Oracle® Linux

Administrator's Solutions Guide for Release 6

E37355-67
January 2019

Abstract

This manual provides information about the advanced features for this version of Oracle Linux that have been engineered by Oracle.

Document generated on: 2019-01-17 (revision: 6924)

Table of Contents

Preface

The *Oracle Linux Administrator's Solutions Guide* provides information about the advanced features of Oracle Linux and, in particular, the Unbreakable Enterprise Kernel (UEK).

Audience

This document is intended for administrators who need to configure the advanced features of Oracle Linux and the Unbreakable Enterprise Kernel (UEK). It is assumed that readers are familiar with web and virtualization technologies and have a general understanding of the Linux operating system.

Document Organization

The document is organized as follows:

- Chapter 1, *The Unbreakable Enterprise Kernel* describes the advanced features that are available with the Unbreakable Enterprise Kernel (UEK).
- Chapter 2, *Yum* describes how to use the `yum` utility to install and upgrade software packages.
- Chapter 3, *Ksplice* describes how to configure Ksplice Uptrack to update a running system kernel.
- Chapter 4, *The Btrfs File System* describes how to deploy and use the advanced features of the btrfs file system.
- Chapter 5, *The XFS File System* describes how to deploy and use the advanced features of the XFS file system.
- Chapter 6, *Oracle Cluster File System Version 2* describes how to configure and use the Oracle Cluster File System Version 2 (OCFS2).
- Chapter 7, *Control Groups* describes how to use Control Groups (cgroups) to manage the resource utilization of sets of processes.
- Chapter 8, *Linux Containers* describes how to use Linux Containers (LXC) to isolate applications and entire operating system images from the other processes that are running on a host system.
- Chapter 9, *HugePages* describes how to set up the HugePages feature on a system that is running several Oracle Database instances.
- Chapter 10, *Using kexec for Fast Rebooting* describes how to use the `kexec` command to enable fast system rebooting.
- Chapter 11, *DTrace* introduces the dynamic tracing (DTrace) facility that you can use to examine the behavior of the operating system and the operating system kernel.
- Chapter 12, *Support Diagnostic Tools* describes the `sosreport`, Kdump, and OSWbb tools that can help diagnose problems with a system.

Related Documents

The documentation for this product is available at:

https://www.oracle.com/technetwork/server-storage/linux/documentation/index.html.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at
http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit
http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Chapter 1 The Unbreakable Enterprise Kernel

Table of Contents

This chapter describes the advanced features that are available with the Unbreakable Enterprise Kernel (UEK).

1.1 About the Unbreakable Enterprise Kernel

In September 2010, Oracle announced the new Unbreakable Enterprise Kernel (UEK) for Oracle Linux as a recommended kernel for deployment with Oracle Linux 5. Beginning with Oracle Linux 5.5, you could choose to use either the Red Hat Compatible Kernel or the UEK. In Oracle Linux 5.6, the UEK became the default kernel.

The prime motivation for creating the UEK was to provide a modern, high performance Linux kernel for the Exadata and Exalogic engineered systems. The kernel needed to scale as the number of CPUs, memory and InfiniBand connects was increased.

Oracle tests the UEK intensively with demanding Oracle workloads, and recommends the UEK for Oracle deployments and all other enterprise deployments. Oracle is committed to offering compatibility with Red Hat, and continues to release and support the Red Hat Compatible Kernel as part of Oracle Linux for customers that require strict RHEL compatibility. Under the Oracle Linux Support Program, customers can receive full support for Oracle Linux running with either kernel.

Oracle releases new versions of the UEK every 12-18 months. The latest version of the UEK receives quarterly patch updates including drivers for new hardware support, bug fixes, and critical security patches. Oracle also provides critical security patches for previous versions of the UEK. These patches are available as new installable kernels and, with the exception of device driver updates, as Ksplice patches.

Using the UEK instead of the Red Hat Compatible Kernel changes only the operating system kernel. There are no changes to any libraries, APIs, or any user-space applications Existing applications run unchanged regardless of which kernel you use. Using a different kernel does not change system libraries such as `glibc`. The version of `glibc` in Oracle Linux 6 remains the same, regardless of the kernel version.

1.1.1 About UEK Release 1

Release 1 of the UEK is based on a stable 2.6.32 Linux kernel and provides additional performance improvements, including:

- Improved IRQ (interrupt request) balancing.
- Reduced lock contention across the kernel.
- Improved network I/O by the use of receive packet steering and RDS improvements.

- Improved virtual memory performance.

The UEK release 1 includes optimizations developed in collaboration with Oracle's Database, Middleware, and Hardware engineering teams to ensure stability and optimal performance for demanding enterprise workloads. In addition to performance improvements for large systems, the following UEK features are relevant to using Linux in the data center:

- The Infiniband OpenFabrics Enterprise Distribution (OFED) 1.5.1 implements Remote Direct Memory Access (RDMA) and kernel bypass mechanisms to deliver high-efficiency computing, wire-speed messaging, ultra-low microsecond latencies and fast I/O for servers, block storage and file systems. This also includes an improved RDS (reliable datagram sockets) stack for high-speed, low-latency networking. As an InfiniBand Upper Layer Protocol (ULP), RDS allows the reliable transmission of IPC datagrams up to 1 MB in size, and is currently used in Oracle Real Application Clusters (RAC), and in the Exadata and Exalogic products.

- A number of additional patches significantly improve the performance of Non-Uniform Memory Access (NUMA) systems with many CPUs, CPU cores, and memory nodes.

- Receive Packet Steering (RPS) is a software implementation of Receive Side Scaling (RSS) that improves overall networking performance, especially for high loads. RPS distributes the load of received network packet processing across multiple CPUs and ensures that the same CPU handles all packets for a specific combination of IP address and port.

  To configure the list of CPUs to which RPS can forward traffic, use `/sys/class/net/interface/queues/rx-N/rps_cpus`, which implements a CPU bitmap for a specified network interface and receive queue. The default value is zero, which disables RPS and results in the CPU that is handling the network interrupt also processing the incoming packet. To enable RPS and allow a particular set of CPUs to handle interrupts for the receive queue on an interface, set the value of their positions in the bitmap to 1. For example, to enable RPS to use CPUs 0, 1, 2, and 3 for the `rx-0` queue on `eth0`, set the value of `rps_cpus` to f (that is, 1+2+4+8 = 15 in hexadecimal):

  ```
  # cat f > /sys/class/net/eth0/queues/rx-0/rps_cpus
  ```

  There is no benefit in configuring RPS on a system with a multiqueue network device as RSS is usually automatically configured to map a CPU to each receive queue.

  For an interface with a single transmit queue, you should typically set `rps_cpus` for CPUs in the same memory domain so that they share the same queue. On a non-NUMA system, this means that you would set all the available CPUs in `rps_cpus`.

  Tip

  To verify which CPUs are handling receive interrupts, use the command `watch -n1 cat /proc/softirqs` and monitor the value of `NET_RX` for each CPU.

- Receive Flow Steering (RFS) extends RPS to coordinate how the system processes network packets in parallel. RFS performs application matching to direct network traffic to the CPU on which the application is running.

  To configure RFS, use `/proc/sys/net/core/rps_sock_flow_entries`, which sets the number of entries in the global flow table, and `/sys/class/net/interface/queues/rx-N/rps_flow_cnt`, which sets the number of entries in the per-queue flow table for a network interface. The default values are both zero, which disables RFS. To enable RFS, set the value of `rps_sock_flow_entries` to the maximum expected number of concurrently active connections, and the value of `rps_flow_cnt` to `rps_sock_flow_entries/Nq`, where *Nq* is the number of receive queues on a device. Any value that you enter is rounded up to the nearest power of 2. The suggested value of `rps_sock_flow_entries` is 32768 for a moderately loaded server.

- The kernel can detect solid state disks (SSDs), and tune itself for their use by bypassing the optimization code for spinning media and by dispatching I/O without delay to the SSD.

- The data integrity features verify data from the database all the way down to the individual storage spindle or device. The Linux data integrity framework (DIF) allows applications or kernel subsystems to attach metadata to I/O operations, allowing devices that support DIF to verify the integrity before passing them further down the stack and physically committing them to disk. The Data Integrity Extensions (DIX) feature enables the exchange of protection metadata between the operating system and the host bus adapter (HBA), and helps to prevent silent data corruption. The data-integrity enabled Automatic Storage Manager (ASM) that is available as an add-on with Oracle Database also protects against data corruption from application to disk platter.

  For more information about the data integrity features, including programming with the block layer integrity API, see http://www.kernel.org/doc/Documentation/block/data-integrity.txt.

- Oracle Cluster File System 2 (OCFS2) version 1.6 includes a large number of features. For more information, see Chapter 6, *Oracle Cluster File System Version 2*.

1.1.2 About UEK Release 2

> **Note**
>
> The kernel version in UEK Release 2 (UEK R2) is stated as 2.6.39, but it is actually based on the 3.0-stable Linux kernel. This renumbering allows some low-level system utilities that expect the kernel version to start with 2.6 to run without change.

UEK R2 includes the following improvements over release 1:

- Interrupt scalability is refined, and scheduler tuning is improved, especially for Java workloads.

- Transcendent memory helps the performance of virtualization solutions for a broad range of workloads by allowing a hypervisor to cache clean memory pages and eliminating costly disk reads of file data by virtual machines, allowing you to increase their capacity and usage level. Transcendent memory also implements an LZO-compressed page cache, or *zcache*, which reduces disk I/O.

- Transmit packet steering (XPS) distributes outgoing network packets from a multiqueue network device across the CPUs. XPS chooses the transmit queue for outgoing packets based on the lock contention and NUMA cost on each CPU, and it selects which CPU uses that queue to send a packet.

  To configure the list of CPUs to which XPS can forward traffic, use `/sys/class/net/interface/queues/tx-N/xps_cpus`, which implements a CPU bitmap for a specified network interface and transmit queue. The default value is zero, which disables XPS. To enable XPS and allow a particular set of CPUs to use a specified transmit queue on an interface, set the value of their positions in the bitmap to 1. For example, to enable XPS to use CPUs 4, 5, 6, and 7 for the `tx-0` queue on `eth0`, set the value of `rps_cpus` to f0 (that is, 16+32+64+128 = 240 in hexadecimal):

  ```
  # cat f0 > /sys/class/net/eth0/queues/tx-0/xps_cpus
  ```

  There is no benefit in configuring XPS for a network device with a single transmit queue.

  For a system with a multiqueue network device, configure XPS so that each CPU maps onto one transmit queue. If a system has an equal number of CPUs and transit queues, you can configure exclusive pairings in XPS to eliminate queue contention. If a system has more CPUs than queues, configure CPUs that share the same cache to the same transmit queue.

- The btrfs file system for Linux is designed to meet the expanding scalability requirements of large storage subsystems. For more information, see Chapter 4, *The Btrfs File System*.

- Cgroups provide fine-grained control of CPU, I/O and memory resources. For more information, see Chapter 7, *Control Groups*.
- Linux containers provide multiple user-space versions of the operating system on the same server.Each container is an isolated environment with its own process and network space. For more information, see Chapter 8, *Linux Containers*.
- Transparent huge pages take advantage of the memory management capabilities of modern CPUs to allow the kernel to manage physical memory more efficiently by reducing overhead in the virtual memory subsystem, and by improving the caching of frequently accessed virtual addresses for memory-intensive workloads. For more information, see Chapter 9, *HugePages*.
- DTrace allows you to explore your system to understand how it works, to track down performance problems across many layers of software, or to locate the causes of aberrant behavior. DTrace is currently available only on ULN. For more information, see Chapter 11, *DTrace*.
- The `configfs` virtual file system, engineered by Oracle, allows you to configure the settings of kernel objects where a file system or device driver implements this feature. `configfs` provides an alternative mechanism for changing the values of settings to the `ioctl()` system call, and complements the intended functionality of `sysfs` as a means to view kernel objects.

  The cluster stack for OCFS2, O2CB, uses `configfs` to set cluster timeouts and to examine the cluster status.

  The low-level I/O (LIO) driver uses `configfs` as a multiprotocal SCSI target to support the configuration of FCoE, Fibre Channel, iSCSI and InfiniBand using the `lio-utils` tool set.

  For more information about the implementation of `configfs`, see http://www.kernel.org/doc/Documentation/filesystems/configfs/configfs.txt.
- The `dm-nfs` feature creates virtual disk devices (LUNs) where the data is stored in an NFS file instead of on local storage. Managed networked storage has many benefits over keeping virtual devices on a disk that is local to the physical host.

  The `dm-nfs` kernel module provides a device-mapper target that allows you to treat a file on an NFS file system as a block device that can be loopback-mounted locally.

  The following sample code demonstrates how to use `dmsetup` to create a mapped device (`/dev/mapper/$dm_nfsdev`) for the file `$filename` that is accessible on a mounted NFS file system:

  ```
  nblks=`stat -c '%s' $filename`
  echo -n "0 $nblks nfs $filename 0" | dmsetup create $dm_nfsdev
  ```

  A sample use case is the fast migration of guest VMs for load balancing or if a physical host requires maintenance. This functionality is also possible using iSCSI LUNs, but the advantage of `dm-nfs` is that you can manage new virtual drives on a local host system, rather than requiring a storage administrator to initialize new LUNs.

  `dm-nfs` uses asynchronous direct I/O so that I/O is performed efficiently and coherently. A guest's disk data is not cached locally on the host. If the host crashes, there is a lower probability of data corruption. If a guest is frozen, you can take a clean backup of its virtual disk, as you can be certain that its data has been fully written out.

1.1.3 About UEK Release 3

The kernel version in UEK Release 3 (UEK R3) is based on the mainline Linux kernel version 3.8.13. Low-level system utilities that expect the kernel version to start with 2.6 can run without change if they use the `UNAME26` personality (for example, by using the `uname26` wrapper utility).

The UEK R3 kernel packages are available on the `ol6_x86_64_UEKR3_latest` channel. For more information, see the *Unbreakable Enterprise Kernel Release 3 Release Notes*.

1.1.4 About UEK Release 4

The kernel version in UEK Release 4 (UEK R4) is based on the mainline Linux kernel version 4.1.12.

The UEK R4 kernel packages are available on the `ol6_x86_64_UEKR4` channel. For more information, see the *Unbreakable Enterprise Kernel Release 4 Release Notes*.

1.2 Obtaining and Installing the UEK Packages

You can obtain and install the UEK and associated firmware packages in the following ways:

- If you have a valid Oracle Linux Support subscription, you can obtain the latest Oracle Linux and UEK packages from the Unbreakable Linux Network (ULN) at https://linux.oracle.com. After you have logged in to ULN and registered your system, you can subscribe the system to the UEK channel for the appropriate Oracle Linux release and machine architecture. This channel will provide the latest Oracle Linux packages and updates for your system as they become available.

  For more information about ULN, see *Oracle Linux Unbreakable Linux Network User's Guide*.

- You can obtain Oracle Linux and UEK packages from the Oracle Linux yum server. You can either edit the yum repository configuration file at `/etc/yum.repos.d/uek-ol6.repo` or you can use `yum-config-manager` to enable and disable the repositories that you wish to configure. For example:

  ```
  # yum-config-manager --disable ol7_UEKR3
  # yum-config-manager --enable ol7_UEKR5
  ```

  For more information about `yum`, see Chapter 2, *Yum*

To list the installed kernel packages and also the kernel packages that are available to be installed from the repositories that you have enabled, use the following `yum` command:

```
# yum list kernel*
Installed Packages
kernel.x86_64                 2.6.32-220.el6        @anaconda-OracleLinuxServer-2011...x86_64/6.2
kernel.x86_64                 2.6.32-279.el6        @ol6_latest
kernel.x86_64                 2.6.32-279.2.1.el6    @ol6_latest
kernel-devel.x86_64           2.6.32-220.el6        @anaconda-OracleLinuxServer-2011...x86_64/6.2
kernel-devel.x86_64           2.6.32-279.el6        @ol6_latest
kernel-devel.x86_64           2.6.32-279.2.1.el6    @ol6_latest
kernel-firmware.noarch        2.6.32-279.2.1.el6    @ol6_latest
kernel-uek.x86_64             2.6.39-200.24.1.el6uek installed
kernel-uek-devel.x86_64       2.6.32-300.32.1.el6uek @ol6_latest
kernel-uek-devel.x86_64       2.6.39-200.24.1.el6uek @ol6_UEK_latest
kernel-uek-devel.x86_64       2.6.39-200.29.2.el6uek @ol6_UEK_latest
kernel-uek-firmware.noarch    2.6.39-200.24.1.el6uek installed
kernel-uek-headers.x86_64     2.6.32-300.32.1.el6uek @ol6_latest
Available Packages
kernel.x86_64                 2.6.32-279.5.2.el6    ol6_latest
kernel-debug.x86_64           2.6.32-279.5.2.el6    ol6_latest
kernel-debug-devel.x86_64     2.6.32-279.5.2.el6    ol6_latest
```

```
kernel-devel.x86_64           2.6.32-279.5.2.el6     ol6_latest
kernel-doc.noarch             2.6.32-279.5.2.el6     ol6_latest
kernel-firmware.noarch        2.6.32-279.5.2.el6     ol6_latest
kernel-headers.x86_64         2.6.32-279.5.2.el6     ol6_latest
kernel-uek.x86_64             2.6.39-200.29.3.el6uek ol6_UEK_latest
kernel-uek-debug.x86_64       2.6.39-200.29.3.el6uek ol6_UEK_latest
kernel-uek-debug-devel.x86_64 2.6.39-200.29.3.el6uek ol6_UEK_latest
kernel-uek-devel.x86_64       2.6.39-200.29.3.el6uek ol6_UEK_latest
kernel-uek-doc.noarch         2.6.39-200.29.3.el6uek ol6_UEK_latest
kernel-uek-firmware.noarch    2.6.39-200.29.3.el6uek ol6_UEK_latest
```

Alternatively, you can use the `rpm -qa` command to list the installed packages:

```
# rpm -qa | grep ^kernel | sort
kernel-2.6.32-220.el6.x86_64
kernel-2.6.32-279.2.1.el6.x86_64
kernel-2.6.32-279.el6.x86_64
kernel-devel-2.6.32-220.el6.x86_64
kernel-devel-2.6.32-279.2.1.el6.x86_64
kernel-devel-2.6.32-279.el6.x86_64
kernel-firmware-2.6.32-279.2.1.el6.noarch
kernel-uek-2.6.39-200.24.1.el6uek.x86_64
kernel-uek-devel-2.6.32-300.32.1.el6uek.x86_64
kernel-uek-devel-2.6.39-200.24.1.el6uek.x86_64
kernel-uek-devel-2.6.39-200.29.2.el6uek.x86_64
kernel-uek-firmware-2.6.39-200.24.1.el6uek.noarch
kernel-uek-headers-2.6.32-300.32.1.el6uek.x86_64
```

1.3 For More Information About the UEK

For more information about the UEK, see https://www.oracle.com/technetwork/server-storage/linux/technologies/uek-overview-2043074.html.

Chapter 2 Yum

Table of Contents

This chapter describes how you can use the `yum` utility to install and upgrade software packages.

2.1 About Yum

Oracle Linux provides the `yum` utility which you can use to install or upgrade RPM packages. The main benefit of using yum is that it also installs or upgrades any package dependencies. `yum` downloads the packages from repositories such as those that are available on the Oracle Linux yum server, but you can also set up your own repositories on systems that do not have Internet access.

The Oracle Linux yum server is a convenient way to install Oracle Linux and Oracle VM packages, including bug fixes, security fixes and enhancements, rather than installing them from installation media. You can access the server at https://yum.oracle.com/.

You can also subscribe to the Oracle Linux and Oracle VM errata mailing lists to be notified when new packages are released. You can access the mailing lists at https://oss.oracle.com/mailman/listinfo/el-errata and https://oss.oracle.com/mailman/listinfo/oraclevm-errata.

2.2 About ULN

The repositories available on the Oracle Linux yum server are aligned with the channels that are available on the Unbreakable Linux Network (ULN), with the exception of ULN channels that are limited to Oracle Linux Premier Support customers. These include channels for products such as Ksplice and DTrace.

ULN is tightly integrated with yum. If you have registered your system with ULN, you can use `yum` commands with ULN channels to maintain the software on your system, as described in *Oracle Linux Unbreakable Linux Network User's Guide*.

2.3 Yum Configuration

The main configuration file for yum is `/etc/yum.conf`. The global definitions for yum are located under the `[main]` section heading of the yum configuration file. The following table lists the important directives.

Directive	Description
cachedir	Directory used to store downloaded packages.
debuglevel	Logging level, from 0 (none) to 10 (all).
exactarch	If set to 1, only update packages for the correct architecture.
exclude	A space separated list of packages to exclude from installs or updates, for example: exclude=VirtualBox-4.? kernel*.
gpgcheck	If set to 1, verify the authenticity of the packages by checking the GPG signatures. You might need to set gpgcheck to 0 if a package is unsigned, but you should be wary that the package could have been maliciously altered.
gpgkey	Pathname of the GPG public key file.
installonly_limit	Maximum number of versions that can be installed of any one package.
keepcache	If set to 0, remove packages after installation.
logfile	Pathname of the yum log file.
obsoletes	If set to 1, replace obsolete packages during upgrades.
plugins	If set to 1, enable plugins that extend the functionality of yum.
proxy	URL of a proxy server including the port number. See Section 2.3.1, “Configuring Use of a Proxy Server”.
proxy_password	Password for authentication with a proxy server.
proxy_username	User name for authentication with a proxy server.
reposdir	Directories where yum should look for repository files with a .repo extension. The default directory is /etc/yum.repos.d.

See the yum.conf(5) manual page for more information.

The following listing shows an example [main] section from the yum configuration file.

```
[main]
cachedir=/var/cache/yum
keepcache=0
debuglevel=2
logfile=/var/log/yum.log
exactarch=1
obsoletes=1
gpgkey=file://media/RPM-GPG-KEY
gpgcheck=1
pligins=1
installonly_limit=3
```

It is possible to define repositories below the [main] section in /etc/yum.conf or in separate repository configuration files. By default, yum expects any repository configuration files to be located in the /etc/yum.repos.d directory unless you use the reposdir directive to define alternate directories.

2.3.1 Configuring Use of a Proxy Server

If your organization uses a proxy server as an intermediary for Internet access, specify the proxy setting in /etc/yum.conf as shown in the following example.

```
proxy=http://proxysvr.example.com:3128
```

If the proxy server requires authentication, additionally specify the proxy_username, and proxy_password settings.

```
proxy=http://proxysvr.example.com:3128
proxy_username=yumacc
proxy_password=clydenw
```

If you use the yum plugin (`yum-rhn-plugin`) to access the ULN, specify the `enableProxy` and `httpProxy` settings in `/etc/sysconfig/rhn/up2date` as shown in this example.

```
enableProxy=1
httpProxy=http://proxysvr.example.com:3128
```

If the proxy server requires authentication, additionally specify the `enableProxyAuth`, `proxyUser`, and `proxyPassword` settings.

```
enableProxy=1
httpProxy=http://proxysvr.example.com:3128
enableProxyAuth=1
proxyUser=yumacc
proxyPassword=clydenw
```

Caution

All `yum` users require read access to `/etc/yum.conf` or `/etc/sysconfig/rhn/up2date`. If these files must be world-readable, do not use a proxy password that is the same as any user's login password, and especially not `root`'s password.

2.3.2 Yum Repository Configuration

The yum configuration file or yum repository configuration files can contain one or more sections that define repositories.

The following table lists the basic directives for a repository.

Directive	Description
`baseurl`	Location of the repository channel (expressed as a `file://`, `ftp://`, `http://`, or `https://` address). This directive must be specified.
`enabled`	If set to 1, permit `yum` to use the channel.
`name`	Descriptive name for the repository channel. This directive must be specified.

Any other directive that appears in this section overrides the corresponding global definition in `[main]` section of the yum configuration file. See the `yum.conf(5)` manual page for more information.

The following listing shows an example repository section from a configuration file.

```
[ol6_u2_base]
name=Oracle Linux 6 U2 - $basearch - base
baseurl=https://yum.oracle.com/repo/OracleLinux/OL6/2/base/$basearch
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
gpgcheck=1
enabled=1
```

In this example, the values of `gpgkey` and `gpgcheck` override any global setting. `yum` substitutes the name of the current system's architecture for the variable *`$basearch`*.

`yum` automatically searches the `/etc/yum.repos.d` directory for files with the suffix `.repo` and appends these to the configuration when it is processing. Use this directory to define repository files for repositories that you want to make available.

2.3.3 Downloading the Oracle Linux Yum Server Repository Files

The Oracle Linux yum server provides a direct mapping of all of the Unbreakable Linux Network (ULN) channels that are available to the public without any specific support agreement. The repository labels used for each repository on the Oracle Linux yum server map directly onto the channel names on ULN. See the *Oracle Linux Unbreakable Linux Network User's Guide* for more information about the channel names and common suffixes used for channels and repositories.

Prior to January 2019, Oracle shipped a single yum repository configuration file for each Oracle Linux release. This configuration file is copied into `/etc/yum.repos.d/public-yum-ol6.repo` at installation, but can also be downloaded from the Oracle Linux yum server directly to obtain updates.

The original configuration file is deprecated in favor of modular repository files that are managed and updated automatically via yum in the form of RPM packages that are more targeted in scope. For example, core repository configuration files required for Oracle Linux 6 are available in the `oraclelinux-release-el6` package. This package includes all of the repository configuration required to install base packages for the release, including packages from the `ol6_latest`, `ol6_addons` repositories and all of the supported repositories for UEK.

The modular yum repository configuration files released as packages that can be maintained via yum can help to simplify repository management and also ensure that your yum repository definitions are kept up to date automatically, whenever you update your system.

A list of all available RPM files to manage all of the possible yum repository configurations for your release can be obtained by running:

```
# yum list *release-el6*
```

To install the yum repository configuration for a particular set of software that you wish to use, use yum to install the corresponding package. For example, to install the yum repository configuration for the Oracle Linux Software Collection Library, run:

```
# yum install oracle-softwarecollection-release-el6
```

If your system is still configured to use the original single yum repository configuration file at `/etc/yum.repos.d/public-yum-ol6.repo`, you should update your system to transition to the current approach to handling yum repository configuration. To do this, ensure that your system is up to date and then run the `/usr/bin/ol_yum_configure.sh` script:

```
# yum update
# /usr/bin/ol_yum_configure.sh
```

The `/usr/bin/ol_yum_configure.sh` script checks the `/etc/yum.repos.d/public-yum-ol6.repo` file to determine which repositories are already enabled and installs the appropriate corresponding packages before renaming the original configuration file to `/etc/yum.repos.d/public-yum-ol6.repo.sav` to disable it in favor of the more recent modular repository configuration files.

If, for some reason, you manage to remove all configuration to access the Oracle Linux yum server repositories, you should create a temporary yum repository configuration file at `/etc/yum.repos.d/ol6-temp.repo` with the following as the minimum required content:

```
[ol6_latest]
name=Oracle Linux $releasever Latest ($basearch)
baseurl=https://yum.oracle.com/repo/OracleLinux/OL6/latest/$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
gpgcheck=1
```

```
enabled=1
```

Then reinstall the `oraclelinux-release-el6` package to restore the default yum configuration:

```
# yum reinstall oraclelinux-release-el6
# rm /etc/yum.repos.d/ol6-temp.repo
```

For more information on manually setting up Oracle Linux yum server repository configuration files, see https://yum.oracle.com/getting-started.html.

You can enable or disable repositories in each repository configuration file by setting the value of the `enabled` directive to 1 or 0 for each repository listed in the file, as required. The preferred method of enabling or disabling repositories under Oracle Linux 6 is to use the `yum-config-manager` command provided in the `yum-utils` package.

2.3.4 Using Yum Utilities to Manage Configuration

The `yum-utils` package includes several utilities that can help you to manage configuration and apply updates safely to your existing configuration. Most significant of these is `yum-config-manager`.

To install the `yum-utils` package:

```
# yum install -y yum-utils
```

You can use `yum-config-manager` to add repositories either at a specified URL, or within a specified repository file. For example, to add the legacy repository configuration file for Oracle Linux 6 from the Oracle Linux yum server:

```
# yum-config-manager --add-repo https://yum.oracle.com/public-yum-ol6.repo
```

Note

The legacy repository configuration file is unmaintained and deprecated. The information in this file may not be current and newer repositories may not be listed.

You can use the same command to automatically generate a repository configuration file for a valid yum repository, by pointing to the URL where the repository is hosted. For example, to create a new configuration file in `/etc/repos.d` for the Unbreakable Enterprise Kernel Release 4 repository. run:

```
# yum-config-manager --add-repo https://yum.oracle.com/repo/OracleLinux/OL6/UEKR4/x86_64
```

To enable a repository using `yum-config-manager`, use the `--enable` option. For example, to enable the `ol6_addons` repository, run:

```
# yum-config-manager --enable ol6_addons
```

You can use the `--disable` option in a similar way to disable a repository.

The `yum-config-manager` tool can also be used to set other configuration options using the `--setopt` and `--save` options. See the `yum-config-manager(1)` manual page for more information.

For a list of the tools included in the yum-utils package and a description of what these tools can do, see the `yum-utils(1)` manual page for more information.

2.4 Using Yum from the Command Line

The following table shows some examples of common tasks that you can perform using `yum`.

Command	Description
`yum repolist`	Lists all enabled repositories.
`yum list`	Lists all packages that are available in all enabled repositories and all packages that are installed on your system.
`yum list installed`	Lists all packages that are installed on your system.
`yum list available`	Lists all packages that are available to be installed in all enabled repositories.
`yum search` *string*	Searches the package descriptions for the specified string.
`yum provides` *feature*	Finds the name of the package to which the specified file or feature belongs. For example: `yum provides /etc/sysconfig/atd`
`yum info` *package*	Displays detailed information about a package. For example: `yum info bind`
`yum install` *package*	Installs the specified package, including packages on which it depends. For example: `yum install ocfs2-tools`
`yum check-update`	Checks whether updates exist for packages that are already installed on your system.
`yum update` *package*	Updates the specified package, including packages on which it depends. For example: `yum upgrade nfs-utils`
`yum update`	Updates all packages, including packages on which they depend.
`yum remove` *package*	Removes the specified package. For example: `yum erase nfs-utils`
`yum erase` *package*	Removes the specified package. This command has the same effect as the `yum remove` command.
`yum update`	Updates all packages, including packages on which they depend.
`yum clean all`	Removes all cached package downloads and cached headers that contain information about remote packages. Running this command can help to clear problems that can result from unfinished transactions or out-of-date headers.
`yum help`	Displays help about `yum` usage.
`yum help` *command*	Displays help about the specified `yum` command. For example: `yum help upgrade`
`yum shell`	Runs the `yum` interactive shell.

See the `yum(8)` manual page for more information.

To list the files in a package, use the `repoquery` utility, which is included in the `yum-utils` package. For example, the following command lists the files that the `btrfs-progs` package provides.

```
# repoquery -l btrfs-progs
/sbin/btrfs
```

```
/sbin/btrfs-convert
/sbin/btrfs-debug-tree
.
.
.
```

Note

yum makes no distinction between installing and upgrading a kernel package. yum always installs a new kernel regardless of whether you specify update or install.

2.5 Yum Groups

A set of packages can themselves be organized as a *yum group*. Examples include the groups for Eclipse, fonts, and system administration tools. The following table shows the yum commands that you can use to manage these groups.

Command	Description
yum grouplist	Lists installed groups and groups that are available for installation.
yum groupinfo *groupname*	Displays detailed information about a group.
yum groupinstall *groupname*	Installs all the packages in a group.
yum groupupdate *groupname*	Updates all the packages in a group.
yum groupremove *groupname*	Removes all the packages in a group.

2.6 Installing and Using the Yum Security Plugin

The yum-plugin-security package allows you to use yum to obtain a list of all of the errata that are available for your system, including security updates. You can also use Oracle Enterprise Manager 12c Cloud Control or management tools such as Katello, Pulp, Red Hat Satellite, Spacewalk, and SUSE Manager to extract and display information about errata.

To install the yum-plugin-security package, enter the following command:

```
# yum install yum-plugin-security
```

To list the errata that are available for your system, enter:

```
# yum updateinfo list
Loaded plugins: refresh-packagekit, rhnplugin, security
ELBA-2012-1518 bugfix         NetworkManager-1:0.8.1-34.el6_3.x86_64
ELBA-2012-1518 bugfix         NetworkManager-glib-1:0.8.1-34.el6_3.x86_64
ELBA-2012-1518 bugfix         NetworkManager-gnome-1:0.8.1-34.el6_3.x86_64
ELBA-2012-1457 bugfix         ORBit2-2.14.17-3.2.el6_3.x86_64
ELBA-2012-1457 bugfix         ORBit2-devel-2.14.17-3.2.el6_3.x86_64
ELSA-2013-0215 Important/Sec. abrt-2.0.8-6.0.1.el6_3.2.x86_64
ELSA-2013-0215 Important/Sec. abrt-addon-ccpp-2.0.8-6.0.1.el6_3.2.x86_64
ELSA-2013-0215 Important/Sec. abrt-addon-kerneloops-2.0.8-6.0.1.el6_3.2.x86_64
ELSA-2013-0215 Important/Sec. abrt-addon-python-2.0.8-6.0.1.el6_3.2.x86_64
ELSA-2013-0215 Important/Sec. abrt-cli-2.0.8-6.0.1.el6_3.2.x86_64
ELSA-2013-0215 Important/Sec. abrt-desktop-2.0.8-6.0.1.el6_3.2.x86_64
...
```

The output from the command sorts the available errata in order of their IDs, and it also specifies whether each erratum is a security patch (*severity*/Sec.), a bug fix (bugfix), or a feature enhancement (enhancement). Security patches are listed by their severity: Important, Moderate, or Low.

You can use the `--sec-severity` option to filter the security errata by severity, for example:

```
# yum updateinfo list --sec-severity=Moderate
Loaded plugins: refresh-packagekit, rhnplugin, security
ELSA-2013-0269 Moderate/Sec. axis-1.2.1-7.3.el6_3.noarch
ELSA-2013-0668 Moderate/Sec. boost-1.41.0-15.el6_4.x86_64
ELSA-2013-0668 Moderate/Sec. boost-date-time-1.41.0-15.el6_4.x86_64
ELSA-2013-0668 Moderate/Sec. boost-devel-1.41.0-15.el6_4.x86_64
ELSA-2013-0668 Moderate/Sec. boost-filesystem-1.41.0-15.el6_4.x86_64
ELSA-2013-0668 Moderate/Sec. boost-graph-1.41.0-15.el6_4.x86_64
ELSA-2013-0668 Moderate/Sec. boost-iostreams-1.41.0-15.el6_4.x86_64
ELSA-2013-0668 Moderate/Sec. boost-program-options-1.41.0-15.el6_4.x86_64
ELSA-2013-0668 Moderate/Sec. boost-python-1.41.0-15.el6_4.x86_64
...
```

To list the security errata by their Common Vulnerabilities and Exposures (CVE) IDs instead of their errata IDs, specify the keyword `cves` as an argument:

```
# yum updateinfo list cves
Loaded plugins: refresh-packagekit, rhnplugin, security
 CVE-2012-5659 Important/Sec. abrt-2.0.8-6.0.1.el6_3.2.x86_64
 CVE-2012-5660 Important/Sec. abrt-2.0.8-6.0.1.el6_3.2.x86_64
 CVE-2012-5659 Important/Sec. abrt-addon-ccpp-2.0.8-6.0.1.el6_3.2.x86_64
 CVE-2012-5660 Important/Sec. abrt-addon-ccpp-2.0.8-6.0.1.el6_3.2.x86_64
 CVE-2012-5659 Important/Sec. abrt-addon-kerneloops-2.0.8-6.0.1.el6_3.2.x86_64
 CVE-2012-5660 Important/Sec. abrt-addon-kerneloops-2.0.8-6.0.1.el6_3.2.x86_64
 CVE-2012-5659 Important/Sec. abrt-addon-python-2.0.8-6.0.1.el6_3.2.x86_64
 CVE-2012-5660 Important/Sec. abrt-addon-python-2.0.8-6.0.1.el6_3.2.x86_64
...
```

Similarly, the keywords `bugfix`, `enhancement`, and `security` filter the list for all bug fixes, enhancements, and security errata.

