systemd(1) - Linux manual page

SYSTEMD(1) systemd *SYSTEMD*(1)

NAME top

systemd, init - systemd system and service manager

SYNOPSIS top

```
/usr/lib/systemd/systemd [OPTIONS...]

init [OPTIONS...]
```

DESCRIPTION top

systemd is a system and service manager for Linux operating systems. When run as first process on boot (as PID 1), it acts as init system that brings up and maintains userspace services. Separate instances are started for logged-in users to start their services.

systemd is usually not invoked directly by the user, but is installed as the /sbin/init symlink and started during early boot. The user manager instances are started automatically through the user@.service(5) service.

When run as a system instance, systemd interprets the configuration file system.conf and the files in system.conf.d directories; when run as a user instance, systemd interprets the configuration file user.conf and the files in user.conf.d directories. See systemd-system.conf(5) for more information.

systemd contains native implementations of various tasks that need to be executed as part of the boot process. For example, it sets the hostname or configures the loopback network device. It also sets up and mounts various API file systems, such as /sys/, /proc/, and /dev/.

systemd will also reset the system clock during early boot if it appears to be set incorrectly. See "System clock epoch" section below.

Note that some but not all interfaces provided by systemd are covered by the Interface Portability and Stability Promise[1].

The D-Bus API of systemd is described in org.freedesktop.systemd1(5) and org.freedesktop.LogControl1(5).

Systems which invoke systemd in a container or initrd environment should implement the Container Interface[2] or initrd Interface[3] specifications, respectively.

UNITS top

systemd provides a dependency system between various entities called "units" of 11 different types. Units encapsulate various objects that are relevant for system boot-up and maintenance. The majority of units are configured in unit configuration files, whose syntax and basic set of options is described in systemd.unit(5), however some are created automatically from other configuration files, dynamically from system state or programmatically at runtime. Units may be in a number of states, described in the following table. Note that the various unit types may have a number of additional substates, which are mapped to the generalized unit states described here.

Table 1. Unit ACTIVE states

State	Description
active	Started, bound, plugged in, ..., depending on the unit type.
inactive	Stopped, unbound, unplugged, ..., depending on the unit type.
failed	Similar to inactive, but the unit failed in some way (process returned error code on exit,

	crashed, an operation timed out, or after too many restarts).
activating	Changing from inactive to active.
deactivating	Changing from active to inactive.
maintenance	Unit is inactive and a maintenance operation is in progress.
reloading	Unit is active and it is reloading its configuration.
refreshing	Unit is active and a new mount is being activated in its namespace.

The following unit types are available:

1. Service units, which start and control daemons and the processes they consist of. For details, see systemd.service(5).

2. Socket units, which encapsulate local IPC or network sockets in the system, useful for socket-based activation. For details about socket units, see systemd.socket(5), for details on socket-based activation and other forms of activation, see daemon(7).

3. Target units are useful to group units, or provide well-known synchronization points during boot-up, see systemd.target(5).

4. Device units expose kernel devices in systemd and may be used to implement device-based activation. For details, see systemd.device(5).

5. Mount units control mount points in the file system, for details see systemd.mount(5).

6. Automount units provide automount capabilities, for on-demand mounting of file systems as well as parallelized boot-up. See

systemd.automount(5).

7. Timer units are useful for triggering activation of other units based on timers. You may find details in systemd.timer(5).

8. Swap units are very similar to mount units and encapsulate memory swap partitions or files of the operating system. They are described in systemd.swap(5).

9. Path units may be used to activate other services when file system objects change or are modified. See systemd.path(5).

10. Slice units may be used to group units which manage system processes (such as service and scope units) in a hierarchical tree for resource management purposes. See systemd.slice(5).

11. Scope units are similar to service units, but manage foreign processes instead of starting them as well. See systemd.scope(5).

Units are named as their configuration files. Some units have special semantics. A detailed list is available in systemd.special(7).

systemd knows various kinds of dependencies, including positive and negative requirement dependencies (i.e. *Requires=* and *Conflicts=*) as well as ordering dependencies (*After=* and *Before=*). NB: ordering and requirement dependencies are orthogonal. If only a requirement dependency exists between two units (e.g. foo.service requires bar.service), but no ordering dependency (e.g. foo.service after bar.service) and both are requested to start, they will be started in parallel. It is a common pattern that both requirement and ordering dependencies are placed between two units. Also note that the majority of dependencies are implicitly created and maintained by systemd. In most cases, it should be unnecessary to declare additional dependencies manually, however it is possible to do this.

Application programs and units (via dependencies) may request state changes of units. In systemd, these requests are encapsulated as 'jobs' and maintained in a job queue. Jobs may succeed or can fail, their execution is ordered based on the ordering dependencies of the units they have been scheduled for.

On boot systemd activates the target unit default.target whose job is to activate on-boot services and other on-boot units by pulling

them in via dependencies. Usually, the unit name is just an alias (symlink) for either graphical.target (for fully-featured boots into the UI) or multi-user.target (for limited console-only boots for use in embedded or server environments, or similar; a subset of graphical.target). However, it is at the discretion of the administrator to configure it as an alias to any other target unit. See systemd.special(7) for details about these target units.

On first boot, systemd will enable or disable units according to preset policy. See systemd.preset(5) and "First Boot Semantics" in machine-id(5).

systemd only keeps a minimal set of units loaded into memory. Specifically, the only units that are kept loaded into memory are those for which at least one of the following conditions is true:

1. It is in an active, activating, deactivating or failed state (i.e. in any unit state except for "inactive")

2. It has a job queued for it

3. It is a dependency of at least one other unit that is loaded into memory

4. It has some form of resource still allocated (e.g. a service unit that is inactive but for which a process is still lingering that ignored the request to be terminated)

5. It has been pinned into memory programmatically by a D-Bus call

systemd will automatically and implicitly load units from disk — if they are not loaded yet — as soon as operations are requested for them. Thus, in many respects, the fact whether a unit is loaded or not is invisible to clients. Use systemctl list-units --all to comprehensively list all units currently loaded. Any unit for which none of the conditions above applies is promptly unloaded. Note that when a unit is unloaded from memory its accounting data is flushed out too. However, this data is generally not lost, as a journal log record is generated declaring the consumed resources whenever a unit shuts down.

Processes systemd spawns are placed in individual Linux control groups named after the unit which they belong to in the private systemd hierarchy. (see Control Groups v2[4] for more information about control groups, or short "cgroups"). systemd uses this to effectively keep track of processes. Control group information is

maintained in the kernel, and is accessible via the file system hierarchy (beneath /sys/fs/cgroup/), or in tools such as systemd-cgls(1) or ps(1) (ps xawf -eo pid,user,cgroup,args is particularly useful to list all processes and the systemd units they belong to.).

systemd is compatible with various established Unix functionality such as /etc/fstab or the utmp database.

systemd has a minimal transaction system: if a unit is requested to start up or shut down it will add it and all its dependencies to a temporary transaction. Then, it will verify if the transaction is consistent (i.e. whether the ordering of all units is cycle-free). If it is not, systemd will try to fix it up, and removes non-essential jobs from the transaction that might remove the loop. Also, systemd tries to suppress non-essential jobs in the transaction that would stop a running service. Finally it is checked whether the jobs of the transaction contradict jobs that have already been queued, and optionally the transaction is aborted then. If all worked out and the transaction is consistent and minimized in its impact it is merged with all already outstanding jobs and added to the run queue. Effectively this means that before executing a requested operation, systemd will verify that it makes sense, fixing it if possible, and only failing if it really cannot work.

