|  |
| --- |
| **实验概述：** |
| **【实验目的及要求】**  **掌握Linux下银行家算法的实现**  **【实验原理】**  我们可以把操作系统看作是银行家，操作系统管理的资源相当于银行家管理的资  金，进程向操作系统请求分配资源相当于用户向银行家贷款。操作系统按照银行家制定的规则为进程分配资源，当进程首次申请资源时，要测试该进程对资源的最大需求量，如果系统现存的资源可以满足它的最大需求量则按当前的申请量分配资源，否则就推迟分配。当进程在执行中继续申请资源时，先测试该进程已占用的资源数与本次申请的资源数之和是否超过了该进程对资源的最大需求量。若超过则拒绝分配资源，若没有超过则再测试系统现存的资源能否满足该进程尚需的最大资源量，若能满足则按当前的申请量分配资源，否则也要推迟分配。  （2）安全状态：如果存在一个由系统中所有进程构成的安全序列P1，…，Pn，则系统处于安全状态。安全状态一定是没有死锁发生。  不安全状态：不存在一个安全序列。不安全状态一定导致死锁。  安全序列：一个进程序列{P1，…，Pn}是安全的，如果对于每一个进程Pi(1<=i<=n)，它以后尚需要的资源量不超过系统当前剩余资源量与所有进程Pj(j<i)当前占有资源量之和。  （3）设request[i]为进程P[i]的请求量，如果request[i][j]=*K*，表示进程P[i]需要*K*个Rj资源。当系统发出请求后，系统按下述步骤开始检查：  如果request[i][j]<=need[i][j]，转向步骤2；否则报告出错，申请的资源已经大于它需要的最大值。  如果request[i][j]<=available[j]，转向步骤3；否则报告出错，尚无足够的资源。  系统试探着把资源分配给p[i]，并修改下列数据结构中的值：  available[j]=available[j]-request[i][j]  allocation[i][j]=allocation[i][j]+request[i][j]  need[i][j]=need[i][j]-request[i][j]  系统进行安全状态检测算法，检查此次分配后，系统是否还处于安全状态，若安全，把资源分配给进程p[i]；否则，恢复原来的资源分配状态，让进程p[i]等待。  （4）安全状态检测算法：  int work[RESOURCE\_NUMBER];  bool finish[PROCESS\_NUMBER];  Work=Available;  Finish=false;  寻找满足条件的i：   * Finish[i]=false; * Need[i]<=Work;   如果不存在，则转。  Work: = Work + Allocation[i] ; Finish[i]: =true; 转。  若对所有i, Finish[i] = true，则系统处于安全状态，否则处于不安全状态。  **【实验环境】（使用的软件）**  **gcc编译器，linux平台，centos8系统** |
| **实验内容：** |
| **【实验方案设计】**  实验方案设计如下：   1. 创建一个包含资源分配和进程请求的银行家算法的实现程序。使用C语言编写程序。 2. 在程序中定义必要的数据结构，包括资源数量、进程数量、已分配资源矩阵、最大需求资源矩阵、需求资源矩阵、可用资源向量等。 3. 实现银行家算法的各个步骤：   - 首先，根据进程首次申请资源的最大需求量，判断系统现存资源是否足够满足该进程的最大需求量，如果是，则按照申请量分配资源，否则推迟分配。  - 当进程在执行中继续申请资源时，先判断该进程已占用的资源数与本次申请的资源数之和是否超过了该进程对资源的最大需求量，如果超过则拒绝分配资源，否则再判断系统现存资源能否满足该进程尚需的最大资源量，如果能满足则按照当前的申请量分配资源，否则推迟分配。  - 每次分配资源后，进行安全状态检测算法，判断系统是否处于安全状态。安全状态检测算法可以按照实验原理中描述的步骤进行实现。   1. 编译和运行程序，验证银行家算法的实现是否正确。   **流程图：**  Y  N  开始  资源请求数<  最大需求数？  资源请求数<  可用需求数？  是否为安全  状态？  修改可用资源、已分配资源、所需资源数目  系统进行安全性检查  恢复资源分配状态  结束  提示资源分配成功  N  N  Y  Y  **【实验过程】（实验步骤、记录、数据、分析）**  实验步骤：  1. 准备实验环境：确保已在Linux系统上安装了所需的编程语言和库。  2. 设计并实现银行家算法：根据银行家算法的原理，在编程语言中实现银行家算法的相关函数和数据结构。  3. 编写测试用例：根据银行家算法的要求，编写一系列测试用例来验证算法的正确性和性能。  4. 运行测试用例：使用编写的测试用例来运行实现的银行家算法，并记录运行结果。  5. 数据分析：根据运行结果，对算法的性能进行分析和评估。  **实验代码：**  #include<stdio.h> //头文件名,包含输入输出函数等  #include<stdlib.h> //头文件名  #include<time.h> //头文件名  #define PROCESS\_NUMBER 5 //宏定义,定义进程个数为5  #define RESOURCE\_NUMBER 3 //宏定义,资源类个数是3  #define true 1 //宏定义,定义true为1  #define false 0 //宏定义，定义false为0  typedef int bool; //宏定义，定义bool为int类型  int Available[RESOURCE\_NUMBER]={4,5,3}; //系统中三类可利用资源数分别为4，5，3  int Max[PROCESS\_NUMBER][ RESOURCE\_NUMBER]={{8,4,2},{3,3,2},{9,0,2},{2,1,2},{4,3,3}}; //每个进程所需求的资源最大数  int Allocation[PROCESS\_NUMBER][RESOURCE\_NUMBER]={  {0,1,1},  {2,0,0},  {2,0,2},  {2,1,1},  {0,0,2}};  int Need[PROCESS\_NUMBER][ RESOURCE\_NUMBER]; //各进程的当前需求向量  bool compare(int \*a,int \*b,int n) //比较两个一维数组，如果a中各项均大于b，则返回true，否则为false  { int i;  for(i=0;i<n;i++)  if(a[i]<b[i])  return false;  return true;  }  void assign(int \*a,int \*b,int n) //将数组b的值赋给a,n为数组的大小  { int i;  for(i=0;i<n;i++)  a[i]=b[i];  }  void add(int \*a,int \*b,int n) //两个一维数组的加法，各对应项相加  {  int i;  for(i=0;i<n;i++)  a[i]+=b[i];  }  void substract(int \*a,int\*b,int n) //两个一维数组的减法，各对应项相减  {  int i;  for(i=0;i<n;i++)  a[i]-=b[i];  }  void print(int \*a,int n) //打印一维数组  {  int i;  for(i=0;i<n;i++)  printf("%4d",a[i]);  printf("\n");  }  bool issafe(int \*sp) //判断是否是安全状态  {  int i;  int count=0; //记录finish[i]=true 的进程个数为0  int n=0;  int work[RESOURCE\_NUMBER];  bool finish[PROCESS\_NUMBER];  //work=av  assign(work,Available, RESOURCE\_NUMBER);  //初始化标记 finish  for(i=0;i< PROCESS\_NUMBER;i++)  finish[i]=false;  n= PROCESS\_NUMBER; //n为进城的个数  while(n--) //循环最多执行n次  for(i=0;i< PROCESS\_NUMBER;i++)  if(finish[i]==false&&compare(work,Need[i], RESOURCE\_NUMBER)) //判断能否满  //足进程i的要求，work＞＝need[i]是否成立  {  add(work,Allocation[i], RESOURCE\_NUMBER); //分配，待进程完成后再释放  finish[i]=true;  sp[count]=i; //记录安全路径  count++; //记录能够满足的进程数+1  }  if(count>= PROCESS\_NUMBER)  return true;  else  return false;  }  bool request(int pid,int \*r,int n)  {  int i;  int sp[5]; //记录安全路径  if(compare(Need[pid],r,n)==true&&compare(Available,r,n)==true) //如果  //request[i]<=need[i]且request[i]<=available  { //尝试分配资源  substract(Available,r, RESOURCE\_NUMBER);  add(Allocation[pid],r, RESOURCE\_NUMBER);  substract(Need[pid],r, RESOURCE\_NUMBER);  if(issafe(sp)) //判断是否是安全状态  {  printf("Security Path:\n\t");  for(i=0;i< PROCESS\_NUMBER;i++) //打印安全路径  printf("p%d",sp[i]);  printf("\n"); //可以分配  return true;  }  else  { add(Available,r, RESOURCE\_NUMBER); //不分配  substract(Allocation[pid],r, RESOURCE\_NUMBER); //恢复到分配前的状态  add(Need[pid],r, RESOURCE\_NUMBER);  return false;  }    }  else return false;  }  //提示信息  char hint()  { char ch;  printf("\t------------------Operation Hint----------\n");  //按A或者a键自动分配资源  printf("\tA(a)------------Apply For resource automated\n");  //按H或者h键自动分配资源  printf("\tH(h)-------------Apply For resource by human\n");  //按Q或者q键自动分配资源  printf("\tQ(q)-------quit\n");  scanf("%c",&ch);  return ch;  }  //显示系统信息  void init(){  int i;;  int temp[RESOURCE\_NUMBER];  printf("Processs Numbers: %d\n Need and Allocation respectively as follow:\n",PROCESS\_NUMBER);  //显示进程最大资源需求  for(i=0;i<PROCESS\_NUMBER;i++)  { printf("\t process %d max need:",i);  print(Max[i],RESOURCE\_NUMBER);  }  printf("\n");  //计算需求向量：Need[i]=MAX[i]-Alocation[i]  for(i=0;i<PROCESS\_NUMBER;i++){  assign(temp,Max[i],RESOURCE\_NUMBER);  substract(temp,Allocation[i],RESOURCE\_NUMBER);  assign(Need[i],temp,RESOURCE\_NUMBER);  }  //显示进程已分配资源  for(i=0;i<PROCESS\_NUMBER;i++){  