

# SelfLinux-0.13.0



## Syslog



Autor: Steffen Dettmer ([steffen@dett.de](mailto:steffen@dett.de))  
Formatierung: Matthias Hagedorn ([matthias.hagedorn@selflinux.org](mailto:matthias.hagedorn@selflinux.org))  
Lizenz: GFDL

Syslog ist ein Dienst, der zentral Meldungen von anderen Diensten und Daemonen des Systems einsammelt, und beispielsweise in Logdateien schreibt, zum Beispiel nach `/var/log/messages` oder über das Netzwerk auf einen zentralen Logserver schickt.

# Inhaltsverzeichnis

## 1 Einleitung

- 1.1 Hintergrundaktivitäten
- 1.2 Logdateien
- 1.3 Syslog - der System Logger

## 2 Meldungen

- 2.1 Quellen von Meldungen
- 2.2 Priorität von Meldungen
- 2.3 Weitere Eigenschaften von Meldungen
- 2.4 Festes Format der Meldungen
- 2.5 Meldungen des Kernels

## 3 Konfiguration von Syslog

- 3.1 Die Konfigurationsdatei
  - 3.1.1 Meldungsbeschreibung
  - 3.1.2 Meldungsaktion
  - 3.1.3 Beispielkonfigurationsdatei
- 3.2 Kommandozeilenoptionen
- 3.3 Remote-Logging
  - 3.3.1 Vorteile des Remote-Logging
  - 3.3.2 Nachteile des Remote-Logging
  - 3.3.3 Konfiguration des Loghosts
  - 3.3.4 Konfiguration der anderen Hosts
  - 3.3.5 Beispiel für Logeinträge
- 3.4 Konfigurationsvorschläge
  - 3.4.1 Serverkonfiguration
  - 3.4.2 Bootkonfiguration
  - 3.4.3 Syslog-Konfiguration
  - 3.4.4 Einheitliche Netzwerkzeit
  - 3.4.5 Firewall-Konfiguration

## 4 Starten und Stoppen von Syslog

## 5 Meldungen selbst erzeugen

## 6 Logdateien auswerten

## 7 Andere Systemlogger

## 8 Weitere Informationsquellen

# 1 Einleitung

Auf einem System laufen, selbst wenn man momentan gerade nicht direkt damit arbeitet, stets viele Dienste. Bei typischen Linuxinstallationen sind es schnell einhundert Prozesse, die im Hintergrund ihre Arbeit verrichten. Auf einem Server ist "ständig was los", auch wenn man diesem das nicht unbedingt ansieht.

## 1.1 Hintergrundaktivitäten

Etliche dieser Hintergrundprozesse sind Dienste, Server oder Daemons. Alle drei Begriffe meinen im Wesentlichen das Gleiche.

Beispiele für solche Dienste sind FTP-, Web- und Mailserver. Da ein Dienst über keine direkte Benutzerschnittstelle verfügt (das heißt, er hat kein "Fenster"), kann er Meldungen und Fehler nicht direkt an den Benutzer melden. Oft sind die Benutzer, die gerade mit einem System arbeiten, auch die nicht gewünschten Empfänger, denn viele Meldungen sind eher für Systemadministratoren gedacht. Auf kleinen Systemen jedoch sind die Benutzer oft gleichzeitig Administratoren.

Um einen ordnungsgemäßen Betrieb zu gewährleisten, muß das System überwacht, und gegebenenfalls auf Fehler reagiert werden.

Auch der Linux-Kernel selbst erzeugt Meldungen, beispielsweise über Hardwarefehler (wie kaputte Festplatten).

## 1.2 Logdateien

Um derartige Meldungen zu verarbeiten, können diese beispielsweise in Logdateien geschrieben werden. Diese können dann von Zeit zu Zeit von einem Administrator geprüft werden, oder bei Fehleranalysen verwendet werden. Der Apache-Webserver schreibt beispielsweise solche Logdateien.

Dieser Weg ist aber für Meldungen von vielen kleineren Diensten umständlich, da man viele Logdateien in unterschiedlichen Formaten analysieren müßte: Zwanzig verschiedene Dienste würden zwanzig verschiedene Logdateien schreiben.

Hier gibt es jedoch einen Dienst, der dafür zuständig ist, von beliebigen Diensten Meldungen einzusammeln, und zentral in Logdateien zu schreiben. Der Dienst, der sehr häufig hierfür eingesetzt wird, ist Syslog.

## 1.3 Syslog - der System Logger

Syslog bietet anderen Diensten eine Schnittstelle, über die Meldungen übergeben werden. Syslog verarbeitet diese Meldungen dann weiter, in dem sie in Dateien geschrieben werden.

Dieser Dienst ist einfach zu handhaben und wird deshalb insbesondere von kleineren Diensten gerne verwendet. Diese Dienste müssen sich dann nicht alle einzeln um Logdateien kümmern. Für den Administrator hat dies Vorteile: Es gibt eine zentrale Konfigurationsdatei und zentrale Logdateien.

Zusätzlich erlaubt es Syslog auch, Logmeldungen an andere Server über das Netzwerk weiterzureichen. Auf dem Zielsystem nimmt wiederum Syslog diese Nachrichten ab, und schreibt sie in Logdateien. Damit lassen sich die Nachrichten von mehreren Systemen auf einem Server zusammenfassen.

## 2 Meldungen

Wenn ein Dienst den Syslog-Dienst verwendet, schickt er seine Meldungen also einfach an Syslog. Eine Meldung ist im Wesentlichen eine Textzeile. Zusätzlich enthält diese noch einige Statusinformationen, beispielsweise wie wichtig diese Meldung ist und zu welchem "Themengebiet" sie gehört bzw. von welcher Quelle sie kommt.

Syslog prüft anhand dieser Werte, ob und wie diese Meldung verarbeitet werden soll. Man kann Syslog zum Beispiel so konfigurieren, dass wichtige Meldungen in der einen und unwichtige in einer anderen Datei stehen, oder dass alle Meldungen, die vom Mailsystem kommen, auf einen anderen Rechner übertragen werden sollen und weiteres.

