Red Hat Enterprise Linux 7
システム管理者のガイド

Red Hat Enterprise Linux 7 の導入、設定、管理

Jaromír Hradílek
Stephen Wadeley
Miroslav Svoboda
Eliška Slobodová
David O'Brien
Douglas Silas
Tomáš Čapek
Petr Bokoč
Eva Kopalová
Michael Hideo
Martin Prpič
Petr Kovář
Peter Ondrejka
John Ha
Don Domingo

Red Hat Enterprise Linux 7 システム管理者のガイド

Red Hat Enterprise Linux 7 の導入、設定、管理

Jaromír Hradílek
Red Hat Red Hat Engineering Content Services
jhradilek@redhat.com

Douglas Silas
Red Hat Red Hat Engineering Content Services
silas@redhat.com

Martin Prpič
Red Hat Red Hat Engineering Content Services
mprpic@redhat.com

Stephen Wadeley
Red Hat Red Hat Engineering Content Services
swadeley@redhat.com

Tomáš Čapek
Red Hat Red Hat Engineering Content Services
tcapek@redhat.com

Petr Kovář
Red Hat Red Hat Engineering Content Services
pkovar@redhat.com

Miroslav Svoboda
Red Hat Red Hat Engineering Content Services
msvoboda@redhat.com

Petr Bokoč
Red Hat Red Hat Engineering Content Services
pbokoc@redhat.com

Peter Ondrejka
Red Hat Red Hat Engineering Content Services
pondrejk@redhat.com

Eliška Slobodová
Red Hat Red Hat Engineering Content Services
eslobodo@redhat.com

Eva Kopalová
Red Hat Red Hat Engineering Content Services

John Ha
Red Hat Red Hat Engineering Content Services

David O'Brien
Red Hat Red Hat Engineering Content Services

Michael Hideo

Red Hat Red Hat Engineering Content Services

Don Domingo
Red Hat Red Hat Engineering Content Services

法律上の通知

概要

システム管理者のガイドは、Red Hat Enterprise Linux 7 の導入、設定、管理の関連情報について説明しています。本書は、システムに関する基本的な理解があるシステム管理者を対象としています。

目次

パート I. システムの基本設定

ここでは、キーボード設定、日付と時刻の設定、ユーザーとグループの管理、権限の取得など基本的なシステム管理タスクを取り上げます。

第1章 システムロケールおよびキーボード設定

システムロケールでは、システムサービスおよびユーザーインターフェイスの言語設定を指定します。**キーボードレイアウト**の設定では、テキストコンソールおよびグラフィカルユーザーインターフェイスで使用するレイアウトを管理します。

これらの設定は、**/etc/locale.conf** 設定ファイルを修正するか**localectl** ユーティリティーを使用して行います。または、グラフィカルユーザーインターフェイスを使ってこのタスクを実行することもできます。この方法についての説明は、Red Hat Enterprise Linux 7 インストールガイド を参照してください。

1.1. システムロケールの設定

システム全体にわたるロケール設定は **/etc/locale.conf** ファイルに保存され、**systemd** デーモンが初期ブート時に読み込みます。**/etc/locale.conf** で設定されたロケール設定は、個別のプログラムやユーザーが上書きしない限り、すべてのサービスやユーザーに継承されます。

/etc/locale.conf の基本的なファイル形式は、改行で区切られた変数割り当ての一覧です。たとえば、ロケールがドイツ語でメッセージが英語の場合、**/etc/locale.conf** は以下のようになります。

```
LANG=de_DE.UTF-8
LC_MESSAGES=C
```

ここでは LC_MESSAGES オプションが標準エラー出力に書き出される診断メッセージ用ロケールを決定します。**/etc/locale.conf** でさらにロケール設定を指定するには、いくつかのオプションが使用でき、関連性の高いものが 表1.1「/etc/locale.conf で設定可能なオプション」 にまとめられています。これらのオプションの詳細については、**locale(7)** man ページを参照してください。すべての可能なオプションを表す LC_ALL オプションは **/etc/locale.conf** で設定しないように注意してください。

表1.1 /etc/locale.conf で設定可能なオプション

オプション	詳細
LANG	システムロケールのデフォルト値になります。
LC_COLLATE	ローカルのアルファベットでの文字列を比較する機能の動作を変更します。
LC_CTYPE	文字処理および分類機能とマルチバイト文字機能の動作を変更します。
LC_NUMERIC	小数点に対して小数点を表すコンマなどの詳細とともに数が通常出力される方法を記述します。
LC_TIME	24時間表記か12時間表記かという現在の時間表記を変更します。
LC_MESSAGES	標準エラー出力に書き出される診断メッセージに使用されるロケールを決定します。

1.1.1. 現行ステータスの表示

localectl コマンドを使うと、システムロケールとキーボードレイアウト設定に対してクエリを行ったり変更することができます。現行設定を表示するには、**status** オプションを使用します。

```
localectl status
```

例1.1 現行ステータスの表示

上記のコマンドの出力は、ロケールの現行設定と、コンソールおよび X11 ウィンドウシステム用に設定されているキーボードレイアウトを一覧表示します。

```
~]$ localectl status
   System Locale: LANG=en_US.UTF-8
       VC Keymap: us
      X11 Layout: n/a
```

1.1.2. 利用可能なロケールの一覧表示

ご使用のシステムで利用可能なロケールを一覧表示するには、以下を入力します。

```
localectl list-locales
```

例1.2 ロケールの一覧表示

特定の英語のロケール選択を希望している場合に、それがシステムで利用可能か分からないとします。以下のコマンドですべての英語ロケールを一覧表示させます。

```
~]$ localectl list-locales | grep en_
en_AG
en_AG.utf8
en_AU
en_AU.iso88591
en_AU.utf8
en_BW
en_BW.iso88591
en_BW.utf8

出力は省略されています
```

1.1.3. ロケールの設定

デフォルトのシステムロケールを設定するには、**root** で以下のコマンドを使用します。

```
localectl set-locale LANG=locale
```

locale を **list-locales** オプションで検索したロケール名で置き換えます。このコマンドでは、表 1.1「/etc/locale.conf で設定可能なオプション」 にあるオプションも設定できます。

例1.3 デフォルトロケールの変更

たとえば、British English をデフォルトのロケールに設定したい場合は、まず **list-locales** を使ってこのロケール名を見つけます。その後に、**root** ユーザーとして以下の形式のコマンドを入力します。

```
~]# localectl set-locale LANG=en_GB.utf8
```

1.2. キーボードレイアウトの変更

キーボードレイアウト設定では、ユーザーはテキストコンソールとグラフィカルユーザーインターフェイス
で使用するレイアウトを管理できます。

1.2.1. 現行設定の表示

上記の説明にあるように、現行のキーボードレイアウト設定は、以下のコマンドでチェックできます。

```
localectl status
```

例1.4 キーボード設定の表示

以下の出力では、仮想コンソールおよび X11 ウィンドウシステム用に設定されているキーボードレイアウトが表示されています。

```
~]$ localectl status
   System Locale: LANG=en_US.utf8
       VC Keymap: us
      X11 Layout: us
```

1.2.2. 利用可能なキーマップの一覧表示

システム上で設定できる利用可能なキーボードレイアウトすべてを一覧表示するには、以下を入力します。

```
localectl list-keymaps
```

例1.5 特定のキーマップの検索

grep を使うと、上記のコマンド出力から特定のキーマップ名を探すことができます。現在設定されているロケールと互換性のあるキーマップは、複数あることがよくあります。たとえば、利用可能な Czech キーボードレイアウトを見つけるには、以下を入力します。

```
~]$ localectl list-keymaps | grep cz
cz
cz-cp1250
cz-lat2
cz-lat2-prog
cz-qwerty
cz-us-qwertz
sunt5-cz-us
sunt5-us-cz
```

1.2.3. キーマップの設定

システムでデフォルトのキーボードレイアウトを設定するには、**root** で以下のコマンドを使用します。

```
localectl set-keymap map
```

map を**list-keymaps** の出力から得られたキーマップ名で置き換えます。**--no-convert** オプションが渡されていなければ、選択された設定は X11 キーボードマッピングに最も適合するものに変換された後、X11 ウィンドウシステムのデフォルトキーボードマッピングにも適用されます。これは逆の方法でも適用できます。以下のコマンドを使って **root** で両方のキーマップを指定できます。

```
localectl set-x11-keymap map
```

X11 レイアウトをコンソールのレイアウトとは別のものにしたい場合は、**--no-convert** オプションを使います。

```
localectl --no-convert set-x11-keymap map
```

このオプションでは、X11 キーマップはこれまでのコンソールレイアウト設定を変更することなく指定されます。

例1.6 X11 キーマップを別個に設定する

グラフィカルインターフェイスには German キーボードレイアウトを使用したいものの、コンソール操作には US キーマップを維持したいとします。この場合、以下を (**root** で) 入力します。

```
~]# localectl --no-convert set-x11-keymap de
```

現在ステータスをチェックすると、設定が正常に行われたかを検証できます。

```
~]$ localectl status
   System Locale: LANG=de_DE.UTF-8
       VC Keymap: us
      X11 Layout: de
```

キーボードレイアウト (*map*) 以外では、他に 3 つのオプションが指定できます。

```
localectl set-x11-keymap map model variant options
```

model はキーボードのモデル名で、*variant* と *options* はキーボードのバリアントとオプションコンポーネントで置き換えます。これを使うと、キーボードの動作を強化することができます。これらのオプションは、デフォルトでは設定されていません。X11 モデル、X11 バリアント、および X11 オプションについての詳細は、**kbd(4)** man ページを参照してください。

1.3. その他のリソース

Red Hat Enterprise Linux 上でキーボードレイアウトを設定する詳細方法については、以下のリソースを参照してください。

インストールされているドキュメント

- **localectl**(1) — **localectl** コマンドラインユーティリティーの man ページには、このツールを使ってシステムロケールとキーボードレイアウトを設定する方法が説明されています。
- **loadkeys**(1) — **loadkeys** コマンドの man ページでは、このツールを使用して仮想コンソールでキーボードレイアウトを変更する方法についての詳細な情報が提供されています。

関連項目

- 4章*権限の取得* では、**su** および **sudo** コマンドを使って管理者権限を取得する方法が説明されています。
- 6章*systemd によるサービス管理*では systemd に関する詳細情報と、**systemctl** コマンドを使ってシステムサービスを管理する方法が説明されています。

第2章 日付と時刻の設定

最新のオペレーティングシステムは、以下の 2 つのタイプのクロックを区別します。

- *リアルタイムクロック*(RTC) は、一般に *ハードウェアクロック* と呼ばれます。これは通常、システムボード上の集積回路で、オペレーティングシステムの現行状態からは完全に独立しており、コンピューターがシャットダウンしても稼働しています。
- *システムクロック*は *ソフトウェアクロック* とも呼ばれ、カーネルが維持し、その初期値はリアルタイムクロックに基づいています。システムが起動するとシステムクロックは初期化され、リアルタイムクロックとは完全に独立したものになります。

リアルタイムクロックは、*ローカルタイム* か *協定世界時* (UTC) のいずれかを使うことができます。リアルタイムクロックが UTC を使うように設定すると、システムクロックは使用中のタイムゾーンにその時間差を適用して計算され、該当する場合は、*夏時間* (DST) も同様に計算されます。対照的に、ローカルタイムは存在地のタイムゾーンにおける実際の時間を表します。ほとんどの場合は、UTC の使用が推奨されます。

Red Hat Enterprise Linux 7 では、以下の 3 つのコマンドラインツールでシステムの日付および時間に関する情報の設定および表示ができます。**timedatectl** ユーティリティーは Red Hat Enterprise Linux 7 で初めて導入されており、**systemd** の一部となります。**date** は従来のコマンドです。**hwclock** ユーティリティーは、ハードウェアクロックにアクセスするためのものです。

2.1. timedatectl コマンドの使用

timedatectl ユーティリティーは、**systemd** システムおよびサービスマネジャーの一部として配布されており、システムクロック設定の確認および変更ができます。このツールを使うと、現在の日付および時間の変更、タイムゾーンの設定、リモートサーバーとシステムクロックとの自動同期の有効化が可能になります。

カスタマイズした形式で現在の日付と時間を表示する方法については、「date コマンドの使用」も参照してください。

2.1.1. 現在の日時の表示

現在の日時をシステムおよびハードウェアクロック設定の詳細情報と共に表示するには、**timedatectl** コマンドをコマンドラインオプションなしで実行します。

```
timedatectl
```

これで、ローカル、ユニバーサル、および RTC 時間と、現在使用しているタイムゾーン、ネットワーク時刻プロトコル (**NTP**) 設定、さらに DST に関する追加情報が表示されます。

例2.1 現在の日時の表示

以下の例は、システムクロックとリモートサーバーの同期に **NTP** を使用しないシステムでの **timedatectl** コマンドの出力です。

```
~]$ timedatectl
      Local time: Mon 2013-09-16 19:30:24 CEST
  Universal time: Mon 2013-09-16 17:30:24 UTC
        Timezone: Europe/Prague (CEST, +0200)
     NTP enabled: no
NTP synchronized: no
```

```
RTC in local TZ: no
     DST active: yes
Last DST change: DST began at
                 Sun 2013-03-31 01:59:59 CET
                 Sun 2013-03-31 03:00:00 CEST
Next DST change: DST ends (the clock jumps one hour backwards) at
                 Sun 2013-10-27 02:59:59 CEST
                 Sun 2013-10-27 02:00:00 CET
```

2.1.2. 現在の日付の変更

現在の日付を変更するには、**root** でシェルプロンプトに以下を入力します。

```
timedatectl set-time YYYY-MM-DD
```

YYYY は 4 桁の年に、*MM* は 2 桁の月に、*DD* は 2 桁の日付に置き換えます。

例2.2 現在の日付の変更

現在の日付を 2013 年 6 月 2 日に変更するには、**root** で以下のコマンドを実行します。

```
~]# timedatectl set-time 2013-06-02
```

2.1.3. 現在の時刻の変更

現在の時刻を変更するには、**root** でシェルプロンプトに以下を入力します。

```
timedatectl set-time HH:MM:SS
```

HH は時間に、*MM* は分に、*SS* は秒にすべて 2 桁の数字で置き換えます。

デフォルトでは、システムは UTC を使うように設定されています。ローカルタイムでクロックを維持するようにシステムを設定するには、**root** で **timedatectl** コマンドを **set-local-rtc** オプションと実行します。

```
timedatectl set-local-rtc boolean
```

ローカルタイムでクロックを維持するには、*boolean* を **yes** に置き換えます。システムが UTC を使用するようにするには、*boolean* を **no** に置き換えます (デフォルトのオプション)。

例2.3 現在の時刻の変更

現在の時刻を午後 11 時 26 分に変更するには、**root** で以下のコマンドを実行します。

```
~]# timedatectl set-time 23:26:00
```

2.1.4. タイムゾーンの変更

利用可能なタイムゾーンすべてを一覧表示するには、シェルプロンプトで以下を入力します。

```
timedatectl list-timezones
```

現在使用中のタイムゾーンを変更するには、**root** で以下を入力します。

```
timedatectl set-timezone time_zone
```

time_zone を **timedatectl list-timezones** コマンドで表示される値に置き換えます。

例2.4 タイムゾーンの変更

ユーザーの位置に最も近いタイムゾーンを特定するには、**timedatectl** コマンドを **list-timezones** コマンドラインオプションと使用します。たとえば、ヨーロッパで利用可能なタイムゾーンすべてを一覧表示するには、以下を入力します。

```
~]# timedatectl list-timezones | grep Europe
Europe/Amsterdam
Europe/Andorra
Europe/Athens
Europe/Belgrade
Europe/Berlin
Europe/Bratislava…
```

タイムゾーンを **Europe/Prague** に変更するには、**root** で以下を入力します。

```
~]# timedatectl set-timezone Europe/Prague
```

2.1.5. システムクロックのリモートサーバーとの同期

上記の手動での設定とは反対に、**timedatectl** コマンドでは **NTP** を使ってシステムクロックを自動でリモートサーバーと同期させることができます。この機能を有効、無効にするには、**root** でシェルプロンフトに以下を入力します。

```
timedatectl set-ntp boolean
```

システムでシステムクロックをリモートの **NTP** サーバーと同期させるには、*boolean* を **yes** に置き換えます (デフォルトのオプション)。この機能を無効にするには、*boolean* を **no** に置き換えます。

例2.5 システムクロックのリモートサーバーとの同期

システムクロックとリモートサーバーとの自動同期を有効にするには、以下を入力します。

```
~]# timedatectl set-ntp yes
```

2.2. date コマンドの使用

date ユーティリティーはすべての Linux システムで利用可能で、現在の日時の表示および設定を可能とします。カスタマイズされた形式でシステムクロックについての詳細情報を表示するためにスクリプト内で使

われることが多くあります。

タイムゾーンの変更またはシステムクロックとリモートサーバーとの自動同期を有効にする方法については、「timedatectl コマンドの使用」を参照してください。

2.2.1. 現在の日時の表示

現在の日時を表示するには、**date** コマンドをコマンドラインオプションなしで実行します。

```
date
```

これで曜日、日付、ローカルタイム、タイムゾーンの省略形、年が表示されます。

デフォルトでは、**date** コマンドはローカルタイムを表示します。UTC で時間を表示するには、**--utc** または **-u** のコマンドラインオプションをつけてこのコマンドを実行します。

```
date --utc
```

表示される情報のフォーマットをカスタマイズするには、**+"*format*"** オプションをコマンドラインに追加します。

```
date +"format"
```

format を例2.6「現在の日時の表示」にあるように、サポートされる 1 つまたは複数のコントロールシーケンスに置き換えます。よく使われるフォーマットオプションのリストについては、表2.1「よく使われるコントロールシーケンス」を参照してください。また、これらのオプションの完全なリストは、**date**(1) man ページを参照してください。

表2.1 よく使われるコントロールシーケンス

コントロールシーケンス	詳細
%H	*HH* フォーマットでの時間 (例、**17**)。
%M	*MM* フォーマットでの分 (例、**30**)。
%S	*SS* フォーマットでの秒 (例、**24**)。
%d	*DD* フォーマットでの日付 (例、**16**)。
%m	*MM* フォーマットでの月 (例、**09**)。
%Y	*YYYY* フォーマットでの年 (例、**2013**)。
%Z	タイムゾーンの省略形 (例、**CEST**)。
%F	*YYYY-MM-DD* フォーマットでの完全な日付 (例、**2013-09-16**)。このオプションは **%Y-%m-%d** と同じです。
%T	*HH:MM:SS* フォーマットでの完全な時間 (例、17:30:24)。このオプションは **%H:%M:%S** と同じです。

例2.6 現在の日時の表示

UTC で現在の日時を表示するには、シェルプロンプトで以下を入力します。

```
~]$ date --utc
Mon Sep 16 17:30:24 CEST 2013
```

date コマンドの出力をカスタマイズするには、以下を入力します。

```
~]$ date +"%Y-%m-%d %H:%M"
```

```
2013-09-16 17:30
```

2.2.2. 現在の日付の変更

現在の日付を変更するには、**root** でシェルプロンプトに以下を入力します。

```
date +%F -s YYYY-MM-DD
```

YYYY は 4 桁の年に、*MM* は 2 桁の月に、*DD* は 2 桁の日付に置き換えます。

例2.7 現在の日付の変更

現在の日付を 2013 年 6 月 2 日に変更するには、**root** で以下のコマンドを実行します。

```
~]# date +%F -s 2013-06-02
```

2.2.3. 現在の時刻の変更

現在の時刻を変更するには、**root** で **date** コマンドを **--set** または **-s** オプションで実行します。

```
date +%T -s HH:MM:SS
```

HH は時間に、*MM* は分に、*SS* は秒にすべて 2 桁の数字で置き換えます。

デフォルトでは、**date** コマンドはシステムクロックをローカルタイムに設定します。システムクロックを UTC に設定するには、このコマンドを **--utc** か **-u** コマンドラインオプションで実行します。

```
date +%T --set HH:MM:SS --utc
```

例2.8 現在の時刻の変更

現在の時刻を午後 11 時 26 分に変更するには、**root** で以下のコマンドを実行します。

```
~]# date +%T --set 23:26:00
```

2.3. hwclock コマンドの使用方法

hwclock は、ハードウェアクロックにアクセスするためのユーティリティーです。ハードウェアクロックは使用中のオペレーティングシステムから独立しており、マシンがシャットダウンしても作動します。このプログラムは、ハードウェアクロックから時刻を見つけ、起動時にシステムタイムを設定するために使用されます。また **hwclock** には、ハードウェアクロック内の体系的なずれを補正する機能もあります。

hwclock は、年、月、日、時間、分、秒の値を保存します。ローカルタイムや協定世界時 (UTC) の時刻を保存したり、夏時間 (DST) を設定したりすることはできません。

注記

Red Hat Enterprise Linux 6 では、**hwclock** コマンドはシステム終了時や再起動時に自動的に実行されていましたが、Red Hat Enterprise Linux 7 では実行されません。

2.3.1. 現在の日時の表示

root でコマンドラインオプションなしに**hwclock** を実行すると、現在の日時の情報についての文字列が返されます。

```
hwclock
```

例2.9 現在の日時の表示

hwclock コマンドは、標準出力で時刻を返します。ハードウェアクロックを UTC で保持しても、表示される時刻はデフォルトでローカルタイムとなります。**/etc/adjtime** ファイルを開くと、デフォルト設定が確認できます。

```
~]# hwclock
  Tue 15 Apr 2014 04:23:46 PM CEST -0.329272 seconds
```

CEST は 中央ヨーロッパ夏時間 (Central European Summer Time) の省略形です。

マシンの日時を UTC で表示するには、以下を実行します。

```
hwclock --utc
```

例2.10 --utc オプションの使用

例2.9「現在の日時の表示」は現在の UTS 仕様を示しているので、**hwclock --utc** コマンドの出力はこれとまったく同じものになります。

```
~]# hwclock --utc
  Tue 15 Apr 2014 04:23:46 PM CEST -0.329272 seconds
```

ローカルタイムで日時を表示するには、**--localtime** オプションを使います。

```
hwclock --localtime
```

例2.11 --localtime オプションの使用

ローカルの日時を表示するには、**hwclock --localtime** コマンドを実行します。

```
~]# hwclock --localtime
  Tue 15 Apr 2014 02:23:46 PM CEST -0.329272 seconds
```

注記

クロックを UTC で維持するか、ローカルタイムで維持するかは、ユーザー次第です。ただし、**hwclock --localtime** を実行する際にハードウェアクロックが UTC に設定されていると、**hwclock** は UTC とローカルタイムの差で構成される時間差を返すことに注意してください。本来の設定や RTC タイムは変更されません。

2.3.2. 日付と時刻の設定と同期

日付と時刻を表示するほかに、**hwclock** ユーティリティーを手動で特定の時間に設定したり、システムタイムをハードウェアクロックに同期させたり、その逆の同期を行うことができます。

ハードウェアクロックの日時を変更するには、仕様に **--set** と **--date** のオプションを追加します。

```
hwclock --set --date "dd mmm yyyy HH:MM"
```

dd は 2 桁の日付に、*mmm* は 3 文字の月の省略形に、*yyyy* は 4 桁の年に、*HH* は 2 桁の時間に、*MM* は 2 桁の分に置き換えます。

例2.12 ハードウェアクロックを特定の日時に設定する

日時をたとえば、2014 年 10 月 21 日 21:17 に設定したい場合は、以下のフォーマットでコマンドを実行します。

```
~]# hwclock --set --date "21 Oct 2014 21:17"
```

また、同期を利用してハードウェアクロックを現在のシステムタイムに設定することもできます。以下のコマンドは、システムタイムをハードウェアクロックに同期させます。

```
hwclock --systohc
```

再調整するには、2 つの方法があります。

- ハードウェアクロックをシステムタイムに再調整し、ローカルタイムに向けることができます。

```
hwclock --systohc --localtime
```

- または、ハードウェアクロックをシステムタイムに再調整し、UTC に向けることもできます。

```
hwclock --systohc --utc
```

このコマンドは **rtc** ユーティリティーを更新します。これにより新たな設定が保存され、**hwclock --localtime** または **hwclock --utc** を実行すると、それぞれの時間が表示されます。同様の機能は **hwclock --adjust** でも提供されます。

この設定は、1 つのマシン上で複数の Linux ディストリビューションを起動していて、各ディストリビューションで異なるタイムゾーンを設定している場合に特に便利です。

例2.13 システムタイムをハードウェアクロックに同期させる

システムタイムをハードウェアクロックに同期させるには、**root** で以下のコマンドを実行します。

```
~]# hwclock --systohc --localtime
```

変更は、システムを再起動すると反映されます。

ハードウェアクロックからシステムタイムを設定するという逆の操作は、以下を実行します。

```
hwclock --hctosys
```

例2.14 ハードウェアクロックをシステムタイムに同期させる

ハードウェアクロックをシステムタイムに同期させるには、**root** で以下のコマンドを実行します。

```
~]# hwclock --hctosys --localtime
```

変更は、システムを再起動すると反映されます。

2.4. その他のリソース

Red Hat Enterprise Linux 7 で日時を設定する詳細情報は、以下に挙げるリソースを参照してください。

インストールされているドキュメント

- **timedatectl**(1) — **timedatectl** コマンドラインユーティリティーの man ページでは、このツールを使ってシステムクロックおよびその設定に対するクエリーと変更を行う方法が説明されています。
- **date**(1) — **date** コマンドの man ページでは、サポートされるコマンドラインオプションの完全なリストが提供されます。
- **hwclock**(8) — **hwclock** コマンドの man ページでは、サポートされるコマンドラインオプションの完全なリストが提供されます。

関連項目

- 1章*システムロケールおよびキーボード設定*は、キーボードレイアウトの設定方法を説明しています。
- 4章*権限の取得* では、**su** および **sudo** コマンドを使って管理者権限を取得する方法が説明されています。
- 6章*systemd によるサービス管理*では systemd に関する詳細情報と、**systemctl** コマンドを使ってシステムサービスを管理する方法が説明されています。

第3章 ユーザーとグループの管理

ユーザーとグループの制御は、Red Hat Enterprise Linux システム管理の中核となる要素です。本章では、グラフィカルユーザーインターフェースおよびコマンドライン上でユーザーとグループを追加、管理、削除する方法について説明し、パスワードエージングの有効化やグループディレクトリーの作成などの高度なトピックを扱います。

3.1. ユーザーとグループの概要

ユーザーが人 (実在のユーザーに関連付けられたアカウント) もしくは使用する特定のアプリケーション用に存在するアカウントのどちらかであるのに対し、グループは共通目的のためにユーザーを統合して組織を論理的に表現したものです。グループ内のユーザーは、そのグループが所有するファイルの読み取り、書き込み、実行が可能です。

各ユーザーは、*ユーザー ID* (UID) と呼ばれる一意の数値 ID に関連付けられています。同様に、各グループは *グループ ID* (GID) に関連付けられています。ファイルを作成するユーザーは、そのファイルの所有者でありグループ所有者でもあります。ファイルには所有者、グループ、その他に対して読み取り、書き込み、実行のパーミッションが別々に割り当てられます。ファイルの所有者は **root** のみが変更可能です。アクセスパーミッションは、**root** ユーザーとファイル所有者の両方が変更可能です。

さらに、Red Hat Enterprise Linux はファイルとディレクトリーに対する *アクセス制御リスト* (ACL) をサポートしています。これにより、所有者以外の特定のユーザーにパーミッションを設定することができます。この機能の詳細については、『Storage Administration Guide (ストレージ管理ガイド)』 の 『Access Control Lists (アクセス制御リスト)』 の章を参照してください。

3.1.1. ユーザープライベートグループ

Red Hat Enterprise Linux では、*UPG* (*ユーザープライベートグループ*) スキームを使用するため、UNIX グループを簡単に管理することができます。ユーザープライベートグループは、新規ユーザーがシステムに追加される度に作成されます。ユーザープライベートグループは作成されたユーザーと同じ名前を持ち、そのユーザーがそのユーザープライベートグループの唯一のメンバーです。

ユーザープライベートグループを使用すると、新規作成されたファイルやディレクトリーに対し安全にデフォルトのパーミッションを設定することができます。また、ユーザーと **そのユーザーのグループ** の両方がファイルやディレクトリーの修正をできるようになります。

新規作成されたファイルやディレクトリーに適用されるパーミッションの設定は *umask* と呼ばれ、**/etc/bashrc** ファイルで設定されます。従来の UNIX システムでは、**umask** は **022** に設定されており、ファイルやディレクトリーを作成したユーザーのみが修正できるようになっています。このスキームでは、**作成者のグループのメンバーを含め**他のどのユーザーも修正を行うことはできません。一方、UPG スキームではどのユーザーもそれぞれ独自のプライベートグループを持っているため、この 「グループ保護」 は必要ありません。

3.1.2. シャドウパスワード

マルチユーザー環境では、*shadow-utils* パッケージで提供される *シャドウパスワード* を使用することが非常に重要です。これを使用することで、システムの認証ファイルのセキュリティーを強化できます。このため、インストールプログラムはシャドウパスワードをデフォルト設定で有効にしています。

以下は、UNIX ベースシステムでパスワードを格納する従来の方法と比べた場合のシャドウパスワードの利点です。

- シャドウパスワードは、暗号化されたパスワードハッシュを、あらゆるユーザーが読み取り可能な **/etc/passwd** ファイルから **root** ユーザーのみが読み取り可能な **/etc/shadow** に移動して、システムセキュリティーを向上させます。

- シャドウパスワードは、パスワードエージングに関する情報を保存します。
- シャドウパスワードを使用すると、**/etc/login.defs** ファイルによるセキュリティーポリシーの実施が可能になります。

shadow-utils パッケージが提供する大半のユーティリティーは、シャドウパスワードが有効かどうかに関わらず適切に動作します。ただし、パスワードエージングの情報は **/etc/shadow** ファイルにのみ格納されているため、パスワードエージングの情報を作成、修正するコマンドはいずれも機能しません。以下は、シャドウパスワードを先に有効にする必要があるユーティリティーとコマンドの一覧です。

- **chage** ユーティリティー
- **gpasswd** ユーティリティー
- **-e** または **-f** オプションを付けた **usermod** コマンド
- **-e** または **-f** オプションを付けた **useradd** コマンド

3.2. ユーザー管理ツールの使用

ユーザー管理 のアプリケーションを使用すると、グラフィカルユーザーインターフェース上でローカルユーザーおよびグループを表示、修正、追加、削除することができます。**ユーザー管理** を起動するには、パネルから **システム** → **管理** → **ユーザーとグループ** の順にクリックします。別の方法として、シェルプロンプトで **system-config-users** と入力することも可能です。スーパーユーザー権限を持っていない場合は、アプリケーションは **root** として認証するようプロンプトします。

3.2.1. ユーザーとグループの表示

ユーザー管理 の主要ウィンドウは、2 つのタブに分かれています。**ユーザー** タブは、ローカルユーザーのユーザー ID、プライマリグループ、ホームディレクトリ、ログインシェル、氏名などの追加情報を含んだローカルユーザーの一覧を表示します。**グループ** タブは、ローカルグループのグループ ID とグループメンバーに関する情報を含むローカルグループの一覧を表示します。

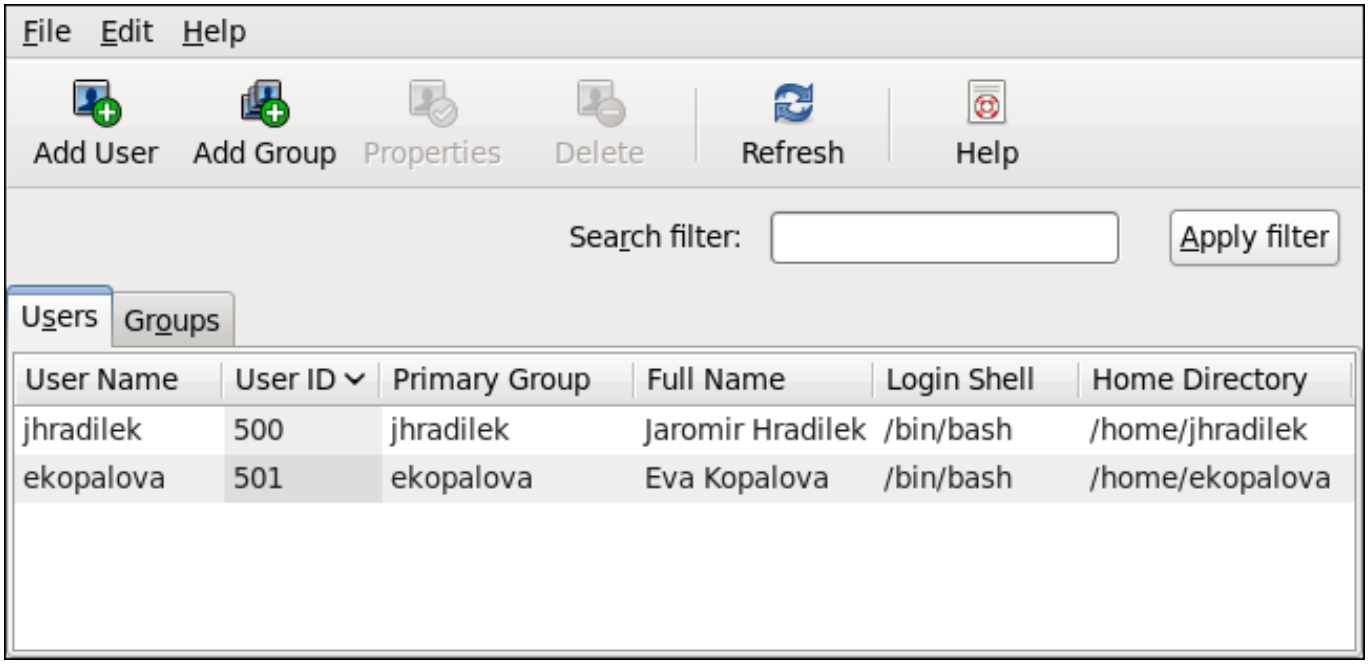


図3.1 ユーザーとグループの表示

特定のユーザーやグループを検索したい場合は、**フィルターの探索** フィールドに検索したい名前の最初の数文字を入力します。そして、**Enter** を押すか **フィルターの適用** ボタンをクリックします。また、いずれかのカラムのヘッダーをクリックするとそのカラムに従って項目を並び替えることもできます。

Red Hat Enterprise Linux 7 は、システムユーザーとグループ用に 1000 未満の数字のユーザー ID とグループ ID を残しています。デフォルトでは、**ユーザー管理** はシステムユーザーを表示しません。すべてのユーザーとグループを表示するには、**編集** → **設定** の順にクリックして、ダイアログボックスの**個人設定** を開きます。そして、**Hide system users and groups (システムユーザーとグループを非表示)** のチェックマークを外します。

3.2.2. 新規ユーザーの追加

新規ユーザーを追加するには、**ユーザーを追加** ボタンをクリックします。図3.2「新規ユーザーの追加」のようなウィンドウが表示されます。

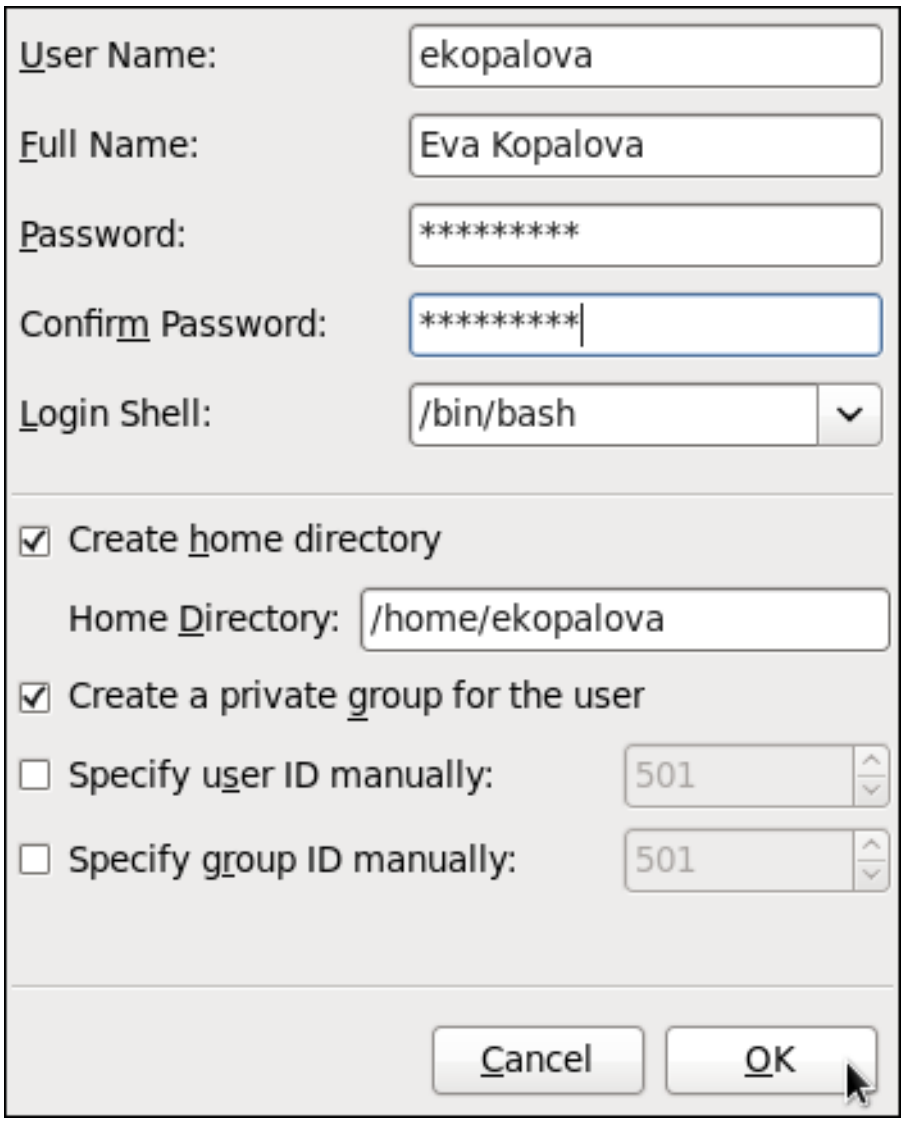


図3.2 新規ユーザーの追加

Add New User (新規ユーザーの追加) ダイアログボックスを使用して、新規作成されたユーザーに関する情報を入力できます。ユーザーを作成するには、該当フィールドにユーザー名と氏名を入力して、**Password (パスワード)** と **Confirm Password (パスワードの確認)** フィールドにユーザーのパスワードを入力します。パスワードは 6 文字以上にする必要があります。

注記

できるだけ長いパスワードを使用することをお勧めします。長いパスワードを使用することで、侵入者がパスワードを推測してパーミッションなしにアカウントにアクセスすることを困難にできます。また、辞書に載っている単語をベースとするパスワードは避け、文字や数字、特殊文字を組み合わせることが推奨されます。

ログインシェル プルダウンの一覧から、ユーザー用のログインシェルを選択できます。どのシェルを選択すべきか不明な場合は、デフォルト値である **/bin/bash** を使用して下さい。

デフォルトでは、**ユーザー管理** アプリケーションは新規ユーザー用のホームディレクトリを **/home/*username*/** に作成します。これを変更するには、**Create home directory (ホームディレクトリの作成)** チェックボックスのチェックを外してホームディレクトリを作成しないように選択するか、**Home Directory (ホームディレクトリ)** テキストボックスの内容を編集してディレクトリを変更します。ホームディレクトリが作成されると、デフォルトの設定ファイルは **/etc/skel/** ディレクトリからそこへコピーされます。

Red Hat Enterprise Linux は、UPG *(ユーザープライベートグループ)*スキームを使用します。新規ユーザーを作成する度に、デフォルトによりそのユーザーと同じ名前を持つ一意のグループが作成されます。このグループを作成したくない場合は、**Create a private group for the user (ユーザー用にプライベートグループを作成)** からチェックマークを外します。

ユーザー用のユーザー ID を指定するには、**Specify user ID manually (ユーザー ID を手動で指定)** を選択します。このオプションが選択されていない場合は、1000 を超える数字の使用可能なユーザー ID が新規ユーザーに割り当てられます。Red Hat Enterprise Linux 7 は、システムユーザー用に 1000 未満の数字のユーザー ID を確保しているため、1 から 999 の数字のユーザー ID を手動で割り当てることは推奨されません。

OK ボタンをクリックすると、新規ユーザーが作成されます。パスワードの有効期限などさらに高度なユーザープロパティを設定するには、ユーザーの追加後にそのプロパティを編集します。

3.2.3. 新規グループの追加

新規ユーザーグループを追加するには、ツールバーの **Add Group (グループを追加)** を選択します。図 3.3「新規グループ」 に似たウィンドウが現れますので、新規グループの名前を入力します。それにグループ ID を指定するには、**Specify group ID manually (グループ ID を手動で指定)** にチェックマークを入れ GID を選択します。Red Hat Enterprise Linux 7 ではシステムグループ用に 1000 未満のグループ ID も確保している点に注意してください。

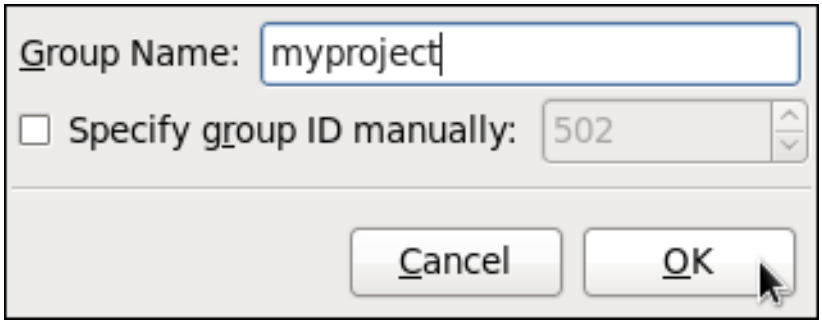


図3.3 新規グループ

OK をクリックするとグループが作成され、グループ一覧に表示されます。

3.2.4. ユーザープロパティーの変更

3.2.4. ユーザープロパティーの変更

既存ユーザーのプロパティを表示するには、**ユーザー** タブをクリックし、ユーザーの一覧から該当するユーザーをクリックし、メニューから **プロパティ** をクリック (またはプルダウンメニューから**ファイル** → **プロパティ** の順に選択) します。図3.4「ユーザープロパティ」 に似たウィンドウが表示されます。

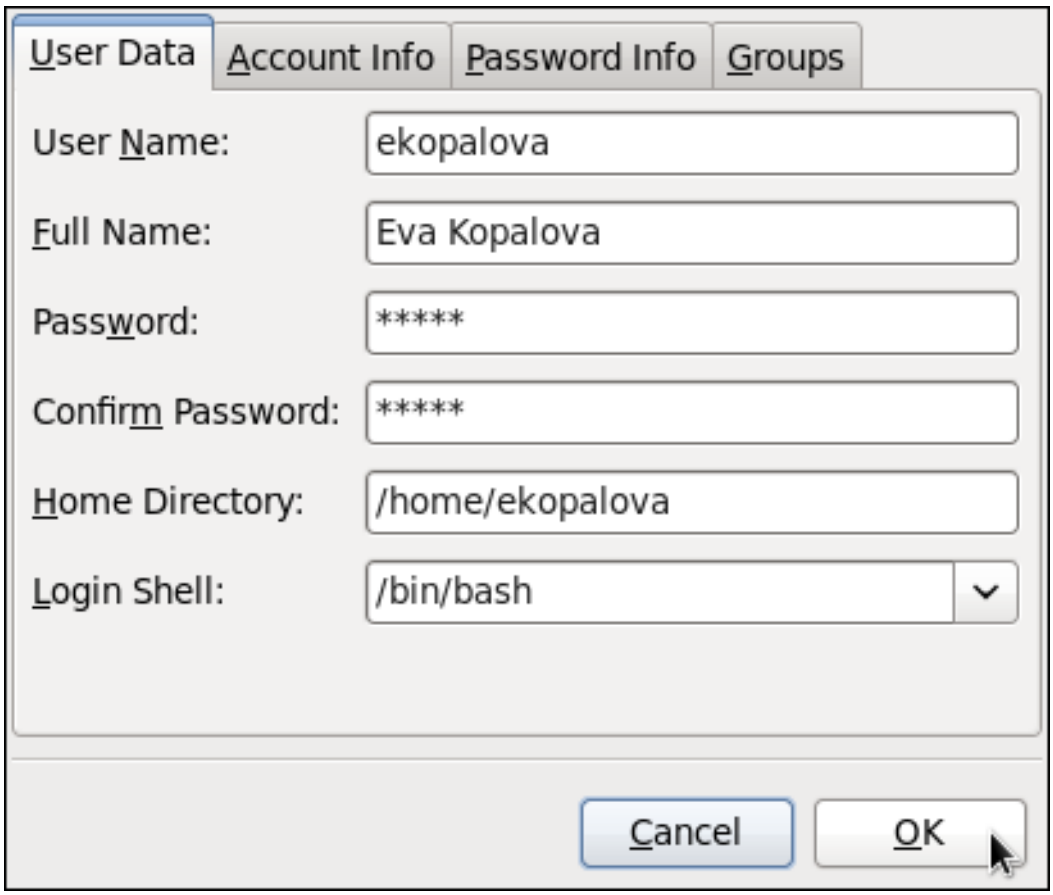


図3.4 ユーザープロパティ

ユーザープロパティ ウィンドウは複数のタブページに分かれています。

- **ユーザーデータ** — ユーザーの追加時に設定する基本情報を示しています。このタブを使用すると、ユーザーの氏名、パスワード、ホームディレクトリ、ログインシェルを変更できます。
- **アカウント情報** — 特定の日にアカウントを失効させたい場合は**アカウント失効を有効にする** を選択します。表示されたフィールドに日付を入力します。ユーザーアカウントをロックしてユーザーがシステムにログインできないようにするには、**ローカルパスワードがロックされています** にチェックマークを入れて下さい。
- **パスワード情報** — ユーザーのパスワードが最後に変更された日を表示します。一定の日数が経過した後にユーザーがパスワードを変更するよう強制的に設定するには、**パスワード失効を有効にする** を選択して、**変更が要求されるまでの日数** フィールドに希望する値を入力します。ユーザーのパスワードが失効するまでの日数、ユーザーがパスワードを変更するよう警告されるまでの日数、アカウントが無効になるまでの日数はすべて変更できます。
- **グループ** — ユーザーのプライマリグループ、さらにはユーザーをメンバーにしたい他のグループも表示/設定することができます。

3.2.5. グループプロパティーの変更

既存グループのプロパティを表示するには、グループ一覧から該当するグループを選択し、メニューから **プロパティ** をクリックします (またはプルダウンメニューから**ファイル** → **プロパティ** の順に選択します)。図3.5「グループのプロパティ」 に似たウィンドウが表示されます。

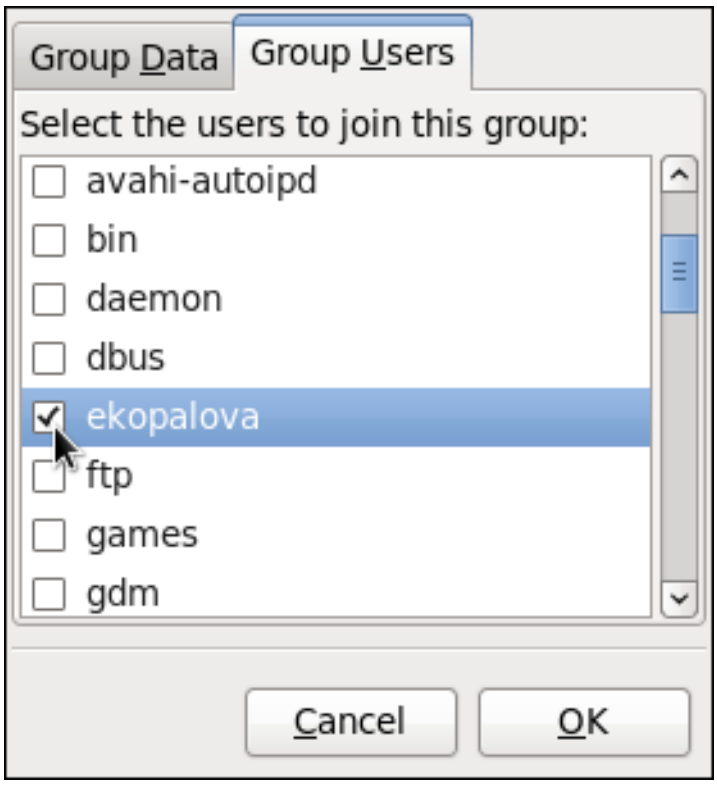


図3.5 グループのプロパティ

グループユーザー タブはグループのメンバーであるユーザーを表示します。このタブを使って、グループに対してユーザーの追加/削除を行います。**OK** をクリックし、変更を保存します。

3.3. コマンドラインツールの使用

Red Hat Enterprise Linux でユーザーを管理する最も簡単な方法は、「ユーザー管理ツールの使用」 で説明のとおり **ユーザー管理** アプリケーションを使用することです。しかし、コマンドラインツールを好む場合や X Window System がインストールされていない場合には、表3.1「ユーザーとグループを管理するためのコマンドラインユーティリティー」 に一覧表示されているコマンドラインユーティリティーを使用することができます。

表3.1 ユーザーとグループを管理するためのコマンドラインユーティリティー

ユーティリティー	詳細
useradd, **usermod**, **userdel**	ユーザーアカウントを追加、修正、削除するための標準ユーティリティーです。
groupadd, **groupmod**, **groupdel**	グループを追加、修正、削除するための標準ユーティリティーです。
gpasswd	**/etc/group** 設定ファイルを管理するための標準ユーティリティーです。
pwck, **grpck**	パスワード、グループ、関連シャドウファイルを検証するためのユーティリティーです。

ユーティリティー	詳細
pwconv, **pwunconv**	通常のパスワードをシャドウパスワードに変換するため、または逆にシャドウパスワードから通常のパスワードに変換するためのユーティリティーです。

3.3.1. 新規ユーザーの追加

システムにユーザーを追加するには、**root** としてシェルプロンプトで以下を入力します:

```
useradd [options] username
```

ここでの *options* は 表3.2「useradd のコマンドラインオプション」 の説明にあるコマンドラインオプションです。

デフォルトでは、**useradd** コマンドはロックされたユーザーアカウントを作成します。アカウントのロックを解除するには、**root** として以下のコマンドを実行して、パスワードを割り当てます:

```
passwd username
```

オプションとして、パスワードエージングポリシーを設定することもできます。パスワードエージングを有効にする方法については、『Red Hat Enterprise Linux 7 Security Guide』 を参照してください。

表3.2 useradd のコマンドラインオプション

オプション	詳細
-c '*comment*'	*comment* はどのような文字列でも置換できます。このオプションは、通常ユーザーの氏名を指定するために使用されます。
-d *home_directory*	デフォルトの **/home/*username*/** の代わりに使用するホームディレクトリーです。
-e *date*	YYYY-MM-DD の形式でアカウントを無効にする日付です。
-f *days*	パスワードが失効してからアカウントが無効になるまでの日数です。**0** を指定すると、パスワードが失効した直後にアカウントは無効になります。**-1** を指定すると、パスワードが失効した後でもアカウントは無効になりません。
-g *group_name*	ユーザーのデフォルトグループ用のグループ名またはグループ番号です。グループはここで指定される以前から存在していなければなりません。
-G *group_list*	ユーザーがメンバーとなる追加 (デフォルト以外) のグループ名またはグループ番号の一覧で、コンマで区切ります。グループはここで指定される以前から存在していなければなりません。
-m	ホームディレクトリーが存在しない場合に作成します。
-M	ホームディレクトリを作成しません。
-N	ユーザー用のユーザープライベートグループを作成しません。
-p *password*	**crypt** で暗号化されたパスワードです。
-r	UID が 1000 未満でホームディレクトリーがないシステムアカウントを作成します。
-s	ユーザーのログインシェルです。デフォルトでは**/bin/bash** に設定されています。
-u *uid*	ユーザーのユーザー ID です。一意の番号で 999 より大きい数でなければなりません。

プロセスの説明

以下のステップは、シャドウパスワードが有効なシステム上で **useradd juan** コマンドを実行したときに発生する内容を解説したものです。

1. **/etc/passwd** に **juan** の新しい行が作成されます。

```
juan:x:1001:1001::/home/juan:/bin/bash
```

この行には以下の特徴があります。

- ユーザー名 **juan** で始まっています。
- パスワードフィールドには **x** があり、システムがシャドウパスワードを使用していることを示しています。
- 999 より大きい数字の UID が作成されます。Red Hat Enterprise Linux 7 では、1000 未満の UID はシステム使用のために確保されています。これらはユーザーに割り当てないことをお勧めします。
- 999 より大きい GID が作成されます。Red Hat Enterprise Linux 7 では、1000 未満の GID はシステム使用のために確保されています。これらはユーザーに割り当てないことをお勧めします。
- オプションの *GECOS* 情報は空白のままです。GECOS フィールドは、氏名や電話番号などユーザーの追加情報を提供するために使用されます。
- **juan** のホームディレクトリーは **/home/juan/** に設定されています。
- デフォルトのシェルは **/bin/bash** に設定されています。

2. **juan** 用の新しい行が **/etc/shadow** に作成されます。

```
juan:!!:14798:0:99999:7:::
```

この行には以下の特徴があります。

- ユーザー名 **juan** で始まります。
- 2 つの感嘆符 (**!!**) が **/etc/shadow** ファイルのパスワードフィールドに表示され、これがアカウントをロックします。

注記

暗号化されたパスワードが **-p** フラグを使用して渡される場合、**/etc/shadow** ファイル内のユーザー用の新しい行に置かれます。

- パスワードは有効期限なしで設定されています。

3. **juan** というグループ用の新しい行が**/etc/group** に作成されます。

```
juan:x:1001:
```

ユーザーと同じ名前のグループは、*ユーザープライベートグループ*と呼ばれます。ユーザープライベートグループの詳細については 「ユーザープライベートグループ」を参照してください。

/etc/group に作成された行には、以下の特徴があります。

- グループ名 **juan** で始まります。
- パスワードフィールドに **x** が表示され、システムがシャドウグループパスワードを使用していることを示しています。
- GID は **/etc/passwd** 内に表示されているユーザー**juan** の GID と一致します。

4. **juan** というグループ用の新しい行が**/etc/gshadow** に作成されます。

```
juan:!::
```

この行には以下の特徴があります。

- グループ名 **juan** で始まります。
- 1 つの感嘆符 (**!**) が **/etc/gshadow** ファイルのパスワードフィールドに表示され、これがグループをロックします。
- その他のフィールドはすべて空白です。

5. ユーザー **juan** 用のディレクトリーが **/home/** ディレクトリー内に作成されます。

```
~]# ls -l /home
total 4
drwx------. 4 juan juan 4096 Mar  3 18:23 juan
```

このディレクトリーは、ユーザー **juan** とグループ **juan** が所有しています。ユーザー**juan** に ***のみ*** *読み取り*、*書き込み* および *実行* の権限が与えられています。その他のパーミッションはすべて拒否されます。

6. **/etc/skel/** ディレクトリー (デフォルトのユーザー設定を含む) 内のファイルが、新しい **/home/juan/** ディレクトリーにコピーされます。

```
~]# ls -la /home/juan
total 28
drwx------. 4 juan juan 4096 Mar  3 18:23 .
drwxr-xr-x. 5 root root 4096 Mar  3 18:23 ..
-rw-r--r--. 1 juan juan   18 Jun 22  2010 .bash_logout
-rw-r--r--. 1 juan juan  176 Jun 22  2010 .bash_profile
-rw-r--r--. 1 juan juan  124 Jun 22  2010 .bashrc
drwxr-xr-x. 2 juan juan 4096 Jul 14  2010 .gnome2
drwxr-xr-x. 4 juan juan 4096 Nov 23 15:09 .mozilla
```

この時点で、**juan** と呼ばれるロックされたアカウントがシステム上に存在します。これをアクティベートするには、管理者が **passwd** コマンドを使用してアカウントにパスワードを割り当てる必要があります。パスワードエージングのガイドラインを設定するというオプションもあります。

3.3.2. 新規グループの追加

システムに新規グループを追加するには、**root** としてシェルプロンプトで以下を入力します。

```
groupadd [options] group_name
```

ここでの *options* は、表3.3「groupadd のコマンドラインオプション」の説明にあるコマンドラインオプションです。

表3.3 groupadd のコマンドラインオプション

オプション	詳細
-f, **--force**	**-g** *gid* と併用し、*gid* がすでに存在している場合、**groupadd** はグループ用に別の一意の *gid* を選択します。
-g *gid*	グループのグループ ID です。一意の番号で 499 より大きい数でなければなりません。
-K, **--key** *key=value*	**/etc/login.defs** のデフォルトを上書きします。
-o, **--non-unique**	グループの複製を許可します。
-p, **--password** *password*	新規グループ用にこの暗号化されたパスワードを使用します。
-r	GID が 1000 未満のシステムグループを作成します。

3.3.3. グループディレクトリーの作成

システム管理者は、通常それぞれの主要なプロジェクトごとにグループを作成し、そのプロジェクトのファイルにアクセスする必要がある場合に人をグループに割り当てます。こうした従来型のスキームの場合、誰かがファイルを作成すると、ファイルはその人が属するプライマリグループに関連付けられるため、ファイル管理は困難です。1 人が複数のプロジェクトに関わっている場合、正しいファイルを正しいグループに関連付けることは難しくなります。一方で、UPG スキームを使用すると、グループは *setgid* ビットセットを持つディレクトリー内に作成されたファイルに自動的に割り当てられます。setgid ビットにより、共通のディレクトリーを共有するグループプロジェクトを非常に簡単に管理することができます。これは、ユーザーがディレクトリー内で作成するすべてのファイルが、ディレクトリーを所有するグループによって所有されるためです。

例えば、あるグループが **/opt/myproject/** ディレクトリー内のファイルで作業する必要があるとします。グループの中にはこのディレクトリーのコンテンツの修正を信頼して任せられる人もいますが、全員ではありません。

1. **root** としてシェルプロンプトで以下を入力して、**/opt/myproject/** ディレクトリーを作成します。

```
mkdir /opt/myproject
```

2. システムに **myproject** グループを追加します。

```
groupadd myproject
```

3. **/opt/myproject/** ディレクトリーのコンテンツと **myproject** グループを関連付けます。

```
chown root:myproject /opt/myproject
```

4. ユーザーがディレクトリー内のファイルを作成できるよう、setgid ビットを設定します。

```
chmod 2775 /opt/myproject
```

この時点で、ユーザーが新規ファイルの書き込みをする度に管理者はファイルのパーミッションを変更する必要なく、**myproject** グループの全メンバーは **/opt/myproject/** ディレクトリー内のファイルを作成/編集することができます。パーミッションが正しく設定されていることを確認するには、以下のコマンドを実行します。

```
~]# ls -l /opt
total 4
drwxrwsr-x. 3 root myproject 4096 Mar  3 18:31 myproject
```

3.4. その他のリソース

Red Hat Enterprise Linux でユーザーとグループを管理する方法についての詳細情報は、下記のリソースを参照してください。

インストールされているドキュメント

ユーザーおよびグループ管理のための各種ユーティリティーの詳細情報については、以下の man ページを参照してください。

- **useradd**(8) — **useradd** コマンドの man ページでは、新規ユーザーの作成方法が説明されています。
- **userdel**(8) — **userdel** コマンドの man ページでは、ユーザーの削除方法が説明されています。
- **usermod**(8) — **usermod** コマンドの man ページでは、ユーザーの変更方法が説明されています。
- **groupadd**(8) — **groupadd** コマンドの man ページでは、新規グループの作成方法が説明されています。
- **groupdel**(8) — **groupdel** コマンドの man ページでは、グループの削除方法が説明されています。
- **groupmod**(8) — **groupmod** コマンドの man ページでは、グループメンバーシップの変更方法が説明されています。
- **gpasswd**(1) — **gpasswd** コマンドの man ページでは、**/etc/group** ファイルの管理方法が説明されています。
- **grpck**(8) — **grpck** コマンドの man ページでは、**/etc/group** ファイルの整合性の検証方法が説明されています。
- **pwck**(8) — **pwck** コマンドの man ページでは、**/etc/passwd** と **/etc/shadow** ファイルの整合性の検証方法が説明されています。
- **pwconv**(8) — **pwconv** コマンドの man ページでは、標準パスワードをシャドーパスワードに変換する方法が説明されています。
- **pwunconv**(8) — **pwunconv** コマンドの man ページでは、シャドーパスワードを標準パスワードに変換する方法が説明されています。

関連する設定ファイルの詳細については、以下をご覧ください。

- **group**(5) — **/etc/group** ファイルの man ページでは、システムグループの定義方法が説明されています。
- **passwd**(5) — **/etc/passwd** ファイルの man ページでは、ユーザー情報の定義方法が説明されています。
- **shadow**(5) — **/etc/shadow** ファイルの man ページでは、システムにおけるパスワードとアカウントの有効期限情報の設定方法が説明されています。

オンラインのドキュメント

- Red Hat Enterprise Linux 7 Security Guide — Red Hat Enterprise Linux 7 の 『Security Guide』 では、パスワードのエージングとユーザーアカウントのロックを有効にしてパスワードとワークステーションのセキュリティーを高める追加情報を提供しています。

関連項目

- 4章*権限の取得* では、**su** および **sudo** コマンドを使って管理者権限を取得する方法が説明されています。

第4章 権限の取得

システム管理者は (時にはユーザーも) 管理者アクセスでタスクを実行する必要があります。システムに **root** でアクセスすることは危険を伴う可能性があり、システムおよびデータの大幅な破損につながる場合もあります。本章では、**su** や **sudo** といった setuid プログラムを使用して管理者権限を取得する方法を説明します。これらのプログラムを使うと、特定のユーザーが高レベルの制御およびシステムセキュリティーを維持しつつ、通常は root ユーザーしかできないタスクを実行することができます。

管理者制御や潜在的な危険、管理者アクセスの不適切な使用によるデータ破損の回避方法についての詳細は、『Red Hat Enterprise Linux 7 Security Guide』 を参照してください。

4.1. su コマンド

ユーザーは、**su** を実行すると root パスワードを求められ、認証後に root シェルプロンプトが表示されます。

su コマンドでログインした後は、そのユーザーは root ユーザーとなり、システムへの絶対管理アクセスを持つことになります [1]。さらに、root になるとユーザーは**su** コマンドを使って、パスワードを求められることなくシステム上の他のユーザーに変わることができます。

このプログラムは非常に強力なので、組織内の管理者はこのコマンドにアクセスできる人を制限してください。

簡単な制限方法は、*wheel* と呼ばれる特別な管理グループにユーザーを追加することです。これを実行するには、以下のコマンドを root で入力します。

```
usermod -G wheel <username>
```

このコマンドで *<username>* の部分を **wheel** グループに追加したいユーザーに置き換えます。

また、**User Manager** を使って以下のようにグループのメンバーを修正することもできます。この手順を実行するには、管理者権限が必要なことに注意してください。

1. パネル上の **システム** メニューをクリックして **管理** から **ユーザーとグループ** をクリックして User Manager を表示させます。別の方法では、シェルプロンプトで **system-config-users** のコマンドを入力します。
2. **ユーザー** タブをクリックして、ユーザーリストの中から必要なユーザーを選択します。
3. ツールバーの **設定** をクリックして、ユーザー設定のダイアログボックスを表示させます (または **ファイル** メニューで **設定** を選択します) 。
4. **グループ** タブをクリックし、wheel グループのチェックボックスにチェックマークを付けて**OK** をクリックします。

User Manager の詳細に関しては、「ユーザー管理ツールの使用」 を参照してください。

wheel グループにユーザーを追加した後は、この追加した特定のユーザーのみに**su** コマンドの使用を許可することが推奨されます。それには、**su**: **/etc/pam.d/su** で PAM 設定ファイルを編集する必要があります。このファイルをテキストエディターで開き、以下の行からコメント (#) を削除します。

```
#auth            required        pam_wheel.so use_uid
```

この変更で、**wheel** の管理グループメンバーのみが**su** コマンドを使って別のユーザーに変わることができるようになります。

注記

root ユーザーはデフォルトで **wheel** グループの一部となっています。

4.2. sudo コマンド

sudo コマンドを使うことで、ユーザーに管理アクセスを付与する別のアプローチができます。信頼できるユーザーが **sudo** で管理コマンドに優先する場合、このユーザーは**ユーザー自身の** パスワードを要求されます。ユーザーが認証され、コマンドが許可されると、管理コマンドは root ユーザーであるかのように実行されます。

sudo コマンドの基本的なフォーマットは、以下の通りです。

```
sudo <command>
```

上記の例では、*<command>* の部分は、通常 root ユーザーのみが使用する **mount** といったコマンドに置き換えます。

sudo コマンドでは、ハイレベルの柔軟性が可能になります。例えば、**/etc/sudoers** 設定ファイルに記載されているユーザーのみが **sudo** コマンドを使うことができ、root シェルではなく、**ユーザーの** シェルでコマンドが実行されます。つまり、『Red Hat Enterprise Linux 7 Security Guide』 にあるように、root シェルは完全に無効化できます。

sudo を使った正常な認証はすべて **/var/log/messages** ファイルに記録され、この発行者のユーザー名で発行されたコマンドは、**/var/log/secure** ファイルに記録されます。新たなログが必要な場合は、以下の行を **/etc/pam.d/system-auth** ファイルに追加して、**pam_tty_audit** モジュールで特定ユーザーの TTY 監査を有効にします。

```
session required pam_tty_audit.so disable=<pattern> enable=<pattern>
```

pattern で表示されるのは、コンマで区切ったユーザーのリストで、グロブをオプションで使います。例えば、以下の設定は、root ユーザーのTTY 監査を有効にし、その他のユーザーについては無効にします。

```
session required pam_tty_audit.so disable=* enable=root
```

sudo コマンドのもう一つの利点は、異なるユーザーのニーズに応じて特定のコマンドへのアクセスを管理者が許可できることです。

sudo 設定ファイルである **/etc/sudoers** の編集を希望する管理者は、**visudo** コマンドを使用することが推奨されます。

完全な管理権限を誰かに付与する場合は、**visudo** と入力し、ユーザー権限の指定セクションに以下と同様の行を追加します。

```
juan ALL=(ALL) ALL
```

この例では、ユーザーの **juan** はどのホストからでも **sudo** を実行して、どんなコマンドも実行できます。

以下の例では、**sudo** を設定する際に可能な粒度を示しています。

```
%users localhost=/sbin/shutdown -h now
```

この例が示しているのは、コンソールからであれば、どのユーザーでも **/sbin/shutdown -h now** コマンドを発行でできるということです。

sudoers の man ページにはこのファイルのオプションの詳細なリストが記載されています。

重要

sudo コマンドの使用時には、潜在的なリスクがいくつかあることを覚えておく必要があります。このリスクは、上記のように **visudo** を使用して **/etc/sudoers** 設定ファイルを編集することで回避できます。**/etc/sudoers** ファイルをデフォルトの状態にしておくと、**wheel** グループのユーザー全員に無制限の root アクセスを与えることになります。

- **sudo** はデフォルトで、sudo コマンド使用者のパスワードをタイムアウトの 5 分間、保存します。この間にコマンドを続けて使うと、ユーザーはパスワードを要求されません。このため、ユーザーがログイン状態のままワークステーションを離れたりロックしない状態にしておくと、攻撃者に悪用されかねません。この動作は、以下の行を **/etc/sudoers** に追加することで変更できます。

  ```
  Defaults    timestamp_timeout=<value>
  ```

  <value> には、指定するタイムアウトの分数を入れます。*<value>* を 0 にすると **sudo** は毎回パスワードを要求します。

- sudo 使用者のアカウントが侵害されると、攻撃者は **sudo** を使って管理権限のある新たな shell を開くことができます。

  ```
  sudo /bin/bash
  ```

  この方法や同様の方法で root として新たな shell を開くと、**/etc/sudoers** ファイルで指定されたタイムアウト時間を無視し、新たに開かれたセッションが閉じられるまで攻撃者に **sudo** パスワード入力を要求することがなく、理論上は攻撃者に時間無制限の管理アクセスを与えることになります。

4.3. その他のリソース

ユーザーに管理者権限を与えるプログラムは潜在的なセキュリティーリスクである一方、セキュリティーそのものは本ガイドの対象外となります。セキュリティーや管理者アクセスに関する情報に関しては、以下に挙げるリソースを参照してください。

インストールされているドキュメント

- **su**(1) — **su** の man ページには、このコマンドで利用可能なオプションの情報があります。
- **sudo**(8) — **sudo** の man ページには、このコマンドの動作のカスタマイズで利用可能なオプションのリストがあります。
- **pam**(8) — この man ページでは、 Linux 向け Pluggable Authentication Modules (PAM) の使用方法が説明されています。

オンラインのドキュメント

- Red Hat Enterprise Linux 7 Security Guide — Red Hat Enterprise Linux 7 の 『Security Guide』は、setuid プログラムに関する潜在的なセキュリティー問題の詳しい見方と、これらのリスクを低減するテクニックを提供します。

関連項目

- 3章*ユーザーとグループの管理*では、グラフィカルユーザーインターフェースとコマンドラインを使ったシステムユーザーとグループの管理方法について説明しています。

---

[1] このアクセスは、SELinux の制限が有効な場合は、この制限対象となります。

パート II. パッケージ管理

Red Hat Enterprise Linux システム上のすべてのソフトウェアは、RPM パッケージとして分割されており、インストールやアップグレード、削除が可能です。このセクションでは、**Yum** を使って Red Hat Enterprise Linux でパッケージを管理する方法を説明しています。

第5章 Yum

Yum は、利用可能なパッケージ情報に関するクエリやリポジトリーからのパッケージ取得、パッケージのインストールおよびアンインストール、さらにシステム全体の最新バージョンへの更新などが実行可能な Red Hat パッケージマネージャーです。Yum は、更新、インストール、削除しているパッケージで依存関係の自動解決を実行します。そのため、利用可能な依存パッケージをすべて自動的に決定、取得、およびインストールすることが可能です。

Yum は、新しい追加のリポジトリーまたは *パッケージソース* を使って設定することができ、機能を強化、拡張する多くのプラグインも提供します。また、Yum は **RPM** が実行可能な同じタスクの多くを行うことができます。さらに、多数のコマンドラインオプションも似ています。Yum を使うことで、1 つのマシンまたはマシンのグループ上でのパッケージ管理を簡単かつシンプルに行うことができます。

以下のセクションでは、ご使用のシステムが Red Hat Enterprise Linux 7 インストールガイド にあるように、インストール中に Red Hat サブスクリプション管理に登録されていることを前提としています。詳細情報については、Using and Configuring Red Hat Subscription Manager を参照してください。

重要

Yum は、GPG (Gnu Privacy Guard: 別名 GnuPG) 署名付きパッケージの GPG 署名認証をすべてのパッケージリポジトリー (パッケージソース) または個々のリポジトリーで有効にすることで、セキュアなパッケージ管理を実現します。署名認証が有効になっていると、Yum は正しいキーで GPG 署名されていないパッケージのそのリポジトリへのインストールを拒否します。つまり、使用中のシステムにダウンロードしてインストールする **RPM** パッケージが Red Hat などの信頼されたソースからのものであり、移動中に変更されていないことが確信できます。Yum の署名認証を有効にする方法については、「Yum と Yum リポジトリーの設定」を参照してください。また、GPG 署名付きの **RPM** パッケージの使用および検証方法全般に関しては「パッケージ署名の確認」を参照してください。

Yum を使用すると、他のマシンへダウンロード、インストールするための **RPM** パッケージのリポジトリーを簡単に設定することもできます。可能な場合は、yum は複数パッケージとメタデータの **並行ダウンロード** を使用してダウンロードのスピードを高めます。

システム管理タスクの実行には Yum が最速の方法であることが多いため、これは習得する価値があります。また、Yum は **PackageKit** グラフィカルパッケージ管理ツールが提供する以上の機能を提供します。

注記

yum を使用して、システムにパッケージをインストール、更新、削除するにはスーパーユーザー権限が必要です。本章のすべての例では、**su** または **sudo** コマンドを使用することでスーパーユーザー権限をすでに持っていると仮定しています。

5.1. パッケージの確認と更新

Yum を使用すると、使用中のシステムに適用される更新があるかどうかをチェックできます。更新が必要なパッケージを一覧表示してこれらを一度に更新することや、個別パッケージを選択して更新することができます。

5.1.1. 更新の確認

使用しているシステムに利用可能な更新があるインストール済みのパッケージを確認するには、以下のコマンドを実行します:

```
yum check-update
```

例5.1 yum check-update コマンド出力の例

yum check-update の出力は以下のようになります。

```
~]# yum check-update
Loaded plugins: langpacks, product-id, subscription-manager
Updating Red Hat repositories.
INFO:rhsm-app.repolib:repos updated: 0
PackageKit.x86_64                    0.5.8-2.el6                    rhel
PackageKit-glib.x86_64               0.5.8-2.el6                    rhel
PackageKit-yum.x86_64                0.5.8-2.el6                    rhel
PackageKit-yum-plugin.x86_64         0.5.8-2.el6                    rhel
glibc.x86_64                         2.11.90-20.el6                 rhel
glibc-common.x86_64                  2.10.90-22                     rhel
kernel.x86_64                        2.6.31-14.el6                  rhel
rpm.x86_64                           4.7.1-5.el6                    rhel
rpm-libs.x86_64                      4.7.1-5.el6                    rhel
rpm-python.x86_64                    4.7.1-5.el6                    rhel
yum.noarch                           3.2.24-4.el6                    rhel
```

上記の出力に表示されているパッケージは、利用可能な更新があるものとして表示されています。一覧の最初のパッケージは、**PackageKit** と呼ばれるグラフィカルパッケージマネージャーです。その行の意味は以下のとおりです。

- **PackageKit** — パッケージの名前
- **x86_64** — パッケージがビルドされた CPU アーキテクチャー
- **0.5.8** — インストールされる更新済みパッケージのバージョン
- **rhel** — 更新済みのパッケージがあるリポジトリ

また上記の出力は、**yum** を使ってカーネル (*kernel* パッケージ)、Yum および RPM (*yum* および *rpm* パッケージ)、さらにはそれらの依存関係 (*rpm-libs*、*rpm-python* パッケージ) をすべて更新できることも示しています。

5.1.2. パッケージの更新

1 回に更新するパッケージ数を 1 つ、複数、またはすべてのパッケージから選択することができます。更新するパッケージの依存関係 (またはパッケージ) に利用可能な更新がある場合は、それらも更新されます。

単一パッケージの更新

単一パッケージを更新するには、**root** で以下のコマンドを実行して下さい:

```
yum update package_name
```

例5.2 rpm パッケージの更新

rpm パッケージを更新するには、以下を入力します。

```
~]# yum update rpm
Loaded plugins: langpacks, product-id, subscription-manager
Updating Red Hat repositories.
INFO:rhsm-app.repolib:repos updated: 0
Setting up Update Process
Resolving Dependencies
--> Running transaction check
---> Package rpm.x86_64 0:4.11.1-3.el7 will be updated
--> Processing Dependency: rpm = 4.11.1-3.el7 for package: rpm-libs-
4.11.1-3.el7.x86_64
--> Processing Dependency: rpm = 4.11.1-3.el7 for package: rpm-python-
4.11.1-3.el7.x86_64
--> Processing Dependency: rpm = 4.11.1-3.el7 for package: rpm-build-
4.11.1-3.el7.x86_64
---> Package rpm.x86_64 0:4.11.2-2.el7 will be an update
--> Running transaction check
...
--> Finished Dependency Resolution

Dependencies Resolved
================================================================================
=======
 Package                    Arch        Version         Repository
Size
================================================================================
=======
Updating:
 rpm                        x86_64      4.11.2-2.el7    rhel
1.1 M
Updating for dependencies:
 rpm-build                  x86_64      4.11.2-2.el7    rhel
139 k
 rpm-build-libs             x86_64      4.11.2-2.el7    rhel
98 k
 rpm-libs                   x86_64      4.11.2-2.el7    rhel
261 k
 rpm-python                 x86_64      4.11.2-2.el7    rhel
74 k

Transaction Summary
================================================================================
=======
Upgrade  1 Package (+4 Dependent packages)

Total size: 1.7 M
Is this ok [y/d/N]:
```

この出力には興味を引くアイテムがいくつかあります。

1. **Loaded plugins: langpacks, product-id, subscription-manager** — **yum** は、どの Yum プラグインがインストールされ有効であるかを常に通知します。Yum プラグインに関する一般的情報は「Yum のプラグイン」を参照してください。また、個別のプラグインに関する説明は「yum プラグインの使用方法」を参照してください。

2. **rpm.x86_64** — 新しい *rpm* パッケージとその依存関係をダウンロードしてインストールすることができます。これらの各パッケージに対してトランザクションチェックが行われます。

3. **yum** は更新情報を表示して、更新を実行したいかどうかを確認します。**yum** はデフォルトではインタラクティブに実行します。**yum** コマンドがどのトランザクションを実行するかが事前にわかっている場合は、**-y** オプションを使用して、**yum** が尋ねるすべての質問に **yes** と自動回答するように設定できます (この場合はノンインタラクティブに実行されます)。ただし、**yum** がシステムに実行しようとしている変更を常に調べることをお勧めします。そうすることで、発生可能な問題を簡単に解決することができます。また、パッケージをインストールせずにダウンロードのみをする選択もできます。これを実行するには、ダウンロードプロンプトで **d** オプションを選択します。これにより、選択されたパッケージのバックグラウンドでのダウンロードが開始されます。

   トランザクションが正しく行われなかった場合は、「トランザクション履歴の活用」にあるように **yum history** コマンドを使用して Yum のトランザクション履歴を閲覧することができます。

重要

rpm -i kernel のコマンドを使用すると、**RPM** が新しいカーネルをインストールするのと同じように、**yum** は常に新しいカーネルを**インストール**します。そのため、**yum** を使用する場合は、カーネルパッケージの **インストール** と **アップグレード** の区別について気にする必要はありません。つまり、**yum update** か **yum install** のコマンドどちらを使用するかに関わらず正しく実行します。

一方で **RPM** を使用する場合は、**rpm -u kernel** (現在のカーネルと **置換**) の代わりに、**rpm -i kernel** コマンド (新しいカーネルのインストール) を使用することが重要です。**RPM** を使ったカーネルのインストール、アップグレードについての詳細は、「パッケージのインストールとアップグレード」を参照してください。

同様に、パッケージグループを更新することできます。**root** で以下を入力します。

```
yum group update group_name
```

ここでは、*group_name* を更新するパッケージグループに置き換えます。パッケージグループについての詳細は、「パッケージグループでの作業」を参照してください。

Yum には **upgrade** コマンドもあります。これは、**obsoletes** 設定オプションを有効にした **update** と同様のものです (「[main] オプションの設定」参照)。デフォルトでは、**obsoletes** は **/etc/yum.conf** で on になっており、これによりこの 2 つのコマンドが同等のものになっています。

すべてのパッケージとそれらの依存関係の更新

すべてのパッケージとそれらの依存関係を更新するには、**yum update** と (引数なしで) 入力します。

```
yum update
```

セキュリティー関連パッケージの更新

どのパッケージに利用可能なセキュリティー更新があるのかを見つけ出し、そうしたパッケージを迅速かつ簡単に更新することは重要です。このためのプラグインを Yum は提供します。**security** プラグインは、**yum** コマンドを非常に便利なセキュリティー中心のコマンド、サブコマンド、オプションのセットで拡張します。具体的な情報は「yum プラグインの使用方法」を参照してください。

5.1.3. 設定ファイルの変更の保存

Red Hat Enterprise Linux システムを使用していると、パッケージがインストールした設定ファイルに変更を加える必要がでてきます。システム変更を行うために Yum が使用する **RPM** には、整合性を確保する仕組みが備わっています。パッケージのアップグレード全般における設定ファイルへ変更の管理方法については、「パッケージのインストールとアップグレード」を参照してください。

5.2. パッケージでの作業

Yum では、パッケージの検索、パッケージについての情報の閲覧、パッケージのインストールおよび削除など、ソフトウェアパッケージの完全な操作が可能です。

5.2.1. パッケージの検索

以下のコマンドを使用することで、すべての RPM のパッケージ名、詳細、サマリーを検索することができます。

```
yum search term…
```

このコマンドで、用語に一致する項目の一覧が表示されます。

例5.3 特定の文字列に一致するパッケージの検索

「meld」 や 「kompare」 に一致するパッケージすべてを一覧表示するには、以下を入力します。

```
~]$ yum search meld kompare
Loaded plugins: langpacks, langpacks, product-id, subscription-manager
Updating Red Hat repositories.
INFO:rhsm-app.repolib:repos updated: 0
============================== N/S matched: kompare
==============================
kompare.x86_64 : Diff tool
...

  Name and summary matches mostly, use "search all" for everything.
Warning: No matches found for: meld
```

yum search コマンドは、パッケージ名は分からないものの、関連用語を知っている場合にパッケージ検索をする際に役立ちます。デフォルトでは、**yum search** はパッケージ名とサマリーを返すことで、検索が速まります。より包括的な検索には **yum search all** を使用しますが、速度は遅くなります。

結果をフィルターする

Yum の一覧 コマンドではすべて、1 つ以上の *glob 表現* を引数として追加して結果をフィルターすることができます。glob 表現は、1 つ以上のワイルドカード文字 ***** (任意の文字サブセットに拡張) と **?** (任意の 1 文字に拡張) を含む通常の文字列です。

yum コマンドに glob 表現を引数として渡す場合には、glob 表現をエスケープするよう注意してください。これを行わないと、bash シェルはこの表現を *パス名の展開* と解釈してしまい、glob と適合する現在のディレクトリー内の全ファイルを **yum** に渡すおそれがあります。確実に glob 表現を**yum** に渡すには、以下のいずれかの方法で行います。

- ワイルドカード文字の前にバックスラッシュ記号を入力して、ワイルドカード文字をエスケープする
- glob 表現全体を二重引用符または単一引用符でくくる

以下のセクションの例では、上記の両方の使用例が説明されます。

5.2.2. パッケージの一覧表示

すべてのインストール済み **および** 利用可能なパッケージに関する情報を一覧表示するには、シェルプロンプトで以下を入力します。

```
yum list all
```

glob 表現に一致するインストール済み **および** 利用可能なパッケージを一覧表示するには、以下のコマンドを使用します。

```
yum list glob_expression…
```

例5.4 ABRT 関連パッケージの一覧

各種の ABRT アドオンとプラグインを持つパッケージは 「abrt-addon-」 か 「abrt-plugin-」 で始まります。これらのパッケージを一覧表示するには、シェルプロンプトで以下を入力します。ワイルドカード文字の前にバックスラッシュ文字を置くことでエスケープしていることに注意してください。

```
~]$ yum list abrt-addon\* abrt-plugin\*
Loaded plugins: langpacks, product-id, subscription-manager
Updating Red Hat repositories.
INFO:rhsm-app.repolib:repos updated: 0
Installed Packages
abrt-addon-ccpp.x86_64                          1.0.7-5.el6
@rhel
abrt-addon-kerneloops.x86_64                    1.0.7-5.el6
@rhel
abrt-addon-python.x86_64                        1.0.7-5.el6
@rhel
abrt-plugin-bugzilla.x86_64                     1.0.7-5.el6
@rhel
abrt-plugin-logger.x86_64                       1.0.7-5.el6
@rhel
abrt-plugin-sosreport.x86_64                    1.0.7-5.el6
@rhel
abrt-plugin-ticketuploader.x86_64               1.0.7-5.el6
@rhel
```

システムにインストール済みのパッケージすべてを一覧表示するには、**installed** キーワードを使用し

ます。出力の右端の列には、パッケージが取得されたリポジトリーが表示されます。

```
yum list installed glob_expression…
```

例5.5 インストール済み krb パッケージの一覧表示

以下の例では、「krb」 で始まり、その後に正確に 1 文字とハイフンが続くインストール済みパッケージすべてを一覧表示する方法を示しています。この方法では数字でバージョンが見分けられるので、特定コンポーネントの全バージョンを一覧表示したい場合に便利です。glob 表現全体を引用符で囲むことで適切な処理が確実になります。

```
~]$ yum list installed "krb?-*"
Loaded plugins: langpacks, product-id, subscription-manager
Updating Red Hat repositories.
INFO:rhsm-app.repolib:repos updated: 0
Installed Packages
krb5-libs.x86_64                          1.8.1-3.el6
@rhel
krb5-workstation.x86_64                   1.8.1-3.el6
@rhel
```

有効なリポジトリーすべてでインストール可能なパッケージを一覧表示するには、以下の形式のコマンドを使用します。

```
yum list available glob_expression…
```

例5.6 利用可能な gstreamer プラグインの一覧表示

たとえば、「gstreamer」 とその後に「plugin」 を含む名前の利用可能なパッケージすべてを一覧表示するには、以下のコマンドを実行します。

```
~]$ yum list available gstreamer\*plugin\*
Loaded plugins: langpacks, product-id, subscription-manager
Updating Red Hat repositories.
INFO:rhsm-app.repolib:repos updated: 0
Available Packages
gstreamer-plugins-bad-free.i686               0.10.17-4.el6
rhel
gstreamer-plugins-base.i686                   0.10.26-1.el6
rhel
gstreamer-plugins-base-devel.i686             0.10.26-1.el6
rhel
gstreamer-plugins-base-devel.x86_64           0.10.26-1.el6
rhel
gstreamer-plugins-good.i686                   0.10.18-1.el6
rhel
```

リポジトリーの一覧表示

リポジトリーの ID、名前、使用中のシステム上で **有効な** 各リポジトリーでのパッケージ数を一覧表示するには、以下のコマンドを実行します。

```
yum repolist
```

これらのリポジトリーについての詳細情報を一覧表示するには、**-v** オプションを追加します。このオプションを有効にすると、各リポジトリーでファイル名や全体のサイズ、最終更新日、ベース URL といった情報が表示されます。別の方法としては、**repoinfo** コマンドを使って同じ出力を作成することもできます。

```
yum repolist -v
```

```
yum repoinfo
```

有効および無効なリポジトリーの両方を表示するには、以下のコマンドを使用します。ステータスのコラムが出力リストに追加され、どのリポジトリーが有効になっているかが分かります。

```
yum repolist all
```

disabled を最初の引数として渡すことで、コマンドの出力を無効なリポジトリーに制限できます。詳細な仕様については、リポジトリーの ID や名前、関連する glob 表現を引数として渡すことができます。引数と完全に一致するリポジトリー ID または名前がある場合は、**enabled** または **disabled** フィルターを通過しないリポジトリーであっても表示されることに注意してください。

5.2.3. パッケージ情報の表示

1 つ以上のパッケージに関する情報を表示するには、以下のコマンドを実行します (ここでは glob 表現も有効)。

```
yum info package_name…
```

package_name をパッケージ名に置き換えます。

例5.7 abrt パッケージの情報を表示する

abrt パッケージに関する情報を表示するには、以下を入力します。

```
~]$ yum info abrt
Loaded plugins: langpacks, product-id, subscription-manager
Updating Red Hat repositories.
INFO:rhsm-app.repolib:repos updated: 0
Installed Packages
Name        : abrt
Arch        : x86_64
Version     : 1.0.7
Release     : 5.el6
Size        : 578 k
Repo        : installed
From repo   : rhel
Summary     : Automatic bug detection and reporting tool
URL         : https://fedorahosted.org/abrt/
License     : GPLv2+
```

```
Description: abrt is a tool to help users to detect defects in
applications
           : and to create a bug report with all informations needed
by
           : maintainer to fix it. It uses plugin system to extend its
           : functionality.
```

yum info ***package_name*** コマンドは **rpm -q --info** ***package_name*** コマンドに似ていますが、前者は RPM パッケージがある Yum リポジトリー ID を追加情報として提供します (出力の **From repo :** の行を参照)。

yumdb の使用

以下のコマンドを使用することで、パッケージに関する代替情報や有用な情報について Yum データベースにクエリすることもできます。

```
yumdb info package_name
```

このコマンドは、パッケージのチェックサム (および SHA-256 などのチェックサムを算出するためのアルゴリズム)、パッケージのインストール開始に使われたコマンドラインのコマンド (存在する場合)、パッケージがシステムにインストールされた理由 (**user** はユーザーがインストールしたことを、**dep** は依存関係として取り入れたことを意味します) などのパッケージに関する追加情報を提供します。

例5.8 yum パッケージに関する情報を yumdb でクエリする

yum パッケージに関する追加情報を表示するには、以下を入力します。

```
~]$ yumdb info yum
Loaded plugins: langpacks, product-id, subscription-manager
yum-3.2.27-4.el6.noarch
     checksum_data =
23d337ed51a9757bbfbdceb82c4eaca9808ff1009b51e9626d540f44fe95f771
     checksum_type = sha256
     from_repo = rhel
     from_repo_revision = 1298613159
     from_repo_timestamp = 1298614288
     installed_by = 4294967295
     reason = user
     releasever = 6.1
```

yumdb コマンドの詳細については、**yumdb**(8) の man ページを参照してください。

5.2.4. パッケージのインストール

単一パッケージおよびそのパッケージのインストールされていない依存関係すべてをインストールするには、(**root**で) 以下の形式でコマンドを入力します。

```
yum install package_name
```

複数パッケージを同時にインストールするには、引数としてその名前を追加します。**root** で以下を入力します。

```
yum install package_name package_name…
```

AMD64 や Intel64 マシンのような *multilib* システムにパッケージをインストールする場合は、パッケージ名に *.arch* を追加することでパッケージのアーキテクチャーを指定できます (ただし、有効なリポジトリーで利用可能な場合のみ)。

```
yum install package_name.arch
```

例5.9 multilib システムでのパッケージのインストール

i686 アーキテクチャー用の *sqlite* パッケージをインストールするには、以下を入力します。

```
~]# yum install sqlite.i686
```

glob 表現を使うと、複数の似通った名前のパッケージを迅速にインストールできます。**root** で以下を入力します。

```
yum install glob_expression…
```

例5.10 すべての audacious プラグインをインストールする

似通った名前の複数のパッケージをインストールするには、glob 表現が便利です。すべての audacious プラグインをインストールするには、以下の形式でコマンドを使用します。

```
~]# yum install audacious-plugins-\*
```

パッケージ名と glob 表現に加えて、**yum install** にはファイル名も追加することができます。インストールするバイナリ名が分かっていて、パッケージ名が分からない場合は、**yum install** にパス名を付けて実行します。**root** で以下を入力します。

```
yum install /usr/sbin/named
```

yum はパッケージ一覧で検索を行い、**/usr/sbin/named** を提供するパッケージを探します。存在すれば、yum はインストールするかどうかを確認します。

上記の例で分かるように、**yum install** コマンドは厳密に定義された引数を必要としません。さまざまな形式のパッケージ名や glob 表現を処理できるので、ユーザーによるインストールを容易にします。一方で、**yum** が入力を正確に分析するにはしばらく時間がかかります。多数のパッケージを指定した場合は、特にそうです。パッケージ検索を最適化するには、以下のコマンドを使って引数の分析方法を明示的に定義できます。

```
yum install-n name
```

```
yum install-na name.architecture
```

```
yum install-nevra name-epoch:version-release.architecture
```

install-n では、**yum** は *name* を正確なパッケージ名として判断します。**install-na** コマンドは **yum** に、その後に続く引数にピリオドで分けられたパッケージ名とアーキテクチャーが含まれていること

を伝えます。**install-nevra** では、**yum** は引数が *name-epoch:version-release.architecture* の形式になっていることを予想します。同様に、**yum remove-n**、**yum remove-na**、および **yum remove-nevra** を使って削除するパッケージを検索することもできます。

注記

named バイナリを含むパッケージをインストールしたいものの、ファイルがインストールされているのが **bin** ディレクトリーか **sbin** ディレクトリーか分からない場合は、glob 表現を付けて **yum provides** コマンドを実行します。

```
~]# yum provides "*bin/named"
Loaded plugins: langpacks, product-id, subscription-manager
Updating Red Hat repositories.
INFO:rhsm-app.repolib:repos updated: 0
32:bind-9.7.0-4.P1.el6.x86_64 : The Berkeley Internet Name Domain
(BIND)
                              : DNS (Domain Name System) server
Repo        : rhel
Matched from:
Filename    : /usr/sbin/named
```

yum provides "*/*file_name*" は *file_name* を含むパッケージを見つけるための一般的で便利な方法です。

例5.11 インストールプロセス

以下の例では、**yum** 使ったインストールの概要を示しています。最新バージョンの*httpd* パッケージをダウンロードしてインストールするとします。これには、**root** で以下を実行します。

```
~]# yum install httpd
Loaded plugins: langpacks, product-id, subscription-manager
Resolving Dependencies
--> Running transaction check
---> Package httpd.x86_64 0:2.4.6-12.el7 will be updated
---> Package httpd.x86_64 0:2.4.6-13.el7 will be an update
--> Processing Dependency: 2.4.6-13.el7 for package: httpd-2.4.6-
13.el7.x86_64
--> Running transaction check
---> Package httpd-tools.x86_64 0:2.4.6-12.el7 will be updated
---> Package httpd-tools.x86_64 0:2.4.6-13.el7 will be an update
--> Finished Dependency Resolution

Dependencies Resolved
```

上記のコマンドを実行した後、**yum** は必要なプラグインを読み込み、トランザクションチェックを実行します。このケースでは、*httpd* は既にインストールされています。インストール済みのパッケージが利用可能な最新バージョンよりも古いことから、これは更新されます。*httpd* が依存する *httpd-tools* にも同様のことが行われます。すると、トランザクションサマリーは以下のように表示されます。

```
================================================================================
```

```
==========
 Package        Arch      Version                  Repository
Size
================================================================================
==========
Updating:
 httpd          x86_64    2.4.6-13.el7             rhel-x86_64-server-7
1.2 M
Updating for dependencies:
 httpd-tools    x86_64    2.4.6-13.el7             rhel-x86_64-server-7
77 k

Transaction Summary
================================================================================
==========
Upgrade  1 Package (+1 Dependent package)

Total size: 1.2 M
Is this ok [y/d/N]:
```

このステップでは、**yum** がインストールを確認するプロンプトを表示します。**y** (はい) または **N** (いいえ) オプションの他に、**d** (ダウンロードのみ) を選択すると、パッケージのダウンロードはしますが、直接インストールはしません。**y** を選択すると、以下のメッセージが出て、インストールが正常に完了するまで続行します。

```
Downloading packages:
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Updating   : httpd-tools-2.4.6-13.el7.x86_64
1/4
  Updating   : httpd-2.4.6-13.el7.x86_64
2/4
  Cleanup    : httpd-2.4.6-12.el7.x86_64
3/4
  Cleanup    : httpd-tools-2.4.6-12.el7.x86_64
4/4
  Verifying  : httpd-2.4.6-13.el7.x86_64
1/4
  Verifying  : httpd-tools-2.4.6-13.el7.x86_64
2/4
  Verifying  : httpd-tools-2.4.6-12.el7.x86_64
3/4
  Verifying  : httpd-2.4.6-12.el7.x86_64
4/4

Updated:
  httpd.x86_64 0:2.4.6-13.el7

Dependency Updated:
  httpd-tools.x86_64 0:2.4.6-13.el7

Complete!
```

システム上のローカルディレクトリーから既にダウンロード済みのパッケージをインストールするには、以下のコマンドを使用します。

```
yum localinstall path
```

path をインストールするパッケージへのパスに置き換えます。

5.2.5. パッケージのダウンロード

例5.11「インストールプロセス」にあるように、インストールプロセスのある時点で、インストールの確認を尋ねる以下のメッセージが表示されます。

```
...
Total size: 1.2 M
Is this ok [y/d/N]:
...
```

d オプションを選択すると、**yum** にパッケージをすぐにインストールすることなくダウンロードするように命じることになります。このようなパッケージは、後でオフラインで **yum localinstall** コマンドを使ってインストールするか、別のデバイスと共有することができます。ダウンロードされたパッケージはキャッシュディレクトリーのサブディレクトリーの 1 つに保存され、デフォルトでは **/var/cache/yum/$basearch/$releasever/packages/** ディレクトリーになります。ダウンロードはバックグラウンドで進行するので、並行して **yum** を他の操作に使うことができます。

5.2.6. パッケージの削除

パッケージのインストールと同様に、yum を使うとパッケージのアンインストールができます (**RPM** および yum 用語では、remove と呼びます)。特定のパッケージとそれに依存するすべてのパッケージをアンインストールするには、**root** で以下のコマンドを実行します。

```
yum remove package_name…
```

複数のパッケージをインストールする場合と同様に、コマンドに複数のパッケージ名を追加することで一度に複数のパッケージを削除することができます。

> **例5.12 複数パッケージの削除**
>
> *totem* を削除するには、シェルプロンプトで以下を入力します。
>
> ```
> ~]# yum remove totem
> ```

install と同じように、**remove** は以下の引数を取ることができます。

- パッケージ名
- glob 表現
- ファイル一覧
- パッケージが提供する機能

警告

Yum では、パッケージに依存しているパッケージを削除せずにそのパッケージを削除することはできません。こうした動作は **RPM** でのみ実行可能であり、推奨されません。システムが機能しなくなる、またはアプリケーションに誤作動やクラッシュが生じる恐れがあるためです。詳細は、**RPM** の章の「パッケージのアンインストール」を参照して下さい。

5.3. パッケージグループでの作業

パッケージグループは、たとえば **システムツール** や **サウンドとビデオ** などの共通の目的でサービスを行うパッケージの集合です。パッケージグループをインストールすると、依存パッケージのセットもプルされるので時間が大幅に短縮できます。**yum groups** コマンドは、yum 内のパッケージグループに作用するすべての操作をカバーするトップレベルのコマンドです。

5.3.1. パッケージグループの一覧表示

summary オプションを使うと、インストール済みのグループ数、利用可能なグループ数、利用可能な環境グループ数、インストール済みの言語グループ数、利用可能な言語グループ数が表示されます。

```
yum groups summary
```

例5.13 yum groups summary の出力例

```
~]$ yum groups summary
Loaded plugins: langpacks, product-id, subscription-manager
Available Environment Groups: 12
Installed Groups: 10
Available Groups: 12
```

yum リポジトリーからすべてのパッケージグループを一覧表示するには、**list** オプションを追加します。コマンドの出力は、グループ名ごとにフィルターすることができます。

```
yum group list glob_expression…
```

このコマンドでは渡すことのできるオプションの引数がいくつかあります。**hidden** はユーザー表示可能とマークされていないグループも一覧表示し、**ids** はグループ ID を表示します。また、**language**、**environment**、**installed**、**available** などのオプションを追加して、出力を特定のグループタイプに制限することもできます。

特定のグループに含まれている必須およびオプションパッケージを一覧表示するには、以下のコマンドを使用します。

```
yum group info glob_expression…
```

例5.14 LibreOffice パッケージグループの情報を表示する

```
 ~]$ yum group info LibreOffice
Loaded plugins: langpacks, product-id, subscription-manager
```

```
Group: LibreOffice
 Group-Id: libreoffice
 Description: LibreOffice Productivity Suite
 Mandatory Packages:
  =libreoffice-calc
   libreoffice-draw
  -libreoffice-emailmerge
   libreoffice-graphicfilter
  =libreoffice-impress
  =libreoffice-math
  =libreoffice-writer
  +libreoffice-xsltfilter
 Optional Packages:
   libreoffice-base
   libreoffice-pyuno
```

上記の例で分かるように、このパッケージグループに含まれているパッケージは、以下の記号でマークされている状態に分けられます。

- " **-** " — パッケージはインストールされておらず、パッケージグループの一部としてインストールされません。
- " **+** " — パッケージはインストールされていませんが、次回の**yum upgrade** または **yum group upgrade** でインストールされます。
- " **=** " — パッケージはインストールされており、パッケージグループの一部としてインストールされました。
- **記号なし** — パッケージはインストールされていますが、パッケージグループ外でインストールされました。このため、**yum group remove** でこのパッケージを削除することはできません。

これらの区別は、**group_command** 設定パラメーターがデフォルト設定の**objects** になっている場合にのみ発生します。yum でパッケージがグループの一部としてインストールされたかまたは別個にインストールされたかを追跡したくない場合は、このパラメーターを異なる値に設定します。すると、"**記号なし**" パッケージと "**=**" パッケージが同じ意味になります。

yum group mark コマンドを使って上記のパッケージの状態を変更することもできます。たとえば、**yum group mark packages** は、特定のインストール済みパッケージを指定されたグループのメンバーとしてマークします。グループ更新で新たなパッケージのインストールを避けるには、**yum group mark blacklist** を使います。**yum group mark** の詳細な機能については、**yum** man ページを参照してください。

注記

@^ 接頭辞を使うと環境グループが特定でき、パッケージグループには@ のマークが付きます。**yum group list**、**info**、**install**、または **remove** を使用する際は、**@group_name** を渡すとパッケージグループが、**@^group_name** を渡すと環境グループが指定され、**group_name** を渡すとこれら両方を含めることができます。

5.3.2. パッケージグループのインストール

パッケージグループにはそれぞれ、名前と *groupid* があります。全パッケージグループの名前とそれらの groupid (括弧内に表示される) を一覧表示するには、以下を入力します。

```
yum group list ids
```

例5.15 パッケージグループの名前と groupid の表示

KDE デスクトップ環境に関連するパッケージグループをインストールする際に、そのパッケージグループの正確な名前や id を思い出せないとします。これらの情報を見つけるには、以下を入力します。

```
~]$ yum group list ids kde\*
Loaded plugins: langpacks, product-id, subscription-manager
Available Groups:
   KDE Desktop (kde-desktop)
Done
```

パッケージグループをインストールするには、**groupinstall** にグループ名全体 (groupid を含めない) を渡します。**root** で以下を入力します。

```
yum group install group_name
```

groupid でインストールすることもできます。**root** で以下のコマンドを実行します。

```
yum group install groupid
```

groupid または引用符で囲んだグループ名を@ マークに続けて、**install** コマンドに渡すことも可能です。これで **group install** を実行したいというメッセージが**yum** に伝わります。**root** で以下を入力します。

```
yum install @group
```

group を groupid または引用符で囲んだグループ名で置き換えます。同様の論理が環境グループにも適用できます。

```
yum install @^group
```

例5.16 KDE Desktop グループをインストールする 4 つの方法

上記で説明したように、パッケージグループをインストールするには、4 つの異なる、ただし同等の方法があります。KDE Desktop の場合、コマンドは以下のようになります。

```
~]# yum group install "KDE Desktop"
~]# yum group install kde-desktop
~]# yum install @"KDE Desktop"
~]# yum install @kde-desktop
```

5.3.3. パッケージグループの削除

パッケージグループ名もしくはその ID を使って、**install** 構文と一致する構文を使用してパッケージグループを削除することができます。**root** で以下を入力します。

```
yum group remove group_name
```

```
yum group remove groupid
```

また、groupid または引用符で囲んだグループ名を@ マークに続けて、**remove** コマンドに渡すことも可能です。これで **group remove** を実行したいというメッセージが**yum** に伝わります。**root** で以下を入力します。

```
yum remove @group
```

group を groupid または引用符で囲んだグループ名で置き換えます。同様に、環境グループに置き換えることもできます。

```
yum remove @^group
```

例5.17 KDE Desktop グループを削除する 4 つの方法

インストールする場合と同様に、パッケージグループを削除するには、4 つの異なる、ただし同等の方法があります。KDE Desktop の場合、コマンドは以下のようになります。

```
~]# yum group remove "KDE Desktop"
~]# yum group remove kde-desktop
~]# yum remove @"KDE Desktop"
~]# yum remove @kde-desktop
```

5.4. トランザクション履歴の活用

yum history コマンドを使用すると、Yum のトランザクションのタイムライン、トランザクションの発生日時、影響を受けたパッケージ数、トランザクション成功の有無、RPM データベースがトランザクション間で変更されたかどうかといった情報を確認することができます。さらに、このコマンドを使うと、特定のトランザクションを元に戻す、やり直すことが可能です。すべての履歴データは、**/var/lib/yum/history/** ディレクトリーの **history DB** に保存されます。

5.4.1. トランザクションの一覧表示

最近発生した 20 件のトランザクションを一覧表示するには、 **root** で引数なしで**yum history** を実行するか、シェルプロンプトで以下を入力します。

```
yum history list
```

すべてのトランザクションを表示するには、**all** のキーワードを追加します。

```
yum history list all
```

特定の範囲内のトランザクションのみを表示したい場合は、以下の形式でコマンドを使用します。

```
yum history list start_id..end_id
```

特定のパッケージに関するトランザクションのみを一覧表示することも可能です。そのためには、パッケージ名か glob 表現を付けてコマンドを実行します。

```
yum history list glob_expression…
```

例5.18 最も古いトランザクション 5 件を表示する

yum history list の出力では、最新のトランザクションがリストの上部に表示されます。履歴データベースにある最も古い 5 件のトランザクションに関する情報を表示するには、以下を入力します。

```
~]# yum history list 1..5
Loaded plugins: langpacks, product-id, subscription-manager
ID     | Login user               | Date and time    | Action(s)
| Altered
-------------------------------------------------------------------------
---------
     5 | Jaromir ... <jhradilek>  | 2013-07-29 15:33 | Install        |
1
     4 | Jaromir ... <jhradilek>  | 2013-07-21 15:10 | Install        |
1
     3 | Jaromir ... <jhradilek>  | 2013-07-16 15:27 | I, U           |
73
     2 | System <unset>           | 2013-07-16 15:19 | Update         |
1
     1 | System <unset>           | 2013-07-16 14:38 | Install        |
1106
history list
```

yum history list コマンドはすべての形式で、以下のコラムで構成される各行を含む表形式出力を生成します。

- **ID** — 特定のトランザクションを識別する整数値です。
- **Login user** — ユーザー名で、この名前のログインセッションを使ってトランザクションが開始されました。この情報は、通常 ***Full Name <username>*** の形式で表示されます。ユーザーが実行しなかったトランザクションに関しては (システムの自動更新など)、代わりに **System <unset>** が使用されます。
- **Date and time** — トランザクションが発生した日時です。
- **Action(s)** — 表5.1「Action フィールドの値」 の説明のとおり、トランザクション中に実行された動作の一覧です。
- **Altered** — 表5.2「Altered フィールドの値」 の説明のとおり、トランザクションにより影響を受けたパッケージ数、場合によっては追加情報も含まれます。

表5.1 Action フィールドの値

Action	省略形	詳細
Downgrade	**D**	1 つ以上のパッケージが旧バージョンにダウングレードされました。
Erase	**E**	1 つ以上のパッケージが削除されました。
Install	**I**	1 つ以上の新しいパッケージがインストールされました。
Obsoleting	**O**	1 つ以上のパッケージが廃止として記録されました。

Action	省略形	詳細
Reinstall	**R**	1 つ以上のパッケージが再インストールされました。
Update	**U**	1 つ以上のパッケージが新しいバージョンに更新されました。

表5.2 Altered フィールドの値

記号	詳細
<	トランザクションが終了する前に、**rpmdb** データベースが yum 以外で変更されました。
>	トランザクションが終了した後に、**rpmdb** データベースが yum 以外で変更されました。
*****	トランザクションは失敗して終了しました。
#	トランザクションは正常に終了しましたが、**yum** はゼロ以外の終了コードを返しました。
E	トランザクションは正常に終了しましたが、エラーまたは警告が表示されました。
P	トランザクションは正常に終了しましたが、**rpmdb** データベースに問題がすでに存在していました。
s	トランザクションは正常に終了しましたが、**--skip-broken** コマンドラインオプションが使用され、特定のパッケージが省略されました。

インストール済みパッケージの **rpmdb** または **yumdb** データベースのコンテンツを、現在使用されている **rpmdb** または **yumdb** データベースと同期させるには、以下を入力します。

```
yum history sync
```

現在使用している履歴データベースについての全体的な統計数字を表示するには、以下のコマンドを使用します。

```
yum history stats
```

例5.19 yum history stats の出力例

```
~]# yum history stats
Loaded plugins: langpacks, product-id, subscription-manager
File        : //var/lib/yum/history/history-2012-08-15.sqlite
Size        : 2,766,848
Transactions: 41
Begin time  : Wed Aug 15 16:18:25 2012
End time    : Wed Feb 27 14:52:30 2013
Counts      :
  NEVRAC :  2,204
  NEVRA  :  2,204
  NA     :  1,759
  NEVR   :  2,204
  rpm DB :  2,204
  yum DB :  2,204
history stats
```

Yum を使用すると、過去に発生したすべてのトランザクションのサマリーを表示することもできます。**root** で以下の形式のコマンドを実行します。

```
yum history summary
```

特定の範囲内でのトランザクションのみを表示するには、以下を入力します。

```
yum history summary start_id..end_id
```

yum history list コマンドと同様に、パッケージの名前または glob 表現を指定することで、特定のパッケージに関するトランザクションのサマリーを表示することできます。

```
yum history summary glob_expression…
```

例5.20 最新のトランザクション 5 件のサマリー

```
~]# yum history summary 1..5
Loaded plugins: langpacks, product-id, subscription-manager
Login user                | Time                | Action(s)        |
Altered
------------------------------------------------------------------------
---------
Jaromir ... <jhradilek>   | Last day            | Install          |
1
Jaromir ... <jhradilek>   | Last week           | Install          |
1
Jaromir ... <jhradilek>   | Last 2 weeks        | I, U             |
73
System <unset>            | Last 2 weeks        | I, U             |
1107
history summary
```

yum history summary コマンドはすべての形式で、**yum history list** の出力に似た、簡略化された表形式出力を生成します。

上記のように、**yum history list** および **yum history summary** とも、トランザクション向けに設定されています。特定のパッケージに関連するトランザクションのみを表示することができますが、パッケージバージョンのような重要な詳細は表示されません。パッケージに関連するトランザクションを一覧表示するには、**root** で以下のコマンドを実行します。

```
yum history package-list glob_expression…
```

例5.21 パッケージ履歴の追跡

たとえば、*subscription-manager* および関連パッケージの履歴を調べるには、シェルプロンプトで以下を入力します。

```
~]# yum history package-list subscription-manager\*
Loaded plugins: langpacks, product-id, subscription-manager
ID     | Action(s)      | Package
------------------------------------------------------------------------
---------
     3 | Updated        | subscription-manager-1.10.11-1.el6.x86_64
     3 | Update         |                       1.10.17-1.el6_1.x86_64
```

```
     3 | Updated         | subscription-manager-firstboot-1.10.11-
1.el6.x86_64
     3 | Update          |                                  1.10.17-
1.el6_1.x86_64
     3 | Updated         | subscription-manager-gui-1.10.11-1.el6.x86_64
     3 | Update          |                                  1.10.17-
1.el6_1.x86_64
     1 | Install         | subscription-manager-1.10.11-1.el6.x86_64
     1 | Install         | subscription-manager-firstboot-1.10.11-
1.el6.x86_64
     1 | Install         | subscription-manager-gui-1.10.11-1.el6.x86_64
history package-list
```

上記の例では、初期のシステムインストール時に *subscription-manager*、*subscription-manager-firstboot*、*subscription-manager-gui* の 3 パッケージがインストールされています。3 つ目のトランザクションでは、これらの全パッケージはバージョン 0.95.11 から 0.95.17 に更新されています。

5.4.2. トランザクションの検証

単一のトランザクションのサマリーを表示するには、**root** で以下の形式で **yum history summary** コマンドを使用します。

```
yum history summary id
```

特定のトランザクションを詳しく調べたい場合は、**root** で以下のコマンドを実行します。

```
yum history info id…
```

id の引数はオプションです。これを省略する場合は、**yum** は自動的に最後のトランザクションを使用します。複数のトランザクションを指定する場合は、範囲を指定することもできます。

```
yum history info start_id..end_id
```

例5.22 yum history info の出力例

以下は、2 つのトランザクションに関する出力のサンプルです。それぞれ新しいパッケージを 1 つインストールしています。

```
~]# yum history info 4..5
Loaded plugins: langpacks, product-id, subscription-manager
Transaction ID : 4..5
Begin time     : Thu Jul 21 15:10:46 2011
Begin rpmdb    : 1107:0c67c32219c199f92ed8da7572b4c6df64eacd3a
End time       :            15:33:15 2011 (22 minutes)
End rpmdb      : 1109:1171025bd9b6b5f8db30d063598f590f1c1f3242
User           : Jaromir Hradilek <jhradilek>
Return-Code    : Success
Command Line   : install screen
Command Line   : install yum-plugin-aliases
Transaction performed with:
    Installed     rpm-4.8.0-16.el6.x86_64
    Installed     yum-3.2.29-17.el6.noarch
```

```
    Installed     yum-metadata-parser-1.1.2-16.el6.x86_64
Packages Altered:
    Install screen-4.0.3-16.el6.x86_64
    Install yum-plugin-aliases-1.1.30-6.el6.noarch
history info
```

また、トランザクション時に使用された設定オプション、特定のパッケージをインストールしたリポジトリー、およびその理由などの追加情報も閲覧できます。特定のトランザクションに関して入手可能な追加情報を知りたい場合は、**root** としてシェルプロンプトで以下を入力します。

```
yum history addon-info id
```

yum history info と同様に、*id* が指定されていない場合は、**yum** は自動的に最新のトランザクションを使用します。別の方法として、最新のトランザクションを参照するには、**last** キーワードを使用することもできます。

```
yum history addon-info last
```

例5.23 yum history addon-info の出力例

履歴の 4 番目のトランザクションには、**yum history addon-info** は以下の出力を返します。

```
~]# yum history addon-info 4
Loaded plugins: langpacks, product-id, subscription-manager
Transaction ID: 4
Available additional history information:
  config-main
  config-repos
  saved_tx

history addon-info
```

yum history addon-info コマンドの出力では、以下の 3 種類の情報が表示されます。

- **config-main** — トランザクション時に使用された yum のグローバルオプションです。グローバルオプションの変更方法の詳細は、「[main] オプションの設定」 を参照してください。
- **config-repos** — 個々の yum リポジトリー用のオプションです。個々のリポジトリー用のオプションを変更する方法については、「[repository] オプションの設定」 を参照してください。
- **saved_tx** — 別のマシンでトランザクションを繰り返すために**yum load-transaction** コマンドにより利用できるデータです (下記参照)。

選択した種類の追加情報を表示するには、**root** で以下のコマンドを実行してください。

```
yum history addon-info id information
```

5.4.3. トランザクションを元に戻す、繰り返す方法

トランザクション履歴の確認以外に、**yum history** コマンドは選択したトランザクションを元に戻す、繰り返す方法を提供します。トランザクションを元に戻すには、**root** としてシェルプロンプトで以下を入力します。

```
yum history undo id
```

特定のトランザクションを繰り返すには、**root** で以下のコマンドを実行します。

```
yum history redo id
```

どちらのコマンドでも **last** キーワードを使用して、最新のトランザクションを元に戻す、または繰り返すことができます。

yum history undo および **yum history redo** コマンドのどちらも、トランザクション中に実行されたステップを元に戻すまたは繰り返すだけである点に注意してください。トランザクションにより新しいパッケージがインストールされた場合、**yum history undo** コマンドはそれをアンインストールします。トランザクションによりパッケージがアンインストールされた場合、このコマンドは再度インストールします。またこのコマンドは、すべての更新済みパッケージを以前のバージョンにダウングレードする試みも実行します (古いパッケージが引き続き利用可能な場合)。

複数の同一システムを管理する場合、yum を使用すると、1 つのシステム上でトランザクションを実行して、そのトランザクションの詳細をファイルに格納し、テスト期間の終了後に残りのシステム上で同じトランザクションを繰り返すことができます。トランザクションの詳細をファイルに格納するには、**root** でシェルプロンプトに以下を入力します。

```
yum -q history addon-info id saved_tx > file_name
```

このファイルを目的のシステムにコピーしたら、**root** で以下のコマンドを使用してトランザクションを繰り返すことができます。

```
yum load-transaction file_name
```

欠けているパッケージや rpmdb バージョンを無視するように **load-transaction** を設定することも可能です。これらの設定オプションについての詳細は、yum.conf の man ページを参照してください。

5.4.4. 新しいトランザクション履歴の開始

Yum は単一の SQLite データベースファイルにトランザクション履歴を格納します。新しいトランザクションの履歴を開始するには、**root** で以下のコマンドを実行します。

```
yum history new
```

これにより **/var/lib/yum/history/** ディレクトリー内に新しい空のデータベースファイルが作成されます。古いトランザクション履歴は保存されますが、新しいデータベースファイルがディレクトリーにある限りアクセスすることはできません。

5.5. Yum と Yum リポジトリーの設定

yum および関連ユーティリティーの設定ファイルは **/etc/yum.conf** になります。このファイルには、必須の **[main]** セクションが 1 つあり、ここでは全体に影響を与える Yum オプションを設定できます。また、1 つ以上の **[*repository*]** セクションを含めることも可能で、リポジトリー固有のオプションを設定できます。ただし、**/etc/yum.repos.d/** ディレクトリー内にある、新規または既存の **.repo** ファイルで個々のリポジトリーを定義することが推奨されます。**/etc/yum.conf** ファイルの **[main]** セクションで定義する値は、個々の **[*repository*]** セクションで設定された値を上書きすることが可能です。

このセクションでは以下の方法を紹介します。

- **/etc/yum.conf** 設定ファイルの **[main]** セクションを編集して yum のグローバルオプションを設定する方法
- **/etc/yum.conf** の **[*repository*]** セクションと **/etc/yum.repos.d/** ディレクトリー内の **.repo** ファイルを編集することで、個々のリポジトリーのオプションを設定する方法
- 動的バージョンとアーキテクチャーの値が適切に処理されるように **/etc/yum.conf** の yum 変数と **/etc/yum.repos.d/** ディレクトリー内のファイルを使用する方法
- コマンドラインで yum リポジトリを追加、有効、無効にする方法
- カスタムの yum リポジトリーを設定する方法

5.5.1. [main] オプションの設定

/etc/yum.conf 設定ファイルには、1 つの **[main]** セクションが含まれます。このセクションにあるキー値ペアの中には **yum** の動作に影響を与えるものもあれば、yum のリポジトリーの処理方法に影響を与えるものもあります。**/etc/yum.conf** 内にある **[main]** のセクション下に、多くのオプションを追加することができます。

以下は、**/etc/yum.conf** 設定ファイルのサンプルです。

```
[main]
cachedir=/var/cache/yum/$basearch/$releasever
keepcache=0
debuglevel=2
logfile=/var/log/yum.log
exactarch=1
obsoletes=1
gpgcheck=1
plugins=1
installonly_limit=3
[ コメントは省略されています ]

# PUT YOUR REPOS HERE OR IN separate files named file.repo
# in /etc/yum.repos.d
```

以下は、**[main]** セクションで最もよく使用されるオプションです。

assumeyes=*value*

assumeyes オプションは、yum が重要なアクションに関する確認を尋ねるかどうかを決定します。*value* を以下のどちらかに置き換えます。

0 — (**デフォルト**)。**yum** は実行する重要な動作の確認を尋ねます。

1 — **yum** が実行する重要な動作の確認を尋ねません。**assumeyes=1** に設定すると、**yum** はコマンドラインオプション **-y** および **--assumeyes** と同様の動作を実行します。

cachedir=*directory*

このオプションを使って、yum がキャッシュおよびデータベースファイルを保存するディレクトリーを設定します。*directory* をディレクトリーへの絶対パスで置き換えます。デフォルトでは、yum のキャッシュディレクトリーは **/var/cache/yum/$basearch/$releasever** です。

$basearch および **$releasever** yum 変数の詳細については、「Yum 変数の使用」を参照してください。

debuglevel=*value*

このオプションは、**yum** が生成するデバッグ出力の詳細を指定します。ここでは、*value* は **1** から **10** までの整数になります。**debuglevel** 値を高い値に設定すると、**yum** はより詳しいデバッグ出力を表示します。**debuglevel=0** は、デバッグ出力を無効にします。デフォルトは **debuglevel=2** です。

exactarch=*value*

このオプションを使うと、インストール済みのパッケージを更新の際に **yum** が正確なアーキテクチャーを考慮するように設定できます。*value* を以下のいずれかで置き換えます。

0 — パッケージの更新時には正しいアーキテクチャーを考慮に入れて実行しません。

1 — (**デフォルト**)。パッケージの更新時に正しいアーキテクチャーを考慮に入れて実行します。この設定では、**yum** はシステムにすでにインストール済みの i386 パッケージを更新するための i686 パッケージをインストールしません。

exclude=*package_name* [*more_package_names*]

exclude オプションを使うと、インストールまたは更新中にキーワードを使ってパッケージを除外することができます。除外する複数のパッケージを一覧表示したい場合は、パッケージを空白で区切り、引用符で囲みます。ワイルドカードを使ったシェルグロブ表現 (***** や **?** など) が利用できます。

gpgcheck=*value*

gpgcheck オプションを使うと、**yum** がパッケージに GPG 署名確認を実行するかどうかを指定できます。*value* を以下のいずれかで置き換えます。

0 — インストールされるローカルパッケージなど、全リポジトリー内のパッケージ上での GPG 署名確認を無効にします。

1 — (**デフォルト**)。インストールされるローカルパッケージなど、全リポジトリーのすべてのパッケージ上で GPG 署名確認を有効にします。**gpgcheck** が有効だと、すべてのパッケージ署名が確認されます。

このオプションが **/etc/yum.conf** ファイルの **[main]** セクションで設定されている場合は、全リポジトリに対して GPG 確認ルールが設定されます。代わりに、個々のリポジトリーに **gpgcheck=*value*** を設定することもできます。つまり、あるリポジトリーで GPG 確認を有効にしながら別のリポジトリでは無効にすることができます。個々のリポジトリーに対応する **.repo** ファイルで **gpgcheck=*value*** を設定すると、**/etc/yum.conf** にデフォルト値がある場合はそれを無効にします。

GPG 署名確認の詳細については、「パッケージ署名の確認」を参照してください。

group_command=*value*

group_command オプションを使うと、**yum group install**、**yum group upgrade**、および **yum group remove** の各コマンドがパッケージグループを処理する方法を指定できます。*value* を以下のいずれかで置き換えます。

simple — パッケージグループのすべてのメンバーをインストールします。以前にインストールされたパッケージのみを更新し、その間にグループに追加されたパッケージはインストールしません。

compat — **simple** に似ていますが、**yum upgrade** は前回の更新以降にグループに追加されたパッケージもインストールします。

objects — (**デフォルト**)。このオプションでは、yum は以前にインストールされたグループを追跡し、グループの一部としてインストールされたパッケージと個別にインストールされたパッケージを区別します。例5.14「LibreOffice パッケージグループの情報を表示する」を参照してください。

group_package_types=*package_type* [*more_package_types*]

このオプションでは、**yum group install** コマンドが呼び出された際にどのタイプのパッケージがインストールされるか (**optional**、**default**、または **mandatory**) を指定できます。**default** および **mandatory** のパッケージタイプがデフォルトで選択されます。

history_record=*value*

このオプションを使うと、**yum** はトランザクション履歴を記録します。*value* を以下のいずれかで置き換えます。

0 — **yum** はトランザクションの履歴エントリーを記録**しません**。

1 — (**デフォルト**)。**yum** はトランザクションの履歴エントリーを記録します。この操作はある程度のディスク領域を使用し、トランザクションの時間がいくらか余分にかかりますが、過去の操作についての多くの情報が提供され、**yum history** コマンドでこの情報が表示されます。**history_record=1** がデフォルトです。

yum history コマンドに関する詳細情報は、「トランザクション履歴の活用」を参照してください。

注記

yum は履歴記録を使って、**yum** 以外でなされた **rpmdb** への修正を検出します。修正が検出されると、**yum** は警告を表示し、**rpmdb** の変更によって起こり得る問題を自動的に検索します。**history_record** がオフになっていると、**yum** はこのような変更を検出できず、自動チェックは実行されません。

installonlypkgs=*space separated list of packages*

ここでは、**yum** が **インストール** できるものの **更新** はしないパッケージのリストを空白で区切って提供できます。デフォルトでインストールのみに設定されているパッケージの一覧については、**yum.conf**(5) の man ページを参照してください。

installonlypkgs 指示文を **/etc/yum.conf** ファイルに追加する場合は、**yum.conf**(5) の **installonlypkgs** セクション下に表示されているものも含め、インストールのみである **すべての** パッケージを一覧表示してください。特に、カーネルパッケージは常に **installonlypkgs** (デフォルトのとおり) に一覧表示するようにしてください。また、デフォルトのカーネルがブートに失敗した場合でもバックアップカーネルを常に利用できるように、**installonly_limit** は常に **2** より大きい値に設定することをお勧めします。

installonly_limit=*value*

このオプションは、**installonlypkgs** 指示文でリストされるパッケージを同時にインストールできる数を設定します。*value* を **installonlypkgs** でリストされている単一パッケージについて同時インストールできるバージョンの最大数を表す整数で置き換えます。

installonlypkgs 指示文のデフォルトには複数の様々なカーネルパッケージが含まれているため、**installonly_limit** の値を変更すると、単一のカーネルパッケージのインストール済みバージョンの最大数にも影響が及ぶ点に注意してください。**/etc/yum.conf** に表示されているデフォルト値は、**installonly_limit=3** です。また、この値を低く、特に **2** より下に設定することは推奨されません。

keepcache=*value*

keepcache は、インストールの成功後に yum がヘッダーおよびパッケージのキャッシュを保持するかどうかを決定します。ここでは、*value* を以下のいずれかに置き換えます。

0 — (**デフォルト**)。インストールの成功後に、ヘッダーとパッケージのキャッシュを保持しません。

1 — インストールの成功後、キャッシュを保持します。

logfile=*file_name*

ログ出力の場所を指定するため、*file_name* を **yum** がログ出力を書き込むファイルへの絶対パスで置き換えます。デフォルトでは、**yum** は **/var/log/yum.log** にログを記録します。

max_connenctions=*number*

ここでの *value* は、同時接続の最大数を表します。デフォルトは 5 です。

multilib_policy=*value*

multilib_policy オプションは、インストールするパッケージに複数のアーキテクチャーバージョンがある場合にインストール動作を設定します。ここでの *value* は、以下のいずれかになります。

best — このシステムに最適なアーキテクチャーをインストールします。例えば AMD64 システムに **multilib_policy=best** を設定すると、**yum** は全パッケージの 64-bit バージョンをインストールします。

all — 常に全パッケージ用の可能なあらゆるアーキテクチャーをインストールします。例えば、AMD64 システムで **multilib_policy** を **all** に設定すると、**yum** はパッケージの i686 および AMD64 が利用可能であれば両バージョンをインストールします。

obsoletes=*value*

obsoletes オプションは、更新の実行時に obsoletes 処理ロジックを有効にします。あるパッケージがスペックファイル内で別のパッケージを *廃止する* ように宣言している場合、宣言しているパッケージがインストールされると、宣言されているパッケージはこのパッケージによって置き換えられます。例えば、パッケージ名が変更された場合などに obsoletes は宣言されます。*value* を以下のいずれかで置き換えます。

0 — 更新の実行時に **yum** の obsoletes 処理ロジックを無効にします。

0 — (**デフォルト**)。更新の実行時に **yum** の obsoletes 処理ロジックを有効にします。

plugins=*value*

これは、**yum** プラグインを有効または無効にするグローバルスイッチです。*value* は以下のいずれかになります。

0 — yum のプラグインをグローバルで無効にします。

重要

一部のプラグインは重要な **yum** サービスを提供するため、すべてのプラグインを無効にすることは推奨されません。特に、**product-id** および **subscription-manager** プラグインは証明書ベースの **Content Delivery Network** (CDN) のサポートを提供します。プラグインをグローバルで無効にするオプションは便利なオプションとして提供されていますが、通常は **yum** に潜在的な問題があると判断された場合にのみ使用することが推奨されます。

1 — (**デフォルト**)。すべての yum プラグインをグローバルで有効にします。**plugins=1** に設定した場合、特定の yum プラグインの設定ファイル内で **enabled=0** と設定してそのプラグインを無効にすることも可能です。

yum の各種プラグインの詳細については、「Yum のプラグイン」を参照してください。プラグインの制御に関する詳細は、「Yum プラグインを有効、設定、無効にする方法」を参照してください。

reposdir=*directory*

ここでの *directory* は **.repo** ファイルがあるディレクトリーへの絶対パスです。すべての **.repo** ファイルには、リポジトリー情報 (**/etc/yum.conf** の **[*repository*]** セクションと類似) が含まれています。**yum** は **.repo** ファイルおよび **/etc/yum.conf** ファイルの **[*repository*]** セクションからすべてのリポジトリー情報を収集し、トランザクションに使用するリポジトリーのマスター一覧を作成します。**reposdir** が設定されていない場合は、**yum** はデフォルトのディレクトリーである **/etc/yum.repos.d/** を使用します。

retries=*value*

このオプションは、エラーを返す前に **yum** がファイルの取得を試行する回数を設定します。*value* は **0** 以上の整数で、**0**に設定すると、**yum** はその試行を何度も続けます。デフォルト値は **10** です。

利用可能な **[main]** オプションの完全なリストは、**yum.conf**(5) の man ページの **[main] OPTIONS** セクションを参照してください。

5.5.2. [repository] オプションの設定

[*repository*] セクションでは、個別の yum リポジトリーを定義することができます。ここでの *repository* は、**my_personal_repo** (空白は使用不可) などの一意のリポジトリー ID になります。

[*repository*] セクションでは、少なくとも以下の例のような形式が必要になります。

```
[repository]
name=repository_name
baseurl=repository_url
```

すべての **[*repository*]** セクションには、以下の指示文を含める必要があります。

name=*repository_name*

ここでは、*repository_name* は、リポジトリーを説明するヒューマンリーダブルな文字列になります。

baseurl=*repository_url*

*repository_url*をリポジトリーの repodata ディレクトリーがあるディレクトリーへの URL で置き

換えます。

- リポジトリーが HTTP にある場合は、**http://path/to/repo** を使用します。
- リポジトリーが FTP にある場合は、**ftp://path/to/repo** を使用します。
- リポジトリーがマシンのローカルにある場合は、**file:///path/to/local/repo** を使用します。
- 特定のオンラインリポジトリーでベーシック HTTP 認証が必要な場合は、***username*:*password*@*link*** として URL にユーザー名とパスワードを先頭に追加して、指定することができます。例えば、http://www.example.com/repo/ にあるリポジトリーでユーザー名 「user」 とパスワード「password」 が必要な場合、**baseurl** のリンクは、**http://user:password@www.example.com/repo/** として指定できます。

通常この URL は以下のような HTTP リンクになります。

```
baseurl=http://path/to/repo/releases/$releasever/server/$basearch/
os/
```

yum は常に URL の **$releasever**、**$arch**、**$basearch** 変数を展開する点に注意して下さい。yum 変数の詳細については、「Yum 変数の使用」 を参照してください。

以下のような便利な **[*repository*]** 指示文もあります。

enabled=*value*

これで簡単に **yum** に特定のリポジトリーを使用するか無視するかを伝えられます。*value* は以下のいずれかになります。

0 — 更新およびインストールの実行時には、パッケージソースとしてこのリポジトリーを含めません。これはリポジトリーを迅速に有効または無効にする簡単な方法です。更新またはインストール用に有効としたくないリポジトリーから、単一パッケージが欲しい場合に便利です。

1 — パッケージソースとしてこのリポジトリーを含めます。

リポジトリーのオンとオフは、**--enablerepo=*repo_name*** もしくは **--disablerepo=*repo_name*** オプションを **yum** に渡すか、**PackageKit** ユーティリティーの **ソフトウェアの追加/削除** ウィンドウから実行できます。

async=*value*

リポジトリーパッケージの並行ダウンロードを制御します。*value* は以下のいずれかになります。

auto — (**デフォルト**) 可能な場合、並行ダウンロードを使用します。つまり、失敗を回避するために **yum** はプラグインが作成したリポジトリーについては自動的にこれを無効にします。

on — リポジトリーの並行ダウンロードが有効になります。

off — リポジトリー並行ダウンロードが無効になります。

他にも多くの **[*repository*]** オプションがあります。このうちのいくつかは、**[*main*]** オプションと同一形式、同一機能になります。全一覧については、**yum.conf**(5) の man ページの **[repository] OPTIONS** セクションを参照してください。

例5.24 /etc/yum.repos.d/redhat.repo ファイルのサンプル

以下は、**/etc/yum.repos.d/redhat.repo** ファイルのサンプルです:

```
#
# Red Hat Repositories
# Managed by (rhsm) subscription-manager
#

[red-hat-enterprise-linux-scalable-file-system-for-rhel-6-entitlement-
rpms]
name = Red Hat Enterprise Linux Scalable File System (for RHEL 6
Entitlement) (RPMs)
baseurl = https://cdn.redhat.com/content/dist/rhel/entitlement-
6/releases/$releasever/$basearch/scalablefilesystem/os
enabled = 1
gpgcheck = 1
gpgkey = file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
sslverify = 1
sslcacert = /etc/rhsm/ca/redhat-uep.pem
sslclientkey = /etc/pki/entitlement/key.pem
sslclientcert = /etc/pki/entitlement/11300387955690106.pem

[red-hat-enterprise-linux-scalable-file-system-for-rhel-6-entitlement-
source-rpms]
name = Red Hat Enterprise Linux Scalable File System (for RHEL 6
Entitlement) (Source RPMs)
baseurl = https://cdn.redhat.com/content/dist/rhel/entitlement-
6/releases/$releasever/$basearch/scalablefilesystem/source/SRPMS
enabled = 0
gpgcheck = 1
gpgkey = file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
sslverify = 1
sslcacert = /etc/rhsm/ca/redhat-uep.pem
sslclientkey = /etc/pki/entitlement/key.pem
sslclientcert = /etc/pki/entitlement/11300387955690106.pem

[red-hat-enterprise-linux-scalable-file-system-for-rhel-6-entitlement-
debug-rpms]
name = Red Hat Enterprise Linux Scalable File System (for RHEL 6
Entitlement) (Debug RPMs)
baseurl = https://cdn.redhat.com/content/dist/rhel/entitlement-
6/releases/$releasever/$basearch/scalablefilesystem/debug
enabled = 0
gpgcheck = 1
gpgkey = file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
sslverify = 1
sslcacert = /etc/rhsm/ca/redhat-uep.pem
sslclientkey = /etc/pki/entitlement/key.pem
sslclientcert = /etc/pki/entitlement/11300387955690106.pem
```

5.5.3. Yum 変数の使用

yum コマンドおよびすべての yum 設定ファイル (つまり**/etc/yum.conf** および **/etc/yum.repos.d/** ディレクトリー内のすべての **.repo** ファイル) 内で、以下の組み込み変数を使用および参照することができます。

$releasever

この変数を使用すると、Red Hat Enterprise Linux のリリースバージョンを参照することができます。yum は **/etc/yum.conf** 設定ファイルにある **distroverpkg=*value*** の行から **$releasever** の値を取得します。**/etc/yum.conf** にそのような行がない場合は、**yum** は *redhat-release* パッケージからバージョン番号を取得することで、正しい値を推測します。

$arch

この変数を使用して、Python の **os.uname()** 機能を呼び出す時に返り値としてシステムの CPU アーキテクチャーを参照できます。**$arch** の有効な値は、**i586**、**i686**、**x86_64** です。

$basearch

$basearch を使用すると、システムのベースアーキテクチャーを参照できます。たとえば、i686 および i586 の両マシンは **i386** のベースアーキテクチャーを持っており、AMD64 および Intel64 マシンは **x86_64** のベースアーキテクチャーを持っています。

$YUM0-9

これら 10 個の変数は、同じ名前を持つシェル環境変数の値でそれぞれ置換されます。これら変数のいずれかが (例えば **/etc/yum.conf** で) 参照され、同じ名前を持つシェル環境変数が存在しない場合は、設定ファイルの変数は置換されません。

カスタム変数の定義、既存の変数値の上書きを行うには、**/etc/yum/vars/** ディレクトリー内に変数と同じ名前を持つファイルを作成して (「**$**」 記号はなし) 、1 行目に希望する値を追加します。

たとえば多くの場合、リポジトリーの詳細にはオペレーティングシステムの名前が含まれます。**$osname** と呼ばれる新しい変数を定義するには、1 行目に 「Red Hat Enterprise Linux」 の名前を持つ新しいファイルを作成して、**/etc/yum/vars/osname** として保存します。

```
~]# echo "Red Hat Enterprise Linux 7" > /etc/yum/vars/osname
```

.repo ファイルでは、「Red Hat Enterprise Linux 7」 の代わりに以下を使用することができます。

```
name=$osname $releasever
```

注記

カスタム変数の作成は現在、変数代用の問題に影響を受けています。(BZ#1102585)

5.5.4. 現行設定の表示

yum グローバルオプションの現在の値 (つまり **/etc/yum.conf** ファイルの **[main]** セクションで指定されたオプション) を表示するには、コマンドラインのオプションなしで **yum-config-manager** を実行します。

```
yum-config-manager
```

別の設定セクションの内容を一覧表示するには、以下の形式でコマンドを実行します。

```
yum-config-manager section…
```

glob 表現を使用して、適合する全セクションの設定を表示することもできます。

```
yum-config-manager glob_expression…
```

例5.25 main セクションの設定を表示する

すべての設定オプションとそれらに対応する値を一覧表示するには、シェルプロンプトで以下を入力します。

```
~]$ yum-config-manager main \*
Loaded plugins: langpacks, product-id, subscription-manager
================================== main
==================================
[main]
alwaysprompt = True
assumeyes = False
bandwith = 0
bugtracker_url = https://bugzilla.redhat.com/enter_bug.cgi?
product=Red%20Hat%20Enterprise%20Linux%206&component=yum
cache = 0[出力は省略されています]
```

5.5.5. Yum リポジトリーの追加、有効化、無効化

「[repository] オプションの設定」では、yum リポジトリーの定義に使用可能なさまざまなオプションについて説明しました。本セクションでは、**yum-config-manager** コマンドを使用してリポジトリーを追加、有効、無効にする方法を説明します。

重要

システムが Red Hat Subscription Management で証明書ベースの **Content Delivery Network** (CDN) に登録されている場合、**/etc/yum.repos.d/redhat.repo** ファイル内のリポジトリー管理には **Red Hat サブスクリプションマネジャー** ツールが使用されます。

Yum リポジトリーの追加

新しいリポジトリを定義するには、**[*repository*]** セクションを **/etc/yum.conf** ファイルか、**/etc/yum.repos.d/** ディレクトリー内の **.repo** ファイルに追加します。**yum** は、このディレクトリー内にある **.repo** ファイル拡張子が付いたすべてのファイルを読み取ることができます。リポジトリーは **/etc/yum.conf** 内ではなく、ここに定義することが推奨されます。

警告

Red Hat の認証ベース **Content Delivery Network** (CDN) 以外の未検証または信頼できないソフトウェアソースからソフトウェアパッケージを取得してインストールする場合には、セキュリティー上のリスクが伴います。セキュリティー、安定性、互換性、保全性に関する問題につながる恐れがあります。

通常 Yum リポジトリーにはそれぞれの **.repo** ファイルがあります。あるリポジトリーをシステムに追加して、有効にするには、**root** で以下のコマンドを実行します。

```
yum-config-manager --add-repo repository_url
```

ここでの *repository_url* は、**.repo** ファイルへのリンクになります。

例5.26 example.repo を追加する

http://www.example.com/example.repo にあるリポジトリーを追加するには、シェルプロンプトで以下を入力します。

```
~]# yum-config-manager --add-repo http://www.example.com/example.repo
Loaded plugins: langpacks, product-id, subscription-manager
adding repo from: http://www.example.com/example.repo
grabbing file http://www.example.com/example.repo to
/etc/yum.repos.d/example.repo
example.repo                                               |  413 B
00:00
repo saved to /etc/yum.repos.d/example.repo
```

Yum リポジトリーの有効化

特定のリポジトリーを有効にするには、シェルプロンプトで **root** として以下を入力します。

```
yum-config-manager --enable repository…
```

ここでの *repository* は、一意のリポジトリー ID になります (利用可能なリポジトリー ID を一覧表示するには **yum repolist all** を使用)。別の方法では、glob 表現を使用すると、一致するすべてのリポジトリーを有効にできます。

```
yum-config-manager --enable glob_expression…
```

例5.27 /etc/yum.conf のカスタムセクションで定義されるリポジトリーを有効にする

[example]、**[example-debuginfo]**、**[example-source]** セクション内で定義されたリポジトリーを有効にするには、以下を入力します。

```
~]# yum-config-manager --enable example\*
Loaded plugins: langpacks, product-id, subscription-manager
================================ repo: example
================================
[example]
bandwidth = 0
base_persistdir = /var/lib/yum/repos/x86_64/6Server
baseurl = http://www.example.com/repo/6Server/x86_64/
cache = 0
cachedir = /var/cache/yum/x86_64/6Server/example[出力は省略されています]
```

成功すると、**yum-config-manager --enable** コマンドは現在のリポジトリー設定を表示します。

Yum リポジトリーの無効化

yum リポジトリーを無効にするには、**root** で以下のコマンドを実行します。

```
yum-config-manager --disable repository…
```

ここでの *repository* は一意のリポジトリー ID になります (利用可能なリポジトリー ID を一覧表示するには **yum repolist all** を使用)。**yum-config-manager --enable** と同様に、glob 表現を使用して、一致するすべてのリポジトリーを同時に無効にすることができます。

```
yum-config-manager --disable glob_expression…
```

成功すると、**yum-config-manager --disable** コマンドは現在の設定を表示します。

5.5.6. Yum リポジトリーの作成

yum リポジトリーを設定するには、以下のステップにしたがいます。

1. シェルプロンプトで **root** として以下を入力し、*createrepo* パッケージをインストールします。

   ```
   yum install createrepo
   ```

2. リポジトリーに配置するパッケージすべてを、**/mnt/local_repo/** などの 1 つのディレクトリーにコピーします。

3. このディレクトリーに移動して、以下のコマンドを実行します。

   ```
   createrepo --database /mnt/local_repo
   ```

   これにより yum リポジトリーに必要なメタデータ、さらには **sqlite** データベースが作成されるため **yum** の動作が迅速化します。

5.5.7. Optional および Supplementary リポジトリーの追加

Optional および Supplementary のサブスクリプションチャンネルでは、オープンソースのライセンス付きソフトウェア (Optional チャンネル) と商用ライセンス付きソフトウェア (Supplementary チャンネル) をカバーする Red Hat Enterprise Linux 用の追加ソフトウェアパッケージが提供されます。

Optional および Supplementary チャンネルをサブスクライブする前に、対象範囲の詳細 を確認してください。これらのチャンネルからパッケージをインストールする場合は、Red Hat j カスタマーポータルの 証明書ベースの管理を使用して、Optional および Supplementary チャンネル、-devel パッケージにアクセスする方法 の記事で説明してあるステップにしたがってください。

5.6. Yum のプラグイン

Yum は、動作を拡張し強化するプラグインを提供します。一部のプラグインは、デフォルトでインストールされています。**yum** コマンドを呼び出すと常に、Yum はどのプラグインが読み込まれ、アクティブかを知らせます。例を示します。

```
~]# yum info yum
Loaded plugins: langpacks, product-id, subscription-manager[出力は省略されています]
```

Loaded plugins に続くプラグインの名前は **--disableplugins=*plugin_name*** オプションに渡すことが可能な名前である点に注意して下さい。

5.6.1. Yum プラグインを有効、設定、無効にする方法

yum プラグインを有効にするには、**plugins=** で始まる行が **/etc/yum.conf** の **[main]** セクションにあり、値が **1** であるようにします。

```
plugins=1
```

すべてのプラグインを無効にするには、この行を **plugins=0** に変更します。

重要

一部のプラグインは重要な **yum** サービスを提供するため、すべてのプラグインを無効にすることは推奨されません。特に、**product-id** および **subscription-manager** プラグインは証明書ベースの **Content Delivery Network** (CDN) のサポートを提供します。プラグインをグローバルで無効にするオプションは便利なオプションとして提供されていますが、通常は **yum** に潜在的な問題があると判断された場合にのみ使用することが推奨されます。

すべてのインストール済みプラグインには、**/etc/yum/pluginconf.d/** ディレクトリーにそれぞれの設定ファイルがあります。これらのファイルでプラグイン固有のオプションを設定できます。たとえば、以下のように **aliases** プラグインの **aliases.conf** 設定ファイルがあるとします。

```
[main]
enabled=1
```

プラグインの設定ファイルには常に **[main]** セクション (yum の **/etc/yum.conf** ファイルに類似) があります。そこでは、**yum** コマンドの実行時にプラグインを有効にするかどうかを制御する**enabled=** オプションがあります (ない場合は設定可能)。

/etc/yum.conf で **enabled=0** と設定してすべてのプラグインを無効にすると、すべてのプラグインに個々の設定ファイルで有効かどうかに関わらず無効になります。

1 回の **yum** コマンドで単にすべての yum プラグインを無効にする場合は、**--noplugins** オプションを使用します。

1 回の **yum** コマンドで 1 つ以上の yum プラグインを無効にしたい場合は、コマンドに**--disableplugin=*plugin_name*** オプションを追加します。たとえば、システムの更新中に**aliases** プラグインを無効にするには、以下を入力します。

```
~]# yum update --disableplugin=aliases
```

--disableplugin= オプションに渡すプラグインの名前は、**yum** コマンドの出力内にある **Loaded plugins** の行の後に表示されている名前と同じです。複数のプラグインを無効にするには、名前をコンマで区切ります。さらに glob 表現を使用すると、複数のプラグイン名の適合や名前の短縮を行うことができます。

```
~]# yum update --disableplugin=aliases, lang*
```

5.6.2. 追加の Yum プラグインのインストール

通常 yum プラグインは **yum-plugin-*plugin_name*** パッケージの命名規則に準拠しますが、常にそうであるとは限りません。たとえば、**kabi** プラグインを提供するパッケージの名前は、**kabi-yum-plugins** です。yum プラグインのインストールは、他のパッケージをインストールする場合と同じよう

に実行できます。たとえば、**yum-aliases** プラグインをインストールするには、シェルプロンプトで以下を入力します。

```
~]# yum install yum-plugin-aliases
```

5.6.3. yum プラグインの使用方法

以下では、便利な yum プラグインのいくつかについての説明と使用方法を紹介しています。プラグインは名前で表示されており、括弧内はパッケージ名になります。

kabi (*kabi-yum-plugins*)

kabi プラグインは、ドライバー更新パッケージが公式の Red Hat*kernel Application Binary Interface* (kABI) と適合するかどうかを確認します。このプラグインが有効な状態で、ユーザーがホワイトリストにないカーネルシンボルを使用するパッケージのインストールを試行する場合、警告メッセージがシステムログに書き込まれます。さらには、プラグインを enforcing モードで実行するように設定すると、そうしたパッケージがインストールすらされなくなります。

kabi プラグインを設定するには、**/etc/yum/pluginconf.d/kabi.conf** にある設定ファイルを編集します。**[main]** セクションで利用できる指示文の一覧は、以下の表に示されています。

