文章编号：1672-5913(2009)14-0144-05

进程管理演示系统的设计与实现

张琼声[1]，蒋玉新[1]，李春华[2]，刘童璇[1]

(1.中国石油大学(华东)计算机与通信工程学院，山东 东营 257061；2.胜利油田物探研究院 计算机室，山东 东营 257022)

摘 要：进程是资源分配和独立运行的基本单位，是操作系统的核心概念。“操作系统”教学中，进程的概念以及进程管理的实现原理抽象难懂，初学者难以掌握。本文阐述如何以图形化方式设计和实现进程管理的演示系统，以辅助课堂教学。该系统的演示内容包括：进程的概念、进程创建、进程组织、进程关系管理、进程阻塞、进程唤醒、进程撤销、进程调度、进程同步。

关键词：进程管理；演示系统；操作系统

中图分类号：G642 **文献标识码：**B

1 前言

进程管理是操作系统原理最主要的教学内容之一，而进程及进程的控制原理是学生学习的重点和难点。如何使学生能够在较短的时间内，深入了解进程的概念及进程控制的原理，如何把进程的概念与程序运行的软硬件环境的变化联系起来？如何把进程管理的功能与数据结构和算法的实现结合起来？使学生从根本上掌握进程的概念，理解操作系统中进程管理功能的实现原理和实现技术，把抽象的理论与具体的实现技术结合起来？是“操作系统”课程教学面临的重要问题。

进程管理演示系统主要用于辅助课堂教学，试图将抽象的理论与系统设计、实现的具体技术相结合，通过动态的、图形化的界面表现进程概念的本质、进程管理的过程、进程管理功能与数据结构和算法实现的关系。把抽象的概念和原理实例化。帮助学生直观地、深入地理解进程的概念和进程管理功能存在的必要性以及相应的实现技术。

本系统主要实现进程概念、进程控制、进程调度、进程同步行为和实现原理的演示。该系统的特点是用图形化的方式把操作系统原理与程序实现结合起来。论文详细说明了该演示系统的设计方案与实现技术。

2 系统设计

2.1 系统模块结构

本系统包括进程概念、进程控制、进程调度、进程同步四个演示模块。其中进程控制演示模块包括进程创建、进程终止、进程阻塞与唤醒三个演示子模块。进程调度演示模块包括单级队列调度和多级队列调度演示两个子模块。进程同步演示模块包括进程互斥和读者—写者问题演示两个子模块。

本系统用 VC++6.0 开发，采用单文档结构，所有演示过程都在视图中通过 VC 控件交互实现。系统使用了延时机制，每当执行一个过程使界面发生变化或执行了关键步骤后，执行一个延时函数，从而给用户足够的时间观察界面的变化。

2.2 进程组织

(1) 链表组织

本系统实现多个进程链表，包括总进程链表、多个优先权不同的就绪进程链表和三个对应不同阻塞事情的阻塞进程链表。

(2) 进程树

系统按照进程的亲属关系，建立进程树，实现了进程树的管理和图形显示。

(3) 进程标识符 PID 的管理

每一个进程都有唯一的内部标识符 PID，本系统通过循环使用来达到有限 PID 资源的合理利用。当进程创建时分配可用的 PID，当进程终止时，释放占用的 PID。

2.3 进程执行过程的模拟

本系统通过定时器和执行时间来模拟进程的执行过程。

创建进程时，给进程一个随机的执行时间。执行时间长短根据系统参数配置进行灵活控制。进程同步中对临界资源的访问过程也通过访问时间来模拟，根据进程的执行时间，进程访问临界资源的时间总是定义为一个比总执行时间小的值。

作者简介：张琼声(1968-)，女，湖北松滋县人，硕士，副教授，从事操作系统、软件工程、决策支持系统方面的教学和研究。

给定进程的执行时间后，通过定时器控制进程的执行。进程执行一条指令用一个定时周期来模拟，在定时处理函数中对进程控制块的相关数据进行修改并同步在界面上更新显示，从而模拟出进程的执行过程。当定时周期数等于给定的进程执行时间时(本系统在进程控制块中添加了已执行时间来记录执行进度，该值就等于定时周期数)，进程正常结束。

