编号：

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 实验 | 一 | 二 | 三 | 四 | 五 | 六 | 七 | 八 | 总评 | 教师签名 |
| 成绩 |  |  |  |  |  |  |  |  |  |  |

武汉大学国家网络安全学院

课程实验(设计)报告

课程名称 ： Linux架构分析与安全设计

实验内容 ： 实验3 Linux下编写键盘监控程序

专业(班) ：

学 号 ：

姓 名 ：

任课教师 ： 王娟

2020 年 11 月 2 日

目 录

[实验三 Linux下编写键盘监控程序 3](#_Toc55840556)

[1. 实验目的 3](#_Toc55840557)

[2. 实验环境 3](#_Toc55840558)

[3. 实验原理 3](#_Toc55840559)

[4. 实验要求 5](#_Toc55840560)

[5. 实验内容和步骤 5](#_Toc55840561)

[6. 实验报告 9](#_Toc55840562)

[7. 实验思考与改进 16](#_Toc55840563)

# 实验三 Linux下编写键盘监控程序

## 实验目的

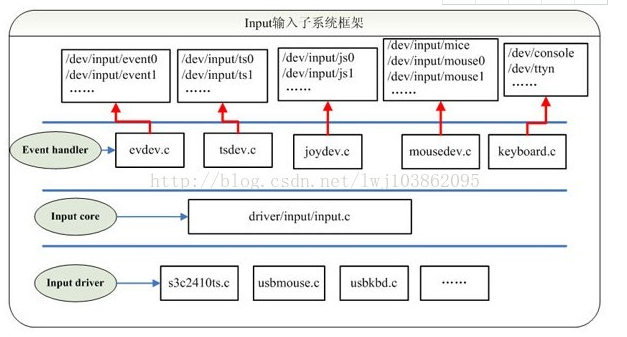
在ubuntu下实现键盘监控功能，并把监控内容写到一个文件里 。

## 实验环境

* 1. 操作系统：Ubuntu 16.04
  2. 软件工具：GCC

## 实验原理

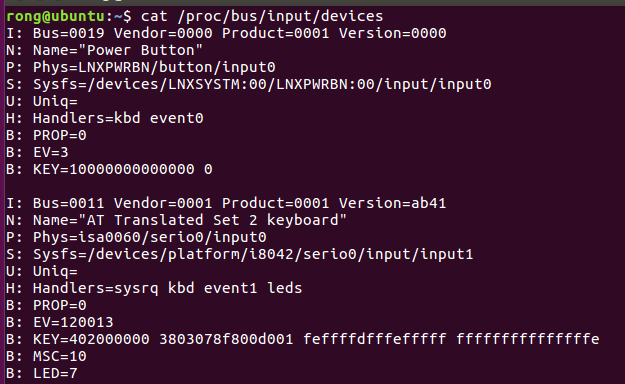
Linux 输入子系统是 Linux内核用于管理各种输入设备（键盘，鼠标，遥控杆，书写板等等）的。linux输入子系统从上到下由三层实现，分别为：输入子系统事件处理层（EventHandler）、输入子系统核心层（InputCore）和输入子系统设备驱动层。其中设备驱动层提供对硬件各寄存器的读写访问和将底层硬件对用户输入访问的响应转换为标准的输入事件，再通过核心层提交给事件处理层；而核心层对下提供了设备驱动层的编程接口，对上又提供了事件处理层的编程接口；而事件处理层就为我们用户空间的应用程序提供了统一访问设备的接口和驱动层提交来的事件处理。所以这使得我们输入设备的驱动部分不在用关心对设备文件的操作，而是要关心对各硬件寄存器的操作和提交的输入事件。Linux的输入子系统框架如下图所示：



/dev/input目录下显示的是已经注册在内核中的设备编程接口，用户通过open这些设备文件来打开不同的输入设备进行硬件操作。事件处理层为不同硬件类型提供了用户访问及处理接口。例如当我们打开设备/dev/input/mice时，会调用到事件处理层的Mouse Handler来处理输入事件，这也使得设备驱动层无需关心设备文件的操作，因为Mouse Handler已经有了对应事件处理的方法。输入子系统由内核代码drivers/input/input.c构成，它的存在屏蔽了用户到设备驱动的交互细节，为设备驱动层和事件处理层提供了相互通信的统一界面。

总的来说。一个事件，如鼠标移动，键盘按下事件，首先通过   
 驱动层Driver --> 输入核心层 Input Core-->事件处理层Event handler-->用户空间user space的顺序来完成事件的响应。

我们可以通过 cat /proc/bus/input/devices 以查看到当前input子系统下面的所有event设备。我们对计算机的输入(包括敲击键盘，移动鼠标等等操作)经过内核(底层驱动,input)处理最后就上报到这些event里面了。而这里event0,event1,..就是用来区分各个外设的。



I：设备的ID

N ：设备的名字

P ：在系统层次结构中的设备的物理路径

S ：sysfs 路径

U ：设备的唯一识别码（如果设备有）

H ：与设备关联的输入句柄列表.

B ：位图

PROP：设备属性.

EV：设备支持的事件类型

KEY：此设备具有的按键/按钮。

MSC：设备支持的各种事件。.

LED：设备上的leds

上图是我的键盘设备,我们可以看到是event1。在/usr/include/linux/input.h中有定义，这个文件定义了event事件的结构体，API和标准按键的编码等；我们需要将要写入的按键编码填充到结构体中，然后写入键盘或者鼠标的事件设备驱动文件中。



结构体内容依次是按键时间。type事件类型：我们关注的是按键事件。code：事件的代码。按键事件的类型代码code是EV\_KEY，该代码为设备键盘代码。在该文头文件中已经定义的0-248中不同的键盘按键代码（/usr/include/linux/input.h）。value： 事件的值。按键事件的类型代码是EV\_KEY,当按键按下时值为1,松开时值为0。

