# 学习文档

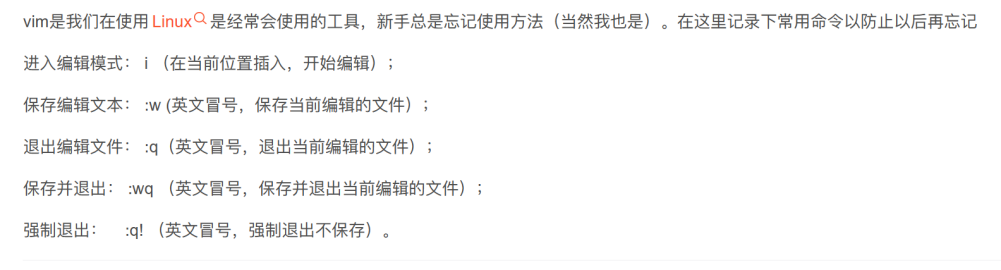
## Level -1 Linux

### 1.如何在Windows系统的电脑中使用Linux系统。

1. 安装Windows自带的Linux子系统；
2. 下载Ubuntu，同时通过powershell（以管理员模式运行）将wsl设置为默认。

### 2.需要用到的一些Linux指令。

1. 目录操作：cd(切换），ls（查看），mkdir（创建，若提示permission denied表示权限不够，需使用sudo），rm（删除）。
2. 文件操作:touch（创建），rm（删除），vim（编辑）。



Vim的常用命令

使用vim时若提示 readonly，可使用 ： set noreadonly改变权限。

## Level 0-1 git

1. 如何在Linux环境下通过命令行将项目托管到github
2. 安装sudo apt-get 自动从互联网的软件仓库中搜索、安装、升级、卸载软件或操作系统。

git：sudo apt-get install git

1. 配置github：在git服务器中，任意非本地git repo中，使用以下命令配置全局配置

git config --global user.name "kdurnd"

git config --global user.email ["2806881473@qq.com"](mailto:\"2806881473@qq.com\")

1. 创建本地仓库

git init <仓库名>

1. 将本地仓库与远程仓库建立联系

git remote add origin <https://github.com/<用户名>/<仓库名>.git>

1. 加入索引

git add <文件1>

git add<文件2> . . . . . .

(git add将内容写入暂存区)

Git commit -m [message]

（git commit 将暂存区内容添加到本地仓库）

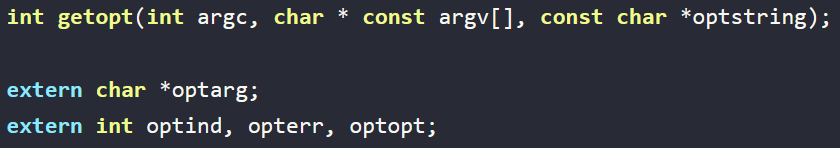
1. 将本地仓库内容推送到github仓库

git push origin master

输入该指令后，当提示输入用户名和密码时，注意密码非github登录密码，而是github令牌。在github用户->setting->Developers settings中可设置令牌。（务必记住令牌，否则下次使用需重新生成令牌）

## Level 0-2 Parsing parameters

查阅资料，发现库函数“getopt”可以帮助解析命令行参数



getopt的定义

getopt 参数说明：

argc：通常由 main 函数直接传入，表示参数的数量

argv：通常也由 main 函数直接传入，表示参数的字符串变量数组

optstring：一个包含正确的参数选项字符串，用于参数的解析。例如 “abc:”，其中 -a，-b 就表示两个普通选项，-c 表示一个必须有参数的选项，因为它后面有一个冒号。若有两个冒号，则为可选参数。例如“abc：：”，-c为可选参数。对于可选参数，所跟参数与选项中间不能有空格，而对于必选参数，则有无空格皆可。