2.4 进程概念演示模块

本模块通过显示当前进程的 PCB 信息、CPU 寄存器的变化来演示进程概念。模拟了进程实体、进程控制块、动态特征、短暂存在性、进程切换、并发执行、独立性等特点。

系统界面上显示的进程信息都直接或间接地从进程控制块中获取。

2.5 进程控制演示模块

(1) 进程创建演示模块

本模块实现进程创建过程的演示。用户可以输入新进程的外部标识符即进程名称，若用户没有输入，则系统自动为进程命名，每一个进程的创建都是有引发事件的，用户可以在界面上选择一个引发进程创建的事件。当用户点击“创建进程”按钮时就开始执行创建进程过程并同步显示进程创建的每一个执行步骤。

(2) 进程终止演示模块

本模块实现并演示各种不同情况下进程终止的过程。用户可以通过输入 PID 来终止某个进程，也可以通过选择界面上进程列表中的某个进程来终止被选中的进程。执行态进程被终止后要转进程调度程序。

(3) 进程阻塞与唤醒演示模块

本模块实现并演示进程阻塞与唤醒的过程。进程执行过程中用户可以选择阻塞事件并阻塞当前进程。当有阻塞态进程时，用户可以选择相关事件唤醒阻塞列表中的进程。

本模块设置了三个阻塞事件：打印机服务、I/O 设备操作和无新数据输入。

2.6 进程调度演示模块

本模块实现了单级队列调度和多级队列调度的演示。分别实现了抢占、非抢占、优先级、时间片等调度策略的模拟。多级队列调度的演示，模拟了 Minix 的三级队列(任务级、服务级和用户级)调度 ，定义了三个优先级不同的就绪队列。

2.7 进程同步演示模块

(1) 进程互斥演示模块

本模块实现并演示了多个进程互斥访问同一个临界资源的控制过程。通过该演示过程，使用户了解操作系统是如何实现进程对临界资源的互斥访问的。

本模块中，对临界资源的访问可以由用户控制，也可以系统自动控制。即在进程执行过程中，用户可以发送访问临界资源的命令，让其执行访问临界资源的过程。自动控制的设计是若某进程没有访问过临界资源，则令其在执行过程的最后时间段自动访问临界资源。

(2) 读者—写者问题演示模块

本模块演示多个读进程与写进程同步访问共享数据区的管理过程。创建进程时，用户要指定新进程的类别(读者进程或写者进程)。用户可以通过进程列表选择任何进程执行，执行过程中，用户可以随时让进程访问资源。