printf("\t processs %d allocated resources:",i);  print(Allocation[i],RESOURCE\_NUMBER);  }  //显示系统可用资源  printf("\t available resources:\t");  print(Available,RESOURCE\_NUMBER);  }  //输入  void input(int \*r,int n,int \*id)  { int i;  //提示输入进程号  printf("please input process id(0~ %d):",n-1);  //从键盘输入进程号  scanf("%d",id);  for(i=0;i<n;i++){  printf("\nthe numbers of needed resource %d(int):",i);  scanf("%d",&r[i]);  }  //显示刚才输入的数据  printf("\ndata you inputed: Request[%d](",\*id);  for(i=0;i<n;i++)  printf("%d",r[i]);  printf(")\n");  }  //检查输入  bool check(int id,int \*r,int n)  {  int i;  //判断申请资源数目是否合法  for(i=0;i<n;i++)  if(r[i]<0)  return false;  //判断进程号是否合法  if(id>=PROCESS\_NUMBER)  return false;  else  return true;  }  int main()  {  //进程id号  int id;  //控制字符  char control;  //资源请求向量  int r[3];  //显示开始信息  init();  //随机数初始化  srand((int)time(0));  //主控过程  while(1)  {  //提示  control=hint();  if(control=='a'||control=='A')  {  //随机申请资源  id=rand()%5;  r[0]=rand()%5;  r[1]=rand()%5;  r[2]=rand()%5;  //显示申请信息  printf("\tRequest[%d](%d,%d,%d)\n",id,r[0],r[1],r[2]);  if(request(id,r,RESOURCE\_NUMBER))  printf("Alloc Success!\n");  else  printf("Alloc Failed!\n");  }  else if(control=='h'||control=='H')  {  //输入申请信息  input(r,RESOURCE\_NUMBER,&id);  //检查输入是否合法  if(check(id,r,RESOURCE\_NUMBER)==false)  {  printf("\nInput Error!please reinput!\n");  continue;  }  //换行  if(request(id,r,RESOURCE\_NUMBER))  printf("Request Succeed!\n");  else  printf("Request Fail!\n");  }  else if(control=='q'||control=='Q')  exit(0);  //显示当前系统资源和进程情况  printf("Available Resource\n");  print(Available,RESOURCE\_NUMBER);  //显示资源最大需求  printf("process %d max need\n",id);  print(Max[id],RESOURCE\_NUMBER);  //显示已分配资源情况  printf("process %d allocated resources\n",id);  print(Allocation[id],RESOURCE\_NUMBER);  }  return 0;  }  **【结论】（结果）**    **运行界面**    **输入a**    **输入h**    **程序退出** |
| **【小结】**  **实验的体会及收获：**   1. 此次实验思路均通过自己的思考，做出的程序符合自己的思路和预期。 2. 链表的每个模块对数据更改的，最好需要输出整个链表相应数据，进行测试。 3. 程序应该设计的完整，如没有安全序列，应该返回语句:这些进程没有安全序列。 4. 链表的操作比数组的实现更容易懂，更好实现。 5. 后面在作业中发现，初始的剩余资源量需要自己输入，代码已经做出修改   **建设性建议：**  银行家算法是一种用于避免死锁的资源分配算法，它通过判断系统是否处于安全状态来决定是否分配资源。根据这一算法的原理，我可以给出一些建设性的建议：  1. 设计合理的资源分配策略：在系统设计阶段，应该考虑到各个进程对资源的需求，并合理安排资源的分配。可以使用银行家算法作为参考，分析系统的资源使用情况，确保资源分配的安全性和可靠性。  2. 实时监控资源使用情况：建议实施一套系统来实时监控资源的使用情况，包括当前可用资源数、进程的资源需求和已分配资源等信息。这样可以及时发现资源紧张的情况，并采取相应的措施来避免死锁的发生。  3. 引入资源预留机制：为了避免资源的过度分配和竞争，可以引入资源预留机制。即在资源分配之前，要求进程提前申请所需资源的数量，系统根据可用资源和进程的需求来判断是否分配资源。这样可以有效地避免资源的浪费和冲突。  4. 定期进行资源回收和释放：建议定期对已分配的资源进行回收和释放，确保资源的有效利用。这可以通过监控系统中的空闲资源和闲置进程来实现，及时回收不再需要的资源，并重新分配给其他需要的进程。  5. 强化系统安全性和稳定性：除了银行家算法外，还应考虑其他安全性和稳定性的措施。例如，设计合理的进程调度算法，确保资源的公平分配；实施事务管理机制，保证资源在使用过程中的完整性和一致性；以及建立异常处理机制，及时处理资源分配过程中的异常情况。 |
| **指导教师评语及成绩：** |
| **评语：**  **成绩： 指导教师签名：**  **批阅日期：** |

