MYS-ZU5EV Linux 软件评估指南

文件状态： [] 草稿 [√] 正式发布	**文件标识：**	MYIR-MYS-ZU5EV-SW-EG-ZH-L5.4.0
	当前版本：	V1.10
	作　　者：	Fengyong
	创建日期：	2021-05-25
	最近更新：	2021-05-25

版 本 历 史

版本	作者	参与者	日期	备注
V1.10	Fengyong		20210525	初始版本，适用于 MYS-ZU5EV

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

MYS-ZU5EV Board 是深圳市米尔电子有限公司推出的一款以 Xilinx ZynqMP ZU5EV 作为主处理器的嵌入式开发板。MYS-ZU5EV Board 采用 Xilinx 的 zynqMP SoC 平台，将 ARM 处理器和 FPGA 架构紧密集成。该产品拥有 4 个 ARM Cortex A53 cpu 核和 2 个 ARM Cortex R5 核的高性能特性，在设计中能更好的满足各种工业需要。

此外 MYIR 提供了丰富的软件资源以及文档资料。用户在评估测试过程中会用到一些配件，参见下面的列表。

表 1-1. 选配模块

配件	接口方式	说明及链接
摄像头	USB 接口	MY-CAM002U http://www.myir-tech.com/product/my_cam002u.htm

1.2. 软件资源

MYS-ZU5EV 开发板提供的开发方式：基于 Petalinux 开发，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《MYS-ZU5EV SDK 发布说明》。

按照《MYS-ZU5EV_Linux 软件开发指南》中“4.1.制作 SD 卡启动器”章节的方法烧录 mys-zu5ev-full 镜像到 tf 卡，使用 tf 卡启动模式，您只需要上电即可使用。说明：（1）关于 vcu 的测试请参考《MYS-ZU5EV_Linux 软件开发指南》中的“8.4.8 VCU 应用示例”章节的“VCU 测试”的方法测试；（2）关于 mipi 摄像头的测试请参考《MYS-ZU5EV_Linux 软件开发指南》中的“8.4.7 MIPI 摄像头应用示例”章节的方法测试。

1.3. 文档资源

根据用户使用开发板的各个不同阶段，SDK 中包含了各阶段的文档，发布说明，评估指南，开发指南等不同类别的文档和手册。具体的文档列表参见《MYS-ZU5EV 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基础环境，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《**MYS-ZU5EV-QSG**》文档。

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，eMMC，RTC 等核心资源的参数进行读取与测试。

2.1. CPU

ZynqMP-zu5ev 是将 4 个 ARM Cortex A53 cpu 核和 2 个 ARM Cortex R5 核结合一个可编程的 FPGA 芯片集成到一个片上系统中。

通常将 ARM 处理器和各种存储外设资源称为处理系统（Processor System，PS），将 FPGA 部分称为可编程逻辑（Programmable Logic，PL）。ARM Cortex-A53 是一个应用级处理器，支持类似 Linux 操作系统的运行。FPGA 采用的 Xilinx7 架构实现了工业标准的 AXI 接口，该接口在 ARM 和 FPGA 之间形成高效耦合，减少了分立芯片产生的额外功耗，不仅实现了高带宽、低延迟的连接，同时带来了物理尺寸和生产成本的降低。

ZynqMP 的 PL 部分用来实现高速逻辑运算和并行数据流处理的子系统是非常理想的，PS 部分支持软件控制或者操作系统。这意味着基于此平台，大多数系统的设计都可以根据功能进行软件和硬件的划分，使得 PS 和 PL 都可以发挥各自的优势，从而让整个系统呈现出最佳性能：

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@mys_zu5ev # cat /proc/cpuinfo
processor       : 0
BogoMIPS        : 199.99
Features        : fp asimd evtstrm aes pmull sha1 sha2 crc32 cpuid
CPU implementer : 0x41
CPU architecture: 8
CPU variant     : 0x0
CPU part        : 0xd03
CPU revision    : 4

processor       : 1
BogoMIPS        : 199.99
```

```
Features        : fp asimd evtstrm aes pmull sha1 sha2 crc32 cpuid
CPU implementer : 0x41
CPU architecture: 8
CPU variant     : 0x0
CPU part        : 0xd03
CPU revision    : 4

processor       : 2
BogoMIPS        : 199.99
Features        : fp asimd evtstrm aes pmull sha1 sha2 crc32 cpuid
CPU implementer : 0x41
CPU architecture: 8
CPU variant     : 0x0
CPU part        : 0xd03
CPU revision    : 4

processor       : 3
BogoMIPS        : 199.99
Features        : fp asimd evtstrm aes pmull sha1 sha2 crc32 cpuid
CPU implementer : 0x41
CPU architecture: 8
CPU variant     : 0x0
CPU part        : 0xd03
CPU revision    : 4
```

- 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
- 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@mys_zu5ev # top
Mem: 65424K used, 3966032K free, 56K shrd, 492K buff, 4528K cached
CPU:   0% usr   0% sys   0% nic  99% idle   0% io   0% irq   0% sirq
Load average: 0.00 0.08 0.05 1/74 184
```

```
  PID   PPID  USER      STAT   VSZ  %VSZ  %CPU  COMMAND
  177      1  root      S      3516   0%    0%  -sh
  184    177  root      R      2560   0%    0%  top
    1      0  root      S      2388   0%    0%  init
  148      1  root      S      2388   0%    0%  /sbin/syslogd  -n
  151      1  root      S      2388   0%    0%  /sbin/klogd  -n
  127      2  root      SW        0   0%    0%  [irq/38-fd4a0000]
   85      2  root      IW        0   0%    0%  [kworker/2:1-eve]
  129      2  root      IW        0   0%    0%  [kworker/0:3-eve]
  107      2  root      IW        0   0%    0%  [kworker/2:2-rcu]
  124      2  root      IW        0   0%    0%  [kworker/0:2-eve]
   31      2  root      IW        0   0%    0%  [kworker/1:1-eve]
  117      2  root      SW        0   0%    0%  [kpktgend_0]
  118      2  root      SW        0   0%    0%  [kpktgend_1]
  119      2  root      SW        0   0%    0%  [kpktgend_2]
  120      2  root      SW        0   0%    0%  [kpktgend_3]
    2      0  root      SW        0   0%    0%  [kthreadd]
    3      2  root      IW<       0   0%    0%  [rcu_gp]
    4      2  root      IW<       0   0%    0%  [rcu_par_gp]
    6      2  root      IW<       0   0%    0%  [kworker/0:0H]
    7      2  root      IW        0   0%    0%  [kworker/u8:0-ev]
```

- %usr：表示用户空间程序的 cpu 使用率（没有通过 nice 调度）
- %sys：表示系统空间的 cpu 使用率，主要是内核程序
- %nic：表示用户空间且通过 nice 调度过的程序的 cpu 使用率
- %idle：空闲 cpu
- %irq：cpu 处理硬中断的数量
- %sirq：cpu 处理软中断的数量

3) 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@mys_zu5ev # echo "scale=5000; 4*a(1)" | bc -l -q &
[1] 716
```

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 top 命令检查 CPU 利用率的变化，如下所示：

```
root@mys_zu5ev # top
top - 08:50:51 up 8 min,  1 user,  load average: 0.65, 0.27, 0.14
Tasks:  94 total,   2 running,  92 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 :   3937.1 total,   3859.4 free,     45.0 used,     32.7 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used.   3801.9 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
  716 root      20   0    2736   1604   1352 R  99.7   0.0   0:57.47 bc
  109 root      20   0       0      0      0 S   0.3   0.0   0:00.03 kpktgend_0
  719 root      20   0    3288   2044   1628 R   0.3   0.1   0:00.05 top
    1 root      20   0    1916    616    556 S   0.0   0.0   0:01.42 init
    2 root      20   0       0      0      0 S   0.0   0.0   0:00.00 kthreadd
    3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
    4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
    7 root      20   0       0      0      0 I   0.0   0.0   0:00.00 kworker/u8:0-events_unbound
    8 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_percpu_wq
    9 root      20   0       0      0      0 S   0.0   0.0   0:00.00 ksoftirqd/0
   10 root      20   0       0      0      0 I   0.0   0.0   0:00.01 rcu_sched
```

```
    11 root      rt    0       0       0       0 S    0.0    0.0    0:00.00 migration/0

    12 root      20    0       0       0       0 S    0.0    0.0    0:00.00 cpuhp/0


. . .
```

约 3 分钟后，PI 结果被计算出来。 在此期间 CPU 使用率达到 100%，没有发生异常，说明 CPU 压力测试通过。还可以继续增加精确值，可进一步提高测试压力。

```
3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706798214808651328230664709384460955058
.....
74351362222477158915049530984448933309634087807693259939780541934144737744184263129860809988868741326047 20
```

4) CPU 工作频率

CPU 支持固定频率，并且 4 个 A53 核是同步操作。

查看CPU 的工作频率

```
root@mys_zu5ev # cat /sys/bus/cpu/devices/cpu0/cpufreq/cpuinfo_cur_freq
1199999
```

查看CPU 最大工作频率

```
root@mys_zu5ev # cat /sys/bus/cpu/devices/cpu0/cpufreq/cpuinfo_max_freq
1199999
```

查看CPU 的所有工作模式

```
root@mys_zu5ev # cat /sys/bus/cpu/devices/cpu0/cpufreq/scaling_available_governors
userspace
```

userspace：用户模式，任何情况下都会控制CPU 运行在配置的频率范围内,配置中的用户自己添加的省电设置。

查看CPU 的当前工作模式