2.8 系统参数配置

本系统为了灵活控制演示过程并满足用户的需要，设置了一些配置参数，如定时周期、最小或最大延时时间、最小或最大执行时间、优先级、最大进程数等。

2.9 系统界面设计

演示界面设计如图 1 所示：

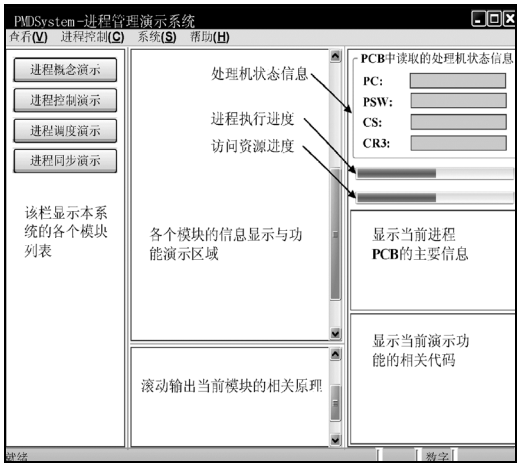

图 1 系统界面框架设计图

3 系统实现

3.1 进程控制块 PCB

(1) PCB 结构体定义

```
struct  PCB
{
                              //进程标识符信息
    UINT  nPID;               //进程的内部标识符
    CString szName;           //进程的外部标识符
    PCB *pParent;             //进程的父进程指针
    PCB *pFirstChild;         //进程的子进程指针
    PCB *pNextSibling;        //兄弟进程指针
                              //处理机状态信息
    DWORD dwPC;               //下一条指令地址
    DWORD dwPSW;              //程序状态字
    DWORD dwCS;               //段地址
    DWORD dwPageTableAddr;  //最外层页表地址,32 位
```

```
                        //进程调度信息
    UINT iState;        //进程的状态
    UINT iPriority;     //进程的优先级
    UINT iPriorUpSteps; //优先级动态提升的级数
    int nTotalTime;     //进程的总执行时间
    int nHasExecTime;   //进程已经执行的时间
    int nWaitTime;      //等待时间
    UINT iWaitEvent;    //等待事件
    UINT iCritSrcSort; //进程访问的临界资源类别
    int nCritSrcTotalT;
    //进程访问临界资源的总时间
    int nCritSrcHasExecT;
    //进程已经访问临界资源 的时间
    UINT iCritSrcVisitPos;
    //进程访问临界资源的阶段
    //进程控制信息
    int nKernelStackPos;//内核栈指针位置
    UINT iUser;         //进程的用户名
    UINT iSort;         //进程的类别
    UINT nWaitedSema; //已经等待过信号量的标志
    list_head  pTasksList; //进程链表的指针
    list_head pStateList; //链接状态链表的指针
    RECT *aAddrSpace[3];//关联进程三部分模拟空间
};
```

(2) PCB 结构体说明

● 组织进程关系的域

pParent 记录每一个进程的父进程，pFirstChild 记录第一个儿子，pNextSibling 记录下一个兄弟，通过这种方式记录了整个父子关系树，进程关系的管理通过这三个域实现。

● 记录寄存器信息的域

dwPC、dwPSW、dwCS、dwPageTableAddr 分别保存了当前 CPU 中主要的寄存器信息，通过这些信息来管理进程并发执行以及独立运行。dwPC 在每一次定时处理中加 1，模拟一条指令的执行，dwPSW 在执行过程中随机变化，模拟指令执行对 CPU 标志的影响。每个进程的 dwCS 和 dwPageTableAddr 值不同且不变，在创建进程时要保证它们的唯一性。

● 优先级

iPriority 是进程的优先级。本系统设定了六种优先级，分别为 PRIO_REALTIME(实时)、PRIO_HIGHER(高)、PRIO_UPSTANDARD(高于标准)、PRIO_STANDARD(标准)、PRIO_DOWNSTANDARD(低于标准)、PRIO_LOWER(低)。该值是动态的，在某些模块中，创建进程时系统自动设为 PRIO_STANDARD(标准)；在某些模块中，创建进程时由用户指定。在所有模块中，每一次进程调度时根据 nWaitTime(等待时间)以及系统参数 m_nUpPrioSpan(优先级升级的基数，针对等待时间)对就绪进程的 iPriority 值进行相应的修改，但最高只能升级到 PRIO_HIGHER(高)。

● 执行时间

nTotalTime 是进程的执行时间，它决定了进程的生存期，用于控制进程的执行。系统进程(0 号进程和 1 号进程)的 PCB 中，其值为-1，表示进程一直存在。

● 已执行时间

nHasExecTime 是进程的已执行时间，用于控制进程的执行进度。进程执行时，在每一次定时处理中加该域值 1，当它等于 nTotalTime 时，表示进程执行完毕。

● 等待时间

nWaitTime 是进程的等待时间，用于动态修改进程的优先级，该值在每一次定时处理中加 1，但执行态进程的 nWaitTime 始终为 0，只有当执行态进程切换为其它状态时，才开始累积该值。当进程的状态发生改变而链入到其它状态链表中时，该值清 0。

