# SystemLogging

April 15, 2024

## 1 System Logging?

System logging is a method of tracking events that happen on a computer system. These events are recorded in a log file, which is a simple text file that lists activities and occurrences within an operating system, an application, a server, or other software.

System logs can provide valuable information for troubleshooting issues and understanding the behavior of the system. They can record a variety of events, including:

- System errors or failures
- Security alerts, such as failed login attempts
- Information about system operations, such as system startup and shutdown
- Changes to system configuration
- Application or software errors

In Unix-like operating systems, the system log often resides in the `/var/log` directory. The `syslog` protocol is a standard for message logging, and tools like `rsyslog` or `syslog-ng` are used to implement the protocol on Linux systems.

In addition to local system logging, logs can also be sent to a centralized logging server, which can be useful for managing logs across multiple systems or for large, distributed applications.

## 2 Syslog Facilities?

In the context of Unix-like operating systems, Syslog facilities are used to categorize the source of the messages being logged. They help in distinguishing between messages from different sources and services, making it easier to filter and manage log messages.

Each Syslog message includes a facility code, which indicates the software type generating the message. Here are some of the standard Syslog facilities:

- **kern**: Messages from the kernel.
- **user**: User-level messages.
- **mail**: Mail system.
- **daemon**: System daemons without separate facility values.
- **auth**: Security/authorization messages.
- **syslog**: Messages generated internally by syslogd.
- **lpr**: Line printer subsystem.
- **news**: Network news subsystem.
- **uucp**: UUCP subsystem.
- **cron**: Clock daemon.

- **authpriv**: Security/authorization messages (private).
- **ftp**: FTP daemon.
- **ntp**: NTP subsystem.
- **security**: Log audit.
- **console**: Log alert.
- **solaris-cron**: Scheduling daemon (Solaris).

In addition to these, there are local use facilities (from local0 to local7) that are not reserved and can be used for custom purposes.

The facility level is often set in the configuration file of the Syslog daemon (like `/etc/syslog.conf` or `/etc/rsyslog.conf`), where you can specify different log files for different facilities.

## 3 Syslog Priorities?

Syslog priorities, also known as severity levels, are used to classify log messages based on the seriousness of the events they represent. Each Syslog message includes a priority value at the beginning of the text. The priority value is calculated by first multiplying the facility number by 8 and then adding the numerical code of the severity level.

Here are the standard Syslog severity levels, from most severe to least severe:

1. **emerg (0)**: Emergency situation where the system is unusable.
2. **alert (1)**: A condition that should be corrected immediately, such as a corrupted system database.
3. **crit (2)**: Critical conditions, such as hard device errors.
4. **err (3)**: Other error conditions.
5. **warning (4)**: Warning messages that indicate a possible problem.
6. **notice (5)**: Normal but significant condition that is worth noting.
7. **info (6)**: Informational messages that are not errors.
8. **debug (7)**: Debug-level messages used for troubleshooting and development.

These severity levels help system administrators and developers to filter and prioritize log messages. For example, in a production system, you might want to be alerted immediately for “emerg” or “alert” messages, while “info” or “debug” messages might be only of interest when diagnosing specific problems.

## 4 The /etc/syslog.conf file?

The `/etc/syslog.conf` file is the main configuration file for the syslogd daemon, which handles the system logging on Unix and Unix-like operating systems. This file defines the rules for logging system messages, including where to save log files and which messages to save.

The file consists of lines with two fields separated by one or more spaces or tabs. The first field specifies a selector, which is a combination of a facility and a priority, separated by a period. The second field specifies an action, which is usually a file where the messages should be logged.

Here’s an example of a line in `/etc/syslog.conf`:

[ ]:
```
mail.info    /var/log/maillog
```

In this example, `mail` is the facility and `info` is the priority. This line tells syslogd to log all mail messages with a priority of `info` or higher to the file `/var/log/maillog`.

The facility can be any of the standard syslog facilities (like `auth`, `cron`, `daemon`, `kern`, `mail`, `user`, etc.), and the priority can be any of the standard syslog priorities (like `emerg`, `alert`, `crit`, `err`, `warning`, `notice`, `info`, `debug`).

You can also specify multiple facilities for a single priority, like this:

[ ]:
```
kern,mail.crit    /var/log/critical
```

This line logs all `kern` and `mail` messages with a priority of `crit` or higher to the file `/var/log/critical`.

Note: In many modern Linux distributions, `rsyslog` has replaced the traditional `syslogd` daemon, and the main configuration file is `/etc/rsyslog.conf` instead of `/etc/syslog.conf`. The syntax is similar but `rsyslog` supports additional features.