表5.3 サポートされている kabi.conf 指示文

指示文	詳細
enabled=*value*	プラグインを有効または無効にできます。*value* は **1** (有効) または **0** (無効) にする必要があります。インストール時には、プラグインはデフォルトで有効です。
whitelists=*directory*	サポートされているカーネルシンボルを持つファイルがある *directory* を指定できます。デフォルトでは、**kabi** プラグインは *kernel-abi-whitelists* パッケージ (**/lib/modules/kabi-rhel70/** ディレクトリー) が提供するファイルを使用します。
enforce=*value*	enforcing モードを有効または無効にできます。*value* は **1** (有効) または **0** (無効) にする必要があります。デフォルトでは、このオプションはコメントアウトされ **kabi** プラグインは警告メッセージのみを表示します。

product-id (*subscription-manager*)

product-id プラグインは、Content Delivery Network (コンテンツ配信ネットワーク) からインストールされた製品の製品識別証明書を管理します。**product-id** プラグインはデフォルトでインストールされています。

langpacks (*yum-langpacks*)

langpacks プラグインは、インストールされているすべてのパッケージ用に選択された言語のロケールパッケージを検索するために使用します。**langpacks** プラグインはデフォルトでインストールされます。

aliases (*yum-plugin-aliases*)

aliases は、yum コマンドでのエイリアスの設定および使用を可能にする**alias** コマンドラインオプションを追加します。

yum-changelog (*yum-plugin-changelog*)

yum-changelog プラグインは、更新の前後でパッケージ変更ログの表示を可能にする**--changelog** コマンドラインオプションを追加します。

yum-tmprepo (*yum-plugin-tmprepo*)

yum-tmprepo プラグインは、リポジトリーファイルの URL を受けてダウンロードし、これを 1 回のみのトランザクションに有効とする **--tmprepo** コマンドラインオプションを追加します。このプラグインはリポジトリーの安全な一時的使用を確保します。デフォルトでは、gpg 確認を無効にしません。

yum-verify (*yum-plugin-verify*)

yum-verify プラグインは、システム上の検証データを表示するための**verify**、**verify-rpm**、**verify-all** の各コマンドラインオプションを追加します。

yum-versionlock (*yum-plugin-versionlock*)

yum-versionlock プラグインは選択されたパッケージの他のバージョンを除外し、パッケージが最新バージョンに更新されることを防ぎます。**versionlock** コマンドラインオプションを使うと、ロックされたパッケージを表示、編集することができます。

5.7. その他のリソース

Red Hat Enterprise Linux でソフトウェアパッケージを管理する方法についての詳細情報は、下記のリソースを参照してください。

インストールされているドキュメント

- **yum**(8) — **yum** コマンドラインユーティリティーの man ページでは、サポートされるオプションとコマンドの完全なリストが提供されます。
- **yumdb**(8) — **yumdb** コマンドラインユーティリティーの man ページでは、このツールを使ってクエリを行い、必要であれば yum データベースを変更する方法が説明されています。
- **yum.conf**(5) — **yum.conf** の man ページでは、利用可能な yum 設定オプションが説明されています。
- **yum-utils**(1) — **yum-utils** の man ページでは、yum 設定の管理、リポジトリーの操作、yum データベースの作業を行う追加ユーティリティーについての一覧表示と簡単な説明が提供されます。

オンラインのドキュメント

- Yum Guides — プロジェクトホームページ上の『Yum Guides』 では、追加のドキュメンテーションのリンクがあります。

関連項目

- 4章*権限の取得* では、**su** および **sudo** コマンドを使って管理者権限を取得する方法が説明されています。
- 付録A *RPM* では、Red Hat Enterprise Linux で使用するパッケージングシステムである**RPM Package Manager** (RPM) について説明しています。

パート III. インフラストラクチャーサービス

ここでは、サービスおよびデーモンの設定方法と、Red Hat Enterprise Linux マシンへのリモートアクセス
を可能にする方法についての情報を提供しています。

第6章 systemd によるサービス管理

6.1. systemd の概要

Systemd は、Linux オペレーティングシステム用のシステムおよびサービスマネージャーです。SysV ini スクリプトと後方互換するように設計されており、起動時のシステムサービスの並行スタートアップやデーモンのオンデマンドのアクティベーション、システム状態のスナップショットのサポート、依存ベースのサービス管理論理などの多くの機能を提供します。Red Hat Enterprise Linux 7 では、systemd は Upstart に代わるデフォルトの init システムです。

Systemd では、*systemd units* という概念が導入されています。これらの unit は表6.2「Systemd Unit の場所」 にあるディレクトリーの 1 つに置かれる unit 設定ファイルで表示され、システムサービスやリスニングソケット、保存されたシステム状態のスナップショットについての情報や init システムに関連するその他のオブジェクトについての情報を要約します。利用可能な systemd unit タイプの完全な一覧については、表6.1「利用可能な systemd Unit タイプ」 を参照してください。

表6.1 利用可能な systemd Unit タイプ

Unit タイプ	ファイル拡張子	詳細
Service unit	**.service**	システムサービス
Target unit	**.target**	systemd unit のグループ
Automount unit	**.automount**	ファイルシステムの自動マウントポイント
Device unit	**.device**	カーネルが認識するデバイスファイル
Mount unit	**.mount**	ファイルシステムのマウントポイント
Path unit	**.path**	ファイルシステム内のファイルもしくはディレクトリー
Scope unit	**.scope**	外部作成のプロセス
Slice unit	**.slice**	システムプロセスを管理する階層的に組織された unit グループ
Snapshot unit	**.snapshot**	systemd マネージャーの保存状態
Socket unit	**.socket**	プロセス間の通信ソケット
Swap unit	**.swap**	スワップデバイスまたはスワップファイル
Timer unit	**.timer**	systemd タイマー

表6.2 Systemd Unit の場所

フォルダー	詳細
/usr/lib/systemd/system/	インストール済みの RPM パッケージで配布された systemd unit
/run/systemd/system/	ランタイム時に作成された systemd unit。このディレクトリーは、インストール済みのサービス unit のディレクトリーに優先します。
/etc/systemd/system/	システム管理者が作成、管理する systemd unit。このディレクトリーは、ランタイム unit のディレクトリーに優先します。

6.1.1. 主な特長

Red Hat Enterprise Linux 7 では、systemd システムおよびサービスマネージャーは以下の主要な機能を提供します。

- **ソケットベースのアクティベーション** — 起動時に systemd は、このタイプのアクティベーションをサポートするすべてのシステムサービス用のリスニングソケットを作成し、サービスが開始するとすぐに

これらのソケットをサービスに渡します。これで systemd がサービスを並行で開始できるだけでなく、サービスが利用可能でない間に送信されたメッセージを失うことなくサービスの再起動が可能になります。これは、対応するソケットがアクセス可能なままで、すべてのメッセージがキューに登録されるためです。

Systemd はソケットベースのアクティベーションに *socket units* を使用します。

- **バスベースのアクティベーション** — プロセス間の通信に D-Bus を使用するシステムサービスは、クライアントアプリケーションがシステムサービスとの通信を試みる初回にオンデマンドでスタートできます。Systemd は、バスベースのアクティベーションに *D-Bus サービスファイル* を使用します。

- **デバイスベースのアクティベーション** — デバイスベースのアクティベーションをサポートするシステムサービスは、特定のタイプのハードウェアがプラグインするか利用可能になった際に、オンデマンドでスタートできます。Systemd は、デバイスベースのアクティベーションに *device units* を使用します。

- **パスベースのアクティベーション** — パスベースのアクティベーションをサポートするシステムサービスは、特定のファイルもしくはディレクトリーの状態が変更される際にオンデマンドでスタートできます。Systemd は、パスベースのアクティベーションに *path units* を使用します。

- **システム状態のスナップショット** — Systemd は、一時的にすべての unit の現状を保存するか、動的に作成されたスナップショットから以前のシステムの状態を復元することができます。システムの現状を保存するために systemd は、動的に作成された *snapshot units* を使用します。

- **マウントおよびオートマウントポイントの管理**— Systemd は、マウントポイントおよびオートマウントポイントを監視、管理します。Systemd はマウントポイントには *mount units* を、オートマウントポイントには *automount units* を使用します。

- **アグレッシブな並列化** — ソケットベースのアクティベーションを使用するため、systemd は すべてのリスニングソケットが配置されると同時に並行してシステムサービスを開始できます。オンデマンドのアクティベーションをサポートするシステムサービスと組み合わせることで、並行アクティベーションはシステム起動に必要となる時間を大幅に短縮します。

- **トランザクション unit アクティベーション論理** — unit のアクティベーションまたは非アクティブ化の前に、systemd はその依存関係を計算して一時的なトランザクションを作成し、このトランザクションの一貫性を検証します。トランザクションに一貫性がない場合、systemd は自動的にこれを正そうとし、エラーをレポートする前に必須でないジョブを削除しようと試みます。

- **SysV init との後方互換性** — 『Linux Standard Base Core Specification』 にあるように、Systemd は完全に SysV init スクリプトをサポートします。これにより、systemd サービス unit への更新パスが容易になります。

6.1.2. 互換性の変更点

systemd システムおよびサービスマネージャーは、SysV init および Upstart とほぼ互換するように設計されています。以下では、Red Hat Enterprise Linux システムの以前のメジャーリリースとの比較で最も顕著な互換性の変更点を挙げています。

- Systemd のランレベルのサポートは限定的なものです。ランレベルに直接マッピング可能なターゲットユニットを数多く提供し、互換性のために以前の **`runlevel`** コマンドで配布されます。ただし、systemd ターゲットのすべてがランレベルに直接マッピング可能なわけではないため、その結果、このコマンドが不明なランレベルを示す **`N`** を返す場合もあります。可能な場合は、**`runlevel`** コマンドの使用を避けることが推奨されます。

  systemd ターゲットとそれらのランレベルとの比較に関する詳細は、「systemd ターゲットでの作業」を参照してください。

- **`systemctl`** ユーティリティーは、カスタマイズされたコマンドをサポートしません。**`start`**、**`stop`**、

および **status** といった標準のコマンドに加えて、SysV init スクリプトのオーサーは、任意の多数のコマンドに対するサポートを実装して追加機能を提供することができます。たとえば、Red Hat Enterprise Linux 6 の **iptables** の init スクリプトは、**panic** コマンドとの実行が可能です。こうすると、パニックモードが即座に有効になり、システムを再設定してすべての受信および送信パケットの切断が開始されます。これは systemd ではサポートされておらず、**systemctl** は文書化されたコマンドのみを受け付けます。

systemctl ユーティリティーの詳細および以前の **service** ユーティリティーとの比較に関する情報は、「システムサービスの管理」を参照してください。

- **systemctl** ユーティリティーは、systemd が開始していないサービスとは通信しません。systemd がシステムサービスを開始すると、メインプロセスの ID を保存してそれを追跡します。すると **systemctl** ユーティリティーはこの PID を使ってクエリを行い、サービスを管理します。このため、ユーザーがコマンドラインで特定のデーモンを直接開始すると、**systemctl** は最新の状態を判断したり、これを停止したりすることができません。

- Systemd が停止するのは、実行中のサービスのみです。以前は、シャットダウンシーケンスが開始されると、Red Hat Enterprise Linux 6 およびそれ以前のシステムのリリースは、**/etc/rc0.d/** ディレクトリーにあるシンボリックリンクを使ってシステムサービスのステータスに関係なく、利用可能なシステムサービスすべてを停止していました。systemd では、シャットダウン時に、実行中のサービスのみが停止されます。

- システムサービスは標準の入力ストリームからは読み込みができません。systemd がサービスを開始すると、標準入力を **/dev/null** に接続し、ユーザーとのインタラクションを防ぎます。

- システムサービスは、(**HOME** および **PATH** 環境変数といった) コンテキストを開始したユーザーやそのセッションからコンテキストを継承しません。各サービスは、クリーンな実行コンテキストで実行されます。

- Systemd は、Linux Standard Base (LSB) ヘッダーにコード化されている依存関係情報を読み込み、ランタイム時に解釈します。

- サービスユニット上のすべての操作は 5 分間でタイムアウトするようになっており、サービスの故障てシステムがフリーズすることを防ぎます。

systemd で導入された互換性変更点の詳細なリストは、Red Hat Enterprise Linux 7 の 移行計画ガイド を参照してください。

6.2. システムサービスの管理

以前のバージョンの Red Hat Enterprise Linux は SysV init もしくは Upstart と配布されており、**/etc/rc.d/init.d/** にある *init スクリプト* を使用していました。これらの init スクリプトは通常、Bash で書かれており、システム管理者がシステムの状態とシステム内のデーモンを管理できるようになっていました。Red Hat Enterprise Linux 7 では、これらの init スクリプトは、*サービスユニット* に代わっています。

サービスユニットは、ファイル拡張子 **.service** で終わり、init スクリプトと同様の目的を果たします。システムサービスの表示、開始、停止、再開、有効化、無効化には、表6.3「service ユーティリティーと systemctl の比較」、表6.4「chkconfig ユーティリティーと systemctl の比較」、および以下のセクションで説明されているように、**systemctl** コマンドラインを使用します。**service** および **chkconfig** はシステム内で利用可能でまだ機能しますが、これらは互換性のために含まれており、使用の回避をお勧めします。

注記

明確化を図るため、本セクションの残りの部分の例では、**.service** ファイル拡張子がついた完全なユニット名を使用します。例を示します。

```
~]# systemctl stop bluetooth.service
```

システムサービスで作業する際は、この拡張子を省略して入力作業を短縮することができます。**systemctl** は、ファイル拡張子がないユニット名に遭遇すると、自動的にサービスユニットだと想定します。以下のコマンドは、上記のものと同等になります。

```
~]# systemctl stop bluetooth
```

表6.3 service ユーティリティーと systemctl の比較

サービス	systemctl	詳細
service *name* start	**systemctl start *name*.service**	サービスを開始します。
service *name* stop	**systemctl stop *name*.service**	サービスを停止します。
service *name* restart	**systemctl restart *name*.service**	サービスを再起動します。
service *name* condrestart	**systemctl try-restart *name*.service**	サービスが実行中の場合のみ、再起動します。
service *name* reload	**systemctl reload *name*.service**	設定を再読み込みします。
service *name* status	**systemctl status *name*.service** **systemctl is-active *name*.service**	サービスが実行中かどうかをチェックします。
service --status-all	**systemctl list-units --type service --all**	すべてのサービスのステータスを表示します。

表6.4 chkconfig ユーティリティーと systemctl の比較

chkconfig	systemctl	詳細
chkconfig *name* on	**systemctl enable *name*.service**	サービスを有効にします。
chkconfig *name* off	**systemctl disable *name*.service**	サービスを無効にします。
chkconfig --list *name*	**systemctl status *name*.service** **systemctl is-enabled *name*.service**	サービスが有効かどうかをチェックします。
chkconfig --list	**systemctl list-unit-files --type service**	サービスすべてを一覧表示し、それらが有効かどうかをチェックします。

6.2.1. サービスの一覧表示

読み込み済みの最新サービスユニットをすべて一覧表示するには、シェルプロンプトで以下を入力します。

```
systemctl list-units --type service
```

このコマンドは、各サービスユニットの完全な名前 (**UNIT**) を表示し、その後にユニットが読み込み済みかどうか (**LOAD**)、高レベル (**ACTIVE**) および低レベル (**SUB**) のユニットのアクティベーション状態、および簡単な説明 (**DESCRIPTION**) が続きます。

デフォルトでは、**systemctl list-units** コマンドはアクティブなユニットのみを表示します。状態に関係なくすべての読み込み済みユニットを表示したい場合は、**--all** または **-a** のコマンドラインオプションを付けてコマンドを実行します。

```
systemctl list-units --type service --all
```

また、すべての利用可能なサービスユニットを一覧表示させて、それらが有効かどうかをチェックできます。これには、以下を入力します。

```
systemctl list-unit-files --type service
```

このコマンドは各サービスユニットの完全な名前 (**UNIT FILE**) を表示し、その後にサービスユニットが有効かどうか (**STATE**) が表示されます。個別のサービスユニットの状態を判断する方法については、「サービスステータスの表示」 を参照してください。

例6.1 サービスの一覧表示

現在読み込み済みの全サービスユニットを一覧表示するには、以下のコマンドを実行します。

```
~]$ systemctl list-units --type service
UNIT                            LOAD   ACTIVE SUB     DESCRIPTION
abrt-ccpp.service               loaded active exited  Install ABRT
coredump hook
abrt-oops.service               loaded active running ABRT kernel log
watcher
abrt-vmcore.service             loaded active exited  Harvest vmcores
for ABRT
abrt-xorg.service               loaded active running ABRT Xorg log
watcher
abrtd.service                   loaded active running ABRT Automated
Bug Reporting Tool
...
systemd-vconsole-setup.service loaded active exited  Setup Virtual
Console
tog-pegasus.service             loaded active running OpenPegasus CIM
Server

LOAD   = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of
SUB.
SUB    = The low-level unit activation state, values depend on unit
type.

46 loaded units listed. Pass --all to see loaded but inactive units,
too.
To show all installed unit files use 'systemctl list-unit-files'
```

インストール済みの全サービスユニットを表示して、それらが有効かどうかを判断するには、以下を入力します。

```
~]$ systemctl list-unit-files --type service
UNIT FILE                                     STATE
abrt-ccpp.service                             enabled
abrt-oops.service                             enabled
abrt-vmcore.service                           enabled
abrt-xorg.service                             enabled
abrtd.service                                 enabled
...
wpa_supplicant.service                        disabled
ypbind.service                                disabled

208 unit files listed.
```

6.2.2. サービスステータスの表示

システムサービスに対応するサービスユニットについての詳細情報を表示するには、シェルプロンプトで以下を入力します。

```
systemctl status name.service
```

name を調べたいサービスユニット名 (たとえば、**gdm**) に置き換えます。このコマンドでは、選択されたサービスユニット名の後に、その説明と 表6.5「利用可能なサービスユニットの情報」 にある 1 つ以上のフィールド、さらに **root** ユーザーが実行している場合には、最新のログエントリーが表示されます。

表6.5 利用可能なサービスユニットの情報

フィールド	詳細
Loaded	サービスユニットが読み込まれているかどうか、ユニットファイルへの絶対パス、ユニットが有効かどうかについての説明。
Active	サービスユニットが実行中かどうかの情報の後に、タイムスタンプが続きます。
Main PID	対応するシステムサービスの PID の後に、その名前が続きます。
Status	対応するシステムサービスについての追加情報。
Process	関連プロセスについての追加情報。
CGroup	関連する Control Group についての追加情報。

特定のサービスユニットが実行中かどうかチェックするだけの場合は、以下のコマンドを実行します。

```
systemctl is-active name.service
```

同様に、特定のサービスユニットが有効かどうかを判断するには、以下を入力します。

```
systemctl is-enabled name.service
```

systemctl is-active および **systemctl is-enabled** は両方とも、指定されたサービスユニットが実行中もしくは有効な場合に、**0** の終了ステータスを返すことに注意してください。現在読み込み済みのサービスユニットすべてを一覧表示する方法については、 「サービスの一覧表示」 を参照してください。

例6.2 サービスステータスの表示

GNOME Display Manager のサービスユニット名は **gdm.service** になります。このサービスユニットの現在のステータスを判断するには、シェルプロンプトで以下を入力します。

```
~]# systemctl status gdm.service
gdm.service - GNOME Display Manager
   Loaded: loaded (/usr/lib/systemd/system/gdm.service; enabled)
   Active: active (running) since Thu 2013-10-17 17:31:23 CEST; 5min
ago
 Main PID: 1029 (gdm)
   CGroup: /system.slice/gdm.service
           ├─1029 /usr/sbin/gdm
           ├─1037 /usr/libexec/gdm-simple-slave --display-id
/org/gno...
           └─1047 /usr/bin/Xorg :0 -background none -verbose -auth
/r...

Oct 17 17:31:23 localhost systemd[1]: Started GNOME Display Manager.
```

6.2.3. サービスの開始

システムサービスに対応するサービスユニットを開始するには、**root** でシェルプロンプトに以下を入力します。

```
systemctl start name.service
```

name を開始するサービスユニット名 (たとえば、**gdm**) に置き換えます。このコマンドは、選択されたサービスを現行セッションで開始します。起動時にサービスユニットを開始するようにする方法は、「サービスの有効化」 を参照してください。特定のサービスユニットのステータスを判断する方法については、「サービスステータスの表示」を参照してください。

例6.3 サービスの開始

Apache HTTP Server のサービスユニット名は **httpd.service** になります。このサービスユニットをアクティブ化して、現行セッションで **httpd** デーモンを開始するには、**root** で以下のコマンドを実行します。

```
~]# systemctl start httpd.service
```

6.2.4. サービスの停止

システムサービスに対応するサービスユニットを停止するには、**root** でシェルプロンプトに以下を入力します。

```
systemctl stop name.service
```

name を停止するサービスユニット名 (たとえば、**bluetooth**) に置き換えます。このコマンドは、選択されたサービスを現行セッションで停止します。起動時にサービスユニットを無効にして開始しないようにする方法は、「サービスの無効化」を参照してください。特定のサービスユニットのステータスを判断する方法については、「サービスステータスの表示」を参照してください。

例6.4 サービスの停止

bluetoothd デーモンのサービスユニット名は**bluetooth.service** になります。このサービスユニットを無効にして **bluetoothd** デーモンを現行セッションで停止するには、**root** で以下のコマンドを実行します。

```
~]# systemctl stop bluetooth.service
```

6.2.5. サービスの再開

システムサービスに対応するサービスユニットを再開するには、**root** でシェルプロンプトに以下を入力します。

```
systemctl restart name.service
```

name を再開するサービスユニット名 (たとえば、**httpd**) に置き換えます。このコマンドは、選択されたサービスを現行セッションで停止し、即座に再起動します。重要な点は、選択されたサービスユニットが実行中でない場合、このコマンドがそのサービスユニットを起動するということです。対応するサービスがすでに実行中の場合にのみ、サービスユニットを再起動するように systemd に指示するには、**root** で以下のコマンドを実行します。

```
systemctl try-restart name.service
```

特定のシステムサービスでは、サービスの実行を中断することなく設定の再読み込みが可能です。これを実行するには、**root** で以下を入力します。

```
systemctl reload name.service
```

この機能をサポートしないシステムサービスは、このコマンド自体を無視することに注意してください。便宜上、代わりに **systemctl** コマンドは、この機能をサポートしないサービスを再起動する**reload-or-restart** および **reload-or-try-restart** コマンドもサポートします。特定のサービスユニットのステータスを判断する方法は、「サービスステータスの表示」を参照してください。

例6.5 サービスの再開

ユーザーが不要なエラーメッセージや部分的に表示される Web ページに遭遇しないようにするため、Apache HTTP Server では設定を再起動せずかつ処理されたリクエストをアクティブに妨害せずに、設定の編集および再読み込みができます。これを行うには、**root** でシェルプロンプトに以下を入力します。

```
~]# systemctl reload httpd.service
```

6.2.6. サービスの有効化

システムサービスに対応するサービスユニットを起動時に自動的に起動するように設定するには、**root** で

シェルプロンプトに以下を入力します。

```
systemctl enable name.service
```

name を有効にするサービスユニット名 (たとえば、**httpd**) に置き換えます。このコマンドは、選択されたサービスユニットの **[Install]** セクションを読み込み、**/etc/systemd/system/** ディレクトリーおよびそのサブディレクトリーにある**/usr/lib/systemd/system/*name*.service** ファイルへの適切なシンボリックリンクを作成します。ただし、このコマンドは既存のリンクの上書きはしません。シンボリックリンクが確実に再度作成されるようにするには、**root** で以下のコマンドを使用します。

```
systemctl reenable name.service
```

このコマンドは、選択されたサービスユニットを無効にし、即座に再度有効にします。特定のサービスユニットが起動時に有効になるかどうかを判断する方法については、「サービスステータスの表示」を参照してください。現行セッションでサービスを開始する方法については、「サービスの開始」を参照してください。

例6.6 サービスの有効化

Apache HTTP Server が起動時に自動的に開始するように設定するには、**root** で以下のコマンドを実行します。

```
~]# systemctl enable httpd.service
ln -s '/usr/lib/systemd/system/httpd.service'
'/etc/systemd/system/multi-user.target.wants/httpd.service'
```

6.2.7. サービスの無効化

システムサービスに対応するサービスユニットを起動時に自動的に起動しないようにするには、**root** でシェルプロンプトに以下を入力します。

```
systemctl disable name.service
```

name を無効にするサービスユニット名 (たとえば、**bluetooth**) に置き換えます。このコマンドは、選択されたサービスユニットの **[Install]** セクションを読み込み、**/etc/systemd/system/** ディレクトリーおよびそのサブディレクトリーから**/usr/lib/systemd/system/*name*.service** ファイルへの適切なシンボリックリンクを削除します。さらに、サービスユニットにマスクをして、手動で開始したり、別のサービスがこれを開始することを防ぐことができます。これを実行するには、**root** で以下のコマンドを実行します。

```
systemctl mask name.service
```

このコマンドは、**/etc/systemd/system/*name*.service** ファイルを **/dev/null** へのシンボリックリンクに置き換え、実際のユニットファイルが systemd にアクセスできないようにします。このアクセスを元に戻してサービスユニットをアンマスクするには、**root** で以下を入力します。

```
systemctl unmask name.service
```

特定のサービスユニットが起動時に有効になるかどうかを判断する方法については、「サービスステータスの表示」を参照してください。現行セッションでサービスを停止する方法については、「サービスの停止」を参照してください。

例6.7 サービスの無効化

例6.4「サービスの停止」では、現行セッションで **bluetooth.service** ユニットを停止する方法を説明しています。このサービスユニットが起動時に開始しないようにするには、**root** でシェルプロンプトに以下を入力します。

```
~]# systemctl disable bluetooth.service
rm '/etc/systemd/system/dbus-org.bluez.service'
rm '/etc/systemd/system/bluetooth.target.wants/bluetooth.service'
```

6.3. systemd ターゲットでの作業

以前のバージョンの Red Hat Enterprise Linux は SysV init もしくは Upstart と配布されており、特定モードのオペレーションを表す事前定義の *ランレベル* を実装していました。これらのランレベルは 0 から 6 までの数字で表示され、システム管理者が特定のランレベルを有効にすると実行されるシステムサービスの選択によって定義されていました。Red Hat Enterprise Linux 7 では、ランレベルの概念は *systemd targets* に代わっています。

Systemd targets は *target units* で表されます。Target units は **.target** ファイル拡張子で終わり、その唯一の目的は依存関係の連鎖で他の systemd units をグループ化することです。たとえば、グラフィカルセッションの開始に使用される **graphical.target** ユニットは、GNOME Display Manager (**gdm.service**) や Accounts Service (**accounts-daemon.service**) といったシステムサービスを開始するとともに、**multi-user.target** ユニットもアクティブ化します。同様に **multi-user.target** ユニットは、NetworkManager (**NetworkManager.service**) や D-Bus (**dbus.service**) といった他の必須のシステムサービスを開始して、かつ **basic.target** という別の target unit をアクティブ化します。

Red Hat Enterprise Linux 7 では、以前のシステムのリリースにおける標準ランレベルとほぼ似通った多くの定義済みターゲットが同梱されています。互換性目的で、これらのターゲットを SysV ランレベルに直接マッピングするエイリアスも提供されています。表6.6「SysV ランレベルと systemd ターゲットの比較」では、SysV ランレベルの完全なリストと対応する systemd ターゲットが表示されています。

表6.6 SysV ランレベルと systemd ターゲットの比較

ランレベル	ターゲットユニット	詳細
0	runlevel0.target、poweroff.target	システムをシャットダウンし、電源を切ります。
1	runlevel1.target、rescue.target	レスキューシェルを設定します。
2	runlevel2.target、multi-user.target	非グラフィカルな複数ユーザーシステムを設定します。
3	runlevel3.target、multi-user.target	非グラフィカルな複数ユーザーシステムを設定します。
4	runlevel4.target、multi-user.target	非グラフィカルな複数ユーザーシステムを設定します。
5	runlevel5.target、graphical.target	グラフィカルな複数ユーザーシステムを設定します。
6	runlevel6.target、reboot.target	システムをシャットダウンして再起動します。

systemd ターゲットを表示、変更、もしくは設定するには、表6.7「SysV init コマンドと systemctl の比較」 および以下のセクションで説明するように**systemctl** ユーティリティーを使用します。**runlevel** および **telinit** の各コマンドはシステムで利用可能なままで、期待通りに機能しますが、これらが含まれているのは互換性が目的であり、使用は避けてください。

表6.7 SysV init コマンドと systemctl の比較

古いコマンド	新しい コマンド	詳細
runlevel	**systemctl list-units --type target**	現在読み込まれているターゲットユニットを表示します。
telinit *runlevel*	**systemctl isolate *name*.target**	現在のターゲットを変更します。

6.3.1. デフォルトターゲットの表示

デフォルトでどのターゲットユニットが使用されるかを決定するには、以下のコマンドを実行します。

```
systemctl get-default
```

このコマンドは **/etc/systemd/system/default.target** にあるシンボリックリンクを解決し、その結果を表示します。デフォルトのターゲットの変更方法については、「デフォルトターゲットの変更」 を参照してください。現在読み込まれているターゲットユニットを一覧表示する方法については、「現在のターゲットの表示」 を参照してください。

例6.8 デフォルトターゲットの表示

デフォルトのターゲットユニットを表示するには、以下を入力します。

```
~]$ systemctl get-default
graphical.target
```

6.3.2. 現在のターゲットの表示

読み込み済みの現在のターゲットユニットをすべて一覧表示するには、シェルプロンプトで以下のコマンドを入力します。

```
systemctl list-units --type target
```

このコマンドは、各ターゲットユニットの完全な名前 (**UNIT**) を表示し、その後にユニットが読み込み済みかどうか (**LOAD**)、高レベル (**ACTIVE**) および低レベル (**SUB**) のユニットのアクティベーション状態、および簡単な説明 (**DESCRIPTION**) が続きます。

デフォルトでは、**systemctl list-units** コマンドはアクティブなユニットのみを表示します。状態に関係なくすべての読み込み済みユニットを表示したい場合は、**--all** または **-a** のコマンドラインオプションを付けてコマンドを実行します。

```
systemctl list-units --type target --all
```

デフォルトのターゲットの表示方法については、「デフォルトターゲットの表示」 を参照してください。現在のターゲットの変更方法については、「現在のターゲットの変更」 を参照してください。

例6.9 現在のターゲットの表示

現在読み込み済みの全ターゲットユニットを一覧表示するには、以下のコマンドを実行します。

```
~]$ systemctl list-units --type target
UNIT                   LOAD   ACTIVE SUB    DESCRIPTION
basic.target           loaded active active Basic System
cryptsetup.target      loaded active active Encrypted Volumes
getty.target           loaded active active Login Prompts
graphical.target       loaded active active Graphical Interface
local-fs-pre.target    loaded active active Local File Systems (Pre)
local-fs.target        loaded active active Local File Systems
multi-user.target      loaded active active Multi-User System
network.target         loaded active active Network
paths.target           loaded active active Paths
remote-fs.target       loaded active active Remote File Systems
sockets.target         loaded active active Sockets
sound.target           loaded active active Sound Card
spice-vdagentd.target loaded active active Agent daemon for Spice
guests
swap.target            loaded active active Swap
sysinit.target         loaded active active System Initialization
time-sync.target       loaded active active System Time Synchronized
timers.target          loaded active active Timers

LOAD   = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of
SUB.
SUB    = The low-level unit activation state, values depend on unit
type.

17 loaded units listed. Pass --all to see loaded but inactive units,
too.
To show all installed unit files use 'systemctl list-unit-files'.
```

6.3.3. デフォルトターゲットの変更

システムがデフォルトで異なるターゲットユニットを使用するよう設定するには、**root** でシェルプロンプトに以下を入力します。

```
systemctl set-default name.target
```

name をデフォルトで使用したいターゲットユニット名で (たとえば、**multi-user**) 置き換えます。このコマンドは、**/etc/systemd/system/default.target** ファイルを **/usr/lib/systemd/system/*name*.target** へのシンボリックリンクで置き換えます。ここでの*name* は、使用するターゲットユニット名になります。現在のターゲットの変更方法については、「現在のターゲットの変更」 を参照してください。

例6.10 デフォルトターゲットの変更

システムがデフォルトで **multi-user.target** ユニットを使用するよう設定するには、**root** で以下のコマンドを使用します。

```
~]# systemctl set-default multi-user.target
rm '/etc/systemd/system/default.target'
ln -s '/usr/lib/systemd/system/multi-user.target'
'/etc/systemd/system/default.target'
```

6.3.4. 現在のターゲットの変更

現行のセッションで異なるターゲットユニットに変更するには、シェルプロンプトで **root** として以下を入力します。

```
systemctl isolate name.target
```

name を使用したいターゲットユニット名で (たとえば、**multi-user**) 置き換えます。このコマンドは、*name* という名前の付いたターゲットユニットとすべての依存ユニットを開始し、即座にその他すべてを停止します。デフォルトのターゲットの変更方法については、「デフォルトターゲットの変更」 を参照してください。現在、読み込み済みのターゲットユニットすべてを一覧表示する方法については、「現在のターゲットの変更」 を参照してください。

例6.11 現在のターゲットの変更

グラフィカルユーザーインターフェイスをオフにして、現行セッションで **multi-user.target** ユニットに変更するには、**root** で以下のコマンドを使用します。

```
~]# systemctl isolate multi-user.target
```

6.3.5. レスキューモードへの変更

*レスキューモード*は、便利なシングルユーザーモードを提供し、通常の起動プロセスを完了できない状況におけるシステムの修復を可能にします。レスキューモードでは、システムはすべてのローカルファイルシステムのマウントといくつかの重要なシステムサービスの開始を試みますが、ネットワークインターフェースのアクティブ化や、システムに同時に他のユーザーによるログインを許可したりすることはしません。Red Hat Enterprise Linux 7 では、レスキューモードは *シングルユーザーモード*と同等であり、root パスワードを必要とします。

現在のターゲットを変更し、現行セッションでレスキューモードに入るには、**root** でシェルプロンプトに以下を入力します。

```
systemctl rescue
```

このコマンドは **systemctl isolate rescue.target** と似ていますが、現在システムにログインしているすべてのユーザーに通知メッセージも送信します。systemd がこのメッセージを送信しないようにするには、このコマンドに **--no-wall** オプションを付けて実行します。

```
systemctl --no-wall rescue
```

緊急モードに入る方法については、「緊急モードへの変更」 を参照してください。

例6.12 レスキューモードへの変更

現行セッションでレスキューモードに入るには、**root** で以下のコマンドを実行します。

```
~]# systemctl rescue

Broadcast message from root@localhost on pts/0 (Fri 2013-10-25 18:23:15
CEST):

The system is going down to rescue mode NOW!
```

6.3.6. 緊急モードへの変更

*緊急モード*は可能な限り最小限の環境を提供し、レスキューモードに入れないシステムの状態におけるシステムの修復を可能にします。緊急モードでは、システムは root ファイルシステムを読み込み専用でマウントし、他のローカルファイルシステムのマウントは試みません。また、ネットワークインターフェイスのアクティブ化も行わず、限定的な必須サービスのみを起動します。Red Hat Enterprise Linux 7 では、緊急モードに root パスワードを必要とします。

現在のターゲットを変更し、緊急モードに入るには、**root** でシェルプロンプトに以下を入力します。

```
systemctl emergency
```

このコマンドは **systemctl isolate emergency.target** と似ていますが、現在システムにログインしているすべてのユーザーに通知メッセージも送信します。systemd がこのメッセージを送信しないようにするには、このコマンドに **--no-wall** オプションを付けて実行します。

```
systemctl --no-wall emergency
```

レスキューモードに入る方法については、「レスキューモードへの変更」を参照してください。

例6.13 緊急モードへの変更

現在システムにログインしている全ユーザーにメッセージを送信せずに緊急モードに入るには、**root** で以下のコマンドを実行します。

```
~]# systemctl --no-wall emergency
```

6.4. システムのシャットダウン、サスペンド、休止状態

Red Hat Enterprise Linux 7 では、**systemctl** ユーティリティーがこれまでのバージョンの Red Hat Enterprise Linux システムで使われていた多くの電源管理コマンドに置き換わっています。表6.8「電源管理コマンドと systemctl の比較」に表示されているコマンドは互換性を理由にまだ利用可能ですが、可能な場合は **systemctl** の使用が推奨されます。

表6.8 電源管理コマンドと systemctl の比較

古いコマンド	新しい コマンド	詳細
halt	**systemctl halt**	システムを停止します。
poweroff	**systemctl poweroff**	システムの電源を切ります。
reboot	**systemctl reboot**	システムを再起動します。
pm-suspend	**systemctl suspend**	システムをサスペンドします。
pm-hibernate	**systemctl hibernate**	システムを休止状態にします。

古いコマンド	新しい コマンド	詳細
`pm-suspend-hybrid`	`systemctl hybrid-sleep`	システムを休止状態にしてサスペンドします。

6.4.1. システムのシャットダウン

システムをシャットダウンしてマシンの電源を切るには、 **root** でシェルプロンプトに以下を入力します。

```
systemctl poweroff
```

マシンの電源を切らずにシステムをシャットダウンして停止するには、**root** で以下のコマンドを実行します。

```
systemctl halt
```

デフォルトでは、このコマンドはどちらも systemd が 現在システムにログインしているすべてのユーザーに通知メッセージを送信します。systemd がこのメッセージを送信しないようにするには、コマンドに **--no-wall** オプションを付けて実行します。例を示します。

```
systemctl --no-wall poweroff
```

6.4.2. システムの再起動

システムを再起動するには、**root** で以下のコマンドを実行します。

```
systemctl reboot
```

デフォルトでは、このコマンドは systemd が 現在システムにログインしているすべてのユーザーに通知メッセージを送信します。systemd がこのメッセージを送信しないようにするには、コマンドに **--no-wall** オプションを付けて実行します。

```
systemctl --no-wall reboot
```

6.4.3. システムの一時停止

システムをサスペンドするには、**root** でシェルプロンプトに以下を入力します。

```
systemctl suspend
```

このコマンドは、システムの状態を RAM に保存し、RAM モジュールを除いてマシンのほとんどのデバイスの電源を切ります。マシンの電源を戻すと、システムは再起動せずに RAM からその状態を復元します。システムの状態がハードディスク上ではなく RAM に保存されるので、システムのサスペンドモードからの復元は、休止状態からの復元よりも大幅に速くなります。ただし結果として、システムをサスペンドした状態は、停電に対して脆弱となります。

システムを休止状態にする方法については、「システムを休止状態にする」を参照してください。

6.4.4. システムを休止状態にする

システムを休止状態にするには、**root** でシェルプロンプトに以下を入力します。

```
systemctl hibernate
```

このコマンドは、システムの状態をハードディスクドライブに保存し、マシンの電源を切ります。マシンの電源を戻すと、システムは再起動せずに保存されたデータからその状態を復元します。システムの状態が RAM ではなくハードディスク上に保存されるので、マシンは RAM モジュールに電力を維持する必要がありません。ただしその結果、システムの休止状態からの復元は、サスペンドモードからの復元よりも大幅に遅くなります。

システムを休止状態にしてサスペンドするには、**root** で以下のコマンドを実行します。

```
systemctl hybrid-sleep
```

システムをサスペンドする方法については、「システムの一時停止」を参照してください。

6.5. リモートマシン上での systemd の制御

systemd システムおよびサービスマネージャーをローカルで制御することに加え、**systemctl** ユーティリティーでは、SSH プロトコルを使ってリモートマシン上で実行している systemd と対話操作することができます。**sshd** サービスがリモートマシン上で実行中であれば、**systemctl** コマンドに **--host** または **-H** コマンドラインオプションを付けて実行すると、このマシンに接続できます。

```
systemctl --host user_name@host_name command
```

user_name をリモートのユーザー名で、*host_name* をマシンのホスト名で、**command** を上記の **systemctl** コマンドのいずれかでそれぞれ置き換えます。指定されたユーザーが SSH プロトコルを使ってリモートアクセスできるようにリモートマシンを設定する必要があることに注意してください。SSH サーバーの設定に関する詳細情報は、*7章OpenSSH* を参照してください。

例6.14 リモート管理

server-01.example.com という名前のリモートマシンに**root** ユーザーとしてログインし、**httpd.service** ユニットの現在の状態を判断するには、シェルプロンプトに以下を入力します。

```
~]$ systemctl -H root@server-01.example.com status httpd.service
root@server-01.example.com's password:
httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled)
   Active: active (running) since Fri 2013-11-01 13:58:56 CET; 2h 48min
ago
 Main PID: 649
   Status: "Total requests: 0; Current requests/sec: 0; Current
traffic:   0 B/sec"
   CGroup: /system.slice/httpd.service
```

6.6. その他のリソース

systemd およびその Red Hat Enterprise Linux 上での使用に関する詳細情報は、以下のリソースを参照してください。

インストールされているドキュメント

- **systemctl**(1) — **systemctl** コマンドラインユーティリティーの man ページでは、サポートされるオプションとコマンドの完全なリストが提供されます。
- **systemd**(1) — **systemd** システムおよびサービスマネージャーの man ページでは、その概念に関する詳細情報が提供され、利用可能なコマンドラインオプションと環境変数、サポートされる設定ファイルとディレクトリー、認識されるシグナル、および利用可能なカーネルオプションが説明されています。
- **systemd.unit**(5) — **systemd.unit** の man ページでは、systemd ユニットファイルについての詳細情報と、利用可能なすべての設定オプションが説明されています。
- **systemd.service**(5) — **systemd.service** の man ページでは、サービスユニットファイルのフォーマットが説明されています。
- **systemd.target**(5) — **systemd.target** の man ページでは、ターゲットユニットファイルのフォーマットが説明されています。

オンラインのドキュメント

- Red Hat Enterprise Linux 7 Networking Guide — Red Hat Enterprise Linux 7 の 『Networking Guide』 は、このシステムにおけるネットワークインターフェイスやネットワーク、ネットワークサービスの設定および管理に関する情報が説明されています。**hostnamectl** ユーティリティーの概要のほか、これを使ってコマンドラインでホスト名を表示、設定する方法が説明されています。また、選択されたホスト名およびドメイン名についての重要な情報も提供されています。
- Red Hat Enterprise Linux 7 Desktop Migration and Administration Guide — Red Hat Enterprise Linux 7 の 『Desktop Migration and Administration Guide』 は、本システム上での GNOME 3 デスクトップの移行計画、導入、設定および管理について説明しています。**logind** サービスの概要とその最重要機能を挙げているほか、**loginctl** ユーティリティーを使ってアクティブなセッションを一覧表示し、マルチシートサポートを有効にする方法を説明しています。
- Red Hat Enterprise Linux 7 SELinux User's and Administrator's Guide — Red Hat Enterprise Linux 7 の 『SELinux User's and Administrator's Guide』 では、SELinux の原則と、SELinux を Apache HTTP Server や Postfix、PostgreSQL、または OpenShift などのさまざまなサービスと設定して使用する方法が詳細に説明されています。また、SELinux アクセスパーミッションを systemd が管理するシステムサービス用に設定する方法も説明しています。
- Red Hat Enterprise Linux 7 インストールガイド — Red Hat Enterprise Linux 7 の 『インストールガイド』 は、AMD64 および Intel 64 システム、64-bit IBM Power Systems サーバー、および IBM System z にシステムをインストールする方法を説明しています。また、キックスタートインストール、PXE インストール、および VNC プロトコルを使ったインストールなどの高度なインストール方法もカバーされています。さらに、インストール後の一般的なタスクと、レスキューモードへの起動方法や root パスワードの回復方法などの詳細な手順を含むインストールの問題に関するトラブルシュートについても説明されています。
- Red Hat Enterprise Linux 7 Security Guide — Red Hat Enterprise Linux 7 の 『Security Guide』 は、ローカルおよびリモートからの侵入、悪用、悪意のある行為に対してワークステーションおよびサーバーを保護するプロセスとプラクティスを学習する際にユーザーおよび管理の役に立ちます。また、重大なシステムサービスを保護する方法についても説明しています。
- systemd Home Page — このプロジェクトのホームページでは、systemd についての詳細情報が提供されています。

関連項目

- 1章*システムロケールおよびキーボード設定*では、システムロケールとキーボードのレイアウトを管理する方法を説明しています。**localectl** ユーティリティーを使って現在のロケールを表示、利用可能なロケールを一覧表示、コマンドラインでシステムロケールを設定、現在のキーボードレイアウトの表

示、利用可能なキーマップの一覧表示、コマンドラインで特定のキーボードレイアウトを有効にする方法が説明されています。

- 2章*日付と時刻の設定*では、システムの日時を管理する方法を説明しています。リアルタイムクロックとシステムクロックの違いや、**timedatectl** ユーティリティーを使って現在のシステムクロックの設定を表示、日時を設定、タイムゾーンを変更、およびシステムクロックをリモートサーバーと同期させる方法が説明されています。

- 4章*権限の取得*では、**su** および **sudo** コマンドを使って管理者権限を取得する方法が説明されています。

- 7章*OpenSSH* は、SSH サーバーの設定方法や、**ssh**、**scp**、および **sftp** クライアントユーティリティーを使ってこのサーバーにアクセスする方法が説明されています。

- 18章*ログファイルの表示と管理*では、journald の概要が提供されています。ジャーナルの説明と **journald** サービスの概要のほか、**journalctl** ユーティリティーを使ってログエントリーを表示、ライブ表示モードへの切り替え、ログエントリーのフィルター方法を説明しています。さらに、この章では、root 以外のユーザーにシステムログへのアクセスを許可し、一貫性のあるログファイルの保存を可能にする方法も説明されています。

第7章 OpenSSH

SSH (Secure Shell) は、クライアント/サーバーアーキテクチャーを使用する 2 つのシステム間でのセキュアな通信を容易にし、ユーザーがリモートでサーバーのホストシステムにログインできるようにするプロトコルです。**FTP** や **Telnet** などの他のリモート通信プロトコルとは異なり、SSH はログインセッションを暗号化するため、接続で侵入者が暗号化されていないパスワードを入手することが困難になります。

ssh プログラムは、**telnet** や **rsh** などのリモートホストへのログインに使用される旧式でセキュリティーの低いターミナルアプリケーションに代わるものとして設計されています。また、**scp** と呼ばれる関連プログラムが、ホスト間でファイルをコピーするために設計された **rcp** のような旧式プログラムの代わりとなります。これら旧式アプリケーションはクライアントとサーバー間で送信するパスワードを暗号化しないので、可能な限り使用を避けるようにしてください。リモートシステムへのログインにセキュアな方法を使用することで、クライアントシステムとリモートホストの両方に対するリスクが低減されます。

Red Hat Enterprise Linux には、一般的な OpenSSH パッケージ (*openssh*) と共に、OpenSSH サーバー (*openssh-server*) およびクライアント (*openssh-clients*) パッケージが含まれています。OpenSSH パッケージでは、いくつかの重要な暗号化ライブラリーをインストールし、OpenSSH の暗号化通信を可能にする OpenSSL パッケージ (*openssl-libs*) が必要になる点に注意してください。

7.1. SSH プロトコル

7.1.1. SSH を使用する理由

潜在的な侵入者は、ネットワークトラフィックの中断、傍受、経路変更を可能にする様々なツールを自由に駆使して、システムに侵入します。一般的には、これらの脅威は以下のとおり分類できます。

2 システム間の通信の傍受

攻撃者は、ネットワーク上で通信を行う二者の間のどこかに潜み、両者間で渡される情報をコピーしている可能性があります。攻撃者は情報を傍受して保持する、または情報を改ざんして対象となる受信者に送信する場合があります。

このような攻撃は、通常 *パケットスニファ* を使用して行われます。パケットスニファは、ネットワークを通過するパケットをキャプチャしてその内容を分析するかなり一般的なネットワークユーティリティーです。

特定のホストの偽装

攻撃者のシステムは、送信の対象となる受信者を装うように設定されます。この戦略が成功すると、ユーザーのシステムは不正なホストと通信していることに気がつかないままとなります。

この攻撃は、*DNS ポイズニング* として知られる手法か *IP スプーフィング* と呼ばれる手法を用いて実行されます。前者の場合、侵入者はクラックされた DNS サーバーを使用して、クライアントシステムを不当に複製されたホストへポイントします。後者の場合は、侵入者は信頼されたホストから送信されたように見せかけた偽装ネットワークパケットを送信します。

いずれの手法でも、潜在的な機密情報を傍受することが可能です。その傍受が悪意のある理由で行われる場合には、悲惨な結果をもたらしかねません。リモートシェルログインとファイルコピー用に SSH を使用すると、こうしたセキュリティー脅威を大幅に軽減することができます。これは、SSH クライアントとサーバーがデジタル署名を使用してそれぞれの ID を確認するためです。さらに、クライアントシステムとサーバーシステム間の全通信は暗号化されます。各パケットはローカルシステムとリモートシステムのみに知られている鍵を使用して暗号化されるため、通信のいずれか一方の ID をスプーフィングする試みは成功しません。

7.1.2. 主な機能

SSH プロトコルは、以下のような保護手段を提供します。

対象のサーバーとして装うことができません

初回接続後に、クライアントは以前に接続したサーバーと同じサーバーに接続していることを確認できます。

認証情報の取得はできません

クライアントは、強力な 128 ビット暗号化を使用してサーバーへ認証情報を送信します。

通信の傍受はできません

セッション中に送受信された全データは、128 ビット暗号化を使用して転送されるため、傍受された送信データの暗号解読と読み取りは非常に困難になります。

さらに、SSH プロトコルは以下のようなオプションも提供します。

ネットワーク上でグラフィカルアプリケーションを使用するセキュアな手段を提供します

クライアントは、*X11 転送* と呼ばれる手法を使用して、サーバーから*X11* (*X Window System*) アプリケーションを転送することが可能です。

セキュアでないプロトコルをセキュアにする手段を提供します

SSH プロトコルは、送受信するものすべてを暗号化します。SSH サーバーは、*ポート転送* と呼ばれる手法を使用して、POP のようなセキュアでないプロトコルをセキュアにするための経路となり、システムとデータ全体のセキュリティーを強化することができます。

セキュアなチャンネルを作成できます

OpenSSH サーバーとクライアントは、サーバーとクライアントマシン間のトラフィックに対し仮想プライベートネットワークに似たトンネルを作成するよう設定できます。

Kerberos 認証に対応します

OpenSSH のサーバーとクライアントは、Kerberos ネットワーク認証プロトコルの GSSAPI (Generic Security Services Application Program Interface: 汎用セキュリティサービス API) 実装を使用して認証を行うよう設定できます。

7.1.3. プロトコルのバージョン

現在、SSH には バージョン 1 と新しいバージョン 2 の 2 種類があります。Red Hat Enterprise Linux の OpenSSH スイートでは、SSH バージョン 2 を使用します。このバージョンは、バージョン 1 での既知のエクスプロイトに対して脆弱性のない拡張された鍵交換アルゴリズムを使用します。ただし、互換性上の理由から OpenSSH スイートはバージョン 1 の接続にも対応しています。

重要

接続のセキュリティを最大限にするために、可能な限り SSH バージョン 2 との互換性があるサーバーとクライアントのみを使用することが推奨されます。

7.1.4. SSH 接続のイベントシーケンス

以下にあげる一連のイベントは、2 つのホスト間で行われる SSH 通信の整合性を保護するために役立ちます。

1. 暗号化ハンドシェイクが行われ、クライアントは正しいサーバーと通信していることを確認できます。
2. クライアントとリモートホスト間の接続のトランスポート層は、対称暗号方式を使用して暗号化されます。
3. クライアントはサーバーに対して自己認証します。
4. リモートクライアントは、暗号化された接続でリモートホストと対話します。

7.1.4.1. トランスポート層

トランスポート層の主な役割は、認証時およびその後の通信中における 2 つのホスト間の安全でセキュアな通信を容易にすることです。トランスポート層は、データの暗号化と復号化を処理し、データパケットの送受信時にその整合性を保護することでその役割を果たします。また、トランスポート層は、情報を圧縮して転送を高速化します。

SSH クライアントがサーバーにコンタクトすると、基本情報が交換されるため両システムはトランスポート層を適正に構築することができます。以下は、こうした基本情報の交換中に発生するステップです。

- 鍵が交換されます
- 公開鍵暗号化アルゴリズムが決定されます
- 対称暗号化アルゴリズムが決定されます
- メッセージ認証アルゴリズムが決定されます
- ハッシュアルゴリズムが決定されます

鍵交換の間、サーバーは一意の *ホスト鍵* を用いて、クライアントに対して自己識別を行います。クライアントがこの特定のサーバーと過去に通信したことがなければ、サーバーのホスト鍵はクライアントには未知であり、接続は成立しません。OpenSSH はこの問題に対処するためにサーバーのホスト鍵を承認します。これは、ユーザーが通知を受けて新規のホスト鍵を受け取り検証した後に行われます。それ以降の接続では、サーバーのホスト鍵は、クライアント上に保存されているバージョンと照合され、クライアントが実際に目的のサーバーと通信していることを確信できます。この後に、ホスト鍵が一致しなくなった場合には、ユーザーは接続前にクライアントの保存してあるバージョンを削除する必要があります。

警告

ローカルシステムは、対象サーバーと攻撃者が設定した偽サーバーとの違いを認識しないため、攻撃者は初回コンタクト中に SSH サーバーをマスカレードすることが可能です。この問題を防ぐために、初回接続の前かホスト鍵の不一致が発生した場合には、サーバー管理者へ連絡して新しい SSH サーバーの整合性を確認してください。

SSH は、ほとんどすべての公開鍵アルゴリズムまたはエンコード形式に対応するように設計されています。初回の鍵交換で、交換に使用されるハッシュ値と共有秘密値が作成された後、2 つのシステムは新しい鍵とアルゴリズムの計算を直ちに開始して、認証と今後この接続で送信されるデータを保護します。

所定の鍵とアルゴリズムを使用して一定量のデータ (正確な量は SSH 実装により異なる) が送信された後に、もう 1 回鍵交換が行われてハッシュ値と新しい共有秘密値の別のセットが生成されます。攻撃者がハッシュ値と共有秘密値を判別できたとしても、その情報が役に立つのは限られた時間のみです。

7.1.4.2. 認証

トランスポート層が 2 つのシステム間で情報を渡すためのセキュアなトンネルを構築すると 、サーバーは秘密鍵でエンコードされた署名の使用やパスワードの入力など、サポートされている別の認証方法をクライアントに伝えます。次に、クライアントはサポートされているいずれかの方法を使ってサーバーに対して自己認証を試みます。

SSH サーバーとクライアントは、異なるタイプの認証を採用できるように設定可能なため、双方の制御が最適化されます。サーバーはそのセキュリティーモデルに基づき、サポートする暗号化方法を決定することができ、クライアントは利用可能なオプションの中から試行する認証方法の順番を選択できます。

7.1.4.3. チャンネル

SSH トランスポート層での認証に成功した後は、*多重化* [2] と呼ばれる手法により複数のチャンネルが開かれます。これらの各チャンネルは、異なるターミナルセッションと転送された X11 セッションの通信を処理します。

クライアントとサーバーの両方で、新しいチャンネルを作成できます。その後、各チャンネルに別々の番号が接続の両端に割り当てられます。クライアントが新しいチャンネルを開こうとする時には、クライアントは要求と共にチャンネル番号を送信します。この情報はサーバーにより保存され、そのチャンネルに通信を移動するために使用されます。これは、異なるタイプのセッションが相互に影響しないように、あるセッションの終了時にそのチャンネルが SSH による一次接続を停止せずに閉じることができるようにするためです。

また、チャンネルは *フロー制御* もサポートしているため規則的な方法でデータを送受信することができます。この方法では、チャンネルが開いているというメッセージをクライアントが受信するまで、データはチャンネル上で送信されません。

クライアントが要求するサービスのタイプとユーザーがネットワークに接続される方法に応じて、クライアントとサーバーは、各チャンネルの特性を自動的にネゴシエートします。これにより、プロトコルの基本インフラストラクチャーを変更しなくても、異なるタイプのリモート接続を非常に柔軟に処理することができます。

7.2. OpenSSH の設定

7.2.1. 設定ファイル

設定ファイルには、クライアントプログラム用 (**ssh**、**scp** および **sftp**) とサーバー用 (**sshd** デーモン) の異なる 2 つのセットがあります。

システム全体の SSH 設定情報は、表7.1「システム全体の設定ファイル」にあるように、**/etc/ssh/** ディレクトリー内に格納されています。ユーザー固有の SSH 設定情報は、ユーザーのホームディレクトリー内の **~/.ssh/** に格納されています。詳細は、表7.2「ユーザー固有の設定ファイル」に記載しています。

表7.1 システム全体の設定ファイル

ファイル	詳細
/etc/ssh/moduli	セキュアなトランスポート層を構築するために非常に重要となる、Diffie-Hellman 鍵交換に使用される Diffie-Hellman グループが格納されています。SSH セッションの始めで鍵が交換される時、共有秘密値が作成されますが、どちらか一方の当事者だけでは決定できません。この値はホスト認証を行う場合に使用されます。
/etc/ssh/ssh_config	デフォルトの SSH クライアント設定ファイルです。**~/.ssh/config** が存在する場合には、これにより上書きされる点に注意して下さい。

ファイル	詳細
/etc/ssh/sshd_config	**sshd** デーモン用の設定ファイルです。
/etc/ssh/ssh_host_ecdsa_key	**sshd** デーモンで使用する ECDSA 秘密鍵です。
/etc/ssh/ssh_host_ecdsa_key.pub	**sshd** デーモンで使用する ECDSA 公開鍵です。
/etc/ssh/ssh_host_key	**sshd** デーモンにより使用される SSH プロトコルのバージョン 1 用の RSA 秘密鍵です。
/etc/ssh/ssh_host_key.pub	**sshd** デーモンにより使用される SSH プロトコルのバージョン 1 用の RSA 公開鍵です。
/etc/ssh/ssh_host_rsa_key	**sshd** デーモンにより使用される SSH プロトコルのバージョン 2 用の RSA 秘密鍵です。
/etc/ssh/ssh_host_rsa_key.pub	**sshd** デーモンにより使用される SSH プロトコルのバージョン 2 用の RSA 公開鍵です。
/etc/pam.d/sshd	**sshd** デーモン用の PAM 設定ファイルです。
/etc/sysconfig/sshd	**sshd** サービスの設定ファイルです。

表7.2 ユーザー固有の設定ファイル

ファイル	詳細
~/.ssh/authorized_keys	サーバー用の認証済み公開鍵の一覧が含まれています。クライアントがサーバーに接続する時、サーバーはこのファイル内に格納されている署名済み公開鍵を確認してクライアントを認証します。
~/.ssh/id_ecdsa	ユーザーの ECDSA 秘密鍵を格納しています。
~/.ssh/id_ecdsa.pub	ユーザーの ECDSA 公開鍵です。
~/.ssh/id_rsa	**ssh** により使用される SSH プロトコルのバージョン 2 用の RSA 秘密鍵です。
~/.ssh/id_rsa.pub	**ssh** により使用される SSH プロトコルのバージョン 2 用の RSA 公開鍵です。
~/.ssh/identity	**ssh** により使用される SSH プロトコルのバージョン 1 用の RSA 秘密鍵です。
~/.ssh/identity.pub	**ssh** により使用される SSH プロトコルのバージョン 1 用の RSA 公開鍵です。
~/.ssh/known_hosts	ユーザーがアクセスする SSH サーバーのホスト鍵が格納されています。このファイルは SSH クライアントが正しい SSH サーバーに接続していることを確認するために非常に重要です。

SSH 設定ファイルに使用可能な各種指示文についての情報は、**ssh_config**(5) および **sshd_config**(5) の man ページを参照してください。

7.2.2. OpenSSH サーバーの起動

OpenSSH サーバーを実行するには、*openssh-server* パッケージがインストールされている必要があります (Red Hat Enterprise Linux 7 に新規パッケージをインストールする方法については「パッケージのインストール」 を参照してください)。

現行のセッションで **sshd** デーモンを開始するには、シェルプロンプトで**root** として以下を入力します。

```
~]# systemctl start sshd.service
```

現行のセッションで **sshd** デーモンを停止するには、**root** として以下のコマンドを使用します。

```
~]# systemctl stop sshd.service
```

デーモンが起動時に自動的に開始するようにするには、**root** で以下を入力します。

```
~]# systemctl enable sshd.service
ln -s '/usr/lib/systemd/system/sshd.service' '/etc/systemd/system/multi-
user.target.wants/sshd.service'
```

Red Hat Enterprise Linux でシステムサービスを管理する詳細情報については、*6章systemd によるサービス管理* を参照してください。

システムを再インストールすると、新しい識別鍵のセットが作成される点に注意してください。そのため、再インストールの前にいずれかの OpenSSH ツールを使用してシステムに接続したことがあるクライアントには、以下のようなメッセージが表示されます。

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle
attack)!
It is also possible that the RSA host key has just been changed.
```

これを防ぐには、**/etc/ssh/** ディレクトリーから関連ファイルをバックアップしておくと (全一覧は表 7.1「システム全体の設定ファイル」 を参照)、システムを再インストールする時にはいつでもこれらを復元することができます。

7.2.3. リモート接続に必要な SSH

SSH を本当に有効なものにするためには、セキュリティー保護されていない接続プロトコルは使用しないことをお勧めします。さもなければ、ユーザーのパスワードは SSH を使用した 1 回のセッションでは保護されても、その後に Telnet を使用してログインした時には結局傍受されてしまうためです。無効にするサービスは、**telnet**、**rsh**、**rlogin**、**vsftpd** などがあります。

vsftpd サービスの設定方法については、「FTP」 を参照してください。Red Hat Enterprise Linux 7 でシステムサービスを管理する方法については、*6章systemd によるサービス管理* を参照してください。

7.2.4. 鍵ベース認証の使用

システムのセキュリティーをさらに強化するには、SSH 鍵のペアを生成し、パスワード認証を無効にすることで鍵ベース認証を強制します。これを行うには、**/etc/ssh/sshd_config** の設定ファイルを **vi** や **nano** などのテキストエディターで開き、**PasswordAuthentication** オプションを以下のように変更します。

```
PasswordAuthentication no
```

新規のデフォルトインストール以外のシステムで作業をしている場合は、**PubkeyAuthentication no** が設定されて **いない** ことを確認してください。リモートで接続している場合は、コンソールもしくは帯域外アクセスを使用せず、パスワード認証を無効にする前にプロセス内で鍵ベースのログをテストすることが推奨されます。

ssh、**scp** または **sftp** を使用してクライアントマシンからサーバーに接続できるようにするには、以下の手順に従って認証鍵ペアを生成します。鍵はユーザーごとに別々に生成する必要がある点に注意してください。

Red Hat Enterprise Linux 7 は、デフォルトでは SSH プロトコル 2 と RSA 鍵を使用します (詳細は「プロトコルのバージョン」を参照)。

重要

これらのステップを **root** で完了すると、鍵を使用できるのは**root** のみになります。

注記

システムを再インストールした場合に、以前に生成された鍵ペアを維持したい時は、**~/.ssh/** ディレクトリーをバックアップします。再インストール後に、このディレクトリーをホームディレクトリーにコピーします。この手順は、**root** を含むシステム上の全ユーザーが実行できます。

7.2.4.1. 鍵ペアの生成

以下の手順にしたがって、SSH プロトコルのバージョン 2 用の RSA 鍵ペアを生成します。

1. RSA 鍵ペアを生成するには、シェルプロンプトで以下を入力します。

```
~]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/john/.ssh/id_rsa):
```

2. **Enter** キーを押して、新規作成された鍵用のデフォルトの場所 (**~/.ssh/id_rsa**) を確認します。

3. パスフレーズを入力します。プロンプトが表示されたら再入力して確認します。セキュリティー上の理由により、アカウントにログインする時と同じパスワードは使用しないでください。

   この後に、以下のようなメッセージが表示されます。

```
Your identification has been saved in /home/john/.ssh/id_rsa.
Your public key has been saved in /home/john/.ssh/id_rsa.pub.
The key fingerprint is:
e7:97:c7:e2:0e:f9:0e:fc:c4:d7:cb:e5:31:11:92:14
john@penguin.example.com
The key's randomart image is:
+--[ RSA 2048]----+
|              E.  |
|             . .  |
|              o . |
|               . .|
|        S .     . |
|         + o o ..|
|          * * +oo|
|           O +..=|
|           o*  o.|
+-----------------+
```

4. **~/.ssh/** ディレクトリーのパーミッションを変更します。

```
~]$ chmod 700 ~/.ssh
```

5. 公開鍵をリモートマシンにコピーするには、以下のフォーマットでコマンドを発行します。

```
 ssh-copy-id user@hostname
```

これですべての **~/.ssh/id*.pub** 公開鍵がコピーされます。別の方法としては、公開鍵ファイル名を以下のように指定します。

```
ssh-copy-id -i ~/.ssh/id_rsa.pub user@hostname
```

これで **~/.ssh/id_rsa.pub** のコンテンツが接続先のマシン上にある **~/.ssh/authorized_keys** ファイルにコピーされます。このファイルがすでに存在する場合は、この鍵はファイルの最後に追加されます。

SSH プロトコルのバージョン 2 用の ECDSA 鍵ペアを生成するには、以下の手順にしたがいます。

1. ECDSA 鍵ペアを生成するには、シェルプロンプトで以下を入力します。

```
~]$ ssh-keygen -t ecdsa
Generating public/private ecdsa key pair.
Enter file in which to save the key (/home/john/.ssh/id_ecdsa):
```

2. **Enter** キーを押して、新規作成された鍵のデフォルトの場所 (**~/.ssh/id_ecdsa**) を確認します。

3. パスフレーズを入力します。プロンプトが表示されたら再入力して確認します。セキュリティー上の理由により、アカウントにログインする時と同じパスワードは使用しないでください。

この後に、以下のようなメッセージが表示されます。

```
Your identification has been saved in /home/john/.ssh/id_ecdsa.
Your public key has been saved in /home/john/.ssh/id_ecdsa.pub.
The key fingerprint is:
fd:1d:ca:10:52:96:21:43:7e:bd:4c:fc:5b:35:6b:63 sjw@my-server
The key's randomart image is:
+--[ECDSA  256]---+
|        .+ +o     |
|        . =.o     |
|         o o +  ..|
|          + + o  +|
|         S o o oE.|
|            + oo+.|
|             + o  |
|                  |
|                  |
+------------------+
```

4. **~/.ssh/** ディレクトリーのパーミッションを変更します。

```
~]$ chmod 700 ~/.ssh
```

5. 公開鍵をリモートマシンにコピーするには、以下のフォーマットでコマンドを発行します。

```
ssh-copy-id user@hostname
```

これですべての **~/.ssh/id*.pub** 公開鍵がコピーされます。別の方法としては、公開鍵ファイル名を以下のように指定します。

```
ssh-copy-id -i ~/.ssh/id_ecdsa.pub user@hostname
```

これで **~/.ssh/id_ecdsa.pub** のコンテンツが接続先のマシン上にある **~/.ssh/authorized_keys** ファイルにコピーされます。このファイルがすでに存在する場合は、この鍵はファイルの最後に追加されます。

システムにパスフレーズを記憶させる設定方法については 「ssh-agent の設定」 を参照してください。

重要

秘密鍵は、個人使用向けのものであり、他人には決して教えないことが重要です。

7.2.4.2. ssh-agent の設定

ssh-agent 認証エージェントを使用するとパスフレーズを保存することができるため、リモートマシンとの接続を開始する度にパスフレーズを入力する必要がなくなります。GNOME を実行している場合は、ログイン時には常にパスフレーズを求めるプロンプトを表示して、セッションを通してそのパスフレーズを記憶させておくように設定できます。それ以外の方法として、特定のシェルプロンプト用にパスフレーズを保存しておくことも可能です。

以下の手順にしたがって、GNOME セッション中にパスフレーズを保存します。

1. *openssh-askpass* パッケージがインストールされていることを確認します。インストールされていない場合には、「パッケージのインストール」 で Red Hat Enterprise Linux での新規パッケージのインストール方法について確認してください。

2. **Super** キーを押してアクティビティーの概要に入り、**Startup Applications** と入力して **Enter** を押します。**Startup Applications Preferences** ツールが表示されます。デフォルトでは、利用可能なスタートアッププログラムの一覧を含むタブが表示されます。

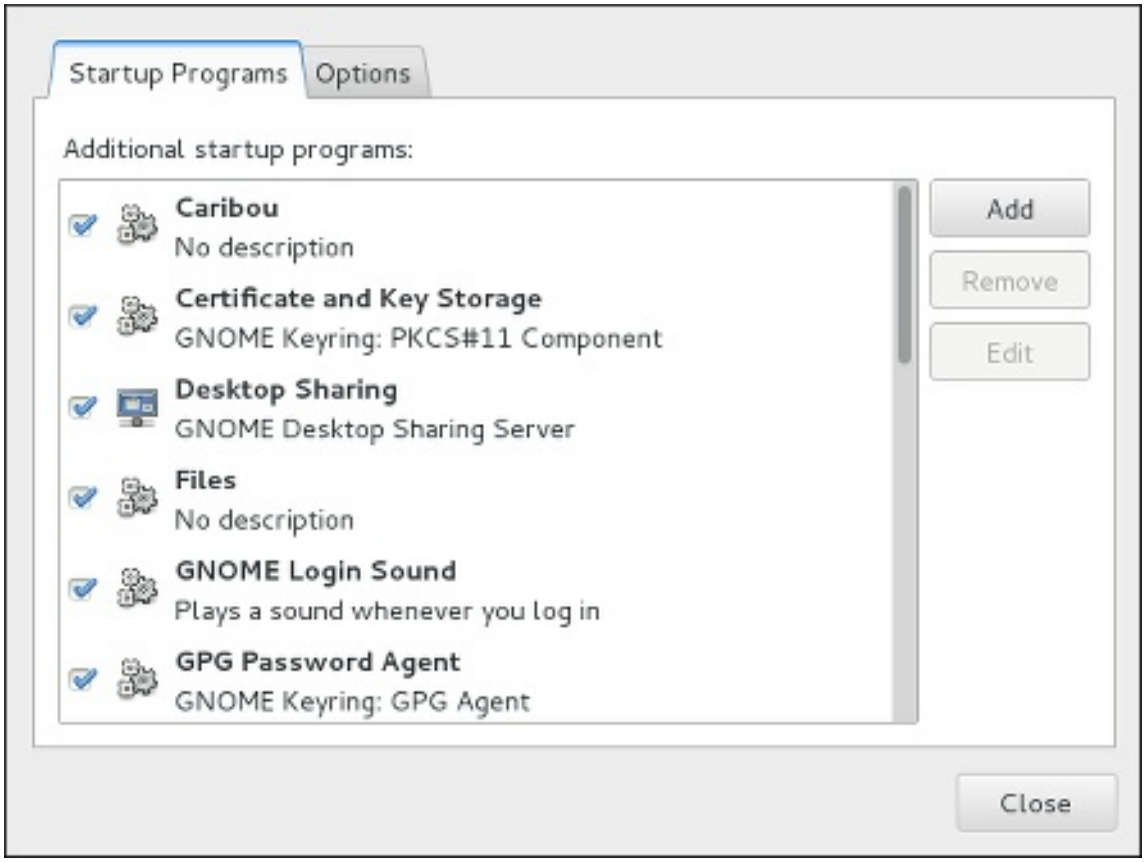


図7.1 自動起動するアプリの設定

3. 右側の **追加** ボタンをクリックして、**コマンド** フィールドに **`/usr/bin/ssh-add`** と入力します。

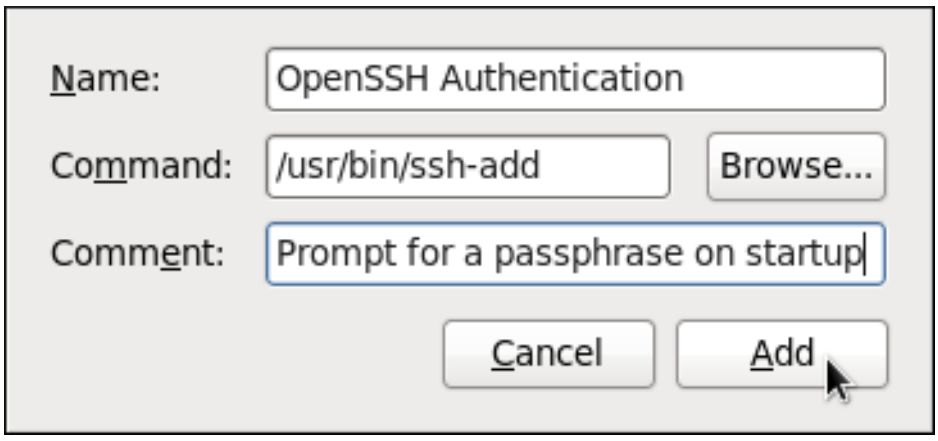


図7.2 新規アプリケーションの追加

4. **追加** をクリックした後に、新しく追加した項目の横のチェックボックスにチェックマークが付いていることを確認してください。

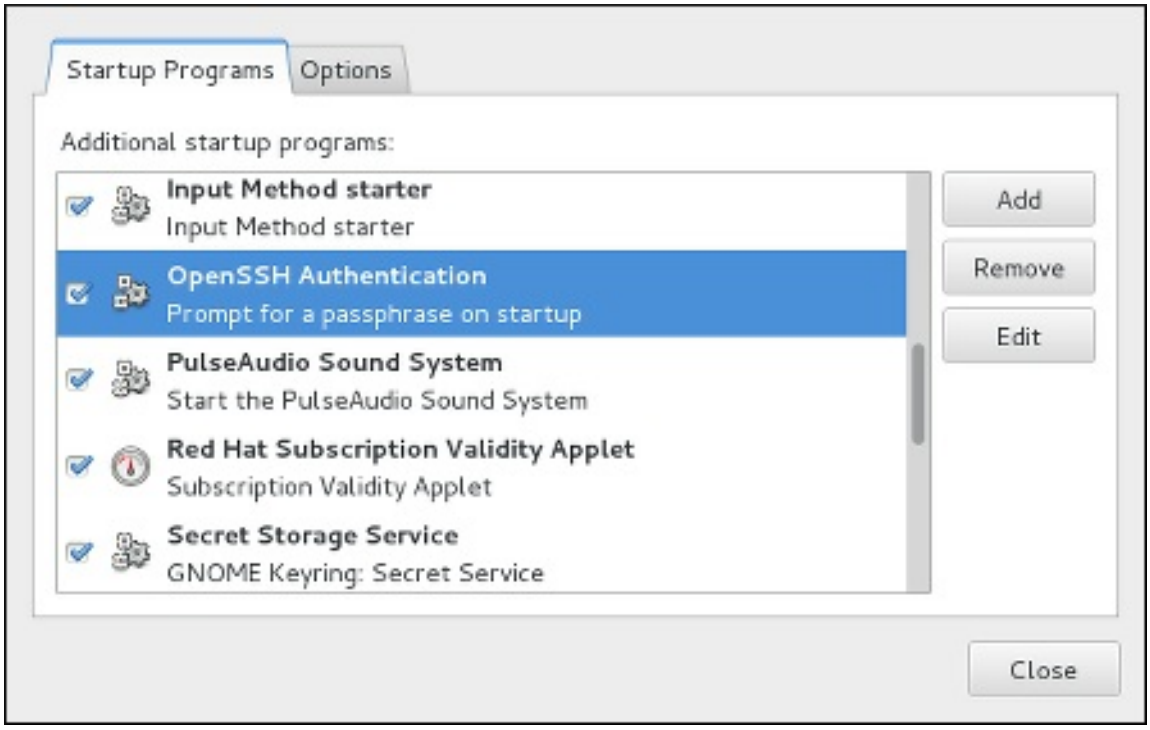


図7.3 アプリケーションの有効化

5. 一度ログアウトしてから再度ログインします。パスフレーズの入力を求めるダイアログボックスが表示されます。これ以降は、**ssh**、**scp** または **sftp** によるパスワードの入力を要求されることはありません。

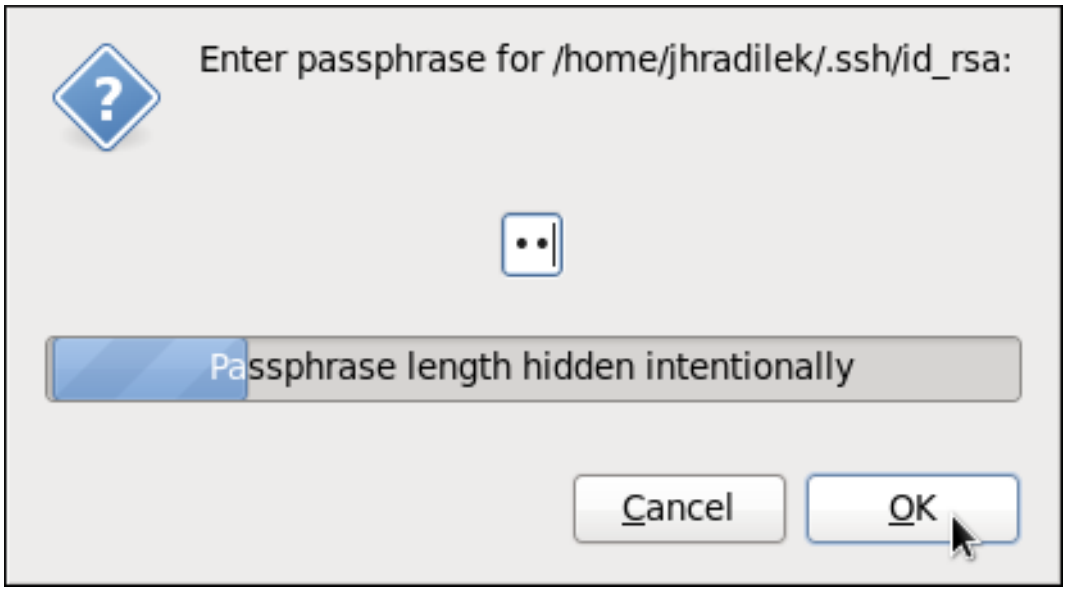


図7.4 パスフレーズの入力

特定のシェルプロンプト用のパスフレーズを保存するには、以下のコマンドを使用します:

```
~]$ ssh-add
Enter passphrase for /home/john/.ssh/id_rsa:
```

ログアウト時には、パスフレーズは記憶されない点に注意してください。仮想コンソールまたはターミナルウィンドウにログインする度にコマンドを実行する必要があります。

7.3. OpenSSH クライアント

クライアントマシンから OpenSSH サーバーに接続するには、*openssh-clients* パッケージがインストールされている必要があります (Red Hat Enterprise Linux に新規パッケージをインストールする方法については 「パッケージのインストール」を参照して下さい)。

7.3.1. ssh ユーティリティーの使用

ssh ユーティリティーを使用すると、リモートマシンにログインしてそのマシン上でコマンドを実行することできます。これは、**rlogin**、**rsh** および **telnet** プログラムに代わるセキュアな手段です。

telnet コマンドと同様に、以下のコマンドを使用してリモートマシンにログインします。

```
ssh hostname
```

例えば、**penguin.example.com** という名前のリモートマシンにログインするには、シェルプロンプトで以下を入力します。

```
~]$ ssh penguin.example.com
```

これで、ローカルマシンで使用しているユーザー名でログインします。別のユーザー名を指定したい場合には、以下の形式のコマンドを使用してください。

```
ssh username@hostname
```

例えば、**john** として **penguin.example.com** にログインするには、以下のように入力します。

```
~]$ ssh john@penguin.example.com
```

初回接続時には、以下のようなメッセージが表示されます。

```
The authenticity of host 'penguin.example.com' can't be established.
ECDSA key fingerprint is 256
da:24:43:0b:2e:c1:3f:a1:84:13:92:01:52:b4:84:ff.
Are you sure you want to continue connecting (yes/no)?
```

このダイアログの質問に答える前に、常にフィンガープリントが正しいか確認してください。ユーザーは、鍵が正しいかサーバー管理者に尋ねることができます。これは、安全で事前に合意した方法で行う必要があります。サーバーのホスト鍵にユーザーがアクセスできる場合、フィンガープリントは以下のように **ssh-keygen** コマンドを使用することで確認できます。

```
~]# ssh-keygen -l -f /etc/ssh/ssh_host_ecdsa_key.pub
256 da:24:43:0b:2e:c1:3f:a1:84:13:92:01:52:b4:84:ff   (ECDSA)
```

yes と入力して鍵を受け入れ、接続を確定します。サーバーが既知ホストの一覧に追加されたことを知らせるメッセージと、パスワードの入力を求めるプロンプトが以下のように表示されます。

```
Warning: Permanently added 'penguin.example.com' (ECDSA) to the list of
known hosts.
john@penguin.example.com's password:
```

重要

SSH サーバーのホスト鍵が変更された場合、クライアントはサーバーのホスト鍵が **~/.ssh/known_hosts** ファイルから削除されるまで接続を開始できないことをユーザーに知らせます。ただし、これを実行する前に、SSH サーバーのシステム管理者に連絡して、サーバーが被害を受けていないことを確認してください。

~/.ssh/known_hosts ファイルから鍵を削除するには、以下のようにコマンドを発行します。

```
~]# ssh-keygen -Rpenguin.example.com
# Host penguin.example.com found: line 15 type ECDSA
/home/john/.ssh/known_hosts updated.
Original contents retained as /home/john/.ssh/known_hosts.old
```

パスワードを入力すると、リモートマシン用のシェルプロンプトが表示されます。

別の方法として、シェルプロンプトにログインせずに、**ssh** プログラムを使用してリモートマシン上でコマンドを実行することができます。

```
ssh [username@]hostname command
```

例えば、**/etc/redhat-release** ファイルは Red Hat Enterprise Linux のバージョンに関する情報を提供します。**penguin.example.com** でこのファイルの内容を表示するには、以下を入力します。

```
~]$ ssh john@penguin.example.com cat /etc/redhat-release
john@penguin.example.com's password:
Red Hat Enterprise Linux Server release 7.0 (Maipo)
```

正しいパスワードを入力すると、ユーザー名が表示され、ローカルのシェルプロンプトに戻ります。

7.3.2. scp ユーティリティーの使用

scp を使用すると、暗号化されたセキュアな接続でマシン間のファイル転送を行うことができます。設計に関しては、**rcp** と非常に似ています。

ローカルファイルをリモートシステムへ転送するには、以下の形式でコマンドを使用します。

```
scp localfile username@hostname:remotefile
```

例えば、**taglist.vim** を **penguin.example.com** という名前のリモートマシンに転送したい場合は、シェルプロンプトで以下のように入力します。

```
~]$ scp taglist.vim john@penguin.example.com:.vim/plugin/taglist.vim
john@penguin.example.com's password:
taglist.vim                                   100%  144KB 144.5KB/s
00:00
```

一度に複数のファイルを指定することも可能です。**.vim/plugin/** の内容を **penguin.example.com** のリモートマシン上の同じディレクトリーに転送するには、以下のコマンドを入力します。

```
~]$ scp .vim/plugin/* john@penguin.example.com:.vim/plugin/
john@penguin.example.com's password:
closetag.vim                                  100%   13KB  12.6KB/s
00:00
snippetsEmu.vim                               100%   33KB  33.1KB/s
00:00
taglist.vim                                   100%  144KB 144.5KB/s
00:00
```

リモートファイルをローカルシステムへ転送するには、以下の構文を使用します。

```
scp username@hostname:remotefile localfile
```

たとえば **.vimrc** 設定ファイルをリモートマシンからダウンロードするには、以下のように入力します。

```
~]$ scp john@penguin.example.com:.vimrc .vimrc
john@penguin.example.com's password:
.vimrc                                        100% 2233     2.2KB/s
00:00
```

7.3.3. sftp ユーティリティーの使用

sftp ユーティリティーを使用すると、セキュアでインタラクティブな FTP セッションを開始することができます。設計に関しては、暗号化されたセキュアな接続を使用する以外は **ftp** と似ています。

リモートシステムに接続するには、以下の形式でコマンドを使用します。

```
sftp username@hostname
```

たとえば **penguin.example.com** という名前のリモートマシンに**john** というユーザー名でログインするには、以下のように入力します。

```
~]$ sftp john@penguin.example.com
john@penguin.example.com's password:
Connected to penguin.example.com.
sftp>
```

正しいパスワードを入力すると、プロンプトが表示されます。**sftp** ユーティリティーは、**ftp** で使用されるコマンドセットと同様のものを使用します (表7.3「利用可能な sftp コマンドの抜粋」を参照)。

表7.3 利用可能な sftp コマンドの抜粋

コマンド	詳細
ls [*directory*]	リモート *directory* の内容を一覧表示します。指定がない場合は、デフォルトで現在の作業ディレクトリーが使用されます。
cd *directory*	リモートの作業ディレクトリーを *directory* に変更します。
mkdir *directory*	リモートの *directory* を作成します。
rmdir *path*	リモートの *directory* を削除します。
put *localfile* [*remotefile*]	*localfile* をリモートマシンに転送します。
get *remotefile* [*localfile*]	*remotefile* をリモートマシンから転送します。

利用可能なコマンドの完全なリストは、**sftp**(1) の man ページを参照してください。

7.4. セキュアシェルの高機能

セキュアなコマンドラインインターフェースは、数多くある SSH の用途の中でも初歩的なものに過ぎません。十分な帯域幅があれば、X11 セッションは SSH チャンネル上で送信できます。あるいは、TCP/IP 転送を使用することで、以前はセキュリティー保護されていなかったシステム間のポート接続を特定の SSH チャンネルにマッピングすることができます。

7.4.1. X11 転送

SSH 接続上で X11 セッションを開始するには、以下の形式でコマンドを使用します:

```
ssh -Y username@hostname
```

たとえば **penguin.example.com** という名前のリモートマシンに**john** というユーザー名でログインするには、以下のように入力します。

```
~]$ ssh -Y john@penguin.example.com
john@penguin.example.com's password:
```

セキュアなシェルプロンプトから X プログラムが実行されると、SSH クライアントとサーバーは新しいセキュアなチャンネルを作成し、X プログラムデータはそのチャンネル上で透過的にクライアントマシンに送信されます。

X11 転送は非常に便利なものです。たとえば、X11 転送を使用すると、**プリンター設定** ユーティリティーのセキュアかつインタラクティブなセッションを作成できます。これを行うには、**ssh** を使用してサーバーに接続し、以下のコマンドを入力します。

```
~]$ system-config-printer &
```

プリンター設定ツールが表示され、リモートユーザーはリモートシステム上で印刷の設定を安全に行うことができます。

7.4.2. ポート転送

SSH は、ポート転送によりセキュリティー保護されていない **TCP/IP** プロトコルをセキュアにすることができます。この手法を使用する場合、SSH サーバーは SSH クライアントをつなぐ暗号化された経路となります。

ポート転送は、クライアント上のローカルポートをサーバー上のリモートポートにマッピングすることで機能します。SSH ではサーバーの任意のポートをクライアント上の任意のポートにマッピングすることが可能です。このテクニックが機能するためにポート番号が一致する必要はありません。

注記

1024 未満のポートで待機するようにポート転送を設定するには、**root** レベルのアクセスが必要です。

localhost 上で接続を待機する TCP/IP ポート転送チャンネルを作成するには、以下の形式でコマンドを使用します。

```
ssh -L local-port:remote-hostname:remote-port username@hostname
```

たとえば、暗号化された接続で **POP3** を使用して、**mail.example.com** と呼ばれるサーバーで電子メールを確認するには、以下のコマンドを使用します。

```
~]$ ssh -L 1100:mail.example.com:110 mail.example.com
```

ポート転送チャンネルがクライアントマシンとメールサーバー間に配置されたら、POP3 メールクライアントに対し **localhost** 上のポート **1100** を使用して、新規の電子メールを確認するように指示します。クライアントシステム上のポート **1100** に送信される要求は、安全に **mail.example.com** サーバーに向けられます。

SSH サーバーを実行しているのが **mail.example.com** ではなく、同一のネットワーク上にある別のマシンの場合でも、SSH を使用して接続の一部をセキュアにすることができます。ただし、若干異なるコマンドが必要になります。

```
~]$ ssh -L 1100:mail.example.com:110 other.example.com
```

この例では、クライアントマシン上のポート **1100** からの POP3 要求がポート **22** の SSH 接続を介して SSH サーバー **other.example.com** に転送されます。次に、**other.example.com** は **mail.example.com** 上のポート **110** に接続して、新規の電子メールを確認します。この手法を使用する場合、クライアントシステムと **other.example.com** SSH サーバー間の接続のみがセキュアである点に注意してください。

ポート転送は、ネットワークのファイアウォール経由でセキュアに情報を取得する場合にも使用できます。ファイアウォールが標準ポート (ポート 22) 経由の SSH トラフィックを許可するよう設定されているものの、他のポートへのアクセスはブロックする場合でも、確立された SSH 接続にそのような通信をリダイレクトすることにより、ブロックされたポートを使用した 2 つのホスト間の接続は可能になります。

重要

この方法でポート転送を使って接続を転送すると、クライアントシステム上のどのユーザーもそのサービスに接続できます。クライアントシステムが侵害された場合、攻撃者は転送されたサービスにアクセスすることもできます。

システム管理者がポート転送に懸念がある場合は、**/etc/ssh/sshd_config** にある **AllowTcpForwarding** の行に **No** パラメーターを指定して **sshd** サービスを再起動することにより、サーバー上でこの機能を無効にすることができます。

7.5. その他のリソース

Red Hat Enterprise Linux 上で OpenSSH を設定し、接続する詳細方法については、以下のリソースを参照してください。

インストールされているドキュメント

- **sshd**(8) — **sshd** デーモンの man ページは、利用可能なコマンドラインオプションを説明し、サポートされる設定ファイルおよびディレクトリーの完全なリストを提供します。
- **ssh**(1) — **ssh** クライアントアプリケーション の man ページは、利用可能なコマンドラインオプションとサポートされる設定ファイルおよびディレクトリーの完全なリストを提供します。
- **scp**(1) — **scp** ユーティリティーの man ページは、このユーティリティーの詳細な説明と使用方法を説明しています。

- **sftp**(1) — **sftp** ユーティリティーの man ページです。
- **ssh-keygen**(1) — **ssh-keygen** ユーティリティーの man ページは、このユーティリティーを使って **ssh** が使用する認証鍵を生成、管理、変換する詳細な方法を説明しています。
- **ssh_config**(5) — **ssh_config** の man ページでは、利用可能な SSH クライアントオプションが説明されています。
- **sshd_config**(5) — **sshd_config** のman ページは、利用可能な SSH デーモン設定オプションを詳細に説明しています。

オンラインのドキュメント

- OpenSSH Home Page — さらに詳しいドキュメントや FAQ、メーリングリストへのリンク、バグレポート、その他役立つリソースを掲載した OpenSSH のホームページです。
- OpenSSL Home Page — さらに詳しいドキュメントや FAQ、メーリングリストへのリンク、その他役立つリソースを掲載した OpenSSL のホームページです。

関連項目

- 4章*権限の取得* では、**su** および **sudo** コマンドを使って管理者権限を取得する方法が説明されています。
- 6章*systemd によるサービス管理*では systemd に関する詳細情報と、**systemctl** コマンドを使ってシステムサービスを管理する方法が説明されています。

---

[2] 多重接続は、共有されている共通の媒体上で送信される信号で構成されます。SSH により、異なるチャンネルが共通のセキュアな接続で送信されます。

第8章 TigerVNC

TigerVNC (Tiger Virtual Network Computing) は、グラフィカルデスクトップシェアリングのシステムで、他のコンピューターのリモート制御を可能にします。

TigerVNC は、クライアント-サーバーのネットワーク上で機能します。**サーバー** はその出力 (**vncserver**) 共有し、**クライアント** (**vncviewer**) はサーバーに接続します。

注記

以前の Red Hat Enterprise Linux ディストリビューションとは異なり、Red Hat Enterprise Linux 7 の **TigerVNC** は、その設定に **systemd** システム管理デーモンを使います。**/etc/sysconfig/vncserver** 設定ファイルは、**/lib/systemd/system/vncserver@.service** に代わりました。

8.1. VNC サーバー

vncserver は、VNC (Virtual Network Computing) デスクトップを開始するユーティリティーです。適切なオプションと Xvnc を実行し、VNC デスクトップ上でウィンドウマネージャーを起動します。**vncserver** を使うと、どこからでもいくつものクライアントがアクセス可能なマシン上で完全に並行なセッションを実行できます。

8.1.1. VNC サーバーのインストール

TigerVNC サーバーをインストールするには、以下のコマンドを **root** で実行します。

```
# yum install tigervnc-server
```

TigerVNC をクライアントとしても使用したい場合は、以下のコマンドを実行します。これでクライアントとサーバーコンポーネントもインストールされます。

```
# yum install vnc
```

8.1.2. VNC サーバーの設定

手順8.1 VNC の初回接続の設定

1. **/etc/systemd/system/vncserver@.service** という名前の設定ファイルが必要になります。このファイルを作成するには、**/lib/systemd/system/vncserver@.service** ファイルを **root** でコピーします。

   ```
   # cp /lib/systemd/system/vncserver@.service
   /etc/systemd/system/vncserver@.service
   ```

2. **/etc/systemd/system/vncserver@.service** を修正して、*USER* を実際のユーザー名で置き換えます。その他の行は、そのままにしておきます。**-geometry** 引数は、作成される VNC デスクトップのサイズを指定します。デフォルトでは、1024x768 に設定されます。

```
ExecStart=/sbin/runuser -l USER -c "/usr/bin/vncserver %i -geometry
1280x1024"
PIDFile=/home/USER/.vnc/%H%i.pid
```

3. 変更を保存します。
4. 変更を直ちに反映させるには、以下のコマンドを実行します。

```
# systemctl daemon-reload
```

5. 設定ファイルでユーザー用のパスワードを設定します。まず、**root** から **user** に切り替わる必要があることに注意してください。

```
# su - user
$ vncpasswd
Password:
Verify:
```

重要

保存されたパスワードは、安全に暗号化されていません。パスワードファイルにアクセスがあれば、誰でもプレーンテキストでパスワードを見つけることができます。

8.1.2.1. ユーザー 2 人用に VNC サーバーを設定する

同一マシン上で複数のユーザーを設定したい場合は、ユーザーごとに異なるテンプレートタイプのサービスファイルを作成します。

例8.1 ユーザー 2 人の用の VNC サーバー

たとえば、**vncserver-USER_1@.service** と **vncserver-USER_2@.service** という 2 つのサービスファイルを作成します。どちらのファイルでも、*USER* を正しい ユーザー名に置き換えます。

両方のユーザーでパスワードを設定します。

```
$ su - USER_1
$ vncpasswd
Password:
Verify:
$ su - USER_2
$ vncpasswd
Password:
Verify:
```

8.1.3. VNC サーバーの起動

サービスを起動もしくは有効にするには、コマンドで直接ディスプレイ番号を指定します。上記で設定したファイルはテンプレートとして機能し、 そこでは **%i** は **systemd** がディスプレイ番号に置き換えます。有効な番号を用いて、以下のコマンドを実行します。

```
# systemctl start vncserver@:display_number.service
```

また、システム起動時に自動的にサービスが開始するようにすることもできます。そうすると、ログイン時に **vncserver** が自動的に開始されます。**root** で以下のコマンドを発行します。

```
# systemctl enable vncserver@:display_number.service
```

この時点では、他のユーザーは **vncviewer** プログラムで定義済みのディスプレイ番号およびパスワードを使って、VNC サーバーに接続することができます。

8.1.3.1. ユーザー 2 人および 2 つの別個のディスプレイ用に VNC サーバーを設定する

vncserver-USER_1@.service および vncserver-USER_2@.service という 2 つの設定済み VNC サーバーで、異なるディスプレイ番号を有効にすることができます。たとえば、以下のコマンドでは、USER_1 の VNC サーバーがディスプレイ 3 で起動し、USER_2 の VNC サーバーがディスプレイ 5 で起動することになります。

```
# systemctl start vncserver-USER_1@:3.service
# systemctl start vncserver-USER_2@:5.service
```

8.1.4. VNC セッションの終了

vncserver サービスの有効化と同様に、システム開始時に自動的にサービスの起動を無効にすることができます。

```
# systemctl disable vncserver@:display_number.service
```

または、システムの実行中に、以下のコマンドを **root** で発行すると、サービスを停止することができます。

```
# systemctl stop vncserver@:display_number.service
```

8.2. VNC ビューワ

vncviewer は、共有グラフィカルユーザーインターフェイスを表示し、サーバーを制御するプログラムです。

vncviewer の操作では、エントリーを含むポップアップメニューがあり、これでフルスクリーンモードの切り替えやビューワの終了などのさまざまなアクションを実行します。別の方法では、ターミナルから **vncviewer** を操作することもできます。**vncviewer** と使用可能な多数のパラメーターがあり、コマンドラインで **vncviewer -h** を入力するとこのリストが表示されます。

8.2.1. VNC ビューワのインストール

TigerVNC ビューワをインストールするには、**root** で以下のコマンドを実行します。

```
# yum install tigervnc
```

8.2.2. VNCサーバーへの接続

VNC サーバーを設定したら、どの VNC からでも接続ができます。これを行うには、以下の形式で **vncviewer** を実行します。

```
vncviewer IP address:port_number
```

例8.2 vncserver にクライアントを接続する

IP アドレスが **192.168.0.4** でディスプレイ番号が **3** の場合、コマンドは以下のようになります。

```
$ vncviewer 192.168.0.4:3
```

8.2.2.1. ファイアウォール

暗号化されていない接続を使用する場合は、**firewalld** が接続を妨害する可能性があります。この障害を回避するには、以下のようにポートを開きます。

注記

VNC サーバーのデフォルトのポートは、5900 です。リモートデスクトップにアクセス可能なポートに到達するには、このデフォルトのポートとユーザーに割り振られたディスプレイ番号の合計を計算する必要があります。たとえば、2 つめのポートは次のようになります。2 + 5900 = 5902

1. **firewalld** 設定についての情報を確認するには、以下のコマンドを実行します。

```
$ firewall-cmd --list-all
```

2. 選択したポートが開いているかどうかを確認するには、以下のコマンドを実行します。

```
$ firewall-cmd --query-port=port_number/tcp
```

8.2.3. SSH を使用した VNC サーバーへの接続

VNC は、クリアなテキストネットワークプロトコルで、通信への攻撃に対抗するセキュリティーはありません。この通信を安全なものにするには、SSH トンネルを使って VNC サーバーとクライアントの通信を暗号化します。

-via オプションを使うことでサーバーとクライアントの接続を暗号化することができます。

1. 最初に実行するコマンドは、以下のような形式になります。

```
$ vncviewer -via user@host :display_number
```

例8.3 -via オプションの使用

```
$ vncviewer -via USER_2@192.168.2.101 :3
```

2. プロンプトが表示されたら、パスワードを入力し、**Enter** を押して確認します。

3. リモートデスクトップのウィンドウが画面に表示されます。

暗号化された接続のみを使用したい場合は、systemd.service ファイルの ExecStart 行で **-localhost** オプションを使うと、暗号化されていない接続をすべて防ぐことができます。

```
ExecStart=/sbin/runuser -l user -c "/usr/bin/vncserver -localhost %i"
```

これで **vncserver** は localhost および **-via** からのポート転送接続以外の接続は受け付けなくなります。

OpenSSH についての詳細情報は、*7章OpenSSH* を参照してください。

8.3. その他のリソース

vncserver(1)

VNC サーバーの man ページです。

vncviewer(1)

VNC ビューワの man ページです。

vncpasswd(1)

VNC パスワードの man ページです。

パート IV. サーバー関係

このセクションでは、ウェブサーバーのセットアップ、あるいはファイルおよびディレクトリーをネットワーク上で共有する方法などのサーバーに関連した各種トピックを取り扱います。

第9章 Web サーバー

HTTP (ハイパーテキスト転送プロトコル) サーバー、または*Web サーバー*は、Web 上でクライアントにコンテンツを提供するネットワークサービスです。通常これは Web ページを意味しますが、他のドキュメントも同様に提供されます。

9.1. Apache HTTP サーバー

Red Hat Enterprise Linux 7 で利用可能な Web サーバーは、Apache HTTP サーバーデーモンである **httpd** です。これは、Apache Software Foundation が開発するオープンソースの Web サーバーです。Red Hat Enterprise Linux 7 では、Apache サーバーは **Apache HTTP Server 2.4** に更新されています。本セクションでは、**httpd** サービスの基本的な設定を説明すると共に、サーバーモジュールの追加、仮想ホストのセットアップ、およびセキュアな HTTP サーバーの設定を取り扱っています。

Apache HTTP サーバー 2.4 とバージョン 2.2 との間には重要な相違が存在します。以前の Red Hat Enterprise Linux のリリースからアップグレードする際には、**httpd** サービスの設定も同様に更新する必要があります。本セクションでは、いくつかの新機能の追加のレビュー、重要な変更の概説、古い設定ファイルの更新を説明します。

9.1.1. 注目すべき変更点

Apache HTTP サーバーバージョン 2.4 では、以下のような変更がなされています。

httpd サービスの制御

SysV init スクリプトからの移行に伴い、サーバー管理者は **service** コマンドから **apachectl** および **systemctl** のコマンドを使用したサービスの制御に切り替えることが推奨されます。以下の例は、 **httpd** サービス固有のものです。

```
service httpd graceful
```

上記のコマンドは、以下のものに置き換えられます

```
apachectl graceful
```

同様に

```
service httpd configtest
```

上記のコマンドも以下のものに置き換えられます

```
apachectl configtest
```

httpd 用の **systemd** ユニットファイルは、以下に記すように init スクリプトとは異なる動作をします。

- サービスがリロードされると、デフォルトで正常な再起動が使用されます。
- サービスが停止されると、デフォルトで正常な停止が使用されます。

Private /tmp

システムのセキュリティーを高めるため、**systemd** ユニットファイルは **httpd** デーモンを実行してプライベートの **/tmp** ディレクトリーを使用します。これは、システムの**/tmp** ディレクトリーとは別のものです。

設定レイアウト

モジュールを読み込む設定ファイルは、**/etc/httpd/conf.modules.d** ディレクトリー内にあります。(*php* パッケージのような) **httpd** 用の追加の読み込み可能なモジュールを提供するパッケージは、ファイルをこのディレクトリーに格納します。**conf.modules.d** ディレクトリー内の設定ファイルはすべて、**httpd.conf** の本文の前に処理されます。**/etc/httpd/conf.d** ディレクトリー内の設定ファイルは、**httpd.conf** の本文の後で処理されるようになりました。

httpd パッケージ自体は、追加の設定ファイルをいくつか提供しています。

- `/etc/httpd/conf.d/autoindex.conf`

  これは、mod_autoindex ディレクトリーのインデックス化を設定します。

- `/etc/httpd/conf.d/userdir.conf`

  これは、**http://example.com/~username/** などのユーザーディレクトリーへのアクセスを設定します。このようなアクセスは、デフォルトではセキュリティーのために無効になっています。

- `/etc/httpd/conf.d/welcome.conf`

  以前のリリースと同様に、これはコンテンツがない場合に **http://localhost/** で表示される「ようこそ」ページを設定します。

デフォルト設定

最小限のデフォルトの **httpd.conf** がデフォルトで提供されるようになりました。**Timeout** や **KeepAlive** といった一般的な設定は、デフォルトでは明示的に設定されることはなくなりました。代わりに、デフォルト設定はハードコーディングされています。このハードコーディングされた全設定指示文用のデフォルト設定は、マニュアルで指定されています。詳細は、「インストールされているドキュメント」 を参照してください。

設定変更

後方互換性がない多くの変更が **httpd** 設定構文になされました。これらは、**httpd 2.2** から **httpd 2.4** に既存設定を移行する場合に必要になります。更新に関する詳細情報は、http://httpd.apache.org/docs/2.4/upgrading.html にある Apache ドキュメントを参照してください。

プロセスモデル

Red Hat Enterprise Linux の以前のリリースでは、異なる *マルチプロセスモデル*(MPM) が異なる **httpd** バイナリーとして利用可能となっていました。つまり、分岐モデルの「prefork」 を **/usr/sbin/httpd** として、またスレッドベースのモデルである「worker」 を **/usr/sbin/httpd.worker** としてしました。

Red Hat Enterprise Linux 7 では、単独の **httpd** バイナリーのみが使われ、3 つの MPM は読み込み可能なモジュール、worker、prefork (デフォルト)、および event として利用可能になっています。設定ファイル **/etc/httpd/conf.modules.d/00-mpm.conf** では、これら 3 つの MPM モジュールのいずれかを読み込む選択ができます。

パッケージ変更

LDAP 認証および権限の各モジュールは、個別のサブパッケージ *mod_ldap* で提供されています。新たなモジュール **mod_session** と関連のヘルパーモジュールは、新サブパッケージ *mod_session* で提供されています。**mod_proxy_html** と **mod_xml2enc** の新モジュールは、新サブパッケージ *mod_proxy_html* で提供されています。

ファイルシステムのレイアウトのパッケージ

/var/cache/mod_proxy ディレクトリーは提供されなくなりました。代わりに、**/var/cache/httpd/** ディレクトリーが **proxy** および **ssl** サブディレクトリーとパッケージ化されています。

httpd と提供されていたパッケージ化されたコンテンツは、**/var/www/** から **/usr/share/httpd/** に移動しています。

- `/usr/share/httpd/icons/`

  /var/www/icons/ は、**/usr/share/httpd/icons** に移動しました。このディレクトリーには、ディレクトリーのインデックスで使用されるアイコンセットが含まれています。これらはデフォルト設定では **http://localhost/icons/** にあり、アイコンの場所および可用性は **/etc/httpd/conf.d/autoindex.conf** で設定可能です。

- `/usr/share/httpd/manual/`

  /var/www/manual/ は **/usr/share/httpd/manual/** に移動しました。このディレクトリーは *httpd-manual* パッケージに含まれており、**httpd** の HTML バージョンのマニュアルが格納されています。これはパッケージがインストール済みであれば **http://localhost/manual/** にあり、マニュアルの場所と可用性は **/etc/httpd/conf.d/manual.conf** で設定可能です。

- `/usr/share/httpd/error/`

  /var/www/error/ は **/usr/share/httpd/error/** に移動しました。カスタムの複数言語の HTTP エラーページです。デフォルトでは設定されておらず、設定ファイルの例は **/usr/share/doc/httpd-*VERSION*/httpd-multilang-errordoc.conf** で提供されています。

認証、認可、およびアクセス制御

認証や認可、さらにはアクセス制御に使用される設定指示文は、大幅に変更されました。**Order**、**Deny** および **Allow** の各指示文を使用している既存の設定ファイルは、新たな **Require** 構文を使うようにしてください。詳細は、http://httpd.apache.org/docs/2.4/howto/auth.html の Apache ドキュメントを参照してください。

suexec

システムのセキュリティーを改善するため、**suexec** バイナリーは **setuid root** にインストールされなくなりました。代わりに、ファイルシステム機能ビットのセットがあり、これにより限定的なパーミッションセットが可能になります。この変更と共に、**suexec** バイナリーは **/var/log/httpd/suexec.log** ログファイルを使うことがなくなりました。代わりに、ログメッセージは **syslog** に送信されます。デフォルトでは、これらのメッセージは **/var/log/secure** ログファイルに表示されます。

モジュールインターフェイス

httpd モジュールインターフェイスへの変更のため、**httpd 2.4** は **httpd 2.2** 対して構築されたサードパーティーのバイナリーモジュールと互換性がありません。このようなモジュールは、必要に応じて **httpd 2.4** モジュールインターフェイス用に調整し、再構築する必要があります。バージョン **2.4** における詳細な API 変更の一覧は、http://httpd.apache.org/docs/2.4/developer/new_api_2_4.html で確認できます。

ソースからのモジュール構築に使用される **apxs** バイナリーは、**/usr/sbin/apxs** から **/usr/bin/apxs** に移動しました。

削除されたモジュール

Red Hat Enterprise Linux 7 で削除された **httpd** モジュールは以下の通りです。

mod_auth_mysql、**mod_auth_pgsql**

httpd 2.4 は、**mod_authn_dbd** モジュールで SQL データベース認証サポートを内部で提供します。

mod_perl

mod_perl はアップストリームでは、**httpd 2.4** で公式にサポートされていません。

mod_authz_ldap

httpd 2.4 は、**mod_authnz_ldap** を使って LDAP サポートを内部で提供します。

9.1.2. 設定の更新

Apache HTTP サーバーバージョン 2.2 の設定ファイルを更新するには、以下の手順にしたがいます。

1. モジュールは変更されている可能性があるため、すべてのモジュール名が正しいことを確認してください。名前変更されたモジュールそれぞれについて **LoadModule** 指示文を調節します。
2. サードパーティーのモジュールは読み込みを試行する前にすべて再コンパイルをします。これは一般的に認証と権限付与のモジュールに該当します。
3. **mod_userdir** モジュールを使用する場合は、ディレクトリー名 (通常 **public_html**) を示す **UserDir** 指示文が備わっていることを確認します。
4. Apache HTTP セキュアサーバーを使用する場合は、**/etc/httpd/conf.d/ssl.conf** を編集して Secure Sockets Layer (SSL) プロトコルを有効にします。

以下のコマンドを使用すると、設定エラーの可能性をチェックすることができます。

```
~]# apachectl configtest
Syntax OK
```

Apache HTTP サーバーの設定をバージョン 2.2 から 2.4 に更新する方法の詳細については、http://httpd.apache.org/docs/2.4/upgrading.html を参照してください。

9.1.3. httpd サービスの実行

このセクションでは、Apache HTTP サーバーの起動、停止、再起動、および現在のステータスのチェック方法を説明しています。**httpd** サービスを有効にするには、まず *httpd* がインストールされていることを確認します。以下のコマンドを使用します。

```
~]# yum install httpd
```

ターゲットの概念と Red Hat Enterprise Linux 内のシステムサービスの管理方法全般についての詳細は、6章*systemd によるサービス管理*を参照してください。

9.1.3.1. サービスの起動

httpd サービスを実行するには、シェルプロンプトで以下を入力します。

```
~]# systemctl start httpd.service
```

起動時にサービスを自動的にスタートしたい場合には、以下のコマンドを入力します。

```
~]# systemctl enable httpd.service
ln -s '/usr/lib/systemd/system/httpd.service' '/etc/systemd/system/multi-
user.target.wants/httpd.service'
```

注記

Apache HTTP サーバーをセキュアサーバーとして実行している場合、暗号化したプライベート SSL キーを使用すると、マシンが起動した後にパスワードを要求される可能性があります。

9.1.3.2. サービスの停止

実行中の **httpd** サービスを停止するには、シェルプロンプトで以下を入力します:

```
~]# systemctl stop httpd.service
```

ブート時にサービスが自動的に起動しないようにする場合は、以下を入力します。

```
~]# systemctl disable httpd.service
rm '/etc/systemd/system/multi-user.target.wants/httpd.service'
```

9.1.3.3. サービスの再起動

実行中の **httpd** サービスを再起動する方法は、3 通りあります。

1. サービスを全面的に再起動するには、以下を入力します。

   ```
   ~]# systemctl restart httpd.service
   ```

   これで実行中の **httpd** サービスが停止し、直ちに再起動します。このコマンドは、PHP のような動的に読み込まれたモジュールをインストールもしくは削除した後に使ってください。

2. 設定を再読み込みするだけの場合は、以下を入力します。

   ```
   ~]# systemctl reload httpd.service
   ```

   これで実行中の **httpd** サービスが設定ファイルを再読み込みします。現在プロセス中の要求はいずれも割り込みされるので、クライアントのブラウザーがエラーメッセージを表示したり、ページの一部のみが表示される可能性があります。

3. アクティブな要求に影響を及ぼさずに設定ファイルを再読み込みするには、以下を入力します。

```
~]# service httpd graceful
```

これにより、実行中の **httpd** サービスは設定ファイルを再読み込みします。現在プロセス中の要求は、いずれも古い設定を使用することになります。

Red Hat Enterprise Linux 7 でシステムサービスを管理する詳細情報については、*6章systemd によるサービス管理* を参照してください。

9.1.3.4. サービスステータスの確認

httpd サービスが実行中であることを確認するには、シェルプロンプトに以下を入力します。

```
~]# systemctl is-active httpd.service
active
```

9.1.4. 設定ファイルの編集

httpd サービスは開始後にデフォルトで、表9.1「httpd サービスの設定ファイル」 に一覧表示してある場所から設定を読み込みます。

表9.1 httpd サービスの設定ファイル

パス	詳細
/etc/httpd/conf/httpd.conf	主要設定ファイル
/etc/httpd/conf.d/	主要設定ファイル内に含まれている設定ファイル用の補助ディレクトリー

デフォルト設定はほとんどの状況に適合しますが、重要な設定オプションをいくつか知っておくと役に立ちます。変更はいかなるものでも、効果が反映されるには最初にサーバーを再起動する必要があることに注意してください。**httpd** サービスの再起動方法の詳細については「サービスの再起動」 を参照してください。

設定でエラーの可能性をチェックするには、シェルプロンプトで以下を入力します。

```
~]# apachectl configtest
Syntax OK
```

失敗からの復旧をより簡単にするために、編集する前にオリジナルファイルのコピーを作成しておくことをお勧めします。

9.1.5. モジュールを使った作業

モジュラーアプリケーションである **httpd** サービスは、必要に応じてランタイムに動的にロード、またはアンロードできる多数の *Dynamic Shared Objects (動的共有オブジェクト)* (DSO) と一緒に配布されています。デフォルトではこれらのモジュールは 32-bit システムでは、**/usr/lib/httpd/modules/** 内に、64-bit システムでは、**/usr/lib64/httpd/modules/** 内に配置されています。

9.1.5.1. モジュールのロード

特定の DSO モジュールをロードするには、**LoadModule** 指示文を使用します。個別のパッケージで提供されるモジュールでは多くの場合、設定ファイルが **/etc/httpd/conf.d/** ディレクトリーにあることに注意してください。

例9.1 mod_ssl DSO のロード

```
LoadModule ssl_module modules/mod_ssl.so
```

操作が終了したら、Web サーバーを再起動して設定を再読み込みします。**httpd** サービスの再起動の方法については「サービスの再起動」を参照してください。

9.1.5.2. モジュールの書き込み

新規の DSO モジュールを作成する予定がある場合は、最初に *httpd-devel* パッケージがインストールされていることを確認してください。インストールするには、シェルプロンプトで以下を入力します。

```
~]# yum install httpd-devel
```

このパッケージには、モジュールをコンパイルするために必要なファイル、ヘッダーファイル、および **APache eXtenSion** (**apxs**) ユーティリティーが含まれています。

書き込みが終了したら、以下のコマンドでモジュールをビルドできます。

```
~]# apxs -i -a -c module_name.c
```

ビルドが成功すると、Apache HTTP サーバーで配布されている他のモジュールと同様にこのモジュールをロードできるはずです。

9.1.6. 仮想ホストのセットアップ

Apache HTTP サーバーに組み込まれている仮想ホストにより、サーバーは要求される IP アドレス、ホスト名、またはポートに基づいて異なる情報を提供できます。

名前ベースの仮想ホストを作成するには、まず設定ファイルの例である **/usr/share/doc/httpd-*VERSION*/httpd-vhosts.conf** を **/etc/httpd/conf.d/** ディレクトリーにコピーし、**@@Port@@** と **@@ServerRoot@@** のプレースホルダーの値を置き換えます。例9.2「仮想ホスト設定のサンプル」にあるように、実際の要件に合わせてオプションをカスタマイズします。

例9.2 仮想ホスト設定のサンプル

```
<VirtualHost *:80>
    ServerAdmin webmaster@penguin.example.com
    DocumentRoot "/www/docs/penguin.example.com"
    ServerName penguin.example.com
    ServerAlias www.penguin.example.com
    ErrorLog "/var/log/httpd/dummy-host.example.com-error_log"
    CustomLog "/var/log/httpd/dummy-host.example.com-access_log" common
</VirtualHost>
```

ServerName はマシンに割り当てられている有効な DNS 名である必要があることに注意してください。**<VirtualHost>** コンテナは高度にカスタマイズ可能で、メインサーバー設定内で利用できるほとんどの指示文を受け付けます。このコンテナ内でサポート**されていない**指示文には **User** と **Group** があり、これらは **SuexecUserGroup** に置き換えられています。

注記

仮想ホストがデフォルト以外のポートでリッスンするように設定する場合は、**/etc/httpd/conf/httpd.conf** ファイルのグローバル設定で **Listen** 指示文を適切に更新することを忘れないでください。

新規に作成された仮想ホストをアクティベートするには、Web サーバーを再起動する必要があります。**httpd** サービスを再起動する方法については、「サービスの再起動」を参照してください。

9.1.7. SSL サーバーのセットアップ

Secure Sockets Layer (SSL) (セキュアソケットレイヤー) は暗号化のプロトコルであり、安全なサーバーとクライアントの通信を可能にします。その拡張および改善バージョンである *Transport Layer Security* (TLS) (トランスポートレイヤーセキュリティ) と共に、SSL はプライバシーとデータの整合性を保証します。Apache HTTP サーバーは、SSL/TLS サポートを提供するために OpenSSL ツールキットを使用するモジュールである **mod_ssl** と一緒に組み合わせて、一般的に *SSL サーバー*と呼ばれます。

傍受できる人は誰でも読み取りと修正が可能な普通の HTTP 接続とは異なり、**mod_ssl** を使用すると、発信されたコンテンツの検査や修正を阻止できます。このセクションでは、Apache HTTP サーバー設定内でこのモジュールを有効にする方法についての基本情報を提供し、秘密鍵と自己署名の証明書の生成プロセスを説明していきます。

9.1.7.1. 証明書とセキュリティーの概要

安全な通信は鍵の使用に基づいています。従来の *symmetric cryptography (対称型暗号化)* では、トランザクションの両方でお互いの通信の解読に使用可能な同一の鍵を備えています。それに対し、パブリックの *asymmetric cryptography (非対称型暗号化)* では、2 つの鍵が共存します。*秘密鍵* は秘密として守られて、*公開鍵* は通常、公共と共有されます。公開鍵でエンコードされたデータは秘密鍵でのみデコードされるのに対し、秘密鍵でエンコードされたデータは逆に公開鍵でのみデコードできます。

SSL を使用して安全な通信を提供するためには、SSL サーバーは *Certificate Authority* (CA) (認証局) で署名された電子証明書を使用する必要があります。証明書はサーバーの様々な属性 (サーバーのホスト名、企業の名前、その住所など) と CA の秘密鍵で生成した署名を一覧表示します。この証明書は特定の認証局が証明書を発行したこと、また証明書がいかなる方法でも修正されていないことを保証するものです。

Web ブラウザーは新規の SSL 接続を確立する際に、Web サーバーから提供される証明書をチェックします。証明書に信頼できる CA からの署名がない場合や、証明書に表記してあるホスト名が接続確立で使用されたホスト名と一致しない場合は、Web ブラウザーはそのサーバーとの交信を拒否して、通常はユーザーに適切なエラーメッセージを表示します。

デフォルトでは、ほとんどのウェブブラウザーが広く使用されている認証局のグループを信用するように設定されています。そのため、セキュアサーバーをセットアップする時には適切な CA (認証局) を選択すべきです。そうすることで、対象ユーザーは接続を信頼できますが、そうでない場合はエラーメッセージが表示されて、手動で証明書を受理する必要があります。証明書のエラーメッセージを無視するようにユーザーに推奨することは攻撃者の侵略を許すことにつながるので、できる限り信頼できる CA を使用してください。この件に関する詳細情報は、表9.2「最も一般的なウェブブラウザー用の CA 一覧」を参照してください。

表9.2 最も一般的なウェブブラウザー用の CA 一覧

ウェブブラウザー	リンク
Mozilla Firefox	Mozilla root CA 一覧
Opera	Opera が使用する Root 証明書
Internet Explorer	Windows ルート証明書プログラムのメンバー

SSL サーバーをセットアップする際には、証明書要求と秘密鍵を生成してから、証明書要求と企業の識別証明と支払いを認証局に送ります。認証局は証明書要求と識別を確認すると、サーバーで使用できる署名付きの証明書を送ってきます。その他の方法として、CA 署名のない自己署名の証明書を作成することができますが、これはテスト目的のみに使用すべきものです。

9.1.7.2. mod_ssl モジュールを有効にする

SSL サーバーをセットアップする予定の場合は、*mod_ssl* (**mod_ssl** モジュール) と *openssl* (OpenSSL ツールキット) のパッケージがインストールされていることを確認してください。インストールするには、シェルプロンプトで以下を入力します。

```
~]# yum install mod_ssl openssl
```

これにより、中心となる Apache HTTP サーバー設定ファイルにデフォルトで含まれている **/etc/httpd/conf.d/ssl.conf** に **mod_ssl** 設定ファイルが作成されます。このモジュールを読み込むには、「サービスの再起動」にあるように **httpd** サービスを再起動します。

9.1.7.3. 既存の鍵と証明書の使用

以前に鍵と証明書が作成されている場合は、新規で作成するのではなく、既存のファイルを使用するように SSL サーバーを設定することができます。ただし、これが実行できないケースが 2 つだけあります。

1. **IP アドレスまたはドメイン名を変更しているケース。**

   証明書は特定の IP アドレスとドメイン名のペアに発行されるものです。これらの値のどちらかが変更されると証明書は無効になります。

2. **VeriSign からの証明書があって、サーバーソフトウェアを変更しているケース。**

   広く使用されている認証局である VeriSign は、特定のソフトウェア製品、IP アドレス、およびドメイン名に証明書を発行します。ソフトウェア製品を変更すると証明書が無効になります。

上記のいずれのケースでも、新しい証明書を取得する必要があります。このトピックに関する情報については、「新しい鍵と証明書の生成」を参照してください。

既存の鍵と証明書を使用する場合は、その関連ファイルを **/etc/pki/tls/private/** ディレクトリーと **/etc/pki/tls/certs/** ディレクトリーにそれぞれ移動します。これを行うには以下のコマンドを入力します。

```
~]# mv key_file.key /etc/pki/tls/private/hostname.key
~]# mv certificate.crt /etc/pki/tls/certs/hostname.crt
```

そして、以下の行を **/etc/httpd/conf.d/ssl.conf** 設定ファイルに追加します。

```
SSLCertificateFile /etc/pki/tls/certs/hostname.crt
SSLCertificateKeyFile /etc/pki/tls/private/hostname.key
```

更新済みの設定を読み込むには、「サービスの再起動」にあるように **httpd** サービスを再起動します。

例9.3 Red Hat セキュアウェブサーバーからの鍵と証明書の使用

```
~]# mv /etc/httpd/conf/httpsd.key
/etc/pki/tls/private/penguin.example.com.key
~]# mv /etc/httpd/conf/httpsd.crt
/etc/pki/tls/certs/penguin.example.com.crt
```

9.1.7.4. 新しい鍵と証明書の生成

新しい鍵と証明書のペアを生成するには、システムに *crypto-utils* パッケージをインストールしておく必要があります。これをインストールするには、シェルプロンプトで以下を入力します。

```
~]# yum install crypto-utils
```

このパッケージは、SSL 証明書と秘密鍵を生成して管理するツールセットを提供すると共に、鍵生成のプロセスで役立つ Red Hat Keypair Generation ユーティリティーである **genkey** を収納しています。

重要

有効な証明書を既に所有していて、それを新規のものと置き換える予定の場合は、異なるシリアル番号を指定します。これにより、クライアントのブラウザーがこの変更の通知を受けて予定どおりに新規の証明書に更新して、このページへのアクセスに失敗しないようにします。カスタムのシリアル番号で新規の証明書を作成するには、**genkey** の代わりに、以下のコマンドを使用します。

```
~]# openssl req -x509 -new -set_serial number -key hostname.key -
out hostname.crt
```

注記

使用中のシステムに特定のホスト名用の鍵ファイルが既に存在する場合は、**genkey** は開始を拒否します。その場合は、以下のコマンドを使用して既存のファイルを削除します。

```
~]# rm /etc/pki/tls/private/hostname.key
```

ユーティリティーを実行するには、**genkey** コマンドの後に該当するホスト名 (たとえば、**penguin.example.com**) を付けて使用します。

```
~]# genkey hostname
```

鍵と証明書の生成を完了するには、以下の手順にしたがいます。

1. 鍵と証明書を保存する場所を確認します。

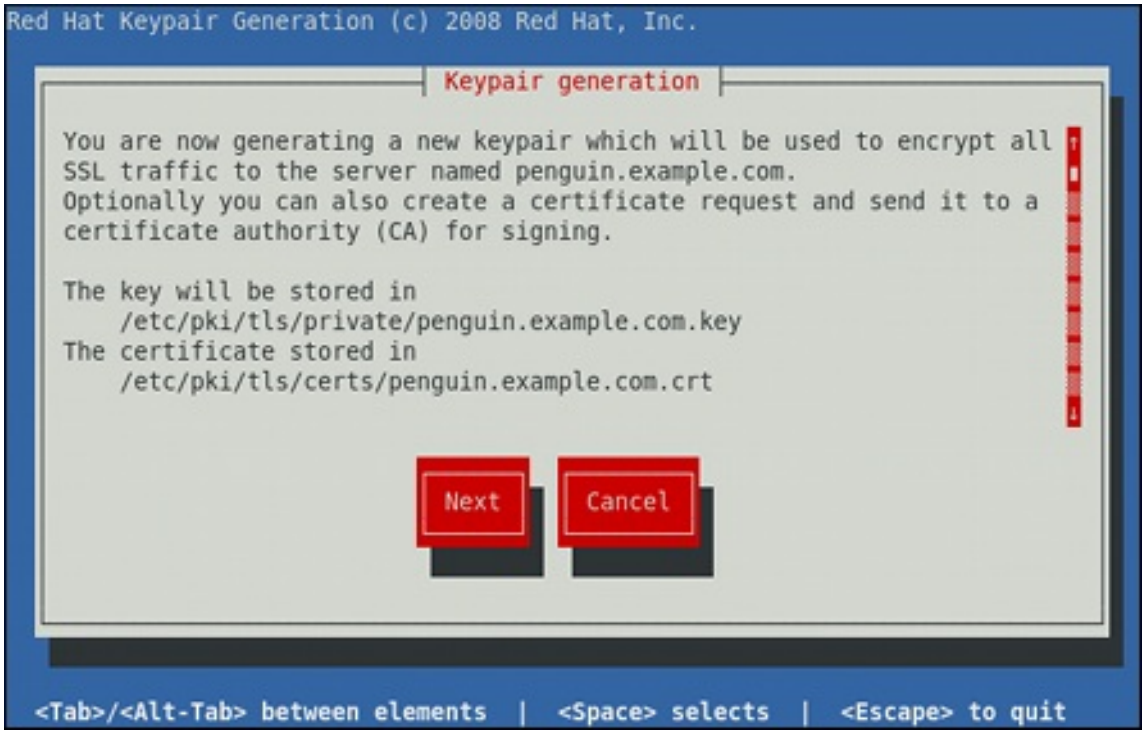


図9.1 genkeyユーティリティーの実行

Tab キーを使用して **Next (次へ)** ボタンを選択してから、**Enter** キーを押すと次の画面を進みます。

2. **Up** と **down** の矢印キーを使用して、適切な鍵のサイズを選択します。大きな鍵はセキュリティーを向上させますが、サーバーの応答時間も長くなることに注意してください。NIST では、**2048 bits** の使用を推奨しています。『NIST Special Publication 800-131A』 を参照してください。

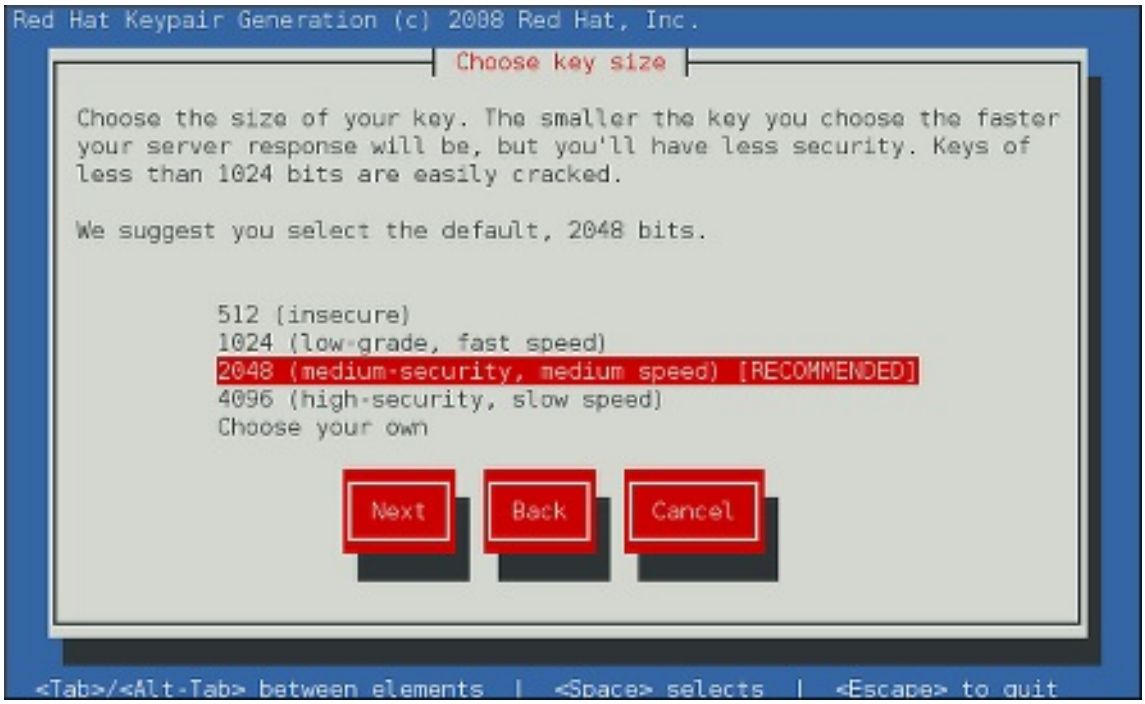


図9.2 鍵のサイズ選択

終了したら、**Tab** キーを使用して **Next （次へ）** ボタンを選択して **Enter** キーを押すと、ランダムなビットの生成プロセスが開始します。選択した鍵のサイズによっては、これに時間がかかりることもあります。

3. 証明書要求を認証局に送信するかどうかを決定します。

図9.3 証明書要求の生成

Tab キーを使用して **Yes （はい）** を選択し、証明書要求を組み立てるか、または**No （いいえ）** を選択して、自己署名の証明書を生成します。その後に、**Enter** を押して選択を確定します。

4. **Spacebar （スペースバー）** キーを使用すると、秘密鍵の暗号化を有効にする (**[*]**) か、無効にする (**[ ]**) 選択ができます。

図9.4 秘密鍵の暗号化

Tab キーを使用して **Next (次へ)** ボタンを選択してから、**Enter** キーを押すと次の画面を進みます。

5. 秘密鍵の暗号化を有効にしている場合は、適切なパスフレーズを入力します。セキュリティーの理由により入力時には文字が表示されませんが、最低でも 5 文字の長さが必要です。

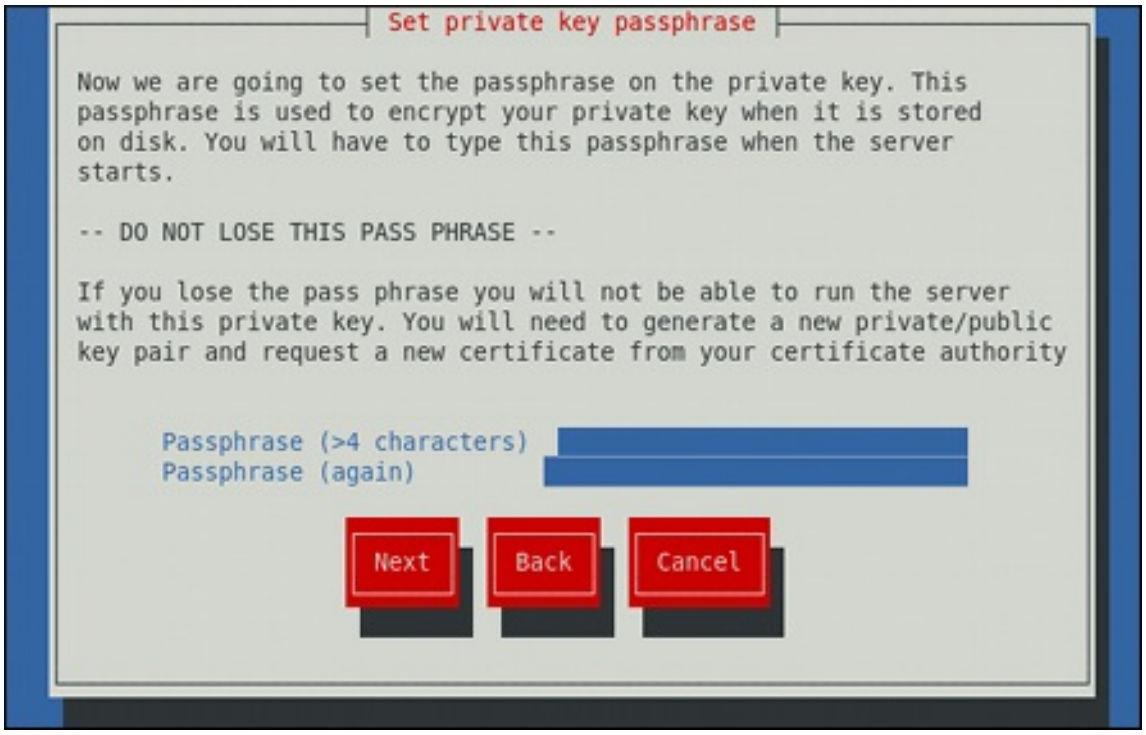


図9.5 パスフレーズの入力

Tab キーを使用して **Next (次へ)** ボタンを選択してから、**Enter** キーを押すと次の画面を進みます。

重要

サーバーを開始するには正しいパスフレーズの入力が必要です。それを紛失したり忘れたりした場合は、新しい鍵と証明書を生成しなければなりません。

6. 証明書詳細のカスタマイズ

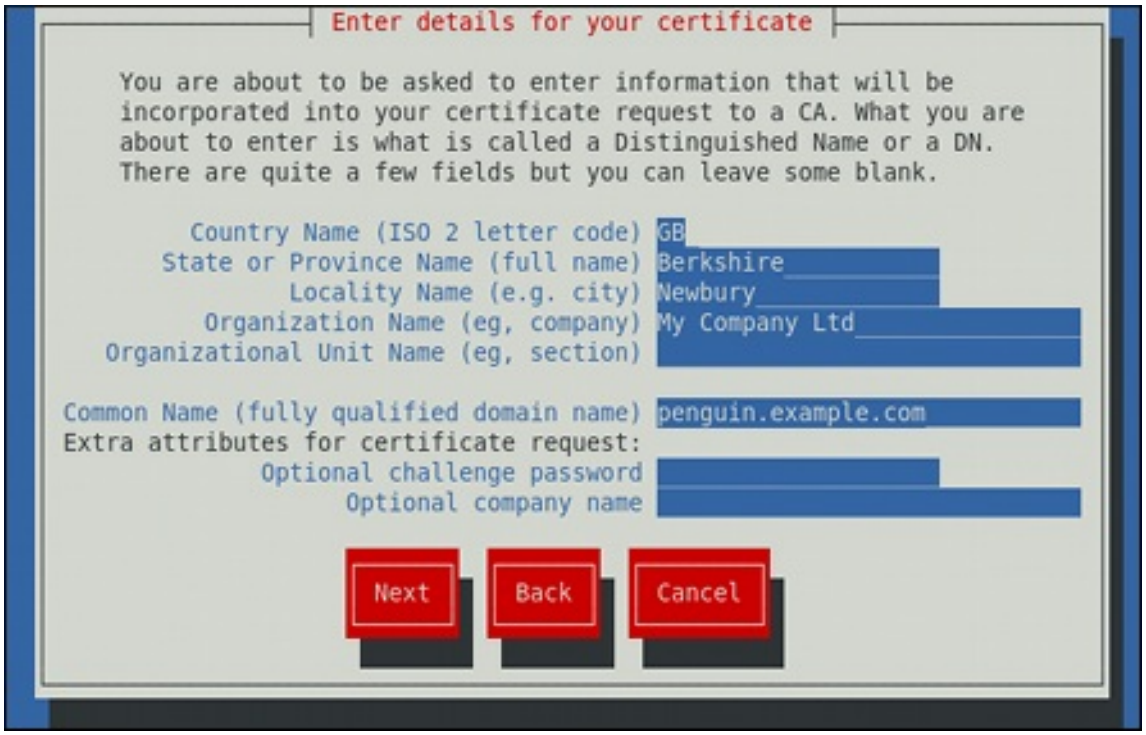


図9.6 証明書情報の指定

Tab キーを使用して **Next** ボタンを選択します。それから**Enter** を押すと鍵の生成が完了します。

7. 以前に証明書要求の生成を有効にしていた場合は、それを認証局に送信するようにプロンプトされます。

図9.7 証明書要求を送信する方法の指示

Enter を押してシェルプロンプトに戻ります。

生成が終了したら、鍵と証明書の場所を **/etc/httpd/conf.d/ssl.conf** 設定ファイルに追加します。

```
SSLCertificateFile /etc/pki/tls/certs/hostname.crt
SSLCertificateKeyFile /etc/pki/tls/private/hostname.key
```

最後に、「サービスの再起動」にあるように**httpd** サービスを再起動します。すると更新された設定が読み込まれます。

9.1.8. その他のリソース

Apache HTTP サーバーに関する詳細については、以下のリソースを参照してください。

9.1.8.1. インストールされているドキュメント

http://localhost/manual/

指示文と利用可能なモジュールの全説明を含む Apache HTTP サーバーの公式ドキュメントです。このドキュメントにアクセスするには、*httpd-manual* パッケージがインストールしてあり、ウェブサーバーが稼働している必要があります。

/usr/share/doc/httpd-*VERSION*/

設定ファイルの例が多く含まれているディレクトリーです。

man httpd

コマンドラインオプションの全一覧が収納されている **httpd** サービス用の man ページです。

man apachectl

Apache HTTP サーバー制御インターフェイスの man ページです。

man genkey

genkey の man ページにはその使用法についての全ドキュメントが含まれています。

9.1.8.2. 役立つ Web サイト

http://httpd.apache.org/

すべての指示文とデフォルトのモジュールに関するドキュメントが収納された、Apache HTTP サーバーの公式ウェブサイトです。

http://www.modssl.org/

mod_ssl モジュールに関する公式の Web サイトです。

http://www.openssl.org/

さらに詳しいドキュメントや FAQ、メーリングリストへのリンク、その他役立つリソースを掲載した OpenSSL のホームページです。

第10章 メールサーバー

Red Hat Enterprise Linux は、電子メールを使用、アクセスするための多くの高度なアプリケーションを提供します。本章では、現在使用されている最新の電子メールプロトコルと電子メールを送受信するプログラムについて説明します。

10.1. 電子メールプロトコル

今日、電子メールはクライアント/サーバーのアーキテクチャーを使用して配信されています。電子メールのメッセージは、メールクライアントプログラムを使用して作成されます。次に、このプログラムがメッセージをサーバーに送信します。メッセージはサーバーが受信者の電子メールサーバーに転送し、そこで受信者の電子メールクライアントに渡されます。

このプロセスを有効にするために、各種の標準のネットワークプロトコルが異なるマシンによる (多くの場合、異なるオペレーティングシステムで、異なる電子メールプログラムを使用) 電子メールの送受信を可能にしています。

以下は、電子メールの転送に最も一般的に使用されているプロトコルです。

10.1.1. メール転送プロトコル

クライアントアプリケーションからサーバーへのメール配信、および送信元サーバーから転送先サーバーへのメール配信は、*SMTP* (*簡易メール転送プロトコル*) により処理されます。

10.1.1.1. SMTP

SMTP の第一の目的は、メールサーバー間における電子メールの転送ですが、これは電子メールクライアントにとっても極めて重要となります。電子メールを送信するには、クライアントはメッセージを送信メールサーバーに送信します。送信メールサーバーはこれを受けて、転送先メールサーバーに配信を連絡します。このため、電子メールクライアントを設定する時には SMTP サーバーを指定する必要があります。

Red Hat Enterprise Linux では、ユーザーはメール配信を処理するようローカルマシン上の SMTP サーバーを設定することができます。ただし、送信メール用にリモート SMTP サーバーを設定することも可能です。

SMTP プロトコルに関して重要なのは認証が不要である点です。これにより、インターネット上の誰でも、個人や大規模なグループに対してでも電子メールを送信することができます。迷惑メールや *スパム* が可能になるのは SMTP のこうした特性が原因です。リレー制限を課すと、インターネット上の任意のユーザーがご使用の SMTP サーバーを介してインターネット上の別のサーバーへ電子メールを送信することが制限されます。リレー制限を課さないサーバーは、*オープンリレー*サーバーと呼ばれます。

Red Hat Enterprise Linux 7 は、Postfix および Sendmail SMTP プログラムを提供しています。

10.1.2. メールアクセスプロトコル

メールサーバーから電子メールを取得するために電子メールクライアントアプリケーションが使用する主要なプロトコルには、*POP (ポストオフィスプロトコル)* と *IMAP (インターネットメッセージアクセスプロトコル)* の 2 つがあります。

10.1.2.1. POP

Red Hat Enterprise Linux のデフォルトの POP サーバーは **Dovecot** で、*dovecot* パッケージにより提供されています。

注記

Dovecot を使用するには、まず使用中のシステムに**dovecot** パッケージがインストールされていることを確認するために、**root** で以下を実行します。

```
~]# yum install dovecot
```

yum を使ったパッケージのインストールについては 「パッケージのインストール」を参照してください。

POP サーバーを使用する場合、電子メールメッセージは電子メールクライアントのアプリケーションがダウンロードします。デフォルトでは、ほとんどの **POP** 電子メールクライアントでは、電子メールサーバーのメッセージが正しく転送されるとそのメッセージは削除されるように自動的に設定されています。ただし、この設定は通常は変更できます。

POP は、電子メールのファイル添付を可能にする *MIME (多目的インターネットメール拡張)* などの重要なインターネットメッセージング標準と完全な互換性があります。

POP は、電子メールを読むためのシステムが 1 つであるユーザーの場合に最適に機能します。また、インターネットやメールサーバーを持つネットワークに常時接続していないユーザーにもうまく機能します。ネットワーク速度が遅いユーザーの場合には不利になりますが、**POP** はクライアントプログラムに対して、認証を行った上で各メッセージのコンテンツ全体をダウンロードするよう要求します。このプロセスは、メッセージに大きなファイルが添付されている場合は長時間かかる場合があります。

標準 **POP** プロトコルの最新版は**POP3** です。

ただし、あまり使用されていない **POP** プロトコルのバリアントにも様々な種類があります。

- **APOP** — **MD5** 認証を使用した **POP3** です。暗号化されていないパスワードを送信するのではなく、エンコードされたユーザーパスワードのハッシュが電子メールクライアントからサーバーへ送信されます。
- **KPOP** — Kerberos 認証を使用した **POP3** です。
- **RPOP** — **RPOP** 認証を使用した **POP3** です。これは、パスワードに似たユーザーごとの ID を使用し、POP 要求を認証します。ただしこの ID は暗号化されていないため、**RPOP** のセキュリティーレベルは標準 **POP** と同程度です。

セキュリティーを強化するには、*SSL (Secure Socket Layer セキュアソケットレイヤ)* 暗号化をクライアント認証とデータ転送セッションに使用することができます。これは、**ipop3s** サービスまたは**stunnel** アプリケーションを使用して有効にすることができます。電子メール通信をセキュアにする方法の詳細については 「通信のセキュリティー保護」を参照してください。

10.1.2.2. IMAP

Red Hat Enterprise Linux でのデフォルト **IMAP** サーバーは **Dovecot** で、*dovecot* パッケージが提供しています。**Dovecot** のインストール方法については「POP」を参照してください。

IMAP メールサーバーを使用する場合、電子メールメッセージはサーバーに残るためユーザーは読み取り、削除を行うことができます。**IMAP** により、クライアントアプリケーションがサーバー上でメールディレクトリーを作成、名前変更、削除を行い電子メールを整理、保存することもできます。

IMAP は複数のマシンを使って電子メールにアクセスするユーザーに特に役立ちます。このプロトコルでは、メッセージが開封されるまでは、電子メールのヘッダー情報しかダウンロードされず帯域幅を節減できるため、低速な接続でメールサーバーに接続するユーザーにも便利です。ユーザーは、メッセージを表示またはダウンロードすることなく削除することも可能です。

便宜上、**IMAP** クライアントアプリケーションはメッセージのコピーをローカルでキャッシュすることが可能です。そのため、ユーザーは **IMAP** サーバーに直接接続していない時でも、既読メッセージを閲覧することができます。

IMAP は **POP** と同様に、電子メールのファイル添付を可能にする MIME などの重要なインターネットメッセージング標準と完全に互換性があります。

セキュリティーを強化するには、**SSL** 暗号化をクライアント認証とデータ転送セッションに使用することができます。これは、**imaps** サービスまたは **stunnel** プログラムを使用して有効にすることができます。電子メール通信をセキュアにする方法の詳細については 「通信のセキュリティー保護」 を参照してください。

無償や商用の IMAP クライアントおよびサーバーは他にも提供されています。これらの多くは、IMAP プロトコルを拡張し、追加機能を提供します。

10.1.2.3. Dovecot

IMAP および **POP3** プロトコルを実装する **imap-login** と **pop3-login** プロセスは、*dovecot* パッケージに含まれているマスター **dovecot** デーモンが生成します。**IMAP** および **POP** の使用は、**/etc/dovecot/dovecot.conf** 設定ファイルで設定します。デフォルトでは **dovecot** は、**SSL** を使用して **IMAP** と **POP3** をセキュアなバージョンとともに実行します。**POP** を使用するよう **dovecot** を設定するには、次のステップを実行します。

1. **protocols** 変数がコメント解除されていて (行頭のハッシュ記号 (#) を削除)、**pop3** 引数を含むよう **/etc/dovecot/dovecot.conf** 設定ファイルを編集します。例えば以下のとおりです。

   ```
   protocols = imap imaps pop3 pop3s
   ```

   protocols 変数がコメントアウトされている場合は、**dovecot** は上記のデフォルト値を使用します。

2. この変更を現行のセッションで有効にするには、以下のコマンドを実行します。

   ```
   ~]# systemctl restart dovecot
   ```

3. この変更を次回の再起動後に有効にするには、以下のコマンドを実行します。

   ```
   ~]# systemctl enable dovecot
   ln -s '/usr/lib/systemd/system/dovecot' '/etc/systemd/system/multi-
   user.target.wants/dovecot'
   ```

注記

dovecot が報告するのは **IMAP** サーバーを起動したことだけですが、**POP3** サーバーも起動する点に注意してください。

SMTP とは違い、**IMAP** と **POP3** はユーザー名とパスワードを使用して接続するクライアントを認証する必要があります。デフォルトでは、両方のプロトコルのパスワードは、暗号化されていないネットワーク上で渡されます。

dovecot で **SSL** を設定するには、以下を実行します。

- **/etc/pki/dovecot/dovecot-openssl.cnf** 設定ファイルを必要に応じて編集します。ただし、標準的なインストールではこのファイルへの変更は必要ありません。
- **/etc/pki/dovecot/certs/dovecot.pem** および **/etc/pki/dovecot/private/dovecot.pem** ファイルの名前変更、移動、削除を行います。
- **/usr/libexec/dovecot/mkcert.sh** のスクリプトを実行して、**dovecot** の自己署名証明書を作成します。証明書は **/etc/pki/dovecot/certs** および **/etc/pki/dovecot/private** ディレクトリーにコピーされます。変更を有効にするには、**dovecot** を再起動します。

