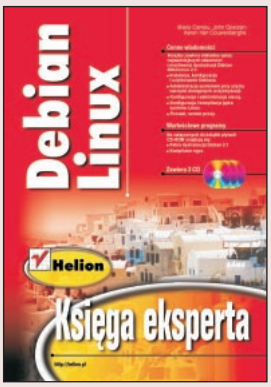
Debian Linux. Księga eksperta



Autorzy: Mario Camou, John Goerzen, Aaron Van Couwenberghe
Tłumaczenie: Maciej Kulawski
ISBN: 83-7197-297-0
Tytuł oryginału: Debian GNU/Linux 2.1 Unleashed
Format: B5, stron: 696
oprawa twarda
Zawiera 3 CD-ROMy

Książka ta zawiera dokładne opisy najważniejszych własności i możliwości dystrybucji Debian GNU/Linux 2.1:

- Instalacja, konfiguracja i użytkowanie systemu.
- Administracja systemem przy użyciu narzędzi dostępnych w dystrybucji.
- Konfiguracja i administracja siecią.
- Konfiguracja i kompilacja jądra systemu Linux.
- ZFirewall, serwer proxy.

„Debian Linux. Księga Eksperta" to wspaniała książka! Będzie szczególnie przydatna wszystkim początkującym użytkownikom, którzy próbowali kiedyś zainstalować Debiana i korzystać z jego mozliwości. Również bardziej zaawansowani znajdą dla siebie wiele cennych informacji.
Na płycie CD-ROM znajdziesz:

- dystrybucję Debian 2.1;
- kompilator egcs;

Kilka słów o Autorach będzie doskonałym uzupełnieniem rekomendacji tej książki:

- Mario Camou pracuje jako programista Javy w jednej z firm z rankingu Fortune 10. Do września 1999 roku pełnił funkcję Director of Technology w firmie Umbral, właścicielu jednego z najszybciej rozwijających się portali internetowych. Z Linuksem pracuje od siedmiu lat.
- Aaron Van Couwenberghe pracuje nad rozwojem Debiana od 1998 roku, a używa go od samego początku.
- John Goerzen pracuje nad rozwojem Debiana od 1996 roku, obecnie zajmuje się integracją pakietów i przenoszeniem tej dystrybucji na platformę DEC Alpha. Zawodowo zajmuje się administracją systemem i programowaniem dla firmy internetowej.

Książki z serii „Księga Eksperta" odbiegają od zwykłych publikacji technicznych. Czytelnik znajdzie w nich praktyczne porady i dogłębne opisy.

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Rozdział 6.
Dzienniki systemowe i rozliczanie wykorzystania zasobów

Pliki dzienników systemowych są bardzo ważnym elementem każdego systemu. Pozwalają na monitorowanie jego pracy i wykrywanie anomalii. Pozwalają także na zbieranie statystyk takich jak np. czas największej zajętości systemu lub liczba dostępów do systemu przypadająca na jednego użytkownika. Wiedza na temat tego, jak pracuje system rejestrowania w Linuksie oraz jak go skonfigurować dla potrzeb naszego ośrodka, jest bardzo istotna.

Wprowadźmy zwyczaj oglądania od czasu do czasu plików dzienników systemowych. Pozwoli nam to na wyrobienie sobie zdania na temat liczby i rodzaju komunikatów, które pojawiają się w czasie poprawnej pracy systemu. W takim przypadku, jeżeli zdarzy się nieprzewidziana sytuacja, łatwo odszukamy te pozycje dzienników, które są symptomem anormalnego zachowania systemu. Jak często powinniśmy sprawdzać pliki dzienników? W bardzo obciążonych serwerach możemy sprawdzać je raz, a nawet dwa razy dziennie. Jednak w osobistej stacji roboczej wystarczy robić to raz na tydzień.

Rozliczanie (ang. *accounting*) jest procesem, dzięki któremu system obserwuje wykorzystanie określonego zasobu. Na przykład rozliczanie procesów pozwala na przechowywanie informacji o tym, przez jaki czas określony proces wykorzystywał procesor i pamięć. Z kolei rozliczanie pracy w sieci pozwala na śledzenie wykorzystanie pasma. Ten mechanizm jest bardzo przydatny, szczególnie w konfiguracji produkcyjnej, gdzie bardzo istotna jest wiedza o tym, kto korzystał z jakiego zasobu (jak np. w systemach realizujących funkcję *chargeback*[1]). Dzięki temu można także obserwować procesy, nad którymi straciliśmy kontrolę, bez wpływu na wydajność systemu.

W tym rozdziale zbadamy, w jaki sposób dwa demony, *syslogd* oraz *klogd*, obsługują wszystkie zadania rejestrowania w systemie Linux oraz jak możemy zarządzać plikami

[1] Chargeback — obciążanie finansowe użytkowników systemu za wykorzystywanie zasobów — *przyp. red.*

dzienników, które tworzą. Następnie przyjrzymy się szerokiemu zakresowi poleceń monitorowania wykorzystania dysku, operacji sieciowych, ładowania procesów oraz logowania użytkowników do systemu.

Dzienniki systemowe

W systemie Linux można zarejestrować praktycznie wszystkie zdarzenia. Pliki dzienników są zazwyczaj umieszczane w katalogu */var/log* i może je odczytywać tylko użytkownik *root*. Jednakże można tak skonfigurować system oraz wiele programów, żeby ich dzienniki były zapisywane w innym miejscu. Często można je nawet wysyłać do innych komputerów w sieci.

Funkcja rejestrowania w systemie Linux opiera się na dwóch demonach: *syslogd* oraz *klogd*. Są one zawarte w pakiecie *sysklogd.* Główne zadania systemu dzienników spoczywają na demonie *syslogd*. Otrzymuje on komunikaty z różnych procesów i rozprowadza je zgodnie z plikiem konfiguracyjnym. Z kolei *klogd* jest demonem dzienników jądra. Jego funkcja to zbieranie komunikatów generowanych przez jądro i przekazywanie ich do demona *syslogd.*

Demon syslogd

Większość funkcji rejestrowania w systemie UNIX jest wykonywana przez demona *syslogd*. Jest on zazwyczaj uruchamiany w czasie ładowania systemu w każdym komputerze z tym systemem. Jego działanie polega na monitorowaniu żądań rejestrowania od programów oraz innych komputerów i wysyłanie ich do różnych plików docelowych, zgodnie ze źródłem oraz stopniem ważności. W niektórych programach można dokonać wyboru pomiędzy stosowaniem plików dzienników programu lub korzystaniem z demona *syslogd*. W takich przypadkach zwykle lepiej zastosować *syslogd*., ponieważ zapewnia on jednolity interfejs konfiguracji rejestrowania. Poza tym *syslogd* gwarantuje, że każdy komunikat przez niego zarejestrowany będzie zawierać co najmniej etykietę czasową, nazwę hosta, który wygenerował ten komunikat i zwykle też nazwę programu (chociaż to ostatnie zależy od programu wysyłającego komunikat).

Spójrzmy na to, w jaki sposób demon *syslogd* przyjmuje komunikaty przekazywane przez inne programy. Przyjrzyjmy się także temu, w jaki sposób można dostosować działanie *syslogd* do każdego rodzaju komunikatu poprzez modyfikację pliku */etc/ syslog.conf*.

Demon *syslogd* jest wszechstronny. Może on być skonfigurowany do wykonywania centralnego rejestrowania, tak by komunikaty z całej sieci trafiały do pojedynczego hosta, a także do przetwarzania komunikatów na różne sposoby. Może na przykład wysyłać niektóre komunikaty na konsolę, inne do pliku, jeszcze inne na terminale konkretnych użytkowników w czasie, kiedy są zalogowani, a niektóre do wszystkich tych miejsc. Konfigurację wykonuje się przez edycję pliku */etc/syslog.conf*. Temu zagadnieniu jest poświęcona osobna część tego rozdziału.

Korzystanie z demona syslogd

Demon *syslogd* jest uruchamiany automatycznie podczas startu systemu przez skrypt */etc/init.d/sysklogd*. Jest on dowiązany do katalogów */etc/rc?.d* jako *S10sysklogd*, tak więc uruchamia się bardzo wcześnie w procesie ładowania. Aby uzyskać więcej informacji na temat procesu ładowania, a także katalogów */etc/init.d* oraz */etc/rc?.d* przeczytaj rozdział 5. „Adaptacja procedury startowej". Demon *syslogd* może być uruchomiony z kilkoma parametrami, które wymieniono w tabeli 6.1. Aby zastosować dowolne z tych opcji, powinniśmy dokonać edycji pliku */etc/init.d/sysklogd* i dodać je do wiersza wywołującego demona *syslogd*.

Tabela 6.1.
Opcje demona syslogd

Opcja	Działanie
`-d`	Włączenie trybu śledzenia. Opcję stosuje się tylko w czasie śledzenia działania programu *syslogd*
`-f plik_konf`	Wykorzystanie innego pliku konfiguracyjnego niż */etc/syslog.conf*
`-h`	Domyślnie *syslogd* nie przekazuje żadnych komunikatów, które otrzymuje od innych hostów. Ten przełącznik określa, ze *syslogd* powinien przekazywać wszystkie otrzymywane komunikaty zależnie od przekazujących hostów określonych w pliku konfiguracyjnym
`-l listahost`	Dowolne hosty wymienione w liście `listahost` oddzielone dwukropkami będą dziennikowane poprzez tylko nazwę hosta z pominięciem domeny
`-m przedział`	Określa przedział w sekundach pomiędzy komunikatami znaczników. Komunikaty znaczników są wysyłane okresowo przez *syslogd* do funkcji `mark`. Domyślną wartością jest 20 sek.
`-n`	Zazwyczaj w czasie uruchamiania *syslogd* przemieszcza się na drugi plan (tak jak wiele innych demonów). Ta opcja określa, że *syslogd* powinien pozostać na pierwszym planie. Jest to bardzo ważne w niektórych przypadkach, np. kiedy *syslogd* uruchamiamy bezpośrednio z pliku */etc/inittab*. Aby uzyskać więcej informacji o */etc/inittab*, przeczytaj rozdział 5.
`-p gniazdo`	Zwykle *syslogd* w celu wydobywania komunikatów od lokalnych procesów monitoruje gniazdo typu UNIX */dev/log*. Ta opcja zmienia wykorzystywane tego gniazda
`-r`	Domyślnie *syslogd* nie monitoruje sieci. Z tą flagą *syslogd* monitoruje sieć w gniazdach wymienionych jako *sysklog* w pliku */etc/services*, zwykle UDP port 514 (aby uzyskać więcej informacji na temat protokołów sieciowych przeczytaj rozdział 9. „Najważniejsze informacje o TCP/IP"
`-s lista_domen`	Dowolne domeny zawarte w `lista_domen` (oddzielone średnikami) będą odrzucone z nazwy hosta przed zapisaniem komunikatu. Zastosowana będzie pierwsza zgodna domena. Tak więc jeżeli komunikat pochodzi z serwera *serwer.wsch.firma.com*, a wpiszemy `-s firma.com:wsch.firma.com`, będzie uwzględniony pierwszy zapis i dziennik będzie zawierał nazwę hosta jako *serwer.wsch*. W tym przypadku powinniśmy zamienić kolejność nazw domen i wpisać: `-s wsch.firma.com:firma.com`
`-v`	wyświetla wersję demona *syslogd* i kończy działanie

Demon *syslogd* odpowiada na kilka sygnałów. Można je wykorzystać do komunikacji z uruchomionym demonem *syslogd*. Numer ID bieżącego procesu *syslogd* można znaleźć w pliku */var/run/syslogd.pid*. Jeżeli chcemy wysłać sygnał do uruchomionego demona *syslogd*, możemy zastosować polecenie `kill -SIGNALL `cat /var/syslogd.pid``. W tabeli 6.2 przedstawiono listę sygnałów, które mogą być przesłane do demona *syslogd*.