● 等待事件

iWaitEvent 是进程的等待事件，本系统将此值的范围进行了扩充。对于就绪进程，该值为 WAIT_CPU(等待 CPU)；对于执行态进程，该值为 WAIT_NULL(没有等待事件)；对于阻塞进程，该值表示阻塞事件，可以取值为 WAIT_PRINTER(等待打印机)，WAIT_IOSTREAM(等待 I/O 设备)，WAIT_NEWDATA(等待新数据)；在进程同步模块中，对于等待临界资源的进程该值为 WAIT_CRITSRC(等待临界资源)，等待的资源保存在 iCritSrcSort 中。

● 访问的临界资源种类

iCritSrcSort 是进程访问临界资源的种类，也就是对应的信号量类别。本系统中可以取值为 CRITSRC_PRINTER(打印机资源)、CRITSRC_GLOBAL (全局变量资源)、CRITSRC_READWRITE(读写资源)。对于不需要访问临界资源的进程来说该值为 CRITSRC_NULL(无临界资源)。

● 访问临界资源的时间

nCritSrcTotalT 是进程访问临界资源的总时间，用于控制进程对临界资源的访问过程，该值是在进程创建时根据 nTotalTime 随机产生的，总是小于 nTotalTime。

● 已访问临界资源时间

nCritSrcHasExecT 是进程已访问临界资源的时间，用于控制进程访问临界资源的进度，在定时处理中加 1，当等于 nCritSrcTotalT 时表示访问临界资源结束，该域主要在进程同步模块中使用，在其他模块中该值一律为 0。

● 访问临界资源的阶段

iCritSrcVisitPos 是进程对临界资源的访问阶段，可以取值为 CRITSRC_VISIT_NULL(没有访问)，CRITSRC_VISIT_ENTEY(进入区)、CRITSRC_VISIT_CRITICAL(临界区)、CRITSRC_VISIT_EXIT(退出区)、CRITSRC_VISIT_FINISH(完成访问)，该域用于标识和记录进程访问资源阶段，在进程同步模块中使用。

● 内核栈栈顶位置

nKernelStackPos 模拟进程的内核栈指针位置，即栈顶

位置，创建进程时置为 0，模拟进程执行过程中内核栈的变化，该域主要在进程概念演示模块中使用。

- 进程用户名

iUser 是进程的用户名，可以取值为 SYSTEM(系统)和 USER(用户)。本系统中为了能够比较全面地模拟操作系统中的进程，设置了 0 号系统进程和 1 号系统进程。

- 进程类别

iSort 是进程的类别，可以取值为 NORMALTASK(一般进程)、READTASK(读者进程)、WRITETASK(写者进程)，该域主要用于区分进程同步模块中的读写进程，在其他模块中，该值为 NORMALTASK(一般进程)。

- 等待信号量标志

nWaitedSema 用于进程同步模块，用来标识进程访问临界资源过程中，是否执行过 wait(s)原语，因为当进程申请信号量失败被阻塞后，它已经对信号量值进行减一操作了，当再次被唤醒时，是从被阻塞的位置往下执行的，即不会再检查信号量值了，由于本系统是模拟这一过程，需要一个标志来做判断，而进程访问临界资源的过程中可能要涉及多个信号量，所以将 nWaitedSema 定义为 UINT 类型，否则会使信号量值发生错误。

- 链表链接域

pTasksList 和 pStateList 用于链接进程链表，它们都是 list_head 类型的通用双向循环链表。list_head 结构的定义：

```
struct  list_head
{
    struct list_head * pPre;
    struct list_head * pNext;
    PCB * pThisPcb;
};
```