### 2.1 Quellen von Meldungen

Syslog definiert eine Reihe von Quellen oder Themengebieten für Meldungen. Diese werden als **facilities** bezeichnet. Dabei ist es nur eine Konvention, wann welche Facility verwendet wird. Manchmal ist eine eindeutige Zuordnung nicht einfach möglich. Hier muß man nachsehen, welche Facility ein Dienst benutzt (bei manchen Diensten läßt sie diese auch einstellen).

Die nachfolgende Übersicht beschreibt die Facilities kurz:

Name	Bedeutung
<code>auth</code> , <code>authpriv</code> <code>cron</code>	Meldungen, die zur Authentifizierung gehören, beispielsweise falsche Passwörter. Meldungen, die von Cron erzeugt wurden, oder von Prozessen, die von Cron gestartet werden (die Standard-Ausgabe und Standard-Fehler-Ausgabe werden jedoch von Cron nicht an Syslog gereicht, sondern per EMail verschickt).
<code>daemon</code> <code>kern</code>	Meldungen von allgemeinen Diensten, wie zum Beispiel einem FTP-Server. Meldungen des Systemkernels. Sollte von keinem Dienst verwendet werden. Hierzu gehören beispielsweise Hardware-bezogene Meldungen.
<code>lpr</code>	Meldungen des Drucksystems (Druckerspooler).
<code>mail</code>	Meldungen des Mailsystems (beispielsweise von sendmail und fetchmail).
<code>mark</code>	Nur für Syslog-interne Zwecke, sollte nie verwendet werden.
<code>news</code>	Meldungen des News-Systems, zum Beispiel eines Newsservers.
<code>syslog</code>	Meldungen von Syslog selbst.
<code>user</code>	Meldungen von Benutzersystemen wie zum Beispiel eigenen Scripten.
<code>uucp</code>	Meldungen von Unix-Unix-Copy (UUCP wird heute kaum noch verwendet).
<code>local0</code> bis <code>local7</code>	Diese sind frei und können nach Belieben verwendet werden. Bei Diensten, bei denen man die zu verwendende Facility einstellen kann, kann man diese verwenden und je nach Bedarf verteilen.

### 2.2 Priorität von Meldungen

Syslog definiert eine Reihe von Namen, die die Wichtigkeit beschreiben. Diese Priorität wird englisch als **priority** oder **severity** bezeichnet. Manchmal spricht man auch vom **log level**. Auch diese Eigenschaft kann man verwenden, um die Meldungen differenziert zu verarbeiten, wie bereits angedeutet.

Die folgende Übersicht nennt die definierten Prioritäten:

Name	Beschreibung
<code>debug</code>	Unwichtige Meldungen, dienen nur zu Debug-Zwecken (Fehlerfindung vor allem bei der Entwicklung).
<code>info</code>	Informative, nicht weiter wichtige Meldungen.
<code>notice</code>	Informative Meldungen, die größere Bedeutung haben als <code>info</code> .
<code>warning</code>	Warnungen, also Meldungen, die nicht-fatale Fehler anzeigen.
<code>err</code>	Fehlermeldungen, die kleine Störungen anzeigen.
<code>crit</code>	Kritische (schwerere) Fehler, die beispielsweise Teilausfälle anzeigen.
<code>alert</code>	Schwere Fehler, die erhebliche Störungen und Ausfälle anzeigen.
<code>emerg</code>	Sehr schwere Fehler, die beispielweise den Totalausfall des Systems anzeigen können und schwere Kernelfehler (Hardwareausfälle).

Als Faustregel gilt hier: Alles, was wichtiger oder gleich `warning` ist, verdient auf jeden Fall Aufmerksamkeit.

## 2.3 Weitere Eigenschaften von Meldungen

Zu diesen beiden essentiellen Eigenschaften kommt die Information über den Zeitpunkt der Meldung hinzu (diese wird von Syslog automatisch beigefügt), die Prozeß-ID des Prozesses, der diese Meldung erzeugte, und ein `Tag`, das meistens den Namen des Programmes enthält, das die Meldung erzeugte (beispielsweise verwendet Sendmail das Tag `sendmail`). Auch der Hostname des Systems wird hinzugefügt, was insbesondere wichtig ist, wenn man von mehreren Systemen über das Netzwerk auf ein zentrales loggt.

## 2.4 Festes Format der Meldungen

Beim Schreiben in Logdateien verwendet Syslog ein festes Format. Eine Meldung ist immer eine Zeile. Zu Beginn steht der Zeitstempel, dann folgt der Hostname des Systems. Das dritte Feld ist das Tag (also meistens der Programmname) und in eckigen Klammern dessen Prozeß-ID. Der Rest der Zeile ist die Textnachricht dieser Meldung. Bei einigen Meldungen weicht das Format geringfügig ab, beispielsweise kann die Prozeß-ID entfallen (wie etwa bei Kernel und syslog Meldungen).

Die Facility und Priority sind aus einer solchen Zeile leider nicht mehr erkennbar.

Ein Beispiel für eine Logmeldung:

/var/log/messages
Mar 10 13:30:30 atlas syslogd 1.3-3: restart.

Diese Meldung wurde am 10. März um 13:30 Uhr auf einem Host namens `atlas` erzeugt, und gibt an, dass Syslog gestartet wurde (diese Meldung erzeugt Syslog selbst beim Start).

## 2.5 Meldungen des Kernels

Der Kernel verschickt seine Meldungen auf eine etwas andere Art. Der Kernel kann sich nicht wie ein normales Programm verhalten, beispielsweise weil er als erstes läuft, kein **normaler** Prozeß ist, und weil er aus Performancegründen nicht auf die Fertigstellung von Schreiboperationen warten kann.

Der Kernel legt alle Meldungen in einem speziellen Speicherbereich ab. Diese kann man zunächst überhaupt nicht lesen. Es gibt nun einen speziellen Dienst, den Kernel-Logger `klogd`. Dieser Dienst holt die Meldungen aus dem Speicherbereich ab und wandelt sie auch in menschenlesbare Meldungen um. Der Kernel-Logger kann

die Kernmeldungen in eine Datei schreiben, oder an Syslog weiterversenden. Die zweite Möglichkeit ist die Voreinstellung. Startet man den Kernel-Logger, holt er die Kernmeldungen sofort ab, und verschickt diese an Syslog. Syslog schreibt sie dann in eine Datei. Normalerweise wird `klogd` automatisch direkt nach `syslogd` gestartet.