Tabela 6.2.
Sygnały obsługiwane przez demona syslogd

Sygnał	Działanie
`SIGHUP`	Ponowna inicjacja programu *syslogd*. Demon *syslogd* zamyka wszystkie otwarte pliki, otwiera je ponownie, czyta jeszcze raz plik konfiguracyjny i uruchamia ponownie system rejestrowania
`SIGTERM`	Zakończenie działania systemu *syslogd*
`SIGINT` oraz `SIGQUIT`	Zakończenie działania *syslogd*, chyba że jest włączone śledzenie
`SIGUSR1`	Włącza i wyłącza śledzenie, ale tylko wtedy, gdy *syslogd* był uruchomiony z opcją `-d`.

Plik /etc/syslog.conf

Demon *syslogd* jest konfigurowany poprzez plik */etc/syslog.conf*. Każdy wiersz pliku */etc/syslog.conf* składa się z dwóch części: selektora oraz akcji oddzielonych od siebie spacją.

Selektor wskazuje, do których komunikatów ma być zastosowane działanie. Akcja określa, co należy zrobić z tymi komunikatami. Każdy wiersz poprzedzony znakiem # jest traktowany jako komentarz.

Selektory

Komunikaty są wybierane według dwóch parametrów. Pierwszy z nich to funkcja. Wskazuje on program lub usługę skąd pochodzi komunikat. Ponieważ liczba nazw funkcji jest ograniczona, niektóre programy są zmuszone do korzystania z nazw funkcji, które nie odpowiadają ich rzeczywistym nazwom. Można zastosować następujące nazwy funkcji:

```
auth          user
auth-priv     uucp
cron          local0
daemon        local1
kern          local2
lpr           local3
mail          local4
mark          local5
news          local6
security      local7
syslog
```

Niektóre z nich mają specjalne znaczenie. Funkcja `security` jest przestarzałym odpowiednikiem funkcji `auth`, natomiast `mark` jest wewnętrzną funkcją wysyłającą komunikaty do demona *syslog* co 20 sekund. Funkcje od `local0` do `local7` mogą być wykorzystane dla dowolnego działania zdefiniowanego lokalnie.

Drugim parametrem wybierania komunikatów jest priorytet. Jest to wskazanie ważności komunikatu. W kolejności rosnących ważności priorytety są następujące:

```
debug          error
info           crit
notice         alert
warnig         emerg
warn           panic
err
```

Priorytety `error`, `warn` oraz `panic`, to przestarzałe odpowiedniki priorytetów `err`, `warning` oraz `emerg`. Nie należy z nich korzystać.

Selektor ma formę `funkcja.priorytet`. Ta forma jest zgodna z wszystkimi komunikatami wybranego priorytetu i wyższymi. Tak więc np. `mail.info` jest zgodny z wszystkimi komunikatami `mail` z wyjątkiem `debug`. Jeżeli chcemy odnieść się wyłącznie do określonego priorytetu, możemy zastosować znak równości (=). Tak więc `mail.=info` będzie odpowiadać tylko komunikatom `mail` o priorytecie `info`.

Istnieją inne znaki o specjalnym znaczeniu. Gwiazdka (*) oznacza wszystko, zatem `mail.*` oznacza wszystkie komunikaty `mail` (co jest równoznaczne z `mail.debug`), natomiast `*.crit` jest zgodne z wszystkimi komunikatami o priorytecie `crit` i wyższymi.

Wykrzyknik (`!`) oznacza negację warunku. Tak więc `mail.!info` jest równoważne z `mail.=debug`, natomiast `mail.info` jest równoważne z `mail.!=debug`.

Możemy połączyć kilka funkcji w jednym selektorze, stosując przecinki, na przykład `mail,news.crit` odpowiada wszystkim komunikatom o priorytecie `crit` lub wyższym dla usług `mail` oraz `news`. Możemy także połączyć kilka selektorów za pomocą średnika. W tym przypadku możemy wykorzystać specjalny priorytet `none` do określenia braku komunikatów z danej usługi. Na przykład `*.crit;mail.none` odpowiada wszystkim komunikatom na poziomie `crit` lub wyższym z wyjątkiem tych pochodzących z funkcji `mail`.

Akcje

Pole akcji określa działanie, jakie ma być wykonane z komunikatami zgodnymi z danym selektorem. Pierwszy znak akcji określa jej typ.

Akcja. która zaczyna się od znaku ukośnika (/) określa nazwę pliku, do którego będą dodawane komunikaty (np. */var/log/messages*). Jeżeli plik jest urządzeniem *tty*, będzie zastosowana specjalna obsługa *tty*. W celu skierowania komunikatów na konsolę, należy zastosować */dev/console*.

Znak minus (-) poprzedzający znak ukośnika (/) również określa plik, ale wskazuje, że zapis do pliku nie będzie wykonywany po każdej aktualizacji. Dzięki temu demon *syslog* jest bardziej wydajny. Jednakże istnieje ryzyko utraty niektórych komunikatów w przypadku awarii systemu. Tej akcji nie powinniśmy stosować z bardzo ważnymi dziennikami np. *-/var/log/ppp.log*.

Znak `@` określa nazwę hosta, do którego mają być wysyłane komunikaty na przykład *@logger.firma.com*.

Znak (|) określa nazwę potoku (*fifo*), gdzie mają być kierowane komunikaty — np. *|/dev/xconsole*. Potok powinien zostać utworzony za pomocą polecenia `mkfifo` przed uruchomieniem demona *syslogd*.

Dowolne znaki alfanumeryczne określają początek listy oddzielanych przecinkami użytkowników. Jeżeli są oni zalogowani, komunikaty będą wysyłane na ich terminale. Lista może mieć np. taką postać: `mario, root, admin`.

Gwiazdka (*) wskazuje, że komunikat będzie wyświetlony na terminalach wszystkich zalogowanych użytkowników.

Wydruk 6.1 przedstawia domyślną zawartość pliku */etc/syslog.conf* systemu Debian.

Wydruk 6.1.
Plik /etc/syslog.conf

```
#  /etc/syslog.conf    Configuration file for syslogd.
#
#                      For more information see syslog.conf(5)
#                      manpage.

#
# First some standard logfiles.  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
uucp.*                          -/var/log/uucp.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

# Logging for INN news system
#
```

```
news.crit                       /var/log/news/news.crit
news.err                        /var/log/news/news.err
news.notice                     -/var/log/news/news.notice

#
# Some `catch-all' logfiles.
#
*.=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                                 *

#
# 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

# The named pipe /dev/xconsole is for the `xconsole' utility.  To # use it,you must
  invoke `xconsole' with the `-file' option:
#
#    $ xconsole -file /dev/xconsole [...]
#
# NOTE: adjust the list below, or you'll go crazy if you have a
#       reasonably busy site..
#
daemon.*;mail.*;\
     news.crit;news.err;news.notice;\
     *.=debug;*.=info;\
     *.=notice;*.=warn |        /dev/xconsole

local2.*              -/var/log/ppp.log
```

Aby stworzyć pojedynczy host rejestrowania w sieci, możemy zmienić zawartość plików *syslog.conf* we wszystkich innych hostach na następującą postać:

```
#Send all logs to logger
*.*  @logger
```

Jeżeli to zrobimy, powinniśmy dokonać edycji pliku */etc/init.d/sysklogd* w komputerze *logger*, dodać opcje `-r` do wiersza uruchomienia demona *syslogd* i wydać polecenie `/etc/init.d/sysklogd restart` (lub ponownie uruchomić hosta *logger*). W innym przypadku demon *syslogd* w komputerze *logger* nie będzie monitorował komunikatów sieciowych.

Demon klogd

W systemie Linux komunikaty jądra są tradycyjnie kierowane na konsolę. Jednakże w wielu przypadkach nie jest to pożądane. Na przykład serwer pracujący zdalnie bez nadzoru. Informacje dotyczące awarii systemu mogą być przesyłane na konsolę, gdzie nikt nie może ich przeczytać. Nawet jeżeli jest tam operator, komunikaty mogą spowodować przewijanie ekranu lub mogą być usunięte w przypadku ponownego uruchomienia serwera.

Demon *klogd* został stworzony do rozwiązania tego problemu. Jest on tak zaprojektowany, by przechwytywał wszystkie komunikaty jądra i przekazywał je do demona *syslogd.* Demon *syslogd* może wykonywać z nimi bardziej skomplikowane operacje (a nawet wyświetlać je na konsoli systemowej, jeżeli tego właśnie sobie życzymy).

