

# 中山大学计算机学院本科生实验报告

课程名称：超级计算机原理与操作

任课教师：吴迪

年级	19	专业（方向）	计算机科学与技术	
学号	19335174	姓名	施天予	
开始日期	2021-4-4	完成日期	2021-4-4	

## 一、实验题目

请使用 `pthread` 中的 `semaphore` 计算  $\pi$  的值（参考课件 5 page 46）。

计算  $\pi$  的方法可以参考课件中的方法，在参考代码中提供了运行所需的主函数。

请同学们下载附件 `homework3.zip` 将并行的代码补充完整并完成实验报告。

## 二、实验内容

当多个线程尝试更新同一个共享变量时，结果是无法预测的。因此需要一个临界区，一次允许只一个线程执行该段代码。信号量 `sem` 初始化为 1，开锁状态。在要保护的临界区前调用 `sem_wait`，线程执行到 `sem_wait` 函数时，如果信号量为 0，线程就会被阻塞。如果信号量是非 0 值，就减 1 后进入临界区。临界区使用 `sum += my_sum` 求得总和。再调用 `sem_post` 对信号量的值加 1，使得在 `sem_wait` 中阻塞的其他线程能够继续运行。

## 三、实验结果

默认是 `n = 10000`，`thread_count = 4`

```
hw3_pthread.c - 无标题 (工作区) - Visual Studio Code
文件(F) 编辑(E) 选择(S) 查看(V) 转到(G) 运行(R) 终端(T) 帮助(H)

C hw3_pthread.c x
Pthreads > C hw3_pthread.c > main(int, char *[])

100 void* Thread_sum(void* rank) {
101     long my_rank = (long) rank;
102     double my_sum = 0.0;
103
104     /******
105     double factor;
106     long long i;
107     long long my_n = n/thread_count;
108     long long my_first_i = my_n*my_rank;
109     long long my_last_i = my_first_i + my_n;
110     if (my_first_i % 2 == 0)
111         factor = 1.0;
112     else
113         factor = -1.0;
114     for (i = my_first_i; i < my_last_i; i++, factor = -factor) {
115         my_sum += factor/(2*i+1);
116     }
117     sem_wait(&sem);
118     sum += my_sum;
119     sem_post(&sem);
120     /******
121
122
123     return NULL;
124 } /* Thread_sum */
```

```
sty@ubuntu: ~/Parallel/Pthreads
sty@ubuntu:~$ cd ~/Parallel/Pthreads
sty@ubuntu:~/Parallel/Pthreads$ gcc -g -Wall -o hw3_pthread hw3_pthread.c -lpthread
sty@ubuntu:~/Parallel/Pthreads$ ./hw3_pthread
With n = 10000 terms,
  Our estimate of pi = 3.141492653590044
The elapsed time is 3.678799e-04 seconds
  Single thread est = 3.141492653590034
The elapsed time is 2.217293e-05 seconds
                    pi = 3.141592653589793
sty@ubuntu:~/Parallel/Pthreads$
```

可以看到，单线程的计算比多线程快一些，虽然两者的计算都不是很精准。