Linux kernel summer school

Laurentiu Mihalcea

Iuliana Prodan

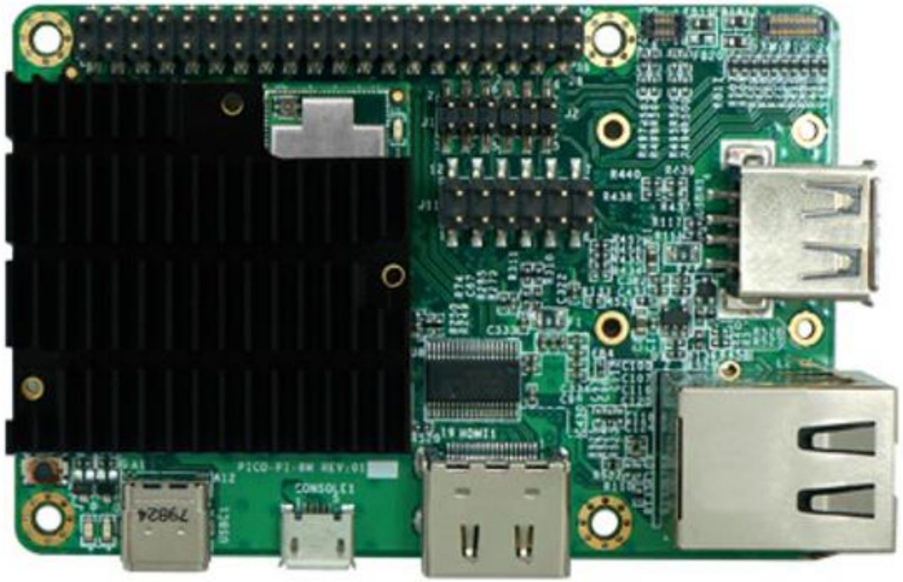
Daniel Baluta

00

Day 1: Introduction to Embedded Linux kernel development

Secondary header

Hardware support - Technexion PICO-PI-IMX8M



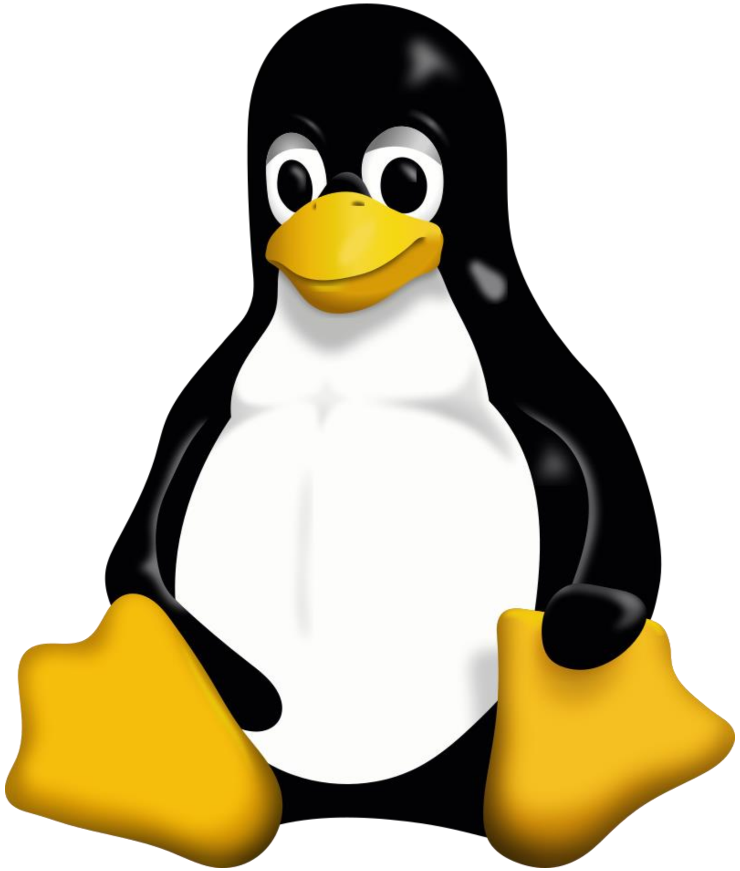
- SoC: NXP i.MX8M Quad
  - CPUs: 4× ARM Cortex-A53 @ 1.3Ghz
  - GPU: Vivante GC7000Lite
  - Memory: Up to 4G LPPDR4
  - Storage: 16G eMMC
  - Micro USB debug
  - Ethernet
  - Wi-Fi

https://www.technexion.com/wp-content/uploads/2022/09/product-brief_pico-pi-imx8m.pdf