- 模拟内存空间

aAddrSpace 是一个具有三个元素的数组，用来模拟进程的逻辑内存空间，是 RECT 类型的。本系统用三个矩形模拟进程三部分(正文段、用户数据段和系统数据段)在内存中的分布，在界面上显示出来，直观地表示出进程实体。在进程创建演示中，由于要演示资源分配的过程，所以在进程创建时对该数组进程赋值(根据显示区域合理的产生随机值)，而其它模块中，在需要显示进程实体的分布时才给该数组赋值。

3.2 信号量

在进程同步演示模块中，用到了记录型信号量。

(1) 信号量结构体定义

```
struct  SEMAPHORE
{
int nValue; //资源数量
CProcessLink * pBlockLink; //阻塞链表
};
```

(2) 结构体说明

每一个信号量都有一个标识临界资源数量的值，即结构体中的 nValue。在本系统的进程同步模块中，用到的都是互斥信号量，所以 nValue 初值都是 1。当该值小于 0 时，nValue 的绝对值表示阻塞链表中的等待进程数。

因等待临界资源而不能继续执行的进程阻塞相应信号量的阻塞链表中，该链表就是结构体中的 pBlockLink，当进程释放临界资源访问权时，若判断得出还有等待该资源的进程，就从该链表中唤醒第一个进程。

3.3 模块类

模块类是本系统中实现进程管理系统演示的核心类。由于本系统有多个模块，且各个模块具有一些共同特性，所以定义了一个基类 CDlgPage，并定义了继承类 CDlgPage 的各个模块或子模块类，如图 2 所示。每个模块都有一些重要的成员，说明如下：

- m_pCurRunPcb 指向当前将要执行的进程或当前正在执行的进程。
- m_nCurTimerEvent 记录定时器定时标志。
- m_pTasksLink 指向总进程链表。
- m_pSysIdleTask 是 0 号系统进程，m_pSysInitTask 是 1 号系统进程，每个模块中都有这两个模拟的系统进程，这两个进程在第一次演示当前模块时创建，一直存在。
- m_nProNum 是当前模块中的进程数。
- m_pSysInitTask 是 1 号系统进程。
- m_nProNum 是当前模块中的进程数。
- virtual void InitPage()该函数的功能是在用户切换到当前模块时进行一些初始化操作，即控件创建、信息显示、等操作。
- virtual void ClearPage()该函数的功能是对当前演示模块进行相关清除操作，即删除界面控件、清除其它界面显示、清除模块对象与标志记录等操作。

用户在切换演示模块时先调用被切换模块对象的 ClearPage()函数，再调用切换到的模块对象的 InitPage()函数，从而实现演示模块的切换。

4 总结

本系统结合多种控件以及延时、同步刷新界面和突出显示变化效果等实现了图形化的进程管理模拟系统，达到了较好的演示效果。进程管理中各个模块独立实现，能较好地辅助操作系统原理的课堂教学。系统代码结构清晰，功能丰富，方案设计合理，扩展性强，有良好的交互性，有助于学生直观、具体地理解进程管理原理。

