

Embedded Linux System



Out Line

- ▶ CH01 Introduction to Embedded System
- ▶ CH02 Embedded Linux (1)
- ▶ CH03 Embedded Linux (2)
- ▶ CH04 Basic Software And Tool
- ▶ CH05 Cross-compile Toolchain
- ▶ CH06 Introduction to Bootloader (u-boot)
- ▶ CH07 Embedded Linux Kernel
- ▶ CH08 RootFS
- ▶ CH09 Linux Device Driver
- ▶ CH10 Control Hardware Driver

Introduction to Embedded System



Embedded System

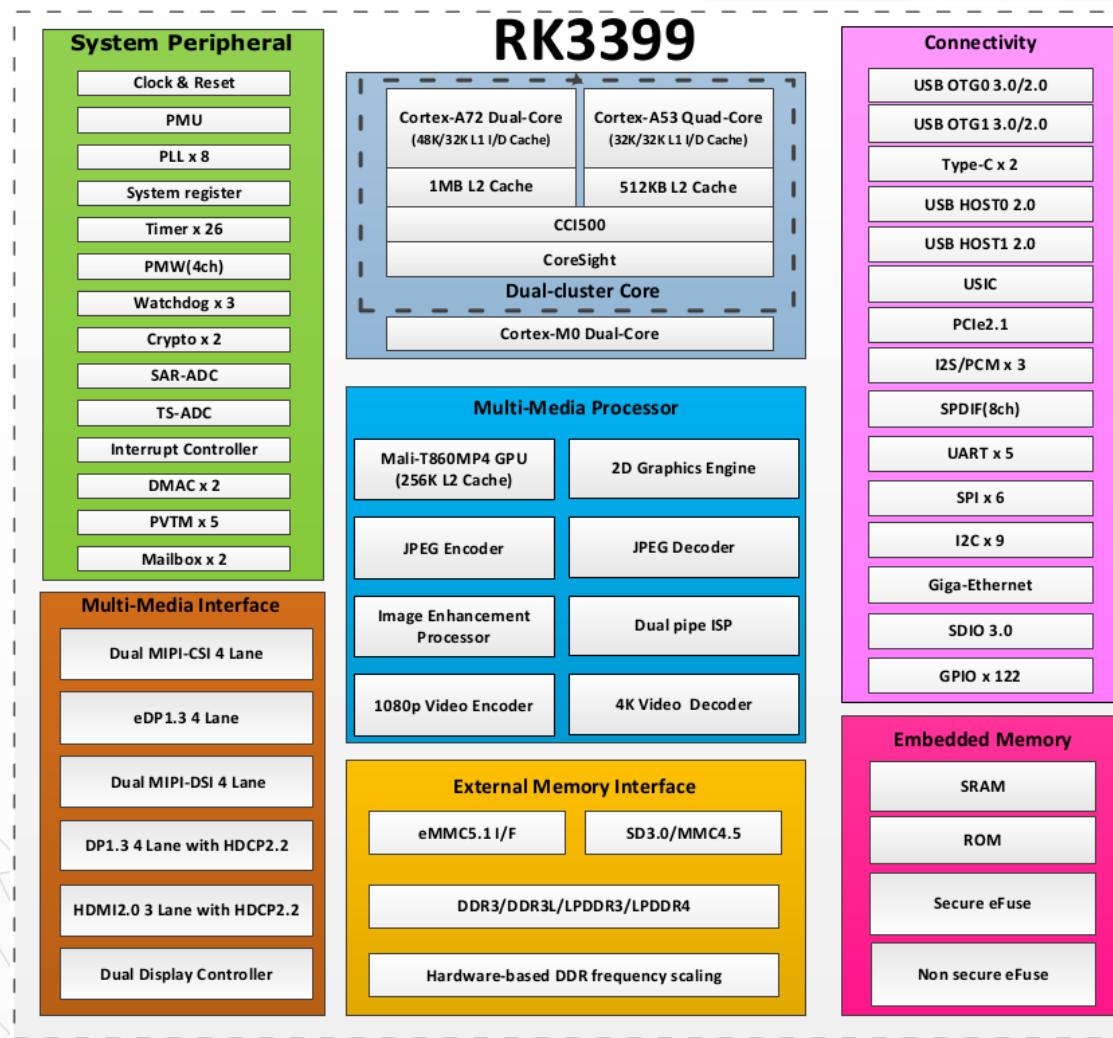
- An embedded system
 - combination of computer hardware and software
 - specifically designed for a particular function
- Applications
 - Mobile phone
 - Digital camera
 - Smart TV
 - Navigation system



Feature

- Designed to do some specific task
 - Low power
 - Small size
 - Special operating ranges
 - Low cost
- Install OS ?

SOC RK3399





SOC – System On Chip

- Processor
 - ARM, X86, MIPS
- RAM
 - 8MB ~ 4 GB
- Storagee
 - Nand, Nor flash
 - SD/MMC/eMMC
- System Bus
 - AMBA, AHB, APB, AXI ...



SOC – System On Chip

- Communication
 - I2C, I2S, USB, PCI/PCIe ...
- Media system
 - JPEG, H.264 ..
- System component
 - DMA, RTC ..



Embedded Linux ?

Embedded Linux is the usage of the
Linux kernel and various
open-source components in
embedded systems
(from Free Electrons)

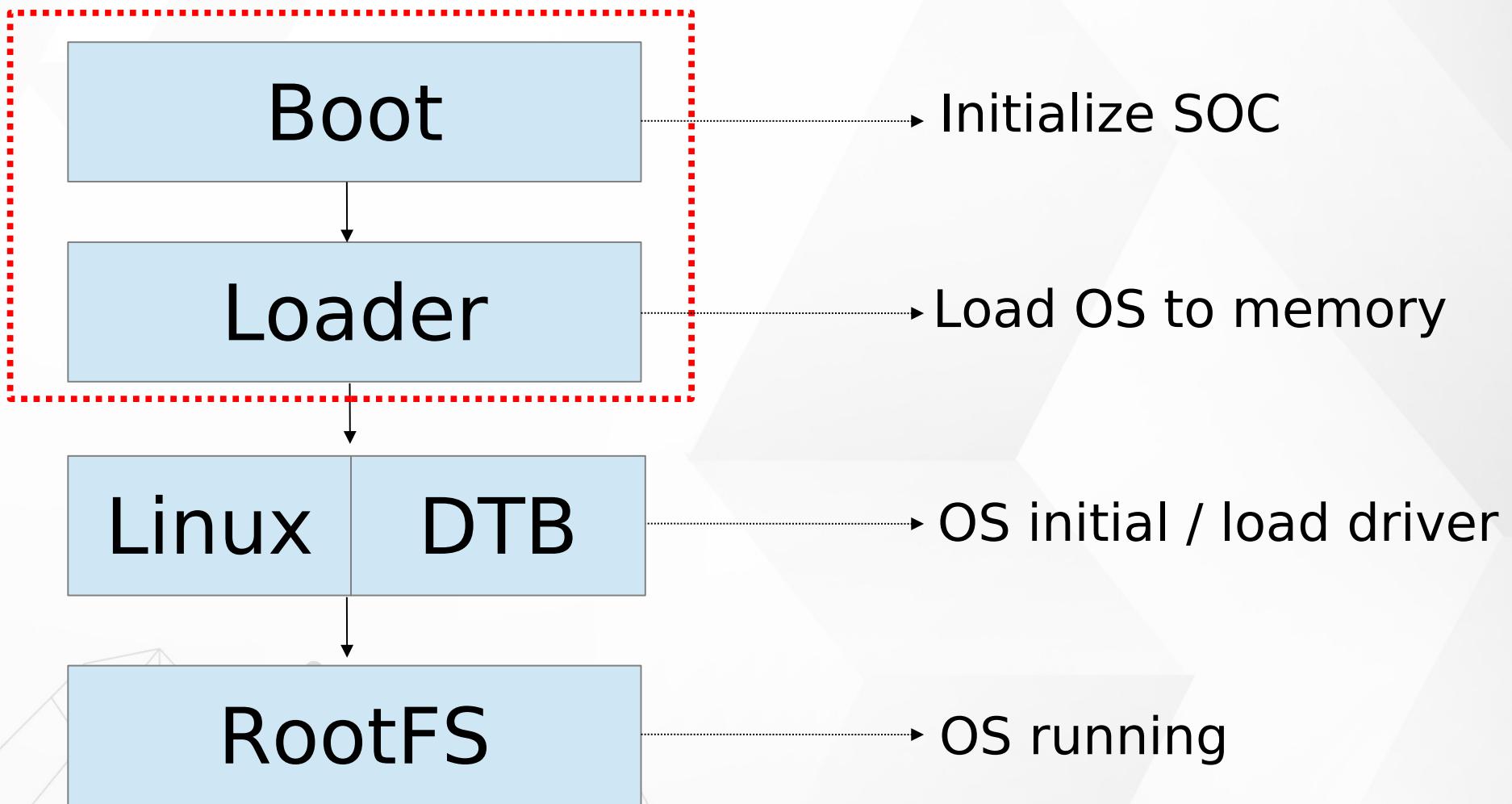


Linux Advantages

- Re-use components
- Quickly design and develop complicated products
- No need to re-develop components
 - TCP/IP stack, USB stack, PCI stack ...
- Allow you modify components



Embedded Linux Booting





Software Components

- Cross-compilation Toolchain
- Boot-loader
- Linux Kernel, DeviceTree
- RootFS
- C library
- Libraries and applications
- BSP (Board Support Package)

Develop Environment



Develop Environment

- Host PC (Linux)
- Toolchain
- BSP (Board Support Package)
- Target Board EVB (RockPi4)



BSP

- Board Support Package
- From chip vendor
 - Distribution
 - Bootloader
 - Linux Kernel
 - Device Driver
 - Rootfs

RK3399 Debian BSP (1)

- RK3399 Debian BSP
 - <https://wiki.radxa.com/Rockpi4>

Setup/Quick start

- Getting started with your ROCK Pi 4, including what you need and how to get it booted.
- GPIO pinout
- Backup and Restore your SD card or eMMC module
- How to mount SSD with M2 extension board

Installation

Installing an operating system on your ROCK PI 4, including microSD card, eMMC module, USB drive and M.2 NVME SSD,

- Install Rockchip Flashing tools
- Install image to eMMC from USB OTG Port
- Install on microSD card
- Install on eMMC module
- Install on SPI Flash
- [Install on USB drive\(wip\)](#)
- Install on M.2 NVME SSD

> More...

[Expand]

Development

Information about Linux and Android development, this is mostly for developers.

- [USB Installation](#) - How to use PC tools to install image on ROCK Pi 4.
- [Serial Console](#) - Serial console on GPIO header
- [Build Debian](#) - Build and generate Debian image
- [Build vendor kernel\(Rockchip 4.4\)](#) - Build vendor kernel for ROCK Pi 4
- [Build Android \(nougat\) TV](#) - Build Android for ROCK Pi 4
- [Build Yocto](#) - Build Yocto for ROCK Pi 4

> More...

[Expand]

Hardware

Technical specifications about the ROCK Pi 4 hardware, including Wi-Fi, display, camera, etc.

- [Blog post](#) from Radxa Team introducing the ROCK Pi hardware design
- [ROCK Pi 4](#) - Introduction of the ROCK Pi 4 hardware
- [Display](#)
- [Camera module](#)
- [Device Tree Overlays](#) - Use other HAT

> More...

[Expand]

Working With Linux

Fundamental Linux usage for beginners and more advanced information for power users.

- [Debian Desktop](#)
- [Ubuntu Server](#)
- [Linux system runs on M.2 NVME SSD](#)
- [Radxa APT](#)
- [Docker](#)
- [Samba](#)

> More...

[Expand]

Working With Android

Fundamental Android usage for beginners and more advanced information for power users.

- [Android7 Tablet\(Support Raspberry Pi official 7" Display\)](#)
- [Android7 TV](#)
- [Android9 Tablet](#)
- [Android9 TV](#)
- [Android9 Run on M.2 NVME SSD](#)
- [Android9 Mraa API](#)
- [Android10 Tablet](#)
- [Android11](#)
- [Solve Google Play Device is not Play Protect certified issue](#) ↗



RK3399 Debian BSP (2)

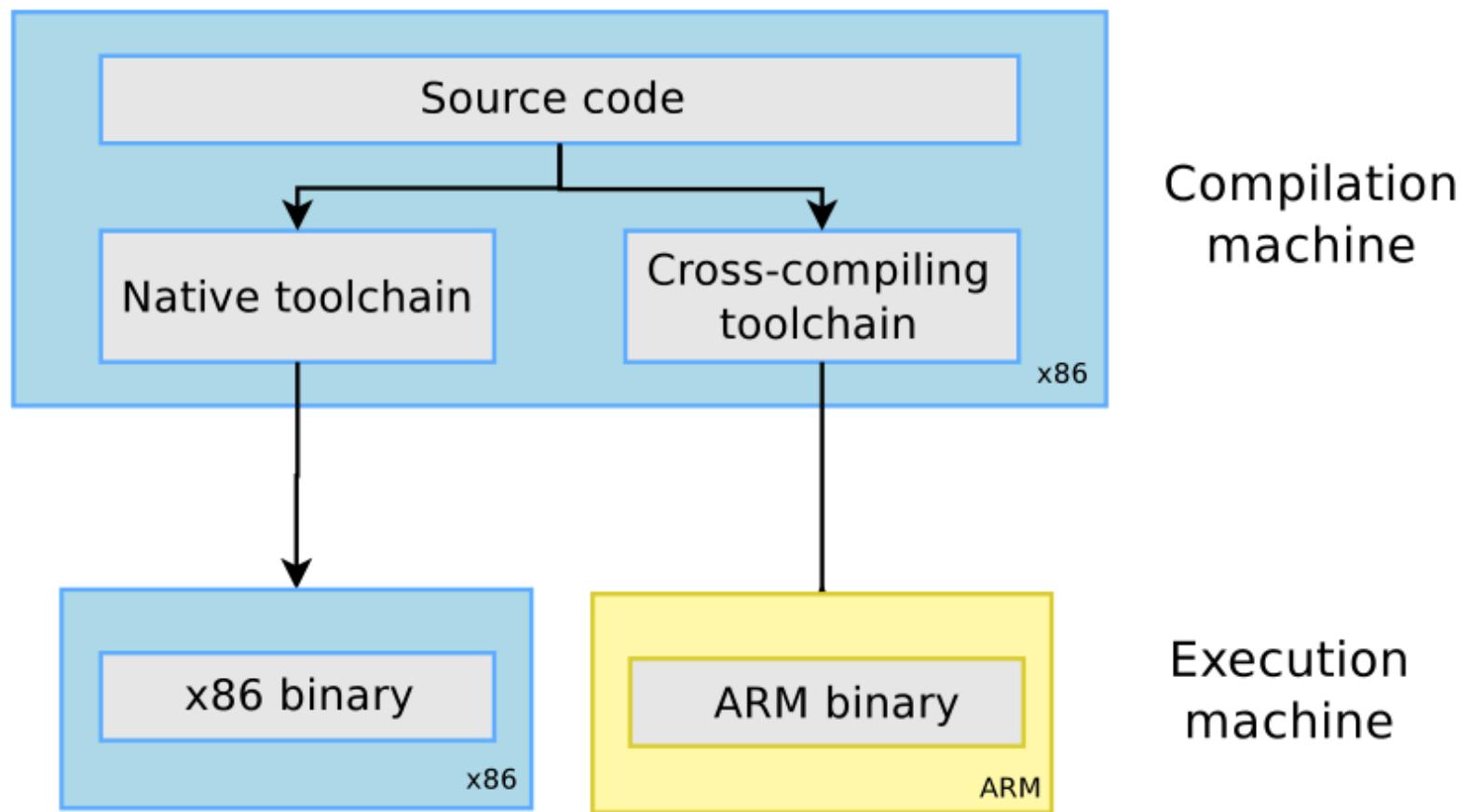
- Boot-Loader
 - RKBin, U-Boot
- Kernel
 - Linux Kernel source
- Rootfs
 - Debian File System
- Tool-Chain
 - Compile tool



Cross Compilation

- Native Environment
 - Host x86PC (Linux)
 - gcc
- Cross-Compile
 - aarch64-linux-gnu-gcc

Cross Compilation



RK3399 and SOC



RockPi WiKi

➤ Rock Pi4 Wiki

➤ <https://wiki.radxa.com/Rockpi4>

➤ Rock Pi 4 Future

➤ https://wiki.radxa.com/Rockpi4/getting_started

➤ Rock Pi 4 Debin

➤ <https://wiki.radxa.com/Rockpi4/Debian>

➤ <https://wiki.radxa.com/Rock4/downloads>

CPU

CPU (1)

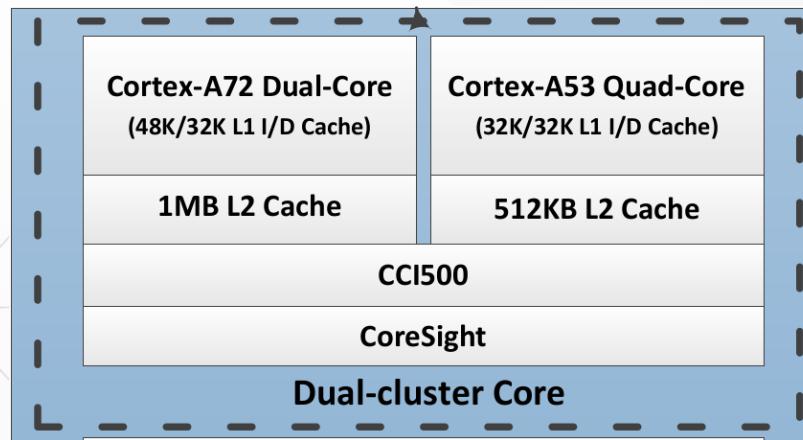
➤ Two CPU clusters

➤ Big cluster with dual-core Cortex-A72

- high-performance

➤ Little cluster with quad-core Cortex-A53

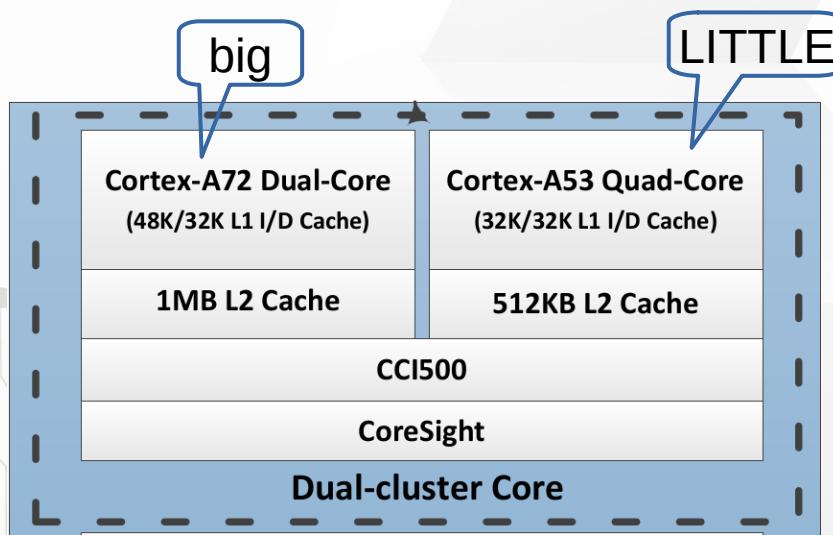
- low power



CPU (2)

► Arm big.LITTLE technology

- "LITTLE" processors are designed for maximum power efficiency
- "big" processors are designed to provide maximum compute performance.



Memory



Memory

▶ Internal ROM

- ▶ Internal BootRom (Size : 32KB)
- ▶ boot from
 - SPI, eMMC, SD/MMC

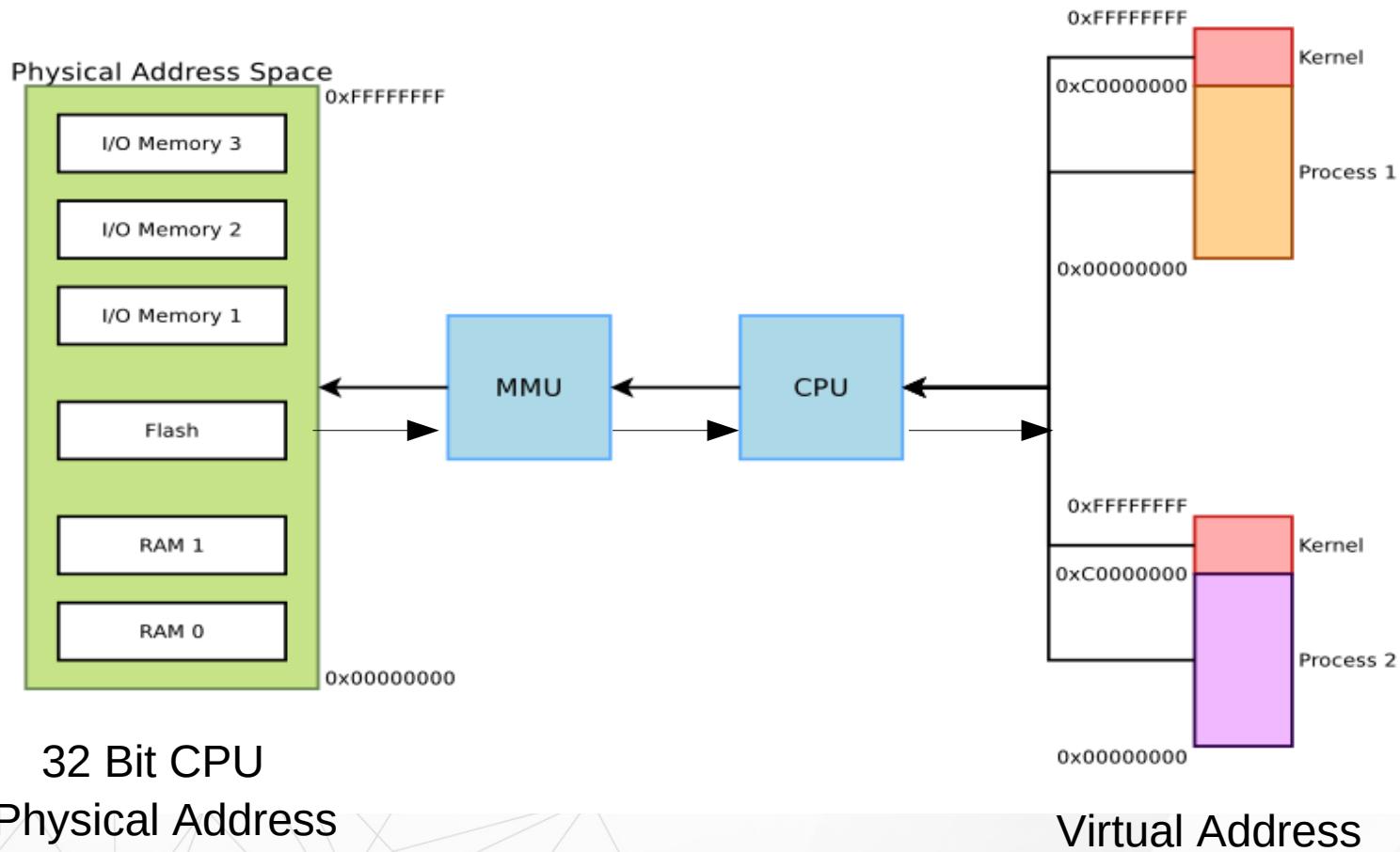
▶ Internal RAM

- ▶ 200KB

▶ External

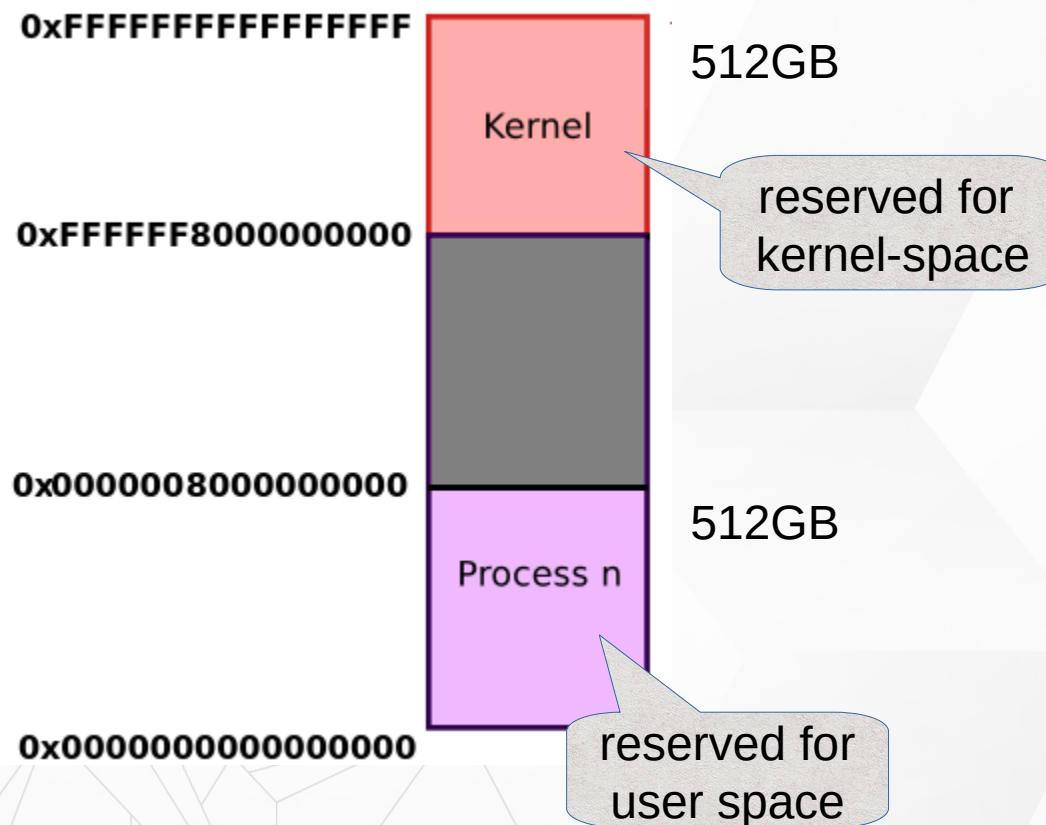
- ▶ DDR3/DDR3L/LPDDR3/LPDDR4
- ▶ SPI NOR/NAND Flash
- ▶ EMMC5.1
- ▶ SD3.0/MMC4.51

MMU (1)



MMU (2)

64 Bit CPU AArch64 Linux memory layout with 4KB pages + 3 levels

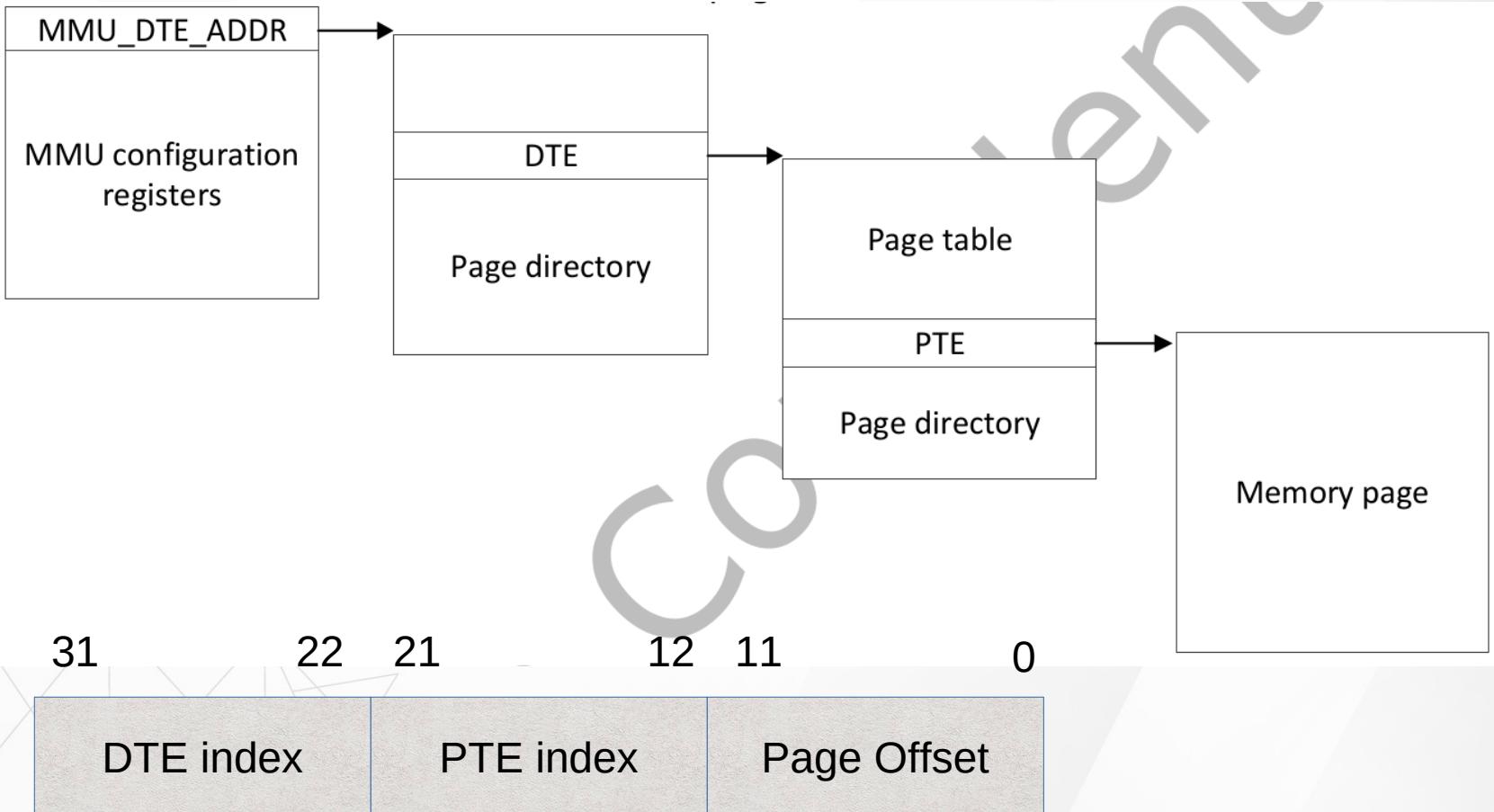




MMU (3)

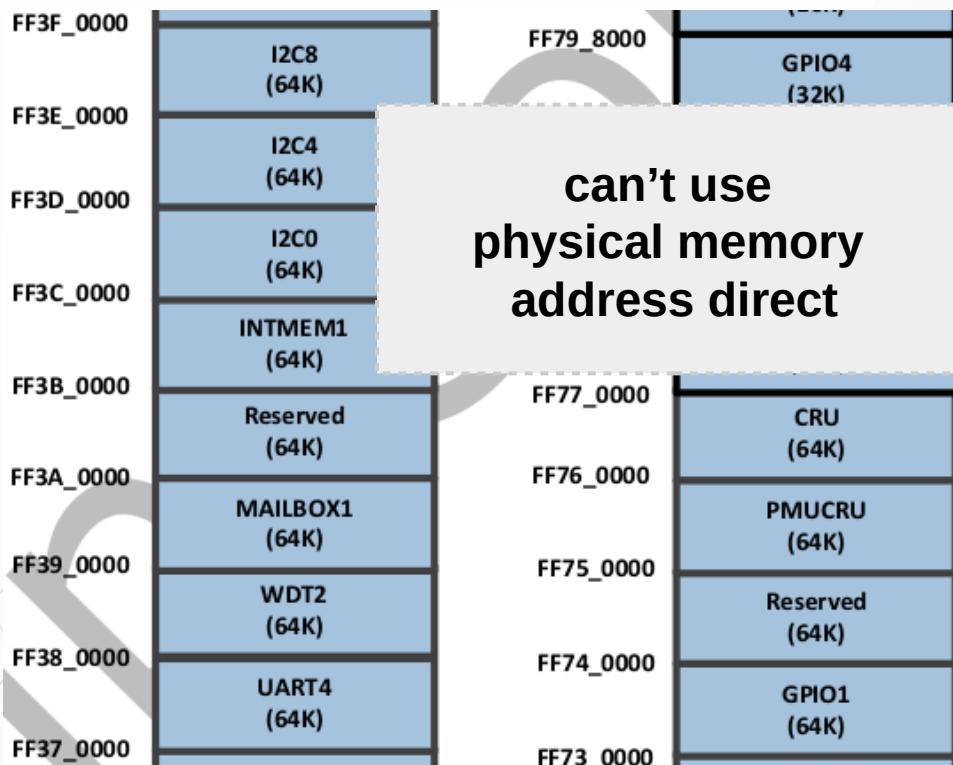
- The MMU divides memory into **4KB** pages
- 2-level page table structure
 - The First level
 - Page Directory consists of **1024** Directory Table Entries (DTEs)
 - Each pointing to a Page Table.
 - The Second level
 - The Page Table consists of **1024** Page Table Entries (PTEs)
 - Each pointing to a page in memory

MMU (4)

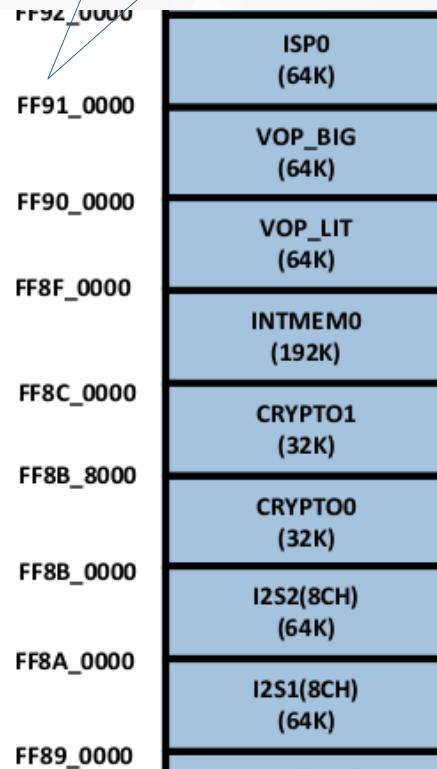


MMU (5)

I/O Address Mapping



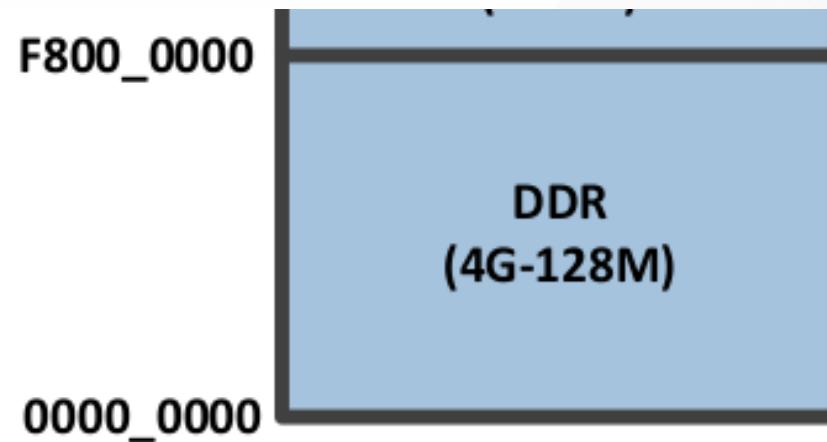
Physical memory address





MMU (6)

DDR SD RAM Memory



Interconnect Connect



Bus Architecture

➤ AMBA : Advanced Microcontroller Bus Architecture

➤ AXI

➤ AHB

➤ APB



GPU



Graphics Engine (1)

- » Graphics Process Unit
- » Mali-T860MP4 GPU
 - » OpenGL ES1.1/2.0/3.0, OpenCL1.2,
 - » 3D Graphics Engine
 - » 2D Graphics Engine



Graphics Engine (2)

▶ OpenGL

- ▶ Open Graphics Library

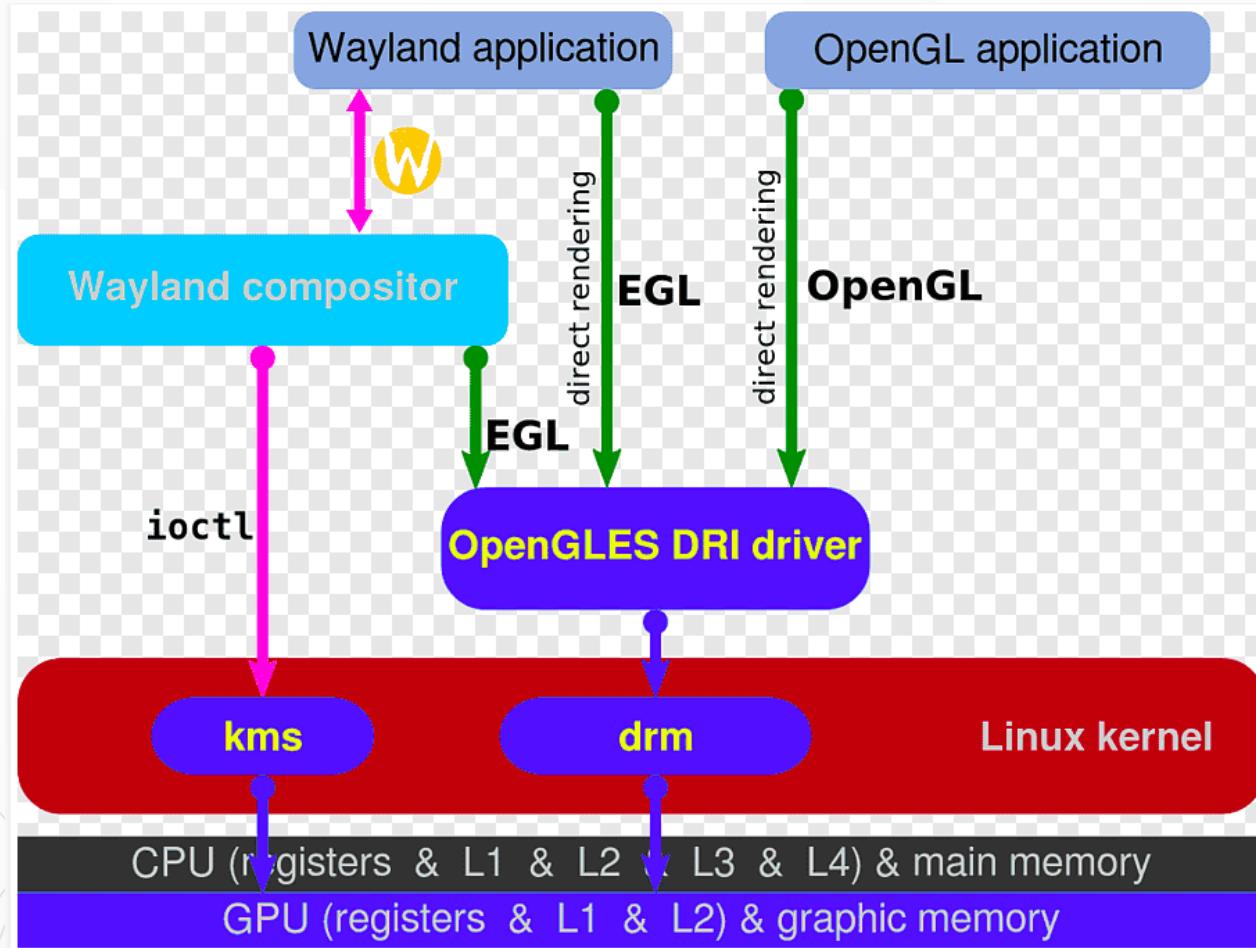
▶ OpenGL ES

- ▶ OpenGL for Embedded Systems

▶ EGL

- ▶ Native Platform Graphics Interface

Graphics Engine (3)



Connect



USB (1)

▶ USB Host

- ▶ RK3399
- ▶ OHCI : 1.1. Hardware Complex
- ▶ UHCI : 1.0, 1.1 Software Complex
- ▶ EHCI : 2.0
- ▶ XHCI : 3.0

▶ USB Device

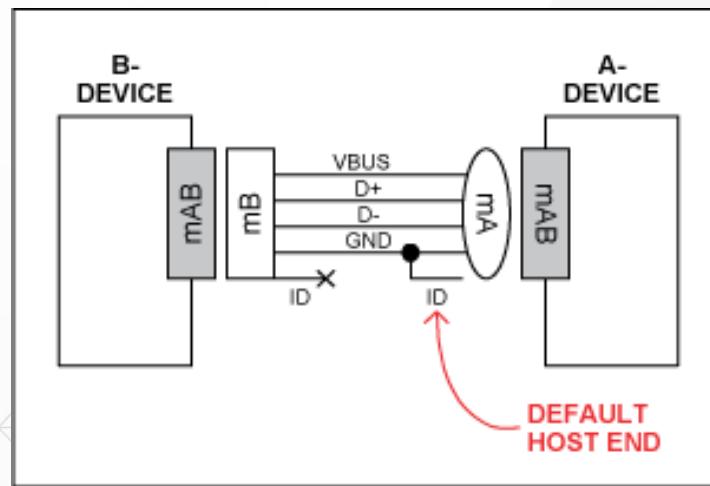
▶ USB Storage

USB (2)

▶ USB OTG

▶ USB_ID 信號為低時，該設備應作為 Host

▶ USB_ID 信號為高時，該設備作為 Slave



UART

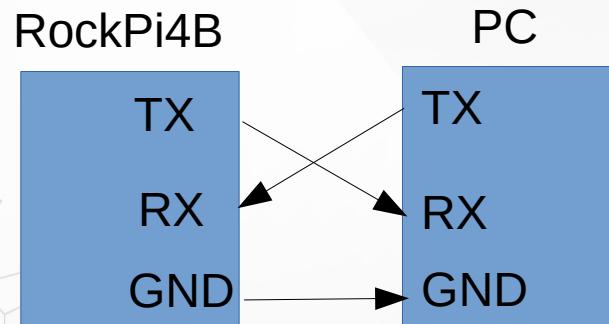
➤ The Universal Asynchronous Receiver/Transmitter

➤ Write Data

- CPU → Data → APB → UART

➤ Read Data

- Data → UART → APB → CPU

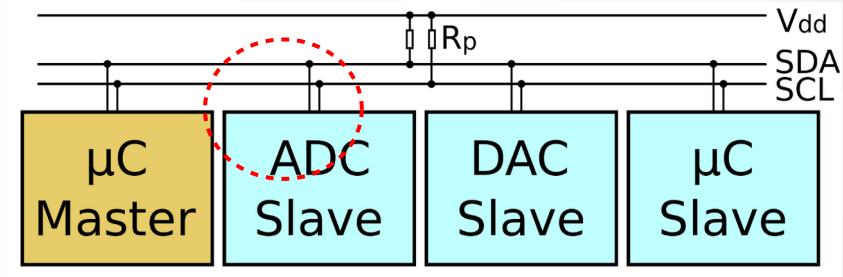


I2C (1)

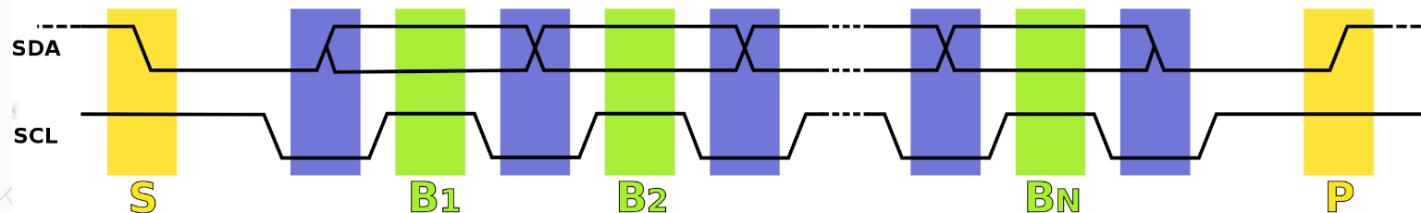
► Serial bus

► SDA data line

► SCL clock line

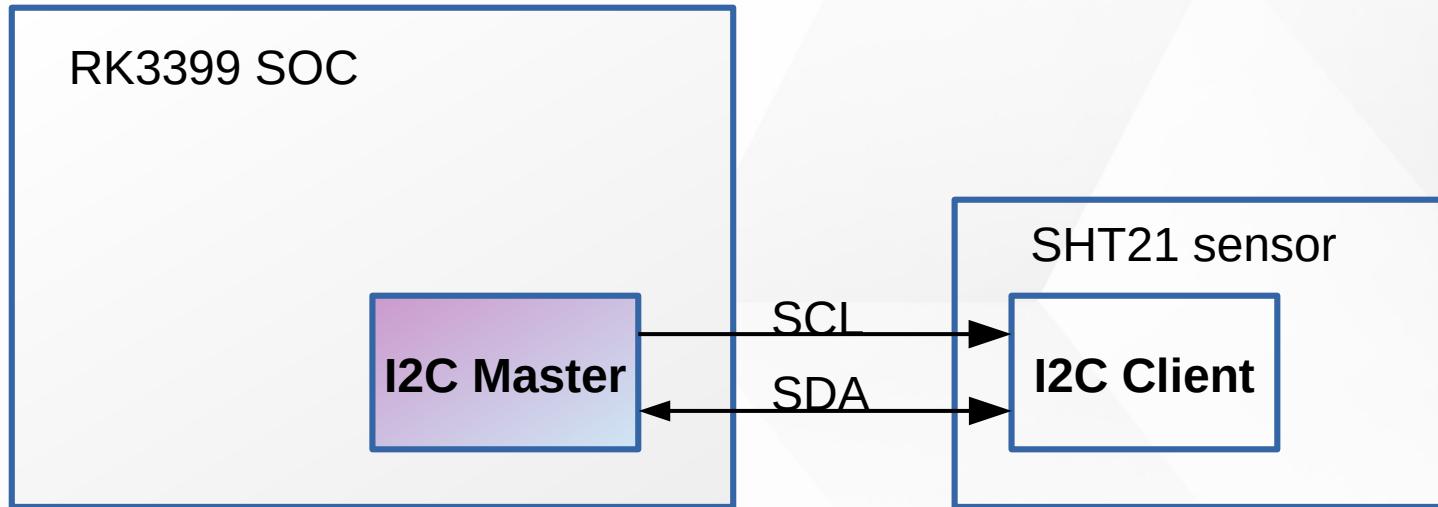


Protocol



I2C (2)