实验报告说明

**1．实验项目名称：**要用最简练的语言反映实验的内容。要求与实验指导书中相一致。

**2．实验类型：**一般需说明是验证型实验还是设计型实验，是创新型实验还是综合型实验。

**3．实验目的与要求**：目的要明确，要抓住重点，符合实验指导书中的要求。

**4．实验原理：**简要说明本实验项目所涉及的理论知识。

**5．实验环境**：实验用的软硬件环境（配置）。

**6．实验方案设计（思路、步骤和方法等）**：这是实验报告极其重要的内容。概括整个实验过程。

对于**操作型实验**，要写明依据何种原理、操作方法进行实验，要写明需要经过哪几个步骤来实现其操作。

对于**设计型和综合型实验**，在上述内容基础上还应该画出流程图、设计思路和设计方法，再配以相应的文字说明。

对于**创新型实验**，还应注明其创新点、特色。

**7．实验过程（实验中涉及的记录、数据、分析）：**写明具体上述实验方案的具体实施，包括实验过程中的记录、数据和相应的分析。

**8．结论（结果）：**即根据实验过程中所见到的现象和测得的数据，做出结论。

**9．小结：**对本次实验的心得体会、思考和建议。

**10．指导教师评语及成绩：**指导教师依据学生的实际报告内容，用简练语言给出本次实验报告的评价和价值。

**注意：**

* 实验报告将记入实验成绩；
* 每次实验开始时，交上一次的实验报告，否则将扣除此次实验成绩。