```
root@mys_zu5ev # cat /sys/bus/cpu/devices/cpu0/cpufreq/scaling_governor
userspace
```

2.2. Memory

MYS-ZU5EV 内存默认是 4GB。如果需要使用 8G 或者其他容量的内存，需要在 vivado 中进行配置 8G 内存的参数，vivado 中其他的配置保持不变，生成的新 xsa 文件，然后按照《MYS-ZU5EV_Linux 软件开发指南》文档中的“5.5.Petalinux 项目下构建 FPGA 新工程的 BSP 软件”章节内容编译出镜像文件，就是适配 8G 内存的 petalinux 软件。

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@mys_zu5ev # cat /proc/meminfo
MemTotal:        4031456 kB
MemFree:         3965528 kB
MemAvailable:    3897128 kB
Buffers:             496 kB
Cached:             4588 kB
SwapCached:            0 kB
Active:             5508 kB
Inactive:            620 kB
Active(anon):       1084 kB
Inactive(anon):       12 kB
Active(file):       4424 kB
Inactive(file):      608 kB
Unevictable:           0 kB
Mlocked:               0 kB
SwapTotal:             0 kB
SwapFree:              0 kB
Dirty:                 0 kB
Writeback:             0 kB
AnonPages:          1040 kB
Mapped:             3256 kB
Shmem:                56 kB
KReclaimable:       9060 kB
Slab:              28460 kB
SReclaimable:       9060 kB
```

```
SUnreclaim:           19400 kB
KernelStack:           1216 kB
PageTables:              76 kB
NFS_Unstable:             0 kB
Bounce:                   0 kB
WritebackTmp:             0 kB
CommitLimit:        3668624 kB
Committed_AS:          7784 kB
VmallocTotal:     262930368 kB
VmallocUsed:           1832 kB
VmallocChunk:             0 kB
Percpu:                 832 kB
AnonHugePages:            0 kB
ShmemHugePages:           0 kB
ShmemPmdMapped:           0 kB
FileHugePages:            0 kB
FilePmdMapped:            0 kB
CmaTotal:            262144 kB
CmaFree:             261888 kB
HugePages_Total:          0
HugePages_Free:           0
HugePages_Rsvd:           0
HugePages_Surp:           0
Hugepagesize:          2048 kB
Hugetlb:                  0 kB
```

- 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@mys_zu5ev # free -m
total        used        free      shared     buffers      cached
Mem:          3936          64        3872           0           0           4
-/+ buffers/cache:          59        3877
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压力上的测试。可使用系统工具 memtester 进行测试，如指定内存大小 300MB，测试次数为 10，测试命令为“memtester 300M 10"。

下列以使用 300MB 内存空间，单次测试为例:

```
root@mys_zu5ev # memtester 300M 1
memtester version 4.3.0 (32-bit)
Copyright (C) 2001-2012 Charles Cazabon.
Licensed under the GNU General Public License version 2 (only).

pagesize is 4096
pagesizemask is 0xfffff000
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. eMMC

本节主要介绍 eMMC 的测试，适用于配置有 eMMC 存储器的开发板。

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。MYS-ZU5EV 核心板部分配置有 32GB eMMC，详细核心板配置参数请查看《MYS-ZU5EV 产品手册》。本节将讲解在 Linux 系统下查看与操作 eMMC 的步骤与方法。

1) 查看 eMMC 容量

通过 fdisk -l 命令可以查询到 eMMC 分区信息及容量。

```
root@mys_zu5ev # fdisk -l
Disk /dev/mmcblk0: 29.1 GiB, 31272730624 bytes, 61079552 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x00000000

Device          Boot Start       End   Sectors  Size Id Type
/dev/mmcblk0p1        63  61079129  61079067 29.1G 83 Linux
```

- /dev/mmcblk0p1 ：用于存放根文件系统

2) 查看 eMMC 分区信息

通过 df 命令可以查询到 eMMC 分区信息，使用情况，挂载目录等信息。

```
root@mys_zu5ev # df -h
Filesystem                Size      Used Available Use% Mounted on
/dev/root                28.5G    163.2M     26.9G   1% /
devtmpfs                  1.8G         0      1.8G   0% /dev
tmpfs                     1.9G         0      1.9G   0% /dev/shm
tmpfs                     1.9G     52.0K      1.9G   0% /tmp
tmpfs                     1.9G     28.0K      1.9G   0% /run
/dev/mmcblk1p1         1017.7M    397.4M    620.3M  39% /mnt/mmcblk1p1
/dev/mmcblk1p2           11.8G    454.7M     10.7G   4% /mnt/mmcblk1p2
/dev/mmcblk0p1           28.5G    163.2M     26.9G   1% /mnt/mmcblk0p1
```

- /dev/root：根文件系统，挂载到根目录下。
- tmpfs :内存虚拟文件系统，挂载到不同的目录下。
- devtmpfs ：用于系统创建 dev。
- /dev/mmcblk0p1 ：根文件系统，挂载在/mnt/mmcblk0p1。

3) eMMC 的性能测试

性能测试主要测试 eMMC 在 linux 系统下对文件的读写速度，一般结合 time 与 dd 双命令进行测试。

●写文件测试

```
root@mys_zu5ev #time dd if=/dev/zero of=tempfile bs=1M count=100 conv=fdat
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 5.11405 s, 20.5 MB/s

real    0m5.117s
user    0m0.000s
sys     0m0.382s
```

使用 dd 命令写文件时，需要加 conv=fdatasync 参数，表示当 dd 写 N 次结束之后，会 flush cache 同步到磁盘。因为对磁盘的写一般是先写到缓存还没有写到磁盘就返回了。这里测试出写磁盘速度为 20.5MB/s。

●读文件测试

在嵌入式系统中，经常需要测试系统文件读写性能，读文件时忽略 cache 的影响。这时可以指定参数 iflag=direct，nonblock。

```
root@mys_zu5ev # time dd if=tempfile of=/dev/null bs=1M count=100 iflag=dire
ct,nonblock
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 4.33127 s, 24.2 MB/s

real    0m4.334s
user    0m0.000s
sys     0m0.019s
```

可知，从磁盘直接读取读速度为 24.2MB/s。

2.4. QSPI

MYS-ZU5EV拥有一颗32M 的qspi，可以用来保存数据，这里请注意它的一个特性。内存写的区域必须是擦除后的区域；擦除是按block（0x10000)作为最小单位。

1）文件系统下读写

下面将演示在文件系统下的测试步骤，下面是一些qspi 读写的命令可以供参考

- hexdump :查看分区内容
- mtd_debug read：用来读取 qspi 数据到文件
- mtd_debug wirte：用来写入文件数据到 qspi
- mtd_debug erase ：擦除 qspi 数据

●查看存储内容

使用hexdump 来查看QSPI 存储的内容：

```
root@mys_zu5ev # hexdump -C /dev/mtd0
00000000  00 00 00 14 00 00 00 14  00 00 00 14 00 00 00 14  |................|
```

●写数据

通过flashcp 命令向QSPI 写数据，并查看写入内容：

```
root@mys_zu5ev # flashcp -v qspi_test.txt /dev/mtd0
Erasing blocks: 1/1 (100%)
Writing data: 0k/0k (100%)
Verifying data: 0k/0k (100%)
[root@myir ~]# hexdump -C /dev/mtd0
00000000  6d 79 69 72 20 71 73 70  69 20 74 65 73 74 0a ff  |myir qspi test..|
```

●读数据

通过dd 命令读取QSPI 数据：

```
root@mys_zu5ev # dd if=/dev/mtd0 of=read.txt bs=14 count=1
1+0 records in
1+0 records out
root@myir:~# cat read.txt
myir qspi test
```

2）uboot 下读写 qspi

下面将演示在 uboot 阶段读取 qspi 步骤，uboot 下内存操作指令一般是 mw 和md。

● Uboot 内存读取命令

➢ mw 命令

uboot 下输入指令 mw,会提示 mw 的用法，内存写命令:

```
ZynqMP> mw
mw - memory write (fill)
Usage:
mw [.b, .w, .l, .q] address value[count]
```

其中 b:8 位 w:16 位 l:32 位（默认值），代表每次写入的位数；address 是要写入内存的地址，value 是要写入的值，count 是从 address 开始要写入多少个数，这些都是 16 进制数。

➢ md 命令

uboot 下输入指令 md,会提示 md 的用法，内存读命令:

```
ZynqMP> md
md - memory
display Usage:
md [.b, .w, .l, .q] address [# of
objects]
```

其中 b/w/l 的意思同上，默认是是以 32 位输出。

● 初始化 QSPI

这里需要用到SPI flash子系统的sf命令，uboot下输入sf，会提示sf命令的用法。上电按任意键进入uboot shell界面，执行以下命令初始化qspi：

```
ZynqMP> sf probe
SF: Detected s25fl256s1 with page size 256 Bytes, erase size 64 KiB, total 32 MiB
```

● 请空内存后读取数据

清空一段内存后读取 qspi 内容：

```
ZynqMP> mw.b 0x48000000 0xff 0x10
ZynqMP> md 0x48000000 0x10
48000000:   ffffffff   ffffffffffffffffffffffff      ................
48000010:  00000000  00000000  00002014  00000000      ........ ......
```

```
48000020: 00000000 00000000 00000000 00000000    ................
48000030: 00002011 00000000 00000001 00000000    . ............
ZynqMP> sf read 0x48000000 0x10 0x10
device 0 offset 0x10, size 0x10
SF: 16 bytes @ 0x10 Read: OK u-
ZynqMP>  md  0x48000000  0x10
u-boot=> md 0x48000000 0x10

48000000: 7269796d 70737120 65742069 ffff7473    myir qspi test..
48000010: 00000000 00000000 00002014 00000000    ........ ......
48000020: 00000000 00000000 00000000 00000000    ................
48000030: 00002011 00000000 00000001 00000000    . ............
```

这里用 sf read 命令从 QSPI 偏移地址 0x10 处读取 16 字节到内存 0x48000000 地址处。由于先前已经在 QSPI 的 0x10 地址写了数据，所以这里能看到读取了文件系统中写入的“myir qspi test“内容。

- **清空内存后写数据**

清空内存后，给内存重新写入 16 字节数据，然后向 qspi 的 0x100 偏移处写入 16 个字节的 0x12：

```
ZynqMP>mw.b 0x48000000 0xff 0x10
ZynqMP>mw.b 0x48000000 0x12 0x10
ZynqMP>md 0x48000000 0x10

48000000: 12121212 12121212 12121212 12121212    ................
48000010: 00000000 00000000 00002014 00000000    ........ ......
48000020: 00000000 00000000 00000000 00000000    ................
48000030: 00002011 00000000 00000001 00000000    . ............
```

```
ZynqMP>sf write 0x48000000 0x100 0x10
device 0 offset 0x100, size 0x10
SF: 16 bytes @ 0x100 Written: OK
u-boot=>
```

●清空内存后再次读取

把内存内容清空后再次把写入 qspi 的内容读入到内存：

```
ZynqMP>mw.b 0x48000000 0xff 0x10
ZynqMP>sf read 0x48000000 0x100 0x10
device 0 offset 0x100, size 0x10
SF: 16 bytes @ 0x100 Written: OK
ZynqMP> mw.b 0x48000000 0xff 0x10
ZynqMP>sf read 0x48000000 0x100 0x10
device 0 offset 0x100, size 0x10
SF: 16 bytes @ 0x100 Read: OK
ZynqMP>md 0x48000000 0x10