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



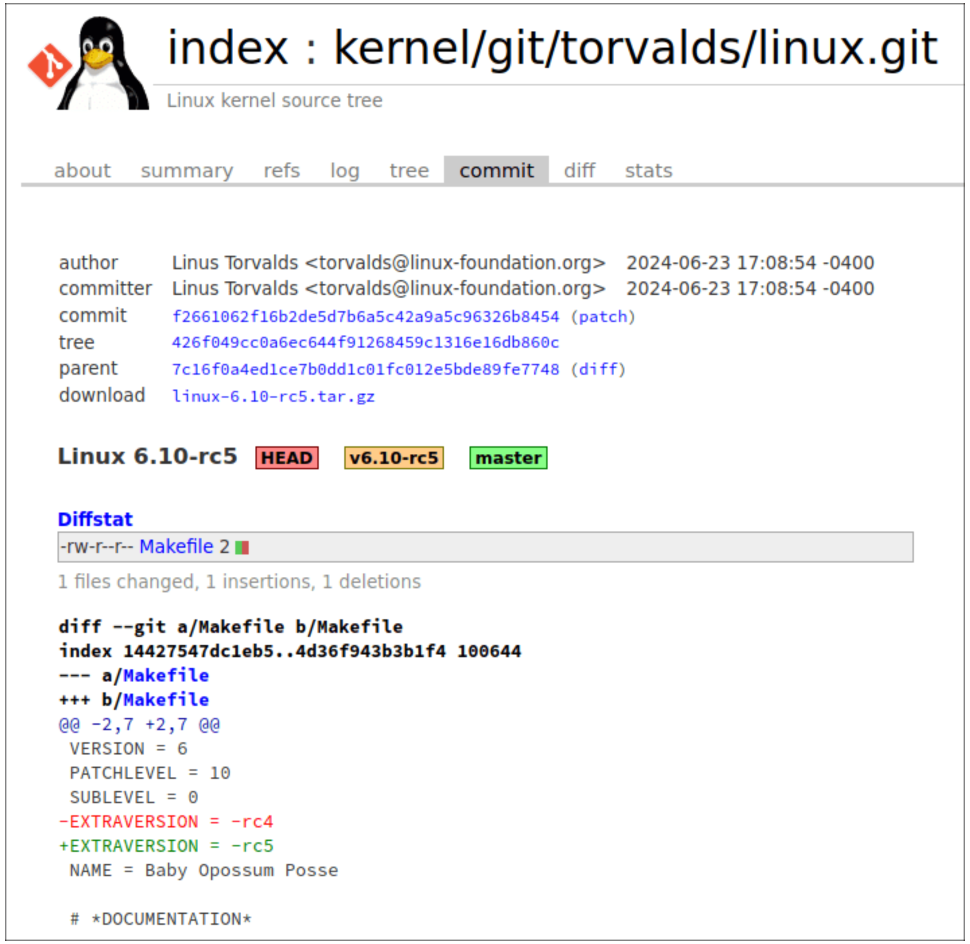
Notes and sources section

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: 6.10
  - ○ Releases every 9-10 weeks
  - ○ Long Term Support
- Stats:
  - ○ Version 6.8 has around 38M lines of code
  - ○ Every release they are around 2000 contributors



index : kernel/git/torvalds/linux.git

Linux kernel source tree

author	Linus Torvalds <torvalds@linux-foundation.org>	2024-06-23 17:08:54 -0400
committer	Linus Torvalds <torvalds@linux-foundation.org>	2024-06-23 17:08:54 -0400
commit	f2661062f16b2de5d7b6a5c42a9a5c96326b8454 (patch)	
tree	426f049cc0a6ec644f91268459c1316e16db860c	
parent	7c16f0a4ed1ce7b0dd1c01fc012e5bde89fe7748 (diff)	
download	linux-6.10-rc5.tar.gz	