Jeżeli jest zamontowany system plików */proc*, jądro udostępnia swoje komunikaty poprzez pseudo-plik */proc/kmsg*. Demon *klogd* czyta te komunikaty z pliku i kończy działanie. Jeżeli system plików */proc* nie jest zamontowany, demon *klogd* czyta komunikaty za pomocą odwołań systemowych. Aby uzyskać więcej informacji na temat systemu plików */proc* przeczytaj rozdział 8. „Zaawansowana administracja systemem"

Demon *klogd* rejestruje komunikaty za pomocą funkcji `kern`. Priorytet jest przydzielany przez samo jądro. Demon *klogd* dokonuje konwersji tych priorytetów na określony priorytet demona *syslogd.* Priorytety jądra są wymienione w pliku */usr/include/linux/kernel.h*. Poniżej przedstawiono fragment tego pliku łącznie z podanym znaczeniem każdego priorytetu:

```
#define KERN_EMERG    "<0>"   /* praca systemu jest niemożliwa      */
#define KERN_ALERT    "<1>"   /* należy natychmiast podjąć działania*/
#define KERN_CRIT     "<2>    /* warunki błędu krytycznego          */
#define KERN_ERR      "<3>"   /* warunki błędu                      */
#define KERN_WARNING  "<4>"   /* warunki ostrzeżenia                */
#define KERN_NOTICE   "<5>"   /* warunki normalne, ale system wymaga*/
                              /* szczególnej uwagi                  */
#define KERN_INFO     "<6>"   /* informacyjne                       */
#define KERN_DEBUG    "<7>"   /* komunikaty śledzenia               */
```

W normalnej sytuacji będziemy rejestrować wszystkie komunikaty dotyczące warunków, które mogą wywrzeć wpływ na działanie systemu. Będą to komunikaty o priorytecie `notice` lub wyższym. Być może będą to także komunikaty o priorytecie `info`. Włączenie na kilka dni rejestrowania wszystkich komunikatów do różnych plików, da nam obraz, jakie komunikaty są przydatne (oczywiście nie powinniśmy otrzymać żadnych komunikatów powyżej priorytetu `notice!`). Aby wykonać tę czynność, dodajmy następujące wiersze do pliku */etc/syslog.conf*:

```
kern.=debug   /var/log/kernel.debug
kern.=info    /var/log/kernel.info
kern.=notice  /var/log/kernel.notice
kern.=warning /var/log/kernel.warning
kern.=err     /var/log/kernel.err
kern.=crit    /var/log/kernel.crit
kern.=emerg   /var/log/kernel.emerg
```

Administracja i utrzymywanie dzienników

Jeżeli dzienniki systemowe pozostawimy bez nadzoru, będą one rosły, aż w końcu zapełnią przypisaną partycję dysku. Dlatego dobrym pomysłem jest przypisanie oddzielnej partycji do katalogu */var* w dowolnym serwerze działającym w firmie. Jeżeli tego nie zrobimy, dzienniki rozrosną się do zbyt dużych rozmiarów i wypełnią partycję *root*, co sprawi, że praca w systemie będzie praktycznie niemożliwa. Aby uzyskać więcej informacji na temat dzielenia dysku na partycje przeczytaj rozdział 4. „Najważniejsze zagadnienia administracji systemem".

Aby zapobiec niekontrolowanemu rozrastaniu się plików dzienników, system Debian zawiera narzędzie o nazwie *savelog*. Zadaniem tego narzędzia jest zarządzanie rotacją dzienników systemowych. Oznacza to okresową obsługę przesuwania plików dzienników i śledzenie konfigurowanej liczby plików archiwalnych.

Za każdym razem, kiedy działa program *savelog*, zmienia on nazwę bieżącego pliku dziennika z *pliklog* na *pliklog.0*. Jeżeli istnieje *pliklog.0*, zmienia mu nazwę na *pliklog.1* itd. Kiedy liczba archiwalnych dzienników systemowych osiągnie liczbę określoną w konfiguracji, program *savelog* może też skompresować te pliki, żeby zajmowały mniej miejsca. Nigdy nie jest kompresowany plik *pliklog.0*, ponieważ mogą być do niego zapisywane informacje w trakcie trwania rotacji.

Program *savelog* jest zwykle uruchamiany okresowo przez narzędzie *anacron* (aby uzyskać więcej informacji na temat programu *anacron* przeczytaj rozdział 4.). Jest on uruchamiany raz dla każdego rotowanego pliku dzienników. Tak więc każda aplikacja wysyłająca komunikaty zwykle ma swój własny skrypt w pliku */etc/cron.daily* lub */etc/cron.weekly*. Możemy dokonać edycji tych plików, jeżeli chcemy wykonać jakieś czynności przed albo po rotacji dzienników (jak np. generowanie statystyki). Program *savelog* akceptuje jako parametr nazwy plików do rotacji. Akceptuje także kilka opcji, które wyszczególniono w tabeli 6.3.

Tabela 6.3.
Opcje programu savelog

Opcja	Działanie
`-m atrybut`	Atrybut pliku historycznego jest zmieniany na `atrybut` (ósemkowo)
`-u uzytkownik`	Właściciel pliku archiwalnego jest zmieniany na `uzytkownik`
`-g grupa`	Grupa pliku historycznego jest zmieniana na `grupa`
`-c cykl`	Liczba cyklów. Pliki archiwalne są przechowywane w plikach z rozszerzeniami *.0* do *.cykl-1*
`-t`	Tworzy pusty plik dziennika po rotacji (domyślnie nie tworzy go)
`-l`	Nie kompresuje plików archiwalnych (domyślnie są kompresowane)
`-p`	Ochrania atrybut, właściciela oraz grupę pliku dzienników

W czasie korzystania z *savelog* może wystąpić jeden problem. Proces zapisujący do pliku dziennika musi go zamknąć, a następnie ponownie otworzyć po dokonaniu rotacji.

W innym przypadku będzie on kontynuował zapis do *pliklog.0*. W takiej sytuacji, kiedy *savelog* uruchomi się następny raz, nazwa pliku *pliklog.0* będzie zmieniona *pliklog.1*, który będzie potem skompresowany. Prawdopodobnie spowoduje to utratę niektórych danych. Aby zapobiec takiej sytuacji, należy zapewnić zamykanie i ponowne otwieranie pliku dziennika natychmiast po uruchomieniu programu *savelog*. Wiele demonów systemowych robi to w czasie, kiedy wysyła sygnał `SIGHUP`, inne muszą być zatrzymane i ponownie uruchomione. Należy sprawdzić dokumentację demona, aby zobaczyć, czy jest to konieczne i jak należy to zrobić.

Demony *syslogd* i *klogd* pozwalają na zarządzanie sposobem zapisu komunikatów generowanych przez procesy oraz jądro systemu Linux do dzienników systemowych. Pliki dzienników są przydatne, kiedy chcemy się przekonać, czy w systemie miało miejsce określone zdarzenie.

Utrzymywanie dzienników jest bardzo ważne, ale nie wystarcza. Musimy także posiadać mechanizm pozwalający na monitorowanie wykorzystania ważnych zasobów takich jak czas procesora, pamięć RAM czy zajętość dysku. Następna część opisuje niektóre z tych mechanizmów.

Rozliczanie wykorzystania zasobów

Jak wspomniano wyżej, rozliczanie jest procesem tworzenia podsumowań wykorzystania zasobu przez użytkownika lub proces. Istnieje kilka rodzajów rozliczania, które mogą być stosowane w różnych celach. Na przykład możemy liczyć czas przebywania online, w przypadku gdy nasz dostawca usług internetowych obciąża nas za czas korzystania z sieci Internet. Możemy też generować dziennik wykorzystania przestrzeni na dysku dla oddziału realizującego funkcję *chargeback* w naszej firmie. Mimo że do realizacji funkcji rozliczania może być wykorzystanych wiele narzędzi, ten fragment skupia się na tych, które występują w dystrybucji Debian, wykorzystując dodatkowe pakiety tylko wtedy, gdy występuje oczywisty brak funkcji.

Rozliczanie wykorzystania dysku

Rozliczanie wykorzystania dysku jest najlepiej wykonywane przy zastosowaniu limitów (ang. *quota*). Jednakże informacje stosowane w metodzie limitów są bardzo nieprecyzyjne. Ograniczają się tylko do ilości przestrzeni na dysku, którą zajmuje użytkownik w każdym systemie plików. Jeżeli potrzebujemy bardziej szczegółowych informacji (np. ile dany plik zajmuje miejsca lub jakie pliki należą do określonych właścicieli), możemy zastosować dwa standardowe polecenia UNIX: `du` oraz `find`. Przyjrzymy się teraz określonym zastosowaniom tych narzędzi.

Wykorzystanie du do odszukiwania plików zajmujących dużo przestrzeni na dysku

Program *du*, razem z poleceniem `sort`, może być wykorzystany w celu dokładnej analizy systemu plików, aby znaleźć plik, który zajmuje dużo miejsca.

Wydruk 6.2 przedstawia przykładową sesję, gdy system plików */var* zapełnia się i chcemy dowiedzieć się, które pliki zajmują miejsce.

Wydruk 6.2.
Wykorzystanie du do odnajdywania plików zajmujących dużo przestrzeni na dysku

```
[root@atman ~]#df -k /var
Filesystem    1024-blocks  Used   Available   Capacity   Mounted on
/dev/hda6       147983   147983          0       100%     /var
[root@atman log]# cd /var
[root@atman /var]# du -sk *.??* | sort -n
du: .??* No such file or directory
1   local
1   nis
1   preserve
1   yp
2   lock
12   lost+found
15   named
20   run
4508   lib
12759   tmp
68620   log
75148   spool
[root@atman /var]#du -sk {log,spool}/{*,.??*}|sort -n
du: log/.??*: No such file or directory
du: spool/.??*: No such file or directory
0   log/spooler
0   log/xferlog
1   log/lastlog.1.gz
[… wycięto część wyniku dla zachowania zwięzłości…]
244   log/maillog.2.gz
248   log/maillog.3.gz
621   log/maillog.8.gz
1174   log/auth
2304   log/maillog
2621   log/kernel
13245   spool/mqueue
57632   log/squid
61896   spool/smap
[root@atman /var]# du -sk{spool/mqueue,log/squid,spool/smap}/{*,.??*}\
                        | sort -n
du: spool/mqueue/.??*: No such file or directory
du: log/squid/.??*: No such file or directory
du: spool/smap/.??*: No such file or directory
0   spool/mgueue/dfEAA24999
0   spool/mgueue/qfEAA24999
0   spool/mgueue/xfEAA24999
0   spool/smap/xma000698
[… wycięto część wyniku dla zachowania zwięzłości…]
797   log/squid/access.log
799   log/squid/store.log.52.gz
813   log/squid/access.log.33.gz
815   log/squid/store.log
864   log/squid/store.log.33.gz
1054   spool/smap/xma014145
1475   spool/smap/xma002102
```

```
1531   spool/smap/xma010683
2118   spool/smap/xma029895
4154   spool/smap/xma010511
5117   spool/smap/xma011449
5117   spool/smap/xma014086
5117   spool/smap/xma016147
5117   spool/smap/xmaa09314
5117   spool/smap/xmaa14439
6840   spool/mqueueu/dfWAB03562
17373   spool/smap/xma008414
[root@atman /var]#
```

Jak widać, możemy wydać polecenie `du -sk *.??`, które spowoduje wyświetlenie wszystkich plików i katalogów należących do bieżącego katalogu włącznie z ich zakumulowanym rozmiarem w KB. Jeżeli, za pomocą potoku, skierujemy wynik do polecenia `sort -n`, będziemy mieli listę posortowaną według rozmiaru zajmowanego na dysku (flaga `-n` określa sortowanie numeryczne) W ten sposób pliki zajmujące dużo przestrzeni na dysku będą na końcu listy (lub jeżeli wykonamy `sort -nr`, będziemy mieli je na początku).