外部变量说明：

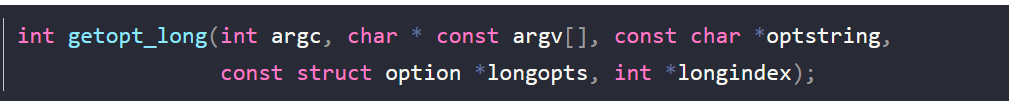
optarg：如果某个选项有参数，这包含当前选项的参数字符串

optind：argv 的当前索引值

opterr：正常运行状态下为 0。非零时表示存在无效选项或者缺少选项参数。

optopt：当发现无效选项字符时，即 getopt() 方法返回 ? 字符，optopt 中包含的就是发现的无效选项字符。

考虑到后续有长选项，改用getopt\_long



Getopt\_long的定义

前三个参数与getopt相同，longopts指明长参数的名字和属性，longindex为longopts的下标值。

## Level 0-3 Video decoder

Q：如何将在Windows环境下下载的头文件与静态库添加到Linux环境的目录下？

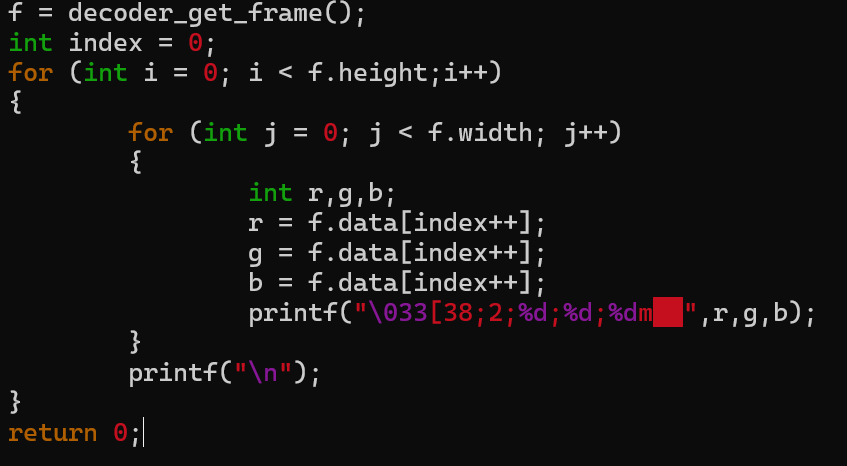
A：cd /mnt可查看Windows环境下的文件，再通过cp将其复制过去。

## Level 1-1Image print

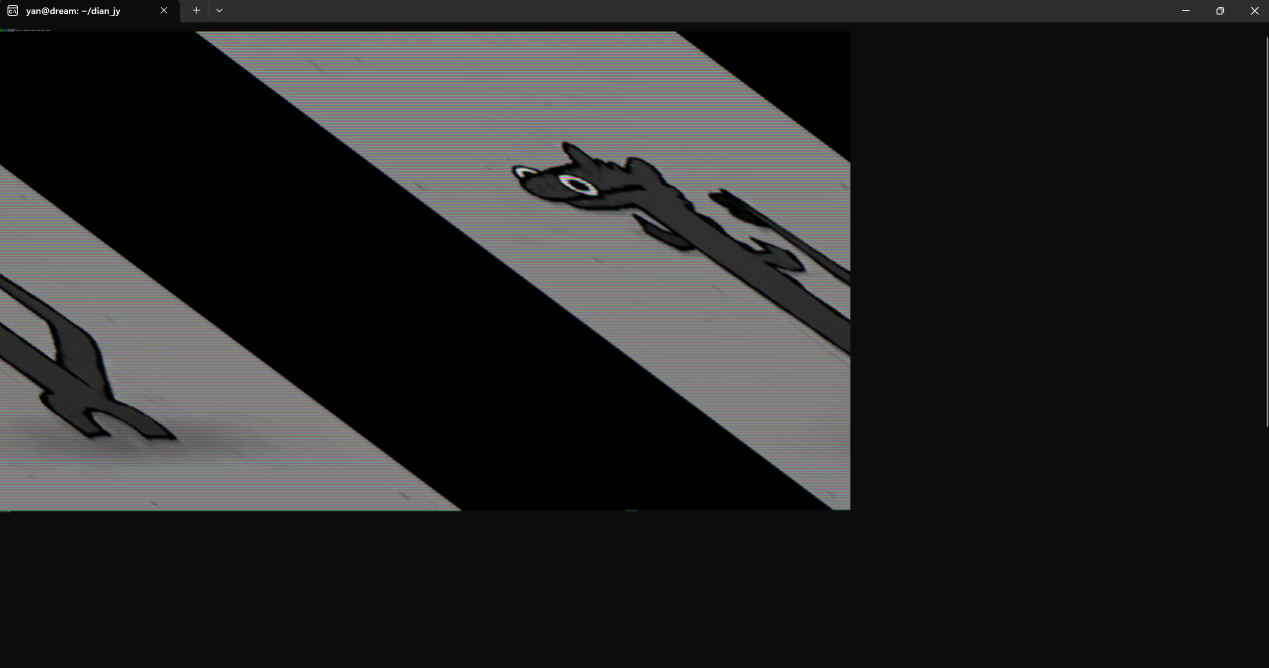
1. 调用所给库初始化视频文件、读取下一帧数据，由于想确保成功读入，因而有了printf Frame各个值的想法，但发现data的值始终没有打印，于是想像Windows中vs一样利用调试窗口看看，从而学习到了gdb在Linux中调试代码

