目录

[作业一 进程调度 3](#_Toc154167125)

[作业二 存储管理调度算法 17](#_Toc154167126)

[作业三 资源分配算法 26](#_Toc154167127)

# 作业一 进程调度

一、作业目的

1. 设计进程的PCB,包括:进程名称,优先级,服务时间,剩余时间,状态等。
2. 设置进程就绪队列、完成队列
3. 进程有三种状态（R运行，W就绪，F完成）
4. 实现两种调度算法：

（1）时间片轮转

（2）抢占优先级

* + 优先级动态变化
  + 进程初始优先数=50-服务时间
  + 进程投入运行后，运行状态时，优先数随运行时间增加而减少，就绪状态时，优先数随等待时间增加而增加

二、作业软硬件要求

Windows11

Jdk11

IntelliJ IDEA 2023.2.1

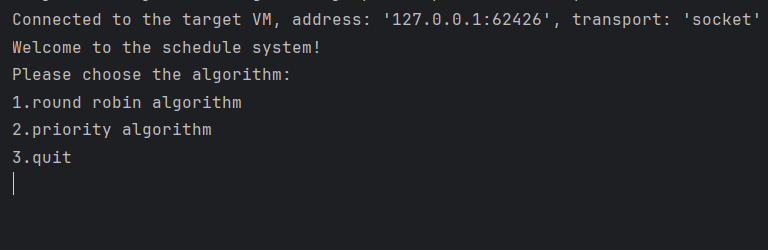
三、作业内容及结果（按要求，若需编程的需写出源程序和运行程序后的结果截图）

1，实现round robin algorithm算法

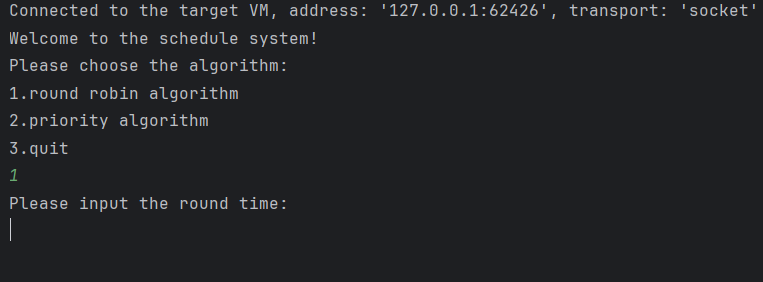
代码如下：

package org.example;  
  
import java.util.ArrayList;  
import java.util.Scanner;  
  
class ScheduleInfo {  
 String name;  
 int remainingTime;  
 int count;  
 int elapsedTime;  
 int round;  
 String state;  
 int priority;  
  
 public int getPriority() {  
 return priority;  
 }  
  
 public void setPriority(int priority) {  
 this.priority = priority;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public int getRemainingTime() {  
 return remainingTime;  
 }  
  
 public void setRemainingTime(int remainingTime) {  
 this.remainingTime = remainingTime;  
 }  
  
 public int getCount() {  
 return count;  
 }  
  
 public void setCount(int count) {  
 this.count = count;  
 }  
  
 public int getElapsedTime() {  
 return elapsedTime;  
 }  
  
 public void setElapsedTime(int elapsedTime) {  
 this.elapsedTime = elapsedTime;  
 }  
  
 public int getRound() {  
 return round;  
 }  
  
 public void setRound(int round) {  
 this.round = round;  
 }  
  
 public String getState() {  
 return state;  
 }  
  
 public void setState(String state) {  
 this.state = state;  
 }  
}  
  
public class Schedule {  
 static ArrayList<ScheduleInfo> scheduleInfo = new ArrayList<>();  
 static int round;  
 static int priority = 50;  
  
 public static void showRRData() {  
 System.out.println("Name\telapse\tremain\tcount\tround\tstate");  
 for (ScheduleInfo info : scheduleInfo) {  
 System.out.print(info.getName() + "\t");  
 System.out.print(info.getElapsedTime() + "\t");  
 System.out.print(info.getRemainingTime() + "\t");  
 System.out.print(info.getCount() + "\t");  
 System.out.print(info.getRound() + "\t");  
 System.out.print(info.getState() + "\t");  
 System.out.println();  
 }  
 printReady();  
 }  
  
 public static void showPRData() {  
 System.out.println("Name\telapse\tremain\tcount\tpriori\tstate");  
 for (ScheduleInfo info : scheduleInfo) {  
 System.out.print(info.getName() + "\t");  
 System.out.print(info.getElapsedTime() + "\t");  
 System.out.print(info.getRemainingTime() + "\t");  
 System.out.print(info.getCount() + "\t");  
 System.out.print(info.getPriority() + "\t");  
 System.out.print(info.getState() + "\t");  
 System.out.println();  
 }  
 printReady();  
 }  
  
 private static void printReady() {  
 System.out.print("Ready query：");  
 for (ScheduleInfo info : scheduleInfo) {  
 if (info.getState().equals("W")) {  
 System.out.print(info.getName() + " ");  
 }  
 }  
 System.out.println();  
 System.out.print("Finished query：");  
 for (ScheduleInfo info : scheduleInfo) {  
 if (info.getState().equals("F")) {  
 System.out.print(info.getName() + " ");  
 }  
 }  
 System.out.println();  
 }  
  
 public static void priorityAlgorithm() {  
 int finishedSum = 0;  
 while (finishedSum != scheduleInfo.size()) {  
 scheduleInfo.sort((o1, o2) -> o2.getPriority() - o1.getPriority());  
 if (!scheduleInfo.get(0).getState().equals("F")) {  
 scheduleInfo.get(0).setState("R");  
 if(scheduleInfo.get(0).getCount()==0)  
 {  
 scheduleInfo.get(0).setCount(1);  
 }  
 showPRData();  
 int remainingTime = scheduleInfo.get(0).getRemainingTime();  
 scheduleInfo.get(0).setRemainingTime(remainingTime - 1);  
 scheduleInfo.get(0).setElapsedTime(scheduleInfo.get(0).getElapsedTime() + 1);  
 scheduleInfo.get(0).setPriority(scheduleInfo.get(0).getPriority() - 1);  
 for (ScheduleInfo info : scheduleInfo) {  
 if (info.getState().equals("W")) {  
 info.setPriority(info.getPriority() + 1);  
 }  
 }  
 if (scheduleInfo.get(0).getRemainingTime() == 0) {  
 finishedSum++;  
 scheduleInfo.get(0).setState("F");  
 } else {  
 scheduleInfo.get(0).setState("W");  
 }  
 } else {  
 scheduleInfo.add(scheduleInfo.get(0));  
 scheduleInfo.remove(0);  
 }  
 }  
 showPRData();  
 }  
  