## 3 Konfiguration von Syslog

Über eine zentrale Datei wird das Verhalten von Syslog gesteuert. Zusätzlich akzeptiert Syslog einige Kommandozeilen-Parameter, die das Verhalten beeinflussen.

### 3.1 Die Konfigurationsdatei

Fast immer heißt diese Datei `/etc/syslog.conf`. Hier wird eingestellt, wie Meldungen in Dateien zu schreiben sind, bzw. wie diese über das Netzwerk übertragen werden.

Diese Datei ist zeilenorientiert. Zeilen, die mit einem Hashmark ("`#`") beginnen, sind Kommentare und werden ignoriert.

Zeilen bestehen aus zwei Teilen, die durch mindestens einen Tabstop getrennt sind (moderne GNU/Linux Syslog Implementierungen erlauben oft auch Leerzeichen, es sollten aber sicherheitshalber Tabstops verwendet werden). Zu Beginn der Zeile, also auf der linken Seite, steht eine Beschreibung der Nachricht. Hier können über die Facility und Priority Meldungen **ausgewählt** werden.

Der zweite Teil gibt dann an, was mit diesen ausgewählten Meldungen passieren soll. Hier steht meistens ein Dateiname. In diese Datei werden dann die Meldungen geschrieben. Es sind nicht nur Dateinamen erlaubt: NetzwerkAdressen (IP-Adressen oder Hostnamen) sind hier erlaubt und auch Benutzernamen können hier stehen. Im letzten Fall werden die Meldungen auf die Konsolen der entsprechenden Benutzer geschrieben. Das kann für sehr wichtige Meldungen Sinn machen, wird aber im Allgemeinen als störend empfunden. Oft verwendet man als einzigen Benutzernamen `root`, damit ein eventuell angemeldeter Administrator wichtige Meldungen sofort auf sein Terminal bekommt (insbesondere bei Störungen und der Suche nach den Ursachen für diese ist das oft hilfreich).

Es werden immer alle Aktionen ausgeführt, deren Beschreibung auf die Meldung paßt. Dadurch kann ein- und dieselbe Meldung in mehrere Dateien geschrieben und gleichzeitig beispielsweise über das Netzwerk verschickt werden.

#### 3.1.1 Meldungsbeschreibung

Die Meldungsbeschreibung ist eine Liste aus Facility/Priority-Paaren. Das wird so verwendet: Ist eine Meldung der entsprechenden Facility mindestens so wichtig, wie die Priority angibt, wird die rechte Seite verwendet (die Aktion wird ausgeführt). Hat man viele Regeln, so werden wichtige Meldungen damit oft in mehrere Dateien geloggt; dies ist erwünscht.

Der Grundaufbau der durch Semikolon (`;`) getrennten Facility/Priority-Paaren ist einfach: Facility und Priority stehen in dieser Reihenfolge durch einen Punkt getrennt. Wichtige Kernmeldungen lassen sich beispielsweise durch:

```
kern.warning
```

beschreiben. Hier sind nicht nur die Meldungen der Priorität **warning**, sondern auch alle höheren (also **err**, **crit**, **alert** und **emerg**) gemeint. Man kann auch Wildcards verwenden, so zum Beispiel **\*.warning** für alle wichtigen Meldungen und `kern.*` für alle Kernmeldungen. Hierbei ist jedoch zu beachten, daß nicht alle Syslogdienste Wildcard verstehen (aber die unter GNU/Linux verwendeten können dies). Bei GNU/Linux-Syslog kann man auch mehrere Facilities durch Komma (`,`) getrennt auflisten. Dies ist jedoch nur zulässig, wenn diese alle die gleiche Priorität haben (die nur an der letzten Facility angehängt steht). Nach dieser komplizierten Erklärung ein einfaches Beispiel: Wichtige Meldungen von News oder Mail:

```
news,mailwarning
```

Oft wird jedoch auch hier lieber eine durch Semikolon getrennte Liste verwendet, weil es als lesbarer und weniger verwirrend empfunden wird:

```
news.warning;mail.warning
```

Verwendet man Wildcards, kann man sämtliche Meldungen mit `*.*` erfassen. GNU/Linux-Syslog bietet neben Wildcards weitere nützliche Erweiterungen: Möchte man nicht, dass alle Meldungen auch höherer Priorität passen, kann man vor die Priority ein Gleichheitszeichen (=) setzen, beispielsweise `*.=warning`. In solchen Fällen muß man aber unbedingt Regeln haben, die auch die wichtigeren Meldungen verarbeiten, sonst fehlen am Ende die wichtigsten Meldungen!

Es ist auch möglich, mit einem Ausrufezeichen (!) eine Priorität auszuschließen, zum Beispiel `*.!=warning`, was man aber sehr selten sieht.

Widersprechen sich Bedingungen einer durch Semikolon getrennten Bedingungsliste, so gilt die zuletzt beschriebene, also

```
kern.!=info;kern.*
```

loggt jegliche Kernmeldung (hier meinte man vermutlich einfach `kern.!=info`). Solche Konstrukte sind natürlich zu vermeiden.

Es gibt eine spezielle Priority `none`, die für keine Priority der dazugehörigen Facility steht:

```
*.info;mail.none
```

bezeichnet alle Meldungen mit Priority `info`, außer von der Facility `mail`.

### 3.1.2 Meldungsaktion

Auf der rechten Seite steht dann, was mit Meldungen geschehen soll. Im einfachsten Fall steht hier ein Dateiname. Diese müssen mit vollem Pfad angegeben werden, sie beginnen also mit einem Slash (/). Beispiel: `/var/log/messages`. Möchte man nicht-synchronisiert schreiben (später mehr dazu), schreibt man noch ein Minus (-) davor: `-/var/log/messages`. Als spezielle Datei kann man auch ein Terminal, zum Beispiel `/dev/tty10`, verwenden. Dann erscheinen die Meldungen auf der Konsole 10 (die man meist über `ATL + F10` oder `STRG + ALT + F10` erreicht).

Neben Dateien kann man auch sogenannte `named FIFOs` verwenden. Diese beginnen mit einem Pipezeichen (`|`), gefolgt vom Dateinamen. Für diese Spezialität erfolgt an dieser Stelle jedoch keine detaillierte Diskussion.

