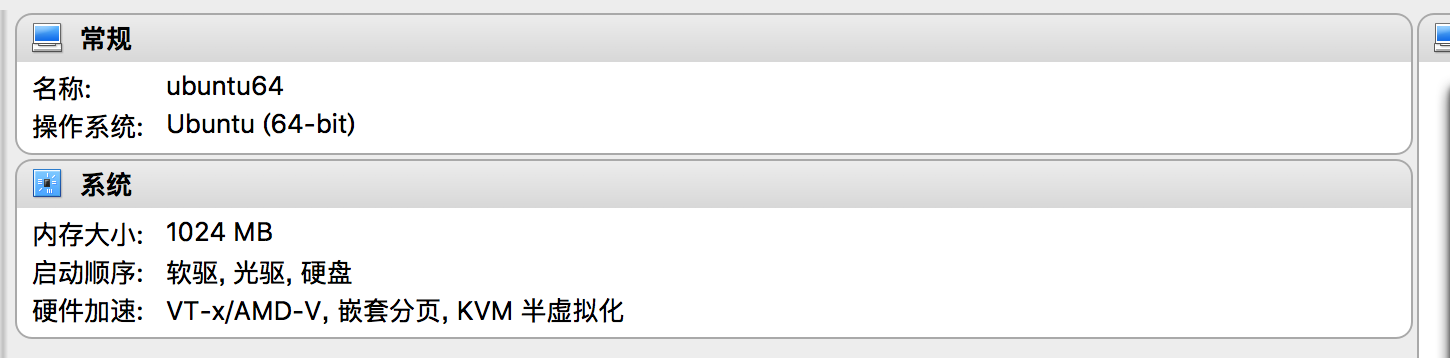
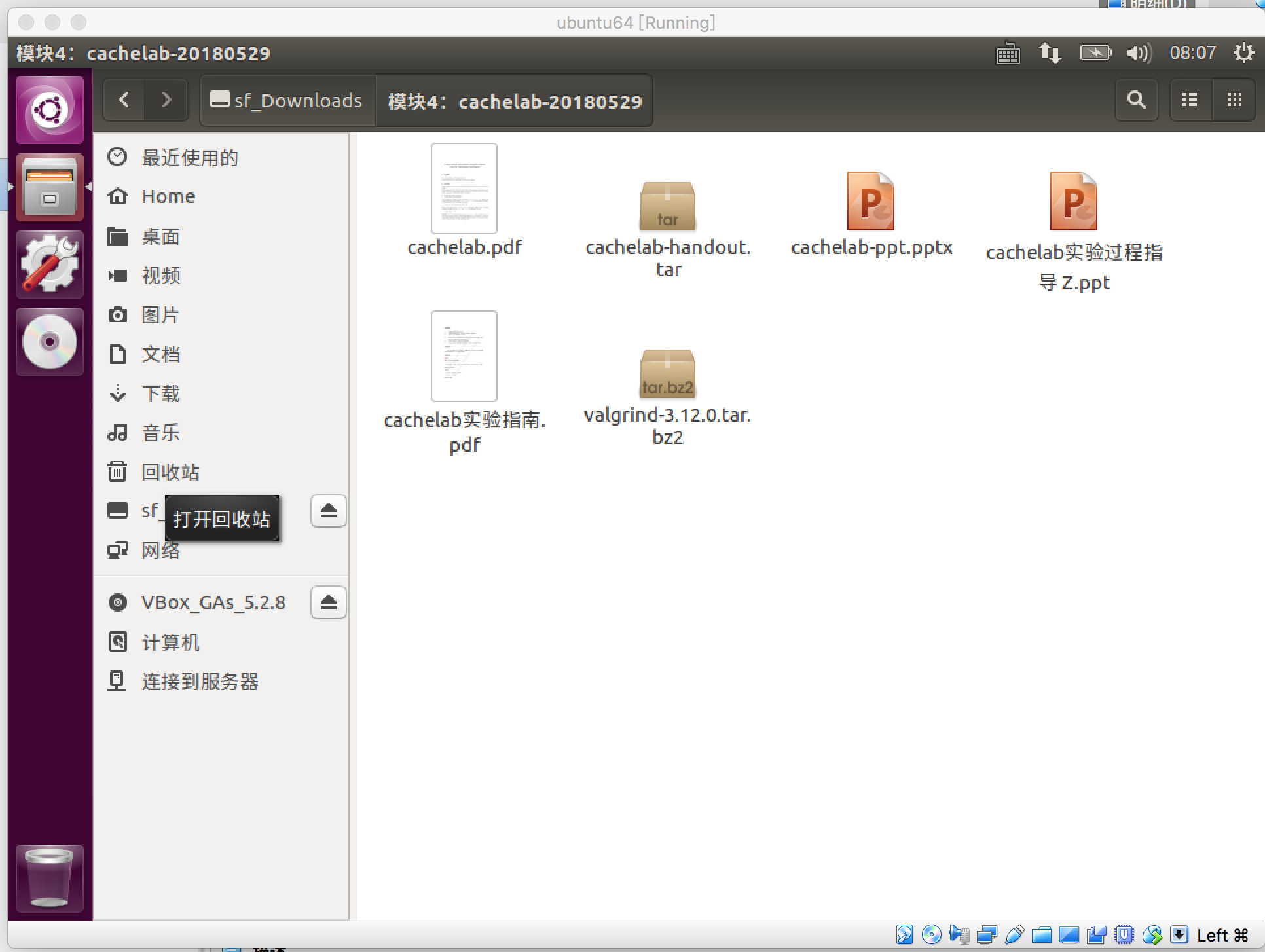
5.29