 public static void RR() {  
 int finishedSum = 0;  
 while (finishedSum != scheduleInfo.size()) {  
 if (!scheduleInfo.get(0).getState().equals("F")) {  
 scheduleInfo.get(0).setState("R");  
 showRRData();  
 int remainingTime = scheduleInfo.get(0).getRemainingTime();  
 if (remainingTime > round) {  
 scheduleInfo.get(0).setRemainingTime(remainingTime - round);  
 scheduleInfo.get(0).setElapsedTime(scheduleInfo.get(0).getElapsedTime() + round);  
 scheduleInfo.get(0).setCount(scheduleInfo.get(0).getCount() + 1);  
 scheduleInfo.get(0).setState("W");  
 scheduleInfo.add(scheduleInfo.get(0));  
 scheduleInfo.remove(0);  
 } else {  
 scheduleInfo.get(0).setCount(scheduleInfo.get(0).getCount() + 1);  
 int temp = round;  
 for (int i = 0; i < scheduleInfo.size(); i++) {  
 remainingTime = scheduleInfo.get(0).getRemainingTime();  
 if (scheduleInfo.get(0).getState().equals("F")) {  
 scheduleInfo.add(scheduleInfo.get(0));  
 scheduleInfo.remove(0);  
 } else {  
 scheduleInfo.get(0).setRemainingTime(Math.max(remainingTime - temp, 0));  
 scheduleInfo.get(0).setElapsedTime(scheduleInfo.get(0).getElapsedTime() + remainingTime - Math.max(remainingTime - temp, 0));  
 temp -= remainingTime;  
 scheduleInfo.get(0).setState("F");  
 finishedSum++;  
 scheduleInfo.add(scheduleInfo.get(0));  
 scheduleInfo.remove(0);  
 if (temp <= 0) {  
 break;  
 }  
 }  
 }  
 }  
 } else {  
 scheduleInfo.add(scheduleInfo.get(0));  
 scheduleInfo.remove(0);  
 }  
 }  
 showRRData();  
 }  
  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.in);  
 int selected;  
 while(true) {  
 System.out.println("Welcome to the schedule system!\n" + "Please choose the algorithm:\n" + "1.round robin algorithm\n" + "2.priority algorithm\n" + "3.quit");  
 selected = sc.nextInt();  
 int serviceNum;  
 if (selected==1) {  
 scheduleInfo.clear();  
 System.out.println("Please input the round time:");  
 round = sc.nextInt();  
 System.out.println("Please input the service number:");  
 serviceNum = sc.nextInt();  
 System.out.println("Please input name and service time:\n" + "For example, if the name is a0 and the service time is 3, then input a0 3");  
 for (int i = 0; i < serviceNum; i++) {  
 ScheduleInfo scheduleTemporary = new ScheduleInfo();  
 scheduleTemporary.setName(sc.next());  
 scheduleTemporary.setRemainingTime(sc.nextInt());  
 scheduleTemporary.setCount(0);  
 scheduleTemporary.setElapsedTime(0);  
 scheduleTemporary.setRound(round);  
 scheduleTemporary.setState("W");  
 scheduleInfo.add(scheduleTemporary);  
 }  
 RR();  
 }  
 else if(selected==2) {  
 scheduleInfo.clear();  
 System.out.println("Please input the service number:");  
 serviceNum = sc.nextInt();  
 System.out.println("Please input name and service time:\n" + "For example, if the name is a0 and the service time is 3, then input a0 3");  
 for (int i = 0; i < serviceNum; i++) {  
 ScheduleInfo scheduleTemporary = new ScheduleInfo();  
 scheduleTemporary.setName(sc.next());  
 scheduleTemporary.setRemainingTime(sc.nextInt());  
 scheduleTemporary.setCount(0);  
 scheduleTemporary.setElapsedTime(0);  
 scheduleTemporary.setPriority(priority - scheduleTemporary.remainingTime);  
 scheduleTemporary.setState("W");  
 scheduleInfo.add(scheduleTemporary);  
 }  
 priorityAlgorithm();  
 }  
 else if(selected==3)  
 {  
 break;  
 }  
 }  
 sc.close();  
 }  
  
