

Embedded Linux System



OutLine

- ▶ CH1 Introduction to Embedded System p. 1
- ▶ CH2 Embedded Linux Arch p. 18
- ▶ CH3 Basic Software And Tool p. 28
- ▶ CH4 Cross-compile Toolchain p. 95
- ▶ CH5 Introduction to Bootloader (u-boot) p. 119
- ▶ CH6 Embedded Linux Kernel p. 176
- ▶ CH7 RootFS p. 205
- ▶ CH8 Linux User Land p. 273
- ▶ CH9 Linux Device Driver p. 313

CH1 Introduction to Embedded System



- ▶ 嵌入式系統 定義
- ▶ 嵌入式系統 相關知識
 - ▶ SOC : ARM, MIPS, PowerPC ...
 - ▶ 週邊設備 : Flash, eMMC, I2C, I2S,
- ▶ 嵌入式作業系統 基本原理
- ▶ 開發版 NanoPi-M4 介紹
- ▶ 外掛模組介紹：
 - ▶ 溫溼度模組，亮度模組 ..
- ▶ 學習建立 嵌入系統開發環境



CH2 Embedded Linux Arch

- ▶ To know Embedded Linux System



CH3 Basic Software And Tool

- » GPL
- » Tool
 - Develop Tool
 - Video and Audio Tool
 - WiFi and Ethernet
 - Build Code Tool

CH4 Cross-compile Toolchain

- ▶ How to Cross-compile with Toolchain
- ▶ Create a Linux library to use

CH5 Introduction to Bootloader

- To know Bootloader
- How to U-boot
- Detail into U-boot

CH6 Embedded Linux Kernel

- How to build Linux kernel
- How to scale Linux kernel
- To know Linux kernel something



CH7 RootFS

- To know RootFS in embedded system
- How to create a RootFS with Buildroot



CH8 Linux User Land

- To know Linux user land
- To know user land application
- How to control device on the Linux user land



CH9 Linux Device Driver

- To know Linux device driver
- How to build a Linux device driver
- How to install/remove a device driver

Introduction to Embedded System



Embedded System

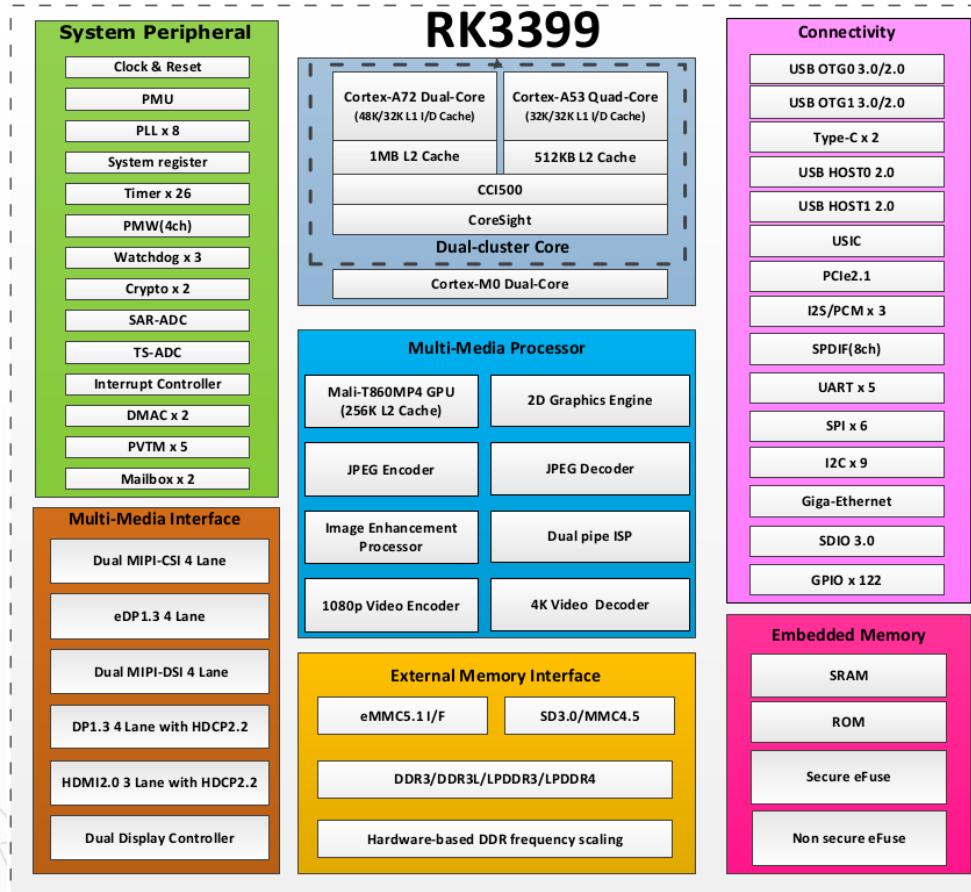
- An embedded system
 - combination of computer hardware and software
 - specifically designed for a particular function
- Applications
 - Mobile phone
 - Digital camera
 - Smart TV
 - Navigation system



Feature

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

SOC RK3399



http://wiki.friendlyarm.com/wiki/index.php/NanoPi_M4#Diagram.2C_Layout_and_Dimension



Component of embedded system

- Processor
 - ARM, X86, MIPS
- RAM
 - 8MB ~ 2 GB
- Storagee
 - Nand, Nor flash
 - SD/MMC/eMMC
- System Bus
 - AMBA, AHB, APB, AXI ...



Component of embedded system

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



Embedded Linux ?

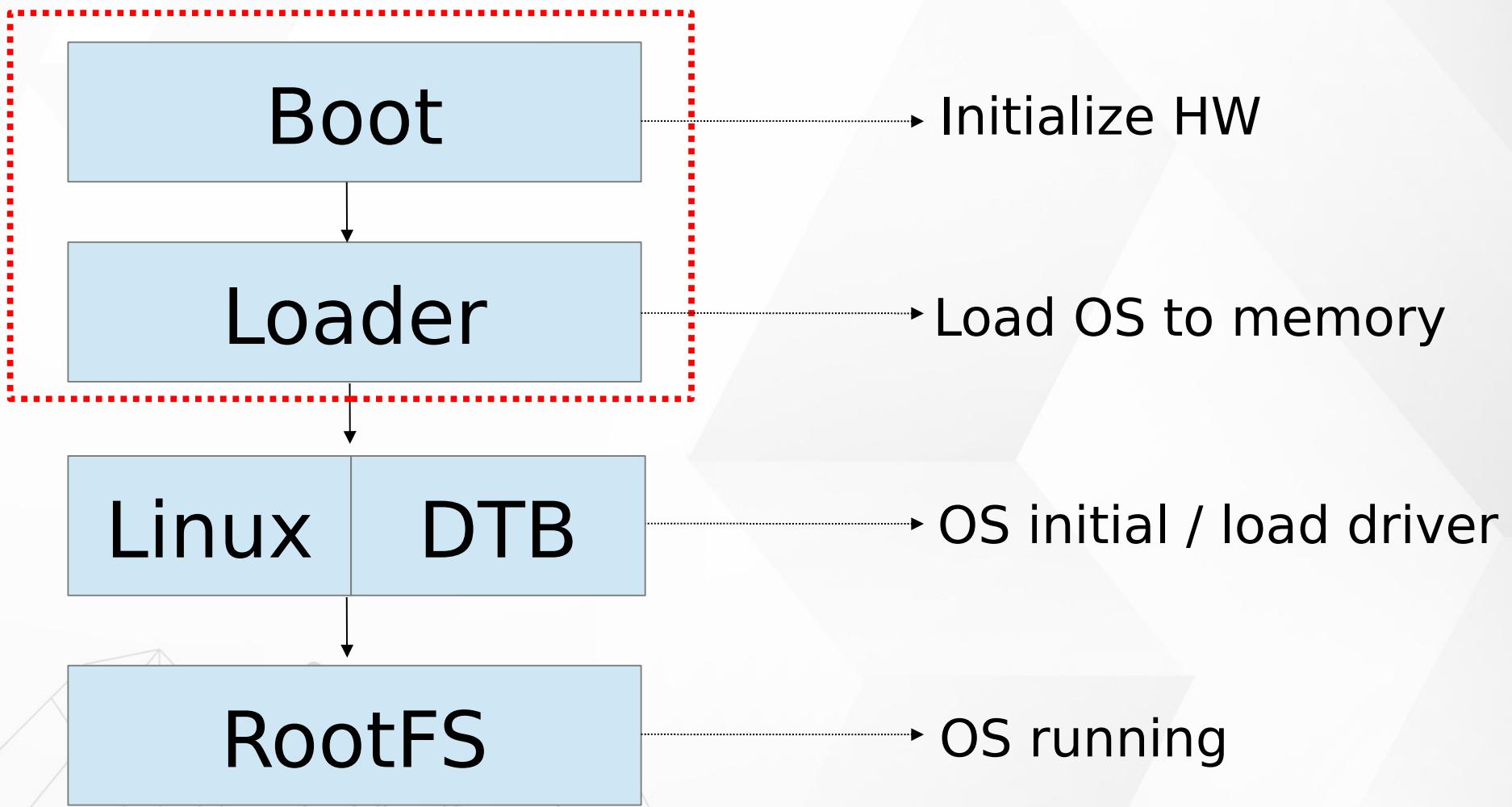
Embedded Linux is the usage of the
Linux kernel and various
open-source components in
embedded systems
(from Free Electrons)



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 System Booting





Embedded Linux System Software components

- Cross-compilation toolchain
- Bootloader
- Linux Kernel, DeviceTree
- Rootfs
- C library
- Libraries and applications
- BSP (Board Support Package)



Develop Environment



Develop Environment

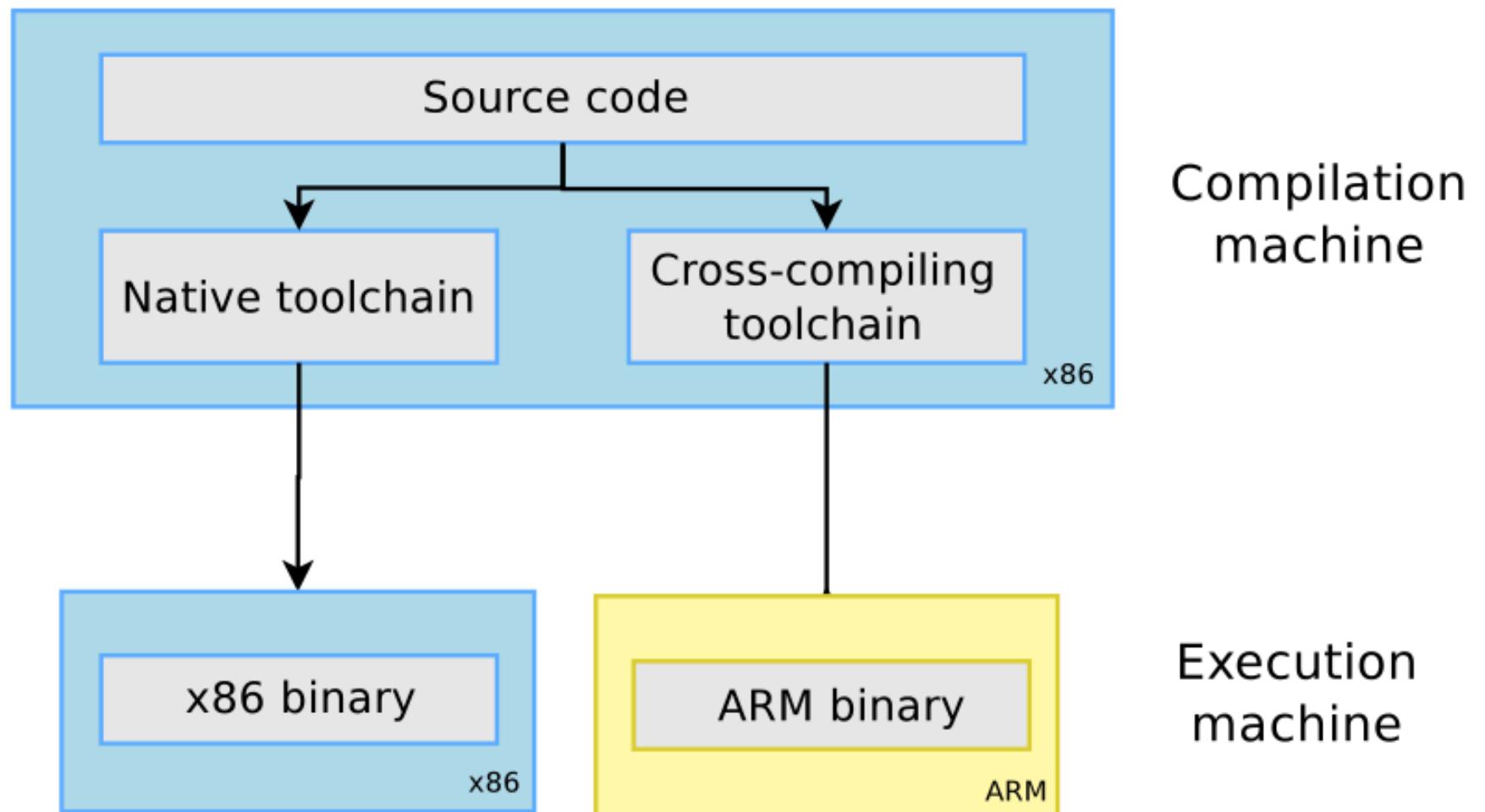
- Host PC
- Toolchain
- Target EVB (NanoPi M4)
- BSP



BSP

- Board Support Package
- From chip vendor
 - Distribution
 - Bootloader
 - Linux kernel
 - Device driver
 - Rootfs

Cross Compilation toolchain





Just do it!

- Understand NanoPi-M4 EVB
- Build develop environment
 - Terminal Setting
 - Gtkterm, minicom ..
 - Prepare NanoPi-M4 BSP
 - U-boot, Linux kernel, RootFS
 - Setting toolchain
 - Build Network Environment



Setup References - 1

RK3399 環境設定 SDCARD and Toolchain

- <https://slashembeddedlinux.blogspot.com/p/rk3399-develop.html>

RK3399 環境設定 – 網路

- <https://slashembeddedlinux.blogspot.com/p/tmp.html>

Debug Port 硬體設定

- <https://slashembeddedlinux.blogspot.com/p/debug-port.html>

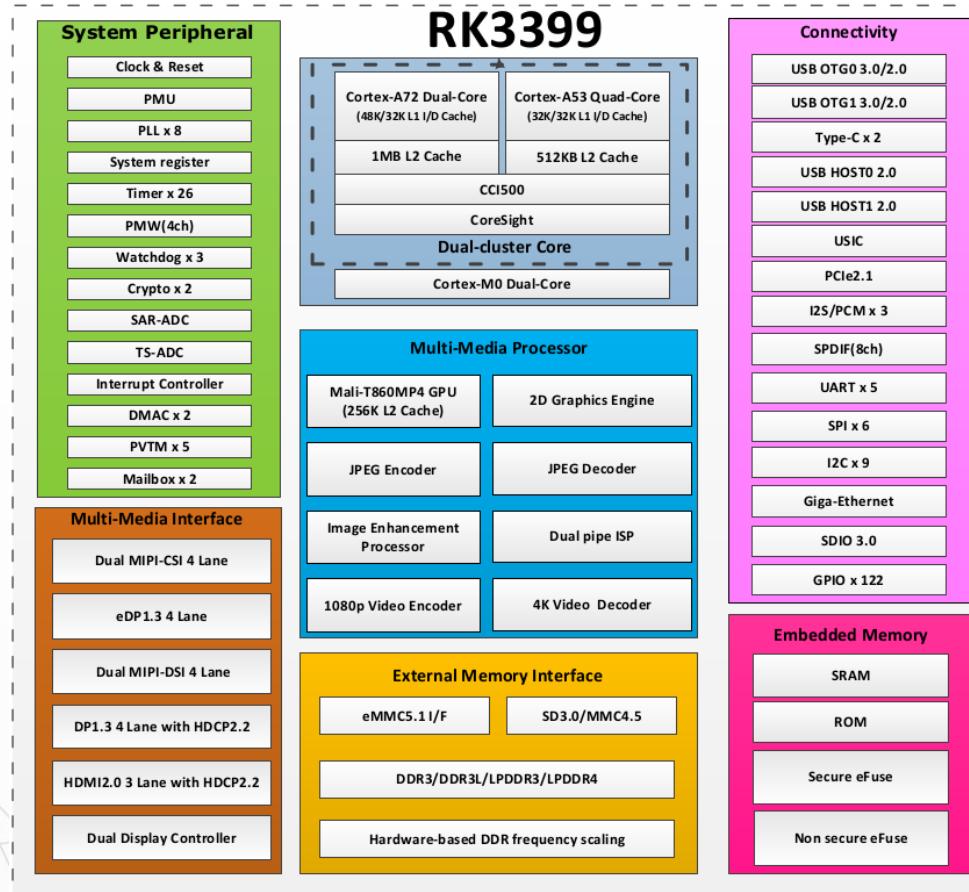


Setup References - 2

- Build U-boot for RK3399
 - \$ cd u-boot-rockchip
 - \$ make nanopi-m4-rk3399_slash_defconfig
- Build Linux Kernel for RK3399
 - \$ cd rockchip-rk3399-nanopi-m4
 - \$ make nanopi4_linux_defconfig

Embedded Linux Arch

SOC RK3399



http://wiki.friendlyarm.com/wiki/index.php/NanoPi_M4#Diagram.2C_Layout_and_Dimension

Embedded Linux System Booting

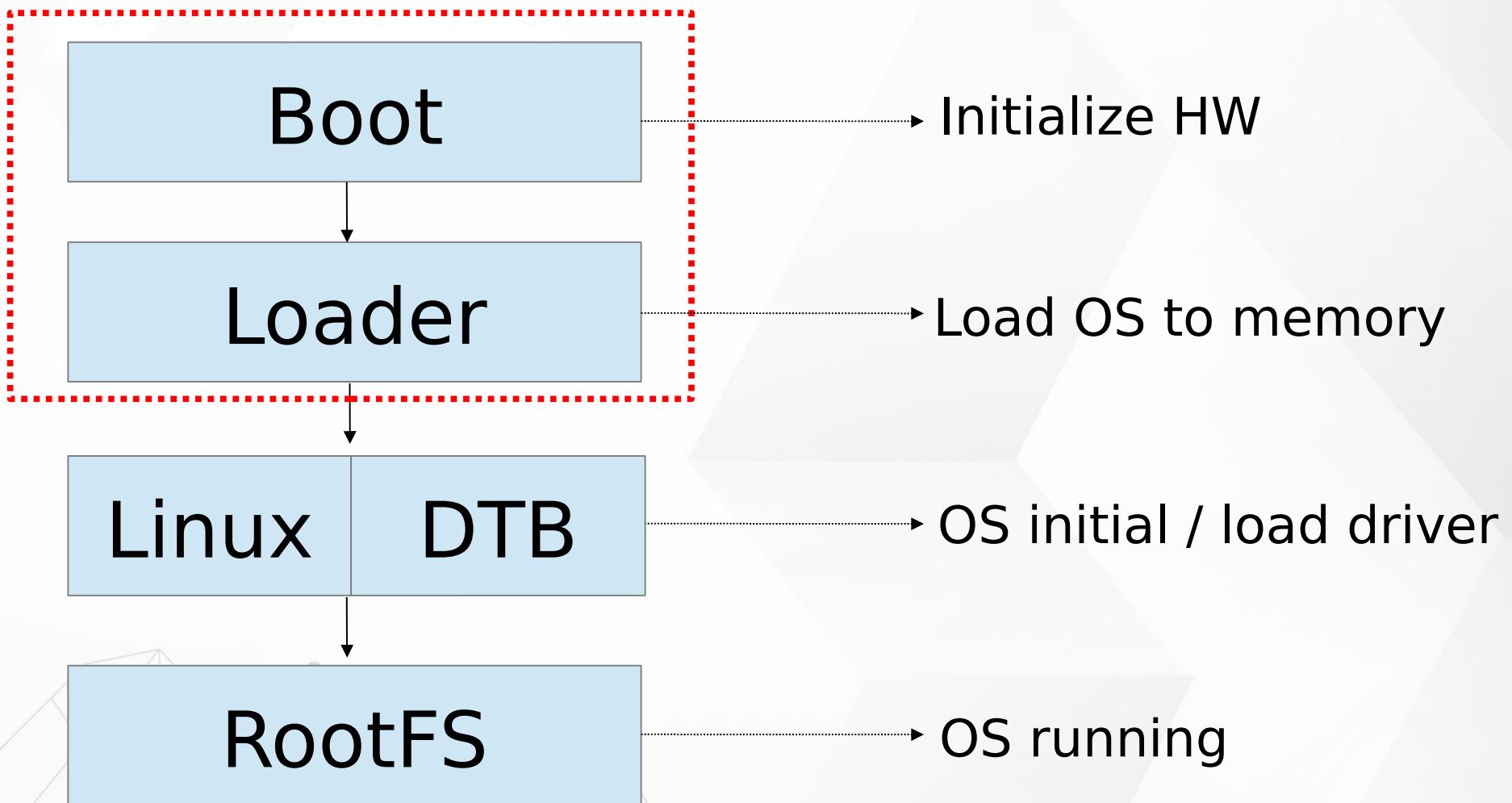




Image Partition

partmap.txt – Image layout in SD card

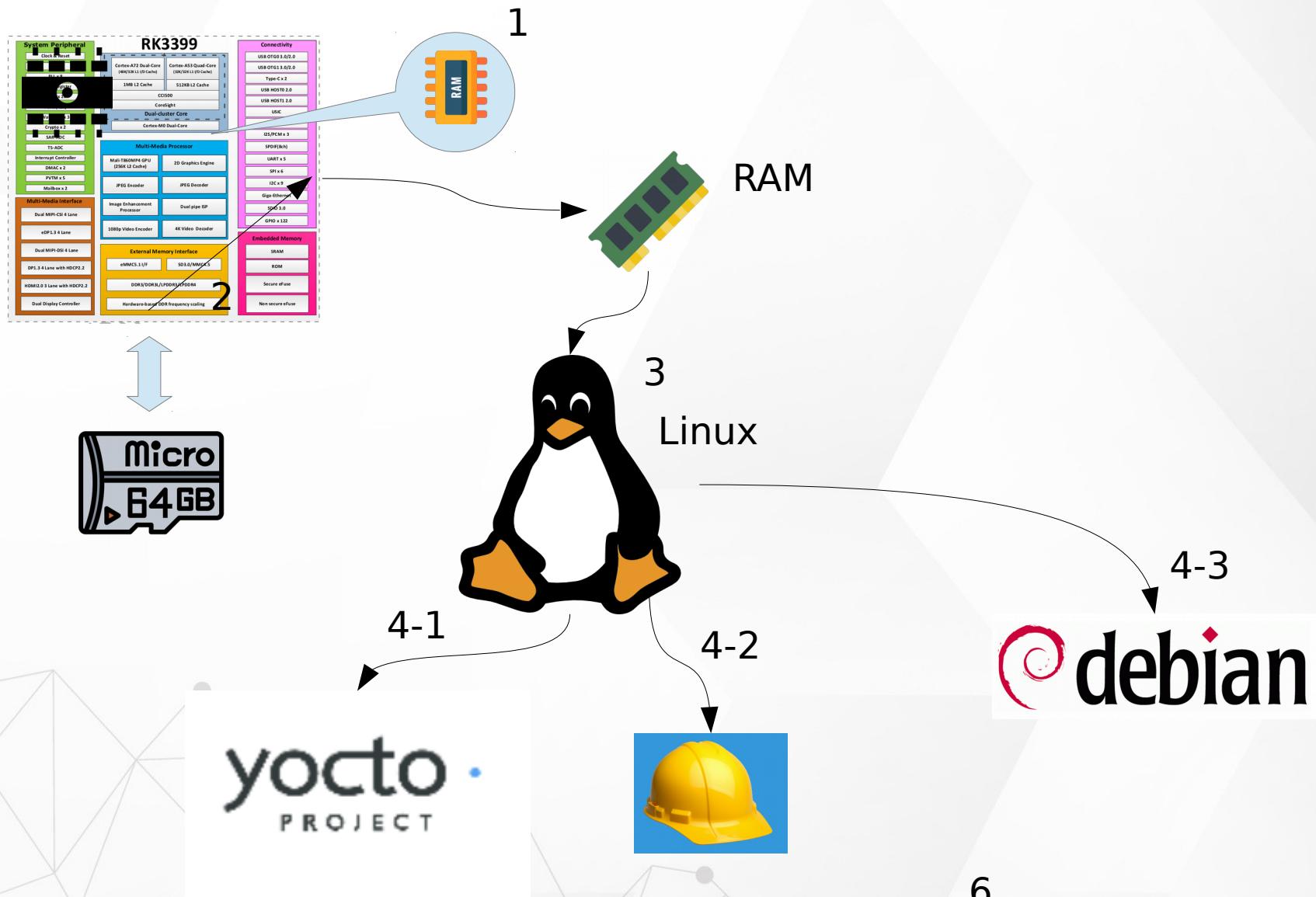
Loader	idbloader.img	0x8000,0x280000
Environment		0x3F8000,0x8000
Parameter	param4sd.txt	0x400000,0x0400000
u-boot	Uboot.img	0x800000,0x0400000
Trust	Trust.img	0xC00000,0x0400000
Misc		0x1000000,0x0400000
Resource	Resource.img	0x1400000,0x0C00000
Linux kernel	Linux kernel	0x2000000,0x2000000
Boot	Boot.img	0x4000000,0x2000000
RootFS	rootfs.img	0x6000000,RootFS Size
User data		

Image Partition

Open source version

Loader	idbloader.img	0x8000,0x280000
Environment		
Parameter		
u-boot	Uboot.img	0x800000,0x0400000
Trust	Trust.img	0xC00000,0x0400000
Misc		
DTB	Nanopim4-rev21.dtb	Boot folder of SD CARD
Linux kernel	Image	Boot folder of SD CARD
Boot		
RootFS	rootfs.tar.bz2	Root of SD CARD
User data		

System Start Up





Boot

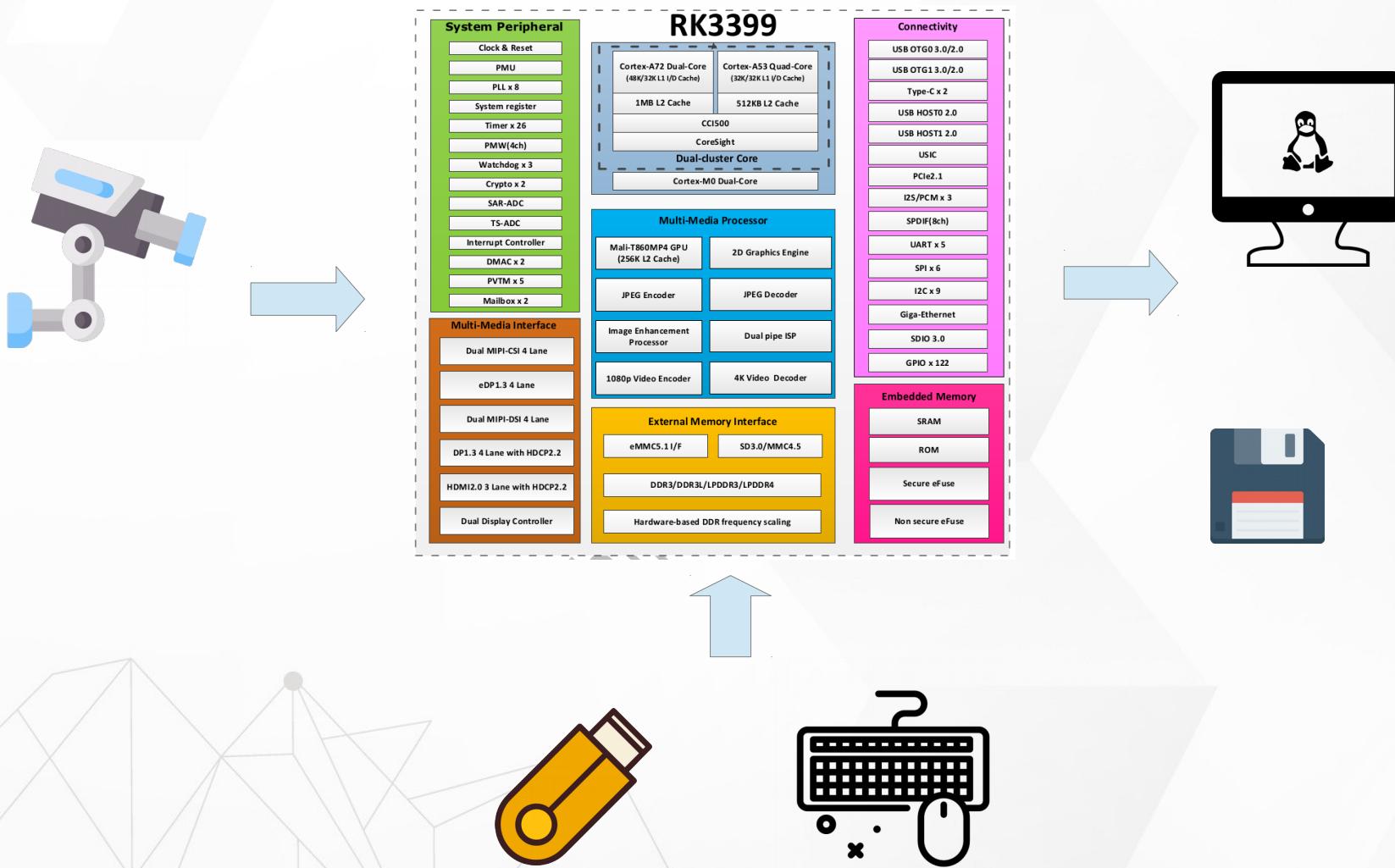
- ▶ Power On BootROM code (work in cache)
 - ▶ Load BL1
- ▶ BL1 (work in cache - IDB_Loader)
 - ▶ Initial simple exception vectors, PLL (clock)
 - ▶ Initial Multi-CPU
 - ▶ Load BL2
- ▶ BL2 (work in cache)
 - ▶ Initial DDR memory
 - ▶ Initial C environment (stack, heap,)
 - ▶ Load BL31



Boot

- ▶ BL31 (work in DDR)
 - ▶ Initial exception vectors
 - ▶ Load BL32 (u-boot)
- ▶ BL32 U-boot
 - ▶ Initial storage device
 - ▶ Load Linux Kernel
- ▶ Kernel
 - ▶ kernel/Documentation/arm64/booting.txt
 - ▶ Load RootFS

Embedded Linux System





Embedded Linux System

User land

Video Play

Read Image

Reocrd Image

Tool

C Library

Qt

OpenCV

GLib

Virtual File System (VFS)

Kernel

Linux System Call

Linux Device Driver

Hardware

Disk

Image Sensor

Keyboard

mouse

Panel

10

CH3 Basic Software and 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

- WiFi, Ethernet, Net tool
- Bluetooth
- SSH, SSHFS
- NFS

► Media Software

- gstreamer
- ALSA Tool - aplay, arecord

Software and Tool



» Bus

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



Open Source License

► GNU General Public License

- 只要在一個軟件中使用 (" 使用 " 指類庫引用，修改後的代碼或者衍生代碼)
GPL 協議的產品，則該軟件產品必須也採用 GPL 協議，既必須也是開源和免費。
這就是所謂的 " 傳染性 "

► BSD License

- 基本上使用者可以 " 為所欲為 "，可以自由的使用，修改源代碼，
也可以將修改後的代碼作為開源或者專有軟件再發佈。

► LGPL

- LGPL 是 GPL 的一個為主要為類庫使用設計的開源協議。LGPL 允許商 業軟件
通過類 庫引用 (link) 方式使用 LGPL 類庫而不需要開源商業軟件的代碼。這使得採用
LGPL 協議的開源代碼可以被商業軟件作為類庫引用並發布和銷售。

Develop Tool



Ubuntu Package Management

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



Geany

► You can find a good edit for programing

→ Geany

<https://www.geany.org/>

\$ sudo apt-get install geany

→ Vim

\$ sudo apt-get install vim

→ gedit



Tracking code command

▶ Linux command

▶ Filter :

→ grep -r -n “function name”

▶ Find special file include “String”

→ find -name “*.c” | xargs grep -n “String”



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`



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



GitHub

► <https://github.com/>

The screenshot shows the GitHub sign-up interface. At the top, there's a navigation bar with the GitHub logo, a search bar, and links for Explore, Features, Enterprise, Pricing, Sign up (in a green button), and Sign in. Below the navigation, a large banner features the text "Where software is built" in white. Underneath, it says "Powerful collaboration, code review, and code management for open source and private projects. Public projects are always free." It also mentions "Private plans start at \$7/mo." To the right of the banner, there are three input fields: "Pick a username", "Your email", and "Create a password". Below the password field is a note: "Use at least one lowercase letter, one numeral, and seven characters." A large green "Sign up for GitHub" button is centered below the password field. At the bottom, a small note states: "By clicking 'Sign up for GitHub', you agree to our [terms of service](#) and [privacy policy](#). We will send you account related emails occasionally."