```
~]# systemctl restart dovecot
```

dovecot の詳細は http://www.dovecot.org でオンラインで参照できます。

10.2. 電子メールプログラムの分類

一般的に、すべての電子メールアプリケーションは 3 つのタイプのうち 1 つ以上に分類されます。それぞれの分類は、電子メールメッセージの移動および管理のプロセスにおいてそれぞれ特定の役割を果たします。大半のユーザーはメッセージの送受信に使用する特定の電子メールプログラムだけを認識しますが、電子メールを正しい送信先に届くにはすべての電子メールプログラムが重要になります。

10.2.1. メール転送エージェント (Mail Transport Agent)

MTA (メール転送エージェント) は、**SMTP** を使用してホスト間で電子メールメッセージを転送します。メッセージは目的の送信先に移動する時、様々な MTA に関わることがあります。

マシン間のメッセージ配信は簡単に見えるかもしれませんが、配信のためにある MTA がメッセージを受け入れることが可能か、または受け入れるべきかを判断する過程全体は非常に複雑です。その上、スパムの問題により、特定の MTA の使用は通常 MTA の設定または MTA が常駐するネットワークのアクセス設定によって制限されます。

最新の電子メールクライアントプログラムの多くは、電子メールを送信する際に MTA として機能することができます。ただし、このアクションが真の MTA の役割であると混同してはいけません。電子メールクライアントプログラムで MTA のような電子メール送信が可能である唯一の理由は、アプリケーションを稼働しているホストに独自の MTA がないためです。これは非 UNIX ベースのオペレーティングシステム上の電子メールクライアントプログラムに特に当てはまります。ただし、これらのクライアントプログラムは、使用を許可されている MTA に対して送信メッセージを送信するだけで、目的の受信者の電子メールサーバーには直接メッセージを配信することはありません。

Red Hat Enterprise Linux は *Postfix* と *Sendmail* という 2 つの MTA を提供しているので、電子メールクライアントプログラムは大抵、MTA として機能する必要がありません。Red Hat Enterprise Linux には、*Fetchmail* と呼ばれる特別用途の MTA も装備されています。

Postfix、Sendmail、Fetchmail の詳細は 「メール転送エージェント (MTA)」 を参照してください。

10.2.2. メール配信エージェント (MDA)

MDA (メール配信エージェント) は MTA により呼び出され、適切なユーザーのメールボックスに受信メールをファイル保存します。多くの場合、MDA は実際には **mail** や Procmail などの *LDA (ローカル配信エージェント)* です。

電子メールクライアントアプリケーションが読み取り可能なポイントに配信されるメッセージを実際に処理するプログラムは、いずれも MDA と見なすことができます。このため、一部の MTA (Sendmail、Postfix など) は、ローカルユーザーのメールスプールファイルに新規の電子メールメッセージを追加する時に、

MDA の役割を果たすことができます。通常、MDA はシステム間でのメッセージの転送やユーザーインターフェースの提供は行いません。MDA は、ローカルマシン上でメッセージの配信と並べ替えを行い、電子メールクライアントアプリケーションがアクセスできるようにします。

10.2.3. メールユーザーエージェント (Mail User Agent)

MUA (メールユーザーエージェント) は電子メールクライアントアプリケーションと同義語です。MUA プログラムにより、ユーザーは最低でも電子メールメッセージの読み取りと作成ができます。多くの MUA では、**POP** または **IMAP** プロトコルを介したメッセージ取得や、メッセージを保管するメールボックスの設定、MTA へのメッセージ送信ができます。

MUA は、**Evolution** のようなグラフィカルインターフェースの場合と、**Mutt** のようなシンプルなテキストベースのインターフェイスの場合があります。

10.3. メール転送エージェント (MTA)

Red Hat Enterprise Linux 7 には、Postfix と Sendmail の 2 つの主要 MTA が装備されています。Postfix がデフォルトの MTA として設定されており、Sendmail は非推奨となっています。デフォルトの MTA を Sendmail に変更する必要がある場合は、Postfix をアンインストールするか、次のコマンドを使って Sendmail に切り替えます。

```
~]# alternatives --config mta
```

以下のコマンドを使って、希望のサービスを有効にすることもできます。

```
~]# systemctl enable <service>
```

同様に、サービスを無効にするには、シェルプロンプトで以下を入力します。

```
~]# systemctl disable <service>
```

Red Hat Enterprise Linux 7 でシステムサービスを管理する詳細情報については、*6章systemd によるサービス管理* を参照してください。

10.3.1. Postfix

当初、IBM のセキュリティエキスパート/プログラマーの Wietse Venema 氏によって開発された Postfix は、Sendmail 互換の MTA で、セキュア、高速、かつ容易に設定できるように設計されています。

セキュリティー向上のために、Postfix ではモジュラー型設計を採用しており、権限が限定された小さなプロセスは、マスターデーモンが起動します。より小さく、権限の低いプロセスは、メール配信の様々な段階に関連する非常に特殊なタスクを実行してルートディレクトリが変更された環境で稼働し、攻撃の影響を制限します。

Postfix がローカルコンピュータ以外のホストからのネットワーク接続を受け入れるよう設定するには、設定ファイルを多少変更するだけでできます。さらに、より複雑なニーズのために、Postfix は様々な設定オプションだけでなくサードパーティのアドオンも提供するため、多用途でフル機能の MTA となっています。

Postfix の設定ファイルはヒューマンリーダブルで、250 以上の指示文に対応しています。Sendmail とは異なり、変更を反映するためにマクロ処理は必要なく、また最も一般的に使用されるオプションの大部分は、多数のコメントが付いたファイルで説明されています。

10.3.1.1. Postfix のデフォルトインストール

Postfix 実行可能ファイルは **postfix** です。このデーモンは、メール配信の処理に必要なすべての関連プロセスを起動します。

Postfix は設定ファイルを **/etc/postfix/** ディレクトリーに格納します。以下は、一般的に使用されるその他のファイルの一覧です。

- **access** — アクセス制御に使用します。このファイルは、Postfix に接続可能なホストを指定します。
- **main.cf** — Postfix のグローバル設定ファイルです。設定オプションの大部分がこのファイルで指定されています。
- **master.cf** — メール配信を完了するために Postfix が様々なプロセスとやりとりを行う方法を指定します。
- **transport** — 電子メールアドレスをリレーホストにマッピングします。

aliases ファイルは **/etc/** ディレクトリーにあります。このファイルは Postfix と Sendmail 間で共有されます。ユーザー ID エイリアスを記述するメールプロトコルが必要な設定可能な一覧です。

重要

デフォルトの **/etc/postfix/main.cf** ファイルでは、Postfix はローカルコンピューター以外のホストからのネットワーク接続を受け付けられないように設定されています。Postfix を他のクライアント用のサーバーとして設定する方法は 「Postfix の基本設定」を参照してください。

/etc/postfix ディレクトリーにある設定ファイルのオプションに変更を加えた後には、変更を反映させるために **postfix** サービスを再起動してください。

```
~]# systemctl restart postfix
```

10.3.1.2. Postfix の基本設定

デフォルトでは、Postfix はローカルホスト以外のホストからのネットワーク接続を受け付けません。ネットワーク上の他のホストを対象としたメール配信を有効にするには、**root** で以下のステップを実行します。

- **vi** などのテキストエディタで **/etc/postfix/main.cf** ファイルを編集します。
- **mydomain** 行のハッシュ記号 (#) を削除してコメント解除してから、*domain.tld* の箇所を **example.com** などのメールサーバーがサービスを提供しているドメインに置き換えます。
- **myorigin = $mydomain** 行のコメントを解除します。
- **myhostname** 行のコメントを解除し、*host.domain.tld* をマシンのホスト名に置き換えます。
- **mydestination = $myhostname, localhost.$mydomain** 行のコメントを解除します。
- **mynetworks** 行のコメントを解除して、*168.100.189.0/28* の箇所を、サーバーに接続可能なホスト用の有効なネットワーク設定に置き換えます。
- **inet_interfaces = all** 行のコメントを解除します。
- **inet_interfaces = localhost** をコメント化します。
- **postfix** サービスを再起動します。

これらの手順が完了したら、ホストは配信のため外部の電子メールを受け入れるようになります。

Postfix には様々な設定オプションがあります。Postfix の設定方法を学習する最適な方法の 1 つは、**/etc/postfix/main.cf** 設定ファイルのコメントを読むことです。Postfix 設定、SpamAssassin 統合、**/etc/postfix/main.cf** パラメータの詳細などの補足情報は http://www.postfix.org/ でオンラインで参照できます。

10.3.1.3. LDAP での Postfix の使用

Postfix は **LDAP** ディレクトリーを様々なルックアップテーブルのソースとして利用できます (たとえば **aliases**、**virtual**、**canonical** など)。これにより **LDAP** は階層的なユーザー情報を保存でき、Postfix は **LDAP** クエリの結果を必要な場合にのみ知らされます。この情報をローカルに保存しないことで、管理者は容易に管理を行うことができます。

10.3.1.3.1. /etc/aliases ルックアップのサンプル

以下は **/etc/aliases** ファイルをルックアップする **LDAP** を使用する基本的な例です。**/etc/postfix/main.cf** ファイルに以下の内容が含まれていることを確認してください。

```
alias_maps = hash:/etc/aliases, ldap:/etc/postfix/ldap-aliases.cf
```

/etc/postfix/ldap-aliases.cf ファイルがない場合は、これを作成します。以下の内容を含めます。

```
server_host = ldap.example.com
search_base = dc=example, dc=com
```

ldap.example.com、***example***、***com*** パラメーターは、既存の利用可能な **LDAP** サーバーの仕様と置き換える必要があります。

注記

/etc/postfix/ldap-aliases.cf ファイルは、**LDAP SSL** と **STARTTLS** を有効にするパラメーターなどの様々なパラメーターを指定することができます。詳細は **ldap_table(5)** の man ページを参照してください。

LDAP の詳細は 「OpenLDAP」 を参照してください。

10.3.2. Sendmail

Sendmail の主な目的は、他の MTA と同様に、通常 **SMTP** プロトコルを使用して、ホスト間で電子メールを安全に転送することです。Sendmail は非推奨とみなされており、可能な場合は Postfix の使用が推奨されていることに注意してください。詳細は、「Postfix」 を参照してください。

10.3.2.1. 用途と制約

認識すべき重要な点は、Sendmail ができないことではなく、Sendmail が何であるか、何ができるのかということです。複数の役割を果たすモノリシックなアプリケーションの時代には、Sendmail は組織内で電子メールサーバーを稼働するために必要な唯一のアプリケーションに思われるかもしれません。技術的にはこれは真実です。Sendmail はメールを各ユーザーのディレクトリーにスプールして、ユーザーに送信メールを配信できるからです。しかし、実際は大半のユーザーは単なるメール配信以上の機能を必要とします。ユーザーは通常、**POP** または **IMAP** を使用する MUA で電子メールとやりとりを行い、ローカルマシンにメッセージをダウンロードする方法を望みます。あるいは、メールボックスへのアクセスに Web インターフェースを好むユーザーもいます。こうした他のアプリケーションを Sendmail と連動させることは可能ですが、実際、それらが存在する理由は異なり、独立して機能することができます。

Sendmail で設定すべき、また設定できるすべての用途の説明は、本セクションの対象外となります。Sendmail には文字通り数百におよぶ様々なオプションやルールセットがあるため、Sendmail のあらゆる機能や問題修正方法に関する専門的な資料が多くあります。Sendmail に関するリソースの一覧は「その他のリソース」を参照してください。

本セクションでは、Sendmail と共にデフォルトでインストール済みのファイルを概説し、迷惑メール (スパム) の停止方法や *LDAP* での Sendmail の拡張方法など基本設定の変更について説明します。

10.3.2.2. Sendmail のデフォルトのインストール

Sendmail を使用するには、**root** で以下を実行し、まず使用中のシステムに*sendmail* パッケージがインストールされていることを確認します。

```
~]# yum install sendmail
```

Sendmail を設定するには、**root** で以下を実行し、使用中のシステムに*sendmail-cf* パッケージがインストールされていることを確認します。

```
~]# yum install sendmail-cf
```

yum を使ったパッケージのインストールについては「パッケージのインストール」を参照してください。

Sendmail を使用する前に、デフォルトの MTA が Postfix から切り替わっている必要があります。デフォルトの MTA の切り替え方法については「メール転送エージェント (MTA)」を参照してください。

Sendmail 実行可能ファイルは **/sendmail** です。

Sendmail の長い詳細設定ファイルは **/etc/mail/sendmail.cf** です。**sendmail.cf** ファイルは直接編集しないようにしてください。Sendmail の設定を変更するには、**/etc/mail/sendmail.mc** ファイルを編集して、元の **/etc/mail/sendmail.cf** ファイルをバックアップした上で、以下のような別の方法で新規設定ファイルを生成します。

- **/etc/mail** にある makefile を使用して新規の **/etc/mail/sendmail.cf** 設定ファイルを作成します。

  ```
  ~]# make all -C /etc/mail/
  ```

  /etc/mail にある生成された他のすべてのファイル (db ファイル) は、必要に応じて再生成されます。旧 makemap コマンドは現在も使用可能です。make コマンドは、**sendmail** サービスを起動もしくは再起動するたびに、自動的に使用されます。

Sendmail の設定に関する詳細は「Sendmail の一般的な設定変更」を参照してください。

以下のような様々な Sendmail 設定ファイルが、**/etc/mail/** ディレクトリーにインストールされています。

- **access** — 電子メールの送信に Sendmail を使用できるシステムを指定します。
- **domaintable** — ドメイン名のマッピングを指定します。
- **local-host-names** — ホストのエイリアスを指定します。
- **mailertable** — 特定のドメインのルーティングを上書きする方法を指定します。
- **virtusertable** — ドメイン固有のエイリアシング (aliasing) 形式を指定し、単一のマシン上における複数の仮想ドメインのホスティングを可能にします。

Sendmail が設定変更を使用可能となる前に、**access**、**domaintable**、**mailertable**、**virtusertable** など **/etc/mail/** にある設定ファイルのいくつかが、実際にそれらの情報をデータベースファイルに保存する必要があります。これらの設定への変更をデータベースファイル内に含めるには、**root** で以下のコマンドを実行します。

```
~]# cd /etc/mail
~]# make all
```

これで、**virtusertable.db**、**access.db**、**domaintable.db**、**mailertable.db**、**sendmail.cf**、および **submit.cf** が更新されます。

上記のデータベースファイルすべておよびカスタムのデータベースファイルを更新するには、以下のフォーマットでコマンドを使用します。

```
make <name>.db all
```

ここでの *<name>* は、更新するカスタムデータベースファイル名になります。

単一のデータベースを更新するには、以下のフォーマットでコマンドを使用します。

```
make <name>.db
```

ここでの *<name>.db* は、更新するデータベースファイル名になります。

変更を反映させるために、以下を実行して **sendmail** サービスを再起動することもできます。

```
~]# systemctl restart sendmail
```

たとえば、**example.com** ドメイン宛のすべての電子メールが**bob@other-example.com** に配信されるようにするには、**virtusertable** ファイルに以下の行を追加します。

```
@example.com bob@other-example.com
```

変更を完了するには、**virtusertable.db** ファイルを更新する必要があります:

```
~]#  make virtusertable.db all
```

all オプションを使用すると、**virtusertable.db** と **access.db** が同時に更新されます。

10.3.2.3. Sendmail の一般的な設定変更

Sendmail 設定ファイルを変更する場合は、既存ファイルを編集せずに、全く新しい **/etc/mail/sendmail.cf** ファイルを生成するのが最適な方法です。

警告

sendmail.cf ファイルを置き換えたり変更したりする前に、バックアップコピーを作成してください。

希望する機能を Sendmail に追加したい場合は、**root** で **/etc/mail/sendmail.mc** ファイルを編集します。編集が終了したら、**sendmail** サービスを再起動します。*m4* パッケージがインストールされている場合は、**m4** マクロプロセッサーが新しい**sendmail.cf** 設定ファイルを自動的に生成します。

```
~]# systemctl restart sendmail
```

重要

デフォルトの **sendmail.cf** ファイルでは、Sendmail はローカルコンピューター以外のホストからのネットワーク接続を受け入れないように設定されています。Sendmail を他のクライアント用のサーバーとして設定するには、**/etc/mail/sendmail.mc** ファイルを編集して、**DAEMON_OPTIONS** 指示文の **Addr=** オプションで指定されているアドレスを**127.0.0.1** からアクティブなネットワークデバイスの IP アドレスに変更するか、行頭に **dnl** を付けて、**DAEMON_OPTIONS** 指示文をすべてコメントアウトします。終了したら、サービスを再起動して **/etc/mail/sendmail.cf** を再生成します。

```
~]# systemctl restart sendmail
```

Red Hat Enterprise Linux のデフォルトの設定ファイルは、大半の **SMTP** 専用サイトで機能します。ただし、*UUCP* (*UNIX-to-UNIX Copy Protocol*) サイトでは機能しません。UUCP メール転送を使用する場合は、**/etc/mail/sendmail.mc** ファイルを再設定して、新しい **/etc/mail/sendmail.cf** ファイルを生成する必要があります。

/usr/share/sendmail-cf ディレクトリー下のディレクトリーにあるファイルを編集する前には、**/usr/share/sendmail-cf/README** ファイルを確認してください。**/etc/mail/sendmail.cf** ファイルの今後の設定に影響を及ぼす場合があるためです。

10.3.2.4. マスカレード

一般的な Sendmail の設定の 1 つとして、単一のマシンがネットワーク上の全マシンのメールのゲートウェイとして機能するように設定する方法があります。たとえば、ある企業が **mail.example.com** という名前のマシンですべての電子メールを処理して、すべての送信メールに対して一貫した返信アドレスを割り当てるとします。

このような状況では、Sendmail サーバーは、返信アドレスが **user@host.example.com** ではなく **user@example.com** となるように、その企業のネットワーク上のマシン名をマスカレードする必要があります。

これを行うには、**/etc/mail/sendmail.mc** に以下の行を追加します。

```
FEATURE(always_add_domain)dnl
FEATURE(`masquerade_entire_domain')dnl
FEATURE(`masquerade_envelope')dnl
FEATURE(`allmasquerade')dnl
MASQUERADE_AS(`example.com.')dnl
MASQUERADE_DOMAIN(`example.com.')dnl
MASQUERADE_AS(example.com)dnl
```

m4 マクロプロセッサーを使用して新しい**sendmail.cf** ファイルを生成すると、この設定によりネットワーク内のメールはすべて **example.com** から送信されたかのように表示されます。

メールサーバー、**DNS** および **DHCP** サーバー、またプロビジョニングアプリケーションの管理者は、組織内で使用するホスト名のフォーマットに合意するべきであることに注意してください。推奨される命名プラクティスについては、Red Hat Enterprise Linux 7 Networking Guide を参照してください。

10.3.2.5. スパムの防止

電子メールのスパムは、通信を要求したことがないユーザーから受信した、不要な迷惑メールと定義することができます。これは、破壊的で高コストを伴う、広く蔓延したインターネット通信標準の悪用です。

Sendmail を使うと、迷惑メールの送信に使用されている新たなスパム技術を比較的に簡単にブロックすることができます。さらに、数多くの一般的なスパム手法もデフォルトでブロックします。Sendmail で利用できる主要なアンチスパム機能は *ヘッダーのチェック*、*リレーの否認* (バージョン 8.9 からデフォルト)、*アクセスのデータベース*、*送信者情報の確認* です。

たとえば、リレーとも呼ばれる **SMTP** メッセージの転送は、Sendmail バージョン 8.9 以降デフォルトでは無効になっています。この変更前には、Sendmail はメールホスト (**x.edu**) に対して、ある当事者 (**y.com**) からのメッセージを受け入れるよう指示し、そのメッセージを別の当事者 (**z.net**) に送信していました。しかし、現在は任意のドメインがサーバーを介してメールをリレーするよう Sendmail を設定する必要があります。リレードメインを設定するには、**/etc/mail/relay-domains** ファイルを編集して Sendmail を再起動してください。

```
~]# systemctl restart sendmail
```

しかし、ユーザーはインターネット上のサーバーからスパムを送信されることもあります。その場合は、**/etc/mail/access** ファイルで利用可能な Sendmail のアクセス制御機能を使用して、迷惑なホストからの接続を阻止することができます。以下の例は、このファイルを使用したブロックの方法と Sendmail サーバーへのアクセスを具体的に許可する方法を示しています。

```
badspammer.com ERROR:550 "Go away and do not spam us anymore"
tux.badspammer.com OK 10.0 RELAY
```

この例では、**badspammer.com** から送信された電子メールはいずれも 550 RFC-821 準拠のエラーコードでブロックされ、メッセージは送り返されます。**tux.badspammer.com** のサブドメインから送信される電子メールは受け入れられます。最後の行は、10.0.*.* ネットワークから送信された電子メールはいずれもメールサーバーを通してリレー可能であることを示しています。

/etc/mail/access.db ファイルはデータベースであるため、**makemap** コマンドを使用して変更を更新します。これを行うには、**root** で以下のコマンドを使用します。

```
~]# makemap hash /etc/mail/access < /etc/mail/access
```

メッセージヘッダーの分析により、ヘッダーのコンテンツに基づいたメール拒否が可能となります。**SMTP** サーバーは、電子メールの送信経路に関する情報をメッセージヘッダー内に保管します。メッセージがある MTA から別の MTA に送られると、その他すべての **Received** ヘッダーの上にそれぞれ **Received** ヘッダーを付けます。ただし、注意すべき重要なポイントは、この情報はスパム送信者が変更できるという点です。

上記の例は、アクセスの許可や阻止に関する Sendmail が持つ機能のほんの一部です。詳細情報と他の例は、**/usr/share/sendmail-cf/README** ファイルを参照してください。

Sendmail は、メールの配信時に Procmail MDA を呼び出すため、SpamAssassin のようなスパムフィルタリングプログラムを使用して、ユーザーに対してスパムを識別してファイルー保存することも可能です。SpamAssassin の使用に関する詳細は、「スパムフィルター」を参照してください。

10.3.2.6. LDAP での Sendmail の使用

LDAP の使用は、大規模なグループからある特定のユーザーに関する特定の情報を検索する、非常に迅速かつ強力な方法です。例えば、**LDAP** サーバーを使用すると、一般的な企業ディレクトリーから特定の電子メールアドレスをユーザーの姓で検索することができます。この種の実装では、**LDAP** はほどんど Sendmail から分離されており、**LDAP** が階層別のユーザー情報を保存して、Sendmail は事前にアドレスが入力された電子メールメッセージの形式で **LDAP** のクエリ結果を知らされるだけです。

ただし、Sendmail は **LDAP** との非常に大規模な統合をサポートします。この統合では、Sendmail は **LDAP** を使用して、中規模からエンタープライズレベルの組織をサポートするために連携する異なるメールサーバー上の **/etc/aliases** や **/etc/mail/virtusertables** などの個別に管理されているファイルを置き換えます。一言でいうならば、**LDAP** はメールルーティングレベルを Sendmail とその別個の設定ファイルから多くの様々なアプリケーションで活用できる強力な **LDAP** クラスタに抽象化します。

Sendmail の現行版は **LDAP** に対応しています。**LDAP** を使用して Sendmail を拡張するには、最初に **OpenLDAP** などの **LDAP** サーバーを稼働して、適切な設定を行います。次に、**/etc/mail/sendmail.mc** を編集して以下を追加します。

```
LDAPROUTE_DOMAIN('yourdomain.com')dnl
FEATURE('ldap_routing')dnl
```

注記

これは **LDAP** を使った非常に基本的な Sendmail の設定にすぎません。実際の設定は**LDAP** の実装によりこれとは大幅に異なる可能性があります。特に、共通の **LDAP** サーバーを使用するために数種の Sendmail マシンを設定する場合がそうです。

詳しい **LDAP** のルーティング設定に関する説明と例は、**/usr/share/sendmail-cf/README** を参照してください。

次に、**m4** マクロプロセッサーを実行し、再び Sendmail を再起動して、**/etc/mail/sendmail.cf** ファイルを再作成します。この方法については、「Sendmail の一般的な設定変更」 を参照してください。

LDAP の詳細は 「OpenLDAP」 を参照してください。

10.3.3. Fetchmail

Fetchmail は、リモートサーバーから電子メールを取得してローカルの MTA に配信する MTA です。多くのユーザーは、リモートサーバー上にあるメッセージをダウンロードするプロセスと、MUA で電子メールを読み取り、整理するプロセスを別々にする機能性を評価してます。ダイアルアップユーザーのニーズを踏まえて設計されている Fetchmail は、**POP3** や **IMAP** などの数々のプロトコルを使用して、メールスプールファイルに接続し、すべての電子メールメッセージを迅速にダウンロードします。また、必要に応じて、電子メールメッセージを **SMTP** サーバーに転送することもできます。

注記

Fetchmail を使用するには、**root** で以下を実行し、まず使用中のシステムに*fetchmail* パッケージがインストールされていることを確認します。

```
~]# yum install fetchmail
```

yum を使ったパッケージのインストールについては「パッケージのインストール」を参照してください。

Fetchmail は、ユーザーのホームディレクトリー内の **.fetchmailrc** ファイルを使用して各ユーザー用に設定されています。これがない場合は、ホームディレクトリーに **.fetchmailrc** ファイルを作成してください。

Fetchmail は **.fetchmailrc** ファイル内の詳細設定を使用して、リモートサーバー上にある電子メールを確認し、ダウンロードします。次に、電子メールをローカルマシン上のポート **25** に配信し、ローカルの MTA を使用して電子メールを適正なユーザーのスプールファイルに配置します。Procmail が利用できる場合は起動して電子メールをフィルターし、MUA が読み込むことができるようにメールボックスに配置します。

10.3.3.1. Fetchmail の設定オプション

Fetchmai を実行する時にすべての必要なオプションをコマンドライン上で渡し、リモートサーバー上の電子メールを確認することは可能ですが、**.fetchmailrc** ファイルを使用した方がはるかに簡単です。希望の設定オプションを **.fetchmailrc** ファイル内に配置し、それらのオプションが、**fetchmail** コマンドを発行する時に毎回使用されるようにします。Fetchmail の実行時にオプションを上書きしたい場合は、コマンドライン上でそのオプションを指定します。

ユーザーの **.fetchmailrc** ファイルには、3 つのクラスの設定オプションが格納されています。

- **グローバルオプション** — プログラムの動作を制御する、または電子メールを確認する全接続の設定を提供する指示を Fetchmail に与えます。
- **サーバーオプション** — ポーリングされるサーバーに関する必要情報を指定します。ホスト名をはじめ、確認するポートやタイムアウトになるまでの秒数などの特定の電子メールサーバーの設定などです。こうしたオプションは、該当するサーバーを使用する全ユーザーに影響を及ぼします。
- **ユーザーオプション** — 指定された電子メールサーバーを使用して、電子メールの認証や確認を行うにあたって必要なユーザー名、パスワードなどの情報を格納します。

グローバルオプションは、**.fetchmailrc** ファイルの最上部に表示されます。その次には 1 つまたは複数のサーバーオプションがあり、それぞれが Fetchmail が確認すべき異なる電子メールサーバーを指定します。サーバーオプションの次には、その電子メールサーバーを確認する各ユーザーアカウントのユーザーオプションがあります。サーバーオプションと同様に、複数のユーザーオプションを指定することで特定のサーバーでの使用、同一サーバー上の複数の電子メールアカウントの確認を行うことができます。

サーバーオプションを **.fetchmailrc** ファイルで利用するには、サーバーの情報の先頭に**poll** または **skip** などの特別なオプションの動詞を使用します。**poll** アクションは、Fetchmail の実行時にこのサーバーオプションを使用して、指定されたユーザーオプションで電子メールを確認するよう Fetchmail に指示します。ただし、**skip** アクションの後にあるサーバーオプションは、Fetchmail が呼び出された時にサーバーのホスト名が指定されていない限り確認されません。**skip** オプションは、特定して呼び出された時にスキップされたサーバーのみを確認し、現在稼働中の設定には影響を及ぼさないため **.fetchmailrc** ファイルの設定をテストする時に役立ちます。

サンプルの **.fetchmailrc** ファイルは以下のとおりです:

```
set postmaster "user1"
set bouncemail

poll pop.domain.com proto pop3
    user 'user1' there with password 'secret' is user1 here

poll mail.domain2.com
    user 'user5' there with password 'secret2' is user1 here
    user 'user7' there with password 'secret3' is user1 here
```

この例では、グローバルオプションにより、最終手段としてユーザーに電子メールが送信されるように指定されており (**postmaster** オプション)、すべての電子メールエラーは送信者ではなく、ポストマスターに送信されます (**bouncemail** オプション)。**set** アクションは、この行にグローバルオプションが含まれていることを Fetchmail に伝えます。次に、2 つの電子メールサーバーが指定されます。1 つは **POP3** を使用して確認するように設定され、もう 1 つは様々なプロトコルを試みて機能するものを見つけるように設定されます。第 2 のサーバーオプションを使用して 2 人のユーザーが確認されますが、どのユーザー宛でも見つかった電子メールはすべて、**user1** のメールスプールに送信されます。これにより、単一の MUA 受信ボックスに表示させながら、複数のサーバー上で複数のメールボックスをチェックすることが可能となります。各ユーザーの固有の情報は、**user** アクションで開始します。

注記

ユーザーはパスワードを **.fetchmailrc** ファイルに配置する必要はありません。**with password '<password>'** のセクションを省略した場合は、Fetchmail は起動時にパスワードを要求するようになります。

Fetchmail には、数々のグローバルオプション、サーバーオプション、ローカルオプションがあります。これらの多くのオプションは、ほとんど使用されないか、非常に独特な状況にのみ適用されます。**fetchmail** の man ページに、各オプションの詳細が記載されていますが、最も一般的なものを以下の 3 セクションで説明します。

10.3.3.2. グローバルオプション

グローバルオプションは、**set** アクションの後に、それぞれ 1 行ずつ配置する必要があります。

- **daemon <*seconds*>** — Fetchmail がバックグラウンドに残るデーモンモードを指定します。*<seconds>* を Fetchmail がサーバーをポーリングするまでの待機時間の秒数に置き換えます。
- **postmaster** — 配信に問題が生じた場合にローカルユーザーがメールを送信するよう指定します。
- **syslog** — エラーとステータスメッセージのログファイルを指定します。デフォルトは **/var/log/maillog** です。

10.3.3.3. サーバーオプション

サーバーオプションは **.fetchmailrc** 内で、**poll** または **skip** アクションの後にそれぞれの行に配置される必要があります。

- **auth <*auth-type*>** — *<auth-type>* を使用する認証のタイプに置き換えます。デフォルトでは、**password** 認証が使用されますが、一部のプロトコルは**kerberos_v5**、**kerberos_v4** および **ssh** を含む他のタイプの認証をサポートしています。**any** の認証タイプを使用した場合、Fetchmail は最初にパスワードを必要としない方法を試みます。次に、パスワードをマスクする方法を試みた後、最後にサーバーに暗号化されていないパスワードを送信して認証を試みます。

- **interval <number>** — 指定されたサーバーを**<number>** の時間間隔でポーリングし、設定された全サーバー上の電子メールを確認します。このオプションは、通常ユーザーがほとんどメッセージを受信しない電子メールサーバーに使用されます。
- **port <port-number>** — <port-number> をポート番号に置き換えます。この値は、指定されたプロトコルのデフォルトのポート番号を上書きします。
- **proto <protocol>** — <protocol> を、サーバー上のメッセージを確認する時に使用する**pop3** や **imap** などのプロトコルに置き換えます。
- **timeout <seconds>** — <seconds> を、Fetchmail が接続の試行を打ち切った後でサーバーが非アクティブとなる秒数に置き換えます。この値が設定されていない場合は、デフォルトの **300** 秒が使用されます。

10.3.3.4. ユーザーオプション

ユーザーオプションは、サーバーオプションの下の各行に配置される場合と、サーバーオプションと同じ行に配置される場合があります。いずれの場合も、定義されるオプションは **user** オプション (以下で説明) にしたがう必要があります。

- **fetchall** — 既読メッセージを含め Fetchmail がキューにあるすべてのメッセージをダウンロードするように命令します。デフォルトでは、Fetchmail は新規メッセージのみをダウンロードするようになっています。
- **fetchlimit <number>** — <number> を、停止する前に取得されるメッセージ数に置き換えます。
- **flush** — 新規メッセージを取得する前に、キューにあるすべての既読メッセージを削除します。
- **limit <max-number-bytes>** — <max-number-bytes> を、Fetchmail で取得する時に許容されているメッセージの最大バイトサイズに置き換えます。このオプションは、大型のメッセージのダウンロードに時間がかかりすぎる場合の低速のネットワークリンクに有用です。
- **password '<password>'** — <password> をユーザーのパスワードに置き換えます。
- **preconnect "<command>"** — <command> を、ユーザー宛のメッセージを取得する前に実行するコマンドに置き換えます。
- **postconnect "<command>"** — <command> を、ユーザー宛のメッセージを取得した後に実行するコマンドに置き換えます。
- **ssl** — SSL 暗号化をアクティベートします。
- **user "<username>"** — <username> を、Fetchmail がメッセージの取得に使用するユーザー名に置き換えます。**このオプションは、他のすべてのユーザーオプションの前に付ける必要があります。**

10.3.3.5. Fetchmail のコマンドオプション

fetchmail コマンドの実行時にコマンドライン上で使用される Fetchmail オプションの大半は **.fetchmailrc** 設定オプションを反映します。この方法では、Fetchmail は設定ファイルの有無を問わず使用することができます。 **.fetchmailrc** 設定オプションは、**.fetchmailrc** ファイルに残しておいた方が簡単なため、大半のユーザーはコマンドライン上では使用しません。

fetchmail コマンドは、特定の用途のオプションと併せて実行した方が望ましい場合もあります。コマンドラインで指定されるオプションはいずれも設定ファイルオプションを上書きするため、エラーが発生した場合は、コマンドオプションを使用してエラーの原因になっている **.fetchmailrc** 設定を一時的に上書きすることが可能です。

10.3.3.6. 情報提供またはデバッグのオプション

fetchmail コマンドの後に使用されるオプションの一部は、重要な情報を提供する場合があります。

- **--configdump** — **.fetchmailrc** および Fetchmail のデフォルト値からの情報に基づいた、可能なオプションをすべて表示します。このオプションを使用すると、どのユーザーの電子メールも取得されません。
- **-s** — Fetchmail をサイレントモードで実行し、**fetchmail** コマンドの後にエラー以外のメッセージが表示されないようにします。
- **-v** — Fetchmail を verbose モードで実行し、Fetchmail とリモート電子メールサーバー間のすべての通信を表示します。
- **-V** — 詳細なバージョン情報の表示、グローバルオプションの一覧表示、電子メールプロトコルや認証メソッドなどの各ユーザーと使用する設定の表示を行います。このオプションを使用すると、どのユーザーの電子メールも取得されません。

10.3.3.7. 特別なオプション

これらのオプションは **.fetchmailrc** ファイルによく見られるデフォルト値を上書きする時に役立つ場合があります。

- **-a** — Fetchmail は、新規または既読を問わず、すべてのメッセージをリモートの電子メールサーバーからダウンロードします。デフォルトでは、Fetchmail は新規メッセージのみをダウンロードします。
- **-k** — Fetchmail はメッセージをダウンロードした後、リモートの電子メールサーバー上にメッセージを残します。このオプションを使用すると、メッセージをダウンロード後に削除するデフォルトの動作は上書きされます。
- **-l *<max-number-bytes>*** — Fetchmail は一定のサイズを超えるメッセージはダウンロードせず、リモートの電子メールサーバー上に残します。
- **--quit** — Fetchmail デーモンのプロセスを終了します。

その他のコマンドと **.fetchmailrc** オプションについては、**fetchmail** の man ページを参照してください。

10.3.4. MTA の設定

MTA は電子メールの送信に不可欠です。**Evolution** や **Mutt** などの *MUA* を使用してメールの読み取り、作成を行うことができます。ユーザーが MUA から電子メールを送信すると、メッセージは MTA に渡されます。MTA は一連の MTA を通じて、メッセージが送信先に届くまで送信します。

ユーザーがシステムから電子メールを送信する予定でなくても、一部の自動化されたタスクまたはシステムプログラムは **mail** コマンドを使用して、ログメッセージを含む電子メールをローカルシステムの**root** ユーザーに送信する場合があります。

Red Hat Enterprise Linux 7 は、Sendmail と Postfix の 2 つの MTA を提供しています。どちらもインストールされている場合は、Postfix がデフォルトの MTA となります。Sendmail は Red Hat Enterprise Linux 7 では非推奨となることに注意してください。

10.4. メール配信エージェント (MDA)

Red Hat Enterprise Linux には Procmail と **mail** の 2 つの主要な MDA が含まれています。どちらのアプリケーションも LDA とみなされ、電子メールを MTA のスプールファイルからユーザーのメールボックスに移動します。ただし、Procmail の方が堅牢なフィルタリングシステムを提供します。

このセクションでは、Procmail についてのみ詳しく説明します。**mail** コマンドの詳細は、man ページ (**man mail**) を参照してください。

電子メールがローカルホストのメールスプールファイルに配置されると、Procmail が配信とフィルタリングを行います。Procmail は強力な上、システムリソースの使用が低いため、幅広く利用されています。Procmail は、電子メールクライアントアプリケーションが読み取る電子メールを配信するという重要な役割を果たします。

Procmail は、様々な方法で呼び出すことができます。MTA が電子メールをメールスプールファイルの中に配置すると、常にProcmail が起動します。次に、Procmail は電子メールを MUA のためにフィルタリング、ファイル保存して、終了します。別の方法としては、メッセージを受信すると常に Procmail を実行するように MUA を設定し、メッセージが正しいメールボックスに移動するようにできます。デフォルトでは、**/etc/procmailrc** または **~/.procmailrc** ファイル (別名 *rc* ファイル) がユーザーのホームディレクトリーにあると、MTA が新規メッセージを受信するたびに Procmail が呼び出されます。

デフォルトでは、**/etc/** ディレクトリーにはシステム全体の**rc** ファイルは存在せず、ユーザーのホームディレクトリーに **.procmailrc** ファイルは存在しません。このため、Procmail を使用するには、各ユーザーが特定の環境変数とルールを用いて **.procmailrc** ファイルを構築する必要があります。

Procmail が電子メールメッセージに対応するかどうかは、そのメッセージが **rc** ファイルの特定の条件または *レシピ* と適合するかどうかによって決まります。あるメッセージが任意のレシピと適合する場合は、電子メールは特定のファイルに配置、削除、それ以外は処理されます。

Procmail が起動すると、電子メールメッセージを読み込み、ヘッダー情報から本文を切り離します。次に、Procmail は **/etc/procmailrcs** ディレクトリー内の **/etc/procmailrc** ファイルと **rc** ファイルで、デフォルトのシステム全体の Pcocmail 環境用変数とレシピを探します。その後 Procmail は、ユーザーのホームディレクトリー内で **.procmailrc** ファイルを探します。多くのユーザーは、Procmail 用に追加の **rc** ファイルも作成します。これは、ホームディレクトリーの**.procmailrc** ファイル内で参照されます。

10.4.1. Procmail の設定

Procmail の設定ファイルには、重要な環境変数が含まれています。これらの変数は、並べ替えするメッセージ、およびどのレシピとも適合しないメッセージの処理を指定します。

これらの環境変数は通常 **~/.procmailrc** ファイルの冒頭に、以下のような形式で表示されます。

```
<env-variable>="<value>"
```

この例では、***<env-variable>*** が変数の名前で、***<value>*** が変数を定義します。

大半の Procmail ユーザーが使用しない環境変数が多々あります。重要な環境変数の多くは、既にデフォルト値で定義されています。大抵の場合は、以下のような変数が使用されます。

- **DEFAULT** — どのレシピにも適合しないメッセージが配置された場合のデフォルトのメールボックスを設定します。

  デフォルトの **DEFAULT** 値は、**$ORGMAIL** と同じです。

- **INCLUDERC** — 照合するメッセージに対する多くのレシピを格納する追加の**rc** ファイルを指定します。これにより、Procmail レシピの一覧は、スパムのブロック、電子メール一覧の管理など異なる役害を果たす個別のファイルに分割されます。その結果、そうしたファイルは、ユーザーの **~/.procmailrc** ファイル内のコメント文字を使用して、オンやオフにすることができます。

  例えば、ユーザーの **.procmailrc** ファイル内の行は以下のようになります。

  ```
  MAILDIR=$HOME/Msgs INCLUDERC=$MAILDIR/lists.rc
  INCLUDERC=$MAILDIR/spam.rc
  ```

電子メールの一覧の Procmail フィルターをオフにしつつスパム制御を維持したい場合は、最初の **INCLUDERC** の行をハッシュ記号 (#) でコメントアウトします。

- **LOCKSLEEP** — Procmail が特定のロックファイルの使用を試みる時間間隔を秒単位で設定します。デフォルトは **8** 秒です。
- **LOCKTIMEOUT** — ロックファイルが最後に修正された後、Procmail がそれは古くて削除可能であると見なすまでに経過する必要のある時間を秒単位で設定します。デフォルトは **1024** 秒です。
- **LOGFILE** — Procmail の情報やエラーメッセージが書き込まれるファイルです。
- **MAILDIR** — Procmail 用の現在作業中のディレクトリーを設定します。設定されると、他の Procmail のパスはすべてこのディレクトリーに対する相対パスになります。
- **ORGMAIL** — 元のメールボックス、またはデフォルトやレシピで必要な場所にメッセージを配置できなかった場合にメッセージを配置する別の場所を指定します。

  デフォルトでは、**/var/spool/mail/$LOGNAME** の値が使用されます。
- **SUSPEND** — スワップ領域など必要なリソースが利用できない場合に、Procmail が一時停止する時間を秒単位で設定します。
- **SWITCHRC** — 追加の Procmail レシピが格納されている外部ファイルをユーザーが指定できるようにします。これは、**INCLUDERC** オプションとよく似ていますが、レシピのチェックが参照先の設定ファイル上で実際に停止され、**SWITCHRC** の指定するファイル上のレシピのみが使用される点が異なります。
- **VERBOSE** — Procmail が詳細な情報をログ記録するようにします。このオプションはデバッグに役立ちます。

その他の重要な環境変数は、シェルから引き出されます。例えば、ログイン名の **LOGNAME**、ホームディレクトリーの場所である **HOME**、デフォルトのシェルである **SHELL** などです。

環境変数すべてについての包括的な説明やデフォルト値については、**procmailrc** の man ページを参照してください。

10.4.2. Procmail のレシピ

多くの場合、新規ユーザーが Procmail の使用法を学習するにあたって最も難しいと感じるのは、レシピの構築です。これは、レシピが適合するストリングの条件を指定するために *正規表現* を使用してメッセージ照合を行うためです。ただ、正規表現の構築はそれほど難しくなく、読んで理解することも簡単です。その上、Procmail のレシピを書く方法は、正規表現にかかわらず一貫性があるため、例を使って学習すると簡単です。Procmail のレシピの例は、「レシピの例」を参照してください。

Procmail レシピは以下の書式を使用します:

```
:0<flags>: <lockfile-name> * <special-condition-character>
        <condition-1> * <special-condition-character>
        <condition-2> * <special-condition-character>
        <condition-N>
        <special-action-character>
        <action-to-perform>
```

Procmail レシピの最初の 2 文字は、コロンとゼロです。ゼロの後に様々なフラグを配置して、Procmail がレシピを処理する方法を制御します。***<flags>*** セクションの後ろにコロンを付けると、このメッセージ用にロックファイルが作成されることを示しています。ロックファイルが作成されると、その名前は ***<lockfile-name>*** を置き換えて指定することが可能です。

レシピは、メッセージと適合させる様々な条件を格納できます。条件がない場合は、あらゆるメッセージがレシピと適合することになります。正規表現は、メッセージ照合を容易にするために、一部の条件で使用されます。複数の条件を使用する場合、アクションが実行されるためにはすべてが適合しなければなりません。条件は、レシピの 1 行目に設定されているフラグに基づいてチェックされます。アスタリスク文字 (*) の後にオプションの特殊文字を配置すると、さらに条件を制御できます。

<action-to-perform> 引数は、メッセージが条件の 1 つに適合する場合にアクションを実行するよう指定します。1 つのレシピにつき 1 つのアクションのみとなります。多くの場合、メールボックスの名前がここで使用され、適合するメッセージをファイルに誘導し、電子メールを効果的に並べ替えます。特別なアクションの文字は、アクションが指定される前に使用することもできます。詳細は 「特別な条件とアクション」 を参照してください。

10.4.2.1. 配信レシピと非配信レシピの比較

レシピがある特定のメッセージと適合した場合に使用されるアクションにより、それが *配信* レシピ、または *非配信* レシピと見なされるかが判断されます。配信レシピには、ファイルへのメッセージの書き込み、別のプログラムへのメッセージ送信、別の電子メールアドレスへのメッセージ転送などのアクションが含まれています。非配信レシピは、*ネストされたブロック*などその他のアクションをカバーします。ネストされたブロックは、中括弧 **{ }** で囲まれたアクションセットで、レシピの条件に適合するメッセージで実行されます。ネストされたブロックは、互いにネストさせることができるため、メッセージに対するアクションを特定、実行するにあたっての制御力が強化されます。

メッセージが配信レシピと適合すると、Procmail は指定されたアクションを実行し、その他のレシピとメッセージとの比較を停止します。非配信レシピと適合するメッセージの場合は、他のレシピに対する照合は継続されます。

10.4.2.2. フラグ

フラグは、レシピの条件をメッセージに照合する方法、またはそれを行うかどうかを決定するにあたって不可欠です。一般的に使用されるフラグは以下の通りです。

- **A** — **A** や **a** のフラグが付いていない以前のレシピもこのメッセージに適合する場合にのみ、このレシピが使用されることを指定します。
- **a** — **A** や **a** のフラグが付いた以前のレシピもこのメッセージに適合し、**かつ** 正常に完了した場合にのみこのレシピが使用されることを指定します。
- **B** — メッセージの本文を解析し、適合する条件を検索します。
- **b** — ファイルへのメッセージの書き込みや転送など、結果として生じるアクションにその本文を使用します。これはデフォルトの動作です。
- **c** — 電子メールのカーボンコピーを生成します。必要なアクションをメッセージで実行し、メッセージのコピーは **rc** のファイル内で引き続き処理することができるため、レシピの配信に役立ちます。
- **D** — **egrep** の照合で大文字と小文字を区別します。デフォルトでは、照合プロセスでは大文字と小文字を区別していません。
- **E** — **A** フラグと類似していますが、レシピ内の条件は、直前にある**E** フラグなしのレシピが適合しない場合のみに、メッセージと照合されます。これは *else* アクションと類似しています。
- **e** — 直前のレシピで指定されたアクションが失敗した場合のみ、レシピがメッセージに照合されます。
- **f** — フィルターとしてパイプを使用します。
- **H** — メッセージのヘッダーを解析し、適合する条件を検索します。これはデフォルトの動作です。
- **h** — 結果として生じるアクションでヘッダーを使用します。これはデフォルトの動作です。

- **w** — Procmail に対して、指定されたフィルターまたはプログラムが終了するのを待ち、メッセージがフィルターされたと見なす前に正常に終了したかどうかを報告するよう指示します。
- **W** — 「プログラム障害」のメッセージが抑制されている点を除いては**w** と全く同じです。

その他のフラグの詳細な一覧は、**procmailrc** の man ページを参照してください。

10.4.2.3. ローカルロックファイルの指定

ロックファイルは、Procmail で複数のプロセスが 1 つのメッセージを同時に変更しないようにするために非常に役立ちます。ローカルロックファイルを指定するには、レシピの 1 行目の任意のフラグの後にコロン (**:**) を配置します。これにより、送信先のファイル名に基づいたローカルロックファイルと、**LOCKEXT** のグローバル環境変数で設定されたものすべてが作成されます。

別の方法としては、このレシピで使用するローカルロックファイルの名前をコロンの後に指定します。

10.4.2.4. 特別な条件とアクション

Procmail レシピの条件とアクションの前に使用される特殊文字により、解釈の仕方が変わります。

以下の文字は、レシピの条件の行頭でアスタリスク文字 (*****) の後に使用できます。

- **!** — 条件の行では、この文字により条件が反転し、条件がメッセージに一致しない場合にのみ、適合が発生するようになります。
- **<** — メッセージが指定されたバイト数内に収まっているかどうかを確認します。
- **>** — メッセージが指定されたバイト数を超えているかどうかを確認します。

以下の文字は、特別なアクションを実行するために使用されます。

- **!** — アクションの行では、この文字は Procmail にメッセージを指定された電子メールアドレスに転送するように指示します。
- **$** — **rc** ファイルで以前に設定された変数を参照します。多くの場合、様々なレシピによって参照される共通のメールボックスを設定するために使用されます。
- **|** — メッセージを処理するための特定のプログラムを起動します。
- **{** および **}** — 適合するメッセージに適用する追加のレシピを格納するために使用される、ネストされたブロックを構築します。

アクションの行頭に特殊文字を使用しない場合、Procmail はアクションの行がメッセージを書き込むためのメールボックスを指定していると仮定します。

10.4.2.5. レシピの例

Procmail は極めて柔軟性の高いプログラムですが、この柔軟性が原因で、新規ユーザーが Procmail のレシピを一から作成するのが難しい場合があります。

Procmail レシピの条件を構築するスキルを向上させる最適な方法は、正規表現をしっかり理解することに加えて、他の人が構築した多くの例を参照することから始まります。正規表現についての詳細な説明は、本セクションの範囲外となります。Procmail のレシピの構造と役立つ Procmail のサンプルレシピは、インターネット上の様々なところに掲載されています。正規表現の適切な使用、調整方法は、これらのレシピ例を参照してください。また、基礎的な正規表現のルールに関する初歩的な情報は、**grep** の man ページを参照してください。

以下にあげる簡単な例は、Procmail のレシピの基本構造を記載しており、構造をさらに複雑にするための基盤を示しています。

以下の例に示すように、基本的なレシピには条件さえも含まれていません。

```
:0: new-mail.spool
```

1 行目では、ローカルロックファイルが作成されたことは指定していますが、名前は指定していません。そのため、Procmail は送信先のファイル名を使用して **LOCKEXT** 環境変数で指定された値を追加します。条件が指定されていないため、すべてのメッセージがこのレシピに適合し、**MAILDIR** 環境変数によって指定されたディレクトリー内にある **new-mail.spool** と呼ばれる単一のスプールファイルに配置されます。その後、MUA はこのファイル内のメッセージを閲覧できるようになります。

このような基本レシピは、**rc** ファイルの末尾に配置され、メッセージをデフォルトの場所に振り向けます。

以下の例では、特定の電子メールアドレスからのメッセージを照合して、削除します。

```
:0 * ^From: spammer@domain.com /dev/null
```

この例では、**spammer@domain.com** から送信されたメッセージはすべて **/dev/null** デバイスに送信され、削除されます。

警告

メッセージを **/dev/null** に送信して永久に削除してしまう前に、ルールが目的通りに機能していることを確認してください。レシピが間違えて目的以外のメッセージを対象にすると、それらのメッセージは消えてしまい、ルールのトラブルシューティングが困難になります。

この問題に対処する優れた方法としては、レシピのアクションを特別なメールボックスに移動させることです。これで、メールボックスを時折確認して、誤検知を探すことができます。メッセージが間違って適合されることがなく満足できる状態になったら、そのメールボックスは削除して、メッセージを **/dev/null** に送信するよう指示します。

以下のレシピでは、特定のメーリングリストから送信された電子メールを取得して、特定のフォルダに配置します。

```
:0: * ^(From|Cc|To).*tux-lug tuxlug
```

tux-lug@domain.com のメーリングリストから送信されたメッセージはすべて、MUA 用に自動的に **tuxlug** メールボックスに配置されます。**From**、**Cc**、**To** の行にメーリングリストの電子メールアドレスが入っている場合は、この例の条件がメッセージに適合する点に注意してください。

さらに詳しい強力なレシピについては、「その他のリソース」の Procmail に関する数々のオンライン資料を参照してください。

10.4.2.6. スパムフィルター

Procmail は、新規の電子メールを受信すると Sendmail、Postfix、Fetchmail によって呼び出されるため、スパム対策の強力なツールとして使用できます。

これは、Procmail が SpamAssassin と併用された場合に特に有効です。これらの 2 つのアプリケーションを併用すると、スパムメールを迅速に特定して、並び替えまたは破棄することができます。

SpamAssassin はヘッダー分析、テキスト分析、ブラックリスト、スパム追跡データベース、自己学習型 Bayesian スパム分析を使用して、迅速かつ正確にスパムの特定とタグ付けを行います。

注記

SpamAssassin を使用するには、**root** で以下を実行して、最初にご使用のシステムに *spamassassin* パッケージがインストールされていることを確認します。

```
~]# yum install spamassassin
```

yum を使ったパッケージのインストールについては 「パッケージのインストール」を参照してください。

ローカルユーザーが SpamAssassin を使用する最も簡単な方法は、**~/.procmailrc** ファイルの最上部付近に以下の行を配置することです。

```
INCLUDERC=/etc/mail/spamassassin/spamassassin-default.rc
```

/etc/mail/spamassassin/spamassassin-default.rc には、シンプルな Procmail ルールが格納されており、受信するすべての電子メールに対して SpamAssassin をアクティベートします。電子メールがスパムであると判断された場合には、ヘッダー内でタグ付けされて、タイトルの先頭には以下のようなパターンが追加されます。

```
*****SPAM*****
```

電子メールのメッセージ本文にも、スパム診断の理由となった要素の継続的な記録が先頭に追加されます。

スパムとしてタグ付けされた電子メールをファイル保存するには、以下と同様のルールを使用することができます。

```
:0 Hw * ^X-Spam-Status: Yes spam
```

このルールにより、スパムとしてヘッダーにタグ付けされた電子メールはすべて、**spam** と呼ばれるメールボックスにファイル保存されます。

SpamAssassin は Perl スクリプトであるため、ビジー状態のサーバーではバイナリ SpamAssassin デーモン (**spamd**) とクライアントアプリケーション (**spamc**) を使用する必要がある場合があります。ただし、SpamAssassin をこのように設定するには、ホストへの **root** アクセスが必要です。

spamd デーモンを起動するには、以下のコマンドを入力します。

```
~]# systemctl start spamassassin
```

システム起動時に SpamAssassin デーモンを起動するには、**サービス設定ツール** (**system-config-services**) などの initscript ユーティリティーを使用して、**spamassassin** サービスを有効にします。サービスの起動と停止に関する詳細は、*6章systemd によるサービス管理* を参照してください。

Procmail が Perl スクリプトではなく、SpamAssassin クライアントアプリケーションを使用するように設定するには、**~/.procmailrc** ファイルの最上部付近に以下の行を配置します。システム全体の設定の場合は、**/etc/procmailrc** に配置してください。

```
INCLUDERC=/etc/mail/spamassassin/spamassassin-spamc.rc
```

10.5. メールユーザーエージェント (Mail User Agents)

Red Hat Enterprise Linux には、様々な利用可能な電子メールプログラムがあります。**Evolution** のようなグラフィカル電子メールクライアントプログラムと、**Mutt** のようなテキストベースの電子メールプログラムがあります。

本セクションではこれ以降、クライアントとサーバー間の通信のセキュリティー保護について重点的に説明していきます。

10.5.1. 通信のセキュリティー保護

Red Hat Enterprise Linux に装備の **Evolution** や **Mutt** など評判の高い MUA は、SSL 暗号化電子メールセッションを提供します。

暗号化されていないネットワークを行き来する他のサービスと同様に、ユーザー名、パスワード、メッセージ全体などの電子メールに関する重要な情報は、ネットワーク上のユーザーによって傍受、閲覧される可能性があります。また、標準の **POP** および **IMAP** プロトコルは、認証情報を暗号化せずに渡すため、ユーザー名とパスワードはネームサーバー上で渡される時に攻撃者がそれらを収集して、ユーザーのアカウントに侵入することが可能性があります。

10.5.1.1. セキュアな電子メールクライアント

リモートサーバー上の電子メールを確認するように設計されている Linux MUA のほとんどは、SSL 暗号化に対応しています。電子メールを取得する時に SSL を使用するためには、SSL は電子メールクライアントとサーバーの両方で有効である必要があります。

SSL をクライアント側で有効にするのは簡単です。多くの場合は、MUA の設定ウインドウでボタンをクリックするか、MUA の設定ファイルのオプションで有効にすることができます。セキュアな **IMAP** と **POP** には既知のポート番号 (それぞれ**993** と **995**) があり、MUA はそれらを使用してメッセージの認証、ダウンロードを行います。

10.5.1.2. 電子メールクライアントの通信のセキュリティー保護

電子メールサーバー上の **IMAP** および **POP** ユーザーに SSL 暗号化を行うことは簡単です。

最初に SSL 証明書を作成します。これは、*認証局* (*CA*) に SSL 証明書を申請するか、自己署名証明書を作成するかのいずれかの方法で可能です。

警告

自己署名証明書は、テスト目的のみで使用することをお勧めします。実稼働環境で使用するサーバーは、CA により交付された SSL 証明書を使用してください。

IMAP や **POP** に対し自己署名 SSL 証明書を作成するには、**/etc/pki/dovecot/** ディレクトリーに移動し、**/etc/pki/dovecot/dovecot-openssl.cnf** 設定ファイルの証明書パラメータを希望に応じて編集し、**root** で次のコマンドを入力します。

```
dovecot]# rm -f certs/dovecot.pem private/dovecot.pem
dovecot]# /usr/libexec/dovecot/mkcert.sh
```

終了したら、**/etc/dovecot/conf.d/10-ssl.conf** ファイルに次の設定ファイルがあることを確認してください。

```
ssl_cert = </etc/pki/dovecot/certs/dovecot.pem
ssl_key = </etc/pki/dovecot/private/dovecot.pem
```

以下のコマンドを実行して、**dovecot** デーモンを再起動します。

```
~]# systemctl restart dovecot
```

別の方法として、**stunnel** コマンドを **IMAP** または **POP** サービスへの標準的なセキュアでない接続の周りに SSL 暗号化ラッパーとして使用することも可能です。

stunnel ユーティリティーは、Red Hat Enterprise Linux に装備されている外部の OpenSSL ライブラリを使用して、強力な暗号化を実現し、ネットワーク接続を保護します。SSL 証明書を取得するためには、CA に申請することが推奨されますが、自己署名証明書を作成することも可能です。

注記

stunnel を使用するには、**root** で以下を実行して、最初にご使用のシステムに*stunnel* パッケージがインストールされていることを確認します。

```
~]# yum install stunnel
```

yum を使ったパッケージのインストールについては 「パッケージのインストール」を参照してください。

自己署名 SSL 証明書を作成するには、**/etc/pki/tls/certs/** ディレクトリーに移動して、以下のコマンドを入力します。

```
certs]# make stunnel.pem
```

すべての質問に回答して、プロセスを完了します。

証明書が生成されたら、以下のコンテンツを含む **/etc/stunnel/mail.conf** のような **stunnel** 設定ファイルを作成します。

```
cert = /etc/pki/tls/certs/stunnel.pem

[pop3s]
accept  = 995
connect = 110

[imaps]
accept  = 993
connect = 143
```

stunnel /etc/stunnel/mail.conf コマンドを使用して作成済みの設定ファイルで**stunnel** を起動したら、**IMAP** または **POP** 電子メールクライアントを使用し、SSL 暗号化によって電子メールサーバーに接続することが可能となります。

stunnel の詳細は **stunnel** の man ページ、または *<version-number>* が **stunnel** のバージョン番号である **/usr/share/doc/stunnel-*<version-number>*** ディレクトリーのドキュメントを参照してください。

10.6. その他のリソース

以下は、電子メールアプリケーションに関する補足のドキュメントの一覧です。

10.6.1. インストールされているドキュメント

- Sendmail の設定に関する情報は、**sendmail** および **sendmail-cf** パッケージに含まれています。
  - **/usr/share/sendmail-cf/README** — **m4** マクロプロセッサーの情報、Sendmail のファイルの場所、対応するメーラー、強化機能へのアクセス方法などの情報が記載されています。

  さらに、**sendmail** および **aliases** の man ページには役立つ情報が記載されており、Sendmail の様々なオプションや、Sendmail **/etc/mail/aliases** ファイルの適切な設定について説明しています。
- **/usr/share/doc/postfix-*<version-number>*** — Postfix の設定方法に関する多くの情報が含まれています。*<version-number>* を Postfix のバージョン番号に置き換えてください。
- **/usr/share/doc/fetchmail-*<version-number>*** — Fetchmail の機能の全一覧を記載した **FEATURES** ファイルと初歩的な **FAQ** ドキュメントが含まれています。*<version-number>* を Fetchmail のバージョン番号に置き換えてください。
- **/usr/share/doc/procmail-*<version-number>*** — Procmail の概要を記載した **README** ファイル、各プログラム機能を詳細に記した **FEATURES** ファイル、設定に関する多数のよくある質問に対する回答をまとめた **FAQ** が含まれています。*<version-number>* を Procmail のバージョン番号に置き換えてください。

  Procmail の仕組みや新しいレシピの作成方法を学習する場合は、以下にあげる Procmail の man ページが非常に役立ちます。
  - **procmail** — Procmail の仕組みと電子メールのフィルタリングに必要な手順について概説しています。
  - **procmailrc** — レシピの構築に使用される **rc** のファイル形式について説明しています。
  - **procmailex** — 実環境向けの役立つ Procmail レシピの例を多数記載しています。
  - **procmailsc** — 特定のレシピとメッセージを適合するために Procmail で使用される加重スコアリング手法について説明しています。
  - **/usr/share/doc/spamassassin-*<version-number>*/** — SpamAssassin に関する多くの情報が含まれています。*<version-number>* を **spamassassin** パッケージのバージョン番号に置き換えてください。

10.6.2. 役立つ Web サイト

- http://www.sendmail.org/ — Sendmail の機能に関する完全な技術詳細、ドキュメント、設定例が記載されています。
- http://www.sendmail.com/ — Sendmail に関連したニュース、インタビュー、記事が掲載されており、利用可能な数多くのオプションの幅広い詳細が含まれています。
- http://www.postfix.org/ — Postfix プロジェクトのホームページで、Postfix に関する豊富な情報が掲載されています。メーリングリストは、特に情報検索に役立ちます。
- http://fetchmail.berlios.de/ — Fetchmail のホームページです。オンラインマニュアルと詳細な FAQ を参照できます。
- http://www.procmail.org/ — Procmail のホームページです。Procmail に特化した各種メーリングリストへのリンクや、様々な FAQ ドキュメントが記載されています。

- http://www.uwasa.fi/~ts/info/proctips.html — Procmail をさらに使い易くするための数々のヒントが掲載されています。`.procmailrc` ファイルのテスト方法や、Procmail スコアリングを使用して特定のアクションを実行すべきかどうかを判断する手順の説明が含まれています。
- http://www.spamassassin.org/ — SpamAssassin プロジェクトの公式サイトです。

10.6.3. 関連資料

- 『Sendmail Milters: A Guide for Fighting Spam』 (Bryan Costales、Marcia Flynt 共著、Addison-Wesley 社出版) — Sendmail の優れたガイドで、メールフィルターのカスタマイズに役立ちます。
- 『Sendmail』 (Bryan Costales、Eric Allman 他共著、O'Reilly & Associates 社出版) — Delivermail と Sendmail の考案者のサポートを受け執筆された優れた Sendmail の参考書籍です。
- 『Removing the Spam: Email Processing and Filtering』 (Geoff Mulligan 著、Addison-Wesley 社出版) — Sendmail や Procmail などの確立されたツールを用いてスパム問題に対処する電子メール管理者が使用する様々な手法を考察した書籍です。
- 『Internet Email Protocols: A Developer's Guide』 (Kevin Johnson 著、Addison-Wesley 社出版) — 主要な電メールプロトコルとそれらによるセキュリティーに関する非常に詳細なレビューをまとめています。
- 『Managing IMAP』 (Dianna Mullet、Kevin Mullet 共著、O'Reilly & Associates 社出版) — IMAP サーバーの設定に必要な手順が詳述されています。

第11章 ディレクトリーサーバー

11.1. OpenLDAP

LDAP (Lightweight Directory Access Protocol) は、ネットワーク上で中央に保存されている情報へアクセスするために使用するオープンプロトコルのセットです。これは、ディレクトリー共有のために **X.500** 標準をベースとしていますが、それほど複雑ではなくリソース集約型です。このため LDAP は 「X.500 Lite」と呼ばれることがあります。

X.500 のように、LDAP はディレクトリーを使用して階層式で情報を組織化します。ディレクトリーには、名前やアドレス、電話番号などの各種情報を保存することができ、また *ネットワーク情報サービス* (NIS) と似た方法で使用することもできます。このため、誰でも LDAP 対応ネットワーク上にあるマシンから自分のアカウントにアクセスできます。

LDAP の一般的な使用目的は、中央で管理されるユーザーとグループ、ユーザー認証、システム設定です。また、仮想の電話帳としても使用できるため、ユーザーは他のユーザーの連絡先情報に容易にアクセスできます。さらに、LDAP は世界各地にある他の LDAP サーバーにユーザーを照会して、グローバルレポジトリーに保存されているアドホックな情報を提供することが可能です。ただし、LDAP は、大学や政府省庁、民間企業など個々の組織内で最もよく使用されます。

本セクションでは、LDAPv2 および LDAPv3 プロトコルのオープンソース実装である **OpenLDAP 2.4** をインストール、設定する方法について取り上げます。

11.1.1. LDAP の概要

LDAP は、クライアント/サーバーのアーキテクチャーを使用して、ネットワークからアクセス可能な中央情報ディレクトリーを作成する、信頼できる手段を提供します。クライアントがこのディレクトリー内の情報を修正しようとすると、サーバーはこのユーザーが修正できる権限を持っていることを確認します。その後、要求された通りにエントリーを追加、更新します。通信の安全性を確保するために、*Secure Sockets Layer* (SSL) または *Transport Layer Security* (TLS) 暗号化プロトコルを使用して攻撃者が送信を傍受できないようにします。

重要

Red Hat Enterprise Linux 7 の OpenLDAP スイートは OpenSSL を使用しなくなりました。代わりに *ネットワークセキュリティーサービス* (NSS) の Mozilla 実装を使用します。OpenLDAP は現行の証明書、鍵、その他の TLS 設定による機能を継続します。Mozilla 証明書と鍵データベースを使用するための Mozilla NSS の設定方法については、『How do I use TLS/SSL with Mozilla NSS (Mozilla NSS で TLS/SSL を使用する方法)』 を参照してください。

LDAP サーバーは複数のデータベースシステムをサポートするため、管理者は適用する予定の情報のタイプに最適なソリューションを柔軟に選択できます。明確に定義された *アプリケーションプログラミングインターフェース* (API) クライアントにより、LDAP サーバーと通信できるアプリケーションは多く、その数と質ともに向上しています。

11.1.1.1. LDAP の用語

以下は、本章で使用されている LDAP 特有の用語一覧です。

エントリー

LDAP ディレクトリー内の単一ユニットです。各エントリーは一意の DN *(Distinguished Name: 識別名)* で識別されます。

属性

エントリーに直接関連付けられた情報です。たとえば、ある組織が LDAP エントリーとして表示されている場合、アドレス、ファックス番号などがこの組織に関連付けられた属性です。同様に、人も電話番号や電子メールアドレスのような共通の属性を持つエントリーとして表示することができます。

属性には、単一値か順不同の空白で区切られた値の一覧のどちらかがあります。一部の属性はオプションですが、その他は必須です。必須の属性は **`objectClass`** 定義を使用して指定され、**`/etc/openldap/slapd.d/cn=config/cn=schema/`** ディレクトリー内のスキーマファイル内にあります。

属性とそれに対応する値のアサーションは、RDN *(Relative Distinguished Name: 相対識別名)* とも呼ばれます。グローバルで一意の識別名とは異なり、相対識別名は エントリーに対してのみ一意なものです。

LDIF

LDAP データ交換形式 (LDIF) は LDAP エントリーをプレーンテキスト形式で表示したものです。以下のような形式を取ります。

```
[id] dn: distinguished_name
attribute_type: attribute_value…
attribute_type: attribute_value…
…
```

オプションの *id* は、エントリーの編集に使用されるアプリケーションが決定する番号です。*attribute_type* と *attribute_value* のペアが該当するスキーマファイルですべて定義されている限り、各エントリーはそれらを必要な数だけ含めることができます。空白の行はエントリーの終了を意味します。

11.1.1.2. OpenLDAP の機能

OpenLDAP スイートは多くの重要な機能を提供します。

- **LDAPv3 サポート** — LDAP バージョン 2 以降に行われたプロトコルへの変更の多くは、LDAP をよりセキュアにするように設計されています。その他の改善点としては、SASL (Simple Authentication and Security Layer)、TLS および SSL プロトコルに対するサポートがあります。
- **IPC 上の LDAP** — プロセス間通信 (IPC) の使用により、ネットワーク上で通信する必要性をなくすことでセキュリティーを強化します。
- **IPv6 サポート** — OpenLDAP は、次世代のインターネットプロトコルである Internet Protocol version 6 (IPv6) に準拠しています。
- **LDIFv1 サポート** — OpenLDAP は LDIF バージョン 1 に完全に準拠しています。
- **更新された C API** — 最新の C API はプログラマーが LDAP ディレクトリーサーバーに接続して使用する方法を改善します。
- **スタンドアロン LDAP サーバーの機能強化**— これには更新されたアクセス制御システム、スレッドプーリング、改善されたツール、その他多くが含まれます。

11.1.1.3. OpenLDAP サーバーの設定

Red Hat Enterprise Linux における LDAP サーバーの標準的な設定手順は、以下のとおりです。

1. Red Hat Enterprise Linux での新規パッケージのインストールに関する詳しい情報は、「パッケージのインストール」を参照してください。
2. 「OpenLDAP サーバーの設定」で説明のとおりに設定をカスタマイズします。
3. 「OpenLDAP サーバーの実行」で説明のとおりに **slapd** サービスを開始します。
4. **ldapadd** ユーティリティーを使用して、エントリーを LDAP ディレクトリーに追加します。
5. **ldapsearch** ユーティリティーを使用して、**slapd** サービスが情報に正しくアクセスしていることを確認します。

11.1.2. OpenLDAP スイートのインストール

OpenLDAP ライブラリとツールのスイートは、以下のパッケージにより提供されます。

表11.1 OpenLDAP パッケージの一覧

パッケージ	詳細
openldap	OpenLDAP サーバー/クライアントアプリケーションを実行するために必要なライブラリを含むパッケージです。
openldap-clients	LDAP サーバー上のディレクトリーを表示、修正するためのコマンドラインユーティリティーを含むパッケージです。
openldap-servers	LDAP サーバーを設定して実行するためのサービスとユーティリティーの両方を含むパッケージです。これには、*スタンドアロン LDAP デーモン* である **slapd** が含まれます。
compat-openldap	OpenLDAP 互換ライブラリを含むパッケージです。

さらに、LDAP サーバーと共によく使用されるパッケージは以下のとおりです。

表11.2 一般的にインストールされるその他の LDAP パッケージの一覧

パッケージ	詳細
nss-pam-ldapd	ユーザーがローカルの LDAP クエリを実行できるようにするローカル LDAP ネームサービスである、**nslcd** を含むパッケージです。

これらのパッケージをインストールするには、以下の形式で **yum** コマンドを使用します。

```
yum install package…
```

たとえば、基本的な LDAP サーバーをインストールするには、シェルプロンプトで以下を入力します。

```
~]# yum install openldap openldap-clients openldap-servers
```

このコマンドを実行するには、スーパーユーザー権限を持っている (すなわち **root** としてログインしている) 必要があります。Red Hat Enterprise Linux に新しいパッケージをインストールする方法の詳細については「パッケージのインストール」を参照してください。

11.1.2.1. OpenLDAP サーバーユーティリティーの概要

管理タスクを実行するために、*openldap-servers* パッケージは **slapd** サービスと共に以下のユーティリティーをインストールします。

表11.3 OpenLDAP サーバーユーティリティーの一覧

コマンド	詳細
slapacl	属性の一覧へのアクセスをチェックできるようにします。
slapadd	LDIF ファイルから LDAP ディレクトリーへエントリーを追加できるようにします。
slapauth	認証および承認のパーミッション用に ID の一覧をチェックできるようにします。
slapcat	LDAP ディレクトリーからデフォルト形式でエントリーをプルして、LDIF ファイル内に保存できるようにします。
slapdn	利用可能なスキーマ構文をベースとした識別名 (DN) の一覧をチェックできるようにします。
slapindex	現在の内容を基にして **slapd** ディレクトリーのインデックスを再構築できるようにします。設定ファイル内のインデックスオプションを変更する時は常にこのユーティリティーを実行します。
slappasswd	**ldapmodify** ユーティリティーと共に、あるいは **slapd** 設定ファイル内で使用される暗号化されたユーザーパスワードを作成できるようにします。
slapschema	データベースの整合性を対応するスキーマとチェックできるようにします。
slaptest	LDAP サーバーの設定をチェックできるようにします。

これらのユーティリティーとその使用方法の詳しい説明については、「インストールされているドキュメント」にある各 man ページを参照してください。

重要

slapadd が実行可能なのは **root** のみですが、**slapd** サービスは **ldap** ユーザーとして実行します。このため、ディレクトリーサーバーは **slapadd** によって作成されたファイルは修正することができません。この問題を解決するには、**slapd** ユーティリティーの実行後に、シェルプロンプトで以下を入力します。

```
~]# chown -R ldap:ldap /var/lib/ldap
```

警告

データの整合性を維持するために、**slapadd**、**slapcat**、**slapindex** を使用する前に **slapd** サービスを停止します。シェルプロンプトで以下を入力します。

```
~]# systemctl stop slapd.service
```

slapd サービスの起動、停止、再起動および現在のステータスの確認を行う方法の詳細については「OpenLDAP サーバーの実行」を参照してください。

11.1.2.2. OpenLDAP クライアントユーティリティーの概要

openldap-clients パッケージは、LDAP ディレクトリー内のエントリーを追加、修正、削除するために使用可能な以下のユーティリティーをインストールします。

表11.4 OpenLDAP クライアントユーティリティーの一覧

表11.4 OpenLDAP クライアントユーティリティーの一覧

コマンド	詳細
ldapadd	ファイルまたは標準入力からエントリーを LDAP ディレクトリーに追加できるようにします。これは **ldapmodify -a** へのシンボリックリンクです。
ldapcompare	任意の属性と LDAP ディレクトリーのエントリーを比較できるようにします。
ldapdelete	LDAP ディレクトリーからエントリーを削除できるようにします。
ldapexop	LDAP の拡張操作を実行できるようにします。
ldapmodify	LDAP ディレクトリー内のエントリーをファイルまたは標準入力から修正できるようにします。
ldapmodrdn	LDAP ディレクトリーエントリーの RDN 値を修正できるようにします。
ldappasswd	LDAP ユーザー用のパスワードを設定、変更できるようにします。
ldapsearch	LDAP ディレクトリーエントリーを検索できるようにします。
ldapurl	LDAP URL を生成または分解できるようにします。
ldapwhoami	LDAP サーバー上で **whoami** 動作を実行できるようにします。

ldapsearch を除いて、他のユーティリティーをより簡単に使用するためには、LDAP ディレクトリー内で変更される各エントリー用にコマンドを入力する代わりに、行われる変更を含むファイルを参照します。このようなファイルの形式は、各ユーティリティーの man ページに要約されています。

11.1.2.3. 一般的な LDAP クライアントアプリケーションの概要

サーバー上のディレクトリーを作成、修正できるグラフィカルな LDAP クライアントは各種存在しますが、どれも Red Hat Enterprise Linux には含まれていません。読み取り専用モードでディレクトリーにアクセスできる人気のアプリケーションとしては、**Mozilla Thunderbird**、**Evolution**、**Ekiga** があります。

11.1.3. OpenLDAP サーバーの設定

デフォルトでは、OpenLDAP 設定は **/etc/openldap/** ディレクトリーに格納されています。以下の表は、このディレクトリー内の最も重要なディレクトリーとファイルを示しています。

表11.5 OpenLDAP 設定ファイルとディレクトリーの一覧

パス	詳細
/etc/openldap/ldap.conf	OpenLDAP ライブラリを使用するクライアントアプリケーション用の設定ファイルです。これには **ldapadd**、**ldapsearch**、**Evolution** などが含まれます。
/etc/openldap/slapd.d/	**slapd** 設定を含むディレクトリーです。

OpenLDAP は設定を **/etc/openldap/slapd.conf** ファイルから読み取らなくなった点に注意してください。代わりに、**/etc/openldap/slapd.d/** ディレクトリーに存在する設定データベースを使用します。以前のインストールから **slapd.conf** ファイルが残っている場合は、以下のコマンドを実行することで新しい形式に変換できます。

```
~]# slaptest -f /etc/openldap/slapd.conf -F /etc/openldap/slapd.d/
```

slapd 設定は、階層ディレクトリー構造で編成されている LDIF エントリーで構成されています。これらのエントリーを編集するための推奨される方法は 「OpenLDAP サーバーユーティリティーの概要」 にあるようにサーバーユーティリティーを使用することです。

重要

LDIF ファイル内でエラーが発生すると **slapd** サービスが起動できなくなることがあります。このため、**/etc/openldap/slapd.d/** 内の LDIF ファイルを直接編集しないことが強く推奨されます。

11.1.3.1. グローバル設定の変更

LDAP サーバーのグローバル設定オプションは、**/etc/openldap/slapd.d/cn=config.ldif** ファイル内に格納されています。一般的に使用される指示文は、以下のとおりです。

olcAllows

olcAllows 指示文を使用すると、有効にする機能を指定できます。以下の形式を取ります。

```
olcAllows: feature…
```

これは、表11.6「利用可能な olcAllows のオプション」で説明のとおり空白で区切られた機能の一覧を使用します。デフォルトオプションは **bind_v2** です。

表11.6 利用可能な olcAllows のオプション

オプション	詳細
bind_v2	LDAP バージョン 2 のバインド要求の受け入れを有効にします。
bind_anon_cred	識別名(DN) が空欄である場合の匿名バインドを有効にします。
bind_anon_dn	識別名(DN) が空欄で **ない** 場合の匿名バインドを有効にします。
update_anon	匿名による更新操作の処理を有効にします。
proxy_authz_anon	匿名によるプロキシ認証制御の処理を有効にします。

例11.1 olcAllows 指示文の使用

```
olcAllows: bind_v2 update_anon
```

olcConnMaxPending

olcConnMaxPending 指示文を使用すると、匿名セッションに対する保留中の要求の最大数を指定できます。以下の形式を取ります。

```
olcConnMaxPending: number
```

デフォルトオプションは **100** です。

例11.2 olcConnMaxPending 指示文の使用

```
olcConnMaxPending: 100
```

olcConnMaxPendingAuth

olcConnMaxPendingAuth 指示文を使用すると、認証済みセッションに対する保留中の要求の最大数を指定できます。以下の形式を取ります。

```
olcConnMaxPendingAuth: number
```

デフォルトオプションは **1000** です。

例11.3 olcConnMaxPendingAuth 指示文の使用

```
olcConnMaxPendingAuth: 1000
```

olcDisallows

olcDisallows を使用すると、無効にする機能を指定できます。以下の形式を取ります。

```
olcDisallows: feature…
```

これは、表11.7「利用可能な olcDisallows のオプション」で説明のとおり空白で区切られた機能の一覧を使用します。デフォルトでは、どの機能も無効になっていません。

表11.7 利用可能な olcDisallows のオプション

オプション	詳細
bind_anon	匿名バインド要求の受け入れを無効にします。
bind_simple	簡易バインド認証のメカニズムを無効にします。
tls_2_anon	STARTTLS コマンドの受け取り時に、匿名セッションの強制を無効にします。
tls_authc	認証時に、STARTTLS コマンドを許可しません。

例11.4 olcDisallows 指示文の使用

```
olcDisallows: bind_anon
```

olcIdleTimeout

olcIdleTimeout 指示文を使用すると、アイドル状態の接続を閉じるまでの待機秒数を指定できます。以下の形式を取ります。

```
olcIdleTimeout: number
```

このオプションは、デフォルトで無効です (すなわち **0** に設定されています)。

例11.5 olcIdleTimeout 指示文の使用

```
olcIdleTimeout: 180
```

olcLogFile

olcLogFile 指示文を使用すると、ログメッセージを書き込むファイルを指定できます。以下の形式を取ります。

```
olcLogFile: file_name
```

ログメッセージは、デフォルトで標準エラーに書き込まれます。

例11.6 olcLogFile 指示文の使用

```
olcLogFile: /var/log/slapd.log
```

olcReferral

olcReferral オプションを使用すると、サーバーが要求を処理できない場合に処理する別のサーバーの URL を指定できます。以下の形式を取ります。

```
olcReferral: URL
```

このオプションは、デフォルトで無効です。

例11.7 olcReferral 指示文の使用

```
olcReferral: ldap://root.openldap.org
```

olcWriteTimeout

olcWriteTimeout オプションを使用すると、未処理の書き込み要求がある接続を閉じるまでの待機秒数を指定できます。以下の形式を取ります。

```
olcWriteTimeout
```

このオプションは、デフォルトで無効です (すなわち **0** に設定されています)。

例11.8 olcWriteTimeout 指示文の使用

```
olcWriteTimeout: 180
```

11.1.3.2. データベース特有の設定の変更

デフォルトでは、OpenLDAP サーバーは Berkeley DB (BDB) をデータベースバックエンドとして使用します。このデータベース用の設定は **/etc/openldap/slapd.d/cn=config/olcDatabase={1}bdb.ldif** ファイルに格納されています。データベース特有の設定でよく使用される指示文は、以下のとおりです。

olcReadOnly

olcReadOnly 指示文を使用すると、データベースを読み取り専用モードで使用できます。以下の形式を取ります。

```
olcReadOnly: boolean
```

これは、**TRUE** (読み取り専用モードを有効) または**FALSE** (データベースの修正を有効) を使用します。デフォルトのオプションは **FALSE** です。

例11.9 olcReadOnly指示文の使用

```
olcReadOnly: TRUE
```

olcRootDN

olcRootDN 指示文を使用すると、アクセス制御により制限されないユーザーまたは LDAP ディレクトリの動作用に設定される管理制限パラメーターを指定できます。以下の形式を取ります。

```
olcRootDN: distinguished_name
```

これは、*識別名* (DN) を使用します。デフォルトのオプションは**cn=Manager,dn=my-domain,dc=com** です。

例11.10 olcRootDN 指示文の使用

```
olcRootDN: cn=root,dn=example,dn=com
```

olcRootPW

olcRootPW 指示文を使用すると、**olcRootDN** 指示文を使って指定されたユーザー用のパスワードを設定できます。以下の形式を取ります。

```
olcRootPW: password
```

これは、プレーンテキスト文字列かハッシュを使用します。ハッシュを生成するためには、シェルプロンプトで以下を入力します。

```
~]$ slappaswd
New password:
Re-enter new password:
{SSHA}WczWsyPEnMchFf1GRTweq2q7XJcvmSxD
```

例11.11 olcRootPW指示文の使用

```
olcRootPW: {SSHA}WczWsyPEnMchFf1GRTweq2q7XJcvmSxD
```

olcSuffix

olcSuffix 指示文を使用すると、情報を提供するドメインを指定できます。以下の形式を取ります。

```
olcSuffix: domain_name
```

これは、*完全修飾ドメイン名* (FQDN) を使用します。デフォルトのオプションは**dc=my-domain,dc=com** です。

例11.12 olcSuffix 指示文の使用

```
olcSuffix: dc=example,dc=com
```

11.1.3.3. スキーマの拡張

OpenLDAP 2.3 以降、**/etc/openldap/slapd.d/** ディレクトリーには、以前 **/etc/openldap/schema/** に配置されていた LDAP 定義も含まれています。デフォルトのスキーマファイルをガイドとして使用して追加の属性タイプとオブジェクトクラスをサポートするために、OpenLDAP により使用されるスキーマを拡張することができます。ただし、このタスクは本章の範囲外です。このトピックの詳細については http://www.openldap.org/doc/admin/schema.html を参照してください。

11.1.3.4. セキュアな接続の確立

OpenLDAP クライアントとサーバーは、Transport Layer Security (TLS) フレームワークを使用することでセキュリティー保護ができます。TLS は、ネットワーク上の通信セキュリティーを提供するように設計された暗号化プロトコルです。上記にあるように、Red Hat Enterprise Linux 7 の OpenLDAP スイーツは、Mozilla NSS を TLS 実装として使用します。

TLS を使用して安全な接続を確立するには、『How do I use TLS/SSL with Mozilla NSS』 にあるように必要な証明を獲得します。そして、クライアントとサーバーの両方で多くのオプションを設定する必要があります。少なくとも、サーバーは認証局 (CA) の証明書とそれ自体のサーバー認証および秘密鍵で設定する必要があります。クライアントは、信頼できる CA 証明書すべてを含むファイル名で設定する必要があります。

通常サーバーは、1 つの CA 証明書にのみ署名する必要があります。クライアントはいくつもの安全なサーバーに接続する可能性があるため、一般的には設定で信頼できるいくつかの CA 一覧を指定します。

サーバー設定

TLS を確立するためには、以下の **slapd** のグローバル設定指示文を OpenLDAP サーバー上の **/etc/openldap/slapd.d/** ディレクトリーで指定する必要があります。

olcTLSCACertificateFile

olcTLSCACertificateFile 指示文は、信頼できる CA 証明書を含む PEM (Privacy-Enhanced Mail) スキーマでエンコードされたファイルを指定します。指示文の形式は以下のようになります。

```
olcTLSCACertificateFile: path
```

path を CA 証明書ファイルへのパスに置き換えます。

olcTLSCACertificatePath

olcTLSCACertificatePath 指示文は、個別ファイルの CA 証明書を含むディレクトリーへのパスを指定します。このディレクトリーは、OpenSSL **c_rehash** ユーティリティーで特別に管理する必要があり、このユーティリティーは実際の証明書ファイルを指すハッシュ化された名前のシンボリックリンクを生成します。一般的に、**olcTLSCACertificateFile** 指示文の使用の方がより簡易な方法になります。指示文は、以下の形式になります。

```
olcTLSCACertificatePath: path
```

path を CA 証明書ファイルを含むディレクトリーへのパスに置き換えます。

例11.13 olcTLSCACertificatePath 指示文の使用

Mozilla NSS では、**olcTLSCACertificatePath** 指示文は NSS 証明書および鍵のデータベースファイルを含むディレクトリーへのパスを指定します。

```
olcTLSCACertificatePath: sql:/home/nssdb/sharednssdb
```

certutil コマンドは、CA 証明書をこれらの NSS データベースファイルに追加するために使用されます。

```
certutil -d sql:/home/nssdb/sharednssdb -A -n "CA_certificate"
-t CT,, -a -i certificate.pem
```

上記のコマンドは、*certificate.pem* という名前の PEM 形式のファイル に保存されている CA 証明書を追加します。**-d** オプションは証明書と鍵のデータベースファイルを含むデータベースディレクトリーを指定し、**-n** オプションは証明書の名前を設定します。**-t CT,,** は、証明書が TLS クライアントおよびサーバーでの使用に信頼できることを意味します。**-A** オプションは既存の証明書を既存のデータベースに追加し、**-a** オプションは出入力での ASCII 形式の使用を可能にします。**-i** オプションは、**certificate.pem** 入力ファイルをコマンドに渡します。

olcTLSCertificateFile

olcTLSCertificateFile 指示文は、**slapd** サーバー証明書を含むファイルを指定します。指示文の形式は以下のようになります。

```
olcTLSCertificateFile: path
```

path を **slapd** サーバー証明書ファイルへのパスに置き換えます。

例11.14 olcTLSCertificateFile 指示文の使用

Mozilla NSS を **olcTLSCACertificatePath** 指示文で証明書および鍵データベースファイルを指定して使用する場合は、使用する証明書の名前の指定に **olcTLSCertificateFile** が使われます。まず、以下のコマンドを実行して、使用中の NSS データベースファイルで利用可能な証明書のリストを表示します。

```
certutil -d sql:/home/nssdb/sharednssdb -L
```

一覧から証明書を選択し、その名前を **olcTLSCertificateFile** に渡します。例を示します。

```
olcTLSCertificateFile slapd_cert
```

olcTLSCertificateKeyFile

olcTLSCertificateKeyFile 指示文は、**olcTLSCertificateFile** で指定されたファイルに保存されている証明書に適合する秘密鍵を含むファイルを指定します。現在の実装では秘密鍵の暗号化はサポートされていないことから、この鍵を含むファイルは十分に保護される必要があることに注意してください。指示文は以下の形式になります。

```
olcTLSCertificateKeyFile: path
```

path を秘密鍵ファイルへのパスに置き換えます。

例11.15 olcTLSCertificateKeyFile 指示文の使用

Mozilla NSS を使用する場合、この指示文は**olcTLSCertificateFile** で指定された証明書の鍵のパスワードを含むファイル名を指定します。

```
olcTLSCertificateKeyFile: slapd_cert_key
```

modutil コマンドを使うと、パスワード保護をオフにしたり、NSS データベースファイルのパスワードを変更したりすることができます。例を示します。

```
modutil -dbdir sql:/home/nssdb/sharednssdb -changepw
```

クライアント設定

以下の指示文を、クライアントシステム上の **/etc/openldap/ldap.conf** 設定ファイルで指定します。これらの指示文のほとんどは、サーバー設定オプションと並行するものです。**/etc/openldap/ldap.conf** 内の指示文は、システム全体ベースで設定されていますが、個別のユーザーは **~/.ldaprc** ファイルでこれを上書きすることができます。

TLS_CACERT

TLS_CACERT 指示文は、クライアントが認識するすべての CA 証明書を格納しているファイルを指定します。これは、サーバー上の **olcTLSCACertificateFile** 指示文と同等です。**TLS_CACERT** は、常に **/etc/openldap/ldap.conf** 内の **TLS_CACERTDIR** の前に指定されるべきです。指示文の形式は以下のようになります。

```
TLS_CACERT path
```

path を CA 証明書ファイルへのパスに置き換えます。

TLS_CACERTDIR

TLS_CACERTDIR 指示文は、個別ファイルにある CA 証明書を含むディレクトリーへのパスを指定します。サーバー上の **olcTLSCACertificatePath** と同様に、指定されたディレクトリーは OpenSSL **c_rehash** ユーティリティーで管理する必要があります。指示文の形式は以下のようになります。

```
TLS_CACERTDIR directory
```

directory を CA 証明書ファイルを含むディレクトリーへのパスで置き換えます。Mozilla NSS では、*directory* は証明書もしくは鍵データベースファイルを指します。

TLS_CERT

TLS_CERT は、クライアントの証明書を格納しているファイルを指定します。この指示文の指定ができるのは、ユーザーの **~/.ldaprc** ファイル内のみです。指示文の形式は以下のようになります。

```
TLS_CERT path
```

path をクライアント証明書ファイルへのパスに置き換えます。

TLS_KEY

TLS_KEY は、**TLS_CERT** 指示文で指定されたファイルに保存されている証明書に適合する秘密鍵を格納するファイルを指定します。サーバー上の **olcTLSCertificateFile** と同様に鍵ファイルの暗号化はサポートされないため、ファイル自体は慎重に保護される必要があります。このオプションが設定できるのは、ユーザーの **~/.ldaprc** ファイル内のみです。

```
TLS_KEY path
```

path をクライアント証明書ファイルへのパスに置き換えます。

11.1.3.5. レプリケーションの設定

レプリケーションは、ある LDAP サーバー (**プロバイダー**) から別の 1 つ以上のサーバーもしくはクライアント (**コンシューマー**) に更新をコピーするプロセスです。プロバイダーはディレクトリー更新をコンシューマーに複製し、この受け取られた更新はさらにコンシューマーから他のサーバーに伝達されます。このため、コンシューマーは同時にプロバイダーとしての動作も行います。また、コンシューマーは LDAP である必要はなく、単なる LDAP クライアントでも構いません。OpenLDAP では、いくつかの複製モードが使用可能で、なかでも目立つのは **mirror** と **sync** です。OpenLDAP 複製モードに関する詳細情報は、*openldap-servers* パッケージとインストールされる**OpenLDAP Software Administrator's Guide** を参照してください (「インストールされているドキュメント」を参照)。

選択した複製モードを有効にするには、プロバイダーとコンシューマーの両方の **/etc/openldap/slapd.d/** で、以下の指示文のいずれかを使用します。

olcMirrorMode

olcMirrorMode 指示文は、mirror 複製モードを有効にします。以下の形式になります。

```
olcMirrorMode on
```

このオプションは、プロバイダーとコンシューマーの両方で指定する必要があります。また、**serverID** を **syncrepl** オプションで指定する必要もあります。**OpenLDAP Software Administrator's Guide** の **18.3.4. MirrorMode** セクションにある詳細例を参照してください (「インストールされているドキュメント」参照)。

olcSyncrepl

olcSyncrepl 指示文は、sync 複製モードを有効にします。以下の形式になります。

```
olcSyncrepl on
```

sync 複製モードは、プロバイダーとコンシューマーの両方で特定の設定が必要になります。この設定は、**OpenLDAP Software Administrator's Guide** の **18.3.1. Syncrepl** セクションで詳しく説明されています (「インストールされているドキュメント」参照)。

11.1.3.6. モジュールとバックエンドの読み込み

slapd サービスは、動的に読み込まれるモジュールで強化することができます。これらのモジュールのサポートは、**slapd** の設定時に **--enable-modules** オプションで有効にする必要があります。モジュールは、ファイルに **.la** 拡張子を付けて保存します。

```
module_name.la
```

*バックエンド*は、LDAP リクエストに対応してデータを保存したり取得したりします。バックエンドは **slapd** に静的にコンパイルするか、モジュールサポートが有効な場合は動的に読み込むことができます。後者の場合、以下の命名慣習が適用されます。

```
back_backend_name.la
```

モジュールもしくはバックエンドを読み込むには、以下の指示文を **/etc/openldap/slapd.d/** で使用します。

olcModuleLoad

olcModuleLoad 指示文は、動的に読み込まれるモジュールを指定します。以下の形式になります。

```
olcModuleLoad: module
```

ここでの *module* は、読み込まれる予定のモジュールを格納しているファイル、もしくはバックエンドを表します。

11.1.4. OpenLDAP サーバーの実行

本セクションでは、**Standalone LDAP Daemon** の起動、停止、再起動、現行ステータスの確認方法を説明しています。システムサービス全般の詳細情報については、*6章systemd によるサービス管理* を参照してください。

11.1.4.1. サービスの開始

現行のセッションで **slapd** サービスを起動するには、シェルプロンプトで**root** として以下を入力します。

```
~]# systemctl start slapd.service
```

サービスが起動時に自動的に開始するように設定するには、**root** で以下のコマンドを実行します。

```
~]# systemctl enable slapd.service
ln -s '/usr/lib/systemd/system/slapd.service' '/etc/systemd/system/multi-
user.target.wants/slapd.service'
```

11.1.4.2. サービスの停止

現行のセッションで実行中の **slapd** サービスを停止するには、シェルプロンプトで**root** として以下を入力します。

```
~]# systemctl stop slapd.service
```

サービスが起動時に自動的に起動しないようにするには、**root** で以下を入力します。

```
~]# systemctl disable slapd.service
rm '/etc/systemd/system/multi-user.target.wants/slapd.service'
```

11.1.4.3. サービスの再起動

実行中の **slapd** サービスを再起動するには、シェルプロンプトで以下を入力します。

```
~]# systemctl restart slapd.service
```

これでサービスが停止し、即座に再起動します。設定を再読み込みするには、このコマンドを使用します。

11.1.4.4. サービスステータスの確認

slapd サービスが実行中かどうかを確認するには、シェルプロンプトで以下を入力します。

```
~]$ systemctl is-active slapd.service
active
```

11.1.5. OpenLDAP を使用した認証用のシステムの設定

OpenLDAP を使用して認証するようシステムを設定するためには、LDAP サーバーとクライアントマシンの両方に適切なパッケージがインストールされていることを確認します。サーバーの設定方法に関する詳細は、「OpenLDAP スイートのインストール」と「OpenLDAP サーバーの設定」の手順にしたがいます。クライアントで、以下をシェルプロンプトで入力します。

```
~]# yum install openldap openldap-clients nss-pam-ldapd
```

LDAP 認証を使用するためのアプリケーションの詳細な設定方法については、『Red Hat Enterprise Linux 7 Authentication Guide』 を参照してください。

11.1.5.1. 旧認証情報を LDAP 形式に移行する

migrationtools パッケージは、認証情報を LDAP 形式に移行する時に役立つシェルスクリプトと Perl スクリプトのセットを提供します。このパッケージをインストールするには、シェルプロンプトで以下を入力します。

```
~]# yum install migrationtools
```

これでスクリプトが **/usr/share/migrationtools/** ディレクトリーにインストールされます。インストール後に、**/usr/share/migrationtools/migrate_common.ph** ファイルを編集して、以下の行を変更して正しいドメインを反映させます。例を示します。

```
# Default DNS domain
$DEFAULT_MAIL_DOMAIN = "example.com";

# Default base
$DEFAULT_BASE = "dc=example,dc=com";
```

別の方法として、コマンドライン上で環境変数を直接指定することもできます。たとえば、**dc=example,dc=com** に設定されているデフォルトベースを使って**migrate_all_online.sh** スクリプトを実行するには、以下を入力します。

```
~]# export DEFAULT_BASE="dc=example,dc=com" \
/usr/share/migrationtools/migrate_all_online.sh
```

ユーザーデータベースの移行を実行するスクリプトを決定するには、表11.8「一般的に使用される LDAP 移行スクリプト」 を参照してください。

表11.8 一般的に使用される LDAP 移行スクリプト

既存のネームサービス	LDAP は実行中ですか?	使用するスクリプト
/etc フラットファイル	はい	**migrate_all_online.sh**
/etc フラットファイル	いいえ	**migrate_all_offline.sh**
NetInfo	はい	**migrate_all_netinfo_online.sh**
NetInfo	いいえ	**migrate_all_netinfo_offline.sh**
NIS (YP)	はい	**migrate_all_nis_online.sh**
NIS (YP)	いいえ	**migrate_all_nis_offline.sh**

これらのスクリプトの使用方法に関する詳細情報は、**/usr/share/doc/migrationtools-*version*/** ディレクトリー内の **README** および **migration-tools.txt** ファイルを参照してください。

11.1.6. その他のリソース

以下のリソースは、LDAP に関する追加情報を提供します。システムに LDAP を設定する前に、以下の資料、特に 『OpenLDAP Software Administrator's Guide (OpenLDAP ソフトウェア管理者ガイド)』 を確認することが強く推奨されます。

11.1.6.1. インストールされているドキュメント

以下のドキュメントは、*openldap-servers* パッケージでインストールされます。

/usr/share/doc/openldap-servers-*version*/guide.html

『OpenLDAP Software Administrator's Guide』 のコピー

/usr/share/doc/openldap-servers-*version*/README.schema

インストールされているスキーマファイルの説明を含む README ファイル

また、*openldap*、*openldap-servers*、*openldap-clients* パッケージでインストールされる多くの man ページがあります。

クライアントアプリケーション

- **man ldapadd** — LDAP ディレクトリーにエントリーを追加する方法を説明しています。
- **man ldapdelete** — LDAP ディレクトリーからエントリーを削除する方法を説明しています。
- **man ldapmodify** — LDAP ディレクトリー内でエントリを修正する方法を説明しています。
- **man ldapsearch** — LDAP ディレクトリー内でエントリーを検索する方法を説明しています。
- **man ldappasswd** — LDAP ユーザーのパスワードを設定または変更する方法を説明しています。
- **man ldapcompare** — **ldapcompare** ツールの使用方法を説明しています。
- **man ldapwhoami** — **ldapwhoami** ツールの使用方法を説明しています。
- **man ldapmodrdn** — エントリーの RDN を修正する方法を説明しています。

サーバーアプリケーション

- **man slapd** — LDAP サーバー用のコマンドラインオプションを説明しています。

管理アプリケーション

- **man slapadd** — **slapd** データベースにエントリーを追加するコマンドラインオプションを説明しています。
- **man slapcat** — **slapd** データベースから LDIF ファイルを生成するために使用するコマンドラインオプションを説明しています。
- **man slapindex** — **slapd** データベースのコンテンツを基にしたインデックスの再生成に使用するコマンドラインオプションを説明しています。
- **man slappasswd** — LDAP ディレクトリー用のユーザーパスワードを生成するコマンドラインオプションを説明しています。

設定ファイル

- **man ldap.conf** — LDAP クライアント用の設定ファイル内で利用可能な形式とオプションを説明しています。
- **man slapd-config** — 設定ディレクトリー内で利用可能な形式とオプションを説明しています。

11.1.6.2. 役立つ Web サイト

http://www.openldap.org/doc/admin24/

『OpenLDAP Software Administrator's Guide (OpenLDAP ソフトウェア管理者ガイド)』 の現行バージョンです。

http://www.kingsmountain.com/ldapRoadmap.shtml

LDAP プロトコルに関する役立つリソースと最新ニュースへのリンクが含まれる Jeff Hodges 著の 『LDAP Roadmap & FAQ』 です。

http://www.ldapman.org/articles/

ディレクトリーツリーの設計とディレクトリー構造のカスタマイズを行う方法が含まれた、LDAP に関する優れた概要を提供する記事を集めたものです。

http://www.padl.com/

役立つ LDAP ツールが記載された開発者の Web サイトです。

11.1.6.3. 関連資料

『OpenLDAP by Example』 (John Terpstra、Benjamin Coles 共著、Prentice Hall 社出版)

OpenLDAP 導入における実用的な練習を集めたものです。

『Implementing LDAP』 (Mark Wilcox 著、Wrox Press, Inc. 社出版)

システム管理者とソフトウェア開発者の両者の観点から LDAP について取り上げた本です。

『Understanding and Deploying LDAP Directory Services』 (Tim Howes 他共著、Macmillan Technical Publishing 社出版)

実稼働環境における LDAP の導入とその設計原理を説明した本です。

第12章 ファイルとプリントサーバー

本章では、SMB (*Server Message Block* サーバーメッセージブロック) および **CIFS** (*common Internet file system*) プロトコルのオープンソース実装である**Samba**、Red Hat Enterprise Linux に同梱されているプライマリ FTP サーバーである **vsftpd** のインストールと設定方法を紹介しています。また、プリンターを設定する **Printer Configuration (プリンター設定)** ツールの使用方法についても説明しています。

12.1. Samba

Samba は、*サーバーメッセージブロック* (**SMB**) プロトコルのオープンソース実装です。このプロトコルの最新バージョンは、*common Internet file system* (**CIFS**) プロトコルとも呼ばれます。Samba は、Microsoft Windows®、Linux、UNIX およびその他のオペレーティングシステムが混在するネットワーク構築と、Windows ベースのファイルおよびプリンター共有を可能にします。Samba は **SMB** を使用しているため、Windows クライアントには Windows サーバーとして表示されます。

注記

Samba を使用するには、最初に **root** として以下のコマンドを実行し、*samba* パッケージがシステムにインストールされていることを確認します。

```
~]# yum install samba
```

yum を使ったパッケージのインストールについては 「パッケージのインストール」を参照してください。

12.1.1. Samba の概要

Red Hat Enterprise Linux 7 には Samba バージョン **4.1** が装備されています。

12.1.1.1. Samba の機能

Samba は強力かつ多用途なサーバーアプリケーションです。

Samba の特徴

- Linux、UNIX、Windows クライアントにディレクトリーツリーとプリンターのサービスを提供
- ネットワークブラウジングのサポート (NetBIOS を使用または不使用)
- Windows ドメインログインの認証
- *Windows Internet Name Service* (**WINS**) ネームサーバー解決
- Windows NT® 式の Primary Domain Controller (PDC) として機能
- Samba ベースの PDC に対し *Backup Domain Controller* (BDC) として機能
- Active Directory ドメインメンバーサーバーとして機能
- Windows NT/2000/2003/2008 PDC/Windows Server 2012 への参加

Samba に搭載されていない機能

- Windows PDC に対する BDC としての機能 (およびその逆)
- Active Directory ドメインコントローラーとしての機能

12.1.2. Samba のデーモンと関連サービス

Samba の各種デーモンとサービスの概要は以下の通りです。

12.1.2.1. Samba のデーモン

Samba は、3 つのデーモン (**smbd**、**nmbd** および **winbindd**) で構成されています。3 つのサービス (**smb**、**nmb** および **winbind**) は、デーモンの起動、停止、その他のサービス関連機能を制御します。各デーモンの詳細やそのデーモンを制御する特定のサービスについて、以下に記載します。

smbd

smbd サーバーのデーモンは、Windows クライアントに対してファイル共有と印刷サービスを提供します。また、**SMB** プロトコルを介したユーザー認証、リソースのロック、データ共有も実行します。サーバーが **SMB** トラフィックをリッスンするデフォルトのポートは、**TCP** ポート、**139**、**445** です。

smbd デーモンは、**smb** サービスによって制御されます。

nmbd

nmbd サーバーデーモンは、Windows ベースのシステムにおいて SMB/CIFS によって生成されるような NetBIOS ネームサービス要求を認識して、それに応答します。これらのシステムには、Windows 95/98/ME、Windows NT、Windows 2000、Windows XP、LanManager クライアントが含まれます。またこのデーモンは、Windows **ネットワークコンピュータ** ビューを構成するブラウジングプロトコルにも関与しています。サーバーが **NMB** トラフィックを待機するデフォルトのポートは、**UDP** ポート **137** です。

nmbd デーモンは、**nmb** サービスによって制御されます。

winbindd

winbind サービスは、Windows NT、2000、2003、Windows Server 2008、または Windows Server 2012 を実行しているサーバーから受け取ったユーザーとグループの情報を解決し、Windows ユーザー/グループの情報が、UNIX プラットフォームで認識可能となります。これは、Microsoft RPC コール、*Pluggable Authentication Modules* (PAM)、および *Name Service Switch* (NSS) を使用することで実現されます。これにより、Windows NT ドメインユーザーは、UNIX マシン上の UNIX ユーザーとして表示され、操作を行うこともできます。**winbind** サービスは Samba ディストリビューションでもバンドルされていますが、**smb** サービスとは別に制御されます。

winbindd デーモンは **winbind** サービスによって制御されており、操作するために **smb** サービスを起動する必要はありません。**winbindd** は、Samba が Active Directory メンバーである場合にも使用されるほか、Samba ドメインコントローラー上で (ネスト化されたグループおよびドメイン間信頼関係の実装のために) 使用される場合もあります。**winbind** は、Windows NT ベースのサーバー接続に使用されるクライアント側のサービスであるため、**winbind** の詳細説明は本章の対象外となります。

注記

Samba ディストリビューションに含まれるユーティリティーの一覧は、「Samba ディストリビューションプログラム」を参照してください。

12.1.3. Samba 共有への接続

Nautilus を使用して、ネットワーク上で利用可能な Samba 共有を表示することができます。ご使用のネットワーク上の Samba ワークグループとドメインの一覧を表示するには、GNOME パネルから **Places (場所)** → **Network (ネットワーク)** の順にクリックして、任意のネットワークを選択します。また、**Nautilus** の **File (ファイル)** → **Open Location (場所を開く)** のバーで **smb:** を入力して、ワークグループ/ドメインを表示することも可能です。

図12.1「Nautilus で表示される SMB ワークグループ」に示されているように、ネットワーク上で利用可能な **SMB** ワークグループまたはドメインはそれぞれアイコンで表示されます。

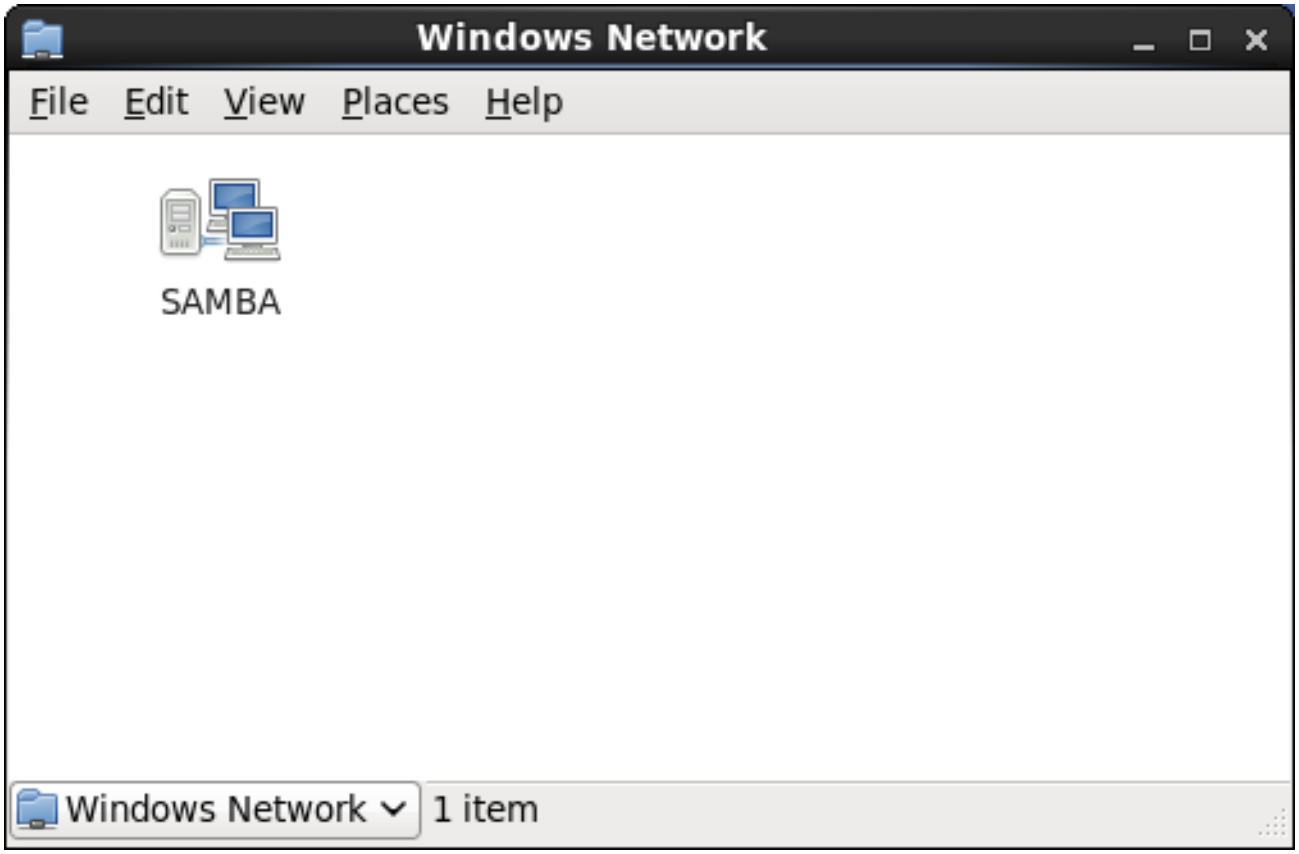


図12.1 Nautilus で表示される SMB ワークグループ

ワークグループ/ドメインのアイコンの一つをダブルクリックすると、そのワークグループ/ドメイン内にあるコンピューターの一覧が表示されます。

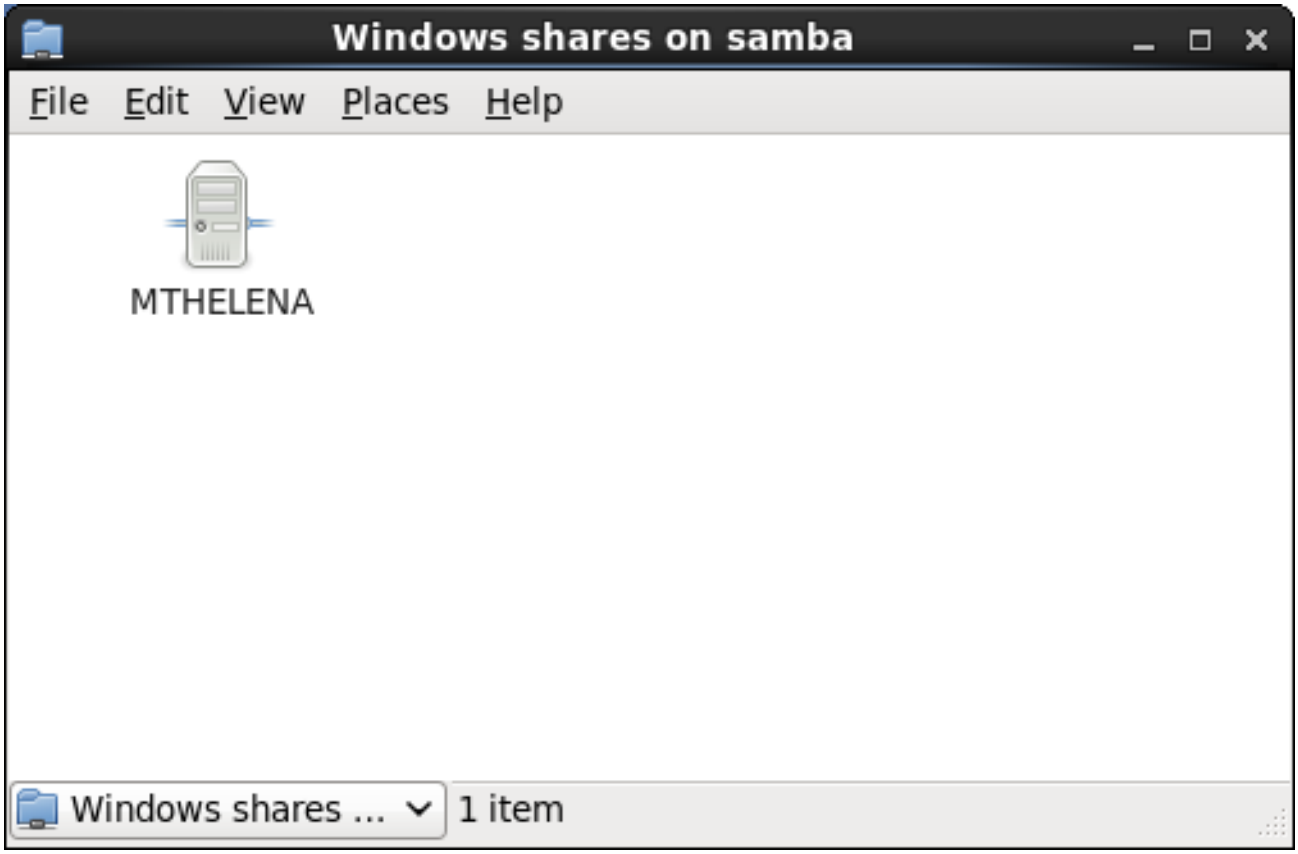


図12.2 Nautilus で表示される SMB マシン

図12.2「Nautilus で表示される SMB マシン」にあるように、ワークグループ内のマシン 1 台につき 1 つのアイコンが表示されます。アイコンをダブルクリックすると、そのマシン上の Samba 共有が表示されます。ユーザー名とパスワードの組み合わせが必要な場合は、プロンプト表示されます。

また、以下のような構文を使用して、**Nautilus** の **場所** のバーで Samba サーバーと共有名を指定することも可能です。(*<servername>* と *<sharename>* は、適切な値に置き換えて下さい)。

```
smb://<servername>/<sharename>
```

12.1.3.1. コマンドライン

シェルプロンプトから Samba 共有に接続するには、以下のコマンドを入力します。

```
~]$ smbclient //<hostname>/<sharename> -U <username>
```

<hostname> を、接続先の Samba サーバーのホスト名または**IP** アドレスに置き換えます。また、*<sharename>* は閲覧する共有ディレクトリー名に、*<username>* は対象システムの Samba ユーザー名に置き換えます。正しいパスワードを入力して下さい。パスワードが不要な場合には、**Enter** キーを押します。

smb:\> のプロンプトが表示されたら、ログインに成功したことになります。ログインしたら、**help** と入力して、コマンドの一覧を表示します。自分のホームディレクトリーの内容を確認したい場合は、*sharename* を自分のユーザー名に置き換えて下さい。**-U** スイッチが使用されていない場合は、現在のユーザーのユーザー名が Samba サーバーに渡されます。

smbclient を終了するには、**smb:\>** プロンプトで**exit** と入力します。

12.1.3.2. 共有のマウント

Samba 共有をディレクトリーにマウントすると、そのディレクトリー内のファイルは、ローカルファイルシステムの一部であるかのように扱うことができるので、その方が便利な場合があります。

Samba 共有をディレクトリーにマウントするには、マウントするためのディレクトリーを新規作成し (既に存在していない場合)、以下のコマンドを**root**として実行します。

```
~]# mount -t cifs //<servername>/<sharename> /mnt/point/ -o
username=<username>,password=<password>
```

このコマンドにより、 ローカルディレクトリ */mnt/point/* 内の *<servername>* から *<sharename>* がマウントされます。

注記

mount.cifs ユーティリティーは、(Samba には依存しない) 別の RPM です。**mount.cifs** を使用するには、まず **root** として以下のコマンド実行し、*cifs-utils* パッケージがシステムにインストールされていることを確認します。

```
~]# yum install cifs-utils
```

yum を使ったパッケージのインストールについては 「パッケージのインストール」 を参照してください。

cifs-utils パッケージには、Kerberos 化した CIFS マウントを実行するためにカーネルから呼び出された **cifs.upcall** バイナリも含まれます。**cifs.upcall** の詳細については、**man cifs.upcall** を参照してください。

Samba 共有のマウントに関する詳細は、**man mount.cifs** を参照してください。

警告

認証のためにプレーンテキストパスワードが必要な CIFS サーバーもあります。プレーンテキストパスワード認証のサポートを有効にするには、次のコマンドを実行します。

```
~]# echo 0x37 > /proc/fs/cifs/SecurityFlags
```

警告: この操作により、パスワードの暗号化を解除することでパスワードが漏洩する可能性があります。

12.1.4. Samba サーバーの設定

デフォルトの設定ファイル (**/etc/samba/smb.conf**) では、ユーザーが自分のホームディレクトリーを Samba 共有として表示できるようになっています。また、システム用に Samba 共有プリンターとして設定されたすべてのプリンターを共有することもできます。システムにプリンターを接続して、ご使用のネットワーク上の Windows マシンから印刷することができます。

12.1.4.1. グラフィカル設定

グラフィカルインターフェースを使用して Samba を設定するには、利用可能な Samba のグラフィカルユーザーインターフェースを使用して下さい。現在利用可能な GUI の一覧は http://www.samba.org/samba/GUI/ をご覧下さい。

12.1.4.2. コマンドラインからの設定

Samba では、**/etc/samba/smb.conf** が設定ファイルとして使用されます。この設定ファイルを変更しても、**root** で以下のコマンドを使用して Samba デーモンを再起動するまで変更内容は反映されません。

```
~]# systemctl restart smb.service
```

Windows ワークグループと Samba サーバーの簡略な説明を指定するには、**/etc/samba/smb.conf** ファイルで以下の行を編集します。

```
workgroup = WORKGROUPNAME
server string = BRIEF COMMENT ABOUT SERVER
```

WORKGROUPNAME の箇所は、このマシンが所属すべき Windows ワークグループの名前に置き換えます。*BRIEF COMMENT ABOUT SERVER* は、Samba システムについての Windows のコメントとして使用されるオプションです。

ご使用の Linux システム上に Samba 共有ディレクトリーを作成するには、**/etc/samba/smb.conf** ファイルに以下のセクションを (ご自分のニーズとご使用のシステムを反映させるよう変更してから) 追加してください。

```
[sharename]
comment = Insert a comment here
path = /home/share/
valid users = tfox carole
public = no
writable = yes
printable = no
create mask = 0765
```

上記の例では、**tfox** と **carole** の 2 人のユーザーが、Samba クライアントから Samba サーバー上の **/home/share** ディレクトリーの読み取り/書き込みを行うことができます。

12.1.4.3. 暗号化されたパスワード

パスワードの暗号化は使用に際してより安全であることから、デフォルトで有効になっています。暗号化されたパスワードを使用するユーザーを作成するには、**smbpasswd -a <username>** のコマンドを使用します。

12.1.5. Samba の起動と停止

Samba サーバーを起動するには、**root** でシェルプロンプトに以下のコマンドを入力します。

```
~]# systemctl start smb.service
```

重要

ドメインメンバーのサーバーを設定するには、**smb** サービスを**起動する前に net join** のコマンドを使用して、まずドメインまたは Active Directory に参加する必要があります。また、**smbd** の前に **winbind** を実行することも推奨されます。

Samba サーバーを停止するには、**root** でシェルプロンプトに以下のコマンドを入力します。

```
~]# systemctl stop smb.service
```

restart オプションは、Samba を停止して起動する、迅速な方法です。これは、Samba の設定ファイルを編集した後に設定の変更内容を有効にする、最も信頼できる方法です。最初にデーモンが稼働していなかった場合でも、restart のオプションにより起動される点に注意して下さい。

サーバーを再起動するには、**root** でシェルプロンプトに以下のコマンドを入力します。

```
~]# systemctl restart smb.service
```

condrestart (*条件付き再起動*) のオプションは、現在稼働中であるという条件でのみ**smb** を起動させます。このオプションは、デーモンが稼働していない場合には開始しないため、スクリプトに有用です。

注記

/etc/samba/smb.conf ファイルに変更が加えられた場合、Samba は数分後にそのファイルを自動的に再読み込みします。手動で **restart** または **reload** のコマンドを発行して設定ファイルを再読み込みすることもできます。

条件付きでサーバーを再起動するには、**root** で以下のコマンドを入力します。

```
~]# systemctl try-restart smb.service
```

手動での **/etc/samba/smb.conf** ファイルの再読み込みは、**smb** サービスによる自動再読み込みが失敗した場合に役立ちます。サービスを再起動せずに Samba サーバーの設定ファイルを確実に再読み込みするには、**root** で以下のコマンドを入力します。

```
~]# systemctl reload smb.service
```

デフォルトでは、**smb** サービスがブート時に自動的に起動することは**ありません**。ブート時に Samba が起動するようにするには、**root** でシェルプロンプトに以下を入力します。

```
~]# systemctl enable smb.service
```

これらのツールに関する詳細情報は、6章*systemd によるサービス管理*を参照してください。

12.1.6. Samba のネットワークブラウジング

*ネットワークブラウジング*により、Windows と Samba のサーバーが Windows の**ネットワークコンピュータ (Network Neighborhood)** に表示されるようになります。**ネットワークコンピュータ** には、アイコンがサーバーとして表示され、それを開くとそのサーバーの利用可能な共有やプリンターが表示されます。

ネットワークブラウジングの機能には **TCP/IP** 上の NetBIOS が必要です。NetBIOS ベースのネットワーキングでは、ブロードキャスト (**UDP**) メッセージングを使用してブラウズリスト管理を行います。NetBIOS と WINS を **TCP/IP** ホスト名解決の第一の方法として使用しない場合は、静的ファイル (**/etc/hosts**) や **DNS** などの他の方法を使用する必要があります。

ドメインマスターブラウザーは、すべてのサブネット上のローカルマスターブラウザーからブラウズリストを照合して、ワークグループとサブネット間のブラウジングを可能にします。独自のサブネットには、ドメインマスターブラウザーをローカルマスターブラウザーにするのが望ましいでしょう。

12.1.6.1. ドメインブラウジング

デフォルトでは、ドメイン用の Windows サーバーの PDC は、そのドメイン用のドメインマスターブラウザでもあります。この種の状況では、Samba サーバーをドメインマスターサーバーとして設定 **しないでください**。

Windows サーバーの PDC が含まれていないサブネットの場合、Samba サーバーはローカルマスターブラウザーとして実装することができます。ドメインコントローラー環境での **/etc/samba/smb.conf** ファイルのローカルマスターブラウザー用 (もしくはブラウジング一切なし) 設定は、ワークグループ設定と同様になります (「Samba サーバーの設定」 を参照してください)。

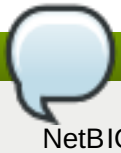
12.1.6.2. WINS (Windows Internet Name Server)

Samba サーバーまたは Windows NT サーバーは、WINS サーバーとして機能させることが可能です。NetBIOS が有効な状態で WINS サーバーを使用する場合には、UDP ユニキャストをルーティングして、ネットワーク全体にわたる名前解決を可能にすることができます。WINS サーバーを使用しない場合は、UDP ブロードキャストがローカルのサブネットに限定されるため、他のサブネット、ワークグループ、ドメインにルーティングすることはできません。Samba は現在 WINS の複製に対応していないため、WINS の複製が必要な場合には、Samba をプライマリ WINS サーバーとして使用しないでください。

NT/2000/2003/2008 サーバーと Samba が混在する環境においては、Microsoft WINS 機能を使用することが推奨されます。Samba のみの環境では、WINS には **単一の** Samba サーバーを使用することが推奨されます。

以下は、Samba サーバーが WINS サーバーとして機能する場合の **/etc/samba/smb.conf** ファイルの一例です:

```
[global]
wins support = Yes
```



注記

NetBIOS 名を解決するには、全サーバー (Samba を含む) を WINS サーバーに接続する必要があります。WINS がない場合は、ブラウジングはローカルのサブネット上でのみ行われます。また、ドメイン全体の一覧を取得したとしても、WINS を使用しないクライアントに対してはホストを解決できません。

12.1.7. Samba ディストリビューションプログラム

net

net *<protocol>* *<function>* *<misc_options>* *<target_options>*

net ユーティリティーは、Windows や MS-DOS で使用する **net** ユーティリティーと似ています。最初の引数は、コマンド実行時のプロトコルを指定するために使用されます。***<protocol>*** オプションには、**ads**、**rap**、または **rpc** を用いて、サーバー接続のタイプを指定することができます。Active Directory では **ads**、Win9x/NT3 では **rap**、Windows NT4/2000/2003/2008 では **rpc** を使用します。プロトコルが省略されている場合には、**net** は自動的に判定を試みます。

以下の例は、**wakko** という名前のホストが利用可能な共有のリストを示しています。

```
~]$ net -l share -S wakko
Password:
Enumerating shared resources (exports) on remote server:
Share name   Type     Description
----------   ----     -----------
data         Disk     Wakko data share
tmp          Disk     Wakko tmp share
IPC$         IPC      IPC Service (Samba Server)
ADMIN$       IPC      IPC Service (Samba Server)
```

以下の例は、**wakko** という名前のホストの Samba ユーザーの一覧を示しています。

```
~]$ net -l user -S wakko
root password:
User name             Comment
-----------------------------
andriusb              Documentation
joe                   Marketing
lisa                  Sales
```

nmblookup

nmblookup *<options> <netbios_name>*

nmblookup プログラムは、NetBIOS 名を **IP** アドレスに解決します。このプログラムは、ターゲットマシンが応答するまでローカルサブネット上のクエリをブロードキャストします。

以下の例では、NetBIOS 名 **trek** の **IP** アドレスを表示しています。

```
~]$ nmblookup trek
querying trek on 10.1.59.255
10.1.56.45 trek<00>
```

pdbedit

pdbedit *<options>*

pdbedit プログラムは、SAM データベース内にあるアカウントを管理します。**smbpasswd**、LDAP、**tdb** データベースライブラリを含むすべてのバックエンドがサポートされています。

以下はユーザーの追加、削除、一覧表示の例です

```
~]$ pdbedit -a kristin
new password:
```

```
retype new password:
Unix username:        kristin
NT username:
Account Flags:        [U          ]
User SID:             S-1-5-21-1210235352-3804200048-1474496110-2012
Primary Group SID:    S-1-5-21-1210235352-3804200048-1474496110-2077
Full Name: Home Directory:       \\wakko\kristin
HomeDir Drive:
Logon Script:
Profile Path:         \\wakko\kristin\profile
Domain:               WAKKO
Account desc:
Workstations: Munged
dial:
Logon time:           0
Logoff time:          Mon, 18 Jan 2038 22:14:07 GMT
Kickoff time:         Mon, 18 Jan 2038 22:14:07 GMT
Password last set:    Thu, 29 Jan 2004 08:29:28
GMT Password can change:  Thu, 29 Jan 2004 08:29:28 GMT
Password must change: Mon, 18 Jan 2038 22:14:07 GMT
~]$ pdbedit -v -L kristin
Unix username:        kristin
NT username:
Account Flags:        [U          ]
User SID:             S-1-5-21-1210235352-3804200048-1474496110-2012
Primary Group SID:    S-1-5-21-1210235352-3804200048-1474496110-2077
Full Name:
Home Directory:       \\wakko\kristin
HomeDir Drive:
Logon Script:
Profile Path:         \\wakko\kristin\profile
Domain:               WAKKO
Account desc:
Workstations: Munged
dial:
Logon time:           0
Logoff time:          Mon, 18 Jan 2038 22:14:07 GMT
Kickoff time:         Mon, 18 Jan 2038 22:14:07 GMT
Password last set:    Thu, 29 Jan 2004 08:29:28 GMT
Password can change:  Thu, 29 Jan 2004 08:29:28 GMT
Password must change: Mon, 18 Jan 2038 22:14:07 GMT
~]$ pdbedit -L
andriusb:505:
joe:503:
lisa:504:
kristin:506:
~]$ pdbedit -x joe
~]$ pdbedit -L
andriusb:505: lisa:504: kristin:506:
```

rpcclient

rpcclient <*server*> <*options*>

rpcclient プログラムは、システム管理用の Windows 管理グラフィカルユーザーインターフェース (GUI) へのアクセスを提供する、Microsoft RPC を使用した管理コマンドを発行します。ほとんどの場合、このプログラムを使用するのは、Microsoft RPC の複雑性を完全に理解している上級ユーザーです。

smbcacls

smbcacls *<//server/share> <filename> <options>*

smbcacls プログラムは、Samba サーバーまたは Windows サーバーによって共有されているファイルやディレクトリー上の Windows ACL を変更します。

smbclient

smbclient *<//server/share> <password> <options>*

smbclient プログラムは、**ftp** と同様の機能を提供する多用途の UNIX クライアントです。

smbcontrol

smbcontrol -i *<options>*

smbcontrol *<options> <destination> <messagetype> <parameters>*

smbcontrol プログラムは、実行中の**smbd**、**nmbd** または**winbindd** デーモンに制御メッセージを送信します。**smbcontrol -i** を実行すると、空白行または ***'q'*** と入力するまでコマンドはインタラクティブに実行されます。

smbpasswd

smbpasswd *<options> <username> <password>*

smbpasswd プログラムは、暗号化されたパスワードを管理します。このプログラムは、スーパーユーザーが任意のユーザーのパスワードを変更したり、一般ユーザーが自分の Samba パスワードを変更するために実行可能です。

smbspool

smbspool *<job> <user> <title> <copies> <options> <filename>*

smbspool プログラムは、Samba への CUPS 対応印刷インターフェースです。**smbspool** は CUPS プリンターでの使用に設計されていますが、非 CUPS プリンターでも機能します。

smbstatus

smbstatus *<options>*

smbstatus プログラムは、Samba サーバーへの現在の接続状態を表示します。

smbtar

smbtar *<options>*

smbtar プログラムは、Windows ベースの共有ファイルおよびディレクトリーのローカルテープアーカイブへのバックアップと復元を実行します。**tar** コマンドと似ていますが、これら 2 つのコマンドには互換性はありません。

testparm

testparm *<options> <filename> <hostname IP_address>*

testparm プログラムは **/etc/samba/smb.conf** ファイルの構文を確認します。ご使用の **smb.conf** ファイルがデフォルトの場所 (**/etc/samba/smb.conf**) にある場合は、場所を指定する必要はありません。**testparm** プログラムにホスト名と **IP** アドレスを指定すると、**hosts.allow** と **host.deny** のファイルが正しく設定されているかどうかを検証します。**testparm** プログラムは、テスト後にご使用の **smb.conf** ファイルの概要やサーバーのロール (スタンドアロン、ドメインなど) も表示します。コメントが除外され、経験のある管理者が読む情報が簡潔に表示されるため、デバッグを行う際に便利です。

例を示します。

```
~]$ testparm
Load smb config files from /etc/samba/smb.conf
Processing section "[homes]"
Processing section "[printers]"
Processing section "[tmp]"
Processing section "[html]"
Loaded services file OK.
Server role: ROLE_STANDALONE
Press enter to see a dump of your service definitions
<enter>
# Global parameters
[global]
 workgroup = MYGROUP
 server string = Samba Server
 security = SHARE
 log file = /var/log/samba/%m.log
 max log size = 50
 socket options = TCP_NODELAY SO_RCVBUF=8192 SO_SNDBUF=8192
 dns proxy = No
[homes]
 comment = Home Directories
 read only = No
 browseable = No
[printers]
 comment = All Printers
 path = /var/spool/samba
 printable = Yes
 browseable = No
[tmp]
 comment = Wakko tmp
 path = /tmp
 guest only = Yes
```

```
[html]
 comment = Wakko www
 path = /var/www/html
 force user = andriusb
 force group = users
 read only = No
 guest only = Yes
```

wbinfo

wbinfo *<options>*

wbinfo プログラムは、**winbindd** デーモンからの情報を表示します。**wbinfo** が機能するには、**winbindd** デーモンが実行されている必要があります。

12.1.8. その他のリソース

以下のセクションには、Samba に関する詳しい情報を調べる方法を記載しています。

12.1.8.1. インストールされているドキュメント

- **/usr/share/doc/samba-*<version-number>*/** — Samba ディストリビューションに同梱されているすべての追加ファイル。ヘルパースクリプトやサンプルの設定ファイル、ドキュメントなどが含まれます。

  特定の **Samba** 機能の詳細については、次の man ページを参照してください。

  - **smb.conf**
  - **samba**
  - **smbd**
  - **nmbd**
  - **winbind**

12.1.8.2. 役立つ Web サイト

- http://www.samba.org/ — Samba 開発チームが作成した Samba ディストリビューションおよびすべての正式なドキュメントのホームページ。数々のリソースを HTML と PDF 形式で入手できるほか、有償配布のみの書籍も取り扱っています。これらのリンクの多くは、Red Hat Enterprise Linux 固有ではありませんが一部のコンセプトは該当します。
- http://samba.org/samba/archives.html — Samba コミュニティのアクティブな電子メール一覧。リストのアクティビティレベルが高いため、ダイジェストモードを有効にすることが推奨されます。
- Samba ニュースグループ — **NNTP** プロトコルを使用する www.gmane.org などの Samba のスレッドのニュースグループも利用可能。メーリングリストの電子メールを受信する代わりに利用することができます。

12.2. FTP

ファイル転送プロトコル(**FTP**) は、今日インターネット上で見られる、最も古く、一般的に使用されてい

るプロトコルです。この目的は、ユーザーがリモートホストに直接ログインしたり、リモートシステムの使用法についての知識がなくとも、ネットワーク上のコンピューターホスト間で確実にファイルを転送することです。これによりユーザーは、標準の簡単なコマンドセットを使用してリモートシステム上のファイルにアクセスすることができます。

本セクションでは、**FTP** プロトコルの基本および Red Hat Enterprise Linux に標準装備されているプライマリ **FTP** サーバーの ***vsftpd*** について概説します。

12.2.1. ファイル転送プロトコル (FTP)

FTP はクライアント-サーバーアーキテクチャーを使い、**TCP** ネットワークプロトコルを使用してファイルを転送します。**FTP** は古いプロトコルであることから、暗号化されていないユーザー名とパスワード認証を使用します。このため、安全でないプロトコルとみなされており、絶対的に必要でない限り、使用するべきではありません。しかし、**FTP** はインターネット上で非常に普及しているので、共有ファイルの公開で必要となる場合がよくあります。このため、システム管理者は、**FTP** プロトコル独自の特性を認識しておくべきです。

本セクションでは、**vsftpd** を設定して **SSL** による安全を確保する接続の確立方法と、**SELinux** を用いて **FTP** サーバーを安全にする方法を説明しています。**FTP** の代用となるのは、**OpenSSH**スイーツからの **sftp** です。**OpenSSH** の設定方法および **SSH** プロトコル全般に関する情報は、*7章OpenSSH* を参照してください。

インターネット上で使用されているほとんどのプロトコルとは異なり、**FTP** が正しく機能するためには複数のネットワークポートを必要とします。**FTP** クライアントアプリケーションが**FTP** サーバーへの接続を開始する際に、*コマンドポート* として知られるポート **21** をサーバー上で開きます。このポートは、すべてのコマンドをサーバーに発行するために使用されます。サーバーから要求されたデータはいずれも *データポート* を介してクライアントに返されます。データ接続が開始されるデータ接続用ポートの番号は、クライアントが *アクティブ* か *パッシブ* のモードでデータを要求するかによって変化します。

これらのモードについての定義は以下の通りです。

アクティブモード

アクティブモードは、**FTP** プロトコルでクライアントへのデータ転送に使用される独自の方法です。**FTP** クライアントがアクティブモードのデータ転送を開始すると、サーバーはサーバー上のポート **20** からクライアントの指定する**IP** アドレスとランダムで権限のないポート (**1024** 以上) への接続を開きます。この方法では、クライアントマシンがポート **1024** 以上での接続を受け入れるように許可されている必要があります。インターネットのようなセキュリティー保護されていないネットワークが増加するにともない、ファイアウォールを使用したクライアントマシンの保護が普及しています。このようなクライアント側のファイアウォールはアクティブモードの **FTP** サーバーから着信する接続を拒否する場合が多いため、パッシブモードが考案されました。

パッシブモード

パッシブモードはアクティブモードと同様に、**FTP** クライアントアプリケーションによって開始されます。サーバーからのデータを要求する際に、**FTP** クライアントはパッシブモードでデータにアクセスしたいことを知らせると、サーバーはサーバー上の **IP** アドレスとランダムな非特権ポート (**1024** 以上) を提供します。クライアントは、サーバー上のそのポートに接続して要求した情報をダウンロードします。

パッシブモードは、クライアント側のファイアウォールによるデータ接続障害の問題を解決しますが、サーバー側のファイアウォール管理を複雑化させてしまう場合があります。**FTP** サーバー上の特権のないポートの範囲を制限することにより、サーバー上で開いておりポート数を減らすことができます。またこの方法により、サーバーを対象としたファイアウォールのルール設定の手順が簡略化されます。

12.2.2. vsftpd サーバー

vsftpd (*The Very Secure FTP Daemon*) は、高速で安定性があり、(より重要なのは) セキュアであるようにゼロから設計されています。**vsftpd** は、多数の接続を効率的かつセキュアに処理できる能力があることから、Red Hat Enterprise Linux に標準装備されている、唯一のスタンドアロン型 **FTP** サーバーです。

vsftpd で使用されるセキュリティーモデルには、以下に挙げる 3 つの主要な側面があります。

- **特権プロセスと非特権プロセスの確固たる分離**— 別個のプロセスが異なるタスクを処理します。各プロセスは、タスクに必要な最低限の権限で稼働します。
- **高い権限を必要とするタスクを、必要最小限の権限をともなうプロセスで処理**— **libcap** ライブラリー内にある互換性を利用して、通常は完全な root 権限を必要とするタスクを、権限が低いプロセスでより安全に実行することができます。
- **ほとんどのプロセスを chroot jail で実行** — 可能な場合は常に、プロセスは共有ディレクトリーにルートディレクトリーを変更します。すると、このディレクトリーは **chroot** jail と見なされます。例えば、ディレクトリー **/var/ftp/** がプライマリー共有ディレクトリーの場合、**vsftpd** は **/var/ftp/** を **/** として知られる、新規 root ディレクトリーに再度割り当てます。これにより、新たな root ディレクトリー下に格納されていないディレクトリーに対する、潜在的な悪質ハッカー行為を行うことができなくなります。

これらのセキュリティープラクティスを使用すると、**vsftpd** によるリクエスト対応方法に以下のような影響があります。

- **親プロセスは、必要最小限の権限で稼働します**— 親プロセスは、リスクレベルを最低限に抑えるために必要とされる権限のレベルを動的に算出します。子プロセスは、**FTP** クライアントとの直接的なインタラクションを処理し、可能な限り権限無しに近い形で稼働します。
- **高い権限を必要とするオペレーションはすべて、小さな親プロセスによって処理されます**— **Apache HTTP Server** の場合とほぼ同様に、**vsftpd** は権限のない子プロセスを起動し、着信接続を処理します。これにより、権限のある親プロセスを最小限に抑えられ、比較的少ないタスクを処理することになります。
- **親プロセスは、権限のない子プロセスからのリクエストはどれも信頼しません**— 子プロセスとの通信はソケット上で受信され、子プロセスからの情報の有効性は動作を実施する前にチェックされます。
- **FTP クライアントとのインタラクションの大半は、chroot jail 内の権限のない子プロセスによって処理されます** — これらの子プロセスには権限がなく、共有ディレクトリーへのアクセスしかないため、プロセスがクラッシュした際に攻撃者がアクセスできるのは共有ファイルのみです。

12.2.2.1. vsftpd の起動と停止

現行のセッションで **vsftpd** サービスを起動するには、シェルプロンプトで**root** として以下を入力します。

```
~]# systemctl start vsftpd.service
```

現在のセッションでサービスを停止するには、**root** で以下を入力します。

```
~]# systemctl stop vsftpd.service
```

vsftpd サービスを再起動するには、**root** で以下のコマンドを実行します。

```
~]# systemctl restart vsftpd.service
```

このコマンドで **vsftpd** サービスを停止させた直後に再起動します。これが、**FTP** サーバーの設定ファイルを編集した後に変更を反映させる最も効率的な方法になります。別の方法では、**vsftpd** サービスが実行中の場合にのみ、以下のコマンドを使って再起動することができます。

```
~]# systemctl try-restart vsftpd.service
```

デフォルトでは、**vsftpd** サービスがブート時に自動的に起動することは**ありません**。ブート時に **vsftpd** が起動するようにするには、**root** でシェルプロンプトに以下を入力します。

```
~]# systemctl enable vsftpd.service
ln -s '/usr/lib/systemd/system/vsftpd.service' '/etc/systemd/system/multi-user.target.wants/vsftpd.service'
```

Red Hat Enterprise Linux 7 でシステムサービスを管理する詳細情報については、*6章systemd によるサービス管理* を参照してください。

12.2.2.2. vsftpd の複数コピーの起動

1 台のコンピューターを複数の **FTP** ドメインに使用する場合があります。これは、*マルチホーミング*と呼ばれるテクニックです。**vsftpd** を使ってマルチホーミングを行う方法の 1 つに、デーモンの複数コピーを実行し、各コピーに設定ファイルを与える方法があります。

これを行うには、まずすべての関連する **IP** アドレスをシステム上のネットワークデバイスまたはエイリアスネットワークデバイスに割り当てます。ネットワークデバイス、デバイスエイリアスの設定、さらにネットワーク設定スクリプトに関する詳細情報は、Red Hat Enterprise Linux 7 Networking Guide を参照してください。

次に、**FTP** ドメイン用の *DNS* サーバーが適切なマシンを参照するように設定する必要があります。Red Hat Enterprise Linux で使用される **DNS** プロトコル実装である**BIND** およびその設定ファイルについての情報は、Red Hat Enterprise Linux 7 Networking Guide を参照してください。

vsftpd が異なる **IP** アドレス上のリクエストに応答するには、デーモンの複数コピーが実行中である必要があります。**vsftpd**デーモンの複数インスタンスの起動を促進するために、特別な*systemd* サービスユニット (**vsftpd@.service**) が **vsftpd** 起動用にインスタンス化されたサービスとして*vsftpd* パッケージ内で提供されています。

このサービスユニットを活用するには、**FTP** サーバーの必要な各インスタンスの個別の**vsftpd** 設定ファイルを作成し、それを **/etc/vsftpd/** ディレクトリーに格納する必要があります。これらの設定ファイルは、(**/etc/vsftpd/*vsftpd-site-2*.conf** などの) 一意の名前を持ち、**root** ユーザーのみが読み込み、書き込み可能とする必要があることに注意してください。

IPv4 ネットワーク上で待機している各**FTP** サーバーの設定ファイル内で、以下の指示文は一意のものである必要があります。

```
listen_address=N.N.N.N
```

N.N.N.N を 使用中の **FTP** サイト用の**一意の IP** アドレスで置き換えます。このサイトが**IPv6** を使用している場合、代わりに **listen_address6** 指示文を使用してください。

複数の設定ファイルが **/etc/vsftpd/** ディレクトリーに格納されると、**vsftpd** デーモンの個別インスタンスは、**root** で以下のコマンドを実行することで開始可能になります。

```
~]# systemctl start vsftpd@configuration-file-name.service
```

上記のコマンドで、*configuration-file-name* を **vsftpd-site-2** などのリクエストしているサーバーの設定ファイルの一意の名前で置き換えます。設定ファイルの **.conf** 拡張子は、コマンドに含めないことに注意してください。

vsftpd デーモンの複数インスタンスを同時に開始したい場合は、systemd ターゲットユニットファイル (**vsftpd.target**) を活用することができます。これは、*vsftpd* パッケージで提供されています。この systemd ターゲットにより、個別の **vsftpd** デーモンが **/etc/vsftpd/** ディレクトリー内で利用可能な **vsftpd** 設定ファイルごとに開始されます。ターゲットをユーザーにするには、**root** で以下のコマンドを実行します。

