MYD-Y6ULX 软件评估指南

文件状态： [ ] 草稿 [√] 正式发布	**文件标识：**	MYIR-i.MX6ULX-SW-EG-ZH-L5.4.3
	当前版本：	V2.0.1
	作　　者：	Alex、Roy
	创建日期：	2020-09-10
	最近更新：	2020-10-10

版 本 历 史

版本	作者	参与者	日期	备注
V2.0.0	Alex		20200910	初始版本，适用于 MYD-Y6ULX
V2.0.1	Alex		20210110	修正文字描述错误

目 录

1. 概述

Linux 软件评估指南用于介绍在米尔的开发板上运行开源 Linux 系统下的核心资源与外设资源的测试步骤与评估方法。本文可作为前期评估指南使用，也可以作为通用系统开发的测试指导书使用。

1.1. 硬件资源

本文档适用于米尔电子的 MYD-Y6ULX 系列板卡，它是基于 NXP 公司的高性能嵌入式 ARM 处理器 i.MX6UL 系列开发的一套开发板。关于硬件部分的详细配置参数请查看《MYD-Y6ULX 产品手册》。

1.2. 软件资源

MYD-Y6ULX 系列开发板的 BSP 是基于 NXP 官方开源社区版 Linux BSP 移植与修改而来，系统镜像采用 Yocto 项目进行构建。Bootloader, Kernel 以及文件系统各部分软件资源全部以源码的形式开放，具体内容请查看《MYD-Y6ULX SDK 发布说明》。

开发板在出厂时已经烧录了 myir-image-full 镜像，您只需要上电即可使用。

1.3. 文档资源

根据用户使用开发板的各个不同阶段，SDK 中包含了除发布说明之外的入门指南，评估指南，开发指南，应用笔记，常用问答等不同类别的文档和手册。具体的文档列表参见《MYD-Y6ULX SDK 发布说明》表 2-4 中的说明。

1.4. 环境准备

在开始评估开发板软件之前，您需要对开发板做一些必要的准备和配置一些环境，包括正确硬件接线，配置调试串口，设置启动等步骤。详细的步骤可以参照《MYD-Y6ULX快速入门指南》。

接下来的部分重点介绍如何对系统的硬件资源和接口以及软件功能进行评估和测试。主要借助一些 Linux 下常用的工具和命令，以及自己开发的应用进行测试。软件评估指南分为多个部分来描述，包括：核心资源，外设资源，网络应用，多媒体应用，开发支持应用，系统工具等几大类。后面的章节会针对各个部分做全方位的讲解，并详细描述各部分资源的具体评估方法和步骤。

2. 核心资源

在 Linux 系统中，提供了 proc 虚拟文件系统来查询各项核心资源的参数以及一些通用工具来评估资源的性能。下面将具体对 CPU、 Memory、Flash、RTC、Watchdog、PowerManger 等核心资源的参数进行读取与测试。

2.1. CPU

MYD-Y6ULX 核心芯片是 i.MX6ULx 系列是基于高性能、超低功率 ARM Cortex-A7 核心处理器，处理器运行速度高达 900MHz,包含 128 KB L2 高速缓存和 16 位 DDR3/LPDDR2 支持。其内部集成了电源管理、安全单元和丰富的互联接口，为物联网应用提供了一种安全、低功耗、高性能的解决方案。

1) 查看 CPU 信息命令：

读取系统中的 CPU 的提供商和参数信息，则可以通过/proc/cpuinfo 文件得到。

```
root@myd-y6ull14x14:~# cat /proc/cpuinfo
processor : 0
model name : ARMv7 Processor rev 5 (v7l)
BogoMIPS : 24.00
Features : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt vfpd
32 lpae
CPU implementer : 0x41
CPU architecture: 7
CPU variant : 0x0
CPU part : 0xc07
CPU revision : 5

Hardware : Freescale i.MX6 Ultralite (Device Tree)
Revision : 0000
Serial : 0323b1d75986e1d8
```

- processor ： 系统中逻辑处理核的编号，对于多核处理器则可以是物理核、或者使用超线程技术虚拟的逻辑核。
- model name ： CPU 属于的名字及其编号。
- bogomips ： 在系统内核启动时粗略测算的 CPU 每秒运行百万条指令数（Million Instructions Per Second）

2) 查看 CPU 使用率

```
root@myd-y6ull14x14:~# top
top - 11:02:51 up  2:17,  1 user,  load average: 0.70, 0.65, 0.64
Tasks:  71 total,   2 running,  69 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.0 us,  0.3 sy,  0.0 ni, 98.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :    490.0 total,    318.7 free,    106.1 used,     65.1 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used.    364.5 avail Mem
```

- %us：表示用户空间程序的 cpu 使用率（没有通过 nice 调度）
- %sy：表示系统空间的 cpu 使用率，主要是内核程序。
- %ni：表示用户空间且通过 nice 调度过的程序的 cpu 使用率。
- %id：空闲 cpu

3) 获取 CPU 温度信息

i.MX6UL 芯片内部有 Temperature Monitor 模块。

```
root@myd-y6ull14x14:~# cat /sys/class/thermal/thermal_zone0/temp
52730
```

- 显示数字为千分之一度，需除以 1000 就是当前温度值。

4) CPU 压力测试

CPU 的压力的测试方式有很多，可通过 bc 命令来计算圆周率方法来测试 CPU 在运算过程中的稳定性。

```
root@myd-y6ull14x14:~# echo "scale=5000; 4*a(1)" | bc -l -q &
[1] 507
```

上述命令将在后台计算的 PI，并精确到小数点后 5000 位。 计算过程需要一段时间。 此时，我们可以通过顶部命令检查 CPU 利用率的变化。

```
root@myd-y6ull14x14:~# top
top - 03:47:29 up 33 min,  1 user,  load average: 0.57, 0.16, 0.05
```

```
Tasks: 126 total,   2 running, 124 sleeping,   0 stopped,   0 zombie
%Cpu(s): 25.1 us,  0.0 sy,  0.0 ni, 74.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   1943.6 total,   1623.3 free,    234.6 used,     85.7 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used.   1617.3 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME
+ COMMAND

    784 root      20   0    2728   1644   1392 R  99.7   0.1   0:42.56
```

约 3 分钟后，PI 结果被计算出来。 在此器件 CPU 使用率很高，没有发生异常，说明 CPU 压力测试可以通过。通过增加精确度要求，还可以进一步提高测试压力。

```
root@myd-y6ull14x14:~# 3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706798214808651328230664709384460955058\
223172535940812848111745028410270193852110555964462294895493038196447\
288109756659334461284756482337867831652712019091456485669234603486107\
454326648213393607260249141273724587006606315588174881520920962829255\
409171536436789259036001133053054882046652138414695194151160943305722\
703657595919530921861173819326117931051185480744623799627495673518855\
**********
```

5）CPU 工作频率

CPU 支持动态调频，也可以设置固定频率。

- **查看 CPU 的工作频率**

```
root@myd-y6ull14x14:~# cat /sys/bus/cpu/devices/cpu0/cpufreq/cpuinfo_cur_freq
198000
```

- **查看 CPU 最大工作频率**

```
root@myd-y6ull14x14:~# cat /sys/bus/cpu/devices/cpu0/cpufreq/cpuinfo_max_freq
528000
```

- **查看 CPU 的当前工作模式**

```
root@myd-y6ull14x14:~# cat /sys/bus/cpu/devices/cpu0/cpufreq/scaling_governor
ondemand
```

- **查看 CPU 的所有工作模式**

```
root@myd-y6ull14x14:~# cat /sys/bus/cpu/devices/cpu0/cpufreq/scaling_available_gove
rnors
conservative userspace powersave ondemand performance
```

这里可以通过设置 CPU 的 performance 模式，让 CPU 一只处于最高频率。

```
root@myd-y6ull14x14:~# echo "performance" > /sys/bus/cpu/devices/cpu0/cpufreq/sc
aling_governor
root@myd-y6ull14x14:~# cat /sys/bus/cpu/devices/cpu0/cpufreq/cpuinfo_cur_freq
528000
```

2.2. Memory

1） 查看内存信息

读取系统中的内存的参数信息，则可以通过/proc/meminfo 文件得到。

```
root@myd-y6ull14x14:~# cat /proc/meminfo
MemTotal:         501788 kB
MemFree:          326380 kB
MemAvailable:     373268 kB
Buffers:            4208 kB
Cached:            56212 kB
SwapCached:            0 kB
Active:            35820 kB
Inactive:          47996 kB
Active(anon):      23860 kB
Inactive(anon):     9268 kB
略
```

- MemTotal ： 所有可用的 RAM 大小，物理内存减去预留位和内核使用
- MemFree ： LowFree + HighFree
- Buffers ： 用来给块设备做缓存的大小
- Cached ： 文件的缓冲区大小
- SwapCached ： 已经被交换出来的内存。与 I/O 相关
- Active ： 经常（最近）被使用的内存
- Inactive ： 最近不常使用的内存

2） 获取内存使用率

可使用 free 命令来读取内存的使用情况，-m 参数代表单位为 MByte。

```
root@myd-y6ull14x14:~# free -m
              total        used        free      shared  buff/cache   available
Mem:            490         106         318           9          65
 364
Swap:             0           0           0
```

- total ： 内存总量。

- used ： 被使用的内存量。
- free ： 可使用的内存量。

3) 内存压力测试

通过给定测试内存的大小和次数，可以对系统现有的内存进行项目上的测试。也方便做压力测试，如指定内存测试 1 次等。内核自带的 Memtester 工具，测试方法如下。

```
root@myd-y6ull14x14:~# memtester 300M 1
memtester version 4.3.0 (64-bit)
Copyright (C) 2001-2012 Charles Cazabon.
Licensed under the GNU General Public License version 2 (only).

pagesize is 4096
pagesizemask is 0xfffffffffffff000
want 300MB (314572800 bytes)
got  300MB (314572800 bytes), trying mlock ...locked.
Loop 1/1:
  Stuck Address       : ok
  Random Value        : ok
  Compare XOR         : ok
  Compare SUB         : ok
  Compare MUL         : ok
  Compare DIV         : ok
  Compare OR          : ok
  Compare AND         : ok
  Sequential Increment: ok
  Solid Bits          : ok
  Block Sequential    : ok
  Checkerboard        : ok
  Bit Spread          : ok
  Bit Flip            : ok
  Walking Ones        : ok
  Walking Zeroes      : ok

Done.
```

2.3. Flash

MYD-Y6ULX 的 flash 有 eMMC 和 Nand 版本。

eMMC 是一个数据存储设备，包括一个 MultiMediaCard (MMC)接口，一个 NAND Flash 组件。它的成本、体积小、Flash 技术独立性和高数据吞吐量使其成为嵌入式产品的理想选择。MYD-Y6ULX 的 EMMC 版本为 4G 容量。

本节将讲解在 Linux 系统下查看与操作 eMMC 的步骤与方法。

1) emmc

通过 fdisk -l 命令可以查询到 emmc 分区信息及容量：

```
root@myd-y6ull14x14:~# fdisk -l
Disk /dev/mmcblk1: 3672 MB, 3850371072 bytes, 7520256 sectors
117504 cylinders, 4 heads, 16 sectors/track
Units: sectors of 1 * 512 = 512 bytes

Device          Boot StartCHS    EndCHS        StartLBA     EndLBA    Sectors
  Size Id Type
/dev/mmcblk1p1      320,0,1     895,3,16         20480     122879     10240
0 50.0M  c Win95 FAT32 (LBA)
/dev/mmcblk1p2      896,0,1     767,3,16        122880    7520255    739737
6 3612M 83 Linux
```

➢/dev/mmcblk2p1 ： 用来存放 kernel 和 dtb 文件

➢/dev/mmcblk2p2 ： 用于存放文件系统

2) Nand

Nand-flash 内存是 flash 内存的一种，其内部采用非线性宏单元模式，为固态大容量内存的实现提供了廉价有效的解决方案。

●查看代码分区定义

nand 的分区结构在 uboot 中定义，uboot 启动后将参数传递给 kernel。定义如下：

```
#define MFG_NAND_PARTITION "mtdparts=gpmi-nand:5m(boot),1m(env),10m(k
ernel),1m(dtb),-(rootfs) "
#else
```

●查看系统启动后的分区情况

进入 linux 系统后可以查看实际的分区情况：

```
root@myd-y6ull14x14:~# cat /proc/mtd
```

```
dev:    size    erasesize    name
mtd0: 00500000 00020000 "boot"
mtd1: 00100000 00020000 "env"
mtd2: 00a00000 00020000 "kernel"
mtd3: 00100000 00020000 "dtb"
mtd4: 0ef00000 00020000 "rootfs"
```

- 查看传递分区参数

```
root@myd-y6ull14x14:~#cat /proc/cmdline
console=ttymxc0,115200 ubi.mtd=4 root=ubi0:rootfs rootfstype=ubifs mtdparts
=gpmi-nand:5m(boot),1m(env),10m(kernel),1m(dtb),-(rootfs)
```

- ubi.mtd ：指定文件系统是第几个分区
- rootfstype ：指定文件系统的格式
- root ：指定分区挂载
- Console ：指定调试串口参数

2.4. RTC

RTC（Real-time clock）本身是一个时钟，用来记录真实时间，当软件系统关机后保留系统时间并继续进行计时，系统重新开启后在将时间同步进软件系统。

i.MX6UL 芯片内部包含 RTC 时钟，建议采用外部 RTC 芯片。RTC 的测试通常采用 Linux 系统常用的 hwclock 和 date 命令配合进行，下面测试将系统时间写入 RTC，读取 RTC 时间并设置为系统时间并进行时间掉电保持的测试。

1） 查看系统 RTC 设备

```
root@myd-y6ull14x14:~# ls -l /dev/rtc*
lrwxrwxrwx 1 root root        4 Sep  5 03:14 /dev/rtc -> rtc0
crw------- 1 root root 251, 0 Sep  5 03:14 /dev/rtc0
crw------- 1 root root 251, 1 Sep  5 03:14 /dev/rtc1
```

rtc 属于 linux 设备，在/dev 下有其设备节点 rtc0，rtc1 可供用户操作。

```
root@myd-y6ull14x14:~# cat /sys/class/rtc/rtc0/name
snvs_rtc 30370000.snvs:snvs-rtc-lp
root@myd-y6ull14x14:~# cat /sys/class/rtc/rtc1/name
rtc-ds1307 1-0032
```

2） 设置系统时间

在将系统时间设置为 Tue Sep 8 09:07:30 UTC 2020

```
root@myd-y6ull14x14:~# date  090809072020.30
Tue Sep  8 09:07:30 UTC 2020
```

3） 将系统时间写入 RTC

将上一步 date 命令设置的系统时间写入到 RTC 设备

```
root@myd-y6ull14x14:~# hwclock -w
```

4） 读取 RTC 时间并设置为系统时间

```
root@myd-y6ull14x14:~# hwclock -r
2020-09-08 09:08:02.588845+00:00
```

5） 掉电保持 RTC 时间，

电池采用 CR1225 尺寸电池，将设备断开电源，经过 2 分钟左右，上电。查看 RTC 时间和系统时间。

```
root@myd-y6ull14x14:~# hwclock -r
2020-09-08 09:10:50.206047+00:00
```

6) 将系统时间与 RTC 时间同步

```
root@myd-y6ull14x14:~# hwclock -s
root@myd-y6ull14x14:~# date
Tue Sep  8 09:11:07 UTC 2020
```

重新开机之后查看的 RTC 时间和系统时间比之前设置的时候增加了大约 2 分钟，说明 RTC 工作正常。如果需要详细测试 RTC 的精度，可以将断电时间延长如 24 小时，测试 RTC 时间与标准时间的差异。

2.5. Watchdog

Linux 内核包含 Watchdog 子系统，硬件设计过程中一般可以利用芯片内部的看门狗定时器或者使用外部看门狗芯片来实现 Watchdog 的功能，用于监测系统的运行。当系统出现异常情况无法喂狗时系统将可以进行自动复位

MYD-Y6ULX 芯片内部有 1 个看门狗，本节演示 watchdog 的使用，测试看门狗的开启、关闭、设置看门狗超时时间

1） 关闭看门狗

```
root@myd-y6ull14x14:~# echo V > /dev/watchdog0
```

2） 使用应用程序测试看门狗

```
root@myd-y6ull14x14:~# watchdog_test
Usage: wdt_driver_test <timeout> <sleep> <test>
    timeout: value in seconds to cause wdt timeout/reset
    sleep: value in seconds to service the wdt
    test: 0 - Service wdt with ioctl(), 1 - with write()
```

运行看门狗应用,超时时间为 4s，每间隔 1s 喂一次狗：

```
root@myd-y6ull14x14:~# watchdog_test  4 1 0
Starting wdt_driver (timeout: 4, sleep: 1, test: ioctl)
Trying to set timeout value=4 seconds
The actual timeout was set to 4 seconds
Now reading back -- The timeout is 4 seconds
```

如果将上面的参数 1 改为 5，则是设置间隔 5s 去喂狗，此时超过了要求的 4s 的看门狗超时时间，开发板会重启。

2.6. Power Management

本章节演示 Linux 电源管理的 Suspend 功能，让开发板睡眠 ，通过外部事件唤醒。Linux 内核一般提供了三种 Suspend: Freeze、Standby 和 STR(Suspend to RAM)，在用户空间向”/sys/power/state”文件分别写入”freeze”、”standby”和”mem”，即可触发它们。

1) 查看当前开发板支持的模式

```
root@myd-y6ull14x14:~# cat /sys/power/state
freeze standby mem
```

2) 在用户空间写入的方法

```
root@myd-y6ull14x14:~# echo "freeze" > /sys/power/state
root@myd-y6ull14x14:~# echo "standby" > /sys/power/state
root@myd-y6ull14x14:~# echo "mem" > /sys/power/state
```

