Laurentiu Mihalcea

Iuliana Prodan

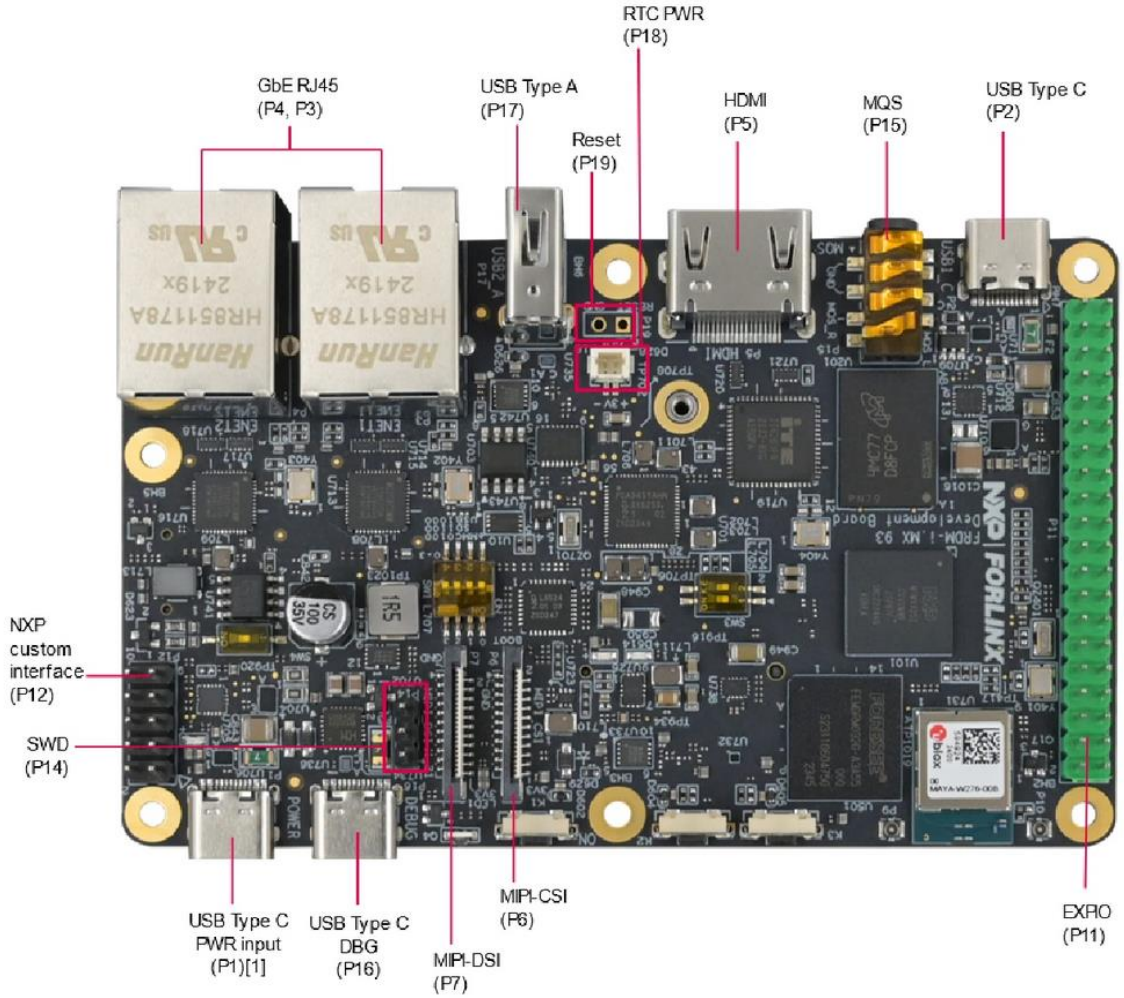
Daniel Baluta

00

Day 1: Introduction to Embedded Linux kernel development

Secondary header

Hardware support - NXP i.MX93 FRDM board

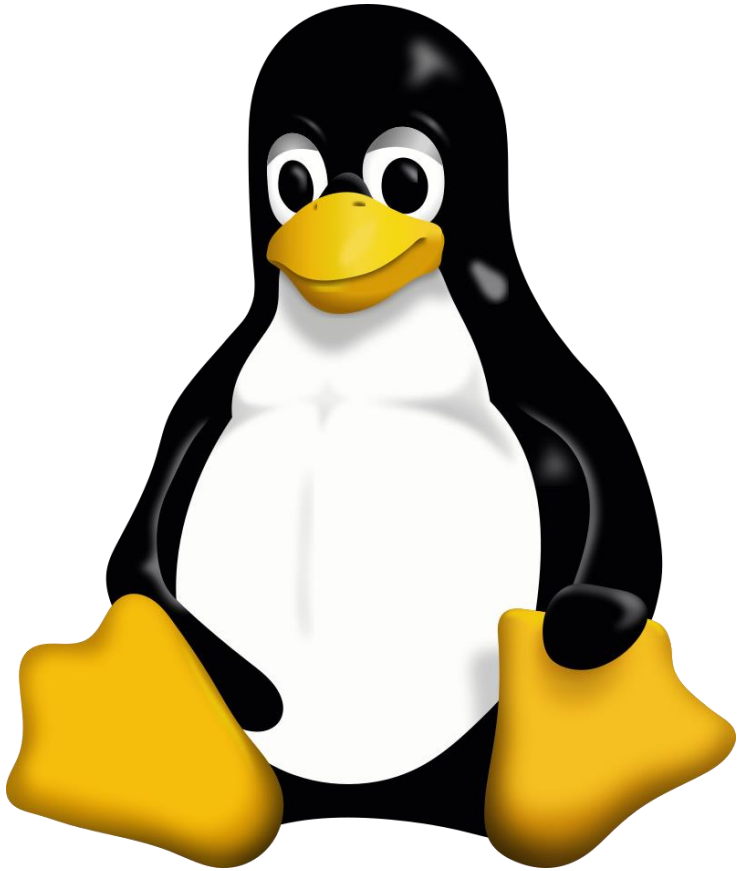


- SoC: NXP i.MX93 FRDM board
  - CPUs: 2× ARM Cortex-A55 @1.7Ghz
  - CPUs: 1 x ARM Cortex-M33 @250M
  - Memory: Up to 2G LPPDR4
  - Storage: 32G eMMC
  - **I/O**: USB, UART, GPIO, I2C, SPI, CAN, Ethernet, MIPI-CSI
  - Audio: MQS

Notes and sources section

Embedded Linux vs Desktop Linux

- Purpose and Use cases
  - General purpose vs Specialized
  - Ubuntu, Fedora, Debian vs Yocto, buildroot, openwrt
- Hardware requirements
  - Power consumption, memory footprint
- Operating system design
  - Full fledged OS vs stripped down version of Linux
- System on a Chip vs Discrete Component System



