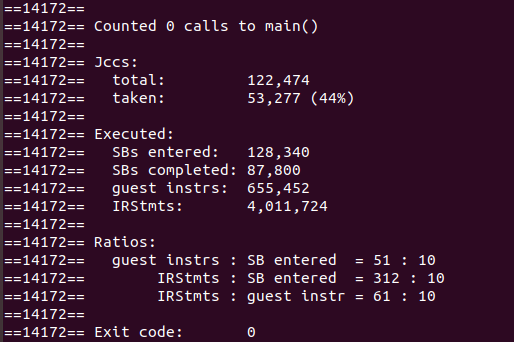
**实验日志**

**一，valgrind的安装**

执行命令：sudo apt-get install valgrind 安装（左图）



安装完成（左图）

执行命令：valgrind --log-fd=1 --tool=lackey -v --trace-mem=yes ls –l（右图）

验证成功（右图）。

**二，学会 PartA 部分参考引用 csim-ref 模拟器使用命令行参数及使用方法**

格式: ./csim-ref [-hv] -s <s> -E <E> -b <b> -t <tracefile>，详细介绍见参考书

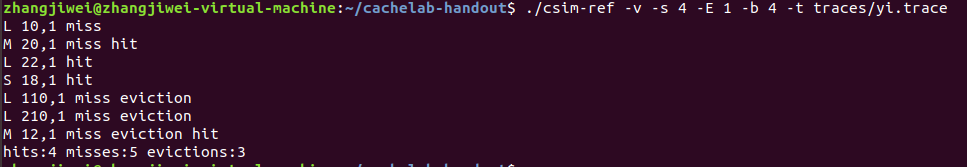
使用方法：

linux> ./csim-ref -s 4 -E 1 -b 4 -t traces/yi.trace 打印出hit、miss、eviction 次数

linux> ./csim-ref -v -s 4 -E 1 -b 4 -t traces/yi.trace 打印出详细信息

**三，认识I、L、S、M 操作含义，并运行在缓存模型上，逐条分析文件操作且对比 LRU 替换策略，能分析hit、miss、eviction 结果产生过程**

“I”表示指令加载，“L”表示数据加载，“S”表示数据存储，“M”表示数据修改（即数据存储之后的数据加载）。



# 分析：

执行的指令设置偏移位的宽度为 4，故可计算出组索引s 的值

# L 10,1 对于地址 0x10 进行访问：

x10=0001 0000，s=1;

初始时 cache 是空的，第一次访问为 miss。

# M 20,1 数据修改（连续对地址 0x20 进行连续两次访问）：

0x20=0010 0000，s=2;

第一次访问时没有要的内容，miss；然后 cache 从低一级存储器读取第一次访问需要的内容。第二次访问时有需要的内容且标记位相等，hit；

# L 22,1 对地址 0x22 进行访问：

0x22=0010 0100，s=2;

由于 M 20,1 以将该块存入高速缓存且标记位都相等，故 hit;