Master and Client



I2C protocol - Write

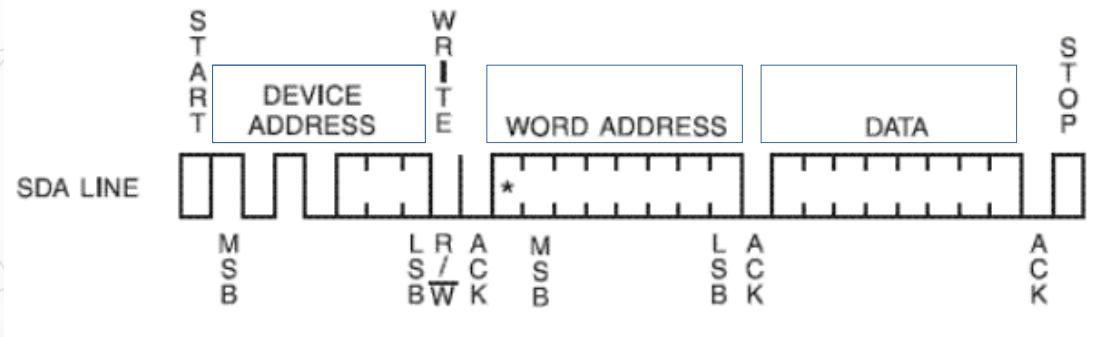
➤ Write

- byte write
- page write

➤ Device address

➤ Read/write bit : 0

➤ ACK





I2C protocol - Read

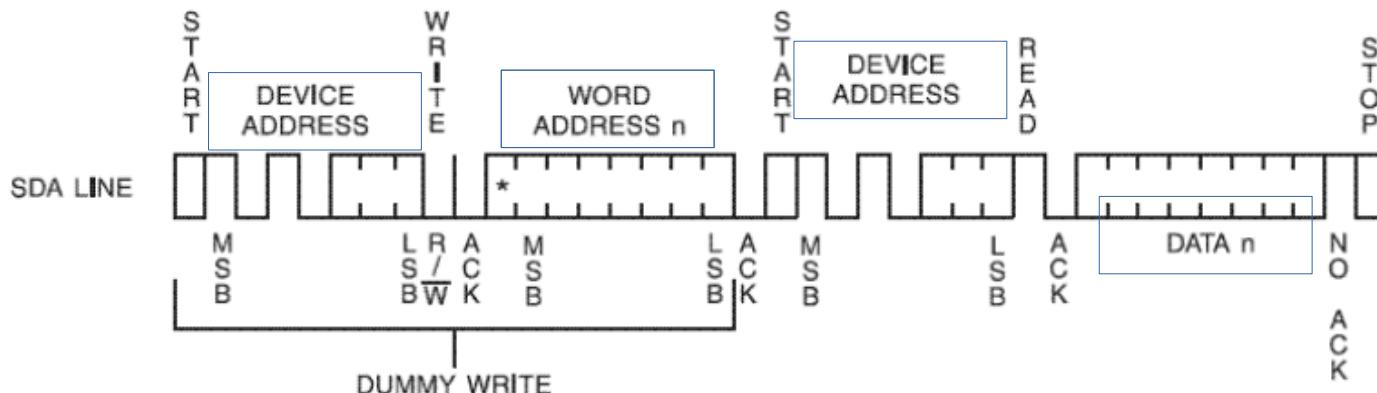
► Read

- byte read
- page read

► Device address

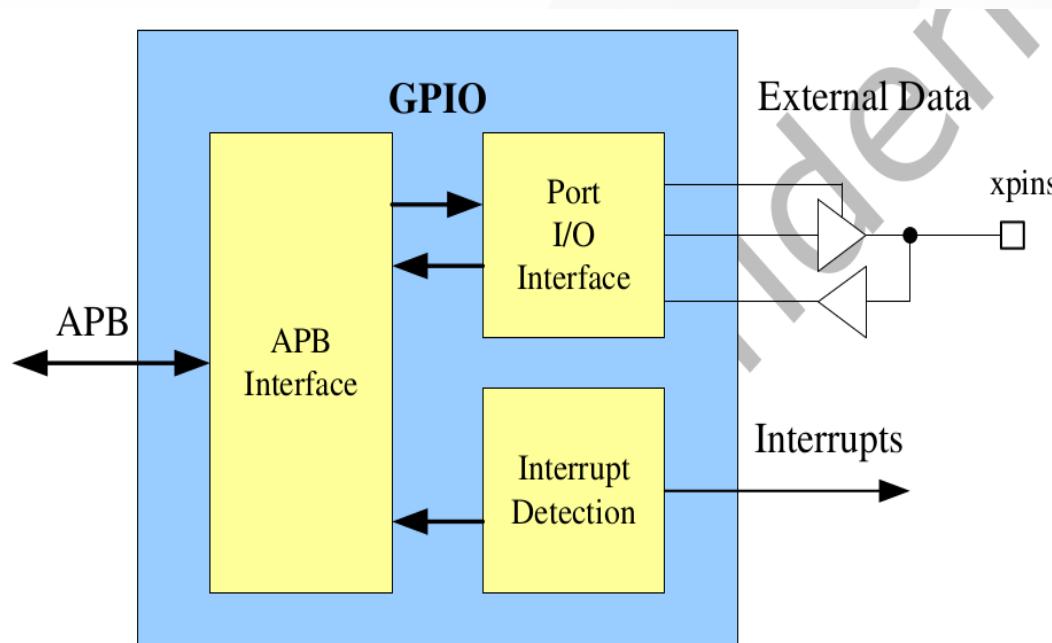
► Read/write bit : 1

► ACK



GPIO

- » General Purpose Programming I/O
- » GPIO controls the output data and direction of external I/O pads

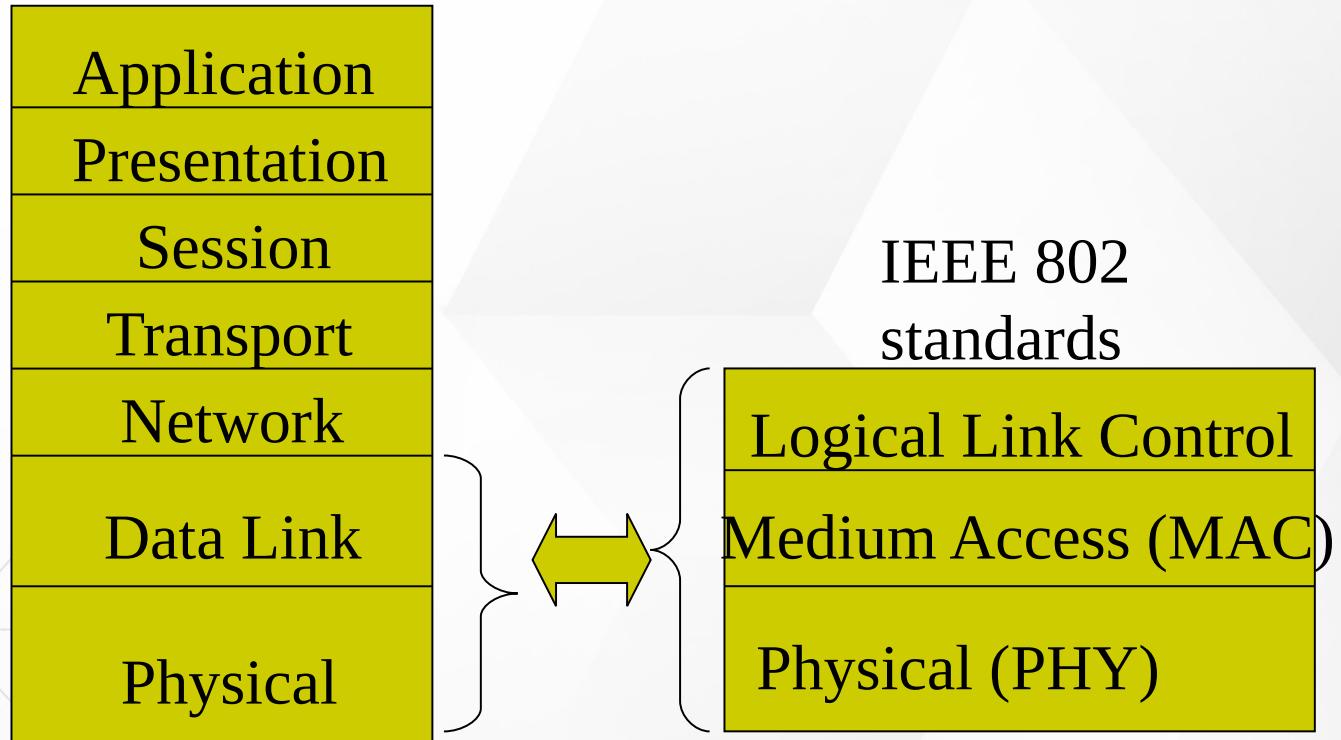


WiFi Basic

WIFI – 802.11 (1)

► IEEE 802.11 – 無線區域網路

ISO
OSI
7-layer
model

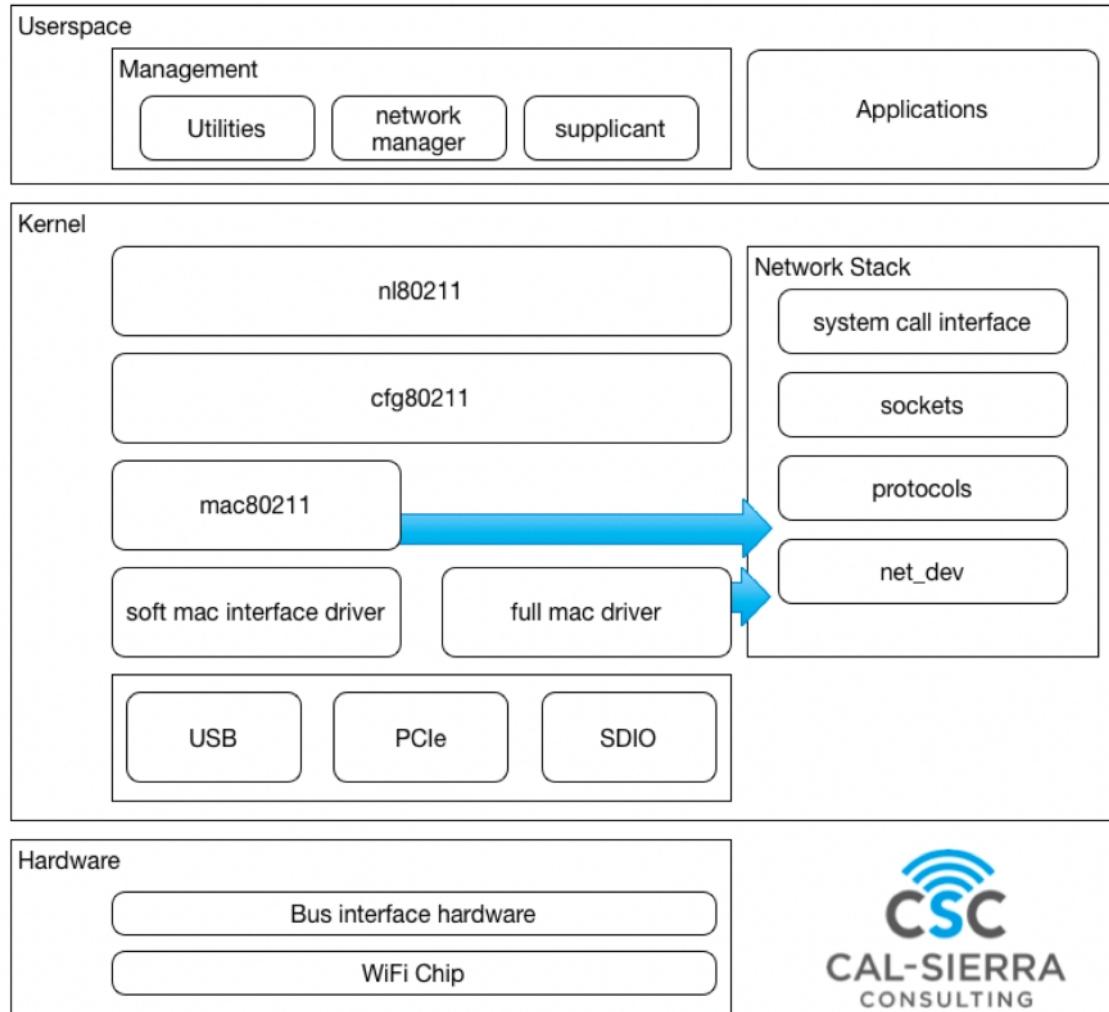


WIFI – 802.11 (2)

世代名稱 ^[註 1]	IEEE標準		最大速率 (Mbit/s)	頻率 (GHz)
	名稱	發布年份		
Wi-Fi 7	802.11be	(2024) ^[註 2]	1376～46120	2.4/5/6
Wi-Fi 6E	802.11ax	2020	574～9608 ^[1]	6 ^[2]
Wi-Fi 6			2019	2.4/5
Wi-Fi 5	802.11ac	2014	433～6933	5 ^[3]
Wi-Fi 4	802.11n	2008	72～600	2.4/5
Wi-Fi 3 ^[註 3]	802.11g	2003	6～54	2.4
Wi-Fi 2 ^[註 3]	802.11a	1999	6～54	5
Wi-Fi 1 ^[註 3]	802.11b	1999	1～11	2.4
Wi-Fi 0 ^[註 3]	802.11	1997	1～2	2.4

1. ^ Wi-Fi是Wi-Fi联盟的商標
2. ^ 預定的發布年份
3. ^ 3.0 3.1 3.2 3.3 Wi-Fi联盟未定義 Wi-Fi 0/1/2/3 的世代名稱^{[4][5]}

WIFI – Linux & 802.11 (1)





WIFI – Linux & 802.11 (2)

- ▶ mac80211
 - ▶ most associated to hardware offloading
 - ▶ the **802.11 protocol state machine lives here**
- ▶ cfg80211
 - ▶ middle-layer **Handles Everything Configurable**
- ▶ nl80211
 - ▶ The **API between user-land and kernel-land**
 - ▶ Relies on the **netlink** protocol to exchange messages between the two worlds



WIFI – Tool

- » Show / manipulate wireless devices and their configuration
 - » iw
- » For connecting to a WPA/WPA2 network
 - » wpa_supplicant

Audio



ALSA Overview

► Advanced Linux Sound Architecture

- Linux kernel
- Software framework

► Sound Servers

- PulseAudio, JACK ...

► ALSA stream is a data flow representing sound

- PCM (Pulse-code modulation)

[https://en.wikipedia.org/wiki/
Advanced_Linux_Sound_Architecture](https://en.wikipedia.org/wiki/Advanced_Linux_Sound_Architecture)



ALSA Overview

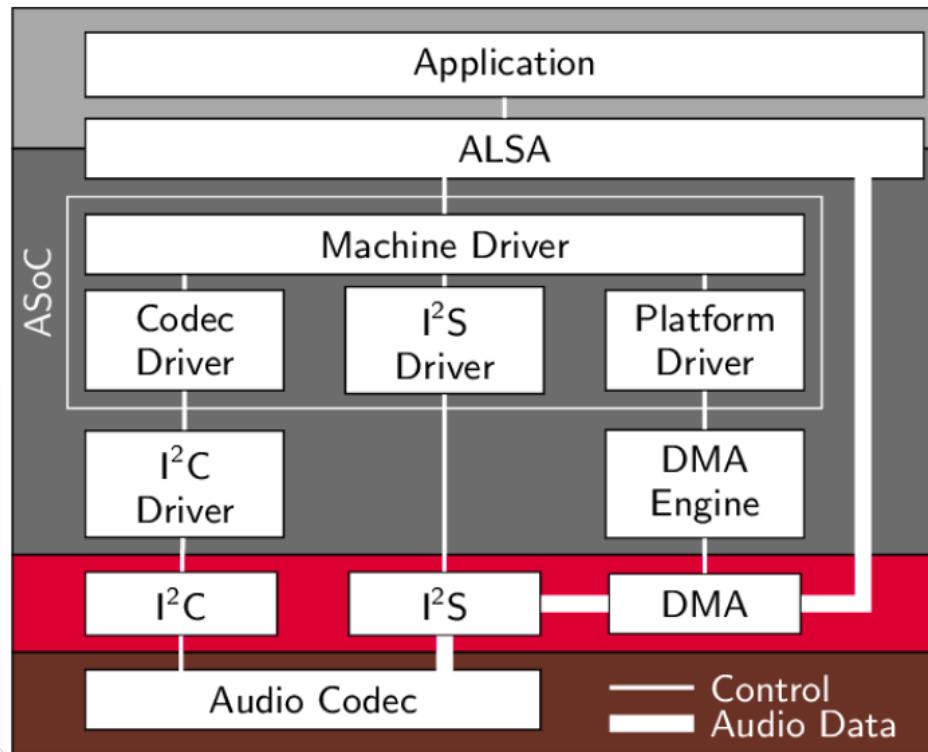
▶ Audio Codec

▶ Audio Codec 就是音樂訊號（Audio）壓縮 / 解壓縮 (Compress/DECompress) 的演算法或程式，前後加起來就是 Audio Codec.

▶ Parameters of the hardware

- ▶ sampling rate : 44100 Hz
- ▶ sample width : 8 bit, 16 bit, 24 bit
- ▶ sample encoding : endianness
- ▶ number of channels : 1 channel, 2 channel ...

ALSA Overview



https://www.researchgate.net/figure/Structure-of-ASoC-and-the-embedding-into-the-Linux-audio-framework_fig2_262112720



Sound Card in Linux (1)

[CMD] ls /proc/asound/ -l

```
lrwxrwxrwx    1 root root 5 Apr 30 06:51 HDMICODEC -> card1
dr-xr-xr-x    4 root root 0 Apr 30 06:51 card0
dr-xr-xr-x    3 root root 0 Apr 30 06:51 card1
-r--r--r--    1 root root 0 Apr 30 06:51 cards
-r--r--r--    1 root root 0 Apr 30 06:51 devices
-r--r--r--    1 root root 0 Apr 30 06:51 hwdep
-r--r--r--    1 root root 0 Apr 30 06:51 pcm
lrwxrwxrwx    1 root root 5 Apr 30 06:51 rockchipes8316c -> card0
dr-xr-xr-x 2 root root 0 Apr 30 06:51 seq
-r--r--r--    1 root root 0 Apr 30 06:51 timers
-r--r--r--    1 root root 0 Apr 30 06:51 version
```



Sound Card in Linux (2)

[CMD] cat /proc/asound/cards

Sound card 0

```
0 [rockchipes8316c]:  rockchip_es8316 - rockchip,es8316-codec  
                      rockchip,es8316-codec
```

```
1 [HDMICODEC ]: HDMI-CODEC - HDMI-CODEC  
                  HDMI-CODEC
```

Sound card 1



Sound Card in Linux (3)

[CMD] ls -l /proc/asound/card0/

```
-r--r--r-- 1 root root 0 Apr 30 06:59 id  
dr-xr-xr-x 3 root root 0 Apr 30 06:59 pcm0c → capture  
dr-xr-xr-x 3 root root 0 Apr 30 06:59 pcm0p → playback
```

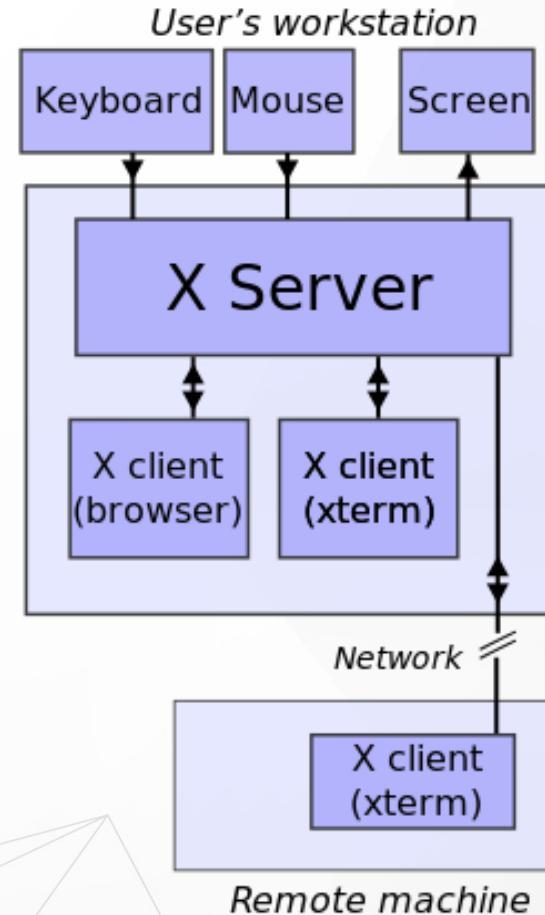
rockchipes8316c

[CMD] ls -l /proc/asound/card1/

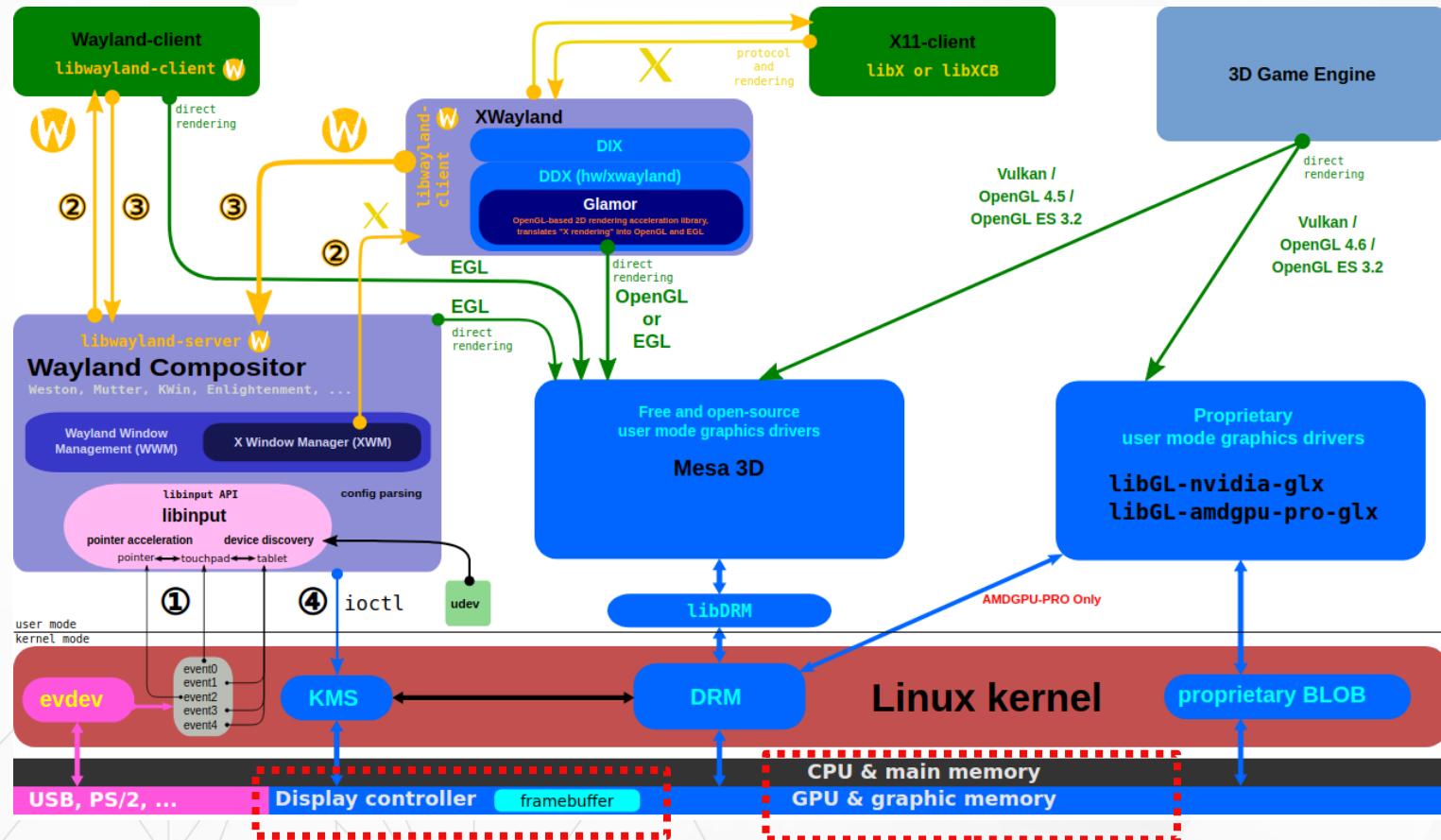
```
-r--r--r-- 1 root root 0 Apr 30 07:01 id  
dr-xr-xr-x 3 root root 0 Apr 30 07:01  
pcm0p
```

Linux Display Subsystem

Linux Windows System (1)

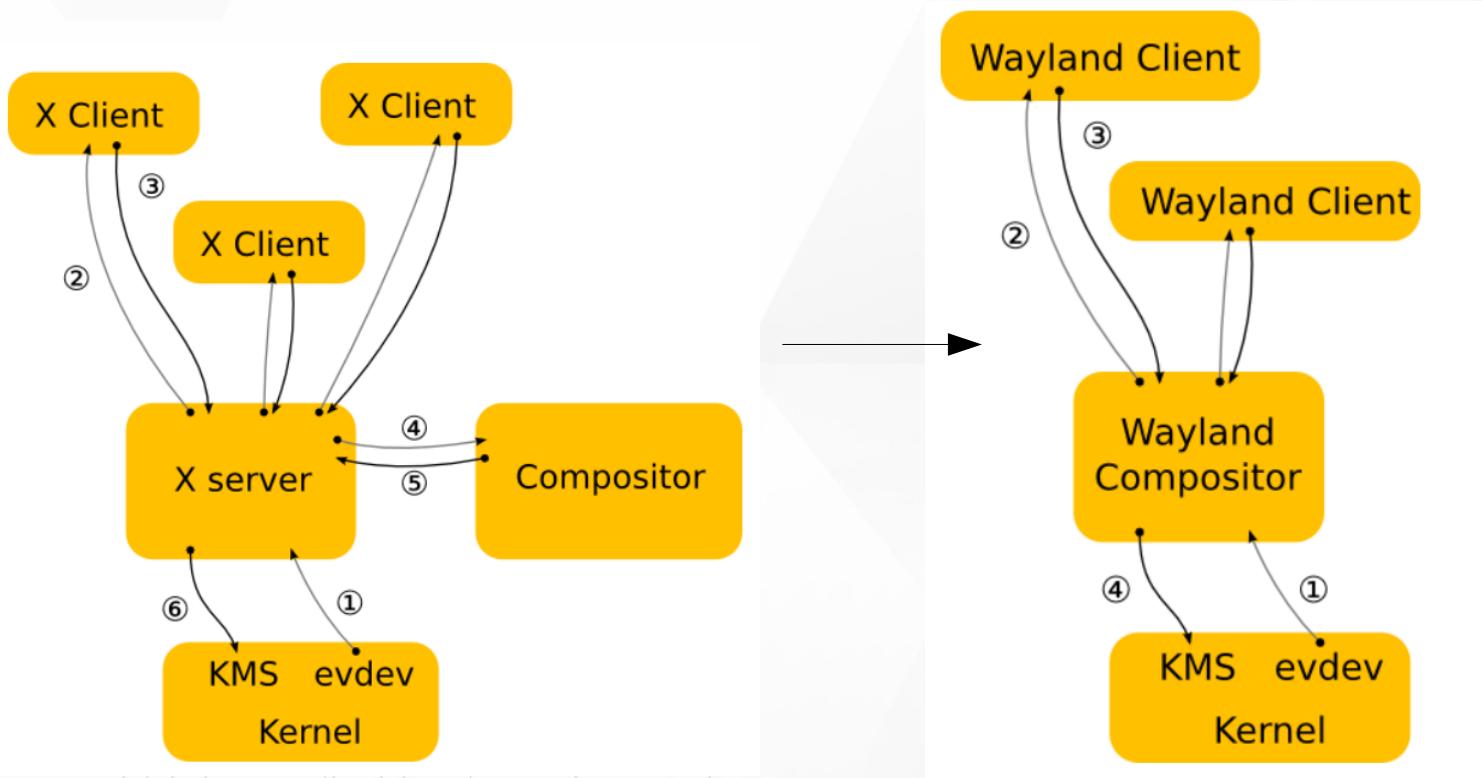


Linux Windows System (2)



Linux Windows System (3)

Wayland is a replacement for the X11 window system protocol





GTK and Gnome

- **GTK**

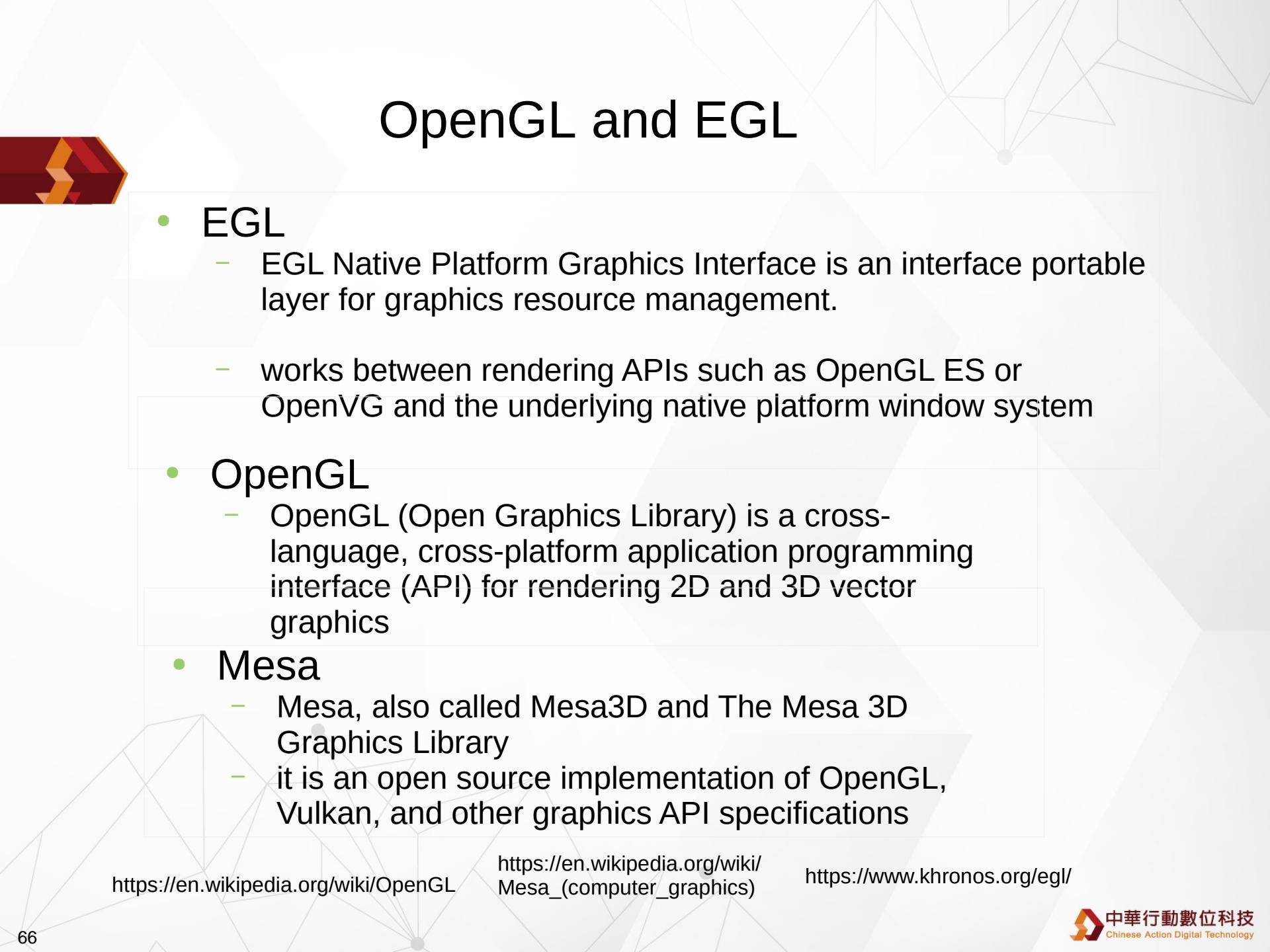
- GTK (formerly GTK+) is a free and open-source cross-platform widget toolkit for creating graphical user interfaces (GUIs).
 -

- **GNOME**

- GNOME is the default desktop environment of many major Linux distributions
 - originally an acronym for GNU Network Object Model Environment
 - free and open-source desktop environment for Linux and other Unix-like[10] operating systems

<https://en.wikipedia.org/wiki/GNOME>

<https://en.wikipedia.org/wiki/GTK>



OpenGL and EGL

- EGL
 - EGL Native Platform Graphics Interface is an interface portable layer for graphics resource management.
 - works between rendering APIs such as OpenGL ES or OpenVG and the underlying native platform window system
- OpenGL
 - OpenGL (Open Graphics Library) is a cross-language, cross-platform application programming interface (API) for rendering 2D and 3D vector graphics
- Mesa
 - Mesa, also called Mesa3D and The Mesa 3D Graphics Library
 - it is an open source implementation of OpenGL, Vulkan, and other graphics API specifications

<https://en.wikipedia.org/wiki/OpenGL>

[https://en.wikipedia.org/wiki/Mesa_\(computer_graphics\)](https://en.wikipedia.org/wiki/Mesa_(computer_graphics))

<https://www.khronos.org/egl/>

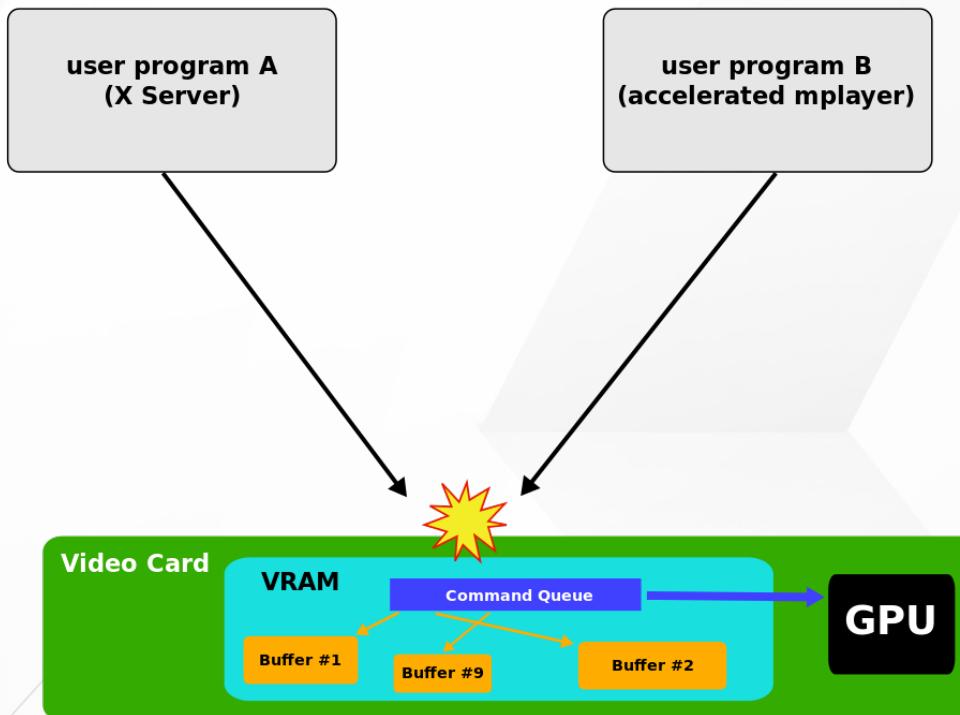


Direct Rendering Manager (DRM)

- Direct Rendering Manager
 - Management of buffers and free space within that memory.
 - Solve Frame buffer driver cannot be used GPU and multi-user process.
- DRM consists of
 - libdrm
 - libdrm provides a user space library for accessing the DRM
 - KMS : Kernel Mode Setting
 - Change resolution and depth
 - DRI : Direct Rendering Infrastructure
 - Interfaces to access hardware directly
 - GEM : Graphics Execution Manager
 - Buffer management
 - DRM Driver in kernel side
 - Access hardware

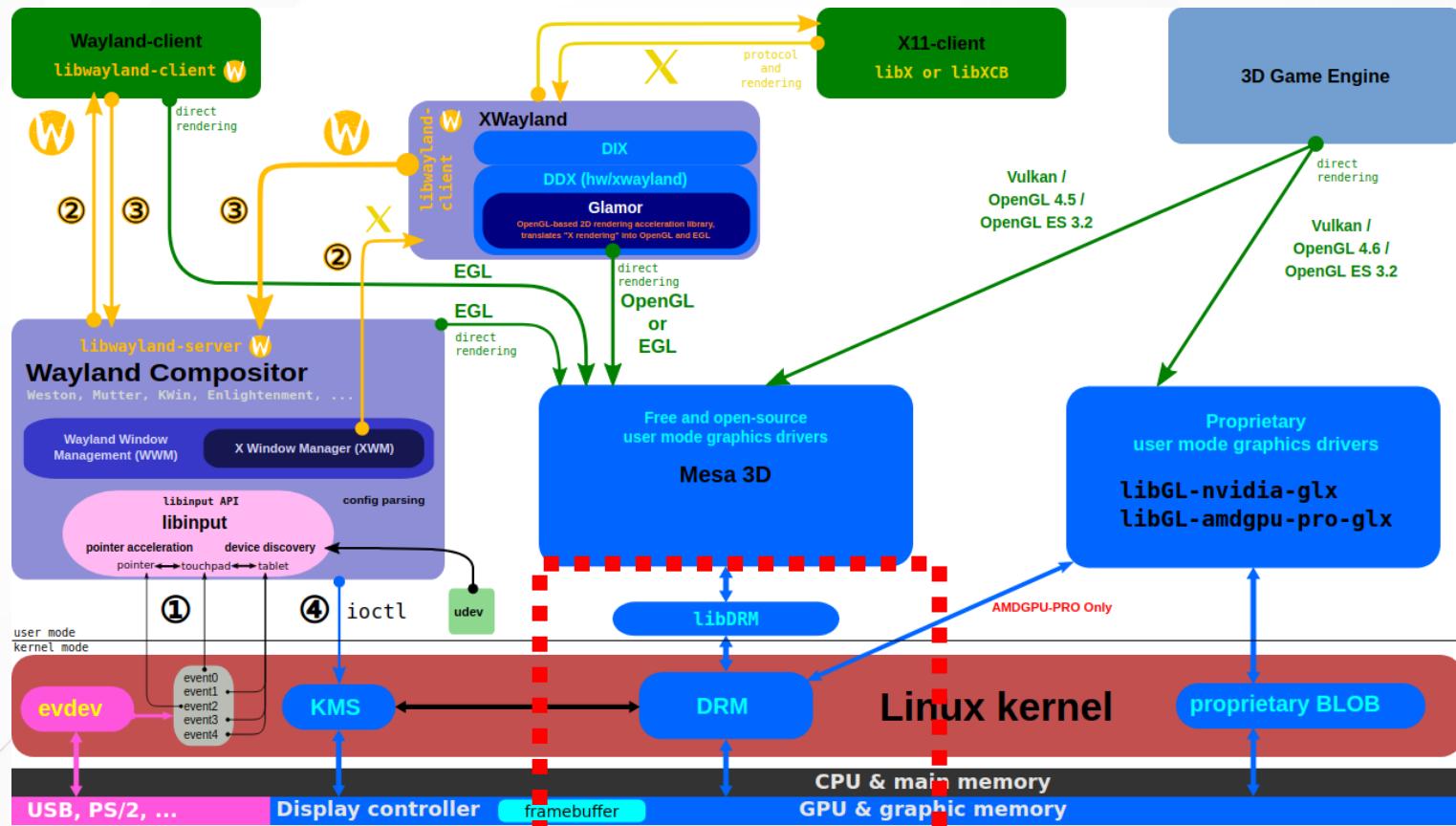
Direct Rendering Manager (DRM)

If no use DRM



© 2014 Javier Cantero - this work is under the Creative Commons Attribution ShareAlike 4.0 license

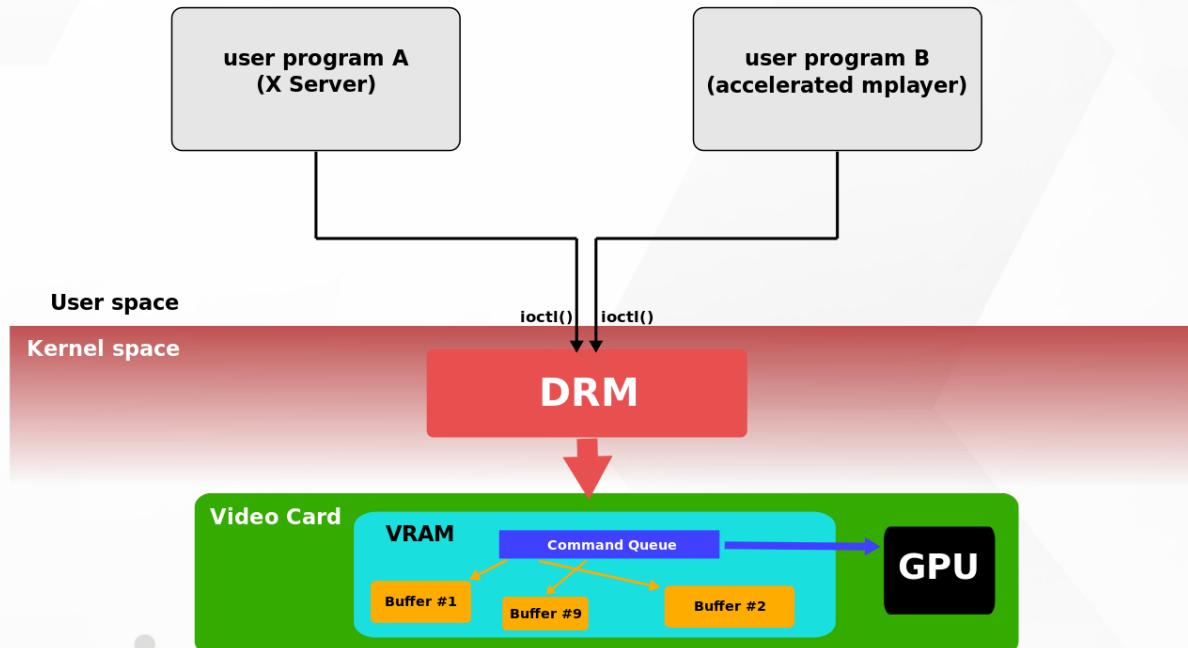
Direct Rendering Manager (DRM)



https://upload.wikimedia.org/wikipedia/commons/2/2d/The_Linux_Graphics_Stack_and_glamor.svg

Direct Rendering Manager (DRM)

Use DRM



© 2014 Javier Cantero - this work is under the Creative Commons Attribution ShareAlike 4.0 license



Kernel Mode Setting (KMS)