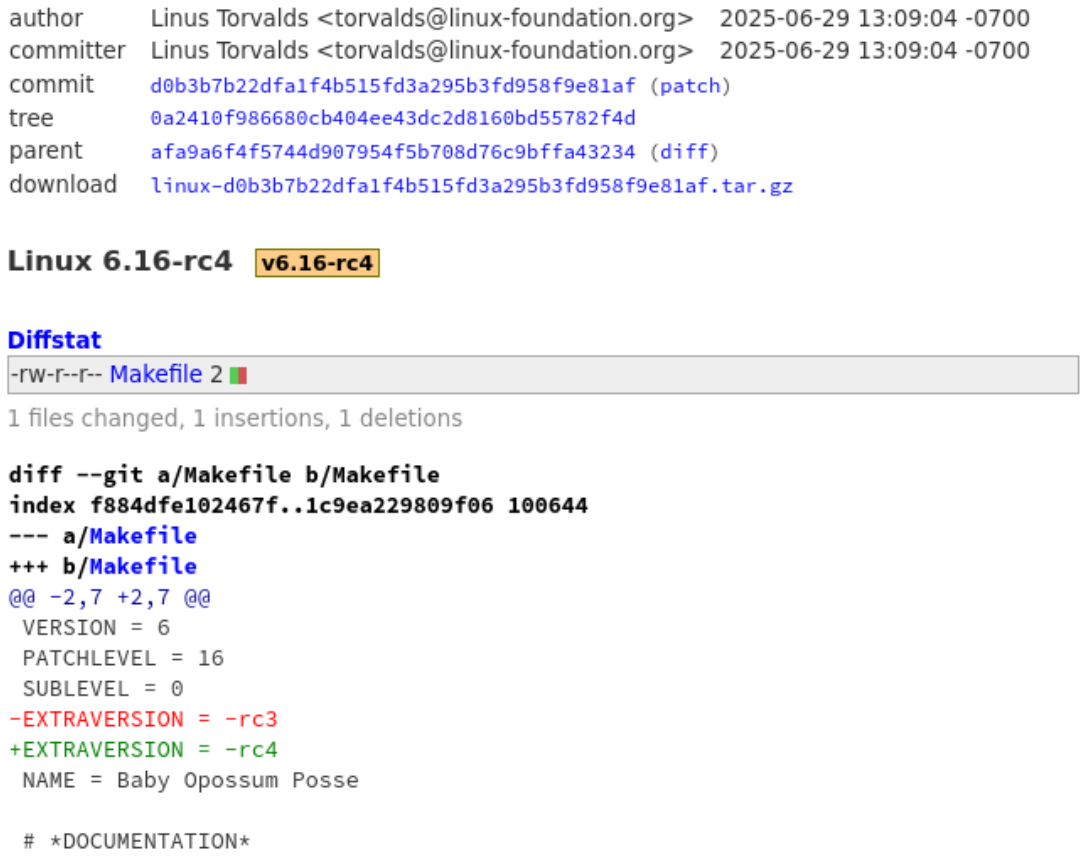
Notes and sources section

Embedded Linux usage

- Consumer electronics
  - Smart TVs and Set-Top boxes
  - Smartphones and Tables
- Wearables
  - Smartwatches and fitness trackers
- Automotive Systems
  - In-Vehicle Infotainment (IVI)
  - Advanced Driver Assistance Systems (ADAS)
- Internat of Things (IoT)
  - Smarthome devices
- Industrial Automation, Medical Devices, Energy and Utilities

Linux kernel

- Started by Linus Torvalds, in 1991
- Split into sub-subsystems handled by maintainers
- https://kernel.org/
- Development
  - Current mainline version: 6.16
  - Releases every 9-10 weeks
  - Long Term Support
- Stats:
  - Version 6.16 has around 40M lines of code
  - Every release they are around 2000 contributors



author Linus Torvalds <torvalds@linux-foundation.org> 2025-06-29 13:09:04 -0700
committer Linus Torvalds <torvalds@linux-foundation.org> 2025-06-29 13:09:04 -0700
commit d0b3b7b22dfa1f4b515fd3a295b3fd958f9e81af (patch)
tree 0a2410f986680cb404ee43dc2d8160bd55782f4d
parent afa9a6f4f5744d907954f5b708d76c9bffa43234 (diff)
download linux-d0b3b7b22dfa1f4b515fd3a295b3fd958f9e81af.tar.gz

Linux 6.16-rc4 v6.16-rc4

Diffstat

-rw-r--r-- Makefile 2

1 files changed, 1 insertions, 1 deletions

```
diff --git a/Makefile b/Makefile
index f884dfe102467f..1c9ea229809f06 100644
--- a/Makefile
+++ b/Makefile
@@ -2,7 +2,7 @@
 VERSION = 6
 PATCHLEVEL = 16
 SUBLEVEL = 0
-EXTRAVERSION = -rc3
+EXTRAVERSION = -rc4
 NAME = Baby Opossum Posse

 # *DOCUMENTATION*
```


