Red Hat Enterprise Linux 8

管理、监控和更新内核

在 Red Hat Enterprise Linux 8 中管理 Linux 内核的指南

Last Updated: 2025-08-06

Red Hat Enterprise Linux 8 管理、监控和更新内核

在 Red Hat Enterprise Linux 8 中管理 Linux 内核的指南

Legal Notice

Abstract

作为系统管理员，您可以配置 Linux 内核以优化操作系统。对 Linux 内核进行修改可以提高系统性能、安全性和稳定性，并可以对系统进行审核并对问题进行故障排除。

Table of Contents

对红帽文档提供反馈

我们感谢您对我们文档的反馈。让我们了解如何改进它。

通过 Jira 提交反馈（需要帐户）

1. 登录到 Jira 网站。
2. 单击顶部导航栏中的 **Create**。
3. 在 **Summary** 字段中输入描述性标题。
4. 在 **Description** 字段中输入您的改进建议。包括到文档相关部分的链接。
5. 点对话框底部的 **Create**。

第 1 章 LINUX 内核

了解由红帽（红帽内核）提供和维护的 Linux 内核和 Linux 内核 RPM 软件包。使红帽内核保持更新，这可以确保操作系统具有最新的程序错误修复、性能增强和补丁，并与新硬件兼容。

1.1. 内核是什么

内核是 Linux 操作系统的核心部分，其管理系统资源，并提供硬件和软件应用程序之间的接口。

红帽内核是一个基于上游 Linux 主线内核的定制内核，红帽工程师对其进行了进一步的开发和强化，专注于稳定性和与最新技术和硬件的兼容性。

在红帽发布新内核版本前，内核需要通过一组严格的质量保证测试。

红帽内核以 RPM 格式打包，以便它们可以通过 **YUM** 软件包管理器轻松地升级和验证。

警告

红帽不支持 不是由红帽 编译的内核。

1.2. RPM 软件包

RPM 软件包由文件的存档和用来安装和擦除这些文件的元数据组成。具体来说，RPM 软件包包含以下部分：

GPG 签名

GPG 签名用于验证软件包的完整性。

标头（软件包元数据）

RPM 软件包管理器使用此元数据确定软件包依赖项、安装文件的位置以及其他信息。

payload

有效负载是一个 **cpio** 归档，其包含要安装到系统的文件。

RPM 软件包有两种类型。这两种类型都共享文件格式和工具，但内容不同，并实现不同的目的：

- 源 RPM（SRPM）
  SRPM 包含源代码和 **spec** 文件，该文件描述了如何将源代码构建为二进制 RPM。另外，SRPM 可以包含源代码的补丁。

- 二进制 RPM
  一个二进制 RPM 包含了根据源代码和补丁构建的二进制文件。

1.3. LINUX 内核 RPM 软件包概述

kernel RPM 是一个元数据软件包，它不包含任何文件，而是保证正确安装了以下子软件包：

kernel-core

提供内核的二进制镜像、所有与 **initramfs**相关的对象来引导系统，以及确保核心功能的最小内核模块数量。仅在虚拟和云环境中使用这个子软件包来为 Red Hat Enterprise Linux 8 内核提供一个快速引导时间和小磁盘空间。

kernel-modules

提供内核 核心中不存在的其余内核模块。

上述 **kernel** 子软件包中的一部分旨在帮助系统管理员减少需要维护的范围，特别是在虚拟化和云环境中。

例如，可选内核软件包：

kernel-modules-extra

为个别硬件提供内核模块。默认禁用模块的加载。

kernel-debug

为内核诊断启用了许多调试选项，以牺牲性能为代价提供内核。

kernel-tools

提供操作 Linux 内核和支持文档的工具。

kernel-devel

提供内核标头和 makefile，它们足以根据 内核软件包构建模块。

kernel-abi-stablelists

提供与 RHEL 内核 ABI 相关的信息，包括外部 Linux 内核模块所需的内核符号列表和 **yum** 插件以协助执行。

kernel-headers

包括指定 Linux 内核和用户空间库以及程序间接口的 C 标头文件。头文件定义构建大多数标准程序所需的结构和常量。

其他资源

- *什么是 kernel-core、kernel-modules 和 kernel-modules-extras 软件包？*

1.4. 显示内核软件包的内容

通过查询存储库，您可以看到内核软件包是否提供了特定的文件，如模块。不需要下载或安装软件包来显示文件列表。

使用 **dnf** 工具查询文件列表，例如 **kernel-core**、**kernel-modules-core** 或 **kernel-modules** 软件包的文件列表。请注意，**kernel** 软件包是一个不包含任何文件的元数据软件包。

流程

1. 列出软件包的可用版本：

```
$ yum repoquery <package_name>
```

2. 显示软件包中的文件列表：

```
$ yum repoquery -l <package_name>
```

其他资源

- 打包和分发软件

1.5. 安装特定的内核版本

使用 **yum** 软件包管理器安装新内核。

流程

- 要安装特定的内核版本，请输入以下命令：

```
# yum install kernel-5.14.0
```

其它资源

- *Red Hat Enterprise Linux 发行日期*

1.6. 更新内核

使用 **yum** 软件包管理器更新内核。

流程

1. 要更新内核，请输入以下命令：

```
# yum update kernel
```

此命令将内核以及所有依赖项更新至最新可用版本。

2. 重启您的系统以使更改生效。

注意

当从 RHEL 7 升级到 RHEL 8 时，请遵循 *从 RHEL 7 升级到 RHEL 8* 文档中的相关部分。

其它资源

- 管理软件包

1.7. 将内核设置为默认

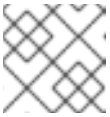
使用 **grubby** 命令行工具和 GRUB 将特定内核设置为默认。

流程

- 使用 **grubby** 工具将内核设置为默认。
  - 输入以下命令，使用 **grubby** 工具将内核设置为默认：

```
# grubby --set-default $kernel_path
```

命令使用不带 **.conf** 后缀的计算机 ID 作为参数。

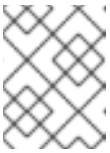


注意

机器 ID 位于 **/boot/loader/entries/** 目录中。

- 使用 **id** 参数将内核设置为默认。
  - 使用 **id** 参数列出引导条目，然后将所需的内核设置为默认：

```
# grubby --info ALL | grep id
# grubby --set-default /boot/vmlinuz-<version>.<architecture>
```



注意

要使用 **title** 参数列出引导条目，请执行 **# grubby --info=ALL | grep title** 命令。

- 仅为下次引导设置默认内核。
  - 执行以下命令，仅在下次使用 **grub2-reboot** 命令重新引导时设置默认内核：

```
# grub2-reboot <index|title|id>
```

警告

小心地为下次启动设置默认内核。安装新的内核 RPM、自构建的内核，并手动将条目添加到 **/boot/loader/entries/** 目录中可能会更改索引值。

第 2 章 管理内核模块

了解内核模块、如何显示其信息，以及如何使用内核模块执行基本的管理任务。

2.1. 内核模块简介

Red Hat Enterprise Linux 内核可以使用内核模块扩展，该模块提供可选的附加功能，而无需重启系统。在 RHEL 8 中，内核模块是内置于压缩的 < **KERNEL_MODULE_NAME>.ko.xz** 对象文件中的额外内核代码。

内核模块启用的最常见功能是：

- 添加用于支持新硬件的设备驱动程序
- 支持文件系统，如 GFS2 或者 NFS
- 系统调用

在现代系统中，在需要时会自动载入内核模块。但在某些情况下，需要手动加载或卸载模块。

与内核类似，模块也接受自定义其行为的参数。

您可以使用内核工具对模块执行以下操作：

- 检查当前运行的模块。
- 检查可用于加载到内核的模块。
- 检查模块接受的参数。
- 启用将内核模块和卸载到运行的内核中的机制。

2.2. 内核模块依赖关系

某些内核模块有时依赖一个或多个内核模块。**/lib/modules/<KERNEL_VERSION>/modules.dep** 文件包含相应内核版本的完整内核模块依赖项列表。

depmod

依赖项文件由 **depmod** 程序生成，包括在 **kmod** 软件包中。**kmod** 提供的很多工具会在执行操作时考虑模块依赖项。因此，很少需要 手动 查找依赖项。

警告

内核模块的代码在不受限制的模式下在内核空间中执行。请注意您要加载的模块。

weak-modules

除了 **depmod** 外，Red Hat Enterprise Linux 还提供了 **weak-modules** 脚本，这是 **kmod** 软件包的一部分。**weak-modules** 决定与安装的内核 kABI 兼容的模块。在检查模块内核的兼容性时，**weak-modules** 按照它们构建的内核的从高到低版本处理符号依赖项。它独立于内核版本单独处理每个模块。

其他资源

- **modules.dep (5)** 手册页
- **depmod (8)** 手册页
- 与 Red Hat Enterprise Linux 提供的 weak-modules 脚本的作用是什么？
- 什么是内核应用程序二进制接口(kABI)？（红帽知识库）

2.3. 列出已安装的内核模块

grubby --info=ALL 命令显示在 **!BLS** 和 **BLS** 安装中安装的内核的一个索引列表。

步骤

- 使用以下命令列出安装的内核：

```
# grubby --info=ALL | grep title
```

下面是所有安装的内核列表：

```
title=Red Hat Enterprise Linux (4.18.0-20.el8.x86_64) 8.0 (Ootpa)
title=Red Hat Enterprise Linux (4.18.0-19.el8.x86_64) 8.0 (Ootpa)
title=Red Hat Enterprise Linux (4.18.0-12.el8.x86_64) 8.0 (Ootpa)
title=Red Hat Enterprise Linux (4.18.0) 8.0 (Ootpa)
title=Red Hat Enterprise Linux (0-rescue-2fb13ddde2e24fde9e6a246a942caed1) 8.0 (Ootpa)
```

这是 GRUB 菜单中安装的 grubby-8.40-17 的内核列表。

2.4. 列出当前载入的内核模块

查看当前载入的内核模块。

先决条件

- 已安装 **kmod** 软件包。

步骤

- 要列出所有当前载入的内核模块，请输入：

```
$ lsmod

Module                  Size  Used by
fuse                  126976  3
uinput                 20480  1
xt_CHECKSUM            16384  1
ipt_MASQUERADE         16384  1
```

```
xt_conntrack          16384  1
ipt_REJECT            16384  1
nft_counter           16384  16
nf_nat_tftp           16384  0
nf_conntrack_tftp     16384  1 nf_nat_tftp
tun                   49152  1
bridge                192512  0
stp                   16384  1 bridge
llc                   16384  2 bridge,stp
nf_tables_set         32768  5
nft_fib_inet          16384  1
…
```

在上例中：

a. **Module** 列提供了当前载入的模块的 名称。

b. **Size** 列显示每个模块的 内存 量（以 KB 为单位）。

c. **Used by** 列显示 依赖于 特定模块的模块的编号，以及名称（可选）。

其他资源

- **/usr/share/doc/kmod/README** 文件
- **lsmod (8)** 手册页

2.5. 列出所有安装的内核

使用 **grubby** 实用程序列出系统上所有安装的内核。

先决条件

- 您有 root 权限。

流程

- 要列出所有安装的内核，请输入：

```
# grubby --info=ALL | grep ^kernel

kernel="/boot/vmlinuz-4.18.0-305.10.2.el8_4.x86_64"
kernel="/boot/vmlinuz-4.18.0-240.el8.x86_64"
kernel="/boot/vmlinuz-0-rescue-41eb2e172d7244698abda79a51778f1b"
```

输出显示了安装的所有内核的路径和版本。

2.6. 显示内核模块信息

使用 **modinfo** 命令显示指定内核模块的一些详细信息。

先决条件

- 已安装 **kmod** 软件包。

步骤

- 要显示关于任何内核模块的信息，请输入：

```
$ modinfo <KERNEL_MODULE_NAME>
```

例如：

```
$ modinfo virtio_net

filename:       /lib/modules/4.18.0-94.el8.x86_64/kernel/drivers/net/virtio_net.ko.xz
license:        GPL
description:    Virtio network driver
rhelversion:    8.1
srcversion:     2E9345B281A898A91319773
alias:          virtio:d00000001v*
depends:        net_failover
intree:         Y
name:           virtio_net
vermagic:       4.18.0-94.el8.x86_64 SMP mod_unload modversions
…
parm:           napi_weight:int
parm:           csum:bool
parm:           gso:bool
parm:           napi_tx:bool
```

您可以查询所有可用模块的信息，无论它们是否已加载。**parm** 条目显示用户可以为模块设置的参数，以及它们预期的值类型。

注意

在输入内核模块的名称时，不要将 **.ko.xz** 扩展附加到名称的末尾。内核模块名称没有扩展名，它们对应的文件有。

其它资源

- **modinfo (8)** 手册页

2.7. 在系统运行时载入内核模块

扩展 Linux 内核功能的最佳方法是加载内核模块。使用 **modprobe** 命令查找并将内核模块载入到当前运行的内核中。

重要

重启系统后，这个过程中描述的更改不会保留。有关如何在系统重启后将内核模块载入为持久性 的详情，请参考 在系统引导时自动载入内核模块。

先决条件

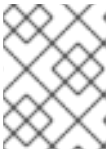
- 根权限

- 已安装 **kmod** 软件包。
- 相关的内核模块没有被加载。要确保情况如此，请列出 当前载入的内核模块 的列表。

步骤

1. 选择您要载入的内核模块。
   模块位于 **/lib/modules/$(uname -r)/kernel/<SUBSYSTEM>/** 目录中。
2. 载入相关内核模块：

```
# modprobe <MODULE_NAME>
```



注意

在输入内核模块的名称时，不要将 **.ko.xz** 扩展附加到名称的末尾。内核模块名称没有扩展名，它们对应的文件有。

验证

- （可选）验证载入了相关模块：

```
$ lsmod | grep <MODULE_NAME>
```

如果正确加载了模块，这个命令会显示相关的内核模块。例如：

```
$ lsmod | grep serio_raw
serio_raw              16384  0
```

其他资源

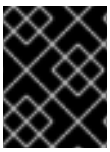
- **modprobe (8)** 手册页

2.8. 在系统运行时卸载内核模块

要从正在运行的内核中卸载某些内核模块，请使用 **modprobe** 命令在当前载入的内核的系统运行时查找和卸载内核模块。

警告

您不能卸载正在运行的系统使用的内核模块，因为它可能会导致不稳定或系统无法正常工作。



重要

完成不活跃内核模块的卸载后，定义为在引导时自动载入的模块，在重启系统后不会保持卸载。有关如何防止此结果的详情，请参考 防止内核模块在系统引导时被自动载入。

先决条件

- 您有 root 权限。
- 已安装 **kmod** 软件包。

流程

1. 列出所有载入的内核模块：

```
# lsmod
```

2. 选择您要卸载的内核模块。
如果内核模块有依赖项，请在卸载内核模块前卸载它们。有关识别使用依赖项的模块的详情，请参阅列出当前载入的内核模块和内核模块依赖项。

3. 卸载相关内核模块：

```
# modprobe -r <MODULE_NAME>
```

在输入内核模块的名称时，不要将 **.ko.xz** 扩展附加到名称的末尾。内核模块名称没有扩展名，它们对应的文件有。

验证

- （可选）验证相关模块是否已卸载：

```
$ lsmod | grep <MODULE_NAME>
```

如果模块成功卸载，这个命令不会显示任何输出。

其他资源

- **modprobe(8)** 手册页

2.9. 在启动过程早期卸载内核模块

在某些情况下，例如，当内核模块有导致系统变得无响应的代码时，用户无法访问该阶段来永久禁用恶意内核模块，您可能需要在引导过程早期卸载内核模块。要临时阻止加载内核模块，您可以使用引导装载程序。

您可以在引导序列继续前编辑相关的引导装载程序条目来卸载所需的内核模块。

重要

此流程中描述的更改在下次重启后 不会持久存在 。有关如何将内核模块添加到 denylist 中，以便在引导过程中不会被自动载入，请参阅 防止在系统引导时自动载入内核模块。

