

《嵌入式系统》大作业实验报告：Part 1

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：

1. 使用系统 I/O 函数读取 WAVE 音频文件，并将 WAVE 音频文件的参数输出到命令行。
2. 将读取音频文件的参数写入开发平台上的文本文件中。注：文件操作需经过交叉编译并在开发平台上运行。

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

程序由以下四个源代码文件组成：

- `waveheader.h`: 定义了 Waveform Audio File Format (WAVE) 的头格式。参考的标准：
<https://datatracker.ietf.org/doc/html/draft-ema-vpim-wav-00>。
- `audioplayer.h`: 包含函数声明，以及定义了保存音频播放器的状态的结构体 `AudioPlayer`。
- `audioplayer.c`: 实现了以下函数：
 - `ap_open`: 读入 WAVE 文件并将信息写入 `AudioPlayer`。
 - `ap_get_header_string`: 格式化 WAVE 文件的参数并写入字符串。

- `ap_print_header`: 将 `ap_get_header_string` 产生的字符串输出到命令行。
 - `ap_save_header`: 将 `ap_get_header_string` 产生的字符串写入文件。
 - `ap_close`: 释放 `ap_open` 所占用的资源。
- `main.c`: 处理命令行参数, 调用 `AudioPlayer` 相关的函数, 输出信息以及错误。

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-gnueabi-gcc`。由于 Part 1 没有使用第三方库, 所以不需要做额外的链接。

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`。

`Makefile` 中有些注释掉的命令是为了编译以及链接 Part 2 中的 `ALSA` 库, 可以忽略。

4 实验结果

```
root@myir:~# ./audioplayer-arm
Usage: ./audioplayer <filename>
root@myir:~# ./audioplayer-arm does-not-exist
Cannot open file
root@myir:~# ./audioplayer-arm audioplayer-arm
Invalid wave file
root@myir:~# ./audioplayer-arm test.wav
RIFFChunk
  ID      RIFF
  Size    589860
  Format   WAVE
FormatChunk
  ID      fmt
  Size    16
  AudioFormat 1
  NumChannels 2
  SampleRate 44100
  ByteRate 176400
  BlockAlign 4
  BitsPerSample 16
DataChunk
```

```
ID      data
Size    589824
Saved header to test.wav.txt
root@myir:~# cat test.wav.txt
RIFFChunk
  ID      RIFF
  Size    589860
  Format   WAVE
FormatChunk
  ID      fmt
  Size    16
  AudioFormat 1
  NumChannels 2
  SampleRate 44100
  ByteRate 176400
  BlockAlign 4
  BitsPerSample 16
DataChunk
  ID      data
  Size    589824
```

从以上在开发板上运行的结果可见此程序有以下功能:

- 提示程序使用方式
- 提示文件不可读
- 提示文件不是格式正确的 `WAVE` 文件
- 输出 `WAVE` 参数
- 将 `WAVE` 参数写入文件

5 实验心得

在实验过程中遇到的问题及其解决方法如下：

- 当开发板直接通过以太网线连主机，没有自动配置 IP 地址。可以在开发板以及主机上手动配置 IP。在开发板上运行：`ifconfig eth0 192.168.2.2`，并且在主机上如下设置：

- IPv4 address: 192.168.2.1
- Gateway: 192.168.2.0
- Subnet mask: 255.255.255.0

当开发板连路由器，会有 DHCP 自动配置 IP 地址。

- 原本使用的虚拟机 Ubuntu 22.10 交叉编译的可执行文件在开发板上运行时出现错误：

```
./audioplayer: /lib/libc.so.6: version `GLIBC_2.34' not found (required by ./audioplayer)
```

由于主机的 glibc 的版本大于开发板上的 glibc 版本，所以在主机上交叉编译的可执行文件在开发板上找不到需要的库。使用 gcc 的 `-static` 编译参数进行静态链接可以解决此问题，但是在预先调研如何链接 ALSA 的 `libasound` 库时遇到了静态链接产生的问题。因此决定使用动态链接并降低主机的 glibc 版本，即从 Ubuntu 22.10 切换到 Ubuntu 20.04。

- 开发板启动后，操作系统会默认开启 `mxapp2` 程序。在 Linux 系统上，通常输入 `Ctrl+Alt+Fn` 会切换到命令行，但是 `mxapp2` 似乎禁用了此功能。将 `/home/mxapp2` 重命名到任何其他名字，比如 `mxapp2.disabled` 可以禁止 `mxapp2` 的启动。开发板会直接进入 Weston 界面，可以访问 Weston 自带的终端模拟器，也可以使用 `Ctrl+Alt+Fn` 切换到纯文本命令行。这对 Part 1 没有影响因为 Part 1 仅仅需要 SSH 运行程序，但 Part 3 需要开发图形界面，所以预先解决了这个问题。

- 开发板上的 SSH 服务器软件是 Dropbear 而不是熟悉的 OpenSSH，则公钥认证设置不同。大概的设置步骤如下：

1. 使用 `dropbearkey` 生成公密钥并存放在 `/etc/dropbear/dropbear_rsa_host_key`。
2. 使用 `dropbearconvert` 将 Dropbear 格式的密钥转换成为 Ubuntu 使用的 OpenSSH 格式的密钥，并将密钥拷贝到 Ubuntu。
3. 修改开发板上的 `/etc/default/dropbear` 文件，为了开启公钥认证内容应该是 `DROPBEAR_EXTRA_ARGS=" -s"`。
4. 重启开发板