```
~]# systemctl enable vsftpd.target
ln -s '/usr/lib/systemd/system/vsftpd.target' '/etc/systemd/system/multi-
user.target.wants/vsftpd.target'
```

上記のコマンドは、systemd サービスマネージャーが **vsftpd** サービスを (設定済みの **vsftpd** サーバーインスタンスとともに) ブート時に起動するように設定します。システムを再起動することなく、サービスを直ちに開始するには、**root** で以下のコマンドを実行します。

```
~]# systemctl start vsftpd.target
```

systemd ターゲットを使用してサービスを管理する方法については、「systemd ターゲットでの作業」 を参照してください。

サーバーごとに変更する指示文には、以下のものがあります。

- **anon_root**
- **local_root**
- **vsftpd_log_file**
- **xferlog_file**

12.2.2.3. SSL を使用した vsftpd 接続の暗号化

ユーザー名やパスワード、データを暗号化せずに送信するという **FTP** の元々の不安定な性質に対抗するために、**vsftpd** デーモンは **SSL** もしくは **TLS** プロトコルを使って接続を認証し、すべての送信を暗号化するように設定できます。**SSL** をサポートする **FTP** クライアントは、SSL が有効になっている **vsftpd** と通信する必要があることに注意してください。

SSL サポートを有効にするには、**vsftpd.conf** ファイル内の **ssl_enable** 設定指示文を **YES** に設定します。**ssl_enable** オプションが有効になると自動的にアクティブになる他の **SSL** 関連指示文のデフォルト設定では、**SSL** は適切に設定されています。特にこれに含まれるのは、全接続に **TLS** v1 プロトコルの使用を必須とすることや、非匿名のログインすべてでパスワードおよびデータ送信には **SSL** の使用を強制することなどです。

SSL の使用において **vsftpd** で他の **SSL** 関連の接続指示文を微調整する方法については、vsftpd.conf(5) man ページを参照してください。

12.2.2.4. vsftpd 用の SELinux ポリシー

(他の **ftpd** プロセスと共に) **vsftpd** デーモンを管理する SELinux ポリシーは、強制アクセス制御を定義します。これはデフォルトでは、必要最小限のアクセスに基づいています。**FTP** デーモンが特定のファイルやディレクトリーにアクセスできるようにするには、それらに適切なラベルを割り当てる必要があります。

たとえば、ファイルを匿名で共有できるようにするには、共有するファイルおよびディレクトリーに **public_content_t** ラベルを割り当てる必要があります。**chcon** コマンドを **root** で使用すると、これが可能になります。

```
~]# chcon -R -t public_content_t /path/to/directory
```

上記のコマンドでは、*/path/to/directory* をラベルを割り当てるディレクトリーへのパスに置き換えます。同様に、ファイルをアップロードするためのディレクトリーを設定するには、**public_content_rw_t** ラベルをそのディレクトリーに割り当てる必要があります。さらに、**allow_ftpd_anon_write** SELinux ブール値オプションを **1** に設定する必要があります。以下のように、**setsebool** コマンドを **root** で実行します。

```
~]# setsebool -P allow_ftpd_anon_write=1
```

ローカルユーザーが **FTP** 経由で自分のホームディレクトリーにアクセスできるようにしたい場合 (Red Ha Enterprise Linux 7 ではこれがデフォルト設定)、**ftp_home_dir** のブール値オプションを **1** に設定する必要があります。**vsftpd** がスタンドアロンモードで実行できるようになっていると (これも Red Hat Enterprise Linux 7 でデフォルトで有効になっています)、**ftpd_is_daemon** オプションも **1** に設定する必要があります。

他の有用なラベルやブール値オプションの例や **FTP** に関する SELinux ポリシーの設定方法についての詳細情報は、ftpd_selinux(8) man ページを参照してください。SELinux 全般に関する詳細情報は、Red Hat Enterprise Linux 7 SELinux User's and Administrator's Guide も参照してください。

12.2.3. その他のリソース

vsftpd についての詳細情報は、以下のリソースを参照してください。

12.2.3.1. インストールされているドキュメント

- **/usr/share/doc/vsftpd-*version-number*/** ディレクトリー — *version-number* を、インストールした *vsftpd* パッケージのバージョンに置き換えます。このディレクトリーには、ソフトウェアの基本的な情報を記載した **README** ファイルが格納されています。**TUNING** ファイルには、基本的なパフォーマンス調整についてのヒントが、また **SECURITY/** ディレクトリーには**vsftpd** で使用されているセキュリティーモデルに関する情報が含まれています。
- **vsftpd** 関連の man ページ — デーモンおよび設定ファイルには数多くの man ページがあります。以下は、重要な man ページのリストです。

  サーバーアプリケーション

  - vsftpd(8) — **vsftpd** で利用可能なコマンドラインオプションを説明しています。

  設定ファイル

  - vsftpd.conf(5) — **vsftpd** の設定ファイル内で利用可能なオプションの詳細な一覧を格納しています。
  - hosts_access(5) — **hosts.allow** および **hosts.deny** の**TCP** ラッパーの設定ファイル内で利用可能な書式とオプションについて説明しています。

  SELinux とのインタラクション

  - ftpd_selinux(8) — **ftpd** プロセスを管理する *SELinux* ポリシーと、SELinux ラベルの割り当て方およびブール値セットが説明されています。

12.2.3.2. オンラインのドキュメント

vsftpd および FTP 全般について

- http://vsftpd.beasts.org/ — **vsftpd** プロジェクトページは、最新のドキュメントやソフトウェアの作成者の連絡先を入手することができる便利なサイトです。
- http://slacksite.com/other/ftp.html — この Web サイトは、FTP のアクティブモードとパッシブモードの相異点について簡潔に説明しています。

Red Hat Enterprise Linux のドキュメンテーション

- Red Hat Enterprise Linux 7 Networking Guide — Red Hat Enterprise Linux 7 の『Networking Guide』は、このシステムにおけるネットワークインターフェイスやネットワーク、ネットワークサービスの設定および管理に関する情報が説明されています。**hostnamectl** ユーティリティーの概要のほか、これを使ってコマンドラインでホスト名を表示、設定する方法が説明されています。
- Red Hat Enterprise Linux 7 SELinux User's and Administrator's Guide — Red Hat Enterprise Linux 7 の『SELinux User's and Administrator's Guide』では、**SELinux** の原則と、**SELinux** を **Apache HTTP Server** や **Postfix**、**PostgreSQL**、または **OpenShift** などのさまざまなサービスで設定して使用する方法が詳細に説明されています。また、**SELinux** アクセスパーミッションを **systemd** が管理するシステムサービス用に設定する方法も説明しています。
- Red Hat Enterprise Linux 7 Security Guide — Red Hat Enterprise Linux 7 の『Security Guide』は、ローカルおよびリモートからの侵入、悪用、悪意のある行為に対してワークステーションおよびサーバーを保護するプロセスとプラクティスを学習する際にユーザーおよび管理の役に立ちます。また、重大なシステムサービスを保護する方法についても説明しています。

関連 RFC ドキュメント

- 『RFC 0959』 — IETF からの **FTP** プロトコルのオリジナルの*Request for Comments* (RFC)。
- 『RFC 1123』 — 短い **FTP** 関連のセクションで、RFC 0959 を拡張、明確化します。
- 『RFC 2228』 — **FTP** セキュリティー拡張。**vsftpd** が TLS および SSL 接続のサポートに必要な小規模のサブセットを実装します。
- 『RFC 2389』 — **FEAT** および **OPTS** コマンドを指定します。
- 『RFC 2428』 — **IPv6** サポート。

12.3. プリンターの設定

プリンター設定 (Printer Configuration) ツールを使うと、プリンター設定、プリンター設定ファイルの管理、印刷スプールディレクトリ、印刷フィルタ、プリンタークラスの管理ができます。

このツールはコモンユニックスプリンティングシステム (Common Unix Printing System : CUPS) を基本にしています。CUPS を使用した Red Hat Enterprise Linux バージョンからシステムをアップグレードする場合は、アップグレードプロセスは設定を行ったプリンターの設定を保持します。

注記

CUPS Web アプリケーションまたはコマンドラインから、直接同一および追加の動作をプリンターで実行できます。Web ブラウザでアプリケーションにアクセスするには http://localhost:631/ を開いてください。CUPS マニュアルについてはこの Web サイトの **ホーム** タブのリンクを参照してください。

12.3.1. プリンター設定ツールの起動

プリンター設定ツールを使うと、既存のプリンターで様々な操作を実行したり、新規プリンターを設定することができます。CUPS を直接使用することも可能です (CUPS にアクセスするには http://localhost:631/ をクリックします)。

ツールを起動するには、パネルから **システム** → **管理** → **印刷** の順にクリックするか、コマンドラインで **`system-config-printer`** コマンドを実行します。

図12.3「プリンター設定のウィンドウ」のような **`Printer Configuration`** ウィンドウが表示されます。

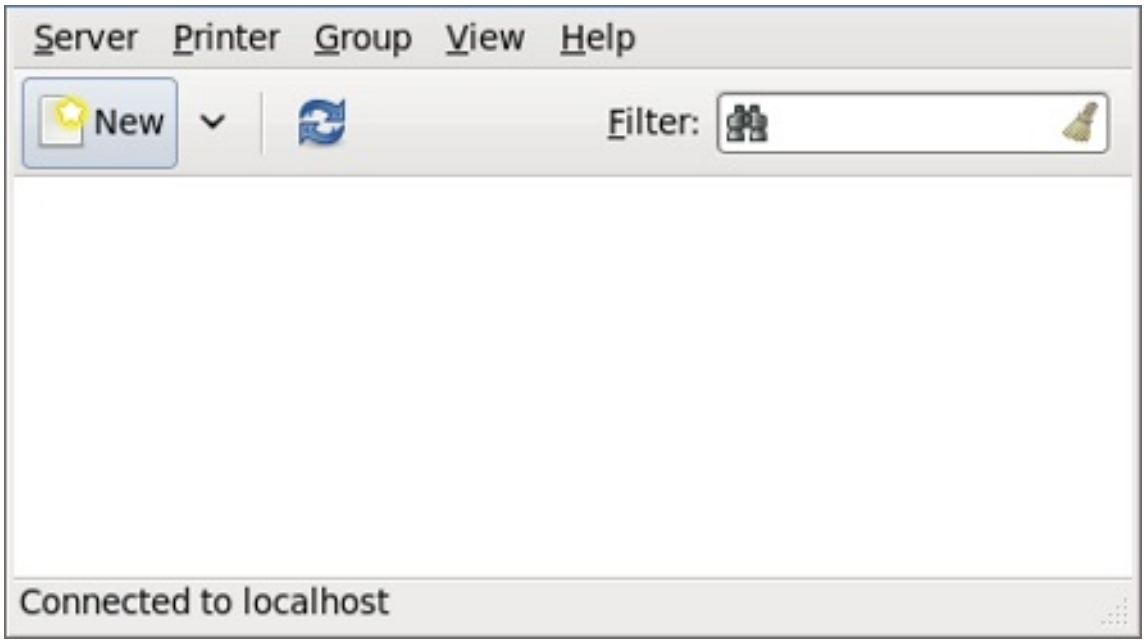


図12.3 プリンター設定のウィンドウ

12.3.2. プリンター設定の開始

プリンターの設定プロセスはプリンターキューのタイプにより異なります。

USB に接続したローカルプリンターを設定する場合は、プリンターは自動的に検出、追加されます。インストールするパッケージの確認、ルートパスワードの入力が求められます。他のポートタイプに接続したローカルプリンターやネットワークプリンターの場合は、手動で設定する必要があります。

プリンターを手動で設定するには、以下の手順にしたがいます。

1. プリンター設定ツールを起動します (「プリンター設定ツールの起動」を参照)。

2. **サーバー** → **新規** → **プリンター** の順にクリックします。
3. **Authenticate** ダイアログボックスで、確認のためルートユーザーパスワードを入力します。
4. プリンターの接続タイプを選択し、右側エリアでその詳細を記入します。

12.3.3. ローカルプリンターの追加

以下の手順にしたがって、シリアルポート以外に接続されたローカルプリンターを追加します。

1. **新規プリンター** ダイアログを開きます (「プリンター設定の開始」 を参照)。
2. デバイスが自動的に表示されない場合は、左側の一覧でプリンターを接続するポートを選択してください (**Serial Port #1** または **LPT #1** など)。
3. 右側で、接続プロパティを入力します。

   その他 の場合

   URI (例えば file:/dev/lp0)

   Serial Port の場合

   通信速度

   パリティ

   データビット

   フロー制御

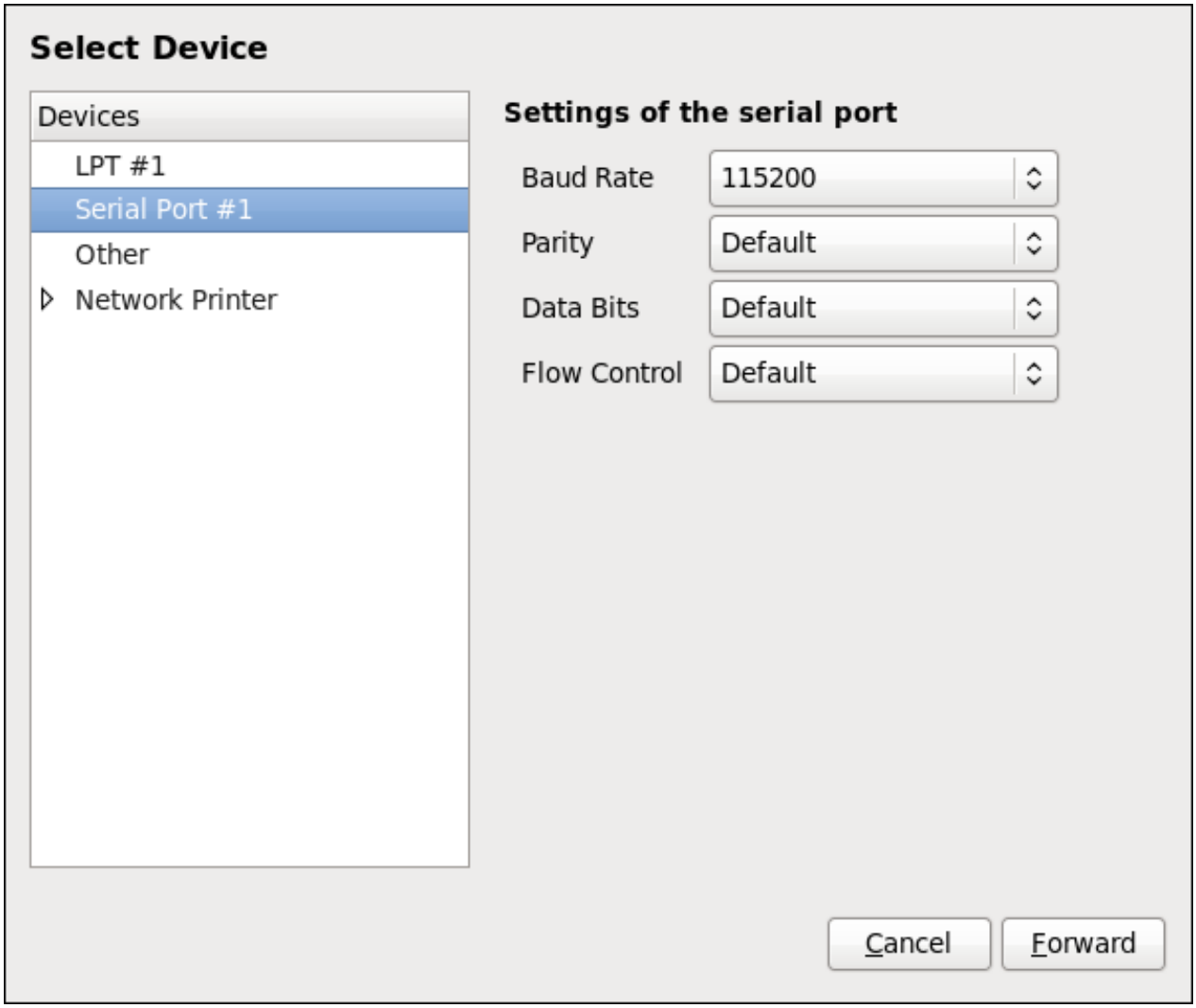


図12.4 ローカルプリンターの追加

4. **進む (Forward)** をクリックします。
5. プリンターのモデルを選択します。詳細は「プリンターモデルの選択と完了」を参照して下さい。

12.3.4. AppSocket/HP JetDirect プリンターの追加

以下の手順にしたがって AppSocket/HP JetDirect プリンターを追加します。

1. **新規プリンター** ダイアログを開きます (「プリンター設定ツールの起動」を参照)。
2. 左側の一覧で **ネットワークプリンター** → **AppSocket/HP JetDirect** の順に選択します。
3. 右側で、接続設定を入力します。

   ホスト名

   プリンターのホスト名または IP アドレス

   ポート番号

   印刷ジョブをリッスンするプリンターポート (デフォルトは **9100**)

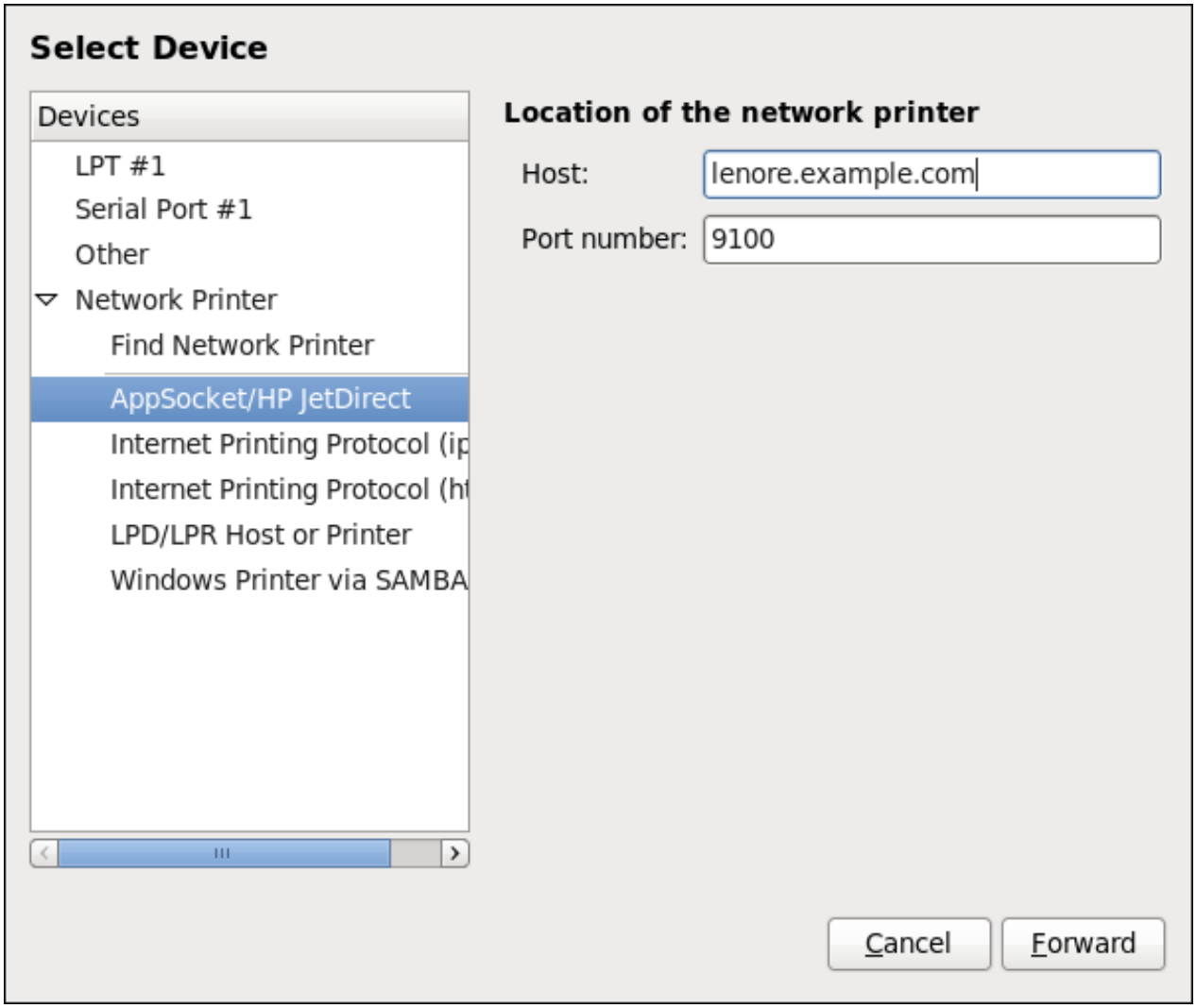


図12.5 JetDirect プリンターの追加

4. **進む (Forward)** をクリックします。
5. プリンターのモデルを選択します。詳細は「プリンターモデルの選択と完了」を参照して下さい。

12.3.5. IPP プリンターの追加

IPP プリンターは同じ TCP/IP ネットワーク上の異なるシステムにつながっているプリンターです。このプリンターが取り付けられているシステムは CUPS を実行しているか、単に IPP を使用するよう設定されているだけです。

プリンターサーバーでファイアウォールが有効な場合は、ポート 631 で受信 TCP 接続が可能なようにファイアウォールを設定する必要があります。プロトコルを参照する CUPS により、クライアントマシンは共有 CUPS キューを自動的に検出することが可能です。これを有効にするには、クライアントマシンのファイアウォールをポート 631 で受信 UDP パッケージを許可するよう設定する必要があります。

以下の手順にしたがって IPP プリンターを追加します。

1. **新規プリンター** ダイアログを開きます (「プリンター設定の開始」を参照)。
2. 左側のデバイスの一覧で、**ネットワークプリンター**、**Internet Printing Protocol (ipp)** または **Internet Printing Protocol (https)** の順に選択します。
3. 右側で、接続設定を入力します。

ホスト

IPP プリンターのホスト名

キュー

新規のキューに与えるキューの名前です(このボックスが空白のままであると、デバイスノードを基にした名前が使用されます)。

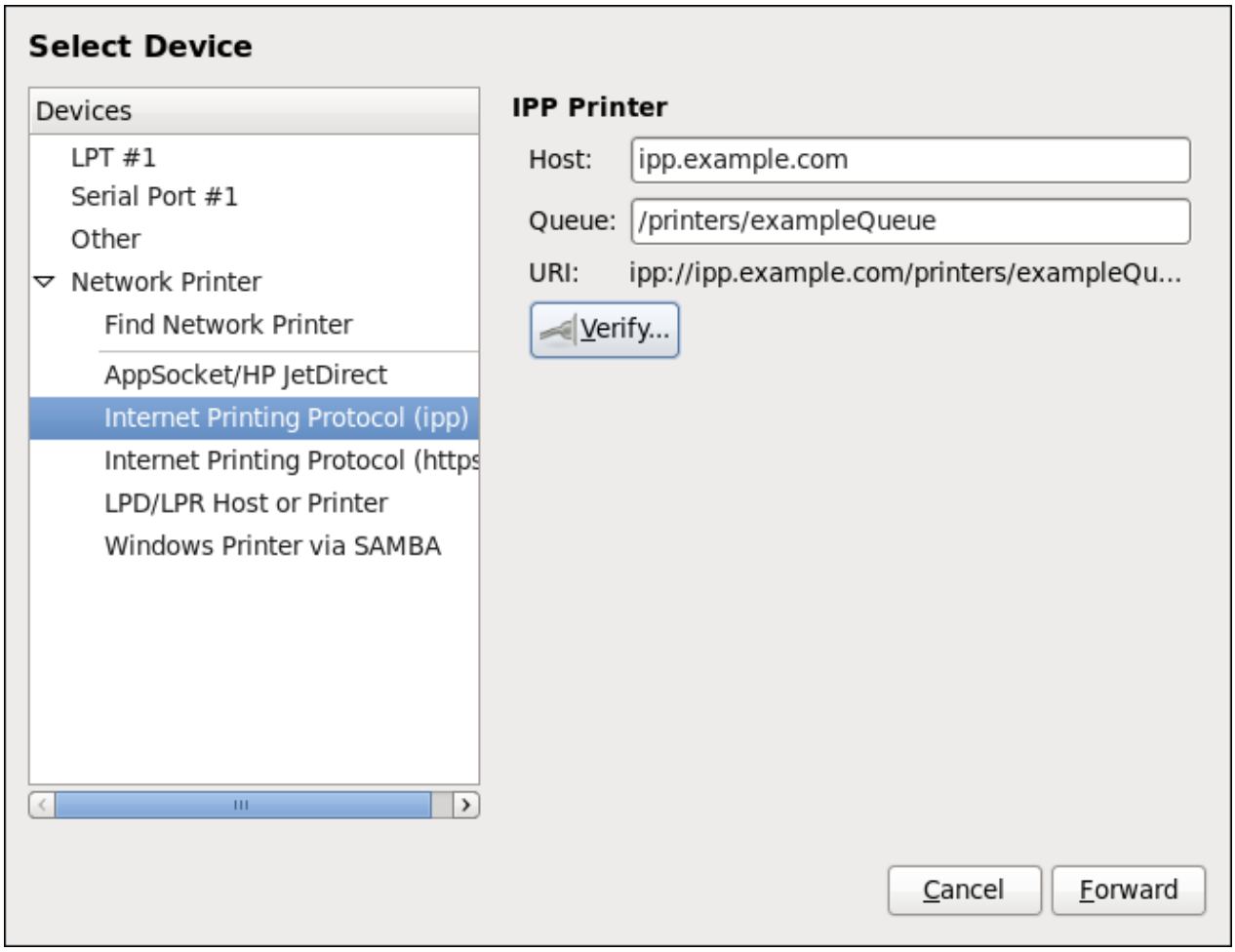


図12.6 IPP プリンターの追加

4. **進む** をクリックし、続けます。
5. プリンターのモデルを選択します。詳細は「プリンターモデルの選択と完了」を参照して下さい。

12.3.6. LPD/LPR Host or Printer の追加

以下の手順にしたがって LPD/LPR host or printer を追加します。

1. **新規プリンター** ダイアログを開きます (「プリンター設定の開始」を参照)。
2. 左のデバイス一覧から、**ネットワークプリンター** → **LPD/LPR Host or Printer** の順に選択します。
3. 右側で、接続設定を入力します。

ホスト

LPD/LPR printer or host のホスト名

オプションで **Probe** をクリックすると、LPD ホスト上のキューを検索できます。

キュー

新規のキューに与えるキューの名前です(このボックスが空白のままであると、デバイスノードを基にした名前が使用されます)。

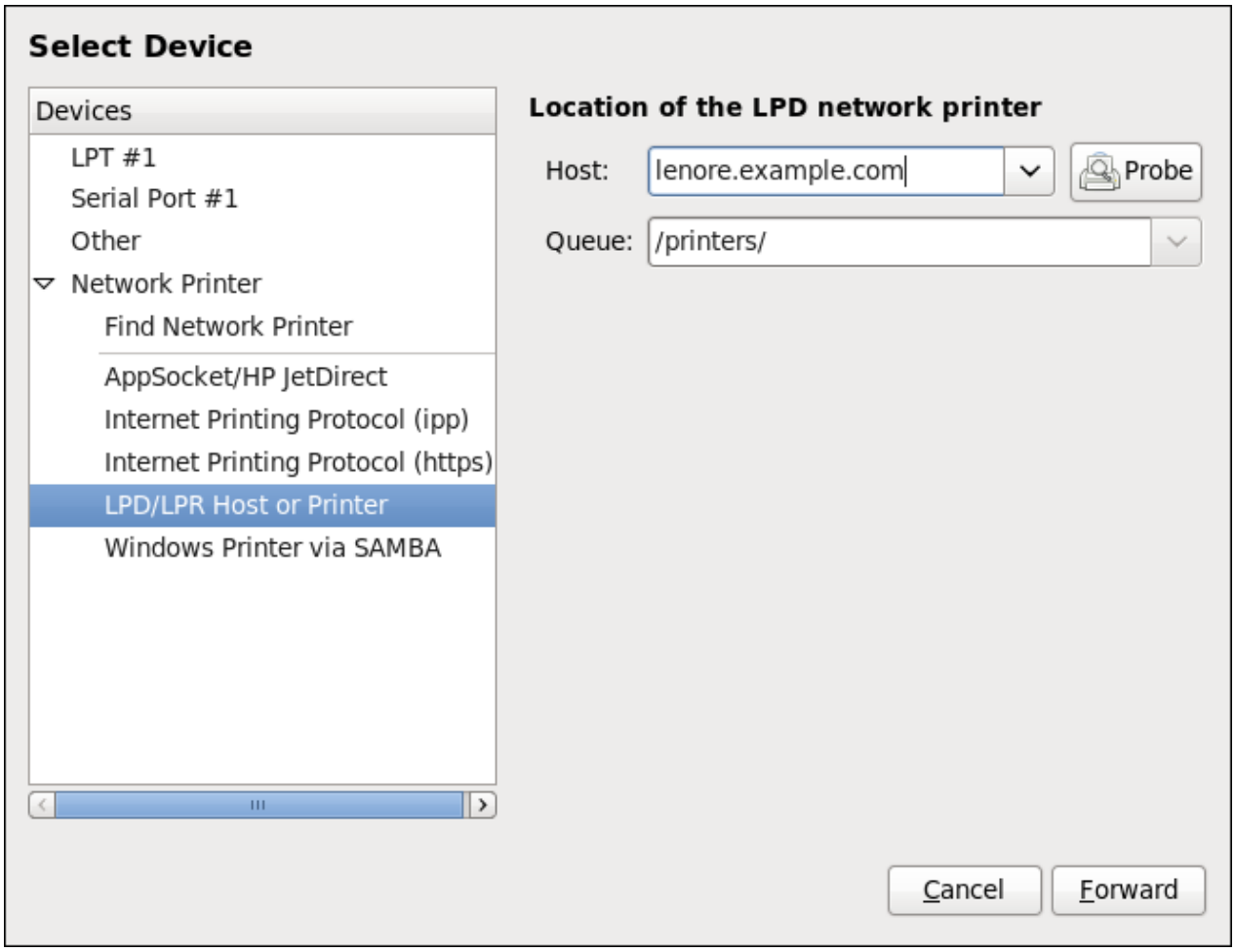


図12.7 LPD/LPR プリンターの追加

4. **進む** をクリックし、続けます。
5. プリンターのモデルを選択します。詳細は「プリンターモデルの選択と完了」を参照して下さい。

12.3.7. Samba (SMB) プリンターの追加

Samba プリンターを追加するには、以下の手順を実行します。

1. **新規プリンター** ダイアログを開きます (「プリンター設定の開始」を参照)。
2. 左側の一覧で、**ネットワークプリンター** → **SAMBA 経由の Windows プリンター**の順に選択します。
3. **smb://** フィールドに SMB アドレスを入力します。入力形式は*computer name/printer share* にします。図12.8「SMB プリンターの追加」では、*computer name* は **dellbox**、*printer share* は **r2** です。

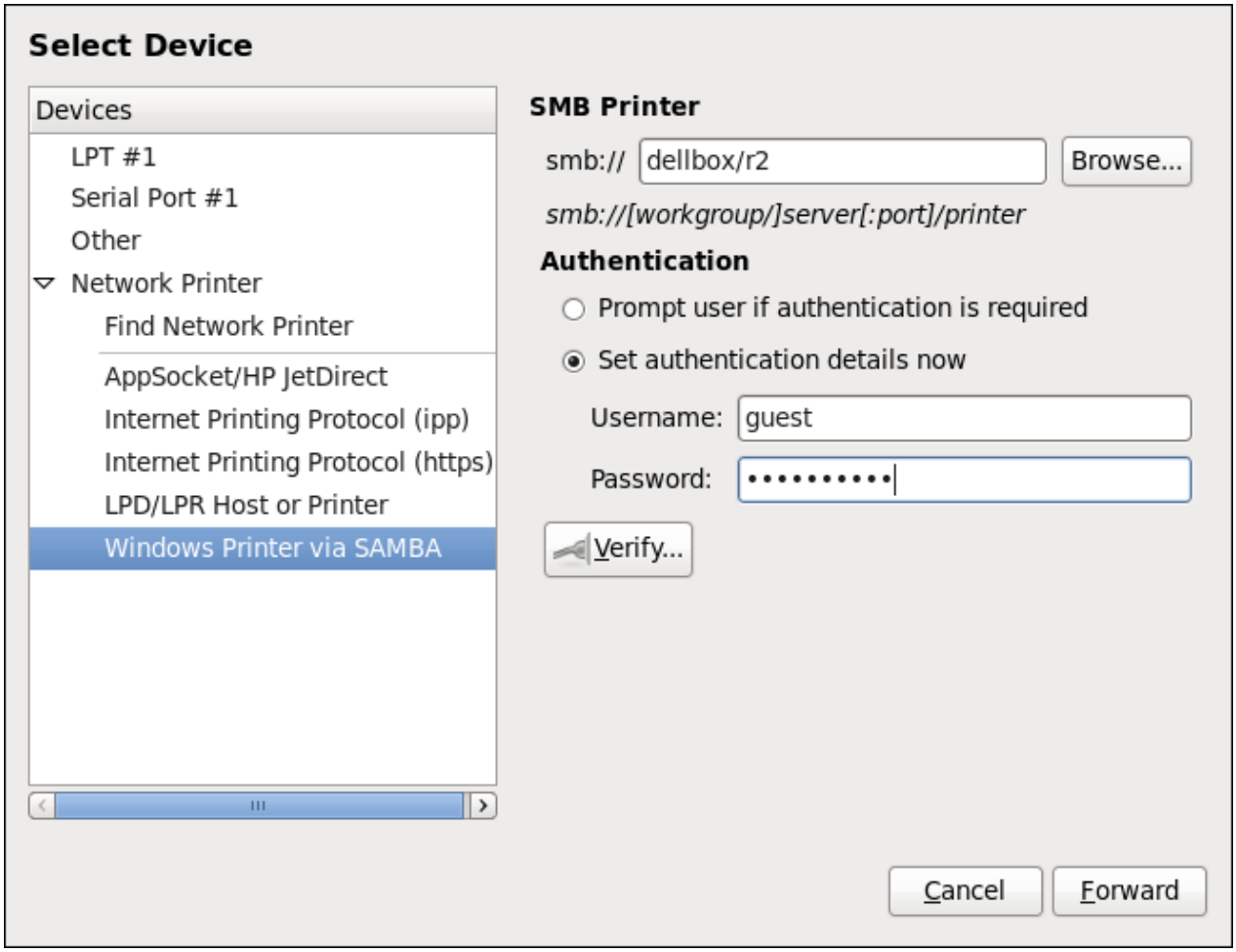


図12.8 SMB プリンターの追加

4. 利用できるワークグループやドメインを確認するには、**閲覧する (Browse)** をクリックします。特定のホストのキューだけを表示させるには、ホスト名 (NetBios 名) を入力して、**閲覧する** をクリックします。

5. 以下のオプションのどちらかを選択します。

   A. **認証が必要は場合にはユーザーに催促する (Prompt user if authentication is required)** を選択すると、文書を印刷する時にはユーザーはユーザー名とパスワードを提供することになります。

   B. **認証の詳細を今設定します (Set authentication details now)** を選択すると、今認証情報を提供するため後では入力が不必要となります。**ユーザー名** フィールドで、ユーザー名を入力してプリンターにアクセスします。このユーザーは SMB システムで存在している必要があり、ユーザーはプリンターへのアクセス権限を持っている必要があります。デフォルトのユーザー名は通常 Windows サーバーでは **guest** または Samba サーバーでは **nobody** です。

6. **ユーザー名** フィールドに指定したユーザーの **パスワード** (必要な場合は) を入力します。

警告

Samba プリンターのユーザー名とパスワードはプリンターサーバーにルート及び lpd が読み取り可能な非暗号化ファイルとして保存されています。そのため、プリンターサーバーにルートアクセスを持つ他のユーザーは、 Samba プリンターへのアクセスに使用するユーザー名とパスワードを閲覧することができます。

Samba プリンターへアクセスするためのユーザー名とパスワードを選択する場合は、ローカルの Red Hat Enterprise Linux システムにアクセスする時に使用するパスワードとは異なるパスワードを選ぶことをお勧めします。

Samba 印刷サーバーで共有するファイルがある場合も、印刷キューで使用されるパスワードとは異なるパスワードを使用することが推奨されます。

7. **確認 (Verify)** をクリックし、接続をテストします。確認が成功すると、ダイアログボックスが表示され、プリンター共有のアクセスを確認します。

8. **進む (Forward)** をクリックします。

9. プリンターのモデルを選択します。詳細は 「プリンターモデルの選択と完了」 を参照して下さい。

12.3.8. プリンターモデルの選択と完了

正しくプリンターの接続タイプを選択すると、システムはドライバーを取得するよう試行します。プロセスが失敗した場合は、ドライバーリソースを手動で検索することができます。

以下の手順に従い、プリンタードライバーを設定してインストールを完了します。

1. 自動的にドライバーが検知され、ウィンドウが表示されます。以下のオプションのうちいずれかを選択します。

   A. **データベースからプリンターを選択 (Select a Printer from database)** — システムは **製造元** の一覧からご使用のプリンターの選択した製造元に基づきドライバーを選択します。ご使用のプリンターのモデルが一覧にない場合は、**Generic** を選択してください。

   B. **PPD ファイルを提供 (Provide PPD file)** — システムは備わっているポストスクリプトプリンタデスクリプション (PostScript Printer Description : PPD) を使用します。PPD ファイルは通常製造元が提供するプリンターに同梱されています。PPD ファイルが利用可能な場合は、このオプションを選択し、オプション詳細の下にあるブラウザバーを使用して、PPD ファイルを選択できます。

   C. **ダウンロードするプリンタードライバーを検出 (Search for a printer driver to download)** — 製造元とご使用のプリンターモデルを**製造元とモデル** フィールドに入力し、OpenPrinting.org で適切なパッケージを検索します。

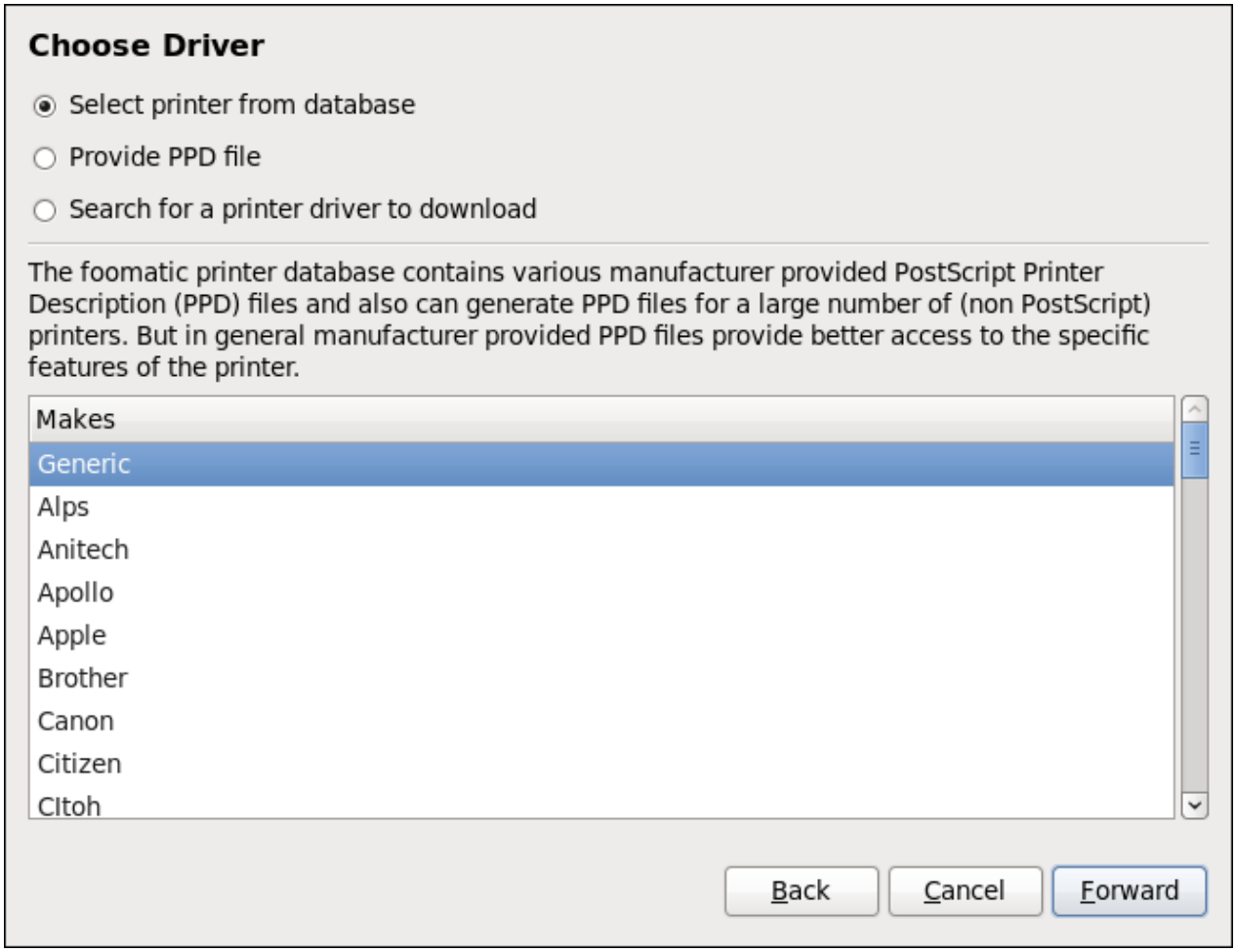


図12.9 プリンターブランドの選択

2. 以前に選択したものにより、以下に表示される詳細は異なります。
   - **データベースからプリンターを選択** オプションの場合は、プリンターブランドを表示
   - **PPD ファイルを提供** オプションの場合は、PPD ファイルの場所を表示
   - **ダウンロードするプリンタードライバーを検出** オプションの場合は、プリンターの製造元とモデルを表示
3. **進む** をクリックし、続けます。
4. 選択したオプションが該当する場合は、図12.10「プリンターモデルの選択」のようなウィンドウが表示されます。左側の **モデル** の列で該当するモデルを選択します。

注記

右側で、推奨される印刷ドライバーが自動的に選択されています。ただし、別の利用可能なドライバーを選ぶことも可能です。印刷ドライバーは、プリンターが理解できる形式で印刷したいデータを処理します。ローカルプリンターは直接ご使用のコンピュータにつながっているため、プリンターに送られるデータを処理するプリンタードライバーが必要になります。

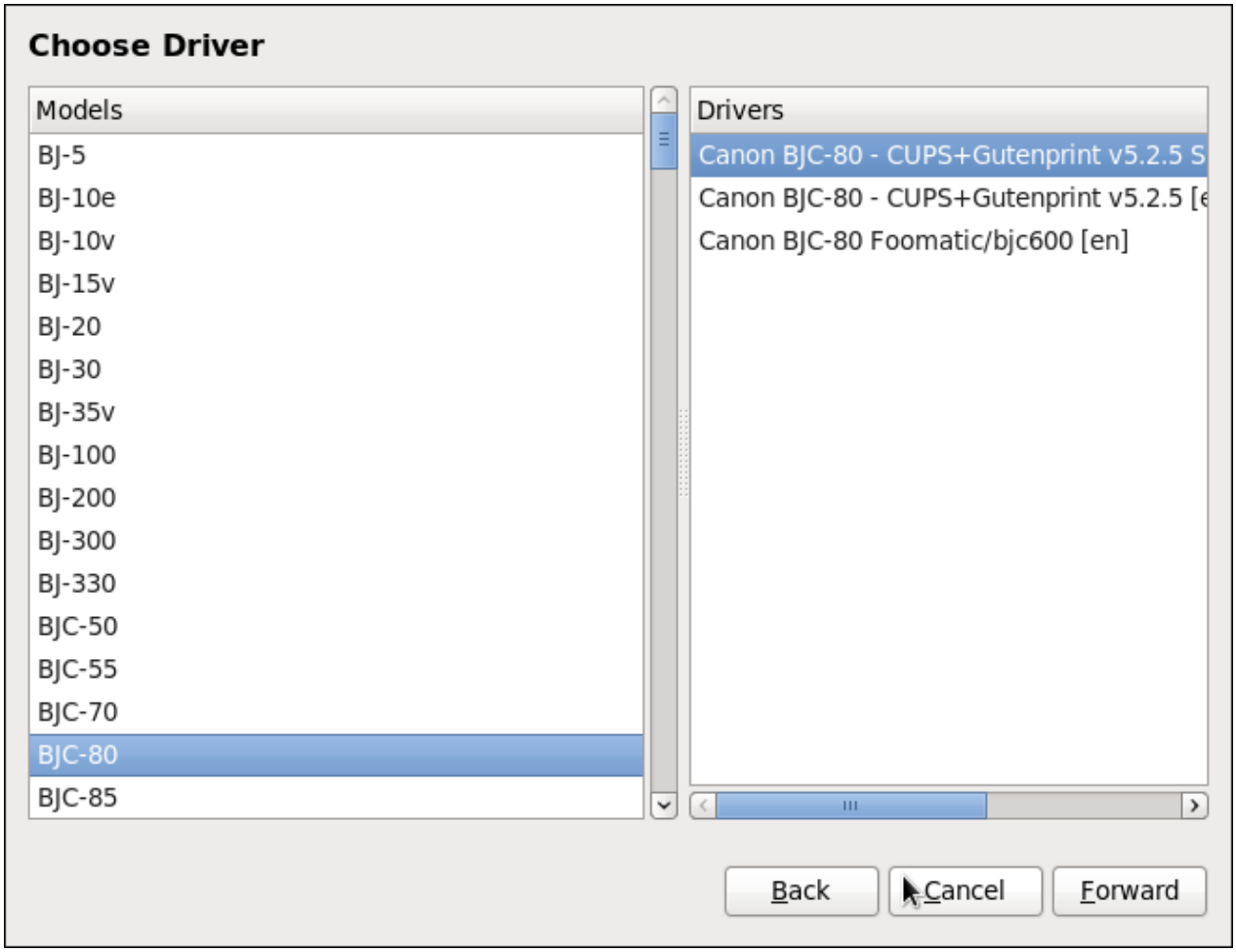


図12.10 プリンターモデルの選択

5. **進む (Forward)** をクリックします。

6. **プリンターの説明 (Describe Printer)** の下の、**プリンター名 (Printer Name)** フィールドに一意のプリンター名を入力します。名前には文字、数字、ダッシュ (-)、アンダースコア (_) を含むことができますが、スペースは含むことが **できません**。また、**説明 (Description)** と **場所 (Location)** フィールドに詳細情報を追加することも可能です。どちらもオプションで、スペースを入れることは可能です。

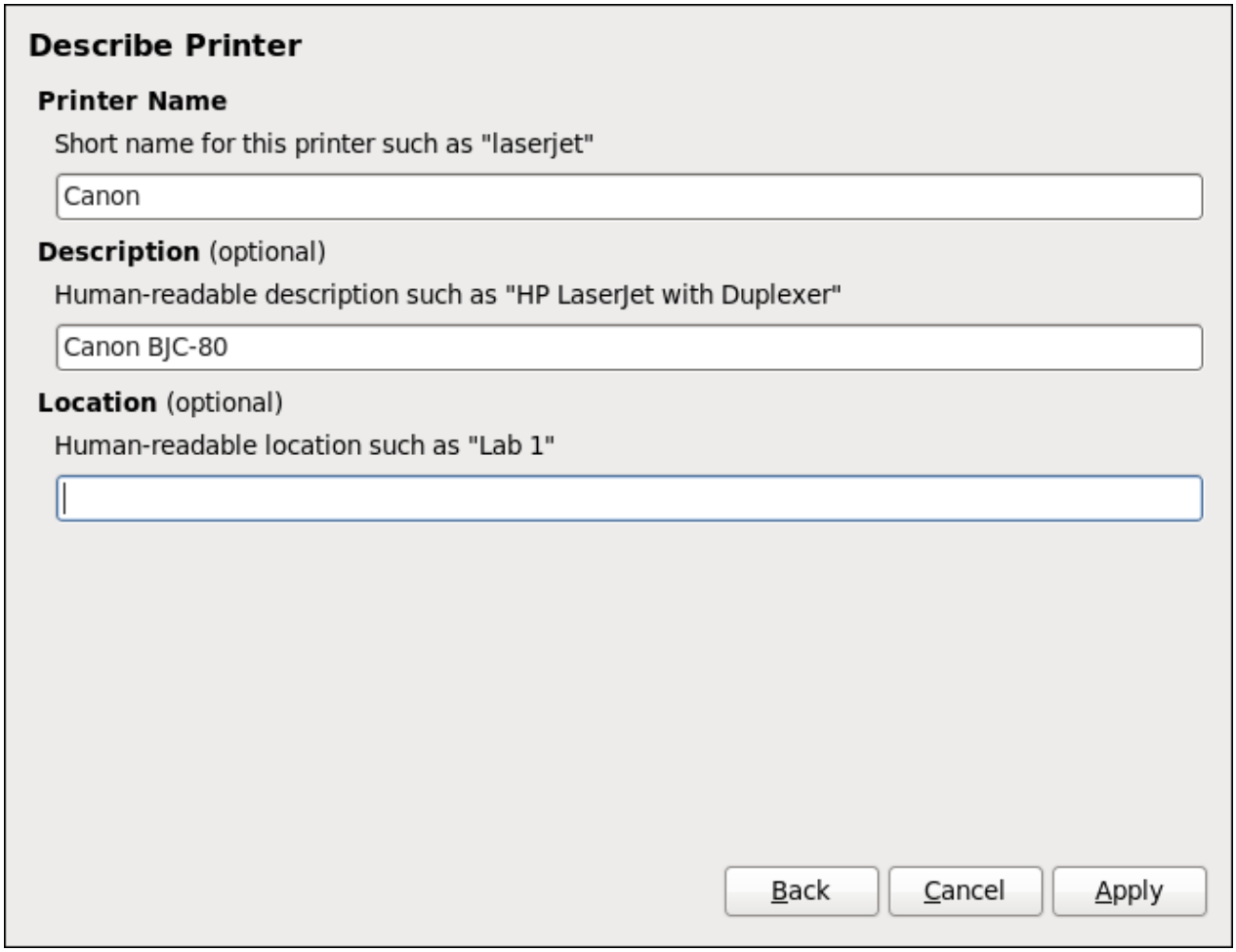


図12.11 プリンターの設定

7. 設定が正しければ、ご使用のプリンター設定を確認して、印刷キューを追加するために **適用 (Apply)** をクリックします。**戻る (Back)** をクリックすると、プリンター設定を変更できます。

8. 変更が適用されると、テストページの印刷を行うダイアログボックスが表示されます。**Yes** をクリックするとテストページが印刷されます。「テストページの印刷」を参照してテストページを後で印刷することもできます。

12.3.9. テストページの印刷

プリンターを設定、またはプリンターの設定を変更した後は、テストページを印刷して、プリンターが適切に機能していることを確認します。

1. **印刷 (Printing)** ウィンドウでプリンターを右クリックし、**プロパティ (Properties)** をクリックします。

2. プロパティーのウィンドウで、左側の **設定 (Settings)** をクリックします。

3. 表示されている **設定** タブで、**テストページの印刷 (Print Test Page)** ボタンをクリックします。

12.3.10. 既存プリンターの修正

既存のプリンターを削除するには、**プリンター設定** ウィンドウで該当するプリンターを選択し、**プリンター → 削除** の順に選択します。プリンターの削除を確認します。別の方法として、**Delete** キーを押しても削除できます。

デフォルトのプリンターを設定するには、プリンターの一覧で該当するプリンターを右クリックし、コンテキストメニューの **デフォルトに設定** ボタンをクリックします。

12.3.10.1. 設定のページ

プリンターのドライバー設定を変更するには、**プリンター** 一覧で該当する名前でダブルクリックします。そして、左側の **設定** ラベルをクリックし、**設定** ページを表示させます。

製造元やモデルなどのプリンター設定の変更、テストページの印刷、デバイスの場所 (URI) の変更など行うことができます。

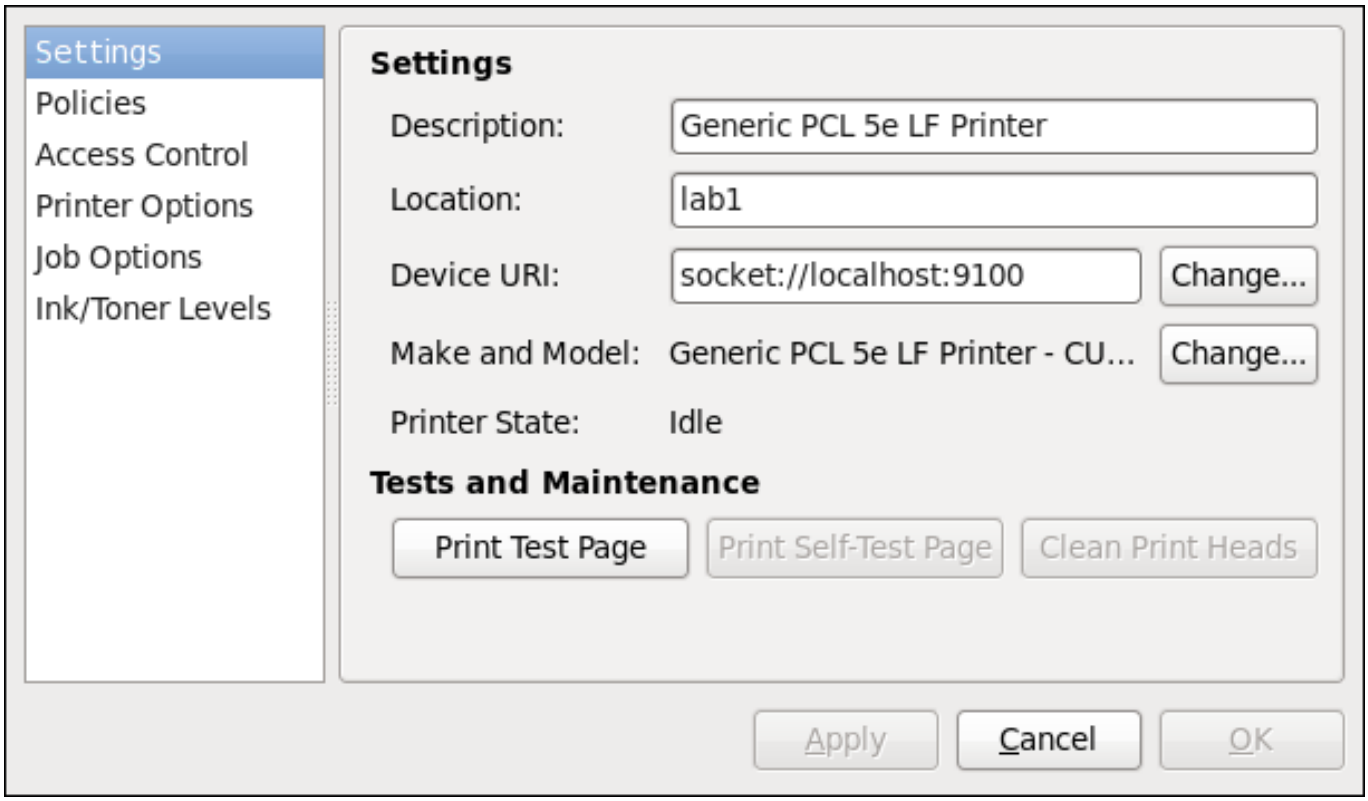


図12.12 設定のページ

12.3.10.2. ポリシーのページ

プリンターの状態や出力を変更するには、左側の **ポリシー (Policies)** ボタンをクリックします。

プリンターの状態を選択したり、プリンターの **エラーポリシー (Error Policy)** を設定することができます (エラーが発生した場合は、印刷ジョブを中止、再試行、停止できます)。

バナーページ (banner page) (送信元プリンター、ジョブを開始したユーザー名、印刷中の文書のセキュリティ状態など、印刷ジョブの特徴を説明するページ) の作成も可能です。**開始バナー (Starting Banner)** または **終了バナー (Ending Banner)** のドロップメニューをクリックし、印刷ジョブの性質に最適なオプションを選択します (**最上位秘密 (topsecret)**、**クラス分類 (classified)**、**秘密 (confidential)** など)。

12.3.10.2.1. プリンターの共有

ポリシー ページでは、プリンターを共有すると印を付けることができます。プリンターを共有にすると、ネットワーク上で公開されているユーザーは使用できます。プリンターの共有機能を有効にするには、**サーバー → 設定** の順にクリックし、**このシステムに接続されている共有プリンターを公開 (Publish shared printers connected to this system)** を選択します。

最後に、ファイアウォールがポート 631 への受信 TCP 接続 (system-config-firewall のネットワーク印刷サーバー) を許可していることを確認します。

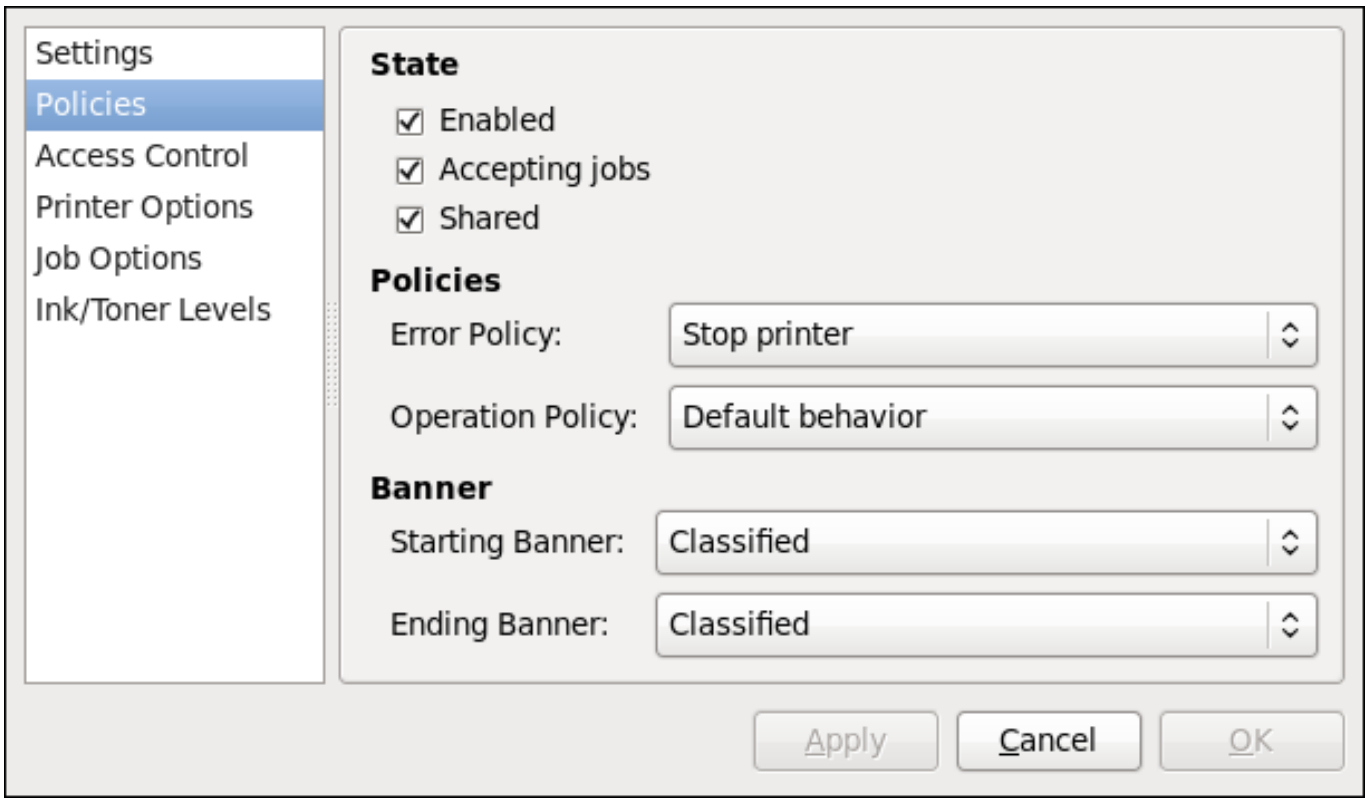


図12.13 ポリシーのページ

12.3.10.2.2. アクセス制御のページ

アクセス制御 ページで設定したプリンターへのユーザーレベルのアクセスを変更できます。左側の**アクセス制御 (Access Control)** のラベルをクリックするとページが表示されます。**次のユーザーを除いて全員に印刷を許可する (Allow printing for everyone except these users)** または **次のユーザー以外の印刷を拒否する (Deny printing for everyone except these users)** のどちらかを選択し、以下のようにユーザーセットを定義します。テキストボックスにユーザー名を入力し、**追加 (Add)** ボタンをクリックしてユーザーセットにユーザーを追加します。

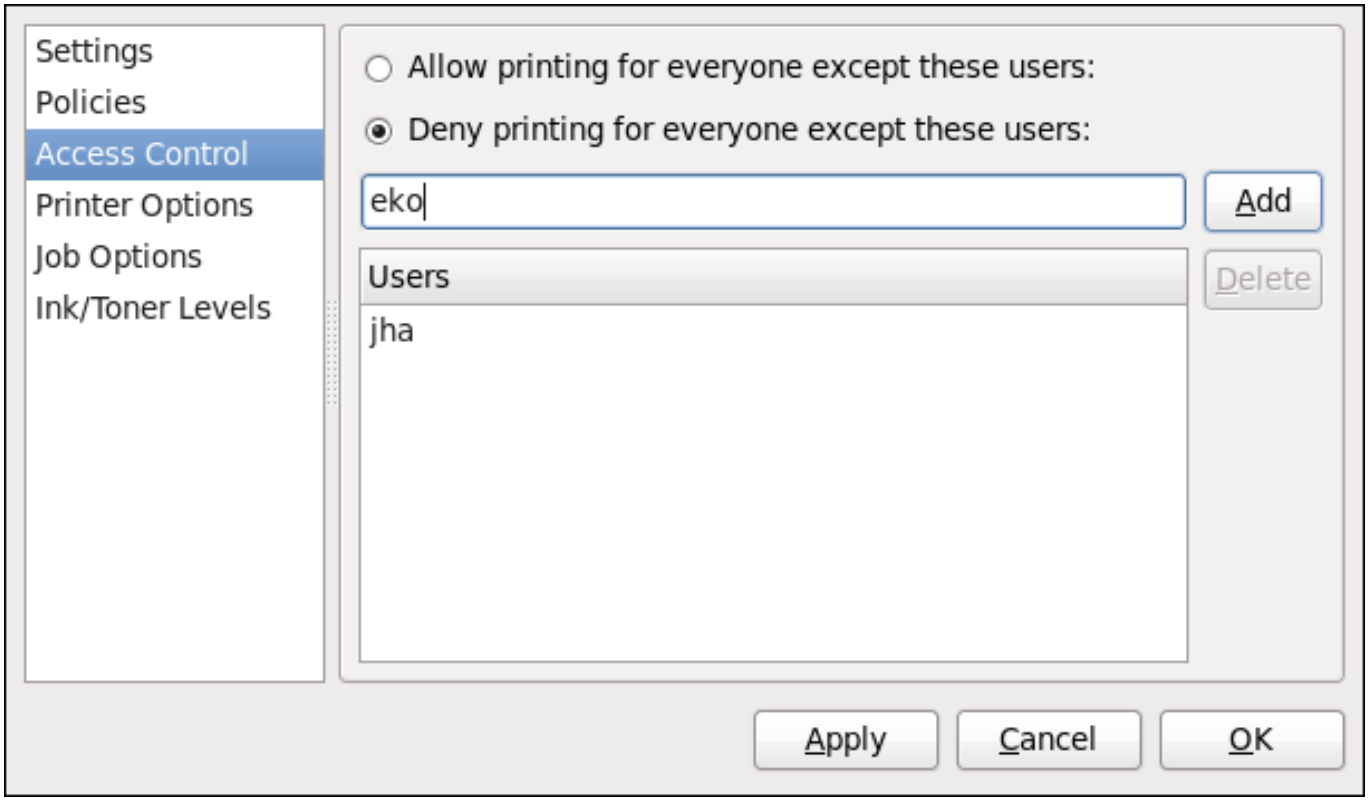


図12.14 アクセス制御のページ

12.3.10.2.3. プリンターオプションのページ

プリンターオプション (Printer Options) ページにはプリンターのメディアや出力の様々な設定オプションがあります。内容はプリンターごとに異なる場合があります。一般的な印刷の用紙、品質、サイズ設定が含まれます。

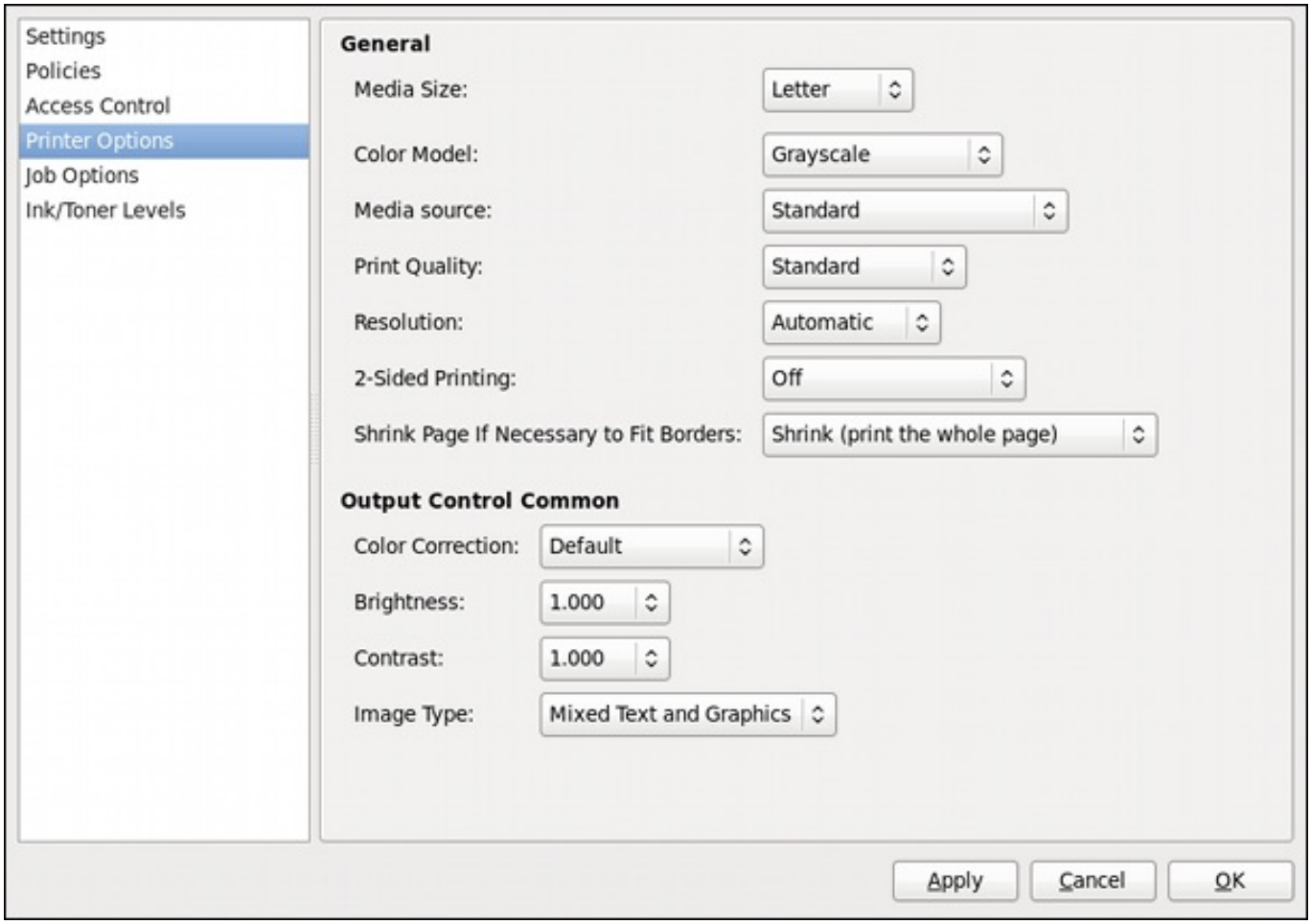


図12.15 プリンターオプションのページ

12.3.10.2.4. 依頼のオプションのページ

依頼のオプション (Job Options) ページでは、プリンターのジョブのオプションを詳細設定できます。左側の **依頼のオプション** をクリックすると、ページが表示されます。デフォルト設定を編集し、部数、印刷の向き、スライドごとのページ、拡大縮小 (印刷可能領域のサイズを拡大または縮小する。この機能によりサイズが印刷領域を超えるものを印刷媒体である用紙に合うようにします)、テキストオプションなど、カスタムの依頼のオプションを適用します。

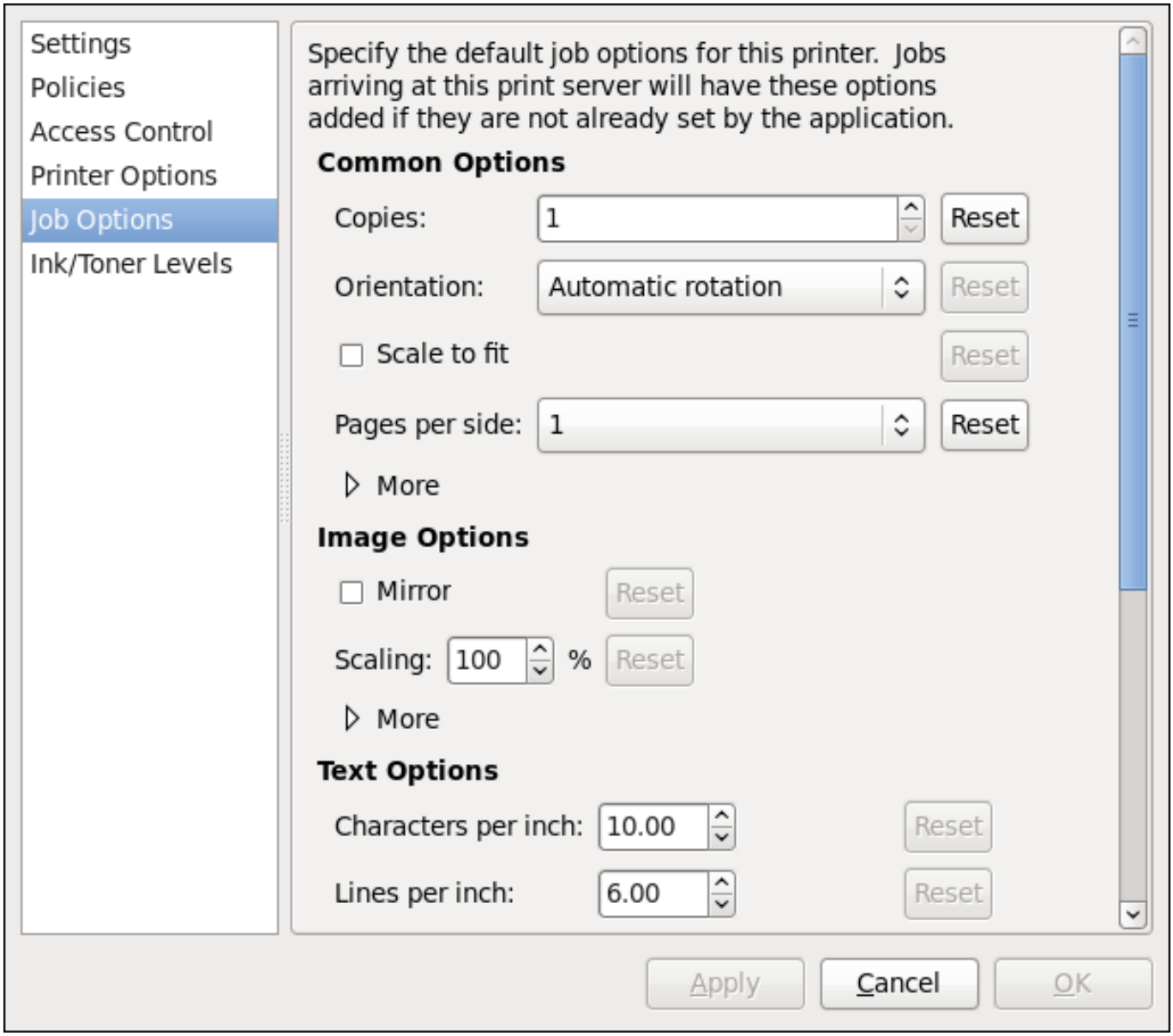


図12.16 依頼のオプションのページ

12.3.10.2.5. インク/トナーのレベルのページ

インク/トナーのレベル (Ink/Toner Levels) ページには該当する場合はトナーの状態の詳細、及びプリンターの状態のメッセージが表示されます。左側の **インク/トナーレベル** のラベルをクリックすると、ページが表示されます。

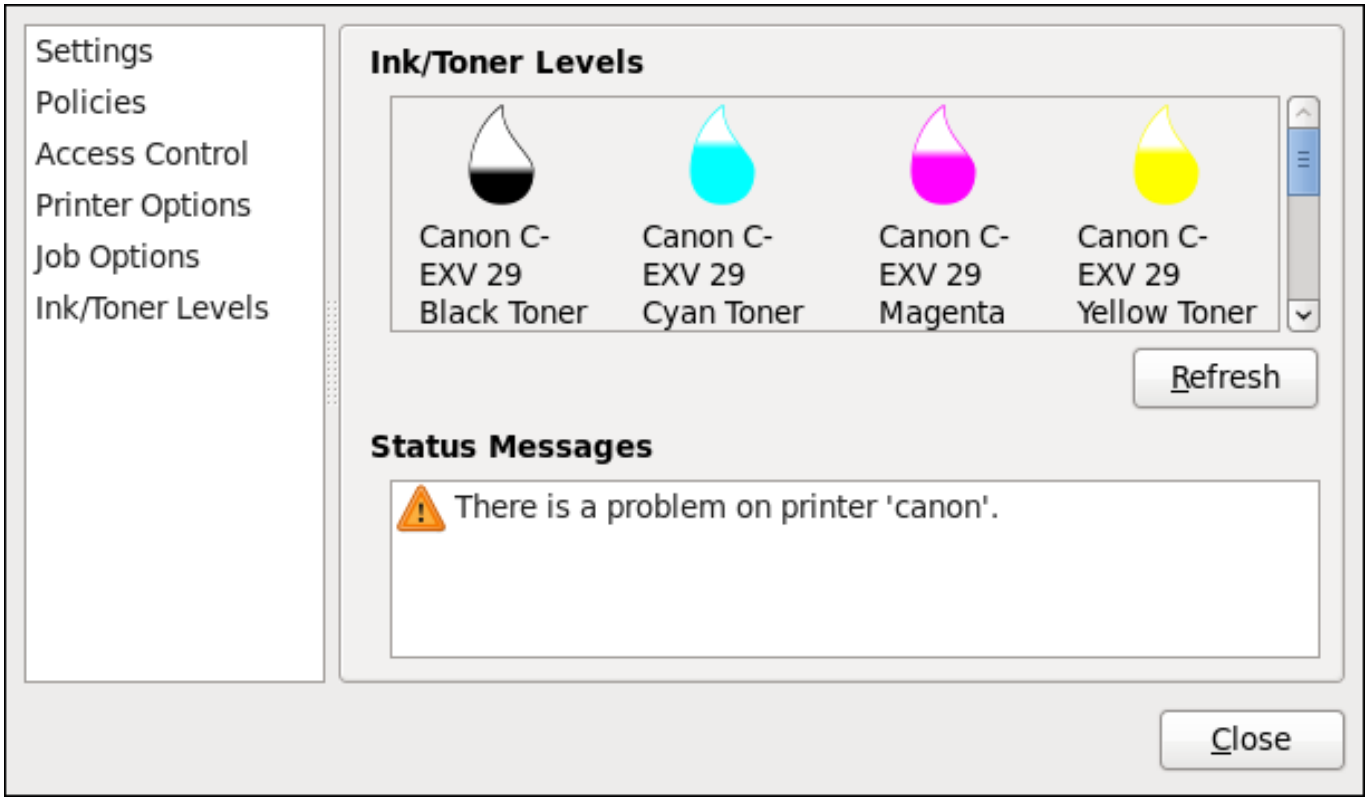


図12.17 インク/トナーのレベルのページ

12.3.10.3. 印刷ジョブの管理

Emacs からのテキストファイルの印刷または**GIMP** からの画像の印刷などプリンターデーモンに印刷ジョブを送ると、印刷ジョブは印刷のスプールキューに追加されます。印刷のスプールキューはプリンターに送られた印刷ジョブの一覧で、各印刷要求に関する情報、例えば印刷要求の状態、ジョブ番号などを表示します。

印刷が処理されている間、**プリンター状態**のアイコンがパネルの**通知スペース** に表示されます。その**プリンター状態**をクリックすると、図12.18「GNOME 印刷の状態」 に似たウィンドウが表示され、印刷ジョブの状態を確認できます。

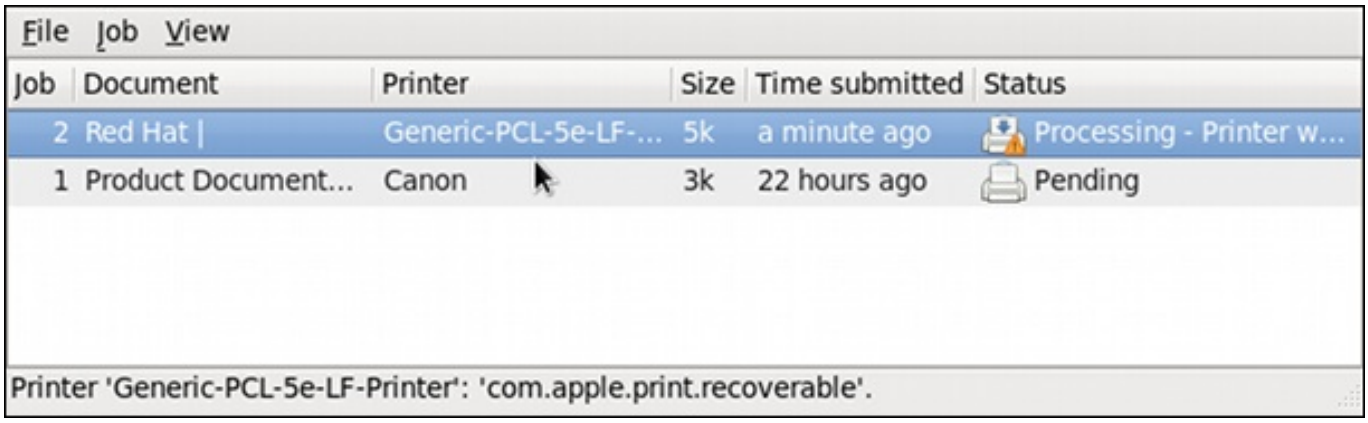


図12.18 GNOME 印刷の状態

印刷ジョブをキャンセル、保留、解除、再印刷、認証するためには、**GNOME 印刷の状態 (GNOME Print Status)** でジョブを選択し、**ジョブ (Job)** メニューでそれぞれコマンドをクリックします。

シェルプロンプトから印刷スプールの印刷ジョブの一覧を表示させるには、**lpstat -o** コマンドを入力します。最後の数行は以下のようになります。

例12.1 lpstat -o 出力の例

```
$ lpstat -o
Charlie-60              twaugh             1024   Tue 08 Feb 2011
16:42:11 GMT
Aaron-61                twaugh             1024   Tue 08 Feb 2011
16:42:44 GMT
Ben-62                  root               1024   Tue 08 Feb 2011
16:45:42 GMT
```

印刷ジョブをキャンセルしたい場合は、**lpstat -o** コマンドを使って依頼したジョブ番号を見つけます。その後 **cancel *ジョブ番号*** コマンドを使用します。例えば、**cancel 60** だと 例12.1「**lpstat -o** 出力の例」 の印刷ジョブを取り消します。他のユーザーによって開始された印刷ジョブを**cancel** コマンドでキャンセルすることはできません。ただし、**cancel -U root *job_number*** コマンドを使用すると、強制的にそうしたジョブを削除することは可能です。そうしたキャンセルを防ぐには、プリンターの操作ポリシーを **認証済** に変更することでルート認証を強制することができます。

シェルプロンプトから直接ファイルを印刷することもできます。例えば **lp sample.txt** コマンドはテキストファイル **sample.txt** を印刷します。印刷フィルターはファイルのタイプを決定し、プリンターが理解できる形式に変換します。

12.3.11. その他のリソース

Red Hat Enterprise Linux の印刷に関する詳細は、以下のリソースを参照して下さい。

12.3.11.1. インストールされているドキュメント

man lp

lpr コマンドの man ページです。コマンドラインからファイルを印刷することができます。

man cancel

印刷キューから印刷ジョブを削除するためのコマンドラインユーティリティの man ページです。

man mpage

1 枚の用紙に複数ページを印刷するためのコマンドラインユーザーの man ページです。

man cupsd

CUPS プリンターデーモンの man ページです。

man cupsd.conf

CUPS プリンターデーモンの設定ファイルの man ページです。

man classes.conf

CUPS のクラス設定ファイルの man ページです。

man lpstat

lpstat の man ページで、クラス、ジョブ、プリンターなどの状態情報を表示します。

12.3.11.2. 役立つ Web サイト

http://www.linuxprinting.org/

『GNU/Linux Printing』 には Linux の印刷に関する豊富な情報が記載されています。

http://www.cups.org/

CUPS のドキュメンテーション、FAQ、ニュースグループについて記載しています。

第13章 chrony スイートを使用した NTP 設定

IT では、多くの理由で正確な時間の維持が重要です。たとえばネットワーキングでは、正確なタイムスタンプがパケットとログで必要になります。Linux システムでは、**NTP** プロトコルがユーザースペースで実行しているデーモンにより実装されます。

ユーザースペースのデーモンは、カーネルで実行しているシステムクロックを更新します。システムクロックは、数々のクロックソースを利用することで時間を維持できます。通常使用されるのは、*タイムスタンプカウンター* (TSC) です。TSC は、最後にリセットされてからのサイクル数を数える CPU レジスタです。これは非常に高速で精度が高く、割り込みがありません。

ntpd と **chronyd** というデーモンの選択が可能で、これらはそれぞれ*ntp* と *chrony* のパッケージから利用できます。本セクションでは、従来の常にネットワーク接続されている専用サーバーのカテゴリーに当てはまらないシステム上のシステムクロックを、**chrony** スイートのユーティリティーを使って更新する方法を説明します。

13.1. chrony スイートの概要

Chrony は、ユーザースペースで実行するデーモンである**chronyd** と、**chronyd** を調整するコマンドラインプログラムである **chronyc** で構成されています。常時接続でない、もしくは常時電源が付いていないシステムでは、**ntpd** を使ったシステムクロックの調整は、比較的時間がかかります。これは、多くの小さな修正がクロックのずれと補正の観測に基づいてなされるためです。温度変化はシステムの電源を入れる際に大きくなる場合がありますが、これはハードウェアクロックの安定性に影響します。調節はシステムの起動のほんの数千秒以内に始まりますが、許容範囲の正確性は機器が温まっている場合の再起動では 10 秒、または要件や操作環境、およびハードウェアによっては数時間という幅の時間がかかる可能性もあります。**chrony** は、**ntpd** とは異なる **NTP** プロトコルの実装で、システムクロックをより迅速に調整することが可能です。

13.1.1. ntpd と chronyd の違い

ntpd と **chronyd** の大きな違いの 1 つは、コンピューターのクロックを管理するために使われるアルゴリズムにあります。**chronyd** が **ntpd** よりも優れている点は、以下のとおりです。

- **chronyd** は外部の時間参照が断続的にアクセス可能な場合でも機能しますが、**ntpd** が機能するには、規則的な時間参照のポーリングが必要になります。
- **chronyd** はネットワークの混雑が長時間にわたる場合でも機能します。
- **chronyd** のクロックの同期は通常より高速で、時間の正確性もより高いものです。
- **chronyd** は、水晶振動子の温度変化などによってクロックのレートが突然変更しても素早く適応します。一方、**ntpd** の場合は、落ち着くまでに長時間かかる場合があります。
- **chronyd** はデフォルト設定では、他の実行中のプログラムを乱さないように、クロックがシステム起動時に同期された後は、時間を更新しません。**ntpd** も時間を更新しないように設定できますが、クロックの調整に異なる手段を用いる必要があり、これは不利益となる面があります。
- **chronyd** は Linux 上のクロックのレートを幅広い範囲で調整できるため、たとえば仮想マシン上など、壊れたクロックもしくは不安定なクロックのあるマシン上でも操作が可能になります。

chronyd では可能で、**ntpd** ではできない点は、以下のとおりです。

- **chronyd** は、時間修正の方法が手動での入力のみという分離したネットワークのサポートを提供します。たとえば、管理者がクロックを見ている場合などです。**chronyd** は、異なる更新で修正されたエラーを見てコンピューターが時間を進めたり遅らせたりするレートを予測し、これを使ってコンピュータークロックを調整することができます。

- **chronyd** は、コンピューターの電源を切った際に時間を維持するハードウェアクロックを進めたり遅らせたりするレートを計算するためのサポートを提供します。システムが起動して、リアルタイムのクロックから取ってきた時間の調整された値を使ってシステム時間を設定する際に、このデータを使うことができます。このガイドの執筆時点でこれが可能なのは、Linux のみです。

ntpd では可能で **chronyd** ではできない点は、以下のとおりです。

- **ntpd** は完全に **NTP** バージョン 4 (『RFC 5905』) をサポートします。これには、ブロードキャスト、マルチキャスト、manycast のクライアントおよびサーバー、さらに孤立モードが含まれます。また、公開鍵の暗号化に基づく特別の認証スキームもサポートします (『RFC 5906』)。**chronyd** は **NTP** バージョン 3 (『RFC 1305』) を使用します。これは、バージョン 4 と互換性があります。

- **ntpd** には多くの参照クロック用のドライバーが含まれていますが、**chronyd** は **gpsd** などの他のプログラムに依存して参照クロックからデータにアクセスします。

13.1.2. NTP デーモンの選択

- **Chrony** は、頻繁にネットワーク接続が一時停止したり、断続的に切断され再接続されるようなシステムの場合に検討してください。たとえば、モバイルや仮想システムなどです。

- **NTP** デーモン (**ntpd**) は、通常、常時接続のシステムの場合に検討してください。ブロードキャストまたはマルチキャスト **IP** を使用する必要のあるシステム、または**Autokey** プロトコル でパケット認証を実行する必要のあるシステムの場合は、**ntpd** の使用を検討してください。**Chrony** は、MD5、SHA1、またはより強力なハッシュ機能のあるメッセージ認証コード (MAC) を使用した対称鍵認証のみをサポートしますが、**ntpd** は PKI システムの活用も可能な**Autokey** 認証プロトコルもサポートします。**Autokey** は 『RFC 5906』 で説明されています。

13.2. chrony の概要および設定

13.2.1. chronyd の概要

chrony デーモンである **chronyd** はユーザースペースで実行され、カーネルで実行しているシステムクロックに調整を加えます。これはネットワークのアクセスが可能な場合、常に **NTP** プロトコルを使って外部の時間ソースを調べることで実行されます。外部の照会が可能でない場合は、**chronyd** は誤差ファイルに保存されている最後に計算された誤差を使用します。または、**chronyc** で手動のコマンドで修正することも可能です。

13.2.2. chronyc の概要

chrony デーモンである **chronyd** は、コマンドラインユーティリティーの**chronyc** で制御できます。このユーティリティーはコマンドプロンプトを表示し、ここでは **chronyd** に変更を加える多くのコマンドが入力できます。デフォルト設定では、**chronyd** は **chronyc** のローカルインスタンスからのコマンドのみを受け付けるようになっていますが、**chronyc** は設定を変更して **chronyd** が外部の制御を受けるようにもできます。つまり、**chronyd** がリモート接続を受け入れるように設定すると、**chronyc** はリモートで実行できるようになります。**chronyd** への接続が許可される **IP** アドレスは、厳格に制御されるべきです。

13.2.3. chrony 設定コマンドの概要

chronyd のデフォルトの設定ファイルは、**/etc/chrony.conf** です。**-f** オプションを使うと、別の設定ファイルパスを指定することができます。その他のオプションについては、**chronyd** man ページを参照してください。使用可能な指示文の全一覧は、『http://chrony.tuxfamily.org/manual.html#Configuration-file』 を参照してください。以下では、設定オプションのいくつかを紹介します。

コメント

コメントの前には、#、%、;、または ! を付けます。

allow

オプションとして、**NTP** サーバーとして動作しているマシンへの接続が許可される**NTP** の接続元となるホスト、サブネット、またはネットワークを指定します。デフォルトでは、接続は許可されません。

例:

1. ```
   allow server1.example.com
   ```

   この形式では、アクセスが許可される特定のホストをホスト名で指定します。

2. ```
   allow 192.0.2.0/24
   ```

   この形式では、アクセスが許可される特定のネットワークを指定します。

3. ```
   allow 2001:db8::/32
   ```

   この形式では、アクセスが許可される **IPv6** アドレスを指定します。

cmdallow

これは **allow** 指示文 (**allow** セクションを参照) と似ていますが、特定のサブネットやホストに (**NTP** クライアントアクセスではなく) 制御アクセスを許可します。(「制御アクセス」 とは、**chronyc** がこれらのホスト上で実行可能であり、このコンピューター上で**chronyd** に正常に接続できることを意味します。) 構文は同じです。また、 **cmdallow** all 指示文と動作が似通っている **cmddeny** all 指示文もあります。

dumpdir

chronyd の再起動にまたがる測定履歴を保存するディレクトリーへのパスです (この実行中にシステムクロックの動作が変更されていないことを想定しています)。この機能が (設定ファイルの **dumponexit** コマンド、もしくは**chronyc dump** コマンド経由で) 使用される場合は、**dumpdir** コマンドを使って測定履歴が保存されるディレクトリーを定義してください。

dumponexit

このコマンドが存在する場合は、プログラムの終了時に常に記録された時間ソースの測定履歴を **chronyd** が保存すべきということを示しています (上記の**dumpdir** コマンドを参照)。

local

local キーワードは、**chronyd** に現行の同期ソースがない場合でも、(クライアントがポーリングしているという観点で) リアルタイムに同期しているように見えるようにします。このオプションは通常、分離されたネットワーク上のコンピューターで使用され、ここではいくつかのコンピューターが相互に同期する必要があり、これが 「master」 となって手動入力でほぼリアルタイムと一致します。

コマンド例:

```
local stratum 10
```

10 という大きな値が示しているのは、参照しているクロックからのホップ数が非常に多いので時間の信頼性がかなり低い、ということです。言い換えると、参照クロックに最終的に同期してい
```

る別のコンピューターにアクセスのあるコンピューターは、確実に 10 よりも下の階層に存在することになります。このため、10 のように高い値を **local** コマンドに選ぶと、リアルサーバーの視認性があるクライアントにリークしたとしても、マシン自体の時間がリアルタイムと混乱することを防ぎます。

log

log コマンドは、特定情報がログ記録されることを意味します。以下のオプションを取ります。

measurements

このオプションは、生の **NTP** 測定と関連情報を **measurements.log** と呼ばれるファイルにログ記録します。

statistics

このオプションは、回帰処理についての情報を **statistics.log** と呼ばれるファイルにログ記録します。

tracking

このオプションは、システムが進むまたは遅れるレートの予測に対する変更、およびなされたスルーを **tracking.log** と呼ばれるファイルにログ記録します。

rtc

このオプションは、システムのリアルタイムクロックについての情報をログ記録します。

refclocks

このオプションは、生およびフィルター処理された参照クロックの測定を **refclocks.log** と呼ばれるファイルにログ記録します。

tempcomp

このオプションは、温度測定とシステムレートの補正を **tempcomp.log** と呼ばれるファイルにログ記録します。

ログファイルは、**logdir** コマンドが指定するディレクトリーに書き込まれます。コマンドの例は以下のようになります。

```
log measurements statistics tracking
```

logdir

この指示文は、ログファイルが書き込まれるディレクトリーを指定します。以下のようになります。

```
logdir /var/log/chrony
```

makestep

通常 **chronyd** は、必要に応じてクロックを早めたり遅らせたりして、システムが徐々に時間の片寄りを修正するようにします。場合によっては、システムクロックに大きな誤差が生じていて、このスループロセスがシステムクロックを修正するには長時間かかる場合があります。この指示文は、調整幅がしきい値よりも大きい場合に **chronyd** がシステムクロックを更新すること

を強制しますが、これは **chronyd** が起動してからのクロック更新回数が特定の制限を超えていない場合のみです (マイナス値を使うと、制限を無効にできます)。**initstepslew** 指示文は **NTP** ソースのみと機能することから、これは参照クロックを使用している場合に特に便利なものです。

この指示文の使用例は、以下のようになります。

```
makestep 1000 10
```

この場合、調整幅が 1000 秒よりも大きければシステムクロックは更新されますが、これは最初の 10 回のクロック更新のみです。

maxchange

この指示文は、クロック更新で修正される片寄りの最大値を設定します。指定された回数の更新の後でのみチェックが実行されることで、システムクロックの初回の調整が大幅なものにできます。指定された最大値よりも大きな片寄りが発生した場合、これは指定された回数無視され、その後に **chronyd** があきらめ、終了します (マイナス値を使うと、終了しないようになります)。どちらの場合も、メッセージは syslog に送信されます。

この指示文の使用例は、以下のようになります。

```
maxchange 1000 1 2
```

最初のクロック更新後に、**chronyd** がすべてのクロック更新で片寄りをチェックし、1000 秒よりも大きい調整を 2 回無視した後に、3 回目に終了します。

maxupdateskew

chronyd のタスクの 1 つは、コンピューターのクロックが参照ソースと比較してどれくらい早くまたは遅く動いているかを調べることです。さらに、予想される値のエラーの範囲を予測することです。エラーの範囲が広すぎる場合は、測定が落ち着いておらず、予想される進みまたは遅れのレートがあまり信頼性の高いものではないことを示すことになります。**maxupdateskew** パラメーターでは、予想の信頼性が高くないため使用すべきでないかどうかを判断するしきい値が使えるようになります。デフォルトでは、しきい値は 1000 ppm になります。構文は以下のようになります。

```
maxupdateskew skew-in-ppm
```

電話を使ったサーバーへのダイアルアップ接続の場合、通常の *skew-in-ppm* の値は 100 で、LAN 回線上のコンピューターの場合は 5 または 10 です。信頼性のない予測の使用に対する保護の方法は、これだけではないことに注意してください。**chronyd** は常に予想される進みまたは遅れのレートと、予測のエラー範囲を追跡しています。ソースの 1 つからの別の測定の後に新たな予測が生成されると、加重組み合わせのアルゴリズムを使ってマスター予測が更新されます。このため、**chronyd** に信頼性の高い既存のマスター予測があり、エラー範囲の広い新たな予測が生成されると、既存のマスター予測が新規のマスター予測を特徴づけることになります。

noclientlog

この指示文は引数を取らず、クライアントのアクセスをログ記録しないように指定します。通常、クライアントアクセスはログ記録され、**chronyc** でクライアントコマンドを使って統計数字が報告されます。

reselectdist

chronyd が利用可能なソースから同期ソースを選ぶ際には、同期距離が最短のものを優先します。しかし、同様の距離のソースが複数存在して再選択を繰り返すことを避けるため、現在選択されていないソースの距離に固定距離が追加されます。これを、**reselectdist** オプションで設定できます。デフォルトでは、この距離は 100 マイクロ秒になります。

構文は以下のようになります。

```
reselectdist dist-in-seconds
```

stratumweight

stratumweight 指示文は、**chronyd** が利用可能なソースから同期ソースを選択する際に、階層ごとに追加される同期距離を設定します。

構文は以下のようになります。

```
stratumweight dist-in-seconds
```

デフォルトでは、*dist-in-seconds* は 1 秒になります。通常これは、低い階層のソースの距離が高い階層のソースよりも大幅に遠くても、低い階層のソースが優先されることを意味します。**stratumweight** を 0 に設定すると、**chronyd** はソース選択の際に階層を無視します。

rtcfile

rtcfile 指示文は、システムのリアルタイムクロック (RTC) の正確性の追跡に関連するパラメーターを **chronyd** が保存することができるファイル名を定義します。構文は以下のようになります。

```
rtcfile /var/lib/chrony/rtc
```

chronyd は、このファイルが存在し、**writertc** コマンドが **chronyc** で発行されると、情報をこのファイルに保存します。保存される情報は、あるエポックでの RTC のエラー、そのエポック (1970 年 1 月 1 日以降の秒数)、および RTC の時間が早まるもしくは遅れるレートです。リアルタイムクロックのコードはシステム固有のものなので、すべてのリアルタイムクロックがサポートされるわけではありません。この指示文を使用する場合、リアルタイムクロックは手動で調整しないでください。リアルタイムクロックがランダムな間隔で調整されると、その片寄りのレートを測定する **chrony** の必要性が干渉されるためです。

rtcsync

rtcsync 指示文は、デフォルトで **/etc/chrony.conf** ファイルにあります。これは、カーネルにシステムクロックが同期されていることを知らせ、カーネルはリアルタイムクロックを 11 分ごとに更新します。

13.2.4. chronyc のセキュリティー

chronyc にアクセスできると **chronyd** を設定ファイルの編集のように変更できるので、**chronyc** へのアクセスは制限されるべきです。パスワードを ASCII または HEX でキーファイルで指定して、**chronyc** の使用を制限することができます。エントリーの 1 つは、操作コマンドの使用を制限するために使われ、コマンドキーとして参照されます。デフォルト設定では、開始時にランダムコマンドキーが自動的に生成されます。手動で指定したり変更したりする必要はありません。

キーファイル内の他のエントリーは、リモートの **NTP** サーバーもしくはピアから受信したパケットを認証する **NTP** キーとして使用できます。両方のサイドでキーファイル内の同一の ID、ハッシュタイプ、およびパスワードのあるキーを共有する必要があります。これには、手動でキーを作成し、**SSH** などの安全な媒体を用いてコピーする必要があります。たとえば、キー ID が 10 の場合、クライアントとして動作するシス

テムは、以下の形式の行を設定ファイルに含める必要があります。

```
server w.x.y.z key 10
peer w.x.y.z key 10
```

キーファイルの場所は、**/etc/chrony.conf** ファイルで指定されます。設定ファイルのデフォルトのエントリーは、以下のようになります。

```
keyfile /etc/chrony.keys
```

コマンドキー番号は、**commandkey** 指示文を使って **/etc/chrony.conf** で指定されます。これは、**chronyd** がユーザーコマンドの認証に使用するキーです。設定ファイル内の指示文は、以下の形式になります。

```
commandkey 1
```

キーファイル **/etc/chrony.keys** 内のデフォルトのエントリーの形式は、コマンドキーの場合、以下のようになります。

```
1 SHA1 HEX:A6CFC50C9C93AB6E5A19754C246242FC5471BCDF
```

ここでの **1** はキー ID で、SHA1 は使用するハッシュ機能、**HEX** はキーの形式、そして **A6CFC50C9C93AB6E5A19754C246242FC5471BCDF** は **chronyd** の初回起動時にランダムに生成されるキーになります。キーは 16進数もしくは ASCII 形式 (デフォルト) で生成されます。

特定の **NTP** サーバーもしくはピアからのパケット認証に使用するキーファイル内の手動でのエントリーは、以下のように簡単なものです。

```
20 foobar
```

ここでの **20** はキー ID で、**foobar** は秘密の認証キーになります。デフォルトのハッシュは MD5 で、ASCII がキーのデフォルトのフォーマットになります。

デフォルトでは、**chronyd** はポート **323** 上の **localhost** (**127.0.0.1** and **::1**) からのコマンドのみを待機する設定になっています。**chronyc** を使って **chronyd** にリモートでアクセスするには、**/etc/chrony.conf** ファイル内の **bindcmdaddress** 指示文すべてを削除して、すべてのインターフェイスの待機を有効にし、**cmdallow** 指示文を使ってリモートの **IP** アドレス、ネットワーク、またはサブネットからのコマンドを許可するようにします。また、リモートシステムから接続するために、ポート **323** をファイアウォールでオープンにする必要があります。**allow** 指示文は **NTP** アクセスのためであり、**cmdallow** 指示文はリモートのコマンド受信を有効にするものであることに注意してください。これらの変更は、**chronyc** をローカルで実行することで、一時的なものにすることができます。変更を永続的なものにするには、設定ファイルを編集してください。

chronyc と **chronyd** の通信は **UDP** で行われるので、操作コマンドを発行する前に権限を与える必要があります。これを行うには、**authhash** および **password** のコマンドを以下のように使用します。

```
chronyc> authhash SHA1
chronyc> password HEX:A6CFC50C9C93AB6E5A19754C246242FC5471BCDF
200 OK
```

chronyc を使ってローカルの **chronyd** を設定する場合は、**-a** オプションを使うと **authhash** と **password** のコマンドが自動的に実行されます。

パスワードを提供せずに使用できるコマンドは、以下のものに限られます。 **activity**, **authhash**, **dns**, **exit**, **help**, **password**, **quit**, **rtcdata**, **sources**, **sourcestats**, **tracking**, **waitsync** 。

13.3. chrony の使用

13.3.1. chrony のインストールを確認する

chrony がインストールされていることを確認するには、**root** で以下のコマンドを実行します。

```
~]# yum install chrony
```

chrony デーモンのデフォルトの場所は、**/usr/sbin/chronyd** です。コマンドラインユーティリティーは、**/usr/bin/chronyc** にインストールされます。

13.3.2. chrony のインストール

chrony をインストールするには、**root** で以下のコマンドを実行します。

```
~]# yum install chrony -y
```

chrony デーモンのデフォルトの場所は、**/usr/sbin/chronyd** です。コマンドラインユーティリティーは、**/usr/bin/chronyc** にインストールされます。

13.3.3. chronyd ステータスの確認

chronyd のステータスを確認するには、以下のコマンドを発行します。

```
~]$ systemctl status chronyd
chronyd.service - NTP client/server
   Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled)
   Active: active (running) since Wed 2013-06-12 22:23:16 CEST; 11h ago
```

13.3.4. chronyd の起動

chronyd を起動するには、**root** で以下のコマンドを発行します。

```
~]# systemctl start chronyd
```

chronyd をシステム起動時に自動的に起動するには、**root** で以下のコマンドを発行します。

```
~]# systemctl enable chronyd
```

13.3.5. chronyd の停止

chronyd を停止するには、**root** で以下のコマンドを発行します。

```
~]# systemctl stop chronyd
```

chronyd がシステム起動時に自動的に起動しないようにするには、**root** で以下のコマンドを発行します。

```
~]# systemctl disable chronyd
```

13.3.6. chrony の同期を確認する

chrony が同期しているかどうかを確認するには、**tracking**、**sources**、および **sourcestats** のコマンドを使用します。

13.3.6.1. chrony トラッキングの確認

chrony のトラッキングを確認するには、以下のコマンドを発行します。

```
~]$ chronyc tracking
Reference ID    : 1.2.3.4 (a.b.c)
Stratum         : 3
Ref time (UTC)  : Fri Feb  3 15:00:29 2012
System time     : 0.000001501 seconds slow of NTP time
Last offset     : -0.000001632 seconds
RMS offset      : 0.000002360 seconds
Frequency       : 331.898 ppm fast
Residual freq   : 0.004 ppm
Skew            : 0.154 ppm
Root delay      : 0.373169 seconds
Root dispersion : 0.024780 seconds
Update interval : 64.2 seconds
Leap status     : Normal
```

各フィールドは、以下のとおりです。

Reference ID

入手可能な場合、コンピューターが現在同期しているサーバーの参照 ID および名前 (または **IP** アドレス) です。これが**127.127.1.1** の場合はコンピューターが外部ソースと同期しておらず、「local」 モードで稼働していることを意味します (**chronyc** の local コマンド、または **/etc/chrony.conf** ファイル内の **local** 経由 (**local** のセクションを参照))。

Stratum

参照クロックのあるコンピューターから何ホップ離れているかを示します。このようなコンピューターは stratum-1 コンピューターなので、上記の例のコンピューターは 2 ホップ離れていることになります (つまり、a.b.c が stratum-2 で、stratum-1 から同期しています)。

Ref time

参照ソースからの最後の測定が処理された時間 (UTC) です。

System time

通常の操作では、**chronyd** はシステムクロックを更新しません。これは、タイムスケールにおけるジャンプはいかなるものでも特定のアプリケーションプログラムに有害な結果をもたらすためです。代わりに、システムクロックのエラーはこれをわずかに早めたり遅くしたりしてエラーがなくなるまで修正します。その後に、システムクロックを通常のスピードに戻します。その結果、(**gettimeofday()** システムコールを使用した他のプログラム、またはシェルの日付コマンドが読み取る) システムクロックが **chronyd** が予測する現在の実際の時間 (サーバーモードで稼働している場合、これを **NTP** クライアントに報告) と異なることになります。この行で報告される値は、この効果による差異です。

Last offset

最後のクロック更新におけるローカルオフセットの予測です。

RMS offset

オフセット値の長期の平均です。

Frequency

「frequency」 は、**chronyd** が修正しない場合にシステムクロックが間違うレートです。これは、ppm (100 万分の 1) で表されます。たとえば、1ppm という値が意味するのは、システムクロックが 1 秒進んだと考えると、実際の時間と比較して 1.000001 秒進んでいるということです。

Residual freq

現在選択されている参照ソースの 「residual frequency」 を示します。これは、参照ソースからの測定が示すあるべき周波数と現在使用されている周波数の差異を反映しています。これが常にゼロとは限らない理由は、補正する手順が周波数に提供されているためです。参照ソースからの測定が取得され、新たな剰余周波数が計算されるごとに、この剰余の予測される正確性は既存の周波数の値の予測される正確性 (次の **skew** を参照) と比較されます。新たな剰余の加重平均は、加重がこれらの正確性に依存して計算されます。参照ソースからの測定に一貫した傾向がある場合、剰余は時間をかけてゼロにされます。

Skew

周波数の予測されるエラー範囲です。

Root delay

コンピューターが究極的に同期される stratum-1 コンピューターへのネットワークパスの遅延の合計です。極端な状況では、この値はマイナスになる場合もあります。(コンピューターの周波数が相互に追跡しておらず、ネットワークの遅れが各コンピューターでの応答時間に比べて非常に短いという対称的なピア配置でこれが発生する可能性があります。)

Root dispersion

コンピューターが究極的に同期される stratum-1 コンピューターにすべてのコンピューターで累積された分散の合計です。分散はシステムクロックの解像度、統計数字の測定変動などに起因します。

Leap status

Leap のステータスで、Normal、Insert second、Delete second、または Not synchronized のいずれかになります。

13.3.6.2. chrony ソースの確認

ソースコマンドは、**chronyd** がアクセスしている現在の時間ソースについての情報を表示します。オプションの引数 -v を指定すると、詳細情報になります。この場合、コラムの意味を表示する見出しの行が表示されます。

```
~]$ chronyc sources
 210 Number of sources = 3
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
=======
#* GPS0                          0   4   377    11   -479ns[ -621ns] +/-
134ns
```

```
^? a.b.c                         2   6   377    23   -923us[ -924us] +/-
43ms
^+ d.e.f                         1   6   377    21  -2629us[-2619us] +/-
86ms
```

各コラムの表示内容は、以下のとおりです。

M

ソースのモードを示します。**^** はサーバーを、**=** はピアを、**#** はローカル接続している参照クロックを意味します。

S

このコラムは、ソースの状態を示します。「*」 は、**chronyd** が現在同期しているソースを示します。「+」 は、選択されたソースと結合される受け入れ可能なソースを示します。「-」 は、受け入れ可能なソースで結合アルゴリズムに除外されたものを示します。「?」 は、接続が切断されたソース、もしくはパケットがすべてのテストをパスしないソースを示します。「x」 は、**chronyd** が falseticker と考える (つまり、その時間が他の大半のソースと一致しない) クロックを示します。「~」 は、時間の変動性が大きすぎるように見えるソースを示します。「?」 条件は、少なくとも 3 つのサンプルが収集されるまで開始時にも表示されます。

Name/IP address

ソースの名前または **IP** アドレス、もしくは参照クロックの参照 ID を表示します。

Stratum

最も間近に受信したサンプルでレポートされている、ソースの stratum を表示します。Stratum 1 は、ローカルで参照クロックに接続されているコンピューターを示します。Stratum 1 コンピューターに同期しているコンピューターは、stratum 2 に存在することになります。Stratum 2 コンピューターに同期しているコンピューターは stratum 3 に存在することになり、以後も同様に続きます。

Poll

ソースがポーリングされるレートで、間隔のベース-2 対数を秒数で示します。つまり、値が 6 の場合、64 秒ごとに測定が行われます。**chronyd** は、支配的な条件に対応して自動的にポーリングレートを変化させます。

Reach

ソースの到達可能性のレジスタで、八進法で表示されます。レジスタは 8 ビットで、ソースからパケットを受信するたび、もしくはミスするたびに更新されます。値が 377 の場合、最近の 8 回の通信から全体用の有効な返信が受信されたことを意味します。

LastRx

このコラムは、ソースから最後のサンプルが受信されたのがいつだったかを表示します。通常は、秒数で表示されます。**m**、**h**、**d**、および **y** の各文字は、それぞれ分、時間、日数、年を意味します。値が 10 年の場合、このソースからまだサンプルを受信していないことを示します。

Last sample

このコラムは、ローカルクロックと最後に測定されたソースのオフセットを表示します。角括弧内の数字は、実際に測定されたオフセットを表示します。これには **ns** (ナノ秒)、**us** (マイクロ秒)、**ms** (ミリ秒)、または **s** (秒) の各接尾辞が付く場合があります。角括弧の左側は元の測定を示し、slew がそれ以降にローカルクロックに適用可能になるように調整されています。**+/-** に続く数字は、測定におけるエラーのマージンを示します。オフセットの値がプラスの場合、ローカルクロックがソースよりも進んでいることを意味します。

13.3.6.3. chrony ソースの統計情報の確認

sourcestats コマンドを使うと、**chronyd** が現在調査している各ソースの誤差のレートとオフセットの予測プロセスについての情報を表示できます。オプション引数 **-v** を使うと、詳細情報になります。この場合、コラムの意味を表示する見出しの行が表示されます。

```
~]$ chronyc sourcestats


210 Number of sources = 1
Name/IP Address            NP  NR  Span  Frequency  Freq Skew  Offset
Std Dev
==============================================================================
=======
abc.def.ghi
```

各コラムの表示内容は、以下のとおりです。

Name/IP address

NTP サーバー (もしくはピア) の名前または**IP** アドレス、もしくは行の残りの部分が関連する参照クロックの参照 ID です。

NP

現在サーバーで保持されているサンプルポイントの数です。誤差レートと現在のオフセットは、これらのポイントを使って線形回帰を実行することで予測されます。

NR

最新の回帰を追跡している同一サインを持つ剰余の実行数です。この数字がサンプル数に対して少なくなりすぎる場合は、直線がデータに適合しなくなったことを意味します。実行数が少なすぎる場合、**chronyd** は古いサンプルを破棄し、実行数が受け入れ可能なものになるまで回帰を再実行します。

Span

一番古いサンプルと最新のサンプルの間隔です。単位が表示されない場合は、秒数を表しています。この例の間隔は 46 分です。

Frequency

これは予測されるサーバーの剰余周波数で、ppm (100 万分の 1) で表されます。このケースでは、コンピューターのクロックはサーバーよりも 10^9 分の 1 遅く実行していると判断されています。

Freq Skew

Freq の予測されるエラー範囲です (ppm (100 万分の 1) で表されます) 。

Offset

ソースの予測されるオフセットです。

Std Dev

サンプルの予測される標準偏差です。

13.3.7. システムクロックを手動で調整する

クロックの回転による進行中の調整を迂回して、システムクロックを直ちに更新するには、**root** で以下のコマンドを発行します。

```
~]# chronyc
      chrony> password commandkey-password
      200 OK
      chrony> makestep
      200 OK
```

ここでの *commandkey-password* は、キーファイルに保存されているコマンドキーもしくはパスワードになります。

rtcfile 指示文を使用している場合は、リアルタイムクロックを手動で調整しないでください。ランダムな調整を行うと、リアルタイムクロックがずれるレートを測定する **chrony** のニーズが妨害されます。

chronyc を使ってローカルの**chronyd** を設定する場合は、**-a** を使うと **authhash** と **password** のコマンドが自動的に実行されます。つまり、上記のインタラクティブのセッションが以下のもので置き換えられます。

```
chronyc -a makestep
```

13.4. 異なる環境での chrony の設定

13.4.1. 常時接続でないシステムにおける chrony の設定

ここでの例は、ダイヤル式のオンデマンド接続を使用するシステムを想定しています。断続的な接続を行うモバイルや仮想デバイスでは、通常設定で十分機能します。まず、**/etc/chrony.conf** のデフォルト設定が以下と同様になっているかを確認します。

```
driftfile /var/lib/chrony/drift
commandkey 1
keyfile /etc/chrony.keys
```

コマンドキー ID はインストール時に生成され、**/etc/chrony.keys** キーファイルの **commandkey** 値に対応します。

1. **root** としてエディターを使用し、以下のように 4 つの**NTP** サーバーのアドレスを追加します。

   ```
   server 0.pool.ntp.org offline
   server 1.pool.ntp.org offline
   server 2.pool.ntp.org offline
   server 3.pool.ntp.org offline
   ```

   offline オプションは、システムが接続をアクティブ化しないようにします。**chrony** デーモンは **chronyc** を待機して、システムがネットワークもしくはインターネットに接続したことを知らせます。

13.4.2. 孤立したネットワークでのシステムにおける chrony の設定

インターネットに決して接続しないネットワークでは、マスターのタイムサーバーに 1 台のコンピューターが選択されます。他のコンピューター群は、マスターの直接のクライアントか、クライアントのクライアントになります。マスター上では、システムクロックの平均誤差レートを使って誤差ファイルを手動で設

定する必要があります。マスターを再起動すると、マスターは周りのシステムから時間を取得し、平均を使ってシステムクロックを設定します。その後は、誤差ファイルに基づいて調整の適用を再開します。誤差ファイルは、**settime** コマンドの使用時に自動的に更新されます。

マスターに選ばれたシステム上でテキストエディターを使って **root** で **/etc/chrony.conf** を以下のように編集します。

```
driftfile /var/lib/chrony/drift
commandkey 1
keyfile /etc/chrony.keys
initstepslew 10 client1 client3 client6
local stratum 8
manual
allow 192.0.2.0
```

ここでの **192.0.2.0** は、クライアントの接続が許可されるネットワークもしくはサブネットアドレスになります。

マスターの直接のクライアントに選ばれたシステム上でテキストエディターを使って **root** で **/etc/chrony.conf** を以下のように編集します。

```
server master
driftfile /var/lib/chrony/drift
logdir /var/log/chrony
log measurements statistics tracking
keyfile /etc/chrony.keys
commandkey 24
local stratum 10
initstepslew 20 master
allow 192.0.2.123
```

ここでの **192.0.2.123** はマスターのアドレスで、**master** はマスターのホスト名になります。この設定になっているクライアントは、システムが再起動するとマスターと再同期を行います。

マスターの直接のクライアントにならないクライアントシステム上では、**local** および **allow** 指示文が省略される以外は、**/etc/chrony.conf** ファイルは同じものになります。

13.5. chronyc の使用

13.5.1. chronyc を使った chronyd の制御

インタラクティブモードで **chronyc** コマンドラインユーティリティーを使って chronyd の設定を変更するには、**root** で以下のコマンドを入力します。

```
~]# chronyc
```

制限のあるコマンドを使用するには、**chronyc** は **root** で実行する必要があります。

以下のように、**chronyc** コマンドプロンプトが表示されます。

```
chronyc>
```

コマンドすべてを一覧表示するには、**help** と入力します。

以下のようにコマンドと併せて呼び出すことで、インタラクティブモード以外でもユーティリティーを開始することができます。

```
~]# chronyc command
```

13.5.2. リモート管理での chronyc の使用

chrony を **chronyd** のリモートインスタンスに接続するように設定するには、**root** で以下の形式でコマンドを発行します。

```
~]# chronyc -h hostname
```

ここでの *hostname* は、**chronyd** を実行してホストからのリモート管理を可能にするシステムの**ホスト名** になります。デフォルトでは、ローカルホスト上のデーモンに接続します。

chrony をデフォルト以外のポート上にある**chronyd** のリモートインスタンスに接続するよう設定するには、**root** で以下の形式でコマンドを発行します。

```
~]# chronyc -h hostname -p port
```

ここでの *port* は、接続先の **chronyd** のインスタンスが制御および監視に使用するポートです。

chrony コマンドプロンプトで発行されるコマンドには、永続性がないことに注意してください。永続性があるのは、設定ファイル内のコマンドのみです。

MD5 と異なるハッシュをキーが使用している場合、システム管理者はリモートシステムから、**authhash** コマンドの後に、**chronyc** コマンドプロンプトで**password** コマンドを使用した後、コマンドを発行できます。

```
chronyc> password secretpasswordwithnospaces
200 OK
```

リモートシステムのコマンドキーに関連するパスワードもしくはハッシュは、**SSH** で入手するのが最善の方法になります。つまり、リモートマシンに **SSH** 接続を確立させ、セッション中は **/etc/chrony.conf** からのコマンドキーの ID と **/etc/chrony.keys** のコマンドキーを安全に記憶もしくは保存してください。

13.6. その他のリソース

以下の情報資料は、**chrony** に関するその他のリソースを提供します。

13.6.1. インストールされているドキュメント

- **chrony(1)** man ページ — **chrony** デーモンとコマンドラインインターフェイスツールを概説しています。
- **chronyc(1)** man ページ — コマンドおよびコマンドオプションを含む**chronyc** コマンドラインインターフェイスツールを説明しています。
- **chronyd(1)** man ページ — コマンドおよびコマンドオプションを含む**chronyd** デーモンを説明しています。
- **chrony.conf(5)** man ページ — **chrony** 設定ファイルを説明しています。

- **/usr/share/doc/chrony*/chrony.txt** — **chrony** スイートのユーザーガイドです。

13.6.2. 役に立つ Web ページ

http://chrony.tuxfamily.org/manual.html

chrony のオンラインユーザーガイドです。

第14章 ntpd を使用した NTP 設定

14.1. NTP の概要

Network Time Protocol (NTP) は正確な日時情報を広く行き渡らせ、ネットワークもしくはインターネットで共通の参照先に同期しているネットワーク化されたコンピューターシステム上のタイムクロックを維持します。世界中の多くの標準機関には原子時計があり、これは参照先として利用可能になっている場合があります。GPS を構成する衛星には 2 つ以上の原子時計が含まれており、時間信号を非常に正確なものにしています。この信号は、軍事的な理由で意図的に弱められる場合があります。理想的な状況では、各サイトに独自の参照時計があるサーバーがあり、これがサイト全体のタイムサーバーとして機能します。低周波の無線伝送や GPS 経由で日時を取得するデバイスは多く存在します。ただし多くの場合、インターネットに接続され各地に分散する公開されたアクセス可能なタイムサーバーを使うことができます。これらの **NTP** サーバーは、「*協定世界時*」 (UTC) を提供します。これらのタイムサーバーに関する情報は、『www.pool.ntp.org』 で確認できます。

IT では、多くの理由で正確な時間の維持が重要です。たとえばネットワーキングでは、正確なタイムスタンプがパケットとログで必要になります。たとえばネットワーキングでは、パケットとログで正確なタイムスタンプが必要になります。ログはサービスとセキュリティーの問題を調査するために使われるので、異なるシステム上で記録されるタイムスタンプが有用になるには、同期されたクロックで記録される必要があります。システムおよびネットワークがますます高速化するにつれ、これに対応してクロックの正確性と精度の必要性も高まっています。国によっては、正確な同期クロックを保持することが法律で定められているところもあります。詳細情報は、『www.ntp.org』 を参照してください。Linux システムでは、**NTP** がユーザースペースで実行しているデーモンにより実装されます。 Red Hat Enterprise Linux 7 のデフォルトの **NTP** ユーザースペースデーモンは、**chronyd** で、**ntpd** を使用する場合は、これを無効にする必要があります。**chrony** についての情報は、13章*chrony スイートを使用した NTP 設定*を参照してください。

ユーザースペースのデーモンは、カーネルで実行しているソフトウェアクロックであるシステムクロックを更新します。Linux はシステムクロックにソフトウェアクロックを使用し、これは 「*リアルタイムクロック*」 (RTC) と呼ばれる通常の組み込みハードウェアクロックよりも精度が高いものです。ハードウェアクロックについての情報は、**rtc(4)** および **hwclock(8)** の man ページを参照してください。システムクロックは、さまざまなクロックソースを使用することで時間を保ちます。通常は、*タイムスタンプカウンター* (TSC) が使われます。TSC は、それが最後にリセットされてからのサイクル数をカウントする CPU レジスタです。これは非常に高速で精度が高く、割り込みがありません。システムクロックはシステム起動時に RTC から日時を読み取ります。RTC が維持している時間は実際の時間と比べて、温度変化によりひと月で最大 5 分間の誤差を生じます。このため、システムクロックは外部の時間参照と常に同期する必要があります。**ntpd** がシステムクロックを同期している時には、カーネルは自動的に RTC を 11 分ごとに更新します。

14.2. NTP Strata (階層)

NTP サーバーは、時間信号のソースとなる原子時計からの同期距離によって分類されます。サーバーは、上は 1 から下は 15 までの stratum (階層) に分類されていると考えられます。このため、特定の層に言及する際は、stratum という言葉が使用されます。原子時計はソースであることから Stratum 0 と呼ばれます。ただし、インターネットに送信される Stratum 0 パケットはなく、すべての stratum 0 原子時計は stratum 1 と呼ばれるサーバーに接続されています。これらのサーバーは、Stratum 1 とマークされているパケットを送信します。stratum **n** のマークがついているパケットで同期されるサーバーは、次の下の stratum に所属し、パケットを stratum **n+1** とマークします。同一 stratum のサーバーは相互にパケットを交換できますが、これらが同期する最善の参照先となっている stratum の一階層下の stratum に所属するように指定されます。Stratum 16 という名称は、サーバーが現在は信頼できるタイムソースと同期していないことを意味します。

デフォルトでは、**NTP** クライアントはこれよりも下位の stratum にあるシステムのサーバーとして機能することに注意してください。

以下は **NTP** Strata のまとめになります。

Stratum 0:

原子時計と無線および GPS による信号送信

- GPS (全地球測位システム)
- 携帯電話システム
- 低周波無線送信 WWVB (米国コロラド州)、JJY-40 および JJY-60 (日本)、DCF77 (ドイツ)、および MSF (英国)

これらの信号は専用デバイスで受信可能で、通常は RS-232 で組織全体またはサイト全体のタイムサーバーとして使用されるシステムに接続されます。

Stratum 1:

電波時計、GPS 時計、または原子時計に接続しているコンピューター

Stratum 2:

stratum 1 から読み取り、下位の strata に提供

Stratum 3:

stratum 2 から読み取り、下位の strata に提供

Stratum *n+1*:

stratum *n* から読み取り、下位の strata に提供

Stratum 15:

stratum 14 から読み取ります。これが最下位の stratum です。

このプロセスが有効な最下位の Stratum 15 まで続きます。Stratum 16 というラベルは、非同期状態を意味します。

14.3. NTP の概要

Red Hat Enterprise Linux で使用する **NTP** のバージョンは、『RFC 1305 Network Time Protocol (Version 3) Specification, Implementation and Analysis』および『RFC 5905 Network Time Protocol Version 4: Protocol and Algorithms Specification』で説明されています。

NTP 実装すると、1 秒以下の正確性が達成できます。インターネット上では、数十ミリ秒の正確性は普通のことです。ローカルエリアネットワーク (LAN) 上では、1 ミリ秒の正確性は理想的な条件下では可能です。時計の誤差が計算され、修正されるようになっているためです。これは、以前の簡単な時間プロトコルシステムではなされていませんでした。64 ビットのタイムスタンプを使用することで、233 ピコ秒の精度が提供されます。ここではタイムスタンプの最初の 32 ビットが秒に使われ、次の 32 ビットが 1 秒以下に使われます。

NTP が示すのは、1900 年 1 月 1 日の GMT 午前 0 時 00 分からの積算秒数です。秒数のカウントには 32 ビットが使われているので、時間は 2036 年に「ロールオーバー」してしまいます。しかし、**NTP** はタイムスタンプ間の差異で機能するため、タイムプロトコルの他の実装ほどの問題はもたらされません。誤差が 68 年以内のハードウェアクロックが起動時に利用可能であれば、**NTP** は正確に現在の日時を解釈します。**NTP4** 仕様は、「Era Number」および「Era Offset」を提供します。これらは、68 年を超える長さの時間を処理する際に、ソフトウェアをより堅牢にするために使用できます。この問題を Unix の 2038 年問題と混同しないように注意してください。

NTP プロトコルは、正確性を高めるために追加情報を提供します。4 つのタイムスタンプを使うことで、往

復時間とサーバー応答時間の計算が可能になります。**NTP** クライアントとしての役割でシステムが参照時間サーバーと同期するために、「送信元タイムスタンプ」 の付いたパケットが送信されます。パケットが届くと、タイムサーバーが 「受信先タイムスタンプ」 を追加します。日時情報の要求を処理した後、パケットの返信前に 「転送先タイムスタンプ」 が追加されます。返信パケットが**NTP** クライアントに届くと、「受信先タイムスタンプ」 が生成されます。クライアントはこれで往復時間が計算でき、処理時間を差し引くことで実際の移動時間が導きだされます。送信時間と受信時間が同じだと仮定すると、**NTP** データを受信する際の 1 回の移動での遅延が計算されます。正式な **NTP** アルゴリズムは、ここで示されているものよりもはるかに複雑です。

時間情報を含むパケットは、受信後に直ちに処理されるのではなく、最初に検証され、その後にいくつかの他の時間サンプルと一緒に処理されて、時間を予想します。そしてシステムクロックと比較して、タイムオフセットを判断します。つまり、システムクロックの時間と **ntpd** が判断した時間の差異という事です。システムクロックは、使用中のカウンターの周波数を変更することで、最大でも 1 秒あたり 0.5 ミリ秒という非常にゆっくりしたペースで調整されてこのオフセットを減らします。この方法でクロックを 1 秒調整するには、少なくとも 2000 秒かかります。このゆっくりした変更はスルーイング (slewing) と呼ばれ、後ろ向きに実行することはできません。クロックのタイムオフセットが 128 ミリ秒 (デフォルト設定) を超える場合は、**ntpd** はクロックを進めるもしくは遅らす「ステップ」 が可能です。システム起動時のタイムオフセットが 1000 秒を超える場合は、ユーザーもしくはインストールスクリプトが手動で調整する必要があります。*2章日付と時刻の設定* を参照してください。**-g** オプションを **ntpd** コマンドで (デフォルトで使用) 使用すると、システム起動時のオフセットは修正されますが、通常の操作中に修正されるオフセットは最大 1000 秒までです。

時間を遅らせると、失敗したりエラーになるソフトウェアもいくつかあります。時間変更のステップに敏感なシステムでは、**-x** オプション (**-g** オプションとは無関係) を使うことでしきい値を 128 ミリ秒から 600 秒に変更できます。ただし、**-x** オプションを使ってステップの制限を 128 ミリ秒から 600 秒に増やすと、クロック制御に異なる方法が使用されるので、マイナス面もあります。カーネルのクロック規範が無効になり、クロックの正確性にマイナスの影響が出る可能性があります。**-x** オプションは、**/etc/sysconfig/ntpd** 設定ファイルに追加することができます。

14.4. 誤差ファイルの概要

誤差ファイルは、通常の周波数で稼働しているシステムクロックと、UTC と同期し続けるために必要な周波数との間の周波数オフセットを保存するために使われます。誤差ファイルに値がある場合は、システム起動時に読み取られ、クロックソースの修正に使われます。誤差ファイルを使用すると、安定的かつ正確な時間の達成に必要な時間が短縮されます。この値は、1 時間ごとに **ntpd** が計算して、誤差ファイルはそのたび置換されます。誤差ファイルは更新されるのではなく置換されるので、**ntpd** が書き込みパーミッションのあるディレクトリーに格納される必要があります。

14.5. UTC、タイムゾーン、および DST

NTP は完全に UTC (協定世界時) を使用しているため、タイムゾーンと夏時間 (DST) はシステムがローカルで適用します。**/etc/localtime** ファイルは、**/usr/share/zoneinfo** のコピーであるか、またはこのファイルへのシンボリックリンクです。RTC は、**/etc/adjtime** の 3 行目に指定してあるとおり、ローカルタイムまたは UTC になります。これは、LOCAL または UTC のいずれかで、RTC クロックの設定方法を示します。この設定は、**日付と時刻** のグラフィカル設定ツール内の **システムクロックで UTC を使用** のチェックボックスを使うと簡単に変更できます。このツールの使用方法については、*2章日付と時刻の設定* を参照してください。夏時間を変更する際には、各種の問題を避けるために RTC を UTC で実行することが推奨されます。

ntpd の詳細な操作方法は、**ntpd(8)** の man ページで説明されています。リソースセクションでは、役に立つ情報ソースを紹介しています。「その他のリソース」 を参照してください。

14.6. NTP の認証オプション

NTPv4 は Autokey Security Architecture のサポートを追加しました。これは、シンメトリックキーの暗号化のサポートを維持する一方で、公開の非対称暗号法に基づいています。Autokey Security Architecture については、『RFC 5906 Network Time Protocol Version 4: Autokey Specification』で説明されています。**ntp_auth(5)** の man ページでは、認証オプションと **ntpd** 用のコマンドが説明されています。

ネットワーク上の攻撃者は、不正確な時間情報のある **NTP** パケットを送信することで、サービス妨害を試みることがあります。**NTP** サーバーのパブリックプールを使用しているシステムでは、**/etc/ntp.conf** のパブリック **NTP** 一覧内に 4 つ以上の **NTP** サーバーを記載することでこのリスクが緩和されます。1 つのタイムソースのみが危険にさらされるか、なりすましを受けた場合、**ntpd** はそのソースを無視します。リスク評価を実行し、不正確な時間がアプリケーションおよび組織に及ぼす影響を検討してください。内部のタイムソースがある場合は、**NTP** パケットが配布されるネットワークを保護する手段を検討してください。リスク評価を実行して、リスクが受け入れられるものでアプリケーションへの影響が最小限であると判断した場合は、認証を使わない選択が可能になります。

ブロードキャストとマルチキャストの各モードでは、デフォルトで認証が必要になります。ネットワークが信頼できると判断した場合は、**ntp.conf** ファイル内の **disable auth** 指示文を使って認証を無効にできます。別の方法では、SHA1 または MD5 シンメトリックキーを使って認証を設定するか、Autokey スキームを使用して公開 (非対称) キー暗号法で認証を設定する必要があります。非対称暗号法の Autokey スキームは、**ntp_auth(8)** man ページで、キーの生成については **ntp-keygen(8)** で説明されています。シンメトリックキー暗号法の実装方法については、「鍵を使った対称認証の設定」の **key** オプションを参照してください。

14.7. 仮想マシン上での時間管理

仮想マシンは実際のハードウェアクロックにアクセスできず、仮想クロックの安定性はホストシステムの作業量に依存することから、十分な安定性がありません。このため、使用する仮想化アプリケーションが準仮想化クロックを提供するべきです。**KVM** のある Red Hat Enterprise Linux では、デフォルトのクロックソースは **kvm-clock** になります。『Virtualization Host Configuration and Guest Installation Guide』の『KVM guest timing management』の章を参照してください。

14.8. うるう秒の概要

グリニッジ標準時 (GMT) は太陽日の測定から導きだされており、これは地球の自転に依存しています。原子時計が最初に作成された際に、より正確な時間の定義が可能になりました。1958 年に国際原子時 (TAI) がより正確で安定的な原子時計を基に導入されました。さらに正確な天文時である世界時 1 (UT1) も導入され、GMT に代わるものとなりました。実際、原子時計は地球の自転よりもはるかに安定しているので、この 2 つの時間の差異が広がり始めました。これが理由で、現実的な方法として UTC が導入されました。これは UT1 の 1 秒以内に維持されますが、わずかな調整を何度も行うことを避けるため、*うるう秒*の概念を導入し、管理可能な方法でこの差異を調整することにしました。UT1 と UTC の差異は、0.5 秒以上になるまで監視されます。1 秒進めるまたは遅らす調整が必要とみなされた場合にのみ、これが実行されます。地球の自転速度は不規則なため、長期の将来にわたって調整の必要性を予測することはできません。調整をいつ行うかについては、『国際地球回転・基準系事業 (IERS)』が判断します。しかし、**NTP** は保留中のうるう秒についての情報を転送し、これらを自動的に適用するので、この発表が重要となるのは、Stratum 1 サーバーの管理者のみになります。

14.9. ntpd 設定ファイルの概要

ntpd デーモンは、システム起動時またはサービスの再起動時に設定ファイルを読み取ります。このファイルのデフォルトの位置は **/etc/ntp.conf** で、以下のコマンド入力すると見ることができます。

```
~]$ less /etc/ntp.conf
```

設定コマンドについては、本章の後半 「NTP の設定」 で簡単に説明されており、詳しくは**ntp.conf(5)** man ページで説明されています。

ここでは、以下でデフォルトの設定ファイルを簡単に説明します。

driftfile エントリー

誤差ファイルへのパスが指定され、Red Hat Enterprise Linux でのデフォルトエントリーは以下のようになります。

```
driftfile /var/lib/ntp/drift
```

これを変更する場合は、ディレクトリーが **ntpd** で書き込み可能となっていることを確認してください。ファイルには、システムもしくはサービス起動時に毎回システムクロックの周波数を調整する値が含まれます。詳細情報は、Understanding the Drift File を参照してください。

アクセス制御エントリー

以下の行は、デフォルトのアクセス制御を設定します。

```
restrict default nomodify notrap nopeer noquery
```

nomodify オプションは、設定の変更を防ぎます。**notrap** オプションは、**ntpdc** 制御メッセージプロトコルトラップを防ぎます。**nopeer** オプションは、ピア関連付けの形成を防ぎます。**noquery** オプションは、**ntpq** および **ntpdc** クエリーへの回答を妨げますが、タイムクエリーは妨げません。**ntpq** および **ntpdc** クエリーは増幅攻撃に使用できるので (詳細は『CVE-2013-5211』 を参照)、アクセスを公開しているシステムでは**restrict default** コマンドから **noquery** オプションを削除しないでください。

127.0.0.0/8 範囲内のアドレスは、種々のプロセスやアプリケーションが必要とすることがあります。上記の "restrict default" 行は明示的に許可されていないすべてのものへのアクセスを妨害するので、**IPv4** および **IPv6** の標準ループバックアドレスへのアクセスは以下の行で許可されます。

```
# the administrative functions.
restrict 127.0.0.1
restrict ::1
```

別のアプリケーションで特に必要とされる場合は、アドレスはすぐ下に追加できます。

ローカルネットワーク上のホストは、上記の "restrict default" 行のために許可されません。これを変更して、たとえば **192.0.2.0/24** ネットワークからのホストが時間および統計情報のみをクエリーできるようにするには、以下の形式の行が必要になります。

```
restrict 192.0.2.0 mask 255.255.255.0 nomodify notrap nopeer
```

特定のホスト、たとえば **192.0.2.250/24** からの無制限のアクセスを許可するには、以下の形式の行が必要になります。

```
restrict 192.0.2.250
```

指定がない場合は、**255.255.255.255** のマスクが適用されます。

制限コマンドは、**ntp_acc(5)** man ページで説明されています。

公開サーバーエントリー

デフォルトでは、以下の 4 つの公開サーバーエントリーが **ntp.conf** ファイルに格納されています。

```
server 0.rhel.pool.ntp.org iburst
server 1.rhel.pool.ntp.org iburst
server 2.rhel.pool.ntp.org iburst
server 3.rhel.pool.ntp.org iburst
```

ブロードキャストマルチキャストサーバーエントリー

デフォルトでは、**ntp.conf** ファイルにはコメントアウトされた例がいくつか含まれています。特定のコマンドについての説明は、「NTP の設定」を参照してください。必要な場合は、コマンドを用例のすぐ下に追加します。

注記

DHCP クライアントプログラムである**dhclient** は、**DHCP** サーバーから **NTP** サーバーのリストを受信すると、これに **ntp.conf** を追加してサービスを再起動します。この機能を無効にするには、**PEERNTP=no** を **/etc/sysconfig/network** に追加します。

14.10. ntpd Sysconfig ファイルの概要

このファイルは、サービス起動時に **ntpd** init スクリプトが読み取ります。デフォルトのコンテンツは以下の通りです。

```
# Command line options for ntpd
OPTIONS="-g"
```

-g オプションを使うと、**ntpd** がオフセットの制限である 1000 秒を無視して、1000 秒を超える場合でも時間の同期を試みます。ただし、これはシステム起動時のみです。このオプションを使用しないと、タイムオフセットが 1000 秒を超える場合、**ntpd** は終了します。また、**-g** オプションを使用した場合でもサービスが再起動してオフセットが 1000 秒を超える場合は、システム起動後に終了します。

14.11. chrony の無効化

ntpd を使用するには、デフォルトのユーザースペースデーモンである**chronyd** を停止して、無効にする必要があります。これには、**root** で以下のコマンドを発行します。

```
~]# systemctl stop chronyd
```

システム起動時に自動的に起動しないようにするには、**root** で以下のコマンドを発行します。

```
~]# systemctl disable chronyd
```

chronyd のステータスを確認するには、以下のコマンドを発行します。

```
~]$ systemctl status chronyd
```

14.12. NTP デーモンのインストールを確認する

ntpd がインストールされていることを確認するには、**root** で以下のコマンドを発行します。

```
~]# yum install ntp
```

NTP デーモンまたはサービス **ntpd** で実装されます。これは、*ntp* パッケージに含まれています。

14.13. NTP デーモン (ntpd) のインストール

ntpd をインストールするには、**root** で以下のコマンドを発行します。

```
~]# yum install ntp
```

システム起動時に **ntpd** を有効にするには、**root** で以下のコマンドを発行します。

```
~]# systemctl enable ntpd
```

14.14. NTP ステータスの確認

ntpd が実行中で、システム起動時に実行される設定になっていることを確認するには、以下のコマンドを発行します。

```
~]$ systemctl status ntpd
```

ntpd から簡単なステータスレポートを取得するには、以下のコマンドを発行します。

```
~]$ ntpstat
unsynchronised
  time server re-starting
   polling server every 64 s
```

```
~]$ ntpstat
synchronised to NTP server (10.5.26.10) at stratum 2
   time correct to within 52 ms
   polling server every 1024 s
```

14.15. 着信 NTP パケットを許可するファイアウォールの設定

NTP トラフィックはポート **123** 上の **UDP** パケットで構成されており、**NTP** が機能するにはネットワークおよびホストベースのファイアウォール通過が許可されている必要があります。

グラフィカルの **Firewall Configuration** ツールを使って、ファイアウォールの設定でクライアントが着信 **NTP** トラフィックを許可しているかどうかを確認します。

グラフィカルの **firewall-config** ツールを起動するには、**Super** キーを押してアクティビティーを開き、**firewall** と入力してから **Enter** を押します。**firewall-config** ツールが表示され、パスワード入力が求められます。

コマンドラインを使ってグラフィカルなファイアウォール設定ツールを起動するには、**root** で以下のコマンドを入力します。

```
~]# firewall-config
```

ファイアウォールの設定 ウィンドウが開きます。このコマンドは一般ユーザーとしても実行できますが、その場合は時折 **root** パスワードの入力を求められることに注意してください。

ウィンドウ左下の 「接続しました」 の表示を確認してください。これは、**firewall-config** ツールがユーザースペースデーモン **firewalld** に接続されていることを示します。

14.15.1. ファイアウォールの設定変更

現行のファイアウォール設定を直ちに変更するには、現在の表示が **実行時** に設定されていることを確認してください。次回のシステム起動時またはファイアウォールの再読み込み時に設定が適用されるようにするには、ドロップダウンリストから **永続** を選択します。

注記

実行時 モードでファイアウォール設定を変更する際には、サービスに関連するチェックボックスにチェックを入れたり消したりすると、その選択が直ちに反映されます。他のユーザーが使用している可能性のあるシステムでこの作業を行う場合は、この点を考慮してください。

永続 モードでファイアウォール設定を変更すると、ファイアウォールを再読み込みした場合かシステムの再起動時にのみ変更が反映されます。**ファイル** メニューの下にある再読み込みアイコンを使用するか、**オプション** メニューをクリックして **Firewalld の再読み込み** を選択します。

14.15.2. NTP パケット用にファイアウォールでポートを開く

特定のポートへのトラフィックがファイアウォールを通過できるようにするには、**firewall-config** ツールを起動して、設定を変更するネットワークゾーンを選択します。**ポート** タブを選んで **追加** ボタンをクリックします。**ポートとプロトコル** ウィンドウが開きます。

ポート番号 **123** を入力し、ドロップダウンリストから **udp** を選択します。

14.16. ntpdate サーバーの設定

ntpdate サービスの目的は、システム起動時にクロックを設定することです。このサービスはこれまで、**ntpdate** が正確な時間を確保して、クロックでジャンプが生じないようにしてからサービスが開始するために使われていました。**ntpdate** および step-tickers リストの使用は非推奨とみなされているため、Red Hat Enterprise Linux 7 では **-g** オプションを付けた **ntpd** コマンドをデフォルトで使用し、**ntpdate** は使用しません。

Red Hat Enterprise Linux 7 の **ntpdate** サービスが役に立つのは、**ntpd** を使わずに単独で使用する場合のみです。サービスを並行して起動する **systemd** では、**ntpdate** サービスが提供する **time-sync.target** への並び順依存関係を指定しない限り、**ntpdate** サービスを有効にしても正確な時間を確保した後に他のサービスが開始されるようにはなりません。あるサービスが正確な時間を伴って開始されるようにするには、そのサービスに **After=time-sync.target** を追加し、ターゲット (**ntpdate** または **sntp**) を提供するサービスの 1 つを有効にします。Red Hat Enterprise Linux 7 上のサービスには、デフォルトでこの依存関係が含まれているものもあります (たとえば、**dhcpd**、**dhcpd6**、および **crond**)。

ntpdate がシステム起動時に実行されるようになっていることを確認するには、以下のコマンドを発行します。

```
~]$ systemctl status ntpdate
```

システム起動時にサービスが実行するようにするには、**root** で以下のコマンドを発行します。

```
~]# systemctl enable ntpdate
```

Red Hat Enterprise Linux 7 では、デフォルトの **/etc/ntp/step-tickers** ファイルに **0.rhel.pool.ntp.org** が含まれています。追加の**ntpdate** サーバーを設定するには、テキストエディターを **root** で実行し、**/etc/ntp/step-tickers** を編集します。**ntpdate** は、システム起動時に日付情報を取得するためにこのファイルを 1 回使用するだけなので、記載されているサーバー数は重要ではありません。内部のタイムサーバーがある場合は、そのホスト名を 1 行目に使います。2 行目に追加のホストをバックアップとしておくのがよいでしょう。バックアップサーバーにどれを選ぶか、また 2 番目のホストを内部または外部とするかは、リスク評価によります。たとえば、1 番目のサーバーに影響する問題が 2 番目のサーバーにも影響する可能性はどの程度か。1 番目のサーバーにアクセスできなくなるネットワーク障害時に、より接続性が高いのは外部サーバーかそれとも内部サーバーか、といった点を考慮します。

14.17. NTP の設定

NTP サービスのデフォルト設定を変更するには、**root** でテキストエディターを使用して **/etc/ntp.conf** ファイルを編集します。このファイルは、**ntpd** と共にインストールされ、Red Hat プールからのタイムサーバーを使用するデフォルト設定になっています。**ntp.conf(5)** man ページでは、アクセスおよびレート制限コマンドを除く、設定ファイルで使用可能なコマンドオプションが説明されています。アクセスおよびレート制限コマンドは、**ntp_acc(5)** man ページで説明されています。

14.17.1. NTP サービスへのアクセス制御の設定

システム上で実行中の **NTP** サービスへのアクセスを制限または制御するには、**ntp.conf** ファイル内の **restrict** コマンドを利用します。コメントアウトされた例は以下のとおりです。

```
# Hosts on local network are less restricted.
#restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap
```

restrict コマンドは以下の形式になります。

```
restrict option
```

ここでの *option* は、以下のいずれか (1 つまたは複数) になります。

- **ignore** — **ntpq** および **ntpdc** クエリーを含むすべてのパケットが無視されます。
- **kod** — 「Kiss-o'-death」 パケットが送信され、不要なクエリーが少なくなります。
- **limited** — パケットがレート制限のデフォルト値もしくは**discard** コマンドで指定された値に違反する場合、タイムサーバー要求に応答しません。**ntpq** および **ntpdc** クエリーには影響ありません。**discard** コマンドおよびデフォルト値に関する詳細情報は、「NTP サービスへのレート制限アクセスの設定」 を参照してください。
- **lowpriotrap** — 一致するホストがトラップを低い優先度に設定します。
- **nomodify** — 設定に変更を加えられないようにします。
- **noquery** — **ntpq** および **ntpdc** クエリーに応答しないようにしますが、タイムクエリーは除外されます。

- **nopeer** — ピア関連付けの形成をできないようにします。
- **noserve** — **ntpq** および **ntpdc** クエリーを除くすべてのパケットを拒否します。
- **notrap** — **ntpdc** 制御メッセージプロトコルトラップをできないようにします。
- **notrust** — 暗号法で認証されないパケットを拒否します。
- **ntpport** — 発信元ポートが標準の **NTP UDP** ポート **123** の場合、一致アルゴリズムが制限のみを適用するように修正します。
- **version** — 現在の **NTP** バージョンに一致しないパケットを拒否します。

レート制限アクセスがクエリーに対してまったく応答しないように設定するには、各 **restrict** コマンドで **limited** オプションが必要になります。**ntpd** が **KoD** パケットで応答するには、**restrict** コマンドは **limited** および **kod** の両方のオプションが必要になります。

ntpq および **ntpdc** クエリーは増幅攻撃に使用可能 (詳細は『CVE-2013-5211』 を参照) なので、アクセスを公開しているシステムでは、**restrict default** コマンドから **noquery** オプションを削除しないでください。

14.17.2. NTP サービスへのレート制限アクセスの設定

システム上で実行中の **NTP** サービスへのレート制限アクセスを有効にするには、「NTP サービスへのアクセス制御の設定」 の説明にあるように、**restrict** コマンドに **limited** オプションを追加します。デフォルトの discard パラメーターを使用したくない場合は、以下の説明にあるように **discard** コマンドも使用できます。

discard コマンドは以下の形式になります。

