Red Hat Enterprise Linux 7

カーネル管理ガイド

カーネル管理のタスク例

Last Updated: 2018-06-04

Red Hat Enterprise Linux 7 カーネル管理ガイド

カーネル管理のタスク例

Marie Dolezelova
Red Hat Customer Content Services
mdolezel@redhat.com

Mark Flitter
Red Hat Customer Content Services

Douglas Silas
Red Hat Customer Content Services

Eliska Slobodova
Red Hat Customer Content Services

Jaromir Hradilek
Red Hat Customer Content Services

Maxim Svistunov
Red Hat Customer Content Services

Robert Krátký
Red Hat Customer Content Services

Stephen Wadeley
Red Hat Customer Content Services

Florian Nadge
Red Hat Customer Content Services

法律上の通知

概要

カーネル管理ガイドは、Red Hat Enterprise Linux 7 カーネルのメンテナンスタスクについて説明します。今回のリリースでは、kpatch の使用、カーネルモジュールの管理、カーネルの手動更新などの情報が含まれます。

目次

前書き

カーネル管理ガイド では、カーネルの活用方法を説明し、複数の実用的なタスクを紹介します。本書は、カーネルクラッシュダンプガイドから始まります。カーネルクラッシュダンプガイドでは、カーネルに障害が発生した場合に **vmcore** コレクションの設定やテストを行うプロセスについて順を追って説明し、また、カーネルモジュールの使用、カーネルの手動アップグレード、**sysfs** ユーティリティーとの対話方法などの情報も含まれます。

また、**カーネル管理ガイド** は、カーネル管理のユースケースの一部についても説明し、コマンドラインオプションの参考資料や、チューニング可能なカーネル (スイッチとも呼ばれる) 、カーネル機能の概要についても紹介します。

第1章 カーネルクラッシュダンプガイド

1.1. KDUMP について

1.1.1. kdump と kexec について

kdump とは、システムのメモリー内容を保存して後で分析できるようにカーネルのクラッシュをダンプする仕組みを指します。kdump は **kexec** に依存し、この kexec を使用して、別のカーネルのコンテキストから Linux カーネルを起動し、BIOS を迂回して通常は失われてしまう 1 番目のカーネルメモリーの内容を保持することができます。

システムクラッシュが発生すると kdump は kexec を使用して 2 番目のカーネルで起動します (**キャプチャーカーネル**)。この 2 番目のカーネルはシステムメモリーの予約部分に収納されていて 1 番目のカーネルからはアクセスできません。2 番目のカーネルは起動するとクラッシュしたカーネルメモリーの内容 (**クラッシュダンプ**) をキャプチャーして保存します。

重要

カーネルクラッシュダンプは、障害時に唯一利用可能な情報である可能性があるので、ビジネスに不可欠な環境ではこのデータの重要性を過小評価してはいけません。Red Hat は、システム管理者に対して、通常のカーネル更新サイクルで **kexec-tools** を定期的に更新、テストすることを推奨します。これは、新しいカーネル機能が実装された場合に特に重要です。

1.1.2. メモリーの要件

kdump でカーネルクラッシュのダンプをキャプチャーして分析のため保存するにはキャプチャーカーネル用にシステムメモリーの一部を永続的に予約しておく必要があります。予約するとその部分はメインカーネルでは使用できなくなります。

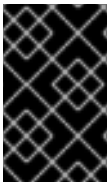
メモリーの要件は、特定のシステムパラメーターにより異なります。主な要因の 1 つとして、システムのハードウェアアーキテクチャーが挙げられます。次のコマンドをシェルプロンプトで入力してマシンの正確なアーキテクチャー名を特定し (**x86_64** など)、標準出力に表示させます。

```
uname -m
```

予約すべきメモリーサイズを左右する別の要因として搭載しているシステムメモリーの総量が影響します。たとえば、x86_64 アーキテクチャーでは予約メモリーは 4 KB のメモリーごとに、160 MB + 2 ビットになります。搭載されている物理メモリーの合計が 1 TB のシステムの場合には 224 MB (160 MB + 64 MB) ということになります。システムアーキテクチャーごとの kdump メモリー要件と物理メモリー量についての詳細は 「kdump メモリー要件」 を参照してください。

多くのシステムでは必要なメモリー量は kdump によって自動的に算出、予約が行われます。この動作はデフォルトで有効になっていますが、利用可能な合計メモリーサイズが一定以上搭載されているシステムに限られます。この自動割り当て動作に必要なメモリーサイズはシステムのアーキテクチャーによって異なります。「メモリー自動予約の最小しきい値」 に自動メモリー割り当てに必要な最小メモリーサイズの一覧をシステムアーキテクチャーごとに示します。

システムメモリーが自動割り当ての動作に必要な最小メモリーに満たない場合、または独自の予約メモリーサイズを必要とするような場合には予約メモリーを手動で設定することができます。コマンドラインでこの作業を行う場合は 「メモリー使用量の設定」 を参照してください。グラフィカルユーザーインターフェースでこの作業を行う場合は 「メモリー使用量の設定」 を参照してください。



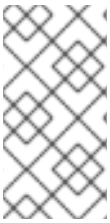
重要

kdump サービスを設定したら、自動メモリー割り当てであっても設定のテストを行うことを強く推奨します。設定のテスト方法については 「kdump 設定のテスト」 を参照してください。

1.2. KDUMP のインストールと設定

1.2.1. kdump のインストール

多くの場合、Red Hat Enterprise Linux 7 の新規インストールで **kdump** サービスはデフォルトでインストールされます。グラフィカルインターフェースまたはテキストインターフェースを使って対話形式でインストールする場合には、**Anaconda** インストーラーに kdump の設定画面があります。インストーラーの画面には **kdump** というタイトルが付けられ、メイン画面の **インストールの概要** からアクセスすることができます。設定については制限があり、ここで選択できるのは kdump を有効にするかどうかという点と予約するメモリーサイズだけです。kdump のメモリー要件については 「kdump メモリー要件」 を参照してください。インストーラーの kdump 設定画面については **Red Hat Enterprise Linux 7 インストールガイド** で説明しています。



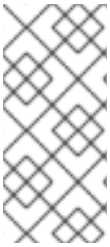
注記

Red Hat Enterprise Linux の以前のリリースでは **Firstboot** ユーティリティーで kdump の設定ができました。このユーティリティーは **インストールの終了後**、システムをはじめて再起動すると自動的に実行されていました。Red Hat Enterprise Linux 7.1 からは kdump の設定がインストーラー内に移動しています。

カスタムのキックスタートを使ったインストールなど一部のインストール方法では、デフォルトで kdump をインストールしない場合または有効にしない場合があります。このような場合に、kdump を追加でインストールするには **root** で次のコマンドをシェルプロンプトから実行します。

```
# yum install kexec-tools
```

お使いのシステムアーキテクチャー向けの **kexec-tools** パッケージが含まれるカスタムのリポジトリーか、アクティブなサブスクリプションがシステムにある場合に、上記のコマンドで kdump およびその他に必要なパッケージすべてが確実にインストールされます。



注記

システムに kdump がインストールされているかわからない場合は **rpm** を使うと確認できます。

```
$ rpm -q kexec-tools
```

この他、グラフィカルな設定ツールもあります。ただし上記のコマンドを使った場合に、デフォルトではグラフィカルな設定ツールはインストールされません。グラフィカルな設定ツールをインストールする場合は次のコマンドを **root** で実行します。グラフィカルな設定ツールの詳細は 「グラフィカルユーザーインターフェースでの kdump の設定」 を参照してください。

```
# yum install system-config-kdump
```

Yum を使用した Red Hat Enterprise Linux 7 の新規パッケージのインストール方法については『**Red Hat Enterprise Linux 7 システム管理者のガイド**』を参照してください。

重要

Red Hat Enterprise Linux 7.4 から、**kdump** は **Intel IOMMU** ドライバーをサポートしています。7.3 以前のバージョンのカーネルを実行する場合は、**Intel IOMMU** のサポートを無効にすることを推奨します。

1.2.2. コマンドライン上での kdump の設定

1.2.2.1. メモリー使用量の設定

kdump カーネル用に予約されるメモリーは必ずシステムの起動時にその予約が行われます。つまり、メモリーのサイズはシステムのブートローダー設定で指定されています。

kdump カーネル用に割り当てるメモリーを指定するには、**crashkernel=** オプションを必要な値に設定します。たとえば、128 MB のメモリーを割り当てるには、以下を使用します。

```
crashkernel=128M
```

AMD64 および Intel 64 システム、IBM Power SYstems サーバーでは **GRUB2** ブートローダーを使用し、IBM System z では **zipl** を使用して **crashkernel=** オプションを変更する方法は、「カーネルコマンドラインパラメーターの設定」 を参照してください。

crashkernel= オプションの指定方法は数種類あります。**auto** の値を使用すると 「kdump メモリー要件」 に記載したガイドラインに沿ってシステムのメモリー合計に基づいた予約メモリーが自動設定されます。メモリーサイズが大きいシステムで、**crashkernel=auto** オプションが指定されている場合には、 **オペレーティングシステムに設定された上限まで** 、<link to OS Limits> が算出されます。

この動作を変更するには、**auto** の値を特定のメモリー量に置き換えます。

crashkernel= オプションは、特にメモリー量が少ないシステムで有用です。たとえば、128 MB のメモリーを確保するには、以下を使用します。

```
crashkernel=128M
```

搭載しているメモリーの合計サイズに応じて予約メモリーサイズが可変するように設定することもできます。可変のメモリー予約を設定する場合の構文は **crashkernel=<range1>:<size1>、<range2>:<size2>** になります。例を示します。

```
crashkernel=512M-2G:64M,2G-:128M
```

上記の例の場合、システムメモリーの合計サイズが 512 MB 以上 2 GB 未満の場合は 64 MB のメモリー、システムメモリーの合計サイズが 2 GB 以上の場合は 128 MB のメモリーが kdump 用に予約されます。

システムによっては、特定の固定オフセットを指定して、メモリーの予約を行う必要があります。オフセットを設定すると予約メモリーはそこから開始されます。予約メモリーにオフセットを指定する場合は次の構文を使用します。

```
crashkernel=128M@16M
```

上記の例の場合、kdump は 128 MB のメモリー予約を 16 MB (物理アドレス 0x01000000) から開始することになります。オフセットパラメーターを 0 に設定する、または完全に省略すると kdump により自動的にオフセットが設定されます。前述の可変の予約メモリーを設定する場合にもこの構文を使用す

ることができます。オフセットは必ず最後に指定します (例: **crashkernel=512M-2G:64M,2G-:128M@16M**)。

1.2.2.2. kdump タイプの設定

カーネルクラッシュをキャプチャーする場合に、コアダンプは、ローカルファイルシステムにファイルとして格納するか、デバイスに直接書き込むか、**NFS** (Network File System) または **SSH** (Secure Shell) のプロトコルを使用してネットワークで送信することが可能です。現時点で設定できるのは、これらのオプションのうちの 1 つのみである点に注意してください。デフォルトのオプションは、**vmcore** ファイルをローカルファイルシステムの **/var/crash/** ディレクトリー内に格納する方法です。このオプションを変更するには、**root** としてテキストエディターで **/etc/kdump.conf** 設定ファイルを開き、以下のようにオプションを編集します。

コアダンプの保存先のローカルディレクトリーを変更する場合は **#path /var/crash** の行頭にあるハッシュ記号 (#) を取り除き、値を変更先のディレクトリーパスに置き換えます。

```
path /usr/local/cores
```

重要

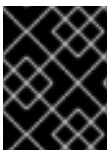
Red Hat Enterprise Linux 7 では kdump のダンプ出力先として **path** ディレクティブで指定されているディレクトリーが **kdump** systemd サービスの起動時に存在していなければなりません。起動時にこのディレクトリーが存在していないとサービスの起動は失敗します。この動作は Red Hat Enterprise Linux の以前のリリースと異なります。以前のリリースではサービスの起動時にこのディレクトリーが存在していないと自動的に作成されていました。

オプションで、ファイルを別のパーティションに書き込む場合は、**#ext4** で始まる行のハッシュ記号を取り除き、値を変更先のディレクトリーパスに置き換えます。値にはデバイス名 (**#ext4 /dev/vg/lv_kdump** 行)、ファイルシステムのラベル (**#ext4 LABEL=/boot** 行)、UUID (**#ext4 UUID=03138356-5e61-4ab3-b58e-27507ac41937** 行) のいずれかを使用できます。例を示します。

```
ext4 UUID=03138356-5e61-4ab3-b58e-27507ac41937
```

重要

ストレージデバイスの指定は **LABEL=** または **UUID=** を使用することを推奨します。**/dev/sda3** などディスクデバイス名は再起動を行うと同じ名前が使用されない恐れがあります。永続的なディスクデバイスの命名については『Red Hat Enterprise Linux 7 ストレージ管理ガイド』を参照してください。



重要

s390x ハードウェア上の DASD にダンプする場合には、続行する前に **/etc/dasd.conf** でダンプサービスが正しく指定されている必要があります。

ダンプをデバイスに直接書き込む場合は **#raw /dev/vg/lv_kdump** の行頭にあるハッシュ記号 (#) を取り除き、値をダンプ出力先のデバイス名に置き換えます。例を示します。

```
raw /dev/sdb1
```

NFS プロトコルを使ってリモートのマシンにダンプを保存する場合は **#nfs my.server.com:/export/tmp** の行頭にあるハッシュ記号 (#) を取り除き、値を有効なホスト名とディレクトリーパスに置き換えます。

```
nfs penguin.example.com:/export/cores
```

SSH プロトコルを使ってリモートのマシンにダンプを保存する場合は **#ssh user@my.server.com** の行頭にあるハッシュ記号 (#) を取り除き、値を有効なユーザー名とホスト名に置き換えます。設定に SSH キーも含める場合は **#sshkey /root/.ssh/kdump_id_rsa** 行の先頭にあるハッシュ記号 (#) を取り除き、値をダンプ出力先となるサーバー上で有効なキーの場所に変更します。

```
ssh john@penguin.example.com
sshkey /root/.ssh/mykey
```

SSH サーバーの設定方法およびキーベースの認証設定については『**Red Hat Enterprise Linux 7 システム管理者のガイド**』を参照してください。

サポートしているダンプ出力先と非サポートのダンプ出力先のタイプ別一覧は 表1.3「サポートしている kdump のダンプ出力先」を参照してください。

1.2.2.3. コアコレクターの設定

vmcore ダンプファイルのサイズを小さくするために、**kdump** では外部アプリケーション (**core collector**) を指定して、データの圧縮や必要に応じた関連性のないすべての情報の除外ができます。現在、完全サポートしているコアコレクターは **makedumpfile** のみになります。

コアコレクターを有効にするには、**root** として、テキストエディターで **/etc/kdump.conf** 設定ファイルを開いて、**#core_collector makedumpfile -l --message-level 1 -d 31** の行頭にあるハッシュ記号 (#) を取り除き、以下の説明通りにコマンドラインのオプションを編集します。

ダンプファイルの圧縮を有効にするには、**-c** パラメーターを追加します。例を示します。

```
core_collector makedumpfile -c
```

ダンプから特定のページを除外するには、**-d value** を追加します。**value** には 表1.4「サポートしているフィルターレベル」で説明しているように、省略するページの値の合計を入力します。ゼロと未使用ページを除外する場合は次のようになります。

```
core_collector makedumpfile -d 17 -c
```

使用できるオプションの一覧については **makedumpfile(8)** の man ページを参照してください。

1.2.2.4. デフォルト動作の設定

kdump が 「kdump タイプの設定」 で指定したダンプ出力先でのコアダンプの作成に失敗すると、デフォルトではルートファイルシステムがマウントされ、**kdump** はコアをローカルに保存しようとします。この動作を変更する場合は **root** として、テキストエディターで **/etc/kdump.conf** 設定ファイルを開きます。**#default shell** の行頭にあるハッシュ記号 (#) を取り除き、表1.5「サポートしているデフォルトの動作」 の説明にしたがって値を目的の動作に変更します。

以下に例を示します。

```
default reboot
```

1.2.2.5. サービスの有効化

起動時に **kdump** 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 enable kdump.service
```

multi-user.target のサービスが有効になります。同様に **systemctl stop kdump** を入力するとこのサービスが無効になります。現在のセッションでサービスを開始する場合は **root** として、次のコマンドを使用します。