Note that transactions are generated independently of a unit's state at runtime, hence, for example, if a start job is requested on an already started unit, it will still generate a transaction and wake up any inactive dependencies (and cause propagation of other jobs as per the defined relationships). This is because the enqueued job is at the time of execution compared to the target unit's state and is marked successful and complete when both satisfy. However, this job also pulls in other dependencies due to the defined relationships and thus leads to, in our example, start jobs for any of those inactive units getting queued as well.

Units may be generated dynamically at boot and system manager reload time, for example based on other configuration files or parameters passed on the kernel command line. For details, see systemd.generator(7).

DIRECTORIES top

System unit directories
    The systemd system manager reads unit configuration from

```
        various directories. Packages that want to install unit files
        shall place them in the directory returned by pkg-config
        systemd --variable=systemdsystemunitdir. Other directories
        checked are /usr/local/lib/systemd/system and
        /usr/lib/systemd/system. User configuration always takes
        precedence.  pkg-config systemd
        --variable=systemdsystemconfdir returns the path of the system
        configuration directory. Packages should alter the content of
        these directories only with the enable and disable commands of
        the systemctl(1) tool. Full list of directories is provided in
        systemd.unit(5).

    User unit directories
        Similar rules apply for the user unit directories. However,
        here the XDG Base Directory specification[5] is followed to
        find units. Applications should place their unit files in the
        directory returned by pkg-config systemd
        --variable=systemduserunitdir. Global configuration is done in
        the directory reported by pkg-config systemd
        --variable=systemduserconfdir. The enable and disable commands
        of the systemctl(1) tool can handle both global (i.e. for all
        users) and private (for one user) enabling/disabling of units.
        Full list of directories is provided in systemd.unit(5).
```

SIGNALS top

```
    The service listens to various UNIX process signals that can be
    used to request various actions asynchronously. The signal
    handling is enabled very early during boot, before any further
    processes are invoked. However, a supervising container manager or
    similar that intends to request these operations via this
    mechanism must take into consideration that this functionality is
    not available during the earliest initialization phase. An
    sd_notify() notification message carrying the
    X_SYSTEMD_SIGNALS_LEVEL=2 field is emitted once the signal
    handlers are enabled, see below. This may be used to schedule
    submission of these signals correctly.

    SIGTERM
        Upon receiving this signal the systemd system manager
        serializes its state, reexecutes itself and deserializes the
        saved state again. This is mostly equivalent to systemctl
        daemon-reexec.

        systemd user managers will start the exit.target unit when
        this signal is received. This is mostly equivalent to
```

```
        systemctl --user start exit.target
        --job-mode=replace-irreversibly.

    SIGINT
        Upon receiving this signal the systemd system manager will
        start the ctrl-alt-del.target unit. This is mostly equivalent
        to systemctl start ctrl-alt-del.target
        --job-mode=replace-irreversibly. If this signal is received
        more than 7 times per 2s, an immediate reboot is triggered.
        Note that pressing Ctrl+Alt+Del on the console will trigger
        this signal. Hence, if a reboot is hanging, pressing
        Ctrl+Alt+Del more than 7 times in 2 seconds is a relatively
        safe way to trigger an immediate reboot.

        systemd user managers treat this signal the same way as
        SIGTERM.

    SIGWINCH
        When this signal is received the systemd system manager will
        start the kbrequest.target unit. This is mostly equivalent to
        systemctl start kbrequest.target.

        This signal is ignored by systemd user managers.

    SIGPWR
        When this signal is received the systemd manager will start
        the sigpwr.target unit. This is mostly equivalent to systemctl
        start sigpwr.target.

    SIGUSR1
        When this signal is received the systemd manager will try to
        reconnect to the D-Bus bus.

    SIGUSR2
        When this signal is received the systemd manager will log its
        complete state in human-readable form. The data logged is the
        same as printed by systemd-analyze dump.

    SIGHUP
        Reloads the complete daemon configuration. This is mostly
        equivalent to systemctl daemon-reload.

    SIGRTMIN+0
        Enters default mode, starts the default.target unit. This is
        mostly equivalent to systemctl isolate default.target.

    SIGRTMIN+1
```

```
        Enters rescue mode, starts the rescue.target unit. This is
        mostly equivalent to systemctl isolate rescue.target.

    SIGRTMIN+2
        Enters emergency mode, starts the emergency.service unit. This
        is mostly equivalent to systemctl isolate emergency.service.

    SIGRTMIN+3
        Halts the machine, starts the halt.target unit. This is mostly
        equivalent to systemctl start halt.target
        --job-mode=replace-irreversibly.

    SIGRTMIN+4
        Powers off the machine, starts the poweroff.target unit. This
        is mostly equivalent to systemctl start poweroff.target
        --job-mode=replace-irreversibly.

    SIGRTMIN+5
        Reboots the machine, starts the reboot.target unit. This is
        mostly equivalent to systemctl start reboot.target
        --job-mode=replace-irreversibly.

    SIGRTMIN+6
        Reboots the machine via kexec, starts the kexec.target unit.
        This is mostly equivalent to systemctl start kexec.target
        --job-mode=replace-irreversibly.

    SIGRTMIN+7
        Reboots userspace, starts the soft-reboot.target unit. This is
        mostly equivalent to systemctl start soft-reboot.target
        --job-mode=replace-irreversibly.

        Added in version 254.

    SIGRTMIN+13
        Immediately halts the machine.

    SIGRTMIN+14
        Immediately powers off the machine.

    SIGRTMIN+15
        Immediately reboots the machine.

    SIGRTMIN+16
        Immediately reboots the machine with kexec.

    SIGRTMIN+17
```

Immediately reboots the userspace.

Added in version 254.

SIGRTMIN+20
Enables display of status messages on the console, as controlled via *systemd.show_status=1* on the kernel command line.

You may want to use SetShowStatus() instead of SIGRTMIN+20 in order to prevent race conditions. See org.freedesktop.systemd1(5).

SIGRTMIN+21
Disables display of status messages on the console, as controlled via *systemd.show_status=0* on the kernel command line.

You may want to use SetShowStatus() instead of SIGRTMIN+21 in order to prevent race conditions. See org.freedesktop.systemd1(5).

SIGRTMIN+22
Sets the service manager's log level to "debug", in a fashion equivalent to *systemd.log_level=debug* on the kernel command line.

SIGRTMIN+23
Restores the log level to its configured value. The configured value is derived from – in order of priority – the value specified with *systemd.log-level=* on the kernel command line, or the value specified with LogLevel= in the configuration file, or the built-in default of "info".

Added in version 239.

SIGRTMIN+24
Immediately exits the manager (only available for --user instances).

Added in version 195.