}

运行结果：



此时，输入1，则提示信息如下：



假设一个时间片的时间长为2，共有5个进程，进程信息为：

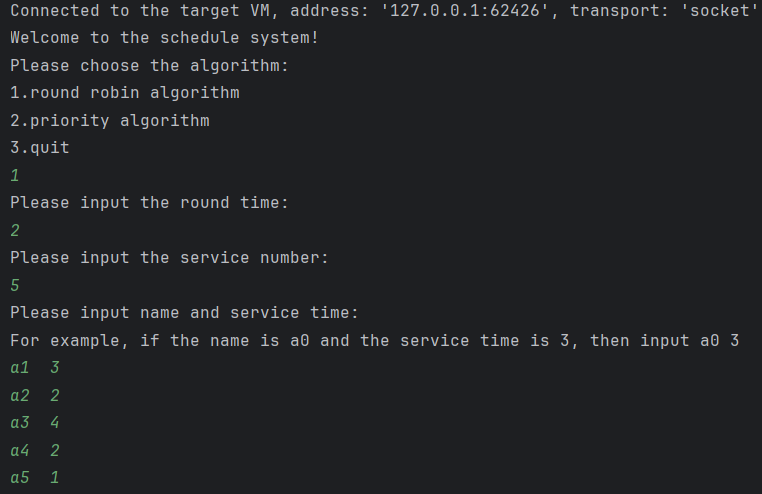
a1 3

a2 2

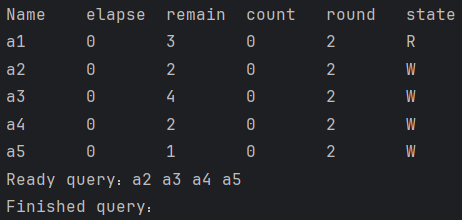
a3 4

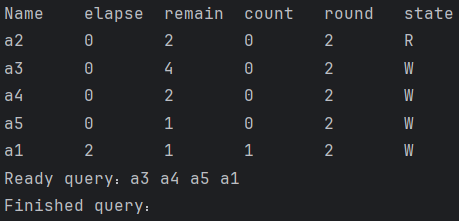
a4 2

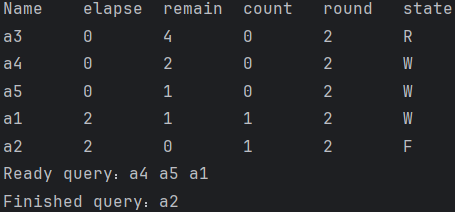
a5 1

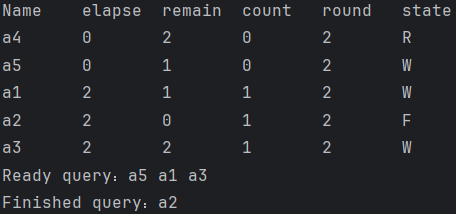


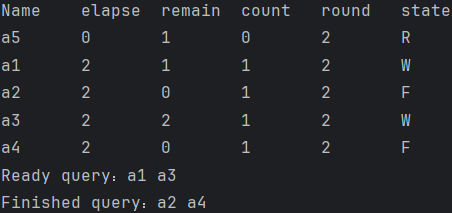
则，在该调度算法下，进程的运行顺序如下：

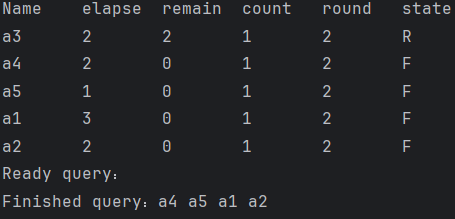


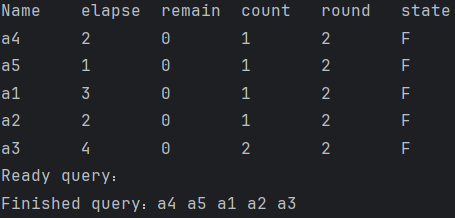
、





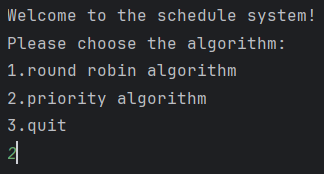






2，实现priority algorithm算法

首先在初始页面输入2



进程数仍然为5，每个进程的服务时间同上：

a1 3

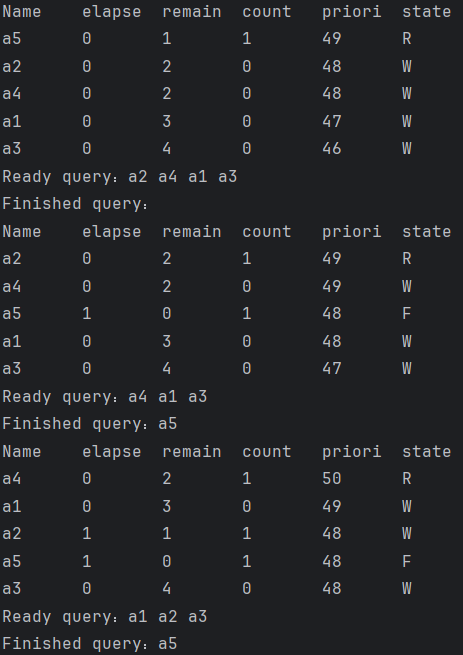
a2 2

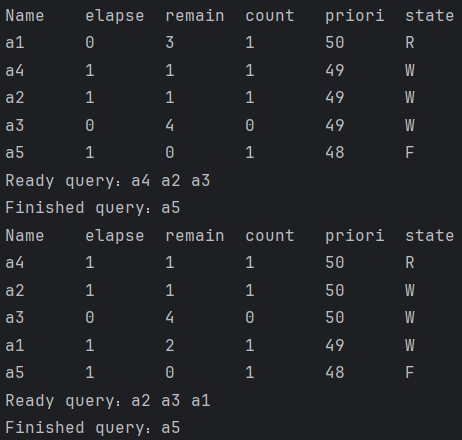
a3 4

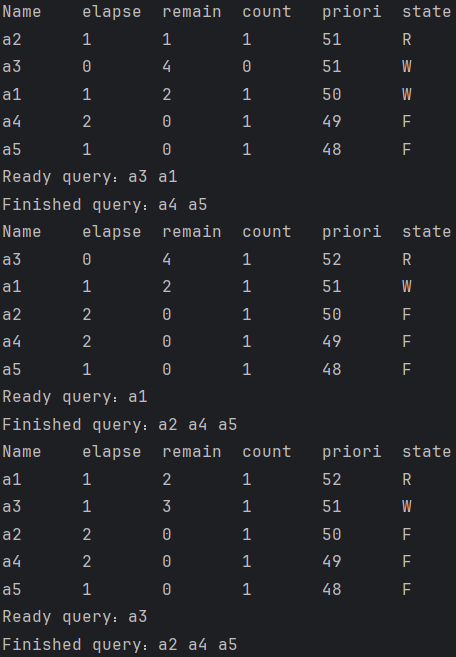
a4 2

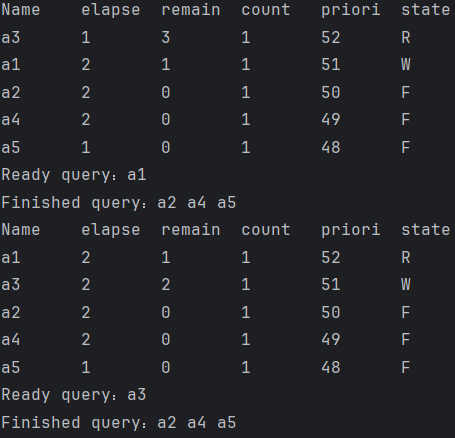
a5 1

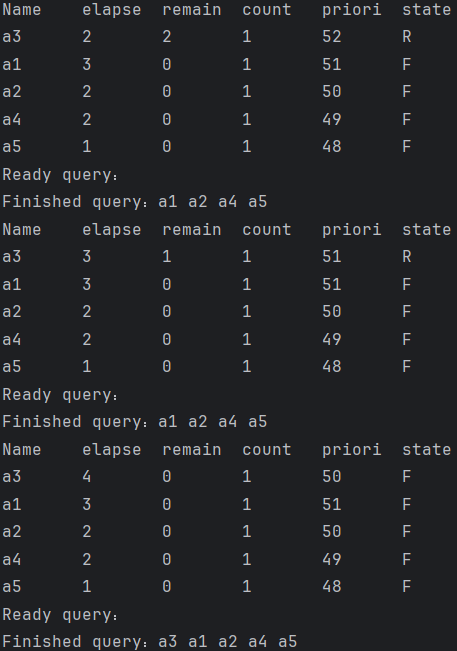
则打印信息如下：











四、自评（评价是否完成实验要求及自己的收获）

在操作系统实验课上完成时间片轮转和抢占优先级两种调度算法的编写后，我对自己的表现感到比较满意。

首先，我在编写时间片轮转算法时，充分理解了该算法的原理和实现方式。我成功地实现了按照设定的时间片大小轮流分配CPU时间给各个进程，并且能够正确地切换进程。我还进行了一些基本的优化，比如减少上下文切换的次数，确保算法的效率。

而在编写抢占优先级算法时，我能够正确地根据进程的优先级来进行调度。对于高优先级的进程，我能够及时抢占正在执行的进程，确保高优先级的任务能够得到优先执行。我也考虑了一些可能出现的情况，比如优先级相等时的处理方式，确保算法的可靠性和公平性。

在整个编写过程中，我充分地运用了所学的知识和技能，能够灵活地运用各种数据结构和算法来实现调度算法的功能。同时，我也注重代码的简洁性和可读性，使得其他人能够轻松理解我的代码。

然而，回顾自己的表现，我也发现了一些不足之处。比如在编写过程中，有时候我在处理一些边界情况时没有考虑全面，导致可能会出现一些潜在的问题。我意识到这一点后，我会更加注重对各种情况的全面考虑和测试，以确保我的算法在各种情况下都能够正确运行。

总的来说，我对自己在操作系统实验课上完成时间片轮转和抢占优先级两种调度算法的表现感到满意。通过这次实践，我不仅提高了自己的编程技能，也对操作系统调度算法有了更深入的理解。这将对我的进一步学习和发展产生积极的影响。

