《操作系统综合实验》

实验指导书

华北电力大学计算机系

操作系统课程组

2020-4-20

一、 实验内容

采用某种程序设计语言，设计与开发仿真操作系统各模块功能。具体内容包括：

（1）处理机调度

（2）银行家算法

（3）主存储器空间的分配与回收

（4）独占设备的分配和回收

（5）文件管理

实验内容详细要求见实验指导书。学生对个模块功能进行设计开发，最后进行综合集成与设计，开发出一个小型仿真操作系统。

二、 考核方式及成绩评定方案

1、考核方式：

考核方式采用视频验收+实验报告评阅进行

（1）实验验收：要求每个学生对每个实验的实验过程和实验成果进行讲解，并将所有实验讲解通过录屏软件录制成一个 10 分钟以内的视频,要求录制过程中打开摄像头，视频中有语音讲解、实验过程和运行结果展示。

（2）实验报告：要求每个学生完成一份实验报告，该报告包括所有实验的相关内容（报告格式在课堂派课堂下载），完成后提交课堂派课堂，该报告要求独立完成。

2、成绩评定方案：

实验成绩=验收成绩+实验报告成绩，任课教师依据学生录制的实验视频、实验报告质量和提交时间给出成绩，成绩分为：优、良、中、及格、不及格；

一、 处理机调度

1．实验目的

在多道程序系统中,调度的实质是一种资源分配,处理机调度是对处理机资源进行分配。在多道批处理系统中，一个作业从提交到获得处理机执行，直至作业运行完毕，可能经历作业调度和进程调度。本实验模拟在单处理器情况下的处理器调度，帮助学生加深了解处理器调度的工作。

2．实验内容与要求

（1）模拟调度程序可任选两种调度之一实现。

（3）程序执行中应能在屏幕上显示出各作业或进程的状态变化，以便于观察调度的整个过程。

3．实验说明

第一题：设计一个按高响应比优先算法实现作业调度的程序。

从输入井中选择作业读入内存，使其获得处理器，得到运行的机会，即为作业调度。输入井中的作业用“作业控制块”（JCB）标识，为了进行作业调度，将作业控制块组成一个队列，这个队列称为后备队列。

模拟实验中没有实际作业，作业控制块中的信息内容只使用了模拟实验中需要的数据。作业控制块中包括作业名、作业大小、所需打印机台数、所需磁带机数量、作业估计执行时间、作业等待时间、指向下一个作业控制块的指针等内容。将作业控制块组成一个队列，实验中采用动态链表的方式模拟作业的后备队列。作业控制块采用结构型数据模拟。

模拟实验中，主存采用可移动的可变分区管理方法，即只要主存空闲区总和比作业大就可以满足作业对主存的需求。对打印机和磁带机这两种独占设备采用静态分配法，即作业执行前必须获得所需资源，并且执行完才归还。

实验中作业的调度采用响应比高者优先算法。响应比为作业的等待时间和作业估计执行时间之比。首先计算出输入井中满足条件的作业的响应比，从中选择响应比最高的一个作业装入主存储器，分配资源。由于是模拟实验，可将作业控制块出队装入主存储器的工作用输出作业名模拟，同时修改系统的资源数量。

模拟实验时，可以首先假设系统的资源情况。假设系统资源只有主存 64MB、磁带机 4 台、打印机 2 台，然后手工输入某个时刻输入井中的各个作业情况，最后进行作业调度，并将结果输出。