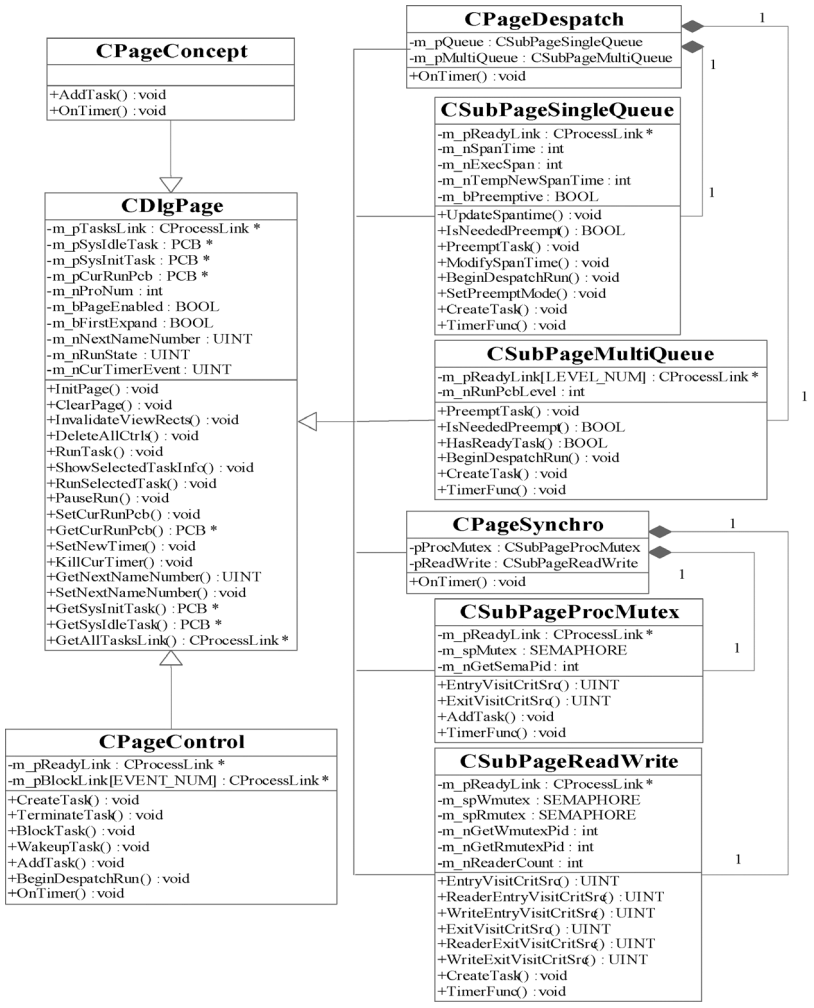

图 2　系统模块类设计图

参考文献：

[1] 汤子瀛, 哲凤评, 汤小丹. 计算机操作系统原理(修订版) [M]. 西安: 西安电子科技大学出版社, 2001: 26-79.

[2] 陈莉君. 深入分析 Linux 内核源代码[M]. 北京: 人民邮电出版社, 2002: 97-138.

[3] DANIEL P. BOVET, MARCO CESATI. 深入理解 Linux 内核[M]. 3 版. 陈莉君, 译. 北京: 中国电力出版社, 2007: 84-134, 258-265.

[4] Andrew S. Tanenbaum. 操作系统设计与实现[M]. 北京: 电子工业出版社, 1998: 19-49.

(上接 140 页)

感觉入手并不难。然后可以渐渐增加动手、创新实验的比重。当然，要保证绝大多数实验是创新型实验，否则无法真正锻炼到学生的动手能力和创新意识。

4　结束语

WRK 让搭建基于 Windows 的操作系统实验平台成为了可能，但是真正要让 WRK 发挥其在操作系统实验教学中的作用，还需要教师们做大量的工作。在同济大学软件学院的教学中，我们建立了一个基于 WRK 的实验平台，并初步在课程中应用。但这仅仅是迈出了第一步，接下来还需要在实际教学的过程中听取各方面的意见与建议，进一步对该平台进行完善。

参考文献：

[1] 冯红伟, 王鹏. 操作系统教学与实验设计研究[J]. 实验室研究与探索, 2007(12): 251-253.

[2] 潘东静. 操作系统实验教学研究[J]. 现代计算机, 2008(1): 70-71.

[3] 邓胜兰, 宁洪. 操作系统实践教学的探索[J]. 计算机教育, 2007(10): 8-9.

[4] 王国华. 《操作系统》实验课程的设置与实践[J]. 山西财经大学学报: 高等教育版, 2006(S1): 108.

[5] 赵福来. 国内高校操作系统课程实验教学实施情况评述[J]. 中国科技信息, 2005(12): 90.

[6] 彭敏, 何炎祥. 基于 WRK 的 Windows 操作系统原理实验教学探索[J]. 计算机教育, 2008(20): 38-40.