

kmain() Initialization Detailed Flow

MINIX Kernel Bootstrap Process (main.c:115-328)

Source: minix/kernel

void kmain(kinfo_t *local_cbi) *main.c:115* Entry from bootloader (EBX = multiboot info)

1. BSS Sanity Check *main.c:123-125* `assert(bss_test == 0); bss_test = 1;`

2. Copy Boot Parameters *main.c:128-129* `memcpy(&kinfo, local_cbi, sizeof(kinfo));`

3. **cstart()** - Low-level Init *main.c:147*

4. **BKL_LOCK()** *main.c:149* Acquire Big Kernel Lock (SMP safety)

5. **proc_init()** *main.c:157* Clear process table, set up mappings

6. **IPCF.POOL_INIT()** *main.c:158* Initialize IPC filter pool

panic() Boot error

Module
count
match?

Yes

7. Boot Image Loop *main.c:165-272* `for (i=0; i < NR_BOOT_PROCS; ++i)`

- Get process attributes: `ip = &image[i]`
- Get process pointer: `rp = proc_addr(ip->proc_nr)`
- Copy name (tasks only), set endpoint

Kernel
task or
VM?

No

`get_priv(rp, static_priv_id(proc_nr))`

Yes

`RTS.SET(rp, RTS.NO_PRIV — VM.F, RSYS.F)`
`RTS.NO_QUANTUM)`
• Set kernel call mask

`arch_boot_proc(ip, rp)` *main.c:257*

Architecture-specific process setup

- `rp->p_rts.flags` —= `RTS.VMINHIBIT` (if not VM)
- `rp->p_rts.flags` —= `RTS.BOOTINHIBIT`
- `rp->p_rts.flags` —= `RTS.PROC_STOP`
- `rp->p_rts.flags` &= `~RTS.SLOT_FREE`

End loop (all boot processes initialized)

8. **arch_post_init()** *main.c:283* Architecture-specific post-initialization

9. Register IPC Call Names *main.c:285-290* `IPCNAME(SEND), IPCNAME(RECEIVE)`

10. **memory_init() + system_init()** *main.c:293-296* Initialize memory subsystem and sys

11. **add_mmap()** *main.c:301* Free bootstrap memory to allocator

CONFIG
SMP?

Yes

smp_init()
Initialize all CPUs

No

12. **bsp_finish_booting()** *main.c:316/324* Final boot steps, switch to Ring 3

NOT_REACHABLE; *main.c:327* `kmain()` never returns

```
// Low-level initialization
void cstart(void) {
    /* Segment setup */
    prot_init();          // GDT, IDT, TSS

    /* Boot verbosity */
    verboseboot = atoi(env_get(VERBOSEBOOTVARNAME));

    /* Clock initialization */
    init_clock();

    /* Get user stack/data limits */
    kinfo.user_sp = USR_STACKTOP_COMPACT;
    kinfo.user_end = USR_DATATOP_COMPACT;

    /* Record system info */
    kinfo.nr_procs = NR_PROCS;
    kinfo.nr_tasks = NR_TASKS;
    strcpy(kinfo.release, OS_RELEASE, ...);

    /* Check APIC/SMP config */
    config.no_apic = atoi(env_get("no_apic"));
    config.no_smp = atoi(env_get("no_smp"));

    /* Initialize interrupt handling */
    intr_init(0);

    /* Architecture-specific init */
    arch_init();
}
```

Key Data Structures

kinfo_t kinfo

- boot params
- memory map

proc[] table

- NR_PROCS entries

boot_image[]

- Kernel tasks
- System servers

priv[] table

- Privilege structs
- IPC permissions

```
void bsp_finish_booting(void) {
    /* Identify CPU features */
    cpu_identify();          // CPUID detection

    /* VM not running yet */
    vm_running = 0;

    /* Setup random number generator */
    random.random_sources = RANDOM_SOURCES;

    /* Set current process pointer to idle */
    proc_ptr = &idle_proc;
    bill_ptr = &idle_proc;

    /* Print startup banner */
    announce();              // "MINIX 3.4.0 ..."

    /* Unblock boot processes */
    for (i=0; i < NR_BOOT_PROCS - NR_TASKS; i++)
        RTS.UNSET(proc_addr(i), RTS.PROC_STOP);

    /* Initialize CPU accounting */
    cycles.accounting_init();

    /* Start timer interrupts */
    boot_cpu_init.timer(system_hz);

    /* Initialize FPU */
    fpu_init();

    /* Set SMP flags */
    cpu_set_flags(bsp_cpu_id, CPU_FLAGS_READY);
    /* Kernel can no longer allocate */
    /* Kernel may allocate privileges assigned */
    /* CRITICAL: Switch to Ring 3 */
    /* Wait for VM to be ready */
    /* NOT REACHABLE */
    RTS.BOOTINHIBIT - wait for boot completion
}
```

Process RTS Flags

`RTS.SLOT_FREE` - slot is free
`RTS.PROC_STOP` - process stopped
`RTS.NO_PRIV` - no privileges assigned
`RTS.NO_QUANTUM` - no CPU time
`RTS.VMINHIBIT` - wait for VM to be ready
`RTS.BOOTINHIBIT` - wait for boot completion