Oracle Linux 7 Managing Core System Configuration



F32772-09 December 2024

ORACLE

Oracle Linux 7 Managing Core System Configuration,

F32772-09

Copyright © 2022, 2024, Oracle and/or its affiliates.

Contents

Preface

Conventions	v
Documentation Accessibility	v
Access to Oracle Support for Accessibility	v
Diversity and Inclusion	vi

1 Working With the GRUB 2 Bootloader and Configuring Boot Services

About the Boot Process	1-1
Working With the GRUB 2 Bootloader	1-2
Customizing GRUB 2 Configuration	1-2
Using the GRUB 2 Bootloader to Set the Default Boot Kernel	1-3
Kernel Boot Parameters	1-3
Modifying Kernel Boot Parameters Before Booting	1-5
Modifying Kernel Boot Parameters in GRUB 2 Configuration	1-6

2 Working With System Services

About the systemd Service Manager	2-1
About System-State Targets	2-1
Displaying the Default and Active System-State Targets	2-2
Changing the Default and Active System-State Targets	2-4
Shutting Down, Suspending, and Rebooting the System	2-5
Starting and Stopping Services	2-5
Enabling and Disabling Services	2-6
Displaying the Status of Services	2-6
Controlling Access to System Resources	2-8
Modifying systemd Configuration Files	2-9
Running systemctl on a Remote System	2-9

3 Configuring System Settings

About the /etc/sysconfig Files	3-1
About the /proc Virtual File System	3-3

Virtual Files and Directories Under /proc	3-4
Changing Kernel Parameters	3-8
Parameters That Control System Performance	3-10
Parameters That Control Kernel Panics	3-11
About the /sys Virtual File System	3-13
Configuring System Date and Time Settings	3-13

4 Device Management

About Device Files	4-1
About the Udev Device Manager	4-3
About Udev Rules	4-3
Querying Udev and Sysfs	4-7
Modifying Udev Rules	4-10

5 Kernel Modules

5-1
5-1
5-3
5-4
5-5
5-5



Preface

WARNING:

Oracle Linux 7 is now in Extended Support. See Oracle Linux Extended Support and Oracle Open Source Support Policies for more information.

Migrate applications and data to Oracle Linux 8 or Oracle Linux 9 as soon as possible.

Oracle[®] Linux 7: Managing Core System Configuration provides information about configuring Oracle Linux 7 systems, including the boot loader configuration and processes, system devices, services and settings, as well as kernel parameters.

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 https://www.oracle.com/corporate/accessibility/.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.



Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.



Working With the GRUB 2 Bootloader and Configuring Boot Services

WARNING:

Oracle Linux 7 is now in Extended Support. See Oracle Linux Extended Support and Oracle Open Source Support Policies for more information.

Migrate applications and data to Oracle Linux 8 or Oracle Linux 9 as soon as possible.

This chapter describes the Oracle Linux boot process and how to configure and use the GRUB 2 bootloader and other boot-related kernel parameters.

About the Boot Process

Understanding the Oracle Linux boot process can help you troubleshoot problems while booting a system. The boot process involves several files and errors in these files are the usual cause of boot problems.

When an Oracle Linux system boots, it performs the following operations:

- 1. The computer's BIOS performs a power-on self-test (POST), and then locates and initializes any peripheral devices including the hard disk.
- 2. The BIOS reads the Master Boot Record (MBR) into memory from the boot device. (For GUID Partition Table (GPT) disks, this MBR is the *protective MBR* on the first sector of the disk.) The MBR stores information about the organization of partitions on that device. On a computer with x86 architecture, the MBR occupies the first 512 bytes of the boot device. The first 446 bytes contain boot code that points to the boot loader program, which can be on the same device or on another device. The next 64 bytes contain the partition table. The final two bytes are the boot signature, which is used for error detection.

The default boot loader program that is used in Oracle Linux is the GRand Unified Bootloader version 2 (GRUB 2).

- 3. The boot loader loads the vmlinuz kernel image file into memory and extracts the contents of the initramfs image file into a temporary, memory-based file system (tmpfs).
- 4. The kernel loads the driver modules from the initramfs file system that are needed to access the root file system.
- 5. The kernel starts the systemd process with a process ID of 1 (PID 1). systemd is the ancestor of all processes on a system. systemd reads its configuration from files in the /etc/systemd directory. The /etc/systemd/system.conf file controls how systemd handles system initialization.

systemd reads the file linked by /etc/systemd/system/default.target, for
example /usr/lib/systemd/system/multi-user.target, to determine the default system
target.



Note:

You can use a kernel boot parameter to override the default system target. See Kernel Boot Parameters.

The system target file defines the services that systemd starts.

systemd brings the system to the state defined by the system target, performing system initialization tasks such as:

- Setting the host name.
- Initializing the network.
- Initializing SELinux based on its configuration.
- Printing a welcome banner.
- Initializing the system hardware based on kernel boot arguments.
- Mounting the file systems, including virtual file systems such as the /proc file system.
- Cleaning up directories in /var.
- Starting swapping.

See About System-State Targets.

6. If you have made /etc/rc.local executable and you have copied /usr/lib/systemd/ system/rc-local.service to /etc/systemd/system, systemd runs any actions that you have defined in /etc/rc.local. However, the preferred way of running such local actions is to define your own systemd unit.

For information about systemd and on how to write systemd units, see the systemd(1), systemd-system.conf(5), and systemd.unit(5) manual pages.

Working With the GRUB 2 Bootloader

The GRUB 2 bootloader can load many operating systems in addition to Oracle Linux and it can chain-load proprietary operating systems. GRUB 2 understands the formats of file systems and kernel executables, which allows it to load an arbitrary operating system without needing to know the exact location of the kernel on the boot device. GRUB 2 requires only the file name and drive partitions to load a kernel.

Customizing GRUB 2 Configuration

You can manage GRUB 2 configuration by using the GRUB 2 menu or by using the command line.



Note:

Do not edit the GRUB 2 configuration file directly. On BIOS-based systems, the configuration file is /boot/grub2/grub.cfg. On UEFI-based systems, the configuration file is /boot/efi/EFI/redhat/grub.cfg.

The grub2-mkconfig command generates the configuration file by using the template scripts in the /etc/grub.d file and menu configuration settings are taken from the /etc/default/grub configuration file.

The default menu entry is determined by the value of the GRUB_DEFAULT parameter in /etc/ default/grub. The value saved allows you to use the grub2-set-default and grub2reboot commands to specify the default entry. grub2-set-default sets the default entry for all subsequent reboots and grub2-reboot sets the default entry for the next reboot only.

If you specify a numeric value as the value of GRUB_DEFAULT or as an argument to either grub2-reboot or grub2-set-default, GRUB 2 counts the menu entries in the configuration file starting at 0 for the first entry.

Using the GRUB 2 Bootloader to Set the Default Boot Kernel

To set the UEK as the default boot kernel:

1. Display the menu entries that are defined in the configuration file, for example:

grep '^menuentry' /boot/grub2/grub.cfg

```
menuentry 'Oracle Linux Everything, with Linux 3.10.0-123.el7.x86_64' ... {
menuentry 'Oracle Linux Everything, with Linux 3.8.13-35.2.1.el7uek.x86_64' ... {
menuentry 'Oracle Linux Everything, with Linux 0-
rescue-052e316f566e4a45a3391cff21b4174b' ... {
```

In this example for a BIOS-based system, the configuration file is /boot/grub2/grub.cfg, which contains menu entries 0, 1, and 2 that correspond to the RHCK, UEK, and the rescue kernel respectively.

2. Enter the following commands to make the UEK (entry 1) the default boot kernel:

```
sudo grub2-set-default 1
sudo grub2-mkconfig -o /boot/grub2/grub.cfg
```

Alternatively, you can specify the value of the text of the entry as a string enclosed in quotes.

```
sudo grub2-set-default 'Oracle Linux Everything, with Linux
3.8.13-35.2.1.el7uek.x86_64'
sudo grub2-mkconfig -o /boot/grub2/grub.cfg
```

For more information about using, configuring, and customizing GRUB 2, see the GNU GRUB Manual, which is also installed as /usr/share/doc/grub2-tools-2.00/grub.html.

Kernel Boot Parameters

There are several kernel boot parameters that you can set. The following table lists some of the more commonly used parameters.



Option	Description	
0, 1, 2, 3, 4, 5, or 6, or systemd.unit=runlevel <i>N</i> .target	Specifies the nearest systemd-equivalent system-state target to an Oracle Linux 6 run level. <i>N</i> can take an integer value between 0 and 6.	
	For a description of system-state targets, see About System-State Targets.	
1,s,S,single, or systemd.unit=rescue.target	Specifies the rescue shell. The system boots to single-user mode prompts for the root password.	
3 or systemd.unit=multi-user.target	Specifies the systemd target for multi-user, non-graphical login.	
5 or systemd.unit=graphical.target	Specifies the systemd target for multi-user, graphical login.	
-b,emergency, or systemd.unit=emergency.target	Specifies emergency mode. The system boots single-user mode and prompts for the root password. Fewer services are started than when in rescue mode.	
KEYBOARDTYPE= <i>kbtype</i>	Specifies the keyboard type, which is writh to /etc/sysconfig/keyboard in the initramfs.	
KEYTABLE= <i>kbtype</i>	Specifies the keyboard layout, which is writte to /etc/sysconfig/keyboard in the initramfs.	
LANG=language_territory.codeset	Specifies the system language and code set, which is written to /etc/sysconfig/i18n in the initramfs.	
max_loop=N	Specifies the number of loop devices (/dev/ loop*) that are available for accessing files as block devices. The default and maximum values of N are 8 and 255.	
nouptrack	Disables Ksplice Uptrack updates from being applied to the kernel.	
quiet	Reduces debugging output.	
rd_LUKS_UUID= <i>UUID</i>	Activates an encrypted Linux Unified Key Sett (LUKS) partition with the specified UUID.	
rd_LVM_VG=vg/lv_vol	Specifies an LVM volume group and volume t be activated.	
rd_NO_LUKS	Disables detection of an encrypted LUKS partition.	
rhgb	Specifies that the Red Hat graphical boot display should be used to indicate the progres of booting.	
rn_NO_DM	Disables Device-Mapper (DM) RAID detection.	
rn_NO_MD	Disables Multiple Device (MD) RAID detection	
ro root=/dev/mapper/vg-lv_root	Specifies that the root file system is to be mounted read only, and specifies the root file system by the device path of its LVM volume (where <i>vg</i> is the name of the volume group).	



Option	Description
rw root=UUID=UUID	Specifies that the root (/) file system is to be mounted read-writable at boot time, and specifies the root partition by its UUID.
selinux=0	Disables SELinux.
SYSFONT=font	Specifies the console font, which is written to /etc/sysconfig/i18n in the initramfs.

The kernel boot parameters that were last used to boot a system are recorded in /proc/ cmdline, as shown in the following example:

```
cat /proc/cmdline
```

```
BOOT_IMAGE=/vmlinuz-3.10.0-123.el7.x86_64 root=UUID=52c1cab6-969f-4872-958d-47f8518267de
ro rootflags=subvol=root vconsole.font=latarcyrheb-sun16 crashkernel=auto
vconsole.keymap=uk
rhgb quiet LANG=en_GB.UTF-8
```

For more information, see the kernel-command-line(7) manual page.