Exercise

- ▶ 0. Create an empty Git repository (In local)
\$ git init
- ▶ 1. Clone code to local
\$ git clone https://github.com/xlloss/tiny4412-uboot.git
- ▶ 2. modify something
\$ gedit README
- ▶ 3. check source status
\$ git status
- ▶ 4. use "git add <file>..." to update what will be committed
\$ git add ./README
- ▶ 5. check status again
\$ git status



Exercise

- ▶ 6. commit code to local repository
 \$ git commit -a "test"
 Or \$ git commit
- ▶ 7. check log
 \$ git log
- ▶ 8. check how many branch in local repository
 \$ git branch
- ▶ 9. create new branch in local repository
 \$ git branch "new_branch_name"
 \$ git branch cadtc_uboot



Exercise

- ▶ 10. check out to new branch
 \$ git checkout "branch_name"
 \$ git checkout slash_uboot
- ▶ 11. check branch again
 \$ git branch
- ▶ 12 . push log branch to remote
 \$ git push origin slash-uboot
- ▶ 13. check remote branch status
 \$ git branch origin/ and push tab x2



Exercise

- ▶ reset your code, but modify code still live
 \$ git reset commit hash coed
- ▶ Hard reset your code, all modify code will discard
 \$ git reset - --hard hash coed
- ▶ Check log
 \$ git log
 \$ git show
- ▶ Download objects and refs from another repository
 \$ git fetch [--all]



BASIC Git Command

- ▶ init Create an empty Git repository
- ▶ add Add file contents to the index
- ▶ branch List, create, or delete branches
- ▶ checkout Checkout a branch or paths to the working tree
- ▶ clone Clone a repository into a new directory
- ▶ commit Record changes to the repository



BASIC Git Command

- ▶ diff Show changes between commits, commit and working tree, etc
- ▶ rm Remove files from the working tree and from the index
- ▶ pull Fetch from and merge with another repository or a local branch
- ▶ push Update remote refs along with associated objects
- ▶ reset Reset current HEAD to the specified state
- ▶ cherry-pick apply changes introduced by some existing commits

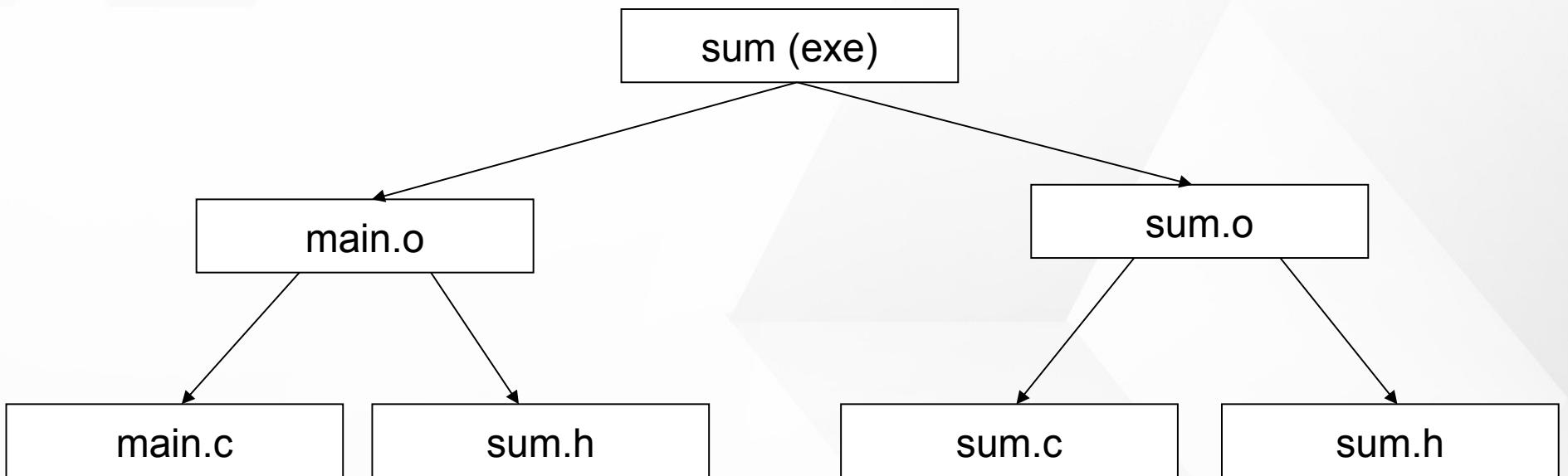
Build Code Tool



Makefile

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

Makefile



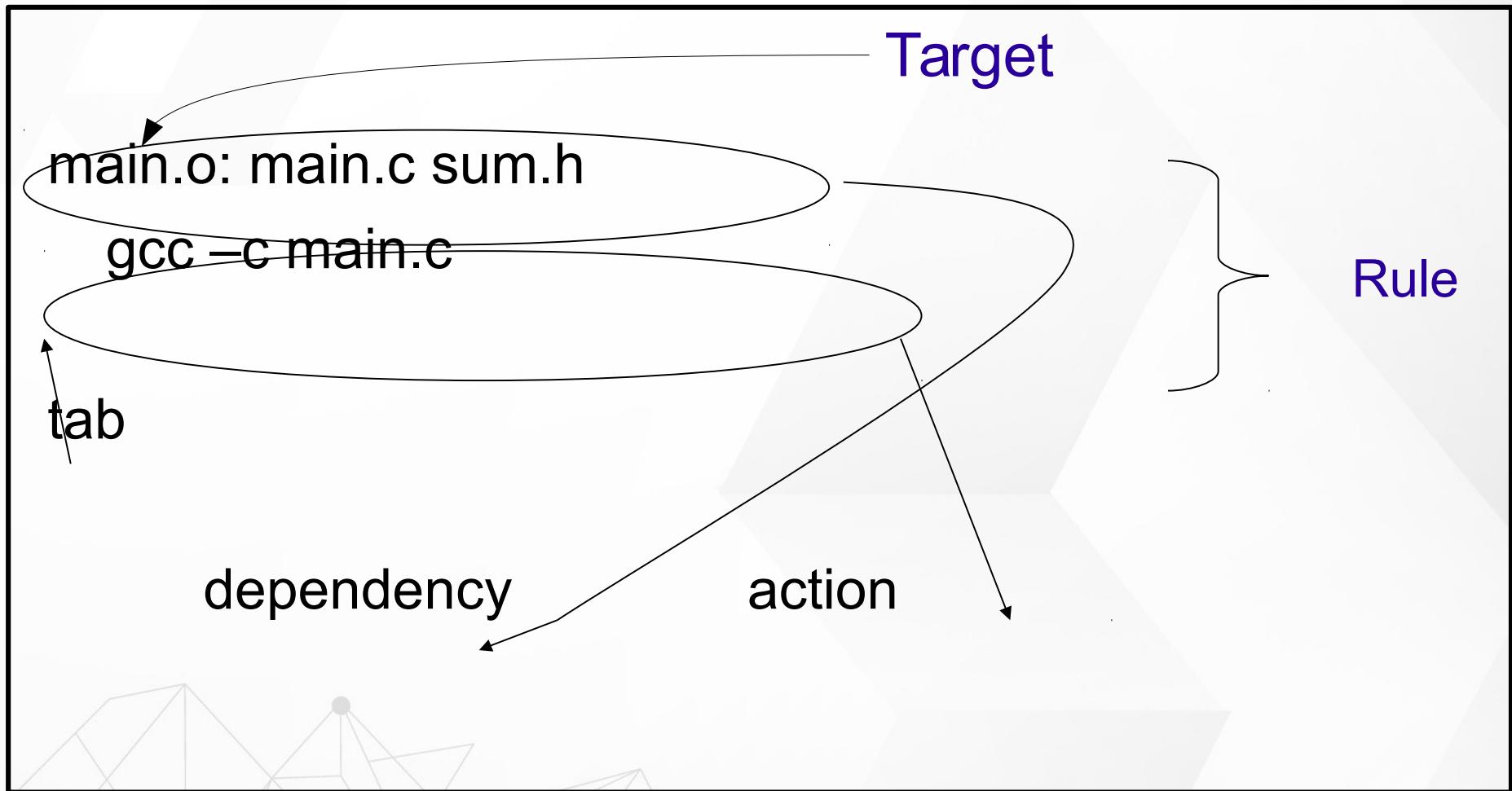
Makefile

```
sum: main.o sum.o  
    gcc -o sum main.o sum.o
```

```
main.o: main.c sum.h  
    gcc -c main.c
```

```
sum.o: sum.c sum.h  
    gcc -c sum.c
```

Rule syntax





Assignment Operators

- ▶ = Defines a recursively expanded variable
- ▶ := Defines a simply expanded variable
- ▶ += Also called the append operator. Appends more characters to the existing value of a variable
- ▶ ?= The conditional assignment operator. Assigns a value to a variable, but only if the variable has no value, otherwise keep original value



The Automatic Variables

- ▶ \$@ The target filename.
- ▶ \$< The first prerequisite.
- ▶ \$^ The list of prerequisites, excluding duplicate elements.



The Automatic Variables

```
CC = gcc
CFLAGS = -Wall -g -std=c99
LDFLAGS = -lm

circle : circle.o circulararea.o
        $(CC) $(LDFLAGS) -o circle circle.o
circulararea.o

circle.o : circle.c
        $(CC) $(CFLAGS) -o circle.o -c circle.c

circulararea.o: circulararea.c
        $(CC) $(CFLAGS) -o circulararea.o -c
circulararea.c
```



The Automatic Variables

```
CC = gcc
```

```
CFLAGS = -Wall -g -std=c99
```

```
LDFLAGS = -lm
```

```
circle : circle.o circulararea.o  
        $(CC) $(LDFLAGS) -o $@ $^
```

```
circle.o : circle.c  
        $(CC) $(CFLAGS) -o $@ -c $<
```

```
circulararea.o: circulararea.c  
        $(CC) $(CFLAGS) -o $@ -c $<
```



Phony Targets

► .PHONY

- Any targets that are prerequisites of .PHONY are always treated as out of date.

```
#Naming our phony targets
.PHONY: clean install

#Removing the executable and the object files
clean:
    rm sample main.o example.o
    echo clean: make complete

#Installing the final product
install:
    cp sample /usr/local
    echo install: make complete
```



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



Exercise

Media Tool



Gstreamer

▶ Gstreamer

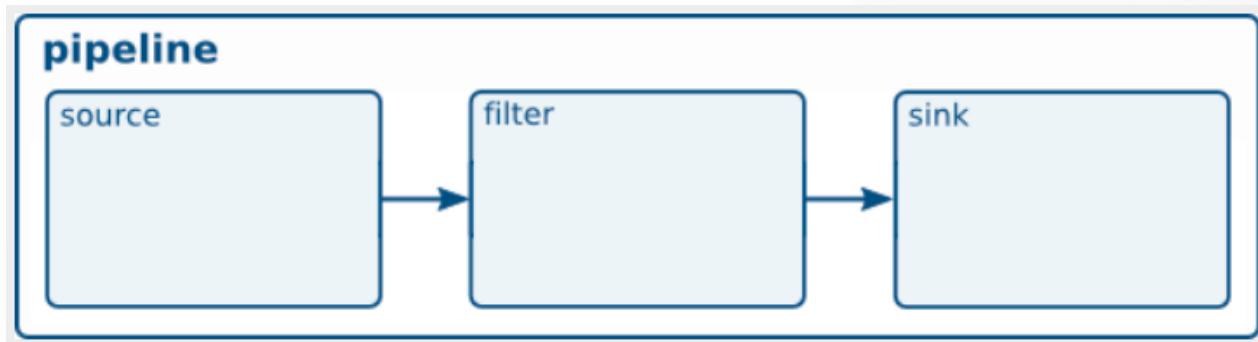
<https://gstreamer.freedesktop.org/>

```
gst-launch-1.0 playbin uri=file:///oem/SampleVideo_1280x720_5mb.mp4
```



Gststreamer

Walkthrough



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



Gststreamer

▶ Play a H.264 video

```
gst-launch-1.0 filesrc location=/oem/200frames_count.h264 ! \
decodebin name=dec ! \
videoconvert ! \
kmssink
```





Gstreamer

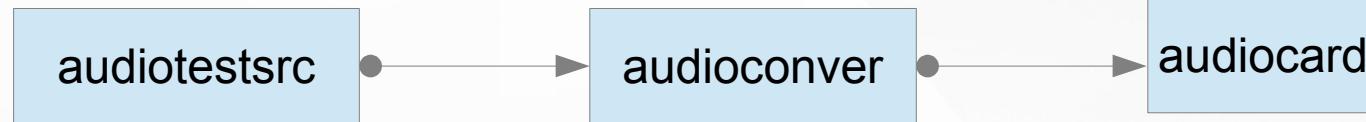
▶ Play a test

```
gst-launch-1.0 audiotestsrc ! audioconvert ! alsasink device-name=realtekrt5651co
```

Source

Format Convert

Sink

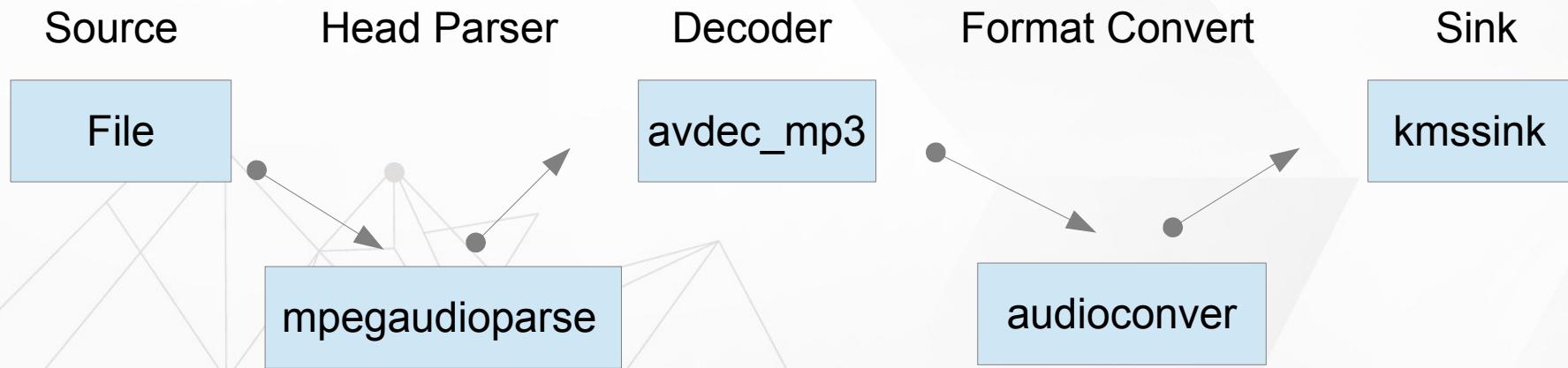




Gststreamer

▶ Play a MP3

```
gst-launch-1.0 filesrc location="oem/piano2-CoolEdit.mp3" ! \
mpegaudioparse ! \
avdec_mp3 ! \
audioconvert ! \
alsasink device=hw:0
```





Exercise



ALSA Tool

▶ ALSA Utile

▶ 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 -Dhw:0,0 /oem/MrBig-ToBeWithYou.wav
- aplay -Dhw:1,0 /oem/MrBig-ToBeWithYou.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%



Exercise

WiFi and Network



Basic Network Tool

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



WPA/WPA2

➤ iw → Finding the WiFi device name

➤ Scan SSID

➤ wpa_supplicant

➤ For connecting to a WPA/WPA2 network



WPA/WPA2 - Device

\$ iw dev

```
[root@rk3399:/]# iw dev  
phy#0  
      Interface wlan0  
            ifindex 3  
            wdev 0x1  
            addr cc:4b:73:92:50:6a  
            type managed  
            txpower 31.00 dBm
```

\$ ls /sys/class/net

```
[root@rk3399:/]# ls /sys/class/net/  
eth0  lo  wlan0  
[root@rk3399:/]# █
```



WPA/WPA2 - iw

\$ iw wlan0 scan

```
BSS 0c:9d:92:d9:e7:78 (on wlan0)
    TSF: 7656316992 usec (0d, 02:07:36)
    freq: 2462
    beacon interval: 100 TUs
    capability: ESS Privacy ShortPreamble ShortSlotTime RadioMeasure (0x1431)
    signal: -73.00 dBm
    last seen: 2 ms ago
    SSID: kevin asus
    Supported rates: 1.0* 2.0* 5.5* 11.0*
    DS Parameter set: channel 11
    ERP: Use_Protection
    Extended supported rates: 6.0 9.0 12.0 18.0 24.0 36.0 48.0 54.0
    RSN:
        * Version: 1
        * Group cipher: CCMP
        * Pairwise ciphers: CCMP
        * Authentication suites: PSK
        * Capabilities: 16-PTKSA-RC 1-GTKSA-RC (0x000c)
    HT capabilities:
        Capabilities: 0x12d
            RX LDPC
            HT20
            SM Power Save disabled
            RX HT20 SGI
            RX STBC 1-stream
            Max AMSDU length: 3839 bytes
            No DSSS/CCK HT40
        Maximum RX AMPDU length 65535 bytes (exponent: 0x003)
        Minimum RX AMPDU time spacing: 2 usec (0x04)
        HT RX MCS rate indexes supported: 0-7
        HT TX MCS rate indexes are undefined
```



WPA/WPA2 – SSID and PASSWD

```
$ wpa_passphrase "SSID" > /etc/wpa_supplicant.conf
```

```
ctrl_interface=/var/run/wpa_supplicant
update_config=1
ap_scan=1

network={
    ssid="ssid"
    #psk="testtest"
    psk="password"
}
```



WPA/WPA2 - Connect

```
$ wpa_supplicant -B -D wext -i wlan0 -c /etc/wpa_supplicant.conf
```

```
[ 29.752634] CFG80211-ERROR) wl_escan_handler : escan is not ready ndev ffffffc0782d1000
[ 29.767372] wl_iw_set_essid: WLC_DISASSOC
[ 29.772806] Setting the D11auth 1
[ 29.788792] wl_iw_set_freq: chan=4
[ 29.794068] wl_iw_set_wap: WLC_REASSOC failed (-22).
[ 29.835315] Connecting with 62:07:b7:ed:02:4d channel (4) ssid "REASSON", len (6)
[ 29.835315]
[ 29.908754] wl_iw_event: Link UP with 62:07:b7:ed:02:4d
[ 29.914341] wl_bss_connect_done succeeded with 62:07:b7:ed:02:4d
[ 29.921748] wl_bss_connect_done succeeded with 62:07:b7:ed:02:4d
```



WPA/WPA2 - DHCP

```
$ udhcpc -i wlan0
```

```
[root@rk3399:/]# udhcpc -i wlan0
udhcpc: started, v1.27.2
udhcpc: sending discover
udhcpc: sending select for 192.168.43.214
udhcpc: lease of 192.168.43.214 obtained, lease time 3599
deleting routers
adding dns 192.168.43.12
[root@rk3399:/]# █
```



WPA/WPA2 - IP

\$ ifconfig wlan0

```
[root@rk3399:/]# ifconfig wlan0
wlan0      Link encap:Ethernet  HWaddr CC:4B:73:92:50:6A
           inet  addr:192.168.43.214  Bcast:192.168.43.255  Mask:255.255.255.0
           inet6 addr: fe80::7e7:9ca:dc48:71ab/64 Scope:Link
                  UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
                  RX packets:6 errors:0 dropped:0 overruns:0 frame:0
                  TX packets:40 errors:0 dropped:0 overruns:0 carrier:0
                  collisions:0 txqueuelen:1000
                  RX bytes:1312 (1.2 KiB)   TX bytes:4477 (4.3 KiB)

[root@rk3399:/]#
```



WPA/WPA2 - Ping

\$ Ping 8.8.8.8 (Google)

```
[root@rk3399:/]# ping -I wlan0 8.8.8.8
PING 8.8.8.8 (8.8.8.8) from 192.168.43.214 wlan0: 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=117 time=49.3 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=117 time=44.4 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=117 time=37.5 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=117 time=35.3 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=117 time=63.8 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=117 time=29.5 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=117 time=40.3 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=117 time=45.8 ms
64 bytes from 8.8.8.8: icmp_seq=9 ttl=117 time=53.0 ms
64 bytes from 8.8.8.8: icmp_seq=10 ttl=117 time=35.6 ms
64 bytes from 8.8.8.8: icmp_seq=11 ttl=117 time=33.8 ms
^C
--- 8.8.8.8 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10018ms
rtt min/avg/max/mdev = 29.590/42.629/63.804/9.521 ms
[root@rk3399:/]#
```



Exercise

- ▶ Try to use iper3 to test Ethernet
- ▶ Use dhcpc to get dynamic IP from DHCP server
- ▶ Connect WIFI route with WPA/WPA2 tool

SSH



SSH

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





SSH

▶ SSH Client

▶ # sudo apt-get install ssh

▶ <https://slashembeddedlinux.blogspot.com/p/tmp.html>



SSHFS

NFS

NFS

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





NFS

▶ <https://slashembeddedlinux.blogspot.com/p/tmp.html>

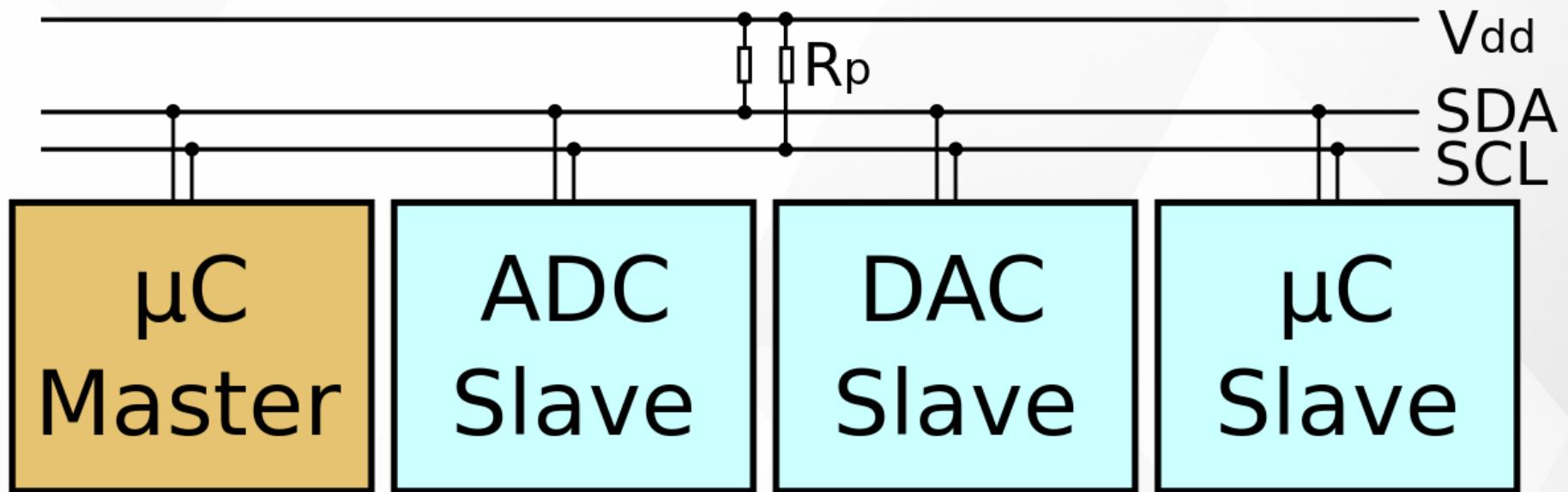


Exercise

- ▶ Try to SSH remote to target board
- ▶ Try to mount remote folder from target board to host

I2C Tool

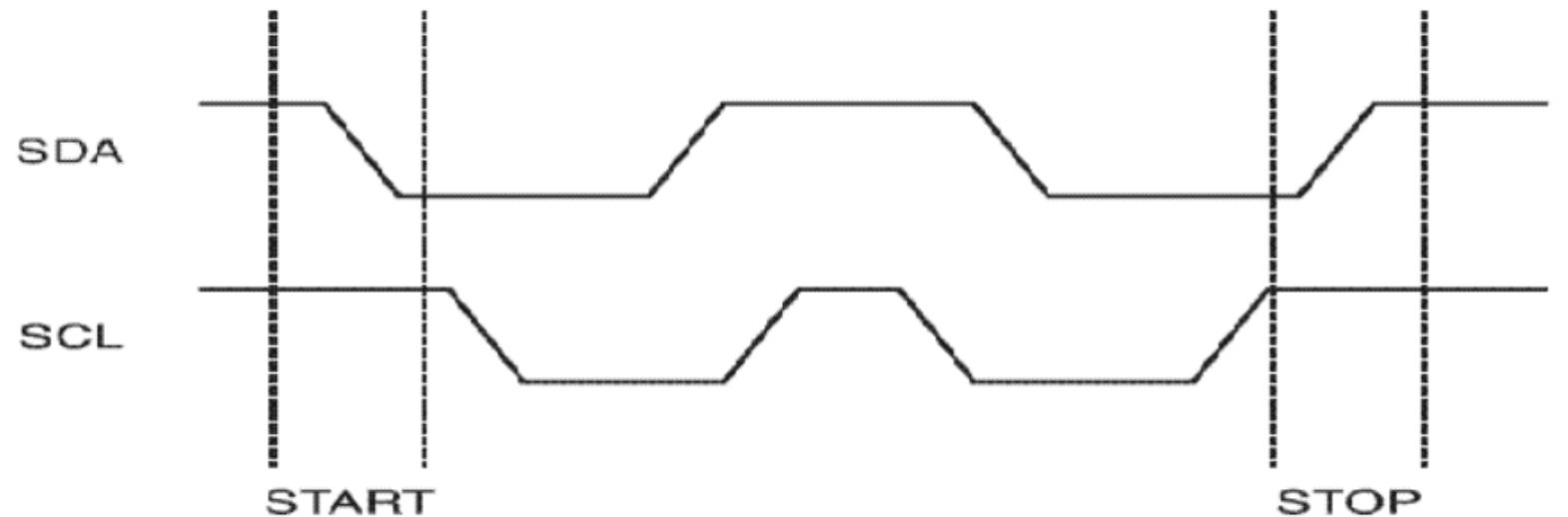
I2C Driver



I2C protocol

► Serial bus

- SDA data line
- SCL clock line



I2C protocol

► Write

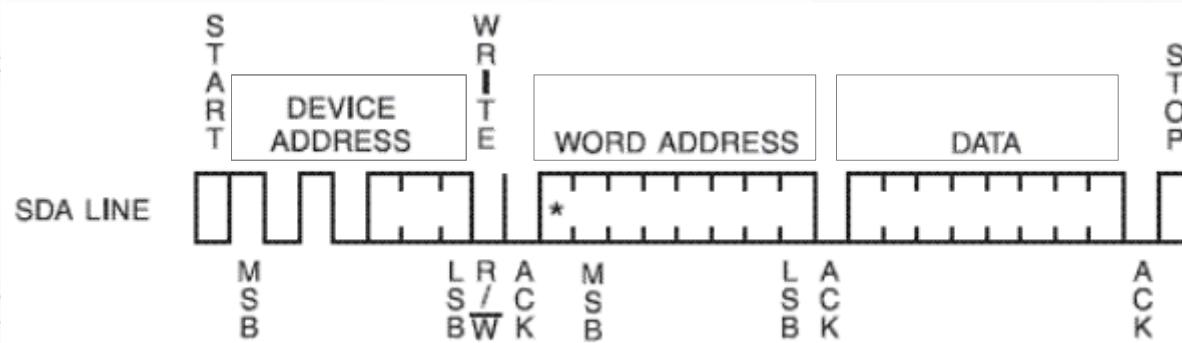
► byte write

► page write

► Device address

► Read/write bit : 0

► ACK



I2C protocol

➤ Read

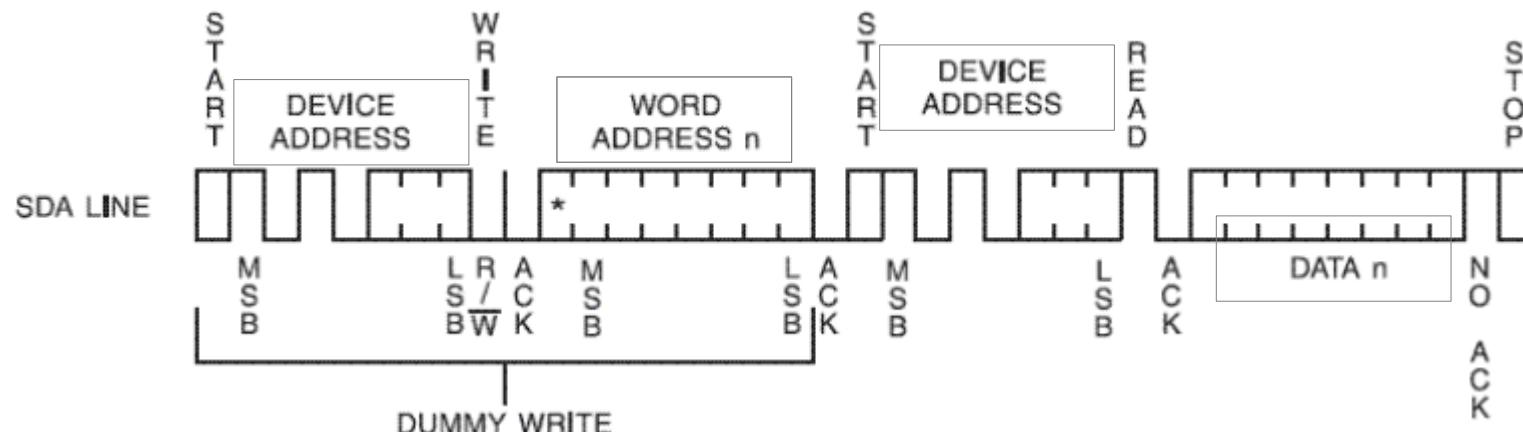
➤ byte read

➤ page read

➤ Device address

➤ Read/write bit : 1

➤ ACK





I2C Dev Interface

- » i2c tool

- » i2cset, i2cget, i2cdump

- » i2cdetect -l

- » /dev/i2c-x

- » /dev/i2c-0, /dev/i2c-1, /dev/i2c-2 ...

- » /sys/class/i2c-dev/

- » i2c-0 i2c-1 i2c-2 i2c-3 i2c-7 i2c-8 ...