48000000: 12121212 12121212 12121212 12121212    ................
48000010: 00000000 00000000 00002014 00000000    ........ ......
48000020: 00000000 00000000 00000000 00000000    ................
48000030: 00002011 00000000 00000001 00000000    . ............
```

可以看到能正常从 QSPI 读出了写入的数据，这里也可以参考文件系统读的方法来查看 qspi 内容。

2.5. 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。

MYS-ZU5EV 拥有一个内部 RTC。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。

●查看系统 RTC 设备

```
root@mys_zu5ev # ls /dev/rtc* -al
lrwxrwxrwx    1 root     root             4 Apr  1 07:24 /dev/rtc -> rtc0
crw-------    1 root     root      252,   0 Apr  1 07:24 /dev/rtc0
root@mys_zu5ev #  cat /sys/class/rtc/rtc0/name
rtc_zynqmpffa60000.rtc
root@mys_zu5ev #
```

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

●设置系统时间

在将系统时间设置为 Sat Jan 30 21:15:30 UTC 2021：

```
root@mys_zu5ev # date  013021152021.30
Sat Jan 30 21:15:30 UTC 2021
```

●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@mys_zu5ev # hwclock -w
```

●读取 RTC 时间并设置为系统时间

```
root@mys_zu5ev # hwclock -r
2021-01-30 21:16:18.178677+00:00
```

●掉电保持 RTC 时间

将开发板关机断开电源，经过 2 分钟左右，重新上电开机。查看 RTC 时间和系统时间：

```
root@mys_zu5ev # hwclock -r
```

```
2021-01-30 21:43:10.955237+00:00
```

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。

●将系统时间与 RTC 时间同步

```
root@mys_zu5ev # hwclock -s
root@mys_zu5ev # date
Sat Jan 30 21:43:46 UTC 2021
```

如果将 hwclock -s 命令添加到启动脚本中就可以保证每次启动都可以将系统时间和 RTC 时间保持同步了。

2.6. 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。MYS-ZU5EV 使用外部看门狗芯片 CAT823TTDI-GT3，该芯片的喂狗引脚连接到了 CPU 的 PS_MIO38 引脚，看门狗还有一个开关连到 PS_MIO33 引脚。

在使用看门狗的时候需要注意看门狗的一些特性，一般有如下两点：

本节将演示 watchdog 的使用，模拟看门狗系统复位功能，并提供修改示例设置看门狗超时时间。

1）关闭看门狗测试

将看门狗开关 PS_MIO33 引脚置零，尝试关闭看门狗：

```
root@mys_zu5ev # echo 0 >/sys/class/leds/wdt_en/brightness
```

2）看门狗喂狗测试

```
root@mys_zu5ev #echo 200 >/sys/class/myir-watchdog/wd_period_ms（正常喂狗）
root@mys_zu5ev # echo 1>/sys/class/leds/wdt_en/brightness
root@mys_zu5ev # echo 4000 >/sys/class/myir-watchdog/wd_period_ms（喂狗超时，看门狗将会复位系统）
```

如果喂狗超时，开发板将被看门狗复位。

3. 基本外设接口

3.1. GPIO

GPIO 的测试是通过文件系统 sysfs 接口来实现的，下面内容以 MIO6 为例说明 zynq MP 平台 GPIO 的使用过程。

在 ZYNQMP 平台的 linux 系统中，Gpio 编号由 Gpio Chip Id + MIO 号组成；EMIO 的 Gpio 编号由 Gpio Chip Id + 78 + EMIO 号组成，其中 ChipID 可通过一下方式查看：

```
root@mys_zu5ev #ls /sys/class/gpio
export          gpiochip338   unexport
cat /sys/class/gpio/gpiochip338/base
338
```

由上可知 ChipID 为 338，所以 MIO6 的 Gpio 编号为 338+6=344

1) 导出 GPIO

```
root@mys_zu5ev # echo 344 > /sys/class/gpio/export
```

导出成功后会在/sys/class/gpio/目录下生成 gpio344 这个目录。

2) 设置/查看 GPIO 方向

●设置输入

```
root@mys_zu5ev # echo "in" > /sys/class/gpio/gpio344/direction
```

●设置输出

```
root@mys_zu5ev # echo "out" > /sys/class/gpio/gpio344/direction
```

●查看 gpio 方向

```
root@mys_zu5ev # cat /sys/class/gpio/gpio344/direction
out
```

返回 in 表示输入，返回 out 表示输出。

3) 设置/查看 GPIO 的值

- **设置输出低**

```
root@mys_zu5ev # echo "0" > /sys/class/gpio/gpio344/value
```

- **设置输出高**

```
root@mys_zu5ev # echo "1" > /sys/class/gpio/gpio344/value
```

- **查看 gpio 的值**

```
root@mys_zu5ev # cat /sys/class/gpio/gpio344/value
1
```

3.2. LED 灯

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。在硬件资源列表中，我们已经列出了开发板上所有的 LED。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/leds

```
root@mys_zu5ev # ls /sys/class/leds/
led_sys@  mmc0::@  mmc1::@  rs485_de@  wdt_en@
```

其中 led_sys 为核心板的可自定义 led，通过向/sys/class/leds/led_sys/brightness 写入不同的值可以改变心跳灯的亮灭占空比。

2）以心跳灯 1 为例测试 LED

●关闭心跳灯

```
root@mys_zu5ev # echo none > /sys/class/leds/led_sys/trigger
```

关闭心跳灯，然后可以单独进行 led 关闭和开启操作。

●熄灭 LED

```
root@mys_zu5ev # echo 1 > /sys/class/leds/led_sys/brightness
```

●点亮 LED

```
root@mys_zu5ev # echo 0 >/sys/class/leds/led_sys/brightness
```

●开启 LED 触发模式

开启“timer”出发模式后，LED 默认以 1Hz 周期闪烁，占空比为 50%：

```
root@mys_zu5ev #  echo "timer" >/sys/class/leds/led_sys/trigger
```

●改变 LED 灯闪烁时的亮灭占空比

通过调整 led 下的 delay_on 和 delay_off 属性，可以调整 LED 闪烁周期和亮灭占空比，例如：

```
root@mys_zu5ev # echo "100" >/sys/class/leds/led_sys/delay_on
root@mys_zu5ev # echo "500" >/sys/class/leds/led_sys/delay_off
```

3.3. USB

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

●查看 USB 设备信息

将 U 盘连接到开发板 USB Host 接口，内核提示信息如下：

```
root@mys_zu5ev #
[  755.579811] usb 2-1.4: new SuperSpeed Gen 1 USB device number 3 using xhci-hcd
[  755.628681] usb 2-1.4: New USB device found, idVendor=1f75, idProduct=0918, bcdDevice= 3.10
[  755.637028] usb 2-1.4: New USB device strings: Mfr=2, Product=3, SerialNumber=4
[  755.644337] usb 2-1.4: Product: STORAGE DEVICE
[  755.648779] usb 2-1.4: Manufacturer: Device Storage
[  755.653651] usb 2-1.4: SerialNumber: 09294138
[  755.658659] usb-storage 2-1.4:1.0: USB Mass Storage device detected
[  755.665157] scsi host0: usb-storage 2-1.4:1.0
[  756.674979] scsi 0:0:0:0: Direct-Access     Specific STORAGE DEVICE   0009 PQ: 0 ANSI: 6
[  756.684483] sd 0:0:0:0: [sda] 61440000 512-byte logical blocks: (31.5 GB/29.3 GiB)
[  756.692636] sd 0:0:0:0: [sda] Write Protect is off
[  756.697992] sd 0:0:0:0: [sda] Write cache: disabled, read cache: enabled, doesn't support DPO or FUA
[  756.723524]  sda: sda1
[  756.727437] sd 0:0:0:0: [sda] Attached SCSI removable disk
[  756.889397] FAT-fs (sda1): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
[  756.898896] vfat filesystem being mounted at /run/media/sda1 supports timestamps until 2107 (0x10391447e)
```

从上述信息可以得出需要挂载的设备为 sda1。

2) U 盘挂载读写

●挂载 U 盘

提供的启动镜像中，USB 存储设备都会自动挂载。

●读文件

需要提前在 U 盘上建立一个 test.txt 文件。

```
root@mys_zu5ev # ls /run/media/sda1/
test.txt
root@mys_zu5ev # cat /run/media/sda1/test.txt
hello world!
```

●写文件

```
root@mys_zu5ev # touch test.txt
root@mys_zu5ev # echo “hello world !!!” > test.txt
root@mys_zu5ev # cp test.txt /run/media/sda1
root@mys_zu5ev # cat /run/media/sda1/test.txt
hello world !!!
```

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 U 盘设备。

3) 卸载 U 盘

●卸载操作

```
root@mys_zu5ev # umount /run/media/sda1/
```

3.4. Micro SD 卡

Micro SD Card，原名 Trans-flash Card(TF 卡)，Micro SD 卡是一种极细小的快闪存储器卡。Micro SD 卡相比标准 SD 卡，外形上更加小巧，是 SD 卡类型中尺寸最小的一种 SD 卡。尽管 Micro SD 卡的外形大小及接口形状与原来的 SD 卡有所不同，但接口规范保持不变，确保了兼容性。若将 Micro SD 插入特定的转接卡中，可当作标准 SD 卡来使用，SD 卡已成为目前消费数码设备中应用最广泛的一种存储卡，具有大容量、高性能、安全等多种特点的多功能存储卡。Micro SD 卡背面一般有 9 个引脚，包含 4 根数据线，支持 4bit 数据传输宽度。

ZynqMP 系列芯片支持 2 路 SD 接口，MYS-ZU5EV 开发板上使用 SD0 连接 Micro SD，这一路接口相关的硬件规格参数如下：

- 完全符合 SDIO 2.0 接口规范
- 支持 SDHC Class 10 MicroSD 卡

本节将讲解在 Linux 系统下查看与操作 TF 卡的步骤与方法。

1) 查看 TF 卡容量

通过 fdisk -l 命令可以查询到 TF 卡分区信息及容量：

```
root@mys_zu5ev # fdisk -l
略
Disk /dev/mmcblk1: 14.5 GiB, 15552479232 bytes, 30375936 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xdc3c8c4b

Device          Boot  Start      End  Sectors     Size Id Type
/dev/mmcblk1p1 *         63  2088449  2088387 1019.7M  c W95 FAT32 (LB
A)
/dev/mmcblk1p2        2088450 30362849 28274400    13.5G 83 Linux
```

