# Wstęp

Każdy system komputerowy, niezależnie od systemu operacyjnego posiada mechanizm rejestrujący zdarzenia systemowe. Mechanizmy takie odgrywają kluczową rolę w pracy administratora takiego systemu komputerowego. Błędy występują zawsze i takie „logi" są pierwszym krokiem w działaniu administratora by te problemy rozwiązać.

Dodatkowym problemem (np. w dużych firmach) może być kwestia zarządzania takimi logami. Dysponując wieloma serwerami (setki, tysiące) proces zarządzania logami odgrywa kluczową rolę (szybkość i czytelność odpowiedzi).

Innymi pytaniami, które powinien zadać sobie każdy administrator dot. logowania zdarzeń systemu są np.:

- kwestia uprawnień (kto może mieć dostęp do takich informacji?),
- jak długo należy trzymać historię logowania zdarzeń systemowych.

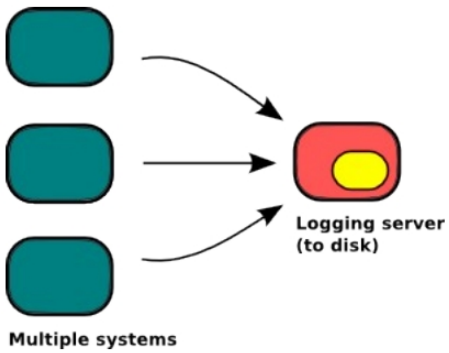
Narzędziem, które do tego się świetnie nadaje i które zaprezentujemy jest rsyslog dla systemów Linux. Postaramy się poruszyć zagadnienia techniczne tego narzędzia jak i zachęcić do jego używania.

# Modele rejestrowania zdarzeń systemu

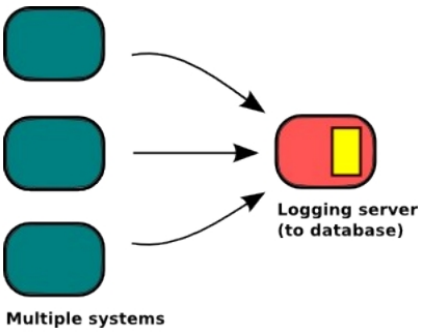
Wyróżniamy kilka typowych modeli architektonicznych związanych z logowaniem zdarzeń systemu:

1. Jeden system → dysk twardy - na jednej maszynie, która służy jako serwer zapisywane są logi


2. Wiele serwerów → jeden dysk - taki model przedstawia tzw. centralne logowanie - wiele maszyn wysyła swoje logi na dysk innej centralnej maszyny przechowującej rejestr zdarzeń.




3. Wiele serwerów → baza danych - takie rozwiązanie pomoga dodatkowo wystawić obslugę logów np. do sieci web jako stronę www (wykonywać zapytania etc.). Baza danych może być wystawiona na innym serwerze.




## Kwestie techniczne

Tradycyjny syslog w Unixie korzysta do łączenia się z serwerami protokołu UDP. Takie rozwiązanie nie jest najlepsze do scentralizowanych serwerów logujących - dlaczego? O UDP można mówić jak o protokole stratnym - w przypadku zerwania połączenia z serwerem tracimy informację o tym jakie wiadomości nie zostały wysłane (w przypadku protokołu TCP sytuacja wygląda podobnie). W przypadku rsysloga mamy implementację protokołu RELP (który w takich sytuacjach nie zawodzi).

Relp używa modelu klient-serwer z (głównie ustalonych ról. "Wysyłanie" nazywa się klientem, "słuchacz" to serwer.

Relp bardziej szczegółowo działa na zasadzie "command-response". Czyli klient wydaje polecenia, na które serwer odpowiada. Aby ułatwić komunikację full-duplex, kilka komend może być wydane w tym samym czasie i wiele odpowiedzi może być wysłanych w danej chwili. Aby zaoszczędzić zasoby, liczba poleceń jest ograniczona. Każda komenda ma przypisany (względnie) unikalny, monotnicznie rosnący ID ("transaction numer" lub skrót "txnr"). Każda odpowiedź zawiera ID polecenia. Odpowiedzi serwera są wysyłane w tej samej kolejności co odebrane polecenia.

W przypadku gdy logujemy zdarzenia do bazy danych musimy również wziąć pod uwagę jej “wytrzymałość” - baza danych w ciągu bardzo krótkiego czasu może obsługiwać tysiące zapytań co znacząco może wpłynąć na jej wydajność.

## Instalacja rsyslog

Instalacja w systemie Ubuntu przebiega bez problemowo, wystarczy wykonać następujące zapytanie:

```
$ sudo aptitude install rsyslog
```

W celu uzyskania dodatkowej dokumentacji trzeba wykonać:

```
$ sudo aptitude install rsyslog-doc
```

Konfiguracja rsyslog znajduje się w pliku: `/etc/rsyslog.conf` bądź w plikach znajdujących się w katalgou `/etc/rsyslog.d`.

W tej chwili rsyslog powinien być już uruchomiony (rsyslog skonwertował sobie plik konfiguracyjny syslog.conf).

Funkcjonalność logowania lokalnego jest dostępna poprzez plugin imuxsock (domyślnie włączony) - pozycja w configu:

```
$ModLoad imuxsock
$ModLoad imklog
```

Dokonując zmian w konfiguracji rsyslog potrzebuje oczywiście restartu, zrestartować usługę można w następujący sposób:

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

# Plik konfiguracyjny

Każdy plik konfiguracyjny jest podzielony na trzy części:

- moduły,
- globalne dyrektywy,
- reguły filtrów.

