# 操作系统实验报告

华中科技大学

姓名： 易子闳

学号： U201613634

班级： 种子1601

时间： 2018.02.04

## 实验目的

理解多进程和多线程，利用多进程和多线程实现累加和计算，理解进程通信等机制；

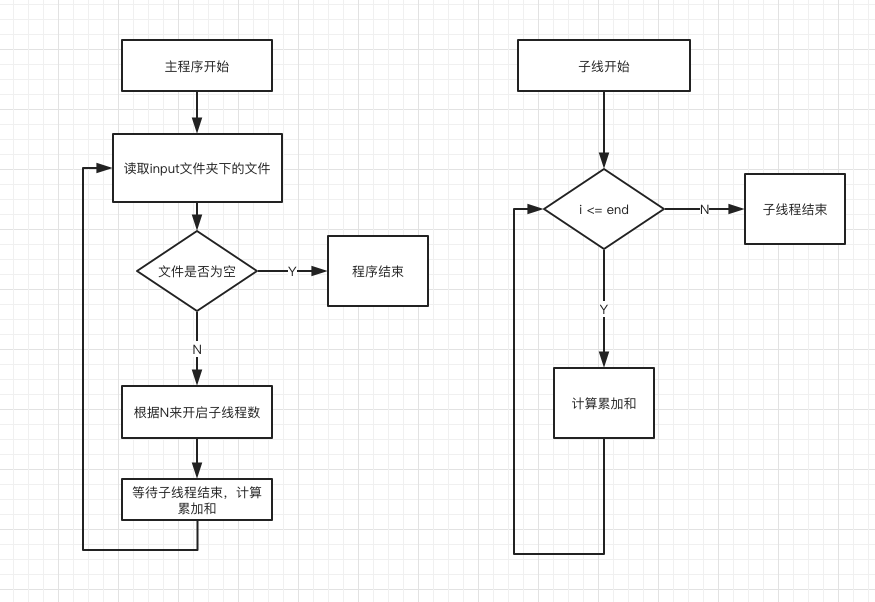
## 实验环境

1. 电脑环境：VMVare Fusion 11.0.2
2. 系统环境: Ubuntu 18.04.1 LTS
3. 运行终端: Ubuntu bash

## 程序设计

### 多线程

* 流程图



* 流程描述

从input文件夹下循环读取input.txt的N，M的值，根据N的值来创建子线程，计算累加和，然后将累加和输出到output文件夹下创建txt文件。

* 代码

//创建N个子线程

for (int i = 0; i < N; i++)

{

indexes[i] = i;

int ret = pthread\_create(&pthread\_id[i], NULL, sum\_work, (void \*)&indexes[i]);

// printf("i的值为：%d\n", i);

}

for (int i = 0; i < N; i++)