```
systemctl start kdump.service
```

重要

Red Hat Enterprise Linux 7 では kdump の出力先として指定されているディレクトリーが **kdump** systemd サービスの起動時に存在していなければなりません。起動時にこのディレクトリーが存在していないとサービスの起動は失敗します。この動作は Red Hat Enterprise Linux の以前のリリースと異なります。以前のリリースではサービスの起動時にこのディレクトリーが存在していないと自動的に作成されていました。

systemd およびサービスの設定全般については『**Red Hat Enterprise Linux 7 システム管理者のガイド**』を参照してください。

1.2.3. グラフィカルユーザーインターフェースでの kdump の設定

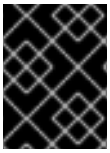
Kernel Dump Configuration ユーティリティーを起動するにはパネルから **Activities** → **Other** → **Kernel crash dumps** の順で選択するか、シェルプロンプトで **system-config-kdump** を入力します。図1.1「基本設定」 に示すウィンドウが表示されます。

このユーティリティーを使用すると **kdump** の設定のほか、起動時にサービスを有効または無効にすることもできます。設定が完了したら **適用** をクリックして変更を保存します。認証が済んでいる場合を除きスーパーユーザーのパスワード入力が求められます。また、設定の変更を適用するにはシステムの再起動が必要な旨を示すメッセージが表示されます。

重要

SELinux が Enforcing モードで実行中の IBM System z または PowerPC システムでは、カーネルダンプ設定ユーティリティーを起動する前に kdumpgui_run_bootloader のブール値を有効化する必要があります。このブール値により、system-config-kdump が bootloader_t SELinux ドメインのブートローダーで実行できるようになります。このブール値を永続的に有効化するには、root として以下のコマンドを実行します。

```
# setsebool -P kdumpgui_run_bootloader 1
```



重要

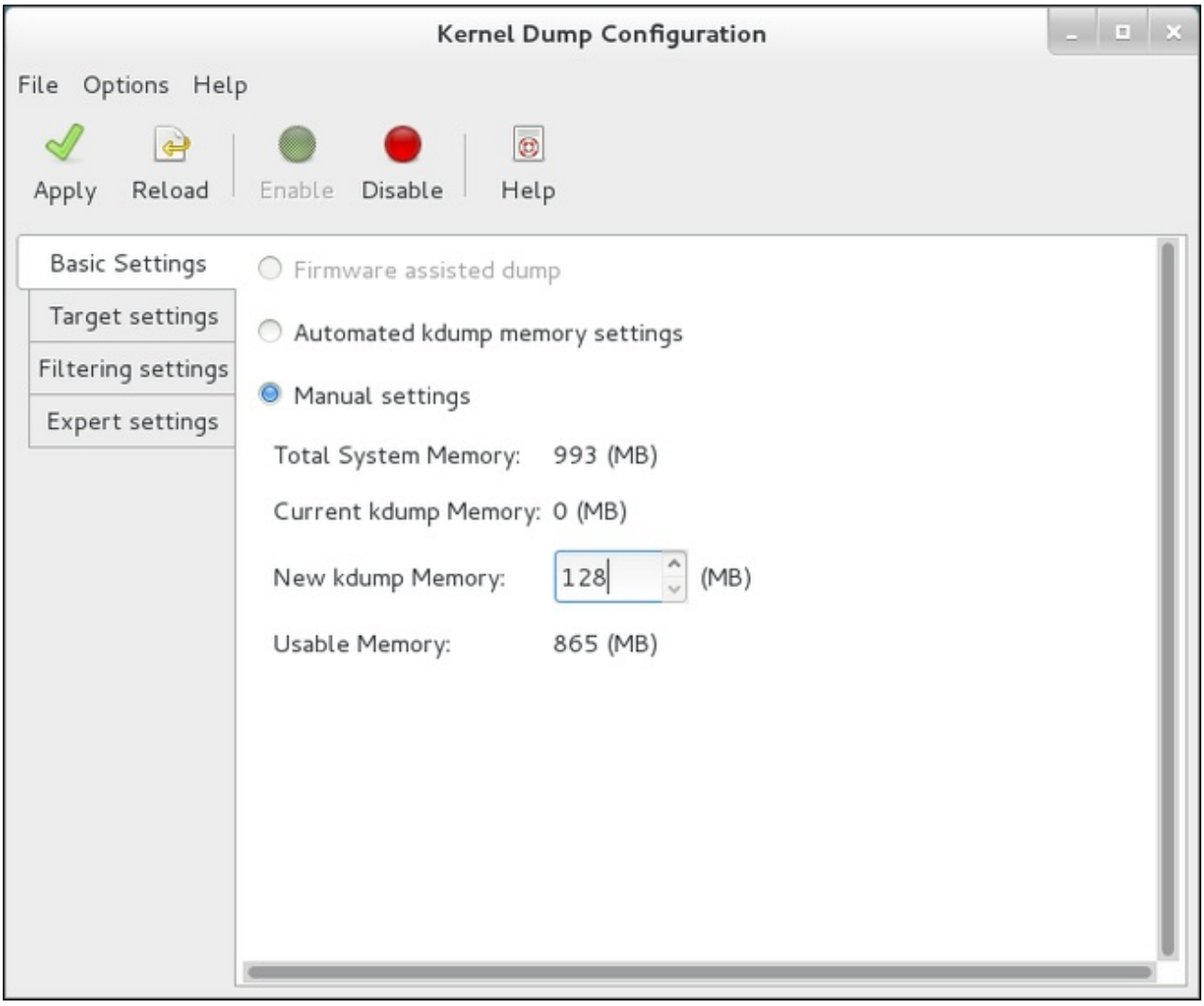
s390x ハードウェア上の DASD にダンプする場合には、続行する前に **/etc/dasd.conf** でダンプサービスが正しく指定されている必要があります。

1.2.3.1. メモリー使用量の設定

基本設定 タブでは **kdump** カーネル用に予約するメモリーサイズを設定できます。**手動セッティング** の

ラジオボタンを選択し、**新規の kdump メモリー** フィールドの横にある上矢印ボタンまたは下矢印ボタンをクリックして予約するメモリーサイズを増減させます。システムで使用できるメモリーの残量に応じて **使用可能なメモリー** フィールドが変化します。kdump のメモリー要件については 「メモリーの要件」 を参照してください。

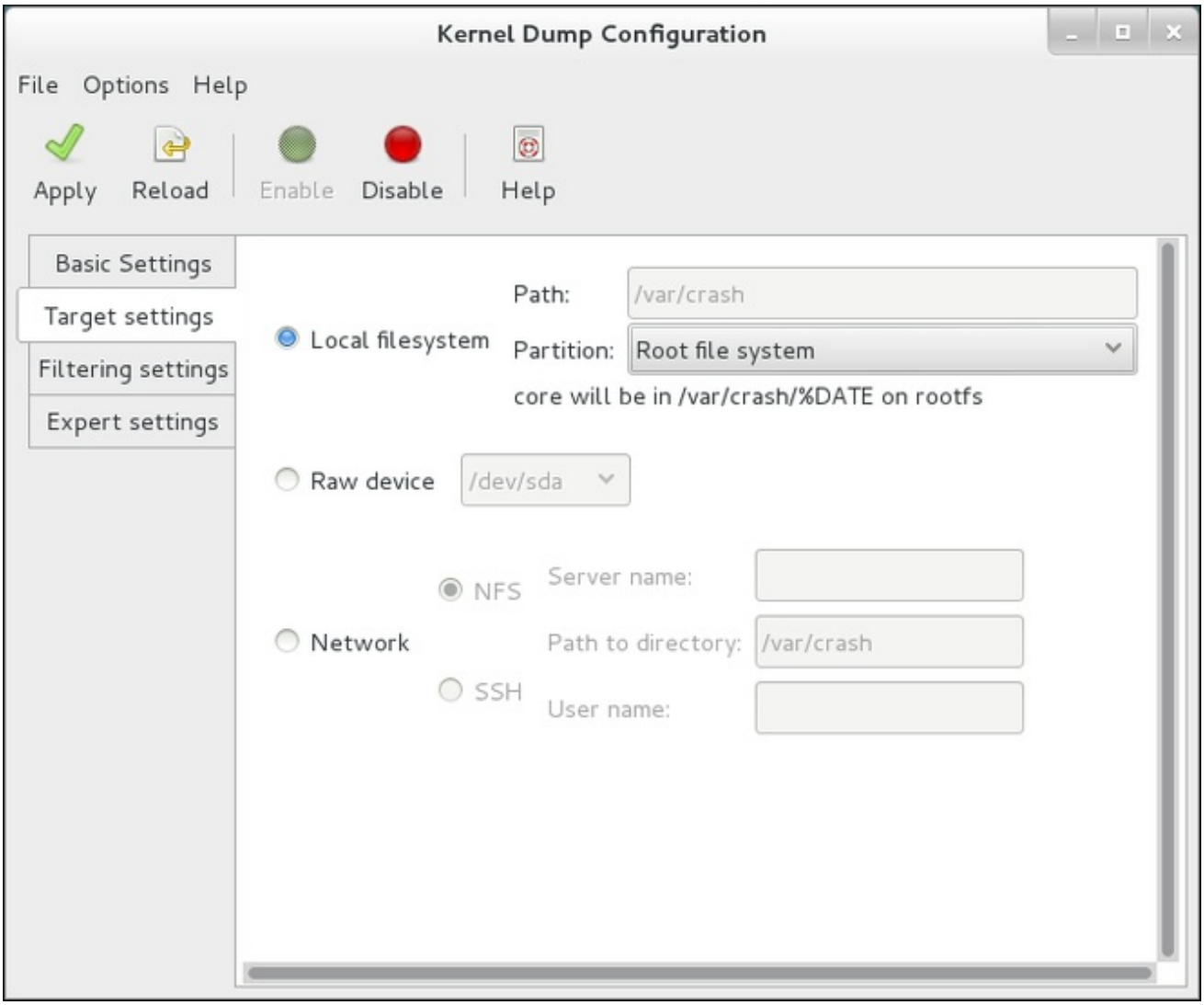
図1.1 基本設定



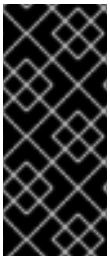
1.2.3.2. kdump タイプの設定

出力先 タブでは **vmcore** ダンプの出力先を指定することができます。ダンプはローカルのファイルシステムにファイルとして保存するか、デバイスに直接書き込むか、または **NFS** (Network File System) や **SSH** (Secure Shell) などのプロトコルを使ってネットワーク経由で送信することができます。

図1.2 出力先



ローカルのファイルシステムにダンプを保存する場合は **ローカルファイルシステム** のラジオボタンを選択します。必要に応じて **パーティション** のドロップダウンリストから別のパーティションを選択し、**パス** フィールドで出力先ディレクトリーを選択して設定をカスタマイズすることもできます。



重要

Red Hat Enterprise Linux 7 では kdump の出力先として指定されているディレクトリーが **kdump** systemd サービスの起動時に存在していなければなりません。起動時にこのディレクトリーが存在していないとサービスの起動は失敗します。この動作は Red Hat Enterprise Linux の以前のリリースと異なります。以前のリリースではサービスの起動時にこのディレクトリーが存在していないと自動的に作成されていました。

デバイスに直接ダンプを書き込む場合は **Raw デバイス** ラジオボタンを選択し、目的の出力先デバイスをその横にあるドロップダウンリストから選択します。

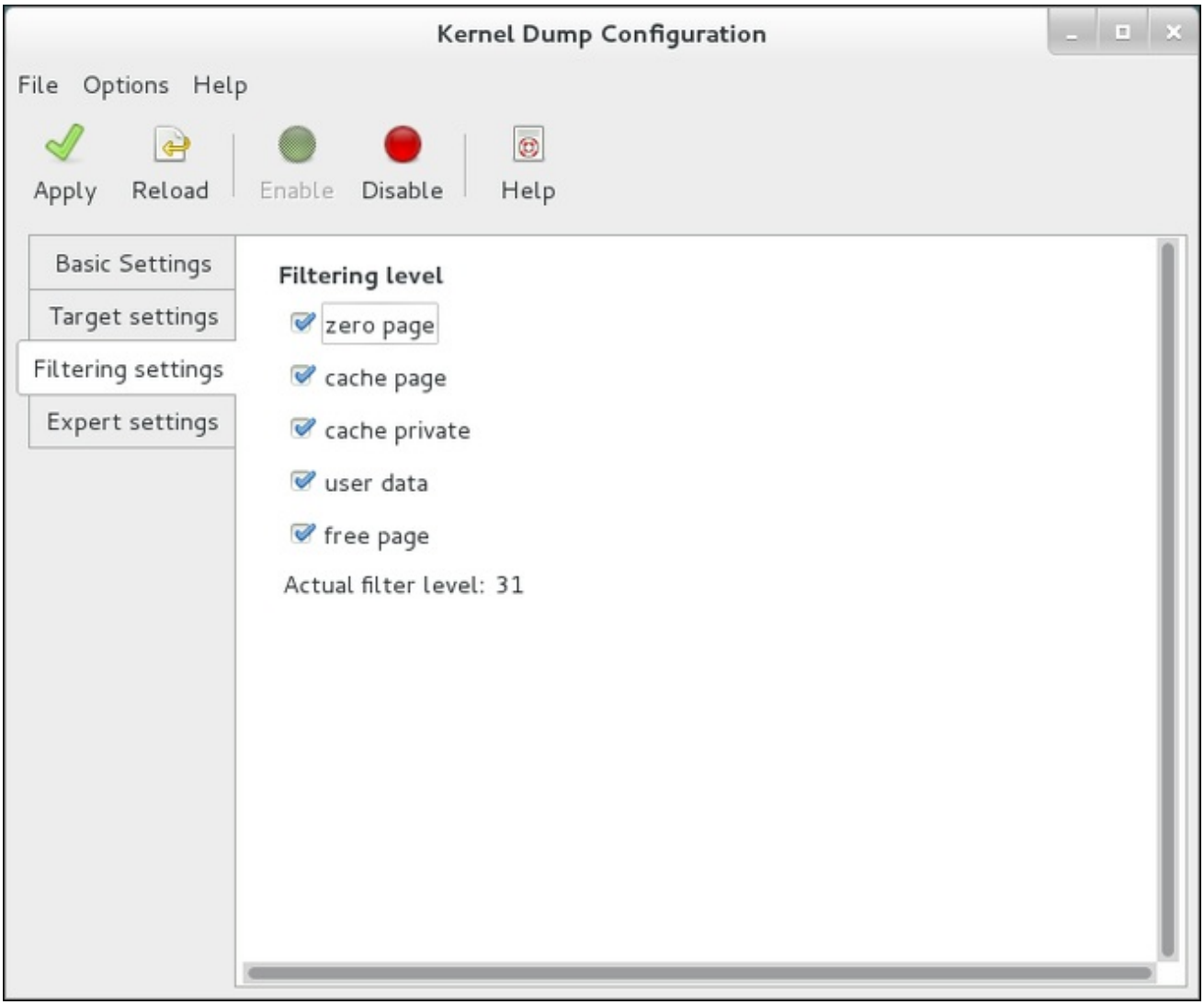
ネットワーク接続経由で、リモートのマシンにダンプを送信する場合は **ネットワーク** ラジオボタンを選択します。**NFS** プロトコルを使用する場合は **NFS** ラジオボタンを選択して **サーバー名** と **ディレクトリーへのパス** フィールドを入力します。**SSH** プロトコルを使用する場合は **SSH** ラジオボタンを選択して **サーバー名**、**ディレクトリーへのパス**、**ユーザー名** のフィールドにリモートサーバーのアドレス、出力先ディレクトリー、有効なユーザー名をそれぞれ入力します。

SSH サーバーの設定方法およびキーベースの認証設定については『**Red Hat Enterprise Linux 7 システム管理者のガイド**』を参照してください。現在サポートしている出力先の一覧については 表1.3「サポートしている kdump のダンプ出力先」 を参照してください。

1.2.3.3. コアコレクターの設定

Filtering Settings (フィルタリングセッティング) タブでは、**vmcore** ダンプのフィルターレベルを選択することができます。

図1.3 フィルタリング

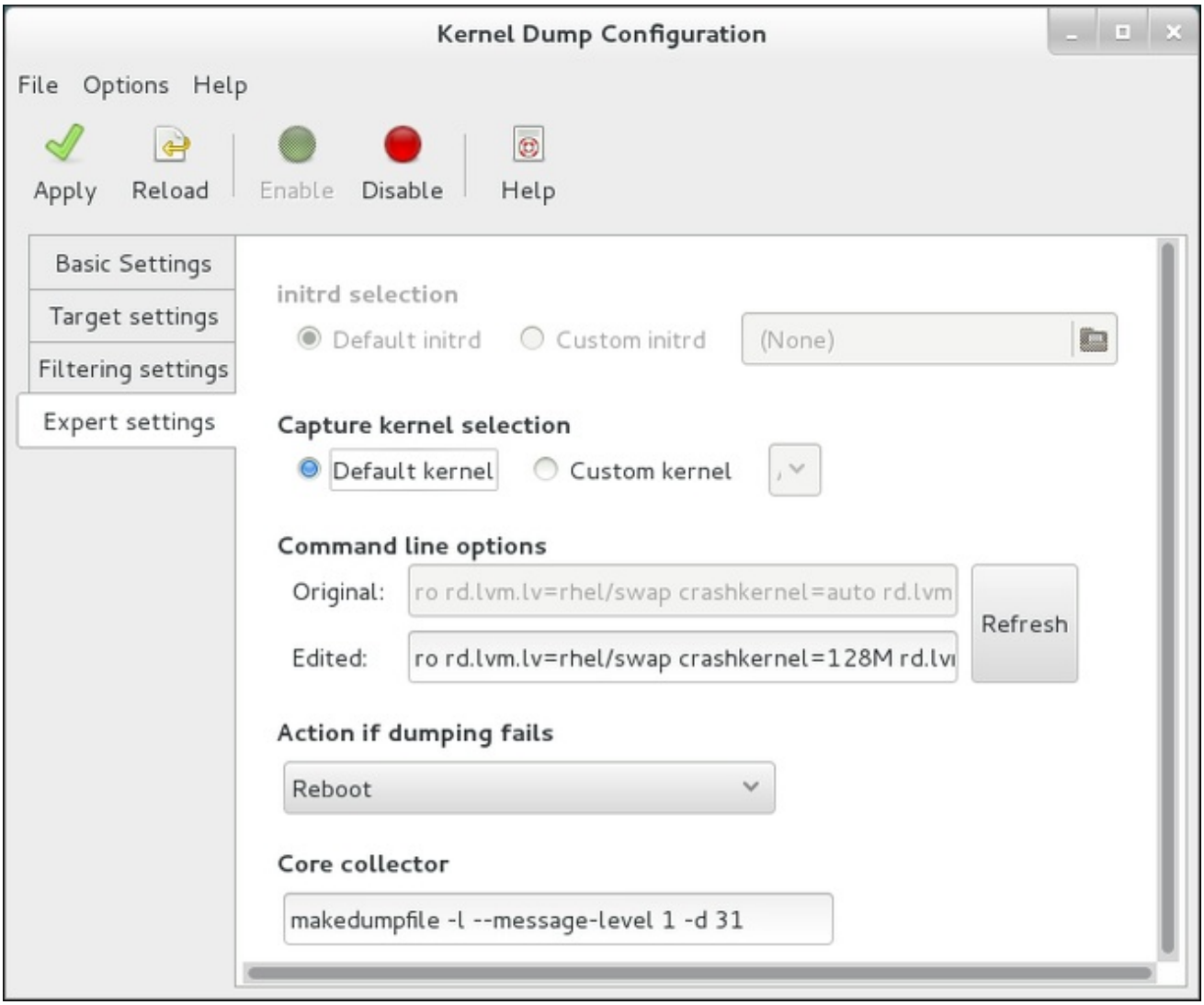


ダンプから **ゼロページ**、**キャッシュページ**、**キャッシュプライベート**、**ユーザーデータ**、または **フリーページ** を除外するには該当ラベルの横にあるチェックボックスを使って選択します。

1.2.3.4. デフォルト動作の設定

kdump がコアダンプの作成に失敗した場合に行う動作を選択するには、**デフォルトの動作** のドロップダウンリストから適切なオプションを選択します。**rootfs にダンプして再起動** (コアをローカルに保存してからシステムを再起動するデフォルトの動作)、**再起動** (システムを再起動)、**shell** (対話式シェルプロンプトを表示)、**停止** (システムを停止)、**電源オフ** (システムの電源を切断) などのオプションを選択することができます。

図1.4 フィルタリング



makedumpfile コアコレクターに渡されるオプションをカスタマイズするには **コアコレクター** のテキストフィールドを編集します。詳細は 「コアコレクターの設定」 を参照してください。

1.2.3.5. サービスの有効化

起動時に **kdump** サービスを開始する場合はツールバーの **有効化** ボタンをクリックしてから **適用** ボタンをクリックします。これにより、**multi-user.target** のサービスが有効になり、起動します。**無効化** ボタンをクリックして**適用** ボタンをクリックするとサービスが直ちに無効になります。

重要

Red Hat Enterprise Linux 7 では kdump の出力先として指定されているディレクトリーが **kdump** systemd サービスの起動時に存在していなければなりません。起動時にこのディレクトリーが存在していないとサービスの起動は失敗します。この動作は Red Hat Enterprise Linux の以前のリリースと異なります。以前のリリースではサービスの起動時にこのディレクトリーが存在していないと自動的に作成されていました。

systemd の出力先およびサービスの設定全般については『**Red Hat Enterprise Linux 7 システム管理者のガイド**』を参照してください。

1.3. KDUMP 設定のテスト

警告

以下のコマンドを実行するとカーネルがクラッシュします。次の手順を行う場合は十分に注意してください。実稼働のシステムでは絶対に実行しないでください。

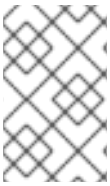
設定をテストするため **kdump** を有効にしてシステムを再起動し、サービスが実行されているか確認します。