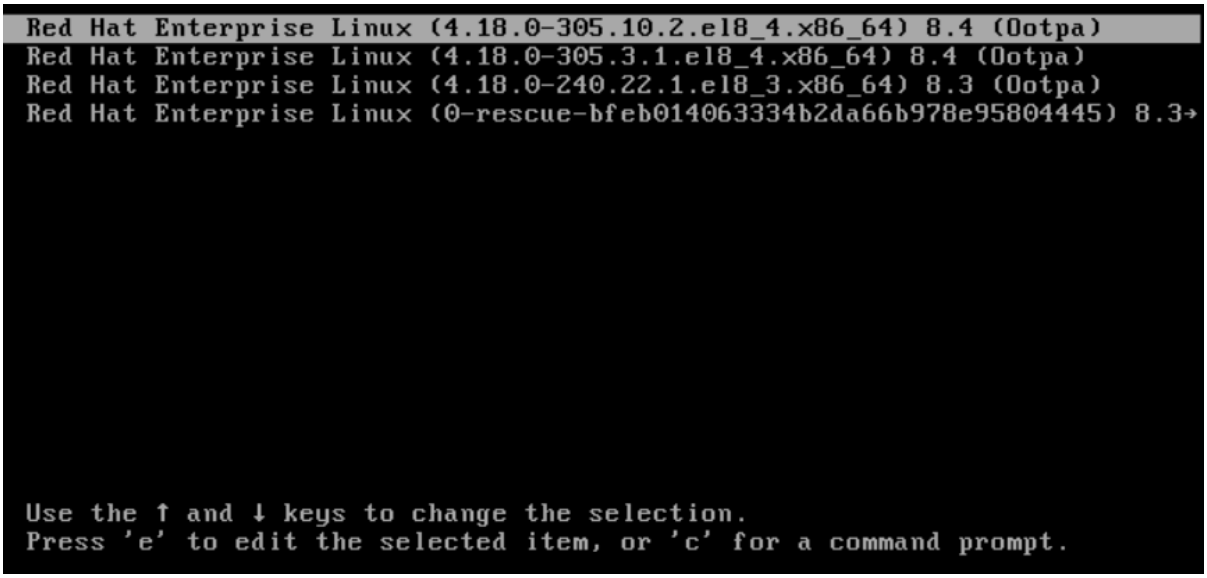
先决条件

- 您有一个可加载的内核模块，但出于某种原因您要防止其加载。

流程

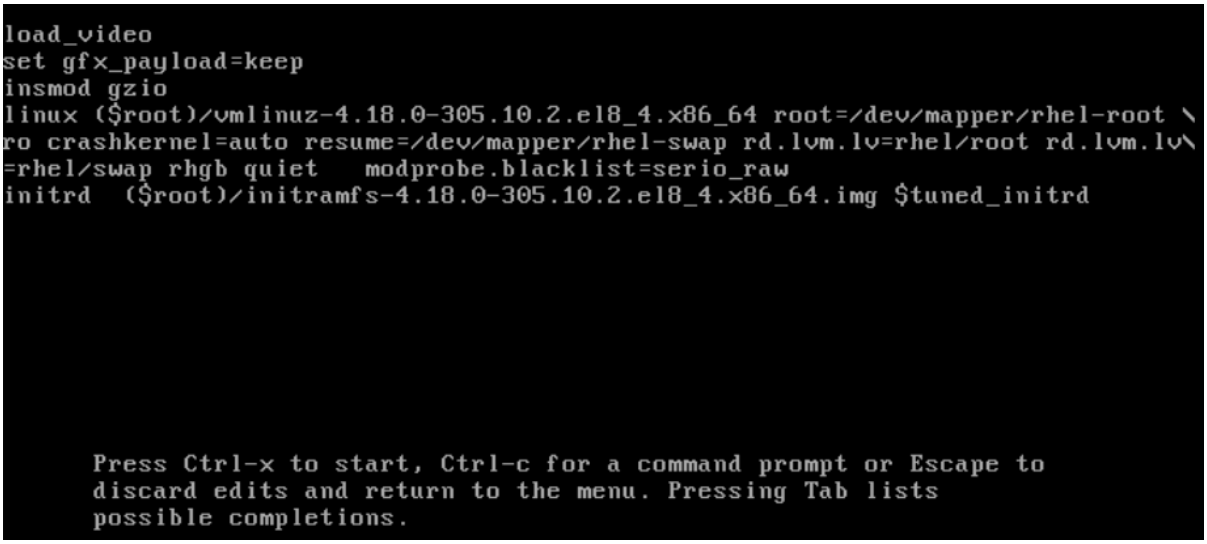
1. 将系统启动到引导装载程序中。
2. 使用光标键突出显示相关的引导装载程序条目。
3. 按 **e** 键编辑条目。

   图 2.1. 内核引导菜单



4. 使用光标键导航到以 **linux** 开头的行。
5. 将 **modprobe.blacklist=*module_name*** 附加到行尾。

   图 2.2. 内核引导条目



   serio_raw 内核模块演示了一个要在引导过程早期卸载的恶意模块。

6. 按 **Ctrl**+**X** 使用修改后的配置启动。

验证

- 引导系统后，验证相关内核模块是否没有载入：

```
# lsmod | grep serio_raw
```

其他资源

- 管理内核模块

2.10. 在系统引导时自动载入内核模块

配置内核模块以在引导过程中自动载入它。

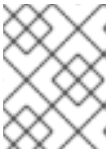
先决条件

- 根权限
- 已安装 **kmod** 软件包。

步骤

1. 选择您要在引导过程中载入的内核模块。
   模块位于 **/lib/modules/$(uname -r)/kernel/<SUBSYSTEM>/** 目录中。
2. 为模块创建配置文件：

```
# echo <MODULE_NAME> > /etc/modules-load.d/<MODULE_NAME>.conf
```



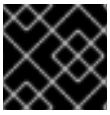
注意

在输入内核模块的名称时，不要将 **.ko.xz** 扩展附加到名称的末尾。内核模块名称没有扩展名，它们对应的文件有。

验证

1. 重启后，验证载入了相关模块：

```
$ lsmod | grep <MODULE_NAME>
```



重要

重启系统后，这个过程中描述的更改将会保留。

其它资源

- **modules-load.d(5)** 手册页

2.11. 防止在系统引导时自动载入内核模块

您可以通过使用相应的命令在 **modprobe** 配置文件中列出模块，来防止系统在引导过程中自动载入内核模块。

先决条件

- 此流程中的命令需要 root 权限。使用 **su -** 切换到 root 用户，或在命令前使用 **sudo** 。
- 已安装 **kmod** 软件包。

- 确定您当前的系统配置不需要您计划拒绝的内核模块。

流程

1. 使用 **lsmod** 命令列出载入到当前运行的内核的模块：

```
$ lsmod
Module                  Size  Used by
tls                   131072  0
uinput                 20480  1
snd_seq_dummy          16384  0
snd_hrtimer            16384  1
…
```

在输出中，找到您要阻止加载的模块。

- 或者，识别您要防止在 **/lib/modules/*<KERNEL-VERSION>*/kernel/*<SUBSYSTEM>*/** 目录中加载的而未加载的内核模块，例如：

```
$ ls /lib/modules/4.18.0-477.20.1.el8_8.x86_64/kernel/crypto/
ansi_cprng.ko.xz        chacha20poly1305.ko.xz  md4.ko.xz
serpent_generic.ko.xz
anubis.ko.xz            cmac.ko.xz…
```

2. 创建一个配置文件作为 denylist ：

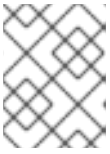
```
# touch /etc/modprobe.d/denylist.conf
```

3. 在您选择的文本编辑器中，使用 **blacklist** 配置命令将您要从自动加载到内核中排除的模块的名称组合在一起，例如：

```
# Prevents <KERNEL-MODULE-1> from being loaded
blacklist <MODULE-NAME-1>
install <MODULE-NAME-1> /bin/false

# Prevents <KERNEL-MODULE-2> from being loaded
blacklist <MODULE-NAME-2>
install <MODULE-NAME-2> /bin/false
…
```

因为 **blacklist** 命令不会阻止模块作为不在 denylist 中的另一个内核模块的依赖项加载，所以您还必须定义 安装 行。在这种情况下，系统运行 **/bin/false**，而不是安装模块。以哈希符号开头的行是注释，您可以用来使文件更易读。



注意

在输入内核模块的名称时，不要将 **.ko.xz** 扩展附加到名称的末尾。内核模块名称没有扩展名，它们对应的文件有。

4. 在重建前，创建当前初始 RAM 磁盘镜像的一个备份副本：

```
# cp /boot/initramfs-$(uname -r).img /boot/initramfs-$(uname -r).bak.$(date +%m-%d-%H%M%S).img
```

- 或者，创建与您要阻止内核模块自动载入的内核版本对应的初始 RAM 磁盘镜像的一个备份副本：

```
# cp /boot/initramfs-<VERSION>.img /boot/initramfs-<VERSION>.img.bak.$(date +%m-%d-%H%M%S)
```

5. 生成一个新的初始 RAM 磁盘镜像以应用更改：

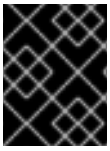
```
# dracut -f -v
```

- 如果您为与您系统当前使用的内核版本不同的系统构建初始 RAM 磁盘镜像，请指定目标 **initramfs** 和内核版本：

```
# dracut -f -v /boot/initramfs-<TARGET-VERSION>.img <CORRESPONDING-TARGET-KERNEL-VERSION>
```

6. 重启系统：

```
$ reboot
```



重要

此流程中描述的更改将在重启后生效并保留。如果您在 denylist 中错误地列出了关键内核模块，您可以将系统切换到不稳定或无法正常工作的状态。

其他资源

- 如何防止内核模块自动加载？（红帽知识库）
- 您系统上的 **modprobe.d (5)** 和 **dracut (8)** 手册页

2.12. 编译自定义的内核模块

您可以根据硬件和软件级别的各种配置的要求构建一个采样内核模块。

先决条件

- **kernel-devel**、**gcc** 和 **elfutils-libelf-devel** 软件包已安装。

```
# dnf install kernel-devel-$(uname -r) gcc elfutils-libelf-devel
```

- 您有 root 权限。
- 您创建了 **/root/testmodule/** 目录,在此编译自定义的内核模块。

步骤

1. 创建包含以下内容的 **/root/testmodule/test.c** 文件：

```
#include <linux/module.h>
#include <linux/kernel.h>
```

```
int init_module(void)
   { printk("Hello World\n This is a test\n"); return 0; }

void cleanup_module(void)
   { printk("Good Bye World"); }
```

test.c 文件是一个源文件，其向内核模块提供主要功能。出于组织需要，该文件已创建在专用的 **/root/testmodule/** 目录中。在模块编译后，**/root/testmodule/** 目录将包含多个文件。

test.c 文件包含来自系统库的文件：

- 示例代码中的 **printk**（） 函数需要 **linux/kernel.h** 头文件。
- **linux/module.h** 头文件包含在多个 C 源文件之间共享的功能声明和宏定义。

2. 按照 **init_module** （） 和 **cleanup_module** （） 函数启动和结束内核日志记录函数 **printk**（），后者会打印文本。

3. 使用以下内容创建 **/root/testmodule/Makefile** 文件。

```
obj-m := test.o
```

Makefile 包含编译器的说明，用于生成名为 **test.o** 的对象文件。**obj-m** 指令指定生成的 **test.ko** 文件将编译为可加载的内核模块。或者，**obj-y** 指令还可指示将 **test.ko** 构建为内置内核模块。

4. 编译内核模块。

```
# make -C /lib/modules/$(uname -r)/build M=/root/testmodule modules
make: Entering directory '/usr/src/kernels/4.18.0-305.el8.x86_64'
  CC [M]  /root/testmodule/test.o
  Building modules, stage 2.
  MODPOST 1 modules
WARNING: modpost: missing MODULE_LICENSE() in /root/testmodule/test.o
see include/linux/module.h for more information
  CC      /root/testmodule/test.mod.o
  LD [M]  /root/testmodule/test.ko
make: Leaving directory '/usr/src/kernels/4.18.0-305.el8.x86_64'
```

编译器将它们链接成最终内核模块(**test.ko**)之前，会为每个源文件(**test.c)**创建一个对象文件(**test.c**)来作为中间步骤。

成功编译后，**/root/testmodule/** 包含与编译的自定义内核模块相关的其他文件。已编译的模块本身由 **test.ko** 文件表示。

验证

1. 可选：检查 **/root/testmodule/** 目录的内容：

```
# ls -l /root/testmodule/
total 152
-rw-r—r--. 1 root root    16 Jul 26 08:19 Makefile
-rw-r—r--. 1 root root    25 Jul 26 08:20 modules.order
-rw-r—r--. 1 root root     0 Jul 26 08:20 Module.symvers
-rw-r—r--. 1 root root   224 Jul 26 08:18 test.c
-rw-r—r--. 1 root root 62176 Jul 26 08:20 test.ko
```

```
-rw-r—r--. 1 root root    25 Jul 26 08:20 test.mod
-rw-r—r--. 1 root root   849 Jul 26 08:20 test.mod.c
-rw-r—r--. 1 root root 50936 Jul 26 08:20 test.mod.o
-rw-r—r--. 1 root root 12912 Jul 26 08:20 test.o
```

2. 将内核模块复制到 **/lib/modules/$(uname -r)/** 目录中：

```
# cp /root/testmodule/test.ko /lib/modules/$(uname -r)/
```

3. 更新模块依赖项列表：

```
# depmod -a
```

4. 载入内核模块：

```
# modprobe -v test
insmod /lib/modules/4.18.0-305.el8.x86_64/test.ko
```

5. 验证内核模块是否成功载入：

```
# lsmod | grep test
test                   16384  0
```

6. 从内核环缓冲中读取最新的消息：

```
# dmesg
[74422.545004] Hello World
               This is a test
```

第 3 章 为安全引导签名内核和模块

您可以使用签名的内核和签名的内核模块来加强系统的安全性。在启用了安全引导机制的基于 UEFI 的构建系统中，您可以自我签名一个私有构建的内核或内核模块。另外，您可以将公钥导入到要部署内核或内核模块的目标系统中。

如果启用了安全引导机制，则必须使用私钥签名以下所有组件，并使用对应的公钥进行身份验证：

- UEFI 操作系统引导装载程序
- Red Hat Enterprise Linux 内核
- 所有内核模块

如果这些组件中的任何一个都没有签名和验证，则系统将无法完成引导过程。

RHEL 8 包括：

- 签名的引导装载程序
- 签名的内核
- 签名的内核模块

此外，签名的第一阶段引导装载程序和签名的内核包括嵌入的红帽公钥。这些签名的可执行二进制文件和嵌入式密钥可让 RHEL 8 安装、引导和使用 Microsoft UEFI 安全引导认证认证机构密钥运行。这些密钥由支持 UEFI 安全引导的系统上的 UEFI 固件提供。

注意

- 不是所有基于 UEFI 的系统都包括对安全引导的支持。
- 构建系统（构建和签署内核模块）不需要启用 UEFI 安全引导，甚至不需要是基于 UEFI 的系统。

3.1. 先决条件

- 要能够为外部构建的内核模块签名，请从以下软件包安装工具：

```
# yum install pesign openssl kernel-devel mokutil keyutils
```

表 3.1. 所需工具

工具	由软件包提供	用于	用途
efikeygen	**pesign**	构建系统	生成公共和专用 X.509 密钥对
openssl	**openssl**	构建系统	导出未加密的私钥
sign-file	**kernel-devel**		

3.2.

-
-
-
-
-
-
-
-
-
-

3.3.

-
-
-
-

3.4.

-
  -
  -
-
-
  -
  -

表 3.2.

3.5.

-
-
-

表 3.3.

-
-
-
-
-
-
-
-
-
-
-

-
-
-

3.6.

警告

-
  -
    ```
    # efikeygen --dbdir /etc/pki/pesign \
                --self-sign \
                --module \
                --common-name 'CN=Organization signing key' \
                --nickname 'Custom Secure Boot key'
    ```
  -
    ```
    # efikeygen --dbdir /etc/pki/pesign \
                --self-sign \
                --kernel \
                --common-name 'CN=Organization signing key' \
                --nickname 'Custom Secure Boot key'
    ```
  -
    ```
    # efikeygen --dbdir /etc/pki/pesign \
                --self-sign \
                --kernel \
                --common-name 'CN=Organization signing key' \
                --nickname 'Custom Secure Boot key'
                --token 'NSS FIPS 140-2 Certificate DB'
    ```

注意

重要

-
-
-

3.7.

-
-

-

例 3.1.

```
# keyctl list %:.builtin_trusted_keys
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...

# keyctl list %:.platform
4 keys in keyring:
...asymmetric: VMware, Inc.: 4ad8da0472073...
...asymmetric: Red Hat Secure Boot CA 5: cc6fafe72...
...asymmetric: Microsoft Windows Production PCA 2011: a929f298e1...
...asymmetric: Microsoft Corporation UEFI CA 2011: 13adbf4e0bd82...

# keyctl list %:.blacklist
4 keys in keyring:
...blacklist: bin:f5ff83a...
...blacklist: bin:0dfdbec...
...blacklist: bin:38f1d22...
...blacklist: bin:51f831f...
```

例 3.2.

```
# dmesg | egrep 'integrity.*cert'
[1.512966] integrity: Loading X.509 certificate: UEFI:db
[1.513027] integrity: Loaded X.509 cert 'Microsoft Windows Production PCA 2011:
a929023...
[1.513028] integrity: Loading X.509 certificate: UEFI:db
[1.513057] integrity: Loaded X.509 cert 'Microsoft Corporation UEFI CA 2011: 13adbf4309...
[1.513298] integrity: Loading X.509 certificate: UEFI:MokListRT (MOKvar table)
[1.513549] integrity: Loaded X.509 cert 'Red Hat Secure Boot CA 5:
cc6fa5e72868ba494e93...
```

-

3.8.

注意

-

1. ```
   # certutil -d /etc/pki/pesign \
           -n 'Custom Secure Boot key' \
           -Lr \
           > sb_cert.cer
   ```
   ```

-

2. ```
   # mokutil --import sb_cert.cer
   ```

3.

4.

5.

3.9.