- 以 mem 为例测试，输入休眠命令后开发板休眠，此时调试串口无法再输入，心跳灯停止闪烁。

```
root@myd-y6ull14x14:~# echo "mem" > /sys/power/state
[  278.716505] PM: suspend entry (deep)
[  278.720724] Filesystems sync: 0.000 seconds
[  278.741713] Freezing user space processes ... (elapsed 0.001 seconds) done.
[  278.750794] OOM killer disabled.
[  278.754120] Freezing remaining freezable tasks ... (elapsed 0.001 seconds) done.
[  278.762886] printk: Suspending console(s) (use no_console_suspend to debug)
```

- 通过按键（KEY USER）唤醒：

```
[  278.872728] fec 20b4000.ethernet eth0: Link is Down
[  278.874269] PM: suspend devices took 0.100 seconds
[  278.878646] Disabling non-boot CPUs ...
[  278.878658] Turn off M/F mix!
[  278.882566] imx-sdma 20ec000.sdma: loaded firmware 3.5
[  279.205157] usb 1-1: reset high-speed USB device number 2 using ci_hdrc
[  279.526926] PM: resume devices took 0.650 seconds
[  279.560349] OOM killer enabled.
```

```
[  279.563503] Restarting tasks ... done.
[  279.630415] PM: suspend exit
root@myd-y6ull14x14:~#
```

此时调试串口可以重新输入，心跳灯开始闪烁。

3. 外围接口

3.1. GPIO

MYD-Y6ULX 的 GPIO 脚是以 GPIOX_Y 形式来定义的，pin 脚标签名和 GPIOX_Y 形式之间的映射关系可以参考《MYC-Y6ULX Pin list_V13.xlsx》手册。GPIOX_Y 转换成引脚编号公式为:

```
(X-1) *32+Y
```

下面介绍 Linux 下操作 GPIO 方法。

- 文件系统 gpio 操作

在文件系统操作 GPIO 时，必须先检查想要操作的 GPIO 是否被其他驱动占用，如果被暂用，后续操作会报 busy 的警告。下面以操作拓展脚（J14-PIN5)的一个接口为例，使用万用表测量电压。

1）导出 GPIO

```
root@myd-y6ull14x14:~#/sys# echo 24 > /sys/class/gpio/export
```

导出成功后会在/sys/class/gpio/目录下生成 gpio24 这个目录。

2）设置/查看 GPIO 方向

- 设置输入:

```
root@myd-y6ull14x14:~# echo "in" > /sys/class/gpio/gpio24/direction
```

- 设置输出:

```
root@myd-y6ull14x14:~# echo "out" > /sys/class/gpio/gpio24/direction
```

- 查看 gpio 方向:

```
root@myd-y6ull14x14:~# cat /sys/class/gpio/gpio24/direction
out
```

返回 in 表示出入，返回 out 表示输出。

3）设置/查看 GPIO 的值

- 设置输出低:

```
root@myd-y6ull14x14:~# echo "0" > /sys/class/gpio/gpio24/value
```

- 设置输出高:

```
root@myd-y6ull14x14:~# echo "1" > /sys/class/gpio/gpio24/value
```

3.2. LED

Linux 系统提供了一个独立的子系统以方便从用户空间操作 LED 设备，该子系统以文件的形式为 LED 设备提供操作接口。这些接口位于/sys/class/leds 目录下。MYD-Y6ULX 只有一个 LED（蓝色指示灯）。下面通过命令读写 sysfs 的方式对 LED 进行测试。下述命令均为通用命令，也是操控 LED 的通用方法。

1）操作 LED 的目录为/sys/class/gpios,该目录内容为：

```
root@myd-y6ull14x14:~# ls  /sys/class/leds/
cpu  mmc0::  mmc1::
```

2）以 cpu 为例测试 LED

- **读取 cpu LED 状态，其中 0 表示 LED 关闭，1~255 表示 LED 开启**

```
root@myd-y6ull14x14:~# cat  /sys/class/leds/cpu/brightness
255
```

可知当前红灯 LED 为开启状态。

- **熄灭 LED**

```
root@myd-y6ull14x14:~# echo 0 > /sys/class/leds/cpu/brightness
```

- **点亮 LED**

```
root@myd-y6ull14x14:~# echo 1 > /sys/class/leds/cpu/brightness
```

- **开启 LED 心跳**

```
root@myd-y6ull14x14:~# echo "heartbeat" > /sys/class/leds/cpu/trigger
```

3.3. RS232

本节将使用 Linux API 配置开发板 RS232 的收发功能。Linux 的串口设备文件一般命名为/dev/ttymxcn(n=0,1,2,3.....)。n 表示串口在 Linux 系统中的设备编号，“ttymxc”是内核已经定义好的串口设备名字。本节是以 MYD-Y6ULX 底板上 J10 接口（即 uart2）为例进行测试，uart2 的编号设备节点为 ttymxc1。

表 3-1. ttymxc 与 uart 的对应关系

ttymxc	uart
/dev/ttymxc0	Uart1
/dev/ttymxc1	Uart2
/dev/ttymxc2	Uart3
/dev/ttymxc3	Uart4
/dev/ttymxc4	Uart5

将 J10 的 TXD 与 RXD 分别与 USB-RS232 转换器的 RXD 和 TXD 连接。

- **PC 端收发数据**

  在 windows 下使用调试串口工具发数据 1234567890，查看接收区

- **开发板测试收发数据：**

```
root@myd-y6ull14x14:~# uart_test -d /dev/ttymxc1 -b 115200
/dev/ttymxc1 RECV 10 total
/dev/ttymxc1 RECV: 1234567890
/dev/ttymxc1 RECV 10 total
/dev/ttymxc1 RECV: 1234567890
/dev/ttymxc1 RECV 10 total
/dev/ttymxc1 RECV: 1234567890
/dev/ttymxc1 RECV 10 total
/dev/ttymxc1 RECV: 1234567890
```

3.4. RS485

本例程演示如何使用 Linux API 测试开发板 RS485 发送和接收数据功能，设备节点为 ttymxc3。以 J10 接口的 RS485 为例进行测试，准备两块 MYD-Y6ULX 开发板，将一块板子的 J10 的 A 与 B 分别与另一个板子 A 和 B 连接。

- **一块开发板做发送端，运行程序发送数据：**

```
# rs485_write -d /dev/ttymxc3 -b 4800 -e 1
SEND[20]: 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0a 0x0b 0x0c 0x0d 0x0e 0x0f 0x10 0x11 0x12 0x13 0x14
SEND[20]: 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0a 0x0b 0x0c 0x0d 0x0e 0x0f 0x10 0x11 0x12 0x13 0x14
SEND[20]: 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0a 0x0b 0x0c 0x0d 0x0e 0x0f 0x10 0x11 0x12 0x13 0x14
SEND[20]: 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0a 0x0b 0x0c 0x0d 0x0e 0x0f 0x10 0x11 0x12 0x13 0x14
```

- **另一个板子做接收端，运行程序接收数据：**

```
# rs485_read -d /dev/ttymxc3 -b 4800 -e 1
RECV[20]: 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0a 0x0b 0x0c 0x0d 0x0e 0x0f 0x10 0x11 0x12 0x13 0x14
RECV[20]: 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0a 0x0b 0x0c 0x0d 0x0e 0x0f 0x10 0x11 0x12 0x13 0x14
RECV[20]: 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0a 0x0b 0x0c 0x0d 0x0e 0x0f 0x10 0x11 0x12 0x13 0x14
RECV[20]: 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0a 0x0b 0x0c 0x0d 0x0e 0x0f 0x10 0x11 0x12 0x13 0x14
```

3.5. CAN BUS

本节采用 Linux 系统常用的 cansend、candump 命令进行 SocketCAN 的通讯测试。这里测试使用的两块开发板对接测试。

J10 座子的 CANH、CANL 引脚和同类型的板子 CANH、CANL 相连。

1) 初始化 CAN 网络接口

设置 can 波特率：

使用 can 需要先设置波特率，并开启 CAN 网络接口。参考下面命令分别将两块板子的仲裁波特率设置为 1M，数据波特率设备为 4M，并开启 can 功能。

```
root@myd-y6ull14x14:~# ip link set can0 type can bitrate 50000 triple-sampling on
root@myd-y6ull14x14:~# ifconfig can0 up
```

2) 收发数据测试

一块板子做发送端：

```
root@myd-y6ull14x14:~# cansend can0 100#01.02.03.04.05.06.07.08
```

一块板子做接收端：

```
root@myd-y6ull14x14:~#  candump can0
root@myd-y6ull14x14:~#  can0 100 [8] 01 02 03 04 05 06 07 08
```

3) 查看数据收发的统计信息

CAN 数据收发之后显示 CAN 设备的详情和收发统计信息，其中“clock”的值代表 can 的时钟，” drop”的值代表丢包，” overrun”的值代表溢出，” error”代表总线错误。

```
root@myd-y6ull14x14:~#  ip -details -statistics link show can0
2: can0: <NOARP,ECHO> mtu 16 qdisc noop state DOWN mode DEFAULT group default qlen 10
    link/can  promiscuity 0 minmtu 0 maxmtu 0
    can state STOPPED (berr-counter tx 0 rx 0) restart-ms 0
```

```
          flexcan: tseg1 4..16 tseg2 2..8 sjw 1..4 brp 1..256 brp-inc 1
          clock 30000000
          re-started bus-errors arbit-lost error-warn error-pass bus-off
          0          0          0          0          0          0          num
txqueues 1 numrxqueues 1 gso_max_size 65536 gso_max_segs 65535
    RX: bytes  packets  errors  dropped overrun mcast
    0          0        0       0       0       0
    TX: bytes  packets  errors  dropped carrier collsns
    0          0        0       0       0       0
```

3.6. KEY(按键)

Linux 的/etc/input/eventxx 设备可以用来方便地调试 鼠标、键盘、触摸板等输入设备。本节主要是测试 key。通过 evtest 命令来查看按键是否有反应。MYD-Y6ULX 有三个按键，K1 是 onoff 按键，K2 系统复位按键，K3 是用户按键；

1）设备树配置信息

打开配套的设备树文件 myb-imx6ul-14x14.dtsi,可以看到对应的 gpio-keys 节点

```
gpio-keys {
        compatible = "gpio-keys";
        pinctrl-names = "default";
        pinctrl-0 = <&pinctrl_gpio_key>;
        user {
                label = "User Button";
                gpios = <&gpio5 0 GPIO_ACTIVE_HIGH>;
                gpio-key,wakeup;
                linux,code = <KEY_1>;
        };
};
```

2）按键测试

- 查看对应的输入设备 event 信息

```
root@myd-y6ull14x14:~# evtest
No device specified, trying to scan all of /dev/input/event*
Available devices:
/dev/input/event0:    20cc000.snvs:snvs-powerkey
/dev/input/event1:    generic ft5x06 (00)
/dev/input/event2:    iMX6UL Touchscreen Controller
/dev/input/event3:    gpio-keys
```

由上可以知 gpio-keys 的对应设备事件为 event2。

- 执行下面命令，操作按键 K3，串口终端会打印出如下信息：

```
Select the device event number [0-3]: 3
Input driver version is 1.0.1
Input device ID: bus 0x19 vendor 0x1 product 0x1 version 0x100
Input device name: "gpio-keys"
```

```
Supported events:
  Event type 0 (EV_SYN)
  Event type 1 (EV_KEY)
    Event code 2 (KEY_1)
Properties:
Testing ... (interrupt to exit)
Event: time 1599641292.878878, type 1 (EV_KEY), code 2 (KEY_1), value 0
Event: time 1599641292.878878, ---------- SYN_REPORT ------------
Event: time 1599641293.049011, type 1 (EV_KEY), code 2 (KEY_1), value 1
Event: time 1599641293.049011, ---------- SYN_REPORT ------------
Event: time 1599641293.718856, type 1 (EV_KEY), code 2 (KEY_1), value 0
Event: time 1599641293.718856, ---------- SYN_REPORT ------------
Event: time 1599641293.858903, type 1 (EV_KEY), code 2 (KEY_1), value 1
Event: time 1599641293.858903, ---------- SYN_REPORT ------------
Event: time 1599641294.198953, type 1 (EV_KEY), code 2 (KEY_1), value 0
Event: time 1599641294.198953, ---------- SYN_REPORT ------------
Event: time 1599641294.308852, type 1 (EV_KEY), code 2 (KEY_1), value 1
Event: time 1599641294.308852, ---------- SYN_REPORT ------------
Event: time 1599641294.558876, type 1 (EV_KEY), code 2 (KEY_1), value 0
Event: time 1599641294.558876, ---------- SYN_REPORT ------------
Event: time 1599641294.678870, type 1 (EV_KEY), code 2 (KEY_1), value 1
Event: time 1599641294.678870, ---------- SYN_REPORT ------------
Event: time 1599641294.898855, type 1 (EV_KEY), code 2 (KEY_1), value 0
Event: time 1599641294.898855, ---------- SYN_REPORT ------------
Event: time 1599641295.038987, type 1 (EV_KEY), code 2 (KEY_1), value 1
Event: time 1599641295.038987, ---------- SYN_REPORT ------------
```

每按一次 K3 当前终端会打印出当前事件码值，即按键正常。

3.7. USB

MYD-Y6ULX 拥有 2 路 USB2.0 接口，一路用于 OTG 下载镜像，另一路 USB2.0 口经过拓展芯片变成 2 路 USB 口。

本节通过相关命令或热插拔、USB HUB 验证 USB Host 驱动的可行性，实现读写 U 盘的功能、usb 枚举功能。

1）查看插入 usb 的打印信息

- 将 U 盘连接到开发板 USB Host 接口，内核提示信息如下：

```
root@myd-y6ull14x14:~# [  755.254972] usb 1-1.1: new high-speed USB device number 3 using ci_hdrc
[  755.413421] usb 1-1.1: New USB device found, idVendor=058f, idProduct=6387, bcdDevice= 1.06
[  755.422370] usb 1-1.1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[  755.437531] usb 1-1.1: Product: Mass Storage
[  755.441955] usb 1-1.1: Manufacturer: Generic
[  755.451900] usb 1-1.1: SerialNumber: 03F004F7
[  755.468433] usb-storage 1-1.1:1.0: USB Mass Storage device detected
[  755.493065] scsi host0: usb-storage 1-1.1:1.0
[  756.567834] scsi 0:0:0:0: Direct-Access     Generic  Flash Disk       8.07 PQ: 0 ANSI: 4
[  756.594293] sd 0:0:0:0: [sda] 31129600 512-byte logical blocks: (15.9 GB/14.8 GiB)
[  756.616305] sd 0:0:0:0: [sda] Write Protect is off
[  756.634066] sd 0:0:0:0: [sda] Write cache: disabled, read cache: enabled, doesn't support DPO or FUA
[  756.694383]  sda:
[  756.711553] sd 0:0:0:0: [sda] Attached SCSI removable disk
[  757.721016] FAT-fs (sda): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
```

从上述信息可以得出需要挂载的设备为 sda。

2）U 盘挂载读写

U 盘挂载读写，由于文件系统已经加入了自动挂载功能，所以只要是 U 盘已经格式化 vfat，ext4，格式，就可以挂载。

- 查看挂载点

```
root@myd-y6ull14x14:~# cat /proc/mounts | grep sda*
cgroup2 /sys/fs/cgroup/unified cgroup2 rw,nosuid,nodev,noexec,relatime,nsdelegate 0 0
/dev/sda /run/media/sda vfat rw,relatime,gid=6,fmask=0007,dmask=0007,allow_utime=0020,codepage=437,iocharset=iso8859-1,shortname=mixed,errors=remount-ro 0 0
```

- 写文件

```
root@myd-y6ull14x14:~# echo "myir udisk test" > /run/media/sda/test_file.txt
```

- 读文件

```
root@myd-y6ull14x14:~# cat /run/media/sda/test_file.txt
myir udisk test
```

写完文件后需要执行下 sync 命令，确保数据完全写入到 U 盘里面之后，才可以卸载设备。

3）关于断电文件保存情况

磁盘文件写入涉及缓存概念，分成全缓存，行缓存和无缓存，默认 io 里面是全缓存，只有当缓存数据满后才会写到磁盘文件。所以在写入文件中突然断电，会让文件写入内容丢失。但是当采用 reboot 时，系统会帮忙把缓冲区数据写入数据，命令行可用 sync 强行将数据写入磁盘。

3.8. Backlight

本例程主要是测试背光的亮度，通过操作 LCD 在/sys 目录下的对应文件，以实现查询、调节背光亮度。

1）查看当前背光级别

- **进入背光参数调节目录**

```
root@myd-y6ull14x14:~# cd /sys/class/backlight/backlight-display/
root@myd-y6ull14x14:/sys/class/backlight/backlight-display# ls
actual_brightness brightness device power subsystem type
bl_power consumers max_brightness scale suppliers uevent
```

- **查看当前背光级别**

```
root@myd-y6ull14x14:/sys/class/backlight/backlight-display# cat brightness
6
```

- **查看背光最大级别**

```
root@myd-y6ull14x14:/sys/class/backlight/backlight-display# cat max_brightnes
s
7
```

2）调节背光级别

- **如果需要调节背光级别，用 echo 向 brightness 写入对应亮度级别，但不能超过最大级别**

```
root@myd-y6ull14x14:/sys/class/backlight/backlight-display# echo 10 > brightn
ess
sh: write error: Invalid argument
```

3.9. Touch Panel

MYD-Y6ULX 系列开发板支持电容触摸和电阻触摸，MYiR 电容屏触摸 IC 为 FT5x16，电阻屏使用 i.MX6UL 芯片内部 TSC 模块读取 ADC 数据。用户可根据实际需求自行购买配件。电容屏在使用中较为灵敏，很少出现问题。触摸屏能点击，就表示它没问题。另外，电容屏不需要较准。因为根据电容屏的原理，电容屏在使用中是可以准确的识别出手指与屏幕接触的位置，具有很高的灵敏性。下面通过 evtest 命令屏的触摸功能做简单测试。

