

Log rotation and configuration



```
sudo bash -c "cat > /etc/logrotate.d/rsyslog" <<EOF
/var/log/syslog
{
rotate 168
maxsize 3M
missingok
notifempty
delaycompress
compress
postrotate
    /usr/lib/rsyslog/rsyslog-rotate
endscript
}
/var/log/mail.info
/var/log/mail.warn
/var/log/mail.err
/var/log/mail.log
/var/log/daemon.log
/var/log/kern.log
/var/log/auth.log
/var/log/user.log
/var/log/lpr.log
/var/log/cron.log
/var/log/debug
/var/log/messages
{
rotate 4
weekly
missingok
notifempty
compress
delaycompress
sharedscripts
postrotate
    /usr/lib/rsyslog/rsyslog-rotate
endscript
}
EOF
```

```
sudo bash -c "cat > /etc/cron.hourly/logrotate" <<EOF
#!/bin/sh
```

```
# this cronjob persists removals (but not purges)
if [ ! -x /usr/sbin/logrotate ]; then
    exit 0
fi

/usr/sbin/logrotate /etc/logrotate.conf
EXITVALUE=$?
if [ $EXITVALUE != 0 ]; then
    /usr/bin/logger -t logrotate "ALERT exited abnormally with [$EXITVALUE]"
fi
exit $EXITVALUE
EOF
```

```
> sudo chmod +x /etc/cron.hourly/logrotate
```

```
sudo bash -c "cat > /etc/rsyslog.d/50-default.conf" <<EOF
# Default rules for rsyslog.
#
# For more information see rsyslog.conf(5) and /etc/rsyslog.conf

#
# First some standard log files. Log by facility.
#
auth,authpriv.*      /var/log/auth.log
*.*;auth,authpriv.none -/var/log/syslog
#cron.*              /var/log/cron.log
#daemon.*             -/var/log/daemon.log
kern.*                -/var/log/kern.log
#lpr.*                -/var/log/lpr.log
mail.*                -/var/log/mail.log
#user.*               -/var/log/user.log

#
# Logging for the mail system. Split it up so that
# it is easy to write scripts to parse these files.
#
#mail.info             -/var/log/mail.info
#mail.warn             -/var/log/mail.warn
mail.err              /var/log/mail.err

#
# Some "catch-all" log files.
#
#*.=debug;\
#   auth,authpriv.none;\
```

```
# news.none;mail.none -/var/log/debug
#*.=info;*.=notice;*.=warn;\n
# auth,authpriv.none;\n
# cron,daemon.none;\n
# mail,news.none -/var/log/messages

#
# Emergencies are sent to everybody logged in.
#
*.emerg                :omusrmsg:*

#
# I like to have messages displayed on the console, but only on a virtual
# console I usually leave idle.
#
#daemon,mail.*;\n
# news.=crit;news.=err;news.=notice;\n
# *.=debug;*.=info;\n
# *.=notice;*.=warn    /dev/tty8
EOF
```

Explain this above code with realtime examples?



This setup is configuring **log rotation, log collection, and cron jobs** on a Linux system. Let's break it down with **real-time examples** for better understanding.

1. Log Rotation Configuration (`/etc/logrotate.d/rsyslog`)

Purpose:

This file tells the system how to manage log file sizes and rotations.

Key Configuration:

```
bash

/var/log/syslog
{
rotate 168
maxsize 3M
missingok
notifempty
delaycompress
compress
postrotate
    /usr/lib/rsyslog/rsyslog-rotate
endscript
}
```

- ``rotate 168`` → Keeps **168 rotated log files** before old ones are deleted (useful for long-term logging).
- ``maxsize 3M`` → If ``syslog`` reaches **3MB**, it rotates (ensures logs don't take up too much space).
- ``missingok`` → If the file is missing, it **won't show an error**.
- ``notifempty`` → Doesn't rotate empty log files.
- ``delaycompress`` → Delays compression until the next rotation.
- ``compress`` → Compresses older logs to save disk space.
- ``postrotate ... endscript`` → Runs ``/usr/lib/rsyslog/rsyslog-rotate`` **after rotation** to signal rsyslog to reload.