# 作业二 存储管理调度算法

一、作业目的

* 建立分区描述区
* 建立自由主存队列

针对不同的放置策略建立相应队列结构

* 编写分区分配算法
* 编写分区回收算法

二、作业软硬件要求

Windows11

Jdk11

IntelliJ IDEA 2023.2.1

三、作业内容及结果（按要求，若需编程的需写出源程序和运行程序后的结果截图）

代码如下：

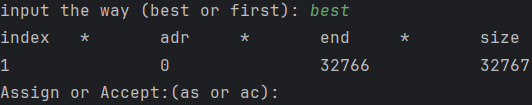
package org.example;  
  
import java.util.ArrayList;  
import java.util.Comparator;  
import java.util.Scanner;  
  
class Block  
{  
 int adress;  
 int size;  
 int end;  
  
 public int getAdress() {  
 return adress;  
 }  
  
 public void setAdress(int adress) {  
 this.adress = adress;  
 }  
  
 public int getSize() {  
 return size;  
 }  
  
 public void setSize(int size) {  
 this.size = size;  
 }  
  
 public int getEnd() {  
 return end;  
 }  
  
 public void setEnd(int end) {  
 this.end = end;  
 }  
 public Block()  
 {  
 setAdress(0);  
 setSize(32767);  
 setEnd(32766);  
 }  
}  
public class SJ\_fit {  
 static ArrayList<Block> blocks = new ArrayList<>();  
 public static void showData()  
 {  
 System.out.print("index\t\*\tadr\t\*\tend\t\*\tsize\n");  
 for(int i=0;i<blocks.size();i++)  
 {  
 System.out.print((i+1)+"\t\t"+blocks.get(i).getAdress()+"\t\t"+blocks.get(i).getEnd()+"\t\t"+blocks.get(i).getSize()+"\n");  
 }  
 }  
 public static void bestFit()  
 {  
 Scanner sc = new Scanner(System.in);  
 while(true) {  
 showData();  
 System.out.print("Assign or Accept:(as or ac): ");  
 String way = sc.next();  
 if (way.equals("ac")) {  
 accept(sc);  
 blocks.sort(Comparator.comparingInt(Block::getSize));  
 } else if (way.equals("as")) {  
 assign(sc);  
 blocks.sort(Comparator.comparingInt(Block::getSize));  
 } else {  
 System.out.println("wrong input!");  
 break;  
 }  
 }  
 }  
 public static void firstFit()  
 {  
 Scanner sc = new Scanner(System.in);  
 while(true) {  
 showData();  
 System.out.print("Assign or Accept:(as or ac): ");  
 String way = sc.next();  
 if (way.equals("ac")) {  
 accept(sc);  
 } else if (way.equals("as")) {  
 assign(sc);  
 } else {  
 System.out.println("wrong input!");  
 break;  
 }  
 }  
 }  
  
 private static void assign(Scanner sc) {  
 System.out.print("input APPLICATION:");  
 int application = sc.nextInt();  
 boolean isFind = false;  
 for (Block block : blocks) {  
 if (block.getSize() >= application) {  
 block.setSize(block.getSize() - application);  
 block.setEnd(block.getEnd() - application);  
 isFind = true;  
 System.out.println("SUCCESS!!!ADDRESS=" + (block.getEnd() + 1));  
 break;  
 }  
 }  
 if(!isFind)  
 {  
 System.out.println("Too large application!");  
 }  
 }  
  
 private static void accept(Scanner sc) {  
 System.out.print("Input adr and size: ");  
 int address = sc.nextInt()-1;  
 int size = sc.nextInt();  
 int addSize = address + size;  
 boolean isExisted = false;  
 for (Block block : blocks) {  
 if (address <= block.getEnd() && addSize > block.getEnd()) {  
 block.setSize(block.getSize() + addSize - block.getEnd());  
 block.setEnd(addSize);  
 isExisted = true;  
 break;  
 } else if ((address <= block.getAdress() && addSize > address)) {  
 block.setSize(block.getSize() + block.getAdress() - address + 1);  
 block.setAdress(address + 1);  
 isExisted = true;  
 break;  
 }  
 }  
 if(!isExisted) {  
 Block newBlock = new Block();  
 newBlock.setAdress(address + 1);  
 newBlock.setEnd(addSize);  
 newBlock.setSize(size);  
 blocks.add(newBlock);  
 }  
 }  
  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.in);  
 blocks.add(new Block());  
 System.out.print("input the way (best or first): ");  
 String way = sc.next();  
 if(way.equals("best"))  
 {  
 bestFit();  
 }  
 else if(way.equals("first"))  
 {  
 firstFit();  
 }  
 else {  
 System.out.println("wrong input!");  
 }  
 sc.close();  
 }  
}