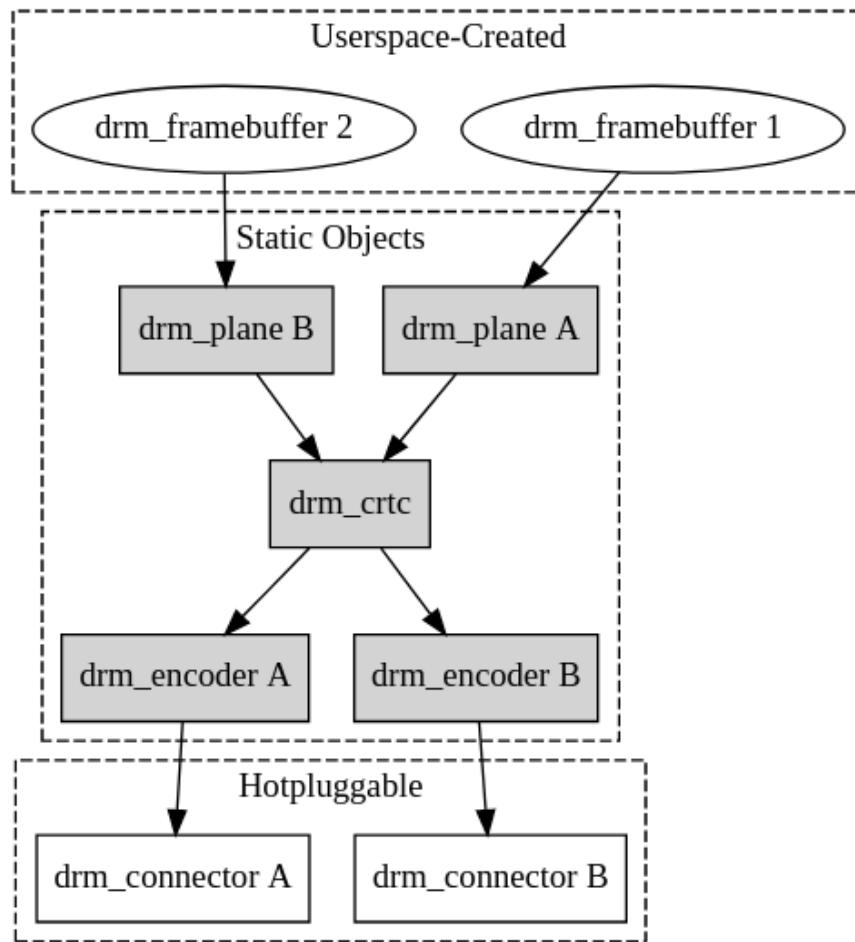
➤ KMS device model

- CRTCs
- Connectors
- Encoders
- Planes

➤ Kernel Mode Setting

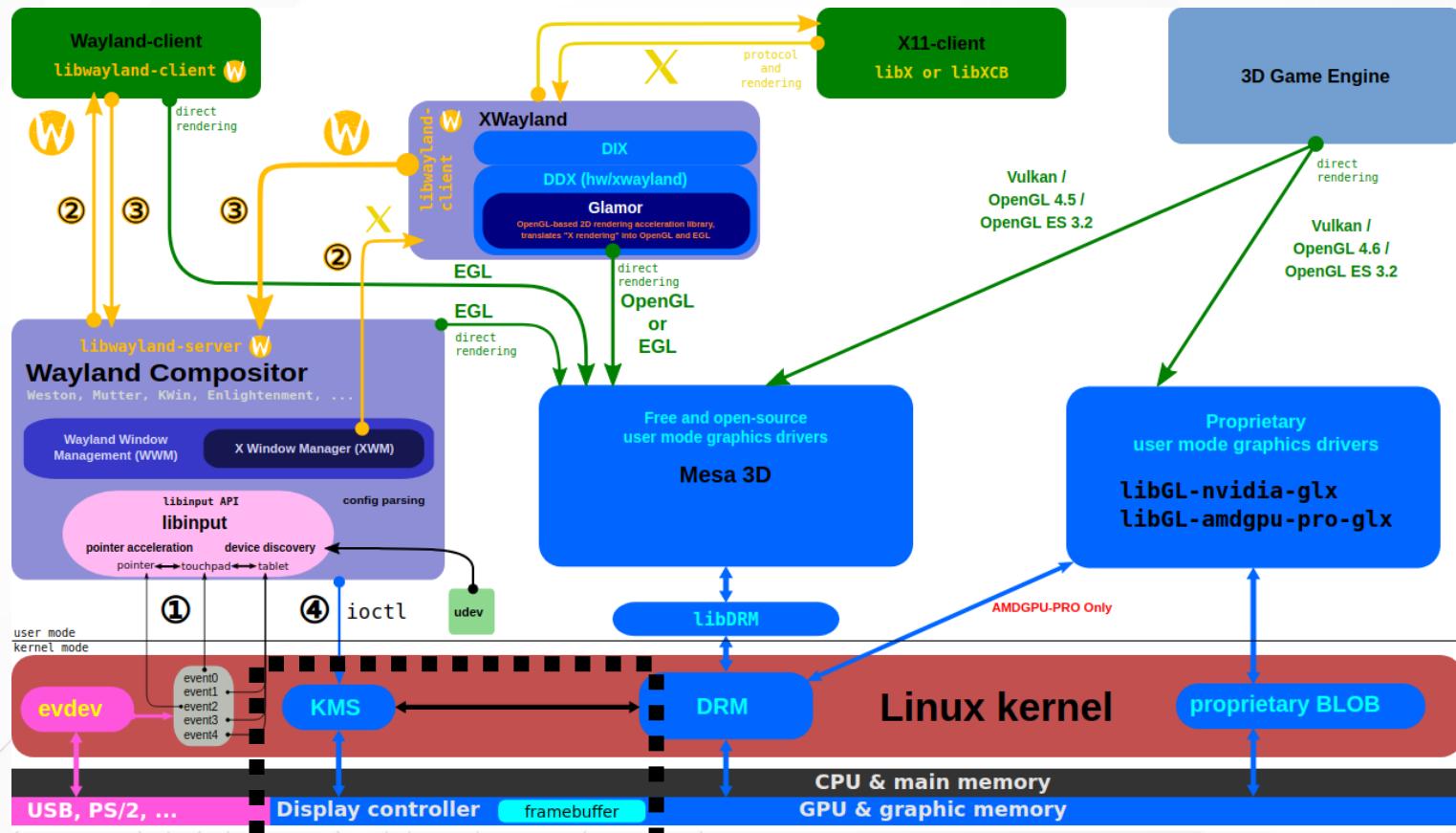
- screen resolution
- color depth and
- refresh rate

Kernel Mode Setting (KMS)



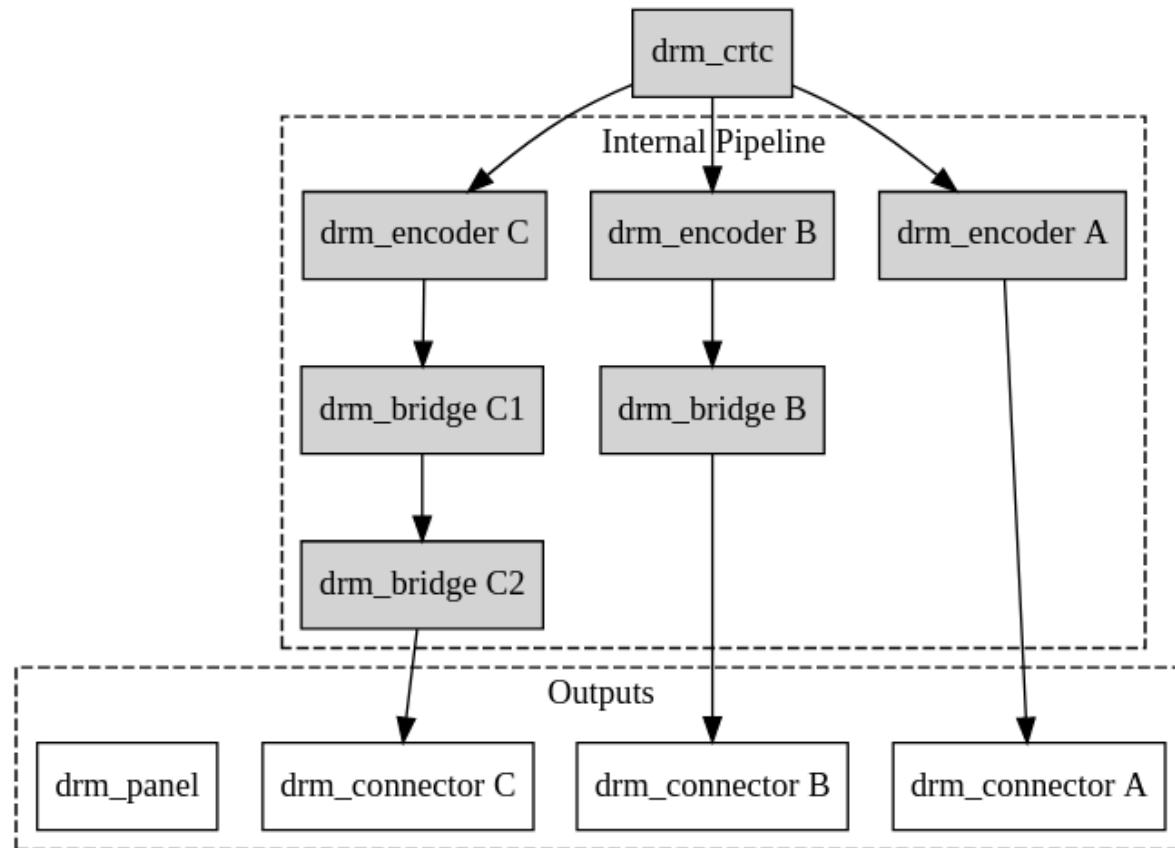
<https://www.kernel.org/doc/html/v4.15/gpu/drm-kms.html>

Kernel Mode Setting (KMS)



https://upload.wikimedia.org/wikipedia/commons/2/2d/The_Linux_Graphics_Stack_and_glamor.svg

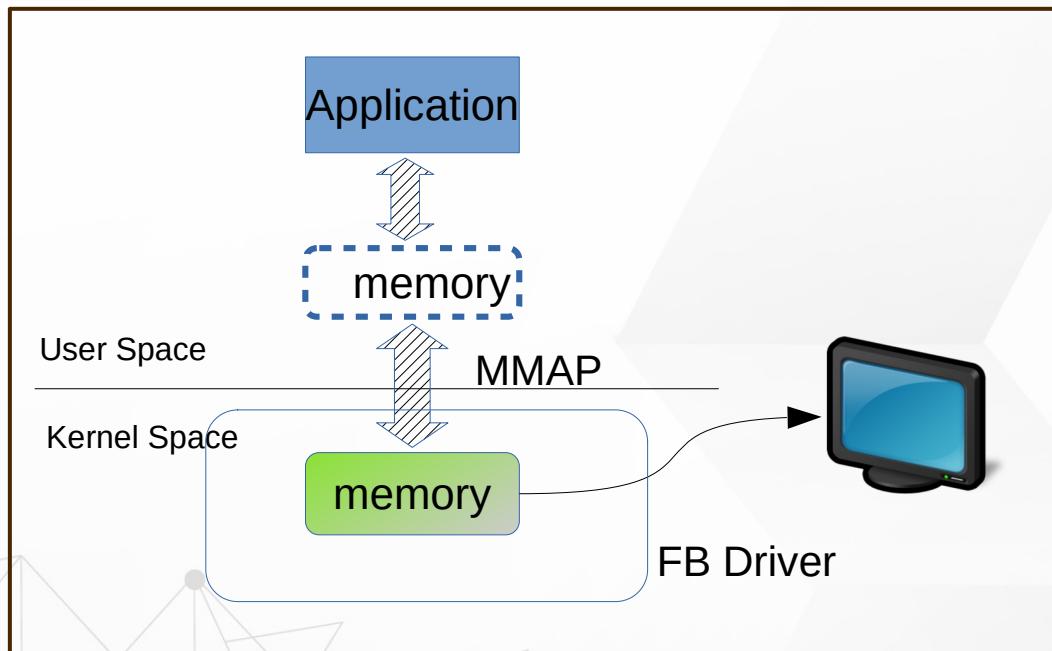
Kernel Mode Setting (KMS)



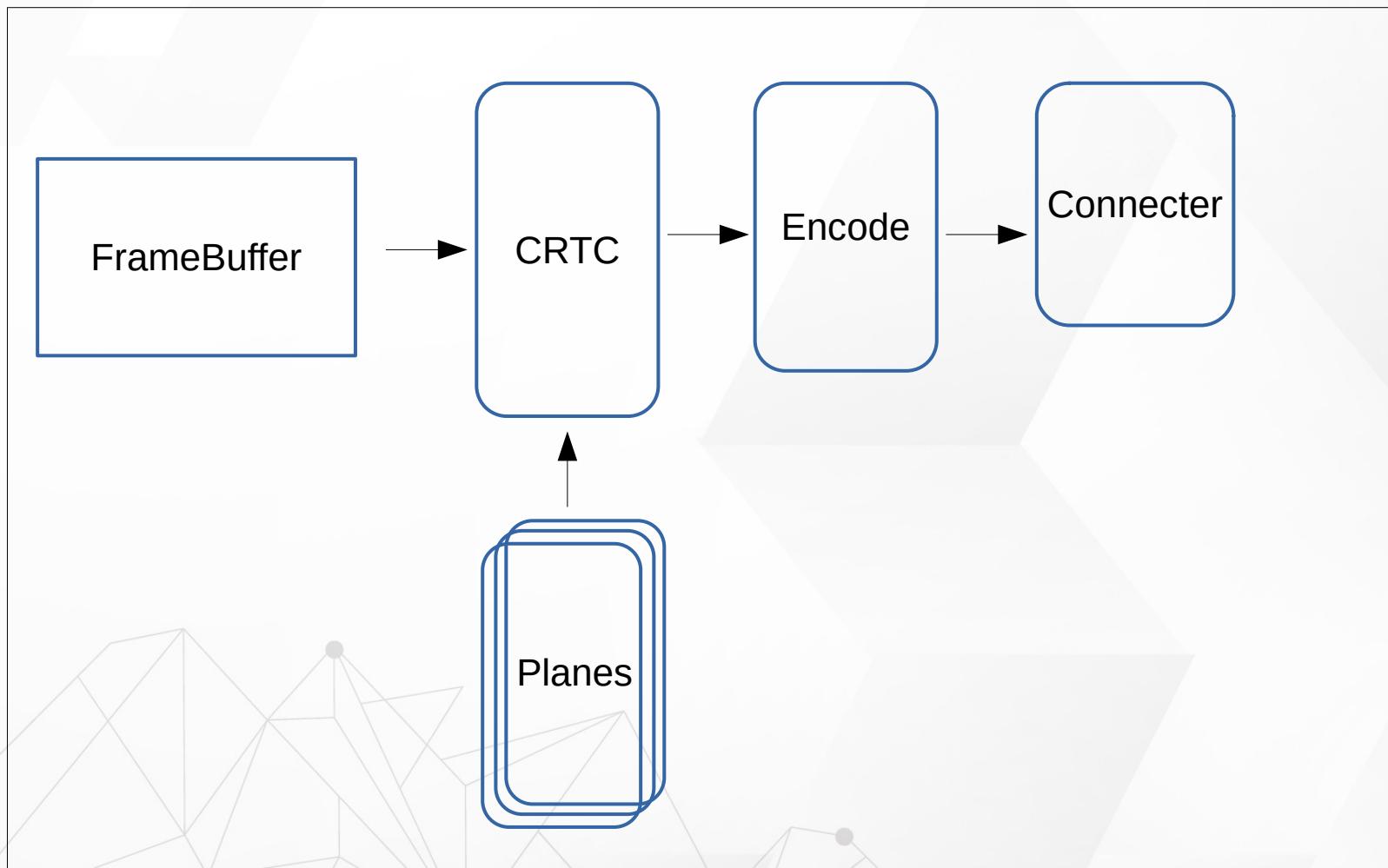
<https://www.kernel.org/doc/html/v4.15/gpu/drm-kms.html>

Video Frame Buffer

- ▶ The frame buffer device provides an abstraction for the graphics hardware.



Kernel Mode Setting (KMS)



Embedded Linux

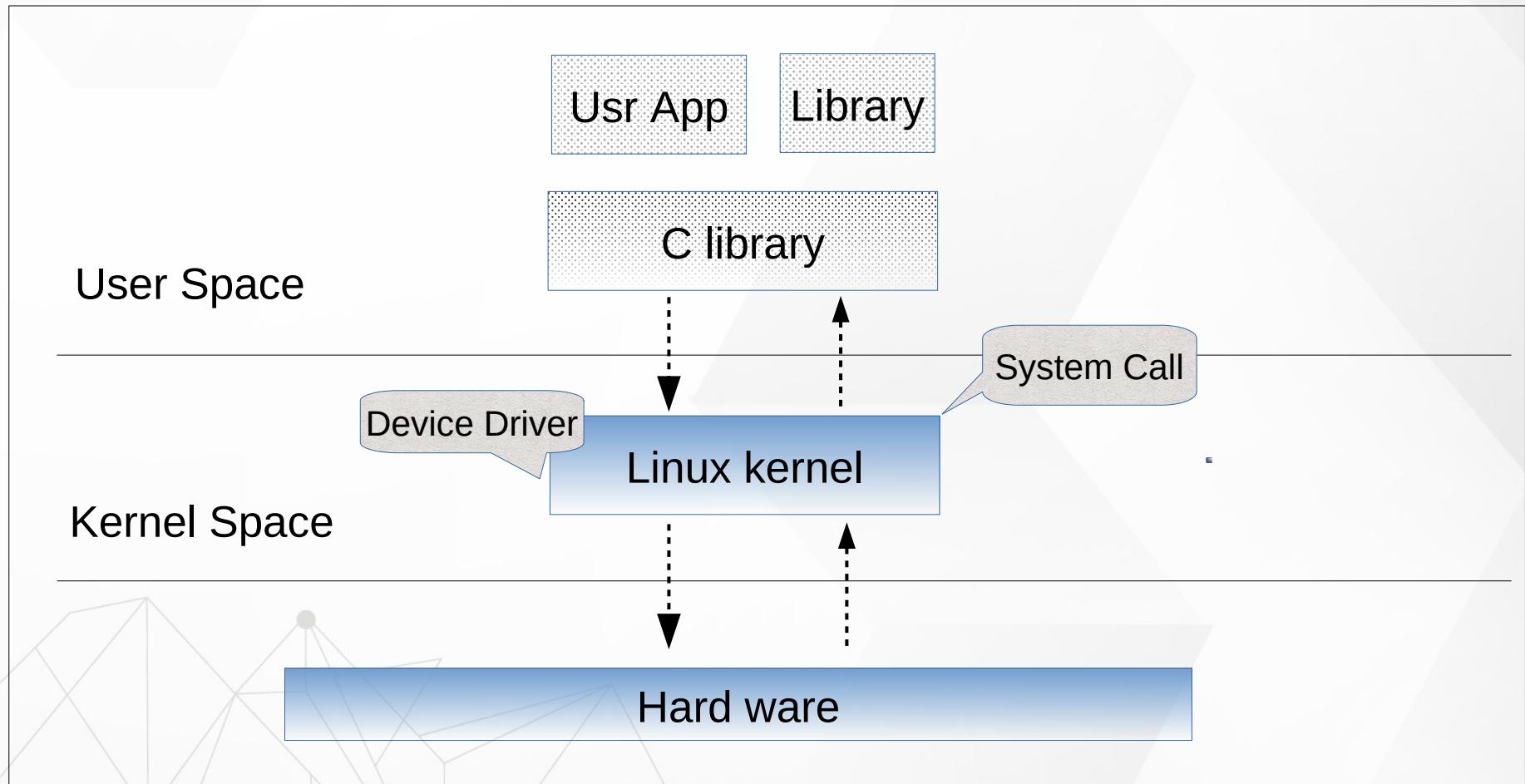
Various Layers within Linux

Various layers within Linux, also showing separation between the **userland** and **kernel space**

User mode		<p>User applications: <i>bash, LibreOffice, GIMP, Blender, 0 A.D., Mozilla Firefox, ...</i></p>					
User mode	System components	init daemon: <i>OpenRC, runit, systemd...</i>	System daemons: <i>polkitd, smbd, sshd, udevd...</i>	Window manager: <i>X11, Wayland, SurfaceFlinger (Android)</i>	Graphics: <i>Mesa, AMD Catalyst, ...</i>	Other libraries: <i>GTK, Qt, EFL, SDL, SFML, FLTK, GNUstep, ...</i>	
	C standard library	<p><i>fopen, execv, malloc, memcpy, localtime, pthread_create</i> ... (up to 2000 subroutines) <i>glibc</i> aims to be fast, <i>musl</i> aims to be lightweight, <i>uClibc</i> targets embedded systems, <i>bionic</i> was written for Android, etc. All aim to be POSIX/SUS-compatible.</p>					
Kernel mode	Linux kernel	<p><i>stat, splice, dup, read, open, ioctl, write, mmap, close, exit</i>, etc. (about 380 system calls) The Linux kernel System Call Interface (SCI), aims to be POSIX/SUS-compatible^[2]</p>					
Kernel mode		Process scheduling subsystem	IPC subsystem	Memory management subsystem	Virtual files subsystem	Network subsystem	
		<p>Other components: <i>ALSA, DRI, evdev, klibc, LVM, device mapper, Linux Network Scheduler, Netfilter</i> Linux Security Modules: <i>SELinux, TOMOYO, AppArmor, Smack</i></p>					
Hardware (CPU, main memory, data storage devices, etc.)							

https://en.wikipedia.org/wiki/User_space_and_kernel_space

Embedded Linux System



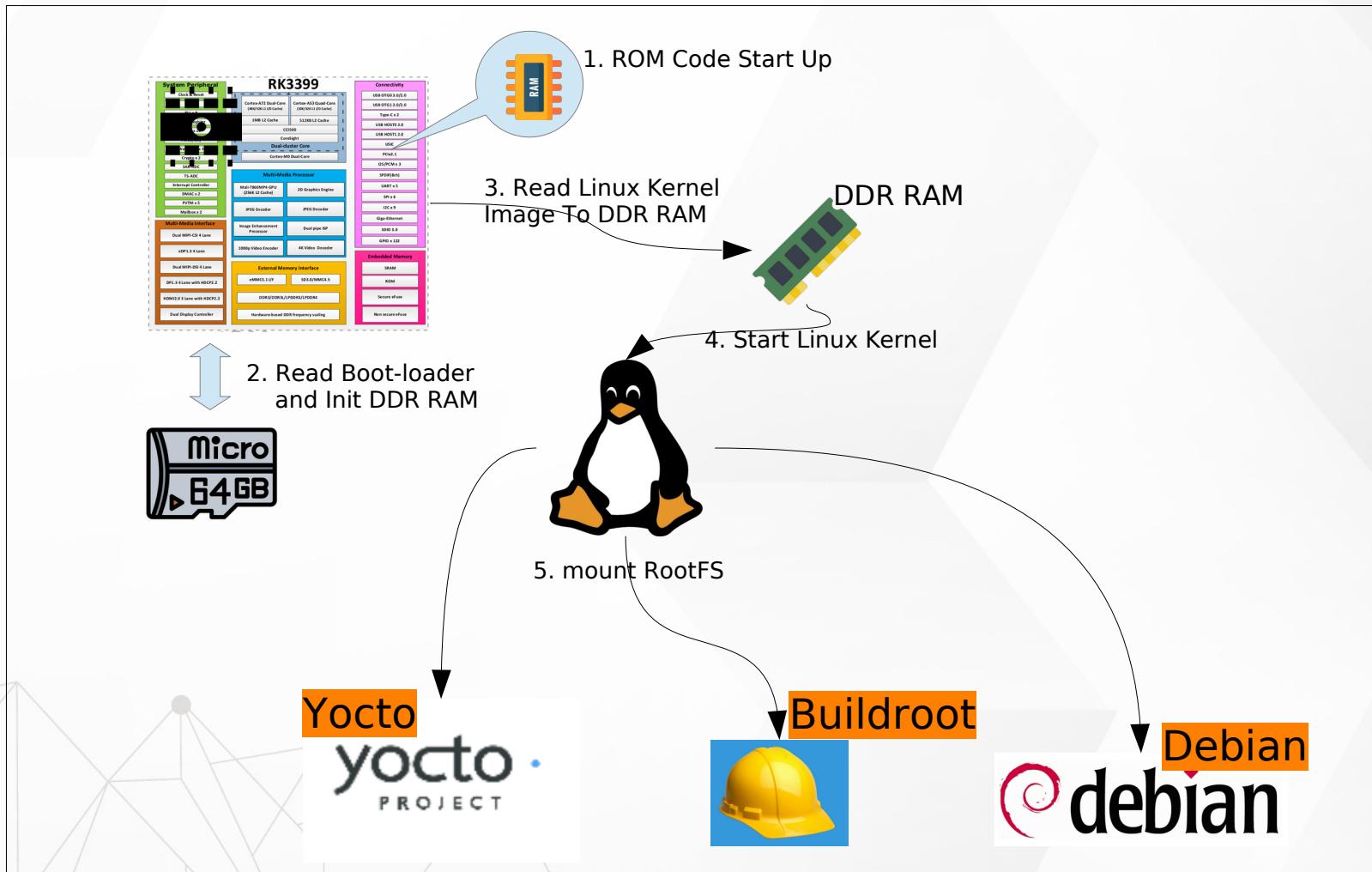


Linux kernel key features

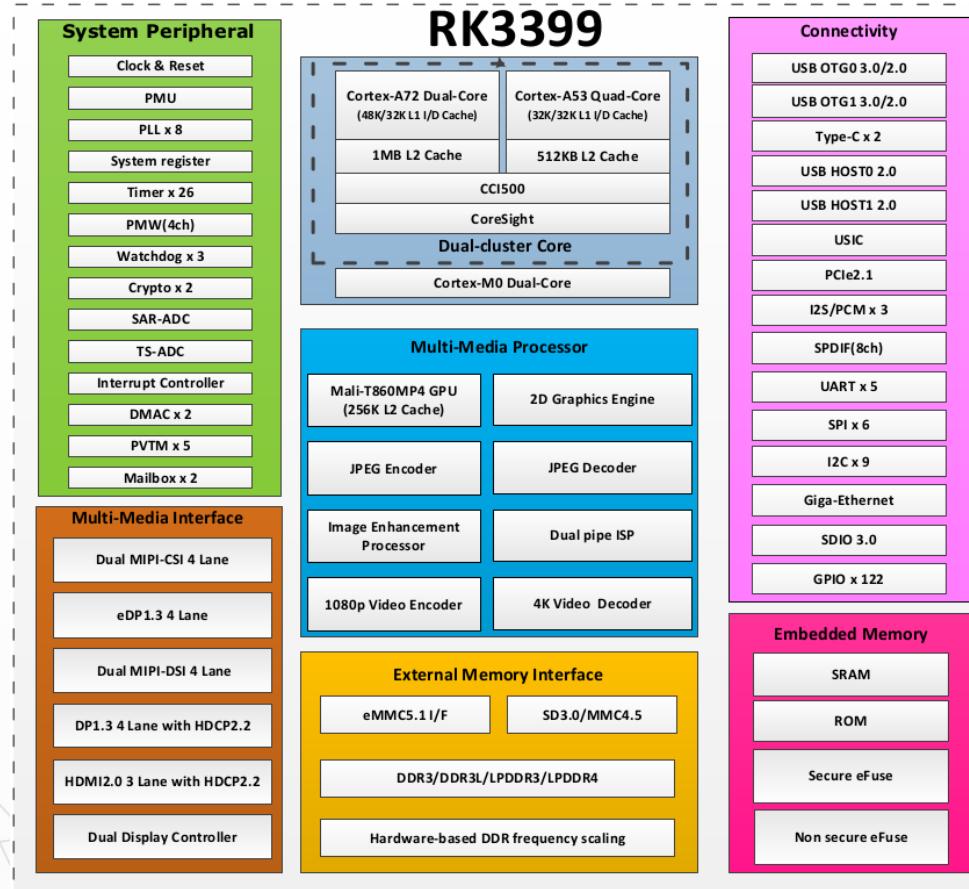
- ▶ Portability and hardware support
- ▶ Scalability
- ▶ Exhaustive networking support
- ▶ Stability and reliability
- ▶ Modularity
- ▶ Easy to program.

System Start Up

Linux Start Up

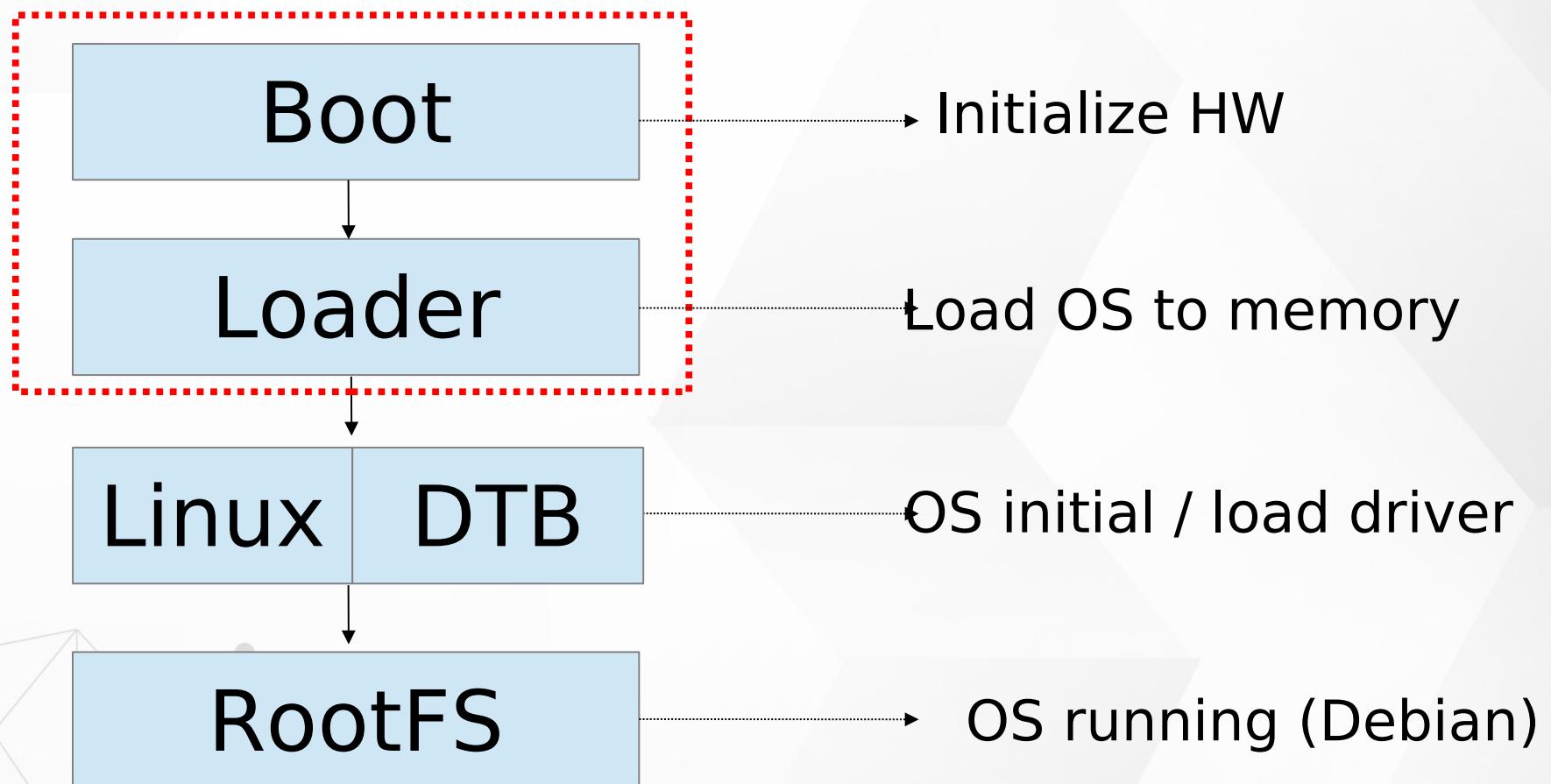


SOC RK3399





Embedded Linux System Booting

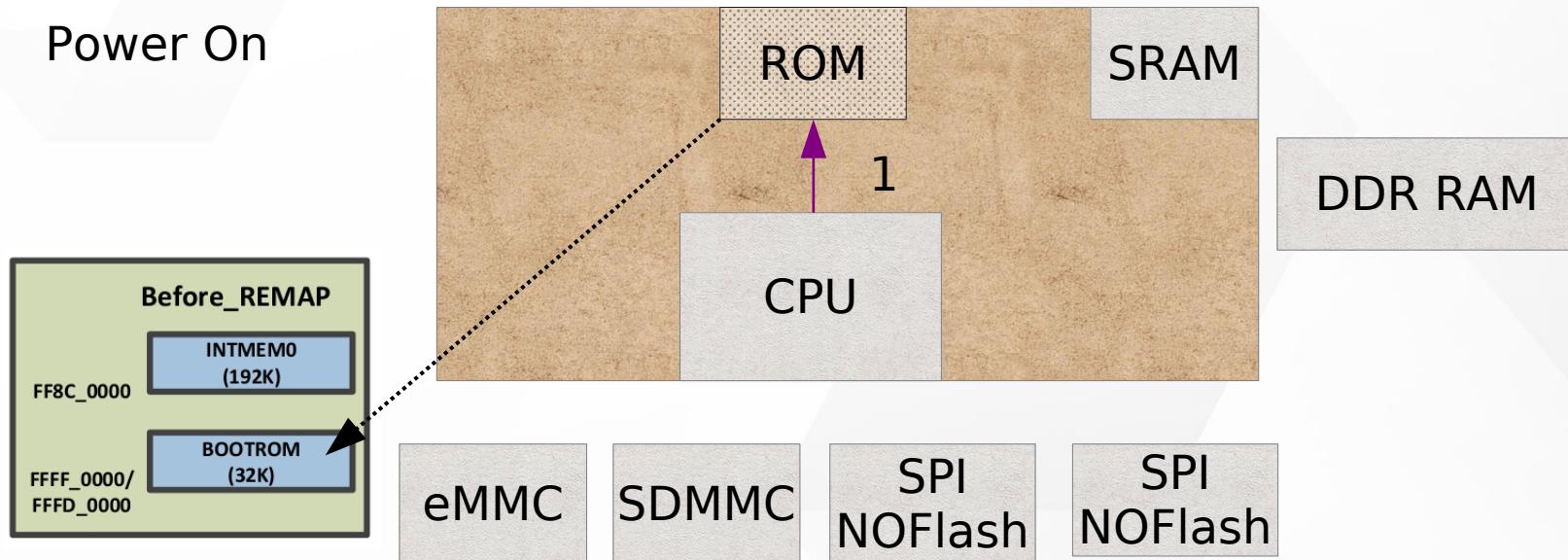


RK3399 System Boot

Reference:

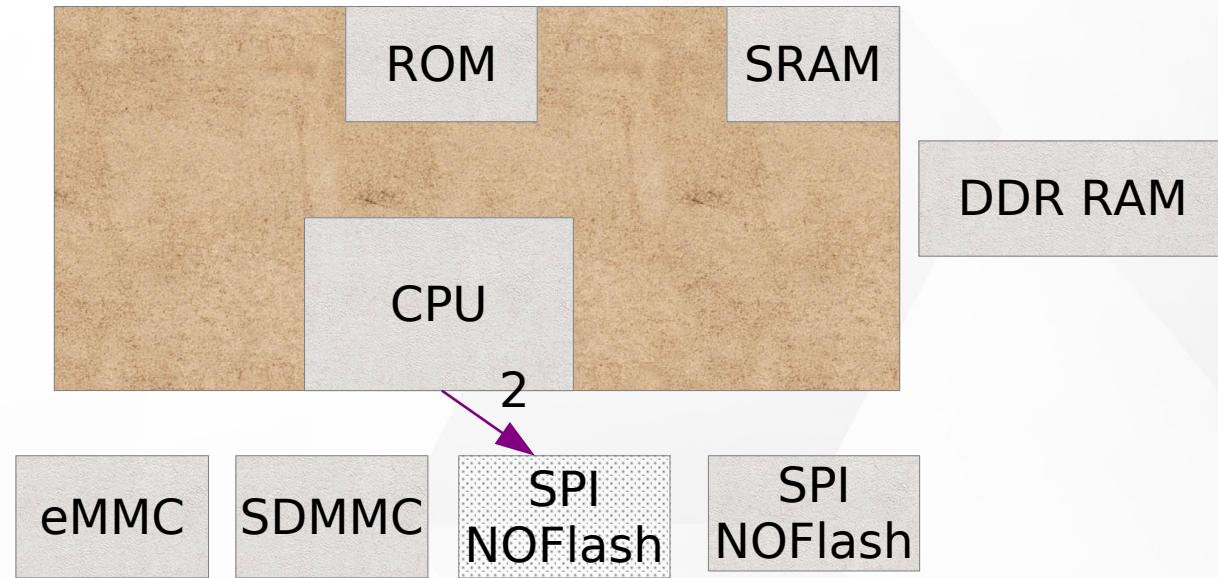
Page 30 of Rockchip_RK3399TRM_V1.3_Part1.pdf

RK3399 System Boot (1)



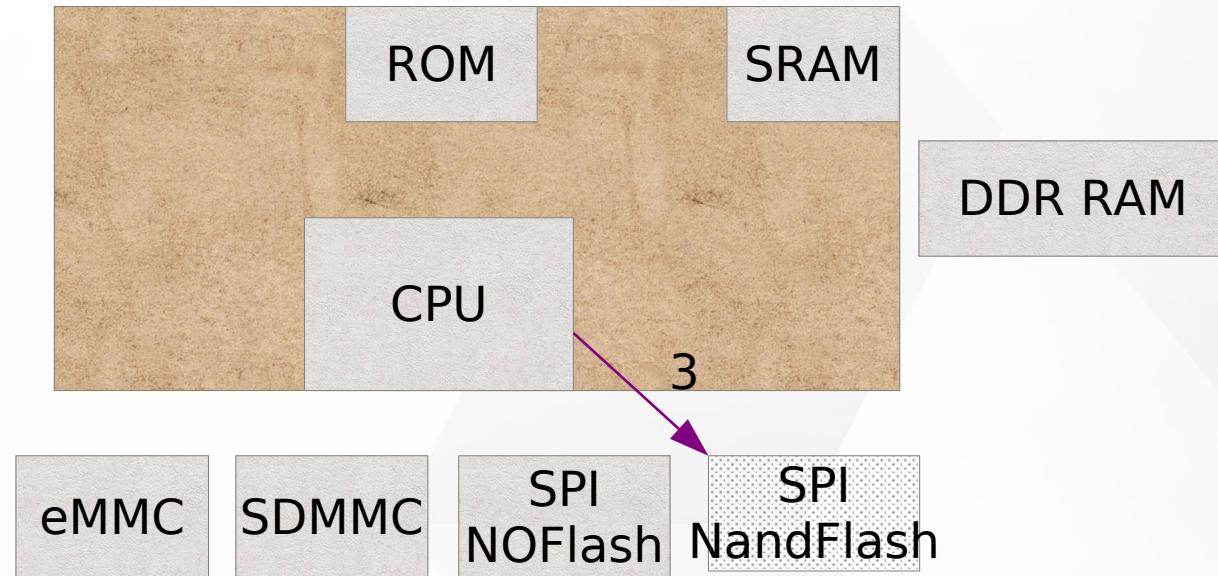
Cortex-A53 get first instruction
from address **0xffff0000 romcode**
start to run

RK3399 System Boot (2)



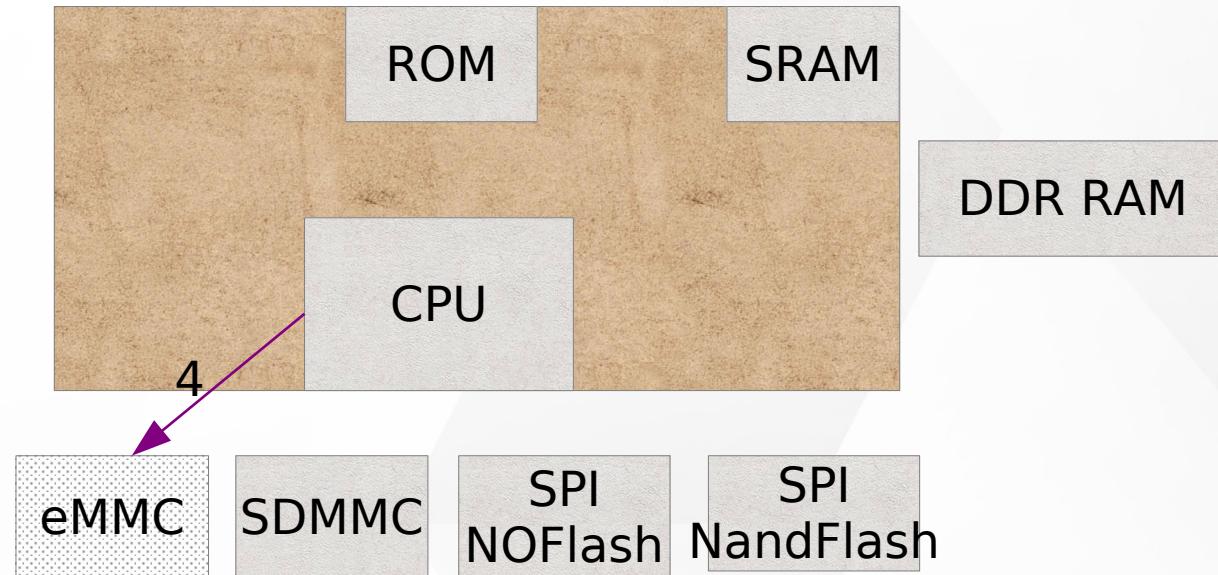
Check ID BLOCK from external **SPI Nor Flash**

RK3399 System Boot (3)



Check ID BLOCK from external **SPI Nand Flash**

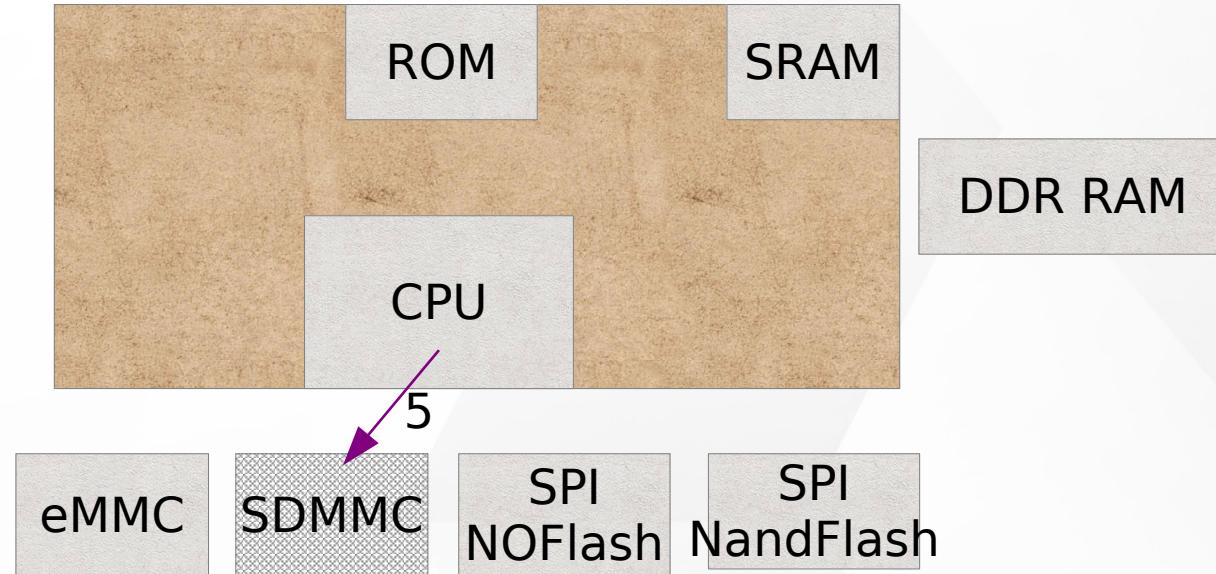
RK3399 System Boot (4)



Check ID BLOCK from external **eMMC**

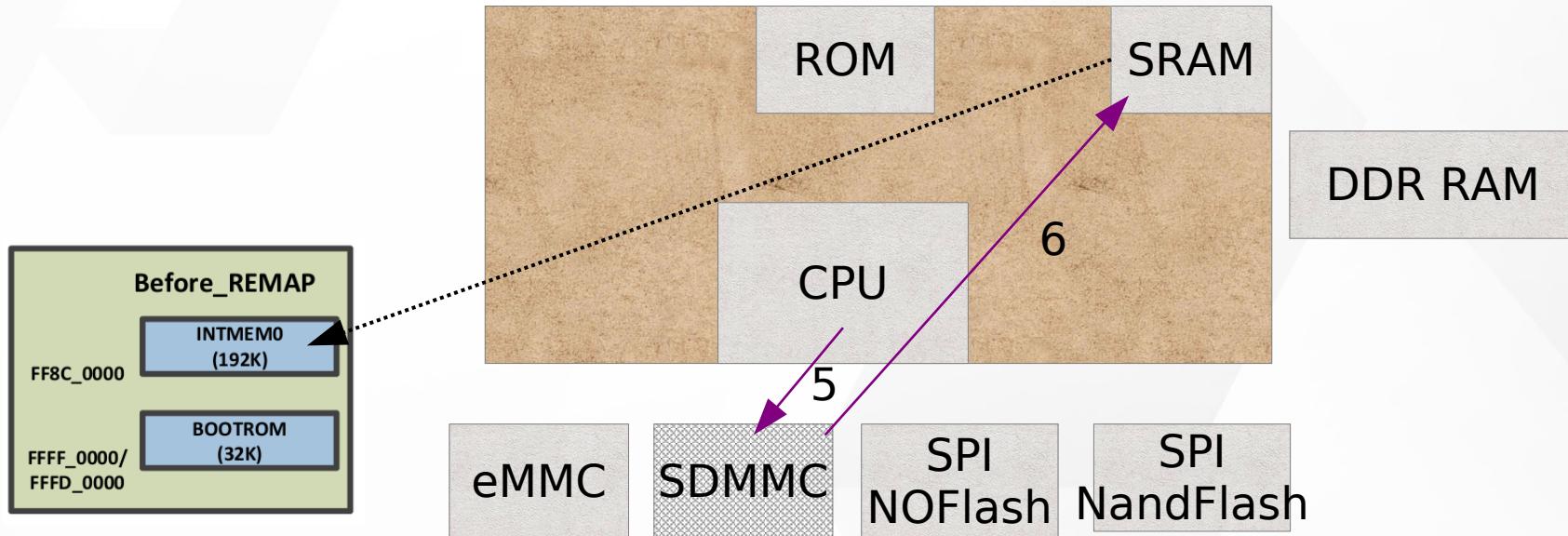
RK3399 System Boot (5)