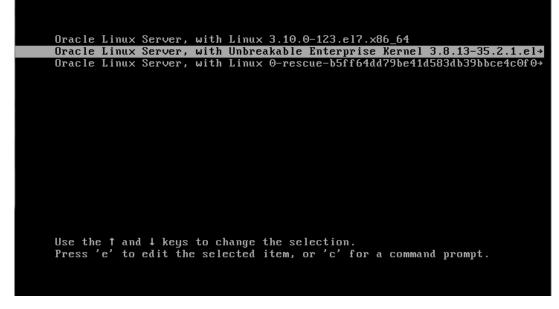
Modifying Kernel Boot Parameters Before Booting

To modify boot parameters before booting a kerne, follow these stepsl:

1. When presented with the GRUB boot menu, use the arrow keys to highlight the required kernel and then press the space bar.

The following figure shows the GRUB menu with the Unbreakable Enterprise Kernel (UEK) boot entry selected.

Figure 1-1 GRUB Menu with UEK boot entry selected



2. Press \mathbb{E} to edit the boot configuration for the kernel.



- 3. Using the arrow keys, scroll down the screen until the cursor is at the start of the boot configuration line for the kernel (which starts linux16).
- 4. Edit this line to change the boot parameters.

For example, press End to go to the end of the line and enter an additional boot parameter.

The following figure shows the kernel boot line with the additional parameter systemd.target=runlevel1.target, which starts the rescue shell.

Figure 1-2 Kernel Boot Line with an Additional Parameter to Select the Rescue Shell



5. Press Ctrl+X to boot the system.

Modifying Kernel Boot Parameters in GRUB 2 Configuration

To modify boot parameters in the GRUB 2 configuration so that they are applied by default at every reboot, follow these steps:

Edit the /etc/default/grub file and modify the parameters in the GRUB_CMDLINE_LINUX definition, for example:

```
GRUB_CMDLINE_LINUX="vconsole.font=latarcyrheb-sun16 vconsole.keymap=uk
crashkernel=auto rd.lvm.lv=ol/swap rd.lvm.lv=ol/root biosdevname=0
rhgb quiet systemd.unit=runlevel3.target"
```

The previous example adds the systemd.unit=runlevel3.target parameter so that the system boots into multi-user, non-graphical mode by default.

2. Rebuild the /boot/grub2/grub.cfg file as follows:

sudo grub2-mkconfig -o /boot/grub2/grub.cfg

This change takes effect for subsequent system reboots of all configured kernels.



WARNING:

Oracle Linux 7 is now in Extended Support. See Oracle Linux Extended Support and Oracle Open Source Support Policies for more information.

Migrate applications and data to Oracle Linux 8 or Oracle Linux 9 as soon as possible.

This chapter describes how to manage system processes, services, and resources on a running Oracle Linux system. Information about how to change the systemd target for a system, as well as how to configure the services that are available for a target is also provided.

About the systemd Service Manager

The systemd service manager replaces the Upstart init daemon in Oracle Linux 7, while also providing backward compatibility for legacy Oracle Linux 6 service scripts. The systemd service manager offers the following benefits over the init daemon:

- Services are started in parallel wherever possible by using socket-based activation and D-Bus.
- Daemons can be started on demand.
- Processes are tracked by using control groups (cgroups).
- Snapshotting of the system state and restoration of the system state from a snapshot is supported.
- mount points can be configured as systemd targets.

The systemd process is the first process that starts after the system boots and is the final process that is running when the system shuts down. systemd controls the final stages of booting and prepares the system for use. systemd also speeds up booting by loading services concurrently.

systemd enables you to manage various types of units on a system, including services
(name .service) and targets (name .target), devices (name .device), file system mount
points (name .mount), and sockets (name .socket).

For example, the following command instructs the system to mount the temporary file system (tmpfs) on /tmp at boot time:

sudo systemctl enable tmp.mount

About System-State Targets

The systemd service manager defines system-state targets that allow you to start a system with only those services that are required for a specific purpose. For example, a server can run



more efficiently with multi-user.target, because it does not run the X Window System at that run level. You should perform diagnostics, backups, and upgrades with <code>rescue.target</code> only when root can use the system. Each run level defines the services that <code>systemd</code> stops or starts. For example, <code>systemd</code> starts network services for <code>multi-user.target</code> and the X Window System for <code>graphical.target</code>; whereas, it stops both of these services for <code>rescue.target</code>.

The following table describes commonly used system-state targets and their equivalent runlevel targets, where compatibility with Oracle Linux 6 run levels is required.

System-State Targets	Equivalent Run-Level Targets	Description
graphical.target	runlevel5.target	Set up a multi-user system with networking and display manager.
multi-user.target	multi-user.target runlevel2.target	
	runlevel3.target	user system with networking.
	runlevel4.target	
poweroff.target	runlevel0.target	Shut down and power off the system.
reboot.target	runlevel6.target	Shut down and reboot the system.
rescue.target	runlevel1.target	Set up a rescue shell.

Table 2-1 System-State Targets and Equivalent Run-Level Targets

The runlevel* targets are implemented as symbolic links.

The nearest equivalent systemd target to the Oracle Linux 6 run levels 2, 3, and 4 is multi-user.target.

For more information, see the systemd.target(5) manual page.

Displaying the Default and Active System-State Targets

To display the default system-state target, use the systemctl get-default command:

sudo systemctl get-default

graphical.target

To display the currently active targets on a system, use the systemctl list-units command:

sudo systemctl list-units --type target

UNIT	LOAD	ACTIVE	SUB	DESCRIPTION
basic.target	loaded	active	active	Basic System
cryptsetup.target	loaded	active	active	Encrypted Volumes
getty.target	loaded	active	active	Login Prompts
graphical.target	loaded	active	active	Graphical Interface
local-fs-pre.target	loaded	active	active	Local File Systems (Pre)
local-fs.target	loaded	active	active	Local File Systems
multi-user.target	loaded	active	active	Multi-User System
network.target	loaded	active	active	Network
nfs.target	loaded	active	active	Network File System Server



```
loaded active active Paths
paths.target
remote-fs.target loaded active active Remote File Systems
                  loaded active active Slices
slices.target
sockets.target
                  loaded active active Sockets
sound.target
                  loaded active active Sound Card
swap.target
sysinit.target
                   loaded active active Swap
                   loaded active active System Initialization
timers.target
                   loaded active active Timers
LOAD = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB
     = The low-level unit activation state, values depend on unit type.
17 loaded units listed. Pass --all to see loaded but inactive units, too.
To show all installed unit files use 'systemctl list-unit-files'.
```

The previous example output for a system with the graphical target active shows that this target depends on 16 other active targets, including network and sound to support networking and sound.

To display the status of all targets on the system, specify the --all option:

sudo systemctl list-units --type target --all

UNIT	LOAD	ACTIVE	SUB	DESCRIPTION
basic.target		active		Basic System
cryptsetup.target		active		Encrypted Volumes
emergency.target	loaded	inactive	dead	Emergency Mode
final.target	loaded	inactive	dead	Final Step
getty.target	loaded	active	active	Login Prompts
graphical.target	loaded	active	active	Graphical Interface
local-fs-pre.target	loaded	active	active	Local File Systems (Pre)
local-fs.target	loaded	active	active	Local File Systems
multi-user.target	loaded	active	active	Multi-User System
network-online.target	loaded	inactive	dead	Network is Online
network.target	loaded	active	active	Network
nfs.target	loaded	active	active	Network File System Server
nss-lookup.target	loaded	inactive	dead	Host and Network Name Lookups
nss-user-lookup.target	loaded	inactive	dead	User and Group Name Lookups
paths.target	loaded	active	active	Paths
remote-fs-pre.target	loaded	inactive	dead	Remote File Systems (Pre)
remote-fs.target		active		Remote File Systems
rescue.target	loaded	inactive	dead	Rescue Mode
shutdown.target	loaded	inactive	dead	Shutdown
slices.target	loaded	active	active	Slices
sockets.target	loaded	active	active	Sockets
sound.target				Sound Card
swap.target	loaded	active	active	Swap
sysinit.target		active		System Initialization
syslog.target	not-fo	und inact		-
time-sync.target		inactive		1 5 5
timers.target	loaded	active	active	Timers
umount.target	loaded	inactive	dead	Unmount All Filesystems
LOAD = Reflects whether the unit definition was properly loaded.				
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.				
				values depend on unit type.
28 loaded units listed.				

To show all installed unit files use 'systemctl list-unit-files'.

For more information, see the systemctl(1) and systemd.target(5) manual pages.

Changing the Default and Active System-State Targets

Use the systemctl set-default command to change the default system-state target:

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

Note:

This command changes the target to which the default target is linked, but does not change the state of the system.

To change the currently active system target, use the systemctl isolate command:

```
sudo systemctl isolate multi-user.target
```

Listing all of the targets shows that the graphical and sound targets are not active:

sudo systemctl list-units --type target --all

UNIT	LOAD	ACTIVE	SUB	DESCRIPTION
basic.target	loaded	active	active	Basic System
cryptsetup.target	loaded	active	active	Encrypted Volumes
emergency.target	loaded	inactive	dead	Emergency Mode
final.target	loaded	inactive	dead	Final Step
getty.target	loaded	active	active	Login Prompts
graphical.target	loaded	inactive	dead	Graphical Interface
local-fs-pre.target	loaded	active	active	Local File Systems (Pre)
local-fs.target	loaded	active	active	Local File Systems
multi-user.target	loaded	active	active	Multi-User System
network-online.target	loaded	inactive	dead	Network is Online
network.target	loaded	active	active	Network
nfs.target	loaded	active	active	Network File System Server
nss-lookup.target	loaded	inactive	dead	Host and Network Name Lookups
nss-user-lookup.target	loaded	inactive	dead	User and Group Name Lookups
paths.target	loaded	active	active	Paths
remote-fs-pre.target	loaded	inactive	dead	Remote File Systems (Pre)
remote-fs.target	loaded	active	active	Remote File Systems
rescue.target	loaded	inactive	dead	Rescue Mode
shutdown.target	loaded	inactive	dead	Shutdown
slices.target	loaded	active	active	Slices
sockets.target	loaded	active	active	Sockets
sound.target	loaded	inactive	dead	Sound Card
swap.target	loaded	active	active	Swap
sysinit.target	loaded	active	active	System Initialization
syslog.target	not-fou	und inact	ive dead	d syslog.target
time-sync.target	loaded	inactive	dead	System Time Synchronized
timers.target	loaded	active	active	Timers
umount.target	loaded	inactive	dead	Unmount All Filesystems
LOAD = Reflects whether the unit definition was properly loaded.				

ACTIVE = The high-level unit activation state, i.e. generalization of SUB. SUB = The low-level unit activation state, values depend on unit type.



```
28 loaded units listed.
To show all installed unit files use 'systemctl list-unit-files'.
```

For more information, see the systemctl(1) manual page.

Shutting Down, Suspending, and Rebooting the System

The following list describes the systemctl commands that are used to shut down, reboot, or otherwise suspend the operation of a system:

- systemctl halt: Halt the system.
- systemctl hibernate: Put the system into hibernation.
- systemctl hybrid-sleep: Put the system into hibernation and suspend its operration.
- systemctl poweroff: Halt and power off the system.
- systemctl reboot: Reboot the system.
- systemctl suspend: Suspend the system.