Linux 6.10-rc5 HEAD v6.10-rc5 master

Diffstat

-rw-r--r-- Makefile 2

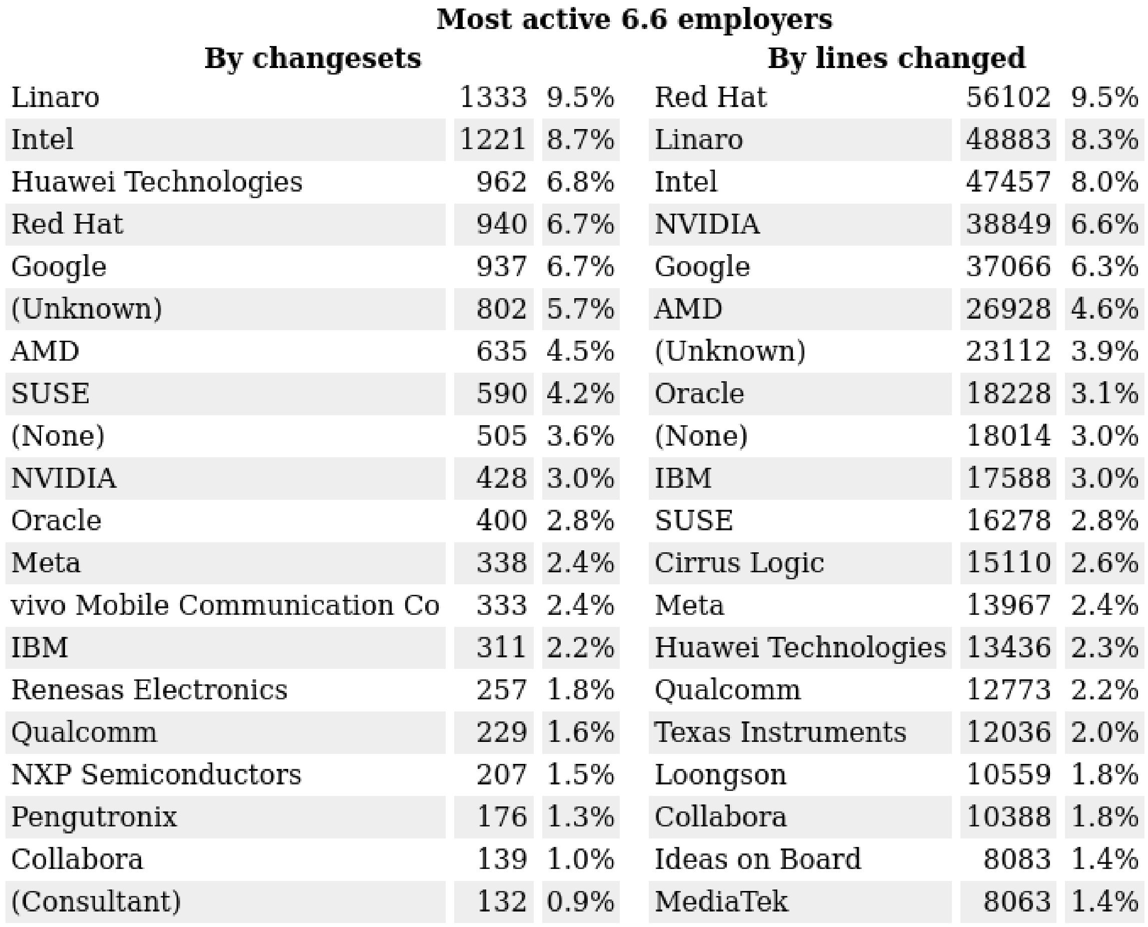
1 files changed, 1 insertions, 1 deletions

```
diff --git a/Makefile b/Makefile
index 14427547dc1eb5..4d36f943b3b1f4 100644
--- a/Makefile
+++ b/Makefile
@@ -2,7 +2,7 @@
 VERSION = 6
 PATCHLEVEL = 10
 SUBLEVEL = 0
-EXTRAVERSION = -rc4
+EXTRAVERSION = -rc5
 NAME = Baby Opossum Posse

 # *DOCUMENTATION*
```