- /dev/mmcblk1p1：用于存放 BOOT.bin、image.ub 等启动文件
- /dev/mmcblk1p2：用于存放根文件系统分区

2) TF 卡的性能测试

性能测试主要测试 TF 卡在 linux 系统下对文件的读写速度，一般结合 time 与 dd 双命令进行测试。挂载需要测试的 TF 卡分区，这里以第一个分区/dev/mmcblk1p1 为例，挂载目录为/mnt/mmcblk1p1/。

●写文件测试

```
root@mys_zu5ev #time dd if=/dev/zero of=tempfile bs=1M count=100 conv=
fdatasync
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 16.6057 s, 6.3 MB/s

real    0m16.609s
user    0m0.008s
sys     0m0.565s
```

使用 dd 命令写文件时，需要加 conv=fdatasync 参数，表示当 dd 写 N 次结束之后，会 flush cache 同步到磁盘。因为对磁盘的写一般是先写到缓存还没有写到磁盘就返回了。这里测试出写磁盘速度为 6.3MB/s。

●读文件测试

读文件时忽略 cache 的影响。这是可以指定参数 iflag=direct，nonblock。

```
root@mys_zu5ev #  time dd if=tempfile of=/dev/null bs=1M count=100 iflag
=direct,nonblock
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 4.45101 s, 23.6 MB/s

real    0m4.454s
user    0m0.000s
sys     0m0.027s
```

可知，直接从 SD 卡读数据速度为 23.6MB/s。

3.5. Display

MYS-ZU5EV 目前支持 DP 显示方案。DP 接口最大支持 4K 60FPS 分辨率输出。

1）DP 显示测试

由于 MYS-ZU5EV 为 DP 显示，进入 linux 系统，然后执行以下命令：

```
root@mys_zu5ev:~#/etc/init.d/xserver-nodmstop
root@mys_zu5ev:~# /usr/share/examples/widgets/painting/deform/deform -platform linuxfb:fb=/dev/fb0
```

可以看到 DP 显示器上的 qt 界面。

3.6. 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。

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@mys_zu5ev # ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0a:35:00:01:22
          UP 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)
Interrupt:28 Base address:0xb000
```

eth0 为实际的以太网设备，默认采用的是固定的硬件 MAC 地址

下面介绍给 eth0 手动配置 IP 地址 192.168.0.100 的方法，命令如下：

```
root@mys_zu5ev # ifconfig eth0 192.168.0.100 netmask 255.255.255.0 up
```

上面的命令手动配置 eth0 的 IP 地址为 192.168.0.100，子网掩码为 255.255.255.0，以及默认配置的广播地址 192.168.0.255，并通过 up 参数进行激活，如下所示：

```
root@mys_zu5ev # ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0a:35:00:01:22
          inet addr:192.168.0.100  Bcast:192.168.1.255  Mask:255.255.255.0
          UP 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)
Interrupt:28 Base address:0xb000
```

●使用 iproute2 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@mys_zu5ev # ip addr flush dev eth0
root@mys_zu5ev # ip addr add 192.168.0.101/24 brd + dev eth0
root@mys_zu5ev # 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@mys_zu5ev # ip addr show eth0
3: eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state
DOWN group default qlen 1000
    link/ether fc:69:47:33:a5:65 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.101/24 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。

● 自动永久配置动态获取 IP 地址

MYS-ZU5EV 开发板默认是通过 DHCP 自动获取 IP 地址的，具体可以查看/etc/network/目录下的配置文件 interfaces。

```
root@mys_zu5ev # cat /etc/network/interfaces -- configuration file for ifup(8),
 ifdown(8)
# The loopback interface
auto lo
iface lo inet loopback
…
auto eth0
iface eth0 inet dhcp
iface eth1 inet dhcp
…
```

以上配置文件将会对 wlan0 和 eth0 匹配的网卡进行配置，如果内核启动参数中不包含 nfsroot 参数时，则通过 DHCP 自动为匹配到的网卡配置 IP 地址，网关，DNS 等信息。详细的配置参数参见以下链接：https://wiki.debian.org/NetworkConfiguration。

● 自动永久配置静态 IP 地址

如果需要对 eth0 配置永久 IP 地址，则需要修改/etc/network/interfaces 文件，修改内容如下内容如下：

```
# Wired or wireless interfaces
auto eth0
#iface eth0 inet dhcp
iface eth0 inet static
        address 192.168.30.100
        netmask 255.255.255.0
gateway 192.168.30.1
        dns-nameservers 223.5.5.5 114.114.114.114
iface eth1 inet dhcp
```

配置完成之后，执行以下命令重启 network 服务重新启动，会发现 eth0 网卡的地址已经配置为设定的 192.168.30.100 了，如下所示：

```
root@mys_zu5ev #/etc/init.d/networking restart
root@mys_zu5ev #ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0A:35:00:22:01
          inet addr:192.168.30.100  Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::20a:35ff:fe00:2201/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6756 errors:0 dropped:0 overruns:0 frame:0
          TX packets:715 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:386149 (377.0 KiB)  TX bytes:105843 (103.3 KiB)
          Interrupt:30
```

3.7. CAN

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。

●初始化 CAN 网络接口

参考下面命令分别将两块开发板的波特率设置为 500K：

```
root@mys_zu5ev #ip link set can0 up type can bitrate 500000
[ 2056.319008] xilinx_can ff070000.can can0: bitrate error 0.0%
[ 2056.324869] IPv6: ADDRCONF(NETDEV_CHANGE): can0: link becomes ready
```

至此，即开启 can 功能。

●发送数据

设置其中一块板子为发送，并使用 cansend 发送 64 字节固定格式的字符串来测试：

```
root@mys_zu5ev #cansend can0 123#DEADBEEF
root@mys_zu5ev #cansend can0 123#DEADBEEF
```

●接收数据

设置另外一块板子为接收，可以使用 candump 来查看 CAN 的接收数据：

```
root@mys_zu5ev # candump can0 -L
(1621329829.902018) can0 123#DEADBEEF
(1621329829.902076) can0 123#DEADBEEF
```

●统计 can0 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@mys_zu5ev # ip -details -statistics link show can0
2: can0: <NOARP,UP,LOWER_UP,ECHO> mtu 16 qdisc pfifo_fast state UP mod
e DEFAULT group default qlen 10
    link/can  promiscuity 0 minmtu 0 maxmtu 0
    can <LOOPBACK> state ERROR-ACTIVE (berr-counter tx 0 rx 0) restart-ms
 0
          bitrate 499999 sample-point 0.875
```

```
          tq 250 prop-seg 3 phase-seg1 3 phase-seg2 1 sjw 1
          xilinx_can: tseg1 1..16 tseg2 1..8 sjw 1..4 brp 1..256 brp-inc 1
          clock 99999999
          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
    20         3        0       0       0       0
    TX: bytes  packets  errors  dropped carrier collsns
    20         3        0       0       0       0
```

4. 网络应用

开发板出厂烧录的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。

4.1. 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 网线将开发板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：

```
root@mys_zu5ev # [  208.032262] macb ff0e0000.ethernet eth0: link up (1000/Full)
```

2）测试外网网址

```
root@mys_zu5ev # ping www.baidu.com -I eth0
PING www.baidu.com (14.215.177.38): 56 data bytes
64 bytes from 14.215.177.38: seq=0 ttl=53 time=19.013 ms
64 bytes from 14.215.177.38: seq=1 ttl=53 time=18.602 ms
64 bytes from 14.215.177.38: seq=2 ttl=53 time=18.590 ms
64 bytes from 14.215.177.38: seq=3 ttl=53 time=18.973 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 14.215.177.38，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。

4.2. 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@mys_zu5ev # ifconfig
eth0      Link encap:Ethernet  HWaddr 26:BB:D9:BC:8C:DC
          inet addr:192.168.40.172  Bcast:192.168.40.255  Mask:255.255.255.0
          inet6 addr: fe80::24bb:d9ff:febc:8cdc/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:383 errors:0 dropped:0 overruns:0 frame:0
          TX packets:95 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:22328 (21.8 KiB)  TX bytes:9174 (8.9 KiB)
          Interrupt:22
```

SSH 服务器的 IP 地址为 192.168.40.172，用 ping 命令可测试开发板和 SSH 服务器之间的连接是否正常。

●SSH 客户端测试

开发板作为客户端连接 SSH 服务器，在开发板上使用 ssh 命令登陆服务器，命令和结果如下：

```
root@mys_zu5ev # ssh fengy@192.168.40.2
The authenticity of host '192.168.40.2 (192.168.40.2)' can't be established.
ECDSA key fingerprint is SHA256:KCTJPoHzafew1fRJmTx2hV4BNymgaZ1WDg2o
vdtqtCw.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.40.2' (ECDSA) to the list of known host
s.
fengy@192.168.40.2's password:
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-101-generic x86_64)
```

```
 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

1677 个可升级软件包。
306 个安全更新。

New release '18.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Thu May 27 15:37:37 2021 from 192.168.40.37
fengy@myir-server1:~$
```

其中 fengy 为服务器上的用户名。

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 fengy 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，其它外部设备远程连接到此台开发板。

开发板端默认启动了 SSH 服务，因此我们也可以在其它具有 SSH 客户端的外部设备（开发板或者 PC）上使用 ssh 命令登陆到当前的开发板上，命令和结果如下：

```
$ ssh root@192.168.40.172
```

上面的示例中，我们从远程以 root 账户登录到了此开发板上，并进入 console 控制台，可以对开发板执行 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电脑主机/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。

4.3. 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。

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

```
fengy@myir-server1:~$ scp BOOT.bin root@192.168.40.172:/root/
root@192.168.40.172's password:
BOOT.bin
                    100% 8790KB    8.6MB/s    00:01
```

进入开发板 home 目录可以看到此文件，如下：

```
root@mys_zu5ev # ls
BOOT.BIN
```

2）从本地拷贝文件到远程

```
root@mys_zu5ev # scp BOOT.bin fengy@192.168.40.2:/home/fengy
fengy@192.168.40.2's password:
```

拷贝的过程中需要按照提示输入，验证成功之后文件从开发板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。

4.4. TFTP

与 FTP 一样，TFTP 使用客户端和服务器软件在不同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@mys_zu5ev # tftp --help
BusyBox v1.31.0 (2020-05-26 14:38:25 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 tftpd 32/64(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 test_file 放置到上面配置的<WORKDIR>/tftpboot/目录，就可以在开发板上使用 tftp 客户端进行文件的下载和上传了。

```
root@mys_zu5ev # tftp -g -r test_file 192.168.0.2
```

