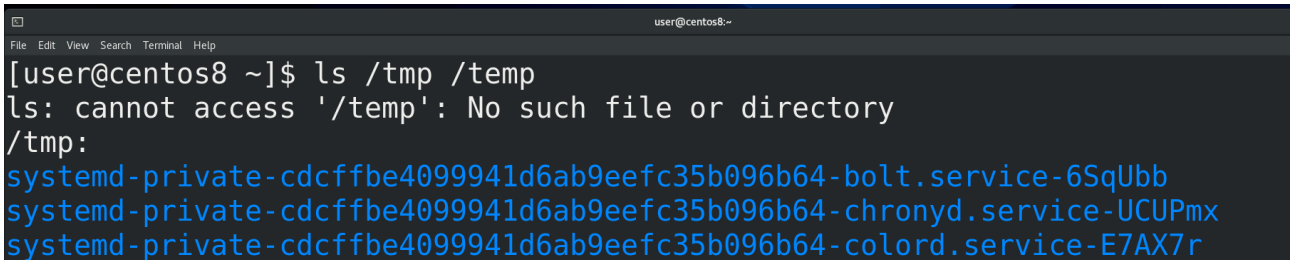
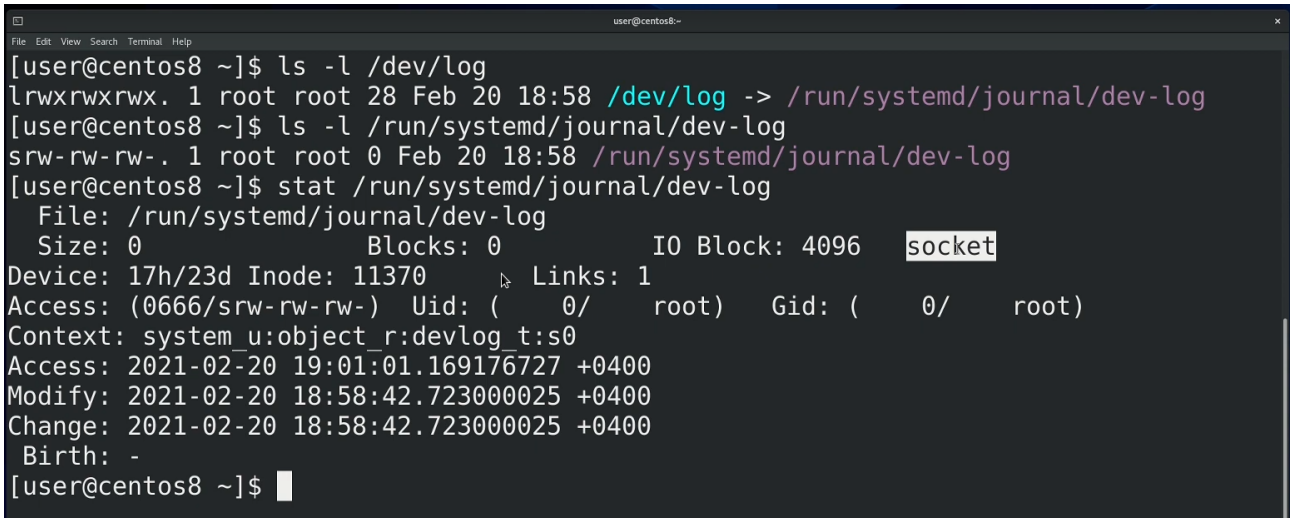