I2C Dev Interface

- ▶ Documentation/i2c/dev-interface
- ▶ i2c-tools
 - ▶ i2cdump
 - ▶ i2cdetect
 - ▶ i2cget
 - ▶ i2cset

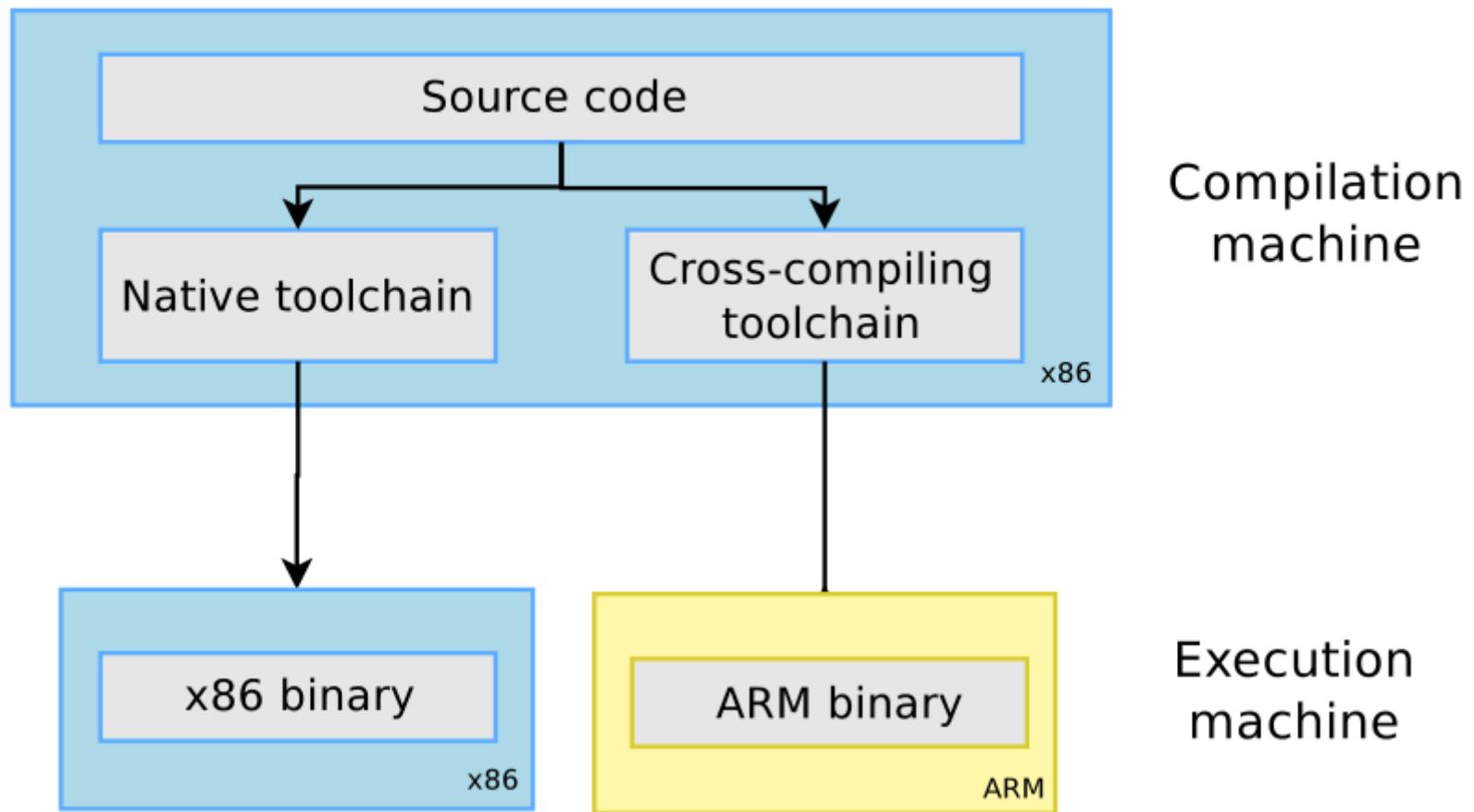


Exercise

- ▶ Try to use I2C tool to send/get i2c command to sensor

CH4 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

- » 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
- » -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.



sysroot

- ▶ The sysroot is the logical root directory for headers and libraries
- ▶ GCC look for head and LD look for library
- ▶ We can assign sysroot locate avoid toolchain change locate
→ `--with-sysroot=<locate>`



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



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
 - `PATH=/${TOOLCHAIN_PATH}/bin/:$PATH`

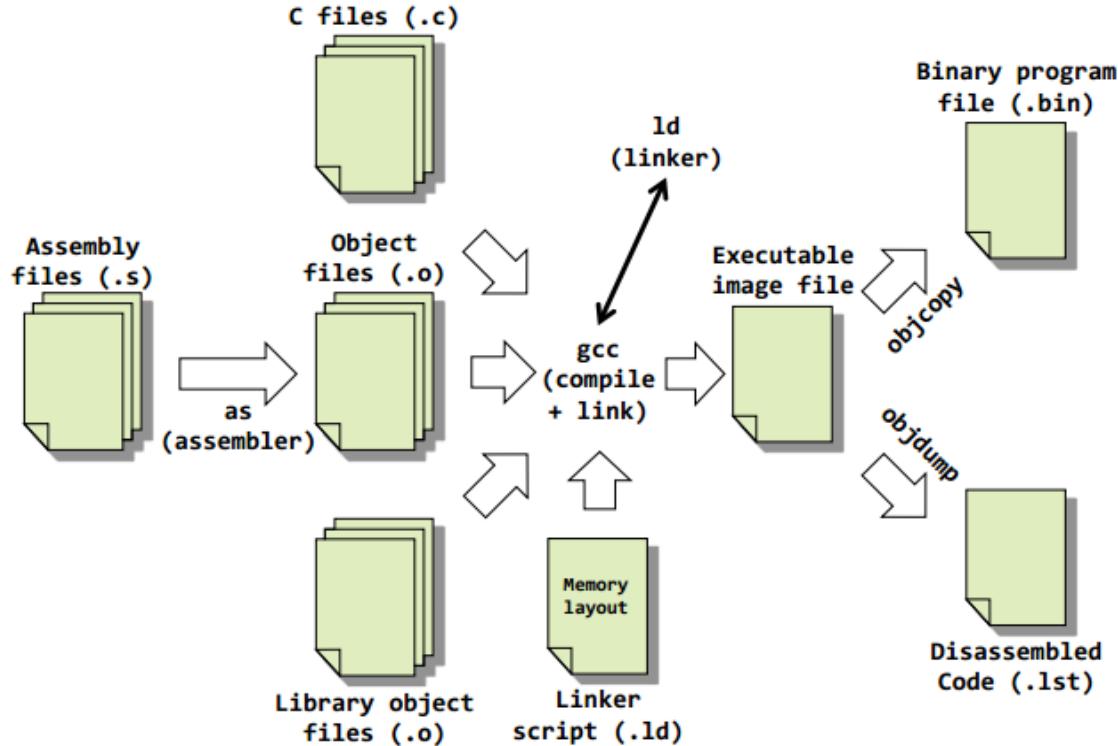
- Compile your applications
 - `PREFIX-gcc -o testme testme.c`

- `PREFIX`

- depends on the toolchain configuration

Compile, Assembler, Linker

Software Development Tools Overview





Tools Descriptions

» C/C++ compiler

» produces ARM machine code object modules

» Assembler

» Translates Assembly Language Source Files Into Machine Language Object modules

» Linker

» Combines object files into a single executable object module



Exercise

- ▶ Download toolchain and install
- ▶ Setup the toolchain environment and check it
- ▶ Browse the toolchain binary locate
- ▶ Use toolchain compile a application

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 -shared -Wl,-soname,libctest.so.1 -o libctest.so.1.0

test1.o test2.o

» 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 ?

Real-name libctest.so.1.0

Soname libctest.so.1 → libctest.so.1.0

Linkname libctest.so → libctest.so.1

Modify

Real-name libctest.so.1.1

Soname libctest.so.1 → libctest.so.1.1

Linkname libctest.so → libctest.so.1

Real-name libctest.so.1.5

Soname libctest.so.1 → libctest.so.1.5

Linkname libctest.so → libctest.so.1

main.c no need to re-compile

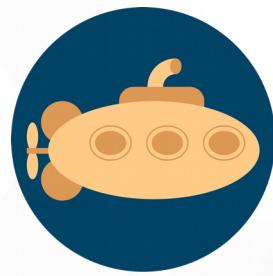
```
gcc -o test main.c -L/library_PATH/ -lctest
```



Exercise

- ▶ Try to know what is soname
- ▶ Create a library and try to let work

<https://tldp.org/HOWTO/Program-Library-HOWTO/shared-libraries.html>



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



All kinds of embedded Linux bootloader

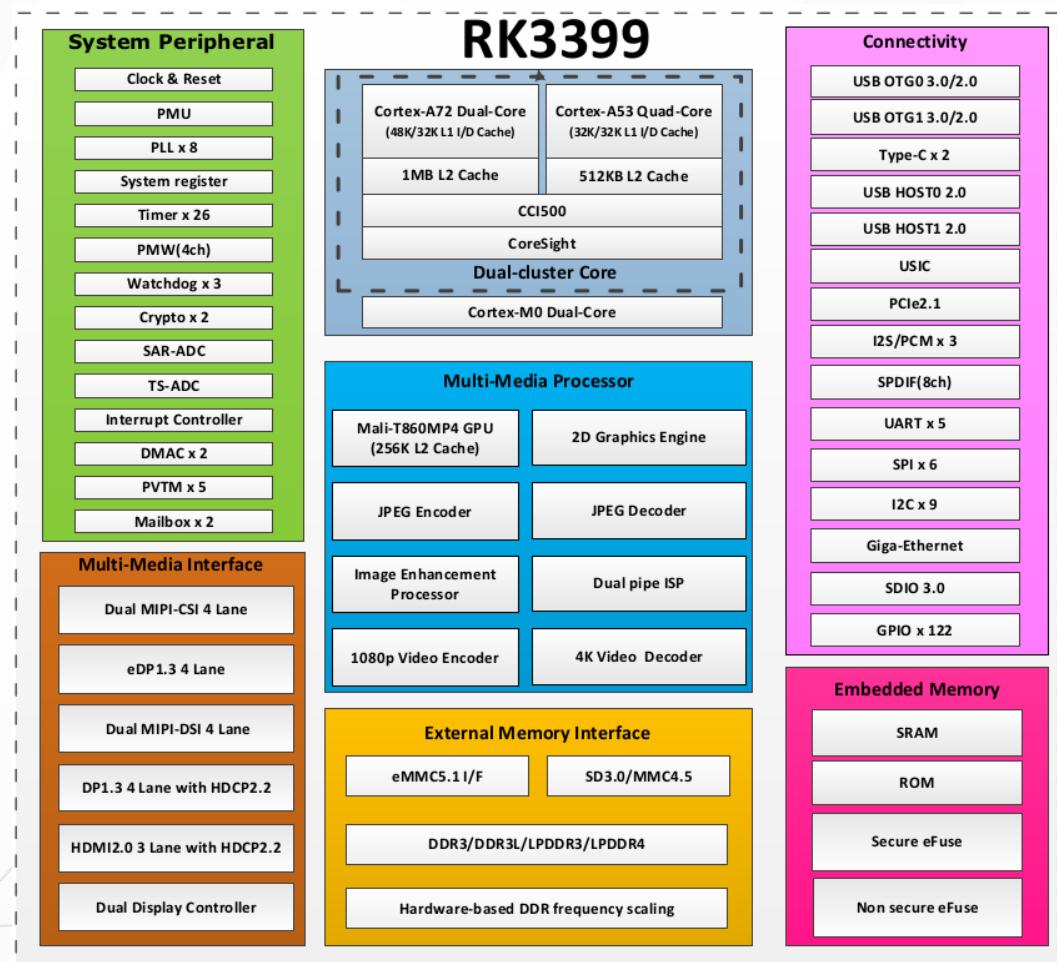
- » U-boot
- » UEFI
- » Redboot
- » Stubby (Linaro) ...
- » Anyway, they are same target
 - » Load and boot OS to RAM from storage



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

Embedded Linux System Booting

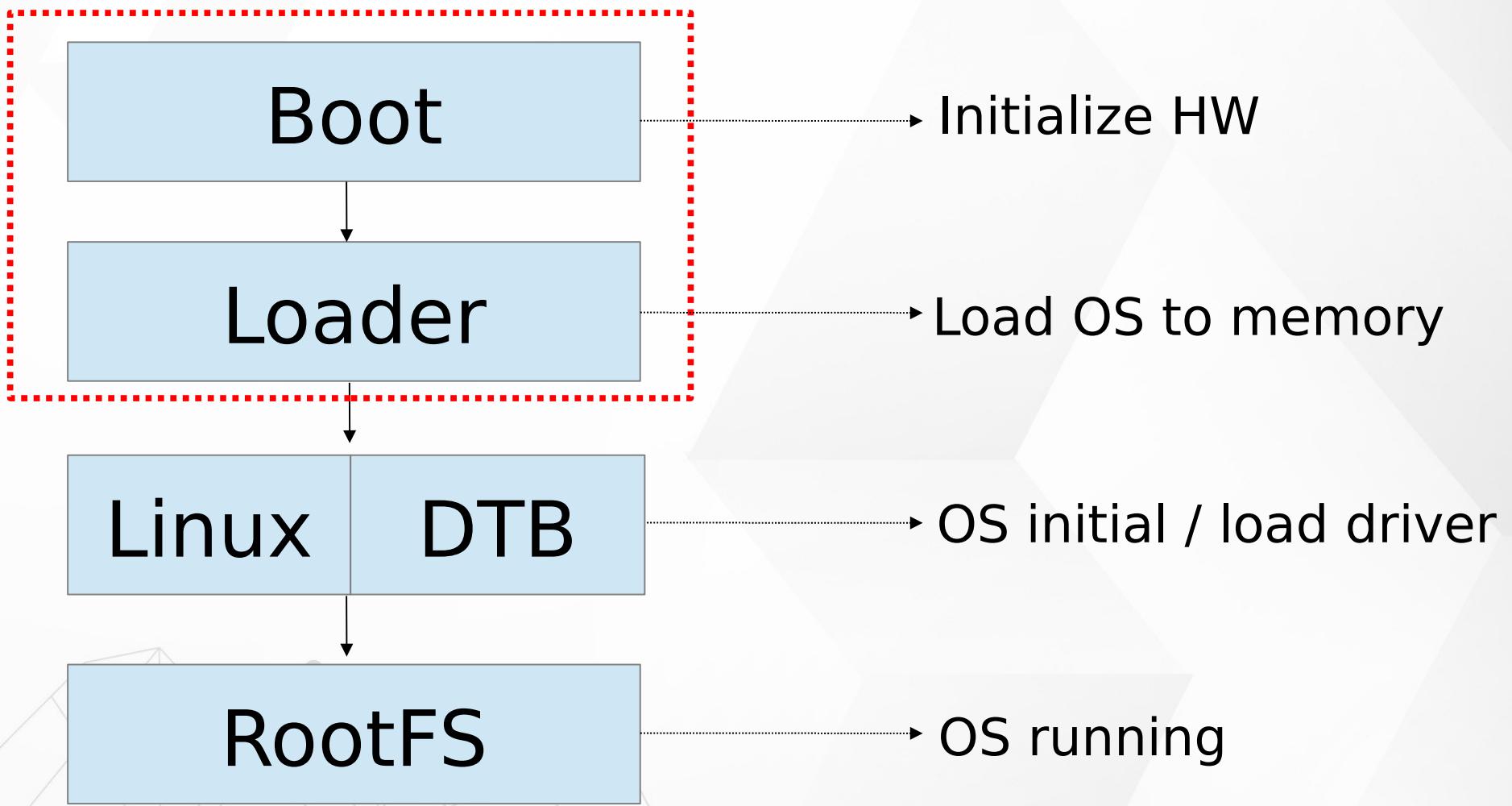




Image Partition

partmap.txt - Image layout in SD card

Loader	idbloader.img	0x8000,0x280000
Environment		0x3F8000,0x8000
Parameter	param4sd.txt	0x400000,0x0400000
u-boot	Uboot.img	0x800000,0x0400000
Trust	Trust.img	0xC00000,0x0400000
Misc		0x1000000,0x0400000
Resource	Resource.img	0x1400000,0x0C00000
Linux kernel	Linux kernel	0x2000000,0x2000000
Boot	Boot.img	0x4000000,0x2000000
RootFS	rootfs.img	0x6000000,RootFS Size
User data		



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
- ▶ Load 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
- » Lib → General public library : CRC32/16, bzlib, ldiv ..
- » Disk → Supply disk driver and partition handling



u-boot directory structure

- Common → Major command and relation environment setting source code
- Api → Implement unrelated hardware code
- nand_spl, onenand_ipl
→ Related nand/onenand flash control
- Example → Standalone application



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



u-boot directory structure about rk3399

- » Common

- » u-boot command related

- » **include/configs/**

- » rk_default_config.h
 - » rk33plat.h



How to build u-boot

- » Clear : **\$ make distclean**
- » Configure : **\$ make nanopi-m4-rk3399_slash_defconfig**
- » Build : **\$ make -j4**
- » Create Image : **\$ tools/loaderimage --pack --uboot ./u-boot-dtb.bin \
 uboot.img**



Configure

include/configs/rk_default_config.h
include/configs/rk33plat.h

```
#define CONFIG_LCD_LOGO
#define CONFIG_LCD_BMP_RLE8
#define CONFIG_CMD_BMP

/* CONFIG_COMPRESS_LOGO_RLE8 or CONFIG_COMPRESS_LOGO_RLE16 */
#undef CONFIG_COMPRESS_LOGO_RLE8
#undef CONFIG_COMPRESS_LOGO_RLE16

#define CONFIG_BMP_16BPP
#define CONFIG_BMP_24BPP
#define CONFIG_BMP_32BPP
#define CONFIG_SYS_WHITE_ON_BLACK
#define LCD_BPP → → → → → LCD_COLOR16

#define CONFIG_LCD_MAX_WIDTH → → → → → 4096
#define CONFIG_LCD_MAX_HEIGHT → → → → → 2048

/* rk lcd size at the end of ddr address */
#define CONFIG_RK_FB_DDREND

#ifndef CONFIG_RK_FB_DDREND
/* support load bmp files for kernel logo */
#define CONFIG_KERNEL_LOGO

/* rk lcd total size = fb size + kernel logo size */
#define CONFIG_RK_LCD_SIZE → → → SZ_32M
#define CONFIG_RK_FB_SIZE → → → SZ_16M
#endif
```



Exercise

- ▶ Compile a u-boot for nanopi-m4
- ▶ Update u-boot for nanopim-m4

U-boot Common Function



Operating U-boot

- Understand and use command
- Understand and modify parameters



Help

» \$ help

→ help mm

» \$?

=> help mm

mm - memory modify (auto-incrementing address)

Usage:

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

=>



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

▶ 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

» fatls

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

```
=> fatls mmc 1:1 boot
<DIR>      4096 .
<DIR>      4096 ..
18385408 Image
104828 rk3399-nanopi4-rev01.dtb
```



FAT

► fatload

- fatload - load binary file from a dos filesystem
- Load binary file 'filename' from 'dev' on 'interface' to address 'addr' from dos filesystem.

```
=> fatload mmc 1:1 0x02080000 boot/Image  
18385408 bytes read in 1639 ms (10.7 MiB/s)
```

```
=> md.1 0x02080000  
02080000: 91005a4d 143c7fff 00080000 00000000 MZ....<.....  
02080010: 01370000 00000000 0000000a 00000000 ..7.....  
02080020: 00000000 00000000 00000000 00000000 .....  
02080030: 00000000 00000000 644d5241 00000040 .....ARMd@...  
02080040: 00004550 0002aa64 00000000 00000000 PE..d.....  
02080050: 00000001 020600a0 1402020b 0136f000 .....6.  
02080060: 00000000 00000000 00f25a60 00001000 .....`Z.....
```



EXT4

» ext4ls

- list files in a directory with ext4
- ext4ls <interface> <dev[:part]> [directory]
 - list files from 'dev' on 'interface' in a 'directory'

```
=> ext4ls mmc 1:1
<DIR>      4096 .
<DIR>      4096 ..
<DIR>    16384 lost+found
<DIR>      4096 bin
            30195 busybox.config
<SYM>        8 data
<DIR>      4096 dev
<DIR>      4096 etc
<DIR>      4096 home
            178 init
```



EXT4

» ext4load

- ext4load - load binary file from a Ext4 filesystem
- ext4load <interface> [<dev[:part]> [addr [filename [bytes [pos]]]]]
load binary file 'filename' from 'dev' on 'interface'
to address 'addr' from ext4 filesystem

```
=> ext4load mmc 1:1 0x02080000 boot/rk3399-nanopi4-rev01.dtb
104828 bytes read in 39 ms (2.6 MiB/s)
=> md 0x02080000
02080000: edfe0dd0 7c990100 38000000 fc710100 .....|....8..q.
02080010: 28000000 11000000 10000000 00000000 ....(.....
02080020: 80270000 c4710100 00000000 00000000 ...'....q.....
02080030: 00000000 00000000 01000000 00000000 .....
02080040: 03000000 27000000 00000000 65697266 .....'....frie
02080050: 796c646e 63656c65 6e616e2c 2d69706f ndlyelec,nanopi-
02080060: 7200346d 636b636f 2c706968 33336b72 m4.rockchip,rk33
02080070: 00003939 03000000 04000000 0b000000 99.....
02080080: 01000000 03000000 04000000 1c000000 .....
02080090: 02000000 03000000 04000000 2b000000 .....+.
020800a0: 02000000 03000000 17000000 37000000 .....7
020800b0: 65697246 796c646e 63656c45 6e614e20 FriendlyElec Nan
020800c0: 2069506f 0000344d 01000000 5f726464 oPi M4.....ddr_
020800d0: 696d6974 0000676e 03000000 14000000 timing.....
020800e0: 00000000 6b636f72 70696863 7264642c ....rockchip,ddr
020800f0: 6d69742d 00676e69 03000000 04000000 -timing.....
```



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

» setenv

→ set environment variables

→ setenv name value

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



saveenv

» setenv

→ save environment variables to persistent storage



Simple Script

```
=> ext4ls mmc 1:1 boot
<DIR>        4096 .
<DIR>        4096 ..
               18385408 Image
                  104828 rk3399-nanopi4-rev01.dtb
<SYM>          7 Image_1
<SYM>          7 Image_2
=> if test readkernel = Image_1;then
> ext4load mmc 1:1 0x02080000 boot/Image_1 ;
> echo read Image_1;
> else
> ext4load mmc 1:1 0x02080000 boot/Image_2;
> echo read Image_2;
> fi
18385408 bytes read in 1625 ms (10.8 MiB/s)
read Image_2
```



run

▶ run

→ run commands in an environment variable

→ run var [...]

```
=> printenv bootcmd
bootcmd=run loadimage; run loadfdt; booti $kernel_addr_r - $fdt_addr_r
=> run bootcmd
18385408 bytes read in 1625 ms (10.8 MiB/s)
104828 bytes read in 26 ms (3.8 MiB/s)
## Flattened Device Tree blob at 01f00000
    Booting using the fdt blob at 0x1f00000
ERROR: reserving fdt memory region failed (addr=0 size=0)
    Loading Device Tree to 00000000f1f1b000, end 00000000f1f3797b ... OK

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 cpacct
[    0.000000] Linux version 4.4.179 (slash@slash-ThinkPad-T420) (gcc vers
08 CST 2021
```



booti

▶ booti

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

```
printenv bootcmd
bootcmd=run loadimage; run loadfdt; booti $kernel_addr_r - $fdt_addr_r
-
=> run bootcmd
18385408 bytes read in 1625 ms (10.8 MiB/s)
104828 bytes read in 26 ms (3.8 MiB/s)
## Flattened Device Tree blob at 01f00000
  Booting using the fdt blob at 0x1f00000
ERROR: reserving fdt memory region failed (addr=0 size=0)
  Loading Device Tree to 00000000f1f1b000, end 00000000f1f3797b ... OK

Starting kernel ...

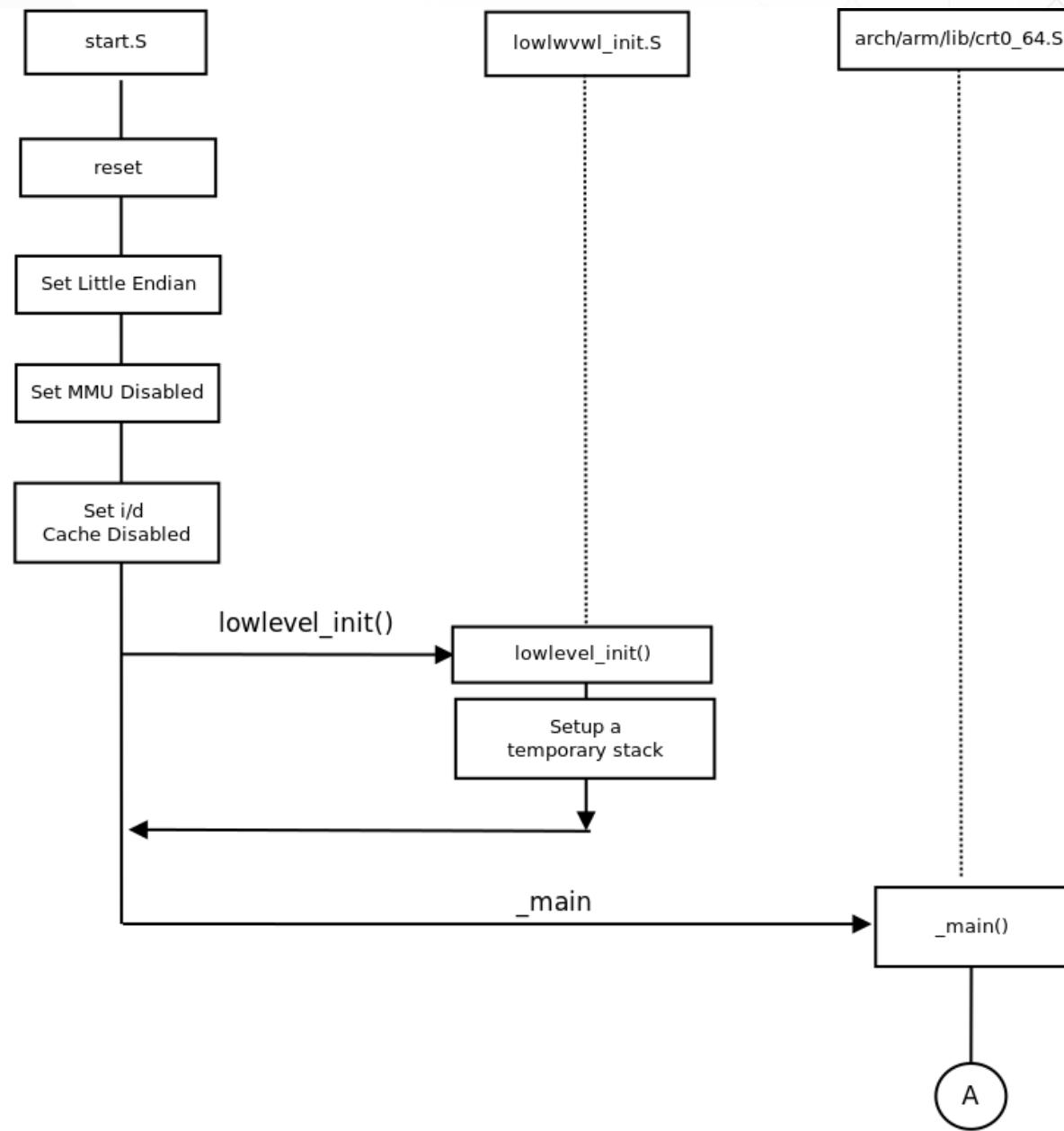
[    0.000000] Booting Linux on physical CPU 0x0
[    0.000000] Initializing cgroup subsys cpuset
```

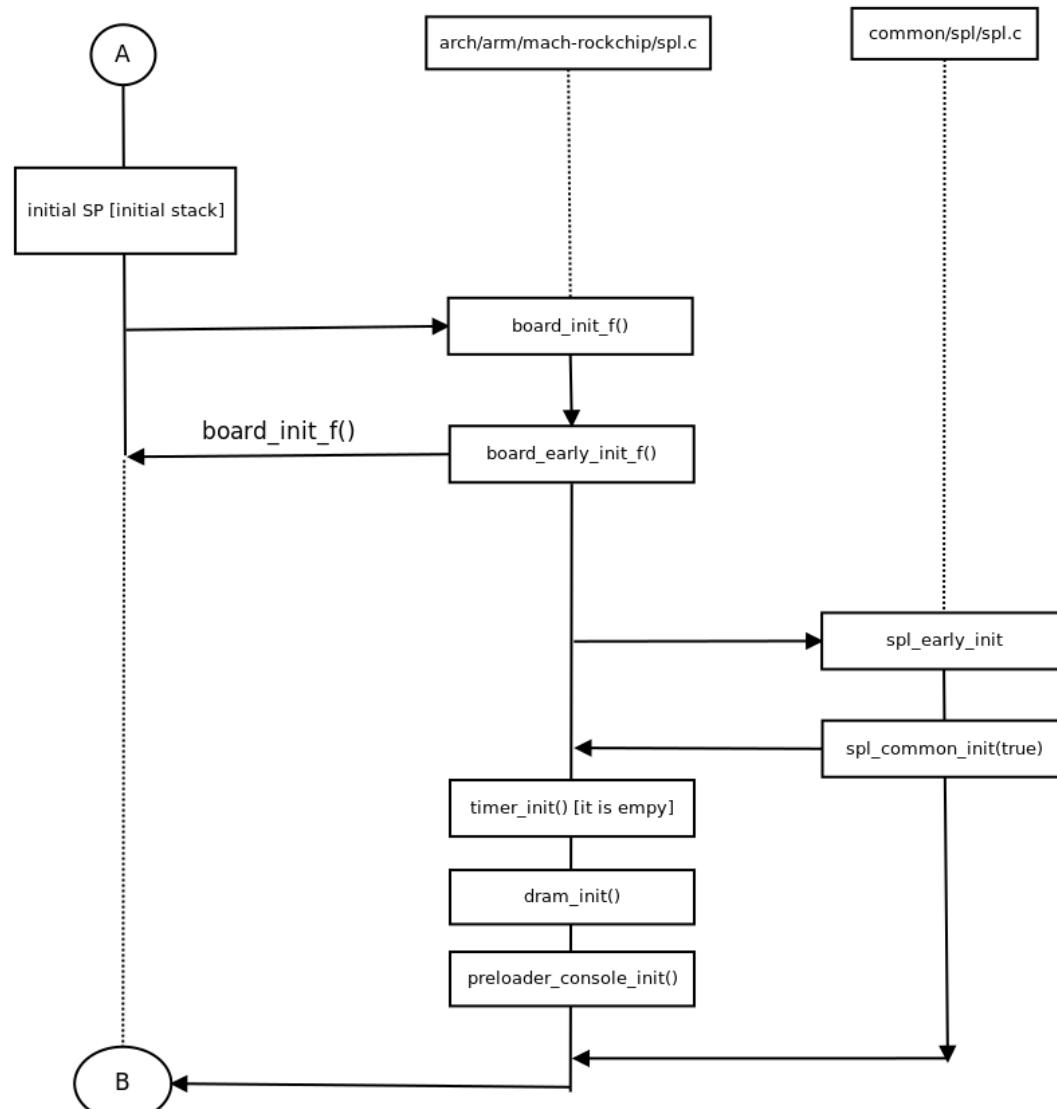


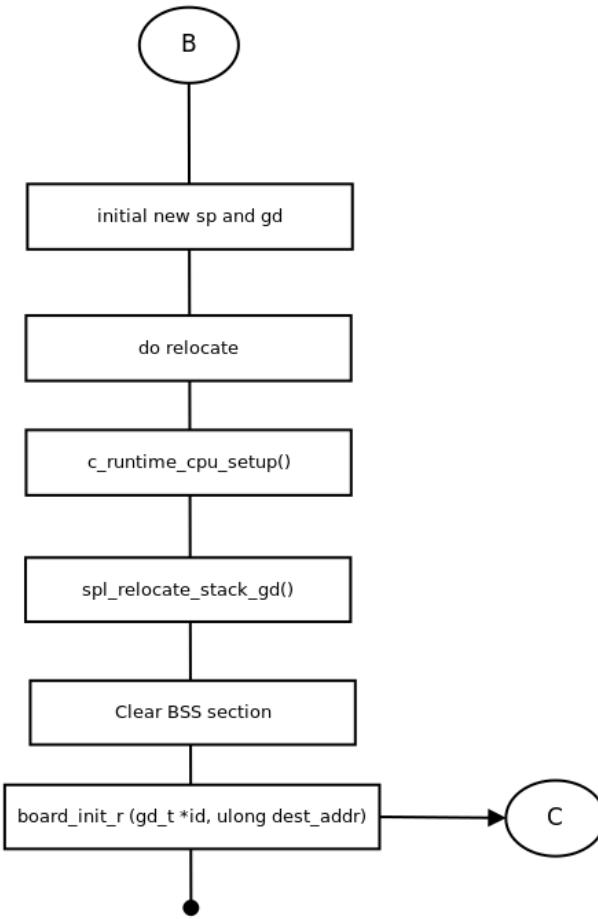
Exercise

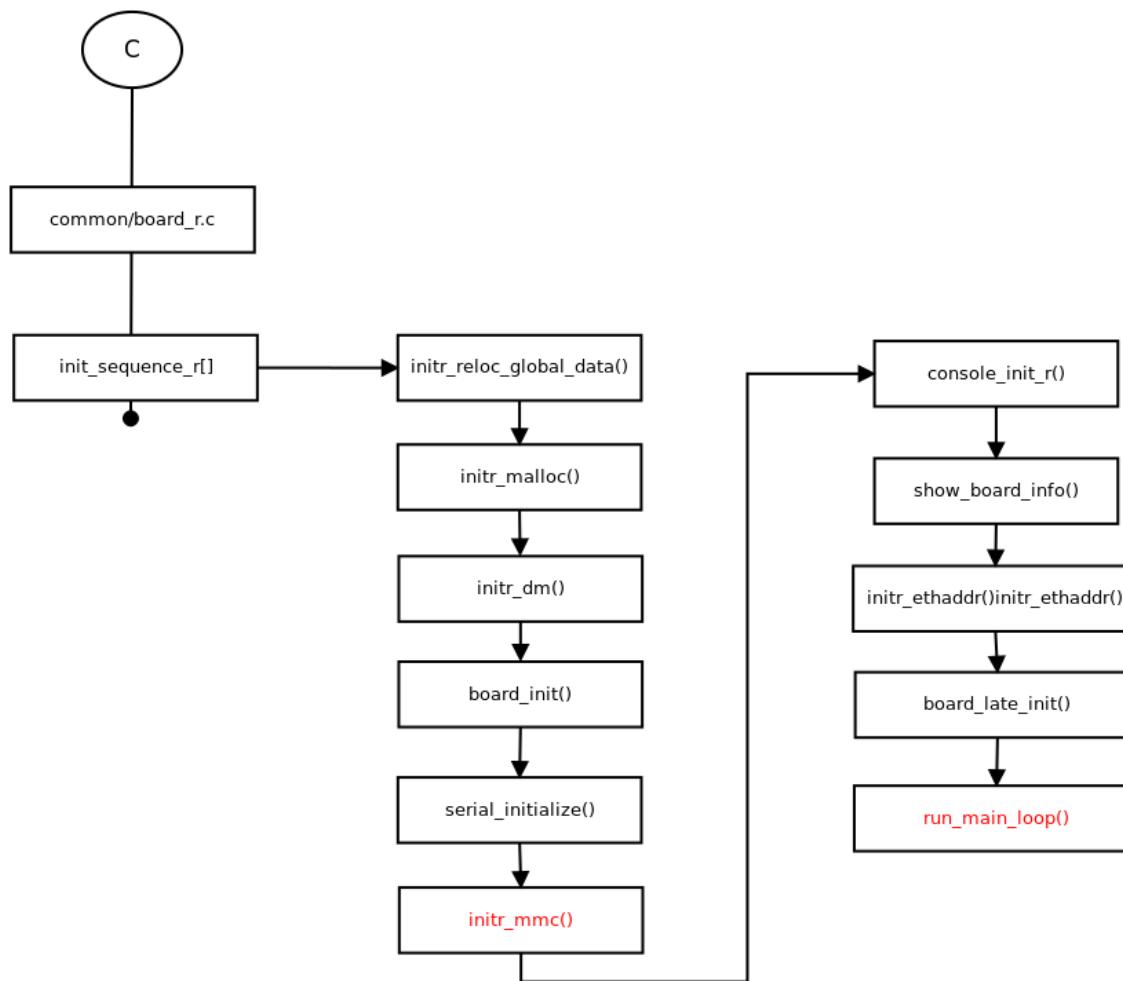
- ▶ Try to use everyone command in u-boot
- ▶ Use help function to know how to use command
- ▶ To know bootcmd and bootargs

