Linux and Unix init and telinit commands

www.computerhope.com/unix/telinit.htm

Quick links

About init, telinit

Process control initialization.

Syntax

```
/sbin/init [ -a ] [ -s ] [ -b ] [ -z xxx ] [
0123456Ss ]
```

```
/sbin/telinit [ -t sec ] [ 0123456sSQqabcUu ]
```

Init

init is the parent of all processes. Its primary role is to create processes from a script stored in the file **/etc/inittab**. This file usually has entries which cause **init** to spawn **gettys** on each line that users can log in. It also controls autonomous processes required by any particular system.

Runlevels

A runlevel is a software configuration of the system which allows only a selected group of processes to exist. The processes spawned by **init** for each of these runlevels are defined in the **/etc/inittab** file. **Init** can be in one of eight runlevels: **0** through **6**, and **S** or **s**. The runlevel is changed by having a privileged user run **telinit**, which sends appropriate signals to **init**, telling it which runlevel to change to.

Runlevels **0**, **1**, and **6** are reserved. Runlevel **0** is used to halt the system, runlevel **6** is used to reboot the system, and runlevel **1** is used to get the system down into single user mode. Runlevel **S** is not really meant to be used directly, but more for the scripts that are executed when entering runlevel **1**.

Runlevels **7** - **9** are also valid, though not really documented. This is because "traditional" Unix variants don't use them. In case you're curious, runlevels **S** and **s** are in fact the same. Internally they are aliases for the same runlevel.

Booting

After **init** is invoked as the last step of the kernel boot sequence, it looks for the file **/etc/inittab** to see if there is an entry of the type **initdefault**. The initdefault entry determines the initial runlevel of the system. If there is no such entry (or no **/etc/inittab** at all), a runlevel must be entered at the system console.

Runlevel **S** or **s** bring the system to single user mode and do not require an **/etc/inittab** file. In single user mode, a root shell is opened on **/dev/console**.

When entering single user mode, **init** initializes the consoles **stty** settings to sane values. "CLocal" mode is set. Hardware speed and handshaking are not changed.

When entering a multi-user mode for the first time, **init** performs the **boot** and **bootwait** entries to allow file systems to be mounted before users can log in. Then all entries matching the runlevel are processed.

When starting a new process, **init** first checks whether the file **/etc/initscript** exists. If it does, it uses this script to start the process.

Each time a child terminates, **init** records the fact and the reason it died in **/var/run/utmp** and **/var/log/wtmp**, provided that these files exist.

Changing Runlevels

After it has spawned all of the processes specified, **init** waits for one of its descendant processes to die, for a "powerfail" signal, or until it is signaled by **telinit** to change the system's runlevel. When one of the above three conditions occurs, it re-examines the **/etc/inittab** file. New entries can be added to this file at any time. However, init still waits for one of the above three conditions to occur. To provide for an instantaneous response, the **telinit Q** (or **q**) command can wake up **init** to re-examine the **/etc/inittab file**.

If init is not in single user mode and receives a powerfail signal (**SIGPWR**), it reads the file **/etc/powerstatus**. It then starts a command based on the contents of this file:

- **F** (FAIL) - Power is failing, UPS is providing the power. Execute the powerwait and powerfail entries.
- **O** (OK) - The power has been restored, execute the powerokwait entries.
- **L** (LOW) - The power is failing and the UPS has a low battery. Execute the powerfailnow entries.

If **/etc/powerstatus** doesn't exist or contains anything else then the letters **F**, **O** or **L**, **init** will behave as if it has read the letter **F**.

Usage of **SIGPWR** and **/etc/powerstatus** is discouraged. Someone wanting to interact with **init** should use the **/dev/initctl** control channel. More information about this is available by viewing the source code of the **sysvinit** package.

When **init** is requested to change the runlevel, it sends the warning signal **SIGTERM** to all processes that are undefined in the new runlevel. It then waits 5 seconds before forcibly terminating these processes via the **SIGKILL** signal. Note that init assumes that all these processes (and their descendants) remain in the same process group which **init** originally created for them. If any process changes its process group affiliation it will not receive these signals. Such processes need to be terminated separately.

