实验日志

一、valgrind 安装

由于实验网站给的安装包总是下载失败，于是我使用了命令行 sudo apt-get install valgrind 直接下载。输入指令 valgrind --log-fd=1 --tool=lackey -v --trace-mem=yes ls -l 检验是否成功。

```
==2466==
==2466== Counted 0 calls to main()
==2466==
==2466== Jccs:
==2466==   total:         173,966
==2466==   taken:         74,546 (43%)
==2466==
==2466== Executed:
==2466==   SBs entered:   184,212
==2466==   SBs completed: 120,651
==2466==   guest instrs:  989,070
==2466==   IRStmts:       5,884,905
==2466==
==2466== Ratios:
==2466==   guest instrs : SB entered  = 53 : 10
==2466==        IRStmts : SB entered  = 319 : 10
==2466==        IRStmts : guest instr = 59 : 10
==2466==
==2466== Exit code:       0
```

证明成功。

二、学会 PartA 部分参考引用 csim-ref 模拟器使用命令行参数的方法

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）。

```
z@z-virtual-machine:~/cache/cachelab-handout$  ./csim-ref -v -s 4 -E 1 -b 4 -t t
races/yi.trace
L 10,1 miss
M 20,1 miss hit
L 22,1 hit
S 18,1 hit
L 110,1 miss eviction
L 210,1 miss eviction
M 12,1 miss eviction hit
hits:4 misses:5 evictions:3
```

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()

1.

```
void get_opt(int argc, char **argv)
{
    int c;

    while ((c = getopt(argc, argv, "h:s:E:b:t:")) != -1)
    {
        switch (c)
        {
            case 's':
                checkOptarg(optarg);
                s = atoi(optarg);
                break;
            case 'E':
                checkOptarg(optarg);
                E = atoi(optarg);
                break;
            case 'b':
                checkOptarg(optarg);
                b = atoi(optarg);
                break;
            case 't':
                checkOptarg(optarg);
                file = optarg;
                break;
            case 'h':
            default:
                printHelpMenu();
                exit(0);
        }
    }
}
```

2.

```
/*打印帮助文档*/
void printHelpMenu(){
    printf("Usage: ./csim-ref [-hv] -s <num> -E <num> -b <num> -t <file>\n");
    printf("Options:\n");
    printf("-h         Print this help message.\n");
    printf("-v         Optional verbose flag.\n");
    printf("-s <num>   Number of set index bits.\n");
    printf("-E <num>   Number of lines per set.\n");
    printf("-b <num>   Number of block offset bits.\n");
    printf("-t <file>  Trace file.\n\n\n");
    printf("Examples:\n");
    printf("linux>  ./csim -s 4 -E 1 -b 4 -t traces/yi.trace\n");
    printf("linux>  ./csim -v -s 8 -E 2 -b 4 -t traces/yi.trace\n");
}
```

3.

```
/*检查参数合法性*/
void checkOptarg(char *curOptarg){
    if(curOptarg[0]=='-'){
        printf("./csim :Missing required command line argument\n");
        printHelpMenu();
        exit(0);
    }
}
```

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

1.
```c
void init_cache()
{
    int S = (1 << s);
    if ( S <= 0)
    {
        fprintf(stderr, "S is nonpositive\n");
        exit(0);
    }
    setptr = (struct oneSet*)malloc(sizeof(struct oneSet) * S);

    for (int ind = 0; ind < S; ++ind)
    {
        setptr[ind].v = (int *)malloc(sizeof(int) * E);
        setptr[ind].last_access_time = (clock_t *)malloc(sizeof(clock_t) * E);
        setptr[ind].tag = (long *)malloc(sizeof(long) * E);

        for(int Eind = 0 ; Eind < E; Eind++)
        {
            setptr[ind].v[Eind] = 0;
            setptr[ind].last_access_time[Eind] = 0;
            setptr[ind].tag[Eind] = 0;
        }
    }
}
```

2.
```c
void free_cache()
{
    int S = (1 << s);

    setptr = (struct oneSet*)malloc(sizeof(struct oneSet) * S);
    for (int ind = 0; ind < S; ++ind)
    {
        free(setptr[ind].v);
        free(setptr[ind].last_access_time);
        free(setptr[ind].tag);
    }

    free(setptr);
}
```

六、检验主函数代码以及结果分析

```c
int main(int argc, char **argv)
{
    get_opt(argc, argv);
    init_cache();

    FILE *fp = fopen(file, "r");
    if (fp == NULL)
    {
        fprintf(stderr, "open file error\n");
        exit(0);
    }

    char op[MAXLINE];
    void *addr;
    int size;
    char buf[MAXLINE];
    while (fgets(buf, MAXLINE, fp) != NULL)
    {
        //printf("\nline read: %s\n", buf);
        sscanf(buf, "%s %p,%d", op, &addr, &size);
        //printf("op = %s, addr = %p, size = %d\n", op, addr, size);
        if (*op == 'L')
        {
            do_L(addr, size);
        }
        else if (*op == 'M')
        {
            do_M(addr, size);
        }
        else if (*op == 'S')
        {
            do_S(addr, size);
        }
    }

    free_cache();
    printSummary(hits, misses, evicts);
    return 0;
}
```

```
Part A: Testing cache simulator
Running ./test-csim
                        Your simulator     Reference simulator
Points (s,E,b)    Hits  Misses  Evicts    Hits  Misses  Evicts
     3 (1,1,1)       9       8       6       9       8       6  traces/yi2.trace
     3 (4,2,4)       4       5       2       4       5       2  traces/yi.trace
     3 (2,1,4)       2       3       1       2       3       1  traces/dave.trac
e
     3 (2,1,3)     167      71      67     167      71      67  traces/trans.tra
ce
     3 (2,2,3)     201      37      29     201      37      29  traces/trans.tra
ce
     3 (2,4,3)     212      26      10     212      26      10  traces/trans.tra
ce
     3 (5,1,5)     231       7       0     231       7       0  traces/trans.tra
ce
     6 (5,1,5)  265189   21775   21743  265189   21775   21743  traces/long.trac
e
    27
```

分析：可以看到我编写的缓存模拟器与样例得分一致。