You can use the `--cve` option to display the errata that correspond to a specified CVE, for example:

```
# yum updateinfo list --cve CVE-2012-2677
Loaded plugins: refresh-packagekit, rhnplugin, security
ELSA-2013-0668 Moderate/Sec. boost-1.41.0-15.el6_4.x86_64
ELSA-2013-0668 Moderate/Sec. boost-date-time-1.41.0-15.el6_4.x86_64
ELSA-2013-0668 Moderate/Sec. boost-devel-1.41.0-15.el6_4.x86_64
ELSA-2013-0668 Moderate/Sec. boost-filesystem-1.41.0-15.el6_4.x86_64
ELSA-2013-0668 Moderate/Sec. boost-graph-1.41.0-15.el6_4.x86_64
ELSA-2013-0668 Moderate/Sec. boost-iostreams-1.41.0-15.el6_4.x86_64
ELSA-2013-0668 Moderate/Sec. boost-program-options-1.41.0-15.el6_4.x86_64
ELSA-2013-0668 Moderate/Sec. boost-python-1.41.0-15.el6_4.x86_64
ELSA-2013-0668 Moderate/Sec. boost-regex-1.41.0-15.el6_4.x86_64
ELSA-2013-0668 Moderate/Sec. boost-serialization-1.41.0-15.el6_4.x86_64
ELSA-2013-0668 Moderate/Sec. boost-signals-1.41.0-15.el6_4.x86_64
ELSA-2013-0668 Moderate/Sec. boost-system-1.41.0-15.el6_4.x86_64
ELSA-2013-0668 Moderate/Sec. boost-test-1.41.0-15.el6_4.x86_64
ELSA-2013-0668 Moderate/Sec. boost-thread-1.41.0-15.el6_4.x86_64
ELSA-2013-0668 Moderate/Sec. boost-wave-1.41.0-15.el6_4.x86_64
updateinfo list done
```

To display more information, specify `info` instead of `list`, for example:

```
# yum updateinfo info --cve CVE-2012-2677
Loaded plugins: refresh-packagekit, rhnplugin, security
===============================================================================
   boost security update
===============================================================================
  Update ID : ELSA-2013-0668
    Release : Oracle Linux 6
       Type : security
     Status : final
```

```
     Issued : 2013-03-21
       CVEs : CVE-2012-2677
Description : [1.41.0-15]
            : - Add in explicit dependences between some boost
            :   subpackages
            :
            : [1.41.0-14]
            : - Build with -fno-strict-aliasing
            :
            : [1.41.0-13]
            : - In Boost.Pool, be careful not to overflow
            :   allocated chunk size (boost-1.41.0-pool.patch)
            :
            : [1.41.0-12]
            : - Add an upstream patch that fixes computation of
            :   CRC in zlib streams.
            : - Resolves: #707624
   Severity : Moderate
updateinfo info done
```

To update all packages for which security-related errata are available to the latest versions of the packages, even if those packages include bug fixes or new features but not security errata, enter:

```
# yum --security update
```

To update all packages to the latest versions that contain security errata, ignoring any newer packages that do not contain security errata, enter:

```
# yum --security update-minimal
```

To update all kernel packages to the latest versions that contain security errata, enter:

```
# yum --security update-minimal kernel*
```

You can also update only those packages that correspond to a CVE or erratum, for example:

```
# yum update --cve CVE-2012-3954

# yum update --advisory ELSA-2012-1141
```

Note

Some updates might require you to reboot the system. By default, the boot manager will automatically enable the most recent kernel version.

For more information, see the `yum-security(8)` manual page.

2.7 Switching CentOS or Scientific Linux Systems to Use the Oracle Linux Yum Server

You can use the `centos2ol.sh` script to convert CentOS 5 and 6 or Scientific Linux 5 and 6 systems to Oracle Linux. The script configures `yum` to use the Oracle Linux yum server and installs a few additional packages that are required. There is no need to reboot the system.

To perform the switch to Oracle Linux, run the following commands as `root`:

```
# curl -O https://linux.oracle.com/switch/centos2ol.sh
# sh centos2ol.sh
```

For more information, see https://linux.oracle.com/switch/centos/.

2.8 Creating and Using a Local ULN Mirror

For information on how to create and use a yum server that acts as a local mirror of the ULN channels, see Creating and Using a Local ULN Mirror in the *Oracle Linux Unbreakable Linux Network User's Guide*.

2.9 Creating a Local Yum Repository Using an ISO Image

Note

The system must have sufficient storage space to host a full Oracle Linux Media Pack DVD image (approximately 3.5 GB for Oracle Linux Release 6 Update 3).

To create a local yum repository (for example, if a system does not have Internet access):

1. On a system with Internet access, download a full Oracle Linux DVD image from the Oracle Software Delivery Cloud at https://edelivery.oracle.com/linux onto removable storage (such as a USB memory stick). For example, `V33411-01.iso` contains the Oracle Linux Release 6 Update 3 Media Pack for x86 (64 bit).

   Note

   You can verify that the ISO was copied correctly by comparing its checksum with the digest value that is listed on `edelivery.oracle.com`, for example:

   ```
   # sha1sum V33411-01.iso
   7daae91cc0437f6a98a4359ad9706d678a9f19de V33411-01.iso
   ```

2. Transfer the removable storage to the system on which you want to create a local yum repository, and copy the DVD image to a directory in a local file system.

   ```
   # cp /media/USB_stick/V33411-01.iso /ISOs
   ```

3. Create a suitable mount point, for example `/var/OSimage/OL6.3_x86_64`, and mount the DVD image on it.

   ```
   # mkdir -p /var/OSimage/OL6.3_x86_64
   # mount -o loop,ro /ISOs/V33411-01.iso /var/OSimage/OL6.3_x86_64
   ```

   Note

   Include the read-only mount option (`ro`) to avoid changing the contents of the ISO by mistake.

4. Create an entry in `/etc/fstab` so that the system always mounts the DVD image after a reboot.

   ```
   /ISOs/V33411-01.iso /var/OSimage/OL6.3_x86_64 iso9660 loop,ro 0 0
   ```

5. Disable all existing yum repositories.

   In the `/etc/yum.repos.d` directory, edit any existing repository files and disable all entries by setting `enabled=0`. If you have the `yum-utils` package installed, as described in Section 2.3.4, "Using Yum Utilities to Manage Configuration", you can disable all repositories by running:

   ```
   # yum-config-manager --disable \*
   ```

6. Create the following entries in a new repository file (for example, `/etc/yum.repos.d/OL63.repo`).

   ```
   [OL63]
   name=Oracle Linux 6.3 x86_64
   ```

```
baseurl=file:///var/OSimage/OL6.3_x86_64
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
gpgcheck=1
enabled=1
```

7. Clean up the yum cache.

```
# yum clean all
```

8. Test that you can use yum to access the repository.

```
# yum repolist
Loaded plugins: refresh-packagekit, security
...
repo id                           repo name                                 status
OL63                              Oracle Linux 6.3 x86_64                   25,459
repolist: 25,459
```

2.10 Setting up a Local Yum Server Using an ISO Image

To set up a local yum server (for example, if you have a network of systems that do not have Internet access):

1. Choose one of the systems to be the yum server, and create a local yum repository on it as described in Section 2.9, “Creating a Local Yum Repository Using an ISO Image”.

2. Install the Apache HTTP server from the local yum repository.

```
# yum install httpd
```

3. If SELinux is enabled in enforcing mode on your system:

   a. Use the semanage command to define the default file type of the repository root directory hierarchy as httpd_sys_content_t:

   ```
   # /usr/sbin/semanage fcontext -a -t httpd_sys_content_t "/var/OSimage(/.*)?"
   ```

   b. Use the restorecon command to apply the file type to the entire repository.

   ```
   # /sbin/restorecon -R -v /var/OSimage
   ```

Note

The semanage and restorecon commands are provided by the policycoreutils-python and policycoreutils packages.

4. Create a symbolic link in /var/www/html that points to the repository:

```
# ln -s /var/OSimage /var/www/html/OSimage
```

5. Edit the HTTP server configuration file, /etc/httpd/conf/httpd.conf, as follows:

   a. Specify the resolvable domain name of the server in the argument to ServerName.

   ```
   ServerName server_addr:80
   ```

   If the server does not have a resolvable domain name, enter its IP address instead.

   b. Verify that the setting of the Options directive in the <Directory "/var/www/html"> section specifies Indexes and FollowSymLinks to allow you to browse the directory hierarchy, for example:

```
Options Indexes FollowSymLinks
```

c. Save your changes to the file.

6. Start the Apache HTTP server, and configure it to start after a reboot.

```
# service httpd start
# chkconfig httpd on
```

7. If you have enabled a firewall on your system, configure it to allow incoming HTTP connection requests on TCP port 80.

   For example, the following command configures `iptables` to allow incoming HTTP connection requests and saves the change to the firewall configuration:

```
# iptables -I INPUT -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT
# service iptables save
```

8. Disable all existing yum repositories on the server and each client system.

   In the `/etc/yum.repos.d` directory, edit any existing repository files and disable all entries by setting `enabled=0`. If you have the `yum-utils` package installed, as described in Section 2.3.4, “Using Yum Utilities to Manage Configuration”, you can disable all repositories by running:

```
# yum-config-manager --disable \*
```

9. Edit the repository file on the server (for example, `/etc/yum.repos.d/OL63.repo`):

```
[OL63]
name=Oracle Linux 6.3 x86_64
baseurl=http://server_addr/OSimage/OL6.3_x86_64
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
gpgcheck=1
enabled=1
```

   Replace *`server_addr`* with the IP address or resolvable host name of the local yum server.

10. On each client, copy the repository file from the server to the `/etc/yum.repos.d` directory.

11. On the server and each client, test that you can use `yum` to access the repository.

```
# yum repolist
Loaded plugins: refresh-packagekit, security
...
repo id                          repo name                                status
OL63                             Oracle Linux 6.3 x86_64                  25,459
repolist: 25,459
```

2.11 For More Information About Yum

For more information about yum, see http://yum.baseurl.org/.

Frequently asked questions about the Oracle Linux yum server are answered at https://yum.oracle.com/faq.html.

For more information about how to download the latest packages from the Unbreakable Linux Network and make the packages available through a local yum server, see https://www.oracle.com/technetwork/articles/servers-storage-admin/yum-repo-setup-1659167.html.

Chapter 3 Ksplice

Table of Contents

This chapter provides a high-level overview of Oracle Ksplice. For detailed information and instructions, see the *Oracle Linux Ksplice User's Guide*.

3.1 Overview of Oracle Ksplice

Linux systems receive regular security updates to core operating system components that necessitate patching and rebooting. Traditionally, applying such updates would require you to obtain and install the updated RPMs, schedule downtime, and reboot the server to the new package version, with any critical updates. However, as system setups become more complex, with many interdependencies, access to services and applications must remain as undisrupted as possible, as scheduling such reboots becomes more difficult and costly.

Oracle Ksplice provides a way for you to keep your systems secure and highly available by enabling you to update them with the latest kernel and key user-space security and bug fix updates, and Xen hypervisor updates on Oracle VM Server 3.4.5 and later.

Note

When using Ksplice to patch the Xen hypervisor on Oracle VM Server 3.4.5 and later, the minimum version that is required is `xen-4.4.4-196.el6.x86_64.rpm`.

Oracle Ksplice updates the running operating system without requiring a reboot. Your systems remains up to date with OS vulnerability patches and downtime is minimized. A Ksplice update takes effect immediately upon application. Note that a Ksplice update is not the same as an on-disk change that requires a subsequent reboot to take effect. However, note that on-disk updates are still required when using Ksplice to ensure that package binaries are updated to the most recent version and can be used in the event that the system or processes are restarted. On-disk updates are handled by subscribing to the Unbreakable Linux Network (ULN) or by using a local ULN mirror.

Oracle creates each Ksplice update from a package update that originates either from Oracle or the open source community.

To learn more about Ksplice, go to http://www.ksplice.com/.

3.1.1 Supported Kernels

You can use Ksplice to bring the following Oracle Linux kernels up to date with the latest important security and bug fix patches:

- All Oracle Unbreakable Enterprise Kernel versions for Oracle Linux 5 and Oracle Linux 6 starting with 2.6.32-100.28.9 (released March 16, 2011).

- All Oracle Linux 6 kernels starting with the official release.
- All Oracle Linux 5 Red Hat Compatible Kernels starting with Oracle Linux 5.4 (2.6.18-164.el5, released September 9, 2009).
- All Oracle Linux 5 Red Hat Compatible Kernels with bug fixes added by Oracle starting with Oracle Linux 5.6 (2.6.18-238.0.0.0.1.el5, released January 22, 2011).

To confirm whether a particular kernel is supported, install the Ksplice Uptrack client or Ksplice Enhanced Client on a system that is running the kernel.

Note

If your system is currently running Red Hat Enterprise Linux and you have recently migrated to Oracle Linux Premier Support, you can use Ksplice to update the existing Red Hat Enterprise Linux kernel. You do not need to switch to the Red Hat Compatible Kernel to use Ksplice kernel patches. These patches are available on ULN as `uptrack-updates-kernel_version` packages in the Ksplice for Oracle Linux channels.

For questions about supported kernels, send e-mail to ksplice-support_ww@oracle.com.

3.1.2 About Ksplice Updates

When a critical bug or security vulnerability is discovered in the Linux kernel, Oracle produces a new kernel release and prepares a rebootless update corresponding to that release. The rebootless update is securely distributed using the Oracle Ksplice Uptrack server and the Unbreakable Linux Network (ULN) and is applied to your systems by the Ksplice Uptrack client or Ksplice Enhanced client with zero downtime. Your infrastructure is again up to date and secure.

For more detailed information, see *About Ksplice Updates* in Chapter 1 of the *Oracle Linux Ksplice User's Guide*.

3.1.3 Patching and Updating Your System

Ksplice patches enable you to keep a system up to date while it is running. You should also use the `yum` command to install the regular kernel RPM packages for released errata that are available from the Unbreakable Linux Network (ULN) or the Oracle Linux yum server. Your system will then be ready for the next maintenance window or reboot. When you restart the system, you can boot it from a newer kernel version. Ksplice Uptrack uses the new kernel as a baseline for applying patches as they become available.

For more detailed information, see *Patching and Updating Your System* in Chapter 1 of the *Oracle Linux Ksplice User's Guide*.

3.2 About the Ksplice Client Software

This section describes the different Ksplice client software types that are available in Oracle Linux. A description of each Ksplice client type, as well as information about when you might use each client, is provided.

3.2.1 About the Ksplice Enhanced Client

The Ksplice Enhanced client is available for Oracle Linux 6, but not Oracle Linux 5. The enhanced version of the Ksplice online client supports kernel and user-space updates and can also be used to patch the Xen hypervisor on Oracle VM Server Release 3.4.5 and later.

Note

To use Ksplice to patch the Xen hypervisor on Oracle VM 3.4.5 and later, the minimum Xen hypervisor version is `xen-4.4.4-196.el6.x86_64.rpm`.

The Ksplice Enhanced client can patch in-memory pages of Ksplice aware shared libraries such as `glibc` and `openssl` for user-space processes, in addition to the kernel updates applied by the traditional Ksplice Uptrack client. User-space patching enables you to install bug fixes and protect your system against security vulnerabilities without having to restart processes and services. Both an online and an offline version of the enhanced client are available.

You manage the Ksplice Enhanced client by using the `ksplice` command rather than `uptrack` commands. Note that the enhanced client shares the same configuration file as the Uptrack client, which is located at `/etc/uptrack/uptrack.conf`. For more information, see *Working With the Ksplice Enhanced Client*

The offline version of the Ksplice Enhanced client removes the requirement that a server on your intranet have a direct connection to the Oracle Uptrack server or to ULN. All available Ksplice updates for each supported kernel version or user-space package are bundled into an RPM that is specific to that version. This package is updated every time a new Ksplice patch becomes available for the kernel. In this way, you can create a local ULN mirror that acts as a mirror for the Ksplice aware channels for Oracle Linux on ULN.

At regular intervals, you can download the latest Ksplice update packages to this server. After installing the offline Ksplice Enhanced client on your local systems, they can then connect to the local ULN mirror to receive updates.

For information about when you might want to use the Ksplice Enhanced client in offline mode, see Section 3.3, "Choosing a Ksplice Client".

When you have set up a local ULN mirror to act as a Ksplice mirror, you can then configure your other systems to receive `yum` updates, as well as Ksplice updates. For task-related information, see *Installing and Configuring the Ksplice Offline Enhanced Client*.

3.2.2 About the Ksplice Uptrack Client

The Ksplice Uptrack client enables you to apply the latest kernel security errata for Common Vulnerabilities and Exposures (CVEs) without halting the system or restarting any applications. Ksplice Uptrack applies the updated patches in the background with negligible impact, and usually only requires a pause of a few milliseconds. You can use Ksplice Uptrack as well as continue to upgrade your kernel through the usual mechanism, such as running the `yum` command.

Ksplice Uptrack is freely available for Oracle customers who subscribe to Oracle Linux Premier Support, and to Oracle Cloud Infrastructure services. If you are an Oracle Linux Basic, Basic Limited, or Network Support subscriber, contact your sales representatives to discuss a potential upgrade of your subscription to a Premier Support plan.

The Ksplice Offline client removes the requirement that a server on your intranet have a direct connection to the Oracle Uptrack server. All available Ksplice updates for each supported kernel version are bundled into an RPM that is specific to that version. This package is updated every time a new Ksplice patch becomes available for the kernel.

A Ksplice Offline client does not require a network connection to be able to apply the update package to the kernel. For example, you could use the `yum` command to install the update package directly from a memory stick. However, a more typical method would be to create a local ULN mirror that acts as a mirror of the Ksplice for Oracle Linux channels on ULN. At regular intervals, you download the latest Ksplice update packages to this server. After installing the Ksplice Offline client on your local systems, the systems can connect to the local ULN mirror to receive updates without requiring access to the Oracle Uptrack server. See *Configuring a Local ULN Mirror to Act as a Ksplice Mirror* for more information.

For information about when you might want to use the Ksplice Uptrack client in offline mode, see Section 3.3, “Choosing a Ksplice Client”.

Note

You cannot use the web interface or the Ksplice Uptrack API to monitor systems that are running Ksplice Offline client, as such systems are not registered with https://uptrack.ksplice.com.

3.3 Choosing a Ksplice Client

To determine which Ksplice client will best suit your needs, refer to information that is described in the following table.

Ksplice Client	User Space Support	Xen Hypervisor Patching Support	Legacy Compatibility
Ksplice Enhanced Client	Supported	Supported	Not supported
Ksplice Uptrack Client	Not supported	Supported	Supported

3.4 Preparing to Use Oracle Ksplice

Refer to the following information before installing and configuring Ksplice:

- Determine which Ksplice client will best suit your needs. Depending on which Ksplice client you are using, you might be required to perform additional tasks. See Section 3.3, “Choosing a Ksplice Client”.
- Register your system with the Unbreakable Linux Network (ULN).

  To use the Oracle Ksplice, your system must have access to the Internet, and you must register your system with the Unbreakable Linux Network (ULN) first, unless the system is configured to use the Oracle Ksplice client as an offline client. If your client is configured to function as an offline client, you must configure a local ULN mirror that the client can access to receive updates. For more information, see *Oracle Linux Unbreakable Linux Network User's Guide*.
- Ensure that you have a valid Oracle Linux Premier, Premier Limited, or Oracle Premier Support for Systems and Operating Systems subscription and a valid Customer Support Identifier (CSI).

  If you have one of the Oracle Linux Premier subscriptions previously mentioned and a valid CSI, your account is automatically registered to use the Ksplice Uptrack server. You can log in to the Ksplice Uptrack server web interface at https://uptrack.ksplice.com by using your Oracle Single Sign-on (SSO) credentials. After logging into the server, you can view the status of your registered systems, the patches that have been applied, and the patches that are available. For more detailed information about Ksplice and ULN registration, see the *Oracle Linux Ksplice User's Guide*.
- If you plan to use an offline mode of either the Ksplice Enhanced client or the Ksplice Uptrack client, you must set up a local ULN mirror that the client can access to receive updates. See *Configuring a Local ULN Mirror to Act as a Ksplice Mirror* for task-related information.
- Using Ksplice with Spacewalk also requires that you set up a local ULN mirror. For further details, see *Configuring a Spacewalk Server to Act as a Ksplice Mirror*.

Chapter 4 The Btrfs File System

Table of Contents

This chapter describes how to deploy and use the advanced features of the btrfs file system.

4.1 About the Btrfs File System

The btrfs file system is designed to meet the expanding scalability requirements of large storage subsystems. As the btrfs file system uses B-trees in its implementation, its name derives from the name of those data structures, although it is not a true acronym. A B-tree is a tree-like data structure that enables file systems and databases to efficiently access and update large blocks of data no matter how large the tree grows.

The btrfs file system provides the following important features:

- Copy-on-write functionality allows you to create both readable and writable snapshots, and to roll back a file system to a previous state, even after you have converted it from an `ext3` or `ext4` file system.

- Checksum functionality ensures data integrity.

- Transparent compression saves disk space.
- Transparent defragmentation improves performance.
- Integrated logical volume management allows you to implement RAID 0, RAID 1, or RAID 10 configurations, and to dynamically add and remove storage capacity.

Starting with Oracle Linux 6 Update 3, the UEK Boot ISO (which boots the Unbreakable Enterprise Kernel as the installation kernel) allows you to configure a btrfs root file system. Prior to Oracle Linux 6 Update 3, you could not create a btrfs root file system during installation. For more information, see Section 4.12, “Installing a Btrfs root File System”.

With UEK R3, btrfs supports the following additional features:

- The send/receive feature allows you to record the differences between two subvolumes, which can either be snapshots of the same subvolume or parent and child subvolumes.
- Quota groups (*qgroups*) allow you to set different size limits for a volume and its subvolumes.
- You can replace devices without unmounting or otherwise disrupting access to the file system.

4.2 Creating a Btrfs File System

> **Note**
>
> If the `btrfs-progs` package is not already installed on your system, use `yum` to install it.

You can use the `mkfs.btrfs` command to create a btrfs file system that is laid out across one or more block devices. The default configuration is to stripe the file system data and to mirror the file system metadata across the devices. If you specify a single device, the metadata is duplicated on that device unless you specify that only one copy of the metadata is to be used. The devices can be simple disk partitions, loopback devices (that is, disk images in memory), multipath devices, or LUNs that implement RAID in hardware.

The following table illustrates how to use the `mkfs.btrfs` command to create various btrfs configurations.

Command	Description
`mkfs.btrfs` *`block_device`*	Create a btrfs file system on a single device. For example: `mkfs.btrfs /dev/sdb1`
`mkfs.btrfs -L` *`label block_device`*	Create a btrfs file system with a label that you can use when mounting the file system. For example: `mkfs.btrfs -L myvolume /dev/sdb2` **Note** The device must correspond to a partition if you intend to mount it by specifying the name of its label.
`mkfs.btrfs -m single` *`block_device`*	Create a btrfs file system on a single device, but do not duplicate the metadata on that device. For example: `mkfs.btrfs -m single /dev/sdc`

Command	Description
`mkfs.btrfs` *`block_device1`* *`block_device2`* ...	Stripe the file system data and mirror the file system metadata across several devices. For example: `mkfs.btrfs /dev/sdd /dev/sde`
`mkfs.btrfs -m raid0` *`block_device1`* *`block_device2`* ...	Stripe both the file system data and metadata across several devices. For example: `mkfs.btrfs -m raid0 /dev/sdd /dev/sde`
`mkfs.btrfs -d raid1` *`block_device1`* *`block_device2`* ...	Mirror both the file system data and metadata across several devices. For example: `mkfs.btrfs -d raid1 /dev/sdd /dev/sde`

`mkfs.btrfs -d raid10 -m raid10` *`block_device1`* *`block_device2`* *`block_device3`* *`block_device4`*	Stripe the file system data and metadata across several mirrored devices. You must specify an even number of devices, of which there must be at least four. For example:
	
	`mkfs.btrfs -d raid10 -m raid10 /dev/sdf \`
	
	`/dev/sdg /dev/sdh /dev/sdi /dev/sdj /dev/sdk`

When you want to mount the file system, you can specify it by any of its component devices, for example:

```
# mkfs.btrfs -d raid10 -m raid10 /dev/sd[fghijk]
# mount /dev/sdf /raid10_mountpoint
```

To find out the RAID configuration of a mounted btrfs file system, use this command:

```
# btrfs filesystem df mountpoint
```

Note

The `btrfs filesystem df` command displays more accurate information about the space used by a btrfs file system than the `df` command does.

Use the following form of the `btrfs` command to display information about all the btrfs file systems on a system:

```
# btrfs filesystem show
```

4.3 Modifying a Btrfs File System

The following table shows how you can use the `btrfs` command to add or remove devices, and to rebalance the layout of the file system data and metadata across the devices.

Command	Description
`btrfs device add` *`device`* *`mountpoint`*	Add a device to the file system that is mounted on the specified mount point. For example: `btrfs device add /dev/sdd /myfs`
`btrfs device delete` *`device`* *`mountpoint`*	Remove a device from a mounted file system. For example:

Command	Description
	`btrfs device delete /dev/sde /myfs`
`btrfs device delete missing` *`mountpoint`*	Remove a failed device from the file system that is mounted in degraded mode. For example: `btrfs device remove missing /myfs` To mount a file system in degraded mode, specify the `-o degraded` option to the `mount` command. For a RAID configuration, if the number of devices would fall below the minimum number that are required, you must add the replacement device before removing the failed device.
`btrfs filesystem balance` *`mountpoint`*	After adding or removing devices, redistribute the file system data and metadata across the available devices.

4.4 Compressing and Defragmenting a Btrfs File System

You can compress a btrfs file system to increase its effective capacity, and you can defragment it to increase I/O performance.

To enable compression of a btrfs file system, specify one of the following `mount` options:

Mount Option	Description
`compress=lzo`	Use LZO compression.
`compress=zlib`	Use zlib compression.

LZO offers a better compression ratio, while zlib offers faster compression.

You can also compress a btrfs file system at the same time that you defragment it.

To defragment a btrfs file system, use the following command:

```
# btrfs filesystem defragment filesystem_name
```

To defragment a btrfs file system and compress it at the same time:

```
# btrfs filesystem defragment -c filesystem_name
```

You can also defragment, and optionally compress, individual file system objects, such as directories and files, within a btrfs file system.

```
# btrfs filesystem defragment [-c] file_name ...
```

Note

You can set up automatic defragmentation by specifying the `autodefrag` option when you mount the file system. However, automatic defragmentation is not recommended for large databases or for images of virtual machines.

Defragmenting a file or a subvolume that has a copy-on-write copy results breaks the link between the file and its copy. For example, if you defragment a subvolume that has a snapshot, the disk usage by the subvolume and its snapshot will increase because the snapshot is no longer a copy-on-write image of the subvolume.

4.5 Resizing a Btrfs File System

You can use the `btrfs` command to increase the size of a mounted btrfs file system if there is space on the underlying devices to accommodate the change, or to decrease its size if the file system has sufficient available free space. The command does not have any effect on the layout or size of the underlying devices.

For example, to increase the size of `/mybtrfs1` by 2 GB:

```
# btrfs filesystem resize +2g /mybtrfs1
```

Decrease the size of `/mybtrfs2` by 4 GB:

```
# btrfs filesystem resize -4g /mybtrfs2
```

Set the size of `/mybtrfs3` to 20 GB:

```
# btrfs filesystem resize 20g /mybtrfs3
```

4.6 Creating Subvolumes and Snapshots

The top level of a btrfs file system is a subvolume consisting of a named b-tree structure that contains directories, files, and possibly further btrfs subvolumes that are themselves named b-trees that contain directories and files, and so on. To create a subvolume, change directory to the position in the btrfs file system where you want to create the subvolume and enter the following command:

```
# btrfs subvolume create subvolume_name
```

Snapshots are a type of subvolume that records the contents of their parent subvolumes at the time that you took the snapshot. If you take a snapshot of a btrfs file system and do not write to it, the snapshot records the state of the original file system and forms a stable image from which you can make a backup. If you make a snapshot writable, you can treat it as a alternate version of the original file system. The copy-on-write functionality of btrfs file system means that snapshots are quick to create, and consume very little disk space initially.

Note

Taking snapshots of a subvolume is not a recursive process. If you create a snapshot of a subvolume, every subvolume or snapshot that the subvolume contains is mapped to an empty directory of the same name inside the snapshot.

The following table shows how to perform some common snapshot operations:

Command	Description
`btrfs subvolume snapshot` *`pathname pathname/snapshot_path`*	Create a snapshot *`snapshot_path`* of a parent subvolume or snapshot specified by *`pathname`*. For example: `btrfs subvolume snapshot /mybtrfs /mybtrfs/snapshot1`
`btrfs subvolume list` *`pathname`*	List the subvolumes or snapshots of a subvolume or snapshot specified by *`pathname`*. For example: `btrfs subvolume list /mybtrfs` **Note** You can use this command to determine the ID of a subvolume or snapshot.
`btrfs subvolume set-default` *`ID pathname`*	By default, mount the snapshot or subvolume specified by its ID instead of the parent subvolume. For example: `btrfs subvolume set-default 4 /mybtrfs`
`btrfs subvolume get-default` *`pathname`*	Displays the ID of the default subvolume that is mounted for the specified subvolume. For example: `btrfs subvolume get-default /mybtrfs`

You can mount a btrfs subvolume as though it were a disk device. If you mount a snapshot instead of its parent subvolume, you effectively roll back the state of the file system to the time that the snapshot was taken. By default, the operating system mounts the parent btrfs volume, which has an ID of 0, unless you use `set-default` to change the default subvolume. If you set a new default subvolume, the system will mount that subvolume instead in future. You can override the default setting by specifying either of the following `mount` options:

Mount Option	Description
`subvolid=`*`snapshot_ID`*	Mount the subvolume or snapshot specified by its subvolume ID instead of the default subvolume.
`subvol=`*`pathname/snapshot_path`*	Mount the subvolume or snapshot specified by its pathname instead of the default subvolume. **Note** The subvolume or snapshot must be located in the root of the btrfs file system.

When you have rolled back a file system by mounting a snapshot, you can take snapshots of the snapshot itself to record its state.

When you no longer require a subvolume or snapshot, use the following command to delete it:

```
# btrfs subvolume delete subvolume_path
```

Note

Deleting a subvolume deletes all subvolumes that are below it in the b-tree hierarchy. For this reason, you cannot remove the topmost subvolume of a btrfs file system, which has an ID of 0.

4.6.1 Cloning Virtual Machine Images and Linux Containers

You can use a btrfs file system to provide storage space for virtual machine images and Linux Containers. The ability to quickly clone files and create snapshots of directory structures makes btrfs an ideal candidate for this purpose. For an example of using the snapshot feature of btrfs to implement Linux Containers, see Section 8.2, “Configuring Operating System Containers”.

4.7 Using the Send/Receive Feature

Note

The send/receive feature requires that you boot the system using UEK R3.

The send operation compares two subvolumes and writes a description of how to convert one subvolume (the *parent* subvolume) into the other (the *sent* subvolume). You would usually direct the output to a file for later use or pipe it to a receive operation for immediate use.

The simplest form of the send operation writes a complete description of a subvolume:

```
# btrfs send [-v] [-f sent_file] ... subvol
```

You can specify multiple instances of the `-v` option to display increasing amounts of debugging output. The `-f` option allows you to save the output to a file. Both of these options are implicit in the following usage examples.

The following form of the send operation writes a complete description of how to convert one subvolume into another:

```
# btrfs send -p parent_subvol sent_subvol
```

If a subvolume such as a snapshot of the parent volume, known as a *clone source*, will be available during the receive operation from which some of the data can be recovered, you can specify the clone source to reduce the size of the output file:

```
# btrfs send [-p parent_subvol] -c clone_src [-c clone_src] ... subvol
```

You can specify the `-c` option multiple times if there is more than one clone source. If you do not specify the parent subvolume, btrfs chooses a suitable parent from the clone sources.

You use the receive operation to regenerate the sent subvolume at a specified path:

```
# btrfs receive [-f sent_file] mountpoint
```

4.7.1 Using Send/Receive to Implement Incremental Backups

The following procedure is a suggestion for setting up an incremental backup and restore process for a subvolume.

1. Create a read-only snapshot of the subvolume to serve as an initial reference point for the backup:

   ```
   # btrfs subvolume snapshot -r /vol /vol/backup_0
   ```

2. Run `sync` to ensure that the snapshot has been written to disk:

   ```
   # sync
   ```

3. Create a subvolume or directory on a btrfs file system as a backup area to receive the snapshot, for example, `/backupvol`.

4. Send the snapshot to `/backupvol`:

   ```
   # btrfs send /vol/backup_0 | btrfs receive /backupvol
   ```

   This command creates the subvolume `/backupvol/backup_0`.

   Having created the reference backup, you can then create incremental backups as required.

5. To create an incremental backup:

   a. Create a new snapshot of the subvolume:

      ```
      # btrfs subvolume snapshot -r /vol /vol/backup_1
      ```

   b. Run `sync` to ensure that the snapshot has been written to disk:

      ```
      # sync
      ```

   c. Send only the differences between the reference backup and the new backup to the backup area:

      ```
      # btrfs send -p /vol/backup_0 /vol/backup_1 | btrfs receive /backupvol
      ```

      This command creates the subvolume `/backupvol/backup_1`.

4.8 Using Quota Groups

> **Note**
>
> The quota groups feature requires that you boot the system using UEK R3.

To enable quotas, use the following command on a newly created btrfs file system before any creating any subvolumes:

```
# btrfs quota enable volume
```

To assign a quota-group limit to a subvolume, use the following command:

```
# btrfs qgroup limit size /volume/subvolume
```

For example:

```
# btrfs qgroup limit 1g /myvol/subvol1
# btrfs qgroup limit 512m /myvol/subvol2
```

To find out the quota usage for a subvolume, use the `btrfs qgroup show` *`path`* command:

4.9 Replacing Devices on a Live File System

> **Note**
>
> The device replacement feature requires that you boot the system using UEK R3.

You can replace devices on a live file system. You do not need to unmount the file system or stop any tasks that are using it. If the system crashes or loses power while the replacement is taking place, the operation resumes when the system next mounts the file system.

Use the following command to replace a device on a mounted btrfs file system:

```
# btrfs replace start source_dev target_dev [-r] mountpoint
```

`source_dev` and `target_dev` specify the device to be replaced (*source device*) and the replacement device (*target device*). `mountpoint` specifies the file system that is using the source device. The target device must be the same size as or larger than the source device. If the source device is no longer available or you specify the `-r` option, the data is reconstructed by using redundant data obtained from other devices (such as another available mirror). The source device is removed from the file system when the operation is complete.

You can use the `btrfs replace status mountpoint` and `btrfs replace cancel mountpoint` commands to check the progress of the replacement operation or to cancel the operation.

4.10 Creating Snapshots of Files

You can use the `--reflink` option to the `cp` command to create lightweight copies of a file within the same subvolume of a btrfs file system. The copy-on-write mechanism saves disk space and allows copy operations to be almost instantaneous. The btrfs file system creates a new inode that shares the same disk blocks as the existing file, rather than creating a complete copy of the file's data or creating a link that points to the file's inode. The resulting file appears to be a copy of the original file, but the original data blocks are not duplicated. If you subsequently write to one of the files, the btrfs file system makes copies of the blocks before they are written to, preserving the other file's content.

For example, the following command creates the snapshot `bar` of the file `foo`:

```
# cp -reflink foo bar
```

4.11 Converting an Ext2, Ext3, or Ext4 File System to a Btrfs File System

You can use the `btrfs-convert` utility to convert an `ext2`, `ext3`, or `ext4` file system to `btrfs`. The utility preserves an image of the original file system in a snapshot named `ext2_saved`. This snapshot allows you to roll back the conversion, even if you have made changes to the btrfs file system.