1） 校准

```
root@myd-y6ull14x14:~# ts_calibrate
xres = 800, yres = 480
Took 7 samples...
Top left : X = 3174 Y = 509
Took 5 samples...
Top right : X = 3120 Y = 3715
Took 4 samples...
Bot right : X = 843 Y = 3686
Took 7 samples...
Bot left : X = 855 Y = 483
Took 1 samples...
Center : X = 1992 Y = 2104
-53.327515 -0.002611 0.218416
573.700073 -0.165326 -0.001703
Calibration constants: -3494872 -171 14314 37598008 -10834 -111 65536
root@myd-y6ull14x14:~#
```

运行校准测试程序，在 LCD 上一次点击

2） 测试过程

首先连接 LCD，电容屏也可使用 evtest 打印触摸的功能，点击一下 LCD 屏会看到下面终端打印信息。

```
root@myd-y6ull14x14:~# evtest
No device specified, trying to scan all of /dev/input/event*
Available devices:
```

```
/dev/input/event0:        30370000.snvs:snvs-powerkey
/dev/input/event1:        generic ft5x06 (00)
/dev/input/event2:        gpio-keys
/dev/input/event3:        bd718xx-pwrkey
Select the device event number [0-3]: 1
Input driver version is 1.0.1
Input device ID: bus 0x18 vendor 0x0 product 0x0 version 0x0
Input device name: "generic ft5x06 (00)"
Supported events:
  Event type 0 (EV_SYN)
  Event type 1 (EV_KEY)
    Event code 330 (BTN_TOUCH)
  Event type 3 (EV_ABS)
    Event code 0 (ABS_X)
    --snip--
    Testing ... (interrupt to exit)
    Event: time 1599614278.142778, type 3 (EV_ABS), code 57 (ABS_MT_TRAC
KING_ID), value 7
    Event: time 1599614278.142778, type 3 (EV_ABS), code 53 (ABS_MT_POSITI
ON_X), value 419
    Event: time 1599614278.142778, type 3 (EV_ABS), code 54 (ABS_MT_POSITI
ON_Y), value 367
    Event: time 1599614278.142778, type 1 (EV_KEY), code 330 (BTN_TOUCH),
 value 1
    Event: time 1599614278.142778, type 3 (EV_ABS), code 0 (ABS_X), value 4
19
    Event: time 1599614278.142778, type 3 (EV_ABS), code 1 (ABS_Y), value 3
67
    Event: time 1599614278.142778, ---------- SYN_REPORT ------------
    Event: time 1599614278.195100, type 3 (EV_ABS), code 53 (ABS_MT_POSITI
ON_X), value 418
    Event: time 1599614278.195100, type 3 (EV_ABS), code 54 (ABS_MT_POSITI
ON_Y), value 365
```

```
    Event: time 1599614278.195100, type 3 (EV_ABS), code 0 (ABS_X), value 4
18
    Event: time 1599614278.195100, type 3 (EV_ABS), code 1 (ABS_Y), value 3
65
    Event: time 1599614278.195100, ---------- SYN_REPORT ------------
    Event: time 1599614278.272333, type 3 (EV_ABS), code 57 (ABS_MT_TRAC
KING_ID), value -1
    Event: time 1599614278.272333, type 1 (EV_KEY), code 330 (BTN_TOUCH),
 value 0
    Event: time 1599614278.272333, ---------- SYN_REPORT ------------
```

3.10. Display

本例程演示对 Linux 的 FrameBuffer 设备操作，实现液晶输出显示 RGB 颜色和颜色合成测试。例程基于 Linux FrameBuffer API 接口开发。测试前需要把 LCD 连接至 J3 接口上。

米尔科技提供三种 LCD 模块，Linux 软件默认配置为 MY-TFT070CV2。

表 3-2 屏配件选项

型号	说明
MY-TFT070CV2	7 寸电容屏
MY-TFT070RV2	7 寸电阻屏
MY-TFT043RV2	4.3 寸电阻屏

1） LCD 显示测试

- 执行程序后，LCD 液晶屏会出现相应颜色

```
# framebuffer_test
The framebuffer device was opened successfully.
vinfo.xres=480
vinfo.yres=272
vinfo.bits_per_bits=16
vinfo.xoffset=0
vinfo.yoffset=0
red.offset=11
green.offset=5
blue.offset=0
transp.offset=0
finfo.line_length=960
finfo.type = PACKED_PIXELS
The framebuffer device was mapped to memory successfully.
color: red rgb_val: 0000F800
color: green rgb_val: 000007E0
color: blue rgb_val: 0000001F
color: r & g rgb_val: 0000FFE0
```

```
color: g & b rgb_val: 000007FF
color: r & b rgb_val: 0000F81F
color: white rgb_val: 0000FFFF
color: black rgb_val: 00000000
```

2) 修改支持 4.3 寸 LCD

开发板模式是使用的 MYiR 的 7 寸屏，此处说明如何修改为使用 4.3 寸屏。

DTS 文件: arch/arm/boot/dts/myb-imx6ul-14x14.dtsi

```
        display0: display@0 {
                bits-per-pixel = <16>;
                bus-width = <16>;
                display-timings {
                        native-mode = <&timing1>;
                        /*4.3-inch*/
                        timing0: timing0 {
                                clock-frequency = <9200000>;
                                hactive = <480>;
                                vactive = <272>;
                                hfront-porch = <8>;
                                hback-porch = <4>;
                                hsync-len = <41>;
                                vback-porch = <2>;
                                vfront-porch = <4>;
                                vsync-len = <10>;
                                hsync-active = <0>;
                                vsync-active = <0>;
                                de-active = <1>;
                                pixelclk-active = <0>;
                        };
                        /*7.0-inch*/
                        timing1: timing1 {
                                clock-frequency = <33000000>;
                                hactive = <800>;
                                vactive = <480>;
```

```
                                hfront-porch = <210>;
                                hback-porch = <46>;
                                hsync-len = <1>;
                                vback-porch = <22>;
                                vfront-porch = <23>;
                                vsync-len = <20>;
                                hsync-active = <0>;
                                vsync-active = <0>;
                                de-active = <1>;
                                pixelclk-active = <0>;
                        };
                };
```

如上面的代码 native-mode = <&timing1>；为默认的参数，选着的是 7 寸的提名参数，将其修改为 native-mode = <&timing0>；即选择使用 4.3 寸的 timing 参数，然后重新编译 dtb 并下载到板子。

4. 网络接口

MYD-Y6ULX 开发板包含两个百兆以太网接口、一个 Wi-Fi 模组（WM-N-BM-02）一个 4G 模块接口。下面分别对这三种网络设备的配置进行介绍。

4.1. Ethernet

Linux 下网络配置的工具很多，常见的有 net-tools, iproute2，systemd-networkd, network manager 以及 connman 等，这些都可以在系统定制的时候根据实际需要进行选择，这里介绍几种常用的以太网手动临时配置和自动永久配置方式。

开发板出厂的镜像配置了静态 IP，配置文件如下：

```
root@myd-y6ull14x14:~# ls -l /etc/systemd/network/
total 8
-rwxr-xr-x 1 root root 88 Sep 9 09:05 10-static-eth0.network
-rwxr-xr-x 1 root root 88 Sep 9 09:05 11-static-eth1.network
lrwxrwxrwx 1 root root 9 Sep 9 09:11 99-default.link -> /dev/null
```

1） 手动临时配置以太网 IP 地址

- **使用 net-tools 工具包中的 ifconfig 对网络进行手动配置**

首先通过通过 ifconfig 命令查看网络设备信息如下：

```
root@myd-y6ull14x14:~# ifconfig -a
eth0 Link encap:Ethernet HWaddr fa:2e:32:6f:4b:55
inet addr:192.168.30.202 Bcast:192.168.30.255 Mask:255.255.255.0
inet6 addr: fe80::f82e:32ff:fe6f:4b55/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:12795 errors:0 dropped:0 overruns:0 frame:0
TX packets:3896 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:1217512 (1.1 MiB) TX bytes:177644 (173.4 KiB)

eth1 Link encap:Ethernet HWaddr da:0e:6a:e6:8a:a6
inet addr:192.168.40.202 Bcast:192.168.40.255 Mask:255.255.255.0
```

```
          inet6 addr: fe80::d80e:6aff:fee6:8aa6/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:46 errors:0 dropped:0 overruns:0 frame:0
          TX packets:49 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3637 (3.5 KiB)  TX bytes:8216 (8.0 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:2596 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2596 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:232480 (227.0 KiB)  TX bytes:232480 (227.0 KiB)
```

eth0 为实际的以太网设备，这里的 MAC 地址，代码中是按以下顺序读取，如果第一条没有设置就查找下面一条。

- uboot 传参 fec.macddr
- 设备树文件写入 MAC
- flash 或者 fuse 写入 mac
- uboot 在 FEC mac 寄存器写入 mac
- 随机 mac 地址

下面介绍给 eth0 手动配置 IP 地址 192.168.1.100 的方法，命令如下：

```
root@myd-y6ull14x14:~# ifconfig eth0 192.168.1.100 netmask 255.255.255.0 up
```

上面的命令手动配置 eth0 的 IP 地址为 192.168.1.100，子网掩码为 255.255.255.0，以及默认配置的广播地址 192.168.1.255，并通过 up 参数进行激活，如下所示：

```
root@myd-y6ull14x14:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 56:34:ca:32:6a:2d
          inet addr:192.168.1.100  Bcast:192.168.1.255  Mask:255.255.255.0
```

```
        inet6 addr: fe80::5434:caff:fe32:6a2d/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:17911 errors:0 dropped:0 overruns:0 frame:0
        TX packets:344 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:1460204 (1.3 MiB)  TX bytes:39787 (38.8 KiB)
```

- **使用 iprout2 工具包中的 ip 命令对网络进行手动配置**

ifconfig 命令手动设置 IP 地址的方法也可以使用 ip addr 和 ip link 进行替代，更多的信息请查看 https://wiki.linuxfoundation.org/networking/iproute2 中的说明。

```
root@myd-y6ull14x14:~# ip addr flush dev eth0
root@myd-y6ull14x14:~# ip addr add 192.168.1.101/24 brd + dev eth0
root@myd-y6ull14x14:~# ip link set eth0 up
```

如果之前已经配置过 IP 地址，再使用 ip addr add 配置的 IP 地址将会成为 Secondary 地址，所以这里先使用 ip addr flush 清除之前的地址之后再进行配置然后激活。完成配置之后，通过 ip addr show 命令查看 eth0 信息如下：

```
root@myd-y6ull14x14:~# ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state U
P group default qlen 1000
    link/ether 4e:ff:97:4d:30:57 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.101/24 brd 192.168.1.255 scope global eth0
       valid_lft forever preferred_lft forever
```

2） 自动永久配置以太网 IP 地址

通过 ifconfig 命令和 ip 命令配置的 IP 地址断电之后就会丢失，如果需要使 IP 地址永久生效，就需要修改网络管理工具相应的配置文件。

MYD-Y6ULX 采用 systemd-networkd 管理网络，以太网开机默认采用自动获取 ip 方式，这里介绍如何配置静态 ip 地址。

- **查看/lib/systemd/network/下面得文件**

```
root@myd-y6ull14x14:~#/lib/systemd/network#  ls  -l
total  20
-rw-r--r--  1  root  root  645  Sep   4  23:35  80-container-host0.network
-rw-r--r--  1  root  root  718  Sep   4  23:35  80-container-ve.network
-rw-r--r--  1  root  root  704  Sep   4  23:35  80-container-vz.network
-rw-r--r--  1  root  root  113  Sep   5  02:38  80-wired.network
-rw-r--r--  1  root  root  441  Sep   4  23:35  99-default.link
```

- **查看 80-wired.network**

```
root@myd-y6ull14x14:~#/lib/systemd/network#  cat  80-wired.network
[Match]
Name=en*  eth*
KernelCommandLine=!nfsroot
[Network]
DHCP=yes
[DHCP]
RouteMetric=10
ClientIdentifier=mac
```

这里可以看到以 eth 开头得网卡设备会以 DHCP 方式自动获取 IP。

文件格式是 数字+名称.network，其实网卡在加载时，会只加载数字最小得文件，如开机会加载我们添加的静态 IP 配置文件：

```
/etc/systemd/network/10-static-eth0.network
/etc/systemd/network/11-static-eth1.network
```

4.2. Wi-Fi

本节主要介绍 Linux 下 Wi-Fi 的配置和使用，通常 Wi-Fi 模块可以支持两种工作模式，分别是 STA 模式和 AP 模式，有些设备还支持 STA 和 AP 模式同时工作。STA 模式允许设备连接外部 Wi-Fi 热点，AP 模式将设备变成 Wi-Fi 热点，供其它设备连接。

MYD-Y6ULX 板载 WM-N-BM-02 模块，当前支持 STA 模式，模块对应的驱动为：

```
root@myd-y6ull14x14:~# lsmod
Module                  Size  Used by
brcmfmac              176188  0
brcmutil               8416  1 brcmfmac
```

驱动加载的过程中会将位于/lib/firmware/brcm 的 Wi-Fi 固件加载到模块内部。Wi-Fi 模块驱动加载成功之后生成 Wi-Fi 设备的网络节点 wlan0，如下所示：

```
root@myd-y6ull14x14:~# ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr d4:12:43:20:a0:e8
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

1） STA 模式手动连接 WiFi 热点

下面尝试手动连接附近的 Wi-Fi 热点”MYIR_TECH"，这是一个采用 WPA2 加密方式的 Wi-Fi 热点。

- **扫描附近 WiFi 热点**

扫描附近的 wifi 热点，得到附近 Wi-Fi 热点列表如下：

```
root@myd-y6ul14x14:~# ifconfig wlan0 up
root@myd-y6ull14x14:~# iw dev wlan0 scan | grep SSID
        SSID: MI8
        SSID: DIRECT-JJFFmsBE
        SSID: feng
        SSID: HP-Print-3C-LaserJet Pro MFP
        SSID: MYIR_TECH
```

```
        SSID: DIRECT-LGDESKTOP-GMPQJ5JmsTZ
        SSID: HUAWEI_B316_CA1C_Guest
        SSID: 360WiFi-4FF7C3
        SSID:
```

- **使用 wpa_supplicant 连接路由器**

执行过程如下：

```
root@myd-y6ull14x14:~# wpa_supplicant -D wext -B -i wlan0 -c  /etc/wpa_supplicant.conf

Successfully initialized wpa_supplicant
root@myd-y6ull14x14:~# [    14.220666] [dhd-wlan0] wl_run_escan : LEGACY_SCAN sync ID: 0, bssidx: 0

root@myd-y6ull14x14:~#
root@myd-y6ull14x14:~# wpa_cli -p/var/run/wpa_supplicant remove_network 0
Selected interface 'wlan0'
OK
root@myd-y6ull14x14:~# wpa_cli -p/var/run/wpa_supplicant ap_scan 1
Selected interface 'wlan0'
OK
root@myd-y6ull14x14:~# wpa_cli -p/var/run/wpa_supplicant add_network
Selected interface 'wlan0'
0
root@myd-y6ull14x14:~# wpa_cli -p/var/run/wpa_supplicant set_network 0 ssid '"MYIR_TECH"'
Selected interface 'wlan0'
OK
root@myd-y6ull14x14:~# wpa_cli -p/var/run/wpa_supplicant set_network 0 key_mgmt WPA-PSK
Selected interface 'wlan0'
OK
root@myd-y6ull14x14:~# wpa_cli -p/var/run/wpa_supplicant set_network 0 psk '"myir2016"'
```

```
Selected interface 'wlan0'
OK
root@myd-y6ull14x14:~# wpa_cli -p/var/run/wpa_supplicant select_network 0
Selected interface 'wlan0'
OK
root@myd-y6ull14x14:~# udhcpc -i wlan0
udhcpc: started, v1.31.0
udhcpc: sending discover
udhcpc: sending select for 192.168.30.109
udhcpc: lease of 192.168.30.109 obtained, lease time 7200
RTNETLINK answers: File exists
/etc/udhcpc.d/50default: Adding DNS 114.114.114.114
/etc/udhcpc.d/50default: Adding DNS 192.168.30.1
root@myd-y6ull14x14:~#
```

- **检查是否连接**

```
root@myd-y6ull14x14:~# wpa_cli -p/var/run/wpa_supplicant status
Selected interface 'wlan0'
bssid=30:fc:68:9a:e8:99
freq=2412
ssid=MYIR_TECH
id=0
mode=station
wifi_generation=4
pairwise_cipher=CCMP
group_cipher=CCMP
key_mgmt=WPA2-PSK
wpa_state=COMPLETED
ip_address=192.168.30.100
address=d4:12:43:20:a0:e8
uuid=2d50eba8-5107-5d98-9a91-566312d04f00
```

手动连接在重新开机之后，连接就会丢失，如果需要开机时自动连接到“MYIR_TECH"，则需要将“MYIR_TECH"热点的 SSID 和 KEY 写入到/etc/wpa_supplicant.conf 配置文件中，如下所示：

```
root@myd-y6ull14x14:~# cat /etc/wpa_supplicant.conf
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
update_config=1

network={
        ssid="MYIR_TECH"
        #psk="myir@2016"
        psk=280389c71e758f6aa470e3766d791b17ef37a3862775af5af2f54cbdf0e4
7ccf
}
root@myd-y6ull14x14:~#
```

这里介绍用 wpa_passphrase 来生成连接信息。

```
root@myd-y6ull14x14:~# head -n 4 /etc/wpa_supplicant.conf > /etc/wpa_suppl
icant.conf.tmp
root@myd-y6ull14x14:~# wpa_passphrase MYIR_TECH myir@2016 >> /etc/wpa
_supplicant.conf.tmp
root@myd-y6ull14x14:~# mv /etc/wpa_supplicant.conf /etc/wpa_supplicant.conf.
bak
root@myd-y6ull14x14:~# mv /etc/wpa_supplicant.conf.tmp /etc/wpa_supplicant.
conf
root@myd-y6ull14x14:~# cat /etc/wpa_supplicant.conf
```

"MYIR_TECH"指要连接路由器名子，myir@2016 为要连接路由器密码。

初始化 wpa_supplicant,<-B> 在后台运行守护进程,<-c> 配置信息的路径,<-i >监听的 wifi 接口。

