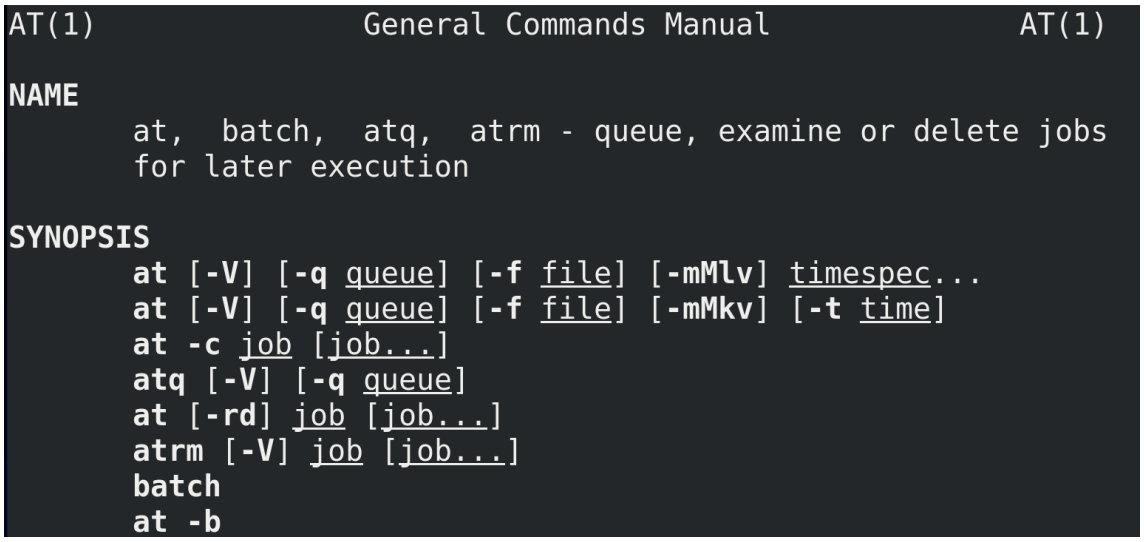
В задачи администратора входит настройка и поддержка IT инфраструктуры. Под поддержкой подразумевается регулярное выполнение обслуживающих задач, например, проверка состояния серверов, обновление операционной системы, бэкапы и всё такое. При изучении скриптов мы разобрались как можно упростить некоторые 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".



Первая утилита - at - man 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 или at -l. Чтобы увидеть детали задачи, пишем - at -c 3 - номер задачи. Команды выполнятся 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 j
 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 посылать 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  men-
       tioned 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 7
 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,
# |  |  |  |  |
# *  *  *  *  * 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.

```
user@centos8:~
File  Edit  View  Search  Terminal  Help
[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. e - 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. 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
```

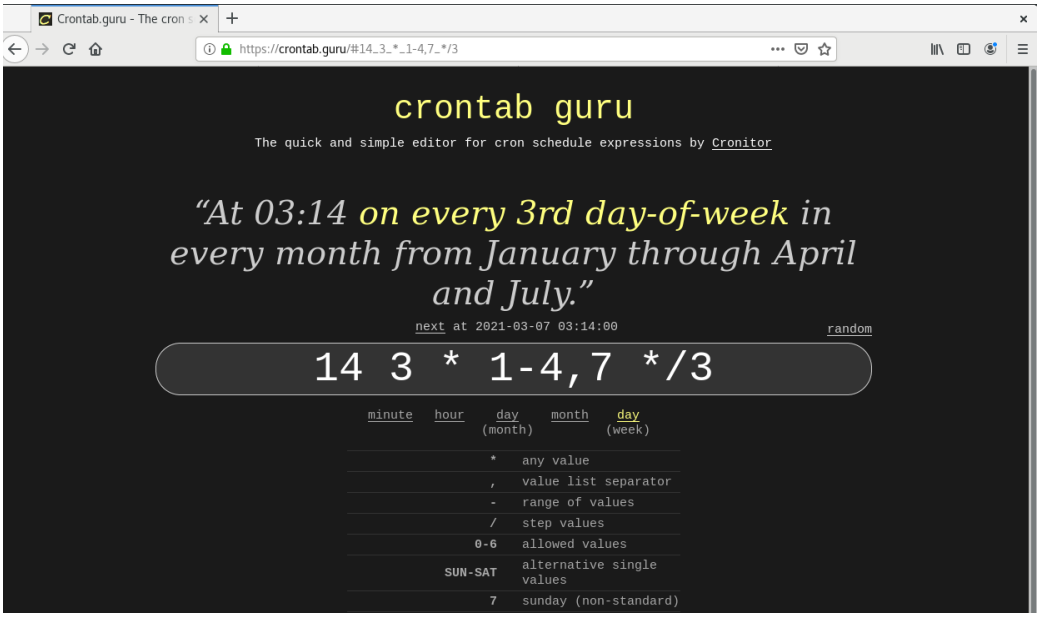
Предположим, мне нужно выполнять команду backup в каждое воскресенье. Для этого мне нужно определиться со временем - во сколько часов и минут. Допустим в 23:30. Тогда в качестве минут выставляю 30, в качестве часов - 23. День месяца нас не интересует, потому что бэкап у нас по воскресеньям, а это может быть любым днём, поэтому ставим звёздочку. Также нас не интересует конкретный месяц, у нас бэкап должен идти каждый месяц, поэтому также звёздочку. Остаётся 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. 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



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 ] ; 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 настроите бэкап раз в день, то, если компьютер будет выключен в запланированное для бэкапа время, бэкап будет сделать 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-collec
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-summar
```

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 ~]$ systemd-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.