ ll -d /home/shared
drwxr-xr-x. 2 root group1 6 Jan 11 00:40 /home/shared
[user@centos8 dir]$ sudo lid -g group1
user4(uid=1111)
user(uid=1000)
[user@centos8 dir]$ sudo chmod 770 /home/shared
[user@centos8 dir]$ ll -d /home/shared
drwxrwx---. 2 root group1 6 Jan 11 00:40 /home/shared
[user@centos8 dir]$ █
```

Как я сказал, права write и execute позволяют изменять содержимое директории, допустим, удалять файлы, даже если у нас нет права на сами файлы. Допустим, создадим общую директорию для группы пользователей:

```
sudo mkdir /home/shared
```

пусть у неё будет владелец root, а группа group1:

```
sudo chgrp group1 /home/shared
ll -d /home/shared
```

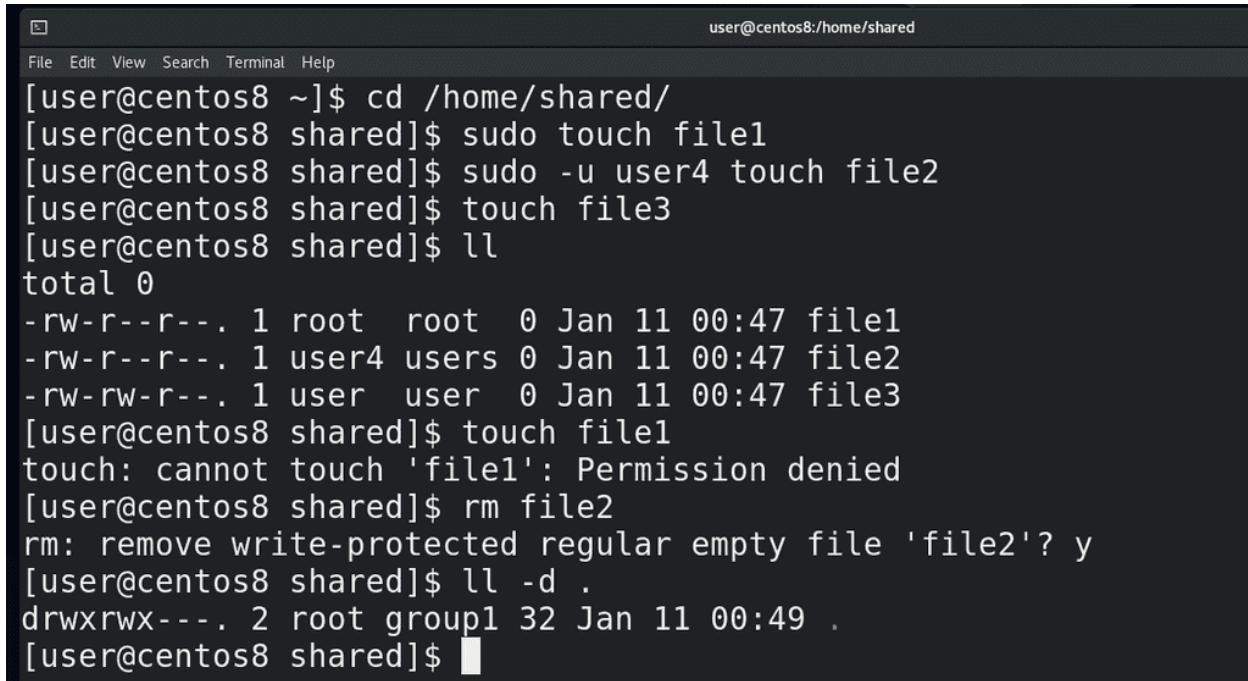
в которую входят user и user4:

```
sudo lid -g group1
```

И выставим права на директорию:

```
sudo chmod 770 /home/shared
```

чтобы у рута и группы были все права, а у остальных никаких.



The screenshot shows a terminal window with the title bar "user@centos8:/home/shared". The menu bar includes "File", "Edit", "View", "Search", "Terminal", and "Help". The terminal output is as follows:

```
[user@centos8 ~]$ cd /home/shared/
[user@centos8 shared]$ sudo touch file1
[user@centos8 shared]$ sudo -u user4 touch file2
[user@centos8 shared]$ touch file3
[user@centos8 shared]$ ll
total 0
-rw-r--r--. 1 root  root  0 Jan 11 00:47 file1
-rw-r--r--. 1 user4 users 0 Jan 11 00:47 file2
-rw-rw-r--. 1 user  user  0 Jan 11 00:47 file3
[user@centos8 shared]$ touch file1
touch: cannot touch 'file1': Permission denied
[user@centos8 shared]$ rm file2
rm: remove write-protected regular empty file 'file2'? y
[user@centos8 shared]$ ll -d .
drwxrwx---. 2 root group1 32 Jan 11 00:49 .
[user@centos8 shared]$ █
```

Значит, пользователь root, user и user4 могут создавать тут файлы:

```
cd /home/shared
sudo touch file1
sudo -u user4 touch file2
touch file3
ll
```

Как видите, тут 3 файла и write права у меня есть только на файл user-a, а другие я изменять не могу:

```
touch file1
```

При этом, я могу запросто удалить чужие файлы:

```
rm file2
```

потому что у группы group1 есть права write на эту директорию:

```
ll -d
```

```

user@centos8:/home/shared
File Edit View Search Terminal Help
[user@centos8 shared]$ sudo chmod +t /home/shared/
[user@centos8 shared]$ ll -d /home/shared
drwxrwx--T. 2 root group1 32 Jan 11 00:49 /home/shared
[user@centos8 shared]$ sudo -u user4 touch file2
[user@centos8 shared]$ rm file2
rm: remove write-protected regular empty file 'file2'? y
rm: cannot remove 'file2': Operation not permitted
[user@centos8 shared]$ rm file3
[user@centos8 shared]$ ll -d /tmp/
drwxrwxrwt. 16 root root 4096 Jan 11 00:56 /tmp/
[user@centos8 shared]$ 

```

Вы, возможно, подумали – это ж как-то неправильно, нехорошо. Кто-то может случайно или специально удалить чужие файлы. Поэтому есть специальный атрибут – дополнительное право, называемое sticky bit, которое защищает файлы внутри директории. Поставить его можно используя буквенное обозначение:

```
sudo chmod +t /home/shared
```

либо используя цифровое обозначение – у стикибита цифра 1 и она ставится перед правами:

```
sudo chmod 1770 /home/shared
```

После того, как вы поставите sticky bit на директорию, у неё появится буква Т после прав:

```
ll -d /home/shared
```

а также ls покажет эту директорию по другому. Так вот, теперь вернём файл пользователю user4:

```
sudo -u user4 touch file2
```

и попытаемся удалить ещё раз:

```
rm file2
```

Как видите, теперь у меня недостаточно прав. Но, как владелец, я могу удалять свои файлы:

```
rm file3
```

А владелец директории, не важно, root он или нет, может удалять все файлы. В системе есть директория /tmp:

```
ll -d /tmp
```

различные программы в процессе работы могут создавать здесь временные файлы. И чтобы другие процессы не удалили эти файлы здесь стоит sticky bit.

```
[user@centos8 shared]$ ll /usr/bin/passwd
-rwsr-xr-x. 1 root root 33600 Apr  7 2020 /usr/bin/passwd
[user@centos8 shared]$ passwd
Changing password for user user.
Current password: [REDACTED]

[User@centos8 shared]$ ps -ef | grep passwd
root      6140     2836  0 01:02 pts/0    00:00:00 passwd
user      6179     6143  0 01:02 pts/1    00:00:00 grep --col
[User@centos8 shared]$ [REDACTED]
```

Есть ещё пара специальных атрибутов – setuid и setgid. Вкратце – они позволяют запускать файл от имени владельца или группы соответственно. Прекрасный пример – программа passwd:

```
ll /usr/bin/passwd
```

Как видите, вместо x у владельца стоит s – это означает что здесь есть атрибут setuid, плюс ls ярким красным выделяет этот файл. Мы знаем, что passwd меняет пароль пользователя. Пользователь запускает программу passwd, вводит пароль, программа хэширует пароль и записывает в /etc/shadow. Но ведь у обычного пользователя нет прав редактировать файл /etc/shadow. И процесс passwd, запущенный от имени обычного пользователя, просто не смог бы редактировать этот файл – а значит пароль не поменялся бы. Поэтому здесь стоит setuid – когда мы запускаем программу passwd, процесс запускается не от имени нашего пользователя:

```
passwd
ps -ef | grep passwd
```

а от имени владельца этого файла – root-а. А вот процесс, запущенный от рута, может редактировать всё что угодно. Если бы владельцем файла являлся user4, то программа запускалась бы от имени пользователя user4. Вот, собственно, для этого и есть setuid.

```
[user@centos8 shared]$ sudo chmod u+s file1
[sudo] password for user:
[user@centos8 shared]$ sudo chmod g+s file2
[user@centos8 shared]$ sudo chmod 4770 file2
[user@centos8 shared]$ [REDACTED]
```

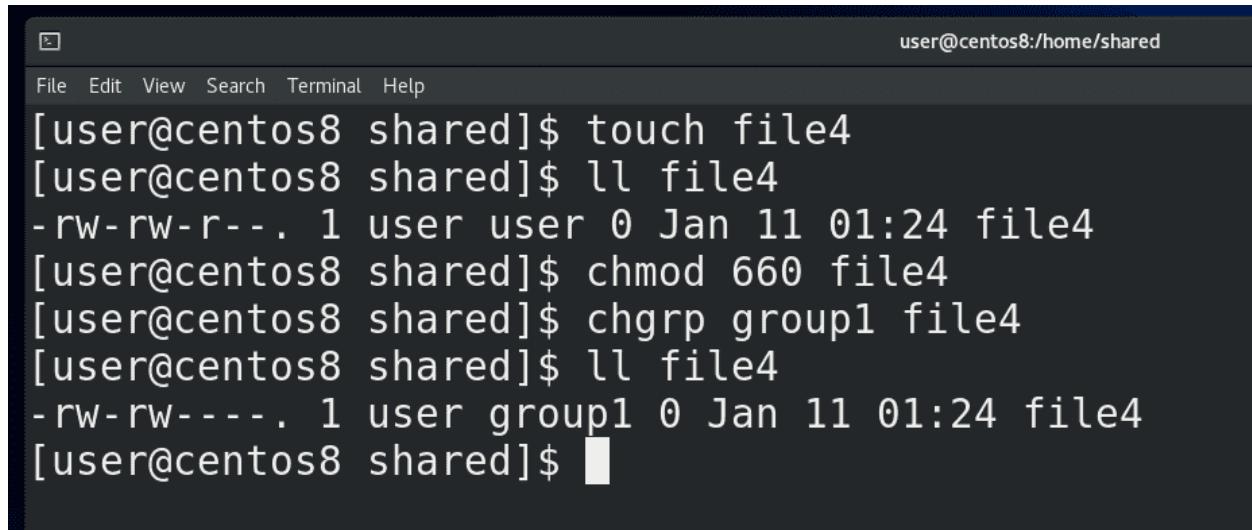
setgid делает примерно тоже самое, но уже от имени группы этого файла. Задаётся setuid, как и sticky bit – с помощью букв или чисел – u+s для setuid и g+s для setgid:

```
chmod u+s file
chmod g+s file
```

Ну и цифрами setuid это 4, setgid это 2 – всё как в правах. Допустим, чтобы поставить setuid и setgid:

```
chmod 6770 file
```

setuid, как и sudo, позволяет повысить привилегии пользователю, а значит это потенциальная брешь в безопасности. Используя программы, где стоит этот атрибут – можно попытаться стать рутом, как я показывал это с sudo. Поэтому использовать setuid вообще нежелательно и он используется только в крайних случаях, как например с passwd.



The screenshot shows a terminal window titled "user@centos8:/home/shared". The window contains the following command history:

```
[user@centos8 shared]$ touch file4
[user@centos8 shared]$ ll file4
-rw-rw-r--. 1 user user 0 Jan 11 01:24 file4
[user@centos8 shared]$ chmod 660 file4
[user@centos8 shared]$ chgrp group1 file4
[user@centos8 shared]$ ll file4
-rw-rw----. 1 user group1 0 Jan 11 01:24 file4
[user@centos8 shared]$ █
```

Также, у setgid есть одно особое применение. Если использовать setgid на директорию, то все файлы, создаваемые в этой директории, будут создаваться от имени группы, владеющей этой директорией. Обычно, когда вы создаёте какой-то файл, то владельцем является вы, а группой – ваша основная группа. Это можно увидеть в той shared директории. Смотрите, user создал файл:

```
touch file4
ll file4
```

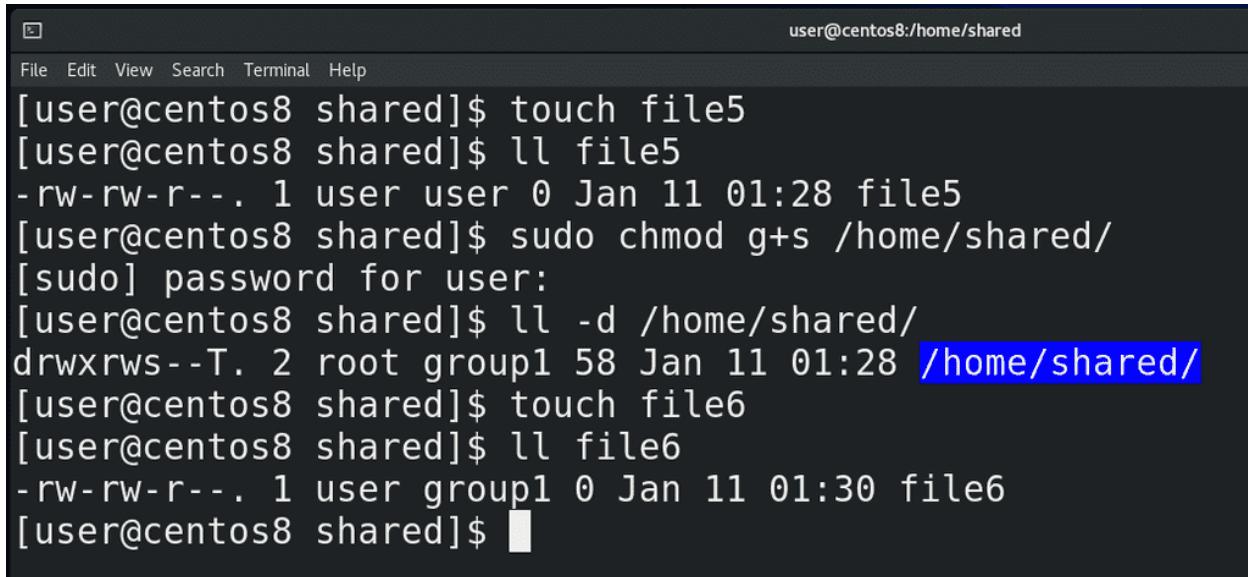
и у него группа user, потому что это его user private group, о котором мы говорили в прошлый раз. Пользователь user4 не входит в группу и не является пользователем user, а значит он для этого файла относится к others. Я не хочу, чтобы все пользователи в системе видели этот файл, поэтому меняю права на 660:

```
chmod 660 file4
```

но хочу, чтобы у пользователей в группе group1 был доступ к этому файлу:

```
chgrp group1 file4
ll file4
```

А так как пользователь user4 тоже в этой группе, он сможет редактировать этот файл.



The screenshot shows a terminal window with the title bar "user@centos8:/home/shared". The menu bar includes "File", "Edit", "View", "Search", "Terminal", and "Help". The terminal content is as follows:

```
[user@centos8 shared]$ touch file5
[user@centos8 shared]$ ll file5
-rw-rw-r--. 1 user user 0 Jan 11 01:28 file5
[user@centos8 shared]$ sudo chmod g+s /home/shared/
[sudo] password for user:
[user@centos8 shared]$ ll -d /home/shared/
drwxrws--T. 2 root group1 58 Jan 11 01:28 /home/shared/
[user@centos8 shared]$ touch file6
[user@centos8 shared]$ ll file6
-rw-rw-r--. 1 user group1 0 Jan 11 01:30 file6
[user@centos8 shared]$ █
```

Но если я создам ещё один файл:

```
touch file5
ll file5
```

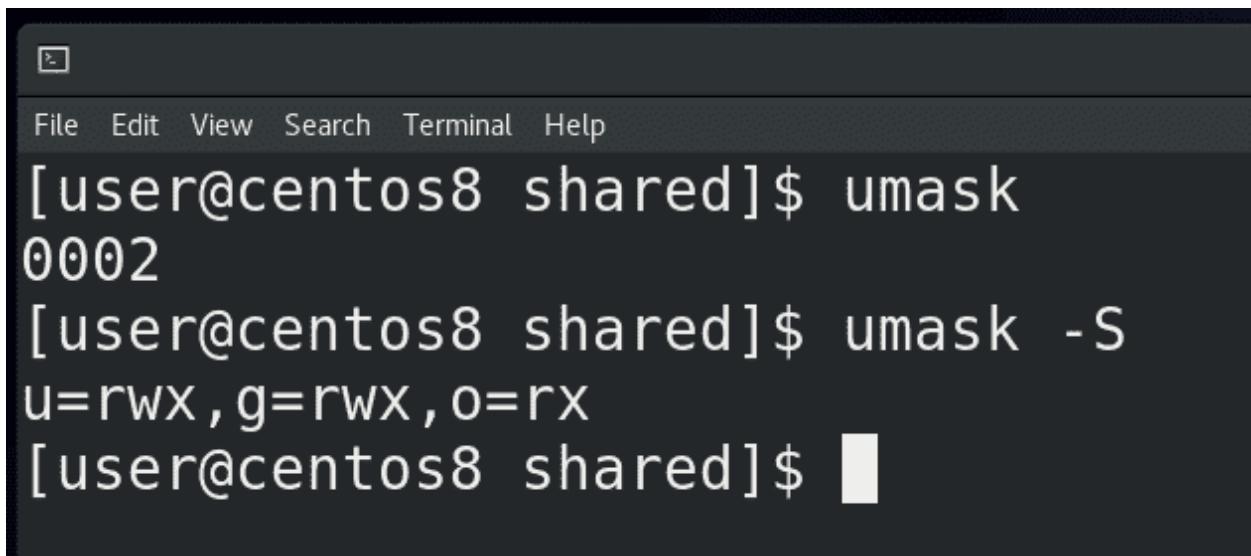
мне придётся опять менять группу для нового файла. Учитывая, что это общая директория, она специально создана для того, чтобы тут несколько пользователей из одной группы работали с файлами, было бы легче, если бы все новые файлы создавались с группой group1. И вот для этого можно использовать setgid:

```
sudo chmod g+s /home/shared
ll -d /home/shared
```

Как видите, для группы теперь стоит setgid. И теперь, когда я создаю новый файл:

```
touch file6
ll file6
```

он автоматом создаётся с этой группой, а не моей основной группой. Все новые файлы в этой директории, независимо от пользователя, будут принадлежать группе group1, благодаря чему пользователям не придётся постоянно менять группу файла, чтобы все могли работать с этими файлами.



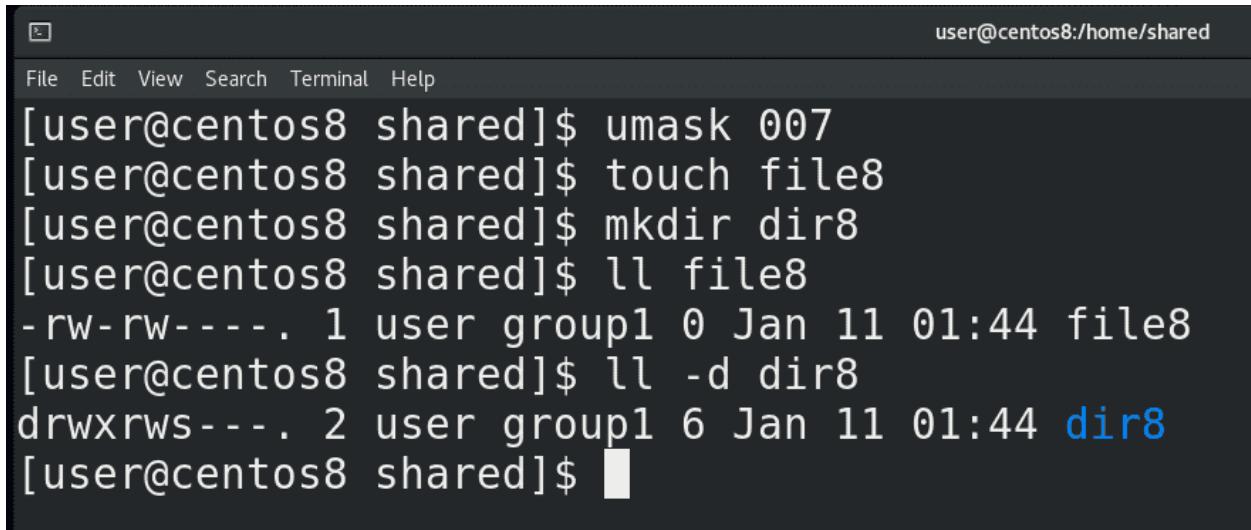
```
[user@centos8 shared]$ umask
0002
[user@centos8 shared]$ umask -S
u=rwx,g=rwx,o=rx
[user@centos8 shared]$ █
```

Ещё один момент – как вы, возможно, заметили, все файлы, которые я создаю, имеют одни и те же стандартные права – 664. Дефолтные права на новые файлы задаются утилитой umask. Если просто запустить umask, можно увидеть 0002. Можно ещё запустить:

`umask -S`

чтобы было понятнее. И так, первая цифра – для sticky bit, setuid и setgid, остальное для прав. Идея такая – берём максимальное значение – это 777 для директорий и 666 для файлов – и отнимаем те дефолтные права, которые мы хотели бы. Допустим, если мы хотим, чтобы у всех новых файлов были права 664, мы от 666 отнимаем 664 – получаем 002. Вот у нас 002 и стоит. Ну и если от 777 отнять 002 получим 775.

Вы скажете – для файлов же максимальные права тоже 777. Но вот просто нельзя создавать новые файлы с правами execute, это большая угроза безопасности. Поэтому для файлов дефолтные максимальные права это 666. А с директориями без execute нормально не поработаешь, поэтому для них 777. Если я хочу, чтобы файлы создавались с правами 660, то я от 666 отнимаю 660 – получаю 006. Но если от 777 отнять 006 получится 771, 1 в конце выглядит бессмысленно, поэтому лучше использовать 007 – тогда права для директорий будут 770, а для файлов 660.



```
user@centos8:/home/shared
File Edit View Search Terminal Help
[user@centos8 shared]$ umask 007
[user@centos8 shared]$ touch file8
[user@centos8 shared]$ mkdir dir8
[user@centos8 shared]$ ll file8
-rw-rw----. 1 user group1 0 Jan 11 01:44 file8
[user@centos8 shared]$ ll -d dir8
drwxrws---. 2 user group1 6 Jan 11 01:44 dir8
[user@centos8 shared]$ █
```

И так, как мне это применить? Я могу это сделать в текущей сессии – написать umask 007, и убедиться,

создав новый файл и директорию:

```
touch file8  
mkdir dir8  
ll file8  
ll -d dir8
```

Как видите, теперь у новых файлов для остальных пользователей нет никаких прав. Если же мы хотим, чтобы этот umask действовал для нашего пользователя всегда, мы добавляем строчку:

```
umask 007
```

в `~/.bash_profile` или `~/.bashrc`. Если помните, login shell у нас все равно считывает `~/.bashrc`, а вот если задействуется non login shell, то он не прочтёт `bash_profile`, поэтому в некоторых случаях лучше писать в `~/.bashrc`. Ну и если мы говорим про всех пользователей, то используйте файлы `/etc/profile` и `/etc/bashrc`.

Стандартные права делят пользователей на владельца, группу и остальных, что в большинстве случаев достаточно, но иногда всё же нет. Что, если мы хотим дать права на файл всем в группе, кроме двух её участников? Ради этого придётся создавать отдельную группу без этих двух участников. Или, допустим, нужно дать права на 3 группы, а не одну, при этом, дать какому-то пользователю больше прав, какому-то меньше. Для этого можно использовать список контроля доступа – acl. В общем-то речь про две команды:

```
getfacl  
setfacl
```

```
[user@centos8 shared]$ getfacl file8
# file: file8
# owner: user
# group: group1
user::rw-
group::rw-
other::---

[user@centos8 shared]$ setfacl -m u:user4:r-- file8
[user@centos8 shared]$ setfacl -m g:wheel:rwx file8
[user@centos8 shared]$ ll file8
-rw-rwx---+ 1 user group1 0 Jan 11 01:44 file8
[user@centos8 shared]$ getfacl file8
# file: file8
# owner: user
# group: group1
user::rw-
user:user4:r--
group::rw-
group:wheel:rwx
mask::rwx
other::---

[user@centos8 shared]$ setfacl -b file8
```

`getfacl file8`

показывает текущие права на файл. Как видим, владельцу user и группе group1 можно читать и изменять этот файл. User4 тоже в группе group1, но я хочу запретить ему изменять этот файл – для этого я использую утилиту setfacl:

`setfacl -m u:user4:r-- file8`

Команда говорит, что нужно модифицировать(-m) права для пользователя user4 и выставить их такими-то. Давайте ещё позволим группе wheel иметь полный доступ на файл:

`setfacl -m g:wheel:rwx file8`

После выставления acl ls показывает рядом с правами значок +, а с помощью:

`getfacl file8`

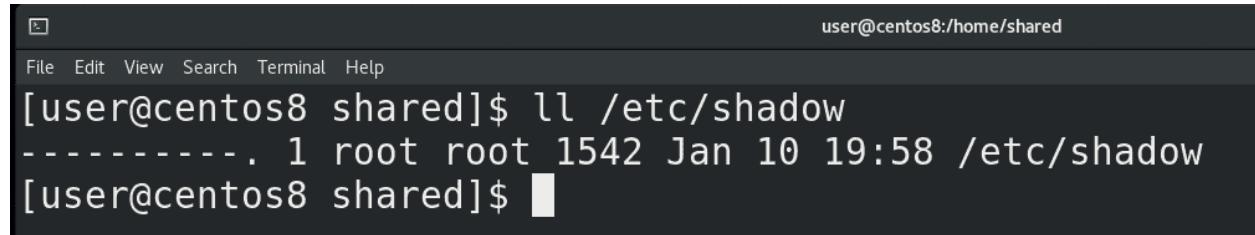
узнаем текущие права на файл. Теперь user4 не может редактировать этот файл:

`sudo -u user4 nano file8`

Чтобы удалить дополнительные права используем ключ -b:

```
setfacl -b file8  
getfacl file8
```

Возможно, тему ACL я еще затрону отдельно, но вкратце этого достаточно.

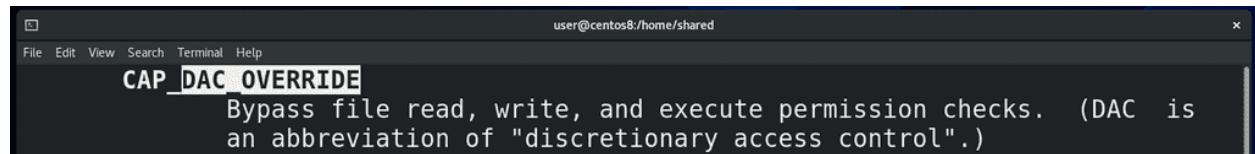


```
[user@centos8 shared]$ ll /etc/shadow  
----- 1 root root 1542 Jan 10 19:58 /etc/shadow  
[user@centos8 shared]$ █
```

Ну и напоследок, есть интересный пример с файлом /etc/shadow:

```
ll /etc/shadow
```

Как вы видите, на этот файл никаких прав нет, даже у его владельца – root-а. Но при этом мы знаем, что при смене пароля с passwd новый пароль прописывается в этом файле, да и если открыть этот файл с nano – то мы можем читать и изменять этот файл.



Суть в том, что все разрешения работы с файлом проверяет ядро операционной системы – оно проверяет, соответствует ли uid пользователя, обращающегося к файлу, с uid-ом владельца файла на файловой системе, есть ли пользователь в группе и т.п. И при некоторых условиях – когда к файлу обращается залогиненный root, в том числе при выполнении команды passwd, ядро просто пропускает проверку и сразу даёт доступ к файлу:

```
man capabilities  
/DAC_OVERRIDE
```

А вот для каких-нибудь сервисов, которые работают от имени рута, но запущены, допустим, при включении системы, а не вручную, этот файл недоступен для чтения. А при работе root может просто игнорировать все права на файлы.

Подводя итоги, мы с вами разобрали стандартные права – read, write и execute, команды chown, chgrp и chmod для смены прав и владельцев файлов, атрибуты sticky bit, setuid и setgid, права по умолчанию – umask, дополнительные права – acl. Администраторы всегда что-то делают с правами, тема хоть и простая, но может иметь много всяких нюансов, которые можно встретить при работе. Также для лучшего понимания советую почитать [статью](#) по ссылке.

2.20.2 Практика

Вопросы

- Запустите ls -l /etc, возьмите 3 случайных файла и опишите, какие права у этих файлов. Представьте эти права в численном виде.

Задания

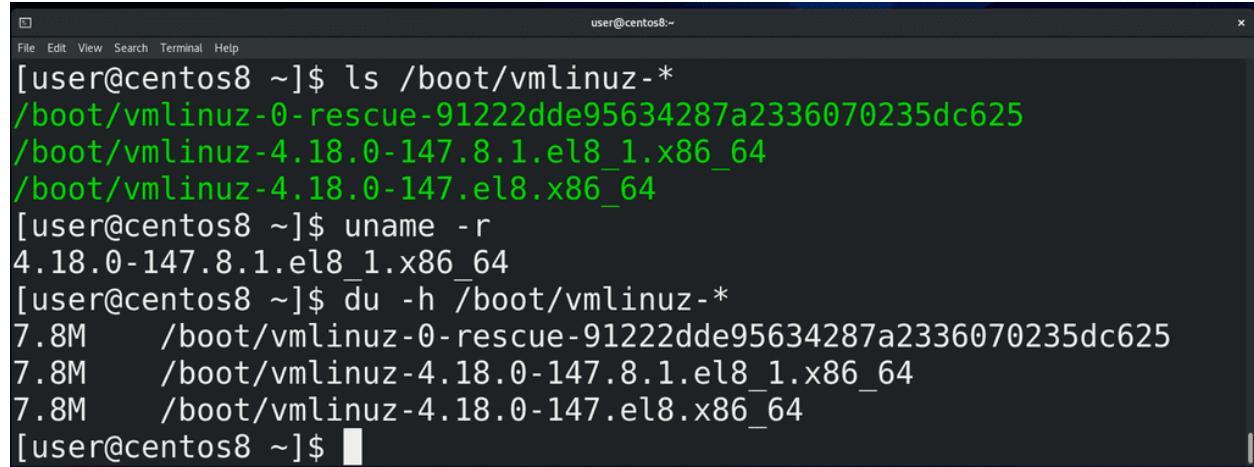
- Создайте группы company, it, sales. Создайте директории /company, /company/it, /company/sales, /company/shared. Создайте пользователей ituser1, ituser2, itadmin, suser1, suser2, sadmin так, чтобы у всех пользователей основной группой должна быть company у пользователей it* оболочкой должен быть bash, дополнительной группой должна быть it, домашняя директория должна находиться внутри директории /company/it/ у пользователей s* оболочкой должен быть sh, дополнительной группой должна быть sales, домашняя директория внутри /company/sales. Содержимое директории /company/sales должны видеть только root и участники группы sales. Содержимое директории /company/it должны видеть только root и участники группы it. Все вышеуказанные пользователи должны видеть содержимое директории /company/shared. При этом каждый пользователь лично должен внутри директории /company/shared создать файл со своим логином.
- Создайте группы academy, teachers и students, соответствующие директории, а также пользователей (student & teacher). У всех пользователей должны быть домашние директории в соответствующих директориях (/academy/students/student). Только пользователи групп должны иметь доступ на соответствующие директории (/academy/students). Также создайте группу staff, у которой будет доступ во все указанные директории. Создайте расшаренную директорию /academy/secret, внутри которой пользователи группы teachers и staff смогут создавать файлы, но не смогут удалять чужие файлы. У пользователей группы students не должно быть доступа. Создайте «программу» spry, с помощью которой студенты смогут смотреть содержимое директории /academy/secret¹. И с помощью sudo разрешите пользователям группы students от имени группы teachers удалять файлы внутри /academy/secret. Создайте файл от учителя внутри директории /academy/secret, найдите имя файла с помощью spry от имени студента и удалите файл с помощью sudo от имени студента.
- Права: Создать директорию /home/company. Внутри этой директории нужно создать директории homedirs, sales, marketing, shared. Создать группы company, sales и marketing, а также пользователей mike (основная группа - sales, дополнительная - company) и john (основная группа - marketing, дополнительная - company). У пользователей домашних директорий должны находиться в директории /home/company/homedirs. Директории /home/company/sales и /home/company/marketing должны принадлежать соответствующим группам. У пользователя и группы должны быть все права на директорию. Директория shared должна быть расшаренной, её группа должна быть company, и чтобы все новые файлы в этой директории принадлежали группе company.
- Создайте 3 группы и директории внутри /data – marketing, sales и it и 4 пользователя – user.marketing, user.sales, admin и backup. Владельцем директорий должны быть соответствующие пользователи – user.marketing владеет директорией /data/marketing и т.д. Группы директорий должны соответствовать названиям. У остальных пользователей не должно быть доступа к директориям. Пользователь admin должен иметь полный доступ ко всем директориям, пользователь backup должен иметь доступ только на чтение.
- Создайте директорию /company, чтобы владельцем был root, а группой – company. У владельца и группы должны быть все права, у остальных никаких. Пользователи не должны иметь возможность удалять чужие файлы в этой директории, а все файлы, созданные в этой директории, должны иметь группу company.

¹ Задание на смекалку

2.21 21. Ядро Linux

2.21.1 21. Ядро Linux

Мы с вами частично знакомы с некоторым функционалом ядра – оно отвечает за time sharing, управление процессами, их приоритетами и т.п, также управление памятью – та же виртуальная память, swap и всё что с этим связано, ну и из недавнего – отвечает за проверку прав на файлы - каким пользователям к каким файлам есть доступ. Это далеко не всё, чем занимается ядро, что-то мы ещё будем разбирать по мере изучения, но пока давайте разберём, что из себя представляет ядро и что администратору с ним делать.



```
[user@centos8 ~]$ ls /boot/vmlinuz-*
/boot/vmlinuz-0-rescue-91222dde95634287a2336070235dc625
/boot/vmlinuz-4.18.0-147.8.1.el8_1.x86_64
/boot/vmlinuz-4.18.0-147.el8.x86_64
[user@centos8 ~]$ uname -r
4.18.0-147.8.1.el8_1.x86_64
[user@centos8 ~]$ du -h /boot/vmlinuz-*
7.8M    /boot/vmlinuz-0-rescue-91222dde95634287a2336070235dc625
7.8M    /boot/vmlinuz-4.18.0-147.8.1.el8_1.x86_64
7.8M    /boot/vmlinuz-4.18.0-147.el8.x86_64
[user@centos8 ~]$
```

Для начала, ядро – это программа. В отличии от других программ, оно лежит в директории `/boot` и называется `vmlinuz`:

```
ls /boot
```

Почему оно лежит здесь и что это за другие файлы – это касается вопроса загрузки операционной системы, что мы будем разбирать в другой раз. Как вы видите, тут несколько файлов с названием `vmlinuz` и они отличаются версиями. Когда мы обновили систему, у нас появилась новая версия ядра, но старая не удалилась – если с новым ядром будут проблемы, всегда можно загрузиться со старого. Версию ядра, которую мы сейчас используем, можно увидеть с помощью команды:

```
uname -r
```

Давайте посмотрим, сколько же весит ядро: для этого воспользуемся утилитой

```
du
```

которая показывает размеры файлов, с ключом `-h` – чтобы отображалось не в килобайтах, а в более удобном для чтения виде:

```
du -h /boot/vmlinuz-*
```

Как видите, ядро весит почти 8 мегабайт. На самом деле, буква `z` в слове `vmlinuz` говорит о том, что ядро сжато. То есть, фактически, оно весит чуть больше.

Возможно, вы знаете – ядро Linux используется везде: android смартфоны, коих больше 3 миллиардов, работают на Linux; огромное количество сетевого оборудования, серверов, всяких медиабоксов, умных телевизоров, холодильников, машин, да даже бортовые компьютеры Space X – всё это работает на

Linux. Это огромное количество разнообразного оборудования, которое должно поддерживать ядро. Поэтому в разработке ядра участвуют тысячи крупнейших компаний и специалистов. И всё ради 8 мегабайтного файла? Конечно нет. В этом файле только основной функционал, необходимый для работы – работа с памятью, управление процессами и т.п. Когда же ядру нужен дополнительный функционал – допустим, чтобы работать с сетевым адаптером, видеокартой или другим оборудованием – ядро обращается к специальным файлам, называемым модулями. В модулях хранится код, необходимый для работы с оборудованием или программный функционал – допустим, драйвер для видеокарты или программа для шифрования. Обычно это происходит незаметно для пользователя – вы вставили флешку, а ядро загрузило модуль для работы с usb флешками, а также модуль для работы с файловой системой на этой флешке. На других операционных системах это может работать по другому – есть различные архитектуры ядер операционных систем. У Linux архитектура модульная – то есть какой-то функционал вынесен в модули и подгружается по необходимости. Также Linux называют монолитным – потому что всё что делает ядро происходит в рамках одной программы – а правильнее сказать – все части ядра работают в одном адресном пространстве. Помните, мы обсуждали, что это такое, когда говорили о процессах? Но так как у Linux-а огромный функционал, который бессмысленно держать одновременно в памяти – поэтому функционал вынесен в модули, благодаря чему ускоряется работа ядра.

```
[user@centos8 ~]$ ls /lib/modules
4.18.0-147.8.1.el8_1.x86_64 4.18.0-147.el8.x86_64
[user@centos8 ~]$ cd /lib/modules/$(uname -r)
[user@centos8 4.18.0-147.8.1.el8_1.x86_64]$ ls
bls.conf      modules.alias.bin   modules.drm          source
build         modules.block       modules.modesetting symvers.gz
config        modules.builtin     modules.networking System.map
extra         modules.builtin.bin modules.order        updates
kernel        modules.dep        modules.softdep    vdso
misc          modules.dep.bin   modules.symbols     vmlinuz
modules.alias modules.devname   modules.symbols.bin weak-updates
[user@centos8 4.18.0-147.8.1.el8_1.x86_64]$ less modules.alias
```

Так вот, модули ядра хранятся в директории `/lib/modules/`:

```
ls /lib/modules
```

где есть директории для каждой версии установленного ядра. Зайдём в директорию текущего ядра:

```
cd /lib/modules/$(uname -r)
ls
```

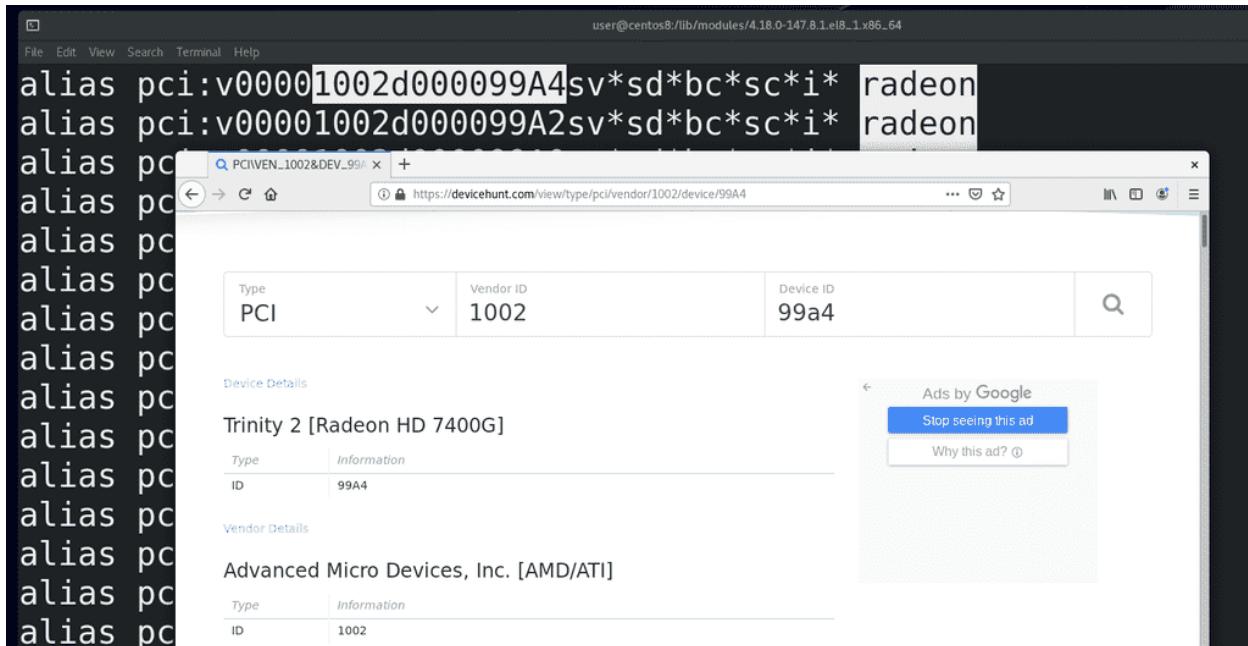
и посмотрим файл `modules.alias`:

```
less modules.alias
```

Тут перечислено – для каких устройств какие модули грузить в ядро.

В отличии от Windows, где вы ставите систему, а потом доустанавливаются драйвера, в Linux большинство драйверов уже предустановлены в виде модулей. Это благодаря тому, что многие производители оборудования сотрудничают с разработчиками ядра и предоставляют открытый исходный код драйверов на оборудование. Но, естественно, далеко не все производители предоставляют исходный код своих драйверов, из-за чего что-то может не работать из коробки – зачастую, это касается драйверов на wi-fi. Иногда, допустим, в случае с драйверами на видеокарты Nvidia, находятся энтузиасты, которые с помощью реверс инжиниринга создают свободные драйвера – т.е. берут драйвер с закрытым исходным кодом, изучают его с помощью специальных программ и методик и стараются воссоздать

этот драйвер. Для видеокарт Nvidia таким образом создан драйвер nouveau. Зачастую это работает – естественно без каких-либо гарантий, потому что драйвер написан не самим производителем. При этом сам производитель – тот же Nvidia, также предоставляет свой драйвер в виде модуля, но уже с закрытым исходным кодом, т.е. проприетарный, поэтому он не бывает включён в ядро по умолчанию, из-за чего нужно самому доустановить этот модуль. К примеру, после установки Centos на Virtualbox, мы с вами установили гостевые дополнения Virtualbox вручную именно потому, что они не под лицензией GPL, хотя сам Virtualbox имеет открытый исходный код с лицензией GPL. Всё это к тому, что если вы поставили Linux и у вас что-то не работает, допустим, wifi, то, скорее всего, производитель wifi адаптера не открыл исходный код своих драйверов и вам придётся искать нужный драйвер на сайте производителя, либо гуглить. Однако, некоторые юзер-френдли дистрибутивы, допустим Ubuntu, делают это за вас – после того, как вы установите Ubuntu, система найдёт нужные проприетарные драйвера и предложит вам их установить, что удобно для новичков.



Возвращаясь к нашему файлу, во втором столбике у нас перечислены идентификаторы оборудования – так называемые hardware id. Возьмём для примера radeon - /radeon – это видеокарты от AMD. Чтобы было удобнее, откроем сайт devicehunt.com – где мы можем увидеть информацию о вендорах и оборудовании. Так вот, если ядро видит, что к PCI шине подключено устройство, у которого vendor id – 1002, а device id – 99A4 – ядро знает, что для этого устройства нужен модуль с названием radeon. Тут могут быть ещё версии какого-то оборудования, классы и подклассы – но нам это сейчас не особо важно. Если вам интересно, что значит все эти обозначения, посмотрите по ссылке.

```

filename:      /lib/modules/4.18.0-147.8.1.el8_1.x86_64/kernel/drivers/gpu/drm/radeon/radeon.ko.xz
license:       GPL and additional rights
description:   ATI Radeon
author:        Gareth Hughes, Keith Whitwell, others.
alias:         pci:v00001002d0000131Dsv*sd*bc*sc*i*
alias:         pci:v00001002d0000131Csv*sd*bc*sc*i*
alias:         pci:v00001002d0000131Bsv*sd*bc*sc*i*
parm:          tv:TV enable (0 = disable) (int)
parm:          audio:Audio enable (-1 = auto, 0 = disable, 1 = enable) (int)
parm:          disp_priority:Display Priority (0 = auto, 1 = normal, 2 =

```

Сам этот файл не статичный, он генерируется от информации из самих модулей. Чтобы увидеть информацию о каком-нибудь модуле, можно использовать команду modinfo – допустим:

```
modinfo radeon | less
```

Тут мы видим расположение и имя файла, причём, у всех модулей расширение .ko, ну а .xz в конце означает, что модуль в сжатом виде. Также лицензия, автор, описание. И чуть ниже у нас alias-ы – собственно на основе этого и генерируется файл modules.alias, который мы смотрели. Ну и ниже – параметры – функционал оборудования на уровне драйвера. Допустим, audio – будет ли у нас видеокарта работать со звуком через тот же hdmi.

```

[user@centos8 ~]$ ls /sys/
block  class  devices  fs      kernel  power
bus    dev    firmware  hypervisor  module
[user@centos8 ~]$ ls /sys/bus/pci/devices/00*
'/sys/bus/pci/devices/0000:00:00.0':
ari_enabled          dma_mask_bits     modalias     revision
brokenparity_status  driver_override  msi_bus     subsystem
class                enable           numa_node   subsystem_device
config               firmware_node   power       subsystem_vendor
consistent_dma_mask_bits  irq           remove     uevent
d3cold_allowed       local_cpulist   rescan     vendor
device               local_cpus     resource

```

Так вот, недавно мы узнали о виртуальной файловой системе procfs, через которую ядро нам показывает информацию о процессах. А для структурированной информации об устройствах и драйверах есть виртуальная файловая система sysfs, доступная в директории /sys:

```
ls /sys
ls /sys/bus/pci/device/00*
```

И хотя тут куча файлов, через которые можно увидеть очень много информации, сидеть и копаться в этих файлах не всегда удобно, есть утилиты, которые показывают эту же информацию в более компактном и простом виде.

```
[user@centos8 ~]$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                1

[user@centos8 ~]$ sudo lshw
[sudo] password for user:
centos8
      description: Computer
      product: VirtualBox
      vendor: innotek GmbH
      version: 1.2
```

Например:

lscpu

показывает информацию о процессоре,

lspci

показывает информацию об устройствах, подключённых на pci шину,

lsusb

устройства, подключённые к usb. Для более подробной информации вы можете использовать lshw:

sudo lshw

а из графических утилит есть hardinfo.

```
trol: RX
[ +0.001168] IPv6: ADDRCONF(NETDEV_UP): enp0s3: link is not ready
[ +0.052074] IPv6: ADDRCONF(NETDEV_UP): enp0s3: link is not ready
[ +0.705776] IPv6: ADDRCONF(NETDEV_CHANGE): enp0s3: link becomes ready
[ +2.484575] virbr0: port 1(virbr0-nic) entered blocking state
[ +0.000236] virbr0: port 1(virbr0-nic) entered listening state
[ +0.070235] virbr0: port 1(virbr0-nic) entered disabled state
[ +3.465436] 11:15:49.685090 main      VBoxService 6.1.16 r140961 (verbosity: 0) linux.amd64 (Oct 15 2020 16:40:53) release log
11:15:49.685091 main      Log opened 2021-01-15T11:15:49.685084000Z
```

Чтобы видеть, что происходит в ядре при запуске системы или сейчас, например, вы вставили флешку и хотите понять, видит ли её система или нет, вы можете использовать утилиту dmesg:

```
sudo dmesg -wH
```

Запустили команду, вставили флешку или любое другое устройство, и тут вы увидите, как ядро распознаёт устройство.

Ядро, при виде какого-нибудь оборудования или при необходимости работы с каким-нибудь программным функционалом, загружает модули автоматически. И хотя работать с этим вам придётся не так часто, вы должны иметь представление, как это работает и как это менять. Например, может быть требование, чтобы система игнорировала флешки, хотя, по умолчанию, вы вставили флешку и она работает.

```
[user@centos8 ~]$ sudo modprobe radeon
[sudo] password for user:
[user@centos8 ~]$ lsmod | grep radeon
radeon                  1626112  0
i2c_algo_bit              16384  1 radeon
drm_kms_helper            217088  3 vmwgfx,vboxvideo,radeon
ttm                      110592  3 vmwgfx,vboxvideo,radeon
drm                     524288  9 vmwgfx,drm_kms_helper,vboxvideo,radeon,ttm
[user@centos8 ~]$ sudo modprobe -r radeon
[user@centos8 ~]$ sudo modprobe -r vboxguest
modprobe: FATAL: Module vboxguest is in use.
```

И так, для управления модулями ядра используется утилита:

```
modprobe
```

Допустим, если хотим загрузить модуль radeon, пишем:

```
sudo modprobe radeon
```

Увидеть можем с помощью утилиты lsmod, которая показывает загруженные модули:

```
lsmod | grep radeon
```

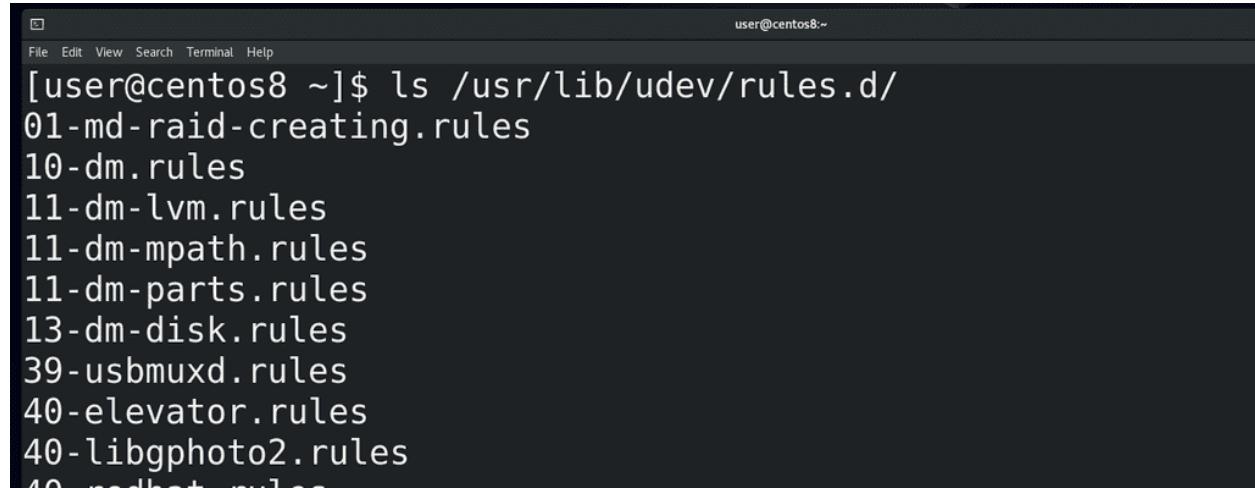
Одни модули могут работать с другими и сами могут использоваться какими-то процессами. Например, чтобы выгрузить модуль из ядра, можно использовать тот же modprobe с ключом -r:

```
sudo modprobe -r radeon
```

Этот модуль не использовался, поэтому мне получилось с лёгкостью его выгрузить. Но есть модуль, допустим, vboxguest, который используется и убрать его с помощью modprobe не получится:

```
sudo modprobe -r vboxguest
```

Предварительно нужно избавиться от процессов, которые используют этот модуль. Например, чтобы выгрузить драйвер видеокарты, вам нужно будет предварительно завершить все приложения, использующие графический интерфейс.



The screenshot shows a terminal window with a dark background and light-colored text. At the top, there's a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The title bar says 'user@centos8:~'. The main area of the terminal displays the command 'ls /usr/lib/udev/rules.d/' followed by a list of files:

```
[user@centos8 ~]$ ls /usr/lib/udev/rules.d/
01-md-raid-creating.rules
10-dm.rules
11-dm-lvm.rules
11-dm-mpath.rules
11-dm-parts.rules
13-dm-disk.rules
39-usbmuxd.rules
40-elevator.rules
40-libgphoto2.rules
40-usb-camera.rules
```

Так вот, возвращаясь к теме про автоматическую загрузку модулей. В системе есть программа, называемая udev – именно она отвечает за управление устройствами. И когда ядро видит новое устройство, оно создаёт событие, которое отслеживает udev. У udev есть большое количество правил:

```
ls /usr/lib/udev/rules.d/
```

которые оно применяет при виде того или иного события. Например, udev видит в событии, что в usb подключено устройство, которое говорит что оно является накопителем – и у udev есть правило, которое при виде такого устройства создаёт для него специальный файл в директории /dev с таким-то названием – допустим, sdb. Этот файл ядро связывает с драйвером, работающим с оборудованием. Таким образом наша флешка становится файлом, благодаря чему мы через этот файл можем взаимодействовать с устройством. Точно также можно в udev прописать, чтобы при виде usb устройства для него передавался специальный параметр, который бы запрещал устройству что-либо делать. Углубляться в udev мы пока не будем, для начала хватит понимания, зачем он вообще нужен.

```
[user@centos8 ~]$ cat /lib/modules/$(uname -r)/modules.builtin
kernel/arch/x86/crypto/glue_helper.ko
kernel/arch/x86/crypto/aes-x86_64.ko
kernel/arch/x86/crypto/aesni-intel.ko
kernel/arch/x86/crypto/sha1-ssse3.ko
kernel/arch/x86/crypto/sha256-ssse3.ko
kernel/arch/x86/kernel/msr.ko
kernel/arch/x86/kernel/cpuid.ko
kernel/arch/x86/platform/intel/iosf_mbi.ko
kernel/mm/zpool.ko
kernel/mm/zbud.ko
kernel/mm/zsmalloc.ko
```

Модули можно скомпилировать в ядро – тогда отпадает необходимость загружать модули из файлов, но это увеличивает размер ядра. Такие модули называются встроенными и их список можно увидеть в файле modules.builtin:

```
cat /lib/modules/$(uname -r)/modules.builtin
```

А те модули, которые загружаются при необходимости и которые можно, теоретически, выгрузить, называются загружаемыми. Встроенные же модули выгрузить из ядра не получится.

The screenshot shows a terminal window with the nano text editor. The title bar indicates it's 'GNU nano 2.9.8' and the file path is '/etc/modules-load.d/radeon.conf'. The main area of the editor contains the word 'radeon'. At the bottom of the screen, there is a status message '[Wrote 1 line]' and a row of keyboard shortcuts: ^G Get Help, ^O Write Out, ^W Where Is, ^K Cut Text, ^J Justify, ^X Exit, ^R Read File, ^\ Replace, ^U Uncut Text, ^T To Spell.

Иногда может понадобится, чтобы загружаемые модули загружались всегда при включении компьютера. Для этого нужно создать файл в директории /etc/modules-load.d/ с .conf в конце имени, допустим, radeon.conf:

```
sudo nano /etc/modules-load.d/radeon.conf
```

где вписываем название модуля, которые мы хотели бы загружать в ядро при включении. Теперь, после перезагрузки, этот модуль автоматом загрузится.

The screenshot shows a terminal window with the nano text editor. The title bar indicates it's 'GNU nano 2.9.8' and the file path is '/etc/modprobe.d/kvm.conf'. The main area of the editor contains several lines of code starting with '#': '# User changes in this file are preserved across upgrades.', '# For Intel', '#options kvm_intel nested=1', '# For AMD', '#options kvm_amd nested=1'.

Если же нам нужно, чтобы какой-то определённый модуль не загружался при включении или загружался с определёнными параметрами – теми параметрами, которые мы видели с modinfo radeon – то для этого нужно создать файл в директории /etc/modprobe.d тоже заканчивающийся на .conf, например, kvm.conf:

```
sudo nano /etc/modprobe.d/kvm.conf
```

Тут у нас есть закомментированные примеры. Чуть подробнее об этом можно почитать на [арчвики](#).

Надеюсь у вас сложилось представление, что же есть ядро. Это далеко не весь функционал ядра, я его и не смогу полностью рассмотреть. Но теперь вы немного знаете про само ядро, про его модули, про драйвера и устройства, в том числе `udev`, который позволит вам более гибко настраивать работу системы с устройствами и директорию `dev`, где у нас специальные файлы, с помощью которых мы можем работать с различным оборудованием.

2.21.2 Практика

Вопросы

1. Какие архитектуры ядер операционной системы существуют?
 2. Что такое модули ядра?
 3. Как посмотреть список загруженных модулей?
 4. Как узнать, есть ли определённый модуль в системе?
 5. Для чего нужна директория /dev?
 6. Как посмотреть список и информацию об устройствах, подключенных к компьютеру?
 7. Как загрузить и выгрузить модуль ядра?
 8. Как запретить загрузку модуля при включении ОС?
 9. Зачем нужен udev?
 10. Как посмотреть последние логи ядра?

2.22 22. Работа с дисками

2.22.1 22. Работа с дисками

```
[user@centos8 ~]$ ls /dev/
autofs          lp0           shm        tty26   tty50   usbmon2
block          lp1           snapshot  tty27   tty51   vboxguest
bsg            lp2           snd        tty28   tty52   vboxuser
bus            lp3           sr0        tty29   tty53   vcs
cdrom          mapper       stderr     tty3    tty54   vcs1
char           mcelog      stdin      tty30   tty55   vcs2
cl_centos8    mem          stdout    tty31   tty56   vcs3
console        memory_bandwidth  tty      tty32   tty57   vcs4
core           mqqueue     tty0      tty33   tty58   vcs5
cpu             net          tty1      tty34   tty59   vcs6
cpu_dma_latency network_latency  tty10    tty35   tty6   vcsa
disk           network_throughput  tty11    tty36   tty60  vcsa1
dm-0           null         tty12    tty37   tty61  vcsa2
dm-1           nvram        tty13    tty38   tty62  vcsa3
```

В прошлый раз мы разобрались, что udev при виде определённых устройств создаёт для них специальные файлы в директории /dev:

```
ls /dev
```

через которые можно взаимодействовать с устройствами. И если с большинством устройств администратору не требуется ничего делать, то вот с устройствами хранения информации практически постоянно нужно работать. При работе с дисками нужно быть крайне осторожными, потому что на дисках данные, а какие-то ошибки могут привести к потере этих данных. Поэтому всегда делайте бэкапы и убеждайтесь, что они в рабочем состоянии.



Есть разные типы накопителей и они могут по разному подключаться к компьютеру – старые диски подключались по IDE, сейчас преимущество у SATA, набирают популярность nvme SSD, есть ещё всякие флешки, подключаемые по usb, во многих компаниях диски выдаются по сети хранения данных, называемой SAN, а в облачных средах вам выдаются виртуальные диски, детали подключения которых вас могут и не интересовать. В зависимости от некоторых факторов – типа подключения диска, правил udev, которые могут отличаться в зависимости от дистрибутива – дискам могут выдаваться различные названия. При этом, несмотря на различия, для работы с устройствами хранения используется протокол SCSI – это касается и USB флешек, и SATA дисков, и сетей хранения данных и многое другого.

```
[user@centos8 ~]$ ls /dev/sd*
/dev/sda  /dev/sda1  /dev/sda2
[user@centos8 ~]$ ls /dev/sr*
/dev/sr0
[user@centos8 ~]$ ls -l /dev/sda
brw-rw----. 1 root disk 8, 0 Feb 14 15:16 /dev/sda
[user@centos8 ~]$ stat /dev/sda
  File: /dev/sda
  Size: 0          Blocks: 0          IO Block: 4096   block special file
Device: 6h/6d  Inode: 12991      Links: 1      Device type: 8,0
Access: (0660;brw-rw----)  Uid: (    0/    root)  Gid: (    6/    disk)
Context: system_u:object_r:fixed_disk_device_t:s0
Access: 2021-02-14 15:16:13.735000259 +0400
Modify: 2021-02-14 15:16:12.652000225 +0400
Change: 2021-02-14 15:16:12.652000225 +0400
 Birth: -
[user@centos8 ~]$ █
```

Поэтому большая часть накопителей будет именоваться sd – т.е. scsi диск:

```
ls /dev/sd*
```

а дальше каждому устройству будет даваться буква по алфавиту – sda, sdb,sdc и т.п. Для cd приводов будет даваться название sr:

```
ls /dev/sr*
```

sr0, sr1 и т.п. А в облаках вы будете натыкаться на названия vda, vdb и т.д. Что объединяет все накопители? Операционная система обменивается с этими устройствами данными в виде блоков данных фиксированной длины. Команда:

```
ls -l /dev/sda  
stat /dev/sda
```

покажет перед правами символ b – указывающий, что это блочное устройство.

```
[user@centos8 ~]$ ls -l /dev/input/mouse0  
crw-rw----. 1 root input 13, 32 Feb 14 15:16 /dev/input/mouse0  
[user@centos8 ~]$ stat /dev/input/mouse0  
  File: /dev/input/mouse0  
  Size: 0          Blocks: 0          IO Block: 4096   character special file  
Device: 6h/6d    Inode: 11121      Links: 1      Device type: d,20  
Access: (0660/crw-rw----)  Uid: (    0/    root)  Gid: (  999/   input)  
Context: system_u:object_r:mouse_device_t:s0  
Access: 2021-02-14 15:16:12.468000219 +0400  
Modify: 2021-02-14 15:16:12.468000219 +0400  
Change: 2021-02-14 15:16:12.468000219 +0400  
 Birth: -  
[user@centos8 ~]$
```

Кроме блочных устройств существуют символьные – такие устройства работают с потоком данных, а не с блоками. Допустим, та же мышка:

```
ls -l /dev/input/mouse0  
stat /dev/input/mouse0
```

Перед такими файлами стоит символ c – character device – символьное устройство.

```
[user@centos8 ~]$ lsscsi -s  
[1:0:0:0]  cd/dvd  VBOX      CD-ROM           1.0    /dev/sr0      -  
[2:0:0:0]  disk    ATA       VBOX HARDDISK     1.0    /dev/sda    21.4GB  
[user@centos8 ~]$
```

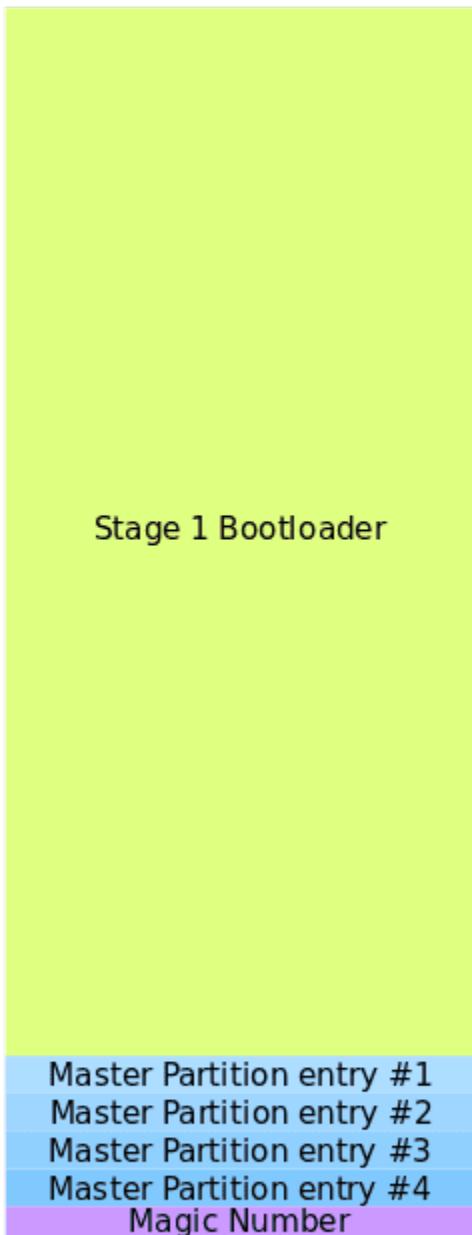
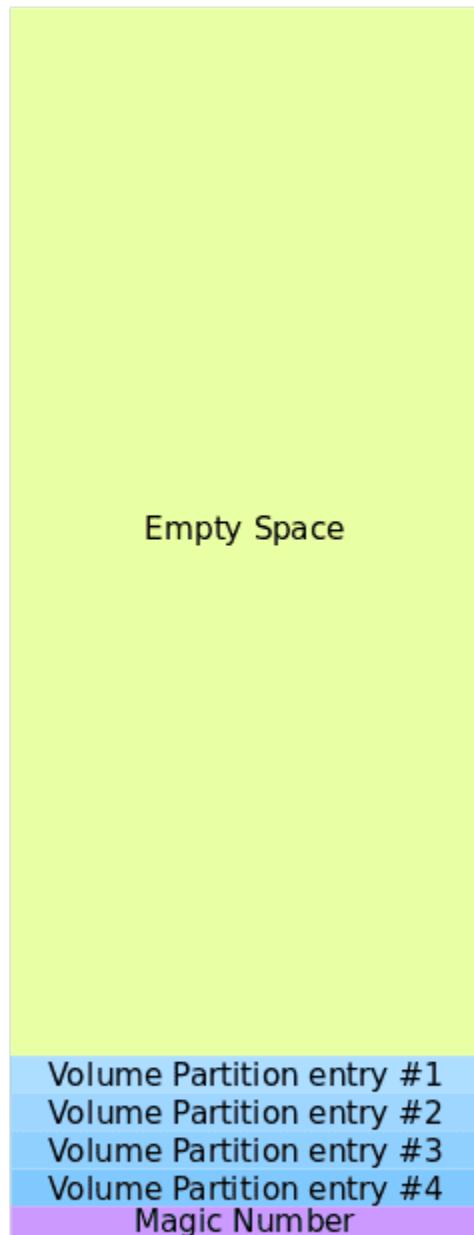
Но, как я уже сказал, чаще всего накопители работают через scsi протоколы – поэтому, как бы там не было написано в правилах udev, нам необязательно гадать – мы можем с помощью утилиты lsscsi увидеть наши диски:

```
lsscsi -s
```

Как видите, тут у меня подключены дисковод и диск на 20 гигабайт, который получил название sda. Но названия, которые даёт udev – sda, sdb и т.п. – не закрепляются за дисками навсегда. Каждый раз, когда вы запускаете систему или подключаете устройства, udev даёт название по порядку. Да, есть порядок обнаружения устройств и зачастую одни и те же диски будут называться одинаково, но ни в коем случае нельзя ориентироваться на эти названия. Допустим, если у вас 3 диска – sda, sdb и sdc и вы переподключите их, либо один перестанет работать – то sdc начнёт называться sdb. В теории это может

привести к потере данных. Как? Мы разберём чуть позже. Просто запомните, что ориентироваться на эти буквы не стоит.

Из темы «О файловых системах» мы выяснили – чтобы мы могли создавать, хранить, изменять и в целом работать с файлами на диске – нам нужна файловая система. Для этого мы можем записать её на диск. В принципе, такая схема будет работать, но в целом это неудобно и может создать нам проблемы в будущем. Например, в будущем нам может понадобится переустановить систему. Как правило, это предполагает удаление старой файловой системы и создание новой – это называется форматированием файловой системы. Но, при этом, все файлы, которые мы хотели бы перенести на новую систему, также затрутся. Конечно, можно заранее закинуть всё на флешку и потом вернуть обратно – но это лишняя работа и потеря времени. Возможно, вы знаете как избежать этой проблемы – на том же Windows у вас может быть том D, на который вы кидаете файлы, и при форматировании эти данные не стираются – потому что стирается файловая система в томе C. На GNU/Linux файлы пользователей хранятся в директории /home, поэтому вам будет достаточно отдельить /home от корня. Т.е. предполагается, что у вас две файловые системы на одном диске. Для этого нужно разделить диск на так называемые разделы, и на каждый раздел записать свою файловую систему. Но чтобы компьютер знал – где начинается один раздел, где он заканчивается и начинается другой – нужно специальное место в начале диска, где указывается эта информация – таблица разделов.

Master Boot Record**Sector #0****Volume Boot Record****Sector #63 - 2₃₂**

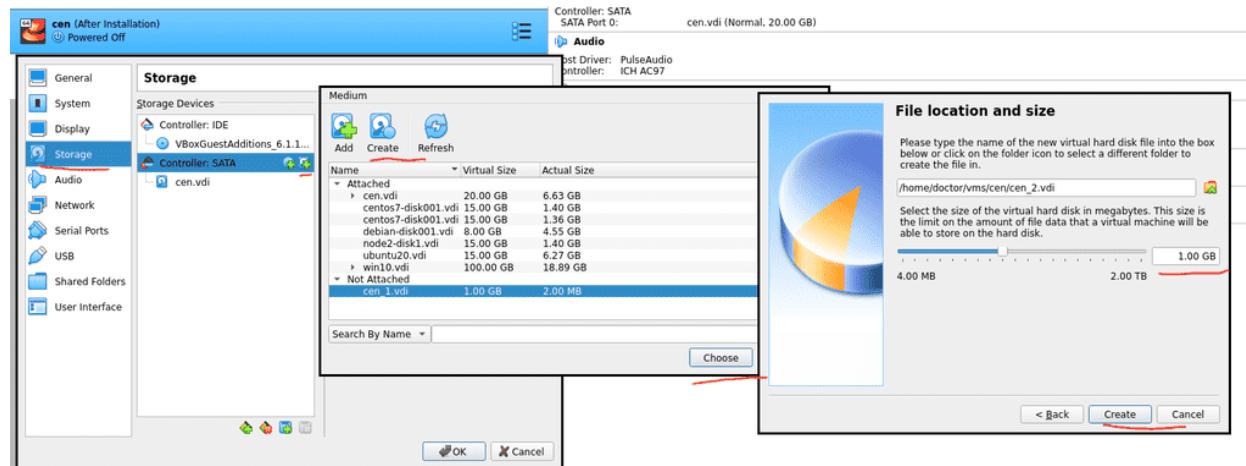
Each partition table entry comprises of 16 octets:

Flag	Start CHS	Type	End CHS	Start LBA	Size
1	3	1	3	4	4 octets

Есть разные типы таблиц разделов: MBR – которую также называют dos или ms-dos; GPT; у Apple и BSD свои таблицы разделов; есть ещё какие-то – но это нас не интересует, в основном вы будете иметь дело с MBR и GPT. У аббревиатуры MBR есть и другое значение – главная загрузочная запись – и сама таблица разделов хранится внутри этой записи. А загрузочная запись MBR была нужна для компьютеров раньше. Дело в том, что раньше на компьютерах был чип BIOS, в котором был

ряд микропрограмм, и, кроме всего прочего, BIOS отвечал за включение компьютера. Но BIOS был сильно ограничен – он должен был быть не больше десятка килобайт, оперативки ему было доступно было максимум мегабайт – ну и в таких условиях сильно не разгуляешься. И BIOS должен был в итоге загрузить операционную систему – но ведь с такими ограничениями не добавить поддержку какой-то файловой системы и программы, загружающей операционную систему. Причём операционные системы то разные, у каждого своя файловая система, каждую по своему грузить. Поэтому BIOS просто обращался к нулевому сектору жёсткого диска, где и находилась главная загрузочная запись – MBR. А там у нас и загрузчик операционной системы и таблица разделов. При этом сам MBR тоже был ограничен – всего 512 байт, из которых 446 байт на загрузчик и 64 на таблицу разделов. Забегая вперёд, скажу, что там ещё первые 63 сектора оставались свободными, в которые и помещается основная часть загрузчика, а не только эти 446 байт, но это тема загрузки операционной системы, мы это рассмотрим в другой раз.

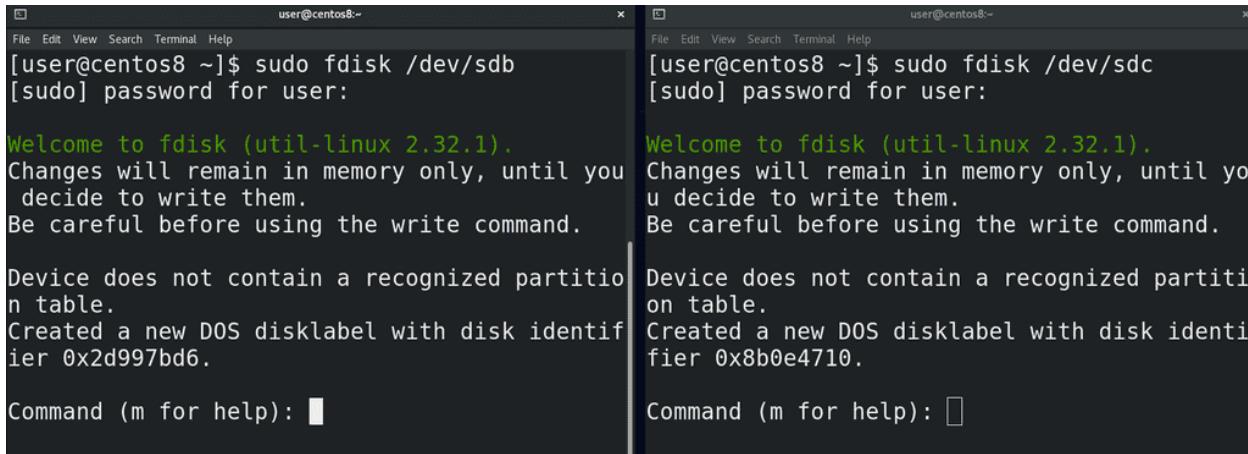
Так вот, MBR нам даёт 64 байта на то, чтобы поместить всю информацию о разделах. А на информацию о каждом разделе требуется 16 байт. В итоге – 4 раздела. Потом появилась расширенная загрузочная запись (VBR), благодаря чему можно было создавать расширенные разделы, которые позволяли обойти ограничение в 4 раздела. Грубо говоря, там вместо самой записи о разделе указывалась ссылка на другую таблицу, в которой указывались дополнительные разделы. Также в таблице разделов MBR не получилось бы указать разделы больше 2Тб, так как адрес не помещается в таблице. По итогу, BIOS своими ограничениями создавал кучу проблем с вынужденными обходными решениями. В нулевых же решили избавиться от этих ограничений и создали новый стандарт – UEFI. Тут уже разгулялись – UEFI может весить десяток мегабайт, понимает файловые системы, может работать с сетью, имеет графический интерфейс. Точнее, это всё можно реализовать в UEFI, но не каждый производитель это делает, разве что какой-то стандартный функционал. И, конечно же, отпала необходимость в загрузчике в MBR – зачем ограничиваться 446 байтами, когда можно в UEFI добавить поддержку файловой системы, где и будет лежать полноценный загрузчик? Плюс растут объёмы дисков – поэтому MBR заменили на GPT. Необходимость держать загрузчик в начальном секторе отпала – UEFI для этого использует специальный раздел EFI с файловой системой FAT32. Ограничение в 2 Тб тоже пропало, у GPT теоретическое ограничение почти в 10 Зеттабайт. Что касается количества разделов – то они, в принципе, не ограничены, разве что только со стороны операционной системы, но там речь про 128 разделов, чего с лихвой хватает. Но, при этом, UEFI в целях совместимости позволяет установить загрузчик в нулевой сектор, как это было в MBR.



```
[user@centos8 ~]$ lsscsi -s
[1:0:0:0] cd/dvd VBOX     CD-ROM      1.0   /dev/sr0   -
[2:0:0:0] disk   ATA      VBOX HARDDISK 1.0   /dev/sda   21.4GB
[3:0:0:0] disk   ATA      VBOX HARDDISK 1.0   /dev/sdb   1.07GB
[4:0:0:0] disk   ATA      VBOX HARDDISK 1.0   /dev/sdc   1.07GB
```

Прежде чем пойдём дальше, давайте немного попрактикуемся с таблицами разделов. Но для начала

нам понадобится добавить диски для нашей виртуальной машины. Поэтому выключаем виртуалку, открываем её настройки, переходим во вкладку Storage – Controller SATA и добавляем два жёстких диска, создаём их – по гигабайту будет достаточно, можно даже обойтись меньшим объёмом. Нажимаем OK и запускаем виртуалку. Теперь lsblk -s покажет нам ещё два диска по гигабайту, с названиями sdb и sdc.



The image shows two side-by-side terminal windows. Both windows have a dark background and white text. The left window is titled 'user@centos8:~' and shows the command [user@centos8 ~]\$ sudo fdisk /dev/sdb followed by the fdisk welcome message and partition table information. The right window is also titled 'user@centos8:~' and shows the command [user@centos8 ~]\$ sudo fdisk /dev/sdc, with similar fdisk output.

```
[user@centos8 ~]$ sudo fdisk /dev/sdb
[sudo] password for user:

Welcome to fdisk (util-linux 2.32.1).
Changes will remain in memory only, until you
decide to write them.
Be careful before using the write command.

Device does not contain a recognized parti-
tion table.
Created a new DOS disklabel with disk identi-
fier 0x2d997bd6.

Command (m for help): █

[User@centos8 ~]$ sudo fdisk /dev/sdc
[sudo] password for user:

Welcome to fdisk (util-linux 2.32.1).
Changes will remain in memory only, until yo
u decide to write them.
Be careful before using the write command.

Device does not contain a recognized parti-
on table.
Created a new DOS disklabel with disk identi-
fier 0x8b0e4710.

Command (m for help): █
```

И так, диски у нас есть – sdb и sdc. Давайте создадим таблицу разделов и пару разделов. Для этого используем утилиту fdisk:

```
sudo fdisk /dev/sdb
sudo fdisk /dev/sdc
```

И так, для начала, нас предупредили, что всё что мы делаем – не сразу происходит, а сохраняется в памяти, и если мы захотим – то пишем w и изменения вступают в силу. Дальше нас предупредили, что на диске не нашли никакой таблицы разделов, поэтому программа создала таблицу DOS – но опять же, без write-а ничего на деле не изменилось.

```

user@centos8:~
```

File Edit View Search Terminal Help

w write table to disk and exit
q quit without saving changes

Create a new label

g create a new empty GPT partition table
G create a new empty SGI (IRIX) partition table
o create a new empty DOS partition table
s create a new empty Sun partition table

Command (m for help): p

Disk /dev/sdb: 1 GiB, 1073741824 bytes, 2097152 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xe45d85a7

Command (m for help): █

Ну и небольшая подсказка по командам с помощью m. Для начала выведем информацию – буква p. Наш диск – /dev/sdb, его размер – 1 Гибайт, количество байт и секторов. Каждый сектор по 512 байт – умножаем на количество секторов – получаем количество байт. Буква o – создаёт таблицу разделов DOS, буква g – GPT.

```

Command (m for help): n
Partition type
  p  primary (0 primary, 0 extended, 4 free)
  e  extended (container for logical partitions)
Select (default p):

Using default response p.
Partition number (1-4, default 1): 1
First sector (2048-2097151, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-2097151, default 2097151): +200M

Created a new partition 1 of type 'Linux' and of size 200 MiB.

Command (m for help): p
Disk /dev/sdb: 1 GiB, 1073741824 bytes, 2097152 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x969824ff

Device      Boot Start   End Sectors  Size Id Type
/dev/sdb1          2048 409600   200M  83 Linux

```

И так, с помощью о создали таблицу разделов, теперь можем создавать разделы. Для этого буква п – new partition. В случае с DOS у нас спрашивает – будет ли это основной раздел или расширенный. Основных может быть 4, а для создания расширенного используется один из основных. Выберем основной – по умолчанию он и создаётся, поэтому просто enter. Дальше нужно выбрать его номер – допустим 1. Потом выбираем начальный сектор – пусть будет 2048 – оставляем по умолчанию. Дальше пишем последний сектор, либо необходимый размер для раздела через +, допустим +200M. Создалось. Теперь пишем р и видим новый раздел. Разделы называются как соответствующий диск – то есть sdb, а в конце указывается номер раздела - sdb1.

```

Command (m for help): n
Partition type
  p  primary (1 primary, 0 extended, 3 free)
  e  extended (container for logical partitions)
Select (default p): e
Partition number (2-4, default 2): 4
First sector (411648-2097151, default 411648):
Last sector, +sectors or +size{K,M,G,T,P} (411648-2097151, default 2097151):

Created a new partition 4 of type 'Extended' and of size 823 MiB.

Command (m for help): p
Disk /dev/sdb: 1 GiB, 1073741824 bytes, 2097152 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x969824ff

Device      Boot Start   End Sectors  Size Id Type
/dev/sdb1          2048 409600   200M  83 Linux
/dev/sdb4          411648 2097151 1685504 823M  5 Extended

```

Давайте, для примера, создадим ещё один раздел. н - выберем extended – в качестве основного используем 4; первый сектор оставим как есть – это первый незанятый сектор; дальше просто нажмём enter – тогда используется всё доступное пространство. Опять р – видим наши разделы. Но сам по себе расширенный раздел мы использовать не будем – он просто позволит создавать внутри него другие разделы, называемые логическими разделами.

```
Command (m for help): n
All space for primary partitions is in use.
Adding logical partition 5
First sector (413696-2097151, default 413696):
Last sector, +sectors or +size{K,M,G,T,P} (413696-2097151, default 2097151): +100M

Created a new partition 5 of type 'Linux' and of size 100 MiB.

Command (m for help): p
Disk /dev/sdb: 1 GiB, 1073741824 bytes, 2097152 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x969824ff

Device      Boot  Start    End Sectors  Size Id Type
/dev/sdb1        2048  411647  409600 200M 83 Linux
/dev/sdb4        411648 2097151 1685504 823M  5 Extended
/dev/sdb5        413696  618495  204800 100M 83 Linux

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
```

Поэтому опять н – оставляем первый сектор как есть, потом указываем размер раздела – +100M, и опять р. Теперь видим, что у нас есть 3 раздела: первый – sdb1 – это основной раздел; дальше у нас расширенный раздел – sdb4 – он у нас вроде платформы, внутри которой мы создаём логические разделы, к примеру – sdb5. Когда нас всё устраивает, мы пишем w и изменения сохраняются.

```
Command (m for help): g
Created a new GPT disklabel (GUID: 02541524-4E41-2846-AD26-7B9A332308F9).

Command (m for help): n
Partition number (1-128, default 1):
First sector (2048-2097118, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-2097118, default 2097118): +200M

Created a new partition 1 of type 'Linux filesystem' and of size 200 MiB.

Command (m for help): p
Disk /dev/sdc: 1 GiB, 1073741824 bytes, 2097152 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 02541524-4E41-2846-AD26-7B9A332308F9

Device      Start    End Sectors  Size Type
/dev/sdc1    2048  411647  409600  200M Linux filesystem

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
```

Теперь создадим разделы на диске sdc с таблицей разделов GPT. Для её создания - g. Для нового раздела – n. У нас спрашивают номер раздела – оставим 1. Первый сектор – оставляем как есть. Последний сектор - +200M. И p, чтобы вывести информацию о разделах. Как видите, никакой возни с расширенными и логическими разделами. Опять же - w – сохраняем изменения.

```
Command (m for help): d
Partition number (1,4,5, default 5): 4

Partition 4 has been deleted.

Command (m for help): p
Disk /dev/sdb: 1 GiB, 1073741824 bytes, 2097152 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x969824ff

Device      Boot Start    End Sectors  Size Id Type
/dev/sdb1          2048  411647  409600  200M 83 Linux

Command (m for help): █
```

Для удаления какого-то раздела:

```
sudo fdisk /dev/sdb
```

буква d – выбираем раздел – 4, enter, а потом p. Как видите, sdb5, который был внутри sdb4, удалился в том числе. Поэтому просто нажимаем q и ничего не сохраняем.

```
[user@centos8 ~]$ sudo fdisk -l /dev/sda
Disk /dev/sda: 20 GiB, 21474836480 bytes, 41943040 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xae357d6b

Device      Boot   Start     End   Sectors  Size Id Type
/dev/sda1    *      2048  2099199  2097152   1G 83 Linux
/dev/sda2          2099200 41943039 39843840  19G 8e Linux LVM
```

Чтобы увидеть, что у нас с разделами в системе:

```
sudo fdisk -l
```

Тут у нас и то, что мы создали только что, а также sda1 и sda2 - там где у нас хранится система. Если посмотреть внимательнее на sda2:

```
sudo fdisk -l /dev/sda
```

то можно увидеть, что его тип – Linux LVM. Про LVM мы ещё поговорим.

```
Disk: /dev/sdc
Size: 1 GiB, 1073741824 bytes, 2097152 sectors
Label: gpt, identifier: 02541524-4E41-2846-AD26-7B9A332308F9

Device      Start     End   Sectors  Size Type
>> /dev/sdc1    2048  411647  409600  200M Linux filesystem
  Free space  411648 2097118 1685471  823M

Partition UUID: 35832FEC-238D-1149-972C-1CA18C5E4117
Partition type: Linux filesystem (0FC63DAF-8483-4772-8E79-3D69D8477DE4)

[ Delete ] [ Resize ] [ Quit ] [ Type ] [ Help ] [ Write ] [ Dump ]
```

Через командную строку вы также можете воспользоваться псевдографической утилитой cfdisk:

```
sudo cfdisk /dev/sdc
```

Также есть и другие утилиты – к примеру parted. В конечном счёте, чем пользоваться – решаете вы. Но в целом, практически на всех системах бывают предустановлены fdisk и parted. На старых системах можно наткнуться на fdisk, который не поддерживает GPT, но сейчас это не так актуально. В любом случае, знать много инструментов – хорошо, но зная основы можно разобраться с любой утилитой.

```
[user@centos8 ~]$ lsblk
NAME      MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda        8:0    0   20G  0 disk 
└─sda1     8:1    0    1G  0 part /boot
└─sda2     8:2    0   19G  0 part
  └─cl_centos8-root 253:0  0   17G  0 lvm   /
  └─cl_centos8-swap 253:1  0    2G  0 lvm   [SWAP]
sdb        8:16   0    1G  0 disk 
└─sdb1     8:17   0  200M 0 part
└─sdb4     8:20   0    1K  0 part
└─sdb5     8:21   0 100M 0 part
sdc        8:32   0    1G  0 disk 
└─sdc1     8:33   0  200M 0 part
sr0       11:0   1 58.2M 0 rom   /run/media/user/VBox_GAs_6.1.14
```

Напоследок:

```
lsblk
```

покажет вам блочные устройства в древовидной форме, где прекрасно видно, что это за диск или раздел и к чему он относится.

```
[user@centos8 ~]$ ll -R /dev/disk/
/dev/disk/:
total 0
drwxr-xr-x. 2 root root 880 Feb 14 18:26 by-id
drwxr-xr-x. 2 root root 60 Feb 14 17:39 by-label
drwxr-xr-x. 2 root root 160 Feb 14 18:26 by-partuuid
drwxr-xr-x. 2 root root 240 Feb 14 18:26 by-path
drwxr-xr-x. 2 root root 120 Feb 14 17:39 by-uuid

/dev/disk/by-id:
total 0
lrwxrwxrwx. 1 root root 9 Feb 14 17:39 ata-VBOX_CD-ROM_VB2-01700376 -> ../../sr0
lrwxrwxrwx. 1 root root 9 Feb 14 18:19 ata-VBOX_HARDDISK_VB0ff5f117-1ebf0c14 -> ../../sdb
lrwxrwxrwx. 1 root root 10 Feb 14 18:19 ata-VBOX_HARDDISK_VB0ff5f117-1ebf0c14-part1 -> ../../sdb1
lrwxrwxrwx. 1 root root 10 Feb 14 18:19 ata-VBOX_HARDDISK_VB0ff5f117-1ebf0c14-part4 -> ../../sdb4
lrwxrwxrwx. 1 root root 10 Feb 14 18:19 ata-VBOX_HARDDISK_VB0ff5f117-1ebf0c14-part5 -> ../../sdb5
lrwxrwxrwx. 1 root root 9 Feb 14 17:39 ata-VBOX_HARDDISK_VB595fd559-360b38c9 -> ../../sda
```

Помните, названия дисков даются udev-ом и могут меняться? Однако у дисков и разделов могут быть свои идентификаторы, которые могут помочь вам опознать различные устройства. И это хорошо видно, если запустить команду:

```
ll -R /dev/disk
```

где по различным идентификаторам есть символические ссылки, указывающие на текущее название каждого диска и раздела. То есть, как бы не изменилась буква диска – sda, sdb или sdc – идентификатор

всегда будет один и тот же, и обращаясь к диску или разделу по идентификатору, вы всегда будете обращаться к нужному диску или разделу.

Получилось довольно много информации, хотя пока мы не затронули тему файловых систем и LVM. Попрактикуйтесь с разделами – создавайте их, удаляйте, используйте различные утилиты – как графические – тот же Disks и Gparted, так и терминальные – fdisk, cfdisk, parted. Всё это позволит вам лучше закрепить материал и в дальнейшем более уверенно чувствовать себя при работе с дисками и разделами.

2.22.2 Практика

Вопросы

1. Как посмотреть список подключенных дисков и cd устройств?
2. Почему названия дисков начинается с sd?
3. Что такое и чем отличается gpt от mbr?
4. Почему в MBR можно создать только 4 раздела и как это можно обойти?
5. Как посмотреть информацию о дисках?

Задания

1. Добавьте 2 диска по 1Гб к виртуалке. На первом диске создайте ТР MBR и 3 основных раздела по 50Мб, на втором ТР GPT и 2 раздела по 100Мб. Сохраните и покажите список разделов.
2. Добавьте на первом диске ещё 2 раздела по 100Мб, при этом не удаляя ранее созданные разделы.
3. Удалите со второго диска второй раздел.

2.23 23. Основы файловых систем

2.23.1 23. Основы файловых систем

```
[user@centos8 ~]$ sudo fdisk -l /dev/sdb
[sudo] password for user:
Disk /dev/sdb: 1 GiB, 1073741824 bytes, 2097152 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x969824ff

Device      Boot   Start     End Sectors  Size Id Type
/dev/sdb1            2048  411647  409600 200M 83 Linux
/dev/sdb4            411648 2097151 1685504 823M  5 Extended
/dev/sdb5            413696  618495  204800 100M 83 Linux
[user@centos8 ~]$ █
```

Теперь, когда у нас готовы диски и разделы, пора заняться файловыми системами. И хотя мы уже говорили о файловых системах, мы прошлись по ним поверхностью, теперь же копнём чуть глубже. Когда мы создавали разделы, мы заметили, что они имеют начало и конец, определённые секторами:

```
sudo fdisk -l /dev/sdb
```

То есть сектор – это физический адрес на диске. Причём, реальный, то есть физический размер секторов на современных жёстких дисках – 4 кибибайта или больше, да и сам термин больше относится к жёстким дискам, в тех же SSD вместо секторов – страницы. Тем не менее, в целях совместимости всё ещё можно работать с секторами размером в 512 байт, но лучше почитайте об этом по [ссылке](#).

```
[user@centos8 ~]$ stat file
  File: file
  Size: 15          Blocks: 8          IO Block: 4096   regular file
Device: fd00h/64768d  Inode: 52698249      Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/    user)  Gid: ( 1000/    user)
Context: unconfined_u:object_r:user_home_t:s0
Access: 2021-01-01 19:48:06.383587155 +0400
Modify: 2021-01-01 19:45:39.540211251 +0400
Change: 2021-01-01 19:45:39.540211251 +0400
 Birth: -
[user@centos8 ~]$ █
```

Так вот, к чему я веду. Файлы хранятся на жёстком диске в секторах. Допустим, файл у нас начинается в секторе с номером 10000 и заканчивается в секторе с номером 20000. Несложно запомнить, когда у нас один файл. А когда 10 файлов? 100? 1000? А если у нас файлы постоянно меняют размеры? Тут, конечно, нужна программа, которая и будет всё это отслеживать – где начинаются и заканчиваются файлы. Причём, если у нас вчера файл заканчивался на секторе 20000, после него мы записали другой файл, а теперь понадобилось увеличить первый файл – то он станет занимать сектора от 10 тысяч до 20, и от 30 тысяч до 35. То есть обязательно, чтобы все сектора файла находились рядом друг с другом. Кроме этого, нужно различать все эти файлы – поэтому для каждого файла есть свой номер:

```
stat file
```

Да и этого недостаточно – у каждого файла свои права, свой владелец и группа. Также неплохо было бы также знать, когда файл создали, когда изменили, когда в последний раз к нему обращались. Ну и файлы нужно как-то структурировать, а не показывать пользователю одну большую кучу файлов.

Всем этим и занимается файловая система. И, условно, её можно разделить на две составляющие: первая – это программа, которая будет решать, куда писать файлы, откуда их читать и изменять какие-то значения, относящиеся к файлу, например, права; а вторая – это сами данные о файлах, откуда программа берёт данные и где она их сохраняет. Первая – программная часть – это модуль ядра. Вторая – метаданные файловой системы, иноды и прочее – хранятся, зачастую, на самом диске или разделе, где эта файловая система, хотя возможны исключения. А вот где именно хранятся метаданные, иноды, и что они из себя представляют – в целом ответить не получится, потому что это сильно разнится от типа файловой системы.

С некоторыми типами файловых систем, скорее всего, вы уже знакомы – FAT32, exFAT, NTFS. И, возможно, вы знаете, что на FAT32 нельзя создать файл размером больше 4 ГиБ – просто потому что в метаданных для указания размера файла выделено 4 байта. Максимальное число, которое можно поместить в 4 байтах – 4 миллиарда. 4 миллиарда байт – 4 Гибайта. Примерно по таким же принципам файловая система может, допустим, ограничить длину имени файла. Хотя сейчас это не так актуально, но это пример того, как тип файловой системы может накладывать ограничения.

Но всегда есть разработчики, которые любят обходить ограничения и добавлять какой-то функционал.

Поэтому типов файловых систем много и для Linux есть много различных модулей для их поддержки – в том числе NTFS, FAT32 и т.п. Но мы с вами разбирали иноды, знаем, что там информация о правах на файлы, о владельце и группе файла – то есть стандартные UNIX-овые права. А в NTFS права реализованы по другому, поэтому, хоть Linux и работает с NTFS, но установить операционную систему на файловую систему NTFS не получится. Зато есть много других файловых систем, на которых всё работает как надо. Самые популярные для GNU/Linux – ext4 и xfs. В рабочей среде чаще всего вы будете иметь дело с ними.

Ещё в начале 90-ых была создана файловая система ext, которую стали использовать в Linux и с тех она постоянно развивалась. За годы работы эта файловая система доказала свою стабильность, а большинству пользователей от файловой системы другого и не надо. О максимальных размерах файловой системы и файлов можно не беспокоиться – там эксбибит и 16 тэбигбайт соответственно. У xfs также история начинается в начале 90-ых, сейчас её развивает компания RedHat и xfs по умолчанию стоит на дистрибутивах, основанных на RHEL. Давайте лучше поговорим о структуре файловой системы.

```
[user@centos8 ~]$ cat /sys/class/block/sdc/queue/physical_block_size
512
[user@centos8 ~]$ cat /sys/class/block/sdc/queue/logical_block_size
512
[user@centos8 ~]$ █
```

И так, мы говорили, что накопители – это блочные устройства и ядро работает с ними с помощью блоков данных фиксированной длины. Так вот, сектора являются минимальной физической единицей и называются физическими блоками, а их размер можно глянуть в файле:

```
cat /sys/class/block/sdc/queue/physical_block_size
```

Не нужно запоминать команды, это в целом информация для общего развития. А те самые блоки фиксированной длины, которыми оперирует ядро – это логические блоки:

```
cat /sys/class/block/sdc/queue/logical_block_size
```

Но, кроме этого, также и у файловых систем существует понятие блоков. Как мы выяснили раньше, секторам, в которых хранится файл, не обязательно находится рядом. В системе бывают тысячи файлов, какие-то растут, какие-то удаляются, появляется промежуток между секторами файла. Происходит так называемая фрагментация диска. И жёсткому диску для прочтения одного файла приходится совершать больше движений, от чего падает производительность. Когда таких файлов много – это беда. И если каждый сектор – это 512 байт, жёсткому диску придётся несладко, прыгая между кучей разделённых секторов. Правда современные файловые системы поступают чуть умнее, оставляя много свободного места между файлами, что позволяет не мешать файлы в кучу мелких секторов, давая возможность файлам расти, не пересекаясь с другими. И тем не менее, если объединять сектора в небольшие группы, называемые блоками, допустим по 8 секторов – т.е. 4 кибибайта, то диску будет проще. Да и дело не только в этом – файловой системе легче работать с блоками. Допустим, если секторов по 512 байт на больших дисках получается огромное количество, то файловой системе может не хватить адресов, в итоге не получится создать файловую систему и файл большого объёма. Но вот объединив сектора в блоки, пусть хоть по 64 кибибайта, получится во много раз увеличить допустимый размер диска и файла. Это как с блокнотом, давая номер каждой клеточке блокнота у вас просто перестанет помещаться номер клетки в вашей таблице. Зато нумеруя не клетки, а страницы, вы сможете использовать куда больше клеток.

Но у блоков есть свои недостатки. Файловая система хранит информацию в блоках, а значит 1 файл – минимум 1 блок. Если предположить, что средний размер одного блока – 4096 байт, то для хранения однобайтного файла вы потратите 4096 байт. Если у вас в системе большинство файлов мелкие, то уйма места не будет использоваться. В таких случаях лучше указывать размер блока минимальным – для ext4 это 1КиБ. В любом случае, куча мелких файлов – большой напряг для диска. Но сейчас,

в основном, файлы довольно большие, поэтому особо места вы не потеряете, если будете использовать 4 кибайтные блоки. А в случае работы с большими файлами можно выставить размер блоков побольше.

-i bytes-per-inode

Specify the bytes/inode ratio. **mke2fs** creates an inode for every bytes-per-inode bytes of space on the disk. The larger the bytes-per-inode ratio, the fewer inodes will be created. This value generally shouldn't be smaller than the blocksize of the filesystem, since in that case more inodes would be made than can ever be used. Be warned that it is not possible to change this ratio on a filesystem after it is created, so be careful deciding the correct value for this parameter. Note that resizing a filesystem changes the number of inodes to maintain this ratio.

Manual page **mke2fs(8)** line 296 (press h for help or q to quit)

Но мало хранить сам файл, нам ещё ведь нужно где-то хранить информацию о файле – иноду. И инода на ext4 весит 256 байт и больше. При этом, на разных типах файловых систем иноды создаются по разному – где-то они создаются динамически, когда это необходимо – так сделано в xfs, а где-то создаются сразу при создании файловой системы, как это сделано в ext4. Причём создаются они на каждые сколько-то байт, по умолчанию 16КиБ:

```
man mke2fs  
/ -i
```

Из этого можно извлечь пару фактов. Количество инод на ext4 – ограничено. Если у вас куча файлов, размером меньше 16 КиБ, то у вас может закончиться количество инод. И тогда, хотя и будет место на диске, использовать его вы не сможете – без свободных инод файл не создать. Но, во первых, по иноде на 16КиБ получается 64 иноды на один мебибайт места, т.е. на реальных дисках вы сможете создать миллионы файлов. Во вторых, вы это значение можете изменить, сделать количество байт на инод больше или меньше. С одной стороны, уменьшив количество байт на иноду, вы сможете создавать больше файлов. С другой – посчитайте, на каждый мебибайт места вы тратите 16 КиБ на иноды, так как каждая инода весит 256 байт, а их 64. Т.е. на каждый гигабайт – 16 мегабайт чисто на иноды. На один терабайт – 16 Гигабайт. Ощущимо, правда? И это ещё при стандартном раскладе. Если у вас файлы большие, то можно и увеличить количество байт на иноду, но всё это делается при создании файловой системы. Поэтому такие вещи надо продумывать заранее – допустим, если вам нужен почтовый сервер, где будут миллионы писем – рассчитайте заранее, хватит ли вам инод? Ну или используйте xfs, тогда количество инод вас не будет беспокоить, так как там иноды выделяются динамически.

Можно долго говорить про структуру – сами блоки объединяются в группы блоков; в нулевой группе и некоторых других группах есть суперблок, где хранится информация о самой файловой системе, также во всех группах содержатся дескрипторы групп, таблицы инод и много чего другого, но... хотя эта тема интересная, но она будет актуальна только для ext4. Возможно, другие файловые системы работают со схожими структурами, но углубляться в эти структуры у нас сейчас нет необходимости. Моя цель была показать вам, что из себя представляют файловые системы, если вам интересно, посмотрите ссылки на структуру ext4 и xfs.

Понимание структуры файловой системы поможет вам лучше понять их работу и различия, но пока вы изучаете азы, лезть в такие дебри не обязательно. Я знаю, что многим интересна тема [различия файловых систем](#), но о поверхностных различиях можно найти на всяких сайтах, а чтобы понять детали нужно разобрать многие термины и технологии, что займёт много времени и пока не критично для основ. Однако кое-что мы всё же разберём сейчас – журналирование.

```

Installed:
  lm_sensors-libs-3.4.0-21.20180522git70f7e08.el8.x86_64
  sysstat-11.7.3-5.el8.x86_64

Complete!
[user@centos8 ~]$ iostat 1
Linux 4.18.0-193.19.1.el8_2.x86_64 (centos8)      02/21/2021      _x86_64_   (1 CPU)

avg-cpu: %user   %nice  %system %iowait  %steal   %idle
          0.24     0.00    0.62    0.19     0.00    98.95

Device      tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
sda       5.00    157.14      30.66    1002040    195501
sdb       0.04      1.49       0.00      9482        0
sdc       0.03      0.99       0.00      6296        0
scd0      0.00      0.01       0.00       92        0
dm-0       4.87    149.43      23.44    952871    149475
dm-1       2.35      1.39       8.30      8872    52908

```

Вы, возможно, знаете – нельзя вытаскивать флешку на ходу – нужно её сначала программно «отсоединить». Ну или допустим, нельзя выключать компьютер сразу из розетки, и в целом резкое выключение электричества может привести к печальным последствиям. Если говорить о дисках, то в операционной системе сотни процессов и часть из них постоянно пишет данные на файловую систему, всякие логи или временные файлы, не говоря о серверах, которые могут писать какие-то важные данные. Чтобы видеть информацию, как сейчас используется диск, можно глянуть утилиту iostat:

```

sudo dnf install sysstat
iostat 1

```

Но операции при работе с файлами – их создание или удаление – требуют от файловой системы нескольких операций – создание жёсткой ссылки, изменение информации в иноде, записывание данных на диск и т.п. Но что, если при создании или удалении файла выполнится только часть этих операций, а потом компьютер резко выключится или вы вытащите флешку? В итоге у вас может появится жёсткая ссылка, указывающая на иноду, в которой нет никаких данных. Или скажем запишется только часть данных, а иногда будет указывать на незанятые блоки, в итоге место будет недоступно. Когда у вас пишутся сотни и тысячи файлов – это создаст кучу ошибок в файловой системе. И если флешку можно не выдергивать, то вот полностью исключить резкое выключение компьютера невозможно. Но если писать все операции с файлами предварительно в специальное место, называемое журналом, а уже потом применять на диск – то файловая система после сбоя всегда может посмотреть в журнале, а что там недописалось и подчистить всякие ошибки при включении. Да и журналировать можно не только операции и метаданные, но и сами файлы, но это зависит от типа и настроек файловой системы. В любом случае, наличие журнала позволяет смягчить урон при резкой потере связи с диском. И журналирование есть в большинстве современных файловых систем.

```
[user@centos8 ~]$ lsmod | grep xfs
xfs                      1519616  1
libcrc32c                16384   3 nf_conntrack,nf_nat,xfs
[user@centos8 ~]$ modinfo ext4 | head
filename:      /lib/modules/4.18.0-193.19.1.el8_2.x86_64/kernel/fs/ext4/ext
4.ko.xz
softdep:       pre: crc32c
license:       GPL
description:   Fourth Extended Filesystem
author:        Remy Card, Stephen Tweedie, Andrew Morton, Andreas Dilger, T
heodore Ts'o and others
alias:         fs-ext4
alias:         ext3
alias:         fs-ext3
alias:         ext2
alias:         fs-ext2
[user@centos8 ~]$ sudo dnf install -y ntfs-3g
```

Ну и напоследок. Как я сказал, программная часть файловых систем выполнена в виде модулей ядра и их можно увидеть с помощью lsmod:

```
lsmod | grep xfs
modinfo ext4 | head
```

Но для работы с некоторыми файловыми системами – допустим, NTFS – надо устанавливать дополнительные пакеты:

```
sudo dnf install -y ntfs-3g
```

Подводя итоги. Для начала мы выяснили, зачем вообще нужны файловые системы. Также затронули тему блоков – физических, логических и блоков файловой системы. Поговорили о том, к чему приводят различия в структурах – как, например, различия в максимальных размерах файлов и блоков. Затронули одну из функций файловой системы – журналирование. На самом деле, мы не слишком углубились в детали, потому что их очень, очень много, но, кое-какое представление о файловых системах у нас сложилось.

2.23.2 Практика

Вопросы

1. Чем отличается блок файловой системы от сектора?
2. Для чего нужны блоки?
3. Чем отличается xfs от ext4?
4. Чем обусловлено максимальное количество файлов в файловой системе?
5. Как посмотреть использование диска?

2.24 24. Работа с файловыми системами

2.24.1 24. Работа с файловыми системами

```
[user@centos8 ~]$ head /etc/mke2fs.conf
[defaults]
    base_features = sparse_super,large_file,filetype,resize_inode,dir_index,
ext_attr
    default_mntopts = acl,user_xattr
    enable_periodic_fsck = 0
    blocksize = 4096
    inode_size = 256
    inode_ratio = 16384
```

Кое-какое представление о файловых системах у нас сложилось, пора уже приступать к делу. Для начала, давайте запишем файловую систему. В этом нам поможет утилита mkfs. Всякие настройки по умолчанию для файловой системы ext4 лежат в файле /etc/mke2fs.conf:

```
head /etc/mke2fs.conf
```

Тут у нас размеры блоков, инод и всё такое.

```
[user@centos8 ~]$ sudo mkfs.ext4 /dev/sdb1
[sudo] password for user:
mke2fs 1.45.4 (23-Sep-2019)
Creating filesystem with 204800 1k blocks and 51200 inodes
Filesystem UUID: 555a6e59-83e6-4331-b55e-c837fdfc1a22
Superblock backups stored on blocks:
    8193, 24577, 40961, 57345, 73729

Allocating group tables: done
Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done
```

И всё что нам остаётся сделать для создания файловой системы – написать команду mkfs точка, желаемый тип файловой системы, а потом раздел или диск:

```
sudo mkfs.ext4 /dev/sdb1
```

Программа создаст файловую систему и покажет нам информацию о ней – сколько блоков какого объёма поместились в файловой системе, какое количество инод, идентификатор файловой системы, где сохранились бэкапы суперблока - того самого, где хранятся метаданные самой файловой системы.

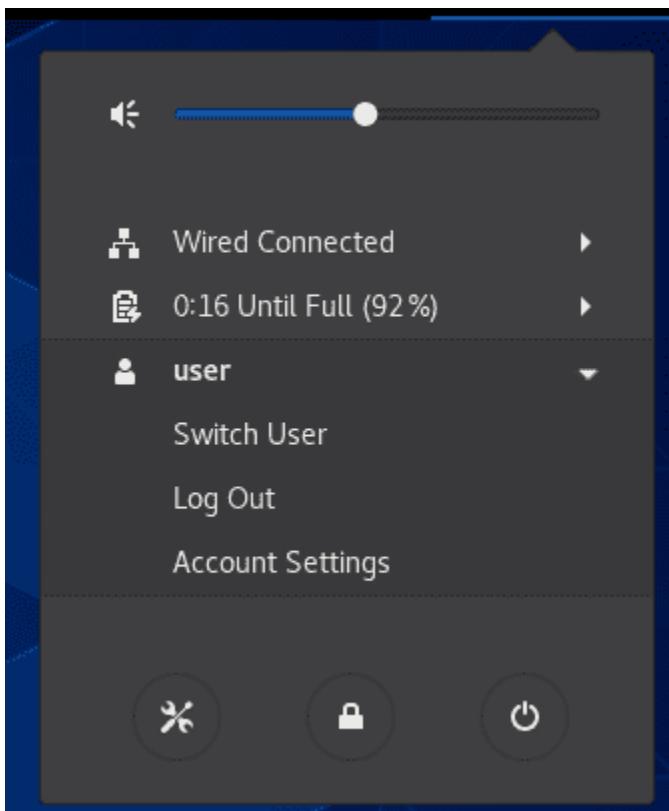
```
[user@centos8 ~]$ sudo tune2fs -l /dev/sdb1
tune2fs 1.45.4 (23-Sep-2019)
Filesystem volume name: <none>
Last mounted on: <not available>
Filesystem UUID: 555a6e59-83e6-4331-b55e-c837fdfc1a22
Filesystem magic number: 0xEF53
Filesystem revision #: 1 (dynamic)
Filesystem features: has_journal ext_attr resize_inode dir_index filetype extent
                     64bit flex_bg sparse_super large_file huge_file dir_nlink extra_isize metadata_csum
Filesystem flags: signed_directory_hash
Default mount options: user_xattr acl
Filesystem state: clean
Errors behavior: Continue
Filesystem OS type: Linux
Inode count: 51200
Block count: 204800
```

Всякие детали о файловой системе можно посмотреть с помощью `tune2fs` и указанием раздела с нужной файловой системой:

```
sudo tune2fs -l /dev/sdb1
```

Но в большинстве случаев с этим всем не нужно заморачиваться. По умолчанию всё настроено так, чтобы работать из коробки для большинства случаев. А если файловая система вам нужна для каких-то более серьёзных задач, то нужно изучать документацию. Пойдём дальше. Ну, есть у нас файловая система, а что с ней делать? Её нужно куда-то прицепить, точнее, примонтировать в какую-то директорию. Собственно, всё зависит от ваших задач. Задачи могут быть разные – скажем, отделить файлы пользователей от системы, чтобы можно было переустановить операционную систему, не стерев ваши файлы. Тут нужно отделить директорию `/home`. Или, допустим, вы хотите на сервере отделить файлы с данными от файлов операционной системы. Или, например, отделить директорию с логами, чтобы в случае какой-нибудь проблемы они внезапно не забили весь диск логами, от чего система вообще может перестать нормально работать. И это только ряд задач, на деле причин может быть много. Но, давайте, например, вынесем файлы пользователей на новую файловую систему.

Если я возьму и просто примонтирую новую файловую систему в `/home`, то я перекрою существующие файлы пользователей и вместо них будет чистая файловая система и директория `/home` окажется пустой. Это не дело, нужно сначала перенести файлы пользователей на новую файловую систему. А это не такая простая задача – если я сейчас сижу от имени пользователя, пусть даже не видно, но я работаю с файлами в домашней директории своего пользователя. Т.е. графический интерфейс подгрузил всякие файлы настроек и всячески использует мою домашнюю директорию, например, браузер может писать кэш, баш сохраняет историю и т.п. А пока файлы используются, перемещать их не стоит – можно случайно повредить их. Поэтому нужно завершить все процессы, использующие директорию `/home` и все поддиректории.



```

CentOS Linux 8 (Core)
Kernel 4.18.0-193.19.1.el8_2.x86_64 on an x86_64

Activate the web console with: systemctl enable --now cockpit.socket

centos8 login: root
Password:
Last login: Sun Feb 28 12:07:20 on tty3
[root@centos8 ~]# lsof +D /home/
[root@centos8 ~]# 

```

Во первых, для этого мне нужно разлогиниться. Но как мне тогда работать? Правильно, через виртуальный терминал. Нажимаем правый Ctrl+F3 (это зависит от гипервизора, на реальном железе Alt+Ctrl+F3) и заходим в виртуальный терминал из под рута. Дальше убеждаемся, что никакой процесс не использует файлы в директории /home, в этом нам поможет утилита lsof, показывающая открытые файлы, с ключом +D, чтобы проверить во всех субдиректориях:

```
lsof +D /home
```

У меня тут пусто, но если у вас какие-то процессы используют файлы, желательно их завершить, вы знаете как.

```
0      /home/shared/dir8
0      /home/shared
140M   /home/
[root@centos8 ~]# fdisk -l /dev/sdb1
Disk /dev/sdb1: 200 MiB, 209715200 bytes, 409600 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
[root@centos8 ~]#
```

Дальше стоит проверить размер директории /home – хотя надо было ещё при создании разделов. Для этого используем утилиту du:

```
du -h /home
```

Как видите, тут файлов на 140 мегабайт, а наш раздел sdb1 почти на 200:

```
fdisk -l /dev/sdb1
```

Хоть тут и хватает места, у меня дальше начинается другой раздел, а значит в будущем я не смогу увеличить размер раздела и файловой системы, по крайней мере без кучи махинаций. Поэтому лучше изначально разделы распределить так, чтобы у каждой файловой системы хватало места с запасом. Но при этом, если нет необходимости, не занимайте всё свободное место на раздел – увеличить раздел и файловую систему всегда можно, если есть свободное место, а вот уменьшить размер файловой системы зачастую проблемно, а на xfs пока вообще нет такой возможности.

```
[root@centos8 ~]# fdisk -l /dev/sdc2
Disk /dev/sdc2: 823 MiB, 862961152 bytes, 1685471 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
[root@centos8 ~]# mkfs.ext4 /dev/sdc2
mke2fs 1.45.4 (23-Sep-2019)
Creating filesystem with 210683 4k blocks and 52752 inodes
Filesystem UUID: 3ef49437-a116-43bb-9152-1cef77fc8473
Superblock backups stored on blocks:
      32768, 98304, 163840

Allocating group tables: done
Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done
```

Но у меня есть раздел sdc2 на 800 мегабайт:

```
fdisk -l /dev/sdc2
```

его и будем использовать. Опять же, нам нужна файловая система:

```
mkfs.ext4 /dev/sdc2
```

```
[root@centos8 ~]# mount /dev/sdc2 /mnt/
[root@centos8 ~]# df -h
Filesystem           Size  Used Avail Use% Mounted on
/devtmpfs             886M    0  886M   0% /dev
tmpfs                914M    0  914M   0% /dev/shm
tmpfs                914M  9.2M  905M  1% /run
tmpfs                914M    0  914M   0% /sys/fs/cgroup
/dev/mapper/cl_centos8-root  17G  5.5G  12G  32% /
/dev/sda1              976M 272M  638M  30% /boot
tmpfs                183M  1.2M  182M  1% /run/user/42
tmpfs                183M  4.0K  183M  1% /run/user/0
/dev/sdc2              795M  1.7M  736M  1% /mnt
[root@centos8 ~]#
```

Теперь мне нужно перенести файлы из директории `/home` на новую файловую систему. А чтобы это сделать, мне нужно временно примонтировать новую файловую систему. Для таких целей даже есть специальная директория - `/mnt`. Я просто пишу:

```
mount /dev/sdc2 /mnt/
```

т.е. какую файловую систему монтирую в какую директорию. Чтобы убедиться, что всё сработало, я использую утилиту `df`:

```
df -h
```

Самая последняя запись - `/dev/sdc2` примонтирован в `/mnt`.

```
[root@centos8 ~]# mv /home/* /mnt/
[root@centos8 ~]# ll /mnt/
total 32
drwxr-xr-x. 3 root  root  4096 Jan 10 19:44 company
drwx----- 2 root  root 16384 Feb 28 12:23 lost+found
drwxrws--T. 3 root group1 4096 Jan 11 01:44 shared
drwx----- 19 user user  4096 Feb 28 12:05 user
drwx----- 3 user3 user3  4096 Jan 10 19:39 user3
[root@centos8 ~]# du -h /mnt | tail -1
141M  /mnt
[root@centos8 ~]# umount /mnt
[root@centos8 ~]#
```

Окей, пора переносить файлы:

```
mv -v /home/* /mnt
```

Проверяем, всё ли нормально:

```
ll /mnt /home
du -h /mnt | tail -1
```

Вроде всё ок – файлы перемещены. Теперь пора отмонтировать новую файловую систему и примонтировать в `/home`. Чтобы отмонтировать - `umount /mnt`.

```
[root@centos8 ~]# cat /etc/fstab
#
# /etc/fstab
# Created by anaconda on Fri Oct 2 03:23:00 2020
#
# Accessible filesystems, by reference, are maintained under '/dev/disk/'.
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info.
#
# After editing this file, run 'systemctl daemon-reload' to update systemd
# units generated from this file.
#
/dev/mapper/cl_centos8-root / xfs defaults 0 0
UUID=c607eb50-9842-4b18-ba8e-2d1139960594 /boot ext4 defaults 1 2
/dev/mapper/cl_centos8-swap swap defaults 0 0
[root@centos8 ~]# blkid /dev/sdc2
/dev/sdc2: UUID="3ef49437-a116-43bb-9152-1cef77fc473" TYPE="ext4" PARTUUID="051ca55c-40a6-ff47-056aff3d21cc"
[root@centos8 ~]#
```

Но прежде чем пойдём дальше, есть ещё одна деталь. То как мы примонтировали сейчас, не означает, что после перезагрузки операционная система также примонтирует. Ей нужно об этом сказать, чтобы в будущем она всегда монтировала sdc2 в /home. Для этого есть файл /etc/fstab:

```
cat /etc/fstab
```

И в нём нам нужно достоверно компьютеру сказать, какую файловую систему куда нужно монтировать. А так как мы выяснили, что sda sdb и sdc – имена, которые могут меняться, то нельзя указывать их. Вместо них мы можем использовать идентификатор файловой системы. Помните, при создании мы видели такой? И чтоб не искать его, можем воспользоваться утилитой blkid:

```
blkid /dev/sdc2
```

Второй столбик – UUID – это наш идентификатор файловой системы. Если мы форматнём эту файловую систему, будет другой UUID.

```
[root@centos8 ~]# blkid /dev/sdc2 | cut -d' ' -f2
UUID="3ef49437-a116-43bb-9152-1cef77fc473"
[root@centos8 ~]# cp /etc/fstab{,.bkp}
[root@centos8 ~]# ls /etc/fstab*
/etc/fstab /etc/fstab.bkp
[root@centos8 ~]# blkid /dev/sdc2 | cut -d' ' -f2 >> /etc/fstab
[root@centos8 ~]#
```

И теперь нам нужно переписать этот UUID в файл /etc/fstab. Но.. тут же много символов? Хотя... мы же знаем, что у нас в системе куча утилит для работы с текстом. Воспользуемся одной из них – cut. И так, blkid делит столбцы по пробелам, значит пробел наш делитель. При этом, нам нужен второй столбец. Значит:

```
blkid /dev/sdc2 | cut -d' ' -f2
```

И мы получили нужную строку. Теперь нужно записать её в файл /etc/fstab. Но прежде чем делать эксперименты с реальными файлами, лучше их забэкапить:

```
cp /etc/fstab{,.bkp}
ls /etc/fstab*
```

Теперь выполняем команду blkid, но вывод добавляем в fstab:

```
blkid /dev/sdc2 | cut -d' ' -f2 >> /etc/fstab
```

Тут будьте очень внимательными, обязательно два символа больше, чтобы не перезаписать файл.

```
GNU nano 2.9.8          /etc/fstab          Modified

#
# /etc/fstab
# Created by anaconda on Fri Oct  2 03:23:00 2020
#
# Accessible filesystems, by reference, are maintained under '/dev/disk/'.
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info.
#
# After editing this file, run 'systemctl daemon-reload' to update systemd
# units generated from this file.
#
/dev/mapper/cl_centos8-root  xfs    defaults        0 0
UUID=c607eb50-9842-4b18-ba8c-d1139960594  /boot      ext4    defaults        1 2
/dev/mapper/cl_centos8-swap  swap   defaults        0 0
UUID="3ef49437-a116-43bb-9152-1cef77fcb473"  /home     ext4    defaults,noatime  0  1
```

Теперь осталось дописать строку:

```
nano /etc/fstab
```

И так, файловая система указана. Можно ориентироваться на примеры выше. После UUID-а указываем точку монтирования - /home. Количество пробелов не важно. Дальше указываем тип файловой системы – ext4. После этого – опции монтирования. Допустим, чтобы в inode не писалась информация о последнем доступе к файлу – это сильно сократит частоту записи на жёсткий диск – напишем defaults,noatime, через запятую, без пробела. А по умолчанию можно оставить defaults. На самом деле, defaults тут чтобы поле не пустовало, мы могли бы убрать defaults и оставить только noatime, но я решил оставить, чтобы показать пример нескольких опций. По [ссылке](#) вы можете найти различные опции монтирования, там же можно найти информацию, что включает в себя defaults. Дальше две цифры. Первая – dump – для создания резервной копии файловой системы. Хотя для работы этого нужна утилита dump, которая сейчас практически не пользуются. Но, в целом, есть такая возможность, и если стоит 1, то будет сниматься дамп, а в большинстве случаев здесь указывается 0 – т.е. без дампа. Вторая цифра – проверка файловой системы.

Во время работы на файловой системе могут накапливаться ошибки, и всякие внештатные ситуации, как, например, при внезапном отключении компьютера, могут создавать ошибки. Да, журналирования частично решает проблемы, но не полностью. Поэтому файловые системы желательно иногда проверять на наличие ошибок и исправлять, для этого есть утилита fsck. Но проверять целостность файловой системы во время работы нельзя, это может её повредить. Её нужно предварительно отмонтировать. Но это не всегда просто – как вы отмонтируете корень, где у вас система? Да и чтобы не заниматься этим вручную, это можно автоматизировать, чтобы при включении компьютера конкретная файловая система проверялась на ошибки и они исправлялись. Тут значение может быть 0 – это не проверять. 1 стоит ставить только для файловой системы, где лежит корень, так как исправление ошибок на нём более приоритетно. Для проверки всех остальных файловых систем можно указать 2. После чего сохраним и выходим.

```
[user@centos8 ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        886M    0  886M  0% /dev
tmpfs          914M    0  914M  0% /dev/shm
tmpfs          914M  9.6M  904M  2% /run
tmpfs          914M    0  914M  0% /sys/fs/cgroup
/dev/mapper/cl_centos8-root  17G  5.3G  12G  32% /
/dev/sda1       976M 272M  638M 30% /boot
/dev/sdc2       795M 142M  595M 20% /home
tmpfs          183M  1.2M  182M  1% /run/user/42
tmpfs          183M  9.1M  174M  5% /run/user/1000
/dev/sr0         59M   59M    0 100% /run/media/user/VBox_GA
s_6.1.14
```

И теперь, когда система знает куда монтировать sdc2, можно просто написать:

```
mount /dev/sdc2
```

или

```
mount /home
```

Теперь можем вернуться к графическому интерфейсу – Ctrl+F1 и залогиниться. Опять напишем df -h и посмотрим, что куда примонтировано. Как видите, файловая система sdc2 примонтирована в директорию /home. И если бы что-то не работало, мы бы просто не смогли залогиниться.

И так, при включении компьютера операционная система смотрит в fstab, видит какую файловую систему куда и как монтировать и делает это. Но что, если файловая система недоступна? Допустим, какие-то проблемы с диском или самой файловой системой, а может опечатка в fstab. В таких случаях операционная система не загружается полностью, а предлагает нам посмотреть и исправить ошибку. Игнорировать проблемы с файловой системой и загрузиться операционная система не может, если это не указано в fstab. Представьте, что на этой файловой системе есть какие-то важные файлы, допустим файлы приложения. И если у нас система запустится, запустит приложение, но оно не увидит свои файлы – приложение может создать кучу проблем. Во избежание такого операционная система не даст запустить всю систему, если ей не удаётся примонтировать какую-то файловую систему.

```
# Created by anaconda on Fri Oct 2 03:23:00 2020
#
# Accessible filesystems, by reference, are maintained under '/dev/di
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for mor
#
# After editing this file, run 'systemctl daemon-reload' to update sy
# units generated from this file.
#
/dev/mapper/cl_centos8-root / xfs defaults
UUID=c607eb50-9842-4b18-ba8e-2d1139960594 /boot ext4
/dev/mapper/cl_centos8-swap swap defaults
UUID=asmdkasdm"3ef49437-a116-43bb-9152-1cef77fcb473" /home ext4
```

Давайте воссоздадим такую ситуацию и попытаемся её решить. Для этого давайте испортим запись в fstab:

```
sudo nano /etc/fstab
```

Добавим здесь перед UUID какие-нибудь буквы, типа мы опечатались, и перезагрузимся:

```
reboot
```

```
-- The start-up result is done.
Feb 28 13:13:11 centos8 kernel: EXT4-fs (sda1): mounted filesystem with ordered data
Feb 28 13:14:38 centos8 systemd[1]: dev-disk-by\x2duuid-asmdkasdm\x5cx223ef49437\x2du
Feb 28 13:14:38 centos8 systemd[1]: Timed out waiting for device dev-disk-by\x2du
-- Subject: Unit dev-disk-by\x2duuid-asmdkasdm\x5cx223ef49437\x2da116\x2d43bb\x2d9152\x2d10
-- Defined-By: systemd
-- Support: https://access.redhat.com/support
-- Unit dev-disk-by\x2duuid-asmdkasdm\x5cx223ef49437\x2da116\x2d43bb\x2d9152\x2d10
-- 
-- The result is timeout.
Feb 28 13:14:38 centos8 systemd[1]: Dependency failed for /home.
-- Subject: Unit home.mount has failed
-- Defined-By: systemd
-- Support: https://access.redhat.com/support
-- 
-- Unit home.mount has failed.
-- 
-- The result is dependency.
Feb 28 13:14:38 centos8 systemd[1]: Dependency failed for Local File Systems.
-- Subject: Unit local-fs.target has failed
```

Как видите, система не смогла запуститься и предложила нам командную строку. Во первых, вводим пароль от пользователя root. Чуть выше командная строка рекомендует посмотреть логи:

```
journalctl -xb
```

Немного полистаем вниз и увидим красную строчку – Failed to mount /home – т.е. не получилось примонтировать /home. Чуть выше написана причина – не найдено устройство.

```
[root@centos8 ~]# fsck /dev/sdc2
fsck from util-linux 2.32.1
e2fsck 1.45.4 (23-Sep-2019)
/dev/sdc2: clean, 963/52752 files, 43799/210683 blocks
[root@centos8 ~]# _
```

Но прежде чем исправлять, представим, что дело не в нашей опечатке, а какая-то проблема с файловой системой. Для этого выполним проверку файловой системы с помощью утилиты fsck:

```
fsck /dev/sdc2
```

А для файловой системы xfs используется утилита xfs_repair. Сейчас у меня ошибок нет, но если бы были, программа бы о них сказала и предложила бы пути решения.

```
[user@centos8 ~]$ ll /
total 28
lrwxrwxrwx.  1 root root    7 May 11  2019 bin -> usr/bin
dr-xr-xr-x.  6 root root 4096 Oct  2 12:36 boot
drwrxr-xr-x. 20 root root 3380 Feb 28 13:42 dev
drwrxr-xr-x. 134 root root 8192 Feb 28 12:43 etc
drwrxr-xr-x.  7 root root 4096 Feb 28 12:30 home
lrwxrwxrwx.  1 root root    7 May 11  2019 lib -> usr/lib
lrwxrwxrwx.  1 root root    9 May 11  2019 lib64 -> usr/lib64
drwrxr-xr-x.  2 root root    6 May 11  2019 media
drwrxr-xr-x.  2 root root    6 May 11  2019 mnt
drwrxr-xr-x.  3 root root   39 Oct  2 12:44 opt
dr-xr-xr-x. 268 root root     0 Feb 28 13:42 proc
dr-xr-x---.  5 root root 238 Feb 28 12:08 root
drwrxr-xr-x. 42 root root 1260 Feb 28 13:44 run
lrwxrwxrwx.  1 root root    8 May 11  2019 sbin -> usr/sbin
drwrxr-xr-x.  2 root root    6 May 11  2019 srv
dr-xr-xr-x. 13 root root     0 Feb 28 13:42 sys
drwrxrwxrwt. 20 root root 4096 Feb 28 13:44 tmp
drwrxr-xr-x. 12 root root 144 Oct  2 11:24 usr
drwrxr-xr-x. 22 root root 4096 Jan 10 19:56 var
```

Также, у многих возникает вопрос – а какие директории стоит выносить на отдельные файловые системы:

11 /

Идеальной схемы нету, где-то что-то имеет смысл отделять, а где-то это просто усложняет. Мы разобрали, зачем это для /home. Вскоре мы разберём, зачем это может понадобится для /boot. Какие-то директории, например dev, proc и sys – мы их уже разбирали – в них и так виртуальные файловые системы. В /etc важные для загрузки операционной системы настройки, тот же fstab – и вынести /etc на отдельный раздел очень сложно и бесполезно. /media и /mnt обычно пустые директории, куда временно монтируются всякие флешки и cd диски, поэтому их выносить бессмысленно. В /opt обычно лежит проприетарный софт, на домашних компьютерах его выносить на отдельный раздел обычно не нужно, а на серверах сильно зависит от самого софта – иногда, для удобства, стоит отделять файлы приложения от файлов системы – это позволит легче бэкапить приложение, переносить его на новые системы и т.д. Это в целом касается приложений на серверах, которые могут использовать в качестве директории с данными /opt, /home, а то и отдельную директорию в корне, например /srv. Домашнюю директорию рута - /root – выносить бессмысленно, там обычно ничего примечательного. В /run – еще одна виртуальная файловая система. Про /tmp мы упоминали, тут у нас всякие процессы и пользователи могут создавать временные файлы при работе. И в большинстве случаев имеет смысл вынести /tmp в виртуальную файловую систему – можете почитать по [ссылке](#). Касаемо /usr всё сложно – раньше имело смысл вынести /usr, чтобы сделать его read only – то есть доступным только для чтения. В таких случаях во время обновления или установки программ её перемонтировали для записи, а потом обратно в read only, что позволяло обеспечить безопасность, так как если нет возможности изменить файлы – то всяким вирусам будет посложнее. Но в современных дистрибутивах происходят изменения и вынести /usr если и можно, то всё равно не рекомендуется. А read only можно сделать весь корень, при определённых условиях. В директории /var лежат динамические файлы, которые меняются, например логи, файлы базы данных, сайты и т.п. В этом и суть директории – если вы хотите обезопасить систему и сделать её read only, то директорию, в которой что-то постоянно меняется стоит вынести в

отдельную файловую систему. При этом монтировать `/var` можно с определёнными опциями, например `noexec` – чтобы нельзя было запускать программы в этой файловой системе. Но тут, конечно, нюансы – у вас тут может лежать сайт в директории `/var/www`, где может требоваться запускать какие-то файлы сайта. Поэтому имеет смысл выносить не весь `/var`, а его определённые субдиректории – например, `/var/log`, где лежат логи. В целом рекомендуется выносить директорию `/var/log` на отдельную файловую систему, потому что бывают случаи, когда какие-то сервисы начинают внезапно писать очень много логов. И если `/var/log` вместе с корнем в одной файловой системе, то когда закончится место на файловой системе, начнут появляться проблемы.

Всё это – базовые операции при работе с файловыми системами – мы знаем, как её создать, как её примонтировать, как сделать так, чтобы она монтировалась при включении компьютера. И мы для примера переместили `/home` на раздел одного из дисков. И, хотя, казалось бы, в целом проделать это всё не сложно, но когда речь идёт о терабайтах данных, все эти махинации могут занять много времени, при этом сервисы, использующие эти данные, будут недоступны для пользователей. Время – это деньги, и пока вы будете перемещать данные на новые диски большего объёма, бизнес будет терять деньги. Поэтому админ должен заранее определить – сколько места понадобится на те или иные файловые системы, какие из них в дальнейшем могут потребовать больше места, какого рода файлы будут храниться – большие или маленькие, в зависимости от этого подкорректировать размеры блоков файловой системы и рассчитать необходимое количество инод. И всё это сделать ещё на этапе установки операционной системы. Но как бы идеально вы не рассчитали, в жизни не всё идёт по плану, поэтому есть более гибкие системы, о которых мы поговорим в следующий раз.

2.24.2 Практика

Вопросы

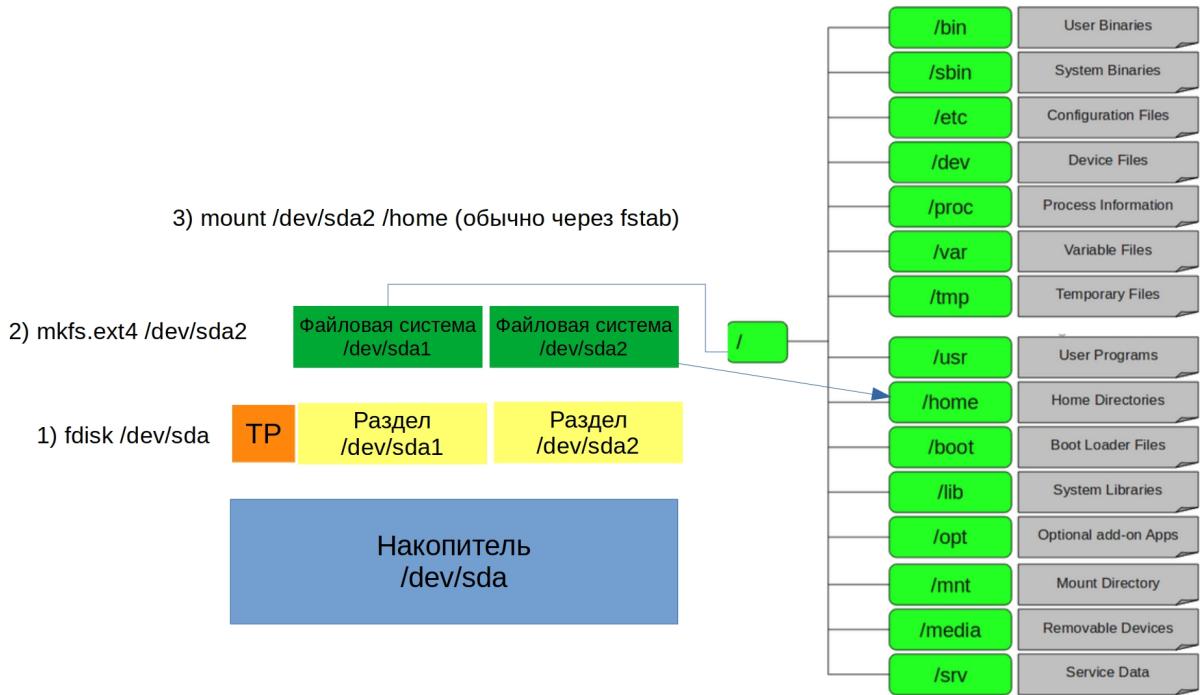
1. Как увидеть детальную информацию о файловой системе?
2. Зачем нужно монтировать файловые системы?
3. Как посмотреть список примонтированный фс?
4. Зачем нужен `fstab`?
5. Как узнать UUID файловой системы и зачем он нужен?
6. Как проверить файловую систему на ошибки?

Задания

1. Создайте директорию `~/myfiles`, скопируйте туда `/etc/passwd` и `/etc/group`. Создайте раздел с ФС `ext4`. Перенесите данные из директории `myfiles` на эту файловую систему. Настройте автомонтирование этой файловой системы в эту директорию.
2. Создайте директорию `~/mybackup`. Создайте раздел с ФС `xfs`, временно примонтируйте ФС в директорию `mybackup` и перенесите данные из `~/myfiles` на эту файловую систему. Отмонтируйте ФС.

2.25 25. Управление логическими томами - LVM

2.25.1 25. Управление логическими томами - LVM



Стандартную схему работы с дисками и файловыми системами мы разобрали – создаём раздел с помощью fdisk, создаём на разделе файловую систему с помощью mkfs и монтируем файловую систему куда нужно.

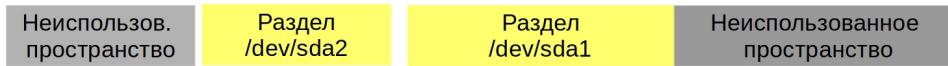
Но представьте себе ситуацию, когда у вас на каком-то разделе перестало хватать места. Если этот раздел крайний и после него есть свободное место – то его можно увеличить. А что если раздел не крайний, если после него есть ещё?

Нужно увеличить sda1

- Переносим все блоки sda1 в свободное пространство



- Увеличиваем sda1



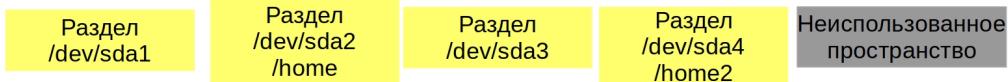
Занимает много времени и требует остановки работы



Стандартные таблицы разделов в этом плане ограничены, для раздела указывается только первый и последний сектор, то есть нельзя указать, чтобы раздел начался в одном месте, в другом закончился, а потом в третьем продолжился.

Нужно больше места для файлов пользователей

- Создаём sda4, создаём файловую систему и монтируем в /home2



- Переносим домашние директории некоторых пользователей в /home2

`usermod user10 -m -d /home2/user10`

Придётся держать часть файлов там, часть файлов тут, неудобно и некрасиво

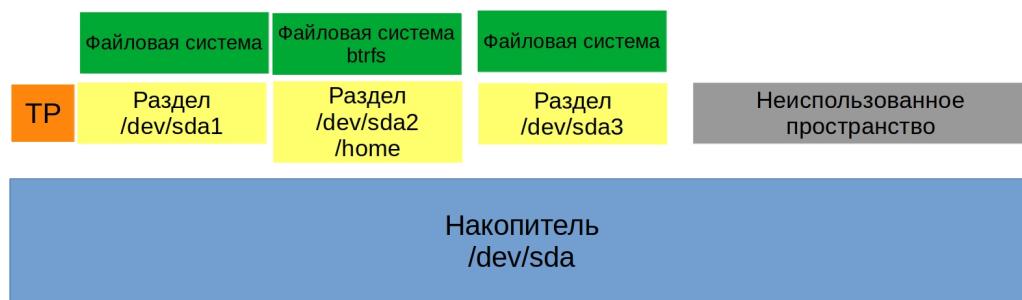


В некоторых случаях это не проблема – можно создать ещё один раздел, на нём поднять новую файловую систему и примонтировать куда-то. Допустим, создать директорию /home2, примонтировать туда новую файловую систему и перенести домашние директории некоторых пользователей в эту директорию. Просто нельзя несколько файловых систем монтировать в одну директорию. Точнее можно, но обходными, ненадёжными путями, чего стоит избегать.

Но иногда встречается софт, которому нужно хранить все свои файлы в одной директории, допустим, базу данных. И вот настаёт день, когда на разделе, где лежат данные, перестаёт хватать места. Примонтировать новую файловую систему в эту директорию не получится. Можно купить диск большего объёма и перенести на него файлы. Но, если там терабайты данных, перенос займёт много времени, а сервис при этом будет недоступен. И потом получится, что на первом диске место освободилось, но мы его не можем использовать. А завтра второй диск забьётся, опять купим новый диск большего объёма, опять перенесём на него файлы, и теперь у нас и первый и второй диск не будут использоваться.

Нужно больше места для файлов пользователей

1) Создаём sda4, с помощью btrfs расширяем файловую систему



Но мы монтируем файловую систему, так? И, если таблицы разделов не позволяют, можно ли на уровне файловой системы сделать так, чтобы одна файловая система могла работать на нескольких разделах и дисках? Да, действительно, есть такие файловые системы – допустим, btrfs. Но это относительно молодые файловые системы, пока не используемые повсеместно. Однако, вполне может быть, что через пару лет все файловые системы будут поддерживать возможность работы на нескольких разделах и дисках.

Хорошо, есть ли что-нибудь проверенное годами, что-то выше таблицы разделов, но ниже файловой системы? Да, есть такое и оно называется LVM – менеджер логических томов. Он позволяет вам объединять несколько разделов и дисков чтобы создать общее пространство. На основе этого общего пространства вы можете создавать логические разделы, на которых вы создадите файловую систему, примонтируете и всё как обычно.

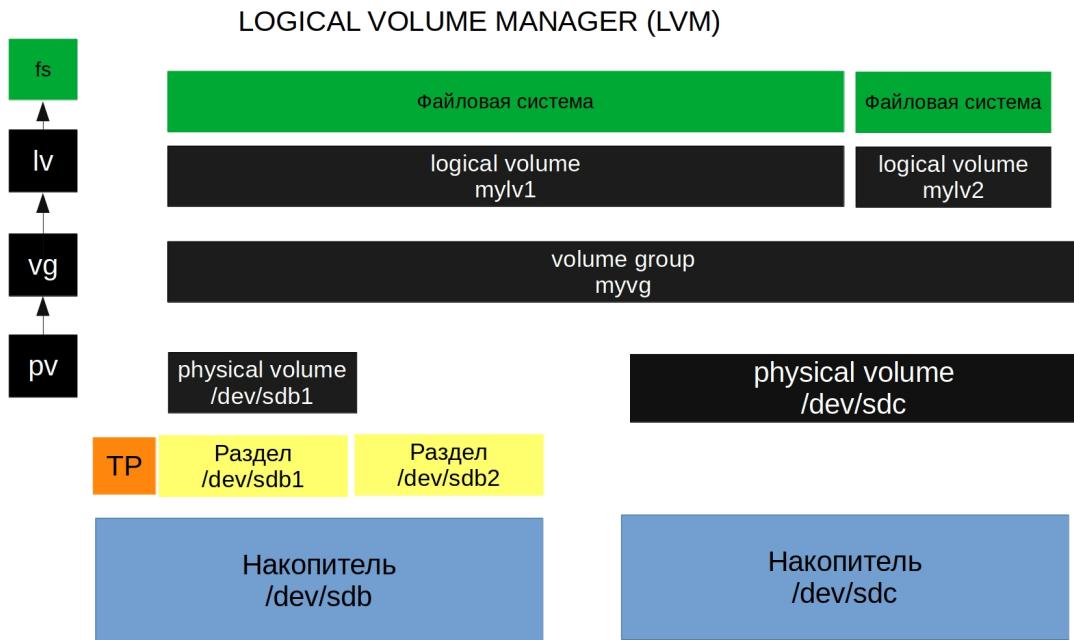


Схема довольно простая – ваши разделы и диски становятся физическими томами – `physical volume` – `pv`. Общее пространство, которое формирует группа физических томов называется группой томов – `volume group` – `vg`. А разделы, которые создаются внутри этой группы – логическими томами – `logical volume` – `lv`. Физический том – группа томов – логический том. Несложно, просто нужно пару раз попрактиковаться. При этом LVM позволяет добавлять разделы и диски в группу томов и увеличивать логические тома с файловыми системами без необходимости останавливать работу. Логические тома ещё можно уменьшать при необходимости, но для этого надо уменьшить файловую систему, а это не всегда просто. Это далеко не весь функционал LVM, но давайте пока немного попрактикуемся.

```
[user@centos8 ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/devtmpfs        886M    0   886M  0% /dev
tmpfs           914M    0   914M  0% /dev/shm
tmpfs           914M  9.6M  904M  2% /run
tmpfs           914M    0   914M  0% /sys/fs/cgroup
/dev/mapper/cl_centos8-root  17G  5.5G   12G  32% /
/dev/sda1        976M 272M  638M  30% /boot
tmpfs           183M  1.2M  182M  1% /run/user/42
tmpfs           183M  9.1M  174M  5% /run/user/1000
/dev/sr0          59M   59M     0 100% /run/media/user/VBox_GAs_6.1.
```

Для начала, я восстановил снапшот виртуалки, чтобы `/home` не был на диске `sdc`, так как нам понадобятся 2 диска для тестов:

```
df -h
```

Вы можете восстановить снапшот, перенести `/home` обратно, либо добавить 2 новых диска.

```
[user@centos8 ~]$ sudo fdisk /dev/sdb

Welcome to fdisk (util-linux 2.32.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Command (m for help): g
Created a new GPT disklabel (GUID: 0E76E16E-C210-714F-AE26-EB6ECE51D37A).

Command (m for help): n
Partition number (1-128, default 1):
First sector (2048-2097118, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-2097118, default 2097118): +200M

Created a new partition 1 of type 'Linux filesystem' and of size 200 MiB.

Command (m for help): n
Partition number (2-128, default 2):
First sector (411648-2097118, default 411648):
Last sector, +sectors or +size{K,M,G,T,P} (411648-2097118, default 2097118): +400M

Created a new partition 2 of type 'Linux filesystem' and of size 400 MiB.
```

```
Command (m for help): p
Disk /dev/sdb: 1 GiB, 1073741824 bytes, 2097152 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 0E76E16E-C210-714F-AE26-EB6ECE51D37A
```

Device	Start	End	Sectors	Size	Type
/dev/sdb1	2048	411647	409600	200M	Linux filesystem
/dev/sdb2	411648	1230847	819200	400M	Linux filesystem

```
Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
```

Начнём с простого – создадим таблицу разделов на диске sdb и два раздела на 200 и 400 мегабайт:

```
sudo fdisk /dev/sdb
g
n
enter
enter
+200M
n
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
enter
enter
+400M
P
w
```

```
[user@centos8 ~]$ sudo pvcreate /dev/sdb1
Physical volume "/dev/sdb1" successfully created.
[user@centos8 ~]$ sudo pvs
PV          VG      Fmt Attr PSize  PFree
/dev/sda2  cl_centos8 lvm2 a--  <19.00g    0
/dev/sdb1            lvm2 ---  200.00m 200.00m
[user@centos8 ~]$ sudo pvdisk /dev/sdb1
"/dev/sdb1" is a new physical volume of "200.00 MiB"
--- NEW Physical volume ---
PV Name           /dev/sdb1
VG Name
PV Size          200.00 MiB
Allocatable      NO
PE Size          0
Total PE         0
Free PE          0
Allocated PE     0
PV UUID          UhB0l1-5CK9-By3e-z2bq-jyKa-7H46-I1QIWh
```

Хотя следующий шаг можно пропустить, я этого делать не буду, чтобы было понятнее. Вспоминаем последовательность – physical volume, volume group и logical volume. Все три нам нужно создать по порядку. И так, physical volume – это pv, создать – create - pvcreate и указываем раздел или диск, который будет использоваться:

```
sudo pvcreate /dev/sdb1
```

Смотрим вывод – success – значит всё окей. У нас теперь есть физический том, посмотреть его мы можем с помощью:

```
sudo pvs
sudo pvdisk /dev/sdb1
```

Как видите, тут помимо sdb1 есть ещё sda2 – это потому что при установке система сама создала LVM, но мы его трогать не будем.

```
[user@centos8 ~]$ sudo vgcreate myvg /dev/sdb1
  Volume group "myvg" successfully created
[user@centos8 ~]$ sudo vgs
  VG          #PV #LV #SN Attr   VSize   VFree
  cl_centos8   1   2   0 wz--n- <19.00g       0
  myvg         1   0   0 wz--n-  196.00m  196.00m
```

Physical volume у нас есть, дальше нужно создать группу томов – volume group – vgcreate. Для этого нужно дать ей имя и указать physical volume, один или несколько:

```
sudo vgcreate myvg /dev/sdb1
sudo vgs
```

Опять видим success – всё окей.

```
[user@centos8 ~]$ sudo vgdisplay myvg
--- Volume group ---
VG Name           myvg
System ID
Format           lvm2
Metadata Areas   1
Metadata Sequence No 1
VG Access        read/write
VG Status        resizable
MAX LV           0
Cur LV           0
Open LV           0
Max PV           0
Cur PV           1
Act PV           1
VG Size          196.00 MiB
PE Size          4.00 MiB
Total PE         49
Alloc PE / Size  0 / 0
Free  PE / Size  49 / 196.00 MiB
VG UUID          xqsUh5-kMmb-0X9Q-541S-8i12-kgA4-ezsLAM
```

Как посмотреть? Правильно:

```
sudo vgdisplay myvg
```

Опять же, у нас тут есть volume group, который создался при установке – cl_centos8 и наша группа – myvg. Отсюда же можем посмотреть информацию о ней, допустим, сколько свободного места – почти 200 мегабайт. Тут ещё есть PE – physical extents. Это блоки, которыми оперирует LVM. Как видите, они у нас по 4 мебибайта и их у нас в сумме 49.

```
[user@centos8 ~]$ sudo lvcreate myvg -n mylv -L 200M
Volume group "myvg" has insufficient free space (49 extents): 50 required.
[user@centos8 ~]$ sudo lvcreate myvg -n mylv -l 80%FREE
Logical volume "mylv" created.
[user@centos8 ~]$ sudo lvs
  LV   VG      Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  root cl_centos8 -wi-ao---- <17.00g
  swap cl_centos8 -wi-ao----  2.00g
  mylv myvg     -wi-a----- 156.00m
```

Последний шаг – logical volume – lvcreate. Тут у нас опций чуть больше – нужно указать в каком volume group мы хотим создать наш логический том, указать имя для logical volume и указать размер:

```
sudo lvcreate myvg -n mylv -L 200M
```

Как видите, у нас не получается выдать 200, потому что 1 блок ушёл на метаданные для LVM. Мы можем указать 196 или поступить по другому – выдать свободное место по процентам. Допустим, выдать всё свободное место, либо 80 процентов свободного – для этого ключ -l:

```
sudo lvcreate myvg -n mylv -l 80%FREE
sudo lvs
```

EXAMPLES

Create a striped LV with 3 stripes, a stripe size of 8KiB and a size of 100MiB. The LV name is chosen by lvcreate.
lvcreate -i 3 -I 8 -L 100m vg00

Create a raid1 LV with two images, and a useable size of 500 MiB. This operation requires two devices, one for each mirror image. RAID metadata (superblock and bitmap) is also included on the two devices.
lvcreate --type raid1 -m1 -L 500m -n mylv vg00

Create a mirror LV with two images, and a useable size of 500 MiB. This operation requires three devices: two for mirror images and one for a disk log.
lvcreate --type mirror -m1 -L 500m -n mylv vg00

Кстати, важное замечание – если открыть man по любой из предыдущих команд lvm, допустим:

```
man lvcreate
```

и спуститься в самый низ, там есть примеры использования. И если вы вдруг забудете какой-то ключ, то эти примеры помогут вам вспомнить.

```
[user@centos8 ~]$ sudo lvdisplay myvg/mylv
--- Logical volume ---
LV Path              /dev/myvg/mylv
LV Name              mylv
VG Name              myvg
LV UUID              riMLEI-28bT-PYNf-gnup-ze9L-gxbl-StUF0v
LV Write Access      read/write
LV Creation host, time centos8, 2021-03-15 14:02:16 +0400
LV Status            available
# open               0
LV Size              156.00 MiB
Current LE           39
Segments             1
Allocation           inherit
Read ahead sectors   auto
 - currently set to 8192
Block device         253:2
```

А сейчас, давайте посмотрим, что у нас получилось:

```
sudo lvdisplay myvg/mylv
```

И так, тут у нас появился путь - /dev/myvg/mylv.

```
[user@centos8 ~]$ sudo mkfs.ext4 /dev/myvg/mylv
[sudo] password for user:
mke2fs 1.45.4 (23-Sep-2019)
Creating filesystem with 159744 1k blocks and 40000 inodes
Filesystem UUID: de9adc0c-b8f7-42dd-934d-23a6f504ccef
Superblock backups stored on blocks:
          8193, 24577, 40961, 57345, 73729

Allocating group tables: done
Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done
```

Именно сюда мы поставим файловую систему с помощью mkfs:

```
sudo mkfs.ext4 /dev/myvg/mylv
```

```
[user@centos8 ~]$ ll /dev/myvg/mylv
lrwxrwxrwx. 1 root root 7 Mar 15 14:18 /dev/myvg/mylv -> ../../dm-2
[user@centos8 ~]$ ll /dev/dm-*
brw-rw----. 1 root disk 253, 0 Mar 15 13:57 /dev/dm-0
brw-rw----. 1 root disk 253, 1 Mar 15 13:57 /dev/dm-1
brw-rw----. 1 root disk 253, 2 Mar 15 14:18 /dev/dm-2
[user@centos8 ~]$ sudo lsblk -f /dev/myvg/mylv
NAME      FSTYPE LABEL UUID                                     MOUNTPOINT
myvg-my lv ext4      de9adc0c-b8f7-42dd-934d-23a6f504cc ef
```

Сам же этот файл:

```
ll /dev/myvg/mylv
```

является символьической ссылкой на /dev/dm-2

```
ll /dev/dm-*
```

который, как и sda, генерируется при обнаружении устройств. А значит, если говорить про fstab, /dev/dm-2 там указывать нельзя. Можно указать UUID:

```
sudo lsblk -f /dev/myvg/mylv
```

или /dev/myvg/mylv.

```
[user@centos8 ~]$ tail -3 /etc/fstab
/dev/mapper/cl_centos8-root /
UUID=c607eb50-9842-4b18-ba8e-2d1139960594 /boot           xfs      defaults      0 0
/dev/mapper/cl_centos8-swap swap                  swap      defaults      0 0
[user@centos8 ~]$ ll /dev/mapper/
total 0
lrwxrwxrwx. 1 root root    7 Mar 15 13:57 cl_centos8-root -> ../../dm-0
lrwxrwxrwx. 1 root root    7 Mar 15 13:57 cl_centos8-swap -> ../../dm-1
crw-----. 1 root root 10, 236 Mar 15 13:57 control
lrwxrwxrwx. 1 root root    7 Mar 15 14:18 myvg-my lv -> ../../dm-2
```

Но, при определённых условиях, о которых вы можете почитать по [ссылке](#), оба варианта могут создавать проблемы. Поэтому лучший способ – способ, который предлагает сам RedHat – уже указан в fstab:

```
tail -3 /etc/fstab
```

/dev/mapper. Если посмотреть:

```
ll /dev/mapper
```

можно увидеть, что, как и /dev/myvg/mylv, тут символьические ссылки.

```
/dev/mapper/cl_centos8-root /
UUID=c607eb50-9842-4b18-ba8e-2d1139960594 /boot           xfs      defaults      0 0
/dev/mapper/cl_centos8-swap swap                  swap      defaults      0 0
/dev/mapper/myvg-my lv /mydata   ext4 noatime 0 0
```

В целях совместимости рекомендую использовать вариант именно с /dev/mapper, потому что на старых системах с другими вариантами могут быть проблемы:

```
sudo nano /etc/fstab
```

```
/dev/mapper/myvg-my lv /mydata ext4 noatime 0 0
```

```
[user@centos8 ~]$ sudo mkdir /mydata
[user@centos8 ~]$ sudo mount /mydata
[user@centos8 ~]$ df -h /mydata
Filesystem           Size   Used  Avail Use% Mounted on
/dev/mapper/myvg-my lv  148M   1.6M  135M   2% /mydata
[user@centos8 ~]$ █
```

Примонтируем и убедимся, что всё работает:

```
sudo mkdir /mydata
sudo mount /mydata
df -h /mydata
```

```
[user@centos8 ~]$ sudo pvcreate /dev/sdc
Device /dev/sdc excluded by a filter.
[user@centos8 ~]$ sudo wipefs -a /dev/sdc
/dev/sdc: 8 bytes were erased at offset 0x000000200 (gpt): 45 46 49 20 50 41 52 54
/dev/sdc: 8 bytes were erased at offset 0x3fffffe00 (gpt): 45 46 49 20 50 41 52 54
/dev/sdc: 2 bytes were erased at offset 0x0000001fe (PMBR): 55 aa
/dev/sdc: calling ioctl to re-read partition table: Success
[user@centos8 ~]$ sudo pvcreate /dev/sdc
Physical volume "/dev/sdc" successfully created.
[user@centos8 ~]$ █
```

Предположим, у нас перестало хватать места и мы решили добавить ещё один диск – sdc. Так, у нас есть volume group, состоящий из одного physical volume – раздела sdb1. Я хочу полностью отдать весь диск sdc volume группе myvg. Первый шаг – создаю physical volume:

```
sudo pvcreate /dev/sdc
```

Но вместо привычного success я тут вижу excluded by a filter. LVM заботливо отказывается использовать весь диск, так как там уже есть таблица разделов. Чтобы избавиться от неё я использую утилиту wipefs с ключом -a:

```
sudo wipefs -a /dev/sdc
```

И теперь всё у меня создаётся как надо:

```
sudo pvcreate /dev/sdc
```

```
[user@centos8 ~]$ sudo vgextend myvg /dev/sdc
Volume group "myvg" successfully extended
[user@centos8 ~]$ sudo vgdisplay myvg
--- Volume group ---
VG Name           myvg
System ID
Format           lvm2
Metadata Areas   2
Metadata Sequence No 3
VG Access        read/write
VG Status        resizable
MAX LV
Cur LV
Open LV
Max PV
Cur PV
Act PV
VG Size          <1.19 GiB
PE Size          4.00 MiB
Total PE         304
Alloc PE / Size  39 / 156.00 MiB
Free  PE / Size  265 / <1.04 GiB
VG UUID          st1kGh-BY4o-NbM1-aNWC-pRZB-0Lse-Vc1M0k
```

Дальше мне нужно расширить volume группу и добавить в неё новый диск. Расширить это extend, а значит пишем:

```
sudo vgextend myvg /dev/sdc
```

Видим success и смотрим:

```
sudo vgdisplay myvg
```

Теперь в myvg 2 physical volume, 304 блока и гигабайт свободного места. Но это пока что группа, нужно ещё увеличить logical volume и файловую систему.

```
[user@centos8 ~]$ df -h /mydata
Filesystem      Size  Used Avail Use% Mounted on
/dev/mapper/myvg-my lv 148M  1.6M 135M  2% /mydata
[user@centos8 ~]$ sudo lvextend myvg/my lv -L +500M
[sudo] password for user:
Size of logical volume myvg/my lv changed from 156.00 MiB (39 extents) to 656.00 MiB (16
Logical volume myvg/my lv successfully resized.
[user@centos8 ~]$ sudo lvs
  LV   VG     Attr       LSize   Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  root cl_centos8 -wi-ao---- <17.00g
  swap cl_centos8 -wi-ao----   2.00g
  my lv myvg      -wi-ao---- 656.00m
```

Да, если просто увеличить раздел или логический том – файловая система об изменениях не узнает. Поэтому нужно проделать 2 операции, но можно и одной командой. И так, смотрим сколько у нас свободного места на файловой системе:

```
df -h /mydata
```

100 с лишним мегабайт. Теперь пишем:

```
sudo lvextend myvg/mylv -L +500M
```

Видим success – значит всё получилось. Смотрим размер logical volume:

```
sudo lvs
```

600 с лишним мегабайт.

```
[user@centos8 ~]$ df -h /mydata/
Filesystem           Size  Used Avail Use% Mounted on
/dev/mapper/myvg-my lv 148M  1.6M 135M  2% /mydata
[user@centos8 ~]$ sudo resize2fs /dev/myvg/mylv
[sudo] password for user:
resize2fs 1.45.4 (23-Sep-2019)
Filesystem at /dev/myvg/mylv is mounted on /mydata; on-line resizing required
old_desc_blocks = 2, new_desc_blocks = 6
The filesystem on /dev/myvg/mylv is now 671744 (1k) blocks long.

[user@centos8 ~]$ df -h /mydata/
Filesystem           Size  Used Avail Use% Mounted on
/dev/mapper/myvg-my lv 632M  2.5M 599M  1% /mydata
```

Смотрим:

```
df -h /mydata
```

100 с лишним, ничего не изменилось. Файловая система изменений не увидела. Но об этом ей можно сказать:

```
sudo resize2fs /dev/myvg/mylv
```

Теперь опять проверяем:

```
df -h /mydata
```

всё как надо.

```
[user@centos8 ~]$ df -h /mydata/
Filesystem           Size  Used Avail Use% Mounted on
/dev/mapper/myvg-my lv 632M  2.5M 599M  1% /mydata
[user@centos8 ~]$ sudo lvextend myvg/mylv -L +100M -r
      Size of logical volume myvg/mylv changed from 656.00 MiB (164 extents) to 756.00 MiB
.
Logical volume myvg/mylv successfully resized.
resize2fs 1.45.4 (23-Sep-2019)
Filesystem at /dev/mapper/myvg-my lv is mounted on /mydata; on-line resizing required
old_desc_blocks = 6, new_desc_blocks = 6
The filesystem on /dev/mapper/myvg-my lv is now 774144 (1k) blocks long.

[user@centos8 ~]$ df -h /mydata/
Filesystem           Size  Used Avail Use% Mounted on
/dev/mapper/myvg-my lv 729M  2.5M 692M  1% /mydata
```

Но можно сразу увеличить размер файловой системы при увеличении logical volume - тот же lvextend но с ключом -r:

```
sudo lvextend myvg/mylv -L +100M -r
df -h /mydata
```

Как видите, никаких лишних действий.

Таким образом, мы смогли увеличить нашу файловую систему и в дальнейшем мы всегда сможем продолжать так делать, без необходимости останавливать работу, просто добавляя новые разделы или диски в нашу volume группу. При этом нам не важно, где заканчивается один lv и начинается другой – нет таких ограничений, как со стандартными разделами. Что касается удаления, то всё также просто:

```
lvremove
vgremove
pvremove
```

В момент удаления логические тома не должны быть никуда примонтированы. Есть ещё много разных базовых команд с LVM – уменьшить размер, удалить physical volume из volume группы, переместить данные с одного physical volume на другой и много всего – но их много, они ситуативны, и всё разбирать в азах бессмысленно.

Но как я говорил, это далеко не весь функционал LVM-а. Как вы заметили, я стараюсь не выделять всё свободное пространство на logical volume. С одной стороны потому, что когда понадобится, увеличить размер файловой системы получится без труда. А вот если мне вдруг перестанет хватать места и я захочу уменьшить один из разделов – это будет сложнее. С другой стороны – свободное место нужно для снапшотов.

Вы уже знаете, что такое снапшот – снимок текущего состояния. Мы делали снапшоты для виртуалки – сняли снапшот после установки системы, и, если вдруг мы где-то что-то испортим, всегда можем вернуться к прежнему состоянию. Это как сохранение в играх. В виртуальной инфраструктуре гипервизоры позволяют нам делать снапшоты. Когда нет гипервизора, допустим, это система, установленная на физический сервер или ваш домашний компьютер – снапшот не сделаешь, но можно делать бэкапы. Но, в случае проблем, восстановление из бэкапа занимает много времени – нужно перенести данные из одного накопителя в другой, это сильно зависит от скорости накопителя. В целом, восстановление из бэкапа чуть ли не последний вариант, к которому прибегают администраторы.

Да и даже восстановление снапшота в виртуальной инфраструктуре не всегда приятно – для этого нужно выключить виртуалку – а если на ней какой-нибудь тяжелый софт, который долго запускается, или, скажем, виртуалку не трогали годами – есть шанс, что всплынет куча других проблем. Тут нам на помощь приходит тот же самый LVM, который сам может делать снапшоты. Тут всё немного проще – вам не нужно перезагружать виртуалку и можно делать на физическом сервере. Всё что нужно чтобы восстановить – отмонтировать файловую систему. Да, если у вас проблемы в корневой файловой системе, что-то там после обновления отвалилось, без перезагрузки не обойтись. Если проблемы с данными какого-нибудь сервиса вам придётся остановить его, чтобы отмонтировать файловую систему с данными и восстановить снапшот. Но это в любом случае быстрее, легче и нередко можно обойтись без перезагрузки.

```
[user@centos8 ~]$ sudo nano /mydata/file
[sudo] password for user:
[user@centos8 ~]$ cat /mydata/file
Before snapshot
[user@centos8 ~]$ sudo lvcreate -s -n mysnapshot -L 100M myvg/mylv
Logical volume "mysnapshot" created.
[user@centos8 ~]$ sudo lvs
  LV        VG          Attr      LSize   Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  root      cl_centos8 -wi-ao---- <17.00g
  swap      cl_centos8 -wi-ao----  2.00g
  mylv      myvg         owi-aos--- 756.00m
  mysnapshot myvg         swi-a-s--- 100.00m    mylv   0.01
```

И так, прежде чем сделаем снапшот, создадим файл в /mydata, на который будем ориентироваться:

```
sudo nano /mydata/file
```

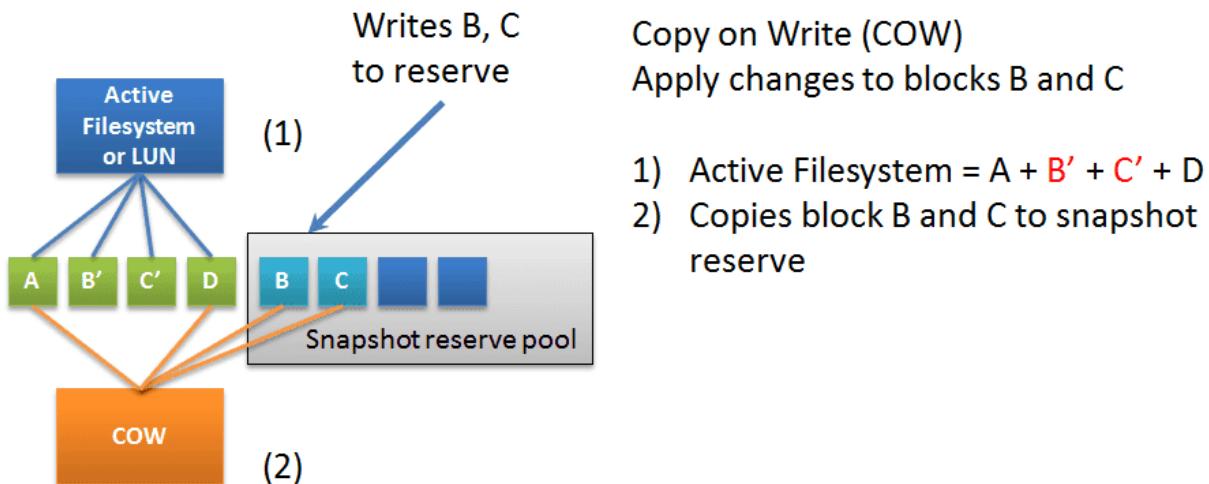
Before snapshot

```
cat /mydata/file
```

Теперь создадим снапшот, нам для этого понадобится та же команда lvcreate с опцией -s - снапшот. Пишем sudo lvcreate -s -n (имя снапшота) mysnapshot -L 100M (размер снапшота) myvg/mylv:

```
sudo lvcreate -s -n mysnapshot -L 100M myvg/mylv
sudo lvs
```

Снапшот создан.



У вас, возможно, возникли два вопроса – что за размер снапшота и почему так мгновенно? LVM использует технологию copy-on-write – копирование при изменении. Это работает так – мы создаём какое-то место для снапшота на диске – допустим на 100 мегабайт. Когда мы что-то меняем на исходном logical volume – старые блоки перед изменением копируются в это специальное место, в итоге на оригинальном lv новые блоки, а в выделенной области старые версии этих блоков. В случае, если у нас прям все блоки заменятся на исходном lv, размер снапшота должен быть размером с исходный логический том, ну, чуть больше, для всяких метаданных. А в большинстве случаев меняются не все блоки, а только часть, поэтому размер снапшота может быть меньше, т.е. нужно думать, а как много измениться на исходной lv, пока я буду держать снапшот.

```
[user@centos8 ~]$ sudo lvs myvg/mysnapshot
  LV      VG  Attr      LSize   Pool Origin Data%  Meta%
  mysnapshot myvg swi-a-s--- 100.00m      mylv    0.01
[user@centos8 ~]$ sudo nano /mydata/file
[user@centos8 ~]$ cat /mydata/file
After snapshot
[user@centos8 ~]$ sudo lvs myvg/mysnapshot
  LV      VG  Attr      LSize   Pool Origin Data%  Meta%
  mysnapshot myvg swi-a-s--- 100.00m      mylv    0.03
```

Теперь изменим наш файл:

```
sudo lvs myvg/mysnaphost
sudo nano /mydata/file
```

After snapshot

```
cat /mydata/file
```

Смотрим lvs:

```
sudo lvs myvg/mysnaphost
```

видим, что у снапшота используемое пространство немного изменилось.

```
[user@centos8 ~]$ sudo mount /dev/myvg/mysnapshot /mnt
[user@centos8 ~]$ cat /mnt/file
Before snapshot
[user@centos8 ~]$ cat /mydata/file
After snapshot
[user@centos8 ~]$ █
```

Кстати, мы можем примонтировать его и увидеть наши файлы в их исходном виде:

```
sudo mount /dev/myvg/mysnapshot /mnt
cat /mnt/file
cat /mydata/file
```

```
[user@centos8 ~]$ sudo umount /mnt /mydata
[user@centos8 ~]$ sudo lvconvert --merge myvg/mysnapshot
Merging of volume myvg/mysnapshot started.
myvg/mylv: Merged: 100.00%
[user@centos8 ~]$ sudo mount /mydata
[user@centos8 ~]$ sudo cat /mydata/file
Before snapshot
[user@centos8 ~]$ █
```

Снапшот можно удалить как и обычный логический том:

```
sudo lvremove myvg/mysnapshot
```

но я этого делать не буду. Вместо этого, давайте восстановимся из этого снапшота. Для этого нам понадобится отмонтировать снапшот и исходный том:

```
sudo umount /mnt /mydata
```

Далее соединить снапшот с исходным lv:

```
sudo lvconvert --merge myvg/mysnapshot
```

Дальше снапшот удалится и мы сможем примонтировать наш lv как раньше и проверить файл:

```
sudo mount /mydata  
cat /mydata/file  
sudo lvs
```

В целом снапшоты увеличивают количество записей на диске, поэтому на активно используемых томах они будут снижать производительность диска, особенно когда снапшотов несколько. Поэтому их стоит делать перед какими-то изменениями – допустим, перед обновлением системы. А потом, если всё прошло успешно, их удалять. Также снапшоты помогают с бэкапом – во время снапшота сохраняется текущее состояние файловой системы, из-за чего софту для бэкапа легче работать с этим самым снапшотом, а не исходной файловой системой, на которую постоянно может идти запись.

Если же в целом говорить про LVM, это отличный инструмент, который даёт администратору возможность более гибко работать с дисками. В то же время, это дополнительный слой абстракции и проблемы с LVM решать сложнее, чем проблемы с обычными разделами. Однако, больше практики, самостоятельного изучения и всяких экспериментов помогут вам лучше освоить этот инструмент. Больше информации про LVM вы можете найти по [ссылке](#). Но, в рамках основ, вам достаточно знать базовые операции – как создать physical volume, volume group, logical volume, как расширить vg и lv с учётом файловой системы, ну и как работать со снапшотами.

2.25.2 Практика

Вопросы

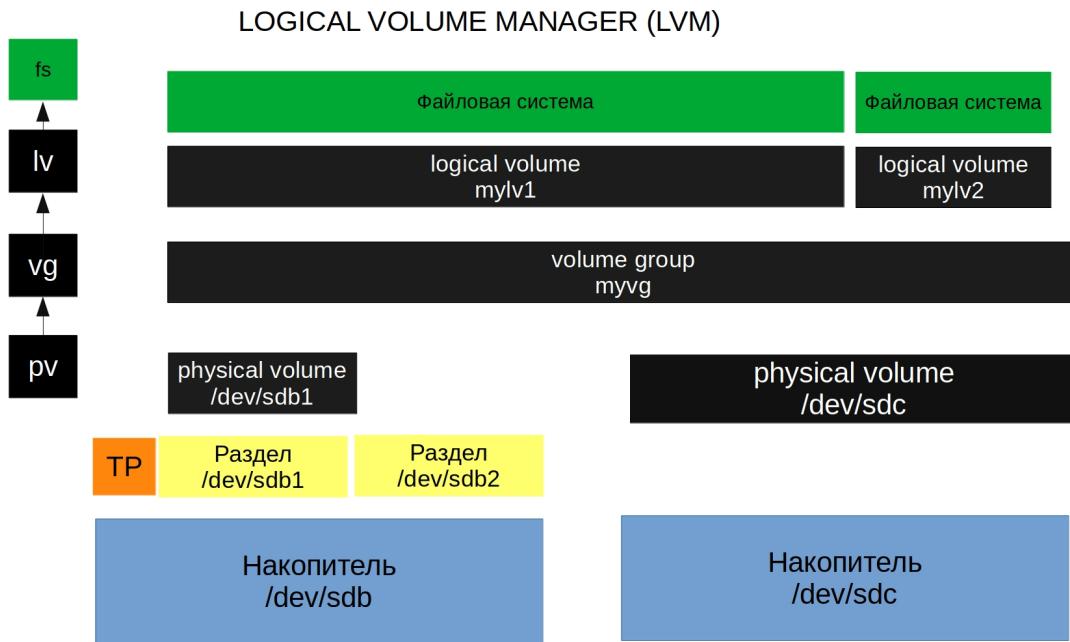
1. Зачем нужен LVM?
2. Опишите весь процесс от добавления диска до настройки fstab с учётом использования LVM.
3. На файловой системе с LV заканчивается место. Вы добавили новый диск. Как сделать так, чтобы файловая система увеличилась?
4. Как посмотреть информацию о физических томах, группах томов и логических томах?
5. Опишите принцип работы copy-on-write?
6. Как создать снапшот? Как его восстановить или удалить?
7. Почему не стоит сразу выделять всё пространство под логический том?

Задания

1. Создайте LVM, состоящий из двух разделов и одного целого диска. Внутри группы томов создайте два логических раздела, каждый из которых должен занимать 30% пространства группы томов. При включении файловая система на первом логическом разделе должна монтироваться в директорию /data, а фс второго логического раздела должна монтироваться в директорию /backup.
2. Добавьте к виртуалке 3 диска. На первом создайте таблицу разделов mbr и 2 раздела. На втором ТР gpt и 1 раздел. На третьем диске не создавайте разделов. Создайте VG из одного PV (второй раздел диска с mbr) и на нём 1 LV с файловой системой ext4. Настройте автомонтирование в /mnt. Создайте на этой ФС директорию и файл внутри. Расширьте VG используя раздел с диска с GPT. Создайте снапшот LV. Создайте пару новых файлов на этой файловой системе, затем восстановите снапшот. Расширьте VG добавлением неразмеченного диска. Снимите ещё один снапшот. Добавьте ещё пару файлов, потом удалите снапшот.

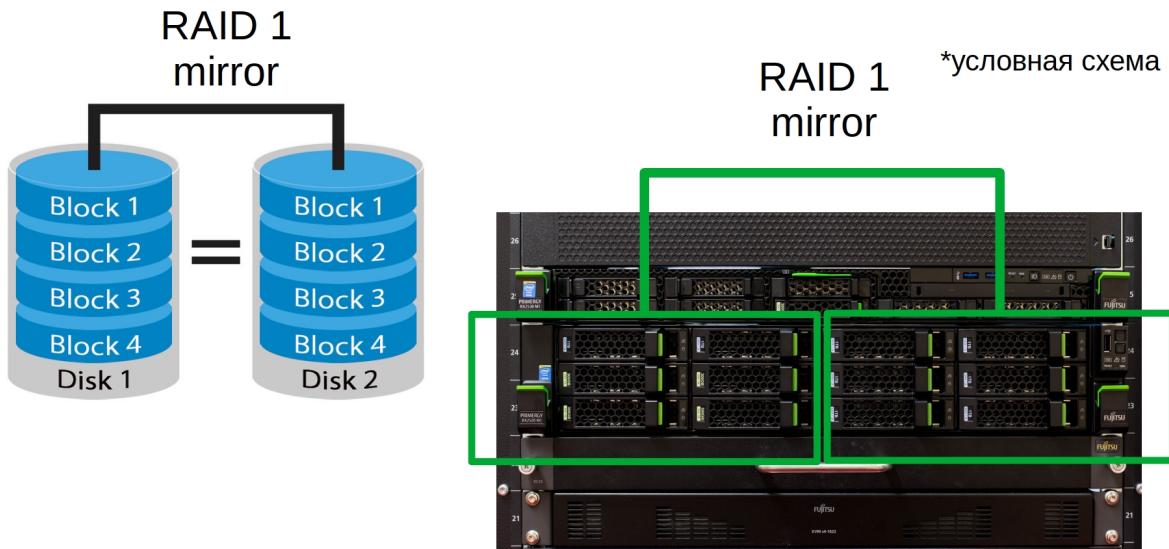
2.26 26. Программный RAID - MD

2.26.1 26. Программный RAID - MD



Представьте – у вас есть сервер, на котором работает важный сервис. Вы подняли на сервере LVM, периодически добавляете диски и вроде всё нормально. Пока в один день не выходит из строя один из дисков. Диски периодически портятся, это совершенно нормально и вы можете по гарантии заменить диск. Гораздо страшнее, что случится с вашими данными. Во первых, восстановить данные из повреждённого диска не всегда возможно, а если что-то и можно восстановить, то это обычно стоит бешеных денег. Но это только часть проблем. Если какой-то логический том использовал несколько дисков, включая проблемный, то придётся потратить много часов, а то и дней, чтобы восстановить хоть что-то, а о полном восстановлении можно забыть. Это уже тот случай, когда нужно восстанавливать из бэкапа.

RAID - Redundant Array of Independent Disks



Учитывая, сколько проблем может создать один повреждённый диск, компании готовы заплатить чуть больше за железо, чтобы избежать таких проблем. И задачи примерно такие – выход диска из строя не должен приводить к потере данных и остановке работы сервиса. Для решения этих задач была разработана технология RAID – избыточный массив независимых дисков. Вариаций RAID-а, а точнее уровней, много и самый простой вариант – записывать одни и те же данные на два или больше дисков. Такой уровень RAID называется RAID 1 или mirror. Выйдет из строя один диск – не проблема, все данные есть на втором диске. Правда есть пара нюансов. Во первых – скорость записи немного уменьшается, так как нужно писать одни и те же данные на несколько дисков, хотя данные и пишутся параллельно. Во вторых – вам нужно минимум в два раза больше дисков. Учитывая, что серверные диски стоят дороже, так как должны работать 24/7 и быть надёжнее, то разница между 10 и 20 дисками довольно большая, не говоря уже о большем количестве дисков. Тут и другие нюансы - у вас на сервере может быть 12 слотов на диски, а значит полезное пространство дадут только 6 дисков. А если в дальнейшем понадобится больше места? Либо покупать новые диски большего объёма и каким-то образом переносить все данные на новые диски. Либо покупать систему хранения данных, либо второй сервер. Всё не так просто и много где придётся искать компромисс.

RAID контроллер



Пример утилиты для настройки рейда



Кроме RAID 1 есть и другие уровни. Большинство компаний, у которых есть сервера, используют RAID с уровнями 1, 5, 6 и 10. Но это большая тема с кучей нюансов, которые не относятся к этому курсу, но вы можете почитать о них по [ссылке](#). RAID можно реализовать как на уровне операционной системы – что называют программным рейдом, так и на уровне отдельного контроллера, или впаянного в материнку чипа, что называется аппаратным рейдом. И, скорее всего, на работе вы увидите именно аппаратный рейд – он независим от операционной системы, не использует ресурсы центрального процессора, работает чуть быстрее и может иметь специальную батарею, которая позволит не потерять данные в случае резкого выключения сервера. Но он стоит денег и в случае проблем с контроллером придётся покупать контроллер у того же вендора, иначе можно потерять все данные. Дело в том, что RAID добавляет уровень абстракции, храня метаданные на дисках. При этом каждый производитель делает это по своему, из-за чего разные контроллеры могут не определить RAID другого вендора. Хотя в случае первого рейда это не всегда актуально. Но в случае программного рейда вы не зависите от контроллера, вам просто нужен модуль в ядре, который и будет работать с рейдом.

MD(4)	Kernel Interfaces Manual	MD(4)
NAME	<code>md</code> - Multiple Device driver aka Linux Software RAID	
SYNOPSIS	<code>/dev/md_n</code> <code>/dev/md_n</code> <code>/dev/md/name</code>	
DESCRIPTION	The <code>md</code> driver provides virtual devices that are created from one or more independent underlying devices. This array of devices often contains redundancy and the devices are often disk drives, hence the acronym RAID which stands for a Redundant Array of Independent Disks.	

Вообще, RAID можно настроить и с помощью LVM. Но сам по себе LVM при работе с рейдом обращается к другому модулю для работы с рейдом – `md` – multiple devices:

```
man md
```

И лучше научиться работать с самим md чтобы не зависеть от LVM, так как LVM это делает поверхностью, а с md можно сделать гораздо больше. Мы с вами для теста поднимем RAID 1 на sdb и sdc.

```
[user@centos8 ~]$ df -h /mydata/
Filesystem      Size  Used Avail Use% Mounted on
/dev/mapper/myvg-my lv 729M  2.5M  691M   1% /mydata
[user@centos8 ~]$ sudo umount /mydata
[user@centos8 ~]$ sudo vgremove myvg
Do you really want to remove volume group "myvg" containing 1 logical volumes? [y/n]: y
Do you really want to remove active logical volume myvg/my lv? [y/n]: y
Logical volume "my lv" successfully removed
Volume group "myvg" successfully removed
[user@centos8 ~]$ sudo wipefs -a /dev/sdb /dev/sdc
/dev/sdb: 8 bytes were erased at offset 0x000000200 (gpt): 45 46 49 20 50 41 52 54
/dev/sdb: 8 bytes were erased at offset 0x3fffffe00 (gpt): 45 46 49 20 50 41 52 54
/dev/sdb: 2 bytes were erased at offset 0x0000001fe (PMBR): 55 aa
/dev/sdb: calling ioctl to re-read partition table: Success
/dev/sdc: 8 bytes were erased at offset 0x000000218 (LVM2_member): 4c 56 4d 32 20 30 30 31
[user@centos8 ~]$ sudo nano /etc/fstab
[user@centos8 ~]$ tail -1 /etc/fstab
#/dev/mapper/myvg-my lv /mydata ext4 noatime 0 0
[user@centos8 ~]$
```

Но сейчас на них находится LVM:

```
df -h /mydata
```

который мы создали в прошлый раз, он примонтирован в /mydata. У меня здесь ничего важного нет, поэтому я могу отмонтировать файловую систему:

```
sudo umount /mydata
```

удалить vg:

```
sudo vgremove myvg
```

которая при удалении также предложит удалить логический том, а потом затереть все метки с дисков:

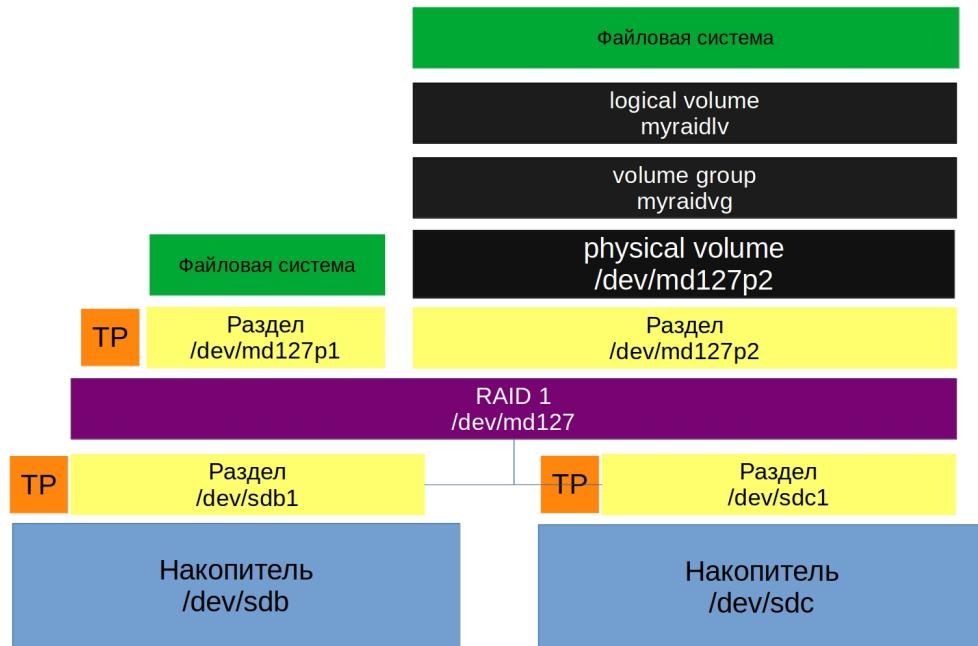
```
sudo wipefs -a /dev/sdb /dev/sdc
```

Также не стоит забывать про строчку в fstab:

```
sudo nano /etc/fstab
tail -1 /etc/fstab
```

её я просто закомментирую, так как она нам ещё понадобится.

Прежде чем создадим RAID, прибегнем к одной особенности программного RAIDа. Если говорить про аппаратный RAID, то очень сильно рекомендуется использовать диски одного вендора, одной модели и чуть ли не одной партии, потому что при одинаковых параметрах дисков – их объёме и скорости чтения и записи, гарантированно всё будет окей. Если диски разные – то при создании RAIDа будет выбираться скорость и объём самого малого и медленного диска. У разных вендоров объёмы дисков могут немного отличаться, пусть хоть на 1 сектор, не важно. И если у вас вышел из строя диск, вы купили другой и внезапно оказалось, что там на 100 секторов меньше, чем на других дисках RAIDа – то ваш RAID просто не примет новый диск. Поэтому для аппаратного RAIDа практически всегда используются одинаковые диски.



Что касается программного рейда, такой проблемы можно избежать. Для этого под рейд можно давать не весь диск, а чуть меньше, чтобы в случае замены подошёл любой новый диск нужного объёма. Для этого можно использовать разделы.

```
[user@centos8 ~]$ sudo fdisk -l /dev/sdb1 /dev/sdc1
Disk /dev/sdb1: 954 MiB, 1000341504 bytes, 1953792 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

```
Disk /dev/sdc1: 954 MiB, 1000341504 bytes, 1953792 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
[user@centos8 ~]$
```

Поэтому я создам раздел на каждом из дисков на 1 гигабайт:

```
sudo fdisk /dev/sdb
g
n
enter
enter
+1GB
p
w
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
sudo fdisk /dev/sdc
g
n
enter
enter
+1GB
p
w
sudo fdisk -l /dev/sdb1 /dev/sdc1
```

что чуть меньше реального объёма, но в дальнейшем мне будет проще заменить один из дисков.

EXAMPLES

```
mdadm --query /dev/name-of-device
```

This will find out if a given device is a RAID array, or is part of one, and will provide brief information about the device.

```
mdadm --assemble --scan
```

This will assemble and start all arrays listed in the standard config file. This command will typically go in a system startup file.

```
mdadm --stop --scan
```

This will shut down all arrays that can be shut down (i.e. are not currently in use). This will typically go in a system shutdown script.

```
mdadm --follow --scan --delay=120
```

If (and only if) there is an Email address or program given in the standard config file, then monitor the status of all arrays listed in that file by polling them ever 2 minutes.

```
mdadm --create /dev/md0 --level=1 --raid-devices=2 /dev/hd[ac]1
```

Create /dev/md0 as a RAID1 array consisting of /dev/hd1 and /dev/hd2.

Дальше нам понадобится утилита mdadm. У этой утилиты также подробный мануал с примерами:

```
man mdadm
/EXAMPLES
```

```
[user@centos8 ~]$ sudo mdadm --create myraid1 --level=1 --raid-devices=2 /dev/sdb1 /dev/sdc1
[sudo] password for user:
mdadm: Note: this array has metadata at the start and
      may not be suitable as a boot device. If you plan to
      store '/boot' on this device please ensure that
      your boot-loader understands md/v1.x metadata, or use
      --metadata=0.90
Continue creating array? y
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md/myraid1 started.
[user@centos8 ~]$
```

Пишем: sudo mdadm --create указываем имя рэйда, допустим, myraid1; указываем уровень рэйда --level=1 и указываем количество устройств и их имена --raid-devices=2 /dev/sdb1 /dev/sdc1:

```
sudo mdadm --create myraid1 --level=1 --raid-devices=2 /dev/sdb1 /dev/sdc1
```

Утилита нас предупреждает, что если мы планируем держать директорию /boot на этих дисках, нужно кое-что проверить или изменить, но пока мы этого не планируем. Причём здесь директория /boot мы разберём в другой раз, а пока продолжим. Пишем y и enter. Видим, что рэйд у нас создался.

```
[user@centos8 ~]$ ls -l /dev/md
total 0
lrwxrwxrwx. 1 root root 8 Mar 28 18:35 myraid1 -> ../../md127
[user@centos8 ~]$ sudo mdadm -D /dev/md127
/dev/md127:
      Version : 1.2
      Creation Time : Sun Mar 28 18:35:44 2021
      Raid Level : raid1
      Array Size : 975872 (953.00 MiB 999.29 MB)
      Used Dev Size : 975872 (953.00 MiB 999.29 MB)
      Raid Devices : 2
      Total Devices : 2
      Persistence : Superblock is persistent

      Update Time : Sun Mar 28 18:35:49 2021
      State : clean
      Active Devices : 2
      Working Devices : 2
      Failed Devices : 0
      Spare Devices : 0
```

Если посмотреть:

```
ls -l /dev/md
```

увидим, что это ссылка на устройство /dev/md127. Информацию о рейде можем посмотреть с помощью утилиты mdadm с опцией detail или просто D:

```
sudo mdadm -D /dev/md127
```

```
echo 'DEVICE /dev/hd*[0-9] /dev/sd*[0-9]' > mdadm.conf
mdadm --detail --scan >> mdadm.conf
This will create a prototype config file that describes currently active arrays that are known to be made from partitions of IDE or SCSI drives. This file should be reviewed before being used as it may contain unwanted detail.

echo 'DEVICE /dev/hd[a-z] /dev/sd*[a-z]' > mdadm.conf
mdadm --examine --scan --config=mdadm.conf >> mdadm.conf
This will find arrays which could be assembled from existing IDE and SCSI whole drives (not partitions), and store the information in the format of a config file. This file is very likely to contain unwanted detail, particularly the devices= entries. It should be reviewed and edited before being used as an actual config file.
```

Дальше нам нужно сохранить текущую конфигурацию рейда. Для этого откроем man mdadm и найдём две строчки с echo:

```
man mdadm
/echo
```

Будет два примера – с разделами или целыми дисками, мы выберем первый, так как мы создали реайд на разделах.

MDADM.CONF(5)	File Formats Manual	MDADM.CONF(5)
NAME	mdadm.conf - configuration for management of Software RAID with mdadm	
SYNOPSIS	/etc/mdadm.conf	
DESCRIPTION	mdadm is a tool for creating, managing, and monitoring RAID devices using the md driver in Linux.	

Станем рутом и зайдём в директорию etc. Правда директория может различаться в зависимости от дистрибутива, поэтому лучше всё же посмотреть в man:

```
man mdadm.conf
```

The screenshot shows a terminal window titled 'user@centos8:etc'. The user has run the command 'echo 'DEVICE /dev/hd*[0-9] /dev/sd*[0-9]' > mdadm.conf' to create an empty configuration file. Then, they have run 'mdadm --detail --scan >> mdadm.conf' to scan for existing RAID devices and add them to the configuration. Finally, they have checked the contents of the file with 'cat mdadm.conf' to verify the results. The output shows a single array entry:

```
DEVICE /dev/hd*[0-9] /dev/sd*[0-9]
ARRAY /dev/md/myraid1 metadata=1.2 name=centos8:myraid1 UUID=b5d2f62e:5004a01b:e1fb91a7:ed707989
```

Первая строчка:

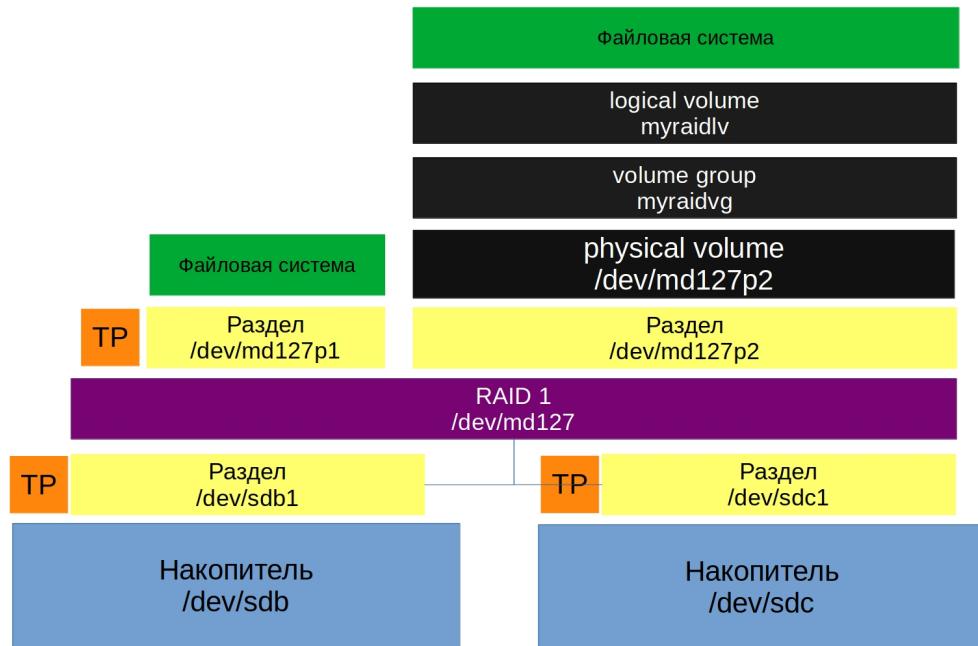
```
echo 'DEVICE /dev/hd*[0-9] /dev/sd*[0-9]' > mdadm.conf
```

создаст файл mdadm.conf со списком разделов, которые нужно проверять при запуске системы на наличие рейда. Можно это подправить, оставив только текущие разделы, либо оставить как есть. Вторая строчка:

```
mdadm --detail --scan >> mdadm.conf
```

добавит в файл информацию о текущем рейде и его идентификатор:

```
cat /etc/mdadm.conf
```



А дальше мы можем записать на файл устройство файловую систему с помощью `mkfs` или указать его как физический том для LVM. Мы сделаем и то и другое, для повторения. У вас может возникнуть вопрос – а зачем LVM, если RAID также объединяет диски, давая общее пространство? Вспомните, что у LVM есть и другие полезные возможности – те же снапшоты, более динамическое управление логическими томами, по сравнению со стандартной таблицей разделов.

```
[user@centos8 ~]$ sudo fdisk -l /dev/md127
Disk /dev/md127: 953 MiB, 999292928 bytes, 1951744 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: E621A24A-4F8B-BD49-BBF3-04CA068EC06A

Device      Start    End Sectors  Size Type
/dev/md127p1  2048  587775  585728  286M Linux filesystem
/dev/md127p2  587776 1951710 1363935  666M Linux filesystem
[user@centos8 ~]$ ls -l /dev/md/
total 0
lrwxrwxrwx. 1 root root 8 Mar 28 19:01 myraid1 -> ../md127
lrwxrwxrwx. 1 root root 10 Mar 28 19:01 myraid1p1 -> ../md127p1
lrwxrwxrwx. 1 root root 10 Mar 28 19:01 myraid1p2 -> ../md127p2
[user@centos8 ~]$ █
```

Чтобы сделать и стандартный раздел и логический том создадим на этом устройстве таблицу разделов:

```
sudo fdisk /dev/md127
g
n
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
enter
enter
+300MB
n
enter
enter
enter
p
w
sudo fdisk -l /dev/md127
```

Как видите, разделы получили названия md127p1 и md127p2. Тоже самое касается ссылки в директории /dev/md:

```
ls -l /dev/md
```

```
[user@centos8 ~]$ sudo mkfs.ext4 /dev/md127p1
mke2fs 1.45.4 (23-Sep-2019)
Creating filesystem with 292864 1k blocks and 73440 inodes
Filesystem UUID: f77f89da-70f0-4339-bb93-55b46ebd88d6
Superblock backups stored on blocks:
      8193, 24577, 40961, 57345, 73729, 204801, 221185

Allocating group tables: done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done
```

На первый раздел сразу запишем файловую систему:

```
sudo mkfs.ext4 /dev/md127p1
```

```
[user@centos8 ~]$ sudo pvcreate /dev/md127p2
  Physical volume "/dev/md127p2" successfully created.
[user@centos8 ~]$ sudo vgcreate myraidvg /dev/md127p2
  Volume group "myraidvg" successfully created
[user@centos8 ~]$ sudo lvcreate myraidvg -n myraidlv -l 80%FREE
  Logical volume "myraidlv" created.
[user@centos8 ~]$ █
```

А на второй создадим логический том. Вспоминаем – физический том – группа томов – логический том:

```
sudo pvcreate /dev/md127p2
sudo vgcreate myraidvg /dev/md127p2
sudo lvcreate myraidvg -n myraidlv -l 80%FREE
```

```
[user@centos8 ~]$ sudo mkfs.ext4 /dev/mapper/myraidvg-myraidlv
mke2fs 1.45.4 (23-Sep-2019)
Creating filesystem with 135168 4k blocks and 33840 inodes
Filesystem UUID: d761e8a3-8821-4dd2-9f53-c8cadb8a9740
Superblock backups stored on blocks:
            32768, 98304

Allocating group tables: done
Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done

[user@centos8 ~]$ sudo nano /etc/fstab
[user@centos8 ~]$ tail -1 /etc/fstab
/dev/mapper/myraidvg-myraidlv /mydata ext4 noatime 0 0
[user@centos8 ~]$ █
```

Смотрим в /dev/mapper - появилась ссылка на логический том. Записываем на него файловую систему:

```
sudo mkfs.ext4 /dev/mapper/myraidvg-myraidlv
```

и подправляем запись в fstab:

```
sudo nano /etc/fstab
tail -1 /etc/fstab
```

```
[user@centos8 ~]$ sudo mount -a
[user@centos8 ~]$ df -h /mydata/
Filesystem              Size  Used Avail Use% Mounted on
/dev/mapper/myraidvg-myraidlv  504M  804K  466M   1% /mydata
[user@centos8 ~]$ █
```

Можно для теста примонтировать:

```
sudo mount -a
df -h /mydata
```

Всё работает. У нас, конечно, прибавился еще один уровень абстракции, но с точки зрения пользователя пространства ничего не изменилось – та же директория /mydata.

```
Raid Devices : 2
Total Devices : 1
    Persistence : Superblock is persistent

        Update Time : Sun Mar 28 19:18:28 2021
                    State : clean, degraded
    Active Devices : 1
Working Devices : 1
Failed Devices : 0
Spare Devices : 0

Consistency Policy : resync

        Name : centos8:myraid1 (local to host centos8)
        UUID : b5d2f62e:5004a01b:e1fb91a7:ed707989
        Events : 19

    Number  Major  Minor  RaidDevice State
        -      0       0       0     removed
        1      8       17      1     active sync   /dev/sdb1
[user@centos8 ~]$ █
```

Напоследок, давайте избавимся от одного диска и посмотрим, что же произойдёт. Для этого я выключаю виртуальную машину и удаляю один из дисков. Далее запускаю систему. После запуска системы, я проверяю, примонтирована ли файловая система:

```
df -h /mydata
```

да, с файловой системой всё окей. Но если посмотреть статус рейда:

```
sudo mdadm -D /dev/md127
```

то можно увидеть, что статус рейда – degraded – а это значит, что есть какие-то проблемы с рейдом, он не в полноценном состоянии. А внизу видно – один из дисков removed.

```
[user@centos8 ~]$ sudo fdisk -l /dev/sdb1
Disk /dev/sdb1: 954 MiB, 1000341504 bytes, 1953792 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
[user@centos8 ~]$ sudo mdadm /dev/md127 --add /dev/sdb1
mdadm: added /dev/sdb1
[user@centos8 ~]$ █
```

Теперь давайте представим, что мы купили новый диск, на замену старому. Опять вырубаю виртуальную машину, создаю ещё один диск – тоже на гигабайт. Хотя тут скорее особенность виртуалбокса, не получится при работе виртуалки добавлять диски. На физических серверах необходимости выключать сервер для добавления диска нет. Для начала определим, какое название получил диск. Для этого выполним команду:

```
sudo fdisk -l
```

и найдём чистый диск. Теперь нужно создать раздел, как мы это уже делали:

```
sudo fdisk /dev/sdb
g
n
enter
enter
+1GB
p
w
```

и добавить его в рейд с помощью ключа add:

```
sudo mdadm /dev/md127 --add /dev/sdb1
```

```
Raid Devices : 2
Total Devices : 2
    Persistence : Superblock is persistent

        Update Time : Sun Mar 28 19:28:42 2021
                    State : clean
        Active Devices : 2
        Working Devices : 2
        Failed Devices : 0
        Spare Devices : 0

Consistency Policy : resync

                    Name : centos8:myraid1 (local to host centos8)
                    UUID : b5d2f62e:5004a01b:e1fb91a7:ed707989
                    Events : 42

      Number  Major  Minor  RaidDevice State
            2        8      17          0    active sync   /dev/sdb1
            1        8      33          1    active sync   /dev/sdc1
```

После добавления статус нового диска поменяется на spare rebuilding:

```
sudo mdadm -D /dev/md127
```

При этом мы можем продолжать работать. Так как у нас информации на дисках не было, процесс прошёл довольно быстро.

И так, мы с вами разобрали, как на Linux настроить программный рейд. Выход из строя диска может создать кучу проблем для администратора, не говоря о других. Это не вопрос вероятности, это вопрос времени. При этом предотвратить один из худших кошмаров можно всего за пару минут, настроив, не важно, программный или аппаратный рейд. И хотя мы рассмотрели только первый рейд, в настройке других уровней рейдов больших отличий нет. Как и в большинстве случаев, за парой простых команд скрывается тонна теории, которую важно знать, потому что неправильное обращение с командами

может привести к печальным последствиям. По [ссылке](#) вы можете более подробно ознакомиться с программный рейдом на Linux.