批处理系统中的作业调度模拟程序主要由创建作业队列的程序段（在主函数中）和作业调度函数组成。

第二题：设计一个按时间片轮转法实现进程调度的程序。

(1) 假定系统有五个进程，每一个进程用一个进程控制块 PCB 来代表。进程控制块的

格式为：

进程名
指针
要求运行时间
已运行时间
状态

其中，进程名——作为进程的标识，假设五个进程的进程名分别为 Q_1，Q_2，Q_3，Q_4，Q_5。

指针——进程按顺序排成循环队列，用指针指出下一个进程的进程控制块的首地址，最后一个进程的指针指出第一个进程的进程控制块首地址。

要求运行时间——假设进程需要运行的单位时间数。

已运行时间——假设进程已经运行的单位时间数，初始值为“0”。

状态——有两种状态，“就绪”和“结束”，初始状态都为“就绪”，用“R”表示。当一个进程运行结束后，它的状态为“结束”，用“E”表示。

(2) 每次运行所设计的处理器调度程序前，为每个进程任意确定它的“要求运行时间”。

(3) 把五个进程按顺序排成循环队列，用指针指出队列连接情况。另用一标志单元记录轮到运行的进程。例如，当前轮到 P_2 执行，则有：

标志单元

K_2

	K_1	K_2	K_3	K_4	K_5
	Q_1	Q_2	Q_3	Q_4	Q_5
	K_2	K_3	K_4	K_5	K_1
	2	3	1	2	4
	1	0	0	0	0
	R	R	R	R	R
	PCB1	PCB2	PCB3	PCB4	PCB5

(4) 处理器调度总是选择标志单元指示的进程运行。由于本实验是模拟处理器调度的功能，所以，对被选中的进程并不实际的启动运行，而是执行：

已运行时间+1

来模拟进程的一次运行，表示进程已经运行过一个单位的时间。

请同学注意：在实际的系统中，当一个进程被选中运行时，必须置上该进程可以运行的时间片值，以及恢复进程的现场，让它占有处理器运行，直到出现等待事件或运行满一个时间片。在这时省去了这些工作，仅用“已运行时间+1”来表示进程已经运行满一个时间片。

(5) 进程运行一次后，应把该进程的进程控制块中的指针值送到标志单元，以指示下一个轮到运行的进程。同时，应判断该进程的要求运行时间与已运行时间，若该进程的要求运行时间≠已运行时间，则表示它尚未执行结束，应待到下一轮时再运行。若该进程的要求运行时间=已运行时间，则表示它已经执行结束，应指导它的状态修改成“结束”（E）且退出队列。此时，应把该进程的进程控制块中的指针值送到前面一个进程的指针位置。

(6) 若“就绪”状态的进程队列不为空，则重复上面的（4）和（5）的步骤，直到所有的进程都成为“结束”状态。

(7) 在所设计的程序中应有显示或打印语句，能显示或打印每次选中进程的进程名以及运行一次后进程队列的变化。

(8) 为五个进程任意确定一组“要求运行时间”，启动所设计的处理器调度程序，显示或打印逐次被选中的进程名以及进程控制块的动态变化过程。

4. 实验报告

(1) 实验题目。

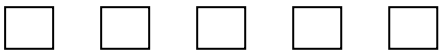
(2) 程序中使用的数据结构及符号说明。

(3) 流程图。

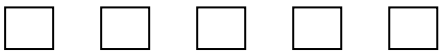
(4) 打印一份源程序并附上注释。

(5) 打印程序运行时的初值和运行结果。要求如下：

i 进程控制块的初始状态。



ii 选中运行的进程名以及选中进程运行后的各进程控制块状态。



对于 ii 要求每选中一个进程运行后都要打印。

二、 银行家算法

1. 实验目的和要求

银行家算法是避免死锁的一种重要方法，要求编写和调试一个简单的银行家算法程序。加深了解有关资源申请、避免死锁等概念，并体会和了解死锁和避免死锁的具体实施方法。

2. 实验内容

(1) 设计进程对各类资源最大申请表示及初值确定。

(2) 设定系统提供资源初始状况。

(3) 设定每次某个进程对各类资源的申请表示。

(4) 编制程序，依据银行家算法，决定其申请是否得到满足。

3. 实验说明

(1)数据结构

假设有 M 个进程 N 类资源，则有如下数据结构：

MAX[M*N] M 个进程对 N 类资源的最大需求量

AVAILABLE[N] 系统可用资源数

ALLOCATION[M*N] M 个进程已经得到 N 类资源的资源量

NEED[M*N] M 个进程还需要 N 类资源的资源量

(2)算法流程

设进程 I 提出请求 Request[N]，则银行家算法按如下规则进行判断。

①如果 Request[N]<=NEED[I，N]，则转(2)；否则，出错。

②如果 Request[N]<=AVAILABLE，则转(3)；否则，出错。

③系统试探分配资源，修改相关数据：

AVAILABLE=AVAILABLE-REQUEST

ALLOCATION=ALLOCATION+REQUEST

NEED=NEED-REQUEST

④系统执行安全性检查，如安全，则分配成立；否则试探险性分配作废，系统恢复原状，进程等待。

(3)安全性检查

①设置两个工作向量 WORK=AVAILABLE；FINISH[M]=FALSE

②从进程集合中找到一个满足下述条件的进程，

FINISH[i]=FALSE

NEED<=WORK

如找到，执行③；否则，执行④

③设进程获得资源，可顺利执行，直至完成，从而释放资源。

WORK=WORK+ALLOCATION

FINISH=TRUE

GO TO ②

④如所有的进程 Finish[M]=true，则表示安全；否则系统不安全。

4. 参考程序

```
#include "string.h"
#include "iostream.h"
#define M 5 //总进程数
#define N 3 //总资源数
#define FALSE 0
#define TRUE 1

//M 个进程对 N 类资源最大资源需求量
int MAX[M][N]={{7,5,3},{3,2,2},{9,0,2},{2,2,2},{4,3,3}};
//系统可用资源数
int AVAILABLE[N]={10,5,7};
//M 个进程已经得到 N 类资源的资源量
int ALLOCATION[M][N]={{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0}};
//M 个进程还需要 N 类资源的资源量
int NEED[M][N]={{7,5,3},{3,2,2},{9,0,2},{2,2,2},{4,3,3}};
int Request[N]={0,0,0};
```

