

**实 验（实训）报 告**

**项 目 名 称**  进程调度算法

**所属课程名称**  操作系统

**项 目 类 型**  验证/设计型

**实验(实训)日期**  2024.11.13

**班 级**  22软件1

**学 号**  220110900602

**姓 名**  陈正

**指导教师**  陈伟锋

浙江财经大学教务处制

|  |
| --- |
| **一、实验（实训）概述：** |
| **【目的及要求】**  **模拟动态优先调度算法。**  **【基本原理】**      **【实施环境】**  **LinuxLTS22.04.3** |
| **二、实验（实训）内容：**  **输入命令：sudo apt-get install gcc**  **安装结束后执行gcc –version，出现以下界面即可认为成功：**    **完成hello.c：**  **编辑文件**    **编译 gcc hello.c**  **生成了a.out文件，执行a.out文件，输出helloWorld：**    **完成fork.c：**  **示例的源码缺少<sys/wait.h>的头文件，加入后才能成功编译：**    **运行结果为：**    **完成processes.c：**  **步骤同上，代码为：**    **最终结果为：**    **完成communication.c：**  **代码为：**        **运行结果为：** | |
| **二、实验（实训）内容：**  **要求：**    **样例输入：**  **5**  **9 0 3 2 3**  **38 0 3 1 1**  **30 0 6 4 2**  **29 0 3 1 1**  **0 0 4 1 2**  **1**  **输出：**            **样例输入：**  **5**  **9 0 3 2 3**  **38 0 3 1 1**  **30 0 6 4 2**  **29 0 3 1 1**  **0 0 4 1 2**  **0**    **代码（在下一页，截图+文本）：**    #include<stdio.h>  #include<string.h>  #include<stdlib.h>  #include<time.h>  #include<iostream>  #include<algorithm>  #include<vector>  using namespace std;  const int N = 1010;  int n;  int arr[N];  vector<int>final;  enum State { ready, block, done };  struct Process {  int ID;  int priority;  int cputime;  int alltime;  int startblock;  int blocktime;  State state;  };  Process process[N];  void print() {  cout << "就绪队列:";  for (int j = 0; j < 5; j++) {  if (process[j].state == ready) {  cout << "->" << process[j].ID + 1;  }  }  cout << endl;  cout << "阻塞队列:";  for (int j = 0; j < 5; j++) {  if (process[j].state == block) {  cout << "->" << process[j].ID + 1;  }  }  cout << endl;  cout << "ID\tpriority\tcputime\talltime\tstartblock\tblocktime\tstate" << endl;  for (int j = 0; j < 5; j++) {  cout << process[j].ID + 1 << '\t' << process[j].priority << '\t' << '\t' << process[j].cputime << '\t' << process[j].alltime << '\t' << process[j].startblock << '\t' << '\t' << process[j].blocktime << '\t' << '\t' << process[j].state << endl;  }  cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;  }  int main() {  cin >> n;  for (int i = 0; i < n; i++) {  process[i].ID = i;  process[i].priority = 0;  process[i].cputime = 0;  process[i].alltime = 0;  process[i].startblock = 0;  process[i].blocktime = 0;  process[i].state = ready;  }  for (int i = 0; i < n; i++) {  int pri, cput, allt, startb, blot;  cin >> pri >> cput >> allt >> startb >> blot;  //cout<<pri<<cput<<allt<<startb<<blot<<endl;  process[i].priority = pri;  process[i].cputime = cput;  process[i].alltime = allt;  process[i].startblock = startb;  process[i].blocktime = blot;  }  int x = 1;  cin>>x;  //cout<<x;  int timeslice = 0;//时间片  int temp[N];//阻塞进程优先级 缓存数组  bool flag = true;  while (flag) {//大坑，如果还有优先级不为-1的  flag = false;  for (int i = 0; i < n; i++) {  if (process[i].priority > 0)flag = true;//也是坑，要用优先级判断，不然阻塞进程会进入死循环  }  //cout << "i:" << i << endl;  //存在进程没有执行完成  for(int i=0;i<n;i++){  for (int j = 0; j < n; j++) {  arr[j] = process[j].priority;  }  //cout<<process[i].state<<" "<<process[i].priority<<endl;  if ((process[i].state == ready) && (process[i].priority == \*max\_element(arr, arr + n))) {  //执行进程i  //cout<<"zhixing"<<endl;  timeslice++;  process[i].priority -= 3;  process[i].cputime += 1;  process[i].alltime -= 1;  for (int j = 0; j < n; j++) {  if (j == i)continue;  if (process[j].alltime > 0 && process[j].state == ready)  process[j].priority += 1;//其余就绪进程优先级加1  }  if (process[i].alltime == 0) {//如果完成  process[i].state = done;  process[i].priority = -1;  }  for (int j = 0; j < n; j++) {  if (process[j].state == block && process[j].blocktime > 0) {  //如果是阻塞进程  process[j].blocktime -= 1;  if (process[j].blocktime == 0) {  //状态改变  process[j].state = ready;  process[j].priority = temp[j];//恢复阻塞前的优先级  }  }  }  if (process[i].startblock > 0) {  process[i].startblock -= 1;  if (process[i].startblock == 0) {  process[i].state = block;  temp[i] = process[i].priority;//存储当前优先级  process[i].priority = -1;  }  }  if (x) {  cout << "时间片为:" << timeslice << endl;  cout << "执行进程:" << process[i].ID << endl;  print();  }  final.push\_back(process[i].ID+1);  }  }  }  if (!x) {  cout << "调度序列ID：" << endl;  for (auto it : final)  cout <<it << " ";  }  return 0;  } |
| **三、指导教师评语及成绩：** |
| **评语：**  **成绩： 指导教师签名：**  **批阅日期：** |