上面的命令会把 tftp 服务端<WORKDIR>/tftpboot 目录下的 test_file 下载到开发板当前目录下。

```
root@mys_zu5ev # tftp -p -r test_file 192.168.0.2
```

上面的命令会把开发板上当前目录下的 test_file 文件上传到 tftp 服务端之前配置的<WORKDIR>/tftpboot 目录下。

4.5. DHCP

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。

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

```
root@mys_zu5ev # udhcpc -i eth0
udhcpc: started, v1.29.3
udhcpc: sending discover
udhcpc: sending select for 192.168.40.172
udhcpc: lease of 192.168.40.172 obtained, lease time 7200
deleting routers
adding dns 223.5.5.5
adding dns 201.104.111.114
```

不管通过哪种方式，最终都可以为 eth0 配置好 IP 地址，以及网关，子网掩码，DNS 等信息如下：

```
root@mys_zu5ev # ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 26:BB:D9:BC:8C:DC
          inet addr:192.168.40.172  Bcast:192.168.40.255  Mask:255.255.255.0
          inet6 addr: fe80::24bb:d9ff:febc:8cdc/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:17369 errors:0 dropped:0 overruns:0 frame:0
          TX packets:11359 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:10147310 (9.6 MiB)  TX bytes:9862414 (9.4 MiB)
          Interrupt:22
[root@myir ~]# cat /etc/resolv.conf
# Generated by Connection Manager
nameserver ::1
```

```
nameserver 127.0.0.1
search bad # eth0
nameserver 223.5.5.5 # eth0
nameserver 201.104.111.114 # eth0
```

4.6. 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 系统的开发板通常使用 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，禁止网络上的其它外部设备对其进行 ping 探测。具体命令使用参见：https://linux.die.net/man/8/iptables

1）配置开发板 iptables

在开发板上使用 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@mys_zu5ev # iptables -A INPUT -p icmp --icmp-type 8 -j DROP
root@mys_zu5ev # iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-A INPUT -p icmp -m icmp --icmp-type 8 -j DROP
```

2）ping 测试

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

```
PC$ ping 192.168.40.172 -w 10
PING 192.168.0.60 (192.168.0.60) 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 通开发板。

3）删掉对应的防火墙规则

```
root@mys_zu5ev # iptables -F
root@mys_zu5ev # iptables -S
```

```
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
```

4) 再次测试 ping 开发板

```
PC$ ping 192.168.40.172 -w 5
PING 192.168.40.172 (192.168.40.172) 56(84) bytes of data.
64 bytes from 192.168.40.172: icmp_seq=1 ttl=64 time=0.254 ms
64 bytes from 192.168.40.172: icmp_seq=2 ttl=64 time=0.219 ms
64 bytes from 192.168.40.172: icmp_seq=3 ttl=64 time=0.222 ms
64 bytes from 192.168.40.172: icmp_seq=4 ttl=64 time=0.226 ms
64 bytes from 192.168.40.172: icmp_seq=5 ttl=64 time=0.238 ms
64 bytes from 192.168.40.172: icmp_seq=6 ttl=64 time=0.236 ms

--- 192.168.40.172 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 开发板，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。

4.7. 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@mys_zu5ev # 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
.....................
```

查看开发板以太网卡的基本信息：

```
root@mys_zu5ev # ethtool eth0
Settings for eth0:
        Supported ports: [ TP MII ]
        Supported link modes:   10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/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
                                1000baseT/Full
        Advertised pause frame use: No
        Advertised auto-negotiation: Yes
        Advertised FEC modes: Not reported
        Link partner advertised link modes:  10baseT/Half 10baseT/Full
                                             100baseT/Half 100baseT/Full
                                             1000baseT/Full
        Link partner advertised pause frame use: Symmetric Receive-only
        Link partner advertised auto-negotiation: Yes
        Link partner advertised FEC modes: Not reported
        Speed: 1000Mb/s
        Duplex: Full
        Port: MII
        PHYAD: 4
        Transceiver: internal
        Auto-negotiation: on
        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 6 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@mys_zu5ev # 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。

4.8. 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(IPV4 和 IPV6 下的 TCP、UDP、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，被测试的开发板作为客户端分别测试开发板网卡 TCP 和 UDP 的性能。

首先在服务器上安装 iperf3，如下：

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

将服务器和开发板通过 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 192.168.40.143，设开发板 IP 为 192.168.40.100，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传输转发的影响。

1）测试 TCP 性能

●服务端（192.168.40.143）

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（192.168.40.100）

开发板上运行的 iperf3 程序工作在客户端，TCP 模式，其中参数说明如下：

- -c 192.168.40.143 ：工作在客户端，连接服务端 192.168.40.143
- -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@mys_zu5ev # iperf3 -c 192.168.40.143 -i 2 -t 10
Connecting to host 192.168.40.143, port 5201
[ 5] local 192.168.40.100 port 43446 connected to 192.168.40.143 port 5201
[ ID] Interval Transfer Bitrate Retr Cwnd
```

```
[  5]   0.00-2.00   sec   223 MBytes   933 Mbits/sec    0    714 KBytes

[  5]   2.00-4.00   sec   218 MBytes   915 Mbits/sec   49    667 KBytes

[  5]   4.00-6.00   sec   221 MBytes   927 Mbits/sec   29    556 KBytes

[  5]   6.00-8.00   sec   220 MBytes   922 Mbits/sec    0    591 KBytes

[  5]   8.00-10.00  sec   219 MBytes   917 Mbits/sec    0    597 KBytes

- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-10.00  sec  1.07 GBytes   923 Mbits/sec   78             sender
[  5]   0.00-10.00  sec  1.07 GBytes   921 Mbits/sec                  receiver

iperf Done.
```

客户端经过 10 秒之后测试结束并显示上面的测试结果，表明 TCP 带宽为 923Mbps 左右，没有重传，测试时 TCP 窗口值为 597KBytes.

同时服务端也显示测试结果如下，然后继续监听 5201 端口等待客户端连接：

```
PC $ iperf3 -s -i 2
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 192.168.40.100, port 43444
[  5] local 192.168.40.143 port 5201 connected to 192.168.40.100 port 43446
[ ID] Interval           Transfer     Bandwidth
[  5]   0.00-2.00   sec   220 MBytes   921 Mbits/sec
[  5]   2.00-4.00   sec   219 MBytes   917 Mbits/sec
[  5]   4.00-6.00   sec   221 MBytes   925 Mbits/sec
[  5]   6.00-8.00   sec   219 MBytes   920 Mbits/sec
[  5]   8.00-10.00  sec   219 MBytes   917 Mbits/sec
[  5]  10.00-10.02  sec  1.38 MBytes   530 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
```

```
[ ID] Interval           Transfer     Bandwidth       Retr
[  5]   0.00-10.02  sec  1.07 GBytes   921 Mbits/sec   78             sender
[  5]   0.00-10.02  sec  1.07 GBytes   919 Mbits/sec                  receiver
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

2）测试 UDP 性能

●服务端（192.168.40.143）

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（192.168.40.100）

开发板上 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 192.168.40.143：工作在客户端，连接服务端 192.168.40.143
- -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 100M：设定 UDP 传输带宽为 100Mbps.

```
root@mys_zu5ev # iperf3 -u -c 192.168.40.143 -i 2 -t 10 -b 100M
Connecting to host 192.168.40.143, port 5201
[  5] local 192.168.40.100 port 50499 connected to 192.168.40.143 port 5201
[ ID] Interval           Transfer     Bitrate         Total Datagrams
[  5]   0.00-2.00   sec  23.8 MBytes   100 Mbits/sec  17263
[  5]   2.00-4.00   sec  23.8 MBytes   100 Mbits/sec  17264
[  5]   4.00-6.00   sec  23.8 MBytes   100 Mbits/sec  17265
[  5]   6.00-8.00   sec  23.8 MBytes   100 Mbits/sec  17265
[  5]   8.00-10.00  sec  23.8 MBytes   100 Mbits/sec  17265
- - - - - - - - - - - - - - - - - - - - - - - - -
```

```
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagra
ms
[  5]   0.00-10.00  sec   119 MBytes   100 Mbits/sec  0.000 ms  0/86322
(0%)  sender
[  5]   0.00-10.00  sec   119 MBytes  99.7 Mbits/sec  0.176 ms  243/86321
(0.28%)  receiver

iperf Done.
```

客户端经过 10 秒之后测试结束并显示上面的测试结果。同时服务端也显示测试结果如下，然后继续监听 5201 端口等待客户端连接：

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

```
root@myir:~# iperf3 -u -c 192.168.40.143 -i 2 -t 10 -b 100M
Connecting to host 192.168.40.143, port 5201
[  5] local 192.168.40.100 port 50499 connected to 192.168.40.143 port 5201
[ ID] Interval           Transfer     Bitrate         Total Datagrams
[  5]   0.00-2.00   sec  23.8 MBytes   100 Mbits/sec  17263
[  5]   2.00-4.00   sec  23.8 MBytes   100 Mbits/sec  17264
[  5]   4.00-6.00   sec  23.8 MBytes   100 Mbits/sec  17265
[  5]   6.00-8.00   sec  23.8 MBytes   100 Mbits/sec  17265
[  5]   8.00-10.00  sec  23.8 MBytes   100 Mbits/sec  17265
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagra
ms
[  5]   0.00-10.00  sec   119 MBytes   100 Mbits/sec  0.000 ms  0/86322
(0%)  sender
[  5]   0.00-10.00  sec   119 MBytes  99.7 Mbits/sec  0.176 ms  243/86321
(0.28%)  receiver