```
root@myd-y6ull14x14:~# wpa_supplicant -B -i wlan0 -c /etc/wpa_supplicant.co
nf -D nl80211
```

自动获取 IP：

```
root@myd-y6ull14x14:~# udhcpc -i wlan0
udhcpc: started, v1.31.0
udhcpc: sending discover
udhcpc: sending select for 192.168.40.101
```

```
udhcpc: lease of 192.168.40.101 obtained, lease time 7200
/etc/udhcpc.d/50default: Adding DNS 223.5.5.5
/etc/udhcpc.d/50default: Adding DNS 201.104.111.114
root@myd-y6ull14x14:~# ping 192.168.30.2
PING 192.168.30.2 (192.168.30.2) 56(84) bytes of data.
64 bytes from 192.168.30.2: icmp_seq=1 ttl=64 time=0.106 ms
64 bytes from 192.168.30.2: icmp_seq=2 ttl=64 time=0.087 ms
^C
--- 192.168.30.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1031ms
rtt min/avg/max/mdev = 0.087/0.096/0.106/0.009 ms
```

4.3. 4G 模块

MYD-Y6ULX 开发板提供一个支持 4G 模块的 PCI-E 插槽，此插槽使用 USB 数据线与 4G 模块通讯。当前仅支持移远 EC20 型号。

1） 硬件连接：

1. 安装移远 EC20 模块到 PCI-E 插槽（U12）
2. 将两头 I-PEX 接口的天线安装在移远 EC20 模块和开发板的 J25 位置
3. 安装 SMA 天线到开发板的 J24 位置

2） 查看驱动加载

系统中已经加入 4G 模块的驱动，启动后会自动加载相应驱动,驱动加载成功后会出现对应的/dev/ttyUSB*设备：

```
root@myd-y6ull14x14:~# ls /dev/ttyUSB*
/dev/ttyUSB0  /dev/ttyUSB1  /dev/ttyUSB2  /dev/ttyUSB3
```

3） quectel-CM 拨号联网

系统中已经加入了 quectel-CM 拨号的程序，相较以前老的 ppp 的方式提升了拨号速度，运行 quectel-CM 后会自动生产 wwan0，获取 IP 地址设置 DNS。操作如下：

```
root@myd-y6ull14x14:~# quectel-CM &
[1] 246
root@myd-y6ull14x14:~# [09-09_08:45:56:467] WCDMA&LTE_QConnectManager
_Linux&Android_V1.1.34
[09-09_08:45:56:469] ./quectel-CM profile[1] = (null)/(null)/(null)/0, pincode =
(null)
[09-09_08:45:56:472] Find /sys/bus/usb/devices/1-1.3 idVendor=2c7c idProduct
=0125
[09-09_08:45:56:474] Find /sys/bus/usb/devices/1-1.3:1.4/net/wwan0
[09-09_08:45:56:475] Find usbnet_adapter = wwan0
[09-09_08:45:56:475] Find /sys/bus/usb/devices/1-1.3:1.4/usbmisc/cdc-wdm0
[09-09_08:45:56:475] Find qmichannel = /dev/cdc-wdm0
[09-09_08:45:56:500] cdc_wdm_fd = 7
[09-09_08:45:56:579] Get clientWDS = 18
[09-09_08:45:56:611] Get clientDMS = 1
[09-09_08:45:56:642] Get clientNAS = 3
```

```
[09-09_08:45:56:675] Get clientUIM = 1
[09-09_08:45:56:706] Get clientWDA = 1
[09-09_08:45:56:739] requestBaseBandVersion EC20CEFDKGR06A04M2G
[09-09_08:45:56:835] requestGetSIMStatus SIMStatus: SIM_READY
[09-09_08:45:56:867] requestGetProfile[1] 3gnet///0
[09-09_08:45:56:899] requestRegistrationState2 MCC: 460, MNC: 0, PS: Attached, DataCap: LTE
[09-09_08:45:56:931] requestQueryDataCall IPv4ConnectionStatus: DISCONNECTED
[09-09_08:45:56:996] requestRegistrationState2 MCC: 460, MNC: 0, PS: Attached, DataCap: LTE
[09-09_08:45:57:058] requestSetupDataCall WdsConnectionIPv4Handle: 0xe17cb430
[09-09_08:45:57:122] requestQueryDataCall IPv4ConnectionStatus: CONNECTED
[09-09_08:45:57:155] ifconfig wwan0 up
[09-09_08:45:57:212] busybox udhcpc -f -n -q -t 5 -i wwan0
udhcpc: started, v1.31.0
udhcpc: sending discover
udhcpc: sending select for 10.93.180.112
udhcpc: lease of 10.93.180.112 obtained, lease time 7200
[09-09_08:45:57:708] /etc/udhcpc.d/50default: Adding DNS 211.137.58.20
[09-09_08:45:57:710] /etc/udhcpc.d/50default: Adding DNS 211.137.64.163

root@myd-y6ull14x14:~# ifconfig wwan0
wwan0     Link encap:Ethernet  HWaddr 1a:ae:d6:28:f8:99
          inet addr:10.93.180.112  Bcast:10.93.180.127  Mask:255.255.255.224
          UP BROADCAST RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

然后使用 ping 命令测试连接网络是否正常:

```
root@myd-y6ull14x14:~# ifconfig eth0 down
```

```
root@myd-y6ull14x14:~# ifconfig eth1 down
root@myd-y6ull14x14:~# ifconfig
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:82 errors:0 dropped:0 overruns:0 frame:0
          TX packets:82 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:6220 (6.0 KiB)  TX bytes:6220 (6.0 KiB)

wwan0     Link encap:Ethernet  HWaddr 1a:ae:d6:28:f8:99
          inet addr:10.93.180.112  Bcast:10.93.180.127  Mask:255.255.255.224
          UP BROADCAST RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@myd-y6ull14x14:~# ping www.baidu.com
PING www.baidu.com (36.152.44.95): 56 data bytes
64 bytes from 36.152.44.95: seq=0 ttl=55 time=54.916 ms
64 bytes from 36.152.44.95: seq=1 ttl=55 time=39.646 ms
64 bytes from 36.152.44.95: seq=2 ttl=55 time=48.746 ms
64 bytes from 36.152.44.95: seq=3 ttl=55 time=54.127 ms
64 bytes from 36.152.44.95: seq=4 ttl=55 time=47.566 ms
^C
--- www.baidu.com ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 39.646/49.000/54.916 ms
```

5. 网络应用

设备出厂烧录的 myir-image-full 镜像默认包含了一些常见的网络应用程序，方便用户进行开发或调试。

5.1. PING

PING 主要用来测试网络的连通性，也可以测试网络延迟以及丢包率。按照 4.1.1 中配置好以太网连接之后就可以使用 PING 对网络连接进行简单的测试。

1） 接线与信息输出

通过 CAT6 网线将设备连接到交换机或路由器，控制台会显示内核输出的连接信息，如下：

```
[ 6601.055679] SMSC LAN8710/LAN8720 20b4000.ethernet-1:01: attached PHY driver [SMSC LAN8710/LAN8720] (mii_bus:phy_addr=20b4000.ethernet-1:01, irq=POLL)
```

2） 测试外网网址

```
root@myd-y6ull14x14:~# ping www.baidu.com -I eth0
PING www.baidu.com (112.80.248.75) 56(84) bytes of data.
64 bytes from 112.80.248.75 (112.80.248.75): icmp_seq=1 ttl=56 time=28.3 ms
64 bytes from 112.80.248.75 (112.80.248.75): icmp_seq=2 ttl=56 time=26.4 ms
64 bytes from 112.80.248.75 (112.80.248.75): icmp_seq=3 ttl=56 time=34.5 ms
64 bytes from 112.80.248.75 (112.80.248.75): icmp_seq=4 ttl=56 time=106 ms
```

注：ping 公网需要确保 DNS 正常工作。

上面结果显示 www.baidu.com 经过域名解析之后的 IP 地址为 112.80.248.75，icmp_seq 代表 icmp 包的编号，如果编号连续说明没有丢包；time 代表响应的延迟时间，当然这个时间越短越好。除了对以太网进行测试，ping 命令也可以用于测试 Wi-Fi。

5.2. SSH

SSH 为 Secure Shell 的缩写，由 IETF 的网络小组（Network Working Group）所制定；SSH 为建立在应用层基础上的安全协议， 是较可靠，专为远程登录会话和其他网络服务提供安全性的协议。通常 Linux 平台下使用 dropbear 或 OpenSSH 来实现 SSH 的服务端和客户端。下面在以太网连接上分别测试 SSH 客户端和服务端的使用。当前出厂默认包含 openssh 7.6p1（http://www.openssh.com/）提供的客户端和服务程序。

首先按照 4.1.1 节配置好开发板以太网接口到 SSH 服务器的连接，配置后的以太网卡地址如下：

```
root@myd-y6ull14x14:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 5a:6a:a5:1f:d3:5b
          inet addr:192.168.30.202  Bcast:192.168.30.255
          Mask:255.255.255.0
          inet6 addr: fe80::586a:a5ff:fe1f:d35b/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:102 errors:0 dropped:0 overruns:0 frame:0
          TX packets:163 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:11122 (10.8 KiB)  TX bytes:17354 (16.9 KiB)
```

SSH 服务器的 IP 地址为 192.168.30.185，用 ping 命令测试开发板和 SSH 服务器之间的连接正常之后即可进行下面的测试。(SSH 服务器为具有 SSH 功能的虚拟机或开发板)

- **SSH 客户端测试：**

开发板作为 SSH 客户端连接 SSH 服务器。

在设备上使用 ssh 命令登陆 SSH 服务器，命令和结果如下：

“ssh root@192.168.30.185”中 root 为服务器登录账号，请以实际为准。

```
myir@myir-server1:~$ ssh root@192.168.30.185
The authenticity of host '192.168.30.185 (192.168.30.185)' can't be established.
RSA key fingerprint is SHA256:8Pu4Od1+FdT34uG6eYpRgXbbvPeKOcUzyed/hnpEsHI.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.30.185' (RSA) to the list of known hosts.
```

```
root@myd-y6ull14x14:~#
```

登录成功之后，自动进入 SSH 服务器上的 console 控制台，用户就可以在客户端对远程服务器执行 root 用户权限内的控制。如果需要退出，直接在当前控制台执行"exit"命令即可。

- **SSH 服务端测试：**

开发板作为 SSH 服务端，其它设备远程连接到设备。

由于设备端默认也启动了 SSH 服务，因此我们也可以在其它具有 SSH 客户端的设备上使用 ssh 命令登陆到当前的设备，命令和结果如下：

```
myir@myir-server1:~$ ssh root@192.168.30.202
The authenticity of host '192.168.30.202 (192.168.30.202)' can't be established.
RSA key fingerprint is SHA256:8Pu4Od1+FdT34uG6eYpRgXbbvPeKOcUzyed/hnpEsHI.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.30.202' (RSA) to the list of known hosts.
root@myd-y6ull14x14:~#
```

上面的示例中，我们从远程以 root 账户登录到了设备上，并进入 console 控制台，可以对设备执行 root 用户权限内的控制。如果需要退出，直接在控制台执行"exit"命令即可。

OpenSSH 是使用 SSH 协议远程登录的主要连接工具。它加密所有流量以消除窃听、连接劫持和其他攻击。此外，OpenSSH 还提供一系列大型安全隧道功能、多种身份验证方法和复杂灵活的配置选项。用户可以根据自身需要修改位于/etc/ssh/目录下的配置文件 ssh_config 和 sshd_config。

例如，如果希望 SSH 服务端允许 root 账户不用密码远程登录，则可以修改 SSH 服务端设备上的/etc/ssh/sshd_config，添加下面两行配置。

```
PermitRootLogin yes
PermitEmptyPasswords yes
```

上面的配置有比较大的安全风险，一般用于调试阶段远程部署。实际产品中考虑到安全性，一般都是关掉的。

5.3. SCP

SCP 是 Secure Copy 的缩写，它是 linux 系统下基于 SSH 协议的安全的远程文件拷贝命令，在系统调试阶段非常实用。

在 4.2.2 中我们已经介绍过使用 SSH 协议以及 SSH 客户端和服务端进行远程登录的示例，这里再介绍通过 SCP 命令进行文件远程拷贝的示例：

1） 从远程拷贝文件到本地（）

```
myir@myir-server1:~$ scp uboot.dtb root@192.168.30.202:/home/root/
uboot.dtb
                                                100% 1499KB    1.5MB/s
 00:00
myir@myir-server1:~$
```

注意 192.168.30.202 为开发板 IP； /home/root 为开发板家目录。

进入开发板 home 目录可以看到此文件，如下：

```
root@myd-y6ull14x14:~# ls
uboot.dtb
root@myd-y6ull14x14:~# pwd
/home/root
```

2） 从开发板本地拷贝文件到远程服务器

```
root@myd-y6ull14x14:~# scp 1.sh roy@192.168.30.185:~/
roy@192.168.30.185's password:
1.sh


  100%    0       0.0KB/s    00:00
root@myd-y6ull14x14:~#
```

1.sh 为开发板家目录下面的文件；roy 为远程服务器账号；192.16830.185 为远程服务器 IP, ~为远程服务器家目录

拷贝的过程中需要按照提示输入，验证成功之后文件从设备上拷贝到服务器上指定账户的$HOME 目录。

通过添加”-r“参数，还可以进行目录的拷贝，具体操作请参照 scp 命令的帮助。

5.4. FTP

FTP(File Transfer Protocol)是一种进行文件传输的网络协议。该协议遵循客户机-服务器通信模型。要使用 FTP 传输文件，用户需要运行一个 FTP 客户端程序，并启动与运行 FTP 服务器软件的远程计算机的连接。建立连接后，客户端可以选择发送和/或接收文件的副本。FTP 服务器在 TCP 端口 21 上侦听来自 FTP 客户端的连接请求。当接收到请求时，服务器使用这个端口来控制连接，并打开一个单独的端口来传输文件数据。

默认出厂的 myir-image-full 镜像包含有 FTP 客户端程序 ftp 和服务端程序 proftpd。启动开发板，设备端 FTP 根目录位于/var/lib/ftp，下面测试在局域网内使用 ftp 命令匿名登陆到目标设备并从开发主机（IP 地址为 192.168.0.2)向目标设备（IP 地址为 192.168.0.60）传输文件的示例。

1） 开发主机使用 FTP 登录目标设备

开发主机为 Windows10 系统的 PC 机，打开 cmd 命令窗口，用 FTP 登陆到开发板，用户名为 ftp，密码任意。

```
C:\Users\myir>ftp 192.168.30.185
连接到 192.168.30.185。
220 ProFTPD Server (ProFTPD Default Installation) [192.168.30.185]
500 OPTS UTF8 not understood
用户(192.168.30.185:(none)): ftp
331 Anonymous login ok, send your complete email address as your password
密码:
230 Anonymous access granted, restrictions apply
ftp>
```

此时当前目录已经是开发板/var/lib/ftp，然后可以下载和上传文件。

2） 目标设备上创建测试文件

```
root@myd-y6ull14x14:~#/var/lib/ftp# touch test
root@myd-y6ull14x14:~#/var/lib/ftp# ls
aaa  test
root@myd-y6ull14x14:~#/var/lib/ftp# rm -r aaa
root@myd-y6ull14x14:~#/var/lib/ftp# touch aaa
root@myd-y6ull14x14:~#/var/lib/ftp# ls
aaa  test
```

```
root@myd-y6ull14x14:~#/var/lib/ftp#
```

3） 开发主机 FTP 查看当前目录

```
ftp>ls
200 PORT command successful
150 Opening ASCII mode data connection for file list
aaa
test
226 Transfer complete
ftp:收到 14 字节，用时 0.00 秒 14000.00 字节/秒
```

4） 下载文件

FTP 命令行支持使用 mget 子命令下载多个文件，但是每个文件中间需要用户确认，如下所示：

```
ftp>mget aaa test
200 Type set to A
Mget aaa? y
200 PORT command successful
150 Opening ASCII mode data connection for aaa
226 Transfer complete
Mget test? y
200 PORT command successful
150 Opening ASCII mode data connection for test
226 Transfer complete
ftp>
```

5） 上传文件

FTP 文件传输支持 ASCII 模式和 Binary 模式，上面下载文件默认采用的是 ASCII 模式，如果是传输二进制文件，如目标设备上的可执行程序，尽量采用 Bianry 模式。如下所示：

```
ftp> binary
200 Type set to I
ftp> put myapp
local: myapp remote: myapp
200 PORT command successful.
150 Opening BINARY mode data connection for 'myapp'.
```

```
226 Transfer complete.
1024 bytes sent in 0.02 secs (63.7308 kB/s)
ftp>exit
D:\01-Document>
```

FTP 可追溯到 20 世纪 70 年代初，编写时没有任何安全考虑。它不对任何内容使用加密。登录凭据（如用户名和密码）以及您下载或上传的数据以明文传输。因此它不太适合在 Internet 上传输敏感信息，然而考虑到其良好的兼容性和稳定性，在局域网内使用还是很方便的。

5.5. TFTP

与 FTP 一样，TFTP 使用客户端和服务器软件在两台设备之间进行连接和传输文件，但不同的是 TFTP 使用的是 UDP 协议，不具备登录功能，它非常简洁，特别适合在设备和服务器端传输和备份固件，配置文件等信息。例如常见的 u-boot 中就支持 TFTP 协议，可以通过网络加载服务器端的 Linux 系统并实现网络启动的功能。

默认的镜像文件包含 busybox 提供的 tftp 客户端程序，其命令语法如下：

```
root@myd-y6ull14x14:~# tftp --help
BusyBox v1.31.0 (2020-09-05 00:06:04 UTC) multi-call binary.

Usage: tftp [OPTIONS] HOST [PORT]
```

详细参数说明如下：

- -g： 获取文件
- -p： 上传文件
- -l ： 本地文件
- -r ： 远程文件
- HOST: 远程主机 IP 地址
- [PORT]: 可忽略，默认为 69

TFTP 服务端可以选择 Linux 平台下的 tftp-hpa,也可以选择 windows 平台下的 tftpd32/64(http://tftpd32.jounin.net/tftpd32_download.html)。下面以 ubuntu 平台为例说明 tftp 服务端的配置。