2.26.2 Практика

Вопросы

1. Зачем нужен RAID?
2. Какие уровни рейда вы знаете и чем они отличаются?
3. Чем отличается программный рейд от аппаратного?
4. Заменяет ли рейд бэкап и почему?
5. Как посмотреть информацию о raid устройстве? Как понять, есть ли проблемы с рейдом?

Задания

1. Создайте raid1 из двух разделов на разных дисках, поверх raid-а настройте LVM с одним логическим разделом, который должен занимать 60% пространства группы, а файловая система которого должна монтироваться в директорию /srv. Выключите виртуалку, удалите один из дисков, добавьте новый диск и восстановите рейд с помощью этого диска.

2.27 27. bash скрипты №1

2.27.1 27. bash скрипты №1

Изучать что-то новое может быть интересно - новые знания, новый опыт. Вот мы изучили рейд, лвм, файловые системы и всё такое, попрактиковались пару раз – всё довольно просто. Но вы устраиваетесь на работу, там есть какая-то инфраструктура и ваша задача – администрировать эту инфраструктуру. И то что вчера было новым и интересным превращается в рутину – вам постоянно нужно работать с одними и теми же командами, изо дня в день, годами. Где-то пользователей создать и права настроить, где-то бэкап сделать, где-то ещё что. И люди ищут способы, как бы это всё автоматизировать. Автоматизировать графический интерфейс довольно сложно. Писать новую программу ради какой-то рутины не всегда стоит того. И тут текстовый интерфейс раскрывает себя во всей красе. Можно написать так называемые сценарии, какие команды и как будут выполняться. И запускать этот сценарий вручную, либо автоматизировать его запуск, что вообще позволит админу избавиться от рутины.

Мы работаем с bash, поэтому будем учиться писать сценарии для него. Более привычное название – скрипты. Кто-то называет это «bash программированием». И давайте сразу поставим себе задачу, которую будем решать с помощью скрипта. Начнём с чего-то простого – создать двух пользователей:

user1 :

1. Основная группа - it
2. it – группа с правами суперпользователя
3. Домашняя директория в /home/it
4. Оболочка - /bin/bash

user2 :

1. Основная группа - users
2. Домашняя директория в /home/users
3. Оболочка - /sbin/nologin

Один из них — user1 — его основная группа будет it, у которой будут права суперпользователя, его домашняя директория будет внутри директории /home/it и его оболочка будет bash. Второй пользователь — user2 — будет в группе users, домашняя директория внутри директории /home/users, а оболочка nologin. Кстати, попробуйте самостоятельно вспомнить все команды, которые необходимо выполнить.

```
[user@centos8 ~]$ sudo groupadd it
[sudo] password for user:
[user@centos8 ~]$ grep users /etc/group
users:x:100:
[user@centos8 ~]$ sudo visudo
[user@centos8 ~]$ sudo grep "%it" /etc/sudoers
%it    ALL=(ALL)      ALL
[user@centos8 ~]$ sudo mkdir -v /home/{it,users}
mkdir: created directory '/home/it'
mkdir: created directory '/home/users'
[user@centos8 ~]$ sudo useradd user1 -g it -b /home/it -s /bin/bash
[user@centos8 ~]$ sudo useradd user2 -g users -b /home/users -s /sbin/nologin
[user@centos8 ~]$ tail -2 /etc/passwd
user1:x:1112:1004::/home/it/user1:/bin/bash
user2:x:1113:100::/home/users/user2:/sbin/nologin
```

Для начала вспомним, как это делается. Во первых, нужно создать группы и директории. Создаём группу it:

```
sudo groupadd it
```

Группа users есть по умолчанию:

```
grep users /etc/group
```

У группы it должны быть права суперпользователя — заходим в:

```
visudo
```

и добавляем строчку:

```
%it ALL=(ALL) ALL
```

Дальше нам нужны директории — /home/it и /home/users:

```
sudo mkdir -v /home/{it,users}
```

И наконец создаём пользователей:

```
sudo useradd user1 -g it -b /home/it -s /bin/bash
sudo useradd user2 -g users -b /home/users -s /sbin/nologin
tail /etc/passwd
```

Всего у нас получилось 5 команд:

```
sudo groupadd it
sudo visudo
sudo mkdir -v /home/{it,users}
sudo useradd user1 -g it -b /home/it -s /bin/bash
sudo useradd user2 -g users -b /home/users -s /sbin/nologin
```

```
GNU nano 2.9.8                                myscript

1 sudo groupadd it
2 sudo visudo
3 sudo mkdir -v /home/{it,users}
4 sudo useradd user1 -g it -b /home/it -s /bin/bash
5 sudo useradd user2 -g users -b /home/users -s /sbin/nologin
6
7
```

Теперь давайте сделаем скрипт – просто вставим все эти 5 команд в файл с любым названием:

```
nano myscript
```

Каждая команда с новой строки.

```
[user@centos8 ~]$ sudo userdel -r user1
[user@centos8 ~]$ sudo userdel -r user2
userdel: group user2 not removed because it is not the primary group of user user2.
[user@centos8 ~]$ sudo groupdel it
[user@centos8 ~]$ sudo rm -rf /home/{it,users}
[user@centos8 ~]$ sudo visudo
[user@centos8 ~]$ sudo grep "%it" /etc/sudoers
[user@centos8 ~]$ █
```

Но прежде чем запускать скрипт, надо вернуть всё как было – удаляем пользователей, группу, директории и запись из sudoers:

```
sudo userdel -r user1
sudo userdel -r user2
sudo groupdel it
sudo rm -r /home/{it,users}
sudo visudo
sudo grep «%it» /etc/sudoers
```

```
[user@centos8 ~]$ bash myscript
[sudo] password for user:
mkdir: created directory '/home/it'
mkdir: created directory '/home/users'
[user@centos8 ~]$ tail -2 /etc/passwd
user1:x:1112:1004::/home/it/user1:/bin/bash
user2:x:1113:100::/home/users/user2:/sbin/nologin
```

Запускаем скрипт с помощью bash:

```
bash myscript
```

У меня открылся visudo – это один из шагов, когда мне нужно добавить группу it в sudoers. Добавил, сохранил. Теперь опять проверяем:

```
tail /etc/passwd
```

всё создалось. И это одной командной, не считая ручного заполнения visudo.

```
GNU nano 2.9.8                                myscript

sudo groupadd it
sudo echo '%it ALL=(ALL) ALL' >> /etc/sudoers
sudo mkdir -v /home/{it,users}
sudo useradd user1 -g it -b /home/it -s /bin/bash
sudo useradd user2 -g users -b /home/users -s /sbin/nologin
```

Но.. мы говорим про автоматизацию, а мне всё равно приходится добавлять записи в sudoers вручную. Давайте сделаем так, чтобы запись добавлялась без нашего участия. Помните, как мы делали с рейдом – писали echo текст и направляли в файл? Сделаем точно также. Заменим в нашем файле visudo на echo:

```
nano myscript
```

```
echo %it ALL=(ALL) ALL
```

Возьмём текст в одинарные кавычки, чтобы баш никак не обработал команду и направим этот текст в файл /etc/sudoers:

```
echo '%it ALL=(ALL) ALL' >> /etc/sudoers
```

с помощью двух символов больше, чтобы не перезаписать файл, а дополнить. Правда сама такая команда не сработает – перенаправление вывода (`>>`) выполняется от моей оболочки, не от команды sudo, а у моего пользователя не хватит прав записать что-то в sudoers.

```
sudo echo '%it ALL=(ALL) ALL' >> /etc/sudoers
```

```
[user@centos8 ~]$ nano myscript
[user@centos8 ~]$ cat myscript
groupadd it
echo '%it ALL=(ALL) ALL' >> /etc/sudoers
mkdir -v /home/{it,users}
useradd user1 -g it -b /home/it -s /bin/bash
useradd user2 -g users -b /home/users -s /sbin/nologin

[user@centos8 ~]$ sudo bash myscript
```

Можно, конечно, использовать команду tee, но, вообще, учитывая, что все команды тут выполняются с sudo, легче сделать по другому. Просто сотрём отсюда все sudo, а при запуске, вместо bash myscript будем писать:

```
sudo bash myscript
```

Тогда весь скрипт будет выполнять в оболочку суперпользователя – это и избавит нас от проблемы с перенаправлением вывода и не будет необходимости внутри скрипта запускать sudo.

```
[user@centos8 ~]$ nano myscript
[user@centos8 ~]$ cat myscript
cp /etc/sudoers{,.bkp}
groupadd it
echo '%it ALL=(ALL) ALL' >> /etc/sudoers
mkdir -v /home/{it,users}
useradd user1 -g it -b /home/it -s /bin/bash
useradd user2 -g users -b /home/users -s /sbin/nologin
[user@centos8 ~]$ sudo userdel -r user1; sudo userdel -r user2; sudo groupdel it; sudo rm -rf /home/{it,users}; sudo visudo
[sudo] password for user:
userdel: group user2 not removed because it is not the primary group of user user2.
```

Но неплохо было бы на всякий случай сделать копию файла sudoers, чтобы случайно его не испортить. Добавим команду в начале скрипта:

```
nano myscript
```

```
cp /etc/sudoers{,.bkp}
```

Хорошо, теперь опять удалим то что создали:

```
sudo userdel -r user1
sudo userdel -r user2
sudo groupdel it
sudo rm -r /home/{it,users}
sudo visudo
cp /etc/sudoers{,.bkp}
```

```
[user@centos8 ~]$ sudo bash myscript
mkdir: created directory '/home/it'
mkdir: created directory '/home/users'
[user@centos8 ~]$ tail -2 /etc/passwd
user1:x:1112:1004::/home/it/user1:/bin/bash
user2:x:1113:100::/home/users/user2:/sbin/nologin
[user@centos8 ~]$ sudo tail -1 /etc/sudoers
%it ALL=(ALL) ALL
[user@centos8 ~]$
```

И заново запустим скрипт:

```
sudo bash myscript
tail -2 /etc/passwd
sudo tail -1 /etc/sudoers
```

Сработало – одной командой мы создали всё что нам нужно.

GNU nano 2.9.8

myscript

```
#!/bin/bash
cp /etc/sudoers{,.bkp}
groupadd it
echo '%it ALL=(ALL) ALL' >> /etc/sudoers
mkdir -v /home/{it,users}
useradd user1 -g it -b /home/it -s /bin/bash
useradd user2 -g users -b /home/users -s /sbin/nologin
```

Чтобы наш скрипт выглядел более самостоятельным, то есть, чтобы нам не приходилось каждый раз перед ним писать bash, а также чтобы мы могли делиться этим скриптом с другими, кто, возможно, использует другие оболочки, мы можем внутри самого скрипта указать, а под какой именно интерпретатор написан скрипт. Так как, теоретически, у нас в скрипте могут быть всякие особенности, присущие только bash, которых нет в других интерпретаторах. Но тут могут быть нюансы – нужно понимать, что bash есть во многих системах, но не во всех. Если вы пишете скрипт, который предполагает использовать не только на GNU/Linux, но и на всяких UNIX-ах, где может не быть bash-а, то лучше писать скрипты под интерпретатор shell. Чтобы указать, с помощью какого интерпретатора запускать скрипт, в первой строчке самого скрипта указываются два символа:

#!

называемые шебанг, после чего указывается путь к интерпретатору, допустим:

#!/bin/bash

```
[user@centos8 ~]$ nano myscript
[user@centos8 ~]$ head -1 myscript
#!/usr/bin/env bash
[user@centos8 ~]$ chmod +x myscript
[user@centos8 ~]$ sudo myscript
[sudo] password for user:
sudo: myscript: command not found
[user@centos8 ~]$ sudo ./myscript
```

Но так как в разных дистрибутивах и системах bash или другой интерпретатор могут находиться в разных директориях, есть более универсальный способ написания пути:

#!/usr/bin/env bash

Давайте так и оставим. Правда, так как теперь мы будем запускать скрипт напрямую, а не передавая его программе bash, ему понадобятся права для запуска – для этого пишем:

chmod +x myscript

Но написать:

```
sudo myscript
```

не получится – sudo будет искать программу myscript в директориях её переменной PATH. Либо нужно поместить myscript в одну из директорий переменной PATH, либо явно указывать путь к этому скрипту – достаточно поставить перед ним точку и слеш, если он в текущей директории:

```
sudo ./myscript
```

```
#!/usr/bin/env bash

user1=user1
user2=user2
group1=it
group2=users

cp /etc/sudoers{,.bkp}
groupadd $group1
echo '%'$group1' ALL=(ALL) ALL' >> /etc/sudoers
mkdir -v /home/{$group1,$group2}
useradd $user1 -g $group1 -b /home/$group1 -s /bin/bash
useradd $user2 -g $group2 -b /home/$group2 -s /sbin/nologin
```

Но наш скрипт получился слишком однозадачным – он нацелен на создание двух конкретных пользователей. Сойдёт на разок, но потом, если понадобится создать ещё пользователей, придётся изменять скрипт. Давайте сделаем наш скрипт более динамическим – добавим в него переменные. Пользователей и группы заменим на переменные – user1,user2,group1,group2, а так как директории совпадают с именами групп, там тоже используем переменные. В строке echo одинарные кавычки, а значит переменная не обрабатывается. Надо немного переделать строчку. Можно по разному, но я просто оставил переменную group1 за кавычками. Теперь, если мне понадобится выполнить скрипт для других пользователей и групп, я могу просто заменить в начале скрипта значения переменных, вместо того, чтобы переделывать весь скрипт.

```
#!/usr/bin/env bash

user=user1
group=it

cp /etc/sudoers{,.bkp}
groupadd $group
echo '%'$group' ALL=(ALL) ALL' >> /etc/sudoers
mkdir -v /home/$group
useradd $user -g $group -b /home/$group -s /bin/bash
```

Опять же получилось так себе – у нас тут скрипт на двух пользователей в двух группах – а если мне нужно добавить одного или трёх пользователей? Давайте уберём всё что касается user2 и group2. Конечно, для второго пользователя придётся опять запускать скрипт, но динамичность скрипта это

компенсирует.

Хотя.. в скрипте всё ещё вручную прописаны переменные user и group. Придётся для каждого пользователя заходить и менять значения. Обычно, когда мы работаем с программами в командной строке, мы передаём ей какие-то параметры. Нельзя ли пользователя и группу передать в виде параметров, чтоб не приходилось менять скрипт? Можно.

```
[user@centos8 ~]$ nano test
[user@centos8 ~]$ chmod +x test
[user@centos8 ~]$ ./test

[user@centos8 ~]$ ./test a b c d e
a b c
[user@centos8 ~]$ cat test
#!/usr/bin/env bash

echo $1 $2 $3
[user@centos8 ~]$ █
```

Давайте сделаем так – создадим новый скрипт:

```
nano test
```

```
#!/usr/bin/env bash
echo $1 $2 $3
```

сохраним и дадим права:

```
chmod +x test
```

Запустим:

```
./test
```

ничего. Теперь напишем после скрипта какие-то параметры:

```
./test a b c d e
```

Как видите, первые три параметра вывелись. Что же произошло? Переменные 1, 2 и 3 в виде значений получили соответствующие аргументы:

```
cat test
```

```
#!/usr/bin/env bash
```

```
user=$1
group=$2
```

```
[user@centos8 ~]$ nano myscript
[user@centos8 ~]$ sudo ./myscript user3 it
groupadd: group 'it' already exists
mkdir: cannot create directory '/home/it': File exists
[user@centos8 ~]$ tail -1 /etc/passwd
user3:x:1114:1004::/home/it:/user3:/bin/bash
[user@centos8 ~]$
```

Вернёмся к нашему скрипту. Заменим значение переменной user на \$1, а group на \$2:

```
user=$1
group=$2
```

Теперь запустим наш скрипт с двумя аргументами – первый – это имя пользователя, второй – имя группы:

```
sudo ./myscript user3 it
tail -1 /etc/passwd
```

Всё сработало – у нас появился user3.

```
GNU nano 2.9.8                                     myscript

#!/usr/bin/env bash

echo Welcome!
read -p "Print username: " user
read -p "Print groupname: " group

cp /etc/sudoers{,.bkp}
groupadd $group
echo '%'$group' ALL=(ALL) ALL' >> /etc/sudoers
mkdir -v /home/$group
useradd $user -g $group -b /home/$group -s /bin/bash
```

Можно сделать наоборот – добавить интерактивности, как это, например, в fdisk. Допустим, сделать так, чтобы скрипт при запуске спрашивал имя пользователя и группу. Добавим для красоты надпись в начале:

```
echo Welcome!
```

Дальше используем команду read, которая будет запрашивать ввод и сохранять его в переменной, с ключом p – которая выведет текст на экран. Допустим:

```
read -p "Print username: " user
read -p "Print groupname: " group
```

Здесь user и group – это переменные, которые получат своё значение от введённого текста.

```
[user@centos8 ~]$ nano myscript
[user@centos8 ~]$ sudo ./myscript
Welcome!
Print username: user4
Print groupname: it
groupadd: group 'it' already exists
mkdir: cannot create directory '/home/it': File exists
[user@centos8 ~]$ tail -1 /etc/passwd
user4:x:1115:1004::/home/it/user4:/bin/bash
[user@centos8 ~]$ █
```

Давайте запустим скрипт и проверим:

```
sudo ./myscript
tail /etc/passwd
```

Всё сработало. Тут, конечно, есть ещё над чем поработать, но это оставим на следующий раз.

Подведём итоги. Мы начали разбирать скрипты. Это, конечно, совсем азы, но тут есть с чем попрактиковаться. Вспомните старые занятия – права, файлы, диски. Вспомните команды, сделайте из них скрипты – что-то интерактивное, что-то не интерактивное. Это большая тема, мы будем к ней возвращаться не раз. Но, в целом, чтобы хорошо писать скрипты нужно много много практиковаться. По [ссылке](#) можете найти цикл статей по скрипtingу на bash.

2.27.2 Практика

Вопросы

1. Для чего нужны скрипты?
2. Что из себя представляет bash скрипт?
3. Для чего нужен шебанг?
4. Как можно запускать bash скрипты?
5. Как передать аргументы из командной строки в скрипт?
6. Как сделать так, чтобы скрипт задавал вопросы и на основе ответов формировал итоговую команду?

Задания

1. Сделайте скрипт, который будет спрашивать имя пользователя и на основе ввода показывать нужную строчку из /etc/passwd.
2. Сделайте скрипт, который будет спрашивать название и создавать файл, затем спрашивать права для этого файла и задавать их.
3. Сделайте скрипт, который будет спрашивать имя пользователя и выводить оболочку этого пользователя.

2.28 28. bash скрипты №2

2.28.1 28. bash скрипты №2

```
[user@centos8 ~]$ cat myscript
#!/usr/bin/env bash

echo Welcome!
read -p "Print username: " user
read -p "Print groupname: " group

cp /etc/sudoers{,.bkp}
groupadd $group
echo '%'$group' ALL=(ALL) ALL' >> /etc/sudoers
mkdir -v /home/$group
useradd $user -g $group -b /home/$group -s /bin/bash
[user@centos8 ~]$ █
```

В прошлый раз мы остановились на том, что сделали скрипт более универсальным – вместо двух определённых пользователей мы теперь можем создавать разных пользователей, используя переменные:

```
cat myscript
```

Но сделав это мы перестали соответствовать начальным требованиям. Во-первых, мы говорили, что если группа it, её следует предварительно добавить в sudoers. Но в нашем скрипте всего одна переменная group и какое бы значение она не получила, она всё равно добавляется в sudoers. Если я буду добавлять пользователя с группой users, то она тоже попадёт в sudoers. Во-вторых, мы говорили, что оболочка bash только у пользователей группы it, у users должен быть nogroup, значит сейчас скрипт подходит только для группы it. Для users мы можем создать отдельный скрипт, подходящий им. Но также можем это сделать в рамках одного скрипта.

Начнём с sudoers. Сформулируем задачу: если переменная group получит значение it, то нужно добавить такую-то строчку в sudoers. Если же переменная group получит любое другое значение, ничего в sudoers добавлять не надо. И так, у нас появилось слово «если» – это значит, что мы имеем дело с условиями. bash умеет работать с условиями, для этого у него есть команда if. Синтаксис команды выглядит так - if условие then команда, которую мы хотим выполнить, в случае если условие выполнилось, и в конце fi, чтобы показать, где у нас заканчивается команда if:

```
if условие
then команда
fi
```

Если с командой понятно, то чем же является условие? Тоже командой – cp, ls, grep и всё в таком духе, то есть просто команда. Вы спросите – if ls, где тут условие? ls просто покажет список файлов, о каком условии идёт речь? На самом деле, if интересует не сама команда, а результат её выполнения, так называемый «код выхода» или «статус выхода». Обычно это числа от 0 до 255.

Мы упоминали статус выхода, когда говорили про зомби процессы – дочерний процесс при завершении должен передать родительскому свой статус выхода. Зачем? Это принятый в программировании способ родительским процессам узнать, как выполнился дочерний процесс, без необходимости копаться в выводе дочернего процесса. Для процесса, который выполнился без всяких проблем, обычно статус выхода – 0. Если же что-то пошло не так, то статус выхода другой, и по нему иногда легче понять причину проблемы.

```
[user@centos8 ~]$ ls
Desktop Documents filelist Music Pictures snmpd.conf Templates
dir1 Downloads hw1.sh myscript Public snmptrapd.conf test
dir3 errors hw.sh output snmp.conf temp Videos
[user@centos8 ~]$ echo $?
0
[user@centos8 ~]$ ls file
ls: cannot access 'file': No such file or directory
[user@centos8 ~]$ echo $?
2
[user@centos8 ~]$ █
```

В bash-е с помощью специальной переменной:

```
$?
```

можно узнать статус выхода последней выполненной команды. Для примера возьмём команду ls. Просто выполнив команду и посмотрев значение переменной:

```
ls
echo $?
```

мы увидим, что код выхода – 0. Теперь давайте выполним команду:

```
ls file
```

Команда ругается, что такого файла нет. Посмотрим статус выхода ещё раз:

```
echo $?
```

Как видите, теперь он 2. Есть определённые правила, в каких случаях какие числа пишутся, но сейчас нас интересует только 0.

```
[user@centos8 ~]$ nano test
[user@centos8 ~]$ cat test
if ls
then echo Success, because exit code is $?
fi
[user@centos8 ~]$ ./test
Desktop Documents filelist Music Pictures snmpd.conf Templates
dir1 Downloads hw1.sh myscript Public snmptrapd.conf test
dir3 errors hw.sh output snmp.conf temp Videos
Success, because exit code is 0
[user@centos8 ~]$ █
```

Если процесс завершился с кодом 0, то всё окей. Все остальные коды считаются за ошибку. Именно так думает if. Давайте напишем маленький пример:

```
nano test
```

```
if ls  
then echo Success, because exit code is $?  
fi
```

```
cat test  
.test
```

Как видите, сработала команда ls, после чего команда echo.

```
[user@centos8 ~]$ nano test  
[user@centos8 ~]$ cat test  
if ls > /dev/null  
then echo Success, because exit code is $?  
fi  
[user@centos8 ~]$ █
```

Давайте направим вывод команды ls в никуда, чтобы не было лишней информации:

```
nano test
```

```
if ls > /dev/null  
then echo File exists  
fi
```

```
./test
```

```
[user@centos8 ~]$ nano test  
[user@centos8 ~]$ cat test  
if ls file > /dev/null  
then echo Success, because exit code is $?  
fi  
[user@centos8 ~]$ ./test  
ls: cannot access 'file': No such file or directory  
[user@centos8 ~]$ █
```

Теперь сделаем так, чтобы if получила не 0, а что-то другое:

```
nano test
```

```
if ls file > /dev/null  
then echo File exists  
fi
```

```
./test
```

Как видите, теперь у нас условие выдаёт не 0, а значит команда после then не сработает.

```
[user@centos8 ~]$ nano test
[user@centos8 ~]$ cat test
if ls file > /dev/null
then echo Success, because exit code is $?
else echo Failure, because exit status is $?
fi
[user@centos8 ~]$ ./test
ls: cannot access 'file': No such file or directory
Failure, because exit status is 2
[user@centos8 ~]$ █
```

Можно ещё if дополнить с помощью else, которая будет запускать команду, если в if условие не срабатывает. Выглядит это так:

```
if условие
then команда, если 0
else команда, если не 0
fi
```

Например:

```
if ls file
then echo Success, because exit code is $?
else echo Failure, because exit status is $?
fi
```

```
cat test
./test
```

Как видите, сработала вторая команда.

Окей, по какому принципу работает if разобрались. Но этого недостаточно – нам нужно узнать, переменная group получила в качестве значение it или что-то другое. То есть нам нужна команда, которая сравнит значение переменной с каким-то текстом. Если всё окей, у неё статус выхода будет 0 и тогда if выполнит нужную команду. Такая команда есть и она, можно сказать, специально написана для if. Команда выглядит как открывающаяся квадратная скобка:

```
[
```

а после выражения внутри ставится закрывающаяся квадратная скобка:

```
[ выражение ]
```

```
[user@centos8 ~]$ ll /usr/bin/[  
-rwxr-xr-x. 1 root root 55048 Jun 10 2020 '/usr/bin/[ '  
[user@centos8 ~]$ type [  
[ is a shell builtin  
[user@centos8 ~]$ type test  
test is a shell builtin  
[user@centos8 ~]$ type [[  
[[ is a shell keyword  
[user@centos8 ~]$ █
```

Причём даже есть такая программа в /usr/bin:

```
ll /usr/bin/[
```

но у bash обычно есть встроенная версия этой программы:

```
type [
```

а внешняя программа может понадобится для других оболочек. Есть такая же программа с названием test:

```
type test
```

и даже более продвинутая версия с двумя скобками:

```
type [[
```

но она есть на bash и паре других оболочек, а не на всех. В общем, если хотим, чтобы нашими скриптами можно было пользоваться не только на bash, будем обходиться без двойных скобок.

С помощью квадратных скобок можно сравнивать много чего – строки, числа, наличие файлов, директорий, их права и многое другое. Программа сравнивает, выдаёт значение 0 или 1, а дальше if действует так, как мы обсуждали. По [ссылке](#) показан синтаксис для различных сравнений, а пока посмотрим нашу ситуацию.

```
[user@centos8 ~]$ [ $group = it ]
bash: [: =: unary operator expected
[user@centos8 ~]$ group=users
[user@centos8 ~]$ [ $group = it ]
[user@centos8 ~]$ echo $?
1
[user@centos8 ~]$ group=it
[user@centos8 ~]$ [ $group = it ]
[user@centos8 ~]$ echo $?
0
[user@centos8 ~]$ █
```

Нам нужно сравнить переменную group и текст it:

```
[ $group = it ]
```

Для примера, дадим переменной значение users, сравним и посмотрим код выхода:

```
group=users
[ $group = it ]
echo $?
```

Как видите = 1, то есть неправильно. Теперь дадим переменной значение it, сравним и посмотрим код выхода:

```
group=it
[ $group = it ]
echo $?
```

0 – значит всё правильно.

```
[user@centos8 ~]$ group="a b c"
[user@centos8 ~]$ [ $group = it ]
bash: [: too many arguments
[user@centos8 ~]$ [ "$group" = it ]
[user@centos8 ~]$ █
```

Кстати, важное замечание, если в переменной есть пробелы, то сравнение работать не будет:

```
group="a b c"
[ $group = it ]
```

так как bash превратит нашу переменную в её значение и в выражении получится много параметров, типа:

```
[ a b c = it ]
```

а это не подходит под синтаксис. Поэтому лучше переменные брать в кавычки:

```
[ "$group" = it ]
```

Хорошо, с выражением разобрались, вернёмся к нашему скрипту и попробуем его добавить.

```
GNU nano 2.9.8 myscript

#!/usr/bin/env bash

echo Welcome!
read -p "Print username: " user
read -p "Print groupname: " group

groupadd $group
if [ "$group" = it ]
then
    cp /etc/sudoers{,.bkp}
    echo '%'$group' ALL=(ALL) ALL' >> /etc/sudoers
fi
mkdir -v /home/$group
useradd $user -g $group -b /home/$group -s /bin/bash
```

Вспомним наше условие - если переменная group получит значение it, то нужно добавить такую-то строчку в sudoers:

```
nano myscript
```

Находим нашу строчку с командой echo и окружаем её условием:

```
if [ "$group" = it ]
then
...
fi
```

Тут else нам не нужен, а команды после then можно чуток подвинуть направо с помощью пробелов, чтобы было проще для глаз.

```
GNU nano 2.9.8                               myscript

#!/usr/bin/env bash

shell=/sbin/nologin

echo Welcome!
read -p "Print username: " user
read -p "Print groupname: " group

groupadd $group
if [ "$group" = it ]
then
    cp /etc/sudoers{,.bkp}
    echo '%'$group' ALL=(ALL) ALL' >> /etc/sudoers
    shell=/bin/bash
fi
mkdir -v /home/$group
useradd $user -g $group -b /home/$group -s $shell
```

Теперь вспомним второе условие – оболочка bash только у пользователей группы it, у users должен быть nologin. Тоже довольно просто – наверху создаём переменную shell, чтобы легче было её поменять при надобности и даём ей значение:

```
shell=/sbin/nologin
```

А в if добавляем:

```
shell=/bin/bash
```

И не забываем в конце указать эту переменную в качестве оболочки:

```
-s $shell
```

Если у нас группа будет it, то значение переменной в if поменяется на bash, а если не it, то условие не выполнится и переменная shell останется nologin.

```
[user@centos8 ~]$ sudo ./myscript
[sudo] password for user:
Welcome!
Print username: user5
Print groupname: users
groupadd: group 'users' already exists
mkdir: cannot create directory '/home/users': File exists
[user@centos8 ~]$ sudo ./myscript
Welcome!
Print username: user6
Print groupname: it
groupadd: group 'it' already exists
mkdir: cannot create directory '/home/it': File exists
[user@centos8 ~]$ tail -2 /etc/passwd
user5:x:1116:100::/home/users/user5:/sbin/nologin
user6:x:1117:1004::/home/it/user6:/bin/bash
[user@centos8 ~]$ █
```

Попробуем запустить скрипт 2 раза – один раз для пользователя с группой `it`, а потом для пользователя с группой `users`:

```
sudo ./myscript
tail /etc/passwd
```

Всё сработало.

```
[user@centos8 ~]$ sudo tail -5 /etc/sudoers
%it  ALL=(ALL)  ALL
[user@centos8 ~]$ █
```

Но если посмотреть `sudoers`:

```
sudo tail -5 /etc/sudoers
```

можно заметить, что строчка `it` повторяется много раз, хотя одной строчки вполне достаточно.

Давайте решим и эту проблему. Сформулируем задачу – если мы добавляем пользователя с группой `it` и если в `sudoers` нет нужной строчки – то её нужно создать. То есть, если выполнилось первое условие, нужно проверить второе условие и если оно тоже выполнилось – то только тогда добавлять строчку.

```
[user@centos8 ~]$ sudo grep '%it' /etc/sudoers
%it ALL=(ALL) ALL
[user@centos8 ~]$ echo $?
0
[user@centos8 ~]$ sudo grep '%itaa' /etc/sudoers
[user@centos8 ~]$ echo $?
1
[user@centos8 ~]$ █
```

Первое условие у нас уже прописано. После его выполнения, то есть после then пишем ещё один if. Теперь нужно написать условие – если такой-то строчки нет в sudoers. Поиском строчек занимается команда grep. Посмотрим, что он нам выдаёт, если мы попытаемся найти строчку. Необязательно искать всю строчку, если есть что-то про группу it, то этого достаточно:

```
sudo grep '%it' /etc/sudoers
```

И посмотрим код выхода:

```
echo $?
```

Ноль. Посмотрим, что будет, если он не найдёт строчку – напишем в grep какую-то несуществующую группу, чтобы он ничего не нашёл:

```
sudo grep '%itaa' /etc/sudoers
```

и посмотрим ещё раз:

```
echo $?
```

Один.

```
[user@centos8 ~]$ if ls file
> then echo File exists
> fi
ls: cannot access 'file': No such file or directory
[user@centos8 ~]$ █
```

И так, если строчек нет, то grep выдаёт 1, если есть – 0. Мне же нужно, чтобы запись создавалась, если строчек нет, то есть grep должен выдать 1, а if должен получить 0. Чтобы вот так вот перевернуть полученное значение, нужно после if поставить восклицательный знак. Давайте проверим на том же ls, чтобы было проще. И так, если ls не выполнился, то команда после then не должна сработать:

```
if ls file
then echo File exists
fi
```

Как видите, ls выдал ошибку и условие провалилось.

```
[user@centos8 ~]$ if ! ls file
> then echo File exists
> fi
ls: cannot access 'file': No such file or directory
File exists
[user@centos8 ~]$ █
```

Но если мы после if поставим восклицательный знак:

```
if ! ls file
then echo File exists
fi
```

то ls, опять же, не сработал, но if перевернул значение, условие выполнилось и команда сработала.

```
if [ "$group" = it ]
then
    if ! grep "%$group" /etc/sudoers
    then
        cp /etc/sudoers{,.bkp}
        echo '%'$group' ALL=(ALL) ALL' >> /etc/sudoers
    fi
    shell=/bin/bash
fi
```

Попробуем тоже самое с нашим скриптом:

```
if ! grep "%$group" /etc/sudoers"
...
```

И так, прочтём – если переменная group равна it запускается второе условие – если grep не нашёл упоминание группы it в sudoers, то следует добавить в sudoers эту группу.

```
[user@centos8 ~]$ sudo visudo
[user@centos8 ~]$ sudo ./myscript
Welcome!
Print username: user7
Print groupname: it
groupadd: group 'it' already exists
mkdir: cannot create directory '/home/it': File exists
[user@centos8 ~]$ sudo ./myscript
Welcome!
Print username: user8
Print groupname: it
groupadd: group 'it' already exists
%it ALL=(ALL) ALL
mkdir: cannot create directory '/home/it': File exists
[user@centos8 ~]$ tail -2 /etc/passwd
user7:x:1118:1004::/home/it/user7:/bin/bash
user8:x:1119:1004::/home/it/user8:/bin/bash
[user@centos8 ~]$ sudo tail -3 /etc/sudoers
#includedir /etc/sudoers.d

%it ALL=(ALL) ALL
```

Давайте проверим. Для начала удалим все упоминания группы it в sudoers:

```
sudo visudo
```

Потом запустим наш скрипт и создадим пользователя с группой it:

```
sudo ./myscript
```

Теперь добавим ещё одного пользователя с группой it:

```
sudo ./myscript
```

Проверяем:

```
sudo tail -3 /etc/sudoers
```

всё также одна строчка. Условие работает.

Мы с вами разобрали условие if, которое добавляет вашим скриптам логики, чтобы они могли проверять какие-то условия и по результатам менять какие-то переменные или выполнять дополнительные команды. Также разобрались, для чего нужны коды выхода и использовали их для команды if. Не забудем и про программу квадратных скобок, которая позволяет нам сравнивать переменные, строки, числа и проверять файлы. Мы ещё разберём, как в команде if проверять другие условия с помощью elif и много чего другого.

2.28.2 Практика

Вопросы

1. Какой синтаксис у команды if?
2. Как узнать код выхода предыдущей команды?
3. Как проверить, равно ли значение какой-то переменной определённому тексту?

4. Почему стоит переменную брать в кавычки при использовании [] ?

Задания

- Сделайте скрипт, который спросит имя пользователя и скажет, является ли пользователь администратором?
- Сделайте скрипт, который проверит bash историю и скажет, использовал ли этот пользователь команду sudo? Если использовал, то скрипт должен дополнительно вывести лог о выполнении из логов sudo.
- Сделайте скрипт, который спросит имя, а потом фамилию. Если результат будет «John Doe», то скрипт должен напечатать «Access Granted».

2.29 29. bash скрипты №3

2.29.1 29. bash скрипты №3

```
[user@centos8 ~]$ cat myscript
#!/usr/bin/env bash

shell=/sbin/nologin

echo Welcome!
read -p "Print username: " user
read -p "Print groupname: " group

groupadd $group
if [ "$group" = it ]
then
    if ! grep "%$group" /etc/sudoers
    then
        cp /etc/sudoers{,.bkp}
        echo '%'$group' ALL=(ALL) ALL' >> /etc/sudoers
    fi
    shell=/bin/bash
fi
mkdir -v /home/$group
useradd $user -g $group -b /home/$group -s $shell
[user@centos8 ~]$ █
```

В прошлый раз мы с вами разобрали if. Есть много других команд, но if ещё кое-где пригодится.

```
[user@centos8 ~]$ ./myscript
Welcome!
Print username: user12
Print groupname: it
groupadd: group 'it' already exists
grep: /etc/sudoers: Permission denied
cp: cannot open '/etc/sudoers' for reading: Permission denied
./myscript: line 15: /etc/sudoers: Permission denied
mkdir: cannot create directory '/home/it': File exists
useradd: Permission denied.
useradd: cannot lock /etc/passwd; try again later.
[user@centos8 ~]$ █
```

Начнём с того, что скрипт, который мы пишем, должен выполняться рутом, так как тут и команда useradd, и добавление строчки в sudoers. В самом скрипте мы нигде не пишем sudo, так как предполагаем, что либо root запустит этот скрипт, либо пользователь с командой sudo. Но что будет, если скрипт запустить без sudo? Выполнятся все команды, правда, некоторые выдадут ошибку, что недостаточно прав. Сейчас у нас скрипт небольшой, можно понять, что ничего страшного не произойдёт. Но в будущем в скрипте могут быть команды, которые могут сработать и у рута, и у обычного пользователя. И если скрипт запустится от обычного пользователя, то некоторые команды выполняются, некоторые нет, в итоге это приведёт к неожиданным последствиям. Поэтому лучше в начале скрипта сделать проверку, а кто именно запускает скрипт. Если это root – то всё окей, если кто-то другой – то нужно остановить скрипт и выдать ошибку, что недостаточно прав.

```
[user@centos8 ~]$ id -u
1000
[user@centos8 ~]$ sudo id -u
0
[user@centos8 ~]$
```

Условие у нас такое – если текущий пользователь является рутом, запускать скрипт, иначе выдавать ошибку. Для этого нужно распознать текущего пользователя – это можно сделать с помощью команды id. Причём, всё завязано не на самом слове root, а на его uid, который равен 0. То есть можно встретить случаи, когда вместо root используется другой пользователь с uid-ом 0, поэтому лучше ориентироваться на uid. Команда id с ключом -u должна быть 0:

```
id -u
sudo id -u
```

```
GNU nano 2.9.8                                         myscript

#!/usr/bin/env bash

if [ "$(id -u)" != 0 ]
then
    echo root permissions required >&2
    exit 1
fi

shell=/sbin/nologin
```

Проверку нужно делать в самом начале, чтобы никакие другие команды не выполнились. Как писать условие мы знаем – if, дальше нам нужно выполнить команду, чтобы узнать uid и убедиться, что он не равен 0:

```
if [ "$(id -u)" != 0 ]
```

И так, если uid не равен нулю, пусть скрипт выдаст ошибку - echo root permissions required - и завершится. Чтобы завершить скрипт, используем команду exit. Также, давайте вспомним, что обычно ошибки ссыпятся в stderr. И чтобы наш скрипт тоже так делал, добавим после echo направление stdout в stderr (>&2), в итоге у нас вывод echo уйдёт в stderr. Также вспомним про статусы выхода. Именно с помощью команды exit мы и будем выдавать статус выхода с ошибкой, то есть единицу:

```
exit 1
```

```
if [ "$(id -u)" != 0 ]
then
    echo root permissions required >&2
    exit 1
fi
```

```
[user@centos8 ~]$ ./myscript
root permissions required
[user@centos8 ~]$ echo $?
1
[user@centos8 ~]$ ./myscript 2> errors
[user@centos8 ~]$ cat errors
root permissions required
[user@centos8 ~]$ nano myscript
[user@centos8 ~]$ head -5 myscript
#!/usr/bin/env bash

# Checking root permissions
if [ "$(id -u)" != 0 ]
then
[user@centos8 ~]$
```

Сохраним и проверим:

```
./myscript
```

Как видите:

```
echo $?
```

статус выхода – 1, т.е. скрипт завершился с ошибкой, как мы и хотели. А чтобы убедиться, что ошибка выводится в stderr, направим stderr в файл:

```
./myscript 2>errors
cat errors
```

Всё работает как надо. Не забудем добавить комментарии к нашему скрипту, чтобы легче было его понимать:

```
nano myscript
```

```
groupadd $group

# Sudoers
if [ "$group" = it -o "$group" = security ]
then
    if ! grep "%$group" /etc/sudoers
```

Теперь давайте переделаем условие по нашей группе. Добавим ещё требований – sudo права теперь должны быть у двух групп – it и security. И если имя пользователя admin, но его группа не it или security, то всё равно у него должны быть sudo права. И так, для групп нам нужно сравнивать переменную group с двумя значениями – it и security. Это можно сделать как в рамках программы скобок,

используя опцию -o, то есть ог:

```
["$group" = it -o "$group" = security ]
```

так и сделать это сравнение на уровне bash-а.

```
[user@centos8 ~]$ ls file1; ls file2
file1
ls: cannot access 'file2': No such file or directory
[user@centos8 ~]$ echo $?
2
[user@centos8 ~]$ ls file1 && ls file2
file1
ls: cannot access 'file2': No such file or directory
[user@centos8 ~]$ echo $?
2
[user@centos8 ~]$ ls file2 && ls file1
ls: cannot access 'file2': No such file or directory
[user@centos8 ~]$ echo $?
2
[user@centos8 ~]$
```

Это работает примерно как с if – bash понимает статусы выхода. Обычно, когда мы хотим выполнить несколько команд разом, без скриптов, мы ставим точку с запятой между командами:

```
ls file1
ls file2
```

И тут даже без if можно добавить логики. Допустим, если мы хотим, чтобы выполнилась первая команда и, если она выполнилась удачно, то есть статус выхода 0, выполнить вторую команду, то между командами мы ставим два амперсанда:

```
ls file1 && ls file2
echo $?
```

Это называется «оператор и». Первая команда выполнилась успешно, вследствие чего выполнилась вторая команда. Теперь перевернём команду:

```
ls file2 && ls file1
echo $?
```

Как видите, первая команда выполнилась неудачно, следствии чего вторая даже не запустилась.

```
[user@centos8 ~]$ ls file1 || ls file2
file1
[user@centos8 ~]$ echo $?
0
[user@centos8 ~]$ ls file2 || ls file1
ls: cannot access 'file2': No such file or directory
file1
[user@centos8 ~]$ echo $?
0
[user@centos8 ~]$ █
```

Теперь сделаем наоборот - если первая команда выполнилась удачно, то вторую даже не запускать, а если первая неудачно, то запустить вторую. Для этого используется «оператор или» – две прямые линии || . Например, в первом случае:

```
ls file1 || ls file2
echo $?
```

первая команда выполнилась успешно, следствии чего вторая команда даже не запустилась. Теперь перевернём:

```
ls file2 || ls file1
echo $?
```

Как видите, первая команда завершилась с ошибкой, вследствие чего запустилась вторая. Обратите внимание, в 3 из 4 случаев (обе команды срабатывают нормально; первая команда выдаёт ошибку; вторая команда выдаёт ошибку; обе команды выдают ошибку) при использовании «оператор и» получается статус выхода не равный 0, а с «оператором или» наоборот.

```
# Sudoers
if [ "$group" = it ] || [ "$group" = security ]
then
    if ! grep "%$group" /etc/sudoers
    then
```

Теперь сделаем тоже самое в скрипте – используем две команды скобок и поставим «оператор или»:

```
if [ "$group" = it] || [ "$group" = security ]
```

Как мы помним, if интересует только статус выхода, и если одно из условий окажется верным, то сработают команды после then.

```
[user@centos8 ~]$ sudo ./myscript
[sudo] password for user:
Welcome!
Print username: user12
Print groupname: security
mkdir: created directory '/home/security'
[user@centos8 ~]$ sudo tail -1 /etc/sudoers
%security ALL=(ALL) ALL
[user@centos8 ~]$
```

Давайте проверим:

```
sudo ./myscript
sudo tail -1 /etc/sudoers
```

Как видите, в sudoers добавилась группа security.

Касательно второй задачи – нам нужно пользователю админ также давать права sudo, если он не в вышеуказанных группах. У нас уже есть условие проверки на группы. Если условие не срабатывает, if закрывается. В прошлый раз я упоминал else – если условие не срабатывает, запускать другую команду. Но нам это не подойдёт – это будет работать для всех пользователей из других групп, а нам нужно, чтобы это работало только у админа. То есть нам опять же нужно проверить условие. И так, если не сработало первое условие – то есть группы другие, то нужно проверить ещё одно условие – пользователя. Для этого используем команду elif, то есть else if. Синтаксис примерно такой:

```
if условие
then команда
elif условие
then команда
elif условие
then команда
else команда
fi
```

Этих elif может быть много. if завершится, когда одно из условий сработает и выполнится команда после then, иначе выполнится else.

```

cp /etc/sudoers{,.bkp}
echo "%$group" ALL=(ALL) ALL' >> /etc/sudoers
fi
shell=/bin/bash
elif [ "$user" = admin ]
then
    if ! grep "$user" /etc/sudoers
    then
        cp /etc/sudoers{,.bkp}
        echo $user' ALL=(ALL) ALL' >> /etc/sudoers
    fi
    shell=/bin/bash
fi

```

В нашем скрипте мы пишем:

```

elif [ "$user" = admin ]
then

```

и копируем строчки проверки и добавления записи в sudoers, заменяя при этом group на user. Можно было бы просто добавить пользователя admin в группу wheel, но на разных дистрибутивах это может быть группа sudo или что-то другое, поэтому надёжнее просто прописать самого пользователя.

```

[user@centos8 ~]$ sudo ./myscript
[sudo] password for user:
Welcome!
Print username: admin
Print groupname: admin
mkdir: created directory '/home/admin'
[user@centos8 ~]$ sudo tail -2 /etc/sudoers
%security ALL=(ALL) ALL
admin ALL=(ALL) ALL
[user@centos8 ~]$ █

```

Окей, теперь проверим:

```

sudo ./myscript
sudo tail -2 /etc/sudoers

```

Как видите, запись создалась.

Пойдёмте дальше. До этого мы могли сделать наш скрипт либо интерактивным, либо вводить параметры при запуске скрипта. Теперь же, с помощью if, мы можем использовать оба варианта. Для этого нам нужно проверять, было ли что-то передано нашему скрипту в виде параметров. Если да, то использовать эти параметры. Если нет, то выводить приглашение ввести данные. И так, задача – проверить, вводил ли пользователь параметры при запуске скрипта.

```
# Check arguments
if [ -z $2 ]
then
    echo Welcome!
    read -p "Print username: " user
    read -p "Print groupname: " group
else
    user=$1
    group=$2
    echo Username: $user      Group: $group
fi
```

Мы знаем, что параметры превращаются в значение переменных \$1, \$2 и т.д. Нам нужно проверить, есть ли у этих переменных значения. Для этого используем тот же if со скобками с опцией -z, которая проверяет, есть ли значение у переменной. z – это zero length, то есть пустая строка. Если у переменной значения нет, то статус выхода 0. Проверяем переменную 2, так как если она есть, есть и первая:

```
[ -z $2 ]
```

И так, если скрипту не передали параметров, или передали всего один параметр, то стоит вывести приглашение ввести данные. Иначе стоит эти параметры использовать как значения переменных user и group. Ну и вывести об этом текст:

```
user=$1
group=$2
echo Username: $user      Group: $group
```

```
[user@centos8 ~]$ sudo ./myscript
[sudo] password for user:
Welcome!
Print username: user16
Print groupname: group16
mkdir: created directory '/home/group16'
[user@centos8 ~]$ sudo ./myscript user17
Welcome!
Print username: user17
Print groupname: group17
mkdir: created directory '/home/group17'
[user@centos8 ~]$ sudo ./myscript user18 group18
Username: user18 Group: group18
mkdir: created directory '/home/group18'
[user@centos8 ~]$ sudo tail -3 /etc/passwd
user16:x:1124:1009::/home/group16/user16:/sbin/nologin
user17:x:1125:1010::/home/group17/user17:/sbin/nologin
user18:x:1126:1011::/home/group18/user18:/sbin/nologin
[user@centos8 ~]$ █
```

Теперь проверяем – сначала без параметров:

```
sudo ./myscript
```

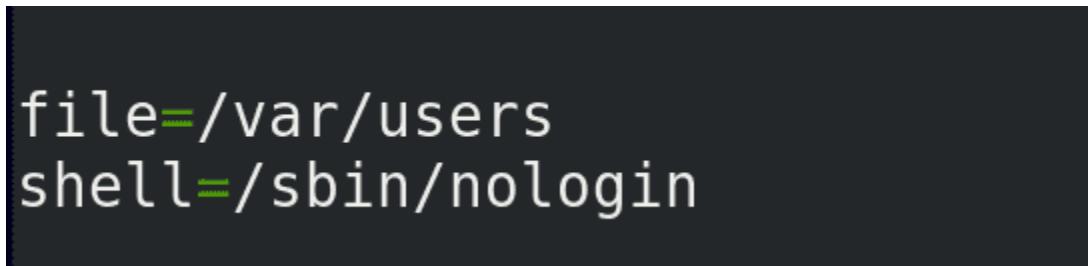
потом с одним:

```
sudo ./myscript user17
```

и наконец с двумя параметрами:

```
sudo ./myscript user18 group18
```

Всё работает. А вы попробуйте самостоятельно сделать так, что если вводится один параметр, скрипт будет использовать его в качестве имени пользователя, а спрашивать только имя группы. Если больше двух параметров – то выдавать ошибку, что синтаксис неправильный и завершаться.



```
file=/var/users
shell=/sbin/nologin
```

И если уж мы это делаем для автоматизации, то нужно сделать скрипт таким, чтобы он не требовал от нас ввода или параметров. Пусть лучше он будет их брать из файла. Предположим, у нас есть компания и HR при приёме на работу заполняет файл, а наш скрипт будет брать из этого файла данные и создавать пользователей. Для начала сделаем что-нибудь примитивное. Во первых, укажем путь к файлу, и зададим его как переменную, чтобы легче было его в дальнейшем менять:

```
file=/var/users
```

Дальше, я хочу оставить способ создания пользователей вручную, поэтому способ с файлом сделаем как одну из опций. То есть, по порядку – если скрипт запустили с параметрами, то использовать эти параметры. Если скрипт запустили без параметров, то пусть он проверит наличие нужного файла. Если файл есть, то пусть скрипт возьмёт данные из файла. Если файла нет, то пусть выведет предложение ввести данные.

```
# Check arguments
if [ ! -z $2 ]
then
    user=$1
    group=$2
    echo Username: $user      Group: $group
elif [ -f $file ]
then
    ...
else
    echo Welcome!
    read -p "Print username: " user
    read -p "Print groupname: " group
fi
```

Для этого я немного поменяю условия. Условие с пустой переменной 2 я переверну, тем самым проверяя, есть ли значение у переменной 2. И её команды поставлю выше. Т.е. если переменная не пустая, то брать значения из аргументов. Дальше использую elif чтобы проверить наличие файла:

```
elif [ -f $file ]
```

А в конце оставил вариант с предложением ввода. Теперь остаётся определиться с файлом.

```
[user@centos8 ~]$ nano myscript
[user@centos8 ~]$ sudo nano /var/users
[sudo] password for user:
[user@centos8 ~]$ cat /var/users
user22 it
[user@centos8 ~]$ cut -d' ' -f1 /var/users
user22
[user@centos8 ~]$ cut -d' ' -f2 /var/users
it
[user@centos8 ~]$ █
```

Создаём файл:

```
sudo nano /var/users
```

и прописываем в одну строчку пользователя и группу:

```
user22 it
```

Теперь нам нужно из этого файла достать эти данные и назначить их в качестве значений переменным. Для этого используем команду cut. Делителем у нас выступает пробел. Столбик 1 это имя пользователя:

```
cut -d' ' -f1 /var/users
```

а столбик два – группа:

```
cut -d' ' -f2 /var/users
```

```
# Check arguments
if [ ! -z $2 ]
then
    user=$1
    group=$2
    echo Username: $user      Group: $group
elif [ -f $file ]
then
    user=$(cut -d' ' -f1 $file)
    group=$(cut -d' ' -f2 $file)
    echo Username: $user      Group: $group
else
    echo Welcome!
    read -p "Print username: " user
    read -p "Print groupname: " group
fi
```

Укажем это в скрипте:

```
user=$(cut -d' ' -f1 $file)
group=$(cut -d' ' -f2 $file)
```

И в конце добавим строчку, чтобы показывала значения:

```
echo Username: $user      Group: $group
```

```
[user@centos8 ~]$ sudo ./myscript
[sudo] password for user:
Username: user22 Group: it
groupadd: group 'it' already exists
%it ALL=(ALL) ALL
mkdir: cannot create directory '/home/it': File exists
[user@centos8 ~]$ tail -1 /etc/passwd
user22:x:1127:1004::/home/it/user22:/bin/bash
[user@centos8 ~]$ █
```

Проверим скрипт:

```
sudo ./myscript
tail -1 /etc/passwd
```

Всё создалось – пользователь 22 с группой it.

В будущем мы разберём, как с помощью одного файла создавать много пользователей, сделаем файл посложнее, добавим больше данных, больше возможностей. Обязательно практикуйтесь – придумывайте себе задания, какими-бы они сложными не казались, любая попытка – это опыт.

2.29.2 Практика

Вопросы

1. Как с помощью if можно проверить несколько условий разом?
2. Как сделать, чтобы команда запустилась если оба условия выполнились? Или выполнилось только одно условие?

Задания

1. Создайте скрипт, который запрашивает имя пользователя, а в ответ выводит информацию о пользователе – его uid, домашнюю директорию, и список групп, в которых он состоит. После этого скрипт должен спросить, что следует поменять – uid, домашнюю директорию или группу. Если uid, то сначала проверить, доступен ли такой uid, если нет – то один раз предложить ввести заново. Если домашнюю директорию, то спросить, на какую директорию следует сменить, а также следует ли перемещать домашнюю директорию. Если группу – то следует спросить, меняем ли мы основную группу или дополнительную. После чего следует вывести на экран итоговую команду.

2.30 30. bash скрипты №4

2.30.1 30. bash скрипты №4

```
[user@centos8 ~]$ sudo nano /var/users
[sudo] password for user:
[user@centos8 ~]$ cut -d' ' -f1 /var/users
user30
user31
user32
user33
user34
user35
[user@centos8 ~]$ █
```

В прошлый раз мы остановились на том, что создали файл и в скрипте добавили возможность брать данные о пользователе и группе из этого файла. Теперь же попробуем в файле указать несколько пользователей:

```
sudo nano /var/users
```

и добавить их разом. Выполнив ту же команду cut:

```
cut -d' ' -f1 /var/users
```

мы увидим весь список пользователей. Если передать команде useradd такой список пользователей, она этого не поймёт – useradd может добавлять только по одному пользователю за раз. А значит нам нужно будет для каждой строчки запускать отдельный useradd.

```
[user@centos8 ~]$ tail -n +30 myscript
groupadd $group

# Sudoers
if [ "$group" = it ] || [ "$group" = security ]
then
    if ! grep "%$group" /etc/sudoers
    then
        cp /etc/sudoers{,.bkp}
        echo '%'$group' ALL=(ALL) ALL' >> /etc/sudoers
    fi
    shell=/bin/bash
elif [ "$user" = admin ]
then
    if ! grep "$user" /etc/sudoers
    then
        cp /etc/sudoers{,.bkp}
        echo $user' ALL=(ALL) ALL' >> /etc/sudoers
    fi
    shell=/bin/bash
fi
mkdir -v /home/$group
useradd $user -g $group -b /home/$group -s $shell
[user@centos8 ~]$ █
```

И так, задача у нас такая – для каждой строчки в файле /var/users создавать группу, проверять sudoers и создавать пользователя. То есть всё что ниже 30 строчки:

```
tail -n +30 myscript
```

Если речь про повторное запускание одной и той же команды – то речь обычно о циклах. Есть две стандартные команды для работы с циклами – for и while. for обычно связан со списком, а while – с условием, хотя нередко можно использовать и ту, и другую. Давайте начнём с for. Синтаксис такой:

```
for переменная in список значений
do команды
done
```

При запуске команды переменная получит первое значение из списка значений, потом выполняются все команды, а после команды переменная получит второе значение из списка значений и опять выполняются все команды. И так будет повторяться до тех пор, пока не закончатся все значения в списке, после чего цикл завершится. Каждое повторение называется итерацией.

```
[user@centos8 ~]$ nano for
[user@centos8 ~]$ cat for
for number in 1 two "line #3"
do echo This is $number
done
[user@centos8 ~]$ chmod +x for
[user@centos8 ~]$ ./for
This is 1
This is two
This is line #3
[user@centos8 ~]$ █
```

Давайте посмотрим пример:

```
nano for
```

```
for number in 1 two "line № 3"
do echo This is $number
done
```

```
chmod +x for
./for
```

Как видите, сначала переменная `number` получила первое значение – `1`, выполнилась команда `echo`. Потом переменная `number` взяла второе значение – `two`. Ну и так далее. Вроде ничего сложного.

```
[user@centos8 ~]$ cat for
for line in $(cat /var/users)
do echo In this line: $line
done
[user@centos8 ~]$ ./for
In this line: user30
In this line: it
In this line: user31
In this line: users
In this line: user32
In this line: it
In this line: user33
In this line: group33
In this line: user34
In this line: it
In this line: user35
In this line: users
[user@centos8 ~]$ █
```

Список значений можно задать по разному, например, взять его из вывода команды:

```
nano for
```

```
for line in $(cat /var/users)
do echo In this line: $line
done
```

```
sudo ./for
```

Но вместо того, чтобы увидеть в виде значения каждую строчку, мы видим пользователя и группу на разных строчках. То есть цикл сначала присвоил переменной `line` в виде значения имени первого юзера, а после итерации значение переменной стало имем группы. И так с каждой строчкой. То есть, вместо того, чтобы разделять значения построчно, значения разделялись по пробелам.

Помните, мы в команде `cut` использовали опцию `-d` – разделитель:

```
sudo cut -d ' ' -f1 /var/users
```

И мы этой опцией указали, что разделителем является пробел. `bash`, чтобы взять список значений, тоже использует разделитель – сначала он попытается разделить значения по пробелу, потом по табуляции и только потом по переводу строки. А чтобы `bash` в качестве разделителя использовал сразу перевод строки, нам нужно об этом ему сказать. Для этого есть переменная `IFS` – внутренний разделитель

полей.

```
IFS=$'\n'  
IFS=$'\t'  
IFS=:  
oldIFS=$IFS  
IFS=$oldIFS
```

Чтобы указать, что мы хотим в качестве разделителя использовать перевод строки, даём переменной такое значение:

```
IFS=$'\n'
```

\n – это newline. Если захотим знак табуляции меняем n на t:

```
IFS=$'\t'
```

Если брать, например, /etc/passwd, то там разделителем выступает двоеточие, тогда можно указать так:

```
IFS=:
```

Но с этой переменной нужно быть осторожным – другие команды в скрипте также могут использовать эту переменную, а значит то что у вас работало с пробелами, может начать работать с новыми строками. И чтобы не пришлось переделывать пол скрипта, мы можем поменять эту переменную временно, а потом вернуть старое значение. Но для этого нужно старое значение предварительно сохранить:

```
oldIFS=$IFS
```

А после выполнения нужных команд можем восстановить старое значение:

```
IFS=$oldIFS
```

```
[user@centos8 ~]$ nano for
[user@centos8 ~]$ cat for
IFS=$'\n'
for line in $(cat /var/users)
do echo This is line: $line
done
[user@centos8 ~]$ ./for
This is line: user30 it
This is line: user31 users
This is line: user32 it
This is line: user33 group33
This is line: user34 it
This is line: user35 users
[user@centos8 ~]$ █
```

Но нам сейчас нужен newline:

```
IFS=$'\n'
```

Попробуем запустить скрипт:

```
sudo ./for
```

Теперь всё окей – при каждой итерации переменная будет получать в качестве значения целую строчку.

```
[user@centos8 ~]$ nano for
[user@centos8 ~]$ cat for
oldIFS=$IFS
IFS=$'\n'
for line in $(cat /var/users)
do
    user=$(echo $line | cut -d' ' -f1)
    group=$(echo $line | cut -d' ' -f2)
    echo Username: $user Group: $group
done
IFS=$oldIFS
[user@centos8 ~]$ ./for
Username: user30 Group: it
Username: user31 Group: users
Username: user32 Group: it
Username: user33 Group: group33
Username: user34 Group: it
Username: user35 Group: users
[user@centos8 ~]$ █
```

Дальше нужно просто из этой переменной достать имя пользователя и группы, допустим, с помощью того же cut. А чтобы передать команде cut значение переменной, можно использовать пайп:

```
echo $line | cut -d' ' -f1
```

В итоге мы достанем из строчки имя пользователя. И чтобы это имя стало переменной, напишем так:

```
user=$(echo $line | cut -d' ' -f1)
```

Тоже самое с группой:

```
group=$(echo $line | cut -d' ' -f2)
```

Последний штрих:

```
echo Username: $user Group: $group
```

```
sudo ./for
```

Как видите, всё сработало как надо. Теперь попытаемся сделать тоже самое с нашим скриптом.

```
# Check arguments
if [ ! -z $2 ]
then
    user=$1
    group=$2
    echo Username: $user      Group: $group
elif [ -f $file ]
then
IFS=$'\n'
for line in $(cat $file)
do
    user=$(echo $line | cut -d' ' -f1)
    group=$(echo $line | cut -d' ' -f2)
    echo Username: $user      Group: $group
done
IFS=$oldIFS
```

[Write 57 lines]

Предварительно сохраним значение переменной IFS:

```
oldIFS=$IFS
```

Тут у нас уже есть строчки назначения переменных из файла, но это нам не подходит, потому что нам нужно брать переменные в цикле, поэтому эти строчки убираем. Попробуем вписать сюда наш цикл:

```
IFS=$'\n'
for line in $(cat $file)
do
    user=$(echo $line | cut -d' ' -f1)
    group=$(echo $line | cut -d' ' -f2)
    echo Username: $user Group: $group
done
IFS=$oldIFS
```

```
[user@centos8 ~]$ sudo ./myscript
[sudo] password for user:
Username: user30 Group: it
Username: user31 Group: users
Username: user32 Group: it
Username: user33 Group: group33
Username: user34 Group: it
Username: user35 Group: users
groupadd: group 'users' already exists
mkdir: cannot create directory '/home/users': File exists
[user@centos8 ~]$ tail -5 /etc/passwd
user16:x:1124:1009::/home/group16/user16:/sbin/nologin
user17:x:1125:1010::/home/group17/user17:/sbin/nologin
user18:x:1126:1011::/home/group18/user18:/sbin/nologin
user22:x:1127:1004::/home/it/user22:/bin/bash
user35:x:1128:100::/home/users/user35:/sbin/nologin
[user@centos8 ~]$ █
```

Запустим и проверим:

```
sudo ./myscript
tail -5 /etc/passwd
```

У нас там было несколько пользователей, а создался только последний. Подумайте, почему так случилось?

Обратите внимание на наш цикл – переменные получают свои значения, выполняется команда echo, а после итерации всё происходит заново, пока не дойдёт до последнего значения. И только после этого начинают выполняться все остальные команды – группы, sudoers и т.д. Нам же нужно, чтобы с каждой итерацией выполнялись все эти команды.

Что мне мешает поставить done в конце скрипта? Мы сейчас находимся в условии – я не могу просто посреди for закончить условие if и продолжить выполнять команды for. Команда началась внутри условия – там же она должна закончиться. Есть вариант скопировать все оставшиеся команды сюда. Но это плохой вариант – это увеличит размер скрипта, в дальнейшем придётся редактировать команды и внутри цикла, и отдельно.

```

create_user() {
IFS=$oldIFS
groupadd $group

# Sudoers
if [ "$group" = it ] || [ "$group" = security ]
then
    if ! grep "%$group" /etc/sudoers
    then
        cp /etc/sudoers{,.bkp}
        echo '%'$group' ALL=(ALL) ALL' >> /etc/sudoers
    fi
    shell=/bin/bash
elif [ "$user" = admin ]

```

Вот у нас есть куча команд и я не хочу, чтобы они повторялись в скрипте в нескольких местах. Чтобы решить эту проблему, я могу объединить все эти команды под одним названием. Для этого я пишу название, допустим: `create_user()` - ставлю после названия скобки, а потом внутри фигурных скобок указываю все нужные команды:

```

create_user() {
    groupadd ...
}

```

Это называется функцией. И в дальнейшем, когда я захочу запустить все эти команды, я просто запущу команду:

```
create_user
```

Но функция должна быть задана до того, как к ней обращаются, поэтому переместим нашу функцию наверх, скажем, после переменных. Но тут ещё один нюанс – IFS возвращает старое значение:

```
IFS=$oldIFS
```

после выполнения цикла, а значит после выполнения всех команд в функции. А так как все наши команды там, то лучше перенести эту команду:

```
IFS=$oldIFS
```

в начало функции.

```

then
    user=$1
    group=$2
    echo Username: $user    Group: $group
    create_user
elif [ -f $file ]
then
IFS=$'\n'
for line in $(cat $file)
do
    user=$(echo $line | cut -d' ' -f1)
    group=$(echo $line | cut -d' ' -f2)
    echo Username: $user    Group: $group
    create_user
done
IFS=$oldIFS
else
    echo Welcome!

```

А теперь пропишем её в наших условиях – просто написав create_user в командах каждого из условий.

Хорошо, давайте пройдёмся по скрипту. Вначале мы проверяем на root права. Задаём переменные. Создаём функцию – create_user – тут у нас все нужные команды для создания группы и пользователя. А в конце у нас проверка, как мы запускали программу – с параметрами, с файлом или интерактивно – в зависимости от этого назначаются переменные и запускается функция.

```

mkdir: cannot create directory '/home/it': File exists
Username: user35 Group: users
groupadd: group 'users' already exists
mkdir: cannot create directory '/home/users': File exists
useradd: user 'user35' already exists
[user@centos8 ~]$ tail -5 /etc/passwd
user30:x:1129:1004::/home/it/user30:/bin/bash
user31:x:1130:100::/home/users/user31:/bin/bash
user32:x:1131:1004::/home/it/user32:/bin/bash
user33:x:1132:1012::/home/gr/user33:/bin/bash
user34:x:1133:1004::/home/it/user34:/bin/bash
[user@centos8 ~]$ █

```

Окей, давайте протестируем:

```

sudo ./myscript
tail -5 /etc/passwd

```

Как видите, все пользователи создались, всё работает.

Теперь немного проработаем наше интерактивное меню, то есть опцию else. Сейчас, при запуске скрипта, в этом режиме оно запросит юзернейм, пароль, создаст пользователя и закроется. Я бы хотел, чтобы после создания пользователя наш скрипт не завершался, а предлагал заново создать пользователя и всякие другие менюшки. Для этого мне понадобятся две команды. Первая будет показывать меню – это команда select. Синтаксис чем-то похож на for:

```
select переменная in список значений
do команды
done
```

```
[user@centos8 ~]$ nano select
[user@centos8 ~]$ cat select
select number in 1 2 3
do echo This is number: $number
done
[user@centos8 ~]$ chmod +x select
[user@centos8 ~]$ ./select
1) 1
2) 2
3) 3
#? 1
This is number: 1
#? 3
This is number: 3
#? 2
This is number: 2
#? ^C[user@centos8 ~]$
```

Например:

```
nano select
```

```
select number in 1 2 3
do echo This is number: $number
done
```

```
chmod+x select
./select
```

select показал нам меню, где с помощью цифр мы можем выбрать какое-то из значений и переменная получит это значение. Дальше выполнится команда и после неё опять появится меню.

Теперь мне нужна команда, которая будет запускать какие-то команды в зависимости от значения

переменной. Речь про команду case. Синтаксис такой:

```
case $переменная in
    значение 1) команда;;
    значение 2) команда ;;
    *) команда, если значения нет в списке ;;
esac
```

```
[user@centos8 ~]$ nano case
[user@centos8 ~]$ cat case
number=one
case $number in
    one) echo 1;;
    two) echo 2;;
    *) echo something wrong;;
esac
[user@centos8 ~]$ chmod +x case
[user@centos8 ~]$ ./case
1
[user@centos8 ~]$ █
```

Например:

```
nano case
```

```
number=one
case $number in
    one) echo 1;;
    two) echo 2;;
    *) echo something wrong ;;
esac
```

```
chmod +x case
./case
```

Как видите, значение переменной было one. Оно подошло под первую опцию, в следствии чего сработала первая команда.

```
[user@centos8 ~]$ cat case
select number in 1 2 3
do
    case $number in
        1) echo One;;
        2) echo Two;;
        3) echo Three ;;
       *) echo something wrong;;
    esac
done
[user@centos8 ~]$ ./case
1)
2)
3)
#? 1
One
#? 3
Three
#? 4
something wrong
#? █
```

Теперь объединим select и case. Например:

```
select number in 1 2 3
do
    case $number in
        1) echo One;;
        2) echo Two;;
        3) echo Three;;
       *) echo something wrong ;;
    esac
done
```

```
./case
```

Теперь, select предлагает нам выбрать одно из значений, это значение назначается переменной number, затем case, в зависимости от значения переменной, запускает соответствующую команду.

```
[user@centos8 ~]$ nano case
[user@centos8 ~]$ ./case
1) 1
2) 2
3) 3
4) stop
#? 1
One
#? 4
[user@centos8 ~]$ █
```

А чтобы не застрять в бесконечной петле, в списке опций пропишем что-нибудь типа stop, а в case используем команду:

`break`

чтобы прекратить цикл. После break цикл прерывается и начинают выполняться другие команды после цикла:

`./case`

```
else
    echo Welcome!
    select option in "Add user" "Show users" "Exit"
    do case $option in
        "Add user") read -p "Print username: " user
                      read -p "Print groupname: " group
                      create_user;;
        "Show users") cut -d: -f1 /etc/passwd ;;
        "Exit") break ;;
        *) echo Wrong option ;;
    esac
done
fi
```

Теперь добавим это в нашем скрипте. Допустим, сделаем так, чтобы можно было добавить пользователя, посмотреть текущих пользователей, либо выйти:

```
select option in "Add user" "Show users" "Exit"
do
    case $option in
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
"Add user") read -p ... ;;
>Show users") cut -d: -f1 /etc/passwd ;;
"Exit") break ;;
*) echo Wrong option ;;
esac
done
```

```
[user@centos8 ~]$ sudo rm /var/users
[user@centos8 ~]$ sudo ./myscript
Welcome!
1) Add user
2) Show users
3) Exit
#? 1
Print username: user36
Print groupname: it
groupadd: group 'it' already exists
%it ALL=(ALL) ALL
mkdir: cannot create directory '/home/it': File exists
#? 2
root
bin
daemon
adm
lp
```

Сохраним, удалим файл:

```
sudo rm /var/users
```

чтобы наш скрипт предложил интерактивное меню и попробуем запустить скрипт:

```
sudo ./myscript
```

Выбираем опцию 1 – у нас выходит приглашение ввести имя пользователя и группу. Окей, нажимаем enter – меню появилось ещё раз. Теперь выбираем 2 – и видим список всех пользователей. Выбираем 3 и выводим.

Подводя итоги, сегодня мы с вами разобрали команду for, с помощью которой мы можем создавать циклы; переменную IFS; функции, с помощью которых можем вызывать одну или несколько заранее прописанных команд; команду select, с помощью которой мы можем создать интерактивное меню; и команду case, с помощью которой мы можем запускать команды в зависимости от значения переменной.

2.30.2 Практика

Вопросы

1. Что можно задать в качестве списка значений для for?
2. Для чего нужен IFS?

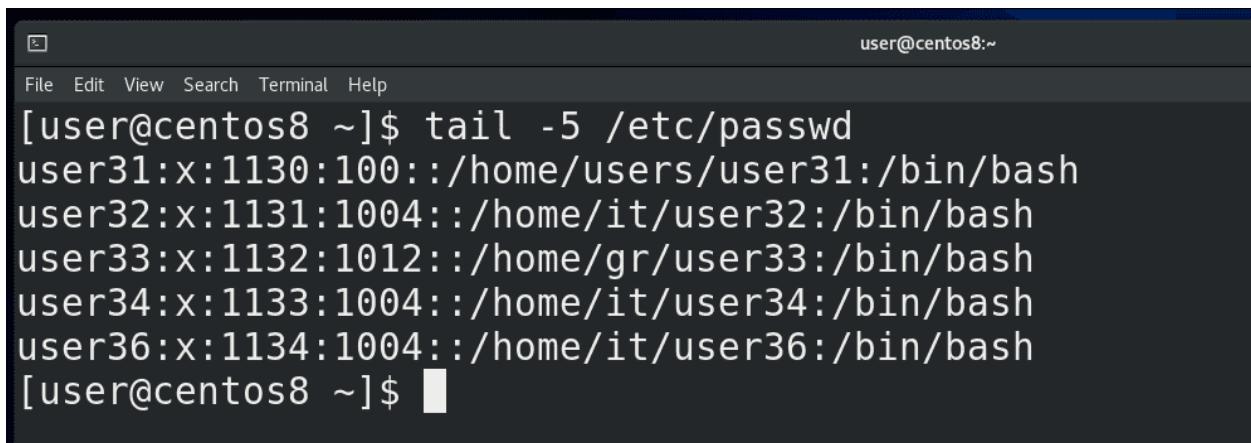
3. Для чего нужны функции?

Задания

1. Создайте файл со списком пользователей. С помощью for выведите на экран текст: «Creating new user: useradd имя_пользователя».
2. Создайте функции просмотра информации о пользователе(1), создания пользователя(2), удаления пользователя (3). Используйте case и select для использования этих функций в виде меню.
3. Разрешите пользователю helpdesk запускать команду newbackup от имени пользователя backup. Команда newbackup должна создавать сжатый архив директории /data и сохранять его в домашней директории пользователя backup с датой в названии.

2.31 31. bash скрипты №5

2.31.1 31. bash скрипты №5



The screenshot shows a terminal window with a dark background. At the top, there's a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The title bar says 'user@centos8:~'. The main area of the terminal displays the output of the command 'tail -5 /etc/passwd'. The output lists five user entries from the passwd file:

```
[user@centos8 ~]$ tail -5 /etc/passwd
user31:x:1130:100::/home/users/user31:/bin/bash
user32:x:1131:1004::/home/it/user32:/bin/bash
user33:x:1132:1012::/home/gr/user33:/bin/bash
user34:x:1133:1004::/home/it/user34:/bin/bash
user36:x:1134:1004::/home/it/user36:/bin/bash
[user@centos8 ~]$
```

В прошлый раз мы с вами разбирали циклы и функции. Если посмотреть список созданных циклом пользователей:

```
tail /etc/passwd
```

можно заметить ошибку – у всех пользователей оболочкой является bash, хотя так должно быть только у определённых пользователей.

```

create_user() {
IFS=$oldIFS
groupadd $group

# Sudoers
if [ "$group" = it ] || [ "$group" = security ]
then
    if ! grep "%$group" /etc/sudoers
    then
        cp /etc/sudoers{,.bkp}
        echo '%'$group' ALL=(ALL) ALL' >> /etc/sudoers
    fi
    shell=/bin/bash
elif [ "$user" = admin ]
then

```

Оболочка в нашем скрипте задаётся переменной shell и значение по умолчанию – nologin. При выполнении цикла запускается функция create_user, где есть условие, что если группой является it или security, указать значение переменной shell:

```
shell=/bin/bash
```

Функция у нас выполнится, создаётся один пользователь, а при итерации значение переменной так и останется bash. Собственно поэтому у всех пользователей оболочкой остался bash.

```

create_user() {
IFS=$oldIFS
shell=/sbin/nologin
groupadd $group

```

Чтобы решить эту проблему, мы можем занести переменную shell в начало функции, чтобы при каждой итерации переменная получала значение nologin.

GNU nano 2.9.8	/var/users
user30 it	
user31 group1	
...	

Касательно списка пользователей. В прошлый раз мы создали максимально простой файл, там были логины пользователей и группы, что легко было использовать в нашем скрипте. Но не всегда данные так хорошо подстроены под нас – вряд ли какой HR будет присыпать нам файл в таком виде. Да и в

большинстве реальных задач у вас есть файлы с большим количеством информации и вам нужно из этих файлов достать только нужное. Это чуть сложнее автоматизировать, но довольно увлекательно. В целом это называется парсингом – когда мы берём данные, структурируем их, чтобы в дальнейшем эти данные можно было использовать для других задач.

A	B	C	D	E
	First Name	Last Name	Birthday	Department
1	Bruce	Wayne	19.02	IT
2	Oliver	Queen	16.05	Security
3	Clark	Kent	18.06	Users
4	Barry	Allen	14.03	IT

Для примера я подготовил excel файл со списком пользователей.

Такой файл не прочтёшь в командной строке - cat users.xlsx, так как это не просто текстовой файл. Но зачастую есть инструменты, которые позволяют превратить нечитаемые файлы в более подходящие форматы. Форматов много, ситуации разные, всё я вам показать не смогу, но с помощью гугла вы сможете найти многие решения. Давайте рассмотрим наш пример. И так, у нас есть excel таблица. Когда речь о каких-то таблицах, то обычно стоит попробовать превратить файл в csv формат.

```
[user@centos8 ~]$ libreoffice --headless --convert-to csv users.xlsx
convert /home/user/users.xlsx -> /home/user/users.csv using filter : Text - t
xt - csv (StarCalc)
[user@centos8 ~]$ cat users.csv
,First Name,Last Name,Birthday,Department
1,Bruce,Wayne,19.02,IT
2,Oliver,Queen,16.05,Security
3,Clark,Kent,18.06,Users
4,Barry,Allen,14.03,IT
[user@centos8 ~]$ █
```

К примеру, это умеет делать программа libreoffice – свободный офисный пакет программ. Хотя можно найти и установить более лёгкие альтернативы. Команда такая:

```
libreoffice --headless --convert-to csv users.xlsx
```

Теперь посмотрим полученный результат:

Каквидите, получился обычный текстовой файл, при этом это таблица, в которой делителем выступает запятая. Получить данные отсюда будет гораздо проще.

И так, основы работы с таблицами. У таблицы есть две составляющие – столбцы и строки. Зачастую, в столбце у нас списки, а в столбике данные по определённому объекту. Например, второй столбец – это список имён, а вторая строка – вся информация про одного пользователя.

```
[user@centos8 ~]$ cut -d, -f5 users.csv
Department
IT
Security
Users
IT
[user@centos8 ~]$ cut -d, -f2,3,5 users.csv
First Name,Last Name,Department
Bruce,Wayne,IT
Oliver,Queen,Security
Clark,Kent,Users
Barry,Allen,IT
[user@centos8 ~]$ █
```

Мы уже знаем команду cut чтобы брать данные по столбцам. Тут у нас делителем выступает запятая. Допустим, выведем список всех департаментов:

```
cut -d, -f5 users.csv
```

Добавим ещё информации о пользователях:

```
cut -d, -f2,3,5 users.csv
```

```
[user@centos8 ~]$ cut -d, -f2,3,5 users.csv | tail -n +2
Bruce,Wayne,IT
Oliver,Queen,Security
Clark,Kent,Users
Barry,Allen,IT
[user@centos8 ~]$ █
```

Но первая строчка нам не очень нужна и будет мешать, поэтому нужно как-то начать со второй. Вспоминаем команду tail — с ключом -n она позволяет выводить текст с определённой строчки:

```
cut -d, -f2,3,5 users.csv | tail -n +2
```

Вот, собственно, это наши будущие пользователи и группы. Но для начала эту информацию надо превратить в более удобный вид.

```
[user@centos8 ~]$ grep -w it /etc/group
it:x:1004:
[user@centos8 ~]$ █
```

Например, посмотрим на департаменты – все они содержат заглавные буквы. Мы знаем, что Linux – система регистрозависимая. У нас уже есть группа it:

```
grep -w it /etc/group
```

но команда groupadd IT создаст абсолютно другую группу, чего нам не нужно. Для удобства лучше держать все логины и группы в строчных буквах. То есть, мне нужно преобразовать все заглавные буквы в строчные.

```
[user@centos8 ~]$ echo Hello World
Hello World
[user@centos8 ~]$ echo Hello World | tr o O
Hello WOrld
[user@centos8 ~]$ echo Hello World | tr o O | tr ' ' '\n'
Hello
WOrld
[user@centos8 ~]$ echo Hello World | tr o O | tr ' ' '\n' | tr a-z A-Z
HELLO
WORLD
[user@centos8 ~]$ echo Hello World | tr o O | tr ' ' '\n' | tr A-Z a-z
hello
wOrld
```

Для этого мы можем использовать команду tr – translate. С помощью этой команды мы можем одни символы преобразовывать в другие, удалять какие-то символы и много всего интересного. Например, заменим о на 0:

```
echo Hello World
echo Hello World | tr o 0
```

Или заменим пробел на символ переноса строки:

```
echo Hello World | tr o 0 | tr ' ' '\n'
```

Заменим все строчные буквы на заглавные:

```
echo Hello World | tr o 0 | tr ' ' '\n' | tr a-z A-Z
```

Или наоборот:

```
echo Hello World | tr o 0 | tr ' ' '\n' | tr A-Z a-z
```

```
[user@centos8 ~]$ echo Hello World | tr '[:upper:]' '[:lower:]'
hello world
[user@centos8 ~]$ cut -d, -f2,3,5 users.csv | tail -n +2 | tr '[:upper:]' '[:lower:]'
bruce,wayne,it
oliver,queen,security
clark,kent,users
barry,allen,it
[user@centos8 ~]$ cut -d, -f2,3,5 users.csv | tail -n +2 | tr '[:upper:]' '[:lower:]' | tr , ' '
bruce wayne it
oliver queen security
clark kent users
barry allen it
[user@centos8 ~]$ █
```

Хотя есть более правильный способ это сделать:

```
echo Hello World | tr '[:upper:]' '[:lower:]'
```

Более правильный, потому что во всех локалях буквы А и Z первая и последняя соответственно. Ну да ладно, вернёмся к нашей задаче. Превратим все заглавные буквы в строчные:

```
cut -d, -f2,3,5 users.csv | tail -n +2 | tr '[:upper:]' '[:lower:]'
```

Ну и для удобства можем все запятые заменить на пробелы:

```
cut -d, -f2,3,5 users.csv | tail -n +2 | tr '[:upper:]' '[:lower:]' | tr , ' '
```

```
[user@centos8 ~]$ users=$(cut -d, -f2,3,5 users.csv | tail -n +2 | tr '[:upper:]'
'[:lower:]' | tr , ' ')
[user@centos8 ~]$ echo $users
bruce wayne it oliver queen security clark kent users barry allen it
[user@centos8 ~]$ echo "$users"
bruce wayne it
oliver queen security
clark kent users
barry allen it
[user@centos8 ~]$ █
```

Сохраним это в переменной:

```
users=$(cut -d, -f2,3,5 users.csv | tail -n +2 | tr '[:upper:]' '[:lower:]' | tr , ' ')
```

Обратите внимание – если просто писать echo \$users – то bash подставляет значение переменной в строку, при этом используя переменную IFS, где разделителем является сначала пробел, потом табуляция и только потом перевод строки. Поэтому у нас эти четыре строки начали разделяться пробелом, а не переводом строки. Это легко решить, если взять переменную в кавычки:

```
echo "$users"
```

```
[user@centos8 ~]$ echo "$users" | head -1
bruce wayne it
[user@centos8 ~]$ echo "$users" | head -1 | cut -c1
b
[user@centos8 ~]$ echo "$users" | head -1 | cut -d' ' -f2
wayne
[user@centos8 ~]$ echo $(echo "$users" | head -1 | cut -c1).$(echo "$users" | head -1
| cut -d' ' -f2)
b.wayne
[user@centos8 ~]$ █
```

И так, у нас есть переменная, в которой содержится имя, фамилия и группа пользователя. Зачастую, в компаниях логин пользователя представляет из себя комбинацию имени и фамилии. Сделаем так, чтобы логин начинался с первой буквы имени, потом точка, а затем фамилия. Например, b.wayne. У нас в скрипте уже есть цикл, который построчно считывает список, поэтому для теста используем одну из строк:

```
echo "$users" | head -1
```

И так, для получения первой буквы имени, всё также используем cut, но с опцией -c1 – первый символ:

```
echo "$users" | head -1 | cut -c1
```

Для получения фамилии cut с делителем в виде пробела:

```
echo "$users" | head -1 | cut -d' ' -f2
```

Теперь объединим выводы этих команд с помощью echo:

```
echo $(echo "$users" | head -1 | cut -c1).$(echo "$users" | head -1 | cut -d' ' -f2)
```

Можно было бы предварительно превратить выводы команд в переменные, а уже потом объединять. Но, в целом, команда не такая сложная, да и head тут лишний, в конечном счёте мы будем работать с каждой строкой отдельно.

```
for line in $(cut -d, -f2,3,5 $file | tail -n +2 | tr '[:upper:]' '[:lower:]' | tr , ' ')
do
    user=$(echo $line | cut -c1).$(echo "$line" | cut -d' ' -f2)
    group=$(echo "$line" | cut -d' ' -f3)
    echo Username: $user Group: $group
    create_user
```

И так, логин мы получили, а группа и так в доступном виде. Теперь давайте добавим это в наш скрипт. Скопируем команду для получения списка пользователей:

```
cut -d, -f2,3,5 users.csv | tail -n +2 | tr '[:upper:]' '[:lower:]' | tr , ' '
```

и заменим в скрипте на эту команду. Заменим users.csv переменной \$file. Скопируем команду для получения логина пользователя и вставим как значение переменной user. Заменим переменную users на line и уберём head:

```
user=$(echo $(echo "$line" | cut -c1).$(echo "$line" | cut -d' ' -f2))
```

Заменим значение переменной group:

```
group=$(echo "$line" | cut -d' ' -f3)
```

И так, вроде переменные user и group должны получить свои значения, остальные команды трогать не нужно.

```
[user@centos8 ~]$ nano myscript
[user@centos8 ~]$ grep file= myscript
file=/var/users.csv
[user@centos8 ~]$ sudo cp -v users.csv /var/
[sudo] password for user:
'users.csv' -> '/var/users.csv'
[user@centos8 ~]$ █
```

Разве что заменим значение переменной file - file=/var/users.csv. Вы, по желанию, можете добавить, чтобы excel файл конвертировался при запуске скрипта. А пока скопируем сам файл:

```
sudo cp users.csv /var/
```

```
[user@centos8 ~]$ sudo ./myscript
Username: b.wayne Group: it
groupadd: group 'it' already exists
%it ALL=(ALL) ALL
mkdir: cannot create directory '/home/it': File exists
Username: o.queen Group: security
groupadd: group 'security' already exists
%security ALL=(ALL) ALL
mkdir: cannot create directory '/home/security': File exists
Username: c.kent Group: users
```

```
[user@centos8 ~]$ tail -5 /etc/passwd
user36:x:1134:1004::/home/it/user36:/bin/bash
b.wayne:x:1135:1004::/home/it/b.wayne:/bin/bash
o.queen:x:1136:1005::/home/security/o.queen:/bin/bash
c.kent:x:1137:100::/home/users/c.kent:/sbin/nologin
b.allen:x:1138:1004::/home/it/b.allen:/bin/bash
[user@centos8 ~]$ █
```

Давайте протестируем:

```
sudo ./myscript
tail /etc/passwd
```

Как видите, все пользователи создались, группы соответствующие, оболочки тоже.

Но задумайтесь вот о чём – если мы генерируем логины на основе имени и фамилии, в компании могут появиться тёзки, однофамильцы, какая-нибудь Clara Kent. Наш скрипт это проигнорирует, а команда useradd откажется создавать пользователя, так как такой логин уже есть. Поэтому такое вам задание – сделайте так, чтобы скрипт определял, есть ли уже такой логин, и, в случае чего, логин второго пользователя был немного другим, допустим, полное имя и фамилия, либо какое-то число в конце.

```
[user@centos8 ~]$ cut -d: -f1 /etc/passwd | grep b.wayne
b.wayne
[user@centos8 ~]$ cut -d: -f1 /etc/passwd | grep user1
user12
user14
user15
user16
user17
user18
[user@centos8 ~]$ █
```

Скорее всего, вы будете определять по списку текущих пользователей. Наверняка вы обратитесь к файлу passwd, будите использовать cut и grep:

```
cut -d: -f1 /etc/passwd | grep b.wayne
```

Но такого grep-а может не хватить. Обратите внимание, я ищу пользователя user1:

```
cut -d: -f1 /etc/passwd | grep user1
```

Такого пользователя нет, но grep всё равно выдал мне результат, а скрипту главное результат. grep ищет соответствие, а все результаты содержат в себе user1. Мне нужно как-то сказать grep-у, что строчка в выводе должна заканчиваться на user1.

```
[user@centos8 ~]$ cut -d: -f1 /etc/passwd | grep user1$
```

```
[user@centos8 ~]$ cut -d: -f1 /etc/passwd | grep user$
```

```
rpcuser
```

```
user
```

```
[user@centos8 ~]$ cut -d: -f1 /etc/passwd | grep ^user
```

```
user
```

```
user2
```

```
user3
```

```
user4
```

```
user5
```

Для этого я могу использовать символ \$ в конце искомой строки, то есть grep user1\$:

```
cut -d: -f1 /etc/passwd | grep user1$
```

Теперь он ничего не нашёл. Но попробуем найти просто user\$:

```
cut -d: -f1 /etc/passwd | grep user$
```

Как видите, опять несоответствие – есть некий grcuser, в нём содержится слово user и оно заканчивается на него. Теперь мне нужно указать, чтобы строка начиналась на user – для этого можно использовать символ ^ – он называется карет. Указывая его перед выражением:

```
cut -d: -f1 /etc/passwd | grep ^user
```

мы говорим grep-у, что строка должна начинаться на эти символы.

```
[user@centos8 ~]$ cut -d: -f1 /etc/passwd | grep ^user$  
user  
[user@centos8 ~]$ cut -d: -f1 /etc/passwd | grep -w user  
user  
[user@centos8 ~]$ █
```

И так, указывая каретку перед выражением, а доллар в конце:

```
cut -d: -f1 /etc/passwd | grep ^user$
```

мы говорим grep-у, что строка должна начинаться и заканчиваться на слове user. Таким образом, с помощью каретки и знака доллара мы использовали так называемые регулярные выражения. Это специальный язык, с помощью которого мы можем более гибко работать с текстом. Регулярных выражений много, их можно применять по разному. Это большая тема, которую мы разберём в другой раз. Однако, в случае с grep, если мы ищем точное совпадение, можно использовать ключ -w:

```
cut -d: -f1 /etc/passwd | grep -w user
```

Подводя итоги, сегодня мы с вами взяли excel файл, вытянули оттуда данные, преобразовали эти данные в нужный нам вид и использовали в нашем скрипте.

2.32 bash скрипты №6

2.32.1 bash скрипты №6

В прошлый раз мы научили наш скрипт вытягивать информацию о пользователях из файла. Всё это для того, чтобы наш скрипт был более самостоятельным. Но создавая пользователей вручную, мы можем предварительно проверить, занят ли такой логин, и, в случае чего, создать пользователя с другим логином. Попробуем научить скрипт делать также.

A	B	C	D	E
	First Name	Last Name	Birthday	Department
1	Bruce	Wayne	19.02	IT
2	Bruce	Wayne	16.05	Security
3	Bruce	Wayne	18.06	Users
4	Bruce	Wayne	14.03	IT

Я немного изменил файл с пользователями. Предположим, так получилось, что у нас 4 тёзки и мы отличаем их по дням рождения и группам.

```
[user@centos8 ~]$ libreoffice --headless --convert-to csv users.xlsx
convert /home/user/users.xlsx -> /home/user/users.csv using filter : Text - txt - csv (StarCalc)
Overwriting: /home/user/users.csv
[user@centos8 ~]$ cat users.csv
,First Name,Last Name,Birthday,Department
1,Bruce,Wayne,19.02,IT
2,Bruce,Wayne,16.05,Security
3,Bruce,Wayne,18.06,Users
4,Bruce,Wayne,14.03,IT
[user@centos8 ~]$
```

Конвертируем файл в csv формат:

```
libreoffice --headless --convert-to csv users.xlsx
cat users.csv
```

```
[user@centos8 ~]$ cut -d, -f2,3,4,5 users.csv | tail -n +2 | tr '[:upper:]' '[:lower:]' |
tr , '
bruce wayne 19.02 it
bruce wayne 16.05 security
bruce wayne 18.06 users
bruce wayne 14.03 it
[user@centos8 ~]$
```

Немного подправим cut, с помощью которого мы брали информацию из csv файла – добавим в него поле 4, чтобы также вытягивать информацию о днях рождения. Проверим в терминале:

```
cut -d, -f2,3,4,5 users.csv | tail -n +2 | tr '[:upper:]' '[:lower:]' | tr , '
```

```
then
IFS=$'\n'
for line in $(cut -d, -f2,3,4,5 $file | tail -n +2 | tr '[:upper:]' '[:lower:]' | tr , ' ')
do
    user=$(echo "$line" | cut -d' ' -f2)
    group=$(echo "$line" | cut -d' ' -f4)
    bday=$(echo "$line" | cut -d' ' -f3)
    echo Username: $user Group: $group
    create_user
done
IFS=$oldIFS
```

Как видите, информация о группе сместилась на 4 столбик, поэтому подправляем это в скрипте:

```
group=$(echo "$line" | cut -d' ' -f4)
```

Добавляем ещё одну переменную – bday – в которой и будет информация о дне рождения:

```
bday=$(echo "$line" | cut -d' ' -f3)
```

```
fi
shell=/bin/bash
fi
mkdir -v /home/$group
useradd $user -g $group -b /home/$group -s $shell -c "Birthday $bday"
}
```

Ну и чтобы использовать эту переменную, добавим в нашу функцию create_user в саму команду useradd опцию -c – то есть комментарий:

```
-c "Birthday $bday"
```

Пока мы только подготовились к тому, чтобы различать тёзок после их создания. Если мы сейчас просто запустим скрипт, useradd не создаст пользователей с одинаковыми логинами. И так, наша задача - скрипт перед созданием пользователя должен проверить, есть ли уже такой логин, и, если есть, создать пользователя с другим логином. Но, конечно, нужно ещё проверить, а не занят ли тот второй логин. Если мы будем проверять просто с помощью if, мы увидим, что есть пользователь b.wayne и создадим пользователя b.wayne2. А если он уже занят? Чтобы проверить, а не занят ли логин b.wayne2, нам придётся написать elif. Тогда мы укажем b.wayne3. А если и он занят? Сколько раз нам придётся писать условие? Мы не можем этого знать. Нам нужно проверять условие до тех пор, пока мы не получим нужный результат.

Для этого у нас есть команда while – комбинация условия и цикла. Пока выполняется условие, будет выполняться цикл. Синтаксис такой:

```
while условие
do команда
done
```

The screenshot shows two terminal windows side-by-side. Both have a dark theme with a light-colored terminal area. The left terminal window has a title bar with the window icon, 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. It contains the command: [user@centos8 ~]\$ nano while. The right terminal window also has a title bar with the same menu items. It contains the command: [user@centos8 ~]\$ cat while. Below these commands, both terminals show the contents of the 'while' script. The script consists of four lines: 'while [-f file]', 'do echo File exists', 'done', and '[user@centos8 ~]\$ chmod +x while'. To the right of the 'done' line, the output of the 'cat' command is shown, repeating 'File exists' five times. Below the 'chmod' command, the right terminal shows '[user@centos8 ~]\$ touch file'. Finally, at the bottom of the right terminal, there is a small square icon with a white border.

```
[user@centos8 ~]$ nano while
[user@centos8 ~]$ cat while
while [ -f file ]
do echo File exists
done
[user@centos8 ~]$ chmod +x while
[user@centos8 ~]$ touch file
[user@centos8 ~]$ 
```

File exists
File exists

Как мы помним, условие – это просто любая команда, главное статус её выхода – 0, или что-то другое. Например:

```
while [ -f file ]
do echo file exists
done
```

То есть, пока файл есть, echo будет писать такой текст. Попробуем создать файл:

```
touch file
```

и запустить команду. Как видите, команда echo постоянно выдаёт текст.

The screenshot shows two terminal windows side-by-side. The left terminal window contains the following commands:

```
[user@centos8 ~]$ nano while
[user@centos8 ~]$ cat while
while [ -f file ]
do echo File exists
done
[user@centos8 ~]$ chmod +x while
[user@centos8 ~]$ touch file
[user@centos8 ~]$ rm file
[user@centos8 ~]$
```

The right terminal window shows the output of the 'cat' command on the 'while' script, which prints 'File exists' nine times, followed by the prompt '[user@centos8 ~]\$'.

Попробуем удалить файл:

```
rm file
```

while получил код выхода 1 и закончил свою работу.

The screenshot shows two terminal windows side-by-side. The left terminal window contains the following commands:

```
[user@centos8 ~]$ nano while
[user@centos8 ~]$ cat while
i=1
while [ -f file ]
do echo $i
  ((i++))
done
[user@centos8 ~]$ touch file
[user@centos8 ~]$
```

The right terminal window shows the output of the 'cat' command on the 'while' script, which prints integer values from 31474 to 31485 sequentially, followed by the prompt '^C [user@centos8 ~]\$'.

С while часто используют инкремент. Это такая операция увеличения переменной. Например, берут переменную *i* и перед циклом дают ей какое-то значение, например 1. Во время выполнения цикла её значение увеличивают. То есть при каждой итерации значение переменной будет увеличиваться. Увеличивать значение в bash-е можно по разному, хоть с помощью математических операций, так и используя специальный оператор `++`:

```
((i++))
```

С помощью `<->` можно, соответственно, уменьшать значение. Заменим echo, чтобы видеть значение переменной:

```
echo $i
```

и попробуем запустить скрипт:

```
./while
```

Как видите, очень быстро переменная достигла больших значений, а значит while сделал столько итераций.

The image shows two terminal windows side-by-side. The left terminal window shows the creation of a file named 'while' with nano, then its content is displayed with cat. The right terminal window shows the execution of the script with ./while, which prints the numbers 1 through 5 to the console before being interrupted with ^C.

```
[user@centos8 ~]$ nano while
[user@centos8 ~]$ cat while
i=1
while [ -f file ]
do echo $i
((i++))
sleep 1
done
[user@centos8 ~]$ 
```

```
[user@centos8 ~]$ ./while
1
2
3
4
5
^C[user@centos8 ~]$ 
```

Можно, кстати, использовать команду sleep, чтобы заставить скрипт подождать сколько-то секунд, прежде чем выполнить следующую команду:

```
sleep 1
```

Как видите, теперь итерация происходит раз в секунду:

```
./while
```

The image shows two terminal windows side-by-side. The left terminal window shows the creation of a file named 'while' with nano, then its content is displayed with cat. The right terminal window shows the execution of the script with ./while, which prints the number 1 and then waits for a second before printing 'New user is o.queen1'.

```
[user@centos8 ~]$ nano while
[user@centos8 ~]$ cat while
user=o.queen
i=1
while cut -d: -f1 /etc/passwd | grep -w $user > /dev/null
do user=$user$i
((i++))
done
echo New user is $user
[user@centos8 ~]$ 
```

```
[user@centos8 ~]$ ./while
[User@centos8 ~]$ New user is o.queen1
[User@centos8 ~]$ 
```

Хорошо, попробуем применить while к нашей задаче. Для начала напишем условие проверки наличия логина. В прошлый раз я показал, как с помощью grep-а найти нужный логин в passwd, сделаем также. Для тестов укажем переменную user=o.queen. Условие в while поставим проверку наличия пользователя:

```
while cut -d: -f1 /etc/passwd | grep -w $user
```

Таким образом мы проверяем, есть ли пользователь в passwd и, если есть, запускаем команды внутри цикла. Сделаем так, чтобы команда внутри цикла меняла значение переменной:

```
user=$user$i
```

то есть о.queen превратится в о.queen1. Переменная i станет 2:

```
((i++))
```

Убираем sleep, он нам не нужен. Попытаемся прочесть наш цикл. При запуске скрипта while проверяет, есть ли пользователь о.queen в passwd. Если он нашёл такого пользователя, значит условие выполнилось. Если условие выполнилось, значит while запускает команды – сначала он меняет значение переменной user на о.queen1. Затем он меняет значение переменной i на 2. Происходит итерация – теперь while ищет в passwd пользователя о.queen1. Если он не находит – цикл заканчивается, значение переменной остаётся о.queen1 и запускаются следующие команды. Если же он нашёл юзера о.queen1, то переменная становится о.queen2, переменная i становится 3 и так до тех пор, пока grep не скажет, что такого пользователя нет. Давайте в конце выведем полученное значение переменной:

```
echo New var is $user
```

Запустим скрипт:

```
./while
```

Как видите, теперь переменная user стала о.queen1, а дальше можно эту переменную использовать для создания пользователя. Ну и чтобы не видеть вывод команды grep, просто направим его в /dev/null:

```
> /dev/null
```

```
GNU nano 2.9.8                                myscript

# Functions
check_user() {
i=1
while cut -d: -f1 /etc/passwd | grep -w $user > /dev/null
do user=$user$i
    ((i++))
done
}

create_user() {
```

Хорошо, теперь скопируем полученный цикл и вставим его в наш скрипт. Сделаем это в виде функции – check_user:

```
check_user() {
    i=1
    ...
}
```

```

elif [ -f "$file" ]
then
IFS=$'\n'
for line in $(cut -d, -f2,3,4,5 $file | tail -n +2 | tr '[:upper:]' '[:lower:]' | tr , ' ')
do
    user=$(echo $line | cut -c1).$(echo $line | cut -d' ' -f2)
    group=$(echo $line | cut -d' ' -f4)
    bday=$(echo $line | cut -d' ' -f3)
    check_user
    echo Username: $user Group: $group
    create_user
done

```

Ну и укажем эту функцию в нашем цикле for, который создаёт пользователей из файла.

```

[user@centos8 ~]$ nano myscript
[user@centos8 ~]$ sudo cp users.csv /var/
[user@centos8 ~]$ sudo ./myscript
Username: b.wayne1 Group: it
groupadd: group 'it' already exists
%it ALL=(ALL) ALL
mkdir: cannot creat
File Edit View Search Terminal Help
Username: b.wayne12 [user@centos8 ~]$ tail -5 /etc/passwd
groupadd: group 'seb.allen:x:1138:1004::/home/it/b.allen:/bin/bash
%security ALL=(ALL)b.wayne1:x:1139:1004:Birthday 19.02:/home/it/b.wayne1:/bin/bash
mkdir: cannot creatb.wayne12:x:1140:1005:Birthday 16.05:/home/security/b.wayne12:/bin/bash
Username: b.wayne12b.wayne123:x:1141:100:Birthday 18.06:/home/users/b.wayne123:/sbin/nologin
groupadd: group 'usb.wayne1234:x:1142:1004:Birthday 14.03:/home/it/b.wayne1234:/bin/bash
mkdir: cannot creat [user@centos8 ~]$
Username: b.wayne12
groupadd: group 'it'

```

Сохраним, скопируем новый файл users.csv в директорию /var/:

```
sudo cp users.csv /var/
```

и запустим скрипт:

```
sudo ./myscript
tail -5 /etc/passwd
```

Как видите, для всех тёзок создались аккаунты, хотя логины получились не такими, как мы ожидали. Почему так получилось и как это исправить – это задача для вас.

```

[user@centos8 ~]$ nano until      File does not exists
[user@centos8 ~]$ cat until      File does not exists
until [ -f file ]                File does not exists
do echo File does not exists    File does not exists
done                             File does not exists
[user@centos8 ~]$ chmod +x until  File does not exists
[user@centos8 ~]$ rm file        File does not exists
[user@centos8 ~]$ 

```

Напоследок, рассмотрим команду until. Если в while цикл продолжает работать пока условие верно, то есть код выхода условия 0, то в until наоборот – цикл будет работать пока условие неверно, то есть код выхода не 0. То есть, условно, while - пока всё хорошо, делать что-то. А until – пока не станет хорошо, делать что-то. Синтаксис практически одинаковый:

```
until [ -f file ]
do echo file does not exists
done
```

Дадим права, удалим файл и попробуем запустить:

```
chmod +x until
rm file
./until
```

Как видите, скрипт говорит, что файла нет.

```
[user@centos8 ~]$ nano until          File does not exists
[user@centos8 ~]$ cat until          File does not exists
until [ -f file ]                  File does not exists
do echo File does not exists      File does not exists
done                                File does not exists
[user@centos8 ~]$ chmod +x until      File does not exists
[user@centos8 ~]$ rm file            File does not exists
[user@centos8 ~]$ touch file        File does not exists
[user@centos8 ~]$ █                   File does not exists
                                         File does not exists
                                         [user@centos8 ~]$ █
```

То есть, пока не выполнится условие, пока не появится файл:

```
touch file
```

until будет продолжать работать.

```
GNU nano 2.9.8

while ! [ -f file ]
do echo File does not exists
done
```

Но, как мы помним, мы можем использовать тот же while с восклицательным знаком:

```
while ! [ -f file ]
...
```

что даст, по сути, тот же результат. Разве что читать скрипт с until где-то проще, вместо того, чтобы обращать значение while.

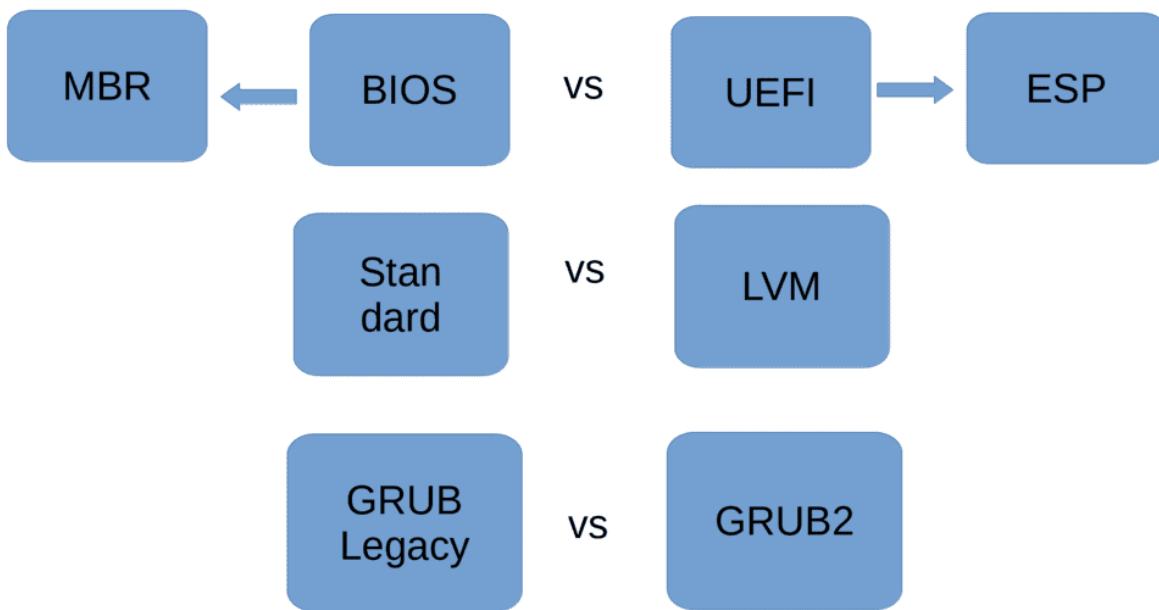
В этот раз мы с вами разобрали while и until. Можно наткнуться на различные способы использовать тот же while, until, for и другие команды. Но если понимать, что, допустим, тому же while нужен статус

выхода и безразлична сама команда, то многие способы применения станут понятнее. Со скриптами мы сделаем перерыв, так как много других важных тем, но, я надеюсь, что вы стали лучше понимать, что такое скрипты, как их читать и писать. Для примера, постараитесь прочитать те же файлы `/etc/bashrc` и `/etc/profile`, которые мы разбирали раньше. Это просто скрипты, которые выполняются при запуске bash-а. Возможно вам неизвестны все ключи – но это нормально, всегда можно обратиться к документации.

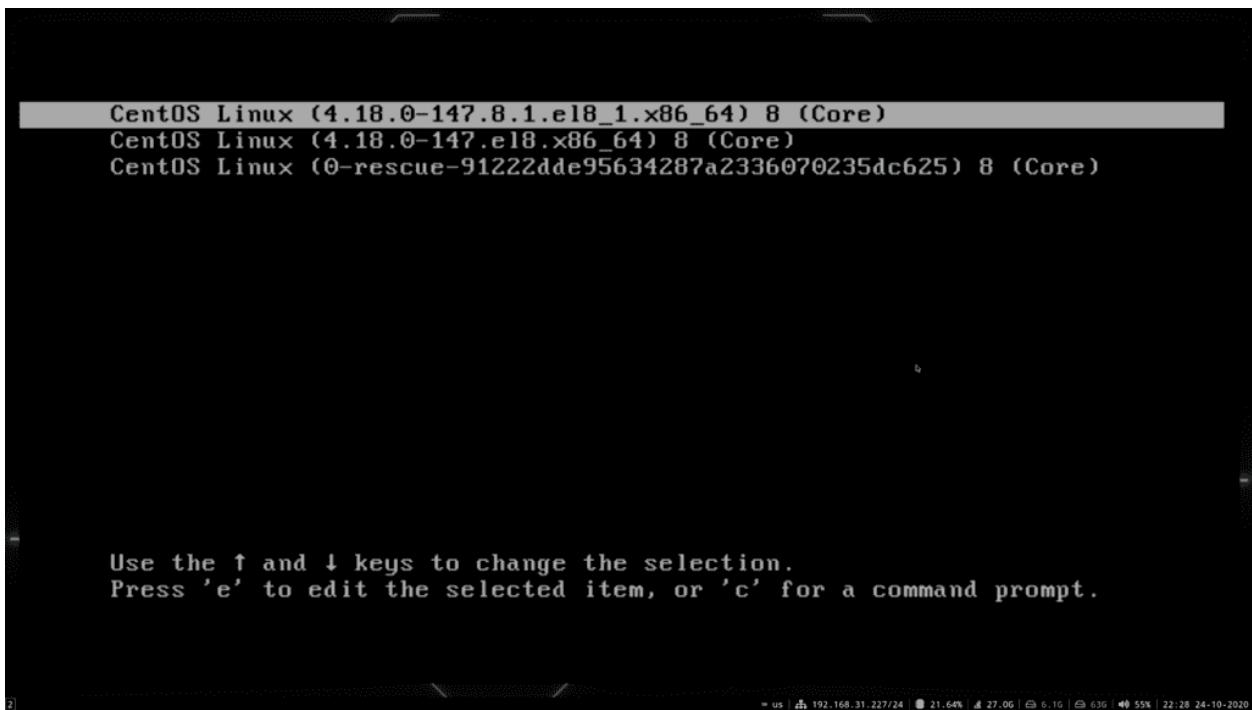
2.33 33. Загрузчик GRUB

2.33.1 33. Загрузчик GRUB

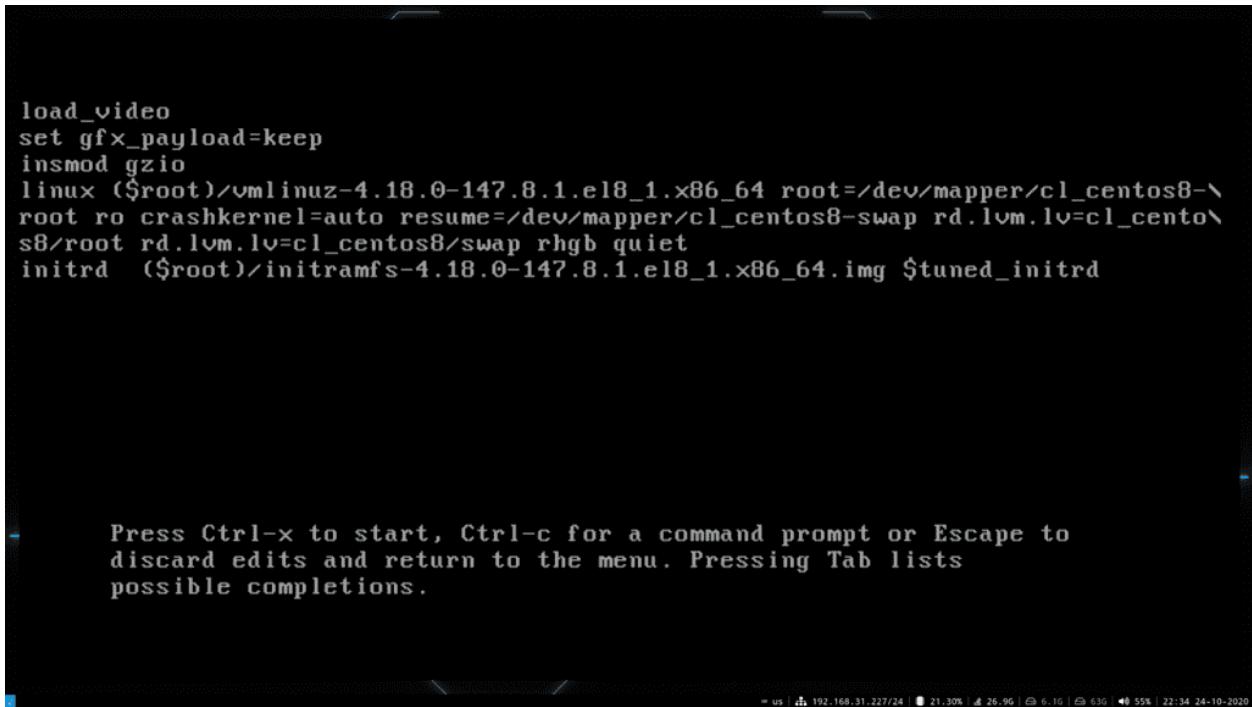
Для работы какой-либо программы её нужно предварительно запустить. Какие-то программы мы запускаем сами, какие-то программы запускают другие программы. И при запуске компьютера есть определённая последовательность, какие программы что и зачем запускают. Процесс запуска операционной системы администратору нужно знать, потому что, во-первых, это помогает выявить и решить какие-то проблемы, во-вторых, это помогает какие-то проблемы предотвратить. Ну и в-третьих, помогает лучше понять работу операционной системы. Например, вспомните, мы с вами испортили запись в `fstab`, у нас система не загрузилась, потом мы её исправили и всё заработало.



Есть много сценариев запуска операционной системы, которые зависят от определённых условий. Например, мы с вами говорили про BIOS и UEFI, и тут, как минимум, уже два сценария запуска – с использованием MBR или раздела EFI system partition. Также мы с вами разбирали стандартные разделы, LVM, RAID – и это тоже добавляет варианты – используется ли для корня стандартные разделы или какой-то нестандартный, допустим, LVM. Ну и сегодня мы будем говорить о загрузчиках, которых много. Даже с учётом того, что мы будем разбирать только один загрузчик – grub, у него есть разные версии, от чего тоже появляются вариации. Но не будем усложнять, обо всём по порядку.

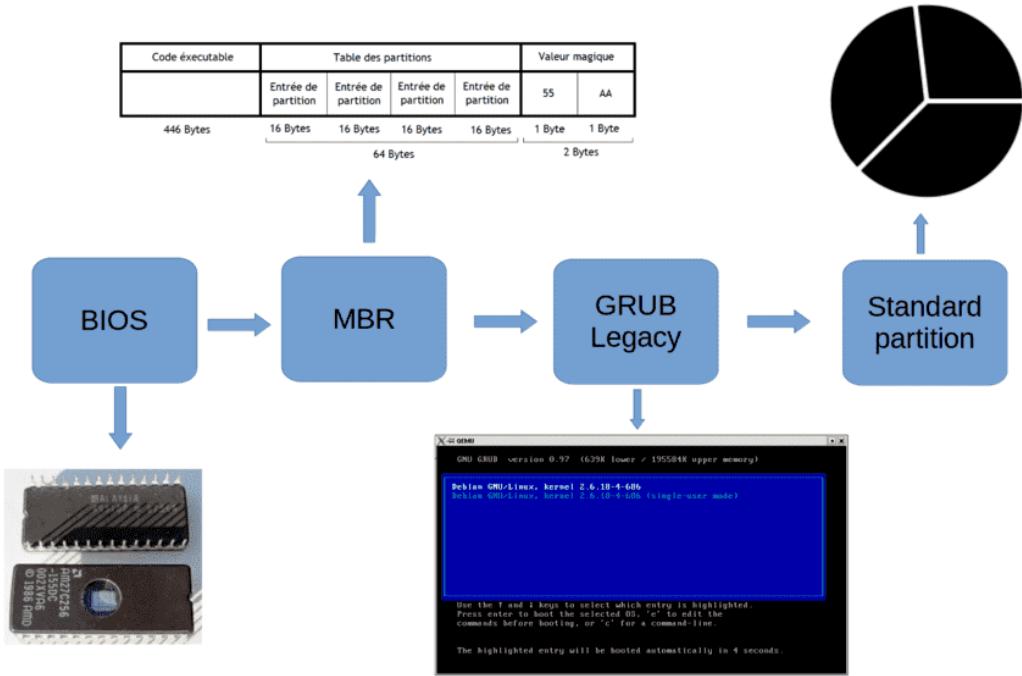


Со всем вышесказанным мы знакомы, кроме загрузчика. Загрузчик – это программа, которая загружает операционную систему. Это особая программа, потому что она работает ещё до того, как загрузилась операционная система, ядро и всякие другие программы. Тут можно понять, что у загрузчика должны быть свои, хотя бы минимальные драйвера для работы с компьютером – где-то он обращается к биосу, где-то напрямую к железу. При включении GNU/Linux-а мы видим такое окно – это такой интерфейс загрузчика Grub. Есть и другие загрузчики, но grub самый популярный, он стоит по дефолту на многих системах и, чтобы не усложнять, мы будем говорить только о нём.

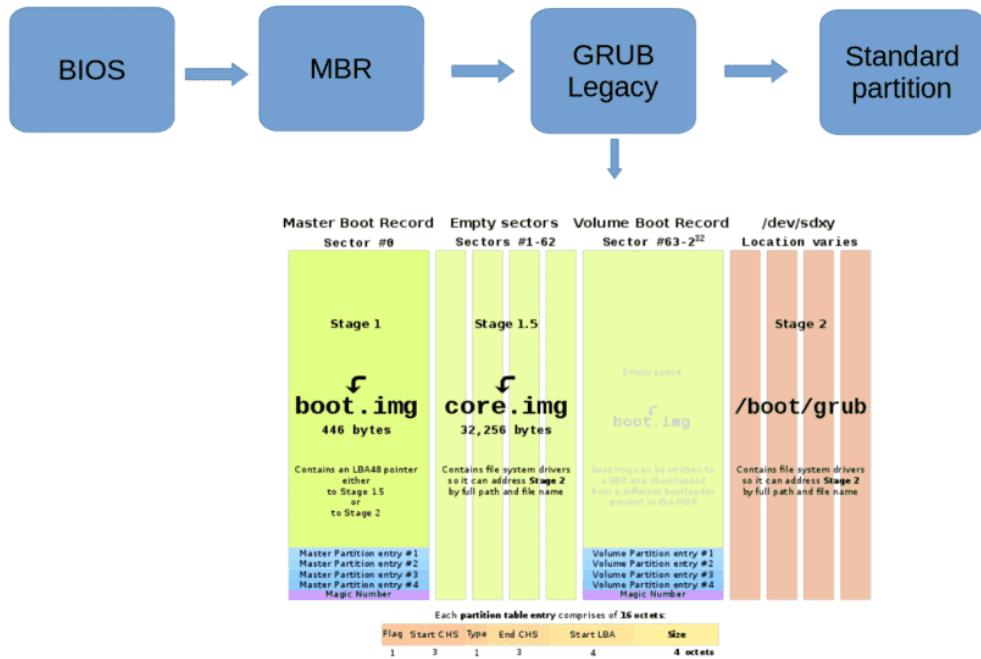


И так, при включении у нас появляется окно grub, где мы можем выбрать нужную операционную

систему. Если у нас на дисках установлены разные операционные системы мы сможем загрузить любую из них. К тому же тут также есть возможность загрузить одну и ту же операционку с разными версиями установленного ядра. Также отсюда мы можем повлиять на процесс загрузки – например, нажать «e» на нужной записи и изменить какие-то параметры. Но мы к этому ещё придём. Плюс grub можно кастомизировать – поменять шрифт, рамки, цвета, поставить фон и всё такое. Чтобы лучше понять grub нам нужно хотя бы разок пройтись по сценарию запуска операционной системы.



Возьмём самый простой сценарий: BIOS – MBR – GRUB – корень на стандартном разделе. И так, вы нажали кнопку включения на компьютере, загрузился BIOS, какие-то свои задачи выполнил, а дальше обращается к порядку загрузки – диски, флешки, сеть и всё такое. Предположим, у нас на первом месте жёсткий диск. BIOS загружает первый сектор – 512 байт этого диска, то есть MBR. Как мы помним, у нас в MBR есть загрузчик и таблица разделов. Для загрузчика выделено 446 байт – это очень маленький объём, куда невозможно поместить полностью grub – поэтому здесь лежит программа, в которой прописан блочный путь к основной части grub-a. Блочный – потому что поддержку файловой системы в такой маленький объём не запихнуть. Этот шаг, когда загружается маленькая часть grub из MBR называется этапом 1.



В случае со старым железом BIOS мог видеть только первые 1024 цилиндра жёсткого диска, т.е. примерно первые 500 мегабайт, а если основная часть grub-а находилась в другом месте – то не получилось бы его загрузить. Поэтому для таких случаев существовал этап 1.5, когда загружалась чуть большая часть grub-а, расположенная в первых 62 секторах диска, где лежали драйвера для файловых систем. Загрузив их grub мог уже полноценно найти и загрузить основную свою часть, которая лежит в директории /boot/grub – т.е. приступить к этапу 2.

```

load_video
set gfx_payload=keep
insmod gzio
linux ($r0)/vmlinuz-4.18.0-147.8.1.el8_1.x86_64 root=/dev/mapper/cl_centos8-
root ro crashkernel=auto resume=/dev/mapper/cl_centos8-swap rd.lvm.lv=cl_cento-
s8/root rd.lvm.lv=cl_centos8/swap rhgb quiet
initrd ($r0)/initramfs-4.18.0-147.8.1.el8_1.x86_64.img $tuned_initrd

```

Grub, загрузив свою основную часть, обращается к файлу /boot/grub/grub.cfg и берёт оттуда записи, что и как грузить. Если говорить про Linux, то задача grub – загрузить ядро и передать ему нужные параметры. Эти параметры видны в строчке linux: файл vmlinuz, который мы видели в директории /boot; файловая система (root=...), где лежит корень; rhgb – чтобы отображать анимацию, пока грузится операционная система; quiet – чтобы скрыть подробный вывод при запуске.

```
[ 4.074911] sd 4:0:0:0: [sdc] 2097152 512-byte logical blocks: (1.07 GB/1.00 GiB)
[ 4.075419] sd 4:0:0:0: [sdc] Write Protect is off
[ 4.075807] sd 4:0:0:0: [sdc] Write cache: enabled, read cache: enabled, doesn't support DPO or FUA
[ 4.082551] sdb: sdb1
[ 4.083980] sd 3:0:0:0: [sdb] Attached SCSI disk
[ 4.092696] sdc: sdc1
[ 4.093898] sd 4:0:0:0: [sdc] Attached SCSI disk
[ 4.135466] sr 1:0:0:0: [sr0] scsi3-mmc drive: 32x/32x xa/form2 tray
[ 4.135868] cdrom: Uniform CD-ROM driver Revision: 3.20
[ 4.154244] e1000 0000:00:03.0 eth0: (PCI:33MHz:32-bit) 00:00:27:c6:77:86
[ 4.154655] e1000 0000:00:03.0 eth0: Intel(R) PRO/1000 Network Connection
[ 4.196348] e1000 0000:00:03.0 enp0s3: renamed from eth0
[ OK ] Found device /dev/mapper/cl_centos8-root.
[ OK ] Reached target Initrd Root Device.
[ 4.833679] random: crng init done
[ 4.834057] random: 7 urandom warning(s) missed due to ratelimiting
[ OK ] Found device /dev/mapper/cl_centos8-swap.
      Starting Resume from hibernation using device /dev/mapper/cl_centos8-swap...
[ OK ] Started Resume from hibernation using device /dev/mapper/cl_centos8-swap.
[ OK ] Reached target Local File Systems (Pre).
[ OK ] Reached target Local File Systems.
      Starting Create Volatile Files and Directories...
[ OK ] Started Create Volatile Files and Directories.
[ OK ] Reached target System Initialization.
[ OK ] Reached target Basic System.
[ OK ] Started dracut initqueue hook.
[ OK ] Reached target Remote File Systems (Pre).
[ OK ] Reached target Remote File Systems.
      Starting File System Check on /dev/mapper/cl_centos8-root...
[ OK ] Started File System Check on /dev/mapper/cl_centos8-root.
      Mounting /sysroot...
[ 5.255925] SGI XFS with ACLs, security attributes, no debug enabled
[ 5.269370] XFS (dm-0): Mounting U5 Filesystem
[ 5.496047] XFS (dm-0): Starting recovery (logdev: internal)
[ 5.515175] XFS (dm-0): Ending recovery (logdev: internal)
```

Давайте, для примера, сотрём последние две опции — используем стрелки для перемещения, на строке linux уберём «rhgb quiet» и нажмём **ctrl+x**, чтобы загрузиться. Как видите, теперь при загрузке отображается куча информации, что может быть полезно, если по какой-то причине система не грузится и мы пытаемся найти причину. После перезагрузки эти опции опять будут на месте, потому что мы их стёрли в текущей сессии, а не в конфиг файле.

```
[user@centos8 ~]$ ls /lib/modules
4.18.0-187.el8.x86_64 4.18.0-193.19.1.el8_2.x86_64 4.18.0-193.el8.x86_64
[user@centos8 ~]$ cat /lib/modules/$(uname -r)/modules.builtin | grep -e scsi -e ext4
kernel/drivers/scsi/scsi_mod.ko
kernel/drivers/scsi/device_handler/scsi_dh_rdac.ko
kernel/drivers/scsi/device_handler/scsi_dh_hp_sw.ko
kernel/drivers/scsi/device_handler/scsi_dh_emc.ko
kernel/drivers/scsi/device_handler/scsi_dh_alua.ko
[user@centos8 ~]$ █
```

Окей, grub запустил ядро. Что дальше? Ядро начинает запускаться, находить устройства и загружать драйвера. Но для этого ядру нужны модули. Вспомните про модульность ядра — у ядра есть встроенные модули и загружаемые. Встроенные уже в самом файле vmlinuz, а загружаемые лежат в директории `/lib/modules/`. Ядро загрузит то что сможет, используя встроенные модули, а дальше ядру нужен доступ в директорию `/lib/modules`. Для этого ядро должно примонтировать корень. Только вот такой нюанс — для того, чтобы примонтировать корень, ядру нужны модули — как минимум, модуль `scsi` и модуль файловой системы. Давайте посмотрим список встроенных модулей:

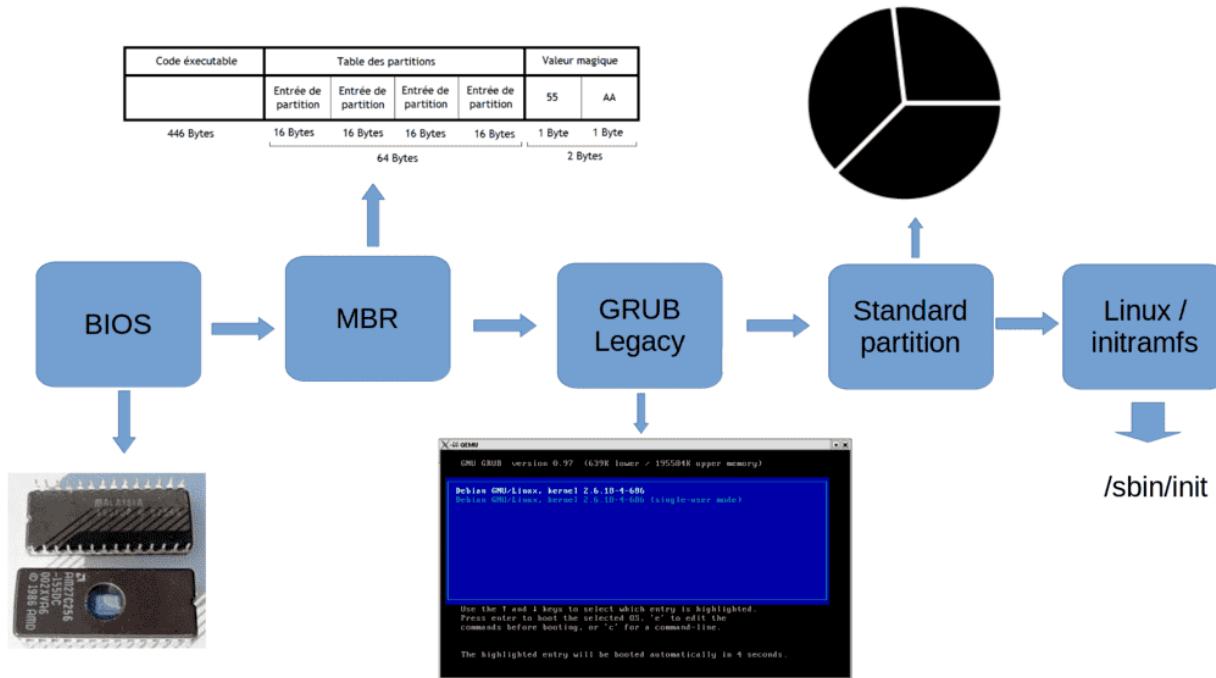
```
cat /lib/modules/$(uname -r)/modules.builtin | grep -e scsi -e ext4
```

Как видите, тут есть модули scsi, но нет модуля ext4. То есть ядро просто не сможет примонтировать корень, чтобы взять оттуда модули. При этом, чтобы эти модули были, ему нужно примонтировать корень.

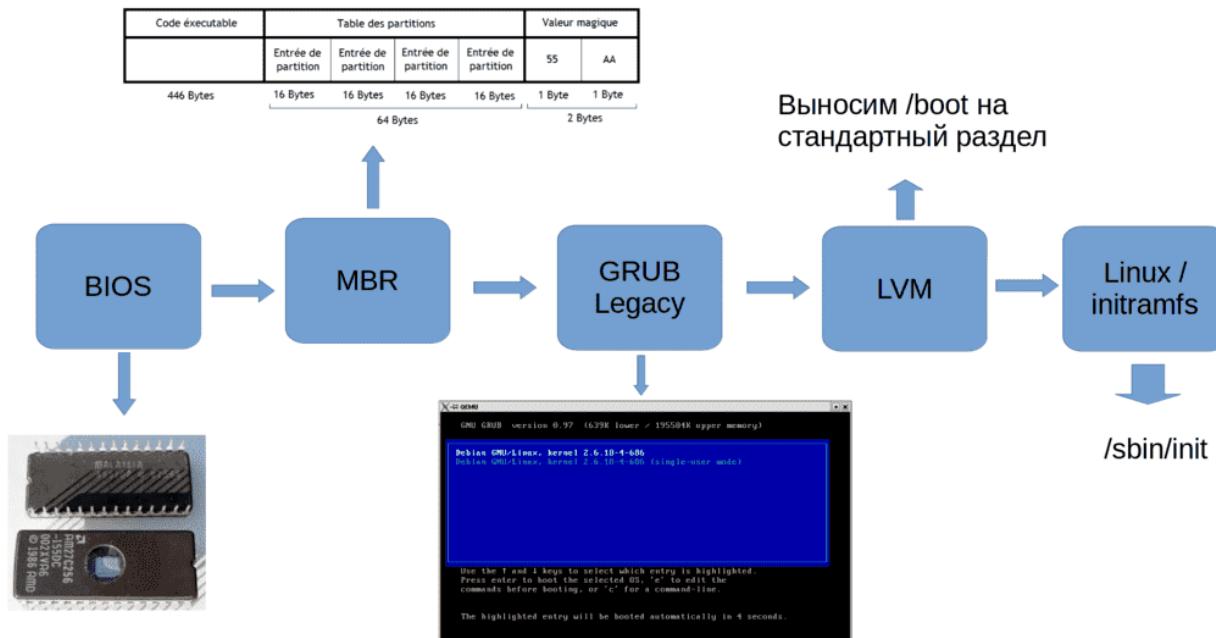
```
load_video
set gfx_payload=keep
insmod gzio
linux ($root)/vmlinuz-4.18.0-147.8.1.el8_1.x86_64 root=/dev/mapper/cl_centos8-
root ro crashkernel=auto resume=/dev/mapper/cl_centos8-swap rd.lvm.lv=cl_cento-
s8/root rd.lvm.lv=cl_centos8/swap rhgb quiet
initrd ($root)/initramfs-4.18.0-147.8.1.el8_1.x86_64.img $tuned_initrd
```

Можно было бы встроить в ядро все нужные модули, но это сделает ядро значительно тяжелее, файловых систем много, да и если брать LVM и прочее.. в общем, не выход. Эта проблема решается по другому. В grub под строкой с linux есть еще одна строка – initrd – initial ram disk. Как видите, она указывает на файл. Этот файл – ram disk – временный образ корня, который специально предназначен для решения указанной проблемы. При загрузке grub монтирует этот файл в качестве корня и ядро может с ним работать. Здесь оно находит все нужные драйвера для того, чтобы загрузить настоящий корень, после чего ядро переключается на основной корень. Этот временный корень не содержит все файлы, а только необходимый минимум, какие-то драйвера файловых систем и т.п., чтобы можно было перейти на нужный корень. Но в каких-то дистрибутивах сюда включены чуть ли не все драйвера и некоторые программы. И если у вас какие-то проблемы с корнем, не получается монтировать, или, например, вы хотите сделать fsck корня, то используя этот временный корень вы можете провести какие-то работы.

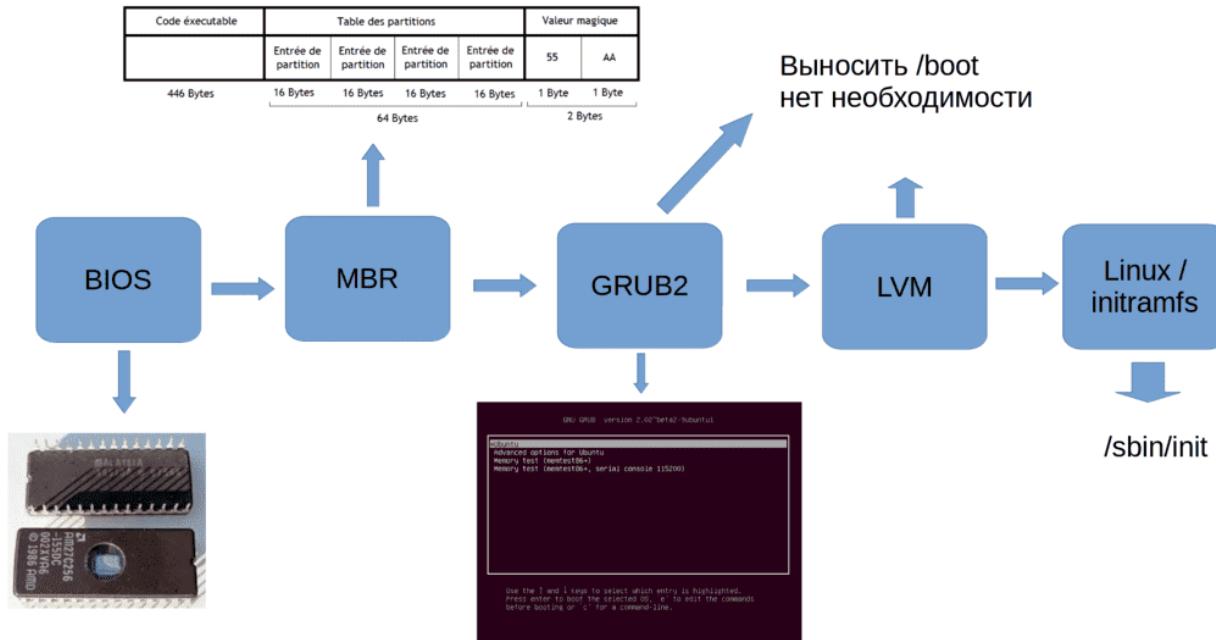
Но, на самом деле, если присмотреться, файл называется не initrd, а initramfs – initial ram file system. Initramfs пришёл на замену initrd, так как у initrd есть определённые недостатки. Например, initrd был файлом образом блочного устройства, внутри которого была файловая система, а значит ядру для работы с ним нужен был хотя бы один встроенный модуль файловой системы. Initramfs это больше архив, который распаковывается в виртуальную файловую систему tmpfs. А tmpfs является частью ядра, предназначеннной для всяких файловых систем, работающих в оперативке. Так вот, когда ядро видит initramfs, оно берёт оттуда нужные модули, а также запускает программу, ну или просто скрипт, с названием init. Она может выполнять какие-то команды, если этого захотели разработчики дистрибутива, в итоге монтирует настоящий корень и переключается на него. Есть небольшие отличия в работе между initrd и initramfs, но такие подробности нам сейчас не нужны.



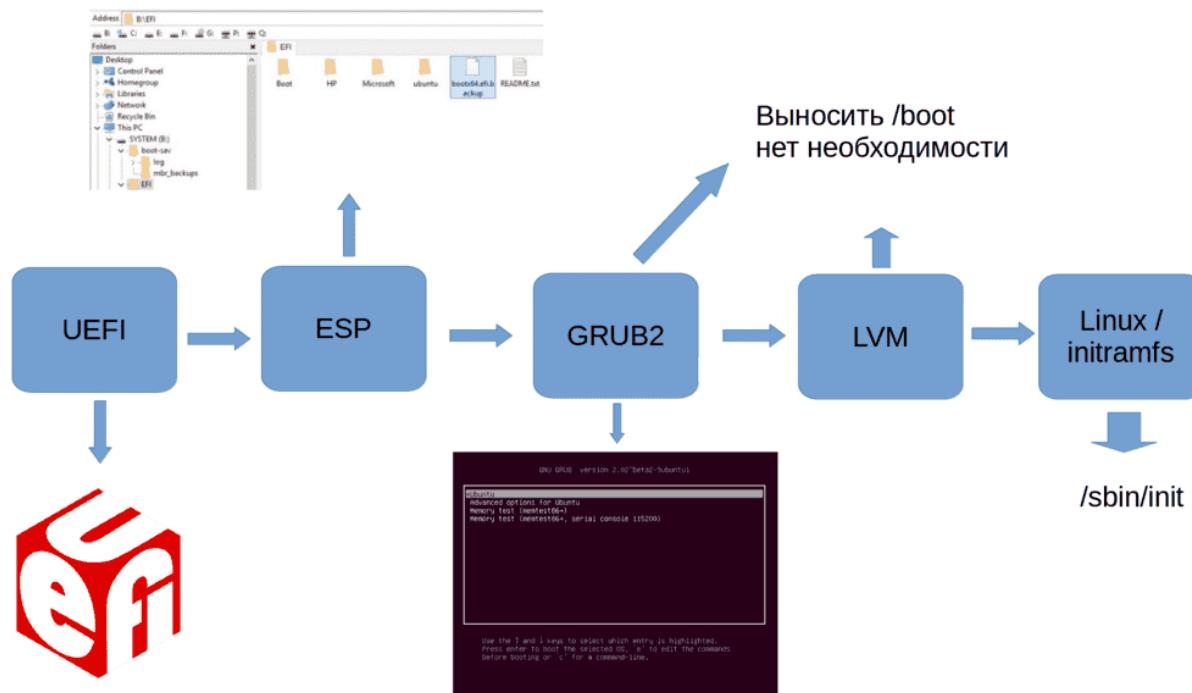
Так вот, после всей этой инициализации, когда ядро взяло нужные модули из initramfs, init запускает программу /sbin/init в настоящем корне, которая стартует систему инициализации. Система инициализации – это отдельная тема, которую будем разбирать в следующий раз. Пока что мы разобрали только один сценарий: BIOS – MBR – GRUB – стандартный раздел. Давайте вкратце пройдёмся по нему – BIOS грузит загрузчик из MBR на 446 байт, это первый этап загрузки GRUB-а. Дальше GRUB грузит свой второй этап, ссылаясь на /boot/grub. Полностью загруженный GRUB видит в конфигах путь к ядру и initramfs, загружает ядро и initramfs в оперативку, ядро подгружает нужные модули благодаря initramfs, запускает init, который в итоге переключается на реальный корень и запускает программу /sbin/init – то есть систему инициализации. В других сценариях в целом многое будет схоже, но отличия всё же есть, давайте их разберём.



Заменим стандартный раздел на LVM. Допустим, корень у нас в LVM. Чтобы grub мог загрузить ядро и initramfs, ему нужен доступ в директорию /boot, но если они находятся на LVM разделе, grub не сможет их увидеть. В таких случаях директорию /boot выносили на отдельный стандартный раздел. Благодаря чему grub мог загрузить ядро и initramfs со стандартного раздела, а дальше ядро находило в initramfs модули для LVM и могло прмонтировать реальный корень.



Но так было раньше, со старой версией Grub, которая сейчас называется grub-legacy. Современная версия grub называется grub2 и она поддерживает LVM, благодаря чему нет необходимости выносить /boot на отдельный раздел.



Если заменить BIOS на UEFI, то отпадает необходимость в загрузчике в MBR. При включении UEFI ищет раздел EFI system partition и загружает оттуда bootx64.efi или grubx64.efi. А это запускает grub, который обращается к файловой системе, где находится директория /boot/grub и дальше всё как мы говорили.

```
[user@centos8 ~]$ sudo fdisk -l /dev/sda | grep type
Disklabel type: dos
[user@centos8 ~]$ mount | grep /boot
/dev/sda1 on /boot type ext4 (rw,relatime,seclabel)
[user@centos8 ~]$ df -h /
Filesystem           Size   Used   Avail   Use%   Mounted on
/dev/mapper/cl_centos8-root    17G   6.4G   11G   38%   /
[user@centos8 ~]$ sudo ls /boot/grub2/
device.map  fonts  grub.cfg  grubenv  i386-pc
[user@centos8 ~]$ █
```

Давайте посмотрим, как всё организовано на нашей виртуалке. Virtualbox для гостевых машин использует BIOS. Таблица разделов у нас dos, т.е. используется MBR:

```
sudo fdisk -l /dev/sda | grep type
```

Ну и если посмотреть файловые системы:

```
mount | grep /boot
```

можно увидеть, что boot вынесен в отдельный раздел, а для корня используется LVM:

```
df -h /
```

При этом grub у нас второй версии:

```
ls /boot/grub2
```

Если вы читали внимательно, вы помните, что grub2 поддерживает LVM, и выносить /boot на стандартный раздел не нужно. Но официально RedHat рекомендует держать /boot всё таки на стандартном разделе. Объяснения почему я не нашёл, но, предполагаю, что дело в решении проблем. Если у вас возникнут проблемы с LVM, вы не сможете загрузить даже ядро с initramfs, потому что они лежат на логическом разделе, вам придётся использовать live-cd. Стандартный раздел для /boot позволит вам прогрузить хотя бы ядро с initramfs. Хотя, возможно, есть и другие причины, но о причинах в документации ничего не сказано.

```
[user@centos8 ~]$ sudo nano /etc/default/grub
[user@centos8 ~]$ cat /etc/default/grub | grep CMDLINE
GRUB_CMDLINE_LINUX="crashkernel=auto resume=/dev/mapper/cl_centos8-swap rd.lvm.lv=cl_centos8/root rd.lvm.lv=cl_centos8/swap"
[user@centos8 ~]$ sudo grub2-mkconfig
Generating grub configuration file ...
#
# DO NOT EDIT THIS FILE
#
# It is automatically generated by grub2-mkconfig using templates
# from /etc/grub.d and settings from /etc/default/grub
#
### BEGIN /etc/grub.d/00_header ###
set pager=1
```

Ну и напоследок, давайте немного подредактируем grub. Хотя сам конфиг это /boot/grub2/grub.cfg, вручную редактировать этот файл не стоит, он будет перезаписываться при обновлениях. Чтобы наши изменения оставались даже после обновления, нужно редактировать файл /etc/default/grub:

```
sudo nano /etc/default/grub
```

Например, уберём rhgb и quiet в строчке GRUB_CMDLINE_LINUX и сохраним. После этого нужно обновить конфиг файла grub-a. Сначала убеждаемся, что всё нормально, просто запускаем команду:

```
grub2-mkconfig
```

Если всё нормально, то мы увидим будущий конфиг.

```
[user@centos8 ~]$ sudo grub2-mkconfig -o /boot/grub2/grub.cfg
Generating grub configuration file ...
done
[user@centos8 ~]$ █
```

Затем перенаправляем вывод этой команды в файл /boot/grub2/grub.cfg:

```
sudo grub2-mkconfig -o /boot/grub2/grub.cfg
```

Ну и для проверки можем перезагрузиться.

Если говорить про initramfs, стоит упомянуть возможность добавлять в него определённые модули. Пример из практики – у вас есть виртуальная машина на гипервизоре ESXi, это такой коммерческий гипервизор от компании VMWare. И вам нужно перенести виртуальную машину на гипервизор KVM – свободный гипервизор, активно используемый в Linux-ах. Если вы просто перенесёте файлы, скорее всего у вас виртуальная машина просто не запустится. Причина – ядро виртуалки при запуске просто не увидит корень, у вас не будет дисков sda, sdb и т.д.

```
[user@centos8 ~]$ cat /lib/modules/$(uname -r)/modules.builtin | grep virtio
kernel/drivers/char/hw_random/virtio-rng.ko
kernel/drivers/virtio/virtio.ko
kernel/drivers/virtio/virtio_ring.ko
kernel/drivers/virtio/virtio_pci.ko
[user@centos8 ~]$ sudo lsinitrd /boot/initramfs-$(uname -r).img | grep virtio
[sudo] password for user:
[user@centos8 ~]$ sudo lsinitrd /boot/initramfs-$(uname -r).img
Image: /boot/initramfs-4.18.0-193.19.1.el8_2.x86_64.img: 50M
=====
Early CPIO image
=====
drwxr-xr-x  3 root      root          0 Apr 24  2020 .
```

Чтобы ядро увидело диски, ему нужны будут другие драйвера scsi – virtio. Но они не встроены ни в ядро:

```
cat /lib/modules/$(uname -r)/modules.builtin | grep virtio
```

ни в initramfs:

```
sudo lsinitrd /boot/initramfs-$(uname -r).img | grep virtio
```

Список модулей в initramfs можно увидеть с помощью команды:

```
lsinitrd /boot/initramfs-$(uname -r).img
```

```
[user@centos8 ~]$ sudo nano /etc/dracut.conf.d/drivers.conf
[user@centos8 ~]$ cat /etc/dracut.conf.d/drivers.conf
add_drivers+=" virtio_blk virtio_scsi "
[user@centos8 ~]$ sudo dracut -f -v /boot/initramfs-$(uname -r).img $(uname -r)
dracut: Executing: /usr/bin/dracut -f -v /boot/initramfs-4.18.0-193.19.1.el8_2.x
86_64.img 4.18.0-193.19.1.el8_2.x86_64
dracut: dracut module 'busybox' will not be installed, because command 'busybox'
could not be found!
dracut: dracut module 'btrfs' will not be installed, because command 'btrfs' cou
ld not be found!
```

Чтобы добавить эти модули в initramfs, нужно создать файл в директории /etc/dracut.conf.d/ с названием, заканчивающимся на .conf и списком необходимых модулей:

```
add_drivers+=" virtio_blk virtio_scsi "
```

Далее необходимо сконфигурировать новый образ initramfs:

```
sudo dracut -f -v /boot/initramfs-$(uname -r).img $(uname -r)
```

```
[user@centos8 ~]$ sudo lsinitrd /boot/initramfs-$(uname -r).img | grep virtio
-rw-r--r-- 1 root      root      8968 Apr 24  2020 usr/lib/modules/4.18.0-19
3.19.1.el8_2.x86_64/kernel/drivers/block/virtio_blk.ko.xz
-rw-r--r-- 1 root      root      8756 Apr 24  2020 usr/lib/modules/4.18.0-19
3.19.1.el8_2.x86_64/kernel/drivers/scsi/virtio_scsi.ko.xz
[user@centos8 ~]$ █
```

После чего можем убедиться, что новые модули будут в initramfs:

```
sudo lsinitrd /boot/initramfs-$(uname -r).img | grep virtio
```

Подведём итоги. Сегодня мы с вами разобрали, как операционная система запускается – куда обращаются BIOS и UEFI, что такое загрузчик и какова его роль, зачем нужна директория /boot, почему ядро и initramfs находятся в этой директории, ну и что такое initramfs.

2.33.2 Практика

Вопросы

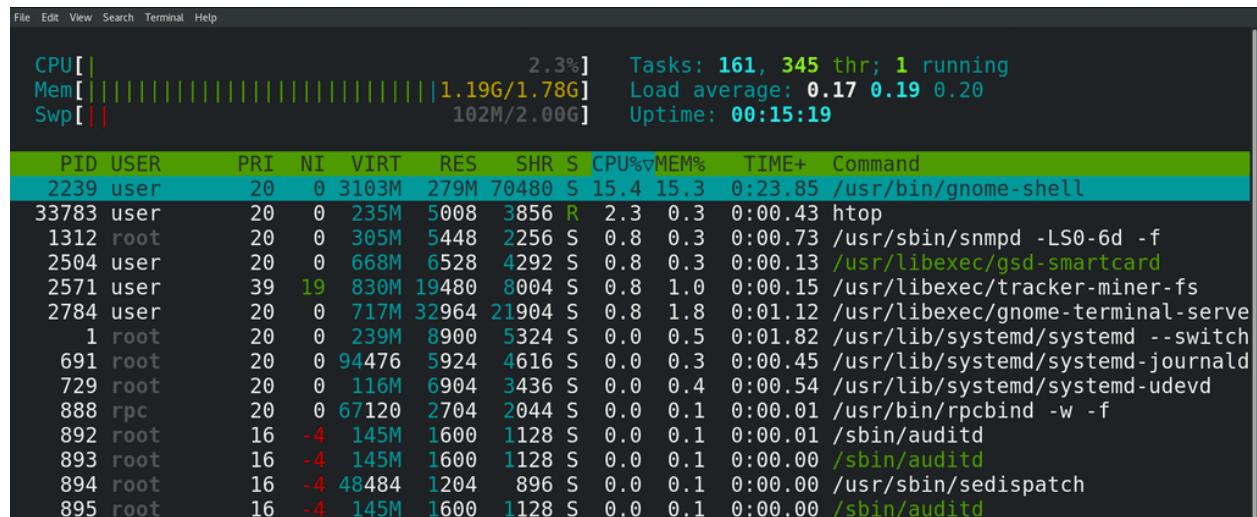
1. Опишите своими словами процесс запуска операционной системы.
2. Для чего нужен загрузчик?
3. В каких случаях нужно выносить /boot в отдельный раздел?
4. Как изменить настройки загрузчика?
5. Для чего нужен initramfs?
6. Чем отличается initrd от initramfs?
7. Как узнать, какие модули есть в файле initramfs?
8. Как понять, какая корневая ФС указана в пункте grub?

Задания

1. Найдите как и поменяйте фон загрузчика.
2. Создайте новый initramfs с модулями virtio.

2.34 34. Система инициализации - systemd

2.34.1 34. Система инициализации - systemd



Мы с вами помним: когда запускается программа – появляется соответствующий процесс. Но если посмотреть список процессов на свежезапущенной операционной системе, мы увидим более сотни процессов, хотя я всего лишь запустил эмулятор терминала, bash и htop. Значит, всё остальное было запущено другими программами.

То окно, в котором я залогинился – это ведь тоже какая-то программа, благодаря которой я и могу войти в систему. И вот этот графический интерфейс – тоже какая-то программа. Если я открою браузер и зайду на какой-то сайт – я может и не думаю об этом, но чтобы я смог зайти на этот сайт, сработала

ещё одна программа, которая отвечает за работу с сетью. Т.е. чтобы я мог выполнить какие-то простые действия, должны работать какие-то программы.

Демон (ст.-слав. *дёмонъ*^[3] от др.-греч. *δαίμον* [даймон] «дух», «божество»^{[4][5]}) — собирательное название сверхъестественных существ или **духов**, занимающих низшее по сравнению с богами положение^[5], которые могут играть как положительную, так и отрицательную роль^[5].

У древних греков существовало философское понятие **Даймоний**. **Сократ** и его последователи — **Платон**, **стоики** и другие, отождествляли с даймонием «внутренний голос» человека, **сознание**. В римской мифологии им соответствует **гений**^[5], в христианстве — **ангел-хранитель**^{[6][7][8]}.

Они всегда работают на фоне и помогают пользователям удобно пользоваться операционной системой. Все эти программы, которые работают на фоне, называются демонами, или даймонами - и тут отсылка не к библейским демонам, а к древнегреческой мифологии. Что-то типа ангелов хранителей.

Т.е., чтобы я мог нормально работать, должны работать демоны. Но чтобы они работали, их должен кто-то запустить. Этим занимается специальная программа - система инициализации. Есть различные реализации этой программы, мы с вами рассмотрим **systemd**, которая используется во многих популярных дистрибутивах, в том числе CentOS и RHEL.

```
[user@centos8 ~]$ ls -l /sbin/init
lrwxrwxrwx. 1 root root 22 Jul 21 2020 /sbin/init -> ../../lib/systemd/systemd
[user@centos8 ~]$
```

В прошлый раз мы остановились на том, что ядро вместе с initramfs запускает программу /sbin/init:

```
ls -l /sbin/init
```

/sbin/init олицетворяет систему инициализации, и, как видите, сейчас она показывает, что системой инициализации является **systemd**.

```
SYSTEMD(1)                      systemd                         SYSTEMD(1)

NAME
    systemd, init - systemd system and service manager

SYNOPSIS
    /usr/lib/systemd/systemd [OPTIONS...]

    init [OPTIONS...] {COMMAND}

DESCRIPTION
    systemd is a system and service manager for Linux operating systems. When run
    as first process on boot (as PID 1), it acts as init system that brings up and
    maintains userspace services.

    For compatibility with SysV, if systemd is called as init and a PID that is
    not 1, it will execute telinit and pass all command line arguments unmodified.
    That means init and telinit are mostly equivalent when invoked from normal
    login sessions. See telinit(8) for more information.
```

Система инициализации:

```
man systemd
```

это первый процесс, запускаемый в пользовательском пространстве. У него PID - 1. Мы упоминали про процесс с pid 1, когда говорили о процессах - если убить родительский процесс, у дочернего процесса родителем станет процесс с номером 1. Так вот, после запуска система инициализации должна запустить демоны. Но это не просто список программ, которые нужно запустить - какие-то программы нужно запускать раньше, чем другие, какие-то программы могут конфликтовать и всё такое. А что

делать, если какая-то из программ не запустилась? Нужно же ещё дать возможность пользователям при желании решать, какие программы запускать при включении, какие не запускать. Все эти задачи и стоят перед системой инициализации.

systemd, кроме того, что является системой инициализации, также отвечает за многое другое, например, за управление сервисами, логами, сетью и т.п. Нужно понимать, что роль системы инициализации, как и говорит название - запуск операционной системы, чтобы всё начало работать как надо. На этом полномочия системы инициализации заканчиваются. Но systemd позиционирует себя как системный и сервисный менеджер, он связывает и заменяет многие компоненты операционной системы. Из-за чего, с одной стороны, многое друг с другом интегрировано, легче разрабатывать и администрировать, а с другой - сложнее заменить один компонент на другой и появляется зависимость дистрибутивов от systemd.

Дуг Макилрой, изобретатель каналов Unix и один из основателей традиции Unix, обобщил философию следующим образом:

«Философия Unix гласит:

Пишите [программы](#), которые делают что-то одно и делают это хорошо.

Пишите программы, которые бы работали вместе.

Пишите программы, которые бы поддерживали [текстовые потоки](#), поскольку это универсальный интерфейс».

Очень сложно заменить systemd на что-то другое - придётся многое переделывать. Это противоречит философии Unix-а - пишите программы, которые делают одну вещь и делают её хорошо. В итоге сообщество пользователей разделилось на сторонников и противников использования systemd. Я всё это к тому, что пусть вас не пугает большой функционал systemd, он нацелен на большее количество задач, чем просто инициализировать систему. Поэтому что-то мы рассмотрим сегодня, а что-то останется на потом.

Демонов много, они по разному запускаются, по разному останавливаются, что-то нужно перезапускать в случае ошибки, что-то не нужно, что-то нужно раньше запустить, что-то позже. Т.е. к ним нужен как массовый подход, чтобы удобно было ими управлять, так и индивидуальный. Поэтому демонов обирачают в так называемые сервисы. Сервис - что-то типа инструкции по эксплуатации демона. В сервисе указано, как запускать демон, как его останавливать, с какими другими демонами он связан и т.п. Какие-то сервисы приходят вместе с операционной системой, другие сервисы появляются при установке программ, а что-то вы можете и сами написать. Например, если вы работаете в компании, где программисты написали какую-то программу, а вам нужно обеспечить работу этой программы на серверах, то это, скорее всего, ваша задача - написать сервис, чтобы программа нормально стартовала при запуске операционной системы, нормально завершалась и всё такое.

<p>The following unit types are available:</p> <ol style="list-style-type: none"> 1. Service units, which start and control daemons and the processes they consist of. For details, see systemd.service(5). 2. Socket units, which encapsulate local IPC or network sockets in the system, useful for socket-based activation. For details about socket units, see systemd.socket(5), for details on socket-based activation and other forms of activation, see daemon(7). 3. Target units are useful to group units, or provide well-known synchronization points during boot-up, see systemd.target(5). 4. Device units expose kernel devices in systemd and may be used to implement device-based activation. For details, see systemd.device(5). 5. Mount units control mount points in the file system, for details see systemd.mount(5). 6. Automount units provide automount capabilities, for on-demand mounting of file systems as well as parallelized boot-up. See systemd.automount(5). 7. Timer units are useful for triggering activation of other units based on timers. You may find details in systemd.timer(5). 	<p>You may find details in systemd.timer(5).</p> <ol style="list-style-type: none"> 8. Swap units are very similar to mount units and encapsulate memory swap partitions or files of the operating system. They are described in systemd.swap(5). 9. Path units may be used to activate other services when file system objects change or are modified. See systemd.path(5). 10. Slice units may be used to group units which manage system processes (such as service and scope units) in a hierarchical tree for resource management purposes. See systemd.slice(5). 11. Scope units are similar to service units, but manage foreign processes instead of starting them as well. See systemd.scope(5). <p>Units are named as their configuration files. Some units have special semantics. A detailed list is available in systemd.special(7).</p>
--	---

И так, мы разобрали, что такое сервис. systemd работает с unit-ами, а сервисы - один из типов таких unit-ов. Есть ещё другие типы юнитов - например, юниты устройств, юниты монтирования и т.д. Но нас сейчас интересуют сервисные юниты и таргеты. Таргеты - это тоже юниты, представляющие из себя группу юнитов. target - от слова цель - говорит о конечном результате, достигаемом с помощью группы юнитов. Допустим, чтобы у нас был графический интерфейс, чтобы я мог открыть браузер и зайти на сайт, послушать музыку и т.п. - одного сервиса недостаточно. А вот если я возьму группу сервисов, отвечающих за графический интерфейс, сеть, звук и т.п. - то это будет готовый результат, который я хочу - т.е. это графический target.

```
[user@centos8 ~]$ systemctl get-default  
graphical.target  
[user@centos8 ~]$ systemctl list-dependencies graphical.target  
graphical.target  
└─accounts-daemon.service  
└─gdm.service  
└─rtkit-daemon.service  
└─systemd-update-utmp-runlevel.service  
└─udisks2.service  
└─multi-user.target  
    ├─atd.service  
    ├─auditd.service  
    ├─avahi-daemon.service  
    ├─chronyd.service  
    └─crond.service
```

И если мы посмотрим вывод команды:

```
systemctl get-default
```

то мы как раз увидим, что systemd по умолчанию грузит таргет с названием graphical.target. А с помощью команды:

```
systemctl list-dependencies graphical.target
```

мы как раз увидим, какие юниты нужны для этого таргета. Как видите, графический таргет предполагает использование multi-user таргета, а внутри него огромное количество других юнитов. На серверах, по умолчанию, вместо графического таргета используется multi-user - примерно тоже самое, но нет графического интерфейса, что позволяет сэкономить ресурсы.

```
[user@centos8 ~]$ sudo systemctl set-default multi-user.target  
[sudo] password for user:  
Removed /etc/systemd/system/default.target.  
Created symlink /etc/systemd/system/default.target → /usr/lib/systemd/system/multi-user.target.  
[user@centos8 ~]$ systemctl get-default  
multi-user.target  
[user@centos8 ~]$ sudo systemctl set-default graphical.target  
Removed /etc/systemd/system/default.target.  
Created symlink /etc/systemd/system/default.target → /usr/lib/systemd/system/graphical.target.  
[user@centos8 ~]$
```

И если мы захотим, чтобы у нас тоже было как на серверах, чтобы всё работало, но без графического интерфейса, мы можем поменять таргет по умолчанию, с помощью команды:

```
sudo systemctl set-default multi-user.target
```

Ну и заметим мы это только при запуске. А пока вернём как было:

```
sudo systemctl set-default graphical.target
```

```
[user@centos8 ~]$ systemctl cat graphical.target
# /usr/lib/systemd/system/graphical.target
# SPDX-License-Identifier: LGPL-2.1+
#
# This file is part of systemd.
#
# systemd is free software; you can redistribute it and/or modify it
# under the terms of the GNU Lesser General Public License as published by
# the Free Software Foundation; either version 2.1 of the License, or
# (at your option) any later version.

[Unit]
Description=Graphical Interface
Documentation=man:systemd.special(7)
Requires=multi-user.target
Wants=display-manager.service
Conflicts=rescue.service rescue.target
After=multi-user.target rescue.service rescue.target display-manager.service
AllowIsolate=yes
[user@centos8 ~]$ █
```

Как видите, при выполнении этих команд у нас создаются и удаляются символические ссылки. И тут участвуют 2 директории - /etc/systemd и /usr/lib/systemd. Когда вы устанавливаете какую-то программу или в целом операционную систему, то все файлы юнитов попадают в директорию /usr/lib/systemd. Допустим, посмотрим файл graphical таргета:

```
systemctl cat graphical.target
```

Тут у нас описание юнита, путь к документации. Ну и как видите, тут написано Requires=multi-user.target, т.е. для загрузки графического таргета требуется загрузка multi-user таргета. Также тут написано Wants=display-manager.service. Display manager - это та программа, которая у нас спрашивает логин при входе в систему, ну и она же грузит рабочее окружение. Wants означает, что если этот сервис есть, то нужно его загрузить, а вот если он не грузится, или нету такого сервиса, то ничего страшного, таргет всё равно прогрузится. Если проблема с Requires то таргет перестанет грузиться. Также у нас тут Conflicts - то есть этот таргет не может одновременно работать с этими сервисами и таргетами, и что запуск этого таргета остановит работу указанных здесь сервиса и таргета. After означает порядок, после каких сервисов и таргетов грузится этот юнит.

Ну и AllowIsolate означает, можно ли использовать этот таргет как состояние, к которому можно перейти с помощью команды systemctl isolate. Т.е. таргеты - это группа юнитов, и, в случае с graphical.target или multi-user.target, их можно использовать как состояние загрузки операционной системы - т.е. с графическим интерфейсом или без, но всё же есть все нужное для работы. Но есть, например, таргет сети - network target - опять же, группа юнитов, нужная для работы сети, но на него переключаться бессмысленно, так как это не готовое состояние операционной системы, в котором можно работать, а просто группа юнитов. Для примера, есть rescue.target - это состояние системы, при котором большинство демонов не работает, и это нужно для решения каких-то проблем, допустим, когда система не грузится.

```
File Edit View Search Terminal Help
[user@centos8 ~]$ sudo systemctl isolate rescue.target █
```

```
[ OK ] Started Session c1 of user gdm.
[ OK ] Started User Manager for UID 42.
You are in rescue mode. After logging in, type "journalctl -xb" to view
system logs, "systemctl reboot" to reboot, "systemctl default" or "exit"
to boot into default mode.
Give root password for maintenance
(or press Control-D to continue):

CPU [ 1.3% ] Tasks: 11, 3 thr; 1 running
Mem [ 1.78G ] Load average: 0.29 0.71 0.35
Swap [ 8.75M/2.00G ] Uptime: 00:03:52

PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
 1 root    20   0 511M 11820 8712 S 0.0  0.6 0:01.81 /usr/lib/systemd/systemd --switched
 692 root    20   0 94352 10208 8584 S 0.0  0.5 0:00.57
 727 root    20   0 116M 11212 7556 S 0.0  0.6 0:00.46
 953 rngd    20   0 156M 6536 5828 S 0.0  0.3 0:05.28
 979 rngd    20   0 156M 6536 5828 S 0.0  0.3 0:03.91
1520 dnsmasq  20   0 71888 2368 1928 S 0.0  0.1 0:00.00
1521 root    20   0 71860 428   0 S 0.0  0.0 0:00.00
2648 user    20   0 247M 340   0 S 0.0  0.0 0:00.00
3022 user    20   0 583M 10096 8572 S 0.0  0.5 0:00.02
3075 user    20   0 583M 10096 8572 S 0.0  0.5 0:00.00
3123 user    20   0 583M 10096 8572 S 0.0  0.5 0:00.00
3265 root    20   0 83436 4496 3696 S 0.0  0.2 0:00.00
3266 root    20   0 230M 5416 3584 S 0.0  0.3 0:00.10
3297 root    20   0 234M 4328 3528 R 0.7  0.2 0:00.10
                                         └─ htop
```

Сделаем:

```
sudo systemctl isolate rescue.target
```

и, как видите, теперь у меня система просит root пароль для перехода в режим восстановления. Вводим пароль и смотрим htop - как видите, процессов очень мало.

```
#root@centos8 ~# systemctl isolate graphical.target _
```

Ну и чтобы вернуться обратно, я делаю:

```
systemctl isolate graphical.target
```

```
[user@centos8 ~]$ ls /etc/rc
rc0.d/    rc1.d/    rc2.d/    rc3.d/    rc4.d/    rc5.d/    rc6.d/    rc.d/    rc.local
[user@centos8 ~]$ ls /etc/rc
```

В старых системах инициализации эти состояния, к которым можно было переходить, назывались runlevel-ами. Они были пронумерованы - от 0 до 6. Среди которых 0 это shutdown, 6 это restart, 1 это аналог rescue таргета, 3 это аналог multi-user таргета, 5 - аналог графического таргета, ну и оставшееся это промежуточные состояния. Были директории с соответствующими номерами, в которых лежали скрипты, которые выполнялись при переходе на определённый runlevel. Собственно, это и было основой системы инициализации. Это просто полезно знать, потому что некоторые люди до сих пор говорят о ранлевлах, могут на собеседовании спросить, ну и вы вполне можете наткнуться на другую систему инициализации. Я не буду сравнивать системы инициализации, но, если вам интересно, у Семаева есть ролики про различные системы инициализации, плюс можете почитать по ссылке.

```
[user@centos8 ~]$ cat /usr/lib/systemd/system/gdm.service
[Unit]
Description=GNOME Display Manager

# replaces the getty
Conflicts=getty@tty1.service
After=getty@tty1.service

# replaces plymouth-quit since it quits
Conflicts=plymouth-quit.service
After=plymouth-quit.service

# Needs all the dependencies of the service
# pulled from getty@.service and plymouth-
# (except for plymouth-quit-wait.service)
# plymouth is quit, which we do)
After=rc-local.service plymouth-start

# GDM takes responsibility for stopping
# for any reason, make sure plymouth
OnFailure=plymouth-quit.service

[Service]
ExecStart=/usr/sbin/gdm
ExecStopPost=-/usr/bin/bash -c 'for f in /run/systemd/sessions
& /usr/bin/fgrep -q SERVICE=gdm $f && systemctl terminate-session
done'
KillMode=mixed
Restart=always
IgnoreSIGPIPE=no
BusName=org.gnome.DisplayManager
StandardOutput=syslog
StandardError=inherit
EnvironmentFile=-/etc/locale.conf
ExecReload=/bin/kill -SIGHUP $MAINPID

[Install]
Alias=display-manager.service

```

Мы поговорили о таргетах - группах юнитов. Теперь же давайте посмотрим на какой-нибудь определённый сервисный unit - допустим, gdm - тот самый дисплейный менеджер:

```
cat /usr/lib/systemd/system/gdm.service
```

Тут у нас кроме секции Unit появились ещё две секции - Service и Install. В секции [Service] у нас есть информация о том, как сервис запускает демон - ExecStart, т.е. просто запускает команду /usr/sbin/gdm. А вот ExecStopPost запускает команду после остановки сервиса. Как видите, это какой-то скрипт, и вы можете в нём разобраться. Все опции я разбирать не буду, по большей части у разных сервисов могут быть свои опции, знать всё наизусть не надо, всегда можно обратиться к документации. Что может быть интересно так это секция [Install]. Как видите, тут написан Alias=display-manager.service.

```
[user@centos8 ~]$ sudo systemctl enable gdm
[sudo] password for user:
[sudo] password for user:
Removed /etc/systemd/system/display-manager.service.
[sudo] password for user:
[sudo] password for user:
Created symlink /etc/systemd/system/display-manager.service → /usr/lib/systemd/system/gdm.service
.

[user@centos8 ~]$ sudo systemctl is-enabled gdm
enabled

```

У нас есть две команды - systemctl enable и systemctl disable. Если мы хотим, чтобы какой-то сервис запускался при включении операционной системы, мы запускаем команду systemctl enable и имя сервиса:

```
sudo systemctl enable gdm
```

Как видите, никакого вывода не было, потому что этот сервис уже был включён. Если мы хотим убрать из автозапуска этот сервис, делаем:

```
sudo systemctl disable gdm
```

В выводе написано, что удалён файл /etc/systemd/system/display-manager.service. Если опять сделаем enable:

```
sudo systemctl enable gdm
```

то увидим, что создалась символьическая ссылка `display-manager.service`, которая ведёт на файл `gdm.service`. Т.е. такое вот название символьской ссылки, на основе того, что было написано в Alias в секции `Install`. Ну и можно посмотреть, включён ли сервис, с помощью команды:

```
systemctl is-enabled gdm
```

```
[user@centos8 ~]$ ls /etc/systemd/system
basic.target.wants
bluetooth.target.wants
dbus-org.bluez.service
dbus-org.fedoraproject.FirewallD1.service
dbus-org.freedesktop.Avahi.service
dbus-org.freedesktop.ModemManager1.service
dbus-org.freedesktop.nm-dispatcher.service
dbus-org.freedesktop.resolve1.service
dbus-org.freedesktop.timedate1.service
default.target
display-manager.service
getty.target.wants
graphical.target.wants
multi-user.target.wants
network-online.target.wants
[user@centos8 ~]$ ls -l /etc/systemd/system/syslog.service
lrwxrwxrwx. 1 root root 39 Oct  2  2020 /etc/systemd/system/syslog.service -> /usr/lib/systemd/
system/rsyslog.service
[user@centos8 ~]$
```

Что ещё интересно - символьская ссылка создаётся в директории `/etc/systemd/system`. Если в `/usr/lib/systemd` у нас файлы этих сервисов, то в `/etc/systemd` преимущественно символьские ссылки, означающие, что данный сервис включён. Ну и если у нас какие-то свои сервисы, написанные нами вручную, то правильнее всего считается класть их именно в `/etc/systemd`.

Подводя итоги. Для нормальной работы операционной системы нужны программы, работающие в фоне - демоны. Для запуска демонов при включении компьютера нужна система инициализации, одной из которых является `systemd`. `systemd` много чего умеет, помимо запуска демонов. Для правильной работы с демонами используются сервисы. `systemd` для этого использует service unit-ы и target unit-ы - т.е. группы юнитов. Чтобы сервисы запускались при запуске операционной системы, они должны быть `enabled`, что мы делали с помощью команды `systemctl enable`, ну или наоборот — чтобы убрать из автозапуска — `systemctl disable`. Таким образом операционная система запускает все нужные программы при включении.

Теперь, объединяя эту и предыдущую тему, вы имеете представление, что именно происходит при запуске компьютера и операционной системы.

2.34.2 Практика

Вопросы

1. Для чего нужна система инициализации?
2. Какая система инициализации используется на нашем дистрибутиве?
3. Что такое таргеты?
4. Какой таргет используется в нашей системе по умолчанию? Как это посмотреть?
5. Как посмотреть, какие сервисы входят в этот таргет?
6. Что такое демоны?
7. Что такое сервисы?
8. Как узнать, какая команда выполняется при запуске сервиса?

9. Как поменять текущий таргет?
10. Как поменять дефолтный таргет?
11. Как добавить или убрать сервис из автозагрузки?
12. Как посмотреть, включён ли сервис?

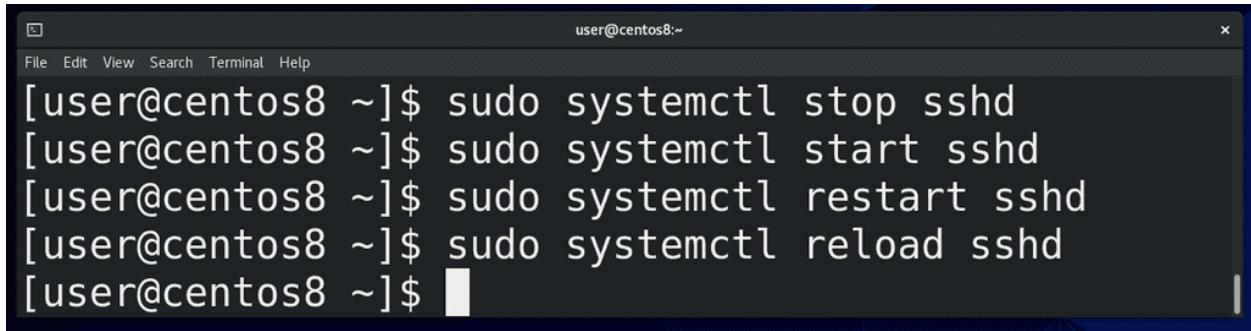
Задания

1. Зайдите в rescue.target, посмотрите содержимое fstab, после чего вернитесь в graphical.target.
2. Сделайте multi-user.target режимом по умолчанию.

2.35 35. Системный менеджер systemd

2.35.1 35. Системный менеджер systemd

Как мы разобрались в прошлый раз, при запуске компьютера система инициализации запускает всякие сервисы, которые работают в фоне. По большей части работа системного администратора заключается в том, чтобы следить за состоянием операционной системы, устанавливать и настраивать всякие сервисы. На серверах крутятся всякие сервисы, допустим, тот же веб сервер, где крутятся сайты.



```
[user@centos8 ~]$ sudo systemctl stop sshd
[user@centos8 ~]$ sudo systemctl start sshd
[user@centos8 ~]$ sudo systemctl restart sshd
[user@centos8 ~]$ sudo systemctl reload sshd
[user@centos8 ~]$
```

systemd, как сервисный менеджер, позволяет управлять этими сервисами. Для примера возьмём сервис под названием sshd. Вы часто будете встречать сервисы с буквой d в конце - она означает daemon, то есть демон программы ssh. Базовые операции - это остановить, запустить и перезапустить сервис - то есть:

```
sudo systemctl stop sshd
sudo systemctl start sshd
sudo systemctl restart sshd
```

соответственно. Как понятно из названия, restart останавливает и запускает сервис заново. Обычно перезапускают сервис, когда изменяются какие-то настройки демона. Нужно его заново запустить, чтобы он перечитал свои настройки. Для каких-то сервисов это не проблема, но бывают важные сервисы, в которых даже секундная остановка вызывает проблемы. Для таких сервисов существует возможность перезапустить основной демон, при этом не затронув текущие процессы. Для этого используется опция reload:

```
sudo systemctl reload sshd
```

```
[user@centos8 ~]$ sudo systemctl mask sshd
Created symlink /etc/systemd/system/sshd.service → /dev/null.
[user@centos8 ~]$ ls -l /etc/systemd/system/sshd.service
lrwxrwxrwx. 1 root root 9 Nov 21 14:48 /etc/systemd/system/sshd.service -> /dev/null
[user@centos8 ~]$ sudo systemctl start sshd
Failed to start sshd.service: Unit sshd.service is masked.
[user@centos8 ~]$ sudo systemctl unmask sshd
Removed /etc/systemd/system/sshd.service.
[user@centos8 ~]$ sudo systemctl start sshd
[user@centos8 ~]$ █
```

Как мы помним

```
systemctl enable sshd
```

даёт возможность запускать сервис автоматом при включении компьютера. Мы можем сделать

```
systemctl disable sshd
```

если не хотим, чтобы сервис стартовал при включении. И тем не менее, другие программы и пользователи могут запустить этот сервис при необходимости. Если же мы хотим, чтобы этот сервис нельзя было запустить, мы можем его замаскировать, с помощью команды:

```
systemctl mask sshd
```

Как видите, создаётся символическая ссылка, ведущая на `/dev/null` - то есть, при попытке запустить сервис ничего не произойдёт. Но это касается только сервиса, если мы вручную запустим программу, она будет работать. Ну и с помощью `unmask`:

```
sudo systemctl unmask sshd
```

мы можем сервис размаскировать.

```
[user@centos8 ~]$ sudo systemctl status sshd
● sshd.service - OpenSSH server daemon
  Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
  Active: active (running) since Sat 2020-11-21 14:59:29 +04; 1s ago
    Docs: man:sshd(8)
          man:sshd_config(5)
  Main PID: 10328 (sshd)
     Tasks: 1 (limit: 11474)
    Memory: 1.2M
   CGroup: /system.slice/sshd.service
           └─10328 /usr/sbin/sshd -D -oCiphers=aes256-gcm@openssh.com, chacha20-poly1305@ope█

Nov 21 14:59:29 centos8 systemd[1]: Starting OpenSSH server daemon...
Nov 21 14:59:29 centos8 sshd[10328]: Server listening on 0.0.0.0 port 22.
Nov 21 14:59:29 centos8 sshd[10328]: Server listening on :: port 22.
Nov 21 14:59:29 centos8 systemd[1]: Started OpenSSH server daemon.
lines 1-15/15 (END)
```

Также мы можем посмотреть статус этого сервиса с помощью команды `status`:

```
systemctl status sshd
```

В строчке `Loaded` мы видим путь до основного файла сервиса. Дальше 2 раза видим слово `enabled`. Первый `enabled` говорит о том, что сервис включён, т.е., он будет запускаться при включении ком-

пьютера. Перед вторым enabled написано vendor preset. То есть, имеется в виду, что этот сервис был включён по умолчанию ещё при установке программы или операционной системы.

```
[user@centos8 ~]$ systemctl status sshd
● sshd.service - OpenSSH server daemon
  Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
  Active: inactive (dead) since Sat 2020-11-21 15:16:37 +04; 8s ago
    Docs: man:sshd(8)
          man:sshd_config(5)
   Process: 12081 ExecStart=/usr/sbin/sshd -D $OPTIONS $CRYPTO_POLICY
 Main PID: 12081 (code=exited, status=0/SUCCESS)
```

В строчке Active мы видим, что сервис работает. Если остановить сервис, здесь будет писаться inactive.

```
● vboxadd.service
  Loaded: loaded (/opt/VBoxGuestAdditions-6.1.4/init/vbox
  Active: active (exited) since Sat 2020-11-21 14:11:47 +
Main PID: 1009 (code=exited, status=0/SUCCESS)
    Tasks: 2 (limit: 11474)
   Memory: 15.9M
     CGroup: /system.slice/vboxadd.service
             └─2168 VBoxClient --vmsvga
                 ├─2173 VBoxClient --vmsvga
```

Иногда, вместо active (running), можно увидеть active (exited):

```
systemctl status vboxadd
```

Собственно, ничего страшного в этом нет. Если это какой-то демон, который постоянно работает на фоне, systemd может отслеживать состояние процесса и говорить, что он работает, то есть running. Но иногда вместо полноценного демона за сервисом стоит какой-то скрипт, который запускается, выполняет свою работу, запускает какие-то программы и завершается. И вроде скрипт больше не работает, но свою работу он сделал. Поэтому так и выходит - вроде это и не демон, отслеживать нечего, но при этом то что нужно работает.

```
[user@centos8 ~]$ sudo systemctl status sshd
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2020-11-21 14:59:29 +04; 1s ago
     Docs: man:sshd(8)
           man:sshd_config(5)
   Main PID: 10328 (sshd)
      Tasks: 1 (limit: 11474)
     Memory: 1.2M
        CGroup: /system.slice/sshd.service
                  └─10328 /usr/sbin/sshd -D -oCiphers=aes256-gcm@openssh.com,chacha20-poly1305@ope>

Nov 21 14:59:29 centos8 systemd[1]: Starting OpenSSH server daemon...
Nov 21 14:59:29 centos8 sshd[10328]: Server listening on 0.0.0.0 port 22.
Nov 21 14:59:29 centos8 sshd[10328]: Server listening on :: port 22.
Nov 21 14:59:29 centos8 systemd[1]: Started OpenSSH server daemon.
lines 1-15/15 (END)
```

Вернёмся к нашему sshd. Дальше у нас ссылки на документацию и PID основного процесса, который запустился при старте сервиса. Дальше Tasks - общее количество процессов - включая основной и его дочерние. Memory - сколько оперативки использует этот сервис. Дальше CGroup - контрольная группа, в которую входят процессы этого сервиса. С помощью контрольных групп на уровне ядра можно изолировать группу процессов и выделять им ограниченные ресурсы - сколько-то оперативки, сколько-то процессора и т.п. Эдакая виртуализация на уровне самой операционной системы. Чуть ниже в статусе мы видим логи. К ним мы ещё вернёмся.

Я не хочу ударяться в объяснение каждой из команд systemctl, типа systemctl cat sshd, show, edit и всё такое. Не все команды используются часто, многие специфичны. На что-то мы наткнёмся и разберём, что-то вы по работе разберёте, а с чем-то и вовсе не столкнётесь.

Unit	Status	Main PID	Description
run-user-42.mount	loaded active mounted	/run/user/42	
sys-fs-fuse-connections.mount	loaded active mounted		FUSE Control File System
sys-kernel-config.mount	loaded active mounted		Kernel Configuration File
sys-kernel-debug.mount	loaded active mounted		Kernel Debug File System
var-lib-nfs-rpc_pipefs.mount	loaded active mounted		RPC Pipe File System
cups.path	loaded active running		CUPS Scheduler
systemd-ask-password-plymouth.path	loaded active waiting		Forward Password Requests
systemd-ask-password-wall.path	loaded active waiting		Forward Password Requests
init.scope	loaded active running		System and Service Manager
session-2.scope	loaded active running		Session 2 of user user

```
lines 38-47
```

Как мы упомянули в прошлый раз, systemd отвечает не только за сервисы. Если запустить команду:

```
systemctl --all
```

мы увидим информацию о всех unitах - а тут и сервисы, и таргеты, и устройства, и mountы, и всякое другое. С сервисами и таргетами мы разобрались, немного поговорим про другие юниты.

```

SysFSPath=/sys/devices/pci0000:00/0000:00:01.1/ata2/host1/target1:0:0/1:0
Id=dev-cdrom.device
Names=dev-cdrom.device
Following=sys-devices-pci0000:00-0000:00:01.1-ata2-host1-target1:0:0-1:0
Description=VBOX_CD-ROM VBox_GAs_6.1.4
LoadState=loaded
ActiveState=active
SubState=plugged
StateChangeTimestamp=Sat 2020-11-21 14:11:35 +04
StateChangeTimestampMonotonic=5712119
InactiveExitTimestamp=Sat 2020-11-21 14:11:35 +04
lines 1-11

```

Помните, когда мы говорили про ядро, мы упомянули udev? Программа, которая делает какие-то действия при виде устройств - даёт названия устройствам, создаёт ссылки в директории /dev, может передать ядру какие-то параметры для устройства, запустить какие-то программы и т.п. Так вот, udev - тоже демон, но, к тому же, он является частью systemd, генерирует юнит файлы для устройств, называемые device unit-ами. Это позволяет сделать связь между устройствами и другими сервисами.

```

Where=/mydata
What=/dev/mapper/myraidvg-myraidlv
Options=rw,relatime,seclabel
Type=ext4
TimeoutUSec=1min 30s
ControlPID=0
DirectoryMode=0755
SloppyOptions=no

```

Или, допустим, mount unit-ы. Для примера возьмём mydata.mount:

```
systemctl show mydata.unit
```

При включении компьютера кто-то же должен примонтировать все файловые системы, указанные в fstab? Так вот, systemd генерирует специальные unit-ы на основе записей fstab и монтирует их. При этом он смотрит зависимости, скажем, отличает локальные файловые системы от сетевых и на основе этого формирует зависимости при создании юнитов. Есть ещё swap unit-ы - примерно тоже самое, но для swap разделов.

Про остальные unit-ы мы поговорим, когда будем разбирать темы, связанные с ними. А сегодня мы разобрали как работать с сервисами, как смотреть информацию о них, и в целом стали лучше понимать, чем же занимается systemd.

2.35.2 Практика

Вопросы

1. Какие операции можно производить с сервисами?
2. Как посмотреть информацию о сервисе?

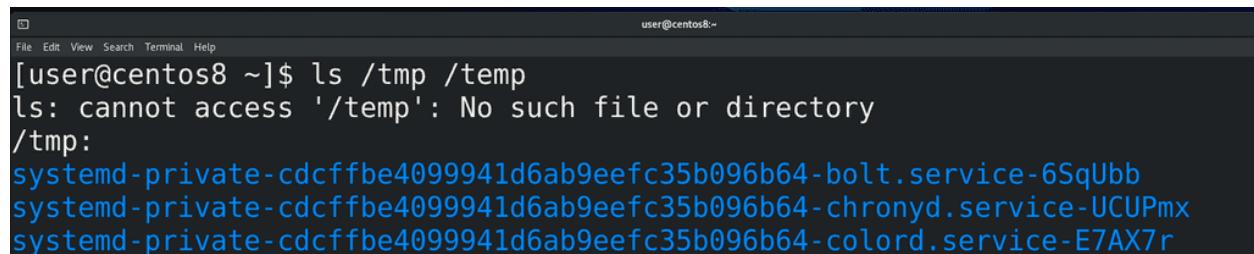
3. Чем отличается restart от reload?
4. Как запретить запуск сервиса и чем это отличается от «отключения» сервиса?
5. Как понять, что с сервисом есть проблемы?
6. Почему некоторые сервисы в статусе пишут active(running), а другие active(exited)?
7. Как посмотреть информацию не только о сервисах, но и о других юнитах systemd?

Задания

1. Отключите сервис sshd, перезапустите систему и посмотрите статус этого сервиса.
2. Запустите сервис sshd и посмотрите его статус.
3. Запретите сервису sshd запускаться.
4. Верните sshd право запускаться и вновь добавьте его в автозапуск. Перезапустите систему и убедитесь, что всё работает.

2.36 36. Логирование

2.36.1 36. Логирование



```
[user@centos8 ~]$ ls /tmp /temp
ls: cannot access '/temp': No such file or directory
/tmp:
systemd-private-cdcffbe4099941d6ab9eefc35b096b64-bolt.service-6SqUbb
systemd-private-cdcffbe4099941d6ab9eefc35b096b64-chronyd.service-UCUPmx
systemd-private-cdcffbe4099941d6ab9eefc35b096b64-colord.service-E7AX7r
```

Когда вы запускаете какую-то программу в текстовом интерфейсе, как правило, вы видите вывод этой команды, будь то нужная информация или какие-то ошибки:

```
ls /tmp /temp
```

Но всякие демоны запускаются сервисным менеджером и работают на фоне, из-за чего их вывод не виден пользователю. Как я говорил, настройка и обеспечение работы демонов - одна из главных задач администратора. А значит очень важно видеть какие ошибки выдаёт программа, да и информационные сообщения также важны. Скажем, количество просмотров на youtube - это тоже сообщения, которые генерируют демоны. И весь этот вывод, будь то ошибки или полезная информация, называется логами.

У большинства демонов есть свои настройки логов и они, в некотором роде, уникальны. Возьмём, для примера, веб сервер. Это программа, которая отвечает за работу сайтов. И на одном веб сервере можно держать несколько сайтов. Так вот, настройки для этого демона позволяют хранить информацию о подключениях на разные сайты в разных файлах. Но это относится к веб серверу, у других сервисов могут быть свои особенности. Поэтому, если вам важны логи определённого сервиса, нужно искать информацию именно про этот сервис. Но логи большинства программ вы даже не просмотрите. Они не так важны и нужны довольно редко, допустим, когда возникла проблема.

```
[user@centos8 ~]$ ls -l /dev/log
lrwxrwxrwx. 1 root root 28 Feb 20 18:58 /dev/log -> /run/systemd/journal/dev-log
[user@centos8 ~]$ ls -l /run/systemd/journal/dev-log
srw-rw-rw-. 1 root root 0 Feb 20 18:58 /run/systemd/journal/dev-log
[user@centos8 ~]$ stat /run/systemd/journal/dev-log
  File: /run/systemd/journal/dev-log
  Size: 0          Blocks: 0          IO Block: 4096   socket
Device: 17h/23d Inode: 11370      Links: 1
Access: (0666/srw-rw-rw-)  Uid: (    0/    root)  Gid: (    0/    root)
Context: system_u:object_r:devlog_t:s0
Access: 2021-02-20 19:01:01.169176727 +0400
Modify: 2021-02-20 18:58:42.723000025 +0400
Change: 2021-02-20 18:58:42.723000025 +0400
 Birth: -
[user@centos8 ~]$
```

И так, программы посыпают свои логи в stdout, stderr и в какие-то файлы, указанные в настройках самих программ. Кроме этого, для логов существует стандарт отправки сообщений о событиях, называемый syslog. Операционная система даёт возможность программам посыпать свои логи с помощью функции syslog в специальный файл /dev/log:

```
ls -l /dev/log
ls -l /run/systemd/journal/dev-log
```

Обратите внимание на первый символ - s. Мы с вами разбирали некоторые типы файлов, это - ещё один тип файла, называемый сокетом. Сокеты используются для взаимодействия между процессами. Таким образом, когда какая-то программа отправляет логи в /dev/log, они передаются демону, стоящему за этим файлом. Этот сокет файл dev-log принадлежит процессу демона journald:

```
man systemd-journald
```

journald - это программа, отвечающая за логи. Кроме syslog сообщений, journald также перехватывает stdout и stderr сервисов, запущенных в systemd.

До journald основным демоном для syslog был rsyslog. Собственно, он стоял за /dev/log и собирал все логи, а дальше, на основе своих настроек, решал, что делать с этими логами - писать в файлы, выводить на экран, посыпать по почте и всё такое. На самом деле rsyslog и сейчас работает, вместе с journald:

```
systemctl status rsyslog
systemctl status systemd-journald
```

Только теперь логи собирает journald, а потом делится ими с rsyslog. Причём, где-то может быть наоборот - логи получает syslogd, а потом отправляет на journald. А где-то может не быть rsyslog-а или journald - это зависит от дистрибутива.

```
[user@centos8 ~]$ ls /var/log/
anaconda          gdm              sa
audit              glusterfs        samba
boot.log          hawkey.log      secure
boot.log-20201120 hawkey.log-20201126 secure-20201126
boot.log-20201121 hawkey.log-20201129 secure-20201129
boot.log-20201126 hawkey.log-20201207 secure-20201207
boot.log-20201129 hawkey.log-20210115 secure-20210115
boot.log-20201205 httpd           speech Dispatcher
```

Но зачем два демона, отвечающих за логи? Всё дело в подходе к хранению логов. rsyslog, при получении логов, смотрит параметры лога и на основе этого записывает их в различные файлы в директории /var/log:

`ls /var/log`

```
[user@centos8 ~]$ sudo tail /var/log/messages
[sudo] password for user:
Feb 20 12:20:04 centos8 systemd[1]: Starting system activity accounting tool...
Feb 20 12:20:04 centos8 systemd[1]: Started system activity accounting tool.
Feb 20 12:20:04 centos8 systemd[2496]: Starting Mark boot as successful...
Feb 20 12:20:04 centos8 systemd[2496]: Started Mark boot as successful.
Feb 20 12:21:25 centos8 org.gnome.Shell.desktop[2605]: Window manager warning: last_user_time (259259) is greater than comparison timestamp (259257). This most likely represents a buggy client sending inaccurate timestamps in messages such as _NET_ACTIVE_WINDOW. Trying to work around...
Feb 20 12:21:25 centos8 org.gnome.Shell.desktop[2605]: Window manager warning: W1 appears to be one of the offending windows with a timestamp of 259259. Working around...
Feb 20 12:21:26 centos8 dbus-daemon[1024]: [system] Activating via systemd: service name='net.reactivated.Fprint' unit='fprintd.service' requested by ':1.430' (uid=0 pid=3995 comm="sudo tail /var/log/messages" label="unconfined u:unconfined r:unconfined t:s0-s0:c0.c1023")
```

К примеру, почти все логи от rsyslog записываются в файл /var/log/messages:

`sudo tail /var/log/messages`

Т.е. это просто текстовые файлы, которые можно прочесть и обработать с помощью различных утилит и скриптов. С одной стороны это плюс - с текстовыми файлами работать проще. Но обычно логов много и попытка найти что-то конкретное в большом текстовом файле может оказаться непростой задачей. И когда речь идёт о большом количестве данных, обычно их записывают в базы данных. Там они хранятся в структурированном виде и найти что-то определённое становится гораздо проще. Такой подход предлагает journald. При этом получается более гибко работать с логами, но логи перестают быть текстовыми и появляется зависимость от определённой утилиты, которая и работает с логами. И если на домашних компьютерах это не так критично, то в компаниях могут быть ситуации, когда какие-то задачи завязаны на скриптах, которые ориентируются на текстовые логи - и просто отказаться от текстовых логов будет слишком болезненно.

Ещё важный момент - в компаниях зачастую стоят централизованные syslog сервера, на которые посылаются логи со всех серверов и различного оборудования. И для посылки логов syslog на централизованный сервер по сети нужен демон, который это поддерживает. У journald с этим неопределённость - вроде они что-то разрабатывали для этого, можно даже скачать и скомпилировать, но какого-то функционала нет, как и упоминаний в официальной документации от Red Hat. А rsyslog это поддерживает, что является ещё одной причиной не отказываться от него. Хотя, справедливо ради, journald может посыпать логи на другой сервер journald, что также можно использовать для центрального хранения логов, но только от линуксов с journald. Поэтому и используются оба демона - один может отправлять по сети и записывать в текстовые файлы(rsyslog), а второй более удобный и гибкий(journald).

```
GNU nano 2.9.8                               /etc/rsyslog.conf

rsyslog configuration file

# For more information see /usr/share/doc/rsyslog-*/*rsyslog_conf.html
# or latest version online at http://www.rsyslog.com/doc/rsyslog_conf.html
# If you experience problems, see http://www.rsyslog.com/doc/troubleshoot.html

##### MODULES #####
module(load="imuxsock" # provides support for local system logging (e.g. via logger co$ SysSock.Use="off") # Turn off message reception via local log socket; # local messages are retrieved through imjournal now.
module(load="imjournal" # provides access to the systemd journal StateFile="imjournal.state") # File to store the position in the journal
#module(load="imklog") # reads kernel messages (the same are read from journald)
#module(load="immark") # provides --MARK-- message capability
```

Начнём с rsyslog. Его основной файл настроек - rsyslog.conf:

```
sudo nano /etc/rsyslog.conf
```

По началу мы видим использование двух модулей - imuxsock и imjournal - в которых можно настроить, будет ли rsyslog создавать сокет файл, куда будут посыпаться логи из программ или перенаправляться с journald, либо будет ли сам rsyslog подтягивать файлы из журналов journald.

```
# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none          /var/log/messages

# The authpriv file has restricted access.
authpriv.*                                         /var/log/secure

# Log all the mail messages in one place.
mail.*                                             -/var/log/maillog

# Log cron stuff
cron.*                                            /var/log/cron
```

Снизу у нас правила - какие логи в какие файлы записывать. О чём здесь речь?

Facility code	Keyword	Description
0	kern	Kernel messages
1	user	User-level messages
2	mail	Mail system
3	daemon	System daemons
4	auth	Security/authentication messages
5	syslog	Messages generated internally by syslogd
6	lpr	Line printer subsystem
7	news	Network news subsystem
8	uucp	UUCP subsystem
9	cron	Clock daemon
10	authpriv	Security/authentication messages
11	ftp	FTP daemon
12	ntp	NTP subsystem
13	security	Log audit
14	console	Log alert
15	solaris-cron	Scheduling daemon
16-23	local0 – local7	Locally used facilities

У стандарта syslog есть объекты логирования. Что-то вроде меток, по которым можно ориентироваться в логах. Например, kern - по нему можно понять, что логи относятся к ядру. Либо mail - логи, связанные с почтой. Также есть 8 объектов - local0 - local7 - это кастомные, которые можно использовать в своих целях, например, для своих скриптов или программ, которые не подходят под перечисленное.

Value	Severity	Keyword	Deprecated keywords	Description	Condition
0	Emergency	emerg	panic [7]	System is unusable	A panic condition.[8]
1	Alert	alert		Action must be taken immediately	A condition that should be corrected immediately, such as a corrupted system database.[8]
2	Critical	crit		Critical conditions	Hard device errors.[8]
3	Error	err	error [7]	Error conditions	
4	Warning	warning	warn [7]	Warning conditions	
5	Notice	notice		Normal but significant conditions	Conditions that are not error conditions, but that may require special handling.[8]
6	Informational	info		Informational messages	
7	Debug	debug		Debug-level messages	Messages that contain information normally of use only when debugging a program.[8]

Кроме объектов есть уровни серьёзности. По ним можно понимать, насколько важное сообщение в логе, что позволяет отсортировывать обычные сообщения от ошибок, всяких проблем и сбоев.

```
# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none          /var/log/messages

# The authpriv file has restricted access.
authpriv.*                                         /var/log/secure

# Log all the mail messages in one place.
mail.*                                              -/var/log/maillog

# Log cron stuff
cron.*                                             /var/log/cron
```

Теперь, взглянув в rsyslog.conf, можно понять и даже самим написать правила. Например, authpriv.*: authpriv - это логи, связанные с безопасностью и аутентификацией; * - это логи любого уровня серьёзности - будут записываться в /var/log/secure. Или, например, первая строчка - *.info; mail.none; authpriv.none; cron.none - логи любых объектов уровня серьёзности info, кроме тех, что связаны с почтой, аутентификацией или планировщиком задач cron - будут записываться в /var/log/messages. Обратите внимание на дефис перед -/var/log/maillog - он говорит о том, чтобы не записывать на диск изменения при каждом новом логе.

```
[user@centos8 ~]$ sync
[user@centos8 ~]$
```

Обычно когда вы изменяете какой-то файл, изменения на файловой системе происходят в оперативке, а потом синхронизируются с диском. А так как жёсткий диск работает медленнее, чем оперативка, то убеждаться в сохранении каждого лога того же почтового демона будет создавать проблему с очередью, так называемый bottleneck. Для этого и нужен дефис перед файлом - так новые логи будут копиться какое-то время в оперативке, прежде чем разом записаться на диск. Кстати, процесс синхронизации изменений в оперативке и с диском можно запустить и вручную, с помощью команды sync. Это касается не только логов, но и любых изменений.

```
# Everybody gets emergency messages
*. emerg :omusrmsg:*

# Save news errors of level crit and higher in a special file.
uucp,news.crit /var/log/spooler

# Save boot messages also to boot.log
local7.* /var/log/boot.log
```

Ещё одна примечательная запись - *.emerg - как видите, здесь указан не файл. Вместо него модуль rsyslog-a, который будет выводить сообщение залогиненным пользователям в виртуальную консоль.

```
[user@centos8 ~]$ sudo systemctl restart rsyslog.service
[user@centos8 ~]$ sudo systemctl status rsyslog.service
● rsyslog.service - System Logging Service
  Loaded: loaded (/usr/lib/systemd/system/rsyslog.service; enabled; vendor pre>
  Active: active (running) since Sat 2021-02-20 15:08:49 +04; 5s ago
    Docs: man:rsyslogd(8)
          http://www.rsyslog.com/doc/
   Main PID: 3715 (rsyslogd)
     Tasks: 3 (limit: 11474)
    Memory: 1.5M
   CGroup: /system.slice/rsyslog.service
           └─3715 /usr/sbin/rsyslogd -n
```

Тут много различных опций и возможностей, всё зависит от ваших задач. Не забудьте - после изменения настроек следует рестартнуть сервис rsyslog:

```
sudo systemctl restart rsyslog
```

```
GNU nano 2.9.8          /etc/systemd/journald.conf

#
# See journald.conf(5) for details.

[Journal]
#Storage=auto
#Compress=yes
#Seal=yes
#SplitMode=uid
#SyncIntervalSec=5m
#RateLimitIntervalSec=30s
#RateLimitBurst=10000
#SystemMaxUse=
```

Теперь перейдём к демону journald:

```
sudo nano /etc/systemd/journald.conf
```

Тут довольно много опций и все разбирать смысла нет, но при желании можно найти информацию о каждом параметре с помощью man journald.conf. Но кое-какие ключи мы всё же разберём. Например, Storage.

```
[Journal]
#Storage=auto

user@centos8:~
```

```
[user@centos8 ~]$ ls /var/log/journal
ls: cannot access '/var/log/journal': No such file or directory
[user@centos8 ~]$ sudo mkdir /var/log/journal
[user@centos8 ~]$ █
```

Если значение auto, то journald проверяет наличие директории /var/log/journal. Если она есть - то хранит там логи. Если же директории нет, то логи хранятся в оперативке, соответственно, при перезагрузке все логи стираются. Если мы хотим, чтобы логи сохранялись, то достаточно просто создать директорию:

```
sudo mkdir /var/log/journal
```

```

GNU nano 2.9.8          /etc/systemd/journald.conf      Modified

#
# See journald.conf(5) for details.

[Journal]
#Storage=auto
#Compress=yes
#Seal=yes
#SplitMode=uid

```

[user@centos8 ~]\$ cd /var/log/journal/91222dde95634287a2336070235dc625/
[user@centos8 91222dde95634287a2336070235dc625]\$ ls -l
total 16384
-rw-r----- 1 root root 8388608 Feb 20 15:30 system.journal
-rw-r-----+ 1 root root 8388608 Feb 20 15:29 user-1000.journal
[user@centos8 91222dde95634287a2336070235dc625]\$ getfacl user-1000.journal
file: user-1000.journal
owner: root
group: root
user::rw-
user:user:r--
group::r--
mask::r--

Ещё одна примечательная опция - SplitMode. При значении uid для каждого обычного пользователя создаётся отдельная база с его логами и автоматом выдаются нужные права, чтобы пользователь мог просматривать свои логи.

```

#SystemMaxUse=
#SystemKeepFree=
#SystemMaxFileSize=
#SystemMaxFiles=100
#RuntimeMaxUse=
#RuntimeKeepFree=
#RuntimeMaxFileSize=
#RuntimeMaxFiles=100
#MaxRetentionSec=
#MaxFileSec=1month

```

Также здесь есть настройки того, сколько хранить логи, сколько разрешено логам заниматься пространства файловой системы и т.п. Например, SystemMaxUse даёт ограничение на используемое логами

пространство. По умолчанию это 10% от файловой системы, а если файловая система большая, то не более 4 Гигабайт. Когда файл логов достигнет максимального значения, старые логи будут удаляться автоматически. SystemMaxFileSize позволяет по достижению какого-то объема выносить старые логи в другую базу, которую вы можете потом перенести куда-нибудь в архивные логи. MaxRetentionSec - сколько времени хранить логи. Скажем, размер логов не достиг максимального значения, но при этом вам не нужны логи старее месяца. Опять же, в конфиге много различных опций, зачастую названия говорят за себя, а если что не понятно - смотрите в мане. Ну и после каких-либо изменений - sudo systemctl restart systemd-journald.

