Homework3

陶静怡 18307130264

**实验（一） 基于 cache 的存储访问**

1.数组a分配在静态区



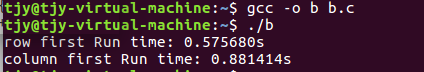
（1）M=1000，N=1000



（2）M=10，N=1000000

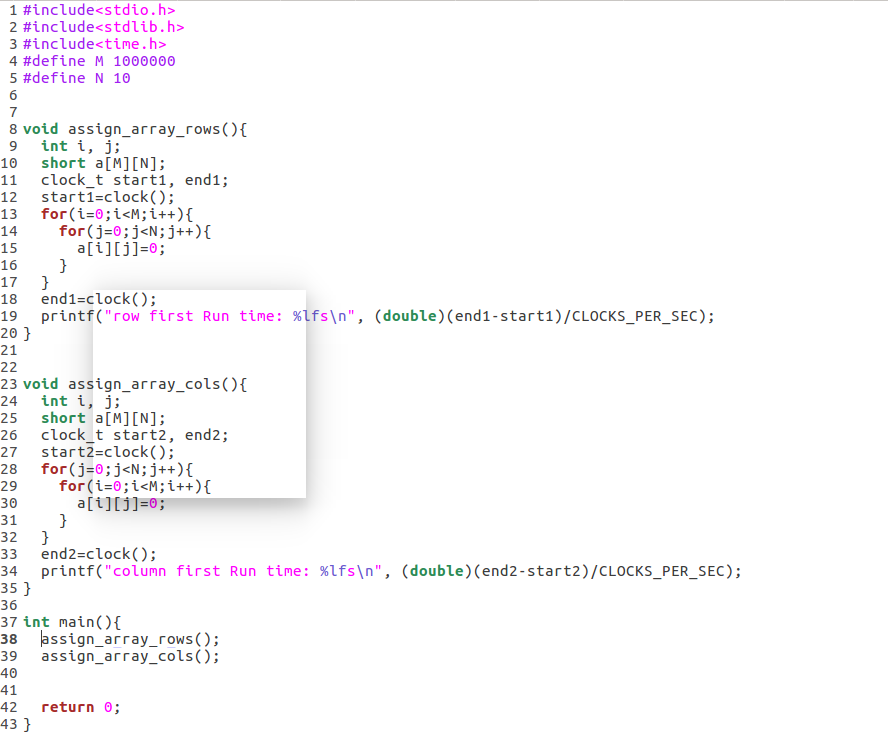


（3）M=1000000，N=10



数组a是在静态区，同时是按行优先存放的，根据执行时间，行优先访问所用的时间短于列优先，按行访问的空间局部性更好，cache的命中率更高。但是在M=10，N=1000000时，由于行数少，例如第一次循环取a[0][0]-a[9][0]所在的不同的10个块冲突少的话，第二次循环命中率会提高，所以导致时间上甚至比行优先快一点。但是可以看到在M=1000000，N=10时两者的差距是非常大的，所以总体来说行优先的执行速度快