```
~]# systemctl is-active kdump
active
```

次に、シェルプロンプトで以下のコマンドを入力します。

```
echo 1 > /proc/sys/kernel/sysrq
echo c > /proc/sysrq-trigger
```

このコマンドにより、Linux カーネルは強制的にクラッシュして **address-YYYY-MM-DD-HH:MM:SS/vmcore** ファイルが設定で選択した場所にコピーされます (デフォルトでは **/var/crash/**)。



注記

このアクションは、設定の妥当性を確認するのに加え、典型的なテストロードで実行された場合にクラッシュダンプが完了するまでの所要時間を記録するために使用できます。

1.3.1. その他のリソース

1.3.1.1. インストールされているドキュメント

- **kdump.conf**(5): **/etc/kdump.conf** 設定ファイルの man ページです。使用できるオプションの詳細な説明が参照できます。
- **zipl.conf**(5): **/etc/zipl.conf** 設定ファイルの man ページです。
- **zipl**(8): IBM System z 向けの **zipl** ブートローダーユーティリティーの man ページです。
- **makedumpfile**(8): **makedumpfile** コアコレクターの man ページです。
- **kexec**(8): **kexec** の man ページです。
- **crash**(8): **crash** ユーティリティーの man ページです。
- **/usr/share/doc/kexec-tools-version/kexec-kdump-howto.txt**: **kdump** および **kexec** のインストールと使用方法に関する概要です。

1.3.1.2. オンラインのドキュメント

https://access.redhat.com/site/solutions/6038

kexec および **kdump** 設定に関する Red Hat ナレッジベースアーティクルです。

https://access.redhat.com/site/solutions/223773

サポートしている **kdump** 出力先に関する Red Hat ナレッジベースのアーティクルです。

http://people.redhat.com/anderson/

crash ユーティリティーのホームページです。

https://www.gnu.org/software/grub/

GRUB2 ブートローダーのホームページとドキュメントです。

1.4. ファームウェア支援ダンプの仕組み

1.4.1. ファームウェア支援ダンプについて

kexec および **kdump** の仕組みは、AMD64 および Intel 64 システムでコアダンプをキャプチャーする、信頼性が高く実績のある手法ですが、長い歴史を持つ一部のハードウェア (特に、ミニシステムおよびメインフレームシステム) では、オンボードのファームウェアを活用して、メモリーの領域を分離してクラッシュ解析に重要なデータが誤って上書きされるのを防ぐことができます。

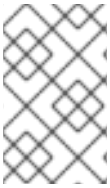
本章では、利用可能なファームウェア支援ダンプの手法について紹介し、ファームウェア支援ダンプを Red Hat Enterprise Linux でどのように活用するかについて説明します。

1.4.2. IBM PowerPC ハードウェアにおける fadump の使用

ファームウェア支援ダンプ (**fadump**) は、IBM PowerPC LPARS で利用可能な **kexec-kdump** に代わる信頼性の高い仕組みです。ファームウェア支援ダンプでは、PCI および I/O デバイスが再初期化され、完全にリセットされたシステムから、vmcore がキャプチャーされます。この仕組みでは、クラッシュ発生時にファームウェアを使ってメモリーが保持されますが、**kdump** ユーザー空間スクリプトが再利用して vmcore を保存します。

そのために、**fadump** では、クラッシュ発生時にシステムファームウェアを使って保持する必要のあるメモリー領域が登録されます。これらの領域には、ブートメモリー、システムレジスター、およびハードウェアのページテーブルエントリー (PTE) を除く、すべてのシステムメモリーコンテンツが含まれます。

PowerPC 特有のハードウェアのリセット方法を含め、**fadump** の仕組みに関する詳細は、**/usr/share/doc/kexec-tools-X.y.z/fadump-howto.txt** を確認してください。ここで「X.y.z」は、お使いのシステムにインストールされている **kexec-tools** のバージョン番号を表します。



注記

ブートメモリー と呼ばれる保持されないメモリー領域は、クラッシュ後にカーネルを正常に起動するのに必要な RAM の容量です。デフォルトのブートメモリーサイズは、256 MB または全システム RAM の 5% のいずれか大きい方です。

kexec で開始されたイベントとは異なり、**fadump** プロセスでは実稼働用のカーネルを使用してクラッシュダンプを復元します。クラッシュ後の起動時に、PowerPC ハードウェアはデバイスノード **/proc/device-tree/rtas/ibm,kernel-dump** が **procfs** で利用できるようにし、fadump 対応の **kdump** スクリプトは vmcore を保存するかどうかを確認します。この処理が完了すると、システムは正しく再起動されます。

fadump の有効化

1. 「kdump のインストールと設定」 の記載通りに、**kdump** をインストールし、設定します。

2. **/etc/default/grub** の **GRUB_CMDLINE_LINUX** の行に、**fadump=on** を追加します。

```
GRUB_CMDLINE_LINUX="rd.lvm.lv=rhel/swap crashkernel=auto
rd.lvm.lv=rhel/root rhgb quiet fadump=on"
```

3. (この操作は必須ではありません) 予約ブートメモリーサイズにデフォルト値を使わずに具体的な値を指定する場合は、**GRUB_CMDLINE_LINUX** in **/etc/default/grub** に **fadump_reserve_mem=xxM** を追加します。**xx** は必要なメモリーサイズ (メガバイト単位) に指定してください。

```
GRUB_CMDLINE_LINUX="rd.lvm.lv=rhel/swap crashkernel=auto
rd.lvm.lv=rhel/root rhgb quiet fadump=on fadump_reserve_mem=xxM"
```

重要

すべてのブート設定オプションと同様に、必要になる前に設定をテストすることを強く推奨します。クラッシュカーネルから起動時に Out of Memory (OOM) エラーが発生する場合は、クラッシュカーネルが正常に起動できるまで **fadump_reserve_mem=** で指定する値を増やします。この場合、試行錯誤が必要になることがあります。

1.4.3. IBM z Systems におけるファームウェア支援ダンプの手法

IBM z Systems には、**Stand-alone Dump** および **VMDUMP** の 2 つのファームウェア支援ダンプの仕組みがあります。

これらのシステムでは **kdump** インフラストラクチャーがサポートされ使用されています。Red Hat Enterprise Linux での設定は、「kdump のインストールと設定」 に説明がありますが、IBM z System ハードウェアが提供するこれらのファームウェア支援の手法を使用すると、いくつかのメリットが得られます。

スタンドアロンダンプ (SADMP) メカニズムはシステムコンソールから開始および制御され、IPL 起動可能デバイス上に保管される必要があります。

VMDUMP は SADMP と類似しています。このツールもシステムコンソールから開始されますが、得られるダンプをハードウェアからコピーし、解析のためにそれをシステムに格納する仕組みがあります。

(他のハードウェアベースのダンプメカニズムと同様に) これらの手法のメリットの 1 つは、(kdump サービスが開始される前の) 起動初期段階におけるマシンの状態をキャプチャーできるという点です。

VMDUMP には、ハードウェアからコピーしたダンプファイルを Red Hat Enterprise Linux システムに格納する仕組みがありますが、IBM z System ハードウェアコンソールから、SADMP および VMDUMP の両方の設定および制御が管理されます。

IBM では、これらの仕組みについて詳細な説明を用意しています。SADMP については http://www.ibm.com/support/knowledgecenter/SSLTBW_2.1.0/com.ibm.zos.v2r1.ieav100/standa.htm を、VMDUMP については http://www.ibm.com/support/knowledgecenter/en/linuxonibm/com.ibm.linux.z.lgdt/lgdt_t_vmdump.html を、それぞれ参照してください。

IBM では、Red Hat Enterprise Linux 7 におけるダンプツールの使用についても文書化しています。http://www.ibm.com/support/knowledgecenter/linuxonibm/com.ibm.linux.z.lgdt/lgdt_t_usingdumptools.h を参照してください。

1.4.4. Fujitsu PRIMEQUEST システムにおける sadump の使用

Fujitsu の **sadump** メカニズムは、**kdump** が正常に完了できない場合にフォールバックダンプキャプチャーが実行されるように設計されています。

sadump プロセスは、システムの ManageMent Board (MMB) インターフェースから手動で呼び出します。

このシステムでは、通常通り kdump を X86_64 サーバーに対して設定し、さらに以下の追加ステップを実施して **sadump** を有効にする必要があります。

sadump に対して **kdump** が予想どおりに起動するように**/etc/sysctl.conf** で以下の行を追加または編集します。

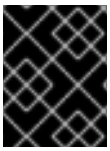
```
kernel.panic=0
kernel.unknown_nmi_panic=1
```

上記の手順に加えて、**/etc/kdump.conf** にいくつかのオプションを追加して、sadump に対して **kdump** が正常に動作するようにする必要もあります。

特に、**kdump** の後にシステムが再起動しないようにする必要があります。**kdump** がコアの保存に失敗した後にシステムが再起動すると、**sadump** を呼び出す機会が失われます。

そのためには、**/etc/kdump.conf** の **default** アクションを **halt** または **shell** のどちらかに設定する必要があります。

```
default shell
blacklist kvm-intel
```



重要

sadump 用にハードウェアを設定する方法は、『FUJITSU Server PRIMEQUEST 2000 Series Installation Manual』を参照してください。

1.5. コアダンプの分析

システムクラッシュの原因を究明するには、GNU Debugger (GDB) と良く似た対話式のプロンプト **crash** ユーティリティーを使用することができます。稼働中の Linux システムだけでなく **netdump**、**diskdump**、**xendump**、**kdump** などで作成したコアダンプなども対話形式で分析することができます。

1.5.1. crash ユーティリティーのインストール

crash 分析ツールをインストールするには、**root** でシェルプロンプトから次のコマンドを実行します。

```
yum install crash
```

crash に加え、実行中のカーネルに対応する **kernel-debuginfo** パッケージもインストールしておく必要があります。このパッケージでダンプ分析に必要なデータが提供されます。**kernel-debuginfo** をインストールするには、**root** として **debuginfo-install** を使用します。

```
debuginfo-install kernel
```

Yum を使用した Red Hat Enterprise Linux の新規パッケージのインストール方法については『**Red Hat Enterprise Linux 7 システム管理者のガイド**』を参照してください。

1.5.2. crash ユーティリティーの実行

シェルプロンプトで次の形式のコマンドを入力してユーティリティーを起動します。

```
crash /usr/lib/debug/lib/modules/<kernel>/vmlinux \
/var/crash/<timestamp>/vmcore
```

kdump によってキャプチャーされたものと同じ **<kernel>** のバージョンを使用してください。現在実行中のカーネルを確認するには、**uname -r** のコマンドを使用します。

例1.1 crash ユーティリティーの実行

```
~]# crash /usr/lib/debug/lib/modules/2.6.32-69.el6.i686/vmlinux \
/var/crash/127.0.0.1-2010-08-25-08:45:02/vmcore

crash 5.0.0-23.el6
Copyright (C) 2002-2010  Red Hat, Inc.
Copyright (C) 2004, 2005, 2006  IBM Corporation
Copyright (C) 1999-2006  Hewlett-Packard Co
Copyright (C) 2005, 2006  Fujitsu Limited
Copyright (C) 2006, 2007  VA Linux Systems Japan K.K.
Copyright (C) 2005  NEC Corporation
Copyright (C) 1999, 2002, 2007  Silicon Graphics, Inc.
Copyright (C) 1999, 2000, 2001, 2002  Mission Critical Linux, Inc.
This program is free software, covered by the GNU General Public
License,
and you are welcome to change it and/or distribute copies of it under
certain conditions.  Enter "help copying" to see the conditions.
This program has absolutely no warranty.  Enter "help warranty" for
details.

GNU gdb (GDB) 7.0
Copyright (C) 2009 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later
<http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show
copying"
and "show warranty" for details.
This GDB was configured as "i686-pc-linux-gnu"...

