第二章　进程的描述与控制

1.前趋图

(1)根据前趋关系集合或给定语句画出前趋图

(2)无边权，点权为该语句/进程执行时间

2.进程

(1) 进程是进程实体的运行过程，是系统进行资源分配和调度的一个独立单位。

(2) 进程三种基本状态：就绪、执行、阻塞。

(3) 进程状态切换过程：

①调用对应原语

②改变进程状态

③处理 PCB（申请、移入、移出）

④处理资源（申请、释放回收）

⑤进行调度（插入队列、转调度程序、停止执行）

(4) 进程的终止过程

① 系统调用终止原语

② 终止进程执行，置调度标志为真

③ 终止其子孙进程

④ 回收全部资源并归还给父进程或系统

⑤ 移除被终止进程的 PCB

3.信号量

(1)记录型信号量

S.value 的绝对值表示在该信号量链表中已阻塞的进程数目

使用前需要设置初值，互斥信号量 mutex 设初值为 1。

定义：

```
typedef struct {
    int value;
    struct PCB *list;
} semaphore;

wait(semaphore S) {
    S.value--;
    if(S.value<=0)  block(S.list);    //资源不足，自我阻塞
}
signal(semaphore S) {
    S.value++;
    if(s.value<=0)  wakeup(S.list);   //归还资源，唤醒队中第一个进程
}
```

(2) AND 型信号量

改为 Swait 和 Ssignal，对多个信号量同时操作。

(3) 使用信号量实现前趋关系

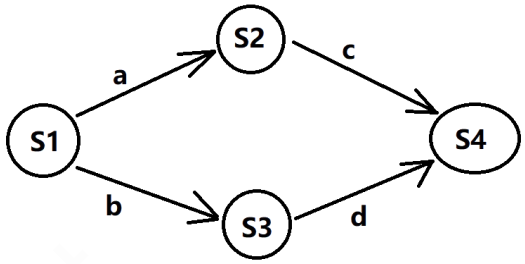
入边为 wait / P，出边为 signal / V，信号量初值为 0 。

答题规范：

```
semaphore a,b,c,d;
a.value = b.value = c.value = d.value = 0;

p1() { S1; signal(a); signal(b); }
p2() { wait(a); S2; signal(c); }
p3() { wait(b); S3; signal(d); }
p4() { wait(c); wait(d); S4; }

main() {
    cobegin
        p1(); p2(); p3(); p4();
    coend
}
```



(4) 进程同步经典问题 P65

① 需要先定义全局信号量 semaphore 并赋初值

② wait 就是-1，signal 就是+1

③ 进程函数中要 while(1) 循环

④ 主函数中要 cobegin-coend，中间填写所有进程函数

⑤ 生产者-消费者问题：先判空满，再判互斥

⑥ 读者-写者问题：设信号量 rmutex, wmutex；整型 readcount

第三章　处理机调度与死锁

1. 处理机调度层次

(1) 高级调度（作业调度）：将作业从外存调入内存

(2) 低级调度（进程调度）：决定哪些进程获得处理机

2. 作业调度算法

(1) 先来先服务 FCFS

(2) 短作业优先 SJF

(3) 优先级调度算法 PSA

(4) 高响应比优先调度算法 HRRN

① 优先权$R_{\mathrm{p}}=1+\dfrac{\text{等待时间}}{\text{要求服务时间}}$

3. 进程调度算法

(1) 时间片轮转调度算法 RR

① 周转时间 = 完成时间 – 到达时间

② 带权周转时间$=\dfrac{\text{周转时间}}{\text{服务时间}}$

③ 时间片使用顺序依据 FCFS 策略

④ 进程完成时，调度下一个进程运行，启用新的时间片

(2) 优先级调度算法：抢占式 / 非抢占式

(3) 多队列调度算法：设置多个就绪队列，队列间和队列内

(4) 多级反馈队列调度算法：时间片递增、抢占、未处理完则放入

下一队列、最后一个用 RR、其他用 FCFS

4. 死锁概念

(1) 如果一组进程中的每一个进程都在等待仅由该组进程中的其他进程才能引发的事件，那么该组进程是死锁的。

5. 死锁原因

(1) 竞争不可抢占性资源

(2) 竞争可消耗资源

(3) 进程推进顺序不当

6. 死锁必要条件

(1) 互斥条件：系统中存在排它性资源

(2) 请求和保持条件：进程保持了资源，但又提出了新的请求

(3) 不可抢占条件：资源只能由进程自己释放

(4) 循环等待条件：存在进程-资源的循环等待链

7. 处理死锁的方法

(1) 预防、避免、检测、解除

(2) 防范程度逐渐减弱，但资源利用率提高、阻塞减少

8. 预防死锁（破坏必要条件）

(1) 破坏“请求和保持”

