```
[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
```

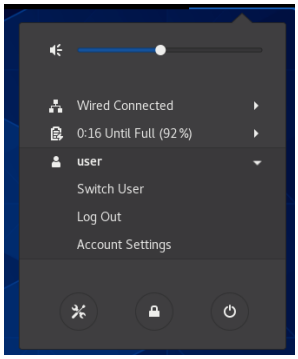
И всё что нам остаётся сделать для создания файловой системы – написать команду mkfs точка, желаемый тип файловой системы, а потом раздел или диск - sudo mkfs.ext4 /dev/sdb1. Программа создаст файловую систему и покажет нам информацию о ней – сколько блоков какого объёма поместилось в файловой системе, какое количество инод, идентификатор файловой системы, где сохранились бэкапы суперблока - того самого, где хранятся метаданные самой файловой системы.

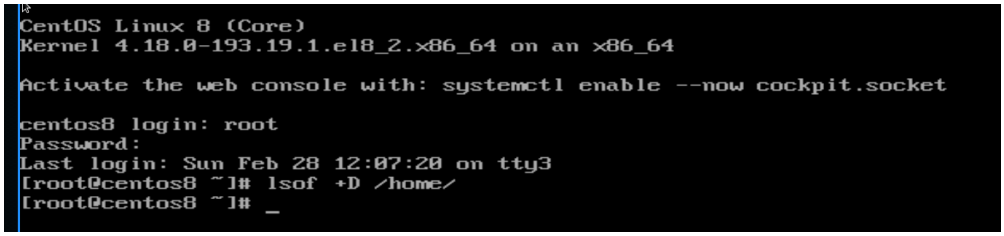
```
[user@centos8 ~]$ sudo tune2fs -l /dev/sdb1
tune2fs 1.45.4 (23-Sep-2019)
Filesystem volume name:   <none>
Last mounted on:          <not available>
Filesystem UUID:          555a6e59-83e6-4331-b55e-c837fdfc1a22
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      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 -l 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.





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-9152-1cef77fcb473
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 монтировать 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.

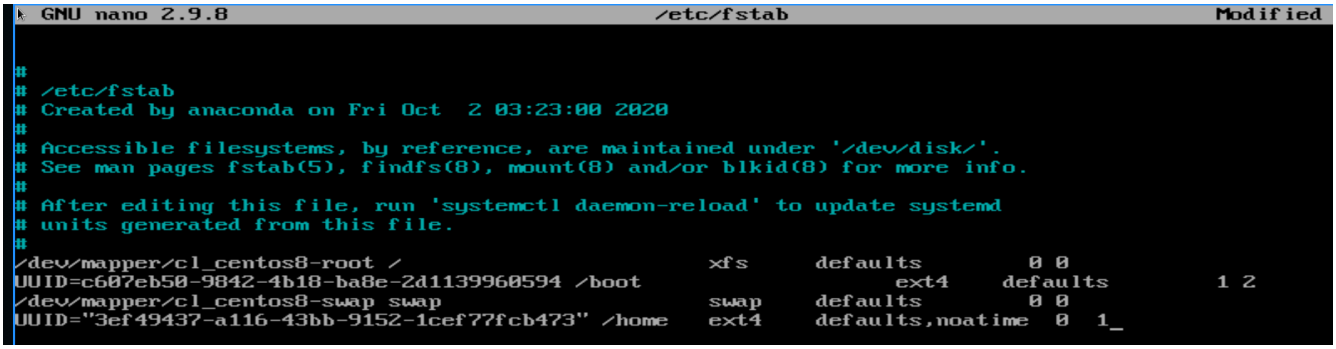
```
[root@centos8 ~]# cat /etc/fstab

#
# /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                    swap    defaults        0 0
[root@centos8 ~]# blkid /dev/sdc2
/dev/sdc2: UUID="3ef49437-a116-43bb-9152-1cef77fcb473" 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-9152-1cef77fcb473"
[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.



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 сохраняем и выходим.