三、 主存储器空间的分配和回收

1．实验目的

一个好的计算机系统不仅要有一个足够容量的、存取速度高的、稳定可靠的主存储器，而且要能合理地分配和使用这些存储空间。当用户提出申请存储器空间时，存储管理必须根据申请者的要求，按一定的策略分析主存空间的使用情况，找出足够的空闲区域分配给申请者。当作业撤离或主动归还主存资源时，则存储管理要收回作业占用的主存空间或归还部分主存空间。主存的分配和回收的实现与主存储器的管理方式有关的，通过本实验帮助学生理解在不同的存储管理方式下应怎样实现主存空间的分配和回收。

2．实验内容

本实验模拟在传统存储器管理方式和虚拟存储器管理方式下的主存分配和回收。

3．实验说明

第一题：在可变分区管理方式下采用最先适应算法实现主存分配和实现主存回收。

(1) 可变分区方式是按作业需要的主存空间大小来分割分区的。当要装入一个作业时，根据作业需要的主存量查看是否有足够的空闲空间，若有，则按需要量分割一个分区分配给该作业；若无，则作业不能装入。随着作业的装入、撤离，主存空间被分成许多个分区，有的分区被作业占用，而有的分区是空闲的。例如：

0	
5k	操作系统
10k	作业 1
14k	作业 3
26k	空闲区
32k	作业 2
128k	空闲区

为了说明哪些区是空闲的，可以用来装入新作业，必须要有一张空闲区说明表，格式如下：

	起　　址	长　　度	状　　态
第一栏	14 K	12 K	未 分 配
第二栏	32 K	96 K	未 分 配
⋮			空 表 目
⋮			空 表 目
			⋮ ⋮

其中，起址——指出一个空闲区的主存起始地址。

长度——指出从起始地址开始的一个连续空闲的长度。

状态——有两种状态，一种是“未分配”状态，指出对应的由起址指出的某个长度的区域是空闲区；另一种是“空表目”状态，表示表中对应的登记项目是空白（无效），可用来登记新的空闲区（例如，作业撤离后，它所占的区域就成了空闲区，应找一个“空表目”栏登记归还区的起址和长度且修改状态）。由于分区的个数不定，所以空闲区说明表中应有适量的状态为“空表目”的登记栏目，否则造成表格“溢出”无法登记。

上述的这张说明表的登记情况是按提示（1）中的例所装入的三个作业占用的主存区域后填写的。

(2) 当有一个新作业要求装入主存时，必须查空闲区说明表，从中找出一个足够大的空闲区。有时找到的空闲区可能大于作业需要量，这时应把原来的空闲区变成两部分：一部分分给作业占用；另一部分又成为一个较小的空闲区。为了尽量减少由于分割造成的空闲区，而尽量保存高地址部分有较大的连续空闲区域，以利于大型作业的装入。为此，在空闲区说明表中，把每个空闲区按其地址顺序登记，即每个后继的空闲区其起始地址总是比前者大。为了方便查找还可使表格“紧缩”，总是让“空表目”栏集中在表格的后部。

(3) 采用最先适应算法（顺序分配算法）分配主存空间。

按照作业的需要量，查空闲区说明表，顺序查看登记栏，找到第一个能满足要求的空闲区。当空闲区大于需要量时，一部分用来装入作业，另一部分仍为空闲区登记在空闲区说明表中。

由于本实验是模拟主存的分配，所以把主存区分配给作业后并不实际启动装入程序装入作业，而用输出“分配情况”来代替。最先适应分配算法如图 4-1。

(4) 当一个作业执行结束撤离时，作业所占的区域应该归还，归还的区域如果与其它空闲区相邻，则应合成一个较大的空闲区，登记在空闲区说明表中。例如，在提示（1）中列举的情况下，如果作业 2 撤离，归还所占主存区域时，应与上、下相邻的空闲区一起合成一个大的空闲区登记在空闲区说明表中。归还主存时的回收算法如图 4-2。