① 进程在请求资源时，不能持有不可抢占资源

② 协定：进程在开始运行之前，必须一次性申请全部资源

(2) 破坏“不可抢占”

① 当一个已经保持了某些不可抢占资源的进程，提出请求但没有得到满足时，必须释放已经保持的所有资源。

(3) 破坏“循环等待”

① 对系统所有资源类型进行线性排序，赋予不同的序号

② 规定每个进程必须按序号递增的顺序请求资源

9. 避免死锁（保证系统时刻处于安全状态）

(1) 安全状态：存在安全序列

(2) 安全序列：系统按照此进程序列的顺序分配资源，就能使每个进程都顺利完成

(3) 银行家算法：试探分配，执行安全检查，决定是否作废试探

(4) 安全性算法（模拟逐个收回资源的过程来找到安全序列）

① Max 最大需求矩阵

② Allocation 已分配矩阵

③ Need（仍）需求矩阵

④ Available 可利用资源向量

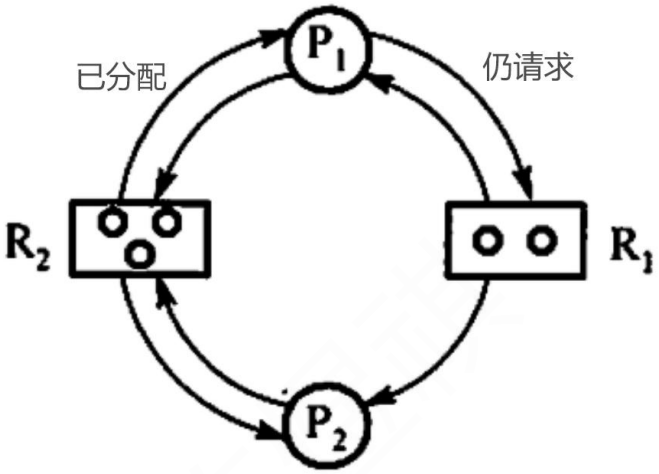
⑤ Work 工作向量（初始化为 Available）

⑥ 找安全序列：Work≥Need → Work+=Allocation

10. 检测死锁

(1) 画资源分配图

(2) 死锁定理：资源分配图不可完全简化（消去所有的边）当且仅当该状态为死锁



第四章　存储器管理

1. 分页地址变换

(1) 页面大小 = 物理块大小

(2) 页表项数 = 逻辑地址空间 ÷ 页面大小

(3) 最大块号 = 物理内存大小 ÷ 页面大小

(4) 单个页表项大小 = $\log_2$ 最大块号 (bit)

(5) 逻辑地址 := {页号,页内偏移量}

(6) 页内偏移量位数→页面大小，页号位数→页表项数

(7) 十六进制地址转换：二进制转换、查页表、拼接、转换回 H

(8) 十进制地址转换

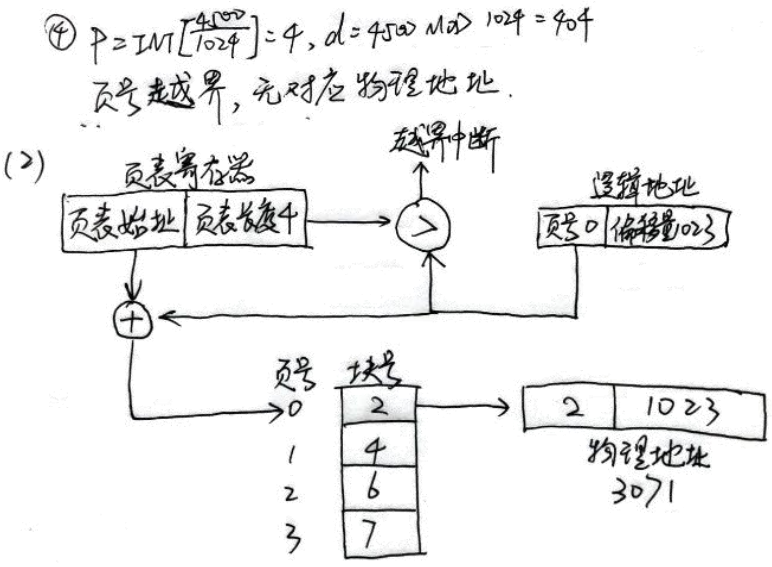
① 页号 P = INT[地址 / 页面大小]