Deklaracje w pliku modułów i dyrektyw muszą znajdować się oddzielnie w każdej linii i dodatkowo zaczynać się znakiem $.

# Budowanie reguł i akcji

Reguły budujemy w taki sposób w jaki jest to zrobione w sysklogu. Format wyglada tak:

```
mail.info       -/var/log/mail.info
news.crit       /var/log/news/news.crit
```

Są to dwa domyślne wpisy z pliku konfiguracyjnego rsysloga.

W pierwszej kolumnie znajdują się tzw. selectory (jest ich z góry ustalona liczba), łączyć je należy z priorytetami. Patrz poniższe tabele:

| **Facilities** | | |
|---|---|---|
| Numerical Code | Keyword | Facility |
| 0 | kern | Kernel |
| 1 | user | Regular user processess |
| 2 | mail | Mail system |
| 3 | daemon | System daemons |
| 4 | auth | Security (authentication and authorisation) related commands |
| 5 | syslog | Syslog internal messages |
| 6 | lpr | Line printers system |
| 7 | news | NNTP subsystem |
| 8 | uucp | UUCP subsystem |
| 10 | authpriv | Private authorisation messages |
| 16-23 | local0-7 | Site specific use |

| **Priorities** | | |
|---|---|---|
| Numerical Code | Keyword | Facility |
| 0 | emerg | Emergency: system is unusable |
| 1 | alert | Alert: action must be taken immediately |
| 2 | crit | Critical: critical conditions |
| 3 | err | Error: error conditions |
| 4 | warning | Warning: warning conditions |
| 5 | notice | Notice: normal but significant conditions |
| 6 | info | Informational: informational messages |
| 7 | debug | Debug: debug level messages |

Selector może zawierać znak “*”, który oznacza “wiele” i mówi to tyle, że możemy do wielu obiektów systemu się odwołać w ten sposób. Również możemy działać na priorytetach w ten sposób, wykorzystujemy do tego specjalne operatory, np.: ‘=’, ‘!’.

Przykłady użycia:
```
auth,authpriv.* /var/log/auth.log
*.*;auth,authpriv.none /var/log/syslog
```
Taki zapisek kombinacji dwóch reguł oznacza, że do dwóch obiektów z dowolnym priorytetem zostanie wysłany komunikat do jednego pliku i wszystko inne zostanie wysłane do drugiego pliku.

```
mail.=info
```
Reguła ta wybiera wiadomości dla konkretnego obiektu o konkretnym priorytecie (info).

```
kern.*;kern.!=warn /var/log/kernel/nowarnings
```
Reguła ta wysyła do jednego obiektu wszystkie komunikaty z wyjątkiem tych oznaczonych jako “warning”.

## Timestamp

Rsyslog posiada precyzyjne wsparcie timestamp’ów, które są domyślnie wyłączone).Funkcja ta jest
kontrolowana przez następujący parametr:
```
$ MainMsgFileDefaultTemplate RSYSLOG_TraditionalFileFormat
```
Wystarczy linię zakomentować aby uzyskać tą funkcję czyli timestamp’y wysokiej precyzji.

Oto porównanie timestamp’ów tradycyjnych i precyzyjnych:

```
Nov 17 12:01:44 client_hostname ubuntu: test
2008-11-17T12:02:47.372490-05:00 client_hostname ubuntu: test
```

## Szablony

Szablony użytkownik może wykorzystać do formatowania kolejkowanych komunikatów, które są wysyłanie do plików.

W pliku konfiguracjynym definijemy szablony następująco:
```
$template template_name,"text %property% text %property%\n"
,<options>
```

Poniżej prosty przykład szablonu, który dodaje określoną frazę po komunikacie logu:
```
$template simple_template,"text_before %msg% text_after\n"
text_before  test text_after
```

## Przetwarzanie kolejki komunikatów

Wyszystkie komunikaty logów trafiają do jednej kolejki, później są one rozdzielane na kolejki wg akcji jakie są skonfigurowane. Wszystkie te kolejki są wykonywane seryjnie, co znaczy tyle, że czas trwania wszystkich operacji jest taki ile wynosi suma kolejek wszystkich akcji. W pewnych przypadkach może to powodować problemy ponieważ jedna potężna kolejka (wolna) może

znacząco wpłynać na cały proces przetwarzania. Takim przykladem może być np. odowłanie się do bazy dancyh przez akcję, która w tym czasie jest niedostępna - przez czas oczekiwania na taką bazę danych na połączenie cały proces się wydłuża bo komunikat o timeoutcie dostaniemy po pewnym czasie.

Aby temu zapobiec w rsyslogu wystarczy zdefiniować na nowo (w wyszczególniony sposób) `$MainMsgQueue`.

## Wsparcie relacyjnych baz danych

Rsyslog wspiera następujące silniki baz danych:

- MySql
- PostgreSQL
- Oracle
- SQLite
- MS SQL
- SyBase

i inne...
Bazy danych MySQL i PostgreSQL są wspierane pluginami i wystarczy je zainstalować.

```
$ sudo aptitude install rsyslog-mysql
```

Następnie konfigurujemy moduł ommysql:

```
$ModLoad ommysql
*.* :ommysql:localhost,Syslog,rsyslog,abc123
```

(należy sprawdzić plik /etc/rsyslog.d/mysql.conf).

Przy instalacji paczki wszystko dzieje sie za nas automatycznie. Rsyslog tworzy bazę Syslog a w niej dwie tabele: SystemEvents i SystemEventsProperties

W czasie instalacji zostaniemy oczywiście poproszeni o hasło (dla użtkownika bazy danych rsyslog, zostanie ono wygenerowane jeżeli będzie to pierwszy raz). Jednym prawem potrzebnym do działania tego użytkownika jest INSERT.