SIGRTMIN+25
Upon receiving this signal the systemd manager will reexecute itself. This is mostly equivalent to systemctl daemon-reexec except that it will be done asynchronously.

The systemd system manager treats this signal the same way as SIGTERM.

Added in version 250.

SIGRTMIN+26
Restores the log target to its configured value. The configured value is derived from – in order of priority – the value specified with *systemd.log-target=* on the kernel command line, or the value specified with LogTarget= in the configuration file, or the built-in default.

Added in version 239.

SIGRTMIN+27, SIGRTMIN+28
Sets the log target to "console" on SIGRTMIN+27 (or "kmsg" on SIGRTMIN+28), in a fashion equivalent to *systemd.log_target=console* (or *systemd.log_target=kmsg* on SIGRTMIN+28) on the kernel command line.

Added in version 239.

ENVIRONMENT top

The environment block for the system manager is initially set by the kernel. (In particular, "key=value" assignments on the kernel command line are turned into environment variables for PID 1). For the user manager, the system manager sets the environment as described in the "Environment Variables in Spawned Processes" section of systemd.exec(5). The *DefaultEnvironment=* setting in the system manager applies to all services including user@.service. Additional entries may be configured (as for any other service) through the *Environment=* and *EnvironmentFile=* settings for user@.service (see systemd.exec(5)). Also, additional environment variables may be set through the *ManagerEnvironment=* setting in systemd-system.conf(5) and systemd-user.conf(5).

Some of the variables understood by systemd:

$SYSTEMD_LOG_LEVEL
The maximum log level of emitted messages (messages with a higher log level, i.e. less important ones, will be suppressed). Takes a comma-separated list of values. A value may be either one of (in order of decreasing importance) emerg, alert, crit, err, warning, notice, info, debug, or an integer in the range 0...7. See syslog(3) for more

information. Each value may optionally be prefixed with one of console, syslog, kmsg or journal followed by a colon to set the maximum log level for that specific log target (e.g. SYSTEMD_LOG_LEVEL=debug,console:info specifies to log at debug level except when logging to the console which should be at info level). Note that the global maximum log level takes priority over any per target maximum log levels.

This can be overridden with --log-level=.

$SYSTEMD_LOG_COLOR
A boolean. If true, messages written to the tty will be colored according to priority.

This can be overridden with --log-color=.

$SYSTEMD_LOG_TIME
A boolean. If true, console log messages will be prefixed with a timestamp.

This can be overridden with --log-time=.

Added in version 246.

$SYSTEMD_LOG_LOCATION
A boolean. If true, messages will be prefixed with a filename and line number in the source code where the message originates.

This can be overridden with --log-location=.

$SYSTEMD_LOG_TID
A boolean. If true, messages will be prefixed with the current numerical thread ID (TID).

Added in version 247.

$SYSTEMD_LOG_TARGET
The destination for log messages. One of console (log to the attached tty), console-prefixed (log to the attached tty but with prefixes encoding the log level and "facility", see syslog(3), kmsg (log to the kernel circular log buffer), journal (log to the journal), journal-or-kmsg (log to the journal if available, and to kmsg otherwise), auto (determine the appropriate log target automatically, the default), null (disable log output).

This can be overridden with --log-target=.

$SYSTEMD_LOG_RATELIMIT_KMSG
Whether to ratelimit kmsg or not. Takes a boolean. Defaults to "true". If disabled, systemd will not ratelimit messages written to kmsg.

Added in version 254.

$XDG_CONFIG_HOME, *$XDG_CONFIG_DIRS*, *$XDG_DATA_HOME*, *$XDG_DATA_DIRS*
The systemd user manager uses these variables in accordance to the XDG Base Directory specification[5] to find its configuration.

$SYSTEMD_UNIT_PATH, *$SYSTEMD_GENERATOR_PATH*, *$SYSTEMD_ENVIRONMENT_GENERATOR_PATH*
Controls where systemd looks for unit files and generators.

These variables may contain a list of paths, separated by colons (":"). When set, if the list ends with an empty component ("...:"), this list is prepended to the usual set of paths. Otherwise, the specified list replaces the usual set of paths.

$SYSTEMD_PAGER, *$PAGER*
Pager to use when --no-pager is not given. *$SYSTEMD_PAGER* is used if set; otherwise *$PAGER* is used. If neither *$SYSTEMD_PAGER* nor *$PAGER* are set, a set of well-known pager implementations is tried in turn, including less(1) and more(1), until one is found. If no pager implementation is discovered, no pager is invoked. Setting those environment variables to an empty string or the value "cat" is equivalent to passing --no-pager.

Note: if *$SYSTEMD_PAGERSECURE* is not set, *$SYSTEMD_PAGER* and *$PAGER* can only be used to disable the pager (with "cat" or ""), and are otherwise ignored.

$SYSTEMD_LESS
Override the options passed to less (by default "FRSXMK").

Users might want to change two options in particular:

K
This option instructs the pager to exit immediately when Ctrl+C is pressed. To allow less to handle Ctrl+C itself to switch back to the pager command prompt, unset this

option.

If the value of *$SYSTEMD_LESS* does not include "K", and the pager that is invoked is less, Ctrl+C will be ignored by the executable, and needs to be handled by the pager.

X
This option instructs the pager to not send termcap initialization and deinitialization strings to the terminal. It is set by default to allow command output to remain visible in the terminal even after the pager exits. Nevertheless, this prevents some pager functionality from working, in particular paged output cannot be scrolled with the mouse.

Note that setting the regular *$LESS* environment variable has no effect for less invocations by systemd tools.

See less(1) for more discussion.

$SYSTEMD_LESSCHARSET
Override the charset passed to less (by default "utf-8", if the invoking terminal is determined to be UTF-8 compatible).

Note that setting the regular *$LESSCHARSET* environment variable has no effect for less invocations by systemd tools.

$SYSTEMD_PAGERSECURE
Common pager commands like less(1), in addition to "paging", i.e. scrolling through the output, support opening of or writing to other files and running arbitrary shell commands. When commands are invoked with elevated privileges, for example under sudo(8) or pkexec(1), the pager becomes a security boundary. Care must be taken that only programs with strictly limited functionality are used as pagers, and unintended interactive features like opening or creation of new files or starting of subprocesses are not allowed. "Secure mode" for the pager may be enabled as described below, *if the pager supports that* (most pagers are not written in a way that takes this into consideration). It is recommended to either explicitly enable "secure mode" or to completely disable the pager using --no-pager or *PAGER=cat* when allowing untrusted users to execute commands with elevated privileges.

This option takes a boolean argument. When set to true, the "secure mode" of the pager is enabled. In "secure mode", LESSSECURE=1 will be set when invoking the pager, which

instructs the pager to disable commands that open or create new files or start new subprocesses. Currently only less(1) is known to understand this variable and implement "secure mode".

When set to false, no limitation is placed on the pager. Setting *SYSTEMD_PAGERSECURE=0* or not removing it from the inherited environment may allow the user to invoke arbitrary commands.

