# 实验5 死锁和饥饿

# 实验目的

了解死锁与饥饿产生的条件

了解死锁的解决方法

掌握利用银行家算法进行死锁避免

# 实验内容

给出进程需求矩阵C、资源向量R以及一个进程的申请序列。使用进程启动拒绝和资源分配拒绝（银行家算法）模拟该进程组的执行情况。要求：

初始状态没有进程启动

计算每次进程申请是否分配？如：计算出预分配后的状态情况（安全状态、不安全状态），如果是安全状态，输出安全序列。

每次进程申请被允许后，输出资源分配矩阵A和可用资源向量V。

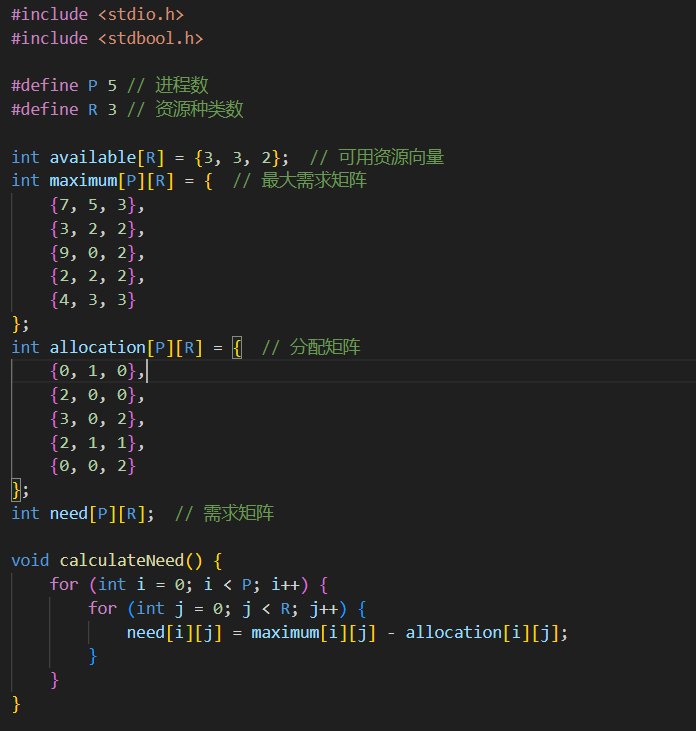
每次申请情况应可单步查看，如：输入一个空格，继续下个申请

