CfgMgmtCamp 2026 | 03.02.2026

Crinit

an embedded, security-aware init system

Andreas Zdziarstek, Thomas Brinker | emlix GmbH

Another init system? Who did this?

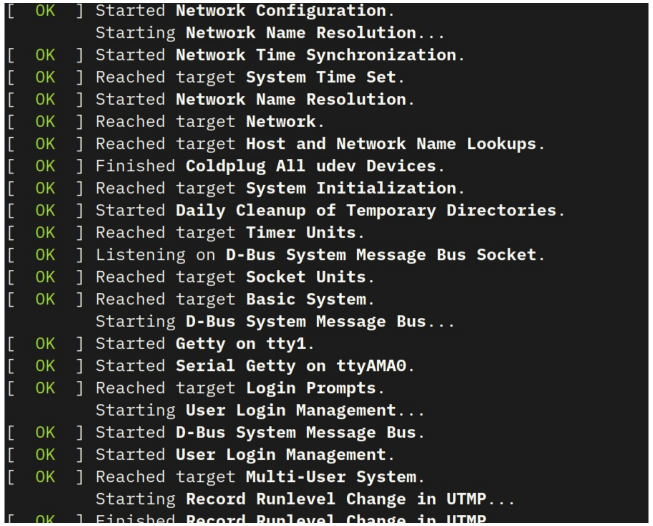
- Andreas Zdziarstek
  - Systems Engineer
- emlix GmbH
  - embedded Linux company
  - BSP and Kernel development
  - product maintenance
  - open-source component qualification
  - test automation
  - ...
  - **And also: an init system**

Our Partner:

- Elektrobit Automotive GmbH
  - Automotive software company
  - ECUs
  - Driver Assistance
  - Infotainment
  - Connected Vehicles

An init system? What’s that again?

- sometimes also called init *manager*
- Examples: systemd and sysvinit (“UNIX System V init”), also busybox-init, runit, upstart,...
- runs as PID 1, started by the Kernel at boot
- do some system setup and housekeeping
- start “everything else” until system is ready
- maybe do some process management at system runtime
- handle shutdown



```
[  OK  ] Started Network Configuration.
         Starting Network Name Resolution...
[  OK  ] Started Network Time Synchronization.
[  OK  ] Reached target System Time Set.
[  OK  ] Started Network Name Resolution.
[  OK  ] Reached target Network.
[  OK  ] Reached target Host and Network Name Lookups.
[  OK  ] Finished Coldplug All udev Devices.
[  OK  ] Reached target System Initialization.
[  OK  ] Started Daily Cleanup of Temporary Directories.
[  OK  ] Reached target Timer Units.
[  OK  ] Listening on D-Bus System Message Bus Socket.
[  OK  ] Reached target Socket Units.
[  OK  ] Reached target Basic System.
         Starting D-Bus System Message Bus...
[  OK  ] Started Getty on tty1.
[  OK  ] Started Serial Getty on ttyAMA0.
[  OK  ] Reached target Login Prompts.
         Starting User Login Management...
[  OK  ] Started D-Bus System Message Bus.
[  OK  ] Started User Login Management.
[  OK  ] Reached target Multi-User System.
         Starting Record Runlevel Change in UTMP...
[  OK  ] Finished Record Runlevel Change in UTMP
```

systemd just doing its thing

Okay,... but *why another one*?

***DISCLAIMER*: Both systemd and sysvinit are great at what they do!**

(Apologies to everyone who came here hoping for a half-hour rant on either topic.)

Motivation

- specifically developed for embedded targets
- small, testable codebase
- simple usage, simple configs

⇒ ~~so busybox-init it is :) ✅~~

er, no because at the same time:

- parallel execution with ordering when necessary
- get by without shell scripts
- configuration signatures
- runtime configuration interface (start/stop/add/list/... tasks)
- possibility to integrate with ***elos*** (daemon to collect and publish system events)