For more information, see the systemctl(1) manual page.

Starting and Stopping Services

To start a service, use the systemctl command with the start argument:

```
sudo systemctl start sshd
```

For legacy scripts in the /etc/init.d file that have not been ported as systemd services, you can run the script directly with the start argument, for example:

/etc/init.d/yum-cron start

To stop a service, use the stop argument to systemctl:

sudo systemctl stop sshd

Note:

Changing the state of a service only lasts as long as the system remains at the same state. If you stop a service and then change the system-state target to one in which the service is configured to run (for example, by rebooting the system), the service restarts. Similarly, starting a service does not enable the service to start following a reboot. See Enabling and Disabling Services for details.

The systemctl service manager supports the disable, enable, reload, restart, start, status, and stop actions for services. For other actions, you must either run the script that the service provides to support these actions; or, for legacy scripts, the /etc/init.d script with the required action argument. For legacy scripts, omitting the argument to the script displays a usage message, for example:

/etc/init.d/yum-cron

Usage: /etc/init.d/yum-cron {start|stop|status|restart|reload|force-reload|condrestart}



For more information, see the systemctl(1) manual page.

Enabling and Disabling Services

You can use the systemctl command to enable or disable a service from starting when the system starts, for example:

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

The previous command enables a service by creating a symbolic link for the lowest-level system-state target at which the service should start. In the example, the command creates the symbolic link httpd.service for the multi-user target.

Disabling a service removes the symbolic link, for example:

```
sudo systemctl disable httpd
sudo rm '/etc/systemd/system/multi-user.target.wants/httpd.service'
```

You can use the is-enabled subcommand to check whether a service is enabled:

```
sudo systemctl is-enabled httpd
```

```
disabled
```

sudo systemctl is-enabled nfs

enabled

After running the systemctl disable command, the service can still be started or stopped by user accounts, scripts and other processes. If that is not your desired behavior, use the systemctl mask command to disable the service completely:

```
sudo systemctl mask httpd
```

```
Created symlink from '/etc/systemd/system/multi-user.target.wants/httpd.service' to '/dev/null'
```

If you try to run the service, you will see an error message stating that the unit has been masked because the service reference was changed to /dev/null:

```
sudo systemctl start httpd
```

Failed to start httpd.service: Unit is masked.

To re-link the service reference back to the matching service unit configuration file, use the systemctl unmask command:

sudo systemctl unmask httpd

For more information, see the systemctl(1) manual page.

Displaying the Status of Services

You can use the is-active subcommand to check whether a service is running (*active*) or not running (*inactive*):

```
sudo systemctl is-active httpd
```



active sudo systemctl is-active nfs inactive

sudo systemd-cgls

You can use the status action to view a detailed summary of the status of a service, including a tree of all the tasks in the *control group* (cgroup) that the service implements:

Apr 28 15:02:40 localhost.localdomain systemd[1]: Started The Apache HTTP Ser... Hint: Some lines were ellipsized, use -1 to show in full.

A cgroup is a collection of processes that are bound together so that you can control their access to system resources. In the previous example, the cgroup for the httpd service is httpd.service, which is in the system *slice*.

Slices divide the cgroups on a system into different categories. To display the slice and cgroup hierarchy, use the systemd-cgls command:

```
-1 /usr/lib/systemd/systemd --system --deserialize 17
-user.slice
 -user-0.slice
  L_session-3.scope
     └-9313 /usr/sbin/anacron -s
  -user-1000.slice
   L_session-5.scope
     -15980 sshd: root [priv]
     -15983 sshd: root@pts/1
     —15984 -bash
     -17605 sudo systemd-cgls
      -17607 systemd-cgls
     L_17608 less
system.slice
 -rngd.service
  └1042 /sbin/rngd -f
 -irqbalance.service
  L-1067 /usr/sbin/irqbalance --foreground
 -libstoragemgmt.service
  └─1057 /usr/bin/lsmd -d
 -systemd-udevd.service
  _24714 /usr/lib/systemd/systemd-udevd
  -polkit.service
  L1064 /usr/lib/polkit-1/polkitd --no-debug
  -chronyd.service
  └-1078 /usr/sbin/chronyd
```



```
-auditd.service
-1012 /sbin/auditd
-tuned.service
-2405 /usr/bin/python2 -Es /usr/sbin/tuned -1 -P
-systemd-journald.service
-820 /usr/lib/systemd/systemd-journald
-atd.service
-1824 /usr/sbin/atd -f
-sshd.service
```

system.slice contains services and other system processes, while user.slice contains user processes, which run within transient cgroups called *scopes*. In the example, the processes for the user with ID 1000 are running in the session-5.scope scope, under the /user.slice/user-1000.slice slice.

You can use the systemctl command to limit the CPU, I/O, memory, and other resources that are available to the processes in service and scope cgroups. See Controlling Access to System Resources.

For more information, see the systemctl(1) and systemd-cgls(1) manual pages.

Controlling Access to System Resources

You use the systemctl command to control a cgroup's access to system resources, for example:

sudo systemctl set-property httpd.service CPUShares=512 MemoryLimit=1G

CPUShare controls access to CPU resources. As the default value is 1024, a value of 512 halves the access that the processes in the cgroup have to CPU time. Similarly, MemoryLimit controls the maximum amount of memory that the cgroup can use.

Note:

You do not need to specify the .service extension to the name of a service.

If you specify the --runtime option, the setting does not persist across system reboots.

sudo systemctl --runtime set-property httpd CPUShares=512 MemoryLimit=1G

Alternatively, you can change the resource settings for a service under the [Service] heading in the service's configuration file in /usr/lib/systemd/system. After editing the file, direct systemd to reload its configuration files and then restart the service, as shown in the following example:

```
sudo systemctl daemon-reload
sudo systemctl restart service
```

You can run general commands within scopes and use the systemctl command to control the access that these transient cgroups have to system resources.

To run a command within in a scope, use the systemd-run command:

```
sudo systemd-run -- scope -- unit=group name [--slice=slice name] command
```



If you do not want to create the group under the default system slice, you can specify another slice or the name of a new slice.

Note: If you do not specify the --scope option, the control group is a created as a service rather than as a scope.

For example, run a command named mymonitor in mymon.scope under myslice.slice:

sudo systemd-run --scope --unit=mymon --slice=myslice mymonitor

Running as unit mymon.scope.

You can then use the systemctl command to control the access that a scope has to system resources in the same way as for a service. However, unlike a service, you must specify the .scope extension, for example:

sudo systemctl --runtime set-property mymon.scope CPUShares=256

For more information see the systemctl(1), systemd-cgls(1), and systemd.resource-control(5) manual pages.

Modifying systemd Configuration Files

If you want to change the configuration of systemd, copy the service, target, mount, socket or other file from /usr/lib/systemd/system to /etc/systemd/system and edit this copy of the original file. Note that the version of the file in /etc/systemd/system takes precedence over the version in /usr/lib/systemd/system and is not overwritten when you update a package that touches files in /usr/lib/systemd/system. To make systemd revert to using the original version of the file, either rename or delete the modified copy of the file in /etc/systemd/system. System.

Running systemctl on a Remote System

If the sshd service is running on a remote Oracle Linux 7 system, you can use the -H option with systemctl to control the system remotely, as shown in the following example:

```
sudo systemctl -H root@10.0.0.2 status sshd
root@10.0.0.2's password: password
sshd.service - OpenSSH server daemon
Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled)
Active: active (running) since Fri 2014-05-23 09:27:22 BST; 5h 43min ago
Process: 1498 ExecStartPre=/usr/sbin/sshd-keygen (code=exited, status=0/SUCCESS)
Main PID: 1524 (sshd)
CGroup: /system.slice/sshd.service
```

For more information see the systemctl(1) manual page.



WARNING:

Oracle Linux 7 is now in Extended Support. See Oracle Linux Extended Support and Oracle Open Source Support Policies for more information.

Migrate applications and data to Oracle Linux 8 or Oracle Linux 9 as soon as possible.

This chapter describes the files and virtual file systems that you can use to change configuration settings for your system.

About the /etc/sysconfig Files

The /etc/sysconfig directory contains files that control your system's configuration. The contents of this directory depend on the packages that you have installed on your system.

Some of the files that you might find in the /etc/sysconfig directory include:

atd

Specifies additional command line arguments for the atd daemon.

authconfig

Specifies whether various authentication mechanisms and options may be used. For example, the entry USEMKHOMEDIR=no disables the creation of a home directory for a user when he or she first logs in.

autofs

Defines custom options for automatically mounting devices and controlling the operation of the automounter.

${\tt crond}$

Passes arguments to the crond daemon at boot time.

firewalld

Passes arguments to the firewall daemon (firewalld) at boot time.

grub

Specifies default settings for the GRUB 2 boot loader. This file is a symbolic link to /etc/ default/grub. For more information, see Working With the GRUB 2 Bootloader.

init

Controls how the system appears and functions during the boot process.

keyboard

Specifies the keyboard.



modules (directory)

Contains scripts that the kernel runs to load additional modules at boot time. A script in the modules directory must have the extension .modules and it must have 755 executable permissions. For an example, see the bluez-uinput.modules script that loads the uinput module. For more information, see Specifying Modules To Be Loaded at Boot Time.

named

Passes arguments to the name service daemon at boot time. The named daemon is a Domain Name System (DNS) server that is part of the Berkeley Internet Name Domain (BIND) distribution. This server maintains a table that associates host names with IP addresses on the network.

nfs

Controls which ports remote procedure call (RPC) services use for NFS v2 and v3. This file allows you to set up firewall rules for NFS v2 and v3. Firewall configuration for NFS v4 does not require you to edit this file.

ntpd

Passes arguments to the network time protocol (NTP) daemon at boot time.

samba

Passes arguments to the smbd, nmbd, and winbindd daemons at boot time to support filesharing connectivity for Windows clients, NetBIOS-over-IP naming service, and connection management to domain controllers.

selinux

Controls the state of SELinux on the system. This file is a symbolic link to /etc/selinux/ config. For more information, see Oracle[®] Linux: Administering SELinux.

snapper

Defines a list of btrfs file systems and thinly-provisioned LVM volumes whose contents can be recorded as snapshots by the snapper utility. For more information, see Oracle Linux 7: Managing File Systems.

sysstat

Configures logging parameters for system activity data collector utilities such as sadc.

For more information, see /usr/share/doc/initscripts*/sysconfig.txt.

Note:

In previous releases of Oracle Linux, the host name of the system was defined in /etc/sysconfig/network. The host name is now defined in /etc/hostname and can be changed by using the hostnamectl command. The host name must be a fully qualified domain name (FQDN), for example, host20.mydomain.com, instead of a simple short name.

Additionally, system-wide default localization settings such as the default language, keyboard, and console font were defined in /etc/sysconfig/i18n. These settings are now defined in /etc/locale.conf and /etc/vconsole.conf.

For more information, see the hostname(5), hostnamectl(1), locale.conf(5), and vconsole.conf(5) manual pages.



About the /proc Virtual File System

The files in the /proc directory hierarchy contain information about your system hardware and the processes that are running on the system. You can change the configuration of the kernel by writing to certain files that have write permission.