Soll die Meldung an einen anderen Server übertragen werden, schreibt man ein At-Zeichen (@) und den Hostnamen oder besser eine IP-Adresse des Systems, zum Beispiel `@192.168.1.14`.

Um die Meldung auf die Terminals von Benutzern zu schreiben, schreibt man einfach dessen Accountnamen, beispielsweise `root`. Hier darf auch ein `*` stehen. Dann werden alle Benutzer (also alle Terminals) informiert. Das kann jedoch stören, denn die Meldungen erscheinen dann "mitten im Terminaltext" und "verunstalten" das Layout der laufenden Anwendung (wenn man zum Beispiel gerade einen Editor offen hat. Etliche Anwendungen haben eine Möglichkeit, die Anzeige neu zu zeichnen, häufig `STRG + L`).

### 3.1.3 Beispielkonfigurationsdatei

Es folgt ein Beispiel mit ausführlichen Kommentaren.

```
/etc/syslog.conf

#/etc/syslog.conf: Syslogkonfigurationsdatei
#Zur Trennung der "linken" und "rechten" Seite sollten
# Tabstops verwendet werden (moderne GNU/Linux Syslog
# Implementierungen kommen meist auch mit Leerzeichen klar)

#sehr wichtige Warnungen vom Kernel, und alle Fehler außer
# evtl. Passwortvertipper auf die Konsole ALT-F10 loggen.
# Zur Erinnerung: .warn schließt höhere Meldungen (also err,
# crit, alert, emerg) mit ein. Diese landen also auch auf
# ALT-F10.
kern.warning;*.err;authpriv.none          /dev/tty10

#Die gleichen Meldungen für die xconsole bereitstellen
# (Hier wird ein FIFO verwendet, der von xconsole
# ausgelesen wird)
kern.warn;*.err;authpriv.none            |/dev/xconsole

#ALLE Meldungen auf ALT-F9. Das ist für einen schnellen Überblick
# bei "Echtzeit-Fehleranalyse" hilfreich
*.*                                       /dev/tty9

#alle sehr schweren Fehler direkt an alle Benutzer in die Konsolen
# schreiben. In solchen Fällen ist das System vermutlich eh kaum
# noch benutzbar. Eventuell sieht man aber kurz vor dem Absturz
# noch eine Fehlermeldung und kann nach einem Neustart etwas
# ändern
*.emerg                                   *

#Root möchte eventuell auch schon "crit" auf der Console sehen,
# wenn er zufällig gerade an dem System arbeitet:
#*.crit                                   root

#Alle EMail-Meldungen in eine eigene Datei. Diese Datei wird
# aus Performance-Gründen nicht nach jeder Zeile synchronisiert,
# (aber schwere Fehler schreiben wir nochmal in eine extra Datei)
mail.*                                    -/var/log/mail

#Warnungen in eine extra Datei. Diese wird hier nicht sync'd (bei
# langsameren Systemen hilfreich)
*.=warn;*.=err                            -/var/log/warn

#"crit" und höhere kommen in die gleiche Datei, aber sync'd
*.crit                                     /var/log/warn

#alles außer "debug" und mail in eine andere Datei
*.info;mail.none                           -/var/log/messages

#Für die Fehlersuche hilft oft eine Datei, die sämtliche
#Informationen enthält
#*.*                                       -/var/log/allmessages

#Hat man einen Loghost, soll dieser eine Kopie von allen
# Meldungen erhalten
#*.*                                       @192.168.1.1

#Weniger wichtige Systeme sollen das Netzwerk nicht unnötig
# belasten
#*.warn                                    @192.168.1.1
```

### 3.2 Kommandozeilenoptionen

Kommandozeilenoptionen steuern das Verhalten von Syslog ebenfalls. Über Kommandozeilenoptionen kann

Syslog konfiguriert werden, dass Meldungen vom Netzwerk (also anderen System-Loggern) akzeptiert und verarbeitet werden. Man kann Syslog auch veranlassen, eine andere Konfigurationsdatei zu verwenden.

Option	Beschreibung
<code>-a</code> <Socket>	Öffnet <Socket> zum Lesen von Meldungen. <Socket> ist auf <code>/dev/log</code> voreingestellt. Hier kann zum Beispiel zusätzlich ein <code>dev/log</code> aus einer "chroot" Umgebung angegeben werden, damit die "chroot" Umgebung auch Syslog verwenden kann.
<code>-d</code>	Debug Modus (für Entwickler gedacht)
<code>-f</code> <Konfigdatei>	Lädt eine andere Konfigdatei. Normalerweise wird <code>/etc/syslog.conf</code> verwendet.
<code>-h</code>	Über das Netzwerk empfangene Meldungen auch über das Netzwerk weitersenden. Damit kann man mehrere Netzwerk-Syslogs "in Reihe schalten", um beispielsweise Meldungen durch mehrere Firewalls oder aus einer DMZ zu bekommen. <code>-t</code> sollte auch verwendet werden, siehe dort.
<code>-l</code> <Hostnamen>	Eine durch <code>:</code> getrennte Liste von Hostnamen, die in kurzer Form in der Logdatei stehen. Gewöhnlich bevorzugt man die Option <code>-s</code> , die ähnliches Verhalten bringt.
<code>-m</code> <Mark Zeit>	Syslog schreibt alle 20 Minuten einen Eintrag <code>--MARK--</code> in ein Logfile. Daran kann man erkennen, dass das System noch lebt. Bei der nachträglichen Analyse kann man dadurch beispielsweise nächtliche Abstürze zeitlich eingrenzen. Durch diese Option kann man anstatt 20 (Minuten) auch einen anderen Wert verwenden. Der Wert <code>0</code> schaltet die Funktion ab.
<code>-n</code>	Syslog soll nicht automatisch in den Hintergrund gehen. Diese Option wird im Normalfall nicht verwendet. Auf speziellen Systemen (Rettungs- oder Installationsystemen) wird diese manchmal gesetzt.
<code>-p</code> <Socket>	Öffnet <Socket> zum Lesen von Meldungen. Siehe Option <code>-a</code> .
<code>-r</code>	Aktiviert den Empfang von Netzwerkmeldungen. Aus Effizienz- und Stabilitätsgründen sollte man alle IPs, von denen man Meldungen empfängt, in die Datei <code>/etc/hosts</code> eintragen (diese werden benutzt, um den Hostnamen für das Logfile zu bilden)
<code>-s</code> <Domains>	<Domains> ist eine durch <code>:</code> getrennte Liste von Domains, die vor dem Loggen von Hostnamen abgeschnitten werden. Das ist in Verbindung mit <code>-r</code> hilfreich, da die FQDNs (vollen Namen) viel Platz im Logfile wegnehmen, und die Hostnamen meistens sowieso schon eindeutig sind. Hat man einen host <code>mail.selflinux.de</code> und ein <code>-s selflinux.de</code> , so wird der Hostname also als <code>mail</code> in den Logdateien stehen.
<code>-t</code>	Weitergeleitete Meldungen (siehe Option <code>-h</code> ) sollen den empfangenen Hostnamen enthalten, nicht den eigenen. Das heißt also, der Hostname der Meldung wird nicht verändert; diese können damit weiterhin eindeutig zugeordnet werden.