      KERNEL: /usr/lib/debug/lib/modules/2.6.32-69.el6.i686/vmlinux
    DUMPFILE: /var/crash/127.0.0.1-2010-08-25-08:45:02/vmcore  [PARTIAL
DUMP]
        CPUS: 4
        DATE: Wed Aug 25 08:44:47 2010
      UPTIME: 00:09:02
LOAD AVERAGE: 0.00, 0.01, 0.00
       TASKS: 140
    NODENAME: hp-dl320g5-02.lab.bos.redhat.com
     RELEASE: 2.6.32-69.el6.i686
     VERSION: #1 SMP Tue Aug 24 10:31:45 EDT 2010
```

```
      MACHINE: i686  (2394 Mhz)
       MEMORY: 8 GB
        PANIC: "Oops: 0002 [#1] SMP " (check log for details)
          PID: 5591
      COMMAND: "bash"
         TASK: f196d560  [THREAD_INFO: ef4da000]
          CPU: 2
        STATE: TASK_RUNNING (PANIC)

crash>
```

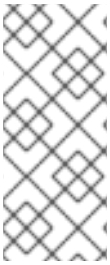
1.5.3. メッセージバッファーの表示

対話式プロンプトで **log** コマンドを入力して、カーネルメッセージバッファーを表示します。

例1.2 カーネルメッセージバッファーの表示

```
crash> log
... several lines omitted ...
EIP: 0060:[<c068124f>] EFLAGS: 00010096 CPU: 2
EIP is at sysrq_handle_crash+0xf/0x20
EAX: 00000063 EBX: 00000063 ECX: c09e1c8c EDX: 00000000
ESI: c0a09ca0 EDI: 00000286 EBP: 00000000 ESP: ef4dbf24
 DS: 007b ES: 007b FS: 00d8 GS: 00e0 SS: 0068
Process bash (pid: 5591, ti=ef4da000 task=f196d560 task.ti=ef4da000)
Stack:
 c068146b c0960891 c0968653 00000003 00000000 00000002 efade5c0 c06814d0
<0> fffffffb c068150f b7776000 f2600c40 c0569ec4 ef4dbf9c 00000002
b7776000
<0> efade5c0 00000002 b7776000 c0569e60 c051de50 ef4dbf9c f196d560
ef4dbfb4
Call Trace:
 [<c068146b>] ? __handle_sysrq+0xfb/0x160
 [<c06814d0>] ? write_sysrq_trigger+0x0/0x50
 [<c068150f>] ? write_sysrq_trigger+0x3f/0x50
 [<c0569ec4>] ? proc_reg_write+0x64/0xa0
 [<c0569e60>] ? proc_reg_write+0x0/0xa0
 [<c051de50>] ? vfs_write+0xa0/0x190
 [<c051e8d1>] ? sys_write+0x41/0x70
 [<c0409adc>] ? syscall_call+0x7/0xb
Code: a0 c0 01 0f b6 41 03 19 d2 f7 d2 83 e2 03 83 e0 cf c1 e2 04 09 d0
88 41 03 f3 c3 90 c7 05 c8 1b 9e c0 01 00 00 00 0f ae f8 89 f6 <c6> 05
00 00 00 00 01 c3 89 f6 8d bc 27 00 00 00 00 8d 50 d0 83
EIP: [<c068124f>] sysrq_handle_crash+0xf/0x20 SS:ESP 0068:ef4dbf24
CR2: 0000000000000000
```

このコマンドの使用方法についての詳しい情報を参照するには、**help log** と入力してください。



注記

カーネルメッセージバッファーには、システムクラッシュに関する不可欠な情報が含まれるので、必ず最初に **vmcore-dmesg.txt** ファイルにダンプされます。これは、保存先の容量が少ないことなどが原因で完全な **vmcore** ファイルの取得に失敗した場合に有用です。デフォルトでは、**vmcore-dmesg.txt** は **/var/crash/** ディレクトリーに配置されています。

1.5.4. バックトレースの表示

対話式プロンプトで **bt** コマンドを入力してカーネルのスタックトレースを表示します。1 プロセスのバックトレースを表示するには **bt <pid>** と入力します。

例1.3 カーネルスタックトレースの表示

```
crash> bt
PID: 5591   TASK: f196d560  CPU: 2   COMMAND: "bash"
 #0 [ef4dbdcc] crash_kexec at c0494922
 #1 [ef4dbe20] oops_end at c080e402
 #2 [ef4dbe34] no_context at c043089d
 #3 [ef4dbe58] bad_area at c0430b26
 #4 [ef4dbe6c] do_page_fault at c080fb9b
 #5 [ef4dbee4] error_code (via page_fault) at c080d809
    EAX: 00000063  EBX: 00000063  ECX: c09e1c8c  EDX: 00000000  EBP:
00000000
    DS:  007b      ESI: c0a09ca0  ES:  007b      EDI: 00000286  GS:
00e0
    CS:  0060      EIP: c068124f  ERR: ffffffff  EFLAGS: 00010096
 #6 [ef4dbf18] sysrq_handle_crash at c068124f
 #7 [ef4dbf24] __handle_sysrq at c0681469
 #8 [ef4dbf48] write_sysrq_trigger at c068150a
 #9 [ef4dbf54] proc_reg_write at c0569ec2
#10 [ef4dbf74] vfs_write at c051de4e
#11 [ef4dbf94] sys_write at c051e8cc
#12 [ef4dbfb0] system_call at c0409ad5
    EAX: fffffffda  EBX: 00000001  ECX: b7776000  EDX: 00000002
    DS:  007b      ESI: 00000002  ES:  007b      EDI: b7776000
    SS:  007b      ESP: bfcb2088  EBP: bfcb20b4  GS:  0033
    CS:  0073      EIP: 00edc416  ERR: 00000004  EFLAGS: 00000246
```

このコマンドの使用方法についての詳しい情報を参照するには、**help bt** と入力してください。

1.5.5. プロセスの状態表示

対話式プロンプトで **ps** コマンドを入力してシステム内のプロセスの状態を表示します。1 プロセスの状態を表示するには **ps <pid>** と入力します。

例1.4 システム内のプロセスの状態表示

```
crash> ps
   PID    PPID  CPU   TASK    ST  %MEM     VSZ    RSS  COMM
>     0      0   0  c09dc560  RU   0.0       0      0  [swapper]
>     0      0   1  f7072030  RU   0.0       0      0  [swapper]
```

```
      0      0   2  f70a3a90  RU   0.0       0      0  [swapper]
>     0      0   3  f70ac560  RU   0.0       0      0  [swapper]
      1      0   1  f705ba90  IN   0.0    2828   1424  init
... several lines omitted ...
   5566      1   1  f2592560  IN   0.0   12876    784  auditd
   5567      1   2  ef427560  IN   0.0   12876    784  auditd
   5587   5132   0  f196d030  IN   0.0   11064   3184  sshd
>  5591   5587   2  f196d560  RU   0.0    5084   1648  bash
```

このコマンドの使用方法についての詳しい情報を参照するには、**help ps** と入力してください。

1.5.6. 仮想メモリー情報の表示

対話式プロンプトで **vm** コマンドを入力して仮想メモリーの基本情報を表示します。1 プロセスの情報を表示するには **vm <pid>** と入力します。

例1.5 現在のコンテキストの仮想メモリー情報の表示

```
crash> vm
PID: 5591   TASK: f196d560  CPU: 2   COMMAND: "bash"
   MM       PGD      RSS    TOTAL_VM
f19b5900  ef9c6000  1648k    5084k
  VMA       START      END    FLAGS  FILE
f1bb0310    242000    260000 8000875  /lib/ld-2.12.so
f26af0b8    260000    261000 8100871  /lib/ld-2.12.so
efbc275c    261000    262000 8100873  /lib/ld-2.12.so
efbc2a18    268000    3ed000 8000075  /lib/libc-2.12.so
efbc23d8    3ed000    3ee000 8000070  /lib/libc-2.12.so
efbc2888    3ee000    3f0000 8100071  /lib/libc-2.12.so
efbc2cd4    3f0000    3f1000 8100073  /lib/libc-2.12.so
efbc243c    3f1000    3f4000 100073
efbc28ec    3f6000    3f9000 8000075  /lib/libdl-2.12.so
efbc2568    3f9000    3fa000 8100071  /lib/libdl-2.12.so
efbc2f2c    3fa000    3fb000 8100073  /lib/libdl-2.12.so
f26af888    7e6000    7fc000 8000075  /lib/libtinfo.so.5.7
f26aff2c    7fc000    7ff000 8100073  /lib/libtinfo.so.5.7
efbc211c    d83000    d8f000 8000075  /lib/libnss_files-2.12.so
efbc2504    d8f000    d90000 8100071  /lib/libnss_files-2.12.so
efbc2950    d90000    d91000 8100073  /lib/libnss_files-2.12.so
f26afe00    edc000    edd000 4040075
f1bb0a18   8047000   8118000 8001875  /bin/bash
f1bb01e4   8118000   811d000 8101873  /bin/bash
f1bb0c70   811d000   8122000 100073
f26afae0   9fd9000   9ffa000 100073
... several lines omitted ...
```

このコマンドの使用方法についての詳しい情報を参照するには、**help vm** と入力してください。

1.5.7. オープンファイルの表示

対話式プロンプトで **files** コマンドを入力してオープンファイルに関する情報を表示します。選択した 1 プロセスだけで開かれているファイルを表示するには **files <pid>** と入力します。

例1.6 現在のコンテキストのオープンファイルについての情報の表示

```
crash> files
PID: 5591   TASK: f196d560  CPU: 2   COMMAND: "bash"
ROOT: /    CWD: /root
 FD    FILE     DENTRY    INODE    TYPE  PATH
  0  f734f640  eedc2c6c  eecd6048  CHR   /pts/0
  1  efade5c0  eee14090  f00431d4  REG   /proc/sysrq-trigger
  2  f734f640  eedc2c6c  eecd6048  CHR   /pts/0
 10  f734f640  eedc2c6c  eecd6048  CHR   /pts/0
255  f734f640  eedc2c6c  eecd6048  CHR   /pts/0
```

このコマンドの使用方法についての詳しい情報を参照するには、**help files** と入力してください。

1.5.8. ユーティリティーの終了

対話式プロンプトを終了して、**crash** を終了するには、**exit** または **q** と入力します。

例1.7 crash ユーティリティーの終了

```
crash> exit
~]#
```

1.6. よくある質問

クラスター化された環境で kdump を使用する場合には、何に注意したら良いですか?

「RHEL 6 および 7 の High Availability アドオンで使用できるように kdump を設定する」の記事で、High Availability アドオンを使用しているシステム管理者が利用可能なオプションを説明しています。

起動初期に kdump が失敗します。どのようにしてブートログをキャプチャーするのですか?

2 番目のカーネルの起動に問題がある場合は、起動初期のブートログを確認する必要があります。問題のあるマシンに対するシリアルコンソールを有効にすることで、このログを取得することができます。

「Red Hat Enterprise Linux でシリアルターミナルやコンソールを設定する」の記事で、起動初期のブートメッセージへアクセスするために必要な設定が説明されています。

デバッグ用に、どのようにして makedumpfile からのメッセージを増やすのですか?

makedumpfile が失敗する時には、何が悪いのかを理解するためにログのレベルを増やさなければならない場合があります。これはダンプレベルを設定するのとは異なる操作です。ログのレベルについては、**/etc/kdump.conf** の **core_collector** 行の内容を編集して、**makedumpfile** に対する **message_level** オプションを増やします。

デフォルトでは **makedumpfile** はレベル 7 に設定されていて、進捗バー、共通メッセージ、およびエラーメッセージの出力が含まれます。このレベルを 31 に設定して、詳細なデバッグ情報を取得します。

core_collector 設定の行は、設定時には以下のようになっているはずです。

```
core_collector makedumpfile -l --message-level 1 -d 31
```

どのようにして Dracut をデバッグするのですか?

dracut が initramfs の構築に失敗することがあります。その場合には、問題を切り分けるために **dracut** のログレベルを増やす必要があります。

/etc/kdump.conf を編集して **dracut_args** 行を変更し、必要なすべての dracut 引数と共にオプション **-L 5** を含めます。

dracut_args で他のオプションを設定していない場合は、次のような結果になるはずです。

```
dracut_args -L 5
```

仮想マシンで利用可能なダンプの手法は何ですか?

多くの場合、クラッシュやパニックの後にマシンからメモリーダンプを取得するには **kdump** の仕組みがあれば十分です。これは、ベアメタルのインストールと同じように設定することができます。

ただし、場合によっては、ハイパーバイザーと直接連携してクラッシュダンプを取得する必要がある場合があります。ハイパーバイザーを使用したクラッシュダンプの取得方法として **libvirt** には **pvpanic** と **virsh dump** という 2 つの仕組みがあります。これらの手法は 『仮想化の導入および管理ガイド』 に記載されています。

pvpanic の仕組みについては、『仮想化の導入および管理ガイド』の「パニックデバイスの設定」に記載されています。

virsh dump コマンドについては、『仮想化の導入および管理ガイド』の「ドメインのコアのダンプファイルの作成」で説明されています。

Red Hat サポートサービスに大きなサイズのダンプをアップロードする場合はどうしたらいいですか？

分析のため、Red Hat グローバルサポートサービスにカーネルクラッシュのダンプファイルを送信していただく必要がある場合があります。ただし、ダンプファイルのサイズはフィルターで特定の情報に絞り込んだ場合でも非常に大きくなる可能性があります。新しいサポートケースの起票時に、250 MB を超えるファイルは Red Hat カスタマーポータルからは直接アップロードできないため、Red Hat では大きなサイズのファイルのアップロード用に FTP サーバーを用意しています。

FTP サーバーのアドレスは **dropbox.redhat.com** で、ファイルは **/incoming/** ディレクトリーにアップロードしてください。ご使用の FTP クライアントを passive モードに設定しておく必要があります。ご使用のファイアウォールでこのモードを使用できない場合は **origin-dropbox.redhat.com** サーバーを active モードでご利用ください。

アップロードするファイルは必ず **gzip** などで圧縮し、ファイル名にはわかりやすい名前を付けてください。ファイル名にサポートケース番号を使用されることをお薦めします。必要なファイルをすべてアップロードしたらサポートケース担当のエンジニアへ正確なファイル名とその SHA1 または MD5 チェックサムをお知らせください。

詳細な説明については「Ret Hat サポートチームにファイル (vmcore、rhev logcollector、sosreport、ヒープダンプ、ログファイルなど) を送付する 」を参照ください。

クラッシュダンプが完了するまでに、どれくらい時間がかかりますか?

災害対策計画を作成するためには、通常、ダンプ完了までの時間を把握している必要がありますが、所要時間は、ディスクにコピーされるメモリーの量や、メモリーとストレージのインターフェースのスピードに大きく左右されます。

テストがどのようなタイミングで行われても、システムは典型的な負荷をかけた状態で稼働させておく必要があります。そうでないと、除外されるページによって、完全に負荷がかかった実稼働システムにおける kdump の動作で誤った見解が提示される可能性があります。特にこの違いは、大量のメモリーが使用されている状況でより顕著に見られます。

ダンプの所要時間を評価する場合に、ストレージインターフェースもプランニング時に考慮する必要があります。ネットワーク制約事項が原因で、ローカルに接続された SATA ディスクと比べると、**ssh** などを使用した接続ダンプは完了までに長く時間がかかる可能性があります。

インストール時に Kdump をどのように設定するのですか?

キックスタートまたは対話型の GUI を使用すれば、わずかなオプションを指定するだけでインストール時に **kdump** を設定することができます。

anaconda インストール GUI を使用した **kdump** の設定については、『インストールガイド』の「Kdump」セクションに詳しい説明が記載されています。

kickstart 構文は以下の通りです。