## 实验要求

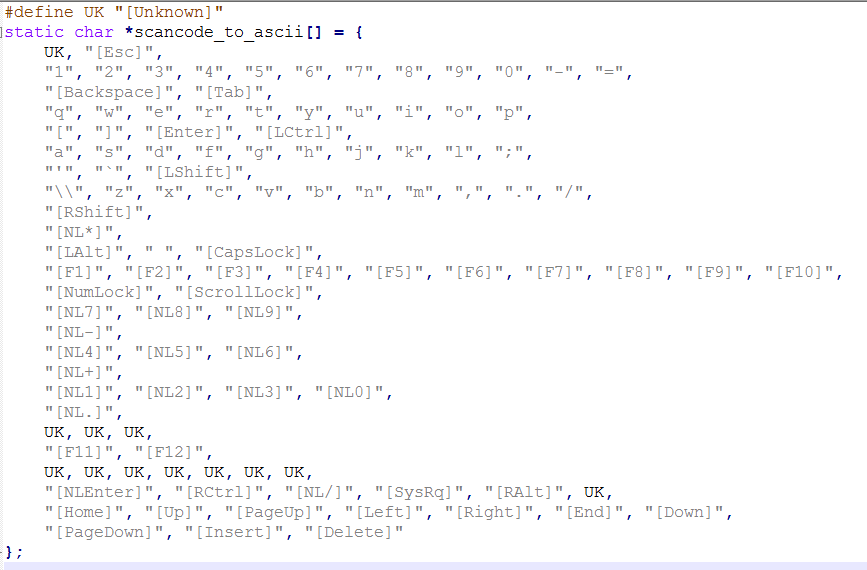
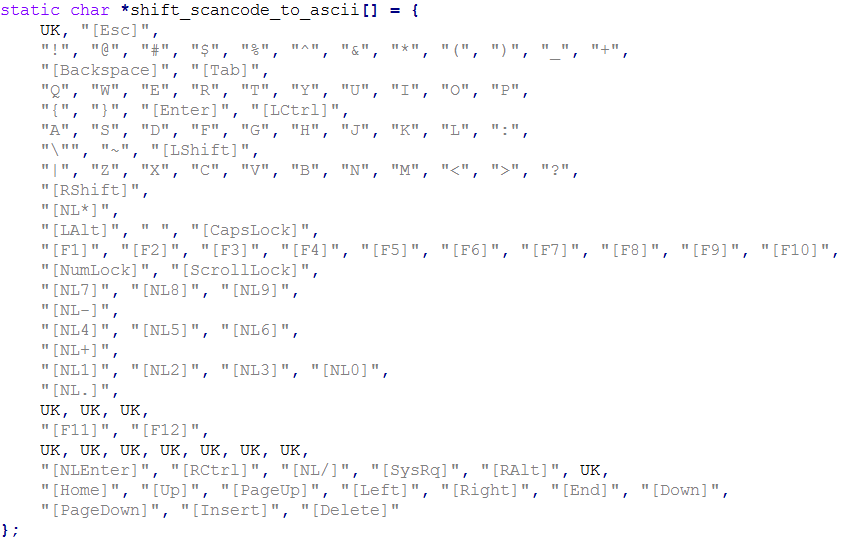
1. 通过认真学习linux键盘监控原理，明确实验目的、原理、方法以及注意事项等。
2. 实验过程中必须认真严肃，并认真学习和记录实验数据，从而进行科学分析。
3. 独立认真完成实验报告，语言简练、表达清晰，适当情况下增加相应的画图信息。

