

# Alternatives to systemd

This page describes the various init systems which are available as alternatives to systemd

**Init** is the first process started during system boot. It is a daemon process that continues running until the system is shut down. Init is the direct or indirect ancestor of all other processes, and automatically adopts all orphaned processes. It is started by the kernel using a hard-coded filename; if the kernel is unable to start it, panic will result. Init is typically assigned process identifier 1.

The init *scripts* (aka *rc* runtime configuration scripts) are launched by the init process to guarantee basic functionality on system start and shutdown. This includes (un)mounting of file systems and launching of daemons. A *service manager* takes this one step further by providing active control over launched processes, or [process supervision](#). An example is to monitor for crashes and restart processes accordingly.

These components combine to the init *system*. Some init systems incorporate the service manager in the init process or have init scripts in close relation to them. Below, such init systems are referred to as *integrated*, although entries in different categories may explicitly depend on each other.

A nice (but still non-comprehensive) overview of init systems can be found in [this](#) blog entry, titled “A history of modern init systems (1992-2015)”

## init systems

System	Description	Latest release
<a href="#">BusyBox</a> init	quite small init without runlevels, solely signal driven IPC (no fifos, sockets, SysV IPC)	2019-10-25
<a href="#">dinit</a>	dependency based C++ init with process supervision, roll-back, and socket activation	2020-01-01
<a href="#">epoch</a>	sequential, non-parallel init without dependency tracking designed for minimal footprint and unified configuration	2015-06-23
<a href="#">finit</a>	fast, event based, modular, extensible (C hooks/plugins) init with SysV runlevels, proper daemon supervision and logging, and optional builtin getty and inetd	2018-01-23
<a href="#">initng</a> (2)	modular, extensible, parallel, asynchronous, dependency based init	2007-03-25
<a href="#">myinit</a>	parallel init with dependency tracking via reference counting (akin to the “ <i>need</i> ” concept) lacking proper supervision (respawn only, no output logging)	2011-07-11
<a href="#">nosh</a>	exec chaining, dependency based <a href="#">daemontools family</a> C++ init and supervision suite with reliable logging, (console) virtual terminal management, systemd unit file compatibility; can also be used as service manager under a different init	2017-12-11
<a href="#">pies</a>	dependency based GNU init daemon and super server with SysV runlevels, daemon supervision and logging, inetd functionality, socket activation, fine grained per service access control, understands SysV inittab, inetd.conf, and MeTA1 config files	2016-10-01
<a href="#">procd</a> (2)	OpenWrt init and process/service management daemon with ubus integration	2018-07-30

System	Description	Latest release
<a href="#">sinit</a>	Simple init initially based on Rich Felker's minimal init	2015-06-16
<a href="#">sninit</a>	Small init implementation with SysV init like (sub)runlevels	2015-12-31
<a href="#">SysV init (2)</a>	Traditional System V init. New release: v2.91 on July 07 2018 ( <a href="#">git commit log</a> )	2018-07-07
<a href="#">ToyBox init (2)</a>	similar to/(almost) compatible with BusyBox init (same config syntax/file)	2018-06-24
<a href="#">ToyBox oneit (2)</a>	very simple init launcher (just (re)spawns a single child process and reacts to incoming signals)	2018-06-24
<a href="#">ueld</a>	simple configuration, solely signal driven (like BSD and Busy/ToyBox init)	2017-06-24
<a href="#">uinit</a>	Smallest init possible	2017-05-16

## abandoned/defunct/discontinued/dormant/halted init projects

Source code is available for download to anyone interested (in reviving them). Some of the projects listed below still compile/work while others may need some tweaking to make them work (again).