（虽然并仍旧没有看到data中的值，再询问学长们后得到答案，不过了解到Linux中的调试方式也挺好的）。

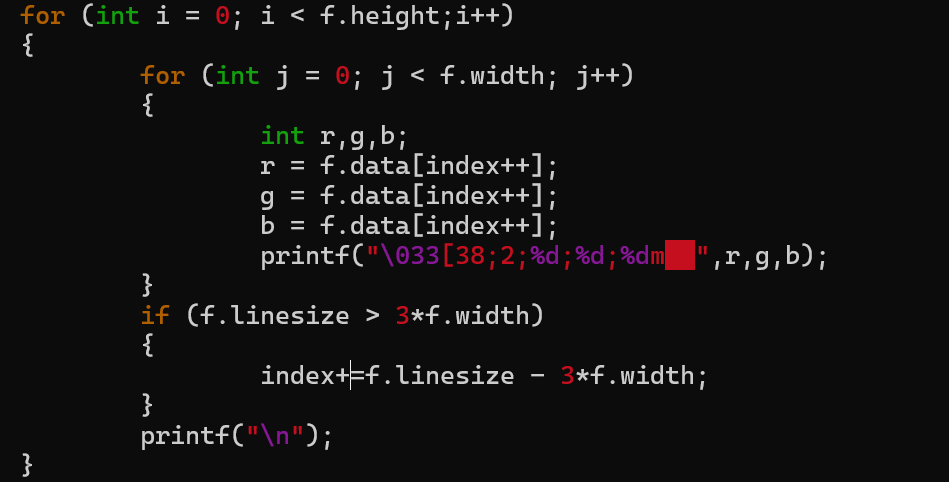
1. 一开始认为打印灰度图可以利用某个转义字符加上灰度值直接确定要打印的颜色，但查了好久资料都没有收获，最终先转向了RGB图的实现，基本代码的实现并不复杂。



但结果却不尽人意，得到了一个奇怪的图片。



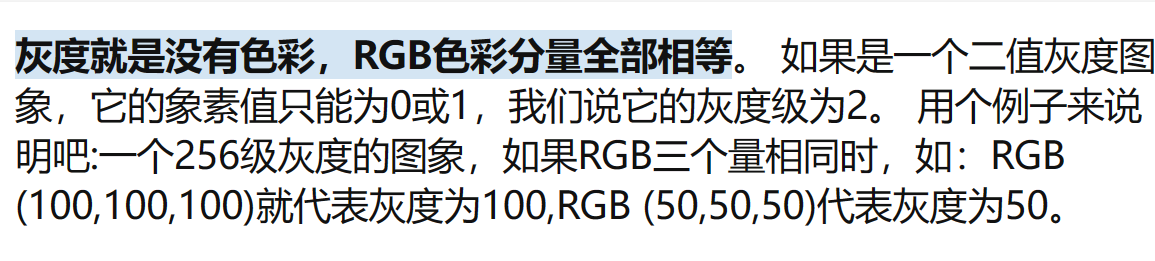
之后发现，linesize ！= 3\*width，这表面在读完一行的数据后，紧接着的并不直接是下一行的数据，启发了我改进代码