iperf Done.
```

客户端修改-b 参数，继续增大指定的 UDP 带宽，当开始出现丢包时测到的 UDP 带宽即为实际的 UDP 带宽。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。

5. 图形系统

本章主要测试的是 MYS-ZU5EV 的图形显示。

Linux 图形系统是 Linux 系统中比较复杂的一个子系统，一直处于不停的变革当中。它位于底层显示相关的设备驱动和上层用户界面应用之间，屏蔽了形形色色的底层硬件差异，同时又向上层用户界面提供统一的 API。

Linux/Unix 环境下最早的图形系统是 Xorg 图形系统，下面是一个典型的嵌入式系统下图形系统的结构图。

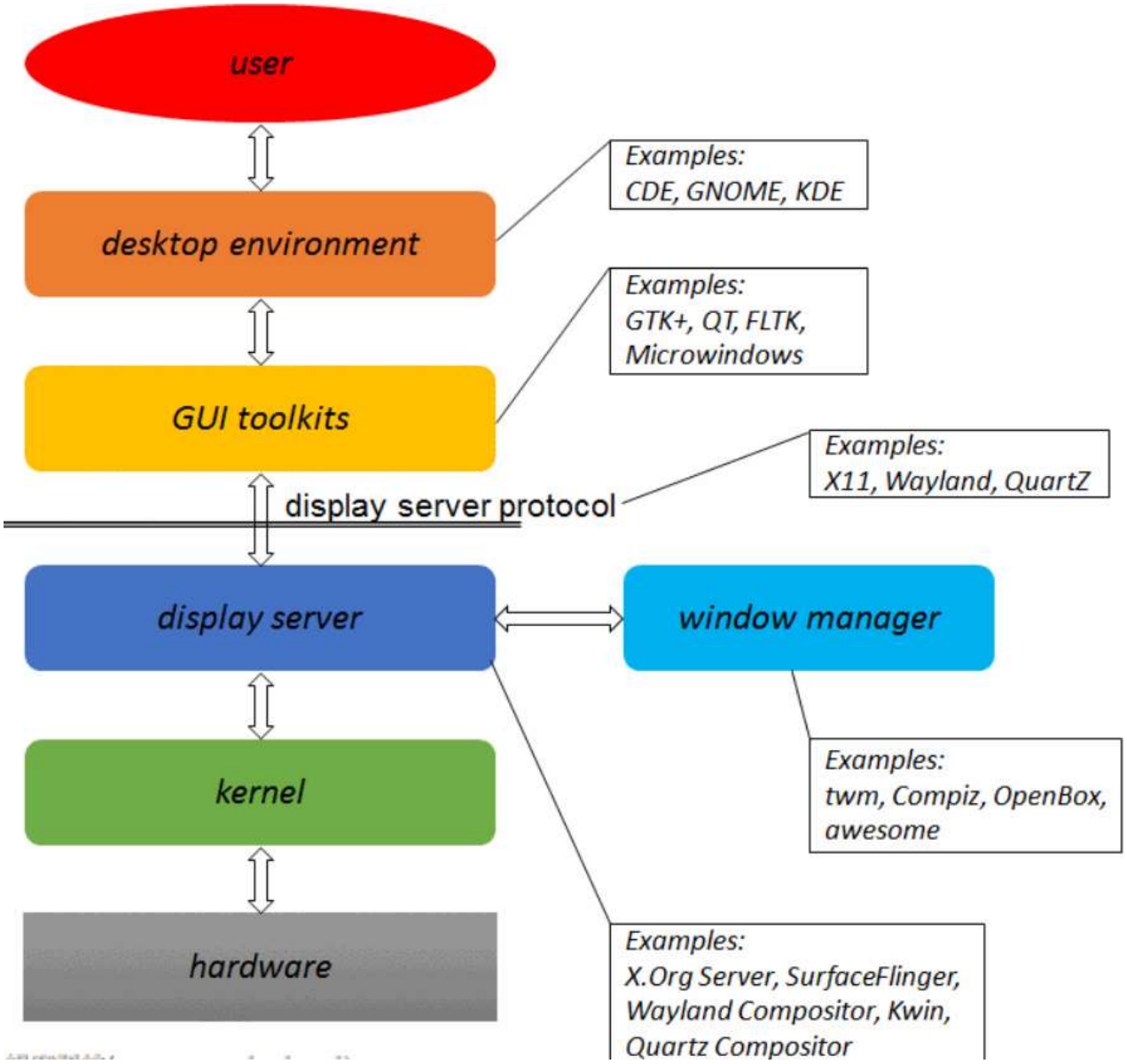

图 5-1. Embedded Linux Graphics Stack Overview

5.1. X11

本节主要介绍 X Window(X11)。X 仅为 GUI 环境构建提供了基本的框架，在屏幕上绘图和移动窗口，以及与鼠标和键盘交互

在 mys-zu5ev-core 镜像中，提供了一个精简但适用于嵌入式系统的 Matchbox-desktop 桌面环境。

1）启动 matchbox-desktop

matchbox-desktop 桌面默认是开机自启动的，下面的示例会先关闭 matchbox 然后再启动

```
root@mys_zu5ev #/etc/init.d/xserver-nodm restart
Stopping XServer
xinit: connection to X server lost

waiting for X server to shut down dbus-daemon[922]: Reloaded configuration
(II) Server terminated successfully (0). Closing log file.

xinit: unexpected signal 15
Starting Xserver
```

5.2. QT

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，并且附带了大量的 demo。

1）获取 qt 的信息

首先查看当前系统支持的 QT 版本，如下：

```
root@mys_zu5ev  #ls  /usr/lib/*Qt5Core*
/usr/lib/libQt5Core.so.5  /usr/lib/libQt5Core.so.5.13   /usr/lib/libQt5Core.so.5.13.2
```

由上可知，Qt 的版本是 5.13.2，附带的 Qt 例程位于 /usr/share/examples 目录。

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 XCB。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。MYS-ZU5EV 开发板上默认使用的是 XCB 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@mys_zu5ev  #  cd  /usr/share/examples/
root@mys_zu5ev  #  export  DISPLAY=:0.0
root@mys_zu5ev  #  export  QT_QPA_PLATFORM  =  xcb
root@mys_zu5ev  #./widgets/touch/pinchzoom/pinchzoom
```

或者使用-platform xcb 指令平台插件，如下：

```
root@mys_zu5ev  #./widgets/touch/pinchzoom/pinchzoom  -platform  xcb
```

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 Linuxfb 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。

●输入外设配置

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
```

MYS-ZU5EV 开发板出厂的 mys-zu5ev-full 镜像带有演示程序，使用 xcb 平台插件和 EvdevTouch 触摸处理程序，对应启动命令程序如下：

```
root@mys_zu5ev # export QT_QPA_FB_HIDECURSOR=1
root@mys_zu5ev #./widgets/touch/pinchzoom/pinchzoom -platform xcb&
```

可以通过 kill 方式对进程直接退出

```
root@mys_zu5ev # killall pinchzoom
```

6. 多媒体应用

本章主要测试的是 MYS-ZU5EV 的多媒体应用，用户需根据开发板主芯片型号选择对应的章节进行评估测试。适用于 zynqMP 系列处理器的开发板，则按章节顺序测试。

6.1. Camera

本节采用系统自带 gstreamer，v4l2-utils，media-ctl 命令对米尔 USB 摄像头(型号 MY-CAM002U)进行评估测试。主要测试 USB 摄像头的预览、抓帧（拍照）。

1) 查看 USB 摄像头拓扑图和分辨率

●查看设备基本信息

插入 usb 摄像头，会生成字符设备/dev/media1：

```
root@mys_zu5ev # ls -l /dev/media1
crw-rw---- 1 root video 252, 1 Nov 27 08:43 /dev/media1
```

●查看设备拓扑

使用 media-ctl -d /dev/media1 -p 列出拓扑图如下：

```
root@mys_zu5ev # media-ctl -d /dev/media1 -p
Media controller API version 5.4.31

Media device information
------------------------
driver          uvcvideo
model           USB 2.0 HD Camera : USB 2.0 HD
serial
bus info        usb-5800d000.usbh-ehci-1.1
hw revision     0x0
driver version  5.4.31

Device topology
- entity 1: USB 2.0 HD Camera : USB 2.0 HD  (1 pad, 1 link)
             type Node subtype V4L flags 1
```

```
                device node name /dev/video1
        pad0: Sink
<- "Processing 2":1 [ENABLED,IMMUTABLE]

- entity 4: USB 2.0 HD Camera : USB 2.0 HD  (0 pad, 0 link)
                type Node subtype V4L flags 0
                device node name /dev/video2

- entity 8: Processing 2 (2 pads, 3 links)
                type V4L2 subdev subtype Unknown flags 0
        pad0: Sink
<- "Camera 1":0 [ENABLED,IMMUTABLE]
        pad1: Source
                -> "USB 2.0 HD Camera : USB 2.0 HD ":0 [ENABLED,IMMUTA
BLE]
                -> "Extension 6":0 [ENABLED,IMMUTABLE]

- entity 11: Extension 6 (2 pads, 1 link)
                type V4L2 subdev subtype Unknown flags 0
        pad0: Sink
<- "Processing 2":1 [ENABLED,IMMUTABLE]
        pad1: Source

- entity 14: Camera 1 (1 pad, 1 link)
                type V4L2 subdev subtype Unknown flags 0
        pad0: Source
                -> "Processing 2":0 [ENABLED,IMMUTABLE]
```

●使用 v4l2-ctl 查看/dev/video2 所支持的分辨率

```
root@mys_zu5ev # v4l2-ctl -D -d /dev/video2  --list-formats-ext
Driver Info:
        Driver name      : uvcvideo
        Card type        : USB 2.0 HD Camera : USB 2.0 HD
```

```
        Bus info          : usb-5800d000.usbh-ehci-1.1
        Driver version    : 5.4.31
        Capabilities      : 0x84a00001
                Video Capture
                Metadata Capture
                Streaming
                Extended Pix Format
                Device Capabilities
        Device Caps       : 0x04200001
                Video Capture
                Streaming
                Extended Pix Format
Media Driver Info:
        Driver name       : uvcvideo
        Model             : USB 2.0 HD Camera : USB 2.0 HD
        Serial            :
        Bus info          : usb-5800d000.usbh-ehci-1.1
        Media version     : 5.4.31
        Hardware revision: 0x00000000 (0)
        Driver version    : 5.4.31
Interface Info:
        ID                : 0x03000002
        Type              : V4L Video
Entity Info:
        ID                : 0x00000001 (1)
        Name              : USB 2.0 HD Camera : USB 2.0 HD
        Function          : V4L2 I/O
        Flags             : default
        Pad 0x01000007    : 0: Sink
          Link 0x02000010: from remote pad 0x100000a of entity 'Processing
2': Data, Enabled, Immutable
ioctl: VIDIOC_ENUM_FMT
        Type: Video Capture
```

