**《操作系统原理》**

**实 验 报 告 书**

**2021 — 2022 学年 第 2 学期**

|  |  |  |  |
| --- | --- | --- | --- |
| **《操作系统原理》实验报告** | | | |
| **实验名称** | **多线程排序应用程序** | **实验序号** | **3** |
| **实验日期** | **2022/6** | **实验人** |  |
| 1. **实验题目**   IMG_256  **编写一个多线程排序程序，它工作如下：一个整数列表分为两个大小相等的较小子列表。两个单独线程(我们称它们为排序线程)采用你所选择的算法,对两个子列表进行排序。这两个子列表，再合并成一个已排好序的列表。** | | | |
| 1. **相关原理与知识**  * **进程的调度算法** * **多线程编程方法** * **C语言编程的基本知识**   **1.线程创建：**  **通过pthread\_create函数来创建线程**  **（1）函数原型：**  **#include <pthread.h>**  **int pthread\_create(pthread\_t \*thread, const pthread\_attr\_t \*attr, void \*(\*start\_routine) (void \*), void \*arg);**  **（2）参数解析：**  **①thread为线程id（是输出型参数，表示线程创建之后通过thread参数可以拿到线程的线程id）；**  **attr表示线程的属性，默认可设为NULL；**  **start\_routine为线程处理函数；**  **arg为线程处理函数的参数（表示对线程编号为thread的线程进行操作）。**  **②可以看到：线程处理函数的参数及返回值都是void\*，因为C语言没有泛型编程，而线程处理的类型不一定相同，所以用void\*接收任意类型，达到泛型编程的目的。**  **注意：在Linux里没有真正的线程，所以需要使用第三方的库，主要采用POSIX线程库，则使用线程相关函数必须包含头文件<pthread.h>，并且编译时必须加上-lpthread，否则就会出现连接错误。**  **（3）返回值：**  **成功返回0.失败返回错误码，而且错误代码以返回值形式返回。（所有以pthread开头的相关函数都满足这个特性）** | | | |
| **三、实验过程**  **1.用vim编辑器编写程序OSwork3.c**  将排序程序分为如下几个函数解决：  Void \*worker(void \*arg)：排序线程；  Void \*master(void \*arg)：合并线程；  Void MergeArray()：将合并后的数组重新排序；  Void print(\*array)：输出结果；  **VFY~}@R(}`8_BDOEGRDRM_I**  **K{}2HA%_)S}`[M@0[MCB00R**  本实验的源代码中还用到了pthread.h，这是一个线程的POSIX标准，定义了一套创建和操纵线程的API供编写程序使用，在源代码的主函数中有明显的体现。  **2.用命令**gcc -pthread -o **OStask3**.c**对该程序进行编译,并用命令 ./OStask3 运行该程序**  IMG_256  **该程序把原列表分成两个子列表之后，创建两个子线程分别排序，最终归并得到一个从小到大排列的列表。** | | | |
| 1. **实验结果与分析**   IMG_256  此程序实现了多线程的数组排序，可将一个数组拆分为两部分交由两个线程来排序，最后对其进行归并。 | | | |
| **五、问题总结**  直接使用gcc -o OSwork3 OSwork3.c会报错，是因为linux系统中的gcc编译器需要对头文件pthread.h头文件进行预处理。只需要将编译命令改成：gcc -pthread -o OSwork3 OSwork3.c 即可。  IMG_256  刚开始运行时出现了以上的错误结果，经过分析和检查后发现程序中归并函数这里出现了一个小错误  IMG_256  改正之后得到正确结果。 | | | |
| **六、源代码**  #include<stdio.h>  #include<pthread.h>  #include<unistd.h>  #define ARRAY\_COUNT 10//count of number that need sort  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;      }  }//bubbling      void \*master(void \*arg){      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 pid1,pid2;      printf("Before sort: ");      print(need\_sort\_array);      pthread\_create(&pid1,NULL,&worker,NULL);      pthread\_create(&pid2,NULL,&master,NULL);      pthread\_join(pid1,NULL);      pthread\_join(pid2,NULL);      printf("Sublist sorted: ");      print(need\_sort\_array);      Merge\_Array();      printf("After Merged: ");      print(sorted);  return 0;  } | | | |