If you convert the root file system to btrfs, you can use snapshots to roll back changes such as upgrades that you have made to the file system.

Note

You cannot convert a bootable partition, such as `/boot`, to a btrfs file system.

4.11.1 Converting a Non-root File System

Caution

Before performing a file system conversion, make a backup of the file system from which you can restore its state.

To convert an `ext2`, `ext3`, or `ext4` file system other than the root file system to `btrfs`:

1. Unmount the file system.

```
# umount mountpoint
```

2. Run the correct version of `fsck` (for example, `fsck.ext4`) on the underlying device to check and correct the integrity of file system.

```
# fsck.extN -f device
```

3. Convert the file system to a btrfs file system.

```
# btrfs-convert device
```

4. Edit the file `/etc/fstab`, and change the file system type of the file system to `btrfs`, for example:

```
/dev/sdb                /myfs          btrfs    defaults  0 0
```

5. Mount the converted file system on the old mount point.

```
# mount device mountpoint
```

4.11.2 Converting the root File System

Caution

Before performing a root file system conversion, make a full system backup from which you can restore its state.

To convert an `ext2`, `ext3`, or `ext4` root file system to `btrfs`:

1. Run the `mount` command to determine the device that is currently mounted as the root file system, and the type of the file system.

   In the following example, the root file system is configured as an LVM logical volume `lv_root` in the volume group `vg_hostol6`, and the file system type is `ext4`. Using the `ls -l` command confirms that the mapped device corresponds to `/dev/vg_hostol6/lv_root`.

```
# mount
/dev/mapper/vg_hostol6-lv_root on / type ext4 (rw)
.
.
.
# ls -l /dev/mapper/vg_hostol6-lv_root
lrwxrwxrwx. 1 root root 7 Sep 14 14:00 /dev/mapper/vg_hostol6-lv_root -> ../dm-0
# ls -l /dev/vg_hostol6/lv_root
lrwxrwxrwx. 1 root root 7 Sep 14 14:00 /dev/vg_hostol6/lv_root -> ../dm-0
```

   In the next example, the root file system corresponds to the disk partition `/dev/sda2`:

```
# mount
...
/dev/sda2 on / type ext4 (rw)
...
```

2. Shut down the system.

3. Boot the system from an Oracle Linux 6 Update 3 or later UEK Boot ISO (which you can burn to CD or DVD if necessary). You can download the UEK Boot ISO from https://edelivery.oracle.com/linux.

   Note

   You must use the UEK Boot ISO. You cannot use the RHCK Boot ISO to perform the conversion.

4. From the installation menu, select **Rescue Installed System**. When prompted, choose a language and keyboard, select **Local CD/DVD** as the installation media, select **No** to bypass starting the network interface, and select **Skip** to bypass selecting a rescue environment.

5. Select **Start shell** to obtain a `bash` shell prompt (`bash-4.1#`) at the bottom of the screen.

6. If the existing root file system is configured as an LVM volume, use the following command to start the volume group (for example, `vg_hostol6`):

```
bash-4.1# lvchange -ay vg_hostol6
```

7. Run the correct version of `fsck` (for example, `fsck.ext3` or `fsck.ext4`) to check and correct the integrity of the file system.

```
bash-4.1# fsck.extN -f device
```

   where *`device`* is the root file system device (for example, `/dev/vg_hostol6/lv_root` or `/dev/sda2`).

8. Convert the file system to a btrfs file system.

```
bash-4.1# btrfs-convert device
```

9. Create a mount point (`/mnt1`) and mount the converted root file system on it.

```
bash-4.1# mkdir /mnt1
bash-4.1# mount -t btrfs device /mnt1
```

10. Use the `vi` command to edit the file `/mnt1/etc/fstab`, and change the file system type of the root file system to `btrfs`, for example:

```
/dev/mapper/vg_hostol6-lv_root  /       btrfs    defaults  1 1
```

11. Create the file `.autorelabel` in the root of the mounted file system.

```
bash-4.1# touch /mnt1/.autorelabel
```

   The presence of the `.autorelabel` file in `/` instructs SELinux to recreate the security attributes of all files on the file system.

   Note

   If you do not create the `.autorelabel` file, you might not be able to boot the system successfully. If you forget to create the file and the reboot fails, either disable SELinux temporarily by specifying `selinux=0` to the kernel boot parameters, or run SELinux in permissive mode by specifying `enforcing=0`.

12. Unmount the converted root file system.

```
bash-4.1# umount /mnt1
```

13. Remove the boot CD, DVD, or ISO, and reboot the system.

4.11.3 Mounting the Image of the Original File System

To mount the image of the original file system read-only:

1. Mount the snapshot of the original file system on a temporary mount point.

```
# mount -t btrfs -o subvol=ext2_saved device temp_mountpoint1
```

2. Mount the image of the original file system read-only on another temporary mount point, specifying the correct file system type (`ext2`, `ext3`, or `ext4`) to the `-t` option.

```
# mount -t extN -o loop,ro temp_mountpoint1/image temp_mountpoint2
```

4.11.4 Deleting the Snapshot of the Original File System

Caution

If you delete the snapshot of the original file system to save storage space, you will no longer be able to recover the original file system.

To delete the snapshot of the original file system and recover the space that it uses:

1. Delete the `ext2_saved` subvolume.

```
# btrfs subvolume delete mountpoint/ext2_saved
```

For example, if you converted the root file system (`/`) file system, you would enter:

```
# btrfs subvolume delete //ext2_saved
```

For another file system, such as `/usr`, you would enter:

```
# btrfs subvolume delete /usr/ext2_saved
```

2. Rebalance the btrfs file system.

```
# btrfs filesystem balance device
```

4.11.5 Recovering an Original Non-root File System

Caution

If you roll back a conversion, you will lose any changes that you have made to the btrfs file system. Make a back up of the changes that you want to reapply to the restored file system.

To roll back the conversion of the file system and recover the original file system:

1. Unmount the btrfs file system and all of its snapshots and images in the reverse order from which you originally mounted them.

```
# umount temp_mountpoint2
# umount temp_mountpoint1/image
# umount mountpoint
```

2. Roll back the conversion.

```
# btrfs-convert -r device
```

3. Mount the original file system.

```
# mount -t extN device mountpoint
```

4.12 Installing a Btrfs root File System

For compatibility reasons, the default installation image of Oracle Linux boots the Red Hat compatible kernel to perform the installation. Oracle provides an alternative installation image (UEK Boot ISO) that supports the installation of Oracle Linux 6 Update 3 or later using the Unbreakable Enterprise Kernel (UEK) as the installation kernel. This installation method allows you to create a btrfs root file system.

As the UEK Boot ISO contains only the bootable installation image, you must set up a network installation server for the RPM packages. This server must have sufficient storage space to host the full Oracle Linux Release 6 Update 3 or later Media Pack DVD image (approximately 3.5 GB), and you must configure it to serve the image files using either NFS or HTTP to the target system on which you want to install Oracle Linux 6 Update 3 or later.

- Section 4.12.1, “Setting up a New NFS Server”
- Section 4.12.2, “Configuring an Existing NFS Server”
- Section 4.12.3, “Setting up a New HTTP Server”
- Section 4.12.4, “Configuring an Existing HTTP Server”
- Section 4.12.5, “Setting up a Network Installation Server”
- Section 4.12.6, “Installing from a Network Installation Server”

4.12.1 Setting up a New NFS Server

> **Note**
>
> This procedure assumes that you are setting up an Oracle Linux 6 system as an NFSv4 server. Using NFSv4 greatly simplifies firewall configuration as you need only configure a single rule for TCP port 2049.

To set up an NFS server:

1. Install the `nfs-utils` package.

   ```
   # yum install nfs-utils
   ```

2. Create the directory where you will copy the full Oracle Linux Release 6 Media Pack DVD image, for example `/var/OSimage/OL6.3`:

   ```
   # mkdir -p /var/OSimage/OL6.3
   ```

3. Edit the configuration file, `/etc/exports`, as follows.

   a. Add an entry for the directory where you will copy the DVD image.

      The following example allows read-only access to the directory `/var/OSimage/OL6.3` for any NFS client on the 192.168.1 subnet:

      ```
      /var/OSimage/OL6.3 192.168.1.0/24(ro)
      ```

   b. Save your changes to the file.

4. Start the NFS server, and configure it to start after a reboot.

   ```
   # service rpcbind start
   # service nfs start
   # service nfslock start
   # chkconfig rpcbind on
   # chkconfig nfs on
   # chkconfig nfslock on
   ```

5. If you have configured a firewall on your system, configure it to allow incoming NFSv4 requests from NFS clients.

   For example, use the following commands to configure `iptables` to allow NFSv4 connections and save the change to the firewall configuration:

```
# iptables -I INPUT -p tcp -m state --state NEW -m tcp --dport 2049 -j ACCEPT
# service iptables save
```

4.12.2 Configuring an Existing NFS Server

To configure an existing NFS server:

1. Create the directory where you will copy the full Oracle Linux Release 6 Media Pack DVD image, for example `/var/OSimage/OL6.3`:

```
# mkdir -p /var/OSimage/OL6.3
```

2. Use the `exportfs` command to export the directory.

```
# exportfs -i -o options client:export_dir
```

   For example, to allow read-only access to the directory `/var/OSimage/OL6.3` for any NFS client on the 192.168.1 subnet:

```
# exportfs -i -o ro 192.168.1.0/24:/var/OSimage/OL6.3
```

4.12.3 Setting up a New HTTP Server

Note

These instructions assume that you are setting up an Oracle Linux 6 system as an Apache HTTP server.

To set up an HTTP server:

1. Install the Apache HTTP server package.

```
# yum install httpd
```

2. Create the directory where you will copy the full Oracle Linux Release 6 Media Pack DVD image, for example `/var/www/html/OSimage/OL6.3`:

```
# mkdir -p /var/www/html/OSimage/OL6.3
```

   Note

   If SELinux is enabled in enforcing mode on your system, create the directory under the `/var/www/html` directory hierarchy so that the `httpd_sys_content_t` file type is set automatically on all the files in the repository.

3. Edit the HTTP server configuration file, `/etc/httpd/conf/httpd.conf`, as follows:

   a. Specify the resolvable domain name of the server in the argument to `ServerName`.

```
ServerName server_addr:80
```

   If the server does not have a resolvable domain name, enter its IP address instead. For example, the following entry would be appropriate for an HTTP server with the IP address 192.168.1.100.

```
ServerName 192.168.1.100:80
```

b. If the directory to which you will copy the DVD image in not under `/var/www/html`, change the default setting of `DocumentRoot`.

In this example, the DVD image will be copied to `/var/www/html/OSimage/OL6.3` so the setting of `DocumentRoot` can remain unchanged.

```
DocumentRoot "/var/www/html"
```

c. Verify that the `<Directory>` setting points to the same setting as `DocumentRoot`.

```
#
# This should be changed to whatever you set DocumentRoot to.
#
<Directory "/var/www/html">
```

d. If you want to be able to browse the directory hierarchy, verify that the `Options` directive specifies the `Indexes` option, for example:

```
Options Indexes FollowSymLinks
```

Note

The `Indexes` option is not required for installation.

e. Save your changes to the file.

4. Start the Apache HTTP server, and configure it to start after a reboot.

```
# service httpd start
# chkconfig httpd on
```

5. If you have enabled a firewall on your system, configure it to allow incoming HTTP connection requests on TCP port 80.

For example, the following command configures `iptables` to allow incoming HTTP connection requests and saves the change to the firewall configuration:

```
# iptables -I INPUT -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT
# service iptables save
```

4.12.4 Configuring an Existing HTTP Server

To configure an existing Apache HTTP server:

1. Under the `DocumentRoot` hierarchy that is defined in the HTTP server configuration file (`/etc/httpd/conf/httpd.conf`), create the directory where you will copy the full Oracle Linux Release 6 Media Pack DVD image, for example `/var/www/html/OSimage/OL6.3`:

```
# mkdir -p /var/www/html/OSimage/OL6.3
```

2. Edit the HTTP server configuration file, `/etc/httpd/conf/httpd.conf`, and add a `<Directory>` section, for example:

```
<Directory "/var/www/html/OSimage/OL6.3">
    Options Indexes FollowSymLinks
    AllowOverride None
    Order allow,deny
```

```
    Allow from all
</Directory>
```

Place this section after the closing `</Directory>` statement for the `<Directory DocumentRoot>` section.

Note

The `Indexes` option is not required for installation. Specify this option if you want to be able to browse the directory hierarchy.

4.12.5 Setting up a Network Installation Server

Note

This procedure assumes that you have set up the system as an NFS or HTTP server.

To set up a network installation server:

1. Download the full Oracle Linux Media Pack DVD image (for example, `V41362-01.iso` for x86_64 (64 bit) Oracle Linux Release 6 Update 5) from the Oracle Software Delivery Cloud at https://edelivery.oracle.com/linux.

2. Mount the DVD image on a suitable mount point (for example, `/mnt`):

   ```
   # mount -t iso9660 -o loop V41362-01.iso mount_dir
   ```

3. Use the following command to extract the contents of the DVD image into a directory (*`output_dir`*) whose contents are shareable using NFS or HTTP:

   ```
   # cp -a -T mount_dir output_dir
   ```

   For example, to copy the DVD image mounted on `/mnt` to `/var/OSimage/OL6.5`:

   ```
   # cp -a -T /mnt /var/OSimage/OL6.5
   ```

   or to `/var/www/html/OSimage/OL6.5`:

   ```
   # cp -a -T /mnt /var/www/html/OSimage/OL6.5
   ```

4. Unmount the DVD image:

   ```
   # umount mount_dir
   ```

5. Download the UEK Boot ISO image for the desired architecture (for example, `V41364-01.iso` for x86_64 (64 bit)).

6. Mount the UEK Boot ISO image:

   ```
   # mount -t iso9660 -o loop V41364-01.iso
   ```

7. Replace the contents of the `images` directory that you copied from the DVD image with the contents of the `images` directory from the UEK Boot ISO image:

   ```
   # rm -rf output_dir/images
   # cp -r mount_dir/images output_dir
   ```

   For example, to replace `/var/OSimage/OL6.5/images`:

   ```
   # rm -rf /var/OSimage/OL6.5/images
   ```

```
# cp -r /mnt/images /var/OSimage/OL6.5
```

or to replace `/var/www/html/OSimage/OL6.5/images`:

```
# rm -rf /var/www/html/OSimage/OL6.5/images
# cp -r /mnt/images /var/www/html/OSimage/OL6.5
```

8. If SELinux is enabled in enforcing mode on your system and you have configured the system as an HTTP server but you did not copy the DVD image to a directory under `/var/www/html`:

   a. Use the `semanage` command to define the default file type of the directory hierarchy as `httpd_sys_content_t`:

      ```
      # /usr/sbin/semanage fcontext -a -t httpd_sys_content_t "/var/OSimage(/.*)?"
      ```

   b. Use the `restorecon` command to apply the file type to the entire directory hierarchy.

      ```
      # /sbin/restorecon -R -v /var/OSimage
      ```

   Note

   The `semanage` and `restorecon` commands are provided by the `policycoreutils-python` and `policycoreutils` packages.

9. Copy the UEK Boot ISO image to a suitable medium from which you can boot the target system on which you want to install Oracle Linux 6 Update 5.

10. Unmount the UEK Boot ISO image:

    ```
    # umount mount_dir
    ```

4.12.6 Installing from a Network Installation Server

To install a target system from a network installation server:

1. Boot the target system using the UEK Boot ISO.

2. Select **Install or upgrade an existing system**, press `Tab`, and enter `askmethod` as an additional parameter on the boot command line:

   ```
   > vmlinuz initrd=initrd.img askmethod
   ```

3. On the **Installation Method** screen, select either **NFS directory** or **URL** depending on whether you configured your installation server to use NFS or HTTP respectively.

4. After configuring the network settings, enter the settings for the NFS or HTTP installation server.

   For installation using NFS, enter the path of the full DVD image, for example `/var/OSimage/OL6.5`.

   For installation using HTTP, enter the URL of the full DVD image, for example `http://192.168.1.100/OSimage/OL6.5`.

5. The default disk layout creates a btrfs root file system.

   Note

   You cannot configure a bootable partition, such as `/boot`, as a btrfs file system.

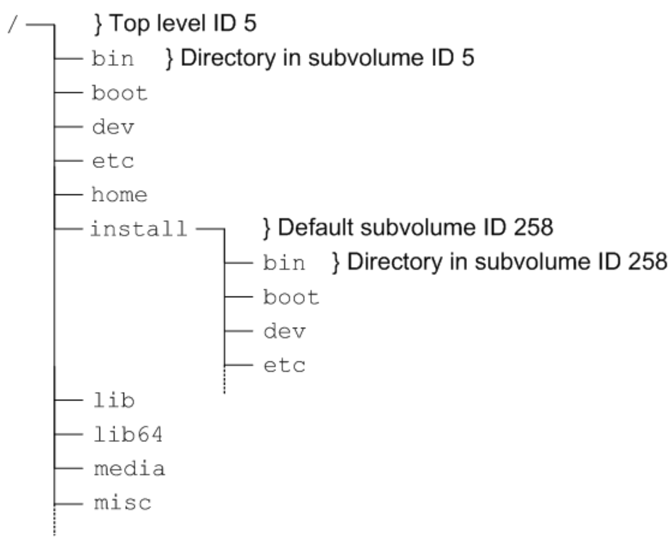
4.12.7 About the Installation root File System

The mounted root file system is a snapshot (named `install`) of the root file system taken at the end of installation. To find out the ID of the parent of the root file system subvolume, use the following command:

```
# btrfs subvolume list /
ID 258 top level 5 path install
```

In this example, the installation root file system subvolume has an ID of 5. The subvolume with ID 258 (`install`) is currently mounted as `/`. Figure 4.1, “Layout of the root File System Following Installation” illustrates the layout of the file system:

Figure 4.1 Layout of the root File System Following Installation



The top-level subvolume with ID 5 records the contents of the root file system file system at the end of installation. The default subvolume (`install`) with ID 258 is currently mounted as the active root file system.

The `mount` command shows the device that is currently mounted as the root file system:

```
# mount
/dev/mapper/vg_btrfs-lv_root on / type btrfs (rw)
...
```

To mount the installation root file system volume, you can use the following commands:

```
# mkdir /instroot
# mount -o subvolid=5 /dev/mapper/vg_btrfs-lv_root /instroot
```

If you list the contents of `/instroot`, you can see both the contents of the installation root file system volume and the `install` snapshot, for example:

```
# ls /instroot
bin    cgroup  etc    install  lib64  misc  net  proc  sbin     srv  tmp  var
boot   dev     home   lib      media  mnt   opt  root  selinux  sys  usr
```

The contents of `/` and `/instroot/install` are identical as demonstrated in the following example where a file (`foo`) created in `/instroot/install` is also visible in `/`:

```
# touch /instroot/install/foo
# ls /
```

```
bin   cgroup  etc  home      lib    media  mnt  opt   root  selinux  sys  usr
boot  dev     foo  instroot  lib64  misc   net  proc  sbin  srv      tmp  var
# ls /instroot/install
bin   cgroup  etc  home      lib    media  mnt  opt   root  selinux  sys  usr
boot  dev     foo  instroot  lib64  misc   net  proc  sbin  srv      tmp  var
# rm -f /foo
# ls /
bin   cgroup  etc   instroot  lib64  misc  net  proc  sbin     srv  tmp  var
boot  dev     home  lib       media  mnt   opt  root  selinux  sys  usr
# ls /instroot/install
bin   cgroup  etc   instroot  lib64  misc  net  proc  sbin     srv  tmp  var
boot  dev     home  lib       media  mnt   opt  root  selinux  sys  usr
```

4.12.8 Creating Snapshots of the root File System

To take a snapshot of the current root file system:

1. Mount the top level of the root file system on a suitable mount point.

   ```
   # mount -o subvolid=5 /dev/mapper/vg_btrfs-lv_root /mnt
   ```

2. Change directory to the mount point and take the snapshot. In this example, the `install` subvolume is currently mounted as the root file system system.

   ```
   # cd /mnt
   # btrfs subvolume snapshot install root_snapshot_1
   Create a snapshot of 'install' in './root_snapshot_1'
   ```

3. Change directory to `/` and unmount the top level of the file system.

   ```
   # cd /
   # umount /mnt
   ```

   The list of subvolumes now includes the newly created snapshot.

   ```
   # btrfs subvolume list /
   ID 258 top level 5 path install
   ID 260 top level 5 path root_snapshot_1
   ```

4.12.9 Mounting Alternate Snapshots as the root File System

If you want to roll back changes to your system, you can mount a snapshot as the root file system by specifying its ID as the default subvolume, for example:

```
# btrfs subvolume set-default 260 /
```

Reboot the system for the change to take effect.

4.12.10 Deleting Snapshots of the root File System

To delete a snapshot:

1. Mount the top level of the file system, for example:

   ```
   # mount -o subvolid=5 /dev/mapper/vg_btrfs-lv_root /mnt
   ```

2. Change directory to the mount point and delete the snapshot.

   ```
   # cd /mnt
   # btrfs subvolume delete install
   Delete subvolume '/mnt/install'
   ```

3. Change directory to `/` and unmount the top level of the file system.

```
# cd /
# umount /mnt
```

The list of subvolumes now does not include `install`.

```
# btrfs subvolume list /
ID 260 top level 5 path root_snapshot_1
```

4.13 For More Information About Btrfs

You can find more information about the btrfs file system at https://btrfs.wiki.kernel.org/index.php/Main_Page.

Chapter 5 The XFS File System

Table of Contents

This chapter describes how to configure and use the XFS file system.

5.1 About the XFS File System

Note

You must have an Oracle Linux Premier Support account to obtain technical support for XFS with Oracle Linux.

The XFS file system is supported for the Unbreakable Enterprise Kernel Release 2 (2.6.39) and the Unbreakable Enterprise Kernel Release 3 (3.8.13) on the x86_64 architecture only.

XFS is a high-performance journaling file system that was initially created by Silicon Graphics, Inc. for the IRIX operating system and later ported to Linux. The parallel I/O performance of XFS provides high scalability for I/O threads, file system bandwidth, file and file system size, even when the file system spans many storage devices.

A typical use case for XFS is to implement a several-hundred terabyte file system across multiple storage servers, each server consisting of multiple FC-connected disk arrays.

XFS is not supported for use with the root (`/`) or `boot` file systems on Oracle Linux.

XFS has a large number of features that make it suitable for deployment in an enterprise-level computing environment that requires the implementation of very large file systems:

- On x86_64 systems, XFS supports a maximum file system size and maximum file size of nearly 8 EB. The maximum supported limit for XFS on Oracle Linux is 100 TB.

- XFS implements journaling for metadata operations, which guarantees the consistency of the file system following loss of power or a system crash. XFS records file system updates asynchronously to a circular buffer (the *journal*) before it can commit the actual data updates to disk. The journal can be located either internally in the data section of the file system, or externally on a separate device to reduce contention for disk access. If the system crashes or loses power, it reads the journal when the file system is remounted, and replays any pending metadata operations to ensure the consistency of the file system. The speed of this recovery does not depend on the size of the file system.

- XFS is internally partitioned into allocation groups, which are virtual storage regions of fixed size. Any files and directories that you create can span multiple allocation groups. Each allocation group manages its own set of inodes and free space independently of other allocation groups to provide both scalability and parallelism of I/O operations. If the file system spans many physical devices, allocation groups can optimize throughput by taking advantage of the underlying separation of channels to the storage components.

- XFS is an extent-based file system. To reduce file fragmentation and file scattering, each file's blocks can have variable length extents, where each extent consists of one or more contiguous blocks. XFS's space allocation scheme is designed to efficiently locate free extents that it can use for file system operations. XFS does not allocate storage to the holes in sparse files. If possible, the extent allocation map for a file is stored in its inode. Large allocation maps are stored in a data structure maintained by the allocation group.

- To maximize throughput for XFS file systems that you create on an underlying striped, software or hardware-based array, you can use the `su` and `sw` arguments to the `-d` option of the `mkfs.xfs` command to specify the size of each stripe unit and the number of units per stripe. XFS uses the information to align data, inodes, and journal appropriately for the storage. On `lvm` and `md` volumes and some hardware RAID configurations, XFS can automatically select the optimal stripe parameters for you.

- To reduce fragmentation and increase performance, XFS implements *delayed allocation*, reserving file system blocks for data in the buffer cache, and allocating the block when the operating system flushes that data to disk.

- XFS supports extended attributes for files, where the size of each attribute's value can be up to 64 KB, and each attribute can be allocated to either a `root` or a `user` name space.

- Direct I/O in XFS implements high throughput, non-cached I/O by performing DMA directly between an application and a storage device, utilising the full I/O bandwidth of the device.

- To support the snapshot facilities that volume managers, hardware subsystems, and databases provide, you can use the `xfs_freeze` command to suspend and resume I/O for an XFS file system. See Section 5.6, “Freezing and Unfreezing an XFS File System”.

- To defragment individual files in an active XFS file system, you can use the `xfs_fsr` command. See Section 5.9, “Defragmenting an XFS File System”.

- To grow an XFS file system, you can use the `xfs_growfs` command. See Section 5.5, “Growing an XFS File System”.

- To back up and restore a live XFS file system, you can use the `xfsdump` and `xfsrestore` commands. See Section 5.8, “Backing up and Restoring XFS File Systems”.

- XFS supports user, group, and project disk quotas on block and inode usage that are initialized when the file system is mounted. Project disk quotas allow you to set limits for individual directory hierarchies within an XFS file system without regard to which user or group has write access to that directory hierarchy.

5.1.1 About External XFS Journals

The default location for an XFS journal is on the same block device as the data. As synchronous metadata writes to the journal must complete successfully before any associated data writes can start, such a layout can lead to disk contention for the typical workload pattern on a database server. To overcome this problem, you can place the journal on a separate physical device with a low-latency I/O path. As the journal typically requires very little storage space, such an arrangement can significantly improve the file system's I/O throughput. A suitable host device for the journal is a solid-state drive (SSD) device or a RAID device with a battery-backed write-back cache.

To reserve an external journal with a specified size when you create an XFS file system, specify the `-l logdev=device,size=size` option to the `mkfs.xfs` command. If you omit the `size` parameter, `mkfs.xfs` selects a journal size based on the size of the file system. To mount the XFS file system so that it uses the external journal, specify the `-o logdev=device` option to the `mount` command.

5.1.2 About XFS Write Barriers

A write barrier assures file system consistency on storage hardware that supports flushing of in-memory data to the underlying device. This ability is particularly important for write operations to an XFS journal that is held on a device with a volatile write-back cache.

By default, an XFS file system is mounted with a write barrier. If you create an XFS file system on a LUN that has a battery-backed, non-volatile cache, using a write barrier degrades I/O performance by requiring data to be flushed more often than necessary. In such cases, you can remove the write barrier by mounting the file system with the `-o nobarrier` option to the `mount` command.

5.1.3 About Lazy Counters

With lazy-counters enabled on an XFS file system, the free-space and inode counters are maintained in parts of the file system other than the superblock. This arrangement can significantly improve I/O performance for application workloads that are metadata intensive.

Lazy counters are enabled by default, but if required, you can disable them by specifying the `-l lazy-count=0` option to the `mkfs.xfs` command.

5.2 Installing the XFS Packages

Note

You can also obtain the XFS packages from the Oracle Linux Yum Server.

To install the XFS packages on a system:

1. Log in to ULN, and subscribe your system to the `ol6_x86_64_latest` channel.

2. On your system, use `yum` to install the `xfsprogs` and `xfsdump` packages:

```
# yum install xfsprogs xfsdump
```

3. If required, use `yum` to install the XFS development and QA packages:

```
# yum install xfsprogs-devel xfsprogs-qa-devel
```

5.3 Creating an XFS File System

You can use the `mkfs.xfs` command to create an XFS file system, for example.

```
# mkfs.xfs /dev/vg0/lv0
meta-data=/dev/vg0/lv0           isize=256    agcount=32, agsize=8473312 blks
```

```
         =                       sectsz=512   attr=2, projid32bit=0
data     =                       bsize=4096   blocks=271145984, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0
log      =internal log           bsize=4096   blocks=32768, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

To create an XFS file system with a stripe-unit size of 32 KB and 6 units per stripe, you would specify the `su` and `sw` arguments to the `-d` option, for example:

```
# mkfs.xfs -d su=32k,sw=6 /dev/vg0/lv1
```

For more information, see the `mkfs.xfs(8)` manual page.

5.4 Modifying an XFS File System

Note

You cannot modify a mounted XFS file system.

You can use the `xfs_admin` command to modify an unmounted XFS file system. For example, you can enable or disable lazy counters, change the file system UUID, or change the file system label.

To display the existing label for an unmounted XFS file system and then apply a new label:

```
# xfs_admin -l /dev/sdb
label = ""
# xfs_admin -L "VideoRecords" /dev/sdb
writing all SBs
new label = "VideoRecords"
```

Note

The label can be a maximum of 12 characters in length.

To display the existing UUID and then generate a new UUID:

```
# xfs_admin -u /dev/sdb
UUID = cd4f1cc4-15d8-45f7-afa4-2ae87d1db2ed
# xfs_admin -U generate /dev/sdb
writing all SBs
new UUID = c1b9d5a2-f162-11cf-9ece-0020afc76f16
```

To clear the UUID altogether:

```
# xfs_admin -U nil /dev/sdb
Clearing log and setting UUID
writing all SBs
new UUID = 00000000-0000-0000-0000-000000000000
```

To disable and then re-enable lazy counters:

```
# xfs_admin -c 0 /dev/sdb
Disabling lazy-counters
# xfs_admin -c 1 /dev/sdb
Enabling lazy-counters
```

For more information, see the `mkfs_admin(8)` manual page.

5.5 Growing an XFS File System

Note

You cannot grow an XFS file system that is currently unmounted.

There is currently no command to shrink an XFS file system.

You can use the `xfs_growfs` command to increase the size of a mounted XFS file system if there is space on the underlying devices to accommodate the change. The command does not have any effect on the layout or size of the underlying devices. If necessary, use the underlying volume manager to increase the physical storage that is available. For example, you can use the `vgextend` command to increase the storage that is available to an LVM volume group and `lvextend` to increase the size of the logical volume that contains the file system.

You cannot use the `parted` command to resize a partition that contains an XFS file system. You must instead recreate the partition with a larger size and restore its contents from a backup if you deleted the original partition or from the contents of the original partition if you did not delete it to free up disk space.

For example, to increase the size of `/myxfs1` to 4 TB, assuming a block size of 4 KB:

```
# xfs_growfs -D 1073741824 /myxfs1
```

To increase the size of the file system to the maximum size that the underlying device supports, specify the `-d` option:

```
# xfs_growfs -d /myxfs1
```

For more information, see the `xfs_growfs(8)` manual page.

5.6 Freezing and Unfreezing an XFS File System

If you need to take a hardware-based snapshot of an XFS file system, you can temporarily stop write operations to it.

Note

You do not need to explicitly suspend write operations if you use the `lvcreate` command to take an LVM snapshot.

To freeze and unfreeze an XFS file system, use the `-f` and `-u` options with the `xfs_freeze` command, for example:

```
# xfs_freeze -f /myxfs
# # ... Take snapshot of file system ...
# xfs_freeze -u /myxfs
```

Note

You can also use the `xfs_freeze` command with `btrfs`, `ext3`, and `ext4` file systems.

For more information, see the `xfs_freeze(8)` manual page.

5.7 Setting Quotas on an XFS File System

The following table shows the `mount` options that you can specify to enable quotas on an XFS file system:

Mount Option	Description
gqnoenforce	Enable group quotas. Report usage, but do not enforce usage limits.
gquota	Enable group quotas and enforce usage limits.
pqnoenforce	Enable project quotas. Report usage, but do not enforce usage limits.
pquota	Enable project quotas and enforce usage limits.
uqnoenforce	Enable user quotas. Report usage, but do not enforce usage limits.
uquota	Enable user quotas and enforce usage limits.

To show the block usage limits and the current usage in the myxfs file system for all users, use the xfs_quota command:

```
# xfs_quota -x -c 'report -h' /myxfs
User quota on /myxfs (/dev/vg0/lv0)
                        Blocks
User ID      Used   Soft   Hard Warn/Grace
---------- ---------------------------------
root            0      0      0  00 [------]
guest           0   200M   250M  00 [------]
```

The following forms of the command display the free and used counts for blocks and inodes respectively in the manner of the df -h command:

```
# xfs_quota -c 'df -h' /myxfs
Filesystem     Size   Used  Avail Use% Pathname
/dev/vg0/lv0 200.0G  32.2M  20.0G   1% /myxfs

# xfs_quota -c 'df -ih' /myxfs
Filesystem   Inodes   Used   Free Use% Pathname
/dev/vg0/lv0  21.0m      4  21.0m   1% /myxfs
```

If you specify the -x option to enter expert mode, you can use subcommands such as limit to set soft and hard limits for block and inode usage by an individual user, for example:

```
# xfs_quota -x -c 'limit bsoft=200m bhard=250m isoft=200 ihard=250 guest' /myxfs
```

Of course, this command requires that you mounted the file system with user quotas enabled.

To set limits for a group on an XFS file system that you have mounted with group quotas enabled, specify the -g option to limit, for example:

```
# xfs_quota -x -c 'limit -g bsoft=5g bhard=6g devgrp' /myxfs
```

For more information, see the xfs_quota(8) manual page.

5.7.1 Setting Project Quotas

User and group quotas are supported by other file systems, such as ext4. The XFS file system additionally allows you to set quotas on individual directory hierarchies in the file system that are known as *managed trees*. Each managed tree is uniquely identified by a *project ID* and an optional *project name*. Being able to control the disk usage of a directory hierarchy is useful if you do not otherwise want to set quota limits for a privileged user (for example, /var/log) or if many users or groups have write access to a directory (for example, /var/tmp).

To define a project and set quota limits on it:

1. Mount the XFS file system with project quotas enabled:

```
# mount -o pquota device mountpoint
```

For example, to enable project quotas for the `/myxfs` file system:

```
# mount -o pquota /dev/vg0/lv0 /myxfs
```

2. Define a unique project ID for the directory hierarchy in the `/etc/projects` file:

```
# echo project_ID:mountpoint/directory >> /etc/projects
```

For example, to set a project ID of 51 for the directory hierarchy `/myxfs/testdir`:

```
# echo 51:/myxfs/testdir >> /etc/projects
```

3. Create an entry in the `/etc/projid` file that maps a project name to the project ID:

```
# echo project_name:project_ID >> /etc/projid
```

For example, to map the project name `testproj` to the project with ID 51:

```
# echo testproj:51 >> /etc/projid
```

4. Use the `project` subcommand of `xfs_quota` to define a managed tree in the XFS file system for the project:

```
# xfs_quota -x -c 'project -s project_name' mountpoint
```

For example, to define a managed tree in the `/myxfs` file system for the project `testproj`, which corresponds to the directory hierarchy `/myxfs/testdir`:

```
# xfs_quota -x -c 'project -s testproj' /myxfs
```

5. Use the `limit` subcommand to set limits on the disk usage of the project:

```
# xfs_quota -x -c 'limit -p arguments project_name' mountpoint
```

For example, to set a hard limit of 10 GB of disk space for the project `testproj`:

```
# xfs_quota -x -c 'limit -p bhard=10g testproj' /myxfs
```

For more information, see the `projects(5)`, `projid(5)`, and `xfs_quota(8)` manual pages.

5.8 Backing up and Restoring XFS File Systems

The `xfsdump` package contains the `xfsdump` and `xfsrestore` utilities. `xfsdump` examines the files in an XFS file system, determines which files need to be backed up, and copies them to the storage medium. Any backups that you create using `xfsdump` are portable between systems with different endian architectures. `xfsrestore` restores a full or incremental backup of an XFS file system. You can also restore individual files and directory hierarchies from backups.

> **Note**
>
> Unlike an LVM snapshot, which immediately creates a sparse clone of a volume, `xfsdump` takes time to make a copy of the file system data.

You can use the `xfsdump` command to create a backup of an XFS file system on a device such as a tape drive, or in a backup file on a different file system. A backup can span multiple physical media that are written on the same device, and you can write multiple backups to the same medium. You can write only a single backup to a file. The command does not overwrite existing XFS backups that it finds on physical media. You must use the appropriate command to erase a physical medium if you need to overwrite any existing backups.

For example, the following command writes a level 0 (base) backup of the XFS file system, `/myxfs` to the device `/dev/st0` and assigns a session label to the backup:

```
# xfsdump -l 0 -L "Backup level 0 of /myxfs `date`" -f /dev/st0 /myxfs
```

You can make incremental dumps relative to an existing backup by using the command:

```
# xfsdump -l level -L "Backup level level of /myxfs `date`" -f /dev/st0 /myxfs
```

A level 1 backup records only file system changes since the level 0 backup, a level 2 backup records only the changes since the latest level 1 backup, and so on up to level 9.

If you interrupt a backup by typing `Ctrl-C` and you did not specify the `-J` option (suppress the dump inventory) to `xfsdump` , you can resume the dump at a later date by specifying the `-R` option:

```
# xfsdump -R -l 1 -L "Backup level 1 of /myxfs `date`" -f /dev/st0 /myxfs
```

In this example, the backup session label from the earlier, interrupted session is overridden.

You use the `xfsrestore` command to find out information about the backups you have made of an XFS file system or to restore data from a backup.

The `xfsrestore -I` command displays information about the available backups, including the session ID and session label. If you want to restore a specific backup session from a backup medium, you can specify either the session ID or the session label.

For example, to restore an XFS file system from a level 0 backup by specifying the session ID:

```
# xfsrestore -f /dev/st0 -S c76b3156-c37c-5b6e-7564-a0963ff8ca8f /myxfs
```

If you specify the `-r` option, you can cumulatively recover all data from a level 0 backup and the higher-level backups that are based on that backup:

```
# xfsrestore -r -f /dev/st0 -v silent /myxfs
```

The command searches the archive looking for backups based on the level 0 backup, and prompts you to choose whether you want to restore each backup in turn. After restoring the backup that you select, the command exits. You must run this command multiple times, first selecting to restore the level 0 backup, and then subsequent higher-level backups up to and including the most recent one that you require to restore the file system data.

Note

After completing a cumulative restoration of an XFS file system, you should delete the `housekeeping` directory that `xfsrestore` creates in the destination directory.

You can recover a selected file or subdirectory contents from the backup medium, as shown in the following example, which recovers the contents of `/myxfs/profile/examples` to `/tmp/profile/examples` from the backup with a specified session label:

```
# xfsrestore -f /dev/sr0 -L "Backup level 0 of /myxfs Sat Mar 2 14:47:59 GMT 2013" \
  -s profile/examples /usr/tmp
```

Alternatively, you can interactively browse a backup by specifying the `-i` option:

```
# xfsrestore -f /dev/sr0 -i
```

This form of the command allows you browse a backup as though it were a file system. You can change directories, list files, add files, delete files, or extract files from a backup.

To copy the entire contents of one XFS file system to another, you can combine `xfsdump` and `xfsrestore`, using the `-J` option to suppress the usual dump inventory housekeeping that the commands perform:

```
# xfsdump -J - /myxfs | xfsrestore -J - /myxfsclone
```

For more information, see the `xfsdump(8)` and `xfsrestore(8)` manual pages.

5.9 Defragmenting an XFS File System

You can use the `xfs_fsr` command to defragment whole XFS file systems or individual files within an XFS file system. As XFS is an extent-based file system, it is usually unnecessary to defragment a whole file system, and doing so is not recommended.

To defragment an individual file, specify the name of the file as the argument to `xfs_fsr`.

