

**Lab report**

|  |  |
| --- | --- |
| **Course**: | Operating System Principle |
| **Semester**: | 2nd semester of the academic year **2023-2024** |
| **Major**: | Software Engineering |
| **Class**: | 2022 |
| **Student Name**: | 吴孜远 |
| **Student ID:** | 222022321062009 |
| **Teacher:** | ZHAO, Hengjun (赵恒军) |

**School of Computer and Information Science**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | | Pthread Library and Concurrent Programming | | | |
| Date | | April，2024 | Type | | √ Confirmatory  √ Design  √ Comprehensive |
| 1. **Objective & Requirements**    1. Grasp the Pthreads API for thread creation, termination operation    2. Grasp random number generation in C    3. Grasp concurrent programming skills using the Pthread Library, in particular, can use multi-threaded programing for array sorting.    4. Grasp the experimental and theoretical analysis of the time complexity of concurrent programs | | | | | |
| 1. **Experimental environment (**platform and software**)**   Virtualbox + Ubuntu (or other platform+linux system combinations) | | | | | |
| 1. **Experimental content and design** (Main Content, Procedure, Codes and Results) 2. Task 1    * 1. Define an integer array of length SIZE, which you determine yourself      2. Randomly initialize the integer array      3. Sort the initialized integer array using, say, bubble sort, selection sort, insertion sort or quicksort. Run the sorting algorithm for a certain number of times and measure the average time cost (hint: use the time command) 3. Task 2   Write a multithreaded sorting program that works as follows:   * + 1. Set the number of CPUs of your virtual machine to at least 2 in VirtualBox     2. Define two GLOBAL integer arrays **a** and **b**, both of length SIZE (the same number as in Task 1)     3. Randomly initialize the array **a**     4. In the main thread, create two new threads to sort the first half and the second half of array **a** respectively     5. The main thread waits for the two new threads to terminate, and then ***merge*** the sorted first and second half of array **a** into array **b**     6. Run the multi-threaded sorting algorithm for a certain number of times and measure the average time cost using, say, the time command     7. Compare the time cost of your multithreaded program with the time you obtained in Task 1, and compute the speedup as follows:  1. Task 3   Repeat Task 1 and Task 2 for different values of SIZE and draw the curves showing the relations between array length and sorting time costs, for both single and multiple threaded programs.   1. Please provide your procedure and source codes to perform the tasks.   Task1  编写程序测试编译链接运行成功，测试SIZE为20    运行时间：  SIZE = 10000    SIZE = 20000    SIZE = 30000    SIZE = 40000    SIZE = 50000    代码：   1. #include<stdio.h> 2. #include<stdlib.h> 3. #include<time.h> 4. #define SIZE 30000 5. void sort(int \*a, int length){ 6. for(int i = 0; i<length; i++){ 7. for(int j = 0; j<length-i-1;j++){ 8. if(a[j]>a[j+1]){ 9. int temp = a[j+1]; 10. a[j+1] = a[j]; 11. a[j] = temp; 12. } 13. } 14. } 15. } 16. void printList(int \*a, int len){ 17. for(int i = 0; i<len; i++){ 18. printf("%d ",a[i]); 19. } 20. printf("\n"); 21. } 22. int main(){ 23. srand((unsigned)time(0)); 24. int a[SIZE]; 25. for(int i = 0; i<SIZE; i++){ 26. a[i] = rand()%100+1; 27. } 28. *//printList(a, SIZE);* 29. sort(a, SIZE); 30. *//printList(a, SIZE);* 31. return 0; 32. }   Task2  首先编写多线程排序程序，这里为了方便管理，我将程序模块化编写，通过make tool编译链接。  