中期检查报告：音频播放

大作业-8：杨碧茹（组长），董文冲，闵安娜，林智鑫

一、已选题目调研

xv6

xv6 是一个类 Unix 操作系统，有现代操作系统的基础框架和功能。

本组选题

本组选题是在 xv6 基础上实现音频播放功能。主要涉及音频文件的读入和解码、与音频驱动（声卡）的交互、播放过程中的控制功能，以及诸如启动程序、错误处理等配套功能。

核心难点

整个项目的重难点有以下几个方面：

1. 读入指定的音频文件；
2. 对音频文件按照格式进行解码；
3. 将解码后的音频正确使用声卡和音频驱动播放；
4. 在播放过程中，播放线程之外另外开启控制线程，接收用户的暂停等指令。

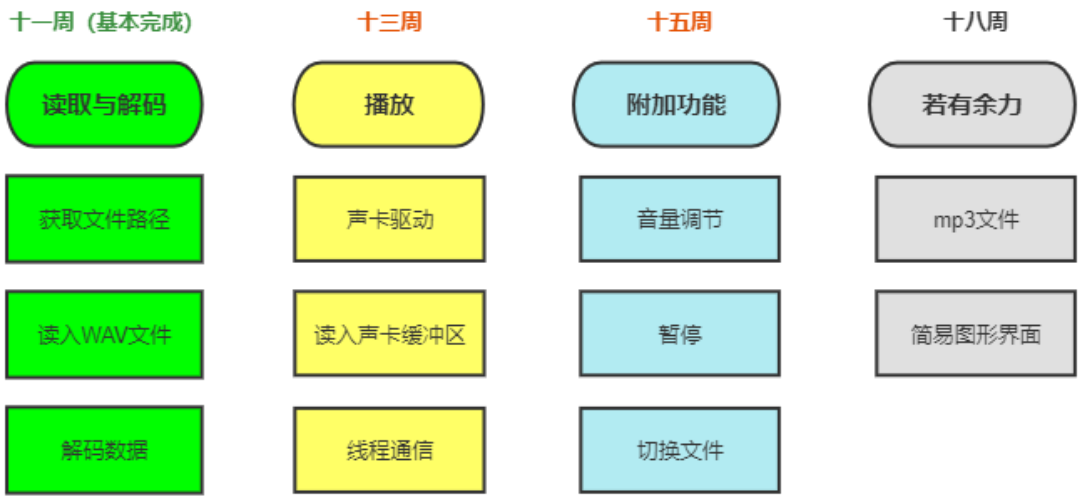
详细的功能点、技术路线将在下文中详细描述。

二、预期功能目标

本项目有以下几个预期功能目标：

- 音频播放器可以从 Shell 中启动，并接收播放文件名、额外选项等参数列表；
- 音频播放器可以定位并读入指定的音频文件，并支持至少一种音频类型(wav)的解码；
- 音频播放器可以将解码后的音频文件正确地通过音频驱动和声卡播放；
- 在播放过程中，可以进行暂停/继续、停止、变速、切换音频等基础操作；
- 对预料之外的用户输入和错误有一定的处理能力。