BL1
work in cache
IDB_Loader



Check ID BLOCK from external **SDMMC**

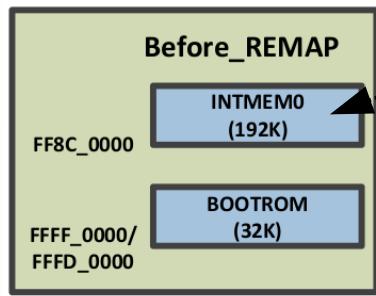
RK3399 System Boot (6)



5. Check ID BLOCK from external **SDMMC**
6. Read **2nK SDRAM initialization image code** to **internal SRAM**

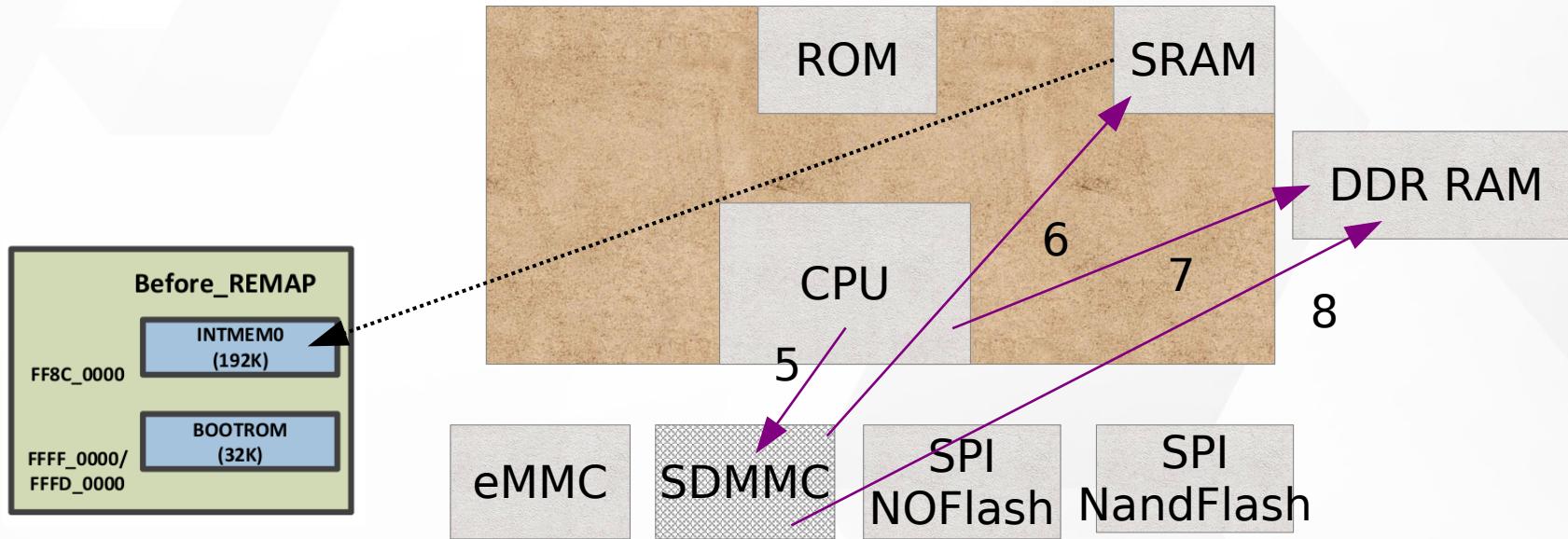
RK3399 System Boot (7)

BL2
work in cache
DDR RAM Init



5. Check ID BLOCK from external **SDMMC**
6. Read **2nK SDRAM initialization image code** to **internal SRAM**
7. Run boot code to do DDR initialization

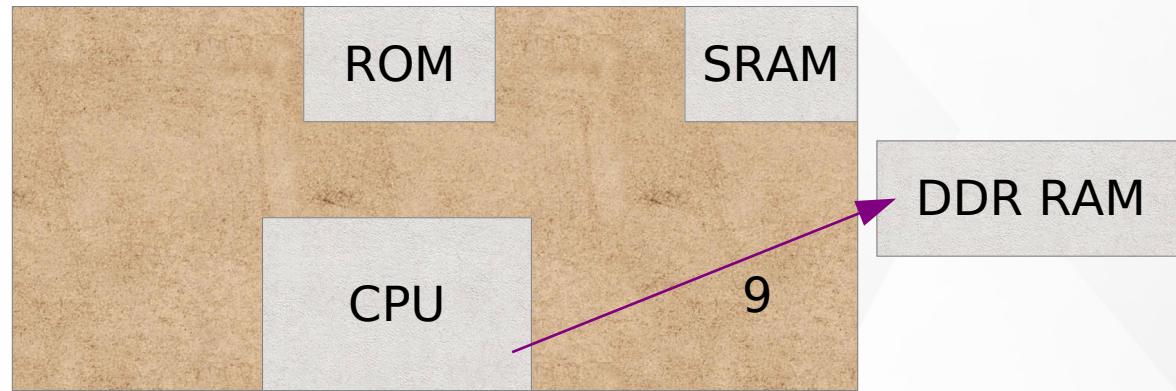
RK3399 System Boot (8)



5. Check ID BLOCK from external **SDMMC**
6. Read **2nK SDRAM initialization image code** to **internal SRAM**
7. Run boot code to do DDR initialization
8. Transfer boot code to DDR then Run boot code

RK3399 System Boot (9)

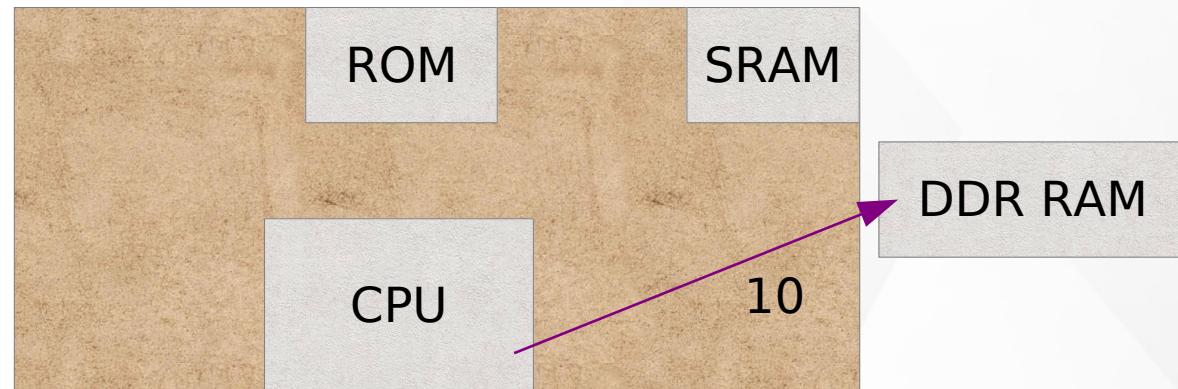
BL3
work in DDR RAM
UBoot



Run UBoot

RK3399 System Boot (10)

work in DDR RAM
Linux



Run Linux



Boot

▶ Power On BootROM code (work in **cache**)

- Load BL1

▶ BL1 (work in **cache** - IDB_Loader)

- **Initial simple exception vectors, PLL (clock)**
- **Initial Multi-CPU**
- Load BL2

▶ BL2 (work in **cache**)

- **Initial DDR memory**
- **Initial C environment** (stack, heap,)
- Load BL31



Boot

▶ BL31 (**work in DDR**)

- Initial exception vectors
- Load BL32 (u-boot)

BL32 U-boot (**work in DDR**)

- **Initial storage device**
- **Load Linux Kernel**

Kernel (**work in DDR**)

- kernel/Documentation/arm64/booting.txt
- **Load RootFS**

Embedded Linux System

User land



Kernel

Virtual File System (VFS)

Linux System Call

Linux Device Driver

Hardware

Disk

Image Sensor

Keyboard

mouse

Panel

CH4 Basic Software & Tool



Software and Tool

» Open Source License

» Develop Tool

- Geany, gedit, vim

- Git

- diff, patch

» Build Code Tool

- ARM Toolchain

- make

- automake, autoconfig



Software and Tool

► Network

- nmcli, ethtool, wpa_supplicant
- SSH, SSHFS
- NFS

► Bluetooth

- hciconfig, bluetoothctl

► Media Software

- gstreamer
- ALSA Tool - aplay, arecord



Software and Tool

➤ Bus

- I2C – i2cset, i2cget, i2cdump
- USB – lsusb

Package Manage



Debian Package Management

- ▶ dpkg : command-line tool for handling packages
- ▶ dpkg --help
 - [CMD] dpkg -i \${PACKAGE_NAME} : install packag
 - [CMD] dpkg -r \${PACKAGE_NAME} : remove package



Ubuntu Package Management

- ▶ apt-get : command-line tool for handling packages
- ▶ apt-get --help
 - [CMD] apt-get update
 - [CMD] apt-get install \${PACKAGE_NAME}
 - [CMD] apt-get remove \${PACKAGE_NAME}
 - [CMD] apt-get autoremove
 - [CMD] apt-get clean

Editor



Editor

► Geany

► [CMD] sudo apt-get install geany

► Vim

► [CMD] sudo apt-get install vim

► Gedit

► [CMD] sudo apt-get install gedit

Diff Tool



diff and patch

➤ diff - compare files line by line

- Create a patch file
 - `diff -Nuar file_a file_b > c.patch`
 - `-N`, treat absent files as empty
 - `-a, --text`
 - `-u`, output NUM (default 3) lines of unified context
 - `-r`, recursively compare any subdirectories found

➤ patch - apply a diff file to an original

- apply a patch file
 - `patch ./hello_1.c < ./tmp.patch`
- Reverse a patch file
 - `patch -R ./hello_1.c < tmp.patch`

Source Code Version Control



Git

- ▶ <https://git-scm.com/book/zh-tw/v1/>
- ▶ 版本控制
- ▶ 程式回溯
- ▶ 管理多人共同開發

Makefile



Makefile

- Simplify compile command
- Automation compile, linker program source
- It can update source in accordance with the dependence



Compile a Hello_World

A

B

C

```
aarch64-linux-gnu-gcc -o helloworld ./hello.c
```

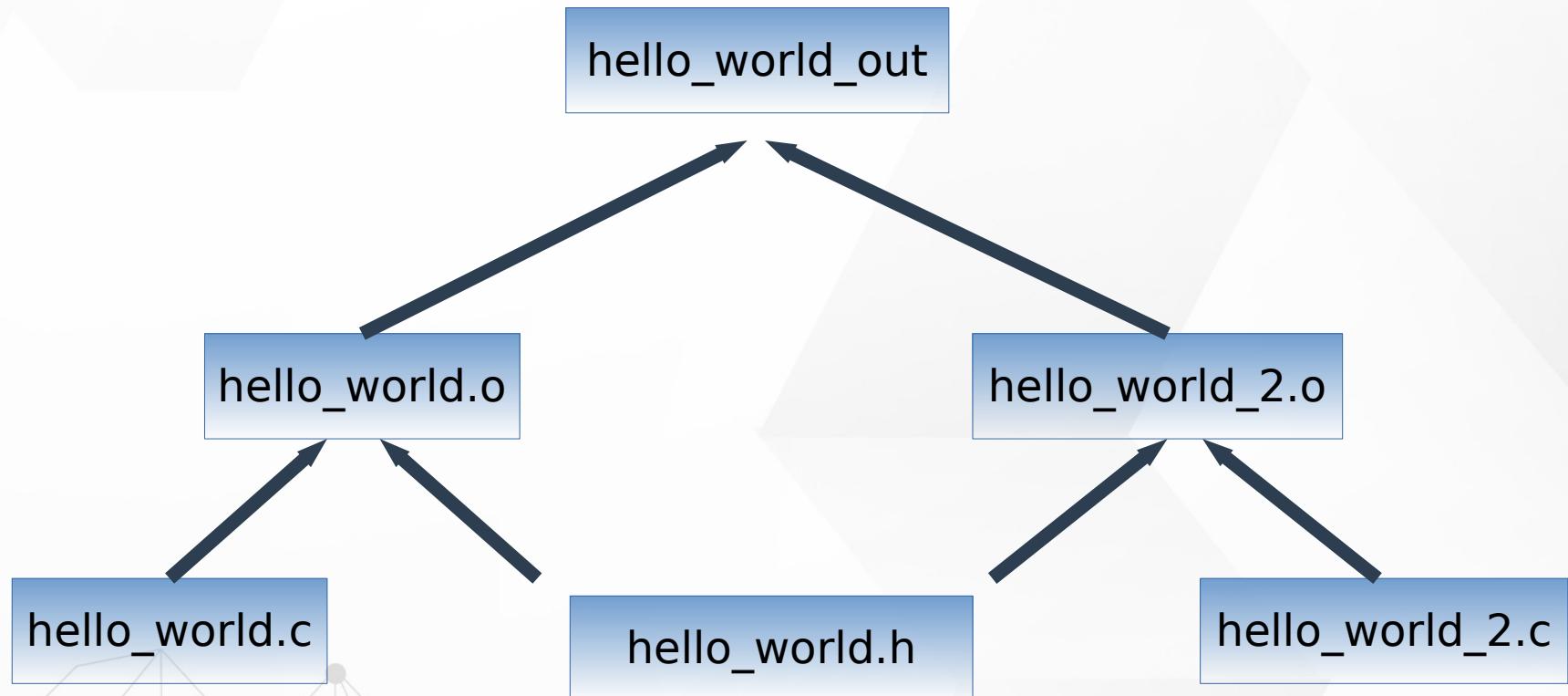
A : ARM C Compile

B : ARM C Compile Parameter
(Output name)

C : C source code



Another Sample





Compile Another Sample

- ▶ Step 1 : gcc -c hello_world.c
- ▶ Step 2 : gcc -c hello_world_2.c
- ▶ Step 3 : gcc -o hello_world hello_world.o hello_world_2.o



Another Sample - Makefile

```
CC=$(CROSS_COMPILE)gcc

all: hello_world

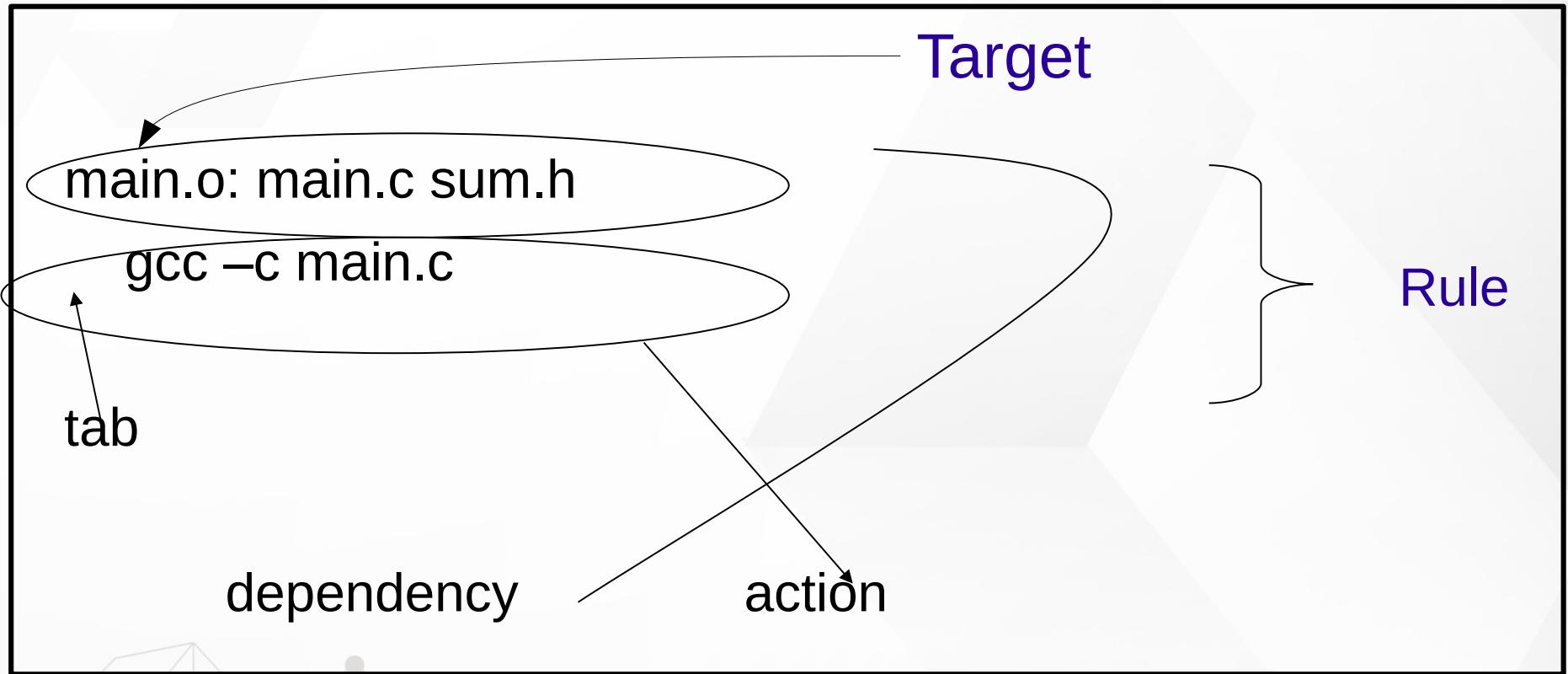
hello_world: hello_world.o hello_world_2.o
    $(CC) -o hello_world hello_world.o hello_world_2.o

hello_world.o: hello_world.c
    $(CC) -c hello_world.c

hello_world_2.o: hello_world_2.c
    $(CC) -c hello_world_2.c

clean:
    rm -r *.o
    rm hello_world
```

Rule Syntax



make 在編譯時，若發現 **target** 比較新，
也就是 **dependencies** 都比 **target** 舊，
那麼將不會重新建立 **target**，如此可以避免不必要的編譯動作

Rule Syntax

```
CC=$(CROSS_COMPILE)gcc
```

```
CFLAGS=-Wl,-Map,out.map -lpthread -lm
```

```
all: hello_world
```

```
hello_world: hello_world.o hello_world_2.o
```

```
$(CC) -o hello_world hello_world.o hello_world_2.o
```

```
hello_world.o: hello_world.c
```

```
$(CC) $(CFLAGS) -c hello_world.c
```

```
hello_world_2.o: hello_world_2.c
```

```
$(CC) $(CFLAGS) -c hello_world_2.c
```

```
clean:
```

```
rm -r *.o
```

```
rm hello_world
```



hello_world_ex1

```
CC=$(CROSS_COMPILE)gcc

AA='1234' '5678'
AA := 'DDDD'

$(info AA=$(AA))

CFLAGS=-Wl,-Map,out.map -lpthread -lm

all: hello_world

hello_world: hello_world.o
    → $(CC) -o hello_world hello_world.o

hello_world.o: hello_world.c
    → $(CC) $(CFLAGS) -c hello_world.c

clean:
    → rm -r *.o
    → rm hello_world
```

→ : Tab



Assignment Operators

- = 定義一個 需做遞迴展開的 變數型態
- := 定義一個 立即運作的 變數型態
- += 將 指定值， 繼加在 原變數中
- ?= 如果之前 無任何設定該變數， 即現在設定，
否則 跳過設定（就是不做任何事）



Assignment Operators Sample 1

```
AA ='1234' '5678'  
BB = ${AA}  
AA = '789'  
AA += 'ABCDE'
```

Output

```
AA='789' 'ABCDE'  
BB='789' 'ABCDE'
```



Assignment Operators Sample 2

```
AA ='1234' '5678'  
BB := ${AA}  
AA = '789'  
AA += 'ABCDE'
```

Output

```
AA='789' 'ABCDE'  
BB='1234' '5678'
```



Assignment Operators Sample 3

```
AA ='1234' '5678'  
BB := ${AA}  
AA = '789'  
AA ?= 'ABCDE'
```

Output

```
AA='789'  
BB='1234' '5678'
```



Command-Line Options

▶ **-C dir, --directory= dir**

- make changes the current working directory to dir before it does anything else. If the command line includes multiple -C options, each directory specified builds on the previous one

▶ **-j [number] , --jobs[= number]**

- Run multiple commands in parallel

Bus Tool



Device Tool

➤ Real Time Clock

➤ hwclock

- -s, --hctosys
- -r, --show

set the system time from the RTC
display the RTC time

➤ USB

➤ lsusb

➤ Block Device

➤ lsblk

➤ I2c-tools

➤ i2cdump, i2cdetect, i2cget, i2cset

Media Tool

Gstreamer

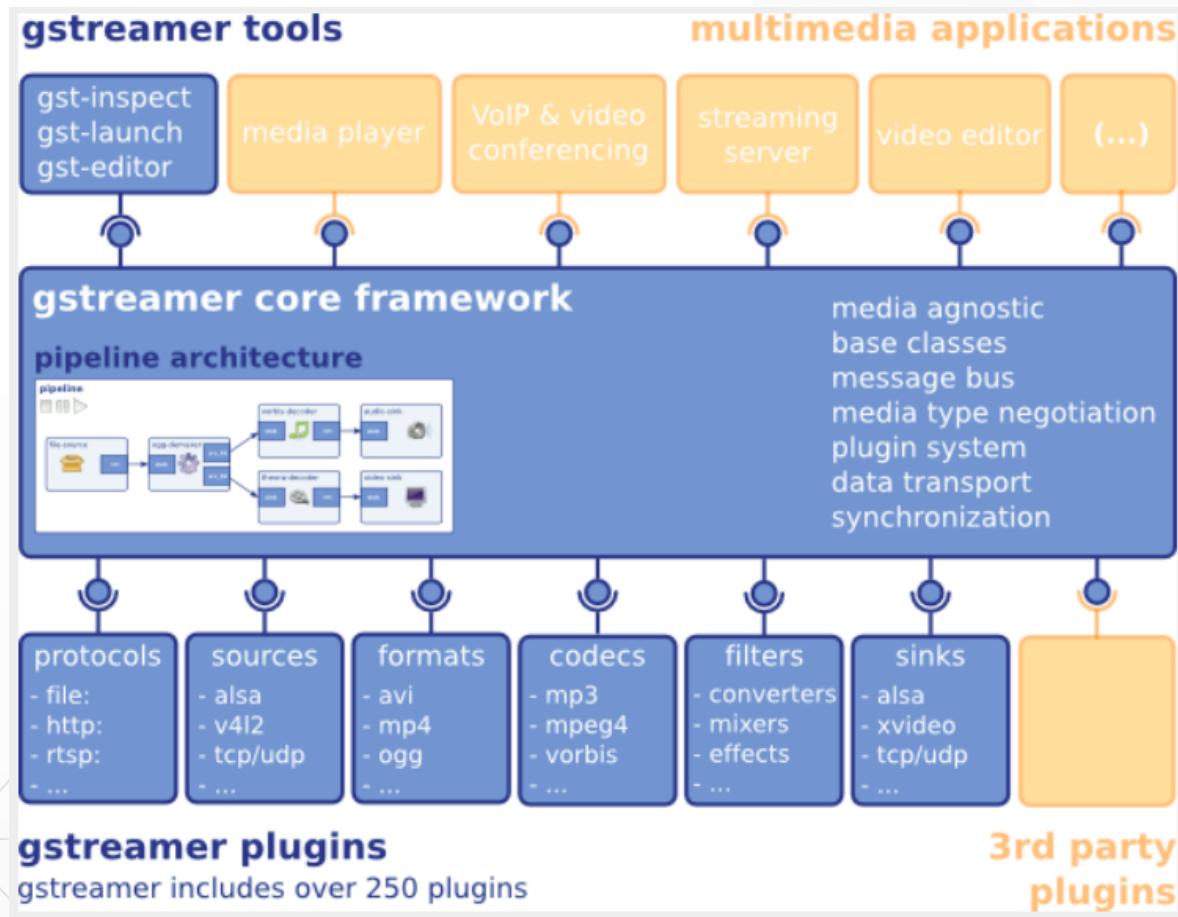


Open Source Multimedia Framework

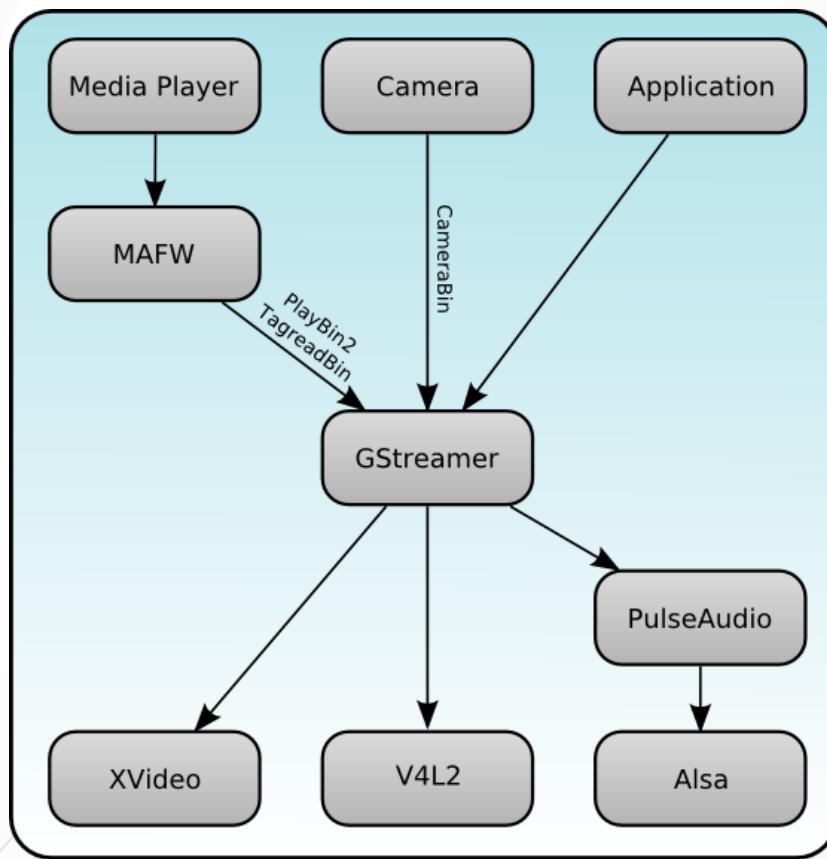
- ▶ <https://gstreamer.freedesktop.org/>
- ▶ Documentation
- ▶ Tutorials



Block Diagram



Block Diagram



http://maemo.org/development/sdks/maemo_5_beta_docs/using_multimedia_components/



Overview

- ▶ GStreamer is a **framework** for creating streaming media applications
- ▶ The framework is based on **plugins** that will provide the various codec and other functionality

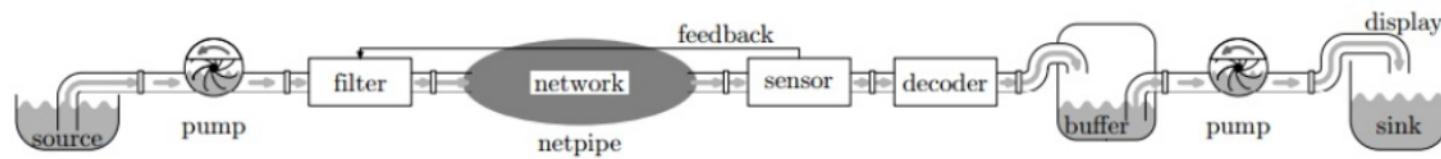


Overview

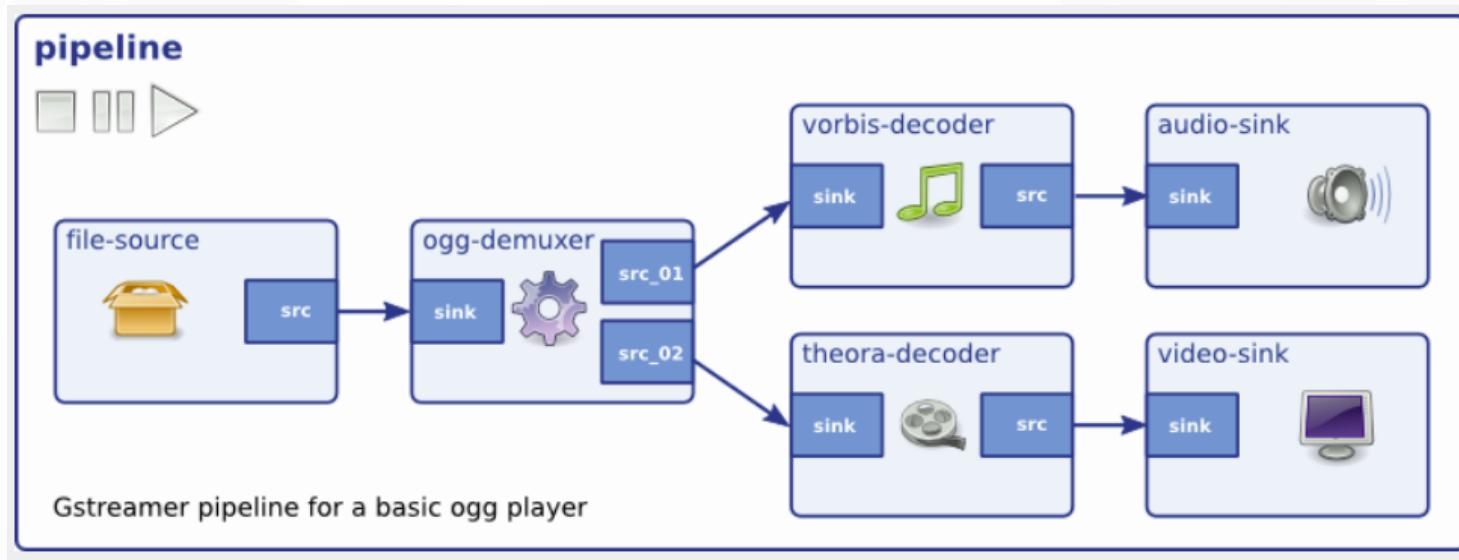
- ▶ **Gst-plugins-base:**
an essential exemplary set of elements
- ▶ **Gst-plugins-good:**
a set of good-quality plug-ins under LGPL
- ▶ **Gst-plugins-ugly:**
a set of good-quality plug-ins that might pose distribution problems
- ▶ **Gst-plugins-bad:**
a set of plug-ins that need more quality

Gstremer Pipe

A streamer pipe



Gstremer Pipe





Gstreamer - Tool

▶ Gst-inspect-1.0

 ▶ Print supported plug-in

▶ Gst-launch-1.0

 ▶ Gstreamer tester

▶ Gst-typefind-1.0

 ▶ Check file for gstreamer plug-in type



Gstreamer - Tool

➤ Gst-inspect-1.0

➤ Check what kind of videosink in Rockpi4b

➤ gst-inspect-1.0 | grep -i videosink

```
rock@rockpi4b:~$ gst-inspect-1.0 | grep -i videosink
debugutilsbad: fakevideosink: Fake Video Sink
debugutilsbad: fpsdisplaysink: Measure and show framerate on videosink
inter: intervideosink: Internal video sink
decklink: decklinkvideosink: Decklink Video Sink
autodetect: autovideosink: Auto video sink
rock@rockpi4b:~$
```

Gststreamer - Tool



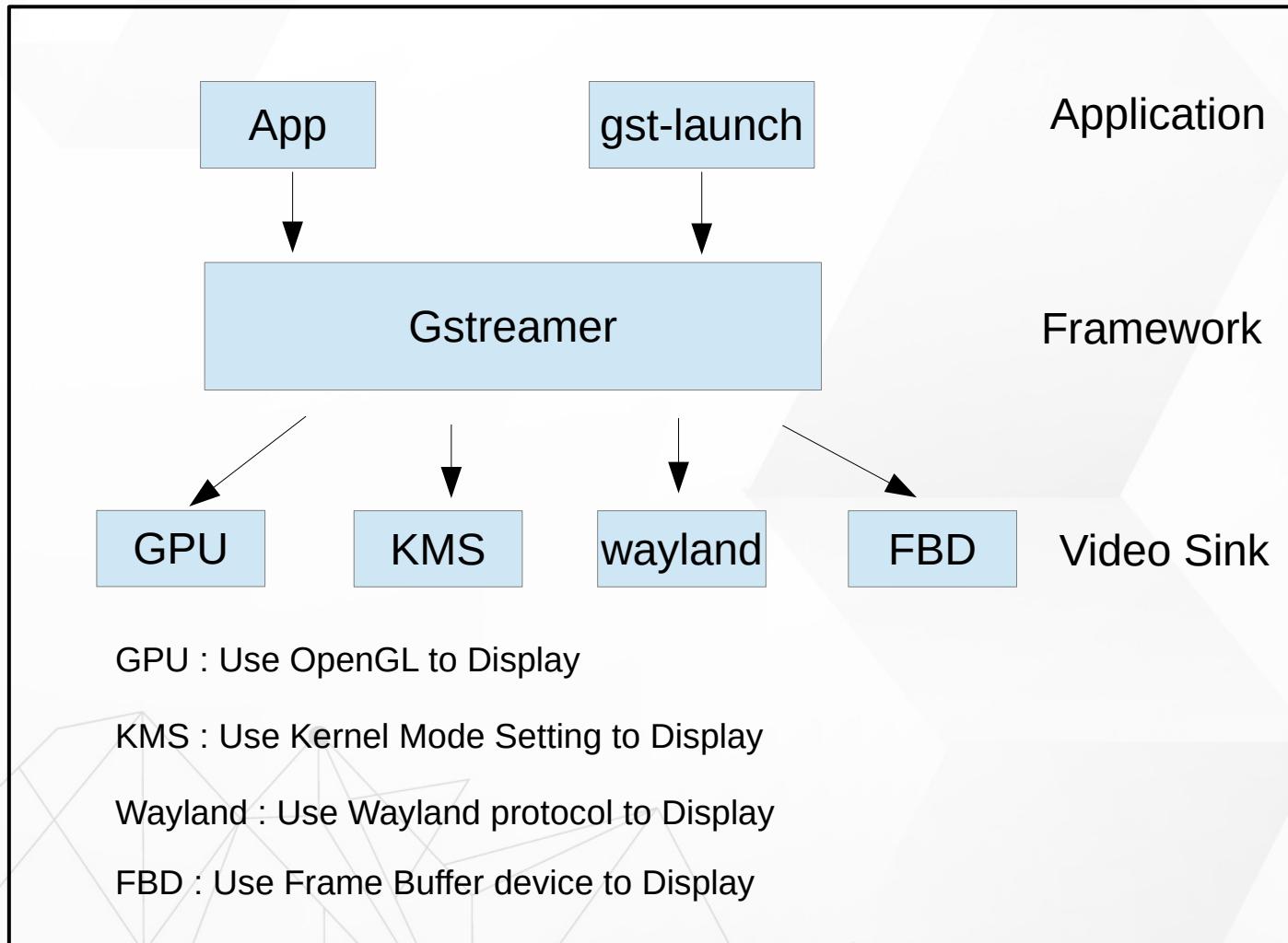
▶ Gst-typefind-1.0

▶ Check Serenity-DVD-320x240.m4v what kind of file type in gstreamer

▶ gst-typefind-1.0 ./Serenity-DVD-320x240.m4v

```
rock@rockpi4b:~$ gst-typefind-1.0 ./Serenity-DVD-320x240.m4v
./Serenity-DVD-320x240.m4v - video/quicktime, variant=(string)iso
```

Gstreamer - Video



GPU : Use OpenGL to Display

KMS : Use Kernel Mode Setting to Display

Wayland : Use Wayland protocol to Display

FBD : Use Frame Buffer device to Display

Play Video Test Stream



```
gst-launch-1.0 videotestsrc ! video/x-raw, width=1280, height=720 ! kmssink
```



Play a H.264 video

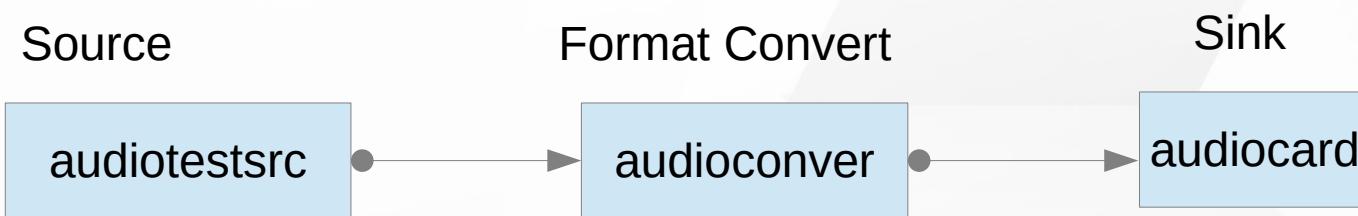
```
gst-launch-1.0 filesrc location=/tmp/Serenity-DVD-320x240.m4v ! \
decodebin name=dec ! \
videoconvert ! \
kmssink
```





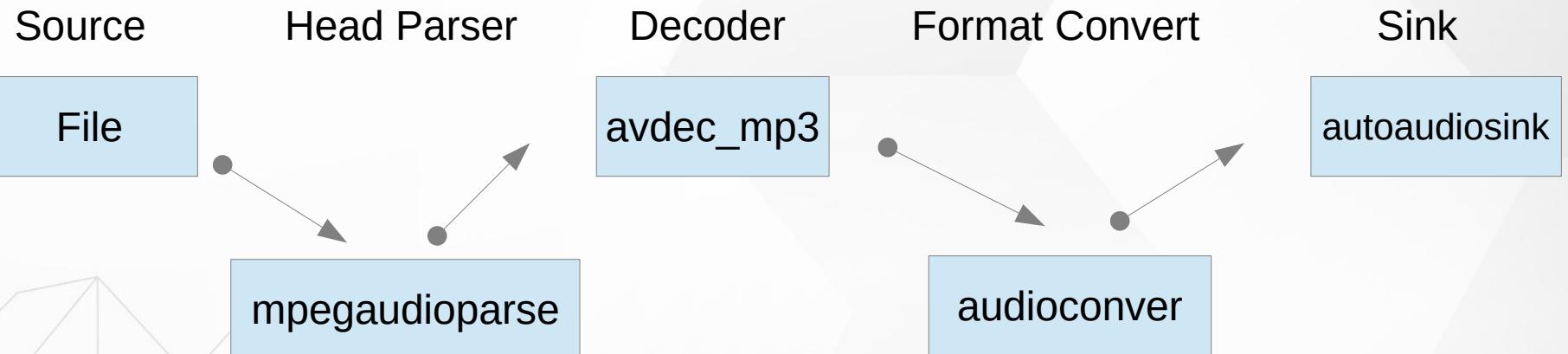
Play a Audio Test

```
gst-launch-1.0 audiotestsrc ! audioconvert ! alsasink device-name=rockchipes8316c
```



Play a MP3

```
gst-launch-1.0 filesrc location="/home/cadtc/audio/piano2-CoolEdit.mp3" ! \
mpegaudioparse ! \
mpg123audiodec ! \
audioconvert ! autoaudiosink
```





ALSA Tool

▶ ALSA Util

- **aplay**
 - Play a WAV file
- **arecord**
 - Record a sound
- **alsamixer**
 - A graph tool for adjusting audio gain
- **amixer**
 - A console tool for adjusting audio gain



ALSA Tool

▶ ALSA Utile

- **aplay**

- aplay -Dsysdefault:CARD=rockchipes8316c /usr/share/sounds/alsa/Front_Center.wav
- aplay -Dsysdefault:CARD=HDMICODEC /usr/share/sounds/alsa/Front_Center.wav

- **arecord**

- arecord -Dhw:0,0 -r 44100 -t wav -f CD -d 5 /tmp/test.wav

- **alsamixer**

- alasmixer

- **amixer**

- amixer scontrols | less
- amixer sget 'HP' 0%
- amixer sset 'HP' 0%

WiFi and Network



Network Tool

- » ifconfig → Network setting
- » ping → Network package test
- » iperf3 → perform network throughput tests
- » dhcpc → used for automatic retrieving of



WIFI

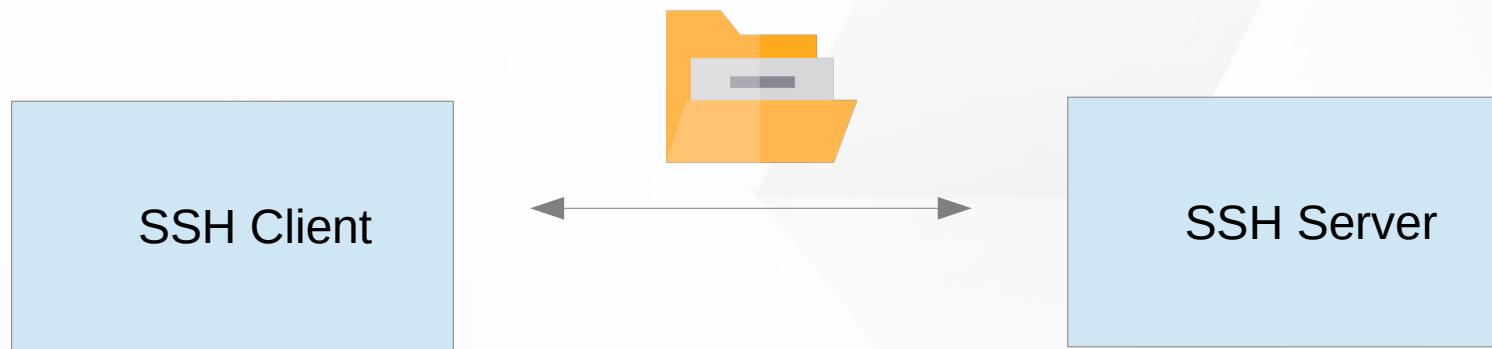
- ▶ NetworkManager Command Line Tool
 - ▶ nmcli
- ▶ Show / manipulate wireless devices and their configuration
 - ▶ iw
- ▶ For connecting to a WPA/WPA2 network
 - ▶ wpa_supplicant

SSH



SSH

- ▶ Secure SHell protocol
- ▶ SSH Client
- ▶ SSH Server





SSH

▶ SSH Client

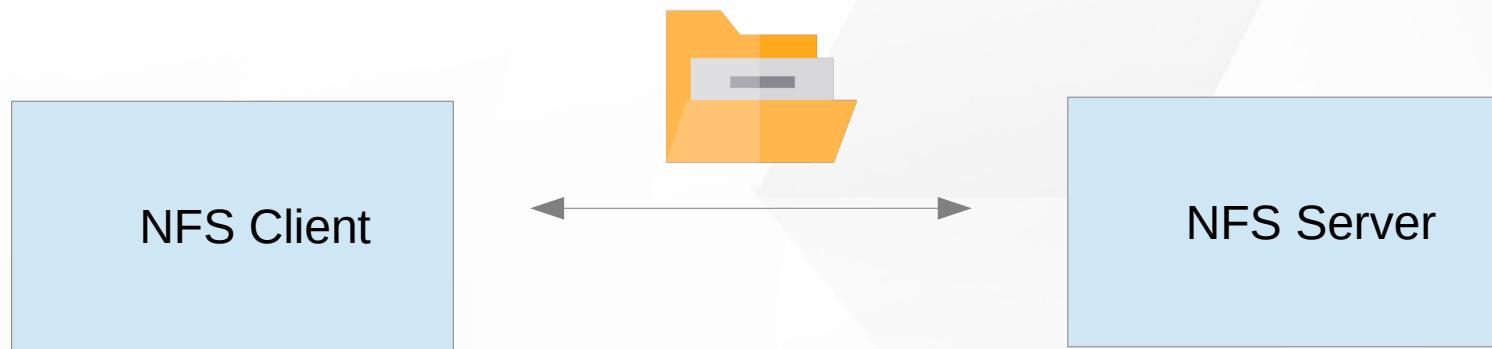
- `# sudo apt-get install ssh`
- [References](#)

NFS



NFS

- ▶ Network File System
- ▶ NFS Client
- ▶ NFS Server



Docker



What is Docker

➤ <https://docs.microsoft.com/zh-tw/dotnet/architecture/containerized-lifecycle/what-is-docker>

➤ 簡單的說

➤ Docker : 模擬 作業系統的 “User space”，沒有底層

➤ VirtualBox : 模擬 整個 作業系統，包含 底層 與 硬體

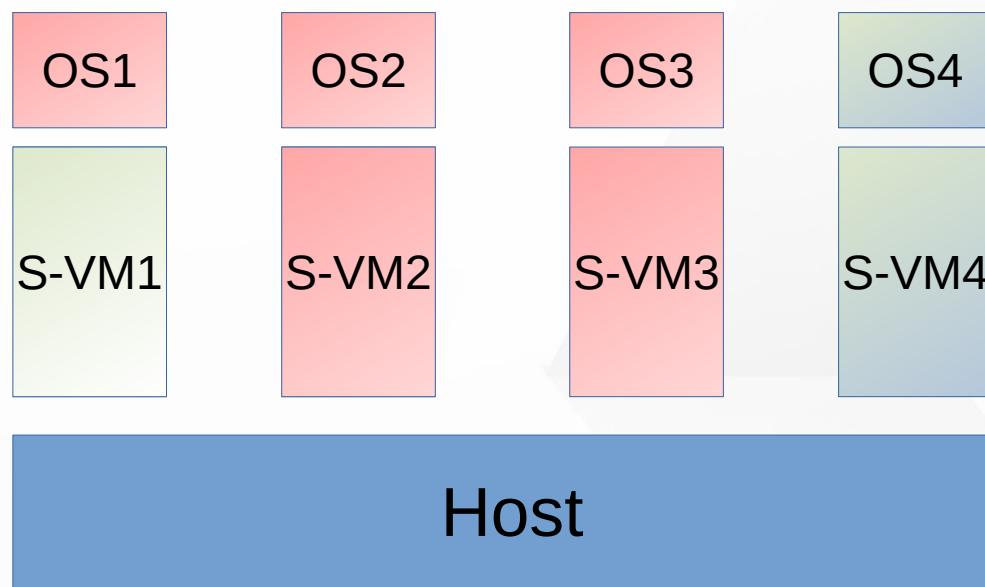
➤ 優點

➤ 對於開發 User space 的程式方便交換環境，快速

➤ 缺點

➤ 無法處理 底層 或是 硬體 相關 部份

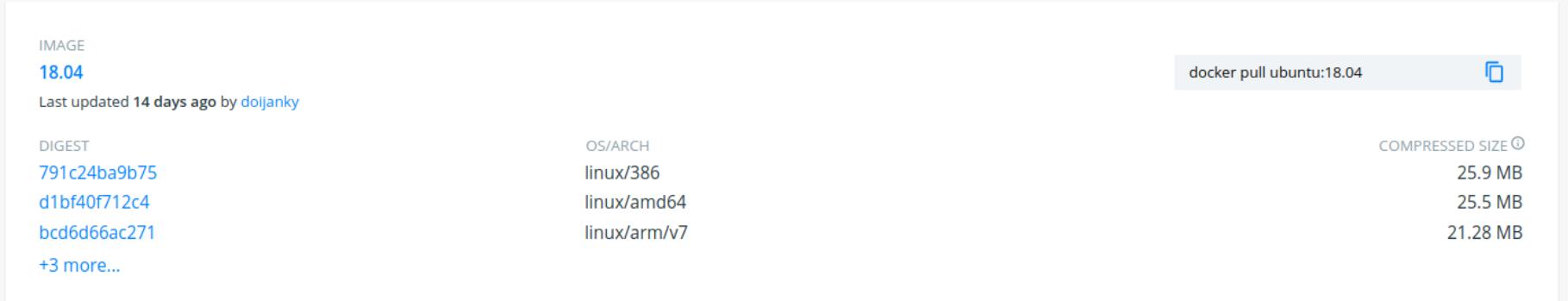
What is Docker





Docker Hub

- » <https://hub.docker.com/>
- » https://hub.docker.com/_/ubuntu
- » Ubuntu 官方已在 Docker Hub 建立不同版本的 docker image
- » 直接從 Docker Hub 下載需要的新環境
 - » 例如 : Ubuntu 18.04



The screenshot shows the Docker Hub page for the Ubuntu 18.04 image. It includes details like the image name, last update, digests, OS/architecture, and compressed size.