终于成功了（哭死）



1. 在完成了RGB图的实现之后，重新考虑灰度图，突然意识到RGB应该可以表示所有颜色，那灰度图是否也可以用RGB表示。



基于此，再加上公式

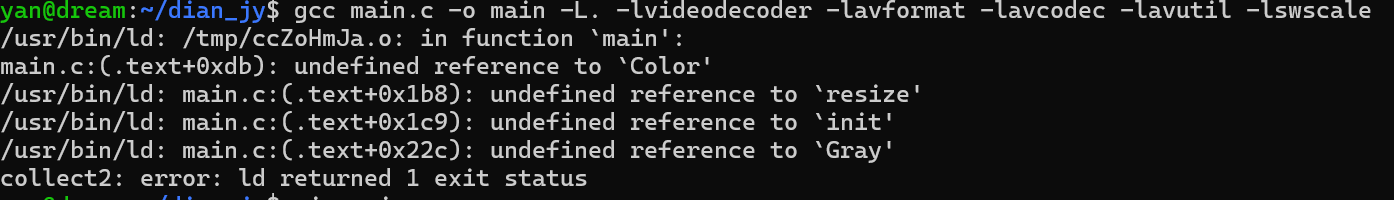


从而解决灰度图的实现。

需要注意的是，颜色通过char（8位整数）存储，因此gray也应定义为int而非double。

关于图像比例问题，由于终端中█横竖比为1:2，因此连续使用两个█可使图像比例正常。

完成上述内容后，意识到没有使用模块化编程，所有的代码都在主程序中，使代码显得冗余，因此重新编写player.c player.h,但发现



查阅资料后得知，编译运行main.c时也同时要编译player.c，否则就会出现未定义问题。

## LEVEL 1-2 Downsample

1. resize与之前遇见的命令行选项有些不同，他有多个参数，假若只使用optarg，那么无法获取所有的参数，因此需要optind，optind指向下一个参数的下标，通过判断argv[optind-1]、argv[optind]可以确定resize的参数是否正确。
2. 通过平均值resize的话，需要计算出一个窗口的r、g、b的平均值，也就是需要计算一个窗口的r、g、b的和，因此为了避免重复计算，先建立二维的前缀和数组来维护原图像。

## Level1-3LEVEL 1-3 Video player

重复读取每一帧与打印每一帧，通过usleep调节帧率，通过system（“clear”）清屏，但由于打印效率过低，每一帧的画面都存在一个从上到下的过程，导致画面非常不连贯。

## Level2 优化（未成功）

1. 使用多线程，一边取缓冲一边写缓存符合队列先进先出的特点，因此想要通过一个链表来实现队列，读进程在链表尾部增加结点（每个结点的数据域存放每一帧的数据），写进程从链表头拿数据打印，打印完成后释放表头结点，当链表中结点不足时阻塞该线程。但是由于读缓存运行的元快鱼写进程，因此常常使内存爆掉，想到的改进方案是控制链表长度，但优化出来的结果也确实微乎其微。（而且还对后续的键盘事件产生了一些影响）
2. 了解了一下printf的机制，正常运行时，printf使用的是行缓冲机制，即先将要打印的内容加载到缓冲区，当遇到换行符时再一次性打出。因此我怀疑是否是这种行缓冲机制造成了视觉效果上从上到下的出现，于是使用setvbuf将缓冲模式改成了全缓冲，并将缓冲区大小夸大到5e6,不过效果依然不是很明显。
3. 打印的时间复杂度与打印字符数量成正比，因此主动使用resize将窗口、步长调大，可以实现较为明显的优化效果。

## Level3 键盘事件

1. 实现一个Linux下的kbhit函数
2. 改变标准输入的模式，通过tcgetattr与tcsetattr将STDIN\_FILENO 的ICANO（标准模式）和ECHO（回显）去掉，从而解决了按回车才能触发的问题。
3. 通过select、FD\_ISSET来检查是否可读。具体使用为先定义一个fd\_set类型变量rdfs，用FD\_ZERO将refs每一位清零，用FD\_SET将STDIN\_FILENO加入rdfs，再用select（STDIN\_FILENO+1,&rdfs,NULL,NULL,&tv）（第2、3、4个参数分别为检查可读性、可写性、是否有异常出现，此处只关心可读性）最后通过FD\_ISSET检查是否可读。
4. 在打印过程中重复判断kbhit（），若条件成立则getchar（），判断是否为空格或d/D，若为空格则暂停，若为d/D则让打印帧后不sleep以达到加速效果（由于打印耗时较长，有时加速可能不明显，但确实是加速了。。。）

未解决的问题：当使用多线程时，若过久长按一个键，可能会导致后续的键盘事件判断失灵或滞后，就算使用了fflush(stdin)或rewind(stdin)也会出现这种情况，而且当画面分辨率越高，这种问题越严重。

根据另建程序测试，不论是fflush（stdin）还是rewind（stdin）都没有实现刷新缓冲区的效果，由此产生了两个怀疑：1.由于tcsetattr,所以对stdin的操作无效。2.这两个函数都没用。

根据测试，两个函数确实没用。。。