1) 安装 TFTP 服务端

```
$ sudo apt-get install tftp-hpa tftpd-hpa
```

2) 配置 TFTP 服务

创建 TFTP 服务器工作目录,并打开 TFTP 服务配置文件,如下:

```
$ mkdir -p <WORKDIR>/tftpboot
$ chmod -R 777 <WORKDIR>/tftpboot
$ sudo vi /etc/default/tftpd-hpa
```

修改或添加以下字段：

```
TFTP_DIRECTORY="<WORKDIR>/tftpboot"
TFTP_OPTIONS="-l -c -s"
```

3) 重启 TFTP 服务

```
$ sudo service tftpd-hpa restart
```

配置好 tftp 服务端之后，将一个测试文件 zImage 放置到上面配置的<WORKDIR>/tftpboot/目录，就可以在目标设备上使用 tftp 客户端进行文件的下载和上传了。

```
root@myd-y6ull14x14:~# tftp -g -r zImage -l zImage 192.168.0.2
```

上面的命令会把 tftp 服务端<WORKDIR>/tftpboot 目录下的 zImage 下载到目标设备当前目录下。

```
root@myd-y6ull14x14:~# tftp -p -l config -r config_01 192.168.0.2
```

上面的命令会把目标设备上当前目录下的 config 文件上传到 tftp 服务端之前配置的<WORKDIR>/tftpboot 目录下，并重新命名为 config_01。

5.6. udhcpc

DHCP（动态主机配置协议）是一个局域网的网络协议。指的是由服务器控制一段 IP 地址范围，客户机登录服务器时就可以自动获得服务器分配的 IP 地址和子网掩码。

DHCP 也包含服务器端和客户端两种角色，在 4.1.1 中我们已经测试过使用 DHCP 客户端模式自动获取 IP 地址；这里再介绍一下使用 udhcpc 命令手动获取 IP 地址的方法，方便用户在调试网络时使用。

用 CAT6 网线连接开发板和路由器，使用命令手动为 eth0 网卡分配 IP 地址，观察 dhcp 获取 ip 的过程。

- **使用 udhcpc 命令配置 IP 地址**

用网线连接开发板和路由器，针对 eth0 网卡进行 dhclient 测试，观察 dhcp 获取 ip 的过程

```
root@myir:/etc# udhcpc -i eth0
Internet Systems Consortium DHCP Client 4.4.2
Copyright 2004-2020 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/eth0/ba:27:38:19:5d:27
Sending on   LPF/eth0/ba:27:38:19:5d:27
Sending on   Socket/fallback
DHCPREQUEST for 192.168.40.107 on eth0 to 255.255.255.255 port 67
DHCPACK of 192.168.40.107 from 192.168.40.1
RTNETLINK answers: File exists
bound to 192.168.40.107 -- renewal in 3355 seconds.
```

由上面测试 log 可以观察到 DHCP 通讯获取地址的四个过程：DHCP Discover、DHCP Offer、DHCP Request、DHCP ACK。用 Ctrl+c 结束 DHCP 测试。

5.7. IPTables

iptables 是一个用于 IPv4 包过滤和 NAT 的管理工具。它用于设置、维护和检查 Linux 内核中的 IP 包过滤规则表。可以定义几个不同的表。每个表包含许多内置链，也可以包含用户定义的链。每个链是一个规则列表，它可以匹配一组数据包。每个规则指定如何处理匹配的数据包。

使用 Linux 系统的设备通常使用 iptables 工具来配置防火墙。iptables 就根据包过滤规则所定义的方法来处理各种数据包，如放行（accept）、拒绝（reject）和丢弃（drop）等。下面使用 iptables 来测试拦截 icmp 包，禁止网络上的其它设备对其进行 ping 探测。具体命令使用参见：https://linux.die.net/man/8/iptables

1） 配置目标设备 iptables

在目标设备上使用 iptables 配置丢弃输入的 icmp 包，不回应其他主机的 ping 探测，命令如下：

```
root@myd-y6ull14x14:~# iptables -A INPUT -p icmp --icmp-type 8 -j DROP
root@myd-y6ull14x14:~# iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-A INPUT -p icmp -m icmp --icmp-type 8 -j DROP
```

2） 测试 ping 目标设备

在开发主机上 ping 目标设备，并指定 deadline 为 10，结果如下：

```
PC$ ping 192.168.30.185 -w 10
PING 192.168.30.2 (192.168.30.2) 56(84) bytes of data.

--- 192.168.0.60 ping statistics ---
10 packets transmitted, 0 received, 100% packet loss, time 9064ms
```

以上结果表明，设置防火墙后开发主机无法 ping 通目标设备。

3） 删掉对应的防火墙规则

```
root@myd-y6ull14x14:~# iptables -F
root@myd-y6ull14x14:~# iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
```

4) **再次测试 ping 目标设备**

```
PC$ ping 192.168.0.60 -w 5
PING 192.168.0.60 (192.168.0.60) 56(84) bytes of data.
64 bytes from 192.168.0.60: icmp_seq=1 ttl=64 time=0.254 ms
64 bytes from 192.168.0.60: icmp_seq=2 ttl=64 time=0.219 ms
64 bytes from 192.168.0.60: icmp_seq=3 ttl=64 time=0.222 ms
64 bytes from 192.168.0.60: icmp_seq=4 ttl=64 time=0.226 ms
64 bytes from 192.168.0.60: icmp_seq=5 ttl=64 time=0.238 ms
64 bytes from 192.168.0.60: icmp_seq=6 ttl=64 time=0.236 ms

--- 192.168.0.60 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 4996ms
rtt min/avg/max/mdev = 0.219/0.232/0.254/0.019 ms
```

清除 iptables 规则之后，再次从开发主机 ping 目标设备，就可以 ping 通了。上述示例只是一个简单的演示，实际上 iptables 配合各种规则可以实现非常强大的功能，这里就不详细介绍了。

5.8. Ethtool

ethtool 是一个查看和修改以太网设备参数的工具，在网络调试阶段具有一定的作用，下面使用该命令查看一下以太网卡的信息，并尝试修改其参数。

首先，我们通过 ethtool -h 查看该命令的帮助信息

```
root@myd-y6ull14x14:~# ethtool --help
ethtool version 5.2
Usage:
        ethtool DEVNAME Display standard information about device
        ethtool -s|--change DEVNAME     Change generic options
                [ speed %d ]
                [ duplex half|full ]
                [ port tp|aui|bnc|mii|fibre ]
                [ mdix auto|on|off ]
                [ autoneg on|off ]
                [ advertise %x ]
                [ phyad %d ]
                [ xcvr internal|external ]
                [ wol p|u|m|b|a|g|s|f|d... ]
                [ sopass %x:%x:%x:%x:%x:%x ]
                [ msglvl %d | msglvl type on|off ... ]
        ethtool -a|--show-pause DEVNAME Show pause options.....
```

查看当前设备以太网卡的基本信息:

```
root@myd-y6ull14x14:~# ethtool eth0
Settings for eth0:
        Supported ports: [ TP MII ]
        Supported link modes:   10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
        Supported pause frame use: Symmetric
        Supports auto-negotiation: Yes
        Supported FEC modes: Not reported
        Advertised link modes:  10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
```

```
        Advertised pause frame use: Symmetric
        Advertised auto-negotiation: Yes
        Advertised FEC modes: Not reported
        Link partner advertised link modes:  10baseT/Half 10baseT/Full
        Link partner advertised pause frame use: Symmetric Receive-only
        Link partner advertised auto-negotiation: Yes
        Link partner advertised FEC modes: Not reported
        Speed: 10Mb/s
        Duplex: Full
        Port: MII
        PHYAD: 1
        Transceiver: internal
        Auto-negotiation: on
        Supports Wake-on: g
        Wake-on: d
        Link detected: yes
```

通过 ethtool 命令可以查看到当前以太网卡支持的连接模式为十兆，百兆和千兆半双工与全双工六种模式，当前连接状态为协商的千兆，全双工模式，使用 MII 接口，PHY 地址为 4 等等。

我们还可以使用 ethtool 工具对以太网的参数进行设置，这些在进行以太网调试和诊断的时候有一定的作用，例如我们强制将以太网设置为百兆全双工，并且关闭自协商，命令如下：

```
root@myd-y6ull14x14:~# ethtool -s eth0 speed 100 duplex full autoneg off
```

关于 ethtool 的更多说明请参考：http://man7.org/linux/man-pages/man8/ethtool.8.html

5.9. iPerf3

iPerf3 是在 IP 网络上主动测量最大可实现带宽的工具。 它支持调节测试时间、缓冲区大小和协议(TCP、UDP、带有 IPv 的 SCTP)等各种参数。 iPerf3 按角色可以分为服务端模式或客户端模式，我们可以用它来测试和查看 TCP 模式下的网络带宽，TCP 窗口值，重传的概率等，也可以测试指定 UDP 带宽下丢包率，延迟和抖动情况。

我们以一台 ubuntu 16.04 系统，带千兆网卡的服务器作为 iperf3 的服务端，被测试的设备作为客户端分别测试设备以太网卡 TCP 和 UDP 的性能。

首先在服务器上安装 iperf3，如下：

```
PC $ sudo apt-get install iperf3
```

将服务器和设备通过 CAT6 网线直连，并配置好各自的 IP 地址。例如我们设置服务器 ip 为 192.168.30.2，设备 IP 为 192.168.30.185，并使用 ping 命令测试确保它们之间是连通的。

注意：尽量不要连接路由器或交换机，以免测试结果受到中间设备的影响。

1）测试 TCP 性能

- **服务端（192.168.30.168）：** 服务器上 iperf3 使用-s 参数表示工作在服务端模式。

```
PC $ $ iperf3 -s -i 2
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

- **客户端（192.168.30.185）：** 设备上 iperf3 工作在客户端，TCP 模式，其中参数说明如下：
  - -c 192.168.30.168 ： 工作在客户端，连接服务端 192.168.30.168
  - -i 2 ： 测试结果报告时间间隔为 2 秒
  - -t 10 ： 总测试时长为 10 秒

```
root@myd-y6ull14x14:~# iperf3  -c 192.168.30.168 -i 2 -t 60
Connecting to host 192.168.30.3, port 5201
[  4] local 192.168.30.156 port 35213 connected to 192.168.30.3 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-2.00   sec  22.9 MBytes  96.2 Mbits/sec    0    229 KBytes
[  4]   2.00-4.00   sec  22.1 MBytes  92.5 Mbits/sec    0    229 KBytes
[  4]   4.00-6.00   sec  22.4 MBytes  93.9 Mbits/sec    0    229 KBytes
[  4]   6.00-8.00   sec  22.7 MBytes  95.1 Mbits/sec    0    229 KBytes
```

```
[  4]   8.00-10.00  sec  22.4 MBytes  93.8 Mbits/sec    0    229 KBytes
[  4]  10.00-12.00  sec  22.4 MBytes  93.8 Mbits/sec    0    229 KBytes
[  4]  12.00-14.00  sec  22.4 MBytes  93.8 Mbits/sec    0    229 KBytes
[  4]  14.00-16.00  sec  22.7 MBytes  95.1 Mbits/sec    0    229 KBytes
[  4]  16.00-18.00  sec  22.4 MBytes  93.8 Mbits/sec    0    229 KBytes
[  4]  18.00-20.00  sec  22.4 MBytes  93.8 Mbits/sec    0    229 KBytes
[  4]  20.00-22.00  sec  22.4 MBytes  93.9 Mbits/sec    0    229 KBytes
[  4]  22.00-24.00  sec  22.7 MBytes  95.1 Mbits/sec    0    229 KBytes
[  4]  24.00-26.00  sec  22.4 MBytes  93.8 Mbits/sec    0    229 KBytes
[  4]  26.00-28.00  sec  22.4 MBytes  93.8 Mbits/sec    0    229 KBytes
[  4]  28.00-30.00  sec  22.4 MBytes  93.8 Mbits/sec    0    229 KBytes
[  4]  30.00-32.00  sec  22.7 MBytes  95.1 Mbits/sec    0    229 KBytes
[  4]  32.00-34.00  sec  22.4 MBytes  93.9 Mbits/sec    0    229 KBytes
[  4]  34.00-36.00  sec  22.4 MBytes  93.8 Mbits/sec    0    229 KBytes
[  4]  36.00-38.00  sec  22.4 MBytes  93.8 Mbits/sec    0    229 KBytes
[  4]  38.00-40.00  sec  22.7 MBytes  95.0 Mbits/sec    0    229 KBytes
[  4]  40.00-42.00  sec  22.4 MBytes  93.9 Mbits/sec    0    229 KBytes
[  4]  42.00-44.00  sec  22.4 MBytes  93.8 Mbits/sec    0    229 KBytes
[  4]  44.00-46.00  sec  22.4 MBytes  93.8 Mbits/sec    0    229 KBytes
[  4]  46.00-48.00  sec  22.7 MBytes  95.0 Mbits/sec    0    229 KBytes
[  4]  48.00-50.00  sec  22.4 MBytes  93.9 Mbits/sec    0    229 KBytes
[  4]  50.00-52.00  sec  22.4 MBytes  93.9 Mbits/sec    0    229 KBytes
[  4]  52.00-54.00  sec  22.4 MBytes  93.8 Mbits/sec    0    229 KBytes
[  4]  54.00-56.00  sec  22.4 MBytes  93.8 Mbits/sec    0    229 KBytes
[  4]  56.00-58.00  sec  22.7 MBytes  95.1 Mbits/sec    0    229 KBytes
[  4]  58.00-60.00  sec  22.4 MBytes  93.8 Mbits/sec    0    229 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-60.00  sec   674 MBytes  94.2 Mbits/sec    0             sender
[  4]   0.00-60.00  sec   673 MBytes  94.1 Mbits/sec                  receive
r
iperf Done.
```

客户端经过 60 秒之后测试结束并显示上面的测试结果，表明 TCP 带宽为 94.2Mbps 左右，没有重传，测试时 TCP 窗口值为 229 KBytes.

2）测试 UDP 性能

- **服务端（192.168.30.168）：** 服务器上继续运行 iperf3 使用-s 参数表示工作在服务端模式。

```
PC $ iperf3 -s -i 2
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

- **客户端（192.168.30.3）：** 设备上 iperf3 工作在客户端，UDP 模式，其中参数说明如下：
  - -u ： 工作在 UDP 模式
  - -c 192.168.30.168 ： 工作在客户端，连接服务端 192.168.30.168
  - -i 2 ： 测试结果报告时间间隔为 2 秒
  - -t 10 ： 总测试时长为 10 秒
  - -b 1G ： 设定 UDP 传输带宽为 1Gbps.

```
root@myd-y6ull14x14:~# iperf3 -c 192.168.30.3 -u -i 2 -t 60 -b 1G -R
Connecting to host 192.168.30.3, port 5201
Reverse mode, remote host 192.168.30.3 is sending
[  4] local 192.168.30.156 port 49521 connected to 192.168.30.3 port 5201
[ ID] Interval           Transfer     Bandwidth       Jitter   Lost/Total Datagrams
[  4]   0.00-2.00   sec  21.8 MBytes  91.5 Mbits/sec  0.175 ms  29/2822 (1%)
[  4]   2.00-4.00   sec  22.8 MBytes  95.8 Mbits/sec  0.194 ms  0/2923 (0%)
[  4]   4.00-6.00   sec  22.9 MBytes  95.9 Mbits/sec  0.168 ms  0/2926 (0%)
[  4]   6.00-8.00   sec  22.8 MBytes  95.8 Mbits/sec  0.190 ms  0/2924 (0%)
[  4]   8.00-10.00  sec  22.9 MBytes  95.8 Mbits/sec  0.176 ms  0/2925 (0%)
[  4]  10.00-12.00  sec  22.8 MBytes  95.8 Mbits/sec  0.173 ms  0/2924 (0%)
[  4]  12.00-14.00  sec  22.8 MBytes  95.8 Mbits/sec  0.174 ms  0/2924 (0%)
[  4]  14.00-16.00  sec  22.9 MBytes  95.8 Mbits/sec  0.172 ms  0/2925 (0%)
[  4]  16.00-18.00  sec  22.8 MBytes  95.8 Mbits/sec  0.176 ms  0/2924 (0%)
[  4]  18.00-20.00  sec  22.9 MBytes  95.8 Mbits/sec  0.207 ms  0/2925 (0%)
[  4]  20.00-22.00  sec  22.8 MBytes  95.8 Mbits/sec  0.168 ms  0/2924 (0%)
[  4]  22.00-24.00  sec  22.8 MBytes  95.8 Mbits/sec  0.166 ms  0/2924 (0%)
```

```
[  4]  24.00-26.00  sec  22.9 MBytes  95.8 Mbits/sec  0.183 ms  0/2925 (0%)
[  4]  26.00-28.00  sec  22.8 MBytes  95.8 Mbits/sec  0.174 ms  0/2924 (0%)
[  4]  28.00-30.00  sec  22.9 MBytes  95.8 Mbits/sec  0.184 ms  0/2925 (0%)
[  4]  30.00-32.00  sec  22.8 MBytes  95.8 Mbits/sec  0.171 ms  0/2924 (0%)
[  4]  32.00-34.00  sec  22.9 MBytes  95.8 Mbits/sec  0.215 ms  0/2925 (0%)
[  4]  34.00-36.00  sec  22.8 MBytes  95.8 Mbits/sec  0.155 ms  0/2924 (0%)
[  4]  36.00-38.00  sec  22.8 MBytes  95.8 Mbits/sec  0.176 ms  0/2924 (0%)
[  4]  38.00-40.00  sec  22.9 MBytes  95.8 Mbits/sec  0.207 ms  0/2925 (0%)
[  4]  40.00-42.00  sec  22.8 MBytes  95.8 Mbits/sec  0.185 ms  0/2924 (0%)
[  4]  42.00-44.00  sec  22.9 MBytes  95.8 Mbits/sec  0.200 ms  0/2925 (0%)
[  4]  44.00-46.00  sec  22.8 MBytes  95.8 Mbits/sec  0.178 ms  0/2924 (0%)
[  4]  46.00-48.00  sec  22.8 MBytes  95.8 Mbits/sec  0.204 ms  0/2924 (0%)
[  4]  48.00-50.00  sec  22.9 MBytes  95.8 Mbits/sec  0.173 ms  0/2925 (0%)
[  4]  50.00-52.00  sec  22.8 MBytes  95.8 Mbits/sec  0.171 ms  0/2924 (0%)
[  4]  52.00-54.00  sec  22.9 MBytes  95.8 Mbits/sec  0.163 ms  0/2925 (0%)
[  4]  54.00-56.00  sec  22.8 MBytes  95.8 Mbits/sec  0.179 ms  0/2924 (0%)
[  4]  56.00-58.00  sec  22.8 MBytes  95.8 Mbits/sec  0.174 ms  0/2924 (0%)
[  4]  58.00-60.00  sec  22.9 MBytes  95.8 Mbits/sec  0.178 ms  0/2925 (0%)
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datag
rams
[  4]  0.00-60.00  sec  685 MBytes  95.7 Mbits/sec  0.177 ms  29/87640 (0.
033%)
[  4] Sent 87640 datagrams

iperf Done.[  5]  0.00-10.00  sec  119 MBytes  100 Mbits/sec  0.078 ms
0/85604 (0%)  receiver
iperf Done.
```