```
%addon com_redhat_kdump [--disable,enable] [--reserve-mb=[auto,value]]
%end
```

キックスタートに対するこのアドオンにより、kdump 機能の無効化/有効化や、オプションとして予約メモリーサイズの定義 (自動のデフォルトオプションを明示的に呼び出す、またはメガバイト単位で数値を指定する) が可能になります。なお、スイッチ全体が省略された場合も、デフォルトオプションが呼び出されます。

キックスタートを使用してシステムのデプロイメントを自動化する方法の詳細は、『インストールガイド』の「キックスタートを使ったインストール」を参照してください。

キックスタートアドオン構文の詳細については、『インストールガイド』の「キックスタート構文の参考資料」を参照してください。

1.7. サポートしている KDUMP の設定とダンプ出力先

1.7.1. kdump メモリー要件

kdump でカーネルクラッシュダンプをキャプチャーしてさらに分析ができるように保存するにはシステムメモリーの一部をキャプチャーカーネル用に永続的に予約する必要があります。以下の表には、システムのアーキテクチャーおよび利用可能な合計物理メモリーを基にした kdump の最小メモリー要件一覧が含まれます。

コマンドラインでメモリー設定を変更する方法については 「メモリー使用量の設定」 を参照してください。グラフィカルユーザーインターフェースで予約メモリーの設定を変更する方法については 「メモリー使用量の設定」 を参照してください。

表1.1 kdump 用に必要な最小予約メモリー

アーキテクチャー	使用可能なメモリー	最小予約メモリー
AMD64 および Intel 64 (**x86_64**)	2 GB 以上	メモリー 4KB 毎に 160 MB + 2 ビット、メモリーが 1 TB のシステムの場合は、最低でも 224 MB 必要です (160 + 64 MB)。
IBM POWER (**ppc64**)	2 GB から 4 GB	256 MB のメモリー
	4 GB から 32 GB	512 MB のメモリー
	32 GB から 64 GB	1 GB のメモリー
	64 GB から 128 GB	2 GB のメモリー
	128 GB 以上	4 GB のメモリー
IBM System z (**s390x**)	2 GB 以上	メモリー 4KB 毎に 160 MB + 2 ビット、メモリーが 1 TB のシステムの場合は、最低でも 224 MB 必要です (160 + 64 MB)。

さまざまな Red Hat Enterprise Linux テクノロジーの機能や制限に関する詳しい情報は、https://access.redhat.com/articles/rhel-limits を参照してください。

1.7.2. メモリー自動予約の最小しきい値

一部のシステムでは、ブートローダーの設定ファイルで **crashkernel=auto** パラメーターを使用するか、グラフィカル設定ユーティリティーで自動割り当ての設定を有効にすると、kdump 用のメモリーを自動的に割り当てることができます。ただし、この自動予約が正常に機能するためには、特定のメモリー容量合計が利用できる状態でなければなりません。この容量はシステムのアーキテクチャーによって異なります。

以下の表で、自動メモリー割り当てのしきい値を一覧で表示します。システムのメモリーが以下に示すしきい値を下回る場合は手動でメモリー予約を行う必要があります。

コマンドラインで設定を変更する方法については 「メモリー使用量の設定」 を参照してください。グラフィカルユーザーインターフェースで予約メモリーのサイズを変更する方法については 「メモリー使用量の設定」 を参照してください。

表1.2 自動メモリー予約に必要な最小メモリーサイズ

アーキテクチャー	必要なメモリー
AMD64 および Intel 64 (**x86_64**)	2 GB

アーキテクチャー	必要なメモリー
IBM POWER (**ppc64**)	2 GB
IBM System z (**s390x**)	4 GB

1.7.3. サポートしている kdump のダンプ出力先

カーネルクラッシュをキャプチャーする際、コアダンプを直接デバイスに書き込んでローカルファイルシステムにファイルとして保存するか、またはネットワーク経由で送信することができます。現在サポートしているダンプ出力先および kdump による非サポートが明確なダンプ出力先の全一覧を以下に示します。

コマンドラインでダンプ出力先を設定する方法については 「kdump タイプの設定」 を参照してください。グラフィカルユーザーインターフェースでダンプ出力先を設定する方法については 「kdump タイプの設定」 を参照してください。

表1.3 サポートしている kdump のダンプ出力先

タイプ	サポートしているダンプ出力先	非サポートのダンプ出力先
Raw デバイス	ローカルで添付されたすべての生デバイスとパーティション	
ローカルファイルシステム	直接アタッチされたディスクドライブ、ハードウェア RAID 論理ドライブ、LVM デバイス、**mdraid** アレイ上の **ext2**、**ext3**、**ext4**、**btrfs** および **xfs** ファイルシステム	**auto** タイプ (自動ファイルシステム検出) など、この表で明示的にサポート対象とされていないローカルファイルシステム
リモートディレクトリー	**IPv4** で **NFS** または **SSH** プロトコルを使用してアクセスするリモートのディレクトリー	**NFS** プロトコルを使用してアクセスする **rootfs** ファイルシステム上のリモートディレクトリー
ハードウェアおよびソフトウェアイニシエーター上で **iSCSI** プロトコルを使用してアクセスするリモートディレクトリー	**be2iscsi** ハードウェア上で **iSCSI** プロトコルを使用してアクセスするリモートディレクトリー	マルチパスベースのストレージ
		IPv6 上でアクセスするリモートディレクトリー
		SMB または **CIFS** プロトコルを使用してアクセスするリモートディレクトリー
		FCoE (**Fibre Channel over Ethernet**) プロトコルを使用してアクセスするリモートディレクトリー

タイプ	サポートしているダンプ出力先	非サポートのダンプ出力先
		ワイヤレスネットワークインターフェースを使ってアクセスするリモートディレクトリー

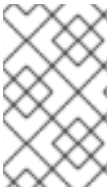
1.7.4. サポートしている kdump のフィルターレベル

ダンプファイルのサイズを縮小させるため kdump では **makedumpfile** コアコレクターを使ってデータを圧縮して必要に応じて関連性のない情報を除外します。以下の表では、現在、**makedumpfile** ユーティリティーでサポートしているフィルターのレベル全一覧を紹介します。

コマンドラインでコアコレクターを設定する方法については 「コアコレクターの設定」 を参照してください。グラフィカルインターフェースでコアコレクターを設定する方法については 「コアコレクターの設定」 を参照してください。

表1.4 サポートしているフィルターレベル

オプション	説明
1	ゼロページ
2	キャッシュページ
4	キャッシュプライベート
8	ユーザーページ
16	フリーページ



注記

makedumpfile コマンドは、Red Hat Enterprise Linux 7.3 以降の透過的なヒュージページおよび hugetlbfs ページの削除をサポートします。これらのタイプのヒュージページは、ユーザーページと見なされ、**-8** レベルを使用して削除されます。

1.7.5. サポートしているデフォルトの動作

kdump がコアダンプの作成に失敗するとデフォルトでは root ファイルシステムをマウントしてコアのローカルへの保存を試行します。第 1 ダンプ出力先へのコアダンプの保存に失敗した場合、kdump に別の動作を行うよう設定することができます。kdump で現在サポートしているデフォルト動作を以下に示します。

コマンドラインでデフォルト動作を設定する方法については 「デフォルト動作の設定」 を参照してください。グラフィカルユーザーインターフェースでデフォルト動作を設定する方法については 「デフォルト動作の設定」 を参照してください。

表1.5 サポートしているデフォルトの動作

オプション	説明
dump_to_rootfs	コアダンプの root ファイルシステムへの保存を試行します。ネットワーク上のダンプ出力先と併用する場合に特に便利なオプションです。ネットワーク上のダンプ出力先にアクセスできない場合、ローカルにコアダンプを保存するよう kdump の設定を行います。ダンプ後システムは再起動されます。
reboot	システムを再起動します。コアダンプは失われます。
halt	システムを停止します。コアダンプは失われます。
poweroff	システムの電源を切ります。コアダンプは失われます。
shell	initramfs 内から shell セッションを実行して、ユーザーが手動でコアダンプを記録できるようにします。

1.7.6. kdump サイズの予測

kdump 環境のプランニングや構築の際に、ダンプファイルに必要な容量がどれくらいか把握してから作成する必要があります。これには、**makedumpfile** コマンドを使用すると役立ちます。

--mem-usage オプションでは、除外可能なページに関する有用なレポートが提供され、これを使用して、割り当てるダンプレベルを判断することができます。このコマンドは、システムに典型的な負荷をかけた状態で実行する必要があります。そうでないと、**makedumpfile** は実稼動環境で必要とされる値よりも小さい値を返します。

```
[root@hostname ~]# makedumpfile --mem-usage /proc/kcore

TYPE            PAGES                   EXCLUDABLE      DESCRIPTION
----------------------------------------------------------------------
ZERO            501635                  yes             Pages filled
with zero
CACHE           51657                   yes             Cache pages
CACHE_PRIVATE   5442                    yes             Cache pages +
private
USER            16301                   yes             User process
pages
FREE            77738211                yes             Free pages
KERN_DATA       1333192                 no              Dumpable kernel
data
```

重要

makedumpfile コマンドは **pages** にレポートを出します。つまり、カーネルページサイズ (Red Hat Enterprise Linux の場合は 4 キロバイトまたは 4096 バイト) に対して使用中のメモリーサイズを算出する必要があります。

https://access.redhat.com/articles/1351013

1.8. KDUMP に関連する PORTAL LABS

Portal Labs は簡単な Web アプリケーションで、システム管理者がさまざまなシステムタスクを実施するのに役立ちます。kdump に関しては現在、Kdump Helper および Kernel Oops Analyzer の 2 つの Labs があります。

1.8.1. Kdump Helper

Kdump Helper は、一連の質問および操作で構成され、**kdump** 設定ファイルの準備を支援します。

Lab のワークフローには、クラスター化された環境およびスタンドアロン環境の両方に関する手順が含まれています。

1.8.2. Kernel Oops Analyzer

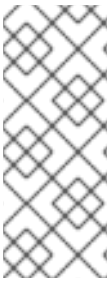
Kernel Oops Analyzer ツールを使うと、Oops メッセージを処理して、クラッシュダンプスタックを解析することなく既知のソリューションがあるかどうかを確認することができます。

Kernel Oops Analyzer では **makedumpfile** からの情報を使用し、クラッシュしたマシンの Oops メッセージとナレッジベースに蓄積された既知の問題を比較します。これにより、予期せぬ障害が発生しても、システム管理者はすばやく既知の問題の可能性を除外して、詳細な解析を依頼するためにサポートチケットを発行することができます。

第2章 カーネルモジュールでの作業

本章では、以下について説明します。

- カーネルモジュールの概要
- **kmod** ユーティリティーを使用してモジュールとその依存関係を管理する方法
- 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制御する方法
- 起動時にモジュールを読み込む方法



注記

本章で説明するカーネルモジュールのユーティリティーを使用するには、最初に root で以下を実行して、ご使用のシステムに **kmod** 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 kmod
```

2.1. カーネルモジュールの概要

Linux カーネルは、モノリシック型として設計されていますが、各ユースケースで必要とされる追加またはオプションのモジュールでコンパイルされています。つまり、動的に読み込まれる **カーネルモジュール** を使用してカーネル機能を拡張することができます。カーネルモジュールでは、以下を提供することができます。

- 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
- **GFS2** または **NFS** などのファイルシステムのサポート

デフォルトのパラメーターは多くの場合、十分に機能しますが、カーネルと同様にモジュールはパラメーターを指定して動作をカスタマイズすることができます。ユーザー空間ツールは、実行中のカーネルに現在ロードされているモジュールを一覧表示することができます。また、利用可能なパラメーターに使用できる全モジュールやモジュール固有の情報をクエリーできます。さらには、実行中のカーネルに対してモジュールを動的にロード/アンロード (削除) することも可能です。**kmod** パッケージにより提供される、このようなユーティリティーの多くは、動作の実行時にモジュールの依存関係を考慮するため、手動による依存関係の追跡が必要になることはほぼありません。

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。

2.2. 現在ロード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。以下に例を示します。

```
# lsmod
Module                  Size  Used by
tcp_lp                 12663  0
bnep                   19704  2
bluetooth             372662  7 bnep
```

```
rfkill                 26536  3 bluetooth
fuse                   87661  3
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_nat_ipv6            13279  1 ip6table_nat
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
[output truncated]
```

lsmod 出力では、3 つのコラムを表示します。

- **Module (モジュール)**
  - メモリーに現在読み込まれているカーネルモジュールの名前
- **Size (サイズ)**
  - カーネルモジュールが使用するメモリー量 (キロバイト単位)
- **Used by (使用フィールド)**
  - 依存関係に関する情報 2 項目を含むフィールド
    - **Module** フィールドにある依存関係の数を表す小数点
    - 依存する **Module** の名前をコンマ区切りにした文字列。この一覧を使用して、アンロードするモジュールに依存するモジュールをすべて先に、アンロードすることができます。

最後に、**lsmod** 出力は **/proc/modules** 擬似ファイルの内容ほど詳細ではないので、はるかに読み取りやすくなっている点に留意してください。

2.3. モジュール情報の表示

カーネルモジュールに関する詳しい情報は、**modinfo module_name** コマンドを使用して表示できます。

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指定する場合には、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。

例2.1 lsmod を使用したカーネルモジュール情報の一覧表示

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、**root** で以下のコマンドを入力します。

```
# 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,
```

第3章 手動のカーネルアップグレード

Red Hat Enterprise Linux カーネルは、サポートしているハードウェアとの整合性と互換性を保てるように、Red Hat Enterprise Linux カーネルチームがカスタムを構築します。Red Hat は、一連の厳格な品質保証テストに合格してから、カーネルをリリースしています。

Red Hat Enterprise Linux カーネルは、**Yum** または **PackageKit** パッケージマネージャーを使用し、簡単にアップグレードして検証できるように、RPM 形式でパッケージされます。**PackageKit** は自動的に Red Hat コンテンツ配信ネットワークサーバーをクエリーし、カーネルパッケージなど利用可能な更新が含まれれるパッケージを通知します。

本章は、**yum** ではなく、**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 **だけ** 有用です。

警告

できる限り **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。理由は、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出てきます。

警告

Red Hat では、カスタムのカーネルは **サポートされていません**。ただし、ナレッジベースドキュメント https://access.redhat.com/solutions/25039 でガイダンスを確認できます。

yum でカーネルパッケージをインストールする方法は、https://access.redhat.com/documentation/ja-JP/Red_Hat_Enterprise_Linux/7/html/System_Administrators_Guide/ch-yum.html#sec-Updating_Packages を参照してください。

Red Hat コンテンツ配信ネットワークに関する情報は、https://access.redhat.com/documentation/ja-JP/Red_Hat_Enterprise_Linux/7/html/System_Administrators_Guide/chap-Subscription_and_Support-Registering_a_System_and_Managing_Subscriptions.html を参照してください。

3.1. 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**: シングルコア、マルチコア、マルチプロセッサーシステム用のカーネルが含まれます。
- **kernel-debug**: カーネル診断ができるように複数のデバッグオプションが有効になっているカーネルが含まれます。デバッグオプションが有効になっているとパフォーマンスが低下します。

- **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**: カーネル診断ができるように複数のデバッグオプションが有効になっている開発バージョンのカーネルが含まれます。デバッグオプションが有効になっているとパフォーマンスが低下します。
- **kernel-doc**: カーネルソースからのドキュメントファイルです。これらのファイルには、同梱で配布される Linux カーネルとデバイスドライバーのさまざまな部分が文書化されています。このパッケージをインストールすると、オプションへの参照が提供され、読み込み時に Linux カーネルモジュールに渡すことができます。
  デフォルトでは、これらのファイルは **/usr/share/doc/kernel-doc-kernel_version/** ディレクトリーに配置されています。
- **kernel-headers**: Linux カーネルと、ユーザー空間ライブラリーおよびプログラムとの間のインターフェースを指定する C ヘッダーファイルが含まれます。このヘッダーファイルはほとんどの標準プログラムを構築するの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実施を支援する **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。

3.2. 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、予防的な前準備手順の実行をお薦めします。

まず、問題が発生した場合に備えて、機能するブートメディアがシステムにあることを確認します。ブートローダーで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には、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** または **キー** などと呼ばれるフラッシュデバイスの形式、または、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。ただし、**ext2**、**ext3**、**ext4** 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、または最低限ブートメディア (minimal boot media) イメージを 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。約 4 GB がディストリビューション DVD イメージ用に必要で、約 700 MB がディストリビューション CD イメージ用に、そして約 10 MB が最低限ブートメディアイメージ用に必要です。

Red Hat Enterprise Linux のインストール DVD、またはインストール CD-ROM#1 からの **boot.iso** ファイルのコピーと、約 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。

USB ストレージデバイスの使用に関する詳しい情報は、https://access.redhat.com/solutions/624423「How to format a USB key」 と https://access.redhat.com/solutions/39373「How to manually mount a USB flash drive in a non-graphical environment 」 を確認してください。

以下の手順では、コピー先のファイルと同じパス名を指定しなければ、USB ストレージデバイス上にある既存のファイルは影響を受けません。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
```

   sdX にはデバイス名を指定します。

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. **ISOLINUX** ファイルを **boot.iso** から USB ストレージデバイスにコピーします。

```
# 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、フロッピ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** の 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。