具体**开发时间表**如下：

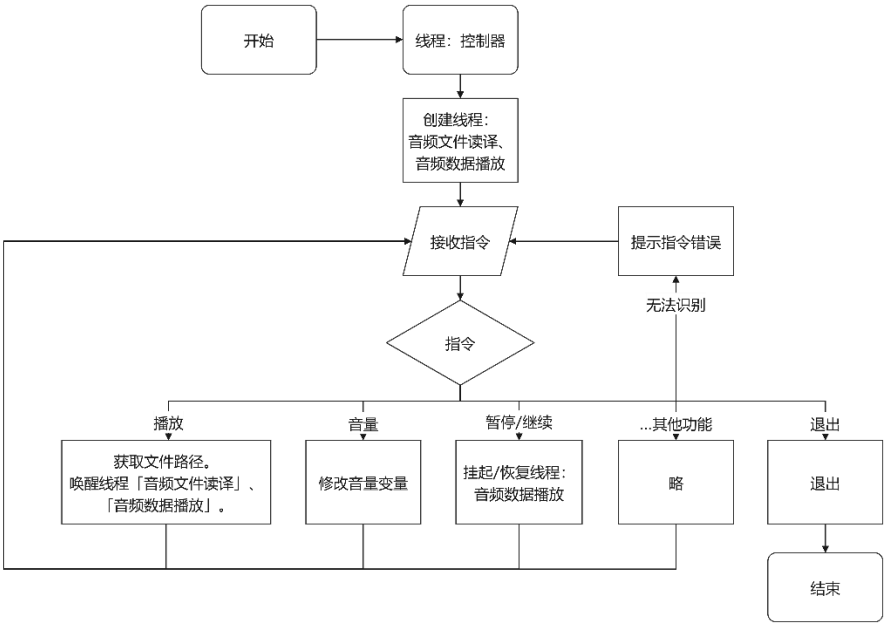


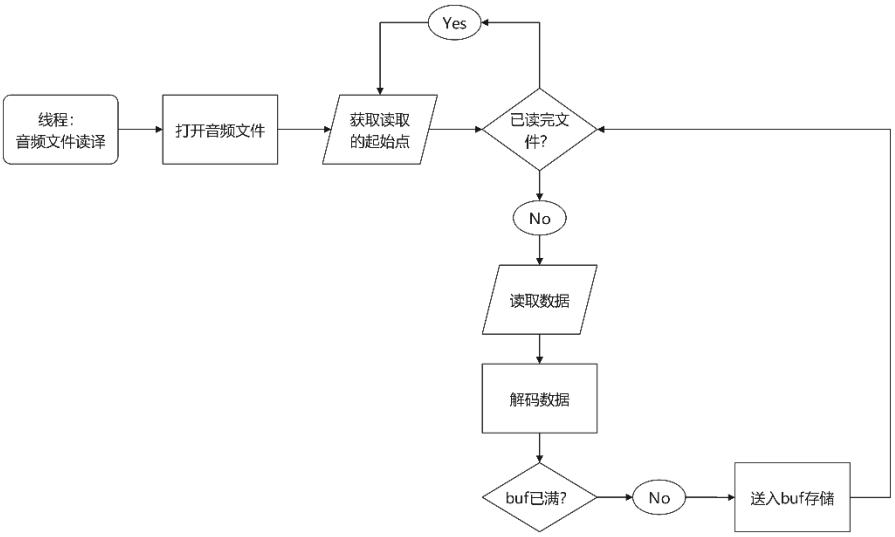
三、技术框架

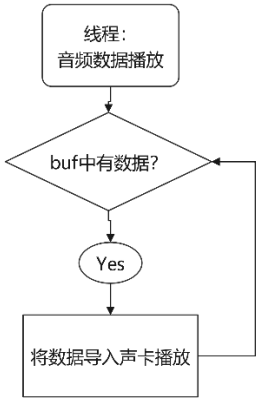
整体思路

使用 3 个线程实现需求。线程 1 负责读入音频文件并解码（音频文件读译）；线程 2 负责解码后的数据导入声卡播放（音频数据播放）；线程 3 负责接收外界指令，对线程 1、2 进行调控（控制器）。

流程图


开始
线程：控制器
创建线程：
音频文件读译、
音频数据播放
接收指令
提示指令错误
无法识别
指令
播放
音量
暂停/继续
...其他功能
退出
获取文件路径。
唤醒线程「音频文件读译」、
「音频数据播放」。
修改音量变量
挂起/恢复线程：
音频数据播放
略
退出
结束


Yes
线程：
音频文件读译
打开音频文件
获取读取
的起始点
已读完文
件?
No
读取数据
解码数据
buf已满?
No
送入buf存储


线程：
音频数据播放
buf中有数据?
Yes
将数据导入声卡播放

四、已实现功能目标

WAV 文件的读入

1. wav 文件以 RIFF 为基础，该格式以 Header、FormatChunk、DataChunk 三部分组成。
2. Header 首先四个字符是大端序 ChunkID，它总是 RIFF，指明其格式；之后是四字节小端序 ChunkSize，表示文件的总字节数 - 8，这个 magic number 是 ChunkID 与 ChunkSize 合占八字节，即 ChunkSize 表示其之后的所有字节的大小。之后是四字节大端序 Format，对于 wav 总是 WAVE。
3. FormatChunk 首先是四字节大端序 Subchunk1ID，其值总是 fmt，表示 FormatChunk 从此开始。在此后是四字节小端序 Subchunk1Size，表示 FormatChunk 总字节数 - 8，这个 8 的含义与 2 中相同。之后是二字节小端序 AudioFormat，对于 wav 总是 1。NumChannels 二字节小端序，表示总声道个数。SampleRate 四字节小端序，表示每个通道上每秒包含多少帧。ByteRate 四字节小端序，大小等于 SampleRate *BlockAlign，表示每秒含多少字节。BlockAlign 二字节小端序，等于 NumChannels* BitsPerSample / 8，表示每帧的多通道总字节数。BitsPerSample 二字节小端序，表示每帧包含多少比特。
4. DataChunk 首先是四字节大端序 Subchunk2ID，其值总是 data，表示 DataChunk 从此开始。在此后是四字节小端序 Subchunk2Size，表示 data 段的总字节数，也就是 DataChunk 的大小 - 8。之后是小端序 data，大小为 Subchunk2Size，表示音频波形的帧数据，各声道按帧交叉排列。