System	Description	Latest release
<a href="#">cinit (2, 3)</a>	dependency based parallel init without subprocess output logging ala <a href="#">minit</a> /daemontools	2009-09-29
<a href="#">daemond</a>	daemon control and init daemon written in C++ (akin to <a href="#">dinit</a> )	2003-11-14
<a href="#">depinit</a>	supports parallel execution, dependencies, true roll-back, log pipelines, improved signaling, and unmounting of filesystems on shutdown	2003-10-06
<a href="#">einit (2, 3, 4)</a>	modular, parallel, asynchronous, dependency based init system; XML (service) config files	2007-12-18
<a href="#">ettcl</a>	custom Tcl-7.6 based interpreter to run the <a href="#">ETLinux</a> init, crond, httpd, and telnetd daemons written in <a href="#">Tcl</a>	2002-??-??
<a href="#">jinit</a>	C++ init daemon supporting dependency tracking akin to the "need" concept	2003-04-22
<a href="#">minit</a>	small dependency ordered, parallel init with proper subprocess supervision and logging	2005-??-??
<a href="#">ninit</a>	dependency based parallel init with supervision, logging, and time-based scheduling akin to crond ( <a href="#">HOWTO</a> )	2010-01-16
<a href="#">pinit (sf)</a>	XML configuration, dependency based fast parallel boot, modular design; uses Glib 2.0 and LibXML 2	2003-05-01
<a href="#">serel (sf)</a>	parallel service launcher with synchronisation and integrity-checking (primarily to speed up SysV init boots, akin to startpar)	2002-08-06
<a href="#">simpleinit(-msb)</a>	dependency based (implements the "need" concept) sequential init without proper supervision (respawn only, no output logging); shipped with util-linux until v2.20; the msb fork is still used in <a href="#">Source Mage Linux</a>	2007-11-08
<a href="#">System XVI</a>	modular, self-healing, and interface-oriented service manager and init system	2016-05-18
<a href="#">twinit</a>	tiny init largely written in x86 assembler	2003-09-19
<a href="#">upstart</a>	"event based" init that reacts eagerly to incoming "events" to emulate dependency tracking; <a href="#">leaks memory</a>	2014-09-04

System	Description	Latest release
<a href="#">uselessd</a> (2)	forked from systemd v208	2015-01-06

## process supervisors (overview) and service managers

service manager: a **suite of programs** which start and stop services, both long-running daemons and one-time initialization scripts, in the proper order according to a dependency tree

Program	Description	Latest release
<a href="#">anopa</a>	service management suite built around <a href="#">s6</a> ; can be used as init via <a href="#">exec chaining</a>	2017-06-20
<a href="#">Circus</a> (docs)	runs, controls and monitors processes and sockets; written in <a href="#">Python</a> ; uses <a href="#">ZeroMQ</a>	2018-06-15
<a href="#">daemontools</a>	collection of tools for managing UNIX services	2001-07-12
<a href="#">daemontools encore</a>	an enhanced version of daemontools ( <a href="#">github repository</a> )	2014-04-02
<a href="#">freedt</a>	an independent daemontools reimplementation	2014-09-03
<a href="#">GNU Shepherd</a>	service manager and init daemon written in <a href="#">Guile</a> (formerly known as GNU dmd)	2018-03-22
<a href="#">god</a> (2)	an easily configurable, extensible, monitoring framework written in <a href="#">Ruby</a>	2014-03-06
<a href="#">MiniBase</a>	small userspace utilities for Linux; includes a process supervisor that can be used as stage 2 of process #1 via <a href="#">exec chaining</a> akin to <a href="#">s6-svscan</a>	2018-02-10
<a href="#">Monit</a>	process and system monitoring daemon; system status is viewable directly from commandline, or via a native HTTP(S) web server	2018-05-29
<a href="#">Mudur + Comar</a>	Python rc script (Mudur) and configuration/network/service manager with dbus integration (Comar) used in <a href="#">Pardus Linux</a>	2011-10-24
<a href="#">OpenRC</a> (2, 3)	dependency based service management suite; includes a process supervisor and a simple init alternative	2018-08-06
<a href="#">perp</a>	persistent process (service) supervisor and management framework for UNIX	2013-01-11
<a href="#">procer</a>	process supervisor developed for <a href="#">mongrel2</a>	2014-03-18
<a href="#">restartd</a>	a process-restarting daemon (aka Debian restartd)	2013-01-11
<a href="#">runit</a>	process supervision suite in the line of daemontools; includes a simple <a href="#">init</a> (2) <a href="#">replacement</a>	2014-08-10
<a href="#">s6</a>	small process supervision suite for UNIX ala daemontools, runit, and perp; can be used as stage 2 of process #1 via <a href="#">exec chaining</a>	2018-08-14
<a href="#">s6-rc</a>	service management suite for <a href="#">s6</a>	2018-08-20
<a href="#">Supervisor</a>	process control daemon written in Python	2018-02-15
<a href="#">ssm</a>	simple service manager bash scripts for Linux	2016-03-31
<a href="#">Supervision Framework</a>	OpenRC-like service management suite for daemontools(-encore), runit, and s6	2018-09-01
<a href="#">systemctl-replacement</a>	drop-in reading systemd service descriptions, with docker-oriented init function, written in Python	2018-09-10

## SysV init enhancement addons

System	Description	Latest release
<a href="#">insserv</a>	can be used with SysV -based init systems; provides dependency-driven system startup (dependencies are specified by LSB headers within init.d scripts)	2012-11-14
<a href="#">startpar</a>	can be used by the SysV RC boot system executor to allow parallel process system startup	2014-02-09

## Safer, supervision-friendly replacements for crond and atd

System	Description	Latest release
<a href="#">bcron</a>	seperate tools for different tasks with strictly controlled communication between them; crontab compatible	2015-08-12
<a href="#">slicd</a> ( <a href="#">github</a> )	modular design (seperate tools); crontab compatible	2016-10-30
<a href="#">uschedule</a>	job scheduling suite that uses seperate tools for different tasks	2004-08-??