Notes and sources section

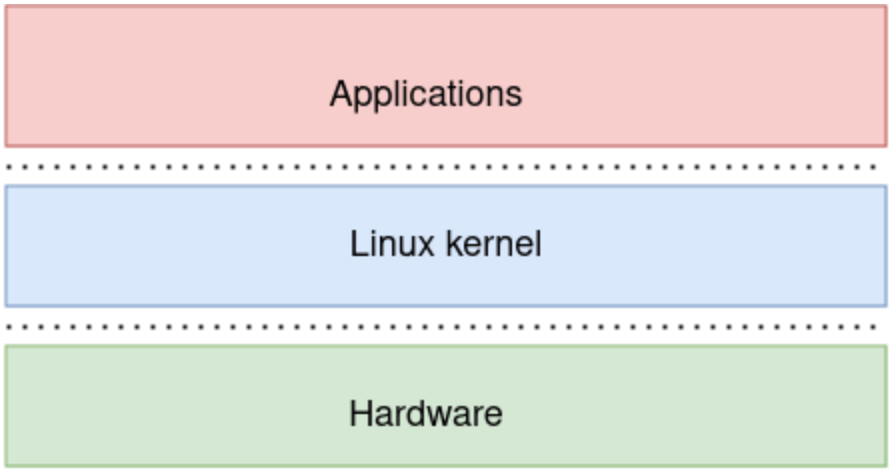
Linux kernel contributors

Most active 6.15 employers

By changesets			By lines changed		
Intel	1755	12.0%	AMD	125923	14.9%
(Unknown)	1302	8.9%	(Unknown)	97908	11.5%
Google	983	6.7%	Intel	94150	11.1%
(None)	930	6.4%	Google	67461	8.0%
Red Hat	889	6.1%	IBM	48682	5.7%
AMD	881	6.0%	(None)	45049	5.3%
Linaro	645	4.4%	Red Hat	43981	5.2%
SUSE	549	3.8%	Qualcomm	34014	4.0%
Meta	493	3.4%	Meta	26182	3.1%
NVIDIA	370	2.5%	Microsoft	19431	2.3%
Huawei Technologies	370	2.5%	Linaro	16389	1.9%
Renesas Electronics	367	2.5%	NVIDIA	16191	1.9%
Qualcomm	319	2.2%	SUSE	15175	1.8%
Arm	301	2.1%	Huawei Technologies	14136	1.7%
Linutronix	296	2.0%	Xilinx	12961	1.5%
Oracle	286	2.0%	Collabora	11640	1.4%
IBM	282	1.9%	Arm	9357	1.1%
Microsoft	259	1.8%	NXP Semiconductors	8857	1.0%
(Consultant)	180	1.2%	Rockchip	8085	1.0%
NXP Semiconductors	179	1.2%	BayLibre	8037	0.9%

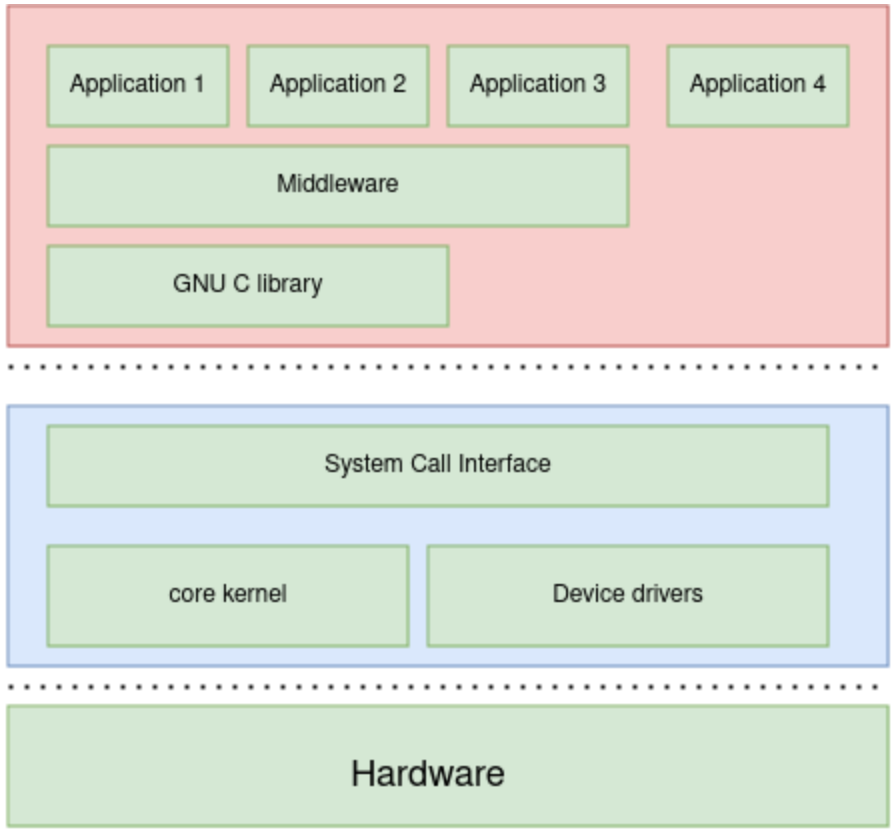
Linux kernel roles (1)

- Resource management
  - Processes, files, memory, scheduling
- Hardware management
  - Device drivers
  - Allows user space apps to use the hardware
- IPC
- Security



Linux kernel roles (2)

- Applications rely on kernel for services
  - functionalities are implemented via libraries
- User – kernel communications
  - via System Calls
- Linux kernel is monolithic
  - Everything happens in a single executable **Image**
  - **..but it has loadable modules!**