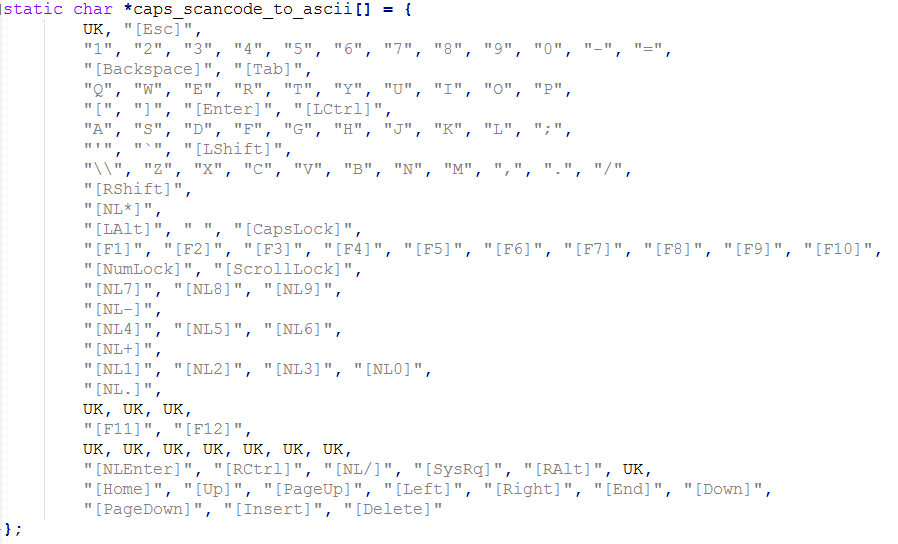
## 实验内容和步骤

1. 用C语言编写一个Linux键盘记录器。

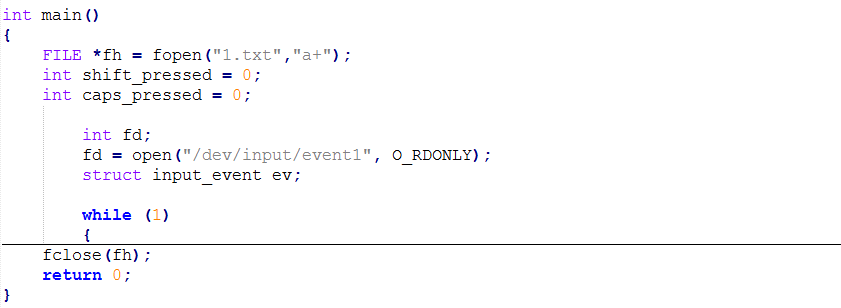
（1）编写键盘记录程序

每当按下Caps lock键和按Shift键时，我们都会制作一个键盘布局，如下所示：

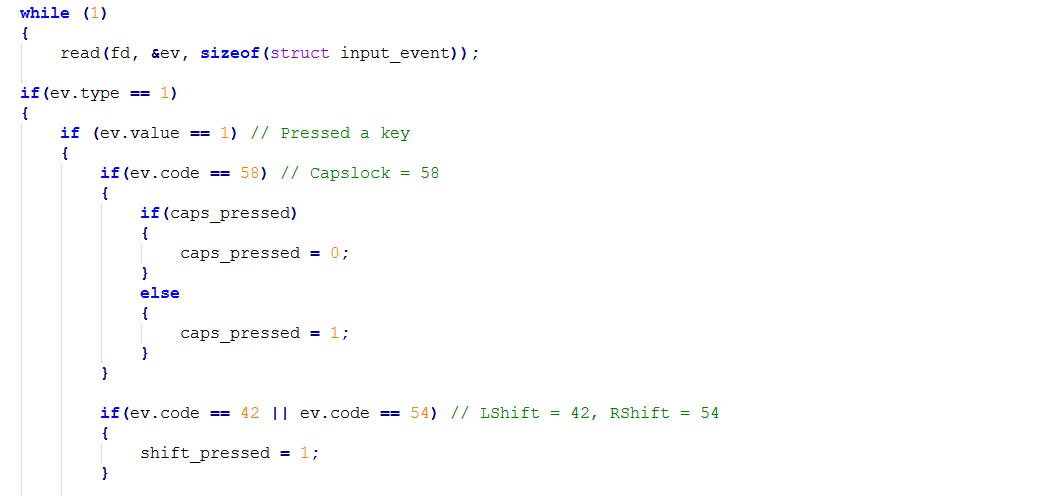
 

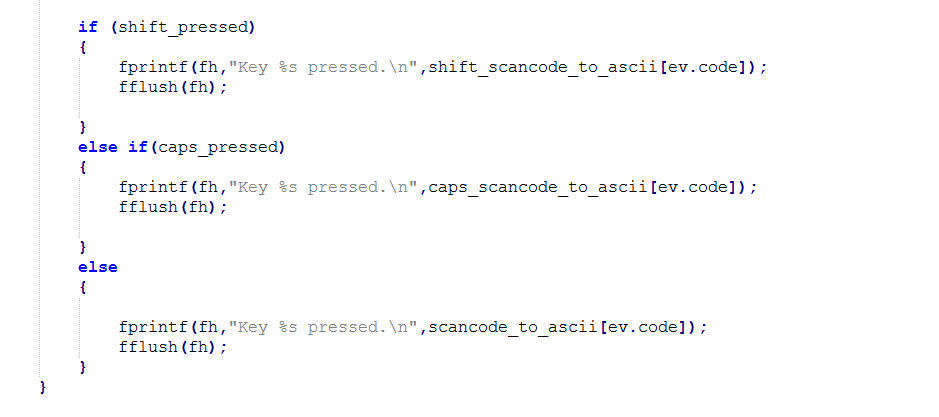


主函数包含一些变量和一个while循环。先创建并打开一个文件1.txt，这个文件将记录我们的按键信息。然后我们读取event1，参数O\_RDONLY 意味着我们只能读取文件。



在while循环中，我们读取fd并将信息存储在ev中，ev.code是扫描码。我们检查第一个if语句，查看按下的键是否是大写键。 如果是，我们将检查ifcaps\_pressed是否已经是1，如果为1，我们将其设置为0.否则我们将其设置为1.我们还检查是否按下shift，如果按下，我们sethift\_pressed为1.每当shift已经被释放，我们将其设置为0. [RShift]，[LShift]和[CapsLock]的扫描码分别为54,42和58。然后我们将按键信息输出到文件里面。





（2）编译运行（在root权限下进行）

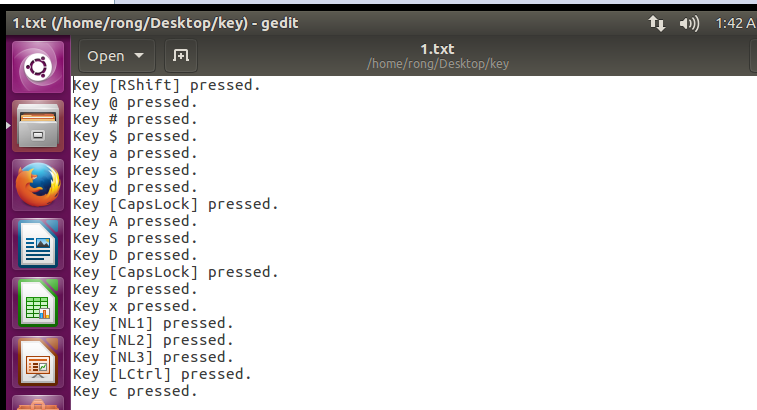
gcc -o keylogger keylogger.c

./keylogger

（3）查看监控记录。

在电脑上随意进行击键操作，发现在同目录下生成了1.txt文件，击键内容也保存到了1.txt中。





二．用Python语言编写一个Linux键盘记录器。

大家自己百度去做实验，原理和C语言版的键盘记录程序一样。参考网址：<http://www.s0nnet.com/archives/linux-keylogger-with-python>。

