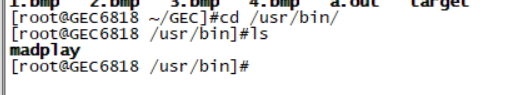
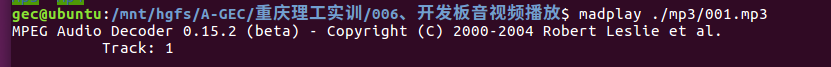
**一、播放音频**

首先查看开发板的 /usr/bin目录下是否存在madplay



madplay：是免费开源播放器，在Linux系统当中可以直接在命令行终端去控制音频

直接在命令终端使用：



在代码当中：

system：该函数在调用的时候，它会去调用fork函数，产生一个子进程，由子进程去执行system传递进来的终端命令

头文件：

#include <stdlib.h>

函数原型:

int system(const char \*command);

参数：

command：要执行的终端命令

举例：system（"ls -l"）

信号的发送：有kill跟killall两种方式发送，kill的方式去发送信号是根据进程号来决定的，kaillall的方式发送信号，是根据进程名字来决定的。

#include <stdio.h>

#include <stdlib.h>

int main(int argc, char const \*argv[])

{

    // system("ls -l\n");

    int n;

    while (1)

    {

        scanf("%d", &n);

        switch (n)

        {

            // 重定向，将所有的信息写入到文件 /dev/null 当中去，包括标准出错，并且程序在后台运行(&)

            // /dev/null：该文件简称为Linux黑洞，它会将写入到该文件仍和数据都丢弃，包括标准出错

            case 1: system("madplay ./mp3/001.mp3 \* < /dev/null > /dev/null 2>1&\n");  // 播放音乐

                break;

            // killall:表示发送一个信号

            // -STOP:停止信号

            case 2: system("killall -STOP madplay");  // 暂停音乐

                break;

            // -CONT：继续信号

            case 3: system("killall -CONT madplay");  // 继续播放

                break;

            // 在切换的时候，madplay正在播放其他音乐，需要先发送一个杀死信号，将madplay关闭

            // 关闭之后在播放新的音乐

            // -KILL：杀死

            case 4: system("killall -KILL madplay");  // 先杀死播放

                system("madplay ./mp3/002.mp3 \* < /dev/null > /dev/null 2>1&\n");  // 播放音乐

                break;

            case 5: system("killall -KILL madplay");  // 退出

                    return -1;

                break;

        }

    }

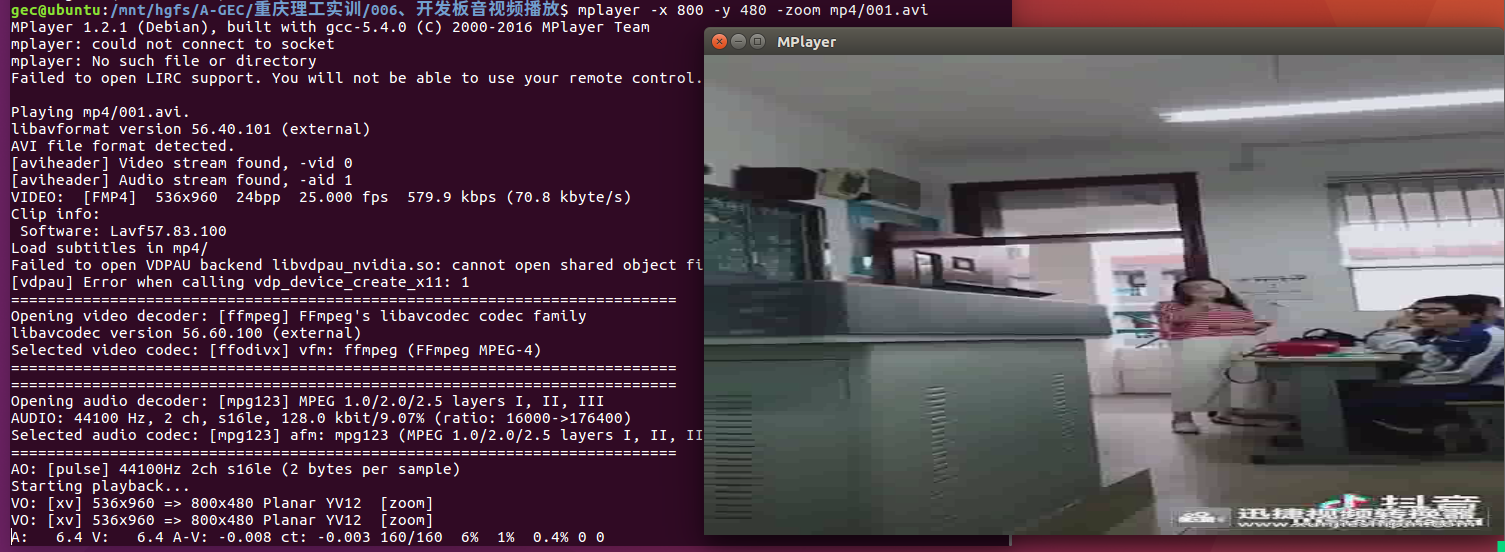
    return 0;

}

**二、播放视频**

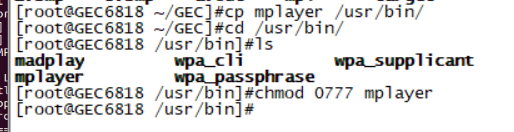
mplayer：使用方法跟madplay一模一样

指定画面大小：mplayer -x 800 -y 480 -zoom mp4/001.avi



移植开发板：

将mplayer下载到开发板的 /usr/bin 目录下面，并修改权限 （chmod）



在代码当中使用：

#include <stdio.h>

#include <stdlib.h>

int main(int argc, char const \*argv[])

{

    system("mplayer -x 800 -y 480 -zoom ./mp4/001.avi \* < /dev/null > /dev/null 2>1&\n");

    return 0;

}