Example in Real-Time:

- A web server generates logs in ``/var/log/syslog``. If this file reaches **3MB**, it will be rotated.
- The system keeps the last **168 rotated logs**, ensuring older logs are available for **security auditing**.
- Older logs get **compressed** to save disk space.

Other Logs (``/var/log/auth.log``, ``/var/log/mail.log``, etc.)

```
bash

/var/log/mail.info
/var/log/mail.warn
/var/log/mail.err
/var/log/mail.log
/var/log/daemon.log
/var/log/kern.log
/var/log/auth.log
/var/log/user.log
/var/log/lpr.log
/var/log/cron.log
/var/log/debug
/var/log/messages
{
rotate 4
weekly
missingok
notifempty
compress
delaycompress
sharedscripts
postrotate
    /usr/lib/rsyslog/rsyslog-rotate
endscript
}
```

- These logs **rotate weekly** (``weekly``).
- **Only 4 rotations** are kept (``rotate 4``).
- **Compression is enabled** (``compress``).
- **Post-rotation script** signals rsyslog to reload.

Example in Real-Time:

- If ``auth.log`` (which logs SSH login attempts) becomes large, it rotates **every week**.
- Only the **last 4 weeks** of logs are kept, so old logs are deleted after a month.

2. Hourly Log Rotation Cron Job (``/etc/cron.hourly/logrotate``)

This script **automates log rotation** every hour.

Script:

```
bash

#!/bin/sh

# this cronjob persists removals (but not purges)
if [ ! -x /usr/sbin/logrotate ]; then
    exit 0
fi

/usr/sbin/logrotate /etc/logrotate.conf
EXITVALUE=$?
if [ $EXITVALUE != 0 ]; then
    /usr/bin/logger -t logrotate "ALERT exited abnormally with [$EXITVALUE]"
fi
exit $EXITVALUE
```

What It Does:

1. Checks if `logrotate` is installed (`/usr/sbin/logrotate`).
2. Runs `logrotate` using `/etc/logrotate.conf` to enforce the rules.
3. If `logrotate` fails, logs an error message using `logger`.

Example in Real-Time:

- If a log file **reaches 3MB at 2:15 PM**, this cron job will **rotate it in the next hour (3:00 PM)**.
- If `logrotate` fails, it writes an error like:

```
csharp

Jan 29 15:00:01 server logrotate: ALERT exited abnormally with [1]
```

3. Rsyslog Logging Configuration (`/etc/rsyslog.d/50-default.conf`)

This file **controls how rsyslog collects logs and writes them to different files**.

Key Configuration:

```
bash

auth,authpriv.*          /var/log/auth.log
*.*;auth,authpriv.none   -/var/log/syslog
kern.*                   -/var/log/kern.log
mail.*                   -/var/log/mail.log
mail.err                 /var/log/mail.err
*.emerg                  :omusrmsg:*
```

Breakdown of Key Rules:

- ``auth,authpriv.*`` → **Logs all authentication logs** (``auth.log``).
- ``*.*;auth,authpriv.none -/var/log/syslog``
→ Logs **everything except authentication logs** to ``syslog``.
- ``kern.*`` → **Kernel logs** (``kern.log``).
- ``mail.*`` → **Logs all mail events** (``mail.log``).
- ``mail.err`` → **Logs only mail errors** (``mail.err``).
- ``*.emerg :omusrmsg:*`` → Sends **emergency messages** to all logged-in users.

Example in Real-Time:

1. **Failed SSH login attempt** → Logged in ``/var/log/auth.log``.
2. **System reboot** → Logged in ``/var/log/syslog``.
3. **Kernel panic error** → Logged in ``/var/log/kern.log``.
4. **Mail system error** → Sent to ``/var/log/mail.err``.
5. **Critical system failure** → Sent to all **logged-in users' terminals**.

Summary:

1. **Log Rotation** (``logrotate``)
 - Prevents logs from growing too large.
 - Compresses old logs for storage efficiency.
 - Automatically removes old logs beyond the retention period.
2. **Hourly Cron Job** (``cron.hourly/logrotate``)
 - Runs log rotation every hour.
 - Ensures logs don't get too big even if heavy logging occurs.
3. **Rsyslog Configuration** (``rsyslog.d/50-default.conf``)
 - Controls **where different types of logs** go.
 - Ensures logs are **categorized properly** for easy monitoring.