```
# xfs_fsr pathname
```

If you run the `xfs_fsr` command without any options, the command defragments all currently mounted, writeable XFS file systems that are listed in `/etc/mtab`. For a period of two hours, the command passes over each file system in turn, attempting to defragment the top ten percent of files that have the greatest number of extents. After two hours, the command records its progress in the file `/var/tmp/.fsrlast_xfs`, and it resumes from that point if you run the command again.

For more information, see the `xfs_fsr(8)` manual page.

5.10 Checking and Repairing an XFS File System

Note

If you have an Oracle Linux Premier Support account and encounter a problem mounting an XFS file system, send a copy of the `/var/log/messages` file to Oracle Support and wait for advice.

If you cannot mount an XFS file system, you can use the `xfs_check` command to check its consistency. Usually, you would only run this command on the device file of an unmounted file system that you believe has a problem. If `xfs_check` displays any output when you do not run it in verbose mode, the file system has an inconsistency.

```
# xfscheck device
```

If you can mount the file system and you do not have a suitable backup, you can use `xfsdump` to attempt to back up the existing file system data, However, the command might fail if the file system's metadata has become too corrupted.

You can use the `xfs_repair` command to attempt to repair an XFS file system specified by its device file. The command replays the journal log to fix any inconsistencies that might have resulted from the file system not being cleanly unmounted. Unless the file system has an inconsistency, it is usually not necessary to use the command, as the journal is replayed every time that you mount an XFS file system.

```
# xfs_repair device
```

If the journal log has become corrupted, you can reset the log by specifying the `-L` option to `xfs_repair`.

Warning

Resetting the log can leave the file system in an inconsistent state, resulting in data loss and data corruption. Unless you are experienced in debugging and repairing XFS file systems using `xfs_db`, it is recommended that you instead recreate the file system and restore its contents from a backup.

If you cannot mount the file system or you do not have a suitable backup, running `xfs_repair` is the only viable option unless you are experienced in using `xfs_db`.

`xfs_db` provides an internal command set that allows you to debug and repair an XFS file system manually. The commands allow you to perform scans on the file system, and to navigate and display its data structures. If you specify the `-x` option to enable expert mode, you can modify the data structures.

```
# xfs_db [-x] device
```

For more information, see the `xfs_check(8)`, `xfs_db(8)` and `xfs_repair(8)` manual pages, and the `help` command within `xfs_db`.

5.11 For More Information About XFS

You can find more information about XFS at http://xfs.org/index.php/XFS_Papers_and_Documentation.

Chapter 6 Oracle Cluster File System Version 2

Table of Contents

This chapter describes how to configure and use the Oracle Cluster File System Version 2 (OCFS2) file system.

6.1 About OCFS2

Oracle Cluster File System version 2 (OCFS2) is a general-purpose, high-performance, high-availability, shared-disk file system intended for use in clusters. It is also possible to mount an OCFS2 volume on a standalone, non-clustered system.

Although it might seem that there is no benefit in mounting `ocfs2` locally as compared to alternative file systems such as `ext4` or `btrfs`, you can use the `reflink` command with OCFS2 to create copy-on-write clones of individual files in a similar way to using the `cp --reflink` command with the btrfs file system. Typically, such clones allow you to save disk space when storing multiple copies of very similar files, such as VM images or Linux Containers. In addition, mounting a local OCFS2 file system allows you to subsequently migrate it to a cluster file system without requiring any conversion. Note that when using the `reflink` command, the resulting filesystem behaves like a clone of the original filesystem. This means that their UUIDs are identical. When using `reflink` to create a clone, you must change the UUID using the `tunefs.ocfs2` command. See Section 6.2.10, “Querying and Changing Volume Parameters” for more information.

Almost all applications can use OCFS2 as it provides local file-system semantics. Applications that are cluster-aware can use cache-coherent parallel I/O from multiple cluster nodes to balance activity across the cluster, or they can use of the available file-system functionality to fail over and run on another node in the event that a node fails. The following examples typify some use cases for OCFS2:

- Oracle VM to host shared access to virtual machine images.
- Oracle VM and VirtualBox to allow Linux guest machines to share a file system.
- Oracle Real Application Cluster (RAC) in database clusters.
- Oracle E-Business Suite in middleware clusters.

OCFS2 has a large number of features that make it suitable for deployment in an enterprise-level computing environment:

- Support for ordered and write-back data journaling that provides file system consistency in the event of power failure or system crash.
- Block sizes ranging from 512 bytes to 4 KB, and file-system cluster sizes ranging from 4 KB to 1 MB (both in increments of powers of 2). The maximum supported volume size is 16 TB, which corresponds to a cluster size of 4 KB. A volume size as large as 4 PB is theoretically possible for a cluster size of 1 MB, although this limit has not been tested.
- Extent-based allocations for efficient storage of very large files.
- Optimized allocation support for sparse files, inline-data, unwritten extents, hole punching, reflinks, and allocation reservation for high performance and efficient storage.
- Indexing of directories to allow efficient access to a directory even if it contains millions of objects.
- Metadata checksums for the detection of corrupted inodes and directories.
- Extended attributes to allow an unlimited number of `name:value` pairs to be attached to file system objects such as regular files, directories, and symbolic links.
- Advanced security support for POSIX ACLs and SELinux in addition to the traditional file-access permission model.
- Support for user and group quotas.
- Support for heterogeneous clusters of nodes with a mixture of 32-bit and 64-bit, little-endian (x86, x86_64, ia64) and big-endian (ppc64) architectures.
- An easy-to-configure, in-kernel cluster-stack (O2CB) with a distributed lock manager (DLM), which manages concurrent access from the cluster nodes.
- Support for buffered, direct, asynchronous, splice and memory-mapped I/O.
- A tool set that uses similar parameters to the `ext3` file system.

6.2 Installing and Configuring OCFS2

The procedures in the following sections describe how to set up a cluster to use OCFS2.

- Section 6.2.1, “Preparing a Cluster for OCFS2”
- Section 6.2.2, “Configuring the Firewall”
- Section 6.2.3, “Configuring the Cluster Software”
- Section 6.2.4, “Creating the Configuration File for the Cluster Stack”
- Section 6.2.5, “Configuring the Cluster Stack”

- Section 6.2.6, “Configuring the Kernel for Cluster Operation”
- Section 6.2.7, “Starting and Stopping the Cluster Stack”
- Section 6.2.9, “Mounting OCFS2 Volumes”

6.2.1 Preparing a Cluster for OCFS2

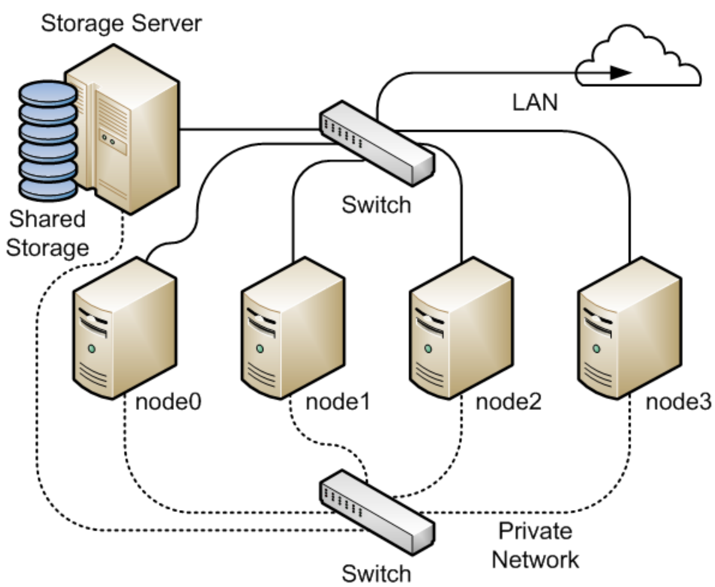
For best performance, each node in the cluster should have at least two network interfaces. One interface is connected to a public network to allow general access to the systems. The other interface is used for private communication between the nodes; the *cluster heartbeat* that determines how the cluster nodes coordinate their access to shared resources and how they monitor each other's state. These interface must be connected via a network switch. Ensure that all network interfaces are configured and working before continuing to configure the cluster.

You have a choice of two cluster heartbeat configurations:

- Local heartbeat thread for each shared device. In this mode, a node starts a heartbeat thread when it mounts an OCFS2 volume and stops the thread when it unmounts the volume. This is the default heartbeat mode. There is a large CPU overhead on nodes that mount a large number of OCFS2 volumes as each mount requires a separate heartbeat thread. A large number of mounts also increases the risk of a node fencing itself out of the cluster due to a heartbeat I/O timeout on a single mount.
- Global heartbeat on specific shared devices. You can configure any OCFS2 volume as a global heartbeat device provided that it occupies a whole disk device and not a partition. In this mode, the heartbeat to the device starts when the cluster comes online and stops when the cluster goes offline. This mode is recommended for clusters that mount a large number of OCFS2 volumes. A node fences itself out of the cluster if a heartbeat I/O timeout occurs on more than half of the global heartbeat devices. To provide redundancy against failure of one of the devices, you should therefore configure at least three global heartbeat devices.

Figure 6.1 shows a cluster of four nodes connected via a network switch to a LAN and a network storage server. The nodes and the storage server are also connected via a switch to a private network that they use for the local cluster heartbeat.

Figure 6.1 Cluster Configuration Using a Private Network



It is possible to configure and use OCFS2 without using a private network but such a configuration increases the probability of a node fencing itself out of the cluster due to an I/O heartbeat timeout.

6.2.2 Configuring the Firewall

Configure or disable the firewall on each node to allow access on the interface that the cluster will use for private cluster communication. By default, the cluster uses both TCP and UDP over port 7777.

To allow incoming TCP connections and UDP datagrams on port 7777 from the private network, use the following commands:

```
# iptables -I INPUT -s subnet_addr/prefix_length -p tcp \
  -m state --state NEW -m tcp --dport 7777 -j ACCEPT
# iptables -I INPUT -s subnet_addr/prefix_length -p udp \
  -m udp --dport 7777 -j ACCEPT
# service iptables save
```

where *subnet_addr/prefix_length* specifies the network address of the private network, for example `10.0.1.0/24`.

6.2.3 Configuring the Cluster Software

Ideally, each node should be running the same version of the OCFS2 software and a compatible version of the Oracle Linux Unbreakable Enterprise Kernel (UEK). It is possible for a cluster to run with mixed versions of the OCFS2 and UEK software, for example, while you are performing a rolling update of a cluster. The cluster node that is running the lowest version of the software determines the set of usable features.

Use `yum` to install or upgrade the following packages to the same version on each node:

- `kernel-uek`
- `ocfs2-tools`

Note

If you want to use the global heartbeat feature, you must install `ocfs2-tools-1.8.0-11` or later.

6.2.4 Creating the Configuration File for the Cluster Stack

You can create the configuration file by using the `o2cb` command or a text editor.

Configure the cluster stack by using the `o2cb` command:

1. Use the following command to create a cluster definition.

   ```
   # o2cb add-cluster cluster_name
   ```

   For example, you would define a cluster named `mycluster` with four nodes as follows:

   ```
   # o2cb add-cluster mycluster
   ```

   The command creates the configuration file `/etc/ocfs2/cluster.conf` if it does not already exist.

2. For each node, use the following command to define the node.

   ```
   # o2cb add-node cluster_name node_name --ip ip_address
   ```

The name of the node must be same as the value of system's `HOSTNAME` that is configured in `/etc/sysconfig/network`. The IP address is the one that the node will use for private communication in the cluster.

For example, to define a node named `node0` with the IP address 10.1.0.100 in the cluster `mycluster`:

```
# o2cb add-node mycluster node0 --ip 10.1.0.100
```

3. If you want the cluster to use global heartbeat devices, use the following commands.

```
# o2cb add-heartbeat cluster_name device1
.
.
.
# o2cb heartbeat-mode cluster_name global
```

Note

You must configure global heartbeat to use whole disk devices. You cannot configure a global heartbeat device on a disk partition.

For example, to use `/dev/sdd`, `/dev/sdg`, and `/dev/sdj` as global heartbeat devices:

```
# o2cb add-heartbeat mycluster /dev/sdd
# o2cb add-heartbeat mycluster /dev/sdg
# o2cb add-heartbeat mycluster /dev/sdj
# o2cb heartbeat-mode mycluster global
```

4. Copy the cluster configuration file `/etc/ocfs2/cluster.conf` to each node in the cluster.

Note

Any changes that you make to the cluster configuration file do not take effect until you restart the cluster stack.

The following sample configuration file `/etc/ocfs2/cluster.conf` defines a 4-node cluster named `mycluster` with a local heartbeat.

```
node:
 name = node0
 cluster = mycluster
 number = 0
 ip_address = 10.1.0.100
 ip_port = 7777

node:
        name = node1
        cluster = mycluster
        number = 1
        ip_address = 10.1.0.101
        ip_port = 7777

node:
        name = node2
        cluster = mycluster
        number = 2
        ip_address = 10.1.0.102
        ip_port = 7777

node:
        name = node3
        cluster = mycluster
```

```
        number = 3
        ip_address = 10.1.0.103
        ip_port = 7777

cluster:
        name = mycluster
        heartbeat_mode = local
        node_count = 4
```

If you configure your cluster to use a global heartbeat, the file also include entries for the global heartbeat devices.

```
node:
        name = node0
        cluster = mycluster
        number = 0
        ip_address = 10.1.0.100
        ip_port = 7777

node:
        name = node1
        cluster = mycluster
        number = 1
        ip_address = 10.1.0.101
        ip_port = 7777

node:
        name = node2
        cluster = mycluster
        number = 2
        ip_address = 10.1.0.102
        ip_port = 7777

node:
        name = node3
        cluster = mycluster
        number = 3
        ip_address = 10.1.0.103
        ip_port = 7777

cluster:
        name = mycluster
        heartbeat_mode = global
        node_count = 4

heartbeat:
        cluster = mycluster
        region = 7DA5015346C245E6A41AA85E2E7EA3CF

heartbeat:
        cluster = mycluster
        region = 4F9FBB0D9B6341729F21A8891B9A05BD

heartbeat:
        cluster = mycluster
        region = B423C7EEE9FC426790FC411972C91CC3
```

The cluster heartbeat mode is now shown as `global`, and the heartbeat regions are represented by the UUIDs of their block devices.

If you edit the configuration file manually, ensure that you use the following layout:

- The `cluster:`, `heartbeat:`, and `node:` headings must start in the first column.
- Each parameter entry must be indented by one tab space.

- A blank line must separate each section that defines the cluster, a heartbeat device, or a node.

6.2.5 Configuring the Cluster Stack

To configure the cluster stack:

1. Run the following command on each node of the cluster:

```
# service o2cb configure
```

The following table describes the values for which you are prompted.

Prompt	Description
`Load O2CB driver on boot (y/n)`	Whether the cluster stack driver should be loaded at boot time. The default response is `n`.
`Cluster stack backing O2CB`	The name of the cluster stack service. The default and usual response is `o2cb`.
`Cluster to start at boot (Enter "none" to clear)`	Enter the name of your cluster that you defined in the cluster configuration file, `/etc/ocfs2/cluster.conf`.
`Specify heartbeat dead threshold (>=7)`	The number of 2-second heartbeats that must elapse without response before a node is considered dead. To calculate the value to enter, divide the required threshold time period by 2 and add 1. For example, to set the threshold time period to 120 seconds, enter a value of 61. The default value is 31, which corresponds to a threshold time period of 60 seconds. **Note** If your system uses multipathed storage, the recommended value is 61 or greater.
`Specify network idle timeout in ms (>=5000)`	The time in milliseconds that must elapse before a network connection is considered dead. The default value is 30,000 milliseconds. **Note** For bonded network interfaces, the recommended value is 30,000 milliseconds or greater.
`Specify network keepalive delay in ms (>=1000)`	The maximum delay in milliseconds between sending keepalive packets to another node. The default and recommended value is 2,000 milliseconds.
`Specify network reconnect delay in ms (>=2000)`	The minimum delay in milliseconds between reconnection attempts if a network connection goes down. The default and recommended value is 2,000 milliseconds.

To verify the settings for the cluster stack, enter the `service o2cb status` command:

```
# service o2cb status
Driver for "configfs": Loaded
Filesystem "configfs": Mounted
Stack glue driver: Loaded
```

```
Stack plugin "o2cb": Loaded
Driver for "ocfs2_dlmfs": Loaded
Filesystem "ocfs2_dlmfs": Mounted
Checking O2CB cluster "mycluster": Online
  Heartbeat dead threshold: 61
  Network idle timeout: 30000
  Network keepalive delay: 2000
  Network reconnect delay: 2000
  Heartbeat mode: Local
Checking O2CB heartbeat: Active
```

In this example, the cluster is online and is using local heartbeat mode. If no volumes have been configured, the O2CB heartbeat is shown as `Not active` rather than `Active`.

The next example shows the command output for an online cluster that is using three global heartbeat devices:

```
# service o2cb status
Driver for "configfs": Loaded
Filesystem "configfs": Mounted
Stack glue driver: Loaded
Stack plugin "o2cb": Loaded
Driver for "ocfs2_dlmfs": Loaded
Filesystem "ocfs2_dlmfs": Mounted
Checking O2CB cluster "mycluster": Online
  Heartbeat dead threshold: 61
  Network idle timeout: 30000
  Network keepalive delay: 2000
  Network reconnect delay: 2000
  Heartbeat mode: Global
Checking O2CB heartbeat: Active
  7DA5015346C245E6A41AA85E2E7EA3CF /dev/sdd
  4F9FBB0D9B6341729F21A8891B9A05BD /dev/sdg
  B423C7EEE9FC426790FC411972C91CC3 /dev/sdj
```

2. Configure the `o2cb` and `ocfs2` services so that they start at boot time after networking is enabled:

```
# chkconfig o2cb on
# chkconfig ocfs2 on
```

These settings allow the node to mount OCFS2 volumes automatically when the system starts.

6.2.6 Configuring the Kernel for Cluster Operation

For the correct operation of the cluster, you must configure the kernel settings shown in the following table:

Kernel Setting	Description
`panic`	Specifies the number of seconds after a panic before a system will automatically reset itself. If the value is 0, the system hangs, which allows you to collect detailed information about the panic for troubleshooting. This is the default value. To enable automatic reset, set a non-zero value. If you require a memory image (`vmcore`), allow enough time for Kdump to create this image. The suggested value is 30 seconds, although large systems will require a longer time.
`panic_on_oops`	Specifies that a system must panic if a kernel oops occurs. If a kernel thread required for cluster operation crashes, the system must reset itself. Otherwise, another node might not be able to tell whether a node is slow to respond or unable to respond, causing cluster operations to hang.

On each node, enter the following commands to set the recommended values for `panic` and `panic_on_oops`:

```
# sysctl kernel.panic = 30
# sysctl kernel.panic_on_oops = 1
```

To make the change persist across reboots, add the following entries to the `/etc/sysctl.conf` file:

```
# Define panic and panic_on_oops for cluster operation
kernel.panic = 30
kernel.panic_on_oops = 1
```

6.2.7 Starting and Stopping the Cluster Stack

The following table shows the commands that you can use to perform various operations on the cluster stack.

Command	Description
`service o2cb status`	Check the status of the cluster stack.
`service o2cb online`	Start the cluster stack.
`service o2cb offline`	Stop the cluster stack.
`service o2cb unload`	Unload the cluster stack.

6.2.8 Creating OCFS2 volumes

You can use the `mkfs.ocfs2` command to create an OCFS2 volume on a device. If you want to label the volume and mount it by specifying the label, the device must correspond to a partition. You cannot mount an unpartitioned disk device by specifying a label. The following table shows the most useful options that you can use when creating an OCFS2 volume.

Command Option	Description
`-b` *`block-size`* `--block-size` *`block-size`*	Specifies the unit size for I/O transactions to and from the file system, and the size of inode and extent blocks. The supported block sizes are 512 (512 bytes), 1K, 2K, and 4K. The default and recommended block size is 4K (4 kilobytes).
`-C` *`cluster-size`* `--cluster-size` *`cluster-size`*	Specifies the unit size for space used to allocate file data. The supported cluster sizes are 4K, 8K, 16K, 32K, 64K, 128K, 256K, 512K, and 1M (1 megabyte). The default cluster size is 4K (4 kilobytes).
`--fs-feature-level=`*`feature-level`*	Allows you select a set of file-system features: `default` — Enables support for the sparse files, unwritten extents, and inline data features. `max-compat` — Enables only those features that are understood by older versions of OCFS2. `max-features` — Enables all features that OCFS2 currently supports.

Command Option	**Description**
`--fs_features=`*`feature`*	Allows you to enable or disable individual features such as support for sparse files, unwritten extents, and backup superblocks. For more information, see the `mkfs.ocfs2(8)` manual page.
`-J size=`*`journal-size`* `--journal-options size=`*`journal-size`*	Specifies the size of the write-ahead journal. If not specified, the size is determined from the file system usage type that you specify to the `-T` option, and, otherwise, from the volume size. The default size of the journal is 64M (64 MB) for `datafiles`, 256M (256 MB) for `mail`, and 128M (128 MB) for `vmstore`.
`-L` *`volume-label`* `--label` *`volume-label`*	Specifies a descriptive name for the volume that allows you to identify it easily on different cluster nodes.
`-N` *`number`* `--node-slots` *`number`*	Determines the maximum number of nodes that can concurrently access a volume, which is limited by the number of node slots for system files such as the file-system journal. For best performance, set the number of node slots to at least twice the number of nodes. If you subsequently increase the number of node slots, performance can suffer because the journal will no longer be contiguously laid out on the outer edge of the disk platter.
`-T` *`file-system-usage-type`*	Specifies the type of usage for the file system: `datafiles`: Database files are typically few in number, fully allocated, and relatively large. Such files require few metadata changes, and do not benefit from having a large journal. `mail`: Mail server files are typically many in number, and relatively small. Such files require many metadata changes, and benefit from having a large journal. `vmstore`: Virtual machine image files are typically few in number, sparsely allocated, and relatively large. Such files require a moderate number of metadata changes and a medium sized journal.

For example, create an OCFS2 volume on `/dev/sdc1` labeled as `myvol` using all the default settings for generic usage (4 KB block and cluster size, eight node slots, a 256 MB journal, and support for default file-system features).

```
# mkfs.ocfs2 -L "myvol" /dev/sdc1
```

Create an OCFS2 volume on `/dev/sdd2` labeled as `dbvol` for use with database files. In this case, the cluster size is set to 128 KB and the journal size to 32 MB.

```
# mkfs.ocfs2 -L "dbvol" -T datafiles /dev/sdd2
```

Create an OCFS2 volume on `/dev/sde1` with a 16 KB cluster size, a 128 MB journal, 16 node slots, and support enabled for all features except refcount trees.

```
# mkfs.ocfs2 -C 16K -J size=128M -N 16 --fs-feature-level=max-features \
  --fs-features=norefcount /dev/sde1
```

Note

Do not create an OCFS2 volume on an LVM logical volume. LVM is not cluster-aware.

You cannot change the block and cluster size of an OCFS2 volume after it has been created. You can use the `tunefs.ocfs2` command to modify other settings for the file system with certain restrictions. For more information, see the `tunefs.ocfs2(8)` manual page.

If you intend the volume to store database files, do not specify a cluster size that is smaller than the block size of the database.

The default cluster size of 4 KB is not suitable if the file system is larger than a few gigabytes. The following table suggests minimum cluster size settings for different file system size ranges:

File System Size	Suggested Minimum Cluster Size
1 GB - 10 GB	8K
10GB - 100 GB	16K
100 GB - 1 TB	32K
1 TB - 10 TB	64K
10 TB - 16 TB	128K

6.2.9 Mounting OCFS2 Volumes

As shown in the following example, specify the `_netdev` option in `/etc/fstab` if you want the system to mount an OCFS2 volume at boot time after networking is started, and to unmount the file system before networking is stopped.

```
myocfs2vol  /dbvol1  ocfs2     _netdev,defaults  0 0
```

Note

The file system will not mount unless you have enabled the `o2cb` and `ocfs2` services to start after networking is started. See Section 6.2.5, “Configuring the Cluster Stack”.

6.2.10 Querying and Changing Volume Parameters

You can use the `tunefs.ocfs2` command to query or change volume parameters. For example, to find out the label, UUID and the number of node slots for a volume:

```
# tunefs.ocfs2 -Q "Label = %V\nUUID = %U\nNumSlots =%N\n" /dev/sdb
Label = myvol
UUID = CBB8D5E0C169497C8B52A0FD555C7A3E
NumSlots = 4
```

Generate a new UUID for a volume:

```
# tunefs.ocfs2 -U /dev/sda
```

```
# tunefs.ocfs2 -Q "Label = %V\nUUID = %U\nNumSlots =%N\n" /dev/sdb
Label = myvol
UUID = 48E56A2BBAB34A9EB1BE832B3C36AB5C
NumSlots = 4
```

6.3 Troubleshooting OCFS2

The following sections describes some techniques that you can use for investigating any problems that you encounter with OCFS2.

6.3.1 Recommended Tools for Debugging

To you want to capture an oops trace, it is recommended that you set up `netconsole` on the nodes.

If you want to capture the DLM's network traffic between the nodes, you can use tcpdump. For example, to capture TCP traffic on port 7777 for the private network interface `eth1`, you could use a command such as the following:

```
# tcpdump -i eth1 -C 10 -W 15 -s 10000 -Sw /tmp/`hostname -s`_tcpdump.log \
  -ttt 'port 7777' &
```

You can use the `debugfs.ocfs2` command, which is similar in behavior to the `debugfs` command for the `ext3` file system, and allows you to trace events in the OCFS2 driver, determine lock statuses, walk directory structures, examine inodes, and so on.

For more information, see the `debugfs.ocfs2(8)` manual page.

The `o2image` command saves an OCFS2 file system's metadata (including information about inodes, file names, and directory names) to an image file on another file system. As the image file contains only metadata, it is much smaller than the original file system. You can use `debugfs.ocfs2` to open the image file, and analyze the file system layout to determine the cause of a file system corruption or performance problem.

For example, the following command creates the image `/tmp/sda2.img` from the OCFS2 file system on the device `/dev/sda2`:

```
# o2image /dev/sda2 /tmp/sda2.img
```

For more information, see the `o2image(8)` manual page.

6.3.2 Mounting the debugfs File System

OCFS2 uses the `debugfs` file system to allow access from user space to information about its in-kernel state. You must mount the `debugfs` file system to be able to use the `debugfs.ocfs2` command.

To mount the `debugfs` file system, add the following line to `/etc/fstab`:

```
debugfs    /sys/kernel/debug      debugfs  defaults  0 0
```

and run the `mount -a` command.

6.3.3 Configuring OCFS2 Tracing

The following table shows some of the commands that are useful for tracing problems in OCFS2.

Command	Description
`debugfs.ocfs2 -l`	List all trace bits and their statuses.
`debugfs.ocfs2 -l SUPER allow`	Enable tracing for the superblock.
`debugfs.ocfs2 -l SUPER off`	Disable tracing for the superblock.
`debugfs.ocfs2 -l SUPER deny`	Disallow tracing for the superblock, even if implicitly enabled by another tracing mode setting.
`debugfs.ocfs2 -l HEARTBEAT \` `ENTRY EXIT allow`	Enable heartbeat tracing.
`debugfs.ocfs2 -l HEARTBEAT off \` `ENTRY EXIT deny`	Disable heartbeat tracing. `ENTRY` and `EXIT` are set to `deny` as they exist in all trace paths.
`debugfs.ocfs2 -l ENTRY EXIT \` `NAMEI INODE allow`	Enable tracing for the file system.
`debugfs.ocfs2 -l ENTRY EXIT \` `deny NAMEI INODE allow`	Disable tracing for the file system.
`debugfs.ocfs2 -l ENTRY EXIT \` `DLM DLM_THREAD allow`	Enable tracing for the DLM.
`debugfs.ocfs2 -l ENTRY EXIT \` `deny DLM DLM_THREAD allow`	Disable tracing for the DLM.

One method for obtaining a trace its to enable the trace, sleep for a short while, and then disable the trace. As shown in the following example, to avoid seeing unnecessary output, you should reset the trace bits to their default settings after you have finished.

```
# debugfs.ocfs2 -l ENTRY EXIT NAMEI INODE allow && sleep 10 && \
  debugfs.ocfs2 -l ENTRY EXIT deny NAMEI INODE off
```

To limit the amount of information displayed, enable only the trace bits that you believe are relevant to understanding the problem.

If you believe a specific file system command, such as `mv`, is causing an error, the following example shows the commands that you can use to help you trace the error.

```
# debugfs.ocfs2 -l ENTRY EXIT NAMEI INODE allow
# mv source destination & CMD_PID=$(jobs -p %-)
# echo $CMD_PID
# debugfs.ocfs2 -l ENTRY EXIT deny NAMEI INODE off
```

As the trace is enabled for all mounted OCFS2 volumes, knowing the correct process ID can help you to interpret the trace.

For more information, see the `debugfs.ocfs2(8)` manual page.

6.3.4 Debugging File System Locks

If an OCFS2 volume hangs, you can use the following steps to help you determine which locks are busy and the processes that are likely to be holding the locks.

1. Mount the debug file system.

```
# mount -t debugfs debugfs /sys/kernel/debug
```

2. Dump the lock statuses for the file system device (`/dev/sdx1` in this example).

```
# echo "fs_locks" | debugfs.ocfs2 /dev/sdx1 >/tmp/fslocks 62
Lockres: M000000000000006672078b84822 Mode: Protected Read
Flags: Initialized Attached
RO Holders: 0 EX Holders: 0
Pending Action: None Pending Unlock Action: None
Requested Mode: Protected Read Blocking Mode: Invalid
```

The `Lockres` field is the lock name used by the DLM. The lock name is a combination of a lock-type identifier, an inode number, and a generation number. The following table shows the possible lock types.

Identifier	Lock Type
`D`	File data.
`M`	Metadata.
`R`	Rename.
`S`	Superblock.
`W`	Read-write.

3. Use the `Lockres` value to obtain the inode number and generation number for the lock.

```
# echo "stat <M000000000000006672078b84822>" | debugfs.ocfs2 -n /dev/sdx1
Inode: 419616   Mode: 0666   Generation: 2025343010 (0x78b84822)
...
```

4. Determine the file system object to which the inode number relates by using the following command.

```
# echo "locate <419616>" | debugfs.ocfs2 -n /dev/sdx1
419616 /linux-2.6.15/arch/i386/kernel/semaphore.c
```

5. Obtain the lock names that are associated with the file system object.

```
# echo "encode /linux-2.6.15/arch/i386/kernel/semaphore.c" | \
  debugfs.ocfs2 -n /dev/sdx1
M000000000000006672078b84822 D000000000000006672078b84822 W000000000000006672078b84822
```

In this example, a metadata lock, a file data lock, and a read-write lock are associated with the file system object.

6. Determine the DLM domain of the file system.

```
# echo "stats" | debugfs.ocfs2 -n /dev/sdX1 | grep UUID: | while read a b ; do echo $b ; done
82DA8137A49A47E4B187F74E09FBBB4B
```

7. Use the values of the DLM domain and the lock name with the following command, which enables debugging for the DLM.

```
# echo R 82DA8137A49A47E4B187F74E09FBBB4B \
  M000000000000006672078b84822 > /proc/fs/ocfs2_dlm/debug
```

8. Examine the debug messages.

```
# dmesg | tail
struct dlm_ctxt: 82DA8137A49A47E4B187F74E09FBBB4B, node=3, key=965960985
  lockres: M000000000000006672078b84822, owner=1, state=0 last used: 0,
  on purge list: no granted queue:
```

```
    type=3, conv=-1, node=3, cookie=11673330234144325711, ast=(empty=y,pend=n),
    bast=(empty=y,pend=n)
  converting queue:
  blocked queue:
```

The DLM supports 3 lock modes: no lock (`type=0`), protected read (`type=3`), and exclusive (`type=5`). In this example, the lock is mastered by node 1 (`owner=1`) and node 3 has been granted a protected-read lock on the file-system resource.

9. Run the following command, and look for processes that are in an uninterruptable sleep state as shown by the `D` flag in the `STAT` column.