## 5 The logger Command?

The `logger` command is a shell command that allows you to add messages to the system log (syslog) on Unix and Unix-like systems. It provides a method for adding log entries from the command line, from scripts, or from other files.

Here are some of the arguments you can use with the `logger` command:

- `-p` or `--priority`: Defines the priority of the message. The priority is specified as a combination of facility and level, separated by a period (for example, `mail.info`).
- `-t` or `--tag`: Adds a tag to the message. This is usually used to identify the source of the message.
- `-s` or `--stderr`: Outputs the message to standard error (stderr) as well as the system log.
- `-i` or `--id`: Logs the process ID of the logger process with each line.
- `-f` or `--file`: Logs the contents of the specified file.
- `-n` or `--server`: Specifies the name of the remote syslog server.
- `-P` or `--port`: Specifies the UDP port of the remote syslog server.
- `-u` or `--socket`: Specifies the socket to use.
- `-d` or `--udp`: Uses UDP only.
- `-T` or `--tcp`: Uses TCP only.

Here's an example of how to use the `logger` command:

[ ]:
```
logger -p mail.info "This is a test log message"
```

This command adds a log message with the text "This is a test log message" to the mail facility with an info priority.

## 6 Log Rotation?

Log rotation is a process that helps manage log files generated by a system. Log files can grow indefinitely if not properly managed, which can consume all the disk space on a system. To prevent this, log rotation is used to periodically archive old log files and create new ones.

Here's how log rotation typically works:

1. A new log file is started.
2. After a certain period of time or when the log file reaches a certain size, the current log file is renamed or moved to a new location. This is often done by appending a number or a date to the filename.
3. If necessary, the old log file is compressed to save disk space.
4. A new log file is started.
5. The process repeats.

In addition to this, log rotation often involves deleting very old log files to prevent them from consuming too much disk space. The number of old log files to keep can usually be configured.

On Unix-like systems, the `logrotate` command is often used to manage log rotation. It's a flexible tool that can be configured to handle the rotation of almost any log file. The configuration file for `logrotate` is typically located at `/etc/logrotate.conf`, and additional configuration files can be placed in the `/etc/logrotate.d/` directory.

A typical `logrotate` configuration might look like this:

[ ]:
```
/var/log/messages {
    rotate 5
    weekly
    compress
    missingok
    notifempty
}
```

This configuration tells `logrotate` to rotate the `/var/log/messages` file weekly, keep the last 5 rotated logs, compress rotated logs, and not rotate the log if it's empty or missing.

### 6.1 How can I configure log rotation for a specific log file using logrotate?

To configure log rotation for a specific log file using `logrotate`, you need to create a configuration file for that log file. Here's a step-by-step guide:

1. Create a new configuration file in the `/etc/logrotate.d/` directory. The name of the file doesn't matter, but it's a good practice to name it after the application or service that's generating the log file. For example, if you're setting up log rotation for a log file generated by an application named `myapp`, you might create a file named `/etc/logrotate.d/myapp`.

2. Open the new file in a text editor. Add a block of configuration options for the log file you want to rotate. Here's an example:

[ ]:
```
/var/log/myapp/myapp.log {
    daily
    rotate 7
    compress
    delaycompress
    missingok
    notifempty
    create 0640 root adm
    postrotate
        /etc/init.d/myapp reload > /dev/null
    endscript
}
```

In this example:

- `/var/log/myapp/myapp.log` is the log file to be rotated.
- `daily` means the log file will be rotated every day.
- `rotate 7` means the system will keep 7 days of logs.
- `compress` means the rotated logs will be gzipped to save space.
- `delaycompress` means the compression will be postponed to the next rotation cycle. This can be useful if the application can't handle the log file being rotated and compressed at the same time.
- `missingok` means it's not an error if the log file doesn't exist.
- `notifempty` means the log file won't be rotated if it's empty.
- `create 0640 root adm` means a new log file will be created with the specified permissions (0640), owner (root), and group (adm) after the old one is rotated.
- The `postrotate`/`endscript` block contains commands that will be executed after the log file is rotated. In this case, it's reloading the `myapp` service.

3. Save the file and exit the text editor.

4. Test the configuration with the `logrotate` command:

[ ]:
```
logrotate -d /etc/logrotate.d/myapp
```

The `-d` option tells `logrotate` to run in debug mode, which means it will check the configuration file for errors but won't actually rotate the logs.

5. If there are no errors, `logrotate` will start rotating the log file according to the schedule specified in the system's cron jobs (usually daily). You can also manually trigger the rotation with the `logrotate` command:

[ ]:
```
logrotate /etc/logrotate.d/myapp
```

Remember to replace `/var/log/myapp/myapp.log` and `/etc/init.d/myapp reload` with the actual path to your log file and the command to reload your application or service.

## 7 Thank You!