Final Real-World Example

Imagine you're an **IT admin** for a **web server**. Your server:

- Handles **SSH logins** (``auth.log``).
- Runs a **mail service** (``mail.log``).
- Logs **kernel events** (``kern.log``).

With this setup:

1. **Logs are properly categorized** → You can easily check ``auth.log`` for failed logins.
2. **Logs don't fill up the disk** → ``logrotate`` prevents logs from exceeding **3MB**.
3. **Log rotation runs automatically** every hour **to avoid performance issues**.
4. **Emergency messages** are **sent to all users** if something critical happens.

This setup ensures your system logs **stay clean, organized, and manageable** .

Would you like to customize this for your specific use case? 



In openwrt that message log how many messages are stored



In **OpenWRT**, the number of messages stored in the **system log** (`logread`) depends on the following factors:

1. Log Storage Mechanism in OpenWRT

- OpenWRT stores logs **in RAM** using `logd`, meaning logs are **lost on reboot**.
- The log buffer size is defined by `log_size` in `/etc/config/system`.

2. Checking the Current Log Size

Run the following command:

```
sh

uci show system.@system[0].log_size
```

This will return the log buffer size in **KB** (default is 64 KB).

3. How Many Messages Can Be Stored?

- The number of log messages stored depends on the **log level (verbosity)** and the **size of each log entry**.
- If each log entry is **150 bytes** on average, a **64 KB buffer** can store:

```
bash

64 KB / 150 bytes ≈ 436 log entries
```

- If you increase the buffer to **256 KB**, you can store:

```
bash

256 KB / 150 bytes ≈ 1,745 log entries
```

4. Increasing Log Storage in OpenWRT

If you want to store **more logs**, increase the buffer size:

```
sh

uci set system.@system[0].log_size='256' # Increase to 256KB
uci commit system
```

```
/etc/init.d/log restart
```

Or, you can send logs to an **external storage** (like a USB or a remote syslog server).

Would you like help setting up **persistent logs** on OpenWRT? 🚀



If i want to store multiple files in log then what can i do ?



In **OpenWRT**, logs are stored in **RAM (volatile memory)** by default using `logd`, which means logs are lost after a reboot. However, if you want to store logs in **multiple files** and make them **persistent**, you have a few options:

1. Enable Persistent Logging (Store Logs on Disk)

By default, OpenWRT only logs in **RAM** (`/var/log`). To store logs in **multiple files** on persistent storage, follow these steps:

Step 1: Enable Persistent Logging

Modify `/etc/config/system`:

```
sh

uci set system.@system[0].log_file='/mnt/storage/system.log' # Change log file location
uci set system.@system[0].log_size='256' # Increase log buffer size (default is 64 KB)
uci commit system
/etc/init.d/log restart
```

- Replace `/mnt/storage/system.log` with a persistent location like `/overlay/logs/` or a mounted USB drive.
- This will store logs **permanently** even after reboots.

2. Use `rsyslog` to Store Multiple Log Files

If you need **multiple log files** for different services (e.g., system logs, firewall logs, VPN logs), install and configure **rsyslog**.

Step 1: Install `rsyslog`

```
sh

opkg update
```



```
opkg install rsyslog
```

Step 2: Configure `rsyslog`

Edit `/etc/rsyslog.conf`:

```
sh

module(load="imuxsock") # Unix socket for local processes
module(load="imklog")   # Kernel log messages

# Store logs in multiple files
auth,authpriv.*        /mnt/logs/auth.log
kern.*                  /mnt/logs/kern.log
daemon.*                /mnt/logs/daemon.log
syslog.*                /mnt/logs/syslog.log
```

Create the log folder:

```
sh

mkdir -p /mnt/logs
chmod 755 /mnt/logs
```

Restart rsyslog:

```
sh

/etc/init.d/rsyslog restart
```

Now, logs will be stored in **separate files** inside `/mnt/logs/`.

3. Use `logrotate` to Manage Multiple Log Files

If logs **grow too large**, you need `logrotate` to rotate and compress old logs.