IMAGE	DIGEST	OS/ARCH	COMPRESSED SIZE
18.04	791c24ba9b75 d1bf40f712c4 bcd6d66ac271 +3 more...	linux/386 linux/amd64 linux/arm/v7	25.9 MB 25.5 MB 21.28 MB
			docker pull ubuntu:18.04 



Install a Docker in Ubuntu

▶ Install docker in Ubuntu

```
▶ $ sudo apt-get install docker
```

▶ Docker help

```
▶ $ docker --help
```



How to use Docker

▶ Pull a Docker image that in Docker Hub

▶ For example : ubuntu 18.04

▶ \$ docker pull ubuntu:18.04

```
slash@slash-HD631-Q87CRM:~$ dicker images
No command 'dicker' found, did you mean:
  Command 'ticker' from package 'ticker' (universe)
  Command 'docker' from package 'docker.io' (universe)
dicker: command not found
slash@slash-HD631-Q87CRM:~$ docker images
REPOSITORY          TAG           IMAGE ID      CREATED        SIZE
slash@slash-HD631-Q87CRM:~$ docker pull ubuntu:18.04
18.04: Pulling from library/ubuntu
f08d8e2a3ba1: Extracting [>                               ] 294.9kB/26.7MB
3baa9cb2483b: Download complete
94e5ff4c0b15: Download complete
1860925334f9: Download complete
```



How to use Docker-2

▶ Check docker Image in local

▶ \$ docker images

```
slash@slash-HD631-Q87CRM:~$ docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
ubuntu          18.04   6526a1858e5d  2 weeks ago  64.2MB
slash@slash-HD631-Q87CRM:~$
```

▶ start to run docker Image in local

▶ \$ docker run --help

▶ \$ docker run --net=host -i -t 6526a1858e5d /bin/bash

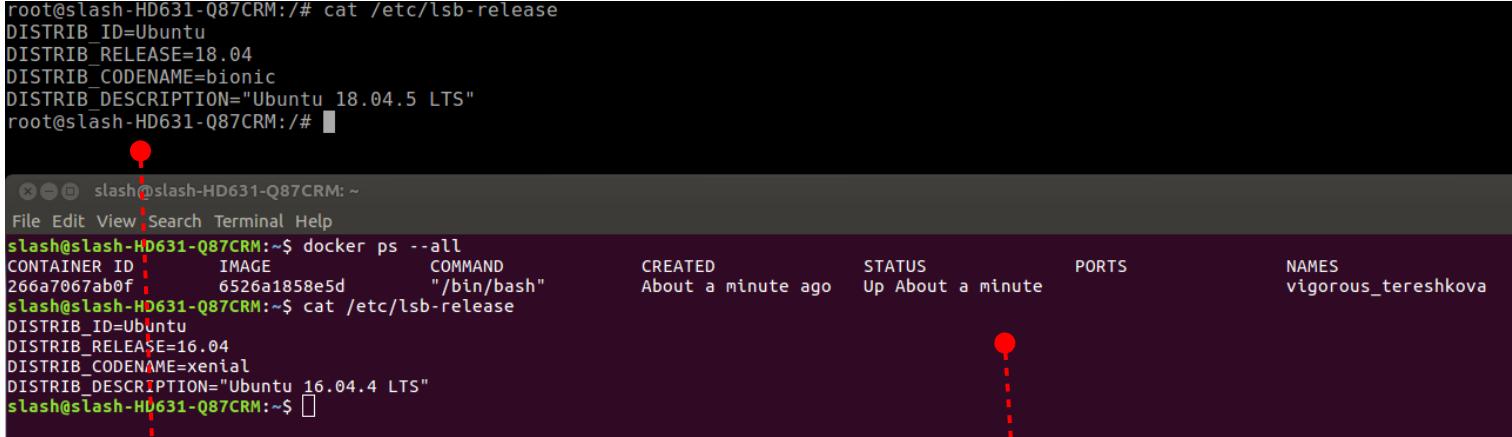
```
slash@slash-HD631-Q87CRM:~$ docker run -i -t --net=host 6526a1858e5d /bin/bash
root@slash-HD631-Q87CRM:/#
```

Enter docker content



How to use Docker-3

▶ Check docker content with docker command



The screenshot shows a terminal window with two sessions. The top session is a root shell on the host system, displaying the contents of /etc/lsb-release. The bottom session is a root shell inside a Docker container named 'vigorous_tereshkova', also displaying the contents of /etc/lsb-release. Red arrows point from the text in each session to the corresponding text in the other session, illustrating that the container's content matches the host's.

```
root@slash-HD631-Q87CRM:/# cat /etc/lsb-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=18.04
DISTRIB_CODENAME=bionic
DISTRIB_DESCRIPTION="Ubuntu 18.04.5 LTS"
root@slash-HD631-Q87CRM:/#
slash@slash-HD631-Q87CRM:~$ docker ps --all
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
266a7067ab0f        6526a1858e5d      "/bin/bash"        About a minute ago   Up About a minute   vigorous_tereshkova
slash@slash-HD631-Q87CRM:~$ cat /etc/lsb-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=16.04
DISTRIB_CODENAME=xenial
DISTRIB_DESCRIPTION="Ubuntu 16.04.4 LTS"
slash@slash-HD631-Q87CRM:~$
```

Docker content (ubuntu 18.04)

Host (ubuntu 20.04)



How to use Docker-4

➤ Install package that you want in docker image

➤ For example

- \$ apt-get install net-tools

```
root@slash-HD631-Q87CRM:/# apt-get install net-tools
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  net-tools
0 upgraded, 1 newly installed, 0 to remove and 5 not upgraded.
Need to get 194 kB of archives.
After this operation, 803 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu bionic/main amd64 net-tools amd64 1.60+git20161116.90da8a0-1ubuntu1 [194 kB]
Fetched 194 kB in 2s (115 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package net-tools.
(Reading database ... 4046 files and directories currently installed.)
Preparing to unpack .../net-tools_1.60+git20161116.90da8a0-1ubuntu1_amd64.deb ...
Unpacking net-tools (1.60+git20161116.90da8a0-1ubuntu1) ...
Setting up net-tools (1.60+git20161116.90da8a0-1ubuntu1) ...
root@slash-HD631-Q87CRM:/#
```



How to use Docker-5

» Commit a docker content and push it to Docker Hub

» 1. check which docker content that you want to commit

- \$ docker ps --all

```
slash@slash-HD631-Q87CRM:~$ docker ps --all
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
266a7067ab0f        6526a1858e5d      "/bin/bash"         16 hours ago       Up 12 minutes
slash@slash-HD631-Q87CRM:~$
```

» 2. check REPOSITORY and TAG of docker image current

- ```
slash@slash-HD631-Q87CRM:~$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
ubuntu 18.04 6526a1858e5d 2 weeks ago 64.2MB
```

### » 3. commit docker content

- \$ docker commit -m "test" 266a7067ab0f ubuntu:18.04

```
slash@slash-HD631-Q87CRM:~$ docker commit -m "test" 266a7067ab0f ubuntu:18.04
sha256:769ce8b9f38081b4cedf764b51e070ba4032b8146f26cf5c5d6726a3f20b831a
slash@slash-HD631-Q87CRM:~$ rm -rf ./build-fb/^C
slash@slash-HD631-Q87CRM:~$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
ubuntu 18.04 769ce8b9f380 11 seconds ago 94.9MB
ubuntu <none> 6526a1858e5d 2 weeks ago 64.2MB
```



# How to use Docker-6

▶ Import a Local Docker Image (Load)

▶ [CMD] docker load < docker\_image\_file.tar

▶ Save a Local Docker Image (Export)

▶ [CMD] docker save \${IMAGE\_ID} > docker\_image\_file.tar



# How to use Docker-7

## ▶ Delete a docker **Content**

▶ \$ docker ps --all

▶ \$ docker rm \${Content ID}

## ▶ Delete a docker **Image**

▶ \$ docker images

▶ \$ docker rmi \${IMAGE ID}



# A Sample Docker Usage

➤ **docker\_info.sh**

➤ Set environment variables

➤ -e --env

➤ -e DISPLAY

➤ Execute application when login to docker content

➤ -c --cpu-shares

➤ -c "\${DOCKER\_BUILD\_CMD}"

➤ Work directory

➤ -w --workdir

➤ -w \${DOCKER\_WORKDIR}

# A Sample Docker Usage

➤ Login with user name

➤ -u --user

➤ -u \${DOCKER\_LOGIN\_ID}

➤ Mount a local disk space in docker content

➤ -v --volume

➤ -v \${LOCAL\_FOLDER}:\${DOCKER\_FOLDER}

➤ Allocate a pseudo-TTY

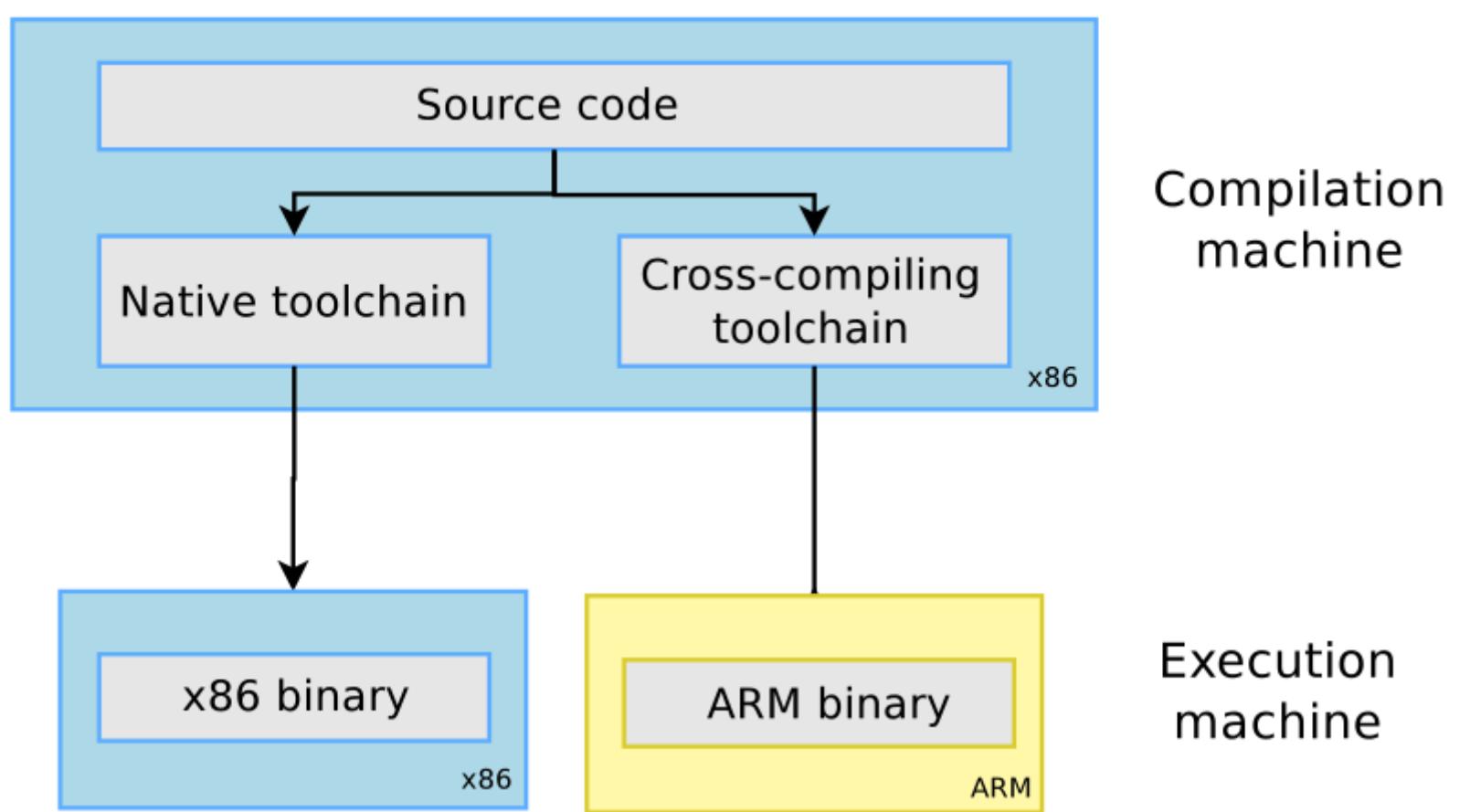
➤ -t --tty

➤ Keep STDIN open even if not attached

➤ -i --interactive

# CH5 Cross Compilation Toolchain

# Cross Compilation Tool-chain





# GCC Components

- » The GNU C Compiler
- » The GNU Compiler Collection

Binutils

Kernel head

C/C++ libraries

GCC compiler

GDB debugger



# Binutils

## » Binutils

- » **as** : the assembler, that generates binary code from assembler source code
- » **ld** : the linker
- » **ar, ranlib** : to generate .a archives, used for libraries
- » **objdump, readelf, size, nm, strings** : to inspect binaries
- » **strip** : to strip useless parts of binaries in order to reduce their size



# Kernel head

- The C library and compiled programs needs to interact with the kernel
- Compiling the C library requires kernel headers, and many applications also require them
- The kernel to user space ABI is backward compatible



# GCC

- GCC originally stood for the "GNU C Compiler."
- GNU Compiler Collection
  - C, C++, Ada, Objective-C, Fortran, JAVA ...
- <http://gcc.gnu.org/>



# GCC flag (1)

► arm-linux-gnueabihf-gcc –help

- -c : Compile and assemble, but do not link
- -o <file> : Place the output into <file>
- -shared : Create a shared library
- -g : add debug information
- -O : sets the compiler's optimization level



# GCC flag (2)

► arm-linux-gnueabihf-gcc –help

- -Wall : enables all compiler's warning messages
- -D : defines a macro to be used by the preprocessor
- -I : adds include directory of header files
- -L,-l :
  - -L looks in directory for library files
  - -l links with a library file



# C library

- » The C library is an essential component of a Linux system
- » Several C libraries are available:
  - **glibc, uClibc, eglIBC, dietlibc, newlib**
- » The choice of the C library must be made at the time of the cross-compiling toolchain generation, as the GCC compiler is compiled against a specific C library.



# Floating point support

- For processors having a **floating point unit**, the toolchain should generate hard float code, in order to use the floating point instructions directly
- For processors without a floating point unit
  - Generate hard float code and rely on the kernel to emulate the floating point instructions
  - Generate soft float code, so that instead of generating floating point instructions, calls to a user space library are generated



# Floating point support

<https://www.linaro.org/downloads/>

| Latest Linux Targeted Binary Toolchain Releases |                                                  |
|-------------------------------------------------|--------------------------------------------------|
| arm-linux-gnueabihf                             | 32-bit Armv7 Cortex-A, hard-float, little-endian |
| armv8l-linux-gnueabihf                          | 32-bit Armv8 Cortex-A, hard-float, little-endian |
| aarch64-linux-gnu                               | 64-bit Armv8 Cortex-A, little-endian             |

## RockPi4 Toolchain

<https://releases.linaro.org/components/toolchain/binaries/7.3-2018.05/aarch64-linux-gnu/>



# Obtain a Toolchain

## ▶ Building a cross-compiling toolchain by ourself

- Crosstool-NG
- <http://crosstool-ng.org/#introduction>

## ▶ Pre-build toolchain

- Linaro - <https://www.linaro.org/downloads/>
- By Linux distribution -
  - `sudo apt-get install gcc-arm-linux-gnueabi`
- BSP
- CodeSourcery



# Installing and using Toolchain

► Add the path to toolchain binaries in your PATH: export

- [CMD] `PATH=${TOOLCHAIN_PATH}/bin/:$PATH`

► Compile your applications

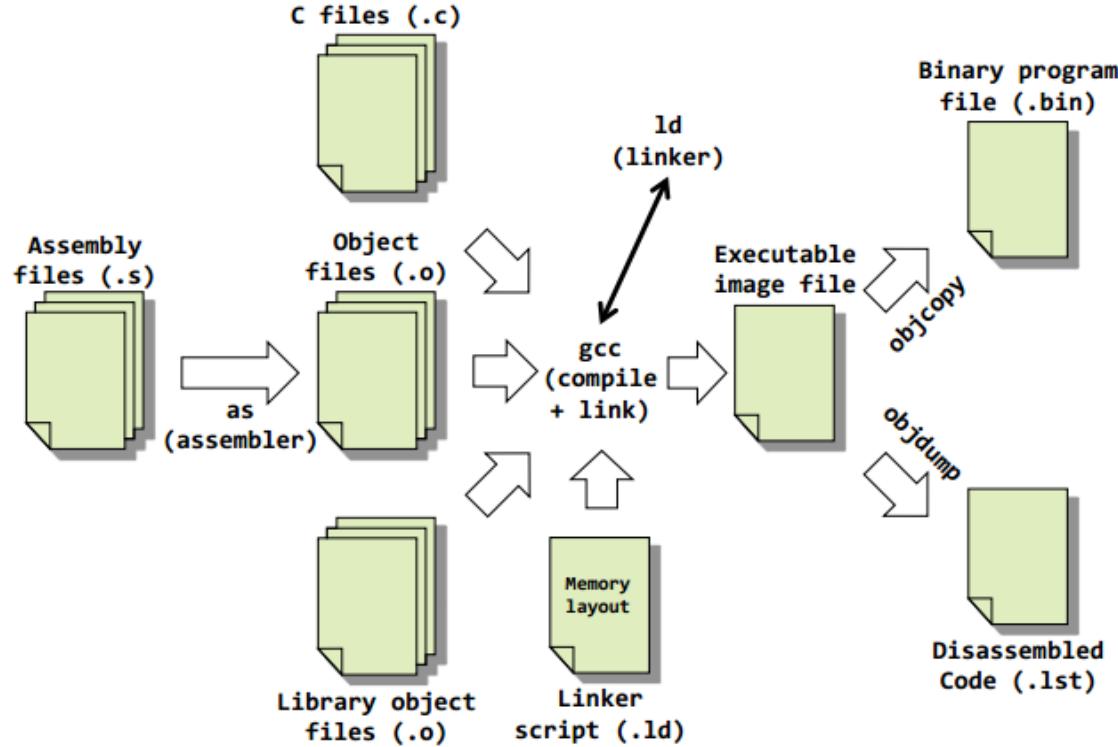
- [CMD]  `${PREFIX}-gcc -o testme testme.c`

► **PREFIX**

- depends on the toolchain configuration

# Compile, Assembler, Linker

# Software Development Tools Overview





# Tools Descriptions

## ► Assembler

- Translates Assembly Language Source Files Into Machine Language Object modules

## ► C/C++ compiler

- produces ARM machine code object modules

## ► Linker

- Combines object files into a single executable object module

# Create Linux Library



# Linux Library

## ► Static Libraries

- statically aware

## ► Dynamically Linked "Shared Object" Libraries

- Dynamically linked at run time



# Static Libraries

➤ static\_lib\_name.a

➤ Create static library with **ar**

- **ar --help**
- **ar -cvq libctest.a test1.o test2.o**

➤ Compile

- **gcc -o test main.c libctest.a**
- **gcc -o test main.c -L/path/to/library-directory -lctest**



# Dynamically Linked "Shared Object" Libraries

➤ Dynamic\_lib\_name.so

➤ Create share library

- gcc -fPIC -c filelist
- gcc -shared -Wl,-soname,soname -o libname filelist
- gcc -fPIC -c test1.c
- gcc -fPIC -c test2.c
- ln -s libctest.so.1.0 libctest.so.1
- ln -s libctest.so.1 libctest.so

➤ gcc -o test main.c -L/library\_PATH/ -lctest

➤ export LD\_LIBRARY\_PATH=LIB\_PATH:\$LD\_LIBRARY\_PATH

➤ ./test



# Dynamically Linked "Shared Object" Libraries

➤ ldconfig

➤ configure dynamic linker run-time bindings

➤ /etc/ld.so.conf

- 1. \$ vim /etc/ld.so.conf
  - and add LIB in path /usr/local
- 2. #ldconfig /usr/local/
  - /etc/ld.so.cache



# What and Need soname ?

|                  |                 |
|------------------|-----------------|
| <b>Real-name</b> | libctest.so.1.0 |
| <b>Soname</b>    | libctest.so.1   |
| <b>Linkname</b>  | libctest.so     |

→ libctest.so.1.0  
→ libctest.so.1

## Modify

|                  |                 |
|------------------|-----------------|
| <b>Real-name</b> | libctest.so.1.1 |
| <b>Soname</b>    | libctest.so.1   |
| <b>Linkname</b>  | libctest.so     |

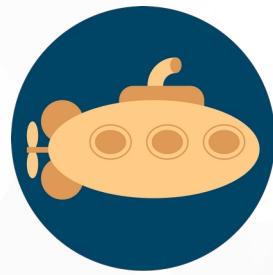
→ libctest.so.1.1  
→ libctest.so.1

|                  |                 |
|------------------|-----------------|
| <b>Real-name</b> | libctest.so.1.5 |
| <b>Soname</b>    | libctest.so.1   |
| <b>Linkname</b>  | libctest.so     |

→ libctest.so.1.5  
→ libctest.so.1

main.c no need to re-compile

```
gcc -o test main.c -L/library_PATH/ -lctest
```



U-Boot

# U-boot



# Bootloader

- What is bootloader
- Boot : short bootstrap

- Initialize basic of SOC (CPU, RAM, CLK)
- BL1, BL2

- Loader

- Load OS to RAM(DDR) form storage
- u-boot



# Different Embedded Linux bootloader

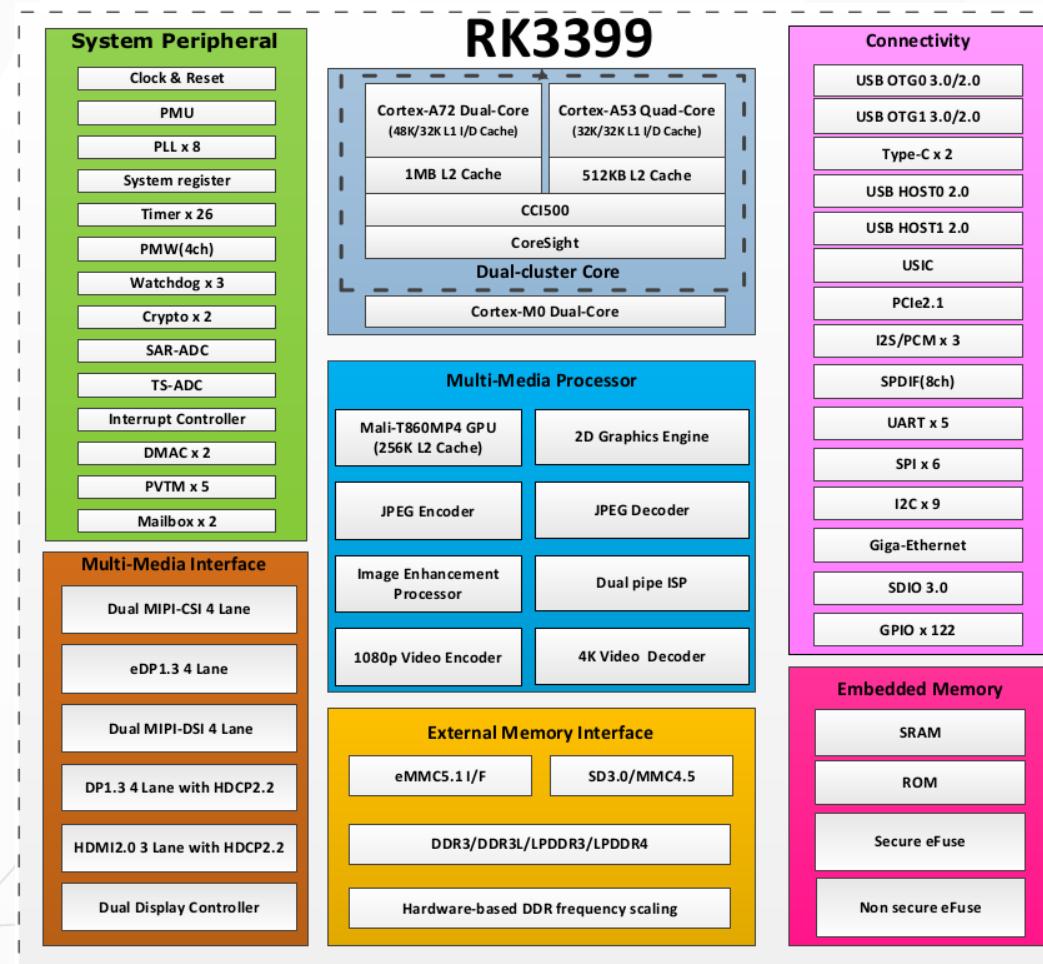
- U-boot
- UEFI
- Redboot
- Stubby (Linaro) ...
- Anyway, they are same target
  - Load and boot OS to RAM from storage



# Concepts of the Boot Loader

- Boot Loader is varied from CPU to CPU, from board to board.
- All the system software and data are stored in some kind of nonvolatile memory.
- Operation Mode of Boot Loader
  - **Boot : Initialize basic of SOC**
  - **Load : load OS to RAM then execute**

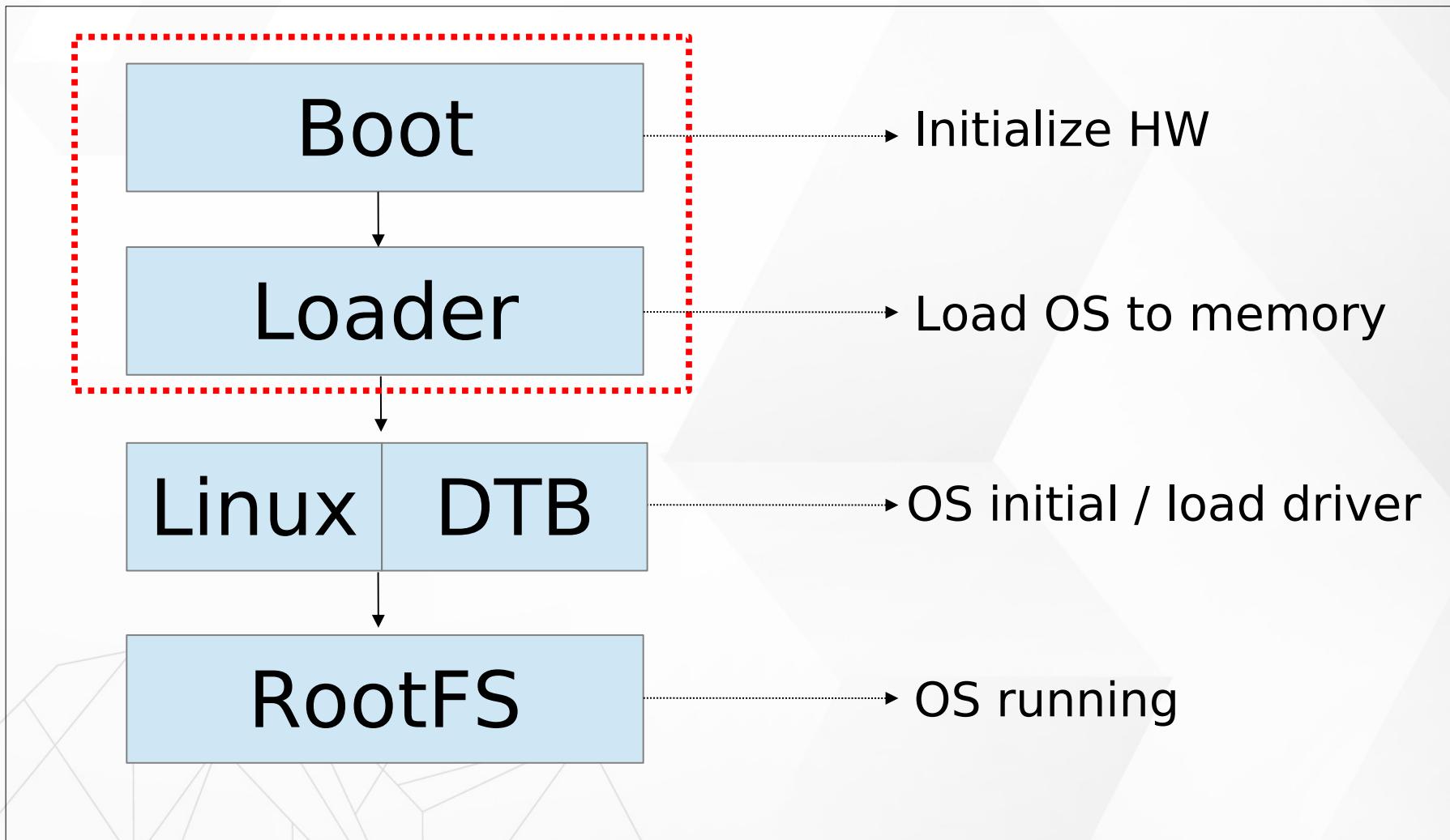
# RK3399 SOC



[http://wiki.friendlyarm.com/wiki/index.php/NanoPi\\_M4#Diagram.2C\\_Layout\\_and\\_Dimension](http://wiki.friendlyarm.com/wiki/index.php/NanoPi_M4#Diagram.2C_Layout_and_Dimension)



# Embedded Linux System Booting





# RockPi4 Image Partition

partmap.txt – Image layout in SD card

|              |               |                     |
|--------------|---------------|---------------------|
| Loader       | idbloader.img | 0x8000,0x280000     |
| u-boot       | Uboot.img     | 0x800000,0x0400000  |
| Trust        | Trust.img     | 0xC00000,0x0400000  |
| Linux kernel | Linux kernel  | SD Card Part List 4 |
| RootFS       | rootfs.img    | SD Card Part List 5 |



# Boot

▶ Power On BootROM code (work in cache)

- Load BL1

▶ BL1 (work in cache)

- Initial simple exception vectors, PLL (clock)
- Initial Multi-CPU
- Load BL2

▶ BL2 (work in cache)

- Initial DDR memory
- Initial C environment (stack, heap, )
- Load BL31



# Boot

## ▶ BL31 (work in DDR)

- Initial exception vectors
- Load BL32 (u-boot)

## ▶ BL32 U-boot

- Initial storage device
- Load Linux Kernel

## ▶ Kernel

- kernel/Documentation/arm64/booting.txt
- Mount RootFS

# Introduce U-boot



# U-boot

- Das U-Boot -- the Universal Boot Loader
- <http://www.denx.de/wiki/U-Boot>
- GitHub for u-boot
- Open Source follow GPL
- Supply many CPU
  - PPC, ARM, x86, MIPS, AVR32 ...
- Supply basic periphery devices
  - UART, Flash, SD/MMC ....



# u-boot directory structure

- Arch → Many types CPU : Arm, mips, i386 ...
- Board → Many types develop board : Samsung, ti, davinci ...
- Tools → Make Image (u-boot, linux ) or S-Record image tool
- Drivers → Some HW control code
- FS → Supply file system : fat, jffs2, ext2, cramfs
- Disk → Supply disk driver and partition handling
- Common → Major command and relation environment setting source code



# u-boot directory structure about rk3399

## ➤ **arch/arm/cpu/armv8/**

- ARMv8 relate

## ➤ **arch/arm/cpu/armv8/rk33xx/**

- rk3399 related
- Clock, i2c, irom, mmc, emmc ...

## ➤ **board/rockchip/rk33xx/**

- Board level related
- Peripheral initial

# Rk3399 Configure File



## ▶ Cmd/

- ▶ u-boot command related

## ▶ Configures/

### ▶ High Level Define

- rock-pi-4b-rk3399\_defconfig

## ▶ include/configs/

### ▶ Low Level Define

- config\_distro\_bootcmd.h
- evb\_rk3399.h
- rk3399\_common.h



# U-boot Configure (1)

## ➤ Change u-boot about

- Add Command
- Modify boot parameter
- Add Function



# U-boot Configure (2)

## ▶ Edit Configure

### ▶ Method 1

- menuconfig
  - [cmd] make menuconfig

### ▶ Method 2

- Edit configure file
  - config\_distro\_bootcmd.h
  - evb\_rk3399.h
  - rk3399\_common.h

# U-boot Common Function



# Help

➤ help

→ print command description/usage

➤ \$ help

→ help mm

➤ \$ ?

=> help mm

mm - memory modify (auto-incrementing address)

Usage:

mm [.b, .w, .l, .q] address

=>



# md

➤ md

→ memory display

→ md [.b, .w, .l, .q] address [# of objects]

```
=> md 0x02080000
02080000: 4e04e260 e461206c 0808a115 2a666646 `...Nl a.....Fff*
02080010: 88689ca1 224002e2 2e000a62 20a26262 ...h...@"b...bb.
02080020: a8207386 60a626e4 00016006 62a40642 .s ..&..`...B..b
02080030: e000a239 62476067 a3284802 24e66242 9...g`Gb.H(.Bb.$
02080040: 22300806 32a0c270 41620081 62042664 ..0"p..2..bAd&.b
```



# mw



→ memory write

→ mw [.b, .w, .l, .q] address value [count]

```
=> mw.l 0x02080000 0x12345678 1
=> md.l 0x02080000
02080000: 12345678 e461206c 0808a115 2a666646 xV4.l a.....Fff*
02080010: 88689ca1 224002e2 2e000a62 20a26262 ..h....@"b...bb.
02080020: a8207386 60a626e4 00016006 62a40642 .s ..&..`..B..b
02080030: e000a239 62476067 a3284802 24e66242 9...g`Gb.H(.Bb.$
02080040: 22300806 32a0c270 41620081 62042664 ..0"p...2..bAd&.b
```



# mmc

➤ mmc list

→ lists available devices

```
=> mmc list
mmc@fe310000: 2
mmc@fe320000: 1 (SD)
sdhci@fe330000: 0
```



# mmc

## ➤ mmc info

→ display info of the current MMC device

```
=> mmc dev 1
switch to partitions #0, OK
mmc1 is current device
=> mmcinfo
Device: mmc@fe320000
Manufacturer ID: 3
OEM: 5344
Name: SU04G
Bus Speed: 50000000
Mode: SD High Speed (50MHz)
Rd Block Len: 512
SD version 3.0
High Capacity: Yes
Capacity: 3.7 GiB
Bus Width: 4-bit
Erase Group Size: 512 Bytes
```



# mmc

## ➤ mmc part

→ lists available partition on current mmc device

```
=> mmc part

Partition Map for MMC device 1 -- Partition Type: DOS

Part Start Sector Num Sectors UUID Type
 1_ 196608 7547904 97ddff01-01 83
```



# mmc

## ➤ mmc dev

- show or set current mmc device [partition]
- mmc dev [dev] [part]

```
=> mmc dev 1
switch to partitions #0, OK
mmc1 is current device
=> mmcinfo
Device: mmc@fe320000
Manufacturer ID: 3
OEM: 5344
Name: SU04G
Bus Speed: 50000000
Mode: SD High Speed (50MHz)
Rd Block Len: 512
SD version 3.0
High Capacity: Yes
Capacity: 3.7 GiB
Bus Width: 4-bit
Erase Group Size: 512 Bytes
```



# FAT List

## fatls

- list files in a directory
- fatls <interface> [<dev[:part]>] [directory]
  - list files from 'dev' on 'interface' in a 'directory'

```
=> fatls mmc 1:4
 155639 config-4.4.154-113-rockchip-gdb9dfc2cdd25
 20371464 vmlinuz-4.4.154-113-rockchip-gdb9dfc2cdd25
 extlinux/
 dtbs/
 overlays/
 0371464 vmlinuz-4.4.154
 1968 hw_intf.conf
 4786387 System.map-4.4.154-00039-g00fccd3
 156441 config-4.4.154
 4786387 System.map-4.4.154
 5038020 initrd.img-4.4.154
```



# FAT Load

## fatload

- fatload - load binary file from a FAT filesystem
- fatload <interface> [<dev[:part]> [addr [filename [bytes [pos]]]]]  
load binary file 'filename' from 'dev' on 'interface'  
to address 'addr' from fat filesystem

```
=> fatload mmc 1:4 ${kernel_addr_r} vmlinuz-4.4.154
reading vmlinuz-4.4.154
20371464 bytes read in 858 ms (22.6 MiB/s)
```

```
=> fatload mmc 1:4 ${fdt_addr_r} dtbs/4.4.154/rockchip/rockpi-4b-linux.dtb
reading dtbs/4.4.154/rockchip/rockpi-4b-linux.dtb
94603 bytes read in 15 ms (6 MiB/s)
```

```
=> fatload mmc 1:4 ${ramdisk_addr_r} initrd.img-4.4.154
reading initrd.img-4.4.154
5038020 bytes read in 215 ms (22.3 MiB/s)
```



# printenv

## ➤ printenv

- print environment variables
- printenv name

```
=> printenv
altbootcmd=setenv boot_syslinux_conf extlinux/extlinux-rollback.conf;run distro_bootcmd
arch=arm
baudrate=1500000
board=evb_rk3399
board_name=evb_rk3399
boot_a_script=load ${devtype} ${devnum}:${distro_bootpart} ${scriptaddr} ${prefix}${script}; so
boot_efi_binary=load ${devtype} ${devnum}:${distro_bootpart} ${kernel_addr_r} efi/boot/bootaa64
ernel_addr_r} ${fdt_addr_r};else bootefi ${kernel_addr_r} ${fdtcontroladdr};fi
boot efi bootmar;if fdt addr ${fdt_addr_r}: then bootefi bootmar ${fdt_addr_r}:else bootefi boo

=> printenv loadimage
loadimage=ext4load mmc 1:1 ${kernel_addr_r} boot/Image
```



# setenv

➤ saveenv

➤ setenv

- set environment variables
- setenv name value

```
=> setenv test 12345
=> printenv test
test=12345
=> setenv test
=> printenv test
Error: "test" not defined
```



# Simple Script

LED Blank

Script

```
1=> while true; do; gpio toggle 125; sleep 1; done
2gpio: pin 125 (gpio 125) value is 0
3gpio: pin 125 (gpio 125) value is 1
4gpio: pin 125 (gpio 125) value is 0
5gpio: pin 125 (gpio 125) value is 1
6gpio: pin 125 (gpio 125) value is 0
7gpio: pin 125 (gpio 125) value is 1
```



# run

## run

- run commands in an environment variable
- run var [...]

```
=> run bootcmd
switch to partitions #0, OK
mmc1 is current device
Scanning mmc 1:4...
Found /extlinux/extlinux.conf
pxefile_addr_str = 0x00500000
bootfile = /extlinux/extlinux.conf
Retrieving file: /extlinux/extlinux.conf
reading /extlinux/extlinux.conf
1646 bytes read in 5 ms (321.3 KiB/s)
select kernel
1: kernel-4.4.154
2: kernel-4.4.154-999-rockchip-gcfa47f25e
3: kernel-4.4.154-113-rockchip-gdb9dfc2cdd25
4: kernel-4.4.154-00039-g00fccd3
5: kernel-4.4.154
Enter choice: Retrieving file: /hw_intfc.conf
reading /hw_intfc.conf
1968 bytes read in 4 ms (480.5 KiB/s)
```

command



# booti

## booti

- booti - boot Linux kernel 'Image' format from memory
- booti [addr [initrd[:size]] [fdt]]

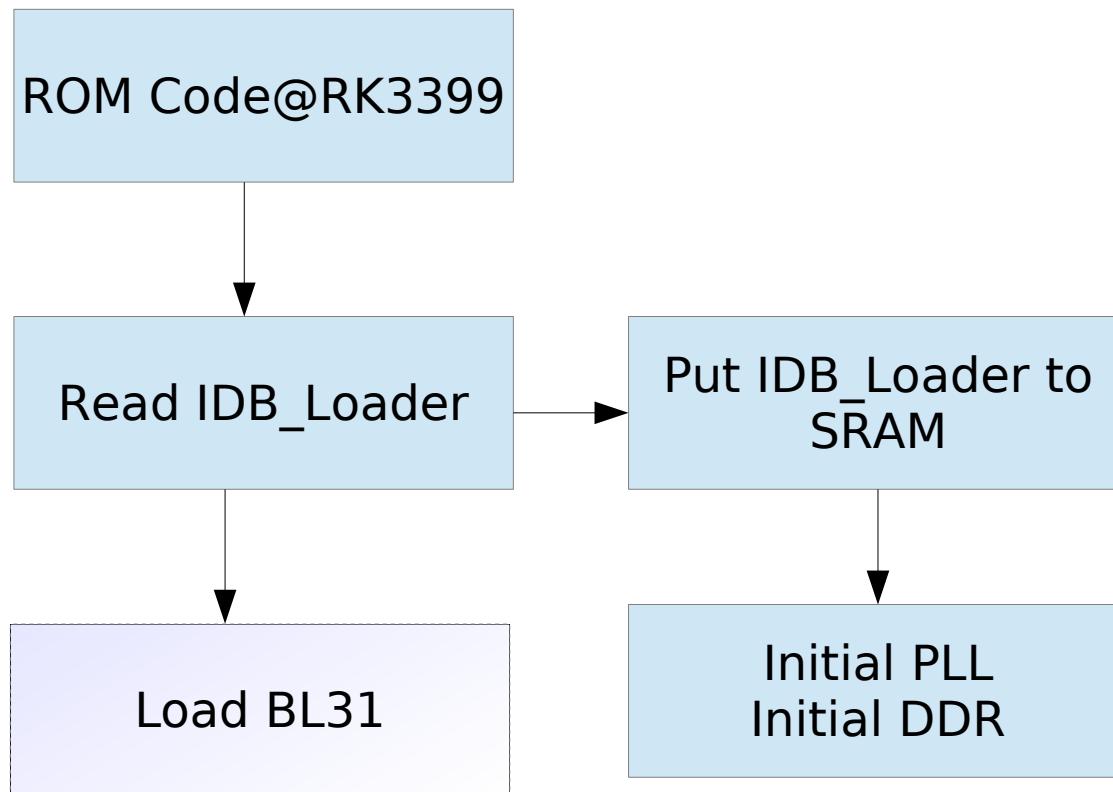
```
=> booti ${kernel_addr_r} ${ramdisk_addr_r}:5038020 ${fdt_addr_r}
Flattened Device Tree blob at 01f00000
Booting using the fdt blob at 0x1f00000
Loading Ramdisk to 76db8000, end 7bdf0020 ... OK
Loading Device Tree to 0000000076d9d000, end 0000000076db718a ... OK
Adding bank: start=0x00200000, size=0x7fe00000

Starting kernel ...

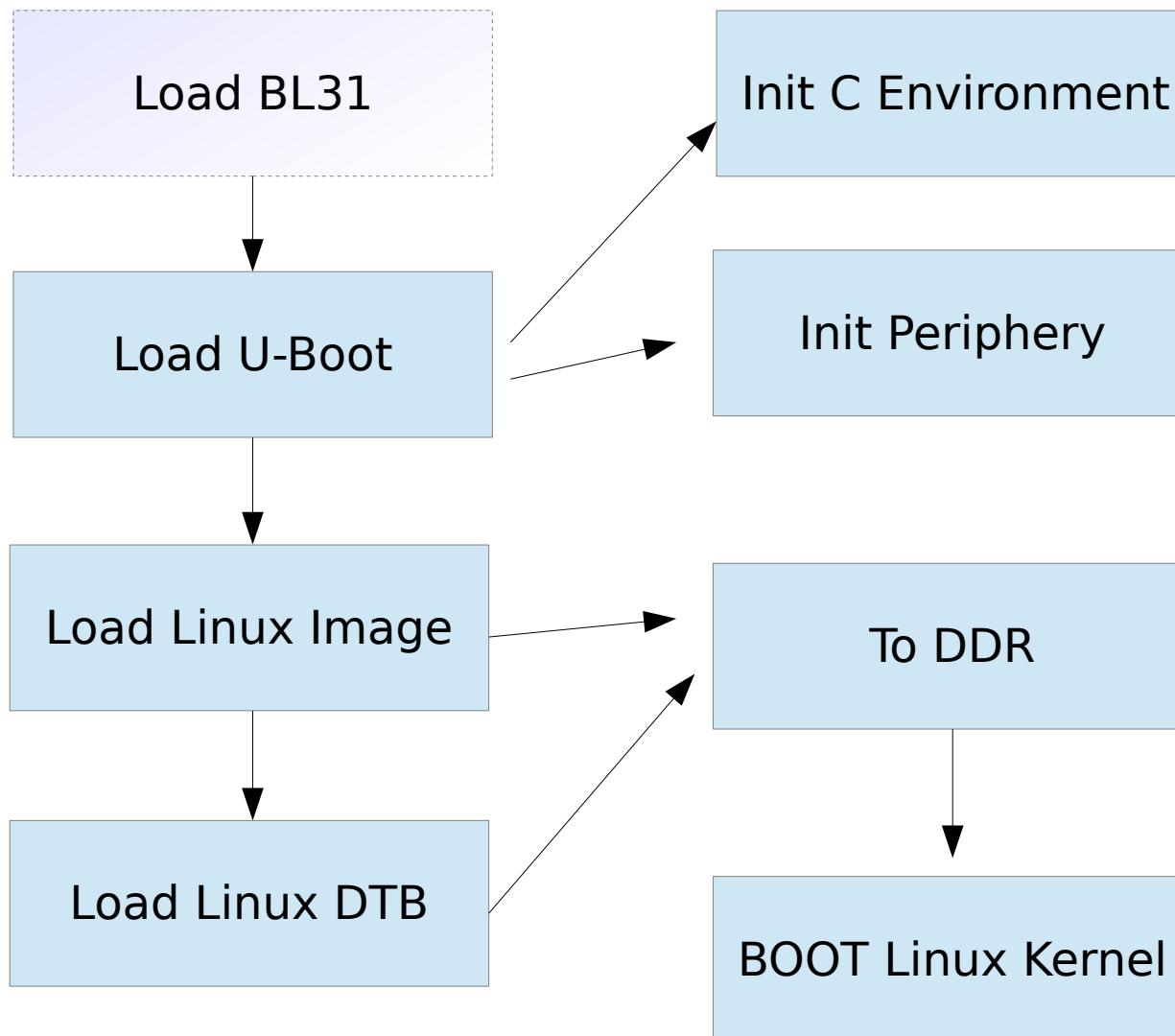
[0.000000] Booting Linux on physical CPU 0x0
[0.000000] Initializing cgroup subsys cpuset
[0.000000] Initializing cgroup subsys cpu
[0.000000] Initializing cgroup subsys cpuacct
[0.000000] Linux version 4.4.154 (slash@slash-ThinkPad-E14-Gen-2) (gcc version 7.3.1 20180425 [linaro-7.3-2018.05
revision d29120a424ecfbc167ef90065c0eeb7f91977701] (Linaro GCC 7.3-2018.05)) #4 SMP Sun Jan 23 15:55:32 CST 2022
```

# Boot Linux kernel

# System Start Up



# System Start Up





# Boot Linux Kernel Command

## ▶ bootcmd

- Power On will **auto run content of bootcmd**

```
[CMD] printenv bootcmd
```

```
bootcmd=run distro_bootcmd;boot_android ${devtype} ${devnum};bootrkp;
```



# Boot OS (Linux) Command

► booti

→ boot Linux **kernel 64 bit standard kernel image**

► boota

→ boot **android style kernel image**

► Boot command

→ booti [Kernel image addr] [RAM disk adr ] [DTB addr]

→ \$ booti **kernel\_addr ramdisk\_addr DTB\_addr**



# Linux Kernel Command (bootargs)

## ▶ bootargs

- \$ printenv bootargs (Linux kernel command)

```
earlyprintk console=ttyFIQ0,1500000n8 rw init=/sbin/init
rootfstype=ext4 rootwait root=UUID=a54358ce-55c2-4f4f-bf6d-
7106997cfb8f console=ttyS2,1500000n8
```

- earlycon : early console device
- console : Linux console device
- root : RootFS device

## ▶ Enter Linux kernel to check

- \$ cat /proc/cmd



# How to jump to kernel

## ➤ Use boot command

→ cmd/cmd\_booti.c

## ➤ Jump to Linux kernel

→ arch/arm/lib/bootm.c

→ boot\_os\_fn \*bootm\_os\_get\_boot\_func(int os)

→ int do\_bootm\_linux(int flag, int argc, char \*const argv[],  
bootm\_headers\_t \*images)

→ void (\*kernel\_entry)( void \*fdt\_addr,  
void \*res0,  
void \*res1,  
void \*res2);

# Add Function to Board Setting File



# evb-rk3399.c

- » board/rockchip/evb\_rk3399/evb-rk3399.c
- » Add function to here

```
diff --git a/board/rockchip/evb_rk3399/evb-rk3399.c b/board/rockchip/evb_rk3399/evb-rk3399.c
index 8e3fce4aa4..e5b4041288 100644
--- a/board/rockchip/evb_rk3399/evb-rk3399.c
+++ b/board/rockchip/evb_rk3399/evb-rk3399.c
@@ -30,6 +30,11 @@ int rk_board_init(void)
 struct udevice *pinctrl, *regulator;
 int ret;

+ printf("%s\n", __func__);
+
+ /* user2 led power-off */
+ run_command("gpio clear 125", 0);
+
+ /*
+ * The PWM does not have decicated interrupt number in dts and can
+ * not get periph_id by pinctrl framework, so let's init them here.
```

# New a Command



# Add Feature

- Add command → cmd/
- Add driver → driver/
- Add application → example/



# Add Command

➤ How to create a command ?