Environment

Init sets the following environment variables for all its children:

PATH	**/bin:/usr/bin:/sbin:/usr/sbin**
INIT_VERSION	As the name says. Useful to determine if a script runs directly from **init**.
RUNLEVEL	The current system runlevel.
PREVLEVEL	The previous runlevel (especially useful after changing runlevel).
CONSOLE	The system console. This is really inherited from the kernel; however if it is not set init will set it to **/dev/console** by default.

Telinit

/sbin/telinit is linked to **/sbin/init**. It takes a one-character argument and signals **init** to perform the appropriate action. The following arguments serve as directives to **telinit**:

0, **1**, **2**, **3**, **4**, **5** or **6**	tell **init** to switch to the specified run level.
a, **b**, **c**	tell **init** to process only those **/etc/inittab** file entries having runlevel **a**, **b** or **c**.
Q or **q**	tell **init** to re-examine the **/etc/inittab** file.
S or **s**	tell **init** to switch to single user mode.
U or **u**	tell **init** to re-execute itself (preserving the state). No re-examining of **/etc/inittab** file happens. Run level should be one of **S**, **s**, **1**, **2**, **3**, **4**, or **5**, otherwise request would be silently ignored.

telinit can also tell **init** how long it should wait between sending processes the **SIGTERM** and **SIGKILL** signals. The default is 5 seconds, but this can be changed with the **-t sec** option.

telinit can be invoked only by users with appropriate privileges.

The **init** binary checks if it is **init** or **telinit** by looking at its process id; the real **init**'s process id is always **1**. From this it follows that instead of calling **telinit** one can also just use **init** instead as a shortcut.

Interface

Init listens on a fifo in **/dev** (**/dev/initctl**) for messages. **Telinit** uses this to communicate with **init**. The interface is not very well documented; to learn more about the interface, users must view the source itself.

Signals

Init reacts to several signals:

SIGHUP	Has the same effect as **telinit q**.
SIGUSR1	On receipt of this signal, **init** closes and re-opens its control fifo, **/dev/initctl**. Useful for bootscripts when /dev is remounted.
SIGINT	Normally the kernel sends this signal to **init** when CTRL-ALT-DEL is pressed. It activates the **ctrlaltdel** action.
SIGWINCH	The kernel sends this signal when the **KeyboardSignal** key is hit. It activates the **kbrequest** action.
CONFORMING TO	**Init** is compatible with the System V init. It works closely together with the scripts in the directories **/etc/init.d** and **/etc/rc{runlevel}.d**. If your system uses this convention, there should be a README file in the directory **/etc/init.d** explaining how these scripts work.

Files

/etc/inittab
/etc/initscript

/dev/console
/var/run/utmp
/var/log/wtmp
/dev/initctl

Init Bootflags

It is possible to pass a number of flags to init from the boot monitor (such as LILO). **Init** accepts the following flags:

-s, **S**, **single**	Single user mode boot. In this mode **/etc/inittab** is examined and the bootup rc scripts are usually run before the single user mode shell is started.
1-5	Runlevel to boot into.
-b, **emergency**	Boot directly into a single user shell without running any other startup scripts.
-a, **auto**	The LILO boot loader adds the word "**auto**" to the command line if it booted the kernel with the default command line (without user intervention). If this is found init sets the "**AUTOBOOT**" environment variable to "**yes**". Note that you cannot use this for any security measures; of course the user could specify "**auto**" or **-a** on the command line manually.
-z *xxx*	The argument to **-z** is ignored. You can use this to expand the command line a bit, so that it takes some more space on the stack. **Init** can then manipulate the command line so that **ps** shows the current runlevel.

telinit can also tell **init** how long it should wait between sending processes the **SIGTERM** and **SIGKILL** signals. The default is 5 seconds, but this can be changed with the **-t sec** option.

telinit can be invoked only by users with appropriate privileges.

The **init** binary checks if it is **init** or **telinit** by looking at its process id; the real init's process id is always **1**. From this it follows that instead of calling **telinit** one can also just use **init** instead as a shortcut.

Related commands

kill — Send a signal to a process, affecting its behavior or killing it.
login — Begin a session on a system.
service — Run a System V init script.
sh — The Bourne shell command interpreter.