

Project3 实验报告

余北辰 519030910245

1 实验概述

1.1 实验名称

Multithreaded Sorting Application & Fork-Join Sorting Application

1.2 实验内容

1. 使用c实现多线程排序，先用两个线程分别进行排序，再用第三个线程将其归并
2. 使用java实现Fork-Join排序，分别实现快速排序和归并排序

2 实验环境

- Ubuntu 18.04.5 LTS
- Linux version 5.4.0-72-generic
- VirtualBox 6.1.18

3 实验过程与结果展示

3.1 Multithreaded Sorting Application

思路：

1. 先将输入数组分割为两半
2. 创建两个线程，分别进行希尔排序
3. 再创建第三个线程，将已排好的两个线程归并

实现的代码如下：

```
1 int *num, *ans;  
2 int begin, mid, end;  
3 void *sorting(void *param);
```

```

4 void *merging(void *param);
5 int main(int argc, char *argv[])
6 {
7     pthread_t tid1, tid2, tid3;
8     pthread_attr_t attr;
9     begin = 0;
10    end = argc - 1;
11    mid = (begin + end) / 2;
12
13    num = (int *)malloc(end * sizeof(int));
14    ans = (int *)malloc(end * sizeof(int));
15
16    void *flag_1 = "1";
17    void *flag_2 = "2";
18
19    if (argc < 2)
20    {
21        fprintf(stderr, "Please input at least one number!\n");
22        return -1;
23    }
24    for (int i = 1; i <= end; ++i)
25    {
26        num[i - 1] = atoi(argv[i]);
27        if (num[i - 1] == 0 && strcmp(argv[i], "0") != 0)
28        {
29            fprintf(stderr, "Please input valid numbers!\n");
30            return -1;
31        }
32    }
33
34    pthread_attr_init(&attr);
35    pthread_create(&tid1, &attr, sorting, flag_1);
36    pthread_create(&tid2, &attr, sorting, flag_2);
37    pthread_join(tid1, NULL);
38    pthread_join(tid2, NULL);
39    pthread_create(&tid3, &attr, merging, NULL);
40    pthread_join(tid3, NULL);
41
42    for (int i = begin; i < end; ++i)
43    {
44        printf("%d ", ans[i]);
45    }
46    printf("\n");

```

```
47     free(num);
48     free(ans);
49     return 0;
50 }
51
```

sorting() 函数的实现如下:

```
1  void *sorting(void *param)
2  {
3      int left, right;
4      if (atoi(param) == 1)
5      {
6          left = begin;
7          right = mid;
8      }
9      else
10     {
11         left = mid;
12         right = end;
13     }
14     int len = right - left;
15     int gap;
16     int tmp;
17     int k;
18     for (gap = len / 2; gap > 0; gap /= 2)
19     {
20         for (int i = left; i < right; ++i)
21         {
22             int tmp = num[i];
23             int j = i - gap;
24             while(j >= left && tmp <= num[j])
25             {
26                 num[j + gap] = num[j];
27                 j -= gap;
28             }
29             num[j + gap] = tmp;
30         }
31     }
32     pthread_exit(0);
33 }
```

merging() 函数的实现如下:

```
1  void *merging(void *param)
```

```

2  {
3      int a = begin, b = mid;
4      int i = begin;
5      while (a < mid && b < end)
6      {
7          if (num[a] <= num[b])
8          {
9              ans[i] = num[a];
10             i++;
11             a++;
12         }
13         else
14         {
15             ans[i] = num[b];
16             i++;
17             b++;
18         }
19     }
20     while (a < mid)
21     {
22         ans[i] = num[a];
23         i++;
24         a++;
25     }
26     while (b < end)
27     {
28         ans[i] = num[b];
29         i++;
30         b++;
31     }
32     pthread_exit(0);
33 }

```

测试结果如下：

```

polaris@polaris-VirtualBox: ~/course/Operating-Systems/Project/Project3/Multithreaded-Sorting-Application
polaris@polaris-VirtualBox:~/course/Operating-Systems/Project/Project3/Multithreaded-Sorting-Application$ ./sorting 3 1 7 9 5 2 10 6 8 4
1 2 3 4 5 6 7 8 9 10
polaris@polaris-VirtualBox:~/course/Operating-Systems/Project/Project3/Multithreaded-Sorting-Application$