```
# ps -e -o pid,stat,comm,wchan=WIDE-WCHAN-COLUMN
```

At least one of the processes that are in the uninterruptable sleep state will be responsible for the hang on the other node.

If a process is waiting for I/O to complete, the problem could be anywhere in the I/O subsystem from the block device layer through the drivers to the disk array. If the hang concerns a user lock (`flock()`), the problem could lie in the application. If possible, kill the holder of the lock. If the hang is due to lack of memory or fragmented memory, you can free up memory by killing non-essential processes. The most immediate solution is to reset the node that is holding the lock. The DLM recovery process can then clear all the locks that the dead node owned, so letting the cluster continue to operate.

6.3.5 Configuring the Behavior of Fenced Nodes

If a node with a mounted OCFS2 volume believes that it is no longer in contact with the other cluster nodes, it removes itself from the cluster in a process termed *fencing*. Fencing prevents other nodes from hanging when they try to access resources held by the fenced node. By default, a fenced node restarts instead of panicking so that it can quickly rejoin the cluster. Under some circumstances, you might want a fenced node to panic instead of restarting. For example, you might want to use `netconsole` to view the oops stack trace or to diagnose the cause of frequent reboots. To configure a node to panic when it next fences, run the following command on the node after the cluster starts:

```
# echo panic > /sys/kernel/config/cluster/cluster_name/fence_method
```

where *`cluster_name`* is the name of the cluster. To set the value after each reboot of the system, add this line to `/etc/rc.local`. To restore the default behavior, use the value `reset` instead of `panic`.

6.4 Use Cases for OCFS2

The following sections describe some typical use cases for OCFS2.

6.4.1 Load Balancing

You can use OCFS2 nodes to share resources between client systems. For example, the nodes could export a shared file system by using Samba or NFS. To distribute service requests between the nodes, you can use round-robin DNS, a network load balancer, or specify which node should be used on each client.

6.4.2 Oracle Real Application Cluster (RAC)

Oracle RAC uses its own cluster stack, Cluster Synchronization Services (CSS). You can use O2CB in conjunction with CSS, but you should note that each stack is configured independently for timeouts, nodes, and other cluster settings. You can use OCFS2 to host the voting disk files and the Oracle cluster registry (OCR), but not the grid infrastructure user's home, which must exist on a local file system on each node.

As both CSS and O2CB use the lowest node number as a tie breaker in quorum calculations, you should ensure that the node numbers are the same in both clusters. If necessary, edit the O2CB configuration file `/etc/ocfs2/cluster.conf` to make the node numbering consistent, and update this file on all nodes. The change takes effect when the cluster is restarted.

6.4.3 Oracle Databases

Specify the `noatime` option when mounting volumes that host Oracle datafiles, control files, redo logs, voting disk, and OCR. The `noatime` option disables unnecessary updates to the access time on the inodes.

Specify the `nointr` mount option to prevent signals interrupting I/O transactions that are in progress.

By default, the `init.ora` parameter `filesystemio_options` directs the database to perform direct I/O to the Oracle datafiles, control files, and redo logs. You should also specify the `datavolume` mount option for the volumes that contain the voting disk and OCR. Do not specify this option for volumes that host the Oracle user's home directory or Oracle E-Business Suite.

To avoid database blocks becoming fragmented across a disk, ensure that the file system cluster size is at least as big as the database block size, which is typically 8KB. If you specify the file system usage type as `datafiles` to the `mkfs.ocfs2` command, the file system cluster size is set to 128KB.

To allow multiple nodes to maximize throughput by concurrently streaming data to an Oracle datafile, OCFS2 deviates from the POSIX standard by not updating the modification time (`mtime`) on the disk when performing non-extending direct I/O writes. The value of `mtime` is updated in memory, but OCFS2 does not write the value to disk unless an application extends or truncates the file, or performs a operation to change the file metadata, such as using the `touch` command. This behavior leads to results in different nodes reporting different time stamps for the same file. You can use the following command to view the on-disk timestamp of a file:

```
# debugfs.ocfs2 -R "stat /file_path" device | grep "mtime:"
```

6.5 For More Information About OCFS2

You can find more information about OCFS2 at https://oss.oracle.com/projects/ocfs2/documentation/.

Chapter 7 Control Groups

Table of Contents

This chapter describes how to use Control Groups (`cgroups`) to manage the resource utilization of sets of processes.

7.1 About cgroups

A cgroup is a collection of processes (*tasks*) that you bind together by applying a set of criteria that control the `cgroup`'s access to system resources. You can create a hierarchy of `cgroups`, in which child `cgroups` inherit its characteristics from the parent `cgroup`. You can use `cgroups` to manage processes in the following ways:

- Limit the CPU, I/O, and memory resources that are available to a group.
- Change the priority of a group relative to other groups.
- Measure a group's resource usage for accounting and billing purposes.
- Isolate a group's files, processes, and network connections from other groups.
- Freeze a group to allow you to create a checkpoint.

You can create and manage `cgroups` in the following ways:

- By editing the `cgroup` configuration file `/etc/cgconfig.conf`.
- By using `cgroups` commands such as `cgcreate`, `cgclassify`, and `cgexec`.

- By manipulating a cgroup's virtual file system, for example, by adding process IDs to `tasks` directories under `/sys/fs/cgroup`.
- By editing the cgroup rules file `/etc/cgrules.conf` so that the rules engine or PAM move processes into `cgroups` automatically.
- By using additional application software such as Linux Containers.
- By using the APIs that are provided in `libvirt`.

Because you might ultimately want to deploy `cgroups` in a production environment, this chapter demonstrates how to configure `cgroups` by editing the `/etc/cgconfig.conf` and `/etc/cgrules.conf` files, and how to configure PAM to associate processes with `cgroups`.

Note

To use `cgroups`, you must install the `libcgroup` package on your system.

7.2 Subsystems

You control the access that `cgroups` have to system resources by specifying parameters to various kernel modules known as *subsystems* (or as *resource controllers* in some `cgroups` documentation).

The following table lists the subsystems that are provided with the `cgroups` package.

Subsystem	Description
`blkio`	Controls and reports block I/O operations. See Section 7.2.1, “blkio Parameters”. **Note** The `blkio` subsystem is enabled in the 2.6.39 UEK, but not in the 2.6.32 UEK.
`cpu`	Controls access to CPU resources. See Section 7.2.2, “cpu Parameters”.
`cpuacct`	Reports usage of CPU resources. See Section 7.2.3, “cpuacct Parameters”.
`cpuset`	Controls access to CPU cores and memory nodes (for systems with NUMA architectures). See Section 7.2.4, “cpuset Parameters”.
`devices`	Controls access to system devices. See Section 7.2.5, “devices Parameters”.
`freezer`	Suspends or resumes cgroup tasks. See Section 7.2.6, “freezer Parameter”.
`memory`	Controls access to memory resources, and reports on memory usage. See Section 7.2.7, “memory Parameters”.
`net_cls`	Tags network packets for use by network traffic control. See Section 7.2.8, “net_cls Parameter”.

You can set the following parameters for each subsystem.

7.2.1 blkio Parameters

The following `blkio` parameters are defined:

`blkio.io_merged`

Reports the number of BIOS requests that have been merged into `async`, `read`, `sync`, or `write` I/O operations.

`blkio.io_queued`

Reports the number of requests for `async`, `read`, `sync`, or `write` I/O operations.

`blkio.io_service_bytes`

Reports the number of bytes transferred by `async`, `read`, `sync`, or `write` I/O operations to or from the devices specified by their major and minor numbers as recorded by the completely fair queueing (CFQ) scheduler, but not updated while it is operating on a request queue.

`blkio.io_serviced`

Reports the number of `async`, `read`, `sync`, or `write` I/O operations to or from the devices specified by their major and minor numbers as recorded by the CFQ scheduler, but not updated while it is operating on a request queue.

`blkio.io_service_time`

Reports the time in nanoseconds taken to complete `async`, `read`, `sync`, or `write` I/O operations to or from the devices specified by their major and minor numbers.

`blkio.io_wait_time`

Reports the total time in nanoseconds that a cgroup spent waiting for `async`, `read`, `sync`, or `write` I/O operations to complete to or from the devices specified by their major and minor numbers.

`blkio.reset_stats`

Resets the statistics for a cgroup if an integer is written to this parameter.

`blkio.sectors`

Reports the number of disk sectors written to or read from the devices specified by their major and minor numbers.

`blkio.throttle.io_service_bytes`

Reports the number of bytes transferred by `async`, `read`, `sync`, or `write` I/O operations to or from the devices specified by their major and minor numbers even while the CFQ scheduler is operating on a request queue.

`blkio.throttle.io_serviced`

Reports the number of `async`, `read`, `sync`, or `write` I/O operations to or from the devices specified by their major and minor numbers even while the CFQ scheduler is operating on a request queue.

`blkio.throttle.read_bps_device`

Specifies the maximum number of bytes per second that a cgroup may read from a device specified by its major and minor numbers. For example, the setting `8:1 4194304` specifies that a maximum of 4 MB per second may be read from `/dev/sda1`.

`blkio.throttle.read_iops_device`

Specifies the maximum number of read operations per second that a cgroup may perform on a device specified by its major and minor numbers. For example, the setting `8:1 100` specifies that a maximum of 100 read operations per second may be performed on`/dev/sda1`.

blkio.throttle.write_bps_device

Specifies the maximum number of bytes per second that a cgroup may write to a device specified by its major and minor numbers. For example, the setting `8:2 2097152` specifies a maximum of 2 MB per second may be written to `/dev/sda2`.

blkio.throttle.write_iops_device

Specifies the maximum number of write operations per second that a cgroup may perform on a device specified by its major and minor numbers. For example, the setting `8:2 50` specifies that a maximum of 50 write operations per second may be performed on `/dev/sda2`.

blkio.time

Reports the time in milliseconds that I/O access was available to a device specified by its major and minor numbers.

blkio.weight

Specifies a bias value from 100 to 1000 that determines a cgroup's share of access to block I/O. The default value is 1000. The value is overridden by the setting for an individual device (see `blkio.weight_device`).

blkio.weight_device

Specifies a bias value from 100 to 1000 that determines a cgroup's share of access to block I/O on a device specified by its major and minor numbers. For example, the setting `8:17 100` specifies a bias value of 100 for `/dev/sdb1`.

7.2.2 cpu Parameters

The following `cpu` parameters are defined:

cpu.rt_period_us

Specifies how often in microseconds that a cgroup's access to a CPU should be rescheduled. The default value is 1000000 (1 second).

cpu.rt_runtime.us

Specifies for how long in microseconds that a cgroup has access to a CPU between rescheduling operations. The default value is 950000 (0.95 seconds).

cpu.shares

Specifies the bias value that determines a cgroup's share of CPU time. The default value is 1024.

7.2.3 cpuacct Parameters

The following `cpuacct` parameters are defined:

cpuacct.stat

Reports the total CPU time in nanoseconds spent in user and system mode by all tasks in the cgroup.

cpuacct.usage

Reports the total CPU time in nanoseconds for all tasks in the cgroup. Setting this parameter to 0 resets its value, and also resets the value of `cpuacct.usage_percpu`.

cpuacct.usage_percpu

Reports the total CPU time in nanoseconds on each CPU core for all tasks in the cgroup.

7.2.4 cpuset Parameters

The following `cpuset` parameters are defined:

cpuset.cpu_exclusive

Specifies whether the CPUs specified by `cpuset.cpus` are exclusively allocated to this CPU set and cannot be shared with other CPU sets. The default value of 0 specifies that CPUs are not exclusively allocated. A value of 1 enables exclusive use of the CPUs by a CPU set.

cpuset.cpus

Specifies a list of CPU cores to which a cgroup has access. For example, the setting `0,1,5-8` allows access to cores 0, 1, 5, 6, 7, and 8. The default setting includes all the available CPU cores.

> **Note**
>
> If you associate the `cpuset` subsystem with a cgroup, you must specify a value for the `cpuset.cpus` parameter.

cpuset.mem_exclusive

Specifies whether the memory nodes specified by `cpuset.mems` are exclusively allocated to this CPU set and cannot be shared with other CPU sets. The default value of 0 specifies that memory nodes are not exclusively allocated. A value of 1 enables exclusive use of the memory nodes by a CPU set.

cpuset.mem_hardwall

Specifies whether the kernel allocates pages and buffers to the memory nodes specified by `cpuset.mems` exclusively to this CPU set and cannot be shared with other CPU sets. The default value of 0 specifies that memory nodes are not exclusively allocated. A value of 1 allows you to separate the memory nodes that are allocated to different `cgroups`.

cpuset.memory_migrate

Specifies whether memory pages are allowed to migrate between memory nodes if the value of `cpuset.mems` changes. The default value of 0 specifies that memory nodes are not allowed to migrate. A value of 1 allows pages to migrate between memory nodes, maintaining their relative position on the node list where possible.

cpuset.memory_pressure

If `cpuset.memory_pressure_enabled` has been set to 1, reports the *memory pressure*, which represents the number of attempts per second by processes to reclaim in-use memory. The reported value scales the actual number of attempts up by a factor of 1000.

`cpuset.memory_pressure_enabled`

Specifies whether the memory pressure statistic should be gathered. The default value of 0 disables the counter. A value of 1 enables the counter.

`cpuset.memory_spread_page`

Specifies whether file system buffers are distributed between the allocated memory nodes. The default value of 0 results in the buffers being placed on the same memory node as the process that owns them. A value of 1 allows the buffers to be distributed across the memory nodes of the CPU set.

`cpuset.memory_spread_slab`

Specifies whether I/O slab caches are distributed between the allocated memory nodes. The default value of 0 results in the caches being placed on the same memory node as the process that owns them. A value of 1 allows the caches to be distributed across the memory nodes of the CPU set.

`cpuset.mems`

Specifies the memory nodes to which a cgroup has access. For example, the setting `0-2,4` allows access to memory nodes 0, 1, 2, and 4. The default setting includes all available memory nodes. The parameter has a value of 0 on systems that do not have a NUMA architecture.

> **Note**
>
> If you associate the `cpuset` subsystem with a cgroup, you must specify a value for the `cpuset.mems` parameter.

`cpuset.sched_load_balance`

Specifies whether the kernel should attempt to balance CPU load by moving processes between the CPU cores allocated to a CPU set. The default value of 1 turns on load balancing. A value of 0 disables load balancing. Disabling load balancing for a cgroup has no effect if load balancing is enabled in the parent cgroup.

`cpuset.sched_relax_domain_level`

If `cpuset.sched_load_balance` is set to 1, specifies one of the following load-balancing schemes.

Setting	Description
`-1`	Use the system's default load balancing scheme. This is the default behavior.
`0`	Perform periodic load balancing. Higher numeric values enable immediate load balancing.
`1`	Perform load balancing for threads running on the same core.
`2`	Perform load balancing for cores of the same CPU.
`3`	Perform load balancing for all CPU cores on the same system.
`4`	Perform load balancing for a subset of CPU cores on a system with a NUMA architecture.
`5`	Perform load balancing for all CPU cores on a system with a NUMA architecture.

7.2.5 devices Parameters

The following `devices` parameters are defined:

`devices.allow`

Specifies a device that a cgroup is allowed to access by its type (`a` for any, `b` for block, or `c` for character), its major and minor numbers, and its access modes (`m` for create permission, `r` for read access, and `w` for write access).

For example, `b 8:17 rw` would allow read and write access to the block device `/dev/sdb1`.

You can use the wildcard `*` to represent any major or minor number. For example, `b 8:* rw` would allow read and write access to any `/dev/sd*` block device.

Each device that you specify is added to the list of allowed devices.

`devices.deny`

Specifies a device that a cgroup is not allowed to access.

Removes each device that you specify from the list of allowed devices.

`devices.list`

Reports those devices for which access control is set. If no devices are controlled, all devices are reported as being available in all access modes: `a *:* rwm`.

7.2.6 freezer Parameter

The following `freezer` parameter is defined:

`freezer.state`

Specifies one of the following operations.

Setting	Description
`FROZEN`	Suspends all the tasks in a cgroup. You cannot move a process into a frozen cgroup.
`THAWED`	Resumes all the tasks in a cgroup.

Note

You cannot set the `FREEZING` state. If displayed, this state indicates that the system is currently suspending the tasks in the cgroup.

The `freezer.state` parameter is not available in the `root` cgroup.

7.2.7 memory Parameters

The following `memory` parameters are defined:

`memory.failcnt`

Specifies the number of times that the amount of memory used by a cgroup has risen to `memory.limit_in_bytes`.

`memory.force_empty`

If a cgroup has no tasks, setting the value to 0 removes all pages from memory that were used by tasks in the cgroup. Setting the parameter in this way avoids a parent cgroup from being assigned the defunct page caches when you remove its child cgroup.

memory.limit_in_bytes

Specifies the maximum usage permitted for user memory including the file cache. The default units are bytes, but you can also specify a `k` or `K`, `m` or `M`, and `g` or `G` suffix for kilobytes, megabytes, and gigabytes respectively. A value of -1 removes the limit.

To avoid an out-of-memory error, set the value of `memory.limit_in_bytes` lower than `memory.memsw.limit_in_bytes`, and set `memory.memsw.limit_in_bytes` lower than the amount of available swap space.

memory.max_usage_in_bytes

Reports the maximum amount of user memory in bytes used by tasks in the cgroup.

memory.memsw.failcnt

Specifies the number of times that the amount of memory and swap space used by a cgroup has risen to `memory.memsw.limit_in_bytes`.

memory.memsw.limit_in_bytes

Specifies the maximum usage permitted for user memory plus swap space. The default units are bytes, but you can also specify a `k` or `K`, `m` or `M`, and `g` or `G` suffix for kilobytes, megabytes, and gigabytes respectively. A value of -1 removes the limit.

memory.memsw.max_usage_in_bytes

Reports the maximum amount of user memory and swap space in bytes used by tasks in the cgroup.

memory.memsw.usage_in_bytes

Reports the total size in bytes of the memory and swap space used by tasks in the cgroup.

memory.move_charge_at_immigrate

Specifies whether a task's charges are moved when you migrate the task between `cgroups`. You can specify the following values.

Setting	Description
`0`	Disable moving task charges.
`1`	Moves charges for an in-use or swapped-out anonymous page exclusively owned by the task.
`2`	Moves charges for file pages that are memory mapped by the task.
`3`	Equivalent to specifying both 1 and 2.

memory.numa_stat

Reports the NUMA memory usage in bytes for each memory node (N0, N1,...) together with the following statistics.

Statistic	Description
`anon`	The size in bytes of anonymous and swap cache.
`file`	The size in bytes of file-backed memory.
`total`	The sum of the `anon`, `file` and `unevictable` values.

Statistic	Description
`unevictable`	The size in bytes of unreclaimable memory.

`memory.oom_control`

Displays the values of the out-of-memory (OOM) notification control feature.

Setting	Description
`oom_kill_disable`	Whether the OOM killer is enabled (0) or disabled (1).
`under_oom`	Whether the cgroup is under OOM control (1) allowing tasks to be stopped, or not under OOM control (0).

`memory.soft_limit_in_bytes`

Specifies a soft, upper limit for user memory including the file cache. The default units are bytes, but you can also specify a `k` or `K`, `m` or `M`, and `g` or `G` suffix for kilobytes, megabytes, and gigabytes respectively. A value of -1 removes the limit.

The soft limit should be lower than the hard-limit value of `memory.limit_in_bytes` as the hard limit always takes precedence.

`memory.stat`

Reports the following memory statistics.

Statistic	Description
`active_anon`	The size in bytes of anonymous and swap cache on active least-recently-used (LRU) list (includes tmpfs).
`active_file`	The size in bytes of file-backed memory on active LRU list.
`cache`	The size in bytes of page cache (includes `tmpfs`).
`hierarchical_memory_limit`	The size in bytes of the limit of memory for the cgroup hierarchy.
`hierarchical_memsw_limit`	The size in bytes of the limit of memory plus swap for the cgroup hierarchy.
`inactive_anon`	The size in bytes of anonymous and swap cache on inactive LRU list (includes `tmpfs`).
`inactive_file`	The size in bytes of file-backed memory on inactive LRU list.
`mapped_file`	The size in bytes of memory-mapped files (includes `tmpfs`).
`pgfault`	The number of page faults, where the kernel has to allocate and initialize physical memory for use in the virtual address space of a process.
`pgmajfault`	The number of major page faults, where the kernel has to actively free physical memory before allocation and initialization.
`pgpgin`	The number of paged-in pages of memory.
`pgpgout`	The number of paged-out pages of memory.
`rss`	The size in bytes of anonymous and swap cache (does not include `tmpfs`). The actual resident set size is given by the sum of `rss` and `mapped_file`.
`swap`	The size in bytes of used swap space.

Statistic	Description
total_*	The value of the appended statistic for the cgroup and all of its children.
unevictable	The size in bytes of memory that in not reclaimable.

memory.swappiness

Specifies a bias value for the kernel to swap out memory pages used by processes in the cgroup rather than reclaim pages from the page cache. A value smaller than the default value of 60 reduces the kernel's preference for swapping out. A value greater than 60 increases the preference for swapping out. A value greater than 100 allows the system to swap out pages that fall within the address space of the cgroup's tasks.

memory.usage_in_bytes

Reports the total size in bytes of the memory used by all the tasks in the cgroup.

memory.use_hierarchy

Specifies whether the kernel should attempt to reclaim memory from a cgroup's hierarchy. The default value of 0 prevents memory from being reclaimed from other tasks in the hierarchy. A value of 1 allows memory to be reclaimed from other tasks in the hierarchy.

7.2.8 net_cls Parameter

The following net_cls parameter is defined:

net_cls.classid

Specifies the hexadecimal class identifier that the system uses to tag network packets for use with the Linux traffic controller.

7.3 Enabling the cgconfig Service

To enable the cgroup services on a system:

1. Install the libcgroup package.

```
# yum install libcgroup
```

2. Start the cgconfig service and configure it to start when the system is booted.

```
# service cgconfig start
# chkconfig cgconfig on
```

7.4 Enabling PAM to Work with cgroup Rules

To configure PAM to use the rules that you configure in the /etc/cgrules.conf file:

1. Install the libcgroup-pam package.

```
# yum install libcgroup-pam
```

   The pam_cgroup.so module is installed in /lib64/security on 64-bit systems, and in /lib/security on 32-bit systems.

2. Edit the `/etc/pam.d/su` configuration file, and add the following line for the `pam_cgroup.so` module:

```
session  optional  pam_cgroup.so
```

> **Note**
>
> For a service that has a configuration file in `/etc/sysconfig`, you can add the following line to the `start` section of the file to start the service in a specified cgroup:
>
> ```
> CGROUP_DAEMON="*:cgroup
> ```

7.5 Restarting the cgconfig Service

If you make any changes to the `cgroups` configuration file, `/etc/cgconfig.conf`, restart the `cgconfig` service to make it reread the file.

```
# service cgconfig restart
```

7.6 About the cgroups Configuration File

The `cgroups` configuration file, `/etc/cgconfig.conf`, contains a mount definition and one or more group definitions.

mount Definitions

A `mount` definition specifies the virtual file systems that you use to mount resource subsystems before you attach them to `cgroups`. The configuration file can contain only one `mount` definition.

The `mount` entry takes the following form:

```
mount {
    subsystem1 = /cgroup/resource_path1;
    [subsystem2 = /cgroup/resource_path2;]
.
.
.
}
```

For example, the following `mount` definition combines the `cpu`, `cpuset`, and `memory` subsystems under the `/cgroup/cpumem` subsystem hierarchy, and also creates entries for the `blkio` and `devices` subsystems under `/cgroup/iolimit` and `/cgroup/devlist`. You cannot include a subsystem in more than one subsystem hierarchy.

```
mount {
    cpu = /cgroup/cpumem;
    cpuset = /cgroup/cpumem;
    memory = /cgroup/cpumem;
    blkio = /cgroup/iolimit;
    devices = /cgroup/devlist;
}
```

group Definitions

A `group` definition specifies a `cgroup`, its access permissions, the resource subsystems that it uses, and the parameter values for those subsystems. The configuration file can contain more than one `group` definition.

A group entry takes the following form:

```
group cgroup_name {
    [perm {
        task {
            uid = task_user;
            gid = task_group;
        }
        admin {
            uid = admin_user;
            gid = admin_group;
        }
    }]
    subsystem {
        subsystem.parameter1 = value1;
        [subsystem.parameter2 = value2;]
        .
        .
        .
    }
    .
    .
    .
}
```

The *cgroup_name* argument defines the name of the cgroup. The task section of the optional perm (permissions) section defines the user and group combination that can add tasks to the cgroup. The admin section defines the user and group combination that can modify subsystem parameters and create subgroups. Whatever settings exist under perm, the root user always has permission to make any admin or task change.

One or more subsystem sections define the parameter settings for the cgroup. You can associate only one virtual subsystem hierarchy from /cgroup with a cgroup. If a several subsystems are grouped in the same hierarchy, you must include definitions for all the subsystems. For example, if the /cgroup/cpumem hierarchy includes the cpu, cpuset, and memory subsystems, you must include definitions for all of these subsystems.

For example, the following group definition defines the cgroup dbgrp for database processes, allows the oracle user to add tasks, and sets various parameters for CPU and memory usage:

```
group dbgrp {
    perm {
        task {
            uid = oracle;
            gid = dba;
        }
        admin {
            uid = root;
            gid = root;
        }
    }
    cpu {
#       Reallocate CPU resources once per second
        cpu.rt_period_us="1000000";
#       Allocate 50% of runtime to tasks in the cgroup
        cpu.rt_runtime_us="500000";
    }
    cpuset {
        cpuset.mems="0";
#       Allocate CPU cores 4 through 7 to tasks in the cgroup
        cpuset.cpus="4-7";
    }
    memory {
#       Allocate at most 4 GB of memory to tasks
```

```
        memory.limit_in_bytes="4G";
#       Allocate at most 8 GB of memory plus swap to tasks
        memory.memsw.limit_in_bytes="8G";
#       Apply a soft limit of 2 GB to tasks
        memory.soft_limit_in_bytes="2G";
    }
}
```

You can include comments in the file by preceding them with a # character, which must be at the start of a line.

7.7 About the cgroup Rules Configuration File

The cgroup rules definition file, /etc/cgrules.conf, defines the control groups to which the kernel should assign processes when they are created. Each line of the file consists of a definition in one of the following formats.

Define a cgroup and permitted subsystems for the named user. The optional *command_name* specifies the name or full pathname of a command. If you specify the subsystem as *, the user can use all subsystems that are associated with the cgroup.

```
user_name[:command_name]
                subsystem_name[,...]
                cgroup_name
```

Define a cgroup and subsystems for the named group.

```
@group_name[:command_name]
                subsystem_name[,...]
                cgroup_name
```

Define a cgroup and subsystems for the same user or group as was specified on the previous line.

```
%[:command_name]
            subsystem_name[,...]
             cgroup_name
```

Define a cgroup and subsystems for all users.

```
*[:command_name]
            subsystem_name[,...]
             cgroup_name
```

You can include comments in the file by preceding them with a # character.

The following example shows some rule definitions for users and groups:

```
# Assign tasks run by the oracle user to dbgrp
oracle    cpu,cpuset,memory    dbgrp
# Assign tasks run by the guest group to devgrp
# except for rm tasks, which are assigned to devgrp/rm
@guest    devices              devgrp
%:rm      devices              devgrp/rm
```

7.8 Displaying and Setting Subsystem Parameters

To display the value of a subsystem parameter, use the cgget command. The following example shows how to display the memory statistics for the cgroup hipri.

```
# cgget -r memory.stat hipri
        rss 168132608
```

```
        mapped_file 57577472
        .
        .
        .
```

You can use the `cgset` command to change the value of subsystem parameters for a cgroup. The next example removes input throttling from the device `/dev/sda1` for the cgroup `iocap1` by setting the value of `blkio.throttle.read_bps_device` to 0.

```
# cgset -r blkio.throttle.read_bps_device="8:1 0" iocap1
```

Any change that you make to a parameter is effective only while the `cgconfig` service continues to run. The `cgset` command does not write the new value to the configuration file, `/etc/cgconfig.conf`. You can use the `cgsnapshot` command to display the current cgroup configuration in a form that you can use as the basis for a new `/etc/cgconfig.conf` file.

```
# cgsnapshot -s > current_cgconfig.conf
```

For more information, see the `cgget(1)`, `cgset(1)`, and `cgsnapshot(1)` manual pages.

7.9 Use Cases for cgroups

The following sections describe sample `/etc/cgconfig.conf` entries for `cgroups` that can control the access that processes have to system resources.

7.9.1 Pinning Processes to CPU Cores

Define two `cgroups` that can be used to assign tasks to run on different sets of CPU cores.

```
mount {
    cpuset = /cgroup/coregrp;
}

group locores {
    cpuset {
        cpuset.mems="0";
#       Run tasks on cores 0 through 3
        cpuset.cpus="0-3";
    }
}

group hicores {
    cpuset {
        cpuset.mems="0";
#       Run tasks on cores 4 through 7
        cpuset.cpus="4-7";
    }
}
```

7.9.2 Controlling CPU and Memory Usage

Define two `cgroups` with different allocations of available CPU time and memory resources.

```
mount {
    cpu = /cgroup/cpumem;
    cpuset = /cgroup/cpumem;
    memory = /cgroup/cpumem;
}

# High priority group
group hipri {
    cpu {
```

```
#       Set the relative share of CPU resources equal to 75%
        cpu.shares="750";
    }
    cpuset {
#       No alternate memory nodes if the system is not NUMA
        cpuset.mems="0";
#       Make all CPU cores available to tasks
        cpuset.cpus="0-7";
    }
    memory {
#       Allocate at most 2 GB of memory to tasks
        memory.limit_in_bytes="2G";
#       Allocate at most 4 GB of memory+swap to tasks
        memory.memsw.limit_in_bytes="4G";
#       Apply a soft limit of 1 GB to tasks
        memory.soft_limit_in_bytes="1G";
    }
}

# Low priority group
group lopri {
    cpu {
#       Set the relative share of CPU resources equal to 25%
        cpu.shares="250";
    }
    cpuset {
#       No alternate memory nodes if the system is not NUMA
        cpuset.mems="0";
#       Make only cores 0 and 1 available to tasks
        cpuset.cpus="0,1";
    }
    memory {
#       Allocate at most 1 GB of memory to tasks
        memory.limit_in_bytes="1G";
#       Allocate at most 2 GB of memory+swap to tasks
        memory.memsw.limit_in_bytes="2G";
#       Apply a soft limit of 512 MB to tasks
        memory.soft_limit_in_bytes="512M";
    }
}
```

7.9.3 Restricting Access to Devices

Define a cgroup that denies access to the disk devices `/dev/sd[bcd]`.

```
mount {
    devices = /cgroup/devlist;
}

group blkdev {
    devices {
#       Deny access to /dev/sdb
        devices.deny="b 8:16 mrw";
#       Deny access to /dev/sdc
        devices.deny="b 8:32 mrw";
#       Deny access to /dev/sdd
        devices.deny="b 8:48 mrw";
    }
}
```

7.9.4 Throttling I/O Bandwidth

Define a cgroup that limits the I/O bandwidth to 50MB/s when reading from `/dev/sda1`.

```
mount {
```

```
    blkio = /cgroup/iolimit;
}

group iocap1 {
    blkio  {
#       Limit reads from /dev/sda1 to 50 MB/s
        blkio.throttle.read_bps_device="8:1 52428800";
    }
}
```

Define a cgroup that limits the number of read transactions to 100 per second when reading from `/dev/sdd`.

```
mount {
    blkio = /cgroup/iolimit;
}

group iocap2 {
    blkio  {
#       Limit read tps from /dev/sdd to 100 per second
        blkio.throttle.read_iops_device="8:48 100";
    }
}
```

Define two `cgroups` with different shares of I/O access to `/dev/sdb` .

```
mount {
    blkio = /cgroup/iolimit;
}

# Low access share group
group iolo {
    blkio  {
#       Set the share of I/O access by /dev/sdb to 25%
        blkio.weight_device="8:16 250";
    }
}

# High access share group
group iohi {
    blkio  {
#       Set the share of I/O access by /dev/sdb to 75%
        blkio.weight_device="8:16 750";
    }
}
```

7.10 For More Information About cgroups

You can find out more information about `cgroups` at http://www.kernel.org/doc/Documentation/cgroups/.

Chapter 8 Linux Containers

Table of Contents

This chapter describes how to use Linux Containers (LXC) to isolate applications and entire operating system images from the other processes that are running on a host system. The version of LXC described here is 1.0.7 or later running under UEK R3 QU6 or later, which provides some significant enhancements over previous versions.

Information on using the Docker engine to manage containers and images under Oracle Linux is provided in the *Oracle Linux Docker User's Guide* available at https://docs.oracle.com/cd/E37670_01/E75728/html/.

8.1 About Linux Containers

Note

Prior to UEK R3, LXC was a Technology Preview feature that was made available for testing and evaluation purposes, but was not recommended for production systems. LXC is a supported feature with UEK R3.

The Linux Containers (LXC) feature is a lightweight virtualization mechanism that does not require you to set up a virtual machine on an emulation of physical hardware. The Linux Containers feature takes the cgroups resource management facilities as its basis and adds POSIX file capabilities to implement process and network isolation. You can run a single application within a container (an *application container*) whose name space is isolated from the other processes on the system in a similar manner to a `chroot` jail. However, the main use of Linux Containers is to allow you to run a complete copy of the Linux operating system in a container (a *system container*) without the overhead of running a level-2 hypervisor such as VirtualBox. In fact, the container is sharing the kernel with the host system, so its processes and file system are completely visible from the host. When you are logged into the container, you only see its file system and process space. Because the kernel is shared, you are limited to the modules and drivers that it has loaded.

Typical use cases for Linux Containers are:

- Running Oracle Linux 5 and Oracle Linux 6 containers in parallel. Both versions of the operating system support the Unbreakable Enterprise Kernel Release 2. You can even run an Oracle Linux 5 container on an Oracle Linux 6 system with the UEK R3 kernel, even though UEK R3 is not supported for Oracle Linux 5. You can also run an i386 container on an x86_64 kernel. However, you cannot run an x86_64 container on an i386 kernel. For more information, see Section 8.1.1, “Supported Oracle Linux Container Versions”.
- Running applications that are supported only by Oracle Linux 5 in an Oracle Linux 5 container on an Oracle Linux 6 host. However, incompatibilities might exist in the modules and drivers that are available.
- Running many copies of application configurations on the same system. An example configuration would be a LAMP stack, which combines Linux, Apache server, MySQL, and Perl, PHP, or Python scripts to provide specialised web services.
- Creating sandbox environments for development and testing.
- Providing user environments whose resources can be tightly controlled, but which do not require the hardware resources of full virtualization solutions.
- Creating containers where each container appears to have its own IP address. For example you can use the `lxc-sshd` template script to create isolated environments for untrusted users. Each container runs an `sshd` daemon to handle logins. By bridging a container's Virtual Ethernet interface to the host's network interface, each container can appear to have its own IP address on a LAN.

When you use the `lxc-start` command to start a system container, by default the copy of `/sbin/init` in the container is started to spawn other processes in the container's process space. Any system calls or device access are handled by the kernel running on the host. If you need to run different kernel versions or different operating systems from the host, use a true virtualization solution such as Oracle VM or Oracle VM VirtualBox instead of Linux Containers.

There are a number of configuration steps that you need to perform on the file system image for a container so that it can run correctly:

- Disable any `init` scripts that load modules to access hardware directly.
- Disable `udev` and instead create static device nodes in `/dev` for any hardware that needs to be accessible from within the container.
- Configure the network interface so that it is bridged to the network interface of the host system.

LXC provides a number of template scripts in `/usr/share/lxc/templates` that perform much of the required configuration of system containers for you. However, it is likely that you will need to modify the script to allow the container to work correctly as the scripts cannot anticipate the idiosyncrasies of your system's configuration. You use the `lxc-create` command to create a system container by invoking a template script. For example, the `lxc-busybox` template script creates a lightweight BusyBox system container.

The example system container in this chapter uses the template script for Oracle Linux (`lxc-oracle`). The container is created on a btrfs file system (`/container`) to take advantage of its snapshot feature. A btrfs file system allows you to create a subvolume that contains the root file system (`rootfs`) of a container, and to quickly create new containers by cloning this subvolume.

You can use control groups to limit the system resources that are available to applications such as web servers or databases that are running in the container.

Application containers are not created by using template scripts. Instead, an application container mounts all or part of the host's root file system to provide access to the binaries and libraries that the application requires. You use the `lxc-execute` command to invoke `lxc-init` (a cut-down version of `/sbin/`

init) in the container. lxc-init mounts any required directories such as /proc, /dev/shm, and /dev/mqueue, executes the specified application program, and then waits for it to finish executing. When the application exits, the container instance ceases to exist.

8.1.1 Supported Oracle Linux Container Versions

The following table shows the tested and supported Oracle Linux container versions for Oracle Linux 6 hosts:

Host	Container Versions
Oracle Linux 6.5 (kernel-uek-3.8.13-16.2.1 or later)	Oracle Linux 5.9 or later Oracle Linux 6.5 or later
Oracle Linux 6.6 or later (kernel-uek-3.8.13-44.1.1 or later)	Oracle Linux 5.9 or later Oracle Linux 6.5 or later Oracle Linux 7.0 or later

Note that subsequent versions of Oracle Linux 6 and UEK are tested to support the listed container versions. Exceptions, if any, are listed in the release notes for the version of Oracle Linux 6 affected.

8.2 Configuring Operating System Containers

The procedures in the following sections describe how to set up Linux Containers that contain a copy of the root file system installed from packages at the Oracle Linux Yum Server.

- Section 8.2.1, “Installing and Configuring the Software”
- Section 8.2.2, “Setting up the File System for the Containers”
- Section 8.2.3, “Creating and Starting a Container”

Note

Throughout the following sections in this chapter, the prompts [root@host ~]# and [root@ol6ctr1 ~]# distinguish between commands run by root on the host and in the container.

The software functionality described requires that you boot the system with at least the Unbreakable Enterprise Kernel Release 2 (2.6.39).

8.2.1 Installing and Configuring the Software

To install and configure the software that is required to run Linux Containers:

1. Use yum to install the btrfs-progs package.

   ```
   [root@host ~]# yum install btrfs-progs
   ```

2. Install the lxc and wget packages.

   ```
   [root@host ~]# yum install lxc wget
   ```

   This command installs all of the required packages, such as libvirt, libcgroup, and lxc-libs. The LXC template scripts are installed in /usr/share/lxc/templates. LXC uses wget to download packages from the Oracle Linux Yum Server.

3. Start the Control Groups (cgroups) service, `cgconfig`, and configure the service to start at boot time.