Step 1: Install `logrotate`

```
sh

opkg update
opkg install logrotate
```

Step 2: Create Logrotate Configuration

Edit `/etc/logrotate.conf`:

```
sh

/mnt/logs/*.log {
    rotate 4
    weekly
    missingok
    notifempty
    compress
    delaycompress
    postrotate
        /etc/init.d/rsyslog restart
    endscript
}
```

- Logs will rotate **weekly**.
- Keeps **4 old logs** before deleting the oldest one.
- Compressed logs save space.

Run logrotate manually to test:

```
sh

logrotate -f /etc/logrotate.conf
```

4. Send Logs to a Remote Syslog Server

If you have a **central logging server**, you can forward logs using ``rsyslog``.

Edit ``/etc/rsyslog.conf``:

```
sh

*. * @@192.168.1.100:514
```

Replace ``192.168.1.100`` with your syslog server IP.

Restart rsyslog:

```
sh

/etc/init.d/rsyslog restart
```

Final Summary

| Method | Purpose |
|---------------------------|---|
| Persistent Logging | Stores logs on disk (<code>`/mnt/storage/system.log`</code>) |
| rsyslog | Saves logs in multiple files (<code>`syslog.log`</code> , <code>`auth.log`</code> , etc.) |

| Method | Purpose |
|-----------------------|---|
| logrotate | Rotates logs to prevent excessive storage usage |
| Remote Logging | Sends logs to a central logging server |

Would you like help with a specific method? 🚀



```
# Example of default configuration:
#
# module(load="imuxsock")
# module(load="imklog")
# module(load="imudp")
# input(type="imudp" port="514")
# $ActionFileDefaultTemplate Rsyslog_TraditionalFileFormat
# *.info;mail.none;authpriv.none;cron.none /var/log/messages
# authpriv.*          /var/log/secure
# mail.*               /var/log/maillog
# cron.*               /var/log/cron
# local7.*             /var/log/boot.log
```

Can You Explain this?



Yes! This is a basic **rsyslog** configuration example. Let me break it down for you **line by line** with real-time examples.

1. Module Loading (`module(load="...")`)

These lines load **rsyslog modules**, which are needed to collect log messages from different sources.

```
sh

module(load="imuxsock") # Enables logging from local applications via Unix sockets (e.g.,
systemd logs)
module(load="imklog")   # Enables logging of kernel messages (dmesg logs)
module(load="imudp")    # Enables UDP-based syslog reception
input(type="imudp" port="514") # Listens for incoming syslog messages on UDP port 514
```

Example Usage:

- If a local application writes logs, **imuxsock** will capture them.
- Kernel logs (**dmesg**) are captured using **imklog**.
- If a remote device (e.g., OpenWRT router) sends logs via **UDP 514**, **imudp** will capture them.

2. Log File Format

```
sh

$ActionFileDefaultTemplate RSYSLLOG_TraditionalFileFormat
```

This sets the **default log file format** to a **traditional syslog format**.

Example Log Entry with Traditional Format:

```
arduino

Jan 29 12:34:56 server1 systemd[1]: Started OpenVPN Server.
```

If this line were missing, ``rsyslog`` might use a different format.

3. Logging Rules

These rules define **which logs** should go into **which files**.

```
sh

*.info;mail.none;authpriv.none;cron.none /var/log/messages
```

- ``*.info`` → All logs with **info level** and above (``notice``, ``warn``, ``err``, etc.).
- ``mail.none`` → **Excludes** mail logs.
- ``authpriv.none`` → **Excludes** authentication logs.
- ``cron.none`` → **Excludes** cron job logs.
- Stores logs in: ``/var/log/messages``.

Example Log in ``/var/log/messages``:

```
less

Jan 29 12:45:00 server1 systemd[1]: NetworkManager started.
```

4. Authentication Logs

```
sh

authpriv.* /var/log/secure
```

- ``authpriv.*`` → **All authentication-related logs** (e.g., SSH, sudo).
- Stored in: ``/var/log/secure``.

Example Log in ``/var/log/secure``:

```
csharp
```

```
Jan 29 13:05:12 server1 sshd[2456]: Failed password for root from 192.168.1.10
```

Useful for **tracking unauthorized login attempts**.

5. Mail Logs

```
sh
```

```
mail.* /var/log/maillog
```

- `mail.*` → **All mail-related logs** (Postfix, Sendmail).
- Stored in: `/var/log/maillog`.