U-boot Initialize Sequence



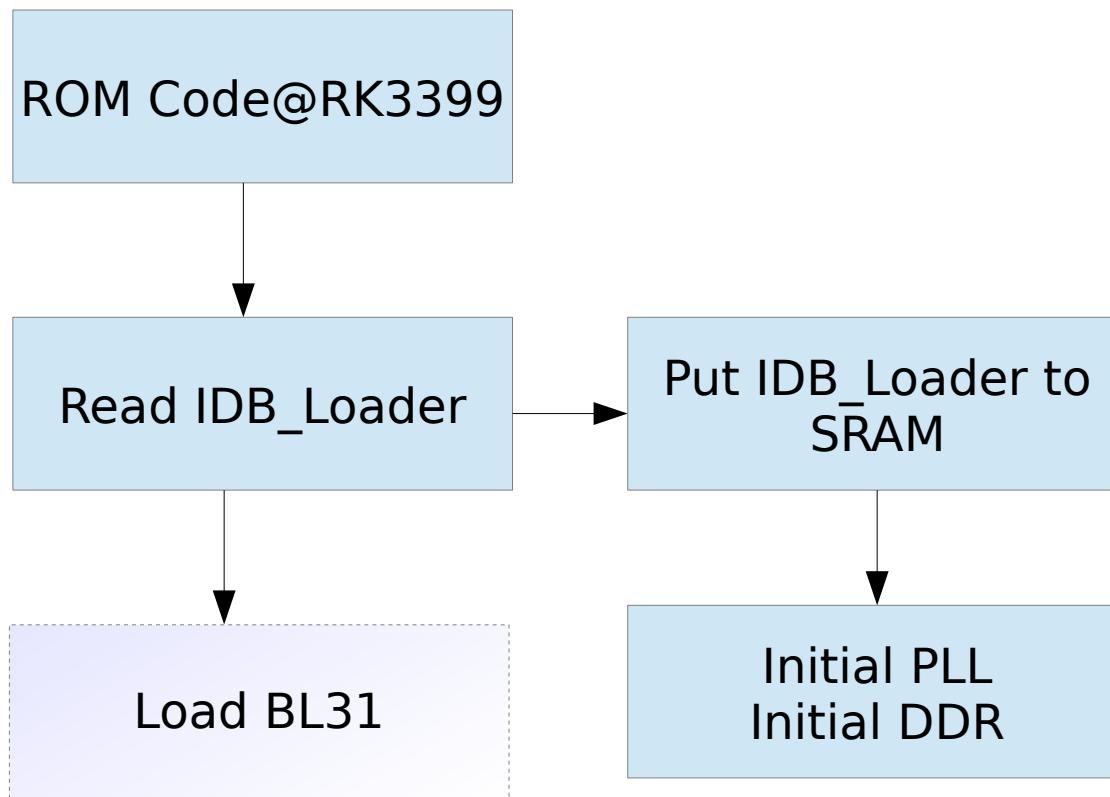




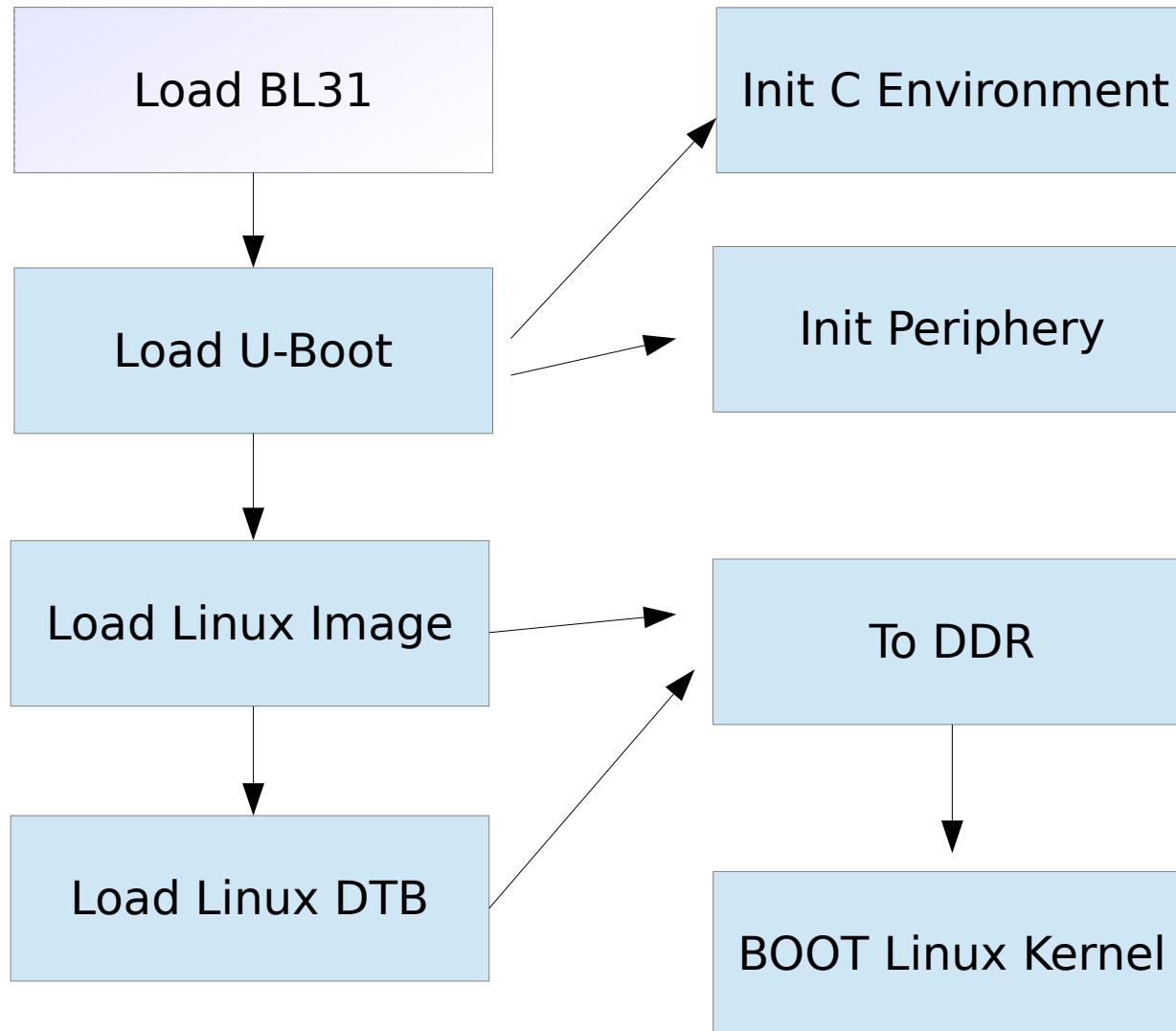


Boot Linux kernel

System Start Up



System Start Up





Boot Linux Kernel Command

▶ bootcmd

▶ \$ printenv bootcmd

```
bootcmd=run loadimage; run loadfdt;  
booti ${kernel_addr_r} - ${fdt_addr_r}
```

▶ \$ printenv loadimage → load Linux kernel Image

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

▶ \$ printenv loadfdt → Load Linux kernel device tree blob

```
loadfdt=ext4load mmc 1:1 ${fdt_addr_r} boot/rk3399-nanopi4-rev01.dtb
```



Boot OS (Linux) Command

▶ booti

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

▶ boota

→ boot **android style kernel image**

▶ bootrk

→ boot **rockchip 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)

```
bootargs=earlycon=uart8250,mmio32,0xff1a0000 swiotlb=1  
console=ttyFIQ0 rootwait root=/dev/mmcblk0p1 rw
```

- ▶ 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_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);



Linux Enter Point

arch/arm/lib/bootm.c

```
static void boot_jump_linux(bootm_headers_t *images, int fl
{
#ifndef CONFIG_ARM64
    void (*kernel_entry)(void *fdt_addr, void *res0, void *
    void *res2);
    int fake = (flag & BOOTM_STATE_OS_FAKE_GO);

    kernel_entry = (void (*)(void *fdt_addr, void *res0, vo
    void *res2))images->ep;
```

```
→ if (IMAGE_ENABLE_OF_LIBFDT && images->ft_len)  
→ → r2 = (unsigned long)images->ft_addr;  
→ else  
→ → r2 = gd->bd->bi_boot_params;
```

```
→ unsigned long machid = gd->bd->bi_arch_number;
→ char *s;
→ void (*kernel_entry)(int zero, int arch, uint params);
→ unsigned long r2;
→ int fake = (flag & B00TM_STATE_OS_FAKE_GO);

→ kernel_entry = (void (*)(int, int, uint))images->ep;
```

Assign DTB

Start To Kernel

Add Feature To U-boot



Add Feature

- » Add command → common/
- » 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)
```



Exercise

- ▶ Reference version command
 - create a helloworld command
- ▶ Reference LED command
 - create a led command for nanopi-m4



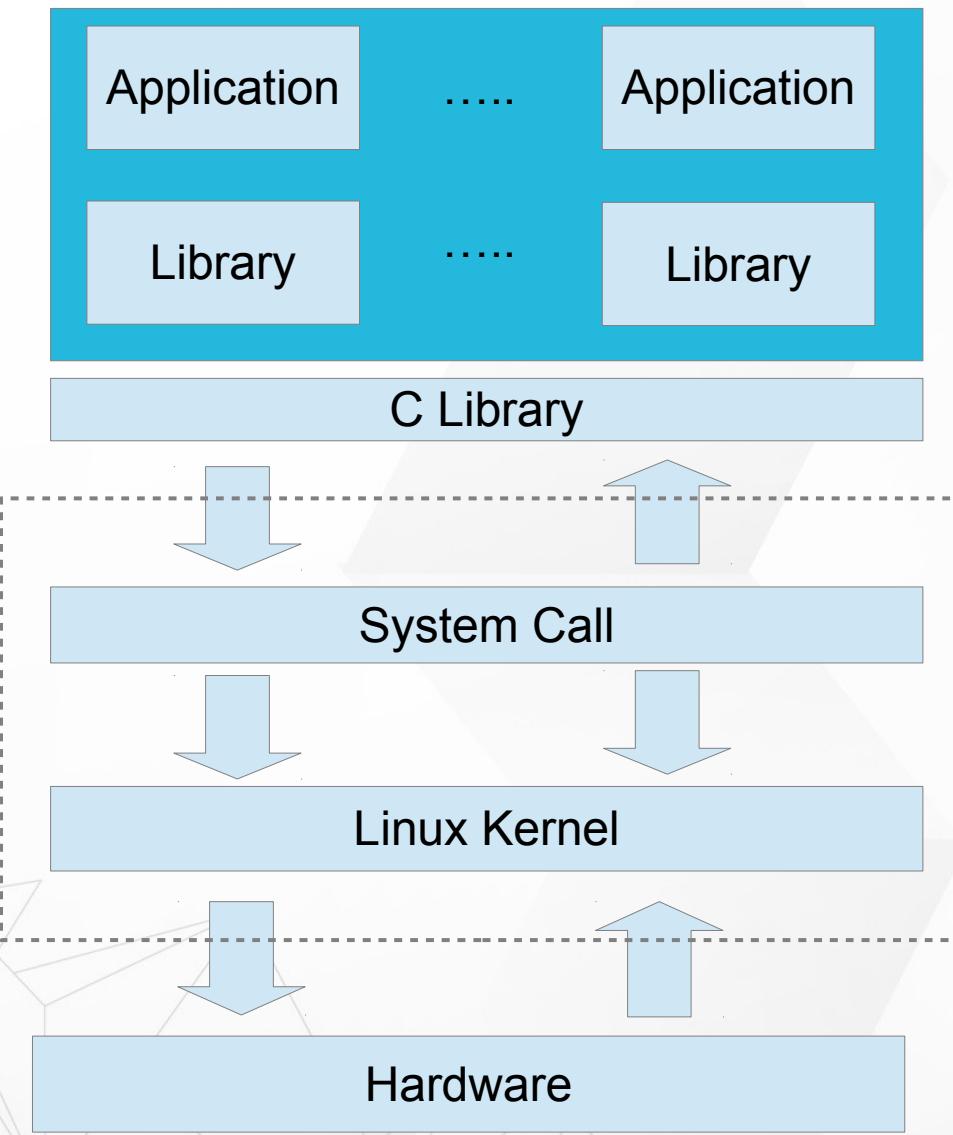
Linux Kernel



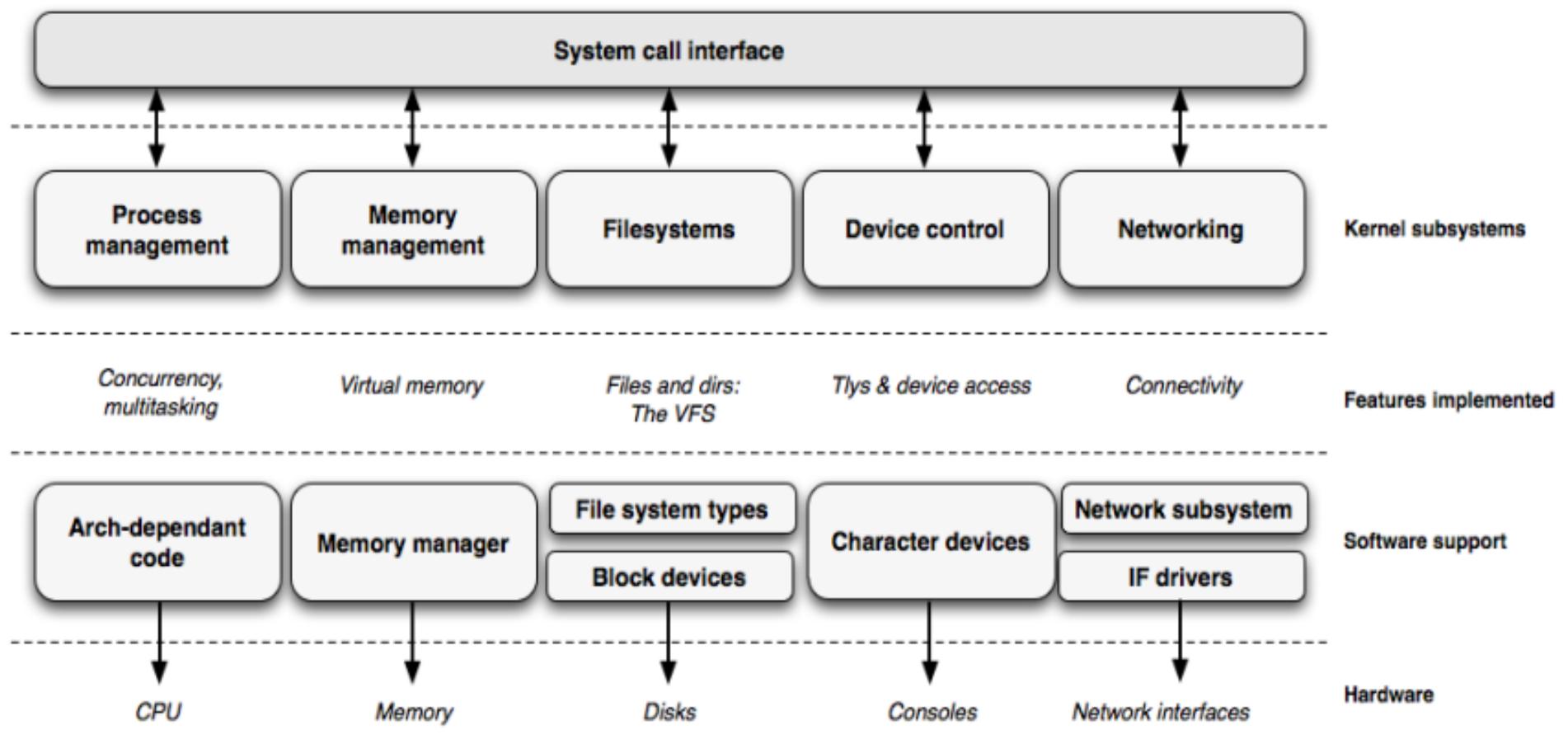
Linux kernel key features

- » 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 INSTALL_MOD_PATH



Basic Build Command

▶ \$ make modules_install

→ Install all modules to INSTALL_MOD_PATH

▶ \$ make mrproper

→ Remove all generated files (.config)

▶ \$ make clean

▶ \$ make distclean

→ Remove editor backup and patch reject files

Make help

\$ 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
ap4evo_defconfig     - Build for ap4evo
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
at91sam9g20_defconfig - Bui           (default: $(INSTALL_MOD_PATH)/lib/firmware)
at91sam9g45_defconfig - Bui dir/         - Build all files in dir and below
at91sam9rl_defconfig - Bui dir/file.[o|S] - Build specified target only
at91x40_defconfig    - Bui dir/file.lst  - Build specified mixed source/assembly target only
badge4_defconfig      - Bui           (requires a recent binutils and recent build (System.map))
bcmring_defconfig     - Bui dir/file.ko   - Build module including final link
bonito_defconfig      - Bui modules_prepare - Set up for building external modules
cam60_defconfig       - Bui tags/TAGS    - Generate tags file for editors
cerfcube_defconfig    - Bui cscope        - Generate cscope index
cm_x2xx_defconfig    - Bui gtags         - Generate GNU GLOBAL index
cm_x300_defconfig    - Bui kernelrelease - Output the release version string
cns3420vb_defconfig   - Bui kernelversion - Output the version stored in Makefile
colibri_pxa270_defconfig - Bui headers_install - Install sanitised kernel headers to INSTALL_HDR_PATH
colibri_pxa300_defconfig - Bui           (default: /home/xlloss/work/tiny-4412/build/linux_3.5.0_tiny4412/usr)
collie_defconfig      - Bui
corgi_defconfig       - Bui
cpu9260_defconfig    - Bui ..          - Build for cpu9260
cpu9020_defconfig    - Bui ..          - Build for cpu9020
```

How to Build Linux Kernel



Specifying cross-compilation

- ▶ **make ARCH=arm64 CROSS_COMPILE=arm-linux- ...**
 - export ARCH=arm64
 - export CROSS_COMPILE=aarch64-linux-gnu-

- ▶ Add above setting to script
 - source \$PATH/**set_toolchain.sh**



set_toolchain.sh

```
export PATH=$PATH:${TOOLCHAIN}/gcc-linaro-6.3.1-2017.05-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=${KERNEL_PATH}/rockchip-rk3399-nanopi-m4
#Set Linux kernel source path
```



Predefined configuration files

» Default configuration

» `arch/<arch>/configs/`

» `make nanopi4_linux_defconfig`

» To create your own default configuration file

→ `make savedefconfig`, to create a minimal configuration file

→ `mv defconfig arch/arm64/configs/myown_defconfig`

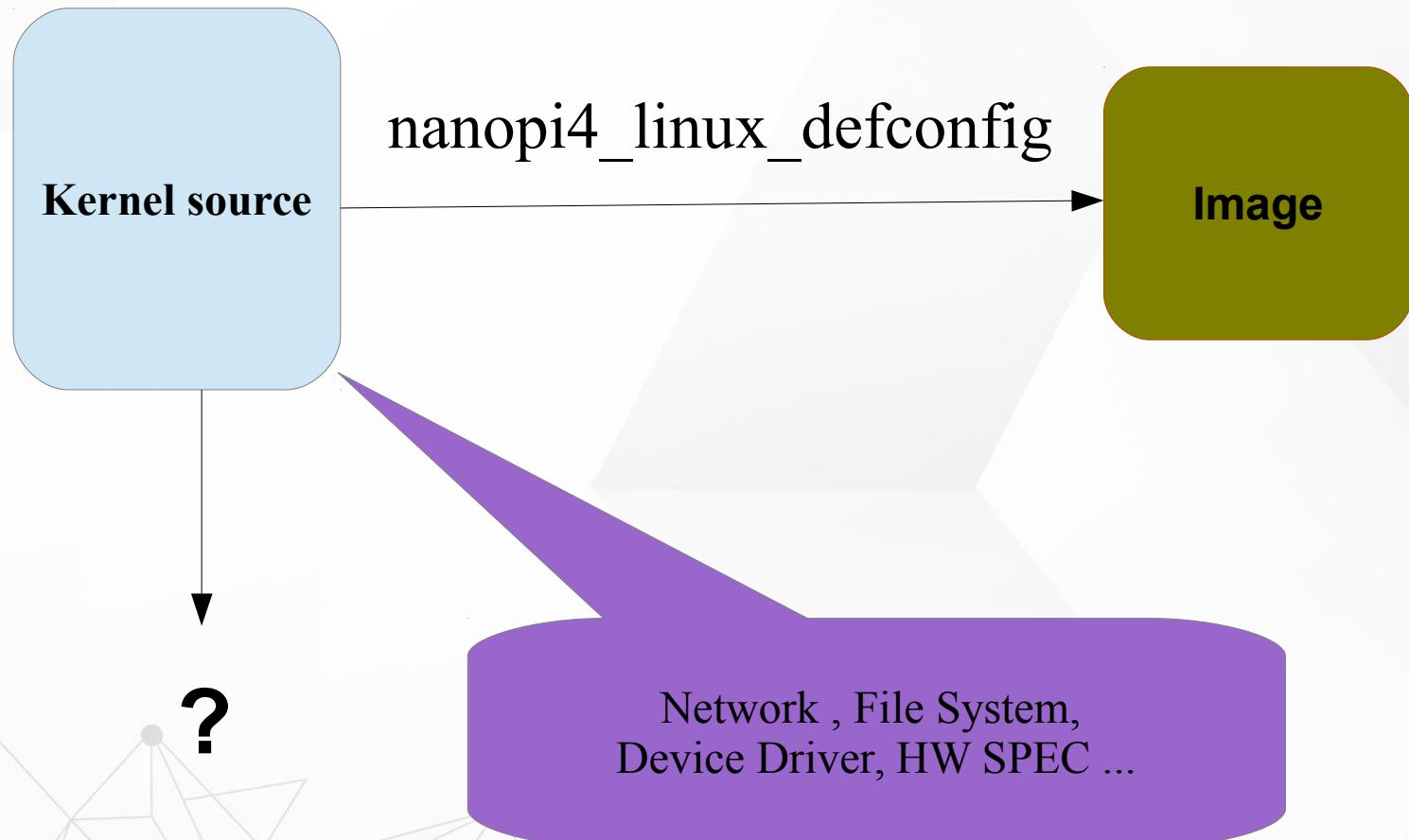


Kernel compilation

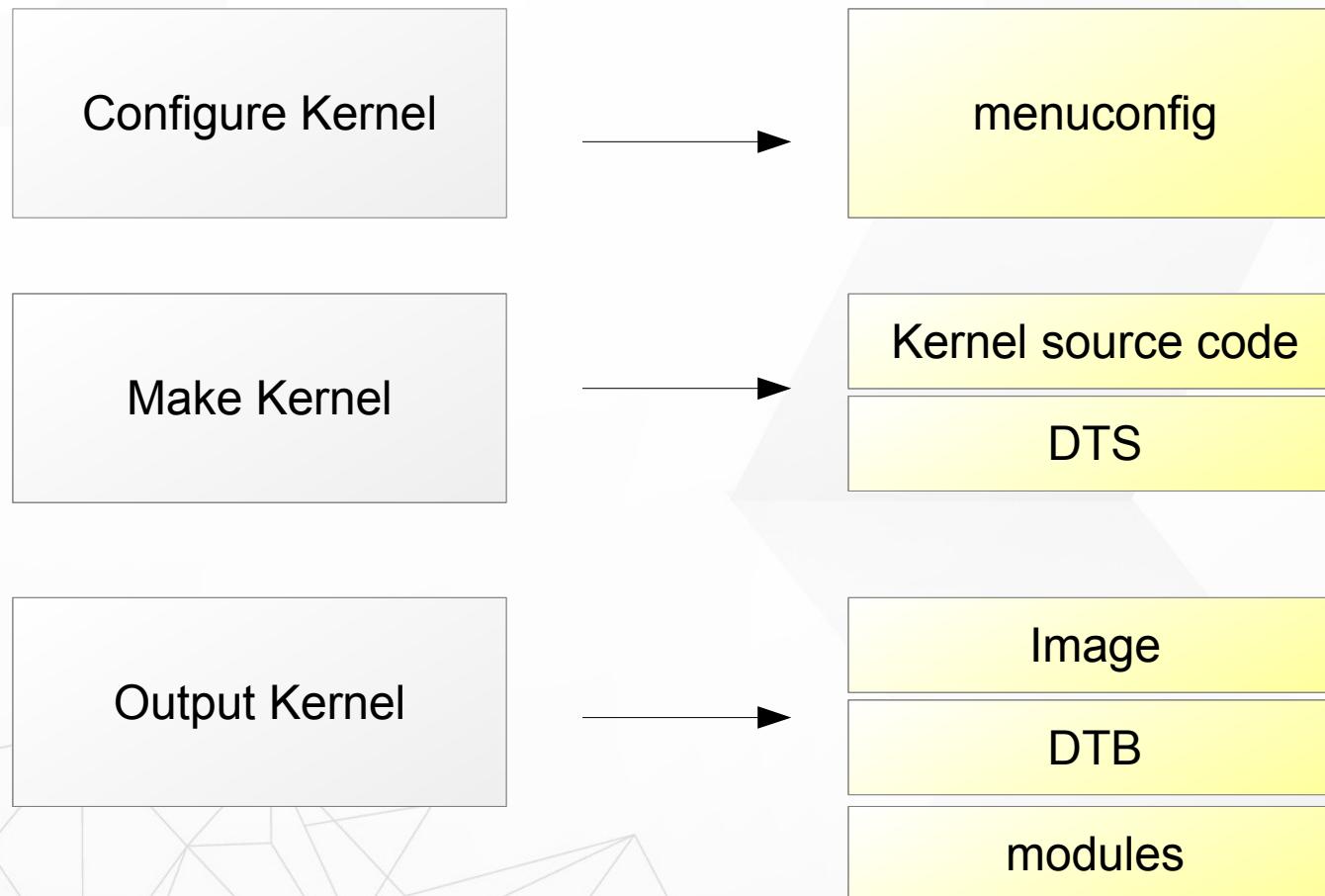
- ▶ Build kernel Image → **make -j4**
 - ▶ To run multiple jobs in parallel if you have multiple CPU cores
- ▶ Generates Image
 - ▶ arch/arm64/boot/Image
 - **Image** for ARM64,
 - ▶ arch/arm64/boot/dts/rockchip/
 - DTB : rk3399-nanopi4-rev01.dtb
 - DTB : rk3399-nanopi4-rev21.dtb



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
 - » `make menuconfig` → Text
 - » `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/
nanopi4_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

```
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

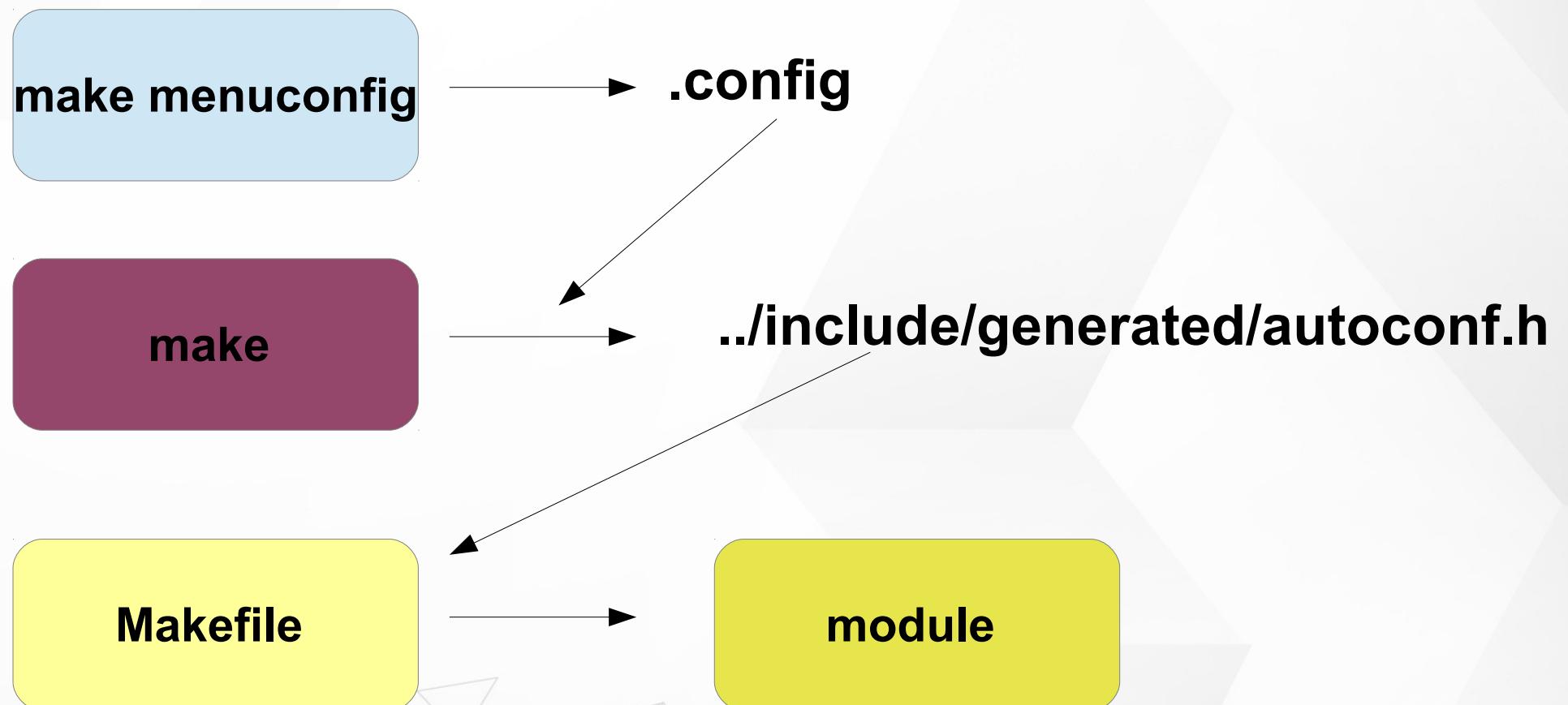
[ ] General setup --->
[*] Enable loadable module support --->
[*] Enable the block layer --->
    Platform selection --->
    Bus support --->
    Kernel Features --->
    Boot options --->
    Userspace binary formats --->
    Power management options --->
    CPU Power Management --->
[*] Networking 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
```



Linux Kernel Booting

- Bootloader run kernel with parameters
 - x0 = DTB (Device Tree Blob)
 - r1 = NULL
 - r2 = NULL

