Project 3-1: Multithreaded Sorting Application

白骐硕 518030910102

任务概述

使用多线程编程进行数组排序：

1. 将输入的数组分为两部分，创建两个子线程对这两个子数组进行排序操作。
2. 创建第三个线程对排序好的两个子数组进行合并，并输出。

具体实现

本任务代码使用c语言编写，具体细节介绍如下文。

读入数组

首先使用scanf函数对用户输入的数组进行读入。用户需要指定数组中元素的个数（个数需要大于1，小于100），之后再进行数组的输入。具体代码如下：

```
//get input
while(array_size<=1 || array_size>=100){
    printf("Input the number of elements(2~99): ");
    scanf("%d", &array_size);
}
printf("Input the elements:");
for(int i=0; i != array_size; ++i){
    scanf("%d", &array1[i]);
}
```

子线程调用冒泡排序

子线程调用冒泡排序函数对分配给该线程的部分数组进行排序。子线程的工作函数的声明，和传递参数类型的定义如下：

```
void *bubbleSort(void *param);

typedef struct{
    int low;
    int high;
}para;
```

在这里，为了一次性传递两个int类型的参数，定义了一个结构体包含两个int类型。其中，low为子数组第一个元素的下标，high为子数组最后一个元素的下标加1。

冒泡函数 bubbleSort 的实现如下：

```
//Bubble Sort
void *bubbleSort(void *param){
    int low = ((para *) param)->low;
    int high = ((para *) param)->high;
```

```
    int i, j, tmp;
    int flag;
    for(i = 1; i<(high - low); ++i){
        flag = 0;
        for(j = low; j<high - i; ++j){
            if(array1[j+1] < array1[j]){
                tmp = array1[j];
                array1[j] = array1[j+1];
                array1[j+1] = tmp;
                flag = 1;
            }
        }
        if(!flag) break;
    }

    pthread_exit(0);
}
```

最后，在main函数中创建两个子线程，分别调用上述冒泡排序函数处理输入数组的一半。

```
pthread_t tid[3];
pthread_attr_t attr;
//prepare the parameters for runner of thread
para param[2];
param[0].low = 0;
param[0].high = array_size/2;
param[1].low = array_size/2;
param[1].high = array_size;

//get the default attributes
pthread_attr_init(&attr);

//create the threads
for(int i=0; i!=2; ++i){
    pthread_create(&tid[i], &attr, bubbleSort, &param[i]);
}
//wait for the threads exit
for(int i=0; i!=2; ++i){
    pthread_join(tid[i], NULL);
}
```

合并两个子数组

当上述两个子线程完成各自的排序任务后，创建一个新的线程，用于合并两个排序好的子数组。

合并函数的声明如下：

```
void *merge(void *param);
```

其所需的参数只有数组的中间下标，函数的实现如下图所示：

```
void *merge(void *param){
    int p1 = 0;
    int p2 = *((int *) param);
    int p3 = 0;
```

```
    while(p1 != array_size/2 || p2 != array_size){
        if(p1 == array_size/2){
            while (p2 != array_size)
            {
                array2[p3++] = array1[p2++];
            }
            break;
        }
        if(p2 == array_size){
            while(p1 != array_size/2){
                array2[p3++] = array1[p1++];
            }
            break;
        }
        if(array1[p1] < array1[p2]){
            array2[p3++] = array1[p1++];
        }else{
            array2[p3++] = array1[p2++];
        }
    }

    pthread_exit(0);
}
```

在main函数中创建新的线程调用上述函数进行合并：

```
int middle = array_size/2;
//create therad 2 to merge
pthread_create(&tid[2], &attr, merge, &middle);
//wait
pthread_join(tid[2], NULL);
```

至此，输入的数组被完全正确的排序。

实验结果

测试所用的输入数组为 7，12，19，3，18，4，2，6，15，8.

```
qsbai@qsbai-virtual-machine:~/os-prj/3/3-1$ ./multithreaded_sort
Input the number of elements:10
Input the elements:7 12 19 3 18 4 2 6 15 8
Thread 0:
3 7 12 18 19
Thread 1:
2 4 6 8 15
Sorted list:
2 3 4 6 7 8 12 15 18 19
```