Раз уж мы заговорили о размерах файлов и времени их хранения, как с этим обстоят дела у rsyslog? Как правило, для rsyslog-а этим занимается программа logrotate. Т.е. rsyslog пишет логи, а logrotate удаляет старые логи, следит за размером и всё такое. Собственно это и называется ротация логов.

```
GNU nano 2.9.8                               /etc/logrotate.conf

# see "man logrotate" for details
# rotate log files weekly
weekly

# keep 4 weeks worth of backlogs
rotate 4

# create new (empty) log files after rotating old ones
create

# use date as a suffix of the rotated file
dateext
```

У logrotate есть основной файл настроек - logrotate.conf:

```
sudo nano /etc/logrotate.conf
```

Здесь обычно хранятся значения по умолчанию. Например, weekly - по умолчанию logrotate будет ротировать логи раз в неделю. rotate 4 - ротаций будет 4.

[user@centos8 ~]\$	ls /var/log/	
anaconda	gdm	sa
audit	glusterfs	samba
boot.log	hawkey.log	secure
boot.log-20201120	hawkey.log-20201126	secure-20201126
boot.log-20201121	hawkey.log-20201129	secure-20201129
boot.log-20201126	hawkey.log-20201207	secure-20201207
boot.log-20201129	hawkey.log-20210115	secure-20210115
boot.log-20201205	httpd	speech-dispatcher

Для примера посмотрим содержимое /var/log, например /var/log/secure. Лог недельной давности logrotate переименовал в другое название, а вместо него оставил чистый secure. И так он делает 4 раза. Когда старых логов уже будет 4 и понадобится опять ротировать, он удалит самый старый лог, чтобы у нас старых логов было на месяц. Собственно, rotate 4 об этом и говорит. А опция dateext, как

вы видите, добавляет к старым логам дату их ротации.

```
[user@centos8 log]$ sudo nano /etc/logrotate.d/
bootlog          frr          psacct          wpa_supplicant
bttmp            iscsiuiolog    samba           wtmp
chrony           libvirtd       sssd
cups              libvirtd.qemu    syslog
dnf               numad         up2date
[user@centos8 log]$ sudo nano /etc/logrotate.d/syslog
```

Но выше были лишь дефолтные настройки, а к каким логам они применяются и кастомные настройки этих лог файлов указываются в директории `/etc/logrotate.d`. Для примера посмотрим `/etc/logrotate.d/syslog`:

```
sudo nano /etc/logrotate.d/syslog
```

```
GNU nano 2.9.8                               /etc/logrotate.d/syslog

/var/log/cron
/var/log/maillog
/var/log/messages
/var/log/secure
/var/log/spooler
{
    missingok
    sharedscripts
    postrotate
        /usr/bin/systemctl kill -s HUP rsyslog.service >/dev/null 2>&1 || true
    endscript
}
```

И здесь мы видим, что для такого-то списка файлов применяются такие-то опции. В фигурных скобках мы можем поменять дефолтные настройки на свои, например, выставить количество ротаций, периодичность и т.п. Тут также можно задать какие-то скрипты, которые будут выполняться перед или после ротации, так как некоторые сервисы могут этого требовать, чтобы увидеть новый файл логов.

```
[user@centos8 ~]$ sudo journalctl
-- Logs begin at Sat 2021-02-20 15:00:53 +04, end at Sat 2021-02-20 16:03:16 +04
Feb 20 15:00:53 centos8 kernel: Linux version 4.18.0-147.8.1.el8_1.x86_64 (mock)
Feb 20 15:00:53 centos8 kernel: Command line: BOOT_IMAGE=(hd0,msdos1)/vmlinuz-4>
Feb 20 15:00:53 centos8 kernel: x86/fpu: Supporting XSAVE feature 0x001: 'x87 f>
Feb 20 15:00:53 centos8 kernel: x86/fpu: Supporting XSAVE feature 0x002: 'SSE r>
Feb 20 15:00:53 centos8 kernel: x86/fpu: Supporting XSAVE feature 0x004: 'AVX r>
Feb 20 15:00:53 centos8 kernel: x86/fpu: xstate_offset[2]: 576, xstate_sizes[2]>
Feb 20 15:00:53 centos8 kernel: x86/fpu: Enabled xstate features 0x7, context s>
Feb 20 15:00:53 centos8 kernel: BIOS-provided physical RAM map:
```

Теперь касательно просмотра логов. Для этого используется утилита:

```
journalctl
```

Обычный пользователь может смотреть только свои логи, для системных логов понадобится sudo. Впрочем можно настроить права на файлы логов, чтобы пользователи какой-то группы, например, `systemd-journal`, могли просматривать все логи.

```
[user@centos8 ~]$ sudo journalctl -e
Feb 20 16:04:37 centos8 kernel: VBGL_IOCTL_ACQUIRE_GUEST_CAPABILITIES failed rc>
Feb 20 16:04:38 centos8 sudo[6784]: pam_unix(sudo:session): session closed for >
Feb 20 16:04:39 centos8 sudo[6798]:      user : TTY=pts/0 ; PWD=/home/user ; USE>
```

Просто запуск journalctl выведет вам огромное количество логов, что редко нужно. Чаще всего важнее посмотреть логи за последнее время:

```
sudo journalctl -e
```

И так, что мы видим обычно в логах? В первом столбце - дата и время лога, во втором - имя компьютера, в третьем - программа, которая отправила этот лог, а также её pid, ну а дальше само сообщение лога.

```
[user@centos8 ~]$ sudo journalctl -eu systemd-udevd
Feb 20 15:00:54 centos8 systemd[1]: Started udev Kernel Device Manager.
Feb 20 15:00:54 centos8 systemd-udevd[423]: link_config: autonegotiation is
Feb 20 15:00:54 centos8 systemd-udevd[429]: link_config: autonegotiation is
Feb 20 15:00:55 centos8 systemd-udevd[429]: Process '/sbin/modprobe -bv sg'
Feb 20 15:00:55 centos8 systemd-udevd[431]: Process '/sbin/modprobe -bv sg'
Feb 20 15:00:55 centos8 systemd-udevd[430]: Process '/sbin/modprobe -bv sg'
```

Ещё чаще хочется посмотреть логи по какому-то сервису. Причём не все логи, а только последние:

```
sudo journalctl -eu systemd-udevd
```

Здесь под -u имеется ввиду unit systemd.

```
[user@centos8 ~]$ sudo journalctl -fu sshd
-- Logs begin at Sat 2021-02-20 15:00:53 +04. --
Feb 20 15:01:01 centos8 systemd[1]: Starting OpenSSH server daemon...
Feb 20 15:01:01 centos8 sshd[1163]: Server listening on 0.0.0.0 port 22.
Feb 20 15:01:01 centos8 sshd[1163]: Server listening on :: port 22.
Feb 20 15:01:01 centos8 systemd[1]: Started OpenSSH server daemon.
```

Чтобы смотреть логи в реальном времени, по мере их добавления, следует использовать ключ -f. Опять же, комбинируем с каким-нибудь unit-ом для удобства:

```
sudo journalctl -fu sshd
```

```
[user@centos8 log]$ sudo journalctl -b
-- Logs begin at Sat 2021-02-20 15:00:53 +04, end at Sat 2021-02-20 16:56:03 +
Feb 20 15:00:53 centos8 kernel: Linux version 4.18.0-147.8.1.el8_1.x86_64 (mod
Feb 20 15:00:53 centos8 kernel: Command line: BOOT_IMAGE=(hd0,msdos1)/vmlinuz-
Feb 20 15:00:53 centos8 kernel: x86/fpu: Supporting XSAVE feature 0x001: 'x87
Feb 20 15:00:53 centos8 kernel: x86/fpu: Supporting XSAVE feature 0x002: 'SSE
Feb 20 15:00:53 centos8 kernel: x86/fpu: Supporting XSAVE feature 0x004: 'AVX
Feb 20 15:00:53 centos8 kernel: x86/fpu: xstate_offset[2]: 576, xstate_sizes[2]
Feb 20 15:00:53 centos8 kernel: x86/fpu: Enabled xstate features 0x7, context
Feb 20 15:00:53 centos8 kernel: BIOS-provided physical RAM map:
Feb 20 15:00:53 centos8 kernel: BIOS-e820: [mem 0x0000000000000000-0x0000000000000000]
```

Можно посмотреть логи запуска операционной системы:

```
sudo journalctl -b
```

```
[user@centos8 log]$ sudo journalctl -u systemd-udevd --since yesterday --until "5 minutes ago" -o json-pretty -p info
{
    "_CURSOR" : "s=4f22e3230884411b90f948b06da71073;i=177;b=a0792d4497d74>",
    "_REALTIME_TIMESTAMP" : "1613818853981214",
    "_MONOTONIC_TIMESTAMP" : "1940247",
    "_BOOT_ID" : "a0792d4497d74ec2a256d8d61e537055",
    "_MACHINE_ID" : "91222dde95634287a2336070235dc625",
    "_HOSTNAME" : "centos8",
    "_PRIORITY" : "6",
    "_SYSLOG_FACILITY" : "3",
    "_SYSLOG_IDENTIFIER" : "systemd",
    "_UID" : "0",
    "_GID" : "0",
    "_SELINUX_CONTEXT" : "kernel",
```

Также можно гибко выбрать временной промежуток, настроить формат вывода, уровень серьёзности и т.п. Но разбирать или учить все ключи нет необходимости, так как всегда можно посмотреть документацию и найти нужное для конкретной задачи.

```
[user@centos8 log]$ logger -p news.alert Hello World!
[user@centos8 log]$ sudo journalctl -p alert
-- Logs begin at Sat 2021-02-20 15:00:53 +04, end at Sat 2021-02-20 17:03:12 +04.
Feb 20 15:02:52 centos8 user[3369]: Hello
Feb 20 15:03:00 centos8 user[3500]: Hello
Feb 20 17:03:12 centos8 user[10697]: Hello World!
```

Ну и напоследок. Есть утилита logger, которая позволяет вам отправлять логи в syslog, что можно использовать, например, в скриптах или для тестирования работы syslog сервера:

```
logger -p news.alert "Hello World!"
```

Подводя итоги. Мы с вами разобрали логирование в Linux - поговорили про работу демонов rsyslog и journald, про ротацию логов с помощью настроек journald и утилиты logrotate, затронули утилиту sync, а также рассмотрели утилиту journalctl. Умение работать с логами очень важно для администратора, это основа его работы.

2.36.2 Практика

Вопросы

1. Куда программы ведут свои логи?
2. Что такое syslog? Как программы отправляют логи в syslog?
3. Какие объекты логирования есть у syslog?
4. В чем отличие journald от rsyslog?
5. Зачем в системе два демона логирования?
6. Как сделать так, чтобы логи с journald оставались после перезагрузки?
7. Как изменить время хранения логов journald?
8. Как посмотреть сообщения ядра?
9. Как посмотреть логи определённого сервиса?

10. Как посмотреть последние логи сервиса?
11. Как посмотреть логи с последнего запуска системы?
12. Что такое ротация логов и как её настроить?
13. Как отправить свои сообщения в syslog?
14. Как смотреть логи определённого сервиса в реальном времени?

Задания

1. Создайте скрипт, который будет создавать файл. После чего скрипт будет прописывать последнюю строчку из сообщений syslog в этот файл. Затем будет отправлять в syslog своё сообщение - «Logs collected».
2. Настройте rsyslog, чтобы логи, посланные этим скриптом, сохранялись в отдельном файле «collect.log» в директории /var/log.
3. Настройте ротацию этих логов, чтобы они сохранялись на протяжении месяца с еженедельной ротацией.
4. Запустите скрипт и убедитесь, что все ранее проделанные шаги выполнили свою задачу.
5. Создайте скрипт, который будет отправлять логи в syslog – This is normal log → debug, This is error → error и This is critical error → crit. Настройте syslog, чтобы записывал эти логи в соответствующие файлы - /var/log/mylog.debug, /var/log/mylog.crit, /var/log/mylog.error. Настройте ротацию этих логов, чтобы они ротировались 5 раз. debug и error должны ротироваться ежедневно, debug с сжатием, а error без сжатия. crit должен ротироваться раз в неделю.
6. Создайте файл /etc/allowedusers, в котором пропишите логин своего пользователя. Создайте скрипт, который сначала проверит, имеет ли данный текущий пользователь право запускать скрипт (пользователь должен быть в списке /etc/allowedusers). Если прав нету, то следует вывести на экран сообщение «This incident will be reported», а в syslog в секцию error написать «User *username* tried to run this script!». После запуска скрипт должен вывести содержимое директории /data/. После чего предложить пользователю указать имя файла и указать дату и время модификации для смены.

2.37 37. Планировщики задач

2.37.1 37. Планировщики задач

В задачи администратора входит настройка и поддержка ИТ инфраструктуры. Под поддержкой подразумевается регулярное выполнение обслуживающих задач, например, проверка состояния серверов, обновление операционной системы, бэкапы и всё такое. При изучении скриптов мы разобрались как можно упростить некоторые задачи. А если автоматизировать запуск скриптов и команд для обслуживания, появится много свободного времени для других задач. В этом нам помогут несколько утилит, которые можно объединить под общим термином «планировщик задач».

AT(1)	General Commands Manual	AT(1)
NAME		
at, batch, atq, atrm - queue, examine or delete jobs for later execution		
SYNOPSIS		
<pre>at [-V] [-q queue] [-f file] [-mMlv] timespec... at [-V] [-q queue] [-f file] [-mMkv] [-t time] at -c job [job...] atq [-V] [-q queue] at [-rd] job [job...] atrm [-V] job [job...] batch at -b</pre>		

Первая утилита:

at

Она позволяет выполнять какие-то команды в будущем, в заданное время. Это годится для нерегулярных задач, которые нужно выполнить один или пару раз.

```
[user@centos8 ~]$ at 23:00
warning: commands will be executed using /bin/sh
at> systemctl stop database
at> reboot
at> <EOT>
job 3 at Sat Mar  6 23:00:00 2021
[user@centos8 ~]$ at -l
3      Sat Mar  6 23:00:00 2021 a user
[user@centos8 ~]$ at -c 3
#!/bin/sh
# atrun uid=1000 gid=1000
# mail user 0
umask 2
LS_COLORS=rs=0:di=38\;5\;33:ln=38\;5\;51:mh=00:pi=40\;38\;5\;11:s
o=38\;5\;13:do=38\;5\;5:bd=48\;5\;232\;38\;5\;11:cd=48\;5\;232\;3
```

```
LESSOPEN=\\|\\|/usr/bin/lesspipe.sh\\ %s; export LESSOPEN
cd /home/user || {
    echo 'Execution directory inaccessible' >&2
    exit 1
}
${SHELL:-/bin/sh} << 'marcinDELIMITER7f19fc7c'
systemctl stop database
reboot
marcinDELIMITER7f19fc7c
```

Предположим, нам нужно, чтобы сервер перезагрузился в 11 вечера. Для этого пишем:

```
at 23:00
```

После этого at предложит нам ввести список команд. Например, надо остановить сервис базы данных и только потом перезагрузиться - пишем одну команду, Enter, потом другую:

```
systemctl stop database
reboot
```

После чего нажимаем Enter и Ctrl+d, чтобы остановить ввод команд. На экране мы видим, что создалась задача - job - с номером 3, которая выполнится сегодня в 11 вечера. Задач может быть несколько, и чтобы увидеть список всех задач, можно выполнить команду:

```
atq
at -l
```

Чтобы увидеть детали задачи, пишем:

```
at -c 3
```

3 - номер задачи. Команды выполняются от нашего пользователя, но так как это скрипт, причём, запускаемый из оболочки shell, at скопировал наши переменные окружения в этот скрипт, чтобы избежать каких-либо проблем. В самом низу скрипта мы увидим наши команды.

```
[user@centos8 ~]$ echo "reboot" | at 23:30
warning: commands will be executed using /bin/sh
job 6 at Sat Mar 6 23:30:00 2021
[user@centos8 ~]$ nano commands
[user@centos8 ~]$ at 23:45 -f /home/user/commands
warning: commands will be executed using /bin/sh
job 7 at Sat Mar 6 23:45:00 2021
[user@centos8 ~]$ at -l
3      Sat Mar 6 23:00:00 2021 a user
6      Sat Mar 6 23:30:00 2021 a user
7      Sat Mar 6 23:45:00 2021 a user
```

Как вы заметили, at ждал ввода из stdin. А значит мы можем передать ему команды через пайп:

```
echo "reboot" | at 23:00
```

Либо можем указать файл с командами с помощью ключа -f:

```
at 23:45 -f /home/user/commands
```

```
[user@centos8 ~]$ echo "touch file" | at now
warning: commands will be executed using /bin/sh
job 8 at Sat Mar 6 14:33:00 2021
[user@centos8 ~]$ echo "touch file" | at now + 10 minutes
warning: commands will be executed using /bin/sh
job 9 at Sat Mar 6 14:43:00 2021
[user@centos8 ~]$ echo "touch file" | at midnight
warning: commands will be executed using /bin/sh
job 10 at Sun Mar 7 00:00:00 2021
[user@centos8 ~]$ echo "touch file" | at 01:00 03212021
warning: commands will be executed using /bin/sh
job 11 at Sun Mar 21 01:00:00 2021
```

```
[user@centos8 ~]$ at -l
3      Sat Mar 6 23:00:00 2021 a user
6      Sat Mar 6 23:30:00 2021 a user
7      Sat Mar 6 23:45:00 2021 a user
9      Sat Mar 6 14:43:00 2021 a user
10     Sun Mar 7 00:00:00 2021 a user
11     Sun Mar 21 01:00:00 2021 a user
```

Касательно времени у at всё довольно гибко - можно настроить практически любое время. Например:

- at now - выполнит сейчас. Хороший вариант, чтобы протестировать работу.
- at now + 10 minutes - через 10 минут
- at midnight - ночью
- at 01:00 03212021 - 21 марта в час ночи

Есть множество других способов указать время, я показал основное, а более подробно вы можете сами найти в документации и в интернете.

```
[user@centos8 ~]$ at -l
3      Sat Mar  6 23:00:00 2021 a user
6      Sat Mar  6 23:30:00 2021 a user
7      Sat Mar  6 23:45:00 2021 a user
9      Sat Mar  6 14:43:00 2021 a user
10     Sun Mar  7 00:00:00 2021 a user
11     Sun Mar 21 01:00:00 2021 a user
[user@centos8 ~]$ atrm 3
[user@centos8 ~]$ at -r 6
[user@centos8 ~]$ at -l
7      Sat Mar  6 23:45:00 2021 a user
9      Sat Mar  6 14:43:00 2021 a user
10     Sun Mar  7 00:00:00 2021 a user
11     Sun Mar 21 01:00:00 2021 a user
```

Для удаления какой-то задачи можно использовать atrm или at -r:

```
atrm 3
at -r 6
```

```
[user@centos8 ~]$ uptime
14:54:16 up 1:39, 1 user, load average: 0.20, 0.10, 0.03
[user@centos8 ~]$ echo "touch file" | batch
warning: commands will be executed using /bin/sh
job 14 at Sat Mar 6 14:54:00 2021
[user@centos8 ~]$ cat /etc/sysconfig/atd
# specify additional command line arguments for atd
#
# -l Specifies a limiting load factor, over which batch :
#   the compile-time
#   choice of 0.8. For an SMP system with n CPUs, you will p
han n-1.
#
#   -b Specifiy the minimum interval in seconds between the sta
.

#example:
#OPTS="-l 4 -b 120"
```

В документации к at также указана команда batch. Она может пригодится на серверах, когда вы хотите запустить какую-то программу, но сервер слишком нагружен. Если вы выполните команду сейчас, то это скажется на производительности. И тут вам в помощь приходит batch - работает она примерно как at, но без указания времени. Она следит за средней нагрузкой сервера - load average - и когда нагрузка упадёт меньше указанного лимита, по умолчанию это 0.8 - uptime, то batch выполнит указанную вами команду. Лимит можно настроить в файле /etc/sysconfig/atd.

```
[user@centos8 ~]$ systemctl status atd
● atd.service - Job spooling tools
  Loaded: loaded (/usr/lib/systemd/system/atd.service; enabled; vendor preset: enabled)
  Active: active (running) since Sat 2021-03-06 13:14:30 +04; 1h 47min ago
    Main PID: 1174 (atd)
      Tasks: 1 (limit: 11474)
     Memory: 788.0K
        CGroup: /system.slice/atd.service
                 └─1174 /usr/sbin/atd -f
```

Кстати, atd - это демон, который стоит за работой at:

```
systemctl status atd
```

И после изменения настроек в /etc/sysconfig/atd стоит рестартнуть этот сервис. Кстати, сам at может посыпать письмо пользователю после выполнения задачи. Но это отдельная тема.

NAME

`at.allow`, `at.deny` - determine who can submit jobs via `at` or `batch`

DESCRIPTION

The `/etc/at.allow` and `/etc/at.deny` files determine which user can submit commands for later execution via `at(1)` or `batch(1)`.

The format of the files is a list of usernames, one on each line. Whitespace is not permitted.

If the file `/etc/at.allow` exists, only usernames mentioned in it are allowed to use `at`.

If `/etc/at.allow` does not exist, `/etc/at.deny` is checked, every username not mentioned in it is then allowed to use `at`.

An empty `/etc/at.deny` means that every user may use `at`.

If neither exists, only the superuser is allowed to use `at`.

Можно разрешить или запретить пользователям использовать команду `at` и `batch`, для этого нужно указать пользователей в соответствующих файлах - `/etc/at.allow` или `/etc/at.deny`:

`man at.allow`

По умолчанию существует пустой файл `/etc/at.deny` - это говорит о том, что всем пользователям разрешено использовать `at`. Если создать `at.allow`, то только пользователи, указанные в этом файле, будут иметь возможность использовать `at`. Если этих файлов нет, то только root может использовать `at`.

```
[user@centos8 ~]$ systemctl status crond
● crond.service - Command Scheduler
  Loaded: loaded (/usr/lib/systemd/system/crond.service; enabled;
    Active: active (running) since Sat 2021-03-06 13:14:30 +04; 2h 7m 44s ago
      Main PID: 1178 (crond)
        Tasks: 1 (limit: 11474)
       Memory: 2.8M
      CGroup: /system.slice/crond.service
              └─1178 /usr/sbin/crond -n
```

`at` позволяет выполнять команду в указанное время, что удобно для разовых задач. Но автоматизация зачастую предполагает регулярное выполнение одних и тех же задач. Для таких целей используется другая утилита - `cron`:

`systemctl status crond`

GNU nano 2.9.8	/etc/crontab
----------------	--------------

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

# For details see man 4 crontabs

# Example of job definition:
# .----- minute (0 - 59)
# | .----- hour (0 - 23)
# | | .----- day of month (1 - 31)
# | | | .---- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | .--- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon ...
# * * * * * user-name command to be executed
```

Для начала определимся с настройками и посмотрим примеры. Основной файл настроек - /etc/crontab:

```
nano /etc/crontab
```

Тут мы видим оболочку, в которой будут выполняться наши команды, переменную PATH и MAILTO - кому будут отправляться письма о выполнении. Этот файл относится к пользователю root, поэтому он и указан в качестве адресата писем. Но как и в at, в cron у каждого пользователя могут быть свои задачи и, соответственно, своя переменная MAILTO.

Ниже у нас пример настройки задачи. Время можно задать с помощью 5 значений - минуты, часы, день месяца, сам месяц и день недели. Если нам нужно выполнять задачу каждый день, то в качестве значения дня месяца оставляем звёздочку. Каждый месяц? Оставляем звёздочку. Каждый день недели? Звёздочку. Так со всеми значениями. После задания времени нужно указать команду. У пользователя root есть дополнительная возможность указать от чьего имени будет запускаться команда.

 File Edit View Search Terminal Help	user@centos8:~
[user@centos8 ~]\$ echo "export EDITOR=nano" >> ~/.bashrc	
[user@centos8 ~]\$ bash	

Чтобы настроить список задач cron для моего пользователя, я от своего имени запускаю команду:

```
crontab -e
```

е - это edit. Запускается текстовой редактор по умолчанию, т.е. vim. Чтобы открывать crontab в другом редакторе, надо выдать значение переменной EDITOR:

```
echo "export EDITOR=nano" >> ~/.bashrc
```

и открыть новую сессию bash:

```
bash
```

После чего crontab откроется в nano:

```
crontab -e
```

```
[user@centos8 ~]$ crontab -e
crontab: installing new crontab
[user@centos8 ~]$ crontab -l
30 23 * * 7 /usr/bin/backup
```

Предположим, мне нужно выполнять команду backup в каждое воскресенье. Для этого мне нужно определиться со временем - во сколько часов и минут. Допустим в 23:30. Тогда в качестве минут выставляю 30, в качестве часов - 23. День месяца нас не интересует, потому что бэкап у нас по воскресеньям, а это может быть любым днём, поэтому ставим звёздочку. Также нас не интересует конкретный месяц, у нас бэкап должен идти каждый месяц, поэтому также звёздочку. Остается указать день недели - воскресенье. По определённым причинам в некоторых странах, в том числе в США, первым днём недели принято считать воскресенье. В стандарте cron значения недели могут варьироваться от 0 до 6, то есть 0 - это воскресенье, поэтому пишем 0. Хотя в Centos можно указать и 7, но суть в том, что демон cron существует во многих unix-подобных системах и он может немного отличаться. От 0 до 6 это стандарт, который будет работать везде. После чего указываем, какую именно команду выполнять - /usr/bin/backup:

```
30 23 * * 7 /usr/bin/backup
```

Сохраняем и закрываем файл. Список задач можно посмотреть с помощью ключа -l:

```
crontab -l
```

Если бы я указал в качестве минут и часов звёздочку, то команда backup запускалась бы каждую минуту каждого часа воскресенья.

```
30 23 * * 0 /usr/bin/backup
* * * * * touch file
0 * * * * touch file2
0 0 * * * touch file3
3 5 15 * * touch file4
10 15 20 5 * touch file5
```

Давайте рассмотрим пару примеров:

* * * * * touch file - каждую минуту выполнять команду touch file.

```
0 * * * * touch file2 - раз в каждый час, в нулевую минуту, выполнять команду touch file2.  
0 0 * * * touch file3 - раз в день, в 00:00, выполнять команду touch file3.  
3 5 15 * * touch file4 - каждого 15 числа любого месяца в 5:03 выполнять команду touch file4.  
10 15 20 5 * touch file5 - каждого 20 мая в 15:10 выполнять команду touch file5.
```

```
*/5 * * * * touch file6  
0 9,18 * * * touch file7  
0 0 * * 1-5 touch file8
```

Также можно использовать несколько значений, диапазоны или шаги. Например:

```
*/5 * * * * touch file6 - раз в 5 минут  
0 9,18 * * * touch file7 - в 9:00 и 18:00  
0 0 * * 1-5 touch file8 - каждую полночь с понедельника по пятницу
```

The screenshot shows a web browser window with the title 'Crontab.guru - The cron editor'. The URL in the address bar is https://crontab.guru/#14_3_*_1-4,7_*3. The main content area is titled 'crontab guru' and contains the text: "At 03:14 on every 3rd day-of-week in every month from January through April and July." Below this, there is a large input field containing the cron expression '14 3 * 1-4,7 */3'. A legend below the input field defines the symbols used: minute, hour, day (month), month, day (week). The legend entries are:

- *
- ,
-
- /
- 0-6
- SUN-SAT
- 7

- any value
- value list separator
- range of values
- step values
- allowed values
- alternative single values
- sunday (non-standard)

Это очень гибкая система времени и по началу вы можете сомневаться, правильно ли вы задали время. В таких случаях вы можете воспользоваться сайтом crontab.guru. Просто напишите здесь предполагаемое время и он распишет это время английским языком.

```
[user@centos8 ~]$ cat /etc/cron.  
cron.d/      cron.daily/   cron.deny     cron.hourly/  cron.monthly/ cron.weekly  
[user@centos8 ~]$ cat /etc/cron.daily/logrotate  
#!/bin/sh  
  
/usr/sbin/logrotate /etc/logrotate.conf  
EXITVALUE=$?  
if [ $EXITVALUE != 0 ]; then  
    /usr/bin/logger -t logrotate "ALERT exited abnormally with [$EXITVALUE]"  
fi  
exit $EXITVALUE  
[user@centos8 ~]$ █
```

В /etc/ есть несколько директорий для cron:

```
ls /etc/cron.*
```

cron.hourly, cron.daily, cron.weekly и cron.monthly. Когда у вас есть скрипт, который нужно выполнять регулярно, при этом не важно, в какую именно минуту, вы можете закинуть такой скрипт в соответствующую директорию. Для примера, так работает logrotate:

```
cat /etc/cron.daily/logrotate
```

у него нет своего демона, его раз в день запускает cron. Из этого следует вывод - если вы хотите ротировать логи по размеру, а не по дням, то имеет смысл переместить этот скрипт из cron.daily в cron.hourly.

```
[user@centos8 ~]$ cat /etc/cron.hourly/0anacron  
#!/bin/sh  
# Check whether 0anacron was run today already  
if test -r /var/spool/anacron/cron.daily; then  
    day=`cat /var/spool/anacron/cron.daily`  
fi  
if [ `date +%Y%m%d` = "$day" ]; then  
    exit 0  
fi  
  
# Do not run jobs when on battery power  
online=1  
for psupply in AC ADP0 ; do  
    sysfile="/sys/class/power_supply/$psupply/online"  
    if [ ! f $sysfile l + then
```

Ещё один интересный файл - /etc/cron.hourly/0anacron:

```
cat /etc/cron.hourly/0anacron
```

Есть такая программа - anacron, которая также позволяет планировать задачи. В чём разница: представьте, что в cron-е написано сделать бэкап в полночь. Но почему-то в это время компьютер был

выключен, соответственно, cron не сработал. Тогда cron просто пропустит задачу и выполнит в следующий раз, по графику. Т.е. cron обычно жёстко привязан к времени, что имеет смысл на серверах, где не стоит делать задачи, тот же бэкап, в рабочее время. Но на пользовательских компьютерах, если я вечером выключил компьютер, а утром включил, то дожидаться следующей полночи для бэкапа не очень хорошая идея, лучше сделать его после включения. Для этого лучше подходит anacron - он работает с периодами времени. И если, скажем, вы в anacron настройте бэкап раз в день, то, если компьютер будет выключен в запланированное для бэкапа время, бэкап будет сделан после включения.

```
[user@centos8 ~]$ cat /etc/anacrontab
# /etc/anacrontab: configuration file for anacron

# See anacron(8) and anacrontab(5) for details.

SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
# the maximal random delay added to the base delay of the jobs
RANDOM_DELAY=45
# the jobs will be started during the following hours only
START_HOURS_RANGE=3-22

#period in days    delay in minutes    job-identifier    command
1      5            cron.daily          nice run-parts /etc/cron.daily
7      25           cron.weekly        nice run-parts /etc/cron.weekly
@monthly 45         cron.monthly       nice run-parts /etc/cron.monthly
```

Настройки anacron лежат в файле /etc/anacrontab:

```
cat /etc/anacrontab
```

Помните директории /etc/cron.daily, weekly и monthly, где лежат скрипты? Раньше их выполнял сам cron, но сейчас этим занимается anacron. И это видно по последним строчкам. Тут у нас вместо конкретики по времени - периоды: раз в день, в неделю и в месяц.

Следующий столбик - время отсрочки. Допустим, cron.daily выполняется раз в день, но не сразу при достижении времени, а минут через 5. Причём, переменная RANDOM_DELAY добавляет к эти 5 минутам рандомное количество минут, максимум 45. Для чего это нужно? Например, когда у вас в сети много компьютеров и все они в одно и тоже время начинают делать бэкап, это начинает грузить сеть. А рандом позволяет компьютерам немного распределить это время, чтобы все разом не грузили сеть. Ещё одна переменная - START_HOURS_RANGE - говорит о том, что эти задачи будут запускаться только в промежуток между 3:00 и 22:00. Естественно, все эти переменные можно изменить под свои задачи.

После отсрочки указывается идентификатор задачи (job-identifier), по которому можно будет найти задачу в логах. А в конце - команда. Можно добавить какую-то свою задачу сюда, со своей периодичностью, либо закинуть скрипты в указанные директории.

Мы с вами разобрали at, cron и anacron, которые позволяют нам запланировать задачи и выполнять их. Администратору очень важно видеть список запланированных задач, это позволяет контролировать происходящее на сервере, да и всякий вирусняк любит себя записывать в планировщики. У некоторых пользователей может быть свой crontab, at и в редких случаях anacron. А пользовательские задачи не лежат в директории /etc/, поэтому их оттуда не отследить.

```
[user@centos8 ~]$ ls /var/spool/
anacron at cron cups lpd mail plymouth up2date
[user@centos8 ~]$ sudo cat /var/spool/cron/user
[sudo] password for user:
30 23 * * 0 /usr/bin/backup
* * * * touch file
0 * * * * touch file2
0 0 * * * touch file3
3 5 15 * * touch file4
10 15 20 5 * touch file5
```

Для таких задач есть директория `/var/spool`:

```
ls /var/spool
```

И у `at`, и у `cron` тут есть свои директории. Если посмотреть содержимое этих директорий, можно понять, какие задачи запланированы у пользователей.

```
[user@centos8 ~]$ systemctl list-timers
NEXT           LEFT      LAST          PASSED      UNIT
Sat 2021-03-06 19:00:00 +04 4min 29s left Sat 2021-03-06 18:50:01 +04 5min ago  sysstat-collect
Sat 2021-03-06 19:27:01 +04 31min left   Sat 2021-03-06 18:27:01 +04 28min ago  dnf-makecache.
Sun 2021-03-07 00:00:00 +04 5h 4min left   Sat 2021-03-06 13:14:28 +04 5h 41min ago unbound-anchor
Sun 2021-03-07 00:07:00 +04 5h 11min left n/a                                n/a          sysstat-summary
```

Также стоит поговорить про таймеры `systemd`. В `systemd` встроен функционал планировщика задач, который выполнен в виде `unit`-ов. Помните, мы говорили, что в `systemd` функционал реализован в виде `unit`-ов - сервисы это юниты, группы сервисов - таргеты - тоже юниты, устройства - юниты, монтируемые файловые системы - тоже юниты. Так вот, есть ещё один тип юнитов - таймеры:

```
systemctl list-timers
```

Эти таймеры можно связать с сервисами, что позволит вам гибко настроить время работы сервиса. Кроме существующих сервисов, вы можете использовать таймеры как замену `cron`-у и `at`, чтобы унифицировать управление регулярными задачами через `systemd`. Обычно таймер надо создавать вручную, что не сложно, но больше относится к теме создания `unit`-ов `systemd`. А я не хочу мешать темы, поэтому когда-нибудь мы эту тему детальнее разберём, но если вам интересно, почитайте по [ссылке](#).

```
[user@centos8 ~]$ systemctl-run --on-calendar=hourly /bin/touch /tmp/file
Running timer as unit: run-u562.timer
Will run service as unit: run-u562.service
[user@centos8 ~]$ systemctl cat run-u562.timer
# /run/systemd/transient/run-u562.timer
# This is a transient unit file, created programmatically via the systemd API. >
[Unit]
Description=/bin/touch /tmp/file

[Timer]
OnCalendar=hourly
RemainAfterElapse=no
[user@centos8 ~]$ systemctl cat run-u562.service
# /run/systemd/transient/run-u562.service
# This is a transient unit file, created programmatically via the systemd API. >
[Unit]
Description=/bin/touch /tmp/file

[Service]
ExecStart=
ExecStart=@/bin/touch "/bin/touch" "/tmp/file"
```

Есть простой способ создать таймеры, хотя и временные:

```
systemd-run --on-calendar=hourly /bin/touch /tmp/file
systemctl cat run-u562.timer
systemctl cat run-u562.service
```

При этом создаётся и таймер, и сервис, но это всё существует, пока не выполнится временное условие, либо пока не перезагрузится сервер. Простой пример - таймер, который раз в час запускает touch file. Но, опять же, это временный таймер, постоянные таймеры надо создавать по другому.

Подводя итоги, мы с вами разобрали несколько планировщиков задач - at, cron, anacron и таймеры systemd, которые одновременно работают на наших системах, у каждого свои возможности, где-то они пересекаются, где-то уникальны. Так или иначе они упрощают работу системному администратору, поэтому стоит научиться ими пользоваться.

2.37.2 Практика

Вопросы

1. Для чего нужен планировщик задач?
2. Чем отличаются планировщики задач друг от друга?
3. В каких случаях стоит использовать cron, а в каких - anacron?
4. Расшифруйте 0 4 8-14 * *
5. Какой планировщик стоит использовать на высоконагруженных серверах, если время выполнения задачи не критично?
6. Как узнать, есть ли у пользователей в системе свои задачи в кроне?

Задания

1. Создайте скрипт, который создаёт файл. Настройте запуск этого скрипта раз в минуту. Проверьте результат.
2. Настройте в at задачу, чтобы она создала файл через 3 минуты.
3. Настройте cron, чтобы он создавал директорию при каждой перезагрузке.

2.38 38. Создание backup скрипта

2.38.1 38. Создание backup скрипта

Планировщики задач используют для разных целей, одна из самых популярных - бэкап данных. Сейчас во многих компаниях для бэкапа используются готовые решения, которые делают это централизовано, со всех систем, умеют самостоятельно восстанавливать и имеют широкий функционал. Однако дедовский метод организации бэкапа с линукс серверов всё ещё имеет большую популярность и применяется не только в малых компаниях. На самом деле, у меня скопилось несколько небольших утилит, о которых нужно рассказать, и все эти утилиты так или иначе применяются при бэкапе. Поэтому мы будем создавать решение для бэкапа и одновременно рассмотрим каждую из утилит.

```
[user@centos8 ~]$ pwd
/home/user
[user@centos8 ~]$ ls
a          Documents  FILE    file7  passwd   Templates
commands   Downloads  FILE    is      Pictures  test
Desktop    file       file2   Music   Public    this
directory  fiLE      file6   new     temp     Videos
[user@centos8 ~]$ du -sh /home/user
291M      /home/user
```

Для начала следует определиться, что мы будем бэкапить. Для примера возьмём домашнюю директорию пользователя /home/user:

```
pwd
ls
```

Следует определиться с размером этой директории, для этого я использую утилиту du с ключами -sh:

```
du -sh /home/user
```

```
[user@centos8 ~]$ du file
0          file
[user@centos8 ~]$ echo Hello World > file
[user@centos8 ~]$ du file
4          file
[user@centos8 ~]$ du -h file
4.0K      file
[user@centos8 ~]$ du /boot/initramfs-0-rescue-91222dde95634287a2336070235dc625.img
70036    /boot/initramfs-0-rescue-91222dde95634287a2336070235dc625.img
[user@centos8 ~]$ du -h /boot/initramfs-0-rescue-91222dde95634287a2336070235dc625.img
69M       /boot/initramfs-0-rescue-91222dde95634287a2336070235dc625.img
```

Приостановимся на этой утилите. du file покажет размер файла. В данном случае файл пустой, поэтому его размер 0. Если я добавлю что-то в файл:

```
echo Hello World > file
```

его размер изменится:

```
du file
```

Теперь мы видим, что размер стал 4. По умолчанию, размер показывается в кибайтах - т.е. 4096 байт. Чтобы не считать размер, особенно если файл большой, лучше использовать опцию -h - human readable - то есть понятный человеку размер:

```
du -h file
du /boot/initramfs-*
du -h /boot/initramfs-*
```

```
[user@centos8 ~]$ du -h Pictures/
17M      Pictures/
[user@centos8 ~]$ du -h Documents/
0        Documents/test
0        Documents/this
0        Documents/is
0        Documents/new
0        Documents/dir
0        Documents/New Dir
0        Documents/1/2/3
0        Documents/1/2
0        Documents/1
0        Documents/
[user@centos8 ~]$ du -sh Documents/
0        Documents/
```

Можно узнать размер директории со всем содержимым, просто указав директорию - du -h Pictures. Но если в этой директории будут поддиректории, то du покажет размер каждой из них:

```
du -h Documents
```

Чтобы показать общий размер директории можно использовать ключ -s:

```
du -sh Documents
```

```
[user@centos8 ~]$ sudo du -sh /boot/*
184K    /boot/config-4.18.0-147.8.1.el8_1.x86_64
184K    /boot/config-4.18.0-147.el8.x86_64
12K     /boot/efi
5.5M    /boot/grub2
69M     /boot/initramfs-0-rescue-91222dde95634287a2336070235dc625.img
31M     /boot/initramfs-4.18.0-147.8.1.el8_1.x86_64.img
22M     /boot/initramfs-4.18.0-147.8.1.el8_1.x86_64kdump.img
28M     /boot/initramfs-4.18.0-147.el8.x86_64.img
19M     /boot/initramfs-4.18.0-147.el8.x86_64kdump.img
```

Чтобы увидеть размеры всех файлов в директории следует передать их команде du с помощью, например, звёздочки:

```
sudo du -sh /boot/*
```

На всякий случай напомню, что звёздочку обрабатывает bash - т.е. при виде звёздочки он собирает имена всех файлов и передаёт их команде du в виде аргументов:

```
du file1 file2 file3
```

```
[user@centos8 ~]$ sudo du -sh /boot/* | sort -h | tail -3
28M      /boot/initramfs-4.18.0-147.el8.x86_64.img
31M      /boot/initramfs-4.18.0-147.8.1.el8_1.x86_64.img
69M      /boot/initramfs-0-rescue-91222dde95634287a2336070235dc625.img
[user@centos8 ~]$ sudo du -sh /boot/* | sort -rh | head -3
69M      /boot/initramfs-0-rescue-91222dde95634287a2336070235dc625.img
31M      /boot/initramfs-4.18.0-147.8.1.el8_1.x86_64.img
28M      /boot/initramfs-4.18.0-147.el8.x86_64.img
[user@centos8 ~]$ sudo ls -lhS /boot/* | head -3
-rw-----. 1 root root 69M Apr 21 2020 /boot/initramfs-0-rescue-91222dde95634287a2336070235dc625.img
-rw-----. 1 root root 31M Dec 5 14:42 /boot/initramfs-4.18.0-147.8.1.x86_64.img
-rw-----. 1 root root 28M Apr 21 2020 /boot/initramfs-4.18.0-147.el8.x86_64.img
```

Весьма полезно бывает отсортировать файлы по размеру, например, чтобы увидеть самые большие файлы. du сам не сортирует, но для этого можно использовать команду sort с ключом -h:

```
sudo du -sh /boot/* | sort -h | tail -3
```

По умолчанию она сортирует от самого маленького файла к самому большому, что не всегда удобно, поэтому сортировку можно перевернуть с помощью ключа -r:

```
sudo du -sh /boot/* | sort -rh | head -3
```

Команда ls также может вывести размеры файлов и сразу их отсортировать:

```
sudo ls -lhS /boot/* | head -3
```

Но, как видите, вывод ls не всегда удобен для чтения.

У du есть и другие интересные ключи, но для большинства случаев сказанного выше будет достаточно. Поэтому продолжим.

```
[user@centos8 ~]$ du -sh /home/user
291M      /home/user
[user@centos8 ~]$ find /home/user -type f | wc -l
5378
[user@centos8 ~]$ find /home/user -type d | wc -l
287
```

И так, мы знаем, что наша домашняя директория занимает столько-то мегабайт:

```
du -sh /home/user
```

И здесь огромное количество файлов и директорий. Давайте узнаем сколько, для этого используем утилиту find:

```
find /home/user -type f | wc -l
find /home/user -type d | wc -l
```

Примерно 5 тысяч файлов и 300 директорий. Теперь разберём, как мы это узнали.

```
[user@centos8 ~]$ find /home/user -type f | head -3
/home/user/.mozilla/firefox/j6ukw64e.default-default/.parentlock
/home/user/.mozilla/firefox/j6ukw64e.default-default/compatibility.ini
/home/user/.mozilla/firefox/j6ukw64e.default-default/permissions.sqlite
[user@centos8 ~]$ find /home/user -type d | head -3
/home/user
/home/user/.mozilla
/home/user/.mozilla/extensions
```

find - важная утилита, с которой нужно научиться работать. Синтаксис примерно такой - find где, а дальше что мы ищем. В данном случае мы искали файлы - ключ -type f, а также директории - -type d. find найденное выводит в виде списка, по одному значению на строку:

```
find /home/user -type f | head -3
find /home/user -type -d | head -3
```

поэтому можно просто посчитать количество полученных строк и это будет количеством файлов. Для этого мы использовали утилиту wc с опцией -l - посчитать количество строк.

```
[user@centos8 ~]$ sudo find /etc -name passwd
/etc/pam.d/passwd
/etc/passwd
[user@centos8 ~]$ sudo find /etc -name "*passwd*"
/etc/security/opasswd
/etc/pam.d/passwd
/etc/passwd-
/etc/passwd
/etc/passwd1
[user@centos8 ~]$ sudo find /etc -name *passwd*
/etc/pam.d/passwd
/etc/passwd
```

find обладает большим функционалом для поиска файлов. Самое простое - поиск файлов по имени:

```
sudo find /etc/ -name passwd
```

При поиске можно использовать регулярные выражения:

```
sudo find /etc/ -name "*passwd"
```

При этом не забывайте брать выражение в кавычки.

```
[user@centos8 ~]$ sudo find / -user root -type f -perm /u=s -exec ls -l {} \; 2> /dev/null
-rwsr-xr-x. 1 root root 38680 May 11 2019 /usr/bin/fusermount
-rwsr-xr-x. 1 root root 133928 Nov 9 2019 /usr/bin/chage
-rwsr-xr-x. 1 root root 156736 Nov 9 2019 /usr/bin/gpasswd
-rwsr-xr-x. 1 root root 88488 Nov 9 2019 /usr/bin/newgrp
-rwsr-xr-x. 1 root root 61856 Nov 9 2019 /usr/bin/mount
-rwsr-xr-x. 1 root root 62104 Nov 9 2019 /usr/bin/su
-rwsr-xr-x. 1 root root 40728 Nov 9 2019 /usr/bin/umount
```

Для примера, найдём все файлы с правами SUID, принадлежащие root пользователю:

```
sudo find / -user root -type f -perm /u=s -exec ls -l {} \; 2> /dev/null
```

Некоторые опции мы уже разбирали, из нового - perm - permissions - права. С помощью этой опции мы можем искать файлы с определёнными правами. Также мы можем запускать команды с помощью -exec и передавать им найденный список файлов. В данном случае я передал список команде ls -l.

```
[user@centos8 ~]$ sudo find /etc/ -name passwd -ls
1771959      4 -rw-r--r--   1 root      root          168 May 11 201
18753023      4 -rw-r--r--   1 root      root         3833 Feb  9 17:1
[user@centos8 ~]$ sudo find /etc/ -name passwd -exec cp -v {} /tmp/ \;
'/etc/pam.d/passwd' -> '/tmp/passwd'
'/etc/passwd' -> '/tmp/passwd'
[user@centos8 ~]$ sudo find /etc/ -name passwd -ok cp -v {} /tmp/ \;
< cp ... /etc/pam.d/passwd > ? y
'/etc/pam.d/passwd' -> '/tmp/passwd'
< cp ... /etc/passwd > ? n
```

У find есть список готовых команд, которые можно выполнить с найденными файлами. Среди них есть тот же самый ls:

```
sudo find /etc/ -name passwd -ls
```

Можно, например, список найденных файлов копировать:

```
sudo find /etc/ -name passwd -exec cp -v {} /tmp/ \;
```

Также можно -exec заменить на -ok, чтобы перед каждым действием спрашивать пользователя:

```
sudo find /etc/ -name passwd -ok cp -v {} /tmp/ \;
```

find заменяет фигурные скобки найденным значением.

```
[user@centos8 ~]$ find /home/user -name file2 -exec echo {} \;
/home/user/file2
/home/user/directory/file2
/home/user/test/file2
[user@centos8 ~]$ find /home/user -name file2 -exec echo {} +
/home/user/file2 /home/user/directory/file2 /home/user/test/file2
```

Точка с запятой означает, что команда будет запущена для каждого значения отдельно. Вместо точки с запятой можно использовать плюс, чтобы запустить всего одну команду со списком значений. Для примера используем echo:

```
find /home/user -name file2 -exec echo {} \;
find /home/user -name file2 -exec echo {} +
```

В первом случае команда echo запустилась 3 раза с разными аргументами, что-то вроде цикла for со списком значений. Во втором случае команда find получила разом все три аргумента, поэтому вывела всё в одну строку.

В общем, с помощью find можно гибко искать по всей системе файлы основываясь на различных значениях - имени файла, владельце, правах, времени доступа и изменении файла и т.п. После чего можно сразу выполнить с файлом какое-либо действие - удалить, скопировать, посмотреть и т.п. Я раскрыл не весь функционал find-а, но вы всегда можете найти необходимую опцию в man-е или гугле. find нам ещё понадобится чуть позже, а пока вернёмся к бэкапу.

И так, у нас есть директория, в которой тысячи файлов и сотни директорий. Самый простой способ бэкапа - просто скопировать директорию. Но нужно понимать, что при этом система будет копировать каждый файл и каждую директорию по отдельности, создавая для каждой копии новый инод, проделывая кучу операций. При этом, если мы будем копировать на другой компьютер - опять же, все операции будут проводиться с каждым файлом по отдельности. Для нас это может быть и прозрачно, но для системы это множество лишних операций. И это у нас ещё тестовая система, где файлов мало, а в рабочей среде файлов может быть во много раз больше. Для бэкапа и передачи большого количества файлов лучше использовать архивацию - объединить несколько файлов в один. Для архивации используется утилита tar.

```
[user@centos8 ~]$ tar -cf /tmp/user.tar /home/user
tar: Removing leading `/' from member names
[user@centos8 ~]$ du -h /tmp/user.tar
289M  /tmp/user.tar
[user@centos8 ~]$ tar -tf /tmp/user.tar | head -5
/home/user/
/home/user/.mozilla/
/home/user/.mozilla/extensions/
/home/user/.mozilla/plugins/
/home/user/.mozilla/firefox/
```

Синтаксис для создания архива такой - tar -cf - где ключ -c это create, т.е. создать, а ключ -f это file - файл архива. После f нужно указать директорию и имя архива, после чего указываются директории и файлы для добавления в архив. Для примера, создадим архив домашней директории пользователя:

```
tar -cf /tmp/user.tar /home/user
```

Этой командой tar создаёт архив в директории /tmp/ с названием user.tar и добавляет в этот архив директорию /home/user. Обычно для архивов используется расширение .tar. Размер архива примерно соответствует размеру директории:

```
du -h /tmp/user.tar
```

Чтобы посмотреть содержимое архива можно использовать ключ -t:

```
tar -tf /tmp/user.tar
```

Кстати, обратите внимание, что при создании архива tar выдал сообщение об удалении символа корня со всех файлов. И при просмотре содержимого архива это видно - в начале каждого файла отсутствует слэш /. Т.е. используются относительные пути. Это позволяет распаковывать архив в любую директорию.

```
[user@centos8 ~]$ cd /tmp/  
[user@centos8 tmp]$ tar -xf user.tar  
[user@centos8 tmp]$ ls -l /tmp/home/  
total 4  
drwx----- 23 user user 4096 Mar 13 15:11 user  
[user@centos8 tmp]$ █
```

Для распаковки архива используется ключ -x - extract - извлечь:

```
cd /tmp/  
tar -xf user.tar  
ls /tmp/home
```

Во первых, я не спроста зашёл в директорию tmp - tar по умолчанию распаковывает содержимое архива в текущую директорию. Т.е. я находился в /tmp и tar распаковал архив в эту директорию. Во вторых, из-за того, что в архиве при создании был убран корень, то все файлы распаковались в директорию /tmp, в итоге получилось /tmp/home/user.

```
[user@centos8 tmp]$ tar -xf user.tar -C /home/user/Documents/ home/user/file7  
[user@centos8 tmp]$ ls -R /home/user/Documents/home  
/home/user/Documents/home:  
user  
  
/home/user/Documents/home/user:  
file7
```

Из архива можно распаковывать определённый файл, а также указывать директорию, куда это будет распаковываться с помощью ключа -C:

```
tar -xf user.tar -C /home/user/Documents home/user/file7
```

При этом, как видите, везде используется относительный путь home/user.

```
[user@centos8 tmp]$ tar -cf user2.tar -C /home/user .
[user@centos8 tmp]$ tar -tf user2.tar | head -3
./
./mozilla/
./mozilla/extensions/
[user@centos8 tmp]$ tar -uf user2.tar /tmp/file2
tar: Removing leading `/' from member names
tar: Removing leading `/' from hard link targets
[user@centos8 tmp]$ tar -tf user2.tar | tail -2
./commands
tmp/file2
```

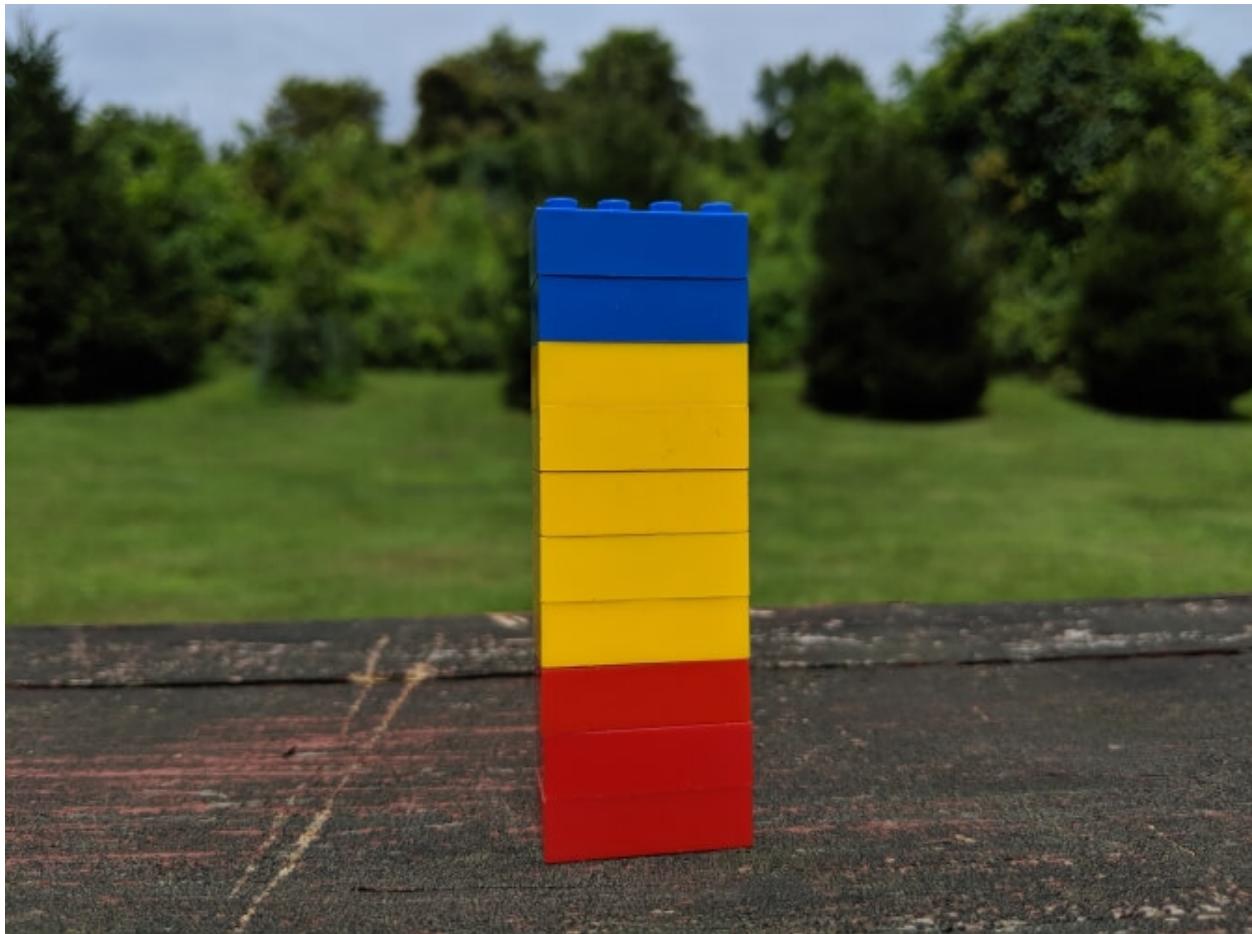
Если при создании архива мы не хотим включать путь к самой директории, как это было с `home/user`, то нужно либо зайти в эту директорию и указать текущую директорию, либо использовать тот же ключ `-C`, который заходит в директорию за нас. А дальше указать точку в качестве текущей директории:

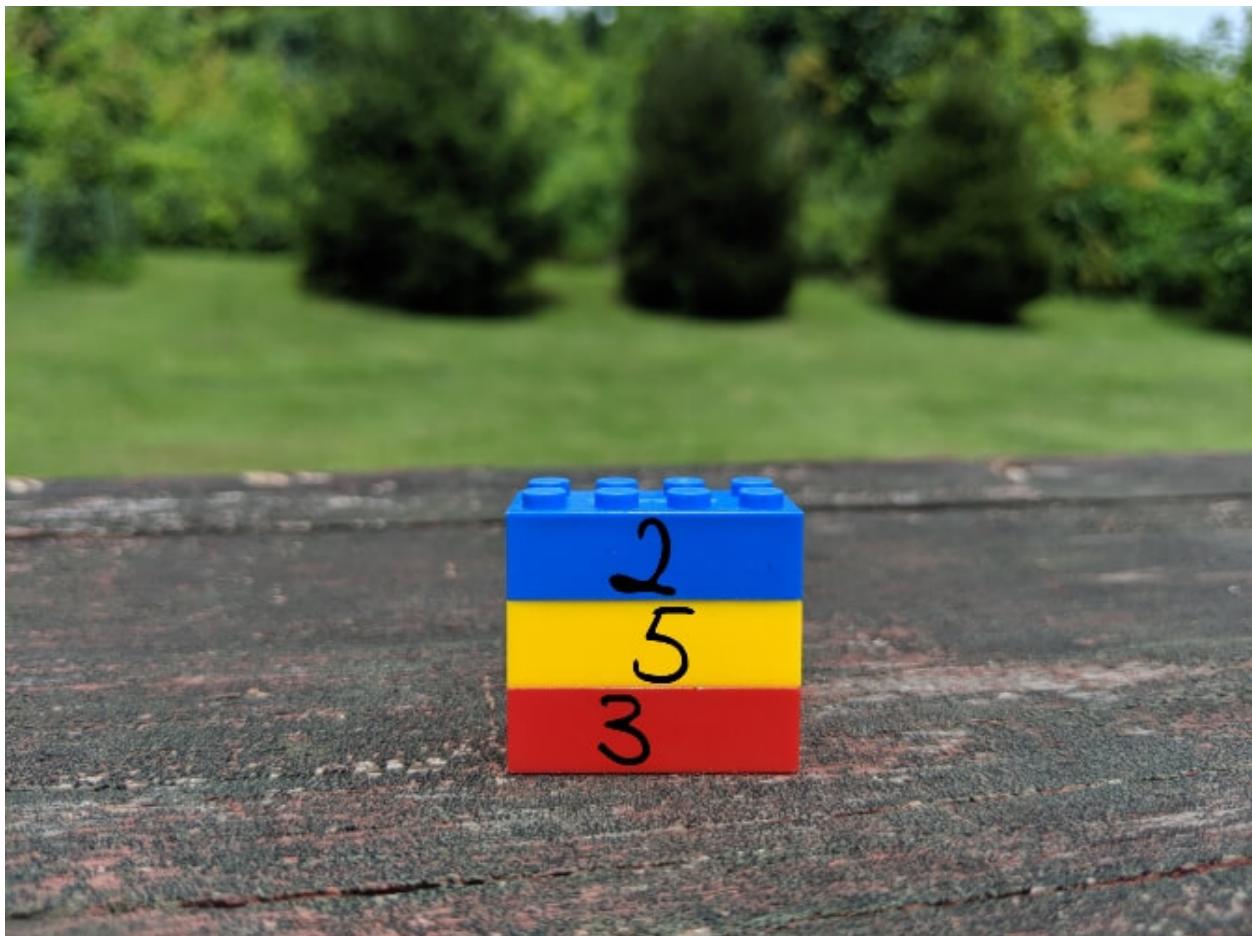
```
tar -cf user2.tar -C /home/user .
tar -xf user2.tar | head -5
```

Если же мы хотим добавить новые файлы в архив используем ключ `-u` - update:

```
tar -uf user2.tar tmp/file2
tar -tf user2.tar | tail -2
```

Ладно, с архивом разобрались. Для бэкапа мы будем создавать архив, в котором будут все наши файлы. Если что - распакуем и достанем нужный файл. Но каждый наш бэкап будет занимать места столько же, сколько сами данные. А если мы делаем ежедневный бэкап и храним их неделю, то нам понадобится в 7 раз больше места. В нашем случае 300 мегабайт не много и не критично, а в рабочих системах данные могут весить очень много, а значит надо как-то экономить место при бэкапе. Для этого можем воспользоваться сжатием данных.





Существуют разные типы сжатия - с потерей или без потерь, разные алгоритмы и утилиты. Вдаваться в детали мы не будем, но эти картинки иллюстрируют простой механизм сжатия без потерь - у нас есть сколько-то блоков данных, в них повторяющиеся данные - мы можем заменить повторяющиеся данные просто ссылками на один блок. Для сжатия мы будем использовать [gzip](#), хотя есть другие алгоритмы и утилиты, но gzip наиболее популярен.

```
[user@centos8 tmp]$ du -h user.tar  
289M    user.tar  
[user@centos8 tmp]$ gzip -k user.tar  
[user@centos8 tmp]$ du -h user.tar.gz  
178M    user.tar.gz  
[user@centos8 tmp]$ bzip2 -k user.tar  
  
[user@centos8 tmp]$  
[user@centos8 tmp]$ du -h user.tar.bz2  
172M    user.tar.bz2
```

И так, для сжатия с помощью gzip можно указать gzip и нужный файл:

```
du -h user.tar  
gzip -k user.tar  
du -h user.tar.gz
```

Здесь ключ k нужен, чтобы не удалять оригинальный файл, так как по умолчанию gzip сжимает файл и удаляет оригинал. В обычных условиях нам оригинал файла не нужен после сжатия, но для тестов он ещё пригодится. gzip создаёт файл с расширением .gz и сжимать можно не только архивы, но и сами файлы. Как видите, сжатый архив весит примерно на треть меньше, но при сжатии задействуется процессор и это занимает время. Для примера используем другую утилиту для сжатия - bzip2 - которая использует другой алгоритм сжатия:

```
bzip2 -k user.tar  
du -h user.tar.bz2
```

Как видите, с этой утилитой получается сжать файл чуть больше, но при этом процесс требует больше ресурсов и времени.

```
[user@centos8 tmp]$ tar -czf /tmp/gzipped.tar.gz -C /home/user .  
[user@centos8 tmp]$ du -h gzipped.tar.gz  
178M    gzipped.tar.gz  
[user@centos8 tmp]$ tar -tf gzipped.tar.gz | head -3  
./  
./mozilla/  
./mozilla/extensions/
```

Обычно процесс архивации и сжатия совмещают, для этого с tar используют ключ -z:

```
tar -czf /tmp/gzipped.tar.gz -C /home/user .  
du -h gzipped.tar.gz
```

Как видите, архив сразу получился сжатым и нам не пришлось вводить команду для сжатия.

```
[user@centos8 tmp]$ mkdir restore
[user@centos8 tmp]$ tar -xf gzipped.tar.gz -C restore/
[user@centos8 tmp]$ ls restore/
a          directory  file   FILE   file7  new      Public    test
commands  Documents  file1  file2  is      passwd  temp     this
Desktop   Downloads  file3  file6  Music   Pictures Templates Videos
```

При этом, для распаковки сжатого архива дополнительных ключей не требуется:

```
mkdir restore
tar -xf gzipped.tar.gz -C restore/
ls restore/
```

```
[user@centos8 tmp]$ hostname
centos8
```

Хорошо, теперь мы знаем как делать бэкап. Но для удобства названия бэкапов должны говорить за себя. Например, они должны содержать имя хоста и время бэкапа. В этом нам поможет bash. Определить имя хоста можно легко - просто команда:

```
hostname
```

```
[user@centos8 tmp]$ date
Sat Mar 13 17:32:10 +04 2021
[user@centos8 tmp]$ date +'%d'
13
[user@centos8 tmp]$ date +' %d.%m'
13.03
[user@centos8 tmp]$ date +' %d.%m.%Y'
13.03.2021
[user@centos8 tmp]$ date +' %d.%m.%Y:%H'
13.03.2021:17
[user@centos8 tmp]$ date +' %d.%m.%Y_%H:%M'
13.03.2021_17:33
[user@centos8 tmp]$ date +' %d.%m.%Y_%H.%M'
13.03.2021_17.37
[user@centos8 tmp]$ █
```

Дальше нам стоит определиться с форматом времени. Для показа времени используем команду date. Но стандартный формат вывода date не подходит для названия файлов. И date очень гибкий в этом плане:

```
date +'%d'
date +'d.%m'
date +'d.%m.%Y'
date +'d.%m.%Y:%H'
date +'d.%m.%Y_%H:%M'
```

На линуксах с двоеточием в имени файла проблем не будет, но если нам вдруг понадобится перенести файл на Windows, придётся переименовывать файл. Поэтому лучше заменим двоеточие на точку:

```
date +'d.%m.%Y:%H'
date +'d.%m.%Y_%H.%M'
```

```
[user@centos8 tmp]$ tar -czf /tmp/$(hostname)_$(date +'%d.%m.%Y_%H.%M').tar.gz -C /home/user .
[user@centos8 tmp]$ du -h /tmp/centos8_13.03.2021_17.40.tar.gz
178M   /tmp/centos8_13.03.2021_17.40.tar.gz
[user@centos8 tmp]$ █
```

Теперь интегрируем это в команду tar:

```
tar -czf /tmp/$(hostname)_$(date +'%d.%m.%Y_%H.%M').tar.gz -C /home/user .
du -h /tmp/centos8*
```

В итоге у нас получился сжатый архив с нашей домашней директорией с датой и временем создания.

```
[user@centos8 tmp]$ sudo mkdir /backup
[user@centos8 tmp]$ sudo chown user:user /backup
[user@centos8 tmp]$ tar -czf /backup/$(hostname)_$(date +'%d.%m.%Y_%H.%M').tar.gz -C /home/user .
[user@centos8 tmp]$ ls /backup/
centos8_13.03.2021_17.52.tar.gz
[user@centos8 tmp]$ █
```

Так как /tmp не подходящая директория для бэкапов, создадим директорию /backup, выдадим права и протестируем:

```
sudo mkdir /backup
sudo chown user:user /backup
tar -czf /backup/$(hostname)_$(date +'%d.%m.%Y_%H.%M').tar.gz -C /home/user .
ls /backup
```

GNU nano 2.9.8

mybackupscript

```

1 #!/bin/bash
2
3 BDIR=/backup
4 BDATE=$(date +'%d.%m.%Y_%H.%M')
5 FILENAME=$BDIR/$(hostname)_$BDATE
6
7 tar -cvzf $FILENAME.tar.gz -C /home/user . &> $FILENAME.log

```

Сделаем из нашей команды скрипт, в команде tar добавим опцию verbose -v - и добавим перенаправление stdout и stderr в файл лога.

```

[user@centos8 ~]$ chmod +x mybackupscript
[user@centos8 ~]$ ./mybackupscript
[user@centos8 ~]$ du -h /backup/*
556K    /backup/centos8_13.03.2021_18.11.log
178M    /backup/centos8_13.03.2021_18.11.tar.gz
[user@centos8 ~]$ tail -3 /backup/centos8_13.03.2021_18.11.log
./nanorcold
./passwd
./commands
[user@centos8 ~]$ 

```

Дадим права и протестируем:

```

chmod +x mybackupscript
./mybackupscript
du -h /backup/*
tail -3 /backup/*.log

```

Скрипт работает.

```

[user@centos8 ~]$ crontab -e
crontab: installing new crontab
[user@centos8 ~]$ crontab -l
16 18 * * * /home/user/mybackupscript
[user@centos8 ~]$ date
Sat Mar 13 18:15:34 +04 2021
[user@centos8 ~]$ date
Sat Mar 13 18:16:51 +04 2021
[user@centos8 ~]$ ls /backup/
centos8_13.03.2021_18.11.log      centos8_13.03.2021_18.16.log
centos8_13.03.2021_18.11.tar.gz   centos8_13.03.2021_18.16.tar.gz
[user@centos8 ~]$ 

```

Теперь добавим его в crontab. Я добавлю на ближайшее время для теста:

```
crontab -e
```

```
16 18 * * * /home/user/mybackupscript
```

```
crontab -l  
date  
ls /backup/
```

Как видите, cron сработал, бэкап создался.

Но, как и с логами, бэкапы надо периодически удалять, чтобы не забивать всё свободное пространство. Для начала нужно определиться, сколько мы будем хранить бэкапы. Предположим, 5 дней. Бэкапы старее 5 дней мне не нужны. Теперь нужно научить систему удалять такие бэкапы. Можно написать какой-нибудь скрипт, который будет считать по датам и сравнивать с именами, но это очень сложно и не лучшее решение. Лучше всего для этого использовать find, который может находить файлы, с ключом -mtime - который ищет по времени модификации файла.

```
[user@centos8 ~]$ touch -t 202103081825 /backup/centos8_13.03.2021_18.16.tar.gz  
[user@centos8 ~]$ ls -l /backup/centos8_13.03.2021_18.16.tar.gz  
-rw-r--r-- 1 user user 186328806 Mar  8 18:25 /backup/centos8_13.03.2021_18.16.tar.gz  
[user@centos8 ~]$
```

Для начала нам понадобится бэкап старее 5 дней. Но сидеть и ждать 5 дней я не хочу, поэтому воспользуюсь утилитой touch с ключом t, которая может изменить время модификации файла:

```
touch -t 202103081825 /backup/*tar.gz  
ls -l /backup/centos8*tar.gz
```

Как видите, время изменилось на 8 марта, это 5 дней назад.

```
[user@centos8 ~]$ find /backup -mtime +4 -ls  
34044081 181964 -rw-r--r-- 1 user user 186328806 Mar  8 18:25 /backup/centos8_13.03.2021_18.16.tar.gz  
[user@centos8 ~]$
```

Теперь воспользуемся find чтобы найти файлы старее 5 дней назад в директории /backup:

```
find /backup -mtime +4 -ls
```

Я пишу +4, потому что mtime начинает отсчёт от 0, и mtime +0 - это в предыдущие 24 часа, соответственно +4 - это старше 5 дней.

```
GNU nano 2.9.8                                mybackupscript

1 #!/bin/bash
2
3 BDIR=/backup
4 BDATE=$(date +'%d.%m.%Y_%H.%M')
5 FILENAME=$BDIR/$(hostname)_$BDATE
6
7 find $BDIR -mtime +4 -delete
8 tar -cvzf $FILENAME.tar.gz -C /home/user . &> $FILENAME.log
```

```
[user@centos8 ~]$ nano mybackupscript
[user@centos8 ~]$ ./mybackupscript
[user@centos8 ~]$ ls -l /backup/
total 730740
-rw-rw-r--. 1 user user 565624 Mar 13 18:11 centos8_13.03.2021_18.11.log
-rw-rw-r--. 1 user user 186329995 Mar 13 18:11 centos8_13.03.2021_18.11.tar.gz
-rw-r--r--. 1 user user 565624 Mar 13 18:16 centos8_13.03.2021_18.16.log
-rw-rw-r--. 1 user user 565624 Mar 13 18:40 centos8_13.03.2021_18.40.log
-rw-rw-r--. 1 user user 186326932 Mar 13 18:40 centos8_13.03.2021_18.40.tar.gz
-rw-rw-r--. 1 user user 565624 Mar 13 18:42 centos8_13.03.2021_18.42.log
-rw-rw-r--. 1 user user 186321438 Mar 13 18:42 centos8_13.03.2021_18.42.tar.gz
-rw-rw-r--. 1 user user 565675 Mar 13 18:51 centos8_13.03.2021_18.51.log
-rw-rw-r--. 1 user user 186446306 Mar 13 18:51 centos8_13.03.2021_18.51.tar.gz
```

Добавим это в наш скрипт, заменим ls на -delete и протестируем:

```
nano mybackupscript
```

```
find $BDIR -mtime +4 -delete
```

```
./mybackupscript
```

Как видите, бэкап, который был старее 5 дней, удалился.

И так, давайте подведём итоги. Сегодня мы с вами написали простенький скрипти, который бэкапит нашу домашнюю директорию. Однако целью был не сам скрипти, а утилиты, стоящие за его работой. Мы с вами познакомились с утилитами du, которая считает размер файлов, find, которая может находить файлы по разным критериям, tar, которая может архивировать файлы, gzip, которая может сжимать файлы, date, которая позволяет нам именовать наши файлы с нужным форматом времени, touch, которая позволяет поменять время изменения файла, чтобы мы могли провести тесты. Воспользовавшись всеми этими утилитами мы создали простенькое скрипти и добавили его в стоп, что даёт нам простой бэкап механизм с автоматическим созданием и удалением старых бэкапов.

2.38.2 Практика

Вопросы

1. Как посмотреть размер файлов и директорий?
2. Чем отличается ; от + у утилиты find?
3. Для чего нужна архивация? Какая утилита используется для архивации?
4. Для чего нужно сжатие? Какие утилиты используются для сжатия, и чем они отличаются?

Задания

1. Найдите самую большую директорию в корне. Затем найдите самую большую директорию/файл в этой директории. И так по цепочке.
2. Найдите все файлы в системе, которые принадлежат вашему пользователю.
3. Найдите все директории в системе, в имени которых встречается «.d» и сохраните список в файл.
4. Найдите все файлы в системе, у которых есть *suid*, и скопируйте их в директорию `~/suidfiles`.
5. Создайте архив со всеми файлами из директории `/var/log` с помощью gzip, а также с помощью bzip. Сравните размеры полученных архивов.

6. Создайте директорию exam. Внутри директории создайте файл myfile. Создайте жёсткую и символьские ссылки на этот файл. Переименуйте файл и символьскую ссылку, чтобы их названия совпадали с их инодами. Создайте сжатый архив со всей директорией. Удалите из архива символьскую ссылку.
7. Создайте пользователя backup и настройте для него cron, чтобы тот раз в день находил файлы старее 5 дней в директории /data и добавлял эти файлы в архив /backup/archive.tar, после чего удалял из исходной директории. Настройте cron у пользователя user, чтобы он каждые 10 минут создавал файл в директории /data. Также у пользователя root создайте cron, чтобы он каждую пятницу в 23:30 записывал содержимое директории /data и содержимое архива /backup/archive.tar в файл /var/log/reports/текущаядата.

2.39 39. Инкрементальные бэкапы с tar

2.39.1 39. Инкрементальные бэкапы с tar

```
[user@centos8 ~]$ du -h /backup/*.gz
178M    /backup/centos8_13.03.2021_20.51.tar.gz
178M    /backup/centos8_13.03.2021_20.52.tar.gz
178M    /backup/centos8_13.03.2021_20.53.tar.gz
```

В прошлый раз мы с вами настроили простой скрипт для бэкапа. Он будет делать полную копию нашей домашней директории и хранить её в сжатом виде. Такой тип бэкапа, когда копируется всё что нужно, называется полным бэкапом. Например, мы бэкапили домашнюю директорию:

```
du -h /backup/*.gz
```

и каждый раз копировалось всё - поэтому размеры файлов совпадают. Но, если подумать - за день никакие файлы не изменились, зачем нам ещё одна копия? Она занимает столько же места, а смысла от неё нет. Да и представьте, что у нас изменился один файл из 5 тысяч, а мы всё равно будем делать полный бэкап - копировать все 5 тысяч файлов. Мы просто теряем место.

```
[user@centos8 ~]$ du -h Pictures/
18M    Pictures/
[user@centos8 ~]$ tar -czf pictures.tar.gz -g pictures.snap -C Pictures/ .
[user@centos8 ~]$ du -h pictures.*
4.0K    pictures.snap
18M    pictures.tar.gz
[user@centos8 ~]$ █
```

Почему бы не бэкапить только изменённые файлы? Такой тип бэкапа называется инкрементальным. У нас есть один полный бэкап, а при следующем будут сохраняться только изменённые файлы. Давайте мы это протестируем. Возьмём какую-нибудь директорию, например, Pictures:

```
du -h Pictures
```

Для начала нужно создать полный бэкап. Но на этот раз нам понадобится дополнительный ключ - g - который указывает на файл с метаданными. По нему tar решает, какие файлы были изменены и в дальнейшем делает инкрементальный бэкап. Если файла с метаданными нету, то он делает полный бэкап и создаёт этот файл:

```
tar -czf pictures.tar.gz -g pictures.snar -C Pictures/ .
du -h pictures.*
```

Как видите, создался сжатый архив и файл snar – именно в нём хранятся метаданные.

```
[user@centos8 ~]$ tar -czf pictures2.tar.gz -g pictures.snar -C Pictures/ .
[user@centos8 ~]$ du -h pictures2.tar.gz
4.0K    pictures2.tar.gz
[user@centos8 ~]$ tar -tf pictures2.tar.gz
./
```

Теперь попытаемся создать ещё один бэкап:

```
tar -czf pictures2.tar.gz -g pictures.snar -C Pictures/ .
du -h pictures2.tar.gz
tar -tf pictures2.tar.gz
```

Архив создался, но он весит всего 4 килобайта и пустой. То есть, если у нас за день никаких изменений не будет, мы не потратим место на диске на новый бэкап.

```
[user@centos8 ~]$ tar -tf pictures2.tar.gz -g /dev/null -vv
drwxr-xr-x user/user        2666 2021-03-23 10:45 .
N Screenshot from 2020-05-30 15-48-27.png
N Screenshot from 2020-05-30 15-48-31.png
N Screenshot from 2020-05-30 17-05-15.png
N Screenshot from 2020-05-31 08-37-14.png
N Screenshot from 2020-06-14 18-20-24.png
N Screenshot from 2020-11-04 00-06-06.png
N Screenshot from 2020-11-04 00-06-09.png
```

Когда речь идёт об инкрементальных архивах, нужна опция `-g`, но для извлечения и просмотра нет нужды указывать файл snar, так как в самом архиве есть эта информация:

```
tar -tf pictures2.tar.gz -g /dev/null -vv
```

Поэтому вместо snar принято указывать `/dev/null`, а два `-vv` позволяют увидеть добавленные или не включённые в архив файлы.

```
[user@centos8 ~]$ cp /etc/passwd Pictures/
[user@centos8 ~]$ rm Pictures/Screenshot\ from\ 2020-0
Screenshot from 2020-05-30 15-48-27.png
Screenshot from 2020-05-30 15-48-31.png
Screenshot from 2020-05-30 17-05-15.png
Screenshot from 2020-05-31 08-37-14.png
Screenshot from 2020-06-14 18-20-24.png
[user@centos8 ~]$ rm Pictures/Screenshot\ from\ 2020-05-31\ 08-37-14.png
[user@centos8 ~]$ tar -czf pictures3.tar.gz -g pictures.snar -C Pictures/ .
[user@centos8 ~]$ du -h pictures3.tar.gz
4.0K    pictures3.tar.gz
[user@centos8 ~]$ tar -tf pictures3.tar.gz
./
./passwd
```

Давайте добавим в директорию Pictures новый файл и удалим какой-то из существующих:

```
cp /etc/passwd Pictures/  
rm Pictures/Screenshot
```

Потом создадим ещё один бэкап:

```
tar -czf pictures3.tar.gz -g pictures.snar -C Pictures/ .  
du -h pictures3.tar.gz  
tar -tf pictures3.tar.gz
```

Как видите, в новом архиве появился файл passwd.

```
[user@centos8 ~]$ tar -xf pictures.tar.gz -g /dev/null -C Pictures/ -v  
. /  
tar: Deleting './passwd'  
. /Screenshot from 2020-05-30 15-48-27.png  
. /Screenshot from 2020-05-30 15-48-31.png  
. /Screenshot from 2020-05-30 17-05-15.png  
. /Screenshot from 2020-05-31 08-37-14.png  
. /Screenshot from 2020-06-14 18-20-24.png  
. /Screenshot from 2020-11-04 00-06-06.png
```

Теперь рассмотрим процесс восстановления. Для этого нужно поочерёдно восстанавливать бэкапы до требуемого дня. Сперва восстановим исходный архив:

```
tar -xf pictures.tar.gz -g /dev/null -C Pictures -v
```

Как видите, он удалил файл passwd, так как его изначально не было.

```
[user@centos8 ~]$ tar -xf pictures3.tar.gz -g /dev/null -C Pictures/ -v  
. /  
tar: Deleting './Screenshot from 2020-05-31 08-37-14.png'  
. /passwd  
[user@centos8 ~]$ ls Pictures/passwd  
Pictures/passwd  
[user@centos8 ~]$
```

Смысла восстанавливать второй архив нет, так как там не было никаких изменений, сразу восстановим третий:

```
tar -xf pictures3.tar.gz -g /dev/null -C Pictures -v
```

Таким образом, у нас удалился тот самый скриншот и восстановился файл passwd.

```
[user@centos8 ~]$ du -h pictures*
4.0K    pictures2.tar.gz
4.0K    pictures3.tar.gz
4.0K    pictures.snap
18M     pictures.tar.gz
[user@centos8 ~]$ █
```

И при том, что я создал 3 бэкапа, инкрементальные копии практически ничего не весят:

```
du -h pictures*
```

Естественно, в реальных условиях они бы занимали больше места, и тем не менее, инкрементальные бэкапы позволяют во много раз сэкономить пространство.

```
TARGET=/home/user
BCUR=/backup/current
BOLD=/backup/old
BDATE=$(date +'%d.%m.%Y_%H.%M')
FILENAME=$(hostname)_$BDATE
EXPIRE=13
```

Теперь попытаемся научить наш скрипт делать инкрементальные бэкапы. Это немного усложнит его, потому что мы не можем просто удалять файлы недельной давности - инкрементальные копии зависят от полного бэкапа, без него они теряют смысл. Но так как мы сэкономили пространство, можем хранить бэкап с прошлой недели. Для начала, чтобы превратить наш скрипт в более универсальный, заменим /home/user на переменную:

```
TARGET=/home/user
```

разделим логи на ошибки и стандартный вывод:

```
> $FILENAME.log 2> FILENAME.err
```

заменим директорию BDIR на две - /backup/current и /backup/old, немного подкорректируем FILENAME:

```
FILENAME=$(hostname)_$BDATE
```

уберём отсюда директорию, а также добавим переменную EXPIRE, по которой и будем удалять старые бэкапы - EXPIRE=13.

```
backup() {
tar -g $BCUR/snar -czvf $BCUR/$FILENAME.tar.gz -C $TARGET . > $BCUR/$FILENAME.log
2> $BCUR/$FILENAME.err
}
delete() {
}
full() {
}
```

Так как удалять по старости бэкапа мы не можем, иначе сотрём полный бэкап, стоит придумать другую схему. Например, раз в неделю архивировать бэкап прошлой недели и создавать новый полный бэкап. Чтобы не решать это в скрипте, а сделать через планировщик задач, в самом скрипте я добавлю 3 функции - backup() {}, delete() {} и full() {}. В функцию backup добавлю нашу старую команду по бэкапу, подкорректированную по директориям и с опцией -g:

```
backup() {
tar -g $BCUR/snar -czvf $BCUR/$FILENAME.tar.gz -C $TARGET . >
$BCUR/$FILENAME.log 2> $BCUR/$FILENAME.err
}
```

```
full() {
tar -czvf $BOLD/$FILENAME.full.tar.gz -C $BCUR . > $BOLD/$FILENAME.full.log
2> $BOLD/$FILENAME.full.err
rm -f $BCUR/*
backup
}
```

Когда будем делать полный бэкап, скрипт заархивирует содержимое директории /backup/current и сохранит его в директории /backup/old. После чего удалит содержимое current директории и создаст новый бэкап, запуская готовую функцию backup:

```
full() {
tar -czvf $BOLD/$FILENAME.full.tar.gz -C $BCUR . >
$BOLD/$FILENAME.full.log 2> $BOLD/$FILENAME.full.err rm -f $BCUR/*
backup
}
```

```
delete() {
find $BOLD/ -mtime +$EXPIRE -delete
}
```

А при необходимости старые бэкапы будут удаляться из директории /backup/old, используя переменную EXPIRE:

```
delete() {
find $BOLD/ -mtime +$EXPIRE -delete
}
```

```
case $1 in
    incremental) backup ;;
    full) full ;;
    delete) delete ;;
    *) echo "Options: incremental, full, delete" ;;
esac
```

Но пока мы только задали функции, их нужно ещё запускать. Для этого мы будем передавать скрипту аргумент - \$1 - и на основе него запускать ту или иную функцию:

```
case $1 in
    incremental) backup ;;
    full) full ;;
    delete) delete ;;
    *) echo "Options: incremental, full, delete" ;;
esac
```

```
[user@centos8 ~]$ rm -rf /backup/*
[user@centos8 ~]$ mkdir /backup/old /backup/current
[user@centos8 ~]$ ./mybackupscript
Options: incremental, full, delete
[user@centos8 ~]$ ./mybackupscript incremental
[user@centos8 ~]$ ls /backup/current/
centos8_23.03.2021_15.31.err  centos8_23.03.2021_15.31.tar.gz
centos8_23.03.2021_15.31.log  snar
[user@centos8 ~]$
```

Теперь давайте проверим работу скрипта. Для начала очистим содержимое директории /backup:

```
rm -rf /backup/*
```

и создадим необходимые директории:

```
mkdir /backup/old /backup/current
```

Попробуем просто запустить скрипт:

```
./mybackupscript
```

Как видите, он выдаёт информацию об опциях. В нашем случае в первый раз лучше запускать incremental, чтобы не архивировать пустую директорию:

```
./mybackupscript incremental
```

После чего можем проверить содержимое директории /backup/current:

```
ls /backup/current/
```

```
[user@centos8 ~]$ head -5 /backup/current/*{err,log}
==> /backup/current/centos8_23.03.2021_15.31.err <==
tar: .: Directory is new
tar: ./cache: Directory is new
tar: ./config: Directory is new
tar: ./local: Directory is new
tar: ./mozilla: Directory is new

==> /backup/current/centos8_23.03.2021_15.31.log <==
./
./cache/
./cache/evolution/
./cache/evolution/WebKitCache/
./cache/evolution/WebKitCache/Version 14/
[user@centos8 ~]$ █
```

Если посмотреть лог ошибок:

```
head -5 /backup/current/*{err,log}
```

можно увидеть, что в ошибки попадают записи «Directory is new». Это не совсем ошибки, просто информация о том, что при прошлом бэкапе этих директорий не было. Но это логично, потому что у нас прошлого бэкапа и не было.

```
[user@centos8 ~]$ touch file100 file101 file102
[user@centos8 ~]$ ./mybackupscrip incremental
[user@centos8 ~]$ head -5 /backup/current/centos8_23.03.2021_15.
centos8_23.03.2021_15.31.err      centos8_23.03.2021_15.41.err
centos8_23.03.2021_15.31.log      centos8_23.03.2021_15.41.log
centos8_23.03.2021_15.31.tar.gz   centos8_23.03.2021_15.41.tar.gz
[user@centos8 ~]$ head -5 /backup/current/centos8_23.03.2021_15.41.err
[user@centos8 ~]$ head -5 /backup/current/centos8_23.03.2021_15.41.log
./
./cache/
./cache/evolution/
[user@centos8 ~]$ tail -8 /backup/current/centos8_23.03.2021_15.41.log
./this/
./file100
./file101
./file102
./cache/tracker/meta.db
./cache/tracker/meta.db-shm
./cache/tracker/meta.db-wal
./local/share/tracker/data/tracker-store.journal
[...]
```

Создадим пару файлов:

```
touch file100 file101 file102
```

и запустим скрипт заново:

```
./mybackupscript incremental
```

В err файле больше ошибок нет - потому что нет новых директорий. А в логах перечислены все директории, а также новые и изменённые файлы.

```
[user@centos8 ~]$ ./mybackupscript full
[user@centos8 ~]$ ls /backup/current/
centos8_23.03.2021_15.49.err  centos8_23.03.2021_15.49.tar.gz
centos8_23.03.2021_15.49.log  snar
[user@centos8 ~]$ ls /backup/old/
centos8_23.03.2021_15.49.full.err  centos8_23.03.2021_15.49.full.tar.gz
centos8_23.03.2021_15.49.full.log
[user@centos8 ~]$ cat /backup/old/centos8_23.03.2021_15.49.full.log
.-
./centos8_23.03.2021_15.31.log
./centos8_23.03.2021_15.31.err
./snar
./centos8_23.03.2021_15.31.tar.gz
./centos8_23.03.2021_15.41.log
./centos8_23.03.2021_15.41.err
./centos8_23.03.2021_15.41.tar.gz
```

Теперь попробуем запустить полный бэкап:

```
./mybackupscript full
```

Как видите:

```
ls /backup/current
```

создались новые файлы, а старые заархивировались:

```
ls /backup/old
cat /backup/old/*full.log
```

```
[user@centos8 ~]$ crontab -l
0 0 * * * /home/user/mybackupscript delete
50 23 * * 1-6 /home/user/mybackupscript incremental
50 23 * * 0 /home/user/mybackupscript full
[user@centos8 ~]$
```

Осталось настроить cron:

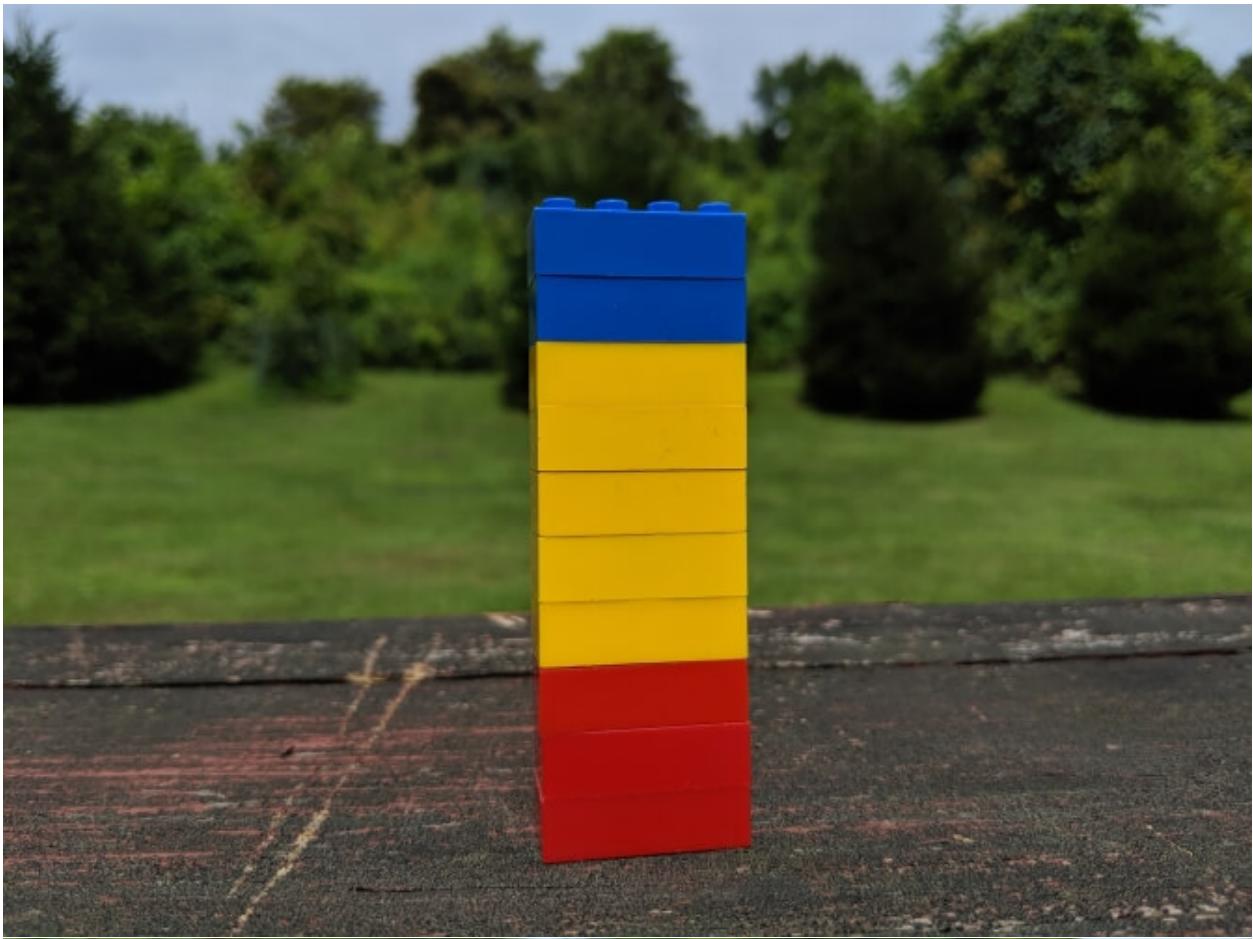
```
crontab -e
crontab -l
```

Пусть каждую ночь find ищет и удаляет старые бэкапы старее 14 дней. С понедельника по субботу пусть выполняется инкрементальное резервное копирование, а в воскресенье полное.

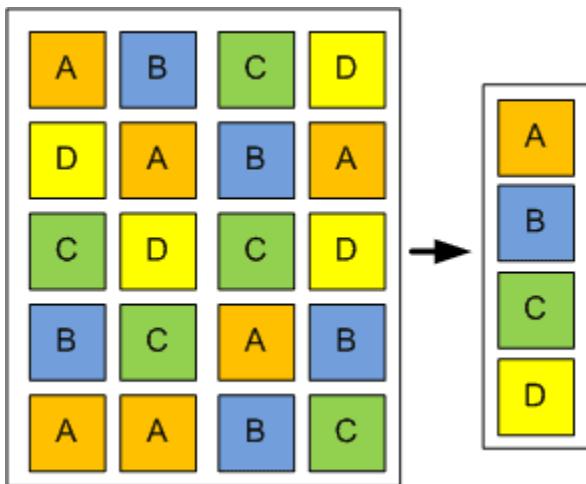
Подведём итоги. Сегодня мы с вами разобрали, что такое полные и инкрементальные бэкапы, зачем они нужны, научились ими пользоваться, а также добавили их в наш скрипт. Это позволяет значительно снизить занимаемое пространство, делать больше копий и в целом делать их быстрее. Но при этом бэкапы становятся зависимы друг от друга, недостаточно просто распаковать один архив, иногда нужно это делать по порядку их создания. Но сэкономленное пространство этого стоит.

2.40 40. Дедупликация с VDO

2.40.1 40. Дедупликация с VDO

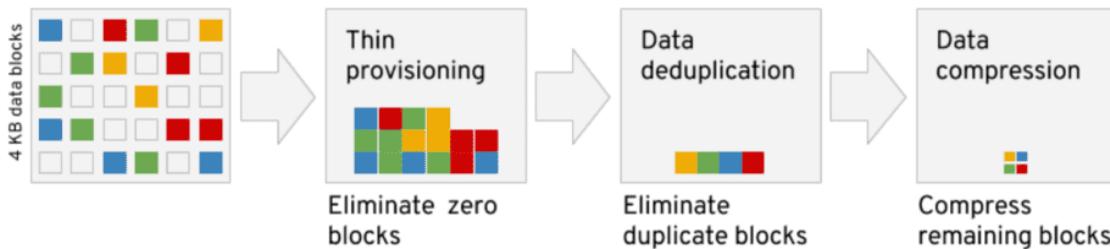


Недавно мы с вами разбирали утилиту gzip, которая позволяет сэкономить пространство с помощью сжатия данных и реализовали это для бэкапа домашней директории. gzip работает с файлами и на этих картинках показан простой пример. Но у такого сжатия есть пара нюансов - со сжатыми файлами работать не получится, их нужно предварительно разжимать. Точнее, есть всякие утилиты, которые позволяют работать с сжатыми файлами, но это не годится для всего. Да и сжимается не абсолютно всё, а только то, что вы укажете.



Однако есть механизм, который позволяет экономить пространство ещё на уровне блоков данных - дедупликация. В целом механизм напоминает сжатие, но работает не на уровне файлов и алгоритмов сжатия, а на уровне блоков - дублирующиеся блоки заменяются ссылками на один. Очень яркий пример использования - если у вас множество виртуальных машин. Представьте, что вы подняли 100 виртуалок с Centos-ом, каждая выполняет свою задачу. На всех одна и та же операционная система, ядро, библиотеки и куча утилит - всё это одно и тоже, дублирующееся на сотни виртуалок. Файлы работающих виртуалок не сожмёшь с помощью gzip-а, да и эффективность от такого сжатия будет сомнительная. А вот дедупликация просто увидит кучу одинаковых блоков и заменит их ссылками, что значительно сэкономит пространство. Минусы, конечно, тоже есть - дедупликация постоянно работает, из-за чего увеличивается использование оперативной памяти и диска, соответственно, где-то проседает скорость работы. Т.е. на домашних компьютерах это не так нужно, в рабочей среде зависит от конкретной задачи.

VDO data reduction processing



С недавних пор у RedHat появилась технология VDO - virtual data optimizer. Она позволяет экономить пространство диска, при этом жертвуя ресурсами процессора и оперативки. И кроме дедупликации она избавляется от пустых блоков, в которых одни нули, а также сжимает итоговые данные алгоритмом LZ4. Также она обеспечивает thin provisioning - т.е. когда вы компьютеру выделили, скажем, 50 гигабайт, а операционной системе говорите, что там 500.

Зачем это нужно? Представьте, что вам нужно поднять несколько виртуальных машин и каждой дать по 50 гигабайт, на будущее, потому что количество данных будет расти. Но, во первых, сейчас каждая

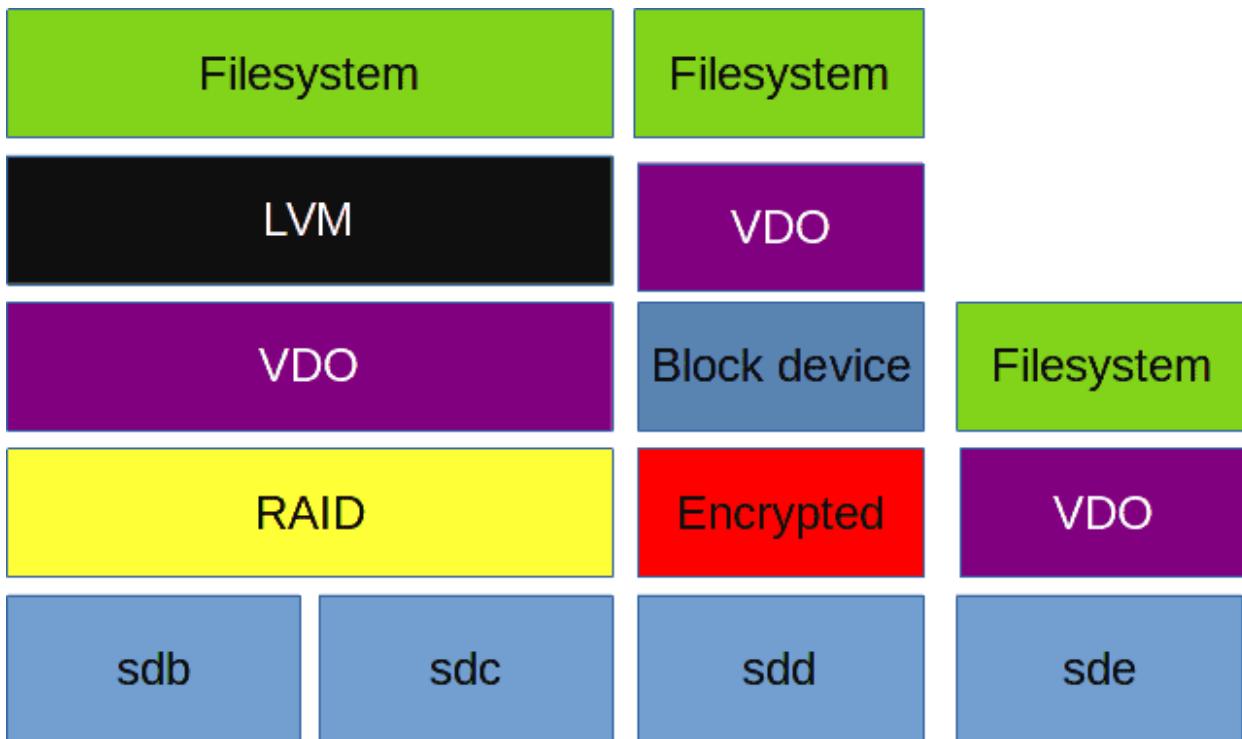
виртуалка будет занимать не больше 5 гигабайт, а во вторых у вас сейчас нет таких больших дисков. Будут расти данные - вы добавите новые диски. Если вы сейчас выделите в виртуалках всего по 5 гигабайт, то вам постоянно нужно будет в них увеличивать пространство по мере добавления дисков. А с thin provisioning-ом вам не нужно это делать - виртуалка будет занимать столько места, сколько ей нужно на самом деле, а вам просто нужно следить, чтобы выделенный вами диск не закончился. Если закончится - будет плохо, потому что виртуалки будут видеть свободное пространство, но не смогут туда писать, потому что реального пространства больше нет. И это приведёт к проблемам. Но если быть внимательным и до этого не доводить - всё будет хорошо.

The following data was collected using spinning disks as backend, on a system with 32 GB RAM and 12 cores at 2.4Ghz.

Column 'deploy to file system' shows the time it took to copy a directory with ~5GB of data from RAM (/dev/shm) to the filesystem. Column 'copy on file system' shows the time required to make a single copy of the 5GB directory which was deployed in the first step. The time was averaged over multiple runs, always after removing the data and emptying the file system caches. Publicly available texts from the [Project Gutenberg](#) were used as data, quite nicely compressible.

filesystem backend	deploy to file system	copy on file system
XFS ontop of normal LVM volume	28sec	35sec
XFS on VDO device, async mode	55sec	58sec
XFS on VDO device, sync mode	71sec	92sec

Ну и стоит упомянуть падение производительности при использовании VDO. По скрину видно, что просадка довольно таки значительная, если без VDO с XFS на LVM копирование 5 гигабайт занимает 28 секунд, то с VDO в асинхронном режиме - 55 секунд, а в синхронном и вовсе 71. При синхронном режиме VDO ждёт ответа от диска о том, что данные на него записались. Но суть в том, что диск может немного соврать об этом, потому что он может перед записью использовать кэш. Только вот если сервер вдруг вырубится, то данные в кэше пропадут. Тут, конечно, есть нюансы - кэш может быть с батарейками или диск со сквозным кэшем - но, в любом случае, sync стоит ставить только если вы уверены в том, что делаете. При асинхронном режиме VDO не гарантирует файловой системе или приложению запись на диск, тем самым они сами заботятся о сохранности. По умолчанию же используется режим auto, который сам подбирает нужный режим в зависимости от накопителя.



Сам VDO ставится поверх блочного устройства, ниже файловой системы и LVM. Если у вас RAID - VDO нужно ставить поверх рейда. Если зашифрованный раздел - VDO ставится поверх него, иначе он не увидит что там дедуплицировать и сжимать. У VDO есть ряд требований к размеру диска и оперативке, с которыми вы можете ознакомиться по [ссылке](#). Например, для диска размером до терабайта нужно примерно 700 мегабайт оперативки под задачи VDO и 2.5 гигабайта пространства под метаданные.

```
[user@centos8 ~]$ sudo fdisk -l /dev/sdd
Disk /dev/sdd: 6 GiB, 6442450944 bytes, 12582912 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
[user@centos8 ~]$ sudo dnf install vdo kmod-kvdo -y
Last metadata expiration check: 0:14:51 ago on Fri 26 Mar 2021 03:53:34 PM +04.
Package vdo-6.2.3.114-14.el8.x86_64 is already installed.
Package kmod-kvdo-6.2.3.114-74.el8.x86_64 is already installed.
```

Для теста я добавил нашей виртуалке ещё один диск на 6 гигабайт - минимально необходимое пространство для VDO это почти 5 гигабайт:

```
sudo fdisk -l /dev/sdd
```

Также нам понадобится установить утилиту и модуль ядра для работы с vdo:

```
sudo dnf install vdo kmod-kvdo
```

```
[user@centos8 ~]$ ls -l /dev/disk/by-id | grep sdd
lrwxrwxrwx. 1 root root 9 Mar 26 16:05 ata-VBOX_HARDDISK_VB8210cd9d-e92a63b6 -> ../
.../sdd
[user@centos8 ~]$ ls /dev/disk/by-id/ata-VBOX_HARDDISK_VB8210cd9d-e92a63b6
/dev/disk/by-id/ata-VBOX_HARDDISK_VB8210cd9d-e92a63b6
[user@centos8 ~]$
```

И так, у нас есть диск sdd, на котором требуется развернуть VDO. Но, как мы обсуждали, имя sdd динамически выдаётся udev-ом при включении, т.е. может поменяться при определённых условиях. VDO предпочитает постоянные имена. Попробуем найти sdd по идентификатору:

```
ls -l /dev/disk/by-id/ | grep sdd
```

Это - идентификатор диска, символьская ссылка, которая постоянно будет вести на нужное устройство, поэтому используем его. Хотя VDO и сам делает попытку найти нужную символьскую ссылку, тем не менее мы это сделали за него, чтобы было нагляднее.

```
[user@centos8 ~]$ sudo vdo create --name myvdo --device /dev/disk/by-id/ata-VBOX_HARDDISK_VB8210cd9d-e92a63b6 --vdoLogicalSize=12G
Creating VDO myvdo
vdo: ERROR - Kernel module kvdo not installed
vdo: ERROR - modprobe: FATAL: Module kvdo not found in directory /lib/modules/4.18.0-147.8.1.el8_1.x86_64
[user@centos8 ~]$ █
```

При создании VDO указываем желаемое имя --name myvdo, указываем на устройство --device и указываем логический размер - то что я говорил про thin provisioning:

```
sudo vdo create --name myvdo --device /dev/disk/by-id/ata-... --vdoLogicalSize=12G
```

Т.е. физический диск у меня 6 гигабайт, а файловая система будет видеть все 12. Но команда у меня не сработала - как видите, вышла ошибка «Kernel module not installed», а дальше нам показывают, что ядро не нашло модуля в директории /lib/modules/4.18.0-147.

```
[user@centos8 ~]$ find /lib/modules -name vdo
/lib/modules/4.18.0-240.15.1.el8_3.x86_64/weak-updates/kmod-kvdo/vdo
/lib/modules/4.18.0-240.10.1.el8_3.x86_64/extr/kmod-kvdo/vdo
[user@centos8 ~]$ ls /boot/vmlinuz-4.18.0-*
/boot/vmlinuz-4.18.0-147.8.1.el8_1.x86_64 /boot/vmlinuz-4.18.0-240.15.1.el8_3.x86_64
/boot/vmlinuz-4.18.0-147.el8.x86_64
[user@centos8 ~]$ uname -r
4.18.0-147.8.1.el8_1.x86_64
[user@centos8 ~]$ reboot
```

Но так как установка прошла успешно, попытаемся найти модуль вручную:

```
find /lib/modules -name vdo
```

Как видите, find нашла две директории с более новыми версиями ядра - 240.10 и 240.15 и в них эти модули. Скорее всего установка модуля подтянула новую версию ядра и в директории /boot как раз есть новые версии:

```
ls /boot/vmlinuz-*
```

На всякий случай проверим текущую версию ядра:

```
uname -r
```

147 - значит нам нужно просто перезагрузиться:

```
reboot
```

чтобы загрузиться с новым ядром. После перезагрузки слетели драйвера виртуалбокса, пришлось переустанавливать их для новой версии - сначала ставим необходимые пакеты:

```
sudo dnf install kernel-headers kernel-devel
```

после чего с диска запускаем установку гостевых дополнений. Потом опять перезагружаемся.

```
[user@centos8 ~]$ sudo vdo create --name myvdo --device /dev/disk/by-id/ata-VBOX_HARDDISK_VB8210cd9d-e92a63b6 --vdoLogicalSize=12G
Creating VDO myvdo
    The VDO volume can address 2 GB in 1 data slab.
    It can grow to address at most 16 TB of physical storage in 8192 slabs.
    If a larger maximum size might be needed, use bigger slabs.
Starting VDO myvdo
Starting compression on VDO myvdo
VDO instance 0 volume is ready at /dev/mapper/myvdo
[user@centos8 ~]$ █
```

Пробуем команду опять:

```
sudo vdo create --name myvdo --device /dev/disk/by-id/ata-... --vdoLogicalSize=12G
```

и всё получается. Утилита нам говорит, что создала slab-ы - т.е. блоки в понимании vdo размером в 2 гигабайта. И с таким размером у нас получится использовать только 16 терабайт пространства, а для больших объёмов требуется указать размер слабов больше. Максимумом являются slab-ы размером 32 гигабайта и пространство объёмом 256 терабайт.

```
[user@centos8 ~]$ sudo mkfs.xfs /dev/mapper/myvdo
meta-data=/dev/mapper/myvdo      isize=512    agcount=4, agsize=786432 blks
                                =           sectsz=4096  attr=2, projid32bit=1
                                =           crc=1        finobt=1, sparse=1, rmapbt=0
                                =           reflink=1
data     =           bsize=4096   blocks=3145728, imaxpct=25
                                =           sunit=0    swidth=0 blks
naming   =version 2            bsize=4096   ascii-ci=0, ftype=1
log       =internal log        bsize=4096   blocks=2560, version=2
                                =           sectsz=4096  sunit=1 blks, lazy-count=1
realtime =none                 extsz=4096   blocks=0, rtextents=0
```

Давайте сразу на него запишем файловую систему:

```
sudo mkfs.xfs /dev/mapper/myvdo
```

```
[user@centos8 ~]$ sudo blkid /dev/mapper/myvdo
/dev/mapper/myvdo: UUID="124403d9-8b46-48f2-8769-f026c5c218ed" TYPE="xfs"
[user@centos8 ~]$ sudo nano /etc/fstab
[user@centos8 ~]$ tail -1 /etc/fstab
UUID="124403d9-8b46-48f2-8769-f026c5c218ed" /backups xfs defaults,x-systemd.requires=vdo.service 0 0
[user@centos8 ~]$ █
```

После чего выясним UUID:

```
sudo blkid /dev/mapper/myvdo
```

и добавим строчку в fstab. Обратите внимание на опцию монтирования:

```
x-systemd.requires=vdo.service
```

Она нужна, чтобы монтировать эту файловую систему только если сервис vdo запущен.

```
[user@centos8 ~]$ sudo mkdir /backups
[user@centos8 ~]$ sudo mount /backups
[user@centos8 ~]$ df -h /backups
Filesystem      Size  Used Avail Use% Mounted on
/dev/mapper/myvdo   12G  118M   12G  1% /backups
[user@centos8 ~]$ █
```

Попробуем создать директорию и примонтировать:

```
sudo mkdir /backups
sudo mount /backups
df -h /backups
```

Как видите, всё примонтировалось.

```
[user@centos8 ~]$ sudo vdostats myvdo --human-readable
Device           Size     Used Available Use% Space saving%
myvdo          6.0G    4.0G     2.0G  66%    98%
[user@centos8 ~]$ █
```

Теперь давайте протестируем. Например, сделаем несколько бэкапов домашней директории. Но перед этим проверим статистику:

```
sudo vdostats myvdo --human-readable
```

Несмотря на то, что пока здесь нет никаких данных, vdo уже зарезервировал 4 гигабайта места для информации о дедупликации. Поэтому нам остаётся всего 2 гигабайта, хотя df нам показал все 12.

```
[user@centos8 ~]$ sudo vdo
activate          disableDeduplication  list          status
changeWritePolicy  enableCompression    modify        stop
create            enableDeduplication  printConfigFile
deactivate         growLogical        remove
disableCompression  growPhysical      start
[user@centos8 ~]$ sudo vdo █
```

Но, как я уже говорил, нужно быть осторожным и не допускать заполнения этих двух гигабайт, либо увеличивать размер диска под vdo. Вообще у VDO множество возможностей - останавливать или запускать дедупликацию и компрессию, менять размер логического и физического диска, менять тип записи с синхронного на асинхронный и наоборот и т.д. Это вполне понятные ключи, но советую быть осторожным и предварительно прочесть [документацию](#), прежде чем выполнять эти операции.

```
[user@centos8 ~]$ sudo tar -czf /backups/user.tar.gz /home/user
tar: Removing leading `/' from member names
[user@centos8 ~]$ sudo vdostats myvdo --hu
Device          Size      Used Available Use% Space saving%
myvdo           6.0G     4.2G    1.8G   69%      5%
[user@centos8 ~]$ sudo tar -czf /backups/user2.tar.gz /home/user
tar: Removing leading `/' from member names
[user@centos8 ~]$ sudo vdostats myvdo --hu
Device          Size      Used Available Use% Space saving%
myvdo           6.0G     4.2G    1.8G   69%     48%
[user@centos8 ~]$ du -h /backups/*
179M  /backups/user2.tar.gz
179M  /backups/user.tar.gz
[user@centos8 ~]$
```

И так, давайте приступим. Для начала создадим один бэкап:

```
sudo tar -czf /backups/user.tar.gz /home/user
```

и посмотрим статистику:

```
sudo vdostats myvdo --hu
```

Как видите, почти 200 мегабайт потратилось под бэкап. Теперь попробуем сделать ещё один бэкап:

```
sudo tar -czf /backups/user2.tar.gz /home/user
```

Проверим ещё раз статистику - а там всё равно 200 мегабайт, при этом обратите внимание, как сильно выросло значение Space saving. В итоге у нас получилось два файла, вместе они занимают почти 400 мегабайт:

```
du -h /backups/*
```

но благодаря vdo второй файл практически не занимает пространства. Да, в реальных условиях вряд ли все файлы будут одинаковы, но так как дедупликация работает на блочном уровне, в определённых случаях получится немало сэкономить.

```
[user@centos8 ~]$ sudo vdo status -n myvdo
VDO status:
  Date: '2021-03-26 17:10:57+04:00'
  Node: centos8
Kernel module:
  Loaded: true
  Name: kvdo
  Version information:
    kvdo version: 6.2.3.114
Configuration:
  File: /etc/vdoconf.yml
  Last modified: '2021-03-26 16:11:12'
VDOs:
  myvdo:
    Acknowledgement threads: 1
    Activate: enabled
[user@centos8 ~]$ sudo vdo status -n myvdo | tail -1
      write policy: async
[user@centos8 ~]$ █
```

Более детальную информацию можно узнать с помощью ключа status:

```
sudo vdo status -n myvdo
```

например, о типе записи.

```
VDO(8)                               System Manager's Manual                VDO(8)
NAME
  vdo - manage kernel VDO devices and related configuration information
SYNOPSIS
  vdo { activate | changeWritePolicy | create | deactivate | disableCompression |
         disableDeduplication | enableCompression | enableDeduplication | growLogical |
         growPhysical | import | list | modify | printConfigFile | remove | start | status |
         stop | version } [ options... ]
DESCRIPTION
  The commands available are:
```

Напоследок, посоветую не забывать пользоваться man, где есть и примеры создания, и то что нужно указывать в fstab и информация по многим ключам.

Подведём итоги. Сегодня мы с вами разобрали VDO - технологию для дедупликации и сжатия данных. Её легко развернуть, но нужно быть внимательным и учитывать требования по оперативке, по занимаемому пространству и падение производительности. Это не так эффективно при небольших объемах данных, но для определенных задач, например, для бэкап хранилища или хранилища для

дисков виртуальных машин, эта технология незаменима.

2.41 41. Создание systemd юнитов

2.41.1 41. Создание systemd юнитов

```
[user@centos8 ~]$ crontab -l
52 20 * * * /home/user/mybackupscript
[user@centos8 ~]$ tail -1 /etc/fstab
UUID="0711d5ea-24fc-4e86-b409-a28fd36e8aaa" /backups xfs defaults,x-systemd.requires=vdo.
service 0 0
[user@centos8 ~]$ cat mybackupscript
#!/bin/bash

BDIR=/backup
BDATE=$(date +'%d.%m.%Y_%H.%M')
FILENAME=$BDIR/$(hostname)_$BDATE

find /backup/ -mtime +4 -delete
tar -czvf $FILENAME.tar.gz -C /home/user . &> $FILENAME.log
```

За последние пару тем мы с вами написали бэкап скрипт, автоматизировали его через cron, а также создали раздел для бэкапов с VDO:

```
crontab -l
tail -1 /etc/fstab
```

Этот бэкап стал частью системы - он периодически выполняется, использует какие-то ресурсы, связан с сервисом vdo и всё такое. Почему бы нам не связать его с systemd? Зачем? Есть несколько преимуществ.

Например, управление. Что мне нужно, чтобы перестать делать бэкапы? Редактировать cron, убрать или закомментировать строчку. А в случае с systemd я могу просто отключить unit - одна простая команда. В принципе, и то и другое не сложно, когда один компьютер. А если у вас компьютеров много? Или, скажем, нужно задачу автоматизировать? Всегда легче выполнить команду, нежели редактировать файл.

Другой пример - связанность с другими задачами. Допустим, у вас есть антивирус, который раз в день проверяет систему. И вы хотите делать бэкап только после проверки антивирусом. Обычно cron запускает задачу в указанный срок, независимо от других задач. Вы, конечно, можете в самом скрипте это как-то связать, но это будет костылём. А systemd позволит вам связать различные сервисы, выстроить зависимости.

Ещё один пример - файловая система, которая монтируется в /backup. При текущем конфиге fstab она монтируется при запуске системы. Но, по хорошему, в большую часть времени эта файловая система не нужна, и, если вдруг что-то произойдёт с системой, кто-то там сделает rm -rf или запустит шифровальщик, то все наши бэкапы полетят. Почему бы не держать эту файловую систему в отмонтированном состоянии и подключать только при необходимости, когда мы делаем бэкап? Да, конечно, это можно сделать и через сам скрипт, и через cron, только это будет костылём - у нас либо скрипт перестанет быть универсальным, либо придётся через cron. В принципе, cron не плохой вариант, но, как я говорил выше, командами всё же предпочтительнее.

1. **Service units**, which start and control daemons and the processes they consist of. For details, see **systemd.service(5)**.
2. **Socket units**, which encapsulate local IPC or network sockets in the system, useful for socket-based activation. For details about socket units, see **systemd.socket(5)**, for details on socket-based activation and other forms of activation, see **daemon(7)**.
3. **Target units** are useful to group units, or provide well-known synchronization points during boot-up, see **systemd.target(5)**.
4. **Device units** expose kernel devices in systemd and may be used to implement device-based activation. For details, see **systemd.device(5)**.
5. **Mount units** control mount points in the file system, for details see **systemd.mount(5)**.
6. **Automount units** provide automount capabilities, for on-demand mounting of file systems as well as parallelized boot-up. See **systemd.automount(5)**.
7. **Timer units** are useful for triggering activation of other units based on timers. You may find details in **systemd.timer(5)**.

И так, что мы сделаем:

```
man systemd
```

/ Service units

Создадим пару различных юнитов systemd: во-первых - сервис, который будет запускать наш скрипт для бэкапа; во-вторых - таймер, который будет задавать периодичность бэкапа; в-третьих - mount, который будет монтировать файловую систему /backup; в четвёртых - автомаунт, который будет запускать третий пункт при необходимости, а не при запуске компьютера; в-пятых - таргет - чтобы объединить несколько сервисов в одну группу. Остальные типы юнитов обычно трогать не надо, их настройка нужна для более редких задач.

```
[user@centos8 ~]$ cd /etc/systemd/system/  
[user@centos8 system]$ systemctl list-units --type=mount  


| UNIT                                 | LOAD   | ACTIVE | SUB     | DESCRIPTION                     |
|--------------------------------------|--------|--------|---------|---------------------------------|
| - .mount                             | loaded | active | mounted | Root Mount                      |
| backups.mount                        | loaded | active | mounted | /backups                        |
| boot.mount                           | loaded | active | mounted | /boot                           |
| dev-hugepages.mount                  | loaded | active | mounted | Huge Pages File System          |
| dev-mqueue.mount                     | loaded | active | mounted | POSIX Message Queue File System |
| mydata.mount                         | loaded | active | mounted | /mydata                         |
| run-media-user-VBox_GAs_6.1.16.mount | loaded | active | mounted | /run/media/user/VBox_GAs_6.1.16 |
| run-user-1000-gvfs.mount             | loaded | active | mounted | /run/user/1000/gvfs             |
| run-user-1000.mount                  | loaded | active | mounted | /run/user/1000                  |
| run-user-42.mount                    | loaded | active | mounted | /run/user/42                    |
| sys-fs-fuse-connections.mount        | loaded | active | mounted | FUSE Control File System        |


```

Свои unit-ы желательно создавать в директории /etc/systemd/system:

```
cd /etc/systemd/system
```

Это просто текстовые файлы с определённым расширением и синтаксисом. Для начала создадим mount юнит, так как он нам пригодится в дальнейшем. Обычно имя mount юнита совпадает с путём монтирования и systemd сам генерирует эти юниты на основе fstab-а:

```
systemctl list-units --type mount
```

```
[user@centos8 system]$ systemctl cat backups.mount
# /run/systemd/generator/backups.mount
# Automatically generated by systemd-fstab-generator

[Unit]
SourcePath=/etc/fstab
Documentation=man:fstab(5) man:systemd-fstab-generator(8)
Before=local-fs.target
After=vdo.service
Requires=vdo.service

[Mount]
Where=/backups
What=/dev/disk/by-uuid/0711d5ea-24fc-4e86-b409-a28fd36e8aaa
Type=xfs
Options=defaults,x-systemd.requires=vdo.service
[user@centos8 system]$ sudo nano backups.mount
```

Давайте просто возьмём готовый юнит и используем его:

```
systemctl cat backups.mount
```

Скопируем всё что здесь есть, создадим файл с таким же названием:

```
sudo nano backups.mount
```

и вставим сюда скопированное. Я собираюсь убрать эту строчку из fstab, чтобы только systemd управлял этой файловой системой, поэтому немного подредактируем файл.

GNU nano 2.9.8	backups.mount
[Unit]	
Description=Mount for backups	
Before=local-fs.target	
After=vdo.service	
Requires=vdo.service	
[Mount]	
Where=/backups	
What=/dev/disk/by-uuid/0711d5ea-24fc-4e86-b409-a28fd36e8aaa	
Type=xfs	
Options=defaults	

Уберём комментарии в начале. Сперва идёт секция [Unit] - она есть во всех юнитах systemd и в ней указывается описание юнита и зависимости от других сервисов. Убираем лишнее - SourcePath и Documentation - они указывают на то, что этот юнит был сгенерирован systemd, что не актуально для нашей ситуации. Добавим Description=Mount for backups. Ниже указаны зависимости - Before, After и Requires. Before и After говорят о том, что данный mount должен быть запущен до local-fs таргета и после vdo.service, т.е. эти опции определяют порядок запуска юнитов. А Requires говорит о

тот, что для работы этого юнита необходим также сервис vdo, если тот сервис не запущен или его не получается запустить, то не нужно запускать этот mount.

В секции [Mount] всё предельно понятно: Where - куда монтировать; What - какую файловую систему; Type - тип этой файловой системы; Options - опции монтирования, всё как в fstab. Опция x-systemd.requires больше не нужна, так как она нужна была для генерации вышеуказанных зависимостей от vdo. Это нужно для fstab-а, а мы сразу создаём mount через systemd. На этом сохраняем файл и выходим.

```
[user@centos8 system]$ sudo nano /etc/fstab
[sudo] password for user:
[user@centos8 system]$ tail -1 /etc/fstab
#UUID="0711d5ea-24fc-4e86-b409-a28fd36e8aaa" /backups xfs defaults,x-systemd.requires=vdo.
service 0 0
[user@centos8 system]$ sudo umount /backups
```

Но, прежде чем пойти дальше, нужно убрать или закомментировать строчку в fstab и отмонтировать файловую систему:

```
sudo nano /etc/fstab
tail -1 /etc/fstab
sudo umount /backups
```

```
[user@centos8 system]$ sudo systemctl daemon-reload
[user@centos8 system]$ systemctl cat backups.mount
# /etc/systemd/system/backups.mount
[Unit]
Description=Mount for backups
Before=local-fs.target
After=vdo.service
Requires=vdo.service

[Mount]
Where=/backups
What=/dev/disk/by-uuid/0711d5ea-24fc-4e86-b409-a28fd36e8aaa
Type=xfs
Options=defaults
```

Чтобы systemd перечитал свои файлы и увидел наш mount файл, нужно запустить:

```
sudo systemctl daemon-reload
```

Проверим, увидел ли он наши изменения:

```
systemctl cat backups.mount
```

Да, это наш изменённый файл.

```
[user@centos8 system]$ df -h /backups
Filesystem Size Used Avail Use% Mounted on
/dev/mapper/cl_centos8-root 17G 7.9G 9.2G 46% /
[user@centos8 system]$ sudo systemctl start backups.mount
[user@centos8 system]$ df -h /backups
Filesystem Size Used Avail Use% Mounted on
/dev/mapper/myvdo 12G 474M 12G 4% /backups
[user@centos8 system]$ sudo systemctl stop backups.mount
[user@centos8 system]$ df -h /backups
Filesystem Size Used Avail Use% Mounted on
/dev/mapper/cl_centos8-root 17G 7.9G 9.2G 46% /
[user@centos8 system]$ █
```

Теперь протестируем наш mount. Посмотрим, примонтировано ли что-то в /backups:

```
df -h /backups
```

Как видите, /backups сейчас принадлежит корневой файловой системе. Запустим наш юнит:

```
sudo systemctl start backups.mount
df -h /backups
```

Теперь файловая система с vdo примонтировалась куда нужно. Остановим юнит:

```
sudo systemctl stop backups.mount
df -h /backups
```

Теперь ничего не примонтировано. Т.е. сейчас за монтирование отвечает только systemd - в зависимости от того, запущен юнит или нет, у нас будет монтироваться файловая система.

```
[user@centos8 system]$ ls /backups
[user@centos8 system]$ systemctl list-units --type automount | head -3
UNIT LOAD ACTIVE SUB DESCRIPTION
proc-sys-fs-binfmt_misc.automount loaded active waiting Arbitrary Executable File Formats File
System Automount Point

[user@centos8 system]$ systemctl cat proc-sys-fs-binfmt_misc.automount | tail -5
ConditionPathExists=/proc/sys/fs/binfmt_misc/
ConditionPathIsReadWrite=/proc/sys

[Automount]
Where=/proc/sys/fs/binfmt_misc
[user@centos8 system]$ █
```

Для примера, рассмотрим ещё юнит automount. В нашей ситуации он не нужен, но в целом полезно знать. Сейчас наша файловая система отмонтирована, а значит /backups пустой:

```
ls /backups
```

automount позволяет монтировать файловую систему при запросе к ней:

```
systemctl list-units --type automount | head -3
systemctl cat proc-sys-fs* | tail -5
```

Т.е. пока нам не нужно, файловая система не примонтирована. Но стоит нам посмотреть или зайти в эту

директорию - systemd сам примонтирует то что нужно. Выйдем, не будем пользоваться - отмонтирует. Это обычно нужно для сетевых файловых систем, для зашифрованных дисков и т.п. - чтобы сократить время запуска операционной системы, и даёт кое-какие определённые плюсы, которые нам сейчас не интересны.

```
GNU nano 2.9.8                                backups.automount

[Unit]
Description=Mount for backups
Before=local-fs.target
After=vdo.service
Requires=vdo.service

[Automount]
Where=/backups
TimeoutIdleSec=10

[Install]
WantedBy=multi-user.target
```

automount юнит должен называться как и сам mount, но после точки - automount:

```
sudo nano backups.automount
```

Так как и при автомонтировании нам нужен сервис vdo, секция [Unit] со всеми зависимостями должна совпадать, поэтому просто скопируем с основного mount:

```
systemctl cat backups.mount
```

Основная секция: [Automount]. Здесь мы указываем `Where=/backups`, т.е. куда мы заходим, чтобы у нас автомонтировалась файловая система. Также здесь время для размонтирования - `TimeoutIdleSec=10`. Т.е. если на протяжении 10 секунд не обращаться к примонтированной файловой системе, она автоматом отмонтируется. Я указал небольшое время для теста, в реальных условиях обычно больше. В отличии от предыдущего юнита, который мы хотим запускать при необходимости, этот юнит должен работать всегда - если этот юнит не будет работать, автомонтирования не будет. Поэтому нам нужно, чтобы юнит запускался при включении компьютера. Для этого нужна секция [Install]. Здесь обычно указывают, каким сервисом или таргетом требуется данный юнит, и, как правило, речь про multi-user.target - `WantedBy=multi-user.target`. Сохраним и выйдем.

```
[user@centos8 system]$ sudo systemctl daemon-reload
[sudo] password for user:
[user@centos8 system]$ sudo systemctl enable --now backups.automount
Created symlink /etc/systemd/system/multi-user.target.wants/backups.automount → /etc/systemd/sys-
tem/backups.automount.
[user@centos8 system]$ systemctl status backups.automount
● backups.automount - Mount for backups
  Loaded: loaded (/etc/systemd/system/backups.automount; enabled; vendor preset: disabled)
  Active: active (waiting) since Thu 2021-04-08 15:23:48 +04; 52s ago
    Where: /backups

Apr 08 15:23:48 centos8 systemd[1]: Set up automount Mount for backups.
[user@centos8 system]$ █
```

Применим наш конфиг:

```
sudo systemctl daemon-reload
```

Затем включить и запустим сервис:

```
sudo systemctl enable --now backups.automount
```

Здесь ключ `--now` нужен, чтобы разом добавить в автозапуск и включить юнит. Также проверим статус:

```
systemctl status backups.automount
```

Всё работает.

```
[user@centos8 system]$ df -h | grep vdo
[user@centos8 system]$ ls /backups
user2.tar.gz user.tar.gz
[user@centos8 system]$ df -h | grep vdo
/dev/mapper/myvdo           12G  474M   12G   4% /backups
[user@centos8 system]$ df -h | grep vdo
[user@centos8 system]$ █
```

Теперь потестим. Для начала убедимся, что ничего не примонтировано:

```
df -h | grep vdo
```

Вроде не примонтировано. А попробуем посмотреть содержимое:

```
ls /backups
```

Файлы на месте. Сразу же проверим список примонтированных файловых систем:

```
df -h | grep vdo
```

Как видите, файловая система примонтировалась. Подождём немного и перепроверим:

```
df -h | grep vdo
```

опять пусто.

```
[user@centos8 system]$ sudo systemctl disable --now backups.automount
[sudo] password for user:
Removed /etc/systemd/system/multi-user.target.wants/backups.automount.
[user@centos8 system]$
```

Но, как я говорил, нам `automount` не нужен, поэтому его убираем:

```
sudo systemctl disable --now backups.automount
```

Теперь сделаем сам сервис:

```
sudo nano mybackups.service
```

В секции [Unit] укажем описание - `Description=Creating /home/user backup`. Укажем, что для работы сервиса нужен юнит `backups.mount` и что этот сервис запускается только после него - `Requires=backups.mount, After=backups.mount`. Дальше добавляем секцию [Service] - здесь мы указываем что и как запускать, какие-то дополнительные команды до и после, от какого пользователя, что делать, если сервис упал и кучу других настроек. Для начала нужно указать тип - `Type=oneshot` - этот тип используется для скриптов.

Типы служб

Есть несколько типов запуска служб, которые нужно иметь в виду при написании файла службы. Тип определяется параметром `Type=` в разделе [Service] :

- `Type=simple` (по умолчанию): systemd запустит эту службу незамедлительно. Процесс при этом не должен разветвляться (`fork`). Не используйте этот тип, если другие службы зависят от этой в плане очередности запуска. Исключение — активация сокета.
- `Type=forking` : systemd считает службу запущенной после того, как процесс разветвляется с завершением родительского процесса. Используйте данный тип для запуска классических демонов за исключением тех случаев, когда в таком поведении процесса нет необходимости. Вам следует также указать `PIDFile=`, чтобы systemd мог отслеживать основной процесс.
- `Type=oneshot` : удобен для скриптов, которые выполняют одно задание и завершаются. При необходимости можно задать параметр `RemainAfterExit=yes`, чтобы systemd считал процесс активным даже после его завершения.
- `Type=notify` : идентичен параметру `Type=simple`, но с той оговоркой, что демон пошлет systemd сигнал о своей готовности. Эталонная реализация данного уведомления представлена в `libsystemd-daemon.so`.
- `Type=dbus` : служба считается находящейся в состоянии готовности, когда указанный параметр `BusName` появляется в системнойшине DBus.
- `Type=idle` : systemd отложит выполнение двоичного файла службы до момента выполнения всех остальных задач. В остальном поведение аналогично `Type=simple`.

Подробнее о параметре `Type` смотрите `systemd.service(5) § OPTIONS`.

В чём суть типа - обычно за сервисом стоят демоны - программы, которые постоянно работают на фоне. И systemd отслеживает состояние процесса, который запустила, по pid-у. И если с этой программой что-то не так, если вдруг процесс остановился, systemd может попробовать перезапустить программу. В случае скриптов всё проще - скрипт запустился, сделал свой дело и завершился. Нужно ведь предупредить systemd, что если запущенная программа завершилась, это не говорит ни о чём плохом - просто скрипт выполнил свою задачу. В общем, программы работают по разному и systemd должен знать, как себя вести в тех или иных случаях.

```
GNU nano 2.9.8                                         mybackups.service

[Unit]
Description=Creating /home/user backup
Requires=backups.mount
After=backups.mount

[Service]
Type=oneshot
#User=backup
ExecStart=/home/user/mybackupscript
StandardOutput=syslog
StandardError=syslog
SyslogIdentifier=mybackup
```

По хорошему, стоит избегать использования root-а и backup должен запускаться от специального пользователя.

зователя. Я не настраивал права и это не совсем входит в данную тему, но для примера я оставлю в закомментированном виде запись `User=backup`, т.е. от чьего имени будет запускаться сервис. Основная строчка - `ExecStart=/home/user/mybackupscript` - собственно, то что нужно запускать. Можем воспользоваться плюсами `systemd` - посыпать стандартный вывод и ошибки в `syslog`, чтобы потом смотреть и сортировать через `journald` - `StandardOutput=syslog`, `StandardError=syslog`, `SyslogIdentifier=mybackup`. Зачастую в сервисе добавляют секцию `Install`, чтобы он автоматом запускался при включении компьютера, но это не актуально для нас - нам не нужно при каждом включении делать бэкап, скрипт будет запускаться таймером. Поэтому с сервисом достаточно, сохраняем и закрываем. Тут ещё может быть миллион разных настроек в зависимости от типа сервиса, но у `systemd` неплохая документация, если вам нужно что-то конкретное - всегда сможете найти или спросить.

```
[user@centos8 system]$ sudo systemctl daemon-reload
[user@centos8 system]$ nano ~/mybackupscript
[user@centos8 system]$ df -h | grep vdo
```

Опять же, перезапускаем демон:

```
sudo systemctl daemon-reload
```

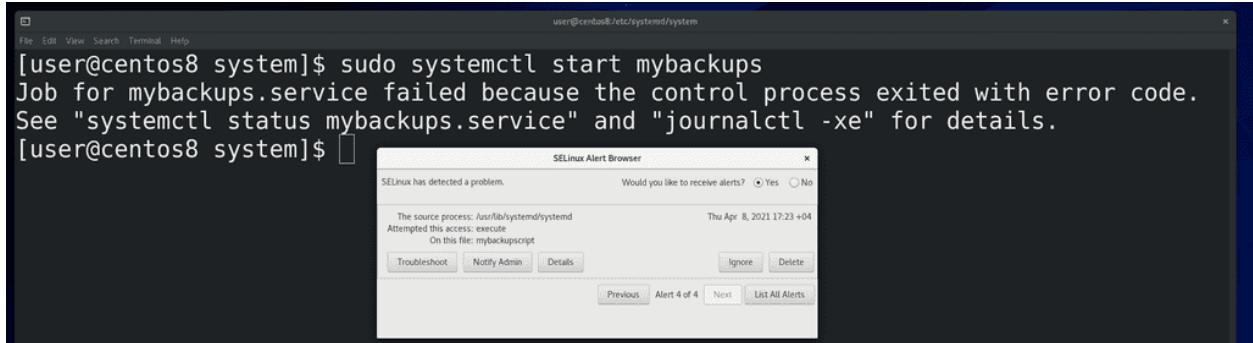
Прежде чем потестить, я немного подправлю скрипт:

```
nano ~/mybackupscript
```

Раньше он сам отвечал за перенаправление вывода в лог файл (`&> $BDIR/$FILENAME.log`), теперь этим занимается `systemd`. Также подправим директорию бэкапа - `/backups`. Прежде чем потестить, проверим, примонтиrovана ли файловая система:

```
df -h | grep vdo
```

нет, отлично.



Попробуем запустить сервис:

```
sudo systemctl start mybackups
```

и видим ошибку. SELinux не разрешает нашему сервису запустить скрипт. О SELinux мы ещё поговорим, но вкратце - это система безопасности.

```
[user@centos8 system]$ sudo cp ~/mybackups /usr/local/bin/  
[user@centos8 system]$ sudo nano mybackups.service  
[user@centos8 system]$ grep ExecStart mybackups.service  
ExecStart=/usr/local/bin/mybackups  
[user@centos8 system]$ sudo systemctl daemon-reload
```

Я подозреваю, в чём дело - ей не нравится, что сервис пытается что-то запустить из домашней директории. Это легко исправить - просто переместим наш скрипт в /usr/local/bin и подправим строчку ExecStart в сервисе:

```
sudo cp ~/mybackups /usr/local/bin  
sudo nano mybackups.service  
grep ExecStart mybackups.service
```

и перечитаем конфиг:

```
sudo systemctl daemon-reload
```

```
[user@centos8 system]$ sudo systemctl start mybackups  
[user@centos8 system]$ ls /backups/  
centos8 08.04.2021 17.33.tar.gz user2.tar.gz user.tar.gz  
[user@centos8 system]$ df -h | grep vdo  
/dev/mapper/myvdo 12G 653M 12G 6% /backups  
[user@centos8 system]$ df -h | grep vdo  
/dev/mapper/myvdo 12G 653M 12G 6% /backups  
[user@centos8 system]$ systemctl status mybackups.service  
● mybackups.service - Creating /home/user backup  
  Loaded: loaded (/etc/systemd/system/mybackups.service; static; vendor preset:  
  Active: inactive (dead)  
  
Apr 08 17:33:59 centos8 mybackup[28903]: ./vboxclient-display-svga-x11.pid  
Apr 08 17:33:59 centos8 mybackup[28903]: ./mybackups
```

Попытка номер 2:

```
sudo systemctl start mybackups
```

Подождали немного - и всё сработало:

```
ls /backups
```

При запуске сервиса у нас выполняется бэкап. И даже файловая система примонтировалась:

```
df -h | grep vdo
```

потому что сервис бэкапа требовал backups.mount. И если посмотреть статус сервиса:

```
sudo systemctl status mybackups
```

внизу видны логи.

```
[user@centos8 system]$ sudo journalctl -eu mybackups
[sudo] password for user:
Apr 08 17:33:59 centos8 mybackup[28903]: ./temp/until
Apr 08 17:33:59 centos8 mybackup[28903]: ./FILE
Apr 08 17:33:59 centos8 mybackup[28903]: ./fILE
Apr 08 17:33:59 centos8 mybackup[28903]: ./fiLE
Apr 08 17:33:59 centos8 mybackup[28903]: ./test/
Apr 08 17:33:59 centos8 mybackup[28903]: ./test/file2
Apr 08 17:33:59 centos8 mybackup[28903]: ./test/file3
Apr 08 17:33:59 centos8 mybackup[28903]: ./test/file1
Apr 08 17:33:59 centos8 mybackup[28903]: ./bash_profile.save
Apr 08 17:33:59 centos8 mybackup[28903]: ./file
```

Их также можно посмотреть через journald:

```
sudo journalctl -eu mybackups
```

```
GNU nano 2.9.8                                backups.mount

[Unit]
Description=Mount for backups
Before=local-fs.target
After=vdo.service
Requires=vdo.service
BindsTo=mybackups.service

[Mount]
Where=/backups
What=/dev/disk/by-uuid/0711d5ea-24fc-4e86-b409-a28fd36e8aaa
Type=xfs
Options=defaults
```

Но так как мы убрали automount, файловая система всё ещё висит, даже если бэкап завершился:

```
df -h | grep vdo
```

Попробуем это подправить. Для этого свяжем юнит mount с сервисом:

```
sudo nano backups.mount
```

и в секцию [Unit] добавим `BindsTo=mybackups.service`. Благодаря этой опции mount будет следить за сервисом - если тот перестанет работать, то и mount завершится. Ладно, сохраним и выйдем.

```
[user@centos8 system]$ sudo systemctl daemon-reload
[user@centos8 system]$ sudo systemctl stop backups.mount
[user@centos8 system]$ df -h /backups/
Filesystem           Size  Used Avail Use% Mounted on
/dev/mapper/cl_centos8-root  17G  7.4G  9.7G  44% /
[user@centos8 system]$ sudo systemctl start mybackups.service
[user@centos8 system]$ df -h /backups/
Filesystem           Size  Used Avail Use% Mounted on
/dev/mapper/cl_centos8-root  17G  7.4G  9.7G  44% /
[user@centos8 system]$ sudo mount /dev/mapper/myvdo /backups/
[user@centos8 system]$ ls /backups/
centos8_08.04.2021_17.33.tar.gz centos8_08.04.2021_18.37.tar.gz user.tar.gz
centos8_08.04.2021_18.35.tar.gz user2.tar.gz
```

Давайте тестировать. Для начала, перечитаем конфиг:

```
sudo systemctl daemon-reload
```

Затем остановим backups.mount:

```
sudo systemctl stop backups.mount
```

и убедимся в этом:

```
df -h /backups
```

Как видите, примонтирована корневая файловая система, а не vdo. Теперь стартанём сервис бэкапа:

```
sudo systemctl start mybackups
```

После чего сразу проверим:

```
df -h /backups
```

Как видите, vdo не примонтирован. Но если примонтировать вручную:

```
sudo mount /dev/mapper/myvdo /backups
ls /backups
```

то последние бэкапы там есть.

```
[user@centos8 ~]$ df -h | grep vdo
[user@centos8 ~]$ sudo mount /dev/mapper/myvdo /backups/
[user@centos8 ~]$ ls /backups/
centos8_08.04.2021_17.33.tar.gz centos8_08.04.2021_19.34.tar.gz
centos8_08.04.2021_19.18.tar.gz centos8_08.04.2021_19.35.tar.gz
centos8_08.04.2021_19.27.tar.gz centos8_08.04.2021_19.38.tar.gz
centos8_08.04.2021_19.29.tar.gz user2.tar.gz
centos8_08.04.2021_19.31.tar.gz user.tar.gz
centos8_08.04.2021_19.32.tar.gz
[user@centos8 ~]$ sudo systemctl status backups.mount
● backups.mount - Mount for backups
  Loaded: loaded (/etc/systemd/system/backups.mount; static; vendor preset: di>
  Active: inactive (dead)
    Where: /backups
      What: /dev/disk/by-uuid/0711d5ea-24fc-4e86-b409-a28fd36e8aaa

Apr 08 19:32:23 centos8 systemd[1]: Unmounting Mount for backups...

```

Правда есть довольно весомый минус - `BindsTo` делает такую привязку, что даже при простом монтировании этой файловой системы в эту директорию, как указано в юните, запускается сервис `mybackups`. Для примера, проверим, примонтирована ли файловая система:

```
df -h | grep vdo
```

пусто. Теперь примонтируем вручную:

```
sudo mount /dev/mapper/myvdo /backups
```

Посмотрим содержимое:

```
ls /backups
```

и сразу тут появился файл с новым бэкапом. Т.е. монтирование привело к запуску `mount` юнита, а он запустил сервис бэкапа. Через какое-то время бэкап закончится, сервис остановится, что приведёт к остановке `mount`-а и файловая система отмонтируется. С одной стороны это легко обойти - при необходимости можно просто монтировать в другую директорию, чтобы лишний раз не делать бэкап. С другой - будь у нас демон, а не скрипт, такой проблемы не было бы, так как есть другая опция - `PartOf`, вместо `BindsTo`, которая не вызывала бы лишний раз сервис бэкапа. Но эта опция отрабатывает только если сервис стопится с помощью `systemd`, а не скрипт завершается. Также есть другая опция - `StopWhenUnneeded` - останавливает юнит, если сейчас никакой другой юнит не требует работы этого. Но почему-то у меня это не сработало.

Timer	Monotonic	Definition
<code>OnActiveSec=</code>	X	This defines a timer relative to the moment the timer is activated.
<code>OnBootSec=</code>	X	This defines a timer relative to when the machine boots up.
<code>OnStartupSec=</code>	X	This defines a timer relative to when the service manager first starts. For system timer units, this is very similar to <code>OnBootSec=</code> , as the system service manager generally starts very early at boot. It's primarily useful when configured in units running in the per-user service manager, as the user service manager generally starts on first login only, not during boot.
<code>OnUnitActiveSec=</code>	X	This defines a timer relative to when the timer that is to be activated was last activated.
<code>OnUnitInactiveSec=</code>	X	This defines a timer relative to when the timer that is to be activated was last deactivated.
<code>OnCalendar=</code>		This defines real-time (i.e., wall clock) timers with calendar event expressions. See <code>systemd.time(7)</code> for more information on the syntax of calendar event expressions. Otherwise, the semantics are similar to <code>OnActiveSec=</code> and related settings. This timer is the one most like those used with the cron service.

Теперь перейдём к таймеру. Имя таймера должно соответствовать имени сервиса, но в конце - `.timer`:

```
sudo nano mybackups.timer
```

Опять же, начинаем с секции [Unit], где добавляем описание - `Description=Timer for my backup service`. Потом добавляем секцию [Timer], где и будем указывать периодичность бэкапа. Время можно задавать по разному - связать его с определёнными часами и минутами, например, в 11 вечера каждое воскресенье, либо со временем запуска компьютера, например, через 15 минут после включения, либо промежуток времени от последнего запуска таймера. Много различных вариантов, поэтому мы сделаем так - делать бэкапы каждый день в 11 вечера, а если вдруг компьютер был выключен и пропустил один бэкап - то сделать его после включения.

```
GNU nano 2.9.8                                         mybackups.timer

[Unit]
Description=Timer for my backup service

[Timer]
OnCalendar=*-*-* 23:50:00
Persistent=true

[Install]
WantedBy=timers.target
```

Для этого мне понадобится опция `OnCalendar` со значением

`*-*-* 23:50:00`

Здесь первые три звёздочки - это год, месяц и день. Хотя можно указывать по разному. Также добавлю опцию `Persistent=true`. Это как раз для того, чтобы в случае пропускания таймера, если компьютер был выключен, после включения автоматом сработал наш бэкап сервис. Ну и для таймеров существует отдельный таргет, который можно указать в секции [Install], чтобы добавить таймер в автозапуск.

```
[user@centos8 system]$ sudo systemctl daemon-reload
[user@centos8 system]$ sudo systemctl enable --now mybackups.timer
Created symlink /etc/systemd/system/timers.target.wants/mybackups.timer → /etc/systemd/system/mybackups.timer.
[user@centos8 system]$ sudo systemctl list-timers
NEXT           LEFT      LAST          PASSED      UNIT            >>
Thu 2021-04-08 21:05:04 +04 2min 26s left n/a          n/a    systemd-tmpfil>
Thu 2021-04-08 21:10:00 +04 7min left   Thu 2021-04-08 21:00:03 +04 2min 34s ago sysstat-collec>
Thu 2021-04-08 22:00:08 +04 57min left   Thu 2021-04-08 21:00:07 +04 2min 30s ago dnf-makecache.>
Thu 2021-04-08 23:50:00 +04 2h 47min left n/a          n/a    mybackups.time>
Fri 2021-04-09 00:00:00 +04 2h 57min left Thu 2021-04-08 11:05:58 +04 9h ago    unbound-anchor>
Fri 2021-04-09 00:07:00 +04 3h 4min left n/a          n/a    sysstat-summar>

6 timers listed.
Pass --all to see loaded but inactive timers, too.
```

Опять же, после сохранения файла перечитываем конфиги:

```
sudo systemctl daemon-reload
```

Потом включаем таймер:

```
sudo systemctl enable --now mybackups.timer
```

Ну и смотрим список активных таймеров и время их срабатывания:

```
sudo systemctl list-timers
```

```
GNU nano 2.9.8                                     testmounts.target

[Unit]
Description=Target for automounts

[Install]
WantedBy=multi-user.target
```

Напоследок, рассмотрим таргеты. В основном таргеты мы создаём, чтобы объединять юниты в группы и через эту группу управлять ими, указывать целую группу в зависимостях и т.п.. Это не совсем подходит под нашу задачу, но для примера используем что-нибудь другое. У нас остался неиспользованный automount, возьмём его для теста. Создадим таргет:

```
sudo nano testmounts.target
```

Создаём секцию [Unit] и добавляем Description=Target for automounts. И создаём секцию [Install], где укажем, чтобы наш таргет грузился при включении компьютера - WantedBy=multi-user.target. Сохраняем и выходим.

```
[user@centos8 system]$ sudo nano testmounts.target
[sudo] password for user:
[user@centos8 system]$ sudo systemctl daemon-reload
[user@centos8 system]$ sudo systemctl enable testmounts.target
Created symlink /etc/systemd/system/multi-user.target.wants/testmounts.target → /etc/systemd/system/
testmounts.target.
[user@centos8 system]$ █
```

Перечитываем конфиг:

```
sudo systemctl daemon-reload
```

Включаем наш таргет, чтобы он грузился при запуске системы:

```
sudo systemctl enable testmounts.target
```

```
[Install]
WantedBy=testmounts.target
```

Дальше надо в юнитах, которые мы хотим объединить, указать этот таргет в секции [Install], вместо multi-user таргета - WantedBy=testmounts.target. Сохраняем и выходим.

```
[user@centos8 system]$ sudo nano backups.automount
[user@centos8 system]$ sudo systemctl daemon-reload
[user@centos8 system]$ sudo systemctl enable backups.automount
Created symlink /etc/systemd/system/testmounts.target.wants/backups.automount → /etc/systemd/system/
backups.automount.
[user@centos8 system]$ sudo systemctl status backups.automount
● backups.automount - Mount for backups
  Loaded: loaded (/etc/systemd/system/backups.automount; enabled; vendor preset: disabled)
    Active: inactive (dead)
      Where: /backups
[user@centos8 system]$
```

Снова перечитываем конфиг:

```
sudo systemctl daemon-reload
```

И включаем automount:

```
sudo systemctl enable backups.automount
```

Обратите внимание, где создалась символьическая ссылка - у таргета появилась своя директория, как это было у multi-user-а. Теперь у нас есть таргет, внутри которого один юнит. Ну и перед тестом проверим статус automount-а - он выключен.

```
[user@centos8 system]$ sudo systemctl start testmounts.target
[user@centos8 system]$ sudo systemctl status backups.automount
● backups.automount - Mount for backups
  Loaded: loaded (/etc/systemd/system/backups.automount; enabled; vendor preset: disabled)
    Active: active (waiting) since Thu 2021-04-08 21:35:52 +04; 3s ago
      Where: /backups

Apr 08 21:35:52 centos8 systemd[1]: backups.automount: Directory /backups to mount over is not empty
Apr 08 21:35:52 centos8 systemd[1]: Set up automount Mount for backups.
[user@centos8 system]$ sudo systemctl stop testmounts.target
[user@centos8 system]$ sudo systemctl status backups.automount
● backups.automount - Mount for backups
  Loaded: loaded (/etc/systemd/system/backups.automount; enabled; vendor preset: disabled)
    Active: active (waiting) since Thu 2021-04-08 21:35:52 +04; 22s ago
      Where: /backups
```

Попробуем стартануть таргет:

```
sudo systemctl start testmounts.target
```

и перепроверим статус automount-а - он запустился. Но при остановке таргета:

```
sudo systemctl stop testmounts.target
```

сервис сам не останавливается:

```
systemctl status backups.automount
```

С одной стороны это хорошо - это немного безопаснее, чтобы случайно не вырубить важный сервис.

```
GNU nano 2.9.8                                backups.automount

[Unit]
Description=Mount for backups
Before=local-fs.target
After=vdo.service
Requires=vdo.service
PartOf=testmounts.target

[Automount]
Where=/backups
TimeoutIdleSec=10

[Install]
WantedBy=testmounts.target
```

Однако, если мы всё же хотим, чтобы юнит перезагружался вместе с таргетом, можно в секцию [Unit] добавить опцию PartOf:

```
sudo nano backups.automount
```

PartOf=testmounts.target

Сохраняем и выходим.

```
[user@centos8 system]$ sudo nano backups.automount
[user@centos8 system]$ sudo systemctl daemon-reload
[user@centos8 system]$ sudo systemctl status backups.automount
● backups.automount - Mount for backups
  Loaded: loaded (/etc/systemd/system/backups.automount; enabled; vendor preset: disabled)
  Active: active (waiting) since Thu 2021-04-08 21:35:52 +04; 4min 32s ago
    Where: /backups

Apr 08 21:35:52 centos8 systemd[1]: backups.automount: Directory /backups to mount over is not empty
Apr 08 21:35:52 centos8 systemd[1]: Set up automount Mount for backups.
[user@centos8 system]$ sudo systemctl stop testmounts.target
[user@centos8 system]$ sudo systemctl status backups.automount
● backups.automount - Mount for backups
  Loaded: loaded (/etc/systemd/system/backups.automount; enabled; vendor preset: disabled)
  Active: inactive (dead) since Thu 2021-04-08 21:40:53 +04; 1s ago
    Where: /backups

Apr 08 21:35:52 centos8 systemd[1]: backups.automount: Directory /backups to mount over is not empty
```

А дальше как обычно - перечитываем конфиг:

```
sudo systemctl daemon-reload
```

Посмотрим статус юнита:

```
sudo systemctl status backups.automount
```

работает. Попробуем остановить таргет:

```
sudo systemctl stop testmounts.target
```

а потом заново посмотреть статус сервиса:

```
sudo systemctl status backups.automount
```

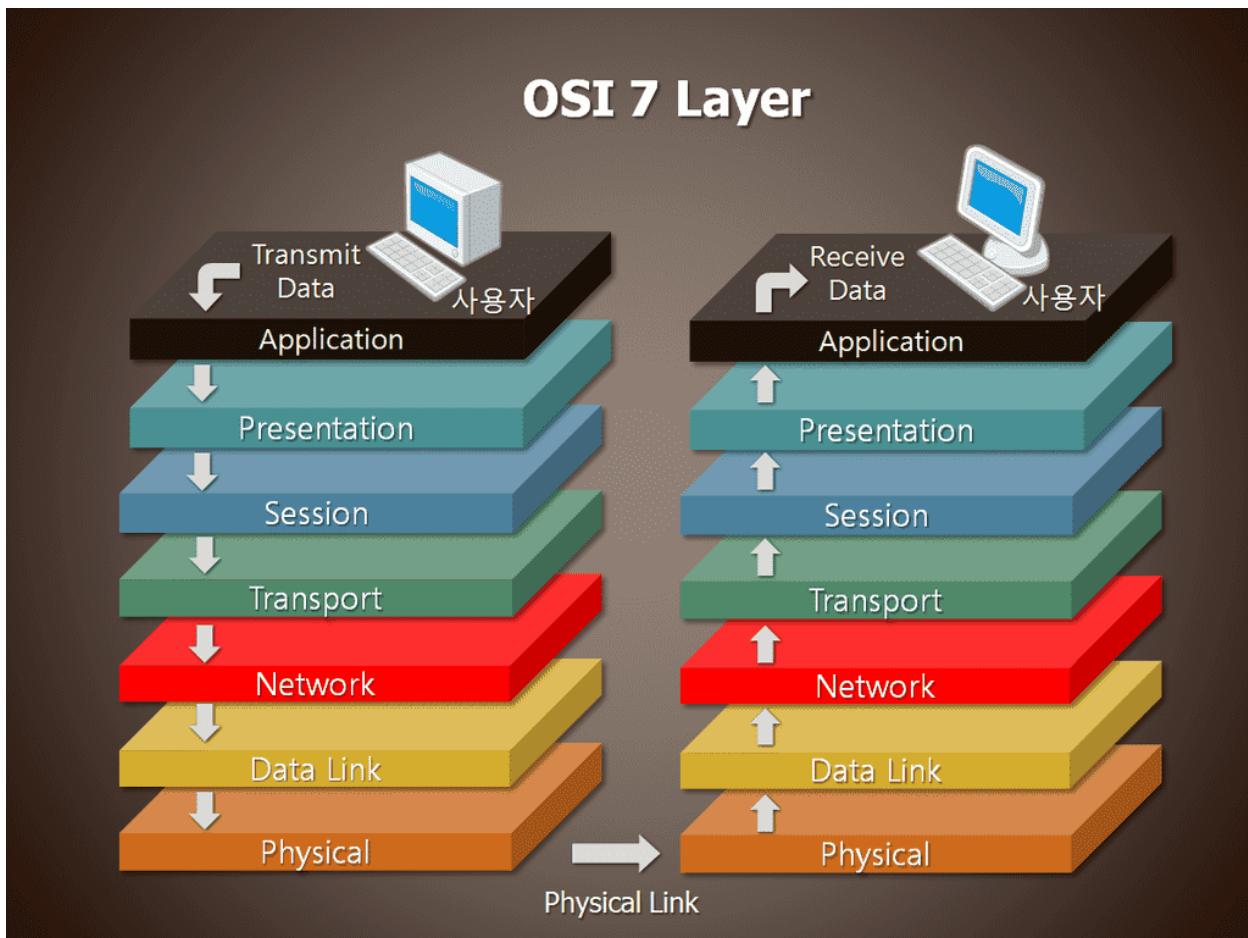
сервис стопнулся. Т.е. всё правильно.

Подведём итоги. Мы с вами научились делать юниты для systemd - mount, automount, service, timer и target. Это довольно простая задача. И хотя эта тема невероятно огромная, в большинстве случаев хватает небольших юнит файлов. В системе и интернете есть документация и большое количество примеров. Хотя mount-ы и timer-ы могут встречаться реже и вместо них всегда можно использовать cron и fstab, с сервисами всё несколько иначе. Нередко администраторам попадаются программы, которые нужно запускать при включении сервера, но, либо производитель не дал готовые файлы для systemd, либо эту программу написали ваши программисты, которые могут не разбираться в systemd, ну или вы сами хотите изменить процесс, построить его иначе. И, вместо того, чтобы при каждом включении вручную всё запускать, вы всегда можете сделать из скриптов и программ сервисы, которые автоматизируют и облегчат вам работу.

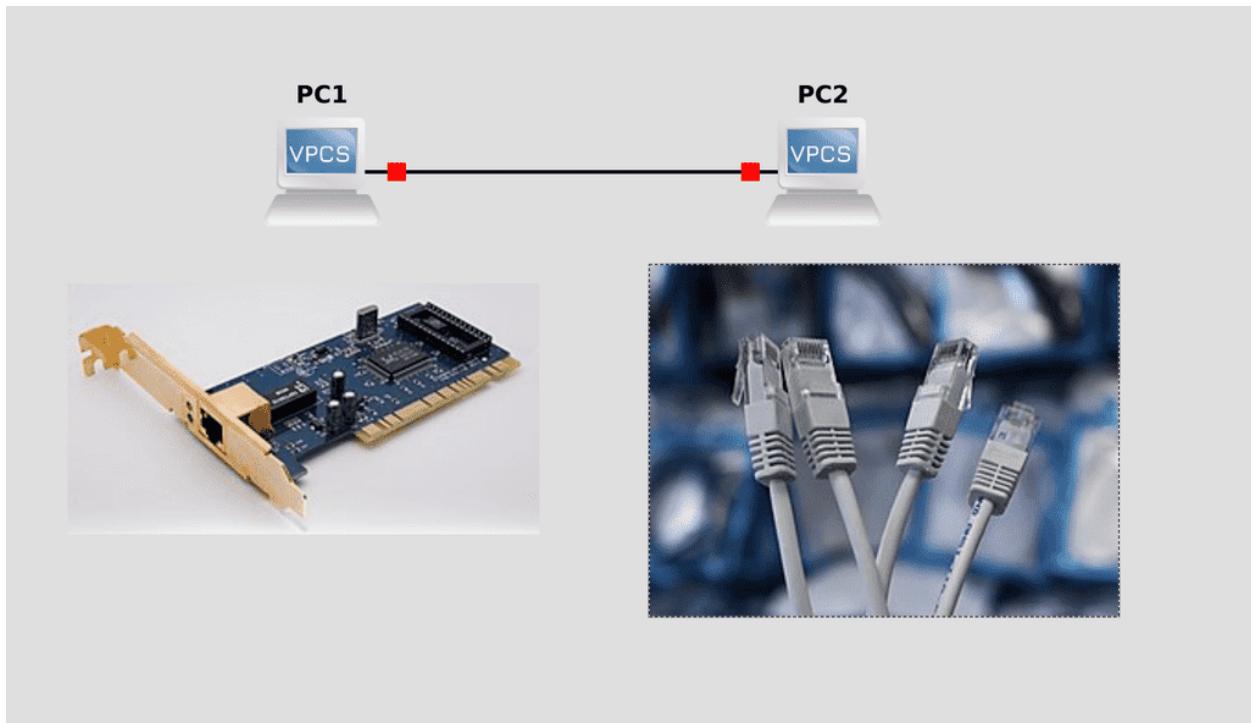
2.42 42. Основы сетей

2.42.1 42. Основы сетей

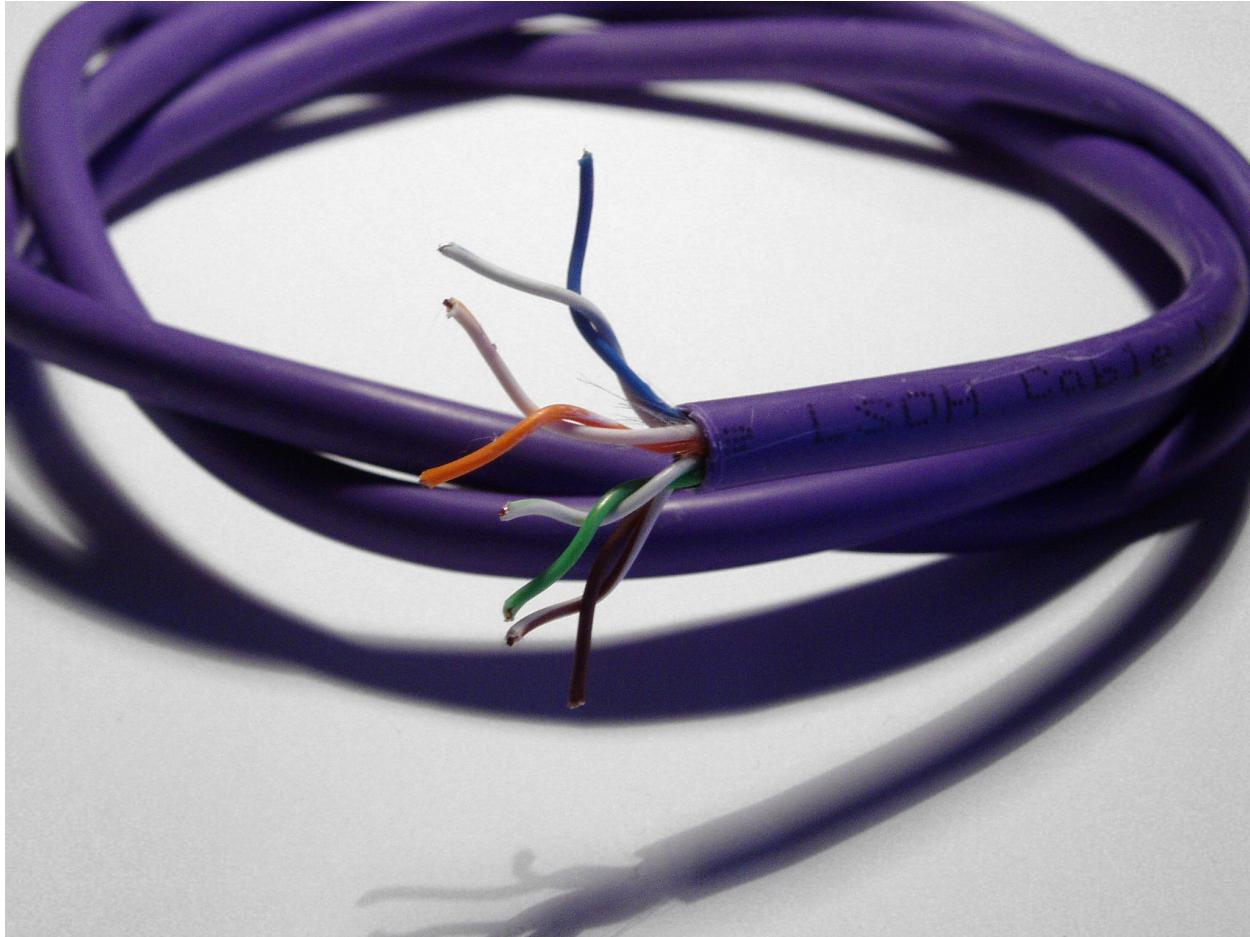
Для продолжения курса потребуются знания работы сети, а это огромный пласт знаний. По сетям в интернете материала, наверное, больше, чем по линуксам. Но я всё же постараюсь объяснить основы и по мере продвижения курса буду затрагивать различные детали работы сети, связанные с той или иной темой. Сразу предупрежу, что я очень сильно упрощаю, так как это не курс по сетям, я расскажу только то, что считаю необходимым на данном этапе.



Если вы смотрите это видео в интернете, вы уже знаете, зачем нужна сеть. Разве что стоит отметить, что почти все компании имеют какие-то внутренние сервисы и для них организуется так называемая локальная сеть. Но мы ещё к этому придём. Итак, для того, чтобы различные компьютеры с различными операционными системами и программами могли взаимодействовать между собой, существует универсальная сетевая модель, называемая OSI, которая определяет стандарты. Эта модель делит взаимодействие на шаги, так называемые уровни, и каждый уровень имеет свои правила. Набор этих правил определяет, как именно должно происходить взаимодействие и называется протоколом. Названия каких-то протоколов вы наверняка где-то видели - IP, DNS, HTTP. Хотя стандартная модель OSI предполагает 7 уровней, администраторы чаще всего работают со вторым, третьим, четвёртым уровнями и объединяют 5, 6 и 7 уровни в один, чаще всего называя седьмым.

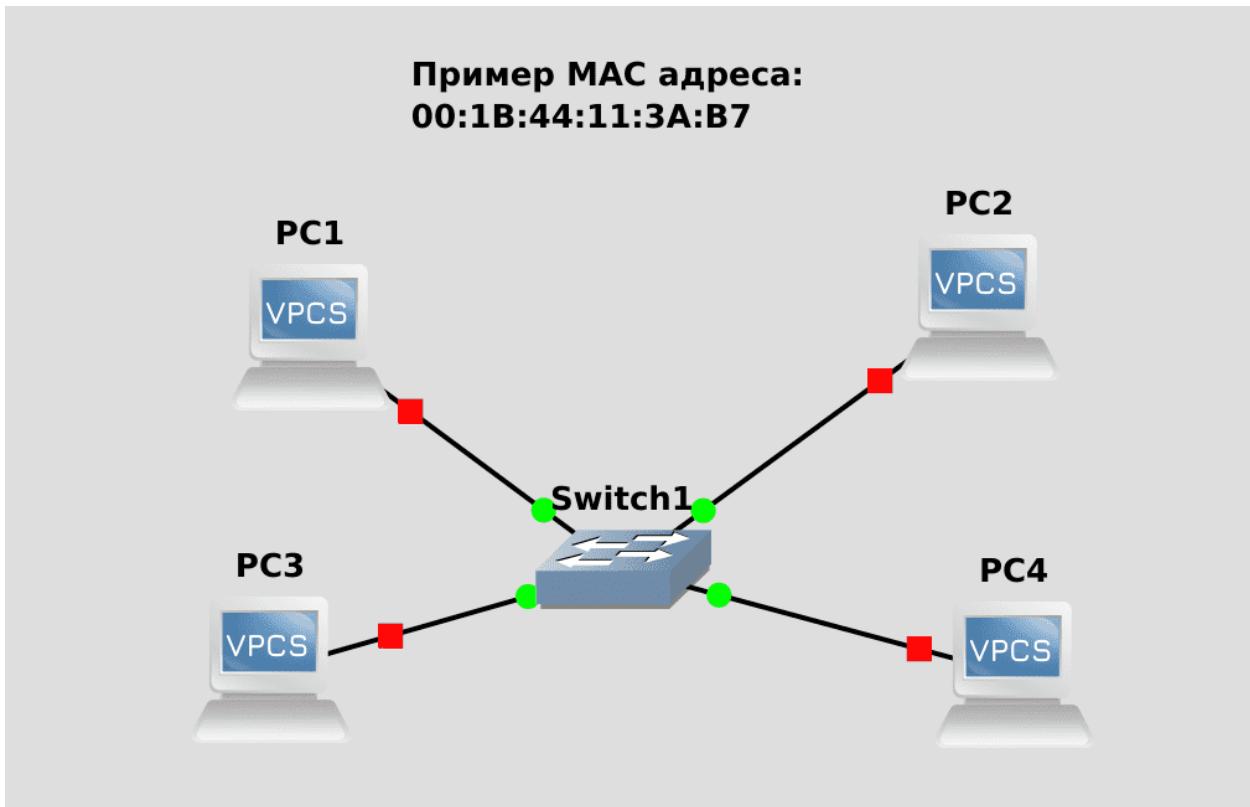


Первый уровень - физический, предполагает стандарты и технологии для физического взаимодействия. Технологии бывают разные - ethernet, wifi, оптика. Для работы с сетью на каждом компьютере имеются сетевые адаптеры - чаще всего это ethernet адаптер, ну и wifi на ноутбуках.



Скорее всего, по работе, вы будете взаимодействовать с ethernet-ом. Для соединения одного компьютера с другим используются кабели определённых категорий, чаще всего в наше время это cat 5. Он состоит из 8 медных проводов, по которым и ходят сигналы. Кабели скручены по два по определённому стандарту и такая пара проводов называется витой парой. И по работе нужно бывает обрезать эти кабели до нужной длины, вставлять их в коннектор и обжимать. Погуглите «обжим витой пары», посмотрите на ютубе, так как новичков часто спрашивают об этом на собеседованиях.

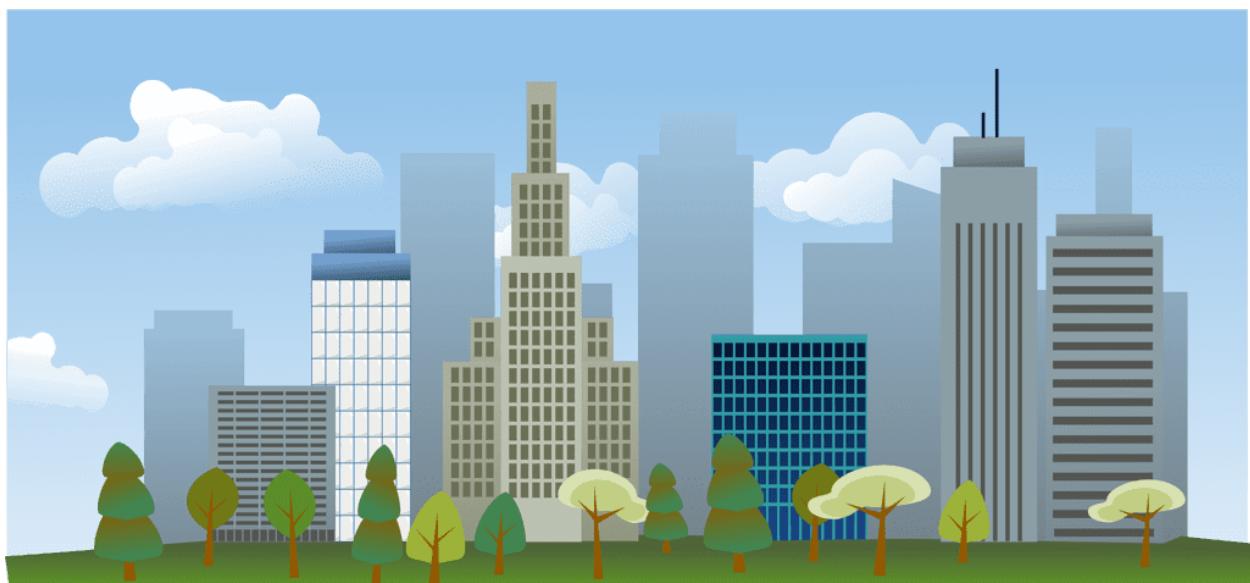
cat 5 поддерживает до 1 гигабита в секунду - 1 gbps. gbps означает gigabits per second. Скорость работы сети исчисляется в битах и чтобы понять пропускную способность в байтах просто делите на 8, так как 1 байт - это 8 бит. Т.е. cat 5 в идеале может передавать файлы со скоростью $1024/8 = 128$ мегабайт в секунду. Но на самом деле чуть меньше, так как окружение всё таки влияет.



На втором уровне, который называется канальный, компьютеры могут различать друг друга. Для этого на каждом сетевом адаптере, на каждом порту есть специальный MAC адрес. Он выглядит как 12 букв и цифр, поделённых двоеточиями по два - 00:1B:44:11:3A:B7. Он уникальный и выставляется производителем адаптера ещё на заводе. Так как мак адрес позволяет опознавать другие компьютеры, можно подключать больше компьютеров друг к другу. Компьютеров может быть много, но нереально на каждом компьютере иметь сотни портов и прокидывать тысячи кабелей от одного компьютера к другому.



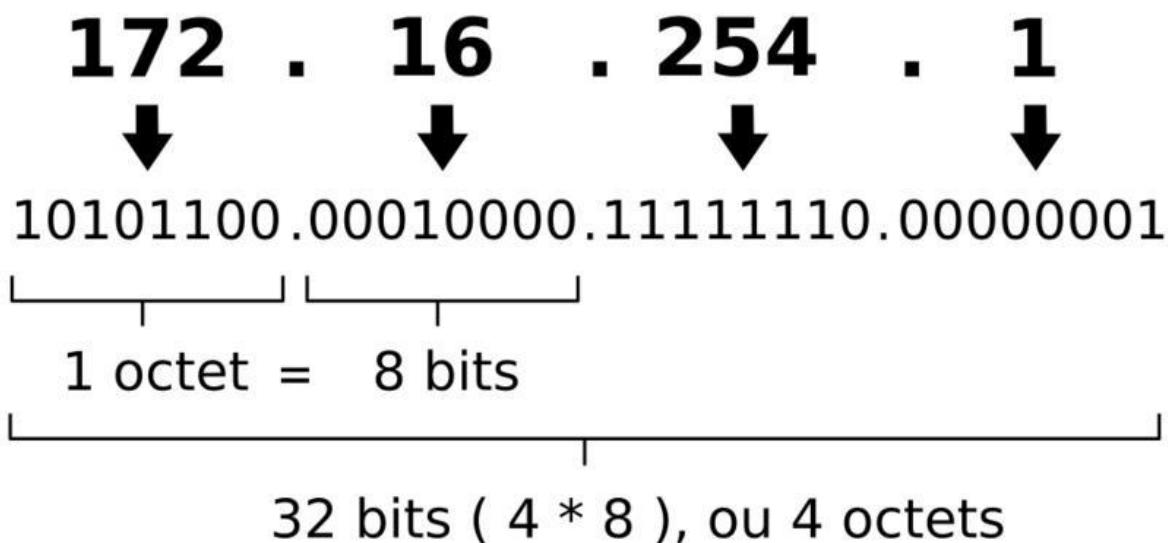
Чтобы связать несколько компьютеров в одну сеть чаще всего используются коммутаторы, в простонародье - свитчи. В простом варианте это устройства, имеющие кучу портов - от 4 до 96, если не говорить о каких-то редких. Если не хватает или устройства расположены на разных этажах - можно подключать несколько свитчей друг к другу. В свитчах есть как минимум таблица мак адресов - свитч запоминает, за каким портом находится какой мак адрес. Когда один компьютер хочет связаться с другим, он обращается к определённому мак адресу, свитч это видит и связывает два устройства через свои порты.



Но такой подход годится, когда компьютеров не слишком много - десяток, сотня, не более. Компьютеров может быть очень много, некоторые могут менять mac адрес, скажем, после замены сетевого адаптера, да и в целом работать с mac адресами не удобно. Поэтому mac адреса используются только в локальных сетях. Скажем, у вас дома, если у вас несколько устройств, или в небольших офисах, в отделах больших компаний и т.п. Представьте небольшой город. Каждое здание - это один компьютер. Локальная сеть - это уличные дороги, связывающие здания.

Так вот, третий уровень модели OSI - сетевой. И на нём вводится понятие IP адрес. У каждого компьютера в локальной сети есть IP адрес - как и у каждого здания в городе есть свой адрес. IP адреса позволяют взаимодействовать компьютерам, находящимся в разных сетях. На самом деле они и в локальных сетях используются, просто компьютер находит соответствующий mac адрес по IP. Есть различные версии протокола IP - ipv4 и ipv6. Более популярный и простой - ipv4 и мы поговорим о нём.

Une adresse IPv4 (notation décimale à point)



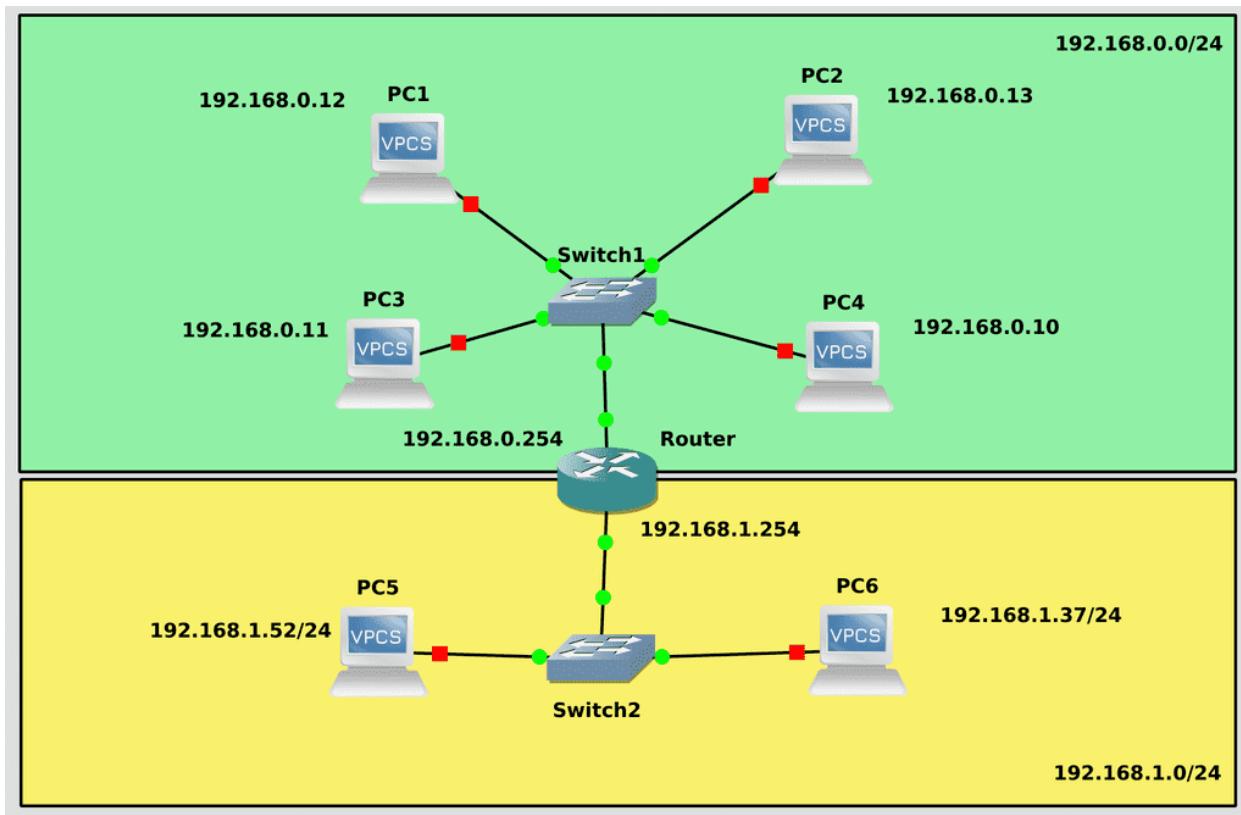
Итак, IP адрес состоит из 4 чисел, разделённых точкой. Числа от 0 до 255. Т.е. технически IP состоит из 4 байт, каждый байт может уместить 8 бит, а каждый бит принять 2 значения. Итого 2 в 8 степени. Получается около 4 миллиардов адресов. Но этого не хватает - у одного компьютера может быть несколько адресов, у каждой виртуалки свои адреса, телефоны и прочие устройства - в общем, много всего. Но сеть позволяет сделать так, чтобы у разных компьютеров были одинаковые адреса, правда не в рамках одной локальной сети.

Скажем, у большинства населения по домам одинаковые IP адреса, допустим 192.168.0.100. Из примера с городом - в разных городах могут быть одинаковые адреса, скажем «улица Ленина, 20». В одном городе так делать не стоит, а в разных - без проблем.

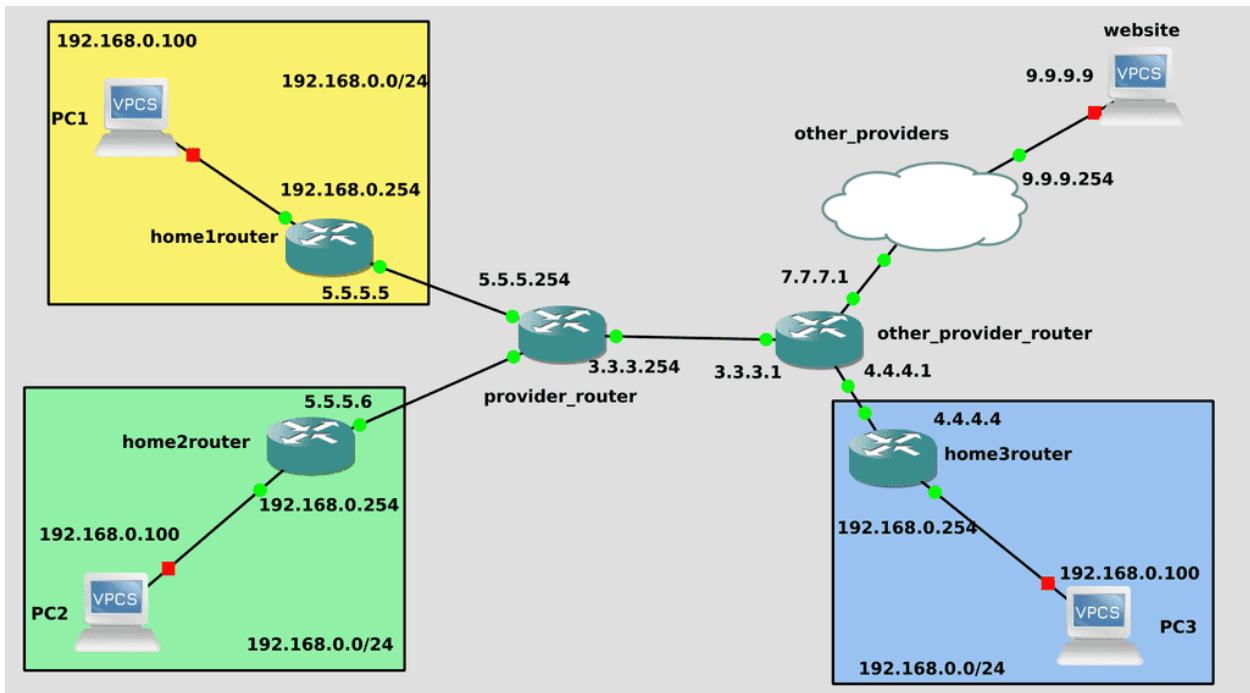
Чтобы компьютеры могли понять, какой адрес находится в их сети, а какой в другой, есть маска подсети. Она указывается рядом с IP адресом. Она тоже состоит из 4 чисел, разделённых точкой, также имеющих значение от 0 до 255. Для примера возьмём самую популярную маску, скорее всего, используемую у вас дома - 255.255.255.0. Последнее число - 0 - говорит о том, что у вас в сети может быть 256 значений, т.е. 256 адресов. Но, на самом деле, первое и последнее значение зарезервированы. В данном случае 0 используется как адрес сети, как адрес самого города. А 255 - broadcast - способ обратиться ко всем компьютерам в этой сети. Скажем, если у вашего компьютера IP адрес - 192.168.0.100

с маской 255.255.255.0, то для него локальной сеткой будут компьютеры с адресами от 192.168.0.1 до 192.168.0.254, адресом сети будет 192.168.0.0, а броадкаст адресом будет 192.168.0.255. Кстати, нередко маску пишут как /24. Это означает, что первые три числа - 255, а это максимальное значение 8 бит. $8+8+8 = 24$. Не будем усложнять тему подсетей на сегодня, поэтому продолжим.

Почему именно 192.168? Есть список зарезервированных IP сетей, выделенных для частного пользования в домах и компаниях. И придя в гости или в какие-то компании, вы будете натыкаться на схожие подсети и IP адреса.



Так вот, допустим, компьютер видит, что какой-то IP адрес находится в другой сети, как он к нему обратится? Для этого есть такой механизм, как маршрутизация, в простонародье - роутинг. Обычно для этого используются маршрутизаторы или роутеры - как, например, ваш домашний роутер. Он находится с вами в одной сети, но у него есть выход и в другую сеть, благодаря чему он может связывать компьютеры вашей сети с компьютерами другой. Обычно роутерам дают граничные IP адреса - либо 1, либо 254, чтобы было понятнее, что это роутер. А на компьютере, при настройке IP адреса, также можно указать gateway - шлюз - и тут указывается адрес роутера. Если вы его указали - компьютер при необходимости связи с другой сетью будет обращаться к роутеру, а тот будет связываться с той стороной. Но роутер не будет передавать мак адреса, только IP адрес.

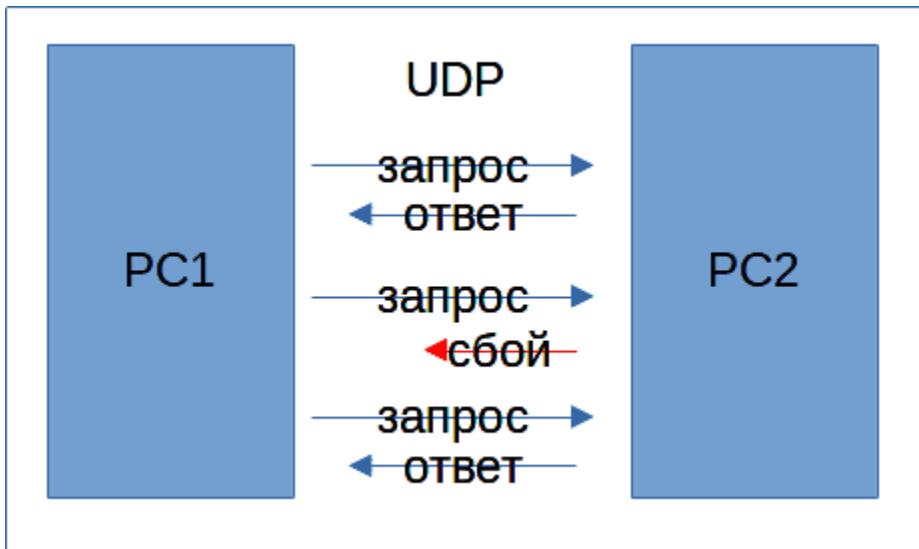


И интернет, в сильно упрощённом виде - это набор роутеров и свитчей - в домах стоят роутеры, которые подключены к роутеру вашего провайдера, а он соединён с роутерами других провайдеров. И все эти связи и формируют интернет. При этом, посмотрите на схему - у домашних компьютеров могут быть одинаковые адреса. Из интернета нельзя попасть в вашу домашнюю сеть - 192.168.0.0 - так как она зарезервирована для частного использования и любой желающий может её у себя поднять. Но при этом, у вашего роутера есть и внешний адрес, например - 5.5.5.5. И когда вы выходите в интернет, ваш роутер подменяет ваш адрес на свой внешний, чтобы вам могли ответить. Такая технология называется NAT - преобразование сетевых адресов. Скажем, если website увидит запрос от 192.168.0.100, то он не будет знать, кому посыпать ответ. Но ваш роутер при запросе подменяет ip адрес, website видит адрес 5.5.5.5 и отвечает ему. А роутер перенаправляет ответ вам, так как при NAT-е он запоминает ваш запрос и ждёт на него ответ.

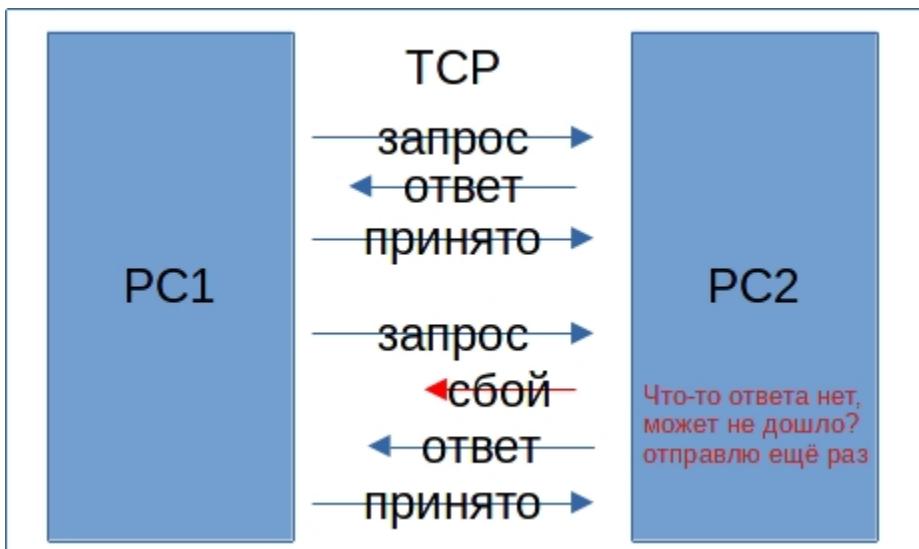
При этом, website или любой другой компьютер в интернете не могут напрямую обратиться к вашему компьютеру, опять же, потому что 192.168.0.0 - недоступен из интернета. Другие компьютеры могут обратиться к вашему внешнему адресу - 5.5.5.5 - но роутер сам по себе не будет перенаправлять запросы на ваш компьютер, так как это слишком опасно, в интернете много вредоносных программ. Да и NAT позволяет держать за одним внешним IP адресом множество компьютеров. Скажем, ваш телефон, компьютер, ноутбук и телевизор в интернет выходят с одного адреса. В компаниях так делают сотни и тысячи компьютеров. Безопасность и экономия IP адресов.

Внутри компаний может быть множество сетей, например, сеть для серверов и сеть для пользователей. В таких случаях NAT обычно не используется, так как бывает нужно, чтобы и пользователи могли достичься до серверов и сервера могли подключиться к компьютерам пользователей. Поэтому внутри компаний не используют одинаковые IP адреса для сетей.

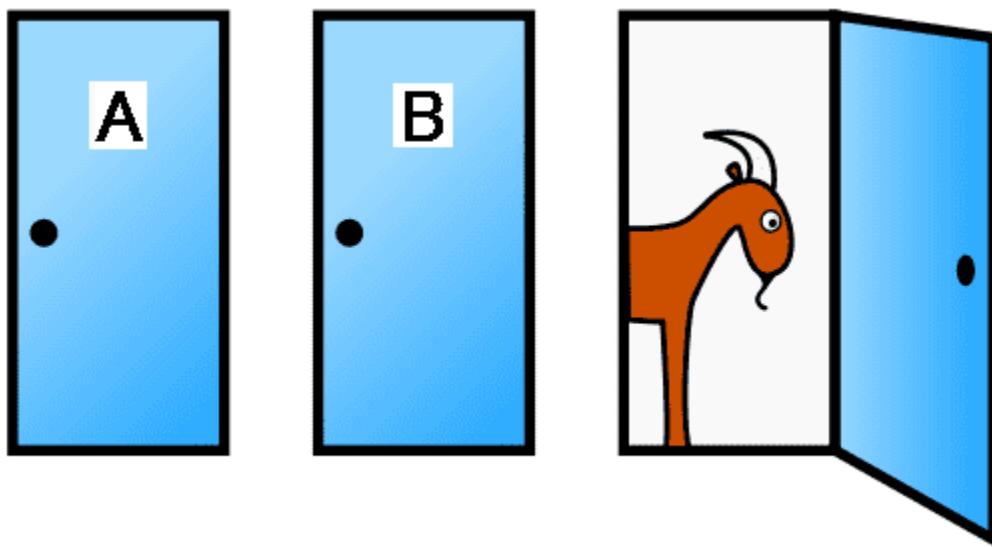
Так вот, всё это был уровень 3 модели OSI - layer 3 - или просто l3. Первые три уровня формируют карту подключений - как дороги в городах. Есть дома, у каждого дома свой адрес, каждый дом находится в каком-то городе и, чтобы добраться из одного города в другой, надо поехать к выходу из города, оттуда доехать до другого города и потом до нужного дома. Компьютер - роутер - другой роутер - другой компьютер. Но суть сетей не в том, чтобы доехать куда-то, а в том, чтобы доставлять посылки, ну или по компьютерному - пакеты. Компьютеру один надо доставить пакет до компьютера два.



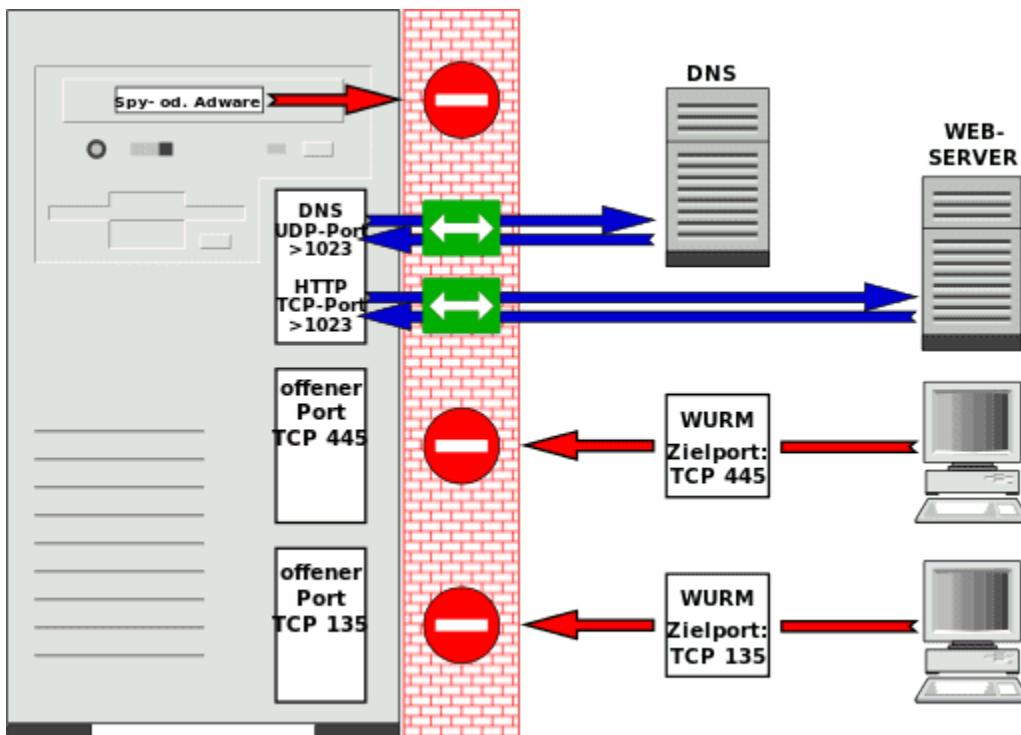
И тут мы добираемся до уровня 4 - транспортный. Посылки ведь бывают разные и их можно доставлять по разному. К примеру, в американских фильмах почтальоны у дома бросают газеты по утрам. Подует ветер, унесёт собака или ещё что - не страшно, всего лишь газета. Допустим, в сетях так работает доставка «пакетов времени» - NTP. Ваш компьютер может запрашивать у сервера текущее время и сервер отправляет вашему компьютеру ответ. И это происходит, допустим, раз в 10 минут. Но если вдруг ваш компьютер не получит ответ - небольшой сбой в сети - ничего страшного. Вы всё равно через 10 минут заново запросите. То есть пакет отправляется, но отправителю не нужно знать, доставился ли пакет или нет. Такой протокол доставки называется UDP.



Но, зачастую, в пакетах может быть важная информация и потерять её не хотелось бы. В таких случаях используют другой протокол - TCP. При TCP соединении отправитель должен убедиться, что пакет дошёл, для этого он ждёт подтверждения от второй стороны. И если в течение определённого времени ответа нет, он заново отправляет копию пакета.



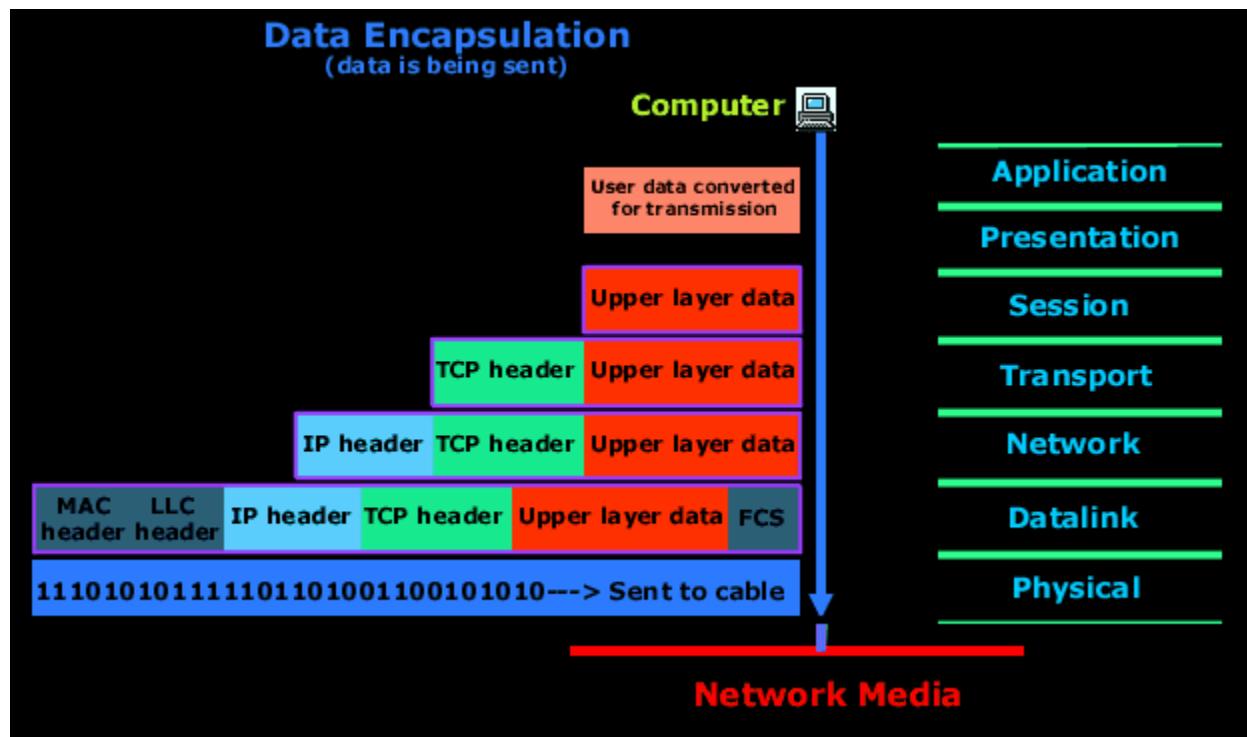
На этом же уровне мы с вами познакомимся с такой вещью, как порты. Не физические порты, а tcp или udp порты. Если компьютер - это дом, то порт - это квартира в этом доме. И абсолютно все дома - многоквартирные. Вы не можете просто принести пакет и положить в дом - курьер должен забрать от двери и доставить до двери. За каждой дверью живёт определённый человек, выполняющий определённую задачу. Допустим, когда вы заходите на сайт через браузер, ваш комп использует какой-то порт для отправки запроса. Браузер знает, что сайты выдаёт вебсервер, а он живёт на 80 порту. И курьер идёт в определённый дом с таким-то IP адресом и стучится в 80 дверь. Ждёт сколько-то. Если вебсервер действительно там живёт, то он отвечает курьеру, отдаёт ему пакет и курьер возвращает обратно ответ на тот порт, с которого был отправлен запрос. Если же никого за 80 дверью нет - грустный курьер возвращается ни с чем и браузер говорит, что не смог достучаться до вебсервера. Портов может быть 65 тысяч, при этом, обычно, для приёма используется 3-4 порта, а порты для исходящих запросов выдаются динамически и зависят от количества соединений.



И тут же познакомимся с таким понятием, как фаервол. Вы можете не хотеть, чтобы кто-то кроме вас или определённые люди могли подключаться к каким-то портам вашего сервера. И вы можете поставить вахтёра, который будет встречать курьеров на входе в дом - если кто-то не будет соответствовать её правилам, она просто запретит курьеру входить. Правила обычно выглядят так - от каких адресов к каким портам есть доступ, или к каким нету. Такие фаерволы называются персональными и стоят в самой системе. Очень часто на роутерах ставят сетевые фаерволы - они уже проверяют на входе в сеть, всё равно что посты на въезде в город.

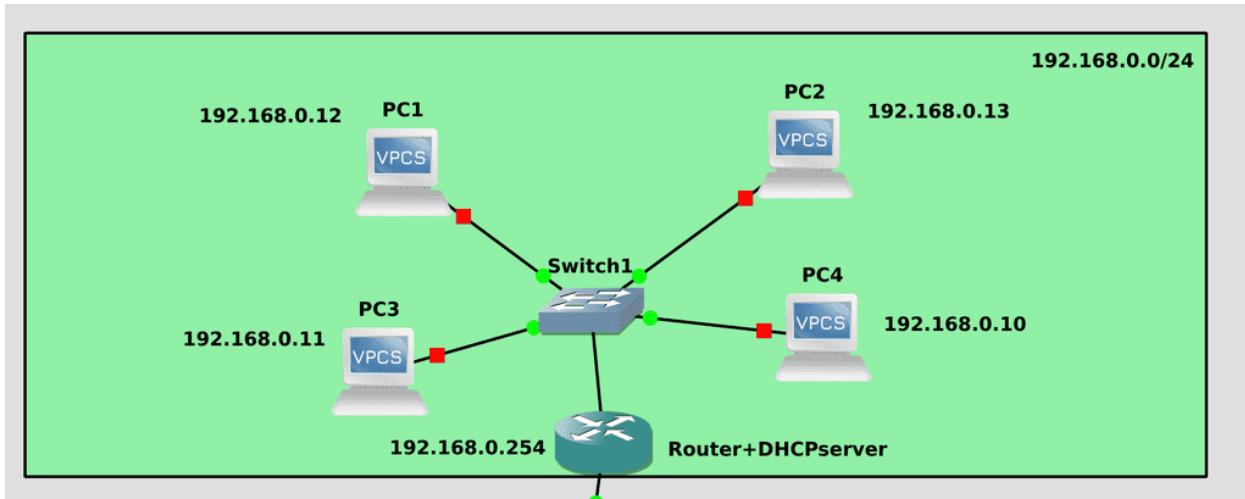
За tcp и udp портами уже находятся сами приложения - обычно это какие-то демоны в системе. Это уже 5, 6 и 7 уровни модели OSI. Один порт - одна программа, но у программы может быть несколько портов. Допустим, вебсервер использует как 80 порт, так и 443. Есть стандарты, какие программы за какими портами живут. Это можно поменять, но зная, что делаешь. Скажем, если поменять стандартные порты вебсервера, то браузеры не будут открывать сайты, пока не укажешь в браузере порт вручную. Но обычные пользователи этого не знают и откуда им знать, на какой порт вы поменяли?

На этом уровне уже работают программы, тот же самый вебсервер и браузер. Обе программы для взаимодействия используют протоколы, например, HTTP. Как и браузеры могут быть разные, так и программы на серверах, выступающие вебсервером. И именно использование единого протокола HTTP позволяет всем работать со всеми, независимо от браузера или вебсервера или операционной системы.

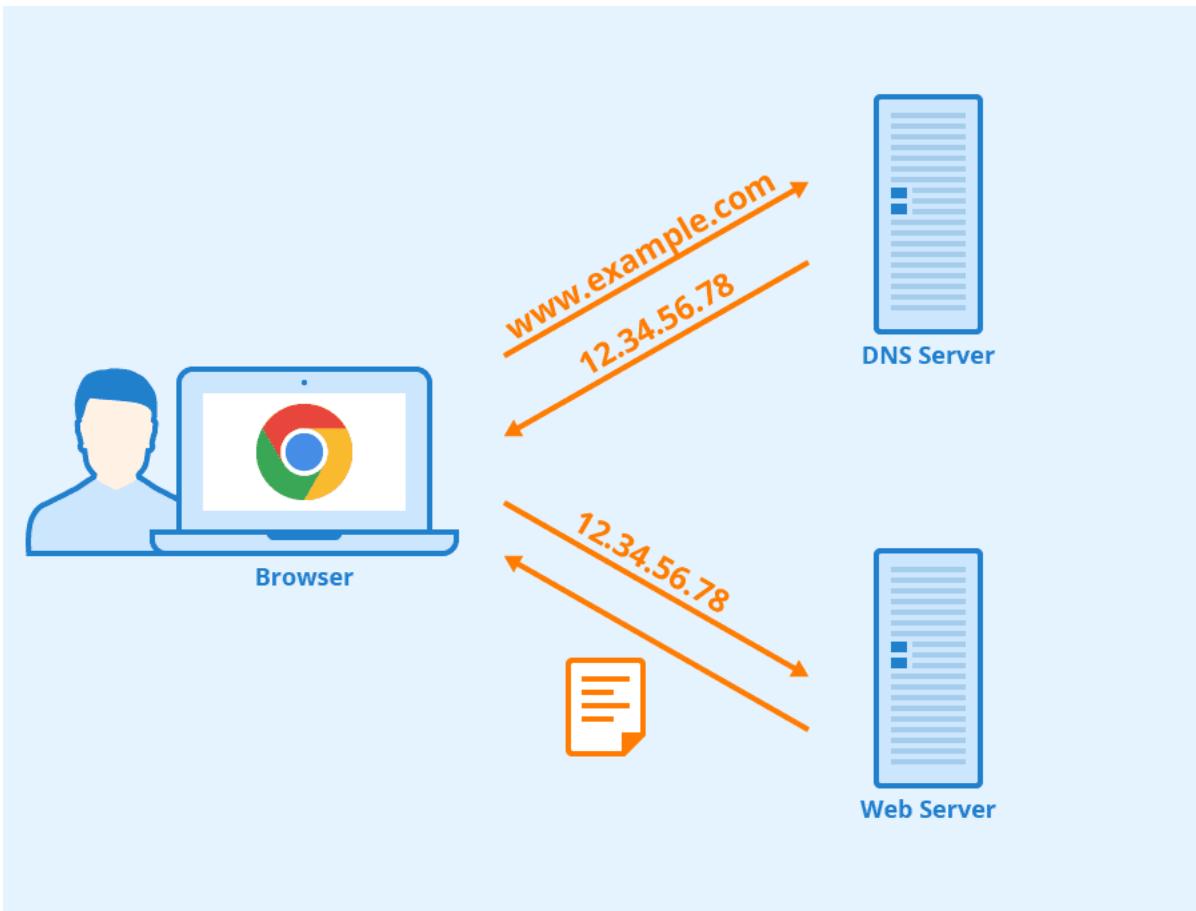


Итак, мы с вами вкратце рассмотрели модель OSI. И когда один компьютер пытается послать что-то другому, этот процесс напоминает упаковку коробок и отправку. Для примера, вы хотите открыть какую-то страничку в интернете. Ваш браузер упаковывает ваш запрос в коробку, на которой пишет информацию согласно протоколу HTTP. Дальше ваша операционная система берёт эту коробку и на克莱ивает на него ещё одну наклейку, в которой добавляет информацию по протоколу TCP, с какого source порта был отправлен запрос и на какой destination порт. Потом добавляет ещё одну наклейку - с какого IP адреса был запрос и на какой. Потом ещё одну наклейку - с какого мак адреса и на какой. Потом компьютер преобразует полученный пакет в 0 и 1 и отправляет на свитч. Весь этот процесс называется энкапсулацией. Свитч преобразует начало - и смотрит, на какой мак адрес нужно отправить - на роутер. Видит в таблице мак адрес роутера - 12 порт. Свитч отправляет на роутер. Роутер раскрывает начало и смотрит, а на какой адрес нужно отправить. Допустим, этот адрес роутеру неизвестен - он отклеивает наклейку l3 и клеит свою, заменяя ваш IP адрес на свой, для NAT-а, и оставляя

вашу наклейку у себя в архиве. Даётся отправляет на свой шлюз - другой роутер. Другой роутер тоже раскрывает, смотрит, не знает - отправляет на свой шлюз - третий роутер. Где-то в промежутке есть ещё свитчи и ещё роутеры и так десятки раз, пока не дойдёт до нужного сервера. В итоге доходит до нужного роутера, который видит нужный сервер в своей сети, он переклеивает наклейку с мак адресом и отправляет на нужный сервер. Сервер преобразует 0 и 1 в данные и видит свой мак адрес, понимает, что это ему. Начинает дальше отклеивать наклейки - видит, что запрос пришёл на его IP адрес. Снимает ещё одну наклейку - видит, tcp порт 80. За последними двумя шагами следует firewall - а можно ли доставлять пакеты на 80 порт с этого IP адреса? Если можно, пакет по итогу доходит до вебсервера. Потом вебсервер готовит ответ, распихивает по разным пакетам, начинает энкапсулировать и отправлять на свой роутер. И весь этот процесс происходит в доли секунд, прозрачно для пользователя, проходя огромные расстояния через сушу и океаны.

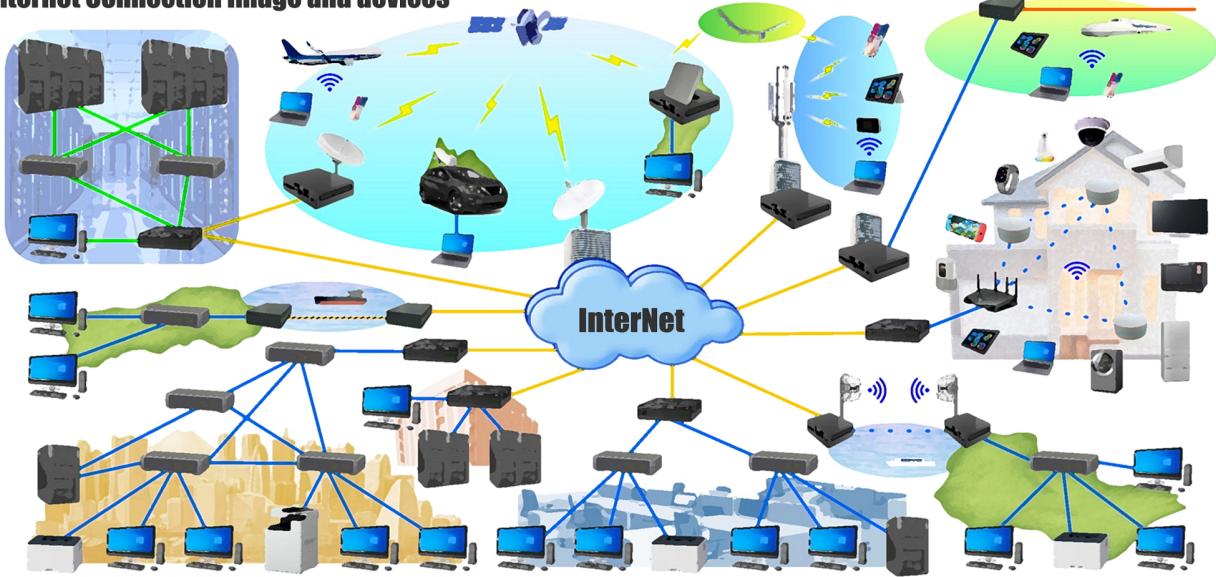


Ладно, как два компьютера общаются разобрались. Теперь стоит упомянуть два важных протокола, обеспечивающих удобство работы. Без них можно, но сложно. Начнём с DHCP. Это протокол позволяет выдавать IP адреса динамически. В отличии от MAC адреса, IP адрес не выдаётся на заводе, на каждом компьютере он настраивается отдельно. Но нельзя же каждому пользователю вручную писать свой адрес. Во-первых, это требует определённых знаний, во-вторых - просто неудобно, когда у вас большое количество компьютеров. Поэтому очень часто в сети есть DHCP сервер - им может быть и домашний роутер, и умный свитч, и даже отдельная виртуалка с линуксом. Когда какой-то компьютер подключается к сети, он отправляет всем в этой сети специальный запрос, мол есть ли в этой сети DHCP сервера? Этот запрос видят DHCP сервер и отвечает пользователю. DHCP решает, какой именно IP адрес выдать из свободных и даёт пользователю этот адрес, вместе с маской подсети, адресом шлюза и другими настройками. На серверах, обычно, IP адреса выдаются статически, т.е. вручную и на постоянно, а на компьютерах пользователей динамически. Т.е. эти адреса могут изменяться при следующем включении.



Второй протокол - DNS - преобразует IP адреса в имена, имена в IP адреса и не только. Вы же в браузере не пишете IP адреса гугла? У гугла серверов много, откуда вам знать и зачем вам помнить их IP адреса? В браузере вы пишете доменное имя, к которому хотите подключиться - google.com. При этом ваш браузер отправляет DNS запрос на специальные сервера. Они у вас также настраиваются при настройке сети, могут выдаваться DHCP сервером. Так вот, написали вы в браузере google.com - ваш браузер отправляет запрос с этим именем на прописанный в системе DNS сервер. DNS сервер либо знает этот адрес, либо обращается к другим DNS серверам, которые также могут обращаться дальше, пока не найдут ответ, кто же находится за этим именем. В итоге DNS возвращает вам IP адрес и ваш браузер отправляет HTTP запрос на этот адрес.

Internet Connection Image and devices



Сеть - громадный и сложный механизм, который для конечных пользователей упрощён до уровня выключателя. В средних и больших компаниях сеть администрируют целые отделы сетевых администраторов, которые изучают сети годами. Для администрирования линукс серверов такого объёма знаний не нужно, однако само понятие сервер предполагает, что к нему подключаются другие компьютеры - будь то в локальной сети или по интернету. И поэтому администратору нужны знания работы с сетью. С изучением дальнейших тем и опытом придёт более глубокое понимание.

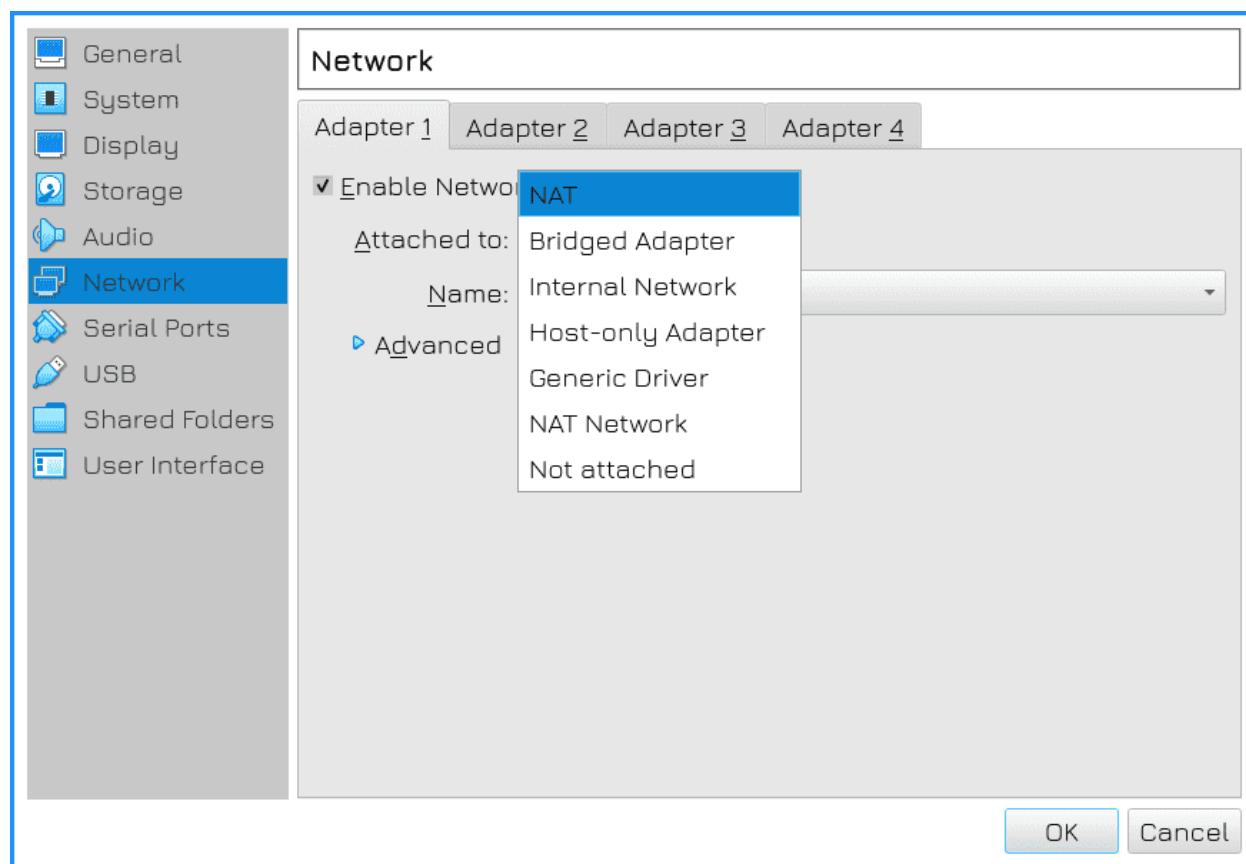
2.42.2 Практика

Вопросы

1. Как происходит обжим витой пары?
 2. Что такое MAC адрес, какая у него битность?
 3. Можно ли назвать MAC адрес «условно уникальным»?
 4. Что такое IP адрес?
 5. С помощью чего IP адреса делятся на подсети?
 6. Для чего нужен broadcast?
 7. В чем отличие роутера от свитча?
 8. Опишите как компьютер отправляет информацию в другую подсеть?
 9. Какие проблемы решает NAT? от чего он может защитить?
 10. Что такое TCP и UDP, и чем они отличаются друг от друга?
 11. Как решить проблему: «Компьютеров в сети стало слишком много, чтобы выдавать им IP адреса вручную»?
 12. Какой протокол используется, чтобы преобразовать название сайта gnuLinux.pro в IP адрес?

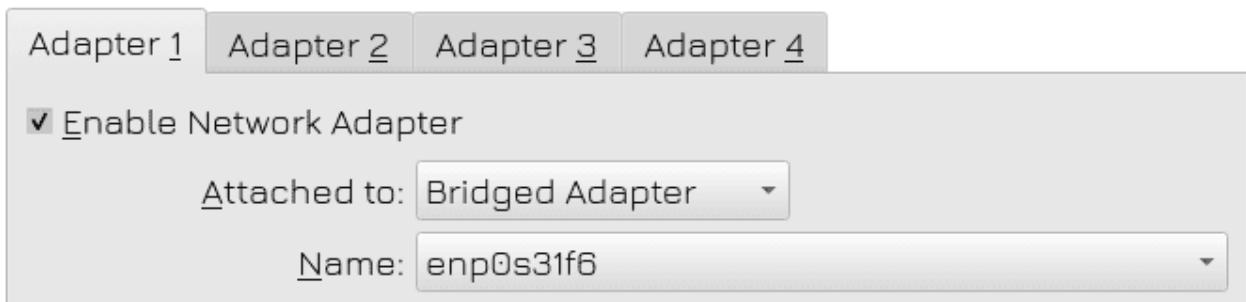
Задания

1. Запишите /22 маску в виде: x.x.x.x, сколько в ней адресов?
2. Приведите примеры протоколов использующих TCP и UDP
3. Соотнесите следующее, с уровнями OSI:
 - Ethernet
 - IP
 - HTTP
 - Витая пара
 - UDP
 - DNS

2.43 43. Работа с сетью**2.43.1 43. Работа с сетью**

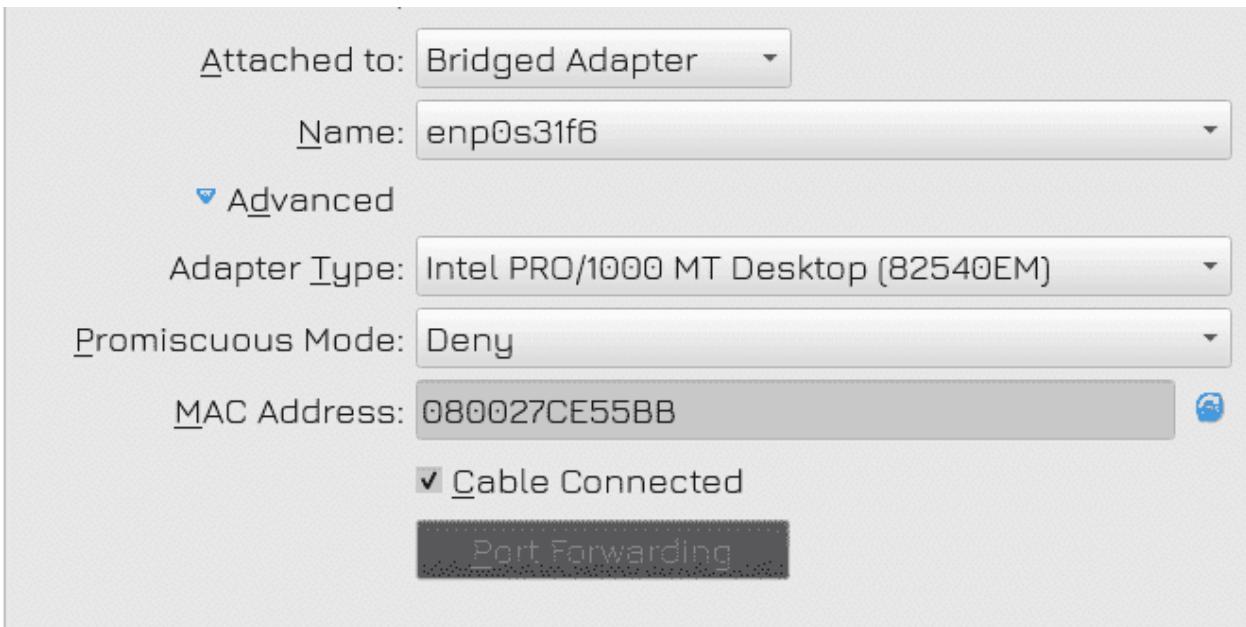
С теорией сетей немного разобрались, попробуем реализовать. Мы с вами работаем с виртуалкой, а за её сеть отвечает гипервизор. Зайдите в настройки виртуалки - Сеть. Наверху есть 4 вкладки - виртуалке можно выделить 4 сетевых адаптера. Нам пока хватает одного, но в дальнейшем это пригодится. В выпадающем меню «Тип подключения» указано, в какую сеть будет смотреть виртуалка. Допустим, NAT означает, что гипервизор будет выступать в роли роутера с включённым NAT-ом для виртуальной машины, точно как ваш домашний роутер для вашего компьютера. Как и домашний роутер, гипервизор

выдаст IP адрес с помощью DHCP виртуальной машине. Из самой виртуалки будет доступ в вашу домашнюю сеть и интернет, а из вашей сети, скажем, с телефона, не будет прямого доступа к виртуалке.



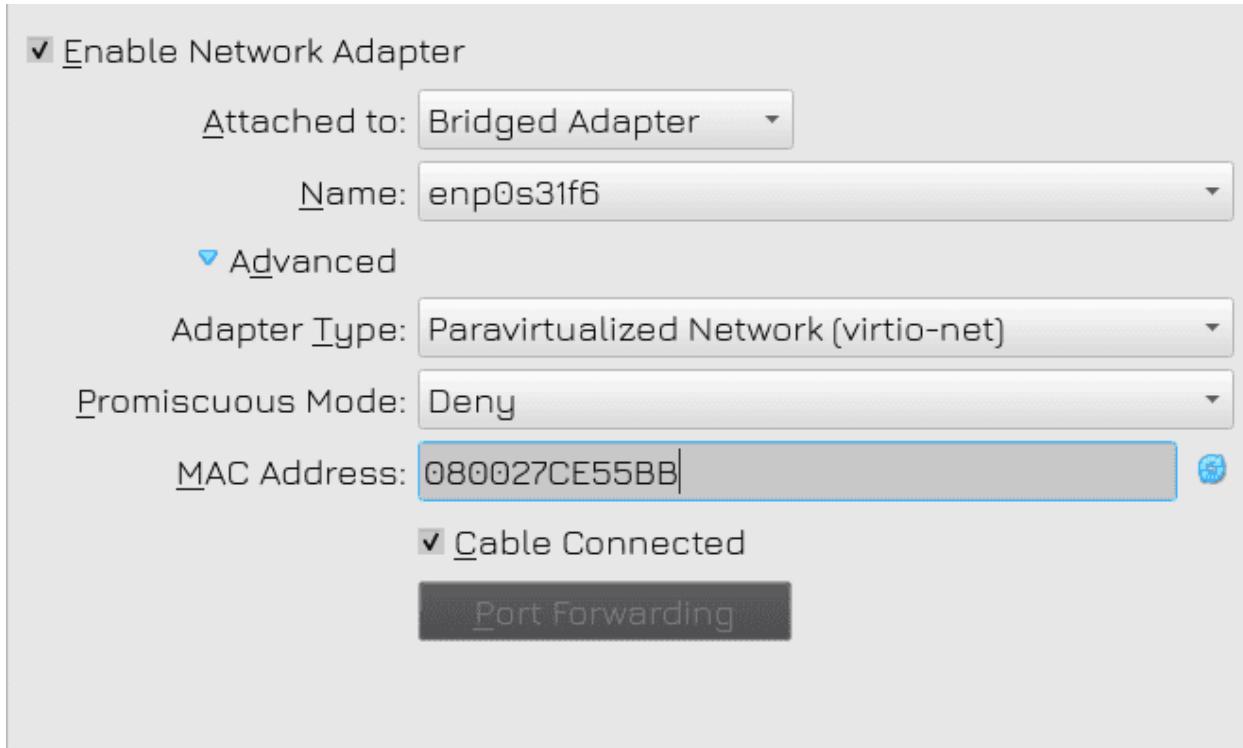
Второй тип сети - это сетевой мост - bridged adapter. В таком случае гипервизор будет выступать как свитч, благодаря чему виртуалка окажется в вашей домашней сети. Она получит IP адрес от вашего роутера, так как он выступает DHCP сервером и будет доступна для любого устройства в вашей домашней сети. Дальше нужно указать имя адаптера вашего компьютера, который подключён к сети. Это может быть ethernet или wifi адаптер.

Другие типы подключения нам пока не интересны, но в целом они позволяют создать изолированную сеть между виртуалками, либо сеть, которая только между хостом и виртуальной машиной и т.п., под различные сценарии.



Идём в дополнительные настройки. Тут можно выбрать тип адаптера. Виртуалбокс эмулирует сетевой адаптер и здесь можно выбрать какой именно. Какие-то старые операционные системы могут не работать с определёнными типами, какие-то быстрее, какие-то функциональнее. Чуть больше о различиях можете почитать по [ссылке](#). В большинстве случаев с современными системами лучше использовать Intel PRO/1000 MT, а если у вас виртуалка с линуксом, можете использовать virtio-net - он не эмулирует виртуальную сетевую карту, а использует другой механизм, который чуть производительнее. Но для гостевых Windows-ов нужно будет устанавливать дополнительные драйвера.

Неразборчивый режим позволяет направлять весь приходящий трафик на эту сетевую карту. По умолчанию, когда стоит Deny, виртуалка будет видеть только трафик, предназначенный ей, так как сам виртуалбокс направляет трафик по мак адресам. Но в определённых случаях, для решения проблем или детального разбора трафика, здесь можно настроить другое значение.



Ну и снизу у нас MAC адрес. Я говорил, что мак адрес состоит из 12 символов, разделённых двоеточиями по два, но в некоторых программах можно увидеть и такое написание, это нормально. При необходимости это значение можно поменять, так как это виртуальный сетевой адаптер, то есть мак адреса здесь генерируются самим гипервизором.

Я сохраняю эти настройки и запускаю виртуалку.