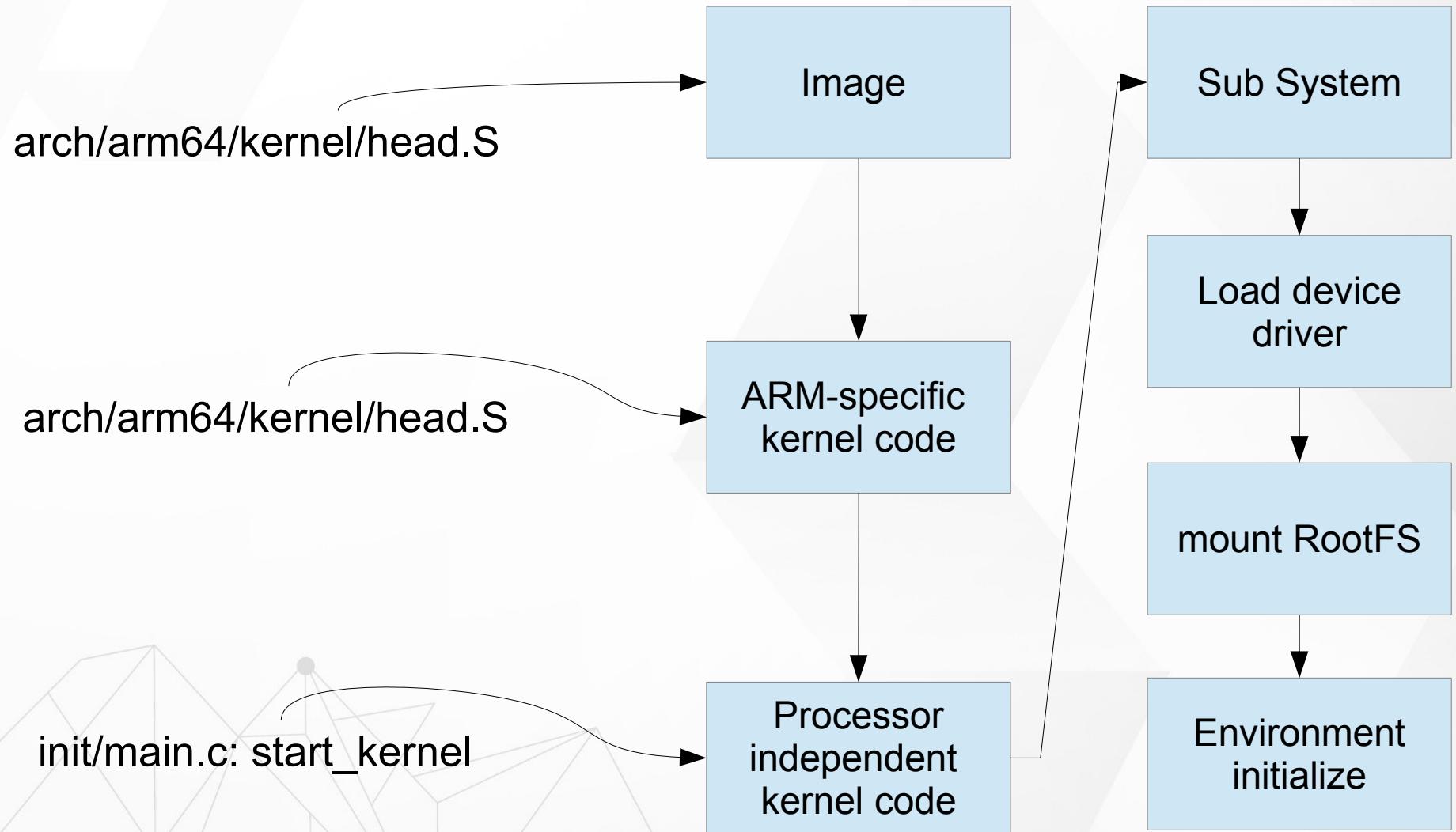


Kernel Startup Entry Point

arch/arm64/kernel/head.S

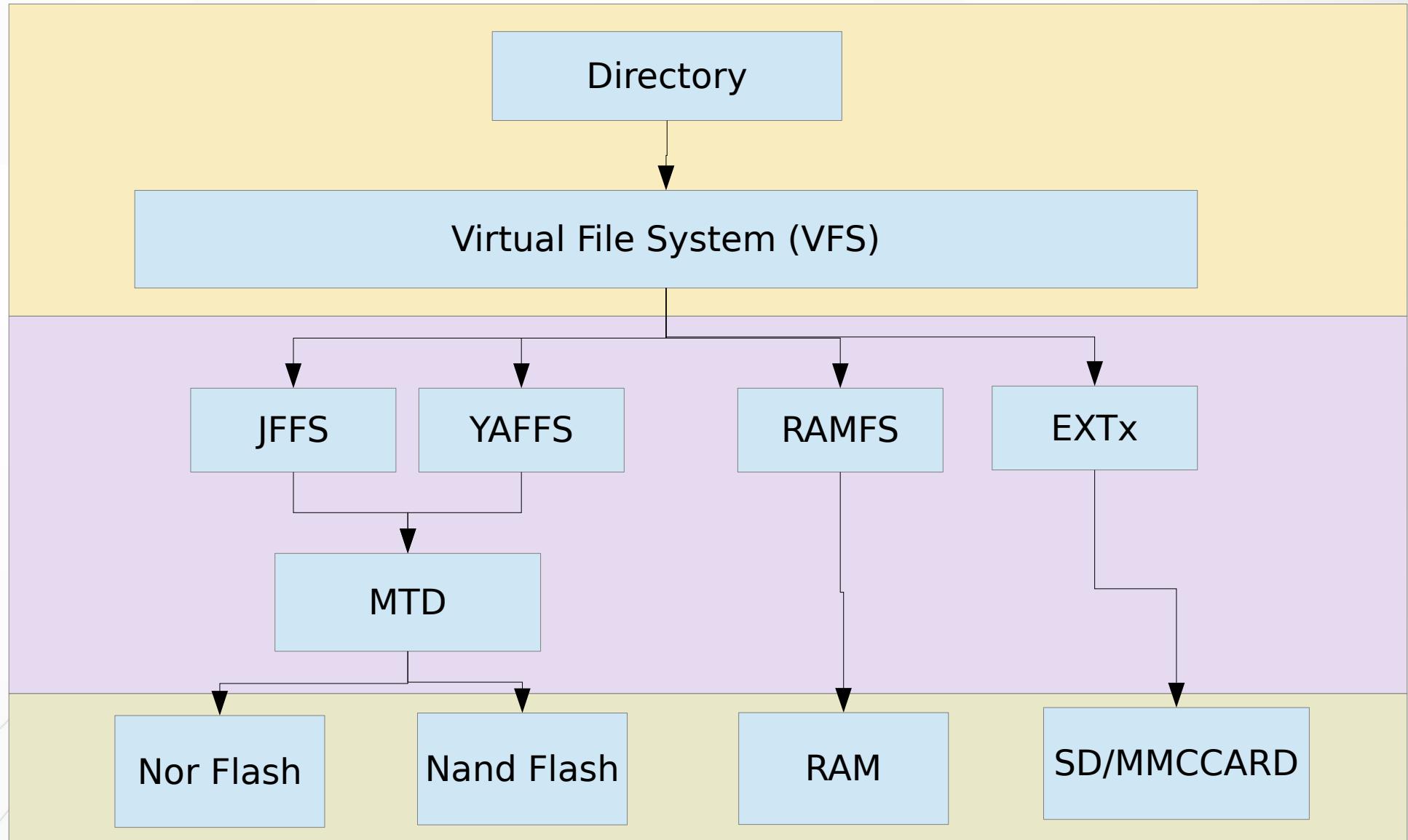
```
ENTRY(stext)
    bl→preserve_boot_args
    bl→el2_setup→→→→// Drop to EL1, w20=cpu_boot_mode
    adrp→x24, __PHYS_OFFSET
    and→x23, x24, MIN_KIMG_ALIGN - 1→→// KASLR offset, defaults to 0
    bl→set_cpu_boot_mode_flag
    bl→__create_page_tables→→→→// x25=TTBR0, x26=TTBR1
    /*
     * The following calls CPU setup code, see arch/arm64/mm/proc.S for
     * details.
     * On return, the CPU will be ready for the MMU to be turned on and
     * the TCR will have been set.
     */
    bl→__cpu_setup→→→→// initialise processor
    adr_l→x27, __primary_switch→→→→// address to jump to after
    →→→→→→→→→→// MMU has been enabled
    b→__enable_mmu
ENDPROC(stext)
```

Linux Kernel Booting

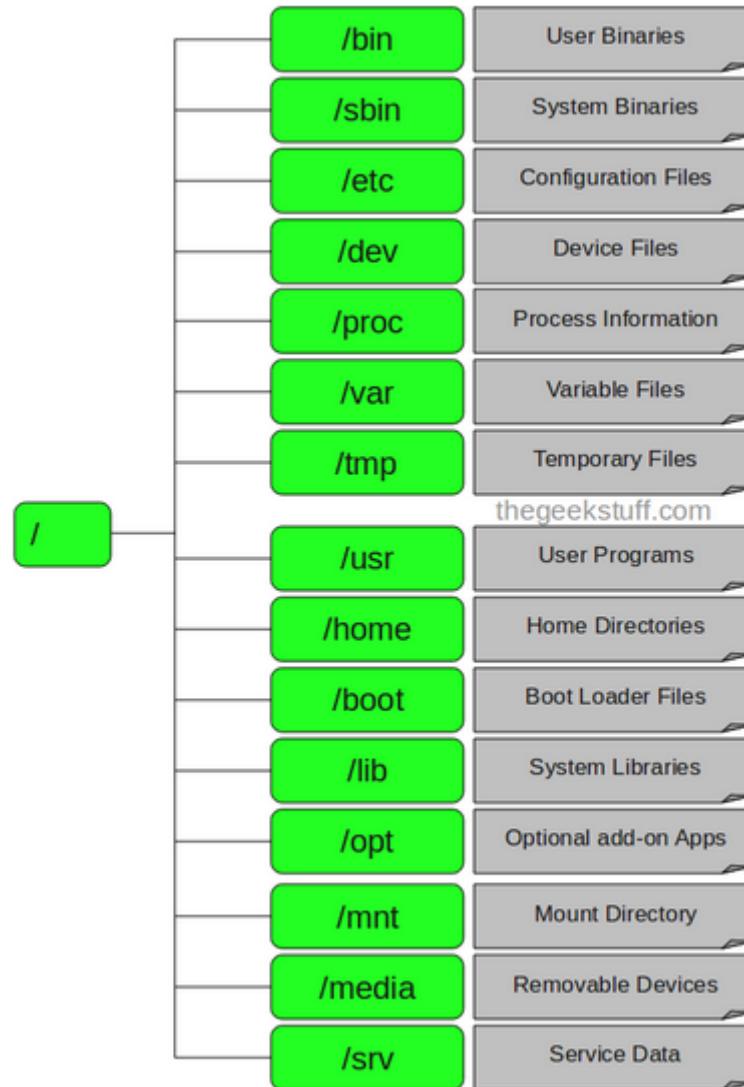


CH 7 Linux Root File System

File System in Linux



Linux RootFS 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



/var – Variable Files

- var stands for variable files.
- Content of the files that are expected to grow can be found under this directory.
- This includes — system log files (/var/log); packages and database files (/var/lib); emails (/var/mail); print queues (/var/spool); lock files (/var/lock); temp files needed across reboots (/var/tmp);



/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.*`



/opt – Optional add-on Applications

- ▶ opt stands for optional.
- ▶ Contains add-on applications from individual vendors.
- ▶ add-on applications should be installed under either /opt/ or /opt/ sub-directory.



/mnt – Mount Directory

- ▶ Temporary mount directory where sysadmins can mount filesystems.



/media – Removable Media Devices

- ▶ Temporary mount directory for removable devices.
- ▶ For examples, /media/cdrom for CD-ROM;
/media/floppy for floppy drives; /media/cdrecorder for
CD writer



Linux System V

- » AT&T developed
- » The first initial process is → init
- » /etc/inittab
- » /etc/init.d/ and /etc/init.d/rcS



Linux System V

► Inittab

→ terminal initial setting table

```
# Startup the system
::sysinit:/bin/mount -t proc proc /proc
::sysinit:/bin/mount -o remount,rw /
::sysinit:/bin/mkdir -p /dev/pts
::sysinit:/bin/mkdir -p /dev/shm
::sysinit:/bin/mount -a 2>/dev/null
::sysinit:/bin/hostname -F /etc/hostname
# now run any rc scripts
::respawn:-/bin/sh
::sysinit:/etc/init.d/rcS

# Put a getty on the serial port
#ttyFIQ0::respawn:/sbin/getty -L  ttyFIQ0 0 vt100 # GENERIC_SERIAL

# Stuff to do for the 3-finger salute
#:ctrlaltdel:/sbin/reboot

# Stuff to do before rebooting
::shutdown:/etc/init.d/rcK
::shutdown:/sbin/swapoff -a
::shutdown:/bin/umount -a -r
```



Linux System V

► respawn

→ The process will be restarted whenever it terminates

► sysinit

→ sysinit actions are started first, and init waits for them to complete

► askfirst

→ For askfirst, before running the specified process, init displays the line "Please press Enter to activate this console"

► shutdown

→ shutdown actions are run on halt/reboot/poweroff, or on SIGQUIT

<https://git.busybox.net/busybox/tree/examples/inittab>



Linux System V

▶ /etc/init.d/S*

→ initial system script

```
[root@rk3399:/etc/init.d]# ls
S01logging      S22hdmiон      S50link_iq          S80dnsmasq
S10init         S30dbus        S50sshd            S99input-event-daemon
S10udev         S40network     S50telnet          rcK
S20urandom      S41dhpcd       S50usbdevice       rcS
S21mountall.sh  S50launcher_  S66load_wifi_modules
```



Linux System V

- » /etc/profile
 - » /etc/profile.d/
- environment initial script

```
[root@rk3399:/etc/init.d]# ls
S01logging      S22hdmiон      S50link_iq          S80dnsmasq
S10init         S30dbus        S50sshd            S99input-event-daemon
S10udev         S40network    S50telnet           rcK
S20urandom     S41dhcpcd    S50usbdevice        rcS
S21mountall.sh  S50launcher_ S66load_wifi_modules
```



Rockchip Launch Script

▶ /etc/init.d/S50launcher

→ launch rockchip or customer applications

```
start)
    printf "Starting launcher: "
    export LC_ALL='zh_CN.utf8'
    export QT_QPA_PLATFORM=wayland

    # music
    /usr/bin/audioservice &

    # bt
    /usr/libexec/bluetooth/bluetoothd --compat &

    #for kmssink
    #export QT_GSTREAMER_WINDOW_VIDEOSINK=kmssink
```

BusyBox



BusyBox

- Linux system needs a basic set of programs to work
 - Init program
 - shell
- In normal Linux systems, command are independent programs
 - Binary size large
- BusyBox down size these program

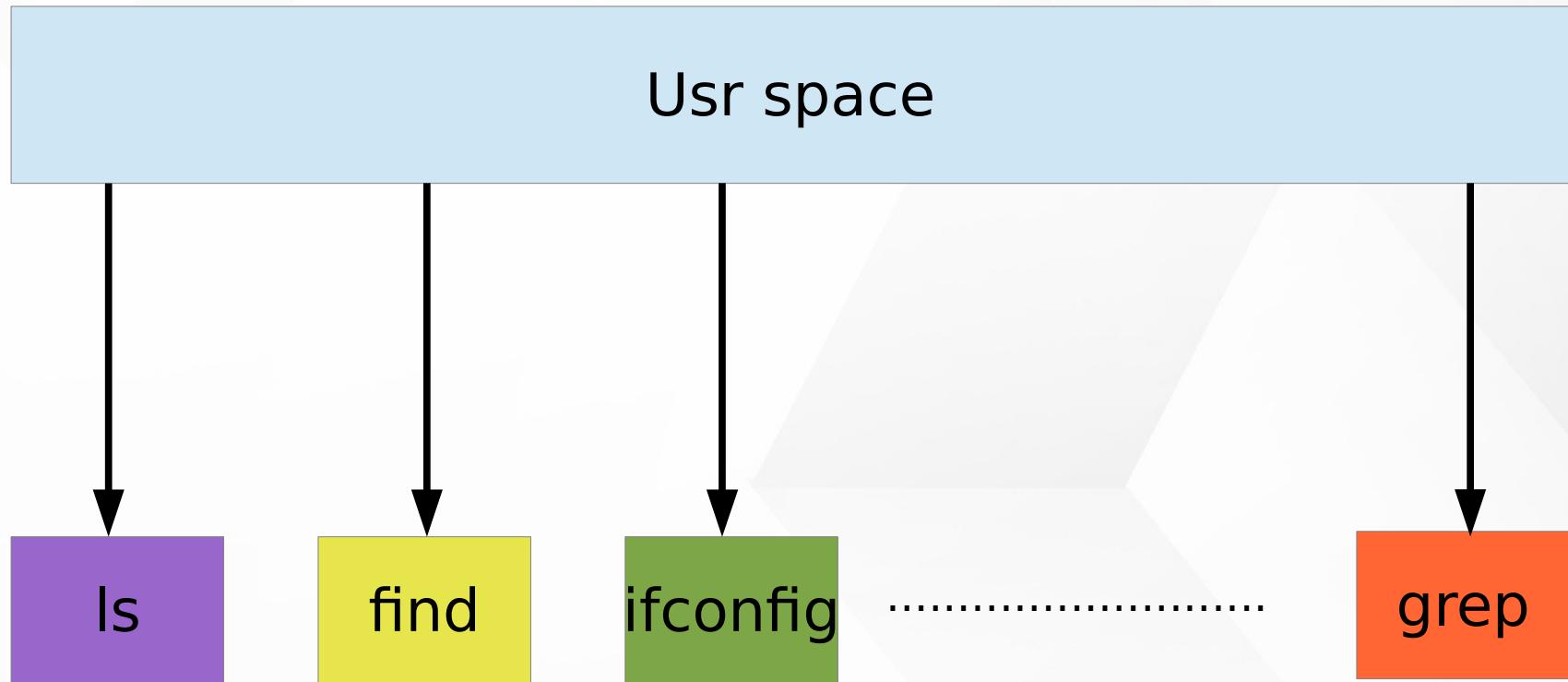


BusyBox

- Rewrite of many useful Unix command
 - Down size
- All the program are compiled into a single executable, /bin/busybox
- It can configure all command
- <http://www.busybox.net/>

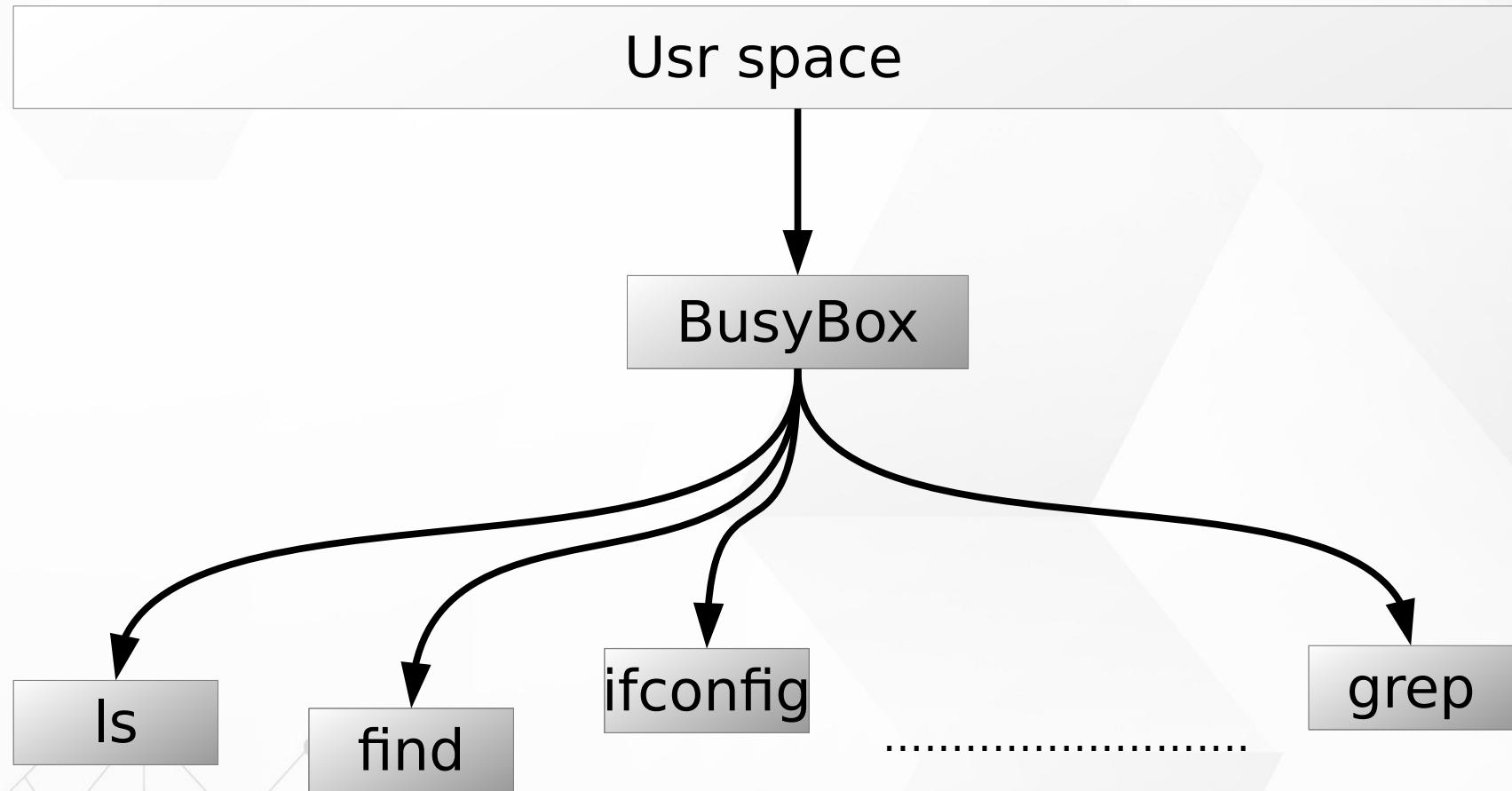


RootFS no BusyBox





RootFS on BusyBox





Busybox init

- Busybox provides an implementation of an init program
- A single configuration file: /etc/inittab
 - Each line has the form <id>::<action>:<process>
- Check examples/inittab in Busybox for details on the configuration



Configuring BusyBox

➤ <http://busybox.net>

→ wget https://busybox.net/downloads/busybox-1.30.0.tar.bz2

➤ Configure BusyBox

→ make menuconfig

→ make defconfig

➤ Install it

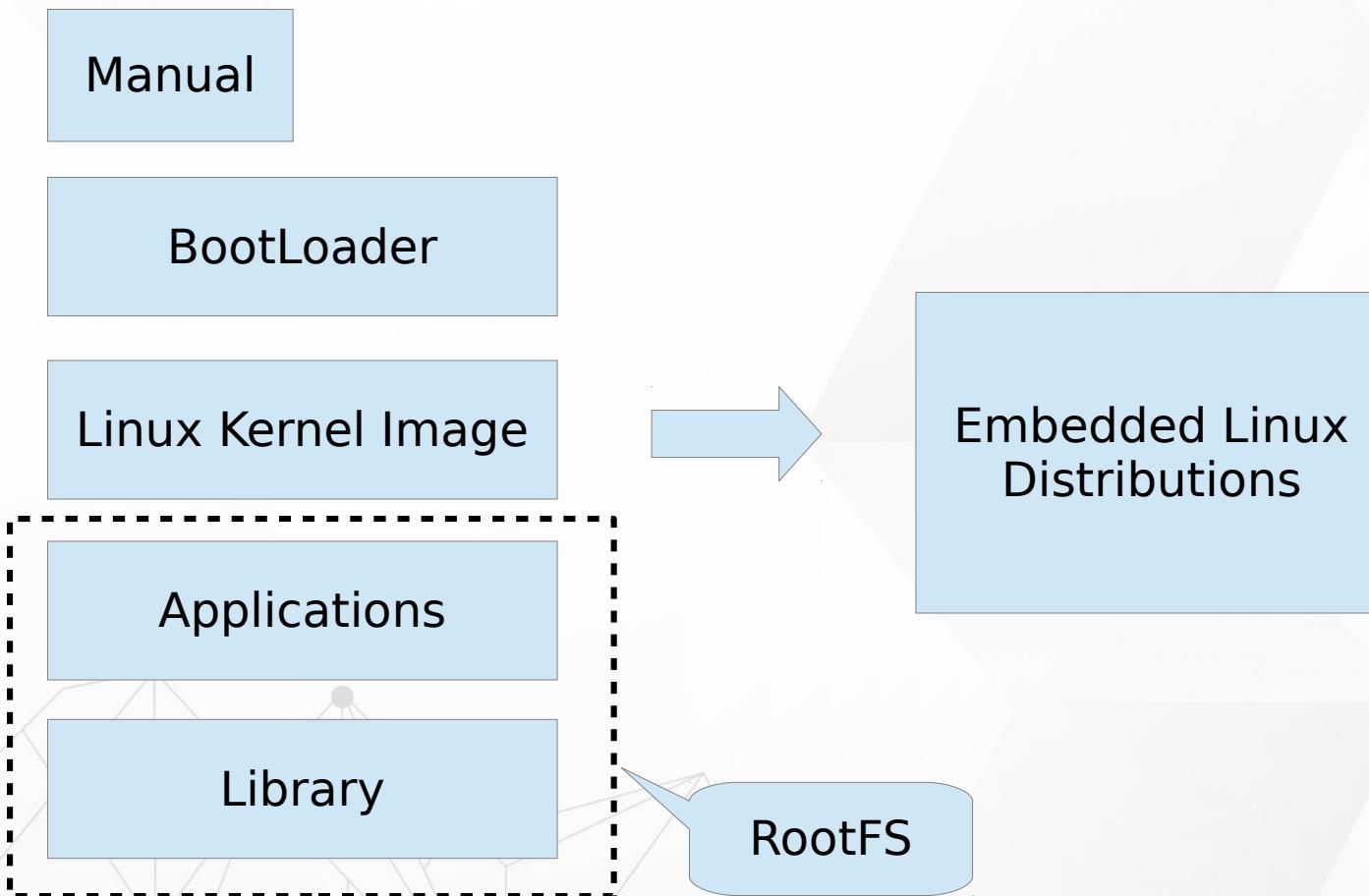
→ make install



Linux Distribution

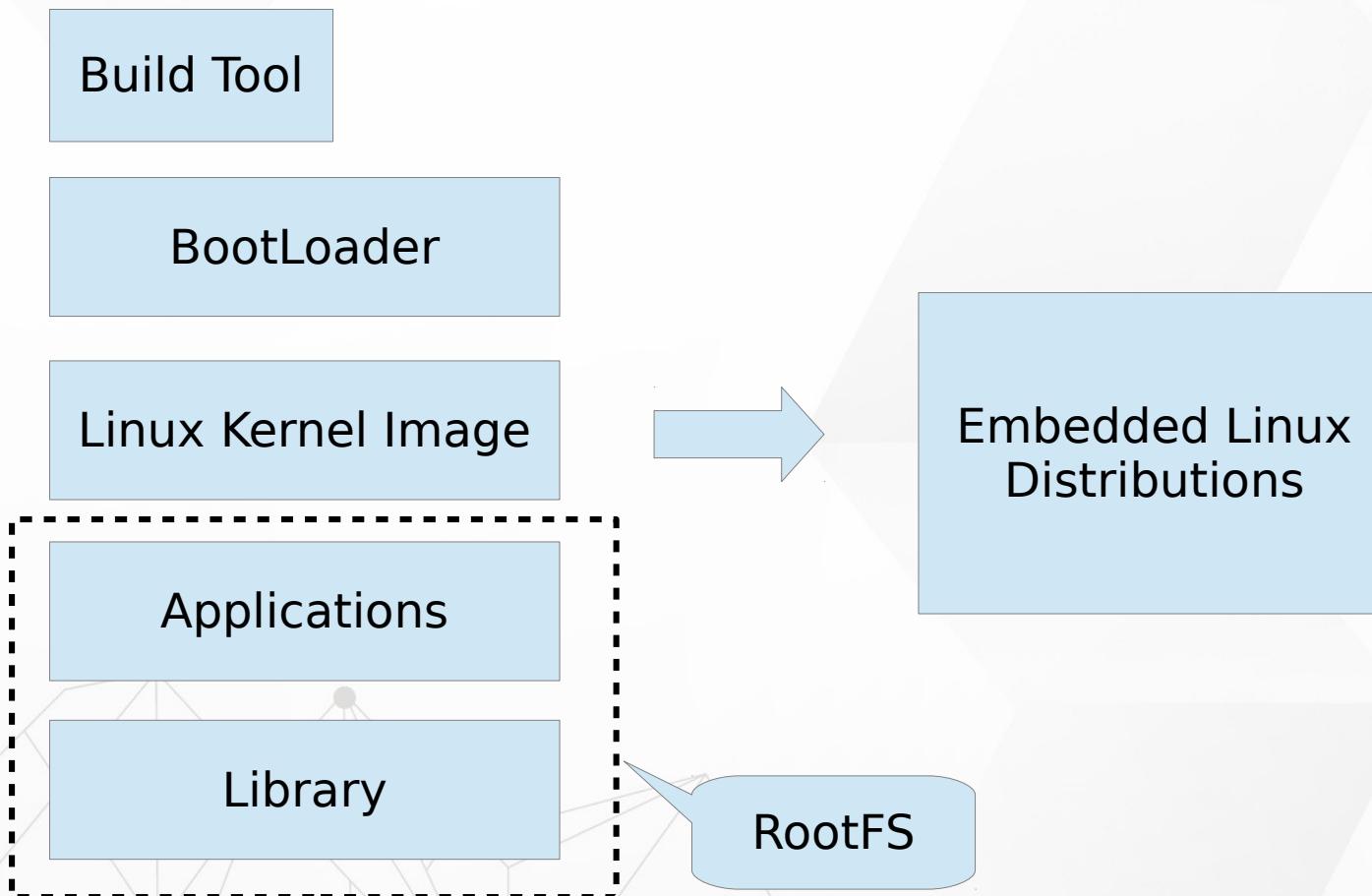
- » Boot-loader
- » Linux kernel
- » RootFS
- » Application
- » Library
- » Linux driver moudes

Build Distribution by Manual



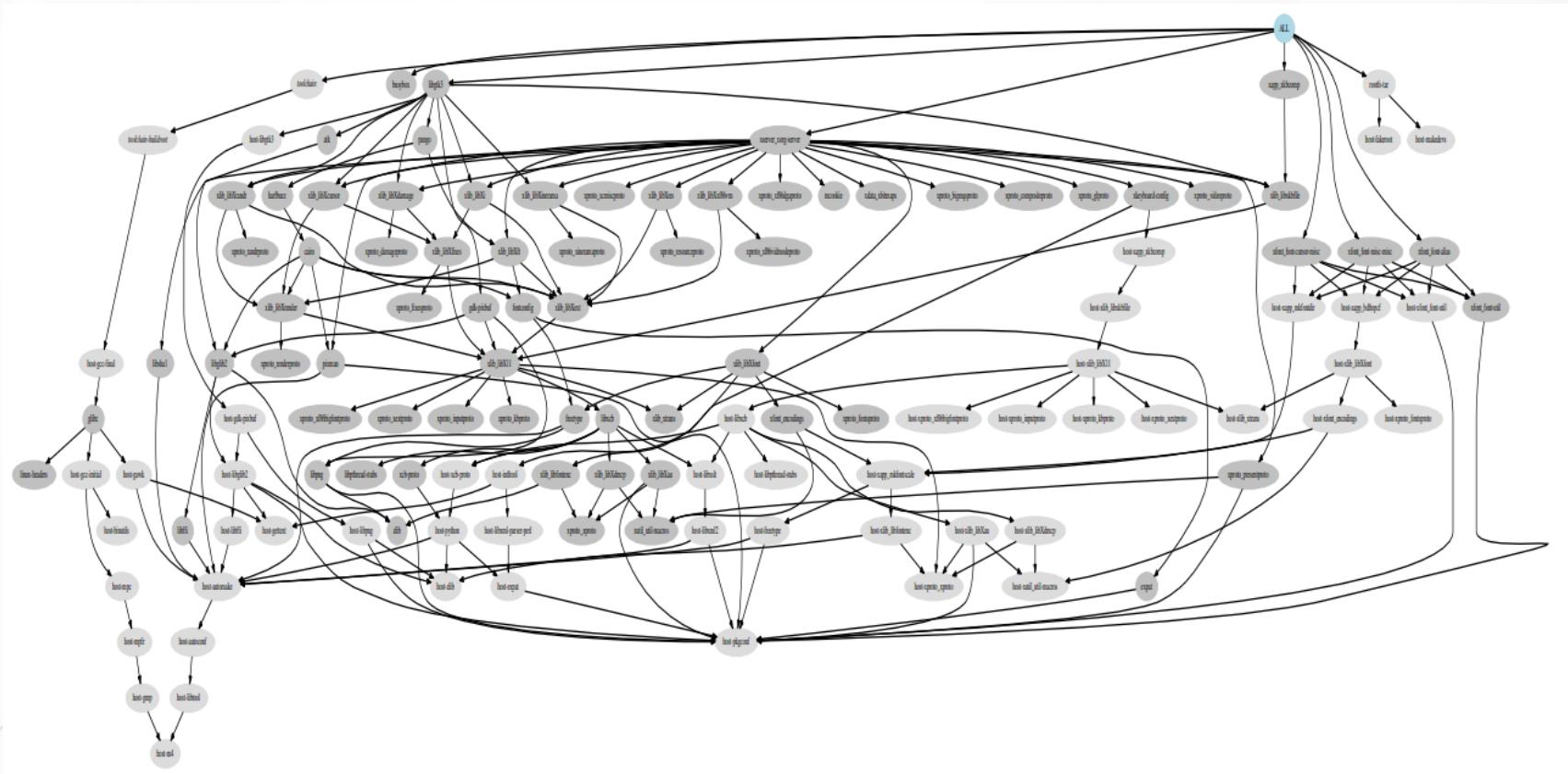


Build Distribution by Tool



Complexity of user space depend

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





System integration: several possibilities

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

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

Buildroot

Buildroot

► <https://buildroot.org/>

The screenshot shows the official Buildroot website. At the top, there's a navigation bar with links for News, Documentation, Support, Contribute, Sponsors, Association, and a prominent green DOWNLOAD button. Below the navigation is a large blue header section featuring a yellow hard hat icon on the left and the text "Buildroot" in large white letters, followed by "Making Embedded Linux Easy". Underneath this, there are two buttons: "LEARN MORE" and "DOWNLOAD". The main content area below the header contains a paragraph about Buildroot's purpose: "Buildroot is a simple, efficient and easy-to-use tool to generate embedded Linux systems through cross-compilation." Below this text are three circular icons: a penguin icon labeled "Can handle everything", a hammer and nail icon labeled "Is very easy", and a gift box icon labeled "Supports several thousand packages".