The name of the proc file system stems from its original purpose on the Oracle Solaris operating system, which was to allow access by debugging tools to the data structures inside running processes. Linux added this interface and extended it to allow access to data structures in the kernel. Over time, /proc became quite disordered and the sysfs file system was created in an attempt to tidy it up. For more information, see About the /sys Virtual File System.

Files under the /proc directory are virtual files that the kernel creates on demand to present a browsable view of the underlying data structures and system information. As such, /proc is an example of a virtual file system. Most virtual files are listed as zero bytes in size, but they contain a large amount of information when viewed.

Virtual files such as /proc/interrupts, /proc/meminfo, /proc/mounts, and /proc/partitions provide a view of the system's hardware. Others, such as /proc/filesystems and the files under /proc/sys provide information about the system's configuration and allow this configuration to be modified.

Files that contain information about related topics are grouped into virtual directories. For example, a separate directory exists in /proc for each process that is currently running on the system, and the directory's name corresponds to the numeric process ID. /proc/1 corresponds to the systemd process, which has a PID of 1.

You can use commands such as cat, less, and view to examine virtual files within /proc. For example, /proc/cpuinfo contains information about the system's CPUs:

```
sudo cat /proc/cpuinfo
```

```
processor
                : 0
vendor_id
cpu family
model
model name
               : GenuineIntel
                : 6
                : 42
                : Intel(R) Core(TM) i5-2520M CPU @ 2.50GHz
moae. ..
stepping
                : 7
cpu MHz
                : 2393.714
cpu MHz
cache size
                : 6144 KB
physical id
                 : 0
siblings
                 : 2
core id
                 : 0
cpu cores
apicid
                 : 2
                : 0
initial apicid : 0
fpu
                 : yes
fpu exception : yes
cpuid level
                : 5
wp
                 : yes
. . .
```

Certain files under /proc require root privileges for access or contain information that is not human-readable. You can use utilities such as lspci, free, and top to access the information in these files. For example, lspci lists all PCI devices on a system:

sudo lspci



00:00.0 Host bridge: Intel Corporation 440FX - 82441FX PMC [Natoma] (rev 02) 00:01.0 ISA bridge: Intel Corporation 82371SB PIIX3 ISA [Natoma/Triton II] 00:01.1 IDE interface: Intel Corporation 82371AB/EB/MB PIIX4 IDE (rev 01) 00:02.0 VGA compatible controller: InnoTek Systemberatung GmbH VirtualBox Graphics Adapter 00:03.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet Controller (rev 02) 00:04.0 System peripheral: InnoTek Systemberatung GmbH VirtualBox Guest Service 00:05.0 Multimedia audio controller: Intel Corporation 82801AA AC'97 Audio Controller (rev 01) 00:06.0 USB controller: Apple Inc. KeyLargo/Intrepid USB 00:07.0 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 08) 00:0b.0 USB controller: Intel Corporation 82801FB/FBM/FR/FW/FRW (ICH6 Family) USB2 EHCI Controller 00:0d.0 SATA controller: Intel Corporation 82801HM/HEM (ICH8M/ICH8M-E) SATA Controller [AHCI mode] (rev 02)

Virtual Files and Directories Under /proc

The following table lists the most useful virtual files and directories under the /proc directory hierarchy.

Virtual File or Directory	Description
PID (Directory)	Provides information about the process with the process ID (<i>PID</i>). The directory's owner and group is same as the process's. Useful files under the directory include:
	cmdline Command path.
	cwd Symbolic link to the process's current working directory.
	environ Environment variables.
	exe Symbolic link to the command executable.
	fd/N File descriptors.
	maps Memory maps to executable and library files.
	root Symbolic link to the effective root directory for the process.
	stack The contents of the kernel stack.
	status Run state and memory usage.
buddyinfo	Provides information for diagnosing memory fragmentation.
bus (directory)	Contains information about the various buses (such as pci and usb) that are available on the system. You can use commands such as lspci lspcmcia, and lsusb to display information for such devices.
cgroups	Provides information about the resource control groups that are in use on the system.
cmdline	Lists parameters passed to the kernel at boot time.
cpuinfo	Provides information about the system's CPUs.
crypto	Provides information about all installed cryptographic cyphers.
devices	Lists the names and major device numbers of all currently configured characters and block devices.

Table 3-1 Useful Virtual Files and Directories Under /proc



Virtual File or Directory	Description		
dma	Lists the direct memory access (DMA) channe that are currently in use.		
driver (directory)	Contains information about drivers used by the kernel, such as those for non-volatile RAM (nvram), the real-time clock (rtc), and memory		
	allocation for sound (snd-page-alloc).		
execdomains	Lists the execution domains for binaries that the Oracle Linux kernel supports.		
filesystems	Lists the file system types that the kernel supports. Entries marked with nodev are not in use.		
fs (directory)	Contains information about mounted file systems, organized by file system type.		
interrupts	Records the number of interrupts per interrup request queue (IRQ) for each CPU since system startup.		
iomem	Lists the system memory map for each physica device.		
ioports	Lists the range of I/O port addresses that the kernel uses with devices.		
irq (directory)	Contains information about each IRQ. You can configure the affinity between each IRQ and th system CPUs.		
kcore	Presents the system's physical memory in co file format that you can examine using a debugger such as crash or gdb. This file is human-readable.		
kmsg	Records kernel-generated messages, which are picked up by programs such as dmesg.		
loadavg	Displays the system load averages (number of queued processes) for the past 1, 5, and 15 minutes, the number of running processes, the total number of processes, and the PID of the process that is running.		
locks	Displays information about the file locks that the kernel is currently holding on behalf of processes. The information provided includes:		
	• lock class (FLOCK or POSIX)		
	• lock type (ADVISORY or MANDATORY)		
	• access type (READ or WRITE)		
	 process ID major device, minor device, and inode numbers bounds of the locked region 		
mdatat			
mdstat	Lists information about multiple-disk RAID devices.		

Table 3-1 (Cont.) Useful Virtual Files and Directories Under /proc



Virtual File or Directory	Description
meminfo	Reports the system's usage of memory in more detail than is available using the free or top commands.
modules	Displays information about the modules that are currently loaded into the kernel. The lsmod command formats and displays the same information, excluding the kernel memory offset of a module.
mounts	Lists information about all mounted file systems.
net (directory)	Provides information about networking protocol, parameters, and statistics. Each directory and virtual file describes aspects of the configuration of the system's network.
partitions	Lists the major and minor device numbers, number of blocks, and name of partitions mounted by the system.
scsi/device_info	Provides information about supported SCSI devices.
scsi/scsi and scsi/sg/*	Provide information about configured SCSI devices, including vendor, model, channel, ID, and LUN data .
self	Symbolic link to the process that is examining proc.
slabinfo	Provides detailed information about slab memory usage.
softirqs	Displays information about software interrupt (<i>softirqs</i>). A softirq is similar to a hardware interrupt (<i>hardirq</i>) and allow the kernel to perform asynchronous processing that would take too long during a hardware interrupt.
stat	Records information about the system since it was started, including:
	cpu Total CPU time (measured in <i>jiffies</i>) spent in user mode, low-priority user mode, system mode, idle, waiting for I/O, handling hardirq events, and handling softirq events.
	сри N Times for CPU N.
swaps	Provides information about swap devices. The units of size and usage are kilobytes.

Table 3-1 (Cont.) Useful Virtual Files and Directories Under /proc



Virtual File or Directory	Description
sys (directory)	Provides information about the system and also allows you to enable, disable, or modify kernel features. You can write new settings to any file that has write permission. See Changing Kernel Parameters.
	The following subdirectory hierarchies of / proc/sys contain virtual files, some of whose values you can usefully alter:
	dev Device parameters.
	fs File system parameters.
	kernel Kernel configuration parameters.
	net Networking parameters.
sysvipc (directory)	Provides information about the usage of System V Interprocess Communication (IPC) resources for messages (msg), semaphores (sem), and shared memory (shm).
tty (directory)	Provides information about the available and currently used terminal devices on the system. The drivers virtual file lists the devices that are currently configured.
vmstat	Provides information about virtual memory usage.

Table 3-1	(Cont.)	Useful	Virtual	Files an	d Directories	S Under	/proc
	(00:::;)	0001011		a			, proc

For more information, see the proc (5) manual page.

Changing Kernel Parameters

Some virtual files under /proc, and under /proc/sys in particular, are writable and you can use them to adjust settings in the kernel. For example, to change the host name, you can write a new value to /proc/sys/kernel/hostname:

echo www.mydomain.com > /proc/sys/kernel/hostname

Other files take value that take binary or Boolean values. For example, the value of / proc/sys/net/ipv4/ip_forward determines whether the kernel forwards IPv4 network packets.

```
cat /proc/sys/net/ipv4/ip_forward
```

0

echo 1 > /proc/sys/net/ipv4/ip_forward
cat /proc/sys/net/ipv4/ip_forward



1

You can use the sysctl command to view or modify values under the /proc/sys directory.

Note:

Even root cannot bypass the file access permissions of virtual file entries under / proc. If you attempt to change the value of a read-only entry such as /proc/partitions, there is no kernel code to service the write() system call.

To display all of the current kernel settings:

```
sudo sysctl -a
kernel.sched_child_runs_first = 0
kernel.sched_min_granularity_ns = 2000000
kernel.sched_latency_ns = 10000000
kernel.sched_wakeup_granularity_ns = 200000
kernel.sched_shares_ratelimit = 500000
...
```

Note:

The delimiter character in the name of a setting is a period (.) rather than a slash (/) in a path relative to /proc/sys. For example, net.ipv4.ip_forward represents net/ ipv4/ip forward and kernel.msgmax represents kernel/msgmax.

To display an individual setting, specify its name as the argument to sysctl:

```
sudo sysctl net.ipv4.ip_forward
net.ipv4.ip forward = 0
```

To change the value of a setting, use the following form of the command:

```
sudo sysctl -w net.ipv4.ip forward=1
```

```
net.ipv4.ip forward = 1
```

Changes that you make in this way remain in force only until the system is rebooted. To make configuration changes persist after the system is rebooted, you must add them to the /etc/ sysctl.d directory as a configuration file. Any changes that you make to the files in this directory take effect when the system reboots or if you run the sysctl --system command, for example:

```
echo 'net.ipv4.ip_forward=1' > /etc/sysctl.d/ip_forward.conf
grep -r ip_forward /etc/sysctl.d
/etc/sysctl.d/ip_forward.conf:net.ipv4.ip_forward=1
sudo sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 0
```



```
sudo sysctl --system
* Applying /usr/lib/sysctl.d/00-system.conf ...
net.bridge.bridge-nf-call-ip6tables = 0
net.bridge.bridge-nf-call-iptables = 0
net.bridge.bridge-nf-call-arptables = 0
* Applying /usr/lib/sysctl.d/50-default.conf ...
kernel.sysrq = 16
kernel.core uses pid = 1
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.all.rp filter = 1
net.ipv4.conf.default.accept_source_route = 0
net.ipv4.conf.all.accept_source_route = 0
net.ipv4.conf.default.promote secondaries = 1
net.ipv4.conf.all.promote secondaries = 1
fs.protected hardlinks = 1
fs.protected symlinks = 1
* Applying /etc/sysctl.d/99-sysctl.conf ...
* Applying /etc/sysctl.d/ip forward.conf ...
net.ipv4.ip forward = 1
* Applying /etc/sysctl.conf ...
# sysctl net.ipv4.ip forward
net.ipv4.ip forward = \overline{1}
```

For more information, see the sysctl(8) and sysctl.d(5) manual pages.