```
[user@centos8 ~]$ ip
address      ila          maddress    neigh       route       token       xfrm
addrlabel    l2tp         monitor     netconf     rule        tunnel
fou          link         mroute     netns       sr         tuntap
help         macsec       mrule      ntable     tcp_metrics vrf
[user@centos8 ~]$ ip
```

Одна из главных команд по работе с сетью:

```
ip
```

У неё множество различных ключей и опций, но чаще всего используются:

```
ip address
ip route
```

Эта команда позволяет не дописывать ключи до конца, по примеру того, как сделано на сетевом оборудовании. Например, команду:

```
ip address show
```

можно написать как:

```
ip addr
ip a
```

```
[user@centos8 ~]$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:ce:55:bb brd ff:ff:ff:ff:ff:ff
    inet 192.168.31.112/24 brd 192.168.31.255 scope global dynamic noprefixroute enp0s3
        valid_lft 42103sec preferred_lft 42103sec
    inet6 fe80::abea:bda9:f72d:1de5/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
3: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default qlen 1000
    link/ether 52:54:00:b4:48:24 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
        valid_lft forever preferred_lft forever
4: virbr0-nic: <BROADCAST,MULTICAST> mtu 1500 qdisc fq_codel master virbr0 state DOWN group default qlen 1000
    link/ether 52:54:00:b4:48:24 brd ff:ff:ff:ff:ff:ff
```

Эта команда позволяет увидеть список IP адресов, мак адресов, интерфейсов и их параметры. А что такое интерфейсы? Физические порты на сетевых адаптерах правильнее называть физическими интерфейсами. А даваемые настройки в операционной системе применяются на программную составляющую этого интерфейса, называемую логическим интерфейсом. И в обычной речи, когда говорят порт, имеют ввиду физический интерфейс, а когда говорят интерфейс - имеют ввиду логический интерфейс. С точки зрения операционной системы мы работаем с интерфейсами, а не с портами. Т.е. вставляешь кабель в порт, а даёшь IP адрес на интерфейс.

Итак, команда:

```
ip a
```

вывела нам 4 интерфейса - lo, enp0s3, virbr0 и virbr0-nic. Последние два нас не интересуют, это тот же самый сетевой мост. Обычно сетевой мост создаётся на хосте, но тут видимо при установке какого-то пакета он создался автоматом. И, в общем, просто проигнорируем эти интерфейсы.

Начнём с lo - loopback. Это специальный интерфейс, который есть в каждой системе. Некоторые программы, особенно работающие с сетью, иногда требуют подключения к этому же компьютеру. Скажем, у вас две программы, работающие через сеть, должны общаться друг с другом, например, вебсервер и система управления базой данных. Вы можете поднять два компьютера, а можете всё держать на одном. И чтобы эти программы общались, в них вы прописываете адрес 127.0.0.1. Это адрес, который есть на всех компьютерах, он указывает компьютеру сам на себя. Многие внутренние компоненты системы могут использовать этот адрес, поэтому он необходим.

```
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:ce:55:bb brd ff:ff:ff:ff:ff:ff
    inet 192.168.31.112/24 brd 192.168.31.255 scope global dynamic noprefixroute enp0s3
        valid_lft 42103sec preferred_lft 42103sec
    inet6 fe80::abea:bda9:f72d:1de5/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

Нас больше интересует enp0s3. Это и есть интерфейс адаптера, который мы настроили на virtualbox. Самый достоверный способ понять - посмотреть на мак адрес, который отображается после link/ether. Достаточно сравнить последние 4 символа - 55bb - тоже самое у нас было в настройках виртуалбокса. Собственно в строке inet видим IP адрес - 192.168.31.112 - это адрес из моей домашней сети. Рядом с ним маска /24 - т.е. в моём случае адрес сети - 192.168.31.0, броадкаст - 192.168.31.255, а диапазон адресов - от 1 до 254. Чуть ниже строка inet6 - это IP адрес по IPv6. Это и многие другие параметры

я опущу, чтобы не усложнять.

```
[user@centos8 ~]$ sudo ip link set enp0s3 down
[sudo] password for user:
[user@centos8 ~]$ ip a show enp0s3
2: enp0s3: <BROADCAST,MULTICAST> mtu 1500 qdisc fq_codel state DOWN group default qlen 1000
    link/ether 08:00:27:ce:55:bb brd ff:ff:ff:ff:ff:ff
        inet 192.168.31.112/24 brd 192.168.31.255 scope global dynamic noprefixroute enp0s3
            valid_lft 38444sec preferred_lft 38444sec
[user@centos8 ~]$ sudo ip link set enp0s3 up
[user@centos8 ~]$ ip a show enp0s3
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:ce:55:bb brd ff:ff:ff:ff:ff:ff
        inet 192.168.31.112/24 brd 192.168.31.255 scope global dynamic noprefixroute enp0s3
            valid_lft 43199sec preferred_lft 43199sec
        inet6 fe80::abea:bda9:f72d:1de5/64 scope link noprefixroute
            valid_lft forever preferred_lft forever
[user@centos8 ~]$ █
```

Выше - state up - говорит о том, что интерфейс поднят, т.е. работает. Это означает, что интерфейс поднят как со стороны операционной системы, так и со стороны железа, т.е. вставлен кабель и есть подключение с другой стороны - со свитча, роутера, другого компьютера или гипервизора. Иногда, при проведении каких-то технических работ или по каким-то другим причинам, вам нужно выключить интерфейс, чтобы не было сети. Но при этом физически выдёргивать кабель или менять настройки гипервизора может быть не очень удобно. Поэтому интерфейс можно выключить в самой операционной системе. Для этого есть команда:

```
sudo ip link set enp0s3 down
ip address show enp0s3
```

Как видите, state теперь показывает Down. Это называется административным выключением интерфейса. Для включения интерфейса:

```
sudo ip link set enp0s3 up
ip address show enp0s3
```

Но то что интерфейс в UP-е ещё не говорит о том, что с сетью всё нормально. В сети может быть огромное количество проблем, которые могут влиять на работоспособность. Одна из самых базовых утилит диагностики:

```
ping
```

Она посылает пакет другому компьютеру и ждёт от него ответа. Другой компьютер, при виде такого пакета, отправляет ответ. И этот цикл повторяется бесконечно, пока мы не прекратим. Что-то наподобие пинг-понга, компьютеры сообщают друг-другу, что между ними есть связь.

```
[user@centos8 ~]$ ping _gateway
PING _gateway (192.168.31.1) 56(84) bytes of data.
64 bytes from XiaoQiang (192.168.31.1): icmp_seq=1 ttl=64 time=0.685 ms
64 bytes from XiaoQiang (192.168.31.1): icmp_seq=2 ttl=64 time=0.620 ms
64 bytes from XiaoQiang (192.168.31.1): icmp_seq=3 ttl=64 time=0.665 ms
64 bytes from XiaoQiang (192.168.31.1): icmp_seq=4 ttl=64 time=0.608 ms
^C
--- _gateway ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 79ms
rtt min/avg/max/mdev = 0.608/0.644/0.685/0.040 ms
```

Попробуем кого-нибудь пингануть. Я не буду пинговать хост, так как, если у вас Windows, на нём

файрвол блокирует пинги. Т.е. если вы будете пинговать Windows с вашей виртуалки, ответа вы не получите. Поэтому возьмём что-то универсальное, например, ваш роутер. Чтобы пинговать кого-то, нужно знать его IP адрес. Так как виртуалка получила IP адрес по DHCP, оттуда же она получила информацию о gateway - шлюзе, откуда можно выходить в другие сети. В этот момент systemd создаёт специальную запись для шлюза, чтобы, независимо от IP адреса, был универсальный способ обратиться к шлюзу - `_gateway`. Попробуем пингануть:

```
ping _gateway
```

Система преобразовала `_gateway` в IP адрес - 192.168.31.1 и теперь пингует мой роутер.

В ответах есть диагностическая информация - размер отправляемого пакета, номер по очереди, ttl и time - время, за которое пакет был доставлен до другого компьютера и получен ответ. Что такое ttl? Когда мы кого-то пингуем, мы назначаем специальное число - ttl, в данном случае по умолчанию 64. Каждый раз, когда наш пакет будет проходить через роутер, тот будет отнимать единичку от этого числа. Если до какого-то роутера дойдёт число 0, то он выбросит этот пакет, так как что-то явно пошло не так и пакет слишком долго блуждает по сети.

```
[user@centos8 ~]$ ping 192.168.31.2
PING 192.168.31.2 (192.168.31.2) 56(84) bytes of data.
From 192.168.31.112 icmp_seq=1 Destination Host Unreachable
From 192.168.31.112 icmp_seq=2 Destination Host Unreachable
From 192.168.31.112 icmp_seq=3 Destination Host Unreachable
^C
--- 192.168.31.2 ping statistics ---
6 packets transmitted, 0 received, +3 errors, 100% packet loss, time 123ms
pipe 4
[user@centos8 ~]$ █
```

Итак, gateway у нас пинговался, теперь возьмём какой-нибудь несуществующий адрес, например, 192.168.31.2:

```
ping 192.168.31.2
```

За этим адресом никакого компьютера нет. В таком случае ping не может достучаться до нужного ip адреса, вследствие чего выдаёт «Destination host unreachable», т.е. целевой хост недоступен.

```
[user@centos8 ~]$ ping 1.1.1.1
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.
64 bytes from 1.1.1.1: icmp_seq=1 ttl=55 time=80.0 ms
64 bytes from 1.1.1.1: icmp_seq=2 ttl=55 time=79.2 ms
^C
--- 1.1.1.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1ms
rtt min/avg/max/mdev = 79.238/79.624/80.011/0.478 ms
[user@centos8 ~]$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=57 time=54.6 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=57 time=53.5 ms
^C
--- 8.8.8.8 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 2ms
rtt min/avg/max/mdev = 53.454/54.009/54.564/0.555 ms
[user@centos8 ~]$ █
```

То, что у меня пингуется gateway говорит о том, что локальная сеть у меня доступна, я могу достучаться до другого компьютера в этой же сети. Но что насчёт других сетей? Например, как проверить, доступен ли интернет? Есть пара адресов, которые знают все администраторы - 1.1.1.1 и 8.8.8.8:

```
ping 1.1.1.1
```

Это адреса DNS серверов Cloudflare и Google. Если пинги доходят до этих адресов - значит у вас есть интернет. В каких-то компаниях может стоять сетевой фаервол, который блокирует исходящие пинги, но я такие ситуации не учитываю. Итак, интернет у нас есть.

```
[user@centos8 ~]$ ip route show
default via 192.168.31.1 dev enp0s3 proto dhcp metric 100
192.168.31.0/24 dev enp0s3 proto kernel scope link src 192.168.31.112 metric 100
192.168.122.0/24 dev virbr0 proto kernel scope link src 192.168.122.1 linkdown
[user@centos8 ~]$ █
```

Но как вообще система понимает, до какого компьютера как добраться? Есть такая вещь, как таблица маршрутизации:

```
ip route show
```

Здесь указано, что для любого компьютера в сети 192.168.31.0 используй интерфейс enp0s3, для сети 192.168.122.0 - интерфейс virbr0. А вот если что-то другое, что-то с непонятной сетью, допустим, 8.8.8.8 - обращайся к 192.168.31.1. Это дефолтный маршрут, который и называют gateway. У этого роутера есть своя таблица маршрутизации, где также написана локальная сеть, а если что-то неизвестное - обращайся к роутеру провайдера. У провайдера, конечно, больше маршрутов, он связан с другими провайдерами и всё такое, но логика та же - если знаешь сеть, иди туда, если нет - отправляй на gateway. Таким образом ваш пакет в итоге доходит до того же гугла.

```
[user@centos8 ~]$ traceroute 1.1.1.1
traceroute to 1.1.1.1 (1.1.1.1), 30 hops max, 60 byte packets
 1 XiaoQiang (192.168.31.1) 1.409 ms 1.267 ms 1.177 ms
 2 10.128.4.1 (10.128.4.1) 2.137 ms 2.056 ms 2.246 ms
 3 host-149.255.155.121.katv1.net (149.255.155.121) 1.854 ms 1.781 ms 1.985 ms
 4 100.127.1.1 (100.127.1.1) 1.879 ms * *
 5 DeltaTelecom-AG-Telecom-link-for-INTERNET-Xchange.AZ-IX.net (94.20.50.181) 3.589 ms 7.418 ms 8
5.132.80.217 (85.132.80.217) 4.363 ms
 6 10.50.10.182 (10.50.10.182) 60.347 ms 10.50.10.186 (10.50.10.186) 73.356 ms 10.50.10.182 (10.5
0.10.182) 58.211 ms
 7 ams-ix.as13335.net (80.249.211.140) 90.021 ms 89.513 ms de-cix-frankfurt.as13335.net (80.81.19
4.180) 68.450 ms
 8 * one.one.one.one (1.1.1.1) 82.329 ms 80.200 ms
[user@centos8 ~]$
```

Система также позволяет узнать, через какие роутеры проходит пакет. Хотя провайдеры могут скрывать некоторые свои роутеры, но общую картину можно понять с помощью утилиты traceroute:

```
traceroute 1.1.1.1
```

Как видите, сначала мой пакет добирается до моего роутера, потом до роутера провайдера и так ещё пару маршрутизаторов, пока не доберётся до самого IP адреса.

```
[user@centos8 ~]$ traceroute 8.7.6.5
traceroute to 8.7.6.5 (8.7.6.5), 30 hops max, 60 byte packets
 1 XiaoQiang (192.168.31.1) 1.659 ms 1.485 ms 1.384 ms
 2 10.128.4.1 (10.128.4.1) 2.187 ms 2.106 ms 2.025 ms
 3 host-149.255.155.121.katv1.net (149.255.155.121) 1.924 ms 5.506 ms 1.912 ms
 4 100.127.1.1 (100.127.1.1) 1.559 ms * *
 5 85.132.80.217 (85.132.80.217) 8.011 ms 6.920 ms 5.928 ms
 6 10.50.10.186 (10.50.10.186) 74.741 ms 74.360 ms 10.50.10.182 (10.50.10.182) 55.313 ms
 7 * * *
 8 * * *
 9 * * *
10 * * *
```

Если у вас не доступен какой-то компьютер или не работает интернет, вы можете запустить traceroute и увидеть, после какого маршрутизатора застревает пакет:

```
traceroute 8.7.6.5
```

Скажем, если пакет доходит до вашего роутера и застревает - проблема либо с ним, либо с сетью между роутером и провайдером. Если же пакет доходит до следующего роутера - роутера провайдера - то значит у вас всё в порядке, проблема на стороне провайдера. В данном примере пакет застрял на каком-то неизвестном маршрутизаторе - т.е. дело, скорее всего, не в провайдере, просто сам этот адрес недоступен.

```
[user@centos8 ~]$ ping google.com
PING google.com (172.217.17.238) 56(84) bytes of data.
64 bytes from sof02s41-in-f14.1e100.net (172.217.17.238): icmp_seq=1 ttl=116 time=55.0 ms
64 bytes from sof02s41-in-f14.1e100.net (172.217.17.238): icmp_seq=2 ttl=116 time=54.5 ms
^C
--- google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1ms
rtt min/avg/max/mdev = 54.464/54.732/55.000/0.268 ms
[user@centos8 ~]$
```

Кроме IP адреса на доступность сайтов и сетевых сервисов может влиять DNS. Так как в большинстве случаев мы к сайтам обращаемся не по IP адресам, а по именам, допустим, google.com:

```
ping google.com
```

то в случае проблем с DNS у нас слово google.com не будет превращаться в IP адрес, а значит мы не сможем добраться до нужного компьютера. Тот же самый пинг позволяет увидеть, есть ли проблемы с DNS. Как видите, у меня google.com превратился в IP адрес, а значит с DNS проблем нет.

```
[user@centos8 ~]$ ping example.com
PING example.com (93.184.216.34) 56(84) bytes of data.
64 bytes from 93.184.216.34 (93.184.216.34): icmp_seq=1 ttl=54 time=160 ms
^C
--- example.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 159.668/159.668/159.668/0.000 ms
[user@centos8 ~]$ ping example.ru
ping: example.ru: Name or service not known
[user@centos8 ~]$
```

Для примера, если пингануть example.com:

```
ping example.com
```

мы увидим IP адрес, но при попытке пинга example.ru:

```
ping example.ru
```

видим ошибку - имя или сервис недоступны. Это означает, что DNS сервер не смог найти или не ответил, кто такой example.ru. Учитывая, что он отвечал на другие запросы, значит он просто не может найти такое имя.

```
[user@centos8 ~]$ cat /etc/resolv.conf
# Generated by NetworkManager
nameserver 192.168.31.1
[user@centos8 ~]$ █
```

Чтобы узнать, кто прописан DNS сервером на моём компьютере, можно посмотреть файл resolv.conf:

```
cat /etc/resolv.conf
```

Как видите, в качестве DNS сервера прописан мой роутер. Когда я пытаюсь зайти на какой-то сайт по имени, мой компьютер отправляет DNS запрос на роутер, тот отправляет этот запрос на другие DNS сервера - скорее всего, полученные от провайдера и так по цепочке, пока нужный DNS сервер не ответит. И, обычно, DNS сервера в цепочке запоминают этот ответ. И в следующий раз, если кто-то опять спросит это же имя, они достанут эту информацию из кэша, что позволит компьютеру быстрее получить ответ и быстрее зайти на сайт, а также позволит сократить трафик провайдера, так как он не запрашивает одну и ту же информацию для тысяч своих клиентов.

```
[user@centos8 ~]$ sudo nano /etc/resolv.conf
[user@centos8 ~]$ cat /etc/resolv.conf
# Generated by NetworkManager
nameserver 94.140.14.14
[user@centos8 ~]$ ping google.com
PING google.com (216.58.207.174) 56(84) bytes of data.
64 bytes from mucl1s04-in-f14.1e100.net (216.58.207.174): icmp_seq=1 ttl=114 time=72.10 ms
^C
--- google.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 72.974/72.974/72.974/0.000 ms
[user@centos8 ~]$
```

Давайте, для примера, поменяем DNS. Сам файл /etc/resolv.conf генерируется при каждом перезапуске сети или компьютера, но для теста мы изменим здесь:

```
sudo nano /etc/resolv.conf
```

```
nameserver 94.140.14.14
```

Этот IP адрес - адрес DNS сервера AdGuard. Их сервера не выдают нам информацию о доменах, которые используются для рекламы, а поэтому наш браузер не может прогрузить рекламные куски сайтов. Окей, протестируем:

```
ping google.com
```

```
[user@centos8 ~]$ nslookup ya.ru
Server: 94.140.14.14
Address: 94.140.14.14#53
```

Non-authoritative answer:

```
Name: ya.ru
Address: 87.250.250.242
Name: ya.ru
Address: 2a02:6b8::2:242
```

Но как мне убедиться, что я использую именно эти DNS сервера? Для этого можем использовать утилиту nslookup:

```
nslookup ya.ru
```

Вначале ответа пишется адрес сервера DNS - именно то, что я указывал. А внизу ответ - нужные IP адреса. nslookup - тоже как способ узнать доступность DNS серверов без необходимости пинга, да и в целом, чтобы найти нужные ответы по DNS.

```
[user@centos8 ~]$ sudo nano /etc/resolv.conf
[user@centos8 ~]$ cat /etc/resolv.conf
# Generated by NetworkManager
nameserver 8.7.6.5
[user@centos8 ~]$ ping google.com
ping: google.com: Name or service not known
[user@centos8 ~]$ nslookup google.com
;; connection timed out; no servers could be reached

[user@centos8 ~]$ █
```

Давайте ещё пропишем несуществующий DNS:

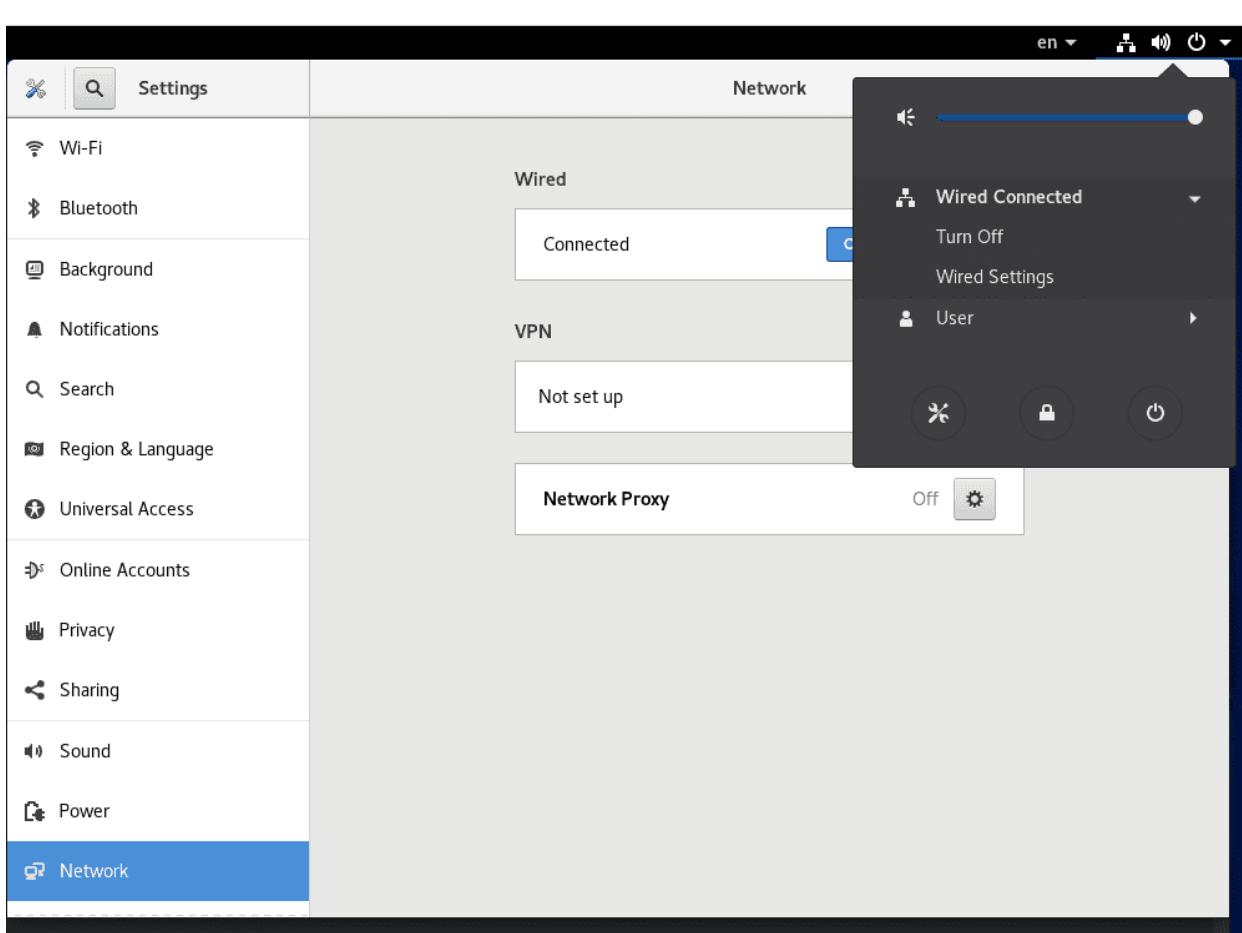
```
sudo nano /etc/resolv.conf
```

```
nameserver 8.7.6.5
```

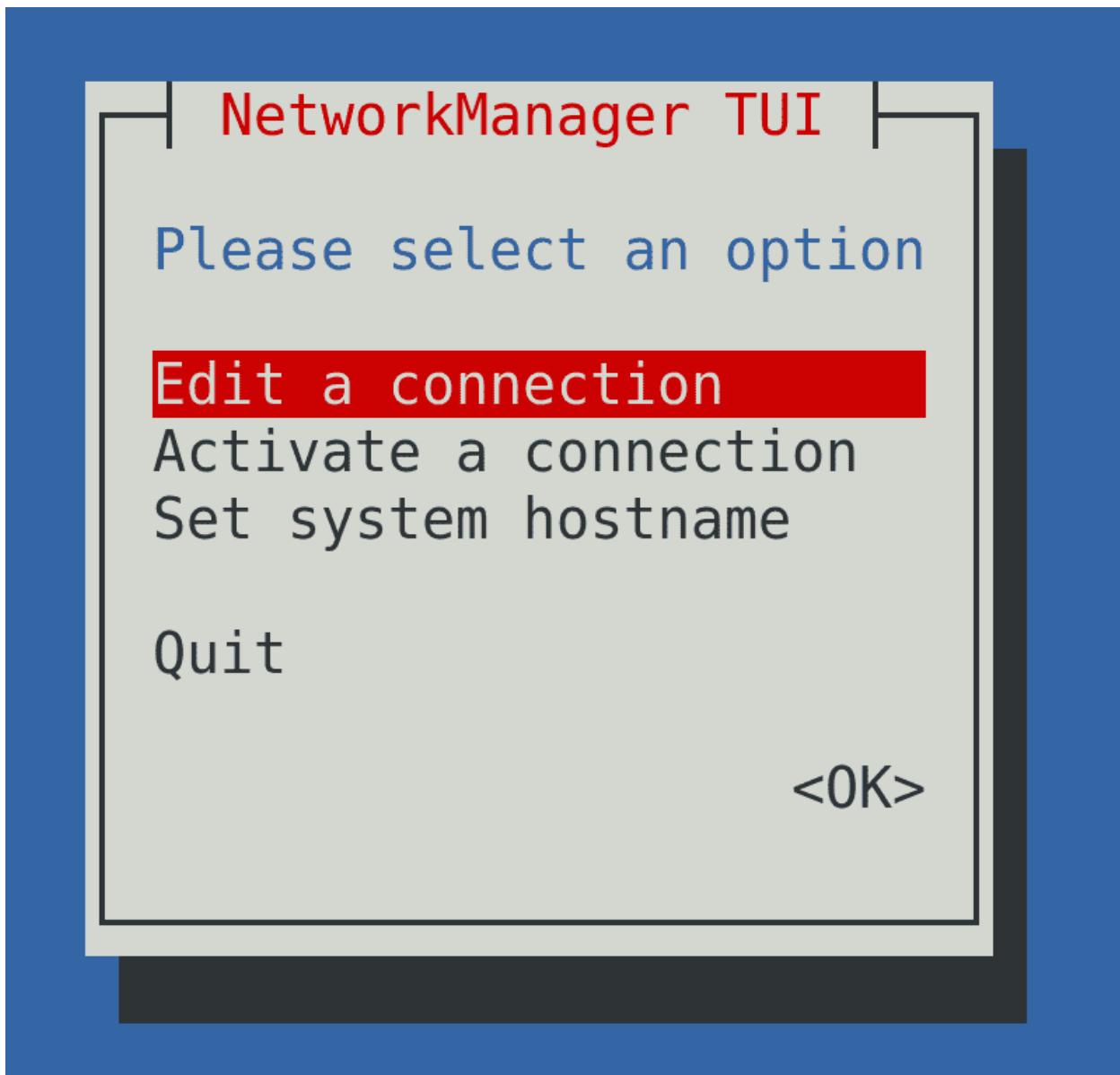
```
ping google.com
nslookup google.com
```

Эта ситуация сравнима с тем, что либо мы неправильно настроили DNS, либо DNS сервер недоступен. Тогда ping будет жаловаться, что это имя неизвестно, а nslookup скажет, что не может достучаться до DNS серверов.

Хорошо, с тем, как проверить работу сети, разобрались. Теперь поговорим о смене настроек. Как я говорил, /etc/resolv.conf генерируется при перезапуске сети или компьютера, т.е. наши изменения там - временные и чисто для тестов. Команда ip также позволяет временно добавить и удалить ip адрес или маршрут, а также поменять кое-какие настройки. Но эти изменения временные, поэтому это рассмотрим в другой раз.



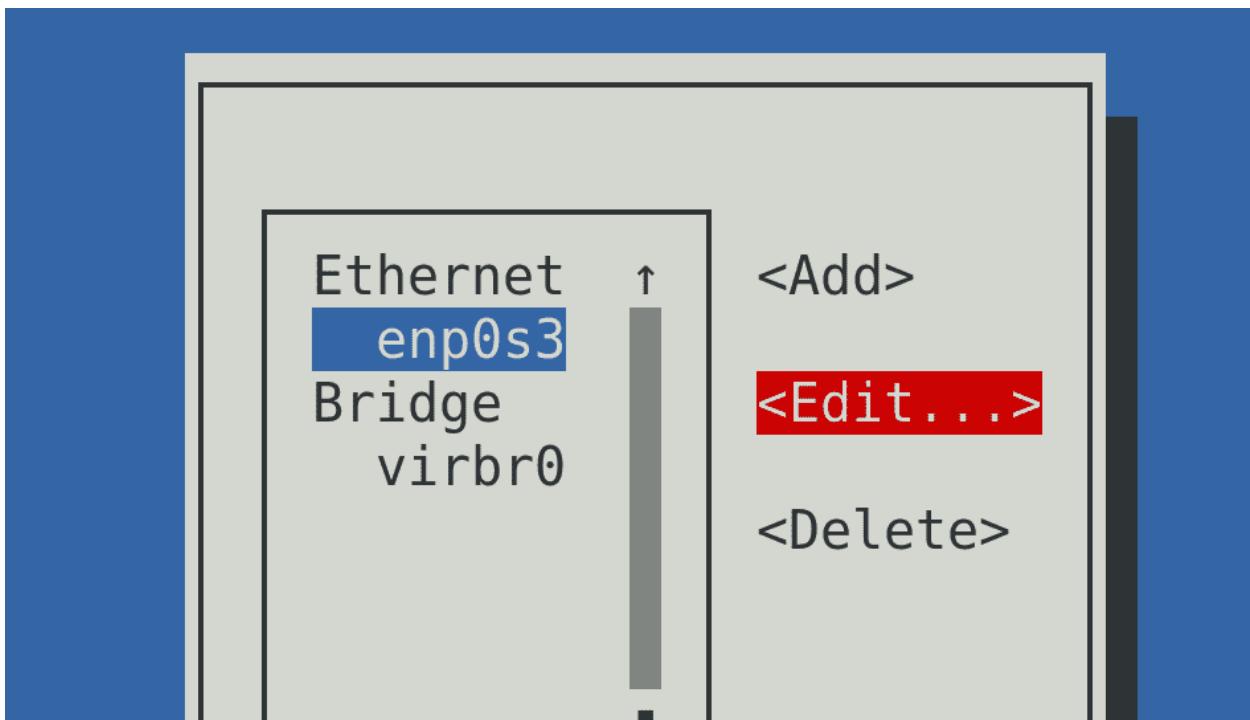
Для постоянных настроек используется сетевой демон NetworkManager. Этот демон есть почти на всех дистрибутивах - где-то он стоит по умолчанию, а где-то его можно установить. Он позволяет гибко управлять настройками и работать с сетью. У него есть 3 интерфейса - графический, псевдографический и текстовой. Если у вас на компе линукс, можете использовать графику - он простой и функционала зачастую хватает.



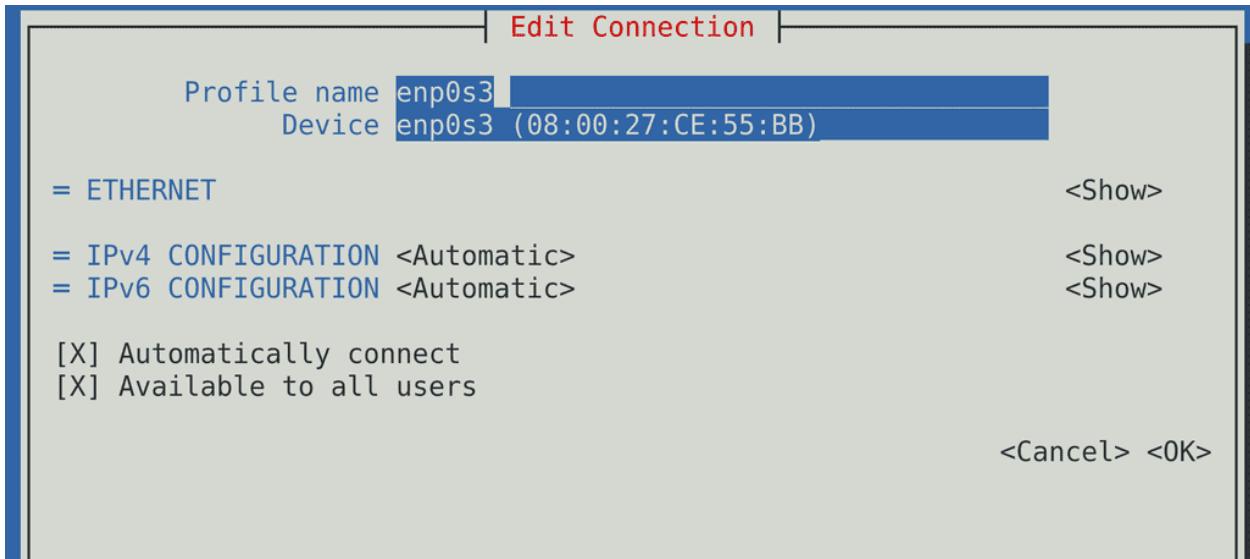
Однако на серверах графики нет, поэтому остаются 2 варианта. Для большинства задач подойдёт псевдографика:

```
sudo nmtui
```

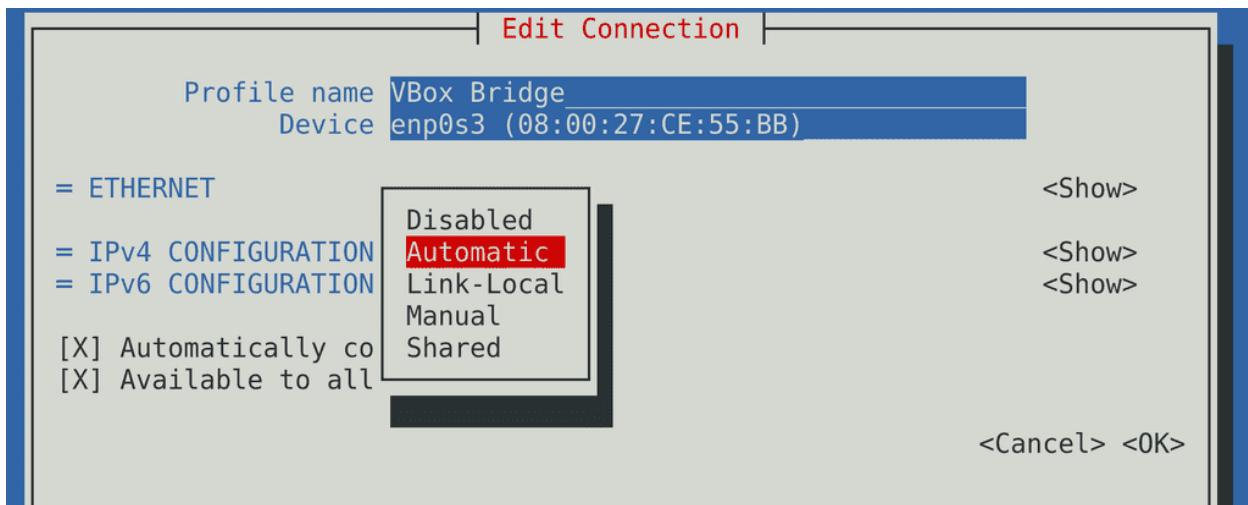
Давайте, для примера, выставим статичный IP. Это нужно, чтобы у компьютера не поменялся IP адрес, так как DHCP раздаёт IP адреса динамически. Т.е., скажем, если я завтра включу эту виртуалку, у неё может быть другой адрес. А если поставлю статический - то у неё этот адрес останется навсегда, или пока я не поменяю.



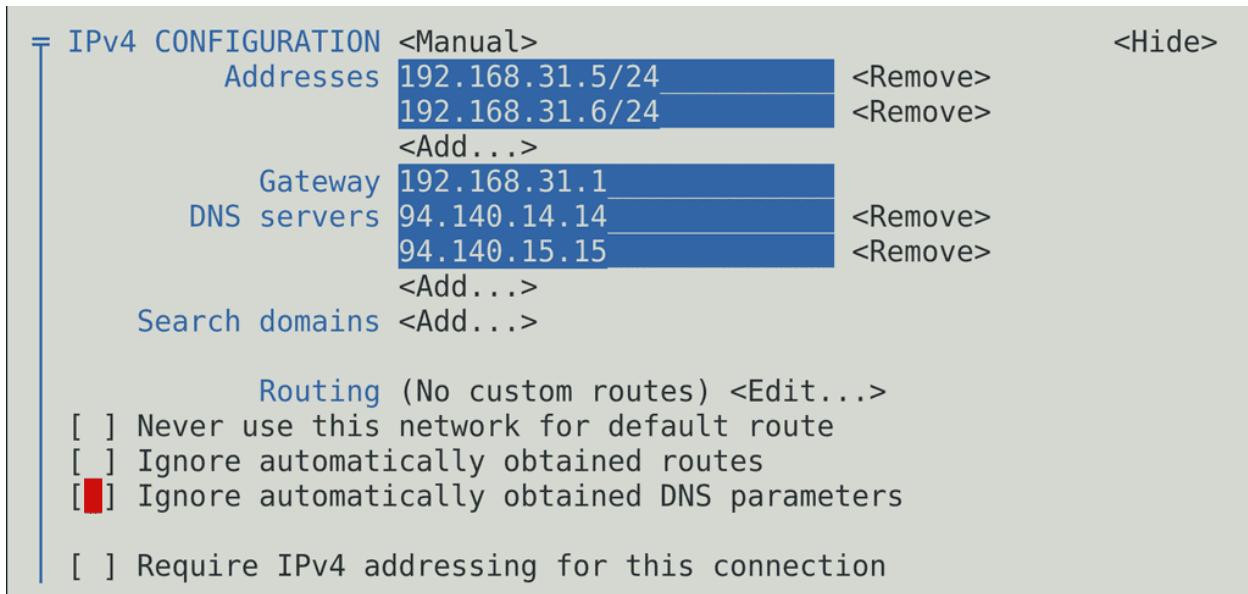
Для этого заходим в «Edit a connection» и выбираем нужный интерфейс, либо нажимаем направо и Edit. После этого открывается окно настроек интерфейса.



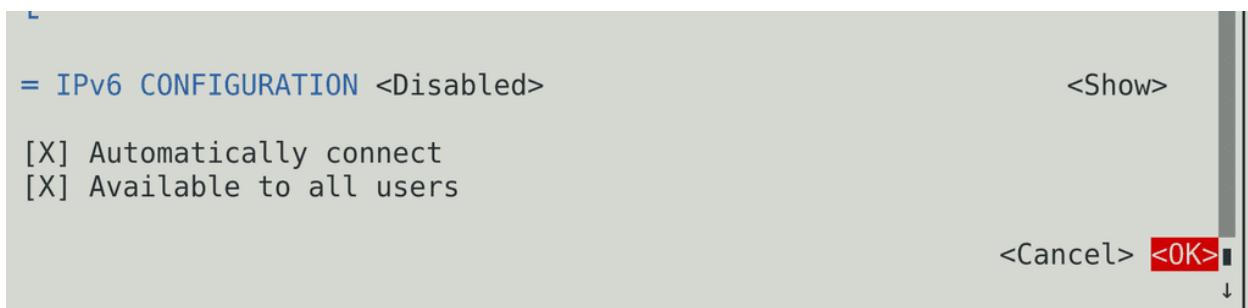
Сверху у нас Profile name - имя профиля. Это имя, которое нужно нам для понимания, поэтому можем поменять его на всё что угодно, допустим, VBox Bridge. Ниже - имя интерфейса, обычно его трогать не стоит.



Наводимся на ipv4 configuration и меняем automatic на manual. Automatic используется для получения IP по DHCP, manual - для статичного IP. Нажимаем справа на Show, чтобы показать настройки.

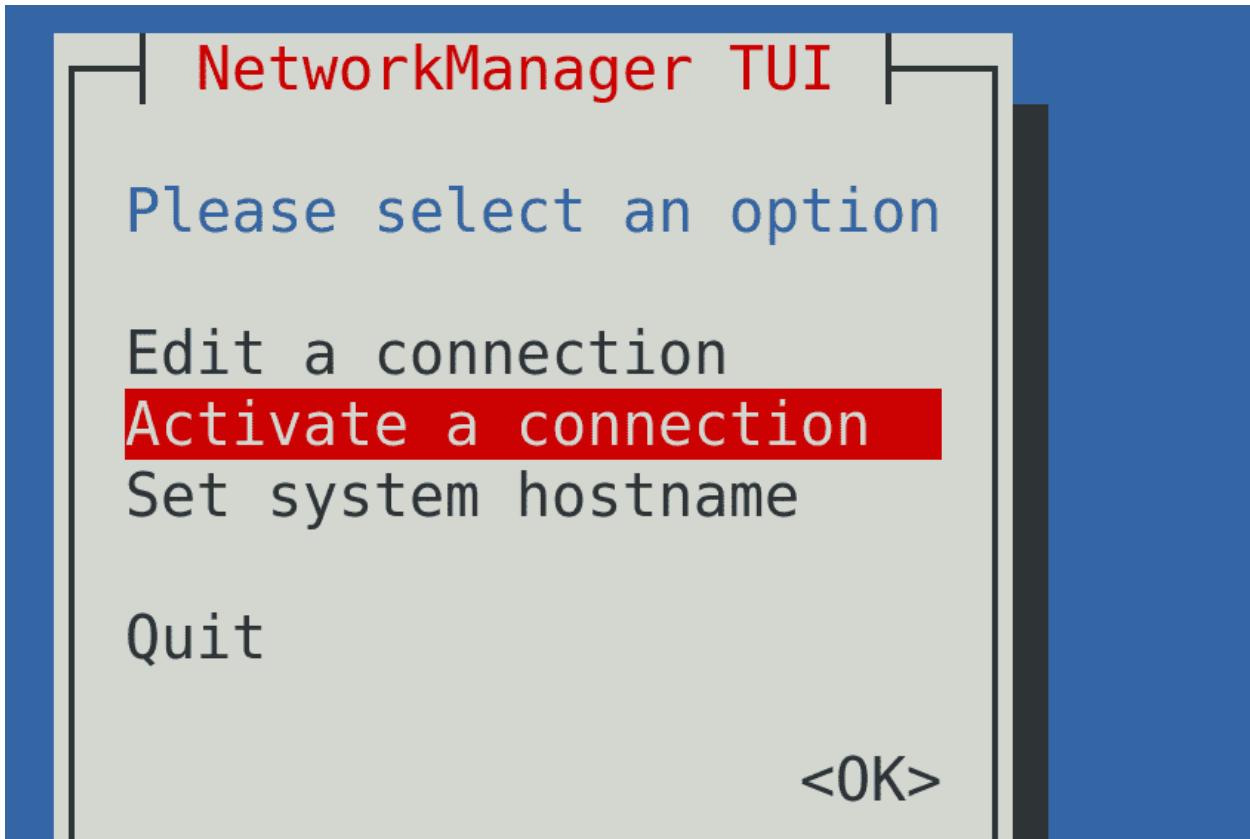


Для начала пропишем не занятый IP адрес из той же сети - 192.168.31.5/24. Технически, на одном интерфейсе можно прописать несколько IP адресов, тогда компьютер будет доступен по этим адресам. Это используется не так часто, но для примера добавим - 192.168.31.6/24. В качестве Gateway пропишем роутер - 192.168.31.1 - иначе у нас не будет работать интернет. Также добавим два DNS сервера - 94.140.14.14, 94.140.15.15. Если один будет недоступен, мы обратимся ко второму. Остальные настройки пока трогать не будем.

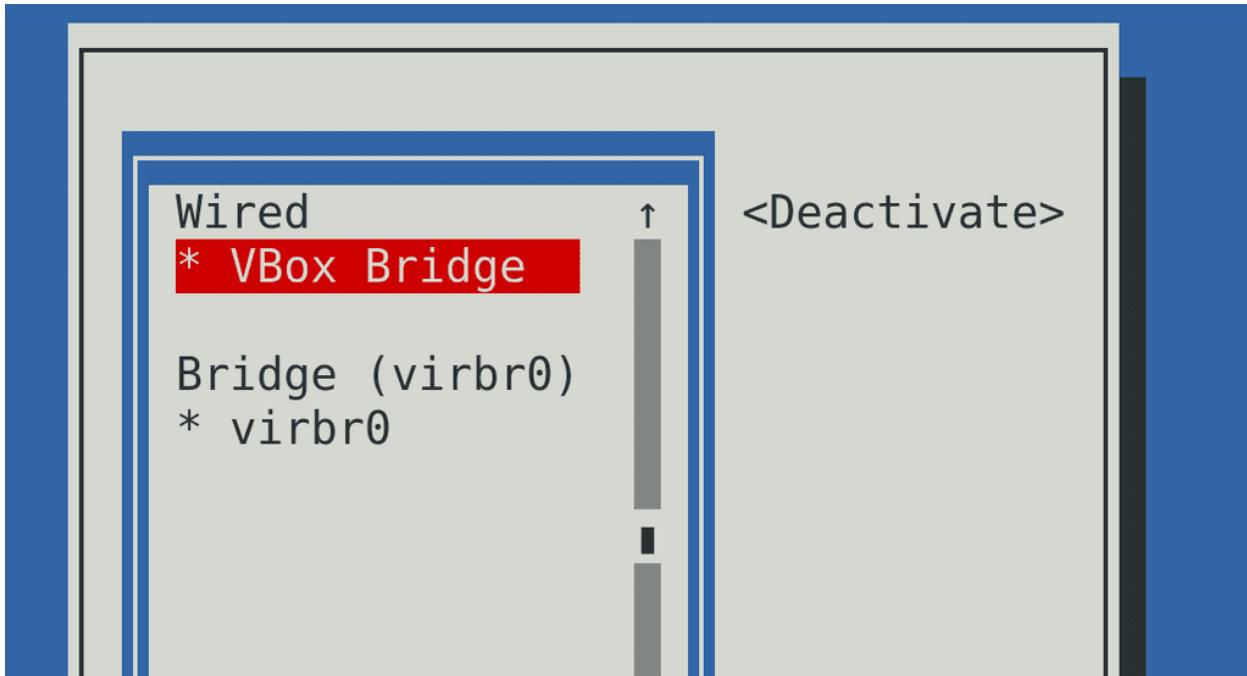


Спустимся ниже, где у нас настройка IPv6. Нам ipv6 не нужен, поэтому меняем значение на Disabled.

- «Automatically connect» говорит о том, что этот интерфейс будет подниматься при каждом включении компьютера. Если галочка не будет стоять, после перезагрузки сеть работать не будет, придётся заходить в nmcli и активировать сеть.
- «Available to all users» нужен, чтобы дать всем пользователям право менять настройки сети, отключать и включать интерфейс без root прав.



Нажмём , потом Back или Escape и зайдём в Activate a connection. Наши изменения не вступят в силу, пока мы не перезапустим интерфейс.



Потом два раза нажимаем на нужный интерфейс, чтобы деактивировать и опять активировать интерфейс. Это тот же способ административно выключить и включить интерфейс, как мы это делали с:

```
ip link set down
ip link set up
```

После переактивации можно два раза нажать Esc, чтобы выйти из nmtui.

```
[user@centos8 ~]$ ip addr show enp0s3
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 100
    link/ether 08:00:27:ce:55:bb brd ff:ff:ff:ff:ff:ff
        inet 192.168.31.5/24 brd 192.168.31.255 scope global noprefixroute enp0s3
            valid_lft forever preferred_lft forever
        inet 192.168.31.6/24 brd 192.168.31.255 scope global secondary noprefixroute enp0s3
            valid_lft forever preferred_lft forever
[user@centos8 ~]$ ip ro sh
default via 192.168.31.1 dev enp0s3 proto static metric 100
192.168.31.0/24 dev enp0s3 proto kernel scope link src 192.168.31.5 metric 100
192.168.31.0/24 dev enp0s3 proto kernel scope link src 192.168.31.6 metric 100
192.168.122.0/24 dev virbr0 proto kernel scope link src 192.168.122.1 metric 425 linkdown
[user@centos8 ~]$ cat /etc/resolv.conf
# Generated by NetworkManager
nameserver 94.140.14.14
nameserver 94.140.15.15
```

После чего проверим настройки сети:

```
ip address show enp0s3
ip ro sh
cat /etc/resolv.conf
```

Здесь видно оба IP адреса, которые мы прописали, gateway и dns сервера.

Подведём итоги. Мы с вами разобрали, как базово работать с сетью на линуксах - как посмотреть информацию об интерфейсах, маршрутах и DNS серверах, разобрали базовые утилиты для диагностики проблем, а также настройку сети.

2.43.2 Практика

Вопросы

1. Что такое «сетевой мост»?
2. Что выведет команда `ip a/ ip address`?
3. Для чего нужен loopback(lo) интерфейс?
4. Для чего используется команда ping? Опишите принцип работы
5. С компьютера 192.168.0.150 мы делаем пинг на компьютер 192.168.1.100 что произойдет с ttl пакета?
6. Как отследить путь пакета?
7. Где можно посмотреть и изменить информацию о том, какой DNS сервер использовать? Можно ли использовать несколько DNS?
8. Что должна вывести команда `nslookup google.com`?

Задания

1. Попробуйте Включить и выключить интерфейс с помощью команды `ip`
2. Пропингуйте сайт gnulinux.pro Сделайте так, чтобы после 4 запросов ping прекратился
3. Настройте получение IP адреса через DHCP с помощью `nmcli`
4. Добавьте сетевой адаптер (Host-only) через гипервизор. Настройте статический ip адрес в виртуальной машине на этом интерфейсе, в качестве dns и gateway укажите адрес хоста.

2.44 44. Удалённый доступ - SSH

2.44.1 44. Удалённый доступ - SSH

Мы с вами долгое время работаем через консоль виртуальной машины. В ней отображается графический интерфейс, но не то чтобы он сильно нужен - практически всё мы делаем через командную строку. Да, половину задач мы могли бы делать через графику, но вспомните скрипты и планировщики. Графический интерфейс программы автоматизировать очень сложно, а вот с текстовым можно сделать всё что угодно. Тогда нужен ли нам вообще графический интерфейс? Для рабочих задач можно обойтись и без него. Большинство серверов и виртуальных машин работает без графического интерфейса, чтобы не тратить ресурсы на оболочку и кучу программ, которые приходят вместе с ней. От этого и другой плюс, меньше программ - меньше уязвимостей.

Без графического интерфейса консоль виртуальной машины будет показывать виртуальный терминал. С ним зачастую неудобно работать - нельзя выделить и копировать текст, невозможно автоматизировать и всё такое. Однако консоль всё же иногда используют - при установке операционной системы, при начальной настройке сети и каких-то проблемах, когда сеть недоступна.

Но вот наличие сети позволяет администраторам через свой графический интерфейс подключаться к текстовому интерфейсу Linux-а и работать с командами ровно также, как мы это делали через эмулятор терминала. Это называется удалённым доступом. Чтобы это работало, используется сетевой протокол под названием SSH - secure shell - безопасная оболочка. SSH состоит из клиента и сервера.

```
[user@centos8 ~]$ systemctl status sshd
● sshd.service - OpenSSH server daemon
  Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
  Active: active (running) since Tue 2021-04-20 13:00:27 +04; 3min 8s ago
    Docs: man:sshd(8)
          man:sshd_config(5)
  Main PID: 1279 (sshd)
     Tasks: 1 (limit: 11424)
    Memory: 1.2M
   CGroup: /system.slice/sshd.service
           └─1279 /usr/sbin/sshd -D -oCiphers=aes256-gcm@openssh.com,chacha20-poly1305@openssh.com,
Apr 20 13:00:27 centos8 systemd[1]: Starting OpenSSH server daemon...
Apr 20 13:00:27 centos8 sshd[1279]: Server listening on 0.0.0.0 port 22.
Apr 20 13:00:27 centos8 sshd[1279]: Server listening on :: port 22.
Apr 20 13:00:27 centos8 systemd[1]: Started OpenSSH server daemon.
```

На сервере, куда вы подключаетесь, должен работать демон sshd:

```
systemctl status sshd
```

По-умолчанию, он использует 22 порт и работает с TCP. На Centos он устанавливается и включается ещё при установке системы, а на некоторых дистрибутивах его нужно устанавливать вручную.

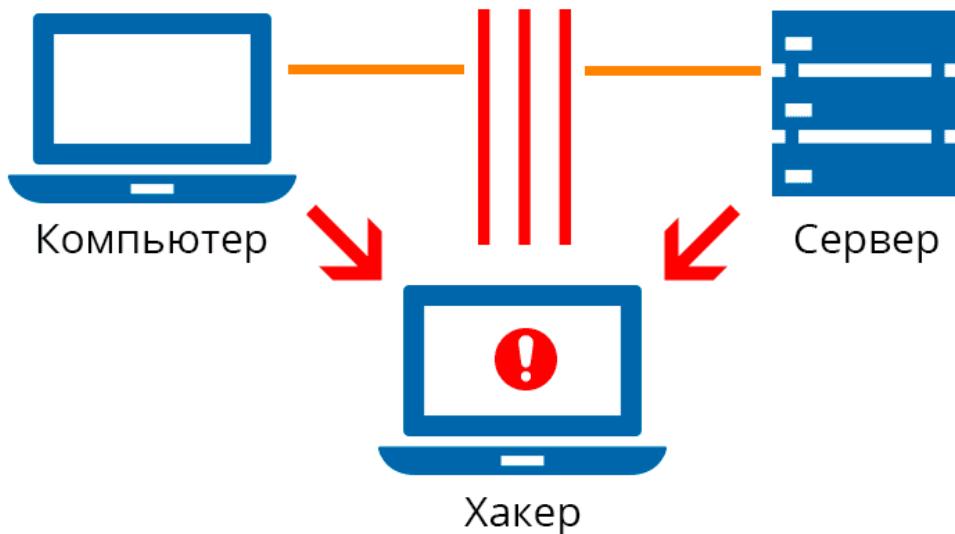
```
[user@centos8 ~]$ ssh root@127.0.0.1
The authenticity of host '127.0.0.1 (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:QfM9+Fr9KM9f3nlySBEq3TV3221e9KsdTXBo0QQaA0E.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '127.0.0.1' (ECDSA) to the list of known hosts.
root@127.0.0.1's password:
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Tue Apr 20 13:06:26 2021 from 192.168.31.5
[root@centos8 ~]# █
```

Для удалённого подключения есть множество программ. Начнём с самой простой - утилиты ssh, она есть практически везде. Попробуем подключиться сами к себе. Для этого нужно знать IP адрес. Но мы помним, что каждый компьютер может обращаться к себе с помощью адреса 127.0.0.1 - этот адрес и используем. Также нужно указать пользователя, к которому мы подключаемся, например, к root-у:

```
ssh root@127.0.0.1
```

Собачка разделяет имя пользователя и IP адрес. При первом подключении вы увидите предупреждение, что ssh клиент не смог распознать сервер и спрашивает нас, стоит ли подключаться к нему?



Дело в том, что сеть зачастую работает через десятки роутеров и свитчей. Особенно, если это в интернете, наше соединение проходит через оборудование незнакомых компаний. И если вы не соединены к серверу напрямую, а подключаетесь удалённо, есть шанс, что какой-то взломщик встанет посереди вашего трафика и сможет его видеть и даже вмешиваться. Такая атака называется *man in the middle* - человек посередине. Технически это возможно, поэтому многие сетевые протоколы, включая SSH, имеют встроенные механизмы защиты.

Например, у ssh сервера есть специальные ключи, что-то вроде удостоверения личности. И у каждого ключа есть отпечаток, что-то вроде серийного номера. Когда мы первый раз подключаемся к серверу, ssh клиент запоминает отпечаток ключа. И если завтра кто-то встанет посереди трафика и подставит свой ключ, то ssh клиент выдаст предупреждение, что этот ключ не соответствует тому, что мы сохранили. Злоумышленник, конечно, может обмануть нас, показав нам этот же ключ, но ключ, который мы видим - лишь один из двух. Мы видим лишь ключ, называемый публичным. А он не имеет смысла без второго, который хранится только на сервере и называется приватным. Но в целом, это большая тема и давайте её пока оставим. Если вам интересно, можете посмотреть по [ссылке](#). Кроме распознавания нужного сервера, ssh ещё шифрует соединение между клиентом и сервером, благодаря чему злоумышленник не увидит команды, которые вы вводите, и в целом трафик, который проходит через SSH.

```
[user@centos8 ~]$ ssh root@127.0.0.1
The authenticity of host '127.0.0.1 (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:QfM9+Fr9KM9f3nlySBEq3TV3221e9KsdTXBo0QQaA0E.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '127.0.0.1' (ECDSA) to the list of known hosts.
root@127.0.0.1's password:
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Tue Apr 20 13:06:26 2021 from 192.168.31.5
[root@centos8 ~]#
```

Поэтому вводим yes, после чего ssh спрашивает пароль пользователя, к которому мы подключаемся. Вводим пароль и попадаем в оболочку этого пользователя - root-а. А дальше всё как обычно, как мы всегда и работали, это полноценный bash со всеми командами.

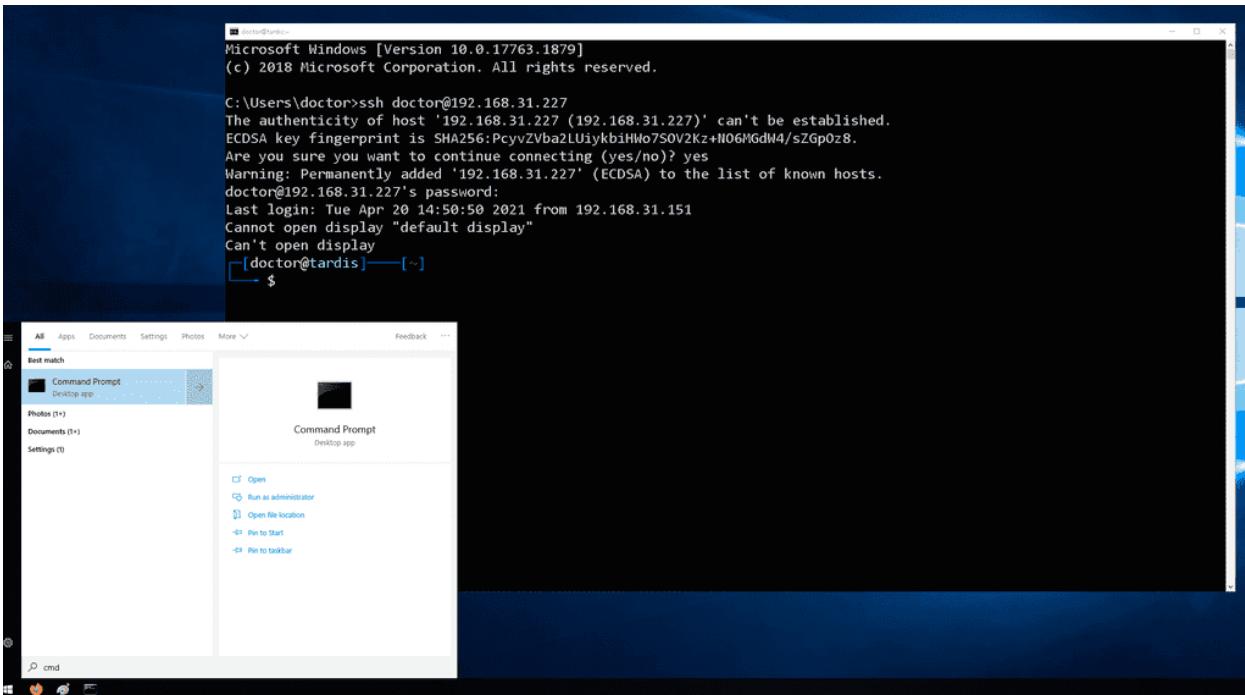
```
[doctor@tardis]—[~]
└─$ ssh user@192.168.31.5
The authenticity of host '192.168.31.5 (192.168.31.5)' can't be established.
ED25519 key fingerprint is SHA256:S0+wghxLE7vh0SGL2tH+ojf0vZDv/jnAhXLYEG/HvRM.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.31.5' (ED25519) to the list of known hosts.
user@192.168.31.5's password:
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Tue Apr 20 13:00:50 2021
[user@centos8 ~]$ ip a show enp0s3
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:ce:55:bb brd ff:ff:ff:ff:ff:ff
        inet 192.168.31.5/24 brd 192.168.31.255 scope global noprefixroute enp0s3
            valid_lft forever preferred_lft forever
        inet 192.168.31.6/24 brd 192.168.31.255 scope global secondary noprefixroute enp0s3
            valid_lft forever preferred_lft forever
[user@centos8 ~]$ logout
Connection to 192.168.31.5 closed.
[doctor@tardis]—[~]
└─$
```

Для наглядности, давайте я подключусь со своего компьютера на виртуалку. В прошлый раз мы дали ей IP - 192.168.31.5:

```
ssh user@192.168.31.5
ip a show enp0s3
```

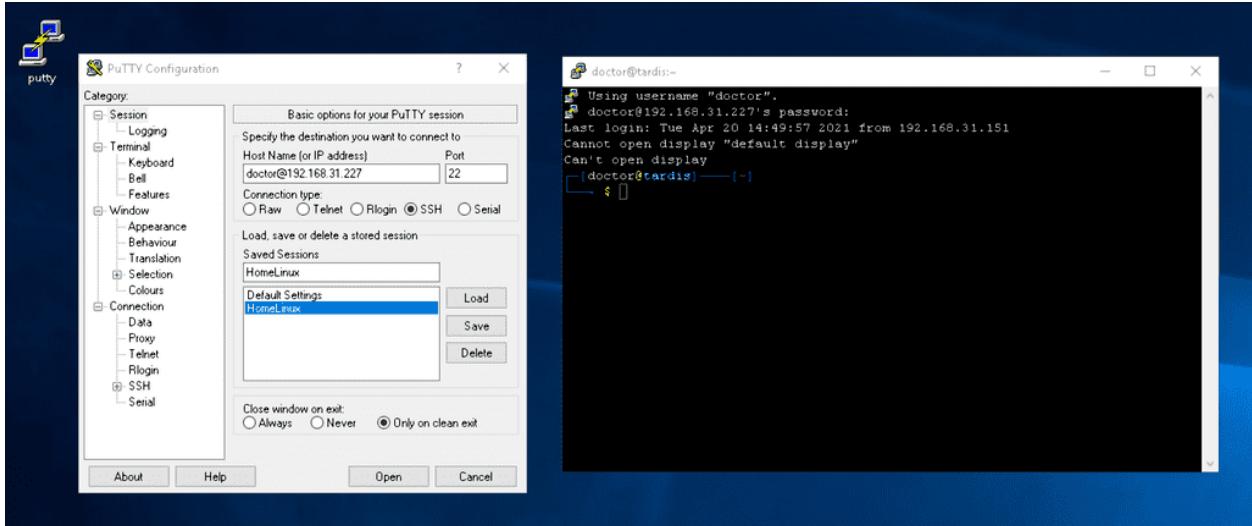
Как видите, соединение прошло успешно и теперь по началу строки, где указан пользователь и имя компьютера, можно различать, где мой компьютер, а где виртуалка. Чтобы выйти из ssh сессии, достаточно написать exit, logout или нажать Ctrl+D.



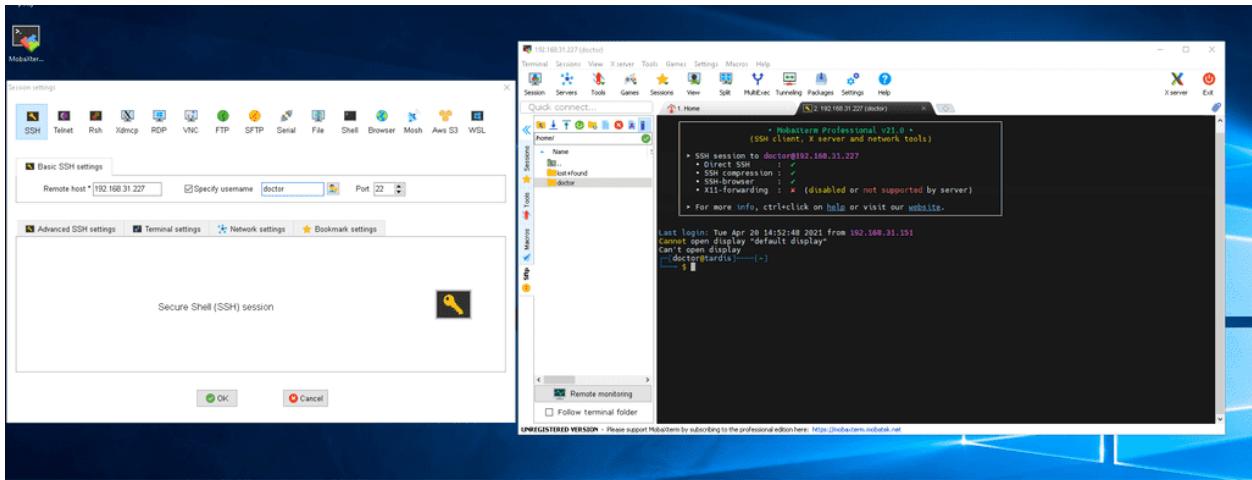
Если у вас основной системой является Windows, то тут несколько вариантов. Пару лет назад в Windows 10 добавили ssh клиент. Работает он почти также, как и на Linux. Предварительно следует запустить программу cmd.exe, которая является текстовым интерфейсом для Windows. И в нём можно запустить ssh:

```
ssh user@192.168.31.227
```

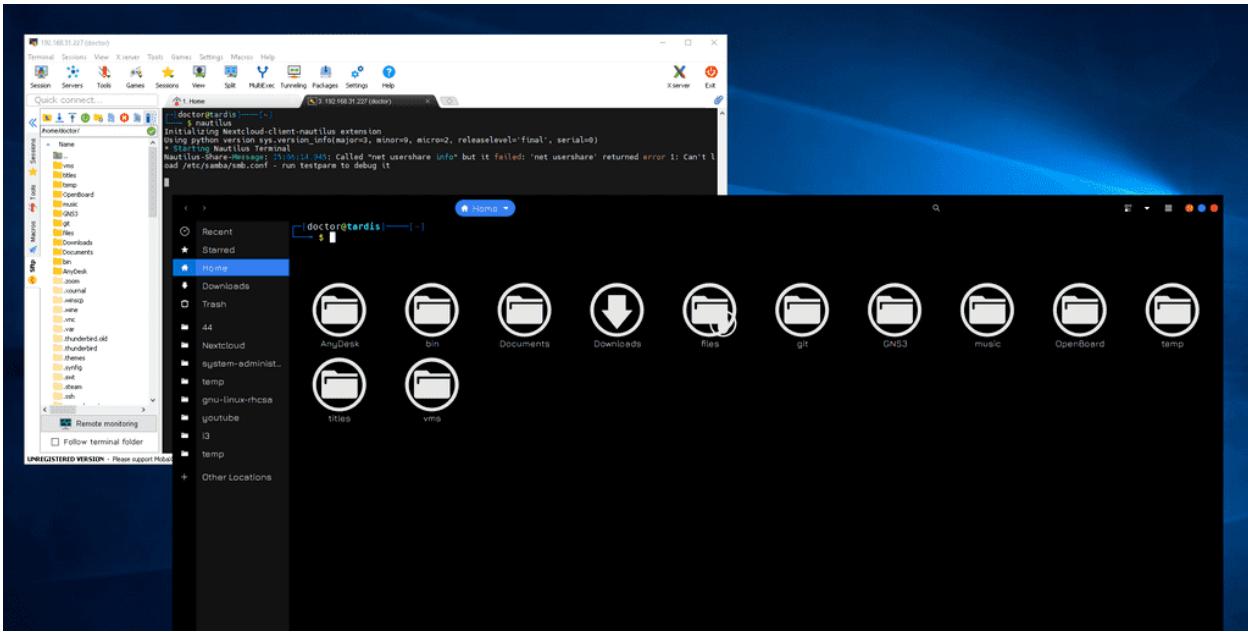
Такой вариант подойдёт для небольших задач - функционала немного, но ничего не нужно доустанавливать.



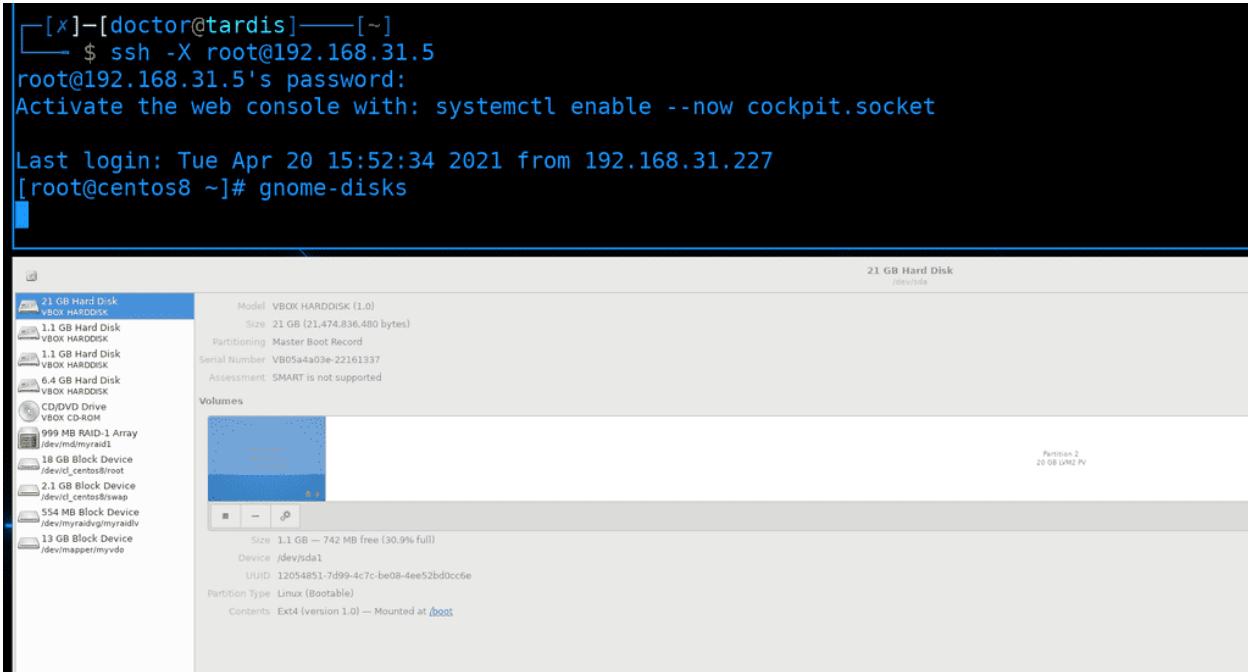
Одна из самых популярных программ - PuTTY. Функционала здесь побольше и её в основном используют администраторы, которым иногда нужно поработать с командной строкой. А также этой программой нередко пользуются для прямого подключения к сетевому оборудованию, например, к свитчам, через специальный порт. Программа весит очень мало, её легко скачать и в целом подойдёт для нечастых задач.



А администраторы, которые постоянно подключаются к большому количеству серверов, которым нужно много различных протоколов и более глубокий функционал, используют продвинутые инструменты, например, SecureCRT или MobaXterm.



Например, ssh позволяет передавать файлы между компьютерами, что удобно реализовано в MobaXterm. Также графическая подсистема в Linux-ах позволяет запускать приложения удалённо, через ssh. Например, я могу запустить программу с графическим интерфейсом на виртуалке без графики, при этом окно будет отображаться на стороне клиента ssh, т.е. на Windows, а сама программа работать на сервере. Это называется проброс графики. Чтобы это работало на Windows через PuTTY, нужно установить дополнительную программу, а в MobaXterm это уже встроено.



Для проброса графики на стороне сервера не нужна графика, достаточно пакета xauth. Если у вас основная система GNU/Linux, то никаких дополнительных программ не надо, достаточно при подключении использовать ключ -X:

```
ssh -X root@192.168.31.5
gnome-disks
```

Обычно, Linux серверов много - в компаниях могут быть десятки, сотни и тысячи Linux машин. И администраторы через свой комп или с помощью скриптов и программ управляют всеми этими компьютерами. Но пароли для такого не подходят - вам либо нужно будет на всех компьютерах держать один и тот же пароль, что очень плохая идея, либо для каждого компьютера отдельный пароль - что хорошо, но сложно. Да, вам нужно хранить пароли в парольных менеджерах, но многим программам и скриптам неудобно работать с паролями. Поэтому есть другой способ аутентификации - с помощью ключей.

Я до этого говорил о ключах в SSH, но то были ключи хоста - они нужны для безопасности, чтобы распознать нужный сервер. У SSH также есть понятие пользовательских ключей, которые можно использовать вместо пароля. Работает это так: у администратора есть два ключа - приватный и публичный. Админ закидывает публичный ключ на сервера, в домашнюю директорию пользователей, к которым будет подключаться. И при следующем подключении демон ssh увидит наличие подходящего ключа у пользователя и аутентифицирует его. Попробуем это реализовать.

```
[user@centos8 ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user/.ssh/id_rsa.
Your public key has been saved in /home/user/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:FMN+mEWbpMNkx2dmyTvGw+KKVLeq3F1P0paBqc/eeQ user@centos8
The key's randomart image is:
+---[RSA 3072]---+
|      .=o+. .
|      +.*ooB
|      .=+o* .
|      ..+...B
|      o .S+oo+*
|      . o +=.oo..
|      ...o.++
|      o.=o E
|      ....oo
+---[SHA256]---+
[user@centos8 ~]$ █
```

Для начала нужно сгенерировать ключи, это делается на стороне клиента, т.е. у администратора. Для генерации ключей используется команда ssh-keygen. По умолчанию ключи и в целом настройки клиента ssh хранятся в домашней директории пользователя, в скрытой директории `.ssh`. А ключи, по умолчанию, называются `id_rsa`. По хорошему, когда у нас много серверов, мы можем создать различные ключи и для подключения к разным серверам использовать разные ключи. Тут нажмём enter, чтобы использовать название по умолчанию. Дальше у нас спрашивается пароль для ключа. Так как ssh ключи позволяют заходить на сервера без пароля, они могут быть очень опасны, если попадут в руки злоумышленникам. Поэтому мы можем защитить приватные ключи паролем. При использовании ключа нам нужно будет вводить пароль от ключа, но, если у нас украдут этот ключ, без пароля хакеры ничего не сделают. Для нашего примера мы не будем использовать пароль, чтобы было нагляднее. Поэтому нажимаем два раза Enter и ключ создаётся. Мы видим, что создались два

файла - `id_rsa` и `id_rsa.pub` - приватный и публичный ключ соответственно. Также внизу есть эдакая картинка, визуализирующая ключ. На самом деле это больше нужно для ключей хоста - вы можете настроить так, чтобы при подключении к серверам вы видели ключ хоста в таком виде. Если вдруг ключ хоста изменится или кто-то встанет посреди трафика, ключ будет другой и визуально вы сразу заметите разницу.

```
[user@centos8 ~]$ ssh-copy-id root@127.0.0.1
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that
are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is
to install the new keys
root@127.0.0.1's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'root@127.0.0.1'"
and check to make sure that only the key(s) you wanted were added.

[user@centos8 ~]$ ssh root@127.0.0.1
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Tue Apr 20 15:55:59 2021 from 192.168.31.227
[root@centos8 ~]# █
```

У нас есть ключи, теперь мы можем закинуть наш публичный ключ на какой-нибудь сервер и проверить. Используем эту же виртуалку и пользователя root. Самый простой способ закинуть ключ - команда `ssh-copy-id`:

```
ssh-copy-id root@127.0.0.1
```

Если не использовать никакие опции, она найдёт дефолтный ключ `id_rsa.pub` и закинет его. При этом, чтобы закинуть ключ, ssh подключается на этот сервер к пользователю, поэтому он попросит пароль пользователя. После чего добавит ключ и предложит нам проверить:

```
ssh root@127.0.0.1
```

На этот раз никакого пароля не потребовалось, мы можем аутентифицироваться по ключу.

```
[user@centos8 ~]$ ls ~/.ssh/
id_rsa  id_rsa.pub  known_hosts
[user@centos8 ~]$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQgQCiDmIiPqTzKRv0k/Ky5xWe2uBdomAchBNPN8waZjheQgGyIhBqn8Kd
oNLHHQeRMYhc2XD6E2KP05efLi1PlvDrv7Kefmb9LcN/0cu7B8NiTCUf0L0XiZ1fq5KXPgwQPC3BZX6hnTYFkb4XgML
YrPkWl6g+djMPffE2rNT1hGDD5vXmwgkOFutP6dhPcvbULEPxijJyrvakrYYQN3vYUAkkgvKtZhlnCHK9FRbCSLf1bs7Y
0dbMznrmuVS4Iyhz0rGTYERHXvhnh6a/kIOA+PNBjQph92zCbI/ZREgBP8Qlqx6WRJjCbHn4MKoD9aQ4MmyNcasI3lap
Brk/6+9/0iBjZ0Qu4mWCYbh14Yx/AE7SS3Pbj8pD2Dlns7FAHPYN0PUhg0tgcUhHwp3vE+JKy4r9F4nXeq7vYu1MhKX3
HG56PSzz8gAqWdhg56aavAxat007/eD69ZdsuISn8rqN7HfNpz26GJh4clVFFAXuqXG15yS29b7DaM/e2wv/6mjD1Vc=
user@centos8
[user@centos8 ~]$ cat ~/.ssh/known_hosts
localhost ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBLrJGBWxsD
krs0adIMYPLDSWG9I0yPUedM+wehoKGbKL4G+o43u1v7LP2lXa4K0+Pj25SrbM6NPZKXC2qsBi+F4=
192.168.31.5 ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBLrJGBW
xsDkrs0adIMYPLDSWG9I0yPUedM+wehoKGbKL4G+o43u1v7LP2lXa4K0+Pj25SrbM6NPZKXC2qsBi+F4=
127.0.0.1 ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBLrJGBWxsD
krs0adIMYPLDSWG9I0yPUedM+wehoKGbKL4G+o43u1v7LP2lXa4K0+Pj25SrbM6NPZKXC2qsBi+F4=
[user@centos8 ~]$ █
```

Посмотрим появившиеся файлы, для начала у клиента. В директории `.ssh` у нас 3 файла - два ключа и файл `known_hosts`:

```
ls ~/.ssh
cat ~/.ssh/id_rsa.pub
cat ~/.ssh/known_hosts
```

Ключи представляют из себя текстовые файлы с непонятным набором символов. А в файле known_hosts указаны адреса, к которым мы подключались, а также отпечатки их ключей. Именно по этому файлу наш клиент будет понимать, изменились ли у сервера ключи или нет.

```
[user@centos8 ~]$ ssh root@127.0.0.1
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Tue Apr 20 20:28:40 2021 from 127.0.0.1
[root@centos8 ~]# ls ~/.ssh/
authorized_keys
[root@centos8 ~]# cat ~/.ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQgQCiDmIiPqTzKRv0k/Ky5xWe2uBdomAchBNPN8waZjheQgGyIhbqn8Kd
oNLHHQeRMYhc2XD6E2KP05efLi1PlvDr7KeFmb9LcN/0cu7B8NiTCUf0L0XiZ1fq5KXPgwQPC3BZX6hnTYFkb4XgML
YrPkWl6g+djMPffE2rNT1hGDD5vXmwgkOFutP6dhPcvbULEPxqJyrvakrYYQN3vYUAkkgvKtZhLNCHK9FRbCSLf1bs7Y
0dbMznrmuVS4IyhzOrGTYERHXvhnh6a/kI0A+PNBjQph92zCbI/ZREgBP8Qlqx6WRJjCbHn4MKoD9aQ4MmyNcasI3lap
Brk/6+9/0iBjZ0Qu4mWCYbh14Yx/AE7SS3PbJ8pD2DlnS7FAHPYN0PUhg0tgcUhhWp3vE+JKy4r9F4nXeq7vYu1MhKX3
HG56PSzz8gAqWdhg56aavAxat007/eD69ZdsuISn8rqN7HfNpz266Jh4c1VFFAXuqXG15yS29b7DaM/e2wv/6mjD1Vc=
user@centos8
[root@centos8 ~]#
```

На сервере, в домашней директории пользователя, к которому я закинул ключ, также появилась директория .ssh с файлом authorized_keys:

```
ssh root@127.0.0.1
ls ~/.ssh/
cat ~/.ssh/authorized_keys
```

В этот файл будут попадать публичные ключи пользователей, т.е. это результат команды ssh-copy-id. На самом деле, хоть и не понятно в терминале, в этом файле сейчас одна строчка с одним ключом. Этот тот же id_rsa.pub. Если кто-то ещё добавит ключ, то он будет на второй строчке, т.е. один ключ - одна строчка.

GNU nano 2.9.8	/home/user/.ssh/config
1 Host me	
2 Hostname 127.0.0.1	
3 User root	
4 IdentityFile ~/.ssh/id_rsa	
5 VisualHostKey yes	
6	

На стороне клиента мы можем настроить подключения в файле ~/.ssh/config. Например, можем дать подключению какое-нибудь понятное название, алиас, чтобы подключаться по нему, а не по ip адресу - Host me. После этого следует указать опцию Hostname с IP адресом, чтобы ssh знал, кто стоит за этим алиасом. Можем указать пользователя - User - чтобы не указывать его при каждом подключении. Если у нас несколько ключей, можем указать ключ, относящийся к этому хосту, чтобы не указывать его как опцию при подключении - IdentityFile. И добавим опцию VisualHostKey yes, чтобы при каждом подключении отображался ключ хоста.

```
Host me
Hostname 127.0.0.1
User root
IdentityFile ~/.ssh/id_rsa
VisualHostKey yes
```

Сохраним и выйдем.

```
[user@centos8 ~]$ ssh me
Bad owner or permissions on /home/user/.ssh/config
[user@centos8 ~]$ ls -l ~/.ssh/
total 16
-rw-rw-r--. 1 user user 90 Apr 20 21:01 config
-rw-----. 1 user user 2602 Apr 20 20:05 id_rsa
-rw-r--r--. 1 user user 566 Apr 20 20:05 id_rsa.pub
-rw-r--r--. 1 user user 516 Apr 20 13:09 known_hosts
[user@centos8 ~]$ chmod 644 ~/.ssh/config
[user@centos8 ~]$ ssh me
Host key fingerprint is SHA256:QfM9+Fr9KM9f3nlySBEq3TV3221e9KsdTXBo0QQaA0E
+---[ECDSA 256]---+
|   o.Eoo o*.|
|   . o o +=.B|
|   . o.++ *0|
|   ...ooo.B|
|   S   .o .=+|
|   o   .=0|
|   .   ..+.+
|   =oo*|
|   o==|
+---[SHA256]---+
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Tue Apr 20 20:54:51 2021 from 127.0.0.1
```

При попытке подключения:

```
ssh me
```

мы увидим ошибку - Bad owner or permissions on /home/user/.ssh/config. Сейчас у файла config права 664, но для безопасности стоит выставить 644, чтобы никто из группы не смог изменить этот файл:

```
chmod 644 ~/.ssh/config
```

Проверим ещё раз:

```
ssh me
```

Как видите, соединение прошло, при этом мы видим ключ хоста.

NAME
 ssh_config – OpenSSH SSH client configuration files

DESCRIPTION
 ssh(1) obtains configuration data from the following sources in the following order:

1. command-line options
2. user's configuration file ([~/.ssh/config](#))
3. system-wide configuration file ([/etc/ssh/ssh_config](#))

For each parameter, the first obtained value will be used. The configuration files contain sections separated by **Host** specifications, and that section is only applied for hosts that match one of the patterns given in the specification. The matched host name is usually the one given on the command line (see the **CanonicalizeHostname** option for exceptions).

Since the first obtained value for each parameter is used, more host-specific declarations should be given near the beginning of the file, and general defaults at the end.

У ssh клиента есть множество опций:

man ssh_config

которые мы можем указывать как в командной строке при подключении, так и в конфиг файле `~/.ssh/config`. Если же мы хотим прописать общие настройки для всех пользователей, то можно использовать файл `/etc/ssh/ssh_config`.

```
[user@centos8 ~]$ ssh me 'touch /root/sshtest; ls -l /root/sshtest'
Host key fingerprint is SHA256:QfM9+Fr9KM9f3nlySBeq3TV3221e9KsdTXBo0QQaA0E
+---[ECDSA 256]---+
|     o.Eoo o*.|
|     . o o +=.B|
|     . o.++ *0|
|     ...ooo.B|
|     S   .o .=+|
|           o   .=o|
|           . .+.+
|           =oo*|
|           o==|
+---[SHA256]---+
-rw-r--r--. 1 root root 0 Apr 20 22:11 /root/sshtest
[user@centos8 ~]$
```

ssh позволяет не только запускать оболочку, но и сразу выполнять команды и выводить результат:

```
ssh me 'touch /root/sshtest; ls -l /root/sshtest'
```

Т.е. я, не заходя на сервер, создал на нём файл и посмотрел вывод. Это и позволяет вам управлять серверами с помощью различных программ и скриптов.

```
GNU nano 2.9.8                               /etc/ssh/sshd_config

# OpenSSH is to specify options with their default value where
# possible, but leave them commented. Uncommented options override the
# default value.

# If you want to change the port on a SELinux system, you have to tell
# SELinux about this change.
# semanage port -a -t ssh_port_t -p tcp #PORTNUMBER
#
#Port 22
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none
```

Хорошо, про работу и настройку клиента поговорили, теперь разберём ssh сервер. Основной конфиг файл - `/etc/ssh/sshd_config`:

```
sudo nano /etc/ssh/sshd_config
```

Здесь у нас множество опций, но пока разберём основные. Одна из главных настроек - `Port`. 22 порт - стандартный для ssh и огромное количество ботов в интернете постоянно ищут компьютеры с открытым 22 портом и пытаются подобрать пароль, чтобы взломать. Даже если у вас будет хороший пароль, теоретически, у ssh могут существовать уязвимости, с помощью которых можно получить доступ к серверу. Поэтому, если ваш Linux сервер будет доступен из интернета или из недоверенной сети, нужно менять стандартный порт на нестандартный, четырёх или пятизначный, чтобы отсеять большинство ботов. Но сразу предупрежу, что порт 2222 также имеет большую популярность, поэтому придумайте что-нибудь оригинальнее, например, 7563. Но пока я менять не буду, потому что подсистема безопасности SELinux не даст это сделать простым способом. Да, чуть выше есть подсказка, как решить проблему с SELinux, но я всё равно отложу это дело, пока мы не разберём SELinux.

- `AddressFamily` - семейство адресов - обычно речь идёт о сетях ipv4 и ipv6. Вы часто будете встречать подобную опцию, и она позволяет ограничить демон, чтобы он работал только с IPv4, только с IPv6 или с обеими сетями, как в нашем случае.
- `ListenAddress` - говорит о том, на каких IP адресах будет работать этот демон. Если к серверу подключено несколько сетей, скажем, кабель от провайдера и локальная сеть, то мы можем настроить, чтобы SSH работал только на локальном IP адресе, т.е. тут указывается адрес этого сервера. В данном случае стоят нули - это означает, что сервер будет работать на всех своих адресах. А два двоеточия - тоже самое, но для IPv6 сетей.
- `HostKey` - указаны ключи хоста. Именно по ним мы и понимаем, на нужный ли сервер мы подключаемся.
- `PermitRootLogin` - говорит о том, разрешено ли пользователю root логиниться через ssh. Сейчас мы это можем, но так как пользователь root есть на всех серверах и многие боты пытаются подключиться именно им, в целях безопасности можно запретить пользователю root логиниться, либо разрешить это делать только с помощью пользовательских ключей - `prohibit-password`.
- `PubkeyAuthentication` - разрешить логиниться по публичным ключам. В большинстве случаев

так и должно быть.

- **PasswordAuthentication** - разрешить логиниться по паролям. Для безопасности, чтобы никто не мог подобрать пароль, мы можем отключить эту опцию. Но, предварительно, стоит закинуть на сервер пользовательские ключи.
- **X11Forwarding** - именно за счёт этой опции мы можем пробрасывать графику, чтобы запускать графические приложения через тот же MobaXTerm. В целом, проброс графики негативно сказывается на безопасности клиента, так как у сервера появляется доступ к вашему компьютеру. К примеру, если на вашем сервере какая-то заражённая программа, которую вы запускаете через X11Forwarding, то пока она работает, она может следить за буфером обмена, вводом клавиатуры и т.п., даже если вы вводите данные в не в самом проброшенном приложении, а где-то в другой программе. Поэтому, в целом, не стоит злоупотреблять пробросом графики, особенно на недоверенных серверах.

```
# Example of overriding settings on a per-user basis
#Match User anoncvs
#       X11Forwarding no
#       AllowTcpForwarding no
#       PermitTTY no
#       ForceCommand cvs server
```

В самом низу конфига есть пример, как можно настроить определённые параметры для определённых пользователей, скажем, кому-то разрешить проброс графики, а кому-то нет. Или при подключении к ssh сразу выполнять какую-то программу, а не запускать bash.

```
[user@centos8 ~]$ sudo systemctl restart sshd
[user@centos8 ~]$ █
```

После проделанных изменений следует перезапустить сервис:

```
sudo systemctl restart sshd
```

чтобы демон перечитал настройки.

Подведём итоги. Мы с вами научились работать с SSH - это и протокол, который позволяет удалённо подключаться к Linux серверам, и утилита, с помощью которой это реализуется. Но кроме утилиты ssh есть и другие SSH клиенты, тот же PuTTY и MobaXTerm. Мы поговорили про настройки клиента, проброс графики, пользовательские ключи и запуск команд через ssh. Также немного рассмотрели настройки демона - порт, адрес и аутентификацию. Хотя и на стороне клиента, и на стороне сервера можно настроить огромное количество опций, зачастую SSH готов к работе из коробки, разве что стоит поменять порт и использовать ключи, а также не использовать простые пароли. На пока этих знаний достаточно, но к настройке SSH мы ещё не раз вернёмся.

2.44.2 Практика

Вопросы

1. Что такое ssh и для чего он нужен?
2. На каком порту работает ssh?

3. В чем отличие публичного и приватного ключа?
4. Можно ли передавать свой открытый ключ публично?
5. Позволяет ли ssh выполнять команды или копировать файлы без явного подключения к серверу?

Задания

2.45 45. Принудительный контроль доступа - SELinux

2.45.1 45. Принудительный контроль доступа - SELinux

```
[doctor@tardis]—[~]
└─$ ssh vbox
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Mon Apr 26 10:17:49 2021 from 192.168.31.227
[user@centos8 ~]$ ls -l ~/.ssh/
total 20
-rw----- 1 user user 395 Apr 26 10:19 authorized_keys
-rw-r--r-- 1 user user 90 Apr 21 00:50 config
-rw----- 1 user user 2602 Apr 21 00:36 id_rsa
-rw-r--r-- 1 user user 566 Apr 21 00:36 id_rsa.pub
-rw-r--r-- 1 user user 171 Apr 21 00:30 known_hosts
[user@centos8 ~]$ █
```

В прошлый раз мы с вами разобрали SSH и затронули тему пользовательских ключей. Мы можем сгенерировать два ключа - приватный и публичный. После чего закинуть публичный ключ на все свои сервера и подключаться к ним без пароля, используя приватный ключ. Это говорит о том, что приватный ключ нужно беречь, как зеницу ока. Давайте глянем права этого файла:

```
ls -l ~/.ssh
```

По нем видно, что к файлу есть доступ только у владельца файла, остальные не могут его прочесть. Вроде бы всё нормально, никакой другой пользователь не сможет украсть этот файл.

```

-----BEGIN OPENSSH PRIVATE KEY-----
b3BlnZaC1rZXktbjEAAAABG5vbmlUAAAAEb9uZQAAAAAAAAAAABAABlwAAAAdzc2gtcn
NhAAAAAwEAAAAYEA3osc2zYdJgwq/spkzPy+c2x9aIceIM5SXaKxuI5bI2BCPeo6Hxtb
RynLIFV0MWPmQn0aX1atbiPEvbDRW8C70Bd/XQqUDkY1dzNLJMCRM56MA79/RilGsKn58H
Xx0Gw72FJBWQ/9FtVDj9jc52tBJRHCElz/W+Vhs8hadu/4LlKqT8j36bFjY8LchqdBuXt
2R8FIGNQ97NMrVcLyDajwNxw5bNRNjNx/QExPIYzqVxmdbnj2M5/6ngpm60isqdYqwDplB
31h+XI33rBp/iZ+s36jd6FkejzWYn2emPp3WWbMghqlYHhwgh6Eeph0lyZA08RwsxrSpud
YntI9456U7d3862Qn/rYgRsar/4atjcvbkGFCjnIdTeLEs8D0UsvjjyXUVfgpIIsmg
ntkWvdRaRM0cvKzW6PLBadu2tQZG5KhrVZDjSQIVvda2tgxw5Xlz0QlCJ7xWz8xB0+yU
YigPbKbaExcIf7NcYrUwmV77mo2wPZiRyqvS5DAAAFiI1030aNUNzmAAAB3NzaC1yc2
EAAAGBAN6LNHns2HSYMKv7KZMz8vnNsFwiHHiDOuL2isbi0WNgQj3q0h8bW0cpyyBVdDFj
5Kjzml9WrW4jxL2w0VvAuzgXf10KLHGNXczSyTckToejAO/f0YprRc+fB18dBs09hSQV
kP/RbV0Q/430drWyURwmHpc/1vlR7PIWnbv+C55qk/I9+mxY2PC31anQbl7dkfBSbjUpEz
TK1XC8g2o8DV80WzUTYzcf0BMtGm6cZnQZ49jf+p4KZujorKnWksA6ZQd9YflyN96wa
f4mfrN+o3ehZho81mJ9npj6d1lmzIIapWB4cIIehHqyDjcmQNPecLMa0qbg2J0yPeEul03
d/0s9kj/8hYEcjBqg/+GrY3L25BhQo3jyHU3oixlPAzLL48l1FX4KSCLJoJ7ZFr3XGkTN
Hlymc0jywWhnVnhGRuSoFq1W0dQkCL73WtrYMFuV5c9EC3Ce8Vs/MQTsvlGIhmz2ym2hM
XCH+zXGK1Fple+sqNsD82Ykqcru0uWAAAMBAEAAAGAMTh1fmp+ke65B7d+CfSqrv182A
SiFkED6J/WTp03b+CXzhLGJw1WQz+Vz1vQIlG4UWuTA3gt+6DUi0sc020utg/nDV91pb6
tk2K/itmcbNOBnx0TTRN5T8C/9MmU6HMrrcQsWce5BDxK3ychzawZweBhaRbuh8WzUsksE
QhMTPqNnq3LxZ0TtvnfSozztXeAfqjCKHoW6pNeD07ZuHm3/yxlixN9v3z5TCBuLcgVoIQ
Svd69HXfx013sCxIoYYCqjz4dmhNNFWAvU+EfsXjKqD0DjsKwtusbnhnyy7I0xFLz/32
McLrv1t1PivvInFoNSdtLrieqKs7Tuu0UFYIlec92Kw9g++dyks62E527U0qH2hrkLC
e8Py20kP3f4Xuc/g540tCzx4dbTBqIqoShCnZle36XXua/aN8Z0sFG/T5/xwEigRRynSgj
V7Qtc68MwLt4AywxHMsrNjnPtwJphY6rTY9tH6B48EnK/pliaRd4cFx6R7pPVmpxAAA
wQDGznA1Y10TMFu9FgDs1GTMgeZ8xy/BIV7CAigYIEfAcvUdfirozF2sQ0ER02cl3Jdk08
k4BxAbF3cPR76vK18yc71lsOLE/wG1IZ3v+6o80PeoA0tGzsTSQWn1wgWdo/BLCVa/ak/
MImyIfLcdp5baR51HIp4mW5afRsqNgrD8u0Aesfb2uZUnBra2bz67M0tRVATwAzIYZQii
kn7ntro8w50p4IkW8hEW8ikebS+zteWo9z/C0+kAQAqXMX/xAAAABAPTC4Yvz6tJGdnq6
RNrDTAgET1H7zf4CC9bc1rpz/feUh0LIm1Cot5svT86am5KhJe2xWk20f0+d2qCRjmpJ8J
IkZ20uVYT09ERRqJJxWkpq0XE7mdy4lKFw65KKWfsP8ULRU71q/0LEALTD+k8LAU8KS7Zh
ZNxmp/pIzemucvQxMTt2mm6n7U1LD7Vqw0VwPyr9u1GscApAqumH2HPGccPcnHL09s
jzOnAtp5yfEOHDMA02Kn3OKlg/hisVKQAAAMEA6MMQvsVj0Lxy2XoMco6AB1lppDagc5NK
DVMQTsgtKB+2hDz9n3fwmK7gU1s80ozpPEBv+tENAzIaiMcD1zr0WEVFUVs6EALSDVTM
k74vDixhwAF/9zpWkVVAa8KVT4kq30ArAnjMCspTt/AhqZAm1HKWeS8yN6H4NYZ1jB0yf
Cc65sSmEPzbkwq0xxjw6xkXLmjTs3tk6sd2QM6PFVuZQCrT+u9ApczjQUbkxFu16h7r0E
Z6aaAUx3x/0EmLAAAADHVZxJAY2VudG9z0AECAwQFBg==
-----END OPENSSH PRIVATE KEY-----

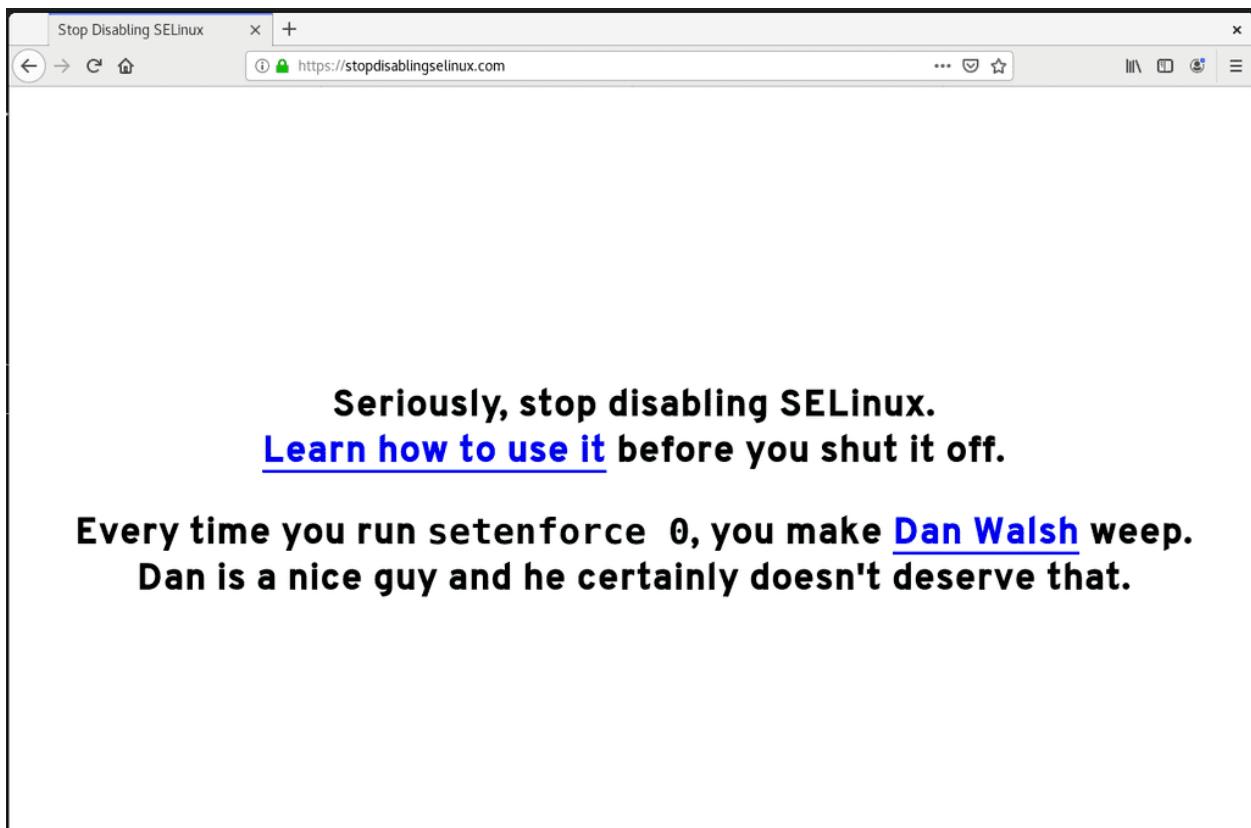
```

А давайте подумаем, что означают права? То, что у процесса, запущенного от имени этого пользователя, будет доступ к этому файлу. И если я запускаю ssh клиент от моего пользователя, то он сможет прочесть этот файл, а если кто-то другой запустит ssh - то это будет процесс от другого пользователя и у него не будет прав к этому файлу. Но что, если я запущу не ssh клиент, а другую программу? Например, firefox. И через firefox попытаюсь прочесть этот файл. С точки зрения стандартных прав всё нормально - это процесс, запущенный от моего пользователя и у него есть доступ ко всем моим файлам. И в обычной ситуации firefox-у не нужны мои ssh ключи. А если он заражён? Что, если я установил какой-то плагин для firefox-а, а он оказался зловредом? Получится, что firefox сможет делать с моими файлами всё что угодно, например, украсть мои ssh ключи. По хорошему, у firefox в принципе не должно быть прав к моим ключам. Но, с точки зрения прав, всё нормально.

А это значит, что стандартных прав недостаточно. У нас есть процессы и файлы, которые могут взаимодействовать с точки зрения прав, но нужно как-то оградить их. И для этого есть системы принудительного доступа. В самом ядре Linux есть фреймворк LSM - что-то типа каркаса, который можно использовать для разработки программ для безопасности. АНБ и RedHat на основе этого каркаса сделали SELinux, который используется многими дистрибутивами на основе RHEL, а на Ubuntu-based дистрибутивах используется AppArmor.

Очень грубо говоря, в SELinux прописаны политики, какой программе что разрешено. Если что-то не прописано - он блокирует это. Политики прописаны на стандартные настройки, а администраторы в программах зачастую меняют эти стандартные настройки. Допустим, мы говорили, что по умолчанию ssh сервер работает на 22 порту, но, если сервер доступен в интернете, стоит изменить стандартный порт. Так вот, если изменить стандартный порт в SSH, SELinux запретит ssh серверу работать на нестандартном порту и демон просто не сможет запуститься. Чтобы он заработал, надо поменять политику

в SELinux. И так со многими программами.



Это отпугивает некоторых администраторов, так как нередко они просто копипастят конфиги и команды из интернета, не вникая в то, что там написано. И тут SELinux блокирует демон из-за нестандартных настроек. Администраторы не понимают почему, не знают как решить и ищут одну простую команду - чтобы всё работало. И зачастую эта команда - отключить SELinux. Но отключать безопасность в угоду лени - плохая затея, даже есть специальный [сайт](#), который просит так не делать.

```
[user@centos8 ~]$ sestatus
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:              targeted
Current mode:                   enforcing
Mode from config file:          enforcing
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Memory protection checking:    actual (secure)
Max kernel policy version:     32
[user@centos8 ~]$ getenforce
Enforcing
[user@centos8 ~]$ █
```

И так, SELinux можно отключать, но не только. В целом у него 3 режима - disabled, permissive и enforcing. Текущий статус можно посмотреть с помощью команд:

```
sestatus
getenforce
```

По умолчанию - enforcing - политики работают, всё что не разрешено политиками - блокируется. В режиме permissive ничего не блокируется, но SELinux всё нестандартное логирует. А в режиме disabled он просто отключён.

```
[user@centos8 ~]$ cat /etc/selinux/config
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of these three values:
#       targeted - Targeted processes are protected,
#       minimum - Modification of targeted policy. Only selected processes are protected.
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted

[user@centos8 ~]$ sudo setenforce 0
[user@centos8 ~]$ getenforce
Permissive
[user@centos8 ~]$ sudo setenforce 1
[user@centos8 ~]$ getenforce
Enforcing
[user@centos8 ~]$ █
```

Режимы можно поменять в файле `/etc/selinux/config`, изменив значение переменной SELINUX. Но это изменение вступит в силу только после перезагрузки, а чтобы сейчас временно его отключить достаточно прописать setenforce 0:

```
sudo setenforce 0
getenforce
```

что переведёт SELinux в режим permissive. Чтобы заново перевести в enforcing - setenforce 1:

```
sudo setenforce 1
getenforce
```

Выключать SELinux не рекомендую, лучше научиться его настраивать. Но иногда в работе встречается софт, в требованиях которого указано - не работает с SELinux. На самом деле, в большинстве случаев можно проигнорировать это требование и настроить его должным образом. Но если в дальнейшем возникнут проблемы и техническая поддержка софта увидит включённый SELinux - начнёт все проблемы валить на него. Поэтому нужно уметь его выключать, но не стоит это делать просто так. Если всё же нельзя использовать SELinux - используйте режим permissive - логи могут быть полезны.

```
[user@centos8 ~]$ sudo nano /etc/ssh/sshd_config
[user@centos8 ~]$ sudo grep Port /etc/ssh/sshd_config
Port 2233
#GatewayPorts no
[user@centos8 ~]$ sudo systemctl restart sshd
Job for sshd.service failed because the control process exited with error code.
See "systemctl status sshd.service" and "journalctl -xe" for details.
[user@centos8 ~]$ sudo systemctl status sshd
● sshd.service - OpenSSH server daemon
    Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
    Active: activating (auto-restart) (Result: exit-code) since Mon 2021-04-26 13:25:06 +04; 5s ago
      Docs: man:sshd(8)
             man:sshd_config(5)
   Process: 7560 ExecStart=/usr/sbin/sshd -D $OPTIONS ${CRYPTO_POLICY} (code=exited, status=255)
 Main PID: 7560 (code=exited, status=255)

Apr 26 13:25:06 centos8 systemd[1]: sshd.service: Main process exited, code=exited, status=255/n/a
Apr 26 13:25:06 centos8 systemd[1]: sshd.service: Failed with result 'exit-code'.
Apr 26 13:25:06 centos8 systemd[1]: Failed to start OpenSSH server daemon.
[user@centos8 ~]$ █
```

О деталях работы SELinux поговорим чуть позже, сейчас для понимания нужен пример. Тот же самый SSH с нестандартным портом. Зайдём в конфиг:

```
sudo nano /etc/ssh/sshd_config
```

раскомментируем Port и поменяем значение на другое - Port=2233. Чтобы изменения вступили в силу, нужно перезапустить демон:

```
sudo systemctl restart sshd
```

Но после команды мы видим ошибку. Если рестарт демона заканчивается ошибкой - значит он перестаёт работать, можем проверить статус:

```
sudo systemctl status sshd
```

Как видите, статус сервиса - activating - это означает, что он пытается запуститься, но не может. Кстати, обратите внимание, что даже при перезапуске SSH сервера и его неработоспособности, я всё ещё подключён по SSH. На самом деле, это скорее исключение из правил, так как большинство других демонов при рестарте сбрасывает соединения. Но SSH так не делает, чтобы администратор не потерял доступ к серверу, так как вернуть доступ может быть довольно сложной задачей.

```
[user@centos8 ~]$ sudo journalctl -eu sshd
Apr 26 13:28:37 centos8 systemd[1]: Failed to start OpenSSH server daemon.
Apr 26 13:29:19 centos8 systemd[1]: sshd.service: Service RestartSec=42s expired, scheduling restart.
Apr 26 13:29:19 centos8 systemd[1]: sshd.service: Scheduled restart job, restart counter is at 7.
Apr 26 13:29:19 centos8 systemd[1]: Stopped OpenSSH server daemon.
Apr 26 13:29:19 centos8 systemd[1]: Starting OpenSSH server daemon...
Apr 26 13:29:19 centos8 systemd[1]: sshd.service: Main process exited, code=exited, status=255/n/a
Apr 26 13:29:19 centos8 systemd[1]: sshd.service: Failed with result 'exit-code'.
Apr 26 13:29:19 centos8 systemd[1]: Failed to start OpenSSH server daemon.
```

Первое, что нужно сделать, если сервис не запустился - посмотреть логи:

```
sudo journalctl -eu sshd
```

В большинстве случаев здесь можно найти причину, но в случае с SELinux - в логах демона об этом ни слова. Большинство демонов не в курсе о SELinux и не знают, что их блокирует.

```
Apr 26 13:43:41 centos8 sedispatch[1018]: AVC Message for setroubleshoot, dropping message
Apr 26 13:43:45 centos8 sedispatch[1018]: AVC Message for setroubleshoot, dropping message
Apr 26 13:43:45 centos8 setroubleshoot[8496]: SELinux is preventing /usr/sbin/sshd from name_bind access on the >
Apr 26 13:43:45 centos8 platform-python[8496]: SELinux is preventing /usr/sbin/sshd from name_bind access on the >

*****  Plugin bind_ports (92.2 confidence) suggests  *****>

If you want to allow /usr/sbin/sshd to bind to network port 2233
Then you need to modify the port type.
Do
# semanage port -a -t PORT_TYPE -p tcp 2233
      where PORT_TYPE is one of the following: ssh_port_t, vnc_port_t

*****  Plugin catchall_boolean (7.83 confidence) suggests  *****>

If you want to allow nis to enabled
Then you must tell SELinux about this by enabling the 'nis_enabled' boolean.

Do
setsebool -P nis_enabled 1
```

Однако сам SELinux также посыпает логи, которые можно увидеть в числе последних:

```
sudo journalctl -e
```

Они сразу выделяются среди остальных логов - здесь и причина блокировки, и подсказка, как решить эту проблему. Здесь мы видим, что SELinux блокирует приложению /usr/bin/sshd использование порта 2233. И тут же подсказка - если вы хотите разрешить, запустите команду - `semanage port -a -t тип порта -p tcp 2233`. В типах порта может быть ssh_port_t, vnc_port_t и т.д., но интуитивно понятно, что речь про ssh_port_t.

Попробуем запустить команду:

```
sudo semanage port -a -t ssh_port_t -p tcp 2233
```

После чего рестартнём сервис ssh и проверим его статус:

```
sudo systemctl restart sshd  
sudo systemctl status sshd
```

Перезапустилось без ошибок и всё работает - внизу видно, что ssh теперь работает на порту 2233.

```
# If you want to change the port on a SELinux system, you have to tell
# SELinux about this change.
# semanage port -a -t ssh_port_t -p tcp #PORTNUMBER
#
Port 2233
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress :  
[REDACTED]
```

Это было просто - мы посмотрели последние логи, где SELinux подсказал нам нужную команду, мы её запустили - и всё работает. И в самих настройках SSH, над строчкой смены порта, также указывалась эта команда. Но, справедливости ради, не всегда так просто. В конфигах каких-то часто используемых демонов над часто изменяемыми опциями бывают подсказки, в логах также указываются такие подсказки. Но где-то может не быть подсказок, где-то подсказки могут быть неточными - поэтому нужно немного разобраться в самом SELinux и его командах. Прежде чем продолжим, вернём в конфигах SSH порт 22, так как по 2233 порту мы не сможем подключиться из-за фаервола. После чего рестартнём сервис.

И так, до этого я говорил о том, что SELinux может блокировать доступ к файлу, на примере приватного ключа SSH, а потом показал пример с tcp портом. Также его можно использовать для ограничения доступа к оборудованию, но, в отличии от файлов и сетевых портов, это настраивается реже. Как это работает? Для примера возьмём блокировку доступа к файлу.



Всю работу операционной системы в очень простом виде можно представить так - какой-то пользователь запустил какую-то программу, которая стала процессом, и этот процесс обращается к каким-то файлам. Пользователь, процесс и файл - 3 основных компонента, которым SELinux ставит метки. Эти метки называются контекстами. И SELinux позволяет написать правила, по которому, например, процесс с меткой 2 может читать файл с меткой 3. Когда какой-то процесс пытается открыть файл, сначала система проверяет на стандартные права - группу, владельца, остальных, а также rwx. Если стандартные права говорят, что всё ок, что у процесса есть доступ к файлу, SELinux делает свою проверку - а есть ли у метки 2 доступ к метке 3? И если такого правила нет, то SELinux запрещает это действие и процесс не может открыть файл.

```
[user@centos8 ~]$ id -Z
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[user@centos8 ~]$ ps -Z
LABEL PID TTY TIME CMD
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 9313 pts/1 00:00:00 bash
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 10340 pts/1 00:00:00 ps
[user@centos8 ~]$ ls -Z file
unconfined_u:object_r:user_home_t:s0 file
[user@centos8 ~]$
```

В программы, которые показывают информацию о пользователях, процессах и файлах, для работы с SELinux добавили ключ `-Z`. Давайте посмотрим контекст нашего пользователя, какого-нибудь процесса и файла:

```
id -Z
ps -Z
ls -Z file
```

Вот этот страшный набор символов – это и есть контекст. Но пусть вас это не пугает, потому что всё довольно просто.

```
[user@centos8 ~]$ id -Z
unconfined:unconfinedr:unconfinedt:s0-s0:c0.c1023
```

u	r	t
---	---	---

```
[user@centos8 ~]$ ps -Z
LABEL PID TTY TIME CMD
unconfined:unconfinedr:unconfinedt:s0-s0:c0.c1023 9313 pts/1 00:00:00 bash
unconfined:unconfinedr:unconfinedt:s0-s0:c0.c1023 10340 pts/1 00:00:00 ps
[user@centos8 ~]$ ls -Z file
unconfined:objectr:user_homet:s0

```

Обратите внимание, что контекст состоит из нескольких значений, разделённых двоеточием. И в конце каждого значения есть нижнее подчёркивание и определённая буква. `u` – это юзер, `r` – это роль, `t` – это тип.

```
[user@centos8 ~]$ sudo semanage login -l
Login Name      SELinux User      MLS/MCS Range      Service
__default__      unconfined_u      s0-s0:c0.c1023      *
root            unconfined_u      s0-s0:c0.c1023      *
[user@centos8 ~]$ id -Z
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[user@centos8 ~]$ sudo id -Z
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[user@centos8 ~]$
```

Начнём с юзера. SELinux не использует системных пользователей, а имеет своих, но при этом их связывает. Это можно увидеть с помощью команды `semanage login` с ключом `-l`:

```
sudo semanage login -l
```

В первом столбце – локальные пользователи – `__default__` подходит под всех пользователей, `root`

только под рута. Во втором столбце - к какому пользователю SELinux они привязаны - в данном случае и рут, и все остальные наши пользователи привязаны к пользователю `unconfined`. Т.е. к одному пользователю SELinux привязаны несколько локальных пользователей. Этот пользователь используется, если никаких ограничений по пользователю мы не хотим. Т.е. в нашей системе SELinux никак не ограничивает по пользовательским меткам.

Table 3.1. SELinux user capabilities

User	Role	Domain	X Window System	su or sudo	Execute in home directory and /tmp (default)	Networking
sysadm_u	sysadm_r	sysadm_t	yes	su and sudo	yes	yes
staff_u	staff_r	staff_t	yes	only sudo	yes	yes
user_u	user_r	user_t	yes	no	yes	yes
guest_u	guest_r	guest_t	no	no	yes	no
xguest_u	xguest_r	xguest_t	yes	no	yes	Firefox only

Однако, для максимальной безопасности, если у вас много пользователей в системе, можно настроить SELinux так, чтобы определённые пользователи могли делать только что-то определённое. Например, создать политику, чтобы пользователь не мог запускать sudo. Есть несколько дефолтных пользователей SELinux с прописанными правилами, но, так как в большинстве случаев это только усложнит работу администратору, по умолчанию, у нас не такой жёсткий режим. Т.е. хоть мы и сказали, что в контексте указан юзер, в нашей системе это ничего не значит, у нас нет блокировок по пользовательским меткам.

Пользователи:

Пользователь 1



Пользователь 2

Роли:

Системный админ

Может настроить только пароли

Сетевой админ

Может настроить только сеть

Но представьте, что ограничения мы будем прописывать по пользователям. Например, пользователю 2 можно настроить только сеть. А если у нас несколько пользователей, которым нужны различные разрешения? Кому-то надо разрешить сеть настраивать, кому-то sudo делать и т.п. Получится, что для каждого пользователя надо отдельные политики писать. А это очень неудобно. Поэтому правила обычно пишутся не на пользователей, а на роли. И у одного пользователя могут быть несколько ролей. Т.е., допустим, у пользователя 2 роль только «сетевой админ», и всё что он может - настроить сеть. А пользователь 1 должен и сеть настроить и пароли задать. Поэтому ему мы даём две роли.



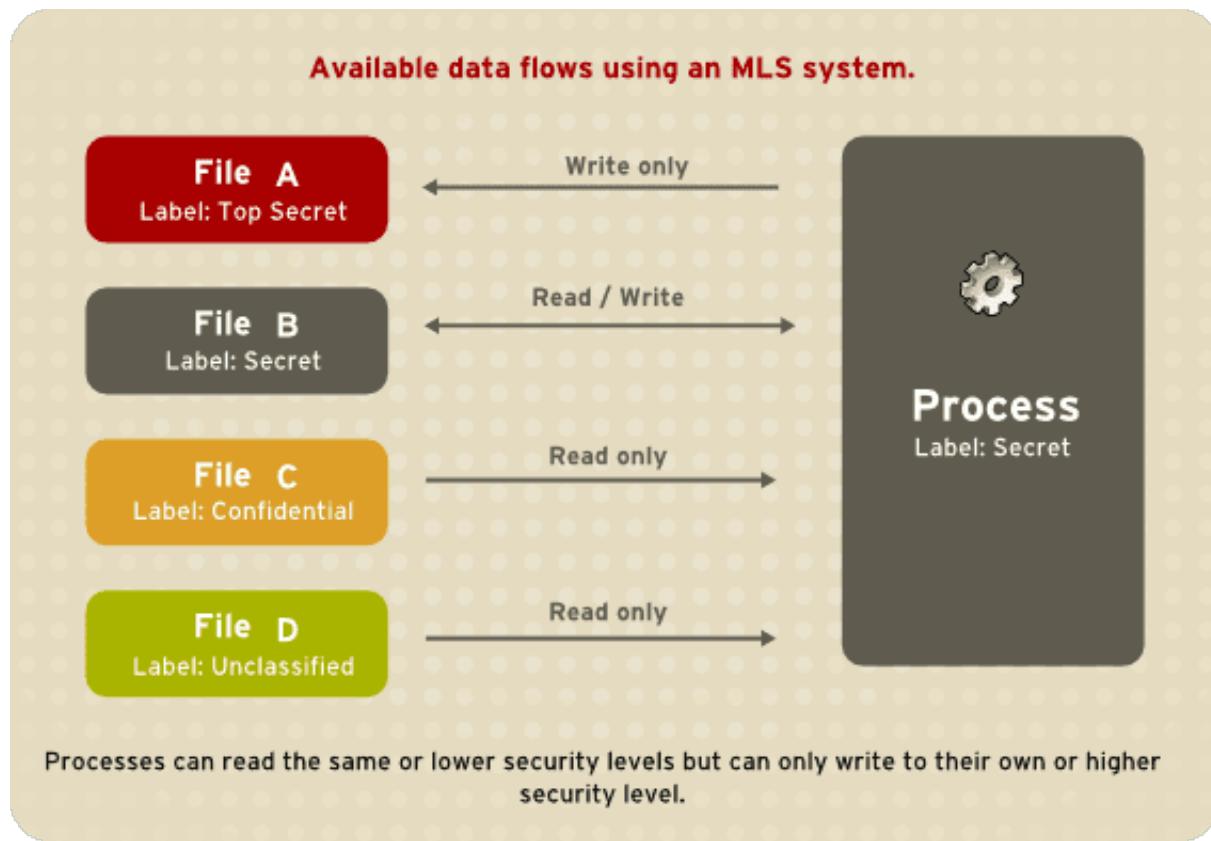
В итоге у нас получается, что несколько обычных пользователей привязываются к одному SELinux пользователю, а у него может быть несколько ролей.

SELinux User	Labeling Prefix	MLS/MCS Level	MLS/MCS Range	SELinux Roles
guest_u	user	s0	s0	guest_r
root	user	s0	s0-s0:c0.c1023	staff_r sysadm_r system_r unconfined_r
staff_u	user	s0	s0-s0:c0.c1023	staff_r sysadm_r system_r unconfined_r
sysadm_u	user	s0	s0-s0:c0.c1023	sysadm_r
system_u	user	s0	s0-s0:c0.c1023	system_r unconfined_r
unconfined_u	user	s0	s0-s0:c0.c1023	system_r unconfined_r
user_u	user	s0	s0	user_r
xguest_u	user	s0	s0	xguest_r

Чтобы посмотреть, у какого пользователя какие роли, можно выполнить команду semanage user с ключом -l:

```
sudo semanage user -l
```

Тут видно, что у некоторых пользователей одна роль, а у некоторых несколько.



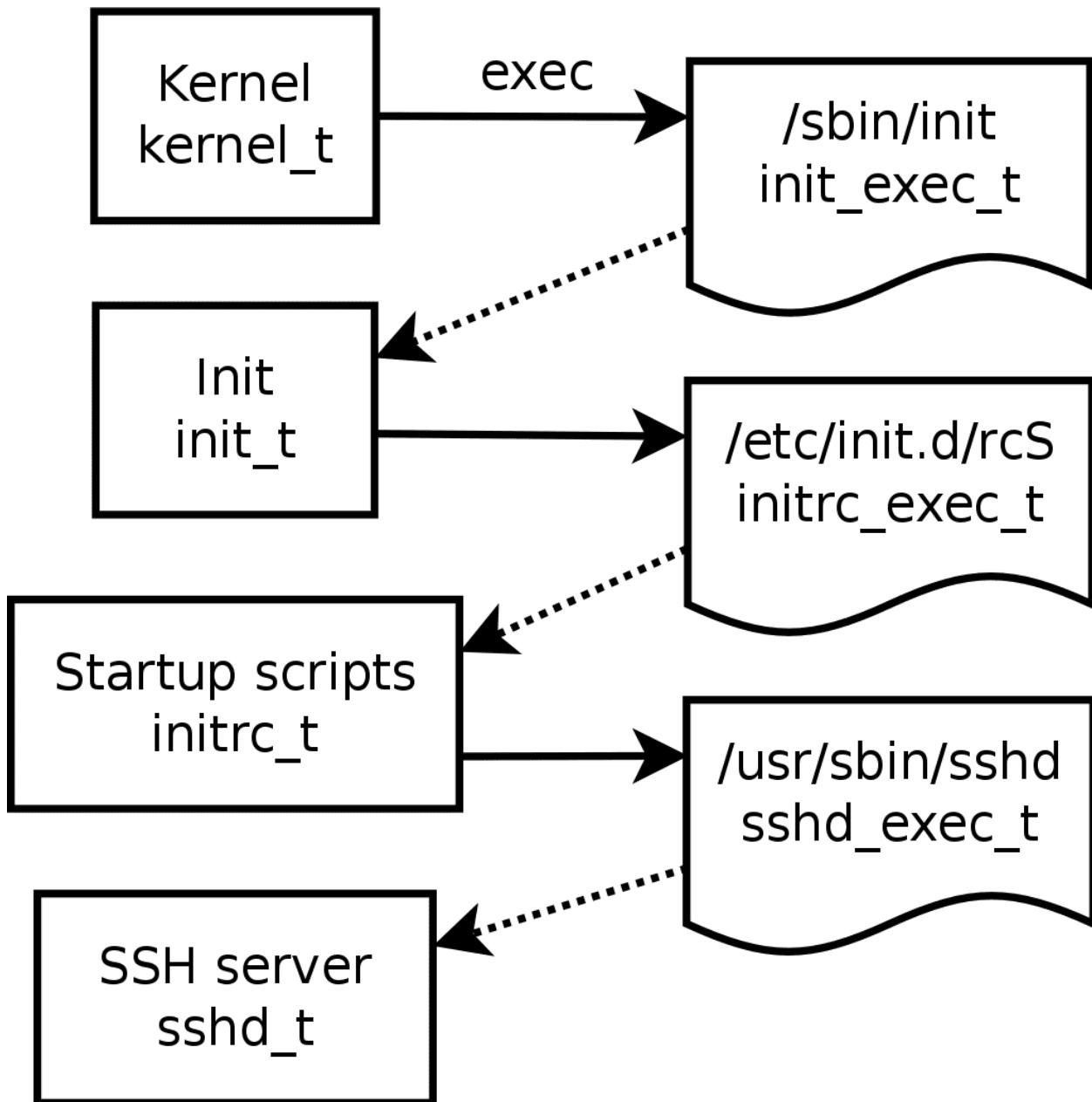
И раз уж у нас перед глазами маячит MLS, давайте его тоже упомянем. MLS - многоуровневая безопасность. Это не совсем часть принудительного контроля доступа, скорее отдельный механизм, но тесно связанный с этой темой. Вкратце, MLS позволяет настроить безопасность по уровням. Скажем, у нас есть процесс, у которого определённый уровень. Файлы с уровнями ниже он может читать, файл с тем же уровнем он может читать и изменять, а файл с уровнем выше он может только изменять, не имея доступ для чтения. Но MLS по умолчанию выключен, мы видим его метки, но сейчас никаких ограничений по ним нет.



После роли у нас идёт тип. Если юзер и роль у нас сейчас никак не воздействуют на систему, то именно тип накладывает ограничения. Помните пример в начале, где какой-то процесс не мог открыть файл, потому что метка не позволяла? Так вот, всё дело в типе. У процесса 1 есть свой тип, у файла - свой тип. SELinux проверяет правила, а может ли тип 1 открывать тип 2? Правда типы, относящиеся к процессам, принято называть доменами, а у файлов типы это типы. Т.е. может ли домен 1 открыть тип 2?

Processes and domains

Objects and types



Иногда один домен может переходить в другой, например, когда одна программа запускает другую. И это также задаётся в правилах SELinux. Т.е., если одна программа попытается запустить другую, то SELinux проверит, есть ли у программы с доменом 1 право на запуск программы с доменом 2?

И так, принцип работы SELinux-а надеюсь понятен: у каждого пользователя, процесса и файла есть контекст, состоящий из юзера, роли и типа. Юзер и роль определяют разрешения по пользователям, но по умолчанию это вырублено. А типы - если он относится к процессу, то это домен, если к файлу - то это тип - определяют, у каких процессов к каким файлам, другим процессам, портам и устройствам есть доступ, и какой именно. Есть ещё MSL, но он по умолчанию выключен и не совсем по теме.

```
[user@centos8 ~]$ sudo semanage port -a -t ssh_port_t -p tcp 2233^C
[user@centos8 ~]$ sudo semanage port -l | grep ssh_port_t
[sudo] password for user:
ssh_port_t          tcp      2233, 22
[user@centos8 ~]$ █
```

Давайте попытаемся проанализировать, что мы сделали с SSH, когда меняли порт:

```
sudo semanage port -a -t ssh_port_t -p tcp 2233
```

Сразу виден тип - `ssh_port_t`. Если посмотреть значение опции `-a` для `semanage-port` станет ясно, что мы для этого типа добавили запись. Т.е. теперь `ssh_port_t` имеет доступ не только к 22 порту, но и к 2233. Это можно увидеть, если посмотреть список всех разрешений по портам:

```
sudo semanage port -l | grep ssh_port_t
```

```
[user@centos8 ~]$ sudo semanage fcontext -l | grep sshd
/etc/rc.d/init.d/sshd          regular file    system_u:object_r:sshd_initrc_exec_t:s0
/etc/ssh/primes                 regular file    system_u:object_r:sshd_key_t:s0
/etc/ssh/ssh_host.*_key          regular file    system_u:object_r:sshd_key_t:s0
/etc/ssh/ssh_host.*_key.pub      regular file    system_u:object_r:sshd_key_t:s0
/usr/lib/systemd/system/sshd-keygen.* regular file    system_u:object_r:sshd_keygen_unit_file_t:s0
0
/usr/lib/systemd/system/sshd.*   regular file    system_u:object_r:sshd_unit_file_t:s0
/usr/libexec/openssh/sshd-keygen regular file    system_u:object_r:sshd_keygen_exec_t:s0
/usr/sbin/gsssshd                regular file    system_u:object_r:sshd_exec_t:s0
/usr/sbin/sshd                   regular file    system_u:object_r:sshd_exec_t:s0
/usr/sbin/sshd-keygen            regular file    system_u:object_r:sshd_keygen_exec_t:s0
/var/empty/sshd/etc/localtime    regular file    system_u:object_r:locale_t:s0
/var/run/sshd..init\.\pid        regular file    system_u:object_r:sshd_var_run_t:s0
/var/run/sshd\.\pid              regular file    system_u:object_r:sshd_var_run_t:s0
[user@centos8 ~]$ ls -Z file
unconfined_u:object_r:user_home_t:s0 file
[user@centos8 ~]$ sudo chcon unconfined_u:object_r:user_home_t:s0 /usr/sbin/sshd
[user@centos8 ~]$ sudo systemctl restart sshd
Job for sshd.service failed because the control process exited with error code.
See "systemctl status sshd.service" and "journalctl -xe" for details.
[user@centos8 ~]$ █
```

Ладно, с портами разобрались, что насчёт файлов? За них у нас отвечает `semanage fcontext`. Например, поищем, какие должны быть контексты у файлов `sshd`:

```
sudo semanage fcontext -l | grep sshd
```

Тут есть разные файлы, давайте испортим метку одного из файлов, например, `/usr/sbin/sshd`. Возьмём контекст любого файла:

```
ls -Z file
```

и скопируем. Для смены используем утилиту `chcon`:

```
sudo chcon ... /usr/bin/sshd
```

. После этого попытаемся рестартнуть сервис:

```
sudo systemctl restart sshd
```

Как видите, ssh отказывается запускаться.

```
[user@centos8 ~]$ sudo journalctl -eu sshd | tail -5
Apr 30 15:11:33 centos8 systemd[1]: Stopped OpenSSH server daemon.
Apr 30 15:11:33 centos8 systemd[1]: Starting OpenSSH server daemon...
Apr 30 15:11:33 centos8 systemd[1]: sshd.service: Main process exited, code=exited, status=203/EXEC
Apr 30 15:11:33 centos8 systemd[1]: sshd.service: Failed with result 'exit-code'.
Apr 30 15:11:33 centos8 systemd[1]: Failed to start OpenSSH server daemon.
[user@centos8 ~]$ sudo journalctl -e
Apr 30 15:11:34 centos8 sedispatch[1018]: AVC Message for setroubleshoot, dropping message
Apr 30 15:11:34 centos8 org.fedoraproject.Setroubleshootd[1066]: error: cannot open Packages index using db5 - P
Apr 30 15:11:34 centos8 org.fedoraproject.Setroubleshootd[1066]: error: cannot open Packages database in /var/lib/rpm
Apr 30 15:11:34 centos8 org.fedoraproject.Setroubleshootd[1066]: error: cannot open Packages index using db5 - P
Apr 30 15:11:34 centos8 org.fedoraproject.Setroubleshootd[1066]: error: cannot open Packages database in /var/lib/rpm
Apr 30 15:11:34 centos8 setroubleshoot[12438]: failed to retrieve rpm info for selinux-policy
Apr 30 15:11:34 centos8 platform-python[12438]: SELinux is preventing /usr/lib/systemd/systemd from execute access
***** Plugin restorecon (99.5 confidence) suggests *****

If you want to fix the label.
/usr/sbin/sshd default label should be sshd_exec_t.
Then you can run restorecon. The access attempt may have been st
Do
# /sbin/restorecon -v /usr/sbin/sshd
```

Посмотрим логи sshd:

```
sudo journalctl -eu sshd
```

Тут ни слова о причине проблемы. Поэтому просто проверим последние логи:

```
sudo journalctl -e
```

И тут сразу видим подсказку - SELinux запрещает systemd запустить sshd. Скорее всего нет политики, которая бы разрешала домену systemd запускать домен со скопированном типом user_home_t.

```
[user@centos8 ~]$ sudo restorecon -v /usr/sbin/sshd
Relabeled /usr/sbin/sshd from unconfined_u:object_r:user_home_t:s0 to unconfined_u:object_r:sshd_exec_t:s0
[user@centos8 ~]$ sudo systemctl restart sshd
[user@centos8 ~]$ █
```

Вернём контекст обратно, для этого есть утилита restorecon:

```
sudo restorecon -v /usr/sbin/sshd
```

Она восстанавливает заданный по умолчанию контекст файла. После этого рестарт сервиса прошёл успешно:

```
sudo systemctl restart sshd
```

Так вот, иногда, по определённым причинам, система может восстановить контекст всех файлов в системе. Это можно сделать и вручную, создав пустой файл `/.autorelabel` в корне и перезагрузившись. Этот процесс занимает определённое время и попросту запускать его не стоит, чтобы случайно ничего не сломать. Поэтому chcon можно использовать как временное решение, иначе случайное восстановление опять всё сломает. И вместо chcon рекомендуется использовать знакомый semanage fcontext.

```
[user@centos8 ~]$ ls -lZ file /usr/sbin/sshd
unconfined_u:object_r:user_home_t:s0 file
unconfined_u:object_r:sshd_exec_t:s0 /usr/sbin/sshd
[user@centos8 ~]$ sudo semanage fcontext -at sshd_exec_t /home/user/file
[user@centos8 ~]$ ls -Z file
unconfined_u:object_r:user_home_t:s0 file
[user@centos8 ~]$ sudo restorecon -v file
Relabeled /home/user/file from unconfined_u:object_r:user_home_t:s0 to unconfined_u:object_r:sshd_exec_t:s0
[user@centos8 ~]$ sudo semanage fcontext -l | grep /home/user/file
/home/user/file          all files      system_u:object_r:sshd_exec_t:s0
[user@centos8 ~]$ █
```

Например, укажем в политике, чтобы у файла был контекст, как у sshd:

```
ls -1Z file /usr/sbin/sshd
sudo semanage fcontext -at sshd_exec_t /home/user/file
```

Обратите внимание, что нужно использовать полный путь к файлу. Если проверить после выполнения команды:

```
ls -Z file
```

казалось бы, ничего не изменилось. Но на самом деле fcontext меняет значение в политике, а не сразу в файле. Поэтому достаточно просто восстановить контекст по умолчанию:

```
sudo restorecon -v file
```

и мы увидим, что тип контекста сменился на sshd_exec_t. Также сможем найти наш файл в политиках:

```
sudo semanage fcontext -l | grep /home/user/file
```

```
[user@centos8 ~]$ sudo semanage fcontext -at sshd_exec_t "/home/user/Documents(/.*)?"
[sudo] password for user:
[user@centos8 ~]$ sudo restorecon -rv /home/user/Documents/
Relabeled /home/user/Documents from unconfined_u:object_r:user_home_t:s0 to unconfined_u:object_r:sshd_exec_t:s0
Relabeled /home/user/Documents/test from unconfined_u:object_r:user_home_t:s0 to unconfined_u:object_r:sshd_exec_t:s0
Relabeled /home/user/Documents/this from unconfined_u:object_r:user_home_t:s0 to unconfined_u:object_r:sshd_exec_t:s0
Relabeled /home/user/Documents/is from unconfined_u:object_r:user_home_t:s0 to unconfined_u:object_r:sshd_exec_t:s0
Relabeled /home/user/Documents/new from unconfined_u:object_r:user_home_t:s0 to unconfined_u:object_r:sshd_exec_t:s0
Relabeled /home/user/Documents/dir from unconfined_u:object_r:user_home_t:s0 to unconfined_u:object_r:sshd_exec_t:s0
Relabeled /home/user/Documents/New Dir from unconfined_u:object_r:user_home_t:s0 to unconfined_u:object_r:sshd_exec_t:s0
```

Смена контекста файлов и директорий часто бывает нужна, когда вы меняете стандартные настройки демона. По умолчанию у демона есть право работать с определёнными файлами/директориями, а вам бывает нужно настроить директорию в нестандартном месте. Вы меняете директорию в настройках, а сервис отказывается перезапускаться, потому что не может прочесть файлы, так как контекст не соответствует политике. И для этого есть определённое выражение, рассмотрим на примере Documents:

```
sudo semanage fcontext -at sshd_exec_t "/home/user/Documents(/.*)?"
```

Это меняет контекст в политиках ко всем файлам и поддиректориям. И в конце не забываем восстановить контекст:

```
sudo restorecon -rv /home/user/Documents
```

Тут уже restorecon с ключом -r - рекурсивно.

```
[user@centos8 ~]$ sudo semanage boolean -l | grep mozilla
mozilla_plugin_bind_unreserved_ports (off , off) Allow mozilla to plugin bind unreserved ports
mozilla_plugin_can_network_connect (on , on) Allow mozilla to plugin can network connect
mozilla_plugin_use_bluejeans (off , off) Allow mozilla to plugin use bluejeans
mozilla_plugin_use_gps (off , off) Allow mozilla to plugin use gps
mozilla_plugin_use_spice (off , off) Allow mozilla to plugin use spice
mozilla_read_content (off , off) Allow mozilla to read content
unconfined_mozilla_plugin_transition (on , on) Allow unconfined to mozilla plugin transition
[user@centos8 ~]$ sudo getsebool -a | grep mozilla_plugin_use_gps
mozilla_plugin_use_gps --> off
[user@centos8 ~]$ sudo setsebool mozilla_plugin_use_gps on
[user@centos8 ~]$ sudo getsebool -a | grep mozilla_plugin_use_gps
mozilla_plugin_use_gps --> on
[user@centos8 ~]$
```

Иногда бывает, что в рамках безопасности контекст не очень подходит или очень сложно переделать контекст файлов. Поэтому частично возможности SELinux вынесены в двоичные опции - типа разрешить что-то или запретить. Их можно увидеть с помощью semanage boolean или getsebool - например, есть возможность заблокировать плагинам мозиллы доступ к использованию gps:

```
sudo semanage boolean -l | grep mozilla
sudo getsebool -a | grep mozilla_plugin_use_gps
```

Как видите, эта опция выключена. Её можно включить используя команду setsebool:

```
sudo setsebool mozilla_plugin_use_gps on
```

В конце можно указать 0, 1 или on, off. И чтобы увидеть изменения используем getsebool:

```
sudo getsebool -a | grep mozilla_plugin_use_gps
```

```
[user@centos8 ~]$ sudo setsebool -P mozilla_plugin_use_gps on
[user@centos8 ~]$ sudo semanage boolean -l | grep mozilla_plugin_use_gps
mozilla_plugin_use_gps (on , on) Allow mozilla to plugin use gps
[user@centos8 ~]$
```

Но setsebool применяет изменения только в текущей сессии. Чтобы сохранить это значение в политиках, следует использовать ключ -P:

```
sudo setsebool -P mozilla_plugin_use_gps on
sudo semanage boolean -l | grep mozilla
```

```
[user@centos8 ~]$ sudo semanage export -f myselinux
[user@centos8 ~]$ cat myselinux
boolean -D
login -D
interface -D
user -D
port -D
node -D
fcontext -D
module -D
ibendport -D
ibpkey -D
boolean -m -l mozilla_plugin_use_gps
port -a -t ssh_port_t -r 's0' -p tcp 2233
fcontext -a -f a -t sshd_exec_t -r 's0' '/home/user/Documents(/.*)?'
fcontext -a -f a -t sshd_exec_t -r 's0' '/home/user/file'
fcontext -a -f a -t mount_exec_t -r 's0' '/opt/VBoxGuestAdditions-6.1.16/other/mount.vboxsf'
fcontext -a -f a -t mount_exec_t -r 's0' '/opt/VBoxGuestAdditions-6.1.4/other/mount.vboxsf'
fcontext -a -f a -t bin_t -r 's0' '/usr/bin/VBoxClient'
[user@centos8 ~]$ sudo semanage import -f myselinux
ValueError: Port tcp/2233 already defined
[user@centos8 ~]$ █
```

Ну и напоследок, мы можем все наши изменения вытащить в виде файла с помощью export:

```
sudo semanage export -f myselinux
cat myselinux
```

Как видите, тут у нас и изменённый порт, и типы контекста и двоичная опция. Потом мы этот файл можем использовать как шаблон для быстрого применения на наших серверах.

Подведём итоги. Сегодня мы с вами разобрали SELinux - систему, которая играет важную роль в безопасности. Она даёт возможность детальной настройки принудительного контроля доступа. Конечно, мы разобрали только вершину айсберга - поняли, как она работает, как смотреть информацию и решать базовые проблемы. Но для основ этого вполне достаточно. Со временем мы не раз ещё столкнёмся с SELinux-ом и разберём больше примеров и возможностей.

И да, чуть не забыл. В начале я рассказывал про важность ssh ключей и про то, что Firefox не должен иметь к ним доступа. Но при этом спокойно нашёл и открыл ключи, и SELinux ничего не сделал. Просто применять политики от пользовательских программ к пользовательским файлам - невероятно сложно и многое легко сломать. Такие политики будут мешать пользователям и те будут постоянно выключать SELinux. Конечно, при максимальной безопасности это можно настроить, но в обыденной жизни между удобством и безопасностью должен быть компромисс.

2.46 46. Межсетевой экран - firewalld

2.46.1 46. Межсетевой экран - firewalld

```
[doctor@tardis]——[~]
└─$ ssh vbox
Activate the web console with: systemctl enable --now cockpit.socket

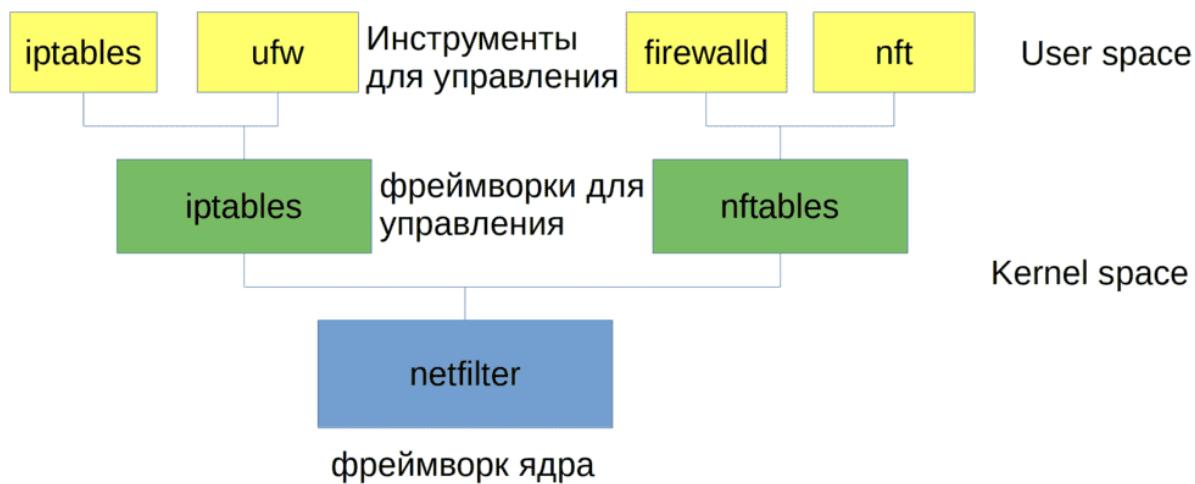
Last login: Sat May  8 10:58:43 2021 from 192.168.31.227
[user@centos8 ~]$ ps -ef | grep sshd
root      1185      1  0 10:58 ?        00:00:00 /usr/sbin/sshd -D -oCiphers=aes256-gcm0
s256-ctr,aes256-cbc,aes128-gcm@openssh.com,aes128-ctr,aes128-cbc -oMACs= hmac-sha2-256-0
c-128-etm@openssh.com,hmac-sha2-512-etm@openssh.com,hmac-sha2-256,hmac-sha1,umac-128@0
ss=ss-gex-sha1-,ss=group14-sha1- -oKexAlgorithms=curve25519-sha256,curve25519-sha256@
```

В прошлый раз мы с вами разобрали SELinux - систему безопасности, защищающую от уязвимостей. Допустим, если у SSH демона есть какая-то недоработка, взломщики не логинясь через него попытаются запустить какую-то другую программу, например, passwd, чтобы задать пароль root пользователю и получить доступ на сервер. Это абстрактный пример, в реале атаки несколько другие, но суть похожая. Так вот, если в системе нет SELinux-а, злоумышленники получат полный доступ к системе, так как sshd работает от пользователя root:

```
ps -ef | grep sshd
```

А если SELinux есть, то он не позволит процессу sshd запустить passwd или любую другую программу, так как контексты не позволяют, тем самым предотвратит атаку.

Но надеяться только на него не стоит - хорошая система безопасности состоит из многих уровней. И одну из главных ролей играет файрвол, который позволяет контролировать соединения ещё до того, как они доберутся до демона. Ведь если злоумышленник не сможет подключиться к SSH - то и не сможет эксплуатировать уязвимости.



Если помните, я говорил, что в ядре Linux есть фреймворк LSM, который является каркасом для SELinux и других подобных программ. Так вот, тут похожая ситуация - в ядре есть фреймворк netfilter, который является каркасом для различных инструментов по управлению файрволом. Самой блокировкой занимается именно netfilter, но он работает на уровне ядра, а чтобы задавать настройки нужны инструменты в пространстве пользователя. Раньше был популярен iptables, но в 2014 его заменил nftables, хотя до сих пор iptables где-то ещё используется. Но при этом сам nftables также выступает каркасом, есть утилита nft для создания политик для nftables, а он уже преобразует эти политики в понятный вид для netfilter. Но на RHEL-based дистрибутивах для управления nftables по умолчанию используется другой инструмент - firewalld. И о нём сегодня мы и будем говорить. Чтобы обобщить, firewalld - это инструмент для управления файрволом, использующий в качестве бэкенда nftables, который в свою очередь является инструментом для управления фреймворком ядра netfilter. Это так, в целях общего развития.

```
[user@centos8 ~]$ systemctl status firewalld.service
● firewalld.service - firewalld - dynamic firewall daemon
  Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; vendor preset: enabled)
  Active: active (running) since Sat 2021-05-08 12:38:14 +04; 19s ago
    Docs: man:firewalld(1)
 Main PID: 1066 (firewalld)
   Tasks: 2 (limit: 11424)
  Memory: 30.3M
    CGroup: /system.slice/firewalld.service
            └─1066 /usr/libexec/platform-python -s /usr/sbin/firewalld --nofork --nopid

May 08 12:38:13 centos8 systemd[1]: Starting firewalld - dynamic firewall daemon...
May 08 12:38:14 centos8 systemd[1]: Started firewalld - dynamic firewall daemon.
May 08 12:38:15 centos8 firewalld[1066]: WARNING: AllowZoneDrifting is enabled. This is considered a
[user@centos8 ~]$ firewall-cmd
```

При этом сам firewalld является демоном:

```
sudo systemctl status firewalld
```

а для просмотра информации и управления используется команда `firewall-cmd`. В некоторых случаях может понадобится выключить файрвол, например, у вас что-то не работает и есть подозрения, что подключение блокирует файрвол. В таких случаях достаточно остановить этот сервис. Но просто так это делать не стоит.

```
[user@centos8 ~]$ sudo firewall-cmd --
--add-forward-port=          --info-icmptype=
--add-icmp-block=            --info-ipset=
--add-icmp-block-inversion=  --info-service=
--add-interface=             --info-zone=
--add-lockdown-whitelist-command= --list-all
--add-lockdown-whitelist-context= --list-all-zones
--add-lockdown-whitelist-uid=  --list-forward-ports
--add-lockdown-whitelist-user= --list-icmp-blocks
--add-masquerade              --list-interfaces
--add-port=                   --list-lockdown-whitelist-commands
--add-protocol=               --list-lockdown-whitelist-contexts
--add-rich-rule=              --list-lockdown-whitelist-uids
--add-service=                --list-lockdown-whitelist-users
--add-source=                 --list-ports
--add-source-port=            --list-protocols
--change-interface=           --list-rich-rules
--change-source=              --list-services
--change-zones=               --list-source-ports
--complete-reload=            --list-sources
--direct=                     --lockdown-off
--get-active-zones=           --lockdown-on
--get-default-zone=           --panic-off
--get-description=            --panic-on
--get-helpers=                 --permanent
--get-icmptypes=               --query-forward-port=
--get-ipset-types=             --query-icmp-block=
--get-log-denied=              --query-icmp-block-inversion
--get-services=                  --query-interface=
--get-short=                   --query-lockdown
--get-zone-of-interface=       --query-lockdown-whitelist-command=
--get-zones=                   --query-lockdown-whitelist-context=
--help=                        --query-lockdown-whitelist-uid=
--info-helpers=                 --query-lockdown-whitelist-user=
--query-forward-port=          --remove-forward-port=
--remove-icmp-block=           --remove-icmp-block=
--remove-icmp-block-inversion= --remove-icmp-block-inversion
--remove-interface=             --remove-interface=
--remove-lockdown-whitelist-command= --remove-lockdown-whitelist-context=
--remove-lockdown-whitelist-uid= --remove-lockdown-whitelist-user=
--remove-lockdown-whitelist-user= --remove-masquerade
--remove-protocol=              --remove-protocol=
--remove-rich-rule=             --remove-rich-rule=
--remove-service=               --remove-service=
--remove-source=                 --remove-source=
--remove-source-port=           --remove-source-port=
--set-default-zone=             --set-default-zone=
--set-description=              --set-description=
--set-log-denied=                --set-short=
--state=                         --state
--version=                      --zone=
```

У `firewall-cmd` большой функционал, поэтому много различных ключей, и почти все начинаются на две дефиса. Но пусть это вас не пугает, потому что всё интуитивно понятно.

```
[user@centos8 ~]$ sudo firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp0s3
  sources:
  services: cockpit dhcpcv6-client ssh
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[user@centos8 ~]$ █
```

Начнём с простого - list-all:

```
sudo firewall-cmd --list-all
```

Она показывает всю информацию - что разрешено, что запрещено и к кому это относится. Пока остановимся на строке services - здесь перечислены разрешённые сервисы - cockpit, dhcpcv6-client и ssh. В firewalld, сервисы - это шаблоны, которые упрощают настройку и понимание. Смотришь и видишь, что ssh разрешён.

```
[user@centos8 ~]$ sudo firewall-cmd --info-service=
Display all 169 possibilities? (y or n)
[user@centos8 ~]$ sudo firewall-cmd --info-service=ssh
ssh
  ports: 22/tcp
  protocols:
  source-ports:
  modules:
  destination:
  includes:
  helpers:
[user@centos8 ~]$ cat /lib/firewalld/services/ssh.xml
<?xml version="1.0" encoding="utf-8"?>
<service>
  <short>SSH</short>
  <description>Secure Shell (SSH) is a protocol for logging into and executing commands on remote machines. It provides encrypted communications. If you plan on accessing your machine remotely via SSH over a firewalled interface, please make sure the openssh-server package is installed for this option to be useful.</description>
  <port protocol="tcp" port="22"/>
</service>
[user@centos8 ~]$ █
```

Но настройки файрвола никак не связаны с настройками демонов. Если поменять стандартный порт демона, того же sshd - это не значит, что и на firewalld он изменится. Чтобы посмотреть информацию про сервис, можно использовать опцию --info-service:

```
sudo firewall-cmd --info-service=ssh
```

Как тут видно, ssh используется 22 порт tcp. Настройки сервисов файрвола прописаны в файлах в /lib/firewalld/services:

```
cat /lib/firewalld/services/ssh
```

В целом сервис отличается от порта - в одном сервисе может быть несколько портов, сервис можно жестко привязать к определённым адресам, исходящим портам, другим сервисам и т.п.

```
[user@centos8 ~]$ sudo nano /etc/ssh/sshd_config
[user@centos8 ~]$ sudo grep Port /etc/ssh/sshd_config
Port 2233
#GatewayPorts no
[user@centos8 ~]$ sudo systemctl restart sshd
[user@centos8 ~]$
[doctor@tardis ~]$ ssh vbox
ssh: connect to host 192.168.31.5 port 22: Connection refused
[x]-[doctor@tardis ~]
$ ssh vbox -p 2233
ssh: connect to host 192.168.31.5 port 2233: No route to host
[x]-[doctor@tardis ~]
$ ssh vbox -p 2233
ssh: connect to host 192.168.31.5 port 2233: No route to host
[x]-[doctor@tardis ~]
$
```

Давайте, к примеру, на sshd поставим нестандартный порт:

```
sudo nano /etc/ssh/sshd_config
sudo systemctl restart sshd
```

Хоть меня и не выбило из текущей сессии, но при попытке заново подключиться по стандартному порту ничего не получится. Но и указав изменённый порт также ничего не выходит. Потому что в файрволе не настроено, чтобы он кого-то пускал по порту 2233. Правда в логах об этом не будет ни слова - обычно, нет необходимости логировать каждое запрещённое соединение, а их, как правило, так много, что для логов не хватит места.

```
[user@centos8 ~]$ sudo firewall-cmd --add-port=2233/tcp
success
[user@centos8 ~]$ sudo firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp0s3
  sources:
  services: cockpit dhcpcv6-client ssh
  ports: 2233/tcp
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[user@centos8 ~]$ [x]-[doctor@tardis ~]
$ ssh vbox -p 2233
Activate the web console with: systemctl enable --now cockpit.socket
Last login: Sat May  8 12:38:20 2021 from 192.168.31.227
[user@centos8 ~]$
```

Самый простой вариант - разрешить порт, add port:

```
sudo firewall-cmd --add-port=2233/tcp
```

Т.е. после знака равно нужно указать номер порта, слэш и нужный протокол - tcp или udp. После этого подключение прошло нормально:

```
ssh vbox -p 2233
```

```
[user@centos8 ~]$ sudo firewall-cmd --reload
success
[user@centos8 ~]$ sudo firewall-cmd --add-port=2233/tcp --permanent
success
[user@centos8 ~]$ sudo firewall-cmd --reload
success
[user@centos8 ~]$ [doctor@tardis ~]
$ ssh vbox -p 2233
ssh: connect to host 192.168.31.5 port 2233: No route to host
[x]-[doctor@tardis ~]
$ ssh vbox -p 2233
ssh: connect to host 192.168.31.5 port 2233: No route to host
[x]-[doctor@tardis ~]
$ ssh vbox -p 2233
Activate the web console with: systemctl enable --now cockpit.socket
Last login: Sat May  8 13:28:38 2021 from 192.168.31.227
[user@centos8 ~]$
```

Но эти изменения только на текущую сессию. Если перезагрузить компьютер или сервис файрвола, то опять подключения будут блокироваться:

```
sudo firewall-cmd --reload  
ssh vbox -p 2233
```

Если мы хотим, чтобы изменения остались навсегда, нужно использовать ключ **--permanent**:

```
sudo firewall-cmd --add-port=2233/tcp --permanent
```

При этом изменения происходят в настройках, а не в текущей сессии, поэтому после команды следует перезапустить файрвол:

```
sudo firewall-cmd --reload
```

После чего всё заработает:

```
ssh vbox -p 2233
```

```
[user@centos8 ~]$ sudo firewall-cmd --list-services  
cockpit dhcpcv6-client ssh  
[user@centos8 ~]$ sudo firewall-cmd --list-ports  
2233/tcp  
[user@centos8 ~]$ sudo firewall-cmd --remove-service=dhcpcv6-client --permanent  
success  
[user@centos8 ~]$ sudo firewall-cmd --reload  
success  
[user@centos8 ~]$ sudo firewall-cmd --list-services  
cockpit ssh  
[user@centos8 ~]$ █
```

Правда сейчас сложилась ситуация, что у нас на файрволе открыт и 22 порт, как сервис, и 2233:

```
sudo firewall-cmd --list-services  
sudo firewall-cmd --list-ports
```

Существует заблуждение, что открытый порт - это опасно, но сам по себе порт не несёт никакой угрозы, она может быть в программе, которая стоит за этим портом, в её уязвимостях и неправильных настройках. И то что сейчас на файрволе остался открытый 22 порт - ничего страшного, всё равно за ним никакого сервиса нет. Однако всё равно стоит убирать лишнее и приводить в порядок - безопасность комплексная штука, а хаос только делает хуже. Поэтому уберём отсюда сервис dhcpcv6-client - он нужен для ipvb, а мы им не пользуемся:

```
sudo firewall-cmd --remove-service=dhcpcv6-client --permanent
```

Команды довольно простые - list показывает, add добавляет, remove убирает, permanent сохраняет в настройках, reload перезагружает.

```
[user@centos8 ~]$ sudo firewall-cmd --permanent --service=ssh --remove-port=22/tcp
success
[user@centos8 ~]$ sudo firewall-cmd --permanent --service=ssh --add-port=2233/tcp
success
[user@centos8 ~]$ sudo firewall-cmd --reload
success
[user@centos8 ~]$ sudo firewall-cmd --info-service=ssh
ssh
  ports: 2233/tcp
  protocols:
  source-ports:
  modules:
  destination:
  includes:
  helpers:
[user@centos8 ~]$ █
```

Можно создавать или изменять сервисы. К примеру, сделаем так, чтобы порт 2233 не висел отдельно, а был в самом сервисе. Для этого уберём из сервиса 22 порт - remove-port:

```
sudo firewall-cmd --permanent --service=ssh --remove-port=22/tcp
```

Потом добавим порт 2233 - add-port:

```
sudo firewall-cmd --permanent --service=ssh --add-port=2233/tcp
```

Перезапустим и проверим:

```
sudo firewall-cmd --reload
sudo firewall-cmd --info-service=ssh
```

```
[user@centos8 ~]$ sudo cat /etc/firewalld/services/ssh.xml
<?xml version="1.0" encoding="utf-8"?>
<service>
  <short>SSH</short>
  <description>Secure Shell (SSH) is a protocol for logging into and executing commands on remote encrypted communications. If you plan on accessing your machine remotely via SSH over a firewalled interface the openssh-server package installed for this option to be useful.</description>
  <port port="2233" protocol="tcp"/>
</service>
[user@centos8 ~]$ █
```

Если стандартные настройки лежат в `/lib/firewalld`, то наши изменения сохраняются в `/etc/firewalld`:

```
sudo cat /etc/firewalld/services/ssh.xml
```

Можно было просто скопировать файл сервиса из lib в etc, сделать изменения в самом файле и перезапустить сервис. Таким же образом можно скопировать существующий сервис и создать новый.

```
[user@centos8 ~]$ sudo firewall-cmd --remove-port=2233/tcp --permanent  
success  
[user@centos8 ~]$ sudo firewall-cmd --reload  
success  
[user@centos8 ~]$ sudo firewall-cmd --list-all  
public (active)  
  target: default  
  icmp-block-inversion: no  
  interfaces: enp0s3  
  sources:  
  services: cockpit ssh  
  ports:  
  protocols:  
  masquerade: no  
  forward-ports:  
  source-ports:  
  icmp-blocks:  
  rich rules:  
[user@centos8 ~]$ █
```

Ну и раз уж мы в самом сервисе заменили порт, то можем убрать отдельно указанный порт:

```
sudo firewall-cmd --remove-port=2233/tcp --permanent  
sudo firewall-cmd --reload  
sudo firewall-cmd --list-all
```

```
[user@centos8 ~]$ tail /etc/protocols  
mpls-in-ip      137      MPLS-in-IP  
manet    138      manet      # MANET Protocols  
hip      139      HIP        # Host Identity Protocol  
shim6    140      Shim6      # Shim6 Protocol  
wesp     141      WESP       # Wrapped Encapsulating Security Payload  
rohc    142      ROHC      # Robust Header Compression  
# 143-252 Unassigned                         [IANA]  
# 253      Use for experimentation and testing      [RFC3692]  
# 254      Use for experimentation and testing      [RFC3692]  
# 255      Reserved                           [IANA]  
[user@centos8 ~]$ cat /etc/protocols | grep ospf  
ospf    89      OSPFIGP      # Open Shortest Path First IGP  
[user@centos8 ~]$ sudo firewall-cmd --add-protocol=ospf --permanent █
```

Кроме сервисов и портов, также можно разрешать сетевые протоколы. Они указаны не в firewalld, а в самой системе и их список можно посмотреть в файле `/etc/protocols`:

```
tail /etc/protocols
```

Для примера, есть протокол `ospf`:

```
grep ospf /etc/protocols
```

которым пользуются роутеры для обмена информацией о сетях. И если бы мы настраивали Linux как роутер, то нужно было бы добавить протокол `ospf` на файрволе. Но сейчас нам это не нужно.

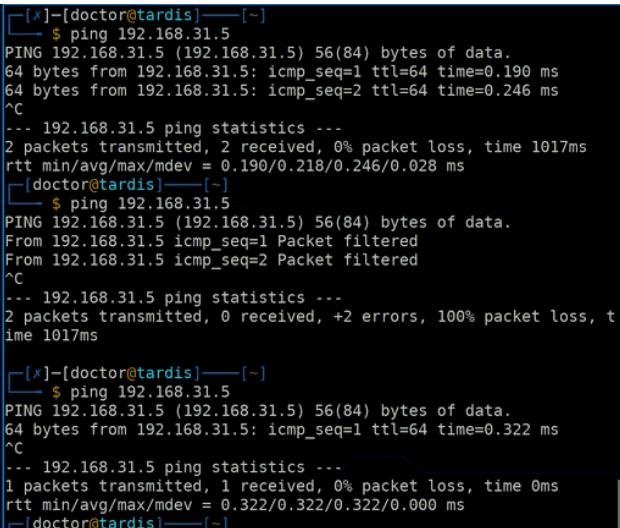
```
[user@centos8 ~]$ sudo firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp0s3
  sources:
  services: cockpit ssh
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[user@centos8 ~]$ sudo firewall-cmd --get-icmptypes
address-unreachable bad-header beyond-scope communication-prohibited destination-unreachable echo-reply echo-request failed-policy fragmentation-needed host-precedence-violation host-prohibited host-redirect host-unknown host-unreachable ip-header-bad neighbour-advertisement neighbour-solicitation network-prohibited network-redirect network-unknown network-unreachable no-route packet-too-big parameter-problem port-unreachable precedence-cutoff protocol-unreachable redirect required-option-missing router-advertisement router-solicitation source-quench source-route-failed time-exceeded timestamp-reply timestamp-request tos-host-redirect tos-host-unreachable tos-network-redirect tos-network-unreachable ttl-zero-during-reassembly ttl-zero-during-transit unknown-head-r-type unknown-option
[user@centos8 ~]$
```

Также в list-all можно увидеть icmp-blocks и icmp-blocks-inversion:

```
sudo firewall-cmd --list-all
sudo firewall-cmd --get-icmptypes
```

ICMP - это специальный протокол, который используют для проверки сети. Помните, в основе сетей мы проверяли доступность сети с помощью ping или traceroute? Эти программы как раз используют ICMP. Так вот, мы использовали ping чтобы понять, доступен ли другой компьютер. Зачастую в интернете боты сканируют сеть с помощью ICMP, чтобы увидеть доступные IP адреса, и потом пытаются их взломать. Нередко администраторы скрывают сервера, блокируя ICMP запросы. Конечно, открытые порты всё ещё будут доступны для подключения, но часть ботов отсеется.

```
[user@centos8 ~]$ sudo firewall-cmd --add-icmp-block=^C
[user@centos8 ~]$ sudo firewall-cmd --add-icmp-block-inversion
success
[user@centos8 ~]$ sudo firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: yes
  interfaces: enp0s3
  sources:
  services: cockpit ssh
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[user@centos8 ~]$ sudo firewall-cmd --remove-icmp-block-inversion
success
[user@centos8 ~]$ sudo firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp0s3
  sources:
  services: cockpit ssh
  ports:
  protocols:
```



С помощью firewalld мы можем заблокировать все ICMP запросы, либо определённые типы. Второе используется реже, но всё же можно, с помощью опции --add-icmp-block и указанием типа. А если блокировать всё, то можно использовать опцию --add-icmp-block-inversion, но перед этим попробуем пингануть виртуалку. Пинг идёт. Потом попробуем включить эту опцию:

```
sudo firewall-cmd --add-icmp-block-inversion
```

Теперь мы видим, что ping выдаёт информацию «Packet filtered». Так мы сразу понимаем, что кто-то фильтрует трафик, а именно файрвол. И чтобы всё вернуть - --remove-icmp-block-inversion. Пинг опять пошёл. И не забывайте, что для сохранения настроек нужно использовать ключ --permanent.

Вас должно было смутить, что по пингу всё равно можно было понять, компьютер доступен или нет,

ведь он ответил нам - packet filtered. Такие блокировки ICMP не столько для скрытия доступности хоста, сколько для скрытия информации о сети, допустим, когда провайдер скрывает адреса своих роутеров. В таких случаях на ICMP запросы посылается ответ REJECT, чтобы было понятно, что эта информация скрыта.

```
[user@centos8 ~]$ sudo firewall-cmd --list-all | head -3
public (active)
  target: default
  icmp-block-inversion: no
[user@centos8 ~]$ sudo firewall-cmd --set-target=DROP --permanent
success
[user@centos8 ~]$ sudo firewall-cmd --reload
success
[user@centos8 ~]$ 
```

```
[x]-[doctor@tardis]—[~]
└─$ ping -c 1 192.168.31.5
PING 192.168.31.5 (192.168.31.5) 56(84) bytes of data.
--- 192.168.31.5 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
[x]-[doctor@tardis]—[~]
└─$ 
```

Если же мы хотим именно скрыть компьютер, то нужно поменять другую опцию - target:

```
sudo firewall-cmd --list-all | head -3
```

Для этого нужно использовать ответ DROP:

```
sudo firewall-cmd --set-target=DROP --permanent
sudo firewall-cmd --reload
```

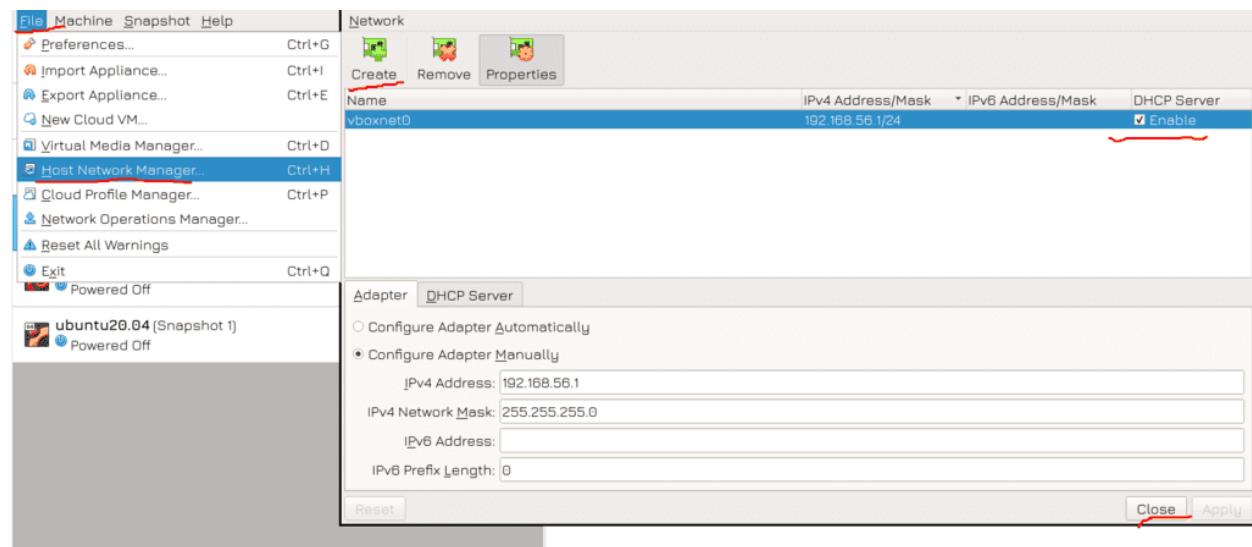
После этого файрвол будет «дропать» пакеты, т.е. выбрасывать. Если при REJECT мы посылали обратный ответ, то при DROP вторая сторона ничего в ответ не получит, поэтому будет считать, что этот IP недоступен.

```
[user@centos8 ~]$ sudo firewall-cmd --set-target=default --permanent
success
[user@centos8 ~]$ sudo firewall-cmd --reload
success
[user@centos8 ~]$ sudo firewall-cmd --set-target=DROP --permanent
success
[user@centos8 ~]$ sudo firewall-cmd --reload
success
[user@centos8 ~]$ 
```

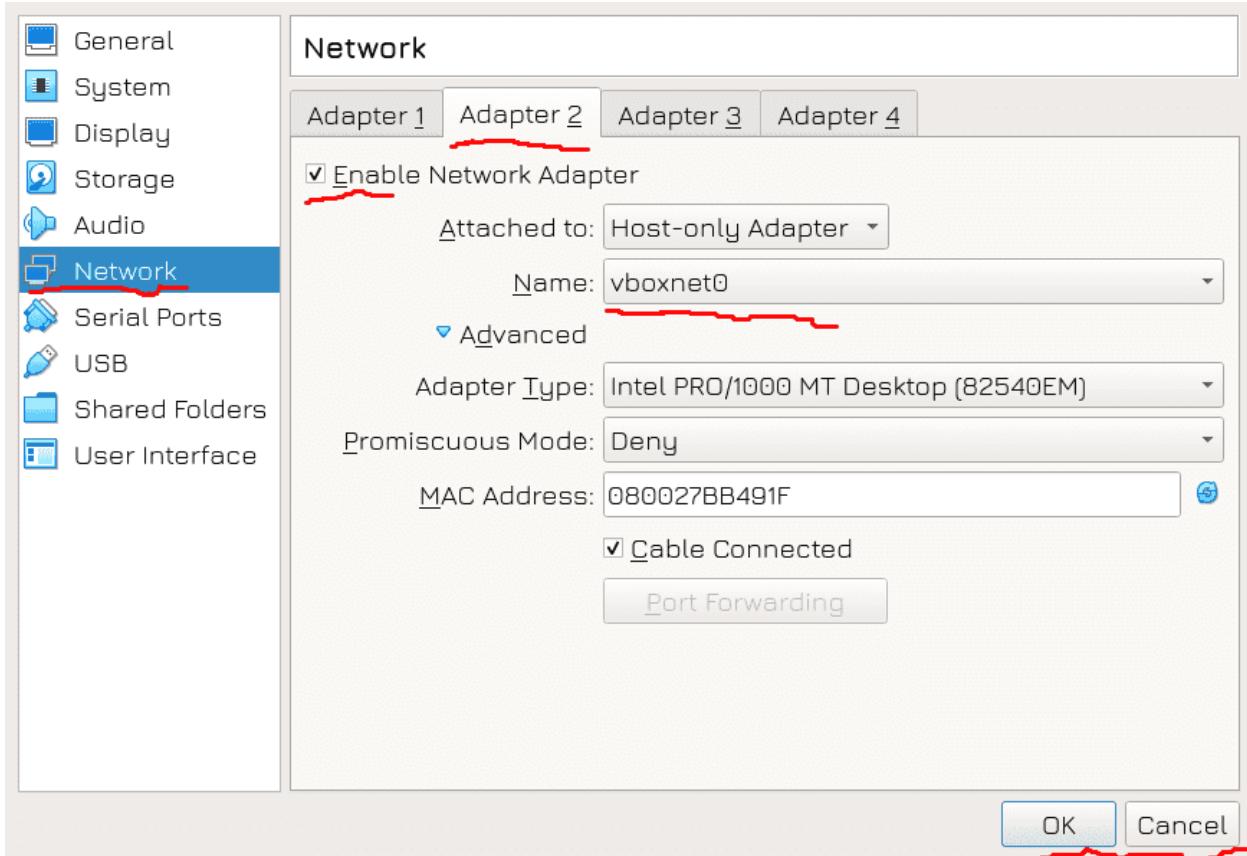
```
[x]-[doctor@tardis]—[~]
└─$ ssh vbox -p 1111
ssh: connect to host 192.168.31.5 port 1111: No route to host
[x]-[doctor@tardis]—[~]
└─$ ssh vbox -p 1111

```

При этом дропаются не только ICMP пакеты, а любые подключения, не соответствующие разрешённым. К примеру, когда target=default, то при попытке подключиться к запрещённому порту я сразу получаю ответ, что не могу. Так можно понять, что где-то блокирует файрвол. А если стоит режим DROP - то программа, которая пытается подключиться, допустим, ssh, будет долго ждать и просто не дождётся ответа. Поэтому это более предпочтительный способ скрыть компьютер.