Notes and sources section

Linux kernel contributors

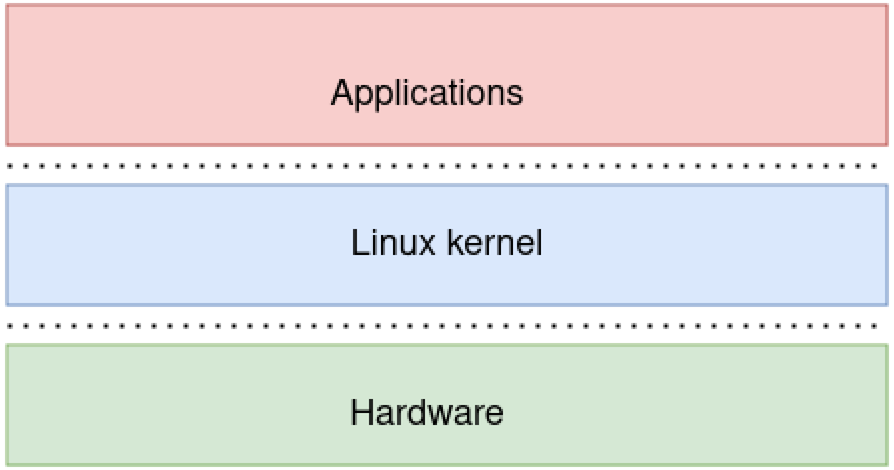


Most active 6.6 employers

By changesets			By lines changed		
Linaro	1333	9.5%	Red Hat	56102	9.5%
Intel	1221	8.7%	Linaro	48883	8.3%
Huawei Technologies	962	6.8%	Intel	47457	8.0%
Red Hat	940	6.7%	NVIDIA	38849	6.6%
Google	937	6.7%	Google	37066	6.3%
(Unknown)	802	5.7%	AMD	26928	4.6%
AMD	635	4.5%	(Unknown)	23112	3.9%
SUSE	590	4.2%	Oracle	18228	3.1%
(None)	505	3.6%	(None)	18014	3.0%
NVIDIA	428	3.0%	IBM	17588	3.0%
Oracle	400	2.8%	SUSE	16278	2.8%
Meta	338	2.4%	Cirrus Logic	15110	2.6%
vivo Mobile Communication Co	333	2.4%	Meta	13967	2.4%
IBM	311	2.2%	Huawei Technologies	13436	2.3%
Renesas Electronics	257	1.8%	Qualcomm	12773	2.2%
Qualcomm	229	1.6%	Texas Instruments	12036	2.0%
NXP Semiconductors	207	1.5%	Loongson	10559	1.8%
Pengutronix	176	1.3%	Collabora	10388	1.8%
Collabora	139	1.0%	Ideas on Board	8083	1.4%
(Consultant)	132	0.9%	MediaTek	8063	1.4%

