

**Lab report**

|  |  |
| --- | --- |
| **Course**: | Operating System Principle |
| **Semester**: | 2nd semester of the academic year **2020-2021** |
| **Major**: | Software Engineering |
| **Class**: | 2019 |
| **Student Name**: | 吴嘉诚 |
| **Student ID:** | 222019321062111 |
| **Teacher:** | ZHAO, Hengjun (赵恒军) |

**School of Computer and Information Science**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | | Pthread Library and Concurrent Programming | | | |
| Date | | April，2021 | Type | | √ Confirmatory  √ Design  √ Comprehensive |
| 1. **Objective & Requirements**    1. Grasp the Pthreads API for thread creation, termination operations    2. Grasp concurrent programming skills | | | | | |
| 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. Task1    * 1. Create a new thread in the main thread      2. Pass to integers to the new thread and calculate the sum of the two integers by the new thread 3. Task2    * 1. Define an integer array of length 200000      2. Randomly initialize the integer array      3. Sort the initialized integer array and measure the time cost (hint: use the time command) 4. Task3    * 1. Write a C program to merge two sorted integer arrays in to a single sorted integer array 5. Task4   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 200000     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. Compare the time cost of your multithreaded program with the time you obtained in Task 2, and compute the speedup. (hint: **time** command)      1. Please provide your procedure and source codes to perform the tasks.   **Task1:**   1. **代码（用数组传递参数的方法已被注释）：**   #include <pthread.h>  #include <stdio.h>  void \*runner(void \*param);  struct pair  {  int x;  int y;  };  int main(int argc, char const \*argv[])  {  //int arg[2] = {1357,2468};  int x = 1357;  int y = 2468;  struct pair arg;  arg.x = x;  arg.y = y;  pthread\_t tid;  pthread\_attr\_t attr;  pthread\_attr\_init(&attr);  pthread\_create(&tid,&attr,runner,&arg);  printf("This is in parent thread\n");  pthread\_join(tid,NULL);  return 0;  }  void \*runner(void \*param)  {  //printf("The sum of two args is: %d\n", \*(int \*)param+\*((int \*)param+1));  printf("The sum of two args is: %d\n", ((struct pair \*)param)->x+((struct pair \*)param)->y);  pthread\_exit(NULL);  }   1. **分析**  * 对于传递两个参数，我的第一反应是使用数组。将两个值存入数组，并将指针作为变量传递。我完成了代码并测试成功。 * 由于ppt提示使用结构体，我定义了一个pair结构体，并将该pair变量的地址传递。实施此方法时遇到了许多语法问题，通过搜索后成功解决问题。  1. **测试**   2021-04-28 23-14-43 的屏幕截图  **Task2:**   1. **代码**   **quick\_sort\_rec.c:**  void swap(int \*x,int \*y)  {  int t = \*x;  \*x = \*y;  \*y = t;  }  void quick\_sort\_rec(int \*nums,int start,int end)  {  if (start >= end)  return;  int mid = nums[end];  int left = start, right = end - 1;  while(left<right)  {  while(nums[left]<mid && left<right)  left++;  while(nums[right]>=mid && left<right)  right--;  swap(nums+left,nums+right);  }  if (nums[left] >= nums[end])  swap(nums+left,nums+end);  else  left++;  if (left)  quick\_sort\_rec(nums,start,left-1);  quick\_sort\_rec(nums,left+1,end);  }  **t2.c:**  #include <time.h>  #include <stdlib.h>  #include <stdio.h>  #include "quick\_sort\_rec.c"  #define SIZE 80000000  #define MAX 200000  int elem[SIZE];  int comp(const void \*a,const void \*b){return \*(int\*)a-\*(int\*)b;}  void initArray()  {  srand((unsigned)time(NULL));  /\*for (int i = 0; i < SIZE; ++i)  elem[i] = rand() % MAX; \*/  }  int main(int argc, char const \*argv[])  {  initArray();  quick\_sort\_rec(elem,0,SIZE-1);  return 0;  }   1. **分析**  * 我选用的是快速排序，在此电脑上合适的数组大小为80000000。 * 为了方便，我直接将接口和实现写在同一个文件里。  