```
[root@host ~]# service cgconfig start
[root@host ~]# chkconfig cgconfig on
```

   LXC uses the cgroups service to control the system resources that are available to containers.

4. Start the virtualization management service, `libvirtd`, and configure the service to start at boot time.

```
[root@host ~]# service libvirtd start
[root@host ~]# chkconfig libvirtd on
```

   LXC uses the virtualization management service to support network bridging for containers.

5. If you are going to compile applications that require the LXC header files and libraries, install the `lxc-devel` package.

```
[root@host ~]# yum install lxc-devel
```

8.2.2 Setting up the File System for the Containers

Note

The LXC template scripts assume that containers are created in `/container`. You must edit the script if your system's configuration differs from this assumption.

To set up the `/container` file system:

1. Create a btrfs file system on a suitably sized device such as `/dev/sdb`, and create the `/container` mount point.

```
[root@host ~]# mkfs.btrfs /dev/sdb
[root@host ~]# mkdir /container
```

2. Mount the `/container` file system.

```
[root@host ~]# mount /dev/sdb /container
```

3. Add an entry for `/container` to the `/etc/fstab` file.

```
/dev/sdb      /container    btrfs    defaults    0 0
```

For more information, see Chapter 4, *The Btrfs File System*.

8.2.3 Creating and Starting a Container

Note

The procedure in this section uses the LXC template script for Oracle Linux (`lxc-oracle`), which is located in `/usr/share/lxc/templates`.

An Oracle Linux container requires a minimum of 400 MB of disk space.

To create and start a container:

1. Create an Oracle Linux 6 container named `ol6ctr1` using the `lxc-oracle` template script.

```
[root@host ~]# lxc-create -n ol6ctr1 -B btrfs -t oracle -- --release=6.latest

lxc-create: No config file specified, using the default config /etc/lxc/default.conf
Host is OracleServer 6.4
Create configuration file /container/ol6ctr1/config
```

```
Downloading release 6.latest for x86_64
  .
  .
  .
  yum-metadata-parser.x86_64 0:1.1.2-16.el6
  zlib.x86_64 0:1.2.3-29.el6

Complete!
```

> **Note**
>
> For LXC version 1.0 and later, you must specify the `-B btrfs` option if you want to use the snapshot features of btrfs. For more information, see the `lxc-create(1)` manual page.

The `lxc-create` command runs the template script `lxc-oracle` to create the container in `/container/ol6ctr1` with the btrfs subvolume `/container/ol6ctr1/rootfs` as its root file system. The command then uses `yum` to install the latest available update of Oracle Linux 6 from the Oracle Linux Yum Server. It also writes the container's configuration settings to the file `/container/ol6ctr1/config` and its `fstab` file to `/container/ol6ctr1/fstab`. The default log file for the container is `/container/ol6ctr1/ol6ctr1.log`.

You can specify the following template options after the `--` option to `lxc-create`:

`-a` | `--arch=i386|x86_64`
: Specifies the architecture. The default value is the architecture of the host.

`--baseurl=`*`pkg_repo`*
: Specifies the file URI of a package repository. You must also use the `--arch` and `--release` options to specify the architecture and the release, for example:

```
# mount -o loop OracleLinux-R7-GA-Everything-x86_64-dvd.iso /mnt
# lxc-create -n ol70beta -B btrfs -t oracle -- -R 7.0 -a x86_64 \
  --baseurl=file:///mnt/Server
```

`-P` | `--patch=`*`path`*
: Patches the `rootfs` at the specified path.

`--privileged`[`=rt`]
: Allows you to adjust the values of certain kernel parameters under the `/proc` hierarchy.

  The container uses a privilege configuration file, which mounts `/proc` read-only with some exceptions. See Section 8.9, “Configuring Kernel Parameter Settings for Oracle Linux Containers”.

  This option also enables the `CAP_SYS_NICE` capability, which allows you to set negative `nice` values (that is, more favored for scheduling) for processes from within the container.

  If you specify the `=rt` (real-time) modifier, you can configure the `lxc.cgroup.cpu.rt_runtime_us` setting in the container's configuration file or when you start the container. This setting specifies the maximum continuous period in microseconds for which the container has access to CPU resources from the base period set by the system-wide value of `cpu.rt_period_us`. Otherwise, a container uses the system-wide value of `cpu.rt_runtime_us`, which might cause it to consume too many CPU resources. In addition, this modifier ensures that rebooting a container terminates all of its processes and boots it to a clean state.

`-R` | `--release=`*`major.minor`*	Specifies the major release number and minor update number of the Oracle release to install. The value of *`major`* can be set to 4, 5, 6, or 7. If you specify `latest` for *`minor`*, the latest available release packages for the major release are installed. If the host is running Oracle Linux, the default release is the same as the release installed on the host. Otherwise, the default release is the latest update of Oracle Linux 6.
`-r` | `--rpms=`*`rpm_name`*	Installs the specified RPM in the container.
`-t` | `--templatefs=`*`rootfs`*	Specifies the path to the root file system of an existing system, container, or Oracle VM template that you want to copy. Do not specify this option with any other template option. See Section 8.4, "Creating Additional Containers".
`-u` | `--url=`*`repo_URL`*	Specifies a yum repository other than Oracle Public Yum. For example, you might want to perform the installation from a local yum server. The repository file in configured in `/etc/yum.repos.d` in the container's root file system. The default URL is `https://yum.oracle.com`.

2. If you want to create additional copies of the container in its initial state, create a snapshot of the container's root file system, for example:

   ```
   # btrfs subvolume snapshot /container/ol6ctr1/rootfs /container/ol6ctr1/rootfs_snap
   ```

   See Chapter 4, *The Btrfs File System* and Section 8.4, "Creating Additional Containers".

3. Start the container `ol6ctr1` as a daemon that writes its diagnostic output to a log file other than the default log file.

   ```
   [root@host ~]# lxc-start -n ol6ctr1 -d -o /container/ol6ctr1_debug.log -l DEBUG
   ```

   Note

   If you omit the `-d` option, the container's console opens in the current shell.

   The following logging levels are available: `FATAL`, `CRIT`, `WARN`, `ERROR`, `NOTICE`, `INFO`, and `DEBUG`. You can set a logging level for all `lxc-*` commands.

   If you run the `ps -ef --forest` command on the host system and the process tree below the `lxc-start` process shows that the `/usr/sbin/sshd` and `/sbin/mingetty` processes have started in the container, you can log in to the container from the host. See Section 8.3, "Logging in to Containers".

8.2.4 About the lxc-oracle Template Script

Note

If you amend a template script, you alter the configuration files of all containers that you subsequently create from that script. If you amend the `config` file for a container, you alter the configuration of that container and all containers that you subsequently clone from it.

The `lxc-oracle` template script defines system settings and resources that are assigned to a running container, including:

- the default passwords for the `oracle` and `root` users, which are set to `oracle` and `root` respectively
- the host name (`lxc.utsname`), which is set to the name of the container
- the number of available terminals (`lxc.tty`), which is set to 4
- the location of the container's root file system on the host (`lxc.rootfs`)
- the location of the `fstab` mount configuration file (`lxc.mount`)
- all system capabilities that are not available to the container (`lxc.cap.drop`)
- the local network interface configuration (`lxc.network`)
- all whitelisted cgroup devices (`lxc.cgroup.devices.allow`)

The template script sets the virtual network type (`lxc.network.type`) and bridge (`lxc.network.link`) to `veth` and `virbr0`. If you want to use a macvlan bridge or Virtual Ethernet Port Aggregator that allows external systems to access your container via the network, you must modify the container's configuration file. See Section 8.2.5, “About Veth and Macvlan” and Section 8.2.6, “Modifying a Container to Use Macvlan”.

To enhance security, you can uncomment `lxc.cap.drop` capabilities to prevent `root` in the container from performing certain actions. For example, dropping the `sys_admin` capability prevents `root` from remounting the container's `fstab` entries as writable. However, dropping `sys_admin` also prevents the container from mounting any file system and disables the `hostname` command. By default, the template script drops the following capabilities: `mac_admin`, `mac_override`, `setfcap`, `setpcap`, `sys_module`, `sys_nice`, `sys_pacct`, `sys_rawio`, and `sys_time`.

For more information, see Chapter 7, *Control Groups* and the `capabilities(7)` and `lxc.conf(5)` manual pages.

When you create a container, the template script writes the container's configuration settings and mount configuration to `/container/name/config` and `/container/name/fstab`, and sets up the container's root file system under `/container/name/rootfs`.

Unless you specify to clone an existing root file system, the template script installs the following packages under `rootfs` (by default, from the Oracle Linux Yum Server at `https://yum.oracle.com`):

Package	**Description**
`chkconfig`	`chkconfig` utility for maintaining the `/etc/rc*.d` hierarchy.
`dhclient`	DHCP client daemon (`dhclient`) and `dhclient-script`.
`initscripts`	`/etc/inittab` file and `/etc/init.d` scripts.
`openssh-server`	Open source SSH server daemon, `/usr/sbin/sshd`.
`oraclelinux-release`	Oracle Linux 6 release and information files.
`passwd`	`passwd` utility for setting or changing passwords using PAM.
`policycoreutils`	SELinux policy core utilities.
`rootfiles`	Basic files required by the `root` user.
`rsyslog`	Enhanced system logging and kernel message trapping daemons.
`vim-minimal`	Minimal version of the VIM editor.
`yum`	`yum` utility for installing, updating and managing RPM packages.

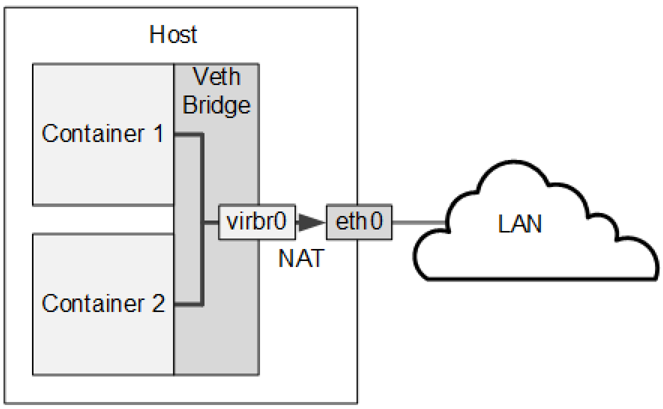
The template script edits the system configuration files under `rootfs` to set up networking in the container and to disable unnecessary services including volume management (LVM), device management (`udev`), the hardware clock, `readahead`, and the Plymouth boot system.

8.2.5 About Veth and Macvlan

By default, the `lxc-oracle` template script sets up networking by setting up a veth bridge. In this mode, a container obtains its IP address from the `dnsmasq` server that `libvirtd` runs on the private virtual bridge network (`virbr0`) between the container and the host. The host allows a container to connect to the rest of the network by using NAT rules in `iptables`, but these rules do not allow incoming connections to the container. Both the host and other containers on the veth bridge have network access to the container via the bridge.

Figure 8.1 illustrates a host system with two containers that are connected via the veth bridge `virbr0`.

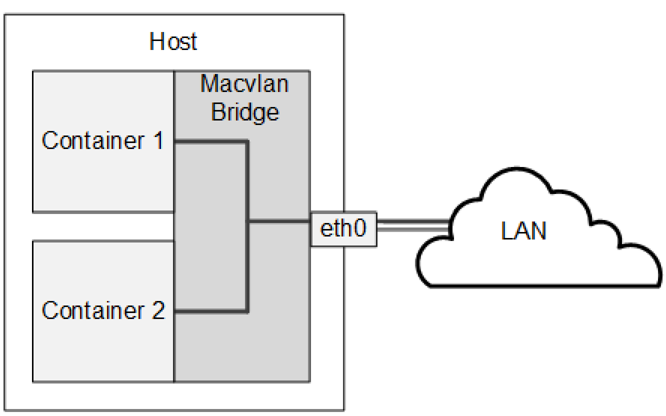
Figure 8.1 Network Configuration of Containers Using a Veth Bridge



If you want to allow network connections from outside the host to be able to connect to the container, the container needs to have an IP address on the same network as the host. One way to achieve this configuration is to use a macvlan bridge to create an independent logical network for the container. This network is effectively an extension of the local network that is connected the host's network interface. External systems can access the container as though it were an independent system on the network, and the container has network access to other containers that are configured on the bridge and to external systems. The container can also obtain its IP address from an external DHCP server on your local network. However, unlike a veth bridge, the host system does not have network access to the container.

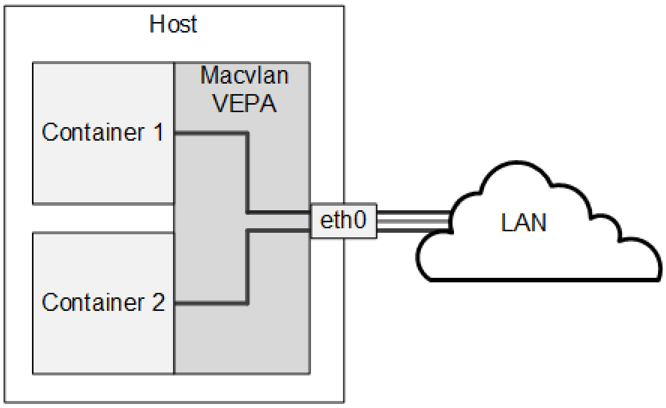
Figure 8.2 illustrates a host system with two containers that are connected via a macvlan bridge.

Figure 8.2 Network Configuration of Containers Using a Macvlan Bridge



If you do not want containers to be able to see each other on the network, you can configure the Virtual Ethernet Port Aggregator (VEPA) mode of macvlan. Figure 8.3 illustrates a host system with two containers that are separately connected to a network by a macvlan VEPA. In effect, each container is connected directly to the network, but neither container can access the other container nor the host via the network.

Figure 8.3 Network Configuration of Containers Using a Macvlan VEPA



For information about configuring macvlan, see Section 8.2.6, “Modifying a Container to Use Macvlan” and the `lxc.conf(5)` manual page.

8.2.6 Modifying a Container to Use Macvlan

To modify a container so that it uses the bridge or VEPA mode of macvlan, edit `/container/name/config` and replace the following lines:

```
lxc.network.type = veth
lxc.network.flags = up
lxc.network.link = virbr0
```

with these lines for bridge mode:

```
lxc.network.type = macvlan
lxc.network.macvlan.mode = bridge
lxc.network.flags = up
lxc.network.link = eth0
```

or these lines for VEPA mode:

```
lxc.network.type = macvlan
lxc.network.macvlan.mode = vepa
lxc.network.flags = up
lxc.network.link = eth0
```

In these sample configurations, the setting for `lxc.network.link` assumes that you want the container's network interface to be visible on the network that is accessible via the host's `eth0` interface.

8.2.6.1 Modifying a Container to Use a Static IP Address

By default, a container connected by macvlan relies on the DHCP server on your local network to obtain its IP address. If you want the container to act as a server, you would usually configure it with a static IP address. You can configure DHCP to serve a static IP address for a container or you can define the address in the container's `config` file.

To configure a static IP address that a container does not obtain using DHCP:

1. Edit `/container/name/rootfs/etc/sysconfig/network-scripts/ifcfg-iface`, where *iface* is the name of the network interface, and change the following line:

   ```
   BOOTPROTO=dhcp
   ```

   to read:

   ```
   BOOTPROTO=none
   ```

2. Add the following line to `/container/name/config`:

   ```
   lxc.network.ipv4 = xxx.xxx.xxx.xxx/prefix_length
   ```

   where *xxx.xxx.xxx.xxx/prefix_length* is the IP address of the container in CIDR format, for example: `192.168.56.100/24`.

   Note

   The address must not already be in use on the network or potentially be assignable by a DHCP server to another system.

   To configure DNS, edit the `hosts` and `resolv.conf` files under `/container/name/rootfs/etc`.

   You might also need to configure the firewall on the host to allow access to a network service that is provided by a container.

8.3 Logging in to Containers

You can use the `lxc-console` command to log in to a running container.

```
[root@host ~]# lxc-console -n name [-t tty_number]
```

If you do not specify a tty number, you log in to the first available terminal.

For example, log in to a terminal on `ol6ctr1`:

```
[root@host ~]# lxc-console -n ol6ctr1
```

To exit an `lxc-console` session, type `Ctrl-A` followed by `Q`.

Alternatively, you can use `ssh` to log in to a container if you install the `lxc-0.9.0-2.0.5` package (or later version of this package).

Note

To be able to log in using `lxc-console`, the container must be running an `/sbin/mingetty` process for the terminal. Similarly, using `ssh` requires that the container is running the SSH daemon (`/usr/sbin/sshd`).

8.4 Creating Additional Containers

To clone an existing container, use the `lxc-clone` command, as shown in this example:

```
[root@host ~]# lxc-clone -o ol6ctr1 -n ol6ctr2
```

Alternatively, you can use the `lxc-create` command to create a container by copying the root file system from an existing system, container, or Oracle VM template. Specify the path of the root file system as the argument to the `--templatefs` template option:

```
[root@host ~]# lxc-create -n ol6ctr3 -B btrfs -t oracle -- --templatefs=/container/ol6ctr1/rootfs_snap
```

This example copies the new container's `rootfs` from a snapshot of the `rootfs` that belongs to container `ol6ctr1`. The additional container is created in `/container/ol6ctr3` and a new `rootfs` snapshot is created in `/container/ol6ctr3/rootfs`.

Note

For LXC version 1.0 and later, you must specify the `-B btrfs` option if you want to use the snapshot features of btrfs. For more information, see the `lxc-create(1)` manual page.

To change the host name of the container, edit the `HOSTNAME` settings in `/container/`*`name`*`/rootfs/etc/sysconfig/network` and `/container/`*`name`*`/rootfs/etc/sysconfig/network-scripts/ifcfg-`*`iface`*, where *`iface`* is the name of the network interface, such as `eth0`.

8.5 Monitoring and Shutting Down Containers

To display the containers that are configured, use the `lxc-ls` command on the host.

```
[root@host ~]# lxc-ls
ol6ctr1
ol6ctr2
```

To display the containers that are running on the host system, specify the `--active` option.

```
[root@host ~]# lxc-ls --active
ol6ctr1
```

To display the state of a container, use the `lxc-info` command on the host.

```
[root@host ~]# lxc-info -n ol6ctr1
state:   RUNNING
pid:     10171
```

A container can be in one of the following states: `ABORTING`, `RUNNING`, `STARTING`, `STOPPED`, or `STOPPING`. Although `lxc-info` might show your container to be in the `RUNNING` state, you cannot log in to it unless the `/usr/sbin/sshd` or `/sbin/mingetty` processes have started running in the container. You must allow time for the `/sbin/init` process in the container to first start networking and the various other services that you have configured.

To view the state of the processes in the container from the host, either run `ps -ef --forest` and look for the process tree below the `lxc-start` process or use the `lxc-attach` command to run the `ps` command in the container.

```
[root@host ~]# ps -ef --forest
UID   PID   PPID  C STIME TTY    TIME     CMD
...
root  3171     1  0 09:57 ?      00:00:00 lxc-start -n ol6ctr1 -d
root  3182  3171  0 09:57 ?      00:00:00  \_ /sbin/init
root  3441  3182  0 09:57 ?      00:00:00      \_ /sbin/dhclient -H ol6ctr1 ...
root  3464  3182  0 09:57 ?      00:00:00      \_ /sbin/rsyslogd ...
root  3493  3182  0 09:57 ?      00:00:00      \_ /usr/sbin/sshd
root  3500  3182  0 09:57 pts/5  00:00:00      \_ /sbin/mingetty ... /dev/console
root  3504  3182  0 09:57 pts/1  00:00:00      \_ /sbin/mingetty ... /dev/tty1
root  3506  3182  0 09:57 pts/2  00:00:00      \_ /sbin/mingetty ... /dev/tty2
root  3508  3182  0 09:57 pts/3  00:00:00      \_ /sbin/mingetty ... /dev/tty3
root  3510  3182  0 09:57 pts/4  00:00:00      \_ /sbin/mingetty ... /dev/tty4
...
```

```
[root@host ~]# lxc-attach -n ol6ctr1 -- /bin/ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1  19284  1516 ?        Ss   04:57   0:00 /sbin/init
root       202  0.0  0.0   9172   588 ?        Ss   04:57   0:00 /sbin/dhclient
root       225  0.0  0.1 245096  1332 ?        Ssl  04:57   0:00 /sbin/rsyslogd
root       252  0.0  0.1  66660  1192 ?        Ss   04:57   0:00 /usr/sbin/sshd
root       259  0.0  0.0   4116   568 lxc/console Ss+ 04:57   0:00 /sbin/mingett
root       263  0.0  0.0   4116   572 lxc/tty1 Ss+  04:57   0:00 /sbin/mingetty
root       265  0.0  0.0   4116   568 lxc/tty2 Ss+  04:57   0:00 /sbin/mingetty
root       267  0.0  0.0   4116   572 lxc/tty3 Ss+  04:57   0:00 /sbin/mingetty
root       269  0.0  0.0   4116   568 lxc/tty4 Ss+  04:57   0:00 /sbin/mingetty
root       283  0.0  0.1 110240  1144 ?        R+   04:59   0:00 /bin/ps aux
```

Tip

If a container appears not to be starting correctly, examining its process tree from the host will often reveal where the problem might lie.

If you were logged into the container, the output from the `ps -ef` command would look similar to the following.

```
[root@ol6ctr1 ~]# ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 07:58 ?        00:00:00 /sbin/init
root       183     1  0 07:58 ?        00:00:00 /sbin/dhclient -H ol6ctr1 ...
root       206     1  0 07:58 ?        00:00:00 /sbin/rsyslogd -i ...
root       247     1  0 07:58 ?        00:00:00 /usr/sbin/sshd
root       254     1  0 07:58 lxc/console 00:00:00 /sbin/mingetty /dev/console
root       258     1  0 07:58 ?        00:00:00 login -- root
root       260     1  0 07:58 lxc/tty2 00:00:00 /sbin/mingetty /dev/tty2
root       262     1  0 07:58 lxc/tty3 00:00:00 /sbin/mingetty /dev/tty3
root       264     1  0 07:58 lxc/tty4 00:00:00 /sbin/mingetty /dev/tty4
root       268   258  0 08:04 lxc/tty1 00:00:00 -bash
root       279   268  0 08:04 lxc/tty1 00:00:00 ps -ef
```

Note that the process numbers differ from those of the same processes on the host, and that they all descend from the process 1, `/sbin/init`, in the container.

To suspend or resume the execution of a container, use the `lxc-freeze` and `lxc-unfreeze` commands on the host.

```
[root@host ~]# lxc-freeze -n ol6ctr1
[root@host ~]# lxc-unfreeze -n ol6ctr1
```

From the host, you can use the `lxc-stop` command with the `--nokill` option to shut down the container in an orderly manner.

```
[root@host ~]# lxc-stop --nokill -n ol6ctr1
```

Alternatively, you can run a command such as `halt` or `init 0` while logged in to the container.

```
[root@ol6ctr1 ~]# halt

Broadcast message from root@ol6ctr1
 (/dev/tty2) at 22:52 ...

The system is going down for halt NOW!
lxc-console: Input/output error - failed to read

[root@host ~]#
```

As shown in the example, you are returned to the shell prompt on the host.

To shut down a container by terminating its processes immediately, use `lxc-stop` with the `-k` option.

```
[root@host ~]# lxc-stop -k -n ol6ctr1
```

If you are debugging the operation of a container, using `lxc-stop` is the quickest method as you would usually destroy the container and create a new version after modifying the template script.

To monitor the state of a container, use the `lxc-monitor` command.

```
[root@host ~]# lxc-monitor -n ol6ctr1
'ol6ctr1' changed state to [STARTING]
'ol6ctr1' changed state to [RUNNING]
'ol6ctr1' changed state to [STOPPING]
'ol6ctr1' changed state to [STOPPED]
```

To wait for a container to change to a specified state, use the `lxc-wait` command.

```
lxc-wait -n $CTR -s ABORTING && lxc-wait -n $CTR -s STOPPED && \
  echo "Container $CTR terminated with an error."
```

8.6 Starting a Command Inside a Running Container

Note

The `lxc-attach` command is supported by UEK R3 with the `lxc-0.9.0-2.0.4` package or later.

You can use `lxc-attach` to execute an arbitrary command inside a container that is already running from outside the container, for example:

```
[root@host ~]# lxc-attach -n ol6ctr1 -- ps aux
```

For more information, see the `lxc-attach(1)` manual page.

8.7 Controlling Container Resources

Linux containers use cgroups in their implementation, and you can use the `lxc-cgroup` command to control the access that a container has to system resources relative to other containers. For example, to display the CPU cores to which a container can run on, enter:

```
[root@host ~]# lxc-cgroup -n ol6ctr1 cpuset.cpus
0-7
```

To restrict a container to cores 0 and 1, you would enter a command such as the following:

```
[root@host ~]# lxc-cgroup -n ol6ctr1 cpuset.cpus 0,1
```

To change a container's share of CPU time and block I/O access, you would enter:

```
[root@host ~]# lxc-cgroup -n ol6ctr2 cpu.shares 256
[root@host ~]# lxc-cgroup -n ol6ctr2 blkio.weight 500
```

Limit a container to 256 MB of memory when the system detects memory contention or low memory; otherwise, set a hard limit of 512 MB:

```
[root@host ~]# lxc-cgroup -n ol6ctr2 memory.soft_limit_in_bytes 268435456
[root@host ~]# lxc-cgroup -n ol6ctr2 memory.limit_in_bytes 53687091
```

To make the changes to a container's configuration permanent, add the settings to the file `/container/`*`name`*`/config`, for example:

```
# Permanently tweaked resource settings
```

```
lxc.cgroup.cpu.shares=256
lxc.cgroup.blkio.weight=500
```

For more information, see Chapter 7, *Control Groups*.

8.8 Configuring ulimit Settings for an Oracle Linux Container

The `ulimit` setting of an Oracle Linux container created using the `lxc-oracle` template script honors the values of `ulimit` settings such as `memlock` and `nofile` in the container's version of `/etc/security/limits.conf/` provided that these values are lower than or equal to the values on the host system.

The values of `memlock` and `nofile` determine the maximum amount of address space in kilobytes that can be locked into memory by a user process and the maximum number of file descriptors that a user process can have open at the same time.

If you require a higher `ulimit` value for a container, increase the value of the settings in `/etc/security/limits.conf` on the host, for example:

```
#<domain>      <type>  <item>         <value>
*              soft    memlock       1048576
*              hard    memlock       2097152
*              soft    nofile        5120
*              hard    nofile        10240
```

A process can use the `ulimit` built-in shell command or the `setrlimit()` system call to raise the current limit for a shell above the soft limit. However, the new value cannot exceed the hard limit unless the process is owned by `root`.

You can use `ulimit` to set or display the current soft and hard values on the host or from inside the container, for example:

```
[root@host ~]# echo "host: nofile = $(ulimit -n)"
host: nofile = 1024
[root@host ~]# echo "host: nofile = $(ulimit -H -n)"
host: nofile = 4096
[root@host ~]# ulimit -n 2048
[root@host ~]# echo "host: nofile = $(ulimit -n)"
host: nofile = 2048
[root@host ~]# lxc-attach -n ol6ctr1 -- echo "container: nofile = $(ulimit -n)"
container: nofile = 1024
```

Note

Log out and log in again or, if possible, reboot the host before starting the container in a shell that uses the new soft and hard values for `ulimit`.

8.9 Configuring Kernel Parameter Settings for Oracle Linux Containers

If you specify the `--privileged` option with the `lxc-oracle` template script, you can adjust the values of certain kernel parameters for a container under the `/proc` hierarchy.

The container mounts `/proc` read-only with the following exceptions, which are writable:

- `/proc/sys/kernel/msgmax`
- `/proc/sys/kernel/msgmnb`

- `/proc/sys/kernel/msgmni`
- `/proc/sys/kernel/sem`
- `/proc/sys/kernel/shmall`
- `/proc/sys/kernel/shmmax`
- `/proc/sys/kernel/shmmni`
- `/proc/sys/net/ipv4/conf/default/accept_source_route`
- `/proc/sys/net/ipv4/conf/default/rp_filter`
- `/proc/sys/net/ipv4/ip_forward`

Each of these parameters can have a different value than that configured for the host system and for other containers running on the host system. The default value is derived from the template when you create the container. Oracle recommends that you change a setting only if an application requires a value other than the default value.

Note

Prior to UEK R3 QU6, the following host-only parameters were not visible within the container due to kernel limitations:

- `/proc/sys/net/core/rmem_default`
- `/proc/sys/net/core/rmem_max`
- `/proc/sys/net/core/wmem_default`
- `/proc/sys/net/core/wmem_max`
- `/proc/sys/net/ipv4/ip_local_port_range`
- `/proc/sys/net/ipv4/tcp_syncookies`

With UEK R3 QU6 and later, these parameters are read-only within the container to allow Oracle Database and other applications to be installed. You can change the values of these parameters only from the host. Any changes that you make to host-only parameters apply to all containers on the host.

8.10 Deleting Containers

To delete a container and its snapshot, use the `lxc-destroy` command as shown in the following example.

```
[root@host ~]# lxc-destroy -n ol6ctr2
Delete subvolume '/container/ol6ctr2/rootfs'
```

This command also deletes the `rootfs` subvolume.

8.11 Running Application Containers

You can use the `lxc-execute` command to create a temporary application container in which you can run a command that is effectively isolated from the rest of the system. For example, the following command creates an application container named `guest` that runs `sleep` for 100 seconds.

```
[root@host ~]# lxc-execute -n guest -- sleep 100
```

While the container is active, you can monitor it by running commands such as `lxc-ls --active` and `lxc-info -n guest` from another window.

```
[root@host ~]# lxc-ls --active
guest
[root@host ~]# lxc-info -n guest
state:   RUNNING
pid:      7021
```

If you need to customize an application container, you can use a configuration file. For example, you might want to change the container's network configuration or the system directories that it mounts.

The following example shows settings from a sample configuration file where the `rootfs` is mostly not shared except for mount entries to ensure that `lxc-init` and certain library and binary directory paths are available.

```
lxc.utsname = guest
lxc.tty = 1
lxc.pts = 1
lxc.rootfs = /tmp/guest/rootfs
lxc.mount.entry=/lib /tmp/guest/rootfs/lib none ro,bind 0 0
lxc.mount.entry=/usr/libexec /tmp/guest/rootfs/usr/lib none ro,bind 0 0
lxc.mount.entry=/lib64 /tmp/guest/rootfs/lib64 none ro,bind 0 0
lxc.mount.entry=/usr/lib64 /tmp/guest/rootfs/usr/lib64 none ro,bind 0 0
lxc.mount.entry=/bin /tmp/guest/rootfs/bin none ro,bind 0 0
lxc.mount.entry=/usr/bin /tmp/guest/rootfs/usr/bin none ro,bind 0 0
lxc.cgroup.cpuset.cpus=1
```

The mount entry for `/usr/libexec` is required so that the container can access `/usr/libexec/lxc/lxc-init` on the host system.

The example configuration file mounts both `/bin` and `/usr/bin`. In practice, you should limit the host system directories that an application container mounts to only those directories that the container needs to run the application.

Note

To avoid potential conflict with system containers, do not use the `/container` directory for application containers.

You must also configure the required directories under the `rootfs` directory:

```
[root@host ~]# TMPDIR=/tmp/guest/rootfs
[root@host ~]# mkdir -p $TMPDIR/lib $TMPDIR/usr/lib $TMPDIR/lib64 $TMPDIR/usr/lib64 \
$TMPDIR/bin $TMPDIR/usr/bin $TMPDIR/dev/pts $TMPDIR/dev/shm $TMPDIR/proc
```

In this example, the directories include `/dev/pts`, `/dev/shm`, and `/proc` in addition to the mount point entries defined in the configuration file.

You can then use the `-f` option to specify the configuration file (`config`) to `lxc-execute`:

```
[root@host ~]# lxc-execute -n guest -f config -- ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
0            1     0  0 08:56 ?        00:00:00 /usr/lib/lxc/lxc-init -- ps -ef
0            2     1  0 08:56 ?        00:00:00 ps -ef
```

This example shows that the `ps` command runs as a child of `lxc-init`.

As for system containers, you can set `cgroup` entries in the configuration file and use the `lxc-cgroup` command to control the system resources to which an application container has access.

> **Note**
>
> `lxc-execute` is intended to run application containers that share the host's root file system, and not to run system containers that you create using `lxc-create`. Use `lxc-start` to run system containers.

For more information, see the `lxc-execute(1)` and `lxc.conf(5)` manual pages.

8.12 For More Information About Linux Containers

For more information about LXC, see https://wiki.archlinux.org/index.php/Linux_Containers and the LXC manual pages.

Chapter 9 HugePages

Table of Contents

This chapter describes how to set up the HugePages feature on a system that is running several Oracle Database instances.

9.1 About HugePages

The HugePages feature enables the Linux kernel to manage large pages of memory in addition to the standard 4KB (on x86 and x86_64) or 16KB (on IA64) page size. If you have a system with more than 16GB of memory running Oracle databases with a total System Global Area (SGA) larger than 8GB, you should enable the HugePages feature to improve database performance.

Note

The Automatic Memory Management (AMM) and HugePages features are not compatible in Oracle Database 11g and later. You must disable AMM to be able to use HugePages.

The memory allocated to huge pages is pinned to primary storage, and is never paged nor swapped to secondary storage. You reserve memory for huge pages during system startup, and this memory remains allocated until you change the configuration.

In a virtual memory system, the tables store the mappings between virtual addresses and physical addresses. When the system needs to access a virtual memory location, it uses the page tables to translate the virtual address to a physical address. Using huge pages means that the system needs to load fewer such mappings into the Translation Lookaside Buffer (TLB), which is the cache of page tables on a CPU that speeds up the translation of virtual addresses to physical addresses. Enabling the HugePages feature allows the kernel to use `hugetlb` entries in the TLB that point to huge pages. The `hugetbl` entries mean that the TLB entries can cover a larger address space, requiring many fewer entries to map the SGA, and releasing entries that can map other portions of the address space.

With HugePages enabled, the system uses fewer page tables, reducing the overhead for maintaining and accessing them. Huges pages remain pinned in memory and are not replaced, so the kernel swap daemon has no work to do in managing them, and the kernel does not need to perform page table lookups for them. The smaller number of pages reduces the overhead involved in performing memory operations, and also reduces the likelihood of a bottleneck when accessing page tables.

Huge pages are 4MB in size on x86, 2MB on x86_64, and 256MB on IA64.

9.2 Configuring HugePages for Oracle Database

The steps in this section are for configuring HugePages on a 64-bit Oracle Linux system running one or more Oracle Database instances.

To configure HugePages:

1. Verify that the `soft` and `hard` values in kilobytes of `memlock` that are configured in `/etc/security/limits.conf` are slightly smaller than the amount of installed memory. For example, if the system has 64GB of RAM, the values shown here would be appropriate:

```
soft memlock 60397977
hard memlock 60397977
```

2. Log in as the Oracle account owner (usually `oracle`) and use the following command to verify the value of `memlock`:

```
$ ulimit -l
60397977
```

3. If your system is running Oracle Database 11g or later, disable AMM by setting the values of both of the initialization parameters `memory_target` and `memory_max_target` to 0.

   If you start the Oracle Database instances with a server parameter file, which is the default if you created the database with the Database Configuration Assistant (DBCA), enter the following commands at the SQL prompt:

```
SQL> alter system set memory_target=0;
System altered.
SQL> alter system set memory_max_target=0;
System altered.
```

   If you start the Oracle Database instances with a text initialization parameter file, manually edit the file so that it contains the following entries:

```
memory_target = 0
memory_max_target = 0
```

4. Verify that all the Oracle Database instances are running (including any Automatic Storage Management (ASM) instances) as they would run on the production system.

5. Create the file `hugepages_settings.sh` with the following content (taken from the My Oracle Support (MOS) note 401749.1).