多线程排序经测试编译运行成功，测试SIZE为20（代码附在最后）    运行时间：  SIZE = 10000    SIZE = 20000    SIZE = 30000    SIZE = 40000    SIZE = 50000    代码：  Main.c   1. #include <stdio.h> 2. #include <stdlib.h> 3. #include <pthread.h> 4. #include <time.h> 5. #include"sort.h" 6. #include"merge.h" 7. #define SIZE 50000 8. int a[SIZE]; 9. int b[SIZE]; 10. *// 定义结构体来保存线程参数* 11. typedef struct { 12. int \*array;      *// 指向需要排序的数组的指针* 13. int length;      *// 数组的长度* 14. } SortParams; 15. typedef struct { 16. int \*a1; 17. int \*a2; 18. int lena1; 19. int lena2; 20. } MergeParams; 21. void printList(int \*a, int len){ 22. for(int i = 0; i<len; i++){ 23. printf("%d ",a[i]); 24. } 25. printf("\n"); 26. } 27. int main(){ 28. srand((unsigned)time(0)); 29. for(int i = 0; i<SIZE; i++){ 30. a[i] = rand()%100+1; 31. } 32. *//printList(a,SIZE);* 33. int \*a1 = (int \*)malloc(SIZE / 2 \* sizeof(int)); 34. int \*a2 = (int \*)malloc((SIZE + 1) / 2 \* sizeof(int)); 36. for(int i = 0; i<SIZE/2; i++){ 37. a1[i] = a[i]; 38. } 39. for(int i = 0; i<(SIZE - SIZE / 2); i++){ 40. a2[i] = a[i+SIZE/2]; 41. } 43. pthread\_t tid1, tid2, tid3; */\* the thread identifier \*/* 45. pthread\_attr\_t attr; */\* set of thread attributes \*/* 47. */\*set the default attributes \*/* 48. pthread\_attr\_init(&attr); 50. SortParams sortParams1; 51. sortParams1.array = a1; 52. sortParams1.length = SIZE/2; 53. SortParams sortParams2; 54. sortParams2.array = a2; 55. sortParams2.length = (SIZE + 1) / 2 ; 56. pthread\_create(&tid1, &attr, sort, (void \*)&sortParams1); 57. pthread\_create(&tid2, &attr, sort, (void \*)&sortParams2); 58. pthread\_join(tid1, NULL); 59. pthread\_join(tid2, NULL); 60. *//printList(a1, SIZE/2);* 61. *//printList(a2, (SIZE + 0) / 2 );* 62. MergeParams mergeParams; 63. mergeParams.a1 = a1; 64. mergeParams.a2 = a2; 65. mergeParams.lena1 = SIZE/2; 66. mergeParams.lena2 = (SIZE + 1) / 2 ; 67. pthread\_create(&tid3, &attr, merge, (void \*)&mergeParams); 68. pthread\_join(tid3, NULL); 69. *//printList(b, SIZE);* 70. return 0;   sort.c   1. #include<stdio.h> 2. #include<stdlib.h> 3. #include <pthread.h> 4. *// 定义结构体来保存线程参数* 5. typedef struct { 6. int \*array;      *// 指向需要排序的数组的指针* 7. int length;      *// 数组的长度* 8. } SortParams; 9. void \*sort(void \*args){ 10. SortParams \*params = (SortParams \*)args; 11. int \*a = params->array; 12. int length = params->length; 14. for(int i = 0; i<length; i++){ 15. for(int j = 0; j<length-i-1;j++){ 16. if(a[j]>a[j+1]){ 17. int temp = a[j+1]; 18. a[j+1] = a[j]; 19. a[j] = temp; 20. } 21. } 22. } 23. pthread\_exit(NULL); 24. }   Merge.c   1. #include<stdio.h> 2. #include<stdlib.h> 3. #include <pthread.h> 4. extern int b[]; 5. typedef struct { 6. int \*a1; 7. int \*a2; 8. int lena1; 9. int lena2; 10. } MergeParams; 11. void\* merge(void \*args){ 12. MergeParams \*params = (MergeParams \*)args; 13. int lena1 = params->lena1; 14. int lena2 = params->lena2; 15. int \*a1 = params->a1; 16. int \*a2 = params->a2; 17. int length = lena1+ lena2; 18. int x = 0, y = 0; 19. for(int i=0; i<length; i++){ 20. if(a1[x] < a2[y] && x<lena1){ 21. b[i] = a1[x]; 22. x++; 23. }else if(y<lena2){ 24. b[i] = a2[y]; 25. y++; 26. } 27. } 28. pthread\_exit(NULL); 29. }   Task3  加速：  SIZE = [10000, 20000, 30000, 40000, 50000]  Sin = [0.184, 0.842, 1.902, 3.495, 5.571]  Mul = [0.043, 0.129, 0.359, 0.702, 1.161]    根据公式：  speedup = [4.28, 6.58, 5.30, 4.98, 4.80]  平均加速为5.188， | | | | | |
| 1. **Result analysis and discussion**   In this part, you are required to provide your analysis of experimental results and summing up the harvest and the existing problems; besides, you are required to provide your thinkings about the questions:   * Suppose your multithreaded sorting program consists of two threads (not including the main thread) which runs concurrently on two cpu cores. Then what is the theorectical maximum speedup you can obtain? And what is the maximum speedup you obtained in your experiments?   排序和合并很简单，问题在于如何将它们封装成为线程函数，如何传入参数（使用结构体），如何接收返回值，如何创建和执行线程。这之间涉及到复杂的指针和引用传递的关系，和数据类型转换，要理清楚。  单线程排序的操作次数为n(n+1)/2，两线程排序为n(n/2+1)/4，差不多能提升4倍的效率。经过我的测试发现，竟然提高了平均5倍的效率。比理论上提升还要多。 | | | | | |
| Comments & Evaluation | Content & Design (A-E) | | |  | |
| Procedure & Codes (A-E) | | |  | |
| Results (A-E) | | |  | |
| Analysis & Discussion (A-E) | | |  | |
| Score (A-E):  Feedback comments: | | | | |