Linux Services in Embedded Systems

Embedded Linux Programming 4201152 - 2025-1S

Juan-Bernardo Gómez-Mendoza
Dept. of Electric, Electronic and Computing Engineering
Universidad Nacional de Colombia Sede Manizales

May 2, 2025

Motivation

- Linux services are essential for automation and background operations.
- In embedded systems, services often control hardware, manage communication, or maintain state.
- Understanding both user-space and kernel-space services enables full-stack control.

What is a Service?

- A service is typically a daemon process started at boot or triggered by events.
- Managed by `systemd` using `.service` unit files.
- Enables structured initialization, monitoring, and lifecycle management.

Using systemctl

- `systemctl list-units --type=service` (list all active services)
- `systemctl status <service>` (get service status)
- `systemctl is-enabled <service>` (enabled at boot?)
- `systemctl is-active <service>` (currently active?)
- `systemctl --state=running --type=service` (list all running services)
- `sudo systemctl start/stop/enable/disable <service>`

Using dmesg

- `dmesg` displays messages from the kernel ring buffer.
- Useful for diagnosing driver issues, module load errors, and hardware detection.
- Common usage:
  - `sudo dmesg | tail` (view latest messages)
  - `sudo dmesg | grep <keyword>` (search specific events)
- Essential when developing kernel-space services.

Hands-on Example: Using dmesg

- Insert a simple kernel message manually:

```
1 echo "Test message for dmesg" | sudo tee /dev/kmsg
```

- Then view it:

```
1 sudo dmesg | tail
```

- Useful for testing kernel-space event visibility.

Systemd Targets

- Targets define stages in system initialization.
- Common targets:
  - `basic.target` – minimal setup
  - `network.target` – networking available
  - `multi-user.target` – non-GUI system ready
  - `graphical.target` – GUI available
  - `default.target` – system boot target
- Link services using `WantedBy=<target>` in `.service` files.

Exploring a Simple Service

heartbeat.sh

```
#!/bin/bash
while true; do
  echo "Heartbeat: $(date)" >> /tmp/heartbeat.log
  sleep 5
done
```

Service Unit for heartbeat.sh

heartbeat.service

```
[Unit]
Description=User-space Heartbeat Logger
After=network.target

[Service]
ExecStart=/usr/local/bin/heartbeat.sh
Restart=always

[Install]
WantedBy=multi-user.target
```

Enable and Monitor the Service

- Copy script to `/usr/local/bin` and make it executable
- Copy service file to `/etc/systemd/system/heartbeat.service`
- `sudo systemctl daemon-reexec`
- `sudo systemctl enable heartbeat`
- `sudo systemctl start heartbeat`
- `tail -f /tmp/heartbeat.log`

From User-space to Kernel-space

- Services can load and interact with kernel modules.
- Kernel-space services allow low-level tasks: sensor polling, GPIO control, watchdogs.
- We'll now build a kernel module that mimics a heartbeat.

Heartbeat Kernel Module

- Uses `kthread` to run periodic task
- Logs heartbeat every 5 seconds
- Cleanly stops on module removal

Module Source Code (1/2)

heartbeat_module.c

```c
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/kthread.h>
#include <linux/delay.h>

static struct task_struct *heartbeat_thread;

static int heartbeat_fn(void *data) {
    while (!kthread_should_stop()) {
        pr_info("Heartbeat at jiffies = %lu\n", jiffies);
        ssleep(5);
    }
    return 0;
}
```

Module Source Code (2/2)

heartbeat_module.c (cont.)

```
static int __init heartbeat_init(void) {
    heartbeat_thread = kthread_run(heartbeat_fn, NULL, "heartbeat_kthread");
    return IS_ERR(heartbeat_thread) ? PTR_ERR(heartbeat_thread) : 0;
}

static void __exit heartbeat_exit(void) {
    if (heartbeat_thread)
        kthread_stop(heartbeat_thread);
}

module_init(heartbeat_init);
module_exit(heartbeat_exit);
MODULE_LICENSE("GPL");
```

Makefile

Makefile

```
obj-m += heartbeat_module.o

all:
  make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:
  make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

Testing the Kernel Module

- `make`
- `sudo insmod heartbeat_module.ko`
- `dmesg | tail -f`
- `sudo rmmod heartbeat_module`

Systemd Unit for Kernel Module

heartbeat.service

```
[Unit]
Description=Kernel Heartbeat Module
After=multi-user.target

[Service]
Type=oneshot
ExecStart=/sbin/insmod /lib/modules/<kernel>/heartbeat_module.ko
ExecStop=/sbin/rmmod heartbeat_module
RemainAfterExit=yes

[Install]
WantedBy=multi-user.target
```

Key Takeaways

- Use `systemctl` to explore and manage services.
- Understand how targets affect when services run.
- Create and control user-space and kernel-space services.
- Use kthreads for periodic kernel tasks.