-
-
-

- a. ```
     # pesign --certificate 'Custom Secure Boot key' \
           --in vmlinuz-version \
           --sign \
           --out vmlinuz-version.signed
     ```

  b. ```
     # pesign --show-signature \
           --in vmlinuz-version.signed
     ```

  c. ```
     # mv vmlinuz-version.signed vmlinuz-version
     ```

- a. ```
     # zcat vmlinuz-version > vmlinux-version
     ```

  b. ```
     # pesign --certificate 'Custom Secure Boot key' \
           --in vmlinux-version \
           --sign \
           --out vmlinux-version.signed
     ```

  c. ```
     # pesign --show-signature \
           --in vmlinux-version.signed
     ```

  d. ```
     # gzip --to-stdout vmlinux-version.signed > vmlinuz-version
     ```

  e. ```
     # rm vmlinux-version*
     ```

## 3.10.

- 
- 
     ```

-

- a.
```
# pesign --in /boot/efi/EFI/redhat/grubx64.efi \
        --out /boot/efi/EFI/redhat/grubx64.efi.signed \
        --certificate 'Custom Secure Boot key' \
        --sign
```

  b.
```
# pesign --in /boot/efi/EFI/redhat/grubx64.efi.signed \
        --show-signature
```

  c.
```
# mv /boot/efi/EFI/redhat/grubx64.efi.signed \
    /boot/efi/EFI/redhat/grubx64.efi
```

- a.
```
# pesign --in /boot/efi/EFI/redhat/grubaa64.efi \
        --out /boot/efi/EFI/redhat/grubaa64.efi.signed \
        --certificate 'Custom Secure Boot key' \
        --sign
```

  b.
```
# pesign --in /boot/efi/EFI/redhat/grubaa64.efi.signed \
        --show-signature
```

  c.
```
# mv /boot/efi/EFI/redhat/grubaa64.efi.signed \
    /boot/efi/EFI/redhat/grubaa64.efi
```

3.11.

-
-
-

1.
```
# certutil -d /etc/pki/pesign \
        -n 'Custom Secure Boot key' \
        -Lr \
        > sb_cert.cer
```

2.
```
# pk12util -o sb_cert.p12 \
        -n 'Custom Secure Boot key' \
        -d /etc/pki/pesign
```

3.

4.
```
# openssl pkcs12 \
        -in sb_cert.p12 \
        -out sb_cert.priv \
        -nocerts \
        -nodes
```

重要

5. # /usr/src/kernels/$(uname -r)/scripts/sign-file \
   sha256 \
   sb_cert.priv \
   sb_cert.cer \
   my_module.ko

重要

1. # modinfo *my_module*.ko | grep signer
   signer: *Your Name Key*

注意

2. # insmod *my_module*.ko
3. # modprobe -r *my_module*.ko

-

3.12.

-
-
-
- **# yum -y install kernel-modules-extra**

1. **# keyctl list %:.platform**
2. **# cp *my_module*.ko /lib/modules/$(uname -r)/extra/**
3. **# depmod -a**
4. **# modprobe -v my_module**
5. **# echo "*my_module*" > /etc/modules-load.d/*my_module*.conf**

- **# lsmod | grep my_module**

-

第 4 章

重要

4.1.

-
-
-

注意

-
-
-
-
-

4.2.

```
6f9cc9cb7d7845d49698c9537337cedc-4.18.0-5.el8.x86_64.conf
```

```
title Red Hat Enterprise Linux (4.18.0-74.el8.x86_64) 8.0 (Ootpa)
version 4.18.0-74.el8.x86_64
linux /vmlinuz-4.18.0-74.el8.x86_64
initrd /initramfs-4.18.0-74.el8.x86_64.img $tuned_initrd
options $kernelopts $tuned_params
id rhel-20190227183418-4.18.0-74.el8.x86_64
grub_users $grub_users
grub_arg --unrestricted
grub_class kernel
```

-

4.3.

-
-
-

```
# grubby --update-kernel=ALL --args="<NEW_PARAMETER>"
```

    - # zipl

- **# grubby --update-kernel=ALL --remove-args="<*PARAMETER_TO_REMOVE*>"**
    - # zipl

注意

-
-

4.4.

-
- **# grubby --update-kernel=/boot/vmlinuz-$(uname -r) --args="<*NEW_PARAMETER*>"**
    - # zipl
- **# grubby --update-kernel=/boot/vmlinuz-$(uname -r) --remove-args="<*PARAMETER_TO_REMOVE*>"**
    - # zipl

注意

重要

-
-
-

4.5.

注意

1.
2.
3.
4.

5.  注意

6. linux ($root)/vmlinuz-4.18.0-348.12.2.el8_5.x86_64 root=/dev/mapper/rhel-root ro crashkernel=auto resume=/dev/mapper/rhel-swap rd.lvm.lv=rhel/root rd.lvm.lv=rhel/swap rhgb quiet emergency

7.

 重要

4.6.

-

1. **GRUB_TERMINAL="serial"**
   GRUB_SERIAL_COMMAND="serial --speed=9600 --unit=0 --word=8 --parity=no --stop=1"

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

3.

第 5 章

重要

5.1.

-
-
-

表 5.1.

-

5.2.

-

1. **# sysctl -a**

注意

2. # sysctl <TUNABLE_CLASS>.<PARAMETER>=<TARGET_VALUE>

注意

-
-

5.3.

-

1. **# sysctl -a**

2. # sysctl -w <TUNABLE_CLASS>.<PARAMETER>=<TARGET_VALUE> >> /etc/sysctl.conf

注意

-
-

5.4.

-

1. **# vim /etc/sysctl.d/<*some_file.conf*>**

2. **<*TUNABLE_CLASS*>.<*PARAMETER*>=<*TARGET_VALUE*>**
   <*TUNABLE_CLASS*>.<*PARAMETER*>=<*TARGET_VALUE*>

3.

4. - **# sysctl -p /etc/sysctl.d/<*some_file.conf*>**

-

5.5.

-

1. **# ls -l /proc/sys/<*TUNABLE_CLASS*>/**

2. **# echo <*TARGET_VALUE*> > /proc/sys/<*TUNABLE_CLASS*>/<*PARAMETER*>**

1. **# cat /proc/sys/<*TUNABLE_CLASS*>/<*PARAMETER*>**

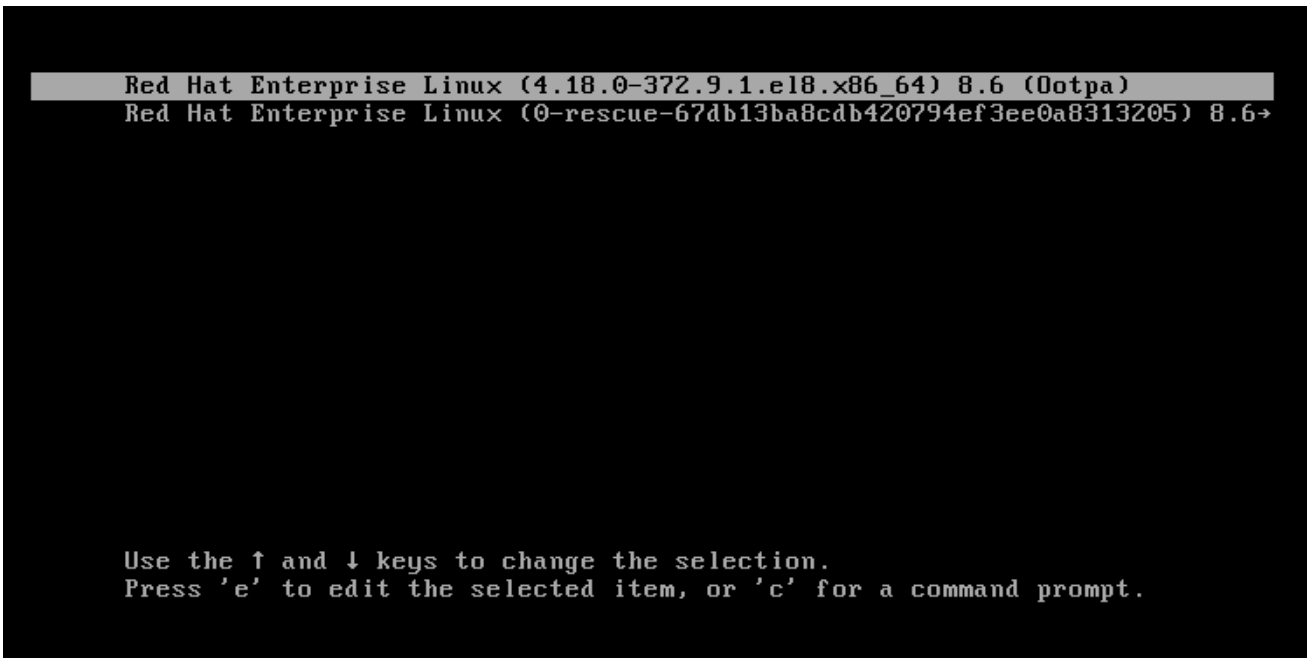
第 6 章

-
-
-
-
-
-
-

重要

6.1.

-
-


Red Hat Enterprise Linux (4.18.0-372.9.1.el8.x86_64) 8.6 (Ootpa)
Red Hat Enterprise Linux (0-rescue-67db13ba8cdb420794ef3ee0a8313205) 8.6
Use the ↑ and ↓ keys to change the selection.
Press 'e' to edit the selected item, or 'c' for a command prompt.

[D]


Minimal BASH-like line editing is supported. For the first word,
TAB lists possible command completions. Anywhere else TAB lists
possible device or file completions. ESC at any time exits.
grub>

[D]

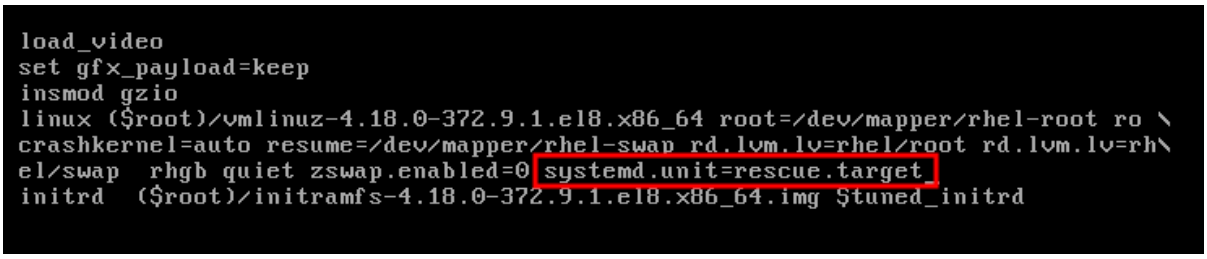
-
-

6.2.

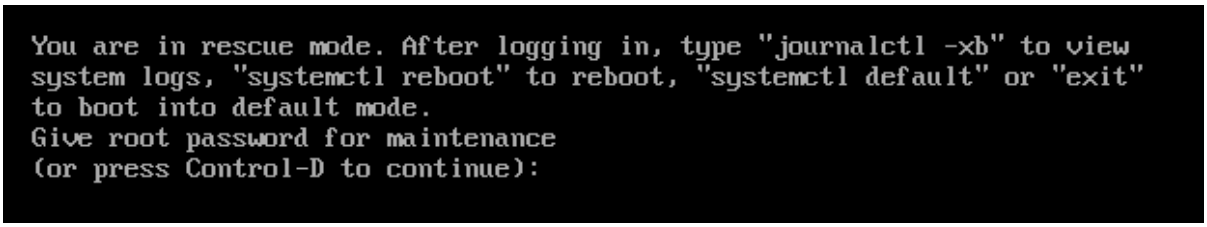
-
-
-
-
-

6.3.

1.
2. **systemd.unit=rescue.target**



[D]

3. 

[D]

6.4.

-
-
-

-

1.

2. **systemd.unit=emergency.target**

```
load_video
set gfx_payload=keep
insmod gzio
linux ($root)/vmlinuz-4.18.0-372.9.1.el8.x86_64 root=/dev/mapper/rhel-root ro \
crashkernel=auto resume=/dev/mapper/rhel-swap rd.lvm.lv=rhel/root rd.lvm.lv=rh\
el/swap  rhgb quiet zswap.enabled=0 systemd.unit=emergency.target
initrd  ($root)/initramfs-4.18.0-372.9.1.el8.x86_64.img $tuned_initrd
```

[D]

3.

```
You are in emergency mode. After logging in, type "journalctl -xb" to view
system logs, "systemctl reboot" to reboot, "systemctl default" or "exit"
to boot into default mode.
Give root password for maintenance
(or press Control-D to continue): _
```

[D]

6.5.

1.

2. **systemd.debug-shell**

```
load_video
set gfx_payload=keep
insmod gzio
linux ($root)/vmlinuz-4.18.0-372.9.1.el8.x86_64 root=/dev/mapper/rhel-root ro \
crashkernel=auto resume=/dev/mapper/rhel-swap rd.lvm.lv=rhel/root rd.lvm.lv=rh\
el/swap  rhgb quiet systemd.debug-shell
initrd  ($root)/initramfs-4.18.0-372.9.1.el8.x86_64.img $tuned_initrd
```

[D]

3. 注意

4.

警告

6.6.

-

1.

2.

sh-4.4#

- **sh-4.4# systemctl status $$**