Buildroot

Making Embedded Linux Easy

LEARN MORE

DOWNLOAD

Buildroot is a simple, efficient and easy-to-use tool to generate embedded Linux systems through cross-compilation.

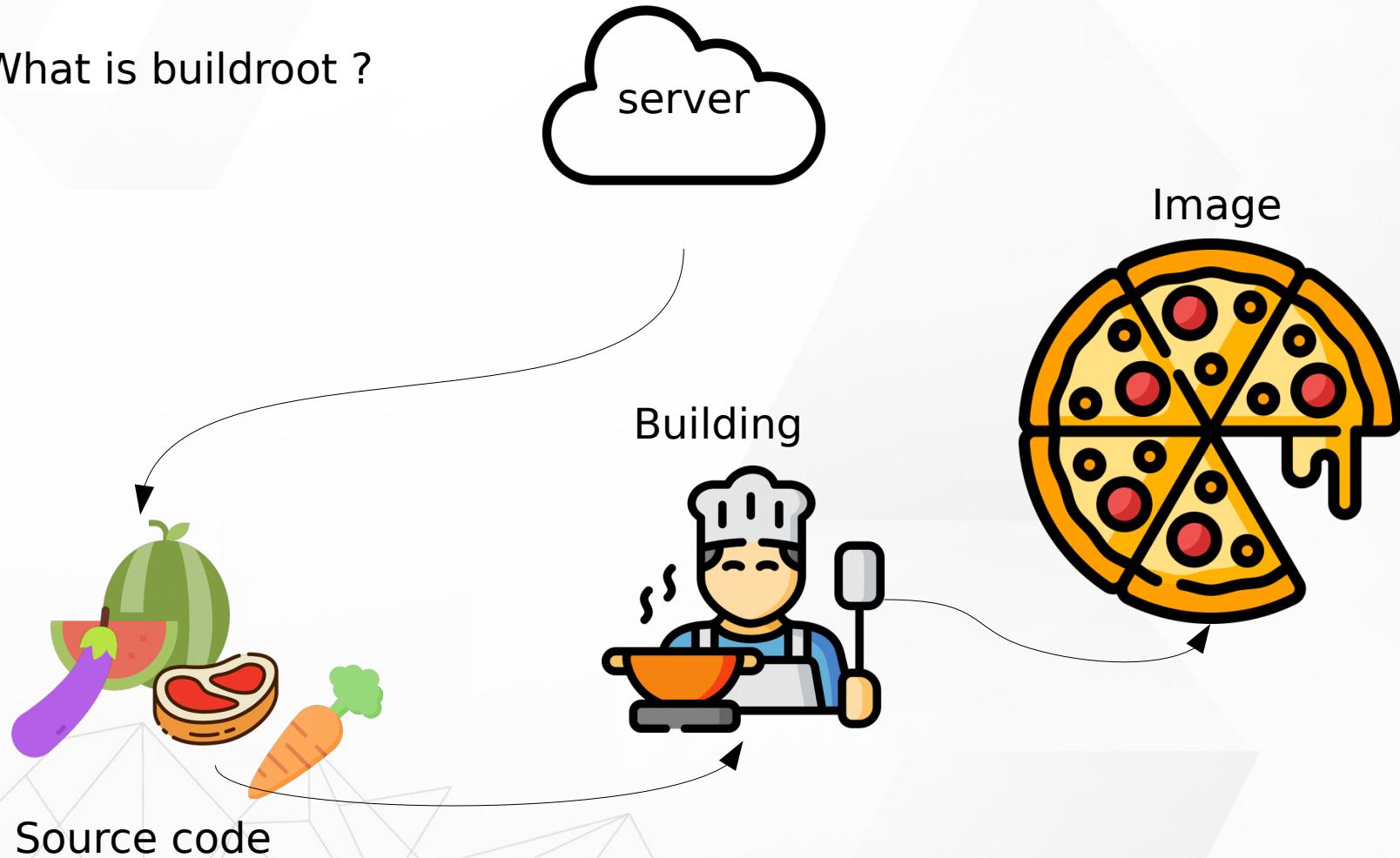
Can handle everything

Is very easy

Supports several thousand packages

What is Buildroot

What is buildroot ?





Get Buildroot

- » wget https://buildroot.org/downloads/buildroot-2020.02.1.tar.gz
- » tar -xvzf buildroot-2-2-.02.1.tar.bz2
- » cd buildroot-2-2-.02.1.tar.bz2
- » make menuconfig



Configure

Arrow keys navigate the menu. <Enter> selects submenus --> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> selects a feature, while <N> excludes a feature. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] feature is selected [] feature is excluded

```
[*] abootimg (NEW)
    *** aufs-util needs a linux kernel and a toolchain w/ threads ***
[ ] autofs (NEW)
[ ] btrfs-progs (NEW)
[ ] cifs-utils (NEW)
    *** cpio needs a toolchain w/ wchar ***
[ ] cramfs (NEW)
    *** curlftpfs needs a toolchain w/ wchar, threads, dynamic library ***
[ ] davfs2 (NEW)
    *** dosfstools needs a toolchain w/ wchar ***
[ ] e2fsprogs (NEW) -----
    *** e2tools needs a toolchain w/ threads, wchar ***
    *** encryptfs-utils needs a toolchain w/ threads, wchar, dynamic library ***
    *** exfat needs a toolchain w/ wchar, threads, dynamic library ***
    *** exfat-utils needs a toolchain w/ wchar ***
    *** f2fs-tools needs a toolchain w/ wchar ***
[ ] flashblock (NEW)
[ ] fscryptctl (NEW)
    *** fwup needs a toolchain w/ wchar ***
[ ] genext2fs (NEW)
[ ] genpart (NEW)
[ ] genromfs (NEW)
[ ] imx-usb-loader (NEW)
[ ] mmc-utils (NEW)
[ ] mtd, jffs2 and ubi/ubifs tools (NEW)
    *** mtools needs a toolchain w/ wchar ***
[ ] ntfs-utils (NEW)
[ ] nilfs-utils (NEW)
    *** ntfs-3g needs a toolchain w/ wchar, threads, dynamic library ***
[ ] sp-oops-extract (NEW)
[ ] squashfs (NEW)
    *** sshfs needs a toolchain w/ wchar, threads, dynamic library ***
    *** udftools needs a toolchain w/ wchar ***
[ ] unionfs (FUSE) (NEW)
[ ] xfsprogs (NEW)
```

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



Toolchain

```
[ ] Toolchain type (Buildroot toolchain) --->
    *** Toolchain Buildroot Options ***
    (buildroot) custom toolchain vendor name (NEW)
        C library (uClibc-ng) --->
            *** Kernel Header Options ***
            Kernel Headers (Linux 4.15.x kernel headers) --->
                *** uClibc Options ***
    (package/uclibc/uClibc-ng.config) uClibc configuration file to use? (NEW)
        () Additional uClibc configuration fragment files (NEW)
        [ ] Enable WCHAR support (NEW)
        [ ] Enable toolchain locale/i18n support (NEW)
            Thread library implementation (Native POSIX Threading (NPTL)) --->
        [ ] Thread library debugging (NEW)
        [ ] Enable stack protection support (NEW)
        [*] Compile and install uClibc utilities (NEW)
            *** Binutils Options ***
            Binutils Version (binutils 2.29.1) --->
        () Additional binutils options (NEW)
            *** GCC Options ***
            GCC compiler Version (gcc 6.x) --->
        () Additional gcc options (NEW)
        [ ] Enable C++ support (NEW)
            *** Fortran support needs a toolchain w/ wchar ***
        [ ] Enable compiler link-time-optimization support (NEW)
        [ ] Enable compiler OpenMP support (NEW)
        [ ] Enable graphite support (NEW)
            *** Host GDB Options ***
        [ ] Build cross gdb for the host (NEW)
            *** Toolchain Generic Options ***
        () Target Optimizations (NEW)
        () Target linker options (NEW)
        [ ] Register toolchain within Eclipse Buildroot plug-in (NEW)
```



Toolchain

- » Easy install different platform toolchain
- » Easy change toolchain version
- » Easy custom toolchain



System Configure

```
System configuration
menus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pre
</> for Search. Legend: [*] feature is selected [ ] feature is excluded

Root FS skeleton (default target skeleton) --->
(buildroot) System hostname (NEW)
(Welcome to Buildroot) System banner (NEW)
    Passwords encoding (md5) --->
        Init system (BusyBox) --->
            /dev management (Dynamic using devtmpfs only) --->
            (system/device_table.txt) Path to the permission tables (NEW)
            [ ] support extended attributes in device tables (NEW)
            [ ] Use symlinks to /usr for /bin, /sbin and /lib (NEW)
            [*] Enable root login with password (NEW)
            () Root password (NEW)
            /bin/sh (busybox' default shell) --->
            [*] Run a getty (login prompt) after boot (NEW) --->
            [*] remount root filesystem read-write during boot (NEW)
            () Network interface to configure through DHCP (NEW)
            [*] Purge unwanted locales (NEW)
            (C en_US) Locales to keep (NEW)
                *** NLS support needs a toolchain w/ wchar, dynamic library ***
            [ ] Install timezone info (NEW)
            () Path to the users tables (NEW)
            () Root filesystem overlay directories (NEW)
            () Custom scripts to run before creating filesystem images (NEW)
            () Custom scripts to run inside the fakeroot environment (NEW)
            () Custom scripts to run after creating filesystem images (NEW)
```



System Configure

- » Init system
 - » BusyBox
 - » SystemV
- » Password setting
- » Login setting
- » System banner
- » RootFS skeleton



Target packages

```
-*- BusyBox
(package/busybox/busybox.config) BusyBox configuration file to use? (NEW)
()    Additional BusyBox configuration fragment files (NEW)
[ ]    Show packages that are also provided by busybox (NEW)
[ ]    Individual binaries (NEW)
[ ]    Install the watchdog daemon startup script (NEW)
[ ] rockchip BSP packages (NEW)  ---
    Audio and video applications  --->
    Compressors and decompressors  --->
    Debugging, profiling and benchmark  --->
    Development tools  --->
    Filesystem and flash utilities  --->
    Fonts, cursors, icons, sounds and themes  --->
    Games  --->
    Graphic libraries and applications (graphic/text)  --->
    Hardware handling  --->
    Interpreter languages and scripting  --->
    Libraries  --->
    Mail  --->
    Miscellaneous  --->
    Networking applications  --->
    Package managers  --->
    Real-Time  --->
    Security  --->
    Shell and utilities  --->
    System tools  --->
    Text editors and viewers  --->
    Libreto cores and retroarch  --->
```



Target packages

```
-*- BusyBox
(package/busybox/busybox.config) BusyBox configuration file to use? (NEW)
()    Additional BusyBox configuration fragment files (NEW)
[ ]    Show packages that are also provided by busybox (NEW)
[ ]    Individual binaries (NEW)
[ ]    Install the watchdog daemon startup script (NEW)
[ ] rockchip BSP packages (NEW)  ---
    Audio and video applications  --->
    Compressors and decompressors  --->
    Debugging, profiling and benchmark  --->
    Development tools  --->
    Filesystem and flash utilities  --->
    Fonts, cursors, icons, sounds and themes  --->
    Games  --->
    Graphic libraries and applications (graphic/text)  --->
    Hardware handling  --->
    Interpreter languages and scripting  --->
    Libraries  --->
    Mail  --->
    Miscellaneous  --->
    Networking applications  --->
    Package managers  --->
    Real-Time  --->
    Security  --->
    Shell and utilities  --->
    System tools  --->
    Text editors and viewers  --->
    Libreto cores and retroarch  --->
```



Configure File

▶ \${BUILDROOT}/configs

```
rockchip_rk3326_recovery_defconfig
rockchip_rk3326_robot32_defconfig
rockchip_rk3326_robot64_defconfig
rockchip_rk3326_robot64_no_gpu_defconfig
rockchip_rk3328_defconfig
rockchip_rk3328_recovery_defconfig
rockchip_rk3399_defconfig
rockchip_rk3399pro_defconfig
rockchip_rk3399pro-npu_defconfig
rockchip_rk3399pro-npu-multi-cam_defconfig
rockchip_rk3399pro_recovery_defconfig
rockchip_rk3399_recovery_defconfig
rockchip_rv1108_defconfig
roseapplepi_defconfig
s6lx9_microboard_defconfig
sheevaplug_defconfig
snps_aarch64_vdk_defconfig
snps_arc700_axs101_defconfig
snps_archs38_axs103_defconfig
snps_archs38_haps_defconfig
snps_archs38_vdk_defconfig
socrates_cyclone5_defconfig
solidrun_macchiatobin_mainline_defconfig
solidrun_macchiatobin_marvell_defconfig
stm32f429_disco_defconfig
```



Output Directory

➤ \${BUILDROOT}/output/rockchip

➤ build

➤ Host → Toolchain

➤ Image → Image file

- rootfs.cpio, rootfs.cpio.gz
- rootfs.ext2 , rootfs.ext4
- rootfs.squashfs
- rootfs.tar
- Linux kernel , u-boot Image

➤ target

Nanopi-m4 SDK



Nanopi-m4 Build System

► Reference Buildroot

► http://wiki.friendlyarm.com/wiki/index.php/Buildroot_for_RK3399

► It will build and output

- Kernel Image
- Boot-loader image
- RootFS
 - System setting file (/etc)
 - Driver modules
 - Application
 - library



Nanopi-m4 Build System

```
./linuxsdk-friendlyelec
└── [Mar 31  8:40]  app
└── [Mar 31  8:40]  buildroot
└── [Mar 31  8:40]  build.sh -> device/rockchip/common/build.sh
└── [Mar 31  8:40]  device
└── [Mar 31  8:40]  distro
└── [Mar 31  8:40]  docs
└── [Mar 31  8:40]  envsetup.sh -> buildroot/build/envsetup.sh
└── [Mar 31  8:41]  external
└── [Mar 31  8:41]  friendlyelec
└── [Apr 20  16:15]  kernel
└── [Mar 31  8:40]  Makefile -> buildroot/build/Makefile
└── [Mar 31  8:40]  mkfirmware.sh -> device/rockchip/common/mkfirmware.sh
└── [Mar 31  8:41]  out -> friendlyelec/rk3399/sd-fuse_rk3399/out
└── [Mar 31  8:41]  prebuilt
└── [Mar 31  8:41]  rkbin
└── [Mar 31  8:40]  rkflash.sh -> device/rockchip/common/rkflash.sh
└── [Mar 31  8:41]  rootfs
└── [Mar 31  8:42]  tools
└── [Mar 31  8:42]  u-boot
└── [Mar 31  8:42]  yocto
```



Setup Compilation Environment

▶ Install Package on host machine

```
sudo apt-get install repo git-core gitk git-gui u-boot-tools device-tree-compiler mtools parted libudev-dev libusb-1.0-0-dev python-linaro-image-tools linaro-image-tools autoconf autotools-dev libsigsegv2 m4 intltool libdrm-dev curl sed make binutils build-essential gcc g++ bash patch gzip bzip2 perl tar cpio python unzip rsync file bc wget libncurses5 libqt4-dev libglib2.0-dev libgtk2.0-dev libglade2-dev cvs git mercurial rsync openssh-client subversion asciidoc w3m dblatex graphviz python-matplotlib libc6:i386 libssl-dev texinfo liblz4-tool genext2fs lib32stdc++6
```



repo

▶ Install repo Utility

Step 1 :

```
git clone https://github.com/friendlyarm/repo
```

Step 2 :

```
cp repo/repo /usr/bin/
```



Download Source Code

► Fix a version

Step 1 :

```
tar xvf linuxsdk-friendlyelec-YYYYMMDD.tar
```

Step 2 :

```
cd linuxsdk-friendlyelec
```

Step 3 :

```
repo sync -l --no-clone-bundle
```



Download Source Code

▶ Retrive Repo Package from Github

Step 1 :

```
mkdir linuxsdk-friendlyelec
```

Step 2 :

```
cd linuxsdk-friendlyelec
```

Step 3 :

```
repo init -u https://github.com/friendlyarm/buildroot_manifests  
-b master -m rk3399_linux_release.xml --repo-  
url=https://github.com/rockchip-linux/repo -no-clone-bundle
```

Step 4 :

```
repo sync -c --no-clone-bundle
```



Compilation

▶ Build Image

- ▶ Build ALL → ./build.sh
- ▶ Build Kernel → ./build.sh kernel
- ▶ Build u-boot → ./build.sh uboot
- ▶ Build RootF → ./build.sh rootfs



Create Image

▶ SDCard Image

▶ sudo ./build.sh sd-img



Create Image

▶ eMMC Image

▶ sudo ./build.sh emmc-img



Create Image

▶ EMMC Image

▶ `sudo ./build.sh emmc-img`



Build Configure File

▶ Configure file

▶ device/rockchip/rk3399/BoardConfig.mk

▶ Use setting build environment variable

▶ # Target arch

- export RK_ARCH=arm64

▶ # uboot defconfig

- export RK_UBOOT_DEFCONFIG=rk3399_defconfig

▶ # Kernel defconfig

- export
RK_KERNEL_DEFCONFIG=nanopi4_linux_defconfig

Rockchip Driver

- ▶ external
- ▶ Put many depend hardware software package

```
external/
└── [Mar 31  8:40]  alsound
└── [Mar 31  8:40]  audioservice
└── [Mar 31  8:40]  bluez-alsa
└── [Mar 31  8:40]  camera_engine_cifisp
└── [Mar 31  8:40]  camera_engine_rkisp
└── [Mar 31  8:40]  ffmpeg
└── [Mar 31  8:40]  gstreamer-camera
└── [Mar 31  8:40]  gstreamer-rockchip
└── [Mar 31  8:40]  libdrm
└── [Mar 31  8:40]  libmali
└── [Mar 31  8:40]  linux-rga
└── [Mar 31  8:40]  minigui
└── [Mar 31  8:40]  mpp
└── [Mar 31  8:40]  mpv
└── [Mar 31  8:40]  powermanager
└── [Mar 31  8:40]  recovery
└── [Mar 31  8:40]  rknn_demo
└── [Mar 31  8:40]  rknn-toolkit
└── [Mar 31  8:40]  rk_pcba_test
└── [Mar 31  8:40]  rkscript
└── [Mar 31  8:40]  rkssd
└── [Mar 31  8:40]  rkupdate
└── [Mar 31  8:40]  rkwifibt
└── [Mar 31  8:40]  security
└── [Mar 31  8:41]  softapDemo
└── [Mar 31  8:41]  softapServer
└── [Mar 31  8:41]  tensorflow
└── [Mar 31  8:41]  uvc_app
```



Toolchain

▶ Prebuilt/

```
slash@slash-HD631-Q87CRM:linuxsdk-friendlyelec$ prebuilts/gcc/linux-x86/aarch64/gcc-linaro-6.3.1-2017.05-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-gcc -v
Using built-in specs.
COLLECT_GCC=prebuilts/gcc/linux-x86/aarch64/gcc-linaro-6.3.1-2017.05-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-gcc
COLLECT_LTO_WRAPPER=/home/slash/work/special_task/cadtc/rk3399/linuxsdk-friendlyelec/prebuilts/gcc/linux-x86/aarch64/gcc-linaro-6.3.1-2017.05-x86_64_aarch64-linu
64-linux-gnu/6.3.1/lto-wrapper
Target: aarch64-linux-gnu
Configured with: '/home/tcwg-buildslave/workspace/tcwg-make-release/builder_arch/amd64/label/tcwg-x86_64-build/target/aarch64-linux-gnu/snapshots/gcc.git-linaro-
bin/bash --with-mpc=/home/tcwg-buildslave/workspace/tcwg-make-release/builder_arch/amd64/label/tcwg-x86_64-build/target/aarch64-linux-gnu/_build/builds/destdir/x
-mpfr=/home/tcwg-buildslave/workspace/tcwg-make-release/builder_arch/amd64/label/tcwg-x86_64-build/target/aarch64-linux-gnu/_build/builds/destdir/x86_64-unknown-
g-buildslave/workspace/tcwg-make-release/builder_arch/amd64/label/tcwg-x86_64-build/target/aarch64-linux-gnu/_build/builds/destdir/x86_64-unknown-linux-gnu --wit
le-libmudflap --enable-lto --enable-shared --without-included-gettext --enable-nls --disable-sjlj-exceptions --enable-gnu-unique-object --enable-linker-build-id
le-c99 --enable-clocale=gnu --enable-libstdcxx-debug --enable-long-long --with-cloog=no --with-ppl=no --with-isl=no --disable-multilib --enable-fix-cortex-a53-83
43419 --with-arch=armv8-a --enable-threads=posix --enable-multiarch --enable-libstdcxx-time=yes --enable-gnu-indirect-function --with-build-sysroot=/home/tcwg-bu
elease/builder_arch/amd64/label/tcwg-x86_64-build/target/aarch64-linux-gnu/_build/sysroots/aarch64-linux-gnu --with-sysroot=/home/tcwg-buildslave/workspace/tcwg-
4/label/tcwg-x86_64-build/target/aarch64-linux-gnu/_build/builds/destdir/x86_64-unknown-linux-gnu/aarch64-linux-gnu/libc --enable-checking=release --disable-boot
,fortran,lto --build=x86_64-unknown-linux-gnu --host=x86_64-unknown-linux-gnu --target=aarch64-linux-gnu --prefix=/home/tcwg-buildslave/workspace/tcwg-make-relea
wg-x86_64-build/target/aarch64-linux-gnu/_build/builds/destdir/x86_64-unknown-linux-gnu
Thread model: posix
gcc version 6.3.1 20170404 (Linaro GCC 6.3-2017.05)
slash@slash-HD631-Q87CRM:linuxsdk-friendlyelec$
```



Other Folders

- » kernel, u-boot
- » friendlyelec
 - » Nano Pi M4 build distribution folder
- » rootfs
 - » Build debian and ubuntu RootFS
- » app
 - » Special user space application
- » Buildroot
 - » Linux distribution build reference



Prebuild Image and Setting File

friendlyelec/rk3399/sd-fuse_rk3399/prebuilt

- idbloader.img
- trust.img
- boot.img
- uboot.img
- generic
 - param4sd.txt
 - partmap.txt



param4sd.txt

```
FIRMWARE_VER: 6.0.1
MACHINE_MODEL: RK3399
MACHINE_ID: 007
MANUFACTURER: RK3399
MAGIC: 0x5041524B
ATAG: 0x00200800
MACHINE: 3399
CHECK_MASK: 0x80
PWR_HLD: 0,0,A,0,1
#KERNEL_IMG: 0x00280000
#FDT_NAME: rk-kernel.dtb
#RECOVER_KEY: 1,1,0,20,0
#in section; per section 512(0x200) bytes
CMDLINE: root=/dev/mmcblk0p1 rw rootfstype=ext4
mtdparts=rk29xxnand:0x00002000@0x00002000(uboot),
0x00002000@0x00004000(trust),
0x00002000@0x00006000(misc),
0x00006000@0x00008000(resource),
0x00010000@0x0000e000(kernel),
0x00010000@0x0001e000(boot),
-@0x00030000(rootfs)
```



partmap.txt

```
flash=mmc,1:loader:idb:0x8000,0x280000:idbloader.img;
flash=mmc,1:env:env:0x3F8000,0x8000;
flash=mmc,1:parm:parm:0x400000,0x0400000:param4sd.txt;
flash=mmc,1:uboot:raw:0x800000,0x0400000:uboot.img;
flash=mmc,1:trust:raw:0xC00000,0x0400000:trust.img;
flash=mmc,1:misc:raw:0x1000000,0x0400000;
flash=mmc,1:resc:raw:0x1400000,0x0C00000:resource.img;
flash=mmc,1:kern:raw:0x2000000,0x2000000:kernel.img;
flash=mmc,1:boot:raw:0x4000000,0x2000000:boot.img;
flash=mmc,1:rootfs:ext4:0x6000000,0x0:rootfs.img;
```



Update Tool

▶ sd_update

[linuxsdk-friendlyelec/friendlyelec/rk3399/sd-fuse_rk3399/tools](https://github.com/linuxsdk-friendlyelec/friendlyelec/rk3399/sd-fuse_rk3399/tools)

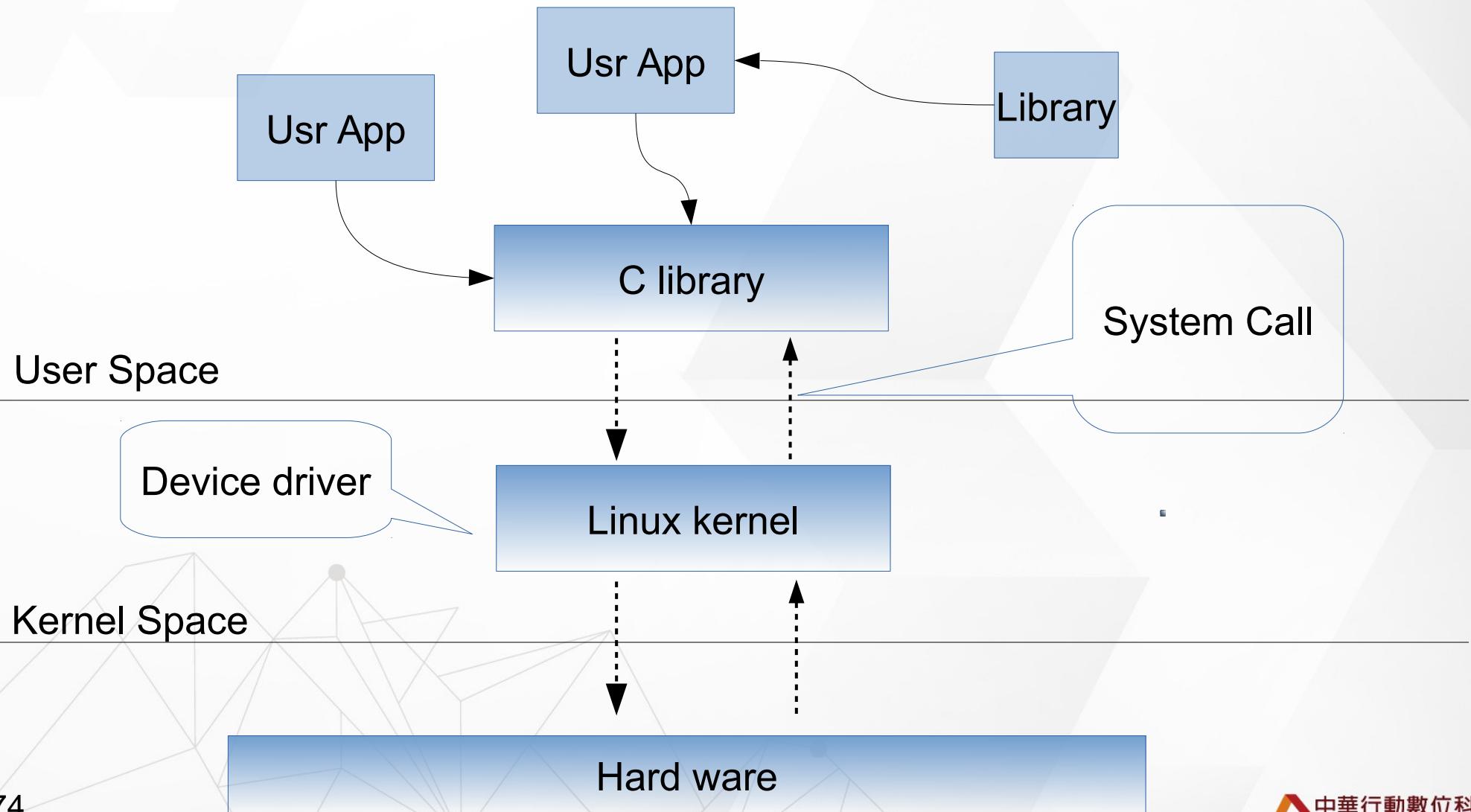
```
Usage: sd_update [ARGS]

Options:
  -d <device node>          default: none
  -p <partmap file>          default: ./partmap.txt
  -i <images path>           default is "partmap.txt" directory
  -r <raw image file>
  -s                          show device partition only
  -f                          force to no warning
  -h                          print this help text

partmap file:
flash=<device>.<dev no>:<partition>:<fstype>:<start>,<length>[:file name];
  <device>                  device name
  <dev no>                  device number
  <partition>                partition name
  <fstype>                  filesystem type, MBR = raw, fat, ext4
  <start>                   partition start address (hex)
  <length>                  partition length (hex)
  <file name>               write file to partition
```

CH9 Linux User Land

Linux kernel





Sys Filesystem

- » Allows kernel code to export information to user processes
- » sysfs is an in-memory filesystem
- » 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



Sys File System

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

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

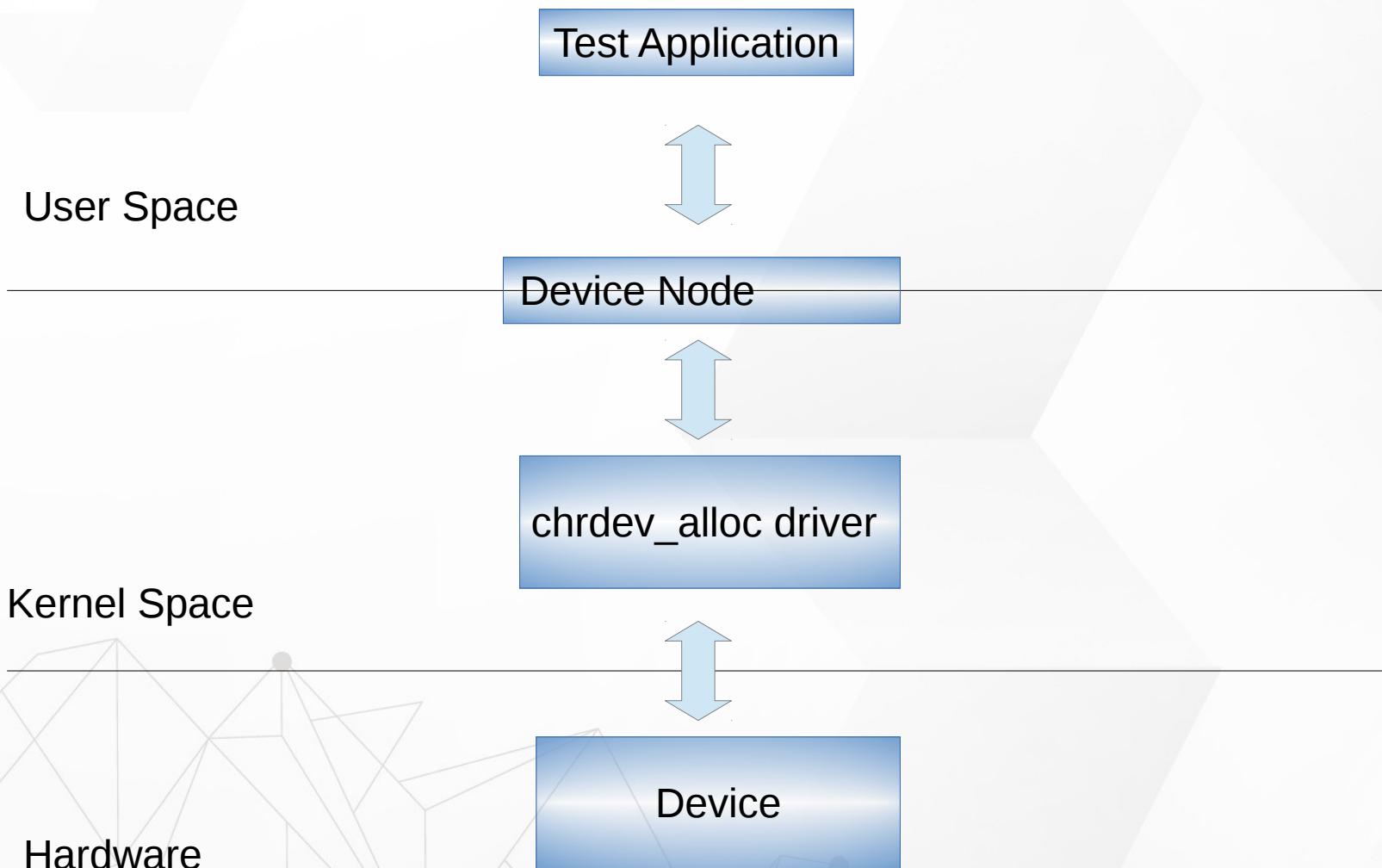
```
# tree -L 1 /sys/class/i2c-dev/i2c-0/
```

