**1、 用户级线程和内核级线程的映射模式有哪些？各有什么特点。**

用户级线程映射模式：Java线程、Pthreads

内核级线程映射模式：Windows线程

特点：①Pthreads应用于Unix类型系统如Linux、Mac OS X、Solaris，可以提供用户级或内核级的库。

②Windows线程用于Windows操作系统的内核级线程库。

③Java线程允许线程在Java程序中直接创建和管理。但是通常使用宿主系统的线程库实现。

**2、 线程库有什么作用？请举一个例子说明利用线程库创建线程的过程。**

作用：线程库为程序员提供创建和管理线程的API。

Pthreads程序中头文件为pthread.h，pthread\_t tid声明创建线程标识符，下一条语句pthreada\_attr\_t attr声明线程属性。而后调用函数pythread\_attr\_init(&attr)设置属性。再调用函数pthread\_create()创建单独线程，需要传递的参数为线程标识符、线程属性、函数名称、命令行参数

**3、 请举例说明为什么线程技术适合多处理器架构的计算机。**

JVM在Windows XP操作系统上通过一对一模型，为每个Java线程映射到一个内核线程，多处理器架构可以充分发挥一对一模型的并发性，提高运行效率。

**4、 一个多处理器系统中某个应用程序采用多对多线程模式编写。假如该程序 的用户线程数量多于系统的处理器数量，讨论下列情况下的性能：**

**a) 该程序分配得到的内核线程的数量比处理器数量少**

性能最差

**b) 该程序分配得到的内核线程的数量和处理器相同**

性能最好

**c) 该程序分配得到的内核线程的数量大于处理器数量，但少于用户线程的 数量**

性能中等，不是最好也不是最差

**5、 有两个 512\*512 的整数矩阵，请用 Pthreads 库写一个程序，该程序利用 4 个线程来计算这两个矩阵的乘积**

#include <stdio.h>  
#include <stdlib.h>  
#include <pthread.h>  
#define MATRIX\_SIZE 512  
#define NUM\_THREADS 4  
void\* multiply\_matrices(void\* arg) {  
   int start\_row = (int)arg / MATRIX\_SIZE;  
    int start\_col = (int)arg % MATRIX\_SIZE;  
    int end\_row = start\_row + MATRIX\_SIZE/NUM\_THREADS;  
    int end\_col = start\_col + MATRIX\_SIZE/NUM\_THREADS;  
    for (int i = start\_row; i < end\_row; i++) {  
        for (int j = start\_col; j < end\_col; j++) {  
        }  
    }  
    return NULL;  
}  
int main() {  
    pthread\_t threads[NUM\_THREADS];  
    int args[NUM\_THREADS];  
    for (int i = 0; i < NUM\_THREADS; i++) {  
        args[i] = i \* MATRIX\_SIZE \* MATRIX\_SIZE;  
    }  
    for (int i = 0; i < NUM\_THREADS; i++) {  
        pthread\_create(&threads[i], NULL, multiply\_matrices, (void\*)args[i]);  
    }  
    // 创建四个子线程  
    for (int i = 0; i < NUM\_THREADS; i++) {  
        pthread\_join(threads[i], NULL);  
    }  
    return 0;  
}