```
sh-4.4# systemctl status $$
● debug-shell.service - Early root shell on /dev/tty9 FOR DEBUGGING ONLY
   Loaded: loaded (/usr/lib/systemd/system/debug-shell.service; enabled-runtime; vendor preset: disabled)
   Active: active (running) since Tue 2022-08-02 08:40:09 EDT; 45s ago
     Docs: man:systemd-debug-generator(8)
 Main PID: 735 (sh)
    Tasks: 3 (limit: 17257)
   Memory: 2.3M
   CGroup: /system.slice/debug-shell.service
           ├─ 735 /bin/sh
           ├─2092 systemctl status 735
           └─2093 less
sh-4.4#
```

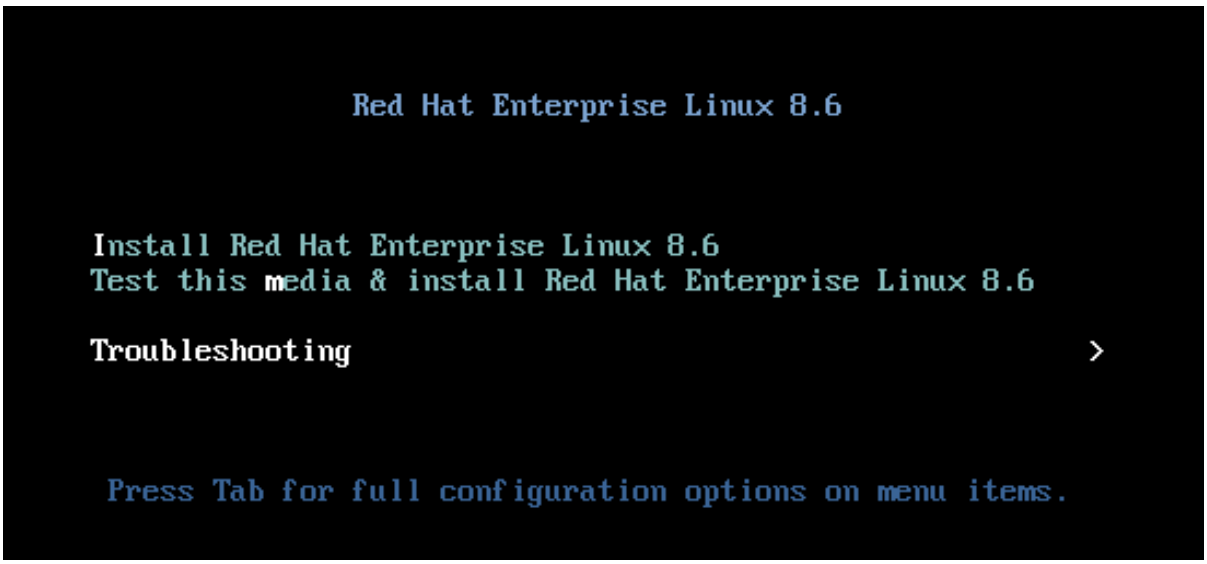
[D]

-

-

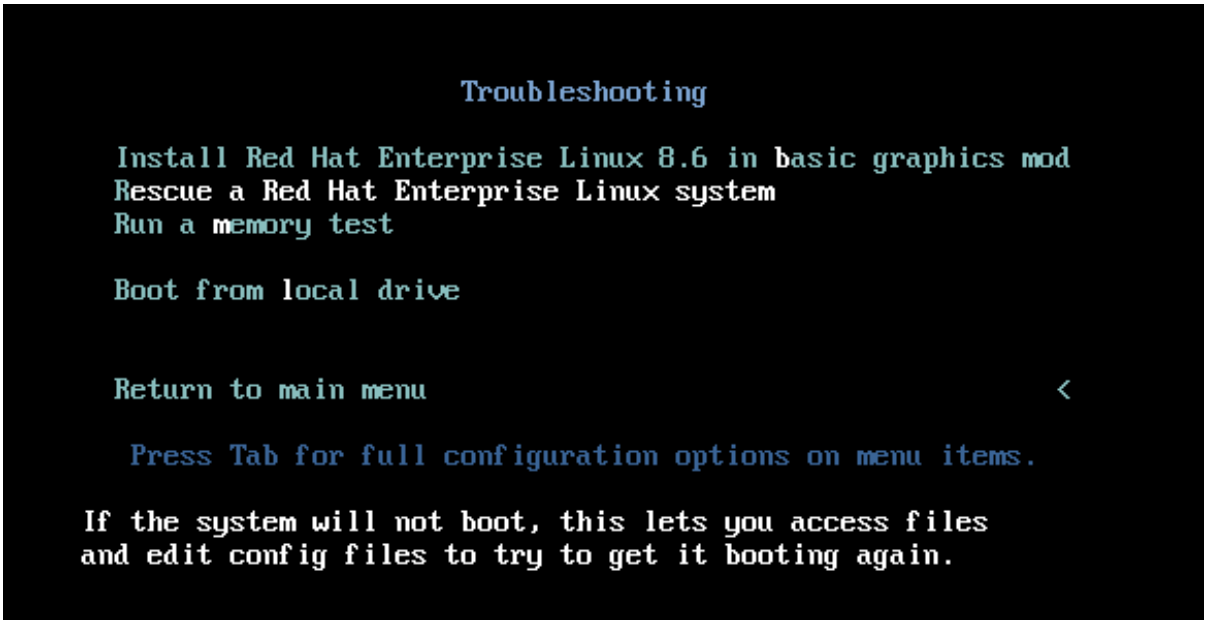
6.7.

1.

2. 

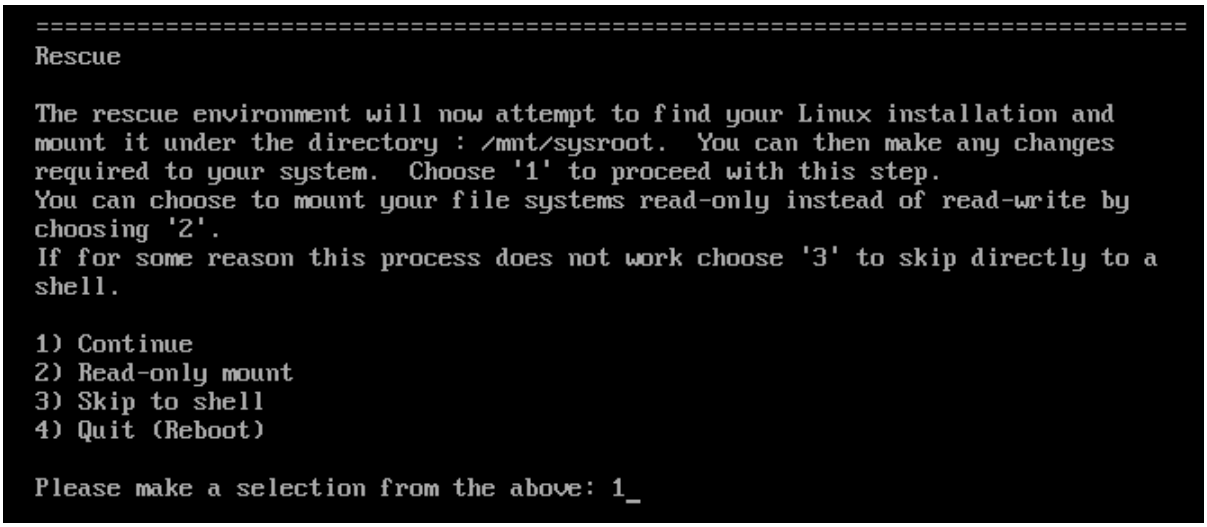
[D]

3.



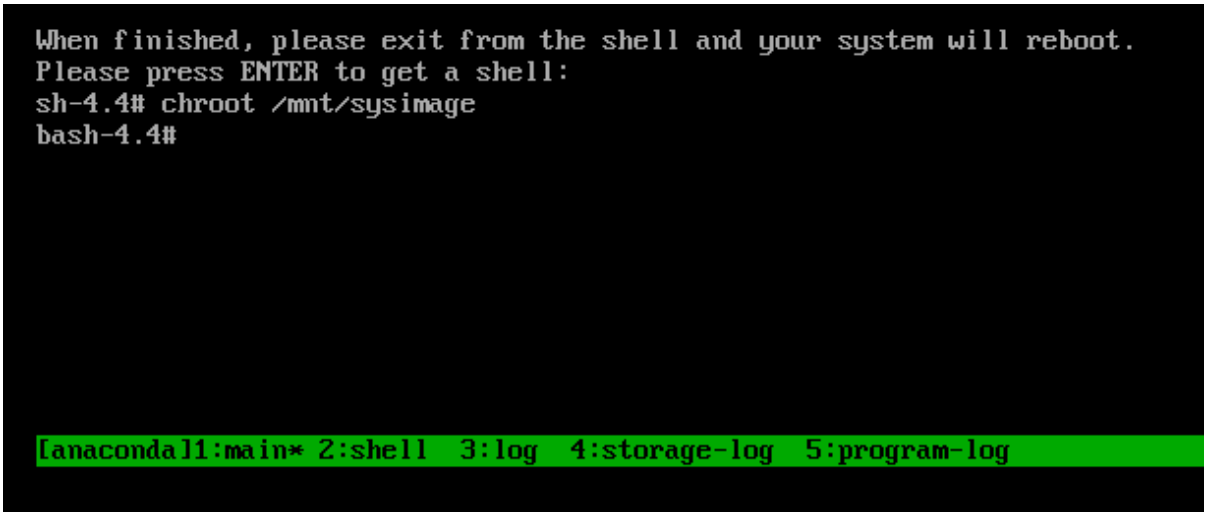
[D]

4.

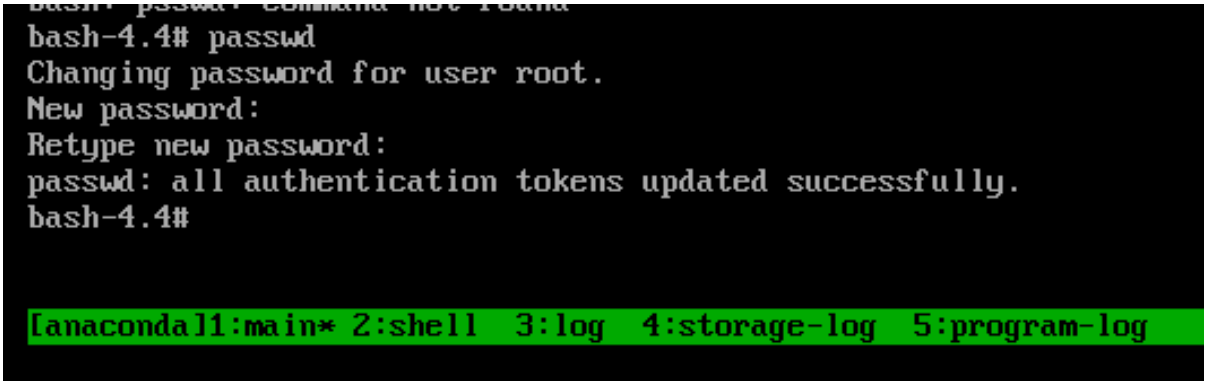


[D]

5. **sh-4.4# chroot /mnt/sysimage**



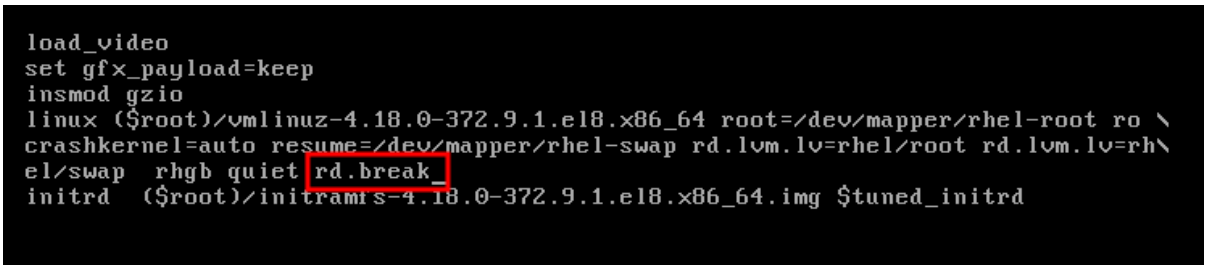
[D]

6. 

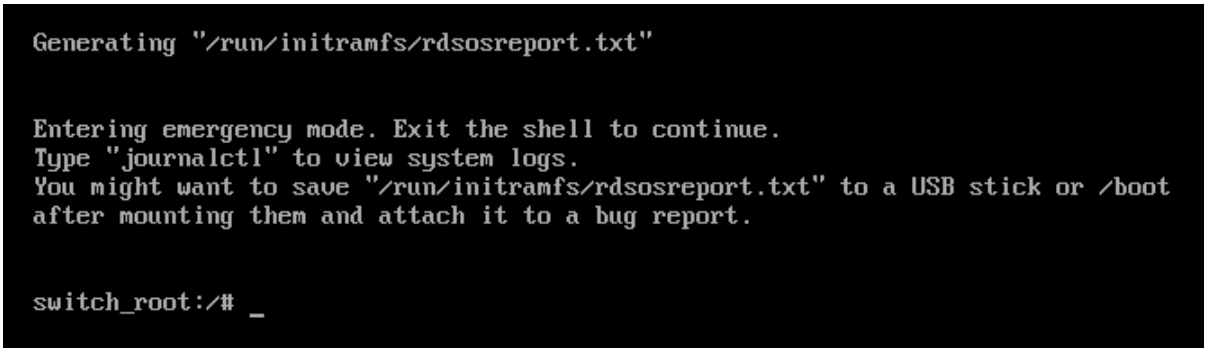
[D]

7. **sh-4.4# rm -f /.autorelabel**
8.
9.

6.8.

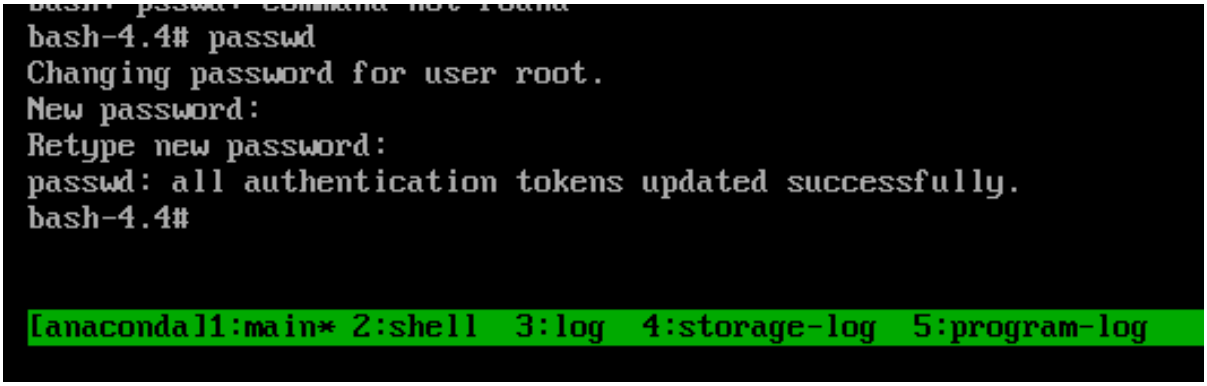
1.
2. 

[D]

3. 

[D]

4. **switch_root:/# mount -o remount,rw /sysroot**
5. **switch_root:/# chroot /sysroot**

6. 

[D]

7. **sh-4.4# touch /.autorelabel**
8. **sh-4.4# mount -o remount,ro /**
9.
10. 注意

提示

1. **rd.break enforcing=0**
2. **# restorecon /etc/shadow**
3. **# setenforce 1**
   # getenforce
   Enforcing

6.9.

-
-

第 7 章

7.1.

-
-

7.2.

-

```
# grubby --default-kernel
/boot/vmlinuz-4.18.0-372.9.1.el8.x86_64
```

-

```
# grubby --default-index
0
```

7.3.

-

```
# grubby --info=ALL
index=0
kernel="/boot/vmlinuz-4.18.0-372.9.1.el8.x86_64"
args="ro crashkernel=auto resume=/dev/mapper/rhel-swap rd.lvm.lv=rhel/root
rd.lvm.lv=rhel/swap rhgb quiet $tuned_params zswap.enabled=1"
root="/dev/mapper/rhel-root"
initrd="/boot/initramfs-4.18.0-372.9.1.el8.x86_64.img $tuned_initrd"
title="Red Hat Enterprise Linux (4.18.0-372.9.1.el8.x86_64) 8.6 (Ootpa)"
id="67db13ba8cdb420794ef3ee0a8313205-4.18.0-372.9.1.el8.x86_64"
index=1
kernel="/boot/vmlinuz-0-rescue-67db13ba8cdb420794ef3ee0a8313205"
args="ro crashkernel=auto resume=/dev/mapper/rhel-swap rd.lvm.lv=rhel/root
rd.lvm.lv=rhel/swap rhgb quiet"
root="/dev/mapper/rhel-root"
initrd="/boot/initramfs-0-rescue-67db13ba8cdb420794ef3ee0a8313205.img"
title="Red Hat Enterprise Linux (0-rescue-67db13ba8cdb420794ef3ee0a8313205) 8.6
(Ootpa)"
id="67db13ba8cdb420794ef3ee0a8313205-0-rescue"
```

-

```
# grubby --info /boot/vmlinuz-4.18.0-372.9.1.el8.x86_64
grubby --info /boot/vmlinuz-4.18.0-372.9.1.el8.x86_64
index=0
kernel="/boot/vmlinuz-4.18.0-372.9.1.el8.x86_64"
args="ro crashkernel=auto resume=/dev/mapper/rhel-swap rd.lvm.lv=rhel/root
rd.lvm.lv=rhel/swap rhgb quiet $tuned_params zswap.enabled=1"
root="/dev/mapper/rhel-root"
initrd="/boot/initramfs-4.18.0-372.9.1.el8.x86_64.img $tuned_initrd"
title="Red Hat Enterprise Linux (4.18.0-372.9.1.el8.x86_64) 8.6 (Ootpa)"
id="67db13ba8cdb420794ef3ee0a8313205-4.18.0-372.9.1.el8.x86_64"
```

注意

7.4.

- **# grubby --args=vconsole.font=latarcyrheb-sun32 --update-kernel /boot/vmlinuz-4.18.0-372.9.1.el8.x86_64**

7.5.

- **# grubby --args=console=ttyS0,115200 --update-kernel /boot/vmlinuz-4.18.0-372.9.1.el8.x86_64**
- **# grubby --remove-args="rhgb quiet" --update-kernel /boot/vmlinuz-4.18.0-372.9.1.el8.x86_64**
- **# grubby --remove-args="rhgb quiet" --args=console=ttyS0,115200 --update-kernel /boot/vmlinuz-4.18.0-372.9.1.el8.x86_64**
- **# grubby --info /boot/vmlinuz-4.18.0-372.9.1.el8.x86_64**
  index=0
  kernel="/boot/vmlinuz-4.18.0-372.9.1.el8.x86_64"
  args="ro crashkernel=auto resume=/dev/mapper/rhel-swap rd.lvm.lv=rhel/root rd.lvm.lv=rhel/swap $tuned_params zswap.enabled=1 console=ttyS0,115200"
  root="/dev/mapper/rhel-root"
  initrd="/boot/initramfs-4.18.0-372.9.1.el8.x86_64.img $tuned_initrd"
  title="Red Hat Enterprise Linux (4.18.0-372.9.1.el8.x86_64) 8.6 (Ootpa)"
  id="67db13ba8cdb420794ef3ee0a8313205-4.18.0-372.9.1.el8.x86_64"

7.6.

1. **# grubby --add-kernel=new_kernel --title="entry_title" --initrd="new_initrd" --copy-default**
2. **# ls -l /boot/loader/entries/***
   -rw-r--r--. 1 root root 408 May 27 06:18 /boot/loader/entries/67db13ba8cdb420794ef3ee0a8313205-0-rescue.conf
   -rw-r--r--. 1 root root 536 Jun 30 07:53 /boot/loader/entries/67db13ba8cdb420794ef3ee0a8313205-4.18.0-372.9.1.el8.x86_64.conf
   -rw-r--r-- 1 root root 336 Aug 15 15:12 /boot/loader/entries/d88fa2c7ff574ae782ec8c4288de4e85-4.18.0-193.el8.x86_64.conf
3. **# grubby --grub2 --add-kernel=/boot/vmlinuz-4.18.0-193.el8.x86_64 --title="Red Hat Enterprise 8 Test" --initrd=/boot/initramfs-4.18.0-193.el8.x86_64.img --copy-default**

- ```
  # ls -l /boot/loader/entries/*
  -rw-r--r--. 1 root root 408 May 27 06:18
  /boot/loader/entries/67db13ba8cdb420794ef3ee0a8313205-0-rescue.conf
  -rw-r--r--. 1 root root 536 Jun 30 07:53
  /boot/loader/entries/67db13ba8cdb420794ef3ee0a8313205-4.18.0-
  372.9.1.el8.x86_64.conf
  -rw-r--r-- 1 root root 287 Aug 16 15:17
  /boot/loader/entries/d88fa2c7ff574ae782ec8c4288de4e85-4.18.0-
  193.el8.x86_64.0~custom.conf
  -rw-r--r--  1 root root 287 Aug 16 15:29
  /boot/loader/entries/d88fa2c7ff574ae782ec8c4288de4e85-4.18.0-193.el8.x86_64.conf
  ```

7.7.

-

```
# grubby --set-default /boot/vmlinuz-4.18.0-372.9.1.el8.x86_64
The default is /boot/loader/entries/67db13ba8cdb420794ef3ee0a8313205-4.18.0-
372.9.1.el8.x86_64.conf with index 0 and kernel /boot/vmlinuz-4.18.0-372.9.1.el8.x86_64
```

7.8.

