```
[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 том, 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-a, 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 монтироваровать 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/sy
stem/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 системной шине 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.

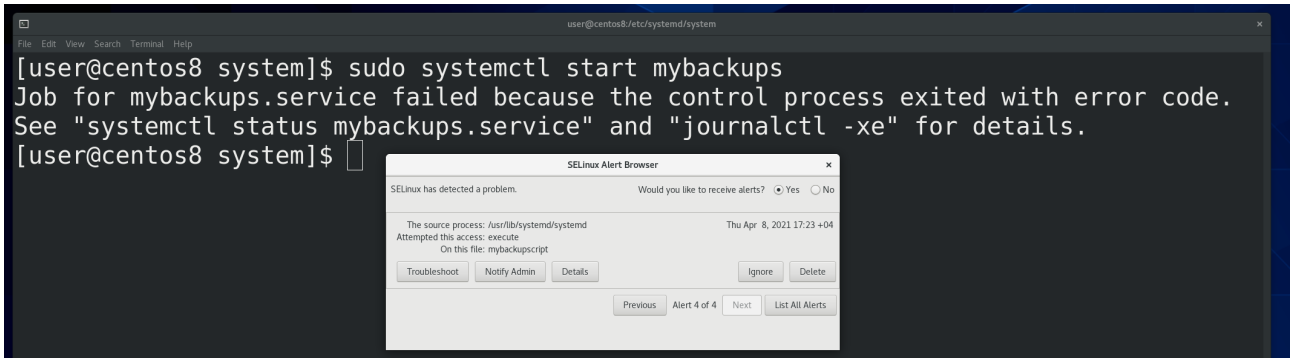
```
  GNU nano 2.9.8                          mybackups.service

[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. 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 - посылать стандартный вывод и ошибки в syslog, чтобы потом смотреть и сортировать через journalctl - 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, примонтирована 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 ~/mybackupscript /usr/local/bin/
[user@centos8 system]$ sudo nano mybackups.service
[user@centos8 system]$ grep ExecStart mybackups.service
ExecStart=/usr/local/bin/mybackupscript
[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 ~/mybackupscript /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]: ./mybackupscript
```

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
```

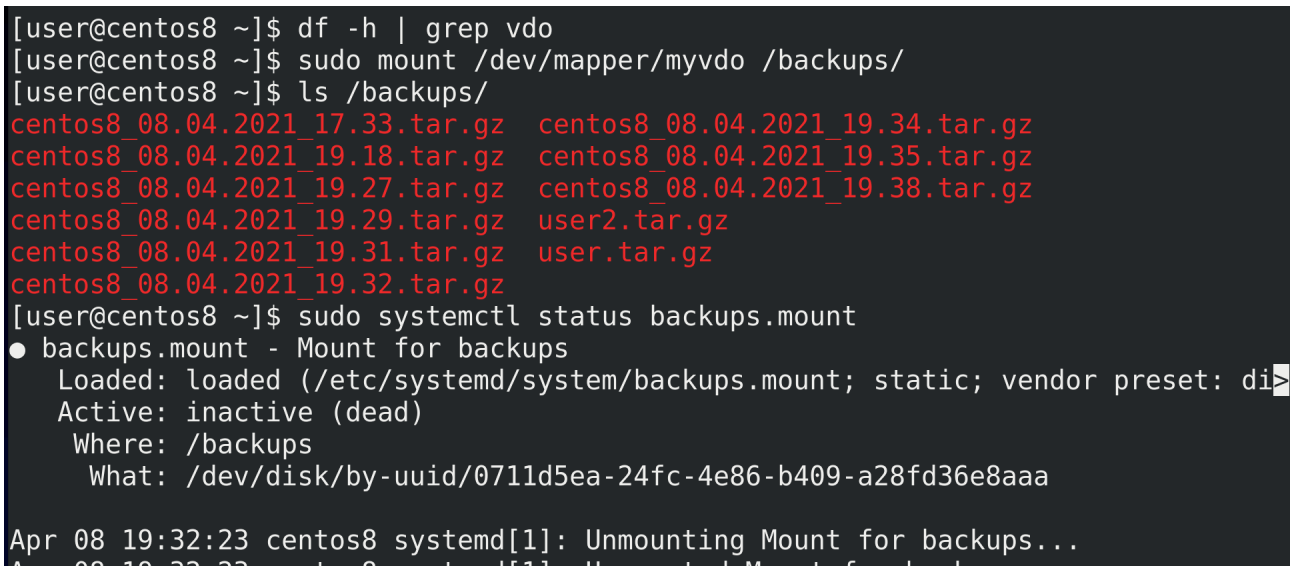
Их также можно посмотреть через journald - sudo journalctl -eu mybackups.

```
  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.



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
`OnActiveSec=`	X	This defines a timer relative to the moment the timer is activated.
`OnBootSec=`	X	This defines a timer relative to when the machine boots up.
`OnStartupSec=`	X	This defines a timer relative to when the service manager first starts. For system timer units, this is very similar to `OnBootSec=`, 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.
`OnUnitActiveSec=`	X	This defines a timer relative to when the timer that is to be activated was last activated.
`OnUnitInactiveSec=`	X	This defines a timer relative to when the timer that is to be activated was last deactivated.
`OnCalendar=`		This defines real-time (i.e., wall clock) timers with calendar event expressions. See `systemd.time(7)` for more information on the syntax of calendar event expressions. Otherwise, the semantics are similar to `OnActiveSec=` 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/myback
ups.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 символическая 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 empt>
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 empt>
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 empt>
```

А дальше как обычно - перечитываем конфиг - sudo systemctl daemon-reload. Посмотрим статус юнита - sudo systemctl status backups.automount - работает. Попробуем стопнуть 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 встречаются 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.