When *$SYSTEMD_PAGERSECURE* is not set, systemd tools attempt to automatically figure out if "secure mode" should be enabled and whether the pager supports it. "Secure mode" is enabled if the effective UID is not the same as the owner of the login session, see geteuid(2) and sd_pid_get_owner_uid(3), or when running under sudo(8) or similar tools (*$SUDO_UID* is set [6]). In those cases, *SYSTEMD_PAGERSECURE=1* will be set and pagers which are not known to implement "secure mode" will not be used at all. Note that this autodetection only covers the most common mechanisms to elevate privileges and is intended as convenience. It is recommended to explicitly set *$SYSTEMD_PAGERSECURE* or disable the pager.

Note that if the *$SYSTEMD_PAGER* or *$PAGER* variables are to be honoured, other than to disable the pager, *$SYSTEMD_PAGERSECURE* must be set too.

$SYSTEMD_COLORS

Takes a boolean argument. When true, systemd and related utilities will use colors in their output, otherwise the output will be monochrome. Additionally, the variable can take one of the following special values: "16", "256" to restrict the use of colors to the base 16 or 256 ANSI colors, respectively. This can be specified to override the automatic decision based on *$TERM* and what the console is connected to.

$SYSTEMD_URLIFY

The value must be a boolean. Controls whether clickable links should be generated in the output for terminal emulators supporting this. This can be specified to override the decision that systemd makes based on *$TERM* and other conditions.

$LISTEN_PID, *$LISTEN_PIDFDID*, *$LISTEN_FDS*, *$LISTEN_FDNAMES*

Set by systemd for supervised processes during socket-based activation. See sd_listen_fds(3) for more information.

$NOTIFY_SOCKET

Set by service manager for its services for status and readiness notifications. Also consumed by service manager for notifying supervising container managers or service managers up the stack about its own progress. See sd_notify(3) and the relevant section below for more information.

For further environment variables understood by systemd and its various components, see Known Environment Variables[7].

KERNEL COMMAND LINE top

When run as the system instance, systemd parses a number of options listed below. They can be specified as kernel command line arguments which are parsed from a number of sources depending on the environment in which systemd is executed. If run inside a Linux container, these options are parsed from the command line arguments passed to systemd itself, next to any of the command line options listed in the Options section above. If run outside of Linux containers, these arguments are parsed from /proc/cmdline instead.

The following variables are understood:

systemd.unit=, *rd.systemd.unit=*
Overrides the unit to activate on boot. Defaults to default.target. This may be used to temporarily boot into a different boot unit, for example rescue.target or emergency.service. See systemd.special(7) for details about these units. The option prefixed with "rd." is honored only in the initrd, while the one that is not prefixed only in the main system.

systemd.dump_core
Takes a boolean argument or enables the option if specified without an argument. If enabled, the systemd manager (PID 1) dumps core when it crashes. Otherwise, no core dump is created. Defaults to enabled.

Added in version 233.

systemd.crash_chvt
Takes a positive integer, or a boolean argument. Can be also specified without an argument, with the same effect as a positive boolean. If a positive integer (in the range 1–63) is specified, the system manager (PID 1) will activate the specified virtual terminal when it crashes. Defaults to

disabled, meaning that no such switch is attempted. If set to enabled, the virtual terminal the kernel messages are written to is used instead.

Added in version 233.

systemd.crash_shell
Takes a boolean argument or enables the option if specified without an argument. If enabled, the system manager (PID 1) spawns a shell when it crashes. Otherwise, no shell is spawned. Defaults to disabled, for security reasons, as the shell is not protected by password authentication.

Added in version 233.

systemd.crash_action=
Takes one of "freeze", "reboot" or "poweroff". Defaults to "freeze". If set to "freeze", the system will hang indefinitely when the system manager (PID 1) crashes. If set to "reboot", the system manager (PID 1) will reboot the machine automatically when it crashes, after a 10s delay. If set to "poweroff", the system manager (PID 1) will power off the machine immediately when it crashes. If combined with *systemd.crash_shell*, the configured crash action is executed after the shell exits.

Added in version 256.

systemd.confirm_spawn
Takes a boolean argument or a path to the virtual console where the confirmation messages should be emitted. Can be also specified without an argument, with the same effect as a positive boolean. If enabled, the system manager (PID 1) asks for confirmation when spawning processes using /dev/console. If a path or a console name (such as "ttyS0") is provided, the virtual console pointed to by this path or described by the give name will be used instead. Defaults to disabled.

Added in version 233.

systemd.service_watchdogs=
Takes a boolean argument. If disabled, all service runtime watchdogs (WatchdogSec=) and emergency actions (e.g. OnFailure= or StartLimitAction=) are ignored by the system manager (PID 1); see systemd.service(5). Defaults to enabled, i.e. watchdogs and failure actions are processed normally. The hardware watchdog is not affected by this option.

Added in version 237.

systemd.show_status
Takes a boolean argument or the constants error and auto. Can be also specified without an argument, with the same effect as a positive boolean. If enabled, the systemd manager (PID 1) shows terse service status updates on the console during bootup. With error, only messages about failures are shown, but boot is otherwise quiet. auto behaves like false until there is a significant delay in boot. Defaults to enabled, unless quiet is passed as kernel command line option, in which case it defaults to error. If specified overrides the system manager configuration file option ShowStatus=, see systemd-system.conf(5).

Added in version 233.

systemd.status_unit_format=
Takes name, description or combined as the value. If name, the system manager will use unit names in status messages. If combined, the system manager will use unit names and description in status messages. When specified, overrides the system manager configuration file option StatusUnitFormat=, see systemd-system.conf(5).

Added in version 243.

systemd.log_color, *systemd.log_level=*, *systemd.log_location*, *systemd.log_target=*, *systemd.log_time*, *systemd.log_tid*, *systemd.log_ratelimit_kmsg*
Controls log output, with the same effect as the *$SYSTEMD_LOG_COLOR*, *$SYSTEMD_LOG_LEVEL*, *$SYSTEMD_LOG_LOCATION*, *$SYSTEMD_LOG_TARGET*, *$SYSTEMD_LOG_TIME*, *$SYSTEMD_LOG_TID* and *$SYSTEMD_LOG_RATELIMIT_KMSG* environment variables described above. *systemd.log_color*, *systemd.log_location*, *systemd.log_time*, *systemd.log_tid* and *systemd.log_ratelimit_kmsg* can be specified without an argument, with the same effect as a positive boolean.

systemd.default_standard_output=, *systemd.default_standard_error=*
Controls default standard output and error output for services and sockets. That is, controls the default for StandardOutput= and StandardError= (see systemd.exec(5) for details). Takes one of inherit, null, tty, journal, journal+console, kmsg, kmsg+console. If the argument is omitted *systemd.default-standard-output=* defaults to journal and

systemd.default-standard-error= to inherit.

systemd.setenv=
Takes a string argument in the form VARIABLE=VALUE. May be used to set default environment variables to add to forked child processes. May be used more than once to set multiple variables.

systemd.machine_id=
Takes a 32 character hex value to be used for setting the machine-id. Intended mostly for network booting where the same machine-id is desired for every boot.