- ```
  # grubby --update-kernel=ALL --args=console=ttyS0,115200
  ```

  注意

7.9.

1. ```
   # grub2-editenv - list | grep kernelopts
   kernelopts=root=/dev/mapper/rhel-root ro crashkernel=auto resume=/dev/mapper/rhel-
   swap rd.lvm.lv=rhel/root rd.lvm.lv=rhel/swap rhgb quiet
   ```

2. ```
   # grub2-editenv - set "$(grub2-editenv - list | grep kernelopts) <debug>"
   ```

3. ```
   # grub2-editenv - list | grep kernelopts
   kernelopts=root=/dev/mapper/rhel-root ro crashkernel=auto resume=/dev/mapper/rhel-
   swap rd.lvm.lv=rhel/root rd.lvm.lv=rhel/swap rhgb quiet debug
   ```

4.

注意

```
# grubby --update-kernel ALL --args="<PARAMETER>"
```

7.10.
```

-
-

第 8 章

8.1.

-
-
-
-
-

8.2.

1. GRUB_TIMEOUT_STYLE=hidden
2. - **# grub2-mkconfig -o /boot/grub2/grub.cfg**
   - **# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg**
3.

重要

8.3.

1. **# grubby --set-default-index=1**
   The default is /boot/loader/entries/d5151aa93c444ac89e78347a1504d6c6-4.18.0-348.el8.x86_64.conf with index 1 and kernel /boot/vmlinuz-4.18.0-348.el8.x86_64

   注意

   注意

2. a. **# grubby --info=ALL**

   b. GRUB_DEFAULT=example-gnu-linux

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

第 9 章

-
-
-

注意

9.1.

重要

1. **# grub2-install */dev/sda***
2. **# reboot**

-

9.2.

重要

1. **# yum reinstall grub2-efi shim**
2. **# reboot**

9.3.

重要

1. # bootlist -m normal -o *sda1*
2. # grub2-install *partition*
3. # reboot

-

9.4.

重要

1. ```
   # rm /etc/grub.d/*
   # rm /etc/sysconfig/grub
   ```

2. - ```
     # yum reinstall grub2-tools
     ```
   - ```
     # yum reinstall grub2-efi shim grub2-tools grub2-common
     ```

3. - ```
     # grub2-mkconfig -o /boot/grub2/grub.cfg
     ```
   - ```
     # grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
     ```

4.
     ```

第 10 章

-
-

10.1.

重要

1. **# grub2-setpassword**

2. **Enter password:**
   Confirm the password:

注意

10.2.

警告

1.

2.

3.

```
title Red Hat Enterprise Linux (4.18.0-221.el8.x86_64) 8.3
(Ootpa)
version 4.18.0-221.el8.x86_64
linux /vmlinuz-4.18.0-221.el8.x86_64
initrd /initramfs-4.18.0-221.el8.x86_64.img $tuned_initrd
options $kernelopts $tuned_params
id rhel-20200625210904-4.18.0-221.el8.x86_64
grub_users root
grub_arg --unrestricted
grub_class kernel
```

1.

注意

```
# grub2-editenv - set grub_users="root"
```

第 11 章

11.1.

-

11.2.

-
-

11.3.

 重要

-

第 12 章

12.1.

-
-
-
-
-
-
-
-
-
-

12.2.

注意

注意

注意

-

第 13 章

13.1.

-

13.2.

```
# sysctl kernel.printk
kernel.printk = 7 4 1 7
```

1.
2.
3.
4.

重要

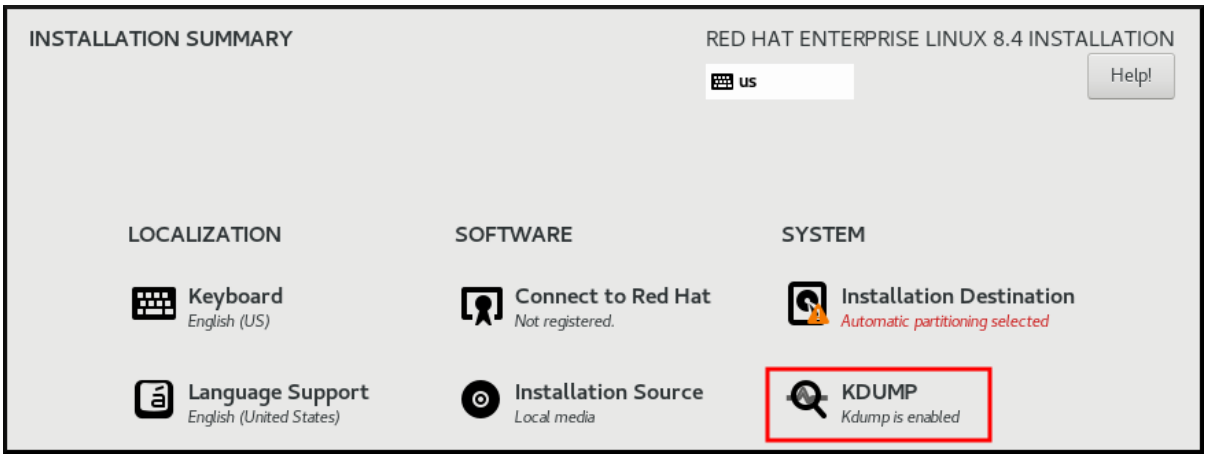
-

第 14 章

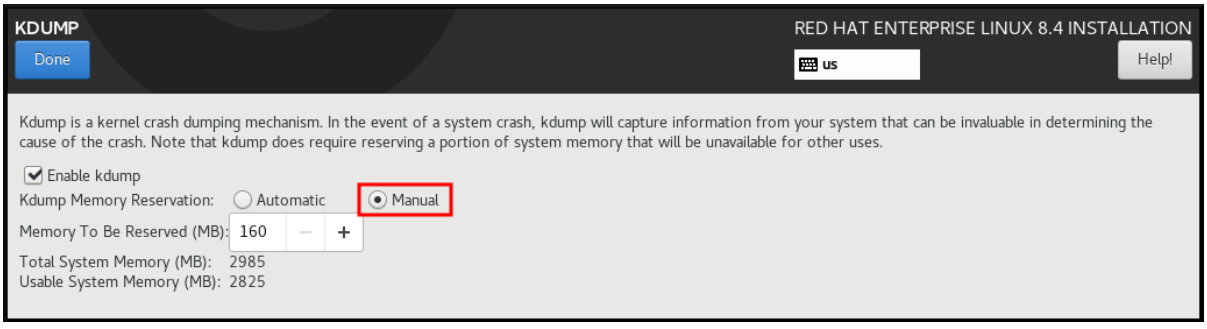
14.1.

重要

14.2.

1. 

[D]

2.
3. 

[D]

14.3.

-
-
-

1. **# rpm -q kexec-tools**

   kexec-tools-2.0.17-11.el8.x86_64

package kexec-tools is not installed

2. **# dnf install kexec-tools**

重要

第 15 章

15.1.

- ```
  # 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

15.2.

注意

-
-

1. - crashkernel=128M
   - crashkernel=512M-2G:64M,2G-:128M
   - crashkernel=128M@16M

     crashkernel=512M-2G:64M,2G-:128M@16M
2. ```
   # grubby --update-kernel=ALL --args="crashkernel=<value>"
   ```

-
-
-
-
-

15.3.

-
-
- **path /var/crash**

  注意

  -
  -

- **kdump_post <path_to_kdump_post.sh>**
-

```
# *grep -v ^# /etc/kdump.conf | grep -v ^$*
ext4 /dev/mapper/vg00-varcrashvol
path /var/crash
core_collector makedumpfile -c --message-level 1 -d 31
```

- a.

  b. **path /usr/local/cores**

  重要

- a.

  -
  -
  -

  b. **ext4 UUID=03138356-5e61-4ab3-b58e-27507ac41937**

  重要

- a.

  b. **raw /dev/sdb1**

- a.

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

c. sudo systemctl restart kdump.service

注意

- a.

  b.

  c. i.

  ii. **ssh john@penguin.example.com**
  sshkey /root/.ssh/mykey

15.4.

-
-
-

注意

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

-
-
-
-
-

1.
2.

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

- core_collector makedumpfile -c -d 31 --message-level 1
- core_collector makedumpfile -p -d 31 --message-level 1
-
-

15.5.

-
-

1.
2. failure_action poweroff

-

15.6.

```
# KDUMP_COMMANDLINE_REMOVE="hugepages hugepagesz slub_debug quiet log_buf_len
swiotlb"
```

```
KDUMP_COMMANDLINE_APPEND="cgroup_disable=memory"
```

-
-

15.7.

警告

-
-
-
-
-
-
-

-
-

1. **# kdumpctl restart**
2. # kdumpctl status
   kdump:Kdump is operational
3. **# echo c > /proc/sysrq-trigger**

-

15.8.

-
-
-
-

15.9.

-
-
-

1. # systemctl enable kdump.service
2. # systemctl start kdump.service
3. # systemctl stop kdump.service
4. # systemctl disable kdump.service

警告

-

15.10.

-
-
-

1. $ lsmod

   ```
   Module                  Size  Used by
   fuse                  126976  3
   xt_CHECKSUM            16384  1
   ipt_MASQUERADE         16384  1
   uinput                 20480  1
   xt_conntrack           16384  1
   ```

2. KDUMP_COMMANDLINE_APPEND="rd.driver.blacklist=*hv_vmbus,hv_storvsc,hv_utils,hv_net vsc,hid-hyperv*"

   KDUMP_COMMANDLINE_APPEND="modprobe.blacklist=*emcp* modprobe.blacklist=*bnx2fc* modprobe.blacklist=*libfcoe* modprobe.blacklist=*fcoe*"

3. # systemctl restart kdump

-

15.11.

1. # *kdumpctl estimate*

   ```
   Encrypted kdump target requires extra memory, assuming using the keyslot with minimum memory requirement
     Reserved crashkernel:    256M
     Recommended crashkernel: 652M

     Kernel image size:   47M
     Kernel modules size: 8M
     Initramfs size:      20M
     Runtime reservation: 64M
     LUKS required size:  512M
     Large modules: <none>
     WARNING: Current crashkernel size is lower than recommended size 652M.
   ```

2.
3.

注意

第 16 章

16.1.

-

1.
2.
```
$ sudo grubby --update-kernel ALL --args crashkernel=512M
```
3.
4.
   -
     -
     -
     -
   -
     -
     -
     -  注意
5.
6.

1.

   2. 警告

-

第 17 章

17.1.

-

1. # grubby --update-kernel=ALL --args="crashkernel=xxM"
2. # reboot
3. # systemctl enable --now kdump.service

- # systemctl status kdump.service

○ kdump.service - Crash recovery kernel arming
Loaded: loaded (/usr/lib/systemd/system/kdump.service; enabled; vendor preset: disabled)
Active: active (live)

17.2.

-

1. # ls -a /boot/vmlinuz-*
/boot/vmlinuz-0-rescue-2930657cd0dc43c2b75db480e5e5b4a9
/boot/vmlinuz-4.18.0-330.el8.x86_64
/boot/vmlinuz-4.18.0-330.rt7.111.el8.x86_64
2. # grubby --update-kernel=*vmlinuz-4.18.0-330.el8.x86_64* --args="crashkernel=*xxM*"
3. # systemctl enable --now kdump.service

- # systemctl status kdump.service

○ kdump.service - Crash recovery kernel arming
Loaded: loaded (/usr/lib/systemd/system/kdump.service; enabled; vendor preset: disabled)
Active: active (live)

17.3.

-
-

1. # systemctl stop kdump.service
2. # systemctl disable kdump.service

警告

-

-

第 18 章

-
-
-

18.1.

```
$ uname -m
```

表 18.1.

重要

-
-
-

18.2.

表 18.2.

注意

18.3.

表 18.3.

	• • • • • • • • •	• • • • • • • • •
	• • • • • • • •	• • • •
	• • • •	

	• •	

-

18.4.

表 18.4.

-

18.5.

表 18.5.

-

18.6.

1.
   -
   -
   -

2. # kdumpctl restart

18.7.

1.
   -
   -
   -
   -
2. # kdumpctl restart

第 19 章

19.1.

注意

19.2.

警告

```
error: ../../grub-core/kern/mm.c:376:out of memory.
Press any key to continue…
```

1.
2. ```
   # grubby --update-kernel=ALL --args="fadump=on"
   ```
3. ```
   # grubby --update-kernel=ALL --args="crashkernel=xxM fadump=on"
   ```

重要

19.3.

-
-
-
-
-
-
-
-
-

19.4.

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

警告

2. failure_action shell

第 20 章

20.1.

1. # subscription-manager repos --enable *baseos repository*

   # subscription-manager repos --enable *appstream repository*

   # subscription-manager repos --enable *rhel-8-for-x86_64-baseos-debug-rpms*

2. # yum install crash

3. # yum install kernel-debuginfo

20.2.

-

1. -

   - # crash /usr/lib/debug/lib/modules/4.18.0-5.el8.x86_64/vmlinux /var/crash/127.0.0.1-2018-10-06-14:05:33/vmcore

     例 20.1.

     ```
     ...
     WARNING: kernel relocated [202MB]: patching 90160 gdb minimal_symbol values

           KERNEL: /usr/lib/debug/lib/modules/4.18.0-5.el8.x86_64/vmlinux
         DUMPFILE: /var/crash/127.0.0.1-2018-10-06-14:05:33/vmcore  [PARTIAL DUMP]
             CPUS: 2
             DATE: Sat Oct  6 14:05:16 2018
           UPTIME: 01:03:57
     LOAD AVERAGE: 0.00, 0.00, 0.00
            TASKS: 586
         NODENAME: localhost.localdomain
          RELEASE: 4.18.0-5.el8.x86_64
          VERSION: #1 SMP Wed Aug 29 11:51:55 UTC 2018
          MACHINE: x86_64  (2904 Mhz)
           MEMORY: 2.9 GB
            PANIC: "sysrq: SysRq : Trigger a crash"
              PID: 10635
          COMMAND: "bash"
             TASK: ffff8d6c84271800  [THREAD_INFO: ffff8d6c84271800]
              CPU: 1
            STATE: TASK_RUNNING (SYSRQ)

     crash>
     ```

2. crash> exit
~]#

注意

-

20.3.

-
```
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
```

注意

-
```
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: ffffffda  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
```

- ```
  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
  ```

- ```
  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 ...
  ```

- ```
  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
  ```

20.4.

-

1.
2. -
3.

-

20.5.

-
  ```

第 21 章

21.1.

-
-
-

1. **# systemctl is-enabled kdump.service && systemctl is-active kdump.service**
   enabled
   active
2. **# dracut -f --add earlykdump**
3. **# grubby --update-kernel=/boot/vmlinuz-$(uname -r) --args="rd.earlykdump"**
4. # reboot

- **# cat /proc/cmdline**
  BOOT_IMAGE=(hd0,msdos1)/vmlinuz-4.18.0-187.el8.x86_64 root=/dev/mapper/rhel-root ro crashkernel=auto resume=/dev/mapper/rhel-swap rd.lvm.lv=rhel/root rd.lvm.lv=rhel/swap rhgb quiet rd.earlykdump

  # journalctl -x | grep early-kdump
  Mar 20 15:44:41 redhat dracut-cmdline[304]: early-kdump is enabled.
  Mar 20 15:44:42 redhat dracut-cmdline[304]: kexec: loaded early-kdump kernel
-
-

第 22 章

-
-
-

警告

注意

-
-

22.1.

-
-

22.2.

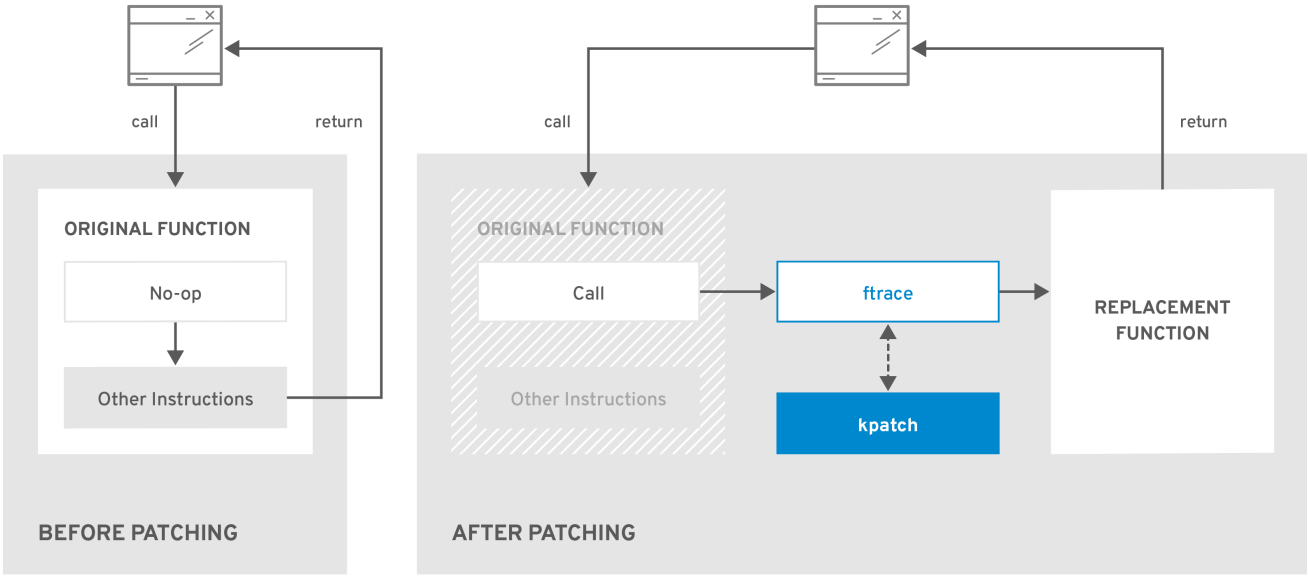
22.3.

-
-
-
-
-

22.4.

1.
2.

图 22.1.



[D]

22.5.

警告

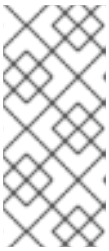
-

1. **# uname -r**
   4.18.0-94.el8.x86_64
2. **# yum search $(uname -r)**
3. **# yum install "kpatch-patch = $(uname -r)"**

注意

- **# kpatch list**
  Loaded patch modules:
  kpatch_4_18_0_94_1_1 [enabled]

  Installed patch modules:
  kpatch_4_18_0_94_1_1 (4.18.0-94.el8.x86_64)
  …



注意