Прежде чем продолжим, давайте добавим ещё один адаптер для виртуалки. Для этого сначала её нужно выключить, затем в меню самого виртуалбокса зайти в File - Host Network Manager и проверить, есть ли там сеть, т.е. в табличке под Name должна быть какая-то запись. Если её нет, то нажмите Create и поставьте галочку у «DHCP Server», затем Close.



После чего надо зайти в настройки самой виртуалки, выбрать Network - Adapter 2, поставить галочку на включение адаптера, выбрать тип «Host-only Adapter» и выбрать из списка созданный адаптер, затем нажать OK. После этого можно включать виртуалку.

```
[user@centos8 ~]$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    group default qlen 1000
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
    net6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:de:15:ba brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.3/24 brd 192.168.56.255 scope global dynamic noprefixroute enp0s8
        valid_lft 497sec preferred_lft 497sec
        inet6 fe80::60e3:263b:871a:9ec4/64 scope link noprefixroute
            valid_lft forever preferred_lft forever
3: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:ce:55:bb brd ff:ff:ff:ff:ff:ff
    inet 192.168.31.5/24 brd 192.168.31.255 scope global noprefixroute enp0s3
        valid_lft forever preferred_lft forever
        inet 192.168.31.6/24 brd 192.168.31.255 scope global secondary noprefixroute enp0s3
            valid_lft forever preferred_lft forever
[doctor@tardis:~] $ ping 192.168.56.3
PING 192.168.56.3 (192.168.56.3) 56(84) bytes of data.
^C
--- 192.168.56.3 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2039ms
[x]-[doctor@tardis:~] $ ssh user@192.168.56.3 -p 2233
The authenticity of host '[192.168.56.3]:2233 ([192.168.56.3]:2233)' can't be established.
ED25519 key fingerprint is SHA256:S0+wghxLE7vh0SGL2tH+ojf0vZDv/jnAhXYEG/HvRM.
This host key is known by the following other names/addresses:
~/ssh/known_hosts:150: 192.168.31.5
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[192.168.56.3]:2233' (ED25519) to the list of known hosts.
Activate the web console with: systemctl enable --now cockpit.socket
Last login: Mon May 10 21:51:40 2021 from 192.168.31.227
[user@centos8 ~]$
```

После изменений в виртуалке появится новый интерфейс:

```
ip a
```

У меня он назывался enp0s8 и получил адрес 192.168.56.3, у вас результат может отличаться. Если попытаемся с хоста пропинговать этот адрес, то ничего не получится - но это и не удивительно, мы ведь запретили пинги. А вот подрубиться по этому адресу к SSH можно, естественно, указав нужный порт при подключении:

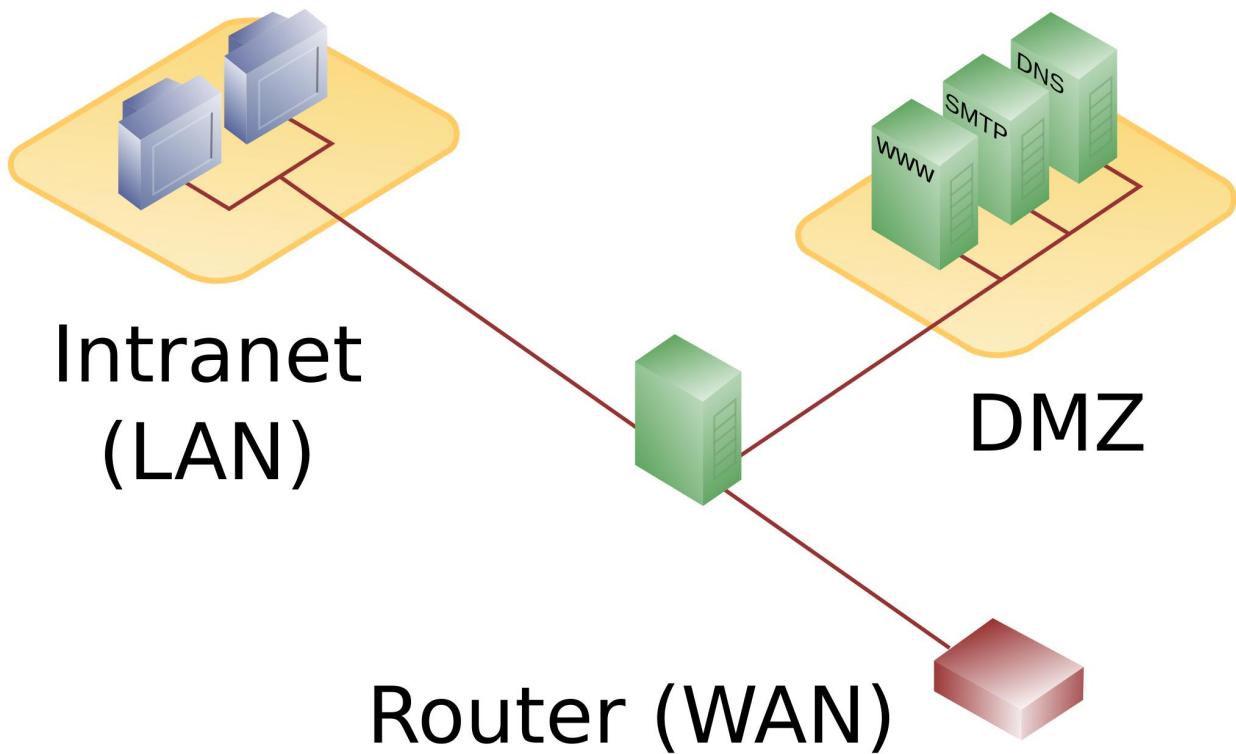
```
ssh user@192.168.56.3 -p 2233
```

```
[user@centos8 ~]$ sudo firewall-cmd --list-all
public (active)
  target: DROP
  icmp-block-inversion: no
  interfaces: enp0s3 enp0s8
  sources:
  services: cockpit ssh
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[user@centos8 ~]$ █
```

А интерфейс мы добавили не просто так. Если посмотреть файрвол:

```
sudo firewall-cmd --list-all
```

можно заметить, что в строке interfaces появился также enp0s8 - наш новый интерфейс. Зачем на файрволе интерфейсы?



Начнём с того, что нередко сервера могут смотреть сразу в несколько сетей. Например, ваш роутер смотрит одновременно в две сети - в домашнюю и в интернет. Роутер это тоже своего рода сервер, да и из линукса можно сделать маршрутизатор. Так вот, к роутеру можно подключаться, чтобы им управлять. Но позволять кому-то подключаться к роутеру из интернета не очень хорошо, это не безопасно. Поэтому на нём можно настроить, чтобы управлять им можно было только с домашней сети, а не из интернета. Т.е. у вашего роутера есть две так называемые зоны - внешняя сеть и локальная. Внешняя сеть - опасная и недоверенная, все запросы оттуда нужно блокировать. А локальная сеть - относительно безопасная. Если пинги из внешней зоны стоит дропать, то из домашней наоборот, стоит оставить, чтобы было понятно - работает ли роутер. Т.е. всё просто - разные зоны - разное доверие - нужно разное запрещать или разрешать.

```
[user@centos8 ~]$ sudo firewall-cmd --get-zones
block dmz drop external home internal libvirt public trusted work
[user@centos8 ~]$ sudo firewall-cmd --get-default-zone
public
[user@centos8 ~]$ sudo firewall-cmd --list-all --zone=block | head -3
block
  target: %%REJECT%%
  icmp-block-inversion: no
[user@centos8 ~]$ sudo firewall-cmd --list-all --zone=drop | head -3
drop
  target: DROP
  icmp-block-inversion: no
[user@centos8 ~]$ sudo firewall-cmd --list-all --zone=trusted | head -3
trusted
  target: ACCEPT
  icmp-block-inversion: no
[user@centos8 ~]$ █
```

На многих современных файрволах можно настроить различные политики для различных зон. Их также называют zone-based firewall-ами. По умолчанию в firewalld уже прописано несколько зон:

```
sudo firewall-cmd --get-zones
```

И если вы внимательно смотрели, то заметили, что каждый раз, как мы смотрели файрвол, там красовалось слово public - это зона по умолчанию:

```
sudo firewall-cmd --get-default-zone
```

Политики public мы уже видели и настроили. Некоторые зоны уже преднастроены под определённые случаи, например, в зоне block всё по умолчанию блокируется, в зоне drop - дропается, а в зоне trusted - разрешается:

```
sudo firewall-cmd --list-all --zone=block | head -3
sudo firewall-cmd --list-all --zone=drop | head -3
sudo firewall-cmd --list-all --zone=trusted | head -3
```

The terminal session shows the configuration of the 'trusted' zone. It lists various parameters like target (ACCEPT), interfaces, and protocols. Then, it changes the interface to enp0s8 and moves it to the 'trusted' zone. Finally, it lists the interfaces again, showing enp0s8 in the 'trusted' zone. The session also includes pings between two hosts (192.168.56.3 and 192.168.31.5) and an SSH session.

```
[user@centos8 ~]$ sudo firewall-cmd --list-all --zone=trusted
trusted
target: ACCEPT
icmp-block-inversion: no
interfaces:
sources:
services:
ports:
protocols:
masquerade: no
forward-ports:
source-ports:
icmp-blocks:
rich rules:
[user@centos8 ~]$ sudo firewall-cmd --change-interface=enp0s8 --zone=trusted
success
[user@centos8 ~]$ sudo firewall-cmd --list-interfaces --zone=trusted
enp0s8
[user@centos8 ~]$ sudo firewall-cmd --list-interfaces --zone=public
enp0s3
[user@centos8 ~]$ [doctor@tardis]—[~]
$ ping 192.168.56.3
PING 192.168.56.3 (192.168.56.3) 56(84) bytes of data.
64 bytes from 192.168.56.3: icmp_seq=1 ttl=64 time=0.338 ms
^C
--- 192.168.56.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.338/0.338/0.338/0.000 ms
[doctor@tardis]—[~]
$ ssh user@192.168.56.3 -p 2233
Activate the web console with: systemctl enable --now cockpit.socket
Last login: Mon May 10 22:39:34 2021 from 192.168.56.1
[user@centos8 ~]$ logout
Connection to 192.168.56.3 closed.
[doctor@tardis]—[~]
$ ping 192.168.31.5
PING 192.168.31.5 (192.168.31.5) 56(84) bytes of data.
^C
--- 192.168.31.5 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3049ms
[x]—[doctor@tardis]—[~]
$
```

Так вот, мы можем назначать интерфейсы на зоны. Сейчас у нас оба интерфейса смотрят в зону public, но давайте переместим второй интерфейс в зону trusted:

```
sudo firewall-cmd --change-interface=enp0s8 --zone=trusted
```

Это будет временно, а как сделать на постоянно вы уже знаете. После смены интерфейс оказался в trusted зоне:

```
sudo firewall-cmd --list-interfaces --zone=trusted
```

Обратите внимание, что теперь к каждой команде я добавляю ключ `--zone` и указываю нужную зону. Потому что все прежние изменения мы делали с дефолтной - public. Если же мы хотим работать с другой зоной, нужно обязательно указывать её в опциях. После смены в дефолтной зоне остался только интерфейс enp0s3. И обратите внимание, так как мы сменили зону для интерфейса, то теперь по адресу на этом интерфейсе идёт пинг и можно подключаться. А адрес, оставшийся на enp0s3 всё ещё не пингуется.

```
[user@centos8 ~]$ sudo firewall-cmd --zone=block --add-source=192.168.56.1
[sudo] password for user:
success
[user@centos8 ~]$ sudo firewall-cmd --list-all --zone=block
block (active)
  target: %REJECT%
  icmp-block-inversion: no
  interfaces:
  sources: 192.168.56.1
  services:
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[user@centos8 ~]$ [x]-[doctor@tardis]---[-]
$ ip a show vboxnet0
8: vboxnet0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
  link/ether 0a:00:27:00:00:00 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.1/24 brd 192.168.56.255 scope global vboxnet0
      valid_lft forever preferred_lft forever
    inet6 fe80::27ff:fe00:0/64 scope link
      valid_lft forever preferred_lft forever
[doctor@tardis]---[~]
$ ping 192.168.56.3
PING 192.168.56.3 (192.168.56.3) 56(84) bytes of data.
From 192.168.56.3 icmp_seq=1 Packet filtered
From 192.168.56.3 icmp_seq=2 Packet filtered
From 192.168.56.3 icmp_seq=3 Packet filtered
^C
--- 192.168.56.3 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2040ms
[x]-[doctor@tardis]---[-]
$
```

Но иногда бывает, что каким-то определённым IP адресам мы хотим что-то разрешить или запретить, независимо от интерфейса. К примеру, вы хотите блокировать все входящие соединения на внешний адрес роутера, но у вас на работе есть внешний IP адрес, с которого вы хотите разрешить подключения. Интерфейс, который смотрит в интернет, вы переместили с зону drop, но вы можете определённые IP адреса или сети назначать в другие зоны. К примеру, сейчас у меня новый интерфейс находится в зоне trusted и только что всё пинговалось и подключалось. Давайте я добавлю IP адрес хоста в зону block:

```
sudo firewall-cmd --zone=block --add-source=192.168.56.1
sudo firewall-cmd --list-all --zone=block
```

Даже если у меня интерфейс находится в другой зоне, IP адрес конкретнее, а поэтому при виде запроса с этого IP адреса будет применяться политика из зоны block - т.е. пинги будут фильтроваться. Если же запрос приходит с адреса, который не прописан не в одной зоне как source - то срабатывает политика, которая применяется ко всему интерфейсу.

```
[user@centos8 ~]$ sudo firewall-cmd --zone=block --add-source=192.168.56.1
[sudo] password for user:
success
[user@centos8 ~]$ sudo firewall-cmd --list-all --zone=block
block (active)
  target: %REJECT%
  icmp-block-inversion: no
  interfaces:
  sources: 192.168.56.1
  services:
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[user@centos8 ~]$ [x]-[doctor@tardis]---[-]
$ ip a show vboxnet0
8: vboxnet0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
  link/ether 0a:00:27:00:00:00 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.1/24 brd 192.168.56.255 scope global vboxnet0
      valid_lft forever preferred_lft forever
    inet6 fe80::27ff:fe00:0/64 scope link
      valid_lft forever preferred_lft forever
[doctor@tardis]---[~]
$ ping 192.168.56.3
PING 192.168.56.3 (192.168.56.3) 56(84) bytes of data.
From 192.168.56.3 icmp_seq=1 Packet filtered
From 192.168.56.3 icmp_seq=2 Packet filtered
From 192.168.56.3 icmp_seq=3 Packet filtered
^C
--- 192.168.56.3 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2040ms
[x]-[doctor@tardis]---[-]
$
```

Но иногда бывает, что каким-то определённым IP адресам мы хотим что-то разрешить или запретить, независимо от интерфейса. К примеру, вы хотите блокировать все входящие соединения на внешний адрес роутера, но у вас на работе есть внешний IP адрес, с которого вы хотите разрешить подключения. Интерфейс, который смотрит в интернет, вы переместили с зону drop, но вы можете определённые IP адреса или сети назначать в другие зоны. К примеру, сейчас у меня новый интерфейс находится в зоне trusted и только что всё пинговалось и подключалось. Давайте я добавлю IP адрес хоста в зону block:

```
sudo firewall-cmd --zone=block --add-source=192.168.56.1
sudo firewall-cmd --list-all --zone=block
```

Даже если у меня интерфейс находится в другой зоне, IP адрес конкретнее, а поэтому при виде запроса с этого IP адреса будет применяться политика из зоны block - т.е. пинги будут фильтроваться. Если же

запрос приходит с адреса, который не прописан не в одной зоне как source - то срабатывает политика, которая применяется ко всему интерфейсу.

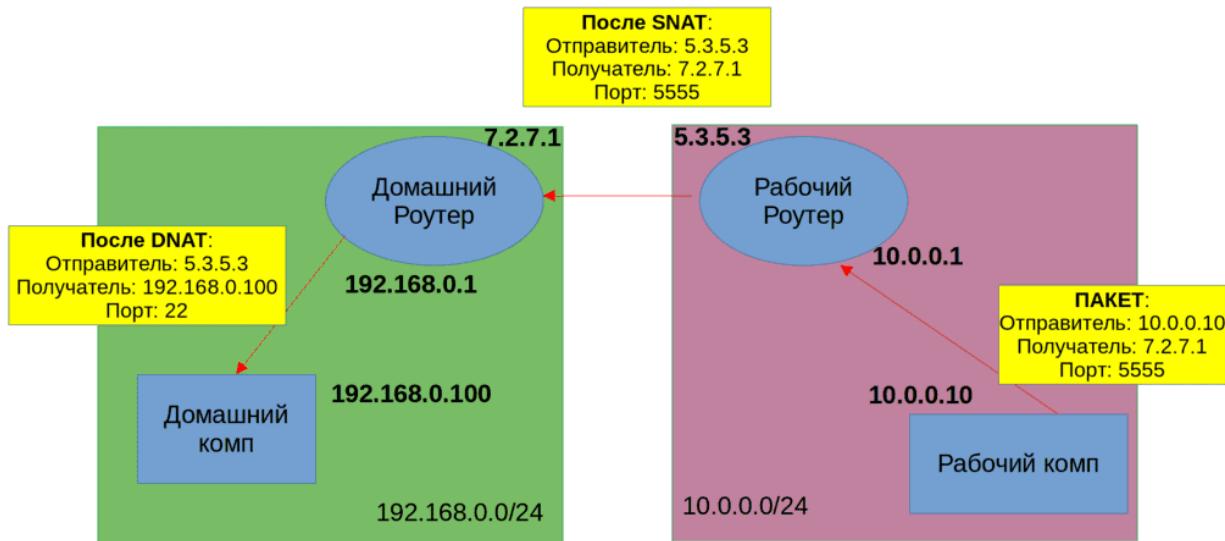
```
[user@centos8 ~]$ sudo firewall-cmd --list-all --zone=external
external
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: ssh
  ports:
  protocols:
  masquerade: yes
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[user@centos8 ~]$ sudo firewall-cmd --add-masquerade --zone=work
success
[user@centos8 ~]$
```

Ещё одна примечательная опция - masquerade:

```
sudo firewall-cmd --list-all --zone=external
```

Мы с вами обсуждали NAT - когда вы выходите в интернет, ваш роутер подменяет ваши домашние IP адреса на свой внешний, чтобы другая сторона знала, куда посыпать ответ. Так вот, такой тип NAT-а называется Source NAT, так как заменяется source адрес, т.е. адрес отправителя. Это также позволяет скрыть реальные IP адреса отправителей, поэтому также такой NAT называют **masquerade**. И так как Linux можно настроить как роутер, на нём также можно врубить source nat с помощью файрвола. По умолчанию он врублён в зоне external, но, по желанию, можно и других зонах:

```
sudo firewall-cmd --add-masquerade --zone=work
```



Чуть ниже masquerade-а есть forward-ports - это тоже про NAT, но уже про destination nat - когда заменяется адрес получателя. К примеру, ваш домашний компьютер недоступен из интернета, а ваш роутер - доступен. И вы можете на роутере выделить какой-нибудь порт, к примеру, 5555. И когда на роутер на этот адрес будут приходить запросы, он будет их пересыпать на ваш домашний компьютер. Таким образом вы сможете из интернета подключаться к домашнему компьютеру. В простонародье это называется проброс портов. Будет отдельная тема, где я нагляднее объясню Source и Destination NAT.

```
[user@centos8 ~]$ sudo firewall-cmd --list-all
public (active)
  target: DROP
  icmp-block-inversion: no
  interfaces: enp0s3
  sources:
  services: cockpit ssh
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[user@centos8 ~]$
```

Ну и наконец source-ports - исходящие порты. Тему портов мы уже затрагивали, однако про исходящие порты не говорили. И так, мы знаем, что ssh сервер по умолчанию работает на 22 порту. И когда мы к нему пытаемся подключиться, для нас этот порт является целевым - destination port. Но для отправки и получения ответа мы и сами, на стороне клиента, должны открыть порт - исходящий порт - source port. Но это не 22 порт, а динамический. Обычно это 5-значные порты, где-то от 30 до 60 тысяч. Когда вы клиентом подключаетесь к ssh, заходите через браузер на какой-то сайт и т.п. -

то операционная система временно выделяет для софта рандомный порт. Как только сессия закроется - порт перестанет использоваться. И, как правило, source порты рандомные и мало где указываются. Однако, в определённых программах можно выбрать исходящие порты. И вот на файрволе есть возможность запретить подключаться с других портов. Но это используется довольно редко.

```
[user@centos8 ~]$ ss -n4
NetidState Recv-Q Send-Q Local Address:Port    Peer Address:Port
tcp  ESTAB  0          192.168.31.5:2233  192.168.31.227:49994
[user@centos8 ~]$ ss -n4lt
State  Recv-Q Send-Q Local Address:Port    Peer Address:Port
LISTEN 0        128      0.0.0.0:2233      0.0.0.0:*
LISTEN 0        128      127.0.0.1:6010     0.0.0.0:*
LISTEN 0        128      0.0.0.0:5355      0.0.0.0:*
LISTEN 0        128      0.0.0.0:111       0.0.0.0:*
LISTEN 0        32       192.168.122.1:53    0.0.0.0:*
LISTEN 0        5        127.0.0.1:631     0.0.0.0:*
[user@centos8 ~]$ ss -a
Netid      State            Recv-Q      Send-Q           Local Address:Port
ort
nl        UNCONN           0          0               rtnl:a
vahi-daemon/957
nl        UNCONN           0          0               rtnl:1
595
nl        UNCONN           0          0               rtnl:N
etworkManager/1186
nl        UNCONN           0          0               rtnl:4
48791742
```

Ну и раз уж мы заговорили о портах, стоит упомянуть команду `ss`. Вкратце - эта команда показывает сокеты. Помните, мы говорили о сокетах, когда речь шла о логах? Сокеты позволяют процессам общаться между собой. Так вот, сетевые соединения также являются сокетами. Сокеты, которые нужны только между процессами внутри одной системы, называются UNIX сокетами. Ну и соответственно есть TCP и UDP сокеты. К примеру, запустим команду:

```
ss -n4
```

Здесь ключ `-n` - показывать номера портов, вместо имён. А ключ `-4` - показывать только сокеты, относящиеся к ipv4. Как видите, вышла одна строка - это tcp сокет, ESTAB - established - означает, что это установленное соединение. Оно установлено на локальный адрес - 192.168.31.5 на порт 2233. И это соединение идёт от адреса 192.168.31.227 - это мой компьютер, с source порта 49994. Это как раз текущее подключение по SSH.

`ss` позволяет понять, а какие порты на сервере сейчас активны. Часто бывает нужно понять - а слушает ли демон на таком-то порту? И тут можно использовать команду:

```
ss -n4lt
```

`l` - это listen, т.е. показывает активные порты, за которыми есть демоны. `t` - это tcp. В итоге мы видим список tcp сокетов с номерами портов, которые слушает наш сервер по ipv4. А если взять список всех сокетов, то их довольно много:

```
ss -a
```

Мы периодически будем пользоваться этой утилитой.

```
[x]-[doctor@tardis]—[~]
└─ $ nc -zv 192.168.31.5 22
^C
[x]-[doctor@tardis]—[~]
└─ $ nc -zv 192.168.31.5 2233
Connection to 192.168.31.5 port [tcp/infocrypt] succeeded!
[x]-[doctor@tardis]—[~]
└─ $ nc -zv 192.168.56.3 2233
nc: connect to 192.168.56.3 port 2233 (tcp) failed: No route to host
[x]-[doctor@tardis]—[~]
└─ $
```

Ещё одна важная утилита:

nc

По работе вы можете сталкиваться с различными проблемами и не всегда есть возможность точно сказать - проблема с самим приложением или файрвол блокирует? А возможно между пользователем и сервером есть другие файрволы? Если к ssh серверу или сайту можно подключиться утилитами, то не со всеми серверами так просто. Поэтому вам может понадобится понять, а есть ли доступ до сервера и не блокирует ли файрвол? Многие админы используют утилиту telnet, но она не всегда предустановлена, да и не самая удобная. На большинстве Linux-ов зачастую предустановлена утилита nc, и обычно я использую её. Синтаксис для проверки порта простой - nc -zv ip адрес порт:

```
nc -zv 192.168.31.5 22
nc -zv 192.168.31.5 2233
nc -zv 192.168.56.3 22
```

Как видите, в первом случае я долго жду ответа, но его нет - это значит, что либо нет демона за этим адресом и портом, либо файрвол блокирует запрос. Во втором случае я получил succeeded - т.е. соединение успешно, nc смог подключиться. В третьем случае соединение не прошло - значит, либо хост недоступен, либо файрвол. Но так можно проверять можно только TCP порты, UDP работает несколько иначе, и проверить его бывает затруднительно.

<pre>[user@centos8 ~]\$ sudo nc -ul 5555 ^C [user@centos8 ~]\$ sudo firewall-cmd --add-port=5555/udp success [user@centos8 ~]\$ sudo nc -ul 5555 Hey Сейчас работает, теперь попробуем tcp ^C [user@centos8 ~]\$ sudo nc -tl 5555 ^C [user@centos8 ~]\$ sudo firewall-cmd --add-port=5555/tcp success [user@centos8 ~]\$ sudo nc -tl 5555 Hello Так Заработало</pre>	<pre>[x]-[doctor@tardis]—[~] └─ \$ nc -u 192.168.31.5 5555 Hello Hey Сейчас работает, теперь попробуем tcp ^C [x]-[doctor@tardis]—[~] └─ \$ nc -t 192.168.31.5 5555 Hello Так Заработало</pre>
--	--

nc также позволяет передавать сообщения, используя tcp или udp. Если у вас есть доступ на сервер, вы можете понять, а работают ли соединения, не блокирует ли кто-то посередине и в целом заранее убедиться, что с сетью проблем не будет, прежде чем настроить какой-то сервис. К примеру, потестируем udp соединение, на стороне сервера слушаем на udp порту 5555:

```
sudo nc -ul 5555
```

С клиента попытаемся подключиться:

```
nc -u 192.168.31.5 5555
```

Как видите, соединение не проходит. Я на файрволе не открывал порт, в этом и причина. Добавим порт:

```
sudo firewall-cmd --add-port=5555/udp
```

после чего опять потестим:

```
sudo nc -ul 5555
```

Теперь я могу что-то написать и это придет на сервер. Для проверки tcp просто заменим ключ -u на -t - и на стороне клиента, и на стороне сервера. Всё довольно просто.

```
[user@centos8 ~]$ sudo firewall-cmd --runtime-to-permanent  
[sudo] password for user:  
success  
[user@centos8 ~]$ sudo firewall-cmd --list-all-zones  
block (active)  
    target: %%REJECT%%  
    icmp-block-inversion: no  
    interfaces:  
    sources: 192.168.56.1  
    services:
```

Возвращаясь к файрволу. Мы сделали много изменений, но не прописывали permanent - а это значит, что после рестарта все наши изменения слетят. Если же мы хотим сохранить настройки, допустим, мы тестировали и всё нас устроило - то можем использовать ключ --runtime-to-permanent:

```
sudo firewall-cmd --runtime-to-permanent
```

А для просмотра информации по всем зонам - --list-all-zones:

```
sudo firewall-cmd --list-all-zones
```

```
[user@centos8 ~]$ sudo firewall-cmd --panic-on ^C  
[user@centos8 ~]$ sudo firewall-cmd --panic-off ^C  
[user@centos8 ~]$ █
```

Ещё из примечательного - режим паники --panic-on:

```
sudo firewall-cmd --panic-on  
sudo firewall-cmd --panic-off
```

При его запуске обрываются все соединения и дропаются все входящие и исходящие пакеты. Может пригодиться, скажем, если вас взломали и вам срочно нужно оборвать всё.

```
[user@centos8 ~]$ sudo firewall-cmd --list-all
public (active)
  target: DROP
  icmp-block-inversion: no
  interfaces: enp0s3
  sources:
    services: cockpit ssh
    ports: 5555/udp 5555/tcp
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[user@centos8 ~]$ man firewalld.
firewalld.conf          firewalld.helper      firewalld.lockdown-whitelist  firewalld.zone
firewalld.dbus           firewalld.icmptype   firewalld.richlanguage       firewalld.zones
firewalld.direct         firewalld.ipset     firewalld.service
[user@centos8 ~]$ man firewalld.richlanguage
```

Ну и под конец - **--rich-rules**. До этого мы говорили про зоны, что можно разделить в файерволе сеть по областям, различая по интерфейсам и IP адресам, и к разным применять различные политики. Но иногда бывают нужны точечные правила - только этому адресу разрешить только этот порт и при этом логировать. В некоторых случаях можно создать зону и прописать один source и один сервис, будет чем-то похоже на точечное правило, но это выглядит коряво, всё таки это не зона. Для этого лучше использовать rich-rules. Правда есть нюанс. Скажем, если посмотреть:

```
firewall-cmd --list-all
```

то по выведенному списку очень легко будет понять синтаксис для команд - add-service, add-port, remove-source, change-interface и всё в таком духе. Можно просто два раза tab нажать и увидеть список почти всех доступных команд, а там интуитивно понятно. А с rich rule-ами так не работает, табуляция не поможет, да и не так интуитивно понятно. Но выход простой - в манах есть хорошие примеры:

```
man firewalld.richlanguage
```

```
General rule structure

rule
  [source]
  [destination]
  service|port|protocol|icmp-block|icmp-type|masquerade|forward-port|source-port
  [log]
  [audit]
  [accept|reject|drop|mark]
```

В таких правилах можно указать source и destination адрес, сервис/порт/протокол, а также логирование, аудит и применяемое в итоге действие - пропустить, блокнуть, дропнуть или добавить метку. Метка штука специфичная и пока нам не нужна, а аудит - это тоже своего рода логирование, эту тему мы когда-нибудь разберём.

EXAMPLES

These are examples of how to specify rich language rules. This format (i.e. one string that specifies whole rule) uses for example `firewall-cmd --add-rich-rule` (see `firewall-cmd(1)`) as well as D-Bus interface.

Example 1

```
Enable new IPv4 and IPv6 connections for protocol 'ah'

rule protocol value="ah" accept
```

Example 2

```
Allow new IPv4 and IPv6 connections for service ftp and log 1 per minute using audit

rule service name="ftp" log limit value="1/m" audit accept
```

Example 3

```
Allow new IPv4 connections from address 192.168.0.0/24 for service tftp and log 1 per minutes using syslog
```

```
rule family="ipv4" source address="192.168.0.0/24" service name="tftp" log prefix="tftp" level="info" limit value="1/m"
accept
```

Спустимся чуть ниже, где у нас примеры - EXAMPLES. Первый пример довольно простой - разрешать протокол ah. Обратите внимание, что в каждом правиле в начале есть слово rule. Второй пример - разрешить сервис с именем ftp, также его логировать, при этом не чаще раза в минуту, об этом говорит limit value=>1/m», ну и включён audit. Третье правило - разрешать соединения по ipv4 от сети 192.168.0.0/24 к сервису tftp, при этом логировать не чаще раза в минуту, и к логам приписывать префикс tftp, а также важность - info. Если вы помните, при разборе логирования мы упоминали, что в syslog указывается важность лога, это может быть просто информационное сообщение или какая-то ошибка. Там есть ещё примеры, но их постараитесь разобрать сами.

<pre>[user@centos8 ~]\$ sudo firewall-cmd --add-rich-rule='rule family="ipv4" source address="192.168.31.227" destination address="192.168.31.6" port port="2233" protocol="tcp" log prefix="MYSSH" level="crit" limit value="1/m" accept' success [user@centos8 ~]\$ sudo firewall-cmd --list-all public (active) target: DROP icmp-block-inversion: no interfaces: enp0s3 sources: services: cockpit ssh ports: 5555/udp 5555/tcp protocols: masquerade: no forward-ports: source-ports: icmp-blocks: rich rules: rule family="ipv4" source address="192.168.31.227" destination address="192.168.31.6" port port="2233" protocol="tcp" log prefix="MYSSH" level="crit" limit value="1/m" accept [user@centos8 ~]\$</pre>	<pre>[doctor@tardis:~]—[~] \$ ssh user@192.168.31.6 -p 2233 Activate the web console with: systemctl enable --now cockpit.socket Last login: Tue May 11 10:27:44 2021 from 192.168.31.227 [user@centos8 ~]\$ sudo journalctl -p crit tail -1 [sudo] password for user: May 11 10:30:22 centos8 kernel: MYSSHIN=enp0s3 OUT= MAC=08:00:27:ce:55:bb:fe:c2:5f:0c:54:38:08:00 SRC=192.168.31.227 DST=192.168.31.6 LEN=60 TOS=0x08 PREC=0x40 TTL=64 ID=56198 DF PROTO=TCP SPT=54822 DPT=2233 WINDOW=64240 RES=0x00 SYN URGP=0 [user@centos8 ~]\$ sudo firewall-cmd --remove-rich-rule='rule family="ipv4" source address="192.168.31.227" destination address="192.168.31.6" port port="2233" protocol="tcp" log prefix="MYSSH" level="crit" limit value="1/m" accept' success [user@centos8 ~]\$</pre>
---	--

Давайте пропишем правило. Например, разрешать подключение ко второму IP адресу сервера только от адреса моего хоста по порту 2233 и логировать эту информацию не чаще раза в минуту с важностью crit, добавляя prefix MYSSH. Для этого используем опцию `--add-rich-rule`, не забываем указать в начале rule, ну и указываем интересующие нас опции, для начала, что это ipv4, source и destination адреса, порт 2233 и его протокол - tcp, префикс и прочую информацию по логам:

```
sudo firewall-cmd --add-rich-rule='rule family="ipv4" source address="192.168.31.227" destination address="192.168.31.6" port port="2233" protocol="tcp" log prefix="MYSSH" level="crit" limit value="1/m" accept'
```

После этого правило появится в зоне, так как мы не указали зону в команде, то оно добавится в дефолтной зоне:

```
sudo firewall-cmd --list-all
```

Ну и протестируем - подключимся к этому адресу и проверим последние логи с уровнем crit:

```
ssh user@192.168.31.6 -p 2233
sudo journalctl -p crit | tail -1
```

Лог появился и в нём указан входящий интерфейс, ip адреса и прочая информация. Да, rich правила не самые удобные в написании, но с map-ом можно легко вспомнить. А для удаления таких правил опция `--remove-rich-rule` со всем содержимым правила.

Давайте подведём итоги. Сегодня мы с вами разобрали firewalld - инструмент, который позволяет гибко и удобно настроить правила в рамках одной или нескольких зон. Синтаксис у команды простой и интуитивно понятный - можно добавлять и убирать порты, сервисы, протоколы, делить сеть на зоны, меняя принадлежность интерфейсов и адресов к какой-либо зоне. Поговорили про ICMP, reject и drop пакетов, а также немного про Source и Destination NAT. Посмотрели утилиту ss, которая выводит информацию о сокетах, а также тестировали соединение с помощью nc. Ну и под конец rich rule-ы - точечные правила, которые позволяют добавлять исключения. И помните, если даже вы забыли синтаксис - всегда есть маны, которые позволят вам вспомнить.

2.47 47. Пакетный менеджер - dnf

2.47.1 47. Пакетный менеджер - dnf

```
[doctor@tardis]——[~]
└─$ ssh vbox
Last login: Fri May 28 12:02:00 2021 from 192.168.31.227
^[[A[user@centos8 sudo dnf install^C
[user@centos8 ~]$ echo $PATH
/home/user/.local/bin:/home/user/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin
[user@centos8 ~]$ which ssh
/usr/bin/ssh
[user@centos8 ~]$ ls /etc/ssh/
moduli      ssh_config.d  ssh_host_ecdsa_key    ssh_host_ed25519_key    ssh_host_rsa_key
ssh_config  sshd_config   ssh_host_ecdsa_key.pub  ssh_host_ed25519_key.pub  ssh_host_rsa_key.pub
[user@centos8 ~]$ ldd /usr/bin/ssh
    linux-vdso.so.1 (0x00007ffeab523000)
    libfipscheck.so.1 => /lib64/libfipscheck.so.1 (0x00007f7afbc89000)
    libcrypto.so.1.1 => /lib64/libcrypto.so.1.1 (0x00007f7afb7aa000)
    libdl.so.2 => /lib64/libdl.so.2 (0x00007f7afb5a6000)
    libutil.so.1 => /lib64/libutil.so.1 (0x00007f7afb3a2000)
    libz.so.1 => /lib64/libz.so.1 (0x00007f7afb18b000)
    libcrypt.so.1 => /lib64/libcrypt.so.1 (0x00007f7fafaf62000)
    libresolv.so.2 => /lib64/libresolv.so.2 (0x00007f7afad4b000)
```

Мы с вами не раз устанавливали программы с помощью команды:

```
dnf install ...
```

но пора детальнее разобрать эту тему. Для начала вспомним, что мы вообще знаем о программах? Чаше всего под программой мы подразумеваем запускаемый файл. И мы знаем, что в переменной \$PATH перечислены директории, в которых bash ищет эти запускаемые программы. Для примера возьмём ssh. С помощью команды:

```
which ssh
```

мы можем узнать, где именно находится запускаемый файл - `/usr/bin/ssh`. Однако одним этим файлом программа не ограничивается, у неё есть настройки в директории `/etc/ssh`:

```
ls /etc/ssh
```

Ещё есть библиотеки, при помощи которых работает эта программа, их мы можем узнать с помощью команды ldd:

```
ldd /usr/bin/ssh
```

Также у программ есть документация, которая лежит в /usr/share/doc, файл сервиса, если это демон, и много других файлов, которые лежат в разных директориях по всей системе.

Исходя из вышесказанного, при установке мы должны расположить все эти файлы по системе. Лучший способ распространять много файлов - в архиве. При этом, если я захочу удалить программу, мне же нужно будет все эти файлы поудалять? А значит надо вести учёт, что и куда было распаковано. Ну и помните, мы говорили, что есть сервисные пользователи - сервисы, по хорошему, должны работать не от рута, а от специальных пользователей. Т.е. если мы ставим веб сервер, то он должен работать от сервисного пользователя. А пользователи - это не файлы, их нужно создавать с помощью команды useradd. А значит не всегда достаточно просто распаковать архив, иногда бывает нужно ещё и запустить команду или скрипт, чтобы, например, добавить сервисного пользователя. Т.е. программы распространяются в виде архивов, но у этих архивов ещё могут быть скрипты и всякая информация о самом архиве. И такие архивы называют пакетами.

```
[user@centos8 test]$ sudo dnf install lvm2 ^C
[user@centos8 test]$ sudo pv
pvchange  pvcreate  pvmove    pvresize   pvscan
pvck      pvdisplay  pvremove  pvs
[user@centos8 test]$ sudo vg
vgcfgbackup  vgconvert   vgextend    vgmknodes   vgs
vgcfgrestore  vgcreate    vgimport   vgreduce    vgscan
vgchange     vgdisplay   vgimportclone  vgremove   vgsplit
vgck        vgexport    vgmerge    vgrename
[user@centos8 test]$ sudo lv
lvchange    lvdisplay   lvmconfig   lvmpolld    lvreduce   lvresize
lvconvert   lvextend    lvmdiskscan  lvmsadc    lvremove   lvs
lvcreate    lvm        lvmdump     lvmsar     lvrename   lvscan
[user@centos8 test]$ sudo lv■
```

Когда говорят про установку программ на Linux-ах, речь идёт о пакетах - мы устанавливаем пакеты. При этом внутри одного пакета может быть сразу несколько программ, например, ставим один пакет lvm2 - а внутри pvcreate, vgcreate, lvdisplay и куча других запускаемых файлов, но все они относятся к программе LVM. При этом можно сделать пакет, внутри которого не будет исполняемых файлов, только конфиг файлы и скрипты - например, так можно распространять настройки. То есть не каждый пакет является программой.



Для работы с пакетами есть программы, называемые пакетными менеджерами. Они устанавливают пакеты, удаляют их, обновляют, ведут учёт, выводят информацию - т.е. управляют пакетами. Пакетных менеджеров много, на разных дистрибутивах могут быть разные. К примеру, на CentOS и многих RHEL-based дистрибутивах используются rpm, yum, dnf, а на Debian-based дистрибутивах, например, на Ubuntu, используется dpkg, apt, apt-get, aptitude. К тому же у самих пакетов могут быть различные форматы, они по разному собираются. На Debian-based дистрибутивах формат .deb, а на RHEL-based - .rpm.

```
[user@centos8 ~]$ mkdir test
[user@centos8 ~]$ cd test/
[user@centos8 test]$ dnf download lvm2
Last metadata expiration check: 1:44:16 ago on Fri 28 May 2021 12:27:53 PM +04.
lvm2-2.03.09-5.el8.x86_64.rpm                                1.7 MB/s | 1.6 MB      00:00
[user@centos8 test]$ ls
lvm2-2.03.09-5.el8.x86_64.rpm
[user@centos8 test]$
```

Для примера скачаем какой-нибудь пакет. Это можно сделать с помощью dnf download:

```
mkdir test
cd test
dnf download lvm2
ls
```

У нас скачался файл с расширением .rpm. В названии пакета указано название программы, её версия, дистрибутив, под который сделана - el8 - это CentOS 8, дальше архитектура, под которую собран пакет - в нашем случае под 64-битные процессоры, ну и расширение rpm.

```
[user@centos8 test]$ rpm2cpio lvm2-2.03.09-5.el8.x86_64.rpm | cpio -idmv
./etc/lvm
./etc/lvm/archive
./etc/lvm/backup
./etc/lvm/cache
```

Как мы говорили, пакеты - это архивы, а значит их можно распаковать. Мы с вами изучали архиватор tar, но rpm пакеты архивируются другой утилитой - cpio. Воспользуемся ей, чтобы распаковать этот архив:

```
rpm2cpio lvm2-2.03.09-5.el8.x86_64.rpm | cpio -idmv
```

```
[user@centos8 test]$ ls
etc  lvm2-2.03.09-5.el8.x86_64.rpm  run  usr
[user@centos8 test]$ ls etc/
lvm
[user@centos8 test]$ ls run/
lock  lvm
[user@centos8 test]$ ls usr/
lib  sbin  share
[user@centos8 test]$ ls usr/sbin/
fsadm    lvm      lvmsar  lvscan   pvremove   vgchange  vgextend  vgremove
lvchange  lvmconfig lvreduce  pvchange  pvresize   vgck     vgimport  vgrename
lvconvert lvmdiskscan lvremove  pvck      pvs       vgconvert  vgimportclone vgs
lvcreate  lvmdump   lvrename  pvcreate  pvscan    vgcreate  vgmerge   vgscan
lvdisplay lvmpolld  lvresize  pvdisplay vgcfgbackup vgdisplay vgmknodes  vgsplit
lvextend  lvmsadc   lvs      pvmove   vgcfgrestore vgexport  vgreduce
[user@centos8 test]$ ■
```

Из архива у нас вышли 3 директории - etc, run и usr. Ничего не напоминает? Правильно, это директории из корня. И когда мы устанавливаем пакет, он разархивируется в корень - в /etc копируется директория /etc/lvm, в /run копируется директория /run/lvm, а в /usr/sbin те самые файлы команд.

```
[user@centos8 test]$ ls -l usr/sbin/
total 2768
-rwxr-xr-x. 1 user user 24108 Aug 18 2020 fsadm
lrwxrwxrwx. 1 user user      3 May 28 14:28 lvchange -> lvm
lrwxrwxrwx. 1 user user      3 May 28 14:28 lvconvert -> lvm
lrwxrwxrwx. 1 user user      3 May 28 14:28 lvcreate -> lvm
lrwxrwxrwx. 1 user user      3 May 28 14:28 lvdisplay -> lvm
lrwxrwxrwx. 1 user user      3 May 28 14:28 lvextend -> lvm
-rwxr-xr-x. 1 user user 2586048 Aug 18 2020 lvm
lrwxrwxrwx. 1 user user      3 May 28 14:28 lvmconfig -> lvm
lrwxrwxrwx. 1 user user      3 May 28 14:28 lvmdiskscan -> lvm
-rwxr-xr-x. 1 user user 10312 Aug 18 2020 lvmdump
-rwxr-xr-x. 1 user user 206128 Aug 18 2020 lvmpolld
lrwxrwxrwx. 1 user user      3 May 28 14:28 lvmsadc -> lvm
```

Причём можно заметить, что большинство файлов этого пакета в usr/bin - это просто символические ссылки, ведущие на одну программу.

```
[user@centos8 test]$ rpm -qp --scripts lvm2-2.03.09-5.el8.x86_64.rpm
postinstall scriptlet (using /bin/sh):

if [ $1 -eq 1 ] ; then
    # Initial installation
    systemctl --no-reload preset blk-availability.service lvm2-monitor.service &>/dev/null ||
|
fi

if [ "$1" = "1" ] ; then
    # FIXME: what to do with this? We do not want to start it in a container/chroot
    # enable and start lvm2-monitor.service on completely new installation only, not on upgr
ades
    systemctl enable lvm2-monitor.service
    systemctl start lvm2-monitor.service >/dev/null 2>&1 ||
|
fi
```

Мы ещё говорили, что кроме архива в пакете также есть скрипты. В .deb пакетах внутри самого пакета находятся 2 архива - в одном скрипты и информация о пакете, а в другом архив с файлами. Что касается .rpm пакетов, то тут информация и скрипты вшиты в сам пакет, и чтобы их увидеть, надо воспользоваться командой rpm. Например, чтобы увидеть скрипты, надо выполнить команду:

```
rpm -qp --scripts lvm2*
```

На первой строчке мы видим postinstall scriptlet - этот скрипт выполняется после установки пакета.

```
preuninstall scriptlet (using /bin/sh):

if [ $1 -eq 0 ] ; then
    # Package removal, not upgrade
    systemctl --no-reload disable --now blk-availability.service lvm2-monitor.service &>/dev/
>null || :
fi

if [ $1 -eq 0 ] ; then
    # Package removal, not upgrade
    systemctl --no-reload disable --now lvm2-lvmpolld.service lvm2-lvmpolld.socket &>/dev/n
ull || :
fi
postuninstall scriptlet (using /bin/sh):

if [ $1 -ge 1 ] ; then
    # Package upgrade, not uninstall
    systemctl try-restart lvm2-lvmpolld.service &>/dev/null || :
fi
```

Чуть ниже также можно увидеть preuninstall scriptlet, а потом ещё один postuninstall - т.е. скриптов может быть несколько, какие-то выполняются до установки пакета, какие-то после. Причём, если обратить внимание, нижние скрипты вообще выполняются при удалении пакета, например, чтобы отключить лишние сервисы.

```
[user@centos8 test]$ rpm -qip lvm2-2.03.09-5.el8.x86_64.rpm
Name        : lvm2
Epoch       : 8
Version     : 2.03.09
Release     : 5.el8
Architecture: x86_64
Install Date: (not installed)
Group       : Unspecified
Size        : 3675019
License     : GPLv2
Signature   : RSA/SHA256, Tue 18 Aug 2020 12:44:13 AM +04, Key ID 05b555b38483c65d
Source RPM  : lvm2-2.03.09-5.el8.src.rpm
Build Date  : Tue 18 Aug 2020 12:23:17 AM +04
Build Host  : x86-02.mbox.centos.org
Relocations : (not relocatable)
Packager    : CentOS Buildsys <bugs@centos.org>
Vendor      : CentOS
URL         : http://sourceware.org/lvm2
Summary     : Userland logical volume management tools
Description :
LVM2 includes all of the support for handling read/write operations on
physical volumes (hard disks, RAID-Systems, magneto optical, etc.,
```

Ну и помимо скриптов пакет содержит информацию о себе, которую также можно увидеть с помощью утилиты rpm:

```
rpm -qip lvm2*
```

Наверху у нас та информация, которую мы видели в названии пакета - имя, версия, релиз и архитектура. Также тут указан Epoch - это тоже касается версии. Просто иногда разработчики софта могут менять нумерацию версии, либо вести её каким-то нестандартным образом. Тогда пакетному менеджеру бывает сложно понять, а какая версия пакета новее, какая старее. И вот epoch номер спасает в таких ситуациях.

Также в информации о пакете указан его размер, лицензия, время сборки, кто собирал пакет, ссылка, небольшое описание и прочая подобная информация.

```
[user@centos8 test]$ which yum
/usr/bin/yum
[user@centos8 test]$ ls -l /usr/bin/yum
lrwxrwxrwx. 1 root root 5 Dec 19 2019 /usr/bin/yum -> dnf-3
[user@centos8 test]$
```

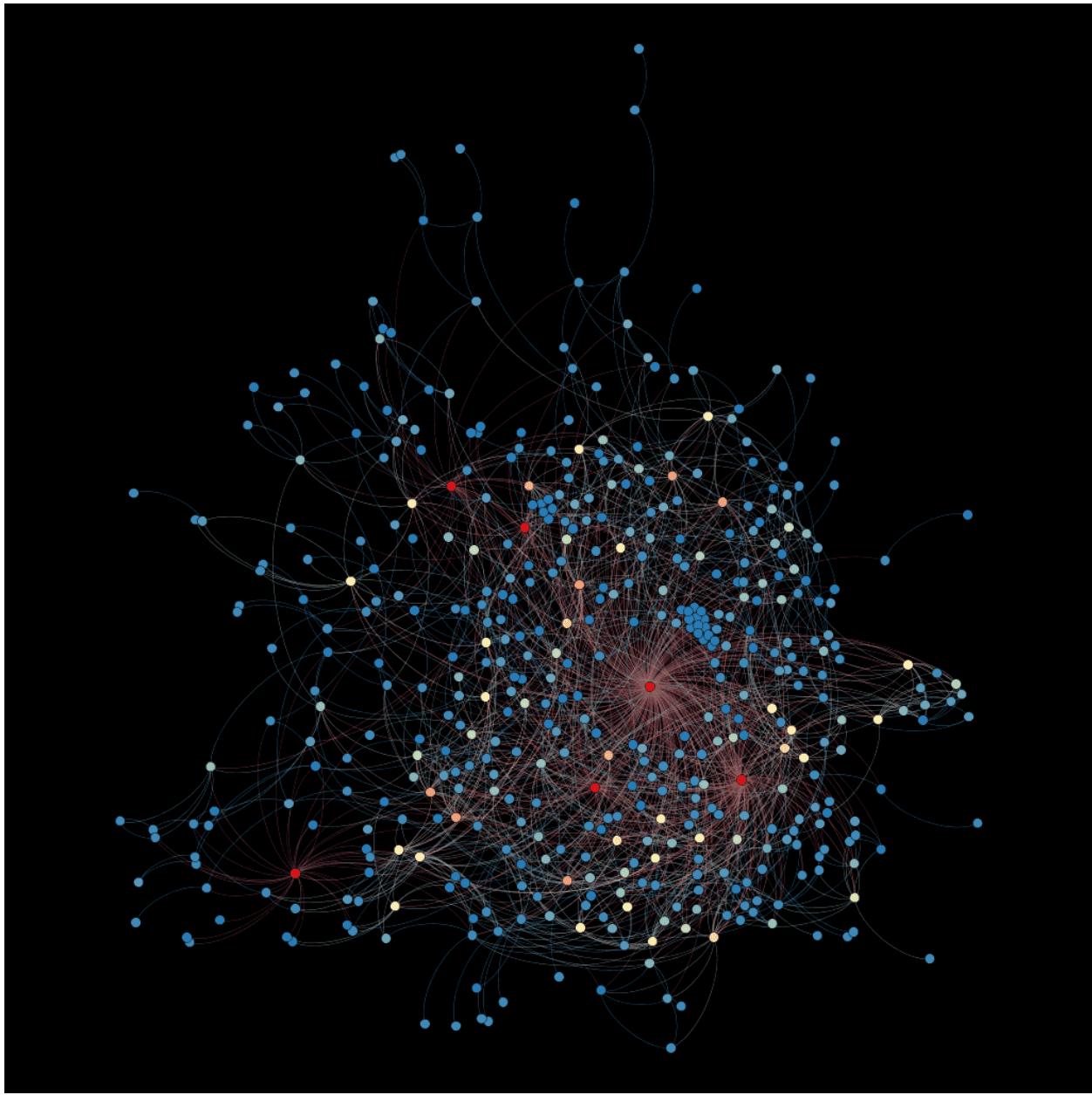
Хорошо, с тем, что из себя представляет пакет, мы разобрались. Поговорим о пакетных менеджерах. На Centos есть 2 основных пакетных менеджера - rpm и dnf. Часто в интернете вы можете натыкаться на команду yum - это также пакетный менеджер, но он использовался в предыдущих версиях Centos. Его переработали и выпустили новую реализацию - dnf. А yum остался как символическая ссылка на dnf, чтобы облегчить переход пользователей и не сломать инструменты и скрипты, работающие с yum:

```
which yum
ls -l /usr/bin/yum
```

Синтаксис во многом совпадает, поэтому на старых системах вы можете вместо dnf писать yum.

	 Free software	 Open-source software	 Freeware	 Public-domain software
Definition	"FREE" is a matter of liberty, not price	"OPEN" doesn't just mean access to the source code	"FREE" refers to price, while freedom of the use is restricted by creator	"PUBLIC DOMAIN" belongs to the public as a whole
Ground philosophy	Social movement	Development methodology	Marketing goals	Copyright disclaimation
Ground rules	Four Freedoms https://www.gnu.org/philosophy/free-sw.html	Open Software initiative https://opensource.org/osd		Creative Common Organization https://creativecommons.org
Free of charge	Not necessary	Not necessary	✓ YES	✓ YES
Covered by copyright law	✓ YES	✓ YES	✓ YES	✗ NO
Examples	   	 		

Чтобы увидеть разницу между rpm и dnf следует понять особенность разработки софта на GNU/Linux. Большая часть ПО под Linux системы сделана с открытыми и свободными лицензиями. Эти лицензии предполагают, что любой желающий может использовать, изменять и распространять программы, не спрашивая ни у кого разрешение, минуя всякую бюрократию. И когда разные разработчики пишут программы, им необязательно изобретать велосипед - они могут взять кусочек чужой программы. Обычно это библиотеки, а иногда и целые программы.



В итоге сотни программ могут использовать одни и те же библиотеки, одна программа может использовать другую, а другая третью и всё такое. Если всё встраивать в саму программу, то программа будет весить много и потреблять много ресурсов. А если делать общие библиотеки между программами, всё делить - то не придётся один и тот же код использовать в разных программах, в итоге программы занимают меньше места. Но при этом появляется понятие зависимостей: в самой программе может не хватать чего-то для работы, и ей для этого нужна другая программа или библиотека. Одни программы начинают зависеть от других. Если вы удалите одну библиотеку, могут перестать работать сотни программ. Или, скажем, у вас установлена библиотека версии 3, и вы пытаетесь установить программу, которая требует ту же библиотеку, но версии 2. Если вы замените новую библиотеку на старую, то могут перестать работать другие программы.

Т.е. пакеты зависят друг от друга, и зачастую недостаточно установить один пакет, нужно ставить несколько. И чтобы не ходить по разным сайтам в поисках их зависимостей и не скачивать пакеты по одному, разработчики дистрибутивов делают специальные сервера, где лежат десятки тысяч пакетов для их дистрибутива. Эти сервера называются репозиториями. Тем более лицензия позволяет

распространять софт любому желающему. И вот среди разработчиков дистрибутива есть люди, которые постоянно проверяют определённый софт на новые версии, зависимости, проверяют стабильность и безопасность, после чего делают пакет с этим софтом и добавляют его в репозиторий. Такие люди называются мейнтейнерами.

```
[user@centos8 test]$ dnf deplist lvm2
Last metadata expiration check: 3:50:27 ago on Fri 28 May 2021 12:27:53 PM +04.
package: lvm2-8:2.03.09-5.el8.x86_64
dependency: /bin/bash
  provider: bash-4.4.19-12.el8.x86_64
dependency: /bin/sh
  provider: bash-4.4.19-12.el8.x86_64
dependency: bash >= 4.0
  provider: bash-4.4.19-12.el8.x86_64
dependency: device-mapper-persistent-data >= 0.7.0-0.1.rc6
  provider: device-mapper-persistent-data-0.8.5-4.el8.x86_64
dependency: libaio.so.1()(64bit)
  provider: libaio-0.3.112-1.el8.x86_64
dependency: libaio.so.1(LIBAIO_0.1)(64bit)
  provider: libaio-0.3.112-1.el8.x86_64
dependency: libaio.so.1(LIBAIO_0.4)(64bit)
  provider: libaio-0.3.112-1.el8.x86_64
```

И они следят за тем, чтобы в рамках одной версии дистрибутива не было проблем. Зачастую бывает невозможно установить разные версии одной и той же библиотеки, а значит нужно выбрать определённую версию для дистрибутива, основываясь на стабильности, безопасности и том, что большинство софта требует именно эту версию. Список зависимостей можно посмотреть с помощью команды `dnf deplist`:

```
dnf deplist lvm2
```

Здесь мы видим, какие программы и библиотеки требуется для `lvm2` и в каких пакетах они есть.

```
[user@centos8 test]$ sudo dnf install lvm2
[sudo] password for user:
CentOS-8 - AppStream
CentOS-8 - Base
CentOS-8 - Extras
Extra Packages for Enterprise Linux Modular 8 - x86_64
Extra Packages for Enterprise Linux 8 - x86_64
Package lvm2-8:2.03.05-5.el8.0.1.x86_64 is already installed.
Dependencies resolved.

=====
 Package          Architecture Version      Repository   Size
=====
Upgrading:
device-mapper      x86_64       8:1.02.171-5.el8  BaseOS      373 k
device-mapper-event x86_64       8:1.02.171-5.el8  BaseOS      268 k
device-mapper-event-libs x86_64     8:1.02.171-5.el8  BaseOS      267 k
device-mapper-libs  x86_64       8:1.02.171-5.el8  BaseOS      406 k
lvm2               x86_64       8:2.03.09-5.el8   BaseOS      1.6 M
lvm2-libs          x86_64       8:2.03.09-5.el8   BaseOS      1.1 M

Transaction Summary
=====
Upgrade 6 Packages

Total download size: 4.0 M
Is this ok [y/N]:
```

Но когда вы устанавливаете программу:

```
sudo dnf install lvm2
```

то вам не нужно беспокоиться о зависимостях, этим занимается пакетный менеджер. Он видит зависимости каждого пакета, он находит подходящие пакеты в репозитории и предлагает их установить. Кроме того, в данной ситуации dnf сказал, что этот пакет уже установлен, но его может обновить - и предлагает обновить его зависимости в том числе.

Если обобщить, нередко для работы одной программы нужна другая программа или библиотека. Есть сервера, называемые репозиториями, на этих серверах лежат пакеты. Пакетный менеджер может найти в репозитории нужный пакет и его зависимости, т.е. другие пакеты, и установить всё вместе.

```
[user@centos8 test]$ rpm -i lvm2-2.03.09-5.el8.x86_64.rpm
error: Failed dependencies:
lvm2-libs = 8:2.03.09-5.el8 is needed by lvm2-8:2.03.09-5.el8.x86_64
[user@centos8 test]$ ls /var/lib/rpm/
Basenames      Dirnames      Name      Requirename      Transfiletriggername
Conflictname   Enhancename   Obsoletename Shalheader      Triggername
  db.001        Filetriggername Packages   Sigmd5
  db.002        Group        Providename Suggestname
  db.003        Installtid   Recommendname Supplementname
[user@centos8 test]$ █
```

Но не каждый пакетный менеджер работает с репозиториями. dnf это умеет, а rpm нет. При этом rpm это главная утилита для работы с пакетами .rpm - она позволяет создавать пакеты, изменять и всё такое, т.е. это низкоуровневая утилита. И хотя она умеет устанавливать пакеты:

```
rpm -i lvm2-*
```

в большинстве случаев не стоит так делать. rpm может знать о зависимостях, но не может их решить - с rpm вам нужно вручную разбираться с ними. Т.е. зачастую лучше пользоваться dnf, но rpm это не бесполезная утилита - она предоставляет библиотеки для работы dnf, а также базу, где учитываются установленные пакеты и вся информация о них:

```
ls /var/lib/rpm
```

```
[user@centos8 test]$ rpm -ql lvm2
/etc/lvm
/etc/lvm/archive
/etc/lvm/backup
/etc/lvm/cache
/etc/lvm/cache/.cache
/etc/lvm/lvm.conf
/etc/lvm/lvmlocal.conf
/etc/lvm/profile
/etc/lvm/profile/cache-mq.profile
```

В базе rpm как раз хранится информация, а какие именно файлы и зависимости были установлены вместе с пакетом:

```
rpm -ql lvm2
```

```
[user@centos8 test]$ rpm -qa | wc -l
1620
[user@centos8 test]$ rpm -qa
ghc-srpm-macros-1.4.2-7.el8.noarch
libcacard-2.7.0-2.el8_1.x86_64
perl-File-Temp-0.230.600-1.el8.noarch
perl-IPC-SysV-2.07-397.el8.x86_64
```

К примеру, rpm может вывести список всех установленных пакетов:

```
rpm -qa
```

Здесь -q - это сделать запрос, а - all - обо всех пакетах. До этого были ключи -ql - вывести список файлов пакета. Я не буду разбирать все ключи, потому что их очень много, но есть man и help, где всё детально расписано. Ну и можете почитать про [топ 20 команд с rpm](#), чтобы иметь чуть больше представления об этом пакетном менеджере.

```
[user@centos8 test]$ dnf repolist
Last metadata expiration check: 6:46:27 ago on Fri 28 May 2021 12:27:53 PM +04.
repo id          repo name           status
AppStream        CentOS-8 - AppStream      5,059
BaseOS           CentOS-8 - Base         1,695
*epel            Extra Packages for Enterprise Linux 8 - x86_64    7,368
*epel-modular   Extra Packages for Enterprise Linux Modular 8 - x86_64 0
extras           CentOS-8 - Extras       34
[user@centos8 test]$
```

Сейчас же перейдём к dnf. Как мы сказали, dnf работает с репозиториями - с серверами, где лежат пакеты. Репозиториев может быть много:

```
dnf repolist
```

Есть основные репозитории, где лежит большинство пакетов дистрибутива. У одного дистрибутива может быть несколько репозиториев - это зависит от разработчика, кто как свои репозитории поделит. К примеру, здесь мы видим Base и AppStream. В Base лежат базовые пакеты для работы операционной системы, а в AppStream - всякий дополнительный софт. Кроме репозиториев от самого дистрибутива, бывают репозитории и от разработчиков программ. Разработчики дистрибутива больше заинтересованы в стабильности своей системы, и ради этого могут не добавлять свежевыпущенные версии программ. Ну и не всегда бывает время оперативно добавить программу в репозиторий, так как зачастую мейнтейнеры отвечают за сотни различных пакетов. А разработчики софта больше заинтересованы, чтобы пользователи получили более новую версию программы и ради этого они могут поднять свои репозитории, где сами будут следить за актуальностью софта. При этом в их репозиториях не содержатся все пакеты системы, а только сама программа и её зависимости. К примеру, у VirtualBox есть свои репозитории.

```
[user@centos8 test]$ dnf repolist BaseOS -v
Loaded plugins: builddep, changelog, config-manager, copr, debug, debuginfo-install, download, generate_completion_cache, needs-restarting, playground, repoclosure, repodiff, repograph, repomage, reposync
DNF version: 4.2.7
cachedir: /var/tmp/dnf-user-i_ng5mwa
Unknown configuration option: countme = 1 in /etc/yum.repos.d/epel-modular.repo
```

```
Repo-id      : BaseOS
Repo-name    : CentOS-8 - Base
Repo-status   : enabled
Repo-revision: 8.3.2011
Repo-distro-tags: [cpe:/o:centos:centos:8]: , 8, C, 0, S, e, n, t
Repo-updated : Tue 27 Apr 2021 06:50:07 PM +04
Repo-pkgs    : 1,695
Repo-size    : 1.1 G
Repo-mirrors : http://mirrorlist.centos.org/?release=8&arch=x86_64&repo=BaseOS&infra=stock
Repo-baseurl : http://mirror.yer.az/CentOS/8.3.2011/BaseOS/x86_64/os/ (9 more)
Repo-expire  : 172,800 second(s) (last: Fri 28 May 2021 12:27:46 PM +04)
Repo-filename: /etc/yum.repos.d/CentOS-Base.repo
Total packages: 1,695
[user@centos8 test]$ █
```

Чуть больше информации о репозитории можно получить с помощью команды dnf repolist -v:

```
dnf repolist BaseOS -v
```

Здесь мы видим, что статус репозитория - enabled, репозиторий включён в настройках dnf. Т.е. мы можем по необходимости временно включать и отключать репозитории, чтобы поставить программу именно из этого репозитория. Также мы видим количество пакетов в репозитории и его размер. Repo-mirrors - это зеркала - серверы, дублирующие содержимое этого репозитория. Они позволяют снизить нагрузку с основных серверов дистрибутива, а также физически они располагаются ближе к вашему региону.

```
[user@centos8 test]$ curl "http://mirrorlist.centos.org/?release=8&arch=x86_64&repo=BaseOS&infra=stock"
http://mirror.yer.az/CentOS/8.3.2011/BaseOS/x86_64/os/
http://centos.mirrors.arminco.com/8.3.2011/BaseOS/x86_64/os/
http://ge.mirror.cloud9.ge/centos/8.3.2011/BaseOS/x86_64/os/
http://repos.silknet.com/centos/8.3.2011/BaseOS/x86_64/os/
http://centos.srv.magticom.ge/8.3.2011/BaseOS/x86_64/os/
http://centos.grena.ge/8.3.2011/BaseOS/x86_64/os/
http://centos-mirror.rbc.ru/pub/centos/8.3.2011/BaseOS/x86_64/os/
http://mirror.reconn.ru/centos/8.3.2011/BaseOS/x86_64/os/
http://mirror.yandex.ru/centos/8.3.2011/BaseOS/x86_64/os/
http://mirror.awanti.com/centos/8.3.2011/BaseOS/x86_64/os/
[user@centos8 test]$ █
```

Скопируем ссылку и используем утилиту curl, чтобы посмотреть содержимое этой ссылки, хотя можно было увидеть это и через браузер. http - это протокол, который используют веб сервера, и именно он используется, когда вы заходите на сайты через браузер.

Index of /CentOS/8.3.2011/BaseOS/x86_64/os

Name	Last modified	Size	Description
Parent Directory	-	-	
EFI/	2020-11-19 01:00	-	
EULA	2020-11-19 00:53	298	
GPL	2020-11-19 00:53	18K	
LICENSE	2020-11-10 19:49	18K	
Packages/	2021-05-08 05:24	-	
extra_files.json	2020-11-19 00:53	487	
images/	2020-11-19 01:00	-	
isolinux/	2021-04-20 06:26	-	
media.repo	2020-11-19 01:39	100	
repodata/	2021-04-27 18:50	-	

А Repo-baseurl содержит ссылку на репозиторий, который мы используем - как раз одно из зеркал. Здесь у нас сами пакеты в директории Packages и информация о репозитории в repodata. Как раз там хранится информация о всех файлах в репозитории, их зависимостях, версиях и т.п. За счёт этих файлов пакетный менеджер, ещё не установив сам пакет, уже знает, какие у него зависимости и в целом может вывести информацию о каждом пакете.

```
[user@centos8 test]$ dnf info httpd
Last metadata expiration check: 7:24:07 ago on Fri 28 May 2021 12:27:53 PM +04.
Available Packages
Name        : httpd
Version     : 2.4.37
Release     : 30.module_el8.3.0+561+97fdbbcc
Architecture : x86_64
Size        : 1.7 M
Source      : httpd-2.4.37-30.module_el8.3.0+561+97fdbbcc.src.rpm
Repository   : AppStream
Summary     : Apache HTTP Server
URL         : https://httpd.apache.org/
License      : ASL 2.0
Description  : The Apache HTTP Server is a powerful, efficient, and extensible
               : web server.

[user@centos8 test]$
```

dnf скачивает информацию о каждом прописанном репозитории, благодаря чему он за доли секунд может вывести информацию о любом пакете, даже если он не установлен:

```
dnf info httpd
```

Здесь же можно увидеть, а в каком именно репозитории находится данный пакет - AppStream.

```
[user@centos8 test]$ systemctl status dnf-makecache.timer
● dnf-makecache.timer - dnf makecache --timer
   Loaded: loaded (/usr/lib/systemd/system/dnf-makecache.timer; enabled; vendor preset: enabled)
     Active: active (waiting) since Fri 2021-05-28 11:58:05 +04; 8h ago
       Trigger: Fri 2021-05-28 20:12:49 +04; 7min left

May 28 11:58:05 centos8 systemd[1]: Started dnf makecache --timer.
[user@centos8 test]$ systemctl cat dnf-makecache.timer
# /usr/lib/systemd/system/dnf-makecache.timer
[Unit]
Description=dnf makecache --timer
ConditionKernelCommandLine=!rd.live.image
# See comment in dnf-makecache.service
ConditionPathExists=/run/ostree-booted
Wants=network-online.target

[Timer]
OnBootSec=10min
OnUnitInactiveSec=1h
Unit=dnf-makecache.service

[Install]
WantedBy=multi-user.target
[user@centos8 test]$ █
```

Но информация о репозитории может устаревать - каждый день в репозиторий могут добавлять новые версии пакетов, исправления багов и уязвимостей. Если dnf не будет постоянно обновлять информацию о репозиториях, то у него останется старая версия этой информации и он не будет знать о новых версиях пакетов. Поэтому в системе есть таймер:

```
systemctl status dnf-makecache.timer
systemctl cat dnf-makecache.timer
```

Как видно, примерно раз в час dnf запускает команду `dnf makecache`, что обновляет метаданные.

Кстати, можно создавать свои локальные репозитории, которые будут синхронизироваться с репозиториями дистрибутива. Это бывает полезно в компаниях, где много Linux серверов - вы можете держать один репозиторий и обновлять только его, а все остальные сервера будут брать пакеты из этого репозитория. Это поможет значительно сократить трафик в интернет и ускорить установку пакетов. Также полезно, когда интернета совсем нет. Кроме того, dvd диски и iso образы дистрибутивов также содержат репозитории, которые можно использовать при установке системы и после.

```
[user@centos8 test]$ ls /etc/yum.repos.d/
CentOS-AppStream.repo    CentOS-Extras.repo      CentOS-Vault.repo
CentOS-Base.repo          CentOS-fasttrack.repo  epel-modular.repo
CentOS-centosplus.repo    CentOS-HA.repo        epel-playground.repo
CentOS-CR.repo            CentOS-Media.repo    epel.repo
CentOS-Debuginfo.repo    CentOS-PowerTools.repo epel-testing-modular.repo
CentOS-Devel.repo         CentOS-Sources.repo  epel-testing.repo
[user@centos8 test]$ █
```

Для примера, добавим репозиторий. Это можно сделать несколькими способами - через саму команду dnf, либо создав файл. Файлы репозиториев лежат в директории `/etc/yum.repos.d`:

```
ls /etc/yum.repos.d
```

и заканчиваются на `.repo`.

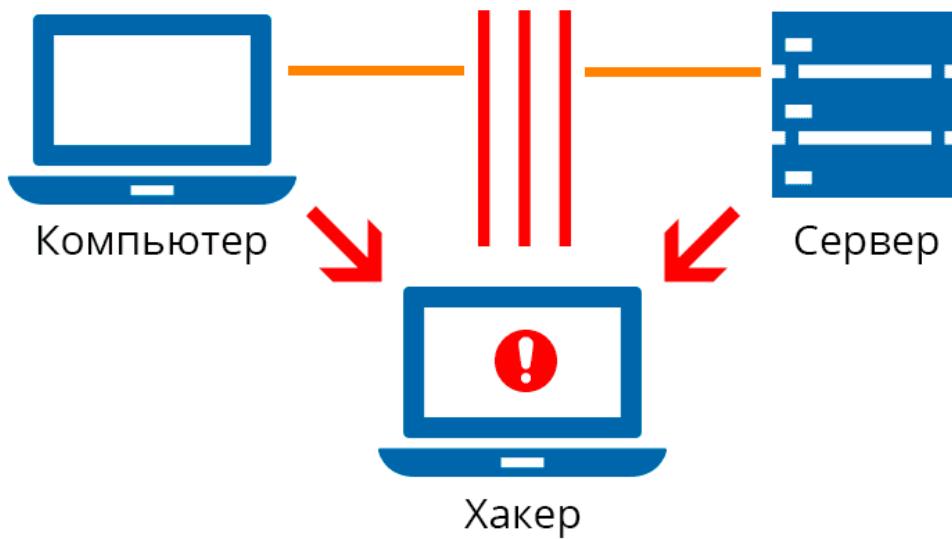
```
[user@centos8 test]$ cat /etc/yum.repos.d/CentOS-Base.repo
# CentOS-Base.repo
#
# The mirror system uses the connecting IP address of the client and the
# update status of each mirror to pick mirrors that are updated to and
# geographically close to the client. You should use this for CentOS updates
# unless you are manually picking other mirrors.
#
# If the mirrorlist= does not work for you, as a fall back you can try the
# remarked out baseurl= line instead.
#
#
[BaseOS]
name=CentOS-$releasever - Base
mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=BaseOS&infra=$infra
#baseurl=http://mirror.centos.org/$contentdir/$releasever/BaseOS/$basearch/os/
gpgcheck=1
enabled=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-centosofficial

[user@centos8 test]$
```

Посмотрим Base.repo:

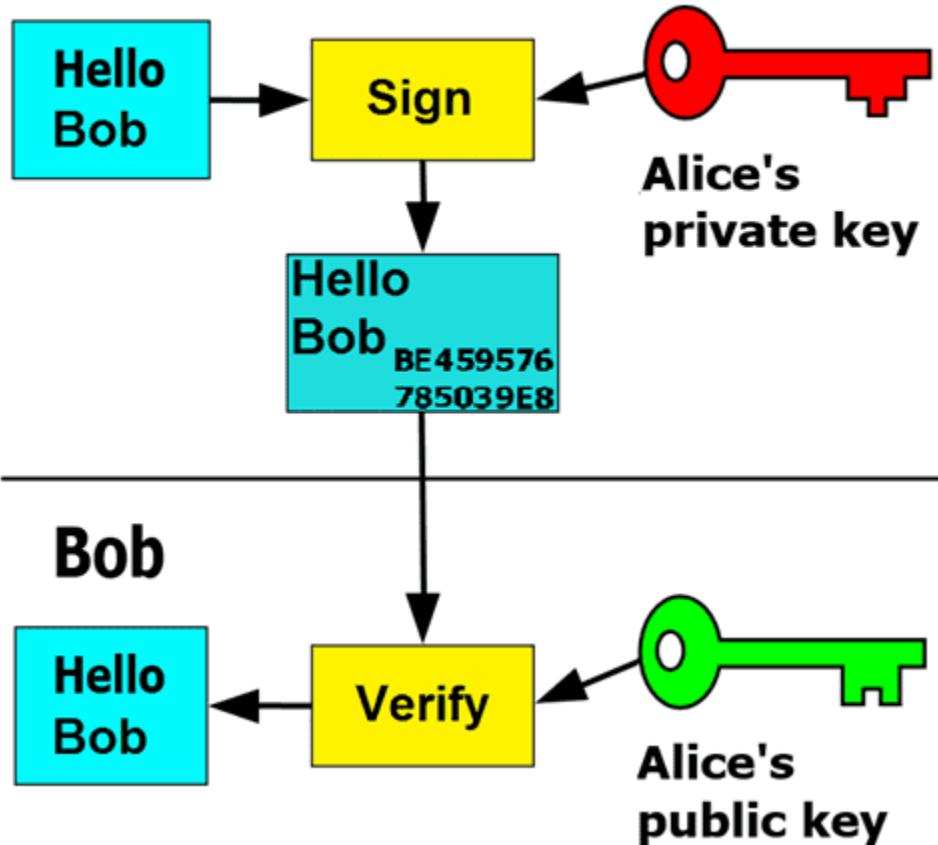
```
cat /etc/yum.repos.d/CentOS-Base.repo
```

В квадратных скобках указан идентификатор репозитория, его и следующее поле - `name` - мы можем задать сами, это просто названия. `mirrorlist` - список зеркал - не у каждого репозитория могут быть зеркала, поэтому мы можем не указывать эту опцию. `baseurl` - это ссылка на репозиторий. В данном случае ссылка закомментирована, потому что прописаны зеркала, но если бы она была, нужно было бы указать свою ссылку. `enabled` - говорит о том, что данный репозиторий включён.



`gpgcheck` - проверка ключа и `gpgkey` - сам ключ. Дело в том, что зачастую репозитории находятся в интернете. И существует MITM атака, которую мы обсуждали при изучении ssh - кто-то может встать посреди трафика и обманывать нас, притворяясь нужным сервером. Если кто-то притвориться репозиторием, то мы будем скачивать и устанавливать пакеты с этого сервера, при этом даже не догадываясь, что устанавливаем изменённые пакеты, в которых могут быть вирусы и прочий нежелательный софт.

Alice



Поэтому репозитории и все пакеты в репозиториях подписываются специальным ключом. Этот ключ есть только у людей, которые добавляют пакеты в репозиторий, и называется приватным. Когда я хочу добавить чей-то репозиторий, владелец репозитория предоставляет мне другой ключ, этот ключ может быть на сайте репозитория, либо придет вместе с системой, т.е. он доступен всем, поэтому называется публичным. Так вот, разработчик подписывает репозиторий и пакеты приватным ключом и я, благодаря публичному ключу, могу убедиться, что это именно тот самый репозиторий или пакет. Если кто-то другой посередине подставит свой сервер, то он подпишет свой репозиторий другим приватным ключом, а значит я, благодаря другому публичному ключу, пойму, что это не тот сервер. Может поначалу звучит сложно, но работает это довольно просто.

The package signature is checked by yum/dnf/zypper as well:

- Users of **Oracle Linux / RHEL** can add [the Oracle Linux repo file](#) to `/etc/yum.repos.d/`.
- Users of **Fedor**a can add [the Fedora repo file](#) to `/etc/yum.repos.d/`.
- Users of **openSUSE** can add [the openSUSE repo file](#) to `/etc/zypp/repos.d/`.

Добавим репозиторий VirtualBox-а. Идём на сайт [virtualbox.org - Downloads - Linux distributions](#) - и спускаемся в самый низ. Находим строчку - **Users of Oracle Linux/RHEL** - и нажимаем по ссылке.

```
[virtualbox]
name=Oracle Linux / RHEL / CentOS-$releasever / $basearch - VirtualBox
baseurl=http://download.virtualbox.org/virtualbox/rpm/el/$releasever/$basearch
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://www.virtualbox.org/download/oracle_vbox.asc
```

Нам откроется файлrepo. По идее достаточно просто его скопировать в нужную директорию, но мы пойдём сложным путём, скопируем только ссылку.

```
GNU nano 2.9.8 /etc/yum.repos.d/vbox.repo

[vbox]
name=VirtualBox Repo
baseurl=http://download.virtualbox.org/virtualbox/rpm/el/$releasever/$basearch
enabled=1
gpgcheck=1
```

Создадим файл:

```
sudo nano /etc/yum.repos.d/vbox.repo
```

Зададим произвольные имена в квадратных скобках и name, вставим ссылку, добавим:

```
enabled=1
gpgcheck=1
```

```
[user@centos8 test]$ sudo dnf repolist
CentOS-8 - AppStream                               6.3 kB/s | 4.3 kB   00:00
CentOS-8 - Base                                    6.6 kB/s | 3.9 kB   00:00
CentOS-8 - Extras                                 1.3 kB/s | 1.5 kB   00:01
Extra Packages for Enterprise Linux Modular 8 - x86_64    28 kB/s | 9.8 kB   00:00
Extra Packages for Enterprise Linux 8 - x86_64       19 kB/s | 8.5 kB   00:00
VirtualBox Repo                                114 kB/s | 161 kB   00:01
repo id          repo name                           status
AppStream        CentOS-8 - AppStream                5,059
BaseOS           CentOS-8 - Base                   1,695
epel             Extra Packages for Enterprise Linux 8 - x86_64    7,368
epel-modular     Extra Packages for Enterprise Linux Modular 8 - x86_64  0
extras           CentOS-8 - Extras                  34
vbox             VirtualBox Repo                  25
[user@centos8 test]$ sudo dnf search virtualbox
Last metadata expiration check: 0:00:27 ago on Fri 28 May 2021 09:05:04 PM +04.
=====
===== Name & Summary Matched: virtualbox =====
VirtualBox-5.2.x86_64 : Oracle VM VirtualBox
VirtualBox-6.0.x86_64 : Oracle VM VirtualBox
VirtualBox-6.1.x86_64 : Oracle VM VirtualBox
[user@centos8 test]$
```

Посмотрим список репозиториев:

```
sudo dnf repolist
```

и в списке появился репозиторий vbox. Попробуем найти пакет virtualbox:

```
sudo dnf search virtualbox
```

И мы видим 3 подходящих пакета.

```
[user@centos8 test]$ sudo dnf install VirtualBox-6.1
Last metadata expiration check: 0:03:08 ago on Fri 28 May 2021 09:05:04 PM +04.
Dependencies resolved.
=====
 Package           Architecture   Version      Repository    Size
=====
Installing:
 VirtualBox-6.1      x86_64        6.1.22_144080_el8-1      vbox          90 M
Upgrading:
 qgnameplatform      x86_64        0.4-3.el8            AppStream     133 k
 qt5-qtbase          x86_64        5.12.5-6.el8          AppStream     3.4 M
(8/9): qt5-qtdeclarative-5.12.5-1.el8.x86_64.rpm           3.1 MB/s | 3.7 MB  00:01
(9/9): VirtualBox-6.1-6.1.22_144080_el8-1.x86_64.rpm       7.1 MB/s | 90 MB   00:12
Total                                         8.1 MB/s | 104 MB  00:12
warning: /var/cache/dnf/vbox-251a95df6cac77a1/packages/VirtualBox-6.1-6.1.22_144080_el8-1.x86_64
.rpm: Header V4 DSA/SHA1 Signature, key ID 98ab5139: NOKEY
Public key for VirtualBox-6.1-6.1.22_144080_el8-1.x86_64.rpm is not installed
The downloaded packages were saved in cache until the next successful transaction.
You can remove cached packages by executing 'dnf clean packages'.
Error: GPG check FAILED
[user@centos8 test]$
```

Попробуем установить пакет:

```
sudo dnf install VirtualBox-6.1
```

Установка запустилась, мы видим что этот пакет будет скачиваться с репозитория vbox, и видим все зависимости, которые подтягиваются с AppStream репозитория. Но как только дело дошло до установки пакета, dnf понял, что для virtualbox не установлен публичный ключ и остановил процесс.

```
GNU nano 2.9.8                               /etc/yum.repos.d/vbox.repo

[vbox]
name=VirtualBox Repo
baseurl=http://download.virtualbox.org/virtualbox/rpm/el/$releasever/$basearch
enabled=1
gpgcheck=0

Upgraded:
qgnameplatform-0.4-3.el8.x86_64             qt5-qtbase-5.12.5-6.el8.x86_64
qt5-qtbase-common-5.12.5-6.el8.noarch        qt5-qtbase-gui-5.12.5-6.el8.x86_64
qt5-qtdeclarative-5.12.5-1.el8.x86_64       qt5-qtxmlpatterns-5.12.5-1.el8.x86_64

Installed:
VirtualBox-6.1-6.1.22_144080_el8-1.x86_64    SDL-1.2.15-38.el8.x86_64
qt5-qtx11extras-5.12.5-1.el8.x86_64

Complete!
[user@centos8 test]$
```

Мы можем отключить проверку пакетов или добавить ключ. Для интернет репозиториев отключать проверку не стоит, но в локальных иногда можно не заморачиваться с ключами. Попробуем отключить проверку - заходим в созданный файл:

```
sudo nano /etc/yum.repos.d/vbox.repo
```

и меняем `gpgcheck=0`. Затем заново запускаем установку. И она проходит успешно.

```
[user@centos8 test]$ sudo dnf remove VirtualBox-6.1
[sudo] password for user:
Dependencies resolved.
=====
 Package          Architecture Version      Repository    Size
 =====
 Removing:
 VirtualBox-6.1      x86_64     6.1.22_144080_el8-1    @vbox        198 M
 Removing unused dependencies:
   SDL              x86_64     1.2.15-38.el8      @AppStream   475 k
   qt5-qtx11extras x86_64     5.12.5-1.el8      @AppStream   115 k

Transaction Summary
=====
 Remove 3 Packages

Freed space: 199 M
Is this ok [y/N]: y
```

А для удаления пакета воспользуемся командой `dnf remove`:

```
sudo dnf remove VirtualBox-6.1
```

Теперь попробуем добавить ключ.

```
[virtualbox]
name=Oracle Linux / RHEL / CentOS-$releasever / $basearch - VirtualBox
baseurl=http://download.virtualbox.org/virtualbox/rpm/el/$releasever/$basearch
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://www.virtualbox.org/download/oracle_vbox.asc
```

В том же файле, который предложил `virtualbox`, в качестве ключа была предложена ссылка на файл на сайте виртуалбокса. Мы же, для большего понимания процесса, сделаем несколько иначе - скачаем ключ и добавим его в систему.

RPM-based Linux distributions

We provide a yum/dnf-style repository for Oracle Linux/Fedora/RHEL/openSUSE. All .rpm packages are signed. The Oracle public key for rpm can be downloaded [here](#). You can add this key (*not normally necessary, see below!*) with

```
sudo rpm --import oracle_vbox.asc
```

or combine downloading and registering:

```
wget -q https://www.virtualbox.org/download/oracle_vbox.asc -O- | rpm --import -
```

The key fingerprint is

```
7B0F AB3A 13B9 0743 5925 D9C9 5442 2A4B 98AB 5139
Oracle Corporation (VirtualBox archive signing key) <info@virtualbox.org>
```

В предыдущей ссылке поднимемся чуть выше, где увидим команду:

```
rpm --import
```

Нам нужно скачать ключ, а потом импортировать. Вторая команда делает это разом - wget скачивает файл с интернета, передаёт его через rpm команде rpm, которая импортирует. Можно сразу ввести эту команду, но для понимания разделим её на две части.

```
[user@centos8 test]$ wget -q https://www.virtualbox.org/download/oracle_vbox.asc
[user@centos8 test]$ ls
etc  lvm2-2.03.09-5.el8.x86_64.rpm  oracle_vbox.asc  run  usr
[user@centos8 test]$ cat oracle_vbox.asc
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.9 (GNU/Linux)

mQGiBEvy0LARBACPBH1AUv6krDseyvbL63CWS9fw43iReZ+NmgmDp4/sPsYHLduQ
rxKSqiK7fgFFE+fas/7DCaZXIQu6hnqeD3CgnX0w1+gYiyqEuPY1LQH9okBR5o92
/s2FLm7oUN4RNGv6vWoNSH1ZiRHknL5D0pKSGTKU+pG6cBYWJytAuJZHfwCguS55
aPZuXfhjaWsXG8TF6GH0K8ED/RY3KbirJ7rueK/7KL7ziwqn4CdhQSL0hbZ/R2E0
bknJQDo+vWJciRRRpTe+AG59ctgDF7lEXpjvCms0atyKtE8h0bNaMJ5p48l/sgFL
```

Для начала скачаем ключ:

```
wget -q https://www.virtualbox.org/download/oracle_vbox.asc
ls
```

Можем посмотреть содержимое ключа:

```
cat oracle_vbox.asc
```

Это текстовой файл, в котором большой набор символов.

```
[user@centos8 test]$ sudo rpm --import oracle_vbox.asc
[user@centos8 test]$ sudo nano /etc/yum.repos.d/vbox.repo
[user@centos8 test]$ tail -1 /etc/yum.repos.d/vbox.repo
gpgcheck=1
[user@centos8 test]$ sudo dnf install VirtualBox-6.1 -y
VirtualBox Repo                                     7.8 kB/s | 2.9 kB   00:00
Dependencies resolved.
=====
Package           Architecture Version       Repository      Size
=====
Installing:
VirtualBox-6.1      x86_64     6.1.22_144080_el8-1    vbox        90 M
Installing dependencies:
SDL                  x86_64     1.2.15-38.el8          AppStream    218 k
qt5-qt5x11extras    x86_64     5.12.5-1.el8          AppStream    40 k
```

Теперь ключ нужно импортировать:

```
sudo rpm --import oracle_vbox.asc
```

После этого не забудем заново включить проверку ключа:

```
sudo nano /etc/yum.repos.d/vbox.repo
```

```
gpgcheck=1
```

И попробуем заново установить виртуалбокс:

```
sudo dnf install VirtualBox-6.1 -y
```

На этот раз добавим ключ -y, чтобы он не спрашивал, уверены ли мы, что хотим установить. И снова установка прошла успешно.

Хорошо, с репозиториями разобрались. Но не для каждой программы есть репозиторий, и не каждый пакет есть в репозитории. Скажем, нам нужен определённый пакет, который есть в репозитории другого дистрибутива, либо в новой версии нашего дистрибутива. Но подключать их не стоит - пакетный менеджер может увидеть какие-то пакеты новее в другом дистрибутиве и попытаться обновиться. Тогда у нас часть пакетов обновится, часть останется и выйдет куча проблем с зависимостями. Поэтому надо стараться избегать использования сторонних репозиториев.

The screenshot shows the rpmfind.net search interface. At the top, there's a navigation bar with links like Index, index by Group, index by Distribution, index by Vendor, index by creation date, index by Name, Mirrors, and Help. Below the navigation bar, a message says: "The search service can find package by either name (**apache**), provides(**webserver**), absolute file names (**/usr/bin/apache**), binaries (**gprof**) or shared libraries support multiple arguments yet...". A note below it says: "The System and Arch are optional added filters, for example System could be "redhat", "redhat-7.2", "mandrake" or "gnome", Arch could be "i386" or "src", etc." There are input fields for 'Search ...' (containing 'atop'), 'System' (containing 'epel'), and 'Arch'. The main content area has a title 'RPM resource atop' and a brief description: "The program atop is an interactive monitor to view the load on a Linux-system. It shows the occupation of the most critical hardware-resources (from a performance perspective) like CPU, memory, disk and network. It also shows which processes are responsible for the indicated load (again CPU-, memory-, disk- and network-load on process-level). It provides system- and process-level information in raw format for long-term analysis." Below this, a section titled 'Found 5 sites for atop' lists five URLs:

- <http://www.atcomputing.nl/Tools/atop/>
- <http://www.atptool.nl/>
- <http://www.ATComputing.nl/atop>
- <ftp://ftp.atcomputing.nl/pub/tools/linux/>
- <http://atptool.nl/>

Below this is another section titled 'Found 8 RPM for atop' which lists eight RPM packages with their details:

Package	Summary	Distribution	Download
atop-2.6.0-6.el8.aarch64.html	An advanced interactive monitor to view the load on system and process level EPEL 8 for aarch64	atop-2.6.0-6.el8.aarch64.rpm	
atop-2.6.0-6.el8.ppc64le.html	An advanced interactive monitor to view the load on system and process level EPEL 8 for ppc64le	atop-2.6.0-6.el8.ppc64le.rpm	
atop-2.6.0-6.el8.s390x.html	An advanced interactive monitor to view the load on system and process level EPEL 8 for s390x	atop-2.6.0-6.el8.s390x.rpm	
atop-2.6.0-6.el8.x86_64.html	An advanced interactive monitor to view the load on system and process level EPEL 8 for x86_64	atop-2.6.0-6.el8.x86_64.rpm	
atop-2.6.0-6.el7.ppc64le.html	An advanced interactive monitor to view the load on system and process level EPEL 7 for ppc64le	atop-2.6.0-6.el7.ppc64le.rpm	
atop-2.6.0-6.el7.x86_64.html	An advanced interactive monitor to view the load on system and process level EPEL 7 for x86_64	atop-2.6.0-6.el7.x86_64.rpm	
atop-2.4.0-4.el7.aarch64.html	An advanced interactive monitor to view the load on system and process level EPEL 7 for aarch64	atop-2.4.0-4.el7.aarch64.rpm	
atop-2.4.0-1.el7.ppc64.html	An advanced interactive monitor to view the load on system and process level EPEL 7 for ppc64	atop-2.4.0-1.el7.ppc64.rpm	

Попробуем найти как-то тестовый пакет и установить. Можно погуглить, либо зайти на сайт rpmfind.net и попытаться что-то найти. Например, atop. Найдём подходящую архитектуру - x86_64.rpm и скопируем ссылку под Download.

```
[user@centos8 test]$ sudo dnf install http://rpmfind.net/linux/epel/8/Everything/x86_64/Packages/a/atop-2.6.0-6.el8.x86_64.rpm
Last metadata expiration check: 0:24:21 ago on Fri 28 May 2021 09:35:42 PM +04.
atop-2.6.0-6.el8.x86_64.rpm                                              246 kB/s | 177 kB     00:00
Dependencies resolved.
=====
 Package          Architecture      Version       Repository      Size
 =====
 Installing:
 atop              x86_64          2.6.0-6.el8   @commandline    177 k
 Installing dependencies:
 python3-py3nvml    noarch         0.2.6-1.el8   epel           80 k
 python3-xmlltodict noarch         0.12.0-6.el8  epel           24 k
 Transaction Summary
 =====
```

А дальше достаточно в командной строке прописать sudo dnf install и ссылку на rpm файл:

```
sudo dnf install http://rpmfind.net/linux/epel/8/Everything/x86_64/Packages/a/atop-2.6.0-6.el8.x86_64.rpm
```

dnf сам скачает этот файл, найдёт для него зависимости и установит пакет. Обратите внимание, как

dnf в качестве репозитория для этого файла указал @commandline.

```
[user@centos8 test]$ sudo dnf install ./lvm2-2.03.09-5.el8.x86_64.rpm
Last metadata expiration check: 0:28:24 ago on Fri 28 May 2021 09:35:42 PM +04.
Dependencies resolved.

=====
Package           Architecture Version      Repository  Size
=====
Upgrading:
device-mapper      x86_64        8:1.02.171-5.el8   BaseOS     373 k
device-mapper-event x86_64        8:1.02.171-5.el8   BaseOS     268 k
device-mapper-event-libs x86_64        8:1.02.171-5.el8   BaseOS     267 k
device-mapper-libs  x86_64        8:1.02.171-5.el8   BaseOS     406 k
lvm2-libs          x86_64        8:2.03.09-5.el8    BaseOS     1.1 M
lvm2               x86_64        8:2.03.09-5.el8    @commandline 1.6 M

Transaction Summary
=====
Upgrade 6 Packages

Total size: 4.0 M
Total download size: 2.4 M
Is this ok [y/N]:
```

Похожему принципу можно установить скачанный файл - sudo dnf install и указать на файл:

```
sudo dnf install ./lvm2-2.03.09-5.el8.x86_64.rpm
```

```
[user@centos8 test]$ sudo dnf upgrade lvm2 ^C
[user@centos8 test]$ sudo dnf upgrade
Last metadata expiration check: 0:02:22 ago on Fri 28 May 2021 10:20:55 PM +04.
Dependencies resolved.

=====
Package           Arch  Version      Repo      Size
=====
Installing:
kernel          x86_64 4.18.0-240.22.1.el8_3  BaseOS    4.4 M
kernel-core      x86_64 4.18.0-240.22.1.el8_3  BaseOS    30 M
kernel-devel     x86_64 4.18.0-240.22.1.el8_3  BaseOS    17 M
kernel-modules   x86_64 4.18.0-240.22.1.el8_3  BaseOS    26 M
Upgrading:
PackageKit       x86_64 1.1.12-6.el8      AppStream 599 k
PackageKit-command-not-found x86_64 1.1.12-6.el8      AppStream 27 k
```

Касательно установки программ разобрались, теперь насчёт обновления. Проверить наличие обновлений можно с помощью:

```
dnf check-upgrade
```

Обновлять программы можно с помощью:

```
sudo dnf upgrade
```

Если указать после команды определённый пакет, то обновится только он и его зависимости. Если же ничего не указывать, то обновится вся система.

```
[user@centos8 test]$ sudo dnf needs-restarting -r
No core libraries or services have been updated since boot-up.
Reboot should not be necessary.
[user@centos8 test]$
```

После некоторых обновлений следует делать перезагрузку, но не после всех, поэтому вы можете быть неуверенны, а нужно ли? Можно воспользоваться командой:

```
sudo dnf needs-restarting -r
```

она подскажет.

```
[user@centos8 test]$ sudo dnf provides nslookup
Last metadata expiration check: 0:25:33 ago on Fri 28 May 2021 10:20:55 PM +04.
bind-utils-32:9.11.4-26.P2.el8.x86_64 : Utilities for querying DNS name servers
Repo          : @System
Matched from:
Filename     : /usr/bin/nslookup

bind-utils-32:9.11.20-5.el8_3.1.x86_64 : Utilities for querying DNS name servers
Repo          : AppStream
Matched from:
Filename     : /usr/bin/nslookup

[user@centos8 test]$ █
```

Иногда бывает, что вы помните название программы, но не знаете, в каком именно пакете она находится. К примеру, вы можете помнить команду nslookup, но не знать про название пакета. И пусть даже не команда, а конфиг файл, библиотека или любой файл программы. dnf provides позволяет найти имя пакета, в котором содержится нужный файл:

```
sudo dnf provides nslookup
```

```
[user@centos8 test]$ sudo dnf grouplist
Last metadata expiration check: 0:31:12 ago on Fri 28 May 2021 10:20:55 PM +04.
Available Environment Groups:
  Server with GUI
  Server
  Minimal Install
  KDE Plasma Workspaces
  Virtualization Host
  Custom Operating System
Installed Environment Groups:
  Workstation
Installed Groups:
  Graphical Administration Tools
  System Tools
Available Groups:
  Container Management
  .NET Core Development
  RPM Development Tools
```

Для удобства некоторые пакеты, которые часто используют вместе, объединены в группы. Посмотреть список доступных групп можно с помощью dnf grouplist:

```
sudo dnf grouplist
```

```
[user@centos8 test]$ sudo dnf groupinfo "Container Management"
Last metadata expiration check: 0:33:10 ago on Fri 28 May 2021 10:20:55 PM +04.

Group: Container Management
Description: Tools for managing Linux containers
Mandatory Packages:
    buildah
    containernetworking-plugins
    podman
Optional Packages:
    python3-psutil
[user@centos8 test]$ sudo dnf groupinstall "Container Management"
```

С помощью groupinfo можно понять, какие пакеты входят в группу, а с помощью groupinstall установить все эти пакеты:

```
dnf groupinfo "Container Management"
sudo dnf groupinstall "Container Management"
```

```
[user@centos8 test]$ dnf module list
Last metadata expiration check: 0:29:37 ago on Fri 28 May 2021 10:32:46 PM +04.
CentOS-8 - AppStream
Name           Stream      Profiles Summary
389-ds         1.4        389 Directory Server (base)
ant            1.10 [d]   common [ Java build tool
                        d]
container-tools rhel8 [d] common [ Most recent (rolling) versions of podman, buildah
                                , skopeo, runc, common, runc, common, CRIU, Udict
                                , etc as well as dependencies such as container-s
                                elinux built and tested together, and updated as
                                frequently as every 12 weeks.
container-tools     1.0       common [ Stable versions of podman 1.0, buildah 1.5, skope
                                o 0.1, runc, common, CRIU, Udict, etc as well as
                                dependencies such as container-selinux built and
                                tested together, and supported for 24 months.
container-tools     2.0       common [ Stable versions of podman 1.6, buildah 1.11, skop
                                eo 0.1, runc, common, etc as well as dependencies
```

Помимо групп есть модули:

```
sudo dnf module list
```

Модули, как и группы, содержат сразу несколько пакетов. Но при этом пакеты в модулях связаны версиями. Обратите внимание, есть несколько модулей container-tools, и в одной версии используются самые актуальные версии пакетов, а в других используется более старая версия пакетов, при этом учтены зависимости и они также не самые свежие.

```
[user@centos8 test]$ sudo dnf module info container-tools:2.0
Last metadata expiration check: 0:48:19 ago on Fri 28 May 2021 10:20:55 PM +04.
Name          : container-tools
Stream        : 2.0
Version       : 8030020210222060435
Context       : 2a301c24
Architecture  : x86_64
Profiles      : common [d]
Default profiles: common
Repo          : AppStream
Summary       : Stable versions of podman 1.6, buildah 1.11, skopeo 0.1, runc, common, etc as
well as dependencies such as container-selinux built and tested together, and supported as docu
mented on the Application Stream lifecycle page.
```

Более детально узнать о каком-то модуле можно с помощью:

```
dnf module info
```

указав при этом нужный модуль и его версию:

```
sudo dnf module info container-tools:2.0
```

```
[user@centos8 test]$ sudo dnf module install container-tools:2.0
Last metadata expiration check: 0:49:54 ago on Fri 28 May 2021 10:20:55 PM +04.
Dependencies resolved.

=====
Package          Arch    Version           Repo      Size
=====
Installing group/module packages:
buildah          x86_64  1.11.6-8.module_el8.3.0+479+69e2ae26  AppStream 8.4 M
cockpit-podman   noarch  11-1.module_el8.3.0+479+69e2ae26  AppStream 1.0 M
```

Ну а для установки, соответственно, `dnf module install` с нужным модулем и версией:

```
sudo dnf module install container-tools:2.0
```

```
[user@centos8 test]$ dnf history
ID | Command line                                | Date and time | Action(s) | Altered
-- | -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- |
37 | remove yum-utils                            | 2021-05-28 22:39 | Removed   | 1
36 | install yum-utils -y                      | 2021-05-28 22:32 | I, U     | 25
35 | install http://rpmfind.net/linux/epel/8/E | 2021-05-28 22:00 | Install   | 3
34 | install VirtualBox-6.1 -y                | 2021-05-28 21:35 | Install   | 3 EE
33 | remove VirtualBox-6.1                     | 2021-05-28 21:19 | Removed   | 3
32 | install VirtualBox-6.1                   | 2021-05-28 21:14 | I, U     | 9 EE
31 | update setroubleshell-server-3.3.24-1.el8 | 2021-04-30 19:44 | Upgrade   | 2
30 | install VirtualBox-6.1                   | 2021-03-26 21:16 | I, U     | 2
```

Кроме этого `dnf` хранит историю всех операций:

```
dnf history
```

В этой истории у каждой операции есть номера, для примера возьмём 32.

```
[user@centos8 test]$ dnf history info 32
Transaction ID : 32
Begin time     : Fri 28 May 2021 09:14:19 PM +04
Begin rpmdb    : 1618:618b59d905f0c163faf4f8e0f96fa2ca629ce99e
End time       : Fri 28 May 2021 09:15:09 PM +04 (50 seconds)
End rpmdb      : 1621:eba750f1473a9064758f47cf627bca26dda46046
User          : User <user>
Return-Code    : Success
Releasever    : 8
Command Line   : install VirtualBox-6.1
Comment        :
Packages Altered:
  Install  SDL-1.2.15-38.el8.x86_64          @AppStream
  Install  qt5-qtx11extras-5.12.5-1.el8.x86_64  @AppStream
  Install  VirtualBox-6.1-6.1.22_144080_el8-1.x86_64 @vbox
  Upgrade  qgnomeplatform-0.4-3.el8.x86_64      @AppStream
  Upgraded qgnomeplatform-0.4-2.el8.x86_64      @@System
```

Мы можем детальнее узнать про эту операцию, указав `history info 32`:

```
dnf history info 32
```

```
[user@centos8 test]$ sudo dnf history undo 32
Last metadata expiration check: 0:55:27 ago on Fri 28 May 2021 10:20:55 PM +04.
Undoing transaction 32, from Fri 28 May 2021 09:14:19 PM +04
  Install  SDL-1.2.15-38.el8.x86_64          @AppStream
  Install  qt5-qtx11extras-5.12.5-1.el8.x86_64    @AppStream
  Install  VirtualBox-6.1-6.1.22_144080_el8-1.x86_64 @vbox
```

```
[user@centos8 test]$ sudo dnf history redo 32
Last metadata expiration check: 0:55:34 ago on Fri 28 May 2021 10:20:55 PM +04.
Repeating transaction 32, from Fri 28 May 2021 09:14:19 PM +04
  Install  SDL-1.2.15-38.el8.x86_64          @AppStream
  Install  qt5-qtx11extras-5.12.5-1.el8.x86_64    @AppStream
  Install  VirtualBox-6.1-6.1.22_144080_el8-1.x86_64 @vbox
  Upgrade  qgnameplatform-0.4-3.el8.x86_64        @AppStream
  Upgraded qgnameplatform-0.4-2.el8.x86_64        @@System
  Upgrades qgnameplatform-0.4-2.el8.x86_64        @AppStream
```

Ну и проделанные операции можно отменить или заново применить, с помощью `dnf history undo` или `dnf history redo`:

```
sudo dnf history undo 32
sudo dnf history redo 32
```

```
[user@centos8 test]$ dnf help
usage: dnf [options] COMMAND

List of Main Commands:

alias                      List or create command aliases
autoremove                  remove all unneeded packages that were originally installed as depende
ncies
check                      check for problems in the packagedb
check-update                check for available package upgrades
clean                       remove cached data
deplist                     List package's dependencies and what packages provide them
distro-sync                 synchronize installed packages to the latest available versions
downgrade                   Downgrade a package
group                       display, or use, the groups information
help                        display a helpful usage message
history                     display, or use, the transaction history
info                         display details about a package or group of packages
install                     install a package or packages on your system
list                        list a package or groups of packages
```

У `dnf` огромное количество различных возможностей и все я разобрать не смогу:

```
sudo dnf help
```

С каким-то функционалом вы познакомитесь со временем, а какой-то вы ни разу и не используетесь. Но есть документация и, если вам что-то понадобится уточнить, скорее всего в ней вы сможете найти ответ.

Подведём итоги. Сегодня мы с вами разобрали, что такое пакеты, чем занимаются пакетные менеджеры, поговорили о репозиториях и разобрали базовые команды для работы с `dnf`. Теперь вы знаете, как устанавливать программы, откуда их брать, как обновлять систему и как это всё работает.

2.48 48. Восстановление доступа

2.48.1 48. Восстановление доступа



🔒 Encrypted

Complete database encryption using industry standard 256-bit AES. Fully compatible with KeePass Password Safe formats. Your password database works offline and requires no internet connection.

🕒 Cross-Platform

Every feature looks, feels, works, and is tested on Windows, macOS, and Linux. You can expect a seamless experience no matter which operating system you are using.

❤️ Open Source

The full source code is published under the terms of the GNU General Public License and made available on GitHub. Use, inspect, change, and share at will; contributions by everyone are welcome.

При работе вы можете сталкиваться с ситуациями, когда вам нужно восстановить доступ к системе. К примеру, потерялся пароль или предыдущий админ его не предоставил. Прежде чем начнём, небольшой совет - используйте парольные менеджеры, например, keepassxc. У вас может быть множество серверов, различные аккаунты на сайтах и везде требуется пароль. Придумать для различных ресурсов разные пароли и потом всё это запомнить - нереально. Использовать один и тот же пароль, каким бы сложным он не был - тоже не правильно - всегда есть риск утечки. А с парольным менеджером вы можете сгенерировать рандомные пароли для различных ресурсов и не беспокоиться, что кто-то где-то украдёт пароль и получит доступ ко всем системам. Но саму базу паролей надо бэкапить, иначе можете остаться без единого пароля.



Сейчас мы займёмся сбросом пароля root пользователя. В рабочей системе без прав суперпользователя это сделать невозможно, если не принимать во внимание всякие уязвимости. Поэтому пароль надо сбрасывать ещё до того, как система запустилась. Чтобы понять, как это сделать, вспомним процесс запуска операционной системы. Мы знаем, что пароли хранятся в файле /etc/shadow. Корень у нас монтируется ещё на этапе с временным корнем - initramfs - откуда берутся необходимые для монтирования модули. Дальше у нас запускается система инициализации. Надо вклиниваться в промежуток после монтирования корня и до запуска системы инициализации. Это можно сделать несколькими способами - я сначала покажу самый простой и быстрый на мой взгляд, ну и потом разберу самый популярный в интернете.

```

CentOS Linux (4.18.0-240.15.1.el8_3.x86_64) 8
CentOS Linux (4.18.0-147.8.1.el8_1.x86_64) 8 (Core)
CentOS Linux (4.18.0-147.el8.x86_64) 8 (Core)
CentOS Linux (0-rescue-91222dde95634287a2336070235dc625) 8 (Core)

Use the ↑ and ↓ keys to change the selection.
Press 'e' to edit the selected item, or 'c' for a command prompt.

```

И так, чтобы изменить процесс запуска, надо изменить настройки загрузчика grub. Для этого при запуске системы в меню grub на первом пункте нажимаем e - edit.

```

load_video
set gfx_payload=keep
insmod gzio
linux ($root)/vmlinuz-4.18.0-240.15.1.el8_3.x86_64 root=/dev/mapper/cl_centos8\
-root ro crashkernel=auto resume=/dev/mapper/cl_centos8-swap rd.lvm.lv=cl_cen\
tos8/root rd.lvm.lv=cl_centos8/swap
initrd  ($root)/initramfs-4.18.0-240.15.1.el8_3.x86_64.img $tuned_initrd

```

Спустимся на строчку с параметрами ядра linux. Здесь есть параметр `ro` - read only. Во время запуска основной корень предварительно монтируется в режиме чтения.

```

load_video
set gfx_payload=keep
insmod gzio
linux ($root)/vmlinuz-4.18.0-240.15.1.el8_3.x86_64 root=/dev/mapper/cl_centos8\
-root rw init=/bin/bash crashkernel=auto resume=/dev/mapper/cl_centos8-swap rd\
.lvm.lv=cl_centos8/root rd.lvm.lv=cl_centos8/swap
initrd  ($root)/initramfs-4.18.0-240.15.1.el8_3.x86_64.img $tuned_initrd

```

Заменим го на `rw` - нам нужно, чтобы корень был доступен для изменений, всё таки мы собираемся изменить файл `/etc/shadow`. Также добавляем опцию `init=/bin/bash` - таким образом мы вместо системы инициализации запускаем bash, тем самым предотвращаем нормальный запуск системы и сразу получаем доступ к оболочке на незапущенной системе. Наша изменения в grub сохраняются только на текущую сессию и после перезагрузки всё сбросится, так что тут ничего страшного нет. Чтобы запуститься с новыми параметрами, нажимаем `ctrl+x`.

```
[ OK ] Started Cleanup udevd DB.
[ 4.097015] systemd-journald[259]: Received SIGTERM from PID 1 (systemd).
[ 4.107789] printk: bash: 20 output lines suppressed due to ratelimiting
bash-4.4# su -
[root@centos8 ~]# load_policy -i /etc/selinux/targeted/policy/policy.31
load_policy: Warning! Policy file argument (/etc/selinux/targeted/policy/policy.31) is no longer supported, installed policy is always loaded. Continuing...
[ 367.815925] audit: type=1404 audit(1625386309.879:2): enforcing=1 old_enforcing=0 auuid=4294967295 ses=4294967295 enabled=1 old-enabled=1 lsm=selinux res=1
[ 368.125819] SELinux: policy capability network_peer_controls=1
[ 368.126045] SELinux: policy capability open_perms=1
[ 368.126259] SELinux: policy capability extended_socket_class=1
[ 368.126474] SELinux: policy capability always_check_network=0
[ 368.126698] SELinux: policy capability cgroup_seclabel=1
[ 368.126971] SELinux: policy capability mnp_nosuid_transition=1
[ 368.135542] audit: type=1403 audit(1625386310.198:3): auuid=4294967295 ses=4294967295 lsm=selinux res=1
[root@centos8 ~]# passwd
Changing password for user root.
New password:
BAD PASSWORD: The password is a palindrome
Retype new password:
passwd: all authentication tokens updated successfully.
[root@centos8 ~]#
```

После этого нас встретит bash. Для начала я покажу все команды, чтобы было легче запомнить, а потом разберём, что и зачем. Пишем:

```
su -
load_policy -i /etc/selinux/targeted/policy/policy.31
```

В зависимости от обновления название последнего файла может отличаться - просто в директории policy нажимаете tab и баш дополняет нужный файл, он там обычно единственный. После этого пишем:

```
passwd
```

и вводим новый пароль дважды. Затем через виртуалбокс перезагружаем виртуалку.

```
su -
load_policy -i /etc/selinux/targeted/policy/policy.31
passwd
```

[user@centos8 ~]\$ su
Password:
[root@centos8 user]# █

После запуска операционной системы проверим новый пароль:

```
su
```

всё работает.

```
bash-4.4#  
bash-4.4# echo $PATH  
/usr/local/bin:/usr/bin  
bash-4.4# su  
[root@centos8 ~]# echo $PATH  
/usr/local/bin:/usr/bin  
[root@centos8 ~]# exit  
bash-4.4# su -  
[root@centos8 ~]# echo $PATH  
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin  
[root@centos8 ~]#
```

Теперь разберёмся с командами. Первое что мы ввели:

```
su -
```

Это нужно, чтобы прогрузилось окружение пользователя. Если проверим bash, который у нас загружается в начале:

```
echo $PATH
```

в нём только две директории в переменной PATH - /usr/local/bin и /usr/bin. Если мы напишем просто:

```
su
```

окружение останется, а значит в этой переменной останутся те же директории:

```
echo $PATH
```

Из темы про su мы помним, что для загрузки окружения пользователя к su надо добавлять дефис:

```
su -
```

После чего в переменной \$PATH:

```
echo $PATH
```

мы увидим новые директории.

```
[root@centos8 ~]# getenforce  
Disabled  
[root@centos8 ~]# ls -lZ /etc/shadow  
-----. 1 root root system_u:object_r:shadow_t:s0 2068 Jul 4 12:12 /etc/shadow  
[root@centos8 ~]# passwd  
Changing password for user root.  
New password:  
BAD PASSWORD: The password is a palindrome  
Retype new password:  
passwd: all authentication tokens updated successfully.  
[root@centos8 ~]# ls -lZ /etc/shadow  
-----. 1 root root ? 2068 Jul 4 12:27 /etc/shadow  
[root@centos8 ~]# _
```

Шаг с load_policy нужен на системах, где стоит selinux. Т.е. на RHEL и Centos-е это нужно, а на том же Debian или Ubuntu - нет, если конечно вы не поставили на них selinux. Зачем этот шаг вообще нужен? При запуске системы у нас не прогрузился selinux:

```
getenforce
```

Как видите, `selinux disabled`. Когда selinux выключен, работать с файлами не стоит, иначе контекст сбрасывается. Смена пароля - изменение файла `/etc/shadow`. Посмотрим контекст этого файла:

```
ls -lZ /etc/shadow
```

Сейчас контекст `shadow_t`. Для наглядности пропустим шаг с `load_policy` и сразу зададим пароль - `passwd`. После этого контекст файла пропадёт:

```
ls -lZ /etc/shadow
```

Мы говорили, что selinux блокирует процессам доступ к файлам, если контекст не совпадает. И в итоге из-за потерянного контекста при запуске система не сможет обратиться к `/etc/shadow` и поэтому не запустится.

```
[root@centos8 ~]# restorecon -v /etc/shadow
[root@centos8 ~]# ls -lZ /etc/shadow
----- 1 root root ? 2068 Jul 4 12:27 /etc/shadow
[root@centos8 ~]# load_policy -i /etc/selinux/targeted/policy/policy.31
load_policy: Warning! Policy file argument (/etc/selinux/targeted/policy/policy.31) is no longer supported, installed policy is always loaded. Continuing...
[ 857.819364] audit: type=1404 audit(1625387525.564:2): enforcing=1 old_enforcing=0 auid=4294967295 ses=4294967295 enabled=1 old-enabled=1 lsm=selinux res=1
[ 857.819384] SELinux: policy capability network_peer_controls=1
[ 857.819381] SELinux: policy capability open_perms=1
[ 857.819326] SELinux: policy capability extended_socket_class=1
[ 857.819336] SELinux: policy capability always_check_network=0
[ 857.819345] SELinux: policy capability cgroup_seclabel=1
[ 857.819353] SELinux: policy capability mnp_nosuid_transition=1
[ 857.826375] audit: type=1403 audit(1625387525.879:3): auid=4294967295 ses=4294967295 lsm=selinux res=1
[root@centos8 ~]# restorecon -v /etc/shadow
Relabeled /etc/shadow from system_u:object_r:unlabeled_t:s0 to system_u:object_r:shadow_t:s0
[root@centos8 ~]# ls -lZ /etc/shadow
----- 1 root root system_u:object_r:shadow_t:s0 2068 Jul 4 12:27 /etc/shadow
[root@centos8 ~]#
```

Но тут ничего ужасного нет и мы можем исправить. Мы помним, что для восстановления контекста из конфига selinux есть утилита `restorecon`:

```
restorecon -v /etc/shadow
ls -lZ /etc/shadow
```

Но, как видите, ничего не произошло - потому что у нас selinux не запущен. И вот тут нам нужна команда `load_policy` - она загружает политики:

```
load_policy -i /etc/selinux/targeted/policy/policy.31
```

И вот после неё `restorecon` всё возвращает:

```
restorecon -v /etc/shadow
ls -lZ /etc/shadow
```

И чтобы не терять контекст, мы сразу после `su` загрузили политики. Как бы вы не делали, просто проследите за тем, чтобы контекст файла был порядком.

```
load_video
set gfx_payload=keep
insmod gzio
linux ($root)/vmlinuz-4.18.0-240.15.1.el8_3.x86_64 root=/dev/mapper/cl_centos8\ 
-root ro crashkernel=auto resume=/dev/mapper/cl_centos8-swap rd.lvm.lv=cl_cen\ 
tos8/root rd.lvm.lv=cl_centos8/swap rd.break_
initrd  ($root)/initramfs-4.18.0-240.15.1.el8_3.x86_64.img $tuned_initrd
```

Теперь попробуем разобрать популярные способы из интернета. Где-то что-то отличается по мелочам, поэтому я покажу что-то среднее. Так или иначе, везде нужно редактировать в `grub`-е строчку `linux`.

Многие вместо вышеуказанных изменений пишут `rd.break` - тогда у нас процесс запуска останавливается ещё на initramfs.

```
Entering emergency mode. Exit the shell to continue.
Type "journalctl" to view system logs.
You might want to save "/run/initramfs/rdsosreport.txt" to a USB stick or /boot
after mounting them and attach it to a bug report.

switch_root:/# ls /bin /sbin
/bin:
awk          dracut-pre-mount    kbd_mode   mv        sort      teamnl
bash         dracut-pre-pivot    kmod       ping      stat      tr
cat          dracut-pre-trigger less        Plymouth  stty      true
chown        dracut-pre-udev    ln         ps        systemctl udevadm
cp           echo              loadkeys   readlink systemd-cgls umount
dmesg        findmnt          logindctl  rm       systemd-detect-virt uname
dracut-cmdline flock            ls         sed      systemd-escape vi
dracut-cmdline-ask gawk           mkdir     setfont   systemd-run
dracut-emergency grep             mknod    setsid   systemd-tmpfiles
dracut-initqueue gzip            mount    sh       teamd
dracut-mount  journalctl        sleep    sleep   teamdctl

/sbin:
arping        dmeventd      ifup        loginit   netroot   reboot   xfs_metadump
biosdevname   dmsetup       init       losetup  nologin  rmmod   xfs_repair
blkid         e2fsck       initqueue  insmod   lsmod    ping    rngd
chroot        fsck         insmod    lvm      ping6    swapoff
depmod        fsck.ext4    insmodpost.sh lvm_scan plymouthd tracekmem
dhclient      fsck.xfs    ip        modinfo  poweroff udevadm
dhclient-script halt        kexec    modprobe rdsosreport xfs_db
switch_root:/# _
```

В принципе, это полезная опция, которая позволяет решить некоторые проблемы, если не грузится корень. В initramfs обычно утилит мало:

```
ls /bin /sbin
```

но их может хватить на базовые операции для решения проблем, скажем, для проверки и исправления проблем с файловой системой, lvm и т.п.

```
switch_root:/# mount | grep /sysroot
/dev/mapper/cl_centos8-root on /sysroot type xfs (ro,relatime,attr2,inode64,logbufs=8,logbsize=32k,n
oquota)
switch_root:/# mount -o remount,rw /sysroot/
switch_root:/# chroot /sysroot/
sh-4.4# su -
[root@centos8 ~]# passwd
Changing password for user root.
New password:
BAD PASSWORD: The password is a palindrome
Retype new password:
passwd: all authentication tokens updated successfully.
[root@centos8 ~]#
```

Настоящий же корень примонтирован в директорию `/sysroot`:

```
mount | grep /sysroot
```

И, обратите внимание, что он примонтирован в режиме `ro` - read only. Поэтому мы изначально в grub меняем `ro` на `rw`. Тут же придётся перемонтировать:

```
mount -o remount,rw /sysroot
```

Дальше нам необходимо перейти с временного корня на настоящий, для этого есть утилита `chroot`:

```
chroot /sysroot
```

После чего надо залогиниться и задать новый пароль:

```
su -
passwd
```

```
[root@centos8 ~]# touch /.autorelabel
[root@centos8 ~]#
```

И мы помним, что это действие сбрасывает контекст с файла /etc/shadow. И для решения этой проблемы советуют создать файл в корне:

```
touch /.autorelabel
```

и перезагрузиться. При виде такого файла при запуске selinux восстанавливает контекст всех файлов. Но это долгий процесс и зависит от количества файлов в системе.

```
load_video
set gfx_payload=keep
insmod gzio
linux ($root)/vmlinuz-4.18.0-240.15.1.el8_3.x86_64 root=/dev/mapper/cl_centos8\
-root ro crashkernel=auto resume=/dev/mapper/cl_centos8-swap rd.lvm.lv=cl_cen\
tos8/root rd.lvm.lv=cl_centos8/swap autorelabel=1
initrd ($root)/initramfs-4.18.0-240.15.1.el8_3.x86_64.img $tuned_initrd
```

Правда у меня после создания файла контекст не восстановился и система отказывалась запускаться. Но если перезагрузиться и добавить в grub параметр autorelabel=1 - контекст восстановится. Потом понадобится ещё одна перезагрузка и всё заново заработает.

Ещё вас может заинтересовать вопрос - значит, любой желающий, не зная пароль, может его сбросить? Насколько это безопасно? Тут два варианта. Если у человека есть доступ только к консоли виртуалки, то можно поставить пароль на grub - тогда в меню grub при попытке редактирования будет запрашиваться пароль. Но если у человека есть физический доступ к компьютеру, то единственный способ защититься - это шифровать диски, иначе злоумышленник просто загрузится в livecd. Можно конечно блокировать паролем загрузочное меню компьютера, но и это можно обойти просто вытащив диск и подключив к другому компьютеру.

Подведём итоги. Сегодня мы с вами разобрали, как восстановить доступ к системе, если вы не знаете пароль root-а или другого пользователя с правами sudo. Также мы научились останавливать процесс запуска ещё на этапе initramfs - а это очень важно для решения проблем.

2.48.2 Практика

Вопросы

- Можно ли сбросить root пароль в рабочей системе?
- Зачем при восстановлении доступа менять параметр загрузки с `ro` на `rw`?
- Как изменится алгоритм сброса пароля при использовании SELinux

Задания

- Восстановите пароль пользователя root, не имея доступа к системе.

2.49 49. Виртуальная память, swap

2.49.1 49. Виртуальная память, swap



Представьте себе, что вы находитесь в библиотеке. Полки - это ваш жёсткий диск, книги - это файлы, а вы - ядро операционной системы. Перед вами есть стол - это оперативная память. Ваша работа - читать книги. Только, как и в реальной жизни, вы не можете читать книгу целиком - вы читаете страницами. Когда вы берёте из полки книжку, вы копируете нужные страницы и кладёте их на стол.



Т.е. на столе у вас разложены страницы. И вы, как Цезарь, многозадачны, читаете множество книг одновременно. Когда стол большой, т.е. когда много оперативки - всё хорошо, вы можете разместить кучу страниц. Но большой стол стоит больших денег, поэтому обычно вы берёте средненький стол, рассчитанный под ваши задачи, ну и чтобы оставалось немного свободного места.

Есть определённые книжки, которые вам нужны не постоянно, но вы с ними работаете относительно часто. Скажем, вы читаете книжку, там много неизвестных терминов и вам периодически нужен словарь. Не то чтобы он нужен всегда на столе, но если у вас есть немного места, то почему бы временно не положить словарь на стол? Чтобы постоянно не тратить время на копирование словаря. А если понадобится место под другие книжки - ничего, уберёте словарь.

[user@centos8 ~]\$ free -m	total	used	free	shared	buff/cache	available
Mem:	1818	1387	127	13	303	261
Swap:	2047	268	1779			

Если посмотреть вывод команды:

```
free -m
```

можно увидеть значение - столько-то места сейчас используется для буфера/кэша. Когда вы хотите прочесть файл с файловой системы, вы получаете блоки. Но в оперативке вы работаете не с блоками, а со страницами, т.е. вам нужно предварительно превратить содержимое блоков в страницы.

```
[user@centos8 ~]$ cat /proc/meminfo
MemTotal:       1862600 kB
MemFree:        188532 kB
MemAvailable:   395280 kB
Buffers:         124 kB
Cached:        342740 kB
```

И вот этот переходный пункт, где вы блоки превращаете в страницы, а также записываете за какой страницей какой блок - называется буфером. Т.е. буфер содержит метаданные файловой системы и блоки, которые пишутся иличитываются с диска. А в кэше хранятся уже страницы. И эти страницы нужны как для более быстрого чтения, так и для более производительной работы. Представьте, что вам нужно несколько раз прочесть одну и ту же страницу. Вместо того, чтобы каждый раз загружать её в память, вы её держите в оперативке и считываете её оттуда.

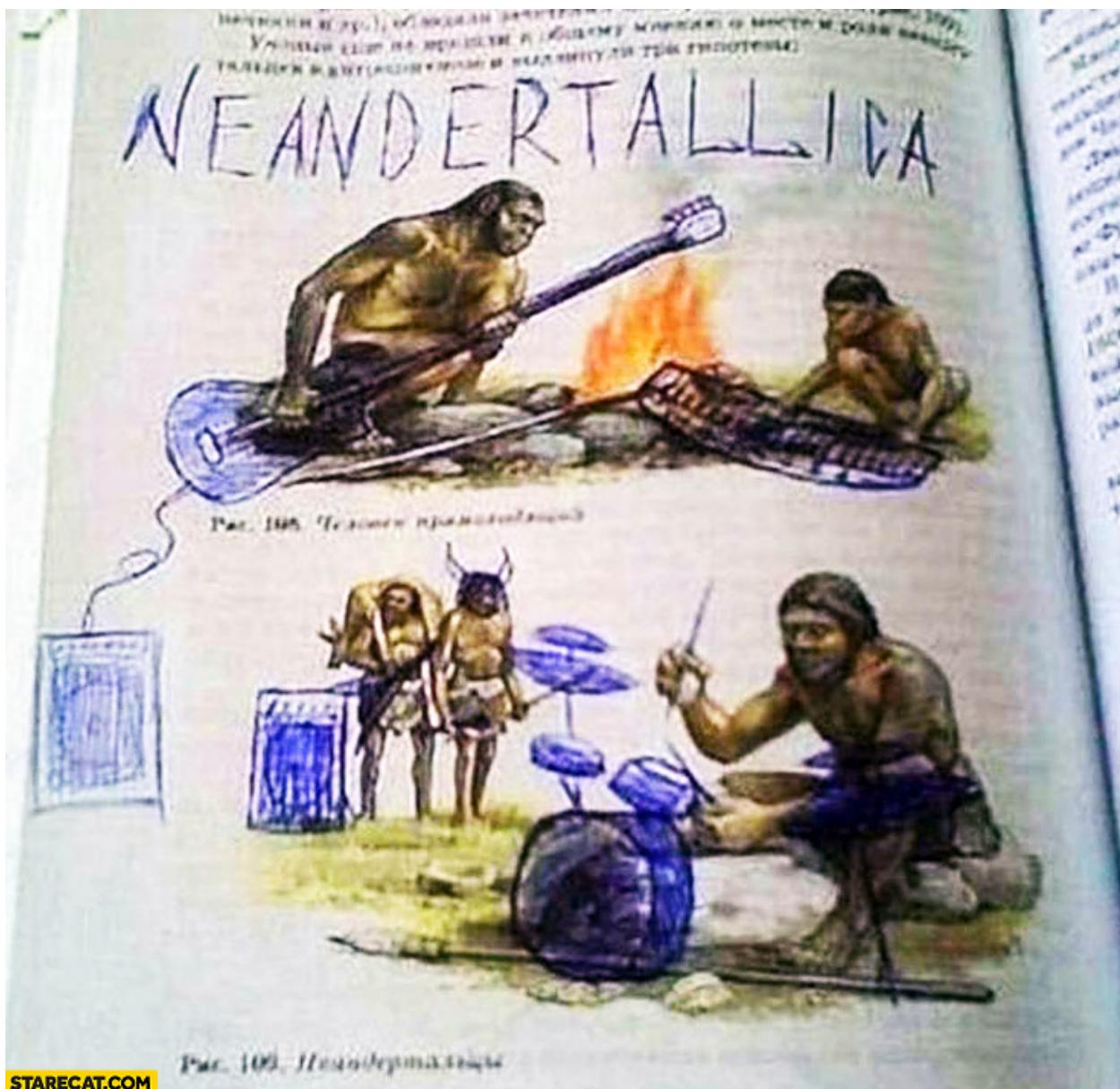


Рис. 109. Неандертальцы

STARECAT.COM

А касательно записи, представьте, что вам нужно что-то дописать в словаре. Если один раз - написали на листе, отнесли и добавили в книгу. А если вы часто что-то пишете? Вместо того, чтобы после каждой строчки относить и добавлять в книгу, вы можете подержать страницу пока на столе - а может ещё что-то понадобится дописать? Когда вы изменяете страницу в оперативке, у неё появляется метка, что она грязная - dirty page. И уже потом, скажем, раз в 5 секунд, вы относите грязные листы и добавляете в книги, т.е. сохраняете изменённые данные с оперативки на диск.

```
[doctor@tardis]—[~]
└─$ cat dirty
sync
grep Dirty /proc/meminfo
dd if=/dev/urandom of=testfile bs=1M count=100
grep Dirty /proc/meminfo
sync
grep Dirty /proc/meminfo
[doctor@tardis]—[~]
└─$ ./dirty
Dirty:          0 kB
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 0.404884 s, 259 MB/s
Dirty:      102328 kB
Dirty:      204 kB
[doctor@tardis]—[~]
└─$
```

Т.е. данные не сразу пишутся на диск, а некоторое время хранятся в оперативке, в кэше. И чтобы вручную синхронизировать изменённые в оперативке данные с диском, надо выполнить команду sync. Размер грязных страниц можно посмотреть в /proc/meminfo:

```
grep Dirty /proc/meminfo
```

И вот небольшой тест с созданием файла. Сначала делаем sync, чтобы убедиться, что все данные записаны на диск. Затем с помощью dd генерируем файл размером в 100 мегабайт. Сразу после этого смотрим размер грязных страниц - 100 мб. Это означает, что файл пока что в оперативке, а не на диске. Делаем sync и снова проверяем - теперь всё записалось на диск.

Такой режим кэша, когда данные сначала сохраняются в оперативке, а потом пишутся на диск, называется writeback. Но если вдруг компьютер потеряет питание и информация не успеет синхронизироваться, грязные страницы пропадут. Соответственно, есть риск потерять данные. Т.е. такой режим очень быстрый, но не самый надёжный. Также есть режим кэша writethrough - когда данные пишутся одновременно и на диск. Т.е. условно при каждом изменении страницы вы относите страницу в книгу на полку. Так всё работает чуть медленнее, зато при внезапном выключении вы ничего не теряете. Есть и другие режимы, но это два основных.

	Private	Shared
Anonymous	1 stack malloc() mmap(ANON, PRIVATE) brk()/sbrk()	2 mmap(ANON, SHARED)
File-Backed	3 mmap(fd, PRIVATE) binary/shared libraries	4 mmap(fd, SHARED)

Память можно поделить на 4 типа по двум категориям. Она может быть личной и общей - т.е. доступна только для одного процесса или нескольких. А также может быть анонимной и основанной на файле, т.е. файловой. File-backed - это когда вы загружаете файл в оперативку - взяли книжку и положили листы на стол. А анонимная - это когда процесс просит вас выделить ему место, не привязанное к какому-либо файлу. Скажем, не книжку взять, а просто чистый А4 положить. Процесс там будет хранить какие-то данные во время работы. Таким образом получается 4 типа - anonymous private, anonymous shared, file-backed private и file-backed shared.



И теперь представим, что у вас на столе осталось мало места, а вам надо ещё добавить листов, т.е. мало оперативки осталось. Первым делом вы поищете листы, которые вам больше не нужны - т.е. освободите память, которую использовали завершённые процессы. Но этого может не хватить. Тогда в расход пойдут старые листы, которые как бы и нужны, но используются очень редко и основаны на книжках - т.е. файловые. Вы избавитесь от этих страниц, а если вдруг они потом понадобятся - то опять возьмёте их из книг, так как вы знаете, что это за файл. А вот от анонимных листов избавиться не получится. То что там написано - нет ни в одной книге. Поэтому выбрасывать эти листы не вариант.

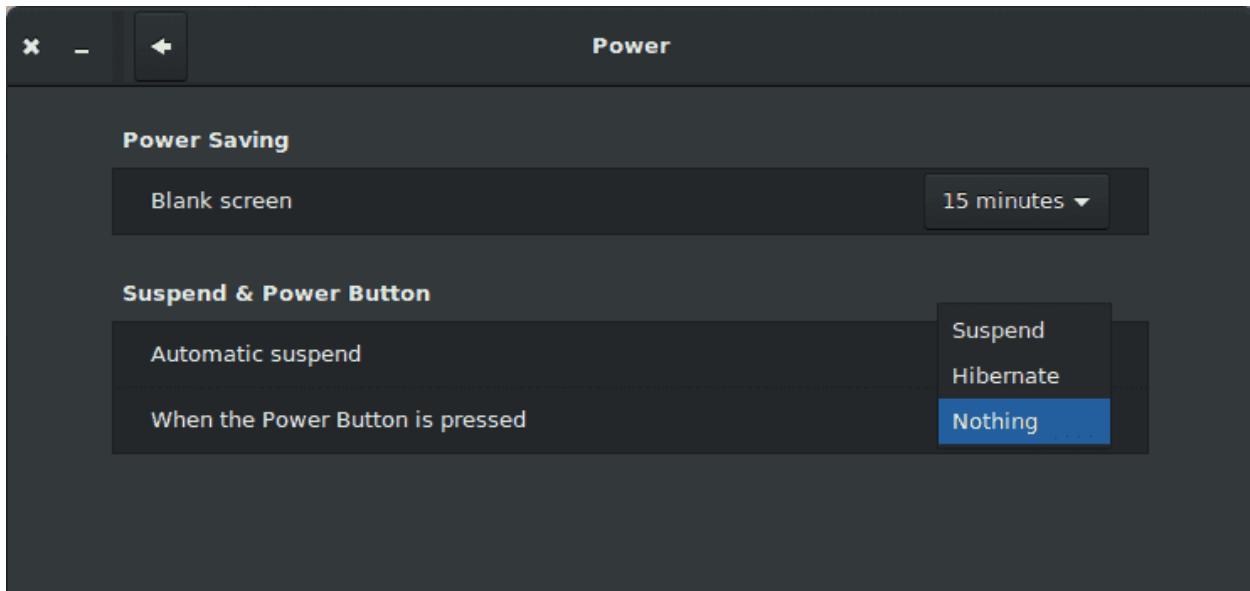


И тут вам на помощь приходят ящики стола. Туда вы можете сложить старые листы, которые сейчас

на столе не нужны, а при необходимости оттуда достать. Это ваш swap - место на диске, выделенное для отгрузки редкоиспользуемых страниц с оперативки. Этот механизм, когда вы что-то кладёте в swap или достаёте оттуда называется подкачка страниц. В качестве места может быть как отдельный файл на файловой системе, так и целый раздел, в зависимости от этого место может называться файл подкачки или раздел подкачки.

Т.е. swap не выступает продолжением оперативки, он выступает времененным хранилищем для редкоиспользуемых страниц. И не обязательно, чтобы он использовался только когда оперативки не хватает. Иногда полезней использовать оперативку для кэша, чем для редкоиспользуемых страниц.

Чтобы вся эта подкачка работала для программ прозрачно, чтобы разработчики сами не заботились о swap-е, ядро ОС выделяет процессам вместо реальной памяти виртуальную, которая может быть больше оперативки. Виртуальная память состоит из оперативки и swap-а. Т.е. условно, книжка может предварительно попросить места под 100 листов на столе. И пусть у вас стол будет поменьше, но, если что, редкоиспользуемые страницы вы добавите в ящик.



Также swap используется для гибернации. Это как спящий режим, только, если в спящем режиме у вас всё оборудование начинает меньше потреблять энергии, но всё ещё потребляет по чуть-чуть, то при гибернации все данные из оперативки сохраняются в swap-е, а компьютер полностью выключается, т.е. ничего не потребляет. А при запуске всё из swap-а возвращается в оперативку. Это позволяет больше сэкономить энергию, но запуск чуть дольше, чем при спящем режиме.

Пара популярных вопросов:

- нужен ли swap, если много оперативки?

Если большой излишек оперативки, т.е. много много оперативки не тратится - то можно обойтись и без swap-а. С другой стороны swap даже на таких системах может пригодится - будет чуть больше места для кэша.

- стоит ли делать swap на ssd?

С одной стороны, из-за скорости подкачка страниц будет быстрее, что увеличит производительность. С другой стороны, у ssd есть ограниченное число циклов перезаписи, и использование swap-а активно их расходует. Но, если у вас оперативки достаточно - подкачка будет происходить довольно редко. Ну и даже с относительно средним использованием, современные SSD проживут довольно долго, а их цена не такая кусачая. Точные сроки никто назвать не сможет, так как это индивидуально для конкретной системы и SSD.

- раздел подкачки или файл подкачки - что лучше?

Раньше из-за прослойки с файловой системой swap-файл был чуть помедленнее, но ядро после версии 2.6 работает с блоками swap-файла минуя файловую систему. С одной стороны пропала разница в производительности с разделом подкачки, с другой - нельзя просто взять и переместить файл подкачки на другую файловую систему, swap надо предварительно отключить. Но файл обычно легче увеличить, чем раздел, хотя если swap-раздел на LVM и есть свободное место - тоже несложно.

- сколько места выделять под swap?

Amount of RAM in the system	Recommended swap space	Recommended swap space if allowing for hibernation
≤ 2 GB	2 times the amount of RAM	3 times the amount of RAM
> 2 GB – 8 GB	Equal to the amount of RAM	2 times the amount of RAM
> 8 GB – 64 GB	At least 4 GB	1.5 times the amount of RAM
> 64 GB	At least 4 GB	Hibernation not recommended

Тут немного индивидуально - зависит от софта, который будет использоваться. Какие-то программы требуют больше swap-а, какие-то меньше. Но есть общие рекомендации - если у вас оперативки меньше 2 гигабайт - то под swap стоит выделить места в два раза больше, чем в оперативке. Если ещё и предполагаете использование гибернации - то в 3 раза больше. Ну и дальше по табличке.

```
[user@centos8 ~]$ free -m
              total        used        free      shared  buff/cache   available
Mem:       1818         954          90          2        774        703
Swap:      2047           4       2043
[user@centos8 ~]$ swapon
NAME      TYPE      SIZE USED PRIO
/dev/dm-1 partition  2G 4.3M   -2
[user@centos8 ~]$ sudo blkid | grep swap
[sudo] password for user:
/dev/mapper/cl_centos8-swap: UUID="753af890-d991-4100-aeb0-3a2e7de9e7b6" TYPE="swap"
[user@centos8 ~]$
```

Хорошо, что такое swap и для чего он нужен мы разобрались. Насколько используется swap мы можем увидеть с помощью утилиты free, либо утилиты swapon. Как видно, у нас для swap-а используется отдельный раздел, а точнее lvm:

```
sudo blkid | grep swap
```

Его размер - 2 гига, а используется небольшой процент. Кроме того видно, что у swap-а есть приоритет.

Мы можем на одной системе сделать несколько swap разделов и файлов. Т.е. сначала будем складывать страницы в ящичек с высоким приоритетом, а если он забьётся - в ящик с приоритетом пониже. Скажем, мы можем выделить под swap место как на ssd, так и на жёстком диске. Также можно приоритет поставить одинаковым - тогда одновременно будут использоваться оба swap-а.

```
[user@centos8 ~]$ sudo dd if=/dev/zero of=/swapfile bs=1K count=2M
[sudo] password for user:
2097152+0 records in
2097152+0 records out
2147483648 bytes (2.1 GB, 2.0 GiB) copied, 3.26206 s, 658 MB/s
[user@centos8 ~]$ du -h /swapfile
2.0G  /swapfile
[user@centos8 ~]$ sudo mkswap /swapfile
mkswap: /swapfile: insecure permissions 0644, 0600 suggested.
Setting up swapspace version 1, size = 2 GiB (2147479552 bytes)
no label, UUID=8390d92c-3826-4900-a046-865db94c2ac8
[user@centos8 ~]$ sudo chmod 600 /swapfile
[user@centos8 ~]$ █
```

Попробуем добавить файл подкачки и перевести наш swap с раздела на файл. Для начала создадим файл размером в 2 гигабайта, для этого используем команду dd:

```
dd if=/dev/zero of=/swapfile bs=1K count=2M
```

Здесь bs - размер блока - 1Килобайт; 2M - 2 миллиона. Т.е. 2 миллиона блоков по 1 килобайту - получается 2 гигабайта. Помните, как после создания раздела мы записывали на него файловую систему - mkfs? Для swap-а мы делаем что-то похожее:

```
sudo mkswap /swapfile
```

Был бы раздел - указали бы название раздела. Команда вывела нам предупреждение, что у файла не безопасные права - 644, и рекомендуется выставить 600. Что ж, сделаем:

```
sudo chmod 600 /swapfile
```

```
[user@centos8 ~]$ sudo swapon -p 1 /swapfile
[sudo] password for user:
[user@centos8 ~]$ swapo
swapoff swapon
[user@centos8 ~]$ swapon
NAME      TYPE      SIZE   USED  PRI0
/dev/dm-1  partition  2G  20.3M    -2
/swapfile  file        2G     0B     1
[user@centos8 ~]$ █
```

И дальше, по аналогии того, как мы монтировали файловую систему, мы должны активировать swap, заодно зададим ему приоритет повыше:

```
sudo swapon -p 1 /swapfile
```

После этого посмотрим активные swap-ы - swapon. В списке появился наш файл.

```
GNU nano 2.9.8 /etc/fstab
# units generated from this file.
#
/dev/mapper/cl_centos8-root / xfs defaults 0 0
UUID=12054851-7d99-4c7c-be08-4ee52bd0cc6e /boot ext4 defaults 0 0
#/dev/mapper/cl_centos8-swap swap swap defaults 0 0
/dev/mapper/myraidvg-myraidlv /mydata ext4 defaults 0 0

/swapfile swap swap pri=1 0 0
```

Но, как и с файловыми системами, чтобы swap работал после перезагрузки, надо его добавить в fstab:

```
sudo nano /etc/fstab
```

Пропишем наш swap-file, а заодно закомментируем старый раздел.

```
[user@centos8 ~]$ sudo swapoff /dev/dm-1
[user@centos8 ~]$ swapon
NAME      TYPE SIZE USED PRI0
/swapfile file 2G   84K   1
[user@centos8 ~]$
```

Чтобы без перезагрузки избавиться от swap раздела, используем команду swapoff:

```
sudo swapoff /dev/dm-1
swapon
```

Таким образом мы убираем все страницы из раздела обратно в память или на другой swap.

```
[user@centos8 ~]$ sysctl vm.swappiness
vm.swappiness = 30
[user@centos8 ~]$
```

Также стоит упомянуть такой параметр, как swappiness:

```
sysctl vm.swappiness
```

Этот параметр определяет, насколько сильно будет задействован swap. Тут значения от 0 до 100 - это баланс между кэшем и выгрузкой анонимных страниц. При низких значениях swap будет стараться

держать анонимные страницы как можно дольше в памяти, при этом чаще выгружая кэш. Это может быть полезно, если программа сама управляет своим кэшем, а не отдаёт это ядру операционной системы. Например, так делают системы управления базами данных. Если же значение ближе к 100, то система агрессивнее будет заниматься подкачкой - перемещать анонимные страницы в swap и обратно. Это обычно системы, где программы сами не кэшируют, но кэш важнее. Например, частая работа с данными, при этом производительностью программ можно немного пожертвовать.

```
[user@centos8 ~]$ sudo nano /etc/sysctl.d/99-sysctl.conf
[user@centos8 ~]$ sudo grep swapp /etc/sysctl.d/99-sysctl.conf
vm.swappiness=35
[user@centos8 ~]$ sudo sysctl -p
vm.swappiness = 35
[user@centos8 ~]$ █
```

Если мы хотим поменять это значение и чтобы оно оставалось после перезагрузки, нужно создать файл или подредактировать существующий в директории `/etc/sysctl.d/`, прописав в нём `vm.swappiness` и нужное значение, после чего применить:

```
sudo sysctl -p
```

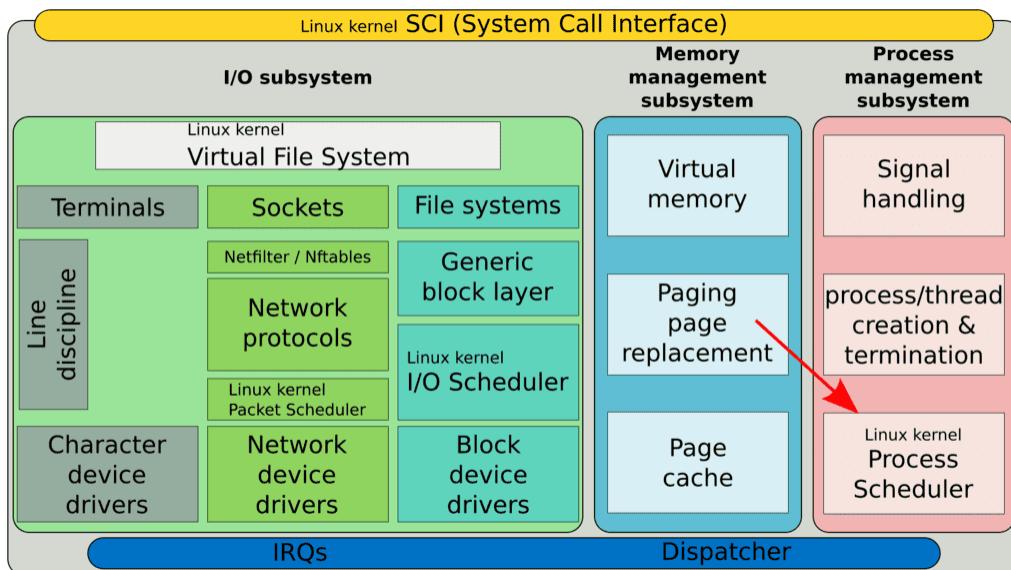
```
[user@centos8 ~]$ dmesg | grep zswap
[    0.447020] zswap: loaded using pool lzo/zbud
[user@centos8 ~]$ sudo grep -R . /sys/kernel/debug/zswap
/sys/kernel/debug/zswap/same_filled_pages:0
/sys/kernel/debug/zswap/stored_pages:0
/sys/kernel/debug/zswap/pool_total_size:0
/sys/kernel/debug/zswap/duplicate_entry:0
/sys/kernel/debug/zswap/written_back_pages:0
/sys/kernel/debug/zswap/reject_compress_poor:0
/sys/kernel/debug/zswap/reject_kmemcache_fail:0
/sys/kernel/debug/zswap/reject_alloc_fail:0
/sys/kernel/debug/zswap/reject_reclaim_fail:0
/sys/kernel/debug/zswap/pool_limit_hit:0
[user@centos8 ~]$ █
```

Также стоит упомянуть, что существует модуль ядра zswap. Он создаёт в оперативной памяти сжатую область, где и хранится swap. Это позволяет подкачке меньше использовать диск и работать быстрее. Но если места в оперативке нет - данные всё же выгружаются на swap, который на диске.

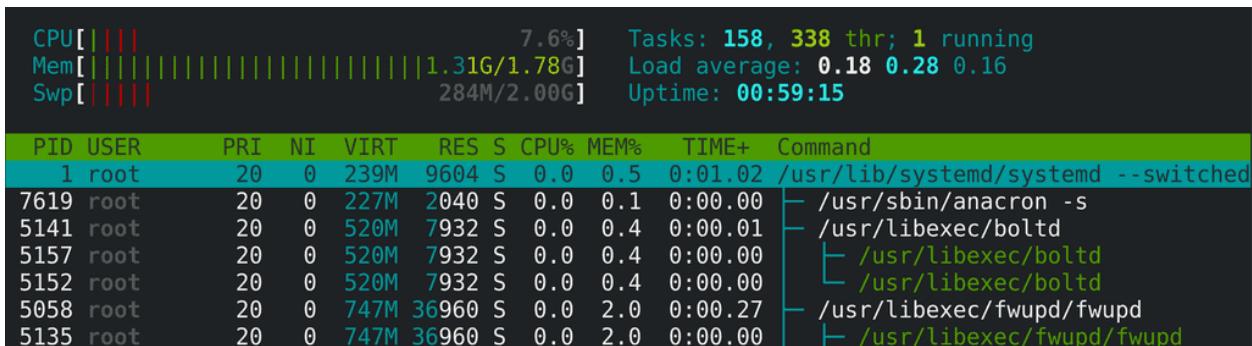
Подведём итоги. Сегодня мы с вами разобрали механизмы работы виртуальной памяти - что такое кэш, что такие грязные страницы, анонимные страницы, разобрались, зачем нужна подкачка, как она работает, как создавать раздел или файл подкачки и как это всё настраивать. Со swap-ом работают относительно редко - один раз настраивается во время установки системы, а больше делать ничего и не надо. Поэтому очень важно сразу всё настроить правильно.

2.50 50. Планировщик процессов

2.50.1 50. Планировщик процессов

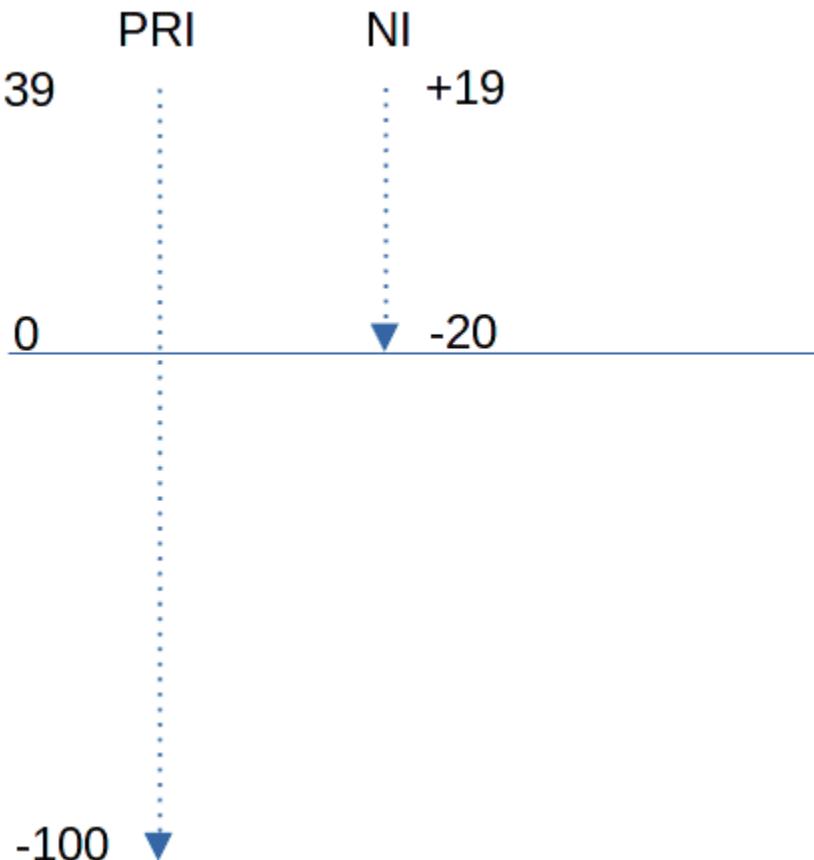


В первой теме мы с вами обсуждали, что одной из главных причин появления операционных систем является необходимость в разделении времени - time sharing. И одна из главных функций ядра операционной системы - планировщик процессов, который и решает, когда и какой процесс должен быть обработан процессором.



С планировщиком, который в текущей реализации называется «Полностью честный планировщик» - мы уже взаимодействовали, когда задавали процессам вежливость. Чем вежливее процесс, тем меньше процессорных ресурсов ему в итоге достаётся. Т.е. это функционал планировщика. Так вот, вежливость влияет на приоритет процесса. Но сразу стоит упомянуть, что не всех процессов. Для нормальных процессов приоритет имеет значение:

$$\text{PRI} = \text{NI} + 20$$



Мы помним, что nice ness задаётся от -20 до 19, т.е. имеет 40 значений. Приоритет же имеет значения от -100 до 39, притом, от 0 до 39 - это те приоритеты, которые мы обычно видим и можем задать с помощью nice. Чем ниже значение, тем больше приоритет. То что ниже 0 - выделено для более важных процессов, которые нельзя ставить в один ряд с пользовательскими процессами.

PID	USER	PRI	NI	VIRT	RES	S	CPU%	MEM%	TIME+	Command
1052	rtkit	RT	1	188M	3572	S	0.0	0.2	0:00.01	/usr/libexec/rtkit-daemon
1300	root	2	-18	130M	26812	S	0.0	1.4	0:00.05	/usr/sbin/dmeventd -f
3532	user	9	-11	1271M	8772	S	0.0	0.5	0:00.03	/usr/bin/pulseaudio --daemonize=no
811	root	16	-4	139M	1860	S	0.0	0.1	0:00.01	/sbin/auditd
817	root	16	-4	139M	1860	S	0.0	0.1	0:00.00	/sbin/auditd
816	root	16	-4	47848	1940	S	0.0	0.1	0:00.00	/usr/sbin/sedispatch

Например, миллисекундные зависания браузера могут быть неощутимы, а те же миллисекунды в звуке будут гораздо неприятнее. Как бы не была важна пользовательская программа, есть процессы, приоритет которых должен быть выше, т.е. ближе к нулю. Тор для таких процессов показывает значение RT - real time.

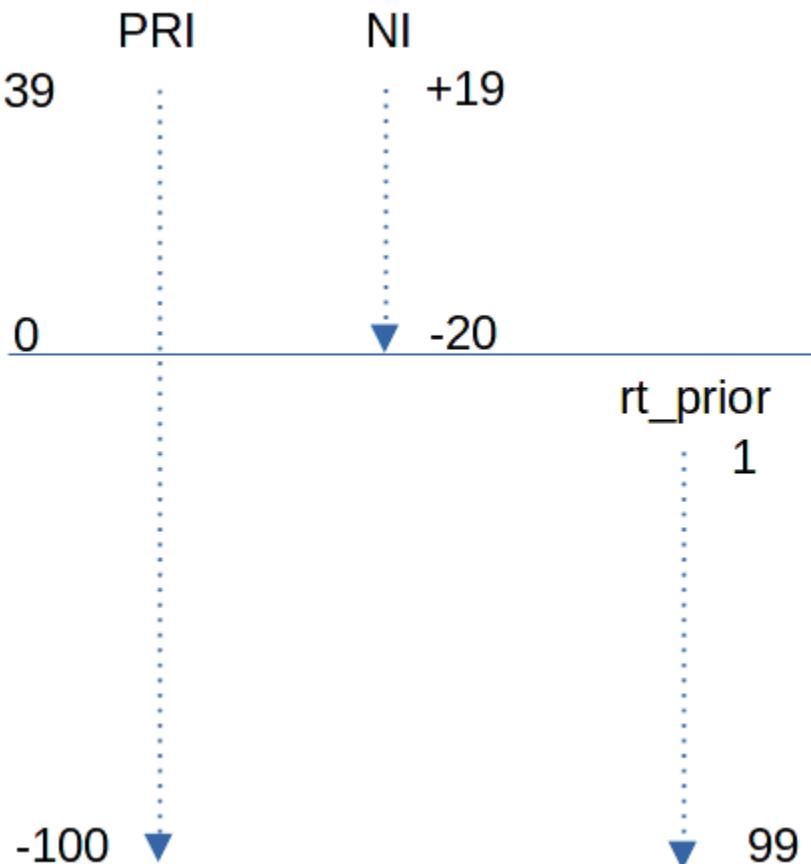
```
[user@centos8 ~]$ htop
[user@centos8 ~]$ chrt -p 1052
pid 1052's current scheduling policy: SCHED_RR|SCHED_RESET_ON_FORK
pid 1052's current scheduling priority: 99
[user@centos8 ~]$ █
```

С помощью утилиты chrt мы можем посмотреть rt приоритет таких процессов:

```
chrt -p 1052
```

Как видите, текущее значение - 99. Но rt приоритет не тоже самое, что обычный приоритет. Чтобы узнать итоговый приоритет, надо от -1 отнять rt значение:

```
PRI = -1 - rt_prior
```



Т.е. в таком случае высокое значение rt приоритета равно низкому значению самого приоритета. А чем ниже значение приоритета, тем сам приоритет выше. Т.е. процесс с rt приоритетом 99 имеет наивысший приоритет - -100.

```
[user@centos8 ~]$ chrt -m
SCHED_OTHER min/max priority      : 0/0
SCHED_FIFO min/max priority       : 1/99
SCHED_RR min/max priority        : 1/99
SCHED_BATCH min/max priority     : 0/0
SCHED_IDLE min/max priority      : 0/0
SCHED_DEADLINE min/max priority  : 0/0
[user@centos8 ~]$ █
```

Но планировщик в ядре распределяет процессор не только по приоритету, но и по политике. Список политик можно увидеть с помощью ключа `-m`:

```
chrt -m
```

Все процессы подчиняются каким-то политикам. На самом деле приоритет относится не к процессам, а к их потокам, но чтобы было легче всё это воспринимать, будем считать, что один процесс - это один поток.



Представьте это как фейс контроль в какой-нибудь клуб, только в клубе можно находить не постоянно, а зайти на время, потанцевать, после чего выйти и опять встать в очередь. Бугай на входе - планировщик - берёт какое-то время, допустим, час и делит на количество ожидающих, допустим, 10. Таким образом каждый в час может потанцевать 6 минут. Но тем, у кого приоритет выше, достаётся больше времени и они танцуют 8 минут. А более вежливые танцуют 2-3 минуты. Это касается нормальных процессов и для них обычно используется политика `SCHED_OTHER`.

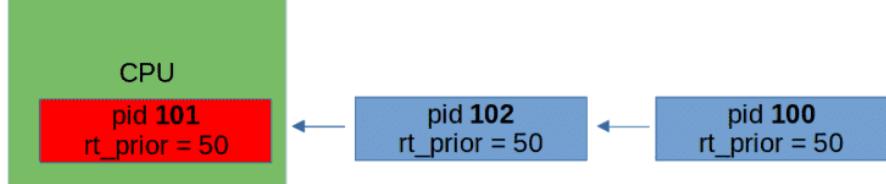
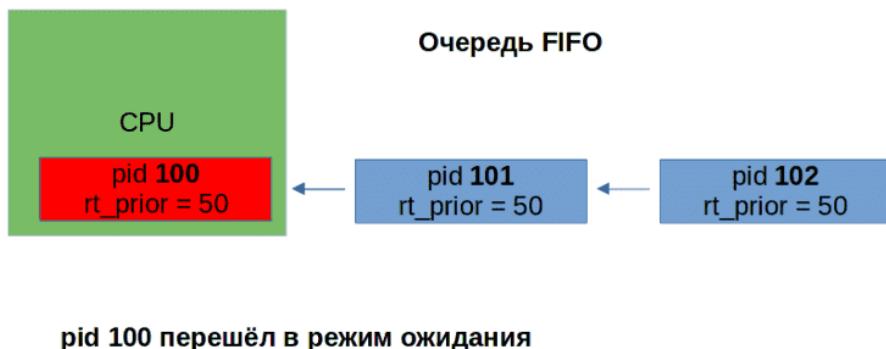
Но есть VIP персоны «RT» - и они не стоят в общей очереди, охранник пропускает их раньше всех. Но и самих VIP персон может быть несколько и они в том числе могут разниться. Скажем, есть музыканты,

а есть известные личности. Музыкантов пускают по одной политике, а известных личностей по другой. Так или иначе, и те и другие важнее обычных людей и их пропускают первыми. Но и среди музыкантов, и среди известных личностей есть rt приоритет.

```
[user@centos8 ~]$ chrt -m
SCHED_OTHER min/max priority : 0/0
SCHED_FIFO min/max priority : 1/99
SCHED_RR min/max priority : 1/99
SCHED_BATCH min/max priority : 0/0
SCHED_IDLE min/max priority : 0/0
SCHED_DEADLINE min/max priority : 0/0
[user@centos8 ~]$ █
```

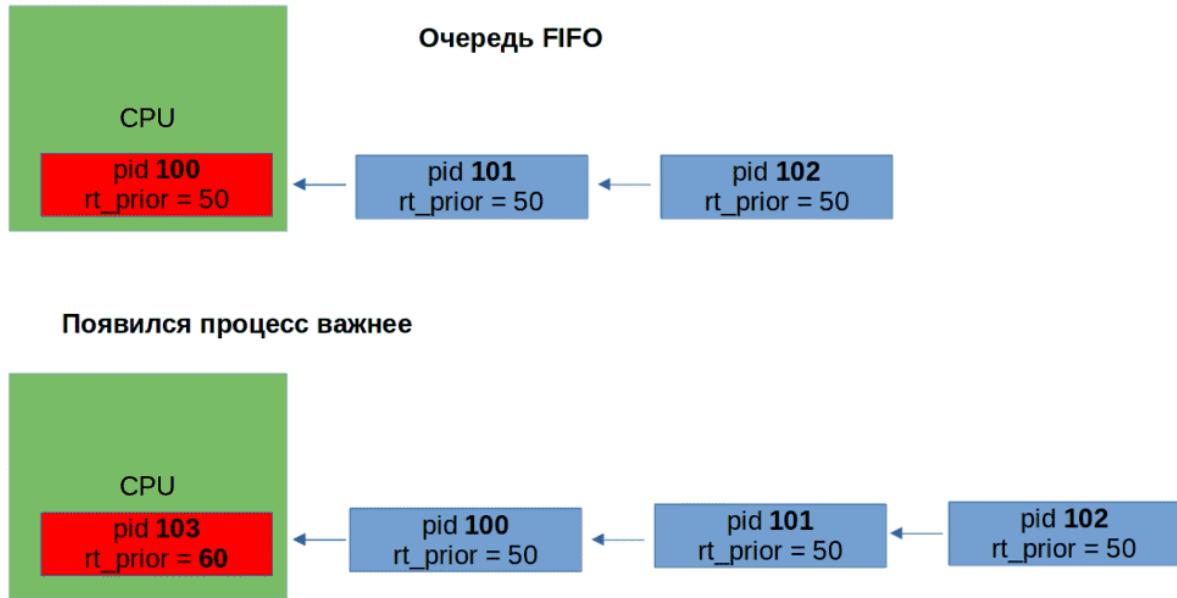
Так вот, есть две политики для rt приоритетов:

- FIFO - first in first out
- RR - round robin

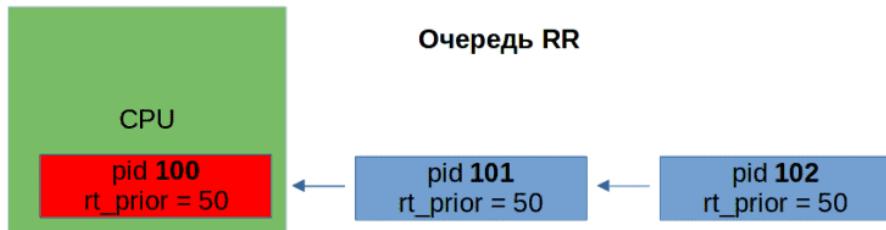


В случае с FIFO процесс будет работать, пока не остановится, скажем, в ожидании данных. В таком случае остановленный процесс помещается в конец списка таких же процессов с тем же приоритетом. Скажем, у вас есть 3 процесса - 100, 101, 102 с одинаковым rt приоритетом 50. Сотый запустился первым, поработал, остановился и стал третьим в списке rt процессов с приоритетом 50, т.е. теперь в

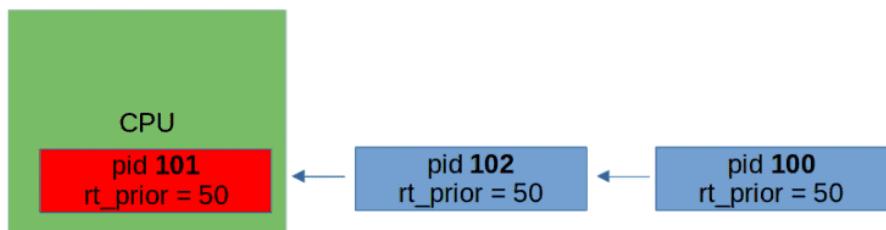
очереди 101, 102 и 100.



Однако если появляется процесс с rt приоритетом выше, допустим, 103, то 100-ый процесс перемещается в начало списка ожидающих, пока 103 работает.



У pid 100 закончилось время



В случае с round robin всё тоже самое, разве что процессам теперь время ограничено и они работают не сколько хотят, а только промежуток времени, после чего идут в конец очереди.

Есть и другие политики, и много всяких нюансов, но вдаваться в такие дебри пока не стоит. Достаточно в целом иметь представление, как это работает.

```
[user@centos8 ~]$ ps -ef | grep bash
root      1055      1  0 09:28 ?          00:00:00 /bin/bash /usr/sbin/ksmtuned
user      7255  7250  0 09:35 pts/0        00:00:00 bash
user     11504  7255  0 14:39 pts/0        00:00:00 grep --color=auto bash
[user@centos8 ~]$ sudo chrt -f -p 50 7255
[sudo] password for user:
[user@centos8 ~]$ chrt -p 7255
pid 7255's current scheduling policy: SCHED_FIFO
pid 7255's current scheduling priority: 50
```

PID	USER	PRI	NI	VIRT	RES	S	CPU%	MEM%	TIME+	Command
1052	rtkit	RT	1	188M	3568	S	0.0	0.2	0:00.04	/usr/libexec/rtkit-d
7255	user	-51	0	228M	5428	S	0.0	0.3	0:00.08	bash
11661	user	-51	0	234M	4624	R	0.8	0.2	0:00.16	htop
1300	root	2	-18	130M	26812	S	0.0	1.4	0:00.15	/usr/sbin/dmeventd -
3532	user	9	-11	1527M	8484	S	0.0	0.5	0:00.07	/usr/bin/pulseaudio

Так вот, утилита chrt позволяет менять политику для процесса и выставлять приоритет. Для примера возьмём процесс bash-а:

```
ps -ef | grep bash
```

С помощью chrt поменяем ему политику на FIFO и зададим rt приоритет 50:

```
sudo chrt -f -p 50 7255  
chrt -p 7255
```

Как видите, политика теперь FIFO и выставлен нужный rt приоритет. В htop видно, что общий приоритет -51, это потому что от -1 отнимаем 50. А также видно, что у дочернего процесса приоритет унаследовался, тоже -51.

```
[user@centos8 ~]$ sudo chrt -r -p 70 7255  
[sudo] password for user:  
[user@centos8 ~]$ chrt -p 7255  
pid 7255's current scheduling policy: SCHED_RR  
pid 7255's current scheduling priority: 70
```

PID	USER	PRI	NI	VIRT	RES	S	CPU%	MEM%	TIME+	Command
1052	rtkit	RT	1	188M	3568	S	0.0	0.2	0:00.04	/usr/libexec/rtkit-d
7255	user	-71	0	228M	5428	S	0.0	0.3	0:00.08	bash
11810	user	-71	0	234M	4624	R	1.5	0.2	0:00.13	htop
1300	root	2	-18	130M	26812	S	0.0	1.4	0:00.16	/usr/sbin/dmeventd -
3532	user	9	-11	1527M	8408	S	0.0	0.5	0:00.07	/usr/bin/pulseaudio
815	root	16	-4	139M	1860	S	0.0	0.1	0:00.00	/sbin/auditd

Теперь поменяем политику на Round Robin и выставим приоритет 70:

```
sudo chrt -r -p 70 7255  
sudo chrt -p 7255  
htop
```

Таким образом у нас политика round robin и приоритет -71.

```
[user@centos8 ~]$ sudo chrt -o -p 0 7255  
[user@centos8 ~]$ chrt -p 7255  
pid 7255's current scheduling policy: SCHED_OTHER  
pid 7255's current scheduling priority: 0  
[user@centos8 ~]$ █
```

Ну и вернём обратно политику OTHER. Так как тут нет rt приоритета, значение -p - 0:

```
sudo chrt -o -p 0 7255  
sudo chrt -p 7255
```

Насколько это часто используется в повседневной работе? Очень редко. Но в определённых случаях - при больших нагрузках - это позволит вам повысить приоритет важных для вас процессов куда больше, чем это делает nice. Но и в целом это полезно знать, чтобы лучше понимать работу операционной системы.

2.51 51. Оптимизация производительности - tuned

2.51.1 51. Оптимизация производительности - tuned

Мы с вами разобрали виртуальную память и планировщик процессов, за работу которых отвечает ядро операционной системы. Знаем, что если поменять параметр swappiness, можно задать насколько сильно будет задействован swap, а это влияет на производительность. Также мы говорили, что ядро отвечает за работу с устройствами посредством драйверов. Возможно вы знаете, что у различного оборудования, например, процессора, бывают различные режимы работы - энергосберегающий или режим высокой производительности. Сетевой адаптер может поддерживать различные скорости работы, скажем, 100Мбит или 1Гбит, и в зависимости от этого потребляет меньше или больше энергии. Это и многое другое можно настроить посредством изменения параметров ядра, а также специальных утилит.

Когда у вас одна система, вы можете это сделать и вручную, но для этого придётся изучить немало документации. На работе же у вас может быть множество систем и совсем мало времени на детальную настройку производительности. В большинстве случаев многие и не заботятся о производительности, так как расчёт на оборудование изначально берётся с запасом.

Set the parameters in the `/etc/sysctl.conf` file and reload with `sysctl -p`:

```
# kernel.shmall = _PHYS_PAGES / 2 # See Shared Memory Pages
kernel.shmall = 197951838
# kernel.shmmax = kernel.shmall * PAGE_SIZE
kernel.shmmax = 810810728448
kernel.shmmni = 4096
vm.overcommit_memory = 2 # See Segment Host Memory
vm.overcommit_ratio = 95 # See Segment Host Memory

net.ipv4.ip_local_port_range = 10000 65535 # See Port Settings
kernel.sem = 500 2048000 200 4096
kernel.sysrq = 1
kernel.core_uses_pid = 1
kernel.msgmnb = 65536
kernel.msgmax = 65536
kernel.msgmni = 2048
net.ipv4.tcp_syncookies = 1
net.ipv4.conf.default.accept_source_route = 0
net.ipv4.tcp_max_syn_backlog = 4096
net.ipv4.conf.all.arp_filter = 1
net.core.netdev_max_backlog = 10000
net.core.rmem_max = 2097152
net.core.wmem_max = 2097152
vm.swappiness = 10
vm.zone_reclaim_mode = 0
vm.dirty_expire_centisecs = 500
vm.dirty_writeback_centisecs = 100
vm.dirty_background_ratio = 0 # See System Memory
vm.dirty_ratio = 0
vm.dirty_background_bytes = 1610612736
vm.dirty_bytes = 4294967296
```

Правда на высокопроизводительных системах всё же нужно менять параметры ядра, иначе можно

столкнуться с проблемами с процессами, сетью, памятью и прочим. Но, обычно, в документации требуемого софта упоминаются необходимые параметры. По [ссылке](#) вы можете посмотреть пример, как производитель софта рекомендует рассчитать и выставить определённые параметры. Однако это специфичные настройки для определённой программы. Не для всех программ есть такие детальные рекомендации.

```
[user@centos8 ~]$ systemctl status tuned
● tuned.service - Dynamic System Tuning Daemon
  Loaded: loaded (/usr/lib/systemd/system/tuned.service; enabled; vendor prese>
  Active: active (running) since Wed 2021-08-18 10:50:09 +04; 1h 43min ago
    Docs: man:tuned(8)
          man:tuned.conf(5)
          man:tuned-adm(8)
   Main PID: 1084 (tuned)
     Tasks: 4 (limit: 11334)
    Memory: 16.0M
   CGroup: /system.slice/tuned.service
           └─1084 /usr/libexec/platform-python -Es /usr/sbin/tuned -l -P

Aug 18 10:50:07 centos8 systemd[1]: Starting Dynamic System Tuning Daemon...
Aug 18 10:50:09 centos8 systemd[1]: Started Dynamic System Tuning Daemon.
[user@centos8 ~]$ █
```

Если вы задумываетесь об оптимизации производительности в ваших системах, вам может помочь демон tuned:

```
systemctl status tuned
```

Он может быть полезен как для тех, кто совсем не разбирается в этой теме, так и для тех, кто знает что к чему. Если вкратце, tuned предоставляет готовые профили, в которых указаны нужные для производительности параметры, а также даёт возможность создать свои профили. Если вы не разбираетесь в теме, вы можете одной командой повысить производительность в разумных пределах, если же разбираетесь, то можете настроить свои профили, закинуть их на все сервера и в итоге это всё будет управляться демоном и готовым инструментом. Но обо всём по порядку.

```
[user@centos8 ~]$ sudo dnf install tuned
[sudo] password for user:
Last metadata expiration check: 1:53:17 ago on Wed 18 Aug 2021 11:01:14 AM +04.
Package tuned-2.15.0-2.el8_4.1.noarch is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[user@centos8 ~]$ sudo systemctl enable --now tuned
```

tuned обычно предустановлен, но его может и не быть. Если статус показывает, что этого сервиса нет, то вам нужно его предварительно установить, запустить и добавить в автозапуск:

```
sudo dnf install tuned
sudo systemctl enable --now tuned
```

```
[user@centos8 ~]$ tuned-adm
active      list      off      profile   recommend verify
[user@centos8 ~]$ tuned-adm list
Available profiles:
- accelerator-performance      - Throughput performance based tuning with disabled higher latency
STOP states
- balanced                   - General non-specialized tuned profile
- desktop                     - Optimize for the desktop use-case
- hpc-compute                 - Optimize for HPC compute workloads
- intel-sst                   - Configure for Intel Speed Select Base Frequency
- latency-performance         - Optimize for deterministic performance at the cost of increased power consumption
- network-latency             - Optimize for deterministic performance at the cost of increased power consumption, focused on low latency network performance
- network-throughput          - Optimize for streaming network throughput, generally only necessary on older CPUs or 40G+ networks
- optimize-serial-console     - Optimize for serial console use.
- powersave                   - Optimize for low power consumption
- throughput-performance      - Broadly applicable tuning that provides excellent performance across a variety of common server workloads
- virtual-guest               - Optimize for running inside a virtual guest
```

Основная утилита для работы с tuned это tuned-adm. Она очень простая и опций довольно мало. Начнём с опции list:

```
tuned-adm list
```

Она показывает все профили. Так как мы своего профиля не создавали, нам отображаются шаблонные профили и их описания. Например:

- desktop - оптимизированный под пользовательские компьютеры
- powersave - оптимизированный под низкое энергопотребление

Многие профили нацелены на высокую производительность сети и процессора при большем потреблении энергии.

```
[user@centos8 ~]$ tuned-adm active
Current active profile: virtual-guest
[user@centos8 ~]$ tuned-adm recommend
virtual-guest
[user@centos8 ~]$ █
```

Узнать текущий профиль можно с помощью опции active:

```
tuned-adm active
```

Как видите, сейчас профилем является virtual-guest. tuned при установке пытается выбрать более подходящий профиль. Он определил, что мы устанавливаем систему как виртуалку, поэтому выбрал готовый профиль для виртуальных машин.

Также tuned может порекомендовать один из шаблонных профилей:

```
tuned-adm recommend
```

```
[user@centos8 ~]$ sudo tuned-adm profile
accelerator-performance    latency-performance      recommend.d
balanced                   network-latency          throughput-performance
desktop                    network-throughput     virtual-guest
hpc-compute                optimize-serial-console virtual-host
intel-sst                  powersave
[user@centos8 ~]$ sudo tuned-adm profile network-latency
[user@centos8 ~]$ █
```

Поменять его можно с помощью опции `profile`, указав нужный профиль:

```
sudo tuned-adm profile network-latency
```

Для примера выберем профиль с низкой задержкой в сети.

```
[user@centos8 ~]$ tuned-adm active
Current active profile: network-latency
[user@centos8 ~]$ tuned-adm verify
Verification failed, current system settings differ from the preset profile.
You can mostly fix this by restarting the Tuned daemon, e.g.:
  systemctl restart tuned
or
  service tuned restart
Sometimes (if some plugins like bootloader are used) a reboot may be required.
See tuned log file ('/var/log/tuned/tuned.log') for details.
[user@centos8 ~]$ sudo systemctl restart tuned
[user@centos8 ~]$ tuned-adm verify
Verification failed, current system settings differ from the preset profile.
You can mostly fix this by restarting the Tuned daemon, e.g.:
  systemctl restart tuned
or
  service tuned restart
Sometimes (if some plugins like bootloader are used) a reboot may be required.
See tuned log file ('/var/log/tuned/tuned.log') for details.
[user@centos8 ~]$ reboot
```

Но зачастую этого недостаточно. Некоторые изменения требует перезапуска сервиса или всей системы. Убедимся, что у нас выбран нужный профиль с помощью `active`. После чего можно использовать опцию `verify` - она проверит, насколько применился профиль и подскажет, если что-то не так. В нашем случае мы видим, что `verify` выдал ошибку - какие-то текущие параметры отличаются от заданных в профиле. Также нам подсказали, что многие проблемы можно решить перезапуском демона. Но и это нам не помогло - `verify` снова жалуется. Т.е. в нашем случае нужно перезагрузить систему.

```
tuned-adm active
tuned-adm verify
sudo systemctl restart tuned
tuned-adm verify
reboot
```

```
[user@centos8 ~]$ tuned-adm verify
Verification failed, current system settings differ from the preset profile.
You can mostly fix this by restarting the Tuned daemon, e.g.:
  systemctl restart tuned
or
  service tuned restart
Sometimes (if some plugins like bootloader are used) a reboot may be required.
See tuned log file ('/var/log/tuned/tuned.log') for details.
[user@centos8 ~]$ tail /var/log/tuned/tuned.log
2021-08-18 18:56:24,918 INFO      tuned.plugins.base: verify: passed: 'vm.dirty_ratio' = '10'
2021-08-18 18:56:24,918 INFO      tuned.plugins.base: verify: passed: 'vm.dirty_background_ratio' = '3'
'
2021-08-18 18:56:24,918 ERROR    tuned.plugins.base: verify: failed: 'vm.swappiness' = '35', expected
'10'
2021-08-18 18:56:24,918 INFO      tuned.plugins.base: verify: passed: 'kernel.sched_migration_cost_ns'
= '5000000'
2021-08-18 18:56:24,918 INFO      tuned.plugins.base: verify: passed: 'net.core.busy_read' = '50'
2021-08-18 18:56:24,918 INFO      tuned.plugins.base: verify: passed: 'net.core.busy_poll' = '50'
2021-08-18 18:56:24,918 INFO      tuned.plugins.base: verify: passed: 'net.ipv4.tcp_fastopen' = '3'
2021-08-18 18:56:24,919 INFO      tuned.plugins.base: verify: passed: 'kernel.numa_balancing' =
'
2021-08-18 18:56:24,919 INFO      tuned.plugins.base: verify: passed: 'transparent_hugepages' = 'never'
```

Проверим:

```
tuned-adm verify
```

ошибка осталась. Но снизу есть подсказка, что стоит проверить лог файл. Проверим:

```
tail /var/log/tuned/tuned.log
```

и тут видно, что значение параметра `vm.swappiness` - 35, а должно быть 10. Помните, мы его вручную прописывали в файле? Видимо сейчас он перебивает значение, задаваемое tuned.

GNU nano 2.9.8	/etc/sysctl.d/99-sysctl.conf
<pre># sysctl settings are defined through files in # /usr/lib/sysctl.d/, /run/sysctl.d/, and /etc/sysctl.d/. # # Vendors settings live in /usr/lib/sysctl.d/. # To override a whole file, create a new file with the same in # /etc/sysctl.d/ and put new settings there. To override # only specific settings, add a file with a lexically later # name in /etc/sysctl.d/ and put new settings there. # # For more information, see sysctl.conf(5) and sysctl.d(5). #Vm.swappiness=35</pre>	

Давайте закомментируем нашу строчку в `/etc/sysctl.d/99-sysctl.conf`.

```
[user@centos8 ~]$ sudo nano /etc/sysctl.d/99-sysctl.conf
[user@centos8 ~]$ sudo tuned-adm profile network-latency
[user@centos8 ~]$ sudo tuned-adm verify
Verification succeeded, current system settings match the preset profile.
See tuned log file ('/var/log/tuned/tuned.log') for details.
[user@centos8 ~]$ █
```

После чего переактивируем профиль, чтобы он перезаписал этот параметр. Ещё раз проверим `verify` - всё как надо.

```
sudo tuned-adm profile network-latency  
tuned-adm verify
```

```
[user@centos8 ~]$ tuned-adm off  
[user@centos8 ~]$ tuned-adm active  
No current active profile.  
[user@centos8 ~]$ sudo tuned-adm profile network-latency  
[sudo] password for user:  
[user@centos8 ~]$ tuned-adm active  
Current active profile: network-latency  
[user@centos8 ~]$
```

Чтобы в целом выключить tuned надо отключать сам сервис с помощью systemctl. Если нужно временно вырубить профиль, то можно использовать опцию off. А чтобы вернуть обратно, достаточно заново активировать профиль:

```
tuned-adm off  
tuned-adm active  
tuned-adm profile network-latency  
tuned-adm active
```

```
[user@centos8 ~]$ dnf search tuned-profiles  
Last metadata expiration check: 0:00:16 ago on Wed 18 Aug 2021 01:58:37 PM +04.  
===== Name Matched: tuned-profiles =====  
tuned-profiles-mssql.noarch : Additional tuned profile(s) for MS SQL Server  
tuned-profiles-atomic.noarch : Additional tuned profile(s) targeted to Atomic  
tuned-profiles-compat.noarch : Additional tuned profiles mainly for backward compatibility with  
                               : tuned 1.0  
tuned-profiles-oracle.noarch : Additional tuned profile(s) targeted to Oracle loads  
tuned-profiles-cpu-partitioning.noarch : Additional tuned profile(s) optimized for CPU  
                                         : partitioning  
[user@centos8 ~]$
```

Кроме предустановленных профилей также в репозиториях можно найти и установить другие:

```
dnf search tuned-profiles
```

```
[user@centos8 ~]$ cd /usr/lib/tuned  
[user@centos8 tuned]$ ls  
accelerator-performance  hpc-compute          network-throughput      throughput-performance  
balanced                 intel-sst            optimize-serial-console virtual-guest  
desktop                  latency-performance   powersave              virtual-host  
functions                network-latency     recommend.d  
[user@centos8 tuned]$ cd network-latency/  
[user@centos8 network-latency]$ ls  
tuned.conf  
[user@centos8 network-latency]$ █
```

Дефолтные профили лежат в директории `/usr/lib/tuned`. Здесь директории соответствуют названиям профилей, а внутри каждой директории есть свой файл `tuned.conf`, где и находятся сами параметры.

```
cd /usr/lib/tuned/network-latency  
cat tuned.conf
```

```
[user@centos8 network-latency]$ cat tuned.conf
#
# tuned configuration
#
[main]
summary=Optimize for deterministic performance at the cost of increased power consumption, focused on low latency network performance
include=latency-performance

[vm]
transparent_hugepages=never

[sysctl]
net.core.busy_read=50
net.core.busy_poll=50
net.ipv4.tcp_fastopen=3
kernel.numa_balancing=0

[bootloader]
cmdline_network_latency=skew_tick=1
[user@centos8 network-latency]$ █
```

Сам файл состоит из нескольких секций. В секции main мы видим описание профиля в опции summary, а также опцию include - она указывает, что в данном профиле также используются настройки из профиля latency-performance. Т.е. можно не с нуля писать профиль, а взять готовый и приписать ему ещё пару параметров.

Следующие секции это плагины. Дело в том, что различные настройки производительности нужно прописывать в различных местах - где-то нужно в загрузчике добавить, где-то нужно поменять параметр ядра через sysctl, где-то через другие файлы. tuned профиль просто объединяет всё это в одном месте. Кроме перечисленных, есть ещё плагины для задания параметров процессора, диска, сети, сервисов, селинукс, запуск скриптов и т.п.

```
[user@centos8 tuned]$ grep -Ev "^#|^$" powersave/tuned.conf
[main]
summary=Optimize for low power consumption
[cpu]
governor=ondemand|powersave
energy_perf_bias=powersave|power
[eeepc_she]
[vm]
[audio]
timeout=10
[video]
radeon_powersave=dpm-battery, auto
[disk]
[net]
[scsi_host]
alpm=min_power
[sysctl]
vm.laptop_mode=5
vm.dirty_writeback_centisecs=1500
kernel.nmi_watchdog=0
```

Для примера посмотрим ещё powersave профиль, так как там задействовано много плагинов:

```
cd ..
grep -Ev "^#|^$" powersave/tuned.conf
```

Так как параметры специфичны для плагинов, а они охватывают различные файлы и утилиты, нет какой-то единой документации, где бы показывались всевозможные опции и объяснения для всех параметров.

```
[user@centos8 ~]$ cd /etc/tuned/
[user@centos8 tuned]$ ls
active_profile  bootcmdline  post_loaded_profile  profile_mode  recommend.d  tuned-main.conf
[user@centos8 tuned]$ █
```

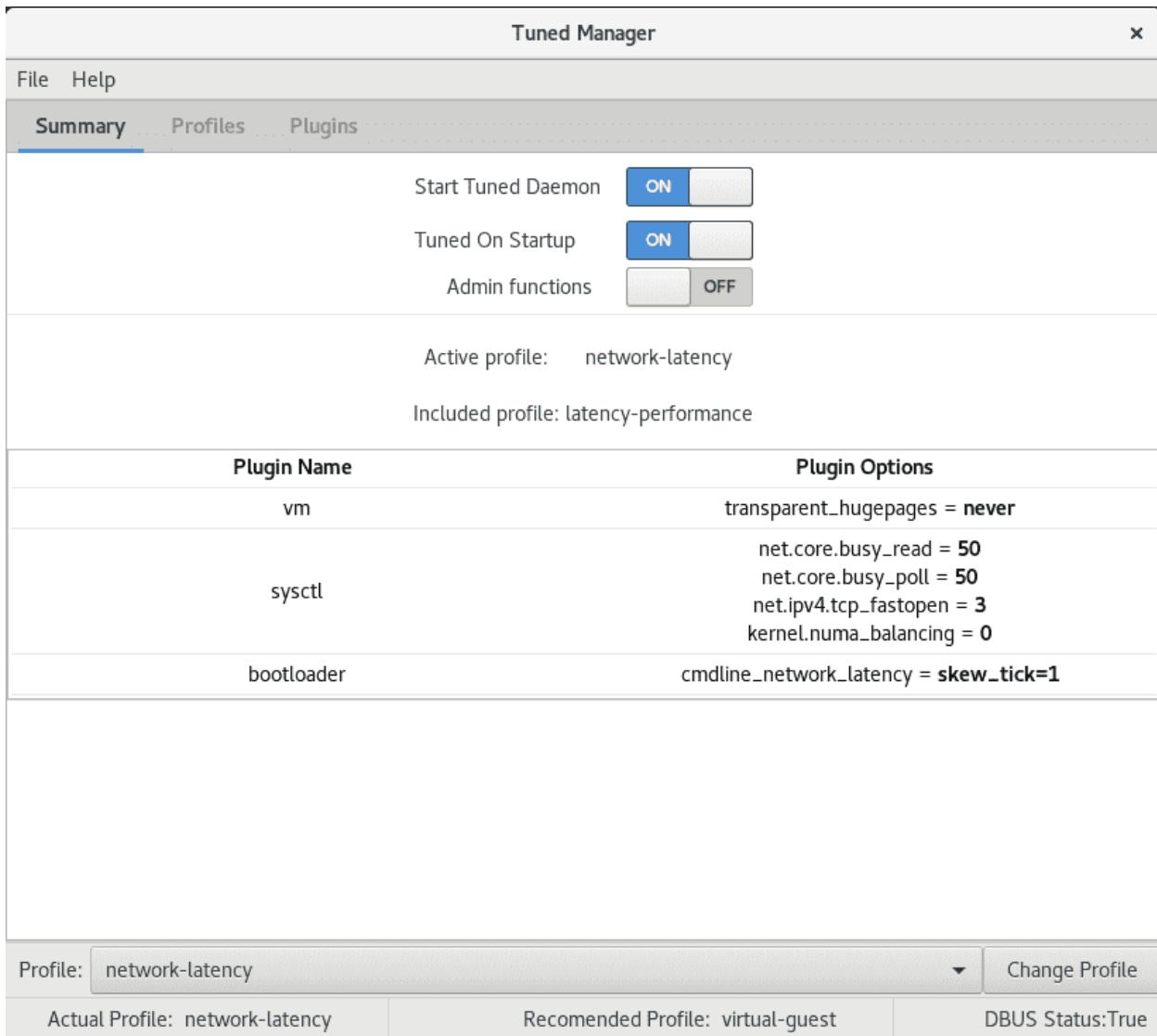
По примеру многих утилит, если дефолтные профили лежат в директории /usr/lib/tuned, то свои кастомные можно создать в директории /etc/tuned. Нужно создать директорию с именем профиля, а внутри файл tuned.conf. Но я этого делать не буду, нам пока нет смысла создавать свои профили.

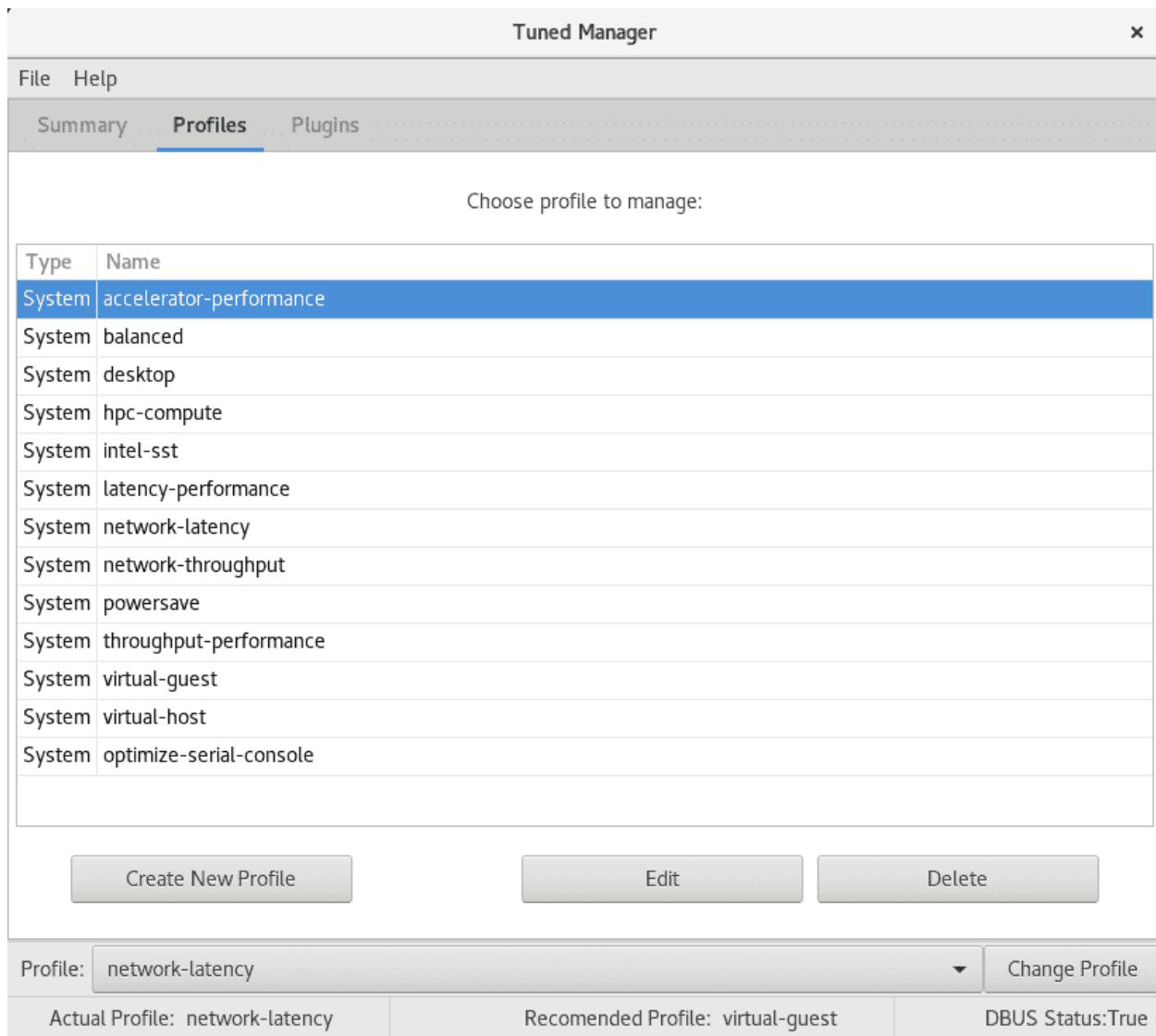
```
[user@centos8 tuned]$ sudo dnf install tuned-gtk
Last metadata expiration check: 0:50:23 ago on Wed 18 Aug 2021 02:52:52 PM +04.
Dependencies resolved.
=====
Package           Architecture      Version       Repository     Size
=====
Installing:
tuned-gtk          noarch        2.15.0-2.el8_4.1   AppStream    60 k
Installing dependencies:
powertop          x86_64        2.12-2.el8      AppStream    260 k
Transaction Summary
=====
Install 2 Packages

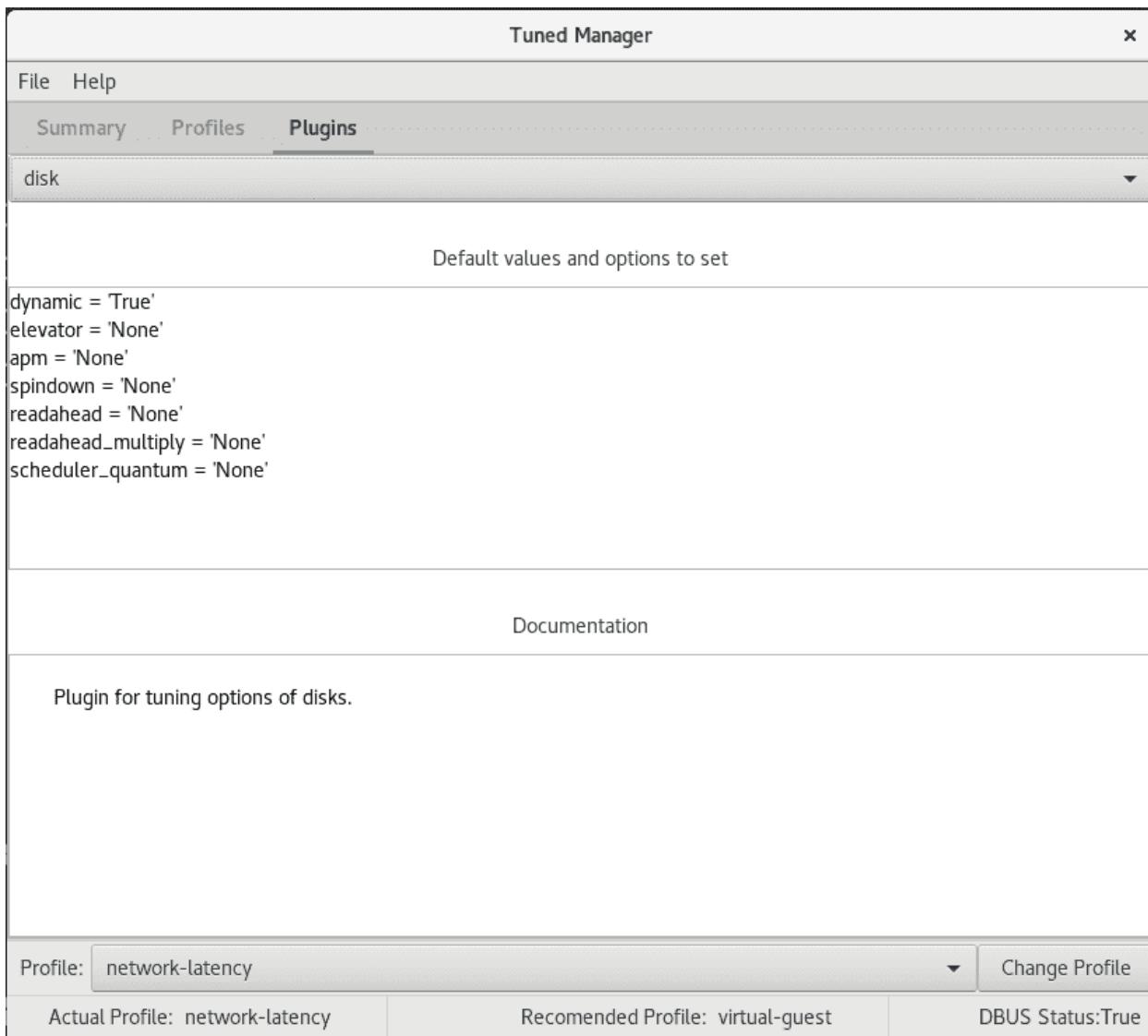
Total download size: 320 k
Installed size: 802 k
Is this ok [y/N]: y
```

Кстати, tuned можно настраивать и через графическое приложение, но для этого надо доустановить пакет.

```
sudo dnf install tuned-gtk
```







Из этого приложения можно как управлять самим tuned, так и настраивать кастомные профили.