```
discard [average value] [minimum value] [monitor value]
```

- **average** — 許可される最小限の平均パケット間隔を指定します。$\log_2$ 秒の引数を取ります。デフォルト値は 3 です (2^3 は 8 秒と同等です)。
- **minimum** — 許可される最小限のパケット間隔を指定し、$\log_2$ 秒の引数を取ります。デフォルト値は 1 です (2^1 は 2 秒と同等です)。
- **monitor** — 許可されるレート制限を超えた場合のパケットの discard の確率を指定します。デフォルト値は、3000 秒です。このオプションは、1 秒あたり 1000 以上のリクエストを受信するサーバーを意図しています。

discard コマンドの例を以下に示します。

```
discard average 4
```

```
discard average 4 minimum 2
```

14.17.3. ピアアドレスの追加

ピアのアドレス、つまり同一 stratum の **NTP** サービスを実行しているサーバーのアドレスを追加するには、**ntp.conf** ファイルの **peer** コマンドを利用します。

peer コマンドは以下の形式になります。

```
peer address
```

ここでの *address* は、**IP** ユニキャストアドレスもしくは**DNS** の解決可能な名前になります。アドレスは、同一 stratum のメンバーに既知のシステムのものである必要があります。ピアは、相互に異なる時間ソースを少なくとも 1 つは持つべきです。ピアは通常、同一管理制御下にあるシステム群です。

14.17.4. サーバーアドレスの追加

サーバーのアドレス、つまり一段階上の stratum の **NTP** サービスを実行しているサーバーのアドレスを追加するには、**ntp.conf** ファイルの **server** コマンドを利用します。

server コマンドは以下の形式になります。

```
server address
```

ここでの *address* は、**IP** ユニキャストアドレスもしくは**DNS** の解決可能な名前になります。パケットの送信元となるリモートリファレンスサーバーもしくはローカルリファレンスクロックのアドレスになります。

14.17.5. ブロードキャストまたはマルチキャストサーバーアドレスの追加

送信用のブロードキャストまたはマルチキャストアドレス、つまり **NTP** パケットをブロードキャストまたはマルチキャストする宛先のアドレスを追加 するには、**ntp.conf** ファイルの **broadcast** コマンドを利用します。

ブロードキャストおよびマルチキャストの両モードは、デフォルトで認証を必要とします。「NTP の認証オプション」 を参照してください。

broadcast コマンドは以下の形式になります。

```
broadcast address
```

ここでの *address* は、パケットの送信先となる **IP** ブロードキャストもしくはマルチキャストアドレスになります。

このコマンドは、システムが **NTP** ブロードキャストサーバーとして作動するように設定します。使用するアドレスは、ブロードキャストかマルチキャストアドレスである必要があります。ブロードキャストアドレスの場合は、**IPv4** アドレス **255.255.255.255** を意味します。デフォルトでは、ルーターはブロードキャストメッセージを渡しません。マルチキャストアドレスの場合は、**IPv4** Class D アドレスか **IPv6** アドレスになります。IANA は **IPv4** マルチキャストアドレス **224.0.1.1** と **IPv6** アドレス **FF05::101** (サイトローカル) を **NTP** に割り当てています。『RFC 2365 Administratively Scoped IP Multicast』 の説明にあるように、管理者が範囲指定した **IPv4** マルチキャストアドレスも使用可能です。

14.17.6. Manycast クライアントアドレスの追加

manycast クライアントアドレスを追加する、つまり **NTP** サーバーの発見に使用するマルチキャストアドレスを設定するには、**ntp.conf** ファイルの **manycastclient** コマンドを利用します。

manycastclient コマンドは以下の形式になります。

```
manycastclient address
```

ここでの *address* は **IP** マルチキャストアドレスで、ここからパケットが受信されます。クライアントはこのアドレスにリクエストを送信し、応答から最善のサーバーを選んで他を無視します。その後は、**NTP** 通信は発見された **NTP** サーバーが **ntp.conf** リストされているかのようにユニキャスト関連付けを使用します。

このコマンドは、システムが **NTP** クライアントのように動作するように設定します。システムは、同時にクライアントとサーバーの両方になることができます。

14.17.7. ブロードキャストクライアントアドレスの追加

ブロードキャストクライアントのアドレスを追加するには、つまり ブロードキャスト **NTP** パケット用にブロードキャストアドレスを監視するように設定するには、**ntp.conf** ファイルの **broadcastclient** コマンドを利用します。

broadcastclient コマンドは以下の形式になります。

```
broadcastclient
```

ブロードキャストメッセージを受信可能にします。デフォルトで認証を必要とします。「NTP の認証オプション」 を参照してください。

このコマンドは、システムが **NTP** クライアントのように動作するように設定します。システムは、同時にクライアントとサーバーの両方になることができます。

14.17.8. Manycast サーバーアドレスの追加

manycast サーバーのアドレスを追加する、つまり **NTP** パケットをマルチキャストすることでクライアントがサーバーを発見できるようにするアドレスを設定するには、**ntp.conf** ファイルの **manycastserver** コマンドを利用します。

manycastserver コマンドは以下の形式になります。

```
manycastserver address
```

マルチキャストメッセージの送信ができるようになります。ここでの *address* は、マルチキャスト送信先のアドレスになります。これは認証と合わせて使用して、サービス中断を防ぎます。

このコマンドは、システムが **NTP** サーバーのように動作するように設定します。システムは、同時にクライアントとサーバーの両方になることができます。

14.17.9. マルチキャストクライアントアドレスの追加

マルチキャストクライアントのアドレスを追加するには、つまりマルチキャスト **NTP** パケット用にマルチキャストアドレスを監視するように設定するには、**ntp.conf** ファイルの **multicastclient** コマンドを利用します。

multicastclient コマンドは以下の形式になります。

```
multicastclient address
```

マルチキャストメッセージの受信ができるようになります。ここでの *address* は、サブスクライブするアドレスになります。これは認証と合わせて使用して、サービス中断を防ぎます。

このコマンドは、システムが **NTP** クライアントのように動作するように設定します。システムは、同時にクライアントとサーバーの両方になることができます。

14.17.10. Burst オプションの設定

公開サーバーに対して **burst** オプションを使用することは、濫用とみなされます。公開**NTP** サーバーでは、このオプションを使用しないでください。このオプションは、組織内のアプリケーションにのみ、使用するようにしてください。

時間オフセットの統計情報の平均的な品質を向上させるには、サーバーコマンドの最後に以下のオプションを追加します。

```
burst
```

サーバーが反応している場合は、ポーリングの間隔ごとにシステムが通常の 1 パケットではなく、最大 8 パケットのバーストを送信します。**server** コマンドを使うと、時間オフセット計算の平均的な質が向上します。

14.17.11. iburst オプションの設定

初回同期にかかる時間を改善するには、サーバーコマンドの最後に以下のオプションを追加します。

```
iburst
```

ポーリングの間隔ごとに、1 パケットではなく、8 パケットのバーストが送信されます。サーバーが応答しない場合、パケットは 16 秒間隔で送信されます。サーバーが応答する場合は、パケットは 2 秒間隔で送信されます。**server** コマンドと使用すると、初回同期にかかる時間が短縮されます。これが設定ファイルてのデフォルトオプションになります。

14.17.12. 鍵を使った対称認証の設定

鍵を使った対称認証を設定するには、サーバーもしくはピアコマンドの最後に以下のオプションを追加します。

```
key number
```

ここでの *number* は、**1** から **65534** までの数字になります。このオプションを使うと、パケット内で*メッセージ認証コード* (MAC) が使えるようになります。このオプションは、**peer**、**server**、**broadcast**、および **manycastclient** の各コマンドで使用します。

このオプションは、**/etc/ntp.conf** 内で以下のように使用します。

```
server 192.168.1.1 key 10
broadcast 192.168.1.255 key 20
manycastclient 239.255.254.254 key 30
```

「NTP の認証オプション」 も参照してください。

14.17.13. ポーリング間隔の設定

デフォルトのポーリング間隔を変更するには、サーバーもしくはピアコマンドの最後に以下のオプションを追加します。

```
minpoll value and maxpoll value
```

デフォルトのポーリング間隔を変更するオプションです。ここでは、間隔の秒数は 2 の *value* の累乗を計算

して出されたものです。つまり、間隔は $\log_2$ 秒で表示されます。デフォルトの**minpoll** 値は 6 なので、2^6 は 64 秒となります。デフォルトの**maxpoll** 値は 10 なので、1024 秒になります。使用可能な値は、3 から 17 なので、8 秒から 36.4 時間までに相当します。これらのオプションは、**peer** または **server** で使用します。**maxpoll** を低く設定すると、クロックの精度が高まります。

14.17.14. サーバー優先順位の設定

特定のサーバーが他の同様の統計情報のサーバーよりも優先されるよう指定するには、サーバーまたはピアコマンドの最後に以下のオプションを追加します。

```
prefer
```

他の同様の統計情報のサーバーに優先して、このサーバーが同期に使用されます。このオプションは、**peer** または **server** コマンドで使用します。

14.17.15. NTP パケットの Time-to-Live (有効期限) の設定

デフォルトで使用される特定の *time-to-live* (TTL: 有効期限) の値を指定するには、サーバーまたはピアコマンドの最後に以下のオプションを追加します。

```
ttl value
```

ブロードキャストサーバーおよびマルチキャスト **NTP** サーバーが送信するパケットで使用される TTL の値を指定します。manycast クライアントが 「expanding ring search」 を使用するには、TTL の最大値を指定します。デフォルト値は **127** です。

14.17.16. 使用する NTP バージョンの設定

デフォルトで使用される特定バージョンの **NTP** を指定するには、サーバーまたはピアコマンドの最後に以下のオプションを追加します。

```
version value
```

作成された **NTP** パケット内の **NTP** セットのバージョンを指定します。値は **1** から **4** になります。デフォルト値は **4** です。

14.18. ハードウェアクロック更新の設定

ハードウェアクロック、またはリアルタイムクロック (RTC)、を更新するシステムクロックを設定するには、**ntpdate** を実行した後に以下の行を **/etc/sysconfig/ntpdate** に追加します。

```
SYNC_HWCLOCK=yes
```

ハードウェアクロックをシステムクロックから更新するには、**root** で以下のコマンドを実行します。

```
~]# hwclock --systohc
```

システムクロックが **ntpd** で同期されている際に、カーネルは自動的に RTC を 11 分ごとに更新します。

14.19. クロックソースの設定

システムで使用可能なクロックソースを一覧表示するには、以下のコマンドを発行します。

```
~]$ cd /sys/devices/system/clocksource/clocksource0/
clocksource0]$ cat available_clocksource
kvm-clock tsc hpet acpi_pm
clocksource0]$ cat current_clocksource
kvm-clock
```

上記の例では、カーネルは **kvm-clock** を使用しています。これは仮想マシンなので、起動時にこのクロックソースが選択されています。

デフォルトのクロックソースを上書きするには、**grub.conf** に以下のような行を追加します。

```
clocksource=tsc
```

利用可能なクロックソースは、アーキテクチャーに依存します。

14.20. その他のリソース

以下の情報ソースで **NTP** および **ntpd** に関する追加リソースが提供されています。

14.20.1. インストールされているドキュメント

- **ntpd(8)** man ページ — **ntpd** の詳細な説明があり、コマンドラインオプションも含まれています。
- **ntp.conf(5)** man ページ — サーバーおよびピアとの関連付けの設定方法に関する情報が含まれています。
- **ntpq(8)** man ページ — **NTP** サーバーの監視およびクエリー用の**NTP** クエリーユーティリティーが説明されています。
- **ntpdc(8)** man ページ — **ntpd** の状態をクエリー、変更するための**ntpd** ユーティリティーを説明しています。
- **ntp_auth(5)** man ページ — **ntpd** の認証オプション、コマンド、および鍵管理を説明しています。
- **ntp_keygen(8)** man ページ — **ntpd** の公開および秘密鍵生成を説明しています。
- **ntp_acc(5)** man ページ — **restrict** コマンドを使ったアクセス制御オプションを説明しています。
- **ntp_mon(5)** man ページ — 統計情報収集に関する監視オプションを説明しています。
- **ntp_clock(5)** man ページ — 基準クロックを設定するコマンドを説明しています。
- **ntp_misc(5)** man ページ — その他のオプションを説明しています。
- **ntp_decode(5)** man ページ — **ntpd** レポーティングおよび監視に使用されるステータス文字、イベントメッセージ、およびエラーコードを一覧表示します。
- **ntpstat(8)** man ページ — ローカルマシン上で実行している**NTP** デーモンの同期状態をレポートするユーティリティーを説明しています。
- **ntptime(8)** man ページ — カーネル時間変数を読み取り、設定するユーティリティーを説明しています。
- **tickadj(8)** man ページ — ティックの長さを読み取り、オプションで設定するユーティリティーを説明しています。

14.20.2. 役に立つ Web ページ

http://doc.ntp.org/

NTP 資料のアーカイブ

http://www.eecis.udel.edu/~mills/ntp.html

Network Time Synchronization Research Project

http://www.eecis.udel.edu/~mills/ntp/html/manyopt.html

NTPv4 での Automatic Server Discovery に関する情報

第15章 ptp4l を使用した PTP の設定

15.1. PTP の概要

Precision Time Protocol (PTP) は、ネットワーク内でのクロック同期に使用されるプロトコルです。ハードウェアサポートと合わせて使用すると **PTP** はマイクロ秒以下の正確性があり、これは**NTP** で得られる正確性よるもはるかに優れています。**PTP** サポートは、カーネルとユーザースペースに分けられます。Red Hat Enterprise Linux のカーネルには **PTP** クロックのサポートが含まれており、これはネットワークドライバーが提供しています。実際のプロトコル実装は **linuxptp** と呼ばれ、これは Linux の IEEE 標準 1588 に準拠した **PTPv2** 実装です。

linuxptp パッケージには、クロック同期用の **ptp4l** および **phc2sys** プログラムが含まれています。**ptp4l** プログラムは、**PTP** 境界クロックと通常のクロックを実装します。ハードウェアタイムスタンプでは **PTP** ハードウェアクロックのマスタークロックとの同期に使用され、ソフトウェアタイムスタンプではシステムクロックのマスタークロックとの同期に使用されます。**phc2sys** プログラムは、システムクロックを *network interface card* (NIC) 上の **PTP** ハードウェアクロックと同期するハードウェアタイムスタンプでのみ必要となります。

15.1.1. PTP を理解する

PTP で同期するクロックは、マスター/スレーブ階層で組織されています。スレーブはマスターと同期し、このマスターは別のマスターのスレーブとなっています。この階層は、*best master clock* (BMC) アルゴリズムで作成され、自動的に更新されます。このアルゴリズムは、すべてのクロック上で実行されます。クロックにポートが 1 つしかない場合、これは *マスター*にも *スレーブ*にもなることができ、*ordinary clock* (OC: 通常のクロック) と呼ばれます。複数のポートがあるクロックではあるポートではマスターに、別のポートではスレーブになることができ、*boundary* clock (BC: 境界クロック) と呼ばれます。トップレベルのクロックは *グランドマスタークロック*と呼ばれ、*全地球測位システム* (GPS) の時間ソースを使って同期できます。GPS ベースの時間ソースを使うことで、高度の正確性を保って異なるネットワークが同期可能になります。

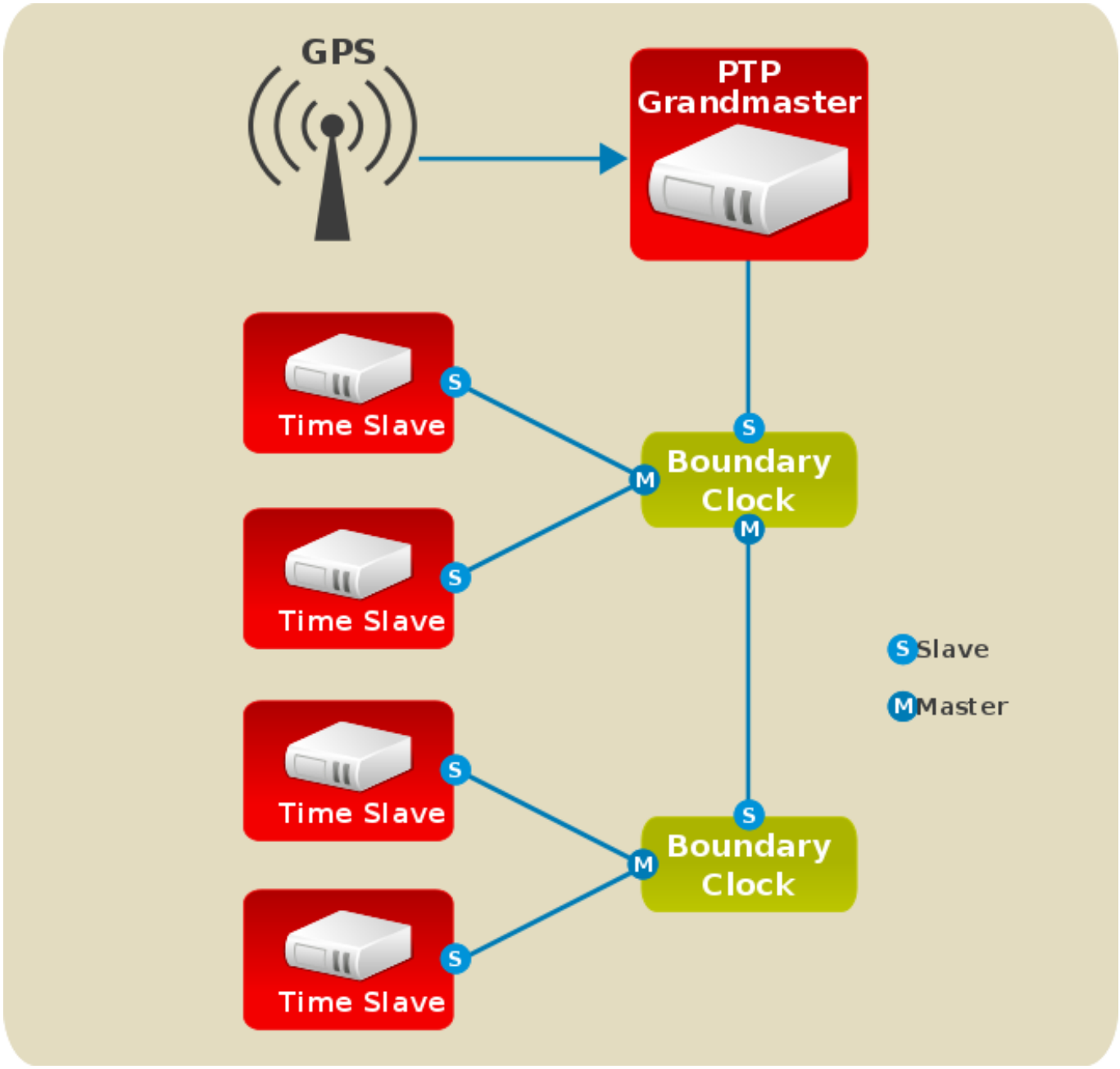


図15.1 PTP グランドマスター、境界、スレーブの各クロック

15.1.2. PTP の利点

PTP が *Network Time Protocol* (NTP) よりも優れている点の 1 つは、さまざまな*network interface controllers* (NIC) およびネットワークスイッチにあるハードウェアサポートです。この特化されたハードウェアにより、**PTP** はメッセージ送信の遅れを説明でき、時間同期の精度を大幅に高められます。ネットワーク内で PTP 以外に対応するハードウェアコンポーネントを使用することは可能ですが、その場合、変動が増えたり、遅れが非対称となったりして、同期が不正確になります。通信パスで使用される PTP を認識しないコンポーネントが複数あると、これがさらに増幅します。最高の精度を達成するには、**PTP** クロック間のすべてのネットワークコンポーネントが **PTP** ハードウェア対応となっていることが推奨されます。ネットワークハードウェアがのすべてが **PTP** をサポートしているわけではない大型のネットワークにおける時間同期には、**NTP** が適している場合があります。

ハードウェア **PTP** サポートでは、NIC には自身のボード上のクロックが備わります。これは、送受信される **PTP** メッセージのタイムスタンプに使われます。**PTP** マスターに同期されるのはこのボード上のクロックで、コンピューターのシステムクロックは NIC 上の **PTP** ハードウェアクロックに同期されます。ソフトウェア **PTP** サポートでは、システムクロックが**PTP** メッセージのタイムスタンプに使われ、直接

PTP マスターに同期されます。ソフトウェア**PTP** サポートではオペレーティングシステムによる追加の **PTP** パケット処理を必要としますが、ハードウェア**PTP** サポートでは、NIC が**PTP** パケットの送受信時の瞬間にタイムスタンプができるので、より優れた正確性が得られます。

15.2. PTP の使用

PTP を使用するには、意図するインターフェイス用のカーネルネットワークドライバーがソフトウェアもしくはハードウェアのタイムスタンプ機能をサポートしている必要があります。

15.2.1. ドライバーおよびハードウェアサポートの確認

ドライバーがハードウェアタイムスタンプをサポートしていることに加え、NIC も物理的なハードウェアでこの機能をサポートできる必要があります。特定のドライバーおよび NIC のタイムスタンプ機能を検証する最善の方法は、以下のように **ethtool** ユーティリティーを使ってインターフェイスにクエリーすることです。

```
~]# ethtool -T eth3
Time stamping parameters for eth3:
Capabilities:
        hardware-transmit     (SOF_TIMESTAMPING_TX_HARDWARE)
        software-transmit     (SOF_TIMESTAMPING_TX_SOFTWARE)
        hardware-receive      (SOF_TIMESTAMPING_RX_HARDWARE)
        software-receive      (SOF_TIMESTAMPING_RX_SOFTWARE)
        software-system-clock (SOF_TIMESTAMPING_SOFTWARE)
        hardware-raw-clock    (SOF_TIMESTAMPING_RAW_HARDWARE)
PTP Hardware Clock: 0
Hardware Transmit Timestamp Modes:
        off                   (HWTSTAMP_TX_OFF)
        on                    (HWTSTAMP_TX_ON)
Hardware Receive Filter Modes:
        none                  (HWTSTAMP_FILTER_NONE)
        all                   (HWTSTAMP_FILTER_ALL)
```

ここでの *eth3* は、チェックするインターフェイスになります。

ソフトウェアタイムスタンプのサポートについては、パラメーター一覧に以下を含めます。

- **SOF_TIMESTAMPING_SOFTWARE**
- **SOF_TIMESTAMPING_TX_SOFTWARE**
- **SOF_TIMESTAMPING_RX_SOFTWARE**

ハードウェアタイムスタンプのサポートについては、パラメーター一覧に以下を含めます。

- **SOF_TIMESTAMPING_RAW_HARDWARE**
- **SOF_TIMESTAMPING_TX_HARDWARE**
- **SOF_TIMESTAMPING_RX_HARDWARE**

15.2.2. PTP のインストール

Red Hat Enterprise Linux のカーネルには、**PTP** のサポートが含まれています。ユーザースペースのサポートは、**linuxptp** パッケージ内のツールが提供します。**linuxptp** をインストールするには、以下のコマンドを **root** で発行します。

```
~]# yum install linuxptp
```

これで **ptp4l** と **phc2sys** がインストールされます。

システムクロックの時間を設定するサービスを同時に 2 つ以上実行しないでください。**NTP** を使用して **PTP** 時間を実行したい場合は、「NTP を使った PTP 時間の実行」 を参照してください。

15.2.3. ptp4l の起動

ptp4l プログラムは、デフォルトでハードウェアタイムスタンプを使用します。ハードウェアタイムスタンプが可能なドライバーおよび NIC で **ptp4l** を使用するには、使用するネットワークインターフェイスに **-i** オプションを提供する必要があります。以下のコマンドを**root** で入力してください。

```
~]# ptp4l -i eth3 -m
```

ここでの *eth3* は、設定するインターフェイスになります。以下は、NIC 上の**PTP** クロックがマスターに同期された際の **ptp4l** からの出力例です。

```
~]# ptp4l -i eth3 -m
selected eth3 as PTP clock
port 1: INITIALIZING to LISTENING on INITIALIZE
port 0: INITIALIZING to LISTENING on INITIALIZE
port 1: new foreign master 00a069.fffe.0b552d-1
selected best master clock 00a069.fffe.0b552d
port 1: LISTENING to UNCALIBRATED on RS_SLAVE
master offset -23947 s0 freq +0 path delay        11350
master offset -28867 s0 freq +0 path delay        11236
master offset -32801 s0 freq +0 path delay        10841
master offset -37203 s1 freq +0 path delay        10583
master offset  -7275 s2 freq -30575 path delay    10583
port 1: UNCALIBRATED to SLAVE on MASTER_CLOCK_SELECTED
master offset  -4552 s2 freq -30035 path delay    10385
```

マスターオフセットの値は、マスターからナノ秒で測定されたオフセットです。**s0**、**s1**、**s2** の各ストリングは、異なるクロックのサーボ状態を示しています。**s0** はアンロック、**s1** はクロックステップ、**s2** はロックになります。サーボがロック状態 (**s2**) になると、(**ptp4l(8)** man ページの説明にあるように) **pi_offset_const** が設定ファイルでプラスの値に設定されている場合を除いて、クロックはステップされません (ゆっくり調整されるのみ)。**adj** の値は、10 億分の 1 (ppb) で表されるクロックの周波数調整です。path delay の値は、ナノ秒で表されるマスターから送信された同期メッセージの予想遅延です。Port 0 は、ローカル **PTP** 管理に使用される Unix ドメインソケットです。Port 1 は、**eth3** インターフェイスです (上記の例に基づく)。INITIALIZING、LISTENING、UNCALIBRATED、および SLAVE は、INITIALIZE、RS_SLAVE、MASTER_CLOCK_SELECTED イベントで変化する可能性のあるポート状態です。最後の状態変更メッセージでは、ポート状態が UNCALIBRATED から SLAVE に変更し、**PTP** マスタークロックとの同期が成功したことを示しています。

ptp4l プログラムは、以下のコマンドを実行してサービスとしても起動できます。

```
~]# systemctl start ptp4l
```

サービスとして実行する場合は、オプションは **/etc/sysconfig/ptp4l** ファイルで指定されます。**ptp4l** の他のオプションおよび設定ファイルについての詳細情報は、**ptp4l(8)** man ページで見ることができます。

デフォルトでは、メッセージは **/var/log/messages** に送信されます。しかし、**-m** オプションを指定すると標準出力へのロギングが可能になり、これはデバッグで役に立ちます。

ソフトウェアタイムスタンプを有効にするには、以下のように **-S** オプションを使用する必要があります。

```
~]# ptp4l -i eth3 -m -S
```

15.2.3.1. 遅延測定メカニズムの選択

遅延測定メカニズムには 2 種類あり、以下のように **ptp4l** コマンドにオプションを追加することで選択できます。

-P

-P オプションは *ピアツーピア*(P2P) 遅延測定メカニズムを選択します。

P2P メカニズムはネットワークトポロジーの変更に他のメカニズムよりも速く反応し、遅延の測定がより正確であることから、より好まれます。P2P メカニズムが使用できるのは、各ポートが最大で他の 1 つの P2P ポートで PTP メッセージを交換するトポロジーだけです。これは、透過クロックを含む通信パス上のすべてのハードウェアでサポートされ、使用される必要があります。

-E

-E オプションは、*エンドツーエンド*(E2E) 遅延測定メカニズムを選択します。これがデフォルトです。

E2E メカニズムは、遅延「リクエスト-レスポンス」 メカニズムとも呼ばれます。

-A

-A オプションは、遅延測定メカニズムの自動選択を有効にします。

自動オプションは、E2E モードで **ptp4l** を起動します。ピアの遅延リクエストを受け取ると、P2P モードに変更します。

注記

単一の **PTP** 通信パス上にあるクロックはすべて、遅延測定で同一のメカニズムを使用する必要があります。E2E メカニズムを使用しているポートでピアの遅延リクエストを受け取ると、警告が表示されます。P2P メカニズムを使用しているポートでE2E 遅延リクエストを受け取ると、警告が表示されます。

15.3. 設定ファイルの指定

コマンドラインオプションやコマンドラインで設定できないその他のオプションは、オプションの設定ファイルで設定することができます。

デフォルトでは、設定ファイルは読み取られないため、ランタイムで **-f** オプションを使って指定する必要があります。例を示します。

```
~]# ptp4l -f /etc/ptp4l.conf
```

上記の **-i eth3 -m -S** オプションと同等の設定ファイルは、以下のようになります。

```
~]# cat /etc/ptp4l.conf
[global]
```

```
verbose                 1
time_stamping           software
[eth3]
```

15.4. PTP 管理クライアントの使用

PTP 管理クライアントである**pmc** を使うと、以下のように**ptp4l** から追加情報を取得することができます。

```
~]# pmc -u -b 0 'GET CURRENT_DATA_SET'
sending: GET CURRENT_DATA_SET
        90e2ba.fffe.20c7f8-0 seq 0 RESPONSE MANAGMENT CURRENT_DATA_SET
                stepsRemoved        1
                offsetFromMaster  -142.0
                meanPathDelay     9310.0
```

```
~]# pmc -u -b 0 'GET TIME_STATUS_NP'
sending: GET TIME_STATUS_NP
        90e2ba.fffe.20c7f8-0 seq 0 RESPONSE MANAGMENT TIME_STATUS_NP
                master_offset              310
                ingress_time               1361545089345029441
                cumulativeScaledRateOffset   +1.000000000
                scaledLastGmPhaseChange    0
                gmTimeBaseIndicator        0
                lastGmPhaseChange          0x0000'0000000000000000.0000
                gmPresent                  true
                gmIdentity                 00a069.fffe.0b552d
```

-b オプションを **zero** に設定すると、範囲がローカルで実行している**ptp4l** インスタンスに制限されます。範囲の値を大きくすると、ローカルクロックに加えて **PTP** ノードからも情報を取得します。取得可能な情報には、以下のものが含まれます。

- **stepsRemoved** は、グランドマスタークロックへの通信パスの数です。
- **offsetFromMaster** および master_offset は、マスターから最後に測定されたオフセットをナノ秒で表します。
- **meanPathDelay** は、マスターから送信された同期メッセージの遅延予測をナノ秒で表します。
- **gmPresent** が true の場合、**PTP** クロックがマスターに同期され、ローカルクロックはグランドマスタークロックに同期されません。
- **gmIdentity** は、グランドマスターの ID です。

pmc コマンドの完全な一覧を表示するには、**root** で以下を入力します。

```
~]# pmc help
```

pmc(8) man ページでは詳細情報が見られます。

15.5. クロックの同期

システムクロックを NIC 上の **PTP** ハードウェアクロック (PHC) と同期するには、**phc2sys** プログラムを使用します。**phc2sys** を起動するには、*eth3* が **PTP** ハードウェアクロックとのインターフェイスの場合、**root** で以下のコマンドを入力します。

```
~]# phc2sys -s eth3 -w
```

-w オプションは、実行中の **ptp4l** アプリケーションが **PTP** クロックを同期するまで待機し、**ptp4l** から TAI から UTC のオフセットを取得します。

通常、**PTP** は *国際原子時* (TAI) のタイムスケールで作動し、システムクロックは *協定世界時* (UTC) で維持されます。TAI と UTC のタイムスケール間の現在のオフセットは、35 秒です。このオフセットは、うるう秒が追加もしくは取り除かれると変化します。通常、2、3 年ごとに起こります。**-O** オプションは、**-w** オプションを使用しない場合にこのオフセットを手動で設定する際に、以下のように使用する必要があります。

```
~]# phc2sys -s eth3 -O -35
```

phc2sys サーボがロック状態になると、**-S** オプションを使用しない限り、クロックはステップされません。つまり、**phc2sys** プログラムは、**ptp4l** プログラムが **PTP** ハードウェアクロックを同期した後に起動すべきということになります。ただし、**-w** オプションを使うと、**ptp4l** によるクロックの同期を **phc2sys** が待機するため、これを必ずしも後に起動する必要はなくなります。

phc2sys プログラムは、以下のコマンドを実行してサービスとしても起動できます。

```
~]# systemctl start phc2sys
```

サービスとして実行する場合は、オプションは **/etc/sysconfig/phc2sys** ファイルで指定されます。**phc2sys** の他のオプションについての詳細情報は、**phc2sys(8)** man ページで見ることができます。

本セクションの例では、コマンドがスレーブシステムまたはスレーブポートで実行されている想定であることに注意してください。

15.6. 時間同期の検証

PTP 時間同期が正常に機能している場合、オフセットおよび周波数調整のある新たなメッセージは定期的に **ptp4l** および **phc2sys** (ハードウェアタイムスタンプが使用されている場合) 出力に表示されます。これらの値は、短時間の後に最終的には収束されます。これらのメッセージは、**/var/log/messages** ファイルで見ることができます。出力の例を以下に示します。

```
ptp4l[352.359]: selected /dev/ptp0 as PTP clock
ptp4l[352.361]: port 1: INITIALIZING to LISTENING on INITIALIZE
ptp4l[352.361]: port 0: INITIALIZING to LISTENING on INITIALIZE
ptp4l[353.210]: port 1: new foreign master 00a069.fffe.0b552d-1
ptp4l[357.214]: selected best master clock 00a069.fffe.0b552d
ptp4l[357.214]: port 1: LISTENING to UNCALIBRATED on RS_SLAVE
ptp4l[359.224]: master offset       3304 s0 freq      +0 path delay
9202
ptp4l[360.224]: master offset       3708 s1 freq  -29492 path delay
9202
ptp4l[361.224]: master offset      -3145 s2 freq  -32637 path delay
9202
ptp4l[361.224]: port 1: UNCALIBRATED to SLAVE on MASTER_CLOCK_SELECTED
ptp4l[362.223]: master offset       -145 s2 freq  -30580 path delay
```

```
9202
ptp4l[363.223]: master offset       1043 s2 freq  -29436 path delay
8972
ptp4l[364.223]: master offset        266 s2 freq  -29900 path delay
9153
ptp4l[365.223]: master offset        430 s2 freq  -29656 path delay
9153
ptp4l[366.223]: master offset        615 s2 freq  -29342 path delay
9169
ptp4l[367.222]: master offset       -191 s2 freq  -29964 path delay
9169
ptp4l[368.223]: master offset        466 s2 freq  -29364 path delay
9170
ptp4l[369.235]: master offset         24 s2 freq  -29666 path delay
9196
ptp4l[370.235]: master offset       -375 s2 freq  -30058 path delay
9238
ptp4l[371.235]: master offset        285 s2 freq  -29511 path delay
9199
ptp4l[372.235]: master offset        -78 s2 freq  -29788 path delay
9204
```

phc2sys 出力の例を以下に示します。

```
phc2sys[526.527]: Waiting for ptp4l...
phc2sys[527.528]: Waiting for ptp4l...
phc2sys[528.528]: phc offset     55341 s0 freq      +0 delay   2729
phc2sys[529.528]: phc offset     54658 s1 freq  -37690 delay   2725
phc2sys[530.528]: phc offset       888 s2 freq  -36802 delay   2756
phc2sys[531.528]: phc offset      1156 s2 freq  -36268 delay   2766
phc2sys[532.528]: phc offset       411 s2 freq  -36666 delay   2738
phc2sys[533.528]: phc offset       -73 s2 freq  -37026 delay   2764
phc2sys[534.528]: phc offset        39 s2 freq  -36936 delay   2746
phc2sys[535.529]: phc offset        95 s2 freq  -36869 delay   2733
phc2sys[536.529]: phc offset      -359 s2 freq  -37294 delay   2738
phc2sys[537.529]: phc offset      -257 s2 freq  -37300 delay   2753
phc2sys[538.529]: phc offset       119 s2 freq  -37001 delay   2745
phc2sys[539.529]: phc offset       288 s2 freq  -36796 delay   2766
phc2sys[540.529]: phc offset      -149 s2 freq  -37147 delay   2760
phc2sys[541.529]: phc offset      -352 s2 freq  -37395 delay   2771
phc2sys[542.529]: phc offset       166 s2 freq  -36982 delay   2748
phc2sys[543.529]: phc offset        50 s2 freq  -37048 delay   2756
phc2sys[544.530]: phc offset       -31 s2 freq  -37114 delay   2748
phc2sys[545.530]: phc offset      -333 s2 freq  -37426 delay   2747
phc2sys[546.530]: phc offset       194 s2 freq  -36999 delay   2749
```

ptp4l には、出力を短縮して統計情報のみを表示する**summary_interval** 指示文もあります。通常は、毎秒ごとにメッセージが表示されます。たとえば、出力を **1024** 秒ごとにするには、以下の行を **/etc/ptp4l.conf** ファイルに追加します。

```
summary_interval 10
```

ptp4l の **summary_interval 6** オプションの出力例は、以下のようになります。

```
ptp4l: [615.253] selected /dev/ptp0 as PTP clock
```

```
ptp4l: [615.255] port 1: INITIALIZING to LISTENING on INITIALIZE
ptp4l: [615.255] port 0: INITIALIZING to LISTENING on INITIALIZE
ptp4l: [615.564] port 1: new foreign master 00a069.fffe.0b552d-1
ptp4l: [619.574] selected best master clock 00a069.fffe.0b552d
ptp4l: [619.574] port 1: LISTENING to UNCALIBRATED on RS_SLAVE
ptp4l: [623.573] port 1: UNCALIBRATED to SLAVE on MASTER_CLOCK_SELECTED
ptp4l: [684.649] rms  669 max 3691 freq -29383 ± 3735 delay  9232 ± 122
ptp4l: [748.724] rms  253 max  588 freq -29787 ± 221 delay  9219 ± 158
ptp4l: [812.793] rms  287 max  673 freq -29802 ± 248 delay  9211 ± 183
ptp4l: [876.853] rms  226 max  534 freq -29795 ± 197 delay  9221 ± 138
ptp4l: [940.925] rms  250 max  562 freq -29801 ± 218 delay  9199 ± 148
ptp4l: [1004.988] rms  226 max  525 freq -29802 ± 196 delay  9228 ± 143
ptp4l: [1069.065] rms  300 max  646 freq -29802 ± 259 delay  9214 ± 176
ptp4l: [1133.125] rms  226 max  505 freq -29792 ± 197 delay  9225 ± 159
ptp4l: [1197.185] rms  244 max  688 freq -29790 ± 211 delay  9201 ± 162
```

phc2sys からの出力を短くするには、以下のように**-u** オプションを使用します。

```
~]# phc2sys -u summary-updates
```

ここでの *summary-updates* は、統計情報に含めるクロック更新数です。例を示します。

```
~]# phc2sys -s eth3 -w -m -u 60
phc2sys[700.948]: rms 1837 max 10123 freq -36474 ± 4752 delay  2752 ±  16
phc2sys[760.954]: rms  194 max  457 freq -37084 ± 174 delay  2753 ±  12
phc2sys[820.963]: rms  211 max  487 freq -37085 ± 185 delay  2750 ±  19
phc2sys[880.968]: rms  183 max  440 freq -37102 ± 164 delay  2734 ±  91
phc2sys[940.973]: rms  244 max  584 freq -37095 ± 216 delay  2748 ±  16
phc2sys[1000.979]: rms  220 max  573 freq -36666 ± 182 delay  2747 ±  43
phc2sys[1060.984]: rms  266 max  675 freq -36759 ± 234 delay  2753 ±  17
```

15.7. NTP を使った PTP 時間の実行

ntpd デーモンは、**ptp4l** または **phc2sys** で同期されたシステムクロック、もしくは LOCAL 参照クロックドライバーを使用して同期されたシステムクロックの時間を配布するように設定可能です。**ntpd** がシステムクロックを調整しないようにするには、**ntp.conf** ファイルで **NTP** サーバーを指定しないようにします。以下に、**ntp.conf** の最小限の例を示します。

```
~]# cat /etc/ntp.conf
server   127.127.1.0
fudge    127.127.1.0 stratum 0
```

注記

DHCP クライアントプログラムである**dhclient** は、**DHCP** サーバーから **NTP** サーバーのリストを受信すると、これに **ntp.conf** を追加してサービスを再起動します。この機能を無効にするには、**PEERNTP=no** を **/etc/sysconfig/network** に追加します。

15.8. PTP を使った NTP 時間の実行

NTP から **PTP** への逆方向の同期も可能です。**ntpd** を使ってシステムクロックを同期する場合、**ptp4l** を **priority1** オプションで (または最善のマスタークロックアルゴリズムに含まれている他のクロックオプションで) グランドマスタークロックに設定して **PTP** 経由でシステムクロックから時間を配布することができます。

```
~]# cat /etc/ptp4l.conf
[global]
priority1 127
[eth3]
# ptp4l -f /etc/ptp4l.conf
```

ハードウェアタイムスタンプでは、**phc2sys** を使って **PTP** ハードウェアクロックをシステムクロックに同期する必要があります。

```
~]# phc2sys -c eth3 -s CLOCK_REALTIME -w
```

PTP クロックの短時間の周波数変更を避けるには、PI サーボでより小さい**P** (比例) および **I** (積分) の各定数を使用することでシステムクロックへの同期を緩やかにすることができます。

```
~]# phc2sys -c eth3 -s CLOCK_REALTIME -w -P 0.01 -I 0.0001
```

15.9. 精度の向上

テストの結果では、ティックレスカーネル機能を無効にするとシステムクロックの安定性が大幅に改善され、**PTP** 同期の精度が高まることが示されています (電力消費量は高まります)。カーネルのティックレスモードは、**nohz=off** をカーネル起動オプションパラメーターに追加することで無効にできます。

15.10. その他のリソース

以下の情報ソースで **PTP** および **ptp4l** ツールに関する追加リソースが提供されています。

15.10.1. インストールされているドキュメント

- **ptp4l(8)** man ページ — 設定ファイルの形式を含む **ptp4l** オプションを説明しています。
- **pmc(8)** man ページ — **PTP** 管理クライアントおよびそのコマンドオプションを説明しています。
- **phc2sys(8)** man page — システムクロックを **PTP** ハードウェアクロック (PHC) に同期させるツールを説明しています。

15.10.2. 役に立つ Web ページ

http://www.nist.gov/el/isd/ieee/ieee1588.cfm

IEEE 1588 規格

パート V. 監視と自動化

ここでは、システム管理者がシステムパフォーマンスのモニター、システムタスクの自動化、バグの報告を行うための各種ツールについて解説します。

第16章 システムモニタリングツール

システムを設定するには、多くの場合システム管理者は空きメモリの容量、空きディスク領域、ハードディスクのパーティション設定状況、実行中のプロセスを決定する必要があります。

16.1. システムプロセスの表示

16.1.1. ps コマンドの使用

ps コマンドは、実行中のプロセスについての情報を表示します。静的な一覧、すなわちコマンドを実行するときに実行しているプロセスのスナップショットです。実行中のプロセスを定期的に更新した一覧を表示させるには、**top** コマンドまたは**システムモニター**アプリケーションを代わりに使用します。

他のユーザーが所有しているプロセスを含め、現在システム上で実行中の全プロセスを一覧表示するには、シェルプロンプトで以下を入力します。

```
ps ax
```

一覧表示された各プロセスについて **ps ax** コマンドが表示するのは次のとおりです。プロセス ID (**PID**)、それに関連付けされたターミナル (**TTY**)、現在のステータス (**STAT**)、累積 CPU 時間 (**TIME**)、実行ファイル名 (**COMMAND**) です。例を示します。

```
~]$ ps ax
  PID TTY      STAT   TIME COMMAND
    1 ?        Ss     0:01 /sbin/init
    2 ?        S      0:00 [kthreadd]
    3 ?        S      0:00 [migration/0]
    4 ?        S      0:00 [ksoftirqd/0]
    5 ?        S      0:00 [migration/0]
    6 ?        S      0:00 [watchdog/0][出力は省略されています]
```

各プロセスと同時に所有者も表示するには、以下のコマンドを使用します。

```
ps aux
```

ps ax コマンドで提供される情報以外に、**ps aux** はプロセス所有者の有効なユーザー名 (**USER**)、CPU のパーセンテージ (**%CPU**) およびメモリ使用率 (**%MEM**)、キロバイト単位での仮想メモリサイズ (**VSZ**)、キロバイト単位での非スワップの物理メモリサイズ (**RSS**)、プロセスの開始日時を表示します。例を示します。

```
~]$ ps aux
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root          1  0.0  0.1  19404   832 ?        Ss   Mar02   0:01
/sbin/init
root          2  0.0  0.0      0     0 ?        S    Mar02   0:00
[kthreadd]
root          3  0.0  0.0      0     0 ?        S    Mar02   0:00
[migration/0]
root          4  0.0  0.0      0     0 ?        S    Mar02   0:00
[ksoftirqd/0]
```

```
root         5  0.0  0.0      0     0 ?        S    Mar02   0:00
[migration/0]
root         6  0.0  0.0      0     0 ?        R    Mar02   0:00
[watchdog/0][出力は省略されています]
```

ps コマンドを **grep** と組み合わせて使用して、特定のプロセスが実行中かどうかを確認することもできます。例えば、**Emacs** が実行中かどうかを知るには、以下を入力します。

```
~]$ ps ax | grep emacs
12056 pts/3    S+     0:00 emacs
12060 pts/2    S+     0:00 grep --color=auto emacs
```

利用可能なコマンドラインオプションの一覧は、**ps**(1) の man ページを参照してください。

16.1.2. top コマンドの使用

top コマンドは、システム上で実行中のプロセスの一覧をリアルタイムで表示します。また、システムのアップタイム、現在の CPU およびメモリ使用率、実行中のプロセスの合計数についての追加情報も表示します。さらには、一覧の並び替えやプロセスの kill などの操作も実行できます。

top コマンドを実行するには、シェルプロンプトで以下を入力します。

```
top
```

一覧表示された各プロセスについて **top** コマンドが表示するのは次のとおりです。プロセス ID (**PID**)、プロセス所有者の有効なユーザー名、(**USER**)、優先度 (**PR**)、nice 値 (**NI**)、プロセスが使用する仮想メモリ容量 (**VIRT**)、プロセスが使用する非スワップの物理メモリ容量 (**RES**)、プロセスが使用する共有メモリ容量 (**SHR**)、CPU のパーセンテージ (**%CPU**) およびメモリ使用率 (**%MEM**)、累積 CPU 時間 (**TIME+**)、実行ファイル名 (**COMMAND**) です。例を示します。

```
~]$ top
top - 02:19:11 up 4 days, 10:37,  5 users,  load average: 0.07, 0.13,
0.09
Tasks: 160 total,   1 running, 159 sleeping,   0 stopped,   0 zombie
Cpu(s): 10.7%us,  1.0%sy,  0.0%ni, 88.3%id,  0.0%wa,  0.0%hi,  0.0%si,
0.0%st
Mem:    760752k total,   644360k used,   116392k free,     3988k buffers
Swap:  1540088k total,    76648k used,  1463440k free,   196832k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
14401 jhradile  20   0  313m  10m 5732 S  5.6  1.4   6:27.29 gnome-
system-mo
 1764 root      20   0  133m  23m 4756 S  5.3  3.2   6:32.66 Xorg
13865 jhradile  20   0 1625m 177m 6628 S  0.7 23.8   0:57.26 java
   20 root      20   0     0    0    0 S  0.3  0.0   4:44.39 ata/0
 2085 root      20   0 40396  348  276 S  0.3  0.0   1:57.13 udisks-
daemon
    1 root      20   0 19404  832  604 S  0.0  0.1   0:01.21 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.01 kthreadd
    3 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
    4 root      20   0     0    0    0 S  0.0  0.0   0:00.02 ksoftirqd/0
    5 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
    6 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 watchdog/0
    7 root      20   0     0    0    0 S  0.0  0.0   0:01.00 events/0
    8 root      20   0     0    0    0 S  0.0  0.0   0:00.00 cpuset
```

```
    9 root      20   0     0     0     0 S  0.0  0.0   0:00.00 khelper
   10 root      20   0     0     0     0 S  0.0  0.0   0:00.00 netns
   11 root      20   0     0     0     0 S  0.0  0.0   0:00.00 async/mgr
   12 root      20   0     0     0     0 S  0.0  0.0   0:00.00 pm[出力は省略
されています]
```

表16.1「インタラクティブな top コマンド」では、**top** で使用可能なインタラクティブコマンドを表示しています。詳細は、**top**(1) の man ページを参照してください。

表16.1 インタラクティブな top コマンド

コマンド	詳細
Enter、**Space**	表示を最新の情報に直ちに更新します。
h、**?**	ヘルプ画面を表示します。
k	プロセスを kill します。プロセス ID およびプロセスに送信するシグナルがプロンプトされます。
n	表示されるプロセス番号を変更します。番号を入力するようプロンプトされます。
u	一覧をユーザー別に並べ替えます。
M	一覧をメモリ使用率で並べ替えます。
P	一覧を CPU 使用率で並べ替えます。
q	ユーティリティを終了して、シェルプロンプトに戻ります。

16.1.3. システムモニターツールの使用

システムモニターツールの **プロセス** タブを使用することで、グラフィカルユーザーインターフェースからプロセスの表示、検索、優先度の変更、kill を行うことができます。

システムモニターツールを起動するには、パネルから**アプリケーション** → **システムツール** → **システムモニター** の順に選択するか、シェルプロンプトで**gnome-system-monitor** と入力します。その後、**プロセス** タブをクリックすると実行中のプロセスの一覧が表示されます。

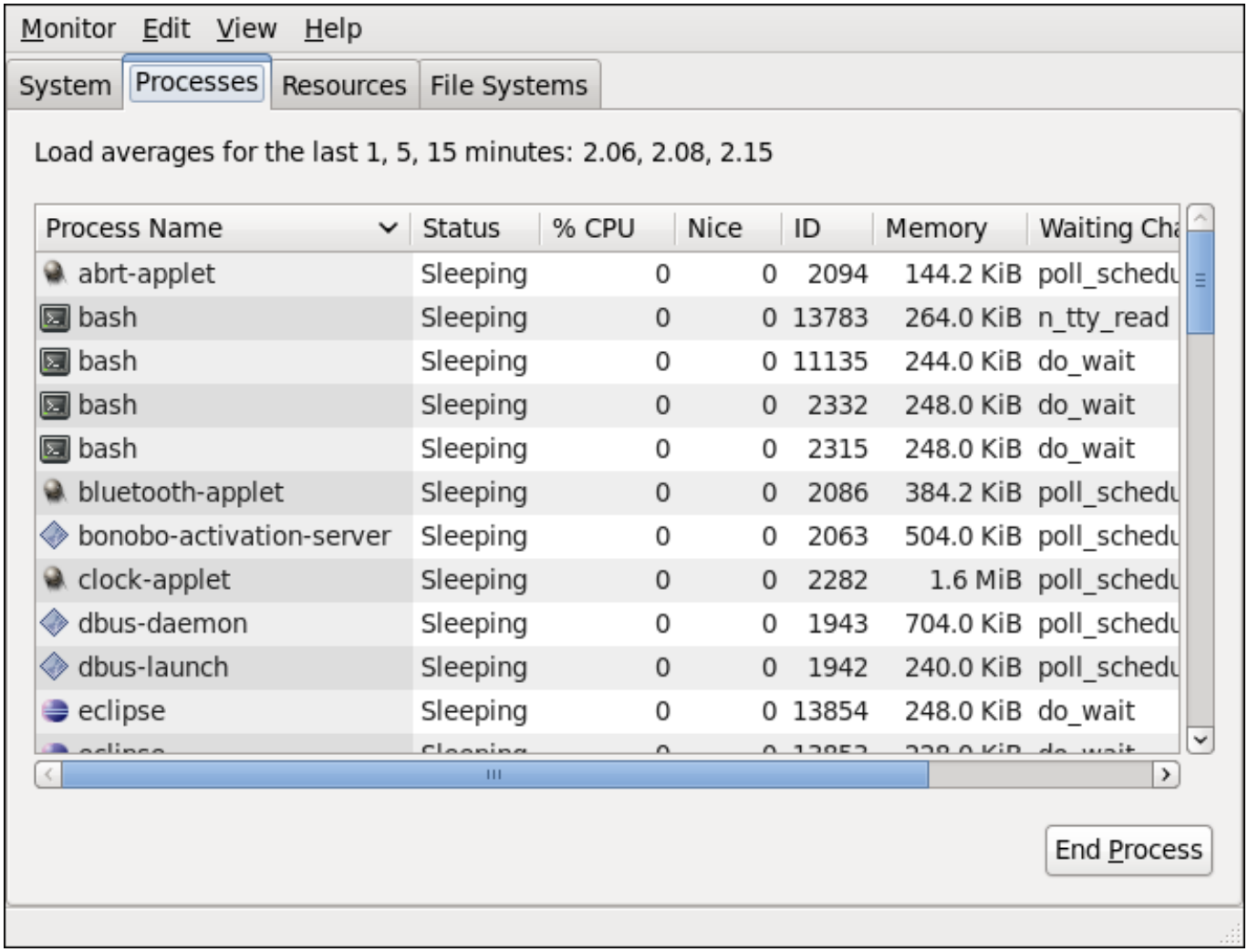


図16.1 システムモニター — プロセス

一覧表示された各プロセスについて **システムモニター**ツールが表示するのは次のとおりです。プロセス名 (**Process Name**)、現在の状態 (**Status**)、CPU 使用量のパーセンテージ (**% CPU**)、nice 値 (**Nice**)、プロセス ID (**ID**)、メモリ使用量 (**Memory**)、プロセスが待機しているチャンネル (**Waiting Channel**)、セッションについての補足情報 (**Session**) です。特定のコラム別に情報を昇順で並び替えるには、コラム名をクリックします。再度コラム名をクリックすると、昇順と降順が切り替わります。

デフォルトでは、**システムモニター**ツールは現在のユーザーが所有しているプロセスの一覧を表示します。**表示** メニューから各種オプションを選択すると、以下を実行できます。

- 実行中のプロセスのみの表示
- すべてのプロセスの表示
- ユーザーのプロセスの表示
- プロセスの依存関係の表示
- 選択したプロセスのメモリマップの表示
- 選択したプロセスで開いたファイルの表示
- プロセス一覧を最新の情報に更新

また、**編集** メニューの各種オプションを使用すると以下を実行できます。

- プロセスの停止

- プロセスの再開
- プロセスの終了
- プロセスの kill (強制終了)
- 選択したプロセスの優先度の変更
- **システムモニター**設定 (プロセスの一覧を更新する間隔や表示する情報) の編集

プロセスを終了するには、一覧からプロセスを選択して、**プロセスの終了** ボタンをクリックしても可能です。

16.2. メモリ使用量の表示

16.2.1. free コマンドの使用

free コマンドは、システムのメモリの空き容量と使用量を表示します。シェルプロンプトで以下を入力してください。

```
free
```

free コマンドは、物理メモリ (**Mem**) およびスワップ領域 (**Swap**) に関する情報を提供します。メモリの合計量 (**total**) だけでなく、使用メモリ容量 (**used**)、空きメモリ容量 (**free**)、共有メモリ容量 (**shared**)、カーネルバッファに使用されるメモリ容量 (**buffers**)、キャッシュに使用されるメモリ容量 (**cached**) を表示します。例を示します。

```
~]$ free
             total       used       free     shared    buffers     cached
Mem:        760752     661332      99420          0       6476     317200
-/+ buffers/cache:     337656     423096
Swap:      1540088     283652    1256436
```

デフォルトでは、**free** は値をキロバイトで表示します。値をメガバイトで表示するには、**-m** コマンドラインオプションを指定してください。

```
free -m
```

例を示します。

```
~]$ free -m
             total       used       free     shared    buffers     cached
Mem:           742        646         96          0          6
309
-/+ buffers/cache:        330        412
Swap:         1503        276       1227
```

利用可能なコマンドラインオプションの一覧は、**free**(1) の man ページを参照してください。

16.2.2. システムモニターツールの使用

システムモニターツールの **リソース** タブは、システムのメモリの空き容量と使用量を表示します。

システムモニターツールを起動するには、パネルから**アプリケーション** → **システムツール** → **システムモニター** の順に選択するか、シェルプロンプトで**gnome-system-monitor** を入力します。その後、**リソース** タブをクリックするとシステムのメモリ使用率が表示されます。

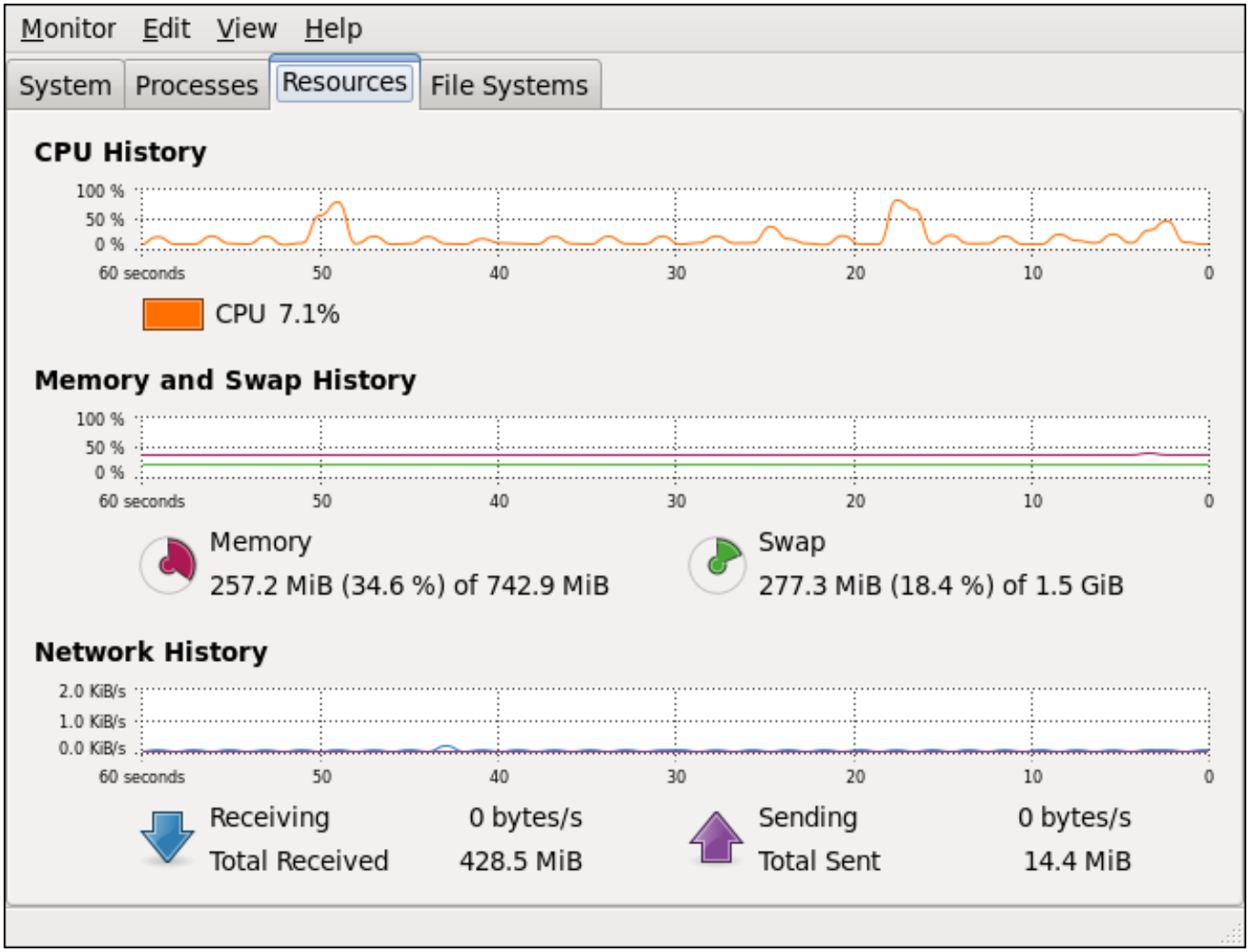


図16.2 システムモニター — リソース

システムモニターツールは、**メモリとスワップの履歴** のセクションでメモリおよびスワップの使用履歴、さらには物理メモリの総容量 (**メモリ**)、スワップ領域 (**スワップ**)、使用量をグラフィカルに表示します。

16.3. CPU 使用率の表示

16.3.1. システムモニターツールの使用

システムモニターツールの **リソース** タブは、システムの現在の CPU 使用率を表示します。

システムモニターツールを起動するには、パネルから**アプリケーション** → **システムツール** → **システムモニター** の順に選択するか、シェルプロンプトで**gnome-system-monitor** を入力します。その後、**リソース** タブをクリックするとシステムの CPU 使用率が表示されます。

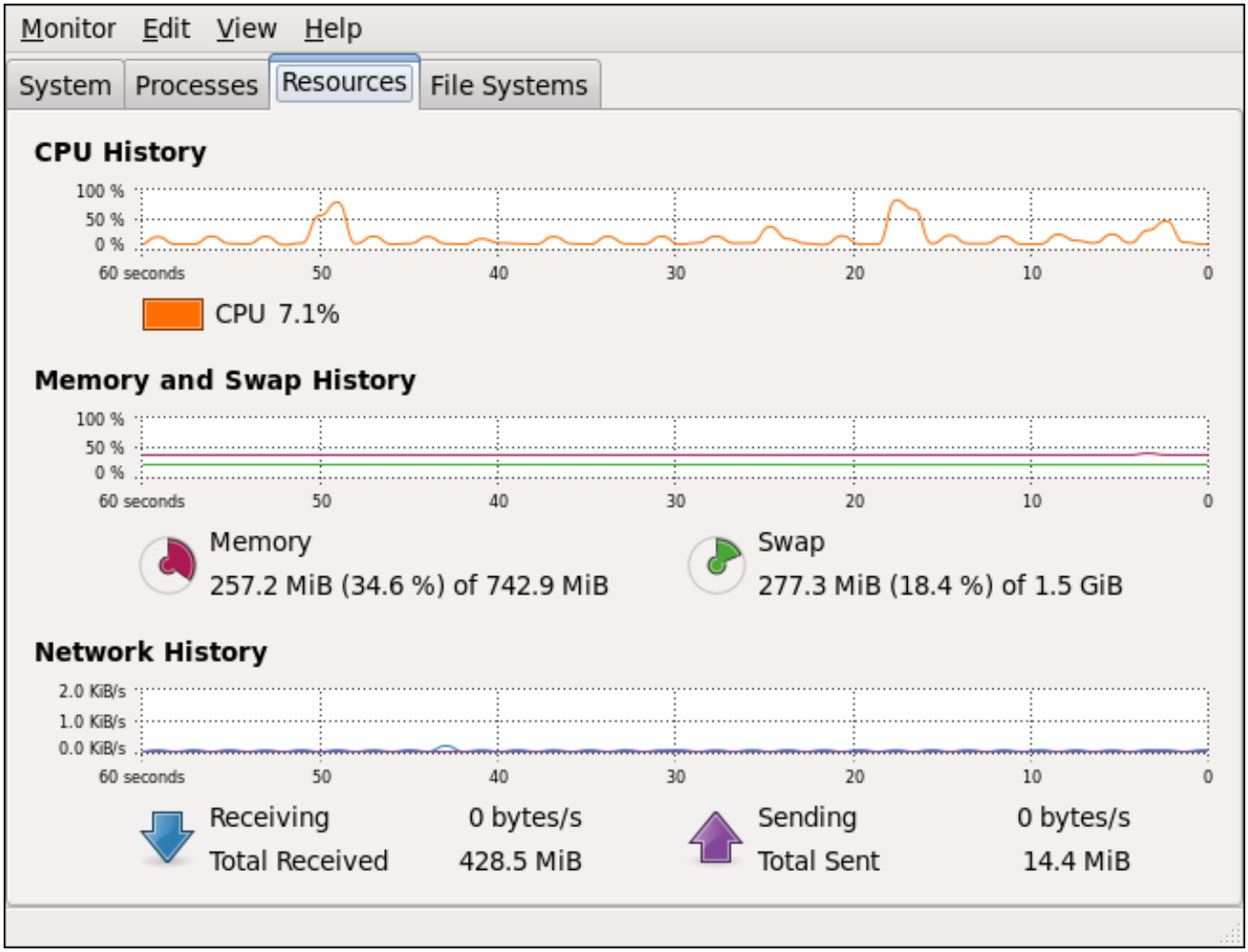


図16.3 システムモニター — リソース

システムモニターツールの**CPU 使用率の履歴** セクションでは、CPU 使用率の履歴をグラフィカルに表示するとともに現在使用中の CPU をパーセンテージで表示します。

16.4. ブロックデバイスとファイルシステムの表示

16.4.1. lsblk コマンドの使用

lsblk コマンドは、利用可能なブロックデバイスの一覧を表示します。シェルプロンプトで以下を入力してください。

```
lsblk
```

一覧表示された各ブロックデバイスについて **lsblk** コマンドが表示するのは次のとおりです。デバイス名 (**NAME**)、メジャーおよびマイナーデバイス番号 (**MAJ:MIN**)、リムーバブルデバイスかどうか (**RM**)、そのサイズ (**SIZE**)、読み取り専用デバイスかどうか (**RO**)、そのタイプ (**TYPE**)、デバイスのマウント先 (**MOUNTPOINT**) です。例を示します。

```
~]$ lsblk
NAME                    MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sr0                      11:0    1  1024M  0 rom
vda                     252:0    0    20G  0 rom
```

```
|-vda1                     252:1    0   500M  0 part /boot
`-vda2                     252:2    0  19.5G  0 part
  |-vg_kvm-lv_root (dm-0) 253:0    0    18G  0 lvm  /
  `-vg_kvm-lv_swap (dm-1) 253:1    0   1.5G  0 lvm  [SWAP]
```

デフォルトでは、**lsblk** はツリーのような形式でブロックデバイスを一覧表示します。通常の一覧として表示するには、**-l** コマンドラインオプションを追加します。

```
lsblk -l
```

例を示します。

```
~]$ lsblk -l
NAME                    MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sr0                      11:0    1  1024M  0 rom
vda                     252:0    0    20G  0 rom
vda1                    252:1    0   500M  0 part /boot
vda2                    252:2    0  19.5G  0 part
vg_kvm-lv_root (dm-0) 253:0    0    18G  0 lvm  /
vg_kvm-lv_swap (dm-1) 253:1    0   1.5G  0 lvm  [SWAP]
```

利用可能なコマンドラインオプションの一覧は、**lsblk**(8) の man ページを参照してください。

16.4.2. blkid コマンドの使用

blkid コマンドは利用可能なブロックデバイスに関する情報を表示します。**root** としてシェルプロンプトで以下を入力してください。

```
blkid
```

一覧表示された各ブロックデバイスについて **blkid** コマンドは、*汎用一意識別子* (**UUID**)、ファイルシステムのタイプ (**TYPE**)、ボリュームラベル (**LABEL**) などの属性を表示します。例を示します。

```
~]# blkid
/dev/vda1: UUID="7fa9c421-0054-4555-b0ca-b470a97a3d84" TYPE="ext4"
/dev/vda2: UUID="7IvYzk-TnnK-oPjf-ipdD-cofz-DXaJ-gPdgBW"
TYPE="LVM2_member"
/dev/mapper/vg_kvm-lv_root: UUID="a07b967c-71a0-4925-ab02-aebcad2ae824"
TYPE="ext4"
/dev/mapper/vg_kvm-lv_swap: UUID="d7ef54ca-9c41-4de4-ac1b-4193b0c1ddb6"
TYPE="swap"
```

デフォルトでは、**lsblk** コマンドはすべての利用可能なブロックデバイスを一覧表示します。特定のデバイスの情報のみを表示するには、コマンドラインでデバイス名を指定します。

```
blkid device_name
```

例えば、**/dev/vda1** に関する情報を表示するには、以下を入力してください。

```
~]# blkid /dev/vda1
/dev/vda1: UUID="7fa9c421-0054-4555-b0ca-b470a97a3d84" TYPE="ext4"
```

上記のコマンドに **-p** および **-o udev** のコマンドラインオプションを使用して、さらに詳しい情報を取得することも可能です。このコマンドを実行するには、**root** 権限が必要な点に注意してください。

```
blkid -po udev device_name
```

例:

```
~]# blkid -po udev /dev/vda1
ID_FS_UUID=7fa9c421-0054-4555-b0ca-b470a97a3d84
ID_FS_UUID_ENC=7fa9c421-0054-4555-b0ca-b470a97a3d84
ID_FS_VERSION=1.0
ID_FS_TYPE=ext4
ID_FS_USAGE=filesystem
```

利用可能なコマンドラインオプションの一覧は、**blkid**(8) の man ページを参照してください。

16.4.3. findmnt コマンドの使用

findmnt コマンドは、現在マウントされているファイルシステムの一覧を表示します。シェルプロンプトで以下を入力します。

```
findmnt
```

一覧表示された各ファイルシステムについて **findmnt** コマンドが表示するのは、次のとおりです。マウントポイント先 (**TARGET**)、ソースデバイス (**SOURCE**)、ファイルシステムのタイプ (**FSTYPE**)、関連するマウントオプション (**OPTIONS**) です。例を示します。

```
~]$ findmnt
TARGET                          SOURCE                          FSTYPE   OPTIONS
/                               /dev/mapper/vg_kvm-lv_root      ext4
rw,relatime,sec
|-/proc                         /proc                           proc
rw,relatime
| |-/proc/bus/usb               /proc/bus/usb                   usbfs
rw,relatime
| `-/proc/sys/fs/binfmt_misc                                    binfmt_m
rw,relatime
|-/sys                          /sys                            sysfs
rw,relatime,sec
|-/selinux                                                       selinuxf
rw,relatime
|-/dev                          udev                            devtmpfs
rw,relatime,sec
| `-/dev                        udev                            devtmpfs
rw,relatime,sec
|   |-/dev/pts                  devpts                          devpts
rw,relatime,sec
|   `-/dev/shm                  tmpfs                           tmpfs
rw,relatime,sec
|-/boot                         /dev/vda1                       ext4
rw,relatime,sec
|-/var/lib/nfs/rpc_pipefs       sunrpc                          rpc_pipe
rw,relatime
```

```
|-/misc                      /etc/auto.misc            autofs
rw,relatime,fd=
`-/net                       -hosts                    autofs
rw,relatime,fd=[出力は省略されています]
```

デフォルトでは、**findmnt** はツリーのような形式でファイルシステムを一覧表示します。通常の一覧として表示するには、**-l** コマンドラインオプションを追加します。

```
findmnt -l
```

例を示します。

```
~]$ findmnt -l
TARGET                   SOURCE                      FSTYPE   OPTIONS
/proc                    /proc                       proc     rw,relatime
/sys                     /sys                        sysfs
rw,relatime,seclabe
/dev                     udev                        devtmpfs
rw,relatime,seclabe
/dev/pts                 devpts                      devpts
rw,relatime,seclabe
/dev/shm                 tmpfs                       tmpfs
rw,relatime,seclabe
/                        /dev/mapper/vg_kvm-lv_root ext4
rw,relatime,seclabe
/selinux                                             selinuxf rw,relatime
/dev                     udev                        devtmpfs
rw,relatime,seclabe
/proc/bus/usb            /proc/bus/usb               usbfs    rw,relatime
/boot                    /dev/vda1                   ext4
rw,relatime,seclabe
/proc/sys/fs/binfmt_misc                             binfmt_m rw,relatime
/var/lib/nfs/rpc_pipefs  sunrpc                      rpc_pipe rw,relatime
/misc                    /etc/auto.misc              autofs
rw,relatime,fd=7,pg
/net                     -hosts                      autofs
rw,relatime,fd=13,p[出力は省略されています]
```

特定のタイプのファイルシステムのみを一覧表示するよう選択することも可能です。そのためには、**-t** コマンドラインオプション、その次にファイルシステムのタイプを追加します。

```
findmnt -t type
```

例えば、**ext4** ファイルシステムをすべて表示するには以下を入力します。

```
~]$ findmnt -t ext4
TARGET SOURCE                     FSTYPE OPTIONS
/      /dev/mapper/vg_kvm-lv_root ext4
rw,relatime,seclabel,barrier=1,data=ord
/boot  /dev/vda1                  ext4
rw,relatime,seclabel,barrier=1,data=ord
```

利用可能なコマンドラインオプションの一覧は、**findmnt**(8) の man ページを参照してください。

16.4.4. df コマンドの使用

df コマンドは、システムのディスク領域の使用量についての詳しいレポートを表示します。シェルプロンプトで以下を入力してください。

```
df
```

一覧表示された各ファイルシステムについて **df** コマンドが表示するのは次のとおりです。ファイルシステム名 (**Filesystem**)、サイズ (**1K-blocks** または **Size**)、使用領域 (**Used**)、利用可能な領域 (**Available**)、使用領域のパーセンテージ (**Use%**)、ファイルシステムのマウント先 (**Mounted on**) です。例を示します。

```
~]$ df
Filesystem                   1K-blocks      Used Available Use% Mounted on
/dev/mapper/vg_kvm-lv_root  18618236   4357360  13315112  25% /
tmpfs                         380376       288    380088   1% /dev/shm
/dev/vda1                     495844     77029    393215  17% /boot
```

デフォルトでは、**df** コマンドは 1 キロバイトブロック単位でパーティションのサイズを、キロバイト単位で使用中および利用可能なディスク領域の容量を表示します。この情報をメガバイトとギガバイトで表示するには、**-h** コマンドラインオプションを指定すると**df** がヒューマンリーダブルな形式で値を表示します。

```
df -h
```

例を示します。

```
~]$ df -h
Filesystem                  Size  Used Avail Use% Mounted on
/dev/mapper/vg_kvm-lv_root   18G  4.2G   13G  25% /
tmpfs                       372M  288K  372M   1% /dev/shm
/dev/vda1                   485M   76M  384M  17% /boot
```

利用可能なコマンドラインオプションの一覧は、**df**(1) の man ページを参照してください。

16.4.5. du コマンドの使用

du コマンドはディレクトリー内のファイルが使用している領域を表示します。現在の作業ディレクトリー内のサブディレクトリー用のディスク使用量を表示するには、コマンドラインオプションなしで以下のコマンドを実行します。

```
du
```

例:

```
~]$ du
14972   ./Downloads
4       ./.gnome2
4       ./.mozilla/extensions
4       ./.mozilla/plugins
12      ./.mozilla
15004   .
```

デフォルトでは、**du** コマンドはキロバイト単位でディスク使用量を表示します。メガバイトおよびギガバイト単位で表示するには、**-h** コマンドラインオプションを指定するとヒューマンリーダブルな形式で値を表示することができます。

```
du -h
```

例を示します。

```
~]$ du -h
15M     ./Downloads
4.0K    ./.gnome2
4.0K    ./.mozilla/extensions
4.0K    ./.mozilla/plugins
12K     ./.mozilla
15M     .
```

du コマンドは、一覧の最後で常に現在のディレクトリーの合計量を表示します。この情報のみを表示するには、**-s** コマンドラインオプションを使用します。

```
du -sh
```

例:

```
~]$ du -sh
15M     .
```

利用可能なコマンドラインオプションの一覧は、**du**(1) の man ページを参照してください。

16.4.6. システムモニターツールの使用

システムモニターツールの **ファイルシステム** タブは、グラフィカルユーザーインターフェースでファイルシステムおよびディスク領域の使用量を表示します。

システムモニターツールを起動するには、パネルから**アプリケーション** → **システムツール** → **システムモニター** の順に選択するか、シェルプロンプトで**gnome-system-monitor** と入力します。その後、**ファイルシステム** タブをクリックするとファイルシステムの一覧が表示されます。

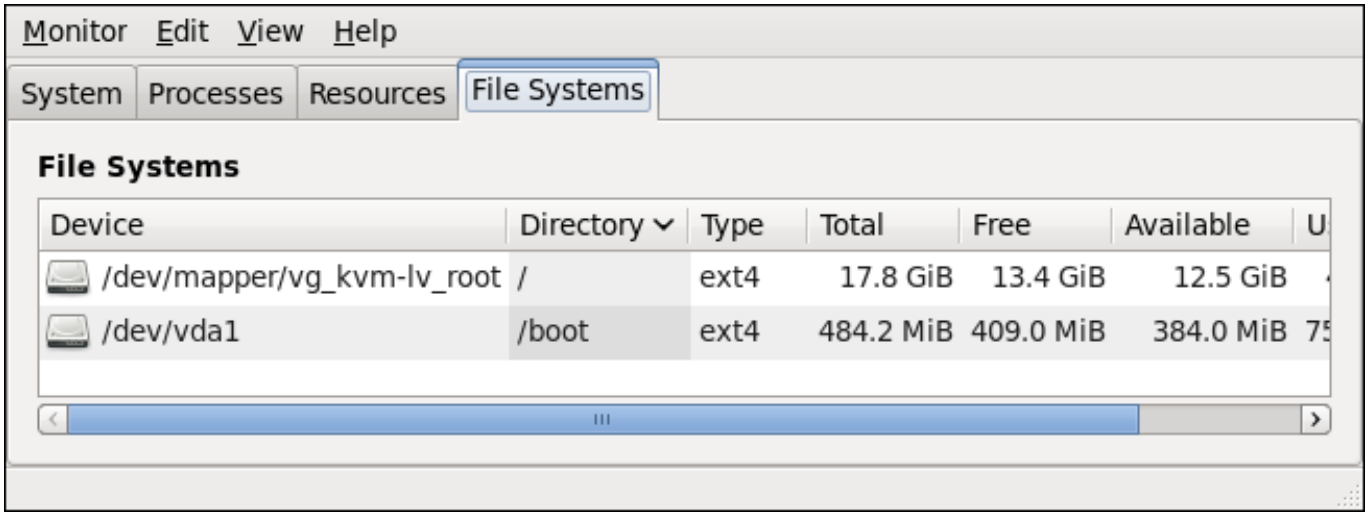


図16.4 システムモニター — ファイルシステム

一覧表示された各ファイルシステムについて **システムモニター**ツールが表示するのは次のとおりです。ソースデバイス (**Device**)、マウントポイント先 (**Directory**)、ファイルシステムのタイプ (**Type**)、さらにはそのサイズ (**Total**)、空き領域 (**Free**)、利用可能な領域 (**Available**)、使用領域 (**Used**) です。

16.5. ハードウェア情報の表示

16.5.1. lspci コマンドの使用

lspci コマンドは、PCI バスおよびそれらに接続されているデバイスの情報を表示します。システム内にあるすべての PCI デバイスを一覧表示するには、シェルプロンプトで以下を入力してください。

```
lspci
```

これは、以下のようにデバイスのシンプルな一覧を表示します。

```
~]$ lspci
00:00.0 Host bridge: Intel Corporation 82X38/X48 Express DRAM Controller
00:01.0 PCI bridge: Intel Corporation 82X38/X48 Express Host-Primary PCI
Express Bridge
00:1a.0 USB Controller: Intel Corporation 82801I (ICH9 Family) USB UHCI
Controller #4 (rev 02)
00:1a.1 USB Controller: Intel Corporation 82801I (ICH9 Family) USB UHCI
Controller #5 (rev 02)
00:1a.2 USB Controller: Intel Corporation 82801I (ICH9 Family) USB UHCI
Controller #6 (rev 02)[出力は省略されています]
```

-v コマンドラインオプションを使用して、詳しい出力を表示することもできます。さらに詳細な出力を表示したい場合は、**-vv** を使用してください。

```
lspci -v|-vv
```

例えば、システムのビデオカードの製造元やモデル、メモリサイズを確認するには、以下を入力します。

```
~]$ lspci -v[出力は省略されています]

01:00.0 VGA compatible controller: nVidia Corporation G84 [Quadro FX 370]
(rev a1) (prog-if 00 [VGA controller])
        Subsystem: nVidia Corporation Device 0491
        Physical Slot: 2
        Flags: bus master, fast devsel, latency 0, IRQ 16
        Memory at f2000000 (32-bit, non-prefetchable) [size=16M]
        Memory at e0000000 (64-bit, prefetchable) [size=256M]
        Memory at f0000000 (64-bit, non-prefetchable) [size=32M]
        I/O ports at 1100 [size=128]
        Expansion ROM at <unassigned> [disabled]
        Capabilities: <access denied>
        Kernel driver in use: nouveau
        Kernel modules: nouveau, nvidiafb
[出力は省略されています]
```

利用可能なコマンドラインオプションの一覧は、**lspci**(8) の man ページを参照してください。

16.5.2. lsusb コマンドの使用

lsusb コマンドは、USB バスおよびそれらに接続されているデバイスの情報を表示します。システム内にあるすべての USB デバイスを一覧表示するには、シェルプロンプトで以下を入力してください。

```
lsusb
```

これは、以下のようにデバイスのシンプルな一覧を表示します。

```
~]$ lsusb
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub[出力は省略されています]
Bus 001 Device 002: ID 0bda:0151 Realtek Semiconductor Corp. Mass Storage Device (Multicard Reader)
Bus 008 Device 002: ID 03f0:2c24 Hewlett-Packard Logitech M-UAL-96 Mouse
Bus 008 Device 003: ID 04b3:3025 IBM Corp.
```

-v コマンドラインオプションを使用すると、さらに詳細な出力を表示することもできます。

```
lsusb -v
```

例を示します。

```
~]$ lsusb -v[出力は省略されています]

Bus 008 Device 002: ID 03f0:2c24 Hewlett-Packard Logitech M-UAL-96 Mouse
Device Descriptor:
  bLength                18
  bDescriptorType         1
  bcdUSB               2.00
  bDeviceClass            0 (Defined at Interface level)
  bDeviceSubClass         0
  bDeviceProtocol         0
  bMaxPacketSize0         8
  idVendor           0x03f0 Hewlett-Packard
  idProduct          0x2c24 Logitech M-UAL-96 Mouse
  bcdDevice           31.00
  iManufacturer           1
  iProduct                2
  iSerial                 0
  bNumConfigurations      1
  Configuration Descriptor:
    bLength                 9
    bDescriptorType         2[出力は省略されています]
```

利用可能なコマンドラインオプションの一覧は、**lsusb**(8) の man ページを参照してください。

16.5.3. lspcmcia コマンドの使用

lspcmcia コマンドは、システム内にあるすべての PCMCIA デバイスを一覧表示します。シェルプロンプトで以下を入力してください。

```
lspcmcia
```

例:

```
~]$ lspcmcia
Socket 0 Bridge:        [yenta_cardbus]         (bus ID: 0000:15:00.0)
```

-v コマンドラインオプションを使用して、詳しい出力を表示することもできます。さらに詳細な出力を表示したい場合は、**-vv** を使用してください。

```
lspcmcia -v|-vv
```

例を示します。

```
~]$ lspcmcia -v
Socket 0 Bridge:        [yenta_cardbus]         (bus ID: 0000:15:00.0)
        Configuration:  state: on       ready: unknown
```

利用可能なコマンドラインオプションの一覧は、**pccardctl**(8) の man ページを参照してください。

16.5.4. lscpu コマンドの使用

lscpu コマンドは、システム内にある CPU に関する情報を一覧表示します。例えば、CPU 数、アーキテクチャー、ベンダー、ファミリ、モデル、CPU キャッシュなどです。シェルプロンプトで以下を入力してください。

```
lscpu
```

例:

```
~]$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                4
On-line CPU(s) list:   0-3
Thread(s) per core:    1
Core(s) per socket:    4
Socket(s):             1
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 23
Stepping:              7
CPU MHz:               1998.000
BogoMIPS:              4999.98
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              3072K
NUMA node0 CPU(s):     0-3
```

利用可能なコマンドラインオプションの一覧は、**lscpu**(1) の man ページを参照してください。

16.6. Net-SNMP を使用したパフォーマンスのモニタリング

Red Hat Enterprise Linux 7 には、柔軟かつ拡張可能な SNMP (*Simple Network Management Protocol:* シン

プルネットワークマネージメントプロトコル) エージェントを含む **Net-SNMP** ソフトウェアスイートが含まれています。このエージェントと関連ユーティリティーを使用すると、多くのシステムからのパフォーマンスデータを SNMP プロトコルによるポーリングに対応する各種ツールに提供することができます。

このセクションでは、ネットワーク上でパフォーマンスデータを安全に提供するための Net-SNMP エージェントの設定方法、SNMP プロトコルを使用したデータの取得方法、カスタムのパフォーマンスメトリックを提供するための SNMP エージェントの拡張方法について説明します。

16.6.1. Net-SNMP のインストール

Net-SNMP ソフトウェアスイートは、Red Hat Enterprise Linux ソフトウェアディストリビューションの RPM パッケージのセットとして利用可能です。表16.2「利用可能な Net-SNMP パッケージ」は、各パッケージと内容を要約したものです。

表16.2 利用可能な Net-SNMP パッケージ

パッケージ	提供する項目
net-snmp	SNMP Agent Daemon とドキュメント。このパッケージは、パフォーマンスデータをエクスポートするために必要です。
net-snmp-libs	**netsnmp** ライブラリと同梱の管理情報ベース (MIB)。このパッケージは、パフォーマンスデータをエクスポートするために必要です。
net-snmp-utils	**snmpget** や **snmpwalk** などの SNMP クライアント。このパッケージは、SNMP によりシステムのパフォーマンスデータをクエリするために必要です。

これらのパッケージをインストールするには、以下の形式で **yum** コマンドを使用します:

```
yum install package…
```

例えば、本セクションで使用される SNMP Agent Daemon および SNMP クライアントをインストールするには、シェルプロンプトで以下を入力します。

```
~]# yum install net-snmp net-snmp-libs net-snmp-utils
```

このコマンドを実行するには、スーパーユーザー権限を持っている (すなわち **root** としてログインしている) 必要があります。Red Hat Enterprise Linux に新しいパッケージをインストールする方法の詳細については「パッケージのインストール」を参照してください。

16.6.2. Net-SNMP Daemon の実行

net-snmp パッケージには、SNMP Agent Daemon である **snmpd** が含まれています。本セクションでは、**snmpd** サービスを起動、停止、再起動する方法、また特定のランレベルでこれを有効にする方法について説明します。ランレベルの概念や Red Hat Enterprise Linux における一般的なシステムサービスの管理方法の詳細については、*6章systemd によるサービス管理*を参照してください。

16.6.2.1. サービスの開始

現行のセッションで **snmpd** サービスを実行するには、シェルプロンプトで**root** として以下を入力します。

```
systemctl start snmpd.service
```

ブート時にサービスが自動的に起動するよう設定するには、以下のコマンドを使用します。

```
systemctl enable snmpd.service
```

16.6.2.2. サービスの停止

実行中の **snmpd** サービスを停止するには、シェルプロンプトで**root** として以下を入力します。

```
systemctl stop snmpd.service
```

ブート時のサービスの起動を無効にするには、以下のコマンドを使用します。

```
systemctl disable snmpd.service
```

16.6.2.3. サービスの再起動

実行中の **snmpd** サービスを再起動するには、シェルプロンプトで以下を入力します。

```
systemctl restart snmpd.service
```

このコマンドで、サービスの停止と再起動が連続して行われます。サービスを停止せずに設定の再読み込みだけを行いたい場合は、代わりに以下のコマンドを実行します。

```
systemctl reload snmpd.service
```

これにより、実行中の **snmpd** サービスが設定を再読み込みします。

16.6.3. Net-SNMP の設定

Net-SNMP Agent Daemon の設定を変更するには、**/etc/snmp/snmpd.conf** 設定ファイルを編集します。Red Hat Enterprise Linux 7 に備わっているデフォルトの **snmpd.conf** ファイルには、多くのコメントが含まれているため、エージェント設定の際の適切なスタート地点となります。

本セクションでは、システム情報と認証の設定という 2 つの一般的なタスクにフォーカスしています。利用可能な設定指示文の詳細については、**snmpd.conf**(5) の man ページを参照してください。また、*net-snmp* パッケージには **snmpconf** と呼ばれるユーティリティーがあり、これを使って有効なエージェント設定を対話形式で作成できます。

本セクションで説明されている **snmpwalk** ユーティリティーを使用するには、*net-snmp-utils* パッケージがインストールされている必要がある点に注意してください。

注記

設定ファイルの変更を反映させるには、**root** として以下のコマンドを実行し、**snmpd** サービスに設定の再読み込みを強制します。

```
systemctl reload snmpd.service
```

16.6.3.1. システム情報の設定

Net-SNMP は、**system** ツリー経由で基本的なシステム情報を提供します。例えば、次の**snmpwalk** コマンドはデフォルトのエージェント設定を持つ **system** ツリーを示しています。

```
~]# snmpwalk -v2c -c public localhost system
SNMPv2-MIB::sysDescr.0 = STRING: Linux localhost.localdomain 2.6.32-
122.el6.x86_64 #1 SMP Wed Mar 9 23:54:34 EST 2011 x86_64
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (99554) 0:16:35.54
SNMPv2-MIB::sysContact.0 = STRING: Root <root@localhost> (configure
/etc/snmp/snmp.local.conf)
SNMPv2-MIB::sysName.0 = STRING: localhost.localdomain
SNMPv2-MIB::sysLocation.0 = STRING: Unknown (edit /etc/snmp/snmpd.conf)
```

デフォルトでは、**sysName** オブジェクトはホスト名に設定されています。**sysLocation** および **sysContact** オブジェクトは、**syslocation** および **syscontact** 指示文の値を変更することで **/etc/snmp/snmpd.conf** ファイルで設定できます。例を示します。

```
syslocation Datacenter, Row 3, Rack 2
syscontact UNIX Admin <admin@example.com>
```

設定ファイルに変更を加えた後は、再度 **snmpwalk** コマンドを実行し、設定を再読み込みしてテストします。

```
~]# systemctl reload snmp.service
~]# snmpwalk -v2c -c public localhost system
SNMPv2-MIB::sysDescr.0 = STRING: Linux localhost.localdomain 2.6.32-
122.el6.x86_64 #1 SMP Wed Mar 9 23:54:34 EST 2011 x86_64
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (158357) 0:26:23.57
SNMPv2-MIB::sysContact.0 = STRING: UNIX Admin <admin@example.com>
SNMPv2-MIB::sysName.0 = STRING: localhost.localdomain
SNMPv2-MIB::sysLocation.0 = STRING: Datacenter, Row 3, Rack 2
```

16.6.3.2. 認証の設定

Net-SNMP Agent Daemon は SNMP プロトコルの 3 つの全バージョンに対応します。最初の 2 つのバージョン (1 と 2c) は、*コミュニティ文字列* を使用した簡易認証を提供します。この文字列は、エージェントとクライアントユーティリティー間で共有される秘密です。この文字列は、ネットワーク上でクリアテキストで渡されますが、安全であるとはみなされません。SNMP プロトコルのバージョン 3 は、各種プロトコルを使用したユーザー認証とメッセージの暗号化に対応しています。また Net-SNMP エージェントは、SSH によるトンネリング、X.509 証明書を用いた TLS 認証、Kerberos 認証にも対応しています。

SNMP Version 2c Community の設定

SNMP version 2c community を設定するには、**/etc/snmp/snmpd.conf** 設定ファイルで **rocommunity** または **rwcommunity** 指示文を使用します。指示文の形式は、以下のとおりです。

```
directive community [source [OID]]
```

ここでの *community* は使用するコミュニティ文字列、*source* は IP アドレスまたはサブネット、*OID* はアクセスを提供する SNMP ツリーです。例えば、次の指示文は、ローカルマシン上でコミュニティ文字列「redhat」 を使用するクライアントに**system** ツリーへの読み取り専用のアクセスを与えます。

```
rocommunity redhat 127.0.0.1 .1.3.6.1.2.1.1
```

設定をテストするには、**-v** と **-c** オプションを付けた **snmpwalk** コマンドを使用します:

```
~]# snmpwalk -v2c -c redhat localhost system
SNMPv2-MIB::sysDescr.0 = STRING: Linux localhost.localdomain 2.6.32-
122.el6.x86_64 #1 SMP Wed Mar 9 23:54:34 EST 2011 x86_64
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (158357) 0:26:23.57
SNMPv2-MIB::sysContact.0 = STRING: UNIX Admin <admin@example.com>
SNMPv2-MIB::sysName.0 = STRING: localhost.localdomain
SNMPv2-MIB::sysLocation.0 = STRING: Datacenter, Row 3, Rack 2
```

SNMP Version 3 User の設定

SNMP version 3 user を設定するには、**net-snmp-create-v3-user** コマンドを使用します。このコマンドは、**/var/lib/net-snmp/snmpd.conf** および **/etc/snmp/snmpd.conf** ファイルへエントリを追加し、ユーザーを作成してそのユーザーにアクセスを付与します。**net-snmp-create-v3-user** コマンドは、エージェントが実行中でない時にのみ実行可能な点に注意してください。以下の例では「redhatsnmp」というパスワードを持つ「admin」ユーザーを作成します。

```
~]# systemctl stop snmpd.service
~]# net-snmp-create-v3-user
Enter a SNMPv3 user name to create:
admin
Enter authentication pass-phrase:
redhatsnmp
Enter encryption pass-phrase:
  [press return to reuse the authentication pass-phrase]

adding the following line to /var/lib/net-snmp/snmpd.conf:
   createUser admin MD5 "redhatsnmp" DES
adding the following line to /etc/snmp/snmpd.conf:
   rwuser admin
~]# systemctl start snmpd.service
```

net-snmp-create-v3-user が **/etc/snmp/snmpd.conf** に追加する **rwuser** 指示文 (または **-ro** のコマンドラインオプションが提供されている場合は **rouser**) の形式は、**rwcommunity** および **rocommunity** 指示文と似ています。

```
directive user [noauth|auth|priv] [OID]
```

ここでの *user* はユーザー名で、*OID* はアクセスを提供する SNMP ツリーです。デフォルトでは、Net-SNMP Agent Daemon は認証済み要求のみを許可します (**auth** オプション)。**noauth** オプションを使うと、認証されていない要求を許可することができ、**priv** オプションは暗号化の使用を強制します。**authpriv** オプションを使用すると、要求の認証、応答の暗号化が必要であると指定します。

例えば、以下の行では、ユーザー「admin」にツリー全体への読み取りと書き込みのアクセスを付与します。

```
rwuser admin authpriv .1
```

設定をテストするには、使用しているユーザーのホームディレクトリー内に **.snmp** ディレクトリーを作成して、そのディレクトリーに次の行を含む **snmp.conf** と呼ばれる設定ファイルを作成します (**~/.snmp/snmp.conf**)。

```
defVersion 3
defSecurityLevel authPriv
defSecurityName admin
defPassphrase redhatsnmp
```

これで、エージェントをクエリする場合、**snmpwalk** コマンドはこれらの認証設定を使用するようになります。

```
~]$ snmpwalk -v3 localhost system
SNMPv2-MIB::sysDescr.0 = STRING: Linux localhost.localdomain 2.6.32-
122.el6.x86_64 #1 SMP Wed Mar 9 23:54:34 EST 2011 x86_64[出力は省略されています]
```

16.6.4. SNMP によるパフォーマンスデータの取得

Red Hat Enterprise Linux の Net-SNMP Agent は、SNMP プロトコルによりパフォーマンスの各種情報を提供します。さらにエージェントは、システム上のインストールされた RPM パッケージの一覧、システム上で現在実行中のプロセス一覧、またはシステムのネットワーク設定をクエリすることもできます。

本セクションでは、SNMP による利用可能なパフォーマンスチューニングに関連する OID の概要について説明します。ここでは、*net-snmp-utils* パッケージがインストールされ、「認証の設定」で説明のとおりユーザーは SNMP ツリーへのアクセスが許可されていることを前提としています。

16.6.4.1. ハードウェアの設定

Net-SNMP に含まれている **Host Resources MIB** は、ホストのハードウェアおよびソフトウェアの現行設定に関する情報をクライアントユーティリティーに表示します。表16.3「利用可能な OID」には、その MIB で利用可能な様々な OID を要約した内容が記載されています。

表16.3 利用可能な OID

OID	詳細
HOST-RESOURCES-MIB::hrSystem	アップタイム、ユーザー数、実行中のプロセス数などのシステム情報全般が含まれています。
HOST-RESOURCES-MIB::hrStorage	メモリおよびファイルシステムの使用に関するデータが含まれています。
HOST-RESOURCES-MIB::hrDevices	すべてのプロセッサー、ネットワークデバイス、ファイルシステムの一覧が含まれています。
HOST-RESOURCES-MIB::hrSWRun	実行中の全プロセス一覧が含まれています。
HOST-RESOURCES-MIB::hrSWRunPerf	HOST-RESOURCES-MIB::hrSWRun からのプロセステーブル上のメモリと CPU 統計が含まれています。
HOST-RESOURCES-MIB::hrSWInstalled	RPM データベースの一覧が含まれています。

入手可能な情報の概要を取得するために使用できる Host Resources MIB には、多くの SNMP テーブルがあります。次の例は、**HOST-RESOURCES-MIB::hrFSTable** を表示しています。

```
~]$ snmptable -Cb localhost HOST-RESOURCES-MIB::hrFSTable
SNMP table: HOST-RESOURCES-MIB::hrFSTable

 Index MountPoint RemoteMountPoint                                  Type
    Access Bootable StorageIndex LastFullBackupDate LastPartialBackupDate
     1        "/"                "" HOST-RESOURCES-TYPES::hrFSLinuxExt2
```

```
readWrite     true           31      0-1-1,0:0:0.0         0-1-1,0:0:0.0
    5 "/dev/shm"               ""     HOST-RESOURCES-TYPES::hrFSOther
readWrite    false           35      0-1-1,0:0:0.0         0-1-1,0:0:0.0
    6    "/boot"               "" HOST-RESOURCES-TYPES::hrFSLinuxExt2
readWrite    false           36      0-1-1,0:0:0.0         0-1-1,0:0:0.0
```

HOST-RESOURCES-MIB の詳細については、**/usr/share/snmp/mibs/HOST-RESOURCES-MIB.txt** ファイルを参照してください。

16.6.4.2. CPU とメモリ情報

大半のシステムのパフォーマンスデータは **UCD SNMP MIB** にあります。**systemStats** OID は、プロセッサー使用率に関する多くのカウンターを提供します。

```
~]$ snmpwalk localhost UCD-SNMP-MIB::systemStats
UCD-SNMP-MIB::ssIndex.0 = INTEGER: 1
UCD-SNMP-MIB::ssErrorName.0 = STRING: systemStats
UCD-SNMP-MIB::ssSwapIn.0 = INTEGER: 0 kB
UCD-SNMP-MIB::ssSwapOut.0 = INTEGER: 0 kB
UCD-SNMP-MIB::ssIOSent.0 = INTEGER: 0 blocks/s
UCD-SNMP-MIB::ssIOReceive.0 = INTEGER: 0 blocks/s
UCD-SNMP-MIB::ssSysInterrupts.0 = INTEGER: 29 interrupts/s
UCD-SNMP-MIB::ssSysContext.0 = INTEGER: 18 switches/s
UCD-SNMP-MIB::ssCpuUser.0 = INTEGER: 0
UCD-SNMP-MIB::ssCpuSystem.0 = INTEGER: 0
UCD-SNMP-MIB::ssCpuIdle.0 = INTEGER: 99
UCD-SNMP-MIB::ssCpuRawUser.0 = Counter32: 2278
UCD-SNMP-MIB::ssCpuRawNice.0 = Counter32: 1395
UCD-SNMP-MIB::ssCpuRawSystem.0 = Counter32: 6826
UCD-SNMP-MIB::ssCpuRawIdle.0 = Counter32: 3383736
UCD-SNMP-MIB::ssCpuRawWait.0 = Counter32: 7629
UCD-SNMP-MIB::ssCpuRawKernel.0 = Counter32: 0
UCD-SNMP-MIB::ssCpuRawInterrupt.0 = Counter32: 434
UCD-SNMP-MIB::ssIORawSent.0 = Counter32: 266770
UCD-SNMP-MIB::ssIORawReceived.0 = Counter32: 427302
UCD-SNMP-MIB::ssRawInterrupts.0 = Counter32: 743442
UCD-SNMP-MIB::ssRawContexts.0 = Counter32: 718557
UCD-SNMP-MIB::ssCpuRawSoftIRQ.0 = Counter32: 128
UCD-SNMP-MIB::ssRawSwapIn.0 = Counter32: 0
UCD-SNMP-MIB::ssRawSwapOut.0 = Counter32: 0
```

特に、**ssCpuRawUser**、**ssCpuRawSystem**、**ssCpuRawWait** および **ssCpuRawIdle** の OID は、システムがカーネル領域、ユーザー領域または I/O でプロセッサー時間の大半を費やしているかどうかを判断する際に役立つカウンターを提供します。**ssRawSwapIn** と **ssRawSwapOut** は、システムがメモリ消費不足かどうかを知る際にも有用な場合もあります。

さらなるメモリ情報は、**free** コマンドと類似するデータを提供する**UCD-SNMP-MIB::memory** OID で入手できます。

```
~]$ snmpwalk localhost UCD-SNMP-MIB::memory
UCD-SNMP-MIB::memIndex.0 = INTEGER: 0
UCD-SNMP-MIB::memErrorName.0 = STRING: swap
UCD-SNMP-MIB::memTotalSwap.0 = INTEGER: 1023992 kB
UCD-SNMP-MIB::memAvailSwap.0 = INTEGER: 1023992 kB
UCD-SNMP-MIB::memTotalReal.0 = INTEGER: 1021588 kB
```

```
UCD-SNMP-MIB::memAvailReal.0 = INTEGER: 634260 kB
UCD-SNMP-MIB::memTotalFree.0 = INTEGER: 1658252 kB
UCD-SNMP-MIB::memMinimumSwap.0 = INTEGER: 16000 kB
UCD-SNMP-MIB::memBuffer.0 = INTEGER: 30760 kB
UCD-SNMP-MIB::memCached.0 = INTEGER: 216200 kB
UCD-SNMP-MIB::memSwapError.0 = INTEGER: noError(0)
UCD-SNMP-MIB::memSwapErrorMsg.0 = STRING:
```

負荷平均は、**UCD SNMP MIB** でも入手可能です。SNMP テーブル**UCD-SNMP-MIB::laTable** には、1 分、5 分、15 分の負荷平均を示した一覧があります。

```
~]$ snmptable localhost UCD-SNMP-MIB::laTable
SNMP table: UCD-SNMP-MIB::laTable

 laIndex laNames laLoad laConfig laLoadInt laLoadFloat laErrorFlag
laErrMessage
       1  Load-1   0.00    12.00         0    0.000000     noError
       2  Load-5   0.00    12.00         0    0.000000     noError
       3 Load-15   0.00    12.00         0    0.000000     noError
```

16.6.4.3. ファイルシステムとディスク情報

Host Resources MIB は、ファイルシステムのサイズと使用量についての情報を表示します。**HOST-RESOURCES-MIB::hrStorageTable** テーブルには、各ファイルシステム (および各メモリプール) のエントリーがあります。

```
~]$ snmptable -Cb localhost HOST-RESOURCES-MIB::hrStorageTable
SNMP table: HOST-RESOURCES-MIB::hrStorageTable

 Index                                          Type           Descr
AllocationUnits    Size   Used AllocationFailures
     1           HOST-RESOURCES-TYPES::hrStorageRam Physical memory
1024 Bytes 1021588 388064                 ?
     3 HOST-RESOURCES-TYPES::hrStorageVirtualMemory  Virtual memory
1024 Bytes 2045580 388064                 ?
     6         HOST-RESOURCES-TYPES::hrStorageOther  Memory buffers
1024 Bytes 1021588  31048                 ?
     7         HOST-RESOURCES-TYPES::hrStorageOther   Cached memory
1024 Bytes  216604 216604                 ?
    10 HOST-RESOURCES-TYPES::hrStorageVirtualMemory      Swap space
1024 Bytes 1023992      0                 ?
    31     HOST-RESOURCES-TYPES::hrStorageFixedDisk              /
4096 Bytes 2277614 250391                 ?
    35     HOST-RESOURCES-TYPES::hrStorageFixedDisk        /dev/shm
4096 Bytes  127698      0                 ?
    36     HOST-RESOURCES-TYPES::hrStorageFixedDisk           /boot
1024 Bytes  198337  26694                 ?
```

HOST-RESOURCES-MIB::hrStorageSize および **HOST-RESOURCES-MIB::hrStorageUsed** の OID を使用すると、それぞれマウントされたファイルシステムの残存容量を算出することができます。

I/O データは **UCD-SNMP-MIB::systemStats** (**ssIORawSent.0** と **ssIORawRecieved.0**) および **UCD-DISKIO-MIB::diskIOTable** にあります。後者は、前者と比べてより詳細なデータを提供します。このテーブルには、**diskIONReadX** および **diskIONWrittenX** の OID があり、システムブートから問題のブロックデバイスに対し読み取りおよび書き込みを実行したバイト数のカウンターを提供します。

```
~]$ snmptable -Cb localhost UCD-DISKIO-MIB::diskIOTable
SNMP table: UCD-DISKIO-MIB::diskIOTable

 Index Device     NRead  NWritten Reads Writes LA1 LA5 LA15    NReadX
NWrittenX
...
    25    sda 216886272 139109376 16409   4894   ?   ?    ? 216886272
139109376
    26   sda1   2455552      5120   613      2   ?   ?    ?   2455552
5120
    27   sda2   1486848         0   332      0   ?   ?    ?   1486848
0
    28   sda3 212321280 139104256 15312   4871   ?   ?    ? 212321280
139104256
```

16.6.4.4. ネットワーク情報

Interfaces MIB はネットワークデバイスの情報を提供します。**IF-MIB::ifTable** は、SNMP テーブルにシステム上の各インターフェースのエントリー、インターフェースの設定、インターフェース用の各種パケットカウンターを提供します。以下の例は、2 つの物理ネットワークインターフェースを持つシステム上の **ifTable** の最初の数コラムを示しています。

```
~]$ snmptable -Cb localhost IF-MIB::ifTable
SNMP table: IF-MIB::ifTable

 Index Descr             Type    Mtu    Speed      PhysAddress AdminStatus
     1    lo softwareLoopback 16436 10000000                          up
     2  eth0   ethernetCsmacd  1500        0 52:54:0:c7:69:58
up
     3  eth1   ethernetCsmacd  1500        0 52:54:0:a7:a3:24        down
```

ネットワークトラフィックは、**IF-MIB::ifOutOctets** および **IF-MIB::ifInOctets** の OID にあります。以下の SNMP クエリは、このシステム上の各インターフェースに対するネットワークトラフィックを取得します。

```
~]$ snmpwalk localhost IF-MIB::ifDescr
IF-MIB::ifDescr.1 = STRING: lo
IF-MIB::ifDescr.2 = STRING: eth0
IF-MIB::ifDescr.3 = STRING: eth1
~]$ snmpwalk localhost IF-MIB::ifOutOctets
IF-MIB::ifOutOctets.1 = Counter32: 10060699
IF-MIB::ifOutOctets.2 = Counter32: 650
IF-MIB::ifOutOctets.3 = Counter32: 0
~]$ snmpwalk localhost IF-MIB::ifInOctets
IF-MIB::ifInOctets.1 = Counter32: 10060699
IF-MIB::ifInOctets.2 = Counter32: 78650
IF-MIB::ifInOctets.3 = Counter32: 0
```

16.6.5. Net-SNMP の拡張

Net-SNMP Agent は、raw システムメトリックに加えてアプリケーションメトリックを提供するために拡張することができます。これにより、パフォーマンス問題のトラブルシューティングだけでなく容量計画も行うことができます。例えば、試験中に電子メールシステムの 5 分の負荷平均が 15 であったことを把握しておくことは役に立つかもしれませんが、毎秒 80,000 メッセージの処理中に電子メールシステムの負荷平

均が 15 であることを知っておく方がはるかに役立ちます。アプリケーションメトリックがシステムメトリックと同じインターフェースで使用可能な場合、システムパフォーマンスの様々な負荷状況の影響も視覚化することができます (例えば 10,000 メッセージが追加されると、負荷平均は 100,000 まで直線的に増加します)。

Red Hat Enterprise Linux に含まれているアプリケーションの多くは、Net-SNMP Agent を拡張して、SNMP のアプリケーション指標を提供します。カスタムアプリケーション用にエージェントを拡張する方法もいくつかあります。本セクションでは、シェルスクリプトを使ったエージェントの拡張方法について説明します。本セクションでは、*net-snmp-utils* パッケージがインストールされ、「認証の設定」 で説明のとおりユーザーが SNMP ツリーへのアクセスを許可されていることを前提としています。

16.6.5.1. シェルスクリプトによる Net-SNMP の拡張

Net-SNMP Agent は、任意のシェルスクリプトをクエリするために使用可能な拡張 MIB (**NET-SNMP-EXTEND-MIB**) を提供します。実行するシェルスクリプトを指定するには、**/etc/snmp/snmpd.conf** ファイルの **extend** 指示文を使用します。定義されると、Agent は SNMP により終了コードとコマンドの出力を提供します。以下の例では、プロセステーブルの **httpd** プロセスの数を決定するスクリプトを使ってこの仕組みを説明しています。

注記

Net-SNMP Agent には、**proc** 指示文によりプロセステーブルを確認する組み込みメカニズムも備わっています。詳細は **snmpd.conf**(5) の man ページを参照してください。

以下のシェルスクリプトの終了コードは、任意の時点におけるシステム上での実行中の **httpd** プロセス数です。

```
#!/bin/sh

NUMPIDS=`pgrep httpd | wc -l`

exit $NUMPIDS
```

SNMP でこのスクリプトを利用可能にするには、システムパス上にある場所にスクリプトをコピー後、実行ビットを設定して、**extend** 指示文を **/etc/snmp/snmpd.conf** ファイルに追加します。**extend** 指示文の形式は、以下のとおりです。

```
extend name prog args
```

ここでの *name* は拡張するための識別文字列、*prog* は実行するプログラム、*args* はプログラムに渡す引数です。たとえば、上記のシェルスクリプトが **/usr/local/bin/check_apache.sh** にコピーされた場合、以下の指示文は SNMP ツリーにスクリプトを追加します。

```
extend httpd_pids /bin/sh /usr/local/bin/check_apache.sh
```

スクリプトは **NET-SNMP-EXTEND-MIB::nsExtendObjects** でクエリできます:

```
~]$ snmpwalk localhost NET-SNMP-EXTEND-MIB::nsExtendObjects
NET-SNMP-EXTEND-MIB::nsExtendNumEntries.0 = INTEGER: 1
NET-SNMP-EXTEND-MIB::nsExtendCommand."httpd_pids" = STRING: /bin/sh
NET-SNMP-EXTEND-MIB::nsExtendArgs."httpd_pids" = STRING:
/usr/local/bin/check_apache.sh
NET-SNMP-EXTEND-MIB::nsExtendInput."httpd_pids" = STRING:
```

```
NET-SNMP-EXTEND-MIB::nsExtendCacheTime."httpd_pids" = INTEGER: 5
NET-SNMP-EXTEND-MIB::nsExtendExecType."httpd_pids" = INTEGER: exec(1)
NET-SNMP-EXTEND-MIB::nsExtendRunType."httpd_pids" = INTEGER: run-on-
read(1)
NET-SNMP-EXTEND-MIB::nsExtendStorage."httpd_pids" = INTEGER:
permanent(4)
NET-SNMP-EXTEND-MIB::nsExtendStatus."httpd_pids" = INTEGER: active(1)
NET-SNMP-EXTEND-MIB::nsExtendOutput1Line."httpd_pids" = STRING:
NET-SNMP-EXTEND-MIB::nsExtendOutputFull."httpd_pids" = STRING:
NET-SNMP-EXTEND-MIB::nsExtendOutNumLines."httpd_pids" = INTEGER: 1
NET-SNMP-EXTEND-MIB::nsExtendResult."httpd_pids" = INTEGER: 8
NET-SNMP-EXTEND-MIB::nsExtendOutLine."httpd_pids".1 = STRING:
```

注意する点は、終了コード (この例では 「8」) は INTEGER (整数) タイプとして、出力は STRING (文字列) タイプとして、示されていることです。複数のメトリックを整数として表示するには、**extend** 指示文を使用してスクリプトに異なる引数を指定します。例えば、以下のシェルスクリプトを使用すると、任意の文字列に一致するプロセスの数を見つけ出すことができ、プロセスの数を示すテキスト文字列も出力します。

```
#!/bin/sh

PATTERN=$1
NUMPIDS=`pgrep $PATTERN | wc -l`

echo "There are $NUMPIDS $PATTERN processes."
exit $NUMPIDS
```

次の **/etc/snmp/snmpd.conf** 指示文は、上記のスクリプトが **/usr/local/bin/check_proc.sh** にコピーされる時に **httpd** PID の数と併せて **snmpd** PID の数を与えます。

```
extend httpd_pids /bin/sh /usr/local/bin/check_proc.sh httpd
extend snmpd_pids /bin/sh /usr/local/bin/check_proc.sh snmpd
```

以下の例は、**nsExtendObjects** OID の **snmpwalk** の出力を示しています。

```
~]$ snmpwalk localhost NET-SNMP-EXTEND-MIB::nsExtendObjects
NET-SNMP-EXTEND-MIB::nsExtendNumEntries.0 = INTEGER: 2
NET-SNMP-EXTEND-MIB::nsExtendCommand."httpd_pids" = STRING: /bin/sh
NET-SNMP-EXTEND-MIB::nsExtendCommand."snmpd_pids" = STRING: /bin/sh
NET-SNMP-EXTEND-MIB::nsExtendArgs."httpd_pids" = STRING:
/usr/local/bin/check_proc.sh httpd
NET-SNMP-EXTEND-MIB::nsExtendArgs."snmpd_pids" = STRING:
/usr/local/bin/check_proc.sh snmpd
NET-SNMP-EXTEND-MIB::nsExtendInput."httpd_pids" = STRING:
NET-SNMP-EXTEND-MIB::nsExtendInput."snmpd_pids" = STRING:
...
NET-SNMP-EXTEND-MIB::nsExtendResult."httpd_pids" = INTEGER: 8
NET-SNMP-EXTEND-MIB::nsExtendResult."snmpd_pids" = INTEGER: 1
NET-SNMP-EXTEND-MIB::nsExtendOutLine."httpd_pids".1 = STRING: There are 8
httpd processes.
NET-SNMP-EXTEND-MIB::nsExtendOutLine."snmpd_pids".1 = STRING: There are 1
snmpd processes.
```

警告

整数の終了コードは 0 から 255 の範囲に制限されています。256 を超える可能性がある値については、スクリプトの標準出力 (文字列として入力されるもの) を使用するか、エージェントを拡張するという別の方法を実行してください。

この最後の例では、システムの空きメモリと **httpd** プロセスの数のクエリを示しています。このクエリは、パフォーマンステスト中にメモリ負担に与えるプロセス数の影響を知るために使用することができます。

```
~]$ snmpget localhost \
    'NET-SNMP-EXTEND-MIB::nsExtendResult."httpd_pids"' \
    UCD-SNMP-MIB::memAvailReal.0
NET-SNMP-EXTEND-MIB::nsExtendResult."httpd_pids" = INTEGER: 8
UCD-SNMP-MIB::memAvailReal.0 = INTEGER: 799664 kB
```

16.7. その他のリソース

システム情報の収集方法についてさらに知るには、以下のリソースを参照してください。

16.7.1. インストールされているドキュメント

- **ps**(1) — **ps** コマンドの man ページです。
- **top**(1) — **top** コマンドの man ページです。
- **free**(1) — **free** コマンドの man ページです。
- **df**(1) — **df** コマンドの man ページです。
- **du**(1) — **du** コマンドの man ページです。
- **lspci**(8) — **lspci** コマンドの man ページです。
- **snmpd**(8) — **snmpd** サービスの man ページです。
- **snmpd.conf**(5) — 利用可能な設定指示文についての全ドキュメントを含む **/etc/snmp/snmpd.conf** ファイルの man ページです。

第17章 OpenLMI

Open Linux Management Infrastructure は、通常 **OpenLMI** と短縮形で呼ばれる Linux システムを管理する一般的なインフラストラクチャーです。これは既存のツール上に構築され、システム管理者から基礎となるシステムの複雑性を隠すために抽出化レイヤーとして機能します。OpenLMI は、ローカルおよびリモートでのアクセスが可能なサービス一式と配布され、ハードウェアやオペレーティングシステム、システムサービスの管理および監視に使用できる複数言語のバインディング、標準 API、および標準スクリプトインターフェイスを提供します。

17.1. OpenLMI の概要

OpenLMI は、物理および仮想マシンの両方で Red Hat Enterprise Linux システムを実行している実稼働サーバーに共通の管理インターフェイスを提供するように設計されています。以下の 3 つのコンポーネントで構成されています。

1. *システム管理エージェント* — このエージェントは管理されるシステム上にインストールされ、標準オブジェクトブローカーに提示されるオブジェクトモデルを実装します。OpenLMI に実装される最初のエージェントにはストレージおよびネットワーク設定が含まれますが、その後の作業がシステム管理の追加要素を処理します。システム管理エージェントは、*Common Information Model プロバイダー* または *CIM プロバイダー* と呼ばれます。

2. *標準オブジェクトブローカー* — オブジェクトブローカーはシステム管理エージェントを管理し、インターフェイスを提供します。標準オブジェクトブローカーは、*CIM オブジェクトモニター* または *CIMOM* とも呼ばれます。

3. *クライアントアプリケーションおよびスクリプト* — クライアントアプリケーションおよびスクリプトは、標準オブジェクトブローカーでシステム管理エージェントを呼び出します。

OpenLMI プロジェクトは、スクリプトまたはシステム管理コンソールで使用可能な低レベルのインターフェイスを提供することで、既存の管理イニシアチブを補完します。OpenLMI と配布されるインターフェイスには、C、C++、Python、Java、およびイニシアチブコマンドラインクライアントが含まれており、これらすべてが各エージェントで実装されている機能に同一の完全なアクセスを提供します。これにより、どのプログラミングインターフェイスを使っていても、まったく同一の機能に常にアクセスできることが保証されています。

17.1.1. 主な特長

以下は、OpenLMI をシステムにインストールして使用する主な利点です。

- OpenLMI は、ローカルおよびリモートのシステムの設定、管理、モニタリングのための標準インターフェイスを提供します。
- 物理および仮想の両方のマシン上の実稼働サーバーの設定、管理、監視ができるようになります。
- CIM プロバイダーの集合とともに配布され、ストレージデバイスおよび複雑なネットワークの設定、管理、監視が可能になります。
- C、C++、Python、および Java プログラムからシステム管理機能を呼び出すことが可能で、コマンドラインインターフェイスも装備されています。
- オープンな業界標準に基づく無料ソフトウェアです。

17.1.2. 管理機能

OpenLMI の主な機能には、ストレージデバイスやネットワーク、システムサービス、ユーザーアカウントの管理、ハードウェアおよびソフトウェアの設定、電源管理、Active Directory との相互作用などがありま

す。Red Hat Enterprise Linux 7 と配布される CIM プロバイダーの完全な一覧については、表17.1「利用可能な CIM プロバイダー」を参照してください。

表17.1 利用可能な CIM プロバイダー

パッケージ名	詳細
openlmi-account	ユーザーアカウント管理用の CIM。
openlmi-logicalfile	ファイルおよびディレクトリー読み込み用の CIM 。
openlmi-networking	ネットワーク管理用の CIM プロバイダー。
openlmi-powermanagement	電源管理用の CIM プロバイダー。
openlmi-service	システムサービス管理用の CIM プロバイダー。
openlmi-storage	ストレージ管理用の CIM プロバイダー。
openlmi-fan	コンピューターファン制御用の CIM プロバイダー。
openlmi-hardware	ハードウェア情報取得用の CIM プロバイダー。
openlmi-realmd	realmd 設定用の CIM プロバイダー。
openlmi-software [a]	ソフトウェア管理用の CIM プロバイダー。

[a] Red Hat Enterprise Linux 7 では、OpenLMI Software プロバイダーは テクノロジープレビューとして含まれています。このプロバイダーは完全に機能するものですが、多くのソフトウェアパッケージをリスト化する際にメモリーと時間が過剰に消費されるという既知のパフォーマンススケーリング問題があります。この問題を回避するには、パッケージ検索ができるだけ少ないパッケージを返すように調整します。

17.2. OpenLMI のインストール

OpenLMI は RPM パッケージのコレクションとして配布され、これには CIMOM、個別の CIM プロバイダー、およびクライアントアプリケーションが含まれます。これにより、管理システムとクライアントシステムの区別ができ、必要なコンポーネントのみをインストールできるようになります。

17.2.1. 管理システムへの OpenLMI のインストール

管理システム は、OpenLMI クライアントツールを使って監視および管理する予定のシステムです。管理システムに OpenLMI をインストールするには、以下のステップにしたがいます。

1. シェルプロンプトで **root** として以下を入力し、*tog-pegasus* パッケージをインストールします。

```
yum install tog-pegasus
```

これで OpenPegasus CIMOM とすべての依存関係がシステムにインストールされ、**pegasus** ユーザーのユーザーアカウントが作成されます。

2. 以下のコマンドを **root** で実行して、必要な CIM プロバイダーをインストールします。

```
yum install openlmi-
{storage,networking,service,account,powermanagement}
```

これでストレージ、ネットワーク、アカウント、および電源管理用の CIM プロバイダーがインストールされます。Red Hat Enterprise Linux 7 と配布される CIM プロバイダーのリストについては、表17.1「利用可能な CIM プロバイダー」を参照してください。

3. **/etc/Pegasus/access.conf** 設定ファイルを編集して、OpenPegasus CIMOM への接続が許可されるユーザーのリストをカスタマイズします。デフォルトでは、**pegasus** ユーザーのみがリモートとローカルの両方で CIMOM にアクセスできます。このユーザーアカウントをアクティブ化するには、**root** で以下のコマンドを実行してユーザーのパスワードを設定します。

```
passwd pegasus
```

4. **tog-pegasus.service** ユニットをアクティブ化して OpenPegasus CIMOM を開始します。現行セッションで **tog-pegasus.service** ユニットをアクティブ化するには、**root** でシェルプロンプトに以下を入力します。

```
systemctl start tog-pegasus.service
```

tog-pegasus.service が起動時に自動的に開始するように設定するには、**root** で以下を入力します。

```
systemctl enable tog-pegasus.service
```

5. リモートマシンから管理システムに対話する予定の場合は、TCP 通信をポート **5989** (**wbem-https**) で有効にします。現行セッションでこのポートを開くには、**root** で以下のコマンドを実行します。

```
firewall-cmd --add-port 5989/tcp
```

ポート **5989** を TCP 通信用に永続的に開いておくには、**root** で以下を入力します。

```
firewall-cmd --permanent --add-port 5989/tcp
```

これで「LMIShell の使用」の説明にあるように、管理システムに接続して OpenLMI クライアントツールを使って対話することができます。管理システム上で直接 OpenLMI 操作を実行する場合は、「クライアントシステムへの OpenLMI のインストール」で説明されているステップも完了してください。

17.2.2. クライアントシステムへの OpenLMI のインストール

管理システムと対話する基となるのが、*クライアントシステム*です。通常のシナリオでは、クライアントシステムと管理システムは別の 2 つのマシンにインストールされますが、管理システムにクライアントツールをインストールして、直接対話することも可能です。

クライアントシステムに OpenLMI をインストールするには、以下のステップを実行します。

1. シェルプロンプトで **root** として以下を入力し、*openlmi-tools* パッケージをインストールします。

```
yum install openlmi-tools
```

これで CIM オブジェクトにアクセスするためのインタラクティブクライアントおよびインタプリターである LMIShell と、そのすべての依存関係がシステムにインストールされます。LMIShell は OpenPegasus が提供します。

2. 「OpenPegasus 用に SSL 証明書を設定する」の説明にあるように、OpenPegasus 用の SSL 証明書を設定します。

これで「LMIShell の使用」にあるように、LMIShell クライアントを使用して管理システムと対話することができます。

17.3. OpenPegasus 用に SSL 証明書を設定する

OpenLMI は、HTTP トランスポート層で機能する WBEM (ウェブベースのエンタープライズ管理) を使用します。標準の HTTP ベーシック認証がこのプロトコルで実行されます。つまり、ユーザー名とパスワードがリクエストと共に送信されることになります。

安全な認証を確保するには、OpenPegasus CIMOM を設定して通信に HTTPS を使用することが必要になります。管理システム上で暗号化チャンネルを確立するには、SSL (Secure Sockets Layer) または TLS (Transport Layer Security) 証明書が必要になります。

システム上で SSL/TLS 証明書を管理するには、2 つの方法があります。

- 自己署名証明書では必要なインフラストラクチャーは少なくなりますが、クライアントへの導入および安全な管理はより難しくなります。
- 認証局署名証明書は一旦設定されるとクライアントへの導入は容易ですが、必要な初期投資が大きくなる可能性があります。

認証局署名証明書を使用する場合は、クライアントシステム上で信頼できる認証局を設定する必要があります。その後に、その権限を使ってすべての管理システムの CIMOM 証明書に署名することができるようになります。証明書は証明書チェーンの一部となることもあるので、管理システムの証明書の署名に使用された証明書が別の、より高い認証局 (Verisign、CAcert、RSA などその他多数) によって署名される場合もあります。

ファイルシステム上のデフォルトの証明書と信頼の保存場所を、表17.2「証明書および信頼の保存場所」に一覧表示します。

表17.2 証明書および信頼の保存場所

設定オプション	場所	詳細
sslCertificateFilePath	**/etc/Pegasus/server.pem**	CIMOM の公開証明書。
sslKeyFilePath	**/etc/Pegasus/file.pem**	CIMOM のみが知っている秘密鍵。
sslTrustStore	**/etc/Pegasus/client.pem**	信頼できる証明書機関の一覧を提供するファイルまたはディレクトリー。

重要

表17.2「証明書および信頼の保存場所」にあるファイルを修正した場合は、**tog-pegasus** サービスを再起動して新たな証明書が確実に認識されるようにします。サービスを再起動するには、**root** でシェルプロンプトに以下を入力します。

```
systemctl restart tog-pegasus.service
```

Red Hat Enterprise Linux 7 でシステムサービスを管理する詳細情報については、*6章systemd によるサービス管理* を参照してください。

17.3.1. 自己署名証明書の管理

自己署名証明書は、自身の秘密鍵を使って署名し、他のいかなる信頼チェーンにも接続されません。管理システム上では、**tog-pegasus** の初回起動時前に管理者が証明書を提供していない場合、システムのプライマリーホスト名を証明書の件名に使用して、自己署名証明書のセットが自動的に生成されます。

重要

自動生成された自己署名証明書は、デフォルトで 10 年間有効ですが、自動的に更新する機能はありません。これらの証明書を修正するには、このサブジェクトについて OpenSSL または Mozilla NSS のドキュメンテーションが提供するガイドラインにしたがって、新たな証明書を手動で作成する必要があります。

クライアントシステムが自己署名証明書を信頼するように設定するには、以下のステップにしたがいます。

1. **/etc/Pegasus/client.pem** 証明書を管理システムからクライアントシステムの **/etc/pki/ca-trust/source/anchors/** ディレクトリーにコピーします。これを実行するには、**root** でシェルプロンプトに以下を入力します。

   ```
   scp root@hostname:/etc/Pegasus/client.pem /etc/pki/ca-trust/source/anchors/pegasus-hostname.pem
   ```

   hostname を管理システムのホスト名に置き換えます。このコマンドは、**sshd** サービスが管理システム上で実行中で、**root** ユーザーが SSH プロトコルを使ってシステムにログインできるような設定でのみ機能することに注意してください。**sshd** サービスのインストールおよび設定、また **scp** コマンドを使って SSH プロトコルでファイルを送信する方法については、*7章OpenSSH* を参照してください。

2. チェックサムを元のファイルのものと比較して、クライアントシステム上の証明書の整合性を検証します。管理システム上の **/etc/Pegasus/client.pem** ファイルのチェックサムを計算するには、そのシステム上で **root** として以下のコマンドを実行します。

   ```
   sha1sum /etc/Pegasus/client.pem
   ```

   クライアントシステム上の **/etc/pki/ca-trust/source/anchors/pegasus-*hostname*.pem** ファイルのチェックサムを計算するには、このシステム上で以下のコマンドを実行します。

   ```
   sha1sum /etc/pki/ca-trust/source/anchors/pegasus-hostname.pem
   ```

   hostname を管理システムのホスト名に置き換えます。

3. 信頼ストアを更新するには、**root** で以下のコマンドを実行します。

   ```
   update-ca-trust extract
   ```

17.3.2. Identity Management を使った認証局署名証明書の管理 (推奨)

Red Hat Enterprise Linux の Identity Management 機能は、ドメインにジョインしたシステム内での SSL 証明書管理を簡素化するドメインコントローラーを提供します。Identity Management サーバーは、埋め込み認証局を提供します。クライアントシステムおよび管理システムをドメインにジョインさせる方法については、Red Hat Enterprise Linux 7 Linux Domain Identity, Authentication, and Policy Guide または FreeIPA ドキュメンテーションを参照してください。

管理システムでは、Identity Management への登録が必須になります。クライアントシステムでは、登録はオプションになります。

管理システムでは、以下のステップが必要です。

1. Red Hat Enterprise Linux 7 Linux Domain Identity, Authentication, and Policy Guide にあるように、*ipa-client* パッケージをインストールして、システムを Identity Management に登録します。

2. **root** で以下のコマンドを入力して、Identity Management の署名した証明書を信頼ストアにコピーします。

```
cp /etc/ipa/ca.crt /etc/pki/ca-trust/source/anchors/ipa.crt
```

3. 信頼ストアを更新するには、**root** で以下のコマンドを実行します。

```
update-ca-trust extract
```

4. 権限のあるドメインユーザーとして以下のコマンドを実行して、Pegasus をサービスとして Identity Management ドメインに登録します。

```
ipa service-add CIMOM/hostname
```

hostname を管理システムのホスト名に置き換えます。

ipa-admintools パッケージがインストール済みの Identity Management ドメイン内のシステムからであれば、このコマンドが実行できます。Identity Management にサービスエントリーが作成され、これを署名済み SSL 証明書の生成に使用できるようになります。

5. **/etc/Pegasus/** ディレクトリーにある PEM ファイルのバックアップを作成します(推奨)。

6. **root** で以下のコマンドを実行して、署名済み証明書を取得します。

```
ipa-getcert request -f /etc/Pegasus/server.pem -k
/etc/Pegasus/file.pem -N CN=hostname -K CIMOM/hostname
```

hostname を管理システムのホスト名に置き換えます。

これで証明書と鍵ファイルが正常な場所に保存されます。**ipa-client-install** スクリプトが管理システム上にインストールする **certmonger** デーモンにより、証明書が必要に応じて最新に保たれ、更新されます。

詳細情報は、Red Hat Enterprise Linux 7 Linux Domain Identity, Authentication, and Policy Guide を参照してください。

クライアントシステムを登録して、信頼ストアをアップデートするには、以下のステップにしたがいます。

1. Red Hat Enterprise Linux 7 Linux Domain Identity, Authentication, and Policy Guide にあるように、*ipa-client* パッケージをインストールして、システムを Identity Management に登録します。

2. **root** で以下のコマンドを入力して、Identity Management の署名した証明書を信頼ストアにコピーします。

```
cp /etc/ipa/ca.crt /etc/pki/ca-trust/source/anchors/ipa.crt
```

3. 信頼ストアを更新するには、**root** で以下のコマンドを実行します。

```
update-ca-trust extract
```

クライアントシステムを Identity Management に登録しない場合は、以下のステップにしたがって信頼ストアを更新します。

1. **root**として、同一の Identity Management ドメインにジョインしている他のシステムから安全に **/etc/ipa/ca.crt** ファイルを信頼ストア **/etc/pki/ca-trust/source/anchors/** ディレクトリーにコピーします。
2. 信頼ストアを更新するには、**root** で以下のコマンドを実行します。

```
update-ca-trust extract
```

17.3.3. 認証局署名証明書を手動で管理する

認証局署名証明書を Identity Management 以外のメカニズムで管理するには、手動での設定が必要になります。

管理システムの証明書に署名する認証局の証明書を、すべてのクライアントが信頼するようにする必要があります。

- デフォルトである認証局が信頼されている場合、これを達成するために実行が必要なステップはありません。
- 認証局がデフォルトで信頼されていない場合、クライアントシステムと管理システムで証明書をインポートする必要があります。
  - **root** で以下のコマンドを入力して、証明書を信頼ストアにコピーします。

```
cp /path/to/ca.crt /etc/pki/ca-trust/source/anchors/ca.crt
```

  - 信頼ストアを更新するには、**root** で以下のコマンドを実行します。

```
update-ca-trust extract
```

管理システムで以下のステップを実行します。

1. 新たな SSL 設定ファイル **/etc/Pegasus/ssl.cnf** を作成し、証明書についての情報を保管します。このファイルのコンテンツは、以下の例のようにする必要があります。

```
[ req ]
distinguished_name     = req_distinguished_name
prompt                 = no
[ req_distinguished_name ]
C                      = US
ST                     = Massachusetts
L                      = Westford
O                      = Fedora
OU                     = Fedora OpenLMI
CN                     = hostname
```

   hostname を管理システムの完全修飾ドメイン名に置き換えます。

2. **root** で以下のコマンドを実行して、管理システム上で秘密鍵を生成します。

```
openssl genrsa -out /etc/Pegasus/file.pem 1024
```

3. **root** で以下のコマンドを実行し、証明書署名要求 (CSR) を生成します。

```
openssl req -config /etc/Pegasus/ssl.cnf -new -key
/etc/Pegasus/file.pem -out /etc/Pegasus/server.csr
```

4. **/etc/Pegasus/server.csr** を認証局に送信して署名します。ファイル提出に関する詳細な手順は、個々の認証局によって異なります。

5. 認証局から署名済み証明書を受け取ったら、**/etc/Pegasus/server.pem** として保存します。

6. **root** で以下のコマンドを実行し、信頼できる認証局の証明書を Pegasus 信頼ストアにコピーして、Pegasus が自身の証明書を信頼できるようにします。

```
cp /path/to/ca.crt /etc/Pegasus/client.pem
```

上記のステップをすべて完了したら、署名の認証局を信頼するクライアントが管理サーバーの CIMOM と正常に通信できるようなります。

重要

Identity Management ソリューションとは異なり、証明書の有効期限が切れて更新が必要となった場合には、上記の手動のステップは再度実行する必要があります。証明書は有効期限が切れる前に更新することが推奨されます。

17.4. LMIShell の使用

LMIShell はインタラクティブクライアントかつ非インタラクティブインタープリターで、OpenPegasus CIMOM が提供する CIM オブジェクトにアクセスするために使用できます。Python インタープリターに基づいていますが、CIM オブジェクトとの対話のための追加の関数およびクラスも実装します。

17.4.1. LMIShell の開始、使用、終了

Python インタープリターと同様に、LMIShell はLMIShell スクリプトでインタラクティブクライアントまたは非インタラクティブインタープリターとして使用できます。

インタラクティブモードでの LMIShell の開始

インタラクティブモードで LMIShell インタープリターを開始するには、**lmishell** コマンドを引数なしで実行します。

```
lmishell
```

デフォルトでは、LMIShell は CIMOM との接続確立を試みる際に、サーバー側の証明書を認証局の信頼ストアに対して確認します。この確認を無効にするには、**lmishell** コマンドを **--noverify** または **-n** コマンドラインオプションとともに実行します。

```
lmishell --noverify
```

Tab Completion (タブ入力) の使用

インタラクティブモードでの実行時には、LMIShell インタープリターでは **Tab** キーを使った基本的なプロ

グラミング構造体や CIM オブジェクトの入力が可能になります。これにはネームスペースやクラス、メソッド、オブジェクト属性が含まれます。

履歴閲覧

デフォルトでは、LMIShell はユーザーがインタラクティブプロンプトで入力したすべてのコマンドを **~/.lmishell_history** ファイルに保存します。これにより、コマンド履歴が閲覧でき、インタラクティブモードで入力したコマンドをプロンプトで再度入力することなく再使用することができます。コマンド履歴で後ろに戻るには**上向き矢印** キーか **Ctrl+p** のキーの組み込み合わせを押します。コマンド履歴で前に進むには、**下向き矢印** キーか **Ctrl+n** のキーの組み合わせを押します。

LMIShell は、増分逆検索もサポートしています。コマンド履歴で特定の行を探すには、**Ctrl+r** を押してからコマンドの一部を入力します。例を示します。

```
> (reverse-i-search)`connect': c = connect("server.example.com",
"pegasus")
```

コマンド履歴を削除するには、以下のように **clear_history()** 機能を使用します。

```
clear_history()
```

コマンド履歴に保存される行数は、**~/.lmishellrc** 設定ファイルの **history_length** オプションの値を変更することで設定できます。さらに、同じ設定ファイルの **history_file** オプションの値を変更することで、履歴ファイルの場所を変えることができます。たとえば、履歴ファイルの場所を **~/.lmishell_history** に設定し、最大 **1000** 行を保存するように LMIShell を設定するには、以下の行を **~/.lmishellrc** ファイルに追加します。

```
history_file = "~/.lmishell_history"
history_length = 1000
```

例外処理

デフォルトでは、LMIShell インタープリターはすべての例外を処理し、戻り値を使用します。すべての例外をコードで処理するためにこの動作を無効にするには、以下のように **use_exceptions()** 関数を使用します。

```
use_exceptions()
```

自動例外処理を再度有効にするには、以下を使用します。

```
use_exception(False)
```

例外処理を永続的に無効にするには、**~/.lmishellrc** の **use_exceptions** オプションを **True** に変更します。

```
use_exceptions = True
```

一時的なキャッシュの設定

デフォルト設定では、LMIShell 接続オブジェクトは一時的なキャッシュを使って CIM クラス名および CIM クラスを保存し、ネットワーク通信を減らします。この一時的なキャッシュを削除するには、以下のように **clear_cache()** メソッドを使用します。

```
object_name.clear_cache()
```

object_name を接続オブジェクトの名前に置き換えます。

特定の接続オブジェクトの一時的なキャッシュを無効にするには、以下のように **use_cache()** メソッドを使用します。

```
object_name.use_cache(False)
```

再度これを有効にするには、以下を使用します。

```
object_name.use_cache(True)
```

接続オブジェクトの一時的なキャッシュを永続的に無効にするには、**~/.lmishellrc** の **use_cache** オプションを **False** に変更します。

```
use_cache = False
```

LMIShell の終了

LMIShell インタープリターを終了してシェルプロンプトに戻るには、**Ctrl+d** のキーの組み合わせを押すか、以下のように **quit()** 関数を発行します。

```
> quit()
~]$
```

LMIShell スクリプトの実行

LMIShell スクリプトを実行するには、以下のように **lmishell** コマンドを実行します。

```
lmishell file_name
```

file_name をスクリプト名に置き換えます。この実行の後に LMIShell スクリプトを検査するには、**--interact** または **-i** のコマンドラインオプションも指定します。

```
lmishell --interact file_name
```

LMIShell スクリプトのファイル拡張子は、**.lmi** が適切です。

17.4.2. CIMOM への接続

LMIShell では、同一システム上でローカルで実行している CIMOM または ネットワーク経由でアクセス可能なリモートマシン上の CIMOM に接続することができます。

リモートの CIMOM への接続

リモートの CIMOM が提供する CIM オブジェクトにアクセスするには、以下のように **connect()** 関数を使って接続オブジェクトを作成します。

```
connect(host_name, user_name[, password])
```

host_name を管理システムのホスト名に、*user_name* をそのシステム上で実行中の OpenPegasus CIMOM に接続許可されているユーザー名に、*password* をユーザーのパスワードに置き換えます。パスワードが省略されると、LMIShell パスワードの入力をユーザーに要求します。この関数は **LMIConnection** オブジェクトを返します。

例17.1 リモートの CIMOM への接続

server.example.com 上で実行中の OpenPegasus CIMOM に ユーザー **pegasus** として接続するには、インタラクティブのプロンプトで以下を入力します。

```
> c = connect("server.example.com", "pegasus")
password:
>
```

ローカルの CIMOM への接続

LMIShell では、Unix ソケットを使用することでローカルの CIMOM に接続できます。この種の接続では、LMIShell インタープリターを **root** ユーザーとして実行し、**/var/run/tog-pegasus/cimxml.socket** ソケットが存在する必要があります。

ローカルの CIMOM が提供する CIM オブジェクトにアクセスするには、以下のように **connect()** 関数を使って接続オブジェクトを作成します。

```
connect(host_name)
```

host_name を **localhost**、**127.0.0.1**、または **::1** のいずれかで置き換えます。関数は **LMIConnection** オブジェクトか **None** を返します。

例17.2 ローカルの CIMOM への接続

localhost 上で実行中の OpenPegasus CIMOM に **root** ユーザーとして接続するには、インタラクティブのプロンプトで以下を入力します。

```
> c = connect("localhost")
>
```

CIMOM への接続の確認

connect() 関数は、接続が確立されないと **LMIConnection** オブジェクトか **None** を返します。さらに、**connect()** 関数が接続確立に失敗すると、標準エラー出力にエラーメッセージをプリントします。

CIMOM への接続が正常に確立されたことを確認するには、以下のように **isinstance()** 関数を使用します。

```
isinstance(object_name, LMIConnection)
```

object_name を接続オブジェクト名で置き換えます。この関数は、*object_name* が **LMIConnection** オブジェクトの場合は **True** を、それ以外の場合は **False** を返します。

例17.3 CIMOM への接続の確認

例17.1「リモートの CIMOM への接続」で作成された **c** 変数に **LMIConnection** オブジェクトが含まれていることを確認するには、インタラクティブプロンプトで以下を入力します。

```
> isinstance(c, LMIConnection)
True
>
```

または、**c** が **None** でないことを確認することもできます。

```
> c is None
False
>
```

17.4.3. ネームスペースを使った操作

LMIShell ネームスペースは、利用可能なクラスを組織化する自然な方法を提供し、他のネームスペースおよびクラスへの階層的なアクセスポイントとして機能します。**root** ネームスペースは、接続オブジェクトの最初のエントリーポイントです。

利用可能なネームスペースの一覧表示

利用可能なネームスペースすべてを一覧表示するには、以下のように **print_namespaces()** メソッドを使用します。

```
object_name.print_namespaces()
```

object_name を検査するオブジェクト名で置き換えます。このメソッドは、利用可能なネームスペースを標準出力にプリントします。

利用可能なネームスペース一覧を取得するには、オブジェクト属性 **namespaces** にアクセスします。

```
object_name.namespaces
```

これは文字列の一覧を返します。

例17.4 利用可能なネームスペースの一覧表示

例17.1「リモートの CIMOM への接続」で作成された **c** 接続オブジェクトの **root** ネームスペースオブジェクトを検査するには、インタラクティブプロンプトで以下を入力します。

```
> c.root.print_namespaces()
cimv2
interop
PG_InterOp
PG_Internal
>
```

これらネームスペースの一覧を **root_namespaces** という名前の変数に割り当てるには、以下を入力します。

```
> root_namespaces = c.root.namespaces
>
```

ネームスペースオブジェクトへのアクセス

特定のネームスペースオブジェクトにアクセスするには、以下の構文を使用します。

```
object_name.namespace_name
```

object_name を検査するオブジェクト名で、*namespace_name* をアクセスするネームスペース名で置き換えます。これは **LMINamespace** オブジェクトを返します。

例17.5 ネームスペースオブジェクトへのアクセス

例17.1「リモートの CIMOM への接続」で作成された **c** 接続オブジェクトの **cimv2** ネームスペースにアクセスするには、インタラクティブプロンプトで以下を入力します。

```
> ns = c.root.cimv2
>
```

17.4.4. クラスを使った操作

LMIShell クラスは、CIMOM が提供するクラスを表します。これらのプロパティーやメソッド、インスタンス、インスタンス名、および ValueMap プロパティーにアクセスしたりこれらを一覧表示することができ、ドキュメンテーションストリングをプリントしたり、新たなインスタンスやインスタンス名を作成することができます。

利用可能なクラスの一覧表示

特定のネームスペースで利用可能なクラスすべてを一覧表示するには、以下のように **print_classes()** メソッドを使用します。

```
namespace_object.print_classes()
```

namespace_object を検査するネームスペースオブジェクトで置き換えます。このメソッドは、利用可能なクラスを標準出力にプリントします。

利用可能なクラス一覧を取得するには、**classes()** メソッドを使用します。

```
namespace_object.classes()
```

このメソッドは文字列の一覧を返します。

例17.6 利用可能なクラスの一覧表示

例17.5「ネームスペースオブジェクトへのアクセス」で作成された **ns** ネームスペースオブジェクトを検査して利用可能なクラスすべてを一覧表示するには、インタラクティブプロンプトで以下を入力します。

```
> ns.print_classes()
CIM_CollectionInSystem
CIM_ConcreteIdentity
```

```
CIM_ControlledBy
CIM_DeviceSAPImplementation
CIM_MemberOfStatusCollection
...
>
```

これらクラスの一覧を **cimv2_classes** という名前の変数に割り当てるには、以下を入力します。

```
> cimv2_classes = ns.classes()
>
```

クラスオブジェクトへのアクセス

CIMOM が提供する特定のクラスオブジェクトにアクセスするには、以下の構文を使用します。

```
namespace_object.class_name
```

namespace_object を検査するネームスペースオブジェクト名に、*class_name* をアクセスするクラス名に置き換えます。

例17.7 クラスオブジェクトへのアクセス

例17.5「ネームスペースオブジェクトへのアクセス」で作成された **ns** ネームスペースオブジェクトの **LMI_IPNetworkConnection** クラスにアクセスしてこれを **cls** という名前の変数に割り当てるには、インタラクティブプロンプトで以下を入力します。

```
> cls = ns.LMI_IPNetworkConnection
>
```

クラスオブジェクトの検査

クラスオブジェクトはすべて、その名前と所属するネームスペースについての情報、および詳細なクラスのドキュメンテーションを保存しています。特定のクラスオブジェクトの名前を取得するには、以下の構文を使用します。

```
class_object.classname
```

class_object を検査するクラスオブジェクト名に置き換えます。これは、オブジェクト名を表す文字列を返します。

クラスオブジェクトが所属するネームスペースについての情報を取得するには、以下を使用します。

```
class_object.namespace
```

これは、ネームスペースを表す文字列を返します。

詳細なクラスのドキュメンテーションを表示するには、以下のように **doc()** メソッドを使用します。

```
class_object.doc()
```

例17.8 クラスオブジェクトの検査

例17.7「クラスオブジェクトへのアクセス」で作成された **cls** クラスオブジェクトを検査してその名前と対応するネームスペースを表示するには、インタラクティブプロンプトで以下を入力します。

```
> cls.classname
'LMI_IPNetworkConnection'
> cls.namespace
'root/cimv2'
>
```

クラスのドキュメンテーションにアクセスするには、以下を入力します。

```
> cls.doc()
Class: LMI_IPNetworkConnection
    SuperClass: CIM_IPNetworkConnection
    [qualifier] string UMLPackagePath: 'CIM::Network::IP'

    [qualifier] string Version: '0.1.0'
...
```

利用可能なメソッドの一覧表示

特定のクラスオブジェクトの利用可能なメソッドすべてを一覧表示するには、以下のように **print_methods()** メソッドを使用します。