```
# rpm -qa | grep kpatch
kpatch-patch-4_18_0-477_21_1-0-0.el8_8.x86_64
kpatch-dnf-0.9.7_0.4-2.el8.noarch
kpatch-0.9.7-2.el8.noarch
```

-

22.6.

-

1. ```
   # yum list installed | grep kernel
   Updating Subscription Management repositories.
   Installed Packages
   ...
   kernel-core.x86_64        4.18.0-240.10.1.el8_3        @rhel-8-for-x86_64-baseos-rpms
   kernel-core.x86_64        4.18.0-240.15.1.el8_3        @rhel-8-for-x86_64-baseos-rpms
   ...

   # uname -r
   4.18.0-240.10.1.el8_3.x86_64
   ```

2. ```
   # yum install kpatch-dnf
   ```

3. ```
   # yum kpatch auto
   Updating Subscription Management repositories.
   Last metadata expiration check: 19:10:26 ago on Wed 10 Mar 2021 04:08:06 PM CET.
   Dependencies resolved.
   ================================================
    Package                    Architecture
   ================================================
   Installing:
    kpatch-patch-4_18_0-240_10_1       x86_64
    kpatch-patch-4_18_0-240_15_1       x86_64

   Transaction Summary
   ================================================
   Install  2 Packages
   …
   ```

注意

-

```
# kpatch list
Loaded patch modules:
kpatch_4_18_0_240_10_1_0_1 [enabled]
```
```

```
Installed patch modules:
kpatch_4_18_0_240_10_1_0_1 (4.18.0-240.10.1.el8_3.x86_64)
kpatch_4_18_0_240_15_1_0_2 (4.18.0-240.15.1.el8_3.x86_64)
```

注意

```
# rpm -qa | grep kpatch
kpatch-patch-4_18_0-477_21_1-0-0.el8_8.x86_64
kpatch-dnf-0.9.7_0.4-2.el8.noarch
kpatch-0.9.7-2.el8.noarch
```

-

22.7.

-

1. ```
   # yum list installed | grep kernel
   Updating Subscription Management repositories.
   Installed Packages
   ...
   kernel-core.x86_64      4.18.0-240.10.1.el8_3      @rhel-8-for-x86_64-baseos-rpms
   kernel-core.x86_64      4.18.0-240.15.1.el8_3      @rhel-8-for-x86_64-baseos-rpms
   ...

   # uname -r
   4.18.0-240.10.1.el8_3.x86_64
   ```

2. ```
   # yum kpatch manual
   Updating Subscription Management repositories.
   ```

- ```
  # yum kpatch status
  ...
  Updating Subscription Management repositories.
  Last metadata expiration check: 0:30:41 ago on Tue Jun 14 15:59:26 2022.
  Kpatch update setting: manual
  ```

-

22.8.

-

- ```
  # yum update "kpatch-patch = $(uname -r)"
  ```

- ```
  # yum update "kpatch-patch"
  ```
  ```

注意

-

22.9.

-
-

1. **# yum list installed | grep kpatch-patch**
   kpatch-patch-4_18_0-94.x86_64 1-1.el8 @@commandline
   …
2. **# yum remove kpatch-patch-4_18_0-94.x86_64**
3.
4. **# yum list installed | grep kpatch-patch**

1. **# kpatch list**
   Loaded patch modules:

重要

-
-

22.10.

-
-
-

1. **# kpatch list**
   Loaded patch modules:
   kpatch_4_18_0_94_1_1 [enabled]

   Installed patch modules:
   kpatch_4_18_0_94_1_1 (4.18.0-94.el8.x86_64)
   …
2. **# kpatch uninstall kpatch_4_18_0_94_1_1**
   uninstalling kpatch_4_18_0_94_1_1 (4.18.0-94.el8.x86_64)

- ```
  # kpatch list
  Loaded patch modules:
  kpatch_4_18_0_94_1_1 [enabled]

  Installed patch modules:
  <NO_RESULT>
  ```

3.

1. ```
   # kpatch list
   Loaded patch modules:
   ...
   ```

-

22.11.

-
-
-

1. ```
   # systemctl is-enabled kpatch.service
   enabled
   ```

2. ```
   # systemctl disable kpatch.service
   Removed /etc/systemd/system/multi-user.target.wants/kpatch.service.
   ```

   - ```
     # kpatch list
     Loaded patch modules:
     kpatch_4_18_0_94_1_1 [enabled]

     Installed patch modules:
     kpatch_4_18_0_94_1_1 (4.18.0-94.el8.x86_64)
     ```

3.

4. ```
   # systemctl status kpatch.service
   ● kpatch.service - "Apply kpatch kernel patches"
     Loaded: loaded (/usr/lib/systemd/system/kpatch.service; disabled; vendor preset:
   disabled)
     Active: inactive (dead)
   ```

5. ```
   # kpatch list
   Loaded patch modules:
   <NO_RESULT>
   ```
   ```

```
Installed patch modules:
kpatch_4_18_0_94_1_1 (4.18.0-94.el8.x86_64)
```

重要

-
-

第 23 章

-
-
-

23.1.

-
-
-
-

注意

-
-
-

23.2.

重要

-
-

23.3.

表 23.1.

-

23.4.

-
-

```
# mount -l | grep cgroup
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,seclabel,mode=755)
cgroup on /sys/fs/cgroup/systemd type cgroup
(rw,nosuid,nodev,noexec,relatime,seclabel,xattr,release_agent=/usr/lib/systemd/systemd-cgroups-agent,name=systemd)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup
(rw,nosuid,nodev,noexec,relatime,seclabel,cpu,cpuacct)
cgroup on /sys/fs/cgroup/perf_event type cgroup
(rw,nosuid,nodev,noexec,relatime,seclabel,perf_event)
cgroup on /sys/fs/cgroup/pids type cgroup
(rw,nosuid,nodev,noexec,relatime,seclabel,pids)
...
```

1.
```
# top
top - 11:34:09 up 11 min,  1 user,  load average: 0.51, 0.27, 0.22
Tasks: 267 total,   3 running, 264 sleeping,   0 stopped,   0 zombie
%Cpu(s): 49.0 us,  3.3 sy,  0.0 ni, 47.5 id,  0.0 wa,  0.2 hi,  0.0 si,  0.0 st
MiB Mem :   1826.8 total,    303.4 free,   1046.8 used,    476.5 buff/cache
MiB Swap:   1536.0 total,   1396.0 free,    140.0 used.    616.4 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 6955 root      20   0  228440   1752   1472 R  99.3   0.1   0:32.71 sha1sum
 5760 jdoe      20   0 3603868 205188  64196 S   3.7  11.0   0:17.19 gnome-shell
 6448 jdoe      20   0  743648  30640  19488 S   0.7   1.6   0:02.73 gnome-terminal-
    1 root      20   0  245300   6568   4116 S   0.3   0.4   0:01.87 systemd
  505 root      20   0       0      0      0 I   0.3   0.0   0:00.75 kworker/u4:4-events_unbound
...
```

2.
```
# mkdir /sys/fs/cgroup/cpu/Example/
```

3.
```
# ll /sys/fs/cgroup/cpu/Example/
```

```
-rw-r—r--. 1 root root 0 Mar 11 11:42 cgroup.clone_children
-rw-r—r--. 1 root root 0 Mar 11 11:42 cgroup.procs
-r—r—r--. 1 root root 0 Mar 11 11:42 cpuacct.stat
-rw-r—r--. 1 root root 0 Mar 11 11:42 cpuacct.usage
-r—r—r--. 1 root root 0 Mar 11 11:42 cpuacct.usage_all
-r—r—r--. 1 root root 0 Mar 11 11:42 cpuacct.usage_percpu
-r—r—r--. 1 root root 0 Mar 11 11:42 cpuacct.usage_percpu_sys
-r—r—r--. 1 root root 0 Mar 11 11:42 cpuacct.usage_percpu_user
-r—r—r--. 1 root root 0 Mar 11 11:42 cpuacct.usage_sys
-r—r—r--. 1 root root 0 Mar 11 11:42 cpuacct.usage_user
-rw-r—r--. 1 root root 0 Mar 11 11:42 cpu.cfs_period_us
-rw-r—r--. 1 root root 0 Mar 11 11:42 cpu.cfs_quota_us
-rw-r—r--. 1 root root 0 Mar 11 11:42 cpu.rt_period_us
-rw-r—r--. 1 root root 0 Mar 11 11:42 cpu.rt_runtime_us
-rw-r—r--. 1 root root 0 Mar 11 11:42 cpu.shares
-r—r—r--. 1 root root 0 Mar 11 11:42 cpu.stat
-rw-r—r--. 1 root root 0 Mar 11 11:42 notify_on_release
-rw-r—r--. 1 root root 0 Mar 11 11:42 tasks
```

4.
```
# echo "1000000" > /sys/fs/cgroup/cpu/Example/cpu.cfs_period_us
# echo "200000" > /sys/fs/cgroup/cpu/Example/cpu.cfs_quota_us
```

-
-

5.
```
# cat /sys/fs/cgroup/cpu/Example/cpu.cfs_period_us
/sys/fs/cgroup/cpu/Example/cpu.cfs_quota_us
1000000
200000
```

6.
```
# echo "6955" > /sys/fs/cgroup/cpu/Example/cgroup.procs
```

1.
```
# cat /proc/6955/cgroup
12:cpuset:/
11:hugetlb:/
10:net_cls,net_prio:/
9:memory:/user.slice/user-1000.slice/user@1000.service
8:devices:/user.slice
7:blkio:/
6:freezer:/
5:rdma:/
4:pids:/user.slice/user-1000.slice/user@1000.service
3:perf_event:/
2:cpu,cpuacct:/Example
1:name=systemd:/user.slice/user-1000.slice/user@1000.service/gnome-terminal-
server.service
```

2.
```
# top
top - 12:28:42 up  1:06,  1 user,  load average: 1.02, 1.02, 1.00
Tasks: 266 total,   6 running, 260 sleeping,   0 stopped,   0 zombie
%Cpu(s): 11.0 us,  1.2 sy,  0.0 ni, 87.5 id,  0.0 wa,  0.2 hi,  0.0 si,  0.2 st
MiB Mem :   1826.8 total,    287.1 free,   1054.4 used,    485.3 buff/cache
```

```
MiB Swap:   1536.0 total,   1396.7 free,    139.2 used.    608.3 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 6955 root      20   0  228440   1752   1472 R  20.6   0.1  47:11.43 sha1sum
 5760 jdoe      20   0 3604956 208832  65316 R   2.3  11.2   0:43.50 gnome-shell
 6448 jdoe      20   0  743836  31736  19488 S   0.7   1.7   0:08.25 gnome-terminal-
  505 root      20   0       0      0      0 I   0.3   0.0   0:03.39 kworker/u4:4-events_unbound
 4217 root      20   0   74192   1612   1320 S   0.3   0.1   0:01.19 spice-vdagentd
...
```

注意

-
-

第 24 章

-
-

24.1.

-

1. **# grubby --update-kernel=/boot/vmlinuz-$(uname -r) --args="systemd.unified_cgroup_hierarchy=1"**

   # grubby --update-kernel=ALL --args="systemd.unified_cgroup_hierarchy=1"

2.

1. **# mount -l | grep cgroup**
   cgroup2 on /sys/fs/cgroup type cgroup2 (rw,nosuid,nodev,noexec,relatime,seclabel,nsdelegate)

2. **# ll /sys/fs/cgroup/**
   -r—r—r--. 1 root root 0 Apr 29 12:03 cgroup.controllers
   -rw-r—r--. 1 root root 0 Apr 29 12:03 cgroup.max.depth
   -rw-r—r--. 1 root root 0 Apr 29 12:03 cgroup.max.descendants
   -rw-r—r--. 1 root root 0 Apr 29 12:03 cgroup.procs
   -r—r—r--. 1 root root 0 Apr 29 12:03 cgroup.stat
   -rw-r—r--. 1 root root 0 Apr 29 12:18 cgroup.subtree_control
   -rw-r—r--. 1 root root 0 Apr 29 12:03 cgroup.threads
   -rw-r—r--. 1 root root 0 Apr 29 12:03 cpu.pressure
   -r—r—r--. 1 root root 0 Apr 29 12:03 cpuset.cpus.effective
   -r—r—r--. 1 root root 0 Apr 29 12:03 cpuset.mems.effective
   -r—r—r--. 1 root root 0 Apr 29 12:03 cpu.stat
   drwxr-xr-x. 2 root root 0 Apr 29 12:03 init.scope
   -rw-r—r--. 1 root root 0 Apr 29 12:03 io.pressure
   -r—r—r--. 1 root root 0 Apr 29 12:03 io.stat
   -rw-r—r--. 1 root root 0 Apr 29 12:03 memory.pressure
   -r—r—r--. 1 root root 0 Apr 29 12:03 memory.stat
   drwxr-xr-x. 69 root root 0 Apr 29 12:03 system.slice
   drwxr-xr-x. 3 root root 0 Apr 29 12:18 user.slice

-

24.2.

-
-

-

1.
```
# top
Tasks: 104 total,   3 running, 101 sleeping,   0 stopped,   0 zombie
%Cpu(s): 17.6 us, 81.6 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.8 hi,  0.0 si,  0.0 st
MiB Mem :   3737.4 total,   3312.7 free,    133.3 used,    291.4 buff/cache
MiB Swap:   4060.0 total,   4060.0 free,      0.0 used.   3376.1 avail Mem

   PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 34578 root      20   0   18720   1756   1468 R  99.0   0.0   0:31.09 sha1sum
 34579 root      20   0   18720   1772   1480 R  99.0   0.0   0:30.54 sha1sum
     1 root      20   0  186192  13940   9500 S   0.0   0.4   0:01.60 systemd
     2 root      20   0       0      0      0 S   0.0   0.0   0:00.01 kthreadd
     3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
     4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
...
```

2.
```
# cat /sys/fs/cgroup/cgroup.controllers
cpuset cpu io memory hugetlb pids rdma
```

3.
```
# echo "+cpu" >> /sys/fs/cgroup/cgroup.subtree_control
# echo "+cpuset" >> /sys/fs/cgroup/cgroup.subtree_control
```

注意

4.
```
# mkdir /sys/fs/cgroup/Example/
```

5.
```
# ll /sys/fs/cgroup/Example/
-r—r—r--. 1 root root 0 Jun  1 10:33 cgroup.controllers
-r—r—r--. 1 root root 0 Jun  1 10:33 cgroup.events
-rw-r—r--. 1 root root 0 Jun  1 10:33 cgroup.freeze
-rw-r—r--. 1 root root 0 Jun  1 10:33 cgroup.max.depth
-rw-r—r--. 1 root root 0 Jun  1 10:33 cgroup.max.descendants
-rw-r—r--. 1 root root 0 Jun  1 10:33 cgroup.procs
-r—r—r--. 1 root root 0 Jun  1 10:33 cgroup.stat
-rw-r—r--. 1 root root 0 Jun  1 10:33 cgroup.subtree_control
…
-rw-r—r--. 1 root root 0 Jun  1 10:33 cpuset.cpus
-r—r—r--. 1 root root 0 Jun  1 10:33 cpuset.cpus.effective
-rw-r—r--. 1 root root 0 Jun  1 10:33 cpuset.cpus.partition
-rw-r—r--. 1 root root 0 Jun  1 10:33 cpuset.mems
-r—r—r--. 1 root root 0 Jun  1 10:33 cpuset.mems.effective
-r—r—r--. 1 root root 0 Jun  1 10:33 cpu.stat
-rw-r—r--. 1 root root 0 Jun  1 10:33 cpu.weight
-rw-r—r--. 1 root root 0 Jun  1 10:33 cpu.weight.nice
…
-r—r—r--. 1 root root 0 Jun  1 10:33 memory.events.local
-rw-r—r--. 1 root root 0 Jun  1 10:33 memory.high
-rw-r—r--. 1 root root 0 Jun  1 10:33 memory.low
…
-r—r—r--. 1 root root 0 Jun  1 10:33 pids.current
-r—r—r--. 1 root root 0 Jun  1 10:33 pids.events
```