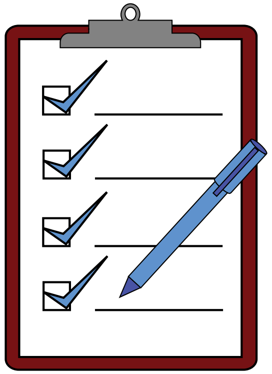


Image Source: openclipart.org, Public Domain

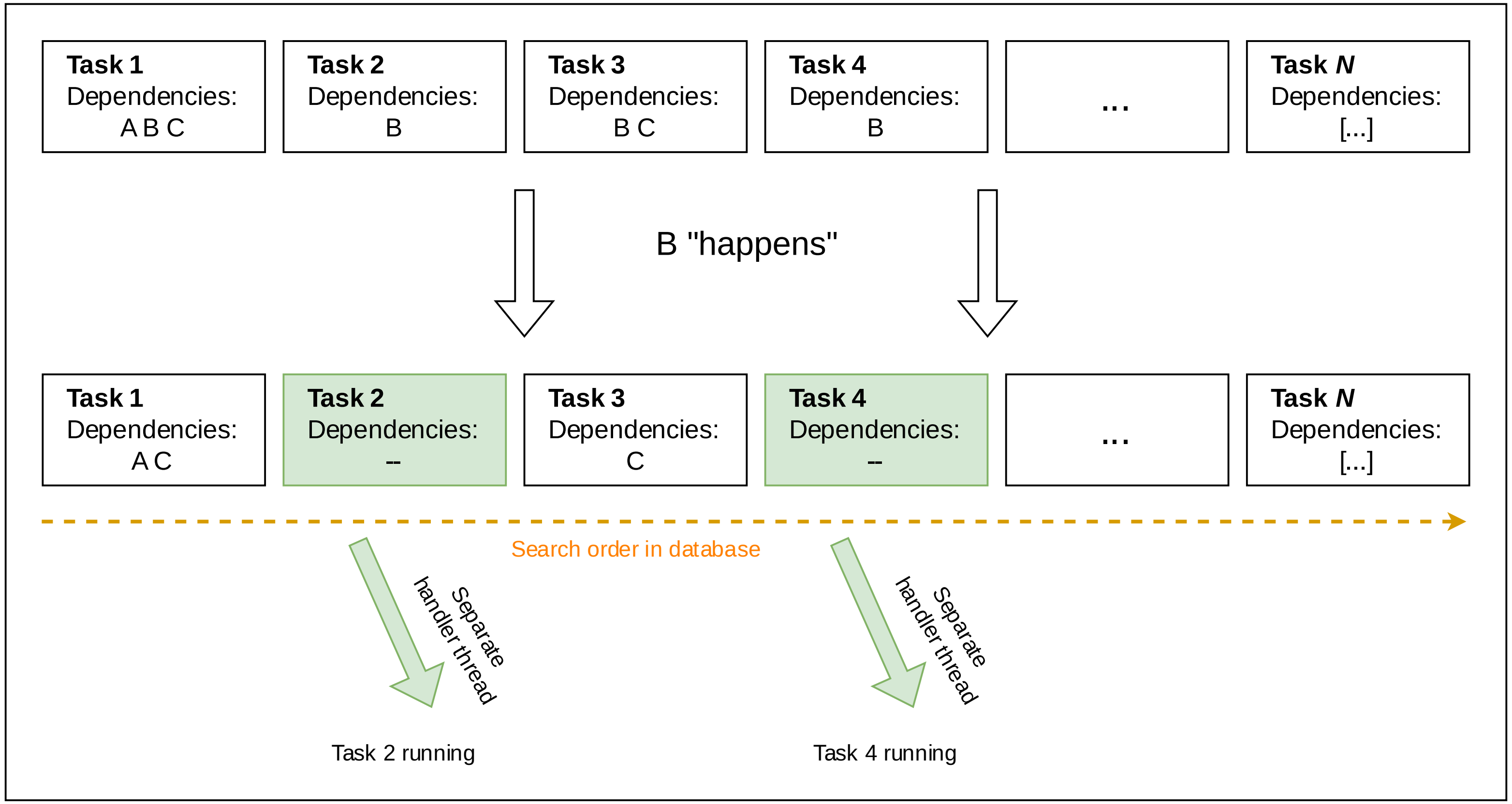
Crinit, what’s (currently) in it?