2.数组a分配在栈区



1. M=1000000，N=10

运行以后发现出现段错误



（2）M=10，N=1000000



同样发现出错

（3）M=1000，N=1000



根据M=1000，N=1000的结果可以看到在数组a分配在栈区时行优先访问所用的时间更短，在栈区中同样是行优先存放的，所以空间局部性好，cache命中率更高

而segmentfault的原因我认为是局部变量过大，大于了系统给的栈的大小

所以修改了一下数据，发现可以运行

M=100000，N=10

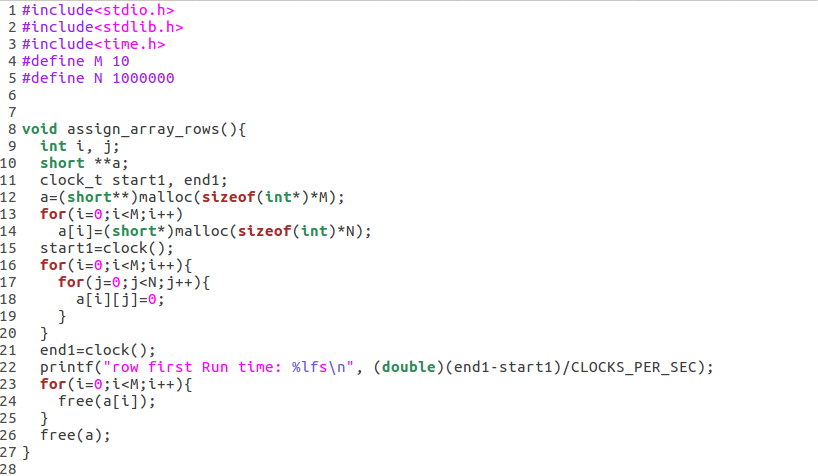


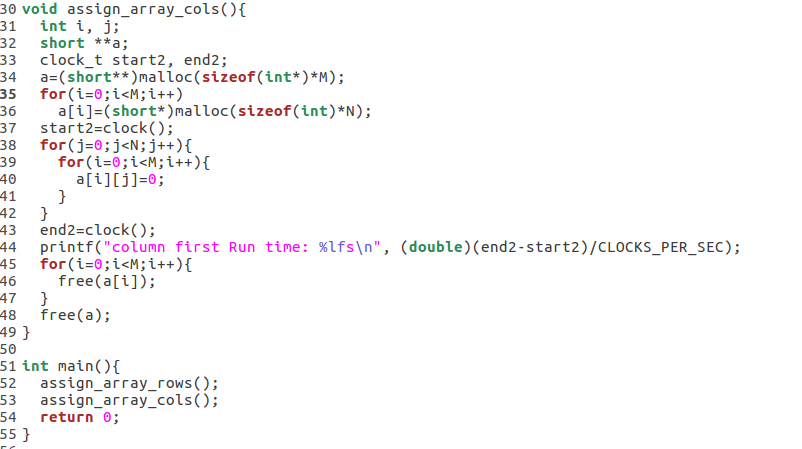
M=10，N=100000



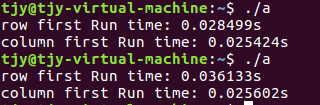
这时行优先所用的时间大于列优先，是由于行数和列数不同，不能很好体现行优先的空间局部性好

3.数组a分配在堆区





（1）M=10，N=1000000



（2）M=1000000，N=10



（3）M=1000，N=1000

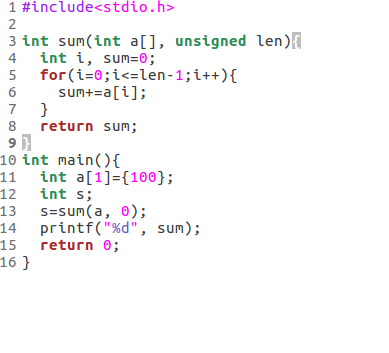


可以看到总体上是行优先访问所用时间短

局部数据块大可以更好利用空间局部性，一次能把更多数据放入cache中，从而提高命中率。但是如果块太大，读取的时间就比较长，同时会让cache行数变少，替换的频率增加。

数组访问顺序：因为数组在存储器中是行优先，所以总体上行优先访问执行时间短，因为空间局部性好，cache命中率高，所用的访存时间少。

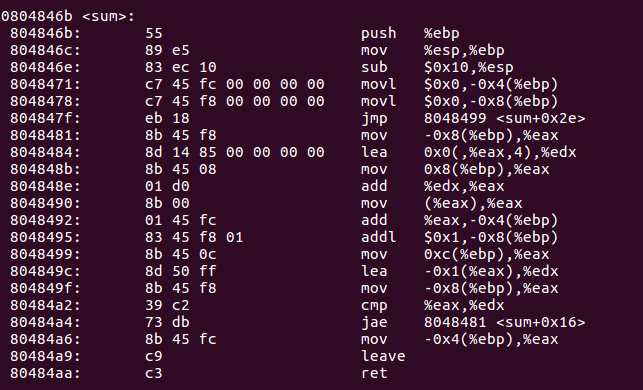
**实验（二） 存储保护**



Linux中无法运行，如图存储器访问异常



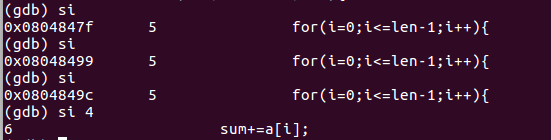
用 gdb 调试，确定发生异常的指令，并指出发生的是什么异常，以及发生访问违例的存储单元地址。



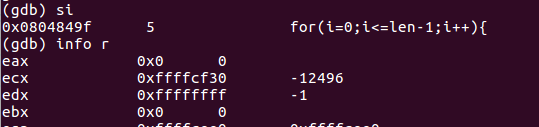
在sum函数处设置断点



发现满足了i<=len-1继续执行了



Eax中存放i，edx中存放len-1的值



执行com %eax，%edx

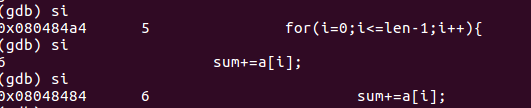
即把0xffffffff-0x0，sub=1， CF=0，ZF=0

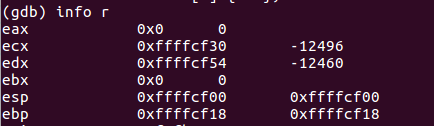
所以满足jae（0x80484a4处）的转移条件：CF=0/ZF=1

既这里当成无符号数比较大小，0xffffffff>0x0



所以跳转执行sum=+a[i]





访问到了内核区

从 CPU 检测到异常到屏幕中出现“Segment fault”的整个过程中，首先CPU产生一个访问异常，进程立即被暂停，然后执行OS页故障处理程序，然后这个程序发现进程确实访问越界了，就会发送SIGSEGV信号给出客户进程

