姓名：张文沁

学号：20337268

4.2 在什么情况下，采用多内核线程的多线程方法比单处理器系统的单线程，提供更好的性能

> 当一个内核线程的页面发生错误时，另外的内核线程会用一种有效的方法被转换成使用交错时间。
>
> 另一方面，当页面发生错误时，一个单一线程进程将不能够发挥有效性能。
>
> 因此，在一个程序可能有频繁的页面错误或不得不等待其他系统的事件的情况下，多线程方案会有比单处理器系统更好的性能。
>
> 多线程技术使程序响应速度更快，因为用户界面可以在进行其他工作的同时一直处于活动状态。而且可以分配优先级为了更好地分配CPU

4.3 在同一进程的多线程之间，下列哪些程序状态部分会被共享?

a 寄存器值 b 堆内存 c 全局变量 d 堆栈内存

> 一个线程程序的线程共享堆内存和全局变量，但每个线程都有属于自己的一组寄存值和栈内存。所以BC会被共享

4.7 设有一个应用，60%为并行部分，而处理核数量分别为 (a) 2个 和 (b) 4个。利用Amdahl定律，计算加速增益

$$加速比 <= 1/(S + (1 - S)/N)$$

> (a) 加速比<= 10/7 （1.428）
>
> (b) 加速比<= 20/11 （1.818）

4.9

> 1. 一个
>
>   只有创建尽可能多的线程来阻止系统调用才有意义，创建额外的线程没有任何好处。因此，创建一个单线程用于输入和输出即可。
>
> 2. 四个。
>
>   线程的数量应与处理内核的数量相同。多了则无法运行，少了则是对计算机资源的浪费。

4.13

> a. 当内核线程数小于处理器数时，一些处理器将保持空闲，因为调度器只将内核线程映射到处理器，而不将用户级线程映射到处理器。
>
> b. 当内核线程的数量正好等于处理器的数量时，它就是所有处理器可能同时使用。然而，当内核线程阻塞时在内核内部（由于页面错误或调用系统调用时），相应的处理器将保持空闲。
>
> c. 当内核线程多于处理器时，可以交换被阻塞的内核线程支持另一个准备好执行的内核线程，从而提高多处理机系统。