Added in version 229.

systemd.set_credential=, *systemd.set_credential_binary=*
Sets a system credential, which can then be propagated to system services using the *ImportCredential=* or *LoadCredential=* setting, see systemd.exec(5) for details. Takes a pair of credential name and value, separated by a colon. The *systemd.set_credential=* parameter expects the credential value in literal text form, the *systemd.set_credential_binary=* parameter takes binary data encoded in Base64. Note that the kernel command line is typically accessible by unprivileged programs in /proc/cmdline. Thus, this mechanism is not suitable for transferring sensitive data. Use it only for data that is not sensitive (e.g. public keys/certificates, rather than private keys), or in testing/debugging environments.

For further information see System and Service Credentials[8] documentation.

Added in version 251.

systemd.import_credentials=
Takes a boolean argument. If false disables importing credentials from the kernel command line, the DMI/SMBIOS OEM string table, the qemu_fw_cfg subsystem or the EFI kernel stub.

Added in version 251.

quiet
Turn off status output at boot, much like *systemd.show_status=no* would. Note that this option is also read by the kernel itself and disables kernel log output. Passing this option hence turns off the usual output from both

the system manager and the kernel.

Added in version 186.

debug
Turn on debugging output. This is equivalent to *systemd.log_level=debug*. Note that this option is also read by the kernel itself and enables kernel debug output. Passing this option hence turns on the debug output from both the system manager and the kernel.

Added in version 205.

emergency, *rd.emergency*, *-b*
Boot into emergency mode. This is equivalent to *systemd.unit=emergency.target* or *rd.systemd.unit=emergency.target*, respectively, and provided for compatibility reasons and to be easier to type.

Added in version 186.

rescue, *rd.rescue*, *single*, *s*, *S*, *1*
Boot into rescue mode. This is equivalent to *systemd.unit=rescue.target* or *rd.systemd.unit=rescue.target*, respectively, and provided for compatibility reasons and to be easier to type.

Added in version 186.

2, *3*, *4*, *5*
Boot into the specified legacy SysV runlevel. *2*, *3*, and *4* are equivalent to *systemd.unit=multi-user.target*; and *5* is equivalent to *systemd.unit=graphical.target*, and provided for compatibility reasons and to be easier to type.

Added in version 186.

locale.LANG=, *locale.LANGUAGE=*, *locale.LC_CTYPE=*, *locale.LC_NUMERIC=*, *locale.LC_TIME=*, *locale.LC_COLLATE=*, *locale.LC_MONETARY=*, *locale.LC_MESSAGES=*, *locale.LC_PAPER=*, *locale.LC_NAME=*, *locale.LC_ADDRESS=*, *locale.LC_TELEPHONE=*, *locale.LC_MEASUREMENT=*, *locale.LC_IDENTIFICATION=*
Set the system locale to use. This overrides the settings in /etc/locale.conf. For more information, see locale.conf(5) and locale(7).

Added in version 186.

For other kernel command line parameters understood by components of the core OS, please refer to kernel-command-line(7).

SYSTEM CREDENTIALS top

During initialization the service manager will import credentials from various sources into the system's set of credentials, which can then be propagated into services and consumed by generators:

- When the service manager first initializes it will read system credentials from SMBIOS Type 11 vendor strings *io.systemd.credential:name=value*, and *io.systemd.credential.binary:name=value*.

- At the same time it will import credentials from QEMU "fw_cfg". (Note that the SMBIOS mechanism is generally preferred, because it is faster and generic.)

- Credentials may be passed via the kernel command line, using the *systemd.set-credential=* parameter, see above.

- Credentials may be passed from the UEFI environment via systemd-stub(7).

- When the service manager is invoked during the initrd → host transition it will import all files in /run/credentials/@initrd/ as system credentials.

Invoke systemd-creds(1) as follows to see the list of credentials passed into the system:

```
# systemd-creds --system list
```

For further information see System and Service Credentials[8] documentation.

The service manager when run as PID 1 consumes the following system credentials:

vmm.notify_socket
: Contains a AF_VSOCK or AF_UNIX address where to send a READY=1 notification message when the service manager has completed booting. See sd_notify(3) and the next section for more information. Note that in case the hypervisor does not support SOCK_DGRAM over AF_VSOCK, SOCK_SEQPACKET will be tried

instead. The credential payload for AF_VSOCK should be a string in the form "vsock:CID:PORT". "vsock-stream", "vsock-dgram" and "vsock-seqpacket" can be used instead of "vsock" to force usage of the corresponding socket type.

This feature is useful for machine managers or other processes on the host to receive a notification via VSOCK when a virtual machine has finished booting.

Added in version 254.

system.machine_id
Takes a 128bit hexadecimal ID to initialize /etc/machine-id from, if the file is not set up yet. See machine-id(5) for details.

Added in version 254.

For a list of system credentials various other components of systemd consume, see systemd.system-credentials(7).

READINESS PROTOCOL top

The service manager implements a readiness notification protocol both between the manager and its services (i.e. down the stack), and between the manager and a potential supervisor further up the stack (the latter could be a machine or container manager, or in case of a per-user service manager the system service manager instance). The basic protocol (and the suggested API for it) is described in sd_notify(3).

The notification socket the service manager (including PID 1) uses for reporting readiness to its own supervisor is set via the usual *$NOTIFY_SOCKET* environment variable (see above). Since this is directly settable only for container managers and for the per-user instance of the service manager, an additional mechanism to configure this is available, in particular intended for use in VM environments: the *vmm.notify_socket* system credential (see above) may be set to a suitable socket (typically an AF_VSOCK one) via SMBIOS Type 11 vendor strings. For details see above.

The notification protocol from the service manager up the stack towards a supervisor supports a number of extension fields that allow a supervisor to learn about specific properties of the system and track its boot progress. Specifically the following fields are sent:

- An *X_SYSTEMD_HOSTNAME=...* message will be sent out once the initial hostname for the system has been determined. Note that during later runtime the hostname might be changed again programmatically, and (currently) no further notifications are sent out in that case.

  Added in version 256.

- An *X_SYSTEMD_MACHINE_ID=...* message will be sent out once the machine ID of the system has been determined. See machine-id(5) for details.

  Added in version 256.

- An *X_SYSTEMD_SIGNALS_LEVEL=...* message will be sent out once the service manager installed the various UNIX process signal handlers described above. The field's value is an unsigned integer formatted as decimal string, and indicates the supported UNIX process signal feature level of the service manager. Currently, only a single feature level is defined:

  - *X_SYSTEMD_SIGNALS_LEVEL=2* covers the various UNIX process signals documented above – which are a superset of those supported by the historical SysV init system.

  Signals sent to PID 1 before this message is sent might not be handled correctly yet. A consumer of these messages should parse the value as an unsigned integer that indicates the level of support. For now only the mentioned level 2 is defined, but later on additional levels might be defined with higher integers, that will implement a superset of the currently defined behaviour.

  Added in version 256.

- *X_SYSTEMD_UNIT_ACTIVE=...* and *X_SYSTEMD_UNIT_INACTIVE=...* messages will be sent out for each target unit as it becomes active or stops being active. This is useful to track boot progress and functionality. For example, once the ssh-access.target unit is reported started SSH access is typically available, see systemd.special(7) for details.

  Added in version 256.