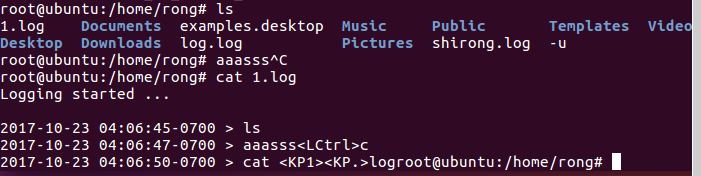
三．Linux 16.04自带keylogger。

1. 安装logkeys。

sudo apt-get install logkeys

logkeys -s -o 1.log //利用logkeys --help查看用法

2.当时输入ls以及aaasss ctrl c时，按键信息都被保存在1.log文件里，查看1.log即可得到信息。



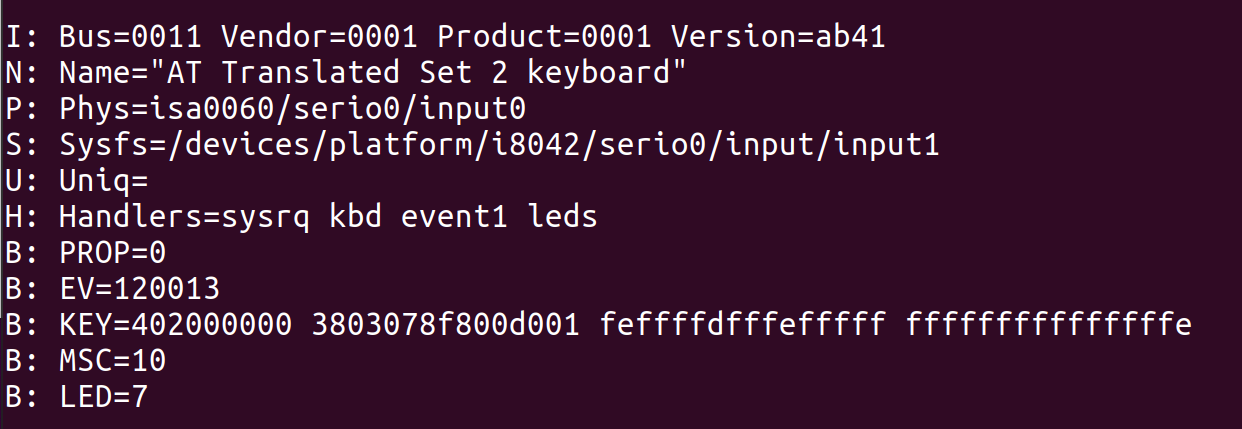
3.logkeys -k //停止

4.Logkeys网址：<https://github.com/kernc/logkeys>。

## 实验报告

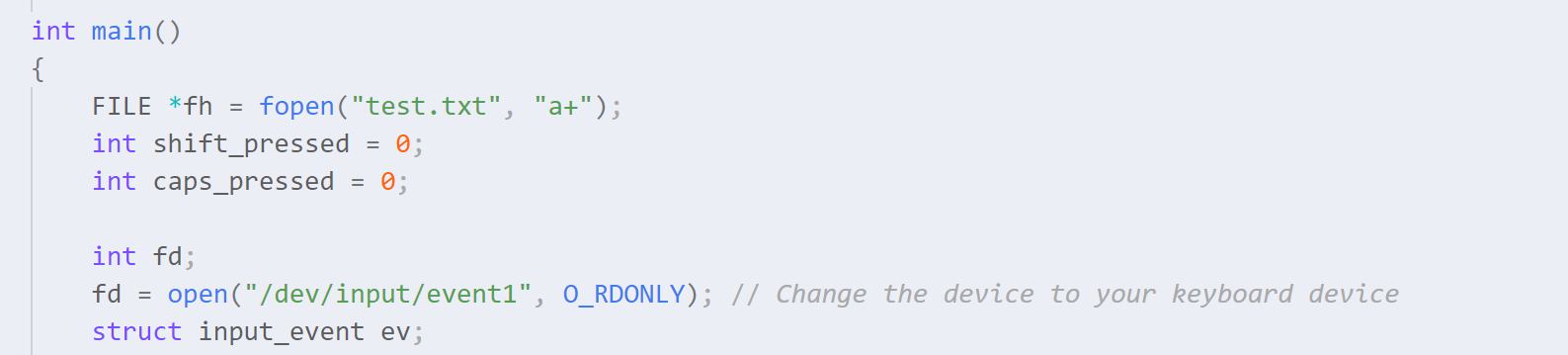
#### 实验一：用C语言写一个Linux键盘记录器

##### 查看键盘设备

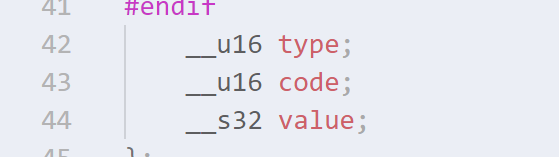
使用cat /proc/bus/input/devices查看键盘设备，可以发现当前键盘设备是event1

##### 编写键盘记录程序

主程序中我们定义了日志文件为test.txt，以追加写模式打开，同时打开了事件event1。



其中struct event结构体的主要内容如下。



在while循环中，我们循环读取fd并将信息存储在ev中，其中ev.code是扫描码。我们首先通过ev.type和ev.value判断是否有键按下，如果有键按下，通过ev.code==58判断按下的键是否是大写键。 如果是，我们将检查ifcaps\_pressed是否已经是1，如果为1，我们将其设置为0.否则我们将其设置为1。我们还检查是否按下shift，如果按下，我们sethift\_pressed为1.每当shift已经被释放，我们将其设置为0. [RShift]，[LShift]和[CapsLock]的扫描码分别为54,42和58。然后我们将按键信息输出到文件里面。