➤ Directory

- cmd/ → booti.c , mmc.c, mem.c ...

➤ U\_BOOT\_CMD(name,maxargs,rep,cmd,usage,help)

- include/command.h



# How to Command

## cmd/cmd\_version.c

```
static int do_version(struct cmd_tbl *cmdtp, int flag, int argc,
 char *const argv[])
{
 char buf[DISPLAY_OPTIONS_BANNER_LENGTH];

 printf("hellow_wold\n");

 printf(display_options_get_banner(false, buf, sizeof(buf)));

 return 0;
}

U_BOOT_CMD(
 version, 1, 1, do_version,
 "print monitor, compiler and linker version",
 ""
);
```



## Include/command.h

```
#define U_BOOT_CMD(_name, _maxargs, _rep, _cmd, _usage, _help)
```



# Hello World Command

```
Command line interface
Use the menu. <Enter> selects submenus ---> (or empty submenus ----). | indicates, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit
in [] excluded <M> module < > module capable

[*] Support U-Boot commands
[*] Command Hello World Sample
 -*- Use hush shell
 (=) Shell prompt
 Autoboot options --->
 *** FASTBOOT ***
[*] Fastboot support --->
 *** Commands ***
 Info commands --->
 Boot commands --->
 Environment commands --->
 Memory commands --->
 Compression commands --->
 Device access commands --->
 Shell scripting commands --->
 Network commands --->
[] Enable memtester for ddr
 Misc commands --->
+(+)
```



# Hello World Command

cmd/Makefile

```
obj-$(CONFIG_CMD_USB_MASS_STORAGE) += usb_mass_storage.o
obj-$(CONFIG_CMD_USB_SDP) += usb_gadget_sdp.o
obj-$(CONFIG_CMD_THOR_DOWNLOAD) += thordown.o
obj-$(CONFIG_CMD_XIMG) += ximg.o
obj-$(CONFIG_CMD_YAFFS2) += yaffs2.o
obj-$(CONFIG_CMD_SPL) += spl.o
obj-$(CONFIG_CMD_ZIP) += zip.o
obj-$(CONFIG_CMD_ZFS) += zfs.o

obj-$(CONFIG_CMD_DFU) += dfu.o
obj-$(CONFIG_CMD_GPT) += gpt.o
obj-$(CONFIG_CMD_ETHSW) += ethsw.o
obj-$(CONFIG_CMD_HELLOWORLD) += helloworld.o

Power
obj-$(CONFIG_CMD_PMIC) += pmic.o
obj-$(CONFIG_CMD_REGULATOR) += regulator.o

obj-$(CONFIG_CMD_BLOB) += blob.o
endif # !CONFIG_SPL_BUILD
```



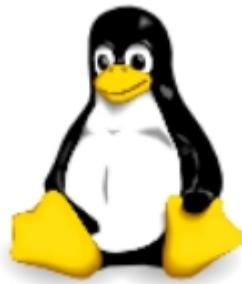
# Hello World Command

## cmd/Kconfig

```
menu "Command line interface"

config CMDLINE
 bool "Support U-Boot commands"
 default y
 help
 Enable U-Boot's command-line functions. This provides a means
 to enter commands into U-Boot for a wide variety of purposes. It
 also allows scripts (containing commands) to be executed.
 Various commands and command categorys can be individually enabled.
 Depending on the number of commands enabled, this can add
 substantially to the size of U-Boot.

config CMD_HELLOWORLD
 bool "Command Hello World Sample"
 depends on CMDLINE
 help
 This option enables the Hello World as command line
 Sample.
```



# Linux Kernel

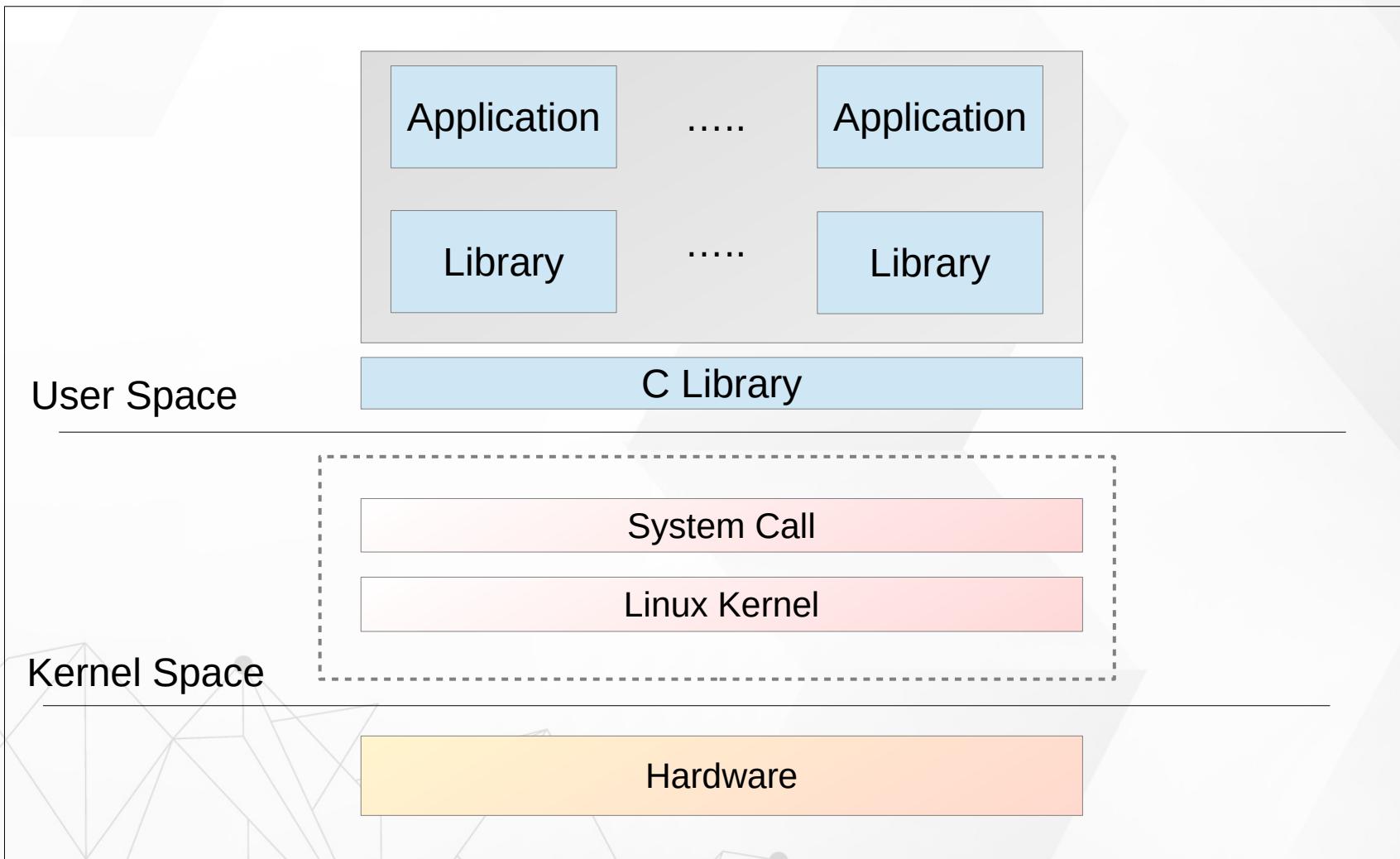


# Linux kernel key features

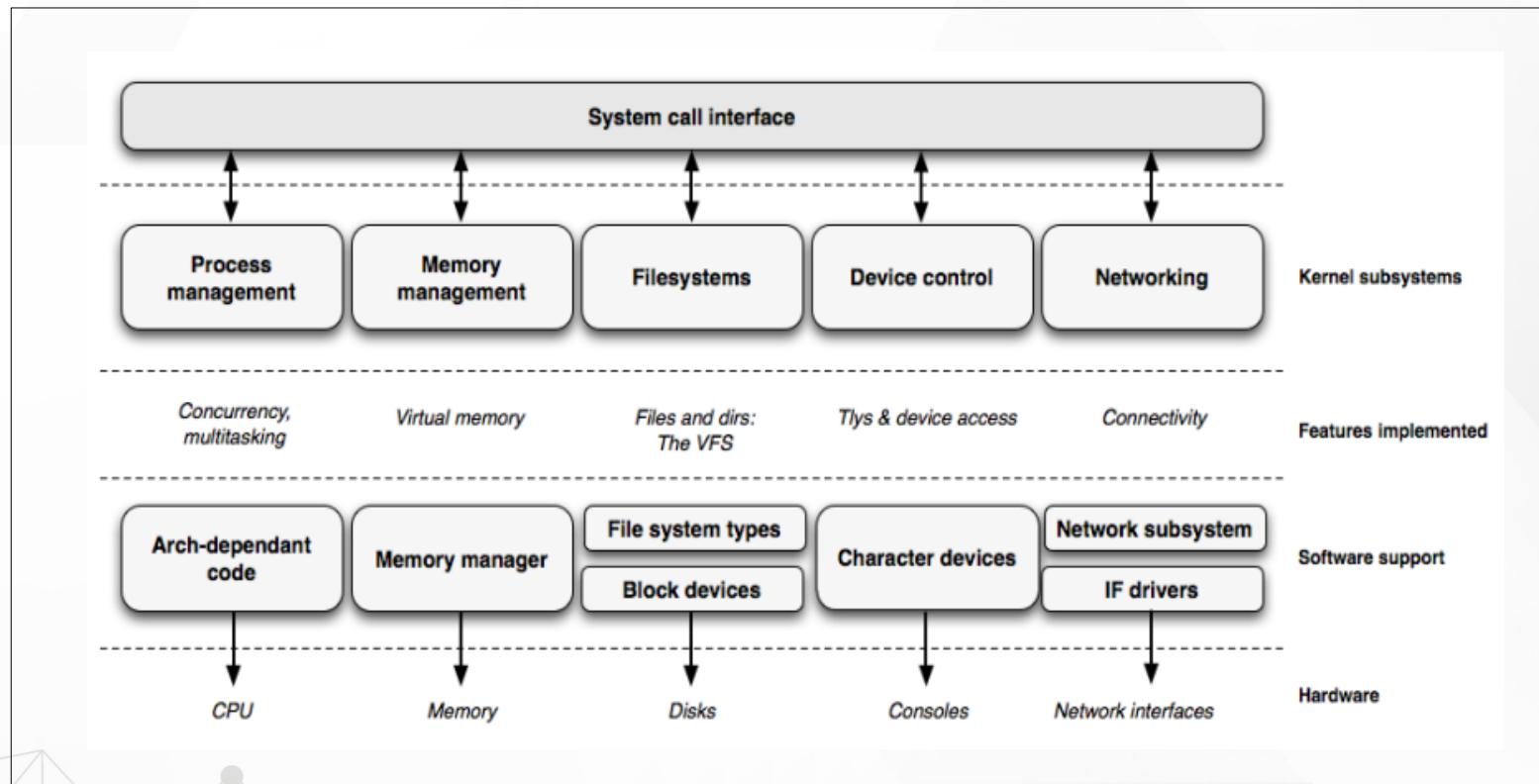
- ▶ Portability and hardware support
- ▶ Scalability
- ▶ Compliance to standards and interoperability
- ▶ Exhaustive networking support
- ▶ Stability and reliability
- ▶ Modularity
- ▶ Easy to program.



# Linux Kernel in the System



# Linux Kernel





# Kernel Source

- » <https://www.kernel.org/>
- » Many chip vendors
- » kernel sub-communities
  - Architecture communities
    - ARM, MIPS, PowerPC ...
  - device drivers communities
    - I2C, SPI, USB, PCI, network ...



# Programming language

- » Implemented in C like all Unix systems
- » A little Assembly is used too
- » **No C++ used**
- » **No C library**
- » **No floating point** computation
- » Kernel code has to supply its own library implementations
  - X : printf(), memset(), malloc(),...
  - O: printk(), memset(), kmalloc()



# Linux Sources Important Folder

## ▶ Kernel Image

- arch/<ARCH>/boot/
- Arch/arm64/boot

## ▶ DTS

- arch/<ARCH>/boot/dts
- Arch/arm64/boot/dts/rockchip/

## ▶ driver/

## ▶ Documentation/

# Kernel Basic Command



# Basic Build Command

» \$ make

- Build all

» \$ make dtbs

- Build Device-tree only

» \$ make modules

- Build kernel modules only

» \$ make modules\_install

- Install all modules to folder



# Basic Build Command

- ▶ [CMD] make deb-pkg
  - ▶ Build both source and binary deb kernel packages
- ▶ [CMD] make modules\_install
  - ▶ Install all modules to INSTALL\_MOD\_PATH
- ▶ [CMD] make mrproper
  - ▶ Remove all generated files (.config)
- ▶ [CMD] make clean
- ▶ [CMD] make distclean
  - ▶ Remove editor backup and patch reject files

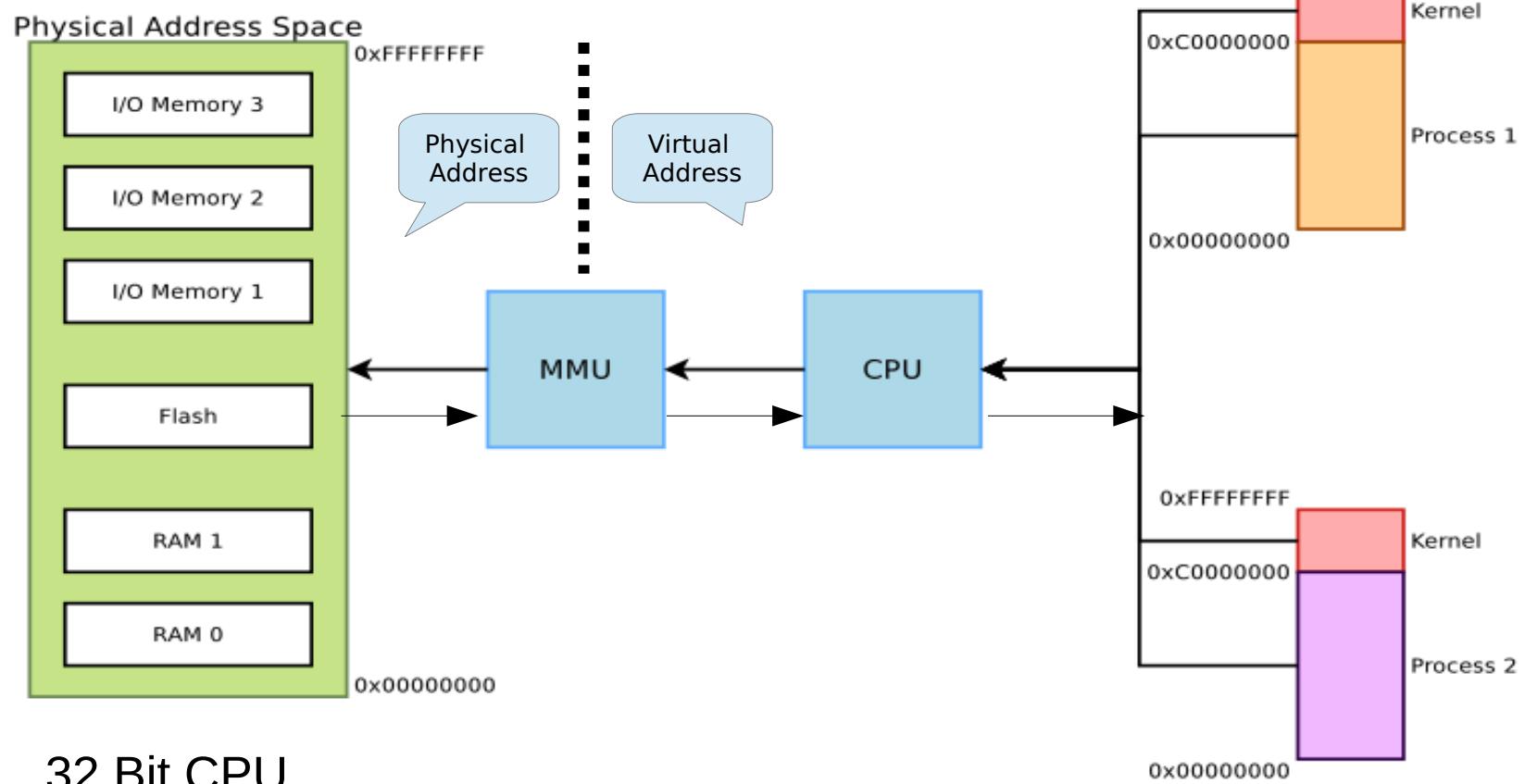
# Make Help

[CMD] make help

```
acs5k_defconfig - Build for acs5k
acs5k_tiny_defconfig - Build for acs5k_tiny
afeb9260_defconfig - Build for afeb9260
ag5evm_defconfig - Build for ag5evm
am200epdkit_defconfig - Build for am200epdkit
ap4evb_defconfig - Build for ap4evb
armadillo800eva_defconfig - Build for armadillo800eva
assabet_defconfig - Build other generic targets:
at91_dt_defconfig - Bui all - Build all targets marked with [*]
at91rm9200_defconfig - Bui * vmlinux - Build the bare kernel
at91sam9260_defconfig - Bui * modules - Build all modules
at91sam9261_defconfig - Bui modules_install - Install all modules to INSTALL_MOD_PATH (default: /)
at91sam9263_defconfig - Bui firmware_install - Install all firmware to INSTALL_FW_PATH
(atdefault: $(INSTALL_MOD_PATH)/lib/firmware)
at91sam9g20_defconfig - Bui dir/ - Build all files in dir and below
at91sam9g45_defconfig - Bui dir/file.[o|S] - Build specified target only
at91sam9rl_defconfig - Bui dir/file.lst - Build specified mixed source/assembly target only
(atrequires a recent binutils and recent build (System.map))
at91x40_defconfig - Bui dir/file.ko - Build module including final link
badge4_defconfig - Bui modules_prepare - Set up for building external modules
bcmring_defconfig - Bui tags/TAGS - Generate tags file for editors
bonito_defconfig - Bui cscope - Generate cscope index
cam60_defconfig - Bui gtags - Generate GNU GLOBAL index
cerfcube_defconfig - Bui kernelrelease - Output the release version string
cm_x2xx_defconfig - Bui kernelversion - Output the version stored in Makefile
cm_x300_defconfig - Bui headers_install - Install sanitised kernel headers to INSTALL_HDR_PATH
(cndefault: /home/xlloss/work/tiny-4412/build/linux_3.5.0_tiny4412/usr)
colibri_pxa270_defconfig - Bui ...
colibri_pxa300_defconfig - Bui ...
collie_defconfig - Build for collie
corgi_defconfig - Build for corgi
cpu9260_defconfig - Build for cpu9260
cpu9n20_defconfig - Build for cpu9n20
```

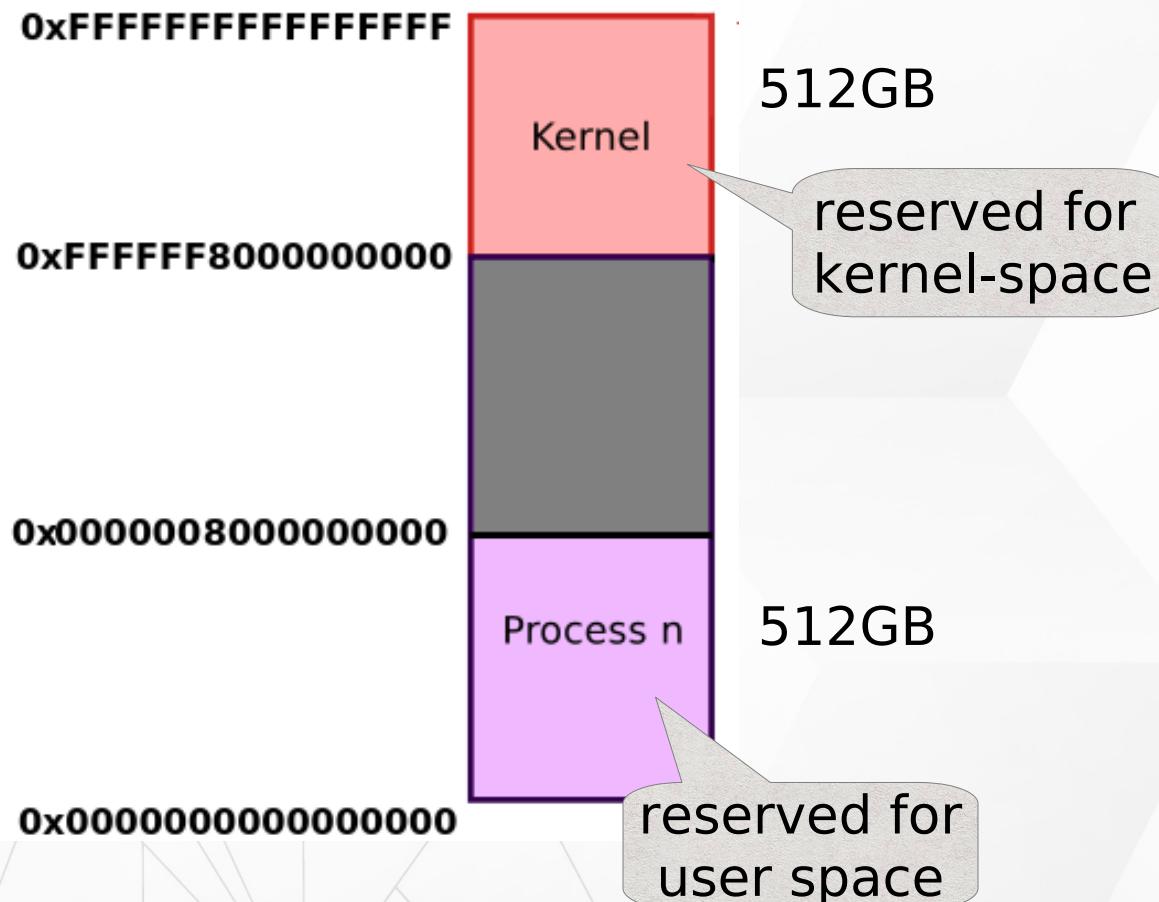
# Linux System Memory Layout

# Physical and Virtual Memory

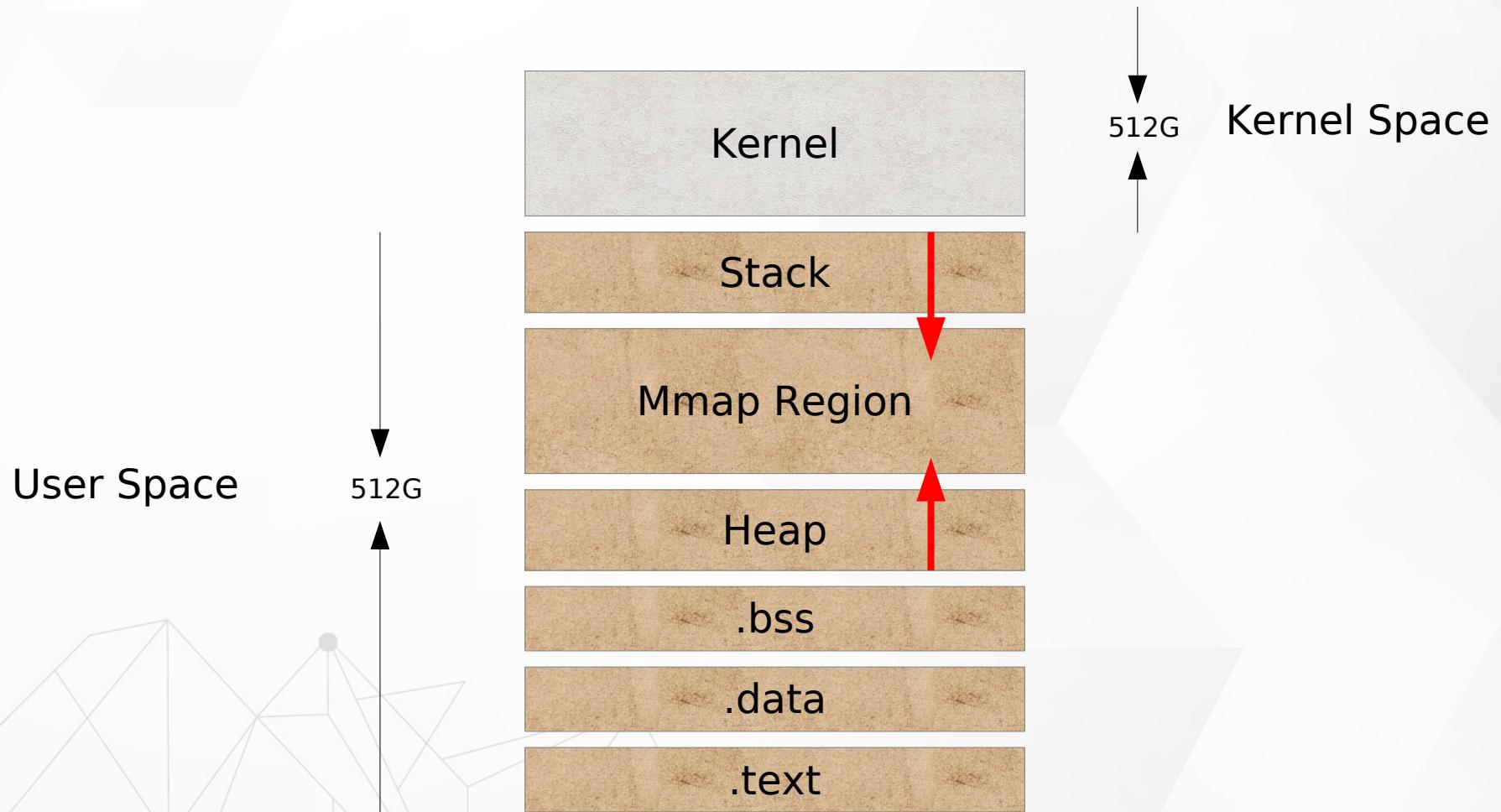


# Virtual Memory Organization

64 Bit CPU AArch64 Linux memory layout with 4KB pages + 3 levels

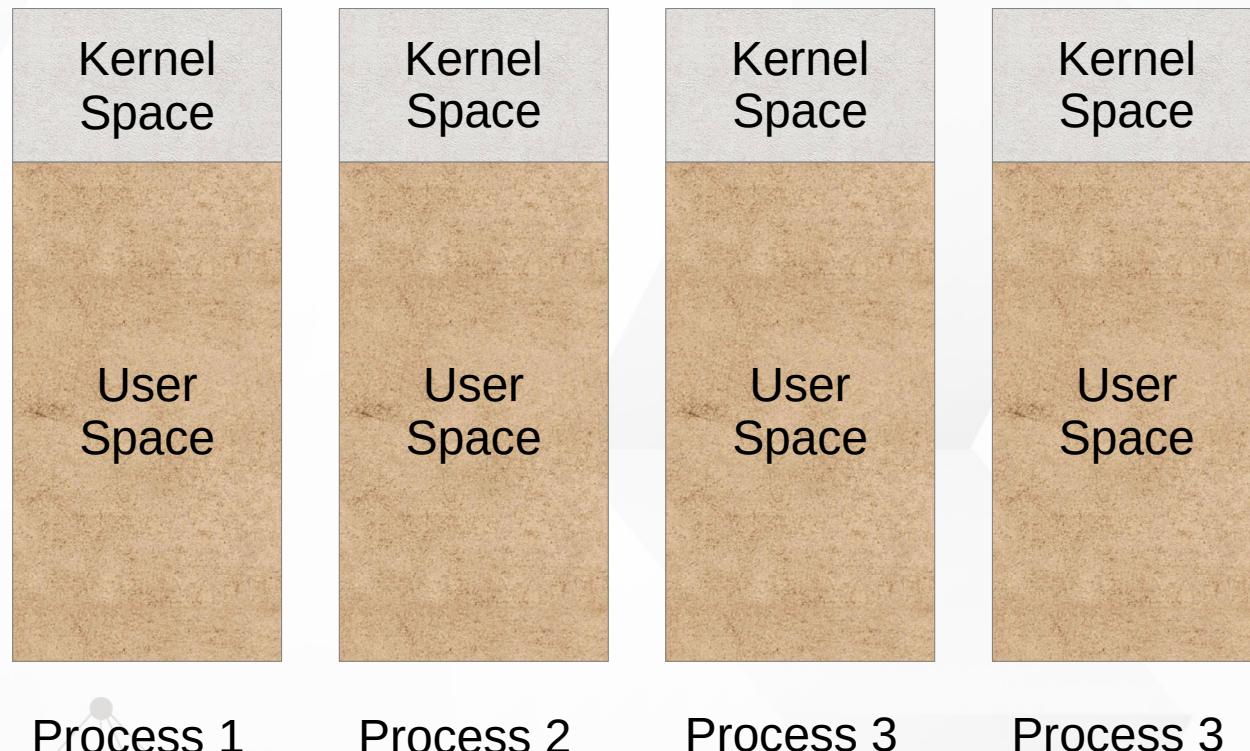


# Linux Process Memory





# Linux Process Memory



# How to Build Linux Kernel



# Specifying Cross-compilation

▶ **make ARCH=arm64 CROSS\_COMPILE=arm-linux- ...**

- [CDM] export ARCH=arm64
- [CMD] export CROSS\_COMPILE=aarch64-linux-gnu-

▶ Add above setting to script

- [CMD] source \$PATH/set\_toolchain.sh



# set\_toolchain.sh

```
export PATH=/home/cadtc/host_share_folder/toolchain/gcc-linaro-7.3.1-2018.05-x86_64_aarch64-linux-gnu/bin/
Toolchain path add to environment variable

export ARCH=arm64
Set SOC architecture type

export CROSS_COMPILE=aarch64-linux-gnu-
#Set compile prefix name

export KERNELDIR=/home/cadtc/rockchip-bsp/kernel
#Set Linux kernel source path
```



# Predefined Configuration Files

## ► Default configuration

→ arch/<arch>/configs/

## ► Use RockPi4 default Configure

→ [CMD] make rockchip\_linux\_defconfig

## ► To create your own default configuration file

→ [CMD] **make savedefconfig**, to create a minimal configuration file

→ [CMD] **mv defconfig** arch/arm64/configs/myown\_defconfig



# Kernel Compilation

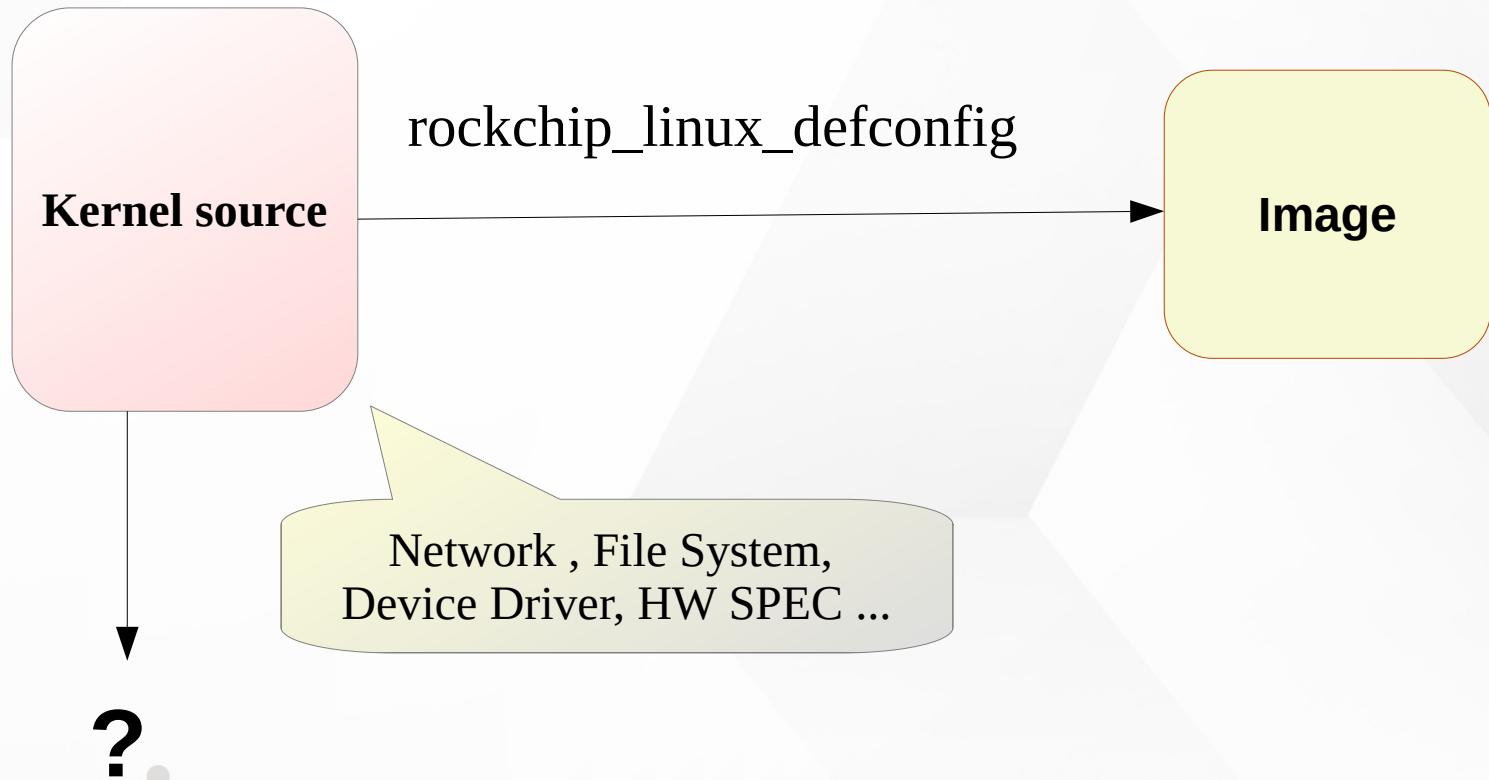
## ▶ Build kernel Image

- [CMD] make -j4
- To run multiple jobs in parallel if you have multiple CPU cores

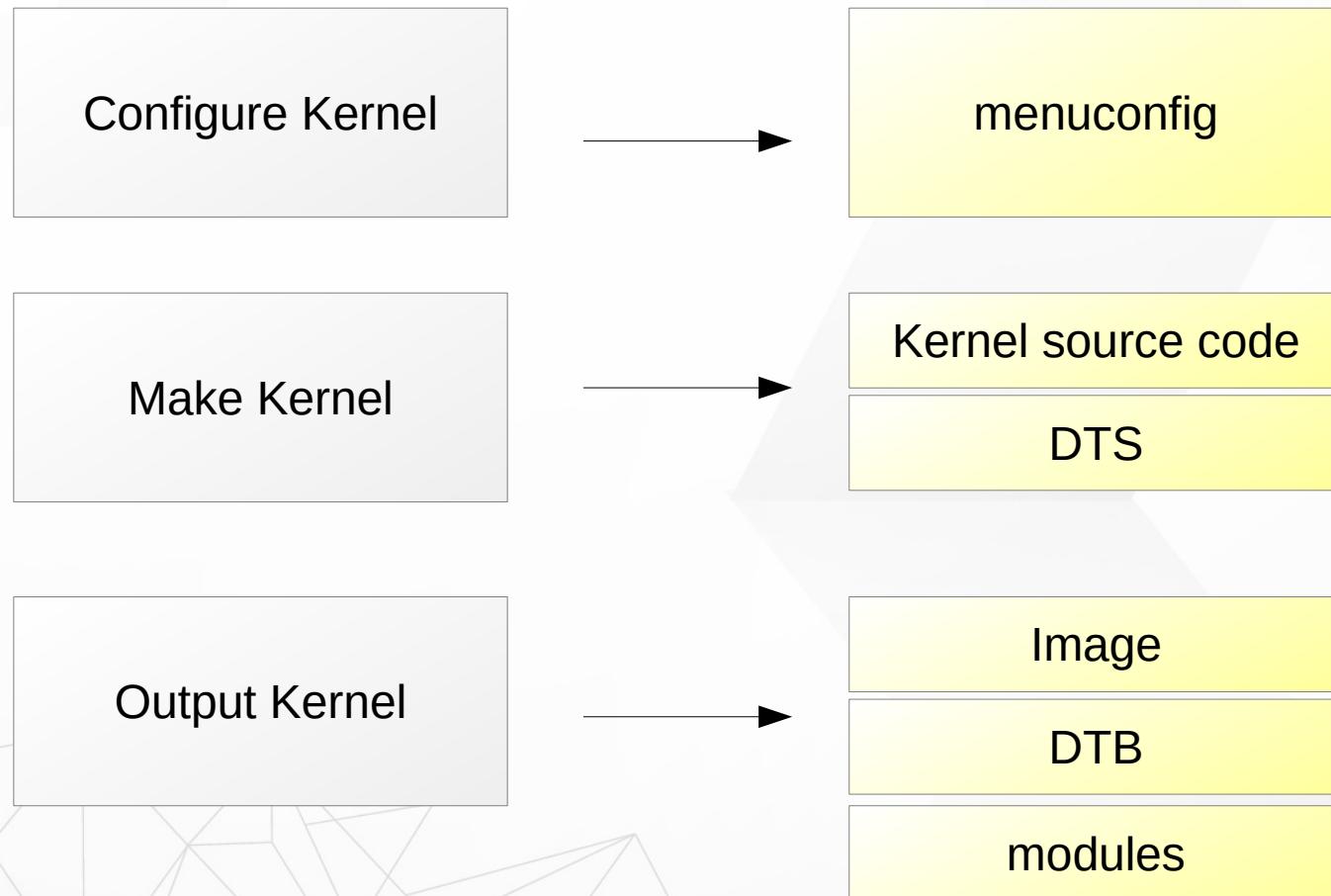
## ▶ Generates Image

- arch/arm64/boot/**Image**
  - **Image** for ARM64,
- arch/arm64/boot/dts/rockchip/
  - DTB : rockpi-4b-linux.dtb

# Kernel Configuration



# Kernel Configuration



# How to Select Feature in Kernel



# Kernel Configuration

- ▶ The kernel configuration and build system is based on multiple **Makefiles**
- ▶ The configuration is stored in the **.config** file at the root of kernel sources
- ▶ As options have dependencies, typically never edited by hand, but through graphical or text interfaces
  - [CMD] make menuconfig → Text
  - [CMD] make xconfig → graphical

# Kernel Configuration

```
4096 Apr 20 11:56 .
4096 Mar 31 08:42 ..
4096 Mar 31 08:41 android
4096 Mar 31 08:41 arch
419 Mar 31 08:41 backported-features
4096 Mar 31 08:41 block
459 Mar 31 08:41 build.config.cuttlefish.aarch64
457 Mar 31 08:41 build.config.cuttlefish.x86_64
296 Mar 31 08:41 build.config.goldfish.arm
303 Mar 31 08:41 build.config.goldfish.arm64
277 Mar 31 08:41 build.config.goldfish.mips
279 Mar 31 08:41 build.config.goldfish.mips64
298 Mar 31 08:41 build.config.goldfish.x86
303 Mar 31 08:41 build.config.goldfish.x86_64
4096 Mar 31 08:41 certs
9 Mar 31 08:41 .checkpatch.conf
154048 Apr 20 11:56 .config
```