Po zidentyfikowaniu największych katalogów, możemy wykorzystać znaki wzorca do wejścia w ich głąb, aż do momentu, kiedy znajdziemy te pliki lub katalogi, które zajmują najwięcej miejsca. W naszym przypadku są to niektóre pliki w kolejkach `smap` oraz `mail` (*smap* jest programem przekazywania poczty wykorzystywanym przez niektóre zapory firewall oraz pliki dzienników *sguid*. Rozwiązaniem w tym przypadku jest sprawdzenie, czy jest potrzebne tyle plików dzienników i być może zmodyfikowanie parametrów polecenia `savelog` tak, by przechowywać mniej kopii archiwalnych.

Należy zapamiętać, że nie można zapomnieć o plikach ukrytych (tych, których nazwa rozpoczyna się kropką). Chociaż w tym przypadku nie było takich plików, w niektórych przypadkach (jak np. w przypadku katalogu macierzystego użytkownika, ukryte pliki i katalogi mogą zajmować ogromną ilość miejsca. Z tego względu zawsze jest bardzo istotne, żeby wyszukiwanie zawierało `.??` (znaki `??` dają pewność, że tylko pliki z co najmniej dwoma znakami po kropce będą uwzględniane. Pozwoli to na pominięcie plików `.` i `..` (katalog bieżący i nadrzędny).

Inną pożyteczną flagą wykorzystywaną z poleceniem `du` jest `-x`, która ogranicza poszukiwania do bieżącego systemu plików. W sytuacji pokazanej na wydruku 6.2, ta flaga byłaby konieczna, gdyby np. */var/log* był umieszczony na innej partycji (i nie chcielibyśmy jej przeszukiwać).

Przydatne może być także codzienne wykonywanie polecenia `du -k |sort` i wysyłanie wyniku działania tego polecenia do pliku. Wtedy zastosowanie polecenia `diff` do porównania dzisiejszego pliku z wczorajszym automatycznie stworzy podsumowanie. Dzięki niemu natychmiast zorientujemy się w różnicy w zajętości dysku. Możemy to zrobić poprzez wykonanie raz dziennie następującego skryptu:

```
#!/bin/bash
if[-f ~/.today]
then
     mv ~/.today] ~/.yesterday
fi
```

```
du -k / | sort -nr > ~/.today
if [-f ~/.yesterday]
then
     diff ~/.yesterday ~/.today
fi
```

Wykorzystanie polecenia find do wyszukiwania plików

W niektórych przypadkach jest konieczne znalezienie wszystkich plików należących do użytkownika. Taka sytuacja może się zdarzyć np. kiedy pracownik opuszcza firmę i jego wszystkie pliki mają być zarchiwizowane i usunięte. Może się także zdarzyć, że użytkownik przekroczył limit miejsca na dysku i chcemy znaleźć jego duże pliki.

Polecenie `find` może być nieocenione w takich przypadkach. Kilka jego opcji, które mogą być niesłychanie pożyteczne w czasie wykonywania rozliczeń dotyczących dysku pokazano w tabeli 6.4.

Tabela 6.4.
Opcje polecenia find, które są przydatne do tworzenia rozliczania wykorzystania dysku

Opcja	Działanie
`-xdev`	powoduje przeszukiwanie tylko danego systemu plików
`-atime ile_dni`	uwzględnia pliki, do których był dostęp `ile_dni` temu. parametr `ile_dni` można poprzedzić znakiem plus (+), minus (-) lub znakiem równości (=) w celu określenia, że czas dostępu jest większy, mniejszy lub dokładnie taki jak parametr
`-mtime ile_dni`	uwzględnia pliki, które były modyfikowane `ile_dni` temu. Można zastosować te same prefiksy jak w przypadku opcji `-atime`
`-user uzytkownik`	uwzględnia pliki należące do użytkownika `uzytkownik`
`-uid uid`	uwzględnia pliki należące do użytkownika o numerze ID `uid`
`- group nazwagr`	uwzględnia pliki, których grupa to `nazwagr`
`- gid gid`	uwzględnia pliki, których identyfikator ID grupy to `gid`
`-nouser`	uwzględnia pliki, które należą do nieznanego użytkownika (jego ID nie ma w pliku */etc/passwd*)
`-nogroup`	uwzględnia pliki, których grupa jest nieznana (nie ma jej w pliku */etc/group*)
`-size rozmiar`	uwzględnia pliki, których rozmiar wynosi `rozmiar`. Rozmiar to liczba, po której występuje jednostka gdzie `c` oznacza znaki (bajty), `b` oznacza bloki dysku, `k` oznacza kilobajty. Można zastosować te same prefiksy jak w przypadku opcji `-atime`

Aby na przykład znaleźć wszystkie pliki, których właścicielem jest użytkownik o nazwie *mario*, należy skorzystać z następującego polecenia:

```
#find /user mario -print
```

Jeżeli chcemy, aby pliki były wyświetlane w sposób pełny (tak jak w przypadku polecenia `ls -l`), możemy zastosować polecenie:

```
#find / -user mario -ls
```

Do wyszukania plików większych niż 1 MB należących do użytkownika *mario* w systemie plików */export/home*, do których nie było dostępu przez ostatnich 30 dni oraz w celu wyświetlenia rozmiaru pliku w bajtach, nazwy oraz czasu ostatniego dostępu zastosujemy następujące polecenie:

```
#find /export/home -xdev -user mario -size +1024K -atime +30\
                -printf '%s %p %a'
```

Wreszcie, aby usunąć wszystkie pliki należące do nieznanych użytkowników (na przykład po usunięciu kilku użytkowników z pliku */etc/passwd*), zastosujemy następujące polecenie:

```
# find /-nouser -exec rm -f \{\} \;
```

Oczywiście ta ostatnia opcja powinna być wykorzystywana bardzo ostrożnie, ponieważ kiedy pliki zostaną usunięte, nie będzie można ich odtworzyć. Lepszą opcją będzie wygenerowanie listy plików, a następnie przed ich usunięciem zastosowanie narzędzia takiego jak *tar* lub *cpio* do wykonania kopii zapasowej:

```
# find /-nouser -print | cpio -oavH crc | gzip -9 > \
/archive/nouser.cpio.gz
```

Jak możemy zauważyć, polecenie `find` to bardzo elastyczne narzędzie, które może pomóc w zarządzaniu dyskami.

Rozliczanie pracy sieci

Administratorzy sieci zazwyczaj muszą odpowiadać na pytania typu „kto wykorzystuje nasze pasmo Internetu?" albo „jak bardzo wykorzystane jest nasze połączenie WAN?" Rozliczanie pracy sieci może dostarczyć wielu informacji potrzebnych do odpowiedzi na te trudne pytania.

Poniżej przedstawiono kilka uwag:

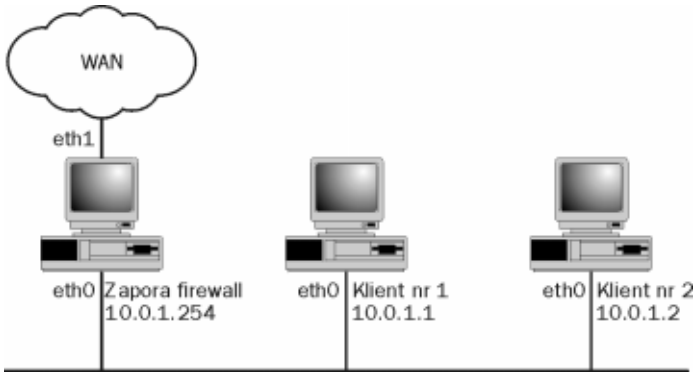
- Potrzebujemy pojedynczego punktu pomiaru. Może to być router, który łączy nas z siecią WAN (WAN przedsiębiorstwa lub Internet) albo zapora firewall. Jednakże musi to być komputer z systemem Linux. Jeżeli chcemy mierzyć ruch przechodzący przez routery innych systemów lub inne urządzenia, musimy zastosować inne narzędzia, jak np. *Multi-Router Traffic Grapher* (MRTG), który można znaleźć pod adresem *http://ee-staff.ethz.ch/~oetiker/webtools/mrtg/pub/*.
- Jeżeli korzystamy z serwera proxy jak np. *Squid* (opisanego w rozdziale 10. „Serwery informacyjne" oraz rozdziale 15 „Zapory firewall oraz serwery proxy") i chcemy monitorować wykorzystanie sieci dla każdego komputera w naszej sieci, musimy zintegrować informacje z naszego serwera z informacjami z plików dzienników w serwerze proxy.

W systemie Linux rozliczenia dotyczące sieci konfiguruje się, stosując albo program *ipfwadm* (jeżeli nasze jądro to 2.0, jak np. to, które dostarczono ze standardową dystry-

bucją 2.1), albo *ipchains* (jeżeli posiadamy jądro 2.2). Aby uzyskać więcej informacji na temat jądra w wersji 2.2 przeczytajmy rozdział 5. Więcej informacji o zaporach firewall, poleceniach `ipfwadm` oraz `ipchains` znajdziemy w rozdziale 15.

Jądro systemu Linux przechowuje wewnętrznie liczniki bajtów oraz liczniki pakietów dla każdej reguły zapory firewall (dla *ipfwadm*) lub łańcucha (dla *ipchains*). Podstawą do rozliczania pracy sieci w systemie Linux jest utworzenie reguł lub łańcuchów dla każdego łącza, które chcemy monitorować. Następnie napiszemy niewielki program, który będzie okresowo odpytywał liczniki zapór firewall (powiedzmy co 5 sekund) i zapisze te informacje do pliku. W efekcie inny program będzie mógł wykorzystać te informacje w dowolny sposób np. utworzy podsumowania, wykresy albo zapisze te informacje do bazy danych.

W następnych punktach, odkryjemy jeden ze sposobów zbierania i przetwarzania tych danych. Pamiętajmy! Nie jest to sposób jedyny. W naszych przykładach będziemy posługiwali się siecią pokazaną na rys. 6.1. Będziemy śledzić wykorzystanie pasma WAN oraz sumę bajtów transmitowanych przez każdy węzeł klienta z osobna, a także podsumowanie zbiorcze. Dla uproszczenia przyjmiemy, że albo nie wykorzystuje się maskarady adresów IP lub maskarada jest wykonywana przez host albo router (nie przez zaporę firewall).