② 页内偏移量 d = 地址 MOD 页面大小

③ 物理地址 = 块号 × 页面大小 + 页内偏移量

(9) 越界中断、缺页中断

(10) 地址转换过程图



2. 分段地址变换

(1) 逻辑地址 := {段号, 段内偏移量}

(2) 段表 := {段号, 段长, 基址}

(3) 查表，根据段长判断越界，物理地址 = 基址 + 段内偏移量

3. 快表与有效存取时间

(1) 常规分页 EAT = 2t

(2) 引入快表 EAT = $\alpha\times(\lambda+t)+(1-\alpha)\times(\lambda+t+t)$

① α 快表命中率

② λ 检索快表所需时间

③ t 内存访问所需时间

4. 访存次数

(1) 页式、段式：2 次（查表一次，取数据一次）

(2) 段页式：3 次（段表页表各一次，取数据一次）

(3) N 级页表：N+1 次

第五章　虚拟存储器

1. 页面置换算法

(1) 最佳置换算法 OPT：换出最长未来时间内不再被访问的页

(2) **先进先出 FIFO**：淘汰最先进入内存的页

(3) **最近最久未使用 LRU**：淘汰最近访问时间最久远的页

(4) 最少使用 LFU：淘汰最近一段时间内使用次数最少的页

(5) CLOCK 置换算法

① 某页被访问时，访问位 A 置 1

② 循环检查，访问位为 0 则换出该页，为 1 则置 0

(6) 改进型 CLOCK 置换算法

① 多设一个修改位 M 来确定最优淘汰页

② 找 00、 找 01,访问位 A 置 0、 找 00、 找 01,必找到

(7) 页面缓冲算法 PBA：设置空闲页面链表与修改页面链表

2. 缺页率计算

(1) 预装页不算缺页

(2) 缺页率 = 缺页次数 ÷ 访问次数

3. 访问内存的有效时间

(1) 未缺页、命中快表：$EAT = \lambda + t$

(2) 未缺页、未命中：$EAT = 2(\lambda + t)$

(3) 缺页：$EAT = \varepsilon + 2(\lambda + t)$

(4) 变量名

① λ 访问快表时间

② t 访存时间

③ ε 缺页中断处理时间

④ a 命中率

⑤ f 缺页率

(5) 若不考虑命中率，只考虑缺页率，则λ=0 且 a=0，则

$$EAT = f \times (\varepsilon + 2t) + (1-f) \times 2t$$

第六章　输入输出系统

1. 对 I/O 设备的控制方式

(1) 推动 I/O 控制方式发展的主要因素

① 尽量减少 CPU 对 I/O 控制的干预，把 CPU 从 I/O 控制中释放出来

② 缓和 CPU 的高速性和 I/O 设备的低速性不匹配的矛盾

③ 提高 CPU 和 I/O 设备操作的并行程度，提高系统资源利用率和吞吐量

(2) 使用轮询的可编程 I/O 方式

① CPU 不断地循环测试设备的忙闲标志

② CPU 的绝大部分时间都用于等待设备完成 I/O 的循环测试

(3) 使用中断的可编程 I/O 方式

① CPU 向设备发出 I/O 命令后，便可以去做其他事；设备操作完成后发出中断告知 CPU。

② 可以使 CPU 和 I/O 设备并行工作，仅当输入完一个数据时，才需 CPU 花费极短的时间进行中断处理。

③ 以字(节)为单位进行 I/O，CPU 以字(节)为单位进行干预。

(4) 直接存储器访问方式 DMA

① 传输的基本单位是数据块

② 数据直接从设备送入内存

③ 仅在传送数据块的开始和结束时，才需要 CPU 干预，数据传送是在 DMA 控制器的控制下完成的

④ CPU 需要访问外存时便发送命令给 DMA 控制器：命令送入 CR，内存起始地址送入 MAR、本次要传送的字节数送入 DC、磁盘源地址送入 DMA 控制器的 I/O 控制逻辑。

