实验 2 线程实验

1.用线程生成Fibonacci数列

- 用pthread线程库，按照第四章习题4.11的要求生成并输出Fibonacci数列

```
lgx@ubuntu:~/Desktop/lab02$ ./1
MAIN:thread start
THREAD:thread start
THREAD:thread end
MAIN:thread end
fib[0]=0
fib[1]=1
fib[2]=1
fib[3]=2
fib[4]=3
fib[5]=5
fib[6]=8
fib[7]=13
fib[8]=21
fib[9]=34
fib[10]=55
fib[11]=89
fib[12]=144
fib[13]=233
fib[14]=377
fib[15]=610
fib[16]=987
fib[17]=1597
fib[18]=2584
fib[19]=4181
```

运行结果

```c
int main() {
        pthread_t tid;
        pthread_attr_t attr;
        int fib[MAX];
        int i, n;
        printf("MAIN:thread start\n");
        pthread_attr_init(&attr);
        pthread_create(&tid,&attr,runner,fib);
        pthread_join(tid,NULL);
        printf("MAIN:thread end\n");
        for (i = 0; i < MAX; i++)
                printf("fib[%d]=%d\n", i, fib[i]);
        return 0;
}
```

主进程

```c
void *runner(void *param) {
        printf("THREAD:thread start\n");
        int i, *fib = param;
        fib[0] = 0;
        fib[1] = 1;
        for (i = 2; i < MAX; i++)
                fib[i] = fib[i-1]+fib[i-2];
        printf("THREAD:thread end\n");
}
```

子线程

2.多线程矩阵乘法

- 矩阵乘法
  - 给定两个矩阵A和B，其中A是具有M行、K列的矩阵，B为K行、N列的矩阵， A和B的矩阵积为矩阵C， C为M行、N列。矩阵C中第i行、第j列的元素Cij就是矩阵A第i行每个元素和矩阵B第j列每个元素乘积的和，即

$$C_{i,j} = \sum_{n=1}^{K} A_{i,n} \times B_{n,j}$$

要求：每个Cij的计算用一个独立的工作线程，因此它将会涉及生成M×N个工作线程。主线程(或称为父线程)将初始化矩阵A和B，并分配足够的内存给矩阵C，它将容纳矩阵A和B的积。这些矩阵将声明为全局数据，以使每个工作线程都能访问矩阵A、B和C。

```
lgx@ubuntu:~/Desktop/lab02$ ./2
>>>>>>>>>start<<<<<<<
intput m, k, n
3 2 3
intput A[3][2]:
1 4
2 5
3 6
intput B[2][3]:
8 7 6
5 4 3
>>>>>>>>>finish<<<<<<<
A[3][2]:
1 4
2 5
3 6

B[2][3]:
8 7 6
5 4 3

C[3][3]:
28 23 18
41 34 27
54 45 36
```

运行结果

```
for (i = 0; i < m; i++)
        for (j = 0; j < n; j++) {
                int *ports = (int*)malloc(sizeof(int)*2);
                ports[0] = i;
                ports[1] = j;
                pthread_attr_init(&attr);
                pthread_create(&tid[i*n+j],&attr,runner,ports);
}
for (i = 0; i < m*n; i++)
        pthread_join(tid[i],NULL);
```

主进程

```
void *runner(void *param) {
        int m, n, i, *ports = param;
        m = ports[0];
        n = ports[1];
        int sum = 0;
        for (i = 0; i < k; i++) {
                sum += mat1[m][i] * mat2[i][n];
        }
        mat3[m][n] = sum;
}
```

子线程