.config

```
cuttlefish_defconfig
defconfig
lsk_defconfig
nanopi4_linux_defconfig
px30_linux_defconfig
px30_linux_robot_defconfig
ranchu64_defconfig
rk1808_linux_defconfig
rk1808_x4_linux_defconfig
rk3308_linux_defconfig
rk3326_linux_defconfig
rk3326_linux_robot_defconfig
rk3399pro_npu_defconfig
rk3399pro_npu_PCIE_defconfig
rockchip_cros_defconfig
rockchip_defconfig
rockchip_linux_defconfig
```

→ \${KERNEL}/arch/arm64/configs/  
rockchip\_linux\_defconfig



# Kernel or Module?

- ▶ The kernel image is a single file, resulting from the linking of all object files that correspond to features enabled in the configuration
- ▶ Some features (device drivers, file-system, etc.) can however be compiled as modules

# Menuconfig

# menuconfig

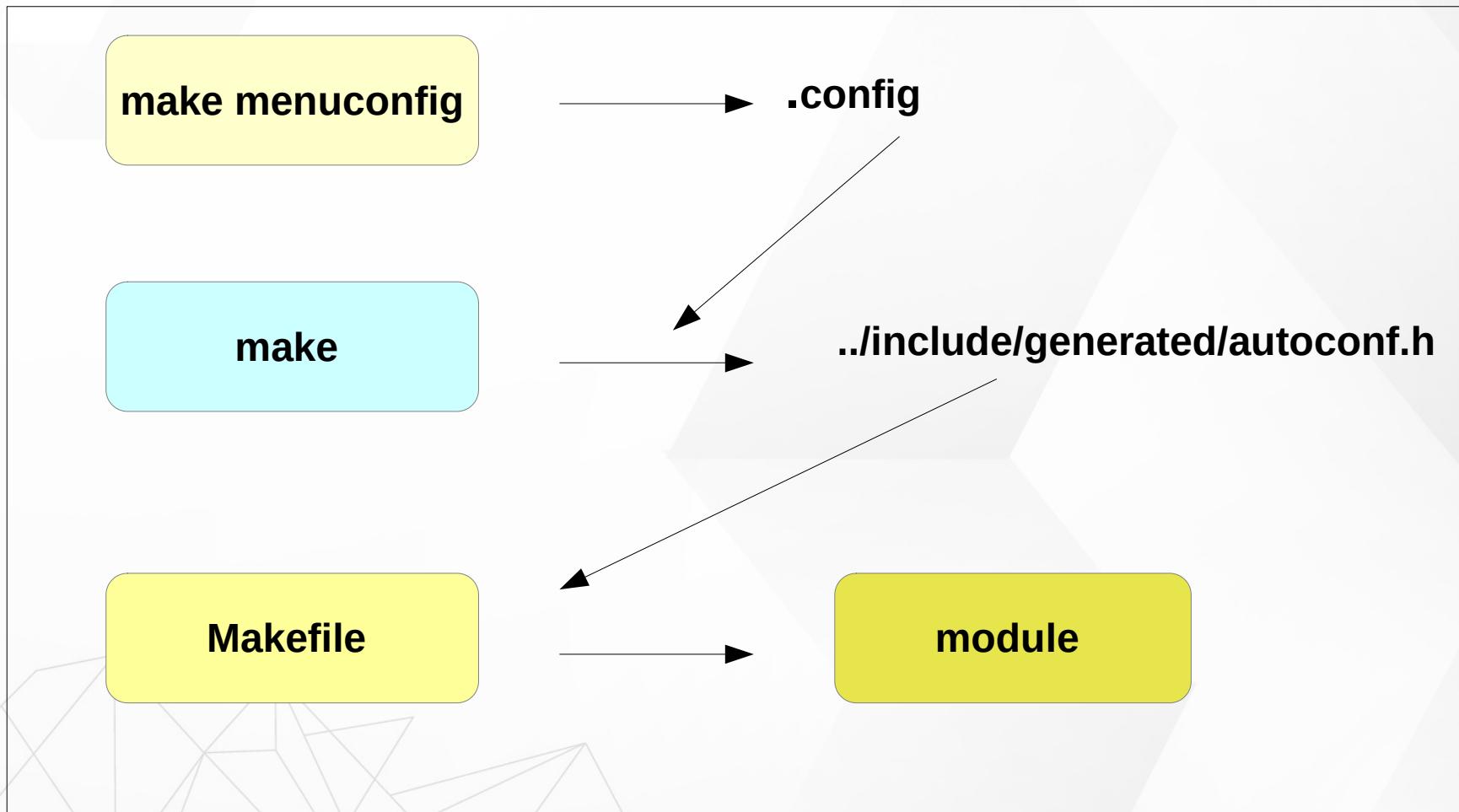
```
Linux/arm64 4.4.179 Kernel Configuration
menus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pres
</> for Search. Legend: [*] built-in [] excluded <M> module < > module

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

# Kernel Configuration Options

```
[] Enable AHB driver for NVIDIA Tegra SoCs
 Generic Driver Options --->
 Bus devices --->
 {M} Connector - unified userspace <-> kernelspace linker ----
 <-> Memory Technology Device (MTD) support ----
 -*- Device Tree and Open Firmware support --->
 <-> Parallel port support ----
 [*] Block devices --->
 <-> NVM Express block device
 Misc devices --->
 SCSI device support --->
 <-> Serial ATA and Parallel ATA drivers (libata) --->
 [*] Multiple devices driver support (RAID and LVM) --->
 <-> Generic Target Core Mod (TCM) and ConfigFS Infrastructure ----
 [] Fusion MPT device support ----
 IEEE 1394 (FireWire) support --->
 [*] Network device support --->
 [] Open-Channel SSD target support ----
 Input device support --->
 Character devices --->
 █ I2C support --->
 [*] SPI support --->
 <-> SPMI support ----
 <-> HSI support ----
 PPS support --->
 PTP clock support --->
 Pin controllers --->
 -*- GPIO Support --->
 <M> Dallas's 1-wire support --->
 -*- Power supply class support --->
 [*] Adaptive Voltage Scaling class support ---->
 <-> Hardware Monitoring support --->
 <-> Generic Thermal sysfs driver --->
 [*] Watchdog Timer Support --->
```

# Configuration



# Corresponding .config File Excerpt

```

I2C system bus drivers (mostly embedded / system-on-chip)

CONFIG_I2C_CADENCE is not set
CONFIG_I2C_CBUS_GPIO is not set
CONFIG_I2C_DESIGNWARE_PLATFORM is not set
CONFIG_I2C_DESIGNWARE_PCI is not set
CONFIG_I2C_EMEV2 is not set
CONFIG_I2C_GPIO=m
CONFIG_I2C_NOMADIK is not set
CONFIG_I2C_OCORES is not set
CONFIG_I2C_PCA_PLATFORM is not set
CONFIG_I2C_PXA_PCI is not set
CONFIG_I2C_RK3X=y
CONFIG_I2C_SIMTEC is not set
CONFIG_I2C_XILINX is not set
```

.config

`$(KERNEL_PATH)/drivers/i2c/buses/  
Makefile`

```
obj-$(CONFIG_I2C_PXA_PCI) += i2c-pxa-pci.o
obj-$(CONFIG_I2C_QUP) += i2c-qup.o
obj-$(CONFIG_I2C_RIIC) += i2c-riic.o
obj-$(CONFIG_I2C_RK3X) += i2c-rk3x.o
obj-$(CONFIG_I2C_S3C2410) += i2c-s3c2410.o
obj-$(CONFIG_I2C_SH7760) += i2c-sh7760.o
obj-$(CONFIG_I2C_SH_MOBILE) += i2c-sh_mobile.o
obj-$(CONFIG_I2C_SIMTEC) += i2c-simtec.o
obj-$(CONFIG_I2C_SIRF) += i2c-sirf.o
```



# Check CONFIG Setting

➤ In Kernel Source

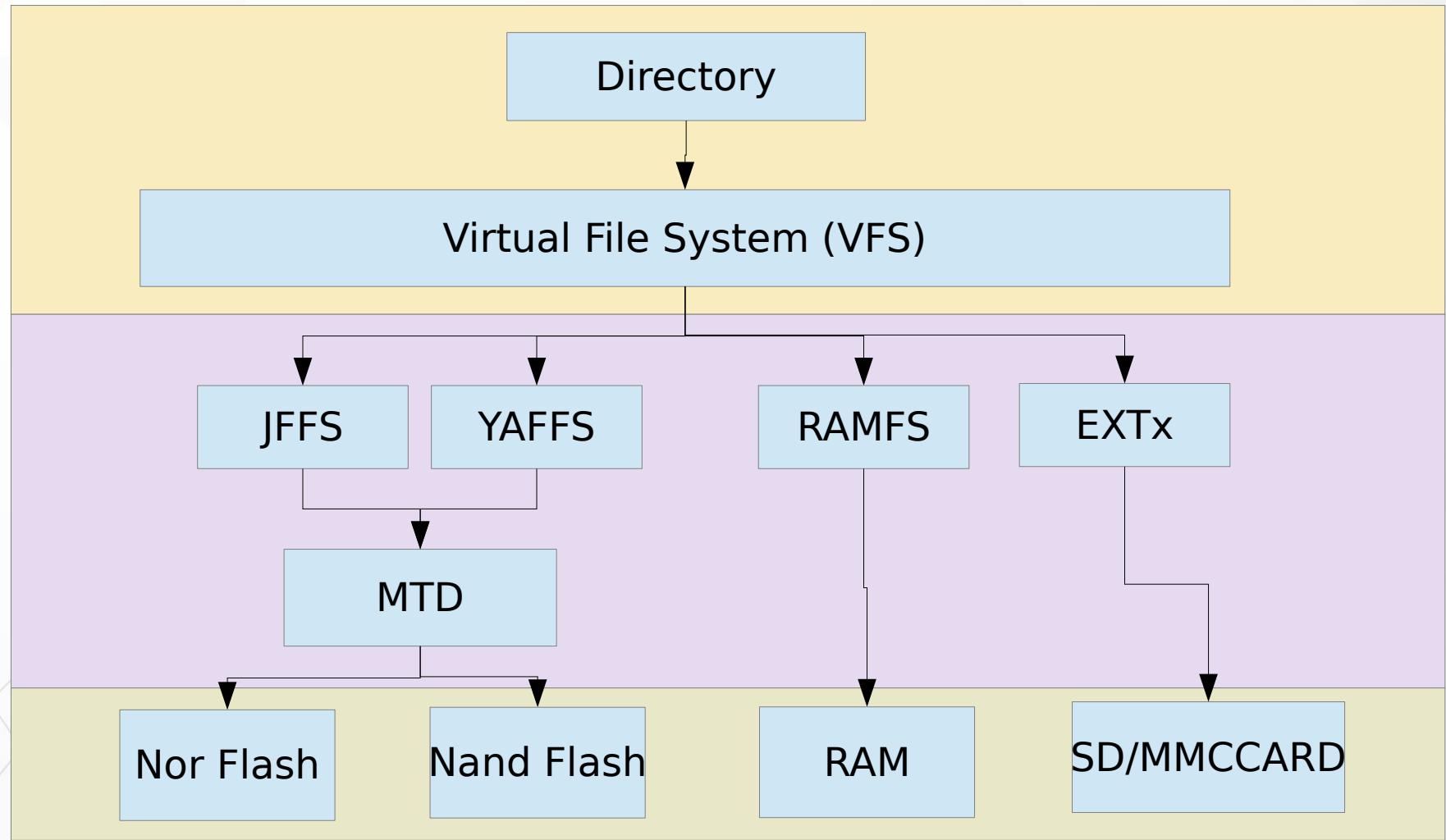
➤ [CMD] cat .config | grep \${CONFIG\_NAME}

➤ In Run Time Environment

➤ [CMD] cat /proc/config.gz | zcat | grep \${CONFIG\_NAME}

# CH 7 Root File System

# File System in Linux





# File System

- A file system defines how files are **named, stored, and retrieved** from a storage device
- For desktop users
  - FAT32, NFS, EXT3, EXT4
- For embedded system
  - Cramfs
  - JFFS2
  - Squashfs
  - YAFFS2
  - EXT2, EXT3



# File System and Kernel

## ► [CMD] make menuconfig

- File systems

```
[] Second extended fs support (NEW)
[] The Extended 3 (ext3) filesystem (NEW)
[] The Extended 4 (ext4) filesystem (NEW)
[] Reiserfs support (NEW)
[] JFS filesystem support (NEW)
[] XFS filesystem support (NEW)
[] GFS2 file system support (NEW)
[] Btrfs filesystem support (NEW)
[] NILFS2 file system support (NEW)
[] F2FS filesystem support (NEW)
[] Direct Access (DAX) support (NEW)
[] FS Encryption (Per-file encryption) (NEW)
[*] Dnotify support (NEW)
[*] Inotify support for userspace (NEW)
[] Filesystem wide access notification (NEW)
[] Quota support (NEW)
[] Kernel automounter version 4 support (also supports v3) (NEW)
[] FUSE (Filesystem in Userspace) support (NEW)
```



# Mount a File System Driver

- Make sure which File-System be supported
  - [CMD] cat /proc/filesystems

```
rock@rockpi4b:~$ cat /proc/filesystems
nodev sysfs
nodev rootfs
nodev ramfs
nodev bdev
nodev proc
nodev cpuset
nodev cgroup
nodev cgroup2
nodev tmpfs
nodev devtmpfs
nodev configfs
nodev debugfs
nodev tracefs
nodev securityfs
nodev sockfs
nodev pipefs
nodev rpc_pipefs
nodev devpts
ext3
ext2
ext4
squashfs
```



# Mount a File System Driver

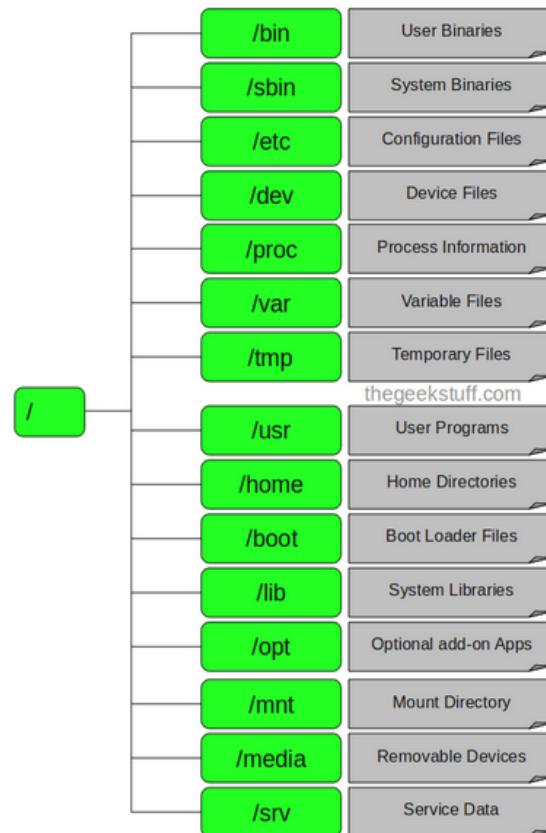
## ► Make sure which File-System be supported

- `mount -t ${FILE_SYS_TYPE} ${DISK} ${MOUNT_FOLDER}`
- Fake Image\_Disk Test
  - [CMD] `sudo dd if=/dev/zero of=/home/rock/vdidk.img bs=1M count=5 status=progress`
  - [CMD] `sudo mkfs.vfat ./vdidk.img`
  - **[CMD] `sudo mount -t vfat ./vdidk.img /tmp/vfat_folder/`**
  - [CMD] `lsblk -f`

# Root File System



# Root File System Structure





# Root

- Every single file and directory starts from the root directory
- Only root user has write privilege under this directory
- Please note that /root is root user's home directory, which is not same as /



# /bin – User Binaries

- » Contains binary executables.
- » Common linux commands you need to use in single-user modes are located under this directory.
- » Commands used by all the users of the system are located here.
- » For example: ps, ls, ping, grep, cp.



# /sbin – System Binaries

- » Just like /bin, /sbin also contains binary executables.
- » But, the linux commands located under this directory are used typically by system administrator, for system maintenance purpose.
- » For example: iptables, reboot, fdisk, ifconfig, swapon



# /etc – Configuration Files

- » Contains configuration files required by all programs.
- » This also contains startup and shutdown shell scripts used to start/stop individual programs.
- » For example: /etc/resolv.conf, /etc/logrotate.conf



# /dev – Device Files

- Contains device files.
- These include terminal devices, usb, or any device attached to the system.
- For example: /dev/tty1, /dev/usbmon0



# /proc – Process Information

- Contains information about system process.
- This is a pseudo filesystem contains information about running process. For example: /proc/{pid} directory contains information about the process with that particular pid.
- This is a virtual filesystem with text information about system resources. For example: /proc/uptime



# /tmp – Temporary Files

- Directory that contains temporary files created by system and users.
- Files under this directory are deleted when system is rebooted.



# /usr – User Programs

- Contains binaries, libraries, documentation, and source-code for second level programs.
- **/usr/bin** contains binary files for user programs. If you can't find a user binary under /bin, look under /usr/bin. For example: at, awk, cc, less, scp
- **/usr/sbin** contains binary files for system administrators. If you can't find a system binary under /sbin, look under /usr/sbin. For example: atd, cron, sshd, useradd, userdel
- **/usr/lib** contains libraries for /usr/bin and /usr/sbin
- **/usr/local** contains users programs that you install from source. For example, when you install apache from source, it goes under /usr/local/apache2



# /home – Home Directories

- Home directories for all users to store their personal files.
- For example: /home/john, /home/nikita



# /boot – Boot Loader Files

- Contains boot loader related files.
- Kernel initrd, vmlinuX, grub files are located under /boot



# /lib – System Libraries

- Contains library files that supports the binaries located under /bin and /sbin
- Library filenames are either `ld*` or `lib*.so.*`

# Linux System Initial Program



# System V

- » Unix System Five (V)
- » AT&T developed
- » The first initial process is → init
  - PID=1
- » SystemV handles startup processes through shell scripts in /etc/init\*
  - /etc/inittab
  - /etc/init.d/ and /etc/init.d/rcS



# System V

## ➤ /etc/init.d/S\*

- initial system script

## ➤ Start a Service

- [CMD] etc/init.d/S50sshd start

[CMD] etc/init.d/S50sshd stop

```
[root@rk3399:/etc/init.d]# ls
S01logging S22hdmion S50link_iq S80dnsmasq
S10init S30dbus S50sshd S99input-event-daemon
S10udev S40network S50telnet rcK
S20urandom S41dhpcd S50usbdevice rcS
S21mountall.sh S50launcher_ S66load_wifi_modules
```



# System D

- System Daemon
- /sbin/init -> /lib/systemd/systemd
  - PID=1
- SystemD is the new system that many distros are moving to
- SystemD handles startup processes through .service files



# System D Configure File

► The unit configuration files are loaded from a set of paths

- "/lib/systemd/system":
  - OS default configuration files
- "/etc/systemd/system":
  - system administrator configuration files
  - override the OS default
- "/run/systemd/system":
  - un-time generated configuration files
  - override the installed configuration files



# Service Control

➤ `systemctl ${CTL} ${SERVICE}`

- [CMD] `systemctl enable ssh`
- [CMD] `systemctl status ssh`
- [CMD] `systemctl start ssh`
- [CMD] `systemctl stop ssh`

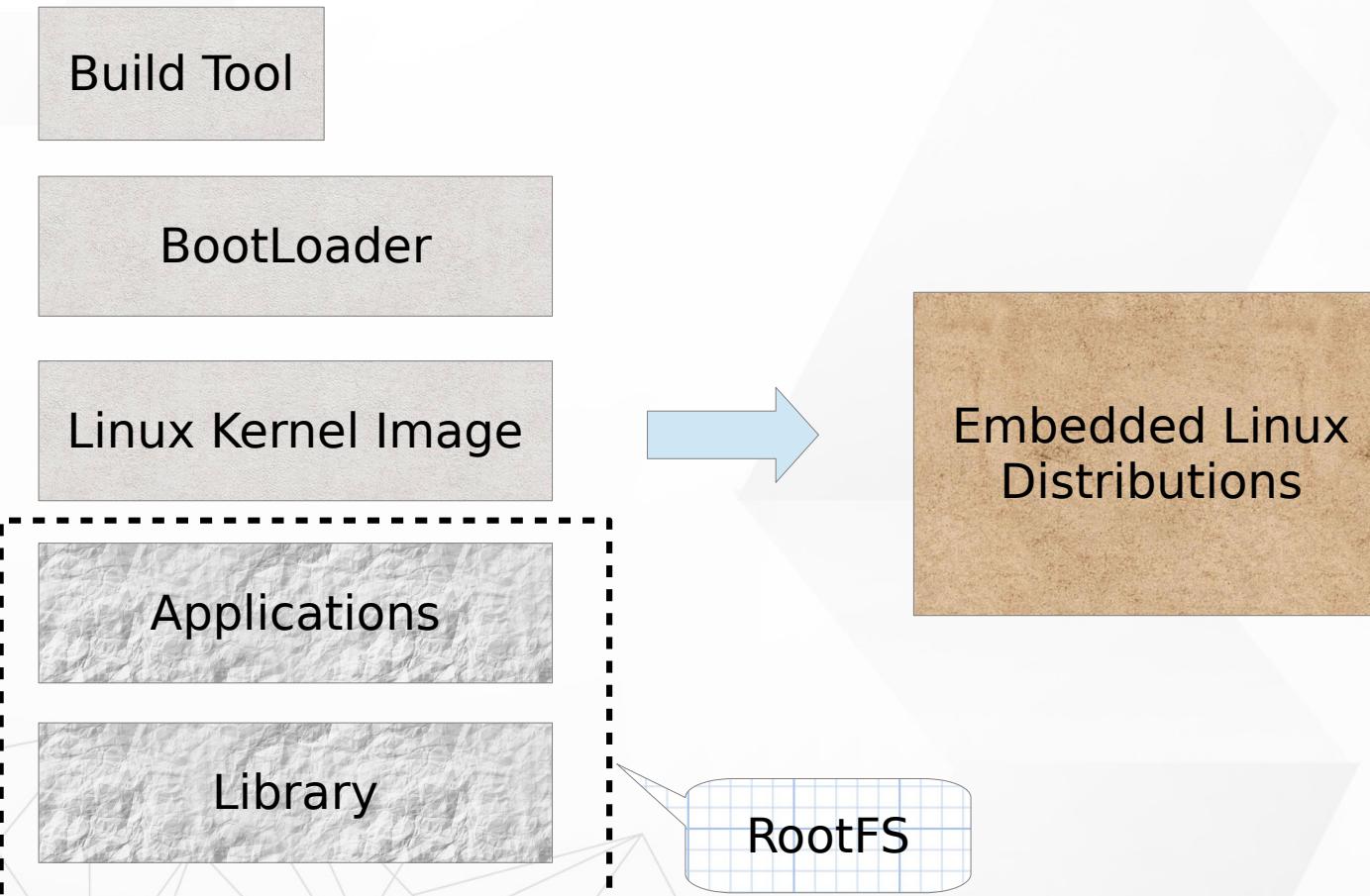
# Linux Distribution



# Linux Distribution

- Boot-loader
- Linux kernel
- RootFS
- Application
- Library
- Linux driver moudes

# Build Distribution by Tool





# System integration

<https://bootlin.com/doc/training/buildroot/buildroot-slides.pdf>

|                                                            | <b>Pros</b>                                                                                                                                                                                                                                | <b>Cons</b>                                                                                                                                                                                                                                                                         |
|------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Building everything manually</b>                        | Full flexibility<br>Learning experience                                                                                                                                                                                                    | Dependency hell<br>Need to understand a lot of details<br>Version compatibility<br>Lack of reproducibility                                                                                                                                                                          |
| <b>Binary distribution</b><br>Debian, Ubuntu, Fedora, etc. | Easy to create and extend                                                                                                                                                                                                                  | Hard to customize<br>Hard to optimize (boot time, size)<br>Hard to rebuild the full system from source<br>Large system<br>Uses native compilation (slow)<br>No well-defined mechanism to generate an image<br>Lots of mandatory dependencies<br>Not available for all architectures |
| <b>Build systems</b><br>Buildroot, Yocto, PTXdist, etc.    | Nearly full flexibility<br>Built from source: customization and optimization are easy<br>Fully reproducible<br>Uses cross-compilation<br>Have embedded specific packages not necessarily in desktop distros<br>Make more features optional | Not as easy as a binary distribution<br>Build time                                                                                                                                                                                                                                  |

# Debian



# RockPi4B Debian

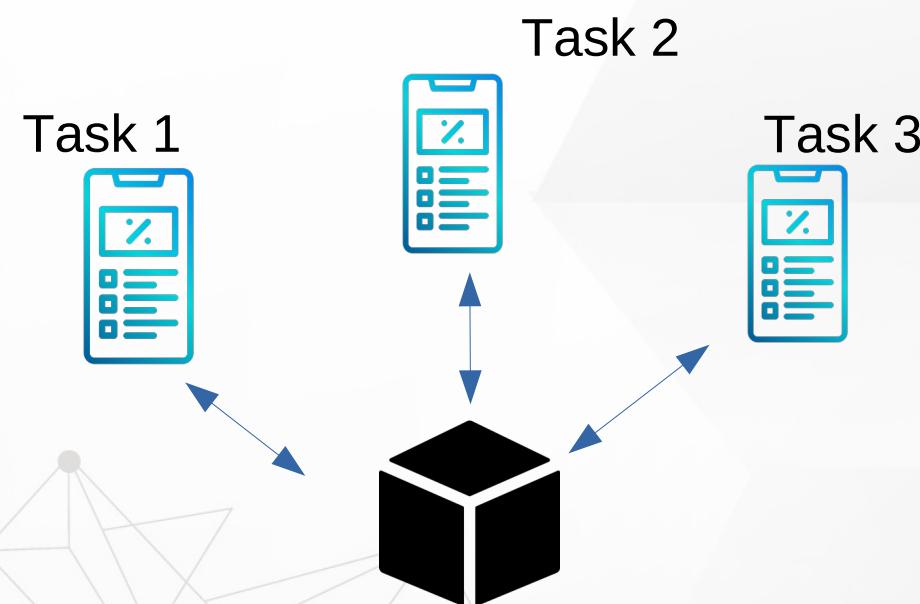
- <https://wiki.radx.com/Rockpi4/dev/Debian>
- <https://github.com/radx/rk-rootfs-build/tree/rockchip-debian>

# CH9 Linux Device Driver

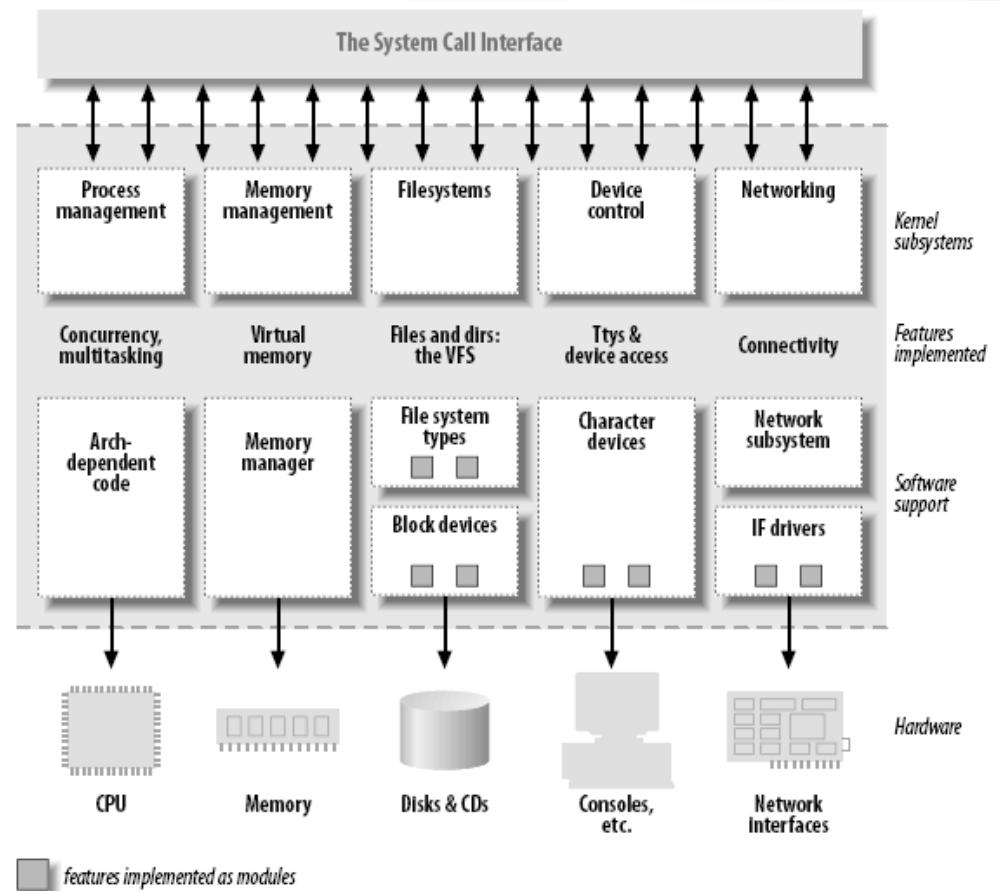
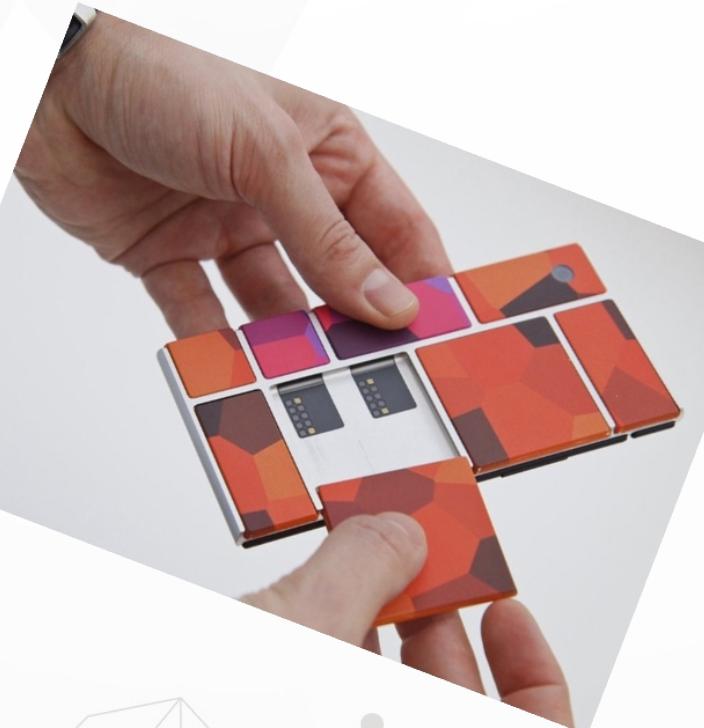
# Introduction

## ▶ Device drivers

- Black boxes to hide details of hardware devices
- Use standardized calls



# Kernel Modularization





# Example

- ▶ [CMD] make
- ▶ [CMD] sudo insmod simple.ko
- ▶ [CMD] dmesg | tail
- ▶ [CMD] lsmod | grep simple
- ▶ [CMD] sudo rmmod simple



# Classes of Devices Driver

## ▶ Char module

- simple
- access stream of bytes

## ▶ Block module

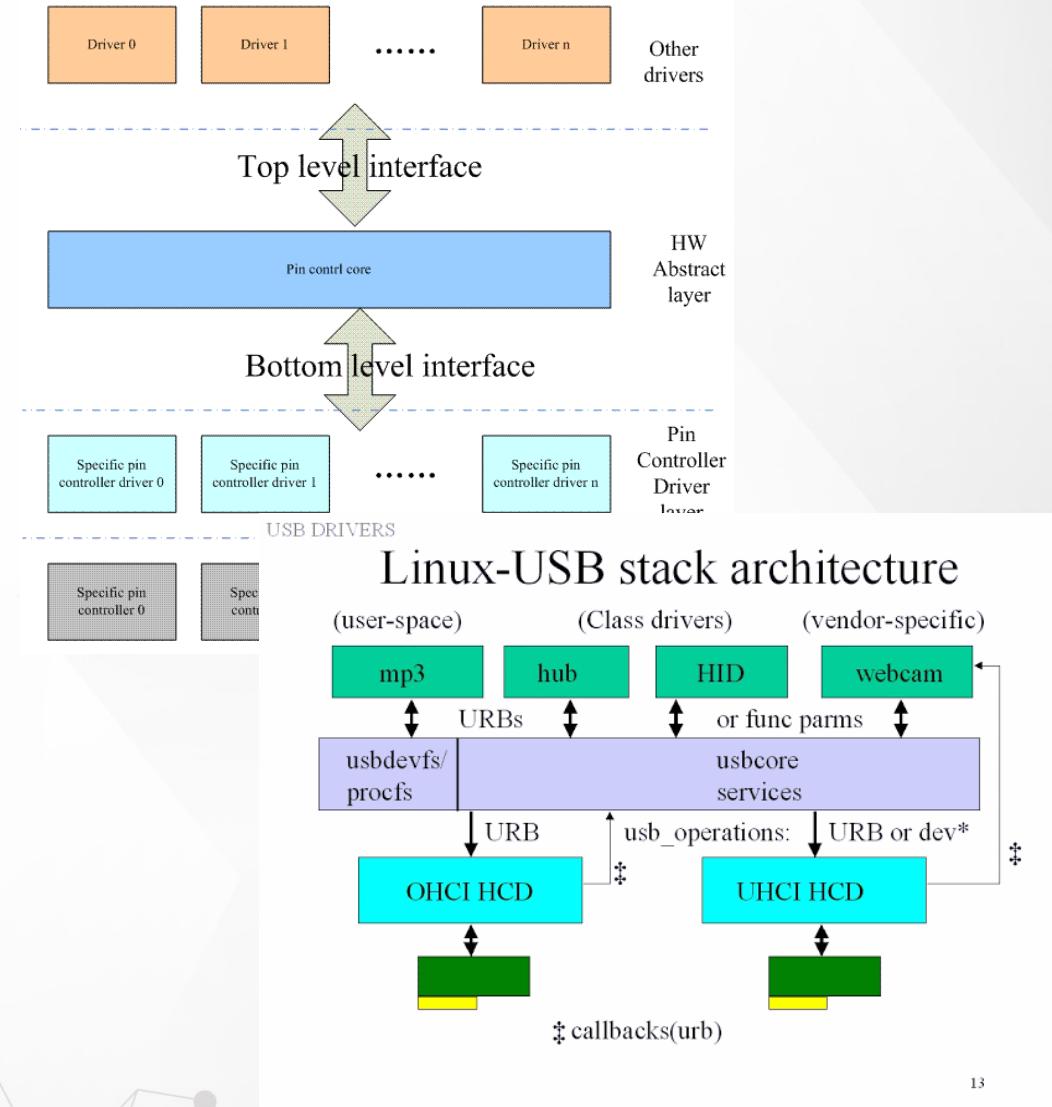
- block and char devices differ only in the way data is managed internally by the kernel

## ▶ Network module

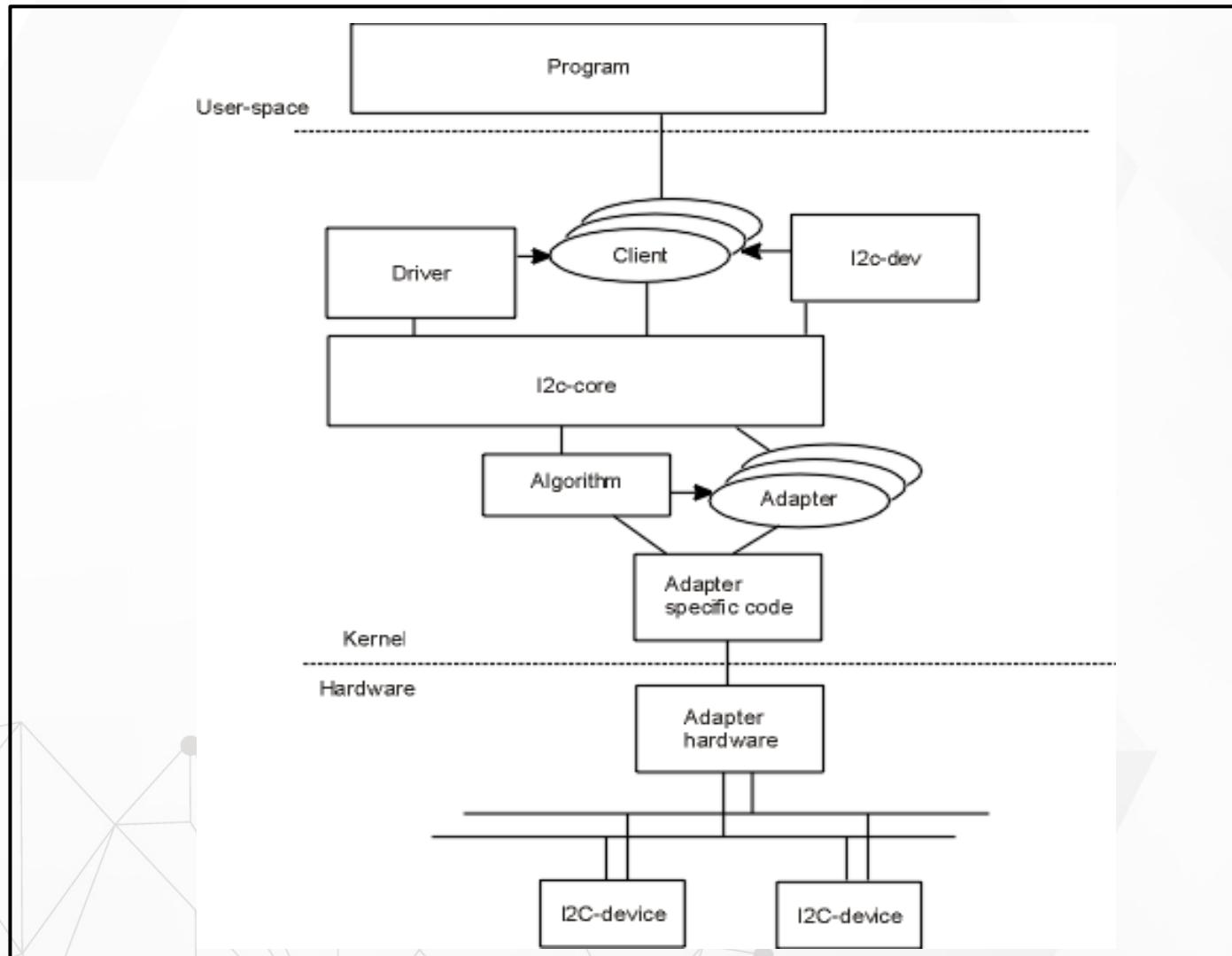
- Manage network data packets

# Subsystem

- » DRM Subsystem
- » GPIO Subsystem
- » I2C Subsystem
- » SPI Subsystem
- » MTD Subsystem



# I2C Sub-system





# Where are Modules in Kernel

- » \${KERNEL}/drivers
  - \${KERNEL}/drivers(chars
  - \${KERNEL}/drivers(i2c
  - \${KERNEL}/drivers(gpio



# Linux Kernel Configure

➤ Kernel build configure

- \${KERNEL}/.config

➤ Kconfig

- \${KERNEL}/drivers/chars/Kconfig

- [CMD] make menuconfig



# Build Modules

## ▶ Build modules

→ [CMD] make modules

## ▶ Add install patch

→ [CMD] export **INSTALL\_MOD\_PATH**=../modules

## ▶ Install module to **INSTALL\_MOD\_PATH**

→ [CMD] make modules\_install

→ Installs all modules in /lib/modules/<version>



# Module Deploy

## » **modules\_install**

- `modules.alias` : Module aliases for module loading utilities.
- `modules.dep` : Module dependencies
- `modules.symbols` : Tells which module a given symbol



# Install Module

## ▶ Install module

- \$ modprobe \${module\_name}
- \$ insmod \${module\_name}

## ▶ Remove module

- \$ modprobe -r \${module\_name}
- \$ rmmod



# modprobe depmod

## » modprobe

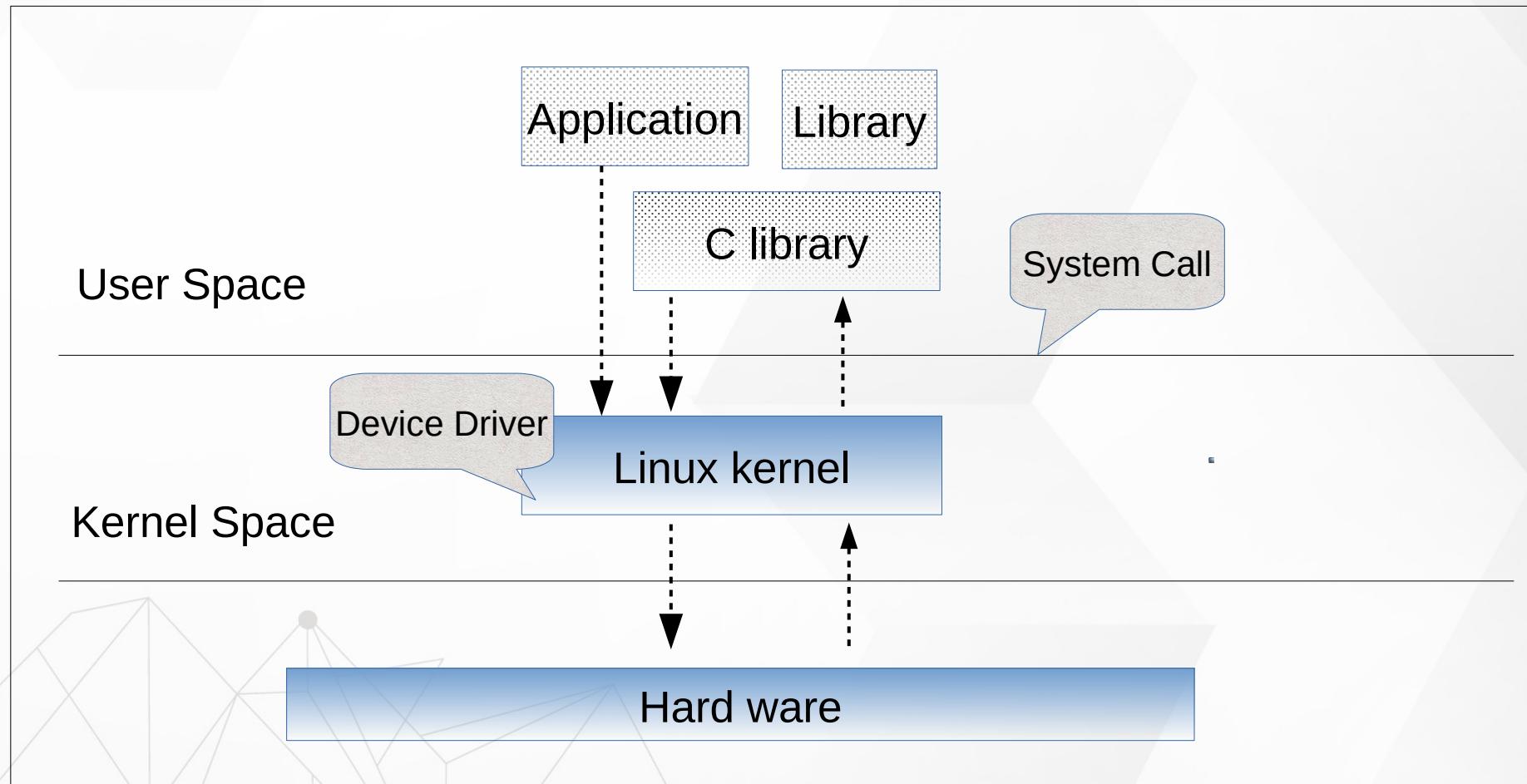
→ `/lib/modules/'uname -r'`

## » depmod

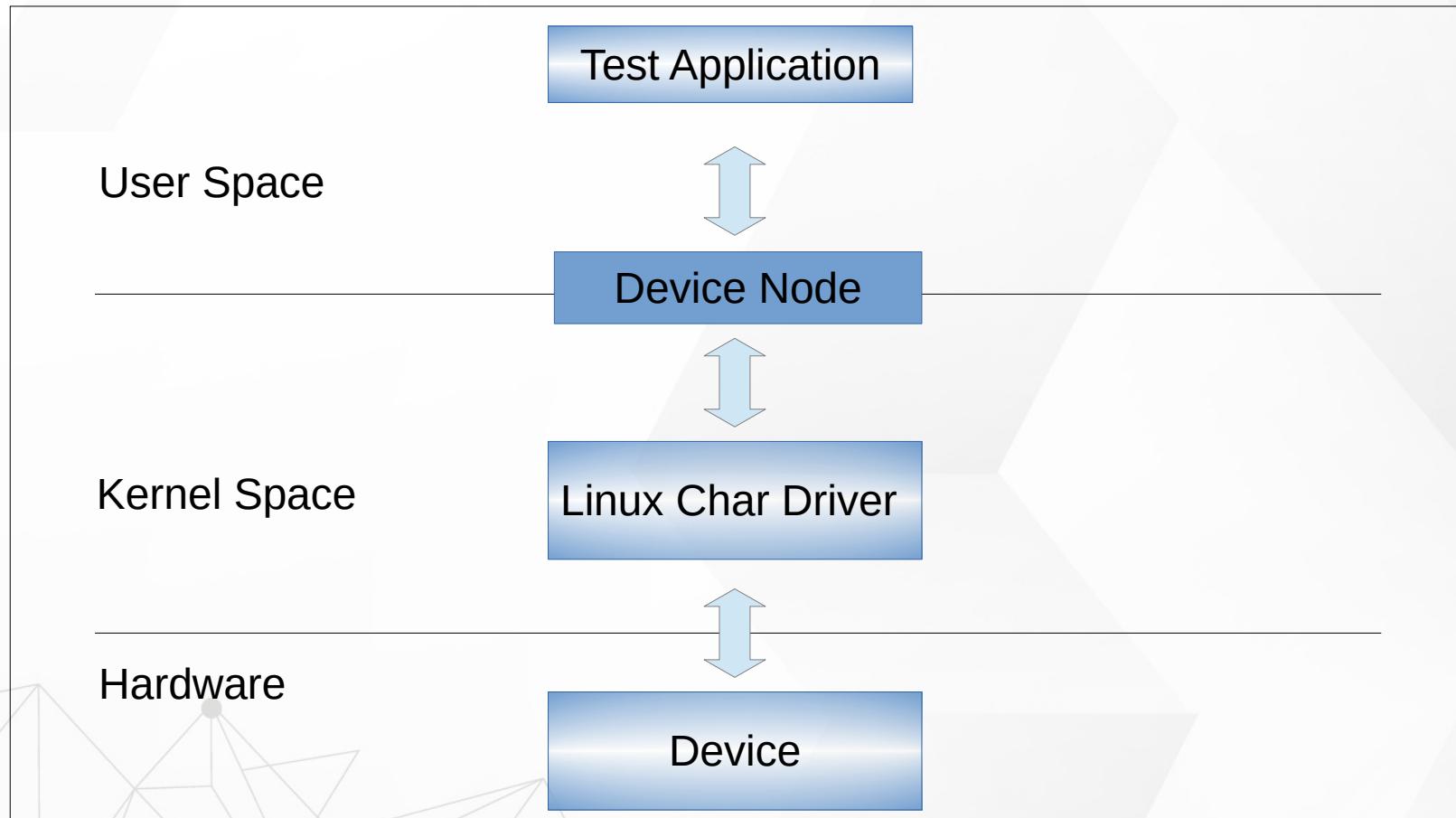
→ creates a list of module dependencies `/lib/modules/version`

# CH10 Control Hardware Driver

# Application and Hardware



# User Space and Kernel Space

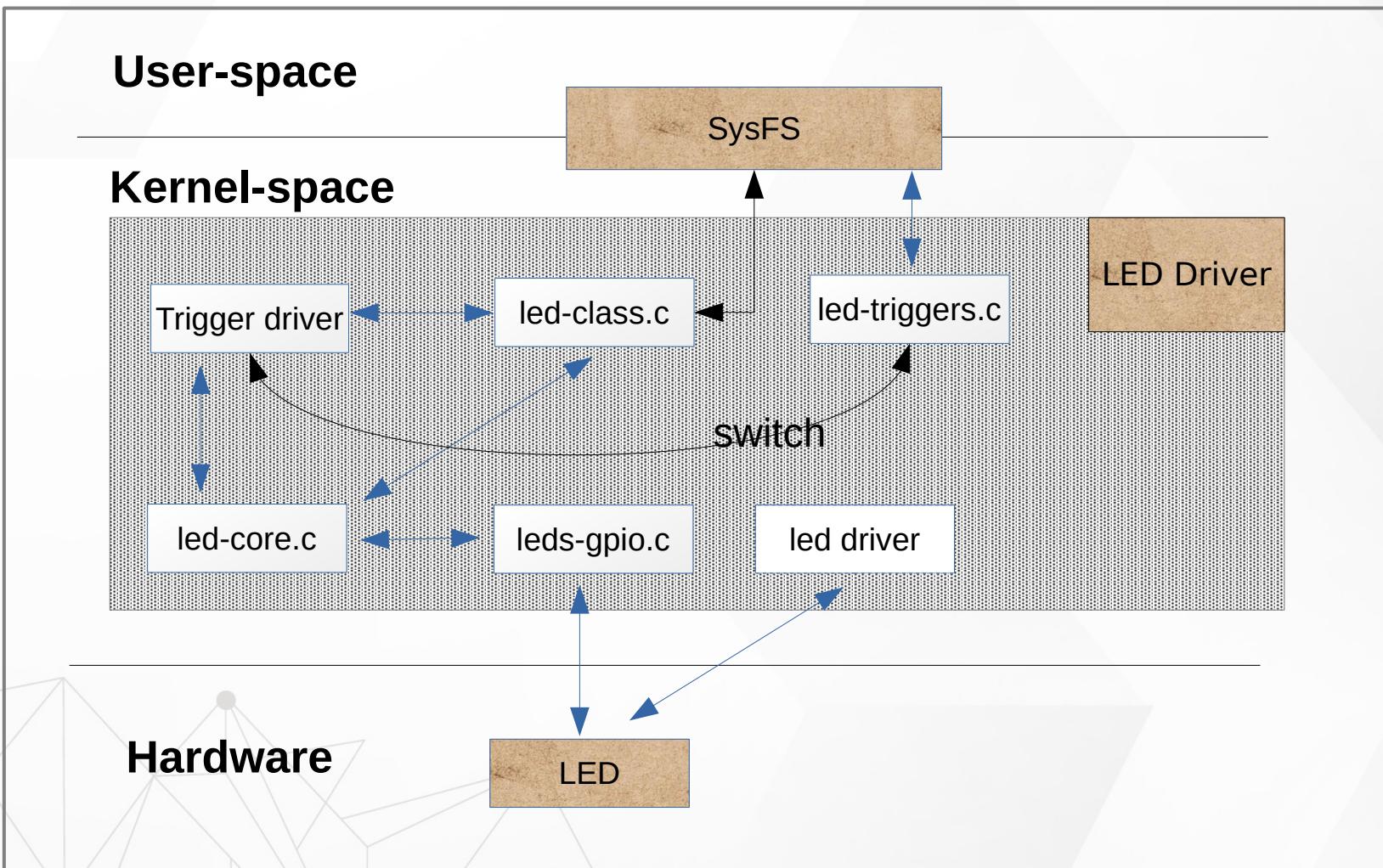




# SysFS

- ▶ Allows kernel code to export information to user processes
- ▶ SysFS is an in-memory file system
- ▶ It provides two components
  - ▶ A kernel programming interface for exporting these items via sysfs
  - ▶ User interface to view and manipulate these items that maps back to the kernel objects which they represent

# LED and SysFS





# Sys File System