```

3.2 Fork-Join Sorting Application

由于没有java的基础，所以我选择对源代码中数组求和的代码进行修改。

快速排序

使用 `fork` 和 `join` 函数代替c中的 `pthread_create` 和 `pthread_join` 函数。

```
1  import java.util.concurrent.*;
2
3  public class Quicksort extends RecursiveAction {
4
5      static final int THRESHOLD = 5;
6
7      private int begin;
8      private int end;
9      private int[] array;
10
11     public Quicksort(int begin, int end, int[] array) {
12         this.begin = begin;
13         this.end = end;
14         this.array = array;
15     }
16
17     protected void compute() {
18         if (end - begin < THRESHOLD) {
19             for (int i = begin; i < end; ++i) {
20                 for (int j = begin; j < end; ++j) {
21                     int tmp;
22                     if (array[j] > array[j + 1]) {
23                         tmp = array[j + 1];
24                         array[j + 1] = array[j];
25                         array[j] = tmp;
26                     }
27                 }
28             }
29         } else {
30             int tmp = array[begin];
31             int i, j;
32             i = begin;
33             j = end;
34
35             while (i != j) {
36                 while (j > i && array[j] >= tmp) {
37                     --j;
38                 }
39                 if (i < j) {
```

```

40         array[i] = array[j];
41         ++i;
42     }
43     while (i < j && array[i] < tmp) {
44         ++i;
45     }
46     if (i < j) {
47         array[j] = array[i];
48         --j;
49     }
50 }
51 array[i] = tmp;
52 Quicksort leftTask = new Quicksort(begin, i - 1, array);
53 Quicksort rightTask = new Quicksort(i + 1, end, array);
54
55 leftTask.fork();
56 rightTask.fork();
57 leftTask.join();
58 rightTask.join();
59 }
60 }
61
62 public static void main(String[] args) {
63     ForkJoinPool pool = new ForkJoinPool();
64     int size = args.length;
65     int[] array = new int[size];
66
67     for (int i = 0; i < size; i++) {
68         array[i] = Integer.parseInt(args[i]);
69     }
70
71     Quicksort task = new Quicksort(0, size - 1, array);
72     pool.invoke(task);
73
74     for (int i = 0; i < size; i++) {
75         System.out.print(array[i] + " ");
76     }
77     System.out.println("");
78
79 }
80 }

```

测试结果：

```
polaris@polaris-VirtualBox: ~/course/Operating-Systems/Project/Project3/Fort-Join-Sorting-Application/Quicksort
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
polaris@polaris-VirtualBox:~/course/Operating-Systems/Project/Project3/Fort-Join-Sorting-Application/Quicksort$ java Quicksort.java 2 6 3 1 5 9 10 8 7 4
1 2 3 4 5 6 7 8 9 10
polaris@polaris-VirtualBox:~/course/Operating-Systems/Project/Project3/Fort-Join-Sorting-Application/Quicksort$
```

归并排序

与快速排序类似，只是换用了归并排序：

```
1  import java.util.concurrent.*;
2
3  public class Mergesort extends RecursiveAction {
4
5      static final int THRESHOLD = 5;
6
7      private int begin;
8      private int end;
9      private int[] array;
10
11     public Mergesort(int begin, int end, int[] array) {
12         this.begin = begin;
13         this.end = end;
14         this.array = array;
15     }
16
17     public static void Merge(int[] array, int left, int midd, int right) {
18
19         int a = left, b = midd;
20         int i = 0;
21         int[] tmp = new int[right - left + 1];
22         while (a < midd && b <= right) {
23             if (array[a] <= array[b]) {
24                 tmp[i] = array[a];
25                 i++;
26                 a++;
27             } else {
28                 tmp[i] = array[b];
29                 i++;
30                 b++;
31             }
```

```

32     }
33     while (a < midd) {
34         tmp[i] = array[a];
35         i++;
36         a++;
37     }
38     while (b <= right) {
39         tmp[i] = array[b];
40         i++;
41         b++;
42     }
43     for (int k = 0; k < i; ++k) {
44         array[left + k] = tmp[k];
45     }
46 }
47
48 protected void compute() {
49     if (end - begin < THRESHOLD) {
50         for (int i = begin; i < end; ++i) {
51             for (int j = begin; j < end; ++j) {
52                 int tmp;
53                 if (array[j] > array[j + 1]) {
54                     tmp = array[j + 1];
55                     array[j + 1] = array[j];
56                     array[j] = tmp;
57                 }
58             }
59         }
60     } else {
61         int mid = (begin + end) / 2;
62         Mergesort left = new Mergesort(begin, mid, array);
63         Mergesort right = new Mergesort(mid + 1, end, array);
64         left.fork();
65         right.fork();
66         left.join();
67         right.join();
68         Merge(array, begin, mid + 1, end);
69     }
70 }
71
72 public static void main(String[] args) {
73     ForkJoinPool pool = new ForkJoinPool();
74     int size = args.length;

```



```

75     int[] array = new int[size];
76
77     for (int i = 0; i < size; i++) {
78         array[i] = Integer.parseInt(args[i]);
79     }
80
81     Mergesort task = new Mergesort(0, size - 1, array);
82     pool.invoke(task);
83
84     for (int i = 0; i < size; i++) {
85         System.out.print(array[i] + " ");
86     }
87     System.out.println("");
88
89 }
90 }

```

测试结果：

```

polaris@polaris-VirtualBox: ~/course/Operating-Systems/Project/Project3/For-Join-Sorting-Application/Mergesort
polaris@polaris-VirtualBox:~/course/Operating-Systems/Project/Project3/For-Join-Sorting-Application/Mergesort$ java Mergesort.java 3 5 1 2 6 9 8 7 10 4
1 2 3 4 5 6 7 8 9 10
polaris@polaris-VirtualBox:~/course/Operating-Systems/Project/Project3/For-Join-Sorting-Application/Mergesort$

```

4 实验总结

1. Multithreaded Sorting Application使用了pthread，所以编译的时候要加上 `lpthread` 参数
2. 这是我第一次接触java代码。由于java和c有很多类似之处，而且实验给出了类似的源代码，只要在上面进行修改即可，因此并没有碰到太多的困难

5 实验参考资料

- 实验参考书籍：Operating System Concept， 10th edition
- 实验源代码网址：<https://github.com/greggagne/osc10e>