{

//等待子线程结束，如果该子线程已经结束，则立即返回

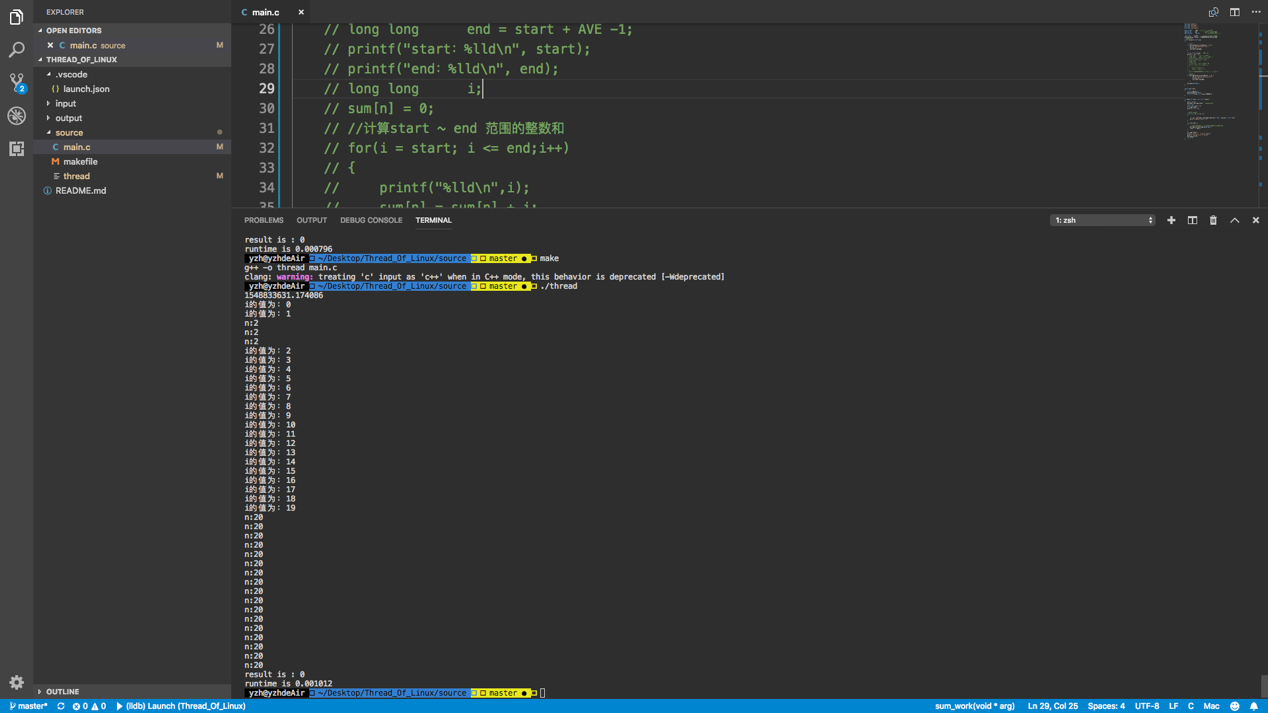
pthread\_join(pthread\_id[i], NULL);

result += sum[i];

}

* 遇到的问题

线程没有加锁，导致访问同一个内存，最后计算结果出错。



解决方法：添加线程锁或者定义另一变量存储各线程需要同时访问的变量。

int indexes[N]; //暂存n

//创建N个子线程

for (int i = 0; i < N; i++)

{

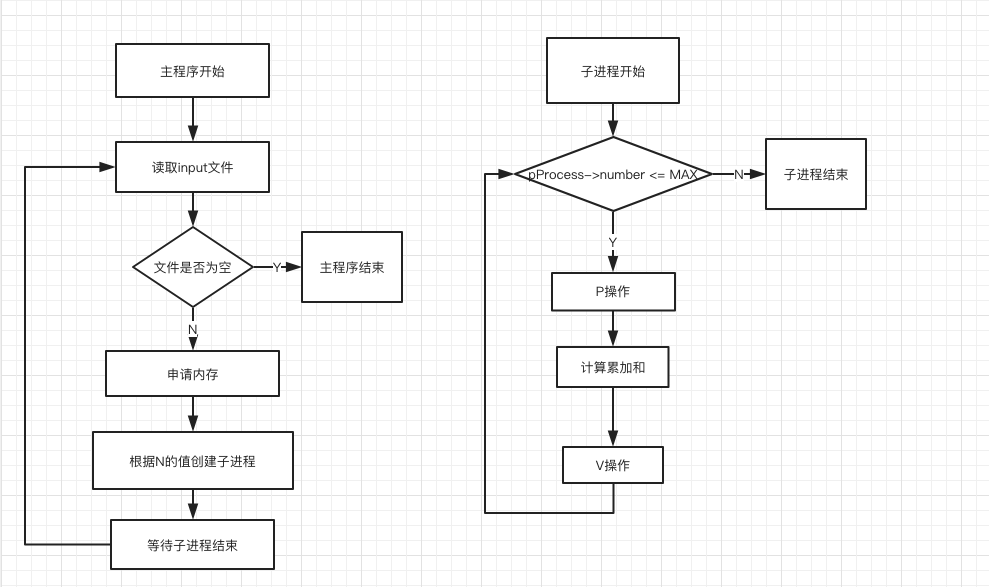
indexes[i] = i;

int ret = pthread\_create(&pthread\_id[i], NULL, sum\_work, (void \*)&indexes[i]);

}

### 多进程

* 流程图



* 流程描述

从input文件夹一次读取txt中N，M的值，根据N的值创建进程，每个子进程在进行累加时需要进行P操作，累加完以后进行V操作。直到所有的子进程结束以后打印最后的结果。

* 代码

while (pProcess->number <= MAX) {

sem\_wait(&pProcess->signal);

if (pProcess->number <= MAX) {

pProcess->sum += pProcess->number++;

}

sem\_post(&pProcess->signal);

}

父进程等待子进程退出,然后销毁信号量,将结果输出到output/output.txt.