⑤ CPU 启动 DMA 控制器进行数据传送，然后可以转向处理其他任务。

⑥ 每读入一个字(节)到 DR，DMA 控制器就挪用一个存储器周期传送数据字至内存。

⑦ 内存地址 MAR+1、数据计数器 DC-1。

⑧ 当 DC 为 0 时，表示传送完毕，向 CPU 发出中断请求。

(5) I/O 通道控制方式

① 进一步减少 CPU 的干预，以一个数据块为单位的干预，减少为以一组数据块为单位的干预。实现 CPU、通道和 I/O 设备三者的并行操作。

② 通道程序结束位 P、记录结束位 R（字符写成一个记录）。

2. SPOOLing 技术

(1) 假脱机技术：在多道程序的环境下，利用多道程序中的一两道程序（输入程序和输出程序）来模拟外围控制机。从而在联机的情况下实现脱机输入输出的功能。将一台独占的物理设备虚拟为多台逻辑设备，从而使该设备可被多个进程共享。

(2) 输入井和输出井：模拟脱机输入输出时的磁盘，用于收容 I/O 设备输入的或者用户程序输出的数据。以井文件形式管理。

(3) 输入缓冲区和输出缓冲区：在内存中开辟的缓冲区，用于缓和 CPU 和磁盘之间速度不匹配的矛盾。

(4) 输入进程和输出进程：模拟外围控制机。

① 输入设备→输入缓冲区→输入井→内存

② 输出设备←输出缓冲区←输出井←内存

(5) 井管理程序：用于控制作业与磁盘井之间的信息交换，由操作系统调用。

(6) 特点

① 提高了 I/O 的速度

② 将独占设备改造为共享设备

③ 实现了虚拟设备功能

3. 磁盘调度算法

(1) 先来先服务 FCFS

(2) 最短寻道时间优先 SSTF

(3) 扫描/电梯调度算法 SCAN（避免饥饿现象）

(4) 循环扫描算法 CSCAN（规定磁头单向移动）

(5) NstepSCAN：N 个队列，队列间 FCFS，队列内 SCAN

(6) FSCAN：两个队列，旧队列和新队列（避免磁臂黏着）

第七章　文件管理

1. 文件目录

(1) 实现按名存取

(2) 提高对目录的检索速度

(3) 实现文件共享

(4) 允许文件重名

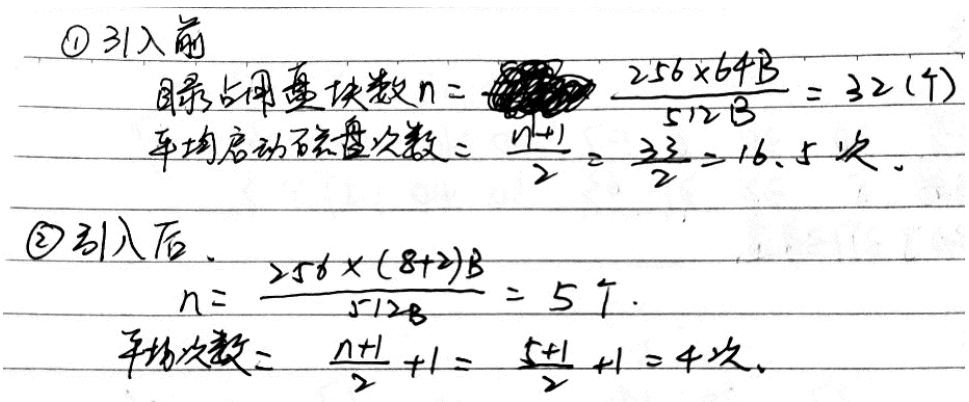
(5) 文件的逻辑结构：顺序文件、索引文件、顺序索引文件

2. 索引结点

(1) 文件控制块 FCB：用于描述和控制文件的数据结构，与文件一一对应。

(2) 索引结点引入原因：文件很多时，目录项占用大量磁盘块，检索效率低。

⑶ 索引结点思想原理：采用把 FCB 中的文件名和文件描述信息分开存储的办法，将文件描述信息单独形成一个称为索引结点的数据结构，目录项仅由文件名和指向该文件所对应的索引结点的指针构成。