1，最佳适应算法

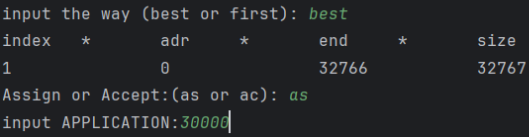
程序运行后，打印结果如下：



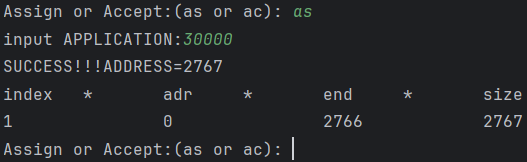
输入best



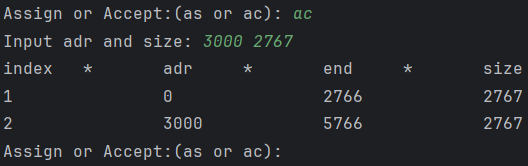
输入as和30000



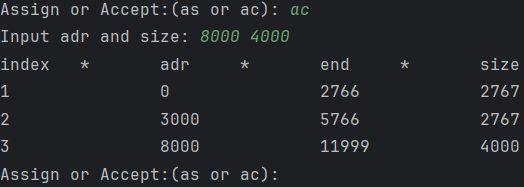
打印信息如下：



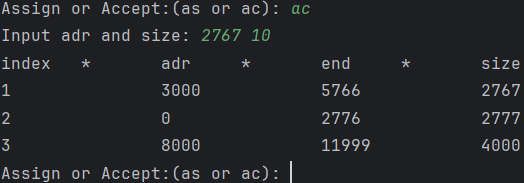
这时输入ac，输入3000 2767，打印信息如下：



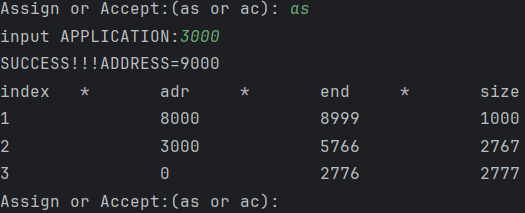
输入ac，输入8000 4000，打印信息如下：



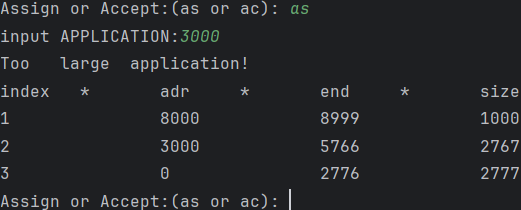
输入ac，输入2767 10，打印信息如下：



输入as，输入3000



再输入as，输入3000

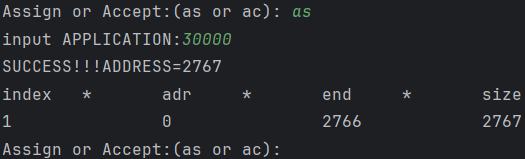


2，最先适应算法

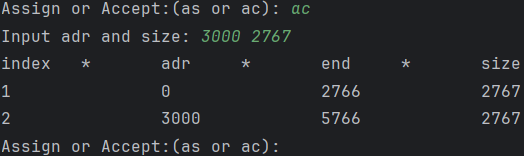
程序运行时，输入first



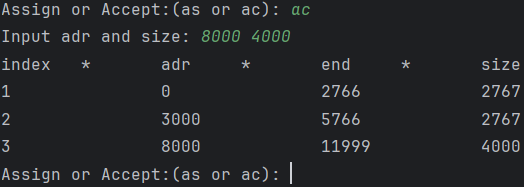
输入as，再输入30000



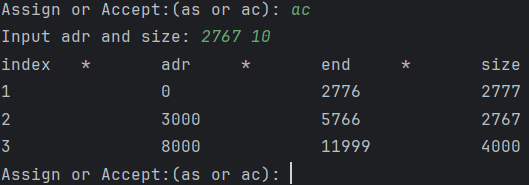
输入ac，再输入3000 2767



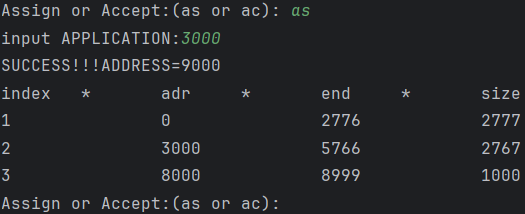
输入ac，再输入8000 4000



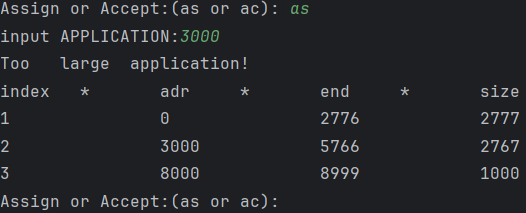
输入ac，再输入2767 10



输入as，再输入3000



再输入as，输入3000



四、自评（评价是否完成实验要求及自己的收获）

在操作系统实验课上学习并实现了存储管理调度算法的代码编写，我对自己的表现有以下评价：

首先，我对最佳适应法的理解和实现相对较为顺利。我能够理解该算法的核心思想，并将其转化为代码逻辑。我能够正确地找到最合适的内存块来分配给进程，并进行相应的内存回收操作。我在代码编写过程中尽量考虑了效率和代码可读性，遵循了良好的编程习惯。在测试中，我能够得到正确的结果，并且对算法的性能和优化进行了一定的思考。

其次，对于最先适应法的实现，我也相对满意。我能够理解该算法的基本原理，并将其转化为代码逻辑。在代码编写中，我能够正确地找到第一个合适的内存块来分配给进程，并处理相应的内存回收。我在测试过程中发现了一些边界情况和可能的问题，并进行了一些优化和改进。虽然可能还有一些改进的空间，但总体而言，我对自己的表现还是比较满意的。

然而，我也意识到在编写代码过程中可能存在一些细节的处理问题，例如内存块的分配和回收的边界情况处理可能需要进一步优化。此外，我还需要更多地深入学习该算法以及其他存储管理调度算法的原理和实现。对于算法复杂度的分析和优化也需要更深入的思考和实践。

总的来说，我通过这次实验课对最佳适应法和最先适应法有了更深入的了解，并成功实现了相关代码。我为自己的努力和成果感到满意，但也认识到自己还有进一步提升的空间。通过 这次实验，我明确了自己的不足之处，同时也为今后的学习和实践制定了一些改进计划。

# 作业三 资源分配算法

一、作业目的

* 实现银行家算法
* 实现安全性算法

二、作业软硬件要求

Windows11

Jdk11

IntelliJ IDEA 2023.2.1

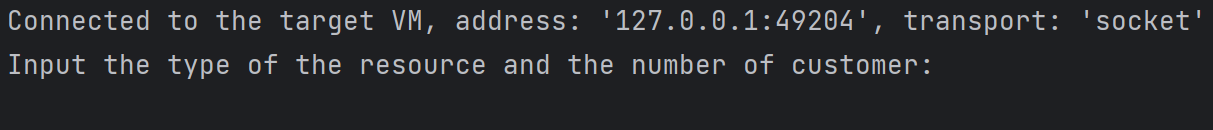
三、作业内容及结果（按要求，若需编程的需写出源程序和运行程序后的结果截图）

代码如下：

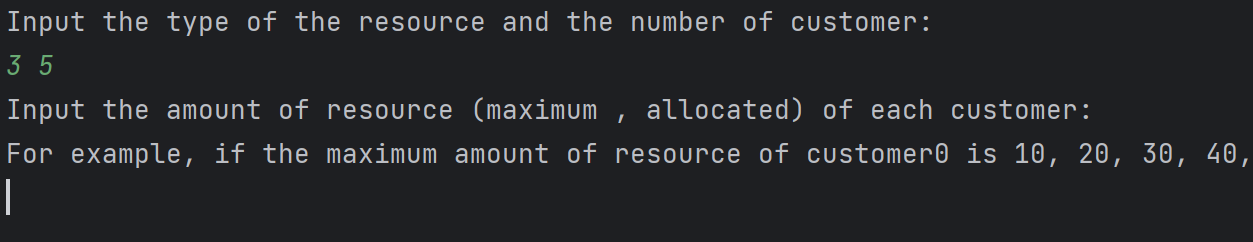
package org.example;  
  
import java.util.Scanner;  
  