* 传统累加方法验证结果

long correct\_answer()

{ long answer = 0;

for (int i = 0; i <= MAX; i++)

{

answer = answer + i;

}

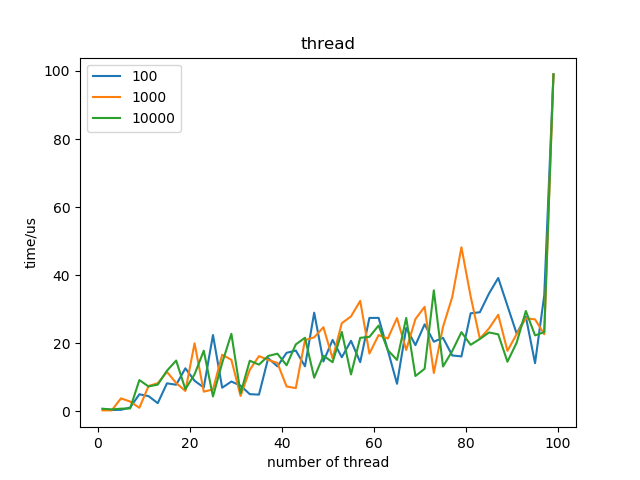
return answer;

}

## 结果分析

### 多线程

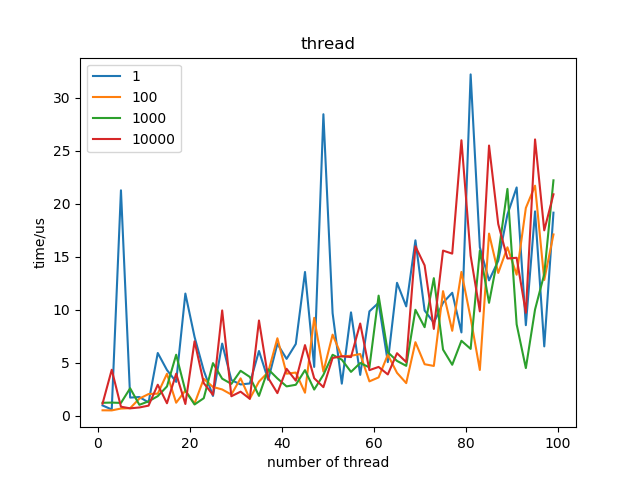
利用python在input文件夹里创建txt输入文件，然后调用./multisum,循环读取input文件下N和M的值，每次的运算结果储存在output文件夹下，运算结果储存在timeresult下，然后读取，画出如下图标。



1. 有上图可知，M的值越大，计算消耗的时间越长；
2. 开启相同线程数量时，累加的计数越大耗时越长；
3. 开启多个线程时，线程需要需要阻塞，所以花费的时间会变长。

### 多进程

利用python在input文件夹里创建txt输入文件，然后调用./multisum,循环读取input文件下N和M的值，每次的运算结果储存在output文件夹下，运算结果储存在timeresult下，然后读取，画出如下图标。



(1)由上图可发现，开启相同进程数量时，累加的计数越大耗时越长.

(2)当进程数较小时，消耗的时间会更少；

(3)由于进程在fork进程后，我们在进行累加时，需要进行PV操作，会影响计算的效率。

## 总结

通过这次课设，我初步理解了多线程和多进程，了解了线程锁，PV操作，锻炼了自己实践能力，和复杂问题逐步分解能力。研究了线程/进程数，最大数的大小等因素对于运行时间的影响，并对其内在原因进行了探究。