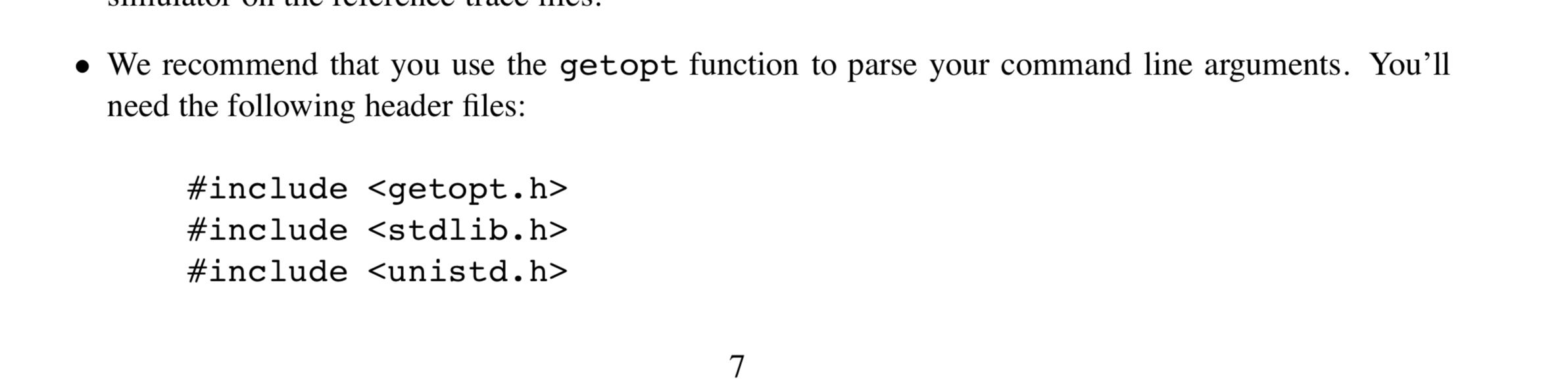
安装64位乌班图，建立实验环境





了解getopt





分析数据结构：

typedef struct{

int valid; //有效位

int tag; //标识位

int LruNumber; //LRU算法需要位

} Line;

typedef struct{

Line\* lines; //cache set包含的行

} Set;

typedef struct {

int set\_num; //组数

int line\_num; //行数

Set\* sets; //模拟cache

} Sim\_Cache;

6.1 Part A

enum TYPE calcu(char instr[]) //get result of the instruction

{

char ins;

long long address;

int siz;

sscanf(instr," %c %llx %d", &ins, &address,&siz);//输入部分的格式

int sel=GetS(address);//sel是set的值

int tag=GetT(address);//tag是tag

int i;

for(i=0;i<E;++i)//E是每个set 的line的个数

{

if(cache[sel][i].is\_valid==1 && cache[sel][i].tag==tag) //检查每一行 如果是vaild并tag相等则是找到了

{

if(ins=='M') //hit\*2 for modify 按照文件中的说明 前面char是M的时候 hit两次

++hits;

++hits;

update\_time(sel,i);//完成hit之后要更新访问时间

return HIT;//这种时候返回的TYPR就是HIT

}

}

++misses; //如果没有hit 那么misses加一

for(i=0;i<E;++i)

