大作业 多线程编程

一、目的

熟悉 Linux 系统线程相关的系统调用。

二、内容

多线程方法计算浮点向量的点积。

三、要求

1. 在 Linux 平台上用多线程方法实现**浮点**向量的点积计算：$s=\sum_{i=0}^{N} a_i b_i$。向量$\{a_i\}$和$\{b_i\}$的元素满足以下规则：$a_i = b_i = \begin{cases} 1.0 & (i \bmod 3 = 0) \\ -1.0 & (i \bmod 3 = 1) \\ 0 & (i \bmod 3 = 2) \end{cases}$

2. 程序的输入命令行：

```
vec_mul thread_num N

参数：
thread_num：线程数，从 1 到 16
N：向量长度，不少于 100,000
```

3. 输出格式：

```
s=xxxx t=tttt(ms)



说明：
xxxx：点积计算结果
tttt：计算所需要时间（不包括向量初始化的时间），以毫秒为单位
```

4. 计算结果和计算时间

Thread_num	N	计算结果	计算时间
1			
1			
1			
1			
2			
4			
8			
16			

请根据上述实验结果，画出两个图。其中图 1 的 X 轴为 N，Y 轴为 Thread_num 为 1 时的计算时间，图 2 的 X 轴为 Thread_num，Y 轴为 N 为 100,000,000 时的计算时间。

四、所用函数

1. 创建线程函数：

```
int pthread_create(
    pthread_t *thread,              // 线程标识符指针
    const pthread_attr_t *attr, // 线程属性（通常为 NULL）
    void *(*start_routine)(void *), // 线程函数指针
    void *arg                       // 传递给线程函数的参数
);
```

2. 等待指定线程终止函数：

```
int pthread_join(
    pthread_t thread,   // 等待的线程标识符
    void **retval       // 存储线程返回值的指针
);
```

3. 获取当前的系统时间函数：

```
int gettimeofday(
    struct timeval *restrict tv,   // 存储时间的结构体
```

```
    struct timezone *restrict tz    // 已废弃的时区参数（通常设为 NULL）
);
struct timeval {
    time_t          tv_sec;    // 秒数（从 1970-01-01 00:00:00 UTC 开始的秒数）
    suseconds_t    tv_usec; // 微秒数（0 到 999,999 之间的值）
};
```