Parameters That Control System Performance

The following parameters control aspects of system performance:

fs.file-max

Specifies the maximum number of open files for all processes. Increase the value of this parameter if you see messages about running out of file handles.

net.core.netdev_max_backlog

Specifies the size of the receiver backlog queue, which is used if an interface receives packets faster than the kernel can process them. If this queue is too small, packets are lost at the receiver, rather than on the network.

net.core.rmem_max

Specifies the maximum read socket buffer size. To minimize network packet loss, this buffer must be large enough to handle incoming network packets.

net.core.wmem_max

Specifies the maximum write socket buffer size. To minimize network packet loss, this buffer must be large enough to handle outgoing network packets.

net.ipv4.tcp_available_congestion_control

Displays the TCP congestion avoidance algorithms that are available for use. Use the modprobe command if you need to load additional modules such as tcp_htcp to implement the htcp algorithm.

net.ipv4.tcp_congestion_control

Specifies which TCP congestion avoidance algorithm is used.



net.ipv4.tcp_max_syn_backlog

Specifies the number of outstanding SYN requests that are allowed. Increase the value of this parameter if you see synflood warnings in your logs, and investigation shows that they are occurring because the server is overloaded by legitimate connection attempts.

net.ipv4.tcp_rmem

Specifies minimum, default, and maximum receive buffer sizes that are used for a TCP socket. The maximum value cannot be larger than net.core.rmem max.

net.ipv4.tcp_wmem

Specifies minimum, default, and maximum send buffer sizes that are used for a TCP socket. The maximum value cannot be larger than net.core.wmem max.

vm.swappiness

Specifies how likely the kernel is to write loaded pages to swap rather than drop pages from the system page cache. When set to 0, swapping only occurs to avoid an out of memory condition. When set to 100, the kernel swaps aggressively. For a desktop system, setting a lower value can improve system responsiveness by decreasing latency. The default value is 60.

Caution:

This parameter is intended for use with laptops to reduce power consumption by the hard disk. Do not adjust this value on server systems.

Parameters That Control Kernel Panics

The following parameters control the circumstances under which a kernel panic can occur:

kernel.hung_task_panic

(UEK R3 only) If set to 1, the kernel panics if any kernel or user thread sleeps in the TASK_UNINTERRUPTIBLE state (*D state*) for more than kernel.hung_task_timeout_secs seconds. A process remains in D state while waiting for I/O to complete. You cannot kill or interrupt a process in this state.

The default value is 0, which disables the panic.

🔵 Tip:

To diagnose a hung thread, you can examine /proc/PID/stack, which displays the kernel stack for both kernel and user threads.

kernel.hung_task_timeout_secs

(UEK R3 only) Specifies how long a user or kernel thread can remain in D state before a warning message is generated or the kernel panics (if the value of kernel.hung_task_panic is 1). The default value is 120 seconds. A value of 0 disables the timeout.

kernel.nmi_watchdog

If set to 1 (default), enables the non-maskable interrupt (NMI) watchdog thread in the kernel. If you want to use the NMI switch or the OProfile system profiler to generate an undefined NMI, set the value of kernel.nmi watchdog to 0.



kernel.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.

kernel.panic_on_io_nmi

If set to 0 (default), the system tries to continue operations if the kernel detects an I/O channel check (IOCHK) NMI that usually indicates a uncorrectable hardware error. If set to 1, the system panics.

kernel.panic_on_oops

If set to 0, the system tries to continue operations if the kernel encounters an oops or BUG condition. If set to 1 (default), the system delays a few seconds to give the kernel log daemon, klogd, time to record the oops output before the panic occurs.

In an OCFS2 cluster. set the value to 1 to specify 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.

kernel.panic_on_stackoverflow

(RHCK only) If set to 0 (default), the system tries to continue operations if the kernel detects an overflow in a kernel stack. If set to 1, the system panics.

kernel.panic_on_unrecovered_nmi

If set to 0 (default), the system tries to continue operations if the kernel detects an NMI that usually indicates an uncorrectable parity or ECC memory error. If set to 1, the system panics.

kernel.softlockup_panic

If set to 0 (default), the system tries to continue operations if the kernel detects a *soft-lockup* error that causes the NMI watchdog thread to fail to update its time stamp for more than twice the value of kernel.watchdog thresh seconds. If set to 1, the system panics.

kernel.unknown_nmi_panic

If set to 1, the system panics if the kernel detects an undefined NMI. You would usually generate an undefined NMI by manually pressing an NMI switch. As the NMI watchdog thread also uses the undefined NMI, set the value of kernel.unknown_nmi_panic to 0 if you set kernel.nmi watchdog to 1.

kernel.watchdog_thresh

Specifies the interval between generating an NMI performance monitoring interrupt that the kernel uses to check for *hard-lockup* and *soft-lockup* errors. A hard-lockup error is assumed if a CPU is unresponsive to the interrupt for more than kernel.watchdog_thresh seconds. The default value is 10 seconds. A value of 0 disables the detection of lockup errors.

vm.panic_on_oom

If set to 0 (default), the kernel's OOM-killer scans through the entire task list and attempts to kill a memory-hogging process to avoid a panic. If set to 1, the kernel panics but can survive under certain conditions. If a process limits allocations to certain nodes by using memory policies or cpusets, and those nodes reach memory exhaustion status, the OOM-killer can kill one process. No panic occurs in this case because other nodes' memory might be free and the system as a whole might not yet be out of memory. If set to 2, the kernel always panics when an OOM condition occurs. Settings of 1 and 2 are for intended for use with clusters, depending on your preferred failover policy.



About the /sys Virtual File System

In addition to /proc, the kernel exports information to the /sys virtual file system (sysfs). Programs such as the dynamic device manager, udev, use /sys to access device and device driver information. The implementation of /sys has helped to tidy up the /proc file system as most hardware information has been moved to /sys.

Note:

/sys exposes kernel data structures and control points, which implies that it might contain circular references, where a directory links to an ancestor directory. As a result, a find command used on /sys might never terminate.

The following list identifies useful virtual directories under the /sys directory hierarchy.

• block

Contains subdirectories for block devices. For example: /sys/block/sda.

• bus

Contains subdirectories for each supported physical bus type, such as pci, pcmcia, scsi, or usb. Under each bus type, the devices directory lists discovered devices, and the drivers directory contains directories for each device driver.

class

Contains subdirectories for every class of device that is registered with the kernel.

devices

Contains the global device hierarchy of all devices on the system. The platform directory contains peripheral devices such as device controllers that are specific to a particular platform. The system directory contains non-peripheral devices such as CPUs and APICs. The virtual directory contains virtual and pseudo devices. See Device Management.

• firmware

Contains subdirectories for firmware objects.

• module

Contains subdirectories for each module loaded into the kernel. You can alter some parameter values for loaded modules. See About Module Parameters.

• power

Contains attributes that control the system's power state.

For more information, see https://www.kernel.org/doc/Documentation/filesystems/sysfs.txt.

Configuring System Date and Time Settings

System time is based on the POSIX time standard, where time is measured as the number of seconds that have elapsed since 00:00:00 Coordinated Universal Time (UTC), Thursday, 1 January 1970. A day is defined as 86400 seconds and leap seconds are subtracted automatically.



Date and time representation on a system can be set to match a specific timezone. To list all of the available timezones, run:

sudo timedatectl list-timezones

To set the system timezone to match a value returned from the available timezones, you can run:

sudo timedatectl set-timezone America/Los Angeles

Substitute America/Los_Angeles with a valid timezone entry.

This command sets a symbolic link from /etc/localtime to point to the appropriate zone information file in /usr/share/zoneinfo/. The setting takes effect immediately. Some long running processes that might use /etc/localtime to detect the current system timezone, may not detect a subsequent change in system timezone until the process is restarted.

Note that timezones are largely used for display purposes or to handle user input. Changing timezone does not change the time for the system clock. You can change the presentation for system time in any console by setting the TZ environment variable. For example, to see the current time in Tokyo, you can run:

TZ="Asia/Tokyo" date

You can check your system's current date and time configuration by running the timedatectl command on its own:

sudo timedatectl

Universal time: RTC time:	Thu 2018-10-25 13:11:30 BST Thu 2018-10-25 12:11:30 UTC Thu 2018-10-25 12:11:17 Europe/London (BST, +0100)
NTP enabled:	yes
NTP synchronized:	yes
RTC in local TZ:	no
DST active:	yes
Last DST change:	DST began at
	Sun 2018-03-25 00:59:59 GMT
	Sun 2018-03-25 02:00:00 BST
Next DST change:	DST ends (the clock jumps one hour backwards) at Sun 2018-10-28 01:59:59 BST Sun 2018-10-28 01:00:00 GMT

To set system time manually, you can use the timedatectl set-time command. For example. you can run:

sudo timedatectl set-time "2018-10-28 01:59:59"

This command sets the current system time based on the time specified assuming the currently set system timezone. The command also updates the system Real Time Clock (RTC).

Consider configuring your system to use network time synchronization for accurate timekeeping. This can be particularly important when setting up high-availability or when using network-based file systems.

If you configure an NTP service, you can enable NTP by running the following command:

sudo timedatectl set-ntp true

This command enables and starts the chronyd service, if available.



WARNING:

Oracle Linux 7 is now in Extended Support. See Oracle Linux Extended Support and Oracle Open Source Support Policies for more information.

Migrate applications and data to Oracle Linux 8 or Oracle Linux 9 as soon as possible.

This chapter describes how the system uses device files and how the udev device manager dynamically creates or removes device node files.

About Device Files

The /dev directory contains *device files* (also sometimes known as *device special files* and *device nodes*) that provide access to peripheral devices such as hard disks, to resources on peripheral devices such as disk partitions, and pseudo devices such as a random number generator.

The /dev directory has several subdirectory hierarchies, each of which holds device files that relate to a certain type of device. For example, the /dev/disk/id-by-uuid directory contains device files for hard disks named according to the universally unique identifier (UUID) for the disk. The device files in subdirectories such as these are actually implemented as symbolic links to device files in /dev. You can access the same device using the file in /dev or the corresponding link to the file listed in /dev/disk/id-by-uuid.

