《嵌入式系统》大作业实验报告：Part 2

陈俊哲　2020010964
梁　烨　2020080093
田正祺　2020080095

1　实验内容

实验背景：本次实验的硬件系统采用 MPU 和 MCU 双平台设计，其中 MPU 用于 Linux 操作系统开发，MCU 用于 ARM 体系结构与裸机编程。本次实验会在 Linux 系统下进行。

实验前置知识：本次实验需要使用 C 语言来编写文件读取函数，并利用虚拟机交叉编译代码后，在开发平台上运行。因此需要提前掌握一定的 C 语言基础。

实验目标：

2　实验部署

程序的测试是在运行 Ubuntu 20.04 虚拟机的 Windows 主机上进行。Ubuntu 中需要装测试以及交叉编译的依赖，可以通过运行 `setup-ubuntu.sh` 安装依赖。我们没有使用提供的交叉编译链。

开发板上需要将 STM32MP157 芯片启动拨码设为 EMMC 启动方式，即 101 状态，并插好电源开机。

开发板与主机可以直接通过以太网线链接（见章节 5），或通过以太网线链接到路由器。连到路由器的优点是可以通过外网访问，让所有组员可以在线上合作。开发板与主机之间的文件拷贝以及命令执行通过 Ubuntu 自带的 SCP 以及 SSH（没有使用 XShell 或 Xftp）（SSH 设置见章节 3.2）。若开放给公网建议在开发板上设置公钥认证（见章节 5）。

3　实验过程

3.1　源代码

Part 2 是在 Part 1 的 `waveheader.h`、`audioplayer.h`、`audioplayer.c`、`main.c` 的基础上开发的。之前的代码在 Part 1 已经解释了所以不再阐述，而新添加的代码如下：

- `audioplayer.h`：在 `AudioPlayer` 结构体中添加了保存播放器状态的变量，包括目前的音量、播放线程、播放/暂停状态、目前的时间戳、播放速度、是否重复播放等等。由于 Part 2 对一些功能没有要求，所以许多变量目前没有起到作用。
- `audioplayer.c`：主要实现了以下函数：
  - `ap_play`：开始播放。其中调用了 `alsa-lib` 的以 `snd_pcm` 开头的函数。播放可以在新线程中运行，以免阻塞 UI 线程。
  - `ap_pause`：暂停播放，目前的实现没有保存时间戳，所以再次调用 `ap_play` 会从头开始播放。
  - `ap_set_volume`：改变播放音量。其中调用了 `alsa-lib` 的以 `snd_mixer` 开头的函数。
- `audioplayer_tui.c`：通过 `ap_tui` 函数实现了文本用户界面。通过输入命令可以打开 WAVE 文件、播放/暂停、改变音量、设置重复播放、输出 WAVE 文件以及播放信息、以及退出程序。

- `main.c`：调用 `ap_tui`。

3.2 编译与运行

测试编译（`make debug`）以及交叉编译（`make xc`）的命令都在 `Makefile` 中定义。使用的交叉编译器是 `arm-linux-gnueabihf-gcc`。需要链接两个库：用于播放声音的 `libasound` 以及用于实现多线程的 `pthread`。在 Ubuntu 测试系统上使用仅仅使用 `-lasound` 以及 `-lpthread` 编译参数。但是在开发板运行需要为 ARM CPU 架构编译的库。可以使用 `make alsa` 下载 `alsa-lib` 源代码并编译动态链接库，或者使用 `make copy-libs` 将开发板上已有 `libasound.so` 以及 `libgthread.so` 的动态链接库拷贝到主机。

将可执行文件拷贝到开发板上用 `make scp`，需要保证 `SXX_KEY`、`SXX_PORT`、`SXX_HOST` 变量与实际情况一致。在主机上和在开发板上正常运行程序的命令分别为 `make drun` 和 `make xrun`。

4 实验结果

以下是程序的操作流程。类似的演示在 `demo.mp4` 中所示。

```
root@myir:~# ./audioplayer-arm
Audio Player
o: open file
i: print file and playback info
p: toggle play/pause
r: toggle repeat
h: print this help
v: set volume
q: quit
> i
No file opened
> o
Filename: doesnotexist
Error: Cannot open file
> o
Filename: example.wav
Opened
> i
RIFF chunk
    riff_id      RIFF
    chunk_size   2665816
    format       WAVE
Format chunk
    format_id    fmt
    format_size  16
    audio_format 1
    channels     2
    sample_rate  44100
    byte_rate    176400
    block_align  4
    bit_depth    16
Data chunk
    data_id      data
    data_size    2665676

Playing   no
Timestamp 0.00s
Speed     0.0x
Repeating no
> p
Playing
> v
Volume: 50
Volume set to: 50
> v
Volume: 100
Volume set to: 100
> q
root@myir:~#
```

5 实验心得

主要的开发环境以及与开发板通讯的问题已经在 Part 1 中解决了。Part 2 实验过程中遇到的问题及其解决方法如下：

- 编译 Alsa:之前不知道需要的动态链接库已存放在开发板上的 `/usr/lib` 中,所以从 `alsa-lib` 源代码编译了 `libasound.so`。编译命令用 `make alsa` 运行,参考了 。编译时需要设定目标平台类型。为了得到开发板的平台信息,需要将 `config.guess` 拷贝到开发板上并运行,得到 `armv7l-unknown-linux-gnueabihf` 由于提前发现 `libgthread.so` 在开发板上的存在，因此没有试图从源代码编译 `libgthread`
- 音量调节：遇到了改变音量但是播放的音量没有变化的问题。通过课程微信群中的同学的提示，发现读出的音量范围比实际范围大了一百倍。因此将音量再除以一百就解决了问题。