- An *X_SYSTEMD_SHUTDOWN=...* message will be sent out very shortly before the system shuts down. The value is one of the

strings "reboot", "halt", "poweroff", "kexec" and indicates which kind of shutdown is being executed.

Added in version 256.

- An *X_SYSTEMD_REBOOT_PARAMETER=...* message will also be sent out very shortly before the system shuts down. Its value is the reboot argument as configured with systemctl --reboot-argument=....

  Added in version 256.

Note that these extension fields are sent in addition to the regular "READY=1" and "RELOADING=1" notifications.

OPTIONS [top](#)

systemd is only very rarely invoked directly, since it is started early and is already running by the time users may interact with it. Normally, tools like systemctl(1) are used to give commands to the manager. Since systemd is usually not invoked directly, the options listed below are mostly useful for debugging and special purposes.

Introspection and debugging options

Those options are used for testing and introspection, and systemd may be invoked with them at any time:

--dump-configuration-items
: Dump understood unit configuration items. This outputs a terse but complete list of configuration items understood in unit definition files.

--dump-bus-properties
: Dump exposed bus properties. This outputs a terse but complete list of properties exposed on D-Bus.

  Added in version 239.

--test
: Determine the initial start-up transaction (i.e. the list of jobs enqueued at start-up), dump it and exit — without actually executing any of the determined jobs. This option is useful for debugging only. Note that during regular service manager start-up additional units not shown by this operation may be started, because hardware, socket, bus or other kinds

```
          of activation might add additional jobs as the transaction is
          executed. Use --system to request the initial transaction of
          the system service manager (this is also the implied default),
          combine with --user to request the initial transaction of the
          per-user service manager instead.

      --system, --user
          When used in conjunction with --test, selects whether to
          calculate the initial transaction for the system instance or
          for a per-user instance. These options have no effect when
          invoked without --test, as during regular (i.e. non---test)
          invocations the service manager will automatically detect
          whether it shall operate in system or per-user mode, by
          checking whether the PID it is run as is 1 or not. Note that
          it is not supported booting and maintaining a system with the
          service manager running in --system mode but with a PID other
          than 1.

      -h, --help
          Print a short help text and exit.

      --version
          Print a short version string and exit.

  Options that duplicate kernel command line settings
      Those options correspond directly to options listed above in
      "Kernel Command Line". Both forms may be used equivalently for the
      system manager, but it is recommended to use the forms listed
      above in this context, because they are properly namespaced. When
      an option is specified both on the kernel command line and as a
      normal command line argument, the latter has higher precedence.

      When systemd is used as a user manager, the kernel command line is
      ignored and only the options described below are understood.
      Nevertheless, systemd is usually started in this mode through the
      user@.service(5) service, which is shared between all users. It
      may be more convenient to use configuration files to modify
      settings (see systemd-user.conf(5)), or environment variables. See
      the "Environment" section above for a discussion of how the
      environment block is set.

      --unit=
          Set default unit to activate on startup. If not specified,
          defaults to default.target. See systemd.unit= above.

      --dump-core
          Enable core dumping on crash. This switch has no effect when
```

running as user instance. Same as *systemd.dump_core=* above.

--crash-vt=*VT*
Switch to a specific virtual console (VT) on crash. This switch has no effect when running as user instance. Same as *systemd.crash_chvt=* above (but not the different spelling!).

Added in version 227.

--crash-shell
Run a shell on crash. This switch has no effect when running as user instance. See *systemd.crash_shell=* above.

--crash-action=
Specify what to do when the system manager (PID 1) crashes. This switch has no effect when systemd is running as user instance. See *systemd.crash_action=* above.

Added in version 256.

--confirm-spawn
Ask for confirmation when spawning processes. This switch has no effect when run as user instance. See *systemd.confirm_spawn* above.

--show-status
Show terse unit status information on the console during boot-up and shutdown. See *systemd.show_status* above.

Added in version 244.

--log-color
Highlight important log messages. See *systemd.log_color* above.

Added in version 244.

--log-level=
Set log level. See *systemd.log_level* above.

--log-location
Include code location in log messages. See *systemd.log_location* above.

Added in version 244.

--log-target=
Set log target. See *systemd.log_target* above.

```
--log-time=
    Prefix console messages with timestamp. See systemd.log_time
    above.

    Added in version 246.

--machine-id=
    Override the machine-id set on the hard drive. See
    systemd.machine_id= above.

    Added in version 229.

--service-watchdogs
    Globally enable/disable all service watchdog timeouts and
    emergency actions. See systemd.service_watchdogs above.

    Added in version 237.

--default-standard-output=, --default-standard-error=
    Sets the default output or error output for all services and
    sockets, respectively. See systemd.default_standard_output=
    and systemd.default_standard_error= above.
```

SYSTEM CLOCK EPOCH [top]

When systemd is started or restarted, it may set the system clock to the "epoch". This mechanism is used to ensure that the system clock remains somewhat reasonably initialized and roughly monotonic across reboots, in case no battery-backed local RTC is available or it does not work correctly.

The epoch is the lowest date above which the system clock time is assumed to be set correctly. When initializing, the local clock is *advanced* to the epoch if it was set to a lower value. As a special case, if the local clock is sufficiently far in the future (by default 15 years, but this can be configured at build time), the hardware clock is assumed to be broken, and the system clock is *rewound* to the epoch.

The epoch is set to the highest of: the build time of systemd, the modification time ("mtime") of /usr/lib/clock-epoch, and the modification time of /var/lib/systemd/timesync/clock.

FILES [top]

```
/run/systemd/notify
    Daemon status notification socket. This is an AF_UNIX datagram
    socket and is used to implement the daemon notification logic
    as implemented by sd_notify(3).

/run/systemd/private
    Used internally as communication channel between systemctl(1)
    and the systemd process. This is an AF_UNIX stream socket.
    This interface is private to systemd and should not be used in
    external projects.

/usr/lib/clock-epoch
    The modification time ("mtime") of this file is used for the
    time epoch, see previous section.

    Added in version 247.

/var/lib/systemd/timesync/clock
    The modification time ("mtime") of this file is updated by
    systemd-timesyncd.service(8). If present, the modification
    time of file is used for the epoch, see previous section.

    Added in version 257.
```

HISTORY top

```
systemd 252
    Kernel command-line arguments systemd.unified_cgroup_hierarchy
    and systemd.legacy_systemd_cgroup_controller were deprecated.
    Please switch to the unified cgroup hierarchy.
```

SEE ALSO top

```
The systemd Homepage[9], systemd-system.conf(5), locale.conf(5),
systemctl(1), journalctl(1), systemd-notify(1), daemon(7),
sd-daemon(3), org.freedesktop.systemd1(5), systemd.unit(5),
systemd.special(7), pkg-config(1), kernel-command-line(7),
bootup(7), systemd.directives(7), org.freedesktop.systemd1(5)

For more information about the concepts and ideas behind systemd,
please refer to the Original Design Document[10].
```