# 实验环境

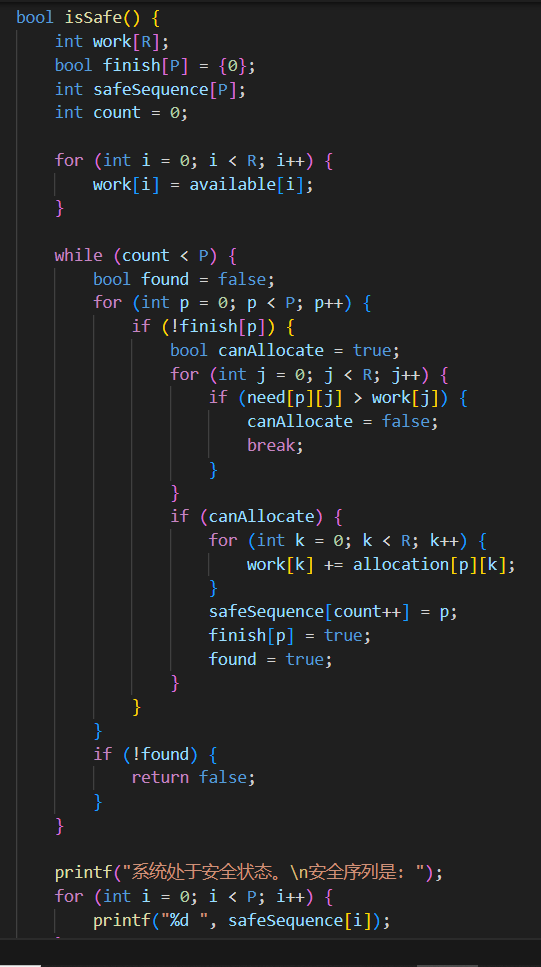
Ubuntu 22.04.4 LTS

gcc version 11.4.0

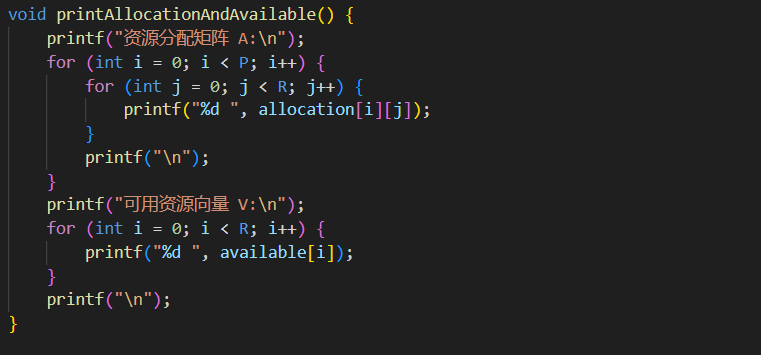
# 源代码



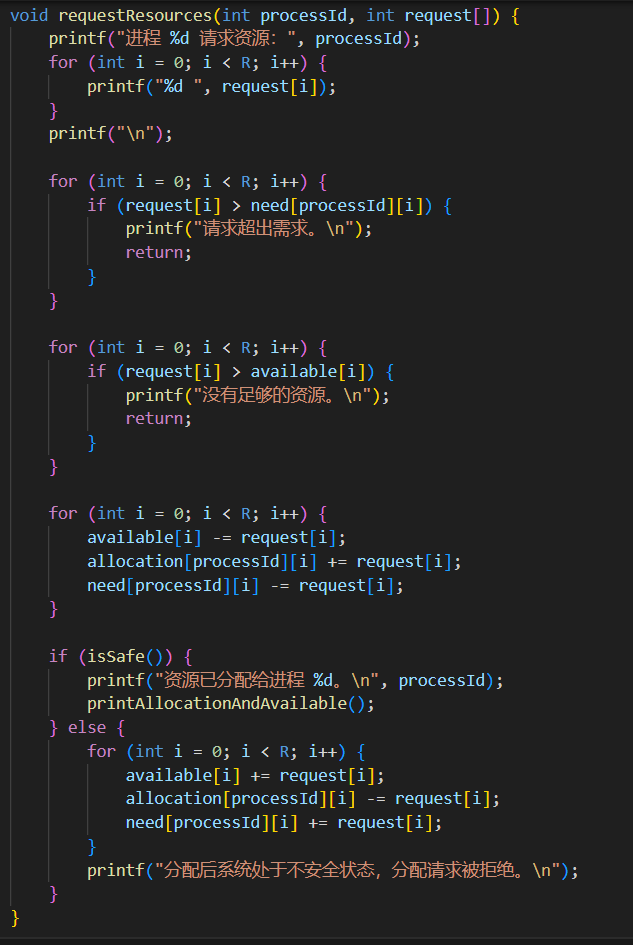
程序直接给出需求矩阵和分配矩阵，方便调试



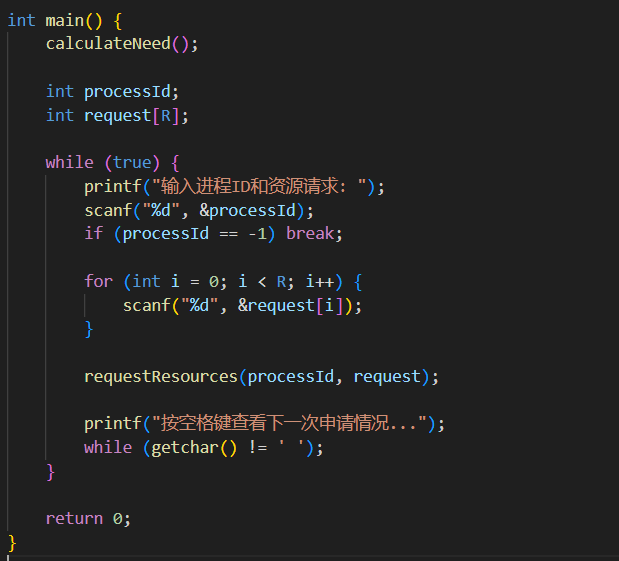
IsSafe函数，判断系统当前是否处于安全状态。通过模拟资源分配和进程完成情况，验证是否有一个安全序列，使得所有进程都能完成。若有，返回 true 并输出安全序列；否则返回 false。