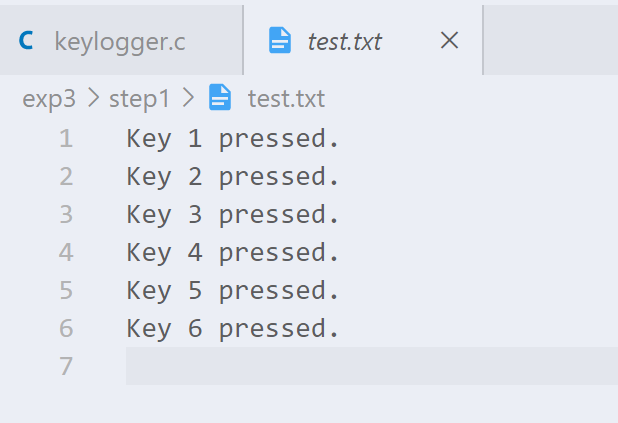


##### 编译并测试结果

通过sudo运行监控程序



在命令行输入密码，可以发现我们的登陆密码被记录到test.txt文件中



#### 实验二：用python语言写一个Linux键盘记录器

该实验实现了用python对键盘和鼠标进行监控。

##### 思路

实验通过pynput模块实现对键盘和鼠标的监控。

logging模块实现将监控信息输入文件。

argparse模块实现对命令行参数的读取。

##### 代码流程

###### 主程序

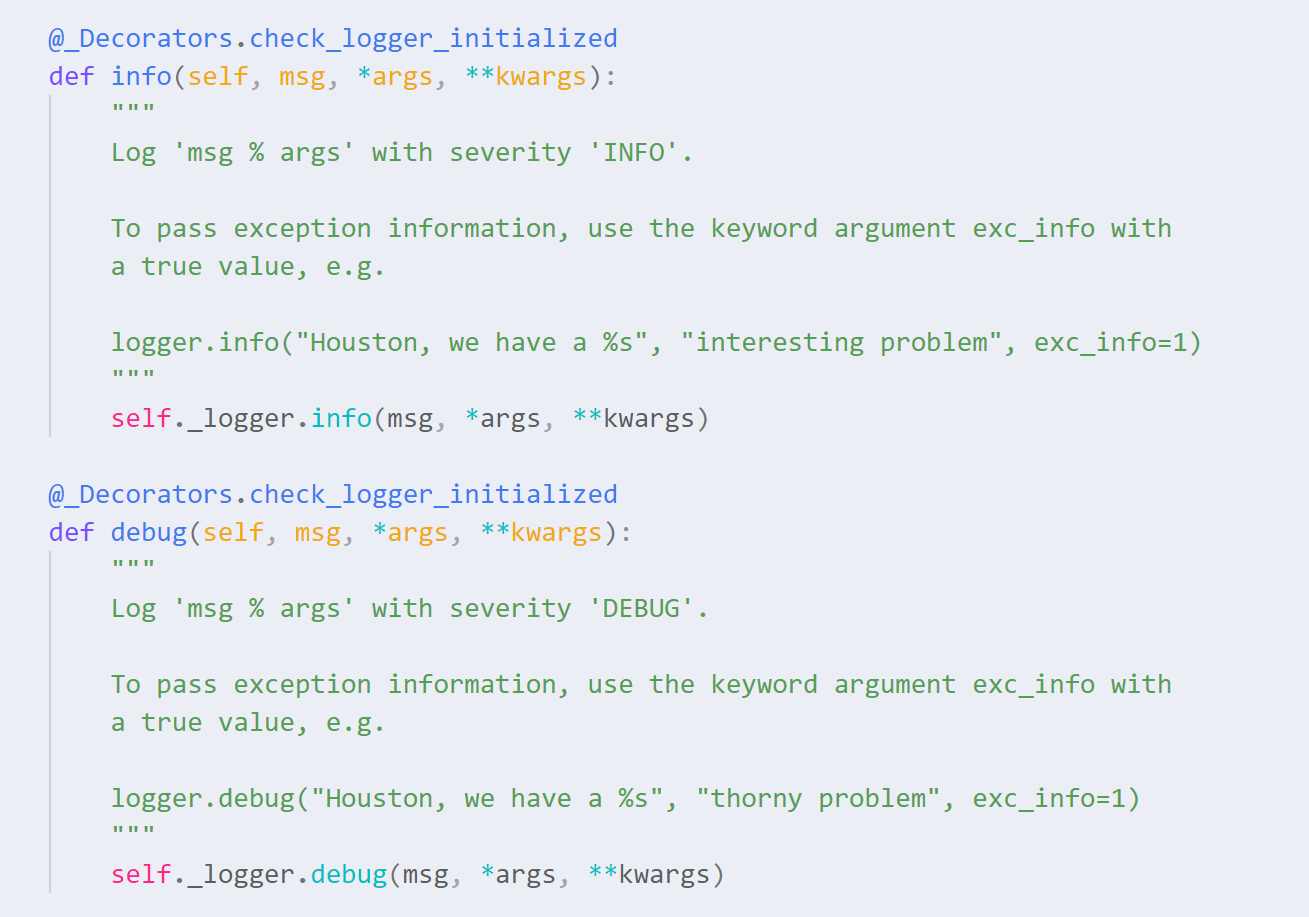
主程序位于main.py



在入口程序中，通过接收到的命令行**占位**参数keyboard和mouse指定要监听的设备，file指定要记录的文件。

###### MyLogger

MyLogger位于logger.py，是对python标准模块logging的一个封装，主要目的是能够**在加载logging配置之后指定文件**(配置文件位于logger\_config.py)。部分内容如下：



###### KeyboardListener

KeyboardListener位于listener.py，可通过调用listen实现对键盘的异步监听。其中\_on\_press和\_on\_release分别定义了键盘按下和释放触发的事件。



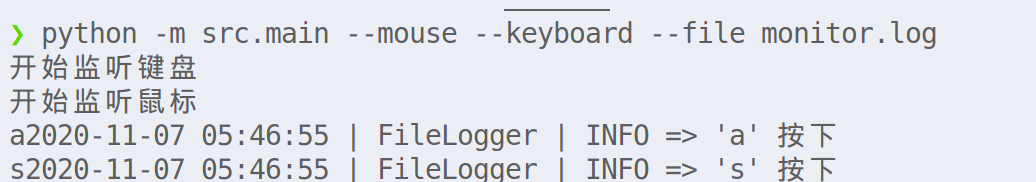
###### MouseListener

MouseListener位于listener.py，可通过调用listen实现对鼠标的异步监听。其中\_on\_move、\_on\_click和\_on\_scroll分别定义了鼠标移动、点击和滑动触发的事件。

