**实验五：死锁处理设计报告（阐述设计程序的思路）**

**5\_1\_1.c:**

定义了最大数量的进程和资源process\_number和resource\_number，接着定义了对于每个资源已经分配出去的资源的向量，每个进程需要某个资源的最大数量的矩阵，每个进程当前需要某个资源的数量的矩阵，每个资源可用数量的向量，以及工作向量和标记每个进程是否已经完成的向量。check函数实现了检查进程i的需求是否小于等于work向量，即遍历每一个进程，对于每一种资源，是否都有此进程需求的资源数小于当前可以取用的资源数。deadlock\_detection函数实现此次分配是否完成以及是否存在安全序列：对于每一个进程，如果这个进程还没有运行完，则进行check，若成功则标记此进程可完成，并在函数末尾输出安全序列（进程可以安全运行的顺序）；若某次分配失败则表示不存在安全序列，输出ERROR。

**上机作业39题:**

根据之前实验学过的信号量的知识，来完成这道作业。首先是定义好所需要的头文件，包括Linux中信号量的#include <semaphore.h>，线程的#include <pthread.h>，sleep函数的#include <unistd.h>；接着用宏定义狒狒的数量为10，再创建三个信号量east\_mutex, west\_mutex, count\_mutex，它们的作用分别是保证同一时间向东通行，向西通行的狒狒只有一边，和互斥改变正在通行的狒狒数量。在monkey函数中，首先用dir记录下传入的参数是向哪个方向通行，接着进入改变count\_mutex的临界区，先让count加加，并判断如果是第一个狒狒，则需要先获取方向信号量，若这个狒狒向西通行，则上锁向东通行的可能，向东也同理。然后让这个狒狒通过绳索，再进入临界区让count减减，并判断如果是最后一只狒狒，则需要释放方向信号量，而如果最后一只狒狒原来是向西通行的，则释放信号量west\_mutex，允许向东通行，另一种情况同理。

在main函数中，先初始化三个信号量，注意根据题意它们的初始值都应该为1。接着就是创建狒狒的进程让狒狒通过绳索即可。

**上机作业40题:**

在主函数中，首先初始化了三个信号量，并创建了MONKEY个线程来模拟狒狒过河的过程。每个线程都有一个方向（向东或向西），并不断尝试通过绳索。在通过绳索之前，线程会先等待绳索互斥信号量，以确保当前没有其它狒狒在绳索上。然后，线程会检查是否有与当前方向相反的狒狒在绳索上，如果没有，则该线程可以通过绳索，向相应的信号量中发送一个信号，以确保其它等待在该方向的狒狒可以通过绳索。最后，线程会等待足够多的狒狒通过绳索，以便向相反方向的狒狒可以通过绳索。在离开绳索之前，线程会释放绳索互斥信号量，以便其它狒狒可以通过绳索。如果有与当前方向相反的狒狒在绳索上，则该线程会释放绳索互斥信号量并等待绳索空闲。

**上机作业41题:**

其实大体一看，书上41题跟已经在之前实现每种类型一个资源的死锁检测程序 & 实现每种类型多个资源的死锁检测程序的类型是一样的，不过书上的题目多了一步要求：将最终结果输出到一个文件中。所以我学了下如何将文件定向输出之后，就直接将第一部分的程序运行了（重新命名为41.c）。

其效果如下：

形状

中度可信度描述已自动生成

这里我将41.c的输出结果用重定位符 > 输入到41\_result.txt中，再输出所需要的参数，然后打开41\_result.txt可以看到如下内容：

图形用户界面, 文本, 应用程序

描述已自动生成

说明文件已经成功将结果保存在了文件中。

之后我又测试了一组数据，可以看到还是能够正确输出到文件并保存。

文本

描述已自动生成

文本

描述已自动生成