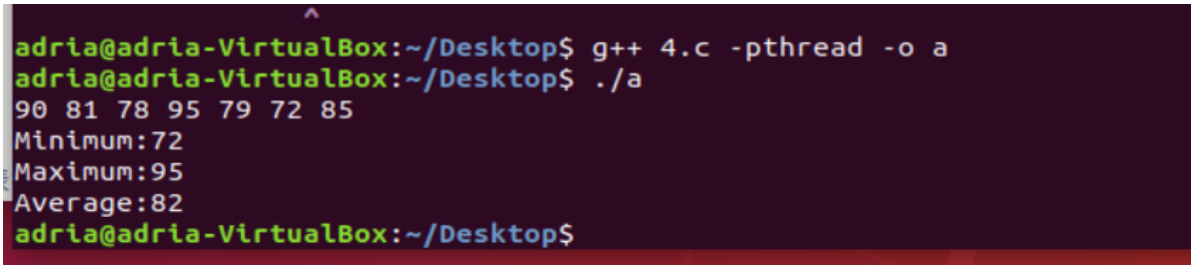
4.16

代码：

```
#include <thread>
#include <stdio.h>
#include <pthread.h>
int minValue = 0;
int maxValue = 0;
int averageValue = 0;

void mmax(int a[], int size)
{
    for (int i = 0; i < size; ++i)
    {
        if (a[i] > maxValue)
        {
            maxValue = a[i];
        }
    }
}
void mmin(int a[], int size)
{
    minValue = 9999;
    for (int i = 0; i < size; ++i)
    {
        if (a[i] < minValue)
        {
            minValue = a[i];
        }
    }
}
void average(int a[], int size)
{
    int sum = 0;
    for (int i = 0; i < size; ++i)
    {
        sum += a[i];

    }
    if (size > 0)
    {
        averageValue = sum / size;
    }
}

int main()
{
    int a[8];
    for (int i = 0;i < 7;++i){
        int sum = 0;
        scanf("%d",&sum);
        a[i] = sum;
    }
    std::thread calcMinWorkThead(mmin, a, 7);
    std::thread calcMaxWorkThead(mmax, a, 7);
    std::thread calcAverageWorkThead(average, a, 7);
    calcMinWorkThead.join();
    calcMaxWorkThead.join();
```

```
    calcAverageWorkThead.join();
    printf("Minimum:%d\nMaximum:%d\nAverage:%d\n", minValue, maxValue,
averageValue);
    return 0;
}
```

结果：



项目2：多线程排序应用程序

代码：

```
#include<stdio.h>
#include<pthread.h>
#include<unistd.h>

#define ARRAY_COUNT 10
int need_sort_array[10]={7,12,19,3,18,4,2,6,15,8};
int sorted[10];

void *worker(void *arg){
    int i,j;
    int tmp;
    int Min;
    for(i=0;i<ARRAY_COUNT/2;i++){
            Min=i;
        for(j=i+1;j<ARRAY_COUNT/2;j++){
            if(need_sort_array[Min]>need_sort_array[j])
                    Min=j;
        }
        tmp=need_sort_array[i];
        need_sort_array[i]=need_sort_array[Min];
        need_sort_array[Min]=tmp;
    }
}


void master(){
    int i,j;
    int tmp;
    int Min;
    for(i=ARRAY_COUNT/2;i<ARRAY_COUNT;i++){
            Min=i;
        for(j=i+1;j<ARRAY_COUNT;j++){
            if(need_sort_array[Min]>need_sort_array[j])
                    Min=j;
        }
        tmp=need_sort_array[i];
```

```
        need_sort_array[i]=need_sort_array[Min];
        need_sort_array[Min]=tmp;
    }
}

void Merge_Array(){
        int i=0,j=ARRAY_COUNT/2,k=0;
        while(i<ARRAY_COUNT/2&&j<ARRAY_COUNT){
            if(need_sort_array[i]<need_sort_array[j])
                    sorted[k++]=need_sort_array[i++];
            else
                    sorted[k++]=need_sort_array[j++];
        }
        while(i<ARRAY_COUNT/2)
                sorted[k++]=need_sort_array[i++];
        while(j<ARRAY_COUNT)
                sorted[k++]=need_sort_array[j++];
}


void print(int *array){
        int i;
        for(i=0;i<ARRAY_COUNT;i++)
                printf("%d ",array[i]);
        printf("\n");
}


int main(){
    pthread_t pid;
    printf("Before sort,need_sort_array: ");
    print(need_sort_array);
    pthread_create(&pid,NULL,&worker,NULL);
    master();
    pthread_join(pid,NULL);
    printf("After sorted,need_sort_array: ");
    print(need_sort_array);
    Merge_Array();
    printf("sorted array: ");
    print(sorted);
    return 0;
}
```

结果:

```
adria@adria-VirtualBox:~/Desktop$ g++ 42.c -pthread -o 42
adria@adria-VirtualBox:~/Desktop$ ./42
Before sort,need_sort_array: 7 12 19 3 18 4 2 6 15 8
After sorted,need_sort_array: 3 7 12 18 19 2 4 6 8 15
sorted array: 2 3 4 6 7 8 12 15 18 19
adria@adria-VirtualBox:~/Desktop$
```

问题：

1. 编译过程缺少代码导致失败，后续加上了 -pthread

2. 一开始直接导入库“#include<pthread.h>”却报错，因为lpthread是链接库，<pthread.h>只 有申明，实现部分都在库里面。创建线程时一般是把函数的指针做参数，所以要加一个取 地址符号。建议要检查一下创建线程的返回值ret是否成功。