Example Log in `/var/log/maillog`:

```
sql
```

```
Jan 29 13:15:45 server1 postfix/smtp[1500]: Delivered mail to user@example.com
```

Useful for **debugging email issues**.

6. Cron Logs

```
sh
```

```
cron.* /var/log/cron
```

- `cron.*` → **All cron job logs** (Scheduled tasks).
- Stored in: `/var/log/cron`.

Example Log in `/var/log/cron`:

```
less
```

```
Jan 29 13:30:00 server1 CRON[1234]: Running daily backup script.
```

Useful for **verifying scheduled jobs ran correctly**.

7. Boot Logs

```
sh

local7.* /var/log/boot.log
```

- ``local7.*`` → **Captures boot messages.**
- Stored in: ``/var/log/boot.log``.

Example Log in ``/var/log/boot.log``:

```
perl

Jan 29 13:40:01 server1 kernel: Initializing system services...
```

Useful for **debugging boot-time issues.**


Summary Table

| Log Type | Stored In |
|--|----------------------------------|
| System logs (<code>`info`</code> level) | <code>`/var/log/messages`</code> |
| Authentication logs (SSH, sudo) | <code>`/var/log/secure`</code> |
| Mail logs (Postfix, Sendmail) | <code>`/var/log/maillog`</code> |
| Cron job logs | <code>`/var/log/cron`</code> |
| Boot logs | <code>`/var/log/boot.log`</code> |

How Can This Be Used in OpenWRT?

If you want to **store logs in multiple files in OpenWRT**, you can modify ``/etc/rsyslog.conf`` to:

- Capture firewall logs separately (``/var/log/firewall.log``).
- Separate OpenVPN logs (``/var/log/openvpn.log``).
- Forward logs to a **central logging server**.

Would you like help with custom OpenWRT log configurations? 



```
sudo bash -c "cat > /etc/rsyslog.d/50-default.conf" <<EOF
# Default rules for rsyslog.
#
# For more information see rsyslog.conf(5) and /etc/rsyslog.conf

#
# First some standard log files. Log by facility.
#
auth,authpriv.* /var/log/auth.log
```

```

*. *;auth,authpriv.none    -/var/log/syslog
#cron.*                    /var/log/cron.log
#daemon.*                 -/var/log/daemon.log
kern.*                    -/var/log/kern.log
#lpr.*                    -/var/log/lpr.log
mail.*                    -/var/log/mail.log
#user.*                   -/var/log/user.log

#
# Logging for the mail system. Split it up so that
# it is easy to write scripts to parse these files.
#
#mail.info                -/var/log/mail.info
#mail.warn                -/var/log/mail.warn
mail.err                  /var/log/mail.err

#
# Some "catch-all" log files.
#
#*. =debug;\
#  auth,authpriv.none;\
#  news.none;mail.none    -/var/log/debug
#*. =info;*. =notice;*. =warn;\
#  auth,authpriv.none;\
#  cron,daemon.none;\
#  mail,news.none        -/var/log/messages

#
# Emergencies are sent to everybody logged in.
#
*.emerg                   :omusrmsg:*

#
# I like to have messages displayed on the console, but only on a virtual
# console I usually leave idle.
#
#daemon,mail.*;\
#  news.=crit;news.=err;news.=notice;\
#  *. =debug;*. =info;\
#  *. =notice;*. =warn    /dev/tty8
EOF

```

Remove the commend line



Here's the cleaned-up version of your `50-default.conf` file with all commented lines removed:

```
sh

sudo bash -c "cat > /etc/rsyslog.d/50-default.conf" <<EOF
auth,authpriv.*                /var/log/auth.log
*.*;auth,authpriv.none         -/var/log/syslog
kern.*                          -/var/log/kern.log
mail.*                          -/var/log/mail.log
mail.err                        /var/log/mail.err
*.emerg                         :omusrmsg:*
EOF
```

This version keeps only the active logging rules and removes all comments (`#`). Let me know if you need any modifications! 🚀



2. Hourly Log Rotation Cron Job (/etc/cron.hourly/logrotate)

How we achieve this in OpenWRT?