public class Banker {  
 static int[][] max;  
 static int[][] allocated;  
 static int[][] need;  
 static int[][] work;  
 static int resource;  
 static int customer;  
 static int[] available;  
 static String[] customerName;  
 public static boolean judge(int[] request,int[] actual)  
 {  
 for(int i=0;i<resource;i++)  
 {  
 if(request[i]>actual[i])  
 {  
 return false;  
 }  
 }  
 return true;  
 }  
 public static void cal\_need()  
 {  
 need = new int[customer][resource];  
 for(int i=0;i<customer;i++)  
 {  
 for(int j=0;j<resource;j++)  
 {  
 need[i][j] = max[i][j] - allocated[i][j];  
 }  
 }  
 }  
 public static void cal\_work(int[] availableCopy,int[][] allocatedCopy,int[][] needCopy)  
 {  
 int[] visited = new int[customer];  
 int record = 0;  
 work = new int[customer][resource];  
 System.out.println("Name\tWork\tNeed\tAllo\tWork+A\tFinish");  
 for(int k=0;k<customer;k++) {  
 for (int i = 0; i < customer; i++) {  
 if (judge(needCopy[i], availableCopy) && visited[i] == 0) {  
 System.out.print(customerName[i] + "\t");  
 for (int j = 0; j < resource; j++) {  
 System.out.print(availableCopy[j] + " ");  
 availableCopy[j] = availableCopy[j] + allocatedCopy[i][j];  
 work[i][j] = availableCopy[j];  
 }  
 System.out.print("\t");  
 for (int j = 0; j < resource; j++) {  
 System.out.print(needCopy[i][j] + " ");  
 }  
 System.out.print("\t");  
 for (int j = 0; j < resource; j++) {  
 System.out.print(allocatedCopy[i][j] + " ");  
 }  
 System.out.print("\t");  
 for (int j = 0; j < resource; j++) {  
 System.out.print(work[i][j] + " ");  
 }  
 System.out.print("\t");  
 System.out.print("T");  
 System.out.println();  
 visited[i] = 1;  
 record++;  
 break;  
 }  
 }  
 }  
 if(record==customer)  
 {  
 System.out.println("SYSTEM SECURITY!!!");  
 }  
 else {  
 System.out.println("No Customer Available!!!");  
 System.out.println("SYSTEM IS NOT SECURITY!!!");  
 }  
 }  
 public static void systemSecurity()  
 {  
 cal\_need();  
 int[] availableCopy = new int[resource];  
 System.arraycopy(available, 0, availableCopy, 0, resource);  
 int[][] allocatedCopy = new int[customer][resource];  
 int[][] needCopy = new int[customer][resource];  
 for(int i=0;i<customer;i++)  
 {  
 System.arraycopy(allocated[i], 0, allocatedCopy[i], 0, resource);  
 System.arraycopy(need[i], 0, needCopy[i], 0, resource);  
 }  
 cal\_work(availableCopy,allocatedCopy,needCopy);  
 }  
 public static void requestSecurity()  
 {  
 Scanner requestScanner = new Scanner(System.in);  
 System.out.println("Please input the customer’s name and request:\n" +  
 "For example, if the customer’s name is customer0 and the request is 1 2 3 4, then input customer0 1 2 3 4");  
 String customerNameTemp = requestScanner.next();  
 int customerIndex = Integer.parseInt(customerNameTemp.substring(customerNameTemp.length()-1));  
 int[] request = new int[resource];  
 for(int i=0;i<resource;i++)  
 {  
 request[i] = requestScanner.nextInt();  
 }  
 cal\_need();  
 int[] availableCopy = new int[resource];  
 System.arraycopy(available, 0, availableCopy, 0, resource);  
 int[][] allocatedCopy = new int[customer][resource];  
 int[][] needCopy = new int[customer][resource];  
 for(int i=0;i<customer;i++)  
 {  
 System.arraycopy(allocated[i], 0, allocatedCopy[i], 0, resource);  
 System.arraycopy(need[i], 0, needCopy[i], 0, resource);  
 }  
 if(judge(request,availableCopy))  
 {  
 for(int i=0;i<resource;i++)  
 {  
 availableCopy[i] = availableCopy[i] - request[i];  
 allocatedCopy[customerIndex][i] = allocatedCopy[customerIndex][i] + request[i];  
 needCopy[customerIndex][i] = needCopy[customerIndex][i] - request[i];  
 }  
 cal\_work(availableCopy,allocatedCopy,needCopy);  
 }  
 else {  
 System.out.println("RESOURCE INSUFFICIENT!!!");  
 System.out.println(customerName[customerIndex]+" CAN NOT OBTAIN RESOURCES IMMEDIATELY.");  
 }  
 }  
  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
 System.out.println("Input the type of the resource and the number of customer:");  
 resource = scanner.nextInt();  
 customer = scanner.nextInt();  
 max = new int[customer][resource];  
 allocated = new int[customer][resource];  
 available = new int[resource];  
 customerName = new String[customer];  
 System.out.println("Input the amount of resource (maximum , allocated) of each customer:");  
 System.out.println("For example, if the maximum amount of resource of customer0 is 10, 20, 30, 40, and the allocated amount is 5, 10, 15, 20, then input customer0 10 20 30 40 5 10 15 20");  
 for(int j=0;j<customer;j++)  
 {  
 String customerNameTemp = scanner.next();  
 int index = Integer.parseInt(customerNameTemp.substring(customerNameTemp.length()-1));  
 customerName[index] = customerNameTemp;  
 int[] maxTem = new int[resource];  
 int[] allocatedTem = new int[resource];  
 for(int i=0;i<resource;i++)  
 {  
 maxTem[i] = scanner.nextInt();  
 }  
 for(int i=0;i<resource;i++)  
 {  
 allocatedTem[i] = scanner.nextInt();  
 }  
 max[index] = maxTem;  
 allocated[index] = allocatedTem;  
 }  
 System.out.println("Input the amount of each available resource:");  
 for(int i=0;i<resource;i++)  
 {  
 available[i] = scanner.nextInt();  
 }  
 int status;  
 label:  
 do {  
 System.out.println("1、judge the system security\n" +  
 "2、judge the request security\n" +  
 "3、quit\n");  
 System.out.println("Input your choice:");  
 status = scanner.nextInt();  
 switch (status) {  
 case 1:  
 systemSecurity();  
 break;  
 case 2:  
 requestSecurity();  
 break;  
 case 3:  
 break label;  
 }  
 }while (true);  
 scanner.close();  
 }  
}