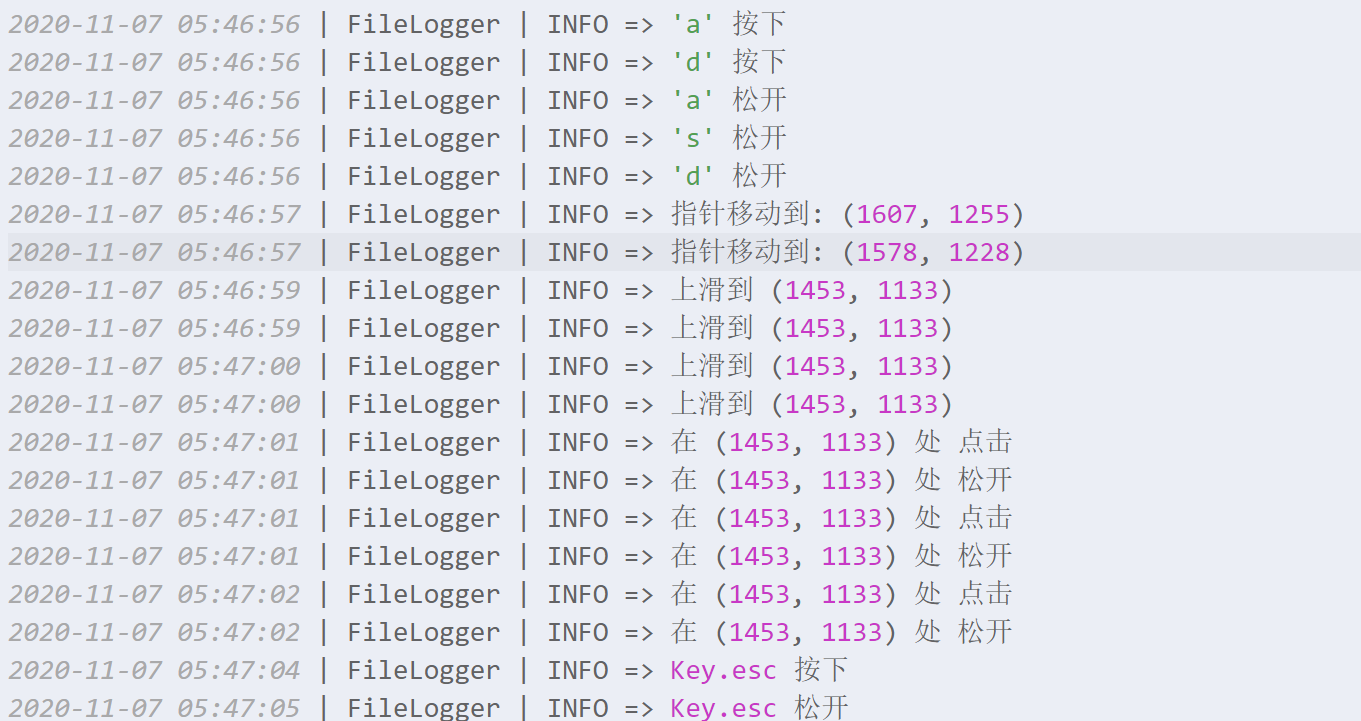


##### 实验结果

运行监控程序

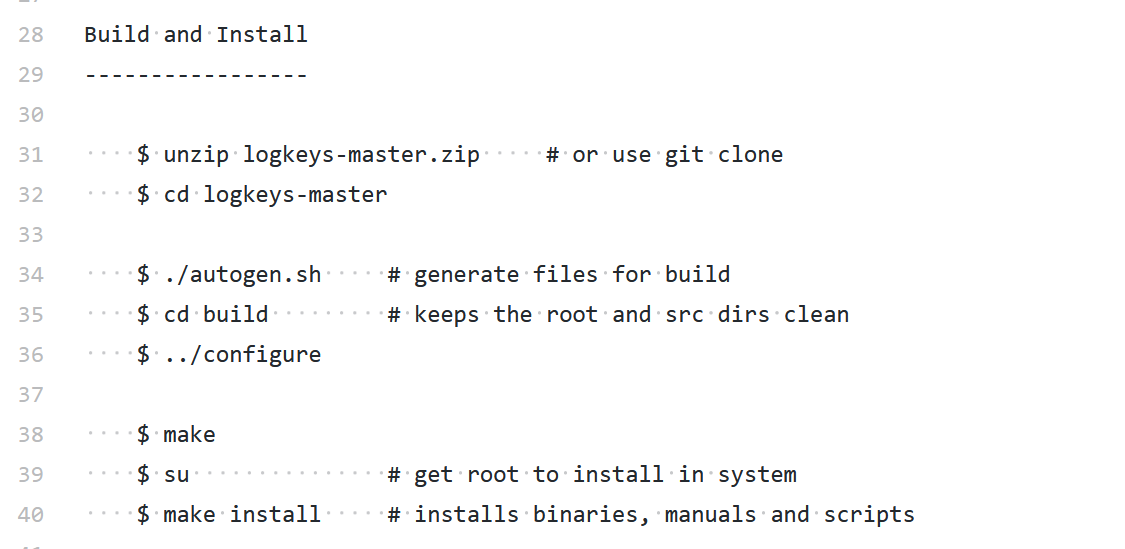


可以在日志中看到我们的鼠标和键盘都被成功监控

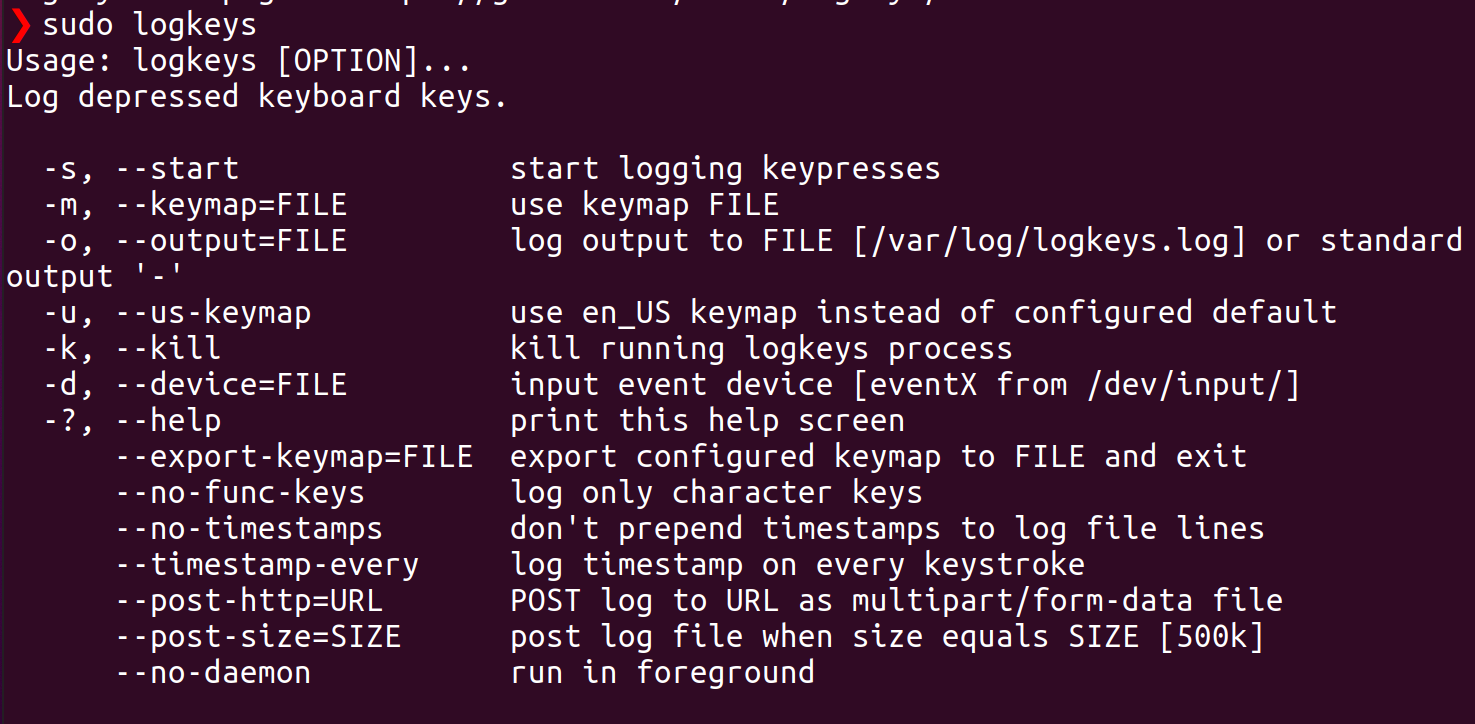


#### 实验三：使用自带的keylogger

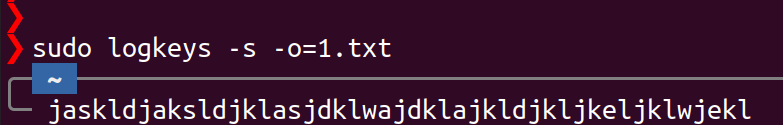
由于Ubuntu 20.04的apt中不含有logkeys，于是根据官网的命令编译并暗黄logkeys