代码打印输出解析数据（此处暂时用 short 打印，测试文件为双声道）如下：

```
xv6 kernel is booting

init: starting sh
$ wav wav-music-1410kb.wav
buf[pos]  bytes
wh->chunk_size 1443038
getData: pos at 64,start pos at 64
getData:bits_per_sample 16

left channel:-5744 -26391 -24680 159 0 18688 17225 19779 21581 12116 559 2 0 12544 13873 13110
right channel:8243 27424 25963 31077 10361 17448 28484 28271 10094 29735 8308 27936 28525 25711 26980 26217
left channel:31078 10617 14889 19514 13900 13366 17972 21830 13141 22323 13399 22836 30809 22648 13144 23091
right channel:18010 21574 27988 25197 23138 11098 14379 12088 21039 14674 27961 20333 14415 22072 16726 17985
left channel:30022 28789 11120 13355 30772 25208 13666 27445 20331 25167 28770 16752 12097 18479 29512 18803
right channel:23113 14170 28471 20335 22863 22105 28246 28782 26992 20073 17486 20548 11088 28971 26481 12903
left channel:28978 22641 26968 19561 22348 12887 31282 13690 20277 25679 12900 27442 22891 22617 29528 27507
right channel:13675 20533 18256 27463 14187 21047 25938 18533 17992 29254 13938 11062 20267 26703 26472 28007
left channel:30829 13944 30006 21109 28242 29294 26482 28007 21101 29266 30322 19062 19786 28749 14704 11065
right channel:14635 22073 17494 24900 22113 19542 20044 17998 17990 17990 12358 17456 13380 17972 23110 28762
left channel:23152 14170 22839 12633 27185 30826 18296 13895 12086 27695 25452 19043 23114 18522 27464 14443
right channel:17208 19011 12362 21808 31317 14202 13623 23093 18266 26951 24937 11105 19755 19277 21067 21330
left channel:20819 18513 19016 28234 19054 23114 23130 13658 16693 23105 25178 20322 17743 13381 24884 19297
right channel:26955 25705 25700 26468 30823 19320 14155 20279 29007 21873 21333 11091 12587 21297 26963 31081
left channel:21881 12373 22064 20566 23120 30298 30326 26230 18790 12105 27183 20586 13136 29747 20084 19534
right channel:19020 11082 21803 11093 25131 28770 18032 30790 13944 21558 27988 28781 28784 14192 13879 20278
left channel:25167 14434 13112 18483 27976 12909 17714 3909 15 0 0 0 0 0 0 0
right channel:0 0 0 0 0 0 0 1280 1285 1285 1285 1285 1285 1285 1285 1285
left channel:1285 1285 1285 1285 1285 1285 1285 1285 1285 1285 1285 1285 1285 1285 1285 1285
right channel:1285 1285 1285 1285 1285 1285 1285 14341 5176 20 0 0 0 0 0 -32000
```

修改 xv6 最大文件大小限制

文件读写

xv6 无法直接在 ubuntu 上读文件，要把文件读入 fs.img（文件系统映像）。

修改限制

WAV 文件大小计算公式是采样频率(kHz) x 采样位数 x 声道数 x 时间(秒) / 8 = 文件大小(kb)，如果采用如下的参数： 采样率：8kHz 采样位数：16 声道数： 2 ，那么：一分钟 WAV 文件的大小 = 8 x 16 x 2 x 60 / 8 = 1920KB , 可近似成 2M 计算。

而目前，xv6 文件被限制在 140 个扇区，即 71,680 字节。这个限制来自于一个 xv6 inode 包含 12 个“直接”块号("direct" block number)和一个“单间接”块号，这个块号指的是一个最多可以容纳 128 个块号的块，总数为 12+128=140。这对于播放 WAV 文件是不够的。

xv6 文件系统由 inode 组成，每个 inode 是单个未命名的文件。整个磁盘读写的最小单元为 block (xv6 为 512 字节)。 xv6 文件系统采用位图块来管理磁盘中的块，

每个块可以管理的大小为 BPB = BSIZE * 8，若该标志为 0，则块空闲，否则已经使用。其中整个磁盘分布如下图。

第 0 个 block 为启动区，第 1 个 block 为超级块 (也是根目录所在的块),接下来是连续分布的 dinode，最后是连续分布的 BPB(块位图)

假设块大小为 BSIZE, 则

1. 每个块包含的 dinode 结构体的数量为: IPB = BSIZE/sizeof(dinode)。等价于第 i 个 dinode 所在的 block 为 IBLOCK(i) = i/IPB+2
2. 第 i 个 block 所在的位图块为： BBLOCK(b, ninodes) = b/BPB+ninodes/IPB+3