Diese Optionen werden in der Regel vom Syslog-Startscript, häufig `/etc/init.d/syslog`, beim Start von Syslog an diesen übergeben. Hier kann man also die eigenen Optionen für den Aufruf angeben.

Bei SuSE-Systemen ist das Startscript intelligenter, es gestattet dem Administrator, auf einfachem Weg weitere Startoptionen zu setzen. Hierzu öffnet man dazu einfach die Datei `/etc/rc.config` und ändert

<code>/etc/rc.config</code>
<code>SYSLOGD_PARAMS=" "</code>

so, dass die erwünschten Startoptionen verwendet werden. Diese werden hier einfach eingetragen.

Auch unter *RedHat* muß man nicht mehr die Datei `/etc/init.d/syslog` bearbeiten, unter `/etc/sysconfig/syslog` kann man durch Ändern von `SYSLOGD_OPTIONS=""` (z.B. `"-r -m 0 -s`

`picard.inka.de:zeibig.net")` die gewünschten Startoptionen angeben.

### 3.3 Remote-Logging

Remote-Logging bedeutet, dass ein Host Syslog-Meldungen auf einen anderen Host weiterversendet. Dieser andere Host schreibt die Meldungen dann in Dateien. Gewöhnlich konfiguriert man das so, daß die Meldungen nicht nur über das Netzwerk verschickt, sondern daneben auch lokal in Dateien geschrieben werden. Dies beugt Informationsverlust bei Netzerkausfällen oder Störungen vor. Da Syslog eine wichtige Informationsquelle zur Analyse von Störungen ist, soll hier natürlich möglichst nichts fehlen.

#### 3.3.1 Vorteile des Remote-Logging

Oft hat man in LANs einen zentralen Host, der Netzwerk-Syslog-Meldungen erhält, und diese in Dateien schreibt. Diesen Host nennt man **Loghost**.

Diese Konfiguration hat etliche Vorteile: So kommen die Meldungen zentral auf einer Maschine an, so dass man auch komplexere Störungen analysieren kann (wenn diese mehrere Server betreffen, zum Beispiel einen Mailserverausfall, weil DNS ausgefallen ist).

Ein weiterer Vorteil liegt im Fall von erfolgreichen Angriffen vor. Wenn ein Angreifer ein System kompromittiert hat, wird er in den meisten Fällen die Syslogdateien löschen oder manipulieren, um sich zu tarnen und seine Herkunft zu verschleiern. Wenn nun aber der Administrator einen Loghost verwendet, ist es unwahrscheinlicher, dass auch dieser gleichzeitig gehackt wird. So kann er auf dem Loghost die Meldungen analysieren und wichtige Informationen über den Angriff erlangen.

Ein dritter Vorteil der Zentralisierung ist die Vereinfachung automatischer Behandlung von Logfiles, zum Beispiel wird das Aufbereiten/Filtern und als Mail Verschicken erleichtert: Man muß diesen Vorgang nur auf einer Maschine pflegen.

#### 3.3.2 Nachteile des Remote-Logging

Remote-Logging hat aber auch Nachteile, gerade, wenn man Syslog einsetzt. Syslog verwendet ausschließlich das UDP Protokoll. UDP Pakete werden direkt verschickt, ihr Empfang wird nicht bestätigt. Auf stark ausgelasteten Netzen kann es daher vorkommen, dass Meldungen verloren gehen, ohne dass man es bemerkt. Weiterhin kann ein Angreifer den Loghost **überfluten**, in dem er sehr viele sinnlose Meldungen verschickt. Dies kann die Last auf dem Loghost stark erhöhen, und im Extremfall dazu führen, dass er nur noch einen Teil der wichtigen Meldungen erhält, und die Netzwerklast kann zu weiteren Störungen führen. Bei sehr massiven Flut-Angriffen ist auch ein Vollaufen der Festplatte denkbar. In diesem Fall ist neben dem Verlust von Logmeldungen in der Regel mit weiteren empfindlichen Störungen zu rechnen, oft mit einem Totalausfall sämtlicher Dienste des Loghosts!


Ein Angreifer, der eine Maschine gehackt hat, und hier über root-Rechte verfügt, kann auch einen Netzwerk-Sniffer verwenden, um die Syslog-Nachrichten, die über das Netzwerk verschickt werden, mitzulesen. Er kann dadurch wichtige Informationen ausspionieren. Syslog gestattet leider keine Verschlüsselung oder andere Absicherung der Netzwerkkommunikation.

#### 3.3.3 Konfiguration des Loghosts

Die Konfiguration des Loghosts ist einfach. Man muß lediglich den Empfang aktivieren, in dem man Syslog mit der Option `-r` startet. Bei SuSE-Systemen öffnet man dazu einfach die Datei `/etc/rc.config`, und ändert

```
                                /etc/rc.config
SYSLOGD_PARAMS=" "
```

so, dass die Startoption `-r` verwendet wird. Die IP Adressen der Hosts, die den Loghost verwenden, sollte man in die Datei `/etc/hosts` eintragen, um Fehlern bei DNS Ausfällen vorzubeugen.