Ну и одна из интересных фишек tuned - динамическая оптимизация. Всё что мы настраивали до этого применяется и работает постоянно. Скажем, мы выбрали, чтобы производительность сети была высокой, а значит она всегда будет такой, независимо от того, есть в этом необходимость или нет. Однако tuned умеет это делать динамически. Т.е., допустим, когда вы запускаете компьютер, скорость диска должна быть высокой, чтобы перенести кучу данных в оперативку. Однако потом диску не нужна такая скорость и при обычной работе данные пишутся ичитываются относительно реже. Или, допустим, у вас сетевая карта на домашнем компьютере большую часть времени простаивает или используется очень слабо, скажем, для просмотра веб страниц. И лишь изредка вам нужно скачать что-то большое. Вместо того, чтобы постоянно использовать вашу сетевую карту на максимум, почему бы не делать это только тогда, когда есть необходимость?

Но, как мы уже разбирали, не все опции можно так легко переключать. Что-то можно изменить на лету, а что-то сработает только после перезагрузки. Динамическая оптимизация не касается всего, её можно настроить для определённых плагинов. И в основном она годится для рабочих станций, а не для серверов.

```
GNU nano 2.9.8 /etc/tuned/tuned-main.conf

# Global tuned configuration file.

# Whether to use daemon. Without daemon it just applies tuning. It is
# not recommended, because many functions don't work without daemon,
# e.g. there will be no D-Bus, no rollback of settings, no hotplug,
# no dynamic tuning, ...
daemon = 1

# Dynamically tune devices, if disabled only static tuning will be used.
dynamic_tuning = 1
```

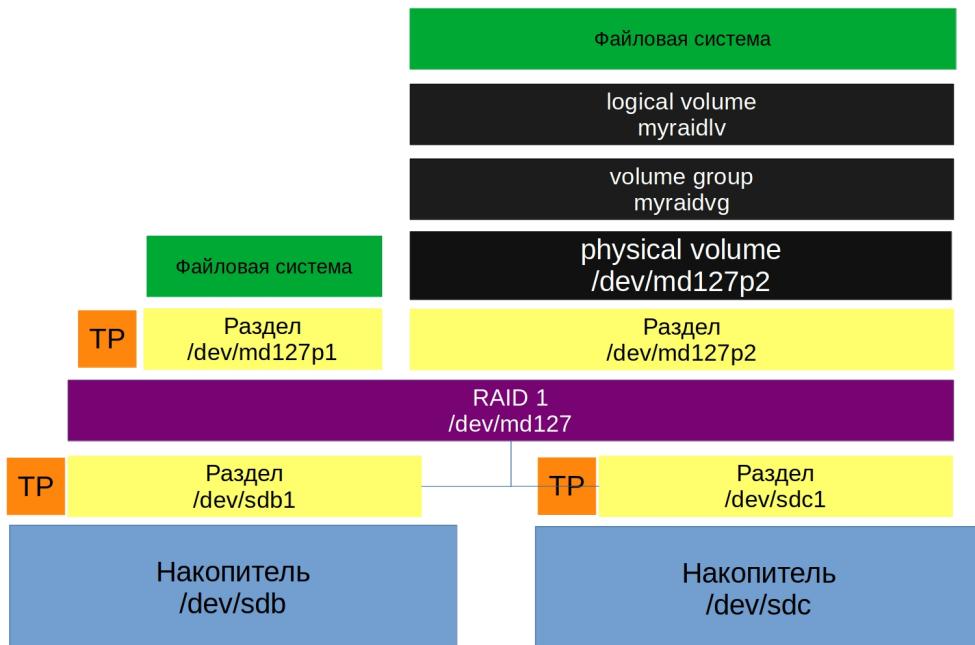
[user@centos8 tuned]\$ sudo nano /etc/tuned/tuned-main.conf
[user@centos8 tuned]\$ sudo systemctl restart tuned

По умолчанию она выключена, и, чтобы её применить, надо в файле `/etc/tuned/tuned-main.conf` поменять значение параметра `dynamic_tuning` на 1, после чего рестартнуть демон `tuned`.

Подведём итоги. Сегодня мы с вами разобрали демон `tuned`, который облегчает настройку производительности и позволяет централизовано хранить все необходимые параметры в одном файле. Стандартные профили позволяют вам очень просто улучшить производительность, а возможность создавать свои профили позволяет профессионалам упростить работу, создавая шаблоны для разного рода серверов.

2.52 52. Управление многоуровневым хранилищем - stratis

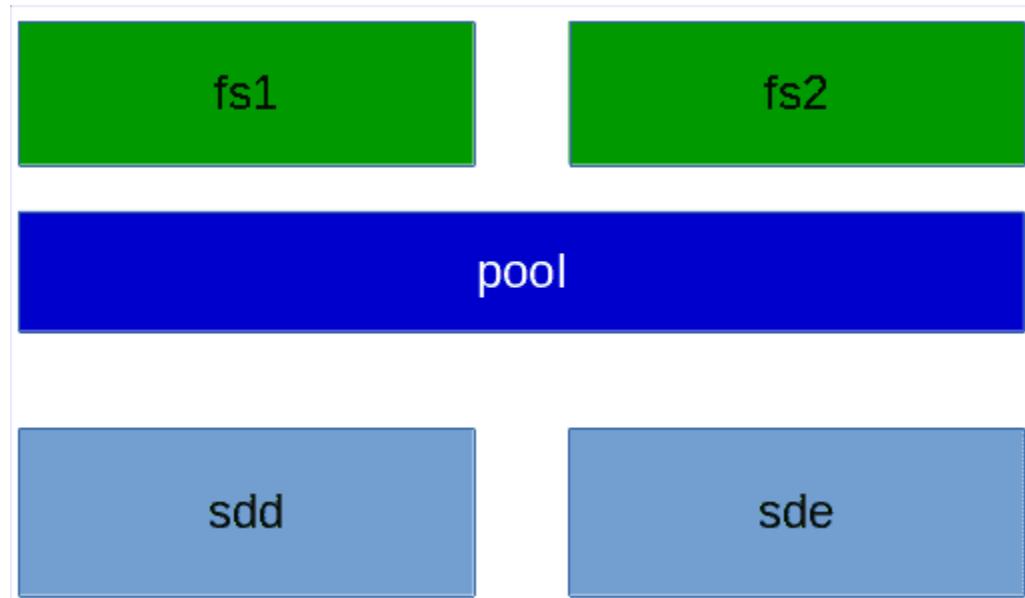
2.52.1 52. Управление многоуровневым хранилищем - stratis



Диски и файловые системы – одна из важнейших тем, к которой мы возвращаемся уже не раз. Стан-

дартные разделы, RAID, LVM, файловые системы - с одной стороны нужно всё это знать и уметь с этим работать. С другой стороны это добавляет слои абстракции, появляется куча утилит, с которыми нужно уметь обращаться и даже какие-то простые задачи становятся гораздо комплекснее, не говоря уже о нетривиальных проблемах. А цена ошибки может быть слишком высокой.

Так как все эти инструменты зачастую используются вместе и нужны для решения одной цели, это в итоге должно было привести к появлению единого решения. Подобные решения уже есть, к примеру, btrfs или zfs, но по определённым причинам Red Hat не берётся внедрять их. Вместо них компания работает над другим проектом - stratis. Сразу отмечу, что проект пока в разработке и доступен для тестирования, но применять его в рабочей среде пока не стоит.



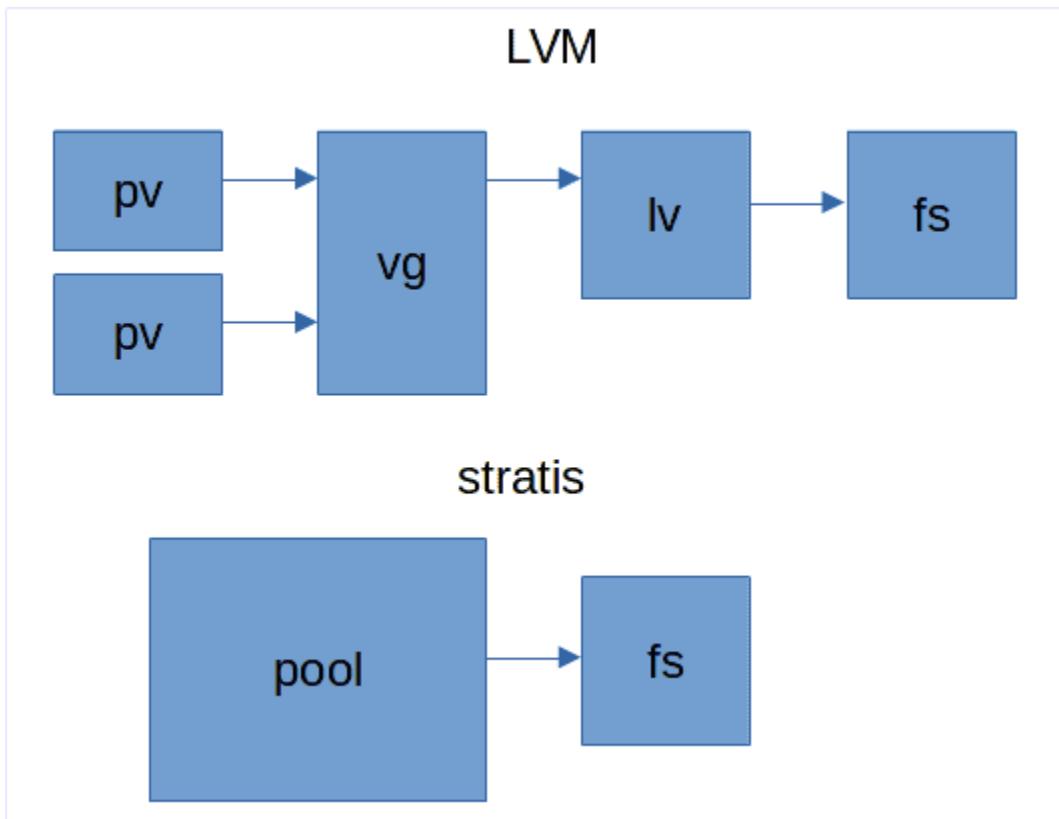
stratis сейчас напоминает упрощённую смесь LVM и mkfs. Очень грубо говоря, с помощью него вы объединяете диски в группы, называемые pool, и создаёте на них файловые системы.

Вкратце о функционале:

- нет RAID-a, т.е. отказоустойчивости. stratis можно поднять и поверх RAIDа, как мы это делали с LVM. Но в будущем, скорее всего, добавят такой функционал.
- есть снапшоты, но, в отличии от LVM, здесь они являются просто полной копией файловой системы с другим UUID-ом.



- есть tiering. Очень грубо говоря, это такая технология, когда берут несколько дисков hdd и ssd, объединяют их в одну группу - один pool, и ssd используют в качестве кэша. Т.е. когда вы будете что-то писать в этот пул, сначала данные будут писаться на ssd, что ускорит скорость записи, а потом в фоне переноситься на жёсткие диски, где больше места. На самом деле tiering чуть более сложнее и у разных производителей это понятие может отличаться, но в такие детали мы пока вдаваться не будем.
- есть шифрование - можно создавать зашифрованные пулы.



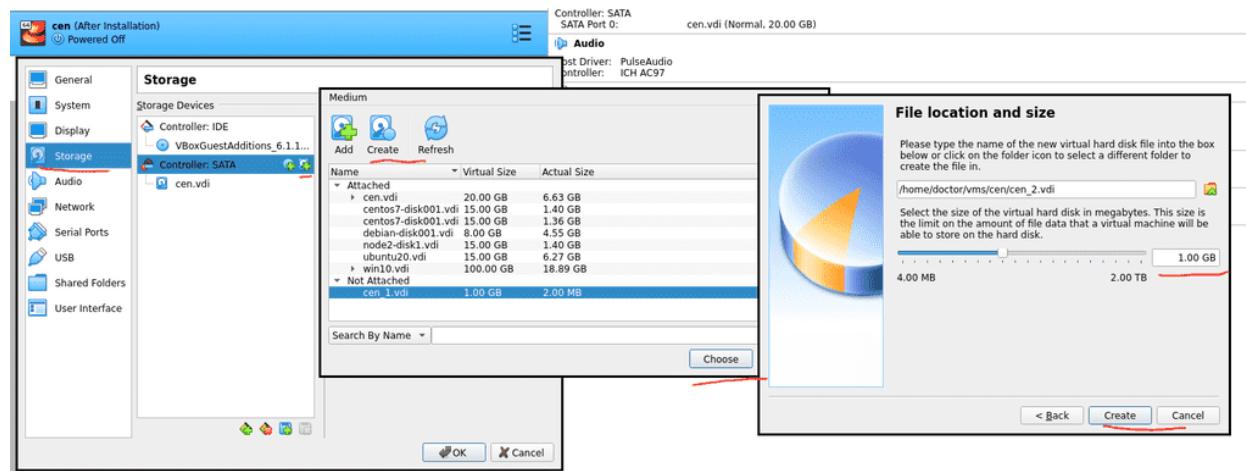
- есть thin provisioning. И тут таится главное отличие от LVM. Нет, в LVM тоже есть thin provisioning, но тут это сделано несколько иначе.

Когда вы работаете с LVM, вы создаёте физические тома, создаёте из них группу томов, в этой группе

создаёте логический том и на нём файловую систему. При создании логического тома вы указываете его размер и файловая система внутри будет того же объёма.

Когда же вы работаете со stratis, вы просто создаёте pool и внутри создаёте файловую систему. Вам нигде не нужно указывать размер. Файловая система автоматом создаётся на 1 ТБ, но она этот объём не занимает. Занимается только то пространство, которое занято реальными данными. При этом, stratis следит за объёмом, и, если файловая система заполняется, то он автоматом увеличивает её. Если, конечно, есть место в пуле.

С одной стороны это удобно - про ручное увеличение логического тома и файловой системы можно забыть. С другой стороны, в stratis используется только xfs, а значит уменьшить файловую систему не получится. Тут очень важно следить за тем, чтобы всегда в пуле оставалось пространство и вовремя добавлять новые диски, иначе с файловой системой могут возникнуть проблемы.



И так, прежде чем продолжим, давайте с помощью виртуалбокса добавим в систему два новых диска, по 1 ГБ. В системе ещё есть 3 диска, первый для системы и два у нас были выделены под raid.

NAME	MAJ:MIN	RM	SIZE	R0	TYPE	MOUNTPOINT
sda	8:0	0	20G	0	disk	
└─sda1	8:1	0	1G	0	part	/boot
└─sda2	8:2	0	19G	0	part	
└─cl_centos8-root	253:0	0	17G	0	lvm	/
└─cl_centos8-swap	253:1	0	2G	0	lvm	[SWAP]
sdb	8:16	0	1G	0	disk	
└─sdb1	8:17	0	954M	0	part	
└─md127	9:127	0	953M	0	raid1	
└─md127p1	259:0	0	286M	0	md	
└─md127p2	259:1	0	666M	0	md	
└─myraidvg-myraidlv	253:2	0	528M	0	lvm	/mydata
sdc	8:32	0	1G	0	disk	
└─sdc1	8:33	0	954M	0	part	
└─md127	9:127	0	953M	0	raid1	
└─md127p1	259:0	0	286M	0	md	
└─md127p2	259:1	0	666M	0	md	
└─myraidvg-myraidlv	253:2	0	528M	0	lvm	/mydata
sdd	8:48	0	1G	0	disk	
sde	8:64	0	1G	0	disk	
sr0	11:0	1	1024M	0	rom	

[user@centos8 ~]\$ █

После запуска системы посмотрим lsblk - здесь мы видим, что на sdb1 и sdc1 у нас raid, а поверх рейда стоит lvm. Ну и два новых пустых диска - sdd и sde.

```
[user@centos8 ~]$ sudo wipefs -a /dev/sdd /dev/sde
[user@centos8 ~]$ █
```

Если бы на этих дисках раньше были какие-то данные, какая-то файловая система, лвм и прочее, то предварительно следовало это всё удалить, а потом сделать wipefs:

sudo wipefs -a /dev/sdd /dev/sde

```
[user@centos8 ~]$ sudo dnf install stratisd stratis-cli
Last metadata expiration check: 0:02:19 ago on Fri 27 Aug 2021 01:54:56 PM +04.
Dependencies resolved.
=====
Package           Architecture Version      Repository    Size
=====
Installing:
stratis-cl       noarch        2.3.0-3.el8   AppStream   79 k
stratisd         x86_64        2.3.0-2.el8   AppStream   2.1 M
=====

```

Также нам нужно установить демон stratisd и утилиту для работы с ним.

sudo dnf install stratisd stratis-cl

```
[user@centos8 ~]$ sudo systemctl status stratisd
● stratisd.service - Stratis daemon
  Loaded: loaded (/usr/lib/systemd/system/stratisd.service; enabled; vendor preset: enabled)
    Active: inactive (dead)
      Docs: man:stratisd(8)
[user@centos8 ~]$ sudo systemctl start stratisd
[user@centos8 ~]$ sudo systemctl status stratisd
● stratisd.service - Stratis daemon
  Loaded: loaded (/usr/lib/systemd/system/stratisd.service; enabled; vendor preset: enabled)
    Active: active (running) since Fri 2021-08-27 13:59:01 +04; 2s ago
      Docs: man:stratisd(8)
  Main PID: 33449 (stratisd)
    Tasks: 1 (limit: 11334)
   Memory: 1.3M
     CGroup: /system.slice/stratisd.service
             └─33449 /usr/libexec/stratisd --log-level debug
```

После установки стоит убедиться, что сервис stratisd включён, он должен автоматом запускаться при включении компьютера:

```
systemctl status stratisd
```

Как видно, сервис включён, но в данный момент не запущен. Давайте запустим его и убедимся, что он работает:

```
sudo systemctl start stratisd
sudo systemctl status stratisd
```

```
[user@centos8 ~]$ sudo stratis
blockdev filesystem --help      pool      report
daemon      -h      key      --propagate --version
[user@centos8 ~]$ sudo stratis pool
add-cache add-data  create      destroy      init-cache  list      rename      unlock
[user@centos8 ~]$ sudo stratis filesystem
create      destroy      list      rename      snapshot
[user@centos8 ~]$ █
```

Основная утилита - stratis. Опций немного и основные это startis pool и stratis filesystem - с помощью них можно смотреть и управлять пулами и файловыми системами.

```
[user@centos8 ~]$ sudo stratis pool create mypool /dev/sdd
[sudo] password for user:
[user@centos8 ~]$ sudo stratis pool list
Name          Total Physical  Properties
mypool  1 GiB / 37.63 MiB / 986.37 MiB  ~Ca,~Cr
[user@centos8 ~]$
```

Для начала надо создать pool - create - указываем название и диски, которые входят в pool. Пока что добавим только один диск:

```
sudo stratis pool create mypool /dev/sdd
```

Команда завершилась без ошибок. Пулов может быть несколько, посмотрим их список:

```
sudo stratis pool list
```

И здесь мы видим созданный нами пул. Во втором столбце мы видим общий объём, занятый и свободный. В Properties мы видим ~Ca и ~Cr. Ca означает, что в пуле есть кэш, Cr - что пул зашифрован, а тильда перед ними означает «нет». Т.е. в нашем пуле нет кэш диска и нет шифрования.

```
[user@centos8 ~]$ sudo stratis pool
add-cache add-data create destroy init-cache list rename unlock
[user@centos8 ~]$ sudo stratis pool init-cache mypool /dev/sde
[user@centos8 ~]$ sudo stratis pool list
Name Total Physical Properties
mypool 1 GiB / 37.63 MiB / 986.37 MiB Ca,~Cr
[user@centos8 ~]$ sudo stratis blockdev list
Pool Name Device Node Physical Size Tier
mypool /dev/sdd 1 GiB Data
mypool /dev/sde 1 GiB Cache
[user@centos8 ~]$ █
```

Давайте второй диск подключим как кэш. У опции pool есть несколько ключей:

- add-cache - добавляет кэш диск. Но она сработает, только если у нас уже есть кэш.
- add-data - добавляет обычный диск, т.е. расширяет пул, как vgextend.
- init-cache - создаёт кэш. Как раз этот ключ нам сейчас нужен.

Указываем его, наш пул и условный ssd диск.

```
sudo stratis pool init-cache mypool /dev/sde
```

Команда завершилась без ошибок, а значит можем посмотреть список пулов. Как вы заметили, теперь перед Са пропала тильда. И, как видно, пространство не увеличилось, так как весь второй диск используется в качестве кэша, а не для постоянного хранения.

```
sudo stratis pool list
```

Список устройств и к каким пулам они относятся можно увидеть с помощью опции blockdev. Здесь видно, что sdd используется для данных, а sde для кэша.

```
sudo stratis blockdev list
```

```
[user@centos8 ~]$ sudo stratis filesystem
create destroy list rename snapshot
[user@centos8 ~]$ sudo stratis filesystem create mypool myfs1
[user@centos8 ~]$ sudo stratis filesystem list
Pool Name Name Used Created Device UUID
mypool myfs1 546 MiB Aug 27 2021 14:38 /dev/stratis/mypool/myfs1 8d18731d410943f7ae6
0df9a5c1d3cd3
[user@centos8 ~]$ █
```

Теперь создадим файловую систему. Для этого используется опция filesystem с ключом create, указанием пула и названия файловой системы.

```
sudo stratis filesystem create mypool myfs1
```

А чтобы посмотреть список файловых систем - list:

```
sudo stratis filesystem list
```

Как видно, у нас есть файловая система, путь к ней - /dev/stratis/mypool/myfs1, а использовано уже где-то 500 мегабайт. Это всё выделено под метаданные, но почему так много?

```
[user@centos8 ~]$ sudo mount /dev/stratis/mypool/myfs1 /mnt/
[user@centos8 ~]$ sudo df -h /mnt/
Filesystem      Size   Used  Avail Use% Mounted on
/dev/mapper/stratis-1-c648fc3d3780341bc9b96171bfe1308be-thin-fs-8d18731d410943f7ae60df9a5c1d3cd3
1.0T    7.2G  1017G   1%  /mnt
[user@centos8 ~]$ █
```

Давайте примонтируем и проверим.

```
sudo mount /dev/stratis/mypool/myfs1 /mnt
df -h /mnt
```

Как видно из вывода df, и, как я говорил ранее, настоящий размер файловой системы 1 ТБ, но из-за thin provisioning-а мы тратим только используемое пространство - 500 мегабайт. Как мы помним из темы о файловых системах, на каждый промежуток пространства создаются иноды прочая информация о файловой системе и её структуре, вследствие чего на 1 ТБ для такой информации пришлось потратить 500 МБ. И то, как видно, это 7.2 гигабайта, но так как эти данные пока пустые, мы потратили поменьше.

```
└stratis-1-private-c648fc3d3780341bc9b96171bfe1308be-physical-cachesub
    253:10    0    16M  0 stratis
    253:5     0    508M  0 stratis
    └stratis-1-private-c648fc3d3780341bc9b96171bfe1308be-physical-cache
        253:6     0   1020M  0 stratis
        └stratis-1-private-c648fc3d3780341bc9b96171bfe1308be-flex-thinmeta
            253:7     0    16M  0 stratis
            └stratis-1-private-c648fc3d3780341bc9b96171bfe1308be-thinpool-pool
                253:9     0   972M  0 stratis
                └stratis-1-c648fc3d3780341bc9b96171bfe1308be-thin-fs-8d18731d410943f7ae60df9a5c1d3cd3
                    253:11    0    1T  0 stratis
    └stratis-1-private-c648fc3d3780341bc9b96171bfe1308be-flex-thindata
        253:8     0   972M  0 stratis
        └stratis-1-private-c648fc3d3780341bc9b96171bfe1308be-thinpool-pool
            253:9     0   972M  0 stratis
            └stratis-1-c648fc3d3780341bc9b96171bfe1308be-thin-fs-8d18731d410943f7ae60df9a5c1d3cd3
                253:11    0    1T  0 stratis
    └stratis-1-private-c648fc3d3780341bc9b96171bfe1308be-flex-mdv
        253:10    0    16M  0 stratis
        8:80      0    2G  0 disk
        11:0      1  1024M  0 rom
sdf
sr0
```

Так как места осталось довольно мало, давайте добавим ещё один диск на 2 ГБ, правда для этого надо будет выключить виртуалку. После чего проверим новый диск - lsblk. Как видите, после stratis здесь сплошная каша, но можно заметить новый диск - sdf.

```
[user@centos8 ~]$ sudo stratis pool add-data mypool /dev/sdf
[sudo] password for user:
[user@centos8 ~]$ sudo stratis pool list
Name          Total Physical  Properties
mypool    3 GiB / 587.65 MiB / 2.43 GiB  Ca,~Cr
[user@centos8 ~]$ sudo stratis blockdev list
Pool Name  Device Node  Physical Size  Tier
mypool      /dev/sdd      1 GiB   Data
mypool      /dev/sde      1 GiB   Cache
mypool      /dev/sdf      2 GiB   Data
[user@centos8 ~]$
```

Добавим новый диск в существующий пул, используя ключ add-data:

```
sudo stratis pool add-data mypool /dev/sdf
```

Теперь проверим наш пул и список устройств:

```
sudo stratis pool list
sudo stratis blockdev list
```

Как видно, на этот раз размер пула увеличился и теперь в нём 2 data диска и один кэш диск.

```
[user@centos8 ~]$ sudo stratis filesystem snapshot mypool myfs1 myfs1-snapshot
[user@centos8 ~]$ sudo stratis filesystem list
Pool Name      Name          Used       Created        Device           U
UID
mypool        myfs1         546 MiB   Aug 27 2021 14:38  /dev/stratis/mypool/myfs1      8
d18731d410943f7ae60df9a5c1d3cd3
mypool        myfs1-snapshot 546 MiB   Aug 27 2021 15:10  /dev/stratis/mypool/myfs1-snapshot    f
c3f14082396477293e2a71d8f9fd6bb
[user@centos8 ~]$
```

И теперь мы можем создать снапшот этой файловой системы, используя ключ snapshot и указав пул, название файловой системы и название снапшота:

```
sudo stratis filesystem snapshot mypool myfs1 myfs1-snapshot
```

Снапшот не стоит создавать на примонтированной файловой системе, ну или по крайней мере нужно убедиться, что сейчас ничего не пишется на неё. Как я говорил, в случае со stratis снапшот это просто копия файловой системы, поэтому его можно увидеть с помощью filesystem list.

```
sudo stratis filesystem list
```

Его можно отдельно примонтировать, ну и в целом обращаться как с обычной файловой системой. Ну и чтобы вернуться к данному снапшоту, можно просто примонтировать её вместо оригинала.

```
[user@centos8 ~]$ sudo stratis filesystem
create  destroy  list  rename  snapshot
[user@centos8 ~]$ sudo stratis filesystem destroy mypool myfs1
[sudo] password for user:
[user@centos8 ~]$ sudo stratis filesystem rename mypool myfs1-snapshot myfs1
[user@centos8 ~]$ sudo stratis filesystem list
Pool Name      Name          Used       Created        Device           UUID
mypool        myfs1         546 MiB   Aug 27 2021 15:10  /dev/stratis/mypool/myfs1      fc3f14082396477293e
2a71d8f9fd6bb
[user@centos8 ~]$
```

Ну и если мы решили вернуть снапшот, можно просто удалить старую файловую систему с помощью destroy, переименовать снапшот с помощью rename и считайте что мы восстановили снапшот.

```
sudo stratis filesystem destroy mypool myfs1
sudo stratis filesystem rename mypool myfs1-snapshot myfs1
sudo stratis filesystem list
```

```
[user@centos8 ~]$ sudo blkid /dev/stratis/mypool/myfs1
/dev/stratis/mypool/myfs1: UUID="fc3f1408-2396-4772-93e2-a71d8f9fd6bb" TYPE="xfs"
[user@centos8 ~]$ sudo nano /etc/fstab
[user@centos8 ~]$ tail -1 /etc/fstab
UUID="fc3f1408-2396-4772-93e2-a71d8f9fd6bb" /mnt xfs defaults,x-systemd.requires=stratisd.service
0 0
[user@centos8 ~]$ sudo mount -a
[user@centos8 ~]$ sudo df -h /mnt/
Filesystem
Size Used Avail Use% Mounted on
/dev/mapper/stratis-1-c648fcd3780341bc9b96171bfe1308be-thin-fs-fc3f14082396477293e2a71d8f9fd6bb
1.0T 7.2G 1017G 1% /mnt
[user@centos8 ~]$ █
```

Чтобы файловая система монтировалась при каждом включении, добавим её в fstab. Но предварительно стоит узнать её UUID:

```
sudo blkid /dev/stratis/mypool/myfs1
```

После чего этот UUID надо прописать в fstab:

```
UUID="fc3f1408-2396-4772-93e2-a71d8f9fd6bb" /mnt xfs defaults,x-systemd.
← requires=stratisd.service 0 0
```

Обратите внимание, что указывается тип файловой системы - xfs, а в опциях монтирования стоит указать сервис stratisd. Мы такую же опцию использовали для VDO.

Ну и чтобы убедиться, что файловая система монтируется, используем mount -a

```
sudo mount -a
sudo df -h /mnt
```

Давайте подведём итоги. Ещё раз напомню, что stratis использовать в рабочей среде не стоит, он пока не прошёл проверку временем, а новый функционал, который постоянно добавляется, может всё сломать. Но цель этой темы в том, чтобы показать, что нас ждёт в будущем - возможно, лет через 5 этот инструмент обрастёт функционалом и начнёт набирать популярность. Основной урок, который можно вынести - всё идёт к упрощению. Многие задачи, такие как задание и изменение размеров логических томов и файловых систем, могут быть автоматизированы. Вам нужна файловая система - и вы её создаёте одной понятной командой. Всем остальным будет управлять система, единственное, что остаётся за вами - следить, чтобы в пуле было достаточно места, и, если что, добавлять новые диски. Но это не значит, что в ближайшем будущем стандартные разделы или LVM потеряют актуальность.

2.53 53. Установка RHEL

2.53.1 53. Установка RHEL

Прекращение поддержки CentOS

CentOS Project shifts focus to CentOS Stream

Tuesday , 8, December 2020 | Rich Bowen | Uncategorized | 705 Comments

The future of the CentOS Project is CentOS Stream, and over the next year we'll be shifting focus from CentOS Linux, the rebuild of Red Hat Enterprise Linux (RHEL), to CentOS Stream, which tracks just *ahead* of a current RHEL release. CentOS Linux 8, as a rebuild of RHEL 8, will end at the end of 2021. CentOS Stream continues after that date, serving as the upstream (development) branch of Red Hat Enterprise Linux.

Meanwhile, we understand many of you are deeply invested in CentOS Linux 7, and we'll continue to produce that version through the remainder of the [RHEL 7 life cycle](#).

50 лет назад мы с вами установили CentOS. Тогда это был один из самых распространённых бесплатных дистрибутивов, во многом благодаря тому, что он был копией дистрибутива Red Hat Enterprise Linux. Сам RHEL не отличался функционалом, но из-за подписки на поддержку и доступа к репозиториям стоил денег и не был доступен для бесплатного использования. Компания Red Hat поддерживала проект CentOS, но в конце 2020 [объявила](#) о прекращении поддержки CentOS Linux и смешения фокуса на CentOS Stream. CentOS 8 будет поддерживаться до конца 2021, а 7 версия до середины 2024.

В качестве альтернатив старому CentOS, кроме AlmaLinux, также позиционируются [Rocky Linux](#) и [Oracle Linux](#). Rocky Linux развивается целиком силами сообщества, не зависит от интересов отдельных компаний, но может испытывать недостаток в ресурсах и энтузиастах. Oracle Linux привязан к компании Oracle, которая в любое время может пересмотреть правила игры. AlmaLinux пытается найти оптимальный баланс между корпоративной поддержкой и интересами сообщества - с одной стороны к разработке будут привлечены ресурсы и разработчики компании CloudLinux, которая имеет большой опыт в сопровождении форков RHEL, а с другой стороны проект будет прозрачен и подконтролен сообществу.

В связи с этим несколько компаний [объявили](#) о выпуске своих дистрибутивов на замену CentOS, которые также будут копировать RHEL. Т.е. функционально эти дистрибутивы отличаться не будут, но могут быть отличия в частоте обновлений, количестве пользователей и прочих административных моментах. Второе, со временем, может привести к тому, что какие-то из этих дистрибутивов также прекратят поддерживаться.

Red Hat Developer Subscriptions for Individuals

We are excited that you are interested in participating in the Red Hat Developer program and, as a benefit of the program, you are receiving Red Hat Developer Subscriptions for Individuals (the "Individual Developer Subscriptions"). The Individual Developer Subscriptions allow you (as an individual, natural person) to use certain Red Hat Subscription Services in connection with Red Hat Software for Individual Development Use and for Individual Production Use subject to these Program Terms at no cost. The Individual Developer Subscriptions are unsupported, intended for your individual use in your personal capacity and are not intended or supported for any other purpose. If you are interested in the Red Hat Developer Subscriptions for Teams other terms apply and please contact your Red Hat sales associate.

При этом сам Red Hat [объявил](#), что теперь индивидуальные пользователи могут бесплатно использовать 16 копий RHEL на виртуалках или физических машинах, для тестирования или небольших рабочих задач, но, опять же, только для индивидуального использования. При этом не будет поддержки, но доступ к репозиториям останется.

Для продолжения курса вы можете выбрать любой из дистрибутивов, вышедших на замену. Свой выбор я остановил на RHEL и на это есть несколько причин. Во-первых, я хочу, чтобы этот курс помог вам с работой. Немало компаний используют бесплатные дистрибутивы, но если говорить о крупных компаниях - они предпочитают дистрибутивы с коммерческой поддержкой. И здесь Red Hat лидер. Хотя RHEL почти во всём идентичен с его бесплатными альтернативами, небольшие отличия всё же есть. И поэтому ваш опыт работы с RHEL может оказаться немножко предпочтительнее для работодателя. Во-вторых, и это куда важнее, у Red Hat есть много других продуктов, которые очень часто используются в крупных компаниях. И эти продукты можно установить только на RHEL. А их знание может сильно помочь вам в трудоустройстве в хорошую компанию. Поэтому в дальнейшем я планирую с вами разобрать хотя бы часть этих продуктов.

И так, зачем вообще мы взялись ставить систему? Для некоторых следующих тем нам нужна ещё одна виртуалка. Почему не скопировать существующую? В прошлый раз, когда мы устанавливали систему, мы ничего о ней не знали, поэтому делали это вслепую. Сейчас же, набравшись знаний, мы можем куда детальнее разобрать этот процесс.

Как скачать?

The screenshot shows a web browser window with the URL <https://developers.redhat.com/products/rhel/download>. The page is titled "Developer" and features a navigation bar with links to Istio, Quarkus, Kubernetes, CI/CD, Serverless, Java, Linux, Microservices, and DevOps. The main content area is titled "Red Hat Enterprise Linux" and describes it as "The world's leading enterprise Linux platform". On the left, there is a sidebar with links to Overview, Download (which is underlined), Hello World!, and Docs and APIs. The "Download" section contains a "TRY IT" button and a large red "DOWNLOAD" button with a download icon. Below the download button, the text "Product: Red Hat Enterprise Linux 8.4.0" is visible. The overall layout is clean and professional, typical of a developer-focused software landing page.

Для начала надо скачать ISO образ. Заходим по ссылке на сайт и нажимаем Download.

Log in to your Red Hat account

Red Hat login or email

Next



[Register for a Red Hat account →](#)

[Forgot your password?](#)

Так как аккаунта у нас пока нет, нажимаем Register.

Choose a Red Hat login *

gabenewell

Your login is a user ID for accessing your account across all Red Hat sites. It must be at least 5 characters and **cannot be changed once created.**

Email address *

gaben@valvesoftware.com

First name *

Gabe

Last name *

Newell

Company name *

Individual

Job role *

Student

Choose a password *

.....

SHOW

Your password must include at least 8 characters. A strong password combines lowercase letters, uppercase letters, numbers, and symbols.

NEXT

Вводим наши данные и нажимаем Next.

I have read and agree to all the terms and conditions below (check all boxes).

* I have read and agree to the [Enterprise Agreement](#).

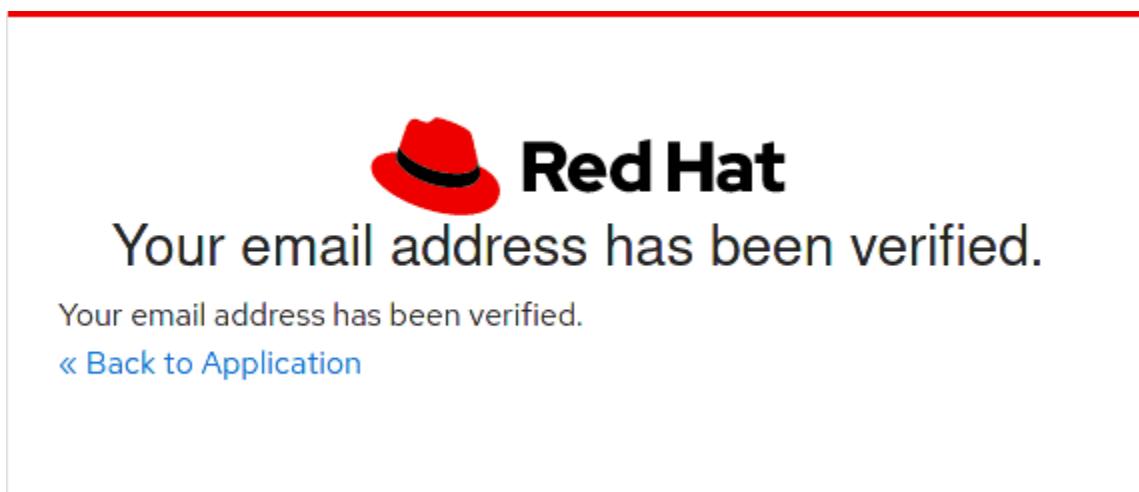
* I have read and agree to the [Red Hat Developer Subscription for Individuals](#).

Email opt-in

Receive email notifications of Red Hat Developer services and events, including invitations and reminders.

CREATE MY ACCOUNT

И опять вводим данные и нажимаем «Create My Account».



Подтверждаем email...

Download for Development Use

TRY IT

 DOWNLOAD

Product: Red Hat Enterprise Linux 8.4.0

ALL DOWNLOADS

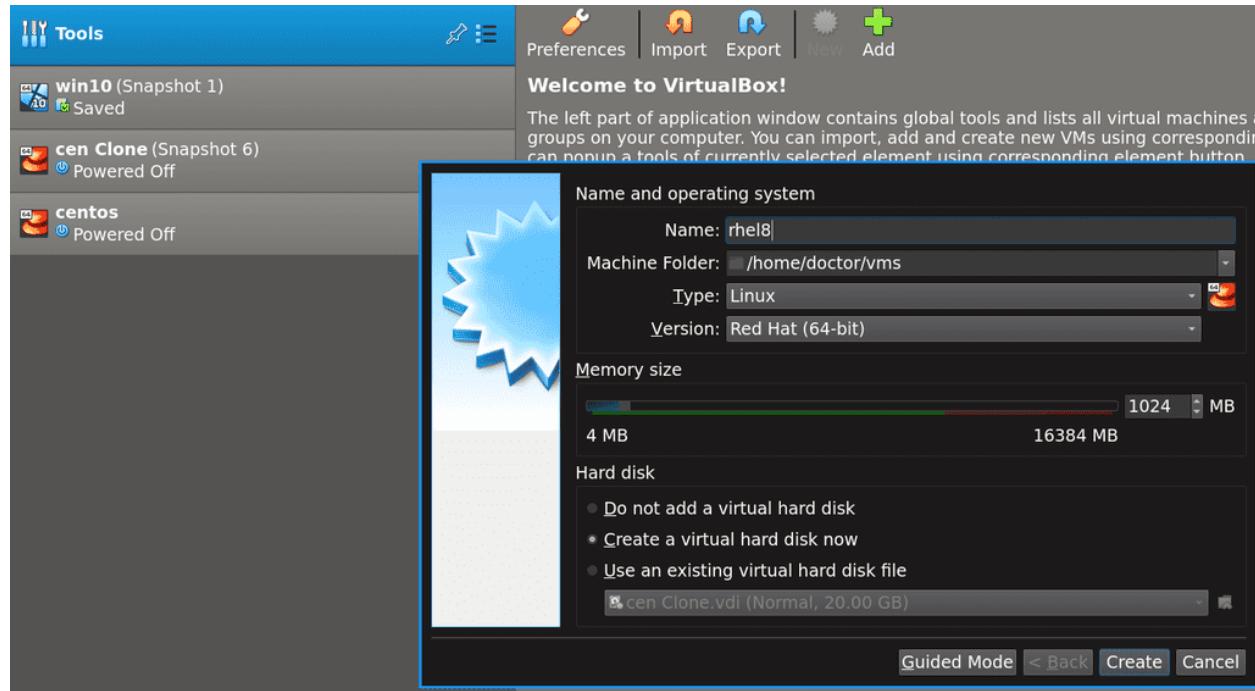
Version	Release Date	Description	Download
8.4.0	2021-05-18	DVD Iso	 x86_64 (9 GB)
		Boot iso	 x86_64 (721 MB)

После чего возвращаемся на [ссылку](#) скачки уже залогиненные. На этот раз скачаем Boot iso. Этот образ установщика весит мало, но всё необходимое ставит из интернета. Так как в дальнейшем мы будем работать с минимальной системой, этого нам вполне хватит.

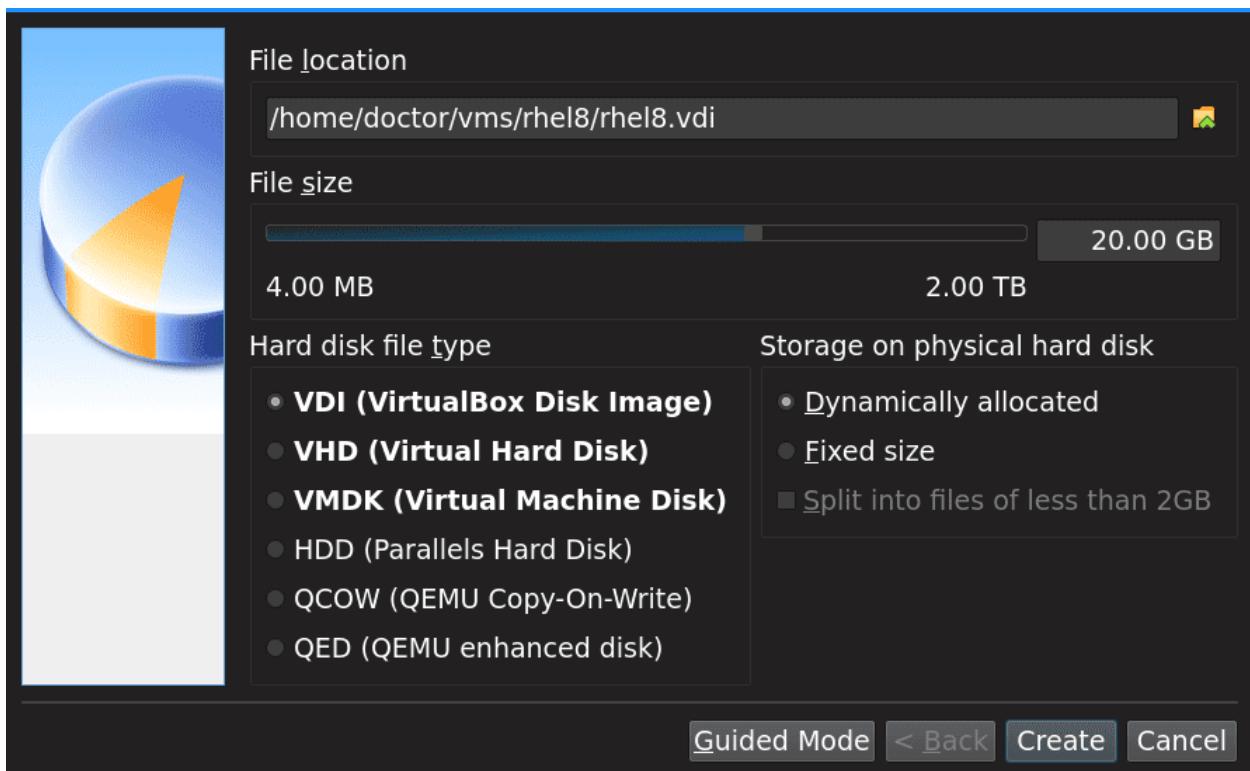


Теперь ждём, пока скачается.

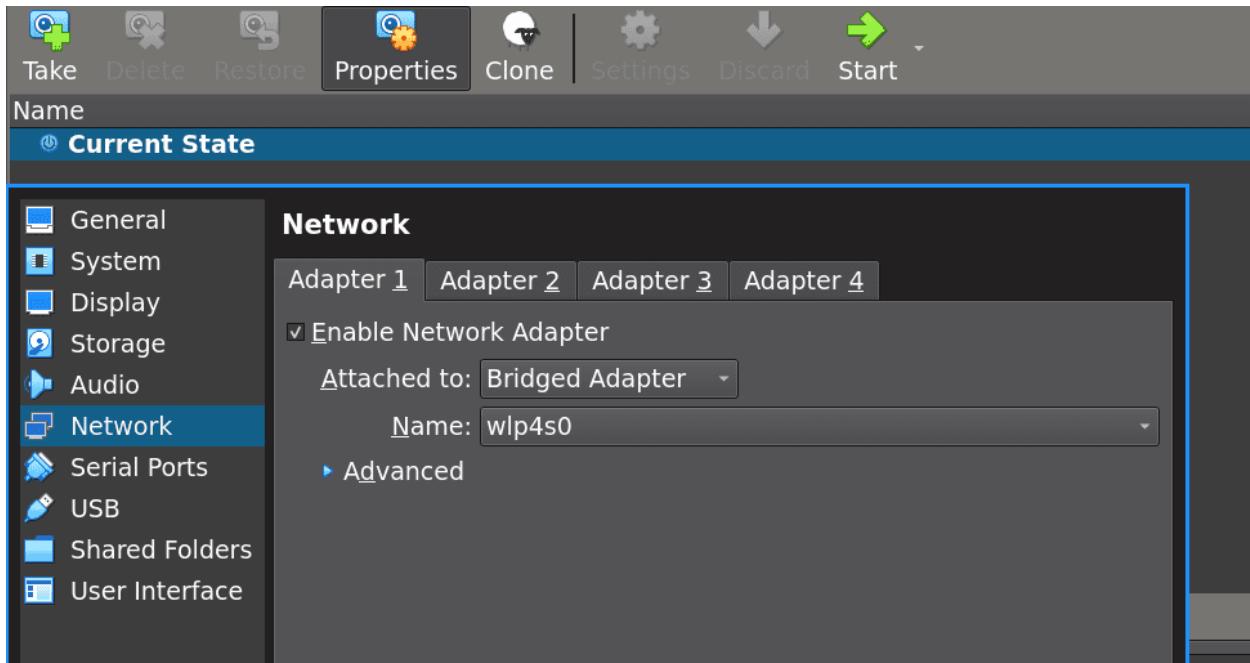
Подготовка VirtualBox



Запускаем VirtualBox, нажимаем New, чтобы создать виртуальную машину и пишем rhel8. При этом автоматом прописывается тип системы и его версия. 1 гигабайта оперативки нам вполне хватит. Также оставляем галочку на создании нового виртуального диска.

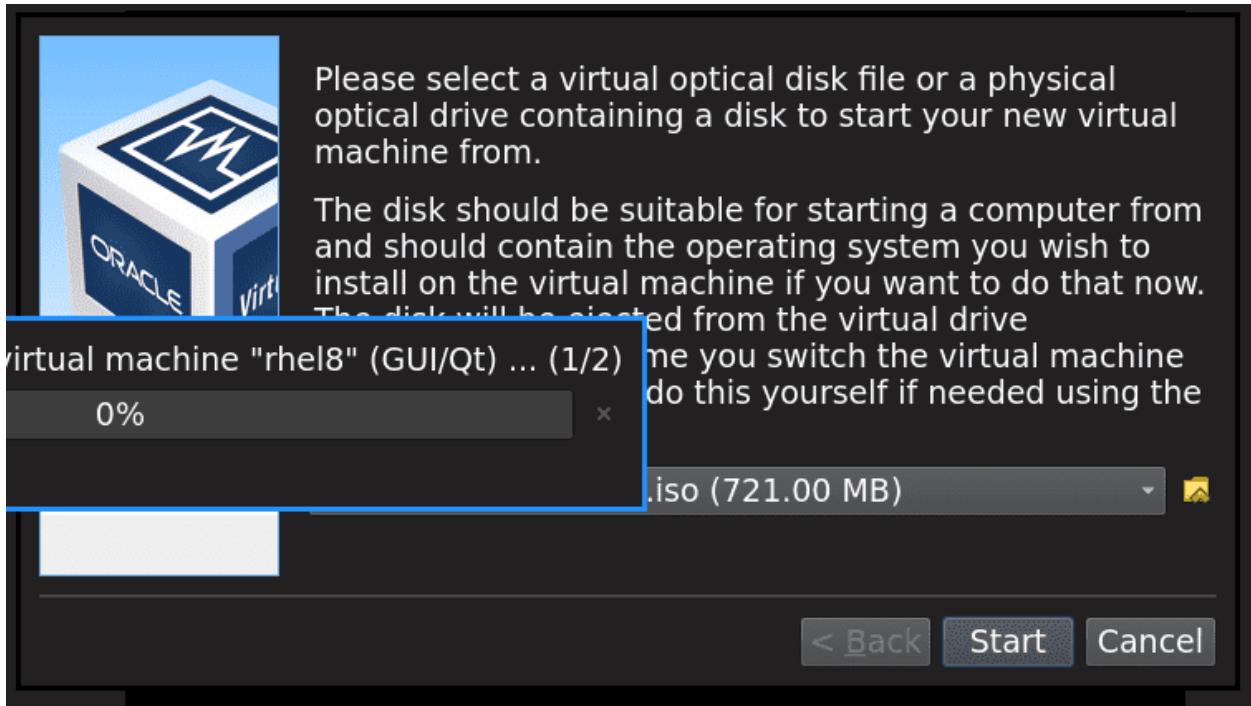


В окне создания диска выбираем размер - 20 Гигабайт. Тип файла оставляем VDI, также оставляем динамическое выделение пространства.



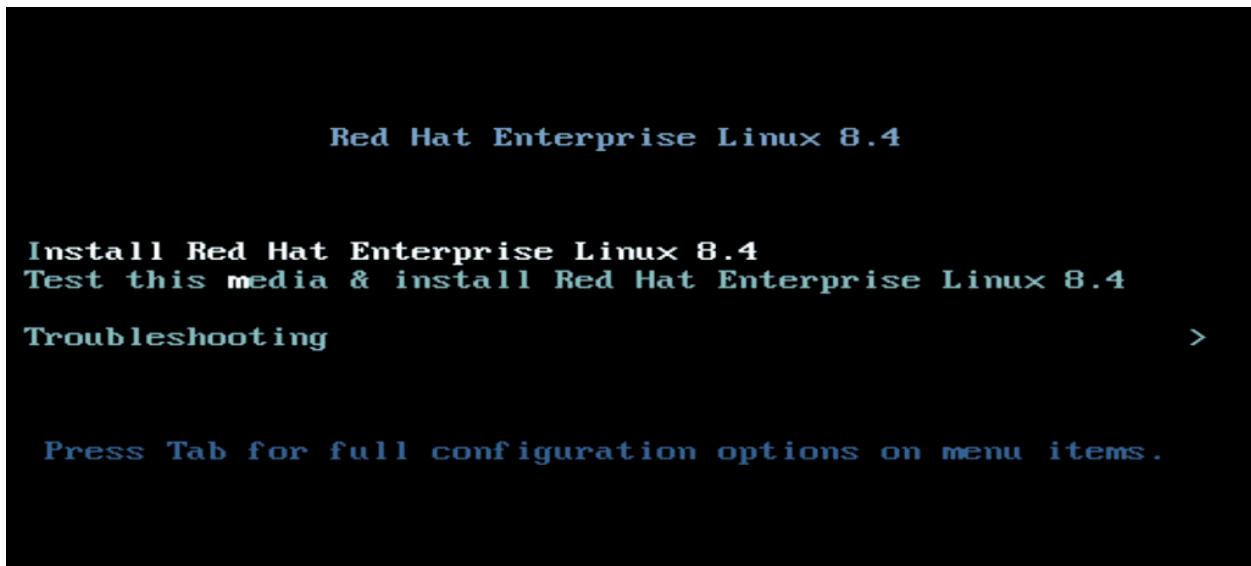
После чего заходим в настройки виртуалки, Network - Adapter 1 и меняем NAT на Bridged, т.е. сетевой мост. В зависимости от того, как вы подключены к сети, под сетевым мостом выбираете ethernet или wifi адаптер. Но, обычно, VirtualBox сам определяет нужный адаптер. После этого нажимаем OK.

Другие настройки нам не нужны. Раньше мы ставили систему с графическим интерфейсом и надо было поднастроить пару параметров, сейчас же графический интерфейс нам не нужен.



Стартуем виртуалку. При этом VirtualBox предложит подцепить ISO образ, находим скачанный файл и выбираем его.

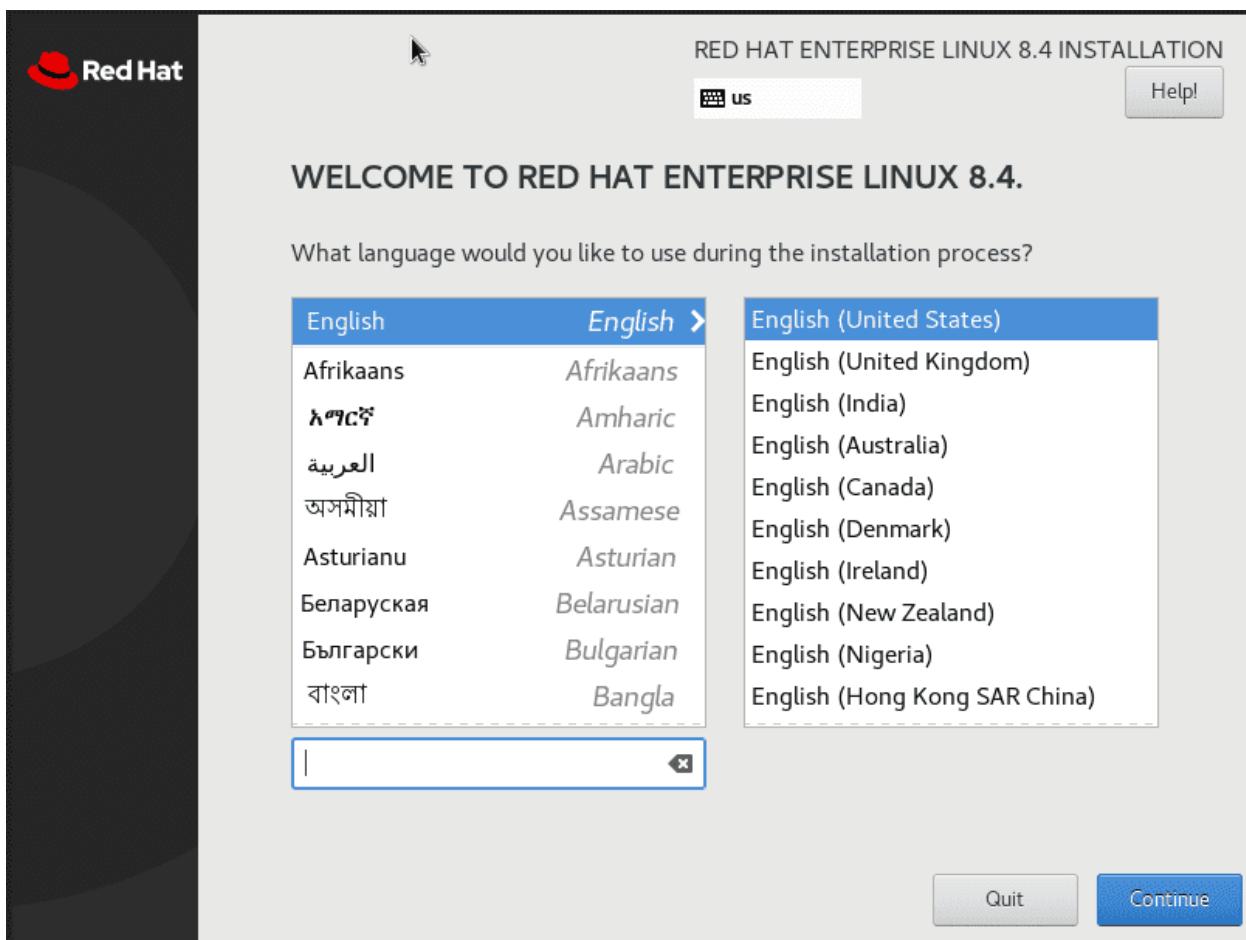
Установка RHEL



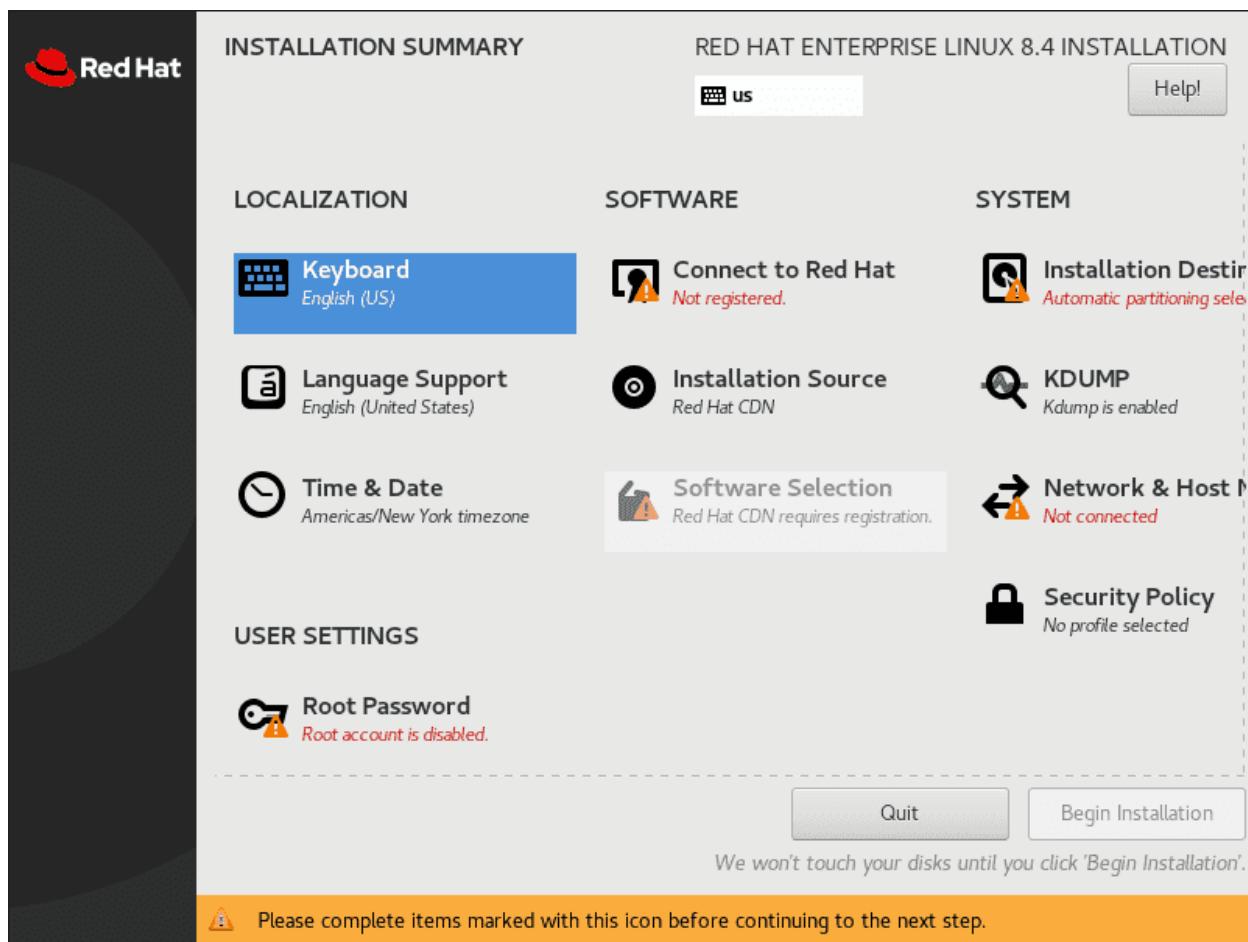
Первое что мы увидим - окно с выбором, где можно:

1. установить систему
2. протестировать установщик на целостность. Обычно нужно, если вы записали ISO образ на оптический диск или флешку и у вас подозрения, что что-то не так записалось.
3. решить какие-то проблемы с установленной системой через troubleshooting.

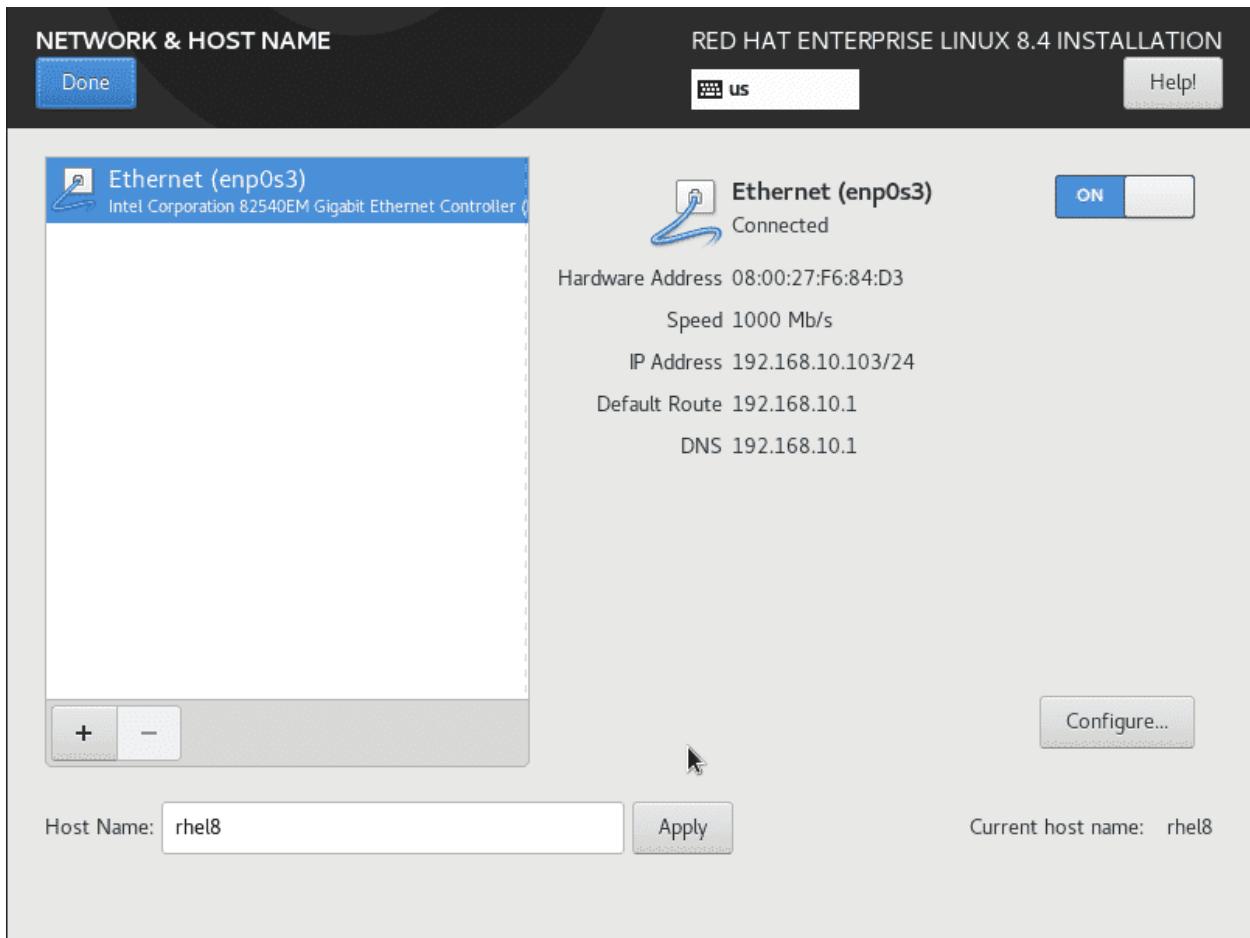
Выберем установку.



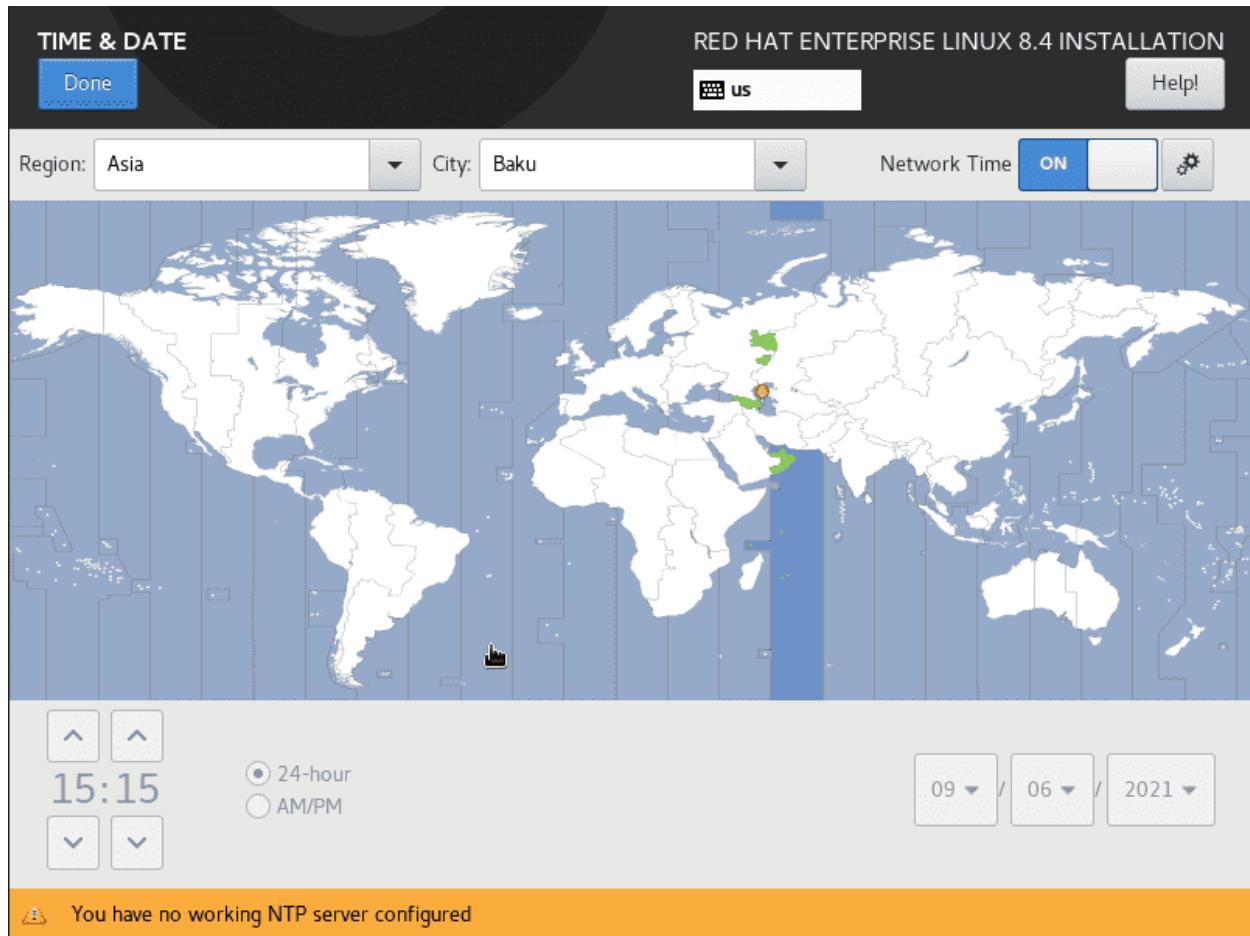
Затем нас встречает окно с предложением выбрать язык установщика. Оставим по умолчанию, английский. Виртуалбокс будет захватывать мышку и, чтобы вернуть мышку на основную систему, следует нажать правый Ctrl.

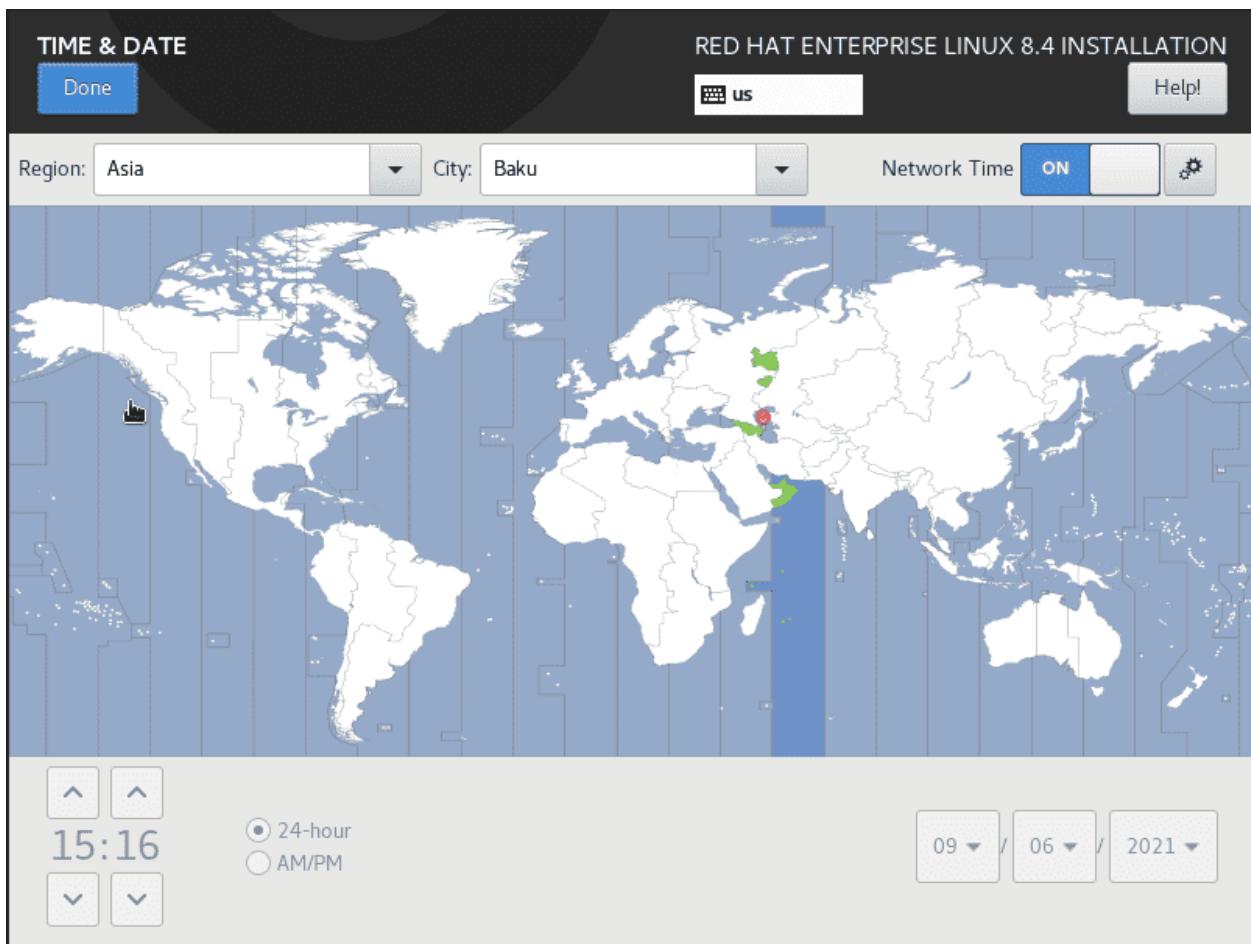


Раскладку клавиатуры и поддержку языков оставим по умолчанию. Наша система будет в качестве сервера, а в подавляющем большинстве случаев на серверах английского языка достаточно.

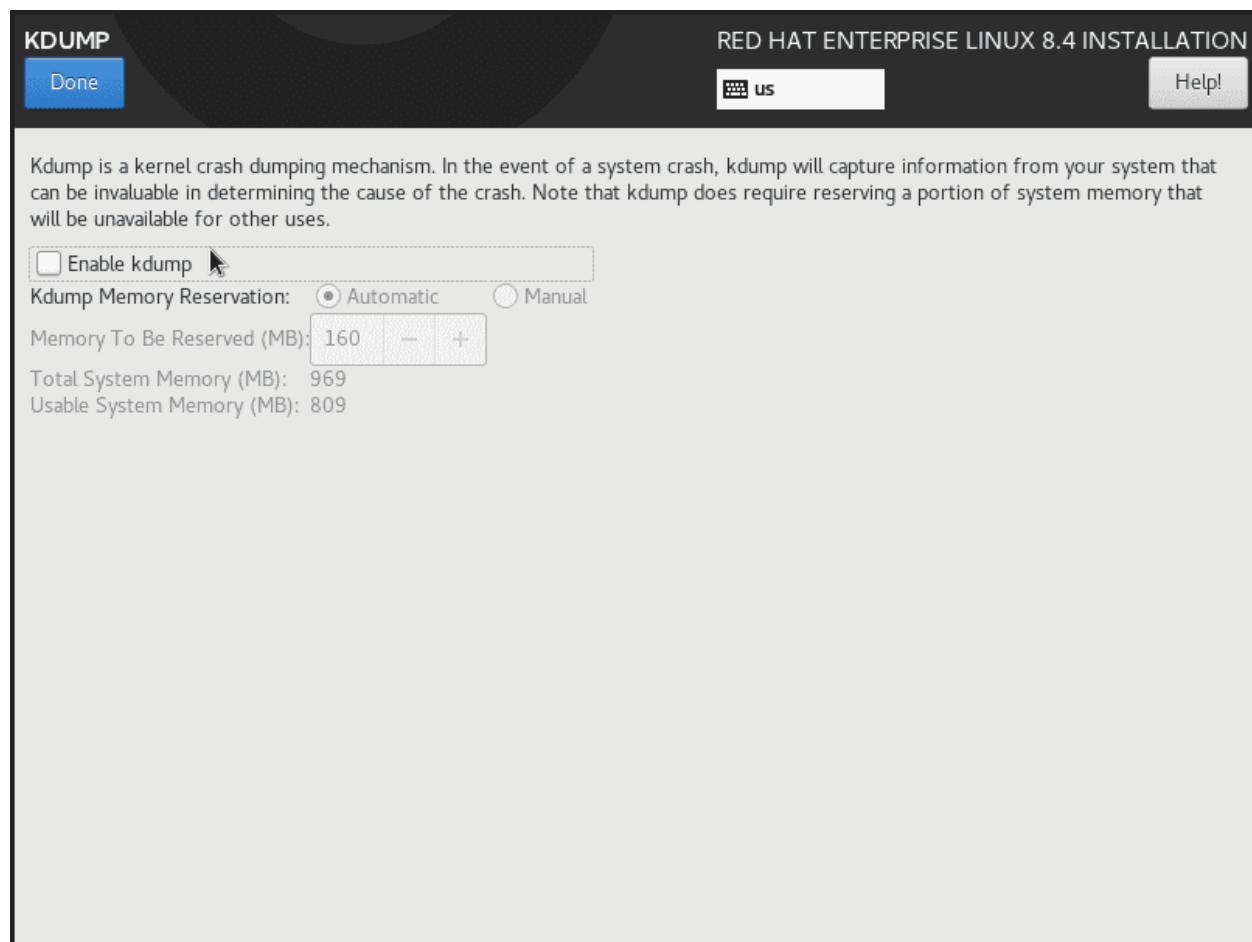


Заходим в Network & Host name, нажимаем галочку на включение адаптера, чтобы виртуалка получила IP адрес по DHCP. Затем снизу пишем hostname - rhel8, нажимаем Apply и Done.

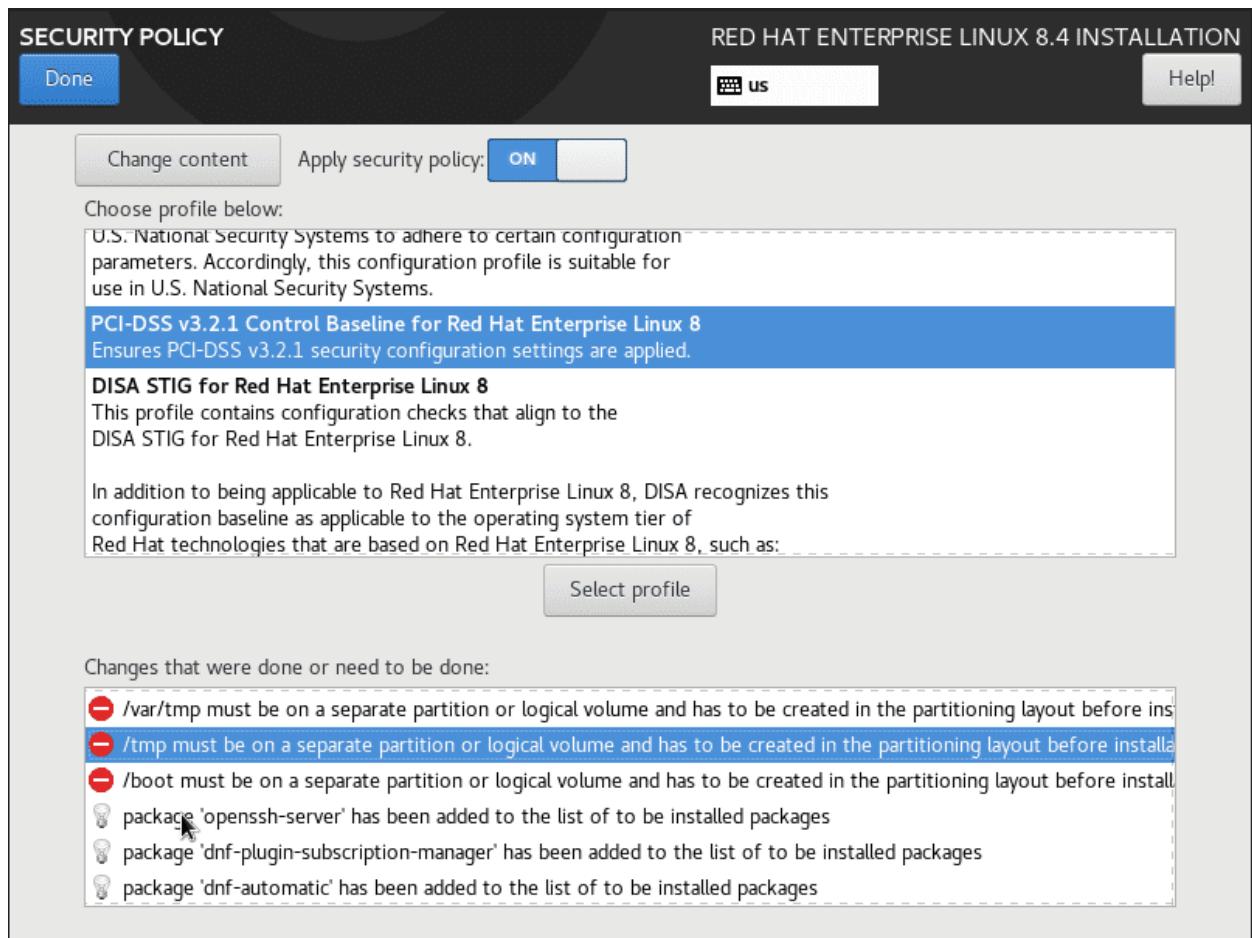


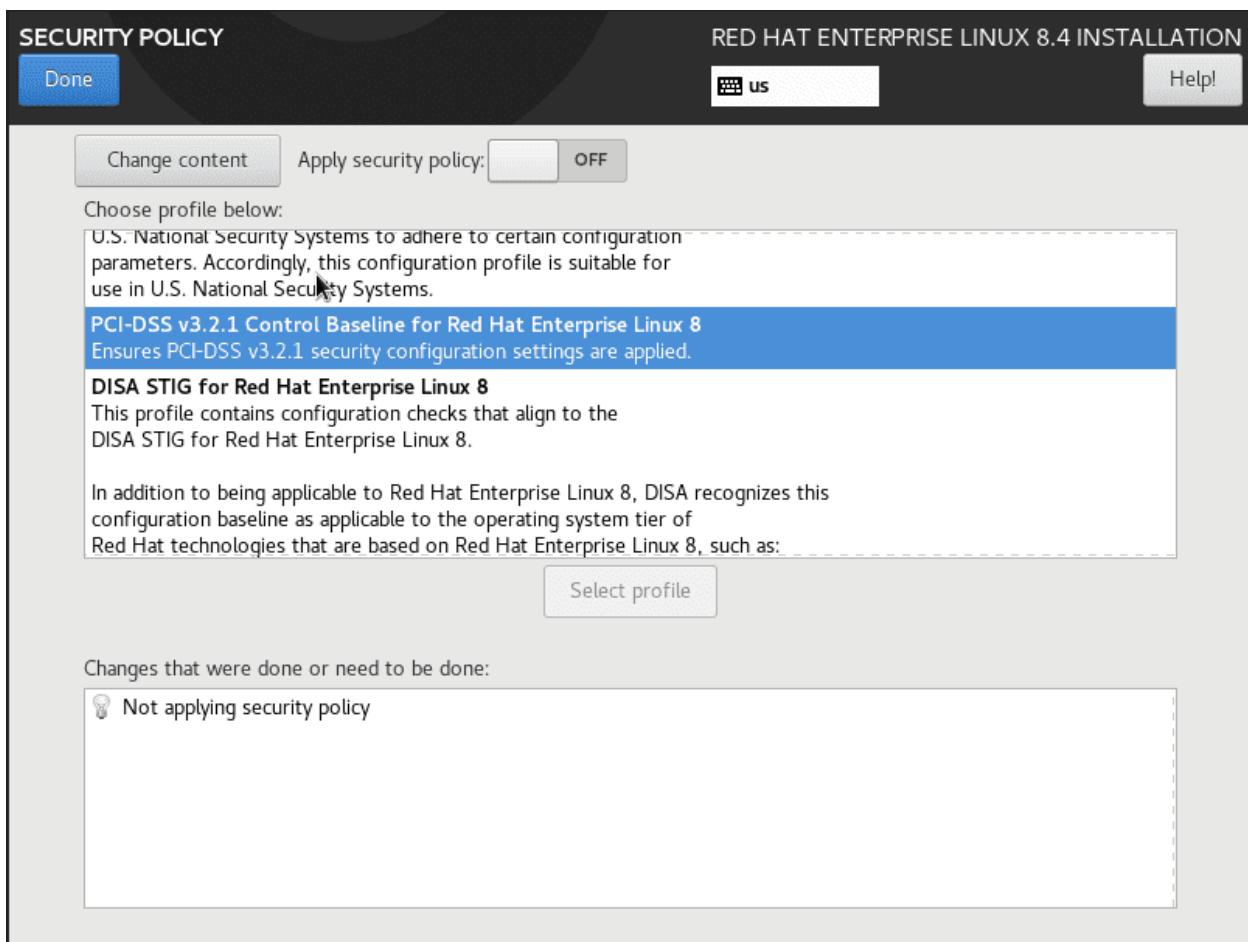


Затем в Time & Date, выбираем наш регион и город. Если у вас снизу отображается уведомление, что NTP сервер не настроен, просто переактивируем галочку Network Time. Это нужно, чтобы система синхронизировала время с сервером из интернета. Иначе в дальнейшем время в системе начнёт различаться с реальным временем, а это может привести к проблемам. Нажимаем Done.



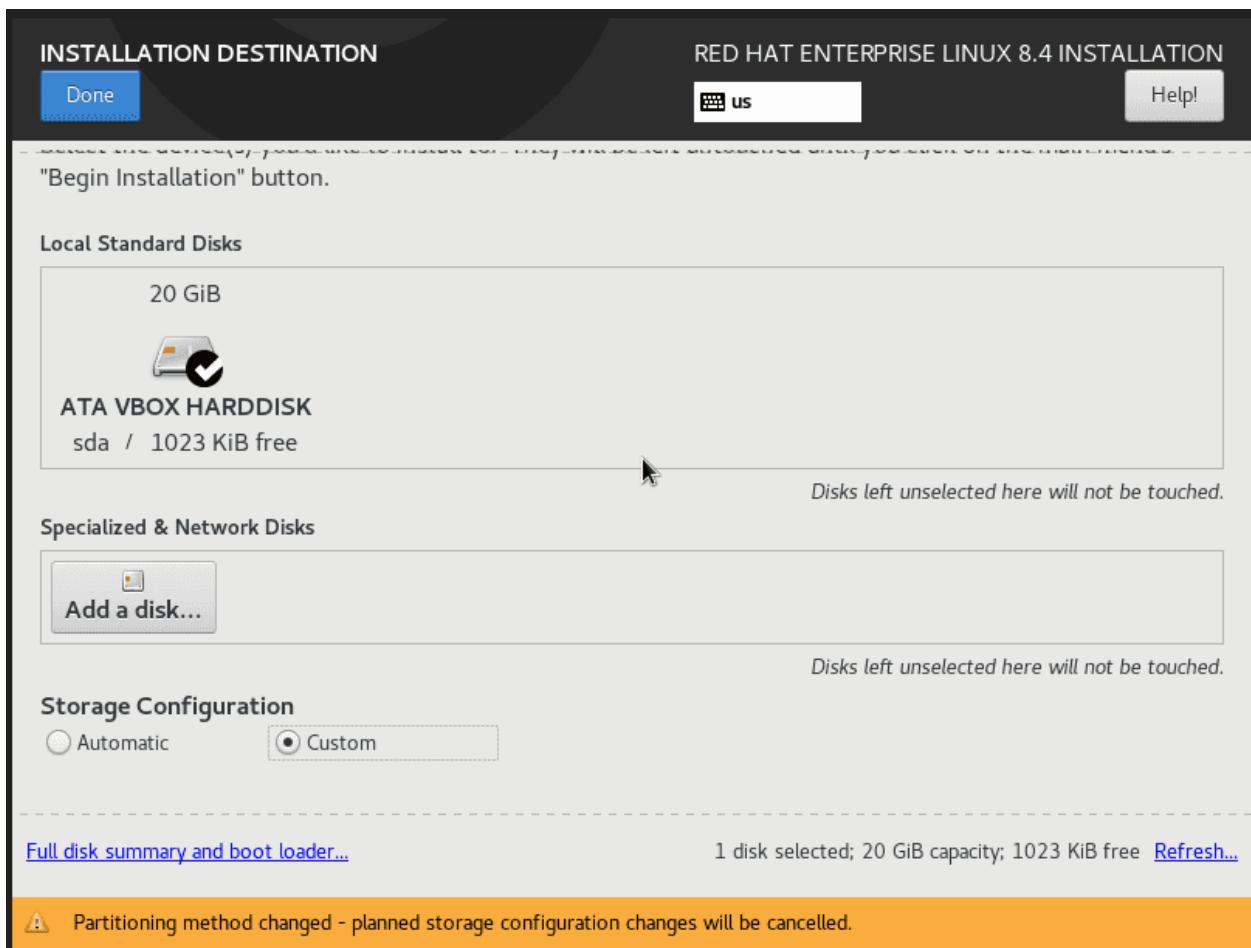
Заходим в KDUMP. Это механизм, который собирает информацию в момент поломки ядра и может быть полезен для диагностирования проблем. В целом это полезная штука и стоит её оставлять включённой в рабочей среде, чтобы, в случае чего, решать проблемы с ядром. Но kdump резервирует себе оперативку, а у нас тестовая среда для обучения, проблемы с ядром нам пока не страшны и маловероятны. Нам куда важнее ресурсы, поэтому отключим его, убрав галочку «Enable kdump» и нажав Done.



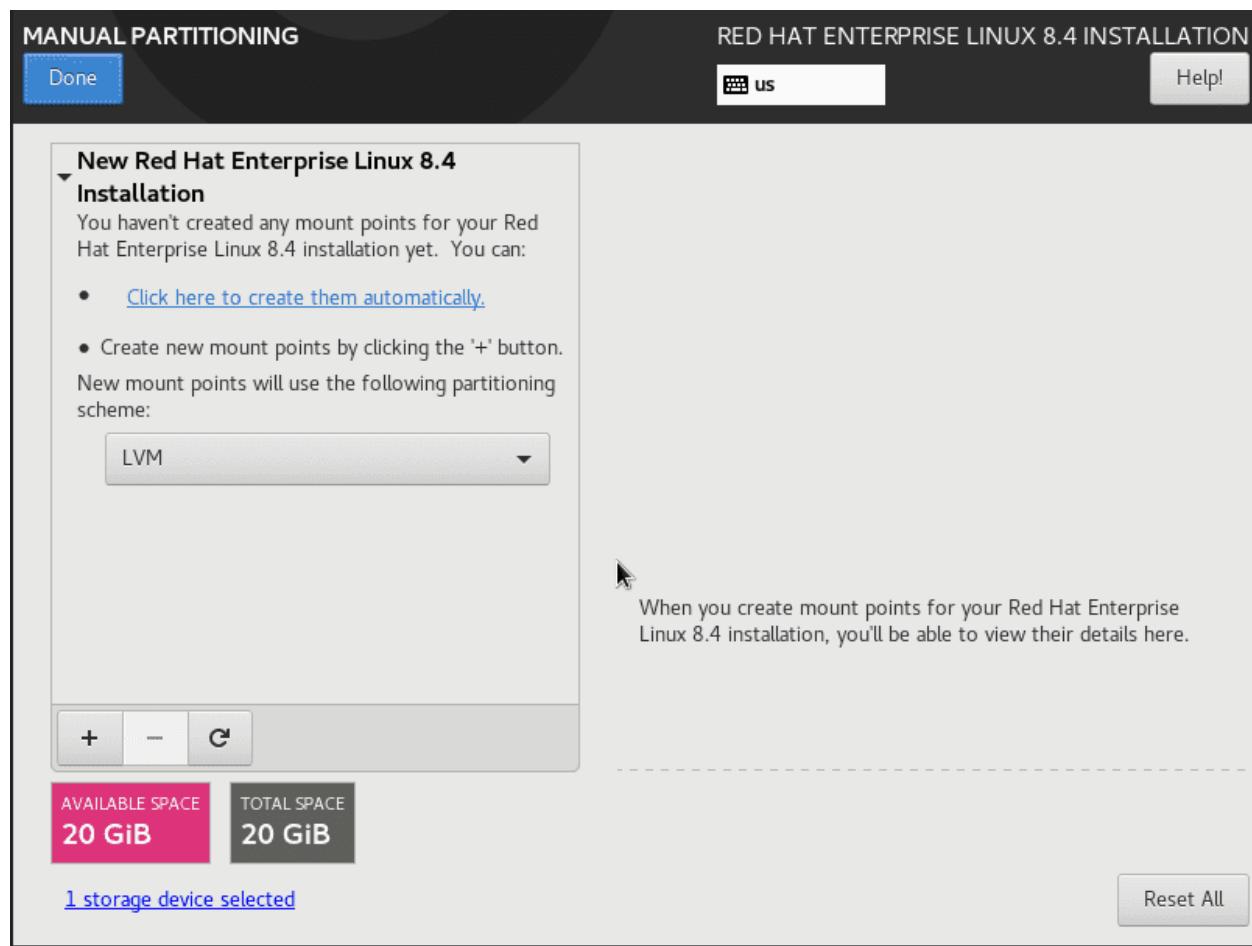


Заходим в Security Policy. Здесь у нас политики безопасности. Есть определённые стандарты настроек системы, особенно с точки зрения отказоустойчивости и безопасности, которым должны соответствовать различные организации и компании. К примеру, различные банки должны соответствовать стандартам PCI-DSS. В этих стандартах говорится как об организации инфраструктуры, так и конкретных настройках различных программ, к примеру, ssh демона, на какие файловые системы должен быть разделён диск, какие параметры должны быть у bash-а и т.д. и т.п. Соответствие стандартам полезно и предотвращает многие проблемы. Но настройка всего этого может занимать много времени, а дальнейшее администрирование настроенной системы немножко усложняется, так как появляется много ограничений с точки зрения безопасности. Возможно, когда-нибудь, мы пройдёмся по какому-нибудь стандарту, но нам пока рано. В этом окне мы можем выбрать один из стандартов и применить его, чтобы свежеустановленная система сразу соответствовала стандартам. Но, пока мы учимся, это будет мешать, поэтому отключим «Apply Security Policy» и нажмём Done.

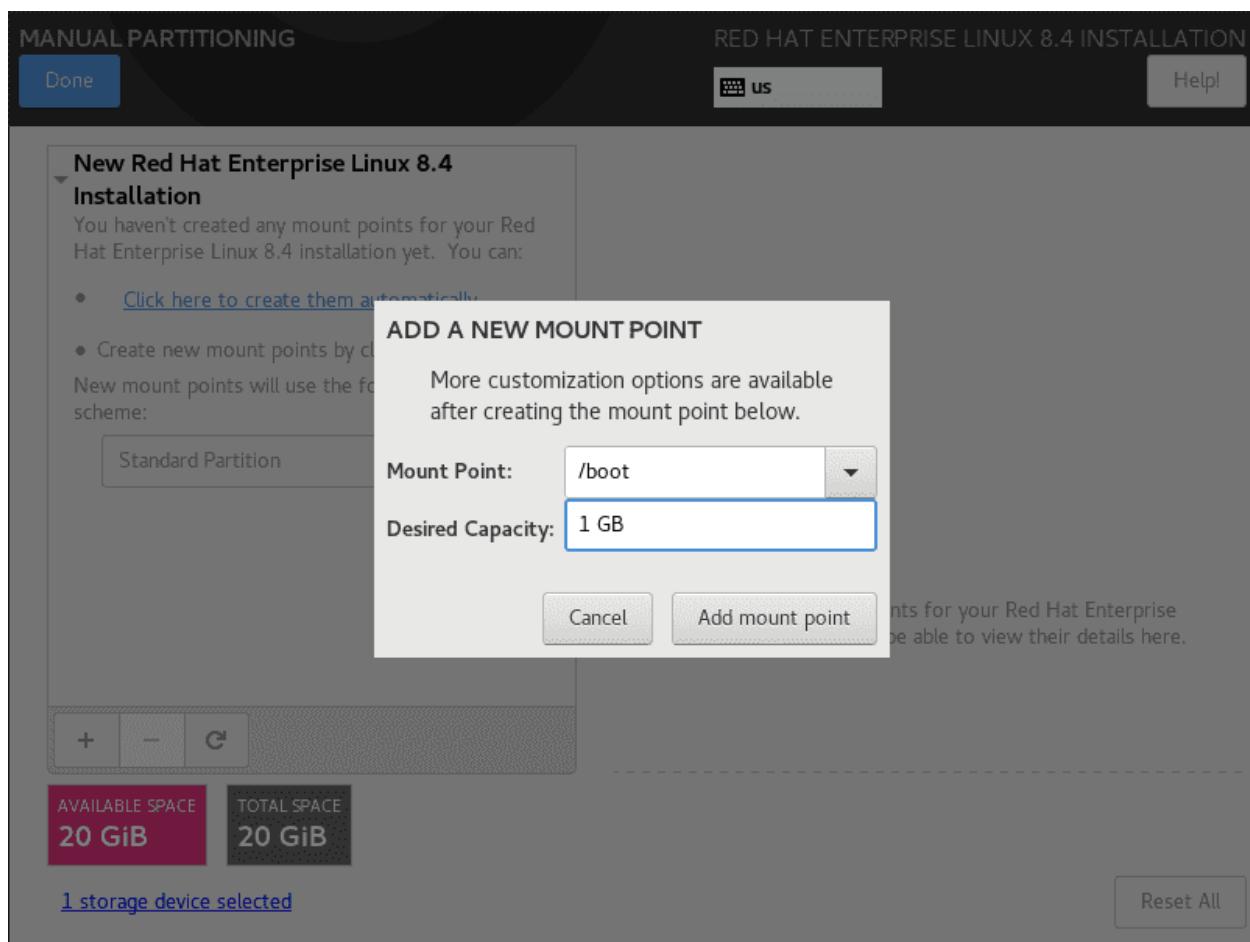
Разметка диска



Теперь приступим к разметке диска - Installation Destination. Мы уже много говорили о дисках и файловых системах, поэтому вместо автоматической разметки можем выбрать «Custom», чтобы настроить самостоятельно. Это важно, потому что в большинстве случаев автоматическая разметка может не подходить под наши задачи. Нажмём Done.



В этом окне можно гибко и удобно настроить разделы и файловые системы. И так, предположим, что мы настраиваем сервер. Зачастую полезно держать операционную систему и всё что ей нужно на одном диске, а данные, с которыми будет работать сервер - на другом. Это позволит в дальнейшем свободно перемещать данные на другую систему просто отключив диск с этой виртуалки и подключив на другую, а также заменить операционку в случае проблем, не затронув данные. Сейчас мы делаем шаблонную виртуалку, и, если что, подключим новый диск, где и будем хранить данные.

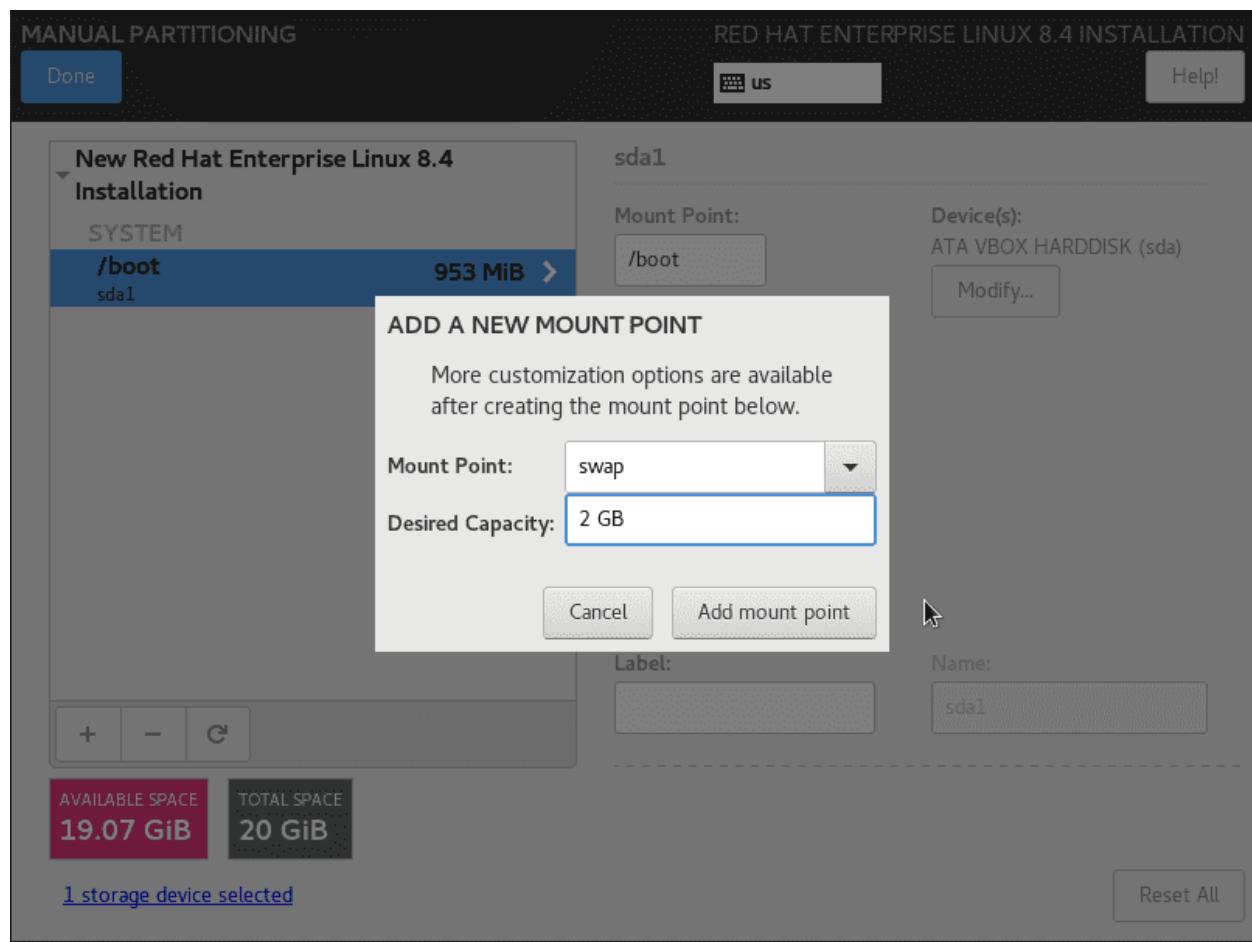


Из темы о LVM мы помним, что со стандартными разделами работать очень неудобно, и, если завтра будет не хватать места, что на серверах происходит очень часто, простым способом это не получится решить. Поэтому нам обязательно нужен LVM. Из темы о загрузчике мы помним, что всю систему можно хранить на LVM, если у нас используется GRUB2. Но Red Hat настоятельно рекомендует держать директорию /boot на стандартном разделе. Поэтому начнём с неё - в выпадающем меню выбираем Standard Partition и нажимаем +. В окне в качестве Mount Point выбираем /boot. В этой директории у нас хранится ядро, initramfs и настройки grub-a, что занимает мало пространства. Но при обновлении сюда же добавляется новое ядро, новый initramfs и если здесь место забьётся, возникнут проблемы. По умолчанию, при обновлении пакетный менеджер сохраняет старую версию ядра и initramfs, чтобы, в случае проблем, можно было загрузиться со старого ядра. И, по умолчанию, сохраняются 3 версии ядра - текущее, предыдущее и предпредыдущее. Ну и при установке новой версии появляется ещё и четвёртое, но потом самое старое ядро удаляется. На 4 версии ядра примерно хватит 1 GB, поэтому столько и выделим. Но если вы будете что-то делать с ядром, генерировать свой initramfs и т.п., проследите, чтобы /boot не забился.

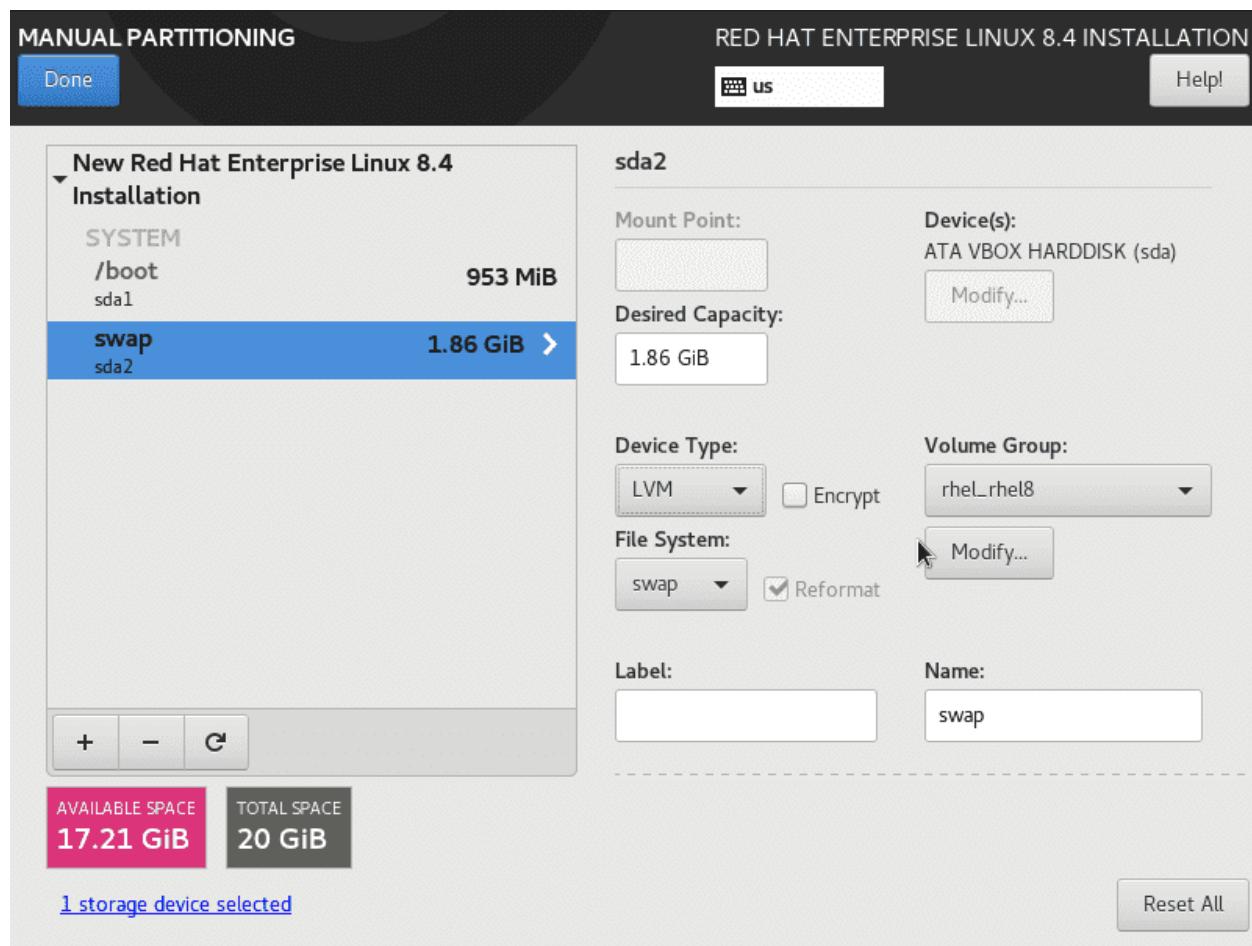
Нажимаем Add mount point.

Amount of RAM in the system	Recommended swap space	Recommended swap space if allowing for hibernation
≤ 2 GB	2 times the amount of RAM	3 times the amount of RAM
> 2 GB – 8 GB	Equal to the amount of RAM	2 times the amount of RAM
> 8 GB – 64 GB	At least 4 GB	1.5 times the amount of RAM
> 64 GB	At least 4 GB	Hibernation not recommended

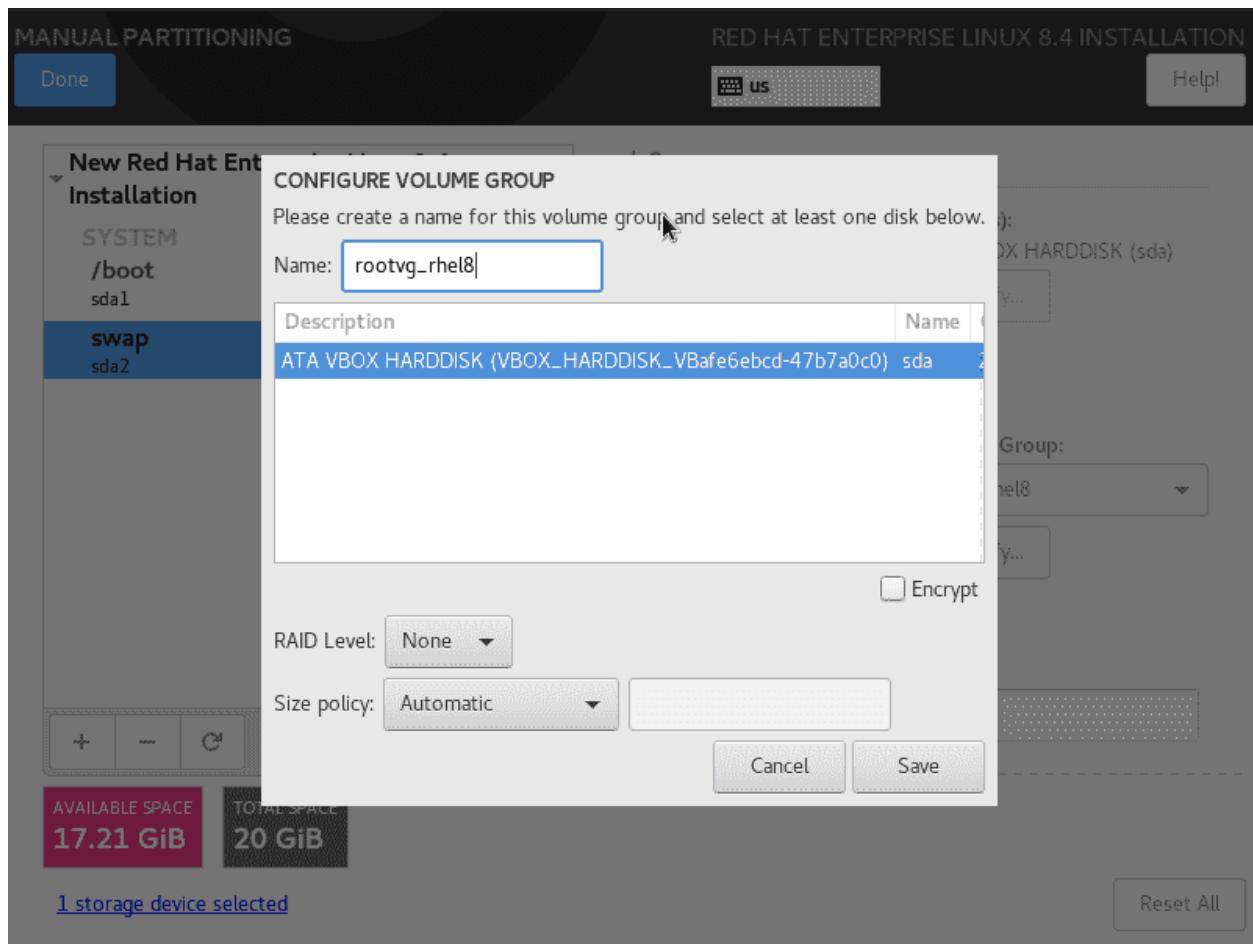
Теперь добавим swap. Тут всё очень индивидуально и зависит от сервиса, который мы будем поднимать на этом сервере. Советую посмотреть тему про swap, чтобы лучше понимать, зачем это нужно. Оттуда же возьмём шаблонную табличку и найдём значение для нашей виртуалки. И так, у нас оперативки всего 1 GB - под это подходит первая строчка. Гибернацию на серверах обычно не используют, поэтому под swap хватит 2 GB.



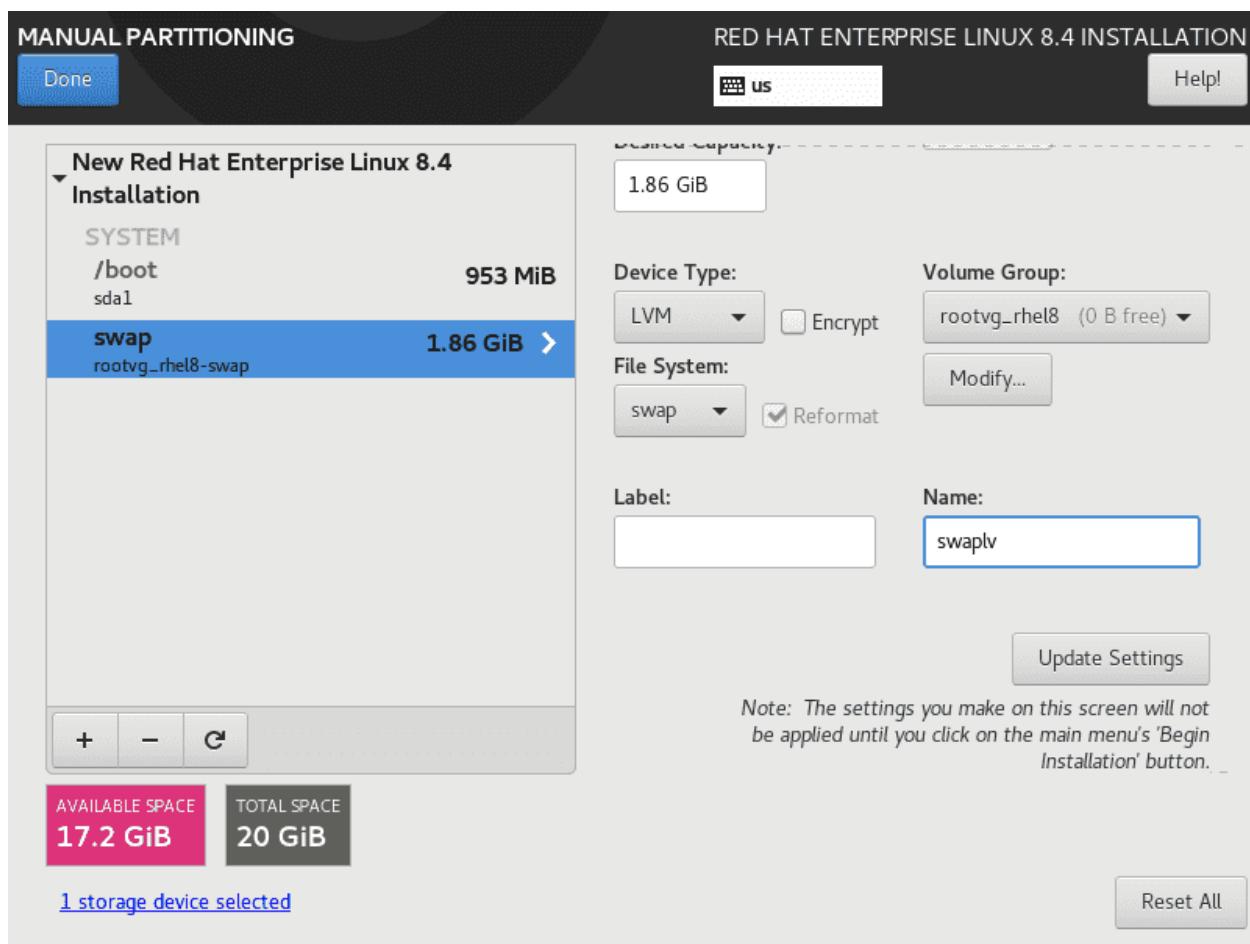
Нажимаем +, выбираем в качестве «Mount Point» swap, и capacity - 2 GB.



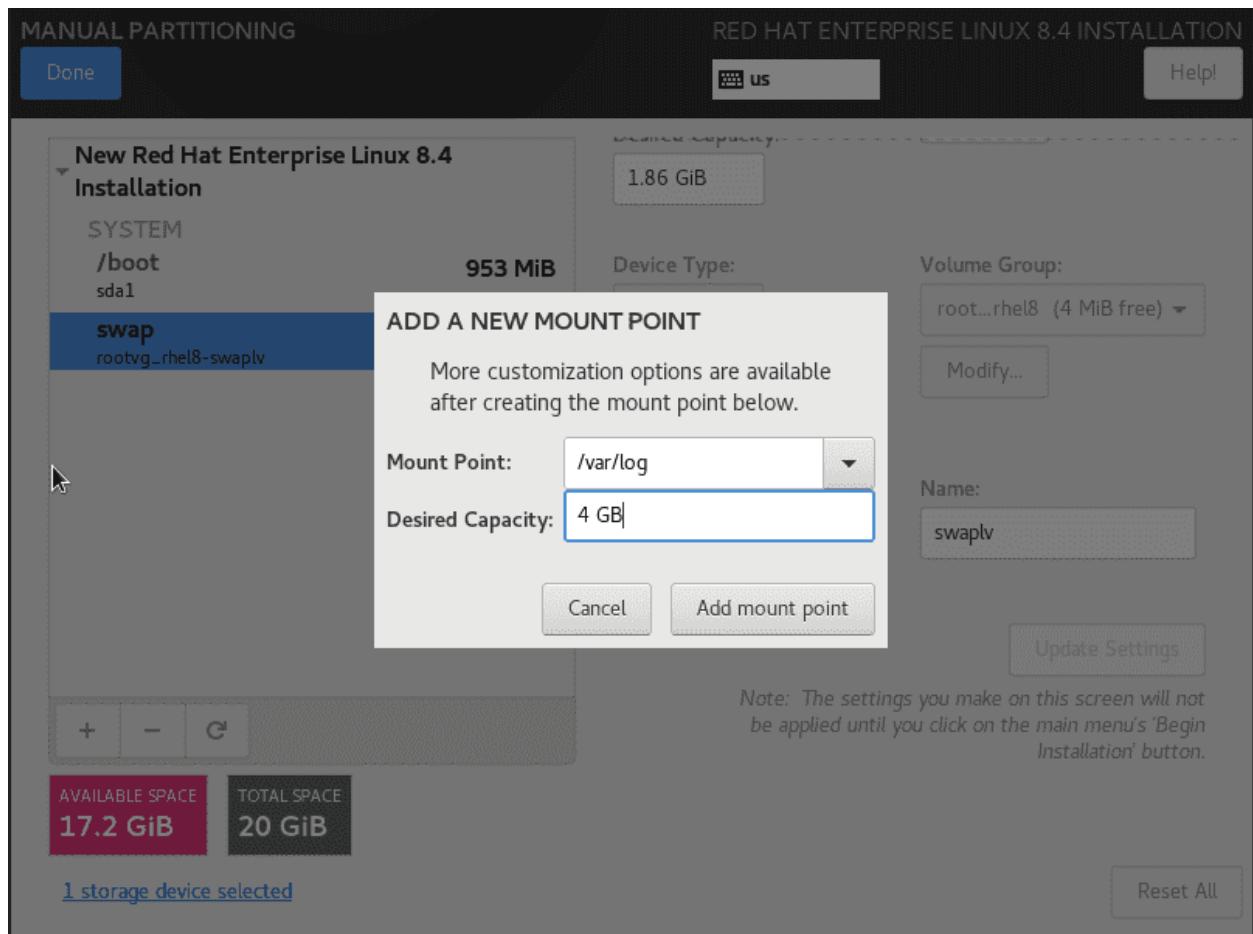
Но мы предполагаем, что в будущем может понадобится больше оперативки, и, соответственно, swap-а. И чтобы мы могли без проблем увеличить swap, следует его сделать на LVM. Для этого в Device Type выбираем LVM. Справа у нас появится название Volume Group-ы - rhel_rhel8. Обычно я придерживаюсь своего стандарта в названии томов, поэтому переименую. Нажимаем «Modify...»



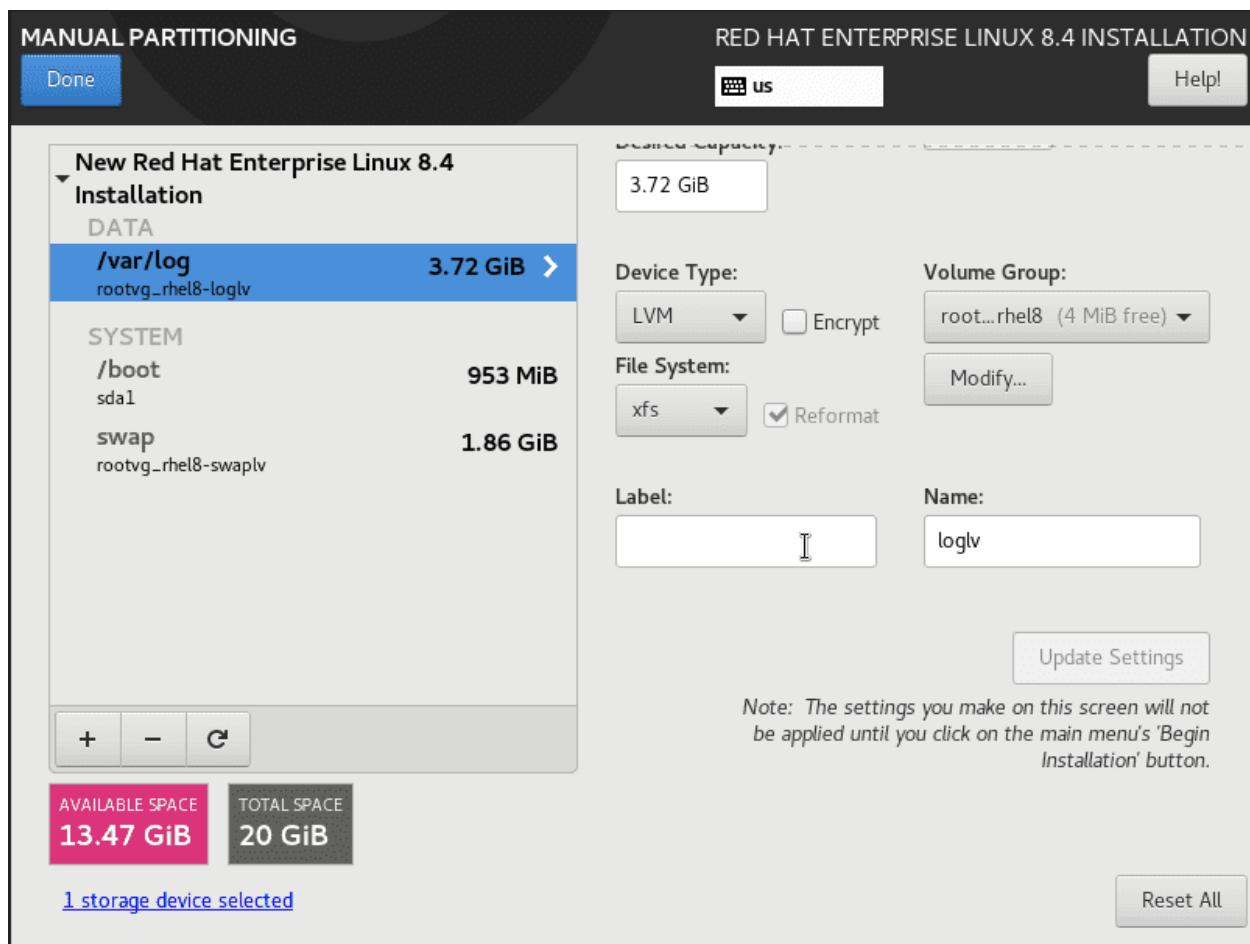
И так, VG, где у меня хранится корень и всё что относится к системе, я называю rootvg, а VG с данными - datavg. Но, если в будущем понадобится подключить этот диск к другому компьютеру, где уже есть VG с таким же названием, возникает проблема с LVM из-за одинаковых названий. Поэтому к rootvg я приписываю хостнейм - rootvg_rhel8. Нажимаем Save.



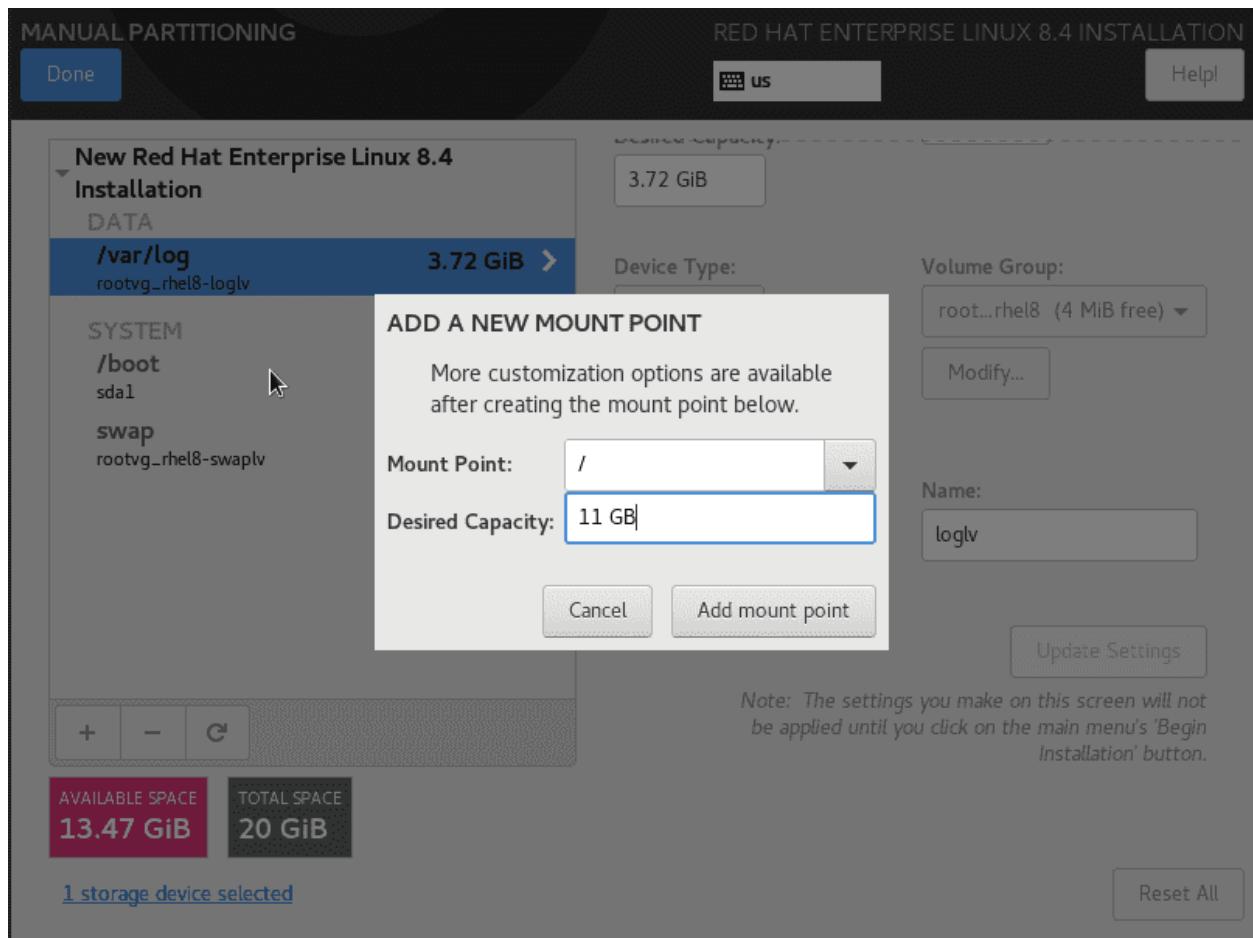
Ниже Volume Group есть Name - имя логического тома. К нему я обычно приписываю lv - swaplv. Если экран установщика маленький, нужно навести мышку направо и используя ползунок спуститься вниз, чтобы появилась кнопка «Update Settings». После наших изменений надо на неё нажать.



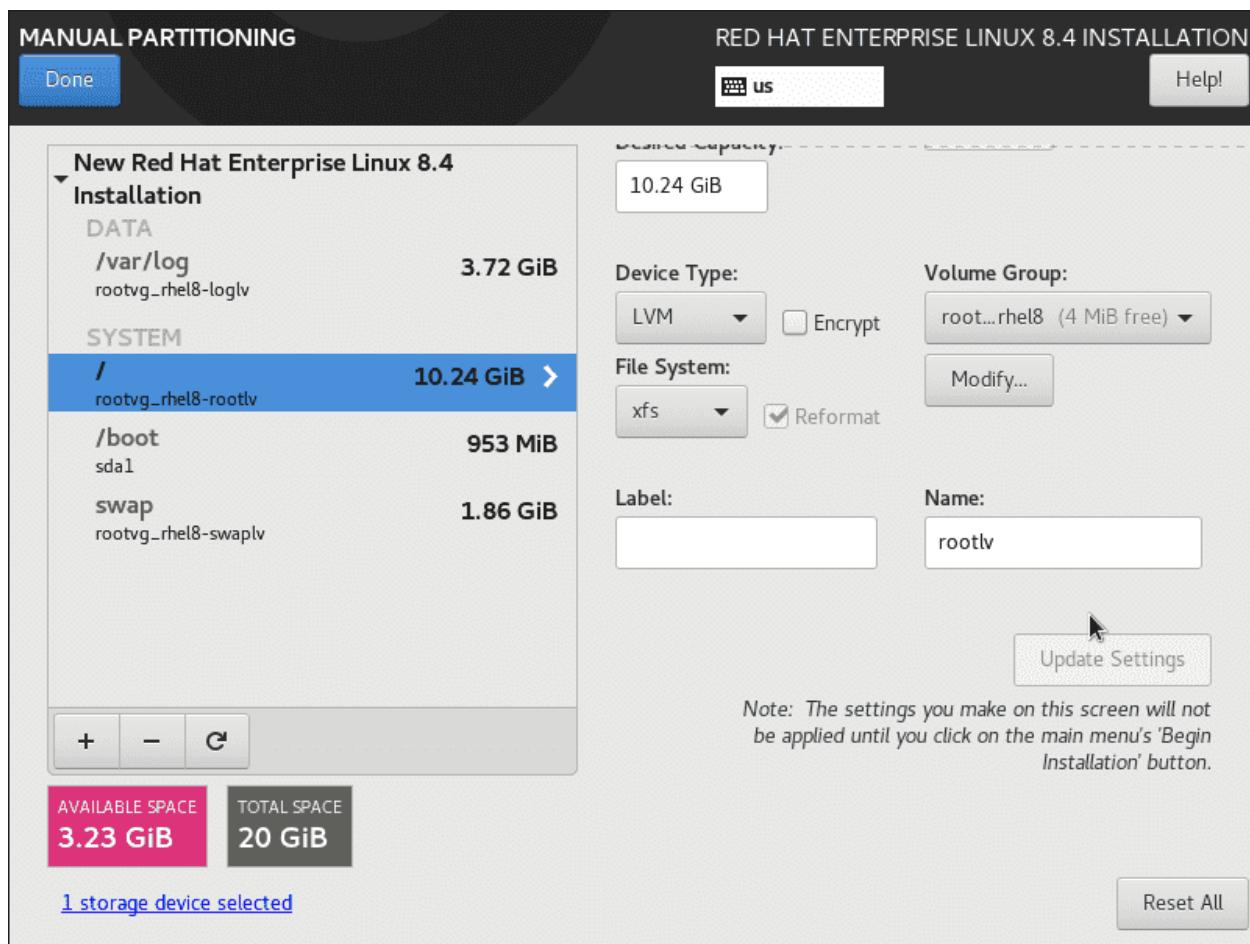
Ещё раз нажимаем **+**. Так как эта виртуалка у нас в качестве сервера и, скорее всего, здесь у нас не будет реальных пользователей и их файлов, **/home** выносить в отдельную файловую систему не надо. Бывают сервисы, которые хранят свои данные в домашней директории своего юзера - тогда имеет смысл отделить **/home**. Но в нашем случае это не надо. Зато на серверах имеет смысл отделять от корня директорию **/var/log**. При каких-то проблемах логи могут быстро накопиться и забить весь корень, из-за чего система перестанет работать. Выделив под логи отдельную файловую систему мы можем обезопасить корень. Сколько выделять под логи? Зависит от многих факторов - как долго вам нужно хранить логи, как много их пишется за день. Зачастую создаётся отдельный лог сервер для центрального хранения логов - тогда на самих машинках хранить много логов не нужно. Выставим 4 GB - для тестов этого объёма вполне хватит. И если что - всё равно мы используем LVM и сможем увеличить.



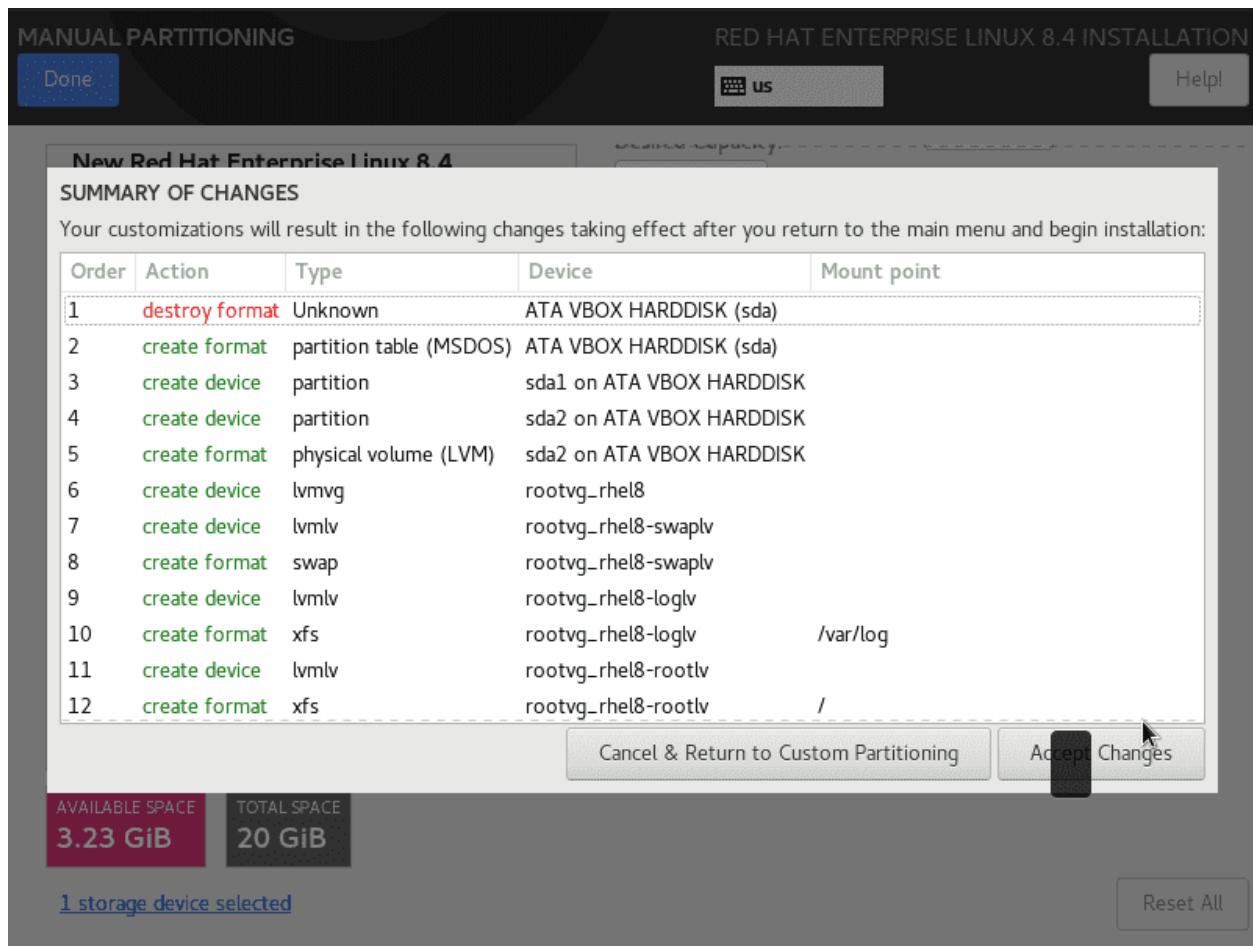
Появившийся стандартный раздел поменяем на LVM в Device Type. На этот раз менять volume group не надо, название уже сохранилось. Осталось поменять название логического тома - loglv и нажать Update Settings.



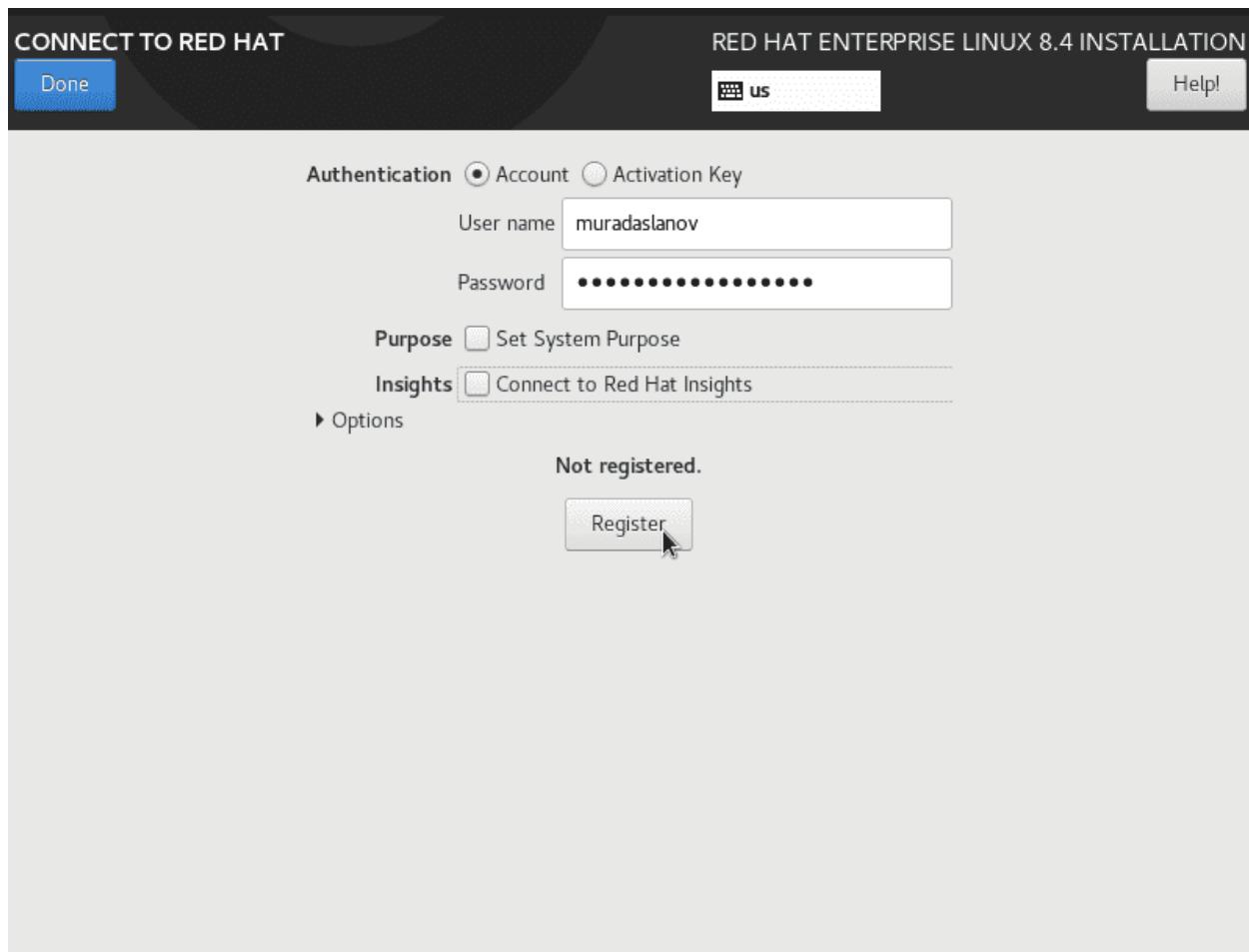
Ещё раз нажимаем +. Осталось только создать логический том для корня. В некоторых случаях имеет смысл выносить и другие директории, но это уже частности. И так, выбираем в качестве Mount Point корень - /. Свободного объёма осталось где-то 13 GB. Под корень выдадим 11 GB - остальное оставим свободным. Если завтра нам где-то перестанет хватать места или мы захотим сделать снапшот, немного лишнего места нас могут спасти.



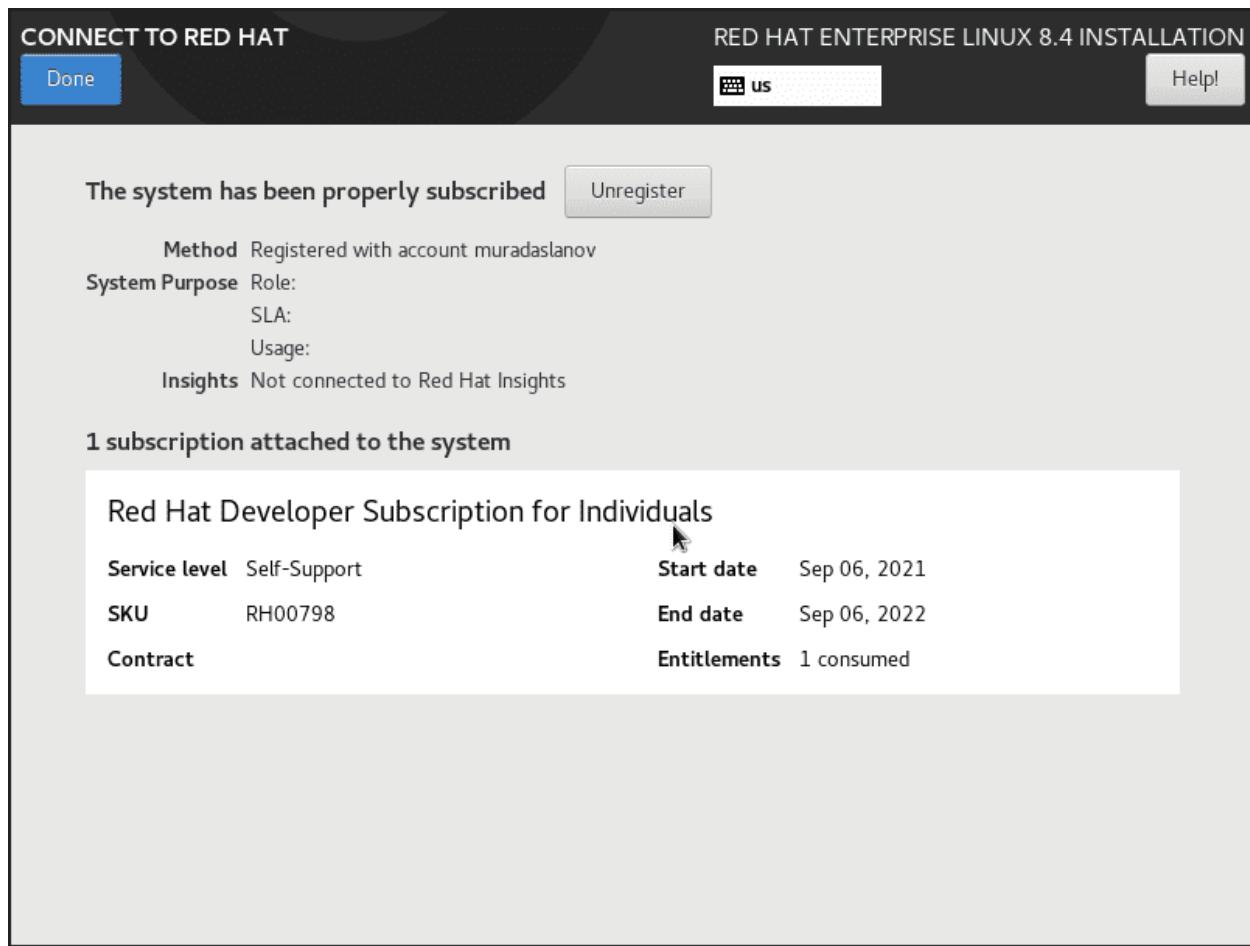
Опять же, выбираем тип - LVM, меняем название на rootlv и нажимаем Update Settings. После чего нажимаем Done.



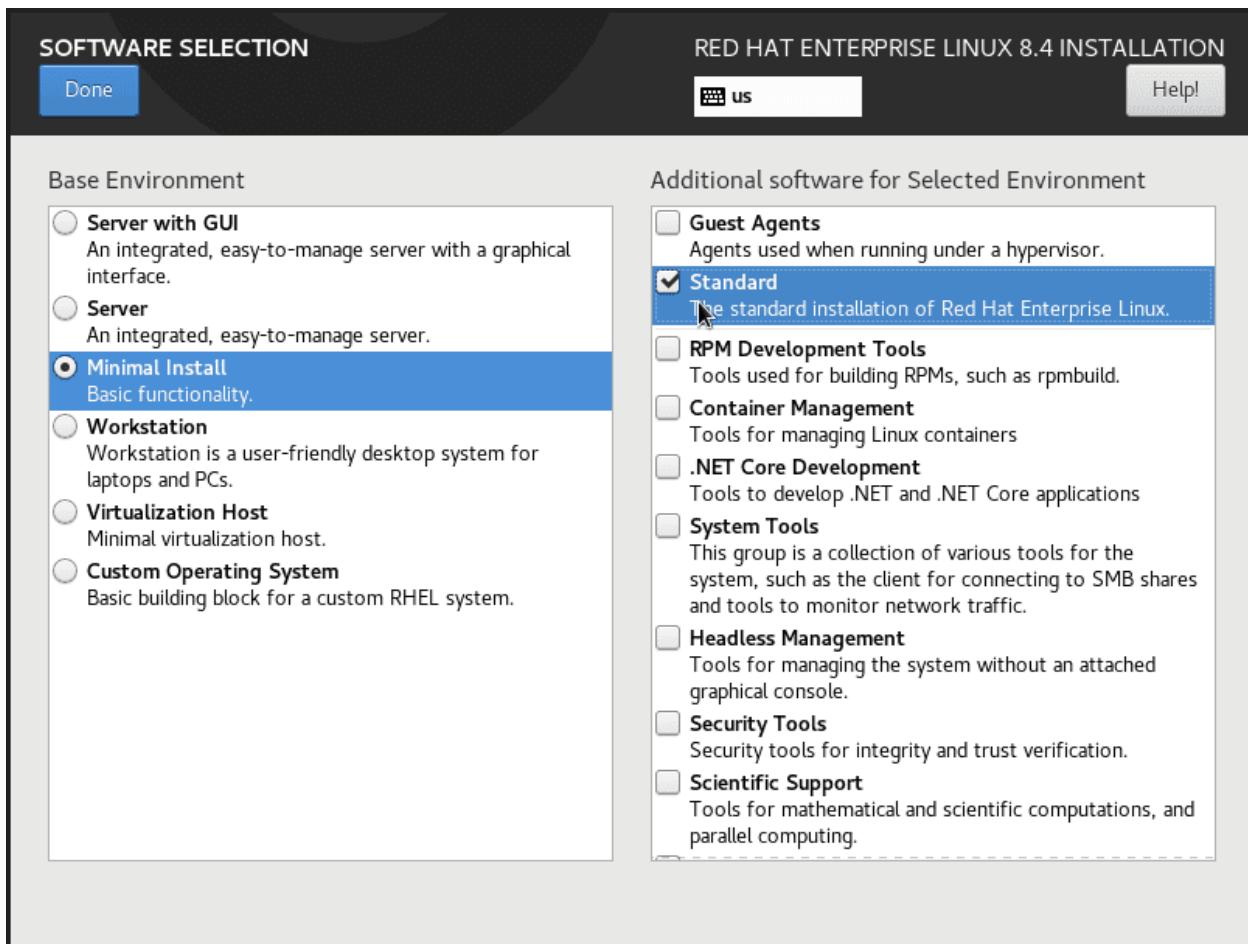
Установщик покажет, что к чему и предложит принять наши настройки - Accept changes.



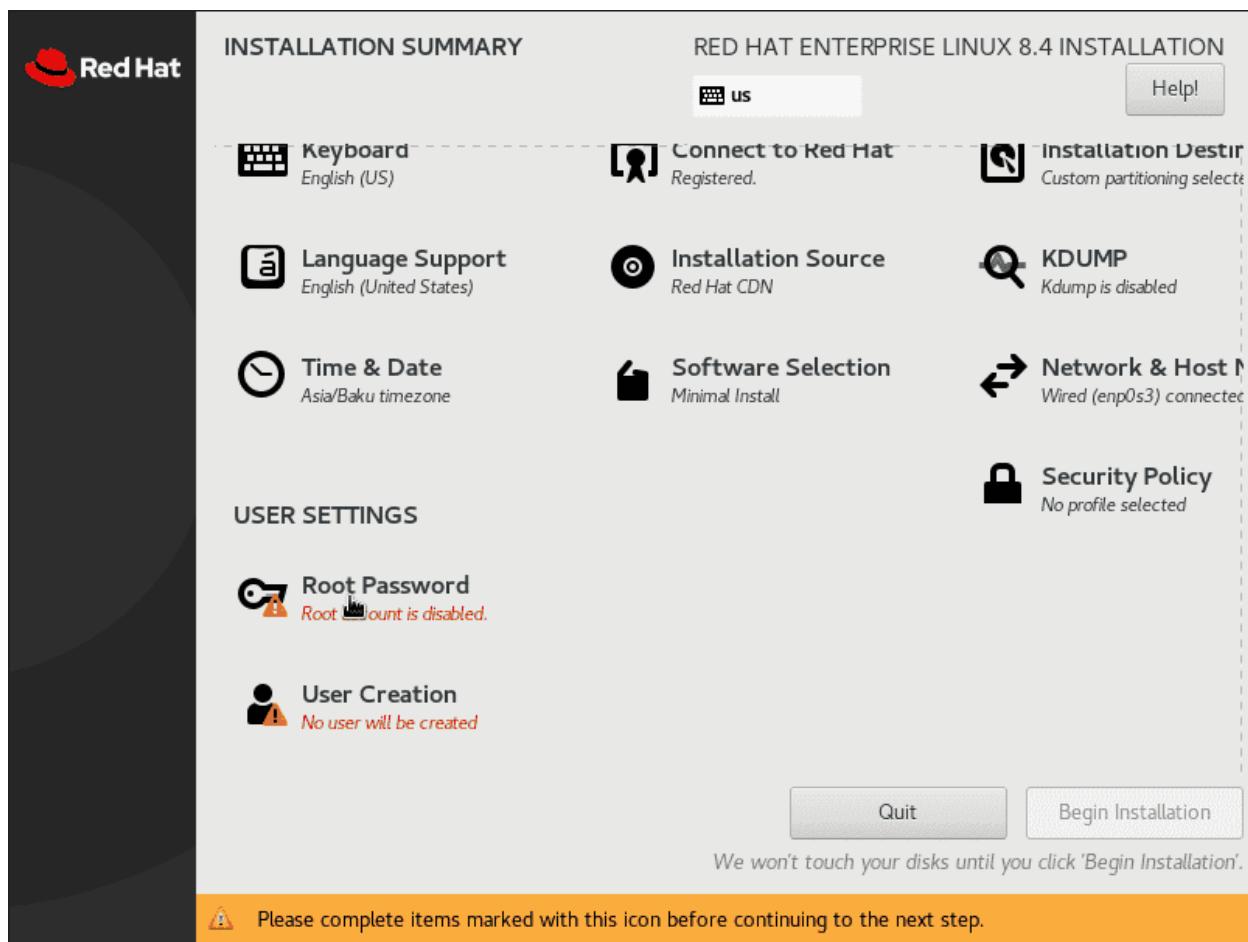
Затем заходим в Connect to Red Hat. Здесь нам нужно ввести логин и пароль, указанные при регистрации в портале Red Hat. Также убираем галочку с «Insights». После чего нажимаем Register.



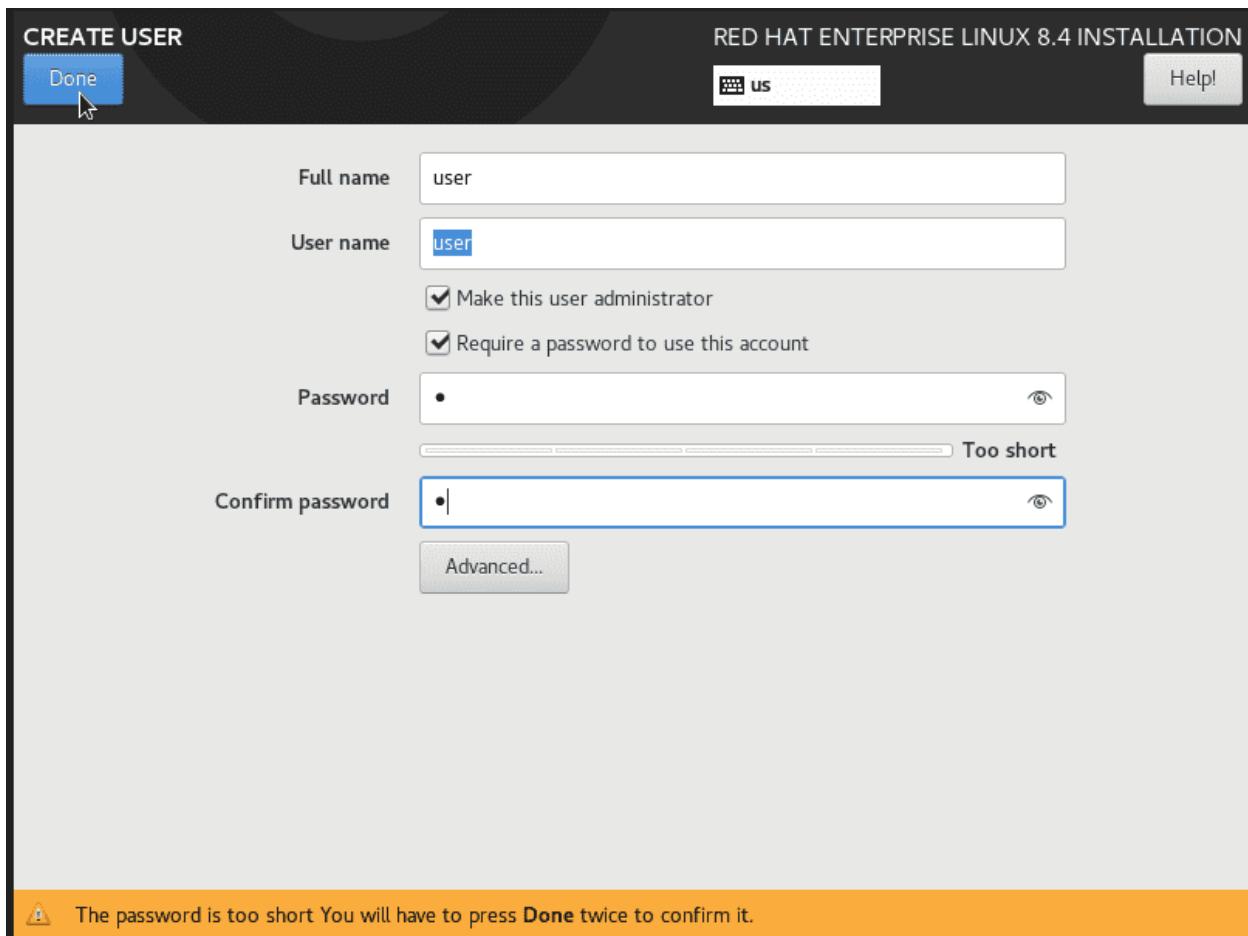
Если всё прошло нормально, вы увидите окно с информацией о подписке. Нажимаем Done.



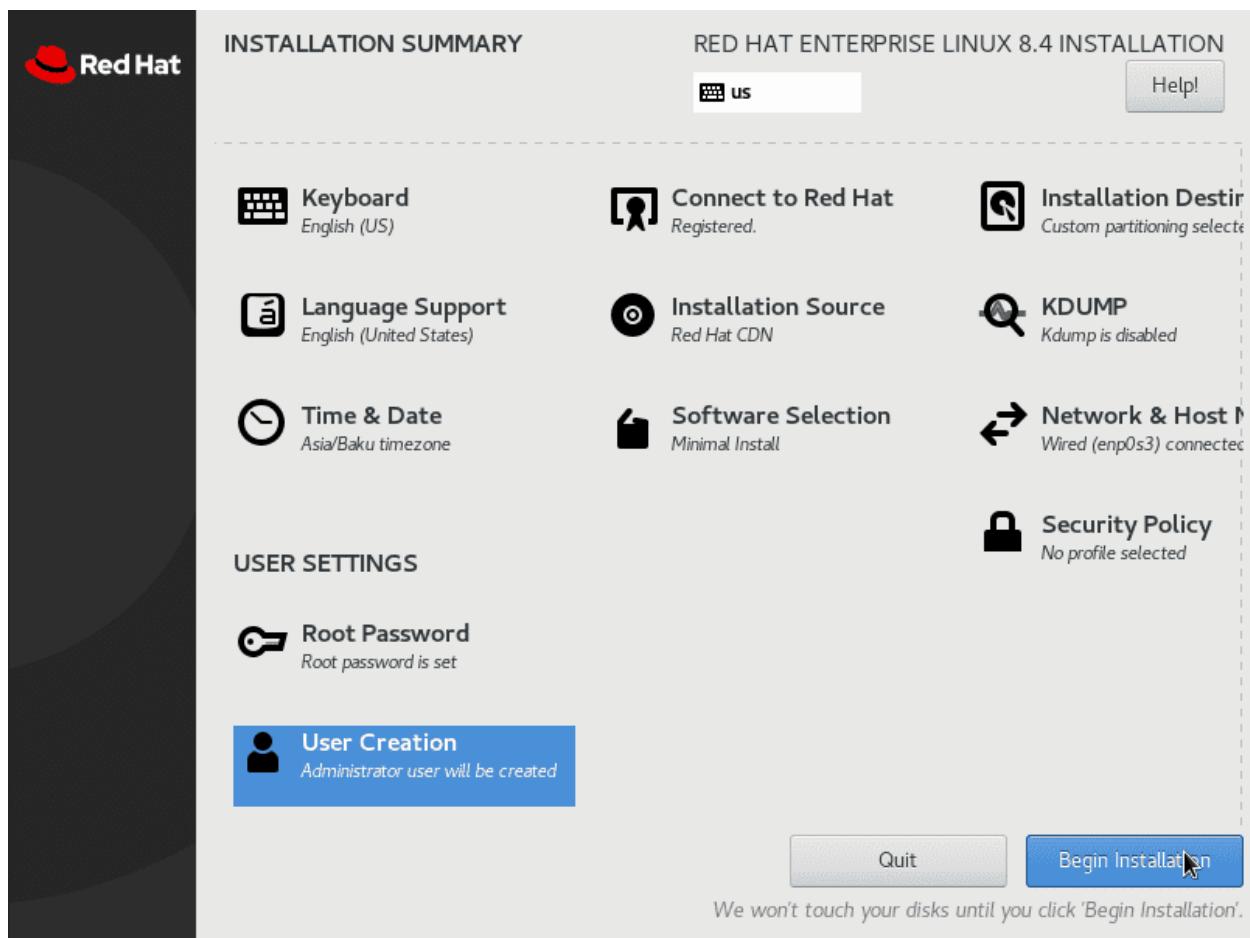
Installation source я пропускаю. Обычно здесь можно указать дополнительные репозитории, или выбрать локальные - если интернет не доступен. Но ни то, ни другое нам пока нужно. После этого идём в Software Selection. Здесь у нас есть готовые шаблоны, которые позволяют ещё при установке поставить все необходимые инструменты под нужную задачу. У нас задача не предопределена, поэтому мы поставим минимальную систему со стандартными утилитами. В стандартные утилиты входит тот же самый nano, всякие утилиты для управления SELinux и некоторые другие пакеты, без которых можно, но с ними удобнее. При этом графического интерфейса у нас не будет. Нажимаем Done.



Используя ползунок, спускаемся вниз. Осталось настроить пароль рута и создать пользователя.



При создании пользователя не забываем поставить галочку, чтобы сделать его администратором.



Ну и в конце концов нажимаем Begin Installation. Установка займёт некоторое время, потому что часть пакетов будет скачиваться с интернета. После установки нажимаете Reboot.

После установки

```

Red Hat Enterprise Linux 8.4 (Ootpa)
Kernel 4.18.0-305.12.1.el8_4.x86_64 on an x86_64

Activate the web console with: systemctl enable --now cockpit.socket

rhel8 login: root
Password:
[root@rhel8 ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:77:f4:80 brd ff:ff:ff:ff:ff:ff
        inet 192.168.31.205/24 brd 192.168.31.255 scope global dynamic nopref ixroute enp0s3
            valid_lft 43157sec preferred_lft 43157sec
        inet6 fe80::a00:27ff:fe77:f480/64 scope link nopref ixroute
            valid_lft forever preferred_lft forever
[root@rhel8 ~]# 

```

После перезагрузки логинимся в систему и узнаём ip адрес, чтобы подключиться по ssh.

```
ip a
```

```
[doctor@tardis] ~ [~]
└─ $ ssh root@192.168.31.205
The authenticity of host '192.168.31.205 (192.168.31.205)' can't be established.
ED25519 key fingerprint is SHA256:/wa/AWmSNZMlm0tvc/xN0sBeQ6gL0d23KY/Ak7oE6u0.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.31.205' (ED25519) to the list of known hosts.
root@192.168.31.205's password:
Activate the web console with: systemctl enable --now cockpit.socket

This system is not registered to Red Hat Insights. See https://cloud.redhat.com/
To register this system, run: insights-client --register

Last login: Mon Sep  6 19:20:10 2021
[root@rhel8 ~]# dnf update -y
Updating Subscription Management repositories.
Red Hat Enterprise Linux 8 for x86_64 - AppStream (RPMs)                                7.0 MB/s | 32 MB   00:04
Red Hat Enterprise Linux 8 for x86_64 - BaseOS (RPMs)                                 7.4 MB/s | 35 MB   00:04
Last metadata expiration check: 0:00:05 ago on Mon 06 Sep 2021 07:22:07 PM +04.
Dependencies resolved.
Nothing to do.
Complete!
[root@rhel8 ~]#
```

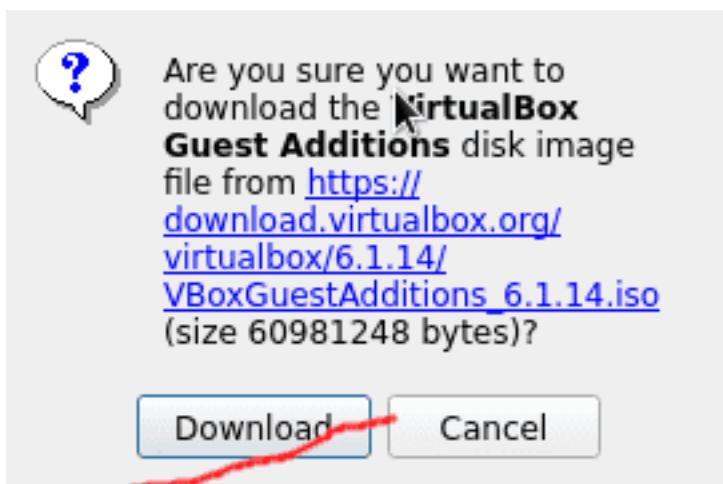
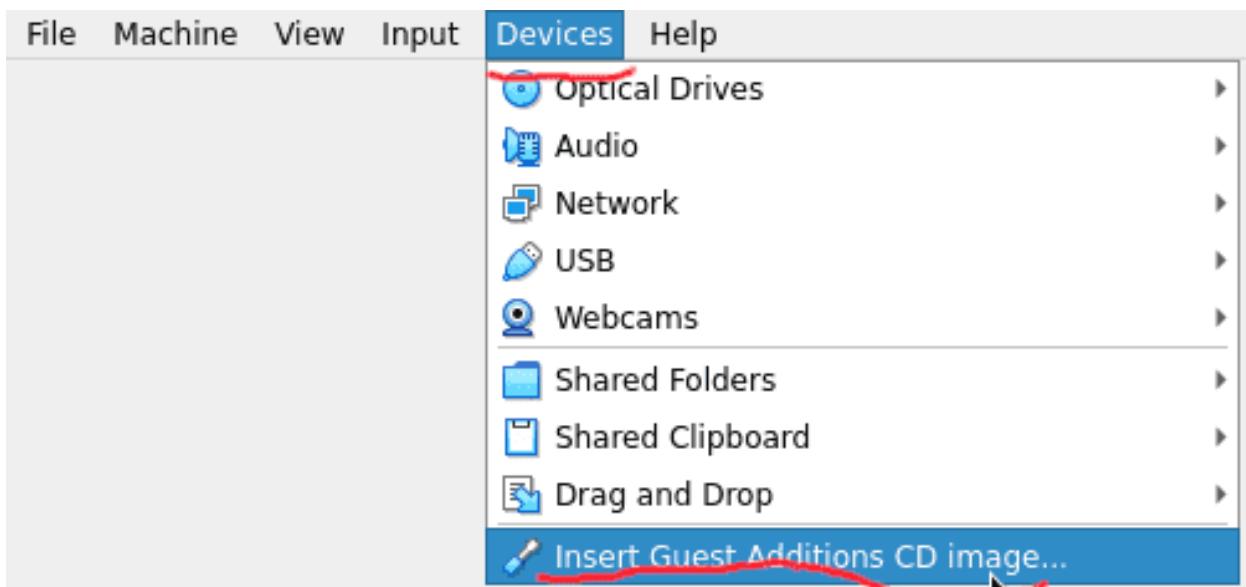
Собственно, подключаемся по ssh и на всякий случай проверяем обновления. Хотя обычно после установки по сети сразу обновления не требуются.

```
dnf update -y
```

```
[root@rhel8 ~]# sudo dnf install kernel-devel kernel-headers elfutils-libelf-devel gcc make perl -y
Updating Subscription Management repositories.
Last metadata expiration check: 0:01:38 ago on Mon 06 Sep 2021 07:22:07 PM +04.
Dependencies resolved.
=====
 Package           Arch    Version            Repository      Size
 =====
Installing:
 elfutils-libelf-devel   x86_64  0.182-3.el8          rhel-8-for-x86_64-baseos-rpms  59 k
 gcc                x86_64  8.4.1-1.el8          rhel-8-for-x86_64-appstream-rpms 23 M
 kernel-devel        x86_64  4.18.0-305.12.1.el8_4    rhel-8-for-x86_64-baseos-rpms   18 M
 kernel-headers       x86_64  4.18.0-305.12.1.el8_4    rhel-8-for-x86_64-baseos-rpms   7.1 M
 make               x86_64  1:4.2.1-10.el8         rhel-8-for-x86_64-baseos-rpms  498 k
 perl               x86_64  4:5.26.3-419.el8_4     rhel-8-for-x86_64-appstream-rpms 73 k
Installing dependencies:
```

Так как мы работаем с виртуалкой, первым делом стоит установить гостевые дополнения. А для этого предварительно нужно установить пару пакетов:

```
sudo dnf install kernel-devel kernel-headers elfutils-libelf-devel gcc make perl -y
```



После чего в виртуалбоксе к виртуалке нужно подцепить гостевые дополнения, ну и по необходимости их скачать.

```
[root@rhel8 ~]# lsscsi
[1:0:0:0] cd/dvd VBOX CD-ROM 1.0 /dev/sr0
[2:0:0:0] disk ATA VBOX HARDDISK 1.0 /dev/sda
[root@rhel8 ~]# mount /dev/sr0 /mnt/
mount: /mnt: WARNING: device write-protected, mounted read-only.
[root@rhel8 ~]# cd /mnt/
[root@rhel8 mnt]# ls
AUTORUN.INF OS2 VBoxDarwinAdditionsUninstall.tool VBoxWindowsAdditions.exe
autorun.sh runasroot.sh VBoxLinuxAdditions.run VBoxWindowsAdditions-x86.exe
cert TRANS.TBL VBoxSolarisAdditions.pkg
NT3x VBoxDarwinAdditions.pkg VBoxWindowsAdditions-amd64.exe
[root@rhel8 mnt]# ./VBoxLinuxAdditions.run
Verifying archive integrity... All good.
Uncompressing VirtualBox 6.1.26 Guest Additions for Linux.....
VirtualBox Guest Additions installer
Copying additional installer modules ...
Installing additional modules ...
VirtualBox Guest Additions: Starting.
VirtualBox Guest Additions: Building the VirtualBox Guest Additions kernel
modules. This may take a while.
VirtualBox Guest Additions: To build modules for other installed kernels, run
VirtualBox Guest Additions: /sbin/rcvboxadd quicksetup <version>
VirtualBox Guest Additions: or
VirtualBox Guest Additions: /sbin/rcvboxadd quicksetup all
VirtualBox Guest Additions: Building the modules for kernel
4.18.0-305.12.1.el8_4.x86_64.
[root@rhel8 mnt]# reboot
```

После того, как вы нажмёте Insert, вернитесь к ssh и запустите команду lsscsi. Она покажет файл CD-ROM-а, который надо будет временно примонтировать, допустим, в /mnt. Затем запустить из этого диска VBoxLinuxAdditions.run. А после установки - перезагрузить виртуалку.

```
lsscsi
mount /dev/sr0 /mnt
/mnt/VBoxLinuxAdditions.run
reboot
```

```
[x]-[doctor@tardis]—[~]
$ ssh root@192.168.31.205
root@192.168.31.205's password:
Activate the web console with: systemctl enable --now cockpit.socket

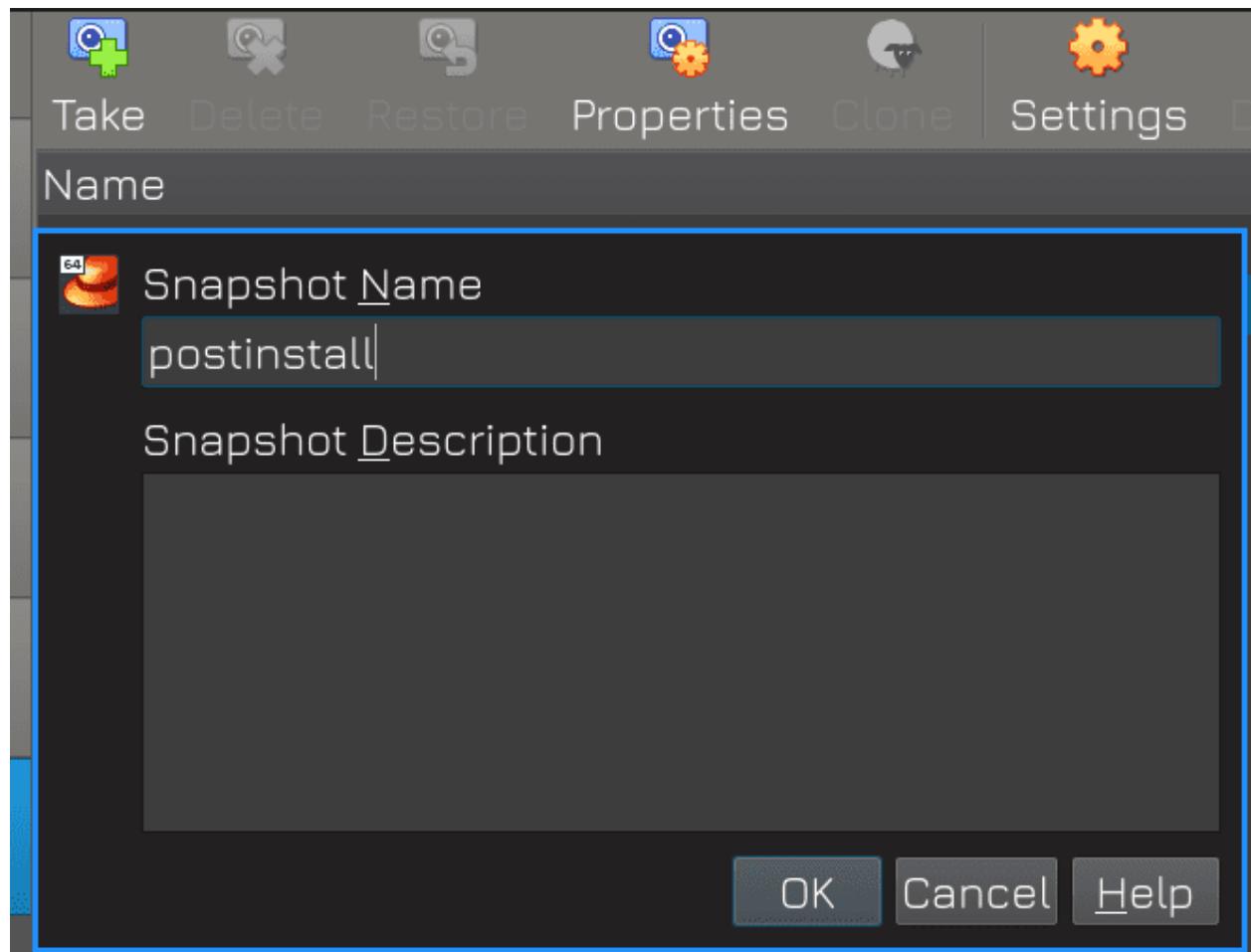
This system is not registered to Red Hat Insights. See https://cloud.redhat.com/
To register this system, run: insights-client --register

Last login: Mon Sep  6 19:25:16 2021 from 192.168.31.227
[root@rhel8 ~]# systemctl status vboxadd
● vboxadd.service
   Loaded: loaded (/opt/VBoxGuestAdditions-6.1.26/init/vboxadd; enabled; vendor preset: disabled)
     Active: active (exited) since Mon 2021-09-06 19:28:49 +04; 22s ago
       Process: 833 ExecStart=/opt/VBoxGuestAdditions-6.1.26/init/vboxadd start (code=exited, status=0/SUCCESS)
    Main PID: 833 (code=exited, status=0/SUCCESS)
      Tasks: 0 (limit: 5967)
        Memory: 0B
       CGroup: /system.slice/vboxadd.service

Sep 06 19:28:44 rhel8 vboxadd[833]: VirtualBox Guest Additions: Building the VirtualBox Guest Additions kernel
Sep 06 19:28:44 rhel8 vboxadd[833]: modules. This may take a while.
Sep 06 19:28:44 rhel8 vboxadd[833]: VirtualBox Guest Additions: To build modules for other installed kernels, run
Sep 06 19:28:44 rhel8 vboxadd[833]: VirtualBox Guest Additions: /sbin/rcvboxadd quicksetup <version>
Sep 06 19:28:44 rhel8 vboxadd[833]: VirtualBox Guest Additions: or
Sep 06 19:28:44 rhel8 vboxadd[833]: VirtualBox Guest Additions: /sbin/rcvboxadd quicksetup all
Sep 06 19:28:49 rhel8 vboxadd[833]: ValueError: File context for /opt/VBoxGuestAdditions-6.1.26/other/mount.vboxsf alr
Sep 06 19:28:49 rhel8 vboxadd[833]: VirtualBox Guest Additions: Running kernel modules will not be replaced until
Sep 06 19:28:49 rhel8 vboxadd[833]: the system is restarted
Sep 06 19:28:49 rhel8 systemd[1]: Started vboxadd.service.
[root@rhel8 ~]# shutdown
```

После перезагрузки ещё раз подключимся по ssh и убедимся, что гостевые дополнения установились. Сервис активен, а это говорит о том, что, скорее всего, всё установилось без проблем. И так, наша виртуалка готова. Давайте напоследок сделаем снимок, чтобы, если что, можно было вернуться к текущему состоянию. Для этого предварительно стоит выключить виртуалку.

```
systemctl status vboxadd
shutdown
```



Затем открываем VirtualBox, переходим в секцию snapshots и нажимаем Take. Назовём снапшот «postinstall» и нажмём OK. Всё готово.

Давайте подведём итоги. Установка операционной системы - простой процесс, с которым многие могут справиться. Однако зачастую у новичков возникают вопросы и сомнения - к примеру, как правильно разметить диск. Это, конечно, очень индивидуально и разнится от системы к системе. Но если понимать, что и зачем делать, то в будущем можно избежать многих проблем. И я надеюсь, что за эти 50 тем вы стали лучше понимать, как устроена операционная система и как с ней работать. А установка CentOS практически идентична установке RHEL, за исключением момента с регистрацией и подпиской.

2.54 54. Настройка времени

2.54.1 54. Настройка времени

Время в операционных системах играет важную роль. Если оно настроено неправильно, то можно столкнуться с различными проблемами:

- невозможно будет зайти на безопасные сайты. На многих сайтах есть сертификаты, благодаря которым наш браузер и система могут доверять этим сайтам. Но у сертификатов есть срок действия. И если время в нашей системе сильно отстает или спешит, то браузер будет считать

- сертификат недействительным и не пускать нас на сайт;
- также сложнее будет разобраться в логах и понять, какие события связаны друг с другом, особенно когда у вас множество систем. Где-то часы спешат на 5 минут, где-то отстают на час, и сложнее становиться коррелировать события.
 - многие сервисы, настроенные на отказоустойчивость, из-за различий во времени могут привести к серьёзным неполадкам.

И это лишь часть проблем. Поэтому администратору важно уметь настраивать время.

Часы реального времени (RTC)

В каждом компьютере в материнской плате есть часы, называемые часами реального времени - real time clock - RTC. Засчёт плоской батарейки они работают всегда, даже когда компьютер выключен. Но они сбиваются, если эта батарейка разрядится или если её вытащить. И эти часы обычно настраиваются через BIOS, но и операционная система может их изменить.

```
[user@rhel8 ~]$ sudo hwclock
2021-09-10 10:15:14.179894+04:00
[user@rhel8 ~]$ █
```

Операционная система при включении узнаёт время по этим часам. Чтобы посмотреть текущее время на них, можно использовать утилиту hwclock. Цифры после точки - это микросекунды, а плюс указывает, что ко времени добавлено 4 часа - учтён часовой пояс.

```
[user@rhel8 ~]$ sudo hwclock --verbose
hwclock from util-linux 2.32.1
System Time: 1631255838.159846
Trying to open: /dev/rtc0
Using the rtc interface to the clock.
Last drift adjustment done at 0 seconds after 1969
Last calibration done at 0 seconds after 1969
Hardware clock is on UTC time
Assuming hardware clock is kept in UTC time.
Waiting for clock tick...
...got clock tick
Time read from Hardware Clock: 2021/09/10 06:36:22
Hw clock time : 2021/09/10 06:36:22 = 1631255782 seconds since 1969
Time since last adjustment is 1631255782 seconds
Calculated Hardware Clock drift is 0.000000 seconds
2021-09-10 10:36:21.843483+04:00
[user@rhel8 ~]$ █
```

Время на таких часах считается в секундах, где нулём является полночь первого января 1970 года, так называемая «Эпоха UNIX», а количество прошедших секунд называется UNIX-временем. По-

умолчанию, Linux предполагает, что на часах стоит время по UTC и добавляет к ним разницу в часовом поясе, а Windows предполагает локальное время. И если вы поставите на один компьютер обе системы, Windows-e и Linux будут перебивать часы друг друга. Но это легко исправимо, достаточно на линуксе указать, что на этих часах локальное время.

Системные часы

```
[user@rhel8 ~]$ date
Fri Sep 10 11:37:10 +04 2021
[user@rhel8 ~]$ sudo hwclock ; date
2021-09-10 11:36:18.612350+04:00
Fri Sep 10 11:37:15 +04 2021
[user@rhel8 ~]$ █
```

Во время запуска компьютера ядро операционной системы считывает RTC, берёт время и запускает свои часы, называемые системными. Оно работает в оперативке, поэтому при каждом выключении пропадает. Узнать время на системных часах можно с помощью утилиты date. Это независимые часы, поэтому спустя какое-то время системные часы могут расходиться с часами реального времени:

```
sudo hwclock; date
```

```
DATE STRING
The --date=STRING is a mostly free format human readable date string such as "Sun, 29 Feb 2004
16:21:42 -0800" or "2004-02-29 16:21:42" or even "next Thursday". A date string may contain
items indicating calendar date, time of day, time zone, day of week, relative time, relative
date, and numbers. An empty string indicates the beginning of the day. The date string for-
mat is more complex than is easily documented here but is fully described in the info documen-
tation.
```

Системное время можно поменять с помощью этой же утилиты. Для этого нужно использовать ключ -s - set. Время можно задавать по разному, найдите в man-е по date строчку «DATE STRING», здесь есть примеры.

```
[user@rhel8 ~]$ sudo date -s '2004-02-29 16:21:42'
[sudo] password for user:
Sun Feb 29 16:21:42 +04 2004
[user@rhel8 ~]$ date
Sun Feb 29 16:21:43 +04 2004
[user@rhel8 ~]$ sudo hwclock
2021-09-10 11:58:59.334659+04:00
[user@rhel8 ~]$ █
```

Скопируем пример и запустим команду:

```
sudo date -s '2004-02-29 16:21:42'  
date  
sudo hwclock
```

Как видите, системное время поменялось, теперь на часах 2004 год. Но это никак не повлияло на RTC.

```
[user@rhel8 ~]$ hwclock --help  
  
Usage:  
  hwclock [function] [option...]  
  
Time clocks utility.  
  
Functions:  
  -r, --show          display the RTC time  
  --get              display drift corrected RTC time  
  --set              set the RTC according to --date  
  -s, --hctosys      set the system time from the RTC  
  -w, --systohc      set the RTC from the system time  
  --systz            send timescale configurations to the kernel  
  -a, --adjust        adjust the RTC to account for systematic drift  
  --predict          predict the drifted RTC time according to --date
```

Можно записать время с системных часов на часы реального времени и наоборот. Это можно сделать через утилиту hwclock с помощью выделенных ключей.

```
[user@rhel8 ~]$ sudo hwclock -s  
[user@rhel8 ~]$ sudo hwclock; date  
2021-09-10 12:02:00.117511+04:00  
Fri Sep 10 12:02:01 +04 2021  
[user@rhel8 ~]$ █
```

Давайте синхронизируем часы, что исправить время.

```
sudo hwclock -s  
sudo hwclock; date
```

И теперь системные часы показывают 2021 год.

```
[user@rhel8 ~]$ timedatectl
      Local time: Fri 2021-09-10 11:42:20 +04
      Universal time: Fri 2021-09-10 07:42:20 UTC
            RTC time: Fri 2021-09-10 07:41:24
            Time zone: Asia/Baku (+04, +0400)
System clock synchronized: yes
          NTP service: active
        RTC in local TZ: no
[user@rhel8 ~]$ timedatectl
list-timezones  set-ntp          set-timezone   show-timesync   timesync-status
set-local-rtc    set-time         show           status
[user@rhel8 ~]$ timedatectl set-local-rtc
false  true
[user@rhel8 ~]$ timedatectl set-local-rtc false
[user@rhel8 ~]$ █
```

Есть утилита, которая объединяет настройку часов реального времени, системных часов и часового пояса - timedatectl. Как тут видно, локальное время и время на RTC отличаются. Это как раз о разнице с Windows. И чтобы Linux не добавлял к RTC таймзону нужно использовать опцию set-local-rtc со значением true:

```
sudo timedatectl set-local-rtc true
```

Но я этого делать не буду, у меня со временем проблем нет.

```
[user@rhel8 ~]$ timedatectl list-timezones
Africa/Abidjan
Africa/Accra
Africa/Addis_Ababa
Africa/Algiers
Africa/Asmara
```

Давайте, для примера, настроим часовой пояс. Для начала найдём таймзону с помощью опции list-timezones и используя поиск с помощью слэша - /Moscow

```
[user@rhel8 ~]$ sudo timedatectl set-timezone Europe/Moscow
[user@rhel8 ~]$ timedatectl
      Local time: Fri 2021-09-10 11:24:07 MSK
      Universal time: Fri 2021-09-10 08:24:07 UTC
            RTC time: Fri 2021-09-10 08:23:11
            Time zone: Europe/Moscow (MSK, +0300)
System clock synchronized: yes
          NTP service: active
        RTC in local TZ: no
[user@rhel8 ~]$ █
```

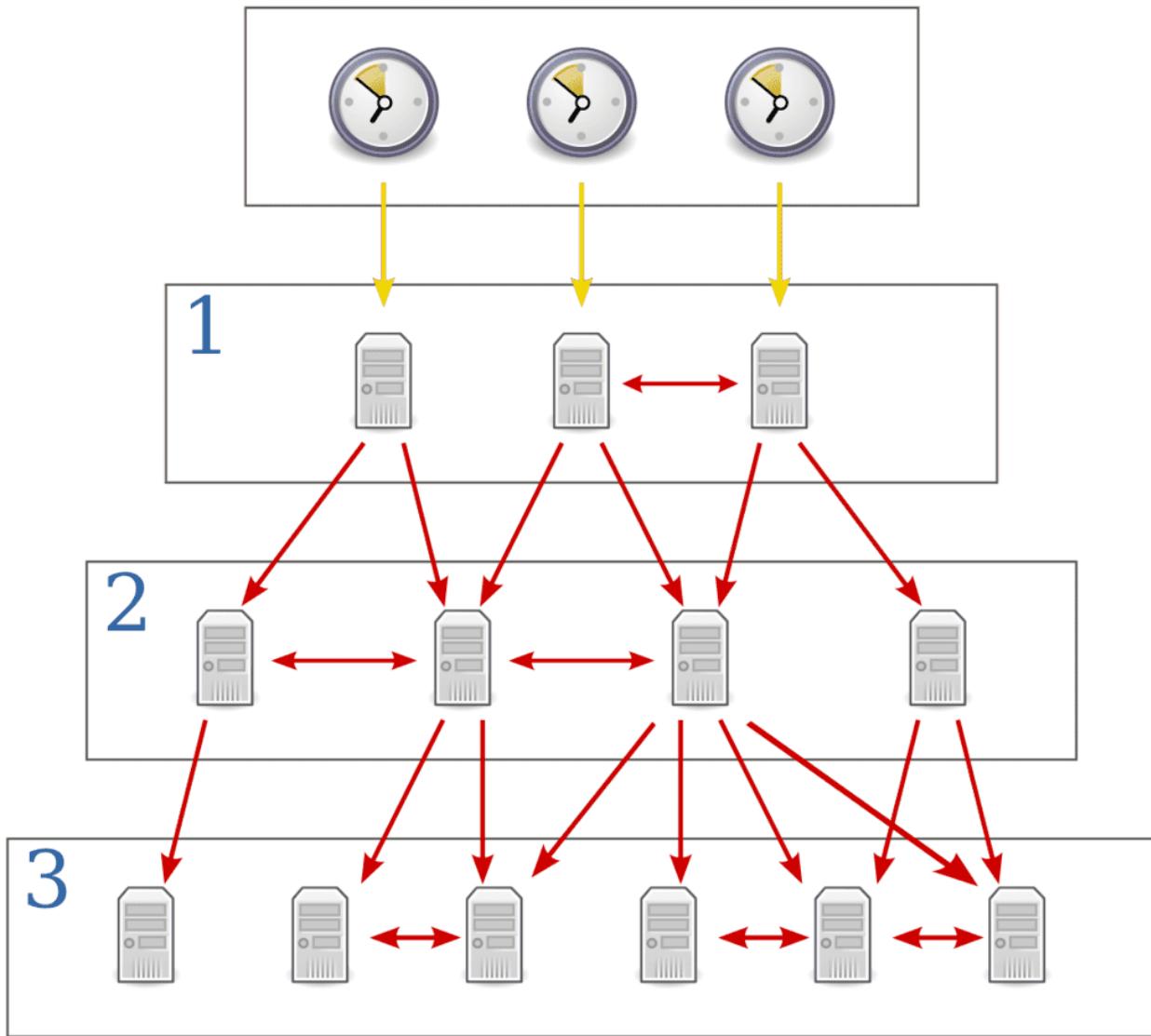
И с помощью опции set-timezone зададим найденное значение:

```
sudo timedatectl set-timezone Europe/Moscow  
timedatectl
```

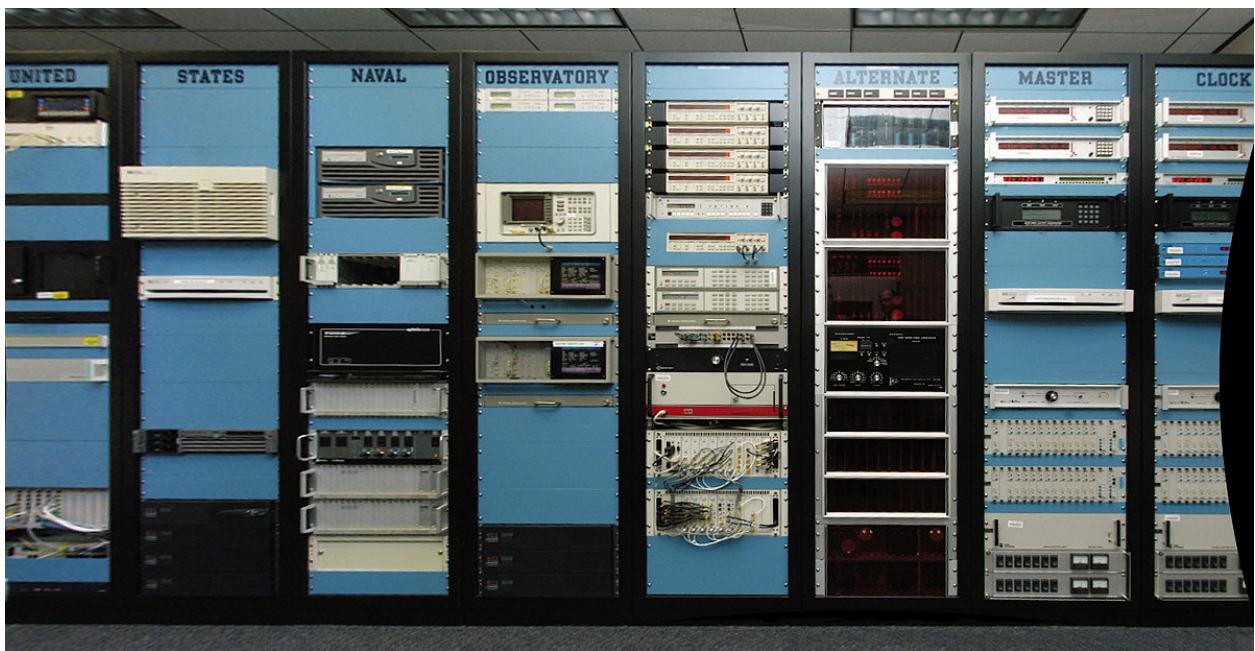
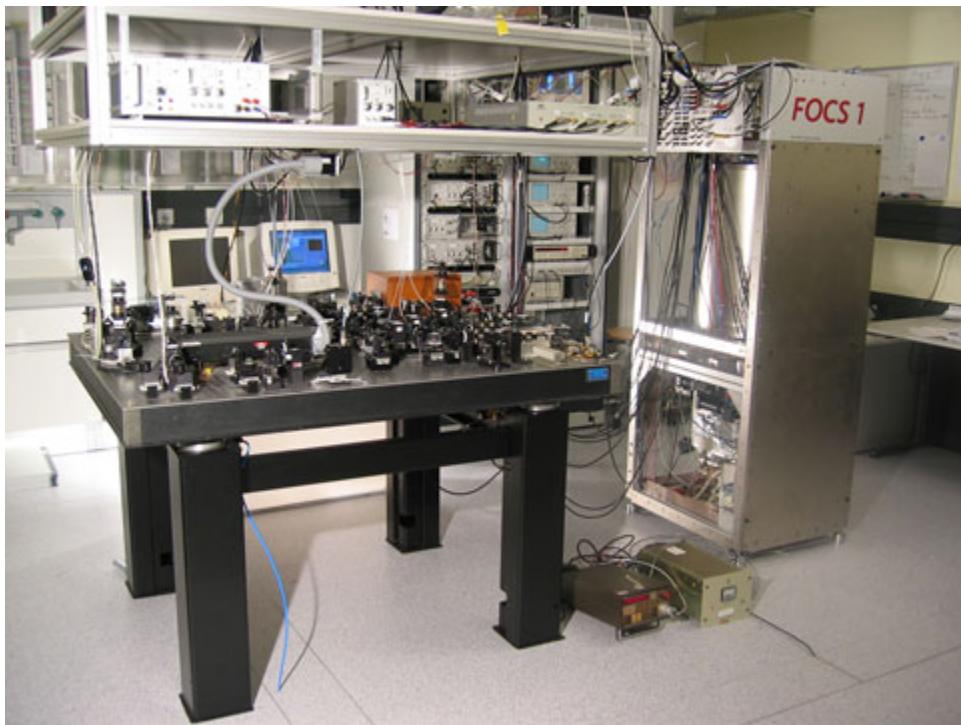
Как видите, теперь часовой пояс - Москва.

NTP

Когда у вас один компьютер, настроить время не проблема. Если время будет спешить или отставать, поправить его тоже не проблема. Но постоянно следить и исправлять часы неудобно. А если дело касается администрирования множества серверов, чувствительных ко времени, то за всеми уследить трудно, да и всё это лишняя траты сил и времени.

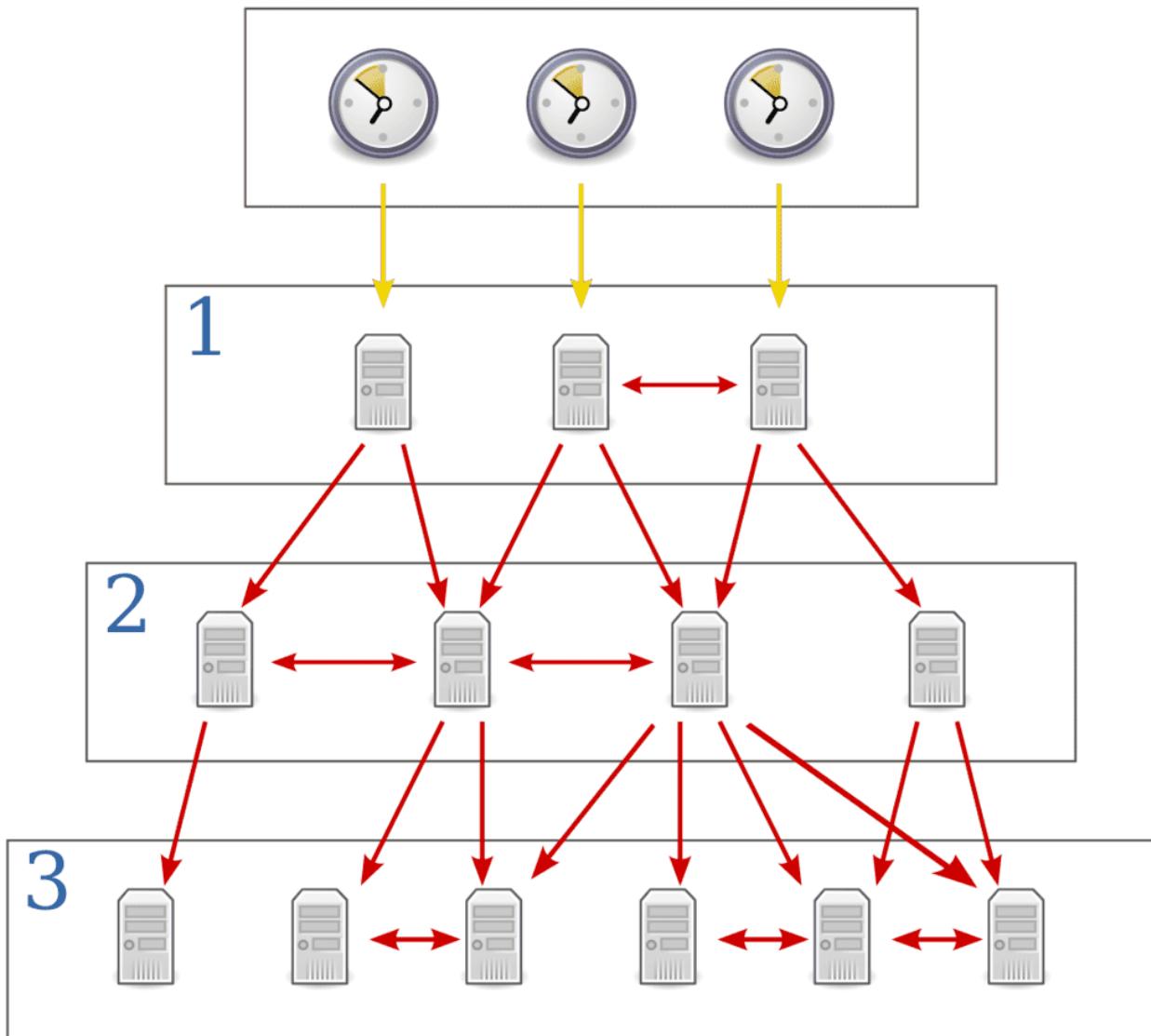


Чтобы на всех ваших системах было одно и тоже время и чтобы оно постоянно было правильным, используется протокол NTP - network time protocol. Грубо говоря, есть сервера, которые знают правильное время и ваши системы могут периодически обращаться к этим серверам, узнавать время и поправлять у себя. Сами сервера тоже в свою очередь берут время у других серверов.



И есть определённая иерархия. Точное время с минимальной погрешностью определяется на специальных часах, называемых атомными. Но таких часов не так много, чтобы каждый мог к ним подключиться. Поэтому к ним подключаются первичные сервера времени.

Но и первичных серверов времени довольно мало, чтобы каждый желающий мог к ним подключаться. К ним могут подключаться только те, кто раздаёт время на большое количество устройств, обычно это вторичные сервера времени.



Т.е. всё по иерархии, как на картинке. У каждого сервера есть так называемый стратум, это то, на каком слое он находится, т.е. stratum 1 - это первичные сервера, stratum 2 - вторичные и т.д. Всего значений может быть 15. Чем ближе сервер к атомным часам, тем меньше стратум и тем точнее часы. Но речь идёт о микросекундах, поэтому, в большинстве случаев, это не критично.

Обычно, в локальной сети вы поднимаете несколько своих ntp серверов и указываете их на других машинках. Ваши NTP сервера будут брать время из публичных серверов, а всякие компьютеры и сервера в сети - с ваших локальных NTP серверов. Если вдруг интернет пропадёт, или вы намеренно запретите всем серверам выход в интернет, всё равно будет работать синхронизация с локальным NTP сервером и время не разбежится.

Есть различные программы, которые могут забирать и раздавать время, т.е. выступать NTP клиентом и сервером. Одни из самых популярных - ntpd и chrony. Есть определённые различия в функционале, но они не так существенны в большинстве случаев. Но если вам интересно, можете почитать по ссылке.

```
[user@rhel8 ~]$ systemctl status chronyd
● chronyd.service - NTP client/server
  Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled; vendor preset: enabled)
  Active: active (running) since Fri 2021-09-10 09:01:51 MSK; 4h 29min ago
    Docs: man:chronyd(8)
          man:chrony.conf(5)
  Process: 856 ExecStartPost=/usr/libexec/chrony-helper update-daemon (code=exited, status=0/SUCCESS)
  Process: 841 ExecStart=/usr/sbin/chronyd $OPTIONS (code=exited, status=0/SUCCESS)
 Main PID: 850 (chronyd)
   Tasks: 1 (limit: 5967)
  Memory: 1.5M
    CGroup: /system.slice/chronyd.service
            └─850 /usr/sbin/chronyd
```

При установке системы мы поставили галочку «Network Time», благодаря чему у нас установился ntp клиент. По умолчанию, это chrony. Чтобы постоянно синхронизировать время, он работает как демон:

```
systemctl status chronyd
```

Если будут расхождения во времени, chrony поправит это. Но не то чтобы он заменит время на правильное - так делать нельзя, так как это может привести к большим проблемам. Вместо этого chrony будет ускорять или замедлять часы на доли секунд, чтобы исправить время. Соответственно, если время отличается сильно, то и смысла синхронизировать зачастую не будет, так как такой процесс может занять годы. Однако, если ничего важного и чувствительного ко времени не работает на сервере, то можно пренебречь ускорением и сразу выставить нужное время.

```
[user@rhel8 ~]$ chronyc sources -v
210 Number of sources = 8

    --- Source mode '^' = server, '=' = peer, '#' = local clock.
    / .- Source state '*' = current synced, '+' = combined , '-' = not combined,
    /   '?' = unreachable, 'x' = time may be in error, '~' = time too variable.
    |           .- xxxx [ yyyy ] +/- zzzz
    |           |   xxxx = adjusted offset,
    |           |   yyyy = measured offset,
    |           |   zzzz = estimated error.
    |           |
    |           Reachability register (octal) - .
    |           Log2(Polling interval) -- .
    |           \   |
    |           \   |
    |           \   |
MS Name/IP address      Stratum Poll Reach LastRx Last sample
=====
^- 208.91.112.63          2 10  377   354    -19ms[ -19ms] +/-  157ms
^- 208.91.112.61          2 10  377  1005    -13ms[ -13ms] +/- 148ms
^- 208.91.112.62          2 10  377   491    -17ms[ -17ms] +/- 152ms
^- 208.91.112.60          2 10  377  1118    -11ms[ -11ms] +/- 159ms
^+ time.cloudflare.com     3 10  377   539   -1644us[-1644us] +/-   37ms
^* time.cloudflare.com     3 10  377   695    +125us[ +88us] +/-   36ms
^? time.cloudflare.com     0  6    0     -      +0ns[ +0ns] +/-    0ns
^? time.cloudflare.com     0  6    0     -      +0ns[ +0ns] +/-    0ns
[user@rhel8 ~]$
```

Также есть утилита chronyc, которая позволяет управлять и смотреть всякую информацию. Одна из главных опций - sources, она позволяет узнать, к каким серверам мы обращаемся и всё ли нормально. А ключ -v - verbose - даёт подсказки, что означают те или иные символы:

```
chronyc sources
```

Если проанализировать вывод: все строчки в таблице начинаются с символа карат - наверху видно, что это обозначение для серверов. Как видите, серверов много. Обычно рекомендуется указывать либо 4 и больше серверов, либо 1. Дело в том, что NTP для доставки информации использует UDP - т.е. не гарантирует целостность данных. Также, говоря о точности времени, речь идёт о микросекундах,

а у разных серверов время может отличаться. Если указать два сервера - сложно будет понять, кто из них выдаёт правильное время. С одним сервером таких вопросов не будет, но, если такой сервер станет недоступен, то и узнать время не получится. Если же указать 4 или больше серверов, будут использоваться алгоритмы комбинирования для определения времени.