```
-rw-r—r--. 1 root root 0 Jun  1 10:33 pids.max
```

6. ```
   # echo "+cpu" >> /sys/fs/cgroup/Example/cgroup.subtree_control
   # echo "+cpuset" >> /sys/fs/cgroup/Example/cgroup.subtree_control
   ```

7. ```
   # mkdir /sys/fs/cgroup/Example/tasks/
   ```

8. ```
   # ll /sys/fs/cgroup/Example/tasks
   -r—r—r--. 1 root root 0 Jun  1 11:45 cgroup.controllers
   -r—r—r--. 1 root root 0 Jun  1 11:45 cgroup.events
   -rw-r—r--. 1 root root 0 Jun  1 11:45 cgroup.freeze
   -rw-r—r--. 1 root root 0 Jun  1 11:45 cgroup.max.depth
   -rw-r—r--. 1 root root 0 Jun  1 11:45 cgroup.max.descendants
   -rw-r—r--. 1 root root 0 Jun  1 11:45 cgroup.procs
   -r—r—r--. 1 root root 0 Jun  1 11:45 cgroup.stat
   -rw-r—r--. 1 root root 0 Jun  1 11:45 cgroup.subtree_control
   -rw-r—r--. 1 root root 0 Jun  1 11:45 cgroup.threads
   -rw-r—r--. 1 root root 0 Jun  1 11:45 cgroup.type
   -rw-r—r--. 1 root root 0 Jun  1 11:45 cpu.max
   -rw-r—r--. 1 root root 0 Jun  1 11:45 cpu.pressure
   -rw-r—r--. 1 root root 0 Jun  1 11:45 cpuset.cpus
   -r—r—r--. 1 root root 0 Jun  1 11:45 cpuset.cpus.effective
   -rw-r—r--. 1 root root 0 Jun  1 11:45 cpuset.cpus.partition
   -rw-r—r--. 1 root root 0 Jun  1 11:45 cpuset.mems
   -r—r—r--. 1 root root 0 Jun  1 11:45 cpuset.mems.effective
   -r—r—r--. 1 root root 0 Jun  1 11:45 cpu.stat
   -rw-r—r--. 1 root root 0 Jun  1 11:45 cpu.weight
   -rw-r—r--. 1 root root 0 Jun  1 11:45 cpu.weight.nice
   -rw-r—r--. 1 root root 0 Jun  1 11:45 io.pressure
   -rw-r—r--. 1 root root 0 Jun  1 11:45 memory.pressure
   ```

9. ```
   # echo "1" > /sys/fs/cgroup/Example/tasks/cpuset.cpus
   ```

重要

1. ```
   # cat /sys/fs/cgroup/cgroup.subtree_control
   /sys/fs/cgroup/Example/cgroup.subtree_control
   cpuset cpu memory pids
   cpuset cpu
   ```

2. ```
   # cat /sys/fs/cgroup/Example/tasks/cpuset.cpus
   1
   ```

-
-
-
-

24.3.

-
-
-
-
-
-

```
...
  ├─── Example
  │    ├─── tasks
...
```

1.
```
# echo "200000 1000000" > /sys/fs/cgroup/Example/tasks/cpu.max
```

2.
```
# cat /sys/fs/cgroup/Example/tasks/cpu.max
200000 1000000
```

3.
```
# echo "34578" > /sys/fs/cgroup/Example/tasks/cgroup.procs
# echo "34579" > /sys/fs/cgroup/Example/tasks/cgroup.procs
```

1.
```
# cat /proc/34578/cgroup /proc/34579/cgroup
0::/Example/tasks
0::/Example/tasks
```

2.
```
# top
top - 11:13:53 up 23:10,  1 user,  load average: 0.26, 1.33, 1.66
Tasks: 104 total,   3 running, 101 sleeping,   0 stopped,   0 zombie
%Cpu(s):  3.0 us,  7.0 sy,  0.0 ni, 89.5 id,  0.0 wa,  0.2 hi,  0.2 si,  0.2 st
MiB Mem :   3737.4 total,   3312.6 free,    133.4 used,    291.4 buff/cache
MiB Swap:   4060.0 total,   4060.0 free,      0.0 used.   3376.0 avail Mem

   PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 34578 root      20   0   18720   1756   1468 R  10.0   0.0  37:36.13 sha1sum
 34579 root      20   0   18720   1772   1480 R  10.0   0.0  37:41.22 sha1sum
     1 root      20   0  186192  13940   9500 S   0.0   0.4   0:01.60 systemd
     2 root      20   0       0      0      0 S   0.0   0.0   0:00.01 kthreadd
     3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
     4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
...
```

24.4.

-

-
-
-
-

```
…
├─── Example
│    ├─── g1
│    ├─── g2
│    └─── g3
…
```

-

1.
```
# echo "150" > /sys/fs/cgroup/Example/g1/cpu.weight
# echo "100" > /sys/fs/cgroup/Example/g2/cpu.weight
# echo "50" > /sys/fs/cgroup/Example/g3/cpu.weight
```

2.
```
# echo "33373" > /sys/fs/cgroup/Example/g1/cgroup.procs
# echo "33374" > /sys/fs/cgroup/Example/g2/cgroup.procs
# echo "33377" > /sys/fs/cgroup/Example/g3/cgroup.procs
```


重要

1.
```
# cat /proc/33373/cgroup /proc/33374/cgroup /proc/33377/cgroup
0::/Example/g1
0::/Example/g2
0::/Example/g3
```

2.
```
# top
top - 05:17:18 up 1 day, 18:25,  1 user,  load average: 3.03, 3.03, 3.00
Tasks:  95 total,   4 running,  91 sleeping,   0 stopped,   0 zombie
%Cpu(s): 18.1 us, 81.6 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.3 hi,  0.0 si,  0.0 st
MiB Mem :   3737.0 total,   3233.7 free,    132.8 used,    370.5 buff/cache
MiB Swap:   4060.0 total,   4060.0 free,      0.0 used.   3373.1 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
```

```
  33373 root      20   0   18720   1748   1460 R  49.5   0.0 415:05.87 sha1sum
  33374 root      20   0   18720   1756   1464 R  32.9   0.0 412:58.33 sha1sum
  33377 root      20   0   18720   1860   1568 R  16.3   0.0 411:03.12 sha1sum
    760 root      20   0  416620  28540  15296 S   0.3   0.7   0:10.23 tuned
      1 root      20   0  186328  14108   9484 S   0.0   0.4   0:02.00 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.01 kthread
...
```

注意

第 25 章

25.1.

- **<*name*>.service**
- **<*name*>.scope**
- **<*parent-name*>.slice**

```
Control group /:
-.slice
 ├─user.slice
 │ ├─user-42.slice
 │ │ ├─session-c1.scope
 │ │ │ ├─ 967 gdm-session-worker [pam/gdm-launch-environment]
 │ │ │ ├─1035 /usr/libexec/gdm-x-session gnome-session --autostart
/usr/share/gdm/greeter/autostart
 │ │ │ ├─1054 /usr/libexec/Xorg vt1 -displayfd 3 -auth /run/user/42/gdm/Xauthority -
background none -noreset -keeptty -verbose 3
 │ │ │ ├─1212 /usr/libexec/gnome-session-binary --autostart
/usr/share/gdm/greeter/autostart
 │ │ │ ├─1369 /usr/bin/gnome-shell
 │ │ │ ├─1732 ibus-daemon --xim --panel disable
 │ │ │ ├─1752 /usr/libexec/ibus-dconf
 │ │ │ ├─1762 /usr/libexec/ibus-x11 --kill-daemon
 │ │ │ ├─1912 /usr/libexec/gsd-xsettings
 │ │ │ ├─1917 /usr/libexec/gsd-a11y-settings
 │ │ │ ├─1920 /usr/libexec/gsd-clipboard
...
 ├─init.scope
 │ └─1 /usr/lib/systemd/systemd --switched-root --system --deserialize 18
 └─system.slice
   ├─rngd.service
   │ └─800 /sbin/rngd -f
   ├─systemd-udevd.service
   │ └─659 /usr/lib/systemd/systemd-udevd
   ├─chronyd.service
   │ └─823 /usr/sbin/chronyd
   ├─auditd.service
   │ ├─761 /sbin/auditd
   │ └─763 /usr/sbin/sedispatch
   ├─accounts-daemon.service
   │ └─876 /usr/libexec/accounts-daemon
   ├─example.service
   │ ├─ 929 /bin/bash /home/jdoe/example.sh
   │ └─4902 sleep 1
  ...
```

-

25.2.

- **# systemd-run --unit=*<name>* --slice=*<name>*.slice *<command>***
  -
  -
  - **# Running as unit *<name>*.service**
- **# systemd-run --unit=*<name>* --slice=*<name>*.slice --remain-after-exit *<command>***
-

25.3.

- **# systemctl enable *<name>*.service**
-

25.4.

- **# systemctl set-property example.service MemoryMax=1500K**
- **# systemctl set-property --runtime example.service MemoryMax=1500K**

注意

-
-

25.5.

1. **…**
   [Service]
   MemoryMax=1500K
   …

   注意

2. **# systemctl daemon-reload**

3. **# systemctl restart example.service**

4.

1. **# cat /sys/fs/cgroup/memory/system.slice/example.service/memory.limit_in_bytes**
   1536000

   
   注意

-
-

25.6.

- **# systemctl stop <*name*>.service**
- **# systemctl kill <*name*>.service --kill-who=*PID,...* --signal=<*signal*>**
-
-
-
-
-

25.7.

1. **# systemctl stop <*name*>.service**
2. **# systemctl disable <*name*>.service**
3. **# rm /usr/lib/systemd/system/<*name*>.service**
4. **# systemctl daemon-reload**

-

25.8.

- **# systemctl**

```
UNIT                                LOAD   ACTIVE SUB       DESCRIPTION
…
init.scope                           loaded active running   System and Service Manager
session-2.scope                        loaded active running   Session 2 of user jdoe
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
…
-.slice                             loaded active active    Root Slice
machine.slice                          loaded active active    Virtual Machine and
Container Slice system-getty.slice                                          loaded
active active    system-getty.slice
system-lvm2\x2dpvscan.slice                  loaded active active    system-
lvm2\x2dpvscan.slice
system-sshd\x2dkeygen.slice                  loaded active active    system-
sshd\x2dkeygen.slice
system-systemd\x2dhibernate\x2dresume.slice       loaded active active    system-
systemd\x2dhibernate\x2dresume>
system-user\x2druntime\x2ddir.slice            loaded active active    system-
user\x2druntime\x2ddir.slice
system.slice                          loaded active active    System Slice
user-1000.slice                        loaded active active    User Slice of UID 1000
user-42.slice                         loaded active active    User Slice of UID 42
user.slice                           loaded active active    User and Session Slice
…
```

- ```
  # systemctl --all
  ```
- ```
  # systemctl --type service,masked
  ```
-
-

25.9.

- ```
  # systemd-cgls
  Control group /:
  -.slice
   ├─user.slice
   │ ├─user-42.slice
   │ │ ├─session-c1.scope
   │ │ │ ├─ 965 gdm-session-worker [pam/gdm-launch-environment]
   │ │ │ ├─1040 /usr/libexec/gdm-x-session gnome-session --autostart
  /usr/share/gdm/greeter/autostart
  …
   ├─init.scope
  ```
  ```

```
|   └─1 /usr/lib/systemd/systemd --switched-root --system --deserialize 18
└─system.slice
 …
  ├─example.service
  |   ├─6882 /bin/bash /home/jdoe/example.sh
  |   └─6902 sleep 1
  ├─systemd-journald.service
    └─629 /usr/lib/systemd/systemd-journald
 …
```

- ```
  # systemd-cgls memory
  Controller memory; Control group /:
  ├─1 /usr/lib/systemd/systemd --switched-root --system --deserialize 18
  ├─user.slice
  |   ├─user-42.slice
  |   |   ├─session-c1.scope
  |   |   |   ├─ 965 gdm-session-worker [pam/gdm-launch-environment]
  …
  └─system.slice
    |
    …
    ├─chronyd.service
    |   └─844 /usr/sbin/chronyd
    ├─example.service
    |   ├─8914 /bin/bash /home/jdoe/example.sh
    |   └─8916 sleep 1
    …
  ```

- ```
  # systemctl status example.service
  ● example.service - My example service
    Loaded: loaded (/usr/lib/systemd/system/example.service; enabled; vendor preset:
  disabled)
    Active: active (running) since Tue 2019-04-16 12:12:39 CEST; 3s ago
   Main PID: 17737 (bash)
     Tasks: 2 (limit: 11522)
    Memory: 496.0K (limit: 1.5M)
    CGroup: /system.slice/example.service
            ├─17737 /bin/bash /home/jdoe/example.sh
            └─17743 sleep 1
  Apr 16 12:12:39 redhat systemd[1]: Started My example service.
  Apr 16 12:12:39 redhat bash[17737]: The current time is Tue Apr 16 12:12:39 CEST
  2019
  Apr 16 12:12:40 redhat bash[17737]: The current time is Tue Apr 16 12:12:40 CEST
  2019
  ```

-

25.10.

1. ```
   # cat /proc/11269/cgroup
   12:freezer:/
   11:cpuset:/
   10:devices:/system.slice
   ```
   ```

```
9:memory:/system.slice/example.service
8:pids:/system.slice/example.service
7:hugetlb:/
6:rdma:/
5:perf_event:/
4:cpu,cpuacct:/
3:net_cls,net_prio:/
2:blkio:/
1:name=systemd:/system.slice/example.service
```

注意

-
-

25.11.

1.
```
# systemd-cgtop
Control Group                          Tasks   %CPU   Memory  Input/s Output/s
/                                        607   29.8     1.5G        -        -
/system.slice                            125      -   428.7M        -        -
/system.slice/ModemManager.service         3      -     8.6M        -        -
/system.slice/NetworkManager.service       3      -    12.8M        -        -
/system.slice/accounts-daemon.service      3      -     1.8M        -        -
/system.slice/boot.mount                   -      -    48.0K        -        -
/system.slice/chronyd.service              1      -     2.0M        -        -
/system.slice/cockpit.socket               -      -     1.3M        -        -
/system.slice/colord.service               3      -     3.5M        -        -
/system.slice/crond.service                1      -     1.8M        -        -
/system.slice/cups.service                 1      -     3.1M        -        -
/system.slice/dev-hugepages.mount          -      -   244.0K        -        -
/system.slice/dev-mapper-rhel\x2dswap.swap -      -   912.0K        -        -
/system.slice/dev-mqueue.mount             -      -    48.0K        -        -
/system.slice/example.service              2      -     2.0M        -        -
/system.slice/firewalld.service            2      -    28.8M        -        -
...
```

-

第 26 章

26.1.

-

26.2.

注意

-
-

26.3.

1. **$ systemctl show --property <*CPU time allocation policy option*> <*service name*>**
2. **# systemctl set-property <*service name*> <*CPU time allocation policy option*>=<*value*>**

- **$ systemctl show --property <*CPU time allocation policy option*> <*service name*>**
-
-

26.4.

26.5.

-
-
-
-
-

1. **$ systemctl show --property <*memory allocation configuration option*> <*service name*>**

2. **# systemctl set-property <*service name*> <*memory allocation configuration option*>= <*value*>**

注意

- **$ systemctl show --property <*memory allocation configuration option*> <*service name*>**

26.6.

注意

26.7.

-
-
-
-
-
-

1. **$ systemctl show --property <*I/O bandwidth configuration option*> <*service name*>**
2. **# systemctl set-property <*service name*> <*I/O bandwidth configuration option*>= <*value*>**

- **$ systemctl show --property <*I/O bandwidth configuration option*> <*service name*>**

26.8.

注意

注意

26.9.

- **# systemctl set-property <*service name*>.service AllowedCPUs=<*value*>**

  # systemctl set-property <*service name*>.service AllowedCPUs=0-5

- **# systemctl set-property <*service name*>.service AllowedMemoryNodes=<*value*>**

  # systemctl set-property <*service name*>.service AllowedMemoryNodes=0

第 27 章

27.1.

注意

1. $ systemctl show --property <CPU affinity configuration option> <service name>
2. # systemctl set-property <service name> CPUAffinity=<value>
3. # systemctl restart <service name>

1. # vi /etc/systemd/system.conf
2.
3.

27.2.

1. **$ systemctl show --property *<NUMA policy configuration option>* *<service name>***
2. **# systemctl set-property *<service name>* NUMAPolicy=*<value>***
3. **# systemctl restart *<service name>***

1.
2.
3. **# systemd daemon-reload**
4.

重要

-
-

27.3.

-
-

第 28 章

28.1.

- ```
  # yum install bcc-tools
  ```

- ```
  # ls -l /usr/share/bcc/tools/
  ...
  -rwxr-xr-x. 1 root root  4198 Dec 14 17:53 dcsnoop
  -rwxr-xr-x. 1 root root  3931 Dec 14 17:53 dcstat
  -rwxr-xr-x. 1 root root 20040 Dec 14 17:53 deadlock_detector
  -rw-r--r--. 1 root root  7105 Dec 14 17:53 deadlock_detector.c
  drwxr-xr-x. 3 root root  8192 Mar 11 10:28 doc
  -rwxr-xr-x. 1 root root  7588 Dec 14 17:53 execsnoop
  -rwxr-xr-x. 1 root root  6373 Dec 14 17:53 ext4dist
  -rwxr-xr-x. 1 root root 10401 Dec 14 17:53 ext4slower
  ...
  ```

28.2.