Achieving Hourly Log Rotation in OpenWRT

OpenWRT does not use `cron.hourly` like traditional Linux distributions. Instead, it uses `crontab` managed by `/etc/crontabs/root` or **procd-based log rotation**. Here's how you can set up hourly log rotation in OpenWRT.

📌 Method 1: Using Crontab (Recommended)

OpenWRT uses `crontabs/root` instead of `/etc/cron.hourly`. You can manually add an **hourly log rotation** entry.

Step 1: Edit OpenWRT Crontab

Run the following command:

```
sh

crontab -e
```

Then add this line at the bottom:

```
sh

0 * * * * /usr/sbin/logrotate /etc/logrotate.conf
```

- `0 * * * *` → Runs every hour at **minute 0**.

- `/usr/sbin/logrotate /etc/logrotate.conf` → Runs log rotation.

Step 2: Restart Cron Service

```
sh

/etc/init.d/cron restart
```

This ensures the cron job is loaded.

Method 2: Using procd-based Log Rotation (Built-in)

If you prefer OpenWRT's built-in log rotation, you can configure it via `/etc/config/system`.

Step 1: Edit System Log Configuration

Modify `/etc/config/system`:

```
sh

uci set system.@system[0].log_size='512' # Max log file size (KB)
uci set system.@system[0].log_file='/var/log/messages' # Log file path
uci set system.@system[0].log_rotate='6' # Keep 6 rotated logs
uci commit system
```

Step 2: Restart System Logging

```
sh

/etc/init.d/log restart
```

Method 3: Custom Script in `/etc/hotplug.d/cron/` (Alternative)

If you want a **custom logrotate script**, create a script in `/etc/hotplug.d/cron/`.

Step 1: Create Logrotate Script

```
sh

mkdir -p /etc/hotplug.d/cron
nano /etc/hotplug.d/cron/99-logrotate
```

Add the following:

```
sh

#!/bin/sh
[ "$ACTION" = "cron" ] && /usr/sbin/logrotate /etc/logrotate.conf
```

Step 2: Make It Executable

```
sh

chmod +x /etc/hotplug.d/cron/99-logrotate
```

This ensures log rotation is triggered by OpenWRT's cron system.

Which Method Should You Use?

| Method | Pros | Cons |
|--|-----------------------------------|--------------------------------|
| Crontab (`crontab -e`) | Simple, traditional way | May not persist across reboots |
| Procd-based Log Rotation (`uci set`) | Uses OpenWRT built-in features | Less flexible for custom rules |
| Hotplug Cron (`/etc/hotplug.d/cron/`) | Integrated with OpenWRT's hotplug | More complex to set up |

✅ **Recommended:** Use **Method 1 (Crontab)** if you want a simple and reliable way to rotate logs hourly.

Let me know if you need additional customization! 🚀



I still confuse ?
Why syslog?
why logrotate and why cron?
Take one example and explain everything in detail?



No worries! Let's break everything down with an example and explain **why** each component (syslog, logrotate, and cron) is needed, and how they fit into the process.

Example Scenario:

Imagine you have a router running OpenWRT, and it needs to log various system events such as network connections, firewall logs, and kernel messages. You want to **log these events** and **rotate the logs hourly** to prevent them from filling up the device's storage.

Step-by-Step Breakdown:

1. Why Syslog?

Syslog is a protocol and system service for **logging messages** from various services and applications. It's used by most Unix-like systems (including OpenWRT) to collect and store log messages.

Why Do We Need Syslog?

- **Centralized Logging:** Syslog allows all the logs to be stored in one place, making it easier to **track and debug** system issues.
- **Different Sources:** Syslog can capture logs from various services, including kernel messages, application logs, user logs, and network logs.

How Syslog Works in Our Example:

Let's say we have the following syslog configuration (stored in `/etc/rsyslog.conf`):

```
sh

kern.*                -/var/log/kern.log
authpriv.*            /var/log/auth.log
*. *                  /var/log/syslog
```

- **Kernel logs** (`kern.*`) are stored in `/var/log/kern.log`.
- **Authentication logs** (`authpriv.*`) go to `/var/log/auth.log`.
- **All other logs** (`*.*`) are sent to `/var/log/syslog`.

Now, every time a new log event occurs (e.g., a user logs in or a network interface goes up), **syslog** captures that event and writes it to the appropriate log file.