```
class_object.print_methods()
```

class_object を検査するクラスオブジェクト名で置き換えます。このメソッドは、利用可能なメソッドを標準出力にプリントします。

利用可能なメソッド一覧を取得するには、**methods()** メソッドを使用します。

```
class_object.methods()
```

このメソッドは文字列の一覧を返します。

例17.9 利用可能なメソッドの一覧表示

例17.7「クラスオブジェクトへのアクセス」で作成された **cls** クラスオブジェクトを検査して利用可能なすべてのメソッドを一覧表示するには、インタラクティブプロンプトで以下を入力します。

```
> cls.print_methods()
RequestStateChange
>
```

これらメソッドの一覧を **service_methods** という名前の変数に割り当てるには、以下を入力します。

```
> service_methods = cls.methods()
>
```

利用可能なプロパティーの一覧表示

特定のクラスオブジェクトの利用可能なプロパティーすべてを一覧表示するには、以下のように **print_properties()** メソッドを使用します。

```
class_object.print_properties()
```

class_object を検査するクラスオブジェクト名で置き換えます。このメソッドは、利用可能なプロパティーを標準出力にプリントします。

利用可能なプロパティー一覧を取得するには、**properties()** メソッドを使用します。

```
class_object.properties()
```

このメソッドは文字列の一覧を返します。

例17.10 利用可能なプロパティーの一覧表示

例17.7「クラスオブジェクトへのアクセス」で作成された **cls** クラスオブジェクトを検査して利用可能なすべてのプロパティーを一覧表示するには、インタラクティブプロンプトで以下を入力します。

```
> cls.print_properties()
RequestedState
HealthState
StatusDescriptions
TransitioningToState
Generation
...
>
```

これらクラスの一覧を **service_properties** という名前の変数に割り当てるには、以下を入力します。

```
> service_properties = cls.properties()
>
```

ValueMap プロパティーの一覧表示と閲覧

CIM クラスには、Managed Object Format (MOF) 定義で *ValueMap プロパティー*が含まれる場合があります。ValueMap プロパティーには定数値が含まれ、これはメソッドを呼び出す場合や戻り値をチェックする場合に便利なものです。

特定のクラスオブジェクトの利用可能な ValueMap プロパティーすべてを一覧表示するには、以下のように **print_valuemap_properties()** メソッドを使用します。

```
class_object.print_valuemap_properties()
```

class_object を検査するクラスオブジェクト名で置き換えます。このメソッドは、利用可能な ValueMap プロパティーを標準出力にプリントします。

利用可能な ValueMap プロパティー一覧を取得するには、**valuemap_properties()** メソッドを使用します。

```
class_object.valuemap_properties()
```

このメソッドは文字列の一覧を返します。

例17.11 ValueMap プロパティーの一覧表示

例17.7「クラスオブジェクトへのアクセス」で作成された **cls** クラスオブジェクトを検査して利用可能なすべての ValueMap プロパティーを一覧表示するには、インタラクティブプロンプトで以下を入力します。

```
> cls.print_valuemap_properties()
RequestedState
HealthState
TransitioningToState
DetailedStatus
OperationalStatus
...
>
```

これら ValueMap プロパティーの一覧を **service_valuemap_properties** という名前の変数に割り当てるには、以下を入力します。

```
> service_valuemap_properties = cls.valuemap_properties()
>
```

特定の ValueMap プロパティーにアクセスするには、以下の構文を使用します。

```
class_object.valuemap_propertyValues
```

valuemap_property をアクセスする ValueMap プロパティー名に置き換えます。

利用可能な定数値すべてを一覧表示するには、以下のように **print_values()** メソッドを使用します。

```
class_object.valuemap_propertyValues.print_values()
```

このメソッドは、名前の付いた利用可能な定数値を標準出力にプリントします。また、**values()** メソッドを使って利用可能な定数値の一覧を取得することもできます。

```
class_object.valuemap_propertyValues.values()
```

このメソッドは文字列の一覧を返します。

例17.12 ValueMap プロパティーへのアクセス

例17.11「ValueMap プロパティーの一覧表示」では、**RequestedState** という名前の ValueMap プロパティーが出てきました。このプロパティーを検査して利用可能な定数値を一覧表示するには、インタラクティブプロンプトで以下を入力します。

```
> cls.RequestedStateValues.print_values()
Reset
NoChange
```

```
NotApplicable
Quiesce
Unknown
...
>
```

これら定数値の一覧を **requested_state_values** という名前の変数に割り当てるには、以下を入力します。

```
> requested_state_values = cls.RequestedStateValues.values()
>
```

特定の定数値にアクセスするには、以下の構文を使用します。

```
class_object.valuemap_propertyValues.constant_value_name
```

constant_value_name を定数値の名前で置き換えます。また、以下のように**value()** メソッドを使用することもできます。

```
class_object.valuemap_propertyValues.value("constant_value_name")
```

特定の定数値の名前を判断するには、**value_name()** メソッドを使用します。

```
class_object.valuemap_propertyValues.value_name("constant_value")
```

このメソッドは文字列を返します。

例17.13 定数値へのアクセス

例17.12「ValueMap プロパティーへのアクセス」では、**RequestedState** プロパティーが**Reset** という名前の定数値を提供していました。この名前の定数値にアクセスするには、インタラクティブプロンプトで以下を入力します。

```
> cls.RequestedStateValues.Reset
11
> cls.RequestedStateValues.value("Reset")
11
>
```

この定数値の名前を判断するには、以下を入力します。

```
> cls.RequestedStateValues.value_name(11)
u'Reset'
>
```

CIMClass オブジェクトの取り込み

クラスメソッドの多くは **CIMClass** オブジェクトへのアクセスを必要としません。呼び出されたメソッドがこのオブジェクトを実際に必要とする際にのみ LMIShell が CIMOM からオブジェクトを取り込むのはこのためです。**CIMClass** オブジェクトを手動で取り込むには、以下のように**fetch()** メソッドを使用します。

```
class_object.fetch()
```

class_object をクラスオブジェクト名に置き換えます。**CIMClass** オブジェクトへのアクセスを必要とするメソッドはこれを自動的に取り込むことに注意してください。

17.4.5. インスタンスを使用する操作

LMIShell インスタンスは、CIMOM が提供するインスタンスを表します。プロパティーの取得や設定、メソッドの一覧表示や呼び出し、ドキュメンテーション文字列のプリント、関連オブジェクト一覧の取得、修正されたオブジェクトの CIMOM へのプッシュ、CIMOM から個別インスタンスの削除ができます。

インスタンスへのアクセス

特定のクラスオブジェクトの利用可能なインスタンスすべてを取得するには、以下のように **instances()** メソッドを使用します。

```
class_object.instances()
```

class_object を検査するクラスオブジェクト名で置き換えます。このメソッドは、**LMIInstance** オブジェクトの一覧を返します。

クラスオブジェクトの最初のインスタンスにアクセスするには、**first_instance()** メソッドを使用します。

```
class_object.first_instance()
```

このメソッドは、**LMIInstance** オブジェクトを返します。

instances() と **first_instance()** は、すべてのインスタンスを一覧表示したり最初のインスタンスを返したりするほか、オプションの引数をサポートして結果をフィルターすることもできます。

```
class_object.instances(criteria)
```

```
class_object.first_instance(criteria)
```

criteria を鍵と値のペアで構成される辞書で置き換えます。ここでの鍵はインスタンスプロパティーを表し、値はこれらプロパティーの必要な値を表します。

例17.14 インスタンスへのアクセス

例17.7「クラスオブジェクトへのアクセス」で作成された **cls** クラスオブジェクトの最初のインスタンスで、 **eth0** と同等の **ElementName** プロパティーがあるものを見つけ、これを**device** という名前の変数に割り当てるには、インタラクティブプロンプトで以下を入力します。

```
> device = cls.first_instance({"ElementName": "eth0"})
>
```

インスタンスの検査

インスタンスオブジェクトはすべて、そのクラス名と所属するネームスペースについての情報、およびプロパティーと値についての詳細なドキュメンテーションを保存しています。さらに、インスタンスオブジェクトを使うと、一意の ID オブジェクトを検索することができます。

特定のインスタンスオブジェクトのクラス名を取得するには、以下の構文を使用します。

```
instance_object.classname
```

instance_object を検査するインスタンスオブジェクト名に置き換えます。これは、クラス名を表す文字列を返します。

インスタンスオブジェクトが所属するネームスペースについての情報を取得するには、以下を使用します。

```
instance_object.namespace
```

これは、ネームスペースを表す文字列を返します。

インスタンスオブジェクト用に一意の ID オブジェクトを検索するには、以下を使用します。

```
instance_object.path
```

これは、**LMIInstanceName** オブジェクトを返します。

最後に、詳細なドキュメンテーションを表示するには、以下のように **doc()** メソッドを使用します。

```
instance_object.doc()
```

例17.15 インスタンスの検査

例17.14「インスタンスへのアクセス」で作成された **device** インスタンスオブジェクトを検査してそのクラス名と対応するネームスペースを表示するには、インタラクティブプロンプトで以下を入力します。

```
> device.classname
u'LMI_IPNetworkConnection'
> device.namespace
'root/cimv2'
>
```

インスタンスオブジェクトのドキュメンテーションにアクセスするには、以下を入力します。

```
> device.doc()
Instance of LMI_IPNetworkConnection
    [property] uint16 RequestedState = '12'

    [property] uint16 HealthState

    [property array] string [] StatusDescriptions
...
```

新規インスタンスの作成

いくつかの CIM プロバイダーでは、特定のクラスオブジェクトの新規インスタンスを作成することができます。クラスオブジェクトの新規インスタンスを作成するには、以下のように **create_instance()** メソッドを使用します。

```
class_object.create_instance(properties)
```

class_object をクラスオブジェクト名で、*properties* を鍵と値のペアで構成される辞書で置き換えます。ここでの鍵はインスタンスプロパティーを表し、値はプロパティーの値を表します。このメソッドは、**LMIInstance** オブジェクトを返します。

例17.16 新規インスタンスの作成

LMI_Group クラスはシステムグループを表し、**LMI_Account** クラスは管理システム上のユーザーアカウントを表します。例17.5「ネームスペースオブジェクトへのアクセス」で作成された **ns** ネームスペースオブジェクトを使用して、**pegasus** という名前のシステムグループと **lmishell-user** という名前のユーザー用にこれら 2 つのインスタンスを作成し、それらを **group** および **user** という名前の変数に割り当てるには、インタラクティブプロンプトで以下を入力します。

```
> group = ns.LMI_Group.first_instance({"Name" : "pegasus"})
> user = ns.LMI_Account.first_instance({"Name" : "lmishell-user"})
>
```

lmishell-user ユーザー用に **LMI_Identity** クラスのインスタンスを取得するには、以下を入力します。

```
> identity = user.first_associator(ResultClass="LMI_Identity")
>
```

LMI_MemberOfGroup クラスは、システムグループのメンバーシップを表します。**LMI_MemberOfGroup** クラスを使って **lmishell-user** を **pegasus** グループに追加するには、以下のようにこのクラスの新規インスタンスを作成します。

```
> ns.LMI_MemberOfGroup.create_instance({
...     "Member" : identity.path,
...     "Collection" : group.path})
LMIInstance(classname="LMI_MemberOfGroup", ...)
>
```

個別インスタンスの削除

CIMOM から特定のインスタンスを削除するには、以下のように **delete()** メソッドを使用します。

```
instance_object.delete()
```

instance_object を削除するインスタンスオブジェクト名に置き換えます。このメソッドは、ブール値を返します。インスタンスを削除すると、そのプロパティーとメソッドにはアクセスできなくなることに注意してください。

例17.17 個別インスタンスの削除

LMI_Account クラスは管理システム上のユーザーアカウントを表します。例17.5「ネームスペースオブジェクトへのアクセス」で作成された **ns** ネームスペースオブジェクトを使用して、**lmishell-user** という名前のユーザー用に **LMI_Account** クラスのインスタンスを作成し、それを **user** という名前の変数に割り当てるには、インタラクティブプロンプトで以下を入力します。

```
> user = ns.LMI_Account.first_instance({"Name" : "lmishell-user"})
>
```

このインスタンスを削除してシステムから **lmishell-user** を取り除くには、以下を入力します。

```
> user.delete()
True
>
```

利用可能なプロパティーの一覧表示とアクセス

特定のインスタンスオブジェクトの利用可能なプロパティーすべてを一覧表示するには、以下のように **print_properties()** メソッドを使用します。

```
instance_object.print_properties()
```

instance_object を検査するインスタンスオブジェクト名で置き換えます。このメソッドは、利用可能なプロパティーを標準出力にプリントします。

利用可能なプロパティー一覧を取得するには、**properties()** メソッドを使用します。

```
instance_object.properties()
```

このメソッドは文字列の一覧を返します。

例17.18 利用可能なプロパティーの一覧表示

例17.14「インスタンスへのアクセス」で作成された **device** インスタンスオブジェクトを検査して利用可能なすべてのプロパティーを一覧表示するには、インタラクティブプロンプトで以下を入力します。

```
> device.print_properties()
RequestedState
HealthState
StatusDescriptions
TransitioningToState
Generation
...
>
```

これらのプロパティーの一覧を **device_properties** という名前の変数に割り当てるには、以下を入力します。

```
> device_properties = device.properties()
>
```

特定のプロパティーの現在の値を取得するには、以下の構文を使用します。

```
instance_object.property_name
```

property_name をアクセスするプロパティー名に置き換えます。

特定のプロパティーの値を修正するには、以下のように値を割り当てます。

```
instance_object.property_name = value
```

value をプロパティーの新たな値に置き換えます。CIMOM への変更を伝達するには、**push()** メソッドも実行する必要があることに注意してください。

```
instance_object.push()
```

このメソッドは、戻り値、戻り値のパラメーター、およびエラー文字列で構成される 3 項目タプルを返します。

例17.19 個別プロパティーへのアクセス

例17.14「インスタンスへのアクセス」で作成された **device** インスタンスオブジェクトを検査して、**SystemName** という名前のプロパティーの値を表示するには、インタラクティブプロンプトで以下を入力します。

```
> device.SystemName
u'server.example.com'
>
```

利用可能なメソッドの一覧表示と使用

特定のインスタンスオブジェクトの利用可能なメソッドすべてを一覧表示するには、以下のように **print_methods()** メソッドを使用します。

```
instance_object.print_methods()
```

instance_object を検査するインスタンスオブジェクト名で置き換えます。このメソッドは、利用可能なメソッドを標準出力にプリントします。

利用可能なメソッド一覧を取得するには、**methods()** メソッドを使用します。

```
instance_object.methods()
```

このメソッドは文字列の一覧を返します。

例17.20 利用可能なメソッドの一覧表示

例17.14「インスタンスへのアクセス」で作成された **device** インスタンスオブジェクトを検査して利用可能なメソッドすべてを一覧表示するには、インタラクティブプロンプトで以下を入力します。

```
> device.print_methods()
RequestStateChange
>
```

これらメソッドの一覧を **network_device_methods** という名前の変数に割り当てるには、以下を入力します。

```
> network_device_methods = device.methods()
>
```

特定のメソッドを呼び出すには、以下の構文を使用します。

```
instance_object.method_name(
    parameter=value,
    ...)
```

instance_object を使用するインスタンスオブジェクト名で、*method_name* を呼び出すメソッド名で、*parameter* を設定するパラメーター名で、*value* をこのパラメーターの値で置き換えます。メソッドは、戻り値、戻り値のパラメーター、およびエラー文字列で構成される 3 項目タプルを返します。

重要

LMIInstance オブジェクトは、自動的にそのコンテンツ (プロパティー、メソッド、修飾子など) を更新**しません**。自動的に更新するようにするには、以下のように**refresh()** メソッドを使用します。

例17.21 メソッドの使用

PG_ComputerSystem クラスは、システムを表します。例17.5「ネームスペースオブジェクトへのアクセス」 で作成された **ns** ネームスペースオブジェクトを使用してこのクラスのインスタンスを作成し、これを **sys** という名前の変数に割り当てるには、インタラクティブプロンプトで以下を入力します。

```
> sys = ns.PG_ComputerSystem.first_instance()
>
```

LMI_AccountManagementService クラスは、システム内でユーザーおよびグループの管理を可能にするメソッドを実装します。このクラスのインスタンスを作成して **acc** という名前の変数に割り当てるには、以下を入力します。

```
> acc = ns.LMI_AccountManagementService.first_instance()
>
```

システム内で **lmishell-user** という名前の新規ユーザーを作成するには、以下のように **CreateAccount()** メソッドを使用します。

```
> acc.CreateAccount(Name="lmishell-user", System=sys)
LMIReturnValue(rval=0, rparams=NocaseDict({u'Account':
LMIInstanceName(classname="LMI_Account"...), u'Identities':
[LMIInstanceName(classname="LMI_Identity"...),
```

```
LMIInstanceName(classname="LMI_Identity"...)]}), errorstr='')
```

LMIShell は、同時メソッド呼び出しをサポートします。このメソッドを使用すると、LMIShell は対応するジョブオブジェクトがその状態を 「finished」 に変更するまで待機し、その後にこのジョブの戻り値パラメーターを返します。特定のメソッドが以下のいずれかのクラスのオブジェクトを返す場合、LMIShell は同時メソッド呼び出しを実行できます。

- **LMI_StorageJob**
- **LMI_SoftwareInstallationJob**
- **LMI_NetworkJob**

LMIShell はまず、待機メソッドに indications を使用します。それが失敗すると、代わりにポーリングメソッドを使います。

同時メソッド呼び出しを実行するには、以下の構文を使用します。

```
instance_object.Syncmethod_name(
    parameter=value,
    ...)
```

instance_object を使用するインスタンスオブジェクト名で、*method_name* を呼び出すメソッド名で、*parameter* を設定するパラメーター名で、*value* をこのパラメーターの値で置き換えます。同時メソッドではすべて、その名前に **Sync** 接頭辞があり、ジョブの戻り値、ジョブの戻り値のパラメーター、およびジョブのエラー文字列で構成される 3 項目タプルを返します。

また、LMIShell がポーリングメソッドのみを使うように強制することもできます。これを行うには、以下のように **PreferPolling** パラメーターを指定します。

```
instance_object.Syncmethod_name(
    PreferPolling=True
    parameter=value,
    ...)
```

ValueMap パラメーターの一覧表示と閲覧

CIM メソッドには、Managed Object Format (MOF) 定義で *ValueMap パラメーター*が含まれる場合があります。ValueMap パラメーターには定数値が含まれます。

特定メソッドの利用可能な ValueMap パラメーターすべてを一覧表示するには、以下のように **print_valuemap_parameters()** メソッドを使用します。

```
instance_object.method_name.print_valuemap_parameters()
```

instance_object を検査するインスタンスオブジェクト名で置き換え、*method_name* を検査するメソッド名で置き換えます。このメソッドは、利用可能な ValueMap パラメーターを標準出力にプリントします。

利用可能な ValueMap パラメーター一覧を取得するには、**valuemap_parameters()** メソッドを使用します。

```
instance_object.method_name.valuemap_parameters()
```

このメソッドは文字列の一覧を返します。

例17.22 ValueMap パラメーターの一覧表示

例17.21「メソッドの使用」 で作成された **acc** インスタンスオブジェクトを検査して **CreateAccount()** メソッドの利用可能なすべての ValueMap パラメーターを一覧表示するには、インタラクティブプロンプトで以下を入力します。

```
> acc.CreateAccount.print_valuemap_parameters()
CreateAccount
>
```

これら ValueMap パラメーターの一覧を **create_account_parameters** という名前の変数に割り当てるには、以下を入力します。

```
> create_account_parameters = acc.CreateAccount.valuemap_parameters()
>
```

特定の ValueMap パラメーターにアクセスするには、以下の構文を使用します。

```
instance_object.method_name.valuemap_parameterValues
```

valuemap_parameter をアクセスする ValueMap パラメーター名に置き換えます。

利用可能な定数値すべてを一覧表示するには、以下のように **print_values()** メソッドを使用します。

```
instance_object.method_name.valuemap_parameterValues.print_values()
```

このメソッドは、名前の付いた利用可能な定数値を標準出力にプリントします。また、**values()** メソッドを使って利用可能な定数値の一覧を取得することもできます。

```
instance_object.method_name.valuemap_parameterValues.values()
```

このメソッドは文字列の一覧を返します。

例17.23 ValueMap パラメーターへのアクセス

例17.22「 ValueMap パラメーターの一覧表示」 では、**CreateAccount** という名前の ValueMap パラメーターが出てきました。このパラメーターを検査して利用可能な定数値を一覧表示するには、インタラクティブプロンプトで以下を入力します。

```
> acc.CreateAccount.CreateAccountValues.print_values()
Operationunsupported
Failed
Unabletosetpasswordusercreated
Unabletocreatehomedirectoryusercreatedandpasswordset
Operationcompletedsuccessfully
>
```

これら定数値の一覧を **create_account_values** という名前の変数に割り当てるには、以下を入力します。

```
> create_account_values =
acc.CreateAccount.CreateAccountValues.values()
>
```

特定の定数値にアクセスするには、以下の構文を使用します。

```
instance_object.method_name.valuemap_parameterValues.constant_value_name
```

constant_value_name を定数値の名前で置き換えます。また、以下のように**value()** メソッドを使用することもできます。

```
instance_object.method_name.valuemap_parameterValues.value("constant_val
ue_name")
```

特定の定数値の名前を判断するには、**value_name()** メソッドを使用します。

```
instance_object.method_name.valuemap_parameterValues.value_name("constan
t_value")
```

このメソッドは文字列を返します。

例17.24 定数値へのアクセス

例17.23「 ValueMap パラメーターへのアクセス」では、**CreateAccount** ValueMap パラメーターが **Failed** という名前の定数値を提供していました。この名前の定数値にアクセスするには、インタラクティブプロンプトで以下を入力します。

```
> acc.CreateAccount.CreateAccountValues.Failed
2
> acc.CreateAccount.CreateAccountValues.value("Failed")
2
>
```

この定数値の名前を判断するには、以下を入力します。

```
> acc.CreateAccount.CreateAccountValues.value_name(2)
u'Failed'
>
```

インスタンスオブジェクトの更新

LMIShell が使用するローカルオブジェクトは、CIMOM 側での CIM オブジェクトを表し、これらが LMIShell のオブジェクトとの作業中に変更されると、古くなる場合があります。特定のインスタンスオブジェクトのプロパティーとメソッドを更新するには、以下のように **refresh()** メソッドを使用します。

```
instance_object.refresh()
```

instance_object を更新するオブジェクト名で置き換えます。このメソッドは、戻り値、戻り値のパラメーター、およびエラー文字列で構成される 3 項目タプルを返します。

例17.25 インスタンスオブジェクトの更新

例17.14「インスタンスへのアクセス」で作成された **device** インスタンスオブジェクトのプロパティーとメソッドを更新するには、インタラクティブプロンプトで以下を入力します。

```
> device.refresh()
LMIReturnValue(rval=True, rparams=NocaseDict({}), errorstr='')
>
```

MOF 表現の表示

インスタンスオブジェクトの MOF (Managed Object Format) 表現を表示するには、以下のように **tomof()** メソッドを使用します。

```
instance_object.tomof()
```

instance_object を検査するインスタンスオブジェクト名で置き換えます。このメソッドは、オブジェクトの MOF 表現を標準出力にプリントします。

例17.26 MOF 表現の表示

例17.14「インスタンスへのアクセス」で作成された **device** インスタンスオブジェクトの MOF 表現を表示するには、インタラクティブプロンプトで以下を入力します。

```
> device.tomof()
instance of LMI_IPNetworkConnection {
        RequestedState = 12;
        HealthState = NULL;
        StatusDescriptions = NULL;
        TransitioningToState = 12;
...
```

17.4.6. インスタンス名を使った操作

LMIShell インスタンス名は、プライマリーキーとその値のセットを持つオブジェクトです。この種類のオブジェクトは、インスタンスを正確に識別します。

インスタンス名へのアクセス

CIMInstance オブジェクトは **CIMInstanceName** オブジェクトによって識別されます。利用可能なすべてのインスタンス名オブジェクト一覧を取得するには、以下のように **instance_names()** メソッドを使用します。

```
class_object.instance_names()
```

class_object を検査するクラスオブジェクト名で置き換えます。このメソッドは、**LMIInstanceName** オブジェクトの一覧を返します。

クラスオブジェクトの最初のインスタンス名オブジェクトにアクセスするには、**first_instance_name()** メソッドを使用します。

```
class_object.first_instance_name()
```

このメソッドは、**LMIInstanceName** オブジェクトを返します。

instance_names() と **first_instance_name()** は、すべてのインスタンス名オブジェクトを一覧表示したり最初のインスタンス名オブジェクトを返したりするほか、オプションの引数をサポートして結果をフィルターすることもできます。

```
class_object.instance_names(criteria)
```

```
class_object.first_instance_name(criteria)
```

criteria を鍵と値のペアで構成される辞書で置き換えます。ここでの鍵は鍵プロパティーを表し、値はこれら鍵プロパティーの必要な値を表します。

例17.27 インスタンス名へのアクセス

例17.7「クラスオブジェクトへのアクセス」で作成された **cls** クラスオブジェクトの最初のインスタンス名で、 **eth0** と同等の **Name** 鍵プロパティーがあるものを見つけ、これを**device_name** という名前の変数に割り当てるには、インタラクティブプロンプトで以下を入力します。

```
> device_name = cls.first_instance_name({"Name": "eth0"})
>
```

インスタンス名の検査

インスタンス名オブジェクトはすべて、そのクラス名と所属するネームスペースについての情報を保存しています。

特定のインスタンス名オブジェクトのクラス名を取得するには、以下の構文を使用します。

```
instance_name_object.classname
```

instance_name_object を検査するインスタンス名オブジェクトの名前に置き換えます。これは、クラス名を表す文字列を返します。

インスタンス名オブジェクトが所属するネームスペースについての情報を取得するには、以下を使用します。

```
instance_name_object.namespace
```

これは、ネームスペースを表す文字列を返します。

例17.28 インスタンス名の検査

例17.27「インスタンス名へのアクセス」で作成された **device_name** インスタンス名オブジェクトを検査してそのクラス名と対応するネームスペースを表示するには、インタラクティブプロンプトで以下を入力します。

```
> device_name.classname
u'LMI_IPNetworkConnection'
```

```
> device_name.namespace
'root/cimv2'
>
```

新規インスタンス名の作成

LMIShell では、リモートオブジェクトのプライマリーキーすべてが分かっている場合、新規のラップ済み **CIMInstanceName** オブジェクトが作成できます。すると、このインスタンス名オブジェクトを使ってインスタンスオブジェクト全体を取得することができるようになります。

クラスオブジェクトの新規インスタンス名を作成するには、以下のように **new_instance_name()** メソッドを使用します。

```
class_object.new_instance_name(key_properties)
```

class_object をクラスオブジェクト名で、*key_properties* を鍵と値のペアで構成される辞書で置き換えます。ここでの鍵は鍵プロパティーを表し、値は鍵プロパティーの値を表します。このメソッドは、**LMIInstanceName** オブジェクトを返します。

例17.29 新規インスタンス名の作成

LMI_Account クラスは管理システム上のユーザーアカウントを表します。例17.5「ネームスペースオブジェクトへのアクセス」 で作成された **ns** ネームスペースオブジェクトを使用して、管理システム上の **lmishell-user** ユーザーを表す **LMI_Account** クラスの新規インスタンス名を作成するには、インタラクティブプロンプトで以下を入力します。

```
> instance_name = ns.LMI_Account.new_instance_name({
...     "CreationClassName" : "LMI_Account",
...     "Name" : "lmishell-user",
...     "SystemCreationClassName" : "PG_ComputerSystem",
...     "SystemName" : "server"})
>
```

鍵プロパティーの一覧表示とアクセス

特定のインスタンス名オブジェクトの利用可能な鍵プロパティーすべてを一覧表示するには、以下のように **print_key_properties()** メソッドを使用します。

```
instance_name_object.print_key_properties()
```

instance_name_object を検査するインスタンス名オブジェクトの名前で置き換えます。このメソッドは、利用可能な鍵プロパティーを標準出力にプリントします。

利用可能な鍵プロパティー一覧を取得するには、**key_properties()** メソッドを使用します。

```
instance_name_object.key_properties()
```

このメソッドは文字列の一覧を返します。

例17.30 利用可能な鍵プロパティーの一覧表示

例17.27「インスタンス名へのアクセス」で作成された **device_name** インスタンス名オブジェクトを検査して利用可能なすべての鍵プロパティーを一覧表示するには、インタラクティブプロンプトで以下を入力します。

```
> device_name.print_key_properties()
CreationClassName
SystemName
Name
SystemCreationClassName
>
```

これらの鍵プロパティーの一覧を **device_name_properties** という名前の変数に割り当てるには、以下を入力します。

```
> device_name_properties = device_name.key_properties()
>
```

特定の鍵プロパティーの現在の値を取得するには、以下の構文を使用します。

```
instance_name_object.key_property_name
```

key_property_name をアクセスする鍵プロパティー名に置き換えます。

例17.31 個別の鍵プロパティーへのアクセス

例17.27「インスタンス名へのアクセス」で作成された **device_name** インスタンス名オブジェクトを検査して、**SystemName** という名前の鍵プロパティーの値を表示するには、インタラクティブプロンフトで以下を入力します。

```
> device_name.SystemName
u'server.example.com'
>
```

インスタンス名をインスタンスに変換する

インスタンス名はインスタンスに変換することが可能です。これを行うには、以下のように **to_instance()** メソッドを使用します。

```
instance_name_object.to_instance()
```

instance_name_object を変換するインスタンス名オブジェクトの名前に置き換えます。このメソッドは、**LMIInstance** オブジェクトを返します。

例17.32 インスタンス名をインスタンスに変換する

例17.27「インスタンス名へのアクセス」で作成された **device_name** インスタンス名オブジェクトをインスタンスオブジェクト変換して、**device** という名前の変数に割り当てるには、インタラクティブプロンプトで以下を入力します。

```
> device = device_name.to_instance()
>
```

17.4.7. 関連するオブジェクトを使った操作

Common Information Model は管理オブジェクト間の関係を定義します。

関連するインスタンスへのアクセス

特定のインスタンスオブジェクトに関連するオブジェクトの全一覧を取得するには、以下のように **associators()** メソッドを使用します。

```
instance_object.associators(
    AssocClass=class_name,
    ResultClass=class_name,
    ResultRole=role,
    IncludeQualifiers=include_qualifiers,
    IncludeClassOrigin=include_class_origin,
    PropertyList=property_list)
```

特定のインスタンスオブジェクトに関連する最初のオブジェクトにアクセスするには、**first_associator()** メソッドを使用します。

```
instance_object.first_associator(
    AssocClass=class_name,
    ResultClass=class_name,
    ResultRole=role,
    IncludeQualifiers=include_qualifiers,
    IncludeClassOrigin=include_class_origin,
    PropertyList=property_list)
```

instance_object を検査するインスタンスオブジェクト名に置き換えます。以下のパラメーターを指定すると結果をフィルターにかけられます。

- **AssocClass** — 返された各オブジェクトは、このクラスもしくはそのサブクラスの 1 つのインスタンスを通してソースオブジェクトに関連付けられている必要があります。デフォルト値は、**None** です。
- **ResultClass** — 返された各オブジェクトは、このクラスのインスタンスかそのサブクラスの 1 つのインスタンスである必要があります。または、このクラスもしくはそのサブクラスの 1 つである必要があります。デフォルト値は、**None** です。
- **Role** — 返された各オブジェクトは、ソースオブジェクトが特定の役割を果たす関連付けでソースオブジェクトと関連付けられている必要があります。ソースオブジェクトを参照する関連付けクラス内のプロパティー名は、このパラメーターの値と合致する必要があります。デフォルト値は、**None** です。
- **ResultRole** — 返された各オブジェクトは、返されたオブジェクトが特定の役割を果たす関連付けでソースオブジェクトと関連付けられている必要があります。返されたオブジェクトを参照する関連付けクラス内のプロパティー名は、このパラメーターの値と合致する必要があります。デフォルト値は、**None** です。

他のパラメーターは以下のとおりです。

- **IncludeQualifiers** — 応答に各オブジェクトのすべての修飾子 (オブジェクトおよび返されたすべてのプロパティー上の修飾子を含む) を QUALIFIER 要素として含めるかどうかを示すブール値。デフォルト値は、**False** です。
- **IncludeClassOrigin** — 返された各オブジェクトですべての適切な要素に CLASSORIGIN 属性が存在すべきかどうかを示すブール値。デフォルト値は、**False** です。
- **PropertyList** — このリストのメンバーは 1 つ以上のプロパティー名を定義します。返されたオブジェクトは、このリストにないプロパティーには要素を含めません。**PropertyList** が空の場合、返されるオブジェクトにはプロパティーは含まれません。**None** の場合は、追加のフィルタリングは定義されません。デフォルト値は、**None** です。

例17.33 関連するインスタンスへのアクセス

LMI_StorageExtent クラスは、システム内で利用可能なブロックデバイスを表します。例17.5「ネームスペースオブジェクトへのアクセス」 で作成された **ns** ネームスペースオブジェクトを使用して、**/dev/vda** という名前のブロックデバイス用に**LMI_StorageExtent** クラスのインスタンスを作成し、これを **vda** という名前の変数に割り当てるには、インタラクティブプロンプトで以下を入力します。

```
> vda = ns.LMI_StorageExtent.first_instance({
...     "DeviceID" : "/dev/vda"})
>
```

このブロックデバイス上のディスクパーティション全一覧を取得して、これを **vda_partitions** という名前の変数に割り当てるには、以下のように **associators()** メソッドを使用します。

```
> vda_partitions = vda.associators(ResultClass="LMI_DiskPartition")
>
```

関連するインスタンス名へのアクセス

特定のインスタンスオブジェクトの利用可能なインスタンス名一覧を取得するには、以下のように **associator_names()** メソッドを使用します。

```
instance_object.associator_names(
    AssocClass=class_name,
    ResultClass=class_name,
    Role=role,
    ResultRole=role)
```

特定のインスタンスオブジェクトの最初の関連するインスタンスにアクセスするには、**first_associator_name()** メソッドを使用します。

```
instance_object.first_associator_name(
    AssocClass=class_object,
    ResultClass=class_object,
    Role=role,
    ResultRole=role)
```

instance_object を検査するインスタンスオブジェクト名に置き換えます。以下のパラメーターを指定すると結果をフィルターにかけられます。

- **AssocClass** — 返された各名前はオブジェクトを識別します。このオブジェクトは、このクラスもしくはそのサブクラスの 1 つのインスタンスを通してソースオブジェクトに関連付けられている必要があります。デフォルト値は、**None** です。
- **ResultClass** — 返された各名前はオブジェクトを識別します。このオブジェクトは、このクラスのインスタンスかそのサブクラスの 1 つのインスタンスである必要があります。または、このクラスもしくはそのサブクラスの 1 つである必要があります。デフォルト値は、**None** です。
- **Role** — 返された各名前は、オブジェクトを識別します。このオブジェクトは、ソースオブジェクトが特定の役割を果たす関連付けでソースオブジェクトと関連付けられている必要があります。ソースオブジェクトを参照する関連付けクラス内のプロパティー名は、このパラメーターの値と合致する必要があります。デフォルト値は、**None** です。
- **ResultRole** — 返された各名前は、オブジェクトを識別します。このオブジェクトは、返された名前オブジェクトが特定の役割を果たす関連付けでソースオブジェクトと関連付けられている必要があります。返されたオブジェクトを参照する関連付けクラス内のプロパティー名は、このパラメーターの値と合致する必要があります。デフォルト値は、**None** です。

例17.34 関連するインスタンス名へのアクセス

例17.33「関連するインスタンスへのアクセス」で作成された **vda** インスタンスオブジェクトを使用して、その関連するインスタンス名一覧を取得し、**vda_partitions** という名前の変数に割り当てるには、以下を入力します。

```
> vda_partitions =
vda.associator_names(ResultClass="LMI_DiskPartition")
>
```

17.4.8. 関連付けオブジェクトを使った操作

Common Information Model は管理オブジェクト間の関係を定義します。関連付けオブジェクトは、他の 2 つのオブジェクト間の関係を定義します。

関連付けインスタンスへのアクセス

特定のターゲットオブジェクトを参照する関連付けオブジェクトの一覧を取得するには、以下のように **references()** メソッドを使用します。

```
instance_object.references(
    ResultClass=class_name,
    Role=role,
    IncludeQualifiers=include_qualifiers,
    IncludeClassOrigin=include_class_origin,
    PropertyList=property_list)
```

特定のターゲットオブジェクトを参照する最初の関連付けオブジェクトにアクセスするには、**first_reference()** メソッドを使用します。

```
instance_object.first_reference(
...     ResultClass=class_name,
...     Role=role,
...     IncludeQualifiers=include_qualifiers,
```

```
...     IncludeClassOrigin=include_class_origin,
...     PropertyList=property_list)
>
```

instance_object を検査するインスタンスオブジェクト名に置き換えます。以下のパラメーターを指定すると結果をフィルターにかけられます。

- **ResultClass** — 返された各オブジェクトは、このクラスのインスタンスかそのサブクラスの 1 つのインスタンスである必要があります。または、このクラスもしくはそのサブクラスの 1 つである必要があります。デフォルト値は、**None** です。
- **Role** — 返された各オブジェクトは、このパラメーターの値に合致する名前を持ったプロパティーでターゲットオブジェクトを参照する必要があります。デフォルト値は、**None** です。

他のパラメーターは以下のとおりです。

- **IncludeQualifiers** — 応答に各オブジェクト (オブジェクトおよび返されたすべてのプロパティー上の修飾子を含む) を QUALIFIER 要素として含めるかどうかを示すブール値。デフォルト値は、**False** です。
- **IncludeClassOrigin** — 返された各オブジェクトですべての適切な要素に CLASSORIGIN 属性が存在すべきかどうかを示すブール値。デフォルト値は、**False** です。
- **PropertyList** — このリストのメンバーは 1 つ以上のプロパティー名を定義します。返されたオブジェクトは、このリストにないプロパティーには要素を含めません。**PropertyList** が空の場合、返されるオブジェクトにはプロパティーは含まれません。**None** の場合は、追加のフィルタリングは定義されません。デフォルト値は、**None** です。

例17.35 関連付けインスタンスへのアクセス

LMI_LANEndpoint クラスは、特定のネットワークインターフェイスデバイスに関連付けられる通信エンドポイントを表します。例17.5「ネームスペースオブジェクトへのアクセス」で作成された **ns** ネームスペースオブジェクトを使用して、eth0 という名前のネットワークインターフェイスデバイス用に **LMI_LANEndpoint** クラスのインスタンスを作成し、これを**lan_endpoint** という名前の変数に割り当てるには、インタラクティブプロンプトで以下を入力します。

```
> lan_endpoint = ns.LMI_LANEndpoint.first_instance({
...     "Name" : "eth0"})
>
```

LMI_BindsToLANEndpoint オブジェクトを参照する最初の関連付けオブジェクトにアクセスし、これを **bind** という名前の変数に割り当てるには、以下を入力します。

```
> bind = lan_endpoint.first_reference(
...     ResultClass="LMI_BindsToLANEndpoint")
>
```

これで **Dependent** プロパティーを使用して、対応するネットワークインターフェイスデバイスの IP アドレスを表す依存 **LMI_IPProtocolEndpoint** クラスにアクセスすることができます。

```
> ip = bind.Dependent.to_instance()
> print ip.IPv4Address
192.168.122.1
>
```

関連付けインスタンス名へのアクセス

特定のインスタンスオブジェクトの関連付けインスタンス名一覧を取得するには、以下のように **reference_names()** メソッドを使用します。

```
instance_object.reference_names(
    ResultClass=class_name,
    Role=role)
```

特定のインスタンスオブジェクトの最初の関連付けインスタンスにアクセスするには、**first_reference_name()** メソッドを使用します。

```
instance_object.first_reference_name(
    ResultClass=class_name,
    Role=role)
```

instance_object を検査するインスタンスオブジェクト名に置き換えます。以下のパラメーターを指定すると結果をフィルターにかけられます。

- **ResultClass** — 返された各オブジェクト名は、このクラスのインスタンスかそのサブクラスの 1 つのインスタンスを識別します。または、このクラスもしくはそのサブクラスの 1 つを識別します。デフォルト値は、**None** です。
- **Role** — 返された各オブジェクトは、このパラメーターの値に合致する名前を持ったプロパティーでターゲットインスタンスを参照するオブジェクトを識別します。デフォルト値は、**None** です。

例17.36 関連付けインスタンス名へのアクセス

例17.35「関連付けインスタンスへのアクセス」で作成された **lan_endpoint** インスタンスオブジェクトを使用して、**LMI_BindsToLANEndpoint** オブジェクトを参照する最初の関連付けインスタンス名にアクセスし、これを **bind** という名前の変数に割り当てるには、以下を入力します。

```
> bind = lan_endpoint.first_reference_name(
...     ResultClass="LMI_BindsToLANEndpoint")
```

これで **Dependent** プロパティーを使用して、対応するネットワークインターフェイスデバイスの IP アドレスを表す依存 **LMI_IPProtocolEndpoint** クラスにアクセスすることができます。

```
> ip = bind.Dependent.to_instance()
> print ip.IPv4Address
192.168.122.1
>
```

17.4.9. Indication を使った操作

Indication は、データにおける特定の変化に反応して発生する特有のイベントへのリアクションです。LMIShell は indication をサブスクライブしてこのようなイベント対応を受信することが可能です。

Indication のサブスクライブ

indication をサブスクライブするには、以下のように **subscribe_indication()** メソッドを使用します。

```
connection_object.subscribe_indication(
    QueryLanguage="WQL",
    Query='SELECT * FROM CIM_InstModification',
    Name="cpu",
    CreationNamespace="root/interop",
    SubscriptionCreationClassName="CIM_IndicationSubscription",
    FilterCreationClassName="CIM_IndicationFilter",
    FilterSystemCreationClassName="CIM_ComputerSystem",
    FilterSourceNamespace="root/cimv2",
    HandlerCreationClassName="CIM_IndicationHandlerCIMXML",
    HandlerSystemCreationClassName="CIM_ComputerSystem",
    Destination="http://host_name:5988")
```

別の方法では、以下のように、このメソッド呼び出しの短いバージョンを使うこともできます。

```
connection_object.subscribe_indication(
    Query='SELECT * FROM CIM_InstModification',
    Name="cpu",
    Destination="http://host_name:5988")
```

connection_object を接続オブジェクトに、*host_name* を indication を配信するシステムのホスト名に置き換えます。

デフォルトでは、LMIShell インタープリターが作成したすべてのサブスクリプションは、インタープリターの終了時にすべて自動的に削除されます。この動作を変更するには、**Permanent=True** キーワードパラメーターを **subscribe_indication()** メソッド呼び出しに渡します。これで、LMIShell がサブスクリプションを削除しなくなります。

例17.37 Indication のサブスクライブ

例17.1「リモートの CIMOM への接続」で作成された **c** 接続オブジェクトを使用して、**cpu** という名前の indication をサブスクライブするには、インタラクティブプロンプトで以下を入力します。

```
> c.subscribe_indication(
...     QueryLanguage="WQL",
...     Query='SELECT * FROM CIM_InstModification',
...     Name="cpu",
...     CreationNamespace="root/interop",
...     SubscriptionCreationClassName="CIM_IndicationSubscription",
...     FilterCreationClassName="CIM_IndicationFilter",
...     FilterSystemCreationClassName="CIM_ComputerSystem",
...     FilterSourceNamespace="root/cimv2",
...     HandlerCreationClassName="CIM_IndicationHandlerCIMXML",
...     HandlerSystemCreationClassName="CIM_ComputerSystem",
...     Destination="http://server.example.com:5988")
LMIReturnValue(rval=True, rparams=NocaseDict({}), errorstr='')
>
```

サブスクライブしている indication の一覧表示

利用可能なサブスクライブしている indication すべてを一覧表示するには、以下のように **print_subscribed_indications()** メソッドを使用します。

```
connection_object.print_subscribed_indications()
```

connection_object を検査する接続オブジェクト名で置き換えます。このメソッドは、サブスクライブしている indication を標準出力にプリントします。

サブスクライブしている indication の一覧を取得するには、**subscribed_indications()** メソッドを使用します。

```
connection_object.subscribed_indications()
```

このメソッドは文字列の一覧を返します。

例17.38 サブスクライブしている indication の一覧表示

例17.1「リモートの CIMOM への接続」で作成された **c** 接続オブジェクトを検査し、サブスクライブしている indication すべてを一覧表示するには、インタラクティブプロンプトで以下を入力します。

```
> c.print_subscribed_indications()
>
```

これらの indication の一覧を **indications** という名前の変数に割り当てるには、以下を入力します。

```
> indications = c.subscribed_indications()
>
```

Indication のサブスクライブ解除

デフォルトでは、LMIShell インタープリターが作成したすべてのサブスクリプションは、インタープリターの終了時にすべて自動的に削除されます。個別のサブスクリプションをこれより前に削除するには、以下のように **unsubscribe_indication()** メソッドを使用します。

```
connection_object.unsubscribe_indication(indication_name)
```

connection_object を検査する接続オブジェクト名で置き換え、*indication_name* を削除する indication の名前で置き換えます。

すべてのサブスクリプションを削除するには、**unsubscribe_all_indications()** メソッドを使用します。

```
connection_object.unsubscribe_all_indications()
```

例17.39 Indication のサブスクライブ解除

例17.1「リモートの CIMOM への接続」で作成された **c** 接続オブジェクトを使用して、例17.37「Indication のサブスクライブ」で作成された indication のサブスクライブを解除するには、インタラクティブプロンプトで以下を入力します。

```
> c.unsubscribe_indication('cpu')
LMIReturnValue(rval=True, rparams=NocaseDict({}), errorstr='')
>
```

Indication ハンドラーの実装

subscribe_indication() メソッドを使うと、indication を配信するシステムのホスト名を指定できます。以下の例では、indication ハンドラーの実装方法を示します。

```
> def handler(ind, arg1, arg2, **kwargs):
...     exported_objects = ind.exported_objects()
...     do_something_with(exported_objects)
> listener = LmiIndicationListener("0.0.0.0", listening_port)
> listener.add_handler("indication-name-XXXXXXXX", handler, arg1, arg2,
**kwargs)
> listener.start()
>
```

ハンドラーの最初の引数は **LmiIndication** オブジェクトで、これには indication がエクスポートしたメソッドとオブジェクトの一覧が含まれます。他のパラメーターはユーザー固有なので、ハンドラーをリスナーに追加する際にこれらの引数を指定する必要があります。

上記の例では、**add_handler()** メソッド呼び出しは、8 つの「X」 文字の特別な文字列を使用しています。この文字は、ハンドラー名が競合しないように、リスナーが生成するランダムな文字列で置き換えられます。ランダムな文字列を使用するには、indication リスナーを最初に開始してその次に indication をサブスクライブします。そうすることで、ハンドラーオブジェクトの **Destination** プロパティーに、***schema://host_name/random_string*** の値が含れます。

例17.40 Indication ハンドラーの実装

以下のスクリプトでは、**192.168.122.1** にある管理システムを監視し、新規ユーザーアカウントの作成時に常に **indication_callback()** 関数を呼び出すハンドラーの書き込み方法を示しています。

```
#!/usr/bin/lmishell

import sys
from time import sleep
from lmi.shell.LMIUtil import LMIPassByRef
from lmi.shell.LMIIndicationListener import LMIIndicationListener

# These are passed by reference to indication_callback
var1 = LMIPassByRef("some_value")
var2 = LMIPassByRef("some_other_value")

def indication_callback(ind, var1, var2):
    # Do something with ind, var1 and var2
    print ind.exported_objects()
    print var1.value
    print var2.value

c = connect("hostname", "username", "password")

listener = LMIIndicationListener("0.0.0.0", 65500)
unique_name = listener.add_handler(
    "demo-XXXXXXXX",     # Creates a unique name for me
    indication_callback, # Callback to be called
    var1,                # Variable passed by ref
```

```
    var2                # Variable passed by ref
)

listener.start()

print c.subscribe_indication(
    Name=unique_name,
    Query="SELECT * FROM LMI_AccountInstanceCreationIndication WHERE
SOURCEINSTANCE ISA LMI_Account",
    Destination="192.168.122.1:65500"
)

try:
    while True:
        sleep(60)
except KeyboardInterrupt:
    sys.exit(0)
```

17.4.10. 使用例

本セクションでは、OpenLMI パッケージと配布されるさまざまな CIM プロバイダーの例を示します。ここではすべての例で、以下の 2 つの変数定義を使用します。

```
c = connect("host_name", "user_name", "password")
ns = c.root.cimv2
```

host_name を管理システムのホスト名に、*user_name* をそのシステム上で実行中の OpenPegasus CIMOM に接続許可されているユーザー名に、*password* をユーザーのパスワードに置き換えます。

OpenLMI サービスプロバイダーの使用

openlmi-service パッケージは、システムサービスの管理用に CIM プロバイダーをインストールします。以下の例では、この CIM プロバイダーを使って利用可能なシステムサービスを一覧表示する方法、およびそれらを開始、停止、有効化、および無効化する方法を示しています。

例17.41 利用可能なサービスの一覧表示

管理システム上で利用可能なすべてのサービスと、そのサービスが開始されているか (**TRUE**) または停止されているか (**FALSE**) についての情報、およびステータス文字列を一覧表示するには、以下のコードスニペットを使用します。

```
for service in ns.LMI_Service.instances():
    print "%s:\t%s" % (service.Name, service.Status)
```

デフォルトで有効になっているサービスのみを一覧表示するには、以下のコードスニペットを使用します。

```
cls = ns.LMI_Service
for service in cls.instances():
    if service.EnabledDefault == cls.EnabledDefaultValues.Enabled:
        print service.Name
```

有効なサービスの場合は、**EnabledDefault** プロパティーの値が **2** と等しくなり、無効なサービスの場合は **3** と等しくなることに注意してください。

cups サービスについての情報を表示するには、以下を使用します。

```
cups = ns.LMI_Service.first_instance({"Name": "cups.service"})
cups.doc()
```

例17.42 サービスの起動と停止

cups サービスを起動して現行ステータスを確認するには、以下のコードスニペットを使用します。

```
cups = ns.LMI_Service.first_instance({"Name": "cups.service"})
cups.StartService()
print cups.Status
cups.StopService()
print cups.Status
```

例17.43 サービスの有効、無効化

cups サービスを有効または無効にして、**EnabledDefault** プロパティーを表示するには、以下のコードスニペットを使用します。

```
cups = ns.LMI_Service.first_instance({"Name": "cups.service"})
cups.TurnServiceOff()
print cups.EnabledDefault
cups.TurnServiceOn()
print cups.EnabledDefault
```

OpenLMI ネットワーキングプロバイダーの使用

openlmi-networking パッケージは、ネットワーキング用の CIM プロバイダーをインストールします。以下の例では、この CIM プロバイダーを使って特定のポート番号に関連付けられた IP アドレスを一覧表示する方法、新規接続を作成する方法、静的 IP アドレスを設定する方法、および接続をアクティブ化する方法を説明しています。

例17.44 特定のポート番号に関連付けられている IP アドレスの一覧表示

eth0 ネットワークインターフェイスに関連付けられた IP アドレスすべてを一覧表示するには、以下のコードスニペットを使用します。

```
device = ns.LMI_IPNetworkConnection.first_instance({'ElementName':
'eth0'})
for endpoint in
device.associators(AssocClass="LMI_NetworkSAPSAPDependency",
ResultClass="LMI_IPProtocolEndpoint"):
    if endpoint.ProtocolIFType ==
ns.LMI_IPProtocolEndpoint.ProtocolIFTypeValues.IPv4:
        print "IPv4: %s/%s" % (endpoint.IPv4Address,
endpoint.SubnetMask)
```

```
    elif endpoint.ProtocolIFType ==
ns.LMI_IPProtocolEndpoint.ProtocolIFTypeValues.IPv6:
        print "IPv6: %s/%d" % (endpoint.IPv6Address,
endpoint.IPv6SubnetPrefixLength)
```

このコードスニペットは、ある **LMI_IPNetworkConnection** クラスに関連付けられた **LMI_IPProtocolEndpoint** クラスを使用します。

デフォルトのゲートウェイを表示するには、以下のコードスニペットを使用します。

```
for rsap in
device.associators(AssocClass="LMI_NetworkRemoteAccessAvailableToElement
", ResultClass="LMI_NetworkRemoteServiceAccessPoint"):
    if rsap.AccessContext ==
ns.LMI_NetworkRemoteServiceAccessPoint.AccessContextValues.DefaultGatew
ay:
        print "Default Gateway: %s" % rsap.AccessInfo
```

デフォルトのゲートウェイは、**AccessContext** プロパティーが **DefaultGateway** と等しい **LMI_NetworkRemoteServiceAccessPoint** インスタンスで表されます。

DNS サーバーの一覧を取得するには、以下のようにオブジェクトモデルをトラバースする必要があります。

1. **LMI_NetworkSAPSAPDependency** を使用して、ある **LMI_IPNetworkConnection** に関連付けられた **LMI_IPProtocolEndpoint** インスタンスを取得する。
2. **LMI_DNSProtocolEndpoint** インスタンスに同一の関連付けを使用する。

LMI_NetworkRemoteAccessAvailableToElement で関連付けられている DNS サーバーと同等の **AccessContext** プロパティーを持つ **LMI_NetworkRemoteServiceAccessPoint** インスタンスには、 **AccessInfo** プロパティーに DNS サーバーのアドレスがあります。

RemoteServiceAccessPath の取得には他のパスもあり、エントリーは重複可能です。以下のコートスニペットは、**set()** 関数を使って DNS サーバーから重複エントリーを削除します。

```
dnsservers = set()
for ipendpoint in
device.associators(AssocClass="LMI_NetworkSAPSAPDependency",
ResultClass="LMI_IPProtocolEndpoint"):
    for dnsedpoint in
ipendpoint.associators(AssocClass="LMI_NetworkSAPSAPDependency",
ResultClass="LMI_DNSProtocolEndpoint"):
        for rsap in
dnsedpoint.associators(AssocClass="LMI_NetworkRemoteAccessAvailableToEle
ment", ResultClass="LMI_NetworkRemoteServiceAccessPoint"):
            if rsap.AccessContext ==
ns.LMI_NetworkRemoteServiceAccessPoint.AccessContextValues.DNSServer:
                dnsservers.add(rsap.AccessInfo)
print "DNS:", ", ".join(dnsservers)
```

例17.45 新規接続の作成および静的 IP アドレスの設定

eth0 ネットワークインターフェイスで静的 IPv4 とステートレスの IPv6 設定を作成するには、以下のコードスニペットを使用します。

```
capability = ns.LMI_IPNetworkConnectionCapabilities.first_instance({
'ElementName': 'eth0' })
result = capability.LMI_CreateIPSetting(Caption='eth0 Static',
        IPv4Type=capability.LMI_CreateIPSetting.IPv4TypeValues.Static,

IPv6Type=capability.LMI_CreateIPSetting.IPv6TypeValues.Stateless)
setting = result.rparams["SettingData"].to_instance()
for settingData in
setting.associators(AssocClass="LMI_OrderedIPAssignmentComponent"):
    if setting.ProtocolIFType ==
ns.LMI_IPAssignmentSettingData.ProtocolIFTypeValues.IPv4:
        # Set static IPv4 address
        settingData.IPAddresses = ["192.168.1.100"]
        settingData.SubnetMasks = ["255.255.0.0"]
        settingData.GatewayAddresses = ["192.168.1.1"]
        settingData.push()
```

このコードスニペットは、**LMI_CreateIPSetting()** メソッドを **LMI_IPNetworkConnectionCapabilities** のインスタンス上で呼び出すことで、新規設定を作成します。これは、**LMI_IPNetworkConnectionElementCapabilities** で **LMI_IPNetworkConnection** に関連付けられています。また、**push()** メソッドを使って設定の修正も行います。

例17.46 接続のアクティベーション

設定をネットワークインターフェイスに適用するには、**LMI_IPConfigurationService** クラスの **ApplySettingToIPNetworkConnection()** メソッドを呼び出します。このメソッドは非同期で、ジョブを返します。以下のコードスニペットでは、このメソッドを同期で呼び出す方法を示しています。

```
setting = ns.LMI_IPAssignmentSettingData.first_instance({ "Caption":
"eth0 Static" })
port = ns.LMI_IPNetworkConnection.first_instance({ 'ElementName':
'ens8' })
service = ns.LMI_IPConfigurationService.first_instance()
service.SyncApplySettingToIPNetworkConnection(SettingData=setting,
IPNetworkConnection=port, Mode=32768)
```

Mode パラメーターは、設定の適用方法に影響を与えます。このパラメーターで最もよく使われる値は、以下のとおりです。

- **1** — 設定を直ちに適用し、自動アクティブ化にします。
- **2** — 設定を自動アクティブ化にしますが、直ちには適用しません。
- **4** — 切断して自動アクティブ化を無効にします。
- 設定状態を変更せず、自動アクティブ化のみを無効にします。
- **32768** — 設定を適用します。
- **32769** — 切断します。

OpenLMI ストレージプロバイダーの使用

openlmi-storage パッケージは、ストレージ管理用の CIM プロバイダーをインストールします。以下の例では、この CIM プロバイダーを使ったボリュームグループおよび論理ボリュームの作成方法、ファイルシステムの構築およびマウント方法、システムが認知するブロックデバイスの一覧表示方法を説明します。

c および **ns** の変数に加え、以下の例では下記の変数定義を使用します。

```
MEGABYTE = 1024*1024
storage_service = ns.LMI_StorageConfigurationService.first_instance()
filesystem_service =
ns.LMI_FileSystemConfigurationService.first_instance()
```

例17.47 ボリュームグループの作成

/dev/myGroup/ にあり、3 つのメンバーがありデフォルトの拡張サイズが 4 MB の新規ボリュームグループを作成するには、以下のコードスニペットを使用します。

```
# Find the devices to add to the volume group
# (filtering the CIM_StorageExtent.instances()
# call would be faster, but this is easier to read):
sda1 = ns.CIM_StorageExtent.first_instance({"Name": "/dev/sda1"})
sdb1 = ns.CIM_StorageExtent.first_instance({"Name": "/dev/sdb1"})
sdc1 = ns.CIM_StorageExtent.first_instance({"Name": "/dev/sdc1"})

# Create a new volume group:
(ret, outparams, err) = storage_service.SyncCreateOrModifyVG(
        ElementName="myGroup",
        InExtents=[sda1, sdb1, sdc1])
vg = outparams['Pool'].to_instance()
print "VG", vg.PoolID, \
        "with extent size", vg.ExtentSize, \
        "and",  vg.RemainingExtents, "free extents created."
```

例17.48 論理ボリュームの作成

サイズが 100 MB の論理ボリューム 2 つを作成するには、以下のコードスニペットを使用します。

```
# Find the volume group:
vg = ns.LMI_VGStoragePool.first_instance({"Name":
"/dev/mapper/myGroup"})

# Create the first logical volume:
(ret, outparams, err) = storage_service.SyncCreateOrModifyLV(
        ElementName="Vol1",
        InPool=vg,
        Size=100 * MEGABYTE)
lv = outparams['TheElement'].to_instance()
print "LV", lv.DeviceID, \
        "with", lv.BlockSize * lv.NumberOfBlocks,\
        "bytes created."

# Create the second logical volume:
(ret, outparams, err) = storage_service.SyncCreateOrModifyLV(
        ElementName="Vol2",
```

```
        InPool=vg,
        Size=100 * MEGABYTE)
lv = outparams['TheElement'].to_instance()
print "LV", lv.DeviceID, \
        "with", lv.BlockSize * lv.NumberOfBlocks, \
        "bytes created."
```

例17.49 ファイルシステムの作成

例17.48「論理ボリュームの作成」の論理ボリューム **lv** に **ext3** ファイルシステムを作成するには、以下のコードスニペットを使用します。

```
(ret, outparams, err) = filesystem_service.SyncLMI_CreateFileSystem(

FileSystemType=filesystem_service.LMI_CreateFileSystem.FileSystemTypeVa
lues.EXT3,
        InExtents=[lv])
```

例17.50 ファイルシステムのマウント

例17.49「ファイルシステムの作成」で作成したファイルシステムをマウントするには、以下のコードスニペットを使用します。

```
# Find the file system on the logical volume:
fs = lv.first_associator(ResultClass="LMI_LocalFileSystem")

mount_service = ns.LMI_MountConfigurationService.first_instance()
(rc, out, err) = mount_service.SyncCreateMount(
     FileSystemType='ext3',
     Mode=32768, # just mount
     FileSystem=fs,
     MountPoint='/mnt/test',
     FileSystemSpec=lv.Name)
```

例17.51 ブロックデバイスの一覧表示

システムが認識するブロックデバイスすべてを一覧表示するには、以下のコードスニペットを使用します。

```
devices = ns.CIM_StorageExtent.instances()
for device in devices:
    if lmi_isinstance(device, ns.CIM_Memory):
        # Memory and CPU caches are StorageExtents too, do not print
them
        continue
    print device.classname,
    print device.DeviceID,
    print device.Name,
    print device.BlockSize*device.NumberOfBlocks
```

OpenLMI ハードウェアプロバイダーの使用

openlmi-hardware パッケージは、ハードウェアを監視するための CIM プロバイダーをインストールします。以下の例では、この CIM プロバイダーを使って CPU、メモリーモジュール、PCI デバイス、およびマシンの製造元およびモデルについての情報を取得する方法を説明しています。

例17.52 CPU 情報の表示

CPU 名やプロセッサーコア数、ハードウェアスレッド数などの基本的な CPU 情報を表示するには、以下のコードスニペットを使用します。

```
cpu = ns.LMI_Processor.first_instance()
cpu_cap = cpu.associators(ResultClass="LMI_ProcessorCapabilities")[0]
print cpu.Name
print cpu_cap.NumberOfProcessorCores
print cpu_cap.NumberOfHardwareThreads
```

例17.53 メモリー情報の表示

メモリーモジュールの個別サイズなどの基本的情報を表示するには、以下のコードスニペットを使用します。

```
mem = ns.LMI_Memory.first_instance()
for i in mem.associators(ResultClass="LMI_PhysicalMemory"):
    print i.Name
```

例17.54 シャーシ情報の表示

製造元やモデルなどのマシンについての基本的情報を表示するには、以下のコードスニペットを使用します。

```
chassis = ns.LMI_Chassis.first_instance()
print chassis.Manufacturer
print chassis.Model
```

例17.55 PCI デバイスの一覧表示

システムが認識する PCI デバイスすべてを一覧表示するには、以下のコードスニペットを使用します。

```
for pci in ns.LMI_PCIDevice.instances():
    print pci.Name
```

17.5. OpenLMI スクリプトの使用

LMIShell インタープリターは、カスタム管理ツールの開発に使用可能な Python モジュール上に構築されます。OpenLMI スクリプトのプロジェクトは、OpenLMI プロバイダーに数多くのインターフェイス用の Python ライブラリーを提供します。さらに、コマンドラインからこれらのライブラリーに対話するために使用可能な拡張ユーティリティーである **lmi** も合わせて配布されます。

OpenLMI スクリプトをシステムにインストールするには、シェルプロンプトで以下を入力します。

```
easy_install --user openlmi-scripts
```

このコマンドで、Python モジュールと **lmi** ユーティリティーが **~/.local/** ディレクトリーにインストールされます。**lmi** ユーティリティーの機能を拡張するには、以下のコマンドを使用して追加の OpenLMI モジュールをインストールします。

```
easy_install --user package_name
```

利用可能なモジュールの一覧は、Python website を参照してください。OpenLMI スクリプトについての詳細は、official OpenLMI Scripts documentation を参照してください。

17.6. その他のリソース

OpenLMI およびシステム管理全般についての詳細情報は、以下に挙げるリソースを参照してください。

インストールされているドキュメント

- lmishell(1) — **lmishell** クライアントおよびインタープリターの man ページで、実行方法および使用方法の詳細情報が提供されています。

オンラインのドキュメント

- Red Hat Enterprise Linux 7 Networking Guide — Red Hat Enterprise Linux 7 の 『Networking Guide』 では、システム上のネットワークインターフェイスおよびネットワークサービスの設定および管理に関する情報が説明されています。
- Red Hat Enterprise Linux 7 Storage Administration Guide — Red Hat Enterprise Linux 7 の 『Storage Administration Guide』 は、システムでのストレージデバイスおよびファイルシステムの管理方法について説明しています。
- Red Hat Enterprise Linux 7 Power Management Guide — Red Hat Enterprise Linux 7 の 『Power Management Guide』 は、システムの電力消費量を効果的に管理する方法を説明しています。サーバーとノートパソコンの両方で電力消費量を抑えるテクニックと、各テクニックがシステムのパフォーマンス全体に与える影響を説明しています。
- Red Hat Enterprise Linux 7 Linux Domain Identity, Authentication, and Policy Guide — Red Hat Enterprise Linux 7 の 『Linux Domain Identity, Authentication, and Policy Guide』 では、サーバーおよびクライアントを含む IPA ドメインのインストール、設定、管理について説明しています。このガイドは、IT および システム管理者用です。
- FreeIPA Documentation — 『FreeIPA Documentation』 は、FreeIPA Identity Management プロジェクトを使用する際のメインのユーザー資料となります。
- OpenSSL Home Page — 『OpenSSL』 のホームページでは、OpenSSL プロジェクトの概要が説明されています。
- Mozilla NSS Documentation — 『Mozilla NSS Documentation』 は、Mozilla NSS プロジェクトを使用する際のメインのユーザー資料となります。

関連項目

- 3章*ユーザーとグループの管理*では、グラフィカルユーザーインターフェースとコマンドラインを使ったシステムユーザーとグループの管理方法について説明しています。
- 5章*Yum* では、**yum** パッケージマネージャーを使って、コマンドラインでパッケージを検索、インストール、更新、アンインストールする方法について説明しています。
- 6章*systemd によるサービス管理*では、**systemd** の概説と、**systemctl** コマンドを使ったシステムサービスの管理方法、systemd ターゲットの設定方法、および電源管理コマンドの実行方法について説明しています。
- 7章*OpenSSH* では、SSH サーバーの設定方法や、**ssh**、**scp**、および **sftp** クライアントユーティリティーを使ってこのサーバーにアクセスする方法が説明されています。

第18章 ログファイルの表示と管理

*ログファイル*は、システムで実行しているカーネル、サービス、およびアプリケーションを含むシステムに関するメッセージを収納しているファイルです。異なる情報を保存するために、いくつかのタイプのログファイルがあります。例えば、デフォルトのシステムログファイル、セキュリティーメッセージ用のログファイル、cron タスク用のログファイルなどがあります。ログファイルはシステム関連の問題のトラブルシューティングなど、多くの状況で非常に便利なものです。たとえば、カーネルドライバーの読み込み時やシステムへの未承認ログインの試行を探す場合などです。

一部のログファイルは **rsyslogd** というデーモンで制御されています。**rsyslogd** デーモンは以前の **syslogd** に代わる強化版で、拡張フィルタリングやさまざまな設定オプション、出入力モジュール、**TCP** または **UDP** プロトコル経由の移動のサポートを提供します。**rsyslogd** が維持するログファイルのリストは、**/etc/rsyslog.conf** 設定ファイルにあります。ほとんどのログファイルは、**/var/log/** ディレクトリーにあります。

ログファイルは、**systemd** のコンポーネントである**journald** デーモンで制御することもできます。**journald** デーモンは全サービスの標準出力および標準エラー出力に書き込まれたメッセージに加えて、Syslog メッセージ、カーネルログメッセージ、初期 RAM ディスクおよび初期起動メッセージを取り込みます。また、これらをインデックス化して、ユーザーが利用できるようにします。ネイティブのジャーナルファイル形式は構造化およびインデックス付きバイナリファイルで、これは検索を改善し、より速い操作を提供します。また、タイムスタンプやユーザー ID といったメタデータ情報も保存します。**journald** が生成するログファイルはデフォルトで永続的なものではなく、メモリーもしくは **/run/log/journal/** ディレクトリーの小型のリングバッファーに保存されるだけです。ログ記録されるデータ量は空きメモリーの量によります。容量の限界に達すると、一番古いエントリーが削除されます。しかし、この設定は変更可能です。「永続的ストレージの有効化」を参照してください。ジャーナルの詳細情報については、「Journal の使用」を参照してください。

デフォルトでは、これら 2 つのロギングツールはシステム上で共存しています。**journald** デーモンは、トラブルシューティング用の主要ツールです。また、構造化ログメッセージの作成に必要な新たなデータも提供します。**journald** が取得したデータは、**/run/systemd/journal/syslog** ソケットに転送され、データのさらなる処理のために **rsyslogd** が使用する場合があります。しかし、デフォルトで **rsyslog** は **imjournal** 入力モジュール経由で実際の統合を行うので、上記のソケットは避けることになります。また、**omjournal** モジュールを使って **rsyslogd** から **journald** へという逆方向のデータ移動もできます。詳細は、「Rsyslog と Journal の相互作用」を参照してください。統合によりテキストベースのログを一貫性のある形式で保持できることになり、**rsyslogd** に依存するアプリケーションや設定との互換性を確保できます。また、構造化形式で rsyslog メッセージを保持することもできます (「Rsyslog での構造化ロギング」を参照)。

18.1. ログファイルの場所の特定

ほとんどのログファイルは、**/var/log/** ディレクトリーに格納されています。**httpd** や **samba** などの一部のアプリケーションは、**/var/log/** 内のディレクトリーにログファイルを所有しています。

/var/log/ ディレクトリー内には末尾に番号が付いた複数のファイル (例、**cron-20100906**) があることに気付くかもしれません。これらの番号はローテーションを行ったログファイルに追加されたタイムスタンプを表します。ログファイルは、ファイルサイズが大きくなり過ぎないようにローテーションが行われます。**logrotate** パッケージには cron タスクが含まれており、これが **/etc/logrotate.conf** 設定ファイルと **/etc/logrotate.d/** ディレクトリー内の設定ファイルにしたがって自動的にログファイルのローテーションを行います。

18.2. Rsyslog の基本設定

rsyslog の主な設定ファイルは **/etc/rsyslog.conf** です。このファイルでは、*グローバル指示文*、*モジュール*、および *フィルター* と *アクション* の部分で構成される *ルール* を指定できます。また、ハッシュ記号 (#) の後にテキスト形式でコメントを追加することもできます。

18.2.1. フィルター

ルールは、syslog メッセージのサブセットを選択する **フィルター**の部分と、選択したメッセージで何をするかを指定する **アクション**の部分で指定されます。**/etc/rsyslog.conf** 設定ファイル内でルールを定義するには、フィルターとアクションの両方を 1 行内で、1 つ以上の空白かタブでそれらを区切って定義します。

rsyslog は、選択されたプロパティーにしたがって syslog メッセージをフィルターするさまざまな方法を提供します。利用可能なフィルタリングの方法は、**ファシリティー/優先度ベース**、**プロパティーベース**、さらに **式ベース** の 3 種類のフィルターに分けられます。

Facility (ファシリティー) /Priority (優先度) ベースのフィルター

syslog メッセージのフィルター操作で最も使用されよく知られているのは、facility/priority ベースのフィルターを使用する方法です。これは、*facility* と *priority* の 2 つの条件をベースにして syslog メッセージをフィルター処理します。これらの条件は、コンマで区切られます。セレクターを作成するには以下の構文を使用します。

```
FACILITY.PRIORITY
```

ここでは、

- *FACILITY* は、特定の syslog メッセージを作成するサブシステムを指定します。たとえば、**mail** サブシステムはメール関連のすべての syslog メッセージを処理します。*FACILITY* は、以下の キーワードのいずれかで表示されます。**auth**、**authpriv**、**cron**、**daemon**、**kern**、**lpr**、**mail**、**news**、**syslog**、**user**、**ftp**、**uucp**、および **local0** から **local7** まで。

- *PRIORITY* は、syslog メッセージの優先度を指定します。*PRIORITY* は以下のキーワード (または数字) のいずれかで表示できます。**debug** (7)、**info** (6)、**notice** (5)、**warning** (4)、**err** (3)、**crit** (2)、**alert** (1)、および **emerg** (0)。

  上記の構文は、定義された優先度もしくは **より高い** 優先度で syslog メッセージを選択します。いずれかの優先度のキーワード前に等号 (**=**) を付けると、指定された優先度の syslog メッセージのみ選択されることが指定されます。他のすべての優先度は無視されます。反対に、感嘆符 (**!**) を優先度のキーワードの前に付けると、この優先度以外のすべての syslog メッセージが選択されます。

上記で指定されているキーワード以外に、アスタリスク (*****) を使用してすべてのファシリティーもしくは優先度を定義することもできます (アスタリスクをコンマの前か後に配置するかによる)。優先度キーワード **none** を指定すると、優先度のないファシリティーを指定することになります。ファシリティーおよび優先度の条件は、どちらも大文字と小文字を区別しません。

複数のファシリティーや優先度を定義するには、コンマ (**,**) でそれらを区切ります。複数のセレクターを 1 行内で定義するには、セミコロン (**;**) でそれらを区切ります。セレクターフィールド内の各セレクターは、以前のものを上書きすることに注意してください。これにより、パターンから優先度が除外される可能性があります。

> **例18.1 ファシリティー/優先度ベースのフィルター**

以下は簡単なファシリティー/優先度ベースのフィルターの例です。これは、**/etc/rsyslog.conf** で指定できます。優先度ですべてのカーネル syslog メッセージを選択するには、設定ファイルに以下のテキストを追加します。

```
kern.*
```

crit およびそれ以上の優先度があるメール syslog メッセージすべてを選択するには、以下の形式を使用します。

```
mail.crit
```

info または **debug** 優先度以外のすべての cron syslog メッセージを選択するには、設定ファイルで以下の形式を設定します。

```
cron.!info,!debug
```

プロパティーベースのフィルター

プロパティベースのフィルターでは、***timegenerated*** や ***syslogtag*** などのプロパティーで syslog メッセージのフィルター処理ができます。プロパティーに関する詳細情報は、18.2.3項「プロパティー」 を参照してください。指定された各プロパティーは、表18.1「プロパティベースの比較処理　」 で一覧表示されている比較処理のいずれかを使用して特定の値に対して比較できます。プロパティー名と比較処理はどちらも大文字と小文字を区別します。

プロパティーベースのフィルターは、コロン (**:**) で開始する必要があります。フィルターの定義には、以下の構文を使用します。

```
:PROPERTY, [!]COMPARE_OPERATION, "STRING"
```

ここでは、

- *PROPERTY* 属性は希望するプロパティーを指定します。
- オプションの感嘆符　(**!**) は比較処理の出力を無効にします。他のブール値演算子は現在、プロパティーベースのフィルターではサポートされていません。
- *COMPARE_OPERATION* 属性は、表18.1「プロパティベースの比較処理　」 に一覧表示してある比較処理のいずれかを指定します。
- *STRING* 属性は、プロパティーが提供するテキストの比較対象となる値を指定します。この値は、引用符で囲む必要があります。この文字列内の特定の文字をエスケープさせるには (たとえば、引用符 (**"**))、バックスラッシュ (****)　を使用します。

表18.1 プロパティベースの比較処理

比較処理	説明
contains	提供された文字列が、プロパティーで提供されたテキストのいずれかの部分に適合するかどうかをチェックします。大文字と小文字を区別しない比較を実行するには、***contains_i*** を使用します。
isequal	用意された文字列をプロパティーで提供されたテキストすべてに対して比較します。これら 2 つの値が適合するには、完全に等しいものである必要があります。

比較処理	説明
startswith	提供された文字列が、プロパティーで提供されたテキストのちょうど最初にあるかどうかをチェックします。大文字と小文字を区別しない比較を実行するには、***startswith_i*** を使用します。
regex	指定された POSIX BRE (Basic Regular Expression) 正規表現をプロパティーが提供したテキストと比較します。
ereregex	指定された POSIX ERE (Extended Regular Expression) 正規表現をプロパティーが提供したテキストと比較します。
isempty	プロパティーが空かどうかをチェックします。値は破棄されます。これは、いくつかのフィールドが正規化の結果に基づいて設定される正規化データでの作業時に特に有用です。

例18.2 プロパティーベースのフィルター

以下は、プロパティーベースのフィルター例です。これは、**/etc/rsyslog.conf** で指定できます。syslog メッセージのテキストに文字列 **error** が含まれているものを選択するには、以下を使用します。

```
:msg, contains, "error"
```

以下のフィルターは、ホスト名 **host1** から受信した syslog メッセージを選択します。

```
:hostname, isequal, "host1"
```

(**fatal lib error** など) **fatal** と **error** の間にテキストがあるかどうかに関わらず、これらを含まない syslog メッセージを選択するには、以下を入力します。

```
:msg, !regex, "fatal .* error"
```

式ベースのフィルター

式ベースのフィルターは、定義されている算術演算、ブール値演算、または文字列演算にしたがって syslog メッセージを選択します。式ベースのフィルターは、**RainerScript** と呼ばれる **rsyslog** 自体のスクリプト言語を使って複雑なフィルターを構築します。このスクリプトの構文定義とさまざまな式ベースのフィルター例については、18.10項「オンラインのドキュメント」を参照してください。RainerScript は、**rsyslog** の新たな設定フォーマットのベースにもなっています。これについては、「新規設定フォーマットの使用」 を参照してください。

式ベースのフィルターの基本的な構文は、以下のようになります。

```
if EXPRESSION then ACTION else ACTION
```

ここでは、

- *EXPRESSION* 属性は、**$msg startswith 'DEVNAME'** や **$syslogfacility-text == 'local0'** などの評価される式を表します。and/or 演算子を使うことで、単一フィルター内に複数の式を指定できます。
- *ACTION* 属性は、式が **true** の値を返す場合に実行される動作を表します。これは単一のアクションの場合と、波括弧で囲まれた任意の複雑なスクリプトになる場合があります。

- 式ベースのフィルターは、行の最初の **if** キーワードで示されます。**then** キーワードは、*EXPRESSION* を *ACTION* から離します。オプションで、**else** キーワードを使って条件が満たされない場合に実行されるアクションを指定することもできます。

式ベースのフィルターでは、例18.3「式ベースのフィルター」にあるように、波括弧に囲まれた式を使うことで条件をネスト化することができます。このスクリプトでは、式内で **facility/priority-based** フィルターを使うことができます。その一方、ここでは**property-based** フィルターは推奨されません。RainerScriptは、特別関数 **re_match()** および **re_extract()** をともなう正規表現をサポートします。

例18.3 式ベースのフィルター

以下の式には、ネスト化された条件が 2 つ含まれています。**prog1** と呼ばれるプログラムが生成したログファイルが、メッセージ内の文字列 "test" の有無に基づいて 2 つのファイルに分割されます。

```
if $programname == 'prog1' then {
   action(type="omfile" file="/var/log/prog1.log")
   if $msg contains 'test' then
     action(type="omfile" file="/var/log/prog1test.log")
   else
     action(type="omfile" file="/var/log/prog1notest.log")
}
```

18.2.2. アクション

アクションは、定義済みのセレクターでフィルターされたメッセージで実行すべきことを指定します。以下にルール内で定義できるアクションをいくつか示します。

ログファイルへの syslog メッセージの保存

アクションの大半は、どのログファイルに syslog メッセージを保存するかを指定します。これは定義済みセレクターの後にファイルパスを指定することで行います。

```
FILTER PATH
```

ここでの *FILTER* はユーザーが指定したセレクターを、*PATH* はターゲットファイルのパスを示します。

たとえば、以下のルールは、すべての **cron** syslog メッセージを選択するセレクターとそれらのメッセージを **/var/log/cron.log** ログファイルに保存するアクションで構成されています。

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

デフォルトでは、syslog メッセージの生成時に毎回ログファイルは同期されます。同期を省略する場合は、ダッシュ記号 (**-**) を該当するファイルパスの接頭辞として使います。

```
FILTER -PATH
```

書き込みの直後にシステムが終了すると、情報が失われる場合があることに注意してください。ただし、この設定では、特に非常に詳細なログメッセージを生成するプログラムを実行する場合には、保存されるパフォーマンスもあります。

指定したファイルパスは、**静的** でも **動的** でも構いません。静的ファイルは、上記の例で示されているように固定ファイルパスで示されます。動的ファイルパスは、受け取ったメッセージによって異なります。動的ファイルパスは、テンプレートと疑問符 (**?**) の接頭辞で示されます。

```
FILTER ?DynamicFile
```

ここでは、*DynamicFile* は出力パスを修正する定義済みテンプレート名になります。ダッシュ記号 (**-**) の接頭辞を使うと同期を無効にでき、またコロン (**;**) 区切りで複数のテンプレートを使用できます。テンプレートの詳細については、18.2.3項「動的なファイル名の生成」 を参照してください。

指定したファイルが既存の **terminal** または **/dev/console** デバイスである場合、syslog メッセージは標準出力 (特別な **terminal** 処理を使用) へ送信されるか、X Window システムの使用時には使用中のコンソール (特別な **/dev/console** 処理を使用) へそれぞれ送信されます。

ネットワークを使った syslog メッセージの送信

rsyslog を使用すると、ネットワークを使って syslog メッセージを送受信できます。この機能により、1 台のマシン上で複数ホストの syslog メッセージを管理できます。syslog メッセージをリモートマシンに転送するには、以下の構文を使用します。

```
@[(zNUMBER)]HOST:[PORT]
```

ここでは、

- アットマーク (**@**) は、syslog メッセージが **UDP** プロトコルを使用してホストへ転送されることを示します。**TCP** プロトコルを使用するには、2 つのアットマークを空白なしで (**@@**) 使用します。
- オプションの ***zNUMBER*** 設定を使用すると、syslog メッセージの **zlib** 圧縮が可能になります。*NUMBER* 属性は、圧縮レベルを指定します (最低の 1 から最高の 9 まで)。圧縮が得られたことは **rsyslogd** が自動的にチェックします。メッセージが圧縮されるのは圧縮が可能になった場合のみで、60 バイト未満のメッセージは圧縮されません。
- *HOST* 属性は、選択した syslog メッセージを受信するホストを指定します。
- *PORT* 属性は、ホストマシンのポートを指定します。

IPv6 アドレスをホストとして指定する場合は、アドレスを角括弧 (**[**, **]**) で囲みます。

例18.4 ネットワークを使った syslog メッセージの送信

以下の例は、ネットワーク上で syslog メッセージを転送するアクションです (注記: すべてのアクションの前には、いずれかの優先度を持つすべてのメッセージを選択するセレクターが付いています)。メッセージを **UDP** 経由で **192.168.0.1** に転送するには、以下を入力します。

```
*.* @192.168.0.1
```

ポート 18 と **TCP** プロトコルを使ってメッセージを "example.com" に転送するには、以下を使用します。

```
*.* @@example.com:18
```

以下ではメッセージを **zlib** (レベル 9 圧縮) で圧縮し、**UDP** プロトコルを使って **2001:db8::1** に転送します。

```
*.* @(z9)[2001:db8::1]
```

出力チャンネル

出力チャンネルは主にログファイルの最大サイズを指定するために使われます。これは、ログファイルのローテーションに非常に便利なものです (詳細は 「ログローテーション」 を参照してください)。出力チャンネルは基本的に出力アクションについての情報を集めたものです。出力チャンネルは、**$outchannel** 指示文で定義されます。**/etc/rsyslog.conf** で出力チャンネルを定義するには、以下の構文を使用します。

```
$outchannel NAME, FILE_NAME, MAX_SIZE, ACTION
```

ここでは、

- *NAME* 属性は、出力チャンネル名を指定します。
- *FILE_NAME* 属性は、出力ファイル名を指定します。出力チャンネルはファイルにのみ書き込み可能で、パイプやターミナル、その他の出力には書き込みできません。
- *MAX_SIZE* 属性は、(*FILE_NAME* 内にある) 指定されたファイルが拡張可能な最大サイズを表します。この値は **バイト** 単位で指定します。
- *ACTION* 属性は、*MAX_SIZE* で定義された最大サイズに到達した際に取るべきアクションを指定します。

定義済みの出力チャンネルをルール内のアクションとして使用するには、以下を入力します。

```
FILTER :omfile:$NAME
```

例18.5 出力チャンネルのログローテーション

以下の出力は、出力チャンネルを使用した簡単なログローテーションを示しています。まず、出力チャンネルは ***$outchannel*** 指示文により定義されます。

```
 $outchannel log_rotation, /var/log/test_log.log, 104857600,
/home/joe/log_rotation_script
```

その後に、優先度を持つすべての syslog メッセージを選択し、取得した syslog メッセージ上の事前定義された出力チャンネルを実行するルール内で使用されます。

```
*.* :omfile:$log_rotation
```

制限 (例では **100 MB**) に達すると、**/home/joe/log_rotation_script** が実行されます。このスクリプトには、ファイルを異なるフォルダーに移動することやその中の特別なコンテンツを編集すること、単にそれを削除することなど、様々なタスクを含めることができます。

特定ユーザーへの syslog メッセージの送信

メッセージの送信先となるユーザー名を指定することで、**rsyslog** は syslog メッセージを送信することができます (例18.7「複数アクションの指定」で表示)。複数のユーザーを指定するには、各ユーザー名をコンマ (**,**) で区切ります。現在ログオンしている全ユーザーにメッセージを送るには、アスタリスク (*****) を使用します。

プログラムの実行

rsyslog は選択した syslog メッセージ用のプログラムを実行可能とし、**system()** 呼び出しを使用してシェル内でプログラムを実行します。実行するプログラムを指定するには、そのプログラムの前に caret 文字(**^**) を付けます。その後に、受信したメッセージをフォーマットしてそれを 1 行のパラメーターとして指定した実行ファイルに渡すテンプレートを指定します (テンプレートに関する詳細は、「テンプレート」を参照)。

```
FILTER ^EXECUTABLE; TEMPLATE
```

ここでは、*FILTER* 条件の出力は *EXECUTABLE* で表されるプログラムで処理されます。このプログラムは、有効な実行ファイルであればどれでも構いません。*TEMPLATE* をフォーマットするテンプレートに置き換えます。

例18.6 プログラムの実行

以下の例では、すべての優先度の syslog メッセージが選択され、***template*** テンプレートでフォーマットされた後にパラメーターとして **test-program** プログラムに渡されます。その後は、提供されているパラメーターで実行されます。

```
*.* ^test-program;template
```

警告

ホストからメッセージを受信して、シェル実行アクションを使用する際には、コマンドインジェクションに対する脆弱性があります。ユーザーが自身のアクションで実行されるように指定しているプログラム内で、攻撃者が別のコマンドの挿入と実行を試みる可能性があります。セキュリティー脅威の可能性を回避するには、シェル実行アクションの使用をよく考慮してください。

syslog メッセージのデータベースでの保存

選択された syslog メッセージは、**データベースライター**の動作を使用して、直接データベーステーブルに書き込むことができます。データベースライターは、以下の構文を使用します:

```
:PLUGIN:DB_HOST,DB_NAME,DB_USER,DB_PASSWORD;[TEMPLATE]
```

ここでは、

- *PLUGIN* はデータベースの書き込みを処理する指定プラグインを呼び出します (例えば、**ommysql** plug-in)。
- *DB_HOST* 属性は、データベースのホスト名を指定します。
- *DB_NAME* 属性はデータベースの名前を指定します。
- *DB_USER* 属性はデータベースのユーザーを指定します。

- *DB_PASSWORD* 属性は上述のデータベースユーザーが使用するパスワードを指定します。
- *TEMPLATE* 属性は syslog メッセージを修正するテンプレートのオプション使用を指定します。テンプレートに関する詳細は、「テンプレート」を参照してください。

重要

現在 **rsyslog** は、**MySQL** データベースと **PostgreSQL** データベースにのみ対応しています。**MySQL** および **PostgreSQL** のデータベースライター機能を使用するには、*rsyslog-mysql* および *rsyslog-pgsql* パッケージをそれぞれインストールします。また、**/etc/rsyslog.conf** 設定ファイルに適切なモジュールを確実に読み込んでください。

```
$ModLoad ommysql    # Output module for MySQL support
$ModLoad ompgsql    # Output module for PostgreSQL support
```

rsyslog モジュールに関する詳細は、「Rsyslog モジュールの使用」を参照してください。

別の方法として、**omlibdb** モジュールが提供する汎用のデータベースインターフェイスを使用することもできます (サポート対象: Firebird/Interbase、MS SQL、Sybase、SQLLite、Ingres、Oracle、mSQL)。

syslog メッセージの破棄

選択したメッセージを破棄するには、チルダ文字 (~) を使用します。

```
FILTER ~
```

破棄するアクションは、ほとんどの場合、さらに処理をする前にメッセージをフィルターするために使用されます。破棄しなければログファイルを満たしてしまう繰り返されるメッセージを削除したい場合に、これは効果的です。破棄のアクションの結果は、設定ファイルのどこで指定されているかに左右されます。最善の結果を得るには、これらのアクションをアクションリストの最上部に置きます。メッセージは一旦破棄されると、設定ファイルの後ろの行で回復することはできないことに注意してください。

たとえば、以下のルールはすべての cron syslog メッセージを破棄します。

```
cron.* ~
```

複数アクションの指定

各セレクターで、複数のアクションを指定することができます。1 つのセレクターに複数アクションを指定するには、各アクションを別々の行に書き込んでそれらの先頭にアンパサンド文字 (&) を付けます。

```
FILTER ACTION
& ACTION
& ACTION
```

指定されたセレクターが評価されるのは 1 回のみであるため、複数の動作を指定すると、希望する結果の全体的なパフォーマンスが向上します。

例18.7 複数アクションの指定

下記の例では、重大の優先度 (***crit***) を持つすべてのカーネル syslog メッセージはユーザー**user1** に送信され、テンプレート***temp*** によって処理されてから、***test-program*** 実行ファイルに渡され、その後に **UDP** プロトコルを介して**192.168.0.1** に転送されます。

```
kern.=crit user1
& ^test-program;temp
& @192.168.0.1
```

すべてのアクションで、メッセージをフォーマットするテンプレートが後に続きます。テンプレートを指定するには、アクションにセミコロン (**;**) を末尾に付けてからテンプレートの名前を指定します。テンプレートについての詳細は、「テンプレート」を参照してください。

警告

テンプレートはアクションで使用される前に定義する必要があり、それ以外の場合は無視されます。つまり、テンプレート定義は常に **/etc/rsyslog.conf** でルール定義の前にある必要があります。

18.2.3. テンプレート

rsyslog で生成された出力はすべて、**テンプレート** を使ってユーザーのニーズに応じて修正とフォーマットができます。テンプレートを作成するには、**/etc/rsyslog.conf** で以下の構文を使用します。

```
$template TEMPLATE_NAME,"text %PROPERTY% more text", [OPTION]
```

ここでは、

- ***$template*** は、後続のテキストがテンプレートを定義することを意味するテンプレート指示文です。
- ***TEMPLATE_NAME*** はテンプレートの名前です。この名前を使ってテンプレートを参照します。
- 二重引用符 (**"**...**"**) の間にあるものはすべて、実際のテンプレートテキストです。このテキストの中では、改行のための **\n** や行頭復帰のための **\r** などの特別文字が使用できます。**%** や **"** などの他の文字をその文字どおりに使用したい場合は、エスケープする必要があります。
- 2 つのパーセントマーク (**%**) の間にあるテキストは、syslog メッセージの特定のコンテンツにアクセスできるようにする *プロパティー* を指定します。プロパティーに関する詳細は、18.2.3項「プロパティー」を参照してください。
- ***OPTION*** 属性は、テンプレート機能を修正するオプションを指定します。現在サポートされているテンプレートのオプションには、SQL クエリとしてテキストのフォーマットに使用される ***sql*** と ***stdsql*** があります。

注記

データベースライター (詳細情報は、「アクション」 内の 『Storing syslog messages in a database』 を参照してください) は、***sql*** または***stdsql*** のオプションがテンプレート内で指定されているかどうかをチェックします。指定されていないと、データベースライターはアクションを実行しません。これは SQL インジェクションなどのセキュリティーの脅威の可能性を回避するためです。

動的なファイル名の生成

テンプレートを使って、動的なファイル名を生成することができます。プロパティーをファイルパスの一部として指定すると、それぞれ一意のプロパティーに対して新しいファイルが作成されます。これは、syslog メッセージを分類する便利な方法です。

たとえば、メッセージからタイムスタンプを抽出する ***timegenerated*** プロパティーを使用すると、各 syslog メッセージ用に一意のファイル名を生成できます。

```
$template DynamicFile,"/var/log/test_logs/%timegenerated%-test.log"
```

$template 指示文は単にテンプレートを指定するだけです。効果を反映するためには、それをルール内で使用しなければなりません。**/etc/rsyslog.conf** 内で、クエスチョンマーク (**?**) をアクション定義に使って、動的ファイル名テンプレートをマークします。

```
*.* ?DynamicFile
```

プロパティー

テンプレート内 (2 つのパーセントマーク (**%**) の内側) で定義されたプロパティーにより、*プロパティー置換関数* を使って syslog メッセージの各種コンテンツにアクセスできるようになります。テンプレート内 (2 つの引用符 (**"**... **"**)の間) でプロパティーを定義するには、以下の構文を使用します。

```
%PROPERTY_NAME[:FROM_CHAR:TO_CHAR:OPTION]%
```

ここでは、

- *PROPERTY_NAME* 属性は、プロパティー名を指定します。利用可能なすべてのプロパティーとその詳細説明の一覧は、**rsyslog.conf(5)** man ページの **Available Properties** セクションでご覧になれます。
- *FROM_CHAR* 属性と *TO_CHAR* 属性は、指定したプロパティーが動作する文字の範囲を表します。他の方法では、正規表現を使用して文字の範囲を指定することもできます。これを行うには、文字 **R** を *FROM_CHAR* 属性として指定し、希望する正規表現を *TO_CHAR* 属性として指定します。
- *OPTION* 属性は、入力を小文字に変換する **lowercase** オプションのようなプロパティーオプションを指定します。利用可能なすべてのプロパティーオプションとその詳細説明の一覧は、**rsyslog.conf(5)** man ページの **Property Options** セクションでご覧になれます。

簡単なプロパティーの例をいくつか以下に示します。

- 以下のプロパティーは、syslog メッセージのメッセージテキスト全体を取得します。

  ```
  %msg%
  ```

- 以下のプロパティは、syslog メッセージにあるメッセージテキストの最初の 2 文字を取得します。

```
%msg:1:2%
```

- 以下のプロパティは、syslog メッセージの全メッセージテキストを取得して、最後のラインフィード文字を省きます。

```
%msg:::drop-last-lf%
```

- 以下のプロパティは、syslog メッセージを受信して『RFC 3999』日付基準にしたがってそれをフォーマットした時に生成されるタイムスタンプの最初の 10 文字を取得します。

```
%timegenerated:1:10:date-rfc3339%
```

テンプレートの例

このセクションでは、**rsyslog** テンプレートの例をいくつか示しています。

例18.8「詳細な syslog メッセージのテンプレート」は、syslog メッセージをフォーマットするテンプレートを示しています。これにより、メッセージの重要性、機能、メッセージが受信された時のタイムスタンプ、ホスト名、メッセージのタグ、メッセージテキストが出力され、改行後に終了します。

例18.8 詳細な syslog メッセージのテンプレート

```
$template verbose, "%syslogseverity%, %syslogfacility%,
%timegenerated%, %HOSTNAME%, %syslogtag%, %msg%\n"
```

例18.9「ウォールメッセージのテンプレート」は、従来のウォールメッセージ (ログインしていてその ***mesg(1)*** パーミッションが ***yes*** に設定されている全ユーザーに送信されるメッセージ) に似ているテンプレートを示しています。このテンプレートは改行後 (**\r** と **\n**を使用) にメッセージテキストと共にホスト名、メッセージタグ、およびタイムスタンプを出力してベル (**\7** を使用) を鳴らします。

例18.9 ウォールメッセージのテンプレート

```
$template wallmsg,"\r\n\7Message from syslogd@%HOSTNAME% at
%timegenerated% ...\r\n %syslogtag% %msg%\n\r"
```

例18.10「データベースフォーマットをしたメッセージのテンプレート」は、syslog メッセージをフォーマットしてデータベースクエリとして使用できるようにするテンプレートを示しています。テンプレートオプションとして指定されている、テンプレートの最後にある ***sql*** オプションの使用に注目してください。これは、データベースライターに対してメッセージを MySQL **SQL** クエリとしてフォーマットするように指示します。

例18.10 データベースフォーマットをしたメッセージのテンプレート

```
$template dbFormat,"insert into SystemEvents (Message,
Facility,FromHost, Priority, DeviceReportedTime, ReceivedAt,
InfoUnitID, SysLogTag) values ('%msg%', %syslogfacility%,
'%HOSTNAME%',%syslogpriority%, '%timereported:::date-mysql%',
```

```
'%timegenerated:::date-mysql%', %iut%, '%syslogtag%')",sql
```

rsyslog には、**RSYSLOG**_ 接頭辞で識別される事前定義のテンプレートのセットも含まれています。これらは syslog の使用に確保されており、競合を防止するためにこの接頭辞を使用したテンプレートを作成しないことが推奨されます。以下の一覧では、これらの事前定義のテンプレートとその定義を示しています。

RSYSLOG_DebugFormat

プロパティー問題のトラブルシューティングに使われる特別なフォーマット。

```
"Debug line with all properties:\nFROMHOST: '%FROMHOST%',
fromhost-ip: '%fromhost-ip%', HOSTNAME: '%HOSTNAME%', PRI:
%PRI%,\nsyslogtag '%syslogtag%', programname: '%programname%',
APP-NAME: '%APP-NAME%', PROCID: '%PROCID%', MSGID:
'%MSGID%',\nTIMESTAMP: '%TIMESTAMP%', STRUCTURED-DATA:
'%STRUCTURED-DATA%',\nmsg: '%msg%'\nescaped msg: '%msg:::drop-
cc%'\nrawmsg: '%rawmsg%'\n\n\"
```

RSYSLOG_SyslogProtocol23Format

IETF のインターネットドラフト ietf-syslog-protocol-23 で指定されるフォーマット。これは新たな syslog 標準 RFC になると想定されています。

```
"%PRI%1 %TIMESTAMP:::date-rfc3339% %HOSTNAME% %APP-NAME% %PROCID%
%MSGID% %STRUCTURED-DATA% %msg%\n\"
```

RSYSLOG_FileFormat

TraditionalFileFormat に類似したモダンスタイルのログファイルフォーマットですが、高精度のタイムスタンプとタイムゾーン情報を備えています。

```
"%TIMESTAMP:::date-rfc3339% %HOSTNAME% %syslogtag%%msg:::sp-if-no-
1st-sp%%msg:::drop-last-lf%\n\"
```

RSYSLOG_TraditionalFileFormat

精度の低いタイムスタンプを持つ旧式のデフォルトのログファイルフォーマット。

```
"%TIMESTAMP% %HOSTNAME% %syslogtag%%msg:::sp-if-no-1st-
sp%%msg:::drop-last-lf%\n\"
```

RSYSLOG_ForwardFormat

高精度のタイムスタンプとタイムゾーン情報を備えた転送フォーマット。

```
"%PRI%%TIMESTAMP:::date-rfc3339% %HOSTNAME%
%syslogtag:1:32%%msg:::sp-if-no-1st-sp%%msg%\"
```

RSYSLOG_TraditionalForwardFormat

精度の低いタイムスタンプを持つ従来の転送フォーマット。

```
"%PRI%%TIMESTAMP% %HOSTNAME% %syslogtag:1:32%%msg:::sp-if-no-1st-
sp%%msg%\"
```

18.2.4. グローバル指示文

グローバル指示文 (Global directives) は、 **rsyslogd** デーモンに適用される設定オプションです。これらは通常、**rsyslogd** デーモンの動作、またはそれに続くルールに影響する特定の事前定義済み変数の値を指定します。グローバル指示文はすべて、ドルマーク (**$**) で始まります。1 行で指定できるのは、1 つの指示文のみです。以下のグローバル指示文の例では、syslog メッセージキューの最大サイズを指定しています。

```
$MainMsgQueueSize 50000
```

この指示文用に定義されたデフォルトのサイズ (**10,000** メッセージ) は、別の値を指定することで上書きされます (上記の例を参照)。

/etc/rsyslog.conf 設定ファイル内で複数の指示文を定義することもできます。1 つの指示文は、同じ指示文の発生が再度検出されるまですべての設定オプションの動作に影響します。グローバル指示文は、アクションやキュー、デバッグの設定に使用できます。利用可能なすべての設定指示文の一覧は、18.10項「オンラインのドキュメント」でご覧になれます。現在、$ ベースの構文 (「新規設定フォーマットの使用」を参照) に代わる新たな設定フォーマットが開発されています。ただし、従来のグローバル指示文はレガシーフォーマットとして引き続きサポートされます。

18.2.5. ログローテーション

以下に、**/etc/logrotate.conf** 設定ファイルのサンプルを示します:

```
# rotate log files weekly
weekly
# keep 4 weeks worth of backlogs
rotate 4
# uncomment this if you want your log files compressed
compress
```

上記の設定ファイル内のすべての行は、ログファイルすべてに適用されるグローバルオプションを定義しています。この例では、ログファイルは毎週交代され、交代されたログファイルは 4 週間保管されます。交代済みログファイルはすべて、**gzip** で **.gz** 形式に圧縮されます。ハッシュマーク (#) で始まる行はすべてコメントで、これは処理されません。

特定のログファイル用に設定オプションを定義して、それをグローバルオプション下に配置することもできます。ただし、**/etc/logrotate.d/** ディレクトリー内に特定ログファイル用の個別の設定ファイルを作成し、そこに設定オプションを定義することが推奨されます。

設定ファイルが **/etc/logrotate.d/** ディレクトリーに配置されている例を以下に示します。

```
/var/log/messages {
    rotate 5
    weekly
    postrotate
    /usr/bin/killall -HUP syslogd
    endscript
}
```

このファイル内の設定オプションは、**/var/log/messages** ログファイル専用の特有なものです。ここで指定された設定は、可能な場合はグローバルオプションを上書きします。そのため、交代された **/var/log/messages** ログファイルは、グローバルオプションで定義された 4 週間ではなく、5 週間保管されます。

以下は、**logrotate** 設定ファイル内で指定できる指示文の一覧です。

- ***weekly*** — ログファイルの週毎のローテーションを指定します。同様な指示文には以下のものがあります。
  - ***daily***
  - ***monthly***
  - ***yearly***
- ***compress*** — 交代したログファイルの圧縮を有効にします。同様な指示文には、以下のものがあります。
  - ***nocompress***
  - ***compresscmd*** — 圧縮に使用するコマンドを指定します。
  - ***uncompresscmd***
  - ***compressext*** — 圧縮に使用する拡張子を指定します。
  - ***compressoptions*** — 使用される圧縮プログラムに渡すオプションを指定できます。
  - ***delaycompress*** — ログファイルの圧縮を次回のログファイルのローテーションまで延期します。
- ***rotate INTEGER*** — ログファイルが削除される、または特定のアドレスに送信されるまでにログファイルがローテーションされる回数を指定します。値 **0** が指定されると、古いログファイルはローテーションではなく削除されます。
- ***mail ADDRESS*** — このオプションは、***rotate*** 指示文で定義された回数ローテーションされたログファイルを特定のアドレスへメール送信できるようにします。同様な指示文には以下のものがあります。
  - ***nomail***
  - ***mailfirst*** — 間もなく期限切れになるログファイルではなく、交代されたばかりのログファイルがメール送信されるよう指定します。
  - ***maillast*** — 交代されたばかりのログファイルではなく、間もなく期限切れになるログファイルがメール送信されるよう指定します。***mail*** が有効の場合は、これがデフォルトのオプションです。

指示文および設定オプションの一覧は、**logrotate(5)** man ページを参照してください。

18.2.6. 新規設定フォーマットの使用

rsyslog のバージョン 6 では、新たな設定構文が導入されました。この新規設定フォーマットが目指すのは、より強力かつ直感的で、特定の無効なコンストラクトを許可しないことで一般的なミスを防ぐことです。この構文強化は、RainerScript に依存する新たな設定プロセッサーによって可能になっています。レガシーフォーマットはまだ完全にサポートされており、**/etc/rsyslog.conf** 設定ファイルでデフォルト使用となっています。

RainerScript は、ネットワークイベントの処理と **rsyslog** などのイベントプロセッサーの設定用に設計されたスクリプト言語です。RainerScript は、例18.3「式ベースのフィルター」 にあるように、主に式ベースフィルターの定義に使われてきました。最新バージョンの RainerScript は、**input()** および **ruleset()** ステートメントを実装し、**/etc/rsyslog.conf** 設定ファイルは新たなスタイルでの書き込みのみが許可されます。

以下の例は、レガシースタイルのパラメーターで書かれた設定です。

```
$InputFileName /tmp/inputfile
$InputFileTag tag1:
$InputFileStateFile inputfile-state
$InputRunFileMonitor
```

同じ設定を新たなフォーマットステートメントを使用すると、以下のようになります。

```
input(type="imfile" file="/tmp/inputfile" tag="tag1:"
statefile="inputfile-state")
```

これにより、設定で使用されるパラメーター数が大幅に削減され、読みやすさが向上するとともに、実行速度も速まります。RainerScript ステートメントおよびパラメーターに関する詳細情報は、18.10項「オンラインのドキュメント」 を参照してください。

18.2.7. Rulesets

特定な指示文を除いて、**rsyslog** は、フィルター条件と条件が当てはまる場合に実行されるアクションで構成される **rules** で定義したようにメッセージを処理します。従来の**/etc/rsyslog.conf** ファイルでは、ルールはすべて、どの入力メッセージにおいても表示順に評価されます。このプロセスは最初のルールで開始し、すべてのルールが処理されるか、ルールのいずれかがメッセージを破棄するまで続きます。

ただし、ルールは *rulesets* と呼ばれるシーケンスにグループ化されます。この rulesets では、特定のルールの効果を選択された入力のみに制限したり、特定の入力にバインドした明確なアクションを定義することで **rsyslog** のパフォーマンスを強化したりすることができます。つまり、特定の種類のメッセージでは必然的に false となるフィルター条件をスキップすることができます。新たな設定フォーマットでは、**input()** と **ruleset()** のステートメントがこの操作に確保されています。**/etc/rsyslog.conf** 内の ruleset 定義は以下のようになります。

```
ruleset(name="rulesetname") {
      rule
      rule2
      call rulesetname2
      …
}
```

rulesetname を、使用する ruleset の識別子で置き換えます。ruleset 名は**RSYSLOG_** で開始することはできません。このネームスペースが **rsyslog** の使用のために確保されているためです。そして、メッセージに他の ruleset が割り当てられていない場合に実行されるデフォルトのルール一式を **RSYSLOG_DefaultRuleset** が定義します。*rule* と *rule2* では、上記で説明したフィルター-アクションのフォーマットでルールを定義できます。***call*** パラメーターでは、他の ruleset ブロック内から ruleset を呼び出すことでこれらをネスト化できます。

ruleset の作成後は、これが適用される入力を指定する必要があります。

```
input(type="input_type" port="port_num" ruleset="rulesetname");
```

ここでは、メッセージを収集する入力モジュールである *input_type* か、ポート番号である *port_num* で入力メッセージを特定できます。**file** や **tag** といった他のパラメーターは、**input()** 用に指定できます。*rulesetname* を、メッセージに対して評価する ruleset 名で置き換えます。入力メッセージが明示的に ruleset にバインドされていない場合は、デフォルトの ruleset が適用されます。

レガシーフォーマットを使って ruleset を定義することもできます。詳細は、18.10項「オンラインのドキュメント」 を参照してください。

例18.11 ruleset の使用

以下の ruleset は、異なるポートからのリモートメッセージが確実に異なる方法で処理されるようにします。以下を **/etc/rsyslog.conf** に追加します。

```
ruleset(name="remote-10514") {
    action(type="omfile" file="/var/log/remote-10514")
}

ruleset(name="remote-10515") {
    cron.* action(type="omfile" file="/var/log/remote-10515-cron")
    mail.* action(type="omfile" file="/var/log/remote-10515-mail")
}

input(type="imtcp" port="10514" ruleset="remote-10514");
input(type="imtcp" port="10515" ruleset="remote-10515");
```

上記の例の ruleset は、2 つのポートからのリモート入力のログの行き先を定義しています。10515 の場合、メッセージはファシリティーにしたがって分けられます。そして、TCP 入力が有効になり、ruleset にバインドされます。この設定が機能するには、必須モジュール (imtcp) の読み込みが必要なことに注意してください。

18.2.8. syslogd との互換性

rsyslog バージョン 6 からは、**-c** オプションで指定される互換性モードが削除されました。また、syslogd スタイルのコマンドラインオプションは非推奨となり、このコマンドラインオプションを使った **rsyslog** の設定も避けるべきです。ただし、いくつかのテンプレートと指示文を使って syslogd のような動作をさせるように **rsyslogd** を設定することは可能です。

rsyslogd オプションについての詳細情報は、**rsyslogd(8)** man ページを参照してください。

18.3. Rsyslog でのキュー (Queue) を使った操作

キューは、**rsyslog** のコンポーネント間でコンテンツ (主にsyslog メッセージ) を渡すために使用されます。キューを使うと、rsyslog は複数のメッセージを同時に処理することができ、複数のアクションを単一メッセージに一度に適用することができます。**rsyslog** 内のデータフローは、以下のようになります。

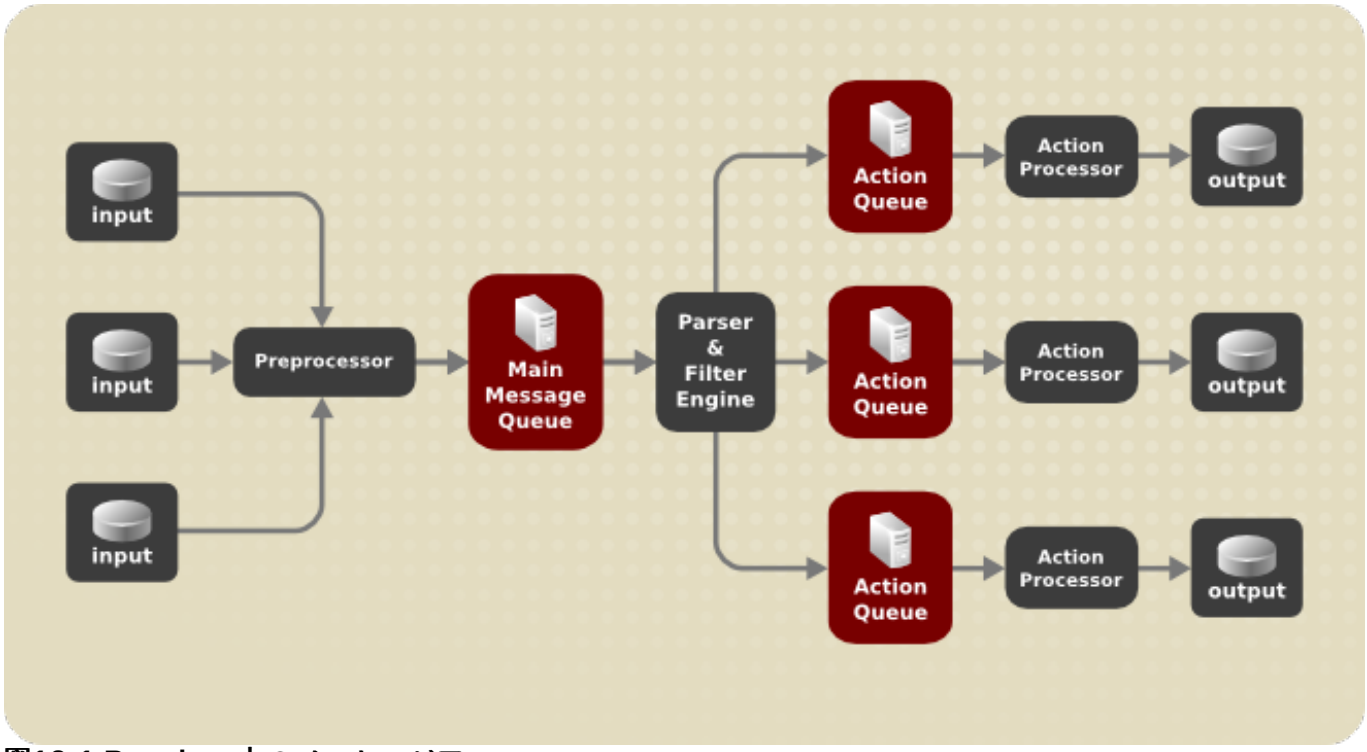


図18.1 Rsyslog 内のメッセージフロー

rsyslog はメッセージを受信すると常に、これをプリプロセッサーに渡します。そして、メインメッセージキュー (*main message queue*) に配置します。メッセージはそこでデキューされ、*rule processor* に渡されるのを待ちます。

rule processor は、分析およびフィルタリングのエンジンです。ここでは、**/etc/rsyslog.conf** で定義されたルールが適用されます。これらのルールに基づいて、rule processor はどのアクションが実行されるかを評価します。アクションはそれぞれ、独自のアクションキューを持っています。メッセージはこのキューにより各アクションプロセッサーに渡され、これが最終的な出力を作成します。この時点では、1 つのメッセージに関して複数のアクションが同時に実行可能であることに注意してください。このためにメッセージは複製され、複数のアクションプロセッサーに渡されます。

1 アクションにつき、1 つのキューのみが可能です。設定により、メッセージはアクションキューなしですぐにアクションプロセッサーに送信されることもあります。これはダイレクトキュー (**direct queues**) (下記参照) と呼ばれる動作です。出力アクションが失敗する場合は、アクションプロセッサーがアクションキューに通知し、すると未処理要素が戻され、しばらくのインターバルの後、アクションが再度試されます。

まとめると、**rsyslog** ではキューは 2 カ所に位置します。1 つは、rule processor の前に単一の**メインメッセージキュー** として。もう 1 つは、さまざまなタイプの出力アクションの前に**アクションキュー** として。キューは、メッセージ処理のパフォーマンス向上につながる以下の 2 つの利点を提供します。

- **rsyslog** 構造内での **decouple** のプロデューサーとコンシューマーのバッファーとして機能します。
- メッセージで実行されるアクションの **並列化** を可能にします。

これら以外には、キューはいくつかの指示文で設定して、システムに最適なパフォーマンスを提供することができます。これらの設定オプションは、以下の章で説明されています。詳細情報は、18.10項「オンラインのドキュメント」 を参照してください。

18.3.1. キューの定義

メッセージが保存される場所によって、キューにはいくつかタイプがあります。最も使われるのは、**direct**、**in-memory**、**disk**、および **disk-assisted in-memory** のキューです。これらのうちのいずれかを、メインメッセージキューおよびアクションキューに選択できます。以下を **/etc/rsyslog.conf** に追加します。

```
$objectQueueType queue_type
```

ここではメインメッセージキューに設定を適用 (*object* を **MainMsg** に置き換える) するか、アクションキューに設定を適用 (*object* を **Action** に置き換える) できます。*queue_type* を、**direct**、**linkedlist**、**fixedarray** (これは in-memory queues)、または **disk** のいずれかに置き換えます。

メインメッセージキューのデフォルト設定は、FixedArray queue で 10,000 メッセージの制限があります。アクションキューはデフォルトで、Direct queues に設定されます。

ダイレクトキュー (Direct Queues)

出力をローカルファイルに書き出すなど、多くの単純な操作では、アクションの前にキューを構築することは不要です。キューの使用を避けるには、以下を使用します。

```
$objectQueueType Direct
```

メインメッセージキューにこのオプションを使用する場合は *object* を **MainMsg** に、アクションキューに使用する場合は **Action** に置き換えます。ダイレクトキューでは、メッセージはプロデューサーからコンシューマーに直接かつ即座に渡されます。

ディスクキュー (Disk Queues)

ディスクキューは、厳密にメッセージをハードドライブに保存します。こうすることで信頼性は非常に高くなりますが、queuing モードのなかでは一番遅くなります。ほとんどのユースケースでは、ディスクキューは推奨されません。ディスクキューを設定するには、**/etc/rsyslog.conf** に以下を入力します。

```
$objectQueueType Disk
```

メインメッセージキューにこのオプションを使用する場合は *object* を **MainMsg** に、アクションキューに使用する場合は **Action** に置き換えます。ディスクキューは、デフォルトサイズの 10 Mb に分けて書き込まれます。このデフォルトサイズは、以下の設定指示文で修正できます。

```
$objectQueueMaxFileSize size
```

ここでは、*size* はディスクキューの一部の指定サイズを表します。定義されたサイズ制限は限定的なものではなく、**rsyslog** はキューエントリーがサイズ制限を超えていても、常に 1 つの完全なキューエントリーを書き込みます。ディスクキューの各部分は、個別ファイルに適合します。これらファイルの命名指示文は、以下のとおりです。

```
$objectQueueFilename name
```

これでファイルに *name* 接頭辞が設定され、1 で始まる 7 桁の数字が続き、各ファイルごとにこれが増加します。

インメモリーキュー (In-memory Queues)

インメモリーキューでは、キューに入ったメッセージがメモリーに保有されるので、プロセスが非常に速くなります。キューに登録済みのデータは、ハードリセットの場合もしくはシャットダウン後に失われます。ただし、**$ActionQueueSaveOnShutdown** 設定を使ってシャットダウン前にデータを保存することができます。インメモリーキューには以下の 2 種類があります。

- **FixedArray** キュー — メインメッセージキューのデフォルトモードで、10,000 要素の制限があります。このタイプのキューは、キュー要素へのポインターを保有する固定かつ事前割り当てのアレイを使用します。これらのポインターのために、キューが空であってもある程度のメモリーが消費されます。しかし、FixedArray は最善のランタイムパフォーマンスを提供し、比較的少ないキューに登録済みのメッセージと高パフォーマンスを期待する場合に最適なものです。
- **LinkedList** キュー — ここではすべての構造物はリンクされたリストに動的に割り当てられるので、メモリーは必要な場合にのみ割り当てられます。LinkedList キューは稀に発生するメッセージバーストにもうまく対応します。

一般的に、疑いが残る場合は LinkedList キューを使ってください。FixedArray と比べてメモリー消費量が少なく、プロセスオーバヘッドが低くて済みます。

インメモリーキューの設定には、以下の構文を使用します。

```
$objectQueueType LinkedList
```

```
$objectQueueType FixedArray
```

メインメッセージキューにこのオプションを使用する場合は *object* を **MainMsg** に、アクションキューに使用する場合は **Action** に置き換えます。

ディスク補助のインメモリーキュー (Disk-Assisted In-memory Queues)

ディスクキューとインメモリーキューは両方とも利点があり、**rsyslog** ではこれらを組み合わせて**ディスク補助のインメモリーキュー** とすることができます。これを行うには、通常のインメモリーキューを設定し、**$objectQueueFileName** 指示文を追加してディスク補助用にファイル名を定義します。するとこのキューは **ディスク補助** となり、インメモリーキューとディスクキューを並べて作業させることになります。

インメモリーキューがいっぱいもしくはシャットダウン後にも継続する必要がある場合に、ディスクキューがアクティベートされます。ディスク補助のキューでは、ディスク固有およびインメモリー固有の設定パラメーターの両方を設定できます。このタイプのキューはおそらく最も広く使われているもので、潜在的に長期間実行され、信頼性が低いアクションで特に便利なものです。

ディスク補助のインメモリーキュー機能を指定するには、**watermarks** と呼ばれるものを使います。

```
$objectQueueHighWatermark number
```

```
$objectQueueLowWatermark number
```

メインメッセージキューにこのオプションを使用する場合は *object* を **MainMsg** に、アクションキューに使用する場合は **Action** に置き換えます。*number* をキューに入っているメッセージの数に置き換えます。インメモリーキューが high watermark で定義している数に達すると、ディスクにメッセージを書き込み始め、インメモリーキューのサイズが low watermark で定義している数に戻るまで続けられます。watermark を正確に設定すると不要なディスク書き込みが最小限に抑えられます。また、ディスクファイルへの書き込みは時間がかかるため、メッセージバースト用にメモリー領域をとっておくこともできます。

つまり、high watermark は、**$objectQueueSize** で設定した全体のキュー容量よりも低くする必要があります。high watermark とキューの全体サイズの違いは、メッセージバースト用に確保されるスペアのメモリーバッファーになります。一方で、high watermark 低くしすぎると、ディスク補助を不要な頻度でオンにすることになります。

18.3.2. キューの管理

キューはすべてのタイプで、使用中の要件に合致するようにさらなる設定が可能です。複数の指示文を使ってアクションキューとメインメッセージキューの両方を修正することができます。現時点では、20 以上のキューパラメーターが利用可能です。18.10項「オンラインのドキュメント」を参照してください。これらの設定のいくつかは一般的に使用されており、また、worker thread management のようにキューの動作を密接に管理する、高度なユーザー用に確保されているものもあります。高度な設定では、**rsyslog** のパフォーマンスやスケジュールキューイングを最適化したり、システムシャットダウン時のキュー動作を修正したりすることができます。

キューのサイズ制限

キューが保有できるメッセージ数は、以下の設定で制限できます。

```
$objectQueueHighWatermark number
```

メインメッセージキューにこのオプションを使用する場合は *object* を **MainMsg** に、アクションキューに使用する場合は **Action** に置き換えます。*number* をキューに入ったメッセージ数で置き換えます。
キューのサイズは、実際のメモリーサイズではなく、メッセージ数でのみ設定可能です。メインメッセージキューおよび ruleset キューのデフォルトサイズは 10,000 メッセージで、アクションキューは 1000 メッセージになります。

ディスク補助キューはデフォルトでは無制限で、この指示文では制限できませんが、以下の設定で物理的なディスク領域をバイト単位で確保することは可能です。

```
$objectQueueMaxDiscSpace number
```

メインメッセージキューにこのオプションを使用する場合は *object* を **MainMsg** に、アクションキューに使用する場合は **Action** に置き換えます。*number* で指定しているサイズ制限に達すると、デキューされたメッセージによって十分なスペースが解放されるまで、メッセージは破棄されます。

メッセージの破棄

キューのメッセージがある数に達すると、重要でないメッセージを破棄して、より優先度が高いエントリーのためにキューのスペースを節約することができます。破棄プロセスを開始するしきい値は、**discard mark** と呼ばれるもので設定できます。

```
$objectQueueDiscardMark number
```

メインメッセージキューにこのオプションを使用する場合は *object* を **MainMsg** に、アクションキューに使用する場合は **Action** に置き換えます。ダイレクトキューでは、メッセージはプロデューサーからコンシューマーに直接かつ即座に渡されます。ここでの *number* は、破棄プロセスを開始する際のキューにあるメッセージ数です。破棄するメッセージを定義するには、以下を使用します。

```
$objectQueueDiscardSeverity priority
```

priority を以下のいずれかのキーワード (もしくは数字) で置き換えます。**debug** (7)、**info** (6)、**notice** (5)、**warning** (4)、**err** (3)、**crit** (2)、**alert** (1)、または **emerg** (0) です。この設定では、新たに入ってくるメッセージとすでにキューに入っていて定義された優先度よりも低いメッセージは、破棄マークに達した後、直ちにキューから消去されます。

タイムフレームの使用

特定の時間帯にキューを処理するよう **rsyslog** を設定することができます。このオプションを使うと、たとえば処理をオフピーク時に移すことが可能です。時間帯を定義するには、以下の構文を使用します。

```
$objectQueueDequeueTimeBegin hour
```

```
$objectQueueDequeueTimeEnd hour
```

hour では、時間帯を区切る時間を指定します。分数を使わずに、24 時間形式を用います。

ワーカースレッドの設定

ワーカースレッドは キューに入っているメッセージに指定されたアクションを実行します。たとえば、メインメッセージキューでは、ワーカーのタスクは、入ってくるメッセージにフィルター論理を適用し、関連のアクションキューに入れることです。メッセージが届くと、ワーカースレッドは自動的に開始します。メッセージ数がある数に達すると、別のワーカースレッドがオンになります。この数字を指定するには、以下を使用します。

```
$objectQueueWorkerThreadMinimumMessages number
```

number を、補助のワーカースレッドを起動するメッセージ数に置き換えます。たとえば*number* を 100 に設定すると、100 を超えるメッセージが届いた際に新たなワーカースレッドが起動します。200 を超えるメッセージが届くと、3 つ目のワーカースレッドが起動する、といったようになります。しかし、多くのワーカースレッドが並列して実行されると効果的でなくなるので、以下を使用してこの最大数を制限できます。

```
$objectQueueWorkerThreads number
```

ここでの *number* は、並列で実行可能なワーカースレッドの最大数になります。メインメッセージキューのデフォルトは、1 スレッドです。ワーカースレッドが一旦起動されると、非アクティブタイムアウトが現れるまで、実行し続けます。タイムアウトを設定するには、以下を入力します。

```
$objectQueueWorkerTimeoutThreadShutdown time
```

time をミリ秒の長さで置き換えます。この設定がないと、ゼロタイムアウトが適用され、ワーカースレッドはメッセージがなくなり次第、直ちに終了します。*time* を **-1** と指定すると、スレッドは終了しません。

バッチのデキュー

複数のメッセージを同時にデキューするように **rsyslog** を設定して、パフォーマンスを高めることができます。このデキューの最大値を設定するには、以下を使用します。

```
$objectQueueDequeueBatchSize number
```

number を同時にデキュー可能な最大数に置き換えます。この数字を高く設定して、許可されるワーカースレッドの結果を大きくすると、メモリー消費量が大きくなることに注意してください。

キューの終了

メッセージを含んでいるキューを終了する際には、ワーカースレッドがキューの処理を完了する間隔を指定することで、データ損失を最小限に抑えることができます。

```
$objectQueueTimeoutShutdown time
```

time をミリ秒で指定します。この期間の後にまだキューに入っているメッセージがある場合は、ワーカーは現在のデータ要素を完了してから終了します。このため、未処理のメッセージは失われます。ワーカーが最終要素を完了する間隔も設定することができます。

```
$objectQueueTimeoutActionCompletion time
```

このタイムアウトが切れると、残りのワーカーはシャットダウンします。シャットダウン時にデータを保存するには、以下を使用します。

```
$objectQueueTimeoutSaveOnShutdown time
```

これが設定されていると、**rsyslog** の終了前にすべてのキュー要素がディスクに保存されます。

18.4. Rsyslog モジュールの使用

モジューラーの設計上、**rsyslog** は、追加の機能性を装備する各種 *モジュール* を提供します。モジュールはサードパーティーによる書き込みが可能であることに注意してください。ほとんどのモジュールは、追加入力 (以下の **入力モジュール** 参照) または出力 (以下の **出力モジュール** 参照) を提供します。その他のモジュールは、各モジュールに固有の機能を提供します。モジュールは、モジュールの読み込み後に利用可能になる追加の設定指示文を提供する場合があります。モジュールを読み込むには、以下の構文を使用します。

```
$ModLoad MODULE
```

ここでの **$ModLoad** は、指定されたモジュールを読み込むグローバル指示文で、*MODULE* は目的のモジュールです。たとえば、**rsyslog** での標準テキストファイルの syslog メッセージへの変換を可能とする Text File Input Module (**imfile**) を読み込む場合、**/etc/rsyslog.conf** 設定ファイル内で以下の行を指定します。

```
$ModLoad imfile
```

rsyslog は多くのモジュールを提供し、これらは以下の主なカテゴリーに分けられます。

- 入力モジュール — 入力モジュールは、様々なソースからメッセージを収集します。入力モジュールの名前は常に **imfile** や **imjournal** のように接頭辞 **im** で始まります。
- 出力モジュール — 出力モジュールはメッセージをネットワーク上に送信したり、データベース内に保存したり、暗号化するなど、様々なターゲットにメッセージを発行するための機能を提供します。出力モジュールの名前は常に **omsnmp** や **omrelp** などのように接頭辞 **om** で始まります。
- パーサーモジュール — これらのモジュールは、カスタムの解析ルールの作成や不正な形式のメッセージ解析に使用されます。C プログラミング言語についてある程度の知識があれば、独自のメッセージパーサーが作成できます。パーサーモジュールの名前は常に **pmrfc5424** や **pmrfc3164** のように接頭辞 **pm** で始まります。

- メッセージ修正モジュール — メッセージ修正モジュールは、syslog メッセージの内容を変更します。このモジュールの名前は、**mm** 接頭辞で始まります。**mmanon** や **mmnormalize**、**mmjsonparse** といったメッセージ修正モジュールは、メッセージの匿名化や正常化に使用されます。
- 文字列生成モジュール — 文字列生成モジュールは、メッセージ内容を基にして文字列を生成し、**rsyslog** で用意されているテンプレート機能と緊密に協力します。テンプレートに関する詳細情報は、「テンプレート」を参照してください。文字列生成モジュールの名前は、**smfile** や **smtradfile** のように常に接頭辞 **sm** で始まります。
- ライブラリーモジュール — ライブラリーモジュールは、他の読み込み可能なモジュール用の機能を提供します。これらのモジュールは、必要であるのにユーザーが設定できない場合に **rsyslog** が自動的に読み込みます。

すべての利用可能なモジュールとそれらの詳しい説明を含む総合一覧は、http://www.rsyslog.com/doc/rsyslog_conf_modules.html を参照して下さい。

警告

rsyslog はモジュールを読み込む際に、モジュールに対して一部の機能とデータへのアクセスを提供します。これはセキュリティー上の脅威となる可能性があります。セキュリティーリスクを最小限にするために、信頼できるモジュールのみを使用するようにしてください。

18.4.1. テキストファイルのインポート

テキストファイル入力モジュールは **imfile** と省略され、**rsyslog** がテキストファイルを syslog メッセージのストリームに変換できるようにします。**imfile** を使って、独自のテキストファイルログを作成するアプリケーションからログメッセージをインポートすることができます。**imfile** を読み込むには、**etc/rsyslog.conf** に以下を追加します。

```
$ModLoad imfile
$InputFilePollInterval int
```

複数ファイルをインポートする際でも、**imfile** の読み込みは 1 回で十分です。**$InputFilePollInterval** グローバル指示文は、**rsyslog** が接続されたテキストファイル内の変更をチェックする頻度を指定します。デフォルト間隔は 10 秒で、これを変更するには、*int* を指定する間隔の秒数で置き換えます。

インポートするテキストファイルを特定するには、**/etc/rsyslog.conf** で以下の構文を使用します。

```
# File 1
$InputFileName path_to_file
$InputFileTag tag:
$InputFileStateFile state_file_name
$InputFileSeverity severity
$InputFileFacility facility
$InputRunFileMonitor

# File 2
$InputFileName path_to_file2
...
```

入力テキストファイルの指定には、4 つの設定が必要になります。

- *path_to_file* をテキストファイルへのパスに置き換えます。

- *tag:* をこのメッセージのタグ名に置き換えます。
- *state_file_name* を **state file** の一意の名前に置き換えます。**State files** は rsyslog 作業ディレクトリーに保存され、監視するファイル用にカーソルを維持し、どのパーティションがすでに処理されたかをマークします。これらを削除すると、ファイル全体が再度読み込まれます。指定する名前は、既存のものと重複しないように注意してください。
- ファイルのモニタリングを有効にする **$InputRunFileMonitor** 指示文を追加します。この設定がないと、テキストファイルは無視されます。

必要な指示文以外に、テキスト入力に適用可能な設定がいくつかあります。*severity* を適切なキーワードで置き換え、*facility* をメッセージを作成したサブシステムを定義するキーワードで置き換えて、メッセージの重要度を設定します。

例18.12 テキストファイルのインポート

Apache HTTP サーバーはテキスト形式でログファイルを作成します。**rsyslog** の処理機能を apache エラーメッセージに適用するには、まず **imfile** モジュールを使ってメッセージをインポートします。**/etc/rsyslog.conf** に以下を追加します。

```
$ModLoad imfile

$InputFileName /var/log/httpd/error_log
$InputFileTag apache-error:
$InputFileStateFile state-apache-error
$InputRunFileMonitor
```

18.4.2. データベースへのメッセージのエクスポート

ログデータの処理は、テキストファイルではなくデータベース内で実行するとより速く、より便利なものになります。使用される DBMS のタイプに基づいて、**ommysql**、**ompgsql**、**omoracle**、または **ommongodb** などの出力モジュールを選択します。別の方法としては、**libdbi** ライブラリーに依存する一般的な **omlibdbi** 出力モジュールを使用します。**omlibdbi** モジュールは、Firebird/Interbase、MS SQL、Sybase、SQLite、Ingres、Oracle、mSQL、MySQL、および PostgreSQL のデータベースシステムをサポートします。

例18.13 データベースへの rsyslog メッセージのエクスポート

rsyslog メッセージを MySQL データベースに保存するには、**/etc/rsyslog.conf** に以下を追加します。

```
$ModLoad ommysql

$ActionOmmysqlServerPort 1234
*.* :ommysql:database-server,database-name,database-userid,database-
password
```

最初に出力モジュールが読み込まれ、その後に通信ポートが指定されます。上記の例では、サーバー名やデータベース名などの追加情報は、最後の行で指定されています。

18.4.3. 暗号化トランスポートの有効化

ネットワーク送信における秘密性と整合性は、**TLS** または **GSSAPI** 暗号化プロトコルで提供することができます。

Transport Layer Security (TLS) は、ネットワーク上の通信セキュリティーを提供するように設計された暗号プロトコルです。TLS を使用する際には、rsyslog メッセージは送信前に暗号化され、送信側と受信側の間で相互認証が存在します。

Generic Security Service API (GSSAPI) は、プログラムがセキュリティーサービスにアクセスするためのアプリケーションプログラミングインターフェイスです。これを **rsyslog** と使用するには、機能する **Kerberos** 環境が必要になります。

18.4.4. RELP の使用

Reliable Event Logging Protocol (RELP) は、コンピューターネットワークにおけるデータロギング用のネットワーキングプロトコルです。信頼性のあるイベントメッセージの配信を提供するように設計されています。これは、メッセージ損失が許されない環境で便利なものです。

18.5. Rsyslog と Journal の相互作用

ここまで説明してきたように、使用中のシステムに存在する **Rsyslog** と **Journal** の 2 つのロギングアプリケーションには、特別のユースケースに適したいくつかの特徴的な機能があります。多くの状況では、これらの機能を組み合わせると便利になります。たとえば、構造化されたメッセージを作成して、ファイルデータベースに保存する場合 (「Rsyslog での構造化ロギング」を参照) などです。この協力で必要となる通信インターフェイスは、**Rsyslog** 側の出入力モジュールと **Journal** の通信ソケットが提供します。

デフォルトでは、**rsyslogd** は、journal ファイルの入力モードに **imjournal** モジュールを使用します。このモジュールを使うと、メッセージだけではなく、**journald** が提供する構造化データもインポートされます。また、古いデータを **journald** からインポートすることもできます (**$ImjournalIgnorePreviousMessages** 指示文で禁止されていない場合。**imjournal** の基本的な設定は 「Journal からのデータのインポート」を参照)。

別の方法では、syslog ベースのアプリケーション用の出力に、**journal** が提供するソケットから読み取るように **rsyslogd** を設定することもできます。ソケットへのパスは、**/run/systemd/journal/syslog** です。プレーンな rsyslog メッセージを維持したい場合は、このオプションを使用します。**imjournal** と比較すると、ソケット入力は現在、ruleset バインディングやフィルタリングなど、より多くの機能を提供しています。ソケットを使って **Journal** データをインポートするには、**/etc/rsyslog.conf** で以下の設定を使用します。

```
$ModLoad imuxsock
$OmitLocalLogging off
```

上記の構文で **imuxsock** モジュールが読み込まれ、**$OmitLocalLogging** 指示文がオフになります。これにより、システムソケットによるインポートが有効になります。このソケットへのパスは、以下のように **/etc/rsyslog.d/listen.conf** で別個に指定されます。

```
$SystemLogSocketName /run/systemd/journal/syslog
```

また、**omjournal** モジュールで **Rsyslog** から **Journal** にメッセージを出力することもできます。**/etc/rsyslog.conf** で出力を以下のように設定します。

```
$ModLoad omjournal

*.* :omjournal:
```

たとえば、以下の設定では、tcp ポート 10514 で受信したものをすべて Journal に転送します。

```
$ModLoad imtcp
$ModLoad omjournal

$RuleSet remote
*.* :omjournal:

$InputTCPServerBindRuleset remote
$InputTCPServerRun 10514
```

18.6. Rsyslog での構造化ロギング

大量のログデータを生成するシステムでは、ログメッセージを **構造化されたフォーマット** で維持すると便利です。構造化メッセージは、特定情報の検索や統計情報の作成、メッセージ構造の変更およびその不整合への対応が容易になります。**Rsyslog** は **JSON** (JavaScript Object Notation) フォーマットを使ってログメッセージに構造を提供します。

以下の非構造化ログメッセージを

```
Oct 25 10:20:37 localhost anacron[1395]: Jobs will be executed
sequentially
```

次の構造化メッセージと比較してください。

```
{"timestamp":"2013-10-25T10:20:37", "host":"localhost",
"program":"anacron", "pid":"1395", "msg":"Jobs will be executed
sequentially"}
```

鍵-値のペアを使った構造化データの検索は、正規表現によるテキストファイルの検索よりも速く、より正確です。構造化データでは、異なるアプリケーションで作成されたメッセージで同一エントリーを検索することもできます。また、JSON ファイルは、追加のパフォーマンスおよび分析機能を提供する MongoDB のようなドキュメントデータベースで保存することもできます。一方で、構造化メッセージは、非構造化メッセージよりも多くのディスクスペースを必要とします。

rsyslog では、メタデータをともなうログメッセージは、**imjournal** を使って **Journal** からプルされます。**mmjsonparse** モジュールでは、**Journal** やその他のソースからインポートしたデータを解析し、たとえばデータベース出力としてさらに処理することができます。解析が成功するには、**mmjsonparse** に入力メッセージが **Lumberjack** プロジェクトで定義された方法で構築されている必要があります。

Lumberjack プロジェクトは、後方互換性がある方法で構造化ロギングを**rsyslog** に追加することを目指しています。構造化メッセージを特定するために、**Lumberjack** は実際の JSON 構造の前に付く **@cee:** 文字列を指定します。**Lumberjack** はまた、JSON 文字列内のエントリーに使用する標準フィールド名の一覧も定義します。**Lumberjack** についての詳細情報は、18.10項「オンラインのドキュメント」 を参照してください。

以下は、lumberjack 形式のメッセージの例です。

```
 @cee: {"pid":17055, "uid":1000, "gid":1000, "appname":"logger",
"msg":"Message text."}
```

この構造を **Rsyslog** 内に構築するには、テンプレートを使用します。「構造化メッセージのフィルタリング」 を参照してください。アプリケーションおよびサーバーは、**libumberlog** ライブラリーを用いて lumberjack 準拠の形式でメッセージを生成できます。**libumberlog** についての詳細情報は、18.10項 「オンラインのドキュメント」 を参照してください。

18.6.1. Journal からのデータのインポート

imjournal モジュールは **Rsyslog** の入力モジュールで、ネイティブに journal ファイルを読み取ります (「Rsyslog と Journal の相互作用」 を参照)。その後、Journal メッセージは、他の rsyslog メッセージのようにテキスト形式でログ記録されます。しかし、さらに処理することで、**Journal** が提供するメタデータを構造化メッセージに変換することが可能です。

Journal から **Rsyslog** にデータをインポートするには、**/etc/rsyslog.conf** で以下の設定を使用します。

```
$ModLoad imjournal

$imjournalPersistStateInterval number_of_messages
$imjournalStateFile path
$imjournalRatelimitInterval seconds
$imjournalRatelimitBurst burst_number
$ImjournalIgnorePreviousMessages off/on
```

- *number_of_messages* では、journal データの保存頻度を指定できます。指定されたメッセージ数に達すると、毎回データが保存されます。
- *path* は、state ファイルへのパスに置き換えます。このファイルは、最後に処理された journal エントリーを追跡します。
- *seconds* では、レート制限の間隔を設定します。この間隔内に処理されるメッセージ数は、*burst_number* で指定して値を超えることはできません。デフォルト設定は、600 秒あたり 20,000 メッセージです。Rsyslog は、この指定された時間枠内で最大バースト後に届いたメッセージを破棄します。
- **$ImjournalIgnorePreviousMessages** を使うと、現在 Journal にあるメッセージを無視して、新しいメッセージのみをインポートできます。このメッセージは、state ファイルが指定されていない場合に使用されます。デフォルト設定は、**off** です。この設定が off で、state ファイルがない場合、Journal 内の全メッセージは前の rsyslog セッションで処理されていても、処理されることに注意してください。

注記

imjournal は、従来のシステムログ入力である**imuxsock** モジュールと同時に使用できます。ただし、メッセージの重複を避けるために、**imuxsock** が Journal のシステムソケットを読み取らないようにする必要があります。これを行うには、**$OmitLocalLogging** 指示文を使用します。

```
$ModLoad imuxsock
$ModLoad imjournal

$OmitLocalLogging on
$AddUnixListenSocket /run/systemd/journal/syslog
```

Journal が保存したデータおよびメタデータはすべて、構造化メッセージに変換することができます。こ

れらメタデータエントリーの一部は、例18.15「詳細な journalctl 出力」に一覧表示されています。journal フィールドの全一覧については、**systemd.journal-fields(7)** man ページを参照してください。たとえば、kernel を元とするメッセージが使用する **kernel journal fields** にフォーカスすることができます。

18.6.2. 構造化メッセージのフィルタリング

rsyslog の解析モジュールで必要となる lumberjack 形式のメッセージを作成するには、以下のテンプレートを使用します。

```
template(name="CEETemplate" type="string" string="%TIMESTAMP% %HOSTNAME%
%syslogtag% @cee: %$!all-json%\n")
```

このテンプレートは **@cee:** 文字列を JSON 文字列の前に付加し、たとえば、**omfile** モジュールで出力ファイルを作成する際に適用することができます。JSON フィールド名にアクセスするには、**$!** 接頭辞を使用します。たとえば、以下のフィルター条件では、特定の **hostname** と **UID** のメッセージが検索されます。

```
($!hostname == "hostname" && $!UID== "UID")
```

18.6.3. JSON の解析

構造化メッセージの解析には、**mmjsonparse** モジュールが使用されます。これらのメッセージは **Journal** から来る場合もあれば、他の入力ソースから来る場合もあり、**Lumberjack** プロジェクトで定義された方法でフォーマットされている必要があります。これらのメッセージは **@cee:** 文字列の存在により識別されます。そして、JSON 構造が有効かどうかを **mmjsonparse** がチェックした後、メッセージが解析されます。

lumberjack 形式の JSON メッセージを **mmjsonparse** で解析するには、**/etc/rsyslog.conf** で以下の設定を使用します。

```
$ModLoad mmjsonparse

*.* :mmjsonparse:
```

この例では、**mmjsonparse** モジュールが最初の行で読み込まれ、その後にすべてのメッセージがそこに転送されます。現在は、**mmjsonparse** で使用可能な設定パラメーターはありません。

18.6.4. MongoDB でのメッセージの保存

Rsyslog は、**ommongodb** 出力モジュールで JSON ログの MongoDB ドキュメントデータベースでの保存をサポートします。

MongoDB にログメッセージを転送するには、**/etc/rsyslog.conf** で以下の構文を使用します (**ommongodb** 用の設定パラメーターは、新たな設定フォーマットでのみ利用可能です。「新規設定フォーマットの使用」を参照してください)。

```
$ModLoad ommongodb

*.* action(type="ommongodb" server="DB_server" serverport="port"
db="DB_name" collection="collection_name" uid="UID" pwd="password")
```

- *DB_server* を MongoDB サーバーの名前もしくはアドレスに置き換えます。*port* を指定して、MongoDB サーバーから非標準ポートを選択します。*port* のデフォルト値は **0** で、通常はこのパラメーターを変換する必要はありません。
- *DB_name* では、出力先となる MongoDB サーバー上のデータベースを特定します。*collection_name* をこのデータベース内のコレクション名で置き換えます。MongoDB ではコレクションはドキュメントのグループで、RDBMS テーブルと同等のものです。
- *UID* と *password* を置き換えて、ログインの詳細を設定します。

テンプレートを使うと、最終的なデータベース出力の形式を形成できます。デフォルトでは、**ryslog** は標準 **lumberjack** フィールド名をベースにしたテンプレートを使用します。

18.7. Rsyslog のデバッグ

rsyslogd をデバッグモードで実行するには、以下のコマンドを使用します。

```
rsyslogd -dn
```

このコマンドでは、**rsyslogd** がデバッグ情報を作成し、標準出力にプリントします。**-n** は "no fork" を意味します。環境変数でデバッグを修正することができ、たとえばログファイルにデバッグ出力を保存することができます。**rsyslogd** を起動する前に、以下をコマンドラインに入力します。

```
export RSYSLOG_DEBUGLOG="path"
export RSYSLOG_DEBUG="Debug"
```

path を、デバッグ情報をログ記録するファイルの場所に置き換えます。RSYSLOG_DEBUG 変数に利用可能なオプション全一覧は、**rsyslogd(8)** man ページの関連セクションを参照してください。

etc/rsyslog.conf ファイルで使用している構文が有効かどうかをチェックするには、以下を使用します。

```
rsyslogd -N 1
```

1 は、出力メッセージの長さのレベルを表します。現在提供されているのは 1 レベルのみなので、これは前方互換性オプションになります。ただし、検証を実行するには、この引数を追加する必要があります。

18.8. Journal の使用

Journal は **systemd** のコンポーネントで、ログファイルの表示および管理を担います。**rsyslogd** などの従来の syslog デーモンとの並列もしくはその代わりとして使用できます。Journal は、従来のロギングに関連する問題を処理するために開発されました。システムの他の部分と緊密に統合されており、さまざまなロギング技術およびログファイルのアクセス管理をサポートします。

ロギングデータは、Journal の **journald** サービスが収集、保存、処理を行います。カーネルやユーザー処理、標準出力、システムサービスの標準エラー出力、またはネイティブ API 経由から受信したロギング情報に基づいて、**journals** と呼ばれるバイナリーファイルを作成、維持します。これらの journals は構造化およびインデックス化され、これによりシーク時間が比較的速くなります。Journal エントリーは、一意の識別子を持つことが可能です。**journald** サービスは、各ログメッセージについて多数のメタデータを収集します。実際の journal ファイルはセキュリティー保護され、手動での編集はできません。

18.8.1. ログファイルの表示

journal ログにアクセスするには、**journalctl** ツールを使用します。ログタイプの基本的なビューを見るには、**root** で以下を入力します。

```
journalctl
```

このコマンドの出力は、システム上で生成されたすべてのログファイル一覧で、これにはシステムコンポーネントやユーザーが生成したメッセージも含まれます。この出力の構造は、**/var/log/messages/** で使われているものに似ていますが、以下の改善点があります。

- エントリーの優先度が視覚的にマークされています。エラー優先度およびそれ以上の行は赤色のハイライト表示がされており、注意および警告の優先度の行には太字フォントが使われています。
- タイムスタンプが使用中のシステムのローカルタイムゾーンに変換されます。
- ローテーションされたログを含めて、すべてのログ記録済みデータが表示されます。
- ブートの最初が特別行にタグ付けされます。

例18.14 journalctl の出力例

以下は、**journalctl** ツールが提供する出力例です。パラメーターなしで呼び出されると、エントリー一覧がタイムスタンプで始まり、その後にホスト名、操作を実行したアプリケーションが示され、実際のメッセージが続きます。この例では、journal ログの初めの 3 つのエントリーを表示しています。

```
# journalctl
-- Logs begin at Thu 2013-08-01 15:42:12 CEST, end at Thu 2013-08-01
15:48:48 CEST. --
Aug 01 15:42:12 localhost systemd-journal[54]: Allowing runtime journal
files to grow to 49.7M.
Aug 01 15:42:12 localhost kernel: Initializing cgroup subsys cpuset
Aug 01 15:42:12 localhost kernel: Initializing cgroup subsys cpu

[...]
```

多くのケースでは、journal ログの最新のエントリーのみに関連性があります。**journalctl** 出力を減らす簡単な方法は、**-n** オプションの使用です。これで、最新のログエントリーは指定された数のみ一覧表示されます。

```
journalctl -n Number
```

Number を表示する行数に置き換えます。数字が指定されない場合は、**journalctl** は最新の 10 エントリーを表示します。

journalctl コマンドを使うと、以下の構文で出力形式を制御することができます。

```
journalctl -o form
```

form を出力形式を指定するキーワードで置き換えます。**verbose** オプションを使うと、全フィールドをともなう完全構造化エントリーアイテムが返されます。**export** を使うと、バックアップおよびネットワーク転送に適したバイナリーストリームを作成します。**json** を使うと、エントリーを JSON データ構造としてフォーマットします。キーワードの全一覧は、**journalctl(1)** man ページを参照してください。

例18.15 詳細な journalctl 出力

全エントリーについての完全なメタデータを表示するには、以下を入力します。

```
# journalctl -o verbose
[...]

