

**实 验（实训）报 告**

**项 目 名 称**  进程通信与进程调度

**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**  2024.10.16

**班 级**  22软件1班

**学 号**  220110900818

**姓 名**  刘景婷

**指导教师**  陈伟锋

浙江财经大学教务处制

|  |
| --- |
| **一、实验（实训）概述：（简单复述实验要求，1页以内，红字部分应删除）** |
| **【目的及要求】**  1.安装gcc  2.完成hello.c  3.完成fork.c  4.完成processes.c  5.完成communication.c  6.用C语言实现对N个进程采用动态优先权算法的调度  **【基本原理】**  通过特定的机制实现进程间的信息交换与同步，并根据一定的调度算法动态地分配CPU资源给多个进程，以确保系统资源的有效利用和任务的高效执行。  **【实施环境】**  **虚拟机centos7，dev C++** |
| **二、实验（实训）内容：** |
| **【实验（实训）过程】（步骤、记录、数据、程序等）**  1.  2.创建hello.c文件  vim hello.c  使用vim编辑代码    保存退出  编译  gcc hello.c -o hello（命名）  运行  ./hello    3.代码：    运行结果：  4.运行结果：  5.运行结果：    6.代码：  #include <iostream>  #include <algorithm>  using namespace std;    enum State {ready,block,done};  struct PCB  {  int ID;//进程标识数  int priority;//进程优先数  int cputime;//进程已占用的CPU时间  int alltime;//进程还需要占用的CPU时间  int startblock;//进程的阻塞时间  int blocktime;//进程被阻塞的时间  State state;//进程状态  struct PCB \* next;//进程控制块的next指针    }pcb[5];    int main()  {  pcb[0] = {1,9,0,3,2,3,ready};  pcb[1] = {2,38,0,3,1,1,ready};  pcb[2] = {3,30,0,6,4,2,ready};  pcb[3] = {4,29,0,3,1,1,ready};  pcb[4] = {5,0,0,4,1,2,ready};  int time=0; //时间片  int temp[5]; // 阻塞进程优先级 缓存数组  while(pcb[0].alltime||pcb[1].alltime||pcb[2].alltime||pcb[3].alltime||pcb[4].alltime) //判断是否存在进程没有执行完成  {    for(int i=0; i<5; i++)  {  int array[]= {pcb[0].priority,pcb[1].priority,pcb[2].priority,pcb[3].priority,pcb[4].priority};//存储数组  if((pcb[i].state==ready) && (pcb[i].priority==\*max\_element(array,array+5)))  {//进程i开始执行，使用库中max\_element(first,last)函数，函数返回值为数组最大元素的地址    time++;//时间片+1  pcb[i].priority-=3;//执行进程优先数减3  pcb[i].cputime+=1;//执行进程已占用的CPU时间+1  pcb[i].alltime-=1;//执行还需要占用的CPU时间alltime-1    for(int j=0; j<5; j++)//其余就绪进程优先级 +1  {    if(j==i) continue;//跳过当前进程的优先级+1操作  if(pcb[j].alltime>0&&pcb[j].state==ready)//占用的CPU时间alltime不为0且将进程状态置为ready  {  pcb[j].priority+=1;//进程优先数+1  }  }    if(pcb[i].alltime==0)//如果占用的CPU时间alltime=0，则将进程状态置为done  {  pcb[i].state=done;//进程状态变为结束  pcb[i].priority=-1;//进程优先数-1  }      for(int j=0; j<5; j++) //其他阻塞进程 blocktime-1  {    if(pcb[j].state==block&&pcb[j].blocktime>0)//进程状态阻塞且进程被阻塞的时间不为0  {    pcb[j].blocktime-=1;//进程被阻塞的时间blocktime-1  if(pcb[j].blocktime==0)//若进程被阻塞的时间为0，则进程状态置为ready  {  pcb[j].state=ready;  pcb[j].priority=temp[j]; //阻塞进程恢复阻塞前的优先级  }  }  }    if(pcb[i].startblock>0)//若进程的阻塞时间不为0，startblock-1  {  pcb[i].startblock-=1;  if(pcb[i].startblock==0)//若进程的阻塞时间为0，进程状态置为block  {  pcb[i].state=block;  temp[i]=pcb[i].priority;//进程i阻塞后 将优先级保存到 temp[i]  pcb[i].priority=-1;//优先数-1  }  }    cout<<"Running\_program "<<pcb[i].ID<<endl;  void show\_PCB();//显示当前调度进程，就绪队列，阻塞队列及每个进程的相应状态  show\_PCB();    }      }    }  return 0;  }  void show\_PCB()//显示当前调度进程，就绪队列，阻塞队列及每个进程的相应状态  {    cout<<"Ready\_queue ";  for(int j=0; j<5; j++)  {  if(pcb[j].state==ready)  {  cout<<"->"<<pcb[j].ID;  }  }  cout<<endl;  cout<<"Block\_queue ";  for(int j=0; j<5; j++)  {  if(pcb[j].state==block)  {  cout<<"->"<<pcb[j].ID;  }  }  cout<<endl;  cout<<"============================================================="<<endl;  cout<<"ID\t";  for(int j=0; j<5; j++) cout<<'\t'<<pcb[j].ID;  cout<<endl;  cout<<"priority\t";  for(int j=0; j<5; j++) cout<<pcb[j].priority<<'\t';  cout<<endl;  cout<<"cputime\t";  for(int j=0; j<5; j++) cout<<'\t'<<pcb[j].cputime;  cout<<endl;  cout<<"alltime\t";  for(int j=0; j<5; j++) cout<<'\t'<<pcb[j].alltime;  cout<<endl;  cout<<"startblock\t";  for(int j=0; j<5; j++) cout<<pcb[j].startblock<<'\t';  cout<<endl;  cout<<"blocktime\t";  for(int j=0; j<5; j++) cout<<pcb[j].blocktime<<'\t';  cout<<endl;  cout<<"state\t";  for(int j=0; j<5; j++) cout<<'\t'<<pcb[j].state;  cout<<endl;  cout<<endl;  }  运行结果：      **【结论与讨论】（结果、分析）**  **本次实验主要任务都是关于进程通信与进程调度，这些任务展示了使用C语言在进程管理、调度算法以及通信协议实现等方面的广泛应用。通过实践这些例子，加深了我对C语言及其在不同领域应用的理解。** |
| **三、指导教师评语及成绩：** |
| **评语：**  **成绩： 指导教师签名：**  **批阅日期：** |