Example Log Entry in `/var/log/syslog`:

```
less

Jan 29 14:22:00 router kernel: [123456.789] eth0: link up
Jan 29 14:23:00 router sshd[3456]: Accepted password for user from 192.168.1.100
```

2. Why Logrotate?

Logrotate is a utility used to **rotate, compress, and manage log files** so that they do not consume too much disk space. Logs can quickly grow to large sizes, which can fill up the storage on embedded systems like OpenWRT.

Why Do We Need Logrotate?

- **Prevent Disk Space Overflow:** Logs grow over time, and without log rotation, they can fill up the disk.
- **Organize Logs:** Logrotate helps organize logs by creating new files periodically (daily, weekly, etc.) and compressing the old logs to save space.
- **Keep Logs Manageable:** You can set a limit on how many rotated log files you want to keep.

How Logrotate Works in Our Example:

Let's say we want to **rotate the logs hourly**. Here's how you'd configure it in `/etc/logrotate.conf`:

```
sh

/var/log/syslog
{
    rotate 24          # Keep 24 rotated logs (1 full day of hourly logs)
    hourly            # Rotate the logs every hour
    compress           # Compress old log files
    missingok          # Ignore if the log file is missing
    notifempty         # Don't rotate if the log is empty
    create 640 root root # Create new logs with specific permissions
}
```

With this setup:

- `rotate 24` means you'll keep the logs for 24 hours (since we're rotating every hour).
- `compress` means older logs (after rotation) will be compressed to save space.
- `missingok` means if the log file is missing, it won't throw an error.
- `notifempty` means empty logs won't be rotated.

This ensures that after each hour, `logrotate` will:

- Move `/var/log/syslog` to `/var/log/syslog.1`, compress the older logs (e.g., `/var/log/syslog.1.gz`), and create a fresh `/var/log/syslog`.
- This continues hourly until the logs reach the `rotate` limit (in this case, 24 rotations = 1 full day).

3. Why Cron?

Cron is a time-based job scheduler in Unix-like operating systems. It allows you to schedule jobs (tasks) to run **at specific times** (hourly, daily, weekly, etc.).

Why Do We Need Cron?

- **Automating Tasks:** Cron ensures tasks like log rotation happen automatically at regular intervals without manual intervention.
- **Timing Control:** You can configure when exactly a task (like log rotation) should run (e.g., every hour, every day at midnight).

How Cron Works in Our Example:

In OpenWRT, to rotate logs every hour, we use cron. We create an entry in `/etc/crontabs/root` that runs `logrotate` at the start of each hour:

```
sh

0 * * * * /usr/sbin/logrotate /etc/logrotate.conf
```

- `0 * * * *` means **at minute 0 of every hour**, i.e., on the hour, every hour.

- `/usr/sbin/logrotate /etc/logrotate.conf` runs the logrotate utility, which will read the rotation rules from `/etc/logrotate.conf` and rotate the logs.

Cron will automatically run this task hourly, ensuring that logs are rotated and managed without manual intervention.

Putting It All Together

1. **Syslog** gathers all the logs from various services (like kernel, authentication, etc.) and writes them to log files (e.g., `/var/log/syslog`).
 2. **Logrotate** takes over periodically (set by cron) to rotate these logs, ensuring that they don't get too large, compressing older logs to save space, and keeping them organized.
 3. **Cron** schedules the log rotation task to run every hour (or at other intervals) automatically, without needing to intervene manually.
-

Real-time Example:

Let's say your router runs OpenWRT and logs network events every minute. Over a week, your logs can grow large. Without log rotation, the log file `/var/log/syslog` might reach several GBs in size.

With the setup:

- **Syslog** collects events like network connections, user logins, etc.
 - **Cron** triggers log rotation every hour.
 - **Logrotate** compresses and rotates these logs, keeping a manageable size of log files, and your router won't run out of storage space!
-

Summary:

- **Syslog** collects log messages.
- **Logrotate** manages and organizes log files, preventing them from growing too large.
- **Cron** automates the periodic execution of log rotation tasks.

This entire process ensures your system logs are kept organized, manageable, and your router's storage remains functional without being overwhelmed by log files.

Let me know if this clears up your confusion! 😊