{

if(cache[sel][i].is\_valid==0) //如果没有vaild的部分 也就是没有空位

{

cache[sel][i].is\_valid=1;

cache[sel][i].tag=tag;

update\_time(sel,i);//强行挤一个位置

if(ins=='M') //miss first and hit later for modify

{

++hits;//如果是M的情况 hit两次 就是MISS\_HIT的情况

return MISS\_HIT;

}

else//如果不是M 直接miss

return MISS;

}

}

++evictions; //means no room,must do eviction

for(i=0;i<E;++i)

{

if(cache[sel][i].access\_time==0) //find the most early one 如果有可以替换的

{

cache[sel][i].tag=tag;

update\_time(sel,i);

if(ins=='M') //first miss and eviction and hit later for modify

{

++hits;

return MISS\_EVICTION\_HIT;

}

else

return MISS\_EVICTION;

}

}

return 0;

}

一：直接hit

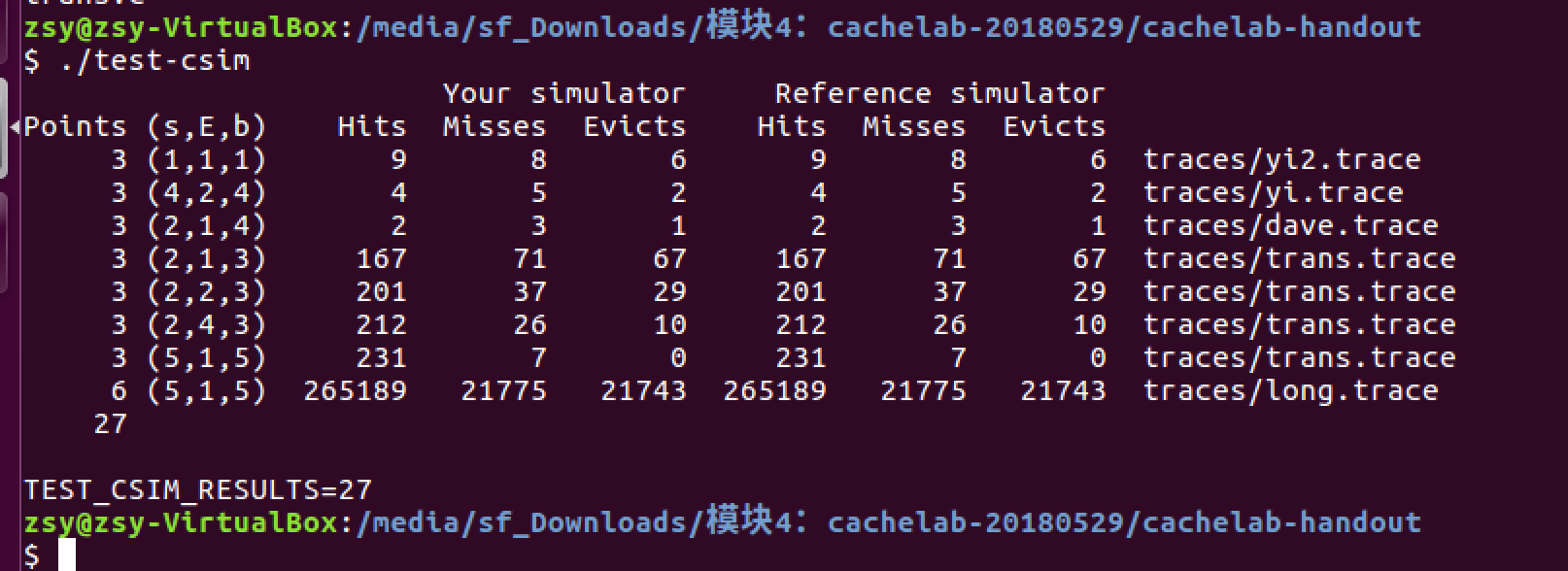
二：直接miss

三：先miss再hit

四：先miss再eviction

五：先miss再eviction再hit

测试结果：



LRU（least recently used)是将近期最不会访问的数据给淘汰掉，其实LRU是认为最近被使用过的数据，那么将来被访问的概率也多，最近没有被访问，那么将来被访问的概率也比较低“，其实这个并不是正确的，但是因为LRU算法简单，存储空间没有被浪费，所以还是用的比较广泛的