3.3. アップグレードされたカーネルのダウンロード

システム用に更新されたカーネルが利用可能かを判定する手段は数種類あります。

- 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 コンテンツ配信ネットワーク: Red Hat コンテンツ配信ネットワークにサブスクライブするシステムについては、**yum** パッケージマネージャーが最新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できます。**Dracut** ユーティリティーは、必要な場合は初期 RAM ディスクイメージを作成し、新規カーネルを起動するためにブートローダーを設定します。Red Hat コンテンツ配信ネットワークからパッケージをインストールする方法についての詳細は、https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/System_Administrators_Guide/ch-yum.html を参照してください。システムを Red Hat コンテンツ配信ネットワークにサブスクライブする方法についての詳細は、https://access.redhat.com/documentation/ja-JP/Red_Hat_Enterprise_Linux/7/html/System_Administrators_Guide/chap-Subscription_and_Support-Registering_a_System_and_Managing_Subscriptions.html を参照してください。

yum が Red Hat Network からの更新されたカーネルのダウンロードとインストールに使用される場合は、「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」 の指示にのみしたがってください。カーネルはデフォルトで起動するように **変更しないでください**。Red Hat Network がデフォルトのカーネルを最新バージョンに自動的に変更します。カーネルを手動でインストールするには、「アップグレードの実行」 に進みます。

3.4. 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

新しいカーネルに問題がある場合を考え、古いカーネルの維持を強く推奨します。

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、ブートローダーの問題を起こすので、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。

3.5. 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仕事は、IDE、SCSI、または RAID などのブロックデバイスモジュールをプ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 Enteprise 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する場合は常に **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。

/etc/sysctl.conf ファイルまたは別の **sysctl** 設定ファイルを変更してカーネル属性を変更し、変更した設定がブートプロセスの初期段階で使用される場合には、**dracut -f** コマンドを使用して、初期 RAM ディスクイメージの再構築が必要な場合があります。たとえば、ネットワーク関連の変更を受けて、ネットワークにアタッチされたストレージから起動する場合などです。

IBM eServer System i (「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では、**dracut** コマンドの実行により **initramfs** を作成できますが、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 配布の 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、それが **grub.cfg** 設定ファイル内で正しく指定されているかを検証するには、以下の手順にしたがいます。

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3.1 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3.1「カーネルと initramfs バージョンの一致を確認」 は以下を示します。

   - 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** ディレクトリーにあります)。
   - 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 ディレクトリーで、複数の **initramfs-kernel_versionkdump.img** ファイルが見つかる場合があります。それらは、カーネルのデバッグ目的で **Kdump** メカニズムで作成される特殊ファイルであり、システムの起動には使用されず、無視しても問題ありません。**kdump** の詳細は、Red Hat Enterprise Linux 7 カーネルクラッシュダンプガイド を参照してください。

2. 使用している **initramfs-kernel_version** ファイルが、**/boot** ディレクトリーにある最新カーネルのバージョンと一致しない場合、または他の特定の状況では、**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場合があります。**root** としてオプションなしで **dracut** を呼び出すと、それが **/boot** 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
```

dracut が既存の **initramfs** を上書きするには (たとえば、 **initramfs** が破損している場合など)、**-f**、**--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
          Does 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)
```

プレロードするカーネルモジュールを指定する必要がある場合には、**/etc/dracut.conf** 設定ファイルの **add_dracutmodules+="module more_modules "** ディレクティブの括弧の中に (**.ko** などの任意のファイル名のサフィックスを取り除いて) 対象のモジュール名を追加します。dracut で作成した **initramfs** イメージファイルのファイルコンテンツを一覧表示するには、**lsinitrd initramfs_file** コマンドを使用します。

```
# 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
[output truncated]
```

オプションと用途に関する詳しい情報は **man dracut** と **man dracut.conf** を参照してください。

3. **/boot/grub2/grub.cfg** 設定ファイルを検査して、起動中のカーネルバージョンについて **initramfs-kernel_version.img** ファイルが存在することを確認します。以下が例になります。

```
# grep initramfs /boot/grub2/grub.cfg
initrd16 /initramfs-3.10.0-123.el7.x86_64.img
initrd16 /initramfs-0-rescue-6d547dbfd01c46f6a4c1baa8c4743f57.img
```

詳しい情報は 「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 配布の 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るので、手動で実行する必要はありません。このファイルが作成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て **/boot/vmlinitrd-kernel_version** ファイルがすでに存在することを確認します。

```
# ls -l /boot/
```

kernel_version は、先程インストールしたカーネルバージョンと一致する必要があります。

初期 RAM ディスクイメージへの変更を戻す方法

たとえば、システムの設定を間違えたことで起動しなくなったような場合は、以下の手順にしたがって初期 RAM ディスクイメージに加えた変更を戻す必要があります。

初期 RAM ディスクイメージへの変更を戻す方法

1. GRUB メニューでレスキューカーネルを選択してシステムを再起動します。
2. **initramfs** の誤動作を引き起こしている間違った設定を変更します。
3. root で以下のコマンドを実行して、正しい設定で **initramfs** を作成し直します。

```
# dracut --kver kernel_version --force
```

上記の手順は、**sysctl.conf** ファイルで **vm.nr_hugepages** を間違って設定してしまった場合などに便利です。**sysctl.conf** ファイルは **initramfs** に含まれているため、新たな **vm.nr_hugepages** 設定は **initramfs** で適用されてしまい、**initramfs** が再構築されてしまいます。ただし、設定が間違っているので、新規の **initramfs** は破損しており、新規に構築されるカーネルは起動しないため、上記の手順を使用した設定の修正が必要になります。

初期 RAM ディスクイメージのコンテンツの一覧表示

initramfs に含まれるファイルを一覧表示するには、root で以下のコマンドを実行します。

```
# lsinitrd
```

/etc ディレクトリーにあるファイルだけを表示するには、以下のコマンドを使用します。

```
# lsinitrd | grep etc/
```

現行カーネルの **initramfs** に保存されている特定ファイルのコンテンツを出力するには、**-f** オプションを使用します。

```
# lsinitrd -f filename
```

たとえば、**sysctl.conf** のコンテンツを出力するには、以下のコマンドを実行します。

```
# lsinitrd -f /etc/sysctl.conf
```

カーネルのバージョンを指定するには、**--kver** オプションを使用します。

```
# lsinitrd --kver kernel_version -f /etc/sysctl.conf
```

たとえば、カーネルバージョン 3.10.0-327.10.1.el7.x86_64 についての情報を一覧表示するには、以下のコマンドを使用します。

```
# lsinitrd --kver 3.10.0-327.10.1.el7.x86_64 -f /etc/sysctl.conf
```

3.6. ブートローダーの検証

yum コマンドまたは **rpm** コマンドで、カーネルをインストールしてください。

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。

いずれのコマンドも、**/etc/sysconfig/kernel** 設定ファイルに以下の設定を含める場合にのみ、新しいカーネルがデフォルトのカーネルとして起動するよう設定することに留意してください。

```
DEFAULTKERNEL=kernel
UPDATEDEFAULT=yes
```

DEFAULTKERNEL オプションは、デフォルトのカーネルパッケージタイプを指定します。**UPDATEDEFAULT** オプションは、新規カーネルパッケージがデフォルトで新しいカーネルにするかを指定します。

第4章 KPATCH の使用

4.1. KPATCH とは?

kpatch は、ライブのカーネルパッチソリューションで、プロセスをリブートまたは再起動することなしに、実行中のカーネルにパッチを当てることができます。kpatch は、長時間のタスク完了、ユーザーのログオフ、計画ダウンタイムなどを待機せずに、システム管理者が重要なセキュリティーパッチをカーネルに即座に適用することができます。セキュリティーや安定性を犠牲にすることなく、システムの稼働時間中に制御できるようになります。

警告

kpatch と他のサブコンポーネントで、互換性がない場合があります。「制限」 セクションを注意深く読んでから、**kpatch** を使用するようにしてください。

4.2. KPATCH のサポート範囲

- **kpatch** を使用したライブでのカーネルパッチの適用は Red Hat Enterprise Linux 7.2 以降でサポートされています。
- ライブでのカーネルパッチの適用については、Premium SLA サブスクリプションをご契約のお客様を対象としてサポートしています。
- ライブでのカーネルパッチ適用は、アクティブな Red Hat Enterprise Linux 7 メンテナンスストリーム (現在の非同期エラータフェーズに含まれる) で **のみ** サポートされます。現在のサポートフェーズに関する情報は、「Red Hat Enterprise Linux ライフサイクル」 を参照してください。
- ライブでのカーネルパッチ適用は、Extended Update Support リリースでは **入手できません**。
- **kpatch** は、Red Hat Enterprise Linux Realtime (RT) カーネルではサポート **されていません**。
- **kpatch** は、Red Hat Enterprise Linux 5 または Red Hat Enterprise Linux 6 ではサポート **されていません**。
- カーネルに、ライブでカーネルパッチを適用できるのは、一度に **1 つ** のみです。
- ハードウェアイネーブルメントなど、すべての問題がライブでのカーネルパッチで対応されているかは不明です。

4.3. アクセスおよび配信

ライブのカーネルパッチ機能は、RPM パッケージとして提供されるカーネルモジュール (kmod) で実装されます。**kpatch** ユーティリティーは、ライブカーネルパッチ用のカーネルモジュールをインストールおよび削除するのに使用します。

Premium サブスクリプションを契約されているお客様は、Red Hat サポートが提供する迅速化された修正ソリューションの一部として、ライブのカーネルパッチをリクエストしていただけます。

Premium サブスクリプションを契約されているお客様で、再起動を必要とする **hotfix** カーネルを通常使用する場合は、ダウンタイムを必要としない **kpatch** kmod をリクエストすることが可能です。**kpatch** パッチは、**hotfix** カーネルと同じ方法で、修正が含まれるエラータがリリースされてから 30 日間サポートされます。

お客様は、Red Hat カスタマーポータル から直接サポートケースを起票して、適切かつ迅速な修正オプションについて相談していただけます。

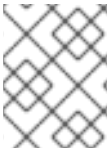
4.4. 制限

kpatch は、汎用のカーネルアップグレードメカニズムではありません。システムをすぐに再起動できない場合など、単純なセキュリティーおよびバグ修正の更新を適用する場合に使用します。

パッチを読み込み中または読み込み後に、**SystemTap** または **kprobe** を使用 **しないでください**。パッチは、プローブが削除されるまで、適用されない可能性があります。

cat /sys/kernel/debug/tracing/trace を実行して **ftrace** 出力ファイルに直接アクセス **しないでください**。代わりに、**trace-cmd** ユーティリティーがサポートされています。

kpatch を使用時には、システムを一時停止またはハイバネート **しないようにしてください**。一時停止またはハイバネートすると、短期間パッチが一時的に無効になってしまいます。



注記

Red Hat は、今後のリリースに向けて積極的に **kpatch** の制約の多くをなくすように努めています。

4.5. KPATCH の有効化および使用方法

kpatch のコンポーネントは以下の通りです。

kpatch のコンポーネント

systemd の統合ポイント

kpatch.service と呼ばれる **systemd** サービス。kpatch モジュールを起動時に読み込む **multiuser.target** で必要です。

パッチモジュール

- 新しいカーネルコードの配信メカニズム
- これは、適用される **kpatch** と一致するように名前がつけられた別のカーネルモジュールです。
- パッチモジュールには、カーネルに対する最新のホットフィックスからコンパイルされたコードが含まれます。
- パッチモジュールは、コアモジュール **kpatch.ko** で登録し、置き換えられるオリジナル機能の情報を提供します。また、置換される機能に一致するポインターも含まれます。

kpatch ユーティリティー

パッチモジュールを管理するためのコマンドラインツール

4.5.1. kpatch ツールのインストール

kpatch モジュールをインストールする前に、**kpatch** ツールパッケージをインストールする必要があります。シェルプロンプトで **root** として以下を入力し、インストールします。

```
# yum install kpatch
```

4.5.2. kpatch ホットフィックスのインストール

kpatch ホットフィックスをインストールするには、**yum** で提示の **kpatch-patch** RPM パッケージをインストールします。たとえば、**kpatch-patch-7.0-1.el7.x86_64.rpm** するには、**root** として以下のコマンドを実行します。

```
# yum install kpatch-patch-7.0-1.el7.x86_64.rpm
```

4.5.3. インストールされた kpatch ホットフィックスの一覧表示

パッチが読み込まれ、インストールされていることを確認するには、**root** として **kpatch list** コマンドを実行します。

```
# kpatch list
Loaded patch modules:
kpatch_7_0_1_el7
Installed patch modules:
kpatch-7-0-1-el7.ko (3.10.0-121.el7.x86_64)
kpatch-7-0-1-el7.ko (3.10.0-123.el7.x86_64)
```

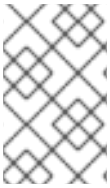
上記の出力では、モジュールがカーネルに読み込まれていることがわかります。つまり、カーネルは、**kpatch-patch-7.0-1.el7.x86_64.rpm** パッケージに含まれる最新のホットフィックスが適用されたことになります。また、今後カーネルバージョン 3.10.0-121 と 3.10.0-123 の再起動を行う際に、**systemd** が読み込めるように **/var/lib/kpatch** に保存されたことがわかります。

4.5.4. kpatch ホットフィックスの更新

新規バージョンの **kpatch-patch** RPM パッケージが後にリリースされた場合には、**yum** で、適用したパッチをアップグレードしてください。たとえば、**kpatch-patch-7.0-2.el7.x86_64.rpm** にアップグレードするには、**root** として以下を実行します。

```
# yum update kpatch-patch-7.0-2.el7.x86_64.rpm
```

RPM パッケージをアップグレードすると、実行中のカーネルのパッチモジュールが自動的に置き換えられ、再起動時に **systemd** によって使用される **/var/lib/kpatch** 構造が更新されます。



注記

kpatch-patch RPM パッケージのパッチモジュールは累積パッチです。したがって、kpatch-patch-7.0-2 を利用できる場合は、kpatch-patch-7.0-1 をインストールせずに代わりに kpatch-patch-7.0-2 をインストールすることもできます。

パッチモジュールをロードすると、**TAINT_LIVEPATCH** カーネル taint フラグ (ビット 15に相当)、**TAINT_OOT_MODULE** (ビット 12 に相当)、および **TAINT_TECH_PREVIEW** (ビット 29 に相当) が

設定されます。 カーネルにパッチが当てられているかどうかを確認するには、**cat** コマンドを実行して **/proc/sys/kernel/tainted** の内容を表示し、ファイルの値を確認します。別の taint フラグが設定されている場合を除き、カーネルにパッチが当たっている場合は、値は **536907776 (0x20009000)** です。

4.5.5. kpatch ホットフィックスの削除

実行中のカーネルから **kpatch** モジュールをアンロードするには、**kpatch unload** コマンドを使用し、パッチモジュールの名前を指定します。たとえば、**kpatch_7_0_2_el7** をアンロードするには、シェルプロンプトから **root** 権限で次のコマンドを実行します。

```
# kpatch unload kpatch_7_0_2_el7
```

また、以下のように **kpatch** アンインストール コマンドを実行して、**/var/lib/kpatch** からパッチモジュールをアンインストールする必要があります。

```
# kpatch uninstall kpatch_7_0_2_el7
```

このコマンドは、デフォルトで現バージョンのカーネルに相当するカーネルから **kpatch** をアンインストールしますが、**kernel-version** オプションを使用すれば別のバージョンのカーネルを指定することもできます。

```
# kpatch uninstall --kernel-version 3.10.0-121.el7.x86_64 kpatch_7_0_2_el7
```

あるいは、**kpatch-patch** RPM パッケージをアンインストールすることができます。これにより、**/var/lib/kpatch** からパッチモジュールも削除されます。

注記

RPM パッケージをアンインストールしても、カーネルから **kpatch** モジュールはアンロードしません。上に示すように、**kpatch unload** を明示的に呼び出す必要があります。

4.6. KPATCH はどのように機能しますか?

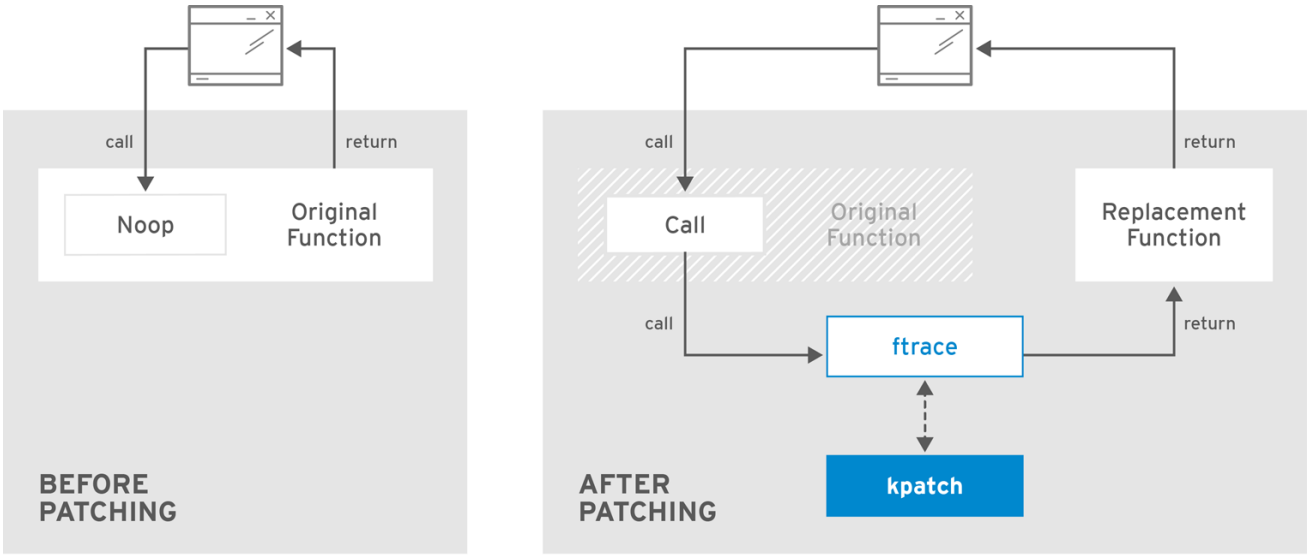
kpatch ユーティリティーは、**ftrace** を使用して、カーネル機能へのポインターの任意の再マッピングを有効にします。カーネルのライブパッチをシステムに適用すると、以下が発生します。

1. モジュールで新たにコンパイルしたコードが **/var/lib/kpatch** にコピーされ、次回の起動時に **systemd** からカーネルへの再適用が登録されます。

2. 実行中のカーネルに **kpatch** モジュールがロードされ、新しいコードのメモリー内の場所を指定するポインターを使用して、新しい機能が **ftrace** メカニズムに登録されます。

3. パッチが当てられた機能にカーネルがアクセスすると、**ftrace** メカニズムにリダイレクトされます。これにより、元々の機能は回避され、パッチを当てたバージョンの機能にカーネルをリダイレクトします。

図4.1 kpatch の仕組み



4.7. サードパーティーのライブパッチソリューションはサポートされますか?