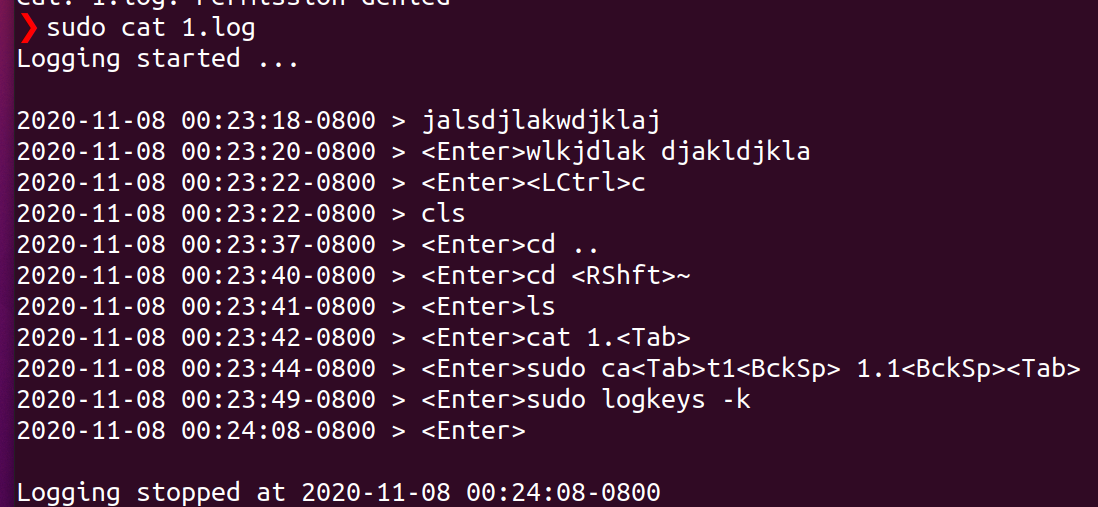
查看帮助选项



执行监控命令



查看记录的信息



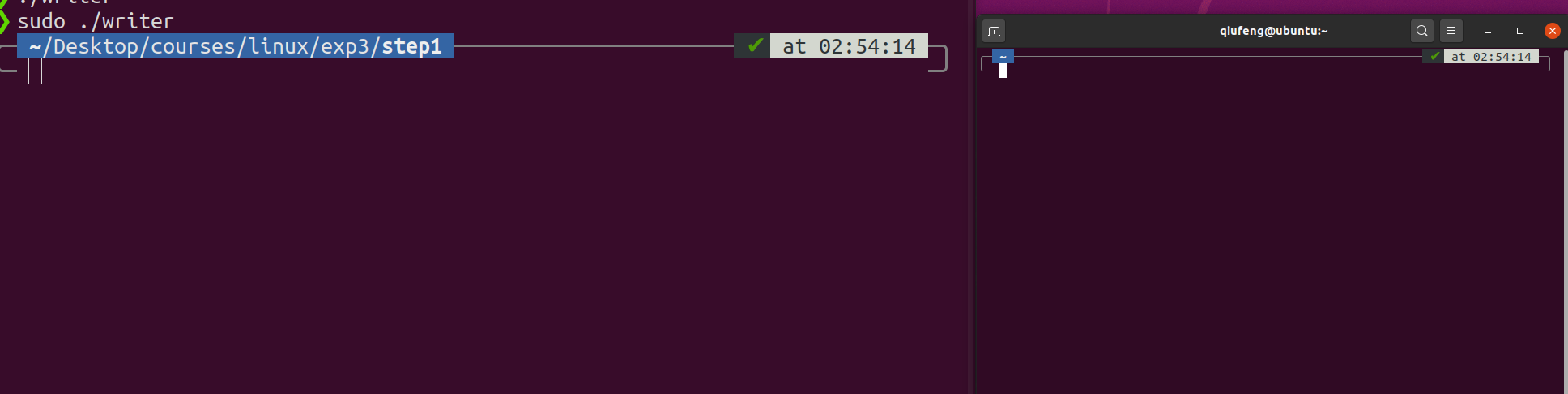
## 实验思考与改进

#### 除了通过读event x事件，我们还可以写event x实现模拟键盘按键

我们测试模拟按下alt+ctrl+t(该按键会在ubuntu中弹出一个终端)