Clone the Linux kernel tree

- git.kernel.org
- Linux kernel is written in C
- Compiled with GCC
- There is also some assembly code
- Rust support
- Development *happens on email*
  - *git send-email*
- Distributed git repo
  - Each maintainer with its own tree
  - Linus Torvalds does the release

Exploring the source code

- vim
- Visual Studio Code
- https://elixir.bootlin.com/linux/latest/source

/ include / linux / sched.h

```
742  #ifdef CONFIG_KMAP_LOCAL
743          int                             idx;
744          pte_t                           pteval[KM_MAX_IDX];
745  #endif
746  };
747
748  struct task_struct {
749  #ifdef CONFIG_THREAD_INFO_IN_TASK
750          /*
751           * For reasons of header soup (see current_thread_info()), this
752           * must be the first element of task_struct.
753           */
754          struct thread_info              thread_info;
755  #endif
756          unsigned int                    __state;
```

Compiling the Linux kernel source code

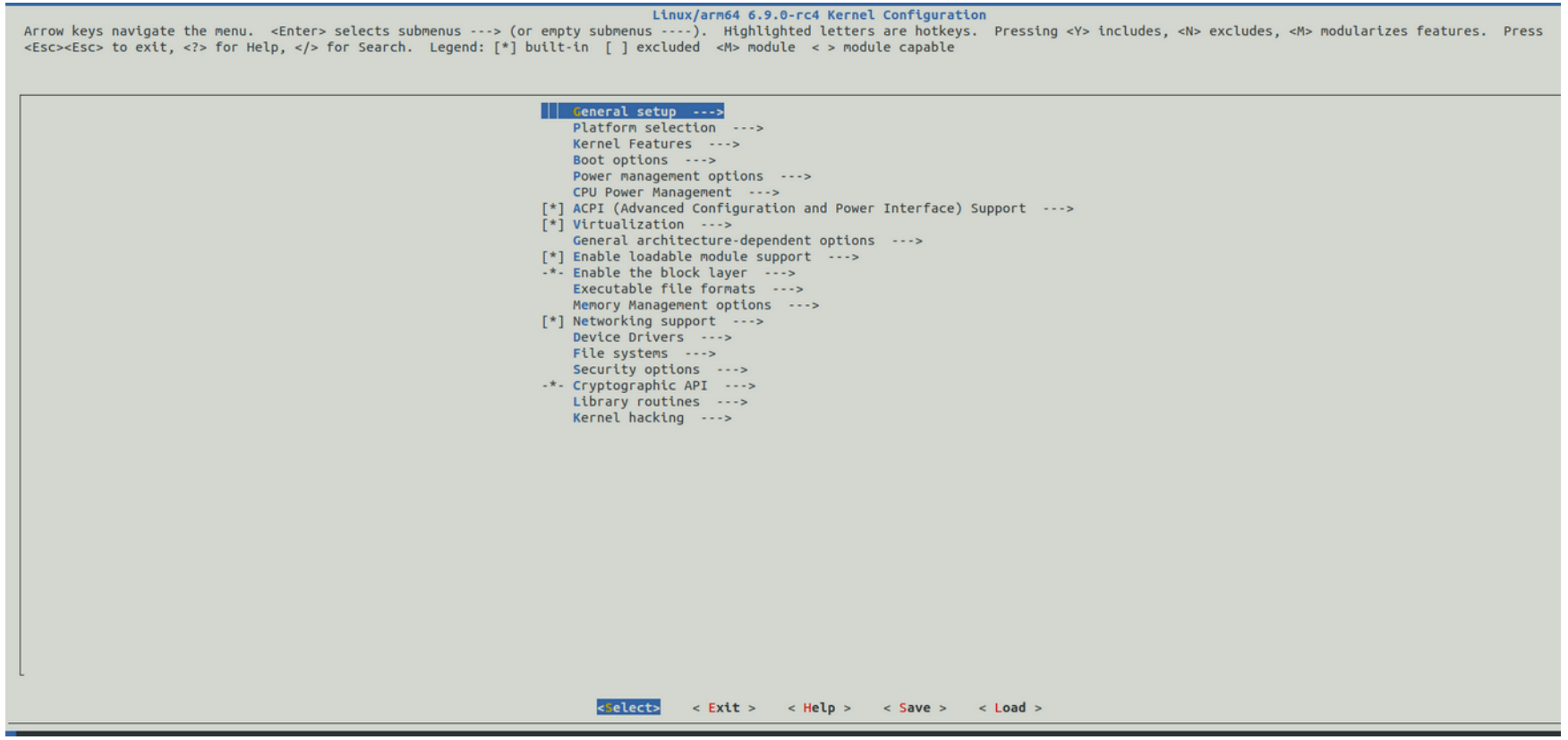
- Cross-compilation
  - We use x86 as host machine but compile for arm64 target
  - sudo apt-get install gcc-aarch64-linux-gnu
- Specify ARCH
  - ARCH=arm64
- Specify CROSS_COMPILE
  - CROSS_COMPILE=aarch64-linux-gnu-