- starting of **Tasks** according to dependencies
  - dependency resolution (starting order) as a directed graph
  - independent branches/subdivisions run in parallel
  - dependencies may be on other tasks, available system features, control API interaction, and defined dependency groupings
- control API in C and a control program (`crinit-ctl`)
  - add new tasks, modify/override existing ones, query status
  - shutdown/reboot
- IO redirection (`STDOUT/ERR/IN`) to files and named pipes
- global and local process environment settings
- UID/GID and cgroup settings
- task event reporting to elos and dependencies on elos events
- timed triggers (similar to cron and systemd timers)
- optional RSA-PSS signature checking of configuration files and task definitions

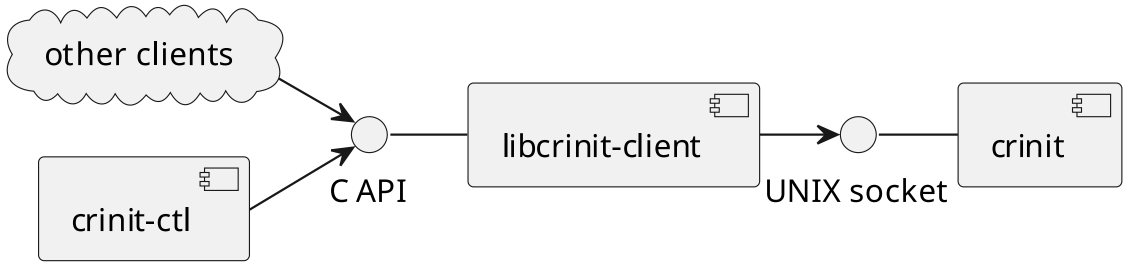
Starting is half the Task(file)

```
# Example Daemon Task file. The daemon is a hypothetical one that does "something" and
# also syslog.

NAME = some-daemon
INCLUDE = daemon_env_preset

COMMAND = /usr/bin/somedaemond -d

DEPENDS = @provided:tmp @provided:network some-daemon-setup:wait
PROVIDES = some-daemon:spawn syslog:spawn

RESPAWN = YES
RESPAWN_RETRIES = 3

ENV_SET = SOME_DAEMON_LISTEN_ADDR "0.0.0.0"
          SOME_DAEMON_SOCKET "1337"
          SOME_DAEMON_FULL_ADDR "${SOME_DAEMON_LISTEN_ADDR}:${SOME_DAEMON_SOCKET}"

IO_REDIRECT = STDOUT "/var/log/some-daemon.log" APPEND 0644
IO_REDIRECT = STDERR STDOUT
```

Dependency (management) is not a weakness!



An API you can depend upon



- Tasks
  - add new ones
  - overwrite old ones
  - enable/disable (temporarily)
  - terminate, kill, restart
  - get status
- Global Settings
  - load a new set of global settings from file
  - reload Tasks if necessary
- System
  - poweroff and reboot

Something with crypto(graphy)

- if configured (through Kernel cmdline), crinit will verify file signatures
  - for global settings and task/include/dependency-group files
- signature is expected as .sig-file
- Algorithm: RSA-PSS (RSA-4096 w. SHA256)
- A trusted root public key must be in the system keyring on boot
  - can be compiled into Kernel, or provided by e.g. HSM
  - secure boot necessary
- additional downstream public keys may be in rootfs but must be signed with root key

Showtime!

Now, we’ll see crinit in action. Hold on to your seats!

∑(crinit)

- **crinit** – a new embedded init system!
- It’s small, fast, and multi-tasky!
- It can manage your processes all simple-like!
- You can tell it to do stuff through a library!
- Can check if someone messed around with your config files!
- Open Source (MIT license): **https://github.com/Elektrobit/crinit**
- Also have a look at **https://github.com/Elektrobit/elos**, they work great together!

Testimonial: “Crinit, huh? Pretty cool, *I think I’ll try this out for my own projects.*” - a discerning colleague

I hope **you** will, too!

Question time!

If you have questions (or strong opinions on init systems), now is the time to share them.

(You can also write them to andreas.zdziarstek@emlix.com)

How can we support you?

emlix GmbH
Göttingen | Berlin | Bonn | München

Headquarters
Berliner Str. 12
D-37073 Göttingen / Germany

Fon +49 (0) 551 / 306 64 - 0
solutions@emlix.com
www.emlix.com