Fri 2013-08-02 14:41:22 CEST
[s=e1021ca1b81e4fc688fad6a3ea21d35b;i=55c;b=78c81449c920439da57da7bd5c56
a770;m=27cc
        _BOOT_ID=78c81449c920439da57da7bd5c56a770
        PRIORITY=5
        SYSLOG_FACILITY=3
        _TRANSPORT=syslog
        _MACHINE_ID=69d27b356a94476da859461d3a3bc6fd
        _HOSTNAME=localhost.localdomain
        _PID=562
        _COMM=dbus-daemon
        _EXE=/usr/bin/dbus-daemon
        _CMDLINE=/bin/dbus-daemon --system --address=systemd: --nofork
--nopidfile --systemd-activation
        _SYSTEMD_CGROUP=/system/dbus.service
        _SYSTEMD_UNIT=dbus.service
        SYSLOG_IDENTIFIER=dbus
        SYSLOG_PID=562
        _UID=81
        _GID=81
        _SELINUX_CONTEXT=system_u:system_r:system_dbusd_t:s0-s0:c0.c1023
        MESSAGE=[system] Successfully activated service
'net.reactivated.Fprint'
        _SOURCE_REALTIME_TIMESTAMP=1375447282839181

[...]
```

この例では、単一ログエントリーを識別するフィールドを一覧表示しています。18.8.4項「高度なフィルタリング」の説明にあるように、これらのメタデータはメッセージのフィルタリングに使用できます。全フィールドの説明については、**systemd.journal-fields(7)** man ページを参照してください。

18.8.2. アクセス制御

デフォルトでは、**root** 権限のない **Journal** ユーザーは、自身の生成したログファイルしか見れません。システム管理者が特定のユーザーを **adm** グループに追加すると、これらユーザーはすべてのログファイルにアクセスできるようになります。これを実行するには、**root** で以下を入力します。

```
usermod -a -G adm username
```

username を **adm** グループに追加するユーザー名で置き換えます。すると、このユーザーは、root ユーザーと同様の **journalctl** コマンド出力を受け取るようになります。アクセス制御が機能するのは、**Journal** の永続ストレージが有効になっている場合のみであることに注意してください。

18.8.3. ライブビューの使用

journalctl はパラメーターなしで呼び出されると、収集されたうちで最も古いものから順にエントリーすべてを一覧表示します。ライブビューでは、新たなエントリーが現れるとこれが継続的にプリントされるので、リアルタイムでログメッセージを監視することができます。**journalctl** をライブビューモードで開始するには、以下を入力します。

```
journalctl -f
```

このコマンドは、最新のログ 10 行を返します。その後は **journalctl** ユーティリティーの実行が継続され、新たな変化があるとそれを直ちに表示します。

18.8.4. メッセージのフィルタリング

パラメーターなしで実行された **journalctl** コマンドの出力は大規模なものになることが多いので、さまざまなフィルタリング方法を使うとユーザーのニーズに合った情報を抽出することができます。

優先度によるフィルタリング

ログメッセージは、システム上の間違った動作を追跡するために使用されることがよくあります。特定のエントリーまたはより高い優先度のエントリーのみを表示するには、以下の構文を使用します。

```
journalctl -p priority
```

priority を **debug** (7)、**info** (6)、**notice** (5)、**warning** (4)、**err** (3)、**crit** (2)、**alert** (1)、または **emerg** (0) のいずれかもしくはそれらの数字に置き換えます。

> **例18.16 優先度によるフィルタリング**
>
> **error** もしくはそれ以上の優先度のエントリーのみを表示するには、以下を使用します。
>
> ```
> journalctl -p err
> ```

時間によるフィルタリング

現在のブートからのログエントリーのみを表示するには、以下を入力します。

```
journalctl -b
```

システム再起動をたまにしか行わない場合は、**-b** は **journalctl** の出力を大幅に削減しません。この場合、時間ベースのフィルタリングが便利になります。

```
journalctl --since=value --until=value
```

--since および **--until** を使うと、特定の時間枠内に作成されたログメッセージのみを表示できます。これらのオプションに渡す *values* は、以下の例で示すように日付か時間、もしくはそれら両方の形式を取ることができます。

> **例18.17 時間および優先度によるフィルタリング**
>
> フィルタリングオプションは組み合わせることで、特定のリクエストによって結果を絞ることができます。たとえば、**warning** またはそれ以上の優先度で特定の時刻以降のメッセージのみを表示するには、以下を入力します。

```
journalctl -p warning --since="2013-3-16 23:59:59"
```

高度なフィルタリング

例18.15「詳細な journalctl 出力」では、ログエントリーを特定し、フィルタリングに使用できるフィールドを一覧表示しています。**systemd** が保存可能なメタデータのすべての説明については、**systemd.journal-fields(7)** man ページを参照してください。このメタデータは、ユーザーが介入することなく、各ログメッセージについて収集されます。値は通常テキストベースですが、バイナリーの大きな値になることもあります。フィールドには複数の値がある場合もありますが、一般的ではありません。

指定されたフィールドで発生する一意の値を一覧表示するには、以下の構文を使用します。

```
journalctl -F fieldname
```

fieldname を関心のあるフィールド名に置き換えます。

特定条件のみに合致するログエントリーのみを表示するには、以下の構文を使用します。

```
journalctl fieldname=value
```

fieldname をフィールド名に、*value* をそのフィールドに含まれる特定の値に置き換えます。そうすると、この条件に合致する行のみが返されます。

注記

systemd に保存されているメタデータフィールドはかなりの数になるので、関心のあるフィールド名そのものを忘れることがよくあります。名前が不確かな場合は、以下を入力します。

```
journalctl
```

そして、**Tab** キーを 2 回押します。これで、利用可能なフィールド名が一覧表示されます。コンテキストベースの **Tab** 補完入力はフィールド名で機能するので、フィールド名の明確な文字を入力して **Tab** を押すと、名前が自動的に完了します。同様に、フィールドから一意の値を一覧表示することもできます。以下を入力します。

```
journalctl fieldname=
```

そして **Tab** を 2 回押します。これは、**journalctl -F** *fieldname* の代わりとなります。

1 つのフィールドに複数の値を指定することもできます。

```
journalctl fieldname=value1 fieldname=value2 ...
```

同一フィールドで 2 つの値を指定すると、**logical OR** 組み合わせに一致するものが表示されます。*value1* または *value2* に合致するエントリーが表示されます。

複数のフィールドと値の組み合わせを指定して、出力をさらにしぼり込むこともできます。

```
journalctl fieldname1=value fieldname2=value ...
```

異なる 2 つのフィールド名での一致を指定すると、**logical AND** と組み合わされることになります。エントリーが表示されるには、両方の条件を満たす必要があります。

+ 記号を使うと、複数フィールドで **logical OR** 組み合わせの一致が設定できます。

```
journalctl fieldname1=value + fieldname2=value ...
```

このコマンドは、両方の条件に合致するエントリーだけでなく、少なくとも条件の 1 つに合致するエントリーを返します。

例18.18 高度なフィルタリング

UID 70 のユーザーが **avahi-daemon.service** または **crond.service** で作成したエントリーを表示するには、以下のコマンドを使用します。

```
journalctl _UID=70 _SYSTEMD_UNIT=avahi-daemon.service
_SYSTEMD_UNIT=crond.service
```

_SYSTEMD_UNIT フィールドに 2 つの値があるため、両方の結果が表示されますが、これは **_UID=70** の条件に合致する場合のみです。これは単に (UID=70 and (avahi or cron)) と表すこともできます。

上記のフィルタリングをライブビューモードで適用して、特定のログエントリーグループでの最新の変更を追跡することもできます。

```
journalctl -f fieldname=value ...
```

18.8.5. 永続的ストレージの有効化

Journal はデフォルトでは、ログファイルをメモリーか **/run/log/journal/** ディレクトリー内の小さいリングバッファーにのみ保存します。これは、**journalctl** の最近のログ履歴を表示するには十分なものです。このディレクトリーは揮発性なので、ログデータは永続的には保存されません。デフォルト設定では、syslog は journal ログを読み取り、**/var/log/** ディレクトリーに保存します。永続的なロギングが有効になると、journal ファイル **/var/log/journal** に保存され、再起動後も維持されます。するとユーザーによっては、Journal が **rsyslog** の代わりとなることも可能です (ただし、本章の序論を参照のこと)。

永続的ストレージを有効にすると、以下の利点があります。

- 長期的にトラブルシュートのためのより豊富なデータが記録されます。
- 直ちにトラブルシュートを行う場合には、再起動後により多くのデータが利用可能になります。
- サーバーコンソールがログファイルからではなく、journal からデータを読み取ります。

永続的なストレージには、不利な点もあります。

- 永続的なストレージを使っても、保存されるデータ量は空きのあるメモリーに依存することから、特定の期間をカバーする保証はありません。
- ログにより多くのディスクスペースが必要になります。

Journal 用に永続的なストレージを有効にするには、以下の例のように手動で journal ディレクトリーを作成します。**root** で以下を入力します。

```
mkdir -p /var/log/journal
```

その後に **journald** を再起動して、変更を適用します。

```
systemctl restart systemd-journald
```

18.9. グラフィカル環境でのログファイルの管理

18.9.1. ログファイルの表示

ほとんどのログファイルはプレーンなテキスト形式で保存され、**Vi** や **Emacs** などのテキストエディターで表示することができます。一部のログファイルはシステム上の全ユーザーが読み取り可能ですが、ほとんどのログファイルは読み取りに root 権限が必要になります。

インタラクティブなリアルタイムアプリケーションでシステムのログファイルを表示するには、**Log File Viewer (ログビューア)** を使用します。

注記

Log File Viewer を使用するには、まず **root** で以下を実行して *gnome-system-log* パッケージがインストールされていることを確認します。

```
~]# yum install gnome-system-log
```

yum を使ったパッケージのインストールについては、「パッケージのインストール」を参照してください。

gnome-system-log パッケージをインストールしたら、**アプリケーション** → **システムツール** → **ログビューア** の順にクリックするか、シェルプロンプトで以下のコマンドを入力して、**ログビューア** を開きます。

```
~]$ gnome-system-log
```

このアプリケーションは、存在するログファイルのみを表示します。そのため、図18.2「ログビューア」で表示されている一覧とは異なる場合があります。

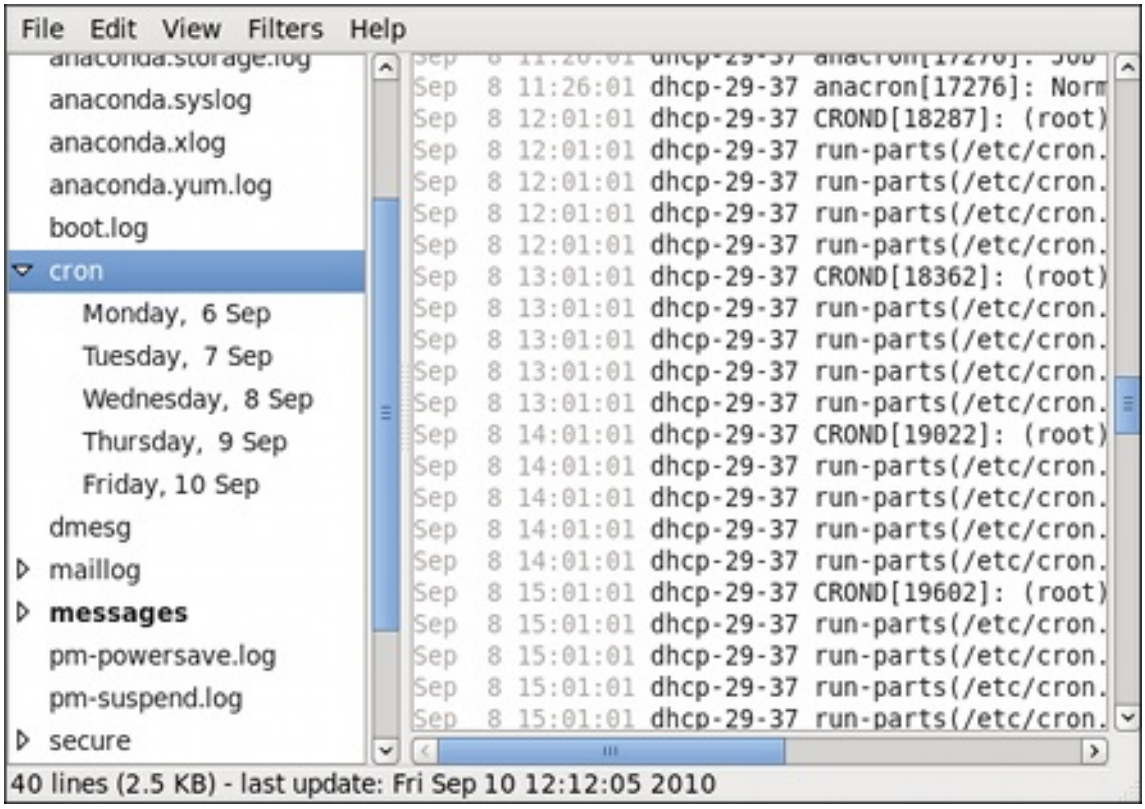


図18.2 ログビューア

Log File Viewer アプリケーションを使うと、既存のログファイルのフィルタリングが可能になります。メニューから **フィルタ** をクリックし、**フィルタを管理** を選択して目的のフィルターを定義、編集します。

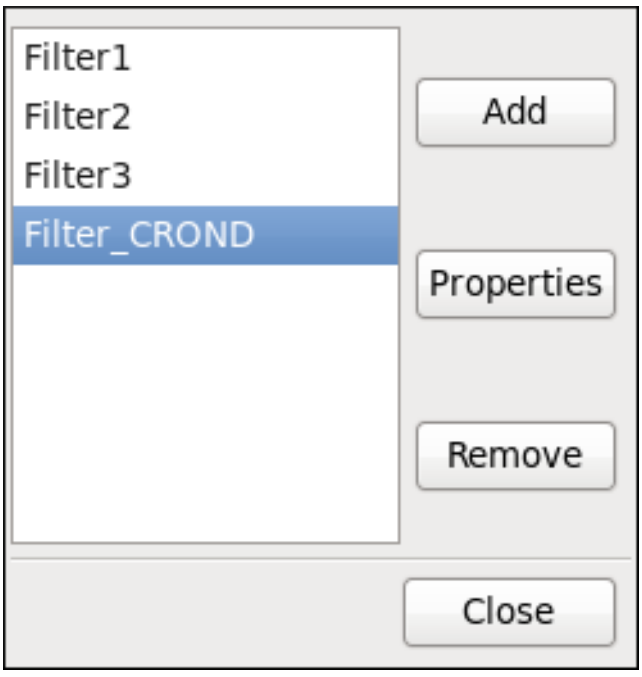


図18.3 ログビューア - フィルター

フィルターを追加/編集することで、図18.4「ログビューア - フィルターの定義」のようにパラメーターを定義することができます。

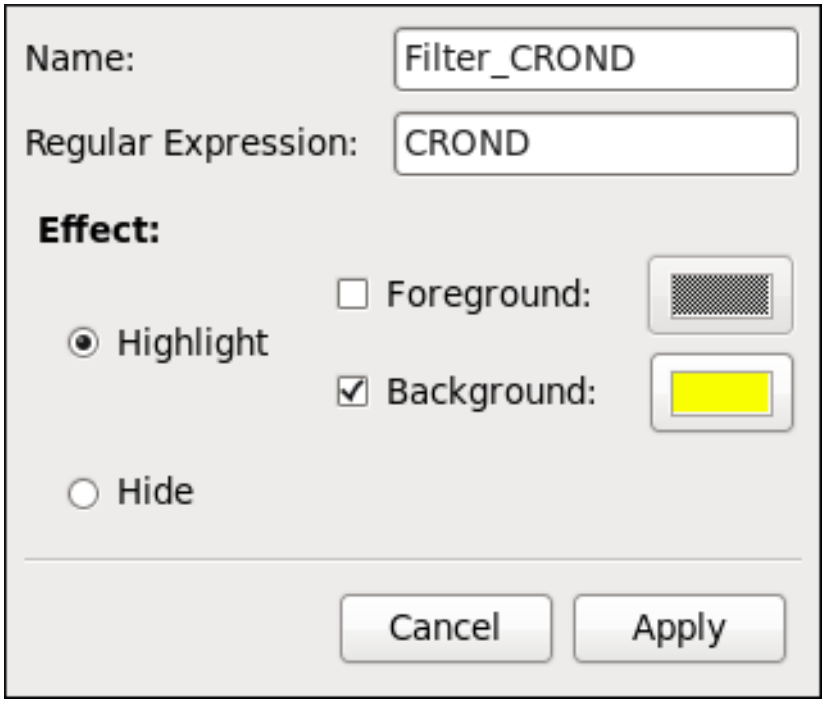


図18.4 ログビューア - フィルターの定義

フィルターを定義する際には、以下のパラメーターを編集できます。

- **名前** — フィルターの名前を指定します。
- **正規表現** — ログファイルに適用され、その中の実行可能なテキストの文字列に一致するよう試行する正規表現を指定します。
- **効果**
  - **強調** — これが有効な場合は、検索結果は選択した色で強調されます。強調させる場所をテキストの背面/前面から選択できます。
  - **非表示** — これが有効な場合は、検索結果は閲覧中のログファイルからは非表示になります。

少なくとも 1 つのフィルターが定義されていれば、**フィルター** メニューからそれを選択することができ、そのフィルターが定義済みの文字列を自動的に検索して、現在表示しているログファイル内でのマッチを強調表示または非表示にします。

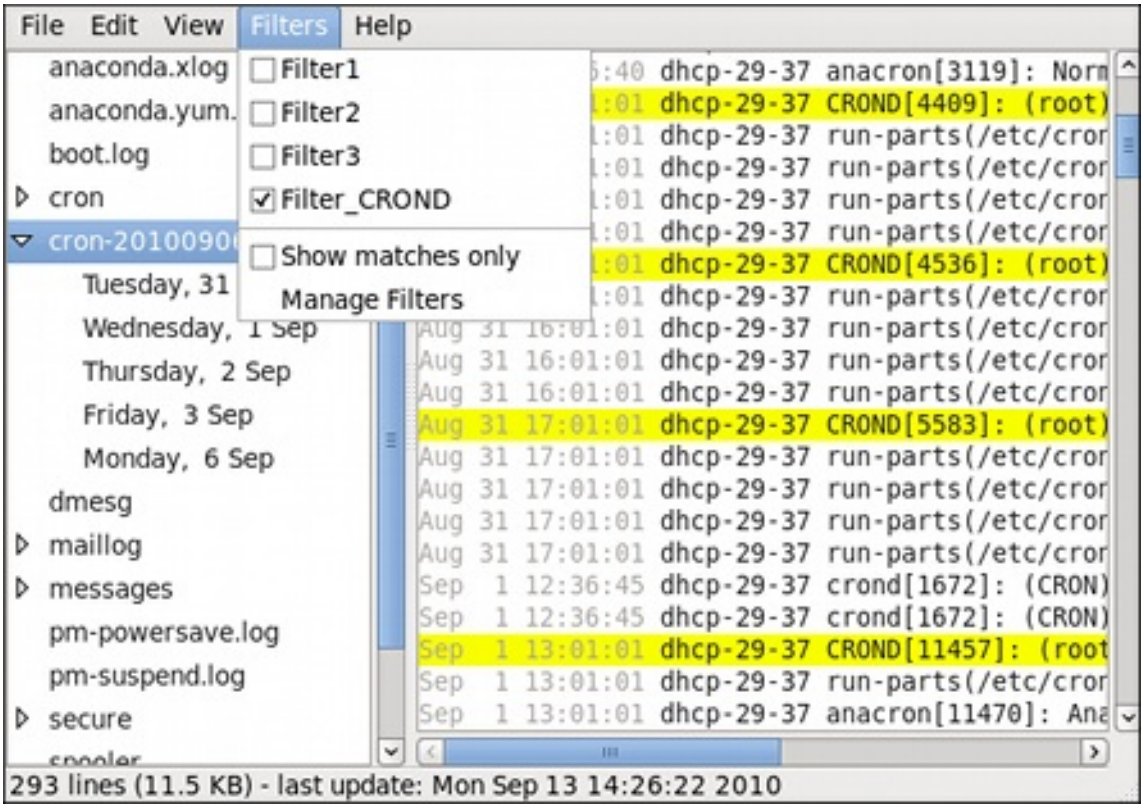


図18.5 ログビューア - フィルターの有効化

マッチしたもののみ表示 オプションを選択すると、現在表示されているログファイル内でマッチした文字列のみが表示されます。

18.9.2. ログファイルの追加

一覧で閲覧したいログファイルを追加するには、**ファイル** → **開く** の順に選択します。その後、閲覧したいログファイルのディレクトリとファイル名を選択できる **ファイルの選択** ウィンドウが表示されます。図18.6「ログビューア - ログファイルの追加」 は、**ファイルの選択** ウィンドウを示しています。

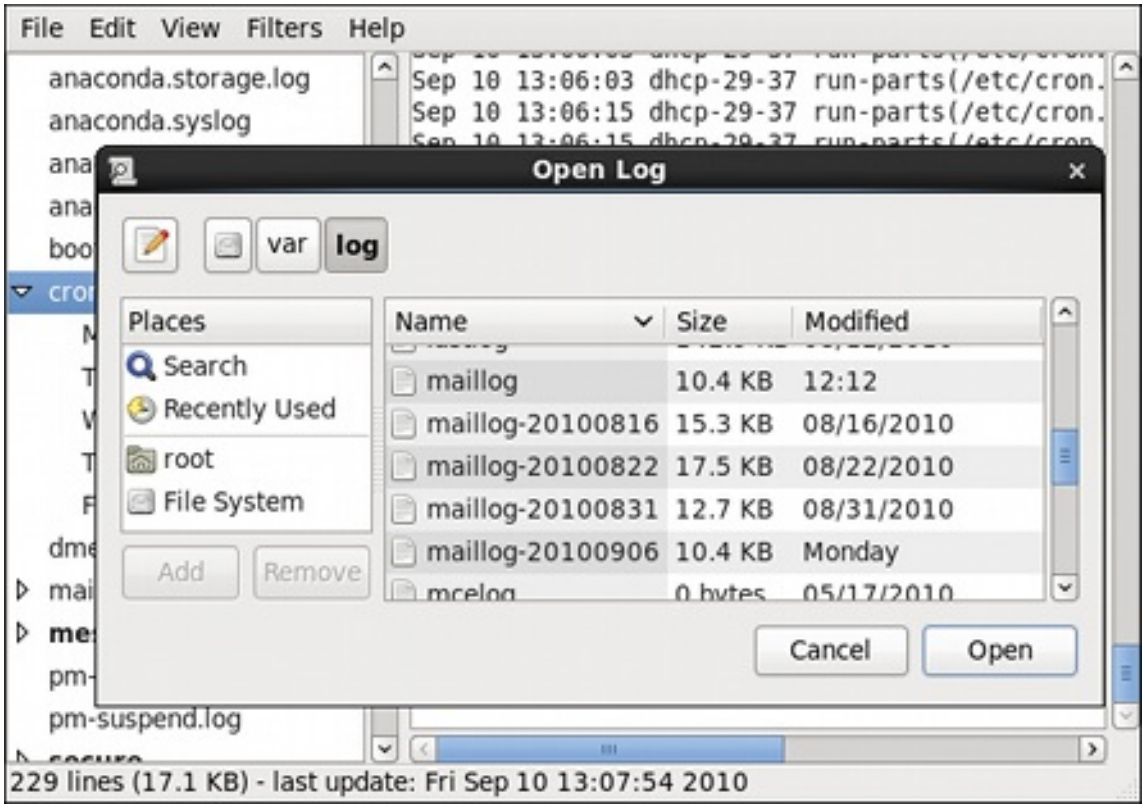


図18.6 ログビューア - ログファイルの追加

ファイルを開くには、**開く** ボタンをクリックします。ファイルは表示中の一覧に直ちに追加されるため、そのファイルを選択してコンテンツを表示することができます。

注記

ログビューア を使用すると **.gz** 形式で圧縮されたログファイルを開くこともできます。

18.9.3. ログファイルのモニタリング

ログビューア は、デフォルトで開いているすべてのログをモニターします。モニターされているログファイルに新しい行が追加されると、そのログ名はログの一覧に太字で表示されます。ログファイルが選択/表示されると、ログファイルの最下部に新しい行が太字で表示されます。図18.7「ログビューア - 新しいログ通知」 は、**cron** ログファイルおよび **messages** ログファイル内の新しい通知を示しています。**cron** ログファイルをクリックすると、ファイル内のログが太字で新しい行に表示されます。

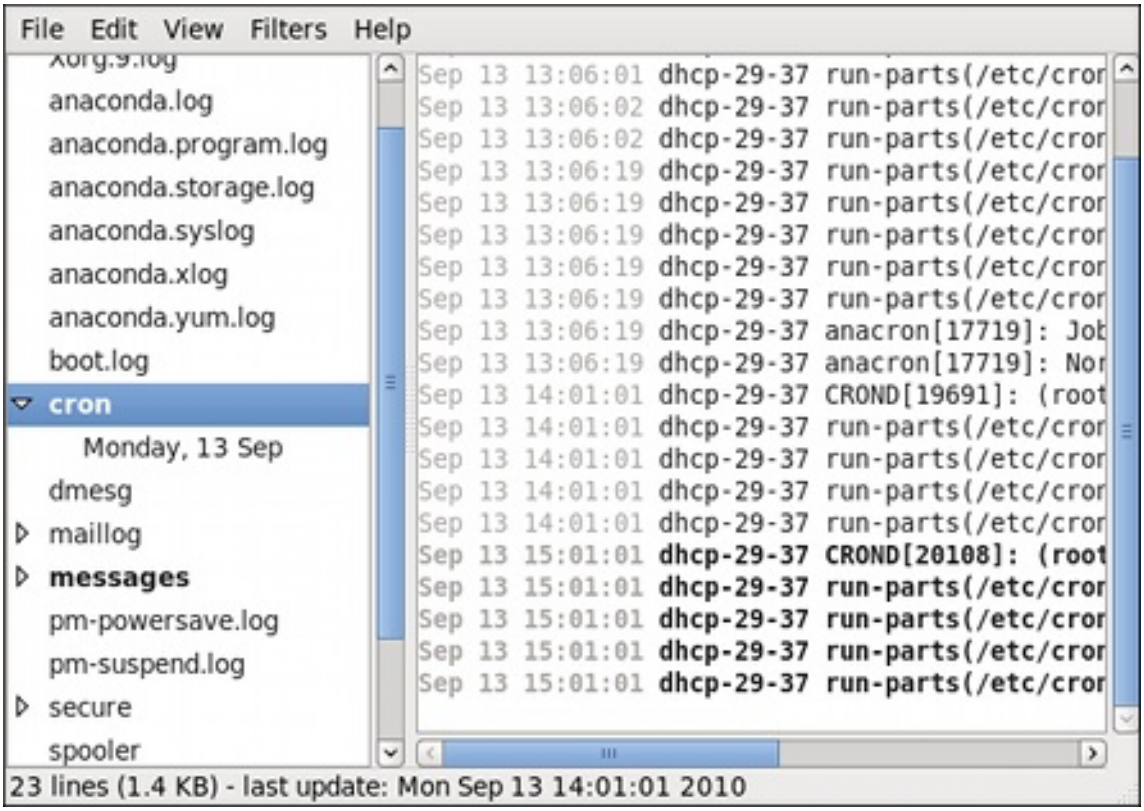


図18.7 ログビューア - 新しいログ通知

18.10. その他のリソース

rsyslog デーモンの設定方法、およびログファイルの場所の特定、表示、モニタリング方法に関する詳細情報は、以下のリソースを参照してください。

インストールされているドキュメント

- **rsyslogd**(8) — **rsyslogd** デーモンの man ページは、その使用方法を説明しています。
- **rsyslog.conf**(5) — **rsyslog.conf** の man ページは、利用可能な設定オプションを説明しています。
- **logrotate**(8) — **logrotate** ユーティリティーの man ページは、その設定方法と使用方法を詳細に説明しています。
- **journalctl**(1) — **journalctl** デーモンの man ページは、その使用方法を説明しています。
- **journald.conf**(5) — この man ページは、利用可能な設定オプションを説明しています。
- **systemd.journal-fields**(7) — この man ページは、特別な **Journal** フィールドを一覧表示しています。

オンラインのドキュメント

- rsyslog Home Page — **rsyslog** のホームページでは、機能の完全な技術的分析、ドキュメンテーション、設定サンプル、およびビデオチュートリアルを提供しています。
- **RainerScript** documentation on the rsyslog Home Page — **RainerScript** で利用可能なデータタイプ、式、関数についての概要です。

- Description of **queues** on the rsyslog Home Page — さまざまなタイプのメッセージキューおよびその使用方法に関する全般情報です。
- rsyslog Wiki — 『The rsyslog Wiki』 には、便利な設定例があります。

関連項目

- 4章*権限の取得* では、**su** および **sudo** コマンドを使って管理者権限を取得する方法が説明されています。
- 6章*systemd によるサービス管理*では systemd に関する詳細情報と、**systemctl** コマンドを使ってシステムサービスを管理する方法が説明されています。

第19章 システムタスクの自動化

タスクは *ジョブ* とも呼ばれ、指定した期日に指定した期間だけ、またはシステムのロード平均が 0.8 を下回る時に、自動的に実行するように設定できます。

Red Hat Enterprise Linux は、システムを最新状態に保つために重要なシステムタスクを実行するように事前設定されています。例えば、**locate** コマンドで使用されるslocate データベースは毎日更新されます。システム管理者は、この自動化タスクを使用して定期的なバックアップ、システムの監視、カスタムスクリプトの実行などの作業を行うことができます。

Red Hat Enterprise Linux は、**cron**、**anacron**、**at**、 **batch** といった自動化タスクユーティリティーを備えています。

ユーティリティーはすべて、異なるジョブタイプのスケジュール管理を目的としています。Cron と Anacron は繰り返されるジョブのスケジュール管理をするのに対し、At と Batch は 1 回のみのジョブのスケジュール管理をします(「Cron と Anacron」 と 「At コマンドと Batch コマンド」 をそれぞれ参照してください)。

Red Hat Enterprise Linux 7 は、特定の時間にジョブを実行するための **systemd.timer** をサポートしています。詳細情報は、**systemd.timer(5)** の man ページを参照してください。

19.1. Cron と Anacron

Cron と Anacron はいずれも、時刻、日付、月、曜日、週で定義されたある時点で定期的なタスクの実行をスケジュールすることができるデーモンです。

Cron は、最高で 1 分おきの頻度でジョブの実行が可能です。しかし、このユーティリティーはシステムが継続的に稼働状態であることを前提としており、ジョブがスケジュールされている時にシステムが稼働していない場合は、ジョブは実行されません。

Anacron は、ジョブがスケジュールされている時点でシステムが稼働していなくても、ジョブを記憶しています。そして、次回にシステムが立ち上がった時にジョブを実行します。しかし、Anacron が実行可能なジョブは 1 日に 1 回のみです。

19.1.1. Cron と Anacron のインストール

Cron と Anacron をインストールするには、Cron は *cronie* パッケージを、Anacron は*cronie-anacron* パッケージを (*cronie-anacron* は *cronie* のサブパッケージ) をインストールする必要があります。

システムにすでにこれらのパッケージがインストール済みかどうかを確認するには、以下のコマンドを発行します。

```
rpm -q cronie cronie-anacron
```

インストール済みの場合は *cronie* および *cronie-anacron* パッケージの完全名が返され、そうでない場合はパッケージが利用可能でないことが通知されます。

これらのパッケージをインストールするには、**root** で **yum** コマンドを以下の形式で使用します。

```
yum install package
```

たとえば、Cron と Anacron の両方をインストールするには、シェルプロンプトで以下を入力します。

```
~]# yum install cronie cronie-anacron
```

Red Hat Enterprise Linux での新しいパッケージのインストール方法の詳細は、「パッケージのインストール」を参照してください。

19.1.2. Crond サービスの実行

cron ジョブと anacron ジョブは、どちらも **crond** サービスが選びます。このセクションでは、**crond** サービスの起動、停止、再起動方法と、このサービスがブート時に自動的に起動するように設定する方法を示します。Red Hat Enterprise Linux 7 でのシステムサービスの管理方法全般についての詳細情報は、*6章systemd によるサービス管理*を参照してください。

19.1.2.1. Cron サービスの起動と停止

サービスが実行中かどうかを確認するには、以下のコマンドを使用します。

```
systemctl status crond.service
```

現行のセッションで **crond** サービスを実行するには、シェルプロンプトで**root** として以下を入力します。

```
systemctl start crond.service
```

サービスがブート時に自動的に開始するように設定するには、**root** で以下のコマンドを実行します。

```
systemctl enable crond.service
```

19.1.2.2. Cron サービスの停止

現行のセッションで **crond** サービスを停止するには、**root** でシェルプロンプトに以下を入力します。

```
systemctl stop crond.service
```

サービスがブート時に自動的に開始しないように設定するには、**root** で以下のコマンドを実行します。

```
systemctl disable crond.service
```

19.1.2.3. Cron サービスの再起動

crond サービスを再起動するには、**root** でシェルプロンプトに以下を入力します。

```
systemctl restart crond.service
```

このコマンドですばやくサービスを停止して、再度開始します。

19.1.3. Anacron ジョブの設定

ジョブをスケジュールするための主要設定ファイルは **/etc/anacrontab** ファイルで、これにアクセスできるのは **root** ユーザーのみです。このファイルには、以下のような行が記載されています。

```
SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
# the maximal random delay added to the base delay of the jobs
```

```
RANDOM_DELAY=45
# the jobs will be started during the following hours only
START_HOURS_RANGE=3-22

#period in days   delay in minutes   job-identifier   command
1         5     cron.daily    nice run-parts /etc/cron.daily
7         25    cron.weekly   nice run-parts /etc/cron.weekly
@monthly  45    cron.monthly  nice run-parts /etc/cron.monthly
```

最初の 3 行は、anacron タスクが実行される環境を設定するための変数を定義します。

- **SHELL** — ジョブの実行に使用されるシェル環境です (例えば、Bash シェル)。
- **PATH** — 実行可能なプログラムへのパスです。
- **MAILTO** — anacron ジョブの出力を電子メールで受け取るユーザーのユーザー名です。

  MAILTO 変数が定義されていない場合、(**MAILTO=**)、電子メールは送信されません。

以下の 2 つの変数は、定義済みジョブのスケジュール時間を修正します。

- **RANDOM_DELAY** — ジョブごとに指定されている **delay in minutes** 変数に追加される最大の時間分数。

  最短の遅延時間は、デフォルトで 6 分に設定されています。

  たとえば、**RANDOM_DELAY** が **12** に設定されていたとすると、その特定の anacrontab における各ジョブの **delay in minutes** に 6 分から 12 分が追加されます。**RANDOM_DELAY** は **6** 未満の設定も可能で、**0** の設定もできます。**0** に設定されると、ランダムな遅延は追加されません。これは例えば、1 つのネットワーク接続を共有する多くのコンピューターが毎日同じデータをダウンロードする必要がある場合などに有用です。

- **START_HOURS_RANGE** — スケジュールされているジョブが実行可能になる時間単位の間隔です。

  停電などで時間間隔を逸してしまった場合は、その日にスケジュールされていたジョブは実行されません。

/etc/anacrontab ファイル内の残りの行は、スケジュールされているジョブを表しており、以下の形式を取ります。

```
period in days   delay in minutes   job-identifier   command
```

- **period in days** — ジョブの実行頻度の日数。

  属性値は整数もしくはマクロ (**@daily**、**@weekly**、**@monthly**) として定義可能です。**@daily** は整数の 1 を、**@weekly** は整数の 7 を表し、**@monthly** は、月の長さに関係なく毎月 1 回ジョブが実行されることを指定します。

- **delay in minutes** — ジョブ実行までに anacron が待機する分数です。

  属性値は整数として定義されます。値が **0** に設定されると、待機は適用されません。

- **job-identifier** — ログファイルで使用される特定のジョブを指す一意の名前です。
- **command** — 実行するコマンドです。

  このコマンドは、**ls /proc >> /tmp/proc** などのコマンドやカスタムスクリプトを実行するコマンドにすることができます。

先頭にハッシュ記号 (#) が付いた行はコメント行で、処理の対象外です。

19.1.3.1. Anacron ジョブの例

以下の例は、簡単な **/etc/anacrontab** ファイルを示しています。

```
SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

# the maximal random delay added to the base delay of the jobs
RANDOM_DELAY=30
# the jobs will be started during the following hours only
START_HOURS_RANGE=16-20

#period in days   delay in minutes   job-identifier   command
1         20    dailyjob      nice run-parts /etc/cron.daily
7         25    weeklyjob     /etc/weeklyjob.bash
@monthly  45    monthlyjob    ls /proc >> /tmp/proc
```

この **anacrontab** ファイル内で定義されているジョブはすべて、ランダムに 6 分から 30 分の間で遅延することになり、16:00 時から 20:00 時の間に実行できます。

最初に定義されているジョブは、毎日、16:26 から 16:50 の間に実行されます (RANDOM_DELAY が 6 分から 30 分の間で、delay in minutes 属性がこれに 20 分追加します) 。このジョブに指定されているコマンドは、**run-parts** スクリプトを使って **/etc/cron.daily/** ディレクトリー内にあるすべての現行プログラムを実行します (**run-parts** スクリプトがディレクトリーをコマンドラインの引数として受け取り、ディレクトリー内のプログラムを順にすべて実行していきます)。**run-parts** スクリプトに関する詳細は、**run-parts** を参照してください。

2 つ目のジョブは、**/etc/** ディレクトリーの **weeklyjob.bash** スクリプトを週 1 回実行します。

3 つ目のジョブは、**/proc** のコンテンツを月 1 回 **/tmp/proc** ファイル (**ls /proc >> /tmp/proc**) に書き込むコマンドを実行します。

19.1.4. Cron ジョブの設定

cron ジョブの設定ファイル **/etc/crontab** (**root** ユーザーのみ修正可能) には、以下の行が記載されています。

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/
# For details see man 4 crontabs
# Example of job definition:
# .---------------- minute (0 - 59)
# | .------------- hour (0 - 23)
# | | .---------- day of month (1 - 31)
# | | | .------- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | .---- day of week (0 - 6) (Sunday=0 or 7) OR
sun,mon,tue,wed,thu,fri,sat
# | | | | |
# * * * * * username  command to be executed
```

最初の 3 行には、**anacrontab** ファイルと同様に、**SHELL** と **PATH**、**MAILTO** の変数定義が記載されています。これらの変数に関する詳細は、「Anacron ジョブの設定」を参照してください。

さらにこのファイルは、**HOME** 変数を定義できます。**HOME** 変数は、ジョブが行うコマンドやスクリプトの実行時にホームディレクトリーとして使用されるディレクトリーを定義します。

/etc/crontab ファイルの残りの行はスケジュールされているジョブを示しており、以下の形式を取ります。

```
minute   hour   day   month   day of week   username   command
```

以下は、ジョブの実行時間を定義します。

- **minute** — 0 から 59 までの任意の整数
- **hour** — 0 から 23 までの任意の整数
- **day** — 1 から 31 までの任意の整数 (月が指定されている場合は、その月の有効な日)
- **month** — 1 から 12 までの任意の整数 (あるいは jan や feb など月の省略形)
- **day of week** — 0 から 7 までの任意の整数。0 および 7 は日曜日を示す (あるいは sun や mon など曜日の省略形)

以下は、他のジョブプロパティーを定義します。

- **username** — ジョブが実行されるユーザーを指定します。
- **command** — 実行するコマンドです。

  コマンドは、**ls /proc /tmp/proc** などのコマンド、もしくはカスタムスクリプトを実行するコマンドを指定可能です。

上記のいずれかの値にアスタリスク (*) を使用すると、すべての有効な値を指定することができます。例えば、月の値をアスタリスクで定義すると、ジョブは他の値による制約の範囲内で毎月実行されます。

整数の間のハイフン (-) は、整数の範囲を指定します。例えば、**1-4** は整数 1、2、3、4 を意味します。

コンマ (,) で区切られた値の一覧は、一覧を指定します。例えば、**3,4,6,8** はそれらの 4 つの整数を示します。

スラッシュ (/) を使用すると、ステップ値を指定することができます。範囲の後に ***/integer*** を付けると、その範囲内で、指定した整数の値をスキップします。例えば、**0-59/2** とすると、分フィールドで 1 分おきの間隔を定義することができます。ステップ値は、アスタリスクと併用することも可能です。例えば、月フィールドで値 ***/3** を指定すると、3 カ月ごとにタスクが実行されます。

先頭にハッシュ記号 (#) が付いた行はコメント行で、処理の対象外です。

root 以外のユーザーは、**crontab** ユーティリティーを使用して cron タスクを設定できます。ユーザー定義の crontab は **/var/spool/cron/** ディレクトリーに保存され、それを作成したユーザーが実行するかのように実行されます。

特定のユーザーとして crontab を作成するには、そのユーザーとしてログインし、コマンド **crontab -e** を入力してから、**VISUAL** または**EDITOR** 環境変数で指定されているエディターを使用してユーザーの crontab を編集します。このファイルは **/etc/crontab** と同じ形式を使用します。crontab への変更を保存すると、crontab はユーザー名に応じて保存され、ファイル **/var/spool/cron/*username*** に書き込まれます。現行ユーザーの crontab ファイルのコンテンツを一覧表示するには、**crontab -l** コマンドを使用します。

/etc/cron.d/ ディレクトリーには **/etc/crontab** ファイルと同じ構文を持つファイルが収納されています。このディレクトリー内でファイルの作成と変更ができるのは、**root** のみです。

注記

cron デーモンは、変更を確認するために **/etc/anacrontab** ファイル、**/etc/crontab** ファイル、**/etc/cron.d/** ディレクトリー、**/var/spool/cron/** ディレクトリーを毎分チェックします。何らかの変更が検出されると、メモリーに読み込まれます。そのため anacrontab または crontab のファイルが変更されても、デーモンを再起動する必要はありません。

19.1.5. Cron へのアクセスの制御

Cron へのアクセスを制御するには、**/etc/cron.allow** ファイルと **/etc/cron.deny** ファイルを使用します。これらアクセス制御ファイルは両方とも、各行にユーザー名を 1 つ配置するという同一形式を使用します。どちらのファイルでも空白は許可されません。

cron.allow ファイルが存在する場合は、そのファイルに記載されているユーザーのみが cron の使用を許可され、**cron.deny** ファイルは無視されます。

cron.allow が存在しない場合は、**cron.deny** に表示されているユーザーは cron を使用できません。

アクセス制御ファイルが変更された場合でも、cron デーモン (**crond**) を再起動する必要はありません。アクセス制御ファイルは、ユーザーが cron ジョブの追加または削除を試みるたびにチェックされます。

アクセス制御ファイルにリストされているユーザー名に関係なく、**root** ユーザーはいつでも cron を使用できます。

アクセスは、PAM (Pluggable Authentication Modules) を介して制御することもできます。これらの設定値は **/etc/security/access.conf** に格納されています。例えば、このファイルに以下の行を追記すると、**root** ユーザー以外のユーザーは crontab の作成ができなくなります。

```
-:ALL EXCEPT root :cron
```

禁止されているジョブは、適切なログファイルにログ記録されます。また、**crontab -e** を使用している時は、標準出力に返されます。詳細は、**access.conf.5** を参照してください (**man 5 access.conf** のコマンドで表示されます)。

19.1.6. Cron ジョブのブラックリストとホワイトリスト

ジョブのブラックリストとホワイトリストは、ジョブの実行不要部分を定義するために使用します。**/etc/cron.daily/** などの cron ディレクトリー上の**run-parts** スクリプトを呼び出す際にこれが有用となります。ユーザーが、ディレクトリー内にあるプログラムをジョブのブラックリストに加えると、**run-parts** スクリプトはこのプログラムを実行しません。

ブラックリストを定義するには、ディレクトリー内で **run-parts** スクリプトの実行元となる **jobs.deny** ファイルを作成します。例えば、**/etc/cron.daily/** から特定のプログラムを省略する必要がある場合には、**/etc/cron.daily/jobs.deny** ファイルを作成します。このファイル内で、実行を省略するプログラムの名前を指定します（同一ディレクトリー内にあるプログラムのみ可能）。**run-parts /etc/cron.daily** のように、**/etc/cron.daily/** ディレクトリーからプログラムを実行するコマンドをジョブが実行する場合は、**jobs.deny** ファイル内で定義されたプログラムは実行されません。

ホワイトリストを定義するには、**jobs.allow** ファイルを作成します。

jobs.deny と **jobs.allow** の原則は、「Cron へのアクセスの制御」セクションの説明にある **cron.deny** と **cron.allow** の原則と同じです。

19.2. At コマンドと Batch コマンド

cron は定期的なタスクのスケジュールに使うのに対して、**At** ユーティリティーは特定の時刻での 1 回限りのタスクのスケジュールに使われます。また、**Batch** ユーティリティーは、システム負荷平均が 0.8 を下回った時に実行される 1 回限りのタスクのスケジュールに使用します。

19.2.1. At と Batch のインストール

システムにすでに *at* パッケージがインストール済みかどうかを確認するには、以下のコマンドを発行します。

```
rpm -q at
```

インストール済みの場合は *at* パッケージの完全名が返され、そうでない場合はパッケージが利用可能でないことが通知されます。

このパッケージをインストールするには、**root** で **yum** コマンドを以下の形式で使用します。

```
yum install package
```

たとえば、At と Batch の両方をインストールするには、シェルプロンプトで以下を入力します。

```
~]# yum install at
```

Red Hat Enterprise Linux での新しいパッケージのインストール方法の詳細は、「パッケージのインストール」を参照してください。

19.2.2. At サービスの実行

At ジョブと Batch ジョブは、どちらも **atd** サービスが選びます。このセクションでは、**atd** サービスの起動、停止、再起動方法と、このサービスがブート時に自動的に起動するように設定する方法を示します。Red Hat Enterprise Linux 7 でのシステムサービスの管理方法全般についての詳細情報は、*6章systemd によるサービス管理* を参照してください。

19.2.2.1. At サービスの起動と停止

サービスが実行中かどうかを確認するには、以下のコマンドを使用します。

```
systemctl status atd.service
```

現行のセッションで **atd** サービスを実行するには、シェルプロンプトで**root** として以下を入力します。

```
systemctl start atd.service
```

サービスがブート時に自動的に開始するように設定するには、**root** で以下のコマンドを実行します。

```
systemctl enable atd.service
```

注記

ブート時に **atd** サービスが自動的に起動するようにシステムを設定することが推奨されます。

19.2.2.2. At サービスの停止

atd サービスを停止するには、**root** でシェルプロンプトに以下を入力します。

```
systemctl stop atd.service
```

サービスがブート時に自動的に開始しないように設定するには、**root** で以下のコマンドを実行します。

```
systemctl disable atd.service
```

19.2.2.3. At サービスの再起動

atd サービスを再起動するには、**root** でシェルプロンプトに以下を入力します。

```
systemctl restart atd.service
```

このコマンドですばやくサービスを停止して、再度開始します。

19.2.3. At ジョブの設定

At ユーティリティーを使って特定の時刻に 1 回限りのジョブをスケジュールするには、以下を実行します。

1. コマンドラインで、コマンド **at *TIME*** を入力します。ここでの***TIME*** は、コマンドを実行する時刻です。

   TIME 引数は以下のいずれかの形式で定義できます。

   - ***HH*:*MM*** は、特定の時間と分数を指定します。例えば、**04:00** は 4:00 a.m. を指定します。
   - **midnight** は、12:00 a.m. を指定します。
   - **noon** は、12:00 p.m. を指定します。
   - **teatime** は、4:00 p.m. を指定します。
   - ***MONTHDAYYEAR*** 形式ではたとえば、**January 15 2012** は 2002 年 1 月 15 日を指定します。年表示はオプションです。
   - ***MMDDYY***、***MM*/*DD*/*YY***、***MM*.*DD*.*YY*** のいずれかの形式ではたとえば**011512** は、2012 年 1 月 15 日となります。
   - **now + *TIME*** では、*TIME* は、整数と分、時間、日、週の値として定義されます。たとえば、**now + 5 days** は、コマンドが 5 日後の現在と同じ時刻に実行されることを指定します。

     時間は、オプションの日付の前に指定する必要があります。時間の形式に関する詳細は、**/usr/share/doc/at-*<version>*/timespec** のテキストファイルを参照してください。

もし指定の時刻が過ぎてしまった場合には、翌日の同じ時刻に実行されます。

2. 表示されている **at>** プロンプトで、ジョブコマンドを定義します。

   A. ジョブが実行するコマンドを入力し、**Enter** を押します。複数のコマンドがある場合は、このステップを繰り返します。

   B. プロンプトでシェルスクリプトを入力し、スクリプトの各行の後で **Enter** を押します。

      ジョブは、ユーザーの **SHELL** 環境で設定されているシェル、ユーザーのログインシェル、**/bin/sh** のいずれか (最初に検出されたもの) を使用します。

3. 完了したら、空白行で **Ctrl**+**D** を押してプロンプトを終了します。

コマンドセットやスクリプトが標準出力に情報を表示しようと試みる場合は、その出力はユーザーに電子メールで送信されます。

保留中のジョブを表示するには、**atq** コマンドを使用します。詳細は、「保留中のジョブの表示」を参照してください。

at コマンドの使用を制限することもできます。詳細は、「at と batch へのアクセスの制御」を参照してください。

19.2.4. Batch ジョブの設定

Batch アプリケーションは、システム負荷平均が 0.8 を下回った時に、定義済みの 1 回限りのタスクを実行します。

Batch ジョブは、以下の手順で定義します。

1. コマンドラインで **batch** コマンドを入力します。

2. 表示されている **at>** プロンプトで、ジョブコマンドを定義します。

   A. ジョブが実行するコマンドを入力し、**Enter** を押します。複数のコマンドがある場合は、このステップを繰り返します。

   B. プロンプトでシェルスクリプトを入力し、スクリプトの各行の後で **Enter** を押します。

      スクリプトが入力される場合は、 ジョブは、ユーザーの **SHELL** 環境で設定されているシェル、ユーザーのログインシェル、**/bin/sh** のいずれか (最初に検出されたもの) を使用します。

3. 完了したら、空白行で **Ctrl**+**D** を押してプロンプトを終了します。

コマンドセットやスクリプトが標準出力に情報を表示しようと試みる場合は、その出力はユーザーに電子メールで送信されます。

保留中のジョブを表示するには、**atq** コマンドを使用します。詳細は、「保留中のジョブの表示」を参照してください。

batch コマンドの使用を制限することもできます。詳細は、「at と batch へのアクセスの制御」を参照してください。

19.2.5. 保留中のジョブの表示

保留中の **At** ジョブと **Batch** ジョブを表示するには、**atq** コマンドを実行します。**atq** コマンドは、1 行につき 1 ジョブが記述された保留中ジョブの一覧を表示します。各行は、ジョブ番号、日付、時間、ジョブクラス、ユーザー名の形式で記述されます。一般ユーザーが表示できるのは、自身のジョブのみです。**root** ユーザーが**atq** コマンドを実行すると、全ユーザーのすべてのジョブが表示されます。

19.2.6. 追加のコマンドラインオプション

at と **batch** には、以下のような追加のコマンドラインオプションがあります。

表19.1 at と batch のコマンドラインオプション

オプション	詳細
-f	コマンドやシェルスクリプトをプロンプトで指定するのではなく、ファイルから読み込む
-m	ジョブが完了した時点で、ユーザーに電子メールを送信する
-v	ジョブが実行される時刻を表示する

19.2.7. at と batch へのアクセスの制御

/etc/at.allow と **/etc/at.deny** のファイルを使用して、**at** コマンドと **batch** コマンドへのアクセスを制限できます。このアクセス制御ファイルは両方とも、各行にユーザー名を 1 つ配置する同一の形式を使用します。どちらのファイルでも空白は許可されません。

at.allow ファイルが存在する場合は、そのファイルに記載されているユーザーのみが**at** または **batch** の使用を許可され、**at.deny** ファイルは無視されます。

at.allow が存在しない場合は、**at.deny** に記載されているユーザーは**at** または **batch** を使用できません。

アクセス制御ファイルが変更された場合でも、**at** デーモン (**atd**) を再起動する必要はありません。アクセス制御ファイルは、ユーザーが **at** または **batch** のコマンドの実行を試みるたびに読み込まれます。

アクセス制御ファイルのコンテンツに関係なく、**root** ユーザーは常に **at** および **batch** のコマンドを実行することができます。

19.3. その他のリソース

自動化タスクの設定についての詳細情報は、以下のインストール済みのドキュメントを参照してください。

- **cron(8)** man ページ — cron の概要を記載しています。
- **cron** man ページのセクション 1 および 5
  - セクション 1 には、**crontab** ファイルの概要が記載されています。
  - セクション 5 には、ファイル形式とサンプルエントリーが記載されています。
- **anacron(8)** man ページ — anacron の概要を記載しています。
- **anacrontab(5)** man ページ — **anacrontab** ファイルの概要を記載しています。
- **run-parts(4)** man ページ — **run-parts** スクリプトの概要を記載しています。
- **/usr/share/doc/at-*<version>*/timespec** には cron ジョブで指定できる時間の値について詳しく説明しています。

- **at** man ページ — **at** と**batch** およびそれらのコマンドラインオプションについて説明しています。

第20章 自動バグ報告ツール (ABRT)

20.1. ABRT の概要

自動バグ報告ツール (Automatic Bug Reporting Tool) は通常 **ABRT** と呼ばれるツールセットで、ユーザーがアプリケーションクラッシュを検出し報告する手助けとなるように設計されています。主な目的は、問題の報告と解決方法の発見のプロセスを容易にすることです。このコンテキストでの解決方法は、Bugzilla チケット、ナレッジベースの記事、修正を含むバージョンへの更新の提案などです。

ABRT は、**abrtd** デーモンと、検出された問題をプロセス、分析、報告するための数多くのシステムサービスとユーティリティーで構成されます。ほとんどの場合、このデーモンはバックグラウンドでなにも表示せずに実行されますが、アプリケーションがクラッシュしたり カーネル oops が検出されると、アクションを起こします。するとデーモンは、コアファイルがある場合はその関連問題データや、クラッシュしているアプリケーションのコマンドラインパラメーター、さらに他のフォレンジックユーティリティーのデータなどを収集します。

ABRT は現在、C/C++、Java、Python、および Ruby プログラミング言語で書かれたアプリケーションにおけるクラッシュと、X.Org クラッシュ、カーネル oopses、およびカーネルパニックの検出をサポートしています。サポート対象のクラッシュのタイプおよびこれらが検出される方法についての詳細情報は、「ソフトウェア問題の検出」 を参照してください。

特定された問題はリモートの問題トラッカーに報告され、このレポーティングは、問題が検出された際に常に自動的に報告されるように設定することが可能です。問題のデータは、ローカルまたは専用のシステムに保存して、ユーザーが手動でレビュー、報告、削除することも可能です。レポーティングツールは、問題データを Bugzilla データベースや Red Hat テクニカルサポート (RHTSupport) サイトに送信できるほか、**FTP/SCP** を使用したアップロード、電子メールでの送信、ファイルへの書き込みなどにも対応しています。

既存の問題データを処理する **ABRT** は (新規問題データ作成などとは対照的に)、**libreport** という別のプロジェクトの一部です。**libreport** ライブラリーは問題の分析、報告のための包括的メカニズムを提供し、**ABRT** 以外のアプリケーションでも使用されます。ただし、**ABRT** と **libreport** の操作と設定は密接に統合されているため、本ガイドでも取り上げています。

20.2. ABRT のインストールとそのサービスの起動

ABRT を使用するためには、*abrt-desktop* または *abrt-cli* パッケージがシステムにインストールされているか確認します。*abrt-desktop* パッケージは **ABRT** 用のグラフィカルユーザーインターフェイスを提供し、*abrt-cli* パッケージにはコマンドラインで**ABRT** を使用するためのツールが含まれています。これら両方をインストールすることもできます。**ABRT** GUI とコマンドラインツールの全般的なワークフローは手順的には似通っており、同様のパターンになります。

警告

ABRT をインストールすると、コアダンプファイルの命名に使用するテンプレートを含む **/proc/sys/kernel/core_pattern** ファイルを上書きすることに注意してください。このファイルのコンテンツは、以下のように上書きされます。

```
|/usr/libexec/abrt-hook-ccpp %s %c %p %u %g %t e
```

Yum パッケージマネージャーでパッケージをインストールする方法の一般的な情報については、「パッケージのインストール」 を参照してください。

20.2.1. ABRT GUI のインストール

ABRT の *グラフィカルユーザーインターフェイス*は、デスクトップ環境での作業用に操作が容易なフロントエンドを提供します。**root** ユーザーで以下のコマンドを実行すると、必要なパッケージがインストールできます。

```
~]# yum install abrt-desktop
```

インストールすると、**ABRT** 通知アプレットは、グラフィカルデスクトップのセッション開始時に自動的に起動するように設定されます。ターミナルで以下のコマンドを発行すると、**ABRT** アプレットが実行中であることを確認できます。

```
~]$ ps -el | grep abrt-applet
0 S   500  2036  1824  0  80   0 - 61604 poll_s ?        00:00:00 abrt-
applet
```

アプレットが実行されていない場合には、以下のように **abrt-applet** プログラムを実行すると、現行のデスクトップセッションでアプレットを手動で起動することができます。

```
~]$ abrt-applet &
[1] 2261
```

20.2.2. コマンドライン用の ABRT のインストール

*コマンドラインインターフェイス*は、ヘッドレスマシンやネットワーク接続されたリモートシステム、またはスクリプト内で役立ちます。**root** ユーザーで以下のコマンドを実行すると、必要なパッケージがインストールできます。

```
~]# yum install abrt-cli
```

20.2.3. 補助 ABRT ツールのインストール

ABRT が捕捉するクラッシュについての電子メール通知を受け取るには、*libreport-plugin-mailx* パッケージがインストール済みである必要があります。**root** で以下のコマンドを実行すると、このパッケージをインストールできます。

```
~]# yum install libreport-plugin-mailx
```

デフォルトでは、これで通知はローカルマシンの **root** ユーザーに送信されます。電子メールの宛先は、**/etc/libreport/plugins/mailx.conf** ファイルで設定できます。

ログイン時にコンソールで通知を表示するようにするには、*abrt-console-notification* パッケージもインストールします。

ABRT が検出、分析、報告可能なさまざまなクラッシュのタイプに対するサポートは、個別のパッケージに含まれています。C/C++ アプリケーションのクラッシュなどの最も一般的なタイプに対するサポートは、基本的な **ABRT** システムをインストールする際に自動的にインストールされます。基本的な**ABRT** 機能を取得するためにインストールする必要のあるパッケージに関する情報は、「ABRT のインストールとそのサービスの起動」 を参照してください。

しかし、特定タイプのクラッシュに対するサポートを含むパッケージは、別個にインストールする必要があります。たとえば、Java 言語で書かれたアプリケーションでの例外検出へのサポートをインストールするには、**root** で以下のコマンドを実行します。

```
~]# yum install abrt-java-connector
```

ABRT が作業可能な言語およびソフトウェアプロジェクト一覧は、「ソフトウェア問題の検出」 を参照してください。このセクションには、さまざまなタイプのクラッシュ検出が機能するためにインストールが必要な対応パッケージの全一覧も含まれています。

20.2.4. ABRT サービスの起動

abrtd デーモンは、ブート時に起動するように設定されています。以下のコマンドを使うと、現在のステータスが確認できます。

```
~]$ systemctl is-active abrtd.service
active
```

systemctl コマンドが **inactive** または **unknown** を返す場合は、デーモンは実行中ではありません。**root** ユーザーで以下のコマンドを入力すると、現行セッションでデーモンを起動することができます。

```
~]# systemctl start abrtd.service
```

これらと同じステップを実行することで、さまざまなタイプのクラッシュを処理するサービスのステータスをチェックしたり、サービスの起動ができます。たとえば、**ABRT** で C/C++ のクラッシュを検出したい場合は、**abrt-ccpp** サービスの実行を確認します。利用可能な **ABRT** の検出サービスと対応するパッケージの全一覧については、「ソフトウェア問題の検出」 を参照してください。

カーネルパニックまたは oops が実際に発生した場合にのみ起動する **abrt-vmcore** および **abrt-pstoreoops** サービスを除いて、各パッケージのインストール時に、**ABRT** の全サービスをブート時に自動的に有効にして起動することができます。**ABRT** サービスはすべて、*6章systemd によるサービス管理* の説明にあるように **systemctl** ユーティリティーを使って無効にしたり有効にしたりすることができます。

20.2.5. ABRT のクラッシュ検出テスト

ABRT が正常に機能することをテストするには、**kill** コマンドを使って SEGV 信号を送信し、プロセスを終了します。たとえば、以下の方法で **sleep** プロセスを開始して、**kill** コマンドでそれを終了します。

```
~]$ sleep 100 &
[1] 2823
~]$ kill -s SEGV 2823
```

ABRT は **kill** コマンドの実行後まもなくクラッシュを検出し、グラフィカルセッションが実行中であれば、GUI 通知アプレットが検出された問題をユーザーに通知します。コマンドライン環境では、**abrt-cli list** コマンドを実行するか、**/var/tmp/abrt/** ディレクトリーで作成されたクラッシュダンプを調べることで、クラッシュが検出されたかどうかをチェックすることができます。検出されたクラッシュの取り扱い方法については、「検出された問題の処理」 を参照してください。

20.3. ABRT の設定

ABRT では、問題のライフサイクルは**イベント** によって決定されます。例を示します。

- イベント 1 — 問題のデータのディレクトリー作成

- イベント 2 — 問題のデータの分析
- イベント 3 — 問題を Bugzilla に報告

問題が検出されると、**ABRT** はその問題を既存の問題データと比較し、同じ問題が記録されているかどうかを確認します。記録されている場合には、その既存の問題データが更新され、最新の (重複している) 問題は再度記録されません。問題が **ABRT** で認識されない場合は、**problem-data directory (問題データディレクトリー)** が作成されます。問題データディレクトリーは、通常、以下のようなファイルで構成されます:
analyzer、**architecture**、**coredump**、**cmdline**、**executable**、**kernel**、**os_release**、**reason**、**time**、**uid**。

backtrace などの他のファイルは、問題の分析中に作成される場合があります。これは、使用するアナライザーメソッドやその設定によって異なります。これらのファイルには、それぞれシステムおよび問題自体についての固有の情報が格納されます。たとえば **kernel** ファイルには、クラッシュしたカーネルのバージョンが記録されます。

問題データディレクトリーが作成され、問題データが収集された後は、 **ABRT** GUI もしくはコマンドラインの **abrt-cli** ユーティリティーを使ってその問題を処理することができます。記録された問題に対処するために提供されている **ABRT** ツールについての詳細は、「検出された問題の処理」を参照してください。

20.3.1. イベントの設定

ABRT のイベントは、*プラグイン*を使って実際のレポーティング操作を行います。プラグインはコンパクトなユーティリティーで、問題データディレクトリーのコンテンツを処理するためにイベントが呼び出します。プラグインを使うことで、**ABRT** は問題をさまざまな宛先に報告できます。また、ほとんどの報告先に関しては、何らかの設定が必要になります。たとえば、Bugzilla では、ユーザー名とパスワード、および Bugzilla サービスのインスタンスを指す URL が必要になります。

設定詳細の中にはデフォルト値を含めることが可能なものもありますが (Bugzilla URLなど)、適切なデフォルト値を設定できないものもあります (ユーザー名など)。**ABRT** は **report_Bugzilla.conf** のような設定ファイルを、システムワイドの設定には **/etc/libreport/events/** ディレクトリーで、ユーザー固有の設定には **$HOME/.cache/abrt/events/** ディレクトリーで探します。設定ファイルには、指示文と値のペアが含まれています。

これらのファイルは、イベントの実行と問題データディレクトリーの処理のために最小限必要なものです。**abrt-gui** ツールと **abrt-cli** ツールはこれらのファイルから設定データを読み取り、実行するイベントにこれを渡します。

イベントに関する追加情報 (環境変数としてイベントに渡すことができるパラメーターの説明、名前、タイプ、その他の属性など) は、**/usr/share/libreport/events/** ディレクトリーの ***event_name*.xml** ファイルに格納されています。このファイルは、ユーザーインターフェイスをより使いやすくするために **abrt-gui** および **abrt-cli** で使用されます。標準インストールを変更する場合以外には、このファイルは編集しないでください。編集する場合は、対象ファイルを **/etc/libreport/events/** ディレクトリーにコピーして、新規ファイルを修正してください。これらのファイルには、以下の情報が含まれています。

- ユーザーフレンドリーなイベント名および説明 (Bugzilla、Bugzilla バグトラッカーへのレポート)
- イベントが正常に実行されるために必要な問題データディレクトリー内のアイテム一覧
- 送信するおよび送信しないデフォルトおよび必須の選択アイテム
- GUI がデータ見直しを要求するかどうか
- 存在する設定オプション、それらのタイプ (文字列、ブール値など)、デフォルト値、プロンプト文字列など。これらにより、GUI が適切な設定ダイアログを構築できます。

たとえば、**report_Logger** イベントは、出力名をパラメーターとして受け入れます。それぞれの ***event_name*.xml** ファイルを使用することで、**ABRT** GUI は、選択されたイベントに対してどのパラ

メーターを指定することができるかを判断し、ユーザーはそれらのパラメーターの値を設定できるようになります。設定された値は、**ABRT** GUI によって保存され、イベントが後で呼び出される際に再利用されます。**ABRT** GUI は **GNOME Keyring** ツールを使って設定オプションを保存し、これらをイベントに渡すことでテキスト設定ファイルからのデータを上書きすることに注意してください。

グラフィカルの **Configuration** ウィンドウを開くには、実行中の **abrt-gui** アプリケーションのインスタンスから **Automatic Bug Reporting Tool** → **Preferences (設定)** を選びます。このウィンドウでは、GUI 使用したレポーティングのプロセス中に選択可能な全イベントの一覧が表示されます。設定可能なイベントを選択すると、**Configure** ボタンがクリックできる状態となり、そのイベントの設定値を修正することができます。

重要

/etc/libreport/ ディレクトリー階層内のファイルはすべて、全ユーザーが読み取り可能となっており、グローバル設定として使用するためにあります。このため、ユーザー名、パスワード、その他の機密データは、これらのファイル内に保存しないことが推奨されます。ユーザー別の設定 (GUI アプリケーションで設定され、**$HOME** の所有者のみが読み取り可能) は、**GNOME Keyring** に安全に保管されます。または、**abrt-cli** で使用するには、**$HOME/.abrt/** 内のテキスト設定ファイルに保管することもできます。

以下の表では、**ABRT** の標準インストールで提供される、デフォルトの分析、収集、レポートイベントの一部を表示しています。ここでは、**/etc/libreport/events.d/** ディレクトリーからの各イベントの名前、識別子、設定ファイルと簡単な説明を一覧表示しています。設定ファイルはイベント識別子を使用しますが、**ABRT** GUI では個別のイベント名が望ましいことに注意してください。また、すべてのイベントで GUI を使った設定ができるわけではないことにも注意してください。カスタムイベントの定義方法についての情報は、「カスタムイベントの作成」を参照してください。

表20.1 標準 ABRT イベント

名前	識別子および設定ファイル	詳細
uReport	report_uReport	μReport を **FAF** サーバーにアップロードします。
Mailx	report_Mailx **mailx_event.conf**	問題レポートを **Mailx** ユーティリティーを介して指定の電子メールアドレスに送信します。
Bugzilla	report_Bugzilla **bugzilla_event.conf**	問題を **Bugzilla** バグトラッカーの指定インストールにレポートします。
Red Hat Customer Support	report_RHTSupport **rhtsupport_event.conf**	問題を Red Hat テクニカルサポートシステムに報告します。
Emergency analysis	report_EmergencyAnalysis **emergencyanalysis_event.conf**	tarball を FAF サーバーにアップロードして、さらに分析します。標準レポーティングメソッドが失敗した場合に使用されます。
Analyze C/C++ Crash	analyze_CCpp **ccpp_event.conf**	コアダンプをリモートの retrace サーバーに送信して分析します。または、リモート分析が失敗した場合は、ローカルの分析を実行します。

名前	識別子および設定ファイル	詳細
Report uploader	report_Uploader **uploader_event.conf**	**FTP** または **SCP** プロトコルを使用して、問題データの tarball (**.tar.gz**) アーカイブを、選択した宛先にアップロードします。
Analyze VM core	analyze_VMcore **vmcore_event.conf**	カーネル oops の問題データに対して**GDB** (GNU デバッガ) を実行し、カーネルの **backtrace** を生成します。
Local GNU Debugger	analyze_LocalGDB **ccpp_event.conf**	アプリケーションの問題データに対して**GDB** (GNU デバッガ) を実行し、問題の **backtrace** を生成します。
Collect .xsession-errors	analyze_xsession_errors **ccpp_event.conf**	**~/.xsession-errors** ファイルから関連性のある行を問題レポートに保存します。
Logger	report_Logger **print_event.conf**	問題レポートを作成して、それを指定のローカルファイルに保存します。
Kerneloops.org	report_Kerneloops **koops_event.conf**	カーネルの問題を kerneloops.org の oops トラッカーに送信します。

20.3.2. カスタムイベントの作成

各イベントは、それぞれの設定ファイル内の単一のルール構造によって定義されます。これらの設定ファイルは、通常 **/etc/libreport/events.d/** ディレクトリーに格納されます。これらの設定ファイルは、主要設定ファイル **/etc/libreport/report_event.conf** によって読み込まれます。

このファイルを変更する場合には、シェルのメタ文字 (*、$、? など) が優先され、その位置に対して相対パスが解釈される点に注意してください。

各 **ルール** は行頭に空白文字を入れない行で開始します。その後に続く、**空白** 文字または **タブ** 文字で始まる行は、すべてこのルールの一部とみなされます。各 **ルール** は、**条件** パートと **プログラム** パートの 2 部で構成されます。条件パートには、以下のいずれかの形式で条件が記載されます。

- *VAR*=*VAL*
- *VAR*!=*VAL*
- *VAL*~=*REGEX*

ここでは、

- *VAR* は、**EVENT** のキーワードもしくは問題データディレクトリーの要素の名前です (**executable**、**package**、**hostname** など)。
- *VAL* は、イベントまたは問題データの要素名です。
- *REGEX* は正規表現です。

プログラムパートは、プログラム名とシェルが解釈可能なコードで構成されます。条件パートにある全条件が有効な場合、プログラムパートがシェルで実行されます。以下は **イベント** の一例です。

```
EVENT=post-create date > /tmp/dt
        echo $HOSTNAME `uname -r`
```

このイベントは、現在の日付と時刻で **/tmp/dt** ファイルの内容を上書きし、マシンのホスト名とカーネルのバージョンを標準出力に表示します。

より複雑なイベント (実際の事前設定済みイベントの一例) の例は以下のようになります。このイベントは、**abrt-ccpp** サービスを使用して処理された問題の問題レポートに**~/.xsession-errors** ファイルの関連する行を保存します。クラッシュ発生時点で、クラッシュしたアプリケーションに X11 ライブラリーが読み込まれていたことが条件になります。

```
EVENT=analyze_xsession_errors analyzer=CCpp dso_list~=.*/libX11.*
        test -f ~/.xsession-errors || { echo "No ~/.xsession-errors";
exit 1; }
        test -r ~/.xsession-errors || { echo "Can't read ~/.xsession-
errors"; exit 1; }
        executable=`cat executable` &&
        base_executable=${executable##*/} &&
        grep -F -e "$base_executable" ~/.xsession-errors | tail -999
>xsession_errors &&
        echo "Element 'xsession_errors' saved"
```

可能性のあるイベントのセットは、最終的なものではありません。システム管理者は、必要に応じて、イベントを追加することができます。現在、**ABRT** と **libreport** の標準インストールでは、以下のイベント名が提供されています。

post-create

このイベントは、**abrtd** により、新規作成された問題データディレクトリーを処理するために実行されます。**post-create** イベントが実行されると、**abrtd** は、新規問題データの **id** 識別子が既存の問題ディレクトリーの **id** と一致するかどうかを確認します。一致する問題ディレクトリーがある場合には、新規問題データは削除されます。

notify、notify-dup

notify イベントは、**post-create** の完了後に実行されます。このイベントが実行されると、問題に注意する必要があることをユーザーが確認できます。**notify-dup** も同様のものですが、これは同一問題が重複して発生した場合に使用されます。

analyze_*name_suffix*

ここでの *name_suffix* は、イベント名の調整可能な部分です。このイベントは、収集データの処理に使用されます。たとえば、**analyze_LocalGDB** は GNU Debugger (**GDB**) ユーティリティー使用してアプリケーションのコアダンプを処理し、クラッシュのバックトレースを生成します。

collect_*name_suffix*

ここでの *name_suffix* は、イベント名の調整可能な部分です。このイベントは、問題に関する追加情報を収集するために使用されます。

report_*name_suffix*

ここでの *name_suffix* は、イベント名の調整可能な部分です。このイベントは、問題を報告するために使用されます。

20.3.3. 自動レポーティングの設定

ABRT は、検出されたすべての問題またはクラッシュの最初の匿名レポート (*μReports* とも呼ぶ) をユーザーのインタラクションなしに自動で送信するように設定できます。自動レポーティングがオンになってい

ると、クラッシュの検出直後に、通常クラッシュレポーティングプロセスの最初に送信される μReport が送信されます。これにより、同一クラッシュに基づくサポートケースの重複が効果的に防ぐことができます。自動レポーティング機能を有効にするには、**root** で以下のコマンドを発行します。

```
~]# abrt-auto-reporting enabled
```

上記のコマンドは、**/etc/abrt/abrt.conf** 設定ファイル内の **AutoreportingEnabled** 指示文を **yes** に設定します。このシステムワイドの設定は、システムの全ユーザーに適用されます。このオプションを有効にすると、グラフィカルデスクトップ環境でも自動レポーティングが有効になることに注意してください。**ABRT** GUI の自動レポーティングのみを有効にするには、**Problem Reporting Configuration** ウィンドウ内で **Automatically send uReport** オプションを **YES** に切り替えてください。このウィンドウを開くには、**abrt-gui** アプリケーションの実行中のインスタンス内で **Automatic Bug Reporting Tool** → **ABRT Configuration** を選択します。このアプリケーションを起動するには、**アプリケーション** → **諸ツール** → **自動バグ報告ツール (ABRT)** に移動します。

デフォルトでは、**ABRT** はクラッシュを検出すると、問題の基本情報とともに μReport を Red Hat の **ABRT** サーバーに提出します。サーバーは問題が既知のものかどうかを判断し、既知の場合はそのケースの URL と問題の簡単な説明を提供します。既知でない場合は、ユーザーに報告するように促します。

注記

μReport (マイクロレポート) は、バイナリークラッシュやカーネル oops などの問題を表す JSON オブジェクトです。これらのレポートは、小規模でシステム内部で使用し、完全に匿名になるように設計されています。自動レポーティングでこれが使用できるのは、このためです。μReports によりバグ発生の追跡が可能になりますが、通常はバグ修正のために十分な情報をエンジニアに提供しません。サポートケースが開かれるには、完全なバグレポートが必要になります。

自動レポーティング機能の動作を、μReport ではなく別のなにかを送信するように変更するには、**/etc/abrt/abrt.conf** 設定ファイル内の **AutoreportingEvent** 指示文の値を別の **ABRT** イベントに向けるように修正する必要があります。標準イベントの概要については、表20.1「標準 ABRT イベント」 を参照してください。

20.4. ソフトウェア問題の検出

ABRT は、多くの異なるプログラミング言語で書かれたアプリケーションにおけるクラッシュを検出、分析、プロセスすることができます。さまざまなタイプのクラッシュ検出のサポートを含むパッケージの多くは、**ABRT** の主要パッケージのどれか (*abrt-desktop*、*abrt-cli*) がインストールされる際に自動的にインストールされます。**ABRT** のインストール方法については、「ABRT のインストールとそのサービスの起動」 を参照してください。下記の表は、サポート対象のクラッシュタイプと対応パッケージを一覧表示したものです。

表20.2 サポート対象のプログラミング言語およびソフトウェアプロジェクト

言語/プロジェクト	パッケージ
C/C++	*abrt-addon-ccpp*
Python	*abrt-addon-python*
Ruby	*rubygem-abrt*
Java	*abrt-java-connector*
X.Org	*abrt-addon-xorg*
Linux (カーネル oops)	*abrt-addon-kerneloops*
Linux (カーネルパニック)	*abrt-addon-vmcore*

言語/プロジェクト	パッケージ
Linux (永続的なストレージ)	*abrt-addon-pstoreoops*

20.4.1. C/C++ クラッシュの検出

abrt-ccpp サービスは、自身のコアダンプハンドラーをインストールします。これが起動すると、カーネルの **core_pattern** 変数のデフォルト値を上書きするので、C/C++ クラッシュは **abrtd** が処理します。**abrt-ccpp** サービスを停止すると、以前に指定された**core_pattern** の値が回復されます。

デフォルトでは、**/proc/sys/kernel/core_pattern** ファイルは **core** 文字列を格納しています。つまり、カーネルは **core.*** という名前のファイルをクラッシュしたプロセスの現行ディレクトリーに作成します。**abrt-ccpp** サービスは **core_pattern** ファイルを上書きして、以下のコマンドを格納します。

```
|/usr/libexec/abrt-hook-ccpp %s %c %p %u %g %t e
```

このコマンドは、カーネルがコアダンプを **abrt-hook-ccpp** プログラムにパイプ処理するように指示します。このプログラムは、**ABRT** のダンプの場所にそれを保存し、**abrtd** デーモンに新しいクラッシュを通知します。また、デバッグ目的で **/proc/*PID*/** ディレクトリー (*PID* はクラッシュしたプロセスの ID) から **maps**、**limits**、**cgroup**、**status** の各ファイルを保存します。これらのファイルのフォーマットおよび意味についての説明は、proc(5) を参照してください。

20.4.2. Python 例外の検出

abrt-addon-python パッケージは、Python アプリケーション用のカスタム例外ハンドラーをインストールします。すると Python インタープリターは、**/usr/lib64/python2.7/site-packages/** にインストール済みの **abrt.pth** ファイルを自動的にインポートします。このファイルは同様に、**abrt_exception_handler.py** をインポートします。これにより Python のデフォルトの **sys.excepthook** がカスタムハンドラーで上書きされ、未処理の例外がソケット API 経由で**abrtd** に転送されます。

サイト固有のモジュールの自動インポートを無効にして、Python アプリケーション実行時に **ABRT** カスタム例外ハンドラーが使われないようにするには、Python インタープリターに **-S** オプションを渡します。

```
~]$ python -S file.py
```

このコマンドでは、*file.py* をサイト固有のモジュール使用なしで実行する Python スクリプトの名前で置き換えます。

20.4.3. Ruby 例外の検出

rubygem-abrt パッケージは、**at_exit** 機能を使用するカスタムハンドラーを登録します。これは、プログラムの終了時に実行されます。これにより、処理されていない可能性のある例外の確認が可能になります。未処理の例外が獲得されると、**ABRT** は毎回バグレポートを準備します。これは標準の**ABRT** ツールを使ってRed Hat Bugzilla に提出することができます。

20.4.4. Java 例外の検出

ABRT Java コネクターは、捕捉されていない Java 例外を**abrtd** にレポートする JVM エージェントです。これは、いくつかの JVMTI イベントコールバックを登録し、**-agentlib** コマンドラインパラメーターを使って JVM に読み込む必要があります。登録済みコールバックの処理は、アプリケーションのパフォーマンスにマイナスの影響を与えることに注意してください。**ABRT** が Java クラスから例外を捕捉するようにするには、以下のコマンドを使用します。

```
~]$ java -agentlib:abrt-java-connector[=abrt=on] $MyClass -
platform.jvmtiSupported true
```

ここでは *$MyClass* をテストする Java クラス名に置き換えます。**abrt=on** オプションをコネクターに渡すことで、例外が **abrtd** に処理されるようになります。コネクターが例外を標準出力に出すようにするには、このオプションを省略します。

20.4.5. X.Org クラッシュの検出

abrt-xorg は、**X.Org server** のクラッシュについての情報を **/var/log/Xorg.0.log** ファイルから収集して処理します。ブラックリスト化された **X.org** モジュールが読み込まれている場合は、レポートが生成されないことに注意してください。代わりに適切な説明の付いた **not-reportable** ファイルが問題データディレクトリーに作成されます。問題となっているモジュール一覧は、**/etc/abrt/plugins/xorg.conf** ファイルで確認できます。デフォルトでは、商用のグラフィックドライバーのモジュールのみがブラックリスト化されています。

20.4.6. カーネル Oops およびカーネルパニックの検出

カーネルログの出力をチェックすることで、**ABRT** は致命的でない Linux カーネルの正常な動作からの逸脱であるカーネル oops を捕捉して処理することができます。この機能は、**abrt-oops** サービスが提供します。

ABRT は、**abrt-vmcore** サービスを使って、致命的で再起動を必要とする回復不可能なエラーであるカーネルパニックも検出して処理することができます。このサービスは、**vmcore** ファイル (カーネルコアダンプ) が **/var/crash/** ディレクトリーに見える時にのみ起動します。コアダンプファイルが見つかると、**abrt-vmcore** が新たな **problem-data directory** を **/var/tmp/abrt/** ディレクトリー内に作成し、コアダンプファイルをこの新規作成の問題データディレクトリーに移動します。**/var/crash/** ディレクトリーの検索後に、このサービスは停止します。

ABRT がカーネルパニックを検出できるようになるには、**kdump** サービスがシステム上で有効になっている必要があります。kdump カーネル用に確保されたメモリー容量が正しく設定されている必要があります。この設定は、 **system-config-kdump** グラフィカルツールを使うか、**/etc/default/grub** 設定ファイルにあるカーネルオプション一覧の **crashkernel** パラメーターを指定することで実行できます。**kdump** の有効化と設定の詳細については、Red Hat Enterprise Linux 7 Kernel Crash Dump Guide を参照してください。

abrt-pstoreoops サービスを使うと、**ABRT** はカーネルパニックについての情報を収集、処理できるようになります。これは、*pstore* 対応のシステムでは、自動マウントされた**/sys/fs/pstore/** ディレクトリーに保存されます。このサービスは、**/sys/fs/pstore/** ディレクトリーにカーネルのクラッシュダンプファイルが現れると、自動的に起動します。

20.5. 検出された問題の処理

abrtd で保存された問題データは、コマンドラインツール**abrt-cli** かグラフィカルツール**abrt-gui** を使用して表示、報告、削除することができます。

注記

ABRT は、新たな問題をローカルに保存されている全問題と比較することにより、問題の重複を特定することに注意してください。繰り返し発生しているクラッシュの場合、**ABRT** が対応を要求するのは 1 回のみですが、その問題のクラッシュダンプを削除してしまうと、その特定の問題が次回発生した際、**ABRT** はその問題を新規クラッシュとして処理することになります。このため、**ABRT** は警告を表示し、その問題の詳細を記入して報告するように要求します。**ABRT** が繰り返し発生する問題の通知をしないようにするには、その問題データを削除しないでください。

20.5.1. コマンドラインツールの使用

コマンドライン環境では、*abrt-console-notification* パッケージがインストール済みであれば、ログイン時に新たなクラッシュがユーザーに通知されます。コンソールでの通知は、以下のようになります。

```
ABRT has detected 1 problem(s). For more info run: abrt-cli list --since
1398783164
```

検出された問題を表示するには、**abrt-cli list** コマンドを入力します。

```
~]$ abrt-cli list
id 6734c6f1a1ed169500a7bfc8bd62aabaf039f9aa
Directory:      /var/tmp/abrt/ccpp-2014-04-21-09:47:51-3430
count:          1
executable:     /usr/bin/sleep
package:        coreutils-8.22-11.el7
time:           Mon 21 Apr 2014 09:47:51 AM EDT
uid:            1000
Run 'abrt-cli report /var/tmp/abrt/ccpp-2014-04-21-09:47:51-3430' for
creating a case in Red Hat Customer Portal
```

abrt-cli list コマンドの出力で一覧表示されるクラッシュにはそれぞれ、一意の識別子と**abrt-cli** を使用してさらなる操作に使用可能なディレクトリーがあります。

特定の問題に関する情報のみを表示するには、**abrt-cli info** コマンドを使用します。

```
abrt-cli info [-d] directory_or_id
```

list および **info** の各サブコマンドの使用時に表示する情報量を増やすには、**-d** (**--detailed**) オプションを渡します。これで各問題の **backtrace** ファイルが生成されていれば、これらを含む問題の保存されているすべての情報が表示されます。

特定の問題を分析して報告するには、**abrt-cli report** コマンドを使用します。

```
abrt-cli report directory_or_id
```

上記のコマンドを実行すると、Red Hat カスタマーサポートでサポートケースを開くための証明書の提供を求められます。次に、**abrt-cli** がレポートコンテンツのあるテキストエディターを開きます。これでレポート内容を確認でき、クラッシュを再現させるための指示やコメントを記入できます。また、バックトレースもチェックしてください。バックトレースは、問題報告イベントの設定によっては公開サーバーに送信され、誰でも見ることができるようになる場合があります。

注記

レポートのチェックに使用されるテキストエディターは選択することができます。**abrt-cli** は **ABRT_EDITOR** 環境変数内で定義されたエディターを使用します。変数が定義されていない場合は、**VISUAL** 変数と **EDITOR** 変数がチェックされます。これらの変数のどちらも設定されていない場合は、**vi** エディターが使用されます。希望のエディターは、**.bashrc** 設定ファイル内で設定できます。たとえば、**GNU Emacs** を使用した場合は、このファイルに以下の行を追加します。

```
export VISUAL=emacs
```

レポートでの作業が終わったら、変更を保存してエディターを終了します。Red Hat カスタマーサポートデータベースに報告した場合には、問題のケースがデータベースに提出されます。この時点から、報告プロセス中に提供した電子メールに問題の解決の進捗状況が通知されるようになります。また、問題のケースが作成された際に提供された URL や Red Hat サポートからの電子メールで問題ケースを確認することもできます。

特定の問題を報告したくないことが分かっている場合は、その問題を削除することができます。問題を削除して、その情報が **ABRT** に保存されないようにするには、以下のコマンドを使用します。

```
abrt-cli rm directory_or_id
```

特定の **abrt-cli** コマンドについてのヘルプを表示するには、**--help** オプションを使用します。

```
abrt-cli command --help
```

20.5.2. GUI の使用

ABRT デーモンは、問題レポートが作成されると常に**D-Bus** メッセージをブロードキャストします。グラフィカルデスクトップ環境で **ABRT** アプレットが実行中の場合は、これがそのメッセージを捕捉して、デスクトップに通知ダイアログを表示します。このダイアログの **Report** ボタンをクリックすれば、**ABRT** GUI を開くことができます。また、**アプリケーション** → **諸ツール** → **自動バグ報告ツール (ABRT)** のメニューアイテムを選択して **ABRT** GUI を開くこともできます。

別の方法では、以下のようにコマンドラインから **ABRT** GUI を実行することもできます。

```
~]$ abrt-gui &
```

ABRT GUI ウィンドウは、検出された問題の一覧を表示します。各問題エントリーは、障害が発生したアプリケーション名、アプリケーションがクラッシュした理由、その問題が最後に発生した日付で構成されます。

問題の詳細な説明にアクセスするには、問題レポートの行をダブルクリックするか、各問題の行を選択し、**Report** ボタンをクリックします。指示にしたがうと、問題の説明、分析方法の決定、問題の報告先に進みます。問題を破棄するには、**Delete** ボタンをクリックします。

20.6. その他のリソース

ABRT および関連トピックについての詳細情報は、以下に挙げるリソースを参照してください。

インストールされているドキュメント

- abrtd(8) — **abrtd** デーモンの man ページは、このデーモンと使用可能なオプションについての情報を提供します。
- abrt_event.conf(5) — **abrt_event.conf** 設定ファイルの man ページは、指示文とルールのフォーマットを説明し、XML ファイル内のイベントメタデータ設定についての参照情報を提供します。

オンラインのドキュメント

- Red Hat Enterprise Linux 7 Networking Guide — Red Hat Enterprise Linux 7 の 『Networking Guide』 では、システム上のネットワークインターフェイスおよびネットワークサービスの設定および管理に関する情報が説明されています。
- Red Hat Enterprise Linux 7 Kernel Crash Dump Guide — Red Hat Enterprise Linux 7 の 『Kernel Crash Dump Guide』 は、**kdump** クラッシュ回復サービスの設定、テスト、使用方法について説明します。また、それによって生じるコアダンプを **crash** デバッグユーティリティーで分析する方法についての概要を提供します。

関連項目

- 18章 *ログファイルの表示と管理* では、**rsyslog** デーモンの設定、ログファイルの検索、表示、監視方法について説明しています。
- 5章 *Yum* では、**yum** パッケージマネージャーを使って、コマンドラインでパッケージを検索、インストール、更新、アンインストールする方法について説明しています。
- 6章 *systemd によるサービス管理* では、**systemd** の概説と、**systemctl** コマンドを使ったシステムサービスの管理方法、systemd ターゲットの設定方法、および電源管理コマンドの実行方法について説明しています。

第21章 OProfile

OProfile は、オーバーヘッドの少ないシステムワイドのパフォーマンス監視ツールです。プロセッサー上にあるパフォーマンス監視ハードウェアを使用して、メモリーの参照時期、L2 キャッシュ要求の回数、および受け取ったハードウェア割り込みの回数など、システム上のカーネルと実行可能ファイルに関する情報を取り込みます。Red Hat Enterprise Linux システムでは、このツールを使用するために **oprofile** パッケージをインストールする必要があります。

多くのプロセッサーには、専用のパフォーマンス監視ハードウェアが含まれています。このハードウェアにより特定のイベント (要求されたデータがキャッシュ内に存在しないなど) の発生を検出することができるようになります。このハードウェアは通常、単数または複数の *カウンター*の形式を取り、イベントの発生の度にカウントが増加します。カウンターの値が増えると、割り込みが生成されます。これによりパフォーマンス監視で生成された詳細 (およびオーバヘッド) の量の制御が可能になります。

OProfile は、カウンターが割り込みを生成する度にこのハードウェア (または、パフォーマンス監視ハードウェア不在の場合には、タイマーベースの代用品) を使用してパフォーマンス関連のデータの *サンプル*を収集します。これらのサンプルは定期的にディスクに書き込まれます。その後、これらのサンプル内に含まれるデータはシステムレベル、およびアプリケーションレベルのパフォーマンスに関する情報の生成に利用されます。

OProfile は有用なツールですが、使用時にはいくつかの制約があります。

- **共有ライブラリーの使用** — 共有ライブラリー内のコードサンプルは、**--separate=library** オプションを使用しない限り、特定のアプリケーションに帰属しません。
- **パフォーマンス監視サンプルが不正確** — パフォーマンス監視レジスタがサンプルを起動する際に、割り込みの処理はゼロでの割り算例外エラーのように正確ではありません。プロセッサーによる命令の無秩序な実行により、サンプルは近隣の命令に記録される可能性があります。
- **opreport はサンプルをインライン関数用に不的確に関連付け** — **opreport** は、単純なアドレス範囲のメカニズムを使用してどの関数にアドレスがあるかを判定します。インライン関数のサンプルは、インライン関数に帰属するのではなく、むしろインライン関数が挿入されている関数に帰属します。
- **OProfile 複数の実行によるデータを集積** — OProfile は、システム全体のプロファイラーであり複数のスタートアップとシャットダウンのプロセスを想定します。そのため、複数実行からのサンプルが集積されます。コマンド **opcontrol --reset** を使用すると以前の実行からのサンプルをクリアすることができます。
- **ハードウェアパフォーマンスカウンターはゲスト仮想マシン上では機能しない** — ハードウェアパフォーマンスカウンターは仮想システム上では利用できないため、**timer** モードを使用する必要があります。コマンド **opcontrol --deinit** を実行し、それから **modprobe oprofile timer=1** を実行することで **timer** モードが有効になります。
- **CPU-限定パフォーマンス以外の問題** — OProfile は CPU 限定のプロセスで問題を見つけるように設定されています。ロック状態か、あるいは他のイベントの発生を待つために (例えば、I/O が動作を終了するまで) 待機して眠っているプロセスを OProfile は識別しません。

21.1. ツールの概要

表21.1「OProfile コマンド」は、**oprofile** パッケージで提供されるツールで最もよく使われるものを概説しています。

表21.1 OProfile コマンド

コマンド	詳細

コマンド	詳細
ophelp	システムプロセッサで使用可能なイベントとその簡単な説明を表示します。
opimport	サンプルデータベースファイルをシステム用に外部のバイナリ形式からネイティブの形式に変換します。異なるアーキテクチャーからのサンプルデータベースを解析する時にのみこのオプションを使用してください。
opannotate	アプリケーションがデバッグシンボルでコンパイルされている場合は、実行可能ファイル用の注釈付きのソースを作成します。詳細については、「**opannotate** の使用」を参照してください。
opcontrol	収集されるデータを設定します。詳細については、「レガシーモードを使った OProfile の設定」を参照してください。
operf	**opcontrol** の代わりにプロファイリング用に使用が推奨されるツール。詳細は、「operf の使用」を参照してください。 **operf** と **opcontrol** の違いについては、「operf vs. opcontrol」を参照してください。
opreport	プロファイルデータを取り込みます。詳細については、「**opreport** の使用」を参照してください。
oprofiled	デーモンとして実行して定期的にサンプルデータをディスクに書き込みます。

21.1.1. operf vs. opcontrol

OProfile でプロファイリングデータを収集するには、2 つの相互排他的な方法があります。新しく、推奨される **operf** か **opcontrol** ツールのどちらかを使うことができます。

operf

こちらがプロファイリングで推奨されるモードです。**operf** ツールは Linux Performance Events Subsystem を使用するので、**oprofile** カーネルドライバーを必要としません。**operf** ツールでは、単一プロセスもしくはシステムワイドとしてより正確なプロファイリングを対象とすることができ、システムでパフォーマンス監視ハードウェアを使用する他のツールと OProfile がよりうまく共存することが可能になります。**opcontrol** とは異なり、**root** 権限なしで使用可能です。ただし、**operf** は **--system-wide** オプションを使ってシステムワイドの操作もできますが、その場合は**root** 権限が必要になります。

operf では、初期設定が不要です。コマンドラインオプションで**operf** 起動して、プロファイリング設定を指定することができます。それ以降は、「データの分析」で説明されている OProfile 後処理ツールを実行できます。詳細は、「operf の使用」を参照してください。

レガシーモード

このモードは、**opcontrol** シェルスクリプト、**oprofiled** デーモン、およびいくつかの後処理ツールで構成されます。**opcontrol** コマンドは、プロファイリングセッションの設定、開始、停止に使われます。OProfile カーネルドライバーは通常カーネルモジュールとして構築されており、サンプルの収集に使用されます。これはその後、**oprofiled** によってサンプルファイルに記録されます。レガシーモードは **root** 権限がある場合にのみ、使用できます。割り込み要求 (IRQ) が無効になっているエリアのサンプルが必要な場合などのケースに、このモードはより適していします。

OProfile をレガシーモードで実行できるようにするには、まず「レガシーモードを使った OProfile の設定」にある設定が必要になります。これらの設定は、OProfile 開始時に適用されます (「レガシーモードを使った OProfile の起動と停止」)。

21.2. operf の使用

上記の説明にあるように、**operf** は起動前に設定が不要で、推奨されるプロファイリングモードです。すべての設定は、コマンドラインオプションとして指定され、プロファイリングプロセスを開始する別のコマンドはありません。**operf** を停止するには、Ctrl-c を押します。通常の**operf** コマンド構文は以下のようになります。

```
operf options range command args
```

options をプロファイリング設定を指定するオプションで置き換えます。オプションの全セットは、**operf** man ページで確認できます。*range* を以下のいずれかに置き換えます。

--system-wide - この設定では、グローバルプロファイリングが可能になります。システムワイドモードでの **operf** の使用 を参照してください。

--pid=*PID* - これは実行中のアプリケーションのプロファイリングを行います。*PID* はプロファイリング対象となるプロセスのプロセス ID です。

command と *args* では、プロファイリング対象となる特定のコマンドもしくはアプリケーションを定義します。また、このコマンドもしくはアプリケーションが必要とする入力引数も定義します。*command*、**--pid**、また、 **--system-wide** のいずれかが必要になりますが、これらの同時使用はできません。

operf を *range* オプションを設定せずにコマンドラインで実行すると、その子プロセス用のデータが収集されます。

注記

operf --system-wide を実行するには、**root** 権限が必要になります。プロファイリングが終了したら、以下の操作で **operf** を停止できます。

```
Ctrl-C
```

operf --system-wide をバックグラウンドプロセスとして (**&** を使って) 実行する場合は、収集したプロファイルデータを処理するために制御した方法でこれを停止します。このため、以下を使用します。

```
kill -SIGINT operf-PID
```

operf --system-wide の実行時は、作業ディレクトリーが **/root** または **/root** のサブディレクトリーであることが推奨されます。こうすると、サンプルのデータファイルが通常ユーザーがアクセス可能な場所に保存されません。

21.2.1. カーネルの指定

カーネルを監視するには、以下のコマンドを実行します。

```
operf --vmlinux=vmlinux_path
```

このオプションを使うと、実行中のカーネルに適合する vmlinux ファイルへのパスを指定できます。カーネルサンプルはこのバイナリーに属することになり、後処理ツールがサンプルを適切なカーネルシンボルに帰することができるようになります。このオプションを指定しないと、カーネルサンプルはすべて、"no-vmlinux" という名前の仮のバイナリーに属することになります。

21.2.2. 監視するイベントのセッティング

ほとんどのプロセッサーにはカウンターが含まれており、OProfile はこれを使用して特定のイベントを監視します。表21.3「OProfile プロセッサーおよびカウンター」で示してあるように、利用可能なカウンターの数はプロセッサーによって異なります。

各カウンター用のイベントはコマンドライン、またはグラフィカルインターフェイスで設定できます。グラフィカルインターフェイスに関する情報については、「グラフィカルインターフェース」を参照してください。カウンターが特定のイベントにセットできない場合は、エラーメッセージが表示されます。

注記

旧モデルのプロセッサーの中には、基礎となる Linux Performance Events Subsystem カーネルがサポートしていないものもあり、このため **operf** も対応していません。**operf** の使用を試みて、

```
Your kernel's Performance Events Subsystem does not support your
processor type
```

上記のメッセージが表示されたら、**opcontrol** でのプロファイリングを試してください。使用中のプロセッサーが OProfile のレガシーモードでサポートされるかどうかが分かります。

注記

ゲスト仮想マシン上ではハードウェアパフォーマンスのカウンターが利用できないので、**timer** モードを有効にして仮想マシン上で **operf** を使用する必要があります。これを行うには、**root** で以下を入力します。

```
opcontrol --deinit
```

```
modprobe oprofile timer=1
```

コマンドラインで各設定可能なカウンター用にイベントを設定するには、以下を使用します。

```
operf --events=event1,event2…
```

ここでは、プロファイリング用のイベント仕様をコンマ区切りのリストで渡します。各イベント仕様は、属性のコロン区切りのリストで、以下の形式になります。

```
event-name:sample-rate:unit-mask:kernel:user
```

表21.2「イベントの仕様」では、これらのオプションをまとめています。最後の 3 つの値は、オプションなので、省略するとデフォルト値に設定されます。特定のイベントでは、ユニットマスクが必要になることに注意してください。

表21.2 イベントの仕様

仕様	詳細
event-name	**ophelp** から取られたシンボリックイベント名です。

仕様	詳細
sample-rate	再度サンプリングを行うまでのイベントの数です。カウントが少ないとサンプリングの頻度が増えます。頻繁に発生しないイベントでは、統計的に意味のあるイベントのインスタンス数を獲得するために低いカウントが必要になる可能性があります。一方で、サンプリングの頻度が高まると、システムのオーバーロードにつながります。デフォルトでは、OProfile は時間ベースのイベントセットを使用し、これは 1 プロセッサーあたり 10 万クロックサイクルにつき 1 つのサンプルを作成します。
unit-mask	ユニットマスクはイベントをさらに定義するもので、**ophelp** で一覧表示されます。"0x" で始まる 16 進数値か、**ophelp** に記載されているユニットマスクの最初の単語に一致する文字列を挿入することができます。名前による定義は、**ophelp** の出力で示される "extra:" パラメーターのあるユニットマスクにのみ有効です。このタイプのユニットマスクは、16 進数値では定義できません。特定のアーキテクチャーでは、同一の 16 進数値を持つ複数のユニットマスクが存在することができます。この場合、名前のみによる指定が必要になります。
kernel	カーネルコードのプロファイリングを行うかどうかを指定します (**0** または **1**(デフォルト) を挿入)。
user	ユーザースペースコードのプロファイリングを行うかどうかを指定します (**0** または **1**(デフォルト) を挿入)。

利用可能なイベントはプロセッサーのタイプによって異なります。イベント仕様が与えられない場合は、実行中のプロセッサータイプのデフォルトイベントがプロファイリングに使用されます。これらのデフォルトイベント一覧は、表21.4「デフォルトのイベント」を参照してください。プロファイリングに使用可能なイベントを判断するには、**ophelp** コマンドを使用します。

```
ophelp
```

21.2.3. サンプルのカテゴリー分け

--separate-thread オプションは、スレッドグループ ID (tgid) とスレッド ID (tid) ごとにサンプルをカテゴリー分けします。これはマルチスレッドのアプリケーションでスレッド別のサンプルを見る際に便利です。**--system-wide** オプションと合わせて使うと、**--separate-thread** は複数のプロセスがプロファイリング実行中に同一プログラムを実行しているケースでのプロセス別 (スレッドグループごと) のサンプルを見る際にも便利です。

--separate-cpu オプションは、cpu 別にサンプルをカテゴリー分けします。

21.3. レガシーモードを使った OProfile の設定

OProfile をレガシーモードで実行できるようにするには設定が必要です。少なくとも、カーネルを監視する選択 (またはカーネルを監視しない選択) が必要です。以下のセクションでは、OProfile を設定するための **opcontrol** ユーティリティーの使用法を説明しています。**opcontrol** コマンドが実行されると、そのセットアップオプションは **/root/.oprofile/daemonrc** ファイル内に保存されます。

21.3.1. カーネルの指定

先ず最初に OProfile がカーネルを監視すべきかどうかを設定します。これが OProfile を開始する前に要求される唯一の設定オプションです。他のすべてはオプションです。

カーネルを監視するには、**root** で以下のコマンドを実行します。

```
opcontrol --setup --vmlinux=/usr/lib/debug/lib/modules/`uname -
r`/vmlinux
```

重要

カーネルの監視には、カーネル用の *debuginfo* パッケージ (展開したカーネルを含む) のインストールが必要になります。 このパッケージのインストール方法については、Red Hat カスタマーポータルの kernel-debuginfo などの debuginfo パッケージをダウンロードする方法の記事を参照してください。

OProfile がカーネルを監視しないように設定するには、**root** で以下のコマンドを実行します。

```
opcontrol --setup --no-vmlinux
```

このコマンドは、**oprofile** カーネルモジュールがまだ読み込まれていない場合にその読み込みも行います。また、**/dev/oprofile/** ディレクトリーが存在しない場合には、そのディレクトリーも作成します。このディレクトリーに関する詳細は、「**/dev/oprofile/** を理解する」を参照してください。

カーネル内でサンプルを収集するかどうかの設定が変更するのは、収集されるデータのみで。収集されたデータの保存方法や保存場所は変更されません。カーネルおよびアプリケーションライブラリー用に異なるサンプルファイルを生成する方法については、「カーネルとユーザースペースのプロファイルの分離」を参照してください。

21.3.2. 監視するイベントのセッティング

ほとんどのプロセッサーには *カウンター*が装備されており、OProfile はこれを使用して特定のイベントを監視します。表21.3「OProfile プロセッサーおよびカウンター」で示してあるように、利用可能なカウンターの数はプロセッサにより異なります。

表21.3 OProfile プロセッサーおよびカウンター

プロセッサー	cpu_type	カウンターの数
AMD64	x86-64/hammer	4
AMD Family 10h	x86-64/family10	4
AMD Family 11h	x86-64/family11	4
AMD Family 12h	x86-64/family12	4
AMD Family 14h	x86-64/family14	4
AMD Family 15h	x86-64/family15	6
IBM eServer System i および IBM eServer System p	timer	1
IBM POWER4	ppc64/power4	8
IBM POWER5	ppc64/power5	6
IBM PowerPC 970	ppc64/970	8
IBM PowerPC 970MP	ppc64/970MP	8
IBM POWER5+	ppc64/power5+	6
IBM POWER5++	ppc64/power5++	6

プロセッサー	cpu_type	カウンターの数
IBM POWER56	ppc64/power6	6
IBM POWER7	ppc64/power7	6
IBM S/390 および IBM System z	timer	1
Intel Core i7	i386/core_i7	4
Intel Nehalem マイクロアーキテクチャー	i386/nehalem	4
Intel Westmere マイクロアーキテクチャー	i386/westmere	4
Intel Haswell マイクロアーキテクチャー (ハイパースレッディングなし)	i386/haswell	8
Intel Haswell マイクロアーキテクチャー (ハイパースレッディング)	i386/haswell-ht	4
Intel Ivy Bridge マイクロアーキテクチャー (ハイパースレッディングなし)	i386/ivybridge	8
Intel Ivy Bridge マイクロアーキテクチャー (ハイパースレッディング)	i386/ivybridge-ht	4
Intel Sandy Bridge マイクロアーキテクチャー (ハイパースレッディングなし)	i386/sandybridge	8
Intel Sandy Bridge マイクロアーキテクチャー	i386/sandybridge-ht	4
TIMER_INT	timer	1

表21.3「OProfile プロセッサーおよびカウンター」を使用して、正しいプロセッサータイプが検出されたことを確認し、同時に監視できるイベントの数を判断します。サポートされているパフォーマンス監視ハードウェアをプロセッサーが装備していない場合、**timer** が、プロセッサータイプとして使用されます。

timer が使用される場合、ハードウェアはハードウェアパフォーマンスカウンター用のサポートを持たないため、イベントをプロセッサー用に設定できません。その代わりに、タイマー割り込み (timer interrupt) がプロファイル用に使用されます。

timer がプロセッサータイプとして使用されない場合は、監視されるイベントは変更できません。そしてプロセッサー用にカウンター 0 がデフォルトでタイムベースのイベントにセットされます。プロセッサー上に複数のカウンターが存在する場合、カウンター 0 以外のカウンターはデフォルトではイベントにセットされません。監視されるデフォルトのイベントは 表21.4「デフォルトのイベント」に示してあります。

表21.4 デフォルトのイベント

プロセッサー	カウンター用のデフォルトイベント	詳細
AMD Athlon および AMD64	CPU_CLK_UNHALTED	プロセッサーのクロックは停止していません
AMD Family 10h, AMD Family 11h, AMD Family 12h	CPU_CLK_UNHALTED	プロセッサーのクロックは停止していません
AMD Family 14h, AMD Family 15h	CPU_CLK_UNHALTED	プロセッサーのクロックは停止していません
IBM POWER4	CYCLES	プロセッサーのサイクル
IBM POWER5	CYCLES	プロセッサーのサイクル
IBM PowerPC 970	CYCLES	プロセッサーのサイクル
Intel Core i7	CPU_CLK_UNHALTED	プロセッサーのクロックは停止していません
Intel Nehalem マイクロアーキテクチャー	CPU_CLK_UNHALTED	プロセッサーのクロックは停止していません

プロセッサー	カウンター用のデフォルトイベント	詳細
Intel Pentium 4 (ハイパースレッド有りとなし)	GLOBAL_POWER_EVENTS	プロセッサーが停止していない期間の長さ
Intel Westmere マイクロアーキテクチャー	CPU_CLK_UNHALTED	プロセッサーのクロックは停止していません
TIMER_INT	(なし)	各タイマー割り込みのサンプル

一度に監視できるイベントの数はプロセッサーのカウンターの数で決定されます。しかし、それは１対１の関係ではなく、一部のプロセッサーでは一定のイベントは特定のカウンターにマップしなければなりません。利用可能なカウンターを判定するには、以下のコマンドを実行します。

```
ls -d /dev/oprofile/[0-9]*
```

利用可能なイベントはプロセッサータイプによって変わります。プロファイリング用に利用可能なイベントを判定するには、root として以下のコマンドを実行します。(一覧はシステムのプロセッサータイプに特有のものです)。

```
ophelp
```

注記

OProfile が適正に設定されていないと、**ophelp** は以下のエラーメッセージが出て失敗します。

```
Unable to open cpu_type file for reading
Make sure you have done opcontrol --init
cpu_type 'unset' is not valid
you should upgrade oprofile or force the use of timer mode
```

OProfile の設定については、「レガシーモードを使った OProfile の設定」に記載の指示にしたがってください。

各カウンター用のイベントはコマンドライン、またはグラフィカルインターフェイスで設定できます。グラフィカルインターフェイスに関する情報については、「グラフィカルインターフェース」を参照してください。カウンターが特定のイベントにセットできない場合は、エラーメッセージが表示されます。

コマンドラインを介して各設定可能なカウンター用にイベントをセットするには、**opcontrol** を使用します。

```
opcontrol --event=event-name:sample-rate
```

event-name を **ophelp** からのイベントの実際の名前で置き換え、*sample-rate* をサンプル間のイベントの数で置き換えます。

21.3.2.1. サンプルのレート

デフォルトでは、時間ベースのイベントセットが選択されます。これは 1 つのプロセッサーにつき 10 万クロックサイクルごとにサンプルを 1 つ作成します。タイマー割り込みが使用される場合は、タイマーはそれぞれのレートに設定され、ユーザーは設定できません。**cpu_type** が **timer** でない場合、各イベントは*サンプルのレート*をそれ自身にセットすることができます。サンプルレートとは、各サンプルのスナップ

ショットの間のイベント数です。

カウンター用にイベントをセットする際には、サンプルレートも指定できます。

```
opcontrol --event=event-name:sample-rate
```

sample-rate を再度サンプルを取るまでのイベントの数で置き換えます。カウントが少ないとサンプル取りの回数が増えます。頻繁に発生しないイベントでは、イベントのインスタンスをキャプチャーするにに低いカウントが必要になります。

警告

サンプルレートをセットする際には細心の注意が必要です。サンプルの頻度が高すぎると、システム負荷が大きくなり過ぎてシステムがフリーズしたかのように見えるか、または実際にフリーズする原因になります。

21.3.2.2. ユニットマスク

一部のユーザーパフォーマンス監視イベントでは、イベントをさらに定義するためにユニットマスクも必要になる場合があります。

各イベントのユニットマスクは **ophelp** コマンドで一覧表示されます。各ユニットマスクの値は 16進法形式で一覧になっています。複数のユニットマスクを指定するには、16進法の値をビットサイズの *or* オペレーションの使用と組み合わせる必要があります。

```
opcontrol --event=event-name:sample-rate:unit-mask
```

特定のアーキテクチャーでは、同一の 16 進数値を持つ複数のユニットマスクが存在することができます。この場合、名前のみによる指定が必要になります。

21.3.3. カーネルとユーザースペースのプロファイルの分離

デフォルトでは、カーネルモードとユーザーモードの情報は各イベントで収集されます。特定のカウンターでカーネルモードのイベントを無視するように OProfile を設定するには、以下のコマンドを実行します。

```
opcontrol --event=event-name:sample-rate:unit-mask:0
```

そのカウンターでカーネルモードのプロファイリングを再度スタートするには、以下のコマンドを実行します。

```
opcontrol --event=event-name:sample-rate:unit-mask:1
```

特定のカウンターでユーザーモードのイベントを無視するように OProfile を設定するには、以下のコマンドを実行します。

```
opcontrol --event=event-name:sample-rate:unit-mask:1:0
```

そのカウンターでユーザーモードのプロファイリングを再度スタートするには、以下のコマンドを実行します。

```
opcontrol --event=event-name:sample-rate:unit-mask:1:1
```

OProfile デーモンがサンプルファイルに対してプロファイルデータを書き込む際には、カーネルとライブラリプロファイルデータを別々のサンプルファイルに分離することができます。デーモンがサンプルファイルに書き込む方法を設定するには、root として以下のコマンドを実行します。

```
opcontrol --separate=choice
```

choice は以下のいずれかになります。

- **none** — プロファイルを分離しません (デフォルト)。
- **library** — ライブラリー用にアプリケーション単位のプロファイルを生成します。
- **kernel** — カーネルとカーネルモジュール用にアプリケーション単位のプロファイルを生成します。
- **all** — ライブラリー用にアプリケーション単位のプロファイルを、カーネルとカーネルモジュール用にアプリケーション単位のプロファイルを生成します。

--separate=library が使用される場合は、サンプルファイルの名前には実行可能ファイル名とライブラリ名が含まれます。

注記

これらの設定の変更は OProfile プロファイラーを再起動すると反映されます。

21.4. レガシーモードを使った OProfile の起動と停止

OProfile でシステム監視を開始するには、root として以下のコマンドを実行します。

```
opcontrol --start
```

出力は以下の表示のようになります。

```
Using log file /var/lib/oprofile/oprofiled.log Daemon started. Profiler
running.
```

/root/.oprofile/daemonrc のセッティングが使用されます。

OProfile のデーモン、**oprofiled** が開始されます。これは定期的にサンプルデータを **/var/lib/oprofile/samples/** ディレクトリーに書き込みます。このデーモンのログファイルは **/var/lib/oprofile/oprofiled.log** にあります。

重要

Red Hat Enterprise Linux 7 システムでは、**nmi_watchdog** は **perf** サブシステムで登録をします。このため、**perf** サブシステムはブート時にパフォーマンスカウンターレジスタの制御を獲得し、OProfile の動作を阻止してしまいます。

これを解決するためには、**nmi_watchdog=0** のカーネルパラメーターセットで起動するか、**root** で以下のコマンドを実行し、ランタイム時の**nmi_watchdog** を無効にします。

```
echo 0 > /proc/sys/kernel/nmi_watchdog
```

nmi_watchdog を再度有効にするには、**root** で以下のコマンドを使用します。

```
echo 1 > /proc/sys/kernel/nmi_watchdog
```

プロファイラーを停止するには、root として以下のコマンドを実行します。

```
opcontrol --shutdown
```

21.5. レガシーモードでのデータ保存

特定の時期にサンプルを保存することが役に立つこともあります。たとえば、実行可能ファイルをプロファイルする際に異なる入力データセットを基にした異なるサンプルを収集することが役に立ちます。監視するイベントの数がプロセッサーで利用可能なカウンター数を超える場合は、OProfile の複数回実行によりデータを収集することで、その度にサンプルデータを異なるファイルに保存することができます。

サンプルファイルの現在のセットを保存するには、以下のコマンドを実行します。*name* は現在のセッション用の一意の記述的名前で置き換えます。

```
opcontrol --save=name
```

ディレクトリー **/var/lib/oprofile/samples/*name*/** が作成されて、現在のサンプルファイルがそこにコピーされます。

サンプルデータを保持するセッションディレクトリーを指定するには、**--session-dir** を使用します。これが指定されない場合は、データは現在のパスの **oprofile_data/** ディレクトリーに保存されます。

21.6. データの分析

レガシーモードで **operf** か **opcontrol** のどちらかを使ってプロファイルを収集する際には、同じ OProfile 後処理ツールが使用されます。

デフォルトでは、**operf** は ***current_dir*/oprofile_data** ディレクトリーにプロファイリングデータを保存します。**--session-dir** オプションを使うと、保存場所を変更できます。**opreport** および **opannotate** などの通常のプロファイリング後の分析ツールを使って、プロファイルレポートを生成することができます。これらのツールは、最初に ***current_dir*/oprofile_data** でサンプルを検索します。このディレクトリーが存在しない場合、分析ツールは標準セッションディレクトリー **/var/lib/oprofile/** を使用します。受け取ったサンプルや喪失したサンプルの総数などの統計情報は、***session_dir*/samples/operf.log** ファイルに書き込まれます。

レガシーモード使用時は、OProfile デーモンである **oprofiled** は定期的にサンプルを収集して、それらを**/var/lib/oprofile/samples/** ディレクトリーに書き込みます。そのデータを読み込む前に、root として以下のコマンドを実行して、このディレクトリーにすべてのデータが書き込まれていることを確認します。

```
opcontrol --dump
```

各サンプル名は実行可能ファイルの名前を基にしています。たとえば、Pentium III プロセッサー上の **/bin/bash** 用のデフォルトイベントのサンプルは以下のようになります。

```
\{root\}/bin/bash/\{dep\}/\{root\}/bin/bash/CPU_CLK_UNHALTED.100000
```

収集された後にそのサンプルデータのプロファイルに使用できるツールは、以下のものです。

- **opreport**
- **opannotate**

プロファイルされたバイナリとの併用でこれらのツールを使用すると更なる分析が可能になるレポートを生成できます。

警告

データを分析するためには、プロファイルされる実行可能ファイルがこれらのツールで使用される必要があります。データの収集後に実行可能ファイルの変更が必要な場合はサンプルの作成に使用された実行可能ファイル、およびサンプルファイルをバックアップしておきます。サンプルファイルとバイナリは一致する必要があることに注意してください。これらが一致しないとバックアップの作成はうまくできません。この問題の対処には **oparchive** を使用できます。

各実行可能ファイルのサンプルは、単独のサンプルファイルに書き込まれます。動的にリンクした各ライブラリからのサンプルも、単独のサンプルファイルに書き込まれます。OProfile の実行中に、監視されている実行可能ファイルが変化して、その実行可能ファイルのサンプルファイルが存在する場合は、この既存のサンプルファイルは自動的に削除されます。そのため、既存のサンプルファイルが必要な場合はバックアップが必要です。既存のサンプルの作成に使用された実行可能ファイルを新バージョンで入れ替える前に、この実行可能ファイルのバックアップも必要です。OProfile の分析ツールは、サンプルを作成した実行可能ファイルを分析中に使用します。実行可能ファイルが変化すると、分析ツールは関連サンプルを分析できなくなります。サンプルファイルのバックアップ方法については、「レガシーモードでのデータ保存」を参照してください。

21.6.1. opreport の使用

opreport ツールは、プロファイルされるすべての実行可能ファイルについての概要を提供します。この情報を表示するには、以下を入力します。

```
opreport
```

サンプル出力の一部を以下に示します。

```
Profiling through timer interrupt
TIMER:0|
samples|      %|
------------------
25926 97.5212 no-vmlinux
```

```
359  1.3504 pi
65  0.2445 Xorg
62  0.2332 libvte.so.4.4.0
56  0.2106 libc-2.3.4.so
34  0.1279 libglib-2.0.so.0.400.7
19  0.0715 libXft.so.2.1.2
17  0.0639 bash
8  0.0301 ld-2.3.4.so
8  0.0301 libgdk-x11-2.0.so.0.400.13
6  0.0226 libgobject-2.0.so.0.400.7
5  0.0188 oprofiled
4  0.0150 libpthread-2.3.4.so
4  0.0150 libgtk-x11-2.0.so.0.400.13
3  0.0113 libXrender.so.1.2.2
3  0.0113 du
1  0.0038 libcrypto.so.0.9.7a
1  0.0038 libpam.so.0.77
1  0.0038 libtermcap.so.2.0.8
1  0.0038 libX11.so.6.2
1  0.0038 libgthread-2.0.so.0.400.7
1  0.0038 libwnck-1.so.4.9.0
```

各実行可能ファイルは独立した行で一覧表示されています。最初のコラムは実行可能ファイル用に記録されたサンプルの数です。2 つめのコラムはサンプル合計に対するサンプルの相対的パーセンテージです。3 つめのコラムは実行可能ファイルの名前です。

利用可能なコマンドラインオプションの一覧については **opreport** man ページを参照してください。サンプルの最小数からサンプルの最大数で実行可能ファイルの出力を分類する **-r** オプションなどがあります。また、**-t** オプションや **--threshold** オプションを使うと、**opcontrol** の出力を調整できます。

21.6.2. 単独実行可能ファイルでの opreport の使用

特定の実行可能ファイルに関する詳細なプロファイル情報を取得するには、以下を使用します。

```
opreport mode executable
```

executable を分析する実行可能ファイルへの完全パスで置き換えます。*mode* は以下のいずれかになります。

-l

このオプションは、シンボル別にサンプルデータを一覧表示します。たとえば、

```
~]# opreport -l /lib/tls/libc-version.so
```

上記のコマンドを実行すると、以下が出力されます。

```
samples % symbol name
12 21.4286 __gconv_transform_utf8_internal
5 8.9286 _int_malloc 4 7.1429 malloc
3 5.3571 __i686.get_pc_thunk.bx
3 5.3571 _dl_mcount_wrapper_check
3 5.3571 mbrtowc
3 5.3571 memcpy
2 3.5714 _int_realloc
```

```
2 3.5714 _nl_intern_locale_data
2 3.5714 free
2 3.5714 strcmp
1 1.7857 __ctype_get_mb_cur_max
1 1.7857 __unregister_atfork
1 1.7857 __write_nocancel
1 1.7857 _dl_addr
1 1.7857 _int_free
1 1.7857 _itoa_word
1 1.7857 calc_eclosure_iter
1 1.7857 fopen@@GLIBC_2.1
1 1.7857 getpid
1 1.7857 memmove
1 1.7857 msort_with_tmp
1 1.7857 strcpy
1 1.7857 strlen
1 1.7857 vfprintf
1 1.7857 write
```

最初のコラムはそのシンボルのサンプル数で、2 つめのコラムは実行可能ファイル用の全体のサンプルに対するこのシンボルのサンプルの相対的なパーセンテージです。3 つめのコラムはシンボルの名前です。

サンプルの最大数から最小数 (逆順) に出力を配列するには、**-l** オプションを**-r** オプションと併用して使用します。

-i *symbol-name*

シンボル名に固有のサンプルデータを一覧表示します。たとえば、

```
~]# opreport -l -i __gconv_transform_utf8_internal
/lib/tls/libc-version.so
```

上記のコマンドを実行すると、以下の出力が返されます。

```
samples % symbol name
12 100.000 __gconv_transform_utf8_internal
```

最初の行はシンボルと実行可能ファイルの組み合わせの要約です。

最初のコラムはメモリーシンボルのサンプルの数です。2 つめのコラムはシンボルのサンプル合計数に対するメモリーアドレスのサンプルの相対的なパーセンテージです。3 つめのコラムはシンボル名です。

-d

これは **-l** よりも詳細に、シンボル別のサンプルデータを一覧表示します。たとえば、

```
~]# opreport -d -i __gconv_transform_utf8_internal
/lib/tls/libc-version.so
```

上記のコマンドを実行すると、以下の出力が返されます。

```
vma samples % symbol name
00a98640 12 100.000 __gconv_transform_utf8_internal
00a98640 1 8.3333
```

```
00a9868c 2 16.6667
00a9869a 1 8.3333
00a986c1 1 8.3333
00a98720 1 8.3333
00a98749 1 8.3333
00a98753 1 8.3333
00a98789 1 8.3333
00a98864 1 8.3333
00a98869 1 8.3333
00a98b08 1 8.3333
```

データは **-l** オプションと同じですが、それ以外に各シンボル毎に使用される各仮想メモリーアドレスが示してあります。各仮想メモリーアドレスについては、そのサンプルの数とシンボル用のサンプル合計数に対するメモリーアドレスサンプルのパーセンテージが表示されています。

-e ***symbol-name*...**

このオプションを使うと、出力から特定のシンボルを除外することができます。*symbol-name* を除外したいシンボルのコンマ区切りで置き換えます。

session:*name*

ここではセッションへの完全パス、つまり、**/var/lib/oprofile/samples/** ディレクトリーに相対的となるディレクトリー、または **operf** を使用している場合は、**./oprofile_data/samples/** に相対的となるディレクトリーを指定することができます。

21.6.3. モジュールについてより詳細な出力を取得する

OProfile は、システムワイドをベースにマシン上で実行中のカーネルとユーザースペースコードのデータを収集します。しかし、モジュールがカーネルに読み込まれると、カーネルモジュールの発生元に関する情報は消失します。モジュールは、起動時に **initrd** ファイルから、各種カーネルモジュールのあるディレクトリーから、またはローカルで作成されたカーネルモジュールから来た可能性があります。このため、OProfile は 1 つのモジュールのサンプルを記録する際に、実行可能ファイルのモジュールのサンプルを root ディレクトリー内に一覧表示しますが、この場所はモジュールの実際のコードがある場所である可能性は低くなります。分析ツールが適切な実行可能ファイルを確実に取得するには、いくつかの手順を実行する必要があります。

モジュールのアクションについてより詳細な見通しを得るには、モジュールを "unstripped" 状態にする (つまり、カスタムビルドからインストールした状態にする) か、カーネル用に *debuginfo* パッケージをインストールする必要があります。

実行中のカーネルを判定するには、**uname -a** コマンドを使用します。そして適切な*debuginfo* パッケージを取得し、マシンにインストールします。

それから以前の実行によるサンプルをクリアするために、以下のコマンドを入力します。

```
opcontrol --reset
```

たとえば Westmere プロセッサーを持つマシンで監視プロセスを開始するには、以下のコマンドを実行します。

```
~]# opcontrol --setup --vmlinux=/usr/lib/debug/lib/modules/`uname -
r`/vmlinux --event=CPU_CLK_UNHALTED:500000
```

その後、たとえば ext4 モジュール用の詳細情報は以下のように取得できます。

```
~]# opreport /ext4 -l --image-path /lib/modules/`uname -r`/kernel
CPU: Intel Westmere microarchitecture, speed 2.667e+06 MHz (estimated)
Counted CPU_CLK_UNHALTED events (Clock cycles when not halted) with a
unit mask of 0x00 (No unit mask) count 500000
warning: could not check that the binary file /lib/modules/2.6.32-
191.el6.x86_64/kernel/fs/ext4/ext4.ko has not been modified since the
profile was taken. Results may be inaccurate.
samples  %        symbol name
1622      9.8381  ext4_iget
1591      9.6500  ext4_find_entry
1231      7.4665  __ext4_get_inode_loc
783       4.7492  ext4_ext_get_blocks
752       4.5612  ext4_check_dir_entry
644       3.9061  ext4_mark_iloc_dirty
583       3.5361  ext4_get_blocks
583       3.5361  ext4_xattr_get
479       2.9053  ext4_htree_store_dirent
469       2.8447  ext4_get_group_desc
414       2.5111  ext4_dx_find_entry
```

21.6.4. opannotate の使用

opannotate ツールは、特定の指示用のサンプルをソースコード内の該当する行に一致させるよう試みます。その結果生成されたファイルは、左側にある行のサンプルを持つはずです。これはまた、各関数の先頭にコメントを入れて関数の合計サンプル一覧表示します。

このユーティリティーが機能するには、実行可能ファイル用の適切な *debuginfo* パッケージをシステム上にインストールする必要があります。Red Hat Enterprise Linux では、*debuginfo* パッケージは、実行可能ファイルを含む対応パッケージと自動的にインストールされません。これらは別個に入手してインストールする必要があります。

opannotate 用の一般的な構文は次のようになります。

```
opannotate --search-dirs src-dir --source executable
```

これらのコマンドラインオプションは必須になります。*src-dir* をソースコードを格納しているディレクトリーへのパスで置き換え、分析する実行可能ファイルを指定します。他のコマンドラインオプションの一覧については、**opannotate** man ページを参照してください。

21.7. /dev/oprofile/ を理解する

OProfile をレガシーモードで使用する際は、**/dev/oprofile/** ディレクトリーは OProfile 用のファイルシステムの保存に使用されます。一方で、**operf** は **/dev/oprofile/** を必要としません。**cat** コマンドを使うと、このファイルシステム内の仮想ファイルの値を表示することができます。たとえば、以下のコマンドは OProfile が検出したプロセッサーのタイプを表示します。

```
cat /dev/oprofile/cpu_type
```

/dev/oprofile/ には、各カウンター用のディレクトリーが存在します。たとえば、2 つのカウンターがある場合は、**/dev/oprofile/0/** ディレクトリーと **dev/oprofile/1/** ディレクトリーが存在します。

カウンター用の各ディレクトリーには以下のファイルが含まれています。

- **count** — サンプル間の間隔
- **enabled** — 0 の場合は、カウンターはオフでありサンプルは収集されません。1 の場合は、カウンターはオンとなりサンプルは収集されます。
- **event** — 監視対象のイベント
- **extra** — 監視するイベントをさらに指定するために Nehalem プロセッサーを持つマシン上で使用されます。
- **kernel** — 0 の場合は、プロセッサーがカーネルスペース内にある時にこのカウンターイベント用のサンプルは収集されません。1 の場合は、プロセッサーがカーネルスペース内にあってもサンプルは収集されます。
- **unit_mask** — カウンター用に有効になっているユニットマスクを定義します。
- **user** — 0 の場合、プロセッサーがユーザースペース内にある時にカウンターイベント用のサンプルは収集されません。1 の場合、プロセッサーがユーザースペース内にあってもサンプルは収集されます。

これらのファイルの値は **cat** コマンドで取り込むことができます。例を示します。

```
cat /dev/oprofile/0/count
```

21.8. 使用例

OProfile は開発者がアプリケーションのパフォーマンスを分析するために使用できるほか、システム管理者がシステム分析を実行するためにも使用できます。以下のような例があります。

- **システム上でどのアプリケーションとサービスが最も使用されるのかを判定— opreport** を使用すると、アプリケーションまたはサービスが使用するプロセッサータイムを判定できます。システムが複数のサービスに使用されていてパフォーマンスが低い場合は、プロセッサータイムの消費が一番高いサービスを専用システムに移動することができます。
- **プロセッサーの使用レベルを判定 — CPU_CLK_UNHALTED** イベントを監視することにより任意の期間内でのプロセッサー負荷を判定できます。このデータはその後、追加のプロセッサーまたはより高速なプロセッサーがシステムのパフォーマンスを向上するかどうかの判定に使用できます。

21.9. Java 対応の OProfile サポート

OProfile の使用により JVM (Java 仮想マシン) の動的にコンパイルしたコード (別名 "just-in-time" または JIT コード) をプロファイルできるようになります。Red Hat Enterprise Linux 7 内の OProfile には、JVMTI (JVM ツールインターフェイス) のエージェントライブラリのビルトインサポートが含まれています。これは Java 1.5 およびそれ以降をサポートします。

21.9.1. Java コードのプロファイル作成

JVMTI エージェントを持つ Java 仮想マシンから JIT コードをプロファイルするには、以下を JVM スタートアップパラメーターに追加します。

```
-agentlib:jvmti_oprofile
```

jvmti_oprofile は、プロファイルエージェントへのパスになります。64 ビット JVM では、パスは以下のようになります。

```
-agentlib:/usr/lib64/oprofile/libjvmti_oprofile.so
```

現時点で追加できるコマンドラインオプションは、 **--debug** です。これで、デバッグモードが有効になります。

注記

OProfile で JIT コードをプロファイルするには、システム上に *oprofile-jit* パッケージをインストールする必要があります。このパッケージがあると、メソッドレベルでの情報が表示できるようになります。

使用している JVM によっては、JVM 用に **debuginfo** パッケージをインストールする必要がある場合があります。OpenJDK では、このパッケージは必須になります。Oracle JDK 用の debuginfo パッケージはありません。non-debug パッケージとデバッグ情報パッケージを同期させておくには、**yum-plugin-auto-update-debug-info** プラグインのインストールも必要になります。このプラグインは、対応する更新用にデバッグ情報リポジトリーを検索します。

セットアップが終了したら、上記セクションで説明された標準プロファイリングと分析ツールを適用できます。

OProfile での Java サポートに関する詳細は、「その他のリソース」からリンクされている OProfile マニュアルを参照してください。

21.10. グラフィカルインターフェース

一部の OProfile 設定項目はグラフィカルインターフェイスで設定できます。開始するには、シェルプロンプトで root として **oprof_start** コマンドを実行します。グラフィカルインターフェイスを使用するには、**oprofile-gui** パッケージをインストールする必要があります。

いずれかのオプションを変更した後には、**保存して終了** ボタンをクリックして変更を保存します。設定は **/root/.oprofile/daemonrc** に書き込まれて、アプリケーションは終了します。

注記

アプリケーションの終了は、OProfile のサンプリングを停止するものではありません。

「監視するイベントのセッティング」で説明してあるように、**Setup** タブでプロセッサカウンター用にイベントをセットするには、プルダウンメニューからカウンターを選択してその一覧からイベントを選択します。するとそのイベントの簡単な説明が一覧の下のテキストボックスに出てきます。特定のカウンターおよび特定のアーキテクチャー用に利用可能なイベントのみが表示されます。このグラフィカルインターフェイスはプロファイラーが実行中であるかどうかとその簡単な統計も表示します。

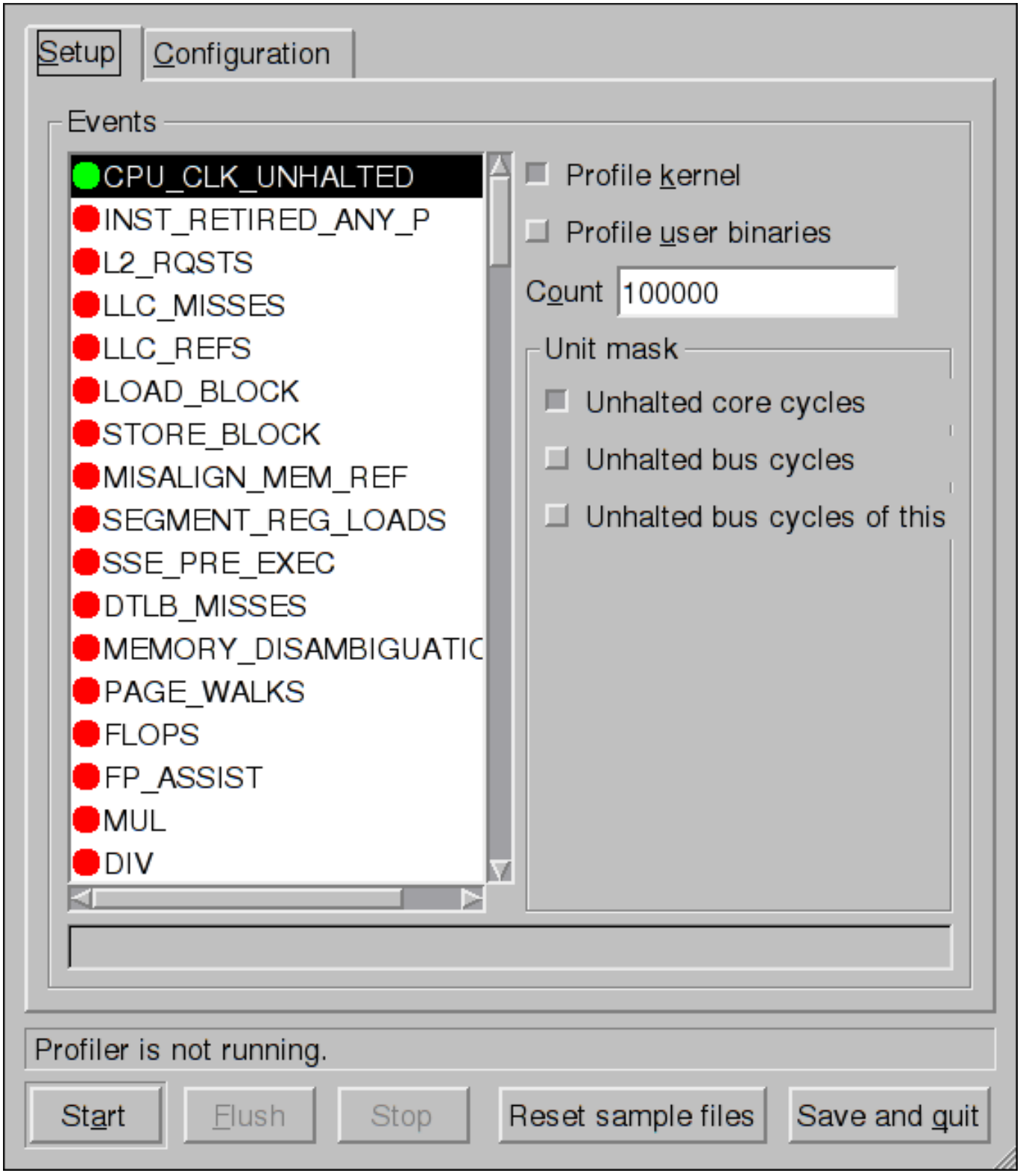


図21.1 OProfile のセットアップ

「カーネルとユーザースペースのプロファイルの分離」の説明にあるように、タブの右側で**Profile kernel** オプションを選択して、現在選択されているイベントのカーネルモードのイベントをカウントします。このオプションが選択されていないと、カーネル用のサンプルは収集されません。

「カーネルとユーザースペースのプロファイルの分離」の説明にあるように、現在選択されているイベント用にユーザーモードでイベントをカウントするには **Profile user binaries** オプションを選択します。このオプションが選択されていないと、ユーザーアプリケーション用のサンプルは収集されません。

「サンプルのレート」の説明にあるように、現在選択されているイベント用にサンプルレートをセットするには **Count** テキストフィールドを使用します。

「ユニットマスク」の説明にあるように、現在選択されているイベント用にいずれかのユニットマスクが利用可能である場合は、それは **Setup** タブの右側にある **Unit Masks** エリアに表示されます。イベント用に有効にすべきユニットマスクの横のチェックボックスを選択します。

Configuration タブでカーネルをプロファイルするには、**Kernel image file** テキストフィールド内に監視するカーネルの名前と **vmlinux** ファイルの場所を入力します。カーネルを監視しないように OProfile を設定するには、**No kernel image** を選択します。

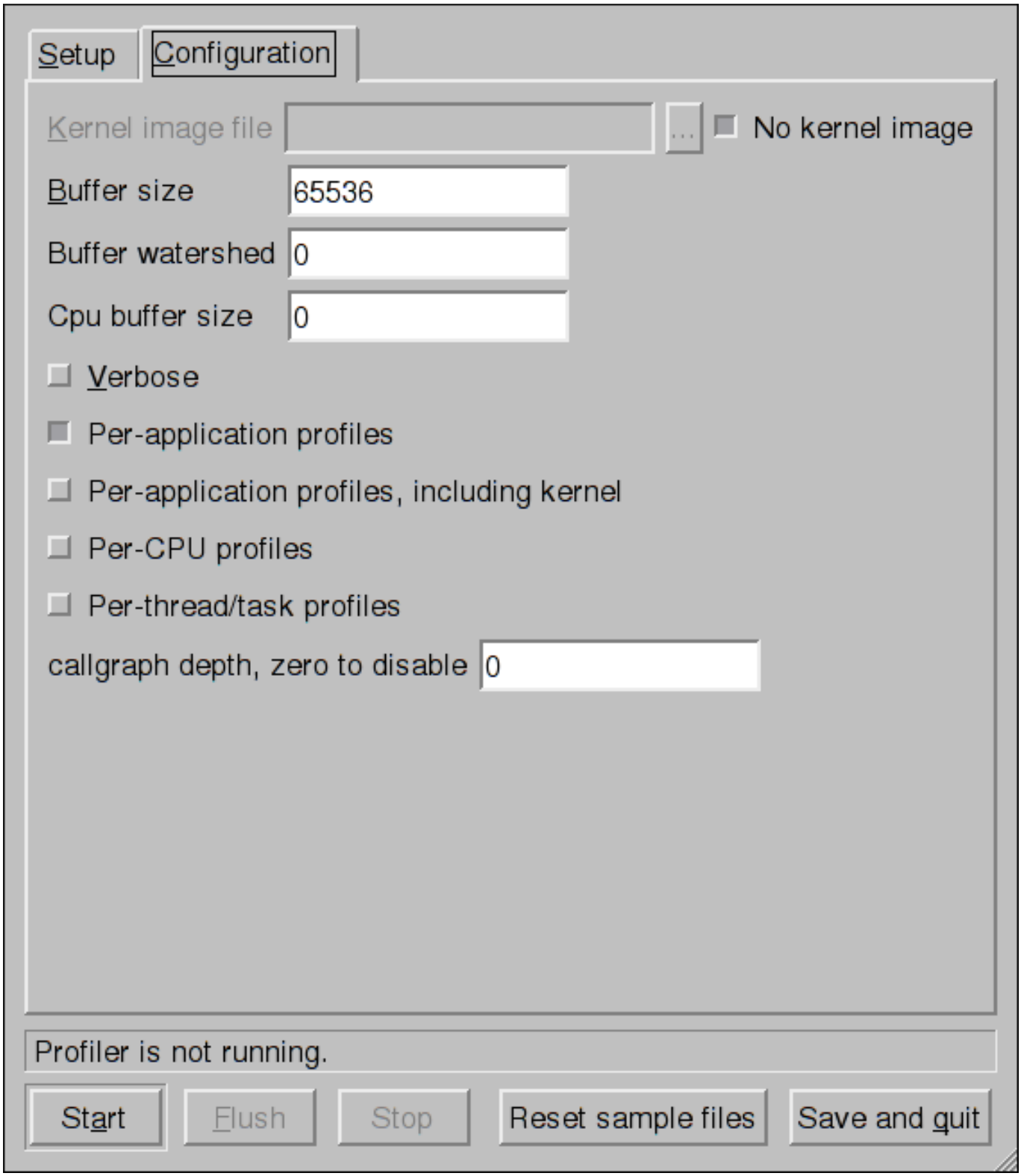


図21.2 OProfile の設定

Verbose オプションが選択されていると、**oprofiled** デーモンログにより詳細な情報が含まれます。

Per-application profiles が選択されていると、OProfile はライブラリ用にアプリケーション単位のプロファイルを生成します。これは、**opcontrol --separate=library** コマンドと同じことになります。**Per-application profiles, including kernel** が選択されていると、「カーネルとユーザースペースのプロファイルの分離」で説明してあるように OProfile はカーネルとカーネルモジュール用にアプリケーション単位のプロファイルを生成します。これは **opcontrol --separate=kernel** コマンドと同じことになります。

「データの分析」で説明してあるように、データがサンプルファイルに書き込まれるように強制するには、**Flush** ボタンをクリックします。これは、**opcontrol --dump** コマンドと同じことになります。

グラフィカルインターフェイスから OProfile を開始するには、**Start** をクリックします。プロファイラーを停止するには、**Stop** をクリックします。アプリケーションの終了は OProfile がサンプル作成を停止することにはなりません。

21.11. OProfile と SystemTap

SystemTap は、追跡と探索のツールとしてユーザーがオペレーティングシステムの動作を詳細に渡って学習し監視できるようにしてくれます。これは **netstat**、**ps**、**top**、および **iostat** などのツールの出力に似た情報を提供します。しかし SystemTap は、収集した情報に対してより多くのフィルタリングと分析オプションを提供するように設計されています。

OProfile の使用は、コードの特定エリアでプロセッサーが時間を費やす場所と理由についてデータを収集するケースで推奨されますが、OProfile はプロセッサーがアイドル状態を継続する理由を見つける目的ではあまり役に立ちません。

コードの特定の場所を計測するには、SystemTap が適しています。SystemTap を使用すると、インスツルメント化されたコードを停止したり再開したりせずにコードのインスツルメント化が実行できるので、カーネルとデーモンのインスツルメント化で特に役に立ちます。

SystemTap に関する詳細情報は、「役立つ Web サイト」の関連ドキュメントを参照してください。

21.12. その他のリソース

本章は、OProfile の設定方法と使用方法についてのみフォーカスしています。その他の情報については、以下のリソースを参照してください。

21.12.1. インストール済みドキュメント

- **/usr/share/doc/oprofile-*version*/oprofile.html** — 『OProfile マニュアル』
- **oprofile** man ページ — **opcontrol**、**opreport**、**opannotate**、および **ophelp** を説明しています
- **operf** man ページ

21.12.2. 役立つ Web サイト

- http://oprofile.sourceforge.net/ — 最新のドキュメント、メーリングリスト、IRC チャンネル、その他を含んでいます。

- SystemTap Beginners Guide — SystemTap を使用して Red Hat Enterprise Linux の各種サブシステムを監視するための基本的な手順を詳細に説明しています。

パート VI. カーネル、モジュール、およびドライバーの設定

このパートでは、管理者がカーネルをカスタマイズする際に役立つ様々なツールを取り上げます。

第22章 GRUB 2 ブートローダーの操作

Red Hat Enterprise Linux 7 には、GNU GRand Unified Boot loader (GRUB) バージョン 2 のブートローダーが同梱されており、システム起動時にユーザーが読み込まれるオペレーティングシステムやカーネルを選択できるようになっています。GRUB 2 では、ユーザーがカーネルに引数を渡すこともできます。