Второй символ в таблице показывает, какой сервер используется - он выделен звёздочкой. Значения с этого сервера могут комбинироваться со значениями от некоторых других серверов, с какими-то не могут. А вот вопрос означает, что такой сервер недоступен.

После адресов серверов мы также видим их стратумы. Обычно это 2 или 3. 0 означает, что мы не можем определить stratum, потому что сервер недоступен.

```
[user@rhel8 ~]$ chronyc tracking
Reference ID      : A29FC801 (time.cloudflare.com)
Stratum          : 4
Ref time (UTC)   : Fri Sep 10 11:30:38 2021
System time      : 0.000206281 seconds slow of NTP time
Last offset      : -0.000422976 seconds
RMS offset       : 1.091378450 seconds
Frequency        : 509.921 ppm slow
Residual freq    : -0.010 ppm
Skew              : 0.090 ppm
Root delay       : 0.070077322 seconds
Root dispersion  : 0.001631786 seconds
Update interval  : 1036.1 seconds
Leap status       : Normal
[user@rhel8 ~]$ █
```

Опция tracking детальнее раскроет информацию. Большая часть этих данных нужна для диагностики проблем. И это не так важно, если вы не работаете с сервисами, щепетильными ко времени.

Настройка chronyd

Зачастую вам нужно уметь поднять NTP сервер и настроить клиенты, чтобы они подключались к вашему серверу. Сделаем так - RHEL настроим в качестве NTP сервера, который будет брать время от публичных NTP серверов и раздавать на наш CentOS.

```
[user@rhel8 ~]$ sudo dnf install chrony
Updating Subscription Management repositories.
Last metadata expiration check: 4:47:42 ago on Fri 10 Sep 2021 10:03:57 AM MSK.
Package chrony-3.5-2.el8.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[user@rhel8 ~]$ sudo systemctl enable chronyd
[user@rhel8 ~]$ █
```

Начнём с NTP сервера. Им у нас будет chronyd, который уже предустановлен. Но, если у вас его нет, следует его установить и включить. Пакет называется chrony, а сервис - chronyd:

```
sudo dnf install chrony
sudo systemctl enable chronyd
```

<https://www.ntppool.org/en>



NTP Pool Project

[JOIN THE POOL](#) [USE THE POOL](#) [MANAGE SERVERS](#)

pool.ntp.org: public ntp time server for everyone

Introduction

The pool.ntp.org project is a big virtual cluster of timeservers providing reliable [easy to use](#) NTP service for millions of clients.

The pool is being used by hundreds of millions of systems around the world. It's the default "time server" for most of the major Linux distributions and many networked appliances (see [information for vendors](#)).

Because of the large number of users we are in need of more servers. If you have a server with a static IP address always available on the internet, please consider [adding it to the pool](#).

Active Servers	
	Africa 55
	Asia 334
	Europe 2940
	North America 983
	Oceania 138

Обычно, пакет уже приходит с настройками, где указаны публичные NTP сервера, но давайте мы их заменим. Есть проект ntp.org, в котором участвуют сервера по всему миру. Заходим на сайт [ntppool.org](https://www.ntppool.org) и выберем регион. Для России это Europe.

<https://www.ntppool.org/zone/europe>

[Norway — no.pool.ntp.org \(42\)](#)
[Poland — pl.pool.ntp.org \(69\)](#)
[Portugal — pt.pool.ntp.org \(15\)](#)
[Romania — ro.pool.ntp.org \(25\)](#)
[Republic of Serbia — rs.pool.ntp.org \(14\)](#)
[Russian Federation — ru.pool.ntp.org \(161\)](#)
[Sweden — se.pool.ntp.org \(80\)](#)

В списке находим РФ.

<https://www.ntppool.org/zone/ru>



[JOIN THE POOL](#) [USE THE POOL](#) [MANAGE SERVERS](#)

Russian Federation — ru.pool.ntp.org

To use this specific pool zone, add the following to your ntp.conf file:

```
server 0.ru.pool.ntp.org
server 1.ru.pool.ntp.org
server 2.ru.pool.ntp.org
server 3.ru.pool.ntp.org
```

In most cases it's best to use **pool.ntp.org** to find an NTP server (or 0.pool.ntp.org, 1.pool.ntp.org, etc if you need multiple server names). The system will try finding the closest available servers for you. If you distribute software or equipment that uses NTP, please see our [information for vendors](#).

Здесь указаны 4 адреса. Но на самом деле это не 4 сервера, а сотня серверов, которые находятся за этими адресами. Снизу также есть подсказка, что в большинстве случаев лучше указывать pool.ntp.org,

чтобы найти ближайшие адреса. Мы можем как скопировать эти 4 адреса и использовать их, так и последовать совету и использовать pool.ntp.org. Параметр server мы будем использовать на CentOS, поэтому давайте на RHEL используем pool.

```
GNU nano 2.9.8                               /etc/chrony.conf

# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).

#pool 2.rhel.pool.ntp.org iburst
pool pool.ntp.org iburst

#
# Record the rate at which the system clock gains/losses time.
driftfile /var/lib/chrony/drift
```

Зайдём в /etc/chrony.conf. Здесь уже есть строчка pool, закомментируем её и напишем свою. Кстати, в строчке pool также указан параметр iburst. Благодаря ему при запуске операционной системы или сервиса синхронизация времени происходит быстрее.

```
pool pool.ntp.org iburst

GNU nano 2.9.8                               /etc/chrony.conf

# the system clock.
#minsources 2

# Allow NTP client access from local network.
allow 192.168.10.0/24
allow 192.168.31.0/24

# Serve time even if not synchronized to a time source.
local stratum 10

# Specify file containing keys for NTP authentication.
keyfile /etc/chrony.keys
```

Спустимся чуть ниже. Чтобы превратить chrony в NTP сервер, т.е. чтобы он также раздавал время, надо раскомментировать строчку allow, в которой нужно указать сети, для которых мы будем раздавать адреса. Допустим, я хочу, чтобы он раздавал в сетях 192.168.10.0/24 и 192.168.31.0/24.

```
allow 192.168.10.0/24
allow 192.168.31.0/24
```

Ещё одно важное замечание. Если вдруг этот компьютер потеряет доступ в интернет и не сможет достучаться до указанных NTP серверов, он перестанет раздавать время. Чтобы он продолжал раздавать время даже когда нет интернета, допустим, в закрытой сети, надо расскомментировать строчку local stratum:

```
local stratum 10
```

```
[user@rhel8 ~]$ sudo nano /etc/chrony.conf
[sudo] password for user:
[user@rhel8 ~]$ sudo systemctl restart chronyd
[user@rhel8 ~]$ chronyc sources
210 Number of sources = 8
MS Name/IP address          Stratum Poll Reach LastRx Last sample
=====

^+ 208.91.112.63            2      6    17      2     -11ms[   -11ms] +/-  156ms
^+ 208.91.112.61            2      6    17      1     -15ms[   -15ms] +/-  154ms
^+ 208.91.112.62            2      6    17      1     -11ms[   -12ms] +/-  154ms
^* 208.91.112.60            2      6    17      1    -9478us[   -10ms] +/-  145ms
^- ntp3.junkemailfilter.com  2      6     7      3    +6931us[+2750us] +/-  123ms
^+ li1210-167.members.linode> 2      6    17      2    +5308us[+4597us] +/-  145ms
^+ hc-007-ntp1.weber.edu     1      6    17      1     +16ms[   +15ms] +/-  120ms
^+ linode.ibendit.com       2      6    17      1    +3464us[+2754us] +/-  158ms
[user@rhel8 ~]$
```

После проделанных изменений стоит перезапустить сервис chronyd, подождать пару секунд и проверить синхронизацию времени, с помощью того же chronyc:

```
sudo systemctl restart chronyd
chronyc sources
```

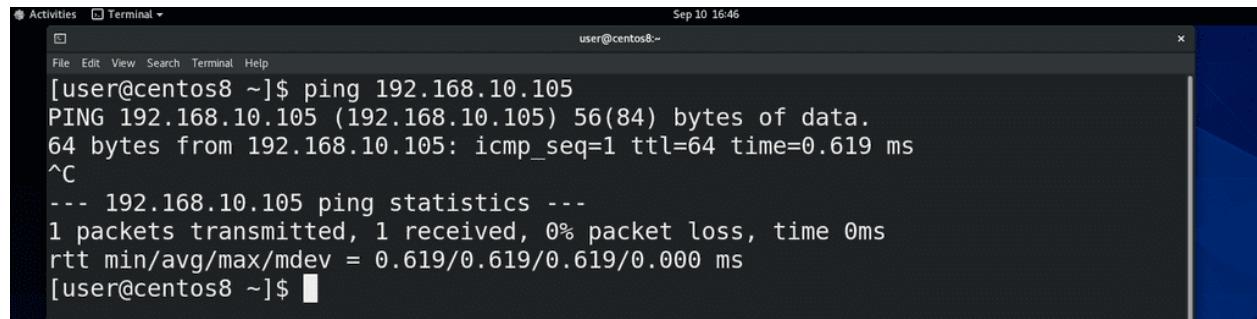
Я вижу, что перед одним из серверов стоит звёздочка, да и справа есть значения, а не нули. Значит, всё работает.

```
[user@rhel8 ~]$ sudo firewall-cmd --add-service=ntp --permanent
success
[user@rhel8 ~]$ sudo firewall-cmd --reload
success
[user@rhel8 ~]$
```

Также, чтобы CentOS мог подключиться к этому серверу и брать отсюда время, мне нужно разрешить NTP на файрволе. NTP работает на 123 порту по UDP, а в firewalld его можно добавить просто как сервис:

```
sudo firewall-cmd --add-service=ntp --permanent
sudo firewall-cmd --reload
```

NTP сервер мы настроили, осталось настроить клиент.



Для начала убедимся, что CentOS видит RHEL:

```
ping 192.168.10.105
```

```
GNU nano 2.9.8          /etc/chrony.conf

# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).
#pool 2.centos.pool.ntp.org iburst

server 192.168.10.105 iburst

# Record the rate at which the system clock gains/losses time.
driftfile /var/lib/chrony/drift
```

Заходим в /etc/chrony.conf, комментируем pool и прописываем server с адресом RHEL.

```
server 192.168.10.105 iburst
```

The screenshot shows a terminal window titled 'Activities Terminal'. The command [user@centos8 ~]\$ sudo nano /etc/chrony.conf is run, followed by [sudo] password for user:. Then, the command sudo systemctl restart chronyd is run. Finally, the command chronyc sources is run, displaying the following output:

MS Name/IP address	Stratum	Poll	Reach	LastRx	Last sample
^* 192.168.10.105	2	6	17	4	+14us [+4126us] +/- 118ms
^- 208.91.112.61	2	6	33	1	-11ms [-11ms] +/- 147ms
^- 208.91.112.63	2	6	17	3	-10ms [-10ms] +/- 147ms
^- 208.91.112.62	2	6	27	1	-8342us [-8342us] +/- 145ms
^- 208.91.112.60	2	6	7	0	-9059us [-9059us] +/- 153ms

Сохраняем изменения и перезапускаем сервис:

```
sudo systemctl restart chronyd
```

Ждём пару секунд и убеждаемся, что NTP сервер доступен и работает

```
chronyc sources
```

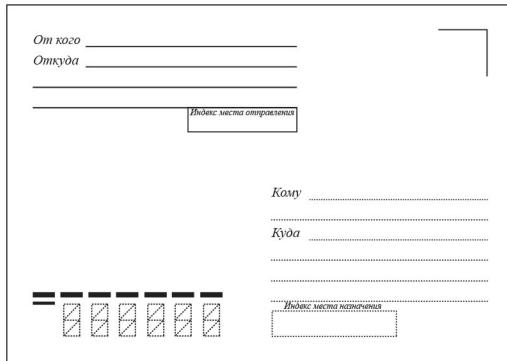
Как видите, в списке отобразился адрес нашего сервера и перед ним есть звёздочка - значит всё работает. Таким образом, мы настроили локальный NTP сервер и подключили к нему какую-то машинку, чтобы она синхронизировала время.

NTP сервер потребляет мало ресурсов, поэтому мы можем сделать 4 таких сервера, чтобы время в сети было точнее, либо оставить один, это уже зависит от количества систем в нашей сети. И хотя функционал NTP сервера довольно большой и мы рассмотрели его поверхностно, этого хватает для большинства задач.

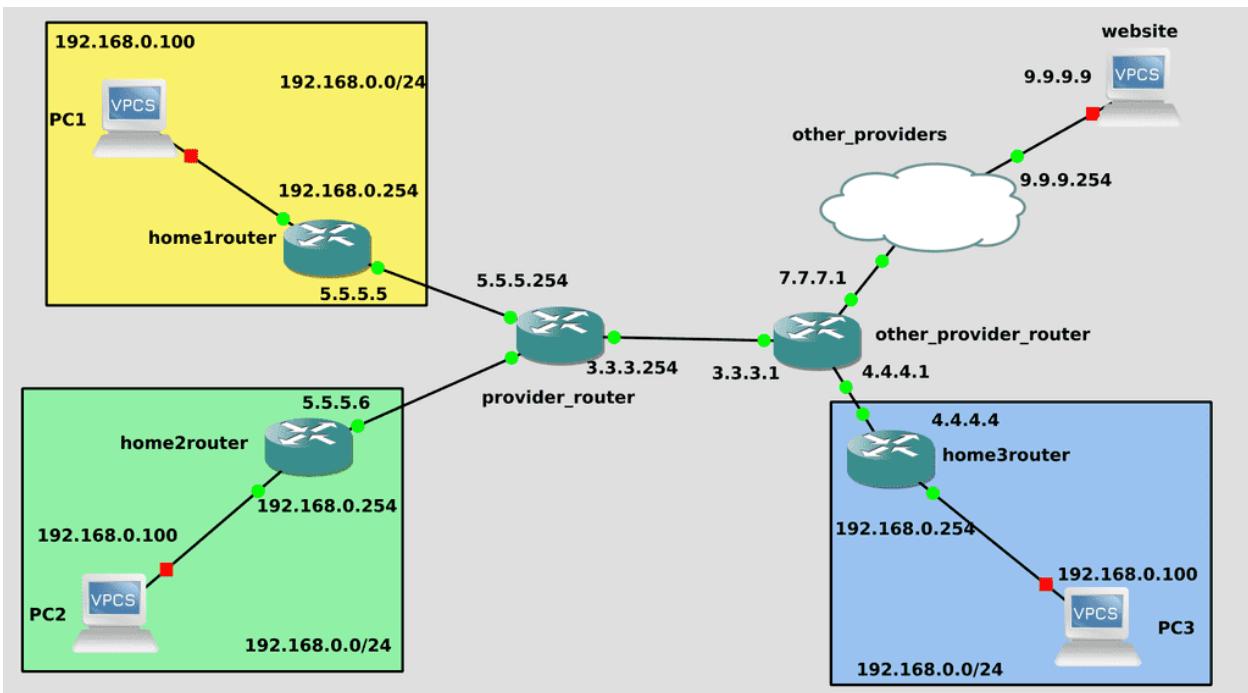
Давайте подведём итоги. Мы с вами разобрали часы реального времени и системные часы. Эти часы можно настроить с помощью различных утилит - hwclock, date и timedatectl. Но лучше всего использовать протокол NTP, чтобы на всех наших серверах было одно актуальное время, что позволит избежать многих проблем.

2.55 55. Работа с IPv6

2.55.1 55. Работа с IPv6



Когда-то давно мы посыпали бумажные письма через почту в конвертах. И на этих конвертах были поля «откуда» и «куда», где приходилось очень мелким шрифтом писать длинный адрес, потому что Земля большая и нужно указывать страну, город, улицу, номер дома и номер квартиры. И если бы каждой стране, городу и улице дали бы какое-то численное значение, получилось бы, скажем, в 10 цифр уместить точный адрес человека. Когда создавали компьютерные сети, нужно было придумать универсальный конверт - т.е. протокол. Этот протокол называется IPv4. И под поля «Откуда» и «Куда» решили выделить по 32 бита, что примерно 4 миллиарда значений. Они не учли то, что у каждого человека будет по несколько устройств, каждому из которых нужен будет адрес, не говоря уже о виртуалках и прочем.



Это ограничение в 4 миллиарда устройств давно пройдено, но сети всё ещё функционируют благодаря

NAT-у. В большинстве домов и компаний, т.е. в локальных сетях адреса одинаковые, и сотни устройств одной компании выходят в глобальную сеть по одному публичному IP адресу. Хотя это и спасает, но накладывает определённые ограничения. Представьте, что в конверте из полей «Откуда» и «Куда» убрали адрес квартиры. И если вы пошлётё курьера доставить письмо в Гугл, то он будет помнить адрес вашей квартиры и сможет вернуться к вам. Но если Гугл сам решит вам что-то отправить - письмо максимум дойдёт до вахтёраши, т.е. до вашего роутера. А она не будет знать, кому конкретно доставить, потому что квартир много, поэтому просто выкинет письмо. С одной стороны это, конечно, хорошо - вахтёраша не будет пускать непрошенных гостей и из интернета ваш компьютер не будет доступен. Но что, если вы хотите этого? Если вам нужно, чтобы ваши устройства были доступны из интернета, скажем, вы захотели у себя поднять вебсайт. Пока что это можно сделать, арендовать у провайдера публичные адреса. Но это стоит денег, с каждым годом адресов остаётся все меньше, а устройств всё больше. Адресов на самом деле не осталось, но где-то компания закроется, где-то перестанут платить - и может освободятся пару адресов. Короче, дефицит.

	Internet Protocol version 4 (IPv4)	Internet Protocol version 6 (IPv6)
Deployed	1981	1999
Address Size	32-bit number	128-bit number
Address Format	Dotted Decimal Notation: 192.149.252.76	Hexadecimal Notation: 3FFE:F200:0234:AB00: 0123:4567:8901:ABCD
Prefix Notation	192.149.0.0/24	3FFE:F200:0234::/48
Number of Addresses	$2^{32} = \sim 4,294,967,296$	$2^{128} = \sim 340,282,366,$ 920,938,463,463,374, 607,431,768,211,456

Поэтому решили переделать конверт. Если раньше выделяли 32 бита на адрес, то теперь целых 128 бит. На сей раз адресов должно хватить надолго. Переход на новый протокол решил бы многие проблемы в сетях, но не всё так просто. Основная проблема - на это должны перейти все, так как нет обратной совместимости. Каждый провайдер должен у себя ввести поддержку IPv6, обновить оборудование, донастроить - а это затраты. Всякие сайты и интернет ресурсы также должны у себя добавить поддержку IPv6 - иначе пользователи не смогут ими пользоваться. Но многих это не волнует, раз IPv4 работает - зачем что-то менять? Поэтому переход затянулся на десятки лет. Какие-то крупные компании сейчас во всю пользуются IPv6, а мелкие и средние даже не собираются переходить с IPv4.

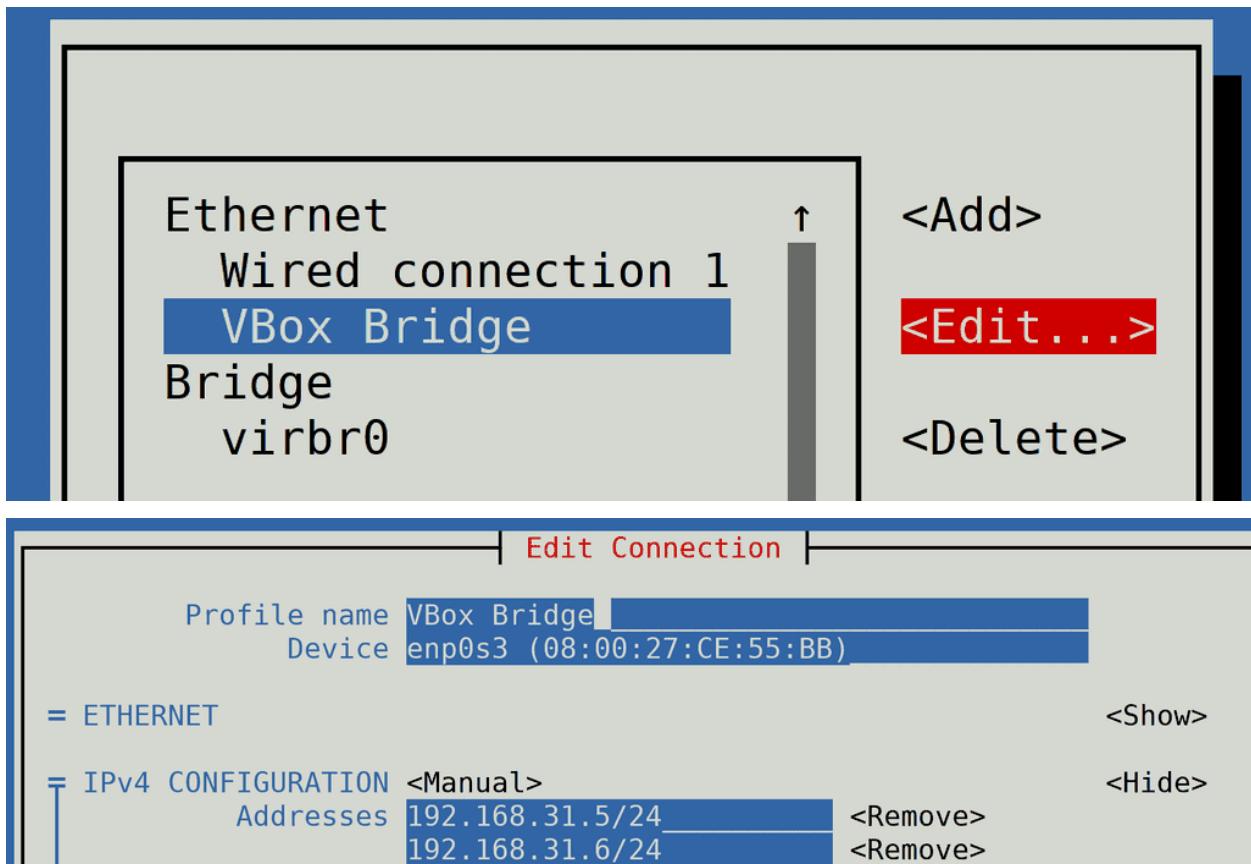
Однако время идёт и с каждым годом потребность растёт, поэтому задача администраторов - уметь с этим работать. Я не буду вдаваться в теорию, поэтому очень рекомендую посмотреть пару роликов по ссылке ([один](#), [два](#)) прежде чем продолжить эту тему. Так вы лучше поймёте, откуда берутся адреса, почему так пишутся и прочие детали. Я же разберу, как работать с IPv6 на линуксах.

```
[user@centos8 ~]$ ip a show enp0s3
3: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu
1500 qdisc fq_codel state UP group default qlen 1
000
    link/ether 08:00:27:ce:55:bb brd ff:ff:ff:ff:ff:ff
    inet 192.168.31.5/24 brd 192.168.31.255 scope
global noprefixroute enp0s3
        valid_lft forever preferred_lft forever
    inet 192.168.31.6/24 brd 192.168.31.255 scope
global secondary noprefixroute enp0s3
        valid_lft forever preferred_lft forever
[user@centos8 ~]$ 
```

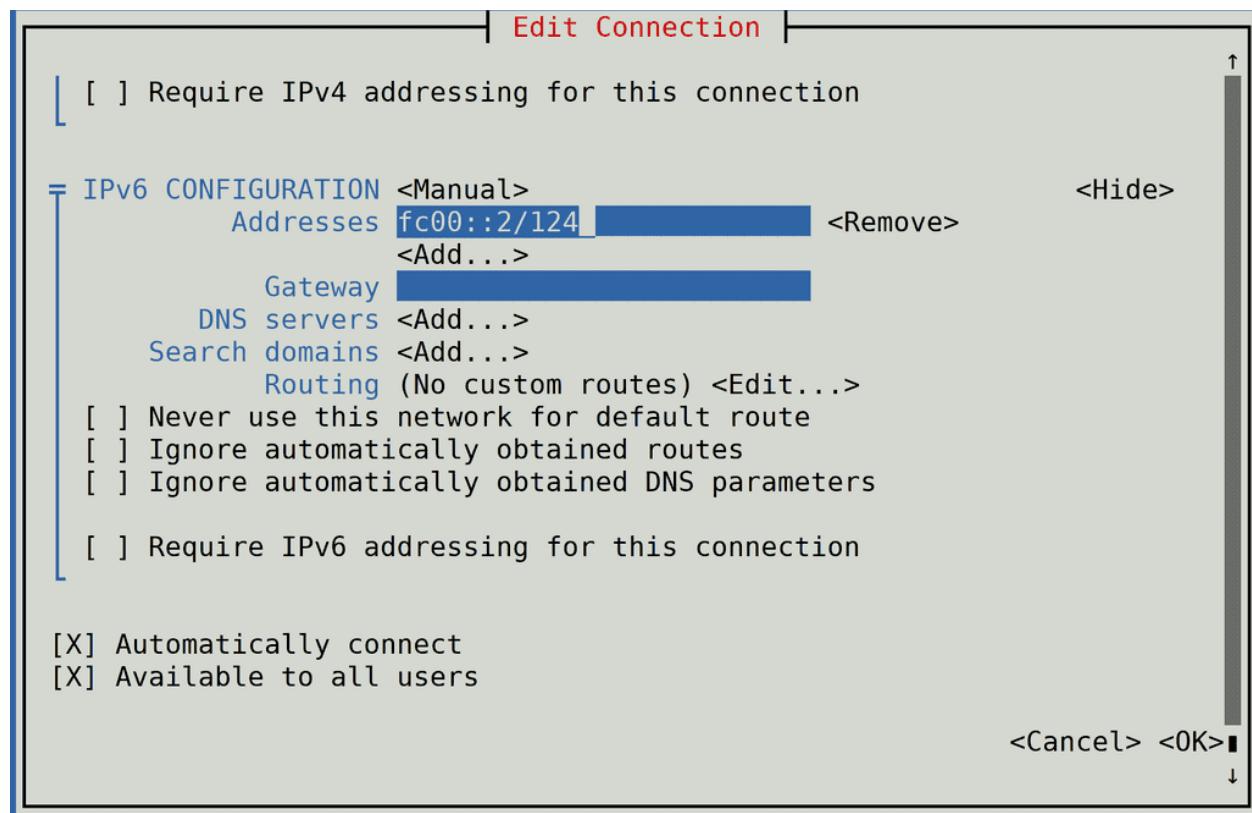


```
[user@rhel8 ~]$ ip a show enp0s3
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu
1500 qdisc fq_codel state UP group default qlen 1
000
    link/ether 08:00:27:77:f4:80 brd ff:ff:ff:ff:ff:ff
    inet 192.168.31.205/24 brd 192.168.31.255 scope
global dynamic noprefixroute enp0s3
        valid_lft 43047sec preferred_lft 43047sec
    inet6 fe80::a00:27ff:fe77:f480/64 scope link
noprefixroute
        valid_lft forever preferred_lft forever
[user@rhel8 ~]$ 
```

Для начала посмотрим список наших адресов - ip a. В строчке inet мы видим наш ipv4 адрес, а в строчке inet6 - IPv6. Видно сходство со строчкой link/ether - т.е. мак адресом. Это link local адрес, он работает только в локальной сети. В роликах по ссылкам это разбирается более детально. На Centos-е в nmtui мы отключали ipv6, поэтому там строчки с inet6 нет. Давайте включим обратно и зададим виртуалкам IPv6 адреса.



Запускаем sudo nmtui - Edit a connection - и выбираем нужное соединение. В теме про сети я переименовывал соединение и оно у меня называется VBox Bridge. Но это просто название, если нажать <Edit...> можно увидеть, что в строчке Device указан enp0s3, т.е. нужный интерфейс.



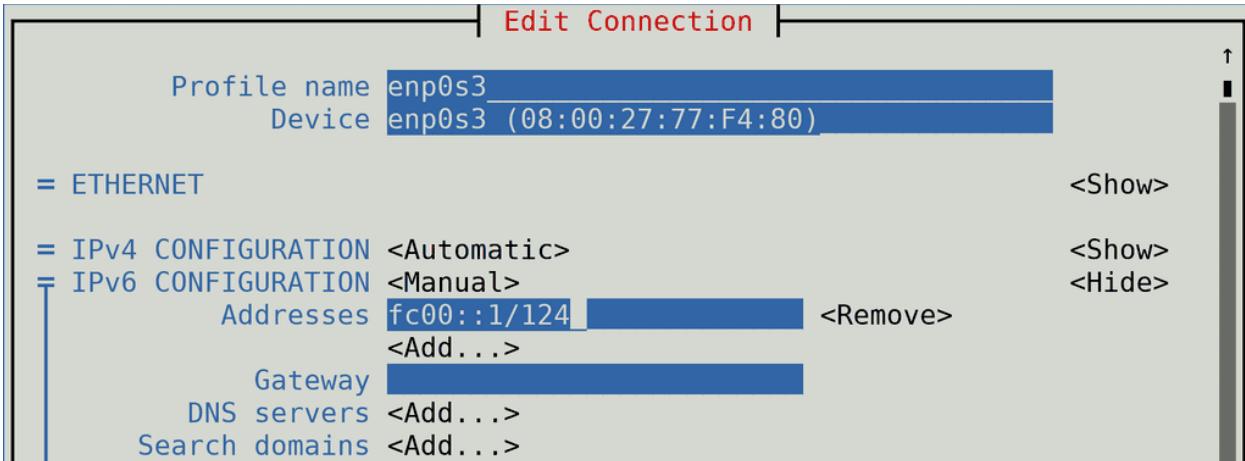
Спускаемся ниже и находим строчку «IPv6 configuration». Я его делал «Disabled», на этот раз поменяем значение на Manual. Можно было бы сделать Link Local и тогда бы интерфейс получил значение автоматически, как в RHEL, с использованием MAC адреса. Но link local адреса не всегда работают в сервисах, поэтому мы для примера используем unique local адреса. Они начинаются на fc00::. Нажимаем Show, чтобы мы могли прописать новый адрес. Дадим центосу адрес fc00::2/124 - это для сети, состоящей из 16 хостов. Gateway, DNS и прочее не буду прописывать, мой роутер не поддерживает IPv6. Мы просто протестируем IPv6 в локальной сети. Спускаемся в самый низ и нажимаем OK.

```
[user@centos8 ~]$ sudo nmcli connection up VBox\ Bridge
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/9)
[user@centos8 ~]$ sudo ip a show enp0s3
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 100
    link/ether 08:00:27:ce:55:bb brd ff:ff:ff:ff:ff:ff
    inet 192.168.31.5/24 brd 192.168.31.255 scope global noprefixroute enp0s3
        valid_lft forever preferred_lft forever
    inet 192.168.31.6/24 brd 192.168.31.255 scope global secondary noprefixroute enp0s3
        valid_lft forever preferred_lft forever
    inet6 fc00::2/124 scope global dadfailed tentative noprefixroute
        valid_lft forever preferred_lft forever
    inet6 fe80::abea:bda9:f72d:1de5/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
[user@centos8 ~]$
```

Нам нужно переактивировать соединение, чтобы применились настройки. Но если это сделать из nmtui, в момент нажатия Deactivate я потеряю связь по SSH, так как подключен по IP адресу из этого интерфейса. Потеряв связь я не смогу заново поднять интерфейс. И чтобы не приходилось это делать через консоль виртуалки, попробуем через командную строку. Для этого просто заново поднимем интерфейс с помощью nmcli:

```
sudo nmcli con up VBox\ Bridge
ip a show enp0s3
```

Как видите, теперь появилась строчка inet6 с новым адресом.



```
[user@rhel8 ~]$ sudo nmcli connection up enp0s3
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/6)
[user@rhel8 ~]$ ip a show enp0s3
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 100
  0
    link/ether 08:00:27:77:f4:80 brd ff:ff:ff:ff:ff:ff
    inet 192.168.31.205/24 brd 192.168.31.255 scope global dynamic noprefixroute enp0s3
      valid_lft 43197sec preferred_lft 43197sec
    inet6 fc00::1/124 scope global noprefixroute
      valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe77:f480/64 scope link noprefixroute
      valid_lft forever preferred_lft forever
[user@rhel8 ~]$
```

Теперь добавим адрес на rhel. Также открываем nmtui и редактируем соединение, но на этот раз пишем адрес fc00::1/124. После чего не забываем переактивировать интерфейс и убедиться, что IP адрес прописался.

```
[user@centos8 ~]$ ping fc00::1
PING fc00::1(fc00::1) 56 data bytes
From fc00::2: icmp_seq=1 Destination unreachable: Address unreachable
From fc00::2: icmp_seq=2 Destination unreachable: Address unreachable
From fc00::2: icmp_seq=3 Destination unreachable: Address unreachable
^C
--- fc00::1 ping statistics ---
4 packets transmitted, 0 received, +3 errors, 100% packet loss, time 99ms
pipe 4
[user@centos8 ~]$ ping6 fc00::1
PING fc00::1(fc00::1) 56 data bytes
From fc00::2: icmp_seq=1 Destination unreachable: Address unreachable
From fc00::2: icmp_seq=2 Destination unreachable: Address unreachable
From fc00::2: icmp_seq=3 Destination unreachable: Address unreachable
^C
--- fc00::1 ping statistics ---
4 packets transmitted, 0 received, +3 errors, 100% packet loss, time 86ms
pipe 4
[user@centos8 ~]$ █
```

Попробуем проверить, видят ли виртуалки друг друга по этим адресам. И так, первая утилита для проверки доступности хоста - ping. Как видите, пинг не идёт, адрес недоступен. Кстати, ping может пинговать как IPv4 адреса, так и IPv6. Мы можем заставить пинговать только по IPv6, если использовать утилиту ping6, или ping -6. Это может быть полезно, если за DNS именем хоста есть как IPv4 адрес, так и IPv6, и мы хотим проверить доступность именно по второму.

```
[user@centos8 ~]$ sudo firewall-cmd --list-all
public (active)
  target: DROP
  icmp-block-inversion: no
  interfaces: enp0s3
  sources:
    services: cockpit ssh
    ports: 5555/udp 5555/tcp
    protocols: ospf
    masquerade: no
    forward-ports:
    source-ports:
    icmp-blocks:
    rich rules:

[user@centos8 ~]$ sudo firewall-cmd --add-protocol=ipv6-icmp --permanent
success
[user@centos8 ~]$ sudo firewall-cmd --reload
success
[user@centos8 ~]$ █
```

Возвращаясь к тому, почему не идёт пинг. При разборе файрвола для зоны public мы поставили target - DROP, из-за чего все входящие пакеты, кроме разрешённых, сбрасывались. Для IPv4 это не вызывает проблем - мы сами можем пинговать, а нас не могут. Но IPv6 устроен несколько иначе и в нашем случае ответы на пинги от второго хоста тоже сбрасываются. И если мы хотим этого избежать, нам надо либо поменять таргет, либо добавить протокол ipv6-icmp. Ну и не забудем после этого перезагрузить

файлрол.

```
sudo firewall-cmd --add-protocol=ipv6-icmp --permanent
sudo firewall-cmd --reload
```

```
[user@centos8 ~]$ ping6 fc00::1
PING fc00::1(fc00::1) 56 data bytes
64 bytes from fc00::1: icmp_seq=1 ttl=64 time=1.81 ms
64 bytes from fc00::1: icmp_seq=2 ttl=64 time=1.03 ms
^C
--- fc00::1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 3ms
rtt min/avg/max/mdev = 1.025/1.416/1.807/0.391 ms
[user@centos8 ~]$ █
```

Опять проверим пинги - на этот раз всё работает.

```
[user@centos8 ~]$ ip -6 addr sh
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 state UNKNOWN qlen 1000
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 state UP qlen 1000
    inet6 fc00::2/124 scope global noprefixroute
        valid_lft forever preferred_lft forever
    inet6 fe80::abea:bda9:f72d:1de5/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
[user@centos8 ~]$ ip -6 route sh
::1 dev lo proto kernel metric 256 pref medium
fc00::/124 dev enp0s3 proto kernel metric 100 pref medium
fe80::/64 dev enp0s3 proto kernel metric 100 pref medium
[user@centos8 ~]$ traceroute ipv6.google.com^C
[user@centos8 ~]$ ping ipv6.google.com.
connect: Network is unreachable
[user@centos8 ~]$ █
```

Многие базовые утилиты, которые мы разбирали в теме про сети, например, ip, ping, traceroute и т.п., имеют альтернативу или опцию для работы с IPv6. К примеру, у утилиты ip есть ключ -6 - ip -6 address show покажет только ipv6 адреса; ip -6 route show покажет таблицу маршрутизации только для ipv6. Но здесь, как видите, пусто, потому что у меня нет роутера с поддержкой IPv6. На примере с pingом, в утилите traceroute также можно использовать ключ -6, либо traceroute6. Если же у вас роутер поддерживает ipv6 и вы хотите убедиться, что ваш провайдер поддерживает - вы можете попробовать пропинговать адрес ipv6.google.com.

```
ip -6 addr sh
ip -6 route sh
traceroute -6 ipv6.google.com
ping ipv6.google.com
```

```
[user@centos8 ~]$ sudo nano /etc/hosts
[user@centos8 ~]$ tail -1 /etc/hosts
fc00::1 rhel8
[user@centos8 ~]$ ping rhel8
PING rhel8(rhel8 (fc00::1)) 56 data bytes
64 bytes from rhel8 (fc00::1): icmp_seq=1 ttl=64 time=1.07 ms
^C
--- rhel8 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.071/1.071/1.071/0.000 ms
[user@centos8 ~]$ █
```

ipv6 адрес запоминать сложно, поэтому вместо адреса обычно используют доменное имя. Своего DNS сервера у нас пока нет, но мы можем прописать адрес в /etc/hosts:

```
fc00::1 rhel8
```

После чего сохраним, выйдем и проверим:

```
ping rhel8
```

Всё работает и теперь можно не мучаться со сложным адресом.

```
GNU nano 2.9.8                               /etc/chrony.conf

#minsources 2

# Allow NTP client access from local network.
#allow 192.168.0.0/16

allow 192.168.10.0/24
allow 192.168.31.0/24

allow fc00::/124█
```

Однако мало просто добавить IP адрес в систему. В настройках различных сервисов можно указывать, будет ли работать программа с IPv6 или нет. В каких-то демонах нужно явно указывать IP адрес, и, соответственно, если мы хотим добавить поддержку IPv6, то не стоит забывать указывать этот адрес. К примеру, недавно мы настраивали NTP сервер и там указали, что мы принимаем соединения только с двух сетей. И обе эти сети - IPv4. Давайте сделаем так, чтобы наш сервер был доступен по IPv6 - добавим в allow сеть. Идём на rhel и заходим в настройки chrony:

```
sudo nano /etc/chrony.conf
```

```
allow fc00::/124
```

fc00 - это адрес локальной сети. Если вам непонятно, почему именно такие значения - посмотрите видео по ссылкам, которые я давал ранее, там всё предельно понятно.

```
sudo systemctl restart chronyd
```

После изменений не забудем перезапустить chronyd.

```
[user@rhel8 ~]$ ss -l4
Netid  State    Recv-Q   Send-Q      Local Address:Port          Peer Address:Port    Process
udp    UNCONN   0          0           0.0.0.0:ntp              0.0.0.0:*
udp    UNCONN   0          0           127.0.0.1:323            0.0.0.0:*
tcp    LISTEN   0          128         0.0.0.0:ssh              0.0.0.0:*
[user@rhel8 ~]$ ss -l6
Netid  State    Recv-Q   Send-Q      Local Address:Port          Peer Address:Port    Process
icmp6  UNCONN   0          0           *:ipv6-icmp             *:*:*
udp    UNCONN   0          0           [::]:ntp                [::]:*
udp    UNCONN   0          0           [::1]:323               [::]:*
tcp    LISTEN   0          128         [::]:ssh                [::]:*
```

Давайте проверим, слушает ли chrony запросы по IPv6, используем утилиту ss:

```
ss -l4
ss -l6
```

Как видно, и в ipv4, и в ipv6 chrony слушает запросы от любых хостов по udp.

GNU nano 2.9.8	/etc/chrony.conf
# Use public servers from the pool.ntp.org project. # Please consider joining the pool (http://www.pool.ntp.org/join.html). #pool 2.centos.pool.ntp.org iburst server rhel8 iburst # Record the rate at which the system clock gains/losses time. driftfile /var/lib/chrony/drift	

Теперь вернёмся на centos и в настройках chrony укажем вместо ipv4 адреса ipv6. Вместо адреса я указал имя, всё равно за этим именем у нас только ipv6 адрес.

```
sudo systemctl restart chronyd
```

```
[user@centos8 ~]$ sudo nano /etc/chrony.conf
[user@centos8 ~]$ sudo systemctl restart chronyd
[user@centos8 ~]$ sudo chronyc sources
210 Number of sources = 1
MS Name/IP address          Stratum Poll Reach LastRx Last sample
=====
^* rhel8                      2      6     17      1  -7307ns[ -47us] +/-    49ms
[user@centos8 ~]$
```

Сохраним файл и перезапустим сервис, подождём пару секунд и проверим chronyc sources. Видим звёздочку - значит всё хорошо.

chrony был лишь примером того, что недостаточно просто прописать IP адрес, в самих сервисах порой тоже нужно что-то настроить. Обычно ничего сложного, но у каждого сервиса могут быть свои специфичные настройки.

Давайте подведём итоги. На данном этапе, пока вы изучаете основы, вам не обязательно хорошо разбираться в IPv6 - но иметь представление нужно. Вам нужно понимать, где его прописывать, как проверять, что сеть работает, делать небольшую диагностику сети и убеждаться в доступности сервисов по этому протоколу. Возможно, вы попадёте в компанию, которая активно использует IPv6 и тогда вы наберётесь знаний и опыта, а, возможно, вы будете работать в компаниях, где IPv4 будет ближайшие лет 10-20.

2.56 56. Передача файлов по сети

2.56.1 56. Передача файлов по сети

Довольно часто бывает нужно передать файлы с одной машинки на другую. В зависимости от некоторых факторов используются различные инструменты. Сегодня речь пойдёт, в основном, про небольшие объёмы данных, скажем, до 10 Гигабайт и не требующие постоянной передачи. Например, закинуть архив с программой на сервер, скопировать логи на локальную машину, сделать небольшой бэкап и всё такое. И всё это можно сделать с помощью нескольких утилит, использующих SSH для передачи данных. Это позволяет обезопасить передачу файлов, но сам протокол не оптимизирован для передачи большого объёма данных.

SCP

Если вам нужно просто передать файлы с одной машинки на другую, подойдёт утилита scp. Во многом она напоминает стандартный cp, но умеет работать с сетью. Синтаксис простой - scp откуда куда. Можно копировать с удалённой системы на локальную, с локальной на удалённую и даже с удалённой на удалённую.

```
[user@centos8 ~]$ ll file
-rw-rw-r--. 1 user user 12 Mar 26 2021 file
[user@centos8 ~]$ scp file user@192.168.31.205:~
The authenticity of host '192.168.31.205 (192.168.31.205)' can't be established.
ECDSA key fingerprint is SHA256:MR1BQ9tp5a/K3xzXCbEIu7CZXZigisvHvKrzqqbJVpA.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.31.205' (ECDSA) to the list of known hosts.
user@192.168.31.205's password:
file                                         100%   12    12.3KB/s   00:00
[user@centos8 ~]$ █
```

Начнём с простого. Мы находимся на Centos и хотим скопировать локальный файл на RHEL. Файл находится в текущей директории, поэтому путь к нему не указываем. Как и в ssh, мы должны указать пользователя, которым логинимся и адрес удалённой системы. Затем ставим двоеточие и указываем путь, куда хотим скопировать. Например, в домашнюю директорию пользователя user на rhel. Так как мы подключаемся в первый раз, увидим сообщение о ключе хоста, пишем yes и вводим пароль user-а от удалённой системы. После чего появится строчка с прогрессом копирования.

```
scp file user@192.168.31.205:~/
```

```
[user@centos8 ~]$ scp -r /home/user/temp/ rhel8:/tmp
user@rhel8's password:
test                                100%   135    98.2KB/s  00:00
myscript.old                         100%  1506   1.4MB/s  00:00
myscript                            100%  1699   1.4MB/s  00:00
errors                               100%    26   24.6KB/s  00:00
if                                    100%   145  137.6KB/s  00:00
for                                   100%   185  164.2KB/s  00:00
select                               100%    52   52.2KB/s  00:00
case                                  100%   157  175.3KB/s  00:00
users.xlsx                           100%  5046   5.4MB/s  00:00
users.csv                            100%   143  178.5KB/s  00:00
./lock.users.xlsx#                   100%    71   94.9KB/s  00:00
while                                 100%   133  161.2KB/s  00:00
until                                 100%    53   68.1KB/s  00:00
[user@centos8 ~]$
```

Теперь попробуем скопировать директорию, для этого нужен ключ `-r` - рекурсивно. На этот раз для примера укажем полный путь к локальной директории. На локальной машинке я сижу от пользователя user и на удалённой машинке есть user с таким же логином, от имени которого я подключаюсь. Поэтому мне не обязательно писать логин для удалённой машинки. Также вместо IP адреса я могу прописать имя хоста, не важно, прописано оно в ssh конфиге, `/etc/hosts`, или на DNS сервере. Ну и после двоеточия укажем полный путь к директории `/tmp`. Введём пароль и увидим, как передались все файлы.

```
scp -r /home/user/temp/ rhel8:/tmp
```

```
[user@centos8 ~]$ scp rhel8:/var/log/*.log Documents/
user@rhel8's password:
scp: /var/log/boot.log: Permission denied
dnf.librepo.log                      100%   37KB  16.3MB/s  00:00
dnf.log                             100%   94KB  38.2MB/s  00:00
dnf.rpm.log                         100%   14KB  10.8MB/s  00:00
hawkey.log                          100%   540   470.4KB/s  00:00
vboxadd-install.log                  100%   476   420.7KB/s  00:00
vboxadd-setup.log                   100%    61   84.5KB/s  00:00
[user@centos8 ~]$
```

Теперь попробуем скопировать с удалённой машинки на локальную. Т.е. теперь сперва указываем удалённую машину, а потом локальную директорию или файл, если мы хотим его перезаписать. На этот раз используем глоббинг, т.е. скопируем все файлы с директории `/var/log` на удалённой системе, заканчивающиеся на `.log`, в локальную директорию `Documents`. И всё прошло успешно.

```
scp rhel8:/var/log/*.log Documents/
```

```
!-[doctor@tardis]-
$ ssh-copy-id root@centos
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/doctor/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are
already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to
install the new keys
root@192.168.31.5's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'root@centos'"
and check to make sure that only the key(s) you wanted were added.

-[doctor@tardis]-
$ ssh-copy-id root@rhel
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/doctor/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are
already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to
```

Теперь попробуем скопировать с удалённой машинки на другую удалённую. Чтобы не приходилось вводить пароли при копировании, сперва закинем ssh ключи.

```
ssh-copy-id root@centos
ssh-copy-id root@rhel
```

```
!-[doctor@tardis]-
$ scp -r root@rhel:/etc/ssh/ root@centos:/backup
-[doctor@tardis]-
$ ssh centos ls /backup
centos8_13.03.2021_20.51.log
centos8_13.03.2021_20.51.tar.gz
centos8_13.03.2021_20.52.log
centos8_13.03.2021_20.52.tar.gz
centos8_13.03.2021_20.53.log
centos8_13.03.2021_20.53.tar.gz
ssh
-[doctor@tardis]-
$ ssh centos ls /backup/ssh
moduli
ssh_config
ssh_config.d
sshd_config
ssh_host_ecdsa_key
```

Когда всё готово, скопируем директорию /etc/ssh с rhel-а в директорию /backup на centos-е. Кстати, когда у нас есть ключи, можно использовать tab для автодополнения, т.е. путь подхватывается с удалённой системы. На этот раз вывода не будет, но можем увидеть результат просто сделав ls через ssh.

```
scp -r root@rhel:/etc/ssh root@centos:/backup
ssh centos ls /backup/ssh
```

```
[user@centos8 ~]$ scp -Cr test/ rhel8:~/  
user@rhel8's password:  
lvm2-2.03.09-5.el8.x86_64.rpm  
lvm.conf  
lvmlocal.conf  
cache-mq.profile  
cache-smq.profile  
command_profile template.profile  
100% 1597KB 27.7MB/s 00:00  
100% 100KB 19.3MB/s 00:00  
100% 2301 2.2MB/s 00:00  
100% 531 502.5KB/s 00:00  
100% 339 321.1KB/s 00:00  
100% 3030 2.8MB/s 00:00
```

Можно ещё использовать ключ **-C** для сжатия данных перед копированием.

```
scp -Cr test/ rhel8:~/
```

Если вы планируете скопировать множество мелких файлов, лучше предварительно добавить их в архив и скопировать одним файлом - так будет гораздо быстрее. Так как при копировании обрабатывается каждый файл и, если файлов тысячи, то это большая нагрузка на процессор и сеть.

Окей, с scp разобрались - утилита, которая позволяет просто скопировать файлы по сети.

SFTP

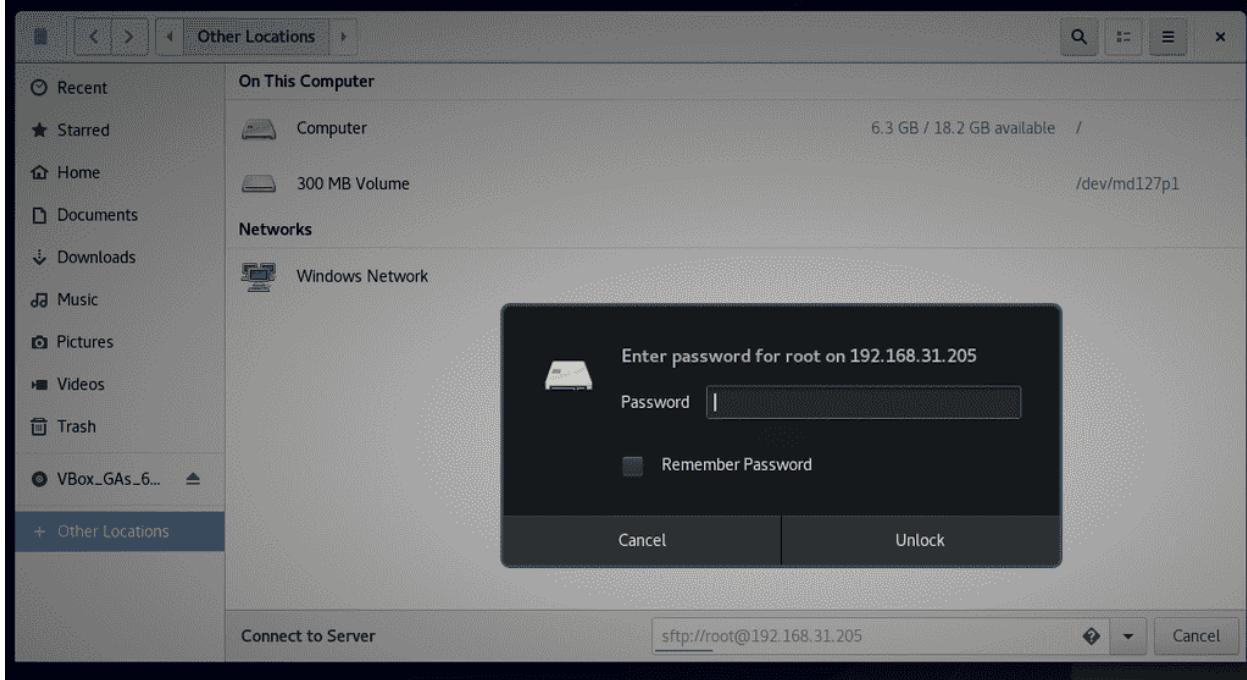
Но иногда этого недостаточно. Нередко вам нужно не только скопировать файлы, но и создать директории, изменить права, удалить какие-то файлы и т.п. Да, это всё можно сделать просто подключившись по ssh, но это не всегда удобно, особенно для тех, кто не привык работать в командной строке. И тут лучше подойдёт sftp - программа, которая позволяет взаимодействовать с файлами через ssh.

Есть отдельный протокол FTP, который позволяет делать тоже самое, и SFTP, грубо говоря, добавляет функционал FTP в SSH. Но отличия всё же есть. Например, FTP использует множество портов в отличии от SFTP, который работает на одном порту. FTP по умолчанию не шифрует соединение, для этого ему нужно добавлять сертификаты и получается FTPS. Шифрование в SFTP и FTPS работает по разному. FTPS гораздо быстрее, но его нужно устанавливать и настраивать, а SFTP по умолчанию есть на любой линукс системе. В наше время FTP и FTPS довольно редко используются, в основном, если компания хочет поделиться какими-то данными с посторонними пользователями и компаниями. В локальной сети с файлами работают с помощью других протоколов, да и в интернете в последнее время публичные облака стали популярнее для передачи файлов. Ну и легче выложить файлы на сайте и качать через HTTPS.

```
[user@centos8 ~]$ sftp user@rhel8  
user@rhel8's password:  
Connected to user@rhel8.  
sftp> ?  
Available commands:  
bye  
cd path  
chgrp [-h] grp path  
chmod [-h] mode path  
chown [-h] own path  
df [-hi] [path]  
  
exit  
get [-afPpRr] remote [local]  
reget [-fPpRr] remote [local]  
reput [-fPpRr] [local] remote  
help  
lcd path  
lls [ls-options [path]]  
lmkdir path  
ln [-s] oldpath newpath  
  
Quit sftp  
Change remote directory to 'path'  
Change group of file 'path' to 'grp'  
Change permissions of file 'path' to 'mode'  
Change owner of file 'path' to 'own'  
Display statistics for current directory or  
filesystem containing 'path'  
Quit sftp  
Download file  
Resume download file  
Resume upload file  
Display this help text  
Change local directory to 'path'  
Display local directory listing  
Create local directory  
Link remote file (-s for symlink)
```

Ладно, перейдём к SFTP. Как я уже сказал, обычно он уже стоит и ничего доустановливать не на-

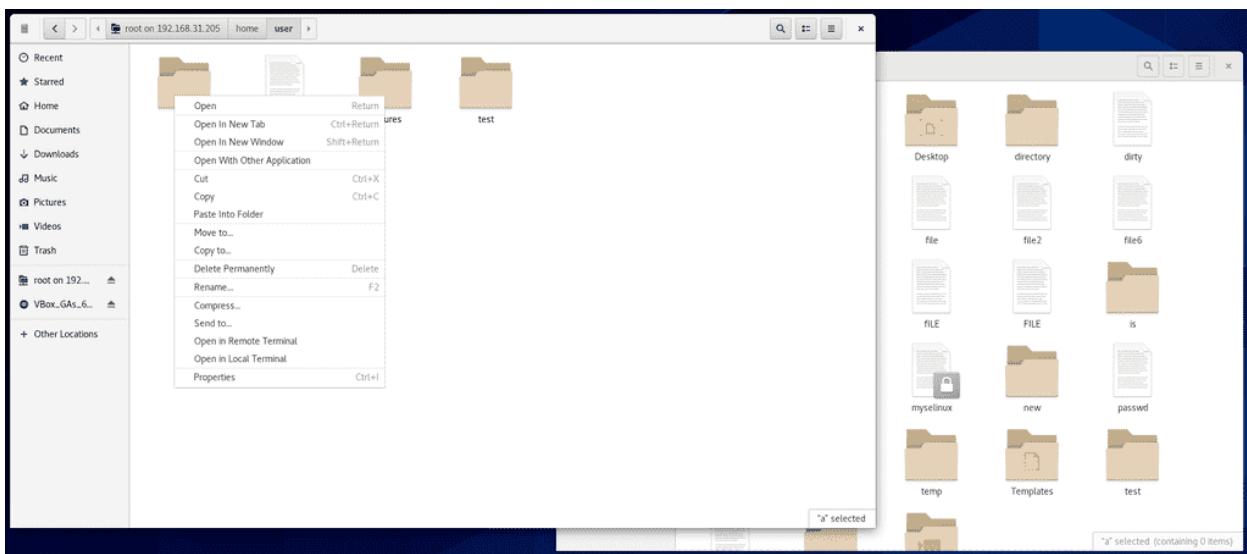
до. Вообще, с помощью утилиты sftp можно подключаться к удалённой машинке. Тогда появляется консоль с определёнными командами, которые позволяют класть или забирать файлы с удалённом машинки, менять их права и многое другое. Но это не удобно для повседневного использования. Обычно с sftp работают через графические приложения.



На Linux-е некоторые файловые менеджеры могут подключаться по SFTP. Откроем Nautilus, нажмём Other Locations и внизу в строке напишем следующее.

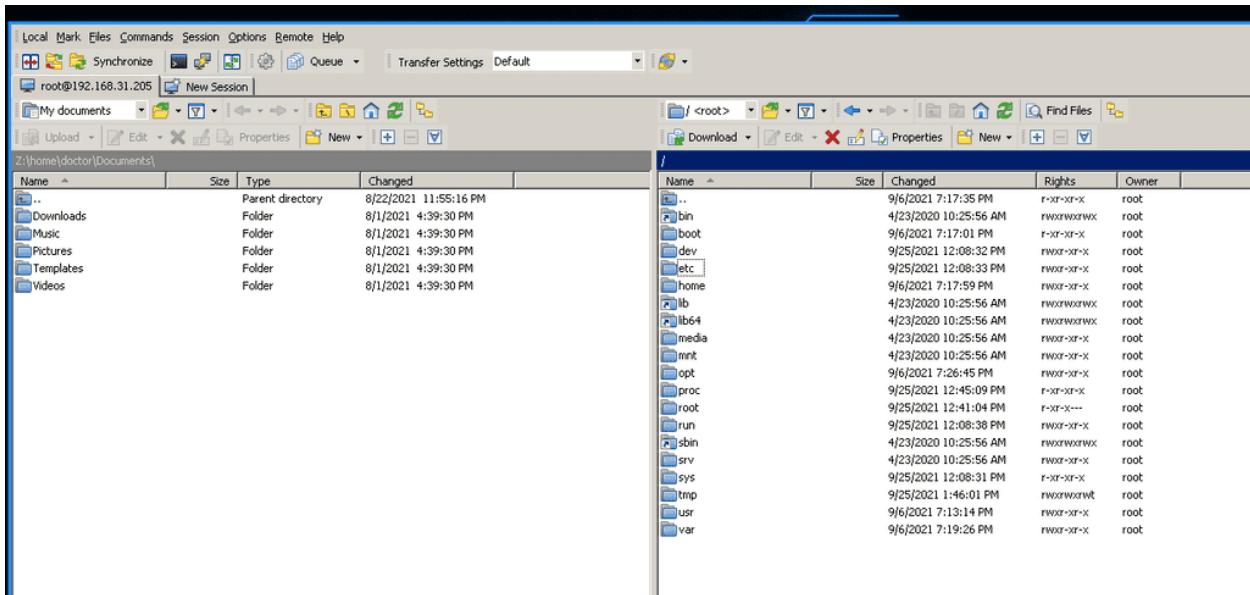
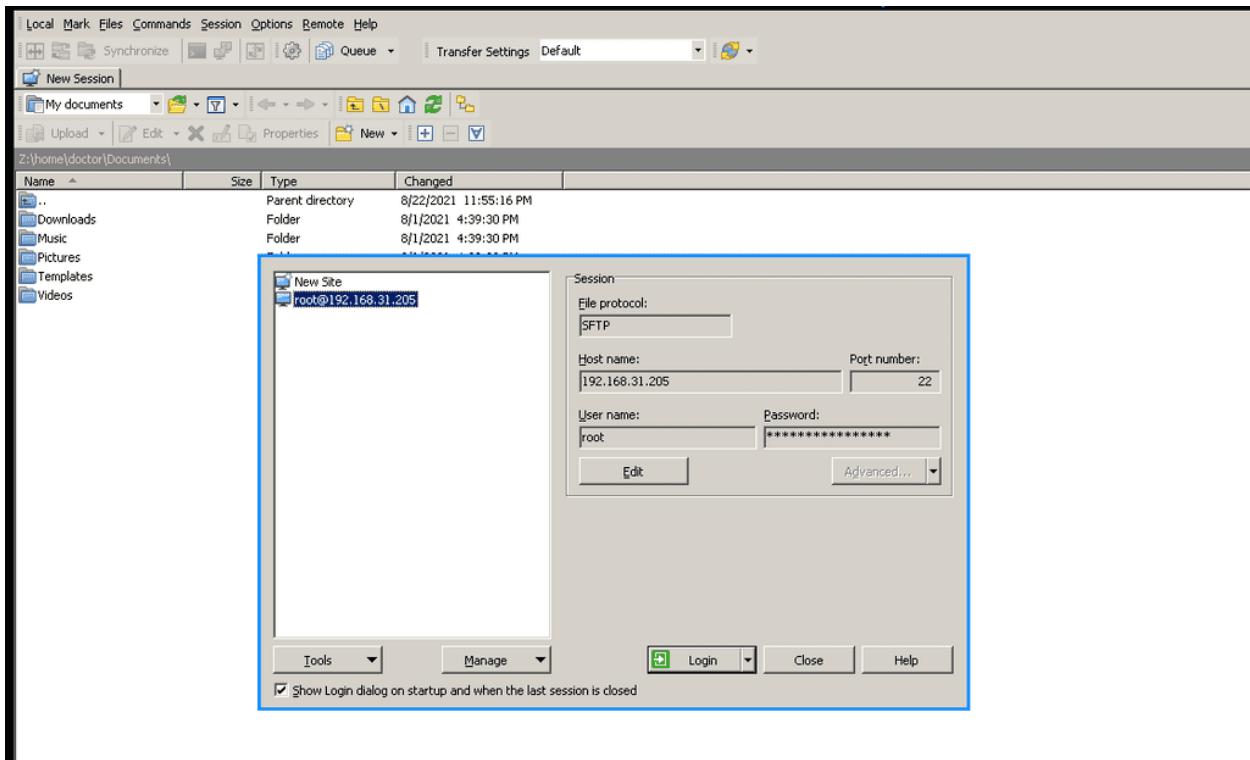
```
sftp://root@192.168.31.205
```

Как в браузере пишется http и https, так и тут можно написать протокол, двоеточие, два слэша, а дальше пользователя и адрес сервера. Программа запросит у нас пароль удалённого пользователя, после чего подключится.



И теперь мы можем взаимодействовать с удалёнными файлами почти как с локальными. Мы можем

создавать директории, изменять текстовые файлы, копировать с локальной машины на удалённую и наоборот и всё в таком духе.



Если у вас основная система Windows, то вы также можете воспользоваться различными программами для подключения к SFTP. Одна из самых популярных - WinSCP - бесплатная и свободная программа. Здесь вы можете сохранить несколько сессий, пароли и прочее. Обычно все используют две панели, где слева - ваша система, а справа - удалённая. И вы простым перетягиванием можете копировать файлы и директории, ну и всячески взаимодействовать с ними.

rsync

Рассмотрим ещё одну утилиту - rsync. В основном её используют для синхронизации данных между двумя директориями или системами. Чем-то напоминает scp, но есть отличия. Представьте, что вы скопировали директорию с различными файлами из одной системы в другую, допустим, сделали бэкап. И за неделю какие-то файлы изменились и вам нужно опять перенести эти данные. scp опять скопирует все файлы с нуля, даже если в файл добавилась всего одна строчка. Если речь о десятках гигабайт и больше - то нужно огромное количество времени.

Если же вы используете rsync, в первый раз у вас точно также передадутся все файлы. Но в следующий раз программа перед передачей проверит отличия файлов, основываясь на разных размерах и дате модификации, используя алгоритм найдёт измененную часть файлов и передаст только её. В итоге во второй раз вместо повторных 10 гигабайт скопируется всего 1 мегабайт. Т.е. произойдёт синхронизация. Изменённые данные называются дельтами, как в математике.

```
[user@centos8:~] $ mkdir test1
[user@centos8:~] $ touch test1/file{1..3}
[user@centos8:~] $ ls test1/
file1 file2 file3
[user@centos8:~] $ rsync -av test1 user@rhel8:~/
user@rhel8's password:
sending incremental file list
test1/
test1/file1
test1/file2
test1/file3

sent 236 bytes received 77 bytes 208.67 bytes/sec
total size is 0 speedup is 0.00
[user@centos8:~] $ rsync -av test1 user@rhel8:~/
user@rhel8's password:
sending incremental file list

sent 116 bytes received 17 bytes 53.20 bytes/sec
total size is 0 speedup is 0.00
[user@centos8:~] $
```

rsync тоже использует SSH для передачи данных. На самом деле он может и по другому работать, но сегодня мы разбираем только SSH. Чтобы rsync работал, он должен быть установлен на обоих системах. Давайте, для теста, создадим директорию и пару файлов в ней.

```
mkdir test1
touch test1/file{1..3}
ls test1
```

После чего использую rsync чтобы синхронизировать эту директорию на удалённую машину.

```
rsync -av test1 user@rhel8:~/
```

Синтаксис как у scp, но мы ещё использовали два ключа.

- Ключ -a - это чтобы скопировать рекурсивно, так как мы копируем директорию, плюс сохранить все права у файлов.
- Ключ -v - это verbose, чтобы мы видели, что скопировалось и какой объём этих данных.

Если мы попробуем ещё раз запустить эту команду, никаких файлов мы не увидим - потому что ничего не изменилось.

```
[user@centos8 ~]$ echo "test" > test1/file1
[user@centos8 ~]$ cp /etc/passwd test1/
[user@centos8 ~]$ rsync -av test1 user@rhel8:~
user@rhel8's password:
sending incremental file list
test1/
test1/file1
test1/passwd

sent 4,077 bytes  received 58 bytes  2,756.67 bytes/sec
total size is 3,838  speedup is 0.93
[user@centos8 ~]$
```

Теперь изменим один из файлов, что-нибудь в него запишем и ещё скопируем новый файл в эту директорию. После чего ещё раз запустим rsync:

```
echo test > test1/file1
cp /etc/passwd test1/
rsync -av test1 user@rhel8:~/
```

Как видите, на этот раз передались всего 2 файла - новый и изменённый.

```

user@centos8:~]$ echo test2 >> test1/file2
user@centos8:~]$ echo test2 >> test1/file3
user@centos8:~]$ echo test2 >> test1/passwd
user@centos8:~]$ rm test1/file1
user@centos8:~]$ rsync -avzP test1 user@rhel8:~/
user@rhel8's password:
sending incremental file list
test1/
test1/file2
      6 100%    0.00kB/s   0:00:00 (xfr#1, to-chk=2/4)
test1/file3
      6 100%    5.86kB/s   0:00:00 (xfr#2, to-chk=1/4)
test1/passwd
      3,845 100%   3.67MB/s   0:00:00 (xfr#3, to-chk=0/4)

sent 396 bytes received 113 bytes  339.33 bytes/sec
total size is 3,857 speedup is 7.58
[user@centos8 ~]$ 

```

Давайте попробуем изменить ещё пару файлов, удалить один из них и ещё раз запустить синхронизацию.

```

echo test2 >> test1/file2
echo test2 >> test1/file3
echo test2 >> test1/passwd
rm test1/file1
rsync -avzP test1 user@rhel8:~/

```

На этот раз я добавил ещё два ключа:

- z - Сжимает данные перед отправкой.
- P - показывает прогресс. С маленькими файлами это не заметно, но если синхронизируется большой файл, будет легче понять, сколько перенеслось.

```

user@centos8:~] $ ssh rhel8 ls test1/
user@rhel8's password:
file1
file2
file3
passwd
[user@centos8 ~]$ rsync -avzP --delete test1 user@rhel8:~
user@rhel8's password:
sending incremental file list
deleting test1/file1

sent 139 bytes received 32 bytes 114.00 bytes/sec
total size is 3,857 speedup is 22.56
[user@centos8 ~]$ ssh rhel8 ls test1/
user@rhel8's password:
file2
file3
passwd
[user@centos8 ~]$ 

```

Только что мы удалили файл, а удалился ли он на той стороне? Давайте проверим:

```
ssh rhel8 ls test1/
```

Как видите, хоть мы и удалили файл тут, синхронизация не удаляет файлы на той стороне. Но если мы хотим, чтобы файлы удалялись, добавим ключ `--delete`.

```
rsync -avzP --delete test1 user@rhel8:~
ssh rhel8 ls test1/
```

В выводе видно, что команда удалила файл. Можем также убедиться в этом сами с помощью `ls`.

```

[user@centos8 ~]$ rsync -avzP --delete test1/ user@rhel8:~
user@rhel8's password:
sending incremental file list
deleting test2/
deleting test1/passwd
deleting test1/file3
deleting test1/file2
deleting test1/
deleting test/usr/sbin/pvck
deleting test/usr/sbin/pvchange
deleting test/usr/sbin/lvscan
deleting test/usr/sbin/lvs

```

Работая с `rsync` важно помнить о слэше. Если вы не ставите слэш после директории, то копируется сама директория со всем содержимым. Если же поставить слэш, скопируется только содержимое директории. Обратите внимание, я просто поставил слэш после директории и это привело к тому, что

все файлы в домашней директории удалённого пользователя удалились. Всё потому, что содержимое директории test1 копировалось в домашнюю директорию, а не в директорию test1. Из-за ключа delete rsync проверил, чтобы файлы в удалённой домашней директории соответствовали файлам в директории test1. Увидев, что есть лишние файлы, rsync просто их удалил. Поэтому будьте очень осторожны с этим.

У rsync много функционала, но на пока достаточно. В целом это очень популярное решение как для бэкапа, так и для постоянной синхронизации данных между двумя серверами. Если scp для большого объёма данных не подходит, то rsync с этим справляется хорошо. Поэтому rsync можно добавить в планировщик и автоматизировать бэкапы, но не забудьте предварительно создать ssh ключи, потому что без них rsync будет просить пароль, а планировщик за вас пароль не введёт.

Давайте подведём итоги. Сегодня мы разобрали различные инструменты для передачи файлов по сети. scp часто используется для передачи небольших файлов, sftp это больше функционал, который позволяет удалённо работать с файлами, а rsync позволяет синхронизировать данные не используя много трафика. Все они используются очень часто, поэтому важно уметь ими пользоваться.

2.57 57. Сетевые файловые системы - NFS

2.57.1 57. Сетевые файловые системы - NFS

Почти в любой компании многое завязано на совместной работе с файлами. Люди постоянно пишут файлы, изменяют их и передают друг другу. Какие-то данные должны быть постоянно доступны внутри одного отдела, какие-то внутри всей компании, что-то изменяется ежеминутно, а что-то раз в год. И если бы приходилось каждый раз при каждом изменении копировать все эти файлы от одного компьютера к другому, особенно если компьютеров тысячи - то это был бы просто ужас. Это не единственная причина, в целом по разным задачам требуется, чтобы какие-то файлы были постоянно доступны на разных компьютерах. А если данные будут храниться где-нибудь на сервере, это также упростит процесс бэкапа, распределения доступов и многое другое.

В общем, всё это решают сетевые файловые системы. Грубо говоря, они позволяют работать с файлами, которые находятся на сервере, как с локальными. Например, заходя в директорию /home/user/documents вы будете видеть файлы, которые находятся на сервере, и вам не придётся постоянно копировать их туда-сюда. Сетевые файловые системы работают по протоколам и самые популярные это NFS и SMB. Они сильно отличаются и у каждого из них своё применение. Но эти протоколы почти всегда используются внутри локальной сети и почти никогда в интернете, так как это очень небезопасно.

Начнём с NFS - Network File System. Если вкратце - NFS работает с юниксовыми правами и без аутентификации. На самом деле можно настроить аутентификацию через Kerberos, но это отдельная тема. В основном NFS применяют между серверами, например, чтобы на нескольких серверах были одни и те же настройки и данные, а также для постоянного бэкапа и тому подобного. Там где обычные пользователи, особенно в Windows среде, это не очень годится, хотя тоже работает. Давайте установим NFS на RHEL-е и примонтируем на CentOS.

```
[user@centos8 ~]$ sudo dnf install nfs-utils
Updating Subscription Management repositories.
Last metadata expiration check: 0:49:10 ago on Sat 02 Oct 2021 09:11:44 AM MSK.
Dependencies resolved.
=====
 Package           Arch    Version            Repository      Size
=====
Installing:
 nfs-utils        x86_64  1:2.3.3-41.el8_4.2   rhel-8-for-x86_64-baseos-rpms  498 k
Installing dependencies:
 gssproxy          x86_64  0.8.0-19.el8       rhel-8-for-x86_64-baseos-rpms  119 k
 keyutils          x86_64  1.5.10-6.el8       rhel-8-for-x86_64-baseos-rpms  63 k
 libverto-libevent x86_64  0.3.0-5.el8       rhel-8-for-x86_64-baseos-rpms  16 k
 rpcbind           x86_64  1.2.5-8.el8       rhel-8-for-x86_64-baseos-rpms  70 k
=====
Transaction Summary
=====
Install 5 Packages

Total download size: 766 k
Installed size: 2.0 M
Is this ok [y/N]: y
```

Для начала установим пакет nfs-utils, в нём есть демон и прочие утилиты для работы с этой сетевой файловой системой :

```
sudo dnf install nfs-utils -y
```

```
[user@centos8 ~]$ sudo systemctl enable --now nfs-server
[sudo] password for user:
Created symlink /etc/systemd/system/multi-user.target.wants/nfs-server.service → /usr/lib/systemd/system/nfs-server.service.
[user@rhel8 ~]$ sudo systemctl status nfs-server
● nfs-server.service - NFS server and services
  Loaded: loaded (/usr/lib/systemd/system/nfs-server.service; enabled; vendor preset: disabled)
  Active: active (exited) since Sat 2021-10-02 10:06:39 MSK; 8s ago
    Process: 28487 ExecStart=/bin/sh -c if systemctl -q is-active gssproxy; then systemctl reload;
   Process: 28473 ExecStart=/usr/sbin/rpc.nfsd (code=exited, status=0/SUCCESS)
   Process: 28472 ExecStartPre=/usr/sbin/exportfs -r (code=exited, status=0/SUCCESS)
 Main PID: 28487 (code=exited, status=0/SUCCESS)

Oct 02 10:06:38 rhel8 systemd[1]: Starting NFS server and services...
Oct 02 10:06:39 rhel8 systemd[1]: Started NFS server and services.
[user@rhel8 ~]$
```

После установки добавим сервис nfs-server в автозагрузку и одновременно запустим его, а потом глянем его статус:

```
sudo systemctl enable --now nfs-server
sudo systemctl status nfs-server
```

Вроде всё работает.

```
[user@centos8 ~]$ sudo firewall-cmd --add-service=nfs --permanent
success
[user@rhel8 ~]$ sudo firewall-cmd --reload
success
[user@rhel8 ~]$ sudo firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp0s3
  sources:
  services: cockpit dhcpcv6-client nfs ntp ssh
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[user@rhel8 ~]$ █
```

Так как это сетевой сервис и к нему будут подключаться с других систем, нужно добавить его на файерволе. Там он есть как сервис и называется nfs. После добавления не забудем перезагрузить файервол:

```
sudo firewall-cmd --add-service=nfs --permanent
sudo firewall-cmd --reload
```

Для наглядности сделаем list-all, чтобы увидеть его в списке сервисов:

```
sudo firewall-cmd --list-all
```

```
[user@centos8 ~]$ sudo firewall-cmd --info-service=nfs
nfs
  ports: 2049/tcp
  protocols:
  source-ports:
  modules:
  destination:
  includes:
  helpers:
[user@rhel8 ~]$ sudo ss -l4tn
State      Recv-Q      Send-Q      Local Address:Port      Peer Address:Port      Process
LISTEN      0          128          0.0.0.0:37145      0.0.0.0:*
LISTEN      0          64           0.0.0.0:2049      0.0.0.0:*
LISTEN      0          128          0.0.0.0:111       0.0.0.0:*
LISTEN      0          128          0.0.0.0:20048     0.0.0.0:*
LISTEN      0          64           0.0.0.0:45585     0.0.0.0:*
LISTEN      0          128          0.0.0.0:22        0.0.0.0:*
[user@rhel8 ~]$ █
```

И давайте просто для информации посмотрим, какой порт для NFS указан на файерволе - 2049/tcp:

```
sudo firewall-cmd --info-service=nfs
```

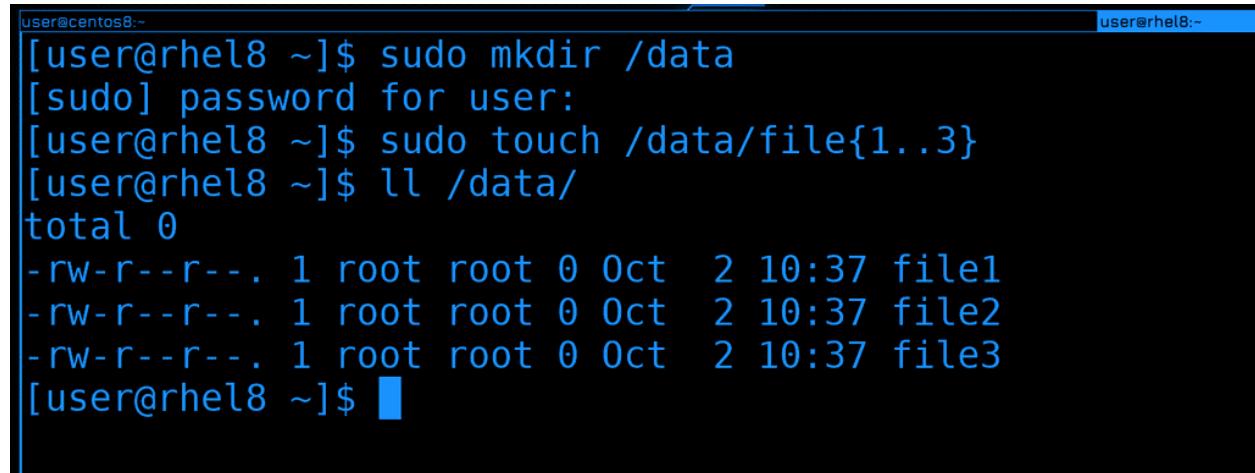
И убедимся, что сервис слушает на этом порту. Для этого у нас есть утилита ss:

```
sudo ss -l4tn
```

Как видите, в списке есть 2049 порт. Я использовал 4 ключа - listen ipv4 tcp numbers - т.е. показать все порты, на которых сервер слушает по ipv4, по протоколу tcp - и показать номера портов, а не названия.

Вся эта проверка была не обязательна, но если мы будем периодически использовать эти инструменты, мы их запомним и в дальнейшем они очень пригодятся при решении проблем.

Окей, пойдёмте дальше. Хотя мы и говорим сетевая файловая система, но нам не нужно для неё создавать раздел, форматировать его и т.п. Да, клиенты действительно монтируют её, как обычно монтируют файловые системы, но на сервере это всё лишь обычная директория. Она может быть как в корневой файловой системе, так и на какой-нибудь другой, тут уже зависит от вас, как вы распределите пространство. В целом, конечно, лучше держать всё это на другом диске с другой файловой системой, чтобы не зависеть от операционной системы. Если будут проблемы с этой операционной системой - просто отцепим диск и подключим к другой. Но для изучения сойдёт и корневая.

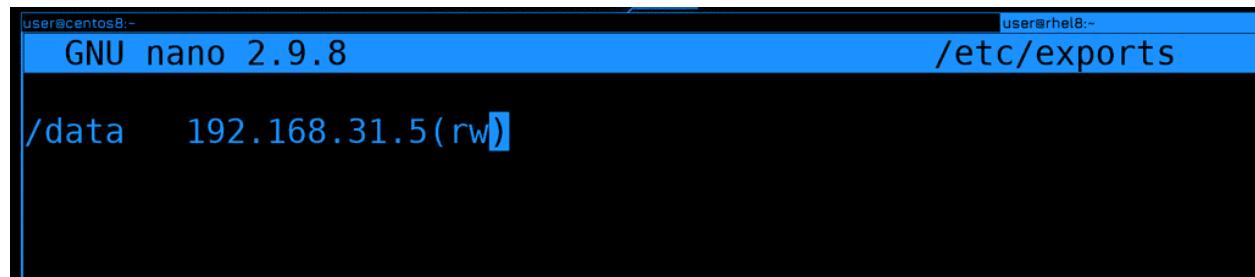


```
user@centos8:~ [user@rhel8 ~]$ sudo mkdir /data
[sudo] password for user:
[user@rhel8 ~]$ sudo touch /data/file{1..3}
[user@rhel8 ~]$ ll /data/
total 0
-rw-r--r--. 1 root root 0 Oct  2 10:37 file1
-rw-r--r--. 1 root root 0 Oct  2 10:37 file2
-rw-r--r--. 1 root root 0 Oct  2 10:37 file3
[user@rhel8 ~]$
```

Для начала создадим директорию /data в корне, название на ваше усмотрение. Ну и сразу создадим в ней пару файлов, они пригодятся нам чуть позже.

```
sudo mkdir /data
sudo touch /data/file{1..3}
ll /data
```

Вот эту директорию мы будем раздавать по сети. В простонародье это называют шарой - от слова share. Но для начала это надо настроить.



```
user@centos8:~ [user@rhel8:~]$ nano /etc/exports
GNU nano 2.9.8
/etc/exports
/data    192.168.31.5(rw)
```

Список расшаренных директорий указывается в файле /etc/exports. Раздавать можно несколько директорий и даже одну директорию можно раздавать с разными параметрами. Сначала мы указываем локальную директорию, после пробела или табуляции пишем IP адрес или адрес сети, кому мы разрешаем подключаться и слитно к адресу в скобках указываем параметры, допустим, rw - read write.

```
/data 192.168.31.5(rw)
```

В нашем случае мы разрешили компьютеру с адресом 31.5 монтировать директорию /data с возможностью изменять содержимое. К этому файлу мы ещё вернёмся, а пока продолжим.

```
[user@rhel8 ~]$ sudo nano /etc/exports
[user@rhel8 ~]$ sudo exportfs -av
exporting 192.168.31.5:/data
[data      192.168.31.5
[user@rhel8 ~]$ sudo exportfs -s
[data 192.168.31.5(sync,wdelay,hide,no_subtree_check,sec=sys,rw,secure,root_squash,no_all_squash)
[user@rhel8 ~]$
```

В случае с NFS не нужно перезапускать сервис, чтобы применились настройки, но нужно выполнить команду exportfs:

```
sudo exportfs -av
```

Здесь ключ -a считывает всё написанное в /etc/exports и применяет, а -v - это verbose - т.е. просто для наглядности.

Если запускать команду без ключей, то покажется то что сейчас раздаётся, а с ключом -s можно увидеть все параметры к этой шаре:

```
sudo exportfs
sudo exportfs -s
```

```
[user@centos8 ~]$ sudo mount 192.168.31.205:/data /mnt
[user@centos8 ~]$ df -h /mnt
Filesystem      Size  Used Avail Use% Mounted on
192.168.31.205:/data  11G  2.3G  8.0G  23% /mnt
[user@centos8 ~]$ df -h /
Filesystem      Size  Used Avail Use% Mounted on
/dev/mapper/cl_centos8-root  17G  12G  5.9G  66% /
[user@centos8 ~]$ mount | grep /mnt
192.168.31.205:/data on /mnt type nfs4 (rw,relatime,vers=4.2,rsize=131072,wsize=131072,namlen=255,hard,proto=tcp,timeo=600,retrans=2,sec=sys,clientaddr=192.168.31.5,local_lock=none,addr=192.168.31.205)
[user@centos8 ~]$
```

Окей, теперь попробуем примонтировать эту директорию на CentOS. На нём уже пакет nfs-utils есть, поэтому сразу запускаем команду mount:

```
sudo mount 192.168.31.205:/data /mnt
```

В качестве файловой системы указываем IP адрес двоеточие и расшаренная директория. Ну и указываем куда хотим примонтировать. Обратите внимание, что никакого пароля не нужно указывать.

Если посмотреть df, можно увидеть объём файловой системы и свободное пространство на сервере. Для сравнения, на локальной машине другие данные:

```
df -h /mnt
df -h /
```

Ну и также можно увидеть с какими параметрами мы примонтировали, если посмотреть команду mount:

```
mount | grep /mnt
```

The screenshot shows two terminal windows side-by-side. The left window is titled 'user@centos8 ~' and displays the command 'ls /mnt' followed by three files: 'file1', 'file2', and 'file3'. The right window is titled 'user@rhel8 ~' and shows the command 'cat /mnt/file1' which outputs nothing. Below these, the command 'sudo touch /mnt/file4' is run, followed by a password prompt and the error message 'touch: cannot touch '/mnt/file4': Permission denied'.

Теперь перейдём к данным. И так, если посмотреть в /mnt, можно увидеть файлы. Т.е. я на CentOS-е в /mnt вижу файлы, которые находятся в /data на RHEL-е. Файлы пустые, поэтому cat ничего не показывает.

```
ls /mnt
cat /mnt/file1
```

Но если я попытаюсь создать какой-то файл, у меня выйдет ошибка, что не хватает прав.

```
sudo touch /mnt/file4
```

В NFS действуют юниксовые права, т.е. на основе UID-ов и GID-ов. Но, по-умолчанию, права рута игнорируются.

The screenshot shows a terminal window titled 'user@centos8 ~' running the command 'stat /mnt/file1'. The output provides detailed information about the file, including its type (regular empty file), size (0), blocks (0), device (31h/49d), inode (9279700), links (1), access and modification times (both 2021-10-02 11:37:49.733889521 +0400), birth time (-), and ownership (uid: 0/root, gid: 0/root). The 'Access' line shows the permissions as (0644/-rw-r--r--).

Для начала вспомним, что права на файлы в линуксах применяются не по логинам, а по идентификатору - UID-у:

```
stat /mnt/file1
```

У file1 UID - 0, т.е. рут.

```
[user@centos8 ~]$ ssh root@rhel8 chown user /data/file1
root@rhel8's password:
[user@centos8 ~]$ stat /mnt/file1
  File: /mnt/file1
  Size: 0          Blocks: 0          IO Block: 131072 regular empty file
Device: 31h/49d Inode: 9279700      Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/    user)   Gid: (     0/    root)
Context: system_u:object_r:nfs_t:s0
Access: 2021-10-02 13:05:25.611343445 +0400
Modify: 2021-10-02 13:11:15.630446593 +0400
Change: 2021-10-02 13:11:21.575431533 +0400
 Birth: -
[user@centos8 ~]$ ssh root@rhel8 id user
root@rhel8's password:
uid=1000(user) gid=1000(user) groups=1000(user),10(wheel)
[user@centos8 ~]$ id user
uid=1000(user) gid=1000(user) groups=1000(user),10(wheel),190(systemd-journal),1001(group1)
[user@centos8 ~]$ █
```

Но права рута не работают, поэтому давайте поменяем владельца файла. Отсюда у нас не хватит прав, поэтому сделаем это через ssh:

```
ssh root@rhel8 chown user /data/file1
```

Чуточку подождём и увидим, что у файла поменялся владелец - теперь он показывается как user. Но это лишь потому, что и на сервере у пользователя user UID - 1000, и у локального пользователя user UID - 1000.

```
[user@centos8:/home/user
[user@centos8 ~]$ echo test > /mnt/file1
[user@centos8 ~]$ cat /mnt/file1
test
[user@centos8 ~]$ su
Password:
[root@centos8 user]# echo test2 > /mnt/file1
bash: /mnt/file1: Permission denied
[root@centos8 user]# cat /mnt/file1
test
[root@centos8 user]# █
```

Давайте проверим права, попробуем записать что-то в файл:

```
echo test > /mnt/file1
cat /mnt/file1
```

Как видите, всё прошло успешно, в файле появилась строка. Теперь попробуем сделать тоже самое от рута. Я специально не использую sudo, потому что перенаправление работает работает от моего юзера, а не от рута. Т.е. если я использую sudo, в файл текст будет добавляться не от рута, а от user-a, поэтому это не показатель. Заходим от рута и пытаемся изменить файл:

```
su
echo test2 > /mnt/file1
```

И снова получаем ошибку. Так почему же права рута игнорируются? Всё дело в том, что без этого пользователь рут на любой из систем, куда примонтирована шара, смог бы сделать с файлами всё что угодно, у него были бы все права на все файлы внутри шары. Он смог бы, допустим, создать файл с setUID-ом и повысить себе привилегии на любой из систем. А это опасно - если кто-то получил доступ к одному из ваших серверов, то так он может и на других повысить себе привилегии. Ну или какой-то из ваших сотрудников благодаря этому получит доступ туда, куда ему нельзя заходить.

```
[user@centos8 ~]$ sudo nano /etc/exports
[sudo] password for user:
[user@centos8 ~]$ cat /etc/exports
/data 192.168.31.5(rw,no_root_squash)
[user@centos8 ~]$ sudo exportfs -rv
exporting 192.168.31.5:/data
[user@centos8 ~]$
```

Не стоит это отключать, но о существовании такой возможности вы должны знать. В /etc/exports вы можете добавить параметр no_root_squash к нужной шаре:

```
/data 192.168.31.5(rw,no_root_squash)
```

Обратите внимание, что параметры указываются через запятую, без пробелов. Чтобы заново расшарить директорию с новыми параметрами, нужно использовать ключ -r:

```
sudo exportfs -rv
```

```
[root@centos8 ~]# echo test2 > /mnt/file1
[root@centos8 ~]# cat /mnt/file1
test2
[root@centos8 ~]#
```

Ну и теперь можно на CentOS-е работать с файлами от рута:

```
echo test2 > /mnt/file1
cat /mnt/file1
```

```
[user@centos8 ~]$ ssh root@rhel8 useradd rheluser -m -b /data/ -u 5000
root@rhel8's password:
[user@centos8 ~]$ ls -ld /mnt/
file1   file2   file3   rheluser/
[user@centos8 ~]$ ls -ld /mnt/rheluser/
drwx----- 2 5000 5000 62 Oct  2 13:42 /mnt/rheluser/
[user@centos8 ~]$ █
```

Чтобы закрепить тему прав, на сервере создадим нового пользователя rheluser с uid-ом 5000. И пусть его домашняя директория будет внутри /data:

```
ssh root@rhel8 useradd rheluser -m -b /data/ -u 5000
```

Теперь давайте проверим права на новую директорию внутри /mnt:

```
ls -ld /mnt/rheluser
```

Обратите внимание, что вместо владельца и группы директории указано 5000, т.е. UID. Всё потому, что на локальной системе нет пользователя с таким идентификатором.

```
[centosuser@centos8 ~]$ sudo useradd centosuser -d /mnt/rheluser/ -u 5000
[sudo] password for user:
useradd: warning: the home directory already exists.
Not copying any file from skel directory into it.
[user@centos8 ~]$ ls -ld /mnt/rheluser/
drwx----- 2 centosuser centosuser 62 Oct  2 13:42 /mnt/rheluser/
[user@centos8 ~]$ sudo su centosuser
[centosuser@centos8 user]$ ls
ls: cannot open directory '.' : Permission denied
[centosuser@centos8 user]$ cd
[centosuser@centos8 rheluser]$ pwd
/mnt/rheluser
[centosuser@centos8 rheluser]$ touch file
[centosuser@centos8 rheluser]$ █
```

Давайте создадим локального пользователя с другим логином, но тем же самым UID-ом. И в качестве домашней директории укажем ему директорию того пользователя на сетевой шаре:

```
sudo useradd centosuser -d /mnt/rheluser/ -u 5000
```

Теперь, если проверить права на ту же директорию, вместо 5000 мы увидим локального пользователя. Т.е. система по UID-у сопоставляет файл с владельцем:

```
ls -ld /mnt/rheluser
```

Давайте попытаемся залогиниться новым юзером:

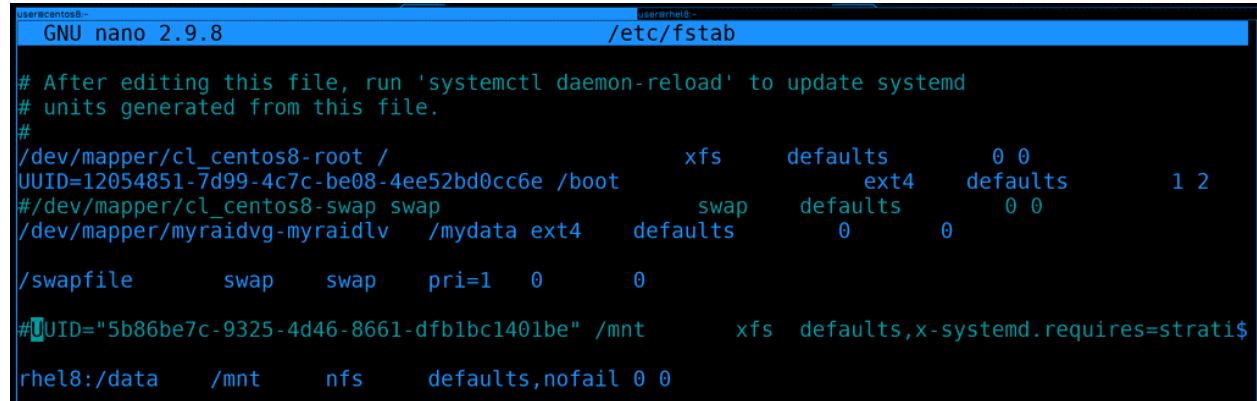
```
sudo su centosuser
cd
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
pwd
touch file
```

Теперь можно зайти к себе домашнюю директорию и работать как обычно, при этом файлы этого пользователя будут храниться на сервере. Но для этого нужно, чтобы эта файловая система была предварительно примонтирована.



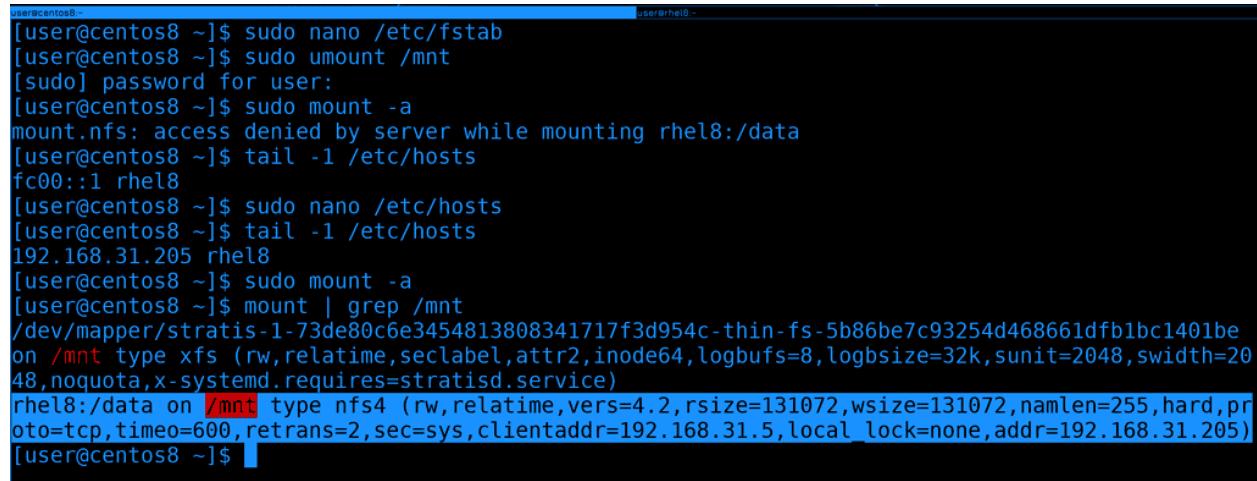
```
user@centos8:~$ nano /etc/fstab
user@rhel8:~$ /etc/fstab

# After editing this file, run 'systemctl daemon-reload' to update systemd
# units generated from this file.
#
#/dev/mapper/cl_centos8-root /           xfs      defaults        0 0
UUID=12054851-7d99-4c7c-be08-4ee52bd0cc6e /boot   ext4    defaults        1 2
#/dev/mapper/cl_centos8-swap swap       swap      defaults        0 0
/dev/mapper/myraidvg-myraidlv /mydata ext4    defaults        0 0
/swappfile     swap     swap     pri=1  0      0
#UUID="5b86be7c-9325-4d46-8661-dfb1bc1401be" /mnt      xfs  defaults,x-systemd.requires=stratis
rhel8:/data   /mnt     nfs      defaults,nofail 0 0
```

А чтобы при каждом включении нам не приходилось монтировать вручную, нам на клиенте, т.е. на CentOS, надо добавить запись в /etc/fstab:

```
rhel8:/data /mnt nfs defaults,nofail 0 0
```

Давайте вместо адреса укажем имя хоста, хотя можно и по IP адресу. Ну и в качестве типа файловой системы указываем nfs. Раньше в опциях монтирования стоило указывать дополнительный параметр _netdev, чтобы эта файловая система при включении монтировалась только после того, как заработает сеть. Сейчас это не нужно, потому что systemd сам за этим следит. Но на всякий случай добавим опцию nofail. Если сеть будет недоступна, операционная система не запустится в нормальном режиме, а с ключом nofail ошибка монтирования проигнорируется. В определённых случаях нельзя давать операционной системе запуститься, если какая-то важная файловая система не примонтировалась, а в каких-то случаях это не критично.



```
[user@centos8 ~]$ sudo nano /etc/fstab
[user@centos8 ~]$ sudo umount /mnt
[sudo] password for user:
[user@centos8 ~]$ sudo mount -a
mount.nfs: access denied by server while mounting rhel8:/data
[user@centos8 ~]$ tail -1 /etc/hosts
fc00::1 rhel8
[user@centos8 ~]$ sudo nano /etc/hosts
[user@centos8 ~]$ tail -1 /etc/hosts
192.168.31.205 rhel8
[user@centos8 ~]$ sudo mount -a
[user@centos8 ~]$ mount | grep /mnt
/dev/mapper/stratis-1-73de80c6e3454813808341717f3d954c-thin-fs-5b86be7c93254d468661dfb1bc1401be
on /mnt type xfs (rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,sunit=2048,swidth=20
48,noquota,x-systemd.requires=stratis.service)
rhel8:/data on /mnt type nfs4 (rw,relatime,vers=4.2,rsize=131072,wsize=131072,namlen=255,hard,pr
oto=tcp,timeo=600,retrans=2,sec=sys,clientaddr=192.168.31.5,local_lock=none,addr=192.168.31.205)
[user@centos8 ~]$
```

Чтобы убедиться, что с fstab всё нормально, попробуем примонтировать с помощью него. Предварительно шару следует отмонтировать, а потом использовать команду mount -a, чтобы примонтировать всё из fstab:

```
sudo umount /mnt
sudo mount -a
```

Как видите, у меня возникла ошибка, что доступ запрещён. Это потому что rhel8 у меня прописан как ipv6 адрес, а его я в exports не указывал. Давайте поменяем hosts файл и пропишем ipv4 адрес, после чего попробуем перемонтировать. На этот раз всё примонтировалось успешно.

```
user@centos8:~$ 
user@rhel8:~$ 
GNU nano 2.9.8
/etc/exports

/data      192.168.31.5(rw,no_root_squash) 192.168.31.0/24(rw)
/data/rheluser *(ro)
```

Теперь давайте вернёмся к файлу exports и разберём пару опций, но я не буду лезть в дебри и показывать какие-то специфичные настройки. На данном этапе нам это не нужно, а если вам интересно, посмотрите man по exports.

И так, до этого мы расширили директорию /data для адреса 31.5. Что, если мы хотим эту же директорию расшарить для всей сети 31.0, но с другими параметрами? Можем поставить пробел и написать другой IP адрес или адрес сети с другими параметрами:

```
/data      192.168.31.5(rw,no_root_squash) 192.168.31.0/24(rw)
```

Но ещё раз обратите внимание, что скобки слитны с адресом. Если поставить пробел - то параметры будут применяться для любых адресов. И так, у нас для 31.5 есть параметр no_root_squash, т.е. он может от рута работать с этой файловой системой, а другие хосты в этой сети могут работать только обычными юзерами, без рут прав. Вообще, желательно, внутри /data создать различные директории, задать им соответствующих владельцев и так шарить, иначе в самой директории /data с её владельцем рутом никто не сможет создавать новые директории и файлы, только читать.

```
/data/rheluser *(ro)
```

Каждую шару надо указывать с новой строки. Это может быть и отдельная директория и директория внутри существующей шары. Так вы можете более гранулярно монтировать, т.е. не всё содержимое /data видеть на клиентах, а только необходимое для данного хоста. В данном случае /data/rheluser раздаётся на все компьютеры с правами read only. Даже если на уровне файлов у клиента будет доступ для изменения, nfs любые попытки изменить файлы будет пресекать.

```
[user@centos8 ~]$ sudo nano /etc/exports
[sudo] password for user:
[user@rhel8 ~]$ sudo exportfs -rv
[sudo] password for user:
exporting 192.168.31.5:/data
exporting 192.168.31.0/24:/data
exporting *:/data/rheluser
[user@rhel8 ~]$ sudo exportfs -s
/data 192.168.31.5(sync,wdelay,hide,no_subtree_check,sec=sys,rw,secure,no_root_squash,no_all_squash)
/data 192.168.31.0/24(sync,wdelay,hide,no_subtree_check,sec=sys,rw,secure,root_squash,no_all_squash)
[data/rheluser *(sync,wdelay,hide,no_subtree_check,sec=sys,ro,secure,root_squash,no_all_squash)
[user@rhel8 ~]$
```

После изменений не забываем переэкспортировать файловые системы:

```
sudo exportfs -rv
sudo exportfs -s
```

Давайте разберём ещё пару нюансов. То что мы обсуждали насчёт NFS, в основном относится к его текущей версии - NFS 4. Где-то теоретически вы можете наткнуться ещё на 3 версию, и вот между версиями есть большая разница. Например, сегодня мы на файрволе открыли порт 2049 и этого хватило, чтобы клиент мог примонтировать шару. Раньше бы этого могло не хватить, потому что nfs изначально разрабатывался с учётом сервиса rpcbind.

RPCbind это специальный сервис, который слушает на порту 111. Грубо говоря, он позволяет клиентам не думать о том, к какому именно порту нужно подключаться. Условно, если вы хотите подключиться к SSH, вам не обязательно знать, на каком именно порту он работает. Условный SSH каждый раз работает на каком-то рандомном порту. И ваша программа за вас посыпает запрос на RPCBind на порт 111 и спрашивает, а как достучаться до SSH. А там rpcbind ведёт учёт всех портов и направляет на нужный. Идея интересная, но не всегда удобная, особенно с точки зрения файрвола. Сейчас этот сервис редко где используется,

Так вот, некоторые функции NFS разбросаны по разным портам. В 4 версии NFS большинство функционала включено в сам протокол, но во 2 и 3 нет. Например, функция блокировки файла. Но даже для NFS 4 RPCbind может быть полезен. В частности для получения информации о шарах.

```
[user@centos8 ~]$ showmount rhel8
clnt_create: RPC: Unable to receive
[user@centos8 ~]$ ssh root@rhel8 firewall-cmd --add-service={rpc-bind,mountd} --permanent
root@rhel8's password:
success
[user@centos8 ~]$ ssh root@rhel8 firewall-cmd --reload
root@rhel8's password:
success
[user@centos8 ~]$ showmount rhel8
Hosts on rhel8:
[user@centos8 ~]$ showmount -e rhel8
Export list for rhel8:
/data/rheluser *
/data          192.168.31.0/24
[user@centos8 ~]$ █
```

К примеру, мы знаем, что на rhel есть nfs сервер, но не знаем, что за директории там раздаются. И по каким-то причинам мы не можем на него зайти и проверить всё. Мы хотим со стороны клиента узнать, что же там на сервере шарится. Для этого есть команда showmount:

```
showmount rhel8
```

Как видите, она выдаёт ошибку «RPC: unable to receive». Т.е. не получает ответ от RPC.

Чтобы решить эту проблему, надо на файрволе сервера разрешить rpc-bind и mountd. Сделаем это прямо отсюда, через ssh:

```
ssh root@rhel8 firewall-cmd --add-service={rpc-bind,mountd} --permanent
ssh root@rhel8 firewall-cmd --reload
```

И теперь команда showmount показывает экспортнутые директории на сервере:

```
showmount -e rhel8
```

```
[user@centos8:~]$ df -h /mnt
Filesystem      Size  Used Avail Use% Mounted on
rhel8:/data     11G   2.4G  8.0G  23% /mnt
[user@centos8 ~]$ ssh root@rhel8 poweroff
root@rhel8's password:
Connection to rhel8 closed by remote host.
[user@centos8 ~]$ df -h /mnt
^C
[user@centos8 ~]$ cd /mnt/
[user@centos8 mnt]$ ls
^C
[user@centos8 mnt]$ cd ..
[user@centos8 ~]$ cat /mnt/file
```

Ну и ещё один нюанс. Предположим NFS сервер вышел из строя или клиент потерял сеть. В общем, просто потерялась связь между клиентом и сервером, в то время как файловая система примонтирована. Давайте создадим такую ситуацию - выключим rhel, пока примонтирована шара:

```
df -h /mnt
ssh root@rhel8 poweroff
```

```
[user@centos8 ~]$ sudo umount /mnt
^C[sudo] password for user:

[user@centos8 ~]$ sudo umount -f /mnt
umount.nfs4: /mnt: device is busy
[user@centos8 ~]$ sudo lsof | grep mnt
^C
[user@centos8 ~]$ sudo umount -f -l /mnt
[user@centos8 ~]$ ls /mnt/
[user@centos8 ~]$ df -h /mnt
Filesystem      Size  Used Avail Use% Mounted on
/dev/mapper/cl_centos8-root  17G   12G  5.9G  66% /
[user@centos8 ~]$ █
```

Если теперь попробовать зайти в директорию или хоть как-то взаимодействовать с файлами, то процесс не будет выдавать результата, он просто не может зайти в эту директорию. Если какой-то процесс работал с файлами внутри этой директории, то он просто зависнет в ожидании ответа. И в такие моменты лучшее решение - попробовать восстановить связь, иначе придётся убивать процессы. Но если у вас нет никаких важных процессов и вы просто забыли отмонтировать, то обычная попытка отмонтировать тоже будет зависать. В таких случаях вы можете попробовать насилием отмонтировать

с ключом -f:

```
sudo umount -f /mnt
```

Но, как видите, у меня даже это не работает, пишет, что устройство занято. Попробуем найти процесс с помощью lsof и убить его:

```
sudo lsof /mnt
```

И даже lsof у меня перестаёт отвечать.

В таких случаях можно попытаться найти процесс в списке процессов или использовать ключ -l с umount:

```
sudo umount -f -l /mnt
```

Тогда файловая система для нас отмонтируется, но где-то в памяти всё ещё останется, пока всё не исправится или пока мы не перезагрузимся.

Давайте подведём итоги. Сегодня мы с вами разобрали NFS - одну из сетевых файловых систем, которая много где используется. Её легко поставить и настроить, но будьте осторожнее с доступами, так как для подключения к ней не нужны никакие логины и пароли.

2.58 58. Сетевые файловые системы - SMB

2.58.1 58. Сетевые файловые системы - SMB

Как мы выяснили, NFS в основном используется в линуксовой среде и не очень годится для совместной работы пользователей с файлами. В компаниях большинство пользователей используют Windows, где права устроены иначе. И в таком окружении почти всегда используется протокол SMB (Server Message Block), который годится не только для работы с файлами, но и с принтерами. Вместо того, чтобы устанавливать на всех компьютерах драйвера от принтеров, достаточно это сделать на одном сервере. Компьютеры будут посыпать запросы на этот сервер, а сервер будет печатать на принтерах.

SMB также часто называют CIFS (Common Internet File System). Зачастую файловую шару с SMB поднимают на Windows серверах, но и на Linux-ах она не редкость. Как и в случае с NFS, файловые шары это просто директории, доступные по сети. Но права здесь работают абсолютно иначе. SMB - это лишь протокол, а для его работы нужна программа.

Установка

```
[user@rhel8 ~]$ sudo dnf install samba
Updating Subscription Management repositories.
Last metadata expiration check: 0:00:13 ago on Sat 09 Oct 2021 12:51:54 PM MSK.
Dependencies resolved.
=====
Package           Arch    Version      Repository      Size
=====
Installing:
samba            x86_64  4.13.3-4.el8_4   rhel-8-for-x86_64-baseos-rpms  846 k
Installing dependencies:
samba-common-tools x86_64  4.13.3-4.el8_4   rhel-8-for-x86_64-baseos-rpms  498 k
samba-libs        x86_64  4.13.3-4.el8_4   rhel-8-for-x86_64-baseos-rpms  168 k
Transaction Summary
=====
Install 3 Packages

Total download size: 1.5 M
Installed size: 4.0 M
Is this ok [y/N]: y
```

На Linux-ах она называется samba. Давайте для начала поднимем файловый сервер на rhel-е:

```
sudo dnf install samba
```

```
[user@rhel8 ~]$ sudo systemctl enable --now smb
Created symlink /etc/systemd/system/multi-user.target.wants/smb.service → /usr/lib/systemd/system/smb.service.
[user@rhel8 ~]$ sudo firewall-cmd --add-service=samba --permanent
success
[user@rhel8 ~]$ sudo firewall-cmd --reload
success
[user@rhel8 ~]$ sudo systemctl status smb
● smb.service - Samba SMB Daemon
  Loaded: loaded (/usr/lib/systemd/system/smb.service; enabled; vendor preset: disabled)
  Active: active (running) since Sat 2021-10-09 13:00:43 MSK; 30s ago
    Docs: man:smbd(8)
          man:samba(7)
          man:smb.conf(5)
  Main PID: 2169 (smbd)
    Status: "smbd: ready to serve connections..."
     Tasks: 4 (limit: 5967)
   Memory: 7.2M
  CGroup: /system.slice/smb.service
         └─2169 /usr/sbin/smbd --foreground --no-process-group
```

После установки включим и запустим сервис, который называется smb. А на файрволе добавим сервис samba.

```
sudo systemctl enable --now smb
sudo firewall-cmd --add-service=samba --permanent
sudo firewall-cmd --reload
```

Обратите внимание, почти тоже самое мы делали для NFS. На самом деле это стандартные действия – каждый раз, когда вы устанавливаете какую-то программу-сервер, вам нужно добавить её в автозапуск, прописать на файрволе и запустить. На других дистрибутивах зачастую сервис автоматом запускается и добавляется в автозапуск, а файрвол не установлен по умолчанию. Но делать это вручную всё таки лучше, это помогает запомнить и немного учит дисциплине.

smb.conf

```
[user@centos8:~]$
[user@rhel8 ~]$ cat /etc/samba/
lmhosts          smb.conf          smb.conf.example
[user@rhel8 ~]$ cat /etc/samba/smb.conf
# See smb.conf.example for a more detailed config file or
# read the smb.conf manpage.
# Run 'testparm' to verify the config is correct after
# you modified it.

[global]
    workgroup = SAMBA
    security = user

    passdb backend = tdbsam

    printing = cups
    printcap name = cups
    load printers = yes
    cups options = raw
```

Основные настройки лежат в директории /etc/samba в файле smb.conf.

```
cat /etc/samba/smb.conf
```

В отличии от NFS здесь настроек много, но всё это знать наизусть не нужно. Основные параметры разберём сегодня, а некоторые вещи оставим на потом.

```
[user@rhel8 ~]$ cat /etc/samba/smb.conf.example
# This is the main Samba configuration file. For detailed information about the
# options listed here, refer to the smb.conf(5) manual page. Samba has a huge
# number of configurable options, most of which are not shown in this example.
#
# The Samba Wiki contains a lot of step-by-step guides installing, configuring,
# and using Samba:
# https://wiki.samba.org/index.php/User_Documentation
#
# In this file, lines starting with a semicolon (;) or a hash (#) are
# comments and are ignored. This file uses hashes to denote commentary and
# semicolons for parts of the file you may wish to configure.
#
# NOTE: Run the "testparm" command after modifying this file to check for basic
# syntax errors.
#
#-----
# Security-Enhanced Linux (SELinux) Notes:
#
# Turn the samba_domain_controller Boolean on to allow a Samba PDC to use the
# useradd and groupadd family of binaries. Run the following command as the
# root user to turn this Boolean on:
```

В соседнем файле smb.conf.example есть объяснение всех параметров, которые указаны в основном файле. Ну и в мане по smb.conf можно найти более детальную информацию. Стоит упомянуть, что

samby можно тесно связать с винтовой сетью, привязав её к домену или даже сделать из неё контроллер домена. Грубо говоря, в компаниях все компьютеры объединяют в одну сеть, называемую доменом. В этой сети есть специальные сервера, называемые «контроллером домена», которые могут централизовано управлять всеми компьютерами компании - хранить учётки пользователей, настраивать компьютеры, выдавать доступы и т.д. и т.п. И не мало настроек samby связаны с этим функционалом, который нам сейчас не нужен. Поэтому мы и пропустим некоторые параметры, сейчас samba интересует нас только как файловый сервер.

```

GNU nano 2.9.8          /etc/samba/smb.conf

# See smb.conf.example for a more detailed config file or
# read the smb.conf manpage.
# Run 'testparm' to verify the config is correct after
# you modified it.

[global]
    workgroup = SAMBA
    security = user

    passdb backend = tdbsam

    printing = cups
    printcap name = cups
    load printers = yes
    cups options = raw

[homes]
    comment = Home Directories
    valid users = %S, %D%w%S

```

Перейдём в smb.conf. Файл состоит из секций и параметров. Секции указаны в квадратных скобках. В секции [global] указаны параметры, относящиеся к самой самбе, а также дефолтные параметры для всех остальных секций. К примеру, параметр security можно указывать только в global. Остальные секции считаются «шарами», т.е. там указаны конкретные ресурсы, которыми делятся - директории и принтеры. Скажем, если в секции global указать параметр read only, он будет по умолчанию во всех секциях, если конечно в них не указано другое значение. Кстати, файл регистронезависимый, т.е можно писать и большими буквами, и маленькими. Пройдёмся по параметрам секции [global].

- Workgroup - это рабочая группа - упрощённый вариант домена, когда нет централизованного управления, но компьютеры должны друг с другом делиться файлами, принтерами и т.п. Не то, чтобы компьютеры без этого не общаются, но те, кто входят в одну группу, могут находить друг друга в одной сети, видеть, кто чем делится и всё такое. Но workgroupы нацелены на домашнее использование или на компании где меньше 20 компьютеров и сейчас они практически не используются. Но чтобы это полноценно работало, надо ещё включать сервис nmb.
- security - в зависимости от того, является ли samba самостоятельным файловым сервером, или частью домена, или работает с Керберосом, пользователи по разному подтверждают доступ. Если это самостоятельный файловый сервер, значение будет user.
- passdb backend - указывает на механизм хранения аккаунтов. В зависимости от значения используется файл или LDAP сервер.

Дальше пара параметров относящиеся к печати. Мы это сегодня разбирать не будем, поэтому я просто сотру эти строки.

```

user@centos8:~          user@rhel8:-
GNU nano 2.9.8           /etc/samba/smb.conf

[homes]
comment = Home Directories
valid users = %S, %D%w%S
browseable = No
read only = No
inherit acls = Yes

[printers]
comment = All Printers
path = /var/tmp
printable = Yes
create mask = 0600
browseable = No

[print$]

```

Далее идут две специальные секции - homes и printers. Всё что касается принтеров оставим только секции global и homes.

```

user@centos8:~          user@rhel8:-
GNU nano 2.9.8           /etc/samba/smb.conf

# See smb.conf.example for a more detailed config file or
# read the smb.conf manpage.
# Run 'testparm' to verify the config is correct after
# you modified it.

[global]
workgroup = SAMBA
security = user

passdb backend = tdb

[homes]
comment = Home Directories
valid users = %S, %D%w%S
browseable = No
read only = No
inherit acls = Yes

```

Как можно догадаться, здесь указаны домашние директории пользователей. Т.е. те кто будет заходить на SMB сервер, будут видеть свои домашние директории.

- comment - это просто комментарий к шаре, виден для клиентов.
- valid users - пользователи, которым разрешено подключаться к этой шаре. Здесь можно указывать

пользователей и группы. Если этого параметра нет, то любой пользователь сможет подключаться. homes - это особая секция. Когда кто-то подключается к файловому серверу, вся эта секция homes копируется для него, но уже с его логином. Это происходит в памяти, а не в этом файле. Условно, если подключился user1, то появится секция [user1] со всеми параметрами секции homes. И здесь %S - это переменная, указывающая на название секции. Т.е. для user1 valid users будет user1, соответственно у него будет разрешение зайти в эту шару. %D%w%S - тоже указывает на юзера, но с учётом его Workgroup-пы.

- browsable - SMB позволяет увидеть список доступных шар, как это показывал showmounts для NFS. Если стоит No, то при подключении нужно вписывать имя шары. Если стоит Yes, то можно просто увидеть эту шару в списке доступных. Но, как я говорил, homes это особая шара - хотя здесь стоит нет, т.е. все домашние директории не видны, но когда кто-то логиниться, лично для него значение меняется на yes и он видит только свою домашнюю директорию.
- read only - доступна ли шара только для чтения. Кстати, у read only есть антоним writeable - т.е. можно написать либо read only = No, либо writeable = Yes, разницы никакой.
- inherit acls - наследовать ли дефолтные acl-ы от директорий.

В теме о правах мы говорили, что стандартных UNIX-овых прав не всегда хватает, особенно это заметно на файловых серверах, где множество пользователей и групп. acl-ы решают эту проблему - мы можем выдать с помощью команды setfacl права для других пользователей и групп.

```
user@centos8:~$ mkdir testdir
user@centos8 ~]$ setfacl -m u:centosuser:rw testdir
user@centos8 ~]$ getfacl testdir
# file: testdir
# owner: user
# group: user
user::rwx
user:centosuser:rwx
group::rwx
mask::rwx
other::r-x

user@centos8 ~]$ touch testdir/testfile
user@centos8 ~]$ getfacl testdir/testfile
# file: testdir/testfile
# owner: user
# group: user
user::rw-
group::rw-
other::r--


user@centos8 ~]$
```

Давайте, для примера, создадим директорию testdir и дадим пользователю centosuser rw права на эту

директорию с помощью setfacl.

```
setfacl -m u:centosuser:rw testdir
getfacl testdir
```

Посмотрим - getfacl - как видно, права сработали. Но что, если создать файл в этой директории? Будет ли он иметь такие же права? Унаследует ли он acl-ы?

```
touch testdir/testfile
getfacl testdir/testfile
```

Как видите, нет. И это очень неудобно, вы создали директорию, дали некоторым пользователям доступ к ней. И когда пользователи будут создавать свои файлы внутри, у новых файлов права будут стандартные и придётся вечно подправлять права на новые файлы.

```
[user@centos8 ~]$ setfacl -dm user:centosuser:rw testdir
[user@centos8 ~]$ touch testdir/file2
[user@centos8 ~]$ getfacl testdir/ testdir/file2
# file: testdir/
# owner: user
# group: user
user::rwx
user:centosuser:rwx-
group::rwx
mask::rwx
other::r-x
default:user::rwx
default:user:centosuser:rwx-
default:group::rwx
default:mask::rwx
default:other::r-x

# file: testdir/file2
# owner: user
# group: user
user::rw-
user:centosuser:rwx-
group::rwx
mask::rw-
other::r-- #effective:rwx-

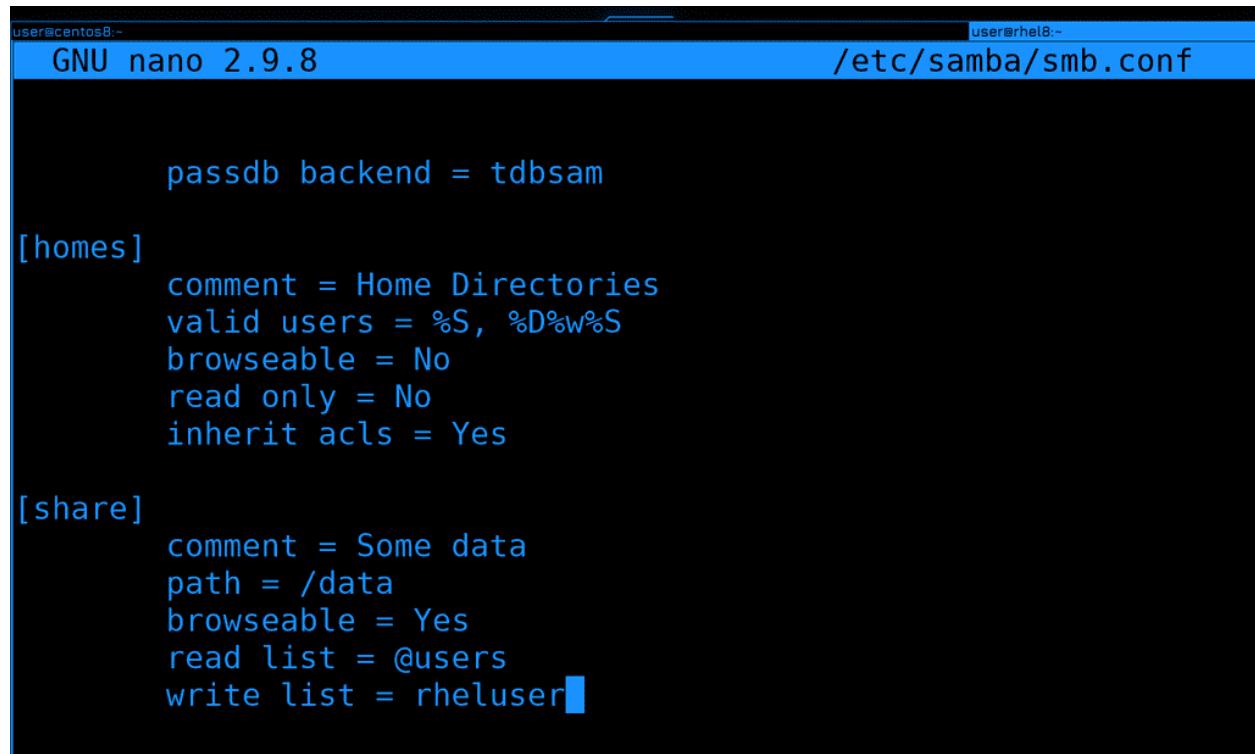
```

Поэтому на директории можно поставить acl-ы как дефолтные, с ключом -d.

```
setfacl -dm user:centosuser:rw testdir
touch testdir/file2
getfacl testdir testdir/file2
```

Это не повлияет на существующие файлы, но все новые файлы будут иметь такие же acl-ы, как и директория.

Так вот, inherit acls как раз об этом - будут ли наследоваться права директории на новые файлы внутри.



```

passdb backend = tdbsam

[homes]
comment = Home Directories
valid users = %S, %D%w%S
browseable = No
read only = No
inherit acls = Yes

[share]
comment = Some data
path = /data
browseable = Yes
read list = @users
write list = rheluser

```

Давайте создадим свою шару.

```
[share]
comment = Some data
path = /data
browseable = Yes
read list = @users
write list = rheluser
```

Назовём её share и пропишем какой-нибудь комментарий. Нужно указать путь к директории, которая будет раздаваться - пусть будет /data, для этого параметр path. И пусть она будет видима - browseable = Yes.

Table 6.1: Share-level Access Options

Option	Parameters	Function	Default	Scope
admin users	string (list of usernames)	Specifies a list of users who can perform operations as root.	None	Share
valid users	string (list of usernames)	Specifies a list of users that can connect to a share.	None	Share
invalid users	string (list of usernames)	Specifies a list of users that will be denied access to a share.	None	Share
read list	string (list of usernames)	Specifies a list of users that have read-only access to a writable share.	None	Share
write list	string (list of usernames)	Specifies a list of users that have read-write access to a read-only share.	None	Share
max connections	numerical	Indicates the maximum number of connections for a share at a given time.	0	Share
guest only (only guest)	boolean	Specifies that this share allows only guest access.	no	Share
guest account	string (name of account)	Names the Unix account that will be used for guest access.	nobody	Share

Пропишем два списка - read list и write list, соответственно, кому можно только смотреть и кому можно изменять содержимое. write list от valid users отличается тем, что даёт права на изменение даже если сама шара read only. Перед группой ставится собачка - @users. То есть это просто локальные пользователи и группы.

```
[user@centos8:-] $ sudo nano /etc/samba/smb.conf
[user@rhel8 ~]$ testparm /etc/samba/smb.conf
Load smb config files from /etc/samba/smb.conf
Loaded services file OK.
Weak crypto is allowed
Server role: ROLE_STANDALONE

Press enter to see a dump of your service definitions

# Global parameters
[global]
    security = USER
    workgroup = SAMBA
    idmap config * : backend = tdb

[homes]
    browseable = No
    comment = Home Directories
    inherit acls = Yes
    read only = No
    valid users = %S %D%w%S

[share]
```

Сохраним и выйдем. Чтобы проверить синтаксис, мало ли мы где-то опечатались, есть утилита testparm:

```
testparm /etc/samba/smb.conf
```

Хотя указывать на дефолтный конфиг не обязательно, но полезно знать, чтобы проверять файлы с другим именем и директорией. Как видите, всё ок, и после нажатия Enter мы видим наши шары.

```
[user@centos8:-] $ [user@rhel8 ~]$ sudo systemctl restart smb
[user@rhel8 ~]$
```

После чего не забудем перезапустить сервис.

```
sudo systemctl restart smb
```

```
[user@centos8:~] [user@rhel8 ~]$ sudo smbpasswd -a rheluser
New SMB password:
Retype new SMB password:
Added user rheluser.
[user@rhel8 ~]$ sudo smbpasswd rheluser
New SMB password:
Retype new SMB password:
[user@rhel8 ~]$ sudo smbpasswd -a user
New SMB password:
Retype new SMB password:
Added user user.
[user@rhel8 ~]$
```

Ещё один важный пункт. Хотя сейчас используются системные пользователи, но пароли для samba надо создавать отдельно, с помощью команды smbpasswd:

```
sudo smbpasswd -a rheluser
```

При первом добавлении пароля надо указывать ключ -а, а для смены этот ключ не нужен. Системный пароль и пароль от самбы могут отличаться, но у пользователя должен быть системный пароль, хотя в качестве shell-а у него может быть nologin.

samba-client

```
[user@centos8:~] [user@rhel8 ~]$ sudo dnf install samba-client
Last metadata expiration check: 0:00:56 ago on Sat 09 Oct 2021 05:54:31 PM +04.
Package samba-client-4.10.4-101.el8_1.x86_64 is already installed.
Dependencies resolved.
=====
 Package           Architecture Version       Repository
 =====
```

Теперь пойдём на Centos и попробуем посмотреть список доступных шар. Для этого нужен samba клиент:

```
sudo dnf install samba-client
```

```

user@centos8:~]$ smbclient -L rhel8
Enter SAMBA\user's password:

      Sharename          Type        Comment
      -----          ----        -----
      share            Disk        Some data
      IPC$             IPC        IPC Service (Samba 4.13.3)
      user            Disk        Home Directories
SMB1 disabled -- no workgroup available
[user@centos8 ~]$ smbclient -L rhel8 -U rheluser
Enter SAMBA\rheluser's password:

      Sharename          Type        Comment
      -----          ----        -----
      share            Disk        Some data
      IPC$             IPC        IPC Service (Samba 4.13.3)
      rheluser         Disk        Home Directories
SMB1 disabled -- no workgroup available
[user@centos8 ~]$ █

```

И так, используем smbclient с ключом -L чтобы посмотреть список доступных шар на rhel-e:

```
smbclient -L rhel8
```

Сервер у нас запросит пароль, но мы можем проигнорировать и просто нажать enter, тогда мы увидим те шары, на которых browseable = Yes. Сейчас мы видим share, который мы создали, user потому что соответствует нашему пользователю и IPC. IPC это не файловая шара, а функционал SMB, который используется для сетевого обнаружения и подключения. Т.е. его просто игнорируем.

Теперь попробуем ещё раз посмотреть список доступных шар, но уже используя пользователя rheluser, для этого используем ключ -U - user:

```
smbclient -L rhel8 -U rheluser
```

Как видите, теперь видим домашнюю директорию пользователя rheluser. Теоретически, там может быть ещё множество невидимых директорий, для которых параметр browseable имеет значение no, но при монтировании мы можем их указать и получить доступ.

```
[user@centos8 ~]$ smbclient //rhel8/share -U rheluser
Enter SAMBA\rheluser's password:
Try "help" to get a list of possible commands.
smb: \> ?
?          allinfo      altname      archive      backup
blocksize   cancel       case_sensitive cd        chmod
chown      close        del          deltree     dir
du         echo         exit         get         getfacl
geteas     hardlink    help         history     iosize
lcd        link         lock         lowercase  ls
l          mask         md          mget       mkdir
more       mput        newer        notify     open
posix      posix_encrypt posix_open   posix_mkdir posix_rmdir
posix_unlink  posix_whoami  print      prompt    put
pwd        q            queue      quit      readlink
rd         recurse     reget      rename    reput
rm         rmdir       showacls   setea     setmode
scopy      stat        symlink   tar      tarmode
timeout    translate   unlock    volume   vuid
wdel      logon       listconnect  showconnect tcon
tdis       tid         utimes   logoff   ..
!
```

smbclient чем-то напоминает sftp и может подключаться к командной строке, в которой доступны определённые команды для работы с файлами:

```
smbclient //rhel8/share -U rheluser
```

Обратите внимание, как указывается путь к шаре - два слэша адрес сервера, затем один слэш и адрес шары. Т.е. с этой стороны мы не знаем, где именно на сервере находится директория. Здесь, после адреса сервера, мы указываем название секции в smb.conf, ну или своего пользователя.

```
[user@centos8 ~]$ sudo dnf install cifs-utils
Last metadata expiration check: 0:00:05 ago on Sat 09 Oct 2021 06:44:47 PM +04.
Package cifs-utils-6.8-2.el8.x86_64 is already installed.
Dependencies resolved.
=====
== Package           Architecture     Version      Repository      Size ==
=====
Upgrading:
cifs-utils          x86_64          6.8-3.el8    BaseOS        96 k
Transaction Summary
=====
Upgrade 1 Package
Total download size: 96 k
Is this ok [y/N]: ■
```

Для того чтобы примонтировать шару, нужен пакет cifs-utils:

```
sudo dnf install cifs-utils
```

```
[user@centos8 ~]$ sudo mount //rhel8/share /mnt -o username=rheluser
Password for rheluser@//rhel8/share: *
mount error(13): Permission denied
Refer to the mount.cifs(8) manual page (e.g. man mount.cifs) and kernel log messages (dmesg)
[user@centos8 ~]$ ssh root@rhel8 setenforce 0
root@rhel8's password:
[user@centos8 ~]$ sudo mount //rhel8/share /mnt -o username=rheluser
Password for rheluser@//rhel8/share: *
[user@centos8 ~]$ df -h /mnt/
Filesystem      Size  Used Avail Use% Mounted on
//rhel8/share    11G  2.3G  8.0G  23% /mnt
[user@centos8 ~]$
```

После установки попробуем примонтировать:

```
sudo mount //rhel8/share /mnt -o username=rheluser
```

Обратите внимание, что при монтировании нужно указывать опцию `username`, чтобы монтировать от имени определённого пользователя на сервере, для которого мы делали `smbpasswd`.

SElinux

На этот раз монтирование не сработало, мол, отказано в правах. Тут дело в selinux-е и в `smb.conf.example` об этом говорится, на сервере надо вешать на директорию определённый контекст, чтобы самба мог раздавать эту директорию. Давайте убедимся, что дело в selinux - просто временно переведём его в permissive режим:

```
ssh root@rhel8 setenforce 0
sudo mount //rhel8/share /mnt -o username=rheluser
df -h /mnt
```

Как видите, теперь всё примонтировалось. По началу вы можете сталкиваться с ситуациями, когда вы всё правильно настроили, но ничего не работает, какие-то ошибки прав. В такие моменты вы можете попробовать сделать `setenforce 0` и проверить, не в selinux-е ли дело.

Окей, предположим, мы поняли, что дело в SELinux. Как это исправить?

```
[user@centos8 ~]$ sudo setenforce 1
[user@centos8 ~]$
```

Давайте, сперва, заново включим selinux:

```
sudo setenforce 1
```

```
[user@rhel8 ~]$ cat /etc/samba/smb.conf.example | grep _t
# with samba_share_t so that SELinux allows Samba to read and write to it. Do
# not label system directories, such as /etc/ and /home/, with samba_share_t, as
# chcon -t samba_share_t /path/to/directory
[user@rhel8 ~]$ sudo semanage fcontext -at samba_share_t "/data(/.*)?"
[sudo] password for user:
[user@rhel8 ~]$ sudo restorecon -rv /data
Relabeled /data/file1 from unconfined_u:object_r:default_t:s0 to unconfined_u:object_r:samba_share_t
:s0
Relabeled /data/file2 from unconfined_u:object_r:default_t:s0 to unconfined_u:object_r:samba_share_t
:s0
Relabeled /data/file3 from unconfined_u:object_r:default_t:s0 to unconfined_u:object_r:samba_share_t
:s0
Relabeled /data/rheluser from svstem_u:object_r:default_t:s0 to svstem_u:object_r:samba_share_t:s0
[user@centos8 ~]$ sudo mount //rhel8/share /mnt -o username=rheluser
Password for rheluser@//rhel8/share: *
[user@centos8 ~]$ █
```

Легче всего подсмотреть в конфиге. И так, мы знаем, что надо подправить контекст для директории. Надо просто найти нужный тег. Контекст с тегом заканчивается на `_t`, и как-то связан с самбой. Если поискать grep-ом в конфиг файле, можно найти нужное:

```
cat /etc/samba/smb.conf.example | grep _t
```

Конечно, это сработает не всегда, но вот сейчас сработало. Не факт что это тот самый нужный тег, но если зайти в файл и прочесть, можно в этом убедиться. Правда мы помним, что `chcon` не лучший вариант, так как он может сбрасываться, поэтому возьмём контекст и воспользуемся командой `semanage`:

```
sudo semanage fcontext -at samba_share_t "/data(/.*)?"
sudo restorecon -rv /data
```

Ну и когда речь идёт о директориях, не забываем прописать к ним специальное выражение, его всегда можно подглядеть в `man semanage-fcontext`. После изменений попробуем примонтировать:

```
sudo mount //rhel8/share /mnt -o username=rheluser
```

Теперь всё работает.

Права

Поговорим о правах. Если в NFS работают UNIX-ые права, то в SMB права как бы раздваиваются на локальные и серверные. Я примонтировал шару от имени серверного пользователя `rheluser`, а значит с точки зрения сервера, все мои действия с файлами будут происходить от имени этого пользователя. При этом на локальной машине действуют локальные права. Я примонтировал в `/mnt` от пута, значит только он сможет изменять файлы. Звучит сложно, но пара примеров всё объяснят.

```
[user@centos8:/mnt] $ ls -l /mnt/
total 4
-rwxr-xr-x. 1 root root 6 Oct  2 18:40 file1
-rwxr-xr-x. 1 root root 0 Oct  2 18:35 file2
-rwxr-xr-x. 1 root root 0 Oct  2 18:35 file3
drwxr-xr-x. 2 root root 0 Oct  2 18:42 rheluser
[user@centos8 mnt]$ ls -ld /mnt/
drwxr-xr-x. 2 root root 0 Oct  2 18:41 /mnt/
[user@centos8 mnt]$ ssh root@rhel8 ls -l /data/
root@rhel8's password:
total 4
-rw-r--r--. 1 user      root      6 Oct  2 17:40 file1
-rw-r--r--. 1 root      root      0 Oct  2 17:35 file2
-rw-r--r--. 1 root      root      0 Oct  2 17:35 file3
drwx-----. 2 rheluser  rheluser 95 Oct  2 17:42 rheluser
[user@centos8 mnt]$ █
```

Для начала проверим права файлов в /mnt, ну и права самой директории:

```
ls -l /mnt
ls -ld /mnt
```

Обратите внимание, что всё что здесь есть принадлежит пользователю root, остальные пользователи могут только просматривать. Если же посмотреть как обстоят дела на сервере:

```
ssh root@rhel8 ls -l /data/
```

можно заметить, что у каких-то файлов владелец root, а у каких-то user и rheluser. И даже сами права отличаются - на centos-e у файлов есть execute права, т.е. права 755, а на rhel-e нет - там 644.

```
[user@centos8:~]$ ls -l /mnt/
total 4
-rwxr-xr-x. 1 root root 6 Oct  2 18:40 file1
-rwxr-xr-x. 1 root root 0 Oct  2 18:35 file2
-rwxr-xr-x. 1 root root 0 Oct  2 18:35 file3
drwxr-xr-x. 2 root root 0 Oct  2 18:42 rheluser
[user@centos8 ~]$ echo test | sudo tee /mnt/file2
tee: /mnt/file2: Permission denied
test
[user@centos8 ~]$ cat /mnt/file2
[user@centos8 ~]$ █
```

Видите, что у пользователя root есть право изменять file2? Давайте попробуем это сделать:

```
echo test | sudo tee /mnt/file2  
cat /mnt/file2
```

Но у нас ошибка прав - ничего не изменилось. Почему? Потому что у пользователя rheluser на сервере нет прав изменять этот файл. Т.е. какой-бы пользователь не был со стороны клиента, какие-бы права у него не были локально, права на сервере важнее.

```
[user@centos8:~] $ ls -l /mnt/rheluser/  
total 0  
-rwxr-xr-x. 1 root root 0 Oct  2 18:42 file  
[user@centos8 ~]$ touch /mnt/rheluser/file2  
touch: cannot touch '/mnt/rheluser/file2': Permission denied  
[user@centos8 ~]$ sudo !!  
sudo touch /mnt/rheluser/file2  
[sudo] password for user:  
[user@centos8 ~]$ ls -l /mnt/rheluser/  
total 0  
-rwxr-xr-x. 1 root root 0 Oct  2 18:42 file  
-rwxr-xr-x. 1 root root 0 Oct 11 22:39 file2  
[user@centos8 ~]$ █
```

Теперь о правах со стороны клиента. У пользователя rheluser есть права на директорию /mnt/rheluser. Но со стороны клиента права есть только у рута. Если я попытаюсь что-то создать от имени обычного пользователя:

```
touch /mnt/rheluser/file2
```

ничего не получится, мне откажет в правах клиентская система. А вот root уже может работать в этой директории:

```
sudo touch /mnt/rheluser/file2
```

Т.е. сначала проверяются локальные права на стороне клиента. Если нет - то отказывается в правах. Если же у локального пользователя с правами всё окей, на сервере проверяются права того, чьим именем примонтировали эту шару, т.е. пользователя rheluser.

```
[user@centos8 ~]$ ls -l /mnt/rheluser/file  
-rwxr-xr-x. 1 root root 0 Oct  2 18:42 /mnt/rheluser/file  
[user@centos8 ~]$ sudo chown user:user /mnt/rheluser/file  
[sudo] password for user:  
[user@centos8 ~]$ ls -l /mnt/rheluser/file  
-rwxr-xr-x. 1 root root 0 Oct  2 18:42 /mnt/rheluser/file  
[user@centos8 ~]$ sudo chmod 777 /mnt/rheluser/file  
[user@centos8 ~]$ ls -l /mnt/rheluser/file  
-rwxr-xr-x. 1 root root 0 Oct  2 18:42 /mnt/rheluser/file  
[user@centos8 ~]$ mount | grep share  
//rhel8/share on /mnt type cifs (rw,relatime,vers=3.1.1,cache=strict,username=rheluser,uid=0,noforce  
uid,gid=0,noforcegid,addr=192.168.31.205,file mode=0755,dir mode=0755,soft,nounix,serverino,mapposix  
,rsize=4194304,wsize=4194304,bsize=1048576,echo_interval=60,actimeo=1)  
[user@centos8 ~]$ █
```

А давайте попробуем на клиенте выдать права. Скажем, поменять владельца или выставить максимальные права:

```
sudo chown user:user /mnt/rheluser/file
sudo chmod 777 /mnt/rheluser/file
ls -l /mnt/rheluser/file
```

Команды вроде бы срабатывают, но в результате ничего, ничего не меняется. Со стороны клиента нельзя менять права стандартными утилитами, не получится chmod-ом или chown-ом поменять что-то локально или удалённо. Права в SMB больше рассчитаны на виндовые, поэтому всякие uid-ы и gid-ы не работают так, как мы привыкли.

Но это не значит, что может работать только root. То что касается прав на стороне клиента настраивается в опциях монтирования. Давайте их посмотрим:

```
mount | grep share
```

Во-первых, обратите внимание на опцию `posix`. В старой версии протокола SMB версии 1 можно задать, чтобы цеплялись юниксовые права между сервером и клиентом, как это сделано в NFS. Опция `posix` отключает использование этого. Ну и в современных версиях протокола это всё равно не будет работать. Кстати, сейчас в основном используется SMB версии 3.1, хотя где-то используются и более старые версии.

Во-вторых, взгляните на опции `file_mode` и `dir_mode`, где указаны права 755. Помните, у файлов в `/mnt` были права 755? Т.е. какие будут права локально выдаются именно этими параметрами при монтировании. Допустим, можно указать при монтировании, чтобы `file_mode` был 777, и тогда на стороне клиента любой пользователь сможет работать со всеми файлами в шаре, потому что в `/mnt` будет `rwxrwxrwx`. Конечно, не стоит забывать, что и на сервере есть проверка прав, от чьего имени примонтирована шара.

Ну и в-третьих, на что особенно стоит обратить внимание, так это на опции `uid` и `gid`, которые равны 0. Они как раз указывают, на какого локального пользователя нужно цеплять эту шару. Т.е., то что на локальной системе у всех файлов в `/mnt` владелец и группа root - благодаря этим опциям. И да, мы можем поставить другие значения. Давайте это и сделаем.

```
[user@centos8 ~]$ sudo umount /mnt
[sudo] password for user:
[user@centos8 ~]$ sudo mount //rhel8/share /mnt -o username=rheluser,uid=1000,gid=1000,file_mode=0640,dir_mode=0750
Password for rheluser@//rhel8/share: *
[user@centos8 ~]$ ls -l /mnt/
total 4
-rw-r----- 1 user user 6 Oct  2 18:40 file1
-rw-r----- 1 user user 0 Oct  2 18:35 file2
-rw-r----- 1 user user 0 Oct  2 18:35 file3
drwxr-x--- 2 user user 0 Oct 12 00:11 rheluser
[user@centos8 ~]$ touch /mnt/file4
touch: cannot touch '/mnt/file4': Permission denied
[user@centos8 ~]$ touch /mnt/rheluser/file4
[user@centos8 ~]$ ls -l /mnt/rheluser/
total 0
-rw-r----- 1 user user 0 Oct  2 18:42 file
-rw-r----- 1 user user 0 Oct 11 22:39 file2
-rw-r----- 1 user user 0 Oct 12 00:14 file4
[user@centos8 ~]$ █
```

Для начала отмонтируем шару:

```
sudo umount /mnt
```

Затем попробуем примонтировать с новыми опциями:

```
sudo mount //rhel8/share /mnt -o username=rheluser,uid=1000,gid=1000,file_mode=0640,dir_
mode=0750
```

Давайте прочтём, что мы написали:

- username - пользователь на сервере, кем мы логинимся и чьи файлы нам нужны.
- uid и gid - 1000 - это наш локальный пользователь и группа на Centos, у кого будут права на примонтированные файлы и директории.
- file_mode и dir_mode - с какими правами будут эти примонтированные файлы и директории.

Теперь проверим:

```
ls -l /mnt
touch /mnt/file4
touch /mnt/rheluser/file4
ls -l /mnt/rheluser
```

И так, теперь файлы в /mnt принадлежат пользователю user и он сможет с ними работать без sudo прав. Но в /mnt мы по прежнему не можем создавать файлы - опять же, потому что у пользователя rheluser на сервере нет таких прав в директории /data. Зато в директории /mnt/rheluser мы можем спокойно создать файл нашим пользователем, и в целом делать всё, что можно пользователю rheluser.

fstab

Но, конечно же, сам пользователь user без root прав не смог бы примонтировать директорию. Нам стоит добавить эту запись в fstab, чтобы монтировалось автоматом, при включении. Но тут есть нюанс - в NFS пароль нам не нужен был, а с SMB то нужно указывать пароль пользователя rheluser. Да, конечно, мы можем прописать пароль в опциях монтирования в fstab просто опцией password, но ведь fstab доступен для чтения всем пользователям. Не хорошо светить паролем. Поэтому надо кое-что подготовить.

The screenshot shows two terminal sessions. The left session is on a CentOS 8 host (user@centos8) where the user creates a directory structure and a secret file. The right session is on another CentOS 8 host (user@rhel8) where the user checks the permissions of the mounted share. The terminal output is as follows:

```
[user@centos8 ~]$ ls -ld ~/
drwx----- 25 user user 4096 Oct 12 00:48 /home/user/
[user@centos8 ~]$ touch .secret
[user@centos8 ~]$ nano .secret
username=rheluser
password=1
[user@centos8 ~]$ mkdir documents
[user@centos8 ~]$ 

user@rhel8:~$ ls -ld /home/user/
drwx----- 25 user user 4096 Oct 12 00:48 /home/user/
```

Для начала создадим файл, где будут храниться логин и пароль пользователя rheluser. Тут зависит от того, должен ли user на локальной системе знать пароль или нет. Если знает - то можем создать в его домашней директории, как раз она доступна для чтения только им и root-ом:

```
ls -ld ~/
```

Если же пользователь не должен знать пароль, то можно где-нибудь в системе, скажем, в /etc, создать файл, доступный для чтения только root-ом. Для примера, я создам файл у самого пользователя в домашней директории и пропишу в нём логин и пароль.

```
nano .secret
```

```
username=rheluser
password=1
```

И здесь пишем - username=rheluser, а на следующей строчке password= и указываем пароль. С echo заполнять файл не стоит, так как это может остаться в истории. Ну и заодно создадим точку монтирования:

```
mkdir documents
```

```
[user@centos8 ~]$ sudo nano /etc/fstab
[sudo] password for user:
[user@centos8 ~]$ tail -1 /etc/fstab
//rhel8/share /home/user/documents cifs credentials=/home/user/.secret,uid=1000,gid=1000,nofail 0
0
[user@centos8 ~]$ sudo umount /mnt
[user@centos8 ~]$ sudo mount -a
[user@centos8 ~]$ ls -l documents/
total 4
-rwxr-xr-x. 1 user user 6 Oct  2 18:40 file1
-rwxr-xr-x. 1 user user 0 Oct  2 18:35 file2
-rwxr-xr-x. 1 user user 0 Oct  2 18:35 file3
drwxr-xr-x. 2 user user 0 Oct 12 00:14 rheluser
[user@centos8 ~]$ mount | grep docu
//rhel8/share on /home/user/documents type cifs (rw,relatime,vers=3.1.1,cache=strict,username=rheluser,uid=1000,forceuid,gid=1000,forcegid,addr=192.168.31.205,file_mode=0755,dir_mode=0755,soft,nounix,serverino,mapposix,rsize=4194304,wsize=4194304,bsize=1048576,echo_interval=60,actimeo=1)
[user@centos8 ~]$
```

Теперь идём в fstab и прописываем строчку:

```
//rhel8/share /home/user/documents cifs credentials=/home/user/.secret,uid=1000,gid=1000,
nofail 0 0
```

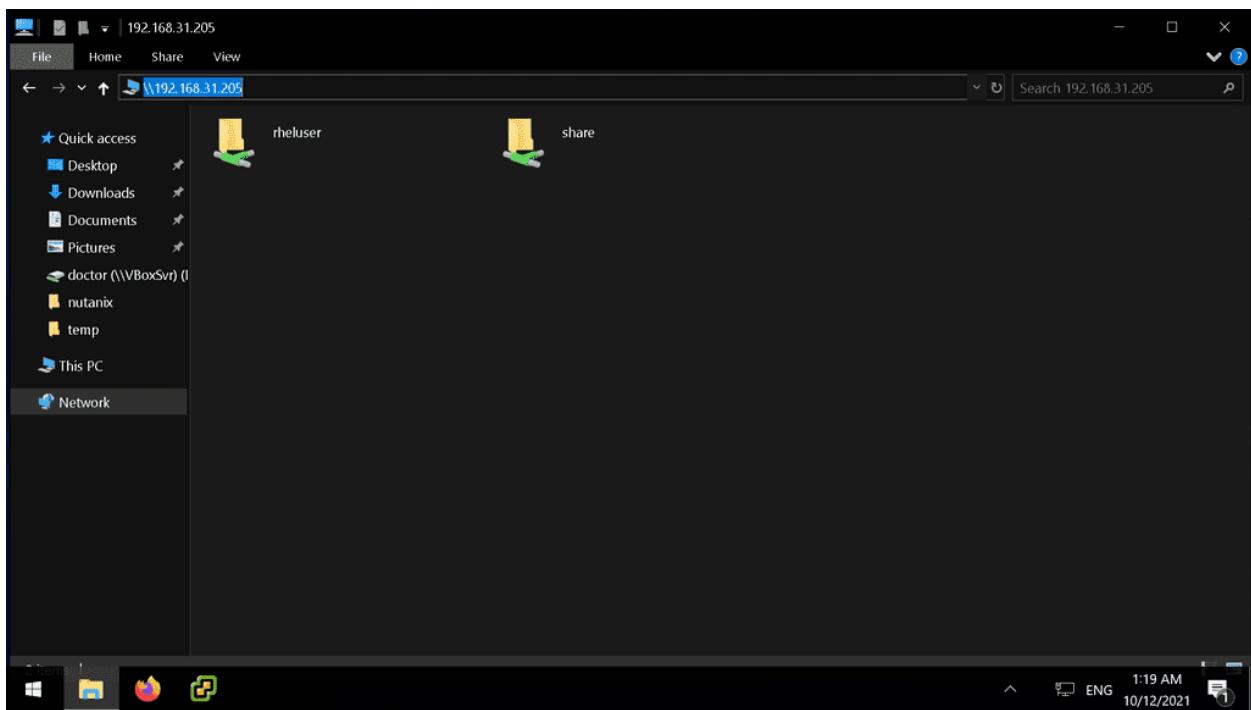
И так, в качестве типа файловой системы указываем cifs. Вместо username и password указываем опцию credentials, в которой указываем полный путь к файлу с логином и паролем. Также добавим опцию uid и gid, чтобы монтировалось от имени юзера, а не рута. file_mode и dir_mode не обязательно добавлять, всё равно в эту директорию никто кроме юзера не сможет зайти из-за прав на его домашнюю директорию. Ну и указываем nofail, чтобы проблемы в сети не приводили к проблемам с запуском системы.

Теперь отмонтируем и убедимся, что мы нигде не ошиблись:

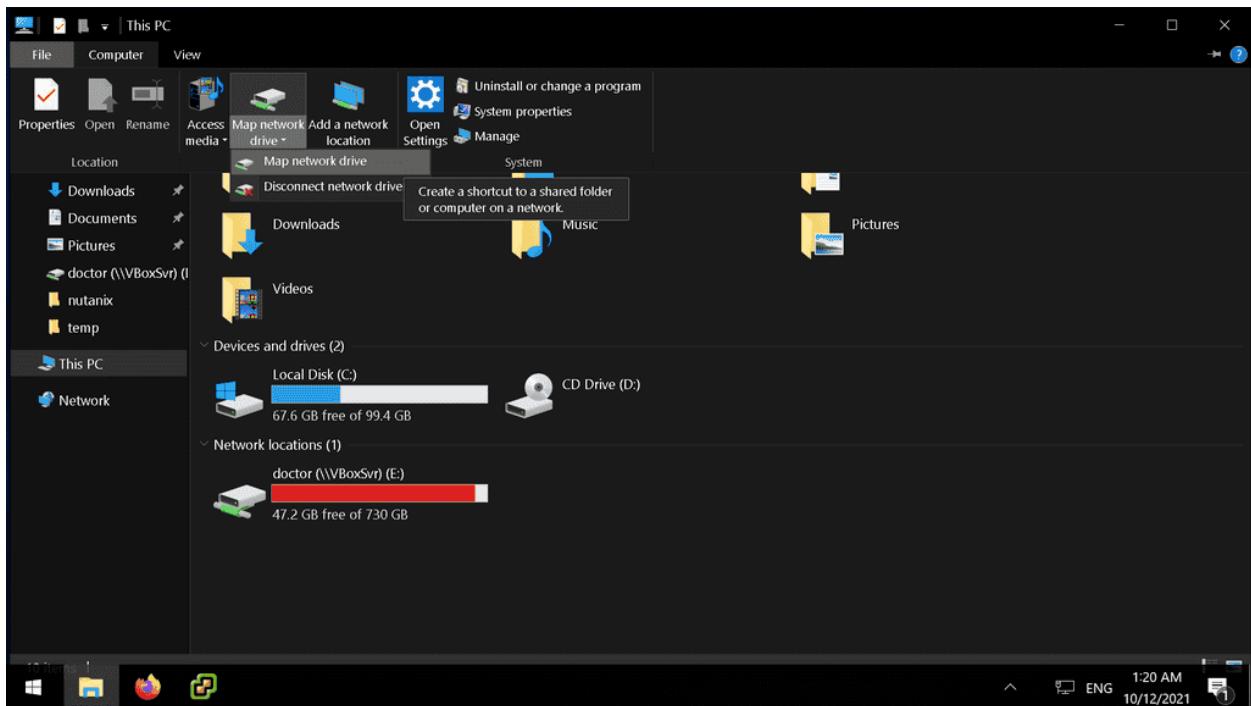
```
sudo umount /mnt
sudo mount -a
ls -l documents
mount | grep docum
```

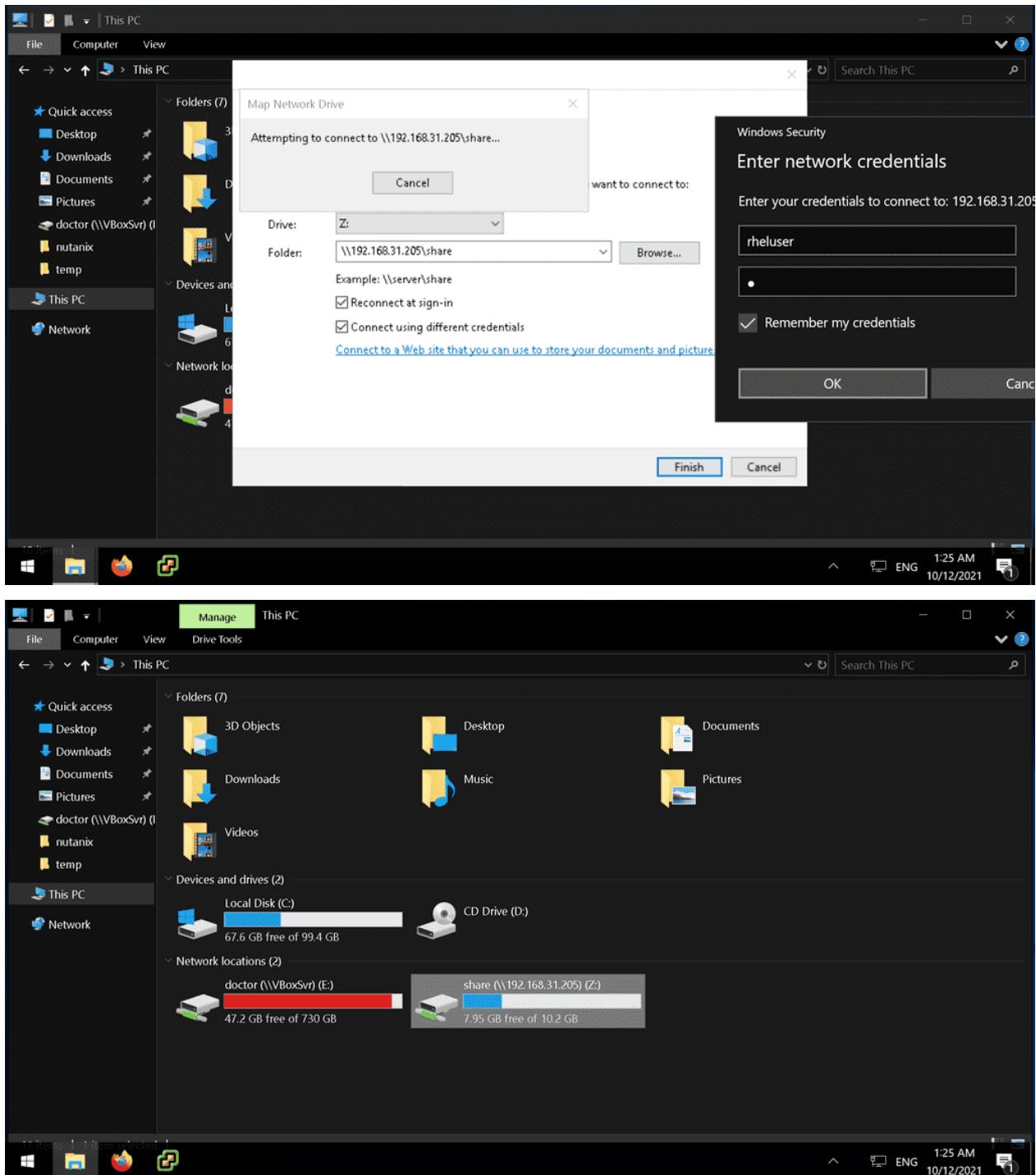
Я вижу файлы в documents и вижу, что шара примонтирована куда нужно. Значит всё работает.

Windows



Ну и напоследок стоит показать, как монтируовать эту шару на Windows. Тут уже вместо прямых слэшей используются обратные. Нужно открыть проводник и в поле адреса указать ip адрес сервера. Можно и по имени, но я это не настраивал. Плюс, можно сразу указать полный адрес шары, а можно указать только адрес сервера, и тогда мы после логина увидим все доступные сервисы.





Ну а чтобы шара была видна всегда в окне «Мой компьютер», нажимаем сверху Computer - Map Network Drive - и указываем адрес шары - \\192.168.31.205\share. Так как логин пользователя Windows отличается от rheluser, ставим галочку «использовать другой логин». После чего вводим логин и пароль. Теперь всё готово.

Давайте подведём итоги. Сегодня мы с вами разобрали протокол SMB и в частности программу samba. Это может показаться сложной темой, здесь относительно большой функционал и много опций, особенно сравнивая с предыдущими нашими темами. Не говоря уже о том, что мы многое оставили на потом. Плюс различия с точки зрения прав могут запутывать. Но вам не нужно знать все опции наизусть,

всегда есть маны и интернет под рукой. Попрактикуйтесь пару раз и всё вам станет понятнее.

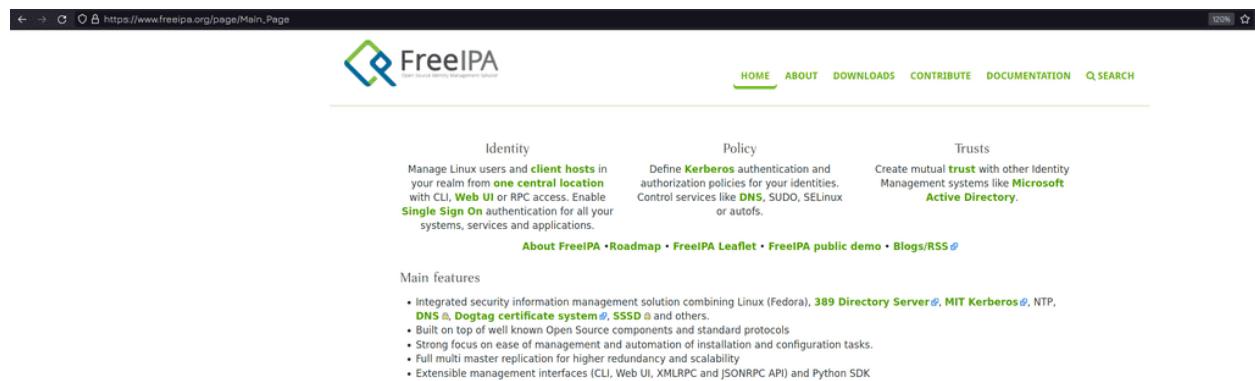
2.59 59. Автоматическое монтирование - **Autofs**

2.59.1 59. Автоматическое монтирование - **Autofs**

В некоторых средних и крупных компаниях, там где много людей, привязка данных к компьютеру усложняет работу. Пользователи часто меняют места, кто-то переходит с одного отдела в другой, кто-то хочет сесть подальше от кондиционера, да и в целом есть такая практика периодически менять места. И постоянно из-за этого таскать и переподключать компьютеры пользователей дело неблагодарное. Большинство данных хранится на файловых серверах, у пользователей в домашних директориях не так много файлов. Поэтому можно просто цеплять домашние директории по сети и, тогда, за какой-бы компьютер человек не сел, везде будут его настройки и файлы. Даже когда компьютер выйдет из строя, вам достаточно будет подключить новый компьютер, а не возиться с восстановлением данных. Как пример, это также удобно в современных учебных заведениях, где в каждом классе есть компьютеры - тогда учителя и студенты могут спокойно ходить из одного кабинета в другой и работать со своими файлами.

Это всё очень здорово, но, нужно учитывать, что выход из строя сети или сервера приведёт к полной остановке работы людей, так как их домашние директории перестанут быть доступны. Есть, конечно, более надёжный механизм - репликация, т.е. как это делает rsync. Вместо того, чтобы монтировать директории по сети, они будут просто синхронизироваться с сервером. Правда из-за этого нужно будет тратить место и на сервере, и на компьютерах. Но сегодня мы разберём именно первый случай, без репликации, т.е. просто автомонтирование домашних директорий. Стоит отметить, что данную тему я рассматриваю из-за экзамена, который требует умения настраивать только клиентскую часть. Я хотел бы рассмотреть и настройку сервера, но это требует знания многих тем, которые не касаются экзамена и которые мы пока не прошли. Когда-нибудь мы до них доберёмся, но пока сконцентрируемся на этой теме.

FreeIPA



The screenshot shows the official website for FreeIPA at https://www.freeipa.org/page/Main_Page. The page features a green header bar with the FreeIPA logo and navigation links for HOME, ABOUT, DOWNLOADS, CONTRIBUTE, DOCUMENTATION, and SEARCH. Below the header, there are three main sections: Identity, Policy, and Trusts. The Identity section describes managing Linux users and client hosts from one central location using CLI, Web UI, or RPC access. It also mentions Single Sign On authentication. The Policy section explains Kerberos authentication and authorization policies. The Trusts section discusses creating mutual trust with other identity management systems like Microsoft Active Directory. At the bottom of the page, there's a footer with links to About FreeIPA, Roadmap, FreeIPA Leaflet, FreeIPA public demo, and Blogs/RSS.

Вкратце, как это работает: есть специальный сервер - FreeIPA, на котором работают и связаны между собой несколько протоколов - Kerberos, LDAP, NFS и многое другое. Kerberos отвечает за безопасную аутентификацию пользователей по сети, а LDAP за хранение информации о том, что кому разрешено, т.е. за авторизацию. Сам FreeIPA - сервер, который объединяет всё это и даёт удобный веб интерфейс для управления.

The FreeIPA server is running on a **Red Hat's OpenStack** instance, on the latest stable **Fedora**. The server controls a **DNS** domain named **demo1.freeipa.org** and the corresponding **Kerberos** realm **DEM01.FREEIPA.ORG**. The server itself is named **ipa.demo1.freeipa.org**.

Sandbox
The FreeIPA demo server is just a sandbox and is **wiped clean every day** at 05:00 UTC. In case you had a testing FreeIPA client enrolled, the easiest recovery is to uninstall your client (`ipa-client-install --uninstall` and **Install it again**). In case you are interested in a more persistent testing environment, try **downloading** a FreeIPA server on a personal virtual machine.

Users
The FreeIPA domain is configured with the following users (the password is **Secret123** for all of them):

- **admin**: The FreeIPA main administrator account, has all the privileges
- **helpdesk**: A regular user with the **helpdesk** role allowing it to modify other users or change their group membership
- **employee**: A regular user with no special privileges
- **manager**: A regular user, set as *manager* of the *employee* user

Groups
To allow testing group-based authentication we created additional groups in addition to the default FreeIPA ones:

- **employees**: contains users *employee* and *manager*
- **managers**: contains user *manager*

Наша виртуалка будет клиентом. Так как у нас своего сервера нет, поэтому мы воспользуемся [демо сервером](#), который доступен для всех. По ссылке есть краткое руководство, как настроить клиент. В качестве клиента я буду использовать Centos. Советую предварительно сделать снапшот виртуалки, а потом, после всех тестов, его восстановить.

```
[user@centos8 ~]$ sudo hostnamectl set-hostname centos8.demo1.freeipa.org
[user@centos8 ~]$ ping ipa.demo1.freeipa.org
PING ipa.demo1.freeipa.org (52.57.162.88) 56(84) bytes of data.
^C
--- ipa.demo1.freeipa.org ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 26ms

[user@centos8 ~]$ nc -zv ipa.demo1.freeipa.org 389
Ncat: Version 7.70 ( https://nmap.org/ncat )
Ncat: Connected to 52.57.162.88:389.
Ncat: 0 bytes sent, 0 bytes received in 0.33 seconds.
[user@centos8 ~]$ █
```

Для начала стоит изменить hostname нашей виртуалки, чтобы он содержал домен:

```
sudo hostnamectl set-hostname centos8.demo1.freeipa.org
```

Затем убедимся, что наша виртуалка видит сервер:

```
ping ipa.demo1.freeipa.org
```

ping на имя ipa.demo1.freeipa.org показывает IP адрес, значит DNS резолвил, правда этот сервер не пингуется. Но это не значит, что сервер недоступен. Можем проверить доступность какого-нибудь сервиса, допустим, LDAP, который работает на 389 порту:

```
nc -zv ipa.demo1.freeipa.org 389
```

И в выводе мы видим Connected, значит соединение прошло успешно. Значит с сетью проблем нет.

The screenshot shows a web browser window with the URL <https://www.freeipa.org/page/Demo>. The page title is "Quick Start". It contains instructions for testing the Web UI and a few easy steps. Step 1: Install required packages, shown as a command: # yum install freeipa-client. Step 2: Configure FreeIPA client, shown as a command: # ipa-client-install --domain demo1.freeipa.org -p admin -w Secret123. The output of this command shows discovery was successful, and it prints the host name, realm, DNS domain, IPA server, and base DN.

Далее, следуя гайду, нужно выполнить две команды - установить freeipa-client и настроить его.

```
[user@centos8 ~]$ sudo dnf install freeipa-client -y
Last metadata expiration check: 0:01:48 ago on Sat 16 Oct 2021 10:16:34 AM +04.
Dependencies resolved.
=====
| Package           | Arch      | Version            | Repository | Size |
|=====             | =====     | =====              | =====      | ===== |
Installing:
| ipa-client        | x86_64   | 4.9.2-3.module_el8.4.0+780+c53619b9 | AppStream | 278 k |
```

Установим freeipa-client с помощью dnf:

```
sudo dnf install freeipa-client -y
```

```
[user@centos8 ~]$ sudo ipa-client-install --domain demo1.freeipa.org -p admin -w Secret123
This program will set up IPA client.
Version 4.9.2

Sudo version 1.8.25p1
Configure options: --build=x86_64-redhat-linux-gnu --host=x86_64-redhat-linux-gnu --program-prefix= --disable-dependency-tracking --prefix=/usr --exec-prefix=/usr --bindir=/usr/bin --sbindir=/usr/sbin --sysconfdir=/etc --datadir=/usr/share --includedir=/usr/include --libdir=/usr/lib64 --libexecdir=/usr/libexec --localstatedir=/var --sharedstatedir=/var/lib --mandir=/usr/share/man --infodir=/usr/share/info --prefix=/usr --sbindir=/usr/sbin --libdir=/usr/lib64 --docdir=/usr/share/doc/ipa-client-4.9.2
```

После чего запустим настройку:

```
sudo ipa-client-install --domain demo1.freeipa.org -p admin -w Secret123
```

```
Sudoers I/O plugin version 1.8.25p1
Discovery was successful!
Do you want to configure chrony with NTP server or pool address? [no]:
Client hostname: centos8.demo1.freeipa.org
Realm: DEMO1.FREEIPA.ORG
DNS Domain: demo1.freeipa.org
IPA Server: ipa.demo1.freeipa.org
BaseDN: dc=demo1,dc=freeipa,dc=org

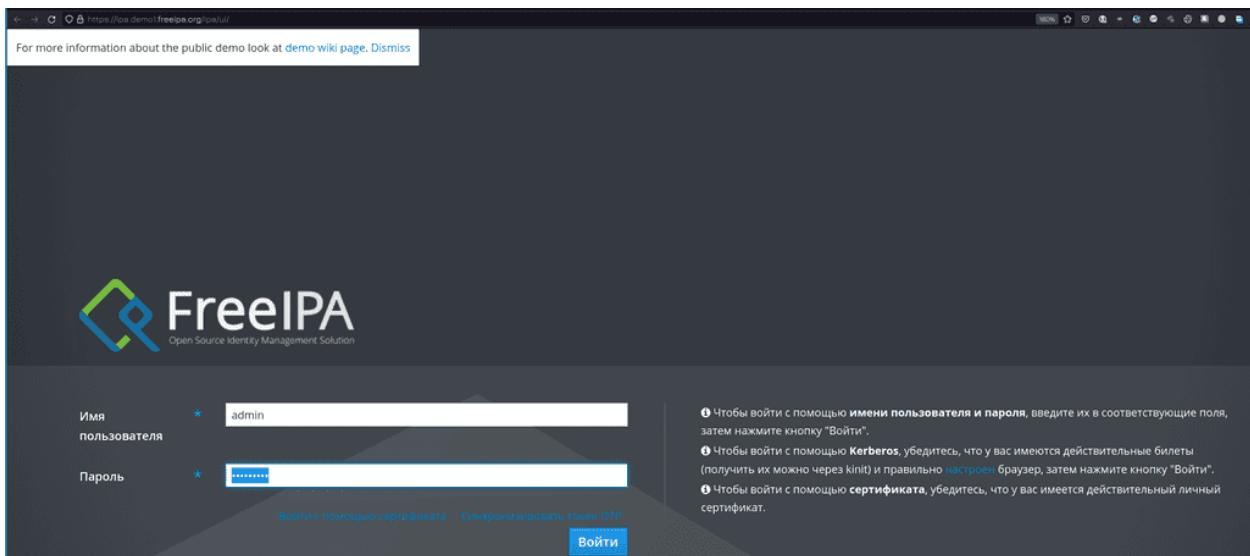
Continue to configure the system with these values? [no]: yes
Synchronizing time
No SRV records of NTP servers found and no NTP server or pool address was provided.
Using default chrony configuration.
Attempting to sync time with chronyc.
```

В какой-то момент программа спросит, хотим ли мы указать ntp сервер - отвечаем **но**, т.е. просто нажимаем enter. Далее она соберёт всю информацию и уточнит, уверены ли мы, что хотим продолжить

с такими параметрами. Пишем yes и нажимаем enter.

```
Configured /etc/openldap/ldap.conf
Configured /etc/ssh/ssh_config
Configured /etc/ssh/sshd_config
Configuring demo1.freeipa.org as NIS domain.
Client configuration complete.
The ipa-client-install command was successful
[user@centos8 ~]$
```

В итоге мы увидим сообщение, что команда завершилась успешно.



Дальше можем перейти по ссылке ipa.demo1.freeipa.org, где находится веб интерфейс для управления сервером FreeIPA. Вводим логин admin и пароль Secret123. Можете потыкать интерфейс и посмотреть, что к чему, но, пожалуйста, уважайте труд других людей и ничего не портьте.

Имя узла	Описание	Зарегистрировано
centos8.demo1.freeipa.org		Да
ipa.demo1.freeipa.org		Да

Также зайдите во вкладку «Узлы», где вы увидите в списке адрес вашего клиента.

```
[user@centos8 ~]$ id admin
uid=1006(admin) gid=100(users) groups=100(users)
[user@centos8 ~]$ id helpdesk
uid=613800004(helpdesk) gid=613800004(helpdesk) groups=613800004(helpdesk)
[user@centos8 ~]$ id employee
uid=613800003(employee) gid=613800003(employee) groups=613800003(employee),613800005(employees)
[user@centos8 ~]$ id manager
uid=613800001(manager) gid=613800001(manager) groups=613800001(manager),613800005(employees),613800006(managers)
[user@centos8 ~]$ id admin@demo1.freeipa.org
uid=613800000(admin) gid=613800000(admins) groups=613800000(admins)
[user@centos8 ~]$ █
```

Теперь вернёмся к нашей виртуалке. И так, на сервере freeipa есть 4 настроенных пользователя, которые могут логиниться на любом клиенте. ipa клиент так настроил нашу систему, что мы теперь можем видеть этих юзеров почти как локальных:

```
id admin
id helpdesk
id employee
id manager
```

Обратите внимание, что у всех этих пользователей uid-ы имеют длинное значение, а у admin нет. Т.е. у нас есть локальный юзер admin и он пересекается с пользователем admin на сервере, т.е. это не тот же самый юзер. Чтобы увидеть серверного, придётся добавить собачку и домен:

```
id admin@demo1.freeipa.org
```

```
[user@centos8 ~]$ su - employee
Password:
su: warning: cannot change directory to /home/employee: No such file or directory
[employee@centos8 user]$ id
uid=613800003(employee) gid=613800003(employee) groups=613800003(employee),613800005(employees)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[employee@centos8 user]$ █
```

Можем даже попробовать залогиниться этими пользователями, у всех пароли Secret123.

```
su - employee
```

Как видите, я залогинился, но пользователь остался без домашней директории, так как нет такой директории.

autofs

И теперь мы можем приступить к сути нашего урока: нужно, чтобы при логине пользователя или когда кто-то обращается к этой директории, она автоматом монтировалась. Можно было бы прописать где-нибудь в fstab, но если у нас много пользователей, директорию каждого прописать нереально. Да и от того, что они просто так висят примонтированными, тоже ничего хорошего, шары постоянно обращаются к серверу, используют сеть и оперативку. Поэтому и нужно автомонтирование.

```
[employee@centos8 user]$ logout
[user@centos8 ~]$ sudo systemctl list-units --type automount
[sudo] password for user:
UNIT                         LOAD   ACTIVE SUB   DESCRIPTION
proc-sys-fs-binfmt_misc.automount loaded active waiting Arbitrary Executable File Formats File System Automount
LOAD  = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB   = The low-level unit activation state, values depend on unit type.

1 loaded units listed. Pass --all to see loaded but inactive units, too.
To show all installed unit files use 'systemctl list-unit-files'.
[user@centos8 ~]$
```

Мы, кстати, говорили об автомонтировании, когда разбирали systemd. У него есть юниты automount, которые этим и занимаются:

```
sudo systemctl list-units --type automount
```

Разве что autofs, который мы будем сегодня разбирать, немного пофункциональнее.

```
[user@centos8 ~]$ sudo dnf install autofs
Last metadata expiration check: 0:38:47 ago on Sat 16 Oct 2021 10:16:34 AM +04.
Package autofs-1:5.1.4-48.el8_4.1.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[user@centos8 ~]$ sudo systemctl status autofs
● autofs.service - Automounts filesystems on demand
  Loaded: loaded (/usr/lib/systemd/system/autofs.service; disabled; vendor preset: disabled)
    Active: inactive (dead)
[user@centos8 ~]$ sudo systemctl enable --now autofs
Created symlink /etc/systemd/system/multi-user.target.wants/autofs.service → /usr/lib/systemd/system/autofs.service.
[user@centos8 ~]$
```

И так, сперва стоит установить пакет autofs:

```
sudo dnf install autofs
```

Пакет уже установлен. Давайте проверим, может и сервис работает?

```
systemctl status autofs
```

Нет, сервис выключен. Давайте добавим его в автозапуск и запустим:

```
sudo systemctl enable --now autofs
```

```
[user@centos8 ~]$ ls /etc/auto*
/etc/autofs.conf           /etc/auto.master    /etc/auto.net
/etc/auto_ldap_auth.conf   /etc/auto.misc     /etc/auto.smb

/etc/auto.master.d:
[user@centos8 ~]$ cat /etc/auto.master
#
# Sample auto.master file
# This is a 'master' automounter map and it has the following format:
# mount-point [map-type[,format]:]map [options]
# For details of the format look at auto.master(5).
#
/misc   /etc/auto.misc
#
# NOTE: mounts done from a hosts map will be mounted with the
#       "nosuid" and "nodev" options unless the "suid" and "dev"
#       options are explicitly given.
#
/net   -hosts
```

Настройка autofs поделена по файлам:

```
ls /etc/auto*
```

Все они лежат в директории /etc. Основной файл - /etc/auto.master, он уже ссылается на другие файлы:

```
cat /etc/auto.master
```

autofs очень гибкий и его можно настроить как для сетевых файловых систем, так и для локальных. Сейчас нас интересует только NFS.

И так, в auto.master мы должны указать, для какой локальной директории какой шаблон autoofs-а. Домашняя директория пользователя employee ссылается на /home/employee.

```
GNU nano 2.9.8 /etc/auto.master

# built as part of autofs or installed from another package,
# uncomment this line to use the fedfs program map to access
# your fedfs mounts.
#/nfs4 /usr/sbin/fedfs-map-nfs4 nobind
#
# Include central master map if it can be found using
# nsswitch sources.
#
# Note that if there are entries for /net or /misc (as
# above) in the included master map any keys that are the
# same will not be seen as the first read key seen takes
# precedence.
#
#+auto.master

/home /etc/auto.home
```

Открываем /etc/auto.master и добавляем строчку:

```
sudo nano /etc/auto.master
```

```
/home /etc/auto.home
```

Здесь мы указали, что всё что касается монтирования внутри директории home - указано в файле /etc/auto.home.

```
GNU nano 2.9.8 /etc/auto.home

employee -rw ipa.demo1.freeipa.org:/home/employee
```

Теперь открываем файл /etc/auto.home и указываем:

```
sudo nano /etc/auto.home
```

```
employee -rw ipa.demo1.freeipa.org:/home/employee
```

Давайте поясню. В auto.master мы указали, что /etc/auto.home касается директории /home, а значит в самом /etc/auto.home не нужно писать полный путь - /home/employee, достаточно просто прописать employee, и так понятно, что это внутри /home. -tw мы указали, чтобы шара монтировалась в режиме read write. Ну и дальше мы указываем адрес nfs сервера и путь к самой директории пользователя на сервере. Теперь, если кто-то зайдёт в директорию /home/employee в ней примонтируется NFS шара.

```
[user@centos8 ~]$ sudo systemctl restart autofs
[sudo] password for user:
[user@centos8 ~]$ su - employee
Password:
Last login: Sat Oct 16 11:52:35 +04 2021 on pts/0
su: warning: cannot change directory to /home/employee: No such file or directory
[employee@centos8 user]$ nc -zv ipa.demo1.freeipa.org 2049
Ncat: Version 7.70 ( https://nmap.org/ncat )
Ncat: Connection timed out.
[employee@centos8 user]$ █
```

Сохраняем файл и выходим. Дело остаётся за малым - перезапустить сервис autofs и залогиниться пользователем employee. Но мы опять видим ошибку - нет такой директории. Наша конфигурация должна была сработать, но вот оказывается на демо сервере не включен NFS:

```
nc -zv ipa.demo1.freeipa.org 2049
```

Как видите, Connection timed out, т.е. мы не можем подключиться. Что ж, придётся изменить планы и поднять NFS сервер у себя.

```
[user@centos8 ~]$ sudo nano /etc/exports
[sudo] password for user:
[user@centos8 ~]$ cat /etc/exports
/data/employee *(rw)
[user@centos8 ~]$ sudo mkdir /data/employee
mkdir: cannot create directory '/data/employee': No such file or directory
[user@centos8 ~]$ sudo mkdir -p /data/employee
[user@centos8 ~]$ sudo chown employee:employee /data/employee/
[user@centos8 ~]$ sudo touch /data/employee/myfile
[user@centos8 ~]$ sudo systemctl enable --now nfs-server
Created symlink /etc/systemd/system/multi-user.target.wants/nfs-server.service → /usr/lib/systemd/system/nfs-server.service.
[user@centos8 ~]$ ls -l /data/employee/
total 0
-rw-r--r-- 1 root root 0 Oct 16 13:03 myfile
[user@centos8 ~]$ ls -ld /data/employee/
drwxr-xr-x. 2 employee employee 20 Oct 16 13:03 /data/employee/
[user@centos8 ~]$ showmount centos8.demo1.freeipa.org
Hosts on centos8.demo1.freeipa.org:
[user@centos8 ~]$ showmount -e centos8.demo1.freeipa.org
Export list for centos8.demo1.freeipa.org:
/data/employee *
[user@centos8 ~]$ █
```

Мы это уже делали, поэтому не буду вдаваться в детали. Просто поднимем nfs сервер, пропишем в exports директорию /data/employee, создадим такую директорию, сделаем её владельцем этого пользователя, создадим внутри файл, включим и запустим сервис, ну и убедимся что видим шару с помощью showmount.

```
sudo nano /etc/exports
```

```
/data/employee *(rw)
```

```
sudo mkdir -p /data/employee
sudo chown employee:employee /data/employee
sudo touch /data/employee/myfile
sudo systemctl enable --now nfs-server
showmount -e centos8.demo1.freeipa.org
```

```
[user@centos8 ~]$ sudo nano /etc/auto.home
[user@centos8 ~]$ cat /etc/auto.home
employee -rw centos8.demo1.freeipa.org:/data/employee
[user@centos8 ~]$ sudo systemctl restart autofs
[user@centos8 ~]$ ls /home/employee
myfile
[user@centos8 ~]$ df -h /home/
Filesystem      Size  Used Avail Use% Mounted on
/etc/auto.home    0     0     0   -  /home
[user@centos8 ~]$ su - employee
Password:
Last login: Sat Oct 16 13:10:50 +04 2021 on pts/0
[employee@centos8 ~]$ pwd
/home/employee
[employee@centos8 ~]$ touch file3
[employee@centos8 ~]$ ls /data/employee/
file3 myfile
[employee@centos8 ~]$ █
```

После чего надо поправить путь к NFS в auto.home:

```
sudo nano /etc/auto.home
```

```
employee -rw centos8.demo1.freeipa.org:/data/employee
```

Теперь у нас получается, что при логине пользователя employee будет создаваться директория /home/employee, куда будет монтироваться /data/employee с нашего же NFS сервера. Т.е. не обязательно, чтобы NFS сервер находился на самом FreeIPA сервере, в идеале это должен быть отдельный сервер. И хотя с точки зрения FreeIPA наш NFS сервер настроен не до конца, но наша сегодняшняя тема - это автомонтирование, поэтому продолжим.

После изменений перезапустим сервис autofs:

```
sudo systemctl restart autofs
```

И теперь стоит нам попытаться обратиться к директории /home/employee - автоматически она промонтируется:

```
ls /home/employee
```

При этом сама директория /home находится на файловой системе auto.home:

```
df -h /home
```

Из-за этого у всех наших локальных пользователей домашние директории остались под этой файловой системой, поэтому недоступны. Если мы используем только удалённых пользователей - это не страшно. Если есть и локальные, и удалённые, то лучше внутри /home создать поддиректорию, скажем, /home/remote.

Теперь попробуем залогиниться нашим пользователем:

```
su - employee
```

На этот раз никакой ошибки, теперь у пользователя есть домашняя директория. Мы можем создать файл и убедиться, что он появился на nfs сервере:

```
pwd  
touch file3  
ls /data/employee
```

Как видите, всё работает.

```
[employee@centos8 ~]$ logout  
[user@centos8 ~]$ su - manager  
Password:  
[su: warning: cannot change directory to /home/manager: No such file or directory  
[manager@centos8 user]$ logout  
[user@centos8 ~]$ █
```

Но то что мы настроили, работает только для одного пользователя. Как же нам сделать, чтобы это работало для всех, при этом, не прописывая их всех вручную?

```
[user@centos8 ~]$ sudo mkdir /data/{manager,helpdesk}  
[sudo] password for user:  
[user@centos8 ~]$ sudo chown manager:manager /data/manager/  
[user@centos8 ~]$ sudo chown helpdesk:helpdesk /data/helpdesk  
[user@centos8 ~]$ ls -ld /data/*  
drwxr-xr-x. 2 employee employee 54 Oct 16 13:14 /data/employee  
drwxr-xr-x. 2 helpdesk helpdesk 6 Oct 16 13:32 /data/helpdesk  
drwxr-xr-x. 2 manager manager 6 Oct 16 13:32 /data/manager  
[user@centos8 ~]$ █
```

Для начала создадим для них директории на nfs сервере и дадим соответствующие права:

```
sudo mkdir /data/{manager,helpdesk}  
sudo chown manager:manager /data/manager  
sudo chown helpdesk:helpdesk /data/helpdesk  
ls -ld /data/*
```

```
[user@centos8 ~]$ sudo nano /etc/exports
[user@centos8 ~]$ cat /etc/exports
/data/*(rw)
[user@centos8 ~]$ sudo exportfs -rv
exporting *:/data
[user@centos8 ~]$ showmount -e centos8.demo1.freeipa.org
Export list for centos8.demo1.freeipa.org:
/data *
[user@centos8 ~]$ █
```

После чего поправим NFS, чтобы он раздавал все директории внутри /data:

```
sudo nano /etc/exports
```

```
/data/*(rw)
```

Затем переэкспортируем шару:

```
sudo exportfs -rv
```

И убедимся, что всё сработало:

```
showmount -e centos8.demo1.freeipa.org
```

```
[user@centos8 ~]$ sudo nano /etc/auto.home
[user@centos8 ~]$ cat /etc/auto.home
* -rw centos8.demo1.freeipa.org:/data/&
[user@centos8 ~]$ sudo systemctl restart autofs
[user@centos8 ~]$ su - manager
Password:
Last login: Sat Oct 16 13:27:58 +04 2021 on pts/0
[manager@centos8 ~]$ touch myfile
[manager@centos8 ~]$ logout
[user@centos8 ~]$ su - helpdesk
Password:
[helpdesk@centos8 ~]$ touch myfile
[helpdesk@centos8 ~]$ logout
```

Ну и подправим сам auto.home:

```
sudo nano /etc/auto.home
```

```
* -rw centos8.demo1.freeipa.org:/data/&
```

Обратите внимание на написание. Вместо указания конкретных директорий, ставим звёздочку. А для NFS шары ставим амперсанд вместо определённой директории.

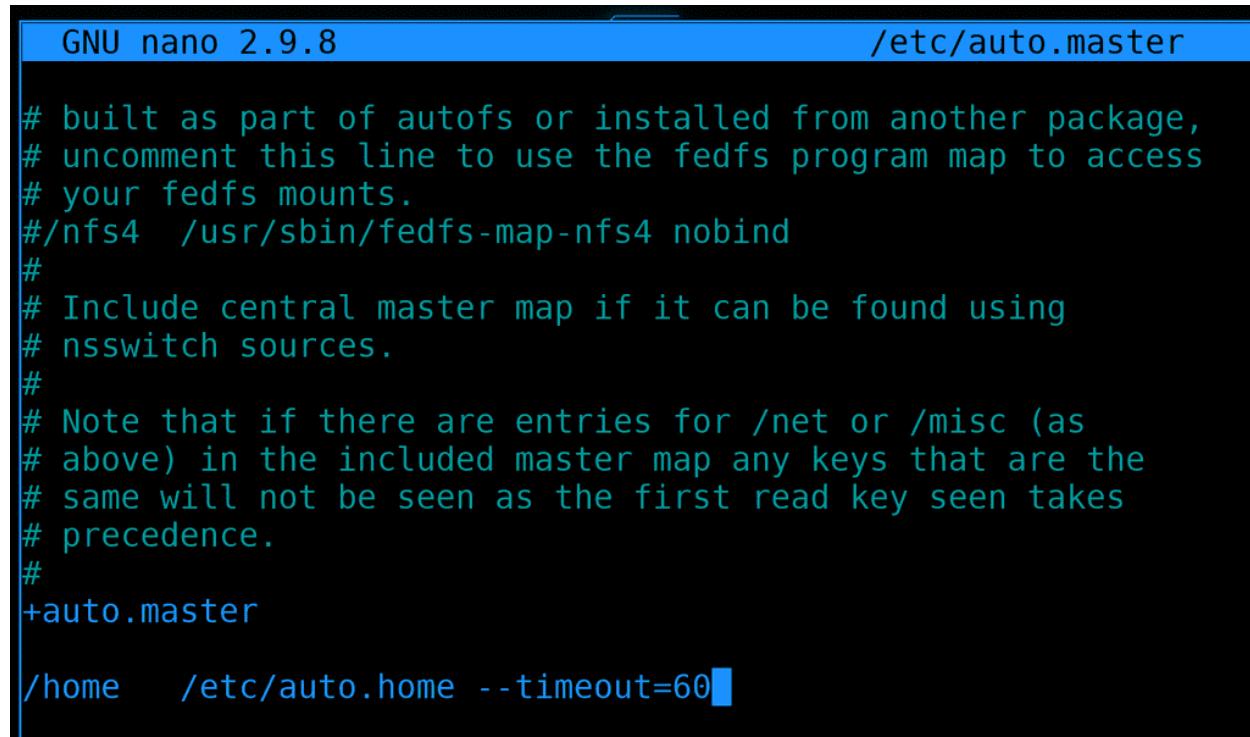
После изменений перезапустим сервис autoofs:

```
sudo systemctl restart autoofs
```

Ну и давайте проверим. Залогинимся пользователями и попробуем создать файлы:

```
su - manager
touch myfile
su - helpdesk
touch myfile
```

Всё работает.



```
GNU nano 2.9.8 /etc/auto.master

# built as part of autoofs or installed from another package,
# uncomment this line to use the fedfs program map to access
# your fedfs mounts.
#/nfs4 /usr/sbin/fedfs-map-nfs4 nobind
#
# Include central master map if it can be found using
# nsswitch sources.
#
# Note that if there are entries for /net or /misc (as
# above) in the included master map any keys that are the
# same will not be seen as the first read key seen takes
# precedence.
#
+auto.master

/home /etc/auto.home --timeout=60
```

Ну и напоследок, добавим timeout для autoofs. Заходим в /etc/auto.master и находим нашу строчку с auto.home:

```
sudo nano /etc/auto.master
```

В строчке с нашим auto.home в конце добавляем опцию --timeout:

```
/home /etc/auto.home --timeout=60
```

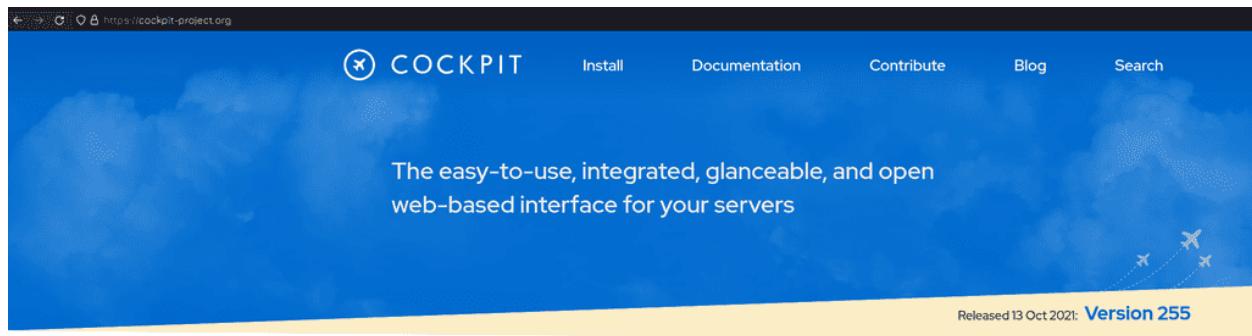
Если в течении указанного количества секунд никакой процесс не будет обращаться к этой директории, то она отмонтируется и не будет использовать ресурсы. Ну и не забудьте после изменений перезагрузить сервис.

Мы не разобрали весь функционал, но главное было научиться с этим работать. Оставшиеся параметры, то, как указывать SMB, локальные устройства и прочее - всё это можно найти в документации. Можете также посмотреть статью про autoofs на [арчвики](#), там тоже понятно объясняется. Единственное, там настройки находятся не в /etc/, а в /etc/autofs, так как там документация для другого дистрибутива.

Давайте подведём итоги. Сегодня мы подключили нашу виртуалку в качестве клиента к FreeIPA серверу, взяли оттуда пользователей, подняли NFS сервер и создали для этих пользователей домашние директории на нём. Дальше мы настроили autofs, чтобы при логине этих пользователей их домашние директории монтировались по сети. Всё это вместе может показаться сложным, но на самом деле это просто комплексно - пару простых действий на нескольких системах. Попрактикуйтесь и всё станет понятнее.

2.60 60. Веб-интерфейс - Cockpit

2.60.1 60. Веб-интерфейс - Cockpit



Introducing Cockpit

Cockpit is a web-based graphical interface for servers, intended for everyone, especially those who are:

- new to Linux
(including Windows admins)
- familiar with Linux
and want an easy, graphical way to administer servers
- expert admins
who mainly use other tools but want an overview on individual systems

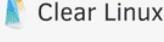
Thanks to Cockpit intentionally using system APIs and commands, a whole team of admins can manage a system in the way they prefer, including the command line and utilities right alongside Cockpit.

Как мы говорили, на серверах обычно не ставят графический интерфейс, так как всё можно сделать через командную строку. А графический интерфейс тянет за собой кучу пакетов, которые могут иметь уязвимости, ну и впустую использует ресурсы. Да и сервера стоят не на локальных машинах, т.е. если мы хотим графический интерфейс, нужно удалённо подключаться через какие-нибудь протоколы, к примеру, VNC. Но при задержках в сети подключение начинает подвисать, изображение задерживается и всё такое, что довольно неприятно.

Многие программы, используемые администраторами в компаниях, имеют веб-интерфейс - к примеру, гипервизоры, антивирусы, файрволы и т.д. и т.п. Веб-интерфейс лучше справляется с задержками в сети, не так сильно грузит сервер и имеет меньше зависимостей. Раньше как было - к примеру, для управления гипервизором была только графическая утилита, написанная под Windows. И пользователям других систем приходилось постоянно подключаться к какой-нибудь виртуалке с графическим интерфейсом, где стоит этот софт. К счастью, сейчас почти всем можно управлять через веб интерфейс.

Не исключение и Linux. Есть множество различных проектов, которые дают интерфейс и могут упростить настройку различных сервисов. Обычно это независимые проекты, но есть и исключение - проект [Cockpit](#), который спонсируется и поддерживается компанией Red Hat.

Installation & Setup

	Supported	Tested	Included	
 fedora	✓	✓	✓	View instructions
 Red Hat Enterprise Linux	✓	✓	✓	View instructions
 fedora COREOS	✓	✓	✓	View instructions
 CentOS	✓	✓	✓	View instructions
 debian	✓	✓	✓	View instructions
 ubuntu	✓	✓	✓	View instructions
 Clear Linux			✓	View instructions
 archlinux			✓	View instructions
 Tumbleweed			✓	View instructions

Хотя изначально Cockpit работал только на дистрибутивах RHEL, сейчас он работает на многих других, в том числе Убунту и Дебиан.

```
└─[doctor@tardis]─[~]
└─$ ssh rhel
X11 forwarding request failed on channel 0
Activate the web console with: systemctl enable --now cockpit.socket

This system is not registered to Red Hat Insights. See https://cloud.redhat.com/
To register this system, run: insights-client --register

Last login: Sat Oct 23 10:11:07 2021 from 192.168.31.227
[user@rhel8 ~]$ sudo firewall-cmd --list-services
[sudo] password for user:
cockpit dhcpcv6-client mountd nfs ntp rpc-bind samba ssh
[user@rhel8 ~]$
```

Мы часто встречали это название, когда логинились через SSH или смотрели firewall:

```
sudo firewall-cmd --list-services
```

Да, cockpit был прописан наряду с ssh.

```
[user@rhel8 ~]$ sudo systemctl enable --now cockpit.socket
Created symlink /etc/systemd/system/sockets.target.wants/cockpit.socket → /usr/lib/systemd/system/cockpit.socket.
[user@rhel8 ~]$ logout
Connection to 192.168.31.205 closed.
[doctor@tardis:~]
└─ $ ssh rhel
X11 forwarding request failed on channel 0
Web console: https://rhel8:9090/ or https://192.168.31.205:9090/

This system is not registered to Red Hat Insights. See https://cloud.redhat.com/
To register this system, run: insights-client --register

Last login: Sat Oct 23 10:19:01 2021 from 192.168.31.227
[user@rhel8 ~]$ sudo firewall-cmd --info-service=cockpit
[sudo] password for user:
cockpit
  ports: 9090/tcp
  protocols:
  source-ports:
    .
```

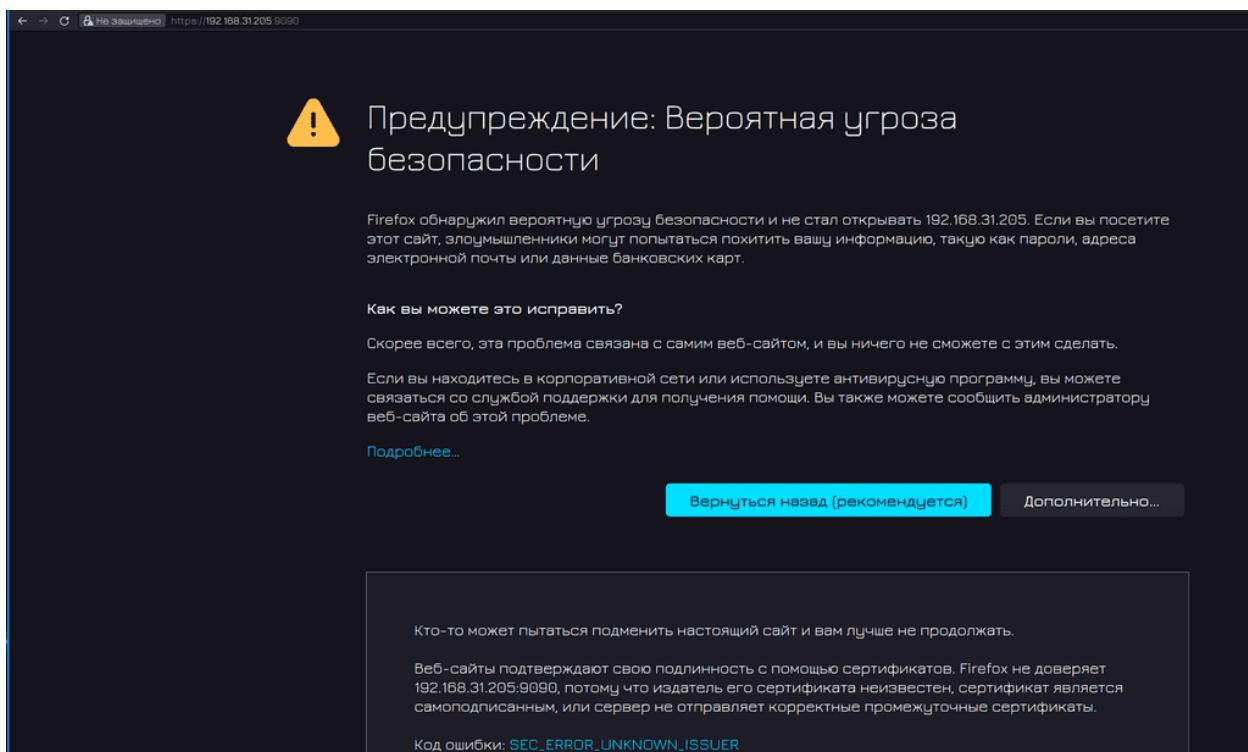
Сообщение при логине говорит о том, что сам cockpit установлен, нужно просто включить и запустить. Так и сделаем:

```
sudo systemctl enable --now cockpit.socket
```

Обратите внимание, что мы запускаем не сервис, а сокет. Это работает так - у нас на сервере работает сокет, который ждёт подключения. Если подключение есть - то systemd сам запускает сервис. Это позволяет сервису не работать постоянно, что немного облегчает нагрузку на сервер. Хотя cockpit использует довольно мало ресурсов.

