《Linux 之下软件开发与分析》实验二

- 题目：利用多线程编程实现矩阵与向量的乘积运算
- 要求：利用 gcc makefile 机制管理项目，分成四个模块：全局数据定义及生成模块；线程函数定义及实现模块，该模块包含同步全局变量的定义；线程创建模块；主模块。把矩阵的行数和列数定义为全局数据模块接口文件的宏，画出运行时间与矩阵大小的时间复杂度试验图标，并提出理论解释。
- 班级：软件 15 级，软件 zy15 级，软件 sy15 级
- 时间：2018-19-1 第 10 周周五 12:00-13:35，地点：鉴主 10 楼计算机学院机房
- 任课老师及指导老师：祁明龙
- 参考代码：
  - 全局数据定义及生成模块

```
Administrator@CN-20160602JVJP ~/MThread
$ cat pthreaddef.h
#ifndef PTHREADDEF_H_INCLUDED
#define PTHREADDEF_H_INCLUDED
#include <stdlib.h>
#include <stdio.h>
#define NROW    5
#define NCOL    5
double v1[NROW];
double v2[NCOL];
double mx[NROW][NCOL];
void fill_data();
void print_vec(int,double*);
void print_mx();
void test_data();

#endif // PTHREADDEF_H_INCLUDED
```

  - 全局数据定义及生成模块实现

```
Administrator@CN-20160602JVJP ~/MThread
$ cat data_impl.c
#include "pthreaddef.h"
void fill_data()
{
    int i,j;
    for(i=0;i<NCOL;i++)
    {
        v2[i]=5;//(double)rand();
    }

    for(i=0;i<NROW;i++)
    {
        for(j=0;j<NCOL;j++)
        {
            if(i==j)mx[i][j]=1;
            else
                mx[i][j]=0;
            //mx[i][j]=(double)rand();
        }
    }
}
```

```
void print_mx()
{
    int i,j;
     for(i=0;i<NROW;i++)
    {
        for(j=0;j<NCOL;j++)
        {
            printf("%f\t",mx[i][j]);;
        }
        printf("\n");
    }
}
void print_vec(int nelm,double * arr)
{
    int i;
    for(i=0;i<nelm;i++)
    {
        printf("%f\t",*(arr+i));
    }
    printf("---\n");
}
```

```
void test_data()
{
    fill_data();
    print_mx();
    print_vec(NROW,v1);
    print_vec(NCOL,v2);
}
```

- 线程函数接口文件

```
Administrator@CN-20160602JVJP ~/MThread
$ cat tfunc.h
#ifndef TFUNC_H_INCLUDED
#define TFUNC_H_INCLUDED
#include <pthread.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include "pthreaddef.h"
pthread_t tworkers[NROW];
pthread_mutex_t lock;
void *thread_func(void* tdata);


#endif // TFUNC_H_INCLUDED
```

- 线程函数实现模块

```
$ cat tfunc_impl.c
#include "tfunc.h"
void* thread_func(void* tdata)
{
    int i,j;
    i=(int)tdata;
    //pthread_t actualThread;
    //actualThread=pthread_self();
    //if(actualThread!=tworkers[i]) return;
    pthread_mutex_lock(&lock);
    v1[i]=0;
    for(j=0;j<NCOL;j++)
    {
        v1[i]+=mx[i][j]*v2[j];

    }
    //print_mx();
    //print_vec(NCOL,v2);
    printf("v1[%d]=%lf\n",i,v1[i]);
    pthread_mutex_unlock(&lock);
}
```

- 线程创建模块接口文件

```
Administrator@CN-20160602JVJP ~/MThread
$ cat job.h
#ifndef JOB_H_INCLUDED
#define JOB_H_INCLUDED
#include "tfunc.h"
int dojob();


#endif // JOB_H_INCLUDED
```

- 线程创建模块实现文件

```
Administrator@CN-20160602JVJP ~/MThread
$ cat job_impl.c
#include "job.h"
int dojob()
{
    int i;
    fill_data();
    if(pthread_mutex_init(&lock,NULL)<0)
    {
        perror("Cannot initialize the mutex");
        exit(1);
    }

       for(i=0;i<NROW;i++)
       {
           if(pthread_create(&tworkers[i],NULL,thread_func,(void*)i)!=0)
           {
               perror("Cannot create threads");
               exit(2);
           }
       }
```

```
       for(i=0;i<NROW;i++)
       {
           pthread_join(tworkers[i],NULL);
       }
       pthread_mutexattr_destroy(lock);
       print_vec(NROW,v1);

       return 0;

}
```

- 主模块

```
Administrator@CN-20160602JVJP ~/MThread
$ cat main.c
#include "pthread.h"
#include "tfunc.h"
#include "job.h"
int main()
{
    //test_data();
    dojob();
    return 0;
}
```

```
Administrator@CN-20160602JVJP ~/MThread
$ cat makefile
OBJS = data_impl.o tfunc_impl.o job_impl.o main.o
prog:$(OBJS)
        gcc -o $@ $^
main.o:main.c pthreaddef.h tfunc.h job.h
        gcc -c $<
data_impl.o:data_impl.c pthreaddef.h
        gcc -c $<
tfunc_impl.o:tfunc_impl.c pthreaddef.h tfunc.h
        gcc -c $<
job_impl.o:job_impl.c pthreaddef.h tfunc.h job.h
        gcc -c $<
clean:
        rm *.o *.exe
```

- make“一下”

```
Administrator@CN-20160602JVJP ~/MThread
$ make
gcc -c job_impl.c
gcc -o prog data_impl.o tfunc_impl.o job_impl.o main.o
```

- 测试软件

```
Administrator@CN-20160602JVJP ~/MThread
$ ./prog
v1[0]=5.000000
v1[1]=5.000000
v1[2]=5.000000
v1[3]=5.000000
v1[4]=5.000000
5.000000        5.000000        5.000000        5.000000        5.000000
--
```

- 思考题：该程序中创建了几个线程？每个线程的计算任务是什么？