3. 文件共享

(1) 文件目录：多级树形目录结构（画图）

(2) 基于有向无循环图和索引结点实现

① 允许每一个文件有多个父目录

② 索引结点解决“文件新增部分不能被共享”问题

③ 文件主不能直接删除文件，只能取消链接并减少 count，否则会导致其他父目录指向该文件的指针悬空。

(3) 利用符号链接实现

① 仅有一个父目录作为属主父目录（实线），其他通过符号链接方式与之相链接的为链接父目录（虚线）。

② 属主结构仍然是简单树

③ 创建 LINK 类型文件，包含被链接文件的路径名，放入链接父目录中

④ 只有文件主才拥有指向索引结点的指针，共享文件的其他用户只有该文件的路径名，避免删除后指针悬空

第八章　磁盘存储器的管理

1. 文件物理结构 / 外存组织方式

(1) 顺序式文件结构 / 连续组织方式

① 为每个文件分配一片连续的磁盘空间

② 支持随机访问、容易产生磁盘碎片

(2) 链接式文件结构 / 链接组织方式

① 为文件分配离散的磁盘空间，通过链接指针将一个文件的所有盘块链接在一起

② 只能顺序访问，但没有碎片、支持灵活插入删改

③ 隐式链接：链接指针隐含在盘块中

④ 显式链接：在内存中建立文件分配表 FAT，将链接各物理块的指针显式地存放在表中

⑤ FAT 表项中存放盘块编号，FAT32 即 32 位表项长度

⑥ 最大磁盘容量 $= 2^{\text{表项位数}} \times$ 簇大小 $\times$ 盘块大小

(3) 索引式文件结构 / 索引组织方式

① 为每一个文件分配一个盘块用作索引块，将属于它的盘块号集中放在索引块中

② 支持直接访问、不会产生碎片，但小文件索引块利用率低

③ 增量式索引组织方式：索引结点所能访问的地址空间大小，决定了系统能存储的文件大小

④ 文件大小 $= \sum \left(\dfrac{\text{索引块大小}}{\text{块编号大小}} \right)^{n} \times$ 数据块大小

⑤ n 为间址次数，直接块可以视为 n=0

⑥ n 级间址需要访存 n+1 次

2. 文件存储空间的管理

(1) 空闲表法、空闲链表法

(2) 位示图法

① 盘块号、行号、列号都从 1 开始

② 1 表示已占用、0 表示空闲

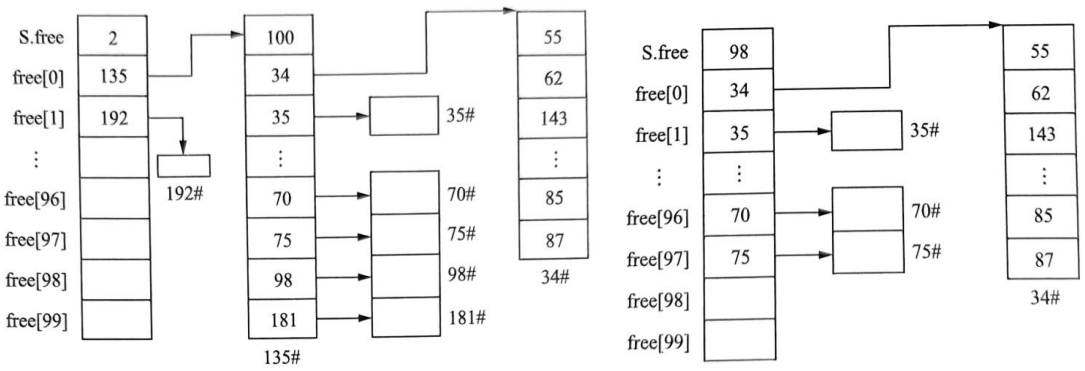
③ 位示图占用空间 = m×n÷8 Byte

④ 盘块号 b = n * (i - 1) + j

(3) 成组链接法

① 分配：从栈顶取出一空闲盘块号，然后修改栈顶指针（空闲盘块数）。若该盘块已是栈底，则将栈底盘块号所对应盘块的内容读入栈中，作为新的盘块号栈的内容。

② 回收：将回收盘块的盘块号放入栈顶，然后修改栈顶指针。若入栈前栈已满，则将现有栈中的 100 个盘块号记入新回收的盘块中，将新回收的盘块作为新的栈底。



3. 提高磁盘 IO 速度的途径

(1) 设置磁盘高速缓存，在内存缓冲区中保存某些盘块的副本

(2) 提前读，预先读取下一个盘块的数据

(3) 延迟写，增加盘块副本在内存缓冲中的留存时间

(4) 优化物理块的分布，减少磁头移动距离

(5) 使用虚拟盘（RAM 盘）

(6) 使用廉价磁盘冗余阵列 RAID，采取并行传输方式，以牺牲容量为代价提升速度，也可以提升磁盘可靠性

4. 提高磁盘可靠性的技术

(1) 第一级容错技术 SFT-Ⅰ（磁盘级）

① 双份目录和双份 FAT

② 热修复重定向和写后读检验

(2) 第二级容错技术 SFT-Ⅱ（主机级）

① 磁盘镜像

② 磁盘双工

(3) 基于集群技术的容错功能

① 双机热备份模式

② 双机互为备份模式

③ 公用磁盘模式