Теперь, при подключении по ssh, мы будем видеть подсказку, как подключиться к веб-интерфейсу - https, ip адрес и порт 9090. Можем ещё убедиться, что у сервиса на файрволе тот же порт:

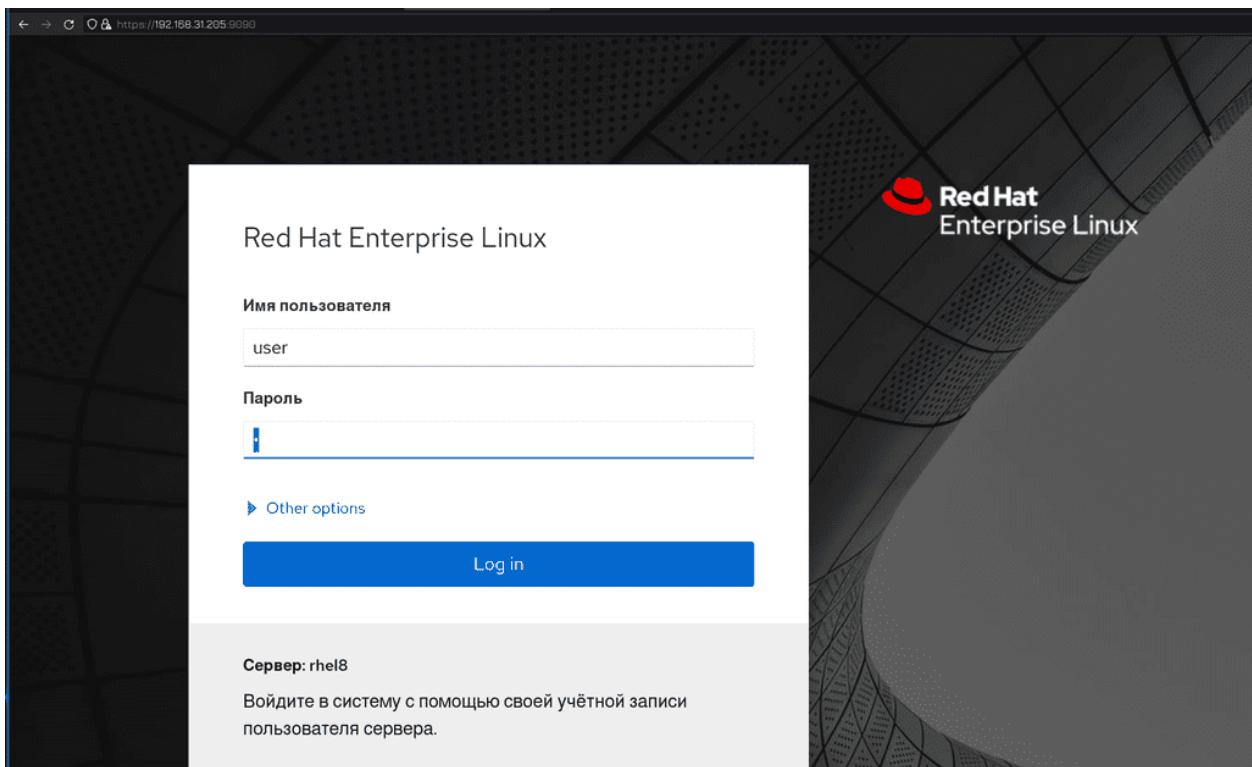
```
sudo firewall-cmd --info-service=cockpit
```



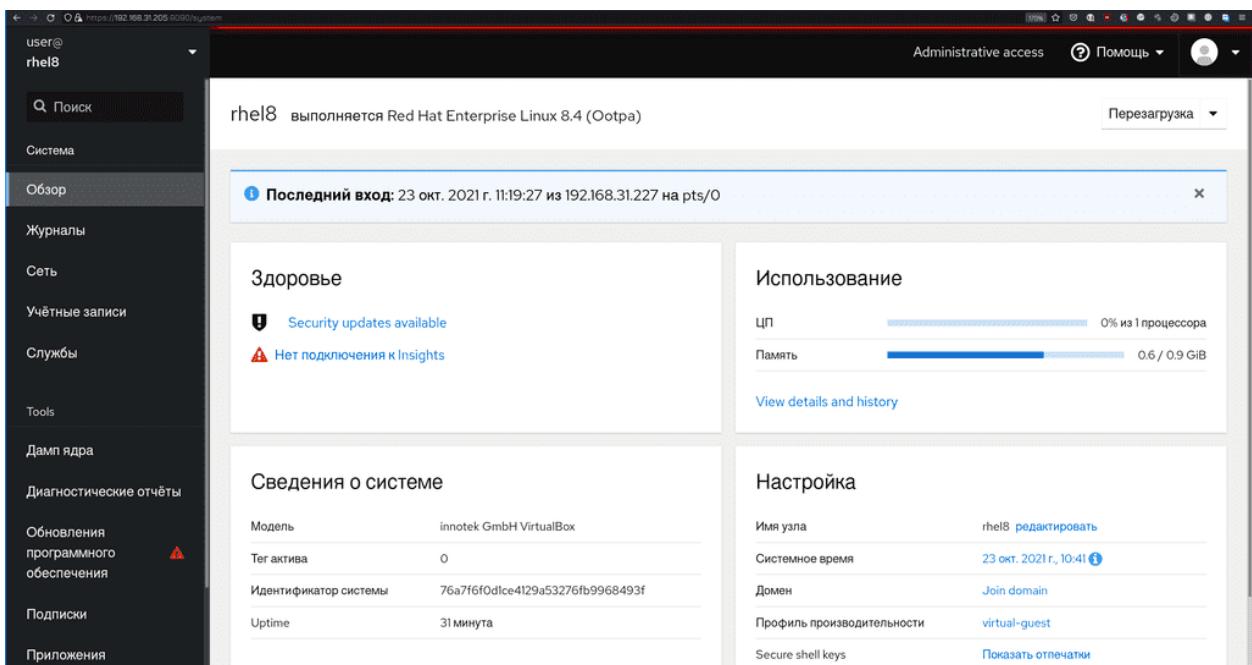
Скопируем ссылку и перейдём по ней в браузере. Сразу после этого вы увидите окно с предупреждением об угрозе безопасности. На самом деле угрозы нет, но, давайте объясню в чём дело.

Есть так называемые центры сертификации. Что-то наподобии гос. органа, выдающего паспорта. Администраторы могут обращаться к ним, чтобы запросить «паспорт» для своего сайта. Эти паспорта - SSL сертификаты. И у каждого паспорта есть специальная метка, уникальная для центра сертификации. Подделать такую метку нельзя. Так вот, браузеры доверяют центрам сертификации и следят, чтобы на сертификатах была метка, соответствующая одному из центров. Если на паспорте метка не такая или истёк срок действия паспорта и т.п., то браузер перестаёт доверять сайту и выдаёт такую ошибку. Так как это очень похоже на то, что кто-то выдаёт себя за другого.

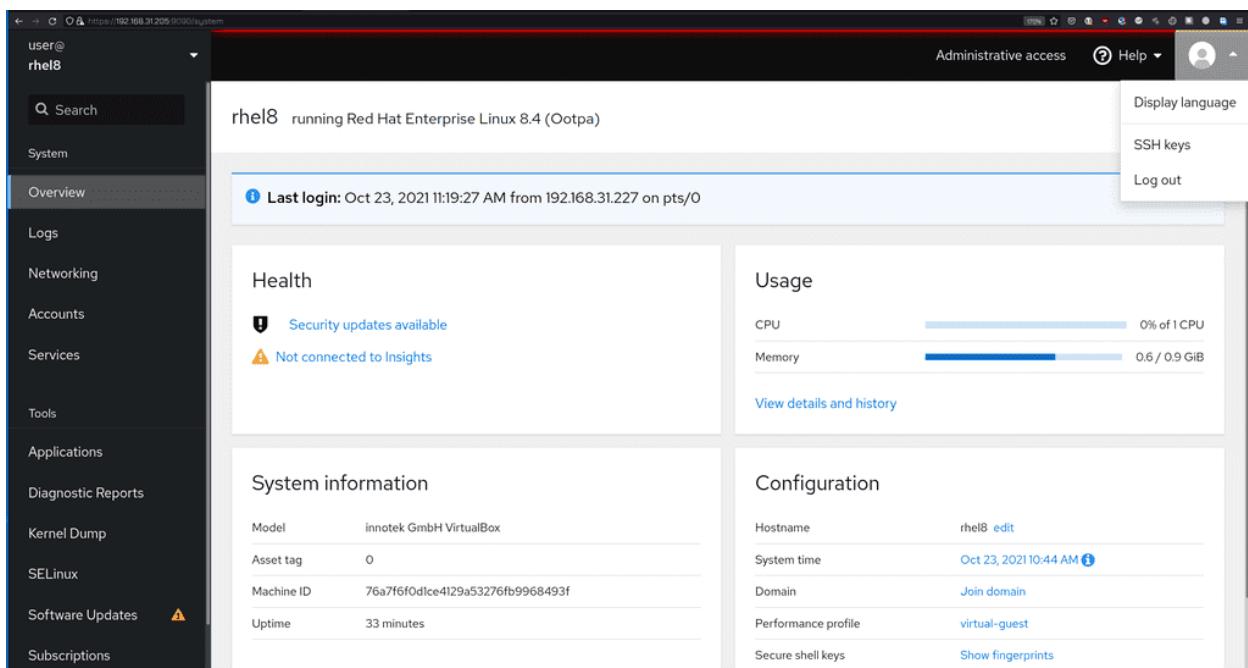
В локальной сети нет необходимости обращаться к центру сертификации, можно построить свой, что обычно и делают в компаниях. Это позволяет пользователям компании заходить на внутренние сайты без всяких ошибок. Но даже если нет своего центра сертификации, в нашем случае нет ничего страшного, мы уверены, что подключаемся именно к нашей виртуалке, поэтому можем проигнорировать ошибку. Нажимаем «Дополнительно», а затем «Принять риск и продолжить».



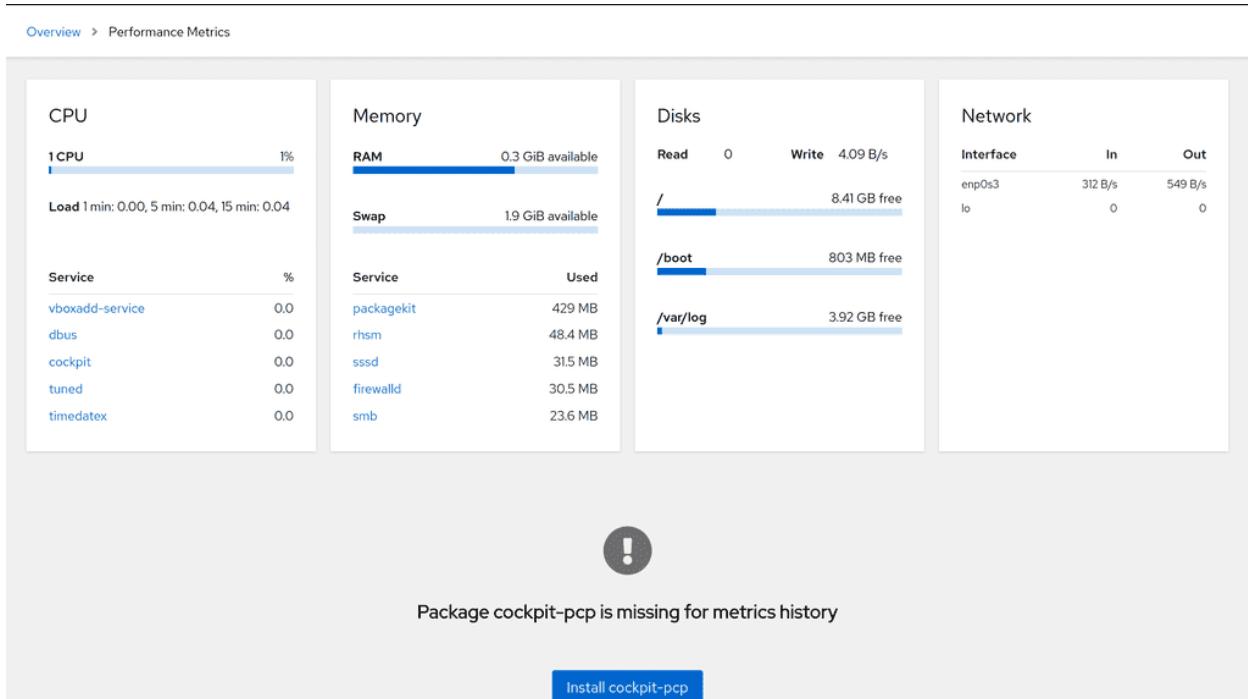
После чего нас встретит окно логина. Здесь мы можем указать пользователя из нашей системы, скажем, ruta или нашего user-а.



После логина нас встретит окно с общей информацией о системе. Так как у меня браузер на русском, то и интерфейс отображается на русском, что немного не привычно. Поэтому справа сверху можно ткнуть на иконку пользователя и сменить язык интерфейса.

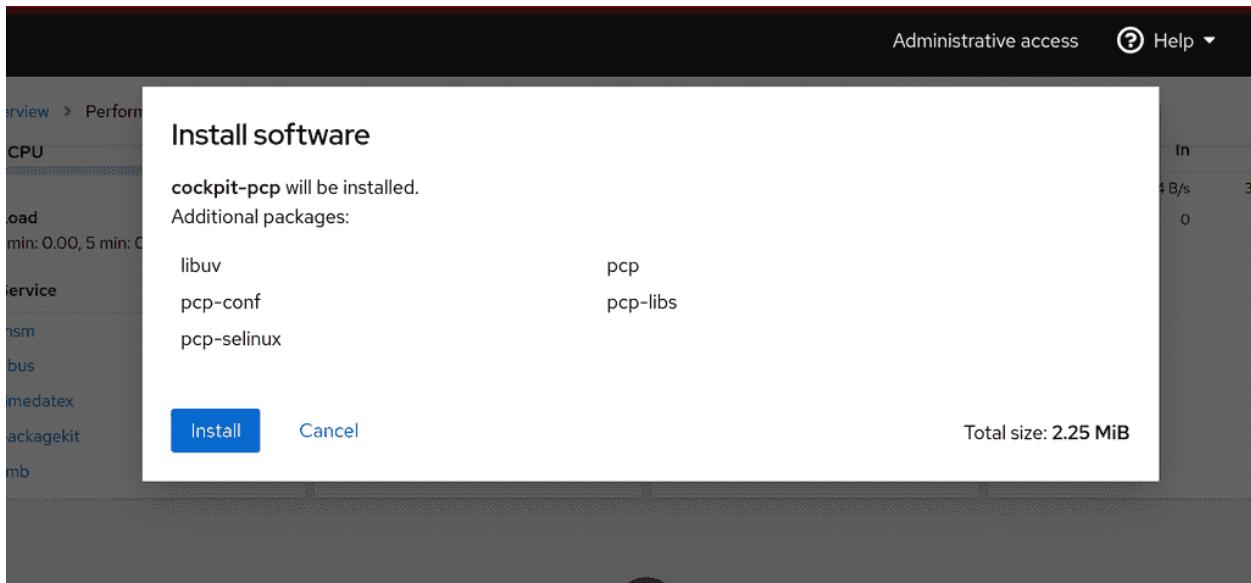


Интерфейс довольно простой и понятный. Я не буду тыкать на все все все кнопки, многое и так понятно, но, давайте пройдёмся по вкладкам. И так, Overview - общая информация о системе. Здесь мы видим, что доступны обновления безопасности, видим нагрузку на процессор, оперативку. Если бы у нас была физическая машина, мы бы видели модель и информацию о ней. Ну и различные настройки, такие как хостнейм, время, профиль производительности, домен и т.д. и т.п. Давайте нажмём на «View details and history» под «Usage».

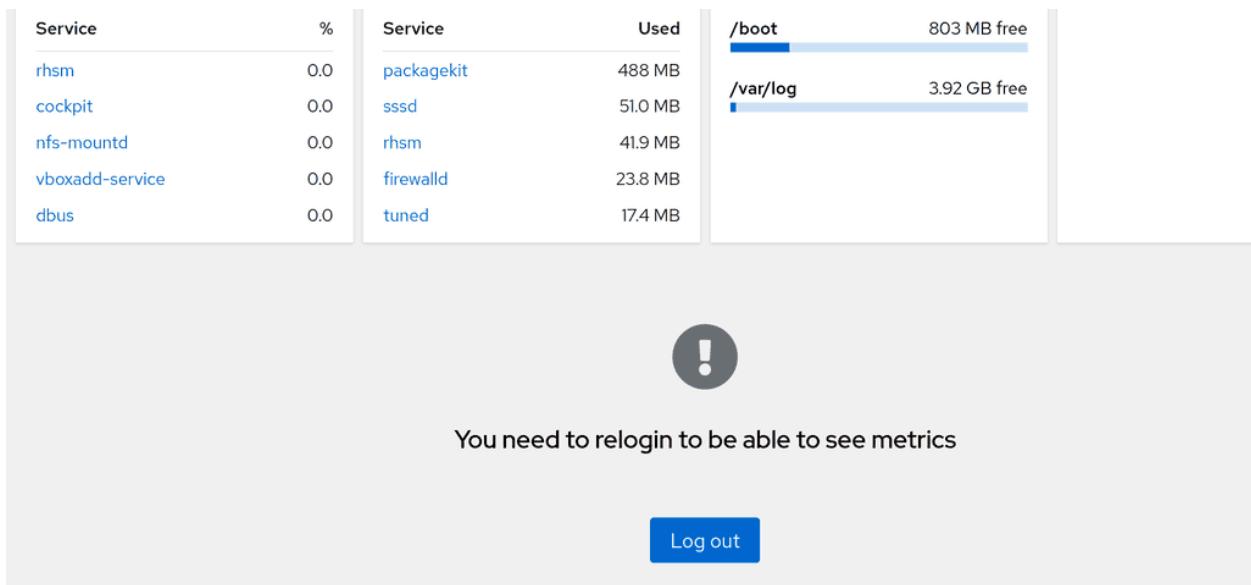


Откроется чуть детальная информация, включая сервисы, использующие CPU и оперативку, swap, использование файловых систем, использование сети и т.п. Вся эта информация о текущем использовании ресурсов называется метриками. Снизу у нас также есть предложение установить cockpit-pcp -

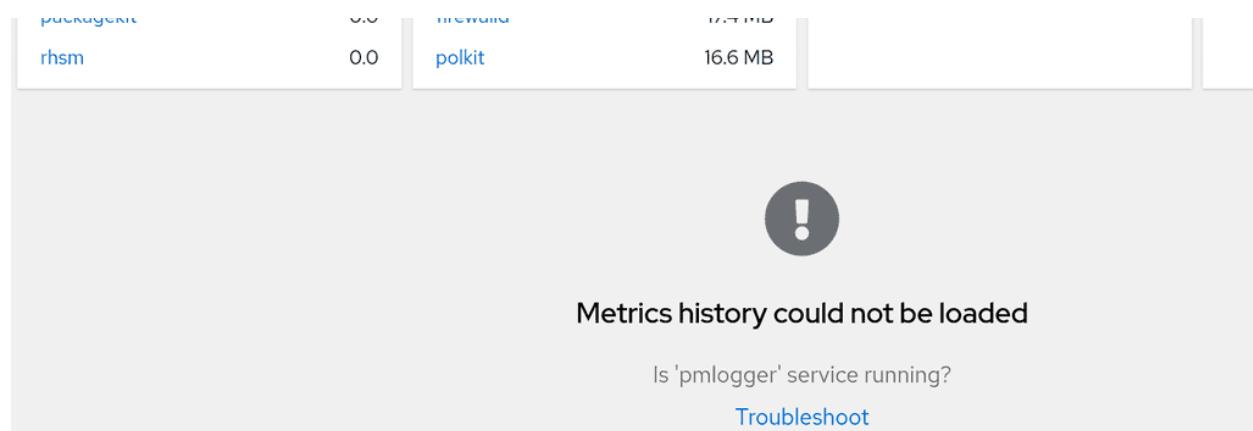
пакет, который позволит сохранять эти метрики на время, визуализировать их и т.п. Давайте нажмём на кнопку «Install».



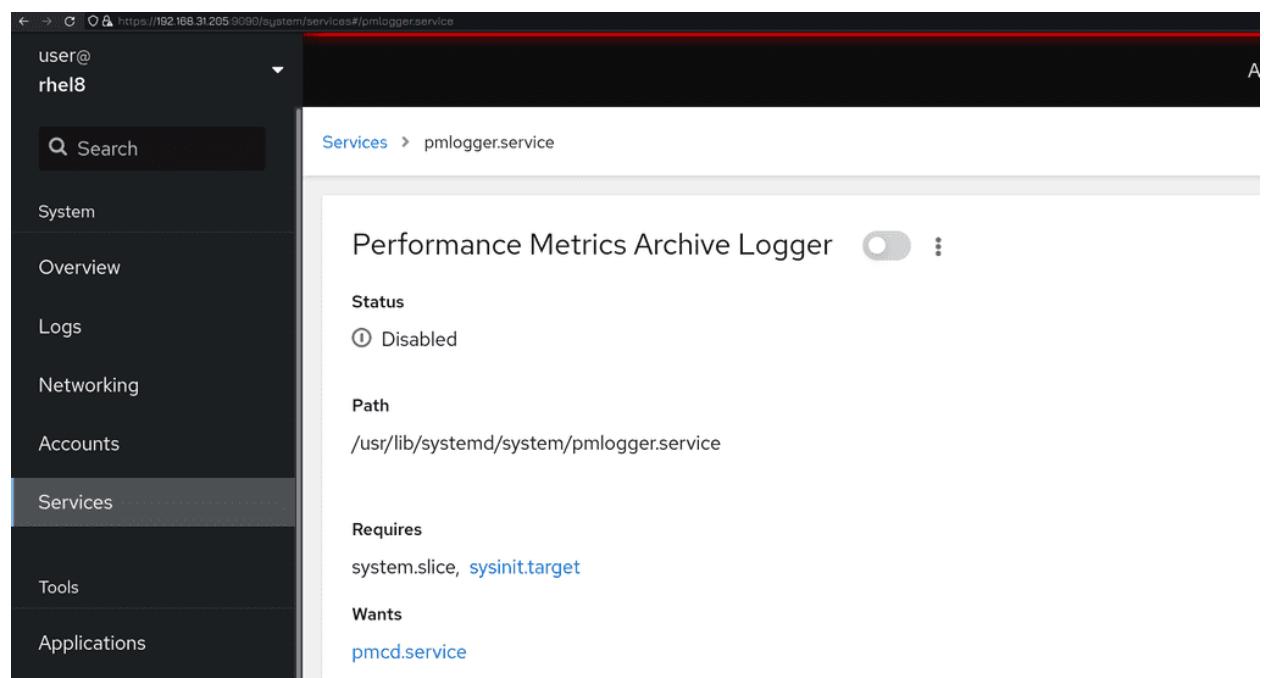
Откроется окно, в котором покажутся необходимые для установки пакеты и их общий объём. Опять нажмём Install.

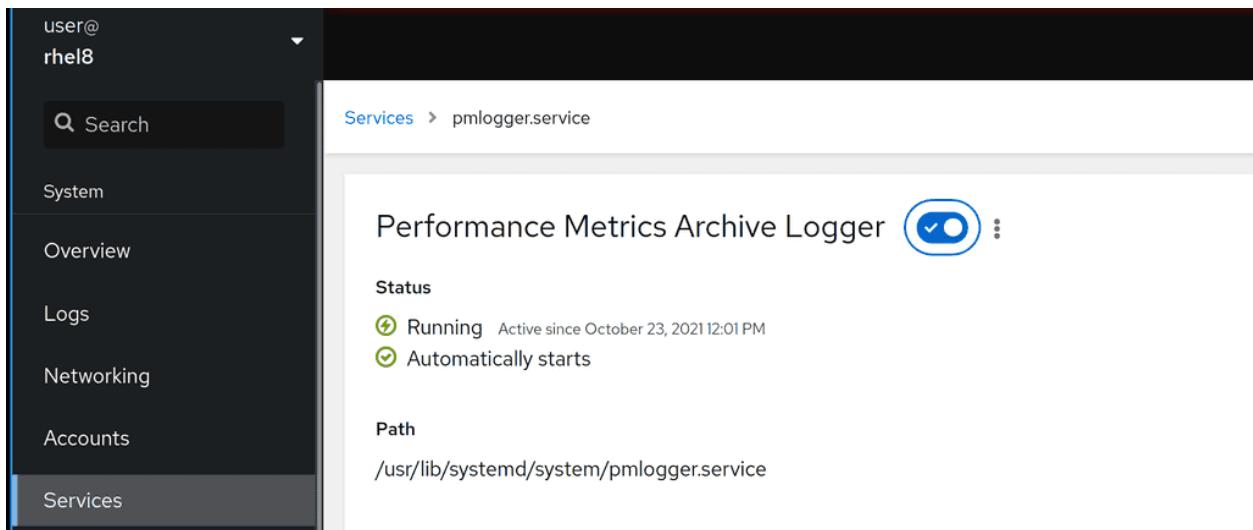


После установки нас просят перелогиниться, чтобы увидеть изменения.

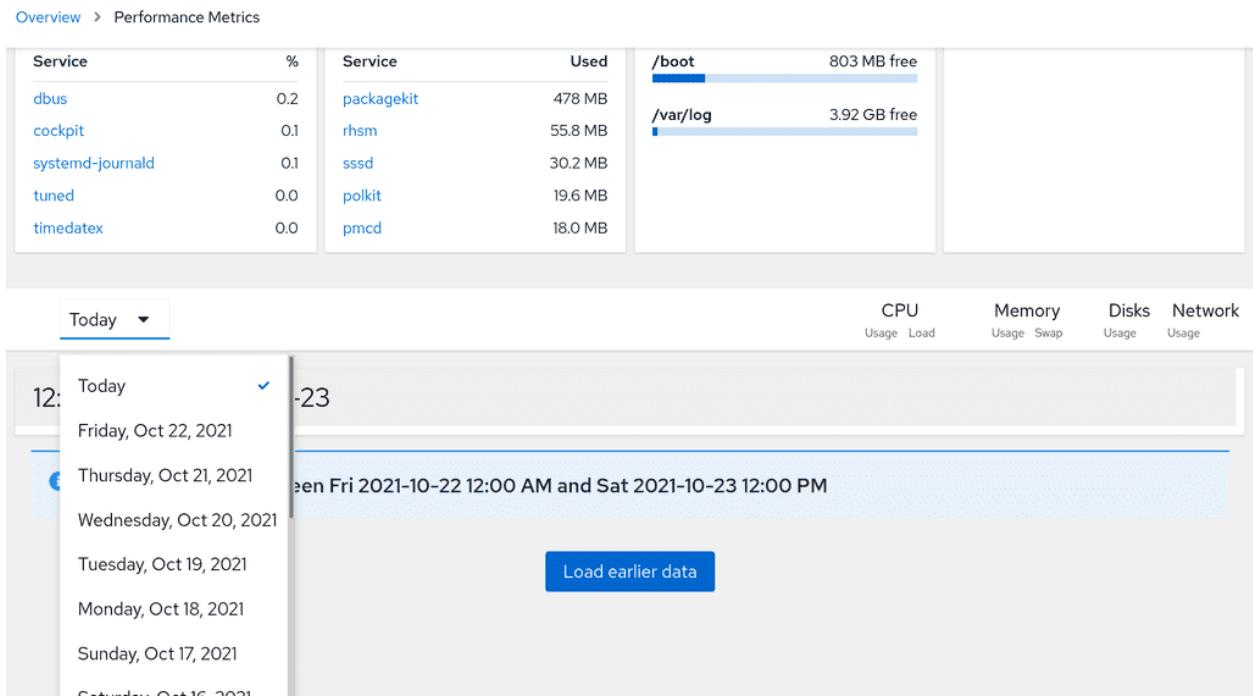


После чего нам говорят, что нет истории метрик, и спрашивают - а работает ли сервис pmlogger? И снизу кнопка troubleshoot. Давайте нажмём.





Нас перекидывает во вкладку сервиса. Как видите, статус сервиса disabled. Т.е. как обычно, после установки надо включить и запустить сервис. Здесь это можно сделать нажав на ползунок.



Теперь вернёмся в окно с метриками и обновим его. На этот раз мы просто видим, что нет информации. Сервис просто должен набрать достаточно информации. Оставим его включённым, чтобы в будущем можно было посмотреть метрики за сегодняшний и предыдущие дни.

The screenshot shows the RHEL 8 desktop environment. On the left, a dark sidebar menu includes options like System, Overview, Logs (which is selected), Networking, Accounts, Services, Tools, and Applications. The main window title is "Logs". At the top right, there are buttons for "Administrative access", "Help", and a user icon. Below the title, there's a search bar with dropdowns for "Priority" (set to "Warning and above"), "Identifier" (set to "All"), and "Text" (set to "priority:warning"). A "Pause" button is also present. The main content area displays log entries for October 23, 2021. The logs show system startup and configuration messages, such as kernel logs for network drivers and systemd services.

Во вкладке Logs, соответственно, логи. Можно их фильтровать, выбирая время, важность и прочие параметры. Ну и нажимая на лог можно увидеть больше деталей о нём.

The screenshot shows the RHEL 8 desktop environment. The sidebar menu is identical to the previous one. The main window title is "Networking". The main content area has several sections: "Transmitting" and "Receiving" network traffic graphs (Kbps vs. time); a "Firewall" section with a toggle switch and a link to "Edit rules and zones"; an "Interfaces" table listing a single interface (enp0s3) with its IP address, sending rate (726 Kbps), and receiving rate (5.58 Kbps); and a "Network logs" section with a link to "All logs".

Во вкладке Networking у нас нагрузка сети, логи, а также настройки файрвола и интерфейсов.

The screenshot shows the Cockpit web interface under the Networking > Firewall section. It displays the configuration for the 'public zone' on interface 'enp0s3'. The table lists various services and their corresponding TCP and UDP ports:

Service	TCP	UDP
ssh	22	
dhcpv6-client		546
cockpit	9090	
ntp		123
nfs	2049	
rpc-bind	111	

Buttons for 'Delete' and 'Add services' are visible at the top right of the table.

В том же самом файрволе можно добавить или удалить сервисы и зоны. Ну и всё это наглядно и просто.

The screenshot shows the Cockpit web interface under the Accounts section. It displays a list of users: 'user' (Your account), 'rheluser', and 'root'. A 'Create new account' button is located at the top left of the user list area.

Во вкладке Accounts обычные пользователи. Отсюда же можно их создавать или изменять.

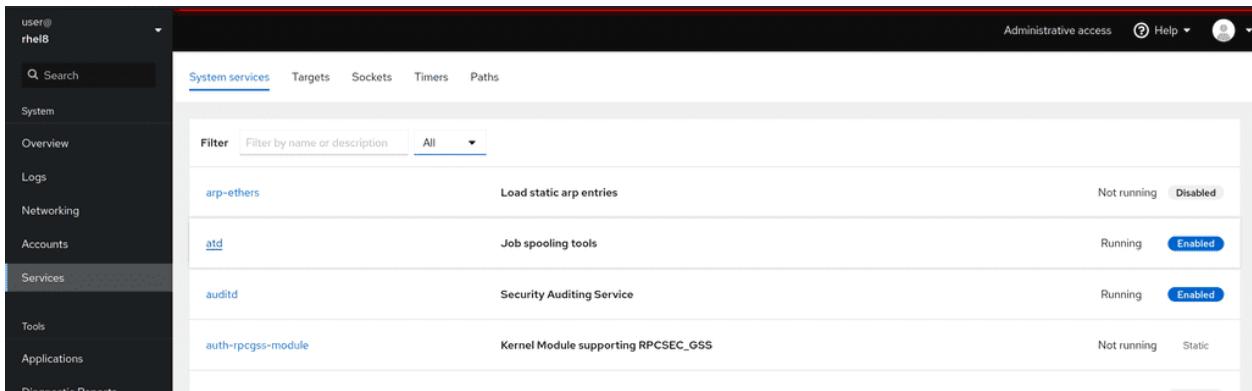
The screenshot shows the Cockpit web interface under the Accounts > user section for the 'user' account. The configuration details are as follows:

- Full name:** user
- User name:** user
- Roles:** Server administrator
- Last login:** Logged in
- Access:** Lock account (unchecked) | Never lock account
- Password:** Set password | Force change | Never expire password

Below the configuration, there is a section for 'Authorized public SSH keys' with a 'doctor@tardis' entry and a 'Remove' button.

Скажем, можно давать или отнимать административный доступ, блокировать аккаунты, менять па-

роли и даже добавлять публичные ssh ключи, чтобы пользователь мог по ssh ключам логиниться.



Ну и во вкладке Services можно смотреть информацию и управлять сервисами, скажем, включать, выключать, перезапускать и т.п.

A screenshot of the Cockpit web interface showing the Applications page. The URL is https://cockpit-project.org/applications. It lists several add-ons: Virtual Machines, Podman Containers, SELinux, and Kernel Dump. Each entry includes a brief description, package information (Package and Source), issue tracker (Issue tracker with a GitHub link), and an "official" badge. The Virtual Machines section also includes a "Create, run, and manage virtual machines in your browser." link.

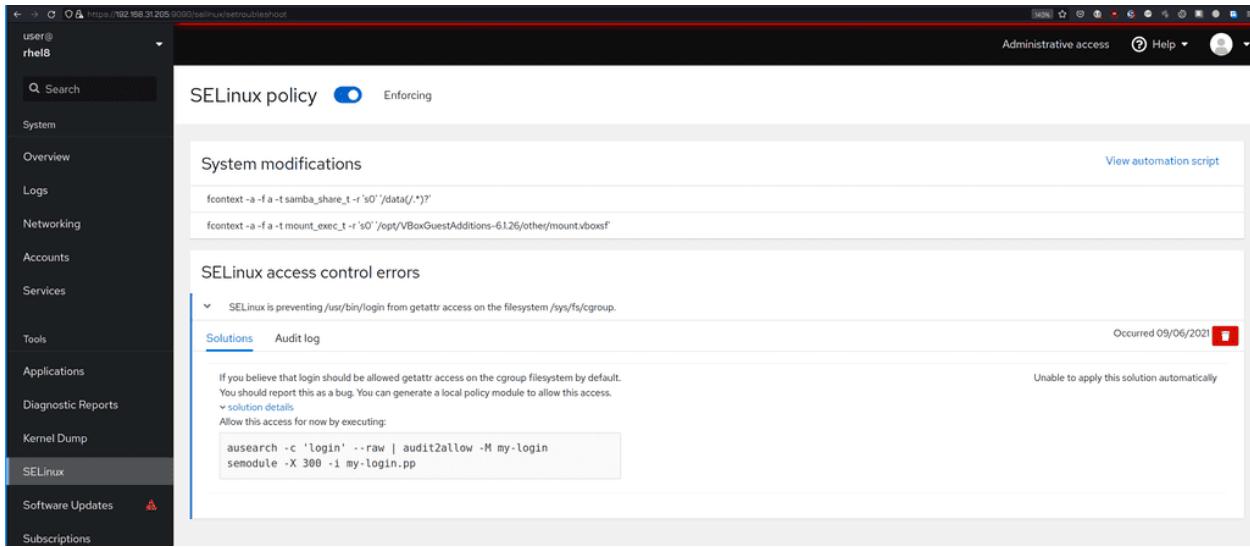
Функционал Cockpit можно расширять модулями. Скажем, есть модули, которые могут управлять виртуалками, контейнерами, файловыми шарами, селинуксом и т.п. На [сайте проекта](#) есть список известных модулей.

```
[user@rhel8 ~]$ sudo dnf search cockpit-
[sudo] password for user:
Updating Subscription Management repositories.
Last metadata expiration check: 10 days, 14:10:22 ago on Tue 12 Oct 2021 09:19:42 PM MSK.
=====
Name Matched: cockpit-
cockpit-bridge.x86_64 : Cockpit bridge server-side component
cockpit-composer.noarch : Composer GUI for use with Cockpit
cockpit-dashboard.noarch : Cockpit remote server dashboard
cockpit-doc.noarch : Cockpit deployment and developer guide
cockpit-machines.noarch : Cockpit user interface for virtual machines
cockpit-packagekit.noarch : Cockpit user interface for packages
cockpit-pcp.x86_64 : Cockpit PCP integration
cockpit-podman.noarch : Cockpit component for Podman containers
cockpit-session-recording.noarch : Cockpit Session Recording
cockpit-storaged.noarch : Cockpit user interface for storage, using udisks
cockpit-system.noarch : Cockpit admin interface package for configuring and troubleshooting a
                         : system
cockpit-ws.x86_64 : Cockpit Web Service
[user@rhel8 ~]$
```

Некоторые модули можно найти в репозиториях и установить:

```
sudo dnf search cockpit-
```

Ну и любой желающий может разработать свои модули.



Вернёмся ко вкладкам. Один из модулей - SELinux - позволяет увидеть наши модификации, посмотреть ошибки и даже предложить решения проблем.

The screenshot shows the Red Hat Enterprise Linux 8 interface. On the left, there's a sidebar with various navigation options: System, Overview, Logs, Networking, Accounts, Services, Tools, Applications, Diagnostic Reports, Kernel Dump, SELinux, Software Updates (which is currently selected), Subscriptions, and Terminal. The main content area has two tabs: 'Status' and 'Automatic updates'. The 'Status' tab shows '43 updates available, including 18 security fixes' and 'Last checked: 42 minutes ago'. The 'Automatic updates' tab says 'Automatic updates are not set up'. Below these tabs is a section titled 'Available updates' with a table. The table columns are Name, Version, Severity, and Details. The details for each update are summarized as follows:

Name	Version	Severity	Details
bpftool, kernel, kernel-core, kernel-devel, ...	418.0-305.19...	严重 (Severe) 1	The kernel packages contain the Linux kernel, the core of any Linux operating system.
krb5-libs	1.18.2-8.3.el8_4	严重 (Severe) 2	Kerberos is a network authentication system, which can improve the security of your network by eliminating the insecure practice of sending passwords over the network in unencrypted form. It allows clients and servers to authenticate to each other with the help of a trusted third party, the Kerberos key distribution center (KDC).
curl, libcurl	7.61.1-18.el8_4.i	严重 (Severe) 3	The curl packages provide the libcurl library and the curl utility for downloading files from servers using various protocols, including HTTP, FTP, and LDAP.
nsspr	4.32.0-1.el8_4	严重 (Severe) 1	Network Security Services (NSS) is a set of libraries designed to support the cross-platform development of security-enabled client and server applications.
nss, nss-softokn, nss-softokn-freebl, nss-sysinit, ...	3.670-6.el8_4	严重 (Severe) 1	Network Security Services (NSS) is a set of libraries designed to support the cross-platform development of security-enabled client and server applications.
ca-certificates	2021.250-80...	严重 (Severe) 1	The ca-certificates package contains a set of Certificate Authority (CA)
dnf-plugin-subscription-manager, python3-subscription-manager-rhsm, python3-svspurpose, rhsm-icons, ...	1.28.13-4.el8_4	严重 (Severe) 1	The subscription-manager packages provide programs and libraries to allow

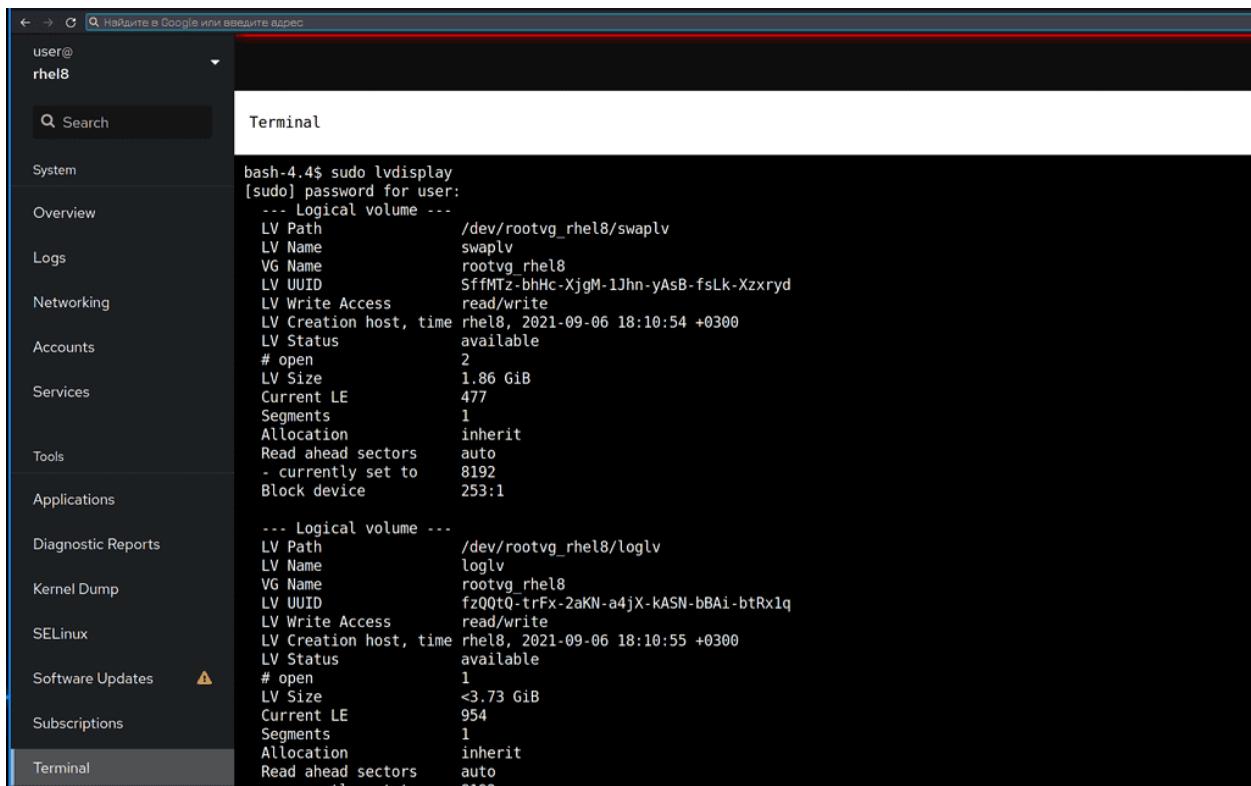
At the bottom right of the 'Available updates' section are two buttons: 'Install security updates' and 'Install all updates'.

Во вкладке Software Updates мы можем посмотреть доступные обновления, их важность и выбрать что необходимо обновить – только обновления безопасности или все пакеты.

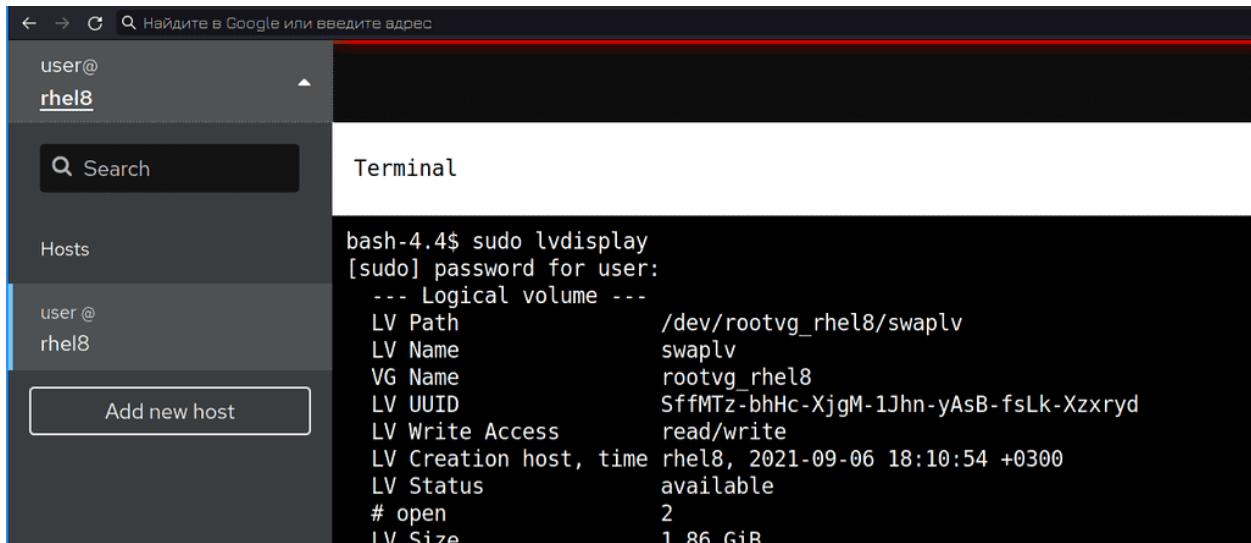
The screenshot shows the Red Hat Enterprise Linux 8 interface. The sidebar is identical to the previous screenshot, with 'Software Updates' selected. The main content area is titled 'Subscriptions'. It shows 'Status: Current' and a 'Unregister' button. Below that is 'System Purpose' with 'Status: Not Specified'. Under 'Installed products', it lists 'Red Hat Enterprise Linux for x86_64' with an 'Auto-attach' button. A detailed view of the product information is shown in a modal window:

Product Name	Red Hat Enterprise Linux for x86_64
Product ID	479
Version	8.4
Arch	x86_64
Status	Subscribed
Starts	09/06/2021
Ends	09/06/2022

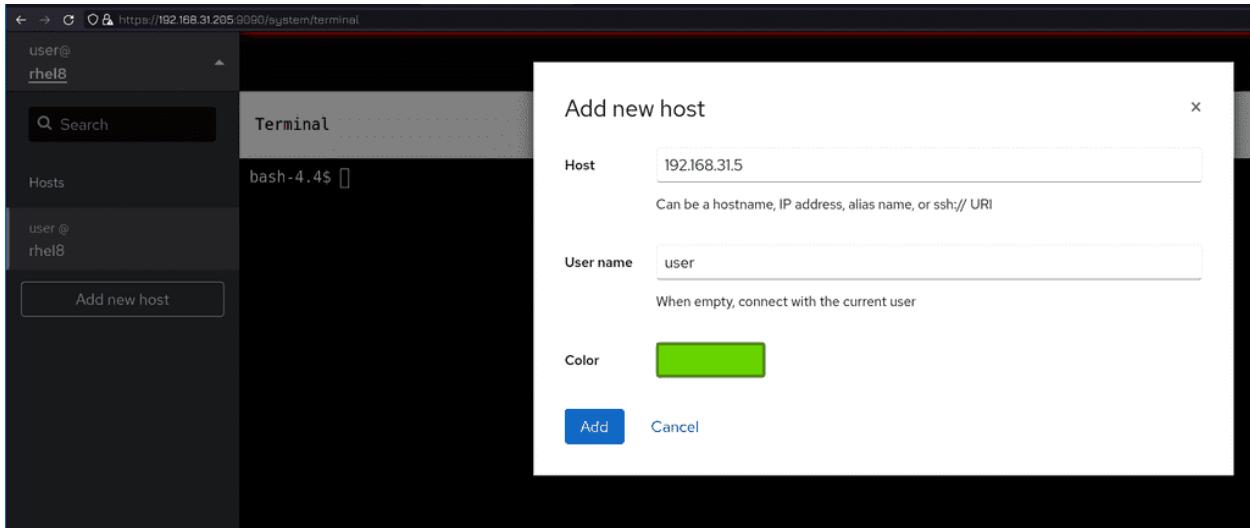
Во вкладке Subscription можно зарегистрировать RHEL или посмотреть статус подписки, время, когда закончится подписка и прочую информацию.



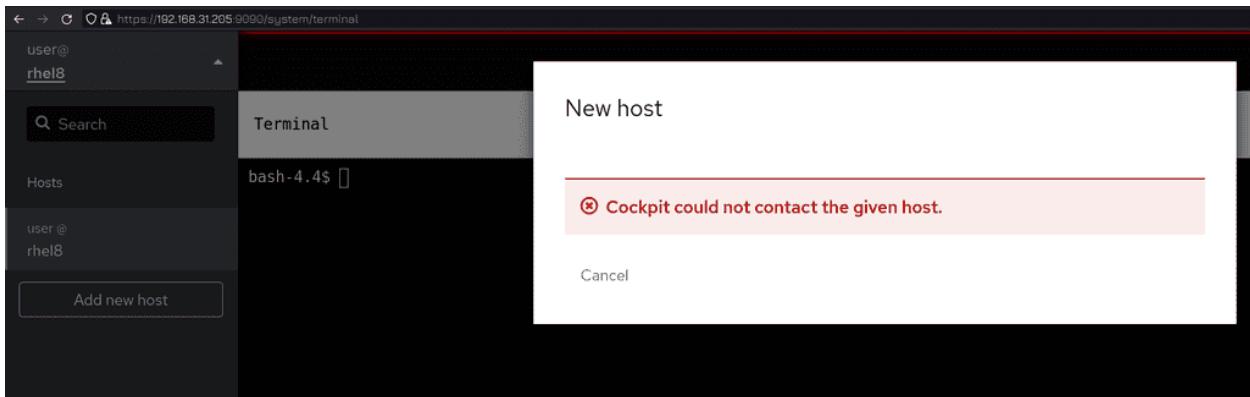
Ну и вкладка Terminal позволяет запускать команды прямо из браузера. Это не так удобно, как работать с терминалом напрямую, так как некоторые горячие клавиши перехватываются браузером, но для простых задач вполне хороший вариант.



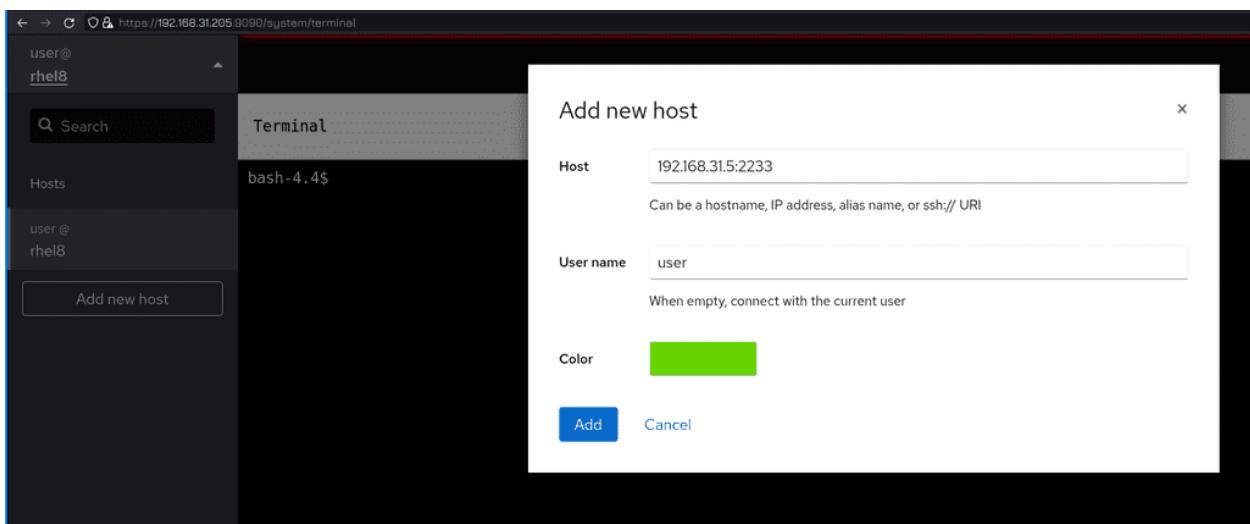
В один Cockpit можно подключить несколько хостов для мониторинга и управления. Чтобы добавить новый хост, нажмите в левом верхнем углу на выпадающее меню и нажмите «Add new host».



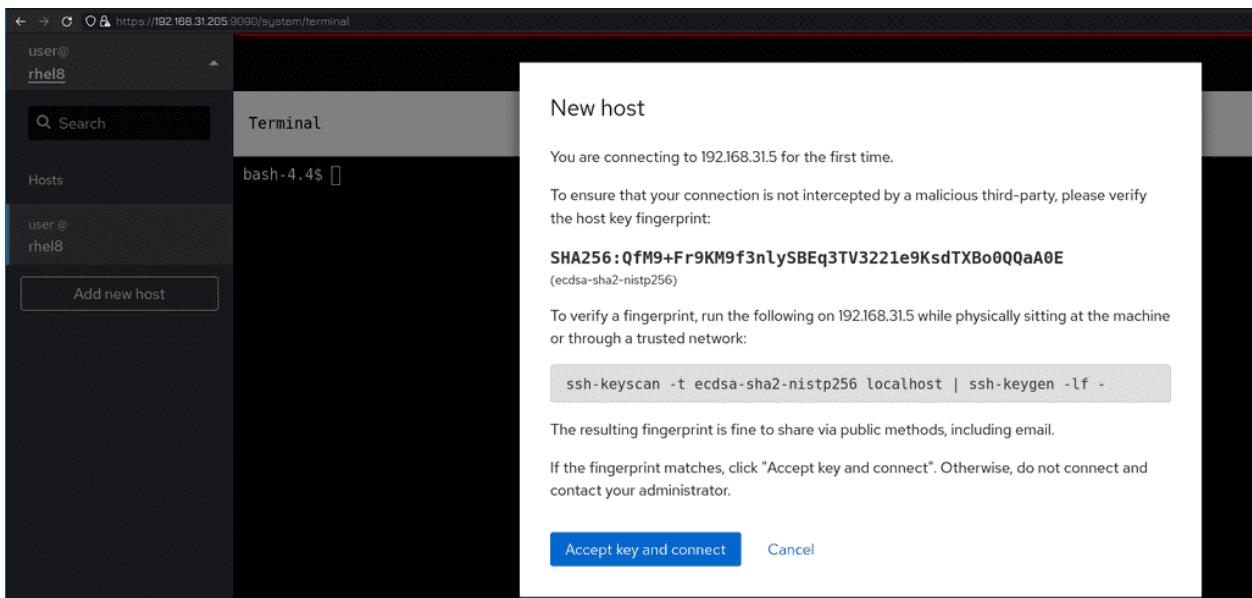
Давайте пропишем здесь наш CentOS, укажем IP адрес и логин пользователя, после чего нажмём Add.



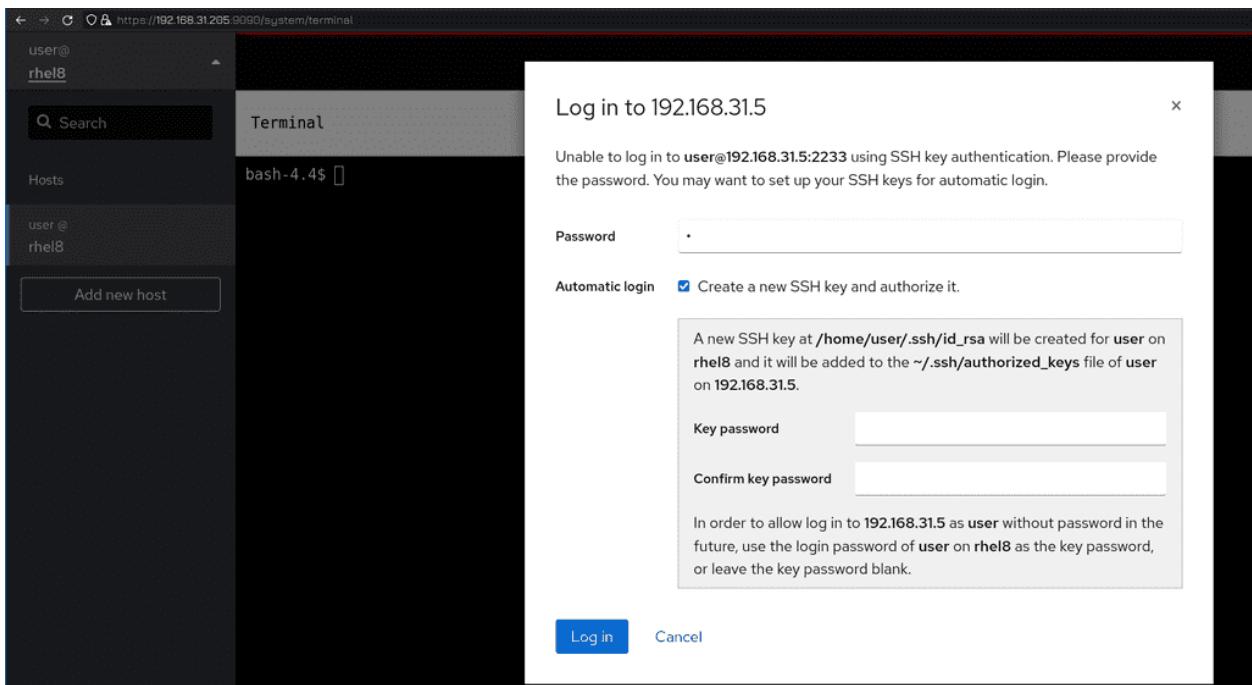
Но мы видим ошибку - не удаётся связаться с хостом. Cockpit пытается подключиться к хосту по ssh. На CentOS-е мы меняли порт ssh на 2233.



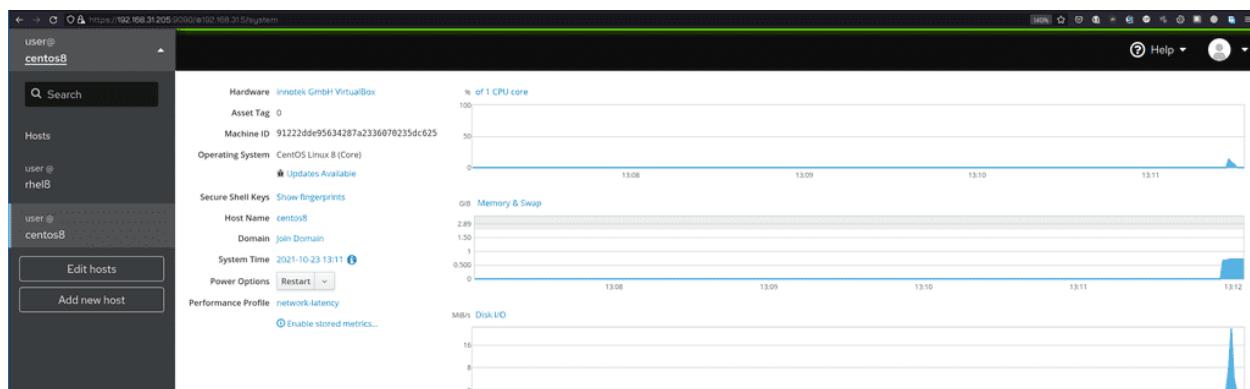
Давайте попробуем заново добавить, указав при этом нужный порт.



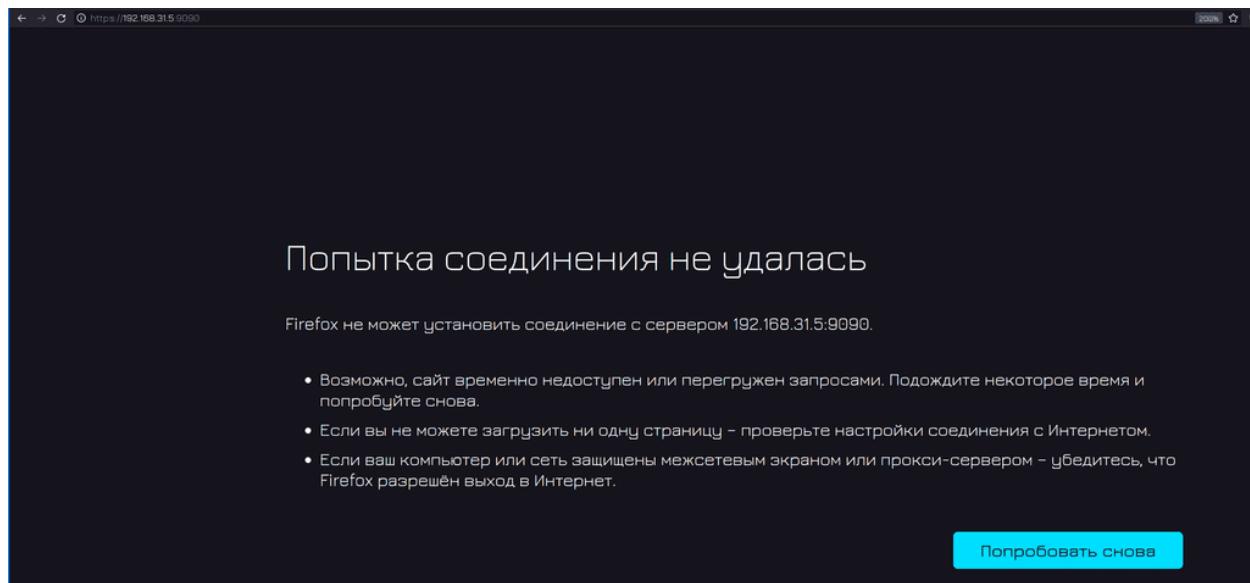
Теперь мы видим сообщение с ключом хоста. Да, мы уверены, что подключаемся к тому самому хосту.



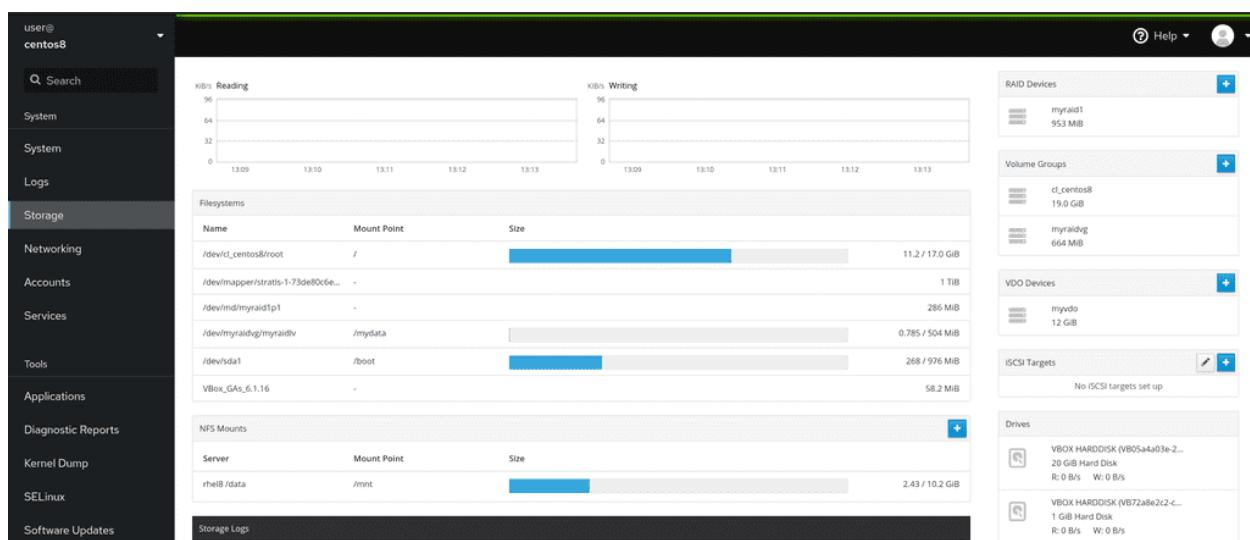
Потом у нас спросят пароль и предложат создать ssh ключ, чтобы user с хоста rhel мог без пароля подключаться к user-у с хоста centos.



Теперь у нас в левом меню два хоста и мы можем выбирать, каким хотим управлять.



При этом на самом Centos Cockpit не запущен, RHEL лишь управляет Centos через SSH.



На Centos-е внешний вид и некоторые вкладки могут отличаться. К примеру, здесь у нас стоит пакет cockpit-storaged, который позволяет управлять дисками, RAIDом, LVM, NFS и прочим. Т.е. у вас могут

быть хосты, на которых нужна виртуализация - для них ставите одни модули. На других хостах контейнеры - ставите другие. И, соответственно, на каждом хосте вы будете видеть и управлять тем, что вам нужно.

Говоря в целом о вкладках, о различном функционале Cockpit, всё это вам должно быть знакомо, потому что мы всё это разбирали. Зная, как это работает на самом деле, что делает каждая из этих кнопок, вы понимаете процесс - что, в каком порядке и зачем делать. Cockpit лишь веб-админка, которая обрамляет часто используемые простые задачи в красивый и простой интерфейс. Он, конечно, не замена командной строке, но некоторые вещи здесь смотреть и настраивать удобнее.

2.61 61. Глоббинг и регулярные выражения

2.61.1 61. Глоббинг и регулярные выражения

Регулярные выражения (regexp) — основы

Тестирование IT-систем *, Регулярные выражения *

Регулярные выражения (их еще называют *regexp*, или *regex*) — это механизм для поиска и замены текста. В строке, файле, нескольких файлах... Их используют разработчики в коде приложения, тестировщики в автотестах, да просто при работе в командной строке!

Чем это лучше простого поиска? Тем, что позволяет задать шаблон.

Например, на вход приходит дата рождения в формате ДД.ММ.ГГГГ. Вам надо передать ее дальше, но уже в формате ГГГГ-ММ-ДД. Как это сделать с помощью простого поиска? Вы же не знаете заранее, какая именно дата будет.

Как мне найти в тексте именно даты?

У нас накопилось несколько тем, которые мы вскользь упомянули в течении курса и которые требуют некоторой ясности. В частности это регулярные выражения, глоббинг и такие утилиты, как grep, sed, awk. Но регулярные выражения огромная тема, которую можно усложнять до бесконечности, они очень специфичны и необходимости всем детально в них разбираться нет. Поэтому я буду ориентироваться лишь на то, с чем лично сталкиваюсь и что легко запомнить. Если вы хотите углубиться в эту тему, то советую начать со [статьи](#) на хабре.

Перечисление

```
[user@rhel8 ~]$ mkdir test
[user@rhel8 ~]$ touch test/file{1..3}
[user@rhel8 ~]$ ls test/
file1 file2 file3
[user@rhel8 ~]$ touch test/file1 test/file2 test/file3
[user@rhel8 ~]$ █
```

Начнём с перечисления. Оно напрямую связано с оболочкой командной строки, например, с тем же bash:

```
mkdir test
touch test/file{1..3}
```

Видите фигурные кавычки? Это один из примеров перечисления. Суть в том, что сама команда touch может не уметь работать с этими кавычками. Прежде чем выполнить команду, bash написанное пропускает через себя, при этом подставляя значения переменных, выполняя перечисление и т.п. И, в итоге, на самом деле, выполняется такая команда:

```
touch test/file1 test/file2 test/file3
```

А это уже простой синтаксис для команды touch. Тоже самое касается любой другой программы или самописного скрипта - bash за вас подставит названия файлов.

```
[user@rhel8 ~]$ sudo cp /etc/fstab{,.bkp}
[sudo] password for user:
[user@rhel8 ~]$ ls /etc/fstab
fstab      fstab.bkp
[user@rhel8 ~]$ sudo cp /etc/fstab /etc/fstab.bkp
[user@rhel8 ~]$ █
```

Очень частый пример использования фигурных кавычек - эдакий бэкап файлов. Например:

```
sudo cp /etc/fstab{,.bkp}
```

До этого мы в кавычках писали диапазон значений. Сейчас же таким выражением мы просто взяли два значения - /etc/fstab и /etc/fstab.bkp. Запятая в фигурных скобках разделяет эти два значения. Т.е. в итоге команда после обработки bash-ем выглядит так:

```
sudo cp /etc/fstab /etc/fstab.bkp
```

Т.е. особого смысла в таком перечислении нет, так как второй раз написать /etc/fstab будет проще, чем заморачиваться с фигурными кавычками. Но если путь длинный или в каких-то скриптах это может иметь смысл. Ну и наткнувшись на такое выражение в какой-нибудь статье в интернете вы будете понимать, что вообще происходит.

Globbing

```
[user@rhel8 ~]$ ls /etc/fstab*
/etc/fstab /etc/fstab.bkp
[user@rhel8 ~]$ ls /etc/l*.conf
/etc/ld.so.conf /etc/libaudit.conf /etc/libuser.conf /etc/locale.conf /etc/logrotate.conf
[user@rhel8 ~]$ ls /etc/[Dd]*
/etc/DIR_COLORS /etc/DIR_COLORS.256color /etc/DIR_COLORS.lightbgcolor /etc/dracut.conf

/etc/dbus-1:
session.conf session.d system.conf system.d
```

Теперь про глоббинг. Глоббинг нужен для подстановки имён файлов и напоминает регулярные выражения. Мы не раз прибегали к глоббингу при работе с файлами. wildcard, или звёздочка (*), заменяют любое количество символов, хоть в конце файла, хоть в начале, хоть посреди:

```
ls /etc/fstab*
ls /etc/l*.conf
```

В квадратных скобках можно указать несколько вариантов одного символа. Скажем, если нужно найти все файлы, начинающиеся с буквы d, маленькой или большой:

```
ls /etc/ [Dd]*
```

```
[user@rhel8 ~]$ echo /etc/???
/etc/dnf /etc/gss /etc/lsm /etc/lvm /etc/opt /etc/pcp /etc/pki /etc/rhc /etc/rpc /etc/rpm /etc/os /etc/ssh /etc/ssl /etc/X11 /etc/xdg /etc/yum
[user@rhel8 ~]$ echo /etc/[sd]*.{conf,d}
/etc/dracut.conf /etc/sestatus.conf /etc/sudo.conf /etc/sudo-ldap.conf /etc/sysctl.conf /etc/depmod.d /etc/dracut.conf.d /etc/sudoers.d /etc/sysctl.d
[user@rhel8 ~]$
```

Также есть вопросительный знак, который заменяет один символ. Например, если мы знаем количество символов в названии файла или не знаем только определённые пару символов:

```
echo /etc/???
```

Ну и естественно, выражения можно совмещать. Например, найдём все файлы, которые начинаются на s или d и заканчиваются на .conf или .d:

```
echo /etc/ [sd]*.{conf,d}
```

В целом, глоббинг небольшая и простая тема. Чуть больше примеров с глоббингом можете найти по [ссылке](#).

Регулярные выражения

В отличии от глоббинга, который нужен для подстановки имён файлов, регулярные выражения нужны для работы с текстом, поиска и замены. Регулярные выражения универсальны для разных программ и их уже не bash обрабатывает, а сами программы.

```
[user@rhel8 ~]$ grep "[AP]" /etc/passwd
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
tss:x:59:59:Account used for TPM access:/dev/null:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
pcp:x:989:985:Performance Co-Pilot:/var/lib/pcp:/sbin/nologin
[user@rhel8 ~]$ █
```

Глоббинг сделан на основе регулярок, поэтому и здесь эти символы актуальны, но уже с точки зрения текста. Например, найдём в тексте все строки, содержащие большую букву А или Р.

```
grep "[AP]" /etc/passwd
```

Но, как я уже сказал, я постараюсь концентрироваться на более реальных случаях. К примеру, возьмём настройку какого-нибудь сервиса, допустим, веб-сервера. Хотя мы его пока не проходили, нас интересует только обработка текста, а в конфигах вебсервера много всяких строк.

```
[user@rhel8 ~]$ sudo dnf install httpd
Updating Subscription Management repositories.
Last metadata expiration check: 0:00:15 ago on Sat 30 Oct 2021 11:31:27 AM MSK.
Dependencies resolved.
=====
 Package           Arch   Version            Repository
 =====
 Installing:
 httpd           x86_64  2.4.37-39.module+el8.4.0+12865+a7065a39.1
                           rhel-8-for-x86_64-appstream-
 Installing dependencies:
=====

```

Начнём с установки, пакет называется httpd:

```
sudo dnf install httpd
```

```
[user@rhel8 ~]$ head /etc/httpd/conf/httpd.conf
#
# This is the main Apache HTTP server configuration file. It contains the
# configuration directives that give the server its instructions.
# See <URL:http://httpd.apache.org/docs/2.4/> for detailed information.
# In particular, see
# <URL:http://httpd.apache.org/docs/2.4/mod/directives.html>
# for a discussion of each configuration directive.
#
# See the httpd.conf(5) man page for more information on this configuration,
# and httpd.service(8) on using and configuring the httpd service.
[user@rhel8 ~]$ wc -l /etc/httpd/conf/httpd.conf
356 /etc/httpd/conf/httpd.conf
[user@rhel8 ~]$ █
```

После установки глянем основной файл настроек - /etc/httpd/conf/httpd.conf. Большой конфиг файл с кучей комментариев. Давайте посмотрим, сколько всего здесь строк:

```
wc -l /etc/httpd/conf/httpd.conf
```

356 строк, довольно много. Ориентироваться в таком файле неудобно, но мы знаем, что большая часть строк просто комментарии и они нам не нужны.

grep

```
[user@rhel8 ~]$ grep -v "^#" /etc/httpd/conf/httpd.conf | head
ServerRoot "/etc/httpd"
Listen 80
Include conf.modules.d/*.conf
User apache
Group apache
[user@rhel8 ~]$
```

Давайте выведем только те строки, в которых нет комментариев. Все строки с комментариями начинаются со знака решётки, а значит с помощью grep мне нужно вывести все строки, которые не начинаются с этого символа. Можно использовать символ карат(^), чтобы найти все строки, начинаяющиеся с решётки и ключ -v, чтобы инверсировать запрос:

```
grep -v "^#" /etc/httpd/conf/httpd.conf
```

Теперь мы уже видим непосредственно настройки, а не комментарии. Хотя всё ещё куча пустых строк, от которых мы тоже можем избавиться.

```
[user@rhel8 ~]$ grep -v "^#" /etc/httpd/conf/httpd.conf | grep -v "^$" | head
ServerRoot "/etc/httpd"
Listen 80
Include conf.modules.d/*.conf
User apache
Group apache
ServerAdmin root@localhost
<Directory />
    AllowOverride none
    Require all denied
</Directory>
[user@rhel8 ~]$
```

Самым простым вариантом будет поставить ещё один grep после пайпа и найти все строки, начинаяющиеся и заканчивающиеся ничем, т.е. являющиеся пустой строкой. Если карат - начало строки, то доллар - конец строки. Объединив их мы можем найти пустые строки:

```
grep -v "^#" /etc/httpd/conf/httpd.conf | grep -v "^$"
```

```
[user@rhel8 ~]$ grep -Ev "^#|^$" /etc/httpd/conf/httpd.conf | head
ServerRoot "/etc/httpd"
Listen 80
Include conf.modules.d/*.conf
User apache
Group apache
ServerAdmin root@localhost
<Directory />
    AllowOverride none
    Require all denied
</Directory>
[user@rhel8 ~]$
```

Мы можем объединить оба глер-а символом «или» - прямой линией. Но чтобы это выглядело чуть более читаемо, grep-у нужен дополнительный ключ - E:

```
grep -Ev "^#|^$" /etc/httpd/conf/httpd.conf
```

```
[user@rhel8 ~]$ grep -Ev "^(#|$)" /etc/httpd/conf/httpd.conf | head
ServerRoot "/etc/httpd"
Listen 80
Include conf.modules.d/*.conf
User apache
Group apache
ServerAdmin root@localhost
<Directory />
    AllowOverride none
    Require all denied
</Directory>
[user@rhel8 ~]$
```

Так как карет относится и к первому выражению, и ко второму, мы можем их сгруппировать с помощью круглых скобок:

```
grep -Ev "^(#|$)" /etc/httpd/conf/httpd.conf
```

```
[user@rhel8 ~]$ grep -Ev "^(#|$)" /etc/httpd/conf/httpd.conf | tail
<IfModule mime_magic_module>
#
# The mod_mime_magic module allows the server to use various hints from the
# contents of the file itself to determine its type. The MIMEMagicFile
# directive tells the module where the hint definitions are located.
#
MIMEMagicFile conf/magic
</IfModule>
EnableSendfile on
IncludeOptional conf.d/*.conf
[user@rhel8 ~]$
```

Но не то, чтобы наше выражение полностью избавило вывод от закомментированных строк. Если

посмотреть весь файл, то можно заметить, что какие-то закомментированные строки остались, потому что они начинаются с пробелов, что не подходит под наше выражение.

```
user@rhel8:~$ grep -Ev "^\\s*(#|\\$)" /etc/httpd/conf/httpd.conf | tail
    AddType application/x-gzip .gz .tgz
    AddType text/html .shtml
    AddOutputFilter INCLUDES .shtml
</IfModule>
AddDefaultCharset UTF-8
<IfModule mime_magic_module>
    MIMEMagicFile conf/magic
</IfModule>
EnableSendfile on
IncludeOptional conf.d/*.conf
[user@rhel8 ~]$
```

То есть, нам желательно ещё убрать все строки, содержащие просто пробелы или начинающиеся с пробелов, за которыми начинается комментарий. Для этого можем добавить специальное выражение `\s`, которое соответствует пробелам и табуляции:

```
grep -Ev "^\s*(#|\\$)" /etc/httpd/conf/httpd.conf
```

И вроде бы мы добились, чего хотели, но в какой момент это выражение из простого превратилось в сложное? Можно ли его запомнить? В принципе, можно. Но это будет одно выражение. А таких выражений может быть сотни и тысячи. Естественно, всё не запомнить и универсального рецепта нет. Если вам по работе часто нужно будет работать с регулярками - запомните выражения и сами научитесь строить, да и заведёте себе заметки.

```
user@rhel8:~$ grep -Ev "^\s*(#|\\$)" /etc/httpd/conf/httpd.conf > ~/httpd.conf
user@rhel8:~$ wc -l httpd.conf
57 httpd.conf
[user@rhel8 ~]$
```

Вывод grep можно направить куда-нибудь в новый файл и будет готовый файл без всяких комментариев. Ну и количество строк в нём будет гораздо меньше, всего 57:

```
grep -Ev "^\s*(#|\\$)" /etc/httpd/conf/httpd.conf > ~/httpd.conf
wc -l httpd.conf
```

С таким файлом работать куда проще.

sed

Другой инструмент, с которым периодически приходится работать - sed. Он позволяет работать с файлами - искать текст, заменять его и прочее, при этом работает из командной строки. Если вам нужно нужно в файле что-то заменить не вручную, т.е. без всяких nano и vi, скажем, через скрипты - то sed чуть ли не единственный вариант. При этом функционал у него почти безграничный. По ссылке вы можете найти кучу примеров его использования.

```
[user@rhel8 ~]$ grep 80 /etc/httpd/conf/httpd.conf
#Listen 12.34.56.78:80
Listen 80
#ServerName www.example.com:80
[user@rhel8 ~]$ sed 's/Listen\ 80/Listen\ 555/g' httpd.conf
ServerRoot "/etc/httpd"
Listen 555
Include conf.modules.d/*.conf
User apache
Group apache
```

Мы же запомним только парочку самых популярных. Для начала - как поменять значение. В файле httpd.conf написано, чтобы вебсервер слушает на 80 порту:

```
grep 80 /etc/httpd/conf/httpd.conf
```

Поменяем стандартный порт на другой, допустим, на 555:

```
sed 's/Listen\ 80/Listen\ 555/g' httpd.conf
```

В таком виде sed просто считывает файл, заменяет значение и выводит на экран, при этом сам файл не изменяется. Я мог бы просто заменить 80 на 555, но может в этом файле и в других местах встречается цифра 80? Чтобы не затронуть лишнее, лучше писать и сам параметр Listen. В этой команде „s“ означает функцию поиска и замены текста, а „g“ - что нужно поменять по всему файлу. Ну и обратите внимание, что пробелы, как и другие специальные символы, надо экранировать.

```
[user@rhel8 ~]$ sudo sed -i 's/Listen\ 80/Listen\ 555/g' /etc/httpd/conf/httpd.conf
[sudo] password for user:
[user@rhel8 ~]$ grep Listen /etc/httpd/conf/httpd.conf
# Listen: Allows you to bind Apache to specific IP addresses and/or
# Change this to Listen on specific IP addresses as shown below to
#Listen 12.34.56.78:80
Listen 555
[user@rhel8 ~]$
```

Окей, в первый раз лучше проверять на тестовом файле или без изменений. Допустим, мы убедились, что команда работает правильно и не перезаписывает ничего лишнего. Как сделать так, чтобы файл всё таки изменился? Нужен ключ -i:

```
sudo sed -i 's/Listen\ 80/Listen\ 555/g' /etc/httpd/conf/httpd.conf
grep Listen /etc/httpd/conf/httpd.conf
```

Как видите, теперь в конфиге вебсервера поменялся порт.

```
[user@rhel8 ~]$ sudo systemctl enable --now httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/httpd.service.
Job for httpd.service failed because the control process exited with error code.
See "systemctl status httpd.service" and "journalctl -xe" for details.
[user@rhel8 ~]$
```

Давайте для теста запустим вебсервер:

```
sudo systemctl enable --now httpd
```

Хоть он и добавился в автозагрузку, но не смог стартануть. Опять же, мы поменяли стандартный порт, а значит дело в selinux.

```
[user@rhel8 ~]$ sudo dnf install setroubleshoot
Updating Subscription Management repositories.
Last metadata expiration check: 1:23:20 ago on Sat 30 Oct 2021 11:31:27 AM MSK.
Dependencies resolved.

Package           Arch      Version       Repository
=====
```

Мы помним, что в логах можно найти подсказку, как исправить проблему. Но чтобы это работало, нужен пакет setroubleshoot, которого может не быть в минимальной системе. Давайте установим его:

```
sudo dnf install setroubleshoot
```

```
[user@rhel8 ~]$ sudo systemctl restart httpd
Job for httpd.service failed because the control process exited with error code.
See "systemctl status httpd.service" and "journalctl -xe" for details.
[user@rhel8 ~]$ sudo journalctl -e | grep semanage
# semanage port -a -t PORT_TYPE -p tcp 555
# semanage port -a -t PORT_TYPE -p tcp 555
# semanage port -a -t PORT_TYPE -p tcp 555
# semanage port -a -t PORT_TYPE -p tcp 555
# semanage port -a -t PORT_TYPE -p tcp 555
```

После установки перезапустим вебсервер и попробуем найти в логах информацию:

```
sudo systemctl restart httpd
```

Можно было и покопаться в выводе journald, но попытаемся найти через grep. Знаем, что команда как-то связана с semanage:

```
sudo journalctl -e | grep semanage
```

И так, команда почти есть, осталось определится с типом порта - PORT_TYPE.

```
[user@rhel8 ~]$ sudo journalctl -e | grep PORT_TYPE
# semanage port -a -t PORT_TYPE -p tcp 555
      where PORT_TYPE is one of the following: http_c
          http_port_t, http_c, jboss_management_port_t, jboss_messaging_port_t, ntop_port_t, puppet_p
          port_t.
```

Ещё раз воспользуемся grep:

```
sudo journalctl -e | grep PORT_TYPE
```

Где мы увидим список вариантов. Из всех описанных больше всего подходит http_port_t, его и используем.

```
[user@rhel8 ~]$ sudo semanage port -a -t http_port_t -p tcp 555
[user@rhel8 ~]$ sudo systemctl restart httpd
[user@rhel8 ~]$ nc -zv localhost 555
Ncat: Version 7.70 ( https://nmap.org/ncat )
Ncat: Connected to ::1:555.
Ncat: 0 bytes sent, 0 bytes received in 0.01 seconds.
[user@rhel8 ~]$ █
```

Используем команду semanage, чтобы разрешить вебсерверу работать на 555 порту:

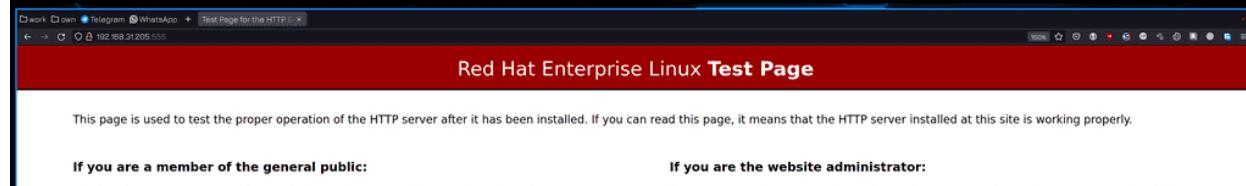
```
sudo semanage port -a -t http_port_t -p tcp 555
```

После чего перезапустим сервис и проверим доступность порта:

```
sudo systemctl restart httpd
nc -zv localhost 555
```

Сервис запустился без ошибок и порт доступен.

```
[user@rhel8 ~]$ sudo firewall-cmd --add-port=555/tcp --permanent
success
[user@rhel8 ~]$ sudo firewall-cmd --reload
success
[user@rhel8 ~]$
[user@rhel8 ~]$ █
```



Остаётся разве что прописать этот порт на файрволе:

```
sudo firewall-cmd --add-port=555/tcp --permanent
sudo firewall-cmd --reload
```

Ну и для теста пропишем в браузере адрес сервера и порт:

```
http://192.168.31.205:555
```

Страница открывается, значит всё правильно.

awk

Ну и напоследок, разберём awk. Он, как и sed, предназначен для обработки текста и даже является отдельным языком, внутри которого можно запускать циклы, условия и т.п.. Как и с sed-ом, у awk-а огромный функционал и большая часть ситуативна. Но одну комбинацию всё же стоит запомнить.

```
[user@rhel8 ~]$ ls -l test/
total 0
-rw-rw-r--. 1 user user 0 Oct 30 10:27 file1
-rw-rw-r--. 1 user user 0 Oct 30 10:27 file2
-rw-rw-r--. 1 user user 0 Oct 30 10:27 file3
[user@rhel8 ~]$ ls -l test/ | cut -d' ' -f9

file1
file2
file3
[user@rhel8 ~]$ █
```

awk очень часто используют для вырезания столбцов, как мы этот делали с cut:

```
ls -l test | cut -d' ' -f9
```

```
[user@rhel8 ~]$ df -h /
Filesystem              Size  Used Avail Use% Mounted on
/dev/mapper/rootvg_rhel8-rootlv  11G  2.5G  7.8G  25% /
[user@rhel8 ~]$ df -h / | cut -d' ' -f1
Filesystem
/dev/mapper/rootvg_rhel8-rootlv
[user@rhel8 ~]$ df -h / | cut -d' ' -f2

[user@rhel8 ~]$ df -h / | cut -d' ' -f3

[user@rhel8 ~]$ df -h / | cut -d' ' -f4
11G
[user@rhel8 ~]$ █
```

Но cut работает только с одним делителем, и если у нас в выводе есть и пробелы, и табуляция, и ещё что-то - то cut справляется плохо:

```
df -h /
df -h / | cut -d' ' -f1
df -h / | cut -d' ' -f2
df -h / | cut -d' ' -f3
df -h / | cut -d' ' -f4
```

Как видите, вместо второго и третьего столбика видим пробелы, а в качестве четвёртого - второй.

```
[user@rhel8 ~]$ df -h /
Filesystem Size Used Avail Use% Mounted on
/dev/mapper/rootvg_rhel8-rootlv 11G 2.5G 7.8G 25% /
[user@rhel8 ~]$ df -h / | awk '{print $1}'
Filesystem
/dev/mapper/rootvg_rhel8-rootlv
[user@rhel8 ~]$ df -h / | awk '{print $2}'
Size
11G
[user@rhel8 ~]$ df -h / | awk '{print $3}'
Used
2.5G
[user@rhel8 ~]$ df -h / | awk '{print $1,$3,$5}'
Filesystem Used Use%
/dev/mapper/rootvg_rhel8-rootlv 2.5G 25%
[user@rhel8 ~]$ █
```

С той же задачей awk справляется лучше, хотя синтаксис у него чуть посложнее:

```
df -h /
df -h / | awk '{print $1}'
df -h / | awk '{print $2}'
df -h / | awk '{print $3}'
df -h / | awk '{print $1,$3,$5}'
```

Но это не значит, что awk всегда лучше. В большинстве случаев cut-a вполне хватает. Но вот такое применение awk стоит запомнить.

И так, сегодня мы с вами разобрали глоббинг и регулярные выражения, а также утилиты grep, sed, awk и в очередной раз разобрали решение проблем с SELinux на примере незнакомого сервиса. У этих инструментов огромный функционал, но кроме пары базовых команд всё остальное очень ситуативно. Поэтому зубрить всё не нужно, просто знайте, что есть такая команда, она делает то-то, и если встанет задача, нагуглить решение будет проще.

2.61.2 Практика

Вопросы

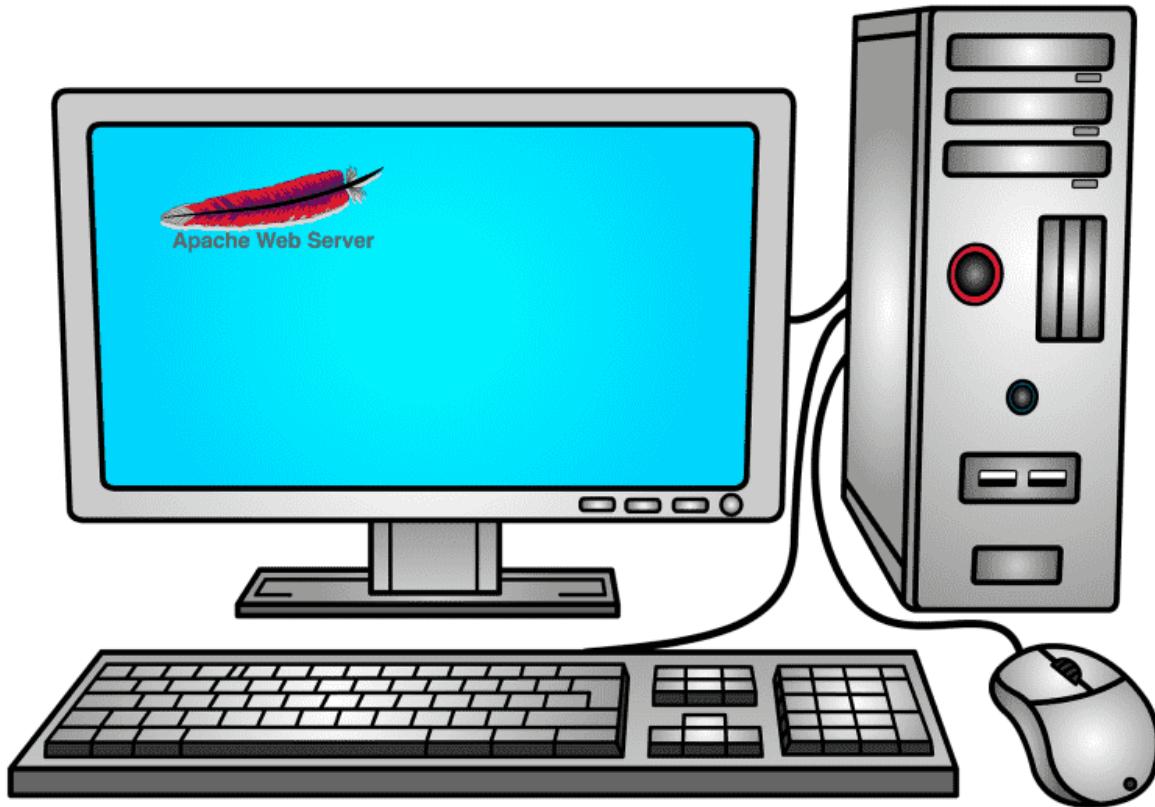
1. В чем отличие глоббинга от регулярных выражений?
2. При работе с sed по умолчанию изменится ли исходный файл?
3. Как прочитать файл конфигурации httpd(apache) без комментариев?
4. Нам понадобилось создать сто папок вида: «backup_server_id сервера от одного до 100» можем ли мы это сделать одной командой?

Задания

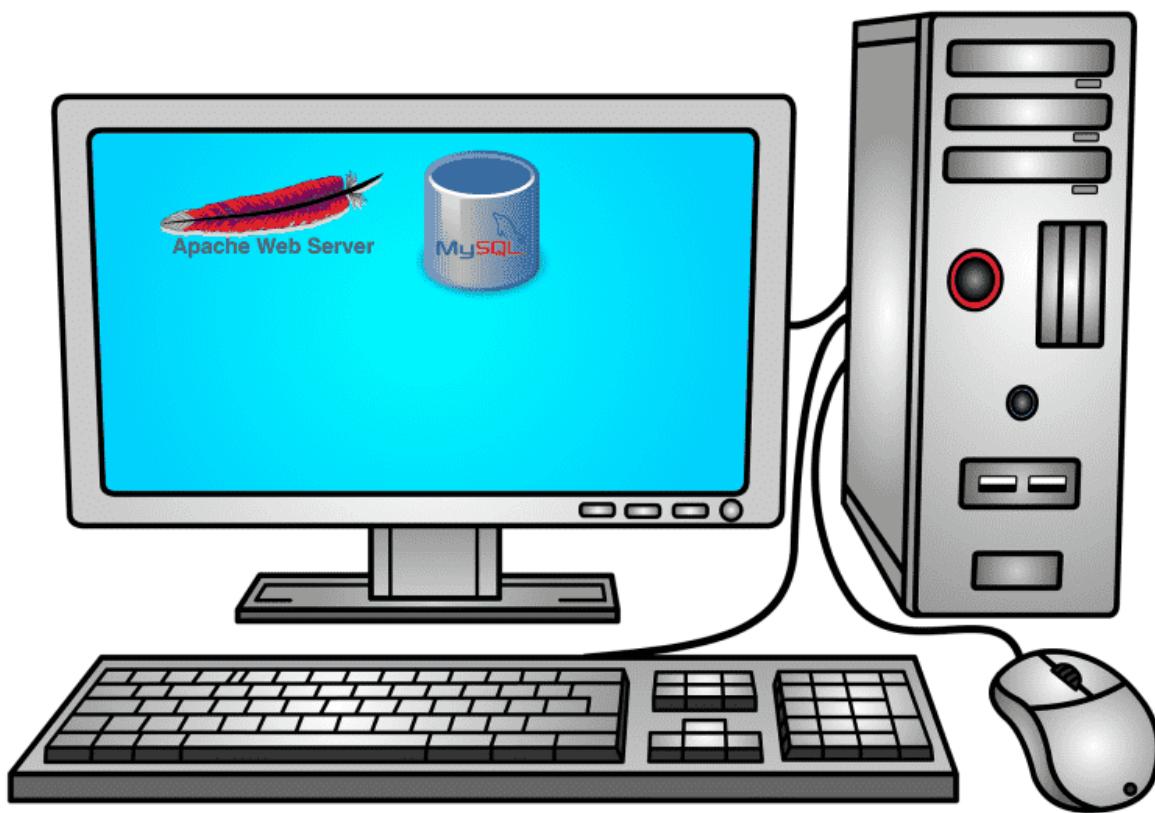
1. Выведите количество tcp сервисов из файла `/etc/services` и подсчитайте их количество

2.62 62. Основы контейнеризации

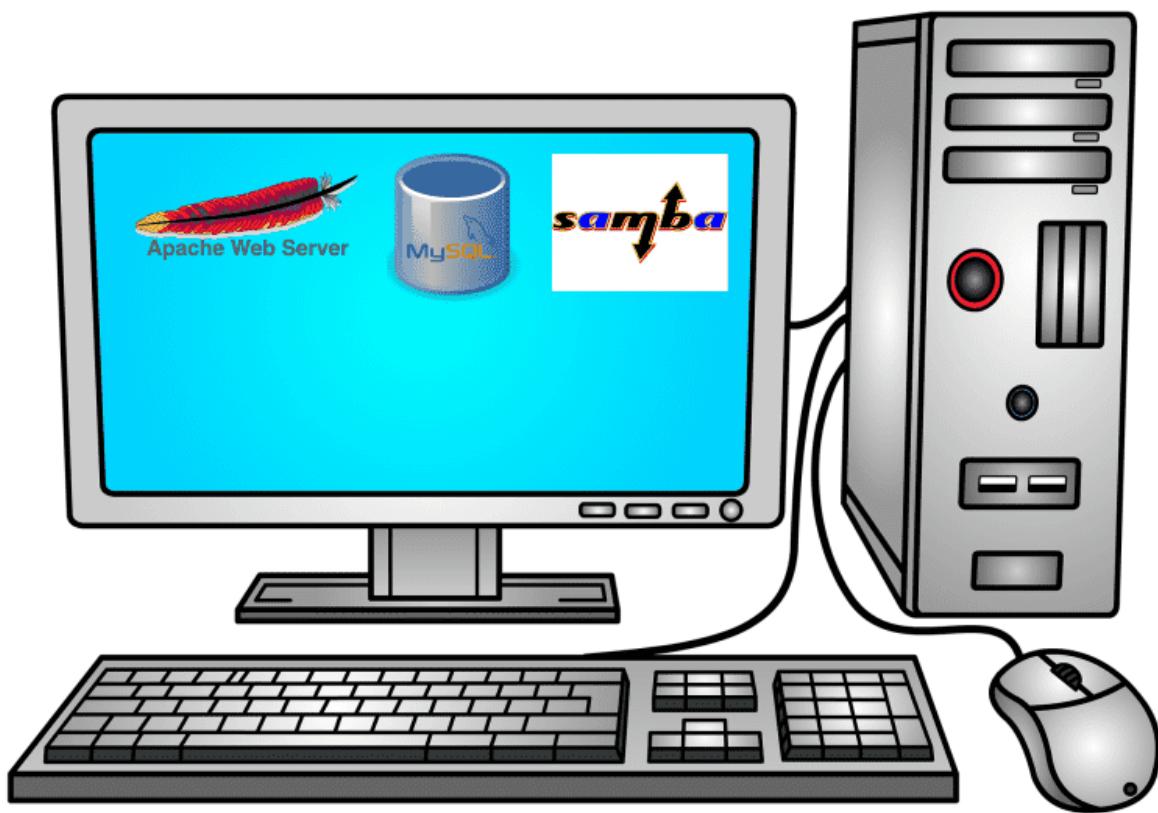
2.62.1 62. Основы контейнеризации



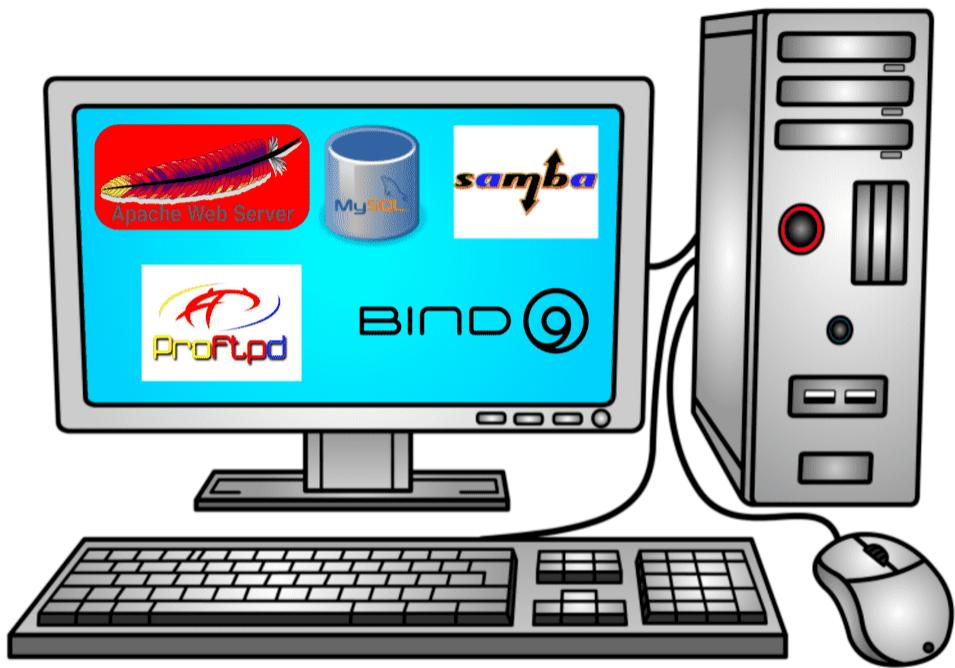
Давайте представим, что вам поручили поднять сайт. Вы берёте один компьютер, ставите на него Linux и поднимаете какое-то приложение, выступающее вебсервером.



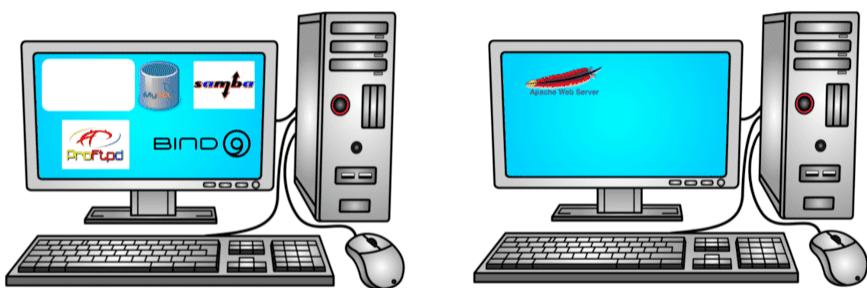
В один из дней этот сайт разрабатывают и вам говорят, что для него теперь нужна база данных. И вы на этом компьютере ставите ещё одно приложение.



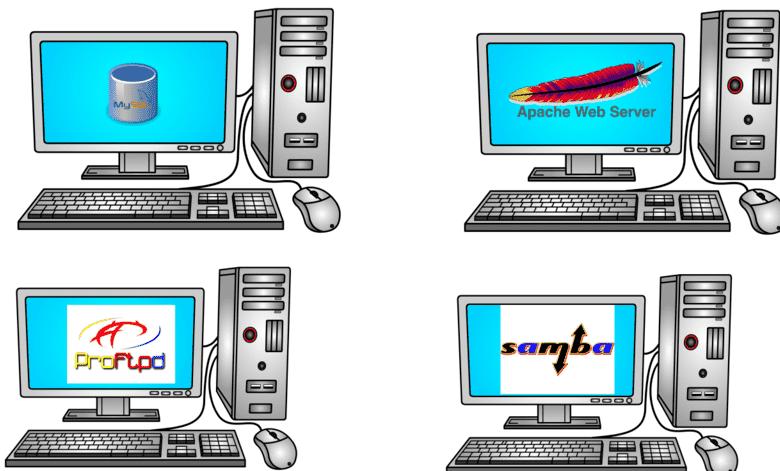
Ещё через пару дней вас просят поднять файловый сервер, чтобы хранить файлы бухгалтерии, данные о клиентах и т.п. И снова вы на этом компьютере поднимаете нужное приложение.



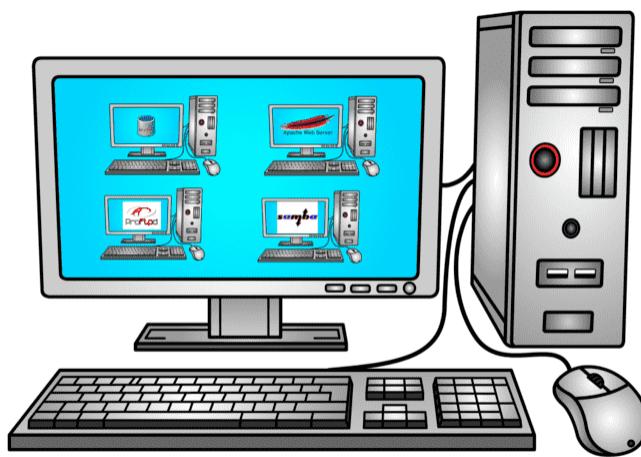
И так продолжается день за днём, вы продолжаете ставить на этот компьютер различный софт. И в один из дней какой-то хакер взламывает ваш вебсервер, получает доступ на сервер и крадёт данные из вашей базы, файлы бухгалтерии и прочую информацию, а напоследок полностью всё стирает.



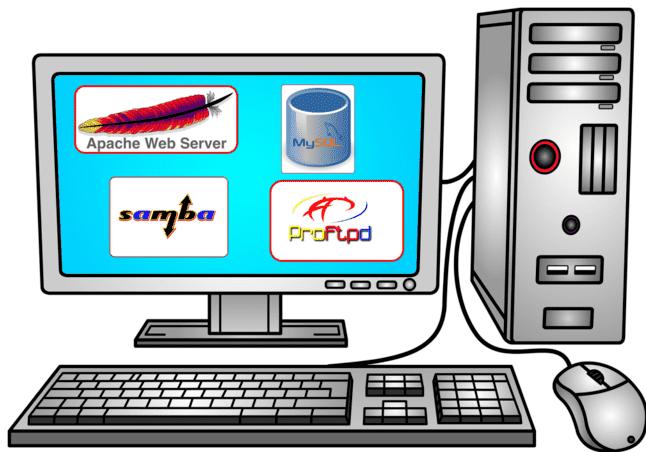
Вы из этого выносите урок и отделяете вебсервер на другой компьютер. И вроде всё нормально, пока вы на нанимаете младшего специалиста, который по ошибке опять всё стирает. А потом, в один из дней, вы ставите программу, которая при установке ломает кучу зависимостей и опять ничего не работает.



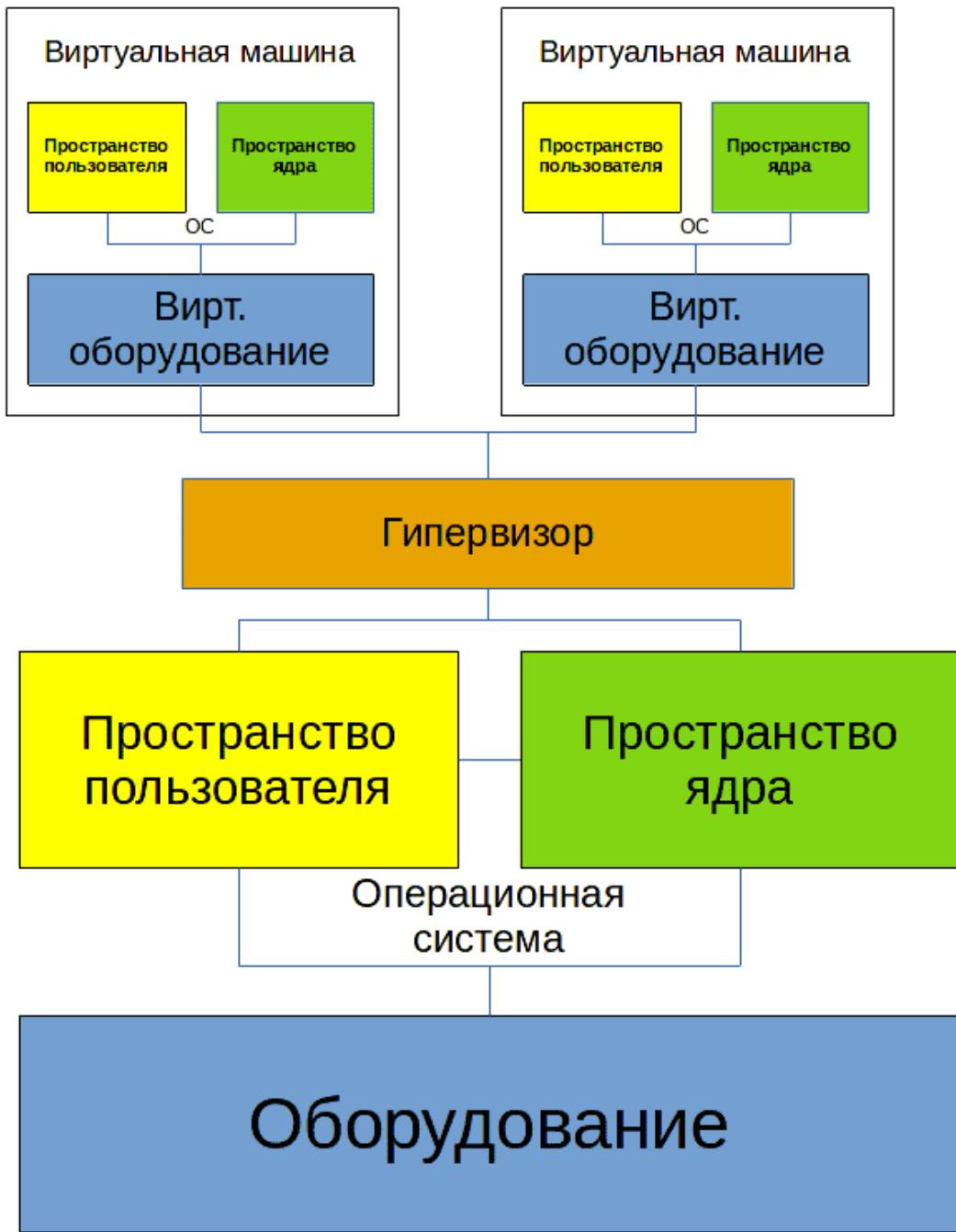
В итоге вы приходите к пониманию, что лучше все приложения держать отдельно, так легче обеспечить их безопасность и смягчить сопутствующий урон, если что-то пойдёт не так. Но для этого вам придётся купить кучу компьютеров, использовать ресурсов каждый компьютер будет не так много, но вот по деньгам это будет очень дорого.



Вы заходите в гугл и узнаёте, что есть такая технология - виртуализация, которая позволяет запускать виртуальные компьютеры внутри реального. И вам теперь не нужно покупать множество компьютеров, всё внутри одного. Но у каждого виртуального компьютера тоже есть виртуальный монитор, виртуальная клавиатура, мышка, виртуальные сетевые адаптеры и кучу прочего оборудования, которые нужно эмулировать. Кроме того, у каждого виртуального компьютера есть своя полноценная операционная система и куча программ и файлов. А значит каждую из этих операционных систем нужно настраивать, обеспечивать безопасность, следить за объёмом диска и т.д. и т.п. Вроде бы все приложения у вас изолированы друг от друга, но вам из-за этого придётся потратиться на довольно дорогой компьютер.



А можно ли как-то изолировать приложения без необходимости поднимать виртуальные компьютеры со всякими устройствами и операционными системами? Да, и это делается через контейнеризацию. Чтобы понять контейнеризацию, для начала абстрактно представим виртуализацию.



Представим компьютер - т.е. оборудование. На него ставится операционная система, которую можно поделить на 2 составляющие:

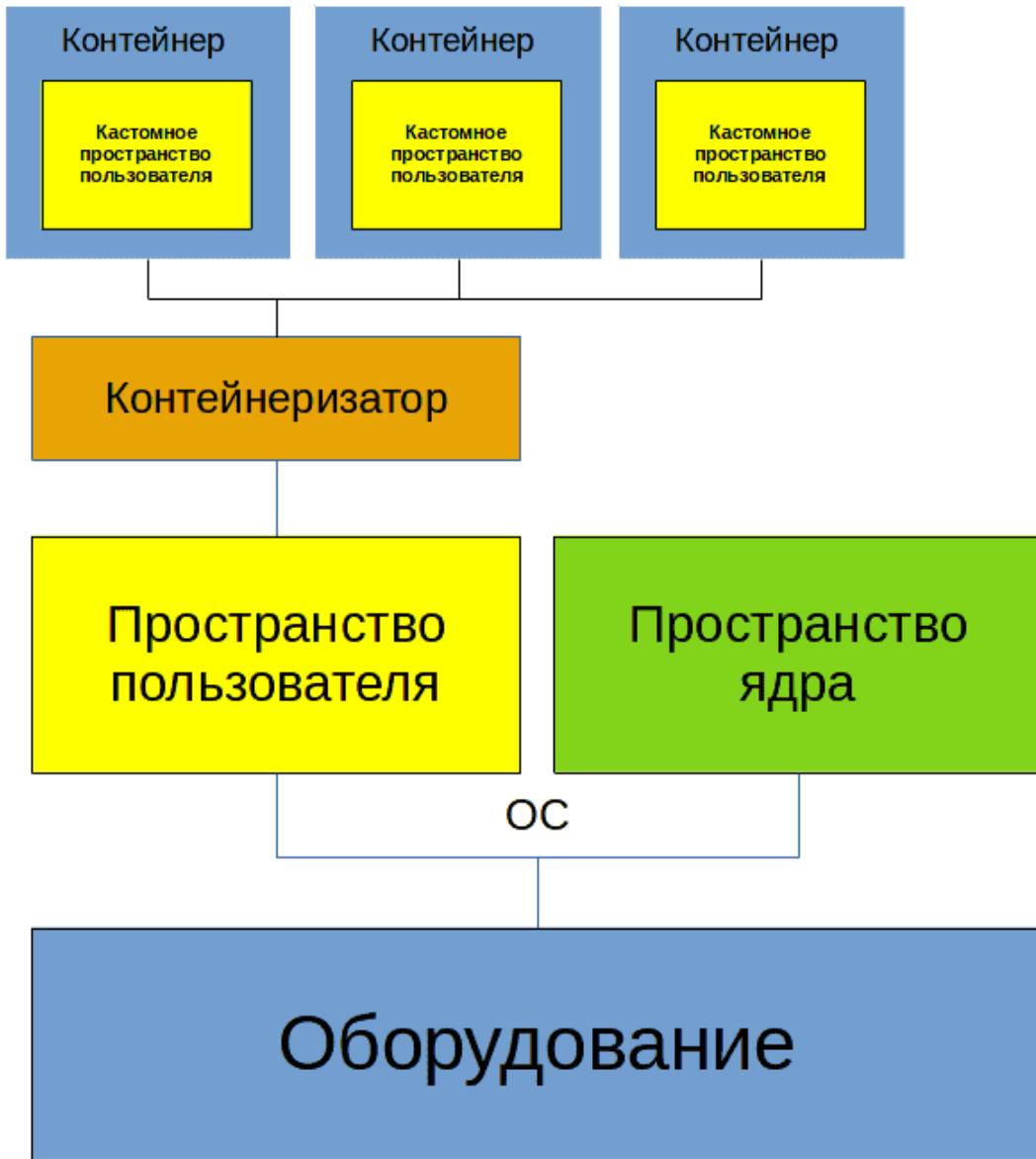
- В пространстве ядра работает ядро операционной системы и его модули. Эта часть отвечает за

управление процессами, памятью, драйвера и т.п.

- Другая часть ядра - пространство пользователя. Тут уже работают все наши программы, включая систему инициализации systemd, оболочку, системные и пользовательские приложения.

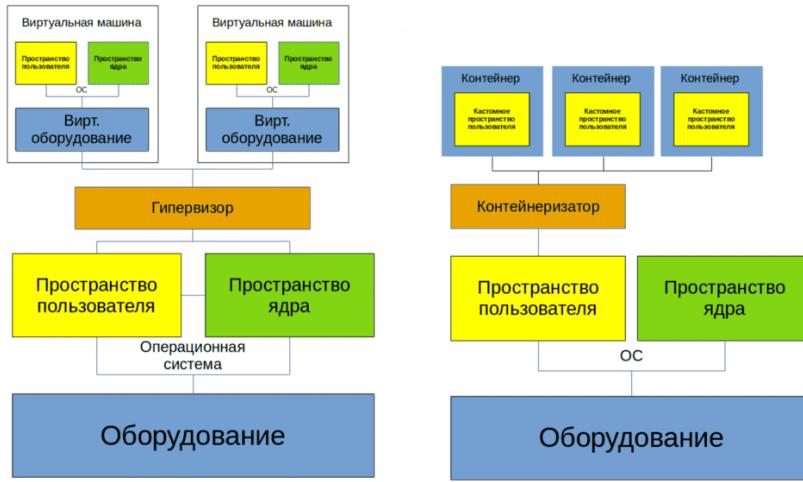
Для виртуализации ставится специальная программа - гипервизор, к примеру, наш VirtualBox. Он работает и в пространстве ядра в качестве модуля, и в пространстве пользователя в качестве софта для эмуляции оборудования и управления виртуальными машинами. Говоря про эмуляцию оборудования, я в основном говорю про периферию, т.е. процессор и оперативка обычно не эмулируются, а используются напрямую.

Так вот, с помощью гипервизора мы можем создавать виртуальные компьютеры, т.е. виртуальное железо и ставить поверх этого виртуального оборудования полноценную операционную систему, которая также состоит из пространства пользователя и пространства ядра. Вот так работает виртуализация, довольно ресурсоёмко, но зато можно внутри одного компьютера поднять несколько операционных систем.



Теперь рассмотрим контейнеризацию. Всё также мы поверх оборудования ставим операционную систему, но теперь вместо гипервизора используем другой софт - контейнеризатор. Если при виртуализации нам нужно было виртуальное железо, чтобы на нём запустить полноценную операционную систему, то в контейнерах нет полноценной операционной системы. Пространство ядра общее для всех контейнеров и основной операционной системы, отличается только пространство пользователя.

При этом, если я хочу запустить вебсервер в контейнере, нужны ли мне systemd и куча другого софта, которые используются в полноценной системе? Да нет, хватит каких-то основных библиотек, какого-то базового софта и всё, а вебсервер можно хоть скриптом запускать. В другом контейнере, где будет крутиться другой софт, будут только те файлы, которые нужны для работы того софта. Т.е. получается мы изолировали программу и обеспечили её зависимостями, это и есть наш контейнер.



И какие выводы можно из этого сделать?

Виртуализация эмулирует железо и ставит полноценную операционную систему, где изначально куча софта, а значит используется много ресурсов. При этом нужно следить за каждой виртуалкой, обеспечивая её безопасность всячими антивирусами, файрволами и прочим, а также следить за использованием диска и прочих ресурсов. Зато здесь можно поставить любую операционную систему и делать почти всё что угодно.

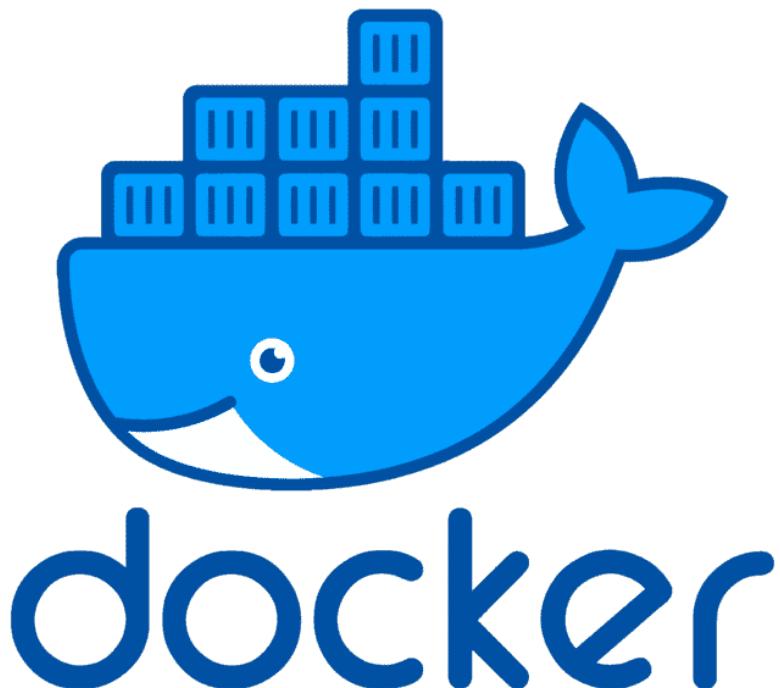
Контейнеризация, грубо говоря, просто изолирует процессы. Работает программа в своём пространстве и видит только то что ей нужно. При этом контейнеры используют общий kernel space. Т.е. если программа создана под линукс, то в своём контейнере она будет работать на любом дистрибутиве. А вот если программа создана под Windows, то контейнер с ней будет работать только на Windows.

Поэтому современная инфраструктура это смесь виртуализации и контейнеризации. Причём, нередко система с контейнерами крутится внутри виртуалки.

Mechanism	Operating system	License	Actively developed since or between	Features									
				File system isolation	Copy on Write	Disk quotas	I/O rate limiting	Memory limits	CPU quotas	Network isolation	Nested virtualization	Partition checkpointing and live migration	Root privilege isolation
chroot	Most UNIX-like operating systems	Varies by operating system	1982	Partial ^[a]	No	No	No	No	No	No	Yes	No	No
Docker	Linux ^[8] , FreeBSD ^[9] , Windows x64 (Pro, Enterprise and Education) ^[10] , macOS ^[11]	Apache License 2.0	2013	Yes	Yes	Not directly	Yes (since 1.10)	Yes	Yes	Yes	Yes	Only in Experimental Mode with CRU ^[11]	Yes (since 1.10)
Linux-VServer (security context)	Linux, Windows Server 2016	GNU GPLv2	2001	Yes	Yes	Yes	Yes ^[b]	Yes	Yes	Partial ^[c]	?	No	Partial ^[d]
lxcmtf	Linux	Apache License 2.0	2013-2015	Yes	Yes	Yes	Yes ^[b]	Yes	Yes	Partial ^[e]	?	No	Partial ^[d]
LXC	Linux	GNU GPLv2	2008	Yes ^[13]	Yes	Partial ^[e]	Partial ^[f]	Yes	Yes	Yes	Yes	Yes	Yes ^[13]
Singularity	Linux	BSD Licence	2015 ^[14]	Yes ^[15]	Yes	Yes	No	No	No	No	No	No	Yes ^[16]
OpenVZ	Linux	GNU GPLv2	2005	Yes	Yes ^[17]	Yes	Yes ^[g]	Yes	Yes	Yes ^[h]	Partial ^[i]	Yes	Yes ^[i]
Virtuozzo	Linux, Windows	Thalware	2000 ^[21]	Yes	Yes	Yes	Yes ^[k]	Yes	Yes	Yes ^[h]	Partial ^[j]	Yes	Yes
Solaris Containers (Zones)	illumos (OpenSolaris), Solaris	CDDL, Proprietary	2004	Yes	Yes (ZFS)	Yes	Partial ^[m]	Yes	Yes	Yes ^{[n][24][25]}	Partial ^[o]	Partial ^{[p][q]}	Yes ^[r]
FreeBSD jail	FreeBSD, DragonFly BSD	BSD License	2000 ^[27]	Yes	Yes (ZFS)	Yes ^[s]	Yes	Yes ^[28]	Yes	Yes ^[29]	Yes	Partial ^{[30][31]}	Yes ^[32]
vkernel	DragonFly BSD	BSD Licence	2006 ^[33]	Yes ^[34]	Yes ^[34]	N/A	?	Yes ^[35]	Yes ^[35]	Yes ^[36]	?	?	Yes
sysjail	OpenBSD, NetBSD	BSD License	2006-2009	Yes	No	No	No	No	No	Yes	No	No	?
WPARs	AIX	Commercial proprietary software	2007	Yes	No	Yes	Yes	Yes	Yes	Yes ^[t]	No	Yes ^[38]	?
iCore Virtual Accounts	Windows XP	Freeware	2008	Yes	No	Yes	No	No	No	No	?	No	?
Sandboxie	Windows	GNU GPLv3	2004	Yes	Yes	Partial	No	No	No	Partial	No	No	Yes
systemd-nspawn	Linux	GNU LGPLv2.1+	2010	Yes	Yes	Yes ^{[39][40]}	Yes	?	?				
Turbo	Windows	Freemium	2012	Yes	No	No	No	No	No	Yes	No	No	Yes
rkt	Linux	Apache License 2.0	2014 ^[41] -2018	Yes	Yes	Yes	Yes	Yes	Yes	Yes	?	?	Yes

Если говорить про контейнеризацию, то идея изолировать процессы и директории, в которых они работают, существует давно. Ещё во времена UNIX-а появился chroot, который запирал процессы в директории и они думали, что корнем является эта директория. Есть cgroups, с помощью которого можно ограничивать процессам ресурсы, также есть namespaces для изоляции процессов, сетевых устройств и прочих ресурсов операционной системы. Можете прочитать чуть подробнее в [статье](#) на хабре.

Но был нюанс - зачастую, чтобы настроить изоляцию, приходилось прочесть много документации и много повозиться. Это отпугивало многих и они предпочитали не заморачиваться с изоляцией. Поэтому нужен был простой и удобный инструмент для изоляции.



И им стал Docker. Как для виртуализации одной из программ является VirtualBox, так и для контейнеризации одной из программ является Docker. Разве что Docker является самым популярным решением. Во-первых, потому что он предоставляет удобный функционал для работы с контейнерами.

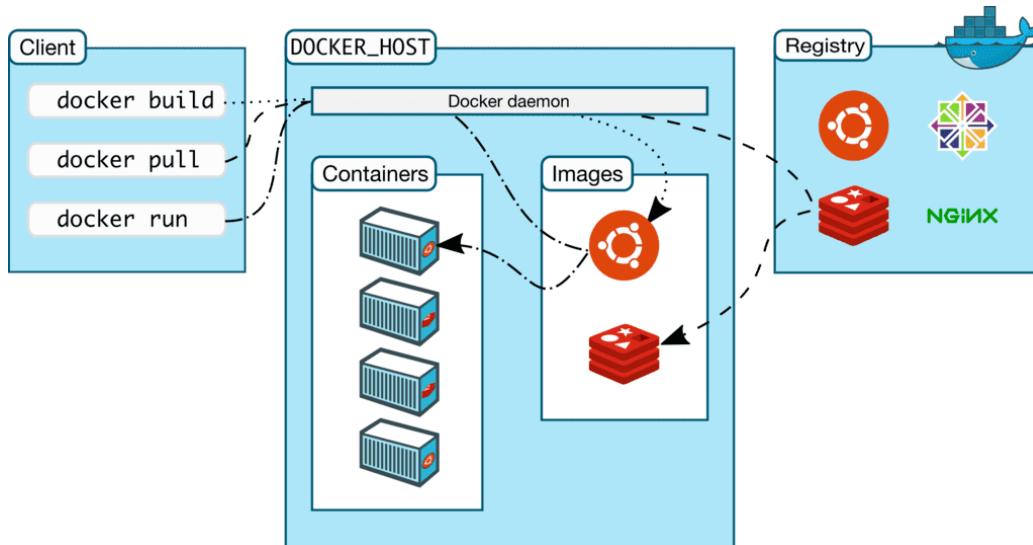
The screenshot shows the Docker Hub search interface at <https://hub.docker.com/search?type=image>. The search bar contains "Search for great content (e.g., mysql)". The top navigation bar includes "Explore", "Pricing", "Sign In", and "Sign Up". Below the search bar, there are tabs for "Docker", "Containers", and "Plugins", with "Containers" being active. On the left, there are filters for "Images" (with options for "Verified Publisher" and "Official Images") and "Categories" (with options for "Analytics", "Application Frameworks", "Application Infrastructure", and "Application Services"). The main search results area displays two entries:

- couchbase** (Official Image) - Updated 23 days ago. Description: Couchbase Server is a NoSQL document database with a distributed architecture. Tags: Container, Linux, x86-64, Storage, Application Frameworks. Downloads: 50M+, Stars: 806.
- alpine** (Official Image) - Updated 2 months ago. Description: Alpine Linux is a small, fast, and secure Linux distribution based on musl libc and busybox. Tags: Container, Linux, x86-64, Storage, Application Frameworks. Downloads: 1B+, Stars: 8.1K.

Во-вторых, потому что [dockerhub](#). Помните, мы говорили про кастомные пространства пользователей

в контейнерах? С другими типами изоляции вам приходилось их вручную собирать, а докер позволяет скачать готовые образы одной командой. Захотели вебсервер? Хватит одной команды и ничего не нужно настраивать, плюс к тому же это изолировано. Хотите связку из вебсервера, прокси сервера и ещё кучи каких-то других приложений? Берёте небольшой файлик с парочкой опций и всё готово. Вручную вы бы это настроили за пару дней, мучаясь с документацией и непонятными проблемами, а тут буквально пару минут и всё работает. А если что-то упадёт или даже всё? Восстановить какой-то сервис или полностью инфраструктуру на виртуалках может занять дни, недели, а то и месяцы. А если всё на контейнерах, то почти всё можно поднять автоматом за пару часов.

Конечно, звучит здорово, но с контейнерами нужно работать несколько иначе, чем с виртуалками, т.е. нужны другие методологии и практики. Например, контейнеры рассчитаны на то, что данные там не будут храниться и что в любой момент вы можете удалить контейнер и заново поднять. Т.е. в контейнере крутится только софт, и если с ним что-то случится - баг, или взлом, или любая другая проблема - вы просто удаляете этот контейнер и поднимаете новый, ни о чём не задумываясь. С виртуалкой так делать нельзя, потому что внутри виртуалки данные - придётся сначала их забрать оттуда. А к контейнерам вы изначально цепляете данные отдельно, как, скажем, к виртуалке цепляете отдельный диск только для данных. Такой подход называется stateless - т.е. без сохранения состояния. Упал контейнер - и фиг с ним, подцепил данные к другому и запустил. Потому что иначе вас ждут проблемы - в этом кастомном пространстве пользователя почти нет всяких утилит, которые бы помогли вам восстановить работу. Вплоть до того, что в контейнере может не быть текстового редактора, баша и прочих прелестей обычных систем.



Но вы можете создать свой образ и положить туда нужные вам инструменты, либо изменить существующий образ. Давайте определимся с некоторыми терминами.

- Image - образ системы, т.е. шаблон. Грубо говоря, это кастомный userspace с какими-то готовыми программами, библиотеками и прочим.
- Dockerfile - вы можете написать текстовой файл, в котором указываете инструкцию по сборке. И запустив команду с этим файлом, вы получите образ - image. Опять же, это не обязательно и вы можете использовать готовые образы.
- Контейнер - запущенный образ. С одного образа вы можете хоть 100 контейнеров запустить. При этом каждый контейнер будет по своему уникален. Как в поговорке - «Нельзя дважды войти в одну и ту же реку». Если вы запустили образ - то там уже запустились процессы, написались логи, какие-то события произошли. Т.е. что-то поменялось и уже это нельзя назвать образом или шаблоном. Но, если вы захотите, сможете из контейнера сделать образ.
- Registry - репозиторий - место, где держат образы. Как тот же самый докерхаб. Но вы можете

поднять внутри компании свой репозиторий для своих кастомных образов.

- Persistent storage - постоянное хранилище - то что цепляется к контейнеру для постоянного хранения данных. Если вы удалите контейнер, эти данные останутся и вы сможете прицепить хранилище к другим контейнерам.

The screenshot shows a web browser displaying the Docker documentation at <https://docs.docker.com/engine/install/rhel/>. The main content is titled "Install Docker Engine on RHEL". It features a "Prerequisites" section with a note about RHEL 8 support and an "OS requirements" section stating that RHEL 7 or 8 is required. On the left, there's a sidebar with links for "Install on Fedora", "Install on RHEL" (which is selected), "Install on SLES", "Install on Ubuntu", "Install binaries", "Optional post-installation steps", "Release notes", "Previous versions", "Deprecated features", "Docker Buildx", "Docker Context", and "Docker Scan".

Касательно установки докера, всегда рекомендую предварительно смотреть документацию на официальном [сайте](#). Обычно всё сводится к добавлению официального репозитория и установке пакетов из него. При этом ставится бесплатная версия докера. Как вы поняли, есть и платная, коммерчески поддерживаемая версия. Но на RHEL docker не поставить, хотя на Centos можно.

- The `docker` package is not shipped or supported by Red Hat for Red Hat Enterprise Linux (RHEL) 8. The `docker` container engine is replaced by a suite of tools in the Container Tools module.
- The `podman` container engine replaced `docker` as the preferred, maintained, and supported container runtime of choice for Red Hat Enterprise Linux 8 systems. Podman provides a `docker` compatible command line experience enabling users to find, run, build, and share containers. Podman uses Buildah and Skopeo as libraries for the build and push.
- The `buildah` utility replaced `docker build` as the preferred, maintained, and supported container images build tool in Red Hat Enterprise Linux 8 systems.
- The `skopeo` utility replaced `docker push` as the preferred, maintained and supported utility for moving container images between registries, and container engines.
- For an overview of all of these tools, please see: [Building, running and managing containers - Chapter 1. Starting with containers](#)
- For a deeper dive on `podman` specifically, please see: [Building, running and managing containers - Chapter 7. Container command line reference](#)

Для своего дистрибутива Red Hat заменили `docker` на `podman`, работать с которым мы научимся в следующем уроке. `Podman`, как и `docker`, сделан по определённым стандартам, поэтому контейнеры, которые работают в докере, будут работать и в подмане. Даже большинство команд докера будут работать в подмане. Есть архитектурные и функциональные отличия, которые мы затронем в следующем

уроке.

Вообще, тема контейнеризации и, в частности, докера, слишком обширная, чтобы рассмотреть в рамках одного, двух или даже десяти уроков. Поэтому в данном курсе я рассмотрю то что касается экзамена, а если вам интересно, в интернете есть множество бесплатных курсов по докеру. Но к теме докера я и сам вернусь в будущих курсах.

2.62.2 Практика

Вопросы

1. Какие преимущества есть у контейнеризации?
2. Чем отличается виртуализация от контейнеризации?
3. Чем „Image“ отличается от контейнера?
4. Что такое Registry? Можно ли создать свой Registry?
5. Как подключить настоящие данные к контейнеру?

Задания

2.63 63. Работа с podman

2.63.1 63. Работа с podman

The screenshot shows a web browser window with the URL <https://podman.io/getting-started/installation>. The page is titled "RHEL7". It contains instructions for installing Podman on RHEL 7, which states: "Subscribe, then enable Extras channel and install Podman." Below this, there is a code block:

```
sudo subscription-manager repos --enable=rhel-7-server-extras-rpms  
sudo yum -y install podman
```

Below the RHEL 7 section is another titled "RHEL8". It states: "Podman is included in the container-tools module, along with Buildah and Skopeo." Below this, there is another code block:

```
sudo yum module enable -y container-tools:rhel8  
sudo yum module install -y container-tools:rhel8
```

At the bottom of the RHEL8 section, there is a note: "The container-tools:rhel8 is the fast application stream, containing most recent rolling versions of the tools. Use the container-tools:2.0 stream for stable versions of Podman 1.6. The command yum module list container-tools shows the available streams."

От теории к практике. Давайте установим podman. Мы бы могли в репозиториях поискать нужный пакет, но так стоит делать только с простыми утилитами. Если дело касается чуть более сложных программ, всегда стоит проверять документацию на официальном сайте этой программы. Поэтому идём на сайт podman.io и находим инструкцию по установке на наш RHEL. Как видите, здесь указано, что нужно включить модуль container-tools и установить его. Напомню, что модули - это группы пакетов, нацеленных на решение одной задачи. Как и в нашем случае - для работы с контейнерами нужны несколько различных инструментов.

```
[user@rhel8 ~]$ sudo dnf module list | grep cont
[sudo] password for user:
container-tools      rhel8 [d] ●      common [d]                                Most recent (rolling) versions of podman, buildah, skopeo, runc, common, runc, common, CRIU, Udev, etc as well as dependencies such as container-selinux built and tested together, and updated as frequently as every 12 weeks.
container-tools      1.0 ●      common [d]                                Stable versions of podman 1.0, buildah 1.5, skopeo 0.1, runc, common, CRIU, Udev, etc as well as dependencies such as container-selinux built and tested together, and supported for 24 months.
container-tools      2.0 ●      common [d]                                Stable versions of podman 1.6, buildah 1.11, skopeo 0.1, runc, common, etc as well as dependencies such as container-selinux built and tested together, and supported as documented on the Application Stream life cycle page.
container-tools      3.0 ●      common [d]                                Stable versions of podman 3.0, buildah 1.19, skopeo 1.2, runc, common, etc as well as dependencies such as container-selinux built and tested together, and supported as documented on the Application Stream life cycle page.
[user@rhel8 ~]$
```

Давайте найдём этот модуль через dnf:

```
sudo dnf module list | grep cont
```

Можно увидеть 4 версии этого модуля, в которых отличаются версии пакетов, где-то podman версии 1.0, где-то 1.6, а где-то 3, ну и также для других связанных инструментов. Документация на сайте советовала нам ставить первый модуль, который обновляется каждые двенадцать недель.

```
[user@rhel8 ~]$ sudo dnf module enable container-tools:rhel8 -y
[sudo] password for user:
Updating Subscription Management repositories.
Last metadata expiration check: 0:31:32 ago on Sat 13 Nov 2021 10:01:41
Dependencies resolved.
=====
Package           Architecture      Version       Rep
=====
Enabling module streams:
  container-tools                         rhel8

Transaction Summary
=====

Complete!
[user@rhel8 ~]$ sudo dnf module install container-tools:rhel8 -y
Updating Subscription Management repositories.
```

В некоторых случаях бывает нужно использовать старую версию, если, допустим, что-то в новой у нас не работает, но для обучения нам сойдёт и последняя версия. Включим модуль с ней и установим его:

```
sudo dnf module install container-tools:rhel8 -y
```

Кстати, вместе с подманом и другими утилитами ставится и модуль для cockpit, т.е. контейнерами можно будет управлять и через веб интерфейс.

Касательно отличий докера и подмана, их не так много, но кое-что есть.

The screenshot shows a browser window with the URL <https://docs.docker.com/config/containers/live-restore/>. The page is part of the Docker documentation under the 'Guides' section. The main title is 'Keep containers alive during daemon downtime'. A sidebar on the left lists various configuration topics, with 'Keep containers alive during daemon downtime' being the selected item. A note in the main content area states: 'Live restore is not supported on Windows containers, but it does work for Linux containers running on Docker Desktop for Windows.'

- Во-первых, для работы докера нужен демон. Где-то это называют единой точкой отказа, т.е. если с демоном что-то случится, он выключит контейнеры. Можно настроить, чтобы контейнеры работали даже если докер демон перестанет работать, но в любом случае управлять ими нормально не получится и в целом может привести к проблемам. А вот podman работает без демона, поэтому такой проблемы можно избежать. Не то, чтобы это решающий фактор отказаться от докера, скорее небольшой плюсик в сторону подмана.



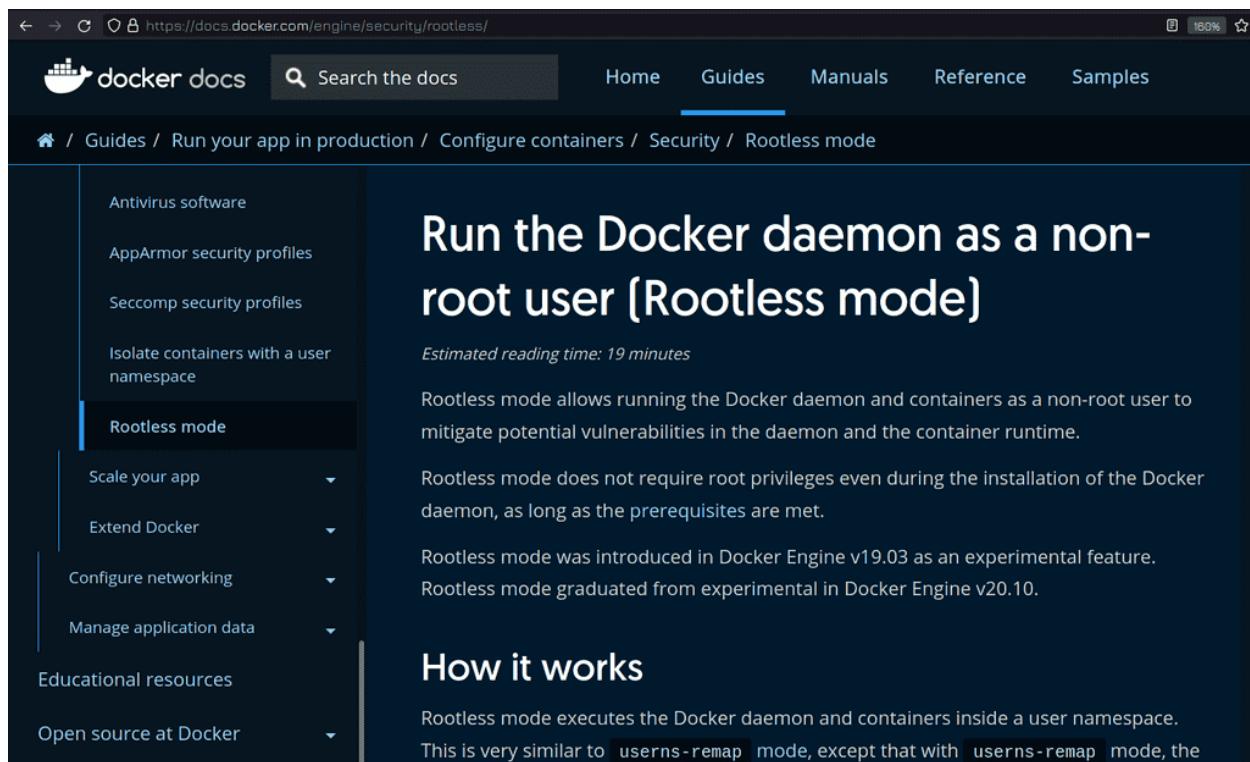
- Во-вторых, podman умеет работать с подами. Собственно, поэтому он так и назван - pod manager. Что такое поды? Хотя мы и говорим, что контейнеризация нужна для изоляции приложений, иногда они должны общаться. Скажем, вебсервера очень часто должны работать с базой данных, а она работает в другом контейнере. Так же нередко для вебсерверов ставят прокси сервера. И хотя получается 3 разных приложения, служат они для одной задачи. Когда у вас приложений много, становится сложно управлять всеми контейнерами по отдельности, и появляется понятие под - эдакая группа контейнеров. Поды позволяют связать контейнеры и управлять ими как единым целым. В чистом виде докер не используется в больших инфраструктурах, обычно там работает кубернетс или подобный софт, который вносит понятие подов и централизовано управляет всем. Но кубернетс большой и сложный продукт, который нужно изучать и не так легко администрировать, как докер, поэтому не всегда подходит для мелких и средних компаний. А podman такой же простой, как и докер, при этом ещё и поддерживает.

New Generation of Container Management Tools

Buildah builds, **Podman** runs, and **Skopeo** transfers **container images**.

These open-source tools are maintained by the **containers** organization on Github. There is some overlap in functionality between Buildah and Podman, but the separation of core responsibilities is clear. None of these tools require a running daemon, and for the most part, don't need root access either, with a few exceptions. So, having set the stage, I'll spend the

- В-третьих, docker - один инструмент с кучей функционала. Разработчики podman-а, в свою очередь, пытались соответствовать философии UNIX - пишите программы, которые делают что-то одно и делают это хорошо. Поэтому функционал докера в подмане разнесён по разным инструментам: для сборки образа используется утилита buildah, а для работы с репозиториями и образами в них - скопео, сам же podman в основном отвечает за управление контейнерами.



The screenshot shows a web browser displaying the Docker documentation at <https://docs.docker.com/engine/security/rootless/>. The page title is "Run the Docker daemon as a non-root user (Rootless mode)". The left sidebar has a "Guides" section with several items, and "Rootless mode" is highlighted with a blue bar. The main content area starts with a sub-section titled "Antivirus software". Below the title, it says "Estimated reading time: 19 minutes". The text explains that Rootless mode allows running the Docker daemon and containers as a non-root user to mitigate potential vulnerabilities. It notes that Rootless mode was introduced in Docker Engine v19.03 and graduated from experimental in v20.10. A "How it works" section follows, explaining that Rootless mode executes the Docker daemon and containers inside a user namespace, similar to usersns-remap mode but with different behavior.

Хотя практически везде пишут, что podman работает без рут прав, в отличии от докера, но и докер тоже можно настроить, чтобы он работал не от рута. Правда некоторые путают это с добавлением пользователя в группу docker, но это разные вещи. Если добавить пользователя в группу докер, он может управлять контейнерами без sudo прав, но сами контейнеры всё равно будут работать от рута. А для настройки работы докер демона без рут прав нужно кое-что перенастраивать. Информацию об этом можете посмотреть по [ссылке](#). Но, справедливости ради, докер надо перенастраивать, а podman так работает из коробки.

```
[user@rhel8 ~]$ podman
attach      (Attach to a running container)
auto-update (Auto update containers according to their auto-update policy)
build       (Build an image using instructions from Containerfiles)
commit      (Create new image based on the changed container)
container   (Manage containers)
cp          (Copy files/folders between a container and the local filesystem)
create     (Create but do not start a container)
diff        (Display the changes to the object's file system)
events     (Show podman events)
exec       (Run a process in a running container)
export     (Export container's filesystem contents as a tar archive)
generate   (Generate structured data based on containers, pods or volumes.)
healthcheck (Manage health checks on containers)
help       (Help about any command)
history    (Show history of a specified image)
image      (Manage images)
images     (List images in local storage)
import    (Import a tarball to create a filesystem image)
info       (Display podman system information)
```

Ладно, давайте теперь займёмся делом. У команды podman множество ключей, если написать podman и два раза нажать tab, можно увидеть список команд и описание к ним:

```
podman
```

Работа с контейнерами напоминает что-то среднее между управлением пакетами с помощью dnf и управления виртуалками.

```
[user@rhel8 ~]$ podman search ssh
INDEX      NAME                                     DESCRIPTION
STARS      OFFICIAL      AUTOMATED
redhat.com  registry.access.redhat.com/rhmap45/gitlab-shell  RHMAP image that provides GitLab Shell, an a... 0
redhat.com  registry.access.redhat.com/rhmap46/gitlab-shell  RHMAP image that provides GitLab Shell, an a... 0
redhat.com  registry.access.redhat.com/rhmap47/gitlab-shell  RHMAP image that provides GitLab Shell, an a... 0
redhat.com  registry.access.redhat.com/rhmap44/gitlab-shell  RHMAP Docker container that provides GitLab ... 0
redhat.com  registry.access.redhat.com/rhmap43/gitlab-shell  RHMAP Docker container that provides GitLab ... 0
redhat.com  registry.access.redhat.com/rhmap42/gitlab-shell  RHMAP Docker container that provides GitLab ... 0
centos.org   registry.centos.org/centos/ssh           0
docker.io   docker.io/jenkinsci/ssh-slave          [OK]
docker.io   docker.io/kroniak/ssh-client            [OK]
docker.io   docker.io/with/ssh...                   38
```

Например, можно искать контейнеры с помощью опции search:

```
podman search ssh
```

При этом, как видите, используются различные репозитории и ссылки на образы.

← → C ⌂ https://hub.docker.com/r/jenkinsci/ssh-slave/

A Jenkins agent using SSH to establish connection.

See [Jenkins Distributed builds](#) for more info.

Running

To run a Docker container

```
docker run jenkins/ssh-slave "<public key>"
```

You'll then be able to connect this agent using the [SSH Build Agents plugin](#) as "jenkins" with the matching private key.

How to use this image with Docker Plugin

To use this image with [Docker Plugin](#), you need to pass the public SSH key using environment variable `JENKINS_SLAVE_SSH_PUBKEY` and not as a startup argument.

In *Environment* field of the Docker Template (advanced section), just add:

```
JENKINS_SLAVE_SSH_PUBKEY=<YOUR PUBLIC SSH KEY HERE>
```

По названию образа в интернете можно найти полезную информацию к нему, например, какие переменные можно передавать образу, с какими ключами стоит использовать и т.п.

```
[user@rhel8 ~]$ grep registries /etc/containers/
certs.d/          policy.json      registries.conf.d/ storage.conf
oci/              registries.conf   registries.d/    toolbox.conf
[user@rhel8 ~]$ grep registries /etc/containers/registries.conf
# For more information on this configuration file, see containers-registries.conf(5).
# intended content. We recommend only adding registries which are completely
# trusted (i.e., registries which don't allow unknown or anonymous users to
# of these registries, it should be added at the end of the list.
# # An array of host[:port] registries to try when pulling an unqualified image, in order.
unqualified-search-registries = ["registry.fedoraproject.org", "registry.access.redhat.com", "re
gistry.centos.org", "docker.io"]
[user@rhel8 ~]$
```

Информация о репозиториях и ссылки на них указаны в файле `registries.conf` в директории `/etc/containers/`:

```
grep registries /etc/containers/registries.conf
```

Видите, указано `unqualified?` Это означает, что если вы просто попытаетесь скачать образ не указав полную ссылку к нему, то образ будет искааться в одном из этих репозиториев. Но это плохая практика, а вдруг кто-то изменит этот список и укажет на свой репозиторий? Образы лучше скачивать указывая полный путь к ним.

INDEX	NAME	STARS	OFFICIAL	AUTOMATED	DESCRIPTION
redhat.com	registry.access.redhat.com/openshift3/ose-docker-registry	0			Supports the V2 Docker Registr
y API. Include...		0			
redhat.com	registry.access.redhat.com/openshift3/registry-console	0			Web console for the Atomic Reg
istry, an open...		0			
redhat.com	registry.access.redhat.com/openshift3/image-inspector	0			Image Inspector can extract th
e RPM composit...		0			
redhat.com	registry.access.redhat.com/openshift3/cri-o	0			CRI-O is an implementation of
the Kubernetes...		0			
redhat.com	registry.access.redhat.com/openshift3/ose-logging-kibana5	0			Starting with 3.10 we need to
support both E...		0			
redhat.com	registry.access.redhat.com/openshift3/ose-logging-curator5	0			Starting with 3.10 we need to
support both E...		0			
centos.org	registry.centos.org/centos/registry	0			
		0			
centos.org	registry.centos.org/centos/registry-v2	0			
		0			
centos.org	registry.centos.org/openshift/origin-docker-registry	0			
		0			
centos.org	registry.centos.org/pipeline-images/docker-registry-frontend	0			
		0			
centos.org	registry.centos.org/projectatomic/atomic-registry-install	0			
		0			
docker.io	docker.io/library/registry	3451	[OK]		The Docker Registry 2.0 imple
mentation for s...		3451	[OK]		

В локальных сетях может понадобится локальный репозиторий. И его можно поднять на том же контейнере. Давайте для начала найдём нужный образ с помощью search:

```
podman search registry
```

В списке вышло довольно много образов, но по полю STARS можно определить, что самый популярный - docker.io/library/registry. Рядом с ним под Official прописано OK, т.е. образ официальный. И ссылка ведёт на официальный сайт докера, т.е. это то что нам нужно.

```
[user@rhel8 ~]$ podman pull docker.io/library/registry
Trying to pull docker.io/library/registry:latest...
Getting image source signatures
Copying blob e2ead8259a04 done
Copying blob 5b27040df4a2 done
Copying blob 79e9f2f55bf5 done
Copying blob 0d96da54f60b done
Copying blob 3790aef225b9 done
Copying config b8604a3fe8 done
Writing manifest to image destination
Storing signatures
b8604a3fe8543c9e6afc29550de05b36cd162a97aa9b2833864ea8a5be11f3e2
[user@rhel8 ~]$ podman images
REPOSITORY          TAG      IMAGE ID   CREATED    SIZE
docker.io/library/registry  latest   b8604a3fe854  5 hours ago  26.8 MB
[user@rhel8 ~]$
```

Чтобы скачать образ используем опцию pull и укажем полную ссылку к образу:

```
podman pull docker.io/library/registry
```

После того, как образ скачается, можно посмотреть список образов с помощью опции images:

```
podman images
```

Чтобы запускать контейнеры свой репозиторий не нужен, мы его делаем только для того, чтобы создавать свои образы и раздавать их на другие наши сервера. Ну и чтобы экономить трафик - один сервер будет скачивать с интернета в локальный репозиторий, а другие сервера будут брать с этого локального репозитория.

```
[user@rhel8 ~]$ mkdir registry
[user@rhel8 ~]$ podman run --name myregistry -p 5000:5000 -v /home/user/registry:/var/lib/registry:Z -d registry
e3765521aa8e2d9ca1bfce8f9c9323cde1138c897c45a48e44b4a5556927f44
[user@rhel8 ~]$ podman ps
CONTAINER ID IMAGE COMMAND CREATED STATUS
PORTS NAMES
e3765521aa8e docker.io/library/registry:latest /etc/docker/regis... 5 seconds ago Up 5 seconds ago 0.0.0.0:5000->5000/tcp myregistry
[user@rhel8 ~]$
```

Теперь попробуем поднять свой репозиторий. Для начала ему нужен persistent storage, чтобы в случае проблем с контейнером мы не потеряли образы. Для этого создаём директорию:

```
mkdir registry
```

Далее нужно запустить контейнер. Для этого используется опция run:

```
podman run --name myregistry -p 5000:5000 -v /home/user/registry:/var/lib/registry:Z -d registry
```

Можно конечно запускать просто:

```
podman run registry
```

но получится просто изолированный контейнер, толку от которого мало. Поэтому давайте разберём наши ключи:

- **-name** - имя контейнера. Можно его не задавать, тогда podman сам генерирует для него имя.
- **-p** - проброс портов. В левой части порты на нашем хосте, справа - порты в контейнере. Т.е. мы говорим, что нужно открыть на локальной системе порт 5000 и пробросить его на 5000 порт контейнера. А в контейнере программа по умолчанию работает на 5000 порту. Но это касается именно registry, это его дефолтный порт. Скажем, если бы это был вебсервер, то мы бы указывали справа порт 80, так как в контейнере приложение слушало бы на 80 порту. Узнать, какой порт слушает по умолчанию образ можно на страничке самого образа в интернете. Так вот, мы пробросили порты, чтобы могли подключаться к репозиторию по сети.
- **-v** - volume. Какую директорию хоста к какой директории контейнера цеплять. Опять же, слева директория на нашей системе, справа - внутри контейнера. Т.е. наша директория /home/user/registry будет монтироваться в /var/lib/registry этого контейнера. Если мы удалим контейнер, эта директория никуда не денется и мы сможем прицепить её к другому контейнеру. И да, какую именно директорию внутри контейнера надо выносить обычно пишется на страничке образа или в документации, куда обычно дают ссылку. А опция :Z в конце нужна, чтобы selinux не блокировал попытки контейнера что-то изменить в проброшенной директории.
- **-d** - detach - отсоединить. Т.е. запустить контейнер в фоновом режиме. Если этот ключ не использовать, контейнер запустится в интерактивном режиме, а чтобы отсоединиться придётся вводить комбинацию клавиш. Легче сразу указать этот ключ при запуске.
- **registry** - название образа. Обратите внимание, что его нужно указывать в конце, после всех ключей. На самом деле после образа тоже можно указывать опции, но они уже будут относиться к самому контейнеру, а не подману.

```
[user@rhel8 ~]$ podman ps
CONTAINER ID IMAGE COMMAND CREATED STATUS
          PORTS NAMES
269596363fe9 docker.io/library/registry:latest /etc/docker/regis... 24 minutes ago Up 24 min
utes ago 0.0.0.0:5000->5000/tcp myregistry
[user@rhel8 ~]$ podman logs myregistry
time="2021-11-13T11:09:48.731973592Z" level=warning msg="No HTTP secret provided - generated ran
dom secret. This may cause problems with uploads if multiple registries are behind a load-balanc
er. To provide a shared secret, fill in http.secret in the configuration file or set the REGISTR
Y_HTTP_SECRET environment variable." go.version=g01.11.2 instance.id=0cd332e9-1a3d-438c-95a7-3c3
67e447da4 service=registry version=v2.7.1
```

Список запущенных контейнеров можно увидеть с помощью опции ps:

```
podman ps
```

При этом можно увидеть проброшенные порты, название, команду, запущенную в контейнере, время создания и текущий статус контейнера. А с помощью опции logs можно сразу просмотреть логи контейнера:

```
podman logs myregistry
```

Обычно туда сыпятся логи самого приложения внутри контейнера, засчёт чего не нужно заходить внутрь контейнера и возиться с поиском логов.

```
[user@rhel8 ~]$ podman exec -it myregistry bash
Error: exec failed: container linux.go:380: starting container process caused: exec: "bash": exe
cutable file not found in $PATH: OCI runtime attempted to invoke a command that was not found
[user@rhel8 ~]$ podman exec -it myregistry sh
/ # echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
/ # ls /usr/bin/
[       dc      hd      mkfifo    readlink   sum      unshare
[       deallocvt  head     mkpasswd  realpath  tac      unxz
awk      diff    hexdump   nc        renice   tail      unzip
basename  dirname  hostid    nl        reset    tee      uptime
bc       dos2unix  iconv    nmeter    resize   test      uudecode
beep      du      id       nohup    scanelf  time      uuencode
blkdiscard dumpleases  install  nproc    seq      timeout  vi
bunzip2   eject   ipcrm   nsenter  setkeycodes  top      vlock
bzcat      env    ipcs    nslookup setsid   tr      volname
bzip2      expand  killall  od      shasum   traceroute  wc
c_rehash   expr   ldd    openvt   sha256sum  traceroute6 wget
cal       factor  less    passwd  sha3sum   truncate which
```

Как я говорил, обычно в контейнере минимум программ, поэтому решать проблемы изнутри довольно сложно. Но такая необходимость иногда появляется, поэтому вы можете подключиться к контейнеру используя опцию exec:

```
podman exec -it myregistry bash
podman exec -it myregistry sh
```

Как видите, bash-а в контейнере нет, зато есть shell. Здесь ключ -i означает интерактивно, а ключ -t - что нужно выделить для этого виртуальную консоль. Если же посмотреть директории с переменной PATH, можно заметить, что утилит здесь довольно мало:

```
ls /usr/bin/
```

Но в некоторых образах программ много и можно управлять контейнером почти как виртуалкой.

3. Open Selinux permission. If SELinux is enabled on your system, you must turn on the `container_manage_cgroup` boolean to run containers with systemd as shown here (see the [Containers running systemd](#) solution for details):

```
# setsebool -P container_manage_cgroup 1
```

А если в контейнере есть systemd, а у вас настроен Selinux, для нормальной работы контейнера следует кое-что разрешить в Selinux на хосте:

```
sudo setsebool -P container_manage_cgroup 1
```

Чтобы выйти из контейнера просто нажимаем Ctrl+D.

```
[user@rhel8 ~]$ podman exec -d myregistry touch /var/lib/registry/file  
79384fea0078d5ea790e1cdf9508b66f0a650ff1722a97bde950b23f62a52ad1  
[user@rhel8 ~]$ ls registry/  
file  
[user@rhel8 ~]$ rm registry/file  
[user@rhel8 ~]$ █
```

Если нам нужно запустить какую-то команду в контейнере, не заходя в него, достаточно ключа `-d`. Например, создадим в проброшенной директории файл и убедимся, что он появился в нашей домашней директории:

```
podman exec -d myregistry touch /var/lib/registry/file  
ls registry/  
rm registry/file
```

```
[user@rhel8 ~]$ podman images  
REPOSITORY TAG IMAGE ID CREATED SIZE  
docker.io/library/registry latest b8604a3fe854 6 hours ago 26.8 MB  
[user@rhel8 ~]$ podman tag docker.io/library/registry:latest rhel8:5000/registry  
[user@rhel8 ~]$ podman images  
REPOSITORY TAG IMAGE ID CREATED SIZE  
docker.io/library/registry latest b8604a3fe854 6 hours ago 26.8 MB  
rhel8:5000/registry latest b8604a3fe854 6 hours ago 26.8 MB  
[user@rhel8 ~]$ podman push rhel8:5000/registry:latest  
Getting image source signatures  
Error: trying to reuse blob sha256:69715584ec78c168981b0925dd7c50f4537bc598dcbe814db2803a10b777  
b5c at destination: pinging container registry rhel8:5000: Get "https://rhel8:5000/v2/": http: server gave HTTP response to HTTPS client  
[user@rhel8 ~]$ █
```

Теперь давайте закинем в наш локальный репозиторий какой-нибудь образ, к примеру, образ самого registry. Прежде чем это сделать, ещё раз взглянем на список наших образов:

```
podman images
```

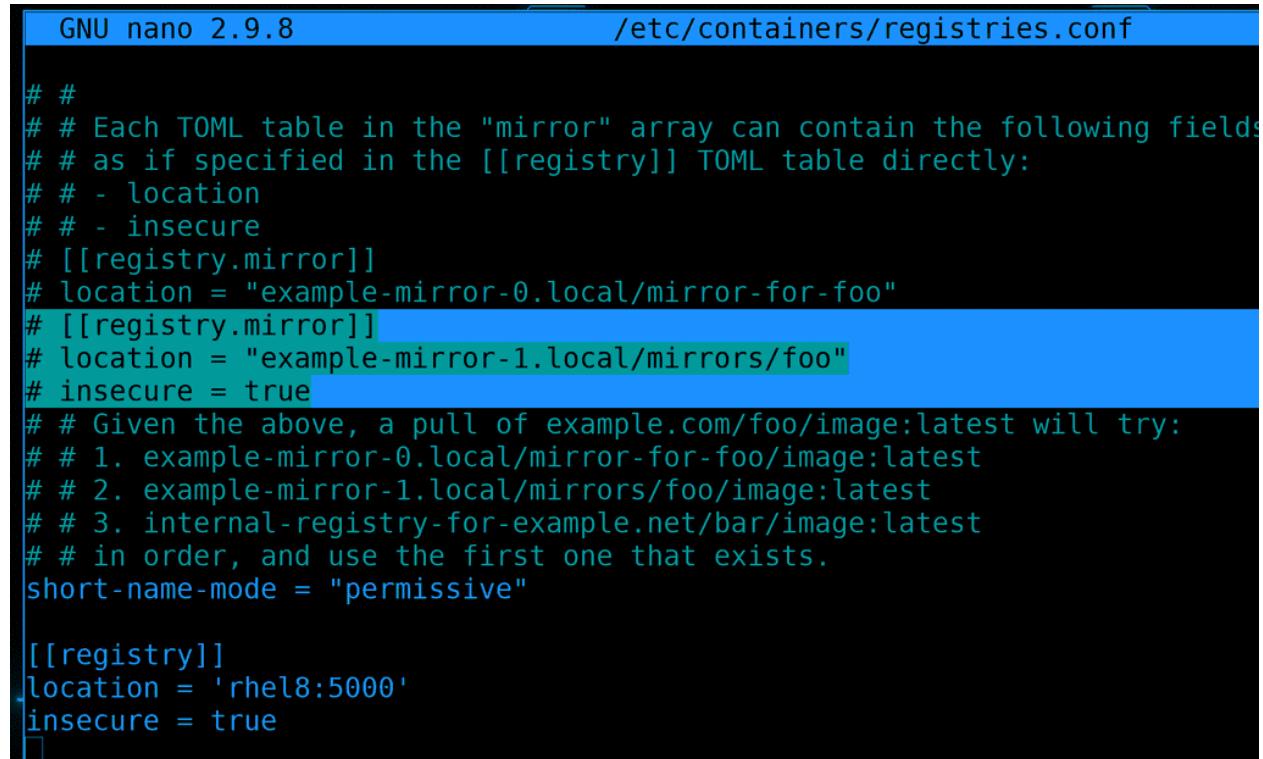
Как видите, у образа указан репозиторий docker.io. Нам нужно поменять этот репозиторий, для этого используем опцию `tag`:

```
podman tag docker.io/library/registry:latest rhel8:5000/registry
podman images
```

После опции tag мы указываем старый репозиторий, а потом новый. Теперь в списке образов мы видим новый образ с локальным репозиторием. Но при попытке залить образ в репозиторий с помощью опции push:

```
podman push rhel8:5000/registry:latest
```

мы получим ошибку, мол, мы пытаемся залить с помощью https, а сервер поддерживает только http. Можно было бы заморочиться с сертификатом, но мы сделаем несколько иначе - разрешим нашей системе работать с локальным репозиторием без сертификата.



```
GNU nano 2.9.8          /etc/containers/registries.conf

# #
# # Each TOML table in the "mirror" array can contain the following fields
# # as if specified in the [[registry]] TOML table directly:
# # - location
# # - insecure
[[registry.mirror]]
location = "example-mirror-0.local/mirror-for-foo"
[[registry.mirror]]
location = "example-mirror-1.local/mirrors/foo"
insecure = true
# # Given the above, a pull of example.com/foo/image:latest will try:
# # 1. example-mirror-0.local/mirror-for-foo/image:latest
# # 2. example-mirror-1.local/mirrors/foo/image:latest
# # 3. internal-registry-for-example.net/bar/image:latest
# # in order, and use the first one that exists.
short-name-mode = "permissive"

[[registry]]
location = 'rhel8:5000'
insecure = true
```

Для этого мы должны в файле /etc/containers/registries.conf прописать наш репозиторий. Пример, что именно нужно писать, есть в самом файле:

```
sudo nano /etc/containers/registries.conf
```

```
[[registry]]
location = 'rhel8:5000'
insecure = true
```

В location мы указываем на локальный адрес и порт, а insecure нам нужен, чтобы разрешить подключение через http.

```
[user@rhel8 ~]$ podman push rhel8:5000/registry:latest
Getting image source signatures
Copying blob ad10b481abe7 skipped: already exists
Copying blob aeccf26589a7 skipped: already exists
Copying blob aa4330046b37 skipped: already exists
Copying blob f640be0d5aad skipped: already exists
Copying blob 69715584ec78 [-----] 0.0b / 0.0b
Copying config b8604a3fe8 [=====] 3.0KiB / 3.0KiB
Writing manifest to image destination
Storing signatures
[user@rhel8 ~]$ ls registry/
docker
[user@rhel8 ~]$ ls registry/docker/
registry
[user@rhel8 ~]$ 
```

После изменения настроек ещё раз запустим push:

```
podman push rhel8:5000/registry:latest
ls registry
ls registry/docker
```

На этот раз всё прошло без проблем и даже в директории `registry` мы теперь можем увидеть наш контейнер.

```
[user@rhel8 ~]$ podman images
REPOSITORY          TAG      IMAGE ID      CREATED     SIZE
docker.io/library/registry  latest    b8604a3fe854  6 hours ago  26.8 MB
rhel8:5000/registry      latest    b8604a3fe854  6 hours ago  26.8 MB
[user@rhel8 ~]$ podman image rm docker.io/library/registry
Untagged: docker.io/library/registry:latest
[user@rhel8 ~]$ podman images
REPOSITORY          TAG      IMAGE ID      CREATED     SIZE
rhel8:5000/registry      latest    b8604a3fe854  6 hours ago  26.8 MB
[user@rhel8 ~]$ 
```

Теперь мы можем удалить наш старый образ и оставить только локальный, для этого нужно использовать опцию `image` с ключом `rm`:

```
podman images
podman image rm docker.io/library/registry:latest
podman images
```

```
[user@rhel8 ~]$ podman search httpd | head -3
INDEX          NAME           DESCRIPTION          STARS      OFFICIAL      AUTOMATED
fedoraproject.org  registry.fedoraproject.org/f29/httpd
redhat.com       registry.access.redhat.com/rhscl/httpd-24-rhel7
Apache HTTP 2.4 Server          0
[user@rhel8 ~]$ skopeo copy docker://registry.fedoraproject.org/f29/httpd docker://rhel8:5000/httpd
Getting image source signatures
Copying blob aaf5ad2e1aa3 done
Copying blob 7692efc5f81c done
Copying blob d77ff9f653ce done
Copying config 25c76f9dcdb done
Writing manifest to image destination
Storing signatures
[user@rhel8 ~]$
```

Но, как вы заметили, чтобы залить публичный образ в локальный репозиторий, надо его скачать с помощью pull, потом поменять тег, а потом залить с помощью push. Вместо этого можно использовать утилиту skopeo. К примеру, найдём ссылку на httpd:

```
podman search httpd | head -3
```

После чего используем утилиту skopeo с опцией copy, указывая откуда и куда:

```
skopeo copy docker://registry.fedoraproject.org/f29/httpd docker://rhel8:5000/httpd
```

Обратите внимание, что перед ссылками я добавил docker://. Это нужно, потому что копирование происходит по определённому механизму. Механизмы могут быть разные, где-то с авторизацией, где-то без, но в большинстве случаев docker:// хватает.

```
[user@rhel8 ~]$ podman search rhel8:5000/httpd
INDEX          NAME           DESCRIPTION          STARS      OFFICIAL      AUTOMATED
rhel8:5000    rhel8:5000/httpd          0
[user@rhel8 ~]$ podman pull rhel8:5000/httpd
Trying to pull rhel8:5000/httpd:latest...
Getting image source signatures
Copying blob 7692efc5f81c done
Copying blob d77ff9f653ce done
Copying blob aaf5ad2e1aa3 done
Copying config 25c76f9dcdb done
Writing manifest to image destination
Storing signatures
25c76f9dcdb5f512fe2d41587bbae84c25adf36126f46fba2ad6bad2ab1afabc
[user@rhel8 ~]$ podman images
REPOSITORY      TAG          IMAGE ID      CREATED      SIZE
rhel8:5000/registry  latest      b8604a3fe854  7 hours ago  26.8 MB
rhel8:5000/httpd    latest      25c76f9dcdb5  2 years ago  482 MB
[user@rhel8 ~]$
```

Кстати, давайте убедимся, что образ есть в локальном репозитории и что его можно поставить. Найдём его с помощью search:

```
podman search rhel8:5000/httpd
```

И скачаем с помощью pull:

```
podman pull rhel8:5000/httpd
podman images
```

Как видите, теперь у меня два образа, оба указывают на локальный репозиторий.

```
[user@rhel8 ~]$ podman image inspect rhel8:5000/httpd:latest
[{"Id": "25c76f9dcdb5f512fe2d41587bbae84c25adf36126f46fba2ad6bad2ab1afabc", "Digest": "sha256:7ce162a2ac97bb2d0e9424e93d28ac50979fde61e977e4c05a32ae96921705da", "RepoTags": ["rhel8:5000/httpd:latest"], "RepoDigests": ["rhel8:5000/httpd@sha256:7ce162a2ac97bb2d0e9424e93d28ac50979fde61e977e4c05a32ae96921705da"], "Parent": "", "Comment": "", "Created": "2019-04-18T13:07:15.407697Z", "Config": {"User": "1001", "ExposedPorts": {"8080/tcp": {}, "8443/tcp": {}}}}
```

Если у вас нет доступа к документации образа, но вам нужно разобраться, что там крутится, какие порты используются и увидеть все подробности про образ, вы можете использовать опцию image inspect. Например, посмотрим, что у нас в образе httpd:

```
podman image inspect rhel8:5000/httpd:latest
```

Здесь видно, какой пользователь в контейнере, какие порты, какие переменные окружения прописаны и прочая необходимая информация.

```
[user@rhel8 ~]$ podman run -d httpd
9fff5ba9d264375c52b872d0e6a125ceed2ba152f760b338da8aeb617615635e
[user@rhel8 ~]$ podman ps
CONTAINER ID IMAGE COMMAND CREATED STATUS
PORTS NAMES
e3765521aa8e docker.io/library/registry:latest /etc/docker/regis... About an hour ago Up About an hour ago 0.0.0.0:5000->5000/tcp myregistry
9fff5ba9d264 rhel8:5000/httpd:latest /usr/bin/run-http... 2 seconds ago Up 2 seconds ago suspicious_hamilton
[user@rhel8 ~]$ podman stop suspicious_hamilton
suspicious_hamilton
[user@rhel8 ~]$ podman ps
CONTAINER ID IMAGE COMMAND CREATED STATUS
PORTS NAMES
e3765521aa8e docker.io/library/registry:latest /etc/docker/regis... About an hour ago Up About an hour ago 0.0.0.0:5000->5000/tcp myregistry
[user@rhel8 ~]$ podman ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS
PORTS NAMES
e3765521aa8e docker.io/library/registry:latest /etc/docker/regis... About an hour ago Up About an hour ago 0.0.0.0:5000->5000/tcp myregistry
9fff5ba9d264 rhel8:5000/httpd:latest /usr/bin/run-http... 14 seconds ago Exited (0) 3 seconds ago suspicious_hamilton
[user@rhel8 ~]$ podman start suspicious_hamilton
suspicious_hamilton
[user@rhel8 ~]$
```

Давайте пройдёмся по частоиспользуемым ключам. И так, мы разбирали, что run позволяет создать

контейнер и сразу же запустить:

```
podman run -d httpd
```

После чего его можно увидеть с помощью опции ps:

```
podman ps
```

Как видите, контейнер получил рандомное, но читаемое название. Контейнер можно остановить с помощью stop:

```
podman stop suspicious_hamilton
```

Обратите внимание, что нужно использовать именно название контейнера, а не образа. После остановки мы перестанем видеть контейнер с помощью ps:

```
podman ps
```

Чтобы увидеть и выключенные контейнеры, нужен ключ -a:

```
podman ps -a
```

Ну и соответственно для запуска контейнера нужен ключ start:

```
podman start suspicious_hamilton
```

Опять же, указываем имя контейнера, а не образа. После остановки контейнера можно использовать опцию rm для его удаления.

```
[user@rhel8 ~]$ podman login
docker.io          registry.centos.org
registry.access.redhat.com  registry.fedoraproject.org
[user@rhel8 ~]$ podman login docker.io
Username: test
Password:
Error: error logging into "docker.io": invalid username/password
[user@rhel8 ~]$
```

В некоторых репозиториях настроена аутентификация. Например, хотя докерхабом могут пользоваться все желающие, чтобы не забивать трафик на сервера там настроен лимит на скачивание образов с одного IP адреса. Хотя лимит довольно большой, но если не пользоваться локальными репозиториями, если много тестировать, можно дойти до лимита. И чтобы платные подписчики не страдали из-за лимита, в докерхаб можно логиниться с опцией login:

```
podman login docker.io
```

Это касается не только докерхаба. Даже в локальных репозиториях иногда стоит настраивать аутентификацию, чтобы не каждый желающий мог скачивать образы.

Хоть сейчас наши контейнеры и работают, но после перезагрузки автоматом не будут стартовать. Обычно за автозапуск приложений отвечает systemd, но podman работает без демона, поэтому после перезагрузки не будут стартовать контейнеры сами по себе. Для этого нам нужно самим создать systemd сервисы. Но есть нюанс - если мы создадим сервисы как мы это делали раньше, то они будут запускаться от рута, чего нам не хотелось бы. Да, конечно мы можем в сервисе указать юзера, но есть уже готовое решение. Точнее, полуготовое.

```
podman-generate-systemd(1)      General Commands Manual      podman-generate-systemd(1)

NAME
    podman-generate-systemd - Generate systemd unit file(s) for a container or pod

SYNOPSIS
    podman generate systemd [options] container|pod

DESCRIPTION
    podman generate systemd will create a systemd unit file that can be used to control a
    container or pod. By default, the command will print the content of the unit files to
    stdout.
```

Но благо есть документация. То что мы ищем относится к опции generate systemd:

```
man podman-systemd-generate
```

```
[user@rhel8 ~]$ podman generate systemd --restart-policy=always -t 1 --files --name myregistry
/home/user/container-myregistry.service
[user@rhel8 ~]$ head container-myregistry.service
# container-myregistry.service
# autogenerated by Podman 3.3.1
# Sat Nov 13 16:55:55 MSK 2021

[Unit]
Description=Podman container-myregistry.service
Documentation=man:podman-generate-systemd(1)
Wants=network-online.target
After=network-online.target

        like iSCSI or other remote block protocols, this ensures that Podman is not executed
prior to any necessary storage operations coming online.

$ podman create --name nginx nginx:latest
$ podman generate systemd --restart-policy=always -t 1 nginx
# container-de1e3223b1b888bc02d0962dd6cb5855eb00734061013ffdd3479d225abacdc6.servi
ce
# autogenerated by Podman 1.8.0
# Wed Mar 09 09:46:45 CEST 2020

[Unit]
```

В примерах есть две команды: podman create и podman generate. Create относится к созданию контейнера, он у нас уже есть, поэтому эту команду пропускаем. Дальше generate - эта команда создаёт файл systemd сервиса. Используем её указав наш контейнер:

```
podman generate systemd --restart-policy=always -t 1 --files --name myregistry
```

К этой команде я добавил ещё два ключа - --files и --name. Оба ключа есть в той же документации, только чуть ниже, в примерах с подами. Они позволяют сразу создать файл, а не выводить файл сервиса на экран. В итоге в нашей домашней директории появится файл systemd сервиса.

```
[user@rhel8 ~]$ mkdir -p ~/.config/systemd/user  
[user@rhel8 ~]$ cp container-myregistry.service ~/.config/systemd/user/  
[user@rhel8 ~]$ █
```

Podman-generated unit files include an [Install] section, which carries installation information for the unit. It is used by the enable and disable commands of systemctl(1) during installation.

Once you have generated the systemd unit file, you can copy the generated systemd file to /etc/systemd/system for installing as a root user and to \$HOME/.config/systemd/user for installing it as a non-root user. Enable the copied unit file or files using systemctl enable.

Для начала, у systemd есть глобальные сервисы, а есть и личные сервисы каждого пользователя. Глобальные лежат в /etc/systemd, а личные - в домашней директории пользователя в директории ~/.config/systemd/user. Чуть ниже в мане можно найти эту директорию, если вдруг забыли, где она находится. По умолчанию, такой директории нет, поэтому стоит её создать и скопировать туда сконструированный файл:

```
mkdir -p ~/.config/systemd/user  
cp container-myregistry.service ~/.config/systemd/user/
```

```
[user@rhel8 ~]$ systemctl --user enable container-myregistry.service  
Created symlink /home/user/.config/systemd/user/multi-user.target.wants/container-myregistry.service → /home/user/.config/systemd/user/container-myregistry.service.  
Created symlink /home/user/.config/systemd/user/default.target.wants/container-myregistry.service → /home/user/.config/systemd/user/container-myregistry.service.  
[user@rhel8 ~]$ loginctl enable-linger user  
[user@rhel8 ~]$ █
```

user, enable the service with --user flag.

```
$ systemctl --user enable <.service>
```

The systemd user instance is killed after the last session for the user is closed. The systemd user instance can be kept running ever after the user logs out by enabling lingering using

```
$ loginctl enable-linger <username>
```

Use systemctl to perform operations on generated installed unit files.

После чего нам надо включить этот сервис. Если сервис принадлежит пользователю, то ему не нужно писать sudo, но нужно добавлять ключ –user:

```
systemctl --user enable container-myregistry.service
```

Опять же, в документации выше есть пример этой команды. Ну и самое главное - пока у пользователя нет сессий, его systemd не будет работать. Значит после перезагрузки контейнеры не запустятся, а если

человек разлогиниться - завершатся. Чтобы позволить сервисам пользователя работать без его логина, надо включить опцию lingering, которая также показана в документации:

```
logindctl enable-linger user
```

```
[user@rhel8 ~]$ sudo reboot
[sudo] password for user:
Connection to 192.168.31.205 closed by remote host.
Connection to 192.168.31.205 closed.
[doctor@tardis:~]
$ ssh root@rhel
X11 forwarding request failed on channel 0
Web console: https://rhel8:9090/ or https://192.168.31.205:9090/
This system is not registered to Red Hat Insights. See https://cloud.redhat.com/
To register this system, run: insights-client --register

Last login: Sat Oct  2 17:46:52 2021 from 192.168.31.5
[root@rhel8 ~]# nc -zv localhost 5000
Ncat: Version 7.70 ( https://nmap.org/ncat )
Ncat: Connected to ::1:5000.
Ncat: 0 bytes sent, 0 bytes received in 0.02 seconds.
[root@rhel8 ~]#
```

И чтобы убедиться в том, что наш контейнер запустится после рестарта и будет работать без нашего участия, давайте перезагрузим систему:

```
sudo reboot
```

После чего сразу подключимся пользователем root, чтобы не вызвать старт сессии пользователя user и просто проверим доступность порта:

```
ssh root@rhel
nc -zv localhost 5000
```

Как видите, порт доступен, а значит контейнер запущен.

```
[root@rhel8 ~]# podman ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
[root@rhel8 ~]# su - user
[user@rhel8 ~]$ podman ps
CONTAINER ID IMAGE COMMAND CREATED STATUS
PORTS NAMES
e3765521aa8e docker.io/library/registry:latest /etc/docker/regis... 2 hours ago Up 2 minutes
ago 0.0.0.0:5000->5000/tcp myregistry
[user@rhel8 ~]$
```

Ну и podman ps от рута ничего нам не будет показывать:

```
podman ps
```

потому что у каждого пользователя свои контейнеры. Поэтому перейдём на нашего пользователя и проверим статус контейнеров:

```
su - user
podman ps
```

Ну и отсюда тоже видно, что контейнер запустился не секунду назад, а уже работает со времени запуска системы.

Ну и стоит учитывать, что из-за того, что наш контейнер работает не от рута, ему запрещено пробрасывать порты ниже 1024. Т.е. внутри контейнера может быть любой порт, а вот снаружи только выше 1024. Порты ниже 1024 называются привилегированными и считается, что сервис, стоящий за ними, работает от рута, а те что выше - непривилегированными, а значит любой желающий сервис может их использовать.

```
[user@rhel8 ~]$ sudo firewall-cmd --add-forward-port=port=80:proto=tcp:toport=5000 --permanent
success
[user@rhel8 ~]$ sudo firewall-cmd --reload
success
[user@rhel8 ~]$ 
[doctor@tardis:~] 
$ nc -zv 192.168.31.205 80
Connection to 192.168.31.205 80 port [tcp/http] succeeded!
[doctor@tardis:~] 
$ 
```

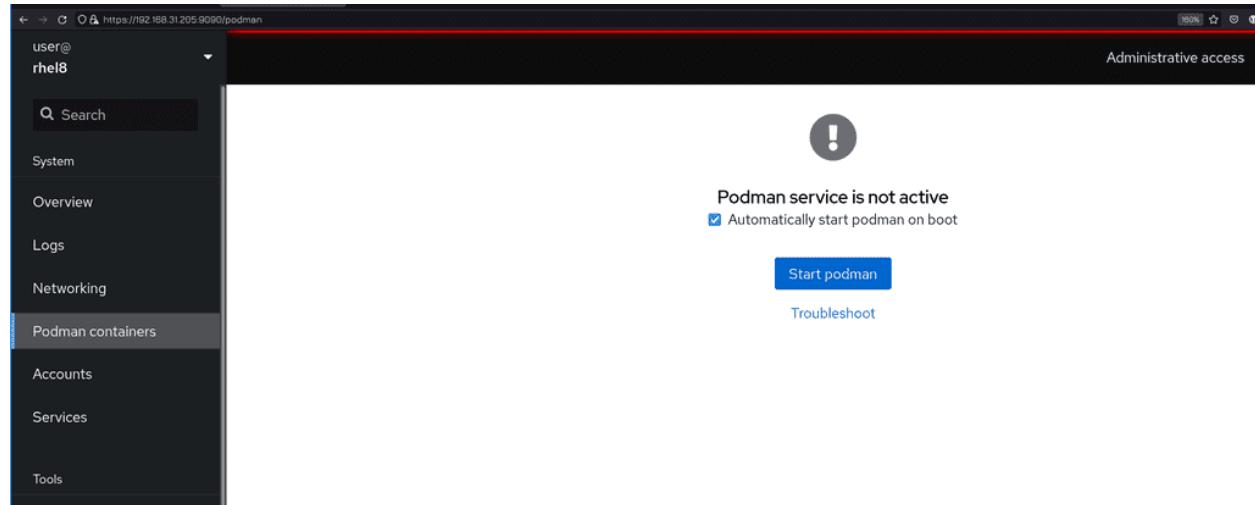
Как же нам тогда использовать порты ниже 1024? Можно, к примеру, делать перенаправление на файерволе. Если нам обязательно, чтобы клиенты подключались к порту 80, то можем прописать такое правило:

```
sudo firewall-cmd --add-forward-port=port=80:proto=tcp:toport=5000 --permanent
sudo firewall-cmd --reload
```

Здесь сказано, что нужно перенаправлять пакеты, приходящие на 80 порт по tcp на порт 5000. Ну и для теста с моей системы запустим проверку доступности порта:

```
nc -zv 192.168.31.205 80
```

Всё работает как надо.



Ну и напоследок, давайте глянем cockpit. Здесь у нас появилась вкладка podman containers. Нас пре-

дупреждают, что podman не настроен как сервис и простым нажатием «Start podman» это можно сделать.

The screenshot shows the podman interface with two main sections: 'Containers' and 'Images'. In the 'Containers' section, there is one container named 'myregistry' which is running. It uses 0.00% CPU and 0.001 / 8.00 GiB memory. In the 'Images' section, there are two images: 'rhel8:5000/httpd:latest' created on 04/18/2019 and 'rhel8:5000/registry:latest' created today at 10:22 AM. Both images are 460 MiB and 25.6 MiB respectively, and both are owned by 'user'.

После чего мы увидим окно с нашими контейнерами и образами.

The screenshot shows the podman interface with the 'Containers' section selected. The 'myregistry' container is listed and is currently running. Below the table, there are tabs for 'Details', 'Logs' (which is selected), and 'Console'. The 'Logs' tab displays the container's log output. The log output shows various messages related to Redis configuration, shared secrets, and registry operations, indicating the container is active and performing its intended function as a registry.

Отсюда мы можем управлять контейнерами, смотреть логи и даже подключаться к консоли контейнера.

Давайте подведём итоги. Сегодня мы разобрали, как работать с подманом - как искать и скачивать образы, поднимать контейнеры, управлять ими, добавлять их в автозапуск. Кроме того мы подняли свой репозиторий для контейнеров и научились бросать туда образы. Как я говорил, тема контейнеризации довольно большая и нам всего не разобрать, но мы на базовом уровне научились работать с контейнерами, а это уже покрывает большую часть задач.

2.63.2 Практика

Вопросы

1. Что можно считать «единой точкой отказа» при работе с docker и как можно этого избежать?
2. Что такое pod?
3. Чем может быть полезен локальный репозиторий?
4. После запуска сервиса в контейнере он перестал отвечать, как посмотреть логи контейнера?
5. При работе с контейнером понадобилось узнать, какой порт слушается внутри контейнера с помощью какой команды можно это посмотреть?

Задания

2.64 64. Про сертификацию RHCSA

2.64.1 64. Про сертификацию RHCSA

	Text
Understand and use essential tools	
Access a shell prompt and issue commands with correct syntax	05
Use input-output redirection (>, >>, , 2>, etc.)	11
Use grep and regular expressions to analyze text	09, 31, 61
Access remote systems using SSH	44
Log in and switch users in multiuser targets	17
Archive, compress, unpack, and uncompress files using tar, star, gzip, and bzip2	38, 64
Create and edit text files	10

Хотя изначальная идея курса была в подготовке к сертификации по RHCSA, но ещё до первой темы я решил, что если уж и делать курс с простыми темами, то лучше сразу делать его для начинающих и затрагивать все основы работы с GNU/Linux. Ну а подготовка к экзамену осталась как путеводитель, как доказательство того, что этот курс привёл нас к каким-то ощутимым результатам. И что теперь у нас хватает знаний для получения сертификата, признанного во всём мире, показывающего, что мы можем администрировать Red Hat Enterprise Linux. Этих знаний хватит для начала работы младшим специалистом и они дадут фундамент для дальнейшего роста.

Но на этом данный курс не закончится, я буду продолжать добавлять в него новые темы. Но и не думаю, что в принципе хоть как-то можно завершить курс, потому что тема основ бесконечная. Давайте лучше поговорим про сертификацию.

Один из частозадаваемых вопросов:

Нужна ли мне сертификация? Что она мне даст?

Суть не в том, что вы пойдёте и сдадите экзамен. Вы этого можете не делать, это не так важно. Главное - подготовка к нему, получение знаний. Когда вы не знаете какую-то тему, скажем, как администрировать линукс, вам очень сложно понять, а что именно нужно изучать и какой уровень знаний считается достаточным и общепринятым. И вот крупнейшая компания даёт перечень тем, которые считает необходимыми для понимания. На самом деле

этот список можно оспорить, каких-то тем не хватает, какие-то темы можно выкинуть, но в целом указаны почти все важные темы для уровня младшего специалиста. Что даёт вам сертификация? Понимание, в каком направлении двигаться. Экзамен лишь доказательство того, что вы этот путь прошли. Но вам не обязательно кому-то это доказывать. Экзамен стоит немало денег, несколько сотен долларов. Некоторые компании оплачивают экзамены за сотрудников, поэтому мой совет - если у вас нет работы, не нужно тратиться на сам экзамен. Экзамен даст вам бумажку, возможно, это чуть улучшит ваше резюме, но это далеко не решающий фактор и адекватный работодатель будет оценивать по уровню знаний, а не по бумажкам. А вот если вы работаете, если компания готова оплатить вам экзамен - то да, стоит его сдать. Вам это обойдется бесплатно, а ваш работодатель может чуть поднять вам зарплату, ну или по крайней мере ваше CV станет чуточку лучше.

Другой вопрос:

Audience for this exam

- Experienced Red Hat Enterprise Linux system administrators seeking validation of their skills
- Students who have attended [Red Hat System Administration I \(RH124\)](#) and [Red Hat System Administration II \(RH134\)](#) and are on the path to becoming an RHCSA
- Experienced Linux system administrators who require a certification either by their organization or based on a mandate (DoD 8570 directive)
- IT professionals who are on the path to becoming a [Red Hat Certified Engineer \(RHCE\)](#)
- An RHCE who is noncurrent or who is about to become noncurrent and wants to recertify as an RHCE
- DevOps professionals who wish to demonstrate their expertise with the fundamentals of container technology

Prerequisites for this exam

- Have either taken [Red Hat System Administration I \(RH124\)](#) and [Red Hat System Administration II \(RH134\)](#) or the [RHCSA Rapid Track course \(RH199\)](#) that combines those courses, or have comparable work experience as a system administrator on Red Hat Enterprise Linux
- Review the Red Hat Certified System Administrator exam (EX200) objectives
- [Take our free assessment](#) to find the course that best supports your preparation for this exam.

Для кого предназначена сертификация, кому стоит идти и сдавать экзамен?

В идеале - для младших линукс администраторов, которые отработали год, набрались знаний, практики и решили их доказать. А также для тех, кто раньше работал Windows администратором, а потом решил изучить администрирование Linux-ов. В общем, для людей, у которых есть небольшой опыт за плечами. Но я нередко вижу, что на экзамен идут люди, которые только хотят устроиться на свою первую работу, мол, сертификат поможет с устройством на работу.

Overview

Objectives

What you need to know

Skills path

Preparation

Red Hat encourages you to consider taking [Red Hat System Administration I \(RH124\)](#) and [Red Hat System Administration II \(RH134\)](#) to help prepare.

Attendance in these classes is not required; you can choose to take just the exam.

While attending Red Hat classes can be an important part of your preparation, attending class does not guarantee success on the exam. Previous experience, practice, and native aptitude are also important determinants of success.

Many books and other resources on system administration for Red Hat products are available. Red Hat does not endorse any of these materials as preparation guides for exams. Nevertheless, you may find additional reading helpful to deepen your understanding.

Что из себя представляет экзамен?

Экзамен практический, т.е. вам дадут компьютер и задания. Всё будет на английском, поэтому нужно уметь читать и понимать задания. У вас не будет интернета, но будет системная документация, т.е. man будет работать. Какого рода будут задания нельзя разглашать, но все темы есть на сайте. Вы можете выполнять задания любыми способами - хоть через терминал, хоть через графический интерфейс, хоть через веб. По итогу всякие скрипты проверят именно результат. И самое главное - проверка будет после перезагрузки, т.е. ваша система после всего должна запуститься и ваши изменения должны в ней остаться. Экзамен длится 3 часа, но времени хватает впритык. Скажем, если вы уверенно будете делать без всяких затыков все задания, у вас останется максимум минут 30-40. Естественно, не всё будет гладко, поэтому это лишнее время вам пригодится. Максимальный балл - 300, а проходный - 210, т.е. где-то пару заданий вы можете не выполнить, но старайтесь делать всё, мало ли где-то будут ошибки.

Какие советы по экзамену?

Во-первых, когда вам дадут задания, прочтите их все и суммируйте для себя. Некоторые задания могут быть связаны с другими и если вы тупо будете делать их по порядку - то в какой-то момент вам придётся всё переделывать, а на это нет времени. Во-вторых, начните с тем, которые могут повлиять на запуск системы. Скажем, работа с дисками и файловыми системами. Если вы всё испортите в начале, будет время всё исправить. Если же вы всё испортите сделав половину заданий, у вас не хватит времени всё восстановить. В-третьих, периодически, раз в пару заданий, перезагружайте систему для проверки и убежайтесь, что есть результат выполненных заданий. Да, это отнимает время, но будет легче понять и сразу исправить ошибку. Ну и самое главное - не нервничайте. После экзамена никто вас не побьёт, никто ничего плохого с вами не сделает. Максимум вы потеряете деньги, а это печально. Лучше замените волнения и переживаний на печаль по деньгам, так будет проще сконцентрироваться на заданиях и быть более сосредоточенным.

Google search results for "rhcsa москва":

- [Экзамен RHCSA — сертификации Red Hat Certified System ...](https://edu.softline.com/certification/ekzamen-rhcsa)
Сертификации Red Hat Certified System Administrator - экзамен **RHCSA**: ✓ стоимость экзамена, ✓ запись онлайн, ✓ подробнее о ... Города: **Москва**, ...
- [Red Hat Certified System Administrator \(RHCSA\) - Инвента](https://www.inventa.ru/certification/rhcsa)
Экзамен **RHCSA** — начальная ступень сертификации, которая ориентирована на начинающих пользователей Linux/Unix. Эта сертификация появилась с выходом 6-й ...

Где сдавать экзамен?

Сейчас есть 2 варианта сдать - офлайн и онлайн. Оффлайн проводится у партнёров компании Red Hat в разных городах и разных компаниях, поэтому просто наберите в гугле «RHCSA» и ваш город. Там же можно найти или спросить цены. Обычно перед датами экзамена эти компании также проводят быстрые тренинги по подготовке к экзамену. Обычно это занимает неделю или две на полный рабочий день, т.е. некоторые компании оплачивают сотрудникам подобные курсы. Эти курсы не гарантия, что вы сдадите экзамен, но если у вас есть опыт и знания, курсы помогут вам закрыть пробелы перед экзаменом.

Red Hat Certified System Administrator (RHCSA) exam

The page shows the following details:

- Overview** (selected)
- Objectives**
- What you need to know**
- Skills path**
- Updates for this exam**
 - You can now test from wherever you are by choosing our remote exam format. [Find out more →](#)
- Location**: Russia
- Format**: Individual Exam
- Duration**: 3.00 hours
- Cost**: 450 USD (Price excludes VAT) or 2 Training Units
- Get started** button
- Find a learning facility near you**
- Already purchased this offering?** [Log in](#)

Exam description

Онлайн экзамен можно заказать на официальном сайте редхата, он выходит дороже, но вы сдаёте его дома. Там есть процедура, что нужно подготовить комнату, загрузочную флешку и прочее. Весь экзамен за вами будут следить с камеры. На мой взгляд, лучше сдавать офлайн, потому что всякие обрывы в сети и нестабильное соединение могут сильно подпортить вам нервы, а если будете сдавать очно, то за работу всего отвечает компания-организатор. И даже если что-то пойдёт не так, отвечать будет организатор.

The screenshot shows a web browser window with the URL <https://basis.gnulinux.pro/rhcsa>. The page title is "Основы GNU/Linux и подготовка к RHCSA". On the left, there's a sidebar with navigation links: Вступление, Оглавление, Ссылки, Команды, and RHCSA (which is highlighted). Below that is a "КУРС" section with two items: 01. Операционные системы и GNU-Linux and 02. Виртуализация. At the bottom of the sidebar is a link "02 Установка CentOS". The main content area displays a table of contents for the RHCSA course:

Understand and use essential tools	Text
Access a shell prompt and issue commands with correct syntax	05
Use input-output redirection (>, >>, , 2>, etc.)	11
Use grep and regular expressions to analyze text	09, 31, 61
Access remote systems using SSH	44
Log in and switch users in multiuser targets	17
Archive, compress, unpack, and uncompress files using tar, star, gzip, and bzip2	38, 64
Create and edit text files	10

А как готовиться к экзамену? Твоего курса хватает для сдачи?

Да, я изначально поставил себе цель сделать полный курс, которого хватит для сдачи экзамена. Сейчас в нём не хватает практических заданий и лабораторных, но я над этим работаю. По [ссылке](#) вы можете увидеть, какие темы экзамена в каких уроках разобраны, но я советую посмотреть весь курс для полного понимания. Если вы посмотрите весь курс, ответите на вопросы, сделаете задания и лабораторные - то этого хватит. Для начинающих я бы посоветовал параллельно смотреть какой-то другой курс, потому что вы где-то можете не так понять, что я имею ввиду, а обрабатывая информацию с нескольких источников вы лучше её усваиваете. В любом случае, вы можете задавать вопросы в телеграмм группе.

Operate running systems	Text
Boot, reboot, and shut down a system normally	64
Boot systems into different targets manually	64

Но есть пара тем, которые остались в сторонке, это пара простых команд, которые я разберу прямо сейчас. Если вы не смотрели курс, то можете пропустить эту часть.

```
[user@rhel8 ~]$ shutdown now ^C
[user@rhel8 ~]$ poweroff ^C
[user@rhel8 ~]$ ls -l /sbin/shutdown
lrwxrwxrwx. 1 root root 16 Jul 28 14:56 /sbin/shutdown -> ../bin/systemctl
[user@rhel8 ~]$ ls -l /sbin/poweroff
lrwxrwxrwx. 1 root root 16 Jul 28 14:56 /sbin/poweroff -> ../bin/systemctl
[user@rhel8 ~]$ systemctl poweroff ^C
[user@rhel8 ~]$ systemctl isolate poweroff.target ^C
[user@rhel8 ~]$ █
```

Как правильно выключать компьютер? Есть пара команд:

```
shutdown now
poweroff
```

Может где-то на старых юниксовых системах эти команды делали разные вещи, но в современных системах, по крайней мере там где systemd, они делают одно и тоже и являются просто символическими ссылками на systemctl:

```
ls -l /sbin/shutdown
ls -l /sbin/poweroff
```

Также для выключения можно использовать сам systemctl:

```
systemctl poweroff
```

На самом же деле выполняется команда

```
systemctl isolate poweroff.target
```

Мы с вами и systemd и таргеты проходили, поэтому не буду на этом акцентировать внимание. Ну, в общем, можно использовать любую из команд.

```
[user@rhel8 ~]$ reboot ^C
[user@rhel8 ~]$ ls -l /sbin/reboot
lrwxrwxrwx. 1 root root 16 Jul 28 14:56 /sbin/reboot -> ../bin/systemctl
[user@rhel8 ~]$ systemctl reboot ^C
[user@rhel8 ~]$ systemctl isolate reboot.target ^C
[user@rhel8 ~]$ █
```

Тоже самое касается перезагрузки, есть команда reboot:

```
reboot
```

которая на самом деле является символьической ссылкой на systemctl:

```
ls -l /sbin/reboot
```

Можно дать перезагрузку через systemctl:

```
systemctl reboot
```

А на самом деле выполняется reboot таргет:

```
systemctl isolate reboot.target
```

```
load_video
set gfx_payload=keep
insmod gzio
linux ($root)/vmlinuz-4.18.0-305.12.1.el8_4.x86_64 root=/dev/mapper/rootvg_rhe\
l8-rootlv ro resume=/dev/mapper/rootvg_rhel8-swaplv rd.lvm.lv=rootvg_rhel8/roo\
tlv rd.lvm.lv=rootvg_rhel8-swaplv rhgb quiet systemd.unit=multi-user.target
initrd ($root)/initramfs-4.18.0-305.12.1.el8_4.x86_64.img $tuned_initrd

Press Ctrl-x to start, Ctrl-c for a command prompt or Escape to
discard edits and return to the menu. Pressing Tab lists
possible completions.
```

Теперь про загрузку системы в разные таргеты вручную. Тут нужно редактировать grub, мы это умеем. Просто в строке ядра надо прописать опцию `systemd.unit` с нужным таргетом, например, `multi-user`:

`systemd.unit=multi-user.target`

Или `rescue.target`:

`systemd.unit=rescue.target`

После чего надо нажать `Ctrl+X`. Система так загрузится только раз, а после перезагрузки будет как раньше. Как менять таргет навсегда мы разбирали в 34 теме.

```
[user@rhel8 ~]$ sudo dnf install star
Updating Subscription Management repositories.
Last metadata expiration check: 13:28:39 ago on Sat 13 Nov 2021 10:00:38 PM MSK.
Dependencies resolved.
=====
 Package      Architecture Version          Repository           Size
 =====
 Installing:
 star         x86_64       1.5.3-13.el8      rhel-8-for-x86_64-baseos-rpms   301 k
 Transaction Summary
```

Ну и последнее - утилита `star`. Это такой же архиватор, как `tar`, только чуть быстрее и есть дополнительный функционал. Для начала установим её:

```
sudo dnf install star
```

```
[user@rhel8 ~]$ star -c -f arch.star /etc/passwd /etc/group
star: 1 blocks + 0 bytes (total of 10240 bytes = 10.00k).
[user@rhel8 ~]$ star -t -f arch.star /etc/passwd /etc/group
/etc/passwd
/etc/group
star: 1 blocks + 0 bytes (total of 10240 bytes = 10.00k).
[user@rhel8 ~]$ star -tf arch.star /etc/passwd /etc/group
star: Bad Option: -tf.
Usage: star cmd [options] [-find] file1 ... filen [find expression]

use      star -help
and      star -xhelp
to get a list of valid cmds and options.

use      star H=help
to get a list of valid archive header formats.

use      star diffopts=help
to get a list of valid diff options.
[user@rhel8 ~]$
```

Синтаксис похож на тот же tar:

```
star -c -f arch.star /etc/passwd /etc/group
star -t -f arch.star /etc/passwd /etc/group
```

разве что ключи слитно не работают:

```
star -cf arch.star /etc/passwd /etc/group
```

```
[user@rhel8 ~]$ star -xattr -H=exustar -c -f arch2.star /etc/passwd
star: 1 blocks + 0 bytes (total of 10240 bytes = 10.00k).
[user@rhel8 ~]$ tar --selinux -cf arch3.tar /etc/passwd
tar: Removing leading '/' from member names
[user@rhel8 ~]$ man star
```

Насколько я понимаю, раньше tar не мог работать с selinux, т.е. при архивации не сохранялся контекст файлов. Этот функционал был в star и поэтому Red Hat требовали знания этой утилиты. Но, честно говоря, синтаксис для сохранения контекста довольно мудрёный:

```
star -xattr -H=exustar -c -f arch2.star /etc/passwd
```

Сейчас же tar также поддерживает сохранение контекста и в нём это сделано попроще, просто добавляем опцию --selinux при архивации и разархивации:

```
tar --selinux -cf arch3.tar /etc/passwd
```

Из и больше примеров использования star вы можете найти в документации:

```
man star
```

Я не буду вдаваться в подробности этой утилиты, потому что она довольно специфична и я ни разу не встречал практического применения. Просто показал, потому что она прописана в темах экзамена.

Если вы собираетесь сдавать экзамен - удачи вам! Будет интересно послушать комментарии тех, кто уже сдал, да и в целом любые комментарии. Пишите, что думаете о курсе, что вам нравится, что не нравится, вопросы и советы другим. На этом тему подготовки к сертификации можно считать закрытой.