客户端经过 60 秒之后测试结束并显示上面的测试结果，表明 UDP 没有丢包。

iperf3 在测试的过程中还有很多参数可以配置，用户可以根据实际应用需要进行有针对性的调整测试。比如可以增大-t 参数的值进行长时间压力测试，或者指定-P 参数进行多个连接并发的压力测试等。关于 iperf3 测试的更多信息请参考：https://iperf.fr/iperf-doc.php#3doc。

6. 图形系统

5.1 QT5

QT 是一种跨平台 C++图形用户界面应用程序开发框架。它既可以开发 GUI 程序，也可用于开发非 GUI 程序，比如控制台工具和服务器。Qt 是面向对象的框架，使用特殊的代码生成扩展以及一些宏，Qt 很容易扩展，并且允许真正地组件编程。

开发板会在出厂的时候烧写带有 Qt 运行时库的系统，并且提供了一个丰富的 HMI 演示系统，具体内容可以查看《MEasy HMI2.0 开发手册》。

1） 获取 qt 的信息

首先查看当前系统的 QT 版本，如下：

```
root@myd-y6ull14x14:~# ls /usr/lib/*Qt5Core*
/usr/lib/libQt5Core.so.5        /usr/lib/libQt5Core.so.5.13.2
/usr/lib/libQt5Core.so.5.13
```

2） QT 运行环境介绍

在运行 Qt 应用程序时，可以根据不同的软硬件要求，对 Qt 的运行环境，如平台插件，显示参数，输入设备以及光标指针等进行适当的配置。

- **平台插件配置**

在嵌入式 Linux 系统上，可以使用多个平台插件：EGLFS，LinuxFB，DirectFB 或 Wayland。但是，这些插件的可用性取决于实际硬件平台的特性以及 Qt 的配置方式。EGLFS 是许多主板上的默认插件。如果不合适，请使用 QT_QPA_PLATFORM 环境变量来请求另一个插件。另外，对于快速测试，请使用-platform 具有相同语法的命令行参数。例如在用户控制台下，可以使用如下命令进行配置。

```
root@myir:/home# export QT_QPA_PLATFORM=linuxfb
root@myir:/home# ./mxapp2
```

或者使用-platform linuxfb 指令平台插件，如下：

```
root@myir:/home# ./mxapp2 -platform linuxfb
```

对于包含 GPU 的现代嵌入式 Linux 设备，推荐使用 EGLFS 插件，使用 GPU 进行渲染。如果嵌入式 Linux 设备不包含 GPU 或者不支持 5.1 节中 OpenGL ES 测试，则只能指定 linuxfb 平台插件通过 CPU 进行软件的渲染了。

注意：从 Qt 5.0 开始，Qt 不再具有自己的窗口系统（QWS）实现，因此不再支持 Qt 早期版本上使用的 -qws 参数。

● 显示参数配置

QT 应用程序可以通过 QScreen 类或者 QDesktopWidget 获取屏幕显示相关的参数，从而编写跟屏幕匹配的应用。通过 QScreen 或 QDesktopWidget 获取屏幕分辨率，颜色深度一般是没有问题的，但由于显示驱动的原因有时候获取的物理尺寸就不一定是正确的了。此时可以通过配置和调整下面的参数，使实际界面上显示的元素适合显示屏幕的大小。

该插件通过 Linux 的 fbdev 子系统直接写入帧缓冲区。仅支持软件渲染的内容。请注意，在某些设置下，显示性能可能会受到限制。

但是，由于在 Linux 内核中已弃用 fbdev，因此从 Qt 5.9 开始，还提供了 DRM 哑缓存支持。要使用它，请将 QT_QPA_FB_DRM 环境变量设置为非零值。设置后，只要系统支持哑缓存，/dev/fb0 就不会访问旧式帧缓存设备。而是通过 DRM API 设置呈现，类似于 eglfs_kmsEGLFS 中的后端。输出经过双缓冲和页面翻转，也为软件渲染的内容提供了适当的垂直同步。

注意：使用哑缓冲区时，以下描述的选项均不适用，因为会自动查询诸如物理和逻辑屏幕尺寸之类的属性。该 linuxfb 插件允许您通过 QT_QPA_PLATFORM 环境变量或-platform 命令行选项指定其他设置。例如，QT_QPA_PLATFORM=linuxfb:fb=/dev/fb1 指定/dev/fb1 必须使用 framebuffer 设备而不是 default fb0。要指定多个设置，请用冒号（:)分隔。

表 5-1. QT fb 插件相关的环境变量

环境变量	描述
fb=/dev/fbN	指定帧缓冲设备。在多个显示器设置上，此设置使您可以在不同的显示器上运行该应用程序。当前，无法从一个 Qt 应用程序中使用多个帧缓冲区。
size=<width>x<height>	指定屏幕尺寸（以像素为单位）。该插件尝试从帧缓冲设备查询物理和逻辑的显示尺寸。但是，此查询可能并不总是会导致正确的结果。可能需要明确指定这些值。
mmsize=<width>x<height>	指定物理宽度和高度（以毫米为单位）。
offset=<width>x<height>	指定屏幕的左上角偏移量（以像素为单位）。默认位置

	为(0, 0)。
nographicsmodeswitch	指定不将虚拟终端切换到图形模式（KD_GRAPHICS）。通常，启用图形模式会禁用闪烁的光标和屏幕空白。但是，设置此参数后，这两个功能也会被跳过。
tty=/dev/ttyN	覆盖虚拟控制台。仅在 nographicsmodeswitch 未设置时使用。

从 Qt 5.9 开始，就窗口大小调整策略而言，EGLFS 和 LinuxFB 的行为已同步：使用两个平台插件，第一个顶级窗口被强制覆盖整个屏幕。如果不需要这样做，请将 QT_QPA_FB_FORCE_FULLSCREEN 环境变量设置为 0 从早期的 Qt 版本恢复行为。

- **输入设备配置**

如果要启用 tslib 支持，需要将 QT_QPA_EGLFS_TSLIB（for eglfs）或 QT_QPA_FB_TSLIB（for linuxfb）环境变量设置为 1。关于 tslib 的具体使用方法参考 https://github.com/libts/tslib/blob/master/README.md。

注意：tslib 输入处理程序常用于电阻触摸，会生成鼠标事件并仅支持单点触摸，初始使用还需要进行屏幕校准。

3） 启动 Qt 程序

通常使用触摸的设备都不需要显示鼠标指针，用户可以在执行应用前设置环境变量将鼠标指针隐藏。

如果使用的是 eglfs 平台插件，设置如下：

```
export QT_QPA_EGLFS_HIDECURSOR=1
```

如果使用的是 linuxfb 平台插件，则设置如下：

```
export QT_QPA_FB_HIDECURSOR=1
```

MYD-Y6ULX 开发板出厂的 myir-image-full 镜像带有 MEasy HMI 演示程序，使用 linuxfb 平台插件和 EvdevTouch 触摸处理程序，对应启动命令程序如下：

```
root@myd-y6ull14x14:/home/root/# export QT_QPA_FB_HIDECURSOR=1
root@myd-y6ull14x14:/home/root/# ./mxapp2 -platform linuxfb &
qt.qpa.input: xkbcommon not available, not performing key mapping
qml: index=0
qml: currentIndex=0
```

```
qml: index=0
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
```

mxapp2 默认已经启动，如果需要运行自己的 Qt 应用程序，需要先终止 mxapp2 再启动其他应用。

7. 多媒体

7.1. Camera

本章节使用开源的 uvc_stream 演示 camera 的使用。

MYD-Y6ULX 上提供一个并行 Camera 接口(J9)，可以连接 MY-CAM011B 型号的 Camera 模块，模块之间使用 FPC 线连接。由于信号序列影响，请勿直接将其它型号的 Camera 的模块插入，否则会引起模块或开发板的损坏。

uvc_stream 是通过的网络传输数据，需要先设置好 MYD-Y6ULX 板的以太网 IP 地址，对应系统中的 eth0 设备。Linux 系统中的 MY-CAM011B 模块的设备，可通通过 v4l2-ctl 命令来查询到，输出信息的 i.MX6S_CSI 表示 Camera 控制器，对应设备是 /dev/video1。uvc_stream 参数中'-y'是使用 yuyv 方式，'-P'后面是设置 web 界面的登录密码，用户名默认为 uvc_user。'-r'是指定分辨率，当前仅支持 800x600。可以用 ctrl + c 来停止。

1） 查看设备信息

```
root@myd-y6ull14x14:~# Ifconfig eth0 192.168.30.42
root@myd-y6ull14x14:~# v4l2-ctl --list-devices
i.MX6S_CSI (platform:21c4000.csi):
        /dev/video1

pxp (pxp_v4l2):
        /dev/video0
```

2） uvc_stream 进行预览

```
# ./uvc_stream -d /dev/video1 -y -P 123456 -r 800x600
```

uvc_stream 提供两种 web 功能，snapshot 和 streaming。snapshot 的请求 URL 是 snapshot.jpeg，streaming 的请求 URL 是 stream.mjpeg。PC 和开发板在同一网络内时，打开流览器，输入地址 http://192.168.30.202:8080/stream.mjpeg，可以看到有登录框，输入用户名为 uvc_user，密码为 123456，就可以看到从 MY-CAM011B 实时采集到的图像了。

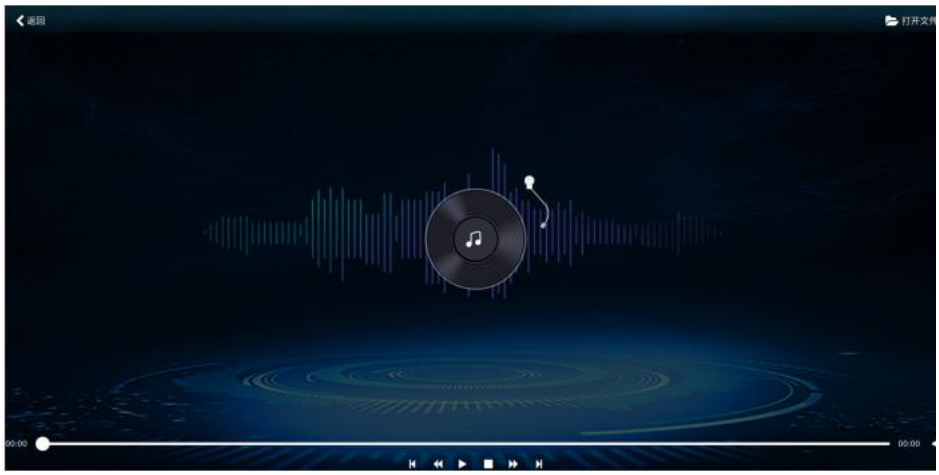
7.2. Auido

MYD-Y6ULX 采用音频编码芯片 WM8904CGEFL/V，拓展出 1 个 3.5mm 耳机输出，1 个音频线性输入。WM8904CGEFL/V 的 I2S 端连接到了处理器的 SAI2 控制器,音频芯片 I2C 挂在 CPU I2C2 接口。

本节是测试播放音频。有两种方式，一种是 HMI2.0 应用直接播放，一种是通过 aplay 工具手动播放音频。

1) HMI2.0 音乐播放

HMI 多媒体音乐播放操作请参考 HMI 手册 《MEasy HMI2.0 开发手册》。



7-2 音频播放界面

2) 手动播放音乐

播放 wav 格式的音乐。

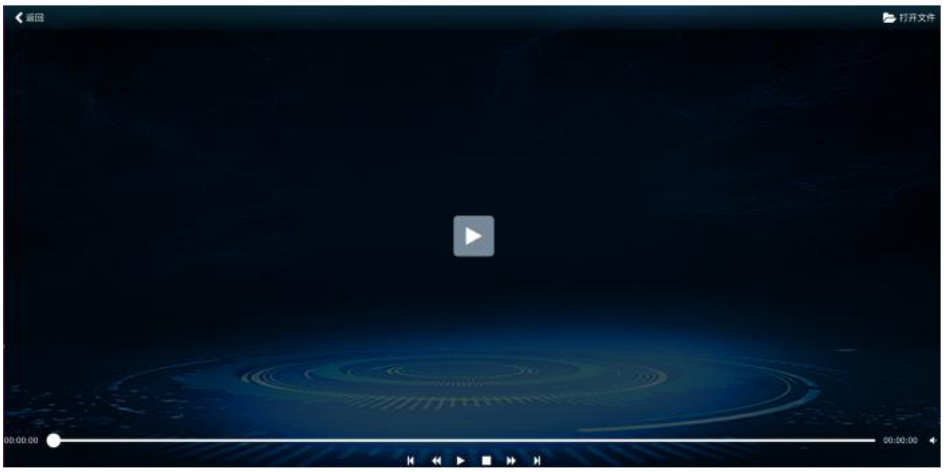
```
root@myir:~/audio# aplay  att16k.wav
Press 'k' to see a list of keyboard shortcuts.
Now playing /home/root/audio/att16k.wav
Redistribute latency...
0:00:14.9 / 0:00:14.9
Reached end of play list.
```

7.3. Video

本节是测试播放视频。这里直接使用 HMI2.0 应用直接播放演示。

1） HMI2.0 视频播放

HMI 多媒体视频播放请参考 HMI 手册 《MEasy HMI2.0 开发手册》。



7-3 视频播放界面

8. 系统工具

8.1. 压缩解压工具

本节主要测试系统的解压缩工具。压缩是可以把多个文件压缩成一个压缩包可以把多个文件压缩成一个压缩包，方便进行文件的传输。而解压可以把经过压缩的压缩文件还原成原始大小方便使用。本节将在文件系统以 tar、gzip、gunzip 等工具为例进行说明。

1）tar 工具

- 现在我们在 Linux 中使用的 tar 工具，它不仅可以对文件打包，还可以对其进行压缩，查看，添加以及解压等一系列操作。这里是将打包操作。输入以下命令查看 tar 语法格式：

```
root@myd-y6ull14x14:~# tar --help
Usage: tar [OPTION...] [FILE]...
GNU 'tar' saves many files together into a single tape or disk archive, and ca
n
restore individual files from the archive.

Examples:
  tar -cf archive.tar foo bar  # Create archive.tar from files foo and bar.
  tar -tvf archive.tar         # List all files in archive.tar verbosely.
  tar -xf archive.tar          # Extract all files from archive.tar.
```

详细参数说明如下：

- -c ：建立一个压缩文件的参数指令(create 的意思)；
- -x ：解开一个压缩文件的参数指令！
- -t ：查看 tarfile 里面的文件！特别注意，在参数的下达中， c/x/t 仅能存在一个！不可同一时候存在！由于不可能同一时候压缩与解压缩。
- -z ：是否同一时候具有 gzip 的属性？亦即是否须要用 gzip 压缩？
- -j ：是否同一时候具有 bzip2 的属性？亦即是否须要用 bzip2 压缩？
- -v ：压缩的过程中显示文件！这个经常使用，但不建议用在背景运行过程！

- -f ：使用档名，请留意，在 f 之后要马上接档名，不要再加参数！比如使用『tar -zcvfP tfile sfile』就是错误的写法，要写成『 tar -zcvPf tfile sfile』才对。
- -p ：使用原文件的原来属性（属性不会根据使用者而变）
- -P ：能够使用绝对路径来压缩！
- -N ：比后面接的日期(yyyy/mm/dd)还要新的才会被打包进新建的文件里！
- --exclude FILE：在压缩的过程中，不要将 FILE 打包！

- 新建 test.txt 文件，并输入以下命令将文件打包成.gz 格式：

```
root@myd-y6ull14x14:~# tar -czf test.tar.gz test.txt
root@myd-y6ull14x14:~# ls
OpenAMP_TTY_echo.elf  rs485_write  test.tar.gz  uart_test
rs485_read            test.c       test.txt     wifi.conf
```

所以上面加 z 参数，则以 .tar.gz 或 .tgz 来代表 gzip 压缩过的 tar file 。

- 把打包成 tar.gz 格式文件解压

```
root@myd-y6ull14x14:~# tar -xvf test.tar.gz
test.txt
root@myd-y6ull14x14:~# ls
OpenAMP_TTY_echo.elf  rs485_write  test.tar.gz  uart_test
rs485_read            test.c       test.txt     wifi.conf
```

4）gzip 压缩工具

- gzip 是在 Linux 系统中经常使用的一个对文件进行压缩和解压缩的命令，既方便又好用。在开发板终端输入以下命令查看 gzip 语法：

```
root@myd-y6ull14x14:~# gzip --help
Usage: gzip [OPTION]... [FILE]...
```

详细参数说明如下：

- 用 gzip 把文件压缩

```
root@myd-y6ull14x14:~# gzip test.txt
root@myd-y6ull14x14:~# ls
OpenAMP_TTY_echo.elf  rs485_write  test.tar.gz  uart_test
rs485_read            test.c       test.txt.gz  wifi.conf
```