If you use the ls -l command to list the files under /dev, you see that some device files are shown as being either type b for *block* or type c for *character*. These devices have a pair of numbers associated with them instead of a file size. These *major* and *minor* numbers identify the device to the system.

```
ls -l /dev
```

```
total O
crw-rw----. 1 root root 10, 56 Mar 17 08:17 autofs
                           640 Mar 17 08:17 block
drwxr-xr-x. 2 root root
                               80 Mar 17 08:16 bsg
drwxr-xr-x. 2 root root
                                60 Mar 17 08:16 bus
drwxr-xr-x. 3 root root
lrwxrwxrwx. 1 root root
                                3 Mar 17 08:17 cdrom -> sr0
                           2880 Mar 17 08:17 char
drwxr-xr-x. 2 root root
                           5, 1 Mar 17 08:17 console
crw-----. 1 root root
lrwxrwxrwx. 1 root
                   root
                           11 Mar 1, 1,
100 Mar 17 08:17 cpu
                                11 Mar 17 08:17 core -> /proc/kcore
drwxr-xr-x. 4 root
                   root
crw-rw----. 1 root
                            10, 61 Mar 17 08:17 cpu dma latency
                   root
drwxr-xr-x. 6 root
                           120 Mar 17 08:16 disk
                   root
brw-rw----. 1 root
                   disk
                           253, 0 Mar 17 08:17 dm-0
brw-rw----. 1 root
                   disk
                           253,
                                 1 Mar 17 08:17 dm-1
. . .
crw-rw-rw-, 1 root.
                    root
                             1,
                                 3 Mar 17 08:17 /dev/null
```

```
drwxr-xr-x. 2 root
                                   0 Mar 17 08:16 pts
                     root.
. . .
                                   8 Mar 17 08:17 random
crw-rw-rw-. 1 root
                     root
                              1,
. . .
brw-rw----. 1 root
                      disk
                              8, 0 Mar 17 08:17 sda
brw-rw----. 1 root
                      disk
                              8,
                                   1 Mar 17 08:17 sda1
brw-rw----. 1 root
                      disk
                              8,
                                  2 Mar 17 08:17 sda2
. . .
                                  15 Mar 17 08:17 stderr -> /proc/self/fd/2
lrwxrwxrwx. 1 root
                     root
lrwxrwxrwx. 1 root
                                  15 Mar 17 08:17 stdin -> /proc/self/fd/0
                     root.
lrwxrwxrwx. 1 root
                     root
                                  15 Mar 17 08:17 stdout -> /proc/self/fd/1
. . .
crw--w---. 1 root tty 4, 0 Mar 17 08:17 tty0
crw--w---. 1 root tty
                             4, 1 Mar 17 08:17 tty1
. . .
crw-rw-rw-. 1 root
                     root
                              1,
                                  9 Mar 17 08:17 urandom
. . .
                              1,
                                   5 Mar 17 08:17 zero
crw-rw-rw-. 1 root
                      root.
```

Block devices support random access to data, seeking media for data, and usually allow data to be buffered while it is being written or read. Examples of block devices include hard disks, CD-ROM drives, flash memory, and other addressable memory devices. The kernel writes data to or reads data from a block device in blocks of a certain number of bytes. In the sample output, sda is the block device file that corresponds to the hard disk, and it has a major number of 8 and a minor number of 0. sda1 and sda2 are partitions of this disk, and they have the same major number as sda (8), but their minor numbers are 1 and 2.

Character devices support streaming of data to or from a device, and data is not usually buffered nor is random access permitted to data on a device. The kernel writes data to or reads data from a character device one byte at a time. Examples of character devices include keyboards, mice, terminals, pseudo-terminals, and tape drives. tty0 and tty1 are character device files that correspond to terminal devices that allow users to log in from serial terminals or terminal emulators. These files have major number 4 and minor numbers 0 and 1.

Pseudo-terminals worker or secondary (slave) devices emulate real terminal devices to interact with software. For example, a user might log in on a terminal device such as /dev/tty1, which then uses the pseudo-terminal primary (master) device /dev/pts/ptmx to interact with an underlying pseudo-terminal device. The character device files for worker and primary pseudo-terminals are located in the /dev/pts directory:

```
ls -l /dev/pts
```

total 0 crw--w---. 1 guest tty 136, 0 Mar 17 10:11 0 crw--w---. 1 guest tty 136, 1 Mar 17 10:53 1 crw--w---. 1 guest tty 136, 2 Mar 17 10:11 2 c----. 1 root root 5, 2 Mar 17 08:16 ptmx

Some device entries, such as stdin for the standard input, are symbolically linked via the self subdirectory of the proc file system. The pseudo-terminal device file to which they actually point depends on the context of the process.

```
ls -l /proc/self/fd/[012]
```

```
total 0
lrwx-----. 1 root root 64 Mar 17 10:02 0 -> /dev/pts/1
lrwx-----. 1 root root 64 Mar 17 10:02 1 -> /dev/pts/1
lrwx-----. 1 root root 64 Mar 17 10:02 2 -> /dev/pts/1
```



Character devices such as null, random, urandom, and zero are examples of pseudo-devices that provide access to virtual functionality implemented in software rather than to physical hardware.

/dev/null is a data sink. Data that you write to /dev/null effectively disappears but the write operation succeeds. Reading from /dev/null returns EOF (end-of-file).

/dev/zero is a data source of an unlimited number of zero-value bytes.

/dev/random and /dev/urandom are data sources of streams of pseudo-random bytes. To maintain high-entropy output, /dev/random blocks if its entropy pool does not contains sufficient bits of noise. /dev/urandom does not block and, as a result, the entropy of its output might not be as consistently high as that of /dev/random. However, neither /dev/random nor /dev/urandom are considered to be truly random enough for the purposes of secure cryptography such as military-grade encryption.

You can find out the size of the entropy pool and the entropy value for /dev/random from virtual files under /proc/sys/kernel/random:

```
cat /proc/sys/kernel/random/poolsize
```

4096

cat /proc/sys/kernel/random/entropy_avail

3467

For more information, see the null(4), pts(4), and random(4) manual pages.

About the Udev Device Manager

The udev device manager dynamically creates or removes device node files at boot time or if you add a device to or remove a device from the system with a 2.6 version kernel or later. When creating a device node, udev reads the device's /sys directory for attributes such as the label, serial number, and bus device number.

Udev can use persistent device names to guarantee consistent naming of devices across reboots, regardless of their order of discovery. Persistent device names are especially important when using external storage devices.

The configuration file for udev is /etc/udev/udev.conf. The file contains the variable udev_log which indicates the logging priority. The variable can be set to err, info and debug. The default value is err.

For more information, see the udev (7) manual page.

About Udev Rules

Udev uses rules files that determine how it identifies devices and creates device names. The udev service (systemd-udevd) reads the rules files at system startup and stores the rules in memory. If the kernel discovers a new device or an existing device goes offline, the kernel sends an event action (*uevent*) notification to udev, which matches the in-memory rules against the device attributes in /sys to identify the device. As part of device event handling, rules can specify additional programs that should run to configure a device. Rules files, which have the file extension .rules, are located in the following directories:



/lib/udev/rules.d Contains default rules files. Do not edit these files.

```
/etc/udev/rules.d/*.rules
```

Contains customized rules files. You can modify these files.

/dev/.udev/rules.d/*.rules

Contains temporary rules files. Do not edit these files.

Udev processes the rules files in lexical order, regardless of which directory they are located. Rules files in /etc/udev/rules.d override files of the same name in /lib/udev/rules.d.

The following rules are extracted from the file /lib/udev/rules.d/50-udev- default.rules and illustrate the syntax of udev rules.

```
# do not edit this file, it will be overwritten on update
SUBSYSTEM=="block", SYMLINK{unique}+="block/%M:%m"
SUBSYSTEM!="block", SYMLINK{unique}+="char/%M:%m"
KERNEL=="pty[pqrstuvwxyzabcdef][0123456789abcdef]", GROUP="tty", MODE="0660"
KERNEL=="tty[pqrstuvwxyzabcdef][0123456789abcdef]", GROUP="tty", MODE="0660"
. . .
# mem
KERNEL=="null|zero|full|random|urandom", MODE="0666"
KERNEL=="mem|kmem|port|nvram", GROUP="kmem", MODE="0640"
. . .
# block
SUBSYSTEM=="block", GROUP="disk"
. . .
# network
                              MODE="0666"
KERNEL=="tun",
KERNEL=="rfkill",
                               MODE="0644"
# CPU
KERNEL=="cpu[0-9] *",
                               MODE="0444"
. . .
# do not delete static device nodes
ACTION=="remove", NAME=="", TEST=="/lib/udev/devices/%k", \
    OPTIONS+="ignore remove"
ACTION=="remove", NAME=="?*", TEST=="/lib/udev/devices/$name", \
    OPTIONS+="ignore remove"
```

Comment lines begin with a # character. All other non-blank lines define a rule, which is a list of one or more comma-separated key-value pairs. A rule either assigns a value to a key or it tries to find a match for a key by comparing its current value with the specified value. The following list shows the assignment and comparison operators that you can use:

- =: Assign a value to a key, overwriting any previous value.
- +=: Assign a value by appending it to the key's current list of values.
- :=: Assign a value to a key. This value cannot be changed by any further rules.
- ==: Match the key's current value against the specified value for equality.
- !=: Match the key's current value against the specified value for equality.

You can use the following shell-style pattern matching characters in values:

- ?: Matches a single character.
- *: Matches any number of characters, including zero.



• []: Matches any single character or character from a range of characters specified within the brackets. For example, tty[ss][0-9] would match ttys7 or ttys7.

The following list shows commonly used match keys in rules.

• ACTION

Matches the name of the action that led to an event. For example, <code>ACTION="add"</code> or <code>ACTION="remove"</code>.

ENV{key}

Matches a value for the device property key. For example, ENV{DEVTYPE}=="disk".

• KERNEL

Matches the name of the device that is affected by an event. For example, KERNEL=="dm-*" for disk media.

• NAME

Matches the name of a device file or network interface. For example, NAME="?*" for any name that consists of one or more characters.

SUBSYSTEM

Matches the subsystem of the device that is affected by an event. For example, SUBSYSTEM=="tty".

• TEST

Tests if the specified file or path exists. For example, TEST=="/lib/udev/devices/\$name", where \$name is the name of the currently matched device file.

Other match keys include ATTR{filename}, ATTRS{filename}, DEVPATH, DRIVER, DRIVERS, KERNELS, PROGRAM, RESULT, SUBSYSTEMS, and SYMLINK.

The following list shows commonly used assignment keys in rules.

ENV{key}

Specifies a value for the device property key. For example, GROUP="disk".

• GROUP

Specifies the group for a device file. For example, GROUP="disk".

• IMPORT{type}:

Specifies a set of variables for the device property, depending on *type*:

- cmdline: Import a single property from the boot kernel command line. For simple flags, udev sets the value of the property to 1. For example, IMPORT {cmdline}="nodmraid".
- db: Interpret the specified value as an index into the device database and import a single property, which must have already been set by an earlier event. For example, IMPORT {db}="DM_UDEV_LOW_PRIORITY_FLAG".
- file: Interpret the specified value as the name of a text file and import its contents, which must be in environmental key format. For example, IMPORT{file}="keyfile".
- parent: Interpret the specified value as a key-name filter and import the stored keys from the database entry for the parent device. For example IMPORT{parent}="ID *".

- program: Run the specified value as an external program and imports its result, which must be in environmental key format. For example IMPORT{program}="usb_id -export %p".
- MODE

Specifies the permissions for a device file. For example, MODE="0640".

• NAME

Specifies the name of a device file. For example, NAME="em1".

• OPTIONS

Specifies rule and device options. For example, OPTIONS+="ignore_remove", which means that the device file is not removed if the device is removed.

• OWNER

Specifies the owner for a device file. For example, GROUP="root".

• RUN

Specifies a command to be run after the device file has been created. For example, RUN+="/usr/bin/eject \$kernel", where \$kernel is the kernel name of the device.