## udev alternatives for automatic Linux device file creation and management

[udev](#) (userspace /dev) is a device manager for the Linux kernel.

In April 2012, udev's source code was merged into the systemd source tree.[\[1\]](#)

Although udev can still be compiled for usage without systemd, Lennart Poettering said that they will not polish udevd outside of systemd[\[2\]](#), adding:

*"Yes, udev on non-systemd systems is in our eyes a dead end, in case you haven't noticed it yet. I am looking forward to the day when we can drop that support entirely."* So to be on the safe side, you probably want to use an alternative to udev:

DevMan	Description	Last release
<a href="#">eudev</a> (2, 3)	Gentoo's maintained fork of udev	2017-11-22
<a href="#">mdev</a>	the plug-and-play manager built into BusyBox	2018-07-31
<a href="#">mdevd</a> ( <a href="#">github</a> )	nonforking Linux kernel uevent handler daemon; uses the mdev config file format	2018-08-14
<a href="#">nldev</a> (2)	Suckless netlink frontend for mdev; replacements for udevd and udevadm (see also <a href="#">nlmon</a> )	2018-08-03
<a href="#">smdev</a>	mostly mdev-compatible Suckless program to manage device nodes	2015-04-12
<a href="#">vdev</a> ( <a href="#">github</a> )	portable UNIX userspace device-file manager	2016-08-08

For more see [this](#) StackExchange question.

Related helper tool: [udevil](#) (2) (un)mounts removable devices without a password, shows device info, and monitors device changes. It can also mount ISO files, nfs://, smb://, ftp://, ssh:// and WebDAV URLs, and tmpfs/ramfs filesystems. Uses Glib and libudev (i. e. requires (e)udev) and can work completely without systemd, consolekit, policykit, dbus, udisks, gvfs & fuse (although it can coexist with any of these).

## See also

- [Gentoo wiki page: Comparison of init systems](#) **\*\*Talk\*\*** presents a more extensive / thorough “Detailed Feature Comparison Table”
- [Gentoo wiki page: Comparison of init systems](#)
- [A history of modern init systems \(1992-2015\)](#)
- [Understanding \(SysV\) init](#)
- [init systems overview](#)
- [init system features and benefits](#)
- [Demystifying the init system](#) (init written in [Ruby](#))
- Process supervision: Problem [apparently](#) solved
- How [process supervision](#) and [service management](#) differ (according to Laurent Bercot, author of [s6\(-rc\)](#))

### Init-specific articles:

- [Upstart leaks memory](#) (reported on 2018-03-16, still no reaction whatsoever)
- Hints on how to build, install, configure, and use [ninit](#) as your default init
- [Hints](#) on how to replace your current init with [epoch](#) and [runit](#)
- [Gentoo wiki :: OpenRC](#)
- [guide: Using OpenRC on Arch Linux](#)
- [VoidLinux wiki :: Runit](#)
- [Obarun wiki :: s6-boot](#)
- [Replace systemd with busybox + minirc](#)
- [How to remove systemd from an Exherbo GNU/Linux installation](#)
- An [overview](#) of daemontools family supervision suites
- [Hints on how to perform socket activation with s6](#)
- [s6 init scheme \(2\)](#): Exec chaining into different “init stages”
- [runit init scheme \(2, 3\)](#): Process #1 starts (and supervises) 3 subprocesses implementing different “init stages”
- Simpler, more reliable, supervision-friendly alternatives to syslogd and klogd: [s6](#) + [s6-ipserver](#) + [ucspilogd](#) + [s6-log](#) or [runit](#) + [socklog](#)

From:

<https://without-systemd.frama.wiki/> - **Without Systemd**

Permanent link:

[https://without-systemd.frama.wiki/alternatives\\_to\\_systemd?rev=1579139311](https://without-systemd.frama.wiki/alternatives_to_systemd?rev=1579139311)

Last update: **2020/01/16 02:48**