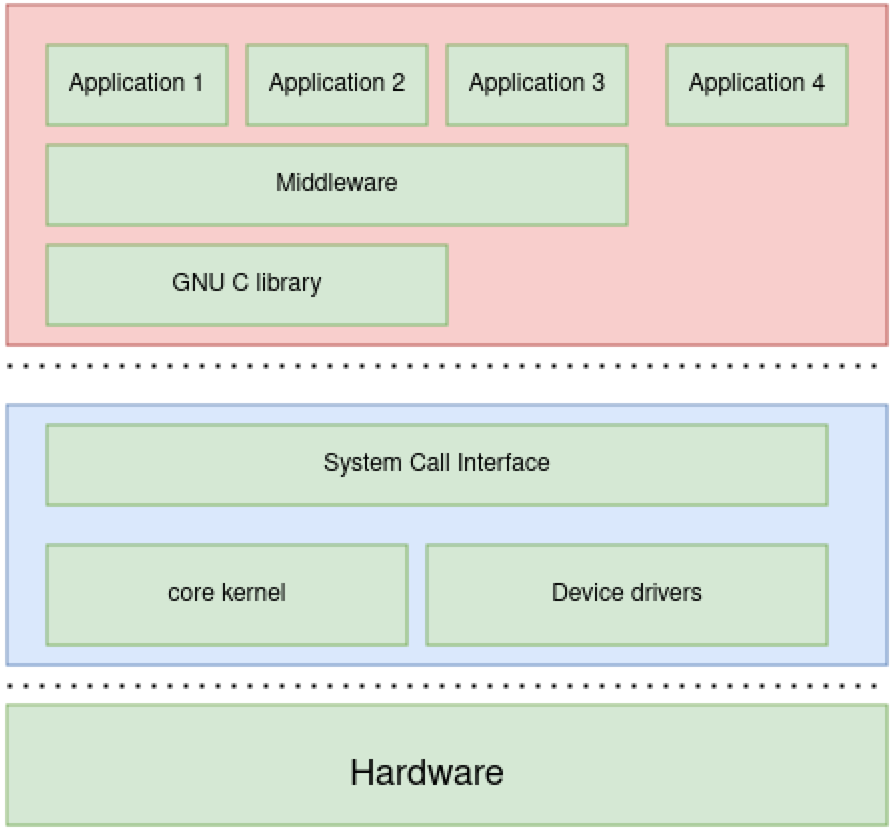
Linux kernel roles (1)

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



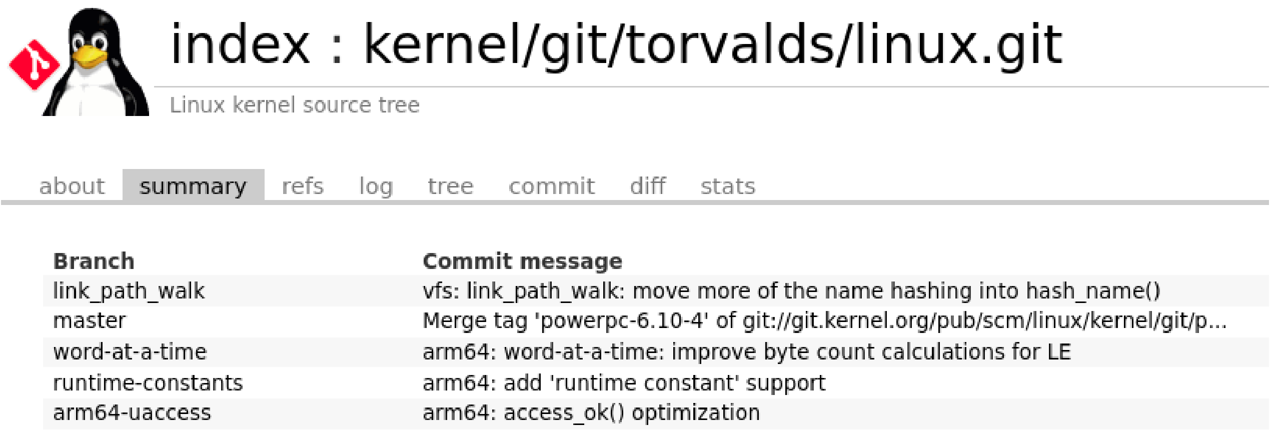
Linux kernel roles (2)