(5) 请按最先适应算法设计主存分配和回收的程序。然后按（1）中假设主存中已装入三个作业，且形成两个空闲区，确定空闲区说明表的初值。现有一个需要主存量为 6K 的作业 4 申请装入主存；然后作业 3 撤离；再作业 2 撤离。请你为它们进行主存分配和回收，把空闲区说明表的初值以及每次分配或回收后的变化显示出来或打印出来。

第二题:设计请求页式存储管理中的页面置换算法,加深了解虚拟存储技术的特点,掌握各种页面置换的算法。

(1) 通过随机数产生一个指令序列,共 320 条指令。指令的地址按下述原则生成:

① 50%的指令是顺序执行的。

② 25%的指令均匀分布在低地址部分。

③ 25%的指令均匀分布在高地址部分。

具体实施的方法是:

① 在[0,319]的指令地址之间随机产生一个起点 m;

② 顺序执行一条指令,即执行 m+1 处的执行;

③ 在地址[0,m+1]中随机选取一条指令执行,该指令的地址为 m1;

④ 顺序执行一条指令,即执行 m1+1 处的执行;

⑤ 在地址[m1+2,319]中随机选取一条指令执行;

⑥ 重复上述步骤直到执行了 320 条指令为止。

(2) 将指令序列改变为页地址流,并假设:

① 页面尺寸为 1K。

② 用户的内存容量为 4 页到 32 页。

③ 用户的虚存容量为 32K。

在用户虚存中,按每 K 存放 10 条指令来排列虚存地址,即 320 条指令在虚存放方式为:

第 0 条~第 9 条为 0 号页(对应的虚存地址为[0,9])。

第 10 条~第 19 条为 1 号页(对应的虚存地址为[10,19])。

… … … …

第 310 条~第 319 条为 31 号页(对应的虚存地址为[310,319])。

按上述方式,用户指令可组成 32 页。

(3) 计算并输出在不同内存容量下的命中率(4 种置换算法选择一种)。

① 先进先出算法(FIFO)。

② 最近最少使用算法(LRU)。

③ 最佳淘汰算法。

④ 最少访问页面算法(LFR)。

其中,命中率为:

命中率=1-缺页次数/页地址流长度

本实验中,页的地址流长度为 320,页面失效次数为每次访问响应指令时,该指令对应的页面不在内存中的次数。

(4)随机数的产生方法

在 VC 中设计到随机数有两个函数 srand() and rand()。srand() 的作用是是一个种子，提供每次获得随机数的基数而已，rand()根据种子而产生随机数。

注意

1：srand() 里的值必须是动态变化的，否则得到的随机数就是一个固定数

2：如果我们想得到一个 0-60 的随机数那么可以写成

```
int i;
srand(GetTickCount());
i=rand()%60;
```

4．实验报告

(1) 实验题目。

(2) 程序中使用的数据结构及符号说明。

(3) 流程图。

(4) 打印一份源程序并附上注释。

(5) 打印程序运行时的初值和运行结果，要求如下：

第一题：打印空闲区说明表的初始状态，作业 4 的申请量以及为作业 4 分配后的空闲区说明表状态；再依次打印作业 3 和作业 2 的归还量以及回收作业 3，作业 2 所占主存后的空闲区说明表。

第二题：打印模拟的页面访问序列，内存物理块的占有情况，页面置换次数和命中率。

四、 独占设备的分配和回收

1．实验目的

加深对设备管理的理解。

2．实验内容

本实验模拟模拟设备的分配与回收。

3．提示与讲解

此实验模拟满足设备独占性的独占设备的分配和回收。对于独占设备，通常采用静态分配方式，即在一个作业执行前，将作业要使用的设备分配给作业，在作业执行期间这个设备均归该作业占用，直到作业执行结束才归还。

操作系统设置“设备分配表”，用来记录计算机系统所配置的独占设备类型、台数以及分配情况等。为了实现设备分配的独立性，一般设备分配表可以由“设备类表”和“设备表”两部分组成。

设备类表记录系统中的各类设备，每类设备占用一个登记项，登记该类设备的总台数、当前有几台设备空闲以及该类设备在设备表中的起始地址。每一台设备在“设备表”中占用一个登记项，同类的若干台设备连续登记在设备表中。设备表中登记每一台设备的绝对号、设备的好坏情况、设备是否分配、设备被哪一个作业占用和设备相对号。模拟实验中，设备类表和设备表均采用结构型的数据类型表示，在设备表中，用“1”表示设备好，用“0”表示设备坏；用“1”表示设备已分配，用“0”表示设备空闲。