1. 实现银行家算法

程序运行后，电脑界面运行如下：



这时，输入3 5



再按照要求，输入资源数和进程数：

P0 7 5 3 0 1 0

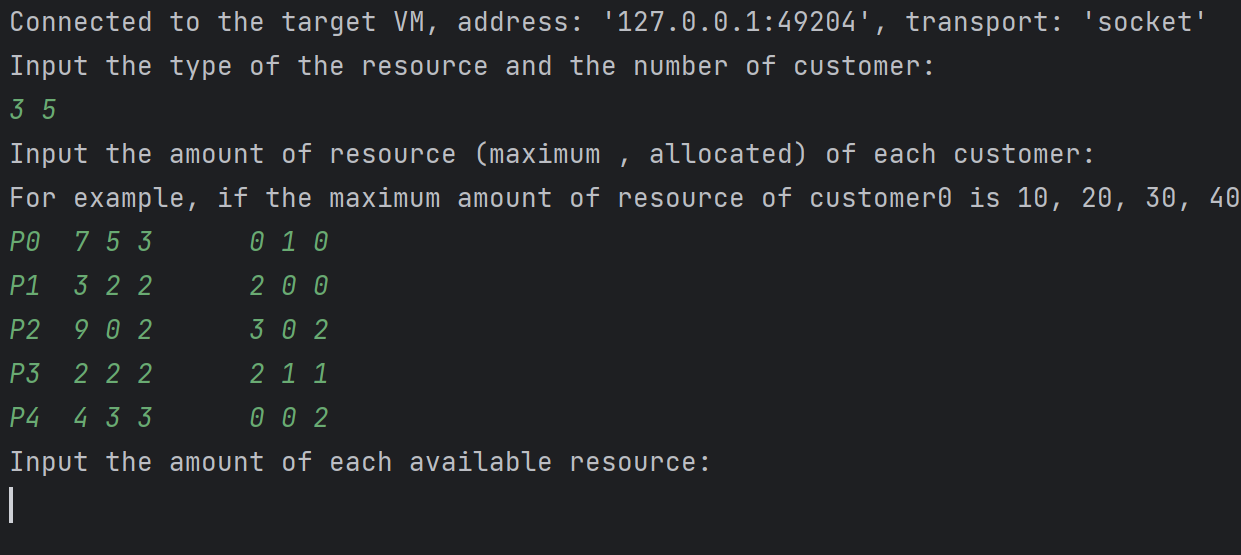
P1 3 2 2 2 0 0

P2 9 0 2 3 0 2

P3 2 2 2 2 1 1

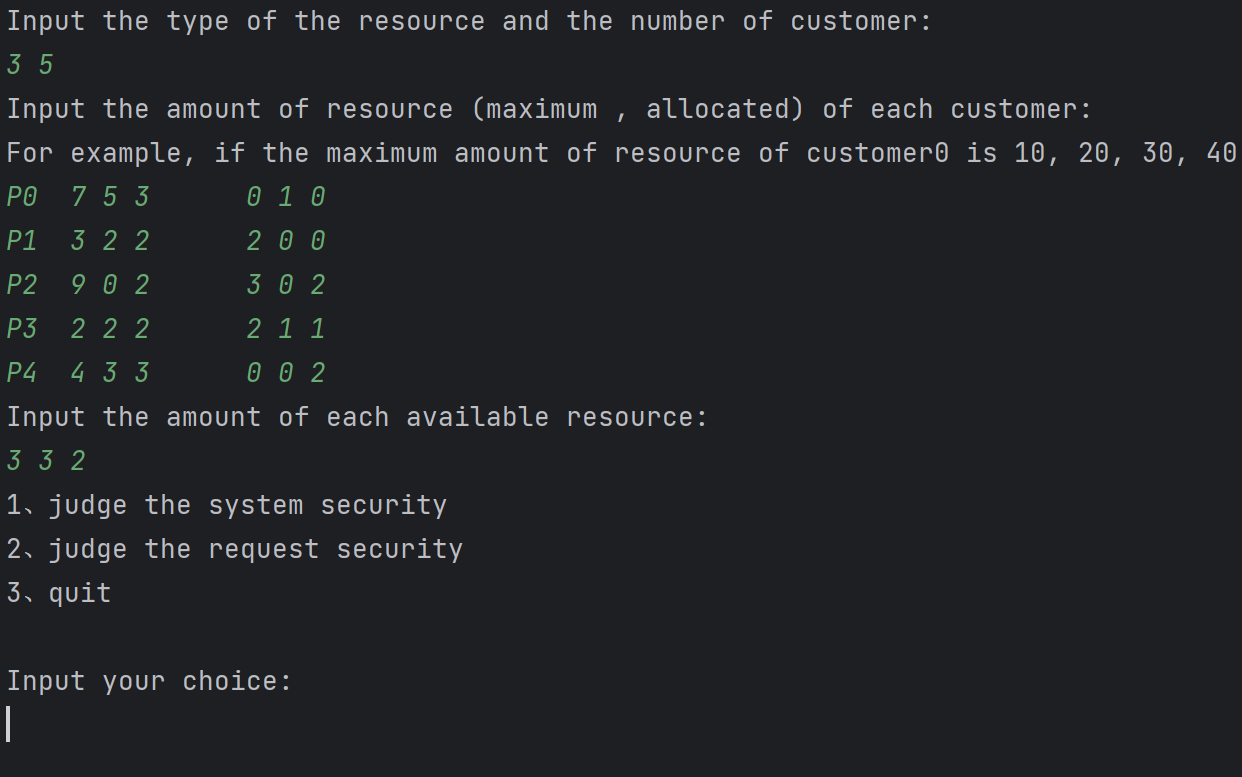
P4 4 3 3 0 0 2

得到如下界面：

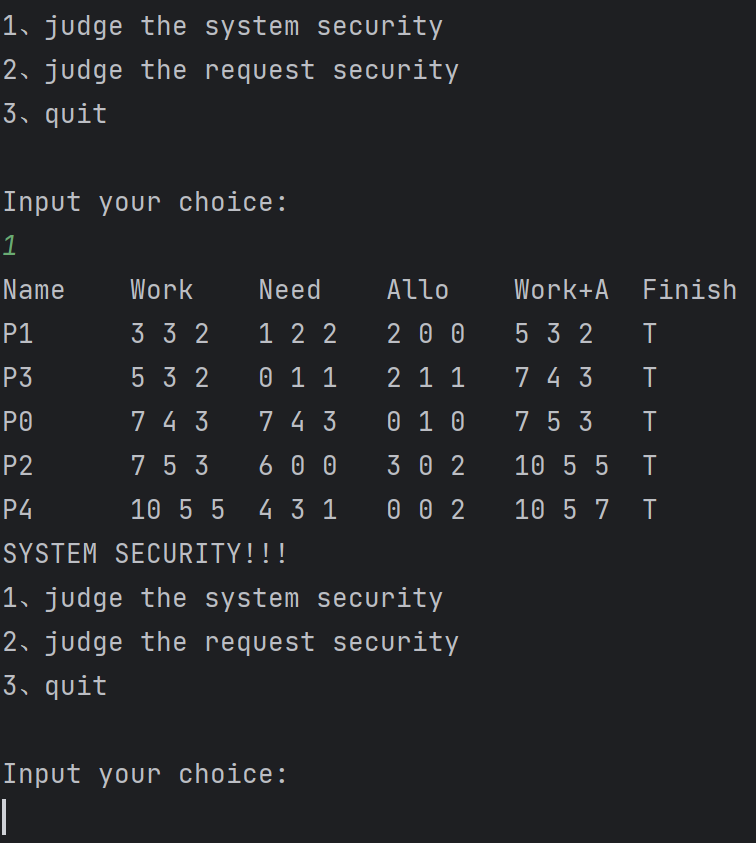


假设初始资源数是：3 3 2

则得到如下界面：



输入1，检测系统安全性

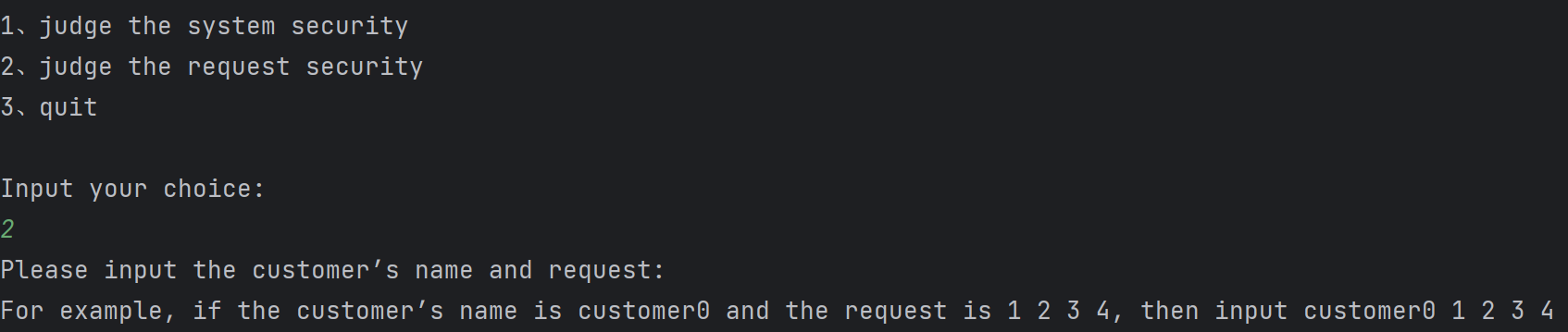


可以看到，系统安全。

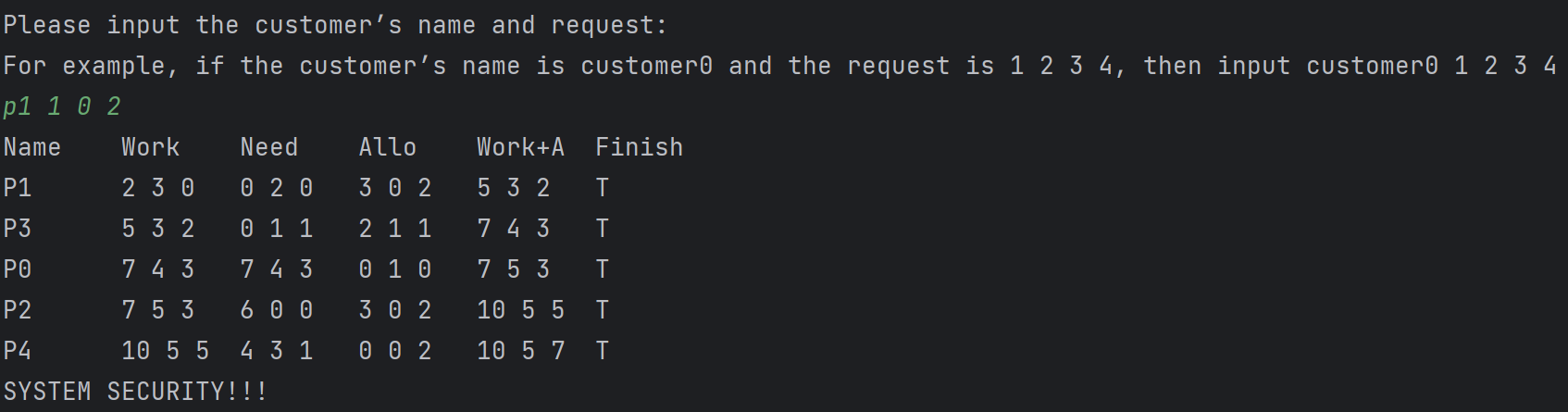
1. 银行家算法

为了判断给某项进程分配一些资源后，是否会导致系统不安全，需要使用到银行家算法。

接着实验1的内容，这时输入2：“判断请求安全性”。



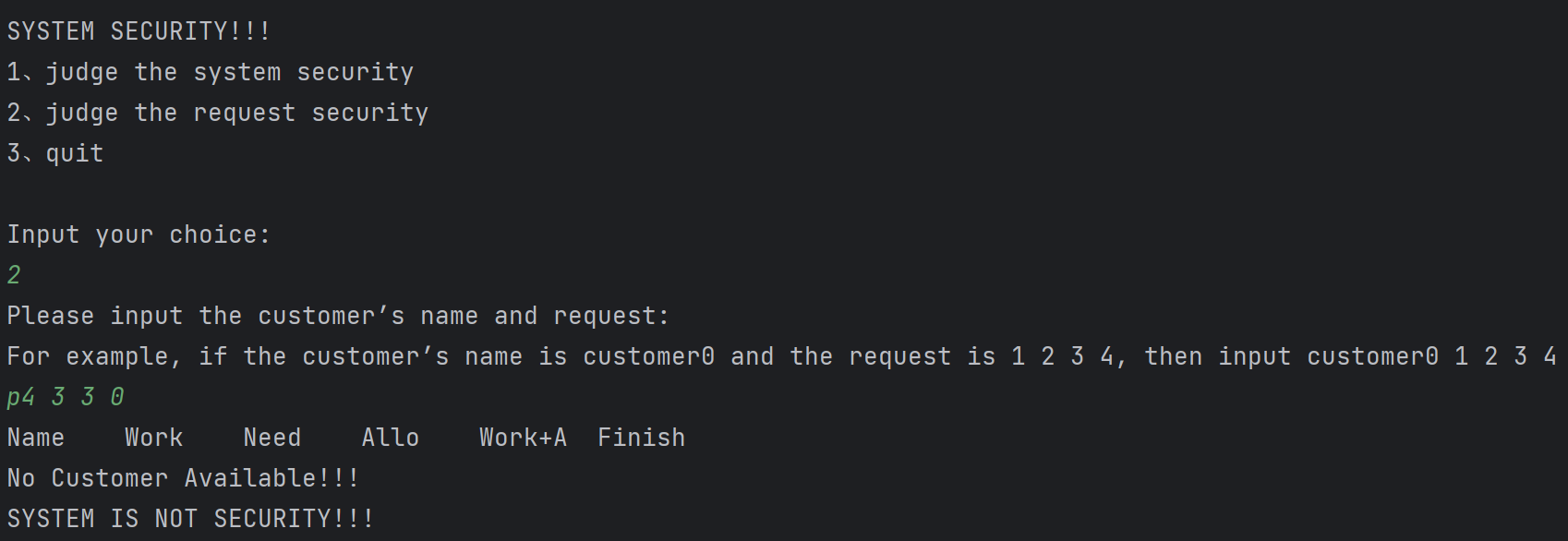
假设进程p1请求了1 0 2的资源，则输入p1 1 0 2。提示信息如下：



说明资源请求安全