NOTES top

1. Interface Portability and Stability Promise
   https://systemd.io/PORTABILITY_AND_STABILITY/

2. Container Interface
   https://systemd.io/CONTAINER_INTERFACE

3. initrd Interface
   https://systemd.io/INITRD_INTERFACE/

4. Control Groups v2
   https://docs.kernel.org/admin-guide/cgroup-v2.html

5. XDG Base Directory specification
   https://standards.freedesktop.org/basedir-spec/basedir-spec-latest.html

6. It is recommended for other tools to set and check *$SUDO_UID*
   as appropriate, treating it is a common interface.

7. Known Environment Variables
   https://systemd.io/ENVIRONMENT

8. System and Service Credentials
   https://systemd.io/CREDENTIALS

9. systemd Homepage
   https://systemd.io/

10. Original Design Document
    https://0pointer.de/blog/projects/systemd.html

COLOPHON top

This page is part of the *systemd* (systemd system and service manager) project. Information about the project can be found at ⟨http://www.freedesktop.org/wiki/Software/systemd⟩. If you have a bug report for this manual page, see ⟨http://www.freedesktop.org/wiki/Software/systemd/#bugreports⟩. This page was obtained from the project's upstream Git repository ⟨https://github.com/systemd/systemd.git⟩ on 2026-01-16. (At that time, the date of the most recent commit that was found in the repository was 2026-01-16.) If you discover any rendering problems in this HTML version of the page, or you believe there is a better or more up-to-date source for the page, or you have corrections or improvements to the information in this COLOPHON (which is *not* part of the original manual page), send a mail to

```
        man-pages@man7.org

systemd 260~devel                                                SYSTEMD(1)
```

Pages that refer to this page: busctl(1), homectl(1), hostnamectl(1), importctl(1), journalctl(1), localectl(1), logger(1), loginctl(1), machinectl(1), oomctl(1), pcp-check(1), pcp-geolocate(1), pcp-reboot-init(1), pmfind_check(1), pmie(1), pmie_check(1), pmlogctl(1), pmlogger(1), pmlogger_check(1), pmlogger_daily(1), pmlogger_janitor(1), pmproxy(1), pmseries_import(1), portablectl(1), resolvectl(1), run0(1), su(1), systemctl(1), systemd-ac-power(1), systemd-analyze(1), systemd-ask-password(1), systemd-cat(1), systemd-cgls(1), systemd-cgtop(1), systemd-creds(1), systemd-cryptenroll(1), systemd-delta(1), systemd-detect-virt(1), systemd-dissect(1), systemd-escape(1), systemd-firstboot(1), systemd-id128(1), systemd-inhibit(1), systemd-machine-id-setup(1), systemd-measure(1), systemd-mount(1), systemd-mute-console(1), systemd-notify(1), systemd-nspawn(1), systemd-path(1), systemd-run(1), systemd-socket-activate(1), systemd-ssh-issue(1), systemd-ssh-proxy(1), systemd-stdio-bridge(1), systemd-tty-ask-password-agent(1), systemd-vmspawn(1), systemd-vpick(1), timedatectl(1), ukify(1), updatectl(1), userdbctl(1), w(1), capget(2), _exit(2), getpid(2), KEYCTL_SESSION_TO_PARENT(2const), pivot_root(2), PR_SET_CHILD_SUBREAPER(2const), ptrace(2), reboot(2), unshare(2), vhangup(2), wait(2), libsystemd(3), libudev(3), __pmServerNotifyServiceManagerReady(3), sd_booted(3), sd-bus(3), sd_bus_add_match(3), sd_bus_attach_event(3), sd_bus_call(3), sd_bus_call_method(3), sd_bus_can_send(3), sd_bus_close(3), sd_bus_creds_get_pid(3), sd_bus_creds_new_from_pid(3), sd_bus_default(3), sd_bus_enqueue_for_read(3), sd_bus_error(3), sd_bus_error_add_map(3), sd-bus-errors(3), sd_bus_get_current_handler(3), sd_bus_get_fd(3), sd_bus_get_name_creds(3), sd_bus_get_name_machine_id(3), sd_bus_get_n_queued_read(3), sd_bus_interface_name_is_valid(3), sd_bus_is_open(3), sd_bus_list_names(3), sd_bus_message_append(3), sd_bus_message_append_array(3), sd_bus_message_append_basic(3), sd_bus_message_append_string_memfd(3), sd_bus_message_append_strv(3), sd_bus_message_at_end(3), sd_bus_message_copy(3), sd_bus_message_dump(3), sd_bus_message_get_cookie(3), sd_bus_message_get_monotonic_usec(3), sd_bus_message_get_signature(3), sd_bus_message_get_type(3), sd_bus_message_new(3), sd_bus_message_new_method_call(3), sd_bus_message_new_method_error(3), sd_bus_message_new_signal(3), sd_bus_message_open_container(3), sd_bus_message_read(3), sd_bus_message_read_array(3), sd_bus_message_read_basic(3), sd_bus_message_read_strv(3), sd_bus_message_rewind(3), sd_bus_message_seal(3), sd_bus_message_sensitive(3), sd_bus_message_set_destination(3), sd_bus_message_set_expect_reply(3), sd_bus_message_skip(3), sd_bus_message_verify_type(3), sd_bus_negotiate_fds(3), sd_bus_new(3), sd_bus_path_encode(3), sd_bus_pending_method_calls(3), sd_bus_process(3), sd_bus_query_sender_creds(3), sd_bus_reply_method_error(3), sd_bus_reply_method_return(3), sd_bus_request_name(3), sd_bus_send(3), sd_bus_set_address(3), sd_bus_set_close_on_exit(3), sd_bus_set_connected_signal(3), sd_bus_set_description(3), sd_bus_set_exit_on_disconnect(3), sd_bus_set_fd(3), sd_bus_set_method_call_timeout(3), sd_bus_set_property(3), sd_bus_set_sender(3), sd_bus_set_server(3), sd_bus_set_watch_bind(3), sd_bus_slot_get_bus(3), sd_bus_slot_ref(3), sd_bus_slot_set_description(3), sd_bus_slot_set_destroy_callback(3), sd_bus_slot_set_floating(3), sd_bus_slot_set_userdata(3), sd_bus_start(3), sd_bus_track_add_name(3), sd_bus_track_new(3), sd_bus_wait(3), sd-daemon(3), sd-