```
#!/bin/bash
#
# hugepages_settings.sh
#
# Linux bash script to compute values for the
# recommended HugePages/HugeTLB configuration
#
# Note: This script does calculation for all shared memory
# segments available when the script is run, no matter it
# is an Oracle RDBMS shared memory segment or not.
# Check for the kernel version
KERN=`uname -r | awk -F. '{ printf("%d.%d\n",$1,$2); }'`
# Find out the HugePage size
HPG_SZ=`grep Hugepagesize /proc/meminfo | awk {'print $2'}`
# Start from 1 pages to be on the safe side and guarantee 1 free HugePage
NUM_PG=1
# Cumulative number of pages required to handle the running shared memory segments
for SEG_BYTES in `ipcs -m | awk {'print $5'} | grep "[0-9][0-9]*"`
do
   MIN_PG=`echo "$SEG_BYTES/($HPG_SZ*1024)" | bc -q`
   if [ $MIN_PG -gt 0 ]; then
      NUM_PG=`echo "$NUM_PG+$MIN_PG+1" | bc -q`
   fi
done
# Finish with results
case $KERN in
   '2.4') HUGETLB_POOL=`echo "$NUM_PG*$HPG_SZ/1024" | bc -q`;
          echo "Recommended setting: vm.hugetlb_pool = $HUGETLB_POOL" ;;
   '2.6') echo "Recommended setting: vm.nr_hugepages = $NUM_PG" ;;
    *) echo "Unrecognized kernel version $KERN. Exiting." ;;
```

```
esac
# End
```

6. Make the file executable, and run it to calculate the recommended value for the `vm.nr_hugepages` kernel parameter.

```
$ chmod u+x ./hugepages_setting.sh
$ ./hugepages_settings.sh
.
.
.
Recommended setting: vm.nr_hugepages = 22960
```

7. As `root`, edit the file `/etc/sysctl.conf` and set the value of the `vm.nr_hugepages` parameter to the recommended value.

```
vm.nr_hugepages = 22960
```

8. Stop all the database instances and reboot the system.

After rebooting the system, verify that the database instances (including any ASM instances) have started, and use the following command to display the state of the huge pages.

```
# grep ^Huge /proc/meminfo
HugePages_Total:   22960
HugePages_Free:     2056
HugePages_Rsvd:     2016
HugePages_Surp:        0
Hugepagesize:       2048 kB
```

The value of `HugePages_Free` should be smaller than that of `HugePages_Total`, and the value of `HugePages_Rsvd` should be greater than zero. As the database instances allocate pages dynamically and proactively as required, the sum of the `Hugepages_Free` and `HugePages_Rsvd` values is likely to be smaller than the total SGA size.

If you subsequenty change the amount of system memory, add or remove any database instances, or change the size of the SGA for a database instance, use `hugepages_settings.sh` to recalculate the value of `vm.nr_hugepages`, readjust the setting in `/etc/sysctl.conf`, and reboot the system.

9.3 For More Information About HugePages

For more information about using HugePages with Oracle Database, see https://docs.oracle.com/cd/E11882_01/server.112/e10839/appi_vlm.htm#CACDCGAH.

Chapter 10 Using kexec for Fast Rebooting

Table of Contents

This chapter describes how to configure the `kexec` to enable fast rebooting of a system.

10.1 About kexec

`kexec` is a fast-boot mechanism that allows a kernel to boot from inside the context of a kernel that is already running without initializing the BIOS or firmware, performing memory and device discovery, or passing through the boot-loader stage.

When you reboot a system, the `init` process goes to run-level 6 and runs the `/etc/init.d/halt` script. If you have configured `kexec` on the system, the script will execute the `kexec -e` command, and cause the system to bypass the standard boot sequence.

The total amount of time saved when rebooting is highly dependent on your server, and can range from several tens of seconds to several minutes.

Caution

As fast reboots bypass device initialization, some devices might fail to work correctly, or a device driver might malfunction if it sees a device in an unexpected state. Before enabling this feature on your systems, test it to ensure that the hardware devices and their drivers continue to behave correctly across fast reboots.

10.2 Setting up Fast Reboots of the Current Kernel

To set up your system so that you can enable fast reboots of the current kernel:

1. Create the file `/etc/init.d/runkexec` with the following contents:

```
#!/bin/sh
#
# runkexec
#
### BEGIN INIT INFO
# Provides: runkexec
# Required-Start:
# Required-Stop:
# Default-Stop:
# Description: Enable or disable fast system rebooting
# Short-Description: enable or disable fast system rebooting
### END INIT INFO

KV=`uname -r`

case "$1" in
  start|restart|load|reload)
        kexec -l --append="`cat /proc/cmdline`" --initrd=/boot/initramfs-${KV}.img \
            /boot/vmlinuz-${KV}
```

```
        ;;
  stop|unload)
        kexec -u && echo "Target kexec kernel unloaded."
        ;;
  status)
        echo "Status not available for kexec."
        ;;
  *)
        echo "Usage: runkexec {start|restart|load|reload|stop|unload|status}"
        exit 2
esac
exit 0
```

2. Set the ownership and mode of the file.

```
# chown root:root /etc/init.d/runkexec
# chmod 755 /etc/init.d/runkexec
```

3. Create the symbolic link `S00kexec` to the file from the `/etc/rc1.d` directory.

```
# ln -s /etc/init.d/runkexec /etc/rc1.d/S00kexec
```

4. To enable fast reboots without needing to reboot the system, enter:

```
# service runkexec start
```

10.3 Controlling Fast Reboots

Once you have enabled fast reboots, running `reboot` will cause the system to shut down all services and then directly execute the kernel image.

If you want to execute the new kernel immediately without shutting down any services, use the following commands.

```
# sync; umount -a; kexec -e
```

To re-enable fast reboots of the current kernel at any time, enter:

```
# service runkexec restart
```

Alternatively, specify a different kernel that you want the system to reboot into by entering the following command:

```
# kexec -l --append="kernel_options" --initrd=initial_ramdisk_image kernel_path
```

where *`kernel_options`* are the options that you want to specify to the kernel, and *`initial_ramdisk_image`* and *`kernel_path`* are the paths to the initial ramdisk image and the kernel that you want to use.

To unload a target kernel, enter:

```
# service runkexec stop
```

Alternatively, you can enter:

```
# kexec -u
```

10.4 For More Information About kexec

For more information, see the `kexec(8)` manual page.

Chapter 11 DTrace

Table of Contents

This chapter introduces the dynamic tracing (DTrace) facility that you can use to examine the behavior of the operating system and the operating system kernel. Version 0.4 of DTrace is described, which is supported for use with UEK R3.

11.1 About DTrace

DTrace is a comprehensive dynamic tracing facility that was first developed for the Oracle Solaris operating system, and subsequently ported to Oracle Linux. DTrace allows you to explore your system to understand how it works, to track down performance problems across many layers of software, or to locate the causes of aberrant behavior.

Using DTrace, you can record data at locations of interest in the kernel, called *probes*. A probe is a location to which DTrace can bind a request to perform a set of actions, such as recording a stack trace, a timestamp, or the argument to a function. Probes function like programmable sensors that record information. When a probe is triggered, DTrace gathers data from it and reports the data back to you.

Using DTrace's D programming language, you can query the system probes to provide immediate, concise answers to arbitrary questions that you formulate.

11.2 Installing and Configuring DTrace

> **Note**
>
> The DTrace `dtrace-utils` package is available from ULN. Your system must be registered with ULN and be installed with or be updated to Oracle Linux 6 Update 4 or later.

To install and configure DTrace, perform the following steps:

1. On ULN, subscribe your system to the following channels:

   - Oracle Linux 6 Latest (x86_64) (`ol6_x86_64_latest`).
   - Unbreakable Enterprise Kernel Release 3 (UEK R3) for Oracle Linux 6 (x86_64) - Latest (`ol6_x86_64_UEKR3_latest`) or Unbreakable Enterprise Kernel Release 4 (UEK R4) for Oracle Linux 6 (x86_64) (`ol6_x86_64_UEKR4`), according to whether you boot the system with UEK R3 or UEK R4.

     Note

     UEK R4 requires Oracle Linux 6 Update 7 or later.

   - Oracle Linux 6 Dtrace Userspace Tools (x86_64) - Latest (`ol6_x86_64_Dtrace_userspace_latest`) for UEK R3 or Oracle Linux 6 Dtrace Userspace Tools (x86_64) (`ol6_x86_64_UEKR4_DTrace_userspace`) for UEK R4, as appropriate.

   Note

   Make sure that your system is *not* subscribed to the following channels:

   - Latest Unbreakable Enterprise Kernel for Oracle Linux 6 (x86_64) (`ol6_x86_64_UEK_latest`).
   - Dtrace for Oracle Linux 6 (x86_64) - Latest (`ol6_x86_64_Dtrace_latest`).
   - Dtrace for Oracle Linux 6 (x86_64) - Beta release (`ol6_x86_64_Dtrace_BETA`).
   - Unbreakable Enterprise Kernel Release 3 (3.8 based) for Oracle Linux 6 (x86_64) - Beta release (`ol6_x86_64_UEK_BETA`).

   These channels are applicable to UEK R2, DTrace for UEK R2, the beta release of DTrace for UEK R2, and the beta release of UEK R3.

2. If your system is not already running the latest version of UEK R3 or UEK R4:

   a. Use `yum` to update your system to use UEK R3 or UEK R4.

   ```
   # yum update
   ```

   b. Reboot the system, selecting the Oracle Linux Server (3.8.13 for UEK R3 or 4.1.12 for UEK R4, as appropriate) kernel in the GRUB menu if it is not the default kernel.

3. Use `yum` to install the DTrace utilities package:

   ```
   # yum install dtrace-utils
   ```

If you subsequently use `yum update` to install a new kernel, `yum` does not automatically install the matching `dtrace-modules` package that the kernel requires. If the appropriate `dtrace-modules` package for the running kernel is not present on the system, the `dtrace` command downloads and installs the package from ULN. To invoke this action without performing a trace, use a command such as the following:

```
# dtrace -l
```

Alternatively, run the following command to install the DTrace module that is appropriate to the running kernel:

```
# yum install dtrace-modules-`uname -r`
```

If you want to implement a `libdtrace` consumer or develop a DTrace provider, use `yum` to install the `dtrace-utils-devel` or `dtrace-modules-provider-headers` package respectively.

To be able to trace user-space processes that are run by users other than `root`, change the mode of the DTrace helper device as described in Section 11.2.1, “Changing the Mode of the DTrace Helper Device”.

You can find files that contain the latest information about the implementation of DTrace in `/usr/share/doc/dtrace-`*`DTrace_version`*.

11.2.1 Changing the Mode of the DTrace Helper Device

The DTrace helper device (`/dev/dtrace/helper`) allows a user-space application that contains DTrace probes to send probe provider information to DTrace.

To trace user-space processes that are run by users other than `root`, you must change the mode of the DTrace helper device to allow the user to record tracing information, for example:

```
# chmod 666 /dev/dtrace/helper
```

Alternatively, if the `acl` package is installed on your system, you can use an ACL rule to limit access to a specific user, for example:

```
# setfacl -m u:guest:rw /dev/dtrace/helper
```

Note

You must change the mode on the device before the user runs the program.

You can create a udev rules file such as `/etc/udev/rules.d/10-dtrace.rules` to change the permissions on the device file when the system starts.

To change the mode of the device file, the udev rules file should contain the following line:

```
kernel=="dtrace/helper", MODE="0666"
```

To change the ACL settings for the device file, use a line such as the following in the udev rules file:

```
kernel=="dtrace/helper", RUN="/usr/bin/setfacl -m u:guest:rw /dev/dtrace/helper"
```

To apply the udev rule without needing to restart the system, run the `start_udev` command.

11.2.2 Loading DTrace Kernel Modules

Use the `modprobe` command to load the modules that support the DTrace probes that you want to use. For example, if you wanted to use the probes that the `proc` provider publishes, you would load the `sdt` module.

```
# modprobe sdt
```

Note

The `fasttrap`, `profile`, `sdt`, and `systrace` modules automatically load the `dtrace` module, and the `dtrace` module automatically loads the `ctf` module.

To list the probes that a specific provider publishes, use the following command:

```
# dtrace -l -P provider
```

To verify that a probe is available:

```
# dtrace -l -n probe_name
```

To display the probes that are available for a specific module:

```
# dtrace -l -m module_name
```

For example, display the probes that are provided by the `libphp5.so` and `mysqld` modules for DTrace-enabled PHP and MySQL:

```
# dtrace -l -m libphp5.so -m mysqld
   ID   PROVIDER            MODULE                          FUNCTION NAME
    4    php3566    libphp5.so               dtrace_compile_file compile-file-entry
    5    php3566    libphp5.so               dtrace_compile_file compile-file-return
    6    php3566    libphp5.so                        zend_error error
    7    php3566    libphp5.so  ZEND_CATCH_SPEC_CONST_CV_HANDLER exception-caught
    8    php3566    libphp5.so     zend_throw_exception_internal exception-thrown
    9    php3566    libphp5.so                 dtrace_execute_ex execute-entry
   10    php3566    libphp5.so           dtrace_execute_internal execute-entry
   11    php3566    libphp5.so                 dtrace_execute_ex execute-return
   12    php3566    libphp5.so           dtrace_execute_internal execute-return
   13    php3566    libphp5.so                 dtrace_execute_ex function-entry
   14    php3566    libphp5.so                 dtrace_execute_ex function-return
   15    php3566    libphp5.so              php_request_shutdown request-shutdown
   16    php3566    libphp5.so               php_request_startup request-startup
...
  121  mysql3684        mysqld _Z16dispatch_command19enum_server_commandP3THDPcj
                                                                 command-done
  122  mysql3684        mysqld _Z16dispatch_command19enum_server_commandP3THDPcj
                                                                 command-start
  123  mysql3684        mysqld        _Z16close_connectionP3THDj connection-done
  124  mysql3684        mysqld   _Z22thd_prepare_connectionP3THD connection-start
  125  mysql3684        mysqld    _Z21mysql_execute_commandP3THD delete-done
  126  mysql3684        mysqld    _ZN7handler13ha_delete_rowEPKh delete-row-done
  127  mysql3684        mysqld    _ZN7handler13ha_delete_rowEPKh delete-row-start
  128  mysql3684        mysqld    _Z21mysql_execute_commandP3THD delete-start
  129  mysql3684        mysqld _Z8filesortP3THDP5TABLEP8FilesortbPyS5_
                                                                 filesort-done
  130  mysql3684        mysqld _Z8filesortP3THDP5TABLEP8FilesortbPyS5_
                                                                 filesort-start
...
```

Note

For DTrace-enabled, user-space programs, this command requires the `fasttrap` module to have been loaded before the program was started, and it does not return any probes if no instance of the program is running. `dtrace` appends the PID of the process to the DTrace provider name that was defined for the program when it was built.

11.3 Differences Between DTrace on Oracle Linux and Oracle Solaris

Note the following main differences that exist in the implementation of DTrace on Oracle Linux relative to Oracle Solaris.

- The following providers are available in the Oracle Linux implementation of DTrace.

Provider	Kernel Module	Description
`dtrace`	`dtrace`	Provides probes that relate to DTrace itself, such as `BEGIN`, `ERROR`, and `END`. You can use these probes to initialize DTrace's

Provider	Kernel Module	Description
		state before tracing begins, process its state after tracing has completed, and handle unexpected execution errors in other probes.
fasttrap	fasttrap	Supports user-space tracing of DTrace-enabled applications.
io	sdt	Provides probes that relate to data input and output. The io provider enables quick exploration of behavior observed through I/O monitoring tools such as iostat.
proc	sdt	Provides probes for monitoring process creation and termination, LWP creation and termination, execution of new programs, and signal handling.
profile	profile	Provides probes associated with an interrupt that fires at a fixed, specified time interval. These probes are associated with the asynchronous interrupt event rather than with any particular point of execution. You can use these probes to sample some aspect of a system's state.
sched	sdt	Provides probes related to CPU scheduling. Because CPUs are the one resource that all threads must consume, the sched provider is very useful for understanding systemic behavior.
syscall	systrace	Provides probes at the entry to and return from every system call. Because system calls are the primary interface between user-level applications and the operating system kernel, these probes can offer you an insight into the interaction between applications and the system.

Other providers, such as the `pid` provider, the Function Boundary Tracing (`fbt`) provider, and the providers for the network protocols (`ip`, `iscsi`, `nfsv3`, `nfsv4`, `srp`, `tcp`, and `udp`), have not yet been implemented.

- Solaris-specific features such as projects, zones, tasks, contracts, and message queues are not supported.
- The names of kernel probes are specific to the Linux kernel.
- The `-Xa`, `-Xc`, and `-Xt` options to `dtrace` all include the option `-std=gnu99` (conformance with 1999 C standard including GNU extensions) when invoking the C preprocessor (`cpp`) on D programs. The `-Xs` option includes the option `-traditional-cpp` (conformance with K&R C).
- Anonymous tracing is not supported (`-a` and `-A` options to `dtrace`).
- The 32-bit data model is not supported (`-32` option to `dtrace`).
- Various definitions in the `<dtrace.h>` header file for flags, types, structures, and function prototypes reflect intrinsic differences between the implementation of Oracle Solaris and Oracle Linux.
- SDT probes do not work in IRQ context. As a result, the `proc:::signal-discard` probe does not fire if a signal that is sent as event notification for a POSIX timer expiration should be discarded.

See the `INCOMPATIBILITIES` file in `/usr/share/doc/dtrace-`*`DTrace_version`* for more information.

11.4 Calling DTrace from the Command Line

The `dtrace` command accepts the following options:

```
dtrace [-CeFGhHlqSvVwZ]
[-b bufsz] [-c command] [-D name[=value]] [-I pathname] [-L pathname]
[-o pathname] [-p PID] [-s source_pathname]
[-U name] [-x option[=value]][-X[a|c|s|t]]
[-P provider[[predicate]action]]
[-m [[provider:]module[[predicate]action]]]
[-f [[provider:]module:]function[[predicate]action]]
[-n [[[provider:]module:]function:]name[[predicate]action]]
[-i probe-id[[predicate]action]]
```

where *predicate* is any D predicate enclosed in slashes `//` and *action* is any D statement list enclosed in braces `{}` according to the D language syntax. If D program code is provided as an argument to the `-P`, `-m`, `-f`, `-n`, or `-i` options. this text must be appropriately quoted to avoid interpretation by the shell.

The options are as follows:

`-b bufsize`	Set the principal trace buffer size, which can include any of the size suffixes `k` (kilobyte), `m` (megabyte), `g` (gigabyte), or `t` (terabyte). If the buffer space cannot be allocated, `dtrace` attempts to reduce the buffer size or exit depending on the setting of the `bufresize` property.
`-c command`	Run the specified command and exit upon its completion. If you specify more than one `-c` option, `dtrace` exits when all the commands have exited, and reports the exit status for each child process as it terminates. `dtrace` makes the process ID of the first command available to D programs as the `$target` macro variable.
`-C`	Run the C preprocessor (`cpp`) on D programs before compiling them. You can pass options to the C preprocessor by using the `-D`, `-H`, `-I`, and `-U` options. You can use the `-X` option to select the degree of conformance with the C standard.
`-D name[=value]`	Define the specified macro name and optional value when invoking `cpp` using the `-C` option. You can specify the `-D` option multiple times to the command.
`-e`	Exit after compiling any requests and before enabling any probes. You can combine this option with the `-D` option to verify that your D programs compile without executing them or enabling the corresponding instrumentation.
`-f [[[provider]:][module]:]function['D-probe_clause']`	Specify a function (optionally specifying the provider and module) that you want to trace or list. You can append an optional D-probe clause. You can specify the `-f` option multiple times to the command.
`-F`	Reduce trace output by combining the output for function and system call entry and return points. `dtrace` indents entry probe reports and leaves return probe reports unindented. `dtrace` prefixes the output from function entry probe reports with `->` and the output from function return probe reports with `<-`. `dtrace` prefixes the output from system call entry probe reports with `=>` and the output from system call return probe reports with `<=`.
`-G`	Generate an ELF file that contains an embedded D program. `dtrace` saves the DTrace probes that are specified in the program using a relocatable ELF

	object that can be linked with another program. If you specify the `-o` option, `dtrace` saves the ELF file to the specified path name. If you do not specify the `-o` option, the ELF file is given the same name as the source file for the D program, except with a `.o` extension instead of `.s`. Otherwise, the ELF file is saved with the name `d.out`.
`-h`	Create a header file based on probe definitions in the file that is specified as the argument to the `-s` option. If you specify the `-o` option, `dtrace` saves the header file to the specified path name. If you do not specify the `-o` option, the header file is given the same name as the source file for the D program, except with a `.h` extension instead of `.d`. You should amend the source file of the program to be traced so that it includes this header file.
`-H`	Print the path names of included files on `stderr` when you invoke `cpp` using the `-C` option.
`-i` *`probe_ID`*`['`*`D-probe_clause`*`']`	Specify a probe identifier that you want to trace or list. You must specify the probe ID as a decimal integer (as displayed by `dtrace -l`). You can append an optional D-probe clause. You can specify the `-i` option multiple times to the command.
`-I` *`pathname`*	Add the specified directory path to the search path for `#include` files when you invoke `cpp` using the `-C` option. The specified directory is inserted at the head of the default directory list.
`-l`	List probes instead of enabling them. `dtrace` filters the list of probes based on the arguments to the `-f`, `-i`, `-m`, `-n`, `-P`, and `-s` options. If no options are specified, `dtrace` lists all probes.
`-L` *`pathname`*	Add the specified directory path to the end of the library search path. Use this option to specify the path to DTrace libraries, which contain common definitions for D programs.
`-m [`*`provider`*`:]`*`module`*`['`*`D-probe_clause`*`']`	Specify a module (optionally specifying the provider) that you want to trace or list. You can append an optional D-probe clause. You can specify the `-m` option multiple times to the command.
`-n [[[`*`provider`*`]:] [`*`module`*`]:] [`*`function`*`]]probe['`*`D-probe_clause`*`']`	Specify a probe name (optionally specifying the provider, module, and function) that you want to trace or list. You can append an optional D-probe clause. You can specify the `-n` option multiple times to the command.
`-o` *`pathname`*	Specify the output file for the `-G` and `-l` options, or for traced data.
`-p` *`PID`*	Grab a process specified by its process ID, cache its symbol tables, and exit upon its completion. If you specify more than one `-p` option, `dtrace` exits when all the processes have exited, and reports the exit status for each process as it terminates. `dtrace` makes the first process ID specified available to D programs as the *`$target`* macro variable.
`-P` *`provider`*`['`*`D-probe_clause`*`']`	Specify a provider that you want to trace or list. You can append an optional D-probe clause. You can specify the `-P` option multiple times to the command.
`-q`	Set quiet mode. `dtrace` suppresses informational messages, column headers, the CPU ID, the probe ID, and

	additional newlines. Only data that is traced and formatted by the `printa()`, `printf()`, and `trace()` D program statements is displayed on `stdout`. This option is equivalent to specifying `#pragma D option quiet` in a D program.
`-s` *`source_pathname`*	Specifies a D program source file to be compiled by `dtrace`. If you specify the `-h` option, `dtrace` creates a header file using the probe definitions in the file. If you specify the `-G` option, `dtrace` generates a relocatable ELF object that can be linked with another program. If you specify the `-e` option, `dtrace` compiles the program, but it does not enable any instrumentation. If you specify the `-l` option, `dtrace` compiles the program and lists the set of matching probes, but it does not enable any instrumentation. If you do not specify an option, `dtrace` enables the instrumentation specified by the D program and begins tracing.
`-S`	Show the D compiler intermediate code. The D compiler writes a report of the intermediate code that it generated for each D program to `stderr`.
`-U` *`name`*	Undefine the specified name when invoking `cpp` using the `-C` option. You can specify the `-U` option multiple times to the command.
`-v`	Set verbose mode. `dtrace` produces a program stability report showing the minimum interface stability and dependency level for any specified D programs.
`-V`	Write the highest D programming interface version supported by `dtrace` to `stdout`.
`-w`	Permit destructive actions by D programs. If you do not specify this option, `dtrace` does not compile or enable a D program that contains destructive actions. This option is equivalent to specifying `#pragma D option destructive` in a D program.
`-x` *`option`*`[=`*`value`*`]`	Enable or modify a DTrace runtime option or D compiler option.
`-X[a|c|t]`	Include the option `-std=gnu99` (conformance with 1999 C standard including GNU extensions) when invoking `cpp` using the `-C` option.
`-Xs`	Include the option `-traditional-cpp` (conformance with K&R C) when invoking `cpp` using the `-C` option.
`-Z`	Permit probe descriptions that do not match any probes. If you do not specify this option, `dtrace` reports an error and exits if a probe description does not match a known probe.

11.5 About Programming for DTrace

When you use the `dtrace` command, you invoke the compiler for the D language. Once DTrace has compiled your program, it sends it to the operating system kernel for execution, where it activates the probes that your program uses.

DTrace enables probes only when you are using them. No instrumented code is present for inactive probes, so your system does not experience performance degradation when you are not using DTrace. Once your D program exits, all of the probes it used are automatically disabled and their instrumentation is removed, returning your system to its original state. No effective difference exists between a system where DTrace is not active and one where the DTrace software is not installed.

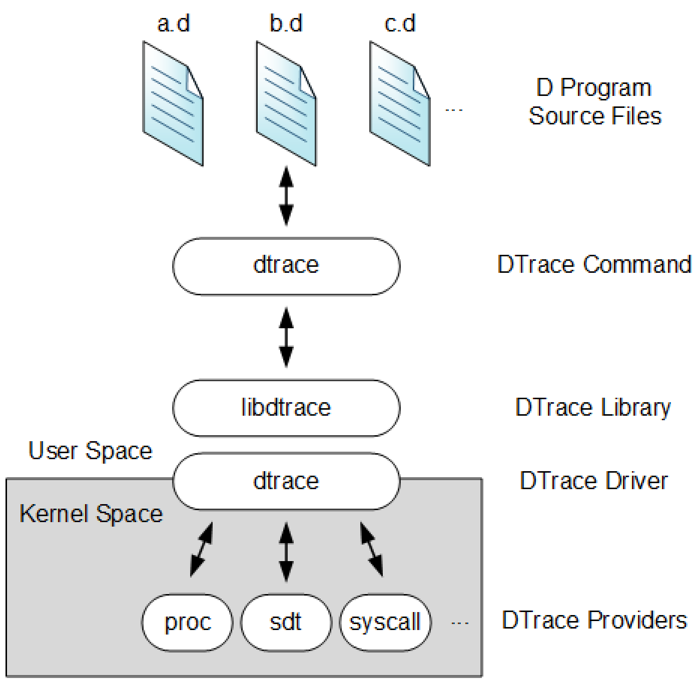
DTrace implements the instrumentation for each probe dynamically on the live, running operating system. DTrace neither quiesces nor pauses the system in any way, and it adds instrumentation code only for the probes that you enable. As a result, the effect of using DTrace probes is limited to exactly what you ask DTrace to do. DTrace instrumentation is designed to be as efficient as possible, and enables you to use it in production to solve real problems in real time.

The DTrace framework provides support for an arbitrary number of virtual clients. You can run as many simultaneous D programs as you like, limited only by your system's memory capacity, and all the programs operate independently using the same underlying instrumentation. This same capability also permits any number of distinct users on the system to take advantage of DTrace simultaneously on the same system without interfering with one another.

Unlike a C or C++ program, but similar to a Java program, DTrace compiles your D program into a safe intermediate form that it executes when a probe fires. DTrace validates whether this intermediate form can run safely, reporting any run-time errors that might occur during the execution of your D program, such as dividing by zero or dereferencing invalid memory. As a result, you cannot construct an unsafe D program. You can use DTrace in a production environment without worrying about crashing or corrupting your system. If you make a programming mistake, DTrace disables the instrumentation and reports the error to you.

Figure 11.1 illustrates the different components of the DTrace architecture, including probe providers, the DTrace driver, the DTrace library, and the `dtrace` command.

Figure 11.1 Components of the DTrace Architecture



11.6 Introducing the D Programming Language

D programs describe the probes that are to be enabled together with the predicates and actions that are bound to the probes. D programs can also declare variables and define new types. This section provides an introduction to the important features that you are likely to encounter in simple D programs.

11.6.1 Probe Clauses

D programs consist of a set of one or more probe clauses. Each probe clause takes the general form shown here:

```
probe_description_1 [, probe_description_2]...
[/ predicate_statement /]
{
  [action_statement;]
  .
  .
  .
}
```

Every probe clause begins with a list of one or more probe descriptions in this form:

```
provider:module:function:probe_name
```

where the fields are as follows:

provider
: The name of the DTrace provider that is publishing this probe. For kernel probes, the provider name typically corresponds to the name of the DTrace kernel module that performs the instrumentation to enable the probe, for example, `proc`. When tracing a DTrace-enabled, user-space application or library, this field takes the form *namePID*, where *name* is the name of the provider as defined in the provider definition file that was used to build the application or library and *PID* is the process ID of the running executable.

module
: The name of the kernel module, library, or user-space program in which the probe is located, if any, for example, `vmlinux`. This module is not the same as the kernel module that implements a provider.

function
: The name of the function in which the probe is located, for example, `do_fork`.

probe_name
: The name of the probe usually describes its location within a function, for example, `create`, `entry`, or `return`.

The compiler interprets the fields from right to left. For example, the probe description `settimeofday:entry` would match a probe with function `settimeofday` and name `entry` regardless of the value of the probe's provider and module fields. You can regard a probe description as a pattern that matches one or more probes based on their names. You can omit the leading colons before a probe name if the probe that you want to use has a unique name. If several providers publish probes with the same name, use the available fields to obtain the correct probe. If you do not specify a provider, you might obtain unexpected results if multiple probes have the same name. Specifying a provider but leaving the module, function, and probe name fields blank, matches all probes in a provider. For example, `syscall:::` matches every probe published by the `syscall` provider.

The optional predicate statement uses criteria such as process ID, command name, or timestamp to determine whether the associated actions should take place. If you omit the predicate, any associated actions always run if the probe is triggered.

You can use the ?, *, and [] shell wildcards with probe clauses. For example, syscall::[gs]et*: matches all syscall probes for function names that begin with get or set. If necessary, use the \ character to escape wildcard characters that form part of a name.

You can enable the same actions for more than one probe description. For example, the following D program uses the trace() function to record a timestamp each time that any process invokes a system call containing the string mem or soc:

```
syscall::*mem*:entry, syscall::*soc*:entry
{
        trace(timestamp);
}
```

By default, the trace() function writes the result to the principal buffer, which is accessible by other probe clauses within a D program, and whose contents dtrace displays when the program exits.

11.6.2 Pragmas

You can use compiler directives called pragmas in a D program. Pragma lines begin with a # character, and are usually placed at the beginning of a D program. The primary use of pragmas is to set run-time DTrace options. For example, the following pragma statements suppress all output except for traced data and permit destructive operations.

```
#pragma D option quiet
#pragma D option destructive
```

11.6.3 Global Variables

D provides fundamental data types for integers and floating-point constants. You can perform arithmetic only on integers in D programs. D does not support floating-point operations. D provides floating-point types for compatibility with ANSI-C declarations and types. You can trace floating-point data objects and use the printf() function to format them for output. In the current implementation, DTrace supports only the 64-bit data model for writing D programs.

You can use declarations to introduce D variables and external C symbols, or to define new types for use in D. The following example program, tick.d, declares and initializes the variable i when the D program starts, displays its initial value, increments the variable and prints its value once every second, and displays the final value when the program exits.

```
BEGIN
{
  i = 0;
  trace(i);
}

profile:::tick-1sec
{
  printf("i=%d\n",++i);
}

END
{
  trace(i);
}
```

When run, the program produces output such as the following until you type Ctrl-C:

```
# dtrace -s tick.d
dtrace: script 'tick.d' matched 3 probes
```

```
CPU     ID                    FUNCTION:NAME
  1      1                           :BEGIN       0
  1    618                           :tick-1sec i=1

  1    618                           :tick-1sec i=2

  1    618                           :tick-1sec i=3

  1    618                           :tick-1sec i=4

  1    618                           :tick-1sec i=5

^C
  0      2                           :END         5
```

Whenever a probe is triggered, `dtrace` displays the number of the CPU core on which the process indicated by its ID is running, and the name of the function and the probe. `BEGIN` and `END` are DTrace probes that trigger when the `dtrace` program starts and finishes.

To suppress all output except that from `printa()`, `printf()`, and `trace()`, specify `#pragma D option quiet` in the program or the `-q` option to `dtrace`.

```
# dtrace -q -s tick.d
0i=1
i=2
i=3
i=4
i=5
^C
5
```

11.6.4 Predicates

Predicates are logic statements that select whether DTrace invokes the actions that are associated with a probe. For example, the predicates in the following program `sc1000.d` examine the value of the variable `i`. This program also demonstrates how to include C-style comments.

```
#pragma D option quiet

BEGIN
{
  /* Initialize i */
  i = 1000;
}

syscall:::entry
/i > 0/
{
  /* Decrement i */
  i--;
}

syscall:::entry
/(i % 100) == 0/
{
  /* Print i after every 100 system calls */
  printf("i = %d\n",i);
}

syscall:::entry
/i == 0/
{
  printf("i = 0; 1000 system calls invoked\n");
  exit(0);  /* Exit with a value of 0 */
```

```
}
```

The program initializes `i` with a value of 1000, decrements its value by 1 whenever a process invokes a system call, prints its value after every 100 system calls, and exits when the value of 1 reaches 0. Running the program in quite mode produces output similar to the following:

```
# dtrace -s sc1000.d
i = 900
i = 700
i = 800
i = 600
i = 500
i = 400
i = 300
i = 200
i = 100
i = 0
i = 0; 1000 system calls invoked
```

Note that the order of the countdown sequence is not as expected. The output for `i=800` appears after the output for `i=700`. If you turn off quiet mode, it becomes apparent that the reason is that `dtrace` is collecting information from probes that can be triggered on all the CPU cores. You cannot expect runtime output from DTrace to be sequential in a multithreaded environment.