22.1. GRUB 2 ブートローダーの選択

GRUB 2 は、従来の BIOS ベースのマシンでは **/boot/grub2/grub.cfg** ファイルから、UEFI マシンの場合は **/boot/efi/EFI/redhat/grub.cfg** ファイルから設定を読み取ります。これらのファイルにはメニュー情報が含まれていますが、**/boot/** ディレクトリーにある Linux カーネル、**/etc/grub.d/** にあるテンプレートファイル、および **/etc/default/grub** ファイル内のカスタム設定に基づいて **/usr/sbin/grub2-mkconfig** ユーティリティーがこの情報を生成しているので、これは編集すべきものではありません。このため、手動で編集すると、変更は更新中に失われます。GRUB 2 設定ファイルの **grub.cfg** は、新規カーネルがインストールされると、常に自動的に更新されます。**/etc/default/grub** への変更は、**grub.cfg** ファイルの再構築が必要になることに注意してください。GRUB 2 設定ファイルを手動で更新するには、以下のように **grub2-mkconfig -o** コマンドを使用してください。

- BIOS ベースのマシンでは、**root** で以下のコマンド発行します。

```
~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

- UEFI ベースのマシンでは、**root** で以下のコマンド発行します。

```
~]# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

grub.cfg 設定ファイルには、多くのコードスニペットや指示文に加えて、1 つまたは複数の **menuentry** ブロックが含まれています。これらはそれぞれ、単一の GRUB 2 ブートメニューエントリーを表しています。これらのブロックは、常に **menuentry** キーワードで始まり、それにタイトル、オプションリスト、および中括弧が続きます。中括弧の中身は、インデントにします。たとえば、以下は Linux カーネル 3.8.0-0.40.el7.x86_64 Red Hat Enterprise Linux 7 の **menuentry** ブロック例です。

```
menuentry 'Red Hat Enterprise Linux Client' --class red --class gnu-linux
--class gnu --class os $menuentry_id_option 'gnulinux-simple-c60731dc-
9046-4000-9182-64bdcce08616' {
        load_video
        set gfxpayload=keep
        insmod gzio
        insmod part_msdos
        insmod xfs
        set root='hd0,msdos1'
        if [ x$feature_platform_search_hint = xy ]; then
          search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1
--hint-efi=hd0,msdos1 --hint-baremetal=ahci0,msdos1 --hint='hd0,msdos1'
19d9e294-65f8-4e37-8e73-d41d6daa6e58
        else
          search --no-floppy --fs-uuid --set=root 19d9e294-65f8-4e37-
8e73-d41d6daa6e58
        fi
        echo    'Loading Linux 3.8.0-0.40.el7.x86_64 ...'
        linux   /vmlinuz-3.8.0-0.40.el7.x86_64 root=/dev/mapper/rhel-
root ro rd.md=0 rd.dm=0 rd.lvm.lv=rhel/swap crashkernel=auto rd.luks=0
```

```
vconsole.keymap=us rd.lvm.lv=rhel/root rhgb quiet
        echo    'Loading initial ramdisk ...'
        initrd  /initramfs-3.8.0-0.40.el7.x86_64.img
}
```

インストール済み Linux カーネルを表す各 **menuentry** ブロックには、**linux** (UEFI システムでは **linuxefi**) と **initrd** 指示文が含まれ、それぞれの後にカーネルへのパスと**initramfs** イメージへのパスが続きます。別個の **/boot** パーティションが作成されている場合は、カーネルと**initramfs** イメージへのパスは、**/boot** に相対的なものになります。上記の例では、**initrd /initramfs-3.8.0-0.40.el7.x86_64.img** の行は、root ファイルシステムのマウント時に**initramfs** イメージが実際には **/boot/initramfs-3.8.0-0.40.el7.x86_64.img** にあり、カーネルパスも同様になっていることを意味します。

各 **menuentry** ブロックの **linux /vmlinuz-kernel_version** 行にあるカーネルのバージョン番号は、**initrd /initramfs-kernel_version.img** 行の **initramfs** イメージのバージョン番号に適合する必要があります。初期 RAM ディスクイメージの検証方法についての情報は、「初期 RAM ディスクイメージの検証」 を参照してください。

注記

menuentry では、**initrd** 指示文は同じカーネルバージョンに対応する**initramfs** ファイルの位置 (別のパーティションの場合、**/boot/** ディレクトリーに対して相対的) を指している必要があります。初期 RAM ディスクイメージを作成した以前のツール、**mkinitrd** が **initrd** と呼ばれるファイルを作成したために、この指示文は **initrd** と呼ばれます。**grub.conf** 指示文は、他のツールとの互換性を維持するために **initrd** のまま残されています。初期 RAM ディスクイメージを作成するための **dracut** ユーティリティーを使用したシステムのファイル命名の規則名は、**initramfs-*kernel_version*.img** となります。

Dracut の使用に関する情報は、「初期 RAM ディスクイメージの検証」 を参照してください。

22.2. GRUB 2 メニューのカスタマイズ

GRUB 2 スクリプトはユーザーのコンピューターを検索して、スクリプトが見つけたオペレーティングシステムに基づくブートメニューを構築します。最新のシステムブートオプションを反映させるために、カーネルが更新されるか新規カーネルが追加されると、ブートメニューは自動的に再構築されます。

しかし、特定のエントリーを含むメニューの構築や、エントリーの特定の順番をユーザーが希望する場合もあります。GRUB 2 では基本的なブートメニューのカスタマイズが可能で、画面に表示されるものをユーザーが制御できます。

GRUB 2 は、メニューの構築に一連のスクリプトを使用します。これらは、**/etc/grub.d/** ディレクトリーに格納されており、以下のファイルが含まれます。

- **00_header** — **/etc/default/grub** ファイルから GRUB 2 設定を読み込みます。
- **01_users** — ブートローダーのパスワードが**kickstart** に割り当てられている場合にのみ、作成されます。
- **10_linux** — Red Hat Enterprise Linux のデフォルトのパーティションでカーネルを見つけます。
- **30_os-prober** — 別のパーティションで見つかったオペレーティングシステム用にエントリーを構築します。
- **40_custom** — 追加のメニューエントリー作成に使用可能なテンプレートです。

/etc/grub.d/ ディレクトリーからのスクリプトはアルファベット順に読み取られるので、名前を変更して特定のメニューエントリーの起動順を変更することができます。

重要

/etc/default/grub ファイルで **GRUB_TIMEOUT** キーが **0** に設定されていると、GRUB 2 はシステム起動時に起動可能なカーネル一覧を表示しません。起動時にこの一覧を表示するようにするには、BIOS 情報が表示されている間に英数字のいずれかのキーを押し続けます。すると、GRUB 2 は GRUB メニューを表示します。

22.2.1. デフォルトのブートエントリーの変更

デフォルトでは、**saved** 値が **/etc/default/grub** ファイルの **GRUB_DEFAULT** キーに使われます。これは、GRUB 2 環境ファイルの **saved_entry** 指示文で指定しているカーネルを GRUB 2 が 読み込むように指示します。この環境ファイルは、**/boot/grub2/grubenv** にあります。**grub2-set-default** コマンドを使って別の GRUB 記録をデフォルトにすることもできます。こうすることで、GRUB 2 環境ファイルが更新されます。デフォルトでは、**saved_entry** 値は直近にインストールされたカーネルの名前に設定されます。

GRUB 2 では、オペレーティングシステムが読み込まれるデフォルトの順番を **saved_entry** 指示文のキーに数値を使うことで変更できます。どのオペレーティングシステムを最初に読み込むかを指定するには、その数字を **grub2-set-default** コマンドに渡します。例を示します。

```
~]# grub2-set-default 2
```

一覧内でのメニューエントリーの場所は、ゼロで始まる数字で示されることに注意してください。このため、上記の例では、3 番目のエントリーが読み込まれます。この値は、次回にインストールされるカーネルの名前で上書きされます。

システムが常に特定のカーネルを使用するようにするには、**/etc/default/grub** ファイルの **GRUB_DEFAULT** 指示文にカーネル名をキーとして使用します。利用可能なカーネル名を一覧表示するには、**root** で以下のコマンドを実行します。

```
~]# awk -F\' '$1=="menuentry " {print $2}' /etc/grub2.cfg
```

/etc/default/grub への変更は、以下のように **grub.cfg** ファイルの再構築を必要とします。

- BIOS ベースのマシンでは、**root** で以下のコマンド発行します。

  ```
  ~]# grub2-mkconfig -o /boot/grub2/grub.cfg
  ```

- UEFI ベースのマシンでは、**root** で以下のコマンド発行します。

  ```
  ~]# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
  ```

22.2.2. エントリーの編集

カーネルパラメーター

GRUB 2 ブートメニューが表示される際に、単一のブートプロセス中のみにカーネルパラメーターを使用するには、カーソルを起動するカーネルに移動し、**e** キーを押してカーネルパラメーターを編集します。たとえば、システムを緊急モードで実行するには、**emergency** パラメーターを **linux16** 行の最後に追加します。

```
linux      /vmlinuz-3.10.0-0.rc4.59.el7.x86_64 root=/dev/mapper/rhel-
root ro rd.md=0 rd.dm=0 rd.lvm.lv=rhel/swap crashkernel=auto rd.luks=0
vconsole.keymap=us rd.lvm.lv=rhel/root rhgb quiet emergency
```

この設定は永続的なものではなく、単一ブートにのみ適用されます。この設定を永続的にするには、**/etc/default/grub** ファイルの **GRUB_CMDLINE_LINUX** キーの値を編集します。たとえば、ブート時に毎回緊急モードを有効にするには、以下を使用します。

```
GRUB_CMDLINE_LINUX="emergency"
```

GRUB 2 ブートメニューにパラメーターを追加する際と同様に、**GRUB_CMDLINE_LINUX** キーには複数のパラメーターを指定できることに注意してください。たとえば、

```
GRUB_CMDLINE_LINUX="console=tty0 console=ttyS0,9600n8"
```

ここでの **console=tty0** は最初の仮想ターミナルで、**console=ttyS0** は使用するシリアルターミナルになります。

/etc/default/grub への変更は、以下のように **grub.cfg** ファイルの再構築を必要とします。

- BIOS ベースのマシンでは、**root** で以下のコマンド発行します。

```
~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

- UEFI ベースのマシンでは、**root** で以下のコマンド発行します。

```
~]# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

22.2.3. 新規エントリーの追加

grub2-mkconfig コマンドを実行すると、GRUB 2 は **/etc/grub.d/** ディレクトリーにあるファイルに基づいて Linux カーネルと他のオペレーティングシステムを探します。**/etc/grub.d/10_linux** スクリプトは、同一パーティション上でインストール済み Linux カーネルを検索します。**/etc/grub.d/30_os-prober** スクリプトは、他のオペレーティングシステムを検索します。また、カーネル更新時には、メニューエントリーが自動的にブートメニューに追加されます。

/etc/grub.d/ ディレクトリー内にある **40_custom** ファイルはカスタムエントリー用のテンプレートで、以下のようになっています。

```
#!/bin/sh
exec tail -n +3 $0
# This file provides an easy way to add custom menu entries.  Simply
type the
# menu entries you want to add after this comment.  Be careful not to
change
# the 'exec tail' line above.
```

このファイルは、編集またはコピーが可能です。有効なメニューエントリーには、少なくとも以下のものを含める必要があることに注意してください。

```
menuentry "<Title>"{
<Data>
}
```

22.2.4. カスタムメニューの作成

メニューエントリーの自動更新を希望しない場合は、カスタムメニューを作成できます。

重要

次に進む前に、後で変更を戻す必要に迫られた場合に備えて、**/etc/grub.d/** ディレクトリーのコンテンツのバックアップを作成してください。

注記

/etc/default/grub ファイルを修正しても、カスタムメニューの作成には影響がないことに注意してください。

1. BIOS ベースのマシンでは **/boot/grub2/grub.cfg** のコンテンツを、UEFI マシンでは **/boot/efi/EFI/redhat/grub.cfg** のコンテンツをコピーします。**grug.cfg** のコンテンツを既存のヘッダー行の下で **/etc/grub.d/40_custom** ファイルに置きます。**40_custom** スクリプトの実行可能な部分は、維持される必要があります。

2. **/etc/grub.d/40_custom** ファイルに置かれたコンテンツからカスタムメニューの作成に必要となるのは、**menuentry** ブロックのみです。**/boot/grub2/grub.cfg** ファイルと **/boot/efi/EFI/redhat/grub.cfg** ファイルには関数の仕様と他のコンテンツが **menuentry** ブロックの上下に含まれる可能性があります。ここまでの手順でこれら不要な行を **40_custom** ファイルに置いた場合は、消去してください。

   以下は、カスタムの **40_custom** スクリプトの例です。

```
#!/bin/sh
exec tail -n +3 $0
# This file provides an easy way to add custom menu entries.
Simply type the
# menu entries you want to add after this comment.  Be careful not
to change
# the 'exec tail' line above.

menuentry 'First custom entry' --class red --class gnu-linux --
class gnu --class os $menuentry_id_option 'gnulinux-3.10.0-
67.el7.x86_64-advanced-32782dd0-4b47-4d56-a740-2076ab5e5976' {
        load_video
        set gfxpayload=keep
        insmod gzio
        insmod part_msdos
        insmod xfs
        set root='hd0,msdos1'
        if [ x$feature_platform_search_hint = xy ]; then
          search --no-floppy --fs-uuid --set=root --
hint='hd0,msdos1'  7885bba1-8aa7-4e5d-a7ad-821f4f52170a
```

```
        else
          search --no-floppy --fs-uuid --set=root
7885bba1-8aa7-4e5d-a7ad-821f4f52170a
        fi
        linux16 /vmlinuz-3.10.0-67.el7.x86_64
root=/dev/mapper/rhel-root ro rd.lvm.lv=rhel/root
vconsole.font=latarcyrheb-sun16 rd.lvm.lv=rhel/swap
vconsole.keymap=us crashkernel=auto rhgb quiet LANG=en_US.UTF-8
        initrd16 /initramfs-3.10.0-67.el7.x86_64.img
}
menuentry 'Second custom entry' --class red --class gnu-linux --
class gnu --class os $menuentry_id_option 'gnulinux-0-rescue-
07f43f20a54c4ce8ada8b70d33fd001c-advanced-32782dd0-4b47-4d56-a740-
2076ab5e5976' {
        load_video
        insmod gzio
        insmod part_msdos
        insmod xfs
        set root='hd0,msdos1'
        if [ x$feature_platform_search_hint = xy ]; then
          search --no-floppy --fs-uuid --set=root --
hint='hd0,msdos1'  7885bba1-8aa7-4e5d-a7ad-821f4f52170a
        else
          search --no-floppy --fs-uuid --set=root
7885bba1-8aa7-4e5d-a7ad-821f4f52170a
        fi
        linux16 /vmlinuz-0-rescue-07f43f20a54c4ce8ada8b70d33fd001c
root=/dev/mapper/rhel-root ro rd.lvm.lv=rhel/root
vconsole.font=latarcyrheb-sun16 rd.lvm.lv=rhel/swap
vconsole.keymap=us crashkernel=auto rhgb quiet
        initrd16 /initramfs-0-
rescue-07f43f20a54c4ce8ada8b70d33fd001c.img
}
```

3. 以下を除いて、すべてのファイルを **/etc/grub.d** ディレクトリーから削除します。

   - **00_header**
   - **40_custom**
   - **README**

   別の方法では、**/etc/grub2.d/** ディレクトリーのファイルを維持したい場合、**chmod a-x <file_name>** コマンドを実行してこれらのファイルを実行可能ファイルにします。

4. **40_custom** ファイル内のメニューエントリーを希望に合わせて編集、追加、削除します。

5. 以下のように **grub2-mkconfig -o** コマンドを実行して、**grub.cfg** ファイルを更新します。

   - BIOS ベースのマシンでは、**root** で以下のコマンド発行します。

     ```
     ~]# grub2-mkconfig -o /boot/grub2/grub.cfg
     ```

   - UEFI ベースのマシンでは、**root** で以下のコマンド発行します。

     ```
     ~]# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
     ```

22.3. GRUB 2 パスワードの保護

GRUB 2 は、GRUB 2 テンプレートファイルで暗号化されていないパスワードを使用する基本的なパスワード保護をサポートします。この機能を有効にするには、保護されたエントリーとパスワードにアクセスできるスーパーユーザーを指定します。他のユーザーもこれらのエントリーにアクセスできるように指定することが可能です。ユーザーを定義することはできますが、メニューエントリーを指定することはできません。「GRUB 2 更新後の設定の維持」にあるように **--users** をメニューエントリーに追加することで、メニューエントリーは起動時のパスワード保護が可能になります。スーパーユーザーを設定することで (**00_header** または **01_users** ファイルで通常は実行) すべてのメニューエントリーがレコード変更についてパスワード保護をかけることができます。

22.3.1. ユーザーおよびパスワード保護の設定、メニューエントリーの指定

1. スーパーユーザーを指定するには、以下の行を **/etc/grub.d/00_header** ファイルに追加します。ここでの **john** はスーパーユーザーと指定するユーザー名に、**johnspassword** はスーパーユーザーのパスワードになります。

```
cat <<EOF
set superusers="john"
password john johnspassword
EOF
```

2. 他のユーザーがメニューエントリーにアクセスできるようにするには、ユーザーごとに追加の行を **/etc/grub.d/00_header** ファイルの最後に追加します。

```
cat <<EOF
set superusers="john"
password john johnspassword
password jane janespassword
EOF
```

3. ユーザーとパスワードを設定したら、パスワード保護をかけるメニューエントリーを指定します。これを指定しないと、デフォルトではすべてのメニューエントリーがパスワードで保護されます。メニューエントリーのフォーマットについては「新規エントリーの追加」を、保護するメニューエントリーの指定例については「GRUB 2 更新後の設定の維持」を参照してください。

4. 以下のように **grub2-mkconfig -o** コマンドを実行して、**grub.cfg** ファイルを更新します。

   - BIOS ベースのマシンでは、**root** で以下のコマンド発行します。

```
~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

   - UEFI ベースのマシンでは、**root** で以下のコマンド発行します。

```
~]# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

22.3.2. GRUB 2 更新後の設定の維持

GRUB 2 更新後も変更を維持したい場合は、以下と同様の形で **/etc/grub.d/01_users** ファイルに行を追加します。

```
set superusers="john"
password john johnspassword
```

```
password jane janespassword

menuentry 'Red Hat Enterprise Linux Client' {
set root=(hd0,msdos1)
linux   /vmlinuz
}

menuentry 'Fedora' --users jane {
set root=(hd0,msdos2)
linux   /vmlinuz
}
```

上記の例では、**john** が **superuser** なので、どのメニューエントリーも起動でき、GRUB 2 コマンドラインの使用とブート中の GRUB 2 メニューアイテムの編集が可能です。このケースでは、**john** は Red Hat Enterprise Linux クライアントと Fedora の両方にアクセスできます。Red Hat Enterprise Linux クライアントは、誰でもブートできます。ユーザー **jane** は設定でパーミッションが与えられているので、Fedora をブートできます。メニューエントリーを指定しないと、パスワード保護機能は作動しません。テンプレートファイルで変更を行った後は、GRUB 2 設定ファイルを更新する必要があります。

以下のように **grub2-mkconfig -o** コマンドを実行して、**grub.cfg** ファイルを更新します。

- BIOS ベースのマシンでは、**root** で以下のコマンド発行します。

```
~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

- UEFI ベースのマシンでは、**root** で以下のコマンド発行します。

```
~]# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

メニューエントリー用のユーザーとパスワードを指定しなくても、このシステムにアクセスする際にはスーパーユーザーのパスワードが要求されます。

22.3.3. パスワードの暗号化

デフォルトでは、パスワードは GRUB 2 スクリプト内でプレーンテキストで保存されます。正しいパスワードがないとブート時にファイルにはアクセスできませんが、**grub2-mkpasswd-pbkdf2** コマンドを使ってパスワードを暗号化することでセキュリティーを高めることができます。このコマンドはパスワードを長いハッシュに変換し、これがプレーンテキストのパスワードの代わりに GRUB 2 スクリプトに置かれます。

1. 暗号化されたパスワードを生成するには、**root** で **grub2-mkpasswd-pbkdf2** コマンドをコマンドラインで実行します。
2. プロンプトで暗号化するパスワードを入力し、再度入力します。すると、コマンドがパスワードを暗号化された形式で出力します。
3. ハッシュをコピーし、ユーザーを設定したファイルにペーストします。**/etc/grub.d/00_header** または **/etc/grub.d/40_custom** になります。**00_header** ファイルでは、以下のフォーマットが適用されます。

```
cat <<EOF
setsuperusers="john"
password_pbkdf2 john
grub.pbkdf2.sha512.10000.19074739ED80F115963D984BDCB35AA671C24325755
377C3E9B014D862DA6ACC77BC110EED41822800A87FD3700C037320E51E9326188D5
```

```
3247EC0722DDF15FC.C56EC0738911AD86CEA55546139FEBC366A393DF9785A8F44
D3E51BF09DB980BAFEF85281CBBC56778D8B19DC94833EA8342F7D73E3A1AA30B20
5091F1015A85
EOF
```

40_header ファイルでは、以下のフォーマットが適用されます。

```
setsuperusers="john"
password_pbkdf2 john
grub.pbkdf2.sha512.10000.19074739ED80F115963D984BDCB35AA671C24325755
377C3E9B014D862DA6ACC77BC110EED41822800A87FD3700C037320E51E9326188D5
3247EC0722DDF15FC.C56EC0738911AD86CEA55546139FEBC366A393DF9785A8F44
D3E51BF09DB980BAFEF85281CBBC56778D8B19DC94833EA8342F7D73E3A1AA30B20
5091F1015A85
```

警告

適切なフォーマットを使用しない場合、もしくは間違った方法で設定を修正した場合は、システムのブートができなくなる可能性があります。

22.4. GRUB 2 の再インストール

GRUB 2 の誤ったインストールやファイルの欠如、システムの破損などで引き起こされる特定の問題を解決するには、GRUB 2 の再インストールが便利で簡単な方法となる場合があります。GRUB 2 を再インストールする理由には、これらの他に以下のものがあります。

- GRUB の以前のバージョンからのアップグレード
- インストール済みのオペレーティングシステムを制御するために、ユーザーが GRUB 2 ブートローダーを必要としている。ただし、オペレーティングシステムのなかには独自のブートローダーとインストールされるものもあります。GRUB 2 を再インストールすることで、希望するオペレーティングシステムに制御が戻されます。
- 別のドライブにブート情報を追加する。

22.4.1. grub2-install コマンドの使用

grub2-install コマンドを使用すると、ブート情報が更新され、欠けていたファイルが回復されます。ファイルが回復されるのは、それらが破損していない場合のみです。**/boot/grub2/** ディレクトリーが見つからない場合は、再度作成されます。

システムが正常に稼働している場合は、**grub2-install <device>** コマンドを使うと GRUB 2 が再インストールされます。例を示します。

```
~]# grub2-install /dev/sda
```

22.4.2. GRUB 2 の再設定と再インストール

この方法はすべての GRUB 2 設定ファイルを完全の削除するので、ユーザーが全設定をデフォルト値にリセットしたい場合に使用されます。設定ファイルの削除して GRUB 2 を再インストールすると、破損ファイルおよび間違った設定による失敗が修正されます。これを実行するには、**root** で以下の手順にしたがいます。

1. **rm /etc/grub.d/*** コマンドを実行します。
2. **rm /etc/sysconfig/grub** コマンドを実行します。
3. **yum reinstall grub2-tools** コマンドを実行します。
4. 以下のように **grub2-mkconfig -o** コマンドを実行して、**grub.cfg** ファイルを更新します。
   - BIOS ベースのマシンでは、**root** で以下のコマンド発行します。

     ```
     ~]# grub2-mkconfig -o /boot/grub2/grub.cfg
     ```

   - UEFI ベースのマシンでは、**root** で以下のコマンド発行します。

     ```
     ~]# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
     ```

5. 「grub2-install コマンドの使用」にある手順にしたがい、**/boot/** パーティション上の GRUB2 を復元します。

22.5. シリアルコンソールでの GRUB 2

画面やキーボードのないコンピューターを使用している場合は、シリアル通信によるマシンの制御が非常に便利です。

22.5.1. GRUB 2 の設定

シリアルラインで GRUB 2 を使うには、**/etc/default/grub** ファイルに以下の 2 行を追加します。

```
GRUB_TERMINAL="serial"
GRUB_SERIAL_COMMAND="serial --speed=9600 --unit=0 --word=8 --parity=no --
-stop=1"
```

最初の行は、グラフィカルターミナルを無効にします。**GRUB_TERMINAL** キーを指定すると、**GRUB_TERMINAL_INPUT** および **GRUB_TERMINAL_OUTPUT** の値を上書きすることに注意してください。2 行目では、ボーレートやパリティー、その他の値を使用中の環境とハードウェアに適合するように調整します。たとえば、**115200** のように非常に高いボーレートは、ログファイルの後のタスクに適しています。**/etc/default/grub** ファイルでの変更後は、GRUB 2 設定ファイルの更新が必要になります。

以下のように **grub2-mkconfig -o** コマンドを実行して、**grub.cfg** ファイルを更新します。

- BIOS ベースのマシンでは、**root** で以下のコマンド発行します。

  ```
  ~]# grub2-mkconfig -o /boot/grub2/grub.cfg
  ```

- UEFI ベースのマシンでは、**root** で以下のコマンド発行します。

  ```
  ~]# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
  ```

注記

このセクションでは、シリアル接続による grub ターミナルのアクセス設定を説明していますが、カーネルがシリアル接続を監視するようにするには、新たなオプションをカーネル定義に追加する必要があります。例を示します。

```
console=ttyS0,9600n8
```

ここでの **console=ttyS0** は使用するシリアルターミナル、**9600** はボーレート、**n** はパリティーなし、**8** はビットでの単語の長さになります。カーネルオプションの追加に関する詳細情報は、「エントリーの編集」を参照してください。シリアルコンソールの設定に関する詳細情報は、https://www.kernel.org/doc/Documentation/serial-console.txt を参照してください。

22.5.2. 画面を使ったシリアルコンソールへの接続

screen ツールは、シリアルターミナルとして機能します。このツールをインストールするには、**root** で以下を実行します。

```
~]# yum install screen
```

シリアルコンソールを使ってマシンに接続するには、以下のコマンドを実行します。

```
~]$ screen /dev/<console_port>
```

デフォルトでは、オプションが指定されないと **screen** は標準の 9600 ボーレートを使用します。別のボーレートを設定するには、以下のコマンドを実行します。

```
~]$ screen /dev/<console_port> 115200
```

screen でセッションを終了するには、**Ctrl+a** を押して、**:quit** と入力し、**Enter** を押します。

追加オプションおよび詳細情報は、**screen(1)** man ページを参照してください。

22.6. ブート中のターミナルメニューの編集

ブート時にメニューエントリーを修正し、カーネルに引数を渡すことができます。これは、メニューエントリーエディターインターフェイスを使って行い、このインターフェイスはブートローダーメニュー内で選択したメニューエントリーで **e** キーを押すと開始されます。**Esc** キーはすべての変更を破棄し、標準メニューインターフェイスを再度読み込みます。**c** キーは、コマンドラインインターフェイスを読み込みます。

コマンドラインインターフェースは最も基本的な GRUB インターフェイスですが、制御を一番多く付与するものでもあります。コマンドラインでは、関連する GRUB コマンドの入力が可能で、それに続けて **Enter** キーを押すとそのコマンドが実行されます。このインターフェイスには、コンテキストをベースにした **Tab** キー自動入力や行頭に移動する **Ctrl+a**、行末に移動する **Ctrl+e** など、**shell** と同様の高度な機能が搭載されています。さらに、**矢印**、**Home**、**End**、および **Delete** のキーは bash シェルでの機能と同様に作動します。

22.6.1. レスキューモードでの起動

レスキューモードは、便利なシングルユーザー環境を提供し、通常の起動プロセスを完了できない状況にお

いてシステムの修復を可能にします。レスキューモードでは、システムはすべてのローカルファイルシステムのマウントといくつかの重要なシステムサービスの開始を試みますが、ネットワークインターフェイスの有効化や、システムに同時に他のユーザーによるログインを許可したりすることはしません。Red Hat Enterprise Linux 7 では、レスキューモードはシングルユーザーモードと同等であり、root パスワードを必要とします。

1. ブート中にレスキューモードに入るには、GRUB 2 ブート画面で **e** キーを押して編集を行います。
2. **linux16** 行の最後に以下のパラメーターを追加します。UEFI システムの場合は、**linuxefi** 行になります。

   ```
   systemd.unit=rescue.target
   ```

   Ctrl+**a** か **Ctrl**+**e** を押して行の最初または最後に移動します。システムによっては、**Home** と **End** が機能するものもあります。

   1、***s***、および ***single*** という同等のパラメーターをカーネルに渡すことも可能であることに注意してください。
3. **Ctrl**+**x** を押して追加したパラメーターでシステムを起動します。

22.6.2. 緊急モードでのブート

緊急モードは、可能な限り最小限の環境を提供し、システムがレスキューモードに入れない状態でもシステムの修復を可能にします。緊急モードでは、システムは root ファイルシステムを読み込み専用でマウントし、他のローカルファイルシステムのマウントは試みません。また、ネットワークインターフェイスのアクティブ化も行わず、限定的な必須サービスのみを起動します。Red Hat Enterprise Linux 7 では、緊急モードに root パスワードを必要とします。

1. 緊急モードに入るには、GRUB 2 ブート画面で **e** キーを押して編集を行います。
2. **linux16** 行の最後に以下のパラメーターを追加します。UEFI システムの場合は、**linuxefi** 行になります。

   ```
   systemd.unit=emergency.target
   ```

   Ctrl+**a** か **Ctrl**+**e** を押して行の最初または最後に移動します。システムによっては、**Home** と **End** が機能するものもあります。

   emergency および ***-b*** という同等のパラメーターをカーネルに渡すことも可能であることに注意してください。
3. **Ctrl**+**x** を押して追加したパラメーターでシステムを起動します。

22.6.3. Root パスワードのリカバリー

Red Hat Enterprise Linux 7 のインストールでは、**root** パスワードの設定が必須となっています。このパスワードを忘れたりなくしたりした場合は、再設定が可能です。

GRUB 2 でのパスワードの再設定は、Red Hat Enterprise Linux 6 での GRUB のようなシングルユーザモードでの実行はなされません。**single-user** モードおよび **emergency** モードでの操作には、**root** パスワードが必要となっています。

手順22.1 root パスワードのリセット

1. システムを起動し、GRUB 2 ブート画面で **e** キーを押して編集を行います。

2. **linux16** 行の最後に以下のパラメーターを追加します。UEFI システムの場合は、**linuxefi** 行になります。

```
init=/bin/sh
```

Ctrl+a か **Ctrl+e** を押して行の最初または最後に移動します。システムによっては、**Home** と **End** が機能するものもあります。

Linux *kernel* は、システム **init** デーモンではなく、**/bin/sh** シェルを実行します。このため、機能によっては制限されるか使えないものもあります。

重要

システムメッセージを有効にするには、**rhgb** と **quiet** のパラメーターを無効にする必要があります。

3. **Ctrl+x** を押して追加したパラメーターでシステムを起動します。

シェルプロンプトが表示されます。

4. ファイルシステムが読み取り専用でマウントされます。ファイルシステムが書き込み可能になっていないと、パスワードの変更はできません。

ファイルシステムを書き込み可能として再マウントするには、**mount -o remount, rw /** コマンドを実行します。

5. **passwd** コマンドを実行し、コマンドラインに表示される指示にしたがって**root** パスワードを変更します。

システムが書き込み可能でないと、**passwd** ツールは以下のエラーメッセージを表示して失敗します。

```
Authentication token manipulation error
```

6. 修正されたファイルの SELinux コンテキストが起動後に正常に復元されるようにするには、以下を実行します。

```
touch /.autorelabel
```

7. **exec /sbin/init** コマンドを実行して初期化を再開し、システム起動を完了します。

別のコマンドを指定して **exec** コマンドを実行すると、シェルに代わって新たなプロセスが作成されます。このケースでは、**init** がこれになります。

別の方法では、システムを再起動するには、代わりに **exec /sbin/reboot** コマンドを実行します。

22.7. UEFI セキュアブート

セキュアブート技術では、システムブートローダーがファームウェアに含まれているデータベースが認可した暗号化キーで署名されているかどうかを、システムファームウェアが確実にチェックします。次のステージのブートローダー、カーネル、および (潜在的に) ユーザースペースでの署名認証を行うことで、無署名コードの実行を防ぐことができます。

セキュアブートは、Unified Extensible Firmware Interface (UEFI) 仕様のブートパス検証コンポーネントです。この仕様は、以下を定義します。

- 揮発性ではないストレージでの暗号で保護された UEFI 変数用のプログラミングインターフェイス
- 信頼できる X.509 root 証明書の UEFI 変数での保存方法
- ブートローダーやドライバーなどの UEFI アプリケーションの検証
- 既知の問題のある証明書およびアプリケーションハッシュを無効にする手順

UEFI セキュアブートは、第 2 ステージのブートローダーのインストールや削除を防ぎません。また、このような変更の明示的なユーザー確認を必要としません。署名は、ブートローダーのインストール時や更新時ではなく、ブート時に検証されます。このため、UEFI セキュアブートは、ブートパス操作を停止しません。ブートパスが修正された場合に、この修正ブートパスをシステムが実行しないようにするだけで、この検出を単純化します。

22.7.1. Red Hat Enterprise Linux 7 における UEFI セキュアブートのサポート

Red Hat Enterprise Linux 7 には UEFI セキュアブート機能が含まれているので、Red Hat Enterprise Linux 7 は UEFI セキュアブートが有効になっているシステム上でインストールおよび実行が可能です。セキュアブート技術が有効になっている UEFI ベースのシステムでは、読み込み済みのすべてのドライバーが有効な証明書で署名されている必要があります。この署名がないとシステムはドライバーを受け付けません。Red Hat 提供のドライバーはすべて、UEFI CA 証明書で署名されています。

Red Hat Enterprise Linux DVD では提供されていない外部構築のドライバーを読み込みたい場合は、これらのドライバーも署名されていることを確認してください。

カスタムドライバーの署名に関する情報は、「セキュアブート用のカーネルモジュールの署名」でご覧になれます。

22.8. その他のリソース

GRUB 2 ブートローダーに関する詳細情報は、以下のリソースを参照してください。

- **`/usr/share/doc/grub2-tools-<version-number>`** — このディレクトリーには、GRUB 2 の使い方や設定に関する情報が含まれています。**`<version-number>`** は、インストールされている GRUB 2 パッケージのバージョンに対応するものです。
- **`info grub2`** — GRUB 2 情報ページにはチュートリアル、ユーザーリファレンスガイド、プログラマーリファレンスガイド、 GRUB 2 と その使い方に関する FAQ などが含まれています。
- Red Hat インストールガイド — インストールガイドは、インストールや用語、インターフェイス、コマンドなどの GRUB 2 に関する基本的な情報を提供しています。

第23章 手動でのカーネルのアップグレード

Red Hat Enterprise Linux カーネルは、サポートされているハードウェアとの整合性と互換性を確実にするために Red Hat Enterprise Linux のカーネルチームによりカスタムメイド作製されたものです。Red Hat がリリースする前に、カーネルは厳格な品質保証テスト群をパスしなければなりません。

Red Hat Enterprise Linux カーネルは RPM 形式でパッケージされているので、**Yum** または **PackageKit** パッケージマネージャを使用して簡単にアップグレードも検証もできます。**PackageKit** は自動的に Red Hat Network サーバーにクエリを出して、カーネルパッケージを含む利用可能な更新があるパッケージを通知します。

そのため、この章は **yum** の代わりに **rpm** コマンドを使用して手動でカーネルパッケージを更新する必要のあるユーザーに***のみ*** 役立つものになっています。

警告

可能な限り、**Yum** または **PackageKit** パッケージマネージャーを使用して新しいカーネルをインストールしてください。これらのツールは、常に新しいカーネルを **インストール** するからです。他の方法で現行バージョンを入れ替えると、(失敗した場合) システムが起動できなくなる可能性があります。

警告

カスタムカーネルの構築は、Red Hat Global Services Support チームではサポートされていません。そのため、このマニュアルではその記述もありません。

Yum を使ったカーネルパッケージのインストールについては、「パッケージの更新」 を参照してください。Red Hat Network についての情報は、カスタマーポータル にある関連ドキュメントを参照してください。

23.1. カーネルパッケージの概要

Red Hat Enterprise Linux には、以下のカーネルパッケージが含まれています。

- *kernel* — シングル、マルチコア、およびマルチプロセッサーシステム用のカーネルが含まれています。
- *kernel-debug* — カーネル診断のために有効になっている数多くのデバッグオプションを持つカーネルが含まれています。パフォーマンスが犠牲になります。
- *kernel-devel* — *kernel* パッケージに対して、モジュールを構築するのに十分なカーネルヘッダーと makefiles を含んでいます。
- *kernel-debug-devel* — *kernel-debug* パッケージに適合するカーネルモジュールの構築に必要なファイルを含んでいます。
- *kernel-headers* — Linux カーネルと、ユーザースペースライブラリーおよびプログラムとの間のインターフェイスを指定する C ヘッダーファイルを含んでいます。このヘッダーファイルは、ほとんどの標準プログラムの構築に必要な構造と定数を定義します。
- *linux-firmware* — さまざまなデバイス操作に必要なファームウェアすべてを含んでいます。
- *perf* — Linux カーネルのパフォーマンスモニタリングを有効にする**perf** ツールを含んでいます。

- *kernel-abi-whitelists* — Red Hat Enterprise Linux カーネル ABI に関連する情報を含んでいます。これには、外部の Linux カーネルモジュールが必要とするカーネル記号の一覧と強化をアシストする *yum* プラグインが含まれます。
- *kernel-tools* — Linux カーネル操作のツールとサポートドキュメントが含まれています。

23.2. アップグレードへの準備

カーネルをアップグレードする前に、予防的な手順の実行が推奨されます。

まず、システム用に機能するブートメディアがあることを確認します。ブートローダーが新しいカーネルをブートするように正しく設定されていない場合、このメディアを使って Red Hat Enterprise Linux をブートすることができます。

USB メディアは多くの場合、*ペンドライブ*、*サムディスク*、*キー*などと呼ばれるフラッシュデバイスの形式で、あるいは外部接続のハードディスクとして提供されています。このタイプのほとんどすべてが **VFAT** ファイルシステムとしてフォーマットされています。**ext2**、**ext3**、または **VFAT** としてフォーマットされているメディア上でブート可能な USB メディアを作成することができます。

ディストリビューションのイメージファイル、または最小ブートメディアイメージは、USB メディアに転送することができます。デバイスには十分な空き領域があることを確認してください。ディストリビューション DVD イメージにはおよそ **4 GB** が、ディストリビューション CD イメージにはおよそ**700 MB** が、そして最小ブートメディアイメージにはおよそ **10 MB** が必要です。

Red Hat Enterprise Linux のインストール DVD、またはインストール CD-ROM #1 からコピーした **boot.iso** ファイルと、およそ **16 MB** の空き領域を持つ **VFAT** ファイルシステムでフォーマットした USB ストレージデバイスが必要になります。以下の手順は、コピー先のファイルと同じパス名でない限りは USB ストレージデバイス上の既存のファイルに影響しません。USB ブートメディアを作成するには、root ユーザーとして以下のコマンドを実行します。

1. *syslinux* パッケージがインストールされていない場合は、これをインストールします。root として **yum install *syslinux*** コマンドを実行して、これを行います。
2. USB ストレージデバイス上に **SYSLINUX** ブートローダーをインストールします。

   ```
   ~]# syslinux /dev/sdX1
   ```

   ここでの *sdX* はデバイス名です。
3. **boot.iso** と USB ストレージデバイス用にマウントポイントを作成します。

   ```
   ~]# mkdir /mnt/isoboot /mnt/diskboot
   ```
4. **boot.iso** をマウントします。

   ```
   ~]# mount -o loop boot.iso /mnt/isoboot
   ```
5. USB ストレージデバイスをマウントします。

   ```
   ~]# mount /dev/<sdX1> /mnt/diskboot
   ```
6. **boot.iso** から USB ストレージデバイスに**ISOLINUX** ファイルをコピーします。

   ```
   ~]# cp /mnt/isoboot/isolinux/* /mnt/diskboot
   ```

7. **boot.iso** からの **isolinux.cfg** ファイルを USB デバイス用の **syslinux.cfg** ファイルとして使用します。

```
~]# grep -v local /mnt/isoboot/isolinux/isolinux.cfg >
/mnt/diskboot/syslinux.cfg
```

8. **boot.iso** と USB ストレージデバイスをアンマウントします。

```
~]# umount /mnt/isoboot /mnt/diskboot
```

9. このブートメディアでマシンを再起動して、ブートできることを確認してから他の操作に移ります。

別の方法としては、フロッピィードライブがあるシステム上でブートディスクを作成するには、*mkbootdisk* パッケージをインストールして、root として **mkbootdisk** コマンドを実行します。このパッケージをインストールした後に使用法について確認するには、**man mkbootdisk** man ページを参照してください。

どのカーネルがインストールされているかを判定するには、シェルプロンプトでコマンド **yum list installed "kernel-*"** を実行します。出力では、システムのアーキテクチャーに応じて、以下のパッケージのすべてか、または一部が示されます。バージョン番号は異なる場合があります。

```
~]# yum list installed "kernel-*"
kernel.x86_64                 3.10.0-54.0.1.el7          @rhel7/7.0
kernel-devel.x86_64           3.10.0-54.0.1.el7          @rhel7
kernel-headers.x86_64         3.10.0-54.0.1.el7          @rhel7/7.0
```

この出力から、カーネルのアップグレード用にダウンロードすべきパッケージを判断します。シングルプロセッサーのシステムでは、必要なパッケージは *kernel* パッケージのみです。別のパッケージの説明は、「カーネルパッケージの概要」を参照してください。

23.3. アップグレードされたカーネルのダウンロード

システム用にアップグレードされたカーネルが利用可能か判定する手段はいくつかあります。

- セキュリティーエラータ — セキュリティー問題を修復するカーネルを含む、セキュリティーエラータについての詳細は、https://access.redhat.com/site/security/updates/active/ を参照してください。
- Red Hat Network 経由 — カーネル RPM パッケージをダウンロードしてインストールします。Red Hat Network は、最新のカーネルをダウンロードし、システム上のカーネルをアップグレードして、必要に応じて初期 RAM ディスクイメージを作成してから新しいカーネルをブートするためにブートローダーを設定することができます。詳細情報は、https://access.redhat.com/site/documentation/en-US/ にある関連ドキュメントを参照してください。

カーネルのダウンロードとインストールに Red Hat Network が使用される場合は、「初期 RAM ディスクイメージの検証」と「ブートローダーの検証」にある指示にしたがいます。ただし、カーネルをデフォルトでブートするように **変更しない** でください。Red Hat Network が自動的に最新バージョンをデフォルトのカーネルに変更します。手動でカーネルをインストールするには、「アップグレードの実行」へ進みます。

23.4. アップグレードの実行

必要なパッケージをすべて取り込んだ後は、既存カーネルをアップグレードします。

重要

新しいカーネルで問題がある可能性を考慮し、古いカーネルを保持することが強く推奨されます。

シェルプロンプトで、カーネル RPM パッケージを格納しているディレクトリーに移動します。**rpm** コマンドに **-i** 引数を使用して古いカーネルを残します。**-U** オプションは現在インストールしてあるカーネルを上書きして、ブートローダー問題を起こすので、これは **使用しないでください**。例を示します。

```
~]# rpm -ivh kernel-<kernel_version>.<arch>.rpm
```

次のステップでは、初期 RAM ディスクイメージが作成されているかどうかを検証します。詳細は、「初期 RAM ディスクイメージの検証」 を参照してください。

23.5. 初期 RAM ディスクイメージの検証

初期 RAM ディスクイメージの役割は、IDE、SCSI、または RAID などのブロックデバイスモジュールをフレロードすることです。そうすることで、それらのモジュールが通常配備されている root ファイルシステムがアクセス可能になり、マウントできるようになります。Red Hat Enterprise Linux 7 システム上では、**Yum**、**PackageKit**、**RPM** パッケージマネージャーのいずれかを使用してカーネルがインストールされる時は常に、**Dracut** ユーティリティーがインストールスクリプトによって呼び出されて、*initramfs* (初期 RAM ディスクイメージ) を作成します。

IBM eServer System i (23.5項「IBM eServer System i 上の初期 RAM ディスクイメージとカーネルの検証」 を参照) 以外のすべてのアーキテクチャー上では、**dracut** コマンドを実行すると **initramfs** を作成できます。しかし、通常は手動で **initramfs** を作成する必要はありません。このステップは、カーネルとその関連パッケージが Red Hatで配布された RPM パッケージからインストールされているか、またはアップグレードされている場合には自動的に実行されます。

現在のカーネルバージョンに該当する **initramfs** が存在していること、およびそれが**grub.conf** 設定ファイル内で正しく指定されていることを検証するには、以下の手順にしたがいます。

手順23.1 初期 RAM ディスクイメージの検証

1. root として、**/boot/** ディレクトリーのコンテンツを一覧表示して、カーネル (**vmlinuz-<kernel_version>**) と最新のバージョン番号を持つ **initramfs-<kernel_version>** を見つけます。

   例23.1 カーネルと initramfs バージョンの一致を確認

   ```
   ~]# ls /boot/
   config-3.10.0-67.el7.x86_64
   config-3.10.0-78.el7.x86_64
   efi
   grub
   grub2
   initramfs-0-rescue-07f43f20a54c4ce8ada8b70d33fd001c.img
   initramfs-3.10.0-67.el7.x86_64.img
   initramfs-3.10.0-67.el7.x86_64kdump.img
   initramfs-3.10.0-78.el7.x86_64.img
   initramfs-3.10.0-78.el7.x86_64kdump.img
   initrd-plymouth.img
   symvers-3.10.0-67.el7.x86_64.gz
   ```

```
symvers-3.10.0-78.el7.x86_64.gz
System.map-3.10.0-67.el7.x86_64
System.map-3.10.0-78.el7.x86_64
vmlinuz-0-rescue-07f43f20a54c4ce8ada8b70d33fd001c
vmlinuz-3.10.0-67.el7.x86_64
vmlinuz-3.10.0-78.el7.x86_64
```

例23.1「カーネルと initramfs バージョンの一致を確認」は以下の点を示しています。

- 3 つのカーネルがインストールされています (より正確には、3 つのカーネルファイルが **/boot/** にあります)。
- 最新のカーネルは **vmlinuz-3.10.0-78.el7.x86_64** です。
- そのカーネルバージョンに一致する **initramfs** ファイルである **initramfs-3.10.0-78.el7.x86_64kdump.img** も存在します。

重要

/boot/ ディレクトリー内で、いくつかの**initramfs-*<version>*kdump.img** ファイルを見つけること場合もあります。これらは、**Kdump** メカニズムがカーネルデバッグ目的で作成した特別なファイルです。システムのブートには使用されないため、無視しても構いません。

2. (オプション) 使用している **initramfs-*<kernel_version>*** ファイルが、**/boot/** にある最新カーネルのバージョンと一致しない場合、または他の特定状況では、**Dracut** ユーティリティーを使用して **initramfs** ファイルを生成する必要があるかも知れません。root としてオプションなしで **dracut** を呼び出すと、それが **/boot/** ディレクトリー内にある最新のカーネル用に **initramfs** ファイルを生成するようになります。

```
~]# dracut
```

既存の **initramfs** を上書きする (例えば、これまでの**initramfs** が破損しているなど) ために **dracut** を使用する場合は、**--force** オプションを使用する必要があります。これを使用しないと、**dracut** は既存の **initramfs** ファイルの上書きを拒否します。

```
~]# dracut
Will not override existing initramfs (/boot/initramfs-3.10.0-
78.el7.x86_64.img) without --force
```

現在のディレクトリーに initramfs を作成するには、 **dracut *<initramfs_name>* *<kernel_version>*** を呼び出します。

```
~]# dracut "initramfs-$(uname -r).img" $(uname -r)
```

事前に読み込む特定のカーネルモジュールを指定する必要がある場合は、それらのモジュール名 (**.ko** などのファイル名の接頭辞を除く) を **/etc/dracut.conf** 設定ファイルにある **add_dracutmodules+="*<module>* [*<more_modules>*]"** 指示文の括弧内に追加します。dracut で作成された**initramfs** イメージファイルのコンテンツは、**lsinitrd *<initramfs_file>*** コマンドを使用すると、一覧表示できます。

```
~]# lsinitrd /boot/initramfs-3.10.0-78.el7.x86_64.img
```

```
Image: /boot/initramfs-3.10.0-78.el7.x86_64.img: 11M
=============================================================================
======
dracut-033-68.el7
=============================================================================
======

drwxr-xr-x  12 root     root            0 Feb  5 06:35 .
drwxr-xr-x   2 root     root            0 Feb  5 06:35 proc
lrwxrwxrwx   1 root     root           24 Feb  5 06:35 init ->
/usr/lib/systemd/systemd
drwxr-xr-x  10 root     root            0 Feb  5 06:35 etc
drwxr-xr-x   2 root     root            0 Feb  5 06:35
usr/lib/modprobe.d[出力は省略されています]
```

オプションと使用法についての詳細情報は、**man dracut** と **man dracut.conf** を参照してください。

3. **/boot/grub2/** ディレクトリー内の **grub.conf** 設定ファイルを検査して、ブートするカーネルバージョン用に **initramfs initramfs-<*kernel_version*>.img** が存在することを確認します。詳細は、「ブートローダーの検証」を参照してください。

IBM eServer System i 上の初期 RAM ディスクイメージとカーネルの検証

IBM eServer System i のマシン上では、初期 RAM ディスクとカーネルファイルは 1 つのファイルに統合してあり、これは **addRamDisk** コマンドで作成されます。カーネルとその関連パッケージがインストールされているか、または Red Hat で配布されている RPM パッケージでアップグレードされている場合は、このステップは自動的に実行されます。その場合、手動で実行する必要はありません。これが作成されていることを確認するには、コマンド **ls -l /boot/** を使用して **/boot/vmlinitrd-<*kernel_version*>** ファイルが存在することを確認します (**<*kernel_version*>** はインストールしたばかりのカーネルバージョンと一致する必要があります)。

23.6. ブートローダーの検証

rpm を使用してカーネルをインストールすると、カーネルパッケージはブートローダー設定ファイル内にその新しいカーネル用のエントリーを作成します。しかし、**rpm** はその新しいカーネルがデフォルトカーネルとしてブートするようには設定**しません。rpm** で新しいカーネルをインストールした時には手動で実行する必要があります。

rpm で新しいカーネルをインストールした後には、常にブートローダー設定ファイルをダブルチェックして設定が正しいことを確認するようにお薦めします。これを実行しないと、システムは Red Hat Enterprise Linux を正しくブートできない可能性があります。ブートしない場合、先に作成したブートメディアでシステムをブートしてブートローダーを再設定します。

第24章 カーネルモジュールでの作業

Linux カーネルはモジュール式であるため、動的に読み込まれる *カーネルモジュール* を使用することでその機能を拡張することができます。カーネルモジュールは、以下のものを提供できます。

- 新しいハードウェアに対するサポートを強化するデバイスドライバー、または
- **btrfs** や **NFS** のようなファイルシステムへのサポート

カーネル自体と同様に、モジュールは動作をカスタマイズするパラメーターを取ることができます。ただし、デフォルトのパラメーターでほとんどのケースで十分に機能します。ユーザースペースツールで、実行中のカーネルに現在読み込まれているモジュールを一覧表示することができます。利用可能なパラメーター用の利用可能なモジュールとモジュール特有の情報をクエリすることができます。また、実行中のカーネルに動的にモジュールを読み込んだり、そこから削除することができます。*kmod* パッケージで提供されるこれらのユーティリティーの多くは、その実行時にはモジュールの依存関係を計算に入れているため、手動の依存関係トラッキングはめったに必要とされません。

最近のシステムでは、必要な状況になると各種メカニズムによって自動的にカーネルモジュールが読み込まれます。しかし、モジュールを手動で読み込み、削除する必要がある状況も時にあります。たとえば、どちらのモジュールも基本的な機能は提供できるものの、どちらかの方が好まれる場合や、モジュールが不正な動作をしている場合などです。

本章では、以下について説明します。

- ユーザースペース **kmod** ユーティリティーを使用して、カーネルモジュールとその依存関係の表示、クエリ、読み込み、削除をする方法
- コマンドラインで動的かつ永続的にモジュールパラメーターを設定して、カーネルモジュールの動作をカスタマイズする方法
- ブート時にモジュールを読み込む方法

注記

この章で説明してあるカーネルモジュールユーティリティーを使用するには、以下を root で実行して、まず *kmod* パッケージがシステムにインストールされていることを確認します。

```
~]# yum install kmod
```

Yum を使ったパッケージのインストール方法については 「パッケージのインストール」を参照してください。

24.1. 読み込み済みモジュールの一覧表示

lsmod コマンドを実行すると、現在カーネルに読み込み済みの全カーネルモジュールを一覧表示できます。

```
~]$ lsmod
Module                  Size  Used by
tcp_lp                 12663  0
bnep                   19704  2
bluetooth             372662  7 bnep
```

```
rfkill                 26536  3 bluetooth
fuse                   87661  3
ip6t_rpfilter          12546  1
ip6t_REJECT            12939  2
ipt_REJECT             12541  2
xt_conntrack           12760  7
ebtable_nat            12807  0
ebtable_broute         12731  0
bridge                110196  1 ebtable_broute
stp                    12976  1 bridge
llc                    14552  2 stp,bridge
ebtable_filter         12827  0
ebtables               30913  3
ebtable_broute,ebtable_nat,ebtable_filter
ip6table_nat           13015  1
nf_conntrack_ipv6      18738  5
nf_defrag_ipv6         34651  1 nf_conntrack_ipv6
nf_nat_ipv6            13279  1 ip6table_nat
ip6table_mangle        12700  1
ip6table_security      12710  1
ip6table_raw           12683  1
ip6table_filter        12815  1
ip6_tables             27025  5
ip6table_filter,ip6table_mangle,ip6table_security,ip6table_nat,ip6table_
raw
iptable_nat            13011  1
nf_conntrack_ipv4      14862  4
nf_defrag_ipv4         12729  1 nf_conntrack_ipv4
nf_nat_ipv4            13263  1 iptable_nat
nf_nat                 21798  4
nf_nat_ipv4,nf_nat_ipv6,ip6table_nat,iptable_nat[出力は省略されています]
```

lsmod 出力の各行は以下を明記しています。

- メモリに現在読み込み済みのカーネルモジュールの名前
- カーネルモジュールが使用するメモリ量
- モジュールとそれに依存する他のモジュールを使用しているプロセスの合計、ある場合はそれらモジュール名の一覧。この一覧を使用して、アンロードしたいモジュールに依存しているモジュールすべてをまず最初にアンロードできます。詳細は、「モジュールのアンロード」を参照してください。

最後に注目する点としては、**lsmod** 出力は **/proc/modules** 擬似ファイルの内容よりも大まかで、かなり読みやすいことです。

24.2. モジュール情報の表示

modinfo *<module_name>* コマンドを実行すると、カーネルモジュールに関する詳細情報を表示することができます。

注記

カーネルモジュール名を **kmod** ユーティリティーのいずれかの引数として記入する際には、その名前の末尾に拡張子 **.ko** を付けないでください。カーネルモジュール名には、拡張子を付けません。カーネルモジュールに対応するファイルには拡張子が付きます。

たとえば、Intel PRO/1000 ネットワークドライバーである **e1000e** モジュールに関する情報を表示するには、以下を実行します。

例24.1 lsmod を使用したカーネルモジュール情報の一覧表示

```
~]# modinfo e1000e
filename:       /lib/modules/3.10.0-
121.el7.x86_64/kernel/drivers/net/ethernet/intel/e1000e/e1000e.ko
version:        2.3.2-k
license:        GPL
description:    Intel(R) PRO/1000 Network Driver
author:         Intel Corporation, <linux.nics@intel.com>
srcversion:     E9F7E754F6F3A1AD906634C
alias:          pci:v00008086d000015A3sv*sd*bc*sc*i*
alias:          pci:v00008086d000015A2sv*sd*bc*sc*i*[alias の数行は省略]
alias:          pci:v00008086d0000105Esv*sd*bc*sc*i*
depends:        ptp
intree:         Y
vermagic:       3.10.0-121.el7.x86_64 SMP mod_unload modversions
signer:         Red Hat Enterprise Linux kernel signing key
sig_key:
42:49:68:9E:EF:C7:7E:95:88:0B:13:DF:E4:67:EB:1B:7A:91:D1:08
sig_hashalgo:   sha256
parm:           debug:Debug level (0=none,...,16=all) (int)
parm:           copybreak:Maximum size of packet that is copied to a
new buffer on receive (uint)
parm:           TxIntDelay:Transmit Interrupt Delay (array of int)
parm:           TxAbsIntDelay:Transmit Absolute Interrupt Delay (array
of int)
parm:           RxIntDelay:Receive Interrupt Delay (array of int)
parm:           RxAbsIntDelay:Receive Absolute Interrupt Delay (array
of int)
parm:           InterruptThrottleRate:Interrupt Throttling Rate (array
of int)
parm:           IntMode:Interrupt Mode (array of int)
parm:           SmartPowerDownEnable:Enable PHY smart power down
(array of int)
parm:           KumeranLockLoss:Enable Kumeran lock loss workaround
(array of int)
parm:           WriteProtectNVM:Write-protect NVM [WARNING: disabling
this can lead to corrupted NVM] (array of int)
parm:           CrcStripping:Enable CRC Stripping, disable if your BMC
needs the CRC (array of int)
```

以下は、**modinfo** 出力にある一部のフィールドの説明です。

filename

.ko カーネルオブジェクトファイルへの絶対パスです。**modinfo -n** は **filename** フィールドのみを表示するショートカットコマンドとして使用できます。

description

モジュールに関する簡単な説明です。**modinfo -d** は、詳細フィールドのみを表示するショートカットコマンドとして使用できます。

alias

alias フィールドは、モジュールにエイリアスがある回数だけ表示されます。ない場合は、完全に省略されます。

depends

このフィールドには、このモジュールが依存するすべてのモジュールのコンマ区切りの一覧が含まれます。

注記

モジュールに依存関係がない場合は、出力から **depends** フィールドは省略される場合があります。

parm

parm フィールドは、***parameter_name*:*description*** の形式で 1 つのモジュールパラメーターを表します。

- *parameter_name* は、コマンドラインまたは**/etc/modprobe.d/** ディレクトリーの **.conf** ファイルのオプション行でモジュールパラメーターとして使用している場合に使用する構文と同一です。
- *description* は、パラメーターが使用する括弧内の値の種類 (int、unit、array of int など) に対する期待値と併せて、パラメーターが実行する内容の簡単な説明です。

-p オプションを使用すると、モジュールが対応するすべてのパラメーターを一覧表示できます。ただし、値のタイプを含む有用な情報は **modinfo -p** 出力から省略されるため、以下を実行した方がより役立ちます。

例24.2 モジュールパラメーターの一覧表示

```
~]# modinfo e1000e | grep "^parm" | sort
parm:           copybreak:Maximum size of packet that is copied
to a new buffer on receive (uint)
parm:           CrcStripping:Enable CRC Stripping, disable if
your BMC needs the CRC (array of int)
parm:           debug:Debug level (0=none,...,16=all) (int)
parm:           InterruptThrottleRate:Interrupt Throttling Rate
(array of int)
parm:           IntMode:Interrupt Mode (array of int)
parm:           KumeranLockLoss:Enable Kumeran lock loss
workaround (array of int)
parm:           RxAbsIntDelay:Receive Absolute Interrupt Delay
```

```
(array of int)
parm:           RxIntDelay:Receive Interrupt Delay (array of
int)
parm:           SmartPowerDownEnable:Enable PHY smart power
down (array of int)
parm:           TxAbsIntDelay:Transmit Absolute Interrupt Delay
(array of int)
parm:           TxIntDelay:Transmit Interrupt Delay (array of
int)
parm:           WriteProtectNVM:Write-protect NVM [WARNING:
disabling this can lead to corrupted NVM] (array of int)
```

24.3. モジュールの読み込み

カーネルモジュールを読み込みには、root で **modprobe *<module_name>*** を実行します。たとえば、**wacom** モジュールを読み込みには、以下を実行します。

```
~]# modprobe wacom
```

デフォルトでは、**modprobe** は **/lib/modules/*<kernel_version>*/kernel/drivers/** からモジュールの読み込みを試行します。このディレクトリー内には、各種モジュールにそれぞれのサブディレクトリーがあります。たとえば、ネットワークドライバーの場合は **net/**、SCSI インターフェースドライバーの場合は **scsi/** です。

一部のモジュールは依存関係を持っています、それらの依存関係は、モジュールが読み込まれる前に読み込み必要がある他のカーネルモジュールです。**modprobe** コマンドは、動作の実行時に常に依存関係を考慮に入れます。**modprobe** に特定のカーネルモジュールの読み込みを要求する場合、そのモジュールの依存関係があれば最初にそれらを調べ、カーネルにまだ読み込まれていない場合は読み込みます。**modprobe** は、再帰的に依存関係を解決します。つまり、依存関係が持つすべての依存関係を読み込み、さらにはそれらの依存関係を読み込むといったように、必要に応じて読み込みます。これにより、すべての依存関係が常に対応しているようにします。

-v (または **--verbose**) オプションを使用すると、**modprobe** が実行している内容の詳細を表示することができます。これには、モジュールの依存関係の読み込みが含まれる場合があります。以下は、**Fibre Channel over Ethernet** モジュールの詳細な読み込み例です。

例24.3 モジュールの依存関係が読み込まれる場合の modprobe -v による表示

```
~]# modprobe -v fcoe
insmod /lib/modules/3.10.0-
121.el7.x86_64/kernel/drivers/scsi/scsi_tgt.ko
insmod /lib/modules/3.10.0-
121.el7.x86_64/kernel/drivers/scsi/scsi_transport_fc.ko
insmod /lib/modules/3.10.0-
121.el7.x86_64/kernel/drivers/scsi/libfc/libfc.ko
insmod /lib/modules/3.10.0-
121.el7.x86_64/kernel/drivers/scsi/fcoe/libfcoe.ko
insmod /lib/modules/3.10.0-
121.el7.x86_64/kernel/drivers/scsi/fcoe/fcoe.ko
```

例24.3「モジュールの依存関係が読み込まれる場合の modprobe -v による表示」は、**modprobe** が **fcoe** を最後に読み込む前に **scsi_tgt**、**scsi_transport_fc**、**libfc**、**libfcoe** のモジュールを依存関係として読み込んだことを示しています。また、**modprobe** がよりプリミティブな **insmod** コマンドを使用して、実行中のカーネルにモジュールを挿入した点にも注意してください。

重要

insmod コマンドを使用してカーネルモジュールを読み込むこともできますが、依存関係は解決しません。そのため、代わりに **modprobe** を **常に** 使用して、モジュールを読み込むことをお勧めします。

24.4. モジュールのアンロード

カーネルモジュールをアンロードするには、root として **modprobe -r *<module_name>*** を実行します。たとえば、カーネルに **wacom** モジュールがすでにロードされていると仮定した場合、そのモジュールをアンロードするには以下を実行します。

```
~]# modprobe -r wacom
```

ただし、プロセスが以下のいずれかを使用している場合は、このコマンドは失敗します。

- **wacom** モジュール
- **wacom** が直接依存するモジュール
- 依存関係ツリーで **wacom** が間接的に依存するすべてのモジュール

lsmod を使用して、特定モジュールのアンロードを妨げているモジュール名を取得する方法についての情報は、「読み込み済みモジュールの一覧表示」を参照してください。

たとえば、**firewire_ohci** モジュールをアンロードしたい場合、ターミナルセッションは以下のようになります。

```
~]# modinfo -F depends firewire_ohci
firewire-core
~]# modinfo -F depends firewire_core
crc-itu-t
~]# modinfo -F depends crc-itu-t
```

これで、読み込み済みの Firewire モジュールに対する依存関係ツリーが明確になりました (この例では分岐していません)。**firewire_ohci** は **firewire_core** に依存しており、**firewire_core** は **crc-itu-t** に依存しています。

firewire_ohci をアンロードするには、**modprobe -v -r *<module_name>*** コマンドを使用します。**-r** は **--remove** を、**-v** は **--verbose** をそれぞれ省略したものです。

```
~]# modprobe -r -v firewire_ohci
rmmod firewire_ohci
rmmod firewire_core
rmmod crc_itu_t
```

アンロードされるモジュールに依存するプロセスがない場合、モジュールは読み込まれた時と逆の順番でアンロードされることをこの出力は示しています。

重要

rmmod コマンドを使用してカーネルモジュールをアンロードすることはできますが、その代わりに **modprobe -r** を使用することが推奨されます。

24.5. モジュールパラメーターの設定

カーネルのように、モジュールは動作を変更するパラメーターを取ることもできます。大半の場合、デフォルトのパラメーターは十分動作しますが、モジュールに対しカスタムパラメーターを設定することが必要な、あるいは望ましい場合があります。実行中のカーネルにすでに読み込まれたモジュールに対してパラメーターを動的に設定することはできないため、それらを設定する方法は 2 つあります。

1. パラメーターを設定したいモジュールが持つすべての依存関係のアンロード、**modprobe -r** を使用したモジュールのアンロード、そしてカスタマイズしたパラメーターの一覧と共に **modprobe** を使用したモジュールの読み込みを実行することができます。この方法がよく使用されるのは、モジュールに多くの依存関係がない場合、または様々なパラメーターの組み合わせを永続的にすることなくテスト目的で使用される場合です。これが本セクションで取り上げる方法です。

2. 別の方法では、**/etc/modprobe.d/** ディレクトリー内にある既存または新規作成のファイルに新しいパラメーターを一覧表示することもできます。モジュールが読み込まれるたびにモジュールパラメーターが設定されていることを確認することで、モジュールパラメーターを永続的にできます。例えば、リブートした後や **modprobe** コマンドを使用した後などです。以下の情報が前提条件ですが、この方法は 「永続的なモジュールの読み込み」 で説明します。

以下のコマンドライン形式を使用して、**modprobe** によりカスタムパラメーターでカーネルモジュールの読み込みができます。

例24.4 カーネルモジュールのロード時にオプションのパラメーターを指定

```
~]# modprobe <module_name> [parameter=value]
```

コマンドラインでカスタムパラメーターを使ってモジュールをロードする場合は、以下の点に注意してください。

- パラメーターと値を空白で区切ることで、複数入力が可能です。
- 引数としてコンマ区切りの値の一覧を必要とするモジュールパラメーターもあります。値の一覧を入力する場合、コンマの後に空白を **入力しないでください**。入力してしまうと、**modprobe** は空白に続く値を追加のパラメーターと間違って解釈することになります。
- 以下のいずれかの場合、**modprobe** コマンドは、成功すると何も表示せずにステータスは**0** という終了に変わります。
  - モジュールの読み込みが成功、**または**
  - モジュールが **すでに** カーネルに読み込み済み

  そのため、カスタムパラメーターを使ったモジュールの読み込み試行前に、モジュールがまだ読み込まれていないことを確認する必要があります。**modprobe** コマンドは、モジュールの自動的な再読み込みやすでにモジュールが読み込まれている旨の警告を行うことはありません。

カスタムパラメーターの設定およびカーネルモジュールの読み込みを行うにあたって推奨される手順は以下のとおりです。これは、例として Intel PRO/1000 ネットワークアダプター用のネットワークドライバーである **e1000e** モジュールを使用した手順について説明しています。

手順24.1 カスタムパラメーターによるカーネルモジュールの読み込み

1. 最初に、モジュールがカーネルにまだ読み込まれていないことを確認します。

```
~]# lsmod |grep e1000e
~]#
```

出力は、モジュールがカーネルにすでに読み込まれていることを示しています。この場合は、まずそれをアンロードする必要があります。安全にアンロードする方法については、「モジュールのアンロード」 を参照してください。

2. モジュールを読み込んで、モジュール名の後にカスタムパラメーターすべてを表示します。例えば、割り込みスロットル値をドライバーの第 1、第 2、第 3 インスタンスに対して毎秒 3000 割り込みに設定した Intel PRO/1000 ネットワークドライバーを読み込みたい場合は、**root** で以下を実行します。

```
~]# modprobe e1000e InterruptThrottleRate=3000,3000,3000 debug=1
```

この例では、複数の値をコンマで区切り、空白を省くことでこれらを 1 つのパラメーターにを渡すことを示しています。

24.6. 永続的なモジュールの読み込み

例24.1「lsmod を使用したカーネルモジュール情報の一覧表示」にあるように、カーネルモジュールの多くはブート時に自動的に読み込まれます。**/etc/sysconfig/modules/** ディレクトリー内に新しい ***<file_name>*.modules** ファイルを作成することで、読み込む追加のモジュールを指定できます。*<file_name>* は、分かりやすければどのような名前でも構いません。***<file_name>*.modules** ファイルは、システムのスタートアップスクリプトによりシェルスクリプトとして扱われ、1 行目は 「感嘆符の行」 とも呼ばれる *interpreter directive (解釈実行プログラム指示文)* で始まる必要があります。

例24.5 *file_name*.modules ファイルの 1 行目

```
#!/bin/sh
```

また、***<file_name>*.modules** ファイルは実行可能でなければなりません。実行可能なファイルに設定する方法は、以下のとおりです。

```
modules]# chmod +x <file_name>.modules
```

たとえば、以下の **bluez-uinput.modules** スクリプトは **uinput** モジュールを読み込みます。

例24.6 /etc/sysconfig/modules/bluez-uinput.modules

```
#!/bin/sh

if [ ! -c /dev/input/uinput ] ; then
        exec /sbin/modprobe uinput >/dev/null 2>&1
fi
```

3 行目の **if** の条件付きステートメントにより、**/dev/input/uinput** ファイルが **存在していない** ようにします (**!** の記号は条件を無効にします)。その場合は、**exec /sbin/modprobe uinput** を呼び出すことで **uinput** モジュールを読み込みます。**uinput** モジュールは **/dev/input/uinput** ファイルを作成するため、そのファイルが存在するかどうか調べることは、**uinput** モジュールがカーネルに読み込まれているかどうかを確認することを意味します。

その行の最後にある **>/dev/null 2>&1** は、すべての出力を **/dev/null** にリダイレクトするため、**modprobe** コマンドは使用されません。

24.7. セキュアブート用のカーネルモジュールの署名

Red Hat Enterprise Linux 7 は UEFI セキュアブート機能をサポートしているので、Red Hat Enterprise Linux 7 は UEFI セキュアブートが有効なシステムでのインストールおよび実行が可能です [3]。セキュアブートが有効になっていると、EFI オペレーティングシステムのブートローダー、Red Hat Enterprise Linux カーネル、およびすべてのカーネルモジュールが秘密鍵で署名され、それに対応する公開鍵で認証される必要があります。Red Hat Enterprise Linux 7 ディストリビューションには、署名されたブートローダー、カーネル、およびカーネルモジュールが含まれています。さらには、署名された第 1 ステージのブートローダーと署名されたカーネルには、埋め込み Red Hat 公開鍵が含まれています。これらの署名済みの実行可能なバイナリーと埋め込み鍵により、UEFI セキュアブートをサポートするシステム上で UEFI ファームウェアが提供する Microsoft UEFI セキュアブート CA 鍵を使って Red Hat Enterprise Linux 7 のインストール、ブート、および実行が可能になります [4]。

以下のセクションでは、セキュアブートが有効になっている UEFI ベースのシステム上で Red Hat Enterprise Linux 7 に使用する、プライベートで構築されたカーネルモジュールへの署名に必要なステップを説明しています。また、カーネルモジュールの実装を希望するターゲットシステムに公開鍵を届けるために利用可能なオプションについても説明しています。

24.7.1. 要件

外部で構築されたモジュールの署名を有効にするには、システムに以下の表で示すツールをインストールする必要があります。

表24.1 必要なツール

ツール	提供するパッケージ	使用対象	目的
openssl	*openssl*	ビルドシステム	公開および秘密 X.509 鍵のペアを生成
sign-file	*kernel-devel*	ビルドシステム	カーネルモジュールの署名に使用する Perl スクリプト
perl	*perl*	ビルドシステム	署名スクリプトの実行に使用する Perl インタープリター
mokutil	*mokutil*	ターゲットシステム	公開鍵を手動で登録する際に使用するオプションツール

ツール	提供するパッケージ	使用対象	目的
keyctl	*keyutils*	ターゲットシステム	システム鍵リンクに公開鍵を表示する際に使用するオプションツール

注記

カーネルモジュールを構築、署名するビルドシステムは、UEFI セキュアブートを有効にする必要がなく、UEFI ベースのシステムである必要すらないことに注意してください。

24.7.2. カーネルモジュールの認証

Red Hat Enterprise Linux 7 では、カーネルモジュールの読み込み時に、カーネルのシステムキーリング上の公開 x.509 鍵を使ってモジュールの署名をチェックします。使用される鍵は、カーネルのシステムブラックリストのキーリングにあるものを除きます。

24.7.2.1. カーネルモジュール認証に使用する公開鍵のソース

ブート中にカーネルは、表24.2「システムキーリングのソース」にある永続的キーストア一式から X.509 鍵をシステムキーリングまたはシステムのブラックリストに読み込みます。

表24.2 システムキーリングのソース

X.509 鍵のソース	キー追加に関するユーザー能力	UEFI セキュアブートの状態	ブート中に読み込まれる鍵
カーネルに埋め込み	×	-	**.system_keyring**
UEFI セキュアブート "db"	限定的	有効でない	×
		有効	**.system_keyring**
UEFI セキュアブート "dbx"	限定的	有効でない	×
		有効	**.system_keyring**
shim.efi ブートローダーに埋め込み	×	有効でない	×
		有効	**.system_keyring**
Machine Owner Key (MOK) リスト	○	有効でない	×
		有効	**.system_keyring**

システムが UEFI ベースでない場合、もしくは UEFI セキュアブートが有効でない場合は、カーネルに埋め込まれた鍵のみがシステムのキーリングに読み込まれ、カーネルの再構築なしにはこの鍵セットを増やすことはできません。システムのブラックリストキーリングは、無効にされた X.509 鍵の一覧です。ブラックリストにある鍵でモジュールが署名されていると、公開鍵がシステムのキーリングにあったとしても、モジュールは認証に失敗します。

システムのキーリング上にある鍵についての情報は、**keyctl** ユーティリティーを使うと表示できます。以下は、UEFI セキュアブートが有効でない Red Hat Enterprise Linux 7 システムからの出力の省略形です。

```
~]# keyctl list %:.system_keyring
3 keys in keyring:
...asymmetric: Red Hat Enterprise Linux Driver Update Program (key 3):
bf57f3e87...
```

```
...asymmetric: Red Hat Enterprise Linux kernel signing key:
4249689eefc77e95880b...
...asymmetric: Red Hat Enterprise Linux kpatch signing key:
4d38fd864ebe18c5f0b7...
```

以下は、UEFI セキュアブートが有効になっている Red Hat Enterprise Linux 7 システムからの出力の省略形です。

```
~]# keyctl list %:.system_keyring
6 keys in keyring:
...asymmetric: Red Hat Enterprise Linux Driver Update Program (key 3):
bf57f3e87...
...asymmetric: Red Hat Secure Boot (CA key 1):
4016841644ce3a810408050766e8f8a29...
...asymmetric: Microsoft Corporation UEFI CA 2011:
13adbf4309bd82709c8cd54f316ed...
...asymmetric: Microsoft Windows Production PCA 2011:
a92902398e16c49778cd90f99e...
...asymmetric: Red Hat Enterprise Linux kernel signing key:
4249689eefc77e95880b...
...asymmetric: Red Hat Enterprise Linux kpatch signing key:
4d38fd864ebe18c5f0b7...
```

上記の出力では、UEFI セキュアブート "db" 鍵から加わった 2 つの鍵と 「Red Hat Secure Boot(CA key 1)」 が示されています。後者は、**shim.efi** ブートローダーに埋め込まれています。また、UEFI セキュアブート db である UEFI セキュアブート関連のソース、埋め込み shim、および MOK リストで鍵を特定するカーネルコンソールメッセージを探すこともできます。

```
~]# dmesg | grep 'EFI: Loaded cert'
[5.160660] EFI: Loaded cert 'Microsoft Windows Production PCA 2011:
a9290239...
[5.160674] EFI: Loaded cert 'Microsoft Corporation UEFI CA 2011:
13adbf4309b...
[5.165794] EFI: Loaded cert 'Red Hat Secure Boot (CA key 1):
4016841644ce3a8...
```

24.7.2.2. カーネルモジュール認証の要件

UEFI セキュアブートが有効になっているか **module.sig_enforce** カーネルパラメーターが指定されている場合、システムのキーリング上の鍵を使って認証された署名済みカーネルモジュールのみが正常に読み込まれます [5]。UEFI セキュアブートが無効でかつ **module.sig_enforce** カーネルパラメーターが指定されていないと、署名されていないカーネルモジュールや公開鍵のない署名済みカーネルモジュールが正常に読み込まれます。これをまとめたものが 表24.3「カーネルモジュールの読み込み認証要件」 です。

表24.3 カーネルモジュールの読み込み認証要件

モジュールの署名	公開鍵ありおよび署名が有効	UEFI セキュアブートの状態	module.sig_enforce	モジュールの読み込み	カーネルの汚染
署名なし	-	有効でない	有効でない	成功	○
		有効でない	有効	失敗	
		有効	-	失敗	-
署名あり	×	有効でない	有効でない	成功	○
		有効でない	有効	失敗	-

モジュールの署名	公開鍵ありおよび署名が有効	UEFI セキュアブートの状態	module.sig_enforce	モジュールの読み込み	カーネルの汚染
		有効	-	失敗	-
署名あり	○	有効でない	有効でない	成功	×
		有効でない	有効	成功	×
		有効	-	成功	×

以下のセクションでは、使用する公開および秘密 X.509 鍵のペアの生成方法、秘密鍵を使ってカーネルモジュールを署名する方法、システムのキーリング用に公開鍵をソースに登録する方法を説明します。

24.7.3. 公開および秘密 X.509 鍵のペアの生成

カーネルモジュールを構築したら、公開および秘密 X.509 鍵のペアを生成してこれに署名する必要があります。対応する公開鍵は、カーネルモジュールの読み込み時にこれを認証するために使用されます。

1. Red Hat Enterprise Linux 7 でのカーネルモジュール署名の要件を満たす鍵ペアの生成には、**openssl** ツールを使うことができます。この鍵生成リクエストのパラメーターの中には、設定ファイルで指定するべきものがいくつかあります。以下の例にしたがって、使用する設定ファイルを作成してください。

```
~]# cat << EOF > configuration_file.config
[ req ]
default_bits = 4096
distinguished_name = req_distinguished_name
prompt = no
string_mask = utf8only
x509_extensions = myexts

[ req_distinguished_name ]
O = Organization
CN = Organization signing key
emailAddress = E-mail address

[ myexts ]
basicConstraints=critical,CA:FALSE
keyUsage=digitalSignature
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid
EOF
```

2. 設定ファイルを作成したら、X.509 公開および秘密鍵のペアを作成できます。公開鍵は **<file_name>.der** ファイルに、秘密鍵は **<file_name>.priv** ファイルに書き込まれます。

```
openssl req -x509 -new -nodes -utf8 -sha256 -days 36500 \
> -batch -config configuration_file.config -outform DER \
> -out public_key.der \
> -keyout private_key.priv
```

3. カーネルモジュールを認証、読み込むすべてのシステムに公開鍵を登録します。

警告

秘密鍵のコンテンツは、適切に保護してください。秘密鍵が間違った人間に渡ると、公開鍵のあるシステムが危険にさらされる可能性があります。

24.7.4. 公開鍵のターゲットシステムでの登録

セキュアブートが有効になっている UFEI ベースのマシンで Red Hat Enterprise Linux 7 を起動すると、無効にされたキーの dbx データベースにあるものを除いて、セキュアブート db キーデータベースにあるすべてのキーがカーネルによってシステムのキーリングに読み込まれます。システムキーリングは、カーネルモジュールの認証に使われます。

24.7.4.1. 公開鍵を含むファクトリーファームウェアイメージ

ターゲットシステムでカーネルモジュールの認証を促す最も容易な方法は、システムベンダーがファクトリーファームウェアイメージで公開鍵を UEFI セキュアブート鍵データベースに組み入れることです。

24.7.4.2. 公開鍵を追加する実行可能な鍵の登録イメージ

鍵を既存の設定済みでアクティブなセキュアブート鍵データベースに追加することは可能です。これは、EFI 実行可能 **enrollment** (登録) イメージを作成、提供することで実行できます。このような登録イメージにはセキュアブート鍵データベースへの鍵に追加する、適切に形成されたリクエストが含まれます。このリクエストには、システムのセキュアブート Key Exchange Key (KEK) データベースに既にある公開鍵に対応する秘密鍵によって適切に署名されたデータを含める必要があります。さらに、この EFI イメージは、鍵データベースに既にある公開鍵に対応する秘密鍵で署名されている必要があります。

Red Hat Enterprise Linux 7 で稼働する登録イメージを作成することもできます。しかし、Red Hat Enterprise Linux 7 イメージは、すでに KEK データベースにある公開鍵に対応する秘密鍵で適切に署名されている必要があります。

このため、いずれのタイプの鍵登録イメージの構築もプラットホームベンダーからの手助けが必要になります。

24.7.4.3. システム管理者が手動で公開鍵を MOK リストに追加する

Machine Owner Key (MOK) 機能は Red Hat Enterprise Linux 7 がサポートしており、これを使用して UEFI セキュアブート鍵データベースを増やすことができます。セキュアブートが有効になっている UEFI ベースのシステムで Red Hat Enterprise Linux 7 が起動すると、鍵データベースの鍵に加えて、MOK リストの鍵もシステムキーリングに追加されます。MOK リストの鍵は、セキュアブート鍵データベースの鍵と同様に永続的かつ安全な方法で保存されますが、これらは別個の機能です。MOK 機能は、shim.efi、MokManager.efi、grubx64.efi、および Red Hat Enterprise Linux 7 **mokutil** ユーティリティーでサポートされています。

MOK 機能が提供する主な機能は、キーチェーンをすでに KEK データベースにある別の鍵に返す必要なしに、公開鍵を MOK リストに加えることができる点です。しかし、MOK 鍵の登録は、各ターゲットシステムの UEFI システムコンソールでユーザーが**物理的に**手動で対応する必要があります。それにもかかわらず、MOK 機能は、新規生成された鍵ペアのテストとこれで署名されたカーネルモジュールのテストにおいて優れた方法を提供します。

MOK リストに公開鍵を追加するには、以下の手順にしたがいます。

1. Red Hat Enterprise Linux 7 ユーザースペースユーティリティーを使って、公開鍵の MOK リストへの追加をリクエストします。

```
~]# mokutil --import my_signing_key_pub.der
```

この MOK 登録リクエストに関するパスワードの入力と確認が求められます。

2. マシンを再起動します。
3. この MOK 鍵登録リクエストは **shim.efi** が発見し、**MokManager.efi** を起動して UEFI コンソールからの登録が完了できるようになります。このリクエストに関連付けたパスワードを入力し、登録を確認します。公開鍵が MOK リストに永続的に追加されます。

鍵が MOK リストに追加されると、UEFI セキュアブートが有効になっているシステムの起動時に毎回、この鍵はシステムのキーリングに自動的に追加されます。

24.7.5. 秘密鍵を使用したカーネルモジュールの署名

カーネルモジュールの署名を準備するには、追加の手順は必要ありません。カーネルモジュールは通常通りに構築します。適切な Makefile と対応するソースがあれば、以下の手順でモジュールを構築して署名します。

1. **my_module.ko** モジュールを通常の方法で構築します。

```
~]# make -C /usr/src/kernels/$(uname -r) M=$PWD modules
```

2. 秘密鍵がカーネルモジュールに署名します。これは、Perl スクリプトで行います。このスクリプトでは、署名するカーネルモジュールファイルの他に、秘密および公開鍵の両方を含むファイルを提供する必要があることに注意してください。

```
~]# perl /usr/src/kernels/$(uname -r)/scripts/sign-file \
> sha256 \
> my_signing_key.priv \
> my_signing_key_pub.der \
> my_module.ko
```

カーネルモジュールは ELF イメージ形式で、このスクリプトは署名を計算して、**my_module.ko** ファイルの ELF イメージに直接追加します。**modinfo** ユーティリティーを使うと、カーネルモジュールの署名がある場合は、それについての情報を表示できます。このユーティリティーの使用方法については、「モジュール情報の表示」 を参照してください。

この追加された署名は ELF イメージセクションには含まれず、また ELF イメージの正式な一部ではないことに注意してください。このため、**readelf** のようなツールは、この署名をカーネルモジュールに表示することができません。

これでカーネルモジュールの読み込み準備が完了しました。署名済みのカーネルモジュールは、UEFI セキュアブートが無効となっているシステムまたは UEFI 以外のシステムでも読み込み可能であることに注意してください。つまり、署名済みのカーネルモジュールと署名なしのカーネルモジュールの両方を提供する必要はないことになります。

24.7.6. 署名済みカーネルモジュールの読み込み

公開鍵が登録されてシステムキーリングに格納されると、通常のカーネルモジュール読み込みのメカニズムが透過的に作動します。以下の例では、**mokutil** を使って公開鍵を MOK リストに追加し、カーネルモジュールを **modprobe** を使って手動で読み込みます。

1. オプションでは、公開鍵の登録前にカーネルモジュールを読み込まないことを確認できます。まず、**keyctl list %:.system_keyring** を root で実行して、現在のブートでどの鍵がシステムキーリングに追加されているかを確認します。公開鍵はまだ登録されていないので、このコマンドの出力には表示されません。

2. 公開鍵の登録をリクエストします。

```
~]# mokutil --import my_signing_key_pub.der
```

3. 再起動して、UEFI コンソールでの登録を完了します。

```
~]# reboot
```

4. システムの再起動後に、再度システムキーリングの鍵を確認します。

```
~]# keyctl list %:.system_keyring
```

5. これでカーネルモジュールを正常に読み込むことができます。

```
~]# modprobe -v my_module
insmod /lib/modules/3.10.0-123.el7.x86_64/extra/my_module.ko
~]# lsmod | grep my_module
my_module 12425 0
```

24.8. その他のリソース

カーネルモジュールとそれらのユーティリティーの詳細については、以下のリソースを参照してください。

man ページ

- **man lsmod** — **lsmod** コマンドの man ページです。
- **man modinfo** — **modinfo** コマンドの man ページです。
- **man modprobe** — **modprobe** コマンドの man ページです。
- **man rmmod** — **rmmod** コマンドの man ページです。
- **man ethtool** — **ethtool** コマンドの man ページです。
- **man mii-tool** — **mii-tool** コマンドの man ページです。

インストールできる外部のドキュメント

- Linux Loadable Kernel Module HOWTO — Linux Documentation Project の 『Linux Loadable Kernel Module HOWTO』 には、カーネルモジュールでの作業に関する詳しい情報が記載されています。

[3] Red Hat Enterprise Linux 7 は UEFI システム上でセキュアブートの使用を必要としません。

[4] すべての UEFI ベースのシステムにセキュアブートのサポートが含まれるわけではありません。

[5] 公開鍵がシステムのブラックリストキーリング上にないことが条件。

付録A RPM

RPM Package Manager (RPM パッケージマネージャー) (**RPM**) は、オープンパッケージングシステムで、Red Hat Enterprise Linux だけでなく他の Linux や UNIX システム上でも動作します。Red Hat と Fedora Project では、他のベンダーに対し、自社製品に **RPM** を使用するよう推奨しています。**RPM** は *GPL* (*GNU 一般公有使用許諾*) 契約に基づいて配布されます。

RPM Package Manager は、**RPM 形式** でビルドされたパッケージのみと作動します。**RPM** 自体は、プレインストールされた *rpm* パッケージとして提供されています。エンドユーザーにとっては、**RPM** でシステム更新が容易になります。**RPM** パッケージのインストール、アンインストール、アップグレードは短いコマンドで実行できます。**RPM** は、インストール済みのパッケージやそのファイルに関するデータベースを維持しているため、システムで強力なクエリや検証を実行することができます。**Yum** や **PackageKi** などのアプリケーションは、**RPM** 形式のパッケージとの作業をより簡素化します。

警告

ほとんどのパッケージ管理タスクでは、**Yum** パッケージマネジャーが **RPM** と同等かそれ以上の機能とユーティリティーを提供します。**Yum** は、複雑なシステムの依存関係解決の実行と追跡も行います。**Yum** はシステムの整合性を維持し、**Yum** ではなく **RPM** のような別のアプリケーションを使用してパッケージがインストールもしくは削除されると、システム整合性チェックを強制します。このため、パッケージ管理タスクを実行する際は、可能な限り **RPM** ではなく **Yum** を使用することが強く推奨されます。5章*Yum* を参照してください。

グラフィカルインターフェースを希望する場合は、**PackageKit** GUI アプリケーションを使用できます。これは、**yum** をバックエンドとして使用して、システムのパッケージを管理します。

RPM はアップグレード時に設定ファイルを慎重に処理するため、カスタマイズ情報が失われることはありません。通常の **.tar.gz** ファイルを使用した場合は、このような機能を実現することはできません。

開発者は **RPM** を使用することで、ソフトウェアのソースコードを取り出し、エンドユーザー用にソースパッケージとバイナリパッケージにパッケージ化することができます。このプロセスは極めて単純で、1 つのファイルとユーザーが作成するオプションのパッチを基に実行されます。ビルドの指示に加えて、純粋なソースとパッチの明確な説明により、ソフトウェアの新バージョンリリース時にパッケージの維持が容易になります。

注記

RPM は、システム自体に変更を加えることができるため、システムワイドでのバイナリーパッケージのインストールやアップグレード、ダウングレード、およびアンインストールといった操作には、ほとんどの場合で **root** 権限が必要になります。

A.1. RPM の設計目標

RPM の設計目標を理解すると、**RPM** の使用の際に役立ちます。

アップグレードの可能性

RPM を使用すると、コンポーネントの完全な再インストールなしに、個別のコンポーネントをアップグレードができます。Red Hat Enterprise Linux などの **RPM** に基づくオペレーティングシステムの新しいリリースを入手すると、マシンにオペレーティングシステムの最新コピーを再

インストールする必要がありません（他のパッケージングシステムに基づくオペレーティングシステムの場合はその必要があります）。**RPM** を使用すれば、インテリジェントで、完全自動化されたインプレースのシステムのアップグレードを実行できます。その上、パッケージに含まれる設定ファイルはアップグレード後にも保持されるので、カスタマイズ情報が失われることはありません。システム上のパッケージのインストールとアップグレードには、同じ **RPM** ファイルが使用されるため、パッケージのアップグレードに特別なアップグレードファイルが必要になることはありません。

強力なクエリ

RPM では、強力なクエリのオプションが提供されています。データベースを通じてパッケージを検索したり、特定のファイル群のみを検索したりすることができます。また、あるファイルがどのパッケージに属し、どこから来たのかをも簡単に検索できます。**RPM** パッケージに含まれるファイルは圧縮アーカイブ形式であり、パッケージとそのコンテンツに関する有用な情報を含むカスタムバイナリヘッダが付いているため、個別のパッケージをすばやく簡単にクエリすることができます。

システムの検証

RPM のもう 1 つの強力な機能は、パッケージ検証を行う能力です。システムにインストールされているファイルが特定パッケージで提供されるものと同一かどうかを検証することができます。不整合が検出されると、**RPM** は通知するので、必要に応じてパッケージを再インストールすることができます。再インストールを行っても、修正した設定ファイルは保持されます。

純粋なソース

極めて重要な設計目標は、ソフトウェアのオリジナル作者が配布した時のままの **純粋な** ソフトウェアソースをユーザーが利用できるようにすることです。**RPM** を使用すると、純粋なソースと適用済みのパッチ、完全なビルド指示が入手できます。多くの理由で、これは大きなメリットとなります。たとえば、あるプログラムの新しいバージョンがリリースされた場合、それをコンパイルするために必ずしも最初から作業を始める必要はなくなります。パッチを見て、必要になる **可能性のある** 作業を確かめることができます。この技術を使えば、コンパイル済みのすべてのデフォルト設定と、ソフトウェアを適切に構築するために行われたすべての変更内容を容易に認識できます。

ソースを純粋な状態に保持するという目標は、開発者にとってのみ重要なことのように思われるかもしれませんが、結果として、エンドユーザーにより高品質のソフトウェアが提供されることにつながります。

A.2. RPM の使用法

RPMには、次の 5 つの基本的作動モードがあります (パッケージ構築は除くカウントしません) : インストール、アンインストール、アップグレード、クエリ、検証です。このセクションでは、各モードについて概説します。詳細な説明やオプションについては **rpm --help** または rpm(8) をご覧ください。**RPM** の詳細については、「その他のリソース」も参照してください。

A.2.1. パッケージのインストールとアップグレード

RPM パッケージには通常、**tree-1.6.0-10.el7.x86_64.rpm** のようなファイル名が付けられています。ファイル名は、パッケージ名 (**tree**)、バージョン (**1.6.0**)、リリース (**10**)、オペレーティングシステムのメジャーバージョン (**el7**)、および CPU アーキテクチャー (**x86_64**) で構成されています。

重要

パッケージのインストール時には、パッケージに使用中のオペレーティングシステムとプロセッサーのアーキテクチャーとの互換性があることを確かめてください。通常は、パッケージ名を確認すると分かります。たとえば、AMD64/Intel 64 コンピューターのアーキテクチャー用にコンパイルされた **RPM** パッケージの名前は、**x86_64.rpm** で終わります。

-U (または **--upgrade**) オプションを使うと、以下が実行できます。

- システム上にの既存パッケージを新しいバージョンにアップグレードする、または
- 旧バージョンがインストールされていない場合、パッケージをインストールする

つまり、**rpm -U *package.rpm*** は、パッケージの必要性に応じて **アップグレード** または **インストール** のいずれかの機能を実行できます。

tree-1.6.0-10.el7.x86_64.rpm パッケージが現在のディレクトリーにあると仮定します。**root** としてログインし、*tree* パッケージをアップグレードまたはインストールするためにシェルプロンプトで以下のコマンドを入力します。

```
~]# rpm -Uvh tree-1.6.0-10.el7.x86_64.rpm
```

注記

-v および **-h** のオプション（**-U** と組み合わされる）により、**rpm** は詳細出力を表示し、ハッシュ記号で進行状況が表示されます。

アップグレードまたはインストールが正常に終了すると、以下のような出力が表示されます。

```
Preparing...                ########################################### [100%]
Updating / installing...
   1:tree-1.6.0-10.el7      ########################################### [100%]
```

警告

rpm は、パッケージをインストールする際に 2 つのオプションを提供します。前述の**-U** オプション (従来は **アップグレード** を意味します) と **インストール** を意味する **-i** オプションです。**-U** オプションには、インストールとアップグレードの両機能が含まれるため、***kernel* パッケージ以外の**すべてのパッケージに **rpm -Uvh** を使用することが推奨されます。

新しいカーネルパッケージはアップグレードするのではなく、**インストール**するための **-i** オプションを常に使ってください。カーネルパッケージをアップグレードする **-U** オプションを使用すると、以前の (旧) カーネルパッケージが削除されるため、新しいカーネルに問題がある場合は、システムがブートできない状態になる可能性があります。そのため、新しいカーネルをインストールするには、**古い *kernel* パッケージを置換せずに**、**rpm -i *kernel_package*** コマンドを使用します。*kernel* パッケージのインストールについては、23章 *手動でのカーネルのアップグレード* を参照してください。

パッケージの署名は、パッケージのインストールまたはアップグレード時に自動的に確認されます。署名により、パッケージが認証機関によって署名されたことが確認できます。署名の検証が失敗すると、エラーメッセージが表示されます。

署名を検証するための適切なキーをインストールしていない場合は、メッセージに **NOKEY** という単語が含まれます。

```
warning: tree-1.6.0-10.el7.x86_64.rpm: Header V3 RSA/SHA256 Signature,
key ID 431d51: NOKEY
```

パッケージ署名の確認に関しては、「パッケージ署名の確認」を参照してください。

A.2.1.1. インストール済みパッケージの置換

同一名で同一バージョンのパッケージがすでにインストールされていると、以下のような出力が表示されます。

```
Preparing...                ########################################### [100%]
        package tree-1.6.0-10.el7.x86_64 is already installed
```

このパッケージをいずれにしてもインストールするには、**--replacepkgs** オプションを使って **RPM** にエラーを無視するよう指示します。

```
~]# rpm -Uvh --replacepkgs tree-1.6.0-10.el7.x86_64.rpm
```

このオプションは、パッケージからインストールされたファイルが削除された場合や、オリジナルの設定ファイルをインストールしたい場合に便利です。

古い バージョン番号のパッケージにアップグレードしている場合 (つまり、パッケージの新しいバージョンがすでにインストールされている場合) は、**RPM** は新しいバージョンが既にインストールされていることを知らせます。**RPM** にダウングレードを強制するには、**--oldpackage** オプションを使用します。

```
rpm -Uvh --oldpackage older_package.rpm
```

A.2.1.2. ファイル競合の解決

別のパッケージによってすでにインストール済みのファイルを含むパッケージをインストールしようとすると、競合メッセージが表示されます。**RPM** がこのエラーを無視するようにするには、**--replacefiles** オプションを使います。

```
rpm -Uvh --replacefiles package.rpm
```

A.2.1.3. 未解決の依存関係の解消

RPM パッケージが他のパッケージに依存することも時々あります。つまり、正しく動作するために他のパッケージのインストールが必要になります。未解決の依存関係を持つパッケージをインストールしようとすると、依存関係のエラーを示すメッセージが表示されます。

Red Hat Enterprise Linux インストールメディア、または アクティブな Red Hat Enterprise Linux ミラーで提案されるパッケージを探し、これをインストールコマンドに追加します。必要なファイルがどのパッケージに含まれているかを判断するには、**--whatprovides** オプションを使用します。

```
rpm -q --whatprovides "required_file"
```

required_file を含むパッケージが **RPM** データベースにあれば、パッケージ名が表示されます。

警告

未解決の依存関係を持つパッケージをインストールするように **rpm** に **強制** することは可能ですが (**--nodeps** オプションを使用)、これは推奨**されません**。通常、インストール済みのソフトウェアが実行できなくなります。**--nodeps** オプションを使ってパッケージをインストールすると、アプリケーションが誤作動もしくはクラッシュする可能性があります。また、深刻なパッケージ管理の問題もしくはシステム障害が発生する可能性もあります。このため、依存関係の欠如についての警告には留意してください。**Yum** パッケージマネジャーは、自動的に依存関係を解決し、オンラインのリポジトリーから依存関係を取得します。

A.2.1.4. 設定ファイルの変更を維持する

RPM は設定ファイルを使用してパッケージのインテリジェントなアップグレードを実行するため、以下のメッセージが表示される場合があります。

```
saving /etc/configuration_file.conf as
/etc/configuration_file.conf.rpmsave
```

このメッセージは、設定ファイルに加えた変更内容が、パッケージ内の新しい設定ファイルと **上位互換性** を持たない可能性があるため、**RPM** が元のファイルを保存してから新しいファイルをインストールしたことを意味しています。できる限り早くこの 2 つのファイルの違いを調査、解決し、引き続きシステムが正しく動作することを確認する必要があります。

別の方法では、**RPM** がパッケージの **新しい** 設定ファイルを、例えば ***configuration_file.conf*.rpmnew** として保存し、修正した設定ファイルをそのままにしておくこともできます。なお、例えば **diff** プログラムを使うことで古い設ファイルから新しい設定ファイルへの変更を組み入れるこで、通常は、修正した設定ファイルと新しい設定ファイルとの競合を解決する必要があります。

A.2.2. パッケージのアンインストール

パッケージのアンインストールは、インストールと同様、簡単に実行できます。**root** としてシェルプロンプトで以下のコマンドを入力します。

```
rpm -e package
```

注記

アンインストール時に指定するのは、オリジナルのパッケージの **ファイル**名ではなく、パッケージの **名前** であることに注意してください。**rpm -e** コマンドでオリジナルの完全なファイル名を提供してパッケージをアンインストールしようとすると、パッケージ名のエラーが表示されます。

削除しようとしているパッケージに別のインストール済みのパッケージが依存している場合、アンインストール時に依存関係のエラーが発生することがあります。例えば、次のように表示されます。

```
~]# rpm -e ghostscript
error: Failed dependencies:
        ghostscript is needed by (installed) ghostscript-cups-9.07-
16.el7.x86_64
        ghostscript is needed by (installed) foomatic-4.0.9-6.el7.x86_64
        libgs.so.9()(64bit) is needed by (installed) libspectre-0.2.7-
4.el7.x86_64
        libijs-0.35.so()(64bit) is needed by (installed) gutenprint-
5.2.9-15.el7.x86_64
        libijs-0.35.so()(64bit) is needed by (installed) cups-filters-
1.0.35-15.el7.x86_64
```

警告

未解決の依存関係を持つパッケージのアンインストールを **rpm** に **強制** することは可能ですが (**--nodeps** オプションを使用)、これは推奨**されません**。**--nodeps** オプションを使ってパッケージを削除すると、依存関係が削除されたパッケージからのアプリケーションが誤作動もしくはクラッシュする可能性があります。また、深刻なパッケージ管理の問題もしくはシステム障害が発生する可能性もあります。このため、依存関係の失敗についての警告には留意してください。

A.2.3. パッケージを最新の状態にする

パッケージを最新の状態にする作業は、インストール済みのパッケージのみがアップグレードされていることを除いては、アップグレードと似ています。**root** としてシェルプロンプトで以下のコマンドを入力します。

```
rpm -Fvh package.rpm
```

-F (または **--freshen**) オプションは、コマンドラインで指定されたパッケージのバージョンと、すでにシステムにインストールされているパッケージのバージョンを比較します。インストール済みのパッケージよりも新しいバージョンのパッケージが **--freshen** オプションで処理されると、そのパッケージは新しいバージョンへアップグレードされます。ただし、**--freshen** では、同じ名前のインストール済みパッケージが存在しない場合は、パッケージはインストールされません。これは通常のアップグレードと異なる点です。アップグレードでは、古いバージョンのパッケージがインストール済みかどうかにかかわらず、指定されたパッケージはすべてインストールされます。

freshen オプションは、単一パッケージでもパッケージグループでも機能します。たとえば、多くの異なるパッケージをダウンロードして、システム上にインストール済みのパッケージのみをアップグレードしたい場合などに、このオプションは便利です。このケースでは、以下のコマンドを ***.rpm** グロブと発行します。

```
~]# rpm -Fvh *.rpm
```

すると **RPM** は、すでにインストールされているパッケージのみを自動的にアップグレードします。

A.2.4. パッケージのクエリ

RPM のデーターベースにはシステムにインストール済みのすべての**RPM** パッケージに関する情報が保存されています。それらは、**/var/lib/rpm/** ディレクトリー内にあり、どのパッケージがインストール済みで、どのパッケージがどのバージョンなのかをクエリしたり、またインストール以降にパッケージ内のファイルに加えられた変更を算出する際など、多くの目的で使用されます。このデータベースにクエリを行うには、 **rpm** コマンドに **-q** (または **--query**) オプションを付けて実行します。

```
rpm -q package_name
```

このコマンドは、インストール済みパッケージ *package_name* のパッケージ名、バージョン、およびリリース番号を表示します。例を示します。

```
~]$ rpm -q tree
tree-1.6.0-10.el7.x86_64
```

クエリをさらに細かくするもしくは限定するには、rpm(8) man ページの **Package Selection Options** 小見出しにあるオプション一覧を参照してください。この**Package Query Options** 小見出しにあるオプションを使うと、クエリを行ったパッケージについて表示する情報を指定することができます。

A.2.5. パッケージの検証

パッケージの検証では、パッケージからインストールされたファイルに関する情報と、元のパッケージからの同一情報を比較します。ファイルのサイズ、MD5 サム、パーミッション、タイプ、所有者、各ファイルのグループなどが比較されます。

パッケージを検証するには、**rpm** コマンドに **-V** (または **--verify**) オプションを付けて使用します。例を示します。

```
~]$ rpm -V tree
```

クエリをさらに細かくするもしくは限定するには、rpm(8) man ページの **Package Selection Options** 小見出しにあるオプション一覧を参照してください。この**Verify Options** 小見出しにあるオプションを使うと、クエリを行ったパッケージで検証する特徴について指定することができます。

すべてが正常に検証された場合は何も出力されません。何らかの矛盾が見つかった場合はその内容が表示されます。出力は、以下のような行で表示されます。

```
~]# rpm -V abrt
S.5....T.  c /etc/abrt/abrt.conf
.M.......    /var/spool/abrt-upload
```

出力フォーマットは、9 個の文字列にオプションの属性マーカー、処理されたファイル名が続きます。

最初の 9 文字は、ファイルで実行されたテスト結果を示します。各テストでは、ファイルのある属性と、**RPM** データベースに記録されたその属性の値が比較されます。シングルピリオド (**.**) は、テストに合格したことを示し、クエスチョンマーク (**?**) は、テストが実行できなかったことを示します。以下の表では、特定の矛盾を示す記号になります。

表A.1 RPM 検証の記号

記号	詳細
S	ファイルサイズの相違
M	モードの相違 (パーミッションおよびファイルタイプを含む)
5	ダイジェスト (以前の MD5 サム) の相違
D	デバイスのメジャー/マイナー番号の不一致
L	readLink(2) パスの不一致
U	ユーザー所有権の相違
G	グループ所有権の相違
T	mtime の相違

記号	詳細
P	機能の相違

属性マーカーがある場合は、特定ファイルの目的を説明しています。以下の表では、利用可能な属性マーカーを一覧表示しています。

表A.2 RPM 検証の記号

マーカー	詳細
c	設定ファイル
d	ドキュメンテーションファイル
l	ライセンスファイル
r	readme ファイル

何かが出力された場合は、パッケージを削除するか、再インストールするか、または別の方法で問題を修正するかを熟慮の上判断してください。

A.3. RPM パッケージの検索および検証

RPM パッケージの使用前に、それがどこにあるか、それらが信頼できるかどうかを調べる方法を知る必要があります。

A.3.1. RPM パッケージの検索

インターネット上には多くの **RPM** リポジトリーがありますが、セキュリティーおよび互換性の面から、Red Hat が提供する公式の RPM パッケージのみをインストールするようにしてください。以下は、**RPM** パッケージのソースリストです。

- 公式 Red Hat Enterprise Linux インストールメディア。
- **Yum** パッケージマネージャーで提供される公式の **RPM** リポジトリー。公式の Red Hat Enterprise Linux パッケージリポジトリーの使用方法については、5章*Yum* を参照してください。
- カスタマーポータルの Red Hat エラーページ https://rhn.redhat.com/rhn/errata/RelevantErrata.do。
- Enterprise Linux 用エキストラパッケージ (EPEL) は、Red Hat Enterprise Linux 用に高品質のアドオンパッケージのリポジトリーを提供するコミュニティー作業です。EPEL **RPM** パッケージの詳細については http://fedoraproject.org/wiki/EPEL を参照してください。
- Red Hat に関連しない非公式のサードパーティーのリポジトリーも RPM パッケージを提供します。

重要

Red Hat Enterprise Linux システムでの使用にサードパーティーのリポジトリーを検討している場合は、パッケージソースとしてリポジトリーを追加する前に、パッケージの互換性に関するリポジトリーのウェブサイトに注意を払ってください。代替のパッケージリポジトリーは、Red Hat Enterprise Linux リポジトリーにすでに含まれているパッケージなどと同じソフトウェアでも互換性のない異なるバージョンを提供することがあります。

A.3.2. パッケージ署名の確認

RPM パッケージは、**GNU Privacy Guard** (**GPG**) を使用して署名することができます。これは、ダウンロードしたパッケージが信頼できること確認する際に役立ちます。**GPG** は安全な通信のためのツールです。**GPG** を使うと、ドキュメントの整合性を確認し、データの暗号化、暗号化解除ができます。

パッケージが破損していないことや改ざんされていないことを確認するには、**rpmkeys** コマンドに **-K** (または **--checksig**) オプションを付けて、**GPG** 署名をチェックします。

```
rpmkeys -K package.rpm
```

Yum パッケージマネージャーはインストールおよびアップグレード中に自動で**GPG** 署名の確認を行うことに注意してください。

GPG は、Red Hat のパッケージ確認用のキーセットと共にデフォルトでインストールされます。**RPM** で使用する追加のキーをインポートする方法については、「GPG キーのインポート」を参照してください。

A.3.2.1. GPG キーのインポート

Red Hat パッケージを検証するには、Red Hat **GPG** キーをインストールする必要があります。基本的なキーのセットはデフォルトでインストールされます。インストールされたキーを一覧表示するには、シェルプロンプトで以下のコマンドを実行します。

```
~]$ rpm -qa gpg-pubkey*
```

特定のキーに関する詳細を表示するには、以下の例のように、**rpm -qi** コマンドの後に先のコマンドの出力を続けて使用します。

```
~]$ rpm -qi gpg-pubkey-fd431d51-4ae0493b
```

RPM で使用する新しいキーをインストールするには、**rpmkeys** コマンドに **--import** オプションを付けて使用します。**RPM** GPG キーのデフォルトの保存場所は、**/etc/pki/rpm-gpg/** ディレクトリーです。新しいキーをインポートするには、**root** ユーザーとして以下のようなコマンドを実行します。

```
~]# rpmkeys --import /etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
```

Red Hat パッケージ署名のプラクティスについてのさらなる情報は、Red Hat カスタマーポータルの Red Hat GPG キー の記事を参照してください。

A.4. 実用的かつ一般的な RPM の使用例

RPM は、システムの管理や問題の診断と修正のための便利なツールです。よく使用されるオプションの概要については、以下の例を参照してください。

- 使用中のシステム全体を検証して、欠落しているファイルを判断するには、**root** で以下のコマンドを発行します。

  ```
  rpm -Va
  ```

  欠落しているまたは破損しているファイルがある場合は、関連パッケージの再インストールを検討してください。

- ファイルを所有しているパッケージを見つけるには、以下を入力します。

  ```
  rpm -qf file
  ```

- 特定のファイルを所有するパッケージを顕正するには、**root** で以下を入力します。

```
rpm -Vf file
```

- ファイルが属するパッケージの一部であるドキュメンテーションファイルの場所を見つけるには、以下を入力します。

```
rpm -qdf file
```

- (インストールされていない) パッケージファイルについての情報を見つけるには、以下のコマンドを使用します。

```
rpm -qip package.rpm
```

- パッケージに含まれるファイルを一覧表示するには、以下を使用します。

```
rpm -qlp package.rpm
```

他のオプションについては、rpm(8) man ページを参照してください。

A.5. その他のリソース

RPM は、パッケージのクエリ、インストール、アップグレード、削除を実行するための多くのオプションと方法を備えた非常に複雑なユーティリティーです。**RPM** のさらなる詳細については、以下のリソースを参照してください。

インストールされているドキュメント

- **rpm --help** — このコマンドを実行すると**RPM** のパラメータのクイックリファレンスが表示されます。
- rpm(8) — **RPM** man ページは利用可能なすべての**RPM** パラメーターについて概説します。

オンラインのドキュメント

- Red Hat Enterprise Linux 7 Security Guide — Red Hat Enterprise Linux 7 の 『Security Guide』 では、**Yum** パッケージマネジャーを使ってシステムを最新に保ち、ダウンロードされたパッケージを検証してインストールする方法を説明しています。
- **RPM** の Web サイト — http://www.rpm.org/
- **RPM** メーリングリスト — http://lists.rpm.org/mailman/listinfo/rpm-list

関連項目

- 5章*Yum* では、**Yum** パッケージマネージャーを使って、コマンドラインでパッケージを検索、インストール、更新、アンインストールする方法について説明しています。

付録B X Window System

Red Hat Enterprise Linux の中核はカーネルですが、多くのユーザーにとってオペレーティングシステムの顔となるのは、別名 *X* と呼ばれる *X Window System* によって提供されるグラフィカル環境です。

1984 年 6 月の X Window System のリリース以前に存在していた環境を含め、UNIX 業界では他のウィンドウ環境が存在していました。それでもなお、X は長年に渡って Red Hat Enterprise Linux など大半の UNIX 系オペレーティングシステムのデフォルトのグラフィカル環境であり続けています。

Red Hat Enterprise Linux のグラフィカル環境は、X Window System と関連技術の開発と戦略を管理するために創設されたオープンソース組織である *X.Org Foundation* により提供されています。X.Org は、世界中の数百人の開発者が関与している大規模で急速に発展しているプロジェクトです。これは、各種のハードウェアデバイスとアーキテクチャーを幅広くサポートし、数え切れないほどのオペレーティングシステムとプラットフォーム上で機能します。

X Window System は、クライアントサーバーアーキテクチャーを使用します。この主な目的は、広範囲なコンピューターとグラフィックマシンで稼働するネットワーク透過的なウィンドウシステムを提供することです。*X サーバー* (**Xorg** バイナリ) は、ネットワークまたはローカルのループバックインターフェースを介した *X クライアント* アプリケーションからの接続を待機します。X サーバーは、ビデオカード、モニター、キーボード、マウスなどのハードウェアと通信します。X クライアントアプリケーションはユーザースペースに存在し、ユーザー用に *グラフィカルユーザーインターフェース* (*GUI*) を作成してユーザーの要求を X サーバーに渡します。

B.1. X サーバー

Red Hat Enterprise Linux 7 は、ビデオドライバー、EXA、特に以前のリリースに加えてプラットフォームサポートの拡張機能を含む X サーバーバージョンを使用します。さらに、このリリースには X サーバー向けの自動設定機能だけでなく、大半のマウスとキーボードなどカーネルが把握している全入力デバイスに対応する汎用入力ドライバーである、**evdev** も含まれています。

X11R7.1 は、X Window System のモジュール化を特別に活用するための最初のリリースでした。このリリースは、X を論理的に明確なモジュールに分割したため、オープンソースの開発者はこれまでより簡単にシステムにコードを書き込むことができます。

現行のリリースでは、すべてのライブラリ、ヘッダー、バイナリは **/usr/** ディレクトリー下にあります。X クライアントおよびサーバーアプリケーション用の設定ファイルは、**/etc/X11/** ディレクトリーにあります。これには、X サーバーや X ディスプレイマネージャーの他、多くのベースコンポーネント用の設定ファイルが含まれます。

新しい Fontconfig ベースのフォントアーキテクチャーの設定ファイルは、今までどおり **/etc/fonts/fonts.conf** です。フォントの設定、追加方法の詳細については、「フォント」 を参照してください。

X サーバーは、幅広いハードウェアで高度なタスクを実行するため、それが機能するハードウェアの詳しい情報が必要となります。X サーバーは、それが実行する大半のハードウェアを自動的に検出し、それに応じて自己設定することができます。別の方法として、設定ファイルでハードウェアを手動で指定することも可能です。

Red Hat Enterprise Linux システムのインストーラーである Anaconda は、X パッケージがインストール用に選択されていれば、自動的に X のインストールと設定を行います。X サーバーが管理しているモニター、ビデオカード、または他のデバイスに変更が加えられると、ほとんどの場合 X はそれらの変更を自動検出して、再設定します。稀に手動で X を再設定する必要がある場合もあります。

B.2. デスクトップ環境とウィンドウマネージャー

X サーバーが稼働していれば、X クライアントアプリケーションはそれに接続し、ユーザー用に GUI を作成することができます。Red Hat Enterprise Linux には、*metacity* のようなスタンドアロンのウィンドウマネジャーから、大半の Red Hat Enterprise Linux ユーザーが精通している、高度に開発されたインタラクティブなデスクトップ環境である *GNOME* や *KDE* まで、幅広い GUI が利用可能です。

より総合的な GUI である後者のデスクトップ環境を作成するには、X クライアントアプリケーションの 2 つのメインクラスである *ウィンドウマネージャー* と *デスクトップ環境* を X サーバーに接続する必要があります。

B.2.1. デスクトップ環境

デスクトップ環境は、様々な X クライアントと統合して、一般的なグラフィカルユーザー環境と開発プラットフォームを作成します。

デスクトップ環境に備わっている高度な機能により、X クライアントと他の実行中のプロセスとの通信が可能になります。また一方で、ドラッグアンドドロップの操作など高度なタスクを実行するために、書き込まれたすべてのアプリケーションがデスクトップ環境で機能できるようにします。

Red Hat Enterprise Linux では 2 つのデスクトップ環境を採用しています。

- **GNOME** — GTK+ 3 グラフィカルツールキットをベースにした Red Hat Enterprise Linux 向けのデフォルトのデスクトップ環境です。
- **KDE** — Qt 4 グラフィカルツールキットを基にしたもう 1 つのデスクトップ環境です。

GNOME と KDE のどちらにも、ワードプロセッサー、スプレッドシート、Web ブラウザーなどの高度な生産性を高めるアプリケーションが備わっており、GUI のルックアンドフィールをカスタマイズするツールも提供しています。さらに、GTK+ 3 と Qt ライブラリの両方が存在している場合は、KDE アプリケーションは GNOME で実行でき、その逆も実行できます。

B.2.2. ウィンドウマネージャー

ウィンドウマネージャー は、デスクトップ環境の一部、または場合によってはスタンドアロンである X クライアントプログラムです。この主要な目的は、グラフィカルウィンドウを配置、サイズ変更、移動する方法を制御することです。また、ウィンドウマネージャーは、タイトルバー、ウィンドウフォーカスの動作、ユーザー指定のキーおよびマウスボタンの組み合わせの制御も行います。

The Red Hat Enterprise Linux リポジトリーは、いくつかの異なるウィンドウマネジャーを提供します。

`gnome-shell`

GNOME Shell は、GNOME のデフォルトのウィンドウマネジャーです。GNOME のユーザーエクスペリエンスを提供し、GNOME デスクトップがインストールされると、自動的にプルされます。

`metacity`

Metacity ウィンドウマネージャーは、シンプルかつ効率的なウィンドウマネージャーです。これは、完全なデスクトップ環境が不要な際に使用できます。

`kwin`

KWin のウィンドウマネージャーは、KDE 用のデフォルトウィンドウマネージャーです。これは、カスタムのテーマをサポートする効率的なウィンドウマネージャーです。このウィンドウマネージャーは、KDE デスクトップがインストールされた場合に依存関係として自動的にプルされます。

`mwm`

Motif Window Manager (**mwm**) は、ベーシックなスタンドアロンのウィンドウマネージャーです。スタンドアロンとして設計されているため、GNOME や KDE との併用は推奨されません。このウィンドウマネージャーを実行するには、*openmotif* パッケージをインストールする必要があります。

B.3. X サーバーの設定ファイル

X サーバーは、単一のバイナリ実行ファイル **/usr/bin/Xorg** であり、このファイルをポイントするシンボリックリンク **X** も提供されています。関連する設定ファイルは、**/etc/X11/** と **/usr/share/X11/** ディレクトリー内に格納されています。

X Window System は 2 種類の設定スキームに対応します。 **xorg.conf.d** ディレクトリーの設定ファイルには、ベンダーおよびディストリビューションより事前設定された設定が含まれています。これらのファイルは手動では編集しないことが推奨されます。一方で、**xorg.conf** ファイル内の設定はすべて手動で実行されますが、ほとんどの状況では必要ではありません。

注記

ディスプレイと周辺機器に必要なすべてのパラメーターは、インストール時に自動検出され、設定されます。X サーバー用の設定ファイルである **/etc/X11/xorg.conf** は、これまでのリリースでは必要でしたが、X Window System の現行のリリースには含まれていません。ただし、新しいハードウェアを設定するために手動でファイルを作成する場合や複数のビデオカードを備えた環境を設定する場合、デバッグする場合に引き続き有用です。

/usr/lib/xorg/modules/ (または **/usr/lib64/xorg/modules/**) ディレクトリーには、ランタイム時に動的に読み込み可能な X サーバーモジュールが含まれています。デフォルトでは、**/usr/lib/xorg/modules/** にある一部のモジュールのみ、X サーバーにより自動的に読み込まれます。

Red Hat Enterprise Linux 7 がインストールされる際に、X の設定ファイルは、インストールプロセス中に HAL (Hardware Abstraction Layer) 設定バックエンドにより収集されたシステムハードウェアに関する情報を使用して作成されます。X サーバーは起動すると常に、HAL に入力デバイスの一覧を要求して、それぞれ各ドライバーを使って追加します。新しい入力デバイスがプラグインされる場合や既存の入力デバイスが削除される場合、HAL は常にその変更を X サーバーに伝えます。この通知システムにより、**xorg.conf** ファイルで設定された **mouse**、**kbd** または **vmmouse** のドライバーを使用するデバイスは、デフォルトで X サーバーが無視します。詳細については、「**ServerFlags** セクション」 を参照してください。追加の設定は **/etc/X11/xorg.conf.d/** ディレクトリーにあり、これは HAL が取得した設定すべてを上書き、拡張することが可能です。

B.3.1. 設定の構造

X 設定ファイルの形式は、多くの様々なセクションで構成されており、システムハードウェアが持つ特有の側面に対応します。各セクションは **Section "*section-name*"** の行で始まります。"*section-name*" はセクションのタイトルで、**EndSection** の行で終わります。各セクションには、オプションの名前と 1 つ以上のオプション値を含む行があります。一部は、二重引用符 (") で囲まれている場合があります。

/etc/X11/xorg.conf ファイル内の一部のオプションは、ブール値スイッチを使用して機能をオンまたはオフにします。使用可能な値は以下のとおりです。

- **1**、**on**、**true** または **yes** — オプションをオンにします。
- **0**、**off**、**false** または **no** — オプションをオフにします。

以下は、キーボード用の標準的な設定ファイルです。ハッシュ記号 (#) で始まる行は、X サーバーからは読み取り不可能ですが、ヒューマンリーダブルなコメントとして使用されます。

```
# This file is autogenerated by system-setup-keyboard. Any
# modifications will be lost.

Section "InputClass"
  Identifier  "system-setup-keyboard"
  MatchIsKeyboard "on"
  Option    "XkbModel"  "pc105"
  Option    "XkbLayout" "cz,us"
# Option    "XkbVariant"  "(null)"
  Option    "XkbOptions"
"terminate:ctrl_alt_bksp,grp:shifts_toggle,grp_led:scroll"
EndSection
```

B.3.2. xorg.conf.d ディレクトリー

X サーバーは 2 つの設定ディレクトリーをサポートします。**/usr/share/X11/xorg.conf.d/** は、ベンダーやサードパーティーのパッケージとは別個の設定ファイルを提供します。このディレクトリー内のファイルへの変更は、**/etc/X11/xorg.conf** ファイル内で指定された設定により上書きできます。**/etc/X11/xorg.conf.d/** ディレクトリーは、ユーザー固有の設定を格納します。

設定ディレクトリーにある **.conf** の接尾辞が付いたファイルは、起動時に X サーバーが解析し、従来の **xorg.conf** 設定ファイルの一部のように扱われます。これらのファイルには、1 つ以上のセクションが含まれます。セクションのオプションの詳細や設定ファイルの一般的なレイアウトについては、「xorg.conf ファイル」 または **xorg.conf(5)** の man ページを参照してください。X サーバーは基本的に、設定ファイルの集まりを、最後の **xorg.conf** からのエントリーのある 1 つの大きなファイルとして扱います。ユーザーは、カスタム設定を **/etc/xorg.conf** に格納し、ディストリビューションが提供する設定スニペット用にディレクトリーを残しておくことが推奨されます。

B.3.3. xorg.conf ファイル

これまでの X Window System のリリースでは、**/etc/X11/xorg.conf** ファイルは X の初期設定を格納するために使用されていました。X サーバーにより管理されているモニター、ビデオカードや他のデバイスが原因で変更が生じた場合、ファイルは手動で編集する必要がありました。Red Hat Enterprise Linux では、**/etc/X11/xorg.conf** ファイルを手動で作成、編集する必要はほとんどありません。ただし、通常とは異なるハードウェア設定のトラブルシューティングや調整を行う時のために、各種セクションと利用可能なオプションのパラメーターについて理解しておくことは有用です。

以下では、重要なセクションが標準的な **/etc/X11/xorg.conf** ファイルに表示される順で説明されています。X サーバーの設定ファイルについての詳しい情報は、**xorg.conf(5)** の man ページに記載されています。下記で説明されるほとんどの設定オプションは、一般的な設定状況では必要ないため、本セクションの大部分は上級ユーザー向けになります。

B.3.3.1. InputClass セクション

InputClass は、ホットプラグデバイスをはじめ、単一のデバイスではなくデバイスのクラスに適用する新しいタイプの設定セクションです。**InputClass** セクションの範囲は、指定されるマッチにより制限されます。入力デバイスに適用するためには、すべてのマッチは以下の例のようにデバイスに適用する必要があります。

```
Section "InputClass"
   Identifier      "touchpad catchall"
   MatchIsTouchpad "on"
```

```
    Driver          "synaptics"
EndSection
```

このスニペットが **xorg.conf** ファイルか **xorg.conf.d** ディレクトリーにある場合、システムにあるすべての touchpad は **synaptics** ドライバーに割り当てられます。

注記

xorg.conf.d ディレクトリー内の設定ファイルは英数字順で並び替えられているため、上記の例の **Driver** 設定は、以前に設定されているドライバーオプションを上書きします。汎用性が高いクラスほど、最初の方に表示されます。

match のオプションは、セクションが適用可能なデバイスを指定します。デバイスがマッチするには、match の全オプションが一致する必要があります。以下は、**InputClass** セクションでよく使用されるオプションです。

- **MatchIsPointer**、**MatchIsKeyboard**、**MatchIsTouchpad**、**MatchIsTouchscreen**、**MatchIsJoystick** — デバイスの種類を指定するブール値のオプションです。
- **MatchProduct "*product_name*"** — このオプションは、*product_name* サブ文字列がデバイスの製品名に存在する場合にマッチします。
- **MatchVendor "*vendor_name*"** — このオプションは、*vendor_name* サブ文字列がデバイスのベンダー名に存在する場合にマッチします。
- **MatchDevicePath "*/path/to/device*"** — このオプションは、デバイスパスが **/dev/input/event*** といった "*/path/to/device*" テンプレートに一致する場合に、デバイスにマッチします。詳細は、**fnmatch(3)** man ページを参照してください。
- **MatchTag "*tag_pattern*"** — このオプションは、HAL 設定バックエンドにより割り当てられた 1 つ以上のタグが *tag_pattern* パターンにマッチする場合にマッチします。

設定ファイルに、**InputClass** セクションが複数ある場合があります。これらのセクションはオプションで、自動的に追加される時に入力デバイスのクラスを設定するために使用されます。入力デバイスは、複数の **InputClass** セクションにマッチすることができます。こうしたセクションを編成する場合、重複が発生すると、各入力クラスは以前のクラスの設定を上書きできるため、汎用的なマッチを特定のマッチより優先させることが推奨されます。

B.3.3.2. InputDevice セクション

各 **InputDevice** セクションは、X サーバーに対し 1 つの入力デバイスを設定します。これまでは、通常システムにはキーボード用に 1 つ以上の **InputDevice** セクションがあり、大半のマウス設定は自動的に検出されていました。

Red Hat Enterprise Linux 7 では、大半の設定には **InputDevice** を設定する必要はありません。*xorg-x11-drv-** 入力ドライバーパッケージは HAL による自動設定を提供します。キーボードとマウス用のデフォルトドライバーは **evdev** です。

以下の例は、キーボード用の標準的な **InputDevice** セクションです。

```
Section "InputDevice"
  Identifier "Keyboard0"
  Driver "kbd"
  Option "XkbModel" "pc105"
```

```
  Option "XkbLayout" "us"
EndSection
```

以下は、**InputDevice** セクションでよく使用されるエントリーです。

- **Identifier** — この **InputDevice** セクションに一意の名前を指定します。このエントリーは必須です。
- **Driver** — X がデバイス用に読み込む必要があるデバイスドライバーの名前を指定します。**AutoAddDevices** オプションが有効な場合 (デフォルト設定) は、**Driver "mouse"** または **Driver "kbd"** があるすべての入力デバイスセクションは無視されます。これが必要な理由は、レガシーなマウスおよびキーボードのドライバーと新しい **evdev** 汎用ドライバーとの間に競合があるためです。その代わりに、サーバーは全入力デバイスにバックエンドからの情報を使用します。**xorg.conf** にあるすべてのカスタム入力デバイス設定は、バックエンドへ移動することをお勧めします。ほとんどの場合、バックエンドは HAL となり、設定場所は **/etc/X11/xorg.conf.d** ディレクトリーになります。
- **Option** — デバイスに関する必要なオプションを指定します。

  マウスを指定して、デバイス用のすべての自動検出値を上書きすることもできます。以下のオプションは、通常 **xorg.conf** ファイルにマウスを追加する場合に含まれます。

  - **Protocol** — **IMPS/2** など、マウスにより使用されるプロトコルを指定します。
  - **Device** — 物理デバイスの場所を指定します。
  - **Emulate3Buttons** — 両方のマウスボタンを同時に押した時に、2 ボタンマウスが 3 ボタンマウスのように機能するかどうかを指定します。

  このセクションに関する有効なオプションの全一覧については、**xorg.conf(5)** の man ページを参照して下さい。

B.3.3.3. ServerFlags セクション

オプションの **ServerFlags** セクションには、X サーバーのその他のグローバル設定が含まれています。このセクションの全設定は、**ServerLayout** セクションにあるオプションにより上書きできます (詳細は 「**ServerLayout** セクション」 を参照)。

ServerFlags セクション内の各エントリーは、1 行で表示されます。また、**Option** という単語で始まり、二重引用符 (") で囲まれたオプションがその後に続きます。

以下は、**ServerFlags** セクションのサンプルです。

```
Section "ServerFlags"
  Option "DontZap" "true"
EndSection
```

最も役立つオプションの一部を以下に表示します。

- **"DontZap" "*boolean*"** — *<boolean>* の値が **true** に設定されている場合、X サーバーを直ちに終了させる **Ctrl+Alt+Backspace** キーの組み合わせは使用できなくなります。

注記

このオプションが有効であっても、X Keyboard Extension (XKB) マップでキーの組み合わせを使用前に設定する必要があります。マップにキーの組み合わせを追加する方法の 1 つとして、以下のコマンドを実行します。

```
setxkbmap -option "terminate:ctrl_alt_bksp"
```

- **"DontZoom" "*boolean*"** — *<boolean>* の値が **true** に設定されている場合、この設定では **Ctrl+Alt+Keypad-Plus** と **Ctrl+Alt+Keypad-Minus** キーの組み合わせを使用して、設定済みのビデオ解像度が変更しないようします。

- **"AutoAddDevices" "*boolean*"** — *<boolean>* の値が **false** に設定されている場合、サーバーは入力デバイスにホットプラグせずに、代わりに **xorg.conf** ファイルで設定済みのデバイスにのみ依存します。入力デバイスの詳細については、「**InputDevice** セクション」 を参照してください。このオプションはデフォルトで有効となっており、HAL (hardware abstraction layer) はデバイス検出のバックエンドとして使用されます。

B.3.3.4. ServerLayout セクション

ServerLayout セクションは、X サーバーにより制御されている出入力デバイスをバインドします。このセクションは、最低限 1 つの入力デバイスと 1 つの出力デバイスを指定する必要があります。デフォルトでは、モニター (出力デバイス) とキーボード (入力デバイス) が指定されています。

以下は、標準的な **ServerLayout** セクションの例です。

```
Section "ServerLayout"
  Identifier "Default Layout"
  Screen 0 "Screen0" 0 0
  InputDevice "Mouse0" "CorePointer"
  InputDevice "Keyboard0" "CoreKeyboard"
EndSection
```

以下は、**ServerLayout** セクションでよく使用されるエントリーです。

- **Identifier** — **ServerLayout** セクションに一意の名前を指定します。

- **Screen** — X サーバーと併用される **Screen** セクションの名前を指定します。**Screen** オプションは複数存在する場合があります。

  以下は、標準的な **Screen** エントリーの例です。

  ```
  Screen 0 "Screen0" 0 0
  ```

  この例の最初の番号である **Screen** エントリー (**0**) は、最初のモニターコネクターまたはビデオカードの ヘッド が、識別子 **"Screen0"** がある **Screen** セクションで指定されている設定を使用していることを示しています。

  識別子 **"Screen0"** がある **Screen** セクションの例は、「**Screen** セクション」 に記載されています。

  ビデオカードにヘッドが複数ある場合、別の番号と別の **Screen** セクション識別子を持つ他の **Screen** エントリーが必要です。

"Screen0" の右側の数字は、画面の左上隅に X と Y の絶対座標を渡します (デフォルトは**0 0**)。

- **InputDevice** — X サーバーにより使用される **InputDevice** セクションの名前を指定します。

  InputDevice エントリーは 2 つ以上あることが推奨されます。1 つはデフォルトのマウス用、もう 1 つはデフォルトのキーボード用です。**CorePointer** と **CoreKeyboard** のオプションは、これらがプライマリのマウスとキーボードであることを示しています。**AutoAddDevices** オプションが有効な場合、このエントリーは **ServerLayout** セクションで指定される必要はありません。**AutoAddDevices** オプションが無効な場合は、マウス、キーボードともデフォルト値で自動検出されます。

- **Option "*option-name*"** — このセクションに追加のパラメーターを指定するオプションのエントリーです。ここに一覧表示されている全オプションは、**ServerFlags** セクションに表示されているものを上書きします。

  <option-name> を **xorg.conf(5)** の man ページにあるこのセクション用に表示されている有効なオプションで置き換えます。

/etc/X11/xorg.conf ファイルには、複数の **ServerLayout** セクションを入れることができます。ただしデフォルトでは、サーバーはそれが見つける 1 番目のセクションのみを読み取ります。別の **ServerLayout** セクションがある場合は、**Xorg -layout <layoutname>** コマンドのように X セッションの開始時にコマンドラインの引数として指定することが可能です。

B.3.3.5. Files セクション

Files セクションは、フォントパスなど X サーバーに不可欠なサービス用のパスを設定します。通常こうしたパスは自動検出されるため、このセクションはオプションになります。このセクションを使用して、自動検出された値を上書きできます。

以下は、標準的な **Files** セクションの例です。

```
Section "Files"
  RgbPath "/usr/share/X11/rgb.txt"
  FontPath "unix/:7100"
EndSection
```

以下は、**Files** セクションでよく使用されるエントリーです。

- **ModulePath** — X サーバーモジュールを格納する別のディレクトリーを指定するオプションのパラメーターです。

B.3.3.6. Monitor セクション

各 **Monitor** セクションでは、システムにより使用されるモニターのタイプを 1 つ設定します。現在、大半のモニターは自動検出されるため、このエントリはオプションになります。

以下は、モニターの標準的な **Monitor** セクションの例です。

```
Section "Monitor"
  Identifier "Monitor0"
  VendorName "Monitor Vendor"
  ModelName "DDC Probed Monitor - ViewSonic G773-2"
  DisplaySize 320 240
  HorizSync 30.0 - 70.0
  VertRefresh 50.0 - 180.0
EndSection
```

以下は、**Monitor** セクションでよく使用されるエントリーです。

- **Identifier** — この **Monitor** セクションに一意の名前を指定します。このエントリーは必須です。
- **VendorName** — モニターのベンダーを指定するオプションのパラメーターです。
- **ModelName** — モニターのモデル名を指定するオプションのパラメーターです。
- **DisplaySize** — モニターの画像領域の物理サイズをミリメートル単位で指定するオプションのパラメーターです。
- **HorizSync** — モニターと互換性のある水平同期周波数の範囲を kHz 単位で指定します。これらの値により、X サーバーはモニター用の組み込みまたは指定された **Modeline** エントリーの有効性を決定できます。
- **VertRefresh** — モニターに対応する垂直リフレッシュ周波数の範囲を kHz 単位で指定します。これらの値により、X サーバーはモニターに対する組み込みまたは指定された **Modeline** エントリーの有効性を決定することができます。
- **Modeline** — 特定の水平同期および垂直リフレッシュ解像度を持つモニターに対して別途ビデオモードを指定するオプションのパラメーターです。**Modeline** エントリーの詳しい説明については、**xorg.conf(5)** man ページを参照してください。
- **Option "*option-name*"** — セクションに別途のパラメーターを指定するオプションのエントリーです。*<option-name>* を **xorg.conf(5)** の man ページにあるこのセクション用に表示されている有効なオプションで置き換えます。

B.3.3.7. Device セクション

各 **Device** セクションは、システム上の 1 つのビデオカードを設定します。1 つの**Device** セクションが最低限度ですが、マシンにインストールされているそれぞれのビデオカードに対し追加のインスタンスが生成する場合があります。

以下は、ビデオカードの標準的な **Device** セクションの例です。

```
Section "Device"
  Identifier "Videocard0"
  Driver "mga"
  VendorName "Videocard vendor"
  BoardName "Matrox Millennium G200"
  VideoRam 8192
  Option "dpms"
EndSection
```

以下は、**Device** セクションでよく使用されるエントリーです。

- **Identifier** — この **Device** セクションに一意の名前を指定します。このエントリーは必須です。
- **Driver** — ビデオカードを使用するために X サーバーが読み込む必要があるドライバーを指定します。ドライバーの一覧は **/usr/share/hwdata/videodrivers** にあり、*hwdata* パッケージでインストールされます。
- **VendorName** — ビデオカードのベンダーを指定するオプションのパラメーターです。
- **BoardName** — ビデオカードの名前を指定するオプションのパラメーターです。
- **VideoRam** — ビデオカードで利用可能な RAM の容量をキロバイト単位で指定するオプションのパラメーターです。この設定は、X サーバーがビデオ RAM の容量を検出するためにプローブできないビデオカードにのみ必要です。

- **BusID** — ビデオカードのバスの場所を指定するエントリです。ビデオカードが 1 つのみのシステム上では、**BusID** エントリーはオプションであり、デフォルトの**/etc/X11/xorg.conf** ファイルに存在しない場合もあります。ただし、複数のビデオカードがあるシステムでは、**BusID** エントリーは必須です。

- **Screen** — **Device** セクションで設定するモニターコネクターまたはビデオカードのヘッドを指定するオプションのエントリーです。このオプションは、複数のヘッドを持つビデオカードに対してのみ有用です。

  複数のモニターが同じビデオカードの異なるヘッドに接続されている場合は、別々の **Device** セクションを終了して、それぞれのセクションに異なる **Screen** 値がなければなりません。

  Screen エントリーの値は、整数でなければなりません。ビデオカードの最初のヘッドには **0** の値があります。それぞれ追加のヘッドの値は、この値を 1 つ増分させたものです。

- **Option "*option-name*"** — セクションに別途のパラメーターを指定するオプションのエントリーです。*<option-name>* を **xorg.conf(5)** の man ページにあるこのセクション用に表示されている有効なオプションで置き換えます。

  さらに一般的なオプションの 1 つは **"dpms"** (VESA 基準である Display Power Management Signaling) です。これにより、モニターのエネルギースター準拠の設定をアクティベートします。

B.3.3.8. Screen セクション

各 **Screen** セクションは、**Device** セクションと **Monitor** セクションをそれぞれ参照することによって、1 つのビデオカード (またはビデオカードヘッド) を 1 つのモニターにバインドします。1 つの **Screen** セクションが最低限度ですが、マシンに存在する各ビデオカードとモニターの組み合わせに対し追加のインスタンスが生成される場合もあります。

以下は、標準的な **Screen** セクションの例です。

```
Section "Screen"
  Identifier "Screen0"
  Device "Videocard0"
  Monitor "Monitor0"
  DefaultDepth 16

  SubSection "Display"
    Depth 24
    Modes "1280x1024" "1280x960" "1152x864" "1024x768" "800x600"
"640x480"
  EndSubSection

  SubSection "Display"
    Depth 16
    Modes "1152x864" "1024x768" "800x600" "640x480"
  EndSubSection
EndSection
```

以下は、**Screen** セクションでよく使用されるエントリーです。

- **Identifier** — この **Screen** セクションに一意の名前を指定します。このエントリーは必須です。

- **Device** — **Device** セクションに一意の名前を指定します。このエントリーは必須です。

- **Monitor** — **Monitor** セクションに一意の名前を指定します。これは、特定の**Monitor** セクションが **xorg.conf** ファイルで定義されている場合にのみ必須です。通常、モニターは自動検出されます。

- **DefaultDepth** — デフォルトの色深度をビット単位で指定します。前の例では、**16** (何千もの色を提供) がデフォルトです。1 つの **DefaultDepth** エントリーのみが許可されますが、Xorg コマンドラインのオプション **-depth <n>** により上書きされる場合があります。**<n>** は指定する追加の深度です。
- **SubSection "Display"** — 特定の色深度で利用できる画面モードを指定します。**Screen** セクションは、複数の **Display** サブセクションを持つことができます。画面モードは自動検出されるためこれは完全にオプションです。

  通常このサブセクションは、自動検出されたモードを上書きします。
- **Option "*option-name*"** — セクションに別途のパラメーターを指定するオプションのエントリーです。*<option-name>* を **xorg.conf(5)** の man ページにあるこのセクション用に表示されている有効なオプションで置き換えます。

B.3.3.9. DRI セクション

オプションの **DRI** セクションは、*DRI (Direct Rendering Infrastructure: ダイレクトレンダリングインフラストラクチャー)* のパラメーターを指定します。DRI は、3D ソフトウェアアプリケーションが大半の最新ビデオハードウェアに組み込まれている 3D ハードウェアアクセラレーションの機能を活用できるようにするインターフェースです。加えて、ビデオカードドライバーによりサポートされている場合は、DRI はハードウェアアクセラレーションにより 2D 性能が向上します。

DRI の Group と Mode はデフォルト値に自動的に初期化されるため、このセクションはほとんど使用されません。別の Group や Mode が必要な場合は、このセクションを **xorg.conf** ファイルに追加すると、デフォルト値が上書きされます。

以下は、標準的な **DRI** セクションの例です。

```
Section "DRI"
  Group 0
  Mode 0666
EndSection
```

それぞれビデオカードは様々な方法で DRI を使用するため、最初に http://dri.freedesktop.org/wiki/ を参照してから本セクションに追加するようにして下さい。

B.4. フォント

Red Hat Enterprise Linux は、*Fontconfig* サブシステムを使用して、X Window System でフォントの管理と表示を行います。*Fontconfig* サブシステムは、フォント管理を簡素化して、アンチエイリアシングなど高度な表示機能を提供します。このシステムは、**Qt 3** もしくは **GTK+ 2** グラフィカルツールキットまたはその最新版を使用してプログラムされたアプリケーション用に自動的に使用されます。

Fontconfig フォントサブシステムにより、アプリケーションは直接システムのフォントにアクセスし、*X FreeType インターフェースライブラリ* (*Xft*) やアンチエイリアシングなど高度な機能を持つ Fontconfig フォントをレンダリングする他のレンダリングメカニズムを使用することができます。グラフィカルなアプリケーションは、Fontconfig と共に Xft ライブラリを使用して画面にテキストを描くことができます。

注記

Fontconfig は **/etc/fonts/fonts.conf** 設定ファイルを使用します。このファイルを手動で編集することはお勧めしません。

警告

ユーザーがリモート X アプリケーションの実行を要求するシステムでは、**fonts** グループがインストールされている必要があります。このためには、インストーラーでグループを指定するか、インストール後に **yum groupinstall fonts** コマンドを実行します。

B.4.1. Fontconfig へのフォントの追加

Fontconfig のサブシステムに新しいフォントを追加するプロセスは簡単です。

1. 個々のユーザー用にフォントを追加するには、ユーザーのホームディレクトリーの **.fonts/** ディレクトリーに新しいフォントをコピーします。

   システム全体にフォントを追加するには、新しいフォントを **/usr/share/fonts/** ディレクトリーにコピーします。ユーザーがインストールしたフォントとデフォルトのフォントを区別できるように、**local/** やそれに類似する新しいサブディレクトリーを作成することは好ましい方法です。

2. フォント情報のキャッシュを更新するには、root として **fc-cache** コマンドを実行します。

   ```
   fc-cache <path-to-font-directory>
   ```

   このコマンドでは、*<path-to-font-directory>* を新しいフォントを含むディレクトリー (**/usr/share/fonts/local/** または **/home/<user>/.fonts/**) で置き換えます。

注記

個々のユーザーがフォントをインタラクティブにインストールするには、**Nautilus** アドレスバーに **fonts:///** を入力して、そこに新しいフォントファイルをドラッグすることもできます。

B.5. ランレベル、ターゲット、および X

Red Hat Enterprise Linux 7 では、ランレベルの概念は *systemd targets* に代わっています。ターゲットに関する詳細は、6章*systemd によるサービス管理*を参照してください。

B.6. その他のリソース

X サーバーやそれに接続するクライアント、様々なデスクトップ環境とウィンドウマネージャーに関する詳しい情報は、多くのものが入手可能です。

B.6.1. インストールされているドキュメント

- **/usr/share/X11/doc/** — X Window System アーキテクチャーに関する詳しいドキュメント、さらには新しいユーザーとしての Xorg プロジェクトに関する追加情報の取得方法が記載されています。
- **/usr/share/doc/gdm-<version-number>/README** — ディスプレイマネージャーがユーザー認証を管理する方法が記載されています。
- **man xorg.conf** — **xorg.conf** 設定ファイル内の様々なセクションに関する意味と構文など、そのファイルに関する情報が含まれています。

- **man Xorg** — **Xorg** ディスプレイサーバーについて説明しています。

B.6.2. 役立つ Web サイト

- http://www.X.org/ — X.Org Foundation のホームページです。Red Hat Enterprise Linux に含まれている X Window System のメジャーリリースを作成して、必要なハードウェアの制御と GUI 環境の提供を行います。
- http://dri.sourceforge.net/ — DRI (Direct Rendering Infrastructure) プロジェクトのホームページです。DRI は、X のコアハードウェア 3D アクセラレーションコンポーネントです。
- http://www.gnome.org/ — GNOME プロジェクトのホームページです。
- http://www.kde.org/ — KDE デスクトップ環境のホームページです。

付録C 改訂履歴

改訂 0.12-0.3 **Fri Aug 21 2015** **Chester Cheng**

ソート番号を更新し、ドキュメントを再ビルドします。

改訂 0.12-0 **Tue 19 Aug 2014** **Stephen Wadeley**

Red Hat Enterprise Linux 7.0 GA リリースのシステム管理者ガイド

索引

シンボル

T

U

V

W

X