假如进程p4请求了3 3 0的资源，则输入p4 3 3 0

那么打印信息如下：



说明进程不安全。

四、自评（评价是否完成实验要求及自己的收获）

在操作系统实验课上，我完成了资源分配算法的学习并成功实现了银行家算法和安全性算法的代码编写。

对于银行家算法，我深入理解了其原理和逻辑，并能够准确地判断系统状态的安全性与否。我能够正确地运用银行家算法来进行资源分配，并确保系统不会产生死锁。在实现代码时，我注重代码的可读性和优雅性，使用了相关的数据结构和算法，同时也进行了详尽的测试确保代码的正确性。我能够正确地处理资源的请求、释放和回收，并有效地保证资源的分配与释放的准确性。

在编写安全性算法的代码过程中，我熟悉了系统的资源分配情况及其对系统安全性的影响。我能够正确地判断系统是否处于安全状态，并能够实现相关算法以保证系统的安全性。在代码实现过程中，我注重了安全性算法的高效性和正确性，通过合理的数据结构和算法设计，确保系统可以正确地评估资源分配的安全性。

总体而言，我在操作系统实验课上对资源分配算法的学习和代码实现取得了较好的成果。我对于银行家算法和安全性算法有着深入的理解，并成功地将其转化为代码实现。通过这个实践，我提高了自己的编码能力、问题解决能力和代码质量意识。在未来的学习和工作中，我相信我能够更好地应用和扩展这些知识，为解决实际问题提供有效的资源分配方案。