在ubuntu中进行测试，可以发现成功弹出新的终端窗口



#### 仔细阅读logkeys的源码，分析实现原理与过程。

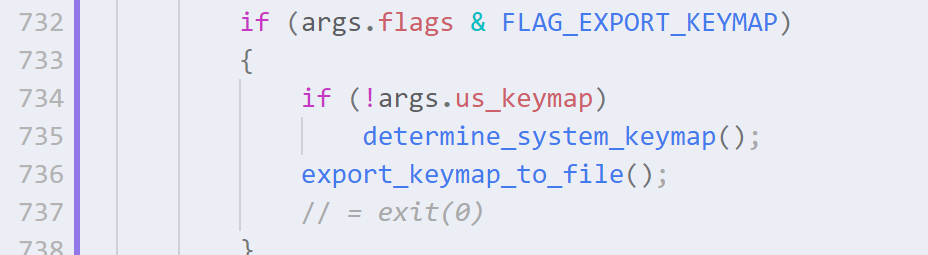
##### 流程分析

首先定位到main函数，可以发现它执行的过程主要包括

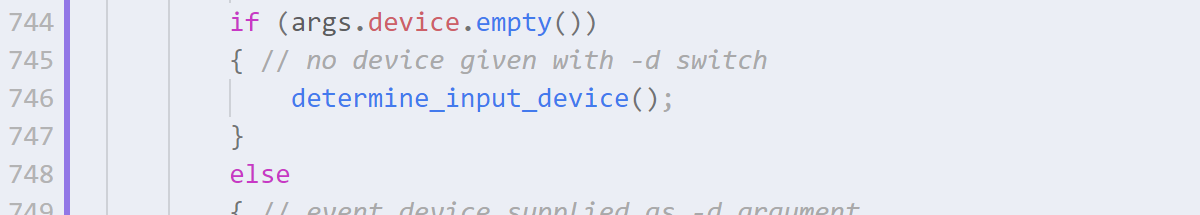
* 处理命令行参数



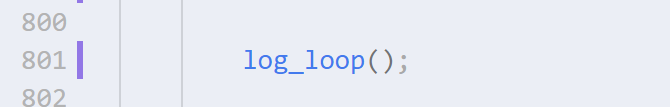
* 获取键盘映射



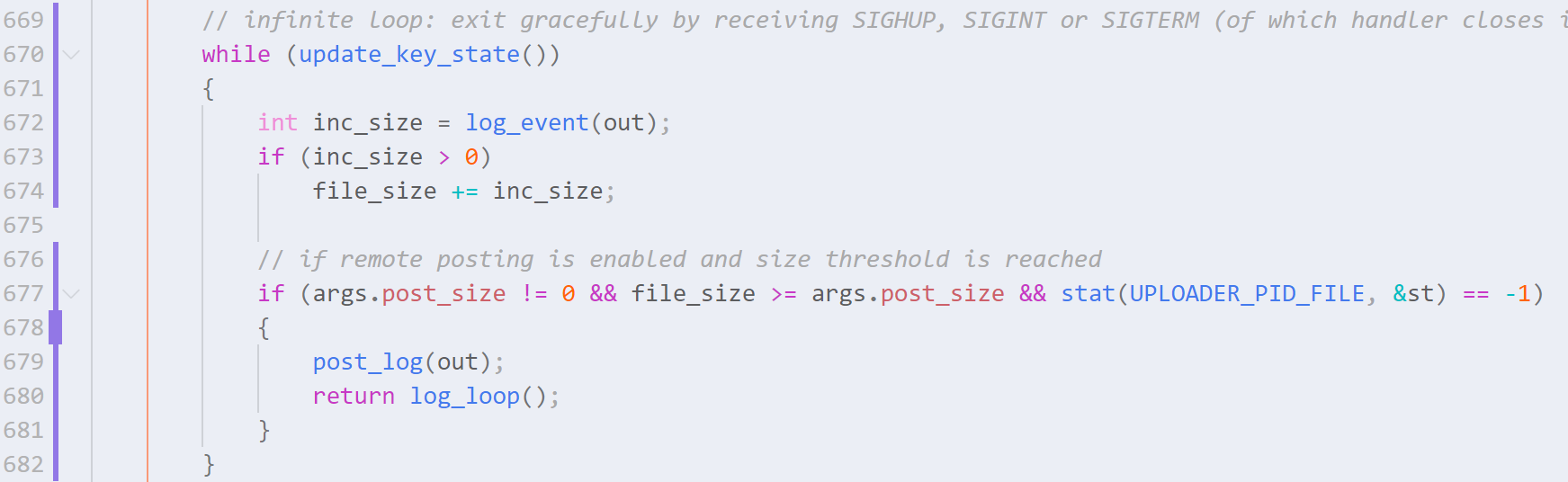
* 获取输入设备



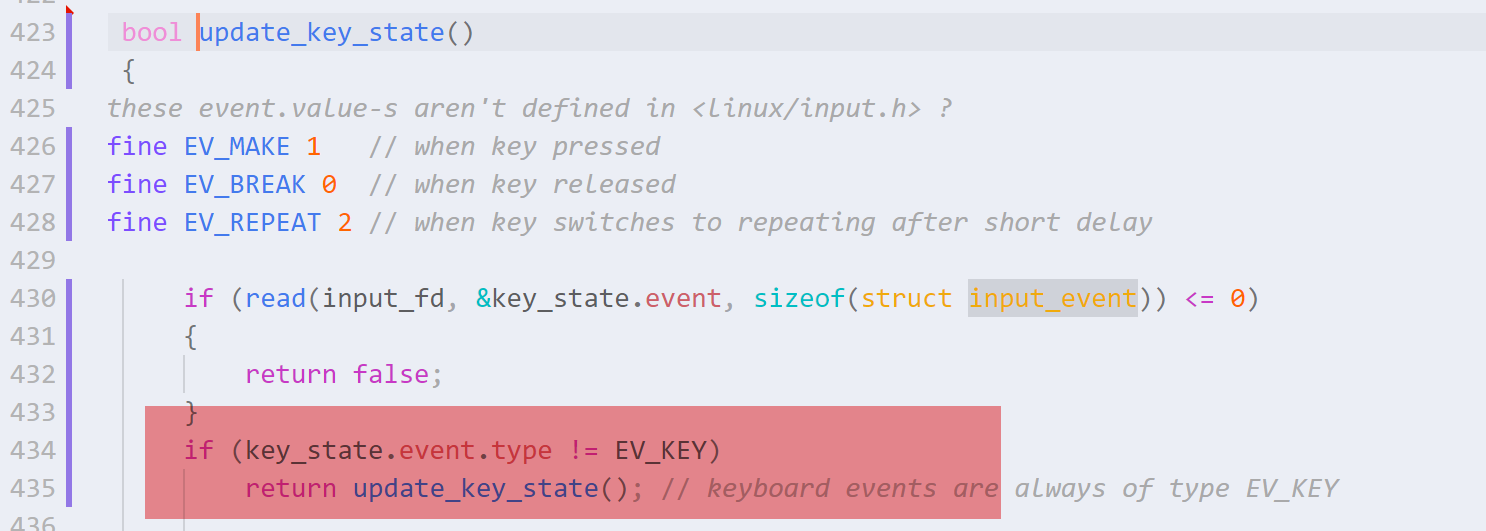
* 进行记录循环



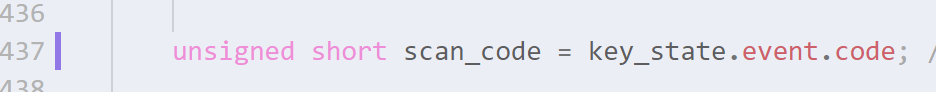
跟踪到其进入记录循环的函数log\_loop，可以发现当update\_key\_state为真时程序循环记录按键信息



进入到update\_key\_state函数，可以发现在这里首先判断是否有键盘输入，如果没有则一直递归调用自身。



如果接收到信息，就将其存入scan\_code变量中



接着是对监控到的信息进行判断，包括对大写键(CAPSLOCK)、SHIFT键、ALT键等判断，以及判断当前键入的是否为字符。



##### 原理分析

该程序的原理和实验一类似，都是通过结构体**struct event循环**获取键盘键入的事件信息，并监控得到的按键码code。