device(3), sd_device_get_syspath(3), sd-event(3), sd_event_add_child(3), sd_event_add_defer(3), sd_event_add_inotify(3), sd_event_add_io(3), sd_event_add_memory_pressure(3), sd_event_add_signal(3), sd_event_add_time(3), sd_event_exit(3), sd_event_new(3), sd_event_now(3), sd_event_run(3), sd_event_set_exit_on_idle(3), sd_event_set_signal_exit(3), sd_event_set_watchdog(3), sd_event_source_set_destroy_callback(3), sd_event_wait(3), sd_get_seats(3), sd-hwdb(3), sd_hwdb_get(3), sd_hwdb_new(3), sd-id128(3), sd_id128_get_machine(3), sd_id128_randomize(3), sd_id128_to_string(3), sd_is_fifo(3), sd-journal(3), sd_journal_add_match(3), sd_journal_enumerate_fields(3), sd_journal_get_catalog(3), sd_journal_get_cursor(3), sd_journal_get_cutoff_realtime_usec(3), sd_journal_get_data(3), sd_journal_get_fd(3), sd_journal_get_realtime_usec(3), sd_journal_get_seqnum(3), sd_journal_get_usage(3), sd_journal_has_runtime_files(3), sd_journal_next(3), sd_journal_open(3), sd_journal_print(3), sd_journal_query_unique(3), sd_journal_seek_head(3), sd_journal_stream_fd(3), sd-json(3), sd_json_dispatch_string(3), sd_listen_fds(3), sd-login(3), sd_login_monitor_new(3), sd_machine_get_class(3), sd_notify(3), sd-path(3), sd_path_lookup(3), sd_pidfd_get_inode_id(3), sd_pid_get_owner_uid(3), sd_seat_get_active(3), sd_session_is_active(3), sd_uid_get_state(3), sd-varlink(3), sd_varlink_is_connected(3), sd_varlink_push_fd(3), sd_varlink_send(3), sd_varlink_server_new(3), sd_varlink_set_description(3), sd_varlink_set_relative_timeout(3), sd_watchdog_enabled(3), ttyslot(3), udev_device_get_syspath(3), udev_device_has_tag(3), udev_device_new_from_syspath(3), udev_enumerate_add_match_subsystem(3), udev_enumerate_new(3), udev_enumerate_scan_devices(3), udev_list_entry(3), udev_monitor_filter_update(3), udev_monitor_new_from_netlink(3), udev_monitor_receive_device(3), udev_new(3), binfmt.d(5), capsule@.service(5), core(5), crypttab(5), dnf4.conf(5), dnssec-trust-anchors.d(5), environment.d(5), homed.conf(5), hostname(5), integritytab(5), journald.conf(5), journal-remote.conf(5), journal-upload.conf(5), locale.conf(5), localtime(5), logind.conf(5), machine-id(5), machine-info(5), modules-load.d(5), networkd.conf(5), oomd.conf(5), org.freedesktop.home1(5), org.freedesktop.hostname1(5), org.freedesktop.import1(5), org.freedesktop.locale1(5), org.freedesktop.LogControl1(5), org.freedesktop.login1(5), org.freedesktop.machine1(5), org.freedesktop.network1(5), org.freedesktop.oom1(5), org.freedesktop.portable1(5), org.freedesktop.resolve1(5), org.freedesktop.systemd1(5), org.freedesktop.sysupdate1(5), org.freedesktop.timedate1(5), org.freedesktop.timesync1(5), os-release(5), pmlogger.control(5), proc(5), proc_sys_kernel(5), repart.d(5), resolved.conf(5), sysctl.d(5), sysext.conf(5), systemd.automount(5), systemd.device(5), systemd.dns-delegate(5), systemd.dnssd(5), systemd.exec(5), systemd.kill(5), systemd.mount(5), systemd.netdev(5), systemd.network(5), systemd.nspawn(5), systemd.path(5), systemd.pcrlock(5), systemd.preset(5), systemd.resource-control(5), systemd.scope(5), systemd.service(5), systemd-sleep.conf(5), systemd.slice(5), systemd.socket(5), systemd.swap(5), systemd-system.conf(5), systemd.target(5), systemd.timer(5), systemd.unit(5), sysupdate.d(5), sysupdate.features(5), sysusers.d(5), timesyncd.conf(5), tmpfiles.d(5), user@.service(5), utmp(5), vconsole.conf(5), veritytab(5), yum.conf(5), boot(7), bootparam(7), bootup(7), cgroups(7), daemon(7), file-hierarchy(7), kernel-command-line(7), lvmthin(7), mount_namespaces(7), pid_namespaces(7), smbios-type-11(7), systemd.directives(7), systemd.environment-generator(7), systemd.generator(7), systemd.image-filter(7), systemd.image-policy(7), systemd.index(7), systemd.journal-fields(7), systemd.offline-updates(7), systemd.special(7), systemd.syntax(7), systemd.system-credentials(7), systemd.time(7), systemd.v(7), agetty(8), cryptsetup(8), ctrlaltdel(8), integritysetup(8), logrotate(8), nss-myhostname(8), nss-mymachines(8), nss-resolve(8), nss-systemd(8), pam_systemd(8), pam_systemd_home(8), poweroff(8), rpm-plugin-prioreset(8), rpm-plugin-

systemd-inhibit(8), shutdown(8), systemd-ask-password-console.service(8), systemd-backlight@.service(8), systemd-battery-check.service(8), systemd-binfmt.service(8), systemd-bless-boot-generator(8), systemd-bless-boot.service(8), systemd-boot-check-no-failures.service(8), systemd-boot-clear-sysfail.service(8), systemd-boot-random-seed.service(8), systemd-bsod.service(8), systemd-cryptsetup(8), systemd-cryptsetup-generator(8), systemd-debug-generator(8), systemd-environment-d-generator(8), systemd-factory-reset(8), systemd-factory-reset-generator(8), systemd-fsck@.service(8), systemd-fstab-generator(8), systemd-getty-generator(8), systemd-gpt-auto-generator(8), systemd-hibernate-resume-generator(8), systemd-hibernate-resume.service(8), systemd-homed.service(8), systemd-hostnamed.service(8), systemd-hwdb(8), systemd-importd.service(8), systemd-import-generator(8), systemd-integritysetup-generator(8), systemd-integritysetup@.service(8), systemd-journald.service(8), systemd-journal-gatewayd.service(8), systemd-localed.service(8), systemd-logind.service(8), systemd-loop@.service(8), systemd-machined.service(8), systemd-machine-id-commit.service(8), systemd-makefs@.service(8), systemd-modules-load.service(8), systemd-mountfsd.service(8), systemd-networkd.service(8), systemd-networkd-wait-online.service(8), systemd-network-generator.service(8), systemd-nsresourced.service(8), systemd-oomd.service(8), systemd-pcrlock(8), systemd-pcrphase.service(8), systemd-portabled.service(8), systemd-poweroff.service(8), systemd-quotacheck@.service(8), systemd-random-seed.service(8), systemd-rc-local-generator(8), systemd-remount-fs.service(8), systemd-repart(8), systemd-resolved.service(8), systemd-rfkill.service(8), systemd-run-generator(8), systemd-socket-proxyd(8), systemd-soft-reboot.service(8), systemd-ssh-generator(8), systemd-storagetm.service(8), systemd-suspend.service(8), systemd-sysctl.service(8), systemd-sysext(8), systemd-system-update-generator(8), systemd-sysupdate(8), systemd-sysupdated.service(8), systemd-sysusers(8), systemd-timedated.service(8), systemd-timesyncd.service(8), systemd-time-wait-sync.service(8), systemd-tmpfiles(8), systemd-tpm2-clear.service(8), systemd-tpm2-generator(8), systemd-tpm2-setup.service(8), systemd-update-done.service(8), systemd-update-utmp.service(8), systemd-userdbd.service(8), systemd-user-sessions.service(8), systemd-validatefs@.service(8), systemd-vconsole-setup.service(8), systemd-veritysetup-generator(8), systemd-veritysetup@.service(8), systemd-volatile-root.service(8), systemd-xdg-autostart-generator(8), uuidd(8), veritysetup(8)