```
        [0]: 'MJPG' (Motion-JPEG, compressed)
                Size: Discrete 1280x720
                        Interval: Discrete 0.040s (25.000 fps)
                Size: Discrete 640x480
                        Interval: Discrete 0.040s (25.000 fps)
                Size: Discrete 320x240
                        Interval: Discrete 0.040s (25.000 fps)
                Size: Discrete 160x120
                        Interval: Discrete 0.040s (25.000 fps)
                Size: Discrete 352x288
                        Interval: Discrete 0.040s (25.000 fps)
                Size: Discrete 176x144
                        Interval: Discrete 0.040s (25.000 fps)
        [1]: 'YUYV' (YUYV 4:2:2)
                Size: Discrete 1280x720
                        Interval: Discrete 0.067s (15.000 fps)
                Size: Discrete 640x480
                        Interval: Discrete 0.040s (25.000 fps)
                Size: Discrete 320x240
                        Interval: Discrete 0.040s (25.000 fps)
                Size: Discrete 160x120
                        Interval: Discrete 0.040s (25.000 fps)
                Size: Discrete 352x288
                        Interval: Discrete 0.040s (25.000 fps)
                Size: Discrete 176x144
                        Interval: Discrete 0.040s (25.000 fps)
```

从以上 log 可清楚的看到对应的分辨率下所支持的帧率，如分别率 320x240 所支持的帧率为 25fps。

2）摄像头窗口预览

●使用 gstream 工具来进行预览

使用 gstream 中的 gst-launch 命令进行摄像头的预览，终端执行以下命令：

```
root@mys_zu5ev #gst-launch-1.0 -v v4l2src device=/dev/video2 ! video/x-raw,
width=640,height=320,framerate=25/1 ! videoconvert ! autovideosink

Setting pipeline to PAUSED ...
Pipeline is live and does not need PREROLL ...
Setting pipeline to PLAYING ...
New clock: GstSystemClock

^Chandling interrupt.
Interrupt: Stopping pipeline ...
EOS on shutdown enabled -- Forcing EOS on the pipeline
Waiting for EOS...
Got EOS from element "pipeline0".
EOS received - stopping pipeline...
Execution ended after 0:03:12.080302175
Setting pipeline to PAUSED ...
Setting pipeline to READY ...
Setting pipeline to NULL ...
Freeing pipeline ...
```

执行完后，屏幕会生成一个宽为 640，高为 320，帧速率为每秒 25 帧的预览窗口。

7. 系统工具

默认映像中包含了一些常用的系统工具，便于用户在系统调试或实际部署的产品中查看和管理系统的各种资源，也可以在 SHELL 脚本或其他应用程序中调用。这些工具可能不完全满足用户的系统定制需求，此时系统开发人员需要根据实际情况做出适当的调整。

7.1. 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。

●语法格式

输入以下命令查看 tar 语法格式：

```
root@mys_zu5ev # tar --help
BusyBox v1.31.0 (2020-05-26 14:38:25 UTC) multi-call binary.

Usage: tar c|x|t [-ZzJjahmvokO] [-f TARFILE] [-C DIR] [-T FILE] [-X FILE] [FILE]...

Create, extract, or list files from a tar file

        c    Create
        x    Extract
        t    List
        -f FILE    Name of TARFILE ('-' for stdin/out)
        -C DIR    Change to DIR before operation
        -v    Verbose
        -O    Extract to stdout
```

```
-m  Don't restore mtime
-o  Don't restore user:group
-k  Don't replace existing files
-Z  (De)compress using compress
-z  (De)compress using gzip
-J  (De)compress using xz
-j  (De)compress using bzip2
-a  (De)compress using lzma
-h  Follow symlinks
-T FILE  File with names to include
-X FILE  File with glob patterns to exclude.
```

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，但不建议用在后台运行过程！
- -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！

●使用 tar 压缩

新建 test.txt 文件，并输入以下命令将文件打包成.gz 格式：

```
root@mys_zu5ev # tar -czf test.tar.gz test.txt
root@mys_zu5ev # ls
test.tar.gz     test.txt
```

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

●使用 tar 解压

把打包成 tar.gz 格式文件解压：

```
root@mys_zu5ev # tar -xvf test.tar.gz
test.txt
root@mys_zu5ev # ls
test.tar.gztest.txt
```

2）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。

●语法格式

在开发板终端输入以下命令查看 gzip 语法：

```
root@mys_zu5ev # gzip --help
BusyBox v1.31.1 () multi-call binary.
Usage: gzip [-cfkdt] [FILE]...
```

●用 gzip 把文件压缩

```
root@mys_zu5ev # gzip test.txt
root@mys_zu5ev # ls
test.txt.gz
```

●用 gunzip 工具解压

用 gunzip 把文件解压，示例如下：

```
root@mys_zu5ev # gunzip test.txt.gz
root@mys_zu5ev # ls
test.txt
```

7.2. 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@mys_zu5ev # 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 sd 卡第四分区示例如下：

```
root@mys_zu5ev # mount /dev/mmcblk1p4 /mnt/
[10677.124115] EXT4-fs (mmcblk1p4): mounted filesystem with ordered data mode. Opts: (null)
[10677.130984] ext4 filesystem being mounted at /mnt supports timestamps until 2038 (0x7fffffff)
```

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@mys_zu5ev # 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 sd 卡第 7 个分区示例如下：

```
root@mys_zu5ev # mkfs -t ext3 -V -c /dev/mmcblk1p7
mkfs from util-linux 2.32.1
mkfs.ext3 -c /dev/mmcblk1p7
mke2fs 1.44.3 (10-July-2018)
/dev/mmcblk1p7 contains a ext4 file system labelled 'userfs'
        created on Thu Aug 13 04:32:02 2020
Proceed anyway? (y,N) y
Creating filesystem with 330532 1k blocks and 82656 inodes
Filesystem UUID: 46bd5bb8-1882-4a68-901a-e11b4e71924b
Superblock backups stored on blocks:
        8193, 24577, 40961, 57345, 73729, 204801, 221185
Checking for bad blocks (read-only test): done
Allocating group tables: done
Writing inode tables: done
Creating journal (8192 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@mys_zu5ev # fsck -a /dev/mmcblk1p7
fsck from util-linux 2.32.1
/dev/mmcblk1p7: clean, 11/82656 files, 20726/330532 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@mys_zu5ev # dumpe2fs -h
dumpe2fs 1.45.4 (23-Sep-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@mys_zu5ev # dumpe2fs /dev/mmcblk0p2
dumpe2fs 1.44.3 (10-July-2018)
Filesystem volume name:   <none>
Last mounted on:          <not available>
Filesystem UUID:          46bd5bb8-1882-4a68-901a-e11b4e71924b
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@mys_zu5ev # dumpe2fs /dev/mmcblk0p2 | grep -i "inode size"
dumpe2fs 1.44.3 (10-July-2018)
Inode size:               128
```

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@mys_zu5ev # dumpe2fs /dev/mmcblk0p2 | grep -i "block size"
dumpe2fs 1.44.3 (10-July-2018)
Block size:               1024
```

7.3. 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 等。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@mys_zu5ev # fdisk -h
Usage:
 fdisk [options] <disk>        change partition table
 fdisk [options] -l [<disk>] list partition table(s)
```

对 eMMC 进行分区：

```
root@mys_zu5ev # fdisk /dev/mmcblk1p1
Welcome to fdisk (util-linux 2.32.1).
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 0xd2dcd6ef.
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 的文件示例如下：

```
root@mys_zu5ev # time dd  if=/dev/zero of=ffmpeg1 bs=2M count=1 conv=f
sync
1+0 records in
```

```
1+0 records out
2097152 bytes (2.1 MB, 2.0 MiB) copied, 0.195515 s, 10.7 MB/s
real    0m 0.20s
user    0m 0.00s
sys     0m 0.05s
```

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@mys_zu5ev # 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@mys_zu5ev # du ffmpeg1
2048    ffmpeg1
root@mys_zu5ev # 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@mys_zu5ev # 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@mys_zu5ev # df -i
Filesystem      Inodes IUsed  IFree IUse% Mounted on
devtmpfs         37543   477  37066    2% /dev
/dev/mmcblk1p6 289600 25068 264532    9% /
tmpfs            54499    17  54482    1% /dev/shm
tmpfs            54499   704  53795    2% /run
tmpfs            54499    14  54485    1% /sys/fs/cgroup
tmpfs            54499    22  54477    1% /tmp
/dev/mmcblk1p4   16384    24  16360    1% /boot
/dev/mmcblk1p5    4096    34   4062    1% /vendor
tmpfs            54499    26  54473    1% /var/volatile
/dev/mmcblk1p7   65536   438  65098    1% /usr/local
tmpfs            54499    10  54489    1% /run/user/0
```

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。

7.4. 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@mys_zu5ev # ps --help
BusyBox v1.29.3 (2021-03-31 10:39:30 CST) multi-call binary.

Usage: ps [-o COL1,COL2=HEADER]

Show list of processes

        -o COL1,COL2=HEADER     Select columns for display
```

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示例如下：

```
root@mys_zu5ev # ps aux
PID   USER     COMMAND
    1 root     init
    2 root     [kthreadd]
    3 root     [rcu_gp]
    4 root     [rcu_par_gp]
    5 root     [kworker/0:0-eve]
    6 root     [kworker/0:0H]
    7 root     [kworker/u8:0-ev]
    8 root     [mm_percpu_wq]
    9 root     [ksoftirqd/0]
   10 root     [rcu_sched]
   11 root     [migration/0]
   12 root     [cpuhp/0]
   13 root     [cpuhp/1]
   14 root     [migration/1]
   15 root     [ksoftirqd/1]
   16 root     [kworker/1:0-pm]
   17 root     [kworker/1:0H-kb]
   18 root     [cpuhp/2]
   19 root     [migration/2]
   20 root     [ksoftirqd/2]
   21 root     [kworker/2:0-eve]
   22 root     [kworker/2:0H-kb]
   23 root     [cpuhp/3]
   24 root     [migration/3]
   25 root     [ksoftirqd/3]
```

```
 26 root      [kworker/3:0-eve]
 27 root      [kworker/3:0H-kb]
 28 root      [kdevtmpfs]
 29 root      [netns]
 30 root      [kauditd]
 31 root      [kworker/1:1-eve]
 32 root      [oom_reaper]
 33 root      [writeback]
 34 root      [kcompactd0]
 35 root      [khugepaged]
 37 root      [kworker/3:1-eve]
 38 root      [kworker/0:1-eve]
 39 root      [kworker/u8:1]
 85 root      [kworker/2:1-eve]
 89 root      [kblockd]
 90 root      [blkcg_punt_bio]
 91 root      [edac-poller]
 92 root      [watchdogd]
 93 root      [rpciod]
 94 root      [kworker/u9:0]
 95 root      [xprtiod]
 96 root      [cfg80211]
 97 root      [kswapd0]
 98 root      [ecryptfs-kthrea]
 99 root      [nfsiod]
102 root      [spi0]
104 root      [sdhci]
105 root      [irq/32-mmc0]
106 root      [sdhci]
107 root      [kworker/1:2-rcu]
108 root      [irq/33-mmc1]
109 root      [kworker/1:3-pm]
116 root      [ion_system_heap]
```

```
117 root      [kpktgend_0]
118 root      [kpktgend_1]
119 root      [kpktgend_2]
120 root      [kpktgend_3]
121 root      [ipv6_addrconf]
122 root      [krfcommd]
123 root      [mmc_complete]
124 root      [kworker/0:2-eve]
125 root      [kworker/1:1H-kb]
126 root      [kworker/0:1H-mm]
127 root      [irq/38-fd4a0000]
128 root      [mmc_complete]
129 root      [kworker/0:3+eve]
130 root      [kworker/0:4-eve]
136 root      [kworker/3:1H-kb]
137 root      [jbd2/mmcblk0p1-]
138 root      [ext4-rsv-conver]
139 root      [kworker/0:5-pm]
151 root      [kworker/1:2H]
153 root      [kworker/2:1H-kb]
156 root      /sbin/syslogd  -n
160 root      /sbin/klogd  -n
171 root      [jbd2/mmcblk1p2-]
172 root      [ext4-rsv-conver]
193 dbus       dbus-daemon  --system
206 root      /usr/sbin/connmand  -n
209 root      [kworker/3:2H]
210 root      [kworker/2:2H]
219 root      /usr/sbin/sshd  -D
221 root      -sh
226 root      ps  aux
```

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@mys_zu5ev # top -help
top: invalid option -- 'h'
BusyBox v1.29.3 (2021-03-31 10:39:30 CST) multi-call binary.