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

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

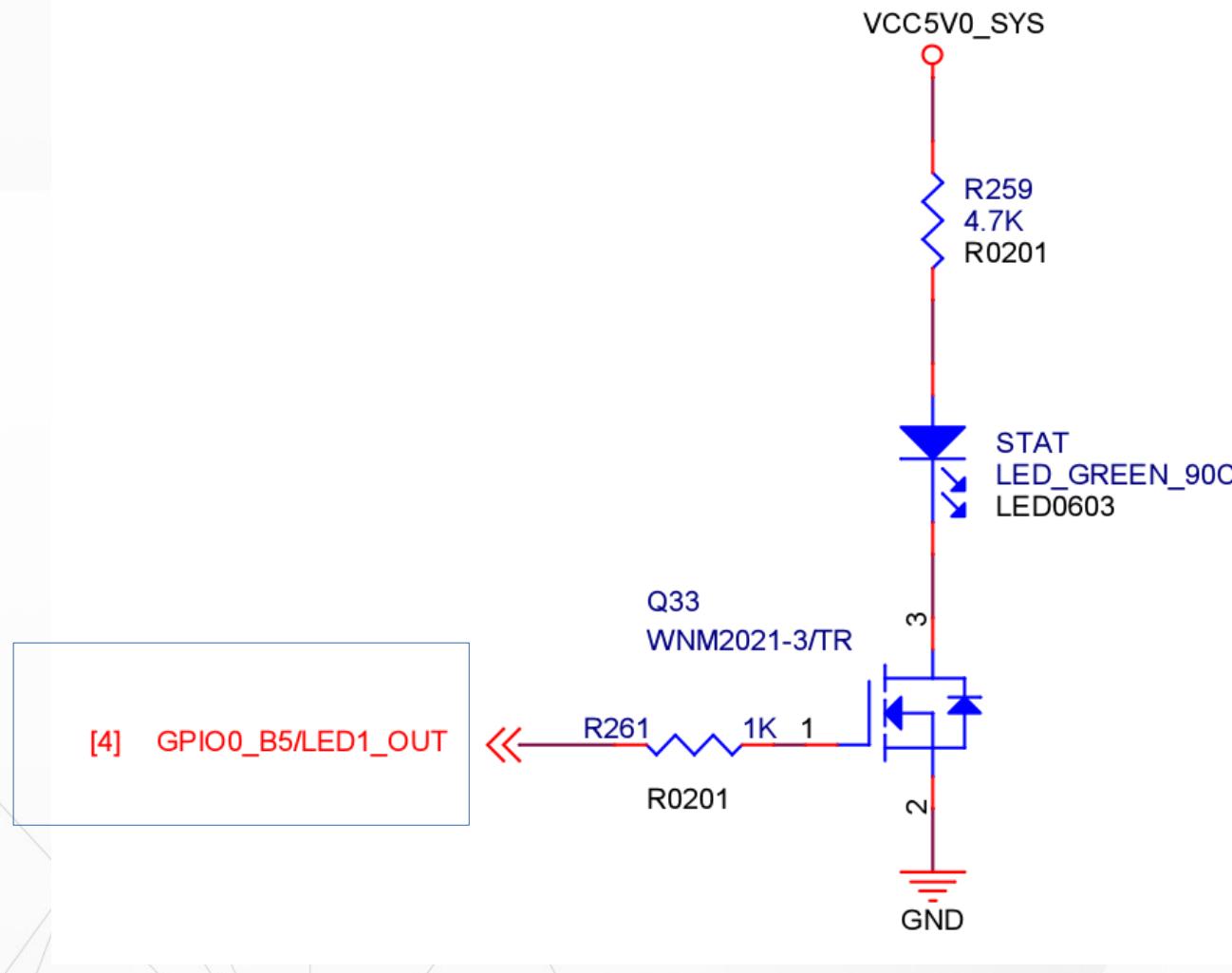
```
/sys/class/i2c-dev/
├── i2c-0 -> ../../devices/pci0000:00/0000:00:02.0/i2c-0/i2c-dev/i2c-0
├── i2c-1 -> ../../devices/pci0000:00/0000:00:02.0/i2c-1/i2c-dev/i2c-1
├── i2c-2 -> ../../devices/pci0000:00/0000:00:02.0/i2c-2/i2c-dev/i2c-2
├── i2c-3 -> ../../devices/pci0000:00/0000:00:02.0/i2c-3/i2c-dev/i2c-3
├── 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
```

User land and Driver



LED Drivers

LED Schematic





LED Subsystem

▶ Control LED convenient with SysFS

▶ For example

- echo 1 > /sys/class/leds/status_led/shot

▶ Switch different LED trigger type in SysFS

▶ For example

- echo "oneshot" > leds/status_led/trigger
- echo "heartbeat" >
/sys/class/leds/status_led/trigger

LED SysFS

```
# ls leds/status_led/ -l  
  
brightness  
device -> ../../../../../gpio-leds  
invert  
max_brightness  
power  
subsystem -> ../../../../../../class/leds  
trigger  
uevent
```

```
# cat leds/status_led/trigger
```

Check trigger	none rc-feedback kbd-scrolllock kbd-numlock kbd-capslock kbd-kanalock kbd-shiftlock kbd-ctrllock kbd-ctrlrlock mmc0 mmc1 timer oneshot [heartbeat] rfkill1 rfkill2
---------------	--

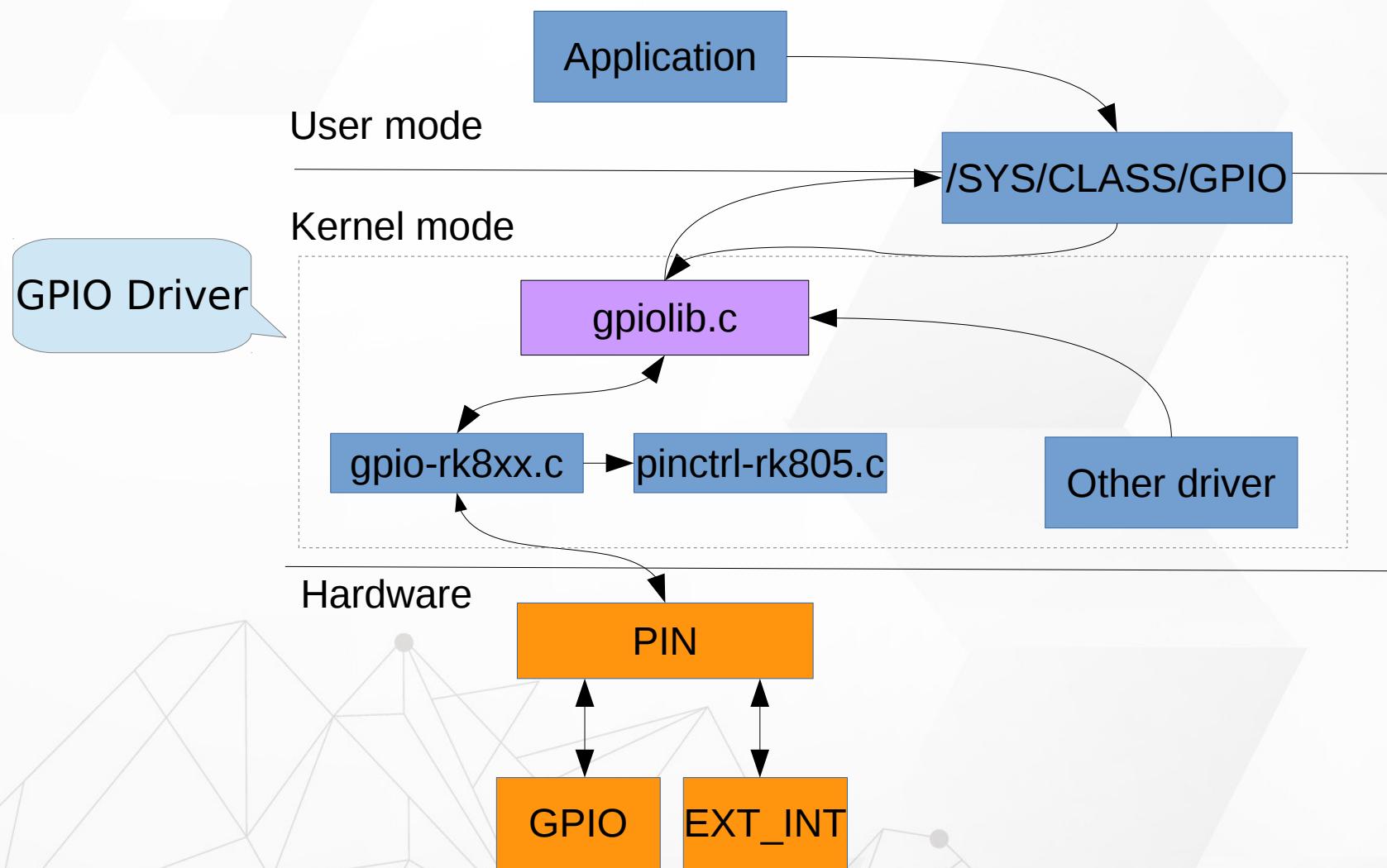
Switch trigger type

```
[root@rk3399:/sys/class]# echo "oneshot" > leds/status_led/trigger  
[root@rk3399:/sys/class]# cat leds/status_led/trigger
```

none rc-feedback kbd-scrolllock kbd-numlock kbd-capslock kbd-kanalock kbd-shiftlock kbd-ctrllock kbd-ctrlrlock mmc0 mmc1 timer [oneshot] [heartbeat] gpio rfkill1 rfkill2

GPIO Control

GPIO Subsystem





Driver LED in User Space

▶ Paths in Sysfs

▶ /sys/class/gpio:

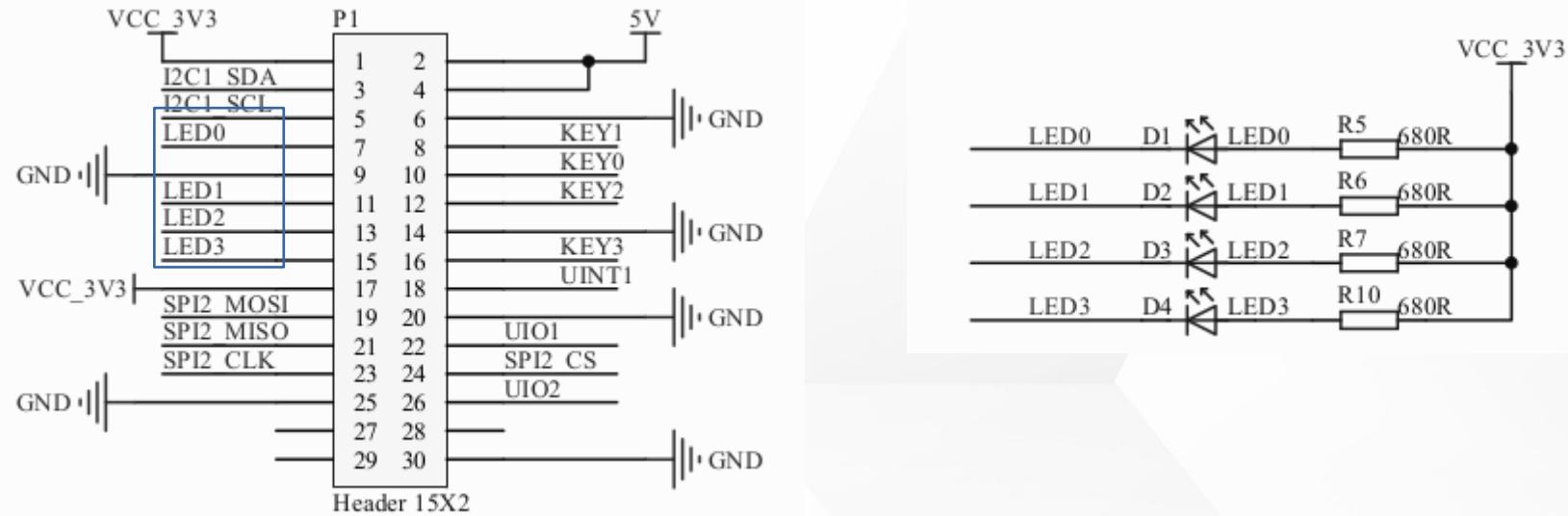
- ▶ Control interfaces used to get userspace control over GPIOs;
- ▶ GPIOs themselves
- ▶ GPIO controllers("gpio_chip" instances)

▶ /sys/class/gpio/

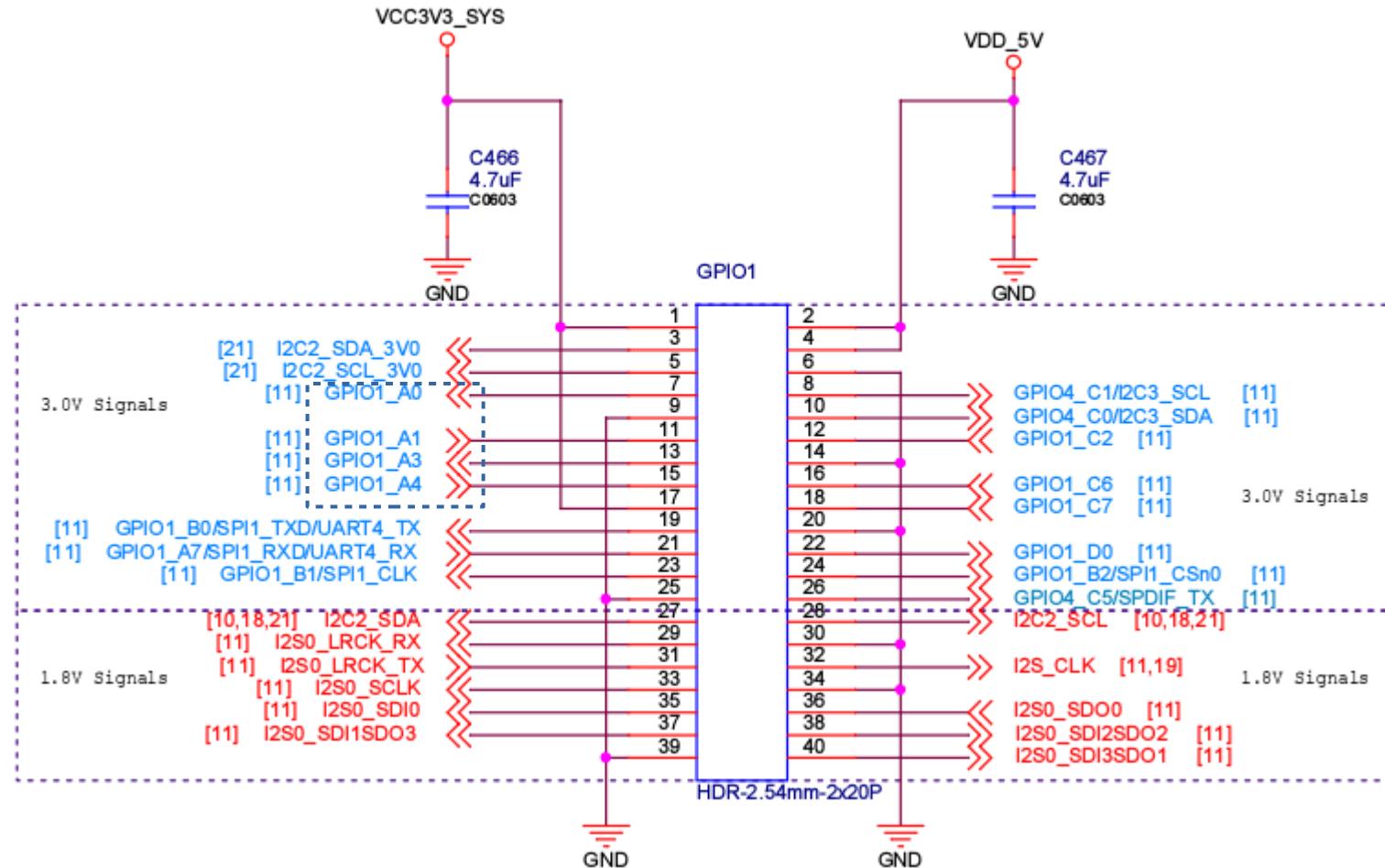
- ▶ "export" : ask the kernel to export GPIO to userspace by writing
 - "echo 19 > export"
 - create a "gpio19" node in /sys/class/gpio

- ▶ "unexport" : Reverses the effect of exporting to userspace
 - "echo 19 > unexport"
 - remove "gpio19" node from /sys/class/gpio

NanoPi-M4 Ext Board LED



NanoPi-M4 GPIO1 HEAD





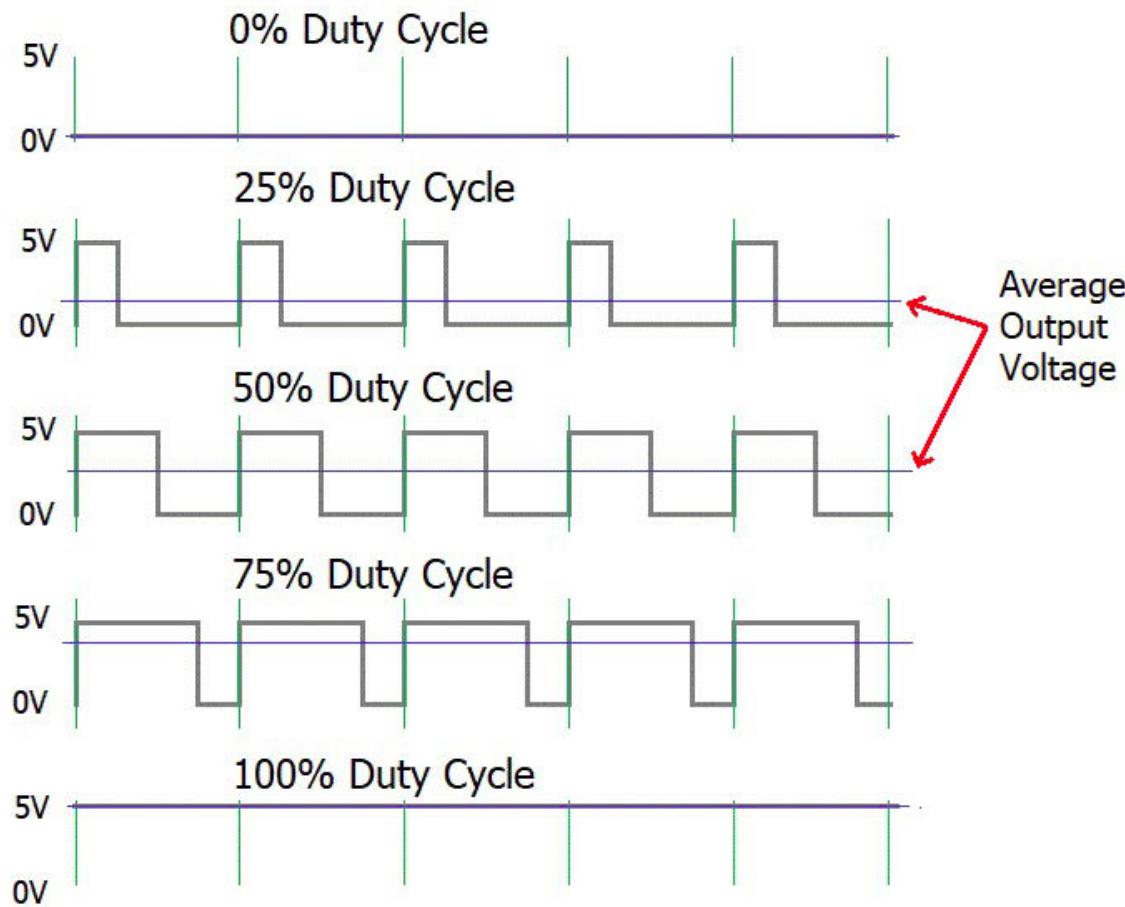
Exercise

- ▶ Use “/sys/class/gpio” to setting LED
- ▶ \$KERNEL/Documentation/gpio/sysfs.txt

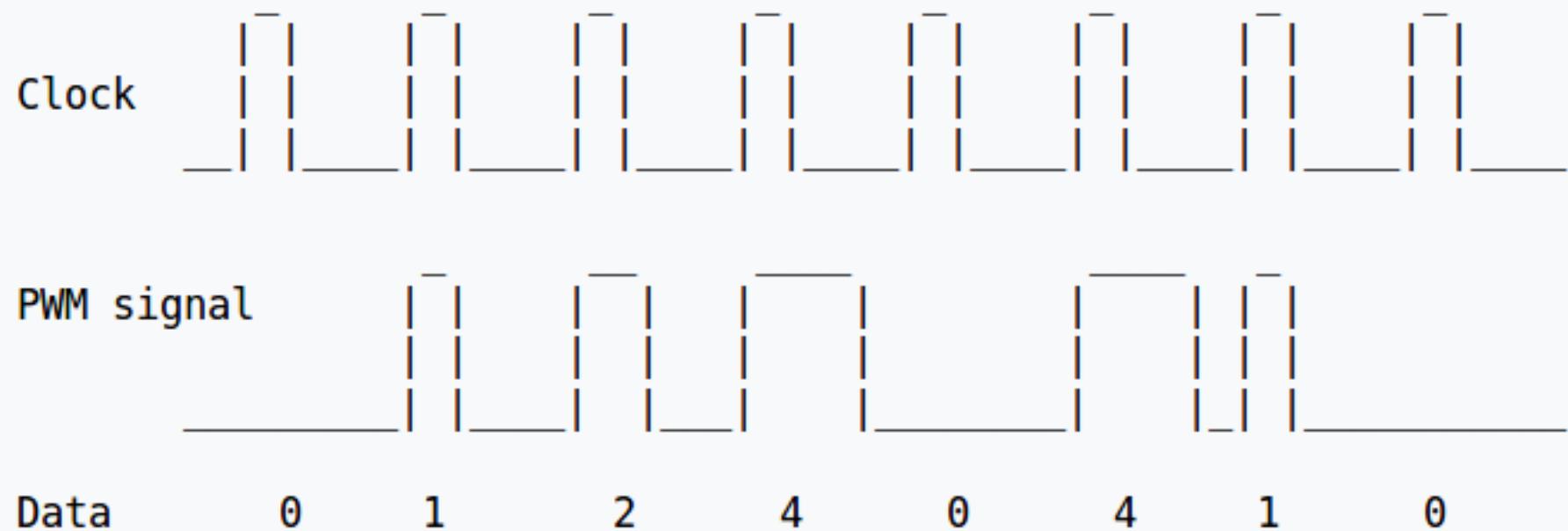
PWM Sub System

PWM

» PWM : Pulse Width Modulation



PWM



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

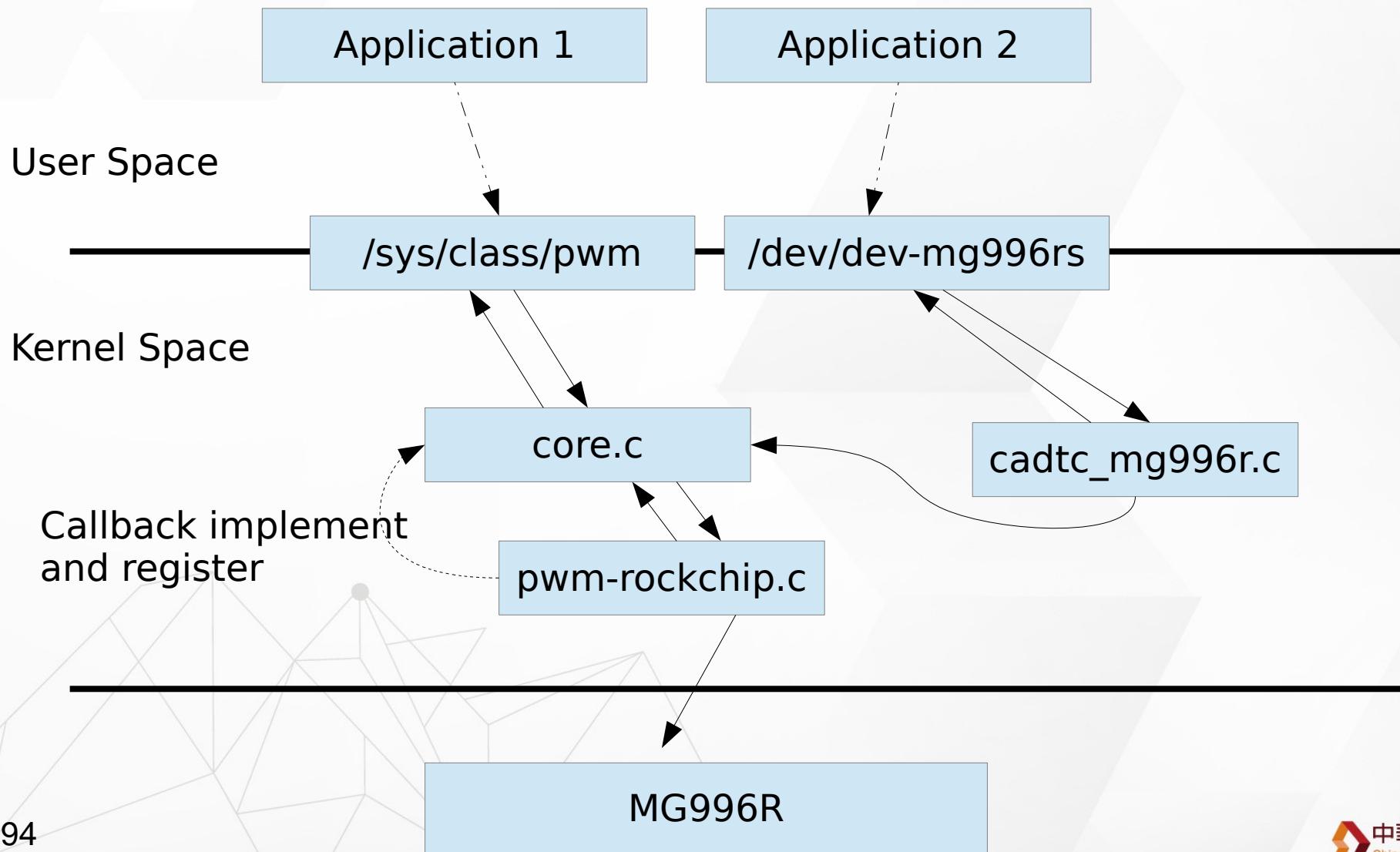
► Enable



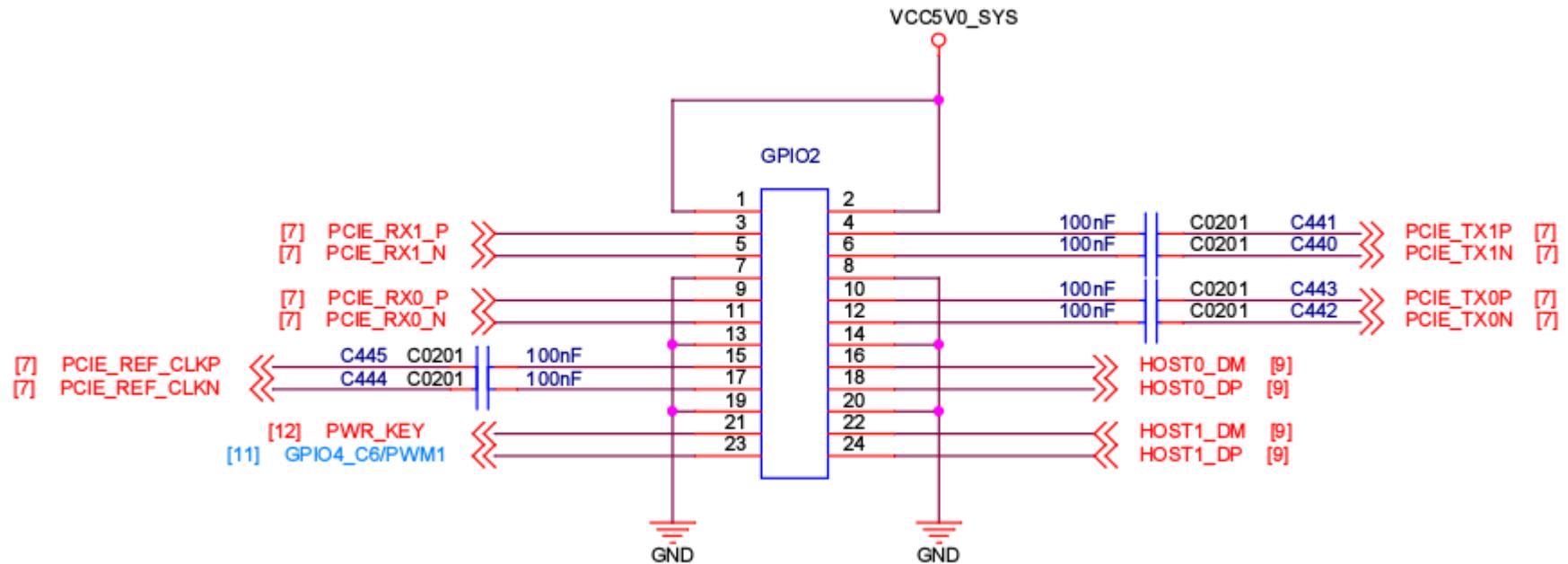
PWM Driver

- \$(KERNEL_SRC)/Documentation/pwm.txt
- Platform Driver
 - drivers/pwm/
 - drivers/pwm/core.c
 - drivers/pwm/pwm-rockchip.c

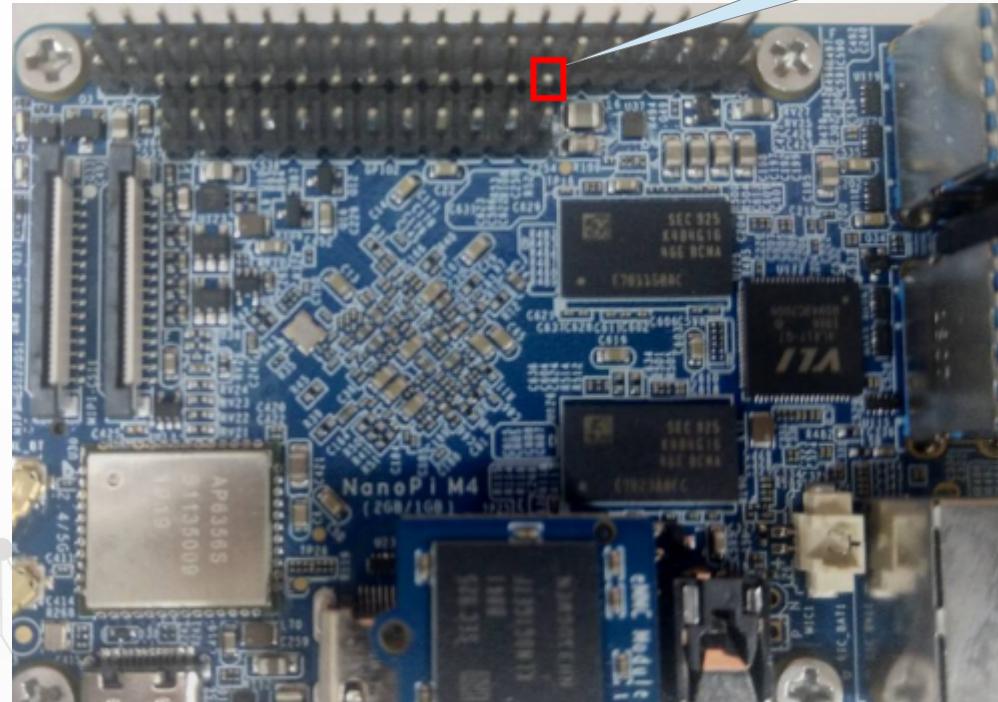
PWM Subsystem



NanoPi-M4 and PWM



NanoPi-M4 and PWM



GPIO4_C6PWM1



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 "20000000" > period      //20ms, 50 Hz
```