辅助打印当前矩阵和可用资源

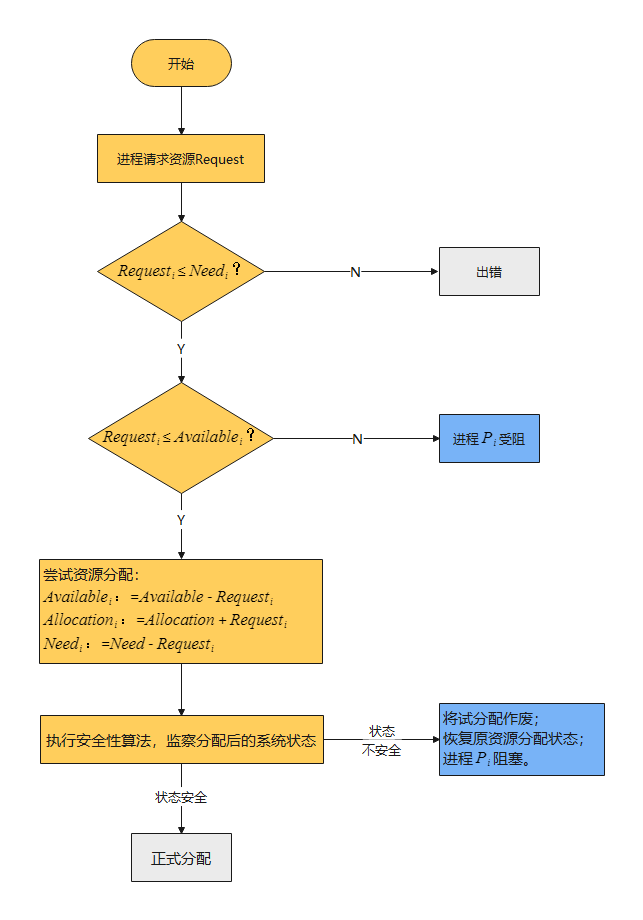


资源请求函数，发出某进程id请求对应需求向量的申请，先检查请求是否合法，然后尝试分配资源并判断系统是否处于安全状态。如果安全，则实际分配资源并输出当前的资源状态；否则撤销分配并输出错误信息。

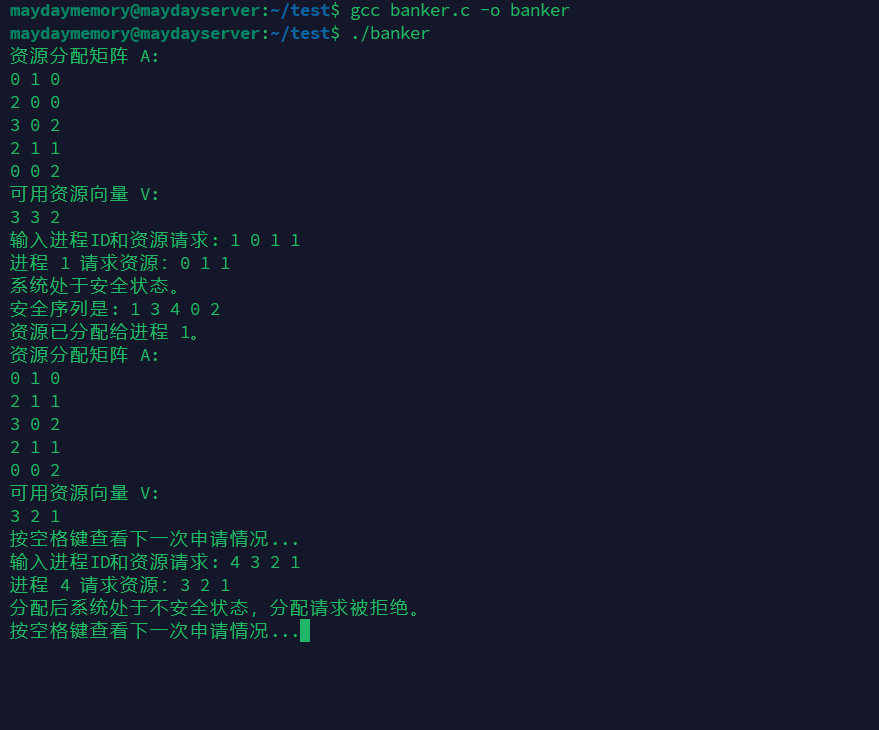


主函数，先计算初始需求矩阵，然后循环接收申请请求

# 流程图



# 实验结果



第一次进行了正常的成功分配，正常输出了安全序列，资源分配矩阵和可用资源向量；

第二次故意尝试进行可能发生死锁的分配（4的最大资源需求为4, 3, 3，用掉所有可用资源也不能满足），程序成功检测出分配后系统会处于不安全状态，并拒绝了分配

# 实验小结

在本次实验中，我们深入了解了进程并发执行和死锁产生的条件，以及死锁的解决方法。实验的主要目标是掌握利用银行家算法进行死锁避免的技术。通过对资源分配拒绝（银行家算法）和进程启动拒绝的应用，我们模拟了多个进程在资源竞争中的执行情况。实验内容包括给出进程需求矩阵、资源向量和一个进程的申请序列，并在初始状态没有进程启动的情况下，逐步计算每次进程申请是否分配。实验通过验证预分配后的状态是否安全来判断是否允许资源分配，并在每次进程申请被允许后输出资源分配矩阵和可用资源向量。通过单步查看每次申请情况，我们能够直观地观察到系统资源的变化和进程的执行情况。整个实验过程帮助我们理解了银行家算法在死锁避免中的作用，并加深了对并发执行、资源分配和系统安全性的认识。通过实践，我们不仅掌握了银行家算法的应用，还提升了处理复杂系统调度问题的能力。此实验提供了重要的理论和实践基础，有助于在未来的计算机系统设计和应用中避免和解决死锁问题。