5）gunzip 工具

用 gunzip 把文件解压

```
root@myd-y6ull14x14:~# gunzip test.txt.gz
root@myd-y6ull14x14:~# ls
OpenAMP_TTY_echo.elf  rs485_write  test.tar.gz  uart_test
rs485_read            test.c       test.txt     wifi.conf
```

8.2. 文件系统工具

主要测试系统的文件系统工具，本节将介绍几种常见的文件系统管理工具。系统自带的文件系统工具 mount、mkfs、fsck、dumpe2fs。

1）mount 挂载工具

- mount 是 Linux 下的一个命令，它可以将分区挂接到 Linux 的一个文件夹下，从而将分区和该目录联系起来，因此我们只要访问这个文件夹，就相当于访问该分区了，其应用语法格式如下：

```
root@myd-y6ull14x14:~# mount -h

Usage:
 mount [-lhV]
 mount -a [options]
 mount [options] [--source] <source> | [--target] <directory>
 mount [options] <source> <directory>
 mount <operation> <mountpoint> [<target>]

Mount a filesystem.
```

- 挂载 U 盘：

```
root@myd-y6ull14x14:~# mount /dev/sda1 /mnt/
```

2）mkfs 格式工具

- 硬盘分区后，下一步的工作是 Linux 文件系统的建立。类似于 Windows 下的格式化硬盘。在硬盘分区上建立文件系统会冲掉分区上的数据，而且不可恢复，因此在建立文件系统之前要确认分区上的数据不再使用。建立文件系统的命令是 mkfs,应用语法格式如下：

```
root@myd-y6ull14x14:~# mkfs -h

Usage:
 mkfs [options] [-t <type>] [fs-options] <device> [<size>]

Make a Linux filesystem.

Options:
```

```
 -t, --type=<type>   filesystem type; when unspecified, ext2 is used
     fs-options      parameters for the real filesystem builder
     <device>        path to the device to be used
     <size>          number of blocks to be used on the device
 -V, --verbose       explain what is being done;
                       specifying -V more than once will cause a dry-run
 -h, --help          display this help
 -V, --version       display version

For more details see mkfs(8).
```

- 格式化 U 盘

```
root@myd-y6ull14x14:~# umount /mnt
root@myd-y6ull14x14:~# umount /run/media/sda1/

root@myd-y6ull14x14:~# mkfs -t ext3 -V -c /dev/sda1
mkfs from util-linux 2.34
mkfs.ext3 -c /dev/sda1
mke2fs 1.45.3 (14-Jul-2019)
/dev/sda1 contains a vfat file system
Proceed anyway? (y,N) y
Creating filesystem with 3890688 4k blocks and 972944 inodes
Filesystem UUID: 97810d2b-76aa-44a4-9409-2c70de71eca0
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 265
4208

Checking for bad blocks (read-only test):
[ 6873.703223] audit: type=1006 audit(1599269401.268:4): pid=947 uid=0 old-
auid=4294967295 auid=0 tty=(none) old-ses=4294967295 ses=3 res=1
done
Allocating group tables: done
Writing inode tables:
done
```

```
Creating journal (16384 blocks):
done
Writing superblocks and filesystem accounting information: done
```

3）fsck 文件修复工具

- fsck 命令主要用于检查文件系统的正确性，当文件系统发生错误时，可用 fsck 指令尝试加以修复。并对 Linux 磁盘进行修复。例如：

```
root@myd-y6ull14x14:~# fsck -a /dev/sda1
fsck from util-linux 2.34
/dev/sda1: clean, 11/972944 files, 86964/3890688 blocks
```

4）dumpe2fs

- 打印特定设备上现存的文件系统的超级块(super block)和块群(blocks group)的信息。开发板输入以下命令查看应用语法：

```
root@myd-y6ull14x14:~# dumpe2fs -h
dumpe2fs 1.45.3 (14-Jul-2019)
Usage: dumpe2fs [-bfghimxV] [-o superblock=<num>] [-o blocksize=<num>]
device
```

- 查看文件系统格式化后的详细属性，例如，输入命令查看某个磁盘的详细信息：

```
root@myd-y6ull14x14:~# dumpe2fs -h
dumpe2fs 1.45.3 (14-Jul-2019)
Usage: dumpe2fs [-bfghimxV] [-o superblock=<num>] [-o blocksize=<num>]
device
root@myd-y6ull14x14:~#
root@myd-y6ull14x14:~# root@myd-y6ull14x14:~# dumpe2fs -h^C
root@myd-y6ull14x14:~# dumpe2fs /dev/sda1
dumpe2fs 1.45.3 (14-Jul-2019)
Filesystem volume name:   <none>
Last mounted on:          <not available>
Filesystem UUID:          97810d2b-76aa-44a4-9409-2c70de71eca0
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_index filetype sp
arse_super large_file
Filesystem flags:         unsigned_directory_hash
```

- 查看某磁盘的 inode 数量，inode 也会消耗硬盘空间，所以硬盘格式化的时候，操作系统自动将硬盘分成两个区域。一个是数据区，存放文件数据；另一个是 inode 区（inode table），存放 inode 所包含的信息。

```
root@myd-y6ull14x14:~# dumpe2fs /dev/sda1 | grep -i "inode size"
dumpe2fs 1.45.3 (14-Jul-2019)
Inode size:                256
```

- 查看某磁盘的 block 数量，操作系统读取硬盘的时候，不会一个个扇区地读取，这样效率太低，而是一次性连续读取多个扇区，即一次性读取一个"块"（block）。这种由多个扇区组成的"块"，是文件存取的最小单位。

```
root@myd-y6ull14x14:~# dumpe2fs /dev/sda1 | grep -i "block size"
dumpe2fs 1.45.3 (14-Jul-2019)
Block size:                4096
```

8.3. 磁盘管理工具

主要测试系统的磁盘管理工具，本节将介绍几种常见的磁盘管理工具。系统自带的磁盘管理工具 fdisk、dd、mkfs、du、df、cfdisk、fsck 。通过这些命令可以监控平时的磁盘使用情况。

1）fdisk 磁盘分区工具

- fdisk 磁盘分区工具在 DOS、Windows 和 Linux 中都有相应的应用程序。在 Linux 系统中，fdisk 是基于菜单的命令。用 fdisk 对硬盘进行分区，可以在 fdisk 命令后面直接加上要分区的硬盘作为参数，其应用语法格式如下：

```
root@myd-y6ull14x14:~# fdisk -h
Usage:
 fdisk [options] <disk>         change partition table
 fdisk [options] -l [<disk>] list partition table(s)
```

- 对 eMMC 进行分区：

```
root@myd-y6ull14x14:~# fdisk /dev/mmcblk2p2

Welcome to fdisk (util-linux 2.34).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

The old ext4 signature will be removed by a write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0x43403b02.

Command (m for help):
```

2）dd 拷贝命令

- dd 命令用于将指定的输入文件拷贝到指定的输出文件上。并且在复制过程中可以进行格式转换。dd 命令与 cp 命令的区别在于：dd 命令可以在没有创建文件系统的软盘上进行，拷贝到软盘的数据实际上是镜像文件。类似于 DOS 中的 diskcopy 命令的作用。dd 命令的格式为： dd [<if=输入文件名/设备名>] [<of=输出文件名/设备名>] [bs=块字节大小] [count = 块数]

- 创建一个大小为 2M 的文件

```
root@myd-y6ull14x14:~# time dd  if=/dev/zero of=ffmpeg1 bs=2M count=1
conv=fsync
1+0 records in
1+0 records out
2097152 bytes (2.1 MB, 2.0 MiB) copied, 0.119542 s, 17.5 MB/s

real    0m0.129s
user    0m0.000s
sys     0m0.019s
```

3）du 磁盘用量统计工具:

- du 命令用于显示磁盘空间的使用情况。该命令逐级显示指定目录的每一级子目录占用文件系统数据块的情况。du 一般使用语法如下：

```
root@myd-y6ull14x14:~# du --help
Usage: du [OPTION]... [FILE]...
  or:  du [OPTION]... --files0-from=F
Summarize disk usage of the set of FILEs, recursively for directories.
```

部分参数说明：

  - -a:显示所有目录或文件的大小
  - -h:以 K,M,G 为单位，提高信息可读性
  - -k:以 KB 为单位输出
  - -m:以 MB 为单位输出

- 统计 dd 命令生成的文件大小：

```
root@myd-y6ull14x14:~# du ffmpeg1
2048    ffmpeg1
root@myd-y6ull14x14:~# du -h ffmpeg1
2.0M    ffmpeg1
```

4）df 磁盘统计工具:

- 用于显示目前在 Linux 系统上的文件系统的磁盘使用情况统计，一般用法如下：

```
root@myd-y6ull14x14:~# df --help
Usage: df [OPTION]... [FILE]...
Show information about the file system on which each FILE resides,
```

```
or all file systems by default.
```

部分参数说明：

- -h：可以根据所使用大小使用适当的单位显示
- -i:查看分区下 inode 的数量和 inode 的使用情况
- -T:打印出文件系统类型

- 查看分区下 inode 的数量和 inode 的使用情况，使用如下命令：

```
root@myd-y6ull14x14:~# df -i
Filesystem      Inodes IUsed  IFree IUse% Mounted on
/dev/root       497488 48320 449168   10% /
devtmpfs        166144   489 165655    1% /dev
tmpfs           248792     1 248791    1% /dev/shm
tmpfs           248792   706 248086    1% /run
tmpfs           248792    13 248779    1% /sys/fs/cgroup
tmpfs           248792    16 248776    1% /tmp
tmpfs           248792    28 248764    1% /var/volatile
tmpfs           248792     9 248783    1% /run/user/0
/dev/mmcblk1p1       0     0      0     - /run/media/mmcblk1p1
/dev/mmcblk2p1       0     0      0     - /run/media/mmcblk2p1
/dev/mmcblk2p2  546176 41910 504266     8% /run/media/mmcblk2p2
```

inode 是我们在格式化的时候系统给我们划分好的 ,inode 与磁盘分区大小有关。当我们的 inode 使用已经达到百分百时，即使我们的磁盘空间还是有剩余，我们也是写不了数据到磁盘的。其他应用例子请参考 2.5 节。

8.4. 进程管理工具

进程也是操作系统中的一个重要概念，它是一个程序的一次执行过程，程序是进程的一种静态描述，系统中运行的每一个程序都是在它的进程中运行的。Linux 系统中所有的进程是相互联系的，除了初始化进程外，所有进程都有一个父进程。新的进程不是被创建，而是被复制，或是从以前的进程复制而来。Linux 中所有的进程都是由一个进程号为 1 的 init 进程衍生而来的。Linux 系统包括 3 种不同类型的进程，每种进程都有自己的特点和属性：

- 交互进程：由一个 Shell 启动的进程，既可以在前台运行，又可以在后台运行。
- 批处理进程：这种进程和终端没有联系，是一个进程序列。这种进程被提交到等待队列顺序执行的进程。
- 监控进程(守护进程)：守护进程总是活跃的，一般是在后台运行，守护进程一般是由系统在开始时通过脚本自动激活启动或 root 启动

对于 linux 系统来说，进程的管理是重要的一环，对于进程的管理通常是通过进程管理工具实现的，Linux 系统中比较常用的进程管理命令有以下几种： ps top vmstat kill 。

1）ps 显示当前进程工具

- 显示当前系统进程的运行情况，一般语法如下：

```
root@myd-y6ull14x14:~# ps --help

Usage:
 ps [options]

 Try 'ps --help <simple|list|output|threads|misc|all>'
  or 'ps --help <s|l|o|t|m|a>'
 for additional help text.

For more details see ps(1).
```

部分参数组合说明：

- -u：以用户为中心组织进程状态信息显示；
- -a：与终端无关的进程；
- -x：与终端有关的进程； （线程，就是轻量级进程；）

通常上面命令组合使用：aux。

- --e：显示所有进程；相当于 ax；
- -f：显示完整格式程序信息；

通常以上命令组合使用：ef

- -H：以进程层级显示进程数的
- -F：显示更多的程序信息

通常组合使用命令：eHF

- 显示所有进程的信息情况

```
root@myd-y6ull14x14:~# ps -aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.3  92272  7176 ?        Ss   Sep04   0:03 /sbin/init
root           2  0.0  0.0      0     0 ?        S    Sep04   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   Sep04   0:00 [rcu_gp]
root           4  0.0  0.0      0     0 ?        I<   Sep04   0:00 [rcu_par_gp]
root           8  0.0  0.0      0     0 ?        I<   Sep04   0:00 [mm_percpu_wq]
root           9  0.0  0.0      0     0 ?        S    Sep04   0:00 [ksoftirqd/0]
root          10  0.0  0.0      0     0 ?        I    Sep04   0:00 [rcu_preempt]
root          11  0.0  0.0      0     0 ?        S    Sep04   0:00 [migration/0]
root          12  0.0  0.0      0     0 ?        S    Sep04   0:00 [cpuhp/0]
root          13  0.0  0.0      0     0 ?        S    Sep04   0:00 [cpuhp/1]
root          14  0.0  0.0      0     0 ?        S    Sep04   0:00 [migration/1]
```

对其上面一些任务栏进行简单解释说明

- VSZ：vittual memory size 虚拟内存集
- RSS：resident size，常驻内存集
- STAT：进程状态有一下几个
  - R：runing
  - S：interuptable sleeping 可中断睡眠

- D：uninteruptable sleeping 不可中断睡眠
- T：stopped
- Z: zombie 僵尸进程
- +:前台进程
- N:低优先级进程
- l:多线程进程

2）top 显示 linux 进程

- top 命令将相当多的系统整体性能信息放在一个屏幕上。显示内容还能以交互的方式进行改变。动态的持续监控进程的运行状态，top 语法一般如下：

```
root@myd-y6ull14x14:~#  top -help
  procps-ng 3.3.15
Usage:
  top -hv | -bcEHiOSs1 -d secs -n max -u|U user -p pid(s) -o field -w [cols]
```

- 动态查看系统进程

```
root@myd-y6ull14x14:~# top
top - 02:03:28 up  2:28,  1 user,  load average: 0.01, 0.35, 0.99
Tasks: 135 total,   1 running, 134 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.4 us,  1.4 sy,  0.0 ni, 97.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   1943.7 total,   1487.8 free,    244.2 used,    211.6 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used.   1597.4 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME
+ COMMAND

    982 root      20   0    3444   2024   1616 R   5.9   0.1   0:00.02 top


      1 root      20   0   92272   7176   5064 S   0.0   0.4   0:03.49 syste
md
```

```
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.03 kthrea
dd

      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
```

3）vmstat 虚拟内存的统计工具

- 这个命令可查看内存空间的使用状态，能获取整个系统的性能粗略信息，vmstat 运行于两种模式：采样模式和平均模式。如果不指定参数，则 vmstat 统计运行于平均模式下，vmstat 显示从系统启动以来所有统计数据的均值。常用语法以及参数如下：

```
root@myd-y6ull14x14:~# vmstat  -h

Usage:
 vmstat [options] [delay [count]]

Options:
 -a, --active           active/inactive memory
 -f, --forks            number of forks since boot
 -m, --slabs            slabinfo
 -n, --one-header       do not redisplay header
 -s, --stats            event counter statistics
 -d, --disk             disk statistics
 -D, --disk-sum         summarize disk statistics
 -p, --partition <dev>  partition specific statistics
 -S, --unit <char>      define display unit
 -w, --wide             wide output
 -t, --timestamp        show timestamp

 -h, --help     display this help and exit
 -V, --version  output version information and exit

For more details see vmstat(8).
```

- vmstat 运行于平均模式，显示从系统启动以来所有统计数据的均值：

```
root@myd-y6ull14x14:~# vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b    swpd    free    buff  cache   si   so    bi    bo   in   cs us sy id wa
 st
 0  0       0 1524140   12552 204240    0    0   441    13   80  103  0  0
 95  5  0
```

对其上面一些任务栏进行简单解释说明

- r：当前可运行的进程数。这些进程没有等待 I/O,而是已经准备号运行。理想状态，可运行进程数与可用 cpu 数量相等
- b：等待 I/O 完成的被阻塞进程数
- forks：创建新进程的次数
- in：系统发生中断的次数
- cs：系统发生上下文切换的次数
- us：用户进程消耗的总 CPU 时间百分比
- sy：系统代码消耗的总 CPU 时间的百分比，其中包括消耗在 system、irq 和 softirq 状态的时间
- wa：等待 I/O 消耗的总 CPU 的百分比
- id：系统空闲消耗的总 CPU 时间的百分比

- 统计系统各种数据详细信息

```
root@myd-y6ull14x14:~# vmstat  -s
        1990336 K total memory
         249516 K used memory
         103972 K active memory
         132832 K inactive memory
        1524012 K free memory
          12568 K buffer memory
         204240 K swap cache
              0 K total swap
              0 K used swap
              0 K free swap
          13051 non-nice user cpu ticks
              1 nice user cpu ticks
           4360 system cpu ticks
```

```
      3403234 idle cpu ticks
       166345 IO-wait cpu ticks
         1782 IRQ cpu ticks
         1637 softirq cpu ticks
            0 stolen cpu ticks
     15733523 pages paged in
       472350 pages paged out
            0 pages swapped in
            0 pages swapped out
      2847443 interrupts
      3669954 CPU context switches
   1599262527 boot time
          990 forks
```

对其上面一些任务栏进行简单解释说明

- total memory：系统总内存
- used memory：已使用内存
- CPU ticks：显示的是自系统启动的 CPU 时间，这里的“tick”是一个时间单位
- forks：大体上表示的是从系统启动开始已经创建的新进程的数量

vmstat 提供了关于 linux 系统性能的众多信息。在调查系统问题时，它是核心工具之一。

1） kill 进程终止工具

- 发送指定的信号到相应进程。不指定型号将发送 SIGTERM（15）终止指定进程。如果任无法终止该程序可用“-KILL” 参数，其发送的信号为 SIGKILL(9) ，将强制结束进程，使用 ps 命令或者 jobs 命令可以查看进程号。root 用户将影响用户的进程，非 root 用户只能影响自己的进程。kill 命令一般语法如下：

```
kill [ -s signal | -p ] [ -a ] pid ...
kill -l [ signal ]
```