Rysunek 6.1. *Sieć wykorzystywana w przykładach rozliczania pracy sieci*

Najważniejsze dane w tym miejscu to adresy IP węzłów klienta (*10.0.1.1* oraz *10.0.1.2*), interfejs, przez który zapora firewall jest podłączona do sieci LAN (*eth0*) oraz interfejs, przez który jest podłączona do sieci WAN (*eth1*). Zauważmy, że może to być dowolny typ interfejsu sieciowego: PPP, Token Ring, X.25 lub cokolwiek innego. Wykorzystując tę informację, możemy się zorientować, że mamy potrzebę monitorowania następujących elementów (patrząc z perspektywy zapory firewall):

- liczby bajtów pochodzących z *eth1* zmierzających do *10.0.1.1;*
- liczby bajtów wychodzących przez *eth1* pochodzących z *10.0.1.1;*
- liczby bajtów pochodzących z *eth1* i zmierzających do *10 .0.1.2;*
- liczby bajtów wychodzących przez *eth1*, pochodzących z *10.0.1.2;*
- liczby bajtów wchodzących do *eth1;*
- liczby bajtów wychodzących z *eth1.*

Po tym, jak zdobędziemy te dane, będzie można w łatwy sposób obliczyć zarówno całkowitą liczbę bajtów (poprzez proste dodawanie) oraz wykorzystanie pasma (poprzez dzielenie przez przedział odpytywania). Aby uprościć ten przykład, po prostu wygenerujemy plik */var/log/bandwidth.log*, który będzie zawierał etykietę czasową w pierwszej kolumnie oraz bieżącą wartość sześciu liczników bajtów. Te dane mogą posłużyć jako dane wejściowe do innego programu, który będzie uruchamiany rzadziej (np. co 5 minut), obliczającego wykorzystanie pasma i tworzący podsumowania. Ten program z kolei może wysłać te dane do programu graficznego, jak np. *gnuplot*.

Zbieranie danych do podsumowań poprzez ipfwadm

Funkcje zapór firewall oraz rozliczania pracy sieci w jądrze 2.0 działają dzięki regułom. Wszystkie reguły zapory firewall są po kolei porównywane z każdym z pakietów, do momentu, kiedy jest on przyjęty, odrzucony albo zabroniony.

Jądra w wersji 2.0 mają 4 rodzaje reguł: wejścia, przekazywania, wyjścia oraz rozliczania. Reguły wprowadzania, przekazywania i wyprowadzania są opisane w rozdziale 15., w którym opisano także ogólną konfigurację zapory firewall. W tej części, opiszemy tylko te reguły podsumowań oraz podzbiór opcji `ipfwadm`, które są potrzebne do wykonywania rozliczeń.

Po pierwsze musimy włączyć rozliczanie IP w jądrze. W czasie konfiguracji jądra należy wybrać grupę opcji *Network options* i uaktywnić opcję *IP accounting*. Skompilujemy jądro i ponownie uruchomimy system. Aby dowiedzieć się więcej na temat konfiguracji i kompilacji jądra przeczytaj rozdział 12.

Kiedy uruchamiamy jądro z włączoną obsługą rozliczania IP, musimy dodać reguły dotyczące rozliczania. Aby dodać reguły dla każdego z sześciu punktów danych, które chcemy śledzić, wpiszemy następujące polecenia w wierszu poleceń powłoki:

```
# ipfwadm -f -A
# ipfwadm -a -A in -D 10.0.1.1/32 -W eth1
# ipfwadm -a -A out -S 10.0.1.1/32 -W eth1
# ipfwadm -a -A in -D 10.0.1.2/32 -W eth1
# ipfwadm -a -A out -S 10.0.1.1/32 -W eth1
# ipfwadm -a -A in -W eth1
# ipfwadm -a -A out -W eth1
```

Najpierw wyczyściliśmy reguły rozliczania. Jest to bardzo ważne, ponieważ reguły `ipfwadm` są zawsze dodawane do poprzednio istniejących reguł. Gdybyśmy ich nie wyczyścili, zostały by one zdublowane i w efekcie otrzymalibyśmy mylne wyniki.

Następnie, za pomocą opcji `-a` dodaliśmy reguły jedna po drugiej. Określiliśmy, że jest to reguła rozliczania (`-A`), kierunek, który chcemy monitorować, adres źródłowy i liczbę bitów części sieciowej (`-S a.b.c.d/n`) oraz interfejs, do którego chcemy dodać tę regułę (`-W eth1`).

Wreszcie wyzerujemy liczniki za pomocą opcji `-z`. Dzięki temu będziemy mogli rozpocząć liczenie od początku. Zauważmy, ze spowoduje to wyzerowanie liczników dla wszystkich reguł rozliczania. Jądro systemu Linux nie umożliwia na zerowanie tylko określonego licznika:

```
ipfwadm -z -A
```

Aby obejrzeć dane z liczników, możemy albo zastosować opcję `-l` w `ipfwadm` razem z opcją `-e`, albo przeczytać je bezpośrednio z pseudo-pliku */proc/net/ip_acct*. Aby uzyskać więcej informacji na temat plików w katalogu */proc*, przeczytaj rozdział 7. Będziemy wykorzystywać drugą opcję, ponieważ otwieranie pliku jest szybsze niż wykonywanie program. Poniżej przedstawiono przykład wiersza pliku */proc/net/ip_acct*:

```
00000000/00000000->00000000/00000000 ppp0 00000000 2000 0 0 30
↳1639 0 0 0 0 0 0 0 0 0 0 AFF XOO
```

Jak widzimy, pola są oddzielane sekwencją spacji. Nas interesuje 8. pole (w naszym przykładzie zawierające wartość `1639`). Wiersze w pliku *ip_acct* będą w tej samej kolejności jak w poleceniu `ipfwadm`, dlatego nie będziemy musieli wykonywać dodatkowego przetwarzania.

Potrzebujemy program, który będzie odpytywał liczniki i zapisywał informacje do pliku */var/log/bandwidth.log*. Ponieważ chcemy odpytywać stosunkowo często (np. co 5 sekund), potrzebujemy program, który zbierając dane, będzie pracował w pętli nieskończonej, a nie takiego, który będzie uruchamiany co jakiś czas przez polecenie `cron`. Powodem jest fakt, iż polecenie `cron` może uruchamiać programy co pełną liczbę minut. Wydruk 6.3 przedstawia program w języku Perl ustanawiający reguły zapory firewall, a następnie odpytujący plik */proc/net/ip_acct*. Aby dowiedzieć się więcej na temat języka Perl, przeczytajmy rozdział 1.)

Wydruk 6.3.
Zbieranie danych do rozliczania za pomocą ipfwadm

```
#!usr/bin/perl
# adresy IP do monitorowania
@LAN_IPS=('10.0.1.1','10.0.1.2');
# nazwa interfejsu WAN
$WAN_IF='eth1';
#nazwa pliku dzienników
$LOGFILE='var/log/bandwidth.log';
# Ścieżka do programu ipfwadm
$IPFWADM="/sbin/ipfwadm";
# Odstęp odpytywania w sekundach
$INTERVAL = 5;
# nazwa pliku /proc do czytania
$PROCFILE="/proc/net/ip_acct";
# ustanowienie reguł zapory firewall dotyczących rozliczania
# najpierw wyczyścimy reguły
system ("$IPFWADM -f -A");
# ustanowienie wchodzących i wychodzących reguł dla każdego # adresu IP
foreach (@LAN_IPS) {
     system("$IPFWADM -a -A in -D $_ -W $WAN_IF");
     system("$IPFWADM -a -A out -S $_ -W $WAN_IF");
}
# Skonfigurowanie reguł rozliczania
```

```
system ("$IPFWADM -a -A in -W $WAN_IF");
system ("$IPFWADM -a -A out -W $WAN_IF");
# wyzerowanie wszystkich liczników zapory firewall
system ("$IPFWADM -z -A");
# wyczyszczenie pliku dzienników
open LOG, ">$LOGFILE";
close LOG;
# pętla nieskończona
while (1) {
    # oczekiwanie na dane do obliczeń
    sleep ($INTERVAL);
    # czytanie danych z pliku /proc
open PROC, "<$PROCFILE";
@data = <PROC>;
close PROC;
# zapis danych do pliku dzienniku
open LOG, ">>$LOGFILE";
# zapis etykiety czasowej
print LOG time();
foreach (@data) {
     # zapis każdego punktu danych. liczba bajtów to ósme pole
     @line=split (/ +/,$_);
     print LOG " ".$line[7];
}
print  LOG "\n";
#  zamknięcie pliku dzienników
close LOG;
}
```

Najpierw określiliśmy kilka stałych. Sprawia to, że utrzymywanie takiego programu jest znacznie łatwiejsze (np. jeżeli potrzebujemy dodać nowy host lub zapisywać dzienniki do innego pliku). Skonfigurowaliśmy reguły rozliczania za pomocą polecenie `ipfwadm`, a następnie przeszliśmy do pętli nieskończonej, odpytując co `$INTERVAL` sekund. Zauważmy, że za każdym razem zamknęliśmy, a następnie ponownie otwarliśmy plik */proc*, tak więc możemy pobierać nowe dane w każdej iteracji. Również za każdym razem zamykamy i otwieramy pliki dzienników. Z jednej strony jest to nieco kosztowne, daje jednak pewność, że plik dziennika będzie zawsze zawierał najświeższe dane oraz że żadne dane nie pozostaną w buforach systemów plików.

Zbieranie danych do rozliczeń za pomocą ipchains

Pojęcia dotyczące tworzenia rozliczeń za pomocą programu *ipchains* są podobne do tych stosowanych w przypadku *ipfwadm*. Należy dodać kilka reguł zapory firewall zbierających potrzebne dane, a następnie czytać je okresowo i zapisywać do pliku dziennika. Jednakże program *ipchains* jest o wiele bardziej elastyczny niż program *ipfwadm*. Ten fakt razem z faktem, że jądro 2.2 nie ma oddzielnych mechanizmów rozliczania powoduje, że niektóre szczegóły są zmienione.

Po pierwsze parę informacji tytułem wstępu. Sposób, w jaki działa program *ipchains*, to tworzenie łańcuchów reguł. Każda reguła w każdym łańcuchu jest stosowana do dowolnego pakietu, który przez nią przechodzi. Jeżeli pakiet jest zgodny z dowolną regułą w łańcuchu, będzie przyjęty, odrzucony lub zabroniony w zależności od stosowanej polityki.

Istnieją trzy predefiniowane łańcuchy: `input`, `output` oraz `forward`. Są one stosowane odpowiednio do pakietów przychodzących do hosta, wychodzących z hosta i przekazywanych. Łańcuch `forward` jest stosowany po łańcuchu `input` i przed łańcuchem `output`. Istnieją także łańcuchy definiowane przez użytkownika. Aby je wykorzystać, najpierw należy je zdefiniować. Następnie do łańcuchów predefiniowanych należy dodać warunki określające skok do łańcuchów zdefiniowanych przez użytkownika. Są to ogólne zasady rozliczania za pomocą *ipchains*.

Wszystkie pakiety, które przechodzą przez każdy łańcuch, są liczone w zależności od łańcucha, z którego pochodzą. W związku z tym, jeżeli zdefiniujemy łańcuch o nazwie `acct` i dodamy go do obu łańcuchów `input` i `output`, jądro będzie liczyć każdy z nich niezależnie.