1. **测试**   2021-04-29 18-12-53 的屏幕截图   1. **冒泡排序：**   void bubbling\_sort(int \*ary, int start, int end)  {  int temp;  for (int i = 0; i < end-start; ++i){  for (int j = 0; j < end-start-i; ++j){  if(ary[j]>ary[j+1]){  temp = ary[j];  ary[j] = ary[j+1];  ary[j+1] = temp;  }  }  }  }   1. **测试**（size = 20000）   2021-04-29 17-04-12 的屏幕截图  **Task3:**   1. **代码**   #include <stdio.h>  #include <stdlib.h>  int \*merge\_ordered(int \*arry1, int \*arry2, int size1, int size2)  {  int \*p1 = arry1;  int \*p2 = arry2;  int \*merged = (int \*)malloc(sizeof(int) \* (size1+size2));  int i = 0,j = 0, m = 0;  for (; i < size1 && j < size2;++m)//merge  {  if(p1[i] < p2[j])  {  merged[m] = p1[i];  ++i;  }  else  {  merged[m] = p2[j];  ++j;  }  }  if (size1 - i == 0)  for (; j < size2; ++j, ++m)  merged[m] = p2[j];  else  for(; i < size1; ++i, ++m)  merged[m] = p1[i];  return merged;  }  int main(int argc, char const \*argv[])  {  int a[] = {1,2,4,5,6,7,7,7,7,7,9,19};//12  int b[] = {5,10,26,33,44,55,67,88,99,99};//10  int \*merged = merge\_ordered(a,b,12,10);  for (int i = 0; i < 22; ++i)  printf("%d ", merged[i]);  return 0;  }   1. **分析**  * 跟老师上课讲的方法一样，依次比较并存到另一数组中  1. **测试**   **2021-04-29 00-54-25 的屏幕截图**  **Task4:**   1. **代码**   #include <pthread.h>  #include <stdio.h>  #include <time.h>  #include <stdlib.h>  #include "quick\_sort\_rec.c"  #include "bubbling\_sort.c"  #include "merge\_ordered.c"  #define SIZE 40000000  //任务2数组大小的一半，两个数组合并后即为80000000  #define MAX 200000  struct ary\_info{  int \*ary;  int size;  };  int ary1[SIZE];  int ary2[SIZE];  void \*runner(void \*param);  void initArray(int \*ary);  int main()  {  initArray(ary1);  initArray(ary2);  struct ary\_info arg1, arg2;  arg1.ary = ary1;  arg1.size = SIZE;  arg2.ary = ary2;  arg2.size = SIZE;  pthread\_t tid1, tid2;  pthread\_attr\_t attr1;  pthread\_attr\_t attr2;  pthread\_attr\_init(&attr1);  pthread\_attr\_init(&attr2);  pthread\_create(&tid1,&attr1,runner,&arg1);  pthread\_create(&tid2,&attr2,runner,&arg2);  pthread\_join(tid1,NULL);  pthread\_join(tid2,NULL);  int \*merge = merge\_ordered(ary1,ary2,SIZE,SIZE);  /\*for (int i = 0; i < SIZE; ++i)  printf("%d ", merge[i]); \*/  return 0;  }  void \*runner(void \*para)  {  struct ary\_info \*p = (struct ary\_info \*)para;  quick\_sort\_rec(p->ary,0,p->size-1);  //bubbling\_sort(p->ary,0,p->size-1);  pthread\_exit(NULL);  }  void initArray(int \*ary)  {  srand((unsigned)time(NULL));  for (int i = 0; i < SIZE; ++i)  ary[i] = rand() % MAX;  }   1. **分析**  * 主进程创建两个线程并等待，在两个子线程中调用排序函数，完成后返回主进程，主线程调用merge\_ordered()函数将两个数组合并为一个。 * 快速排序和冒泡排序都测试成功。  1. **测试，并通过ps命令查看到的确创建了两个线程**   **2021-04-29 18-06-42 的屏幕截图**  **2021-04-29 18-06-39 的屏幕截图**   1. **计算**   **多次修改数组大小并得到数据如下：**  （size=40000000）  2021-04-29 18-21-22 的屏幕截图  （size=20000000）  2021-04-29 18-23-10 的屏幕截图  （size=2000000）  2021-04-29 18-24-19 的屏幕截图  通过计算得到   * speedup = （size=80000000）82.746/28.332 = 2.920 * speedup = （size=40000000）26.498/11.164 = 2.373 * speedup = （size=20000000）9.518/4.194 = 2.269 * speedup = （size=2000000）0.594/0.325 = 1.827   **产生这种结果的原因：**   * **双核的运算更快，所需时间减半** * **将数组分割后，复杂度下降，所以速度更快甚至大于两倍（在大数组中效果尤其明显）** * **数组很小时，speedup低于2的原因可能是我自己写的merge函数的效率不够高。** | | | | | |
| 1. **Result analysis and discussion**（Analysis of experimental results and summing up the harvest and the existing problems）   **实验收获：**   * **提高了debug能力，学会了一些gcc编译参数如-pthread，-g（用于debug），以及gdb的使用。** * **学到了更多关于ps的用法。** * **理解了多核cpu对运算的加成作用。**   **遇到的问题:**   1. **Task1** 2. **gcc编译时忘记加参数 -pthread** 3. **传过来自定义pair类型指针后，将指针类型改为自定义结构体类型时，需要加struct关键字指定。**   **2021-04-28 23-02-44 的屏幕截图**   1. **解引用时少加了括号** 2. **Task2** 3. **写排序算法时总出现段错误，在这里即数组越界。** 4. **Task3** 5. **无** 6. **Task4** 7. **发现双线程运算与单线程运算所需时间没区别，最后发现是双线程中数组大小忘记修改为单线程中的一半（合并后相同）**   **仍存在的问题：**   * **在main函数中定义的int数组大小超过2000000时，出现段错误如下：**   **2021-04-29 13-21-22 的屏幕截图** | | | | | |
| Comments & Evaluation | Content & Design (A-E) | | |  | |
| Procedure & Codes (A-E) | | |  | |
| Results (A-E) | | |  | |
| Analysis & Discussion (A-E) | | |  | |
| Score (A-E):  Feedback comments: | | | | |