NAME

systemd, init - systemd system and service manager

SYNOPSIS

systemd [OPTIONS...]

init [OPTIONS...] {COMMAND}

DESCRIPTION

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.

For compatibility with SysV, if systemd is called as **init** and a PID that is not 1, it will execute **telinit** and pass all command line arguments unmodified. That means **init** and **telinit** are mostly equivalent when invoked from normal login sessions. See **telinit(8)** for more information.

When run as system instance, systemd interprets the configuration file system.conf, otherwise user.conf. See **systemd-system.conf**(5) for more information.

OPTIONS

The following options are understood:

--test
Determine startup sequence, dump it and exit. This is an option useful for debugging only.

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

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

--system, **--user**
For **--system**, tell systemd to run a system instance, even if the process ID is not 1, i.e. systemd is not run as init process. **--user** does the opposite, running a user instance even if the process ID is 1. Normally it should not be necessary to pass these options, as systemd automatically detects the mode it is started in. These options are hence of little use except for debugging. Note that it is not supported booting and maintaining a full system with systemd running in **--system** mode, but PID not 1. In practice, passing **--system** explicitly is only useful in conjunction with **--test**.

--dump-core
Dump core on crash. This switch has no effect when run as user instance.

--crash-shell
Run shell on crash. This switch has no effect when run as user instance.

--confirm-spawn
Ask for confirmation when spawning processes. This switch has no effect when run as user instance.

--show-status=
Show terse service status information while booting. This switch has no effect when run as user instance. Takes a boolean argument which may be omitted which is interpreted as **true**.

--log-target=
Set log target. Argument must be one of **console**, **journal**, **syslog**, **kmsg**, **journal-or-kmsg**, **syslog-or-kmsg**, **null**.

--log-level=
Set log level. As argument this accepts a numerical log level or the well-known **syslog(3)** symbolic names (lowercase): **emerg**, **alert**, **crit**, **err**, **warning**, **notice**, **info**, **debug**.

--log-color=
Highlight important log messages. Argument is a boolean value. If the argument is omitted,

it defaults to **true**.

--log-location=
Include code location in log messages. This is mostly relevant for debugging purposes. Argument is a boolean value. If the argument is omitted it defaults to **true**.

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

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

--version
Print a short version string and exit.

CONCEPTS

systemd provides a dependency system between various entities called units of 12 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, dynamically from system state or programmatically at runtime. Units may be active (meaning started, bound, plugged in, ..., depending on the unit type, see below), or inactive (meaning stopped, unbound, unplugged, ...), as well as in the process of being activated or deactivated, i.e. between the two states (these states are called activating, deactivating). A special failed state is available as well, which is very similar to inactive and is entered when the service failed in some way (process returned error code on exit, or crashed, or an operation timed out). If this state is entered, the cause will be logged, for later reference. Note that the various unit types may have a number of additional substates, which are mapped to the five generalized unit states described here.

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. Snapshot units can be used to temporarily save the state of the set of systemd units, which later may be restored by activating the saved snapshot unit. For more information see **systemd.snapshot**(5).
8. Timer units are useful for triggering activation of other units based on timers. You may find details in **systemd.timer**(5).
9. 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).
10. Path units may be used to activate other services when file system objects change or are modified. See **systemd.path**(5).
11. 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).

12. 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.

Processes systemd spawns are placed in individual Linux control groups named after the unit which they belong to in the private systemd hierarchy. (see **cgroups.txt**[1] 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/systemd/), or in tools such as **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 the SysV init system to a large degree: SysV init scripts are supported and simply read as an alternative (though limited) configuration file format. The SysV /dev/initctl interface is provided, and compatibility implementations of the various SysV client tools are available. In addition to that, various established Unix functionality such as /etc/fstab or the utmp database are supported.

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.

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 or /proc.

For more information about the concepts and ideas behind systemd, please refer to the **Original Design Document**[2].

Note that some but not all interfaces provided by systemd are covered by the **Interface Stability Promise**[3].

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 the **Generators Specification**[4].

Systems which invoke systemd in a container or initrd environment should implement the **Container Interface**[5] or **initrd Interface**[6] specifications, respectively.

DIRECTORIES

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 /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**[7] 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).

SysV init scripts directory

The location of the SysV init script directory varies between distributions. If systemd cannot find a native unit file for a requested service, it will look for a SysV init script of the same name (with the .service suffix removed).

SysV runlevel link farm directory

The location of the SysV runlevel link farm directory varies between distributions. systemd will take the link farm into account when figuring out whether a service shall be enabled. Note that a service unit with a native unit configuration file cannot be started by activating it in the SysV runlevel link farm.

SIGNALS

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**.

SIGINT

Upon receiving this signal the systemd system manager will start the ctrl-alt-del.target unit. This is mostly equivalent to **systemctl start ctl-alt-del.target**.

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 **systemctl 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 start 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**.

SIGRTMIN+4
Powers off the machine, starts the poweroff.target unit. This is mostly equivalent to **systemctl start poweroff.target**.

SIGRTMIN+5
Reboots the machine, starts the reboot.target unit. This is mostly equivalent to **systemctl start reboot.target**.

SIGRTMIN+6
Reboots the machine via kexec, starts the kexec.target unit. This is mostly equivalent to **systemctl start kexec.target**.