Najpierw musimy włączyć obsługę zapory firewall w jądrze. W czasie konfigurowania jądra, należy w grupie opcji *Networking options*, włączyć opcję *Network Firewalls* oraz *IP: firewalling*. Następnie należy skompilować jądro i ponownie uruchomić system. Kiedy uruchamiamy jądro z włączona obsługą zapory firewall, możemy dodać reguły *ipchains* wykorzystywane w celach rozliczania. W naszym przykładzie powinniśmy wydać następujące polecenia:

```
# ipchains -F input
# ipchains -F output
# ipchains -F acct
# ipchains -X acct
# ipchains -N acct
# ipchains -A input -d 10.0.1.1/32 -i eth1 -j acct
# ipchains -A output -s 10.1.0.1/32 -i eth1 -j acct
# ipchains -A input -d 10.0.1.2/32 -i eth1 -j acct
# ipchains -A output -s 10.0.1.2/32 -i eth1 -j acct
# ipchains -A input -i eth1 -j acct
# ipchains -A output -i eth1 -j acct
# ipchains -Z acct
```

Najpierw wyczyściliśmy łańcuchy `input` oraz `output`. Jest to konieczne z tego względu, że odnoszą się one do łańcucha `acct` i nie byłoby możliwe ich wyczyszczenie lub usunięcie. Jeżeli pracujemy na komputerze z zaporą firewall, będziemy musieli indywidualnie usunąć (za pomocą opcji `-D`) każdą regułę, która odnosi się do łańcucha `acct`. Wyczyszczenie spowodowałoby usunięcie także reguł zapory firewall. Po wykonaniu tej czynności musimy wyczyścić i usunąć łańcuch `acct`, aby się upewnić, że nie zostało żadnych danych z poprzedniego uruchomienia.

Następnie, należy utworzyć każdą z potrzebnych reguł, określając, do którego łańcucha należy ją dodać (`input` lub `output`), ustalić odpowiadające im adresy źródłowe i docelowe (opcje `-s` lub `-d`), żądany interfejs (`-i`) oraz to, że chcemy wykonać skok do łańcucha `acct` (`-j`). Łańcuch *acct* sam w sobie nie ma żadnych reguł, ponieważ jego jedynym celem jest zliczanie bajtów wchodzących i wychodzących, a ta funkcja jest wykonywana automatycznie przez jądro.

Wreszcie wyzerowaliśmy liczniki łańcucha `acct` (`-Z`), po to, by móc zliczać od początku. W poleceniu `ipchains`, opcja `-Z` może wyzerować liczniki dla pojedynczego łańcucha, tak więc inne łańcuchy nie będą naruszone.

Aby przeczytać dane, możemy albo zastosować opcję `-L` lub `-v` w *ipchains*, albo czytać bezpośrednio z pseudo-pliku */proc/net/ip_fwchains* (aby dowiedzieć się więcej o systemie plików */proc*, przeczytaj rozdział 15.). Czytanie bezpośrednio z katalogu */proc* jest bardziej wydajne niż uruchamianie za każdym razem nowego programu. Oprócz tego, wynik polecenia `ipchains` jest przygotowany do odczytywania przez ludzi, co oznacza, że trudniej go przetworzyć niż zawartość pliku *ip_fwchains*. W związku z tym skorzystamy z drugiego sposobu.

Poniżej pokazano przykład pliku */proc/net/ip_fwchains*:

```
input 00000000/00000000->0A000101/FFFFFFFF eth1 0 0 0 0
↳1725       0              1704500        0-65535 0-65535
↳AFF X00 00000000  0 0  acct
input 00000000/00000000->0A000102/FFFFFFFF eth1 0 0 0 0
↳2843       0              4382420        0-65535 0-65535
↳AFF X00 00000000  0 0  acct
input 00000000/00000000->00000000/00000000 eth1 0 0 0 0
↳38506      0              23048401       0-65535 0-65535
↳AFF X00 00000000  0 0  acct
output 0A000101/FFFFFFFF->00000000/00000000 eth1 0 0 0 0
↳520        0              83000          0-65535 0-65535
↳AFF X00 00000000  0 0  acct
output 0A000102/FFFFFFFF->00000000/00000000 eth1 0 0 0 0
↳2340       0              2843020        0-65535 0-65535
↳AFF X00 00000000  0 0  acct
```

Każdy rekord składa się z kilku pól oddzielonych spacjami. Niektóre z nich są nam potrzebne. Pierwsze pole określa, czy jest to łańcuch wejściowy czy wyjściowy. Jak można zauważyć, *ipchains* porządkuje reguły według łańcuchów. Z tego względu przed wyprowadzeniem danych, będzie potrzebne ich przetworzenie. Jednakże w obrębie łańcucha rekordy są w tym samym porządku, w jakim zostały utworzone.

Dziesiąte pole określa liczbę bajtów, co stanowi właściwe wymagane przez nas dane, a ostatnie (osiemnaste) określa łańcuch. W naszym przypadku, interesują nas wyłącznie dane z łańcucha `acct`, dlatego musimy go wybrać.

Wydruk 6.4 przedstawia program w języku Perl pobierający dane z pliku *ip_fwchains* i zapisujący je do pliku */var/log/bandwidth.log*.

Wydruk 6.4.
Zbieranie danych do rozliczeń za pomocą ipchains

```
#!usr/bin/perl

# adresy IP do monitorowania
@LAN_IPS=('10.0.1.1','10.0.1.2');
# nazwa interfejsu WAN
$WAN_IF='eth1';
# nazwa pliku dzienników
$LOGFILE='var/log/bandwidth.log';
# Ścieżka do programu ipchains
$IPCHAINS="/sbin/ipchains";
# Odstęp odpytywania w sekundach
$INTERVAL =5;
```

```
# Nazwa pliku /proc do czytania
$PROCFILE="/proc/net/ip_fwchains";
# Nazwa łańcucha dotyczącego zestawień
$CHAIN="act";
# Numer pola etykiety łańcuchów input/output
$IO_FIELD=0
# Numer pola licznika bajtów
$BYTES_FIELD=9
# Numer pola naszego łańcucha
$CHAIN FIELD=17
# znaczniki dla pól IO
$IO_INPUT="input"
$IO_OUTPUT="output"
# Skonfigurowanie reguły zestawień
# Najpierw wyczyścimy łańcuchy
# OSTRZEŻENIE: nie należy czyścić reguł input oraz output
# jeżeli w systemie jest zainstalowana działająca zapora
# firewall !
system ("$IPCHAINS -F input");
system ("$IPCHAINS -F output");
system ("$IPCHAINS -F $CHAIN");

# Usuniemy i utworzymy na nowo łańcuch zestawień
system ("$IPCHAINS -X $CHAIN");
system ("$IPCHAINS -N $CHAIN");

# Skonfigurujemy reguły przychodzące i wychodzące dla
# każdego adresu IP
foreach (@LAN_IPS) {
    system("$IPCHAINS -A input -d $_ -i $WAN_IF -j $CHAIN");
    system("$IPCHAINS -A output -s $_ -i $WAN_IF -j $CHAIN");
}
# skonfigurowanie reguł sumowania
system ("$IPCHAINS -A input -i $WAN_IF -j $CHAIN");
system ("$IPCHAINS -A output -i $WAN_IF -j $CHAIN");

# Wyzerowanie wszystkich liczników zapory firewall
system ("$IPCHAINS -Z $CHAIN");

# Wyczyszczenie pliku dzienników
open LOG, ">$LOGFILE";
close LOG;

# Pętla nieskończona
while (1) {
  # Oczekiwanie na dane do obliczeń
  sleep ($INTERVAL);

  # czytanie danych z pliku /proc
  open PROC, "<$PROCFILE";
  @data = <PROC>;
  close PROC;

  # Utworzenie tablic dla danych
  @input=();
  @output=();
```

```
  # Zapis danych do pliku dzienniku
  foreach (@data) {
    chop;
    s/^+//;
    @line=split (/ +/,$_);
    #Sprawdzenie czy należy do naszego łańcucha
    if ($line[$CHAIN_FIELD] eq $CHAIN) {
      # Dodanie wartości odnośnie liczby bajtów do
      # odpowiedniejtablicy
      push (@input,  $line[$BYTES_FIELD])
              if ($line[$IO_FIELD] eq $IO_INPUT);
      push (@output, $line[$BYTES_FIELD])
              if ($line[$IO_FIELD] eq $IO_OUTPUT);
    }
  }
# Wpisanie wyniku do pliku dzienników
open LOG, ">>$LOGFILE";
print LOG time();
for ($i = 0; $i < @input; $i++) {
   print LOG " $input[$i] $output[$i]";
}
print  LOG "\n";
close LOG;
          }
```

Jak można zauważyć, wykonujemy te same czynności, które wykonaliśmy, wykorzystując *ipfwadm*. Jedyna różnica polega na tym, że filtrujemy dane na podstawie nazwy łańcucha (ponieważ wszystkie łańcuchy znajdują się w tym samym pliku) oraz wykorzystujemy parę tablic w celu dokonania reorganizacji danych przed ich zapisaniem do pliku dziennika.

Program *ipchains* daje nam większe możliwości niż *ipfwadm*. Możemy z łatwością rozszerzyć monitorowanie ruchu o inne charakterystyki poprzez zwykłe dodanie dodatkowych zmiennych `$IO_XXX`. Za pomocą *ipfwadm* musielibyśmy dodawać reguły, określając każdy rodzaj pakietu, który chcielibyśmy monitorować. W przypadku *ipchains* po prostu dodajemy polecenie z opcją `-j` w celu wykonania skoku do naszego łańcucha `acct`, gdziekolwiek tylko chcemy zbierać dane do rozliczeń.

Rozliczenia dotyczące procesów oraz analiza wydajności

W rozliczeniach dotyczących procesów, skupiamy się na ilości zasobów systemu (pamięci oraz procesora) wykorzystywanych przez proces. Zestawienia dotyczące procesów są przydatne w przypadku wykonywania funkcji *chargeback*, analizowania wydajności serwera oraz w czasie wykonywania planowania możliwości systemu. System Linux zawiera kilka narzędzi do wykonywania rozliczeń dotyczących procesów. W tym miejscu zostaną opisane narzędzia *ps*, *top*, *vmstat* oraz część narzędzi należących do pakietu *GNU Accounting Utilities*.

Wykorzystanie ps do zestawień dotyczących procesów

System Linux obejmuje kilka narzędzi, które są pomocne w wykonywaniu rozliczeń dotyczących procesów. Pierwszym narzędziem jest polecenie `ps`, które powoduje wyświetlenie listy uruchomionych procesów w systemie w danym momencie.

Najbardziej przydatne opcje polecenia `ps` w celach rozliczania procesów przedstawiono w tabeli 6.5 . Tabela 6.6 zawiera opis najważniejszych danych polecenia `ps`.