• SYMLINK

Specifies the name of a symbolic link to a device file. For example, SYMLINK+="disk/by-uuid/\$env{ID_FS_UUID_ENC}", where \$env{} is substituted with the specified device property.

Other assignment keys include ATTR{key}, GOTO, LABEL, RUN, and WAIT FOR.

The following list shows string substitutions that are commonly used with the GROUP, MODE, NAME, OWNER, PROGRAM, RUN, and SYMLINK keys:

• \$attr{file} **Or** %s{file}

Specifies the value of a device attribute from a file under /sys. For example, ENV{MATCHADDR}="\$attr{address}".

• \$devpath **Or** %p

The device path of the device in the sysfs file system under /sys. For example, RUN+="keyboard-force-release.sh \$devpath common-volume-keys".

• \$env{*key*} **Of** %E{*key*}

Specifies the value of a device property. For example, SYMLINK+="disk/by-id/md-name-\$env{MD NAME}-part%n".

• \$kernel **Or** %k

The kernel name for the device.

• \$major **Or** %M

Specifies the major number of a device. For example, IMPORT{program}="udisks-dm-export %M %m".

• \$minor **Or** %m

Specifies the minor number of a device. For example, RUN+="\$env{LVM_SBIN_PATH}/lvm pvscan --cache --major \$major --minor \$minor".

• \$name



```
Specifies the device file of the current device. For example, TEST=="/lib/udev/
devices/$name".
```

Udev expands the strings specified for RUN immediately before its program is executed, which is after udev has finished processing all other rules for the device. For the other keys, udev expands the strings while it is processing the rules.

For more information, see the udev (7) manual page.

Querying Udev and Sysfs

You can use the udevadm command to query the udev database and sysfs.

For example, to query the sysfs device path relative to /sys that corresponds to the device file /dev/sda:

sudo udevadm info --query=path --name=/dev/sda

/devices/pci0000:00/0000:00:0d.0/host0/target0:0:0/0:0:0:0/block/sda

To query the symbolic links that point to /dev/sda:

sudo udevadm info --query=symlink --name=/dev/sda

```
block/8:0
disk/by-id/ata-VBOX_HARDDISK_VB6ad0115d-356e4c09
disk/by-id/scsi-SATA_VBOX_HARDDISK_VB6ad0115d-356e4c09
disk/by-path/pci-0000:00:0d.0-scsi-0:0:0:0
```

To query the properties of /dev/sda:

sudo udevadm info --query=property --name=/dev/sda

```
UDEV LOG=3
DEVPATH=/devices/pci0000:00/0000:00:0d.0/host0/target0:0:0/0:0:0/block/sda
MAJOR=8
MINOR=0
DEVNAME=/dev/sda
DEVTYPE=disk
SUBSYSTEM=block
ID ATA=1
ID TYPE=disk
ID BUS=ata
ID MODEL=VBOX HARDDISK
ID REVISION=1.0
ID SERIAL=VBOX HARDDISK VB579a85b0-bf6debae
ID SERIAL SHORT=VB579a85b0-bf6debae
ID ATA WRITE CACHE=1
ID ATA WRITE CACHE ENABLED=1
ID ATA FEATURE SET PM=1
ID ATA FEATURE SET PM ENABLED=1
ID ATA SATA=1
ID ATA SATA SIGNAL RATE GEN2=1
ID SCSI COMPAT=SATA VBOX HARDDISK VB579a85b0-bf6debae
ID PATH=pci-0000:00:0d.0-scsi-0:0:0:0
ID PART TABLE TYPE=dos
LVM SBIN PATH=/sbin
UDISKS PRESENTATION NOPOLICY=0
UDISKS PARTITION TABLE=1
UDISKS PARTITION TABLE SCHEME=mbr
```



```
UDISKS PARTITION TABLE COUNT=2
UDISKS ATA SMART IS AVAILABLE=0
DEVLINKS=/dev/block/8:0 /dev/disk/by-id/ata-VBOX_HARDDISK_VB579a85b0-bf6debae ...
To guery all information for /dev/sda:
sudo udevadm info --query=all --name=/dev/sda
P: /devices/pci0000:00/0000:00:0d.0/host0/target0:0:0/0:0:0:0/block/sda
N: sda
W: 37
S: block/8:0
S: disk/by-id/ata-VBOX HARDDISK VB579a85b0-bf6debae
S: disk/by-id/scsi-SATA VBOX HARDDISK VB579a85b0-bf6debae
S: disk/by-path/pci-0000:00:0d.0-scsi-0:0:0:0
E: UDEV LOG=3
E: DEVPATH=/devices/pci0000:00/0000:00:0d.0/host0/target0:0:0/0:0:0/block/sda
E: MAJOR=8
E: MINOR=0
E: DEVNAME=/dev/sda
E: DEVTYPE=disk
E: SUBSYSTEM=block
E: ID ATA=1
E: ID TYPE=disk
E: ID BUS=ata
E: ID MODEL=VBOX HARDDISK
E: ID_SERIAL=VBOX_HARDDISK_VB579a85b0-bf6debae
E: ID SERIAL SHORT=VB579a85b0-bf6debae
E: ID ATA WRITE CACHE=1
E: ID ATA WRITE CACHE ENABLED=1
E: ID ATA FEATURE SET PM=1
E: ID ATA FEATURE SET PM ENABLED=1
E: ID ATA SATA=1
E: ID ATA SATA SIGNAL RATE GEN2=1
E: ID SCSI COMPAT=SATA VBOX HARDDISK VB579a85b0-bf6debae
E: ID PATH=pci-0000:00:0d.0-scsi-0:0:0:0
E: ID PART TABLE TYPE=dos
E: LVM SBIN PATH=/sbin
E: UDISKS PRESENTATION NOPOLICY=0
E: UDISKS_PARTITION TABLE=1
E: UDISKS PARTITION TABLE SCHEME=mbr
E: UDISKS PARTITION TABLE COUNT=2
E: UDISKS ATA SMART IS AVAILABLE=0
E: DEVLINKS=/dev/block/8:0 /dev/disk/by-id/ata-VBOX HARDDISK VB579a85b0-bf6debae ...
To display all properties of /dev/sda and its parent devices that udev has found in /sys:
sudo udevadm info --attribute-walk --name=/dev/sda
. . .
  looking at device '/devices/pci0000:00/0000:00:0d.0/host0/target0:0:0/0:0:0:0/block/
sda':
   KERNEL=="sda"
    SUBSYSTEM=="block"
    DRIVER==""
    ATTR{range}=="16"
    ATTR{ext range}=="256"
    ATTR{removable} == "0"
    ATTR{ro} == "0"
```



ATTR{size}=="83886080"
ATTR{alignment offset}=="0"

```
ATTR{capability}=="52"
   ATTR{stat}==" 20884
                           15437 1254282 338919
                                                     5743 8644
                                                                     103994
109005 ...
   ATTR{inflight}=="
                          0
                                   0"
  looking at parent device '/devices/pci0000:00/0000:00:0d.0/host0/target0:0:0/0:0:0:0':
   KERNELS=="0:0:0:0"
   SUBSYSTEMS=="scsi"
   DRIVERS=="sd"
   ATTRS{device blocked} == "0"
   ATTRS{type}=="0"
   ATTRS{scsi level}=="6"
   ATTRS { vendor } == "ATA
                           ...
   ATTRS{model}=="VBOX HARDDISK
   ATTRS{rev} == "1.0 "
   ATTRS { state } == "running"
   ATTRS{timeout}=="30"
   ATTRS{iocounterbits}=="32"
   ATTRS{iorequest cnt}=="0x6830"
   ATTRS{iodone cnt}=="0x6826"
   ATTRS{ioerr cnt}=="0x3"
   ATTRS {modalias} == "scsi:t-0x00"
   ATTRS{evt media change} == "0"
   ATTRS{dh state}=="detached"
   ATTRS{queue_depth}=="31"
   ATTRS{queue ramp up period}=="120000"
   ATTRS{queue type}=="simple"
  looking at parent device '/devices/pci0000:00/0000:00:0d.0/host0/target0:0:0':
   KERNELS=="target0:0:0"
   SUBSYSTEMS=="scsi"
   DRIVERS==""
 looking at parent device '/devices/pci0000:00/0000:00:0d.0/host0':
   KERNELS=="host0"
   SUBSYSTEMS=="scsi"
   DRIVERS==""
  looking at parent device '/devices/pci0000:00/0000:00:0d.0':
   KERNELS=="0000:00:0d.0"
   SUBSYSTEMS=="pci"
   DRIVERS=="ahci"
   ATTRS {vendor} == "0x8086"
   ATTRS{device}=="0x2829"
   ATTRS{subsystem vendor}=="0x0000"
   ATTRS{subsystem device} == "0x0000"
   ATTRS{class}=="0x010601"
   ATTRS{irg}=="21"
03"
   ATTRS{local cpulist}=="0-1"
   ATTRS{modalias}=="pci:v00008086d00002829sv0000000sd000000bc01sc06i01"
   ATTRS{numa node}=="-1"
   ATTRS{enable}=="1"
   ATTRS{broken parity status}=="0"
   ATTRS{msi bus}==""
   ATTRS{msi_irqs}==""
  looking at parent device '/devices/pci0000:00':
   KERNELS=="pci0000:00"
```



SUBSYSTEMS=="" DRIVERS==""

The command starts at the device specified by its device path and walks up the chain of parent devices. For every device that it finds, it displays all possible attributes for the device and its parent devices in the match key format for udev rules.

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

Modifying Udev Rules

The order in which rules are evaluated is important. Udev processes rules in lexical order. If you want to add your own rules, you need udev to find and evaluate these rules before the default rules.

The following example illustrates how to implement a udev rules file that adds a symbolic link to the disk device /dev/sdb.

1. Create a rule file under /etc/udev/rules.d with a file name such as 10-local.rules that udev will read before any other rules file.

For example, the following rule in 10-local.rules creates the symbolic link /dev/my_disk, which points to /dev/sdb:

```
KERNEL=="sdb", ACTION=="add", SYMLINK="my_disk"
```

Listing the device files in /dev shows that udev has not yet applied the rule:

sudo ls /dev/sd* /dev/my_disk

ls: cannot access /dev/my_disk: No such file or directory
/dev/sda /dev/sda1 /dev/sda2 /dev/sdb

 To simulate how udev applies its rules to create a device, you can use the udevadm test command with the device path of sdb listed under the /sys/class/block hierarchy, for example:

sudo udevadm test /sys/class/block/sdb

```
calling: test
version ...
This program is for debugging only, it does not run any program
specified by a RUN key. It may show incorrect results, because
some values may be different, or not available at a simulation run.
...
LINK 'my_disk' /etc/udev/rules.d/10-local.rules:1
...
creating link '/dev/my_disk' to '/dev/sdb'
creating symlink '/dev/my_disk' to 'sdb
...
ACTION=add
DEVLINKS=/dev/disk/by-id/ata-VBOX_HARDDISK_VB186e4ce2-f80f170d
/dev/disk/by-uuid/a7dc508d-5bcc-4112-b96e-f40b19e369fe
/dev/my_disk
...
```

3. Restart the systemd-udevd service:

sudo systemctl restart systemd-udevd

After udev processes the rules files, the symbolic link /dev/my disk has been added:

sudo ls -F /dev/sd* /dev/my_disk

/dev/my_disk@ /dev/sda /dev/sda1 /dev/sda2 /dev/sdb

To undo the changes, remove /etc/udev/rules.d/10-local.rules and /dev/my_disk and run systemctl restart systemd-udevd again.