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

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

SIGRTMIN+22, **SIGRTMIN+23**
Sets the log level to debug (or info on **SIGRTMIN+23**), as controlled via

systemd.log_level=debug (or *systemd.log_level=info* on **SIGRTMIN+23**) on the kernel command line.

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

SIGRTMIN+26, **SIGRTMIN+27**, **SIGRTMIN+28**, **SIGRTMIN+29**
Sets the log level to journal-or-kmsg (or console on **SIGRTMIN+27**, kmsg on **SIGRTMIN+28**, or syslog-or-kmsg on **SIGRTMIN+29**), as controlled via *systemd.log_target=journal-or-kmsg* (or *systemd.log_target=console* on **SIGRTMIN+27**, *systemd.log_target=kmsg* on **SIGRTMIN+28**, or *systemd.log_target=syslog-or-kmsg* on **SIGRTMIN+29**) on the kernel command line.

ENVIRONMENT

SYSTEMD_LOG_LEVEL
systemd reads the log level from this environment variable. This can be overridden with **--log-level=**.

SYSTEMD_LOG_TARGET
systemd reads the log target from this environment variable. This can be overridden with **--log-target=**.

SYSTEMD_LOG_COLOR
Controls whether systemd highlights important log messages. This can be overridden with **--log-color=**.

SYSTEMD_LOG_LOCATION
Controls whether systemd prints the code location along with log messages. This can be overridden with **--log-location=**.

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**[7] to find its configuration.

SYSTEMD_UNIT_PATH
Controls where systemd looks for unit files.

SYSTEMD_SYSVINIT_PATH
Controls where systemd looks for SysV init scripts.

SYSTEMD_SYSVRCND_PATH
Controls where systemd looks for SysV init script runlevel link farms.

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

NOTIFY_SOCKET
Set by systemd for supervised processes for status and start-up completion notification. See **sd_notify(3)** for more information.

KERNEL COMMAND LINE

When run as system instance systemd parses a number of kernel command line arguments[8]:

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 initial RAM disk (initrd), while the one that is not prefixed only in the main system.

systemd.dump_core=
Takes a boolean argument. If **true**, systemd dumps core when it crashes. Otherwise, no core

dump is created. Defaults to **true**.

systemd.crash_shell=
Takes a boolean argument. If **true**, systemd spawns a shell when it crashes. Otherwise, no shell is spawned. Defaults to **false**, for security reasons, as the shell is not protected by any password authentication.

systemd.crash_chvt=
Takes an integer argument. If positive systemd activates the specified virtual terminal when it crashes. Defaults to **-1**.

systemd.confirm_spawn=
Takes a boolean argument. If **true**, asks for confirmation when spawning processes. Defaults to **false**.

systemd.show_status=
Takes a boolean argument or the constant **auto**. If **true**, shows terse service status updates on the console during bootup. **auto** behaves like **false** until a service fails or there is a significant delay in boot. Defaults to **true**, unless **quiet** is passed as kernel command line option in which case it defaults to **auto**.

systemd.log_target=, *systemd.log_level=*, *systemd.log_color=*, *systemd.log_location=*
Controls log output, with the same effect as the *SYSTEMD_LOG_TARGET*, *SYSTEMD_LOG_LEVEL*, *SYSTEMD_LOG_COLOR*, *SYSTEMD_LOG_LOCATION* environment variables described above.

systemd.default_standard_output=, *systemd.default_standard_error=*
Controls default standard output and error output for services, with the same effect as the **--default-standard-output=** and **--default-standard-error=** command line arguments described above, respectively.

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.

quiet
Turn off status output at boot, much like *systemd.show_status=false* 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.

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.

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

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

2, *3*, *4*, *5*
Boot into the specified legacy SysV runlevel. These are equivalent to *systemd.unit=runlevel2.target*, *systemd.unit=runlevel3.target*, *systemd.unit=runlevel4.target*, and *systemd.unit=runlevel5.target*, respectively, and provided for compatibility reasons and to be easier to type.

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)**.

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

SOCKETS AND FIFOS

/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/shutdownd
Used internally by the **shutdown(8)** tool to implement delayed shutdowns. This is an **AF_UNIX** datagram socket.

/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.

/dev/initctl
Limited compatibility support for the SysV client interface, as implemented by the systemd-initctl.service unit. This is a named pipe in the file system. This interface is obsolete and should not be used in new applications.

SEE ALSO

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

NOTES

1. cgroups.txt
   https://www.kernel.org/doc/Documentation/cgroups/cgroups.txt
2. Original Design Document
   http://0pointer.de/blog/projects/systemd.html
3. Interface Stability Promise
   http://www.freedesktop.org/wiki/Software/systemd/InterfaceStabilityPromise
4. Generators Specification
   http://www.freedesktop.org/wiki/Software/systemd/Generators
5. Container Interface
   http://www.freedesktop.org/wiki/Software/systemd/ContainerInterface
6. initrd Interface
   http://www.freedesktop.org/wiki/Software/systemd/InitrdInterface
7. XDG Base Directory specification
   http://standards.freedesktop.org/basedir-spec/basedir-spec-latest.html
8. If run inside a Linux container these arguments may be passed as command line arguments 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.
9. systemd Homepage
   http://www.freedesktop.org/wiki/Software/systemd/