- 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



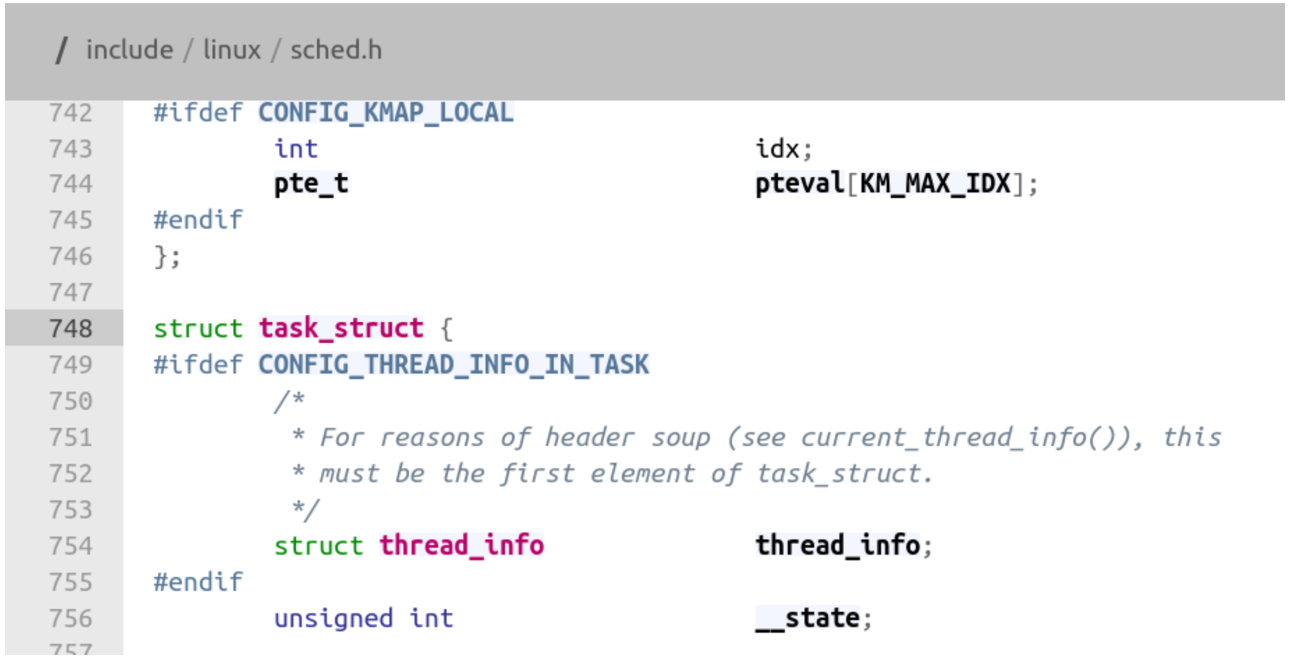
index : kernel/git/torvalds/linux.git

Linux kernel source tree

Branch	Commit message
link_path_walk	vfs: link_path_walk: move more of the name hashing into hash_name()
master	Merge tag 'powerpc-6.10-4' of git://git.kernel.org/pub/scm/linux/kernel/git/p...
word-at-a-time	arm64: word-at-a-time: improve byte count calculations for LE
runtime-constants	arm64: add 'runtime constant' support
arm64-uaccess	arm64: access_ok() optimization

Exploring the source code

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



/ include / linux / sched.h

```
#ifdef CONFIG_KMAP_LOCAL
        int                             idx;
        pte_t                           pteval[KM_MAX_IDX];
#endif
};

struct task_struct {
#ifdef CONFIG_THREAD_INFO_IN_TASK
        /*
         * For reasons of header soup (see current_thread_info()), this
         * must be the first element of task_struct.
         */
        struct thread_info              thread_info;
#endif
        unsigned int                    __state;
```