```
# dtrace -s sc1000.d
dtrace: script 'sc1000.d' matched 889 probes
CPU     ID                FUNCTION:NAME
  0    457           clock_gettime:entry i = 900

  0    413                   futex:entry i = 700

  1     41                   lseek:entry i = 800

  1     25                    read:entry i = 600

  1     25                    read:entry i = 500

  1     25                    read:entry i = 400

  1     71                  select:entry i = 300

  1     71                  select:entry i = 200

  1     25                    read:entry i = 100

  1     25                    read:entry i = 0

  1     25                    read:entry i = 0; 1000 system calls invoked
```

The next example is an executable DTrace script that displays the file descriptor, output string, and string length specified to the `write()` system call whenever the `date` command is run on the system.

```
#!/usr/sbin/dtrace -s
#pragma D option quiet

syscall::write:entry
/execname == "date"/
{
  printf("%s(%d, %s, %4d)\n", probefunc, arg0, copyinstr(arg1), arg2);
}
```

If you run the script from one window, while typing the `date` command in another, you see output such as the following in the first window:

```
write(1, Wed Aug 15 10:42:34 BST 2012
```

```
,   29)
```

11.6.5 Scalar Arrays and Associative Arrays

The D language supports *scalar arrays*, which correspond directly in concept and syntax with arrays in C. A scalar array is a fixed-length group of consecutive memory locations that each store a value of the same type. You access scalar arrays by referring to each location with an integer starting from zero. In D programs, you would usually use scalar arrays to access array data within the operating system.

For example, you would use the following statement to declare a scalar array `sa` of 5 integers:

```
int sa[5];
```

As in C, `sa[0]` refers to the first array element, `sa[1]` refers to the second, and so on up to `sa[4]` for the fifth element.

The D language also supports a special kind of variable called an *associative array*. An associative array is similar to a scalar array in that it associates a set of keys with a set of values, but in an associative array the keys are not limited to integers of a fixed range. In the D language, you can index associative arrays by a list of one or more values of any type. Together the individual key values form a *tuple* that you use to index into the array and access or modify the value that corresponds to that key. Each tuple key must be of the same length and must have the same key types in the same order. The value associated with each element of an associative array is also of a single fixed type for the entire array.

For example, the following statement defines a new associative array `aa` of value type `int` with the tuple signature `string, int`, and stores the integer value 828 in the array:

```
aa["foo", 271] = 828;
```

Once you have defined an array, you can access its elements in the same way as any other variable. For example, the following statement modifies the array element previously stored in a by incrementing the value from 828 to 829:

```
a["foo", 271]++;
```

You can define additional elements for the array by specifying a different tuple with the same tuple signature, as shown here:

```
aa["bar", 314] = 159;
aa["foo", 577] = 216;
```

The array elements `aa["foo", 271]` and `aa["foo", 577]` are distinct because the values of their tuples differ in the value of their second key.

Syntactically, scalar arrays and associative arrays are very similar. You can declare an associative array of integers referenced by an integer key as follows:

```
int ai[int];
```

You could reference an element of this array using the expression such as `ai[0]`. However, from a storage and implementation perspective, the two kinds of array are very different. The scalar array `sa` consists of five consecutive memory locations numbered from zero, and the index refers to an offset in the storage allocated for the array. An associative array such as `ai` has no predefined size and it does not store elements in consecutive memory locations. In addition, associative array keys have no relationship to the storage location of the corresponding value. If you access the associative array elements `a[0]` and `a[-5]`, DTrace allocates only two words of storage, which are not necessarily consecutive in memory. The tuple keys that you use to index associative arrays are abstract names for the corresponding value, and they bear no relationship to the location of the value in memory.

If you create an array using an initial assignment and use a single integer expression as the array index, for example, `a[0] = 2;`, the D compiler always creates a new associative array, even though `a` could also be interpreted as an assignment to a scalar array. If you want to use a scalar array, you must explicitly declare its type and size.

11.6.6 Pointers and External Variables

The implementation of pointers in the D language gives you the ability to create and manipulate the memory addresses of data objects in the operating system kernel, and to store the contents of those data objects in variables and associative arrays. The syntax of D pointers is the same as the syntax of pointers in ANSI-C. For example, the following statement declares a D global variable named `p` that is a pointer to an integer.

```
int *p;
```

This declaration means that `p` itself is a 64-bit integer whose value is the address in memory of another integer.

If you want to create a pointer to a data object inside the kernel, you can compute its address by using the `&` reference operator. For example, the kernel source code declares an `unsigned long max_pfn` variable. You can access the value of such an *external variable* in the D language by prefixing it with the `` ` `` (backquote) scope operator:

```
value = `max_pfn;
```

If more than one kernel module declares a variable with the same name, prefix the scoped external variable with the name of the module. For example, ``foo`bar`` would refer to the address of the `bar()` function provided by the module `foo`.

You can extract the address of an external variable by applying the `&` operator and store it as a pointer:

```
p = &`max_pfn;
```

You can use the `*` dereference operator to refer to the object that a pointer addresses:

```
value = *p;
```

You cannot apply the `&` operator to DTrace objects such as associative arrays, built-in functions, and variables. If you create composite structures, it is possible to construct expressions that retrieve the kernel addresses of DTrace objects. However, DTrace does not guarantee to preserve the addresses of such objects across probe firings.

You cannot use the `*` dereference operator on the left-hand side of an assignment expression. You may only assign values directly to D variables by name or by applying the array index operator `[]` to a scalar array or an associative array.

You cannot use pointers to perform indirect function calls. You may only call DTrace functions directly by name.

11.6.7 Address Spaces

DTrace executes D programs within the address space of the operating system kernel. Your entire Oracle Linux system manages one address space for the operating system kernel, and one for each user process. As each address space provides the illusion that it can access all of the memory on the system, the same virtual address might be used in different address spaces, but it would translate to different locations in physical memory. If your D programs use pointers, you need to be aware which address space corresponds to those pointers.

For example, if you use the syscall provider to instrument entry to a system call such as pipe() that takes a pointer to an integer or to an array of integers as an argument, it is not valid to use the * or [] operators to dereference that pointer or array. The address is in the address space of the user process that performed the system call, and not in the address space of the kernel. Dereferencing the address in D accesses the kernel's address space, which would result in an invalid address error or return unexpected data to your D program.

To access user process memory from a DTrace probe, use one of the copyin(), copyinstr(), or copyinto() functions with an address in user space.

The following D programs show two alternate and equivalent ways to print the file descriptor, string, and string length arguments that a process passed to the write() system call:

```
syscall::write:entry
{
  printf("fd=%d buf=%s count=%d", arg0, stringof(copyin(arg1, arg2)), arg2);
}

syscall::write:entry
{
  printf("fd=%d buf=%s count=%d", arg0, copyinstr(arg1, arg2), arg2);
}
```

The arg0, arg1 and arg2 variables contain the value of the fd, buf, and count arguments to the system call. Note that the value of arg1 is an address in the address space of the process, and not in the address space of the kernel.

In this example, it is necessary to use the stringof() function with copyin() so that DTrace converts the retrieved user data to a string. The copyinstr() function always returns a string.

To avoid confusion, you should name and comment variables that store user addresses appropriately. You should also store user addresses as variables of type uintptr_t so that you do not accidentally compile D code that dereferences them.

11.6.8 Thread-local Variables

Thread-local variables are defined within the scope of execution of a thread on the system. To indicate that a variable is thread-local, you prefix it with self-> as shown in the following example.

```
#pragma D option quiet

syscall::read:entry
{
  self->t = timestamp; /* Initialize a thread-local variable */
}

syscall::read:return
/self->t != 0/
{
  printf("%s (pid:tid=%d:%d) spent %d microseconds in read()\n",
  execname, pid, tid, ((timestamp - self->t)/1000)); /* Divide by 1000 -> microseconds */

  self->t = 0; /* Reset the variable */
}
```

This D program (dtrace.d) displays the command name, process ID, thread ID, and expired time in microseconds whenever a process invokes the read() system call.

```
# dtrace -s readtrace.d
nome-terminal (pid:tid=2774:2774) spent 27 microseconds in read()
```

```
gnome-terminal (pid:tid=2774:2774) spent 16 microseconds in read()
hald-addon-inpu (pid:tid=1662:1662) spent 26 microseconds in read()
hald-addon-inpu (pid:tid=1662:1662) spent 17 microseconds in read()
Xorg (pid:tid=2046:2046) spent 18 microseconds in read()
...
```

11.6.9 Speculations

The speculative tracing facility in DTrace allows you to tentatively trace data and then later decide whether to commit the data to a tracing buffer or discard the data. Predicates are the primary mechanism for filtering out uninteresting events. Predicates are useful when you know at the time that a probe fires whether or not the probe event is of interest. However, in some situations, you might not know whether a probe event is of interest until after the probe fires.

For example, if a system call is occasionally failing with an error code in `errno`, you might want to examine the code path leading to the error condition. You can write trace data at one or more probe locations to speculative buffers, and then choose which data to commit to the principal buffer at another probe location. As a result, your trace data contains only the output of interest, no post-processing is required, and the DTrace overhead is minimized.

To create a speculative buffer, use the `speculation()` function. This function returns a speculation identifier, which you use in subsequent calls to the `speculate()` function.

Call the `speculate()` function before performing any data-recording actions in a clause. DTrace directs all subsequent data that you record in a clause to the speculative buffer. You can create only one speculation in any given clause.

Typically, you assign a speculation identifier to a thread-local variable, and then use that variable as a predicate to other probes as well as an argument to `speculate()`. For example:

```
#!/usr/sbin/dtrace -Fs

syscall::open:entry
{
  /*
   * The call to speculation() creates a new speculation. If this fails,
   * dtrace will generate an error message indicating the reason for
   * the failed speculation(), but subsequent speculative tracing will be
   * silently discarded.
   */
  self->spec = speculation();
  speculate(self->spec);

  /*
   * Because this printf() follows the speculate(), it is being
   * speculatively traced; it will only appear in the data buffer if the
   * speculation is subsequently commited.
   */
  printf("%s", copyinstr(arg0));
}

syscall::open:return
/self->spec/
{
  /*
   * To balance the output with the -F option, we want to be sure that
   * every entry has a matching return. Because we speculated the
   * open entry above, we want to also speculate the open return.
   * This is also a convenient time to trace the errno value.
   */
  speculate(self->spec);
  trace(errno);
```

```
}
```

If a speculative buffer contains data that you want to retain, use the `commit()` function to copy its contents to the principal buffer. If you want to delete the contents of a speculative buffer, use the `discard()` function. The following example clauses commit or discard the speculative buffer based on the value of the `errno` variable:

```
syscall::open:return
/self->spec && errno != 0/
{
  /*
   * If errno is non-zero, we want to commit the speculation.
   */
  commit(self->spec);
  self->spec = 0;
}

syscall::open:return
/self->spec && errno == 0/
{
  /*
   * If errno is not set, we discard the speculation.
   */
  discard(self->spec);
  self->spec = 0;
}
```

Running this script produces output similar to the following example when the `open()` system call fails:

```
# ./specopen.d
dtrace: script './specopen.d' matched 4 probes
CPU FUNCTION
  1  => open                              /var/ld/ld.config
  1  <= open                                      2
  1  => open                              /images/UnorderedList16.gif
  1  <= open                                      4
...
```

11.6.10 Aggregations

DTrace provides the following built-in functions for aggregating the data that individual probes gather.

Aggregating Function	Description
`avg(scalar_expression)`	Returns the arithmetic mean of the expressions that are specified as arguments.
`count()`	Returns the number of times that the function has been called.
`lquantize(scalar_expression, lower_bound, upper_bound, step_interval)`	Returns a linear frequency distribution of the expressions that are specified as arguments, scaled to the specified lower bound, upper bound, and step interval. Increments the value in the highest bucket that is smaller than the specified expression.
`max(scalar_expression)`	Returns the maximum value of the expressions that are specified as arguments.
`min(scalar_expression)`	Returns the minimum value of the expressions that are specified as arguments.
`quantize(scalar_expression)`	Returns a power-of-two frequency distribution of the expressions that are specified as arguments. Increments

Aggregating Function	Description
	the value of the highest power-of-two bucket that is smaller than the specified expression.
`stddev(scalar_expression)`	Returns the standard deviation of the expressions that are specified as arguments.
`sum(scalar_expression)`	Returns the sum of the expressions that are specified as arguments.

DTrace indexes the results of an aggregation using a tuple expression similar to that used for an associative array:

```
@name[list_of_keys] = aggregating_function(args);
```

The name of the aggregation is prefixed with an `@` character. All aggregations are global. If you do not specify a name, the aggregation is anonymous. The keys describe the data that the aggregating function is collecting.

For example, the following command counts the number of `write()` system calls invoked by processes until you type Ctrl-C.

```
# dtrace -n syscall::write:entry'{ @["write() calls"] = count(); }'
dtrace: description 'syscall:::' matched 1 probe
^C

  write() calls                                             9
```

The next example counts the number of both `read()` and `write()` system calls:

```
# dtrace -n syscall::write:entry,syscall::read:entry\
'{ @[strjoin(probefunc,"() calls")] = count(); }'
dtrace: description 'syscall::write:entry,syscall::read:entry' matched 2 probes
^C

  write() calls                                           150
  read() calls                                           1555
```

Note

If you specify the `-q` option to `dtrace` or `#pragma D option quiet` in a D program, DTrace suppresses the automatic printing of aggregations. In this case, you must use a `printa()` statement to display the information.

11.7 DTrace Command Examples

Display the probes that are available with the `proc` provider.

```
 # dtrace -l -P proc
   ID   PROVIDER            MODULE                          FUNCTION NAME
 4066       proc           vmlinux                     schedule_tail start
 4067       proc           vmlinux                     schedule_tail lwp-start
 4069       proc           vmlinux             get_signal_to_deliver signal-handle
 4074       proc           vmlinux                   do_sigtimedwait signal-clear
 4075       proc           vmlinux                           do_fork lwp-create
 4076       proc           vmlinux                           do_fork create
 4077       proc           vmlinux                           do_exit lwp-exit
 4078       proc           vmlinux                           do_exit exit
 4079       proc           vmlinux                  do_execve_common exec-failure
 4080       proc           vmlinux                  do_execve_common exec
 4081       proc           vmlinux                  do_execve_common exec-success
```

```
4085       proc          vmlinux                      __send_signal signal-send
4086       proc          vmlinux                      __send_signal signal-discard
```

Monitor the system as it loads and executes process images.

```
# dtrace -n 'proc::do_execve_common:exec { trace(stringof(arg0)); }'
dtrace: description 'proc:::exec' matched 1 probe
CPU     ID                    FUNCTION:NAME
  0    600        do_execve_common:exec   /bin/uname
  0    600        do_execve_common:exec   /bin/mkdir
  0    600        do_execve_common:exec   /bin/sed
  0    600        do_execve_common:exec   /usr/bin/dirname
  1    600        do_execve_common:exec   /usr/lib64/qt-3.3/bin/firefox
  1    600        do_execve_common:exec   /usr/local/bin/firefox
  1    600        do_execve_common:exec   /usr/bin/firefox
  1    600        do_execve_common:exec   /bin/basename
  1    600        do_execve_common:exec   /bin/uname
  1    600        do_execve_common:exec   /usr/bin/mozilla-plugin-config
  1    600        do_execve_common:exec   /usr/lib64/nspluginwrapper/plugin-config
  1    600        do_execve_common:exec   /usr/lib64//xulrunner-1.9.2/mozilla-xremote-client
  1    600        do_execve_common:exec   /bin/sed
  1    600        do_execve_common:exec   /usr/lib64/firefox-3.6/run-mozilla.sh
  1    600        do_execve_common:exec   /bin/basename
  1    600        do_execve_common:exec   /bin/uname
  1    600        do_execve_common:exec   /usr/lib64/firefox-3.6/firefox
```

Display the names of commands that invoke the `open()` system call and the name of the file being opened.

```
# dtrace -q -n 'syscall::open:entry { printf("%-16s %-16s\n",execname,copyinstr(arg0)); }'
udisks-daemon    /dev/sr0
devkit-power-da  /sys/devices/LNXSYSTM:00/.../PNP0C0A:00/power_supply/BAT0/present
devkit-power-da  /sys/devices/LNXSYSTM:00/.../PNP0C0A:00/power_supply/BAT0/energy_now
devkit-power-da  /sys/devices/LNXSYSTM:00/.../PNP0C0A:00/power_supply/BAT0/voltage_max_design
devkit-power-da  /sys/devices/LNXSYSTM:00/.../PNP0C0A:00/power_supply/BAT0/voltage_min_design
devkit-power-da  /sys/devices/LNXSYSTM:00/.../PNP0C0A:00/power_supply/BAT0/status
devkit-power-da  /sys/devices/LNXSYSTM:00/.../PNP0C0A:00/power_supply/BAT0/current_now
devkit-power-da  /sys/devices/LNXSYSTM:00/.../PNP0C0A:00/power_supply/BAT0/voltage_now
VBoxService      /var/run/utmp
firefox          /home/guest/.mozilla/firefox/qeaojiol.default/sessionstore.js
firefox          /home/guest/.mozilla/firefox/qeaojiol.default/sessionstore-1.js
firefox          /home/guest/.mozilla/firefox/qeaojiol.default/sessionstore-1.js
^C
```

Display the system calls invoked by the process with ID 3007 and the number of times that it invoked each system call.

```
# dtrace -p 3007 -n 'syscall:::entry { @num[probefunc] = count(); }'
dtrace: description 'syscall:::entry ' matched 296 probes
^C

  getuid                                                            1
  ptrace                                                            1
  socket                                                            1
  waitid                                                            1
  lseek                                                             3
  statfs                                                            3
  access                                                            4
  write                                                             6
  munmap                                                           15
  newfstat                                                         16
  newstat                                                          17
  mmap                                                             19
  fcntl                                                            20
  close                                                            24
  alarm                                                            30
```

```
  inotify_add_watch                                              30
  open                                                           32
  rt_sigaction                                                   50
  nanosleep                                                      52
  rt_sigprocmask                                                 64
  ioctl                                                         117
  futex                                                         311
  clock_gettime                                                 579
  rt_sigreturn                                                  744
  gettimeofday                                                 1461
  setitimer                                                    2093
  select                                                       2530
  writev                                                       3162
  poll                                                         4720
  read                                                        10552
```

Display the distribution of the sizes specified to `read()` calls invoked by running `firefox`.

```
# dtrace -n 'syscall::read:entry /execname=="firefox"/{@dist["firefox"]=quantize(arg2);}'
dtrace: description 'syscall::read:entry ' matched 1 probe
^C

  firefox
           value  ------------- Distribution ------------- count
               0 |                                         0
               1 |@                                        566
               2 |                                         0
               4 |                                         0
               8 |                                         7
              16 |                                         4
              32 |                                         0
              64 |                                         0
             128 |                                         8
             256 |@                                        436
             512 |                                         8
            1024 |@@                                       959
            2048 |@                                        230
            4096 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@       13785
            8192 |                                         3
           16384 |                                         4
           32768 |                                         0
           65536 |                                         0
          131072 |                                         73
          262144 |                                         0
```

Run the `syscalls.d` script to examine which system calls PID 5178 is using and the number of times that it invoked each system call.

```
# ls -l syscalls.d
-rwxr-xr-x. 1 root root 85 Aug 14 14:48 syscalls.d

# cat syscalls.d
#!/usr/sbin/dtrace -qs
syscall:::entry
/pid == $1/
{
  @num[probefunc] = count();
}

# ./syscalls.d 5178
^C

  ftruncate                                                       1
  newuname                                                        1
  clone                                                           5
  close                                                           5
  sched_setscheduler                                              5
```

```
  newlstat                                                          6
  access                                                            7
  open                                                              7
  newfstat                                                          9
  sched_get_priority_max                                           10
  sched_get_priority_min                                           10
  fcntl                                                            12
  lseek                                                            73
  newstat                                                         100
  write                                                           155
  futex                                                           752
  writev                                                         1437
  poll                                                           4423
  read                                                           5397
  gettimeofday                                                   9292
```

11.8 Tracing User-Space Applications

A number of DTrace-enabled applications will be made available following the release of DTrace 0.4, including MySQL and PHP. These applications have been instrumented to contain statically defined DTrace probes. You can find details about the probes for MySQL at http://dev.mysql.com/doc/refman/5.5/en/dba-dtrace-mysqld-ref.html and about the probes for PHP at http://php.net/manual/features.dtrace.php.

The MySQL query-probes `query-start(`*`query, connectionid, database, user, host`*`)` and `query-done(`*`status`*`)` are triggered when the MySQL server receives a query is received by the server and when the query has been completed and the server has successfully sent the information to the client.

For example, the following script reports the execution time for each database query:

```
#!/usr/sbin/dtrace -qs

dtrace:::BEGIN
{
   printf("%-20s %-10s %-40s %-9s\n", "Who", "Database", "Query", "Time(microseconds)");
}

mysql*:::query-start
{
   self->query = copyinstr(arg0);
   self->connid = arg1;
   self->db    = copyinstr(arg2);
   self->who   = strjoin(copyinstr(arg3),strjoin("@",copyinstr(arg4)));
   self->querystart = timestamp;
}

mysql*:::query-done
{
   printf("%-20s %-10s %-40s %-9d\n",self->who,self->db,self->query,
          (timestamp - self->querystart) / 1000);
}
```

The following is sample output from this script:

```
# ./query.d
Who                 Database   Query                                   Time(microseconds)
webuser@localhost   namedb     select * from table1 order by n ASC     1135
webuser@localhost   namedb     delete from table1 where n='Bill'       10383
```

The MySQL query-parsing probes `query-parse-start(`*`query`*`)` and `query-parse-done(`*`status`*`)` are triggered immediately before and after MySQL parses a SQL statement. For example, you could use the following script to monitor the execution time for parsing queries:

```
#!/usr/sbin/dtrace -qs
```

```
mysql*:::query-parse-start
{
  self->parsestart = timestamp;
  self->parsequery = copyinstr(arg0);
}

mysql*:::query-parse-done
/arg0 == 0/
{
  printf("Parsing %s: %d microseconds\n", self->parsequery,
    ((timestamp - self->parsestart)/1000));
}

mysql*:::query-parse-done
/arg0 != 0/
{
  printf("Error parsing %s: %d microseconds\n", self->parsequery,
    ((timestamp - self->parsestart)/1000));
}
```

The following is sample output from this script:

```
# ./query-parse.d
Parsing select * from table1 where n like 'B%' order by n ASC: 29 microseconds
Error parsing select from table1 join (table2) on (table1.i = table2.i)
    order by table1.s,table1.i limit 10: 36 microseconds
```

The following script uses the PHP probe `error(error_message, request_file, line_number)`, to report PHP errors:

```
#!/usr/sbin/dtrace -qs

php*:::error
{
    printf("PHP error\n");
    printf("  error message             %s\n", copyinstr(arg0));
    printf("  request file              %s\n", copyinstr(arg1));
    printf("  line number               %d\n\n", (int)arg2);
}
```

For example, you can use the PHP `trigger_error()` function to trigger a PHP error if a MySQL function returns an error:

```
<?php
ini_set('error_reporting', E_ALL); /* Report all errors */
ini_set('display_errors', 'Off');  /* but do not display them */
...
  $mysqli->query($QUERY) or trigger_error($mysqli->error."[$QUERY]",E_USER_ERROR);
...
?>
```

You could use the script to report errors that might indicate incorrectly queries or attempted SQL injection attacks, for example:

```
# ./php_error.d
...
PHP error
  error message             You have an error in your SQL syntax; check the manual that
                            corresponds to your MySQL server version for the right syntax
                            to use near '='1'; --'' at line 1[select * from table1 where n
                            like 'B%' or '1'='1'; --']
  request file              /var/www/html/example.php
  line number               61
```

```
...
PHP error
  error message              You have an error in your SQL syntax; check the manual that
                             corresponds to your MySQL server version for the right syntax
                             to use near 'drop table table1; --'' at line 1[select * from
                             table1 where n like 'B%';drop table table1; --']
  request file               /var/www/html/example.php
  line number                61
...
```

11.8.1 Examining the Stack Trace of a User-Space Application

You can use the `ustack()` function to perform a stack trace of any user-space application, for example:

```
# dtrace -n syscall::write:entry'/pid == $target/ \
{ustack(); \
exit(0)}' -c "ls -l /"
dtrace: description 'syscall::write:entry' matched 1 probe
total 125
dr-xr-xr-x.   2 root root       4096 Apr 22 09:11 bin
dr-xr-xr-x.   5 root root       4096 Sep 24 09:42 boot
...
drwxr-xr-x.  14 root root       4096 Nov  2  2012 usr
drwxr-xr-x.  25 root root       4096 Apr 20 13:18 var
CPU     ID                    FUNCTION:NAME
  1      6                      write:entry
              libc.so.6`_IO_file_write+0x43
              libc.so.6`_IO_do_write+0x95
              libc.so.6`_IO_file_close_it+0x160
              libc.so.6`fclose+0x178
              ls`0x411fc9
              ls`close_stdout+0x14
              libc.so.6`exit+0xe2
              ls`0x409620
              libc.so.6`_IO_file_underflow+0x138
              libc.so.6`flush_cleanup
              libc.so.6`fclose+0x14d
              libc.so.6`fclose+0x14d
              libselinux.so.1`0x3f1840ce6f
              ls`0x412040
              ls`0x40216b
              ls`0x4027e0
              libc.so.6`__libc_start_main+0xfd
              ls`0x408480
              ls`0x4027e0
              ls`0x4027e0
              ls`0x402809
```

DTrace can translate the stack frames into symbols for shared libraries (such as `libc`) and unstripped executables. As `ls` is a stripped executable, the addresses remain unconverted. `dtrace` can translate stack frames for stripped executables if the `--export-dynamic` option was specified when the program was linked. This option causes the linker to add all symbols to the dynamic symbol table.

11.9 For More Information About DTrace

For more information, see the *Oracle Linux Dynamic Tracing Guide*.

Chapter 12 Support Diagnostic Tools

Table of Contents

This chapter describes the `sosreport`, Kdump, and OSWbb tools that you can use to help diagnose problems with a system.

12.1 About sosreport

The `sosreport` utility collects information about a system such as hardware configuration, software configuration, and operational state. You can also use `sosreport` to enable diagnostics and analytical functions. To assist in troubleshooting a problem, `sosreport` records the information in a compressed file that you can send to a support representative.

12.1.1 Configuring and Using sosreport

If the `sos` package is not already installed on your system, use `yum` to install it.

Use the following command to list the available plugins and plugin options.

```
# sosreport -l
The following plugins are currently enabled:

acpid            acpid related information
anaconda         Anaconda / Installation information
.
.
.
The following plugins are currently disabled:

amd                Amd automounter information
cluster            cluster suite and GFS related information
.
.
.
The following plugin options are available:
apache.log             off gathers all apache logs
auditd.syslogsize      15 max size (MiB) to collect per syslog file
.
.
.
```

See the `sosreport(1)` manual page for information about how to enable or disable plugins, and how to set values for plugin options.

To run `sosreport`:

1. Enter the command, specifying any options that you need to tailor the report to report information about a problem area.

   ```
   # sosreport [options ...]
   ```

   For example, to record only information about Apache and Tomcat, and to gather all the Apache logs:

   ```
   # sosreport -o apache,tomcat -k apache.log=on

   sosreport (version 2.2)
   .
   .
   .
   Press ENTER to continue, or CTRL-C to quit.
   ```

   To enable all boolean options for all loaded plugins except the `rpm.rpmva` plugin that verifies all packages, and which takes a considerable time to run:

   ```
   # sosreport -a -k rpm.rpmva=off
   ```

2. Type Enter, and enter additional information when prompted.

   ```
   Please enter your first initial and last name [email_address]: AName
   Please enter the case number that you are generating this report for: case#

     Running plugins. Please wait ...

     Completed [55/55] ...
   Creating compressed archive...

   Your sosreport has been generated and saved in:
     /tmp/sosreport-AName.case#-datestamp-ID.tar.xz

   The md5sum is: checksum

   Please send this file to your support representative.
   ```

   `sosreport` saves the report as an `xz`-compressed `tar` file in `/tmp`.

12.2 About Kdump

Kdump is the Linux kernel crash-dump mechanism. Oracle recommends that you enable the Kdump feature. In the event of a system crash, Kdump creates a memory image (`vmcore`) that can help in determining the cause of the crash. Enabling Kdump requires you to reserve a portion of system memory for exclusive use by Kdump. This memory is unavailable for other uses.

Kdump uses `kexec` to boot into a second kernel whenever the system crashes. `kexec` is a fast-boot mechanism which allows a Linux kernel to boot from inside the context of a kernel that is already running without passing through the bootloader stage.

12.2.1 Configuring and Using Kdump

During installation, you are given the option of enabling Kdump and specifying the amount of memory to reserve for it. If you prefer, you can enable kdump at a later time as described in this section.

If the `kexec-tools` and `system-config-kdump` packages are not already installed on your system, use `yum` to install them.

To enable Kdump by using the Kernel Dump Configuration GUI.

1. Enter the following command.

```
# system-config-kdump
```

The Kernel Dump Configuration GUI starts. If Kdump is currently disabled, the green **Enable** button is selectable and the **Disable** button is greyed out.

2. Click **Enable** to enable Kdump.

3. You can select the following settings tags to adjust the configuration of Kdump.

   Basic Settings
   : Allows you to specify the amount of memory to reserve for Kdump. The default setting is 128 MB.

   Target Settings
   : Allows you to specify the target location for the `vmcore` dump file on a locally accessible file system, to a raw disk device, or to a remote directory using NFS or SSH over IPv4. The default location is `/var/crash`.

     You cannot save a dump file on an eCryptfs file system, on remote directories that are NFS mounted on the `rootfs` file system, or on remote directories that access require the use of IPv6, SMB, CIFS, FCoE, wireless NICs, multipathed storage, or iSCSI over software initiators to access them.

   Filtering Settings
   : Allows to select which type of data to include in or exclude from the dump file. Selecting or deselecting the options alters the value of the argument that Kdump specifies to the `-d` option of the core collector program, `makedumpfile`.

   Expert Settings
   : Allows you to choose which kernel to use, edit the command line options that are passed to the kernel and the core collector program, choose the default action if the dump fails, and modify the options to the core collector program, `makedumpfile`.

     For example, if Kdump fails to start, and the following error appears in `/var/log/messages`, set the offset for the reserved memory to 48 MB or greater in the command line options, for example `crashkernel=128M@48M`:

     ```
     kdump: No crashkernel parameter specified for running kernel
     ```

     The Unbreakable Enterprise Kernel supports the use of the `crashkernel=auto` setting for UEK Release 3 Quarterly Update 1 and later. If you use the `crashkernel=auto` setting, the output of the `dmesg` command shows `crashkernel=XM@0M`, which is normal. The setting actually reserves 128 MB plus 64 MB for each terabyte of physical memory.

     Note

     You cannot configure `crashkernel=auto` for Xen or for the UEK prior to UEK Release 3 Quarterly Update 1. Only standard settings such as `crashkernel=128M@48M` are supported. For systems with more than 128 GB of memory, the recommended setting is `crashkernel=512M@64M`.

Click **Help** for more information on these settings.

4. Click **Apply** to save your changes. The GUI displays a popup message to remind you that you must reboot the system for the changes to take effect.
5. Click **OK** to dismiss the popup messages.
6. Select **File > Quit**.
7. Reboot the system at a suitable time.

12.2.2 Files Used by Kdump

The Kernel Dump Configuration GUI modifies the following files:

File	Description
`/boot/grub/grub.conf`	Appends the `crashkernel` option to the kernel line to specify the amount of reserved memory and any offset value.
`/etc/kdump.conf`	Sets the location where the dump file can be written, the filtering level for the `makedumpfile` command, and the default behavior to take if the dump fails. See the comments in the file for information about the supported parameters.

If you edit these files, you must reboot the system to have the changes take effect.

12.3 About OSWatcher Black Box

Oracle OSWatcher Black Box (OSWbb) collects and archives operating system and network metrics that you can use to diagnose performance issues. OSWbb operates as a set of background processes on the server and gathers data on a regular basis, invoking such Unix utilities as `vmstat`, `netstat`, `iostat`, and `top`.

From release v4.0.0, you can use the OSWbba analyzer to provide information on system slowdowns, system hangs and other performance problems, and also to graph data collected from `iostat`, `netstat`, and `vmstat`. OSWbba requires that you have installed Java version 1.4.2 or higher on your system. You can use `yum` to install Java, or you can download a Java RPM for Linux from http://www.java.com.

OSWbb is particularly useful for Oracle RAC (Real Application Clusters) and Oracle Grid Infrastructure configurations. The RAC-DDT (Diagnostic Data Tool) script file includes OSWbb, but does not install it by default.

12.3.1 Installing OSWbb

To install OSWbb:

1. Log on to My Oracle Support (MOS) at https://support.oracle.com.
2. Download the file `oswbb601.tar`, which is available at https://support.oracle.com/epmos/main/downloadattachmentprocessor?attachid=301137.1:OSW_file.
3. Copy the file to the directory where you want to install OSWbb, and run the following command:

```
# tar xvf oswbb601.tar
```

   Extracting the tar file creates a directory named `oswbb`, which contains all the directories and files that are associated with OSWbb, including the `startOSWbb.sh` script.

4. If the ksh package is not already installed on your system, use yum to install it.

```
# yum install ksh
```

5. Create a symbolic link from /usr/bin/ksh to /bin/ksh.

```
# ln -s /bin/ksh /usr/bin/ksh
```

   This link is required because the OSWbb scripts expect to find ksh in /usr/bin.

6. To enable the collection of iostat information for NFS volumes, edit the OSWatcher.sh script in the oswbb directory, and set the value of nfs_collect to 1:

```
nfs_collect=1
```

Note

This feature is available from release v5.1.

12.3.2 Running OSWbb

To start OSWbb, run the startOSWbb.sh script from the oswbb directory.

```
# ./startOSWbb.sh [frequency duration]
```

The optional frequency and duration arguments specifying how often in seconds OSWbb should collect data and the number of hours for which OSWbb should run. The default values are 30 seconds and 48 hours. The following example starts OSWbb recording data at intervals of 60 seconds, and has it record data for 12 hours:

```
# ./startOSWbb.sh 60 12
Testing for discovery of OS Utilities
.
.
.
VMSTAT found on your system.
IOSTAT found on your system.
MPSTAT found on your system.
NETSTAT found on your system.
TOP found on your system.
Testing for discovery of OS CPU COUNT
.
.
.
Starting Data Collection...
oswbb heartbeat: date/time
oswbb heartbeat: date/time + 60 seconds
.
.
.
```

To stop OSWbb prematurely, run the stopOSWbb.sh script from the oswbb directory.

```
# ./stopOSWbb.sh
```

OSWbb collects data in the following directories under the oswbb/archive directory:

Directory	Description
oswiostat	Contains output from the iostat utility.
oswmeminfo	Contains a listing of the contents of /proc/meminfo.

Directory	Description
oswmpstat	Contains output from the mpstat utility.
oswnetstat	Contains output from the netstat utility.
oswprvtnet	If you have enable private network tracing for RAC, contains information about the status of the private networks.
oswps	Contains output from the ps utility.
oswslabinfo	Contains a listing of the contents of /proc/slabinfo.
oswtop	Contains output from the top utility.
oswvmstat	Contains output from the vmstat utility.

OSWbb stores data in hourly archive files named *system_name_utility_name_timestamp*.dat, and each entry in a file is preceded by the characters *** and a timestamp.

12.4 For More Information About the Diagnostic Tools

For more information about sosreport, see the sosreport(1) manual page.

For more information about Kdump, refer to the help in the Kernel Dump Configuration GUI, and the makedumpfile(8) manual page.

For more information about OSWbb and OSWbba, refer to the *OSWatcher Black Box User Guide* (Article ID 301137.1) and the *OSWatcher Black Box Analyzer User Guide* (Article ID 461053.1), which are available from My Oracle Support (MOS) at https://support.oracle.com.