Demystifying Linux Kernel Initialization

Opportunity Open Source 4.0, IIIT, Allahabad, India
August 28 - 30 2026

Shuah Khan
Kernel Maintainer & Linux Fellow
The Linux Foundation
skhan@linuxfoundation.org
LinkedIn: shuah-khan

Abstract

If you ever wondered what happens between the time you power up your laptop and when you get back with that steaming cup of your favorite beverage? This talk will take the lid off and demystify the Linux kernel boot process.

In addition it covers how to enable KNL (kernel start-up parameter) initcall_debug to allows trace initcalls as they are executed to get insight into early boot sequence.

Shuah Khan

Agenda

- ➔ What’s my system doing?
- ➔ System logging
- ➔ System log levels
- ➔ dmesg
- ➔ Types of initcalls
- ➔ Skipping initcalls
- ➔ Debugging using initcalls - initcall_debug
- ➔ Speeding up kernel boot

What’s my system doing?

dmesg command

- ➔ Display or control the kernel ring buffer
- ➔ Supports several log facilities
- ➔ Supports several log levels

What’s my system doing?

dmesg log facilities

- ➔ kern - kernel messages
- ➔ user - random user-level messages
- ➔ daemon - system daemons
- ➔ syslog - messages generated internally by syslogd
- ➔ Others - check manpage

What’s my system doing?

dmesg log levels

- ➔ emerg - system is unusable
- ➔ alert - action must be taken immediately
- ➔ crit - critical conditions
- ➔ err - error conditions
- ➔ warn - warning conditions
- ➔ notice - normal but significant condition
- ➔ info - information messages
- ➔ debug - debug-level messages

What’s my system doing?

dmesg examples

- ➔ dmesg -f kern
- ➔ dmesg -f daemon
- ➔ dmesg -l emerg
- ➔ dmesg -l alert
- ➔ dmesg -l crit
- ➔ dmesg -l err
- ➔ dmesg -l warn
- ➔ dmesg -l notice
- ➔ dmesg -l info
- ➔ dmesg -l debug

What’s my system doing?

dmesg examples

- ➔ dmesg -H, –human #human readable format
- ➔ dmesg -J, –json # use JSON output format
- ➔ dmesg -C, –clear #clear the kernel ring buffer
- ➔ dmesg -c, –read-clear
- ➔ dmesg -w, –follow # wait for new messages
- ➔ dmesg -W, –follow-new # wait and print only new messages
- ➔ Other options - check manpage

early_initcall()

- ➔ Early init calls run before initializing Symmetric Multi-processing (SMP)
- ➔ Only for statically built-in code, not for modules
- ➔ Linux 7.2: “git grep early_initcall()” command run shows 115 calls
- ➔ include/linux/init.h:early_initcall(fn) __define_initcall(fn, early)

Example early initcalls

- arch/x86/events/core.c:early_initcall(init_hw_perf_events);
- arch/x86/kernel/apic/apic_numachip.c:early_initcall(numachip_system_init);
- arch/x86/kernel/apic/hw_nmi.c:early_initcall(register_nmi_cpu_backtrace_handler)
- kernel/sched/core.c:early_initcall(migration_init);
- kernel/signal.c:early_initcall(init_signal_sysctls);
- mm/memory.c:early_initcall(init_zero_pfn);
- kernel/kthread.c:early_initcall(kthreads_init);
- kernel/printk/printk.c:early_initcall(printk_set_kthreads_ready);

pure_initcall()

- ➔ No dependencies
- ➔ Only for statically built-in code, not for modules
- ➔ Initialize variables that cannot be initialized statically
- ➔ Linux 7.2: “git grep pure_initcall()” command run shows 11 calls
- ➔ include/linux/init.h:#define pure_initcall(fn) __define_initcall(fn, 0)

Pure initcalls

- arch/arm/mach-tegra/hotplug.c:pure_initcall(tegra_hotplug_init);
- arch/loongarch/kernel/perf_event.c:pure_initcall(init_hw_perf_events);
- arch/sparc/kernel/perf_event.c:pure_initcall(init_hw_perf_events);
- arch/x86/kernel/cpu/bus_lock.c:pure_initcall(setup_split_lock_delayed_work);
- drivers/pci/pci.c:pure_initcall(pci_realloc_setup_params);
- ipc/shm.c:pure_initcall(ipc_ns_init);
- kernel/bpf/core.c:pure_initcall(bpf_jit_charge_init);
- kernel/params.c:pure_initcall(param_sysfs_init);
- net/ipv4/inet_fragment.c:pure_initcall(inet_frag_wq_init);
- security/lsm_init.c:pure_initcall(security_initcall_pure);
- security/min_addr.c:pure_initcall(mmap_min_addr_init);

What comes after early and pure initcalls?

- #define core_initcall(fn) __define_initcall(fn, 1)
- #define core_initcall_sync(fn) __define_initcall(fn, 1s)
- #define postcore_initcall(fn) __define_initcall(fn, 2)
- #define postcore_initcall_sync(fn) __define_initcall(fn, 2s)
- #define arch_initcall(fn) __define_initcall(fn, 3)
- #define arch_initcall_sync(fn) __define_initcall(fn, 3s)
- #define subsys_initcall(fn) __define_initcall(fn, 4)
- #define subsys_initcall_sync(fn) __define_initcall(fn, 4s)

What comes after early and pure initcalls?

- #define fs_initcall(fn) __define_initcall(fn, 5)
- #define fs_initcall_sync(fn) __define_initcall(fn, 5s)
- #define rootfs_initcall(fn) __define_initcall(fn, rootfs)
- #define device_initcall(fn) __define_initcall(fn, 6)
- #define device_initcall_sync(fn) __define_initcall(fn, 6s)
- #define late_initcall(fn) __define_initcall(fn, 7)
- #define late_initcall_sync(fn) __define_initcall(fn, 7s)

initcall_debug

- ➔ KNL (kernel start-up parameter)
- ➔ e.g: GRUB_CMDLINE_LINUX="earlyprintk=vga printk.time=1 initcall_debug"
- ➔ depends on CONFIG_PRINTK_TIME and CONFIG_KALLSYMS
- ➔ related config options: TRACEPOINTS_ENABLED
- ➔ enables initcall tracing
- ➔ Increased number of messages
- ➔ Set CONFIG_LOG_BUF_SHIFT=18 to configure log buffer size of 256K

Skipping initcalls using KNL initcall_blacklist

- ➔ Specify a list of initcall functions to skip
- ➔ To debug built-in modules and initcalls

initcall_debug

- ➔ Use dmesg command to see initcalls and their times
- ➔ Use bootgraph.pl script to visualize dmesg output as a SVG graph to see
- ➔ initcall times in a graphical form

- ➔ Understand the boot sequence
- ➔ Debug problems
- ➔ Keep track of kernel boot times from one kernel release to another

Speed up kernel boot

- ➔ Enable driver_async_probe to speed up kernel boot
- ➔ Not all drivers support asynchronous probing

Questions?

Thank you!