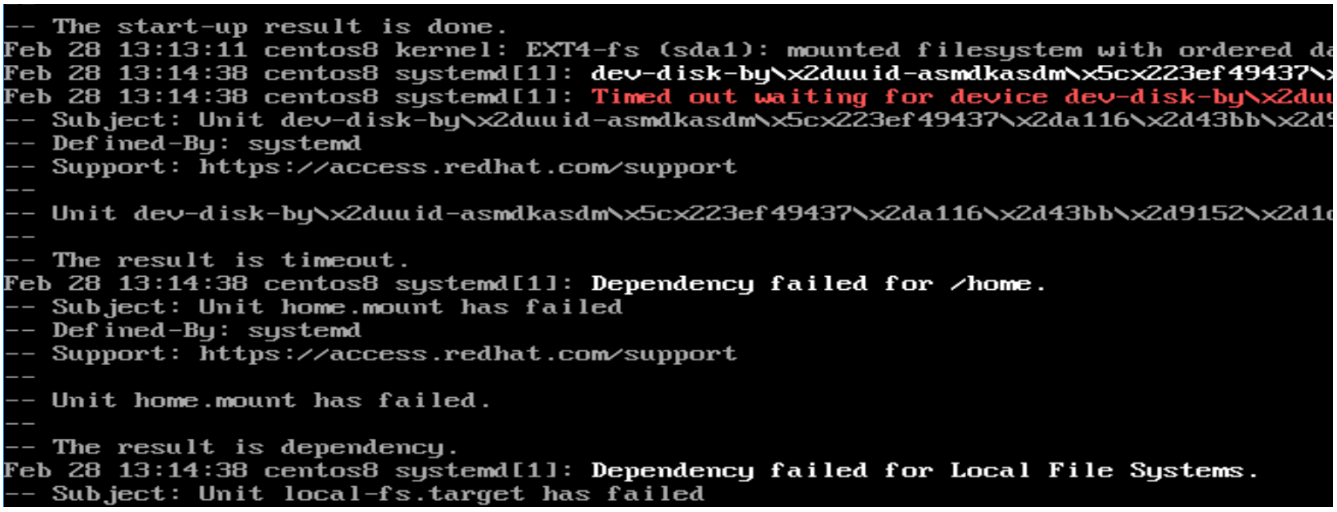
```
[user@centos8 ~]$ df -h
Filesystem                   Size  Used Avail Use% Mounted on
devtmpfs                     886M     0  886M   0% /dev
tmpfs                        914M     0  914M   0% /dev/shm
tmpfs                        914M  9.6M  904M   2% /run
tmpfs                        914M     0  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.

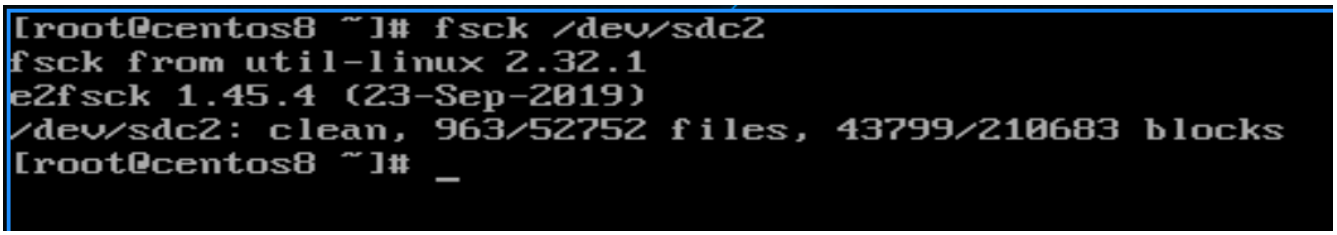
И так, при включении компьютера операционная система смотрит в fstab, видит какую файловую систему куда и как монтировать и делает это. Но что, если файловая система недоступна? Допустим, какие-то проблемы с диском или самой файловой системой, а может опечатка в fstab. В таких случаях операционная система 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                   ext
/dev/mapper/cl_centos8-swap 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.



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.



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. Окей, теперь можем исправить наш fstab - nano /etc/fstab, перезагрузиться и убедиться, что всё работает – reboot.

```
[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
drwxr-xr-x.  20 root root 3380 Feb 28 13:42 dev
drwxr-xr-x. 134 root root 8192 Feb 28 12:43 etc
drwxr-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
drwxr-xr-x.   2 root root    6 May 11  2019 media
drwxr-xr-x.   2 root root    6 May 11  2019 mnt
drwxr-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
drwxr-xr-x.  42 root root 1260 Feb 28 13:44 run
lrwxrwxrwx.   1 root root    8 May 11  2019 sbin -> usr/sbin
drwxr-xr-x.   2 root root    6 May 11  2019 srv
dr-xr-xr-x.  13 root root    0 Feb 28 13:42 sys
drwxrwxrwt.  20 root root 4096 Feb 28 13:44 tmp
drwxr-xr-x.  12 root root  144 Oct  2 11:24 usr
drwxr-xr-x.  22 root root 4096 Jan 10 19:56 var
[user@centos8 ~]$
```

Также, у многих возникает вопрос – а какие директории стоит выносить на отдельные файловые системы (ll /). 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.