默认的 xv6 直接分配的时候，文件尺寸的大小为 NDIRECT·BSIZE，索引分配的时候，文件尺寸的大小为 NINDIRECT·BSIZE ,其中 NINDIRECT = BSIZE/sizeof(uint)

以下有 3 种方法可以更改 xv6 文件系统代码（以防后续有冲突，都进行尝试，目前没有冲突），使每个 inode 中支持“双间接”块("doubly-indirect" block)，其中包含 128 个单间接块地址，每个单间接块最多可以包含 128 个数据块地址。其结果是，一个文件将能够由最多 16523 个扇区(或大约 8.5 mb)组成:11+128+128*128=16523。

1. 最直接的是改变 block size 大小和直接分配/索引分配时的设定大小。

2. 或者改变直接分配的模式，把 inode 结构体中 uint addrs[NDIRECT+1]中所有的索引都指向一个块（即都变成 INDIRECT 模式，这时可以最多支持(512/4) *(512/4)* 512 = 8M。

3. 或者在一级索引节点后增加二级索引节点，改变文件大小限制所在扇区数量。修改 bmap()，使它除了直接块和单间接块之外，还实现了双间接块，使每个 inode 中支持“双间接”块，其中包含 128 个单间接块地址，每个单间接块最多可以包含 128 个数据块地址。其结果是，一个文件将能够由最多 16523 个扇区(或大约 8.5 mb)组成:11+128+128*128=16523。

尚需修补的部分

1. 应当读取 BitsPerSample 比特（也就是 BitsPerSample / 8 字节大小，可以考虑开一个多个 short 的 union，或者干脆为了好写只允许它是特定的值）的数据作为一帧，每通道读取一帧之后统一送到某个地方，每秒重复这个过程 SampleRate 次。

2. 为了便于增加暂停/变速播放等功能，每个 batch 读取数据的大小应该设定新的函数来求。buf 区大小和每次 getData 的大小参数，后续还需进行修改测试。

3. 耳机声效，暂停等功能应该在 wav reader 和导入声卡 buf 之间加。

代码截图

```
#include "kernel/types.h"
#include "user/user.h"
#include "kernel/fcntl.h"

struct WaveHeader
{
    char chunk_id[4];
    uint chunk_size;
    char format[4];
    char fmt_chunk_id[4];
    uint fmt_chunk_size;
    ushort audio_fomat;
    ushort num_channels;
    uint sample_rate;
    uint byte_rate;
    ushort block_align;
    ushort bits_per_sample;
    char data_chunk_id[4];
    uint data_chunk_size;
    int num_frame;
    int start_pos;
};

void getHead(char *fname, struct WaveHeader *wh);
void getData(char *fname, struct WaveHeader *wh);
```

```
void getHead(char *fname, struct WaveHeader *wh)
{
    int fd = open(fname, O_RDONLY);
    char buf[HEAD_LENGTH];
    int n;
    if (fd < 0)
    {
        fprintf(2, "getHead: cannot open %s\n", fname);
        exit(1);
    }
    if ((n = read(fd, buf, HEAD_LENGTH)) > 0)
    {
        int pos = 0;
        // 寻找“RIFF”标记
        while (pos < HEAD_LENGTH)
        {
            if (buf[pos] == 'R' && buf[pos + 1] == 'I' && buf[pos + 2] == 'F' && buf[pos + 3] == 'F')
            {
                wh->chunk_id[0] = 'R';
                wh->chunk_id[1] = 'I';
                wh->chunk_id[2] = 'F';
                wh->chunk_id[3] = 'F';
                pos += 4;
                break;
            }
            ++pos;
        }
        wh->chunk_size = *(int *)&buf[pos];
        pos += 4;
        wh->format[0] = buf[pos];
        wh->format[1] = buf[pos + 1];
        wh->format[2] = buf[pos + 2];
        wh->format[3] = buf[pos + 3];
        pos += 4;

        //寻找“fmt”标记
        while (pos < HEAD_LENGTH)
        {
            if (buf[pos] == 'f' && buf[pos + 1] == 'm' && buf[pos + 2] == 't')
            {
```

五、目前遇到的问题和困难

1. 难以对声卡做调试或测试。我们甚至不知道添加虚拟声卡后如果操作系统播放了声音会输出到电脑的实际声卡还是怎么样。
2. 缺乏与硬件打交道的具体知识。需要通过总线传输数据、直接与声卡寄存器交互，并且需要查阅大量相关手册。

六、人员分工

调研与文档：全组

音频读入与解码：董文冲、闵安娜

声卡驱动与播放：杨碧茹、林智鑫