カーネルのライブパッチを提供するサードパーティーまたはプロプライエタリーのツールをいつくか利用できますが、Red Hat がサポートするのは **kpatch** と、Red Hat サポート契約から提供される RPM モジュールに限定されます。Red Hat は、サードパーティーのライブパッチにサポートを提供しませんが、エンジニアリングと公式な Red Hat **kpatch** への依頼は常に受け付けています。

サードパーティーのライブパッチに対する任意の Red Hat のレビューについては、以下の基準を満たしているかどうかを判定するため、ソースコードの提供が必要です。

1. 異常発生時にカーネルが経験したものと同じサブシステムとコードパスに影響します。

2. 該当するストリーム内にて、サポートしている方法で同じパッチを適用すると失敗しません。

根本原因の特定が必要なため、調査の開始時にカーネルのライブパッチのベンダーと話すことを Red Hat では推奨しています。これにより、ベンダーの許可が得られた場合はソースコードを提供できるようになります。また、Red Hat サポートが調査に乗り出す前に、ベンダーのサポート組織が原因特定のために協力できることになります。

サードパーティーのカーネルのライブパッチでシステムを実行している場合は、Red Hat が提供し、サポートしているソフトウェアでこの問題を再現するよう依頼する権利を Red Hat は保有しています。再現できない場合は、同じ挙動が見られるかどうかを確認するために、ライブパッチが適用されていないテスト環境にデプロイされたのと同様のシステムとワークロードが必要になります。

サードパーティーソフトウェアのサポートポリシーに関する情報は、https://access.redhat.com/articles/1067 のナレッジベースにある 「Red Hat グローバルサポートサービスは、サードパーティーのソフトウェア、ドライバー、そして認定されていないハードウェアおよびハイパーバイザー、もしくはゲストのオペレーティングシステムについてどのようなサポートを提供していますか?」 を参照してください。

第5章 SYSCTL およびチューニング可能なカーネルの使用

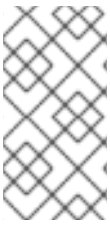
5.1. チューニング可能なカーネルの概要

チューニング可能なカーネルを使用することで、Red Hat Enterprise Linux の起動時に、またはシステムを実行中にオンデマンドで動作をカスタマイズできます。ハードウェアのパラメーターによっては、起動時のみに指定され、システムが実行し始めると変更できないものもありますが、ほとんどの場合は必要に応じて変更でき、次の起動に向けてパーマネント設定ができます。

5.2. チューニング可能なカーネルの使用方法

チューニング可能なカーネルを修正するには、以下の 3 つの方法があります。

1. **sysctl** コマンドを使用する方法
2. **/etc/sysctl.d/** ディレクトリーで設定ファイルを手動で修正する方法
3. シェルを経由して、**/proc/sys** にマウントされた仮想ファイルと相互作用する方法



注記

すべての起動時パラメーターが sysfs サブシステムの制御下にあるわけではありません。いくつかのハードウェア固有のオプションは、カーネルのコマンドライン上に設定する必要がありますが、これらのオプションについては、本ガイドの Kernel Parameters セクションを参照してください。

5.2.1. sysctl コマンドの使用

sysctl コマンドを使用して、チューニング可能なカーネルを表示、読み取り、および設定します。表示または読み取りの際のチューニング可能なカーネルのフィルター、そしてチューニング可能なカーネルの一時的または永続的な設定が可能となります。

1. 変数の表示

```
# sysctl -a
```

2. 変数の読み取り

```
# sysctl kernel.version +
kernel.version = #1 SMP Fri Jan 19 13:19:54 UTC 2018
```

3. 変数の一時的な書き込み

```
# sysctl <tunable class>.<tunable>=<value>
```

4. 変数の永続的な書き込み

```
# sysctl -w <tunable class>.<tunable>=<value>
```

5.2.2. /etc/sysctl.d 内のファイルの変更

起動時にデフォルトをオーバーライドするには、手動で **/etc/sysctl.d** にファイルを追加することも可能です。

1. **/etc/sysctl.d** に新しいファイルを作成します。

```
# vim /etc/sysctl.d/99-custom.conf
```

2. 以下の形式で、設定したい変数を 1 行に 1 つずつ入れていきます。

```
<tunable class>.<tunable> = <value> +
<tunable class>.<tunable> = <value>
```

3. ファイルを保存します。

4. 変更を有効にするためにマシンを再起動するか、
   または
   sysctl -p /etc/sysctl.d/99-custom.conf を実行して再起動せずに変更を適用します。

5.3. 制御可能なチューニング可能なもの

チューニング可能なものは、カーネルのサブシステムによっていくつかのグループに分けられます。Red Hat Enterprise Linux システムにおけるチューニング可能なものには、以下のクラスがあります。

表5.1 sysctl インターフェースの表

クラス	サブシステム
abi	実行ドメインおよびパーソナリティー
crypto	暗号化インターフェース
debug	カーネルのデバッグインターフェース
dev	デバイスの詳細
fs	グローバルおよび固有のチューニング可能なファイルシステム
kernel	グローバルなチューニング可能なカーネル
net	チューニング可能なネットワーク
sunrpc	Sun Remote Procedure Call (NFS)
user	ユーザー名前空間の制限
vm	メモリー、バッファー、およびキャッシュのチューニングと管理

5.3.1. チューニング可能なネットワークインターフェース

システム管理者は、チューニング可能なネットワークを経由して実行中のシステムのネットワーク設定を変更できます。

チューニング可能なネットワークは、**/proc/sys/net** ディレクトリーにあります。このディレクトリーには、さまざまなネットワークトピックに関する複数のサブディレクトリーが格納されています。システム管理者はこれらのサブディレクトリー内のファイルを変更し、ネットワーク設定を調整する必要があります。

以下は、最も頻繁に使用されるディレクトリーです。

1. **/proc/sys/net/core/**

2. **/proc/sys/net/ipv4/**

/proc/sys/net/core/ ディレクトリーには、カーネルとネットワーク層との間の相互作用を制御するさまざまな設定が格納されています。これらのチューニング可能なものをいくつか調整することで、システムのパフォーマンスを向上させることが可能です。たとえば、受信キューのサイズを増加したり、ネットワークインターフェース専用の接続およびメモリーの最大値を上げたりすることで、パフォーマンスを向上させることができます。システムのパフォーマンスは、個々の問題のさまざまな側面に左右されることに留意してください。

/proc/sys/net/ipv4/ ディレクトリーには、追加のネットワーク設定が格納されています。これは、システムに対する攻撃を防止したり、システムをルーターとして機能させるように使用している場合に役立ちます。ディレクトリーには、IP および TCP の両方の変数が格納されています。これらの変数に関する詳細は **/usr/share/doc/kernel-doc-<version>/Documentation/networking/ip-sysctl.txt** を参照してください。

/proc/sys/net/ipv4/ ディレクトリー内のその他のディレクトリーは、ネットワークスタックのさまざまな側面を扱っています。

1. **/proc/sys/net/ipv4/conf/** - 各システムインターフェースを異なる方法で設定できるようにします。これには、未設定のデバイス用のデフォルト設定の使用や、すべての特殊設定をオーバーライドする設定が含まれます。

2. **/proc/sys/net/ipv4/neigh/** - システムに直接接続されたホストとの通信のための設定や、1 ステップ以上離れたシステム用の異なる設定も含まれています。

3. **/proc/sys/net/ipv4/route/** - システム上の任意のインターフェースとのルーティングに適用される指定値が格納されています。

チューニング可能なネットワークのリストは IPv4 インターフェースと関係があり、**/proc/sys/net/ipv4/{all,<interface_name>}/** ディレクトリーからアクセスできます。

log_martians

カーネルログに無効なアドレスを持つパケットをログします。

タイプ	デフォルト
ブール値	0

1 つ以上の **conf/{all,interface}/log_martians** が TRUE に設定されている場合に有効です。

Further Resources

- 「カーネルパラメーター ipv4.conf.all.log_martians はどんなことに使用されますか?」
- 「メッセージファイルに "martian source" と記録される理由」

accept_redirects

ICMP のリダイレクトメッセージを受信します。

タイプ	デフォルト
ブール値	1

インターフェース用の **accept_redirects** が、以下の条件下で有効となります。

- **conf/{all,interface}/accept_redirects** が両方とも TRUE の場合 (インターフェースへの転送が有効な場合)
- **conf/{all,interface}/accept_redirects** のうち少なくとも 1 つが TRUE の場合 (インターフェースへの転送は無効)

Further Resources

- 「ICMP リダイレクトを無効にする方法」

転送

インターフェース上で IP 転送を有効にします。

タイプ	デフォルト
ブール値	0

Further Resources

- 「パケット転送および非ローカルバインディングの有効化」

mc_forwarding

マルチキャストルーティングを実施します。

タイプ	デフォルト
ブール値	0

- 読み取り専用値
- マルチキャストルーティングのデーモンが必要です。
- インターフェース用にマルチキャストルーティングを有効にするためには **conf/all/mc_forwarding** も TRUE に設定する必要があります。

Further Resources

- 読み取り専用の動作に関する詳細は、「Why system reports "permission denied on key" while setting the kernel parameter "net.ipv4.conf.all.mc_forwarding"?」 を参照してください。

medium_id

接続しているメディアがデバイスを区別するために使用する任意の値です。

タイプ	デフォルト
整数	0

注記

- 同じメディア上の 2 つのデバイスにおいて、このうちの 1 つでしかブロードキャストパケットが 受信されない場合、2 つのデバイスの id 値が異る可能性があります。
- デフォルト値の 0 は、メディアに対してそのデバイスが唯一のインターフェースであることを意味します。
- 値が -1 の場合、メディアが不明であることを示します。
- 現在は、proxy_arp の動作を変更するために使用されています。
- proxy_arp の機能は、異るメディアに接続された 2 つのデバイスの間で転送されたパケット用に有効となります。

Further Resources - たとえば、「Using the "medium_id" feature in Linux 2.2 and 2.4」 を参照してください。

proxy_arp

Proxy ARP の実行

タイプ	デフォルト
ブール値	0

conf/{all,interface}/proxy_arp のうち少なくとも 1 つが TRUE に設定されている場合、インターフェース用の **proxy_arp** は有効です。そうでない場合は無効です。

proxy_arp_pvlan

プライベート VLAN Proxy ARP

タイプ	デフォルト
ブール値	0

RFC 3069 などの機能をサポートするために、Proxy ARP が同じインターフェースに応答できるようにします。

shared_media

共有メディアのリダイレクト RFC1620 を送信 (ルーター) または受信 (ホスト) します。

タイプ	デフォルト
ブール値	1

注記

- secure_redirects をオーバーライドします。
- **conf/{all,interface}/shared_media** のうち少なくとも 1 つが TRUE に設定されている場合、インターフェース用の **shared_media** が有効となります。

secure_redirects

インターフェースの現在のゲートウェイリストにリストされているゲートウェイに対する ICMP リダイレクトメッセージのみを受信します。

タイプ	デフォルト
ブール値	1

注記

- 無効となった場合でも、RFC1122 リダイレクトルールが引き続き適用されます。
- shared_media によってオーバーライドされます。
- **conf/{all,interface}/secure_redirects** のうち少なくとも 1 つが TRUE に設定されている場合、インターフェース用の secure_redirects は有効となります。

send_redirects

ルーターの場合、リダイレクトを送信します。

タイプ	デフォルト
ブール値	1

注記
conf/{all,interface}/send_redirects のうち少なくとも 1 つが TRUE に設定されている場合、インターフェース用の **send_redirects** が有効となります。

bootp_relay

ローカル用としてこのホストに予定されていないソースアドレス 0.b.c.d のパケットを受信します。

タイプ	デフォルト
ブール値	0

注記

- これらのパケットを管理するために BOOTP デーモンを有効にする必要があります。
- インターフェース用に BOOTP リレーを有効にするために **conf/all/bootp_relay** も TRUE に設定する必要があります。
- 実装されていない場合は、『Red Hat Enterprise Linux ネットワークガイド』の 「DHCP リレーエージェント」 を参照してください。

accept_source_route

SRR オプションのあるパケットを受信します。

タイプ	デフォルト
ブール値	1

注記

- インターフェース上の SRR オプションのあるパケットを受信するために、**conf/all/accept_source_route** も TRUE に設定する必要があります。

accept_local

ローカルソースアドレスのあるパケットを受信します。

タイプ	デフォルト
ブール値	0

注記

- これは、適切なルーティングとのコンビネーションにて、2 つのローカルインターフェース間のパケットをワイヤー上で移動し、適切に受信させるために使用することができます。
- accept_local に効果をもたらすためには、**rp_filter** をゼロ以外の値に設定する必要があります。

route_localnet

ルーティング中は、ループバックアドレスを martian ソースまたは宛先として考慮しません。

タイプ	デフォルト
ブール値	0

注記

- これにより、ローカルルーティング目的での **127/8** の使用が有効となります。

rp_filter

ソースの検証を有効化

タイプ	デフォルト
整数	0

値	効果
0	ソースの検証はありません。
1	RFC3704 厳密な逆方向パスで定義された厳密モード
2	RFC3704 緩やかな逆方向パスで定義された緩やかなモード

注記

- RFC3704 における現在の推奨プラクティスは、DDos 攻撃による IP スプーフィングを回避するために厳密モードを有効にすることです。
- 非対称のルーティングまたは別の複雑なルーティングを使用する場合は、緩やかなモードが推奨されます。
- {interface} でソースの検証を行う際、**conf/{all,interface}/rp_filter** の中の最大値が使用されます。

arp_filter

タイプ	デフォルト
ブール値	0

値	効果
0	(デフォルト) カーネルは、別のインターフェースからのアドレスの ARP 要求に対応できます。正常な通信の可能性を向上させるので、通常は理にかなっています。
1	同じサブネットで複数のネットワークインターフェースを持つことを可能にします。また、カーネルがインターフェースから ARP 要求 の IP パケットをルーティングするかどうかに基づいて、各インターフェースの ARP が応答できるようにします (したがって、これを機能させるためにはソースベースのルーティングを使用する必要があります)。つまり、ARP 要求に応答するカード (通常は 1) の制御が可能となります。

注記

- IP アドレスは、特定のインターフェースではなく、Linux の完全なホストが所有します。この動作が問題を起こすのは、負荷分散などのより複雑なセットアップの場合だけです。