Compiling the Linux kernel source code

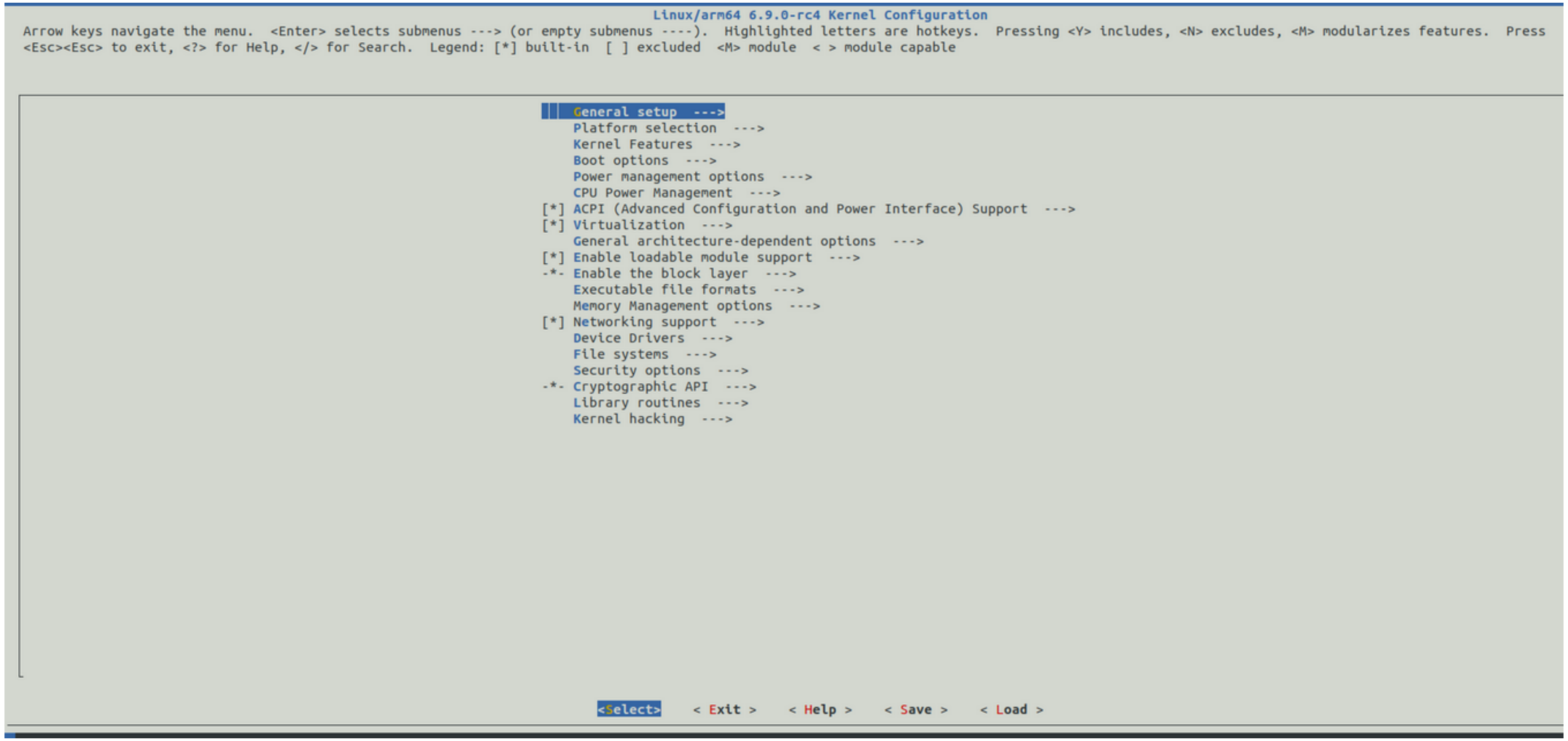
- Cross-compilation
  - We use x86 as host machine but compile for arm64 target
  - sudo apt-get install gcc-aarch64-linux-gnu
- Specify ARCH
  - ARCH=arm64
- Specify CROSS_COMPILE
  - CROSS_COMPILE=aarch64-linux-gnu-

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



```
                                           Linux/arm64 6.9.0-rc4 Kernel Configuration
Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes features.  Press
<Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

                                    General setup  --->
                                    Platform selection  --->
                                    Kernel Features  --->
                                    Boot options  --->
                                    Power management options  --->
                                    CPU Power Management  --->
                                [*] ACPI (Advanced Configuration and Power Interface) Support  --->
                                [*] Virtualization  --->
                                    General architecture-dependent options  --->
                                [*] Enable loadable module support  --->
                                -*- Enable the block layer  --->
                                    Executable file formats  --->
                                    Memory Management options  --->
                                [*] Networking support  --->
                                    Device Drivers  --->
                                    File systems  --->
                                    Security options  --->
                                -*- Cryptographic API  --->
                                    Library routines  --->
                                    Kernel hacking  --->

                                       <Select>    < Exit >    < Help >    < Save >    < Load >
```

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**