Usage: top [-b] [-nCOUNT] [-dSECONDS]
```

动态查看系统进程示例如下：

```
root@mys_zu5ev # top
Mem: 65704K used, 3965760K free, 80K shrd, 2304K buff, 9748K cached
CPU:   0% usr   0% sys   0% nic  99% idle   0% io   0% irq   0% sirq
Load average: 0.00 0.01 0.02 1/81 231
  PID  PPID USER     STAT   VSZ %VSZ %CPU COMMAND
  219     1 root     S     6116   0%   0% /usr/sbin/sshd -D
  206     1 root     S     4836   0%   0% /usr/sbin/connmand -n
  221     1 root     S     3800   0%   0% -sh
  231   221 root     R     3224   0%   0% top
    1     0 root     S     3088   0%   0% init
  156     1 root     S     3088   0%   0% /sbin/syslogd -n
  160     1 root     S     3088   0%   0% /sbin/klogd -n
  193     1 dbus     S     2936   0%   0% dbus-daemon --system
  127     2 root     SW       0   0%   0% [irq/38-fd4a0000]
```

```
  31     2 root     IW       0   0%   0% [kworker/1:1-eve]
 129     2 root     IW       0   0%   0% [kworker/0:3-eve]
 117     2 root     SW       0   0%   0% [kpktgend_0]
 118     2 root     SW       0   0%   0% [kpktgend_1]
 119     2 root     SW       0   0%   0% [kpktgend_2]
 120     2 root     SW       0   0%   0% [kpktgend_3]
 107     2 root     IW       0   0%   0% [kworker/1:2-rcu]
 124     2 root     IW       0   0%   0% [kworker/0:2-eve]
 227     2 root     IW       0   0%   0% [kworker/3:2-eve]
 228     2 root     IW       0   0%   0% [kworker/3:0-eve]
   2     0 root     SW       0   0%   0% [kthreadd]
```

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@mys_zu5ev # 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
```

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@mys_zu5ev # 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 200640  13556 131852    0    0    39    11  148  227  6  3 90  0  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如下：

```
root@myir:~# vmstat -s
        435992 K total memory
         90008 K used memory
         82380 K active memory
         88924 K inactive memory
        200528 K free memory
         13556 K buffer memory
        131900 K swap cache
             0 K total swap
             0 K used swap
             0 K free swap
         20273 non-nice user cpu ticks
          4412 nice user cpu ticks
         19691 system cpu ticks
```

```
630164 idle cpu ticks
   241 IO-wait cpu ticks
     0 IRQ cpu ticks
    22 softirq cpu ticks
     0 stolen cpu ticks
127150 pages paged in
 36195 pages paged out
     0 pages swapped in
     0 pages swapped out
969112 interrupts
1492562 CPU context switches
1581090644 boot time
 10649 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。

4) 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@mys_zu5ev # ps -ef | grep mxapp2
root        1045       1 15 15:51 ?      00:00:02 /home/mxapp2 -platform eglfs
root        1174    1117  0 15:51 ttySTM0   00:00:00 grep mxapp2
```

然后输入以下命令终止进程：

```
root@mys_zu5ev # kill 1045
```

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@myir:~# killall mxapp2
root@myir:~#
```

8. 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，请参考《MYS-ZU5EV SDK 发布说明》。

8.1. 开发语言

●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)

这里重点介绍，也是当前 SDK 默认支持的 bash。下面示例为 mys-zu5ev-full 镜像中 Matchbox-desktop 桌面开机自启动的脚本 xserver-nodm 内容如下：

```
#!/bin/sh
#
### BEGIN INIT INFO
# Provides: xserver
# Required-Start: $local_fs $remote_fs dbus
# Required-Stop: $local_fs $remote_fs
# Default-Start:        5
# Default-Stop:         0 1 2 3 6
### END INIT INFO

killproc() {                # kill the named process(es)
        pid=`/bin/pidof $1`
        [ "$pid" != "" ] && kill $pid
}
```

```
read CMDLINE < /proc/cmdline
for x in $CMDLINE; do
        case $x in
        x11=false)
           echo "X Server disabled"
           exit 0;
                ;;
        esac
done

case "$1" in
  start)
        . /etc/profile

        #default for USER
        . /etc/default/xserver-nodm
        echo "Starting Xserver"
        if [ "$USER" != "root" ]; then
            # setting for rootless X
            chmod o+w /var/log
            chmod g+r /dev/tty[0-3]
            # hidraw device is probably needed
            if [ -e /dev/hidraw0 ]; then
                chmod o+rw /dev/hidraw*
            fi
        fi

        # Using su rather than sudo as latest 1.8.1 cause failure [YOCTO #121
1]
        su -l -c '/etc/xserver-nodm/Xserver &' $USER
        # Wait for the desktop to say its finished loading
        # before loading the rest of the system
        # dbus-wait org.matchbox_project.desktop Loaded
```

```
  ;;

  stop)
        echo "Stopping XServer"
        killproc xinit
        sleep 1
        chvt 1 &
  ;;

  restart)
       $0 stop
        $0 start
  ;;

  *)
        echo "usage: $0 { start | stop | restart }"
  ;;
esac

exit 0
```

●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 Petalinux 构建的 SDK，安装步骤请参《**MYS-ZU5EVLinux 软件开发指南**》，安装完成后需要配置一下 SDK 环境，如下：

```
PC$ $CC --version
aarch64-xilinx-linux-gcc (GCC) 9.2.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PU
RPOSE.
```

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用 C 编写的演示程序 hello-CC.c：

```
//file:   hello-CC.c
#include<stdio.h>
 int main(int argc,char *argv[])
 {
       printf("hello world!\n");
       return 0;
 }
```

C++编写的演示程序 hello-CXX.cpp

```
//file: hello-CXX.cpp
#include <iostream>
using namespace std;
int main(int argc,char *argv[])
{
          cout << "hello world!";
             return 0;
}
```

接着编译应用程序，这里 c 语言用$CC 编译，c++用$CXX 编译，因为编译的时候需要对应的头文件和链接，$CC/$CXX 包含有对应的系统库和配置信息，如果直接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。

```
$CC -v hello-CC.c -o hello-CC
or
$CXX -v hello-CXX.cpp -o hello-CXX
.............
```

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@mys_zu5ev # ./hello-CC
hello world!
or
root@mys_zu5ev # ./hello-CXX
hello world!
```

更复杂的示例和开发方式请参考《**MYS-ZU5EVLinux** 软件开发指南》应用移植部分的说明。

● **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 python3 版本

```
root@mys_zu5ev # python3 –version
Python 3.7.6
```

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@mys_zu5ev # python3
Python 3.7.6 (default, Feb 25 2020, 10:39:28)
[GCC 9.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print ("Hello, Python!")
```

在 Python 3.7.6 版本中,以上实例输出结果如下：

```
Hello, Python!
>>>
```

退出 Python 命令行，执行 exit()即退出 Python：

```
>>> exit()
root@mys_zu5ev #
```

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@mys_zu5ev # vi test.py
root@mys_zu5ev # cat test.py
#!/usr/bin/python3
print ("Hello, Python!")
```

执行脚本文件，Python3 解释器在/usr/bin 目录中，使用以下命令执行脚本。

```
root@mys_zu5ev # chmod +x test.py
```

```
root@mys_zu5ev # ./test.py
Hello, Python!
```

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。当前系统不支持 pip 命令与 pip 安装 python 包，如果客户需要 pip 工具需要自行移植。

8.2. 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。

●System SQLite

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，在嵌入式系统中，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，MYS-ZU5EV 使用的是 SQlite 3，下面进行简单测试。

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@mys_zu5ev # sqlite3 testDB.db
SQLite version 3.25.3 2018-11-05 20:37:38
Enter ".help" for usage hints.
```

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: /root/testDB.db
sqlite>
```

使用 .quit 命令退出 sqlite 提示符，如下所示：

```
sqlite> .quit
root@mys_zu5ev #
```

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

9. 参考资料

- **Xilinx 官方 WiKi**

  https://xilinx-wiki.atlassian.net/wiki/home

- **Petalinux2020.1 版本使用手册**

  https://www.xilinx.com/support/documentation/sw_manuals/xilinx2020_1/ug1144-petalinux-tools-reference-guide.pdf

- **嵌入式 Linux 下的 Qt**

  https://doc.qt.io/qt-5/embedded-linux.html

- **X.org Wiki**

  https://www.x.org/wiki/

- **网络配置**

  https://wiki.debian.org/NetworkConfiguration

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 18924653967
地址：深圳市龙岗区坂田街道发达路云里智能园 2 栋 6 楼 604 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：021-62087019 18924632515
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 13316862895
地址：北京市昌平区东小口镇中滩村润枫欣尚 1 号楼 505

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：0755-22316235（深圳），027-59621647/027-59621648（武汉）
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

凡是通过米尔电子直接购买或经米尔电子授权的正规代理商处购买的米尔电子全系列产品，均可享受以下权益：

1、6 个月免费保修服务周期
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、无产品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，我们会与用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。