- **conf/{all,interface}/arp_filter** のうち少なくとも 1 つが TRUE に設定されている場合、インターフェース用の **arp_filter** は有効となります。

arp_announce

インターフェース上に送信された ARP 要求の IP パケットからローカルソースの IP アドレスを発表するための異る制限レベルを定義します。

タイプ	デフォルト
整数	0

値	効果
0	(デフォルト) 任意のインターフェース上に設定された任意のローカルアドレスを使用します。
1	このインターフェースでは、出力先のサブネットにないローカルアドレスは使用しないようにします。このインターフェースを経由してアクセス可能な出力先ホストが 、ARP 要求のソース IP アドレスが受信側インターフェース上に設定されるロジカルなネットワークの一環となるよう要求した場合、このモードは役立ちます。要求を生成する際、出力先 IP を含むすべてのサブネットを確認し、そのようなサブネットからのソースアドレスである場合は保持します。そのようなサブネットがない場合は、レベル 2 のルールにしたがってソースアドレスを選択します。
2	この出力先には常に最適のローカルアドレスを使用します。このモードでは、IP パケットのソースアドレスを無視し、出力先ホストとの対話には好みのローカルアドレスを選択するようにします。このようなローカルアドレスは、出力先 IP アドレスを含む発信インターフェース上のすべてのサブネット上にある主要な IP アドレスを探すことで選択されます。適切なローカルアドレスが見つからない場合は、発信インターフェース上またはその他すべてのインターフェース上にある最初のローカルアドレスを選択します。この時、アナウンスするソース IP アドレスに関係なく、要求に対する応答があることを期待します。

注記

- **conf/{all,interface}/arp_announce** の中の最大値が使用されます。
- 制限レベルを上げると、解決済み出力先から応答がある可能性が高くなり、制限レベルを下げると、より有効な送信者情報をアナウンスします。

arp_ignore

受信した ARP 要求に対して応答するさまざまなモードを定義します。この ARP 要求は、ローカル出力先 IP アドレスを解決するものです。

タイプ	デフォルト
整数	0

値	効果
0	(デフォルト): 任意のインターフェース上に設定された任意のローカル出力先 IP アドレスに応答します。
1	出力先 IP アドレスが受信インターフェース上で設定されたローカルアドレスの場合にのみ応答します。
2	出力先 IP アドレスが受信インターフェース上で設定されたローカルアドレスで、送信者の IP アドレスと出力先 IP アドレスの両方がこのインターフェース上の同じサブネットの一部である場合にのみ応答します。
3	スコープホストで設定されたローカルアドレスには応答しません。グローバルおよびリンク用のアドレス解決のみに応答します。
4-7	予備
8	ローカルアドレスの場合はすべて、応答しません。{interface} 上で ARP 要求が受信された際に conf/{all,interface}/arp_ignore の最大値が使用されます。

注記

arp_notify

アドレスおよびデバイスの変更を通知するモードを定義します。

タイプ	デフォルト
ブール値	0

値	効果
0	何もしません。
1	デバイスの停止またはハードウェアのアドレス変更の際、余計な ARP 要求を生成します。

注記

arp_accept

IP がまだ ARP テーブルに存在しない余計な ARP フレームの動作を定義します。

タイプ	デフォルト
ブール値	0

値	効果
0	ARP テーブルに新しいエントリーを作成しません。
1	ARP テーブルに新しいエントリーを作成します。

注記
この設定がオンの場合、応答および要求の両タイプの余計な ARP が、ARP テーブルをアップデートするようトリガーします。ARP テーブルが余計な ARP フレームの IP アドレスをすでに格納している場合、この設定がオンまたはオフであることに関係なく、ARP テーブルがアップデートされます。

app_solicit

マルチキャストプローブへドロップバックする前にネットリンクを経由してユーザー空間の ARP デーモンに送信するプローブの最大数 (mcast_solicit を参照してください)。

タイプ	デフォルト
整数	0

注記
mcast_solicit を参照してください。

disable_policy

このインターフェースの IPSEC ポリシー (SPD) を無効にします。

タイプ	デフォルト
ブール値	0

needinfo

disable_xfrm

いかなるポリシーであろうと、このインターフェースの IPSEC 暗号化を無効にします。

タイプ	デフォルト
ブール値	0

needinfo

igmpv2_unsolicited_report_interval

次の未承諾の IGMPv1 または IGMPv2 レポートの再送信が実行されるミリ秒単位の間隔。

タイプ	デフォルト
整数	10000

注記
ミリ秒

igmpv3_unsolicited_report_interval

次の未承諾の IGMPv3 レポートの再送信が実行されるミリ秒単位の間隔。

タイプ	デフォルト
整数	1000

注記
ミリ秒

tag

必要に応じて使用可能な数字の書き込みが可能です。

タイプ	デフォルト
整数	0

needinfo

xfrm4_gc_thresh

IPv4 宛先キャッシュエントリー用のガベージコレクションを開始するしきい値。

タイプ	デフォルト
整数	1

注記
この値が 2 倍になると、システムは新しい割り当てを拒否します。

第6章 カーネル機能

本章では、多くのユーザー空間ツールを有効にするカーネル機能の目的および使用について説明します。また、これらのツールの詳細についてのリソースも紹介します。

6.1. コントロールグループ

6.1.1. コントロールグループとは?

注記

コントロールグループの名前空間は、Red Hat Enterprise Linux 7.5 ではテクノロジープレビューとして提供されています。

Linux コントロールグループ (cgroup) は、システムハードウェアの使用上の制限を有効にし、**cgroup** 内で実行する個々のプロセスが **cgroup** の設定で許可された分だけ活用していることを確認します。

コントロールグループ は、**名前空間** が有効にしたリソースの使用量を制限します。たとえば、ネットワーク名前空間により、あるプロセスの特定のネットワークカードへのアクセスが可能となり、cgroup はこのプロセスのカードの使用量が 50%を超えないように確認し、他のプロセスが帯域幅を確実に利用できるようにします。

コントロールグループの名前空間は、/proc/self/ns/cgroup インターフェースを経由して個々の cgroup の仮想化ビューを提供します。

目的は、グローバルな名前空間から cgroup への特権付きデータの漏えいを回避し、コンテナーマイグレーションなどの他の機能を有効にすることです。

現在、コンテナーを単一の cgroup と関連付けることが格段に容易となっていることから、コンテナーにはより一貫した cgroup のビューがあります。また、コンテナー内部のタスクを有効にし、属している cgroup の仮想化ビューが可能となります。

6.1.2. 名前空間とは?

名前空間は、分離したシステムリソースの仮想化ビューを可能にするカーネル機能です。システムリソースからプロセスを分離させることで、プロセスが相互作用可能なものを指定および制御できます。名前空間は、コントロールグループにとって不可欠なものです。

6.1.3. サポートしている名前空間

以下の名前空間は、Red Hat Enterprise Linux 7.5 以降でサポートされています。

- マウント
  - マウント名前空間は、ファイルシステムのマウントポイントを分離します。これにより各プロセスは明確なファイルシステム領域を持つことができ、その領域内で操作します。
- UTS
  - ホスト名および NIS ドメイン名
- IPC
  - System V IPC、POSIX メッセージキュー

- PID
  - プロセス ID
- ネットワーク
  - ネットワークデバイス、スタック、ポートなど。
- ユーザー
  - ユーザーおよびグループ ID
- コントロールグループ
  - cgroup の分離

注記

コントロールグループの使用方法は、『リソース管理ガイド』 で説明しています。

6.2. カーネルソースチェッカー

Linux カーネルモジュールのソースチェッカー (ksc) は、所定のカーネルモジュール内におけるホワイトリスト以外の記号を確認するツールです。Red Hat パートナーもこのツールを使用して、Red Hat Bugzilla データベースにバグを報告することで、ホワイトリストに含まれる記号のレビューを要求できます。

6.2.1. 使用方法

本ツールは "-k" オプションでモジュールへのパスを受け取ります。

```
# ksc -k e1000e.ko
Checking against architecture x86_64
Total symbol usage: 165 Total Non white list symbol usage: 74

# ksc -k /path/to/module
```

出力は、**$HOME/ksc-result.txt** に保存されます。ホワイトリストへ追加された記号のレビューが要求されると、ホワイトリストに記載されていない各記号の使用方法の説明が **ksc-result.txt** ファイルに追加される必要があります。"-p" オプションで **ksc** を実行することで、リクエストバグを報告することができます。

注記

KSC は現在、**xz** 圧縮をサポートしていません。**ksc** ツールは、**xz** 圧縮方法を処理できず、以下のエラーメッセージを表示します。

```
Invalid architecture, (Only kernel object files are supported)
```

この制限がなくなるまで、システム管理者は **ksc** ツールを実行する前に、**xz** 圧縮を使用して任意のサードパーティーモジュールを手動で展開する必要があります。

6.3. ファイルの DIRECT ACCESS (DAX)

DAX で知られるファイルの Direct Access により、アプリケーションは、ファイルシステムへのアクセスをバッファーするためにページキャッシュを使用することなく、ストレージデバイスへのデータの読み取りおよび書き込みが可能となります。

この機能は、'direct_access' ブロックデバイス操作を実装するファイルシステムで利用可能で、'-o' でファイルシステムをマウントするか、または **/etc/fstab** のオプションセクションに 'dax' を追加するかのいずれかで有効となります。

コードの実例を含む詳しい情報は、**kernel-doc** パッケージを参照してください。これは、**/usr/share/doc/kernel-doc-<version>/Documentation/filesystems/dax.txt** に保存されていて、'<version>' が対応するカーネルバージョン番号になります。

6.4. ユーザー空間用のメモリー保護キー (別名 PKU または PKEYS)

メモリー保護キーは、ページベースの保護を強化するメカニズムを提供しますが、アプリケーションが保護ドメインを変更した時にページテーブルを修正する必要はありません。これは、各ページテーブルエントリーで以前無視されていた 4 ビットを「保護キー」に割り当て、可能なキーを 16 個提供することで有効となります。

メモリー保護キーは、いくつかの Intel CPU チップセットのハードウェア機能です。 プロセッサーがこの機能をサポートしているかどうかは、**pku** in **/proc/cpuinfo** の存在を確認します。

```
$ grep pku /proc/cpuinfo
```

この機能をサポートするために、CPU は各キーに対して 2 つの別々のビット (Access Disable と Write Disable) を持つ、ユーザーがアクセス可能な新しいレジスター (PKRU) を提供します。この新しいレジスターの読み取りおよび書き込み用の新しい命令が 2 つ (RDPKRU と WRPKRU) 存在します。

プログラミングの実例を含む詳しい情報は、**kernel-doc** パッケージが提供する **/usr/share/doc/kernel-doc-*/Documentation/x86/protection-keys.txt** を参照してください。

6.5. KERNEL ADRESS SPACE LAYOUT RANDOMIZATION

Kernel Adress Space Layout Randomization (KASLR) は 2 つの部分で構成され、これらが共に機能して Linux カーネルのセキュリティーを強化します。

- カーネルテキスト KASLR
- メモリー管理の KASLR

カーネルテキストの物理アドレスと仮想アドレスの場所が、個別にランダム化されます。カーネルの物理アドレスは 64 TB の任意の場所に配置できますが、カーネルの仮想アドレスは、[0xffffffff80000000, 0xffffffffc0000000] の間の 1 GB 領域に制限されます。

メモリー管理の KASLR には 3 つのセクションがあり、これらのセクションの開始アドレスは特定のエリアでランダム化されます。したがって、悪意のコードが、カーネルアドレス領域にその記号が置かれていることを知る必要がある場合に、KASLR は悪意のコードにカーネルの実行を挿入またはリダイレクトしないようにすることができます。

メモリー管理の KASLR には、以下の 3 つのセクションがあります。

- 直接マッピングセクション
- vmalloc セクション

- vmemmap セクション

KASLR コードが Linux カーネルにコンパイルされ、デフォルトで有効になりました。明示的に無効にするには、**nokaslr** カーネルオプションをカーネルコマンドラインに追加します。

第7章 カーネルパラメーターおよび値の表示

7.1. カーネルコマンドラインパラメーター

カーネル引数の別名を持つカーネルコマンドラインパラメーターは、Red Hat Enterprise Linux の動作をカスタマイズするために起動時のみに使用されます。

7.1.1. カーネルコマンドラインパラメーターの設定

本セクションでは、**GRUB2** ブートローダーを使用した AMD64 および Intel 64 システムならびに IBM Power Systems サーバー、 そして **zipl** を使用した IBM System z で、カーネルコマンドラインパラメーターを変更する方法を説明します。

カーネルコマンドラインパラメーターは、**boot/grub/grub.cfg** 設定ファイルに保存されています。この設定ファイルは、**GRUB2** ブートローダーにより生成されます。この設定ファイルは編集しません。このファイルへの変更ができるのは、設定スクリプトのみです。

AMD64 および Intel 64 システムならびに IBM Power Systems ハードウェア向けに、GRUB2 のカーネルコマンドラインパラメーターを変更します 。

1. **vim** または **Gedit** などのプレーンテキストエディターを使って **/etc/default/grub** 設定ファイルを **root** として開きます。

2. このファイル内で、以下のように **GRUB_CMDLINE_LINUX** で始まるラインを探します。

```
GRUB_CMDLINE_LINUX="rd.lvm.lv=rhel/swap crashkernel=auto
rd.lvm.lv=rhel/root rhgb quiet"
```

3. 必要なカーネルコマンドラインパラメーターの値を変更します。続いて、ファイルを保存してエディターを終了します。

4. 編集された **default** ファイルを使用して、**GRUB2** 設定を再生成します。BIOS ファームウェアを使用しているシステムの場合は次のコマンドを実行します。

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
```

UEFI ファームウェアを使用しているシステムの場合は次のコマンドを実行します。

```
# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

上記の手順を完了するとブートローダーが再設定され、設定ファイルに指定したカーネルコマンドラインパラメーターが、次回の再起動後に適用されます。

BM System z ハードウェア用に zipl のカーネルコマンドラインパラメーターを変更します。

1. **vim** または **Gedit** などのプレーンテキストエディターを使って **/etc/zipl.conf** 設定ファイルを **root** として開きます。

2. このファイル内で **parameters=** セクションを探し、必要なパラメーターを編集するか、見つからない場合はパラメーターを追加します。続いてファイルを保存し、エディターを終了します。

3. **zipl** 設定を再生成します。

```
# zipl
```

注記

オプションを何も付けずに **zipl** コマンドのみを実行すると、デフォルト値が使用されます。使用できるオプションの一覧については **zipl(8)** の man ページを参照してください。

上記の手順を完了するとブートローダーが再設定され、設定ファイルに指定したカーネルコマンドラインパラメーターが、次回の再起動後に適用されます。

7.1.2. 制御可能なカーネルコマンドラインパラメーター

カーネルコマンドラインパラメーターの全一覧は、https://www.kernel.org/doc/Documentation/admin-guide/kernel-parameters.txt を参照してください。

7.1.2.1. ハードウェア固有のカーネルコマンドラインパラメーター

pci=option[,option...]

PCI ハードウェアサブシステムの動作を指定

設定	効果
earlydump	[X86] カーネルが何らかの変更をする前に PCI 設定領域をダンプします。
off	[X86] PCI バスを調べません。
noaer	[PCIE] PCIEAER カーネルパラメーターが有効な場合、このカーネル起動オプションを使用して PCIE Advanced Error Reporting (AER) の使用を無効にすることができます。
noacpi	[X86] Interrupt Request (IRQ) ルーティングまたは PCI スキャニング用に Advanced Configuration and Power Interface (ACPI) は使用しません。
bfsort	PCI デバイスを幅優先順に並べ替えます。この並べ替えは、古い (⇐ 2.4) カーネルに対応するデバイスの順番となります。
nobfsort	PCI デバイスを幅優先順に並べ替えません。

さらなる PCI オプションは、**kernel-doc-<version>.noarch** パッケージのディスクドキュメントに文書化されています。ここでは、'<version>' は対応するカーネルバージョンに置き換える必要があります。

acpi=option

Advanced Configuration and Power Interface (ACPI) の動作を指定します。

設定	効果
acpi=off	ACPI の無効化
acpi=ht	ACPI の起動テーブル構文解析を使用しますが、ACPI インタープリターは有効にしません これにより、ハイパースレッディングで **必要ではない** 任意の ACPI 機能が無効となります。
acpi=force	ACPI サブシステムの有効化が必要
acpi=strict	ACPI 仕様と完全に準拠していないプラットフォームに対する ACPI 層の耐性を低くします。
acpi_sci=<value>	ACPI SCI の割り込みをセットアップします。ここで <value> は、エッジ、レベル、ハイ、ローのうちの 1 つとなります。
acpi=noirq	IRQ ルーティングに ACPI は使用しません。
acpi=nocmcff	訂正されたエラーの firmware first (FF) モードを無効にします。これにより、ファームウェアが FF フラグを設定したかどうかを確認する HEST CMC エラーソースの構文解析が無効となります。この場合、訂正されたエラー報告が重複する可能性があります。

第8章 改訂履歴

0.1-4

2018 年 3 月 26 日 (月)、Marie Doleželová (mdolezel@redhat.com)

- 7.5 GA 公開用ドキュメントバージョン

0.1-2

2017 年 7 月 31 日 (月)、Mark Flitter (mflitter@redhat.com)

- 7.4 GA 公開用ドキュメントバージョン

0.1-0

2017 年 4 月 20 日 (木)、Mark Flitter (mflitter@redhat.com)

- レビュー向けの初期ビルド