

操作系统 实验报告

实验名称: 实验 3 多线程程序实验

姓名: 刘亚辉

学号: 16340157

实验 3 多线程程序实验

一、实验目的：

1. 加深对线程概念的理解。
2. 认识线程生成的过程，学会使用 `pthread_create` 生成线程，并知道如何使多线程来提高程序效率。

二、实验要求：

1. 用线程生成 Fibonacci 数列
2. 多线程矩阵乘法

三、实验过程：

1. 用线程生成 Fibonacci 数列

首先学习了解 `pthread` 编程所要学习的方法和技巧，可以利用 `pthread_create` 方法来创建一个线程，并为其绑定运行函数 `fib_generate`，然后在所有线程创建完之后，调用 `pthread_join` 方法来等待并合并线程，只保留主线程，然后输出结果即可。

具体代码结构如下：

创建一个可以保存 Fibonacci 数列的结构体：

```
1. struct Fibonacci{
2.     int fib_num;
3.     long long fib_array[100];
4. }fib;
```

创建线程如下：

```
1. pthread_t tid;
2. pthread_create(&tid, NULL, fib_generate, (void*)&fib);
```

线程运行函数如下：

```
1. void *fib_generate(void *_fib){
2.     struct Fibonacci *fib=(struct Fibonacci*)_fib;
3.     fib->fib_array[0]=0;
```

```

4.     fib->fib_array[1]=1;
5.     for(int i=2;i<=fib->fib_num;i++)
6.         fib->fib_array[i]=fib->fib_array[i-1]+fib->fib_array[i-2];
7. }

```

等待并合并线程：

```

1.     pthread_join(tid, NULL);

```

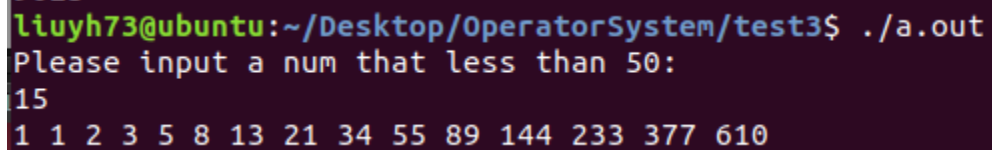
输出运算结果：

```

1.     for(int i=1;i<=fib.fib_num;i++)
2.         printf("%lld ", fib.fib_array[i]);
3.     printf("\n");

```

最终执行结果如下：



```

liuyh73@ubuntu:~/Desktop/OperatorSystem/test3$ ./a.out
Please input a num that less than 50:
15
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610

```

2. 多线程矩阵乘法

依然将代码分块：输入初始化模块，创建与合并线程模块，线程执行函数模块，输出函数模块。

数据区如下所示：

```

1. int A[10][10];
2. int B[10][10];
3. int C[10][10];
4. int M,K,N;
5. struct element{
6.     int row;
7.     int col;
8. }elem[100];
9.

```

其中element结构体保存的是每个将要计算的矩阵单元元素 C[row][col] 的行和列。

输入数据：分别输入两个矩阵 A， B

```

1. void init(){
2.     printf("Please input the M, K, N: \n");
3.     scanf("%d%d%d", &M, &K, &N);

```

```

4.     //matrix A
5.     printf("Please input the matrixA with row=M and col=K: \n");
6.     for(int i=1;i<=M;i++)
7.         for(int j=1;j<=K;j++)
8.             scanf("%d", &A[i][j]);
9.
10.    //matrix B
11.    printf("Please input the matrixB with row=K and col=N: \n");
12.    for(int i=1;i<=K;i++)
13.        for(int j=1;j<=N;j++)
14.            scanf("%d", &B[i][j]);
15. }

```

初始化参数:

```

1.     // obtain the parameter
2.     for(int i=1;i<=M;i++){
3.         for(int j=1;j<=N;j++){
4.             elem[(i-1)*N+j-1].row=i;
5.             elem[(i-1)*N+j-1].col=j;
6.         }
7.     }

```

创建线程:

```

1. // create the thread and run the function
2.     for(int i=1;i<=M;i++){
3.         for(int j=1;j<=N;j++){
4.             pthread_create(&tid[(i-1)*N+j-
1], NULL, &matrix_multiply, (void*)&elem[(i-1)*N+j-1]);
5.         }
6.     }

```

线程执行函数: 在该函数中, 计算每个矩阵 C 的单元的结果

```

1. void* matrix_multiply(void* _elem){
2.     struct element *elem=(struct element*)_elem;
3.     for(int i=1;i<=K;i++)
4.         C[elem->row][elem->col]+=A[elem->row][i]*B[i][elem->col];
5.     //pthread_exit(0);
6. }

```

合并线程：

```
1. // join here
2. for(int i=1;i<=M;i++){
3.     for(int j=1;j<=N;j++){
4.         pthread_join(tid[(i-1)*N+j-1],NULL);
5.     }
6. }
```

输出运算结果：

```
1. void print(){
2.     printf("The result matrix is as follows: \n");
3.     for(int i=1;i<=M;i++){
4.         for(int j=1;j<=N;j++){
5.             printf("%d ",C[i][j]);
6.             printf("\n");
7.         }
8.     }
```

最终输出结果如下：

```
liuyh73@ubuntu:~/Desktop/OperatorSystem/test3$ ./a.out
Please input the M, K, N:
2 3 4
Please input the matrixA with row=M and col=K:
1 2 3
2 3 4
Please input the matrixB with row=K and col=N:
1 2 3 4
2 3 4 5
3 4 5 6
14 20 26 32
20 29 38 47
```