```
tree -L 1 /sys/
```

```
/sys/
├── block
├── bus
├── class
├── dev
├── devices
├── firmware
├── fs
├── hypervisor
├── kernel
├── module
└── power
```

```
tree -L 1 /sys/class/i2c-dev/i2c-0/
```

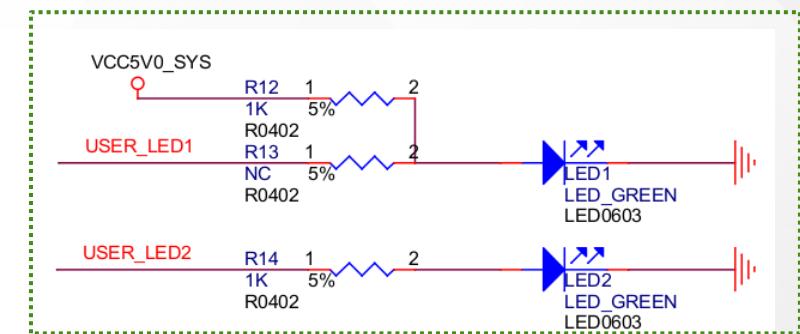
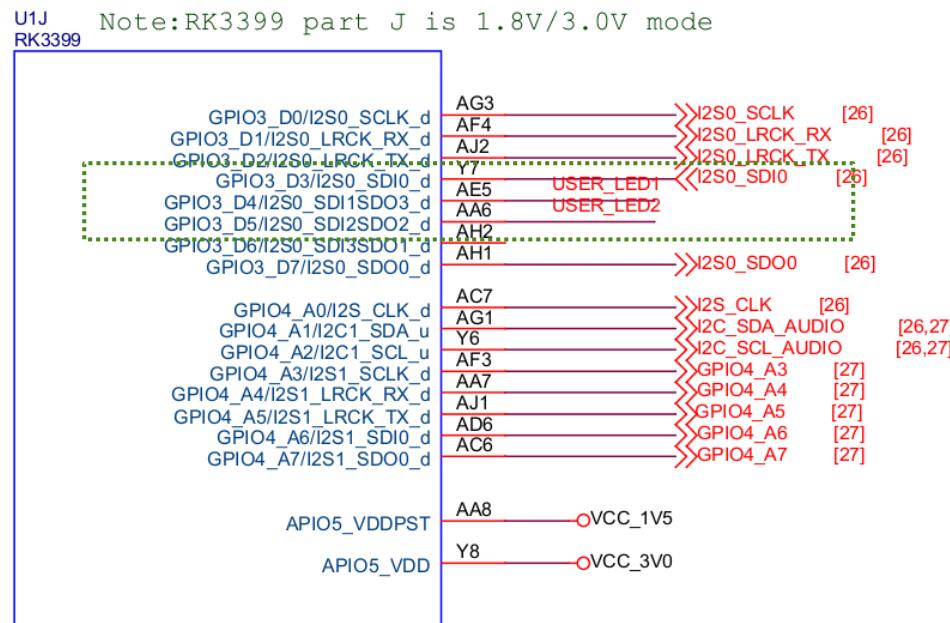
```
/sys/class/i2c-dev/i2c-0/
├── dev
├── device -> ../../i2c-0
├── name
├── power
└── subsystem -> ../../../../../../class/i2c-dev
├── uevent
```

```
tree -L 1 /sys/class/i2c-dev
```

```
/sys/class/i2c-dev/
├── i2c-0 -> ../../devices/pci0000:00/0000:00:02.0/i2c-0/i2c-dev/i2c-0
├── i2c-1 -> ../../devices/pci0000:00/0000:00:02.0/i2c-1/i2c-dev/i2c-1
├── i2c-2 -> ../../devices/pci0000:00/0000:00:02.0/i2c-2/i2c-dev/i2c-2
├── i2c-3 -> ../../devices/pci0000:00/0000:00:02.0/i2c-3/i2c-dev/i2c-3
├── i2c-4 -> ../../devices/pci0000:00/0000:00:02.0/i2c-4/i2c-dev/i2c-4
├── i2c-5 -> ../../devices/pci0000:00/0000:00:02.0/i2c-5/i2c-dev/i2c-5
├── i2c-6 -> ../../devices/pci0000:00/0000:00:02.0/drm/card0/card0-DP-1/i2c-6/i2c-dev/i2c-6
├── i2c-7 -> ../../devices/pci0000:00/0000:00:02.0/drm/card0/card0-DP-2/i2c-7/i2c-dev/i2c-7
└── i2c-8 -> ../../devices/pci0000:00/0000:00:02.0/drm/card0/card0-DP-3/i2c-8/i2c-dev/i2c-8
```

# LED Drivers

# LED Schematic



GPIO3\_D4/I2S0\_SDIO1SDO3\_d → LED1

GPIO3\_D5/I2S0\_SDIO2SDO2\_d → LED2



# LED Subsystem

▶ Control LED convenient with SysFS

▶ For example

- echo 1 > /sys/class/leds/user-led2/shot

▶ Switch different LED trigger type in SysFS

▶ For example

- echo "gpio" > /sys/class/leds/user-led2/trigger
- echo "1" > /sys/class/leds/user-led2/brightness
- echo "0" > /sys/class/leds/user-led2/brightness

# LED SysFS

```
root@rockpi4b:/sys/class/leds/user-led2# ls -l
```

```
brightness
device -> ../../../../../gpio-leds
max_brightness
power
subsystem -> ../../../../../../class/leds
trigger
uevent
```

Check trigger type

```
root@rockpi4b:/sys/class/leds/user-led2# cat trigger
none rc-feedback kbd-scrolllock kbd-numlock kbd-capslock
kbd-kanalock kbd-shiftlock kbd-altgrlock kbd-ctrllock kbd-altlock
kbd-shiftllock kbd-shiftrlock kbd-ctrllock kbd-ctrlrlock mmc0 mmc1
timer oneshot heartbeat backlight [gpio] cpu0 cpu1 cpu2 cpu3
cpu4 cpu5
```

Switch trigger type

```
root@rockpi4b:/sys/class/leds/user-led2# echo heartbeat > trigger
```

```
root@rockpi4b:/sys/class/leds/user-led2# cat trigger
none rc-feedback kbd-scrolllock kbd-numlock kbd-capslock
kbd-kanalock kbd-shiftlock kbd-altgrlock kbd-ctrllock kbd-altlock
kbd-shiftllock kbd-shiftrlock kbd-ctrllock kbd-ctrlrlock mmc0 mmc1
timer oneshot [heartbeat] backlight gpio cpu0 cpu1 cpu2 cpu3
cpu4 cpu5
```

# GPIO Control



# Driver LED in User Space

## ▶ Paths in Sysfs

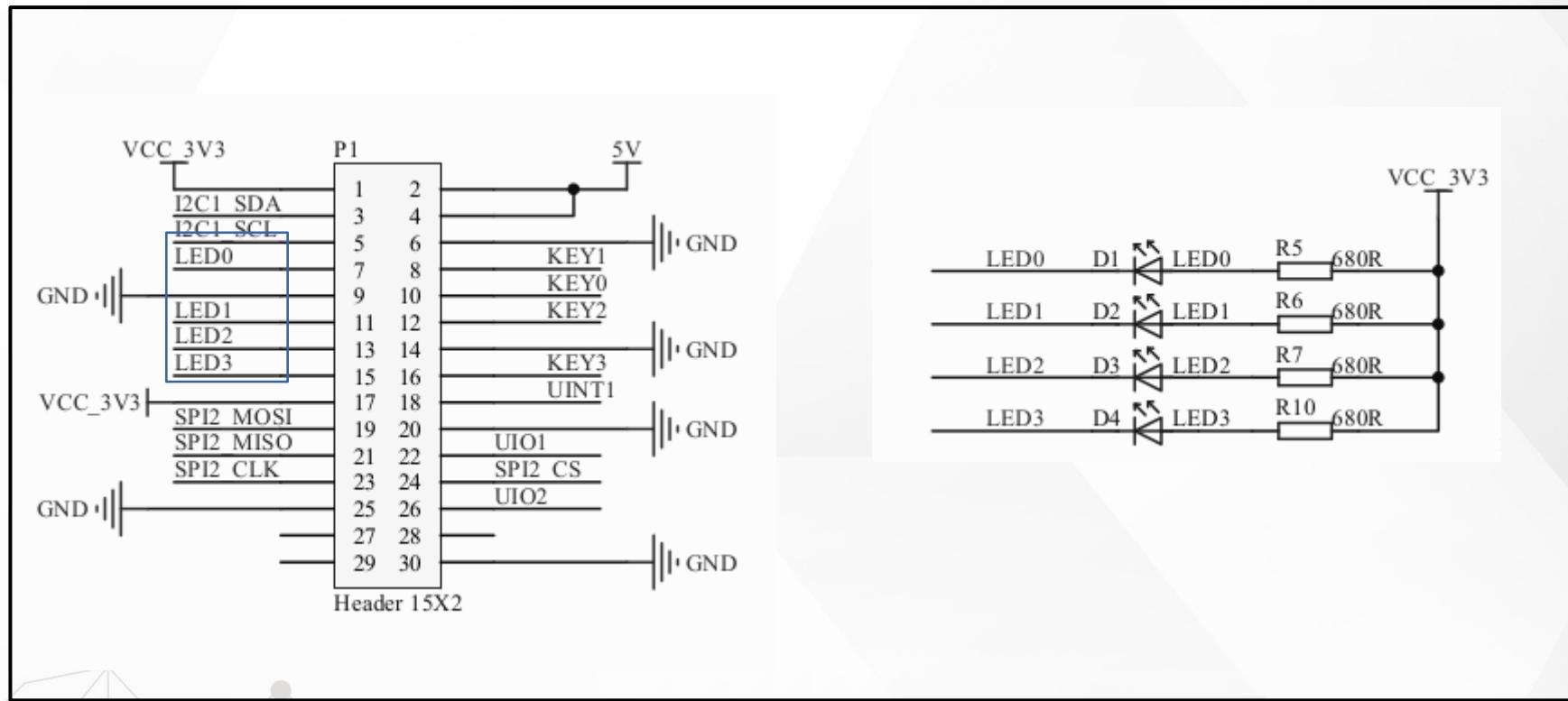
### ▶ /sys/class/gpio:

- Control interfaces used to get userspace control over GPIOs;
- GPIOs themselves
- GPIO controllers("gpio\_chip" instances)

### ▶ /sys/class/gpio/

- "export" : ask the kernel to export GPIO to userspace by writing
  - "echo 19 > export"
  - create a "gpio19" node in /sys/class/gpio
- "unexport" : Reverses the effect of exporting to userspace
  - "echo 19 > unexport"
  - remove "gpio19" node from /sys/class/gpio

# Cadtc Ext Board LED



# RockPi4B HEAD

## Rock Pi 4 A/B/C general purpose input-output (GPIO) connector

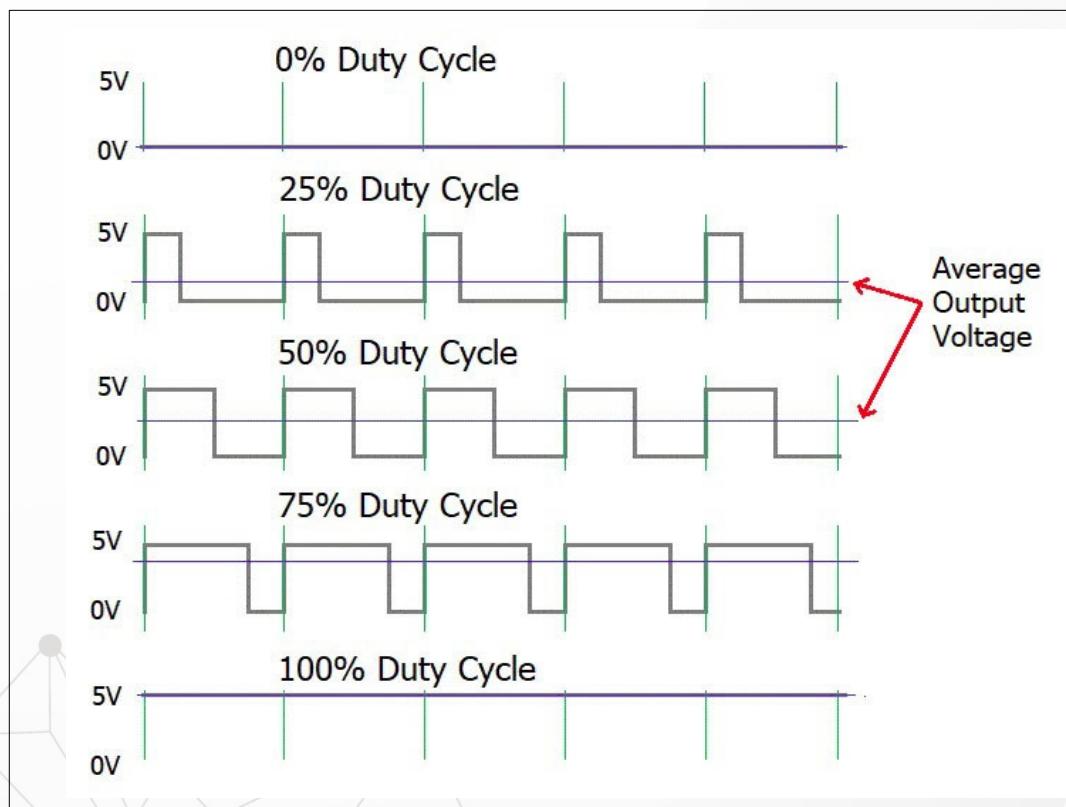
ROCK Pi 4 has a 40-pin expansion header. Each pin is distinguished by color.

| GPIO number | Function2 | Function1    | GPIO     | Pin# | Pin# | GPIO     | Function1    | Function2  | GPIO number |
|-------------|-----------|--------------|----------|------|------|----------|--------------|------------|-------------|
|             |           | +3.3V        |          | 1    | 2    |          | +5.0V        |            |             |
| 71          |           | I2C7_SDA     | GPIO2_A7 | 3    | 4    |          | +5.0V        |            |             |
| 72          |           | I2C7_SCL     | GPIO2_B0 | 5    | 6    |          | GND          |            |             |
| 75          |           | SPI2_CLK     | GPIO2_B3 | 7    | 8    | GPIO4_C4 | UART2_TXD    |            | 148         |
|             |           | GND          |          | 9    | 10   | GPIO4_C3 | UART2_RXD    |            | 147         |
| 146         |           | PWM0         | GPIO4_C2 | 11   | 12   | GPIO4_A3 | I2S1_SCLK    |            | 131         |
| 150         |           | PWM1         | GPIO4_C6 | 13   | 14   |          | GND          |            |             |
| 149         |           | SPDIF_TX     | GPIO4_C5 | 15   | 16   | GPIO4_D2 |              |            | 154         |
|             |           | +3.3V        |          | 17   | 18   | GPIO4_D4 |              |            | 156         |
| 40          | UART4_TXD | SPI1_TXD     | GPIO1_B0 | 19   | 20   |          | GND          |            |             |
| 39          | UART4_RXD | SPI1_RXD     | GPIO1_A7 | 21   | 22   | GPIO4_D5 |              |            | 157         |
| 41          |           | SPI1_CLK     | GPIO1_B1 | 23   | 24   | GPIO1_B2 | SPI1_CSn     |            | 42          |
|             |           | GND          |          | 25   | 26   |          | ADC_IN0      |            |             |
| 64          |           | I2C2_SDA     | GPIO2_A0 | 27   | 28   | GPIO2_A1 | I2C2_CLK     |            | 65          |
| 74          | I2C6_SCL  | SPI2_TXD     | GPIO2_B2 | 29   | 30   |          | GND          |            |             |
| 73          | I2C6_SDA  | SPI2_RXD     | GPIO2_B1 | 31   | 32   | GPIO3_C0 | SPDIF_TX     | UART3_CTSn | 112         |
| 76          |           | SPI2_CSn     | GPIO2_B4 | 33   | 34   |          | GND          |            |             |
| 133         |           | I2S1_LRCK_TX | GPIO4_A5 | 35   | 36   | GPIO4_A4 | I2S1_LRCK_RX |            | 132         |
| 158         |           |              | GPIO4_D6 | 37   | 38   | GPIO4_A6 | I2S1_SD      |            | 134         |
|             |           | GND          |          | 39   | 40   | GPIO4_A7 | I2S1_SDO     |            | 135         |

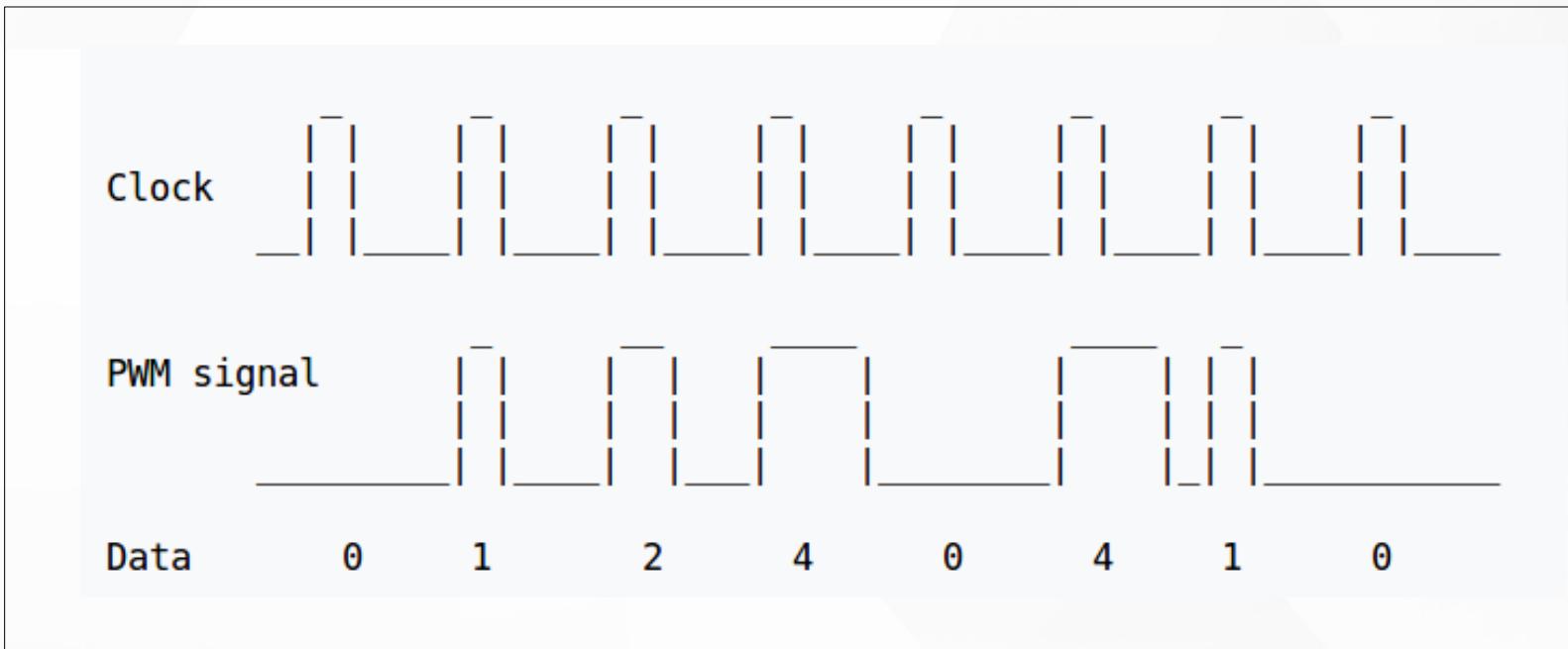
# PWM Sub System

# PWM

▶ PWM : Pulse Width Modulation



# PWM



[https://en.wikipedia.org/wiki/Pulse-width\\_modulation](https://en.wikipedia.org/wiki/Pulse-width_modulation)



# PWM Parameter in Linux

## ▶ Period

- The total period of the PWM signal
- Value is in nanoseconds
- sum of the active and inactive time of the PWM

## ▶ duty\_cycle

- The active time of the PWM signal
- Value is in nanoseconds
- must be less than the period.



# PWM Parameter in Linux

## ► Polarity

→ The polarity of the PWM signal

## ► Enable

→ Enable PWM Signal



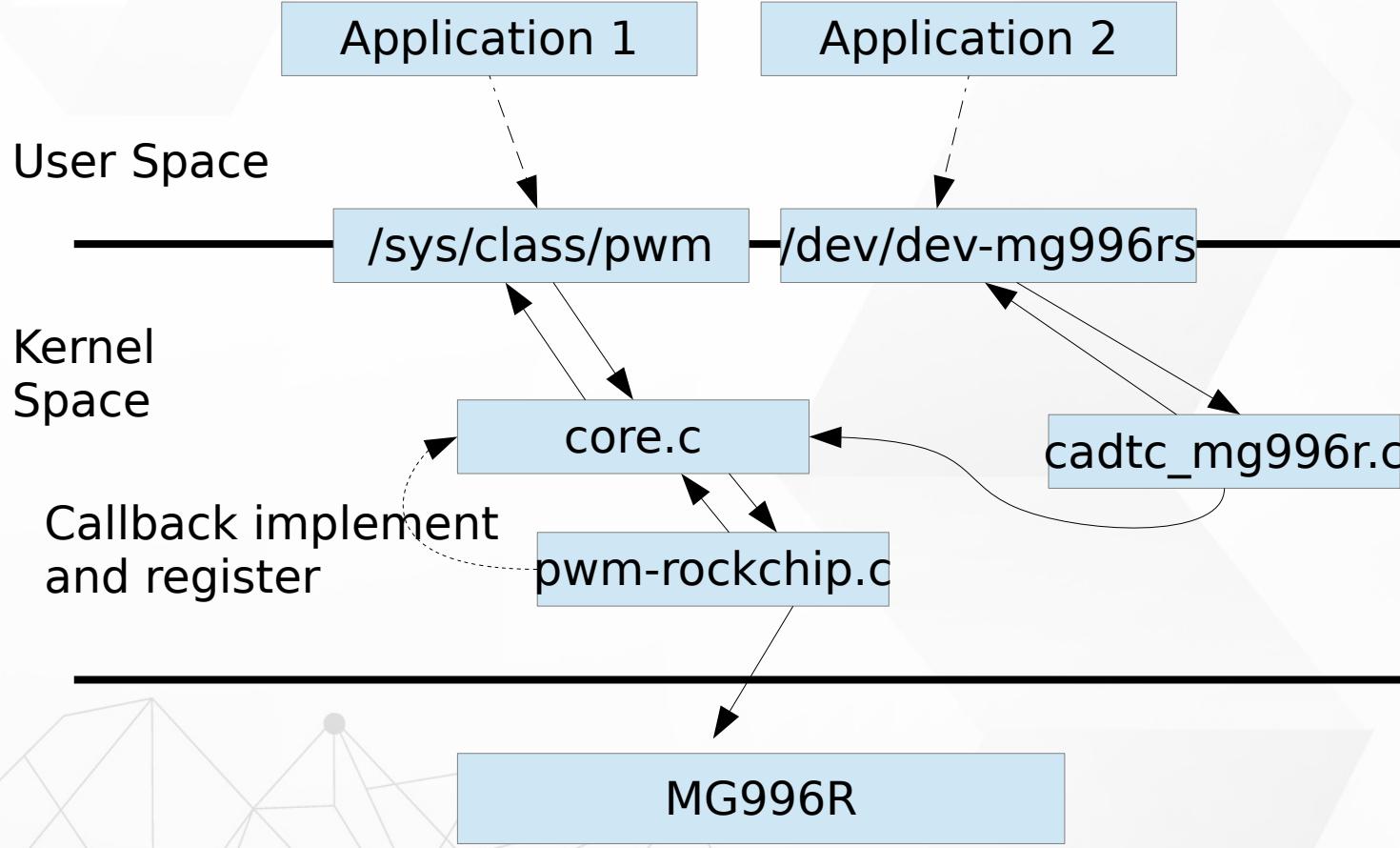
# PWM Driver

➤ \$(KERNEL\_SRC)/Documentation/pwm.txt

➤ Platform Driver

- drivers/pwm/
- drivers/pwm/core.c
- drivers/pwm/pwm-rockchip.c

# PWM Subsystem



# RockPi4B HEAD

## Rock Pi 4 A/B/C general purpose input-output (GPIO) connector

ROCK Pi 4 has a 40-pin expansion header. Each pin is distinguished by color.

| GPIO number | Function2 | Function1    | GPIO     | Pin# | Pin# | GPIO     | Function1    | Function2  | GPIO number |
|-------------|-----------|--------------|----------|------|------|----------|--------------|------------|-------------|
|             |           | +3.3V        |          | 1    | 2    |          | +5.0V        |            |             |
| 71          |           | I2C7_SDA     | GPIO2_A7 | 3    | 4    |          | +5.0V        |            |             |
| 72          |           | I2C7_SCL     | GPIO2_B0 | 5    | 6    |          | GND          |            |             |
| 75          |           | SPI2_CLK     | GPIO2_B3 | 7    | 8    | GPIO4_C4 | UART2_TXD    |            | 148         |
|             |           | GND          |          | 9    | 10   | GPIO4_C3 | UART2_RXD    |            | 147         |
| 146         |           | PWM0         | GPIO4_C2 | 11   | 12   | GPIO4_A3 | I2S1_SCLK    |            | 131         |
| 150         |           | PWM1         | GPIO4_C6 | 13   | 14   |          | GND          |            |             |
| 149         |           | SPDIF_TX     | GPIO4_C5 | 15   | 16   | GPIO4_D2 |              |            | 154         |
|             |           | +3.3V        |          | 17   | 18   | GPIO4_D4 |              |            | 156         |
| 40          | UART4_TXD | SPI1_TXD     | GPIO1_B0 | 19   | 20   |          | GND          |            |             |
| 39          | UART4_RXD | SPI1_RXD     | GPIO1_A7 | 21   | 22   | GPIO4_D5 |              |            | 157         |
| 41          |           | SPI1_CLK     | GPIO1_B1 | 23   | 24   | GPIO1_B2 | SPI1_CSn     |            | 42          |
|             |           | GND          |          | 25   | 26   |          | ADC_IN0      |            |             |
| 64          |           | I2C2_SDA     | GPIO2_A0 | 27   | 28   | GPIO2_A1 | I2C2_CLK     |            | 65          |
| 74          | I2C6_SCL  | SPI2_TXD     | GPIO2_B2 | 29   | 30   |          | GND          |            |             |
| 73          | I2C6_SDA  | SPI2_RXD     | GPIO2_B1 | 31   | 32   | GPIO3_C0 | SPDIF_TX     | UART3_CTSn | 112         |
| 76          |           | SPI2_CSn     | GPIO2_B4 | 33   | 34   |          | GND          |            |             |
| 133         |           | I2S1_LRCK_TX | GPIO4_A5 | 35   | 36   | GPIO4_A4 | I2S1_LRCK_RX |            | 132         |
| 158         |           |              | GPIO4_D6 | 37   | 38   | GPIO4_A6 | I2S1_SD      |            | 134         |
|             |           | GND          |          | 39   | 40   | GPIO4_A7 | I2S1_SDO     |            | 135         |



# PWM SYSFS

```
/sys/class/pwm/pwmchip0
```

```
device export npwm power subsystem uevent unexport
```

```
echo 0 > export
```

```
capture enable polarity uevent duty_cycle period power
```

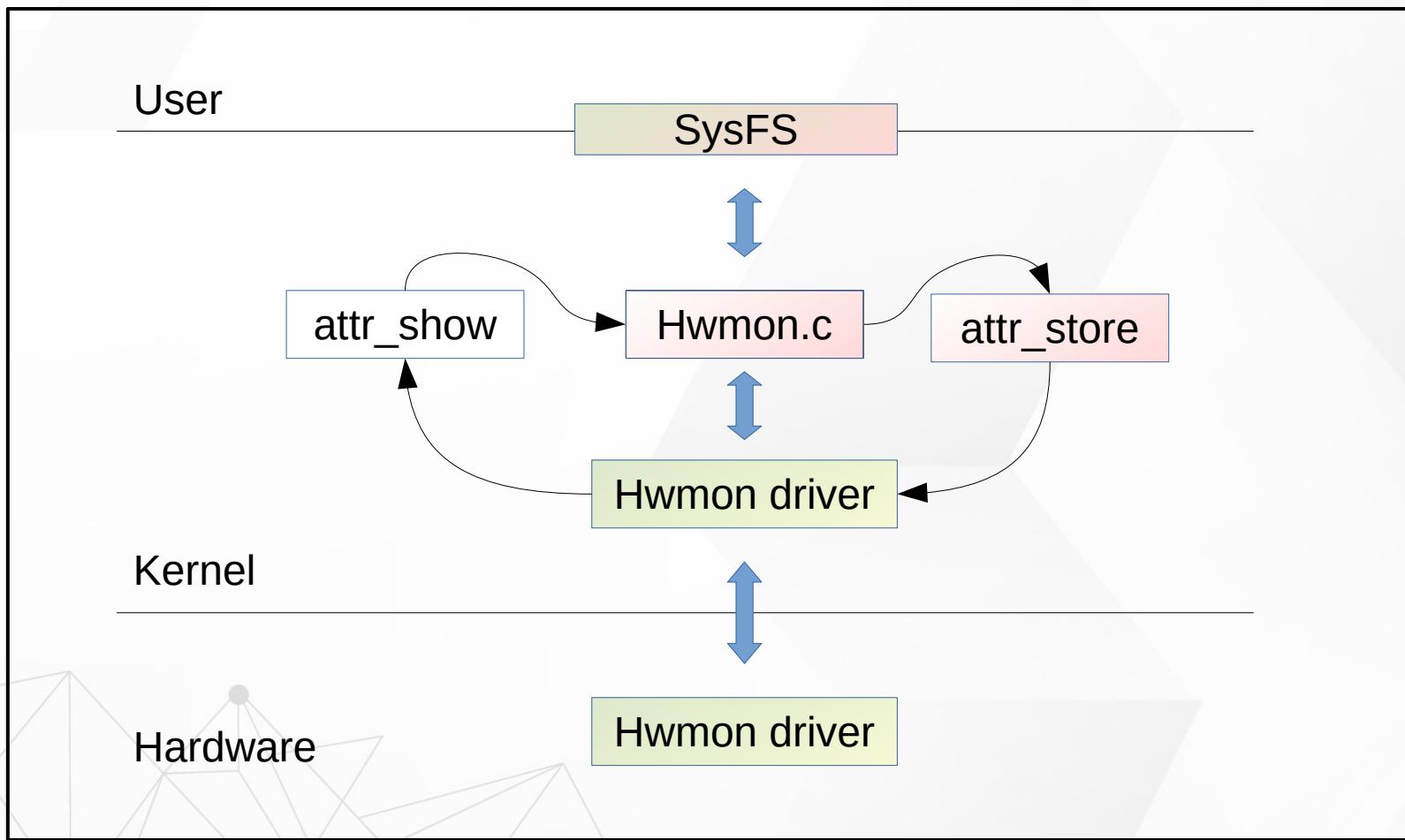
```
echo "2000000" > period //20ms, 50 Hz
```

```
echo "200000" > duty_cycle //2ms
```

```
echo 1 > enable //Enable
```

# Hwmon Subsystem

# Hwmon Subsystem





# SHT21

► Simple interface

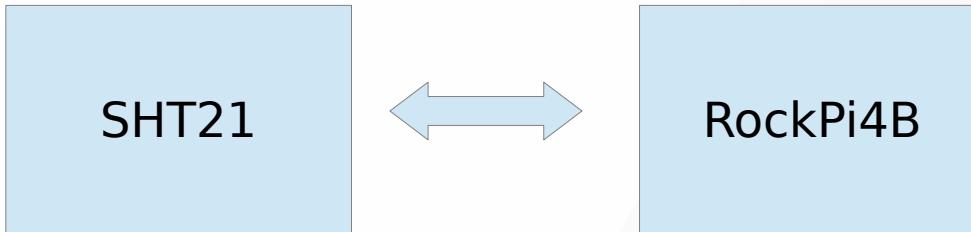
► Bus interface

- I2C, GPIO, SPI

► Sensors

- Temperature
- Voltage
- Humidity
- Fan speed
- PWM control

# SHT21



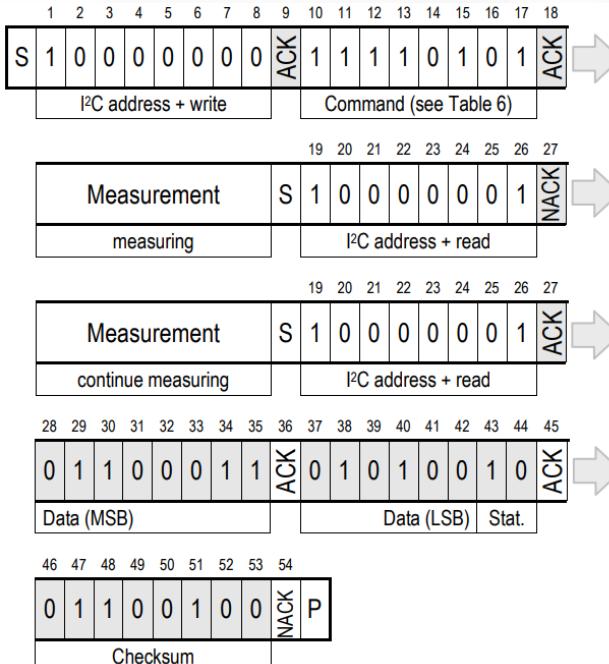
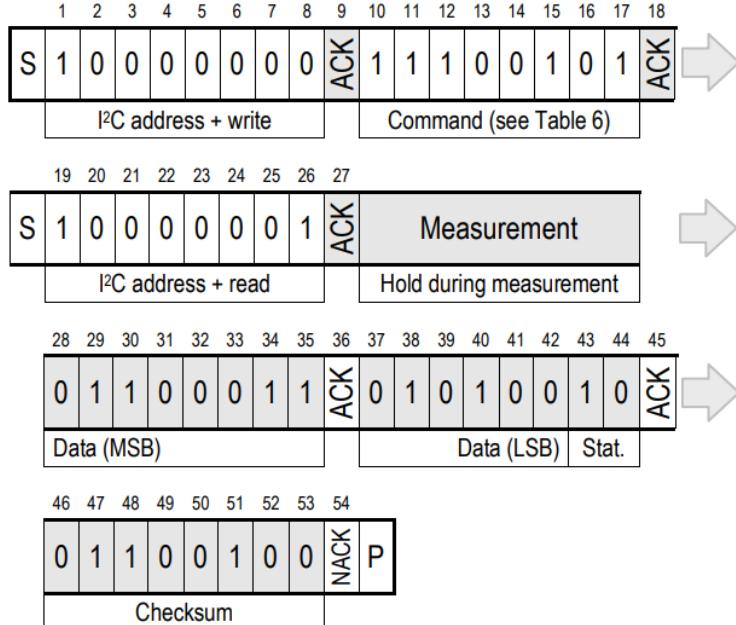
| Pin | Name | Comment                     |   |
|-----|------|-----------------------------|---|
| 1   | SDA  | Serial Data, bidirectional  | 4 |
| 2   | VSS  | Ground                      | 5 |
| 5   | VDD  | Supply Voltage              | 6 |
| 6   | SCL  | Serial Clock, bidirectional | 3 |
| 3,4 | NC   | Not Connected               | 2 |
|     |      |                             | 1 |

A small gray rectangle represents the physical pin layout of the SHT21 sensor. The pins are numbered 1 through 6 around the perimeter. Pin 1 is at the top, pin 2 is at the bottom, pin 3 is at the right, pin 4 is at the top-left, pin 5 is at the bottom-left, and pin 6 is at the right.

| Command                | Comment        | Code      |
|------------------------|----------------|-----------|
| Trigger T measurement  | hold master    | 1110'0011 |
| Trigger RH measurement | hold master    | 1110'0101 |
| Trigger T measurement  | no hold master | 1111'0011 |
| Trigger RH measurement | no hold master | 1111'0101 |
| Write user register    |                | 1110'0110 |
| Read user register     |                | 1110'0111 |
| Soft reset             |                | 1111'1110 |

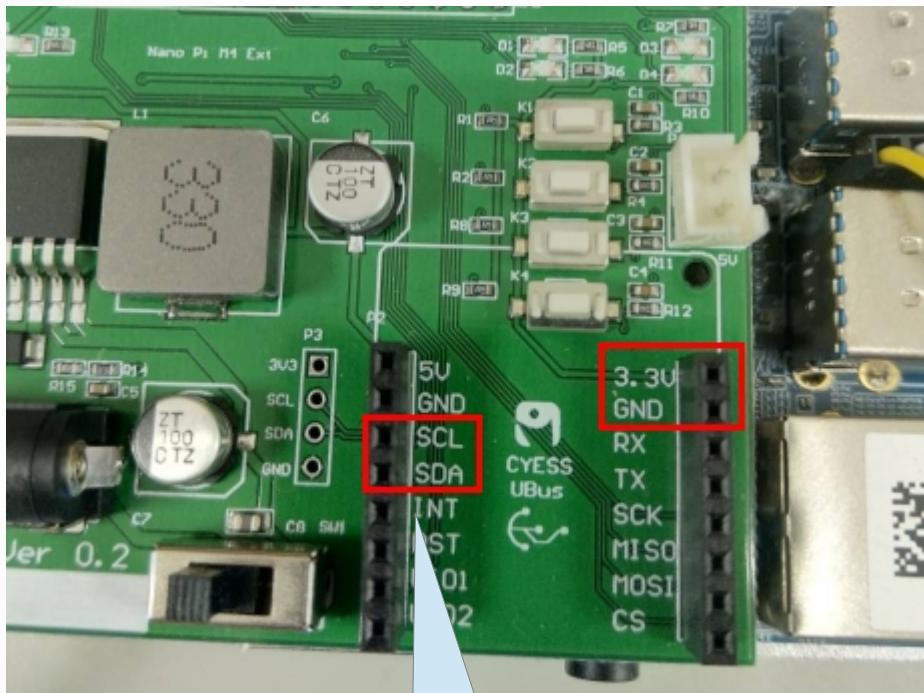
# SHT21

## Hold master communication sequence      No Hold master communication sequence

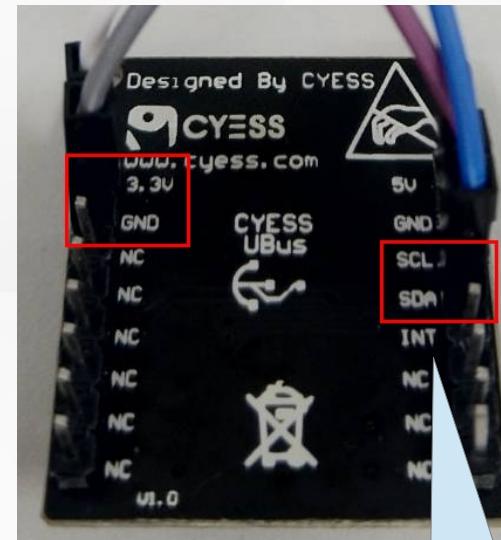




# SHT21



I2C



I2C



# SHT21

## Hwmon Sysfs

```
ls /sys/class/hwmon/hwmon0
device name subsystem uevent
humidity1_input power temp1_input
```

## temperature

```
cat /sys/class/hwmon/hwmon0/temp1_input
32279
```

## humidity

```
cat /sys/class/hwmon/hwmon0/humidity1_input
34512
```