5 Kernel Modules

WARNING:

Oracle Linux 7 is now in Extended Support. See Oracle Linux Extended Support and Oracle Open Source Support Policies for more information.

Migrate applications and data to Oracle Linux 8 or Oracle Linux 9 as soon as possible.

This chapter describes how to load, unload, and modify the behavior of kernel modules.

About Kernel Modules

The boot loader loads the kernel into memory. You can add new code to the kernel by including the source files in the kernel source tree and recompiling the kernel. Kernel modules, which can be dynamically loaded and unloaded on demand, provide device drivers that allow the kernel to access new hardware, support different file system types, and extend its functionality in other ways. To avoid wasting memory on unused device drivers, Oracle Linux supports loadable kernel modules (LKMs), which allow a system to run with only the device drivers and kernel code that it requires loaded into memory.

Listing Information about Loaded Modules

Use the lsmod command to list the modules that are currently loaded into the kernel.

sudo lsmod		
Module	Size 1405	Used by
nls_utf8		
fuse	59164	0
tun	12079	0
autofs4	22739	3
•••		
ppdev	7901	0
parport_pc	21262	0
parport	33812	<pre>2 ppdev,parport_pc</pre>
•••		

Note:

This command produces its output by reading the /proc/modules file.

The output shows the module name, the amount of memory it uses, the number of processes using the module and the names of other modules on which it depends. In the sample output,



the module parport depends on the modules ppdev and parport_pc, which are loaded in advance of parport. Two processes are currently using all three modules.

To display detailed information about a module, use the modinfo command, for example:

```
sudo modinfo ahci
```

```
/lib/modules/2.6.32-300.27.1.el6uek.x86 64/kernel/drivers/ata/ahci.ko
filename:
version:
                 3.0
license:
                GPL
description: AHCI SATA low-level driver
author: Jeff Garzik
srcversion: AC5EC885397BF332DE16389
               pci:v*d*sv*sd*bc01sc06i01*
alias:
. . .
depends:
             2.6.32-300.27.1.el6uek.x86_64 SMP mod_unload modversions
skip host reset:skip global host reset (0=don't skip, 1=:
vermagic:
parm:
                skip host reset:skip global host reset (0=don't skip, 1=skip) (int)
                ignore sss:Ignore staggered spinup flag (0=don't ignore, 1=ignore) (int)
parm:
. . .
```

The output includes the following information:

filename

Absolute path of the kernel object file.

version

Version number of the module.

description

Short description of the module.

srcversion

Hash of the source code used to create the module.

alias

Internal alias names for the module.

depends

Comma-separated list of any modules on which this module depends.

vermagic

Kernel version that was used to compile the module, which is checked against the current kernel when the module is loaded.

parm

Module parameters and descriptions.

Modules are loaded into the kernel from kernel object (ko) files in the /lib/modules/ kernel_version/kernel directory. To display the absolute path of a kernel object file, specify the -n option, for example:

sudo modinfo -n parport

/lib/modules/2.6.32-300.27.1.el6uek.x86 64/kernel/drivers/parport/parport.ko

For more information, see the lsmod(5) and modinfo(8) manual pages.



Loading and Unloading Modules

The modprobe command loads kernel modules, for example:

sudo modprobe nfs
sudo lsmod | grep nfs

nfs 266415 0
lockd 66530 1 nfs
fscache 41704 1 nfs
nfs_acl 2477 1 nfs
auth_rpcgss 38976 1 nfs
sunrpc 204268 5 nfs,lockd,nfs acl,auth rpcgss

Use the -v verbose option to show if any additional modules are loaded to resolve dependencies.

```
sudo modprobe -v nfs
```

```
insmod /lib/modules/2.6.32-300.27.1.el6uek.x86_64/kernel/net/sunrpc/auth_gss/
auth_rpcgss.ko
insmod /lib/modules/2.6.32-300.27.1.el6uek.x86_64/kernel/fs/nfs_common/nfs_acl.ko
insmod /lib/modules/2.6.32-300.27.1.el6uek.x86_64/kernel/fs/fscache/fscache.ko
...
```

To determine the dependencies, the modprobe command queries the /lib/modules/ kernel_version/modules.dep file, which the depmod utility creates when you install kernel modules.

Note:

modprobe does not reload modules that are already loaded. You must first unload a module before you can load it again.

Use the -r option to unload kernel modules, for example:

```
sudo modprobe -rv nfs
rmmod /lib/modules/2.6.32-300.27.1.el6uek.x86_64/kernel/fs/nfs/nfs.ko
rmmod /lib/modules/2.6.32-300.27.1.el6uek.x86_64/kernel/fs/lockd/lockd.ko
rmmod /lib/modules/2.6.32-300.27.1.el6uek.x86_64/kernel/fs/fscache/fscache.ko
...
```

Modules are unloaded in the reverse order that they were loaded. Modules are not unloaded if a process or another loaded module requires them.

Note:

modprobe uses the insmod and rmmod utilities to load and unload modules. As insmod and rmmod do not resolve module dependencies, do not use these utilities.

For more information, see the modprobe(8) and modules.dep(5) manual pages.



About Module Parameters

Modules accept parameters that you can specify using modprobe to modify a module's behavior:

sudo modprobe module name parameter=value ...

Use spaces to separate multiple parameter/value pairs. Array values are represented by a comma-separated list, for example:

sudo modprobe foo arrayparm=1,2,3,4

You can also change the values of some parameters for loaded modules and built-in drivers by writing the new value to a file under /sys/module/module name/parameters, for example:

echo 0 > /sys/module/ahci/parameters/skip_host_reset

The /etc/modprobe.d directory contains .conf configuration files specify module options, create module aliases, and override the usual behavior of modprobe for modules with special requirements. The /etc/modprobe.conf file that was used with earlier versions of modprobe is also valid if it exists. Entries in the /etc/modprobe.conf and /etc/modprobe.d/*.conf files use the same syntax.

The following are commonly used commands in modprobe configuration files:

alias

Creates an alternate name for a module. The alias can include shell wildcards. For example, create an alias for the sd-mod module:

alias block-major-8-* sd mod

As a result, a command such as modprobe block-major-8-0 has the same effect as modprobe sd mod.

blacklist

Ignore a module's internal alias that is displayed by the modinfo command. This command is typically used if the associated hardware is not required, if two or more modules both support the same devices, or if a module invalidly claims to support a device. For example, to blocklist the alias for the frame-buffer driver cirrusfb:

blacklist cirrusfb

The /etc/modprobe.d/blacklist.conf file prevents hotplug scripts from loading a module, usually so that a different driver binds the module instead, regardless of which driver happens to be probed first.

install

Runs a shell command instead of loading a module into the kernel. For example, load the module snd-emu10k1-synth instead of snd-emu10k1:

```
install snd-emul0k1 /sbin/modprobe --ignore-install snd-emul0k1 && \ /sbin/modprobe snd-emul0k1-synth
```

options

Defines options for a module,. For example, define the nohwcrypt and qos options for the b43 module:



options b43 nohwcrypt=1 qos=0

remove

Runs a shell command instead of unloading a module. For example, unmount /proc/fs/nfsd before unloading the nfsd module:

```
remove nfsd { /bin/umount /proc/fs/nfsd > /dev/null 2>&1 || :; } ; \
/sbin/modprobe -r --first-time --ignore-remove nfsd
```

For more information, see the modprobe.conf(5) manual page.

Specifying Modules To Be Loaded at Boot Time

The system loads most modules automatically at boot time. If necessary, you can specify an additional module that should be loaded.

To specify a module to be loaded at boot time:

- Create a file in the /etc/sysconfig/modules directory. The file name must have the extension .modules, for example foo.modules.
- 2. Edit the file to create the script that loads the module.

The script to load a module can be a simple modprobe call, for example:

#!/bin/sh
modprobe foo

or more complex to include error handling:

```
#!/bin/sh
if [ ! -c /dev/foo ] ; then
    exec /sbin/modprobe foo > /dev/null 2>&1
fi
```

3. Use the following command to make the script executable:

sudo chmod 755 /etc/sysconfig/modules/foo.modules

About Weak Update Modules

External modules, such as drivers installed using a driver update disk, are usually installed into /lib/modules/kernel-version/extra. Modules stored in this directory are given preference over matching modules included with the kernel, itself, when you attempt to load them. This means that external drivers and modules can be installed to override kernel modules where hardware issues may need resolution. For each subsequent kernel update, it is important that the external module is made available to each compatible kernel to avoid potential boot issues resulting from driver incompatibilities with the affected hardware.

Since the requirement to load the external module with each compatible kernel update is system critical, a mechanism is in place so that external modules can be loaded as weak update modules for compatible kernels. Weak update modules are made available by creating symbolic links to compatible modules in the /lib/modules/kernel-version/weak-updates directory. The package manager handles this process automatically when it detects driver modules installed in any /lib/modules/kernel-version/extra directories for compatible kernels. For example, installation of the kmod-megaraid_sas-uek driver update package on the Driver Update Disk (DUD) for the Oracle Linux 7.4 might install the following:

/lib/modules/4.1.12-61.1.18.el7uek.x86_64/extra/megaraid_sas
/lib/modules/4.1.12-61.1.18.el7uek.x86_64/extra/megaraid_sas/megaraid_sas.ko



The new driver module is installed into the extra directory for the 4.1.12-61.1.18.el7uek.x86_64, which was the kernel version that was originally used to build the module.

A subsequent kernel update means that the system is now running the 4.1.12-112.14.13.el7uek.x86_64 version of the kernel. This kernel is compatible with the module installed for the previous kernel, so the external module is automatically added, as a symbolic link, in the weak-updates directory as part of the installation process:

```
ls -1 /lib/modules/4.1.12-112.14.13.el7uek.x86_64/weak-updates/megaraid_sas/*.ko
```

```
lrwxrwxrwx. 1 root root 76 Jan 30 04:52 /lib/modules/4.1.12-112.14.13.el7uek.x86_64\
    /weak-updates/megaraid_sas/megaraid_sas.ko \
    -> /lib/modules/4.1.12-61.1.18.el7uek.x86 64/extra/megaraid_sas/megaraid_sas.ko
```

The output means that the external module is loaded for subsequent kernel updates.

In most cases, weak updates make sense and ensure that no extra work must be done to carry an external module through subsequent kernel updates. This prevents possible driver related boot issues after kernel upgrades and maintains the predictable running of a system and its hardware.

In some cases you may wish to remove weak update modules for a newer kernel. For instance, if an issue has been resolved for a driver that is shipped in the newer kernel and you would prefer to use this driver over the external module that you installed as part of a driver update.

You can remove weak update modules by removing the symbolic links for each kernel, manually. For example:

sudo rm -rf /lib/modules/4.1.12-112.14.13.el7uek.x86_64/weak-updates/megaraid_sas/

For more information about external driver modules and driver update disks, see Oracle Linux 7: Installation Guide.