```
echo "2000000" > duty_cycle //2ms
```

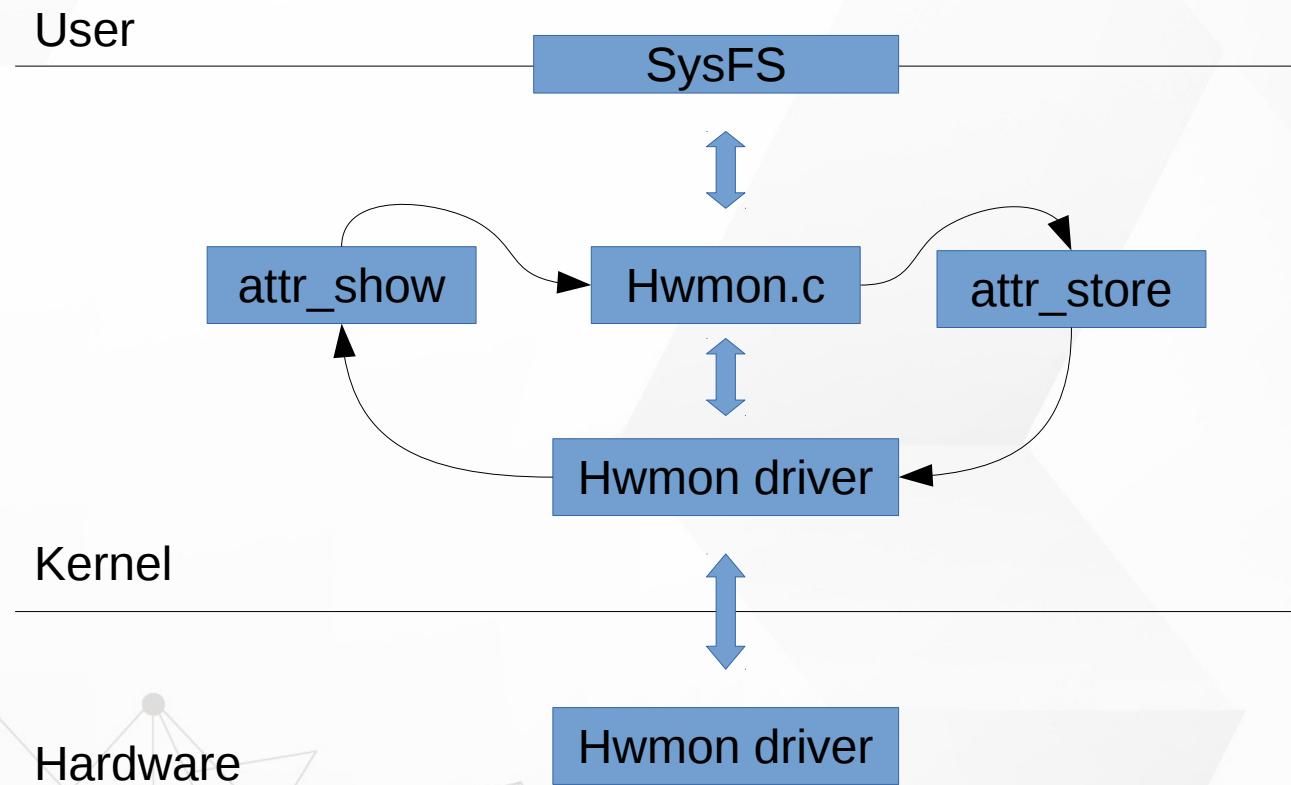
```
echo 1 > enable             //Enable
```

PWM DoReMi

	Frequency (Hz)
C4	261.63
C4#	277.18
D4	293.66
D4#	311.13
E4	329.63
F4	349.23
F4#	369.99
G4	392.00
G4#	415.30
A4	440.00
A4#	466.16
B4	493.88
C5	523.25

Hwmon Subsystem

Hwmon Subsystem

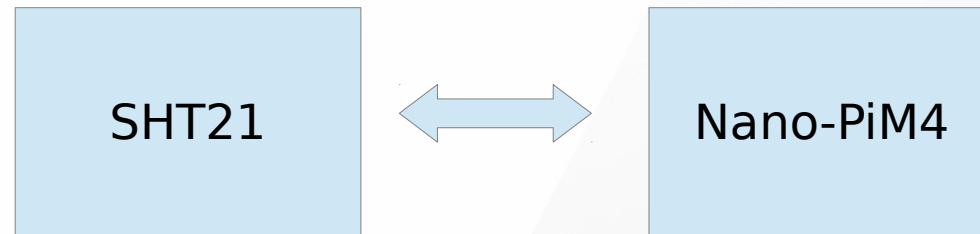




SHT21

- » Simple interface
- » Bus interface
 - » I2C, GPIO, SPI
- » Sensors
 - » Temperature
 - » Voltage
 - » Humidity
 - » Fan speed
 - » PWM control

SHT21



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

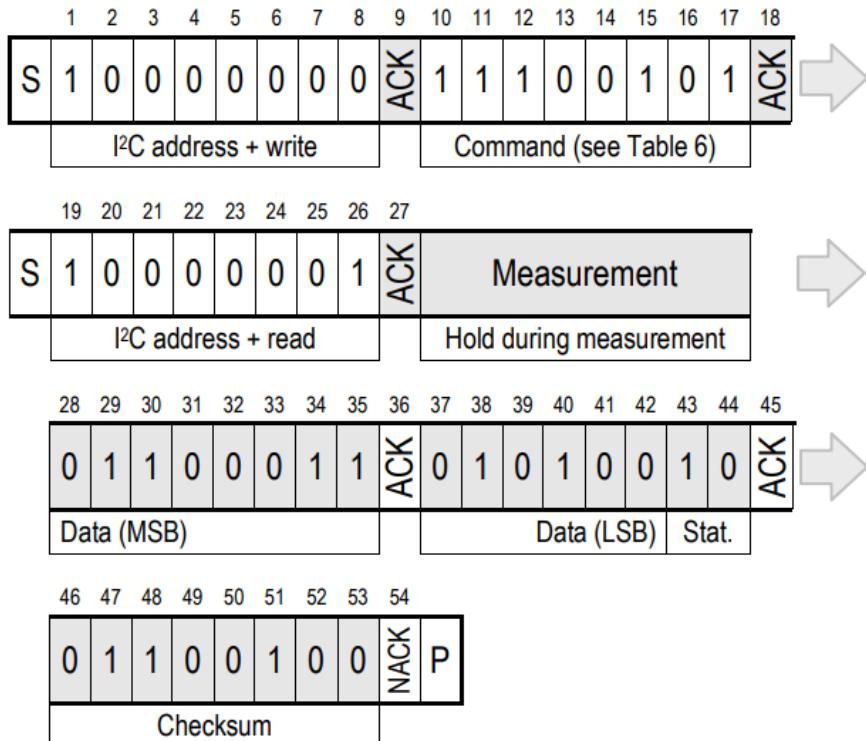
A pinout diagram for the SHT21 sensor. It shows a grey rectangular package with six pins numbered 1 through 6. Pin 1 is at the bottom right, pin 2 is at the top right, pin 3 is at the top left, pin 4 is at the bottom left, pin 5 is at the top center, and pin 6 is at the bottom center.

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

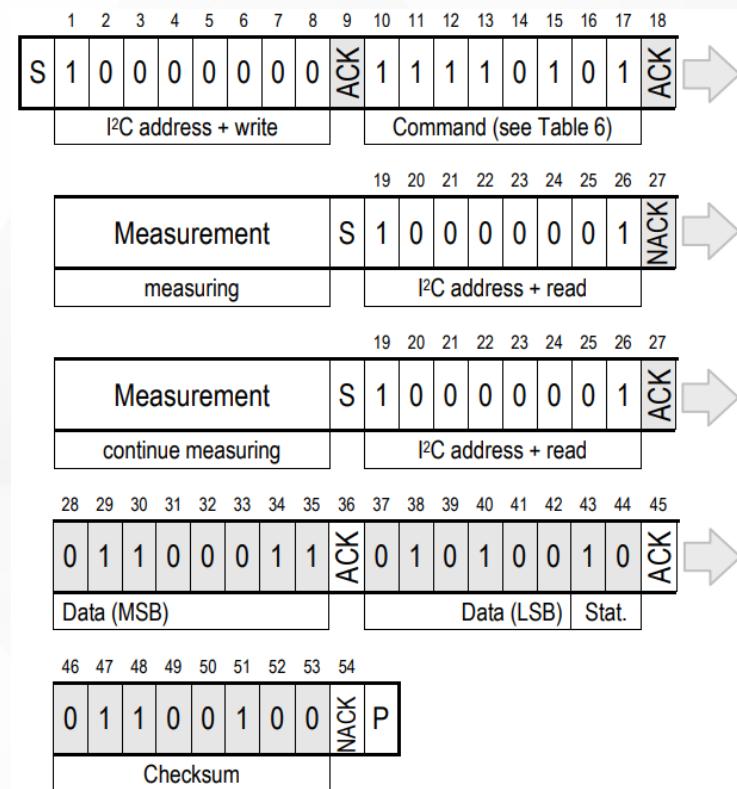


SHT21

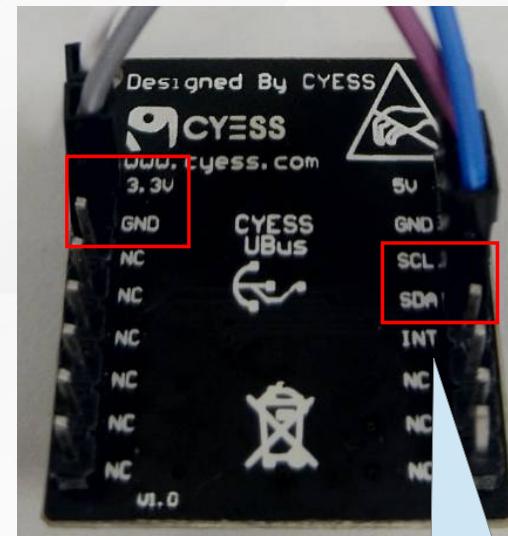
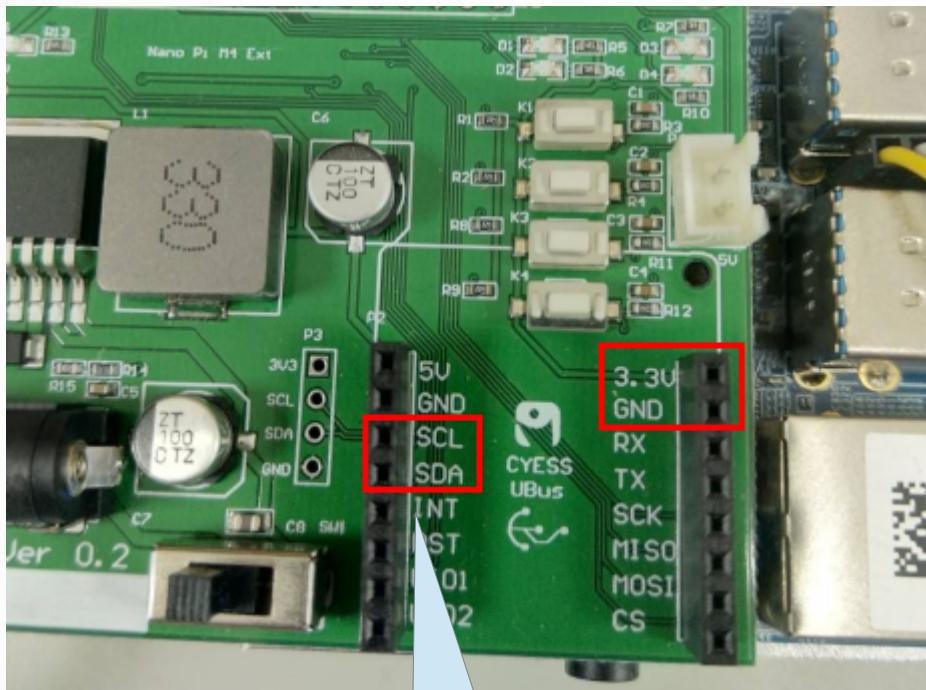
Hold master communication sequence



No Hold master communication sequence



NanoPi-M4 and SHT21





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
```

I/O Subsystem



IIO Introduction

- » IIO - The Industrial I/O
- » support for devices that in some sense
 - » analog to digital (ADC)
 - » digital to analog converters (DAC)
- » Devices that fall into this category are
 - » ADCs
 - » Accelerometers
 - » Gyros
 - » DAC
 - » Pressure Sensors

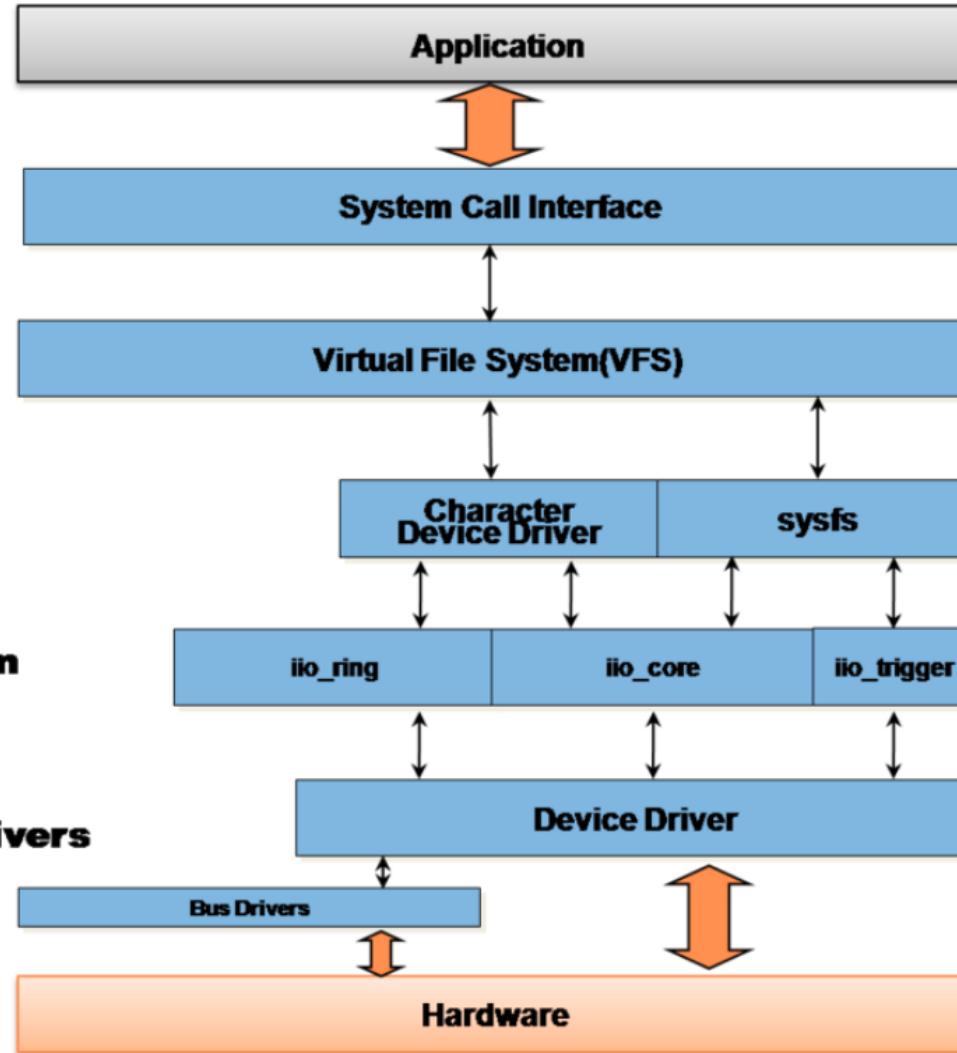


IIO Introduction

- Fill the gap between the somewhat similar hwmon and input subsystems
- Hwmon is very much directed at low sample rate sensors used in applications
 - fan speed control
 - temperature measurement.
- Input is, as it's name suggests focused on human interaction input devices

IIO Introduction

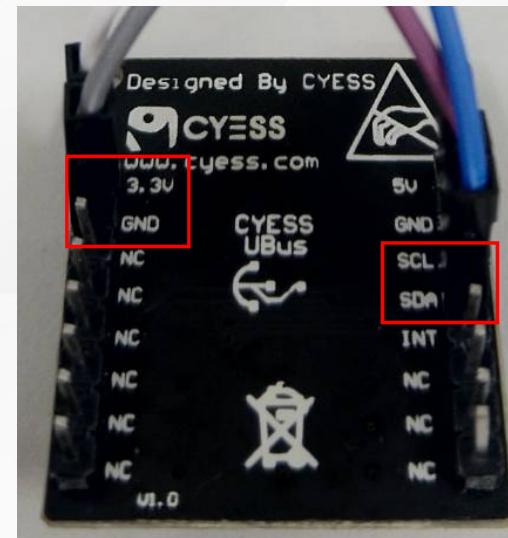
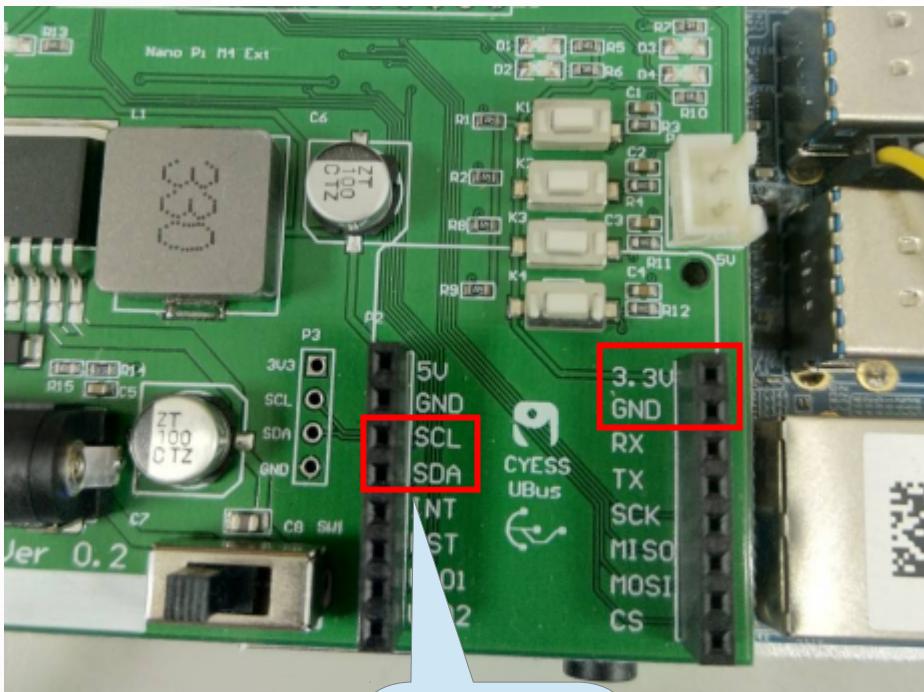
Application Area



IIO Interface

- » There are 2 ways for a user space application to interact with an IIO driver
- » **/sys/bus/iio/iio:deviceX/**
 - » data channels
- » **/dev/iio:deviceX**
 - » buffered data transfer
 - » events information

NanoPi-M4 and ISL29023





IIO and ISL29023

Get value from SysFS attribute

```
$ cat /sys/bus/iio/devices/iio:device1/in_illuminance0_input
```

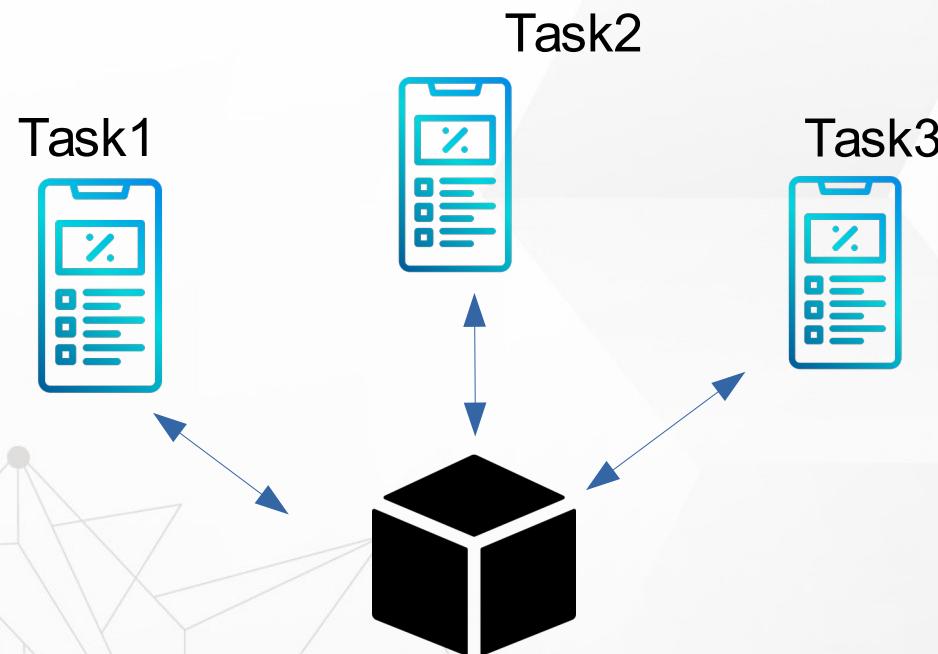
221

CH9 Linux Device Driver Module

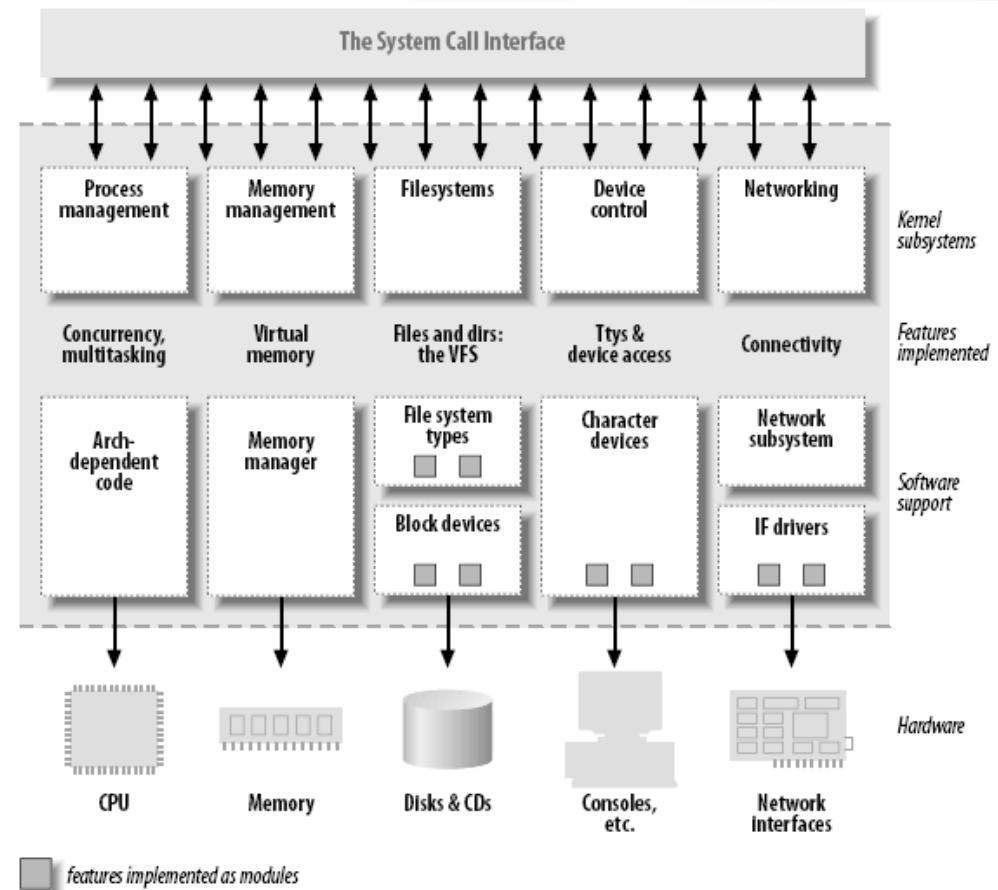
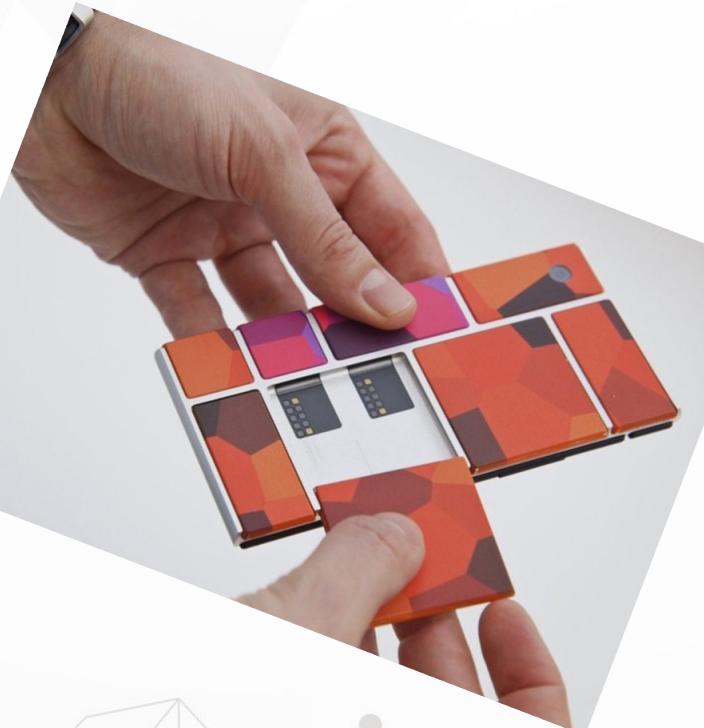
Introduction

▶ Device drivers

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



Kernel Modularization





Example

- » \$ make
- » \$ sudo insmod simple.ko
- » \$ dmesg | tail
- » \$ lsmod | grep simple
- » \$ sudo rmmod simple



Classes of Devices Driver

▶ Char module

- ▶ simple
- ▶ access stream of bytes

▶ Block module

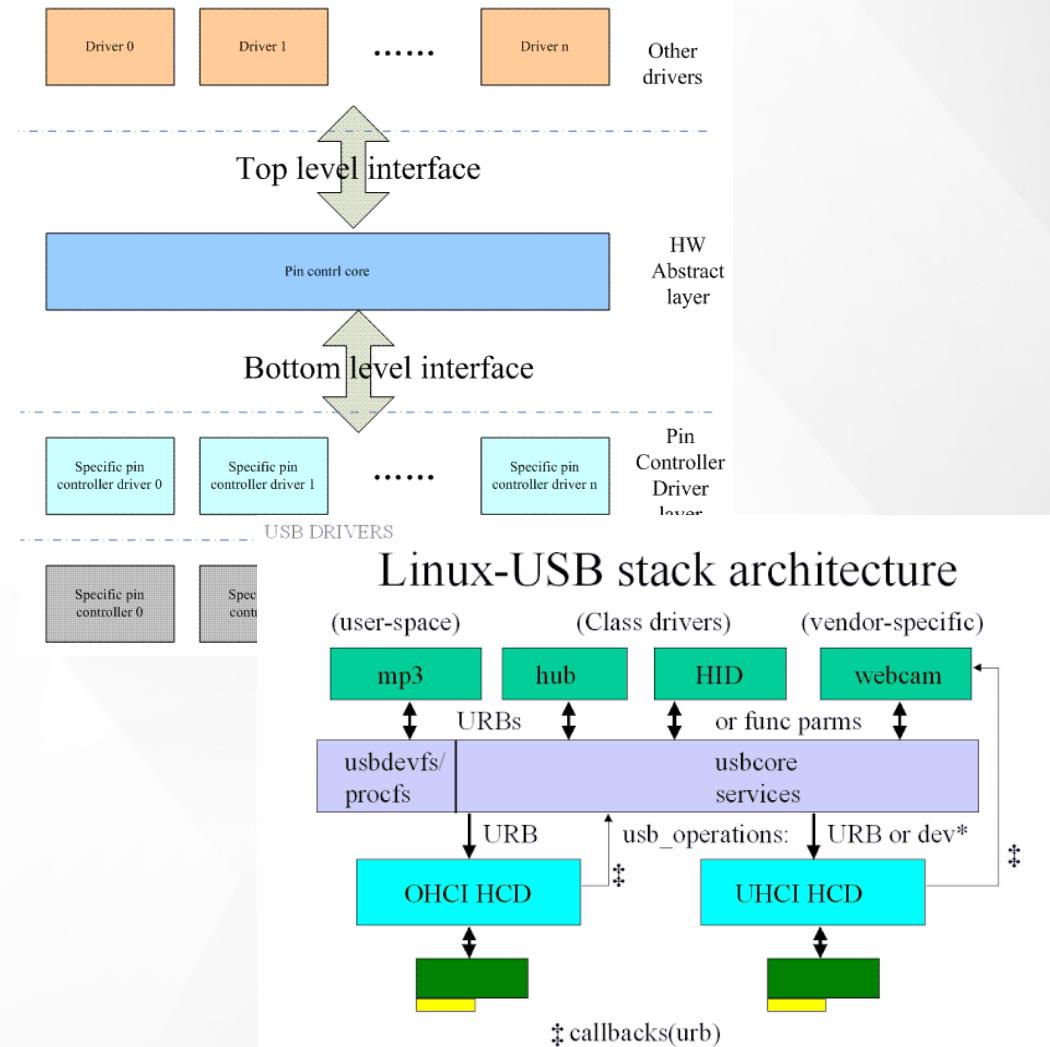
- ▶ block and char devices differ only in the way data is managed internally by the kernel

▶ Network module

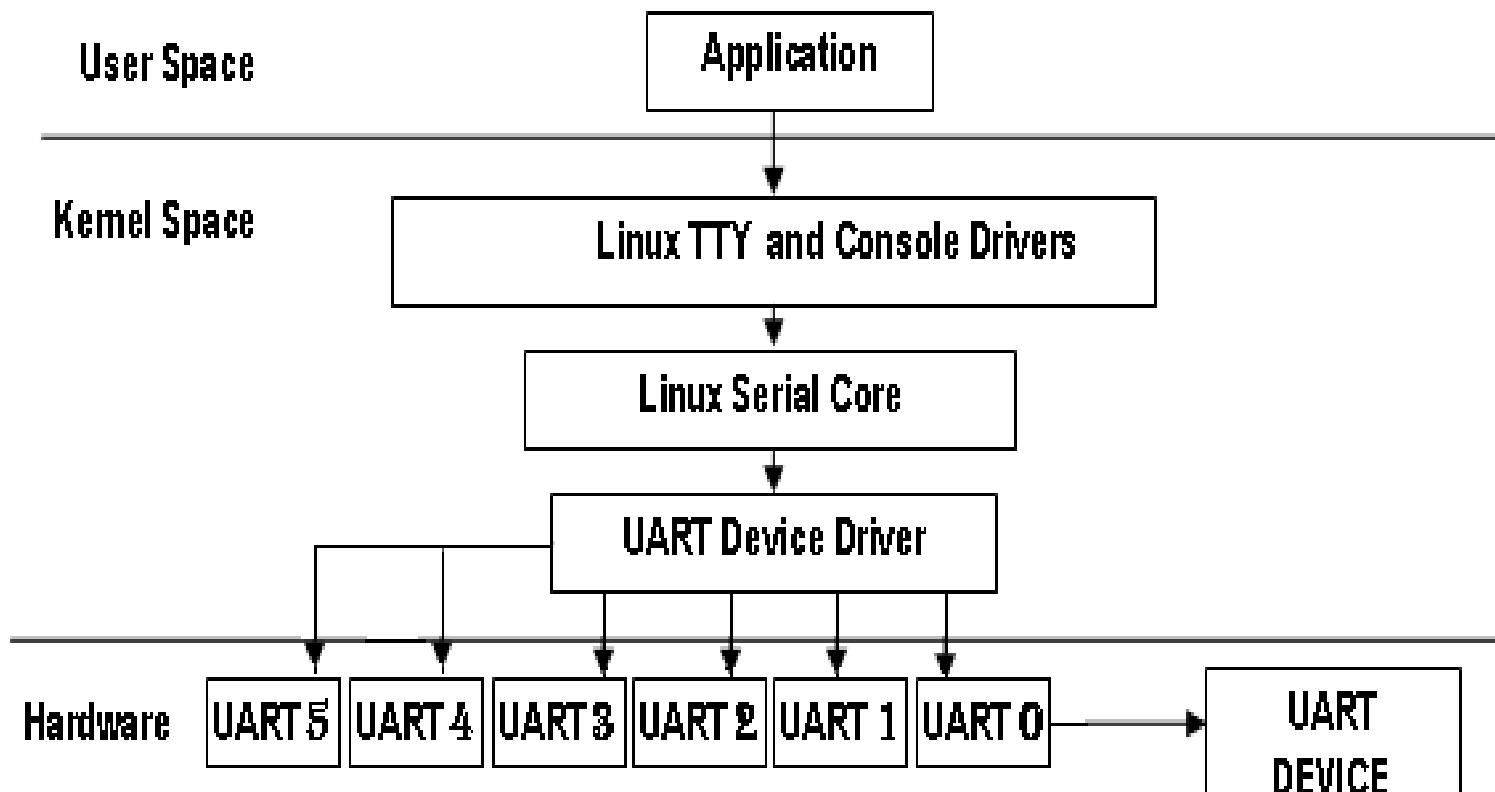
- ▶ Manage network data packets

Subsystem

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



Sub-system





Where are Modules in Kernel

➤ \${KERNEL}/drivers

- \${KERNEL}/drivers/chars
- \${KERNEL}/drivers/i2c
- \${KERNEL}/drivers/gpio
- Module aliases for module loading utilities.

➤ Kernel build configure

- \${KERNEL}/.config

➤ Kconfig

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

➤ \$make menuconfig



Build Modules

► Build modules

→ \$ make modules

► Add install patch

→ \$ export **INSTALL_MOD_PATH=../modules**

► Install module to **INSTALL_MOD_PATH**

→ \$ 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}
- \$ insmode \${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`



Exercise

- ▶ Deploy Linux modules to target board from nanopi-m4 Linux kernel
- ▶ Use depmod
- ▶ Use modprobe, insmod, rmmod