部分参数组合说明：

- -s：指定发送的信号
- -p：模拟发送信号
- -l：指定信号的名称列表
- pid：要中止进程的 ID 号
- Signal：表示信号

- 首先使用 ps -ef 与管道命令确定要杀死进程的 PID，

```
root@myd-y6ull14x14:~# ps -ef | grep wpa_supplicant
root         564     1  0 Sep04 ?        00:00:00 /usr/sbin/wpa_supplicant
 -u
root         989   850  0 02:05 ttymxc1  00:00:00 grep wpa_supplicant
```

- 然后输入以下命令终止进程：

```
root@myd-y6ull14x14:~# kill 564
```

- killall 命令终止同一进程组内的所有进程，允许指定要终止的进程的名称而非 PID 进程号

```
root@myd-y6ull14x14:~# killall wpa_supplicant
root@myd-y6ull14x14:~#
```

9. 开发支持

本章主要介绍针对当前 SDK 进行二次开发的一些基本信息，当前 SDK 提供两种配置的参考镜像，一种是 myir-image-core，主要针对无 GUI 的应用；另外一种是 myir-image-full，在 myir-image-core 的基础上增加一些需要 GUI 的应用，如 QT 运行时库以及参考的 HMI 界面。关于这两种镜像的信息请参考《MYD-Y6ULX SDK 发布说明》。

9.1. 开发语言

9.1.1. SHELL

Shell 是一个用 C 语言编写的程序，它是用户使用 Linux 的桥梁。Shell 既是一种命令语言，又是一种程序设计语言。常见的 Linux 的 Shell 种类众多，常见的有：

- Bourne Shell（/usr/bin/sh 或/bin/sh）
- Bourne Again Shell（/bin/bash）
- C Shell（/usr/bin/csh）
- K Shell（/usr/bin/ksh）
- Shell for Root（/sbin/sh）

MYD-Y6ULX 支持 bourne shell 和 Bourne Again Shell 2 种

```
root@myd-y6ull14x14:~# echo "echo 'myir test'" > shell_demo.sh
root@myd-y6ull14x14:~# sh shell_demo.sh
myir test
root@myd-y6ull14x14:~# bash shell_demo.sh
myir test
```

9.1.2. C/C++

C/C++是 Linux 平台下进行底层应用开发最为常用的编程语言，也是仅次于汇编的最为高效的语言。使用 C/C++进行开发通常采用的是交叉开发的方式，即在开发主机端进行开发，编译生成目标机器上运行的二进制执行文件，然后部署到目标机器上运行。

采用这种方式，首先需要安装基于 Yocto 构建的 SDK，安装步骤请参《MYD-Y6ULX Yocto 软件开发指南》

本节通过编写一个简单的 Hello World 实例来演示应用程序的开发，以下为在开发主机端编写的演示程序 hello.c：

```
1 #include<stdio.h>
2 int main(int argc,char *argv[])
3 {
4       printf("hello world!\n");
5       return 0;
6 }
```

接着编译应用程序，这里用$CC 编译，因为编译的时候需要对应的头文件和链接，$CC 包含有对应的系统库和配置信息，如果直接用 arm-poky-linux-gnueabi-gcc 来编译会出现找不到头文件的情况，这个时候可以加入参数-v 来查看详细的链接过程。

```
myir@myir-server1:~/test$ source /home/alex/workspace/imx6ul_tools_5.4/tools_qt5/environment-setup-cortexa7t2hf-neon-poky-linux-gnueabi

myir@myir-server1:~/test$ cat main.c
#include <stdio.h>


int main(int argc, char *argv[])
{
        print("myir test!\n");
        return 0;
}
myir@myir-server1:~/test$ $CC main.c -o main
```

然后通过 scp 命令把生成的执行文件拷贝到目标机器上执行，结果如下：

```
root@myd-y6ull14x14:~# ./main
myir test!
```

9.1.3. Python

Python 是一种解释型、面向对象、动态数据类型的高级程序设计语言。Python 由 Guido van Rossum 于 1989 年底发明，第一个公开发行版发行于 1991 年。像 Perl

语言一样, Python 源代码同样遵循 GPL(GNU General Public License) 协议。本节主要测试 python 的使用，从 python 命令行和脚本两个方面来说明。

1） 查看系统支持的 phthon 版本

```
root@myd-y6ull14x14:~# ls /usr/bin/python*
/usr/bin/python3      /usr/bin/python3.7m
/usr/bin/python3.7    /usr/bin/python3.7m-config-lib
```

2） python 命令行测试

启动 python，并在 python 提示符中输入以下文本信息，然后按 Enter 键查看运行效果：

```
root@myd-y6ull14x14:~# python3
Python 3.7.5 (default, Sep  5 2020, 00:13:46)
[GCC 9.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("myir test")
```

在 Python 3.7.5 版本中,以上实例输出结果如下：

```
myir test
>>>
```

退出 Python 命令行，执行 exit()即退出 Python：

```
>>> exit()
root@myd-y6ull14x14:~#
```

3） 编写脚本测试 Python

编写一个简单的 Python 脚本程序，所有 Python 文件将以 .py 为扩展名。

```
root@myd-y6ull14x14:~# vi test.py
root@myd-y6ull14x14:~# cat test.py
#!/usr/bin/env python3
print("myir test")
```

执行脚本文件，解释器在/usr/bin/env 中以 python3 运行目录中，使用以下命令执行脚本。

```
root@myd-y6ull14x14:~# chmod a+x test.py
root@myd-y6ull14x14:~# ./test.py
myir test
```

通过脚本参数调用 Python3 解释器开始执行脚本，直到脚本执行完毕。当脚本执行完成后，解释器不再有效。

9.2. 数据库

数据库(Database)是按照数据结构来组织、存储和管理数据的仓库。数据库有很多种类型，常用的数据库有 Access、Oracle、Mysql、SQL Server、SQLite 等。

9.2.1. System SQLlite

SQLite 是一个嵌入式 SQL 数据库引擎。与大多数其他 SQL 数据库不同，SQLite 没有单独的服务器进程。SQLite 直接读写普通磁盘文件。包含多个表、索引、触发器和视图的完整 SQL 数据库包含在单个磁盘文件中。这是一款轻型的数据库，是遵守 ACID 的关联式数据库管理系统，它的设计目标是嵌入式的，而且目前已经在很多嵌入式产品中使用了它，它占用资源非常的低，在嵌入式设备中，可能只需要几百 K 的内存就够了。这款数据库的运行处理速度比 Mysql、PostgreSQL 这两款都要快

1）SQLlite 创建数据库

启动 sqlite3，并创建一个新的数据库 <testDB.db>,在终端界面输入以下命令就可以进入操作界面。

```
root@myd-y6ull14x14:~# sqlite3 testDB.db
SQLite version 3.31.1 2020-01-27 19:55:54
Enter ".help" for usage hints.
sqlite>
```

上面的命令将在当前目录下创建一个文件 testDB.db。该文件将被 SQLite 引擎用作数据库。注意到 sqlite3 命令在成功创建数据库文件之后，将提供一个 sqlite> 提示符。

数据库被创建，您就可以使用 SQLite 的 .databases 命令来检查它是否在数据库列表中，如下所示：

```
sqlite> .databases
main: /home/root/testDB.db
sqlite>
```

使用 .quit 命令退出 sqlite 提示符，如下所示：

```
sqlite> .quit
root@myd-y6ull14x14:~#
```

如果想要详细了解 SQLlite 相关信息，请参考官网 https://www.sqlite.org/docs.html。

9.3. 本地化

本节讨论本地化相关的设置和测试。本地化，指的是一个程序或软件在支持国际化的基础上，给定程序特定区域的语言信息使其在信息的输入输出等处理上适应特定区域人群的使用。这里允许程序所使用的一些语言环境变量在程序执行时动态配置。本章主要以米尔演示镜像 MEasy HMI 2.0 为例说明。

9.3.1. 多语言

本节主要以米尔演示镜像 MEasy HMI 2.0 为例说明多语言在 QT 项目中的实际应用。

阅读下面内容前请参考《MEasy HMI 2.0 开发手册》第 3.1 章 环境搭建，搭建起 MEasy HMI 的编译环境。

1） 打开 mxapp2 工程

MEasy HMI 2.0 对应的工程源码为 mxapp2.tar.gz,拷贝至上述文档所构建的环境里面去，并使用 QT Creator 打开这个工程。

2） 生成 ts 文件

通过终端进入 mxapp2 工程所在源码目录，并执行下面命令生成翻译文件。

```
qinlh@qinlh-VirtualBox:~/download/mxapp2$ lupdate mxapp2.pro
Info: creating stash file /home/qinlh/download/MXAPP/.qmake.stash
Updating 'languages/language_zh.ts'...
    Found 202 source text(s) (0 new and 202 already existing)
Updating 'languages/language_en.ts'...
Found 202 source text(s) (0 new and 202 already existing)
```

工程在发布之前已完成翻译的工作，此处不会重新生成 language_zh.ts 和 language_en.ts 文件，中文显示使用的是 language_zh.ts 生成的 qm 文件，英文显示使用的是 language_en.ts 生成的 qm 文件。

3） 翻译文本

打开 language_en.ts 文件，针对翻译的源字符串为<source>节点里面的内容，翻译的目标字符串为<translation>里面的内容，用户可根据需求进行修改。

```
<message>
        <location filename="../Album.qml" line="50"/>
        <source>返回</source>
```

```
        <translation type="unfinished">Return</translation>
</message>
```

4) 生成 qm 文件

Ts 文件修改完成后需要手动生成翻译模版文件供应用使用，使用如下命令即可生成翻译模版文件。

```
qinlh@qinlh-VirtualBox:~/download/mxapp2/languages$ lrelease language_en.ts
 -qm language_en.qm
Updating 'language_en.qm'...
    Generated 183 translation(s) (2 finished and 181 unfinished)
Ignored 21 untranslated source text(s)
qinlh@qinlh-VirtualBox:~/download/mxapp2/languages$ lrelease language_zh.ts
 -qm language_zh.qm
Updating 'language_en.qm'...
    Generated 183 translation(s) (2 finished and 181 unfinished)
    Ignored 21 untranslated source text(s)
```

5) 应用翻译文件

翻译文件的使用是需要通过应用的代码来调用的，具体调用方法参考 translator.cpp 文件里面的 loadLanguage 成员函数。

9.3.2. 字体

本节主要以米尔演示镜像 MEasy HMI 2.0 为例说明字体在 QT 项目中的实际应用。阅读下面内容前请参考《MEasy HMI 2.0 开发手册》第 3.1 章 环境搭建，搭建起 MEasy HMI 的编译环境。

1）安装字体文件

字体文件可以直接放置到开发板文件系统/usr/lib/fonts/目录：

```
# ls /usr/lib/fonts/msyh.ttc
/usr/lib/fonts/msyh.ttc
```

或者直接添加的 QT 工程里面。

```
qinlh@qinlh-VirtualBox:~/download/mxapp2/fonts$ tree
├── DIGITAL
│   ├── DIGITAL.TXT
```

```
|   └── DS-DIGIB.TTF
└── fontawesome-webfont.ttf
```

2）使用字体文件

字体文件的使用是需要通过应用的代码来调用的，具体调用方法参考 main.cpp.中 iconFontInit()函数。

```
void iconFontInit()
{
    int fontId_digital = QFontDatabase::addApplicationFont(":/fonts/DIGITAL/DS-DIGIB.TTF");
    int fontId_fws = QFontDatabase::addApplicationFont(":/fonts/fontawesome-webfont.ttf");
    QString fontName_fws = QFontDatabase::applicationFontFamilies(fontId_fws).at(0);
    QFont iconFont_fws;
    iconFont_fws.setFamily(fontName_fws);
    QApplication::setFont(iconFont_fws);
    iconFont_fws.setPixelSize(20);
}
```

9.3.3. 软键盘

本章节主要以米尔演示镜像 MEasy HMI 2.0 为例说明软键盘在 QT 项目中的实际应用。

阅读下面内容前请参考《MEasy HMI 2.0 开发手册》第 3.1 章 环境搭建，搭建起 MEasy HMI 的编译环境。

QT 从 5.7 版本开始在官方源码里面加入了 qt vritual keyboard 组件，myir 在发布的 MEasy HMI 2.0 配套的镜像中已经使用了 QT 5.13 版本,并配置了 vritual keyboard 这个组件。

1) 软键盘嵌入到 qml 代码

Qt 提供的软键盘只能在 qml 代码里面进行调用，调用前需要定义软键盘弹出的位置及软键盘的大小，如下面代码所示。

```
InputPanel {
id: inputPanel
```

```
x: adaptive_width/8
y: adaptive_height/1.06
z:99
anchors.left: parent.left
anchors.right: parent.right

states: State {
name: "visible"
when: inputPanel.active
PropertyChanges {
target: inputPanel
        y: adaptive_height/1.06 - inputPanel.height
}
}
```

2) 触发软键盘

软键盘的出发是通过 QML 的 TextField、TextEdit 组件触发的，用户只需要在 UI 组件中加入此组件即可在 UI 上调出软键盘并使用。如果是 qt 系统，可以使用 QT 自带的 QT virtualkeyboard 输入。Weston 没安装对应的输入法，当插入键盘时只能输入拼音或中文。

```
TextField{
id: netmask_input
InputMethodHints: Qt.ImhFormattedNumbersOnly
onAccepted: digitsField.focus = true
font.family: "Microsoft YaHei"
color: "white"
}
```

3) 使用软键盘

软键盘的使用参考《MEasy HMI 2.0 开发手册》第 2.6 章系统设置 UI 界面进行设置操作就会弹出软键盘。

10. 参考资料

- **Yoto 项目 BSP 开发指南**
  https://www.yoctoproject.org/docs/3.1.1/bsp-guide/bsp-guide.html
- **Yocto 项目 Linux 内核开发手册**
  https://www.yoctoproject.org/docs/3.1.1/kernel-dev/kernel-dev.html
- **Linux 内核看门狗介绍**
  https://www.kernel.org/doc/html/latest/watchdog/index.html
- **嵌入式 Linux 下的 Qt**
  https://doc.qt.io/qt-5/embedded-linux.html
- **Wayland 架构**
  https://wayland.freedesktop.org/architecture.html
- **Systemd 网络配置**
  https://www.freedesktop.org/software/systemd/man/systemd.network.html

附录一 联系我们

深圳总部

负责区域：广东 / 四川 / 重庆 / 湖南 / 广西 / 云南 / 贵州 / 海南 / 香港 / 澳门

电话：0755-25622735 0755-22929657

传真：0755-25532724

邮编：518020

地址：深圳市龙岗区坂田街道发达路云里智能园 2 栋 6 楼 04 室

上海办事处

负责区域：上海 / 湖北 / 江苏 / 浙江 / 安徽 / 福建 / 江西

电话：021-60317628 15901764611

传真：021-60317630

邮编：200062

地址：上海市普陀区中江路 106 号北岸长风 I 座 302

北京办事处

负责区域：北京/天津/陕西/辽宁/山东/河南/河北/黑龙江/吉林/山西/甘肃/内蒙古/宁夏

电话：010-84675491 13269791724

传真：010-84675491

邮编：102218

地址：北京市昌平区东小口镇中滩村润枫欣尚 2 号楼 1009

销售联系方式

网址：www.myir-tech.com

邮箱：sales.cn@myirtech.com

技术支持联系方式

电话：027-59621648

邮箱：support.cn@myirtech.com

如果您通过邮件获取帮助时，请使用以下格式书写邮件标题：

[公司名称/个人--开发板型号] 问题概述

这样可以使我们更快速跟进您的问题，以便相应开发组可以处理您的问题。

附录二 售后服务与技术支持

凡是通过米尔科技直接购买或经米尔科技授权的正规代理商处购买的米尔科技全系列产品，均可享受以下权益：

1、6 个免免费保修服务周期
2、终身免费技术支持服务
3、终身维修服务
4、免费享有所购买产品配套的软件升级服务
5、免费享有所购买产品配套的软件源代码，以及米尔科技开发的部分软件源代码
6、可直接从米尔科技购买主要芯片样品，简单、方便、快速；免去从代理商处购买时，漫长的等待周期
7、自购买之日起，即成为米尔科技永久客户，享有再次购买米尔科技任何一款软硬件产品的优惠政策
8、OEM/ODM 服务

如有以下情况之一，则不享有免费保修服务：

1、超过免费保修服务周期
2、无无序列号或无产品有效购买单据
3、进液、受潮、发霉或腐蚀
4、受撞击、挤压、摔落、刮伤等非产品本身质量问题引起的故障和损坏
5、擅自改造硬件、错误上电、错误操作造成的故障和损坏
6、由不可抗拒自然因素引起的故障和损坏

产品返修：

用户在使用过程中由于产品故障、损坏或其他异常现象，在寄回维修之前，请先致电米尔科技客服部，与工程师进行沟通以确认问题，避免故障判断错误造成不必要的运费损失及周期的耽误。

维修周期：

收到返修产品后，我们将即日安排工程师进行检测，我们将在最短的时间内维修或更换并寄回。一般的故障维修周期为 3 个工作日（自我司收到物品之日起，不计运输过程时间），由于特殊故障导致无法短期内维修的产品，我们会与客户另行沟通并确认维修周期。

维修费用：

在免费保修期内的产品，由于产品质量问题引起的故障，不收任何维修费用；不属于免费保修范围内的故障或损坏，在检测确认问题后，我们将与客户沟通并确认维修费用，我们仅收取元器件

材料费，不收取维修服务费；超过保修期限的产品，根据实际损坏的程度来确定收取的元器件材料费和维修服务费。

运输费用：

产品正常保修时，用户寄回的运费由用户承担，维修后寄回给用户的费用由我司承担。非正常保修产品来回运费均由用户承担。