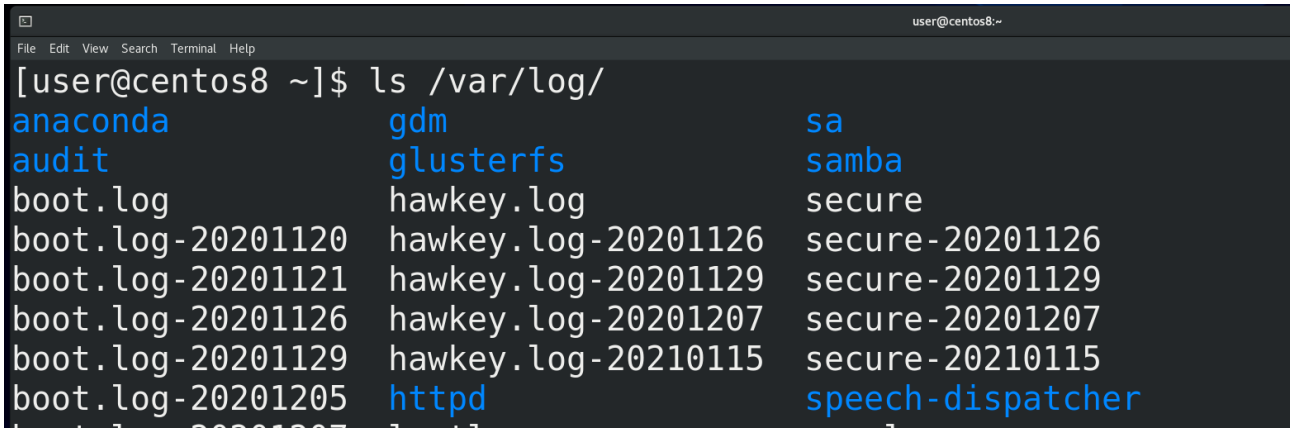
Когда вы запускаете какую-то программу в текстовом интерфейсе, как правило, вы видите вывод этой команды, будь то нужная информация или какие-то ошибки - ls /tmp /temp. Но всякие демоны запускаются сервисным менеджером и работают на фоне, из-за чего их вывод не виден пользователю. Как я говорил, настройка 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.



И так, программы посылают 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 посылать 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 используют 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.

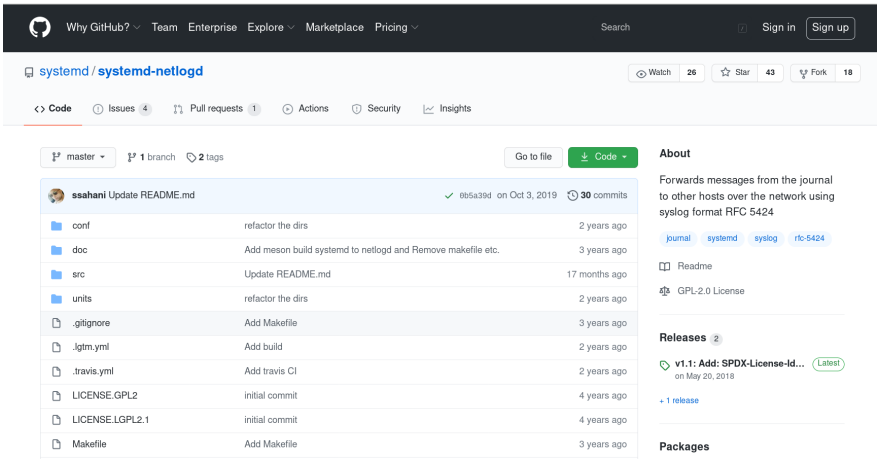
До journald основным демоном для syslog был rsyslog. Собственно, он стоял за /dev/log и собирал все логи, а дальше, на основе своих настроек, решал, что делать с этими логами - писать в файлы, выводить на экран, посылать по почте и всё такое. На самом деле rsyslog и сейчас работает, вместе с journald - systemctl status rsyslog, systemctl status systemd-journald. Только теперь логи собирает journald, а потом делиться 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.



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 (2592
59) is greater than comparison timestamp (259257).  This most likely represents a buggy client send
ing 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.react
ivated.Fprint' unit='fprintd.service' requested by ':1.430' (uid=0 pid=3995 comm="sudo tail /var/lo
g/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, справедливости ради, journald может посылать 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 посылаться 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 связанны 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-а, 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.