Kommt nun eine Nachricht über das Netzwerk, schaut Syslog anhand der Sender-IP Adresse nach, wie der Hostname des Systems lautet. Ist dieser beispielsweise  [www.selflinux.de](http://www.selflinux.de), so wird dieser Name im Logfile eingetragen. Dies ist unübersichtlich, und man möchte vermutlich die Ausgaben von `".selflinux.de` unterdrücken (sofern der Teil davor eindeutig ist). Dazu verwendet man am einfachsten die Option `-s`, die die Domainanteile abschneidet. In unserem Beispiel würde der Administrator also `-r -s selflinux.de` verwenden. Hat er ein SuSE-System, trägt er einfach in `/etc/rc.config` ein:

```
                                /etc/rc.config
SYSLOGD_PARAMS="-r -s selflinux.de"
```

Syslog verwendet den UDP Port `syslog`. Dieser wird in der Datei `/etc/services` nachgesehen. Normalerweise soll Syslog die `Portnummer 514` verwenden. Demzufolge muß folgende Zeile in der Datei `/etc/services` vorhanden sein:

```
                                /etc/services
syslog          514/udp
```

Dies ist bei gängigen Distributionen (SuSE, RedHat) jedoch bereits richtig eingetragen.

Nun muß Syslog neu gestartet werden, damit die Änderungen aktiv werden. Dazu schreibt man beispielsweise:

```
root@linux ~/ # /etc/rc.d/syslog restart
```

Auf SuSE Systemen kann man auch schreiben:

```
root@linux ~/ # rcsyslog restart
```

Nun akzeptiert der Syslog Nachrichten vom Netzwerk.

### 3.3.4 Konfiguration der anderen Hosts

Die Maschinen, die nun den Loghost verwenden sollen, müssen hierzu angepaßt werden. Ein neuer Eintrag in der Datei `/etc/syslog.conf` ist auf jedem Server notwendig. Möchte man alle Nachrichten auf den Loghost `192.168.1.1` loggen, verwendet man:

```
                                /etc/syslog.conf
*.*                @192.168.1.1
```

Um nur wichtige Meldungen zu verschicken, kann man

/etc/syslog.conf	
*.warn	@192.168.1.1

verwenden. Es kann auch mehrere solcher Zeilen geben, so kann man sich auch eine Konfiguration mit zwei Loghosts vorstellen. Über den Sinn solchen Vorgehens kann man natürlich streiten.

Nach dem Ändern dieser Datei muß Syslog neu geladen oder neu gestartet werden. Dazu kann man Syslog ein Hangup-Signal senden (SIGHUP), in dem man beispielsweise schreibt:

```
root@linux ~/ # killall -HUP syslog
```

oder auf SuSE-Systemen das Startscript verwenden:

```
root@linux ~/ # rcsyslog reload
```

Man kann Syslog aber auch einfach neu starten (*stop/start*). Allerdings gehen hier möglicherweise für einige Sekunden Meldungen verloren.

### 3.3.5 Beispiel für Logeinträge

Auf dem Loghost kann man dann gut das Netzwerksystem beobachten. Ein fiktives Beispiel:

/var/log/messages (fiktives Beispiel)	
<pre>Mar 10 13:30:30 atlas syslogd 1.3-3: restart. Apr  1 13:02:01 ns1 named[124]: XX+/127.0.0.1/1.1.168.192.                         in-addr.arpa/PTR/IN Apr  1 13:02:01 www httpd[123]: GET /login.cgi?username=steffen Apr  1 13:02:03 www httpd[123]: Starting authorization for                         "steffen" from "wsl.selflinux.de" Apr  1 13:02:04 radius radiusd[125]: autorization request from                         "www.selflinux.de" for "steffen" Apr  1 13:02:04 db kernel: end_request: I/O error, dev 03:02 (hda),                         sector 58138452 Apr  1 13:02:04 db postmaster[111]: Database error: disk read failed                         (I/O error) Apr  1 13:02:04 radius radiusd[125]: authorization request for                         "steffen" failed (database error) Apr  1 13:02:03 www httpd[123]: Authorization for "steffen" failed                         (incorrect password)</pre>	

In diesem fiktiven Szenario sieht man eine fehlgeschlagene Web-Anmeldung. Der Webserver (httpd) löste die IP Adresse auf (über "named" auf "ns1") und fragte dann bei einem Radius-Dienst auf einem separaten Server nach. Dieser wiederum verwendete eine PostgreSQL Datenbank auf einem anderen Server. Diese hat ein großes Problem: Eine kaputte Festplatte ( **sector 58138452** konnte nicht gelesen werden). Demzufolge kann PostgreSQL (*postmaster*) die Anfrage nicht bestätigen. Radius meldet also einen Fehler, den der Webserver als **incorrect password** fehlinterpretiert. Nicht das falsche Passwort war das Problem, sondern eine

kaputte Festplatte! In diesem Fall würde man im Webserverlog sehr viele `incorrect password` finden, und einen Angriff vermuten. Doch durch die Verwendung eines Loghosts sind die Meldungen aller Komponenten zentral verfügbar. Hier hat das die Fehlersuche erheblich beschleunigt (der Administrator hat die Festplatte sofort gewechselt und die Bandsicherung zurückgespielt).

## 3.4 Konfigurationsvorschläge

### 3.4.1 Serverkonfiguration

Bei der Installation eines Servers kann man überlegen, wie man mit großen Logfiles umgehen möchte. Hier ist zunächst von Interesse, dass große Logdateien Filesysteme füllen können. Hat man die Logdateien (in der Regel also das Verzeichnis `/var/log`) im selben Filesystem gemoutet wie beispielsweise das Root-Filesystem `/`, so ist damit zu rechnen, dass nach einer Logflut das System vollkommen unbenutzbar ist, also mit dem Ausfall sämtlicher Dienste.

Diese Gefahr kann man durch den Einsatz von Werkzeugen wie `logrotate` oder dem SuSE-Linux `/etc/logfiles` Mechanismus senken. Auf SuSE-Systemen trägt man hierzu jede Logdatei in `/etc/logfiles` ein. Hinter den Dateinamen schreibt man die Größe (z.B. `+1024k`) und den Zugriffsmodus (beispielsweise `640`) und den Eigentümer (beispielsweise `root.root`). Die erste Option wird für das Dienstkommando `find` als Parameter verwendet, der zweite für `chmod` und der dritte für `chown`. Die manpages dieser drei Werkzeuge geben Auskunft über Art der verwendbaren Werte. Auf SuSE-Systemen sind in dieser Datei die voreingestellten Logdateien bereits eingetragen. Eigene Dateien muß man hier natürlich hinzufügen.