-
-

1.

```
# /usr/share/bcc/tools/execsnoop
```

1. ```
   $ ls /usr/share/bcc/tools/doc/
   ```

2. ```
   PCOMM PID    PPID   RET ARGS
   ls    8382   8287     0 /usr/bin/ls --color=auto /usr/share/bcc/tools/doc/
   ...
   ```

1.

```
# /usr/share/bcc/tools/opensnoop -n uname
```

1. ```
   $ uname
   ```

2. ```
   PID    COMM  FD ERR PATH
   8596   uname  3  0   /etc/ld.so.cache
   8596   uname  3  0   /lib64/libc.so.6
   8596   uname  3  0   /usr/lib/locale/locale-archive
   ...
   ```

-

1.

```
# /usr/share/bcc/tools/biotop 30
```

注意

1.
```
# dd if=/dev/vda of=/dev/zero
```

2.
```
PID    COMM             D MAJ MIN DISK       I/O  Kbytes     AVGms
9568   dd               R 252 0   vda      16294 14440636.0  3.69
48     kswapd0          W 252 0   vda       1763 120696.0    1.65
7571   gnome-shell      R 252 0   vda        834 83612.0     0.33
1891   gnome-shell      R 252 0   vda       1379 19792.0     0.15
7515   Xorg             R 252 0   vda        280  9940.0     0.28
7579   llvmpipe-1       R 252 0   vda        228  6928.0     0.19
9515   gnome-control-c  R 252 0   vda         62  6444.0     0.43
8112   gnome-terminal-  R 252 0   vda         67  2572.0     1.54
7807   gnome-software   R 252 0   vda         31  2336.0     0.73
9578   awk              R 252 0   vda         17  2228.0     0.66
7578   llvmpipe-0       R 252 0   vda        156  2204.0     0.07
9581   pgrep            R 252 0   vda         58  1748.0     0.42
7531   InputThread      R 252 0   vda         30  1200.0     0.48
7504   gdbus            R 252 0   vda          3  1164.0     0.30
1983   llvmpipe-1       R 252 0   vda         39   724.0     0.08
1982   llvmpipe-0       R 252 0   vda         36   652.0     0.06
...
```

1.
```
# /usr/share/bcc/tools/xfsslower 1
```

注意

2.
```
$ vim text
```

3.
```
TIME     COMM           PID    T BYTES   OFF_KB   LAT(ms) FILENAME
13:07:14 b'bash'        4754   R 256     0          7.11 b'vim'
13:07:14 b'vim'         4754   R 832     0          4.03 b'libgpm.so.2.1.0'
13:07:14 b'vim'         4754   R 32      20         1.04 b'libgpm.so.2.1.0'
13:07:14 b'vim'         4754   R 1982    0          2.30 b'vimrc'
13:07:14 b'vim'         4754   R 1393    0          2.52 b'getscriptPlugin.vim'
13:07:45 b'vim'         4754   S 0       0          6.71 b'text'
13:07:45 b'pool'        2588   R 16      0          5.58 b'text'
...
```

-
-
-

第 29 章

29.1.

-
-
-
-
-
-
-
-
-
-
-
-

29.2.

注意

```
$ cat /sys/class/tpm/tpm0/tpm_version_major
2
```

29.3.

- ```
  # modprobe trusted
  ```
-

注意

1. a.
   ```
   # TPM_DEVICE=/dev/tpm0 tsscreateprimary -hi o -st
   Handle 80000000
   # TPM_DEVICE=/dev/tpm0 tssevictcontrol -hi o -ho 80000000 -hp 81000001
   ```

   b.
   ```
   # tpm2_createprimary --key-algorithm=rsa2048 --key-context=key.ctxt
   ```
   ```

```
name-alg:
  value: sha256
  raw: 0xb
…
sym-keybits: 128
rsa: xxxxxx…

# tpm2_evictcontrol -c key.ctxt 0x81000001
persistentHandle: 0x81000001
action: persisted
```

2. a.
```
# keyctl add trusted kmk "new 32 keyhandle=0x81000001" @u
642500861
```

   b.
```
# keyctl add trusted kmk "new 32" @u
```

3.
```
# keyctl show
Session Keyring
      -3 --alswrv    500   500  keyring: ses 97833714 --alswrv 500 -1 \ keyring: uid.1000
642500861 --alswrv 500 500 \ trusted: kmk
```

4.
```
# keyctl pipe 642500861 > kmk.blob
```

5.
```
# keyctl add trusted kmk "load `cat kmk.blob`" @u
268728824
```

6.
```
# keyctl add encrypted encr-key "new trusted:kmk 32" @u
159771175
```

-

29.4.

-
```
# modprobe encrypted-keys
```

1.
```
# keyctl add user kmk-user "$(dd if=/dev/urandom bs=1 count=32 2>/dev/null)" @u
427069434
```

2.
```
# keyctl add encrypted encr-key "new user:kmk-user 32" @u
1012412758
```

1.
```
# keyctl list @u
2 keys in keyring:
427069434: --alswrv  1000  1000 user: kmk-user
1012412758: --alswrv  1000  1000 encrypted: encr-key
```

-

29.5.


重要

- 
注意

- ```
  # mount
  ...
  securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
  ...
  ```

- ```
  # grep <options> pattern <files>
  ```

  ```
  # dmesg | grep -i -e EVM -e IMA -w
  [ 0.598533] ima: No TPM chip found, activating TPM-bypass!
  [ 0.599435] ima: Allocated hash algorithm: sha256
  [ 0.600266] ima: No architecture policies found
  [ 0.600813] evm: Initialising EVM extended attributes:
  [ 0.601581] evm: security.selinux
  [ 0.601963] evm: security.ima
  [ 0.602353] evm: security.capability
  [ 0.602713] evm: HMAC attrs: 0x1
  [ 1.455657] systemd[1]: systemd 239 (239-74.el8_8) running in system mode. (+PAM
  +AUDIT +SELINUX +IMA -APPARMOR +SMACK +SYSVINIT +UTMP +LIBCRYPTSETUP
  +GCRYPT +GNUTLS +ACL +XZ +LZ4 +SECCOMP +BLKID +ELFUTILS +KMOD +IDN2 -
  IDN +PCRE2 default-hierarchy=legacy)
  [ 2.532639] systemd[1]: systemd 239 (239-74.el8_8) running in system mode. (+PAM
  +AUDIT +SELINUX +IMA -APPARMOR +SMACK +SYSVINIT +UTMP +LIBCRYPTSETUP
  +GCRYPT +GNUTLS +ACL +XZ +LZ4 +SECCOMP +BLKID +ELFUTILS +KMOD +IDN2 -
  IDN +PCRE2 default-hierarchy=legacy)
  ```

1. ```
   # grubby --update-kernel=/boot/vmlinuz-$(uname -r) --
   args="ima_policy=appraise_tcb ima_appraise=fix evm=fix"
   ```

2.

3. ```
   # cat /proc/cmdline
   BOOT_IMAGE=(hd0,msdos1)/vmlinuz-4.18.0-167.el8.x86_64 root=/dev/mapper/rhel-
   root ro crashkernel=auto resume=/dev/mapper/rhel-swap rd.lvm.lv=rhel/root
   rd.lvm.lv=rhel/swap rhgb quiet ima_policy=appraise_tcb ima_appraise=fix evm=fix
   ```

4. ```
   # keyctl add user kmk "$(dd if=/dev/urandom bs=1 count=32 2> /dev/null)" @u
   748544121
   ```

5. ```
   # keyctl add encrypted evm-key "new user:kmk 64" @u
   641780271
   ```

重要

6. **# mkdir -p /etc/keys/**

7. **# keyctl pipe $(keyctl search @u user kmk) > /etc/keys/kmk**

8. **# keyctl pipe $(keyctl search @u encrypted evm-key) > /etc/keys/evm-key**

9. **# keyctl show**
   Session Keyring
   **974575405 --alswrv 0 0 keyring: *ses 299489774 --alswrv 0 65534* **
   keyring: *uid.0 748544121 --alswrv 0 0* \ user: kmk
   641780271 --alswrv 0 0 _ encrypted: evm-key

   # ls -l /etc/keys/
   total 8
   -rw-r--r--. 1 root root 246 Jun 24 12:44 evm-key
   -rw-r--r--. 1 root root 32 Jun 24 12:43 kmk

10. a. **# keyctl add user kmk "$(cat /etc/keys/kmk)" @u**
    451342217

    b. **# keyctl add encrypted evm-key "load $(cat /etc/keys/evm-key)" @u**
    924537557

11. **# echo 1 > /sys/kernel/security/evm**

12. **# find / -fstype xfs -type f -uid 0 -exec head -n 1 '{}' >/dev/null \;**

警告

- **# dmesg | tail -1**
  […] evm: key initialized

29.6.

-
-
-

1. **# echo "Test text" > test_file**

1. **# echo <*test_text*> > test_file**

2. **# getfattr -m . -d test_file**
   # file: test_file
   security.evm=0sAnDly4VPA0HArpPO/EqiutnNyBql
   security.ima=0sAQOEDeuUnWzwwKYk+n66h/vby3eD

-

第 30 章

-
-

重要

30.1.

-
-
-
-
-
-

1.
```
---
- name: Configuring kernel settings
  hosts: managed-node-01.example.com
  tasks:
    - name: Configure hugepages, packet size for loopback device, and limits on
simultaneously open files.
      ansible.builtin.include_role:
        name: redhat.rhel_system_roles.kernel_settings
      vars:
        kernel_settings_sysctl:
          - name: fs.file-max
            value: 400000
          - name: kernel.threads-max
            value: 65536
        kernel_settings_sysfs:
          - name: /sys/class/net/lo/mtu
            value: 65000
        kernel_settings_transparent_hugepages: madvise
        kernel_settings_reboot_ok: true
```

1.
```
$ ansible-playbook --syntax-check ~/playbook.yml
```

2.
```
$ ansible-playbook ~/playbook.yml
```

-
```
# ansible managed-node-01.example.com -m command -a 'sysctl fs.file-max
kernel.threads-max net.ipv6.conf.lo.mtu'
# ansible managed-node-01.example.com -m command -a 'cat
```

```
/sys/kernel/mm/transparent_hugepage/enabled'
```

-
-

第 31 章

-
-
-

31.1.

-
-
-
-

1.
```
---
- name: Configuration and management of GRUB boot loader
  hosts: managed-node-01.example.com
  tasks:
    - name: Update existing boot loader entries
      ansible.builtin.include_role:
        name: redhat.rhel_system_roles.bootloader
      vars:
        bootloader_settings:
          - kernel:
              path: /boot/vmlinuz-5.14.0-362.24.1.el9_3.aarch64
            options:
              - name: quiet
                state: present
        bootloader_reboot_ok: true
```

2.
```
$ ansible-playbook --syntax-check ~/playbook.yml
```

3.
```
$ ansible-playbook ~/playbook.yml
```

-
```
# ansible managed-node-01.example.com -m ansible.builtin.command -a 'grubby --
info=ALL'
managed-node-01.example.com | CHANGED | rc=0 >>
...
index=1
kernel="/boot/vmlinuz-5.14.0-362.24.1.el9_3.aarch64"
args="ro crashkernel=1G-4G:256M,4G-64G:320M,64G-:576M rd.lvm.lv=rhel/root
rd.lvm.lv=rhel/swap $tuned_params quiet"
root="/dev/mapper/rhel-root"
initrd="/boot/initramfs-5.14.0-362.24.1.el9_3.aarch64.img $tuned_initrd"
```

```
title="Red Hat Enterprise Linux (5.14.0-362.24.1.el9_3.aarch64) 9.4 (Plow)"
id="2c9ec787230141a9b087f774955795ab-5.14.0-362.24.1.el9_3.aarch64"
...
```

-
-
-
-
-
-

31.2.

-
-
-

1. a.

```
$ ansible-vault create ~/vault.yml
New Vault password: <vault_password>
Confirm New Vault password: <vault_password>
```

   b.

```
pwd: <password>
```

   c.

2.

```
---
- name: Configuration and management of GRUB boot loader
  hosts: managed-node-01.example.com
  vars_files:
    - ~/vault.yml
  tasks:
    - name: Set the bootloader password
      ansible.builtin.include_role:
        name: redhat.rhel_system_roles.bootloader
      vars:
        bootloader_password: "{{ pwd }}"
        bootloader_reboot_ok: true
```

重要

3.

```
$ ansible-playbook --syntax-check --ask-vault-pass ~/playbook.yml
```

4. **$ ansible-playbook --ask-vault-pass ~/playbook.yml**

1. 

[D]

2. 

[D]

3. 

[D]

-
-

31.3.

-
-
-

1.
```
---
- name: Configuration and management of the GRUB boot loader
  hosts: managed-node-01.example.com
  tasks:
    - name: Update the boot loader timeout
      ansible.builtin.include_role:
        name: redhat.rhel_system_roles.bootloader
      vars:
        bootloader_timeout: 10
```

2.
```
$ ansible-playbook --syntax-check ~/playbook.yml
```

3.
```
$ ansible-playbook ~/playbook.yml
```

-

1.
```
# ansible managed-node-01.example.com -m ansible.builtin.reboot
managed-node-01.example.com | CHANGED => {
    "changed": true,
    "elapsed": 21,
    "rebooted": true
}
```

2. 

[D]

-
```
# ansible managed-node-01.example.com -m ansible.builtin.command -a "grep
'timeout' /boot/grub2/grub.cfg"
managed-node-01.example.com | CHANGED | rc=0 >>
if [ x$feature_timeout_style = xy ] ; then
  set timeout_style=menu
  set timeout=10
# Fallback normal timeout code in case the timeout_style feature is
  set timeout=10
if [ x$feature_timeout_style = xy ] ; then
    set timeout_style=menu
    set timeout=10
    set orig_timeout_style=${timeout_style}
    set orig_timeout=${timeout}
      # timeout_style=menu + timeout=0 avoids the countdown code keypress
check
      set timeout_style=menu
      set timeout=10
      set timeout_style=hidden
      set timeout=10
if [ x$feature_timeout_style = xy ]; then
  if [ "${menu_show_once_timeout}" ]; then
    set timeout_style=menu
    set timeout=10
```

```
unset menu_show_once_timeout
save_env menu_show_once_timeout
```

-
-

31.4.

-
-
-
-
-
-

1.
```
---
- name: Configuration and management of GRUB boot loader
  hosts: managed-node-01.example.com
  tasks:
    - name: Gather information about the boot loader configuration
      ansible.builtin.include_role:
        name: redhat.rhel_system_roles.bootloader
      vars:
        bootloader_gather_facts: true

    - name: Display the collected boot loader configuration information
      debug:
        var: bootloader_facts
```

2.
```
$ ansible-playbook --syntax-check ~/playbook.yml
```

3.
```
$ ansible-playbook ~/playbook.yml
```

-
```
...
    "bootloader_facts": [
        {
            "args": "ro crashkernel=1G-4G:256M,4G-64G:320M,64G-:576M
rd.lvm.lv=rhel/root rd.lvm.lv=rhel/swap $tuned_params quiet",
            "default": true,
            "id": "2c9ec787230141a9b087f774955795ab-5.14.0-362.24.1.el9_3.aarch64",
            "index": "1",
            "initrd": "/boot/initramfs-5.14.0-362.24.1.el9_3.aarch64.img $tuned_initrd",
            "kernel": "/boot/vmlinuz-5.14.0-362.24.1.el9_3.aarch64",
            "root": "/dev/mapper/rhel-root",
```

```
            "title": "Red Hat Enterprise Linux (5.14.0-362.24.1.el9_3.aarch64) 9.4 (Plow)"
        }
    ]
...
```

-
-
-

第 32 章

32.1.

-
-
-

例 32.1.

```
Feb  5 15:41:33 hostname kernel: pcieport 10003:00:00.0: AER: Corrected error received:
id=ae00
Feb  5 15:41:33 hostname kernel: pcieport 10003:00:00.0: AER: Multiple Corrected error
received: id=ae00
Feb  5 15:41:33 hostname kernel: pcieport 10003:00:00.0: PCIe Bus Error:
severity=Corrected, type=Data Link Layer, id=0000(Receiver ID)
Feb  5 15:41:33 hostname kernel: pcieport 10003:00:00.0:   device [8086:2030] error
status/mask=000000c0/00002000
Feb  5 15:41:33 hostname kernel: pcieport 10003:00:00.0:    [ 6] Bad TLP
Feb  5 15:41:33 hostname kernel: pcieport 10003:00:00.0:    [ 7] Bad DLLP
Feb  5 15:41:33 hostname kernel: pcieport 10003:00:00.0: AER: Multiple Corrected error
received: id=ae00
Feb  5 15:41:33 hostname kernel: pcieport 10003:00:00.0: PCIe Bus Error:
severity=Corrected, type=Data Link Layer, id=0000(Receiver ID)
Feb  5 15:41:33 hostname kernel: pcieport 10003:00:00.0:   device [8086:2030] error
status/mask=00000040/00002000
```

32.2.

1.
```
# yum install rasdaemon
```

2.
```
# systemctl enable --now rasdaemon
Created symlink /etc/systemd/system/multi-user.target.wants/rasdaemon.service →
/usr/lib/systemd/system/rasdaemon.service.
```

3.
```
# ras-mc-ctl --summary
# ras-mc-ctl --errors
```

-
-