Tabela 6.5.
Opcje polecenie ps, które są przydatne do rozliczania procesów

Opcja	Działanie
`m`	Informacje o pamięci. ważne pola to `SIZE`, `RSS` oraz `SHRD`
`u`	Informacje wyjściowe użytkownika. Ważne pola to `%CPU`, `%MEM`, `SIZE`, `RSS` oraz `TIME`
`x`	Wyświetla procesy, które nie są sterowane przez terminal (demony)
`a`	Wyświetla procesy należące do innych użytkowników

Tabela 6.6.
Ważne pola polecenia ps

Pole	Działanie
`USER`	nazwa użytkownika, który jest właścicielem procesu
`PID`	numer ID procesu
`COMMAND`	polecenie wykonywane przez proces
`SIZE`	całkowity, wirtualny rozmiar procesu w KB, włącznie z kodem, danymi oraz stosem
`RSS`	rozmiar pamięci RAM w KB aktualnie wykorzystywanej przez proces (razem z pamięcią dzieloną)
`SHARE`	rozmiar pamięci RAM dzielonej z innymi procesami w KB
`%CPU`	procentowy udział czasu procesora wykorzystywany przez proces
`%MEM`	procent pamięci systemowej RAM wykorzystywany przez proces
`TIME`	sumaryczny czas procesora wykorzystany przez proces przedstawiony w postaci `minuty:sekundy`

Wykorzystanie narzędzia top w celu monitorowania procesów w czasie rzeczywistym

Polecenie `top` jest interaktywną wersją polecenia `ps` pracująca w czasie rzeczywistym. Pokazuje ono te same podstawowe statystyki jak polecenie `ps`, uaktualniając wyświetlanie co 5 sekund. Pozwala ono na sortowanie wyników wg `%MEM`, `%CPU` oraz `TIME`, a także pozwala na wyświetlanie tylko procesów określonego użytkownika albo procesów, które nie znajdują się w stanie oczekiwania. Pozwala także na niszczenie zadań.

Narzędzie *q* jest przydatne w codziennym monitorowaniu hosta, w celu wykrycia procesów niekontrolowanych lub procesów, które zużywają zbyt dużo pamięci RAM. Kolumny wyświetlane przez to polecenie są takie same jak wyświetlane przez polecenie ps.

Korzystanie z vmstat w celu monitorowania statystyk dotyczących pamięci wirtualnej

Polecenie `q` jest prawdopodobnie najważniejszym w mechanizmie tworzenia rozliczania procesów oraz zarządzania wydajnością w systemie UNIX. W czasie kiedy pracuje, okresowo próbkuje stan systemu i wyświetla w wyniku jeden wiersz. Zawiera on średnią wartość statystyki obciążenia systemu podczas tego przedziału. Program *vmstat* jest zazwyczaj wywoływany z jednym argumentem określającym liczbę sekund pomiędzy próbkami. Przydatne wartości mieszczą się w zakresie od 3 do 30 sekund, zależnie od szukanej dokładności.

Możemy także podać drugi argument liczbowy, który będzie określał liczbę próbek, jakie należy wykonać. Jest to przydatne w celu szybkiego pobierania informacji o stanie systemu. Inną przydatną opcja jest `-n`. Zwykle wynik *vmstat* będzie co 20 wierszy wyświetlał nagłówki. Jeżeli, w celu dalszego przetwarzania, kierujemy wynik działania do pliku dziennika, możemy dodać opcję `-n`, aby nagłówki znalazły się tylko na początku.

Ważną rzeczą, jaką należy zauważyć w przypadku korzystania z *vmstat*, jest fakt, że powinniśmy zawsze ignorować pierwszy wiersz wyniku. Ten wiersz określa średnią od momentu załadowania systemu i zazwyczaj nie ma znaczenia. Użyteczne wartości zaczynają się od drugiego wiersza wyniku.

Jeżeli używamy *vmstat* do rejestrowania przez długi czas, zapamiętajmy, że jako próbkę generuje on wiersz o długości 72 znaków. Na przykład *vmstat* z 5 sekundowym przedziałem próbkowania generuje więcej niż 1 MB danych dziennie.

Mimo że *vmstat* zajmuje się statystykami dotyczącymi pamięci wirtualnej, jest on przydatny do monitorowania aktywności dysku i procesora naszego komputera. Okresowe uruchamianie programu *vmstat,* pozwoli na zapoznanie się z liczbami, jakie wyświetla w różnych warunkach obciążenia systemu. Polecenia `vmstat` można także wykorzystywać do wykrywania, czy problem wydajności wynika ze względu na problemy dysku czy obciążenie procesora.

Narzędzia GNU rozliczania wykorzystania zasobów

Narzędzia GNU rozliczania zasobów są zaprojektowane do dostarczenia administratorowi systemu informacji na temat wykorzystania systemu: połączeń, uruchamianych programów oraz wykorzystania zasobów systemowych. Można je odnaleźć w pakiecie *acct*.

Istnieje kilka narzędzi, które mogą być zastosowane zarówno do rozliczania procesów, jak i użytkowników. W tej części skoncentrujemy się na tych pierwszych. Narzędzia wykorzystywane w celu rozliczania użytkowników zostaną przedstawione później.

Tabela 6.7.
Pola vmstat

Pole	Działanie
`procs: r`	Liczba procesów czekająca na uruchomienie. Zwykle r nie powinno być większe niż liczba procesorów w systemie. Jeżeli r ma ciągle wysoką wartość, w systemie warto by zainstalować dodatkowe procesory (szczególnie jeżeli korzystamy z jądra 2.2)
`procs: w`	Liczba procesów w stanie „uśpienia", czekających na jakieś zdarzenie. Zwykle czekają one na zakończenie żądania wejścia-wyjścia
`procs: b`	Liczba innych procesów czekających na uruchomienie znajdujących się w pamięci wirtualnej
`mem: swpd`	Ilość wykorzystywanego obszaru wymiany (w KB)
`mem: free`	Ilość wolnej pamięci RAM (w KB)
`mem: buf, cache`	Ilość wykorzystywanej pamięci, odpowiednio przez bufory dyskowe oraz jako dyskowa pamięć podręczna w KB. Ta pamięć jest zarządzana dynamicznie przez jądro systemu Linux, tak więc większość z tych wartości może być włączona do pamięci wolnej w przypadku braku pamięci RAM
`-swap: si, so`	Liczba KB pamięci wymieniana do pamięci wirtualnej i z powrotem na sekundę. Jeżeli te wartości są stosunkowo wysokie, host potrzebuje więcej pamięci RAM
`-io: bi, bo`	Liczba bloków na sekundę, odpowiednio otrzymanych z i wysłanych do urządzenia blokowego
`-system: in`	Liczba otrzymanych przerwań na sekundę, włącznie z przerwaniami zegara
`system: cs`	Liczba przełączeń zawartości na sekundę. Przełączenie zawartości ma miejsce, kiedy procesor przełącza się od jednego procesu do innego
`cpu: sy, us, id`	Procentowy udział czasu procesora odpowiednio w systemie, w procesie użytkownika oraz w stanie oczekiwania. Jeżeli procentowy udział stanu oczekiwania jest ciągle wyższy niż 70%, procesor hosta jest przeciążony

Narzędzia rozliczania GNU zapisują swoje dane w pliku */var/account/pacct*. Rozliczanie rozpoczyna się uruchomieniem polecenia `accton`, które jest wywoływane przez plik */etc/init.d/acct*. Istnieją dwa polecenia wydobywania danych do rozliczania procesów: `lastcomm` oraz `sa`.

Wykorzystanie polecenia lastcomm w celu wyświetlania listy programów, które były wykonywane

Polecenie `lastcomm` powoduje wyświetlenie informacji na temat wszystkich programów, które były uruchamiane (i zakończyły swoje działanie) od założenia pliku */var/account/pacct*. Domyślnie ten plik jest rotowany codziennie poprzez plik */etc/cron.daily/acct*. Jeżeli chcemy zebrać dane za kilka dni, możemy zastosować flagę `-f` w celu czytania starszych plików (po ich uprzedniej dekompresji) lub zmodyfikować skrypt */etc/cron.daily/acct*. Na przykład, jeżeli chcemy dokonywać rotacji pliku co tydzień, powinniśmy przesunąć skrypt do katalogu */etc/cron.weekly* i prawdopodobnie zmienić parametr `-c` polecenia `savelog`, żeby zapisywało mniej niż siedem plików archiwalnych.

Polecenie `lastcomm` wyświetla dane w następujących polach:

- uruchomione polecenie;
- flagi `-S` oznacza, że polecenie było wykonywane przez superużytkownika. `F` oznacza, że program wykonał rozwidlenie (ang. *fork*) (rozdzielenie na dwa procesy) i nie wykonał żadnego polecenia `exec` po tym fakcie (co spowodowałoby rozdzielenie rozliczania na dwa procesy). `D` oznacza, że program zakończył działanie w nadzwyczajnych warunkach, generujac plik *core*. `X` oznacza, że program został zatrzymany przez sygnał `SIGTERM` (zwykle generowane przez *Ctrl+C*);
- użytkownik, który uruchamiał polecenie;
- terminal *tty*, z którego było uruchomione polecenie;
- ilość czasu procesora wykorzystana przez proces;
- czas zakończenia procesu.

Te informacje są najbardziej przydatne w przypadku wykonywania funkcji *chargeback* za wykorzystane zasoby systemowe, ponieważ można na ich podstawie stwierdzić, kto wykonywał dane polecenia, oraz jak dużo czasu procesora zajmowało każde polecenie. Do polecenia `lastcomm` można dodać parametry, które będą działały jak klucze wyszukiwania, wyświetlając tylko te rekordy, które są zgodne z dowolnymi parametrami w dowolnych polach. Na przykład `lastcomm tty1` wyświetla tylko te programy, które były uruchomione z konsoli, natomiast `lastcomm tty1 mario` wyświetla tylko programy, dla których jedno z pól jest zgodne z *tty1* lub *mario*. Jeżeli chcemy, aby było zgodne określone pole, możemy wykorzystać opcje `--user`, `--command` oraz `--tty`. Tak więc możemy zastosować polecenie: `lastcomm --tty tty1 --user mario`.

Zastosowanie polecenia sa do tworzenia podsumowań danych z rozliczeń

Polecenie `sa` jest wykorzystywane do wykonywania podsumowań informacji zawartych w pliku */var/account/pacct*. Dodatkowo może ono zapisywać sumaryczne informacje w pliku */var/account/savacct* oraz */var/account/usracct*.

Polecenie `sa` może wyświetlać pola pokazane w tabeli 6.8.