Eine weitere Möglichkeit wäre der Einsatz von separaten Filesystemen. Man kann z.B. `/var` auf eine andere Partition oder ein anderes LVM LV (logical volume) mounten. Dies hat jedoch auch Nachteile: es wird nicht der gesamte verfügbare Platz für Logfiles verwendet (demzufolge fällt das Logging früher aus), und insbesondere Angriffe und Störungen sind damit nicht mehr rekonstruierbar. Daher ist eine regelmäßige Überprüfung der freien Plattenkapazität durchzuführen, vorzugsweise automatisch, dann kann man es nicht vergessen.

### 3.4.2 Bootkonfiguration

Syslog sollte stets laufen. Syslog benötigt meist Schreibzugriff auf Festplatten. Bei Konfigurationen mit einem zentralen Loghost wird weiterhin das Netzwerk benötigt. Daher sollte man Syslog unmittelbar nach dem Hochfahren des Netzwerkes starten. Auf SuSE-Systemen verhält sich das bereits so.

Es ist denkbar, den Start von Syslog vor dem des Netzwerkes durchzuführen, wenn man keinen Loghost verwendet. Eventuell erhält man so mehr Meldungen.

Nach dem Start von Syslog sollte man den Kernel-Logger `klogd` starten. Sicherheitshalber wartet man z.B. eine Sekunde dazwischen. Die GNU/Linux-Startscripte sollten dies bereits so machen (bei SuSE-Systemen ist es der Fall).

### 3.4.3 Syslog-Konfiguration

Man sollte darauf achten, dass keine Meldungen überhaupt nicht geloggt werden. Meistens möchte man verschiedene Dateien haben, um schnell Meldungen zu finden. Oft werden mindestens die Dateien `/var/log/messages` und `/var/log/warn` verwendet. Letzere enthält nur wichtige Meldungen. Sehr üblich ist auch eine Datei `/var/log/mail` und eventuell `/var/log/news` für Meldungen des Mail- bzw. Newssystems. Häufig sieht man auch `/var/log/allmessages` oder `/var/log/allinone`, die sämtliche Meldungen enthalten.

Zusätzlich empfiehlt es sich, Meldungen auf einer virtuellen Konsole zu haben.

Weitere Informationen und ein Beispiel finden sich im Abschnitt "Die Konfigurationsdatei", siehe oben.

### 3.4.4 Einheitliche Netzwerkzeit

Bei der Analyse von Störungen ist es wichtig, Reihenfolgen und Abstände von Fehlermeldungen oder Fehlermails richtig feststellen zu können. Oft sind Zeitstempel bekannt. Diese sind natürlich Rechner-übergreifend nur verwendbar, wenn auch alle Maschinen die gleiche Vorstellung der Netzwerkzeit haben, deren Uhren also genau gleich bzw. synchron sind. Es empfiehlt sich also (insbesondere, wenn man keinen Loghost verwendet), die Netzwerkzeiten zu synchronisieren. Dazu verwendet man üblicherweise einen NTP (Network Time Protocol) Dienst, beispielsweise xntpd.

### 3.4.5 Firewall-Konfiguration

Firewalls sollten verhindern, dass UDP/514 Pakete vom Internet in das LAN geroutet werden, um Flut-Angriffen zu entgehen. Selbst wenn man keinen Loghost verwendet, könnte möglicherweise ein interner Host den Empfang von Netzwerk-Meldungen aktiviert haben. Auskunft darüber gibt das Kommando:

```
root@linux ~/ # netstat -an --inet | grep 514
```

Sicherheitshalber sollten Firewalls ohnehin alle nicht benötigten und nicht verwendeten Ports sperren.

Interne Firewalls müssen diese Pakete natürlich zwischen Loghost und den Logsystemen erlauben, aber sollten so eingestellt werden, daß nur die betreffenden IP-Adressen erlaubt sind.

Es ist zu beachten, dass die **Absender-Adresse** von UDP Paketen sehr einfach fälschbar ist. Demzufolge kann man bei Firewalls diese Adressen nicht zum Filtern verwenden. Man sollte hier an Interfaces blocken. Hat eine Firewall beispielsweise ein externes Interface `eth1`, sollte eine entsprechende Firewall-Regel den Empfang und Versand von Syslog-Paketen über dieses Interface unterbinden. Wenn die Firewall über `eth0` an das interne Netz angebunden ist, kann hier dennoch ein Loghost verwendet werden, der die Firewallmeldungen empfängt.

Bei der Verwendung von Firewalls und Loghosts ist zu beachten, daß normalerweise unerlaubte Pakete geloggt werden, was zu einem hohen Aufkommen an Meldungen führt. Hier sollte Syslog so konfiguriert werden, dass diese Meldungen nicht über das Netzwerk geschrieben werden, wenn sich hier Probleme ergeben (Portscans könnten beispielsweise zu Flut-Verhalten führen).

Häufig sind aber die Außenanbindungen vergleichsweise langsam (beispielsweise E1 (2MBit) extern und 100Mbit intern), so daß hier interne Netzwerküberlastungen eher unwahrscheinlich sind.

## 4 Starten und Stoppen von Syslog

Der Start erfolgt fast immer und ausschließlich automatisch beim Hochfahren des Systems über ein `rc-Script`, häufig `/etc/rc.d/syslog`. Bei gängigen GNU/Linux Distributionen ist das bereits so konfiguriert. Dieses Script startet oft (zum Beispiel bei SuSE-Systemen) auch automatisch den Kernel-Logger. Andere Systeme haben hier eventuell ein eigenes `rc-Script`. Dieses sollte dann in jedem Fall direkt nach Syslog gestartet werden.

Um Syslog manuell zu starten, verwendet man:

```
root@linux ~/ # /etc/rc.d/syslog start
```

oder bei SuSE-Systemen kurz:

```
root@linux ~/ # rcsyslog start
```

Um den Dienst zu stoppen, schreibt man analog dazu `stop` anstatt `start`.