在作业申请某类设备时，系统先查“设备类表”，如果该台设备的现存台数可以满足申请要求，则从“设备类表”中得到该类设备的设备表起始地址，然后找到“设备表”中该类设备的起始地址，依次查询该类设备的每一个登记项，找出“好的且未分配”的设备分配给该作业。分配后要修改设备类表中的现存台数，把分配给该作业的设备状态改为“已分配”，且填上占用该设备的作业的作业名和程序中定义的相对号，最后，把设备的绝对号与相对号的对应关系通知用户。

当作业运行完回收设备时，首先要查看设备表，比较每一项，找到占用作业名与运行完作业的作业名相同的一栏，将这一栏的“已/未分配”置为“未分配”（即置为“0”）。然后将设备表中对应设备类的“现存台数”增 1。

模拟实验的程序包括建立设备类表和设备表的程序段（在主函数中）、分配设备和回收设备的函数。分配设备要求输入作业名、设备类名和相对号；回收设备要求输入作业名和设备类名。

4． 参考程序

系统中主要数据结构和变量的意义见注释。

```
#define  false  0
#define  true  1
#define  n  4
#define  m  10
struct
{
char  type[10];  /*设备类名*/
int  count;  /*拥有设备台数*/
int  remain;  /*现存的可用设备台数*/
int  address;  /*该类设备在设备表中的起始地址*/
```

```
}equiptype[n];  /*设备类表定义，假定系统有 n 个设备类型*/
struct
{
int  number;  /*设备绝对号*/
int  status;  /*设备好坏状态*/
int  remain;  /*设备是否已分配*/
char  jobname[4];/*占有设备的作业名*/
int  lnumber;  /*设备相对号*/
}equipment[m];  /*设备表定义，假定系统有 m 个设备*/ allocate(J,type,mm)
char  *J,*type;
```

五、 文件系统

1. 实验目的

本实验模拟文件管理的工作过程，从而对各种文件操作命令的实质内容和执行过程有比较深入的了解。模拟一个文件系统，包括目录文件，普通文件，并实现对它们的一些基本操作。

2. 实验内容

假定每个目录文件最多只能占用一个块；一个目录项包括文件名（下一级目录名），文件类型，文件长度，指向文件内容（下一级目录）的指针内容。普通文件可以只用目录项（FCB）代表。需要实现一个命令行操作界面，包含如下命令：

（1） 改变目录

格式：CD〈目录名〉

功能：工作目录转移到指定的目录下，只要求完成改变到当前目录的某一个子目录下的功能，不要求实现相对目录以及绝对目录。

（2）创建文件

格式：CREATE〈文件名〉〈文件长度〉

功能：创立一个指定名字的新文件，即在目录中增加一项，不考虑文件内容，但必须能输入文件长度。

（3） 删除文件

格式：DEL〈希望删除的文件名〉

功能：删除指定的文件

（4）显示目录

格式：LSALL

功能：显示全部目录以及文件，输出时要求先输出接近根的目录，再输出子目录。图示如图。

（5）创建目录

格式：MD〈目录名〉

功能：在当前路径下创建指定的目录

（6）删除目录

格式：RD〈目录名〉

功能：功能：删除当前目录下的指定目录，如果该目录为空，则可删除，否则应提示是否作删除，删除*作将该目录下的全部文件和子目录都删除。对于上述功能要求，完成 1-4 为及格，完成 1-5 为良，完成 1-6 为优。

3. 实验要求

（1）对于重名（创建时），文件不存在（删除时），目录不存在（改变目录时）等错误 * 作情况，程序应该作出相应处理并给出错误信息，但是程序不得因此而退出。

（2） 界面友好，程序强壮。

（3）界面的提示符为#，提示的命令以及调试的方法应和前面的要求一致。不要自己设计命令或者附加不要求的功能。

4. 参考程序

```
#include "stdio.h"
#include "string.h"
#include "malloc.h"
struct NomalFile
```

```
{
        char name[50]; /*file name*/
        int type; /*0-directory file,1-common file*/
        int size; /*file size*/
};

struct DirectoryNode
{
    int tag; /*0-dump node,1-real node*/
    char name[50];
    int type; /*0-directory file 1-comman file*/
    union
    {
        struct NomalFile *p_nomFile; /*point to comman file */
        struct DirectoryFile *p_dirFile; /*point to dirctory file*/
        }p_File; /*look on initializing code*/
};

struct DirectoryFile
{/*directory file composed by a node array*/
/*nodes*/
          struct DirectoryNode nodesArray[10];
};
/*initializing file system*/
```