Initial configuration

- Linux kernel is huge!
- We need to be able to select parts of the code to be compiled in
- Configuration symbols (e.g CONFIG_NET)
- Default configuration
  - arch/x86/configs
  - Arch/arm64/configs
- Configuration symbols
  - **Y**, code is compiled inside the Linux kernel image
  - **M,** code is compiled as a separate Linux kernel module
  - **N,** code is not considered for compilation
  - ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- make imx_v8_defconfig

Create your own configuration

- ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- make menuconfig



Kernel compilation & output binaries

- ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- **make –j4**
- This will result in:
  - arch/arm64/boot/**Image** - Linux kernel image
  - Arch/arm64/boot/dts/freescale/
  - Linux kernel modules scattered around the tree
- Install Linux kernel modules
  - INSTALL_MOD_PATH=/path/to/modules make modules_install

Booting the kernel

- Uboot
  - Bootloader used to bootstrap the system, load the DTB and then start the kernel
- Linux kernel
  - Image
  - DTB
  - Modules
- Root file system

- … and now go to **Practical Lab exercises**

Simple "Hello World" kernel module

```c
// SPDX-License-Identifier: GPL-2.0

#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/module.h>

static int __init hello_init(void)
{
        printk(KERN_INFO "Hello from kernel space!\n");
        return 0;
}

static void __exit hello_exit(void)
{
        pr_info("Goodbye from kernel space!\n");
}

module_init(hello_init);
module_exit(hello_exit);

MODULE_LICENSE("GPL");
MODULE_AUTHOR("NXP Linux Kernel Summer School");
MODULE_DESCRIPTION("A simple Hello World kernel module");
```

Linux kernel logging

- Mechanism
  - **printk**, similar to printf but outputs to kernel buffer
- Log access tools
  - dmesg
  - journalctl
- Log levels
  - 0 (emerg) to 7 (debug)
  - printk(KERN_INFO "Hello World");
  - pr_info("Hello World")

Build / Load the Linux kernel module

- Build
  - source the env: ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu-
  - make menuconfig
  - make M=drivers/lkss/lab1 modules
- On the target board
  - modinfo
  - modprobe / insmod
  - rmmod

Kernel errors: Oops vs Panic

- Oops
  - A non-fatal error detected in kernel space
  - System may continue running
  - Prints diagnostic info: stack trace, registers
- Panic
  - A fatal error detected in kernel space
  - System will halt

Kernel API: Timers

- allows scheduling activities in the future
- **Jiffies,** stores the number of timer ticks
- Timer API
  - struct timer_list
  - timer_setup
  - mod_timer
  - del_timer_sync

```
9   #define TIMER_INTERVAL_MS 1000  /* 1000 milliseconds = 1 second */
10
11  static struct timer_list my_timer;
12
13  /* Timer callback */
14  static void my_timer_callback(struct timer_list *t)
15  {
16          pr_info("Timer callback executed at jiffies=%lu\n", jiffies);
17          /* TODO schedule the timer again, using `mod_timer` function */
18
19  }
20
21  static int __init lkss_timer_init(void)
22  {
23          pr_info("Initializing the timer module\n");
24
25          /* Initialize the timer */
26          timer_setup(&my_timer, my_timer_callback, 0);
27
28          /* schedule the timer for the first time */
29          mod_timer(&my_timer, jiffies + msecs_to_jiffies(TIMER_INTERVAL_MS));
30          return 0;
31  }
32
33  static void __exit lkss_timer_exit(void)
34  {
35          /* Delete the timer if still active */
36          timer_delete_sync(&my_timer);
37
38          pr_info("Timer module exited\n");
39  }
```

Platform drivers & Device tree

- Platform drivers
  - Used for non-discoverable hardware
  - Handle platform devices described in device tree
  - Key API
    - Probe
    - Remove
  - Device tree
    - A data structure used to describe hardware
    - Used at boot to inform the kernel about the HW hierarchy
    - Kernel matches compatible string with platform driver