## 5 Meldungen selbst erzeugen

Es gibt mehrere Fälle, in denen man selbst Meldungen erzeugen möchte. Denkbar sind beispielsweise Cron-Jobs, die neben dem Verschicken der EMail mit den Script-Ausgaben kurze Erfolgs- oder Mißerfolgsmeldung über Syslog schreiben sollen. Eine weitere Anwendung sind automatisch ausgeführte Scripte, beispielsweise `/etc/ppp/ip-up` oder `rc-Scripte`.

Es gibt ein kleines, einfaches Werkzeug, mit dem man Syslog-Meldungen erzeugen kann. Es heißt `logger`. Diesem Werkzeug kann man über Optionen sagen, wie eine Meldung zu loggen ist. Hier kann man beispielsweise Priority und Facility angeben.

Die wichtigsten Optionen sind:

Option	Beschreibung
<code>-i</code>	Prozeß-ID mit in die Nachricht schreiben
<code>-p &lt;Fac.&gt;.&lt;Pri.&gt;</code>	Angegebene Priorität und Facility verwenden. Mögliche Werte sind die gleichen wie die in <code>/etc/syslog.conf</code> verwendbaren, siehe Abschnitt "Quellen von Meldungen" und "Priorität von Meldungen". Wird diese Option nicht angegeben, wird <code>user.notice</code> verwendet.
<code>-t &lt;Tag&gt;</code>	Angegebenes "Tag" verwenden. Meist wird hier der Name des Scriptes verwendet.

Es gibt zwei Arten, `logger` aufzurufen. Einmal kann man die Meldungen mit in die Kommandozeile schreiben, direkt hinter die Optionen. Mindestens wenn die Meldung Minuszeichen (-) enthalten könnte, sollte man sie mit `--` abtrennen, um zu vermeiden, dass Meldungsteile als Optionen mißinterpretiert werden. Ein Beispiel:

```
user@linux ~/ $ logger -i -t MeinProgramm -- Ich bin eine Meldung.
```

Dies erzeugt dann folgenden Logdatei-Eintrag:

```
user@linux ~/ $ logger -i -t MeinProgramm -- Ich bin eine Meldung.  
Apr 13 13:02:04 atlas MeinProgramm[6178]: Ich bin eine Meldung.
```

Die zweite Art des Aufrufes unterscheidet sich in der Art, wie die Meldung übergeben wird. Wird nämlich keine auf der Kommandozeile angeben, liest `logger` die Standard-Eingabe. Damit kann man also schreiben:

```
user@linux ~/ $ echo "Ich bin eine Meldung." | logger -i -t MeinProgramm
```

was zum gleichen Ergebnis führt. `logger` geht auch mit mehreren Eingabezeilen korrekt um (jede Zeile wird eine Meldung). In Scripten kann man deshalb beispielsweise komfortabel schreiben:

Beispielscript
----------------

<pre>LOGGER="/usr/bin/logger -t `basename \$0`[\$\$]" {   test -e /etc/irgendeinedatei    echo "Error, Datei nicht gefunden!";   date;   echo "eine andere Fehlermeldung"; }   \$LOGGER</pre>
---


Dies erzeugt auf elegante Art und Weise folgende Einträge:

/var/log/messages
-------------------

<pre>Apr 13 13:20:45 atlas script.sh[6338]: Error, Datei nicht gefunden! Apr 13 13:20:45 atlas script.sh[6338]: Sat Apr 13 13:20:45 MEST 2002 Apr 13 13:20:45 atlas script.sh[6338]: eine andere Fehlermeldung</pre>
--

## 6 Logdateien auswerten

Sehr wichtig ist es natürlich, die erzeugten Logdateien auszuwerten. Gewöhnlich hat ein Administrator jedoch nicht die Zeit, täglich tausende von Zeilen zu lesen.

Hier benutzt man Werkzeuge, um diese Arbeit zu automatisieren. Ein Beispiel hierfür ist `LogWatch`, das von RedHat verwendet wird. Ein weiteres ist `logmail`, das man sich von  <http://sws.dett.de/logmail> herunterladen kann (es enthält auch eine deutsche Dokumentation). Sicherlich gibt es viele weitere ähnliche und leistungsfähigere Werkzeuge.

Verwendet man `logmail`, so kann man sich eine Konfigurationsdatei anlegen, in dem spezifiziert wird, welche Einträge per EMail an wen verschickt werden. Bei Loghosts kann man zum Beispiel alle Meldungen von `postmaster` an den PostgreSQL Administrator und alle Meldungen von `named` an den DNS-Administrator verschicken, und alle Meldungen an einen dritten Administrator. Des weiteren sollte konfiguriert werden, welche Meldungen man nicht haben möchte, sonst erhalten die Administratoren viele **langweilige** Meldungen. `Logmail` gestattet es auch, Loghost-Meldungen anhand des Hostnamens per EMail zu verteilen. So kann man den entsprechenden Systemadministratoren **ihre** Meldungen zustellen.

Man kann sich alle Meldungen, die man per EMail erhält, aber nicht mehr erhalten möchte, einfach in die Konfigurationsdatei als Filter mit aufnehmen. Dann werden diese in Zukunft nicht mehr verschickt. Derartige Systeme bedürfen natürlich der Pflege. In wenigen Tagen erhält man dann aber nur noch neue, interessante Meldungen per EMail, und kann nicht vergessen, Logfiles auszuwerten. So erkennt man auch Störungen, die nachts auftraten, oder bekommt Mitteilungen über Angriffsversuche. Zusätzlich senkt das die Gefahr, dass ein Angreifer Logfiles manipulieren kann, da unter Umständen bereits eine EMail verschickt wurde, die der Angreifer nicht mehr **aufhalten** oder ungeschehen machen kann.

Nur durch derartiges Vorgehen ist es möglich, mehrere Server richtig zu betreuen, da im Produktionsbetrieb natürlich keine Zeit bleibt, täglich stundenlang Logfiles auszuwerten, die fast immer langweilig sind.

## 7 Andere Systemlogger

Neben Syslog gibt es weitere Systemlogger, die man verwenden kann. Beispiele sind `socklock` und `syslog-ng`. Beide haben interessante Funktionen und Vorteile gegenüber Syslog.

## 8 Weitere Informationsquellen

Es folgt eine unvollständige Liste:

- \* `syslog manpage`

- \* `klogd manpage`

HOWTOs oder andere Dokumentationen sind mir nicht bekannt. Über Hinweise freue ich mich natürlich jederzeit.