《操作系统》课下作业 (OS-HW2)

中国人民大学 信息学院 崔冠宇 2018202147

P219, 4.4 Consider an environment in which there is a one-to-one mapping between user-level threads and kernel-level threads that allows one or more threads within a process to issue blocking system calls while other threads continue to run. Explain why this model can make multithreaded programs run faster than their single-threaded counterparts on a uniprocessor computer.

解. (题目大意: 考虑这样一个环境, 用户级线程和内核级线程为一对一的映射关系, 它允许进程中的一个或多个线程产生会引发阻塞的系统调用, 而其它进程则继续运行. 解释为什么在单处理器机器上, 这个模型会使得多线程程序的运行速度快于相应单线程程序的运行速度.)

在多线程编程环境中, 一个内核级线程被阻塞的同时, 同进程内的其他线程可以正常运行; 而在单处理器单线程的条件下, 一旦这个线程被阻塞了, 会导致这个进程全部阻塞, 没有可以继续执行的部分, 相对而言就慢一些.

P219, 4.5 An application has 20% of code that is inherently serial. Theoretically, what will its maximum speedup be if it is run on a multicore system with four processors?

解. (题目大意: 一个应用程序有 20% 的代码是固有串行的. 如果它运行在一个有四个处理器的多核系统上, 理论上的最大加速比是多少?)

使用 **Amdahl's Law** :

$$\text{Speedup} = \frac{1}{(1-f) + \frac{f}{N}}.$$

由题意, $f = 0.80, N = 4$, 代入计算得:

$$\text{Speedup} = \frac{1}{0.20 + \frac{0.80}{4}} = 2.5.$$

所以理论上最大加速比为 2.5 倍.

P219, 4.7 Many current language specifications, such as for C and C++ , are inadequate for multithreaded programs. This can have an impact on compilers and the correctness of code, as this problem illustrates. Consider the following declarations and function definition:

```
1   int global_positives = 0;
2   typedef struct list {
3       struct list *next;
4       double val;
5   } * list;
6   void count_positives(list l)
7   {
8       list p;
9       for (p = l; p; p = p -> next)
10          if (p -> val > 0.0)
11              ++global_positives;
12  }
```

Now consider the case in which thread A performs

```
1   count_positives(<list containing only negative values>);
```

while thread B performs

```
1   ++global_positives;
```

a. What does the function do?

b. The C language only addresses single-threaded execution. Does the use of two parallel threads create any problems or potential problems?

解. (题目大意: 在很多诸如 C 和 C++ 语言的现行规范中, 并未提供对多线程的支持. 如上面的程序所示, 这一问题可能会对编译器和代码的正确性造成影响. 考虑上面的函数定义和声明:(略) 考虑如下情况: 当一个线程A执行操作(略)的同时, 另一个线程执行(略). **a.** 该函数的功能是什么? **b.** C 语言只对单个线程的执行进行了规范. 对于本题中所声明的函数, 在两个并发的线程中使用是否会有明显或潜在的问题?)

a. 这个函数的功能是计数链表中正数的个数.

b. 在这种特殊的情境下没有问题. 因为线程 A 此时并没有更新 global_positives, 也没有重新赋值(下一题的情景), 所以没有明显的问题.

P220, 4.9 Consider the following code using the POSIX Pthreads API:

thread2.c

```c
#include <pthread.h>
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
int myglobal;
void *thread_function(void *arg) {
    int i,j;
    for ( i=0; i<20; i++ ) {
        j=myglobal;
        j=j+1;
        printf(".");
        fflush(stdout);
        sleep(1);
        myglobal=j;
    }
    return NULL;
}

int main(void) {
    pthread_t mythread;
    int i;
    if ( pthread_create( &mythread, NULL, thread_function, NULL) ) {
        printf("error creating thread.");
```

```
24              abort();
25          }
26          for ( i=0; i<20; i++) {
27              myglobal=myglobal+1;
28              printf("o");
29              fflush(stdout);
30              sleep(1);
31          }
32          if ( pthread_join ( mythread, NULL ) ) {
33              printf("error joining thread.");
34              abort();
35          }
36          printf("\nmyglobal equals %d\n",myglobal);
37          exit(0);
38      }
```

In `main()` we first declare a variable called mythread, which has a type of `pthread_t`. This is essentially an ID for a thread. Next, the `if` statement creates a thread associated with `mythread`. The call `pthread_create()` returns zero on success and a nonzero value on failure. The third argument of `pthread_create()` is the name of a function that the new thread will execute when it starts. When this `thread_function()` returns, the thread terminates. Meanwhile, the main program itself defines a thread, so there are two threads executing. The `pthread_join` function enables the main thread to wait until the new thread completes.

a. What does this program accomplish?

b. Here is the output from the executed program:

```
1   $ ./thread2
2   ..o.o.o.o.oo.o.o.o.o.o.o.o.o.o..o.o.o.o.o
```

```
3   myglobal equals 21
```

Is this the output you would expect? If not, what has gone wrong?

解. (题目大意: 考虑上面使用了 POSIX Pthreads API 的代码: (略) `main()` 中首先声明一个变量 mythread, 其类型为 `pthread_t`, 它是一个线程的 ID; 然后, `if` 语句创建一个与 `mythread` 关联的进程. 调用 `pthread_create()`, 若成功则返回0, 若失败则返回非零值. `pthread_create()` 的第三个参数是一个新线程开始时将执行的函数名, 当 `thread_function()` 返回时, 线程终止. 此时主程序本身定义了一个线程, 所以有两个线程正在执行. 函数 `pthread_join` 允许主线程等待直到新的线程结束.)

a. 这段程序创建了一个新线程, 并且分别和主线程一起, 各自增加全局变量 myglobal 20次, 同时各自输出一些内容.

b. 不符合预期. 最终 myglobal 的值按预期应该为40, 但实际却为21. 注意到在新线程内部, 全局变量被赋值给局部变量 j, 进程 `sleep(1)` 后, 再将局部变量**赋值**给全局变量 myglobal, 这段时间内主进程对 myglobal 的修改无法体现, 这就是问题所在.