

**实 验（实训）报 告**

**项 目 名 称**  （实验名称）

**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**  2024/10/23

**班 级**  22软工1班

**学 号**  220110900624

**姓 名**  宋思泽

**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】**  安装gcc，完成hello.c、fork.c、proceses.c、communication.c  **【基本原理】**  **【实施环境】** |
| **二、实验（实训）内容：（记录实验内容，要求逻辑清晰、页面整洁、必要的图文并茂，核心的数据及程序代码，不能完全复制所有代码，完整代码若有必要，可以增加附录，尽量控制在6页以内，红字部分应删除）** |
| **【实验（实训）过程】（步骤、记录、数据、程序等）**  安装gcc:    完成hello.c:    完成fork.c:    完成Processes.c:    完成communication.c:      实验六：      代码如下：  #include <iostream>  #include <algorithm>  using namespace std;  enum State {ready,block,done}; //定义一个枚举类型 0-ready（准备完毕）   1-block（阻塞）    2-done（结束）  struct PCB  {      int ID;//进程标识数id      int priority;//进程优先数priority      int cputime;//进程已占用的CPU时间cputime      int alltime;//进程还需要占用的CPU时间alltime      int startblock;//进程的阻塞时间startblock      int blocktime;//进程被阻塞的时间blocktime      State state;//进程状态state      struct PCB \* next;//进程控制块的next指针    }pcb[5];//进程结构体声明，同时定义进程结构体变量，5个进程    int main()  {        //定义五个进程的初始状态      pcb[0] = {0,4,1,4,2,3,ready};      pcb[1] = {1,3,0,3,-1,0,ready};      pcb[2] = {2,7,0,4,-1,0,ready};      pcb[3] = {3,8,0,5,-1,0,ready};      pcb[4] = {4,0,0,6,-1,0,ready};      int time=0; //定义一个时间片，初始值为0      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<<"时间片为: "<<time<<endl;                  cout<<"执行进程: "<<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<<"就绪队列: ";      for(int j=0; j<5; j++)      {          if(pcb[j].state==ready)          {              cout<<"->"<<pcb[j].ID;          }      }      cout<<endl;      cout<<"阻塞队列: ";      for(int j=0; j<5; j++)      {          if(pcb[j].state==block)          {              cout<<"->"<<pcb[j].ID;          }      }      cout<<endl;      cout<<"ID\tpriority\tcputime\talltime\tstartblock\tblocktime\tstate"<<endl;      for(int j=0; j<5; j++)      {          cout<<pcb[j].ID<<'\t'<<pcb[j].priority<<'\t'<<'\t'<<pcb[j].cputime<<'\t'<<pcb[j].alltime<<'\t'<<pcb[j].startblock<<'\t'<<'\t'<<pcb[j].blocktime<<'\t'<<'\t'<<pcb[j].state<<endl;      }      cout<<"----------------------------------------------------------------------------"<<endl;  }  **【结论与讨论】（结果、分析）** |
| **三、指导教师评语及成绩：** |
| **评语：**  **成绩： 指导教师签名：**  **批阅日期：** |