# S 18,1 对地址 0x18 进行访问：

0x18=0001 0100,s=1；

由于之前的操作，该块已存入高速缓存且标记位都为 0，hit

# L 110,1 对地址 0x110 进行访问：

0x110=0001 0001 0000 s=1；

由于 S 18,1 将该块存入高速缓存了但标记位不相等，故 miss,发生一次 eviction；

# L 210,1 对地址 0x210 进行访问：

0x210=0010 0001 0000， s=1，

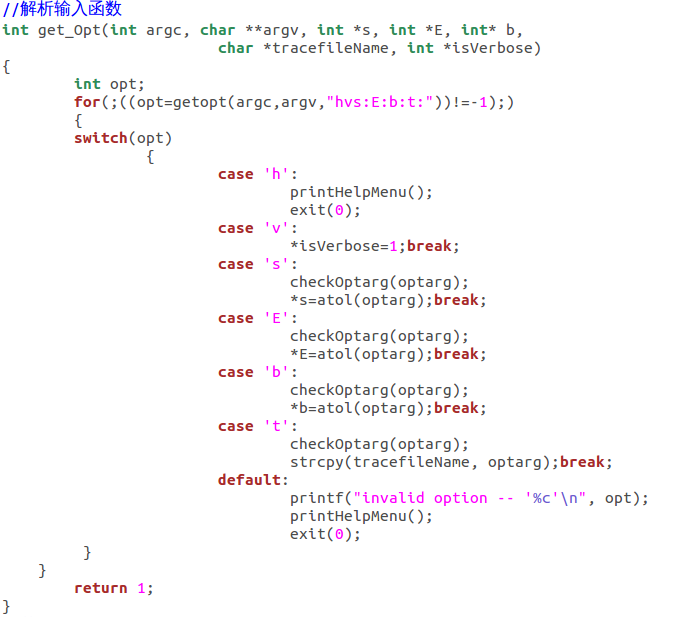
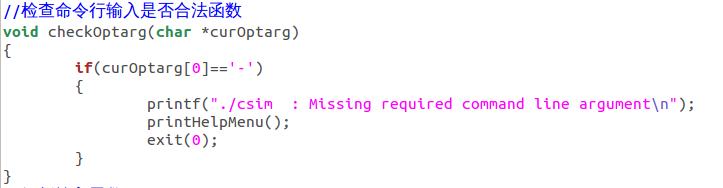
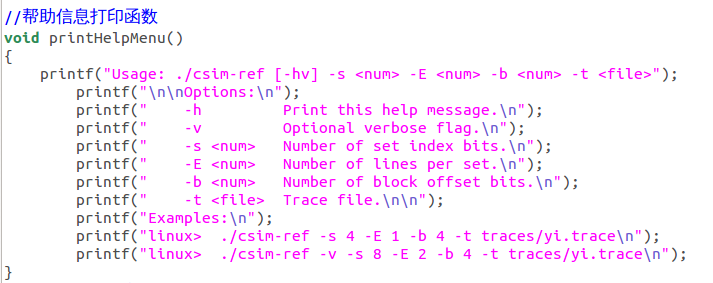
这里标记位为 2 跟L 110,1 操作读取新的行的标记位不匹配，miss,cache 读取新的行，发生一次 eviction。

# M 12,1 对地址 0x12 进行连续两次访问：

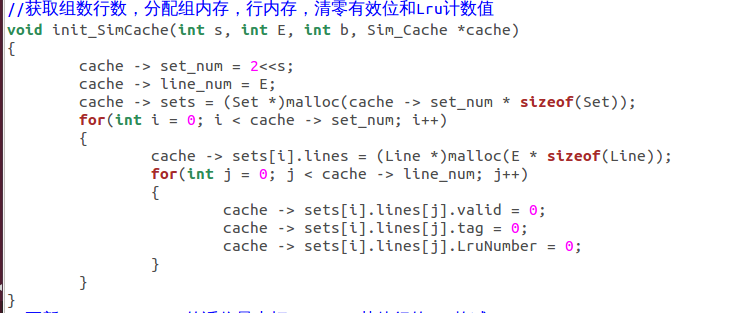
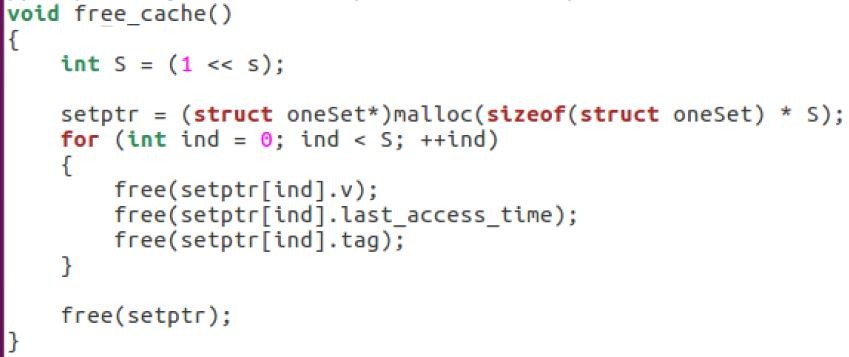
0x12=0001 0010,s=1;

第一次 miss，因为操作 L 210,1 时发生了一次行替换把该块驱逐了（即使标记位相等）,cache 重新读取该块，发生一次eviction，第二次肯定为 hit。

**四，编写函数体get\_Opt()、helpPrint()、checkOptArg()**



**五、编写创建并释放部分代码的程序：initCache()、freeCache()**



**六、检验主函数代码以及结果分析**

**分析**：右图可以看到我编写的缓存模拟器与样例得分一致，验证正确。