Tabela 6.8.
Pola polecenia sa

Pole	Działanie
`cp`	całkowity czas CPU, suma czasu systemowego i użytkownika
`re`	całkowity rzeczywisty czas uruchomienia w sekundach
`re/cp`	współczynnik pomiędzy czasem rzeczywistym a czasem CPU
`u`	czas CPU użytkownika w sekundach
`s`	systemowy czas CPU w sekundach

Nazwy pól nie są wyświetlane w nagłówkach kolumn. Zamiast tego, są one dodane do każdej z wartości. Zatem możemy zobaczyć `a 25.7cpu`, co oznacza 25,7 sekund procesora.

Polecenie `sa` może być wywoływane z kilkoma opcjami. Najbardziej przydatne do rozliczania procesów oraz użytkowników przedstawiono w tabeli 6.9.

Tabela 6.9.
Opcje polecenia sa

Opcja	Działanie
`-a`	domyślnie polecenie `sa` grupuje wszystkie polecenia zawierające niedrukowalne znaki w ich nazwach oraz te, które były uruchomione tylko raz, w pojedynczym zapisie oznaczonym `***other`. Opcja `-a` powoduje, że są wyświetlane wszystkie polecenia
`-c`	wyświetlenie procentowego współczynnika całkowitego czasu dla każdej z wartości użytkownik, system oraz czas rzeczywisty
`-l`	oddziela czas użytkownika od czasu systemowego. Normalnie, są one dodane i wyświetlone jako czas CPU
`-m`	wyświetla liczbę procesów oraz liczbę minut procesora dla każdego użytkownika
`-s`	podsumowuje plik */var/account/pacct* do plików *savacct* oraz *usracct*. Wymaga uruchomienia przez użytkownika *root*
`-t`	wyświetla współczynnik czasu rzeczywistego w stosunku do całkowitego czasu procesora dla każdego procesu. Jeżeli wartość jest za mała (czas procesora wynosi zero), wyświetla `**ignore**` w odpowiednim polu
`-u`	wyświetla tylko identyfikator ID użytkownika oraz nazwę polecenia dla każdego zapisu w pliku *pacct*. Jeżeli określono opcję `-u`, wszystkie inne opcje są ignorowane

Rozliczanie użytkowników

Podobnie jak rozliczanie procesów zajmuje się ilością zasobów systemowych, które są konsumowane przez proces, celem rozliczania użytkowników jest śledzenie ilości zasobów systemowych zużytych przez każdego z użytkowników. W związku z tym to zagadnienie jest blisko związane z rozliczaniem procesów i niektóre programy wykorzystywane do rozliczania procesów mogą dostarczać danych do rozliczania użytkowników (np. `lastcomm` oraz `sa`). Jednakże istnieje kilka narzędzi, których jedynym celem jest rozliczanie użytkowników, szczególnie tych, którzy logują się bezpośrednio na serwerze (w odróżnienie od tych, którzy korzystają z niego jako serwera plików lub serwer bazy danych). Te polecenia to `last` oraz `ac`.

Powyższe polecenia wyświetlają dane zapisane w pliku */var/log/wtmp*. Dane do tego pliku są dostarczane przez procesy *login* oraz *init*, dlatego w obciążonych systemach może on rozrastać się do bardzo dużych rozmiarów. Skrypt */etc/cron.monthly/acct* powoduje jego rotację po podsumowaniu jego zawartości w pliku */var/log/wtmp.report*. Jeżeli plik */var/log/wtmp* nie istnieje, rejestrowanie nie jest możliwe.

Zastosowanie polecenia last do przeglądania danych o logowaniu oraz kończeniu pracy

Polecenie `last` wyświetla rekordy z pliku *wtmp*. Każdy rekord składa się z następujących danych:

- nazwa użytkownika;
- terminal *tty*, w którym miało miejsce logowanie;
- host, z którego nastąpiło logowanie lub konsola, jeżeli było to logowanie lokalne;
- etykieta czasowa logowania, a następnie znak myślnika (`-`) oraz czas zakończenia pracy (ang. *logout*) lub tekst `still logged`, jeżeli użytkownik jeszcze nie zakończył pracy;
- całkowity czas sesji podany w nawiasach oraz w formie `godziny : minuty`.

Specjalny użytkownik *reboot* oznacza czasy ładowania systemu. Nie jest jednak możliwe uzyskanie informacji o wyłączeniu systemu.

Polecenie `last` może być wywołane z parametrem `-x`, który wyświetla zmiany trybów działania. Jest to przydatne do obserwacji planowych wyłączeń systemu (w odróżnieniu od awarii systemu), ponieważ są one zapisane jako zmiany do trybu działania 0 (dla zamknięć systemu) oraz 6 (dla ponownych uruchomień).

Zastosowanie polecenia ac do przeglądania podsumowań czasów połączenia

Bez podanych parametrów polecenie `ac` po prostu wyświetla listę całkowitego czasu połączenia w godzinach dla wszystkich użytkowników systemu. Polecenie `ac` przyjmuje kilka opcji:. `ac -d` wyświetla podsumowania dzienne zamiast pojedynczych podsumowań, `ac -p` wyświetla podsumowania dla każdego użytkownika, `ac -pd` podsumowuje wg dnia oraz użytkownika.

Możemy także określić listę użytkowników do wyświetlenia, co spowoduje, że podsumowania czasów będą się odnosiły tylko dla tych użytkowników. Na przykład `ac mario root` wyświetla całkowitą liczbę godzin, jaką użytkownicy *root* oraz *mario* byli zalogowani, natomiast `ac-p mario root` wyświetla również osobne podsumowania dla każdego użytkownika.

Jak można zauważyć, wiele narzędzi pozwala na monitorowanie zachowania systemu Linux. W istocie, jest ich tak wiele, że jak pokazano w następnej części, czasem jest wygodne korzystanie z narzędzia, które śledzi wszystko.

Zautomatyzowane narzędzia monitorowania

Pliki dzienników nie są dobrym rozwiązaniem, jeżeli nie są okresowo monitorowane. Jednakże, zwykle rosną one tak szybko, że nie jest możliwe ich ciągłe monitorowanie, a co za tym idzie wykrywanie możliwych błędów. Oprócz tego wiele problemów (na przykład zapełnienie dysku) lepiej wykrywać, zanim one nastąpią (np. dysk jest zapełniony w 95%).

W tej sytuacji możemy napisać programy, które automatycznie monitorują pliki dzienników oraz statystyki i w określonych warunkach wykonują pewne działania. Na przykład, jeżeli system plików */tmp* jest zapełniony, skrypt może usunąć niektóre tymczasowe dane, o których wiemy, że nie są potrzebne. Jeżeli w systemie wyczerpuje się pamięć, możemy automatycznie dodać plik wymiany. Jeżeli serwer bazy danych wyłącza się, możemy wysłać pocztę lub wywołać za pomocą systemu *pager* administratora systemu.

Istnieje kilka systemów do automatyzacji zbierania danych z pików dzienników. Doskonałym narzędziem w tej kategorii jest PIKT (*Problem Information/Killer Tool*), który może być pobrany pod adresem *http://opikt.uchicago.edu.pikt*. Narzędzie PIKT jest kompletnym językiem skryptów i wyjaśnienie, jak należy z niego korzystać, wymagałoby co najmniej jednego rozdziału. Jednakże dla małej sieci lub pojedynczego systemu taki system może być zbyt rozbudowany. Wtedy lepiej, jeżeli administrator systemu napisze swoje własne skrypty.

Przypuśćmy, na przykład, że chcemy otrzymać e-mail, jeżeli system plików *root* na serwerze jest pełny. Wydruk 6.5 przedstawia skrypt w języku Perl, który może to wykonać.

Wydruk 6.5.
Monitorowanie pojemności systemu plików

```
# !/usr/bin/perl
#Polecenie monitorowania
$ CMD ="bin/df / |"
# numer wiersza do monitorowania (pierwszy wiersz ma numer 0)
$LINE_NO=1;
# wyrażenie oznaczające separator pól
$FIELD_SEP="[ %]+";
# wartość graniczna, wysłanie poczty, jeżeli większa
$THRESHOLD=75
# odstęp pomiędzy próbkami (w sekundach)
$INTERVAL=300;
# osoba, do której ma być wysłana poczta
$ADMIN="sysadmin\@company.com";

while(1) {
  # pobranie wyniku polecenia
  open DF, "$CMD";
  @lines=<DF>;
  close DF;

  # podział interesującego nas wiersza
  @fields= split ($FIELD_SEP, $lines[$LINE_NO]);
```

```
    #sprawdzenie wartości
    if ($fields[$FIELD] > $THRESHOLD) {
    open MAIL, "| mail -s ' UWAGA: przekroczony limit !' $ADMIN";
    print MAIL "Wartość graniczna w monitorowanym systemie \n";
    print MAIL "przekroczona.\n";
    print MAIL "Wiersz $LINE_NO, pole $FIELD ma wartość ↘$fields[$FIELD], większą niż
    ↳$THRESHOLD \n\n";
    print MAIL "To jest wynik działania polecenia "$CMD':\n\n";
    print MAIL @lines;
    close MAIL;
  }

  # Oczekiwanie na następne odpytywanie
  sleep ($INTERVAL);
}
```

Program działa w pętli, uruchamiając polecenie `df /` co 5 minut i sprawdzając wynik działania. Skonfigurowanie programu ze zmiennymi na początku pozwala na jego bardzo łatwe wykorzystanie do sprawdzania wyniku działania innych programów (np. polecenia `ping` do innego serwera lub polecenia `free` w celu sprawdzania obszaru wymiany). Można go łatwo przystosować do sprawdzania pliku (np. */var/log/meminfo*, gdzie także można uzyskać informacje o dostępnym obszarze wymiany).

Monitorowanie ciągle rosnących plików dzienników może być nawet łatwiejsze. Sztuczka podana tutaj polega na otwarciu pliku do odczytu, zastosowanie funkcji `seek` w celu osiągnięcia końca pliku, a następnie czytanie z tego pliku. Wydruk 6.6 przedstawia program, który wykonuje polecenie `sendpage` (fikcyjne polecenie, które powinno wysyłać informacje do systemu *pager*), kiedy tylko użytkownik *root* loguje się do serwera. Może to być stosowane do monitorowania bezpieczeństwa ośrodka.

Wydruk 6.6.
Monitorowanie plików dzienników

```
#!/usr/bin/perl
#Polecenie do monitorowania
$FILE=tail -f /var/log/auth.log |;
# wyrażenie do sprawdzenia
$REGEXP="login\\{.*\\]: ROOT LOGIN";
#skrypt do wykonania (wiersz syslog dodany do polecenia)
$CMD="/usr/local/bin/sendpage 5835544 ";

#Otwarcie pliku do monitorowania
open LOGFILE, "$FILE";

while (<LOGFILE>) {
  chop;
  tr/'`//d;
  system ($CMD.$_) if (/$REGEXP/;
}
```

Ten skrypt również można przystosować do indywidualnych potrzeb.