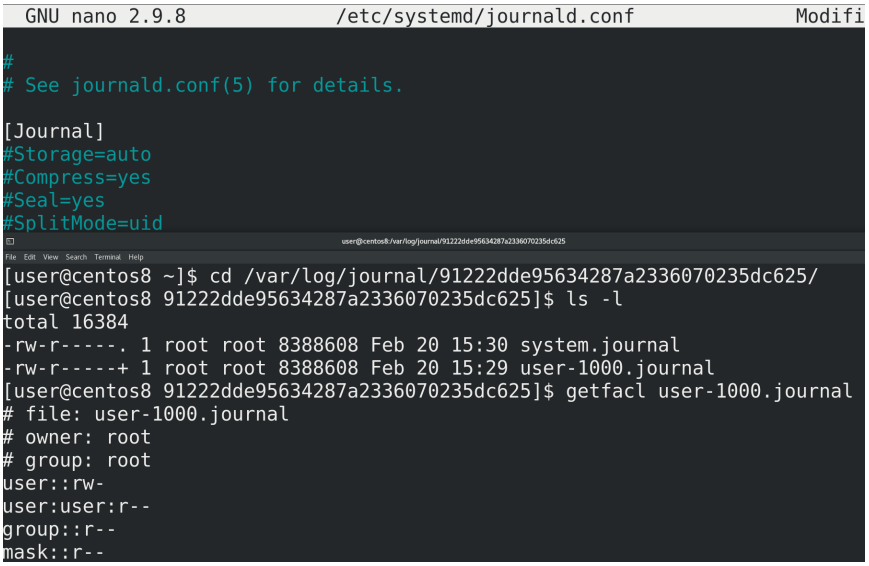
```
  GNU nano 2.9.8                /etc/systemd/journald.conf

#
# 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.

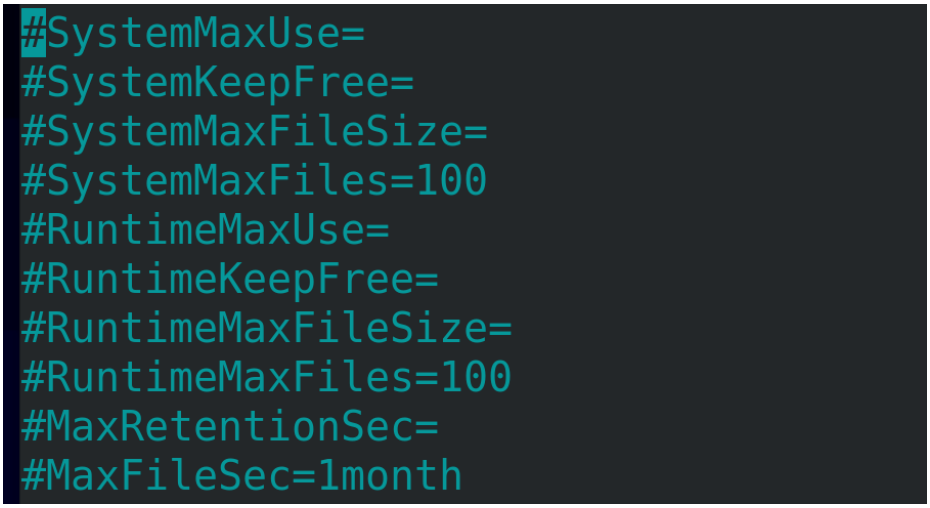
```
[Journal]
#Storage=auto
```

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



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.



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 объёма 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 log]$ ls /var/log/
anaconda           gdm                README
audit              glusterfs          sa
boot.log           hawkey.log         samba
boot.log-20201121  hawkey.log-20201129 secure
boot.log-20201126  hawkey.log-20201207 secure-20201129
boot.log-20201129  hawkey.log-20210115 secure-20201207
boot.log-20201205  hawkey.log-20210220 secure-20210115
boot.log-20201207  httpd               secure-20210220
```

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
btmp            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 +0>
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 (moc
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[
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 0x0000000000000000-0x000000000
```

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 202
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.

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.