1. 概述

1.1. 操作系统的定义：

是计算机系统的一种系统软件，用于管理计算机的资源和控制程序的执行。

1.2. 提供用户的使用方式

命令行、系统调用、图标窗口。

1.3. 五大功能；

处理机与进程管理、存储器管理、文件管理、设备管理、网络管理。

1.4. 操作系统的分类、各种类别的特点；

单用户操作系统（用户独占全机、CPU 等待手工操作）、批处理操作系统（用户不能直接控制作业运行）、实时操作系统（事件驱动设计）、分时操作系统（同时、独立、交互、及时）、网络操作系统（提供网络服务）、分布式操作系统（资源共享、无主次、两两通信、程序并行运行）、嵌入式操作系统（管理系统资源、抽象硬件、提供开发工具库函数）。

1.5. 特权指令

只能由操作系统运行的指令是特权指令，用户不能运行。

1.6. 处理器的状态

核心状态、管理状态、用户程序状态（管态和目态）

1.7. 程序状态字。

指示处理机状态

1.8.

1.9. 操作系统结构

电子线路、指令集、过程、中断、进程原语、局部辅存、虚拟存储器、通信（管道）、文件系统、设备、目录、用户进程、外壳。13 层

1.10. 现代操作系统特点

个人独占；多处理机；技术结构：微内核、多线程、对称多处理机、面向对象与软件工程原则。

2. 进程与线程

2.1. 多道程序并发执行的特点

资源共享性、失去了封闭性和可再现性、相互制约性。

2.2. 进程的定义

进程是能和其他程序并行执行的程序段在某数据集合上的一次运行过程，是系统资源分配和调度的一个独立单位。

2.3. 进程与程序的区别

程序是指令集合，进程是程序一次运行活动；进程独立调度与其他进程并行，程序不能。一个程序可以对应多个进程。

2.4. 进程的组成

程序、数据集合、进程控制块 PCB（与进程一一对应，是进程存在的标志）。

2.5. 进程的三态及转换

就绪、执行、阻塞

2.6. 进程控制原语

创建、撤销、阻塞、唤醒。

2.7. 线程的定义

线程是进程中的一个实体，是比进程更小的能独立运行的基本单位，不独立拥有资源。

2.8. 线程与进程的区别。

进程是资源的拥有者，有独立地址空间，不是处理器调度基本单位。线程不独立拥有资源，只有 TCB 和堆栈，共享地址空间，是处理器调度基本单位。

2.9. 多核、多线程与超线程

1.多核计算机打破单核环境下的正确性与可靠性。多核在一个 CPU 布置多个执行核（ALU）。

2.超线程在一个 CPU 上同时执行多个程序共同分享该 CPU 内资源，让应用程序在同一时间使用芯片不同部分。

3.多线程：任意时刻多个线程同时被调度运行。

3. 互斥与同步

3.1. 并发进程产生与时间有关错误

结果不唯一、永远等待。

3.2. 临界区，对临界区的管理要求

在某段时间内只允许一个进程使用的资源是临界资源，并发进程访问临界资源的程序段是临界区。要求：互斥、进展、有限等待。

3.3. 信号量的定义，其物理意义

一个进程检测到某个信号后被强迫停止在一个特定的地方，直到收到一个专门的信号后才能继续执行。这个信号就称为“信号量”。信号量被定义为含有整型数据项的结构变量，其整型值大于 0 代表可供并发进程使用的资源实体数，小于 0 表示正在等待使用临界区的进程数。

3.4. 原语概念

执行时不可中断的过程。

3.5. 用信号量实现进程互斥与同步

3.5.1. 哲学家餐问题同步与互斥：

3.5.1.1. 方案一（最多允许四个哲学家拿筷子）：

```
semaphore chopstick[5]={l，1，l，1，1}，s=4；
void philosopher(int i)  /*哲学家进程*/
{ while (true) {
P(s) ;
P(chopstick[i])；P(chopstick[(i+1)％5])；
…；
eat；  /*进餐*/
…；
V(chopstick[i])；  V(chopstick[(i+1)％5])；
V(s) ;
…；
think；  /*思考*/
…；
}
}
```

解释：

这段代码使用信号量的机制来避免死锁，主要体现在两个方面：

1. **有限的资源：** 通过`s`信号量，限制同时进行进餐的哲学家数量。`P(s)`和`V(s)`分别是对`s`信号量的等待和释放操作。初始时`s`为 4，表示最多允许 4 个哲学家同时进餐。当一个哲学家开始进餐时，首先要通过`P(s)`等待信号量，如果信号量的值大于 0（有可用资源），则允许进餐，同时`V(s)`会释放信号量，增加可用资源的数量。这样可以确保系统不会因为资源不足而导致死锁。

2. **筷子的获取：** 哲学家获取筷子的过程通过`P(chopstick[i])`和`P(chopstick[(i+1)%5])`操作，其中`P`操作是对信号量的等待操作。在这里，如果哲学家不能同时获取左手边和右手边的筷子（即等待两个`P`操作的过程中某一个失败），他会一直等待，不会一直占有一个筷子而导

致其他哲学家无法获取筷子。这避免了资源争夺的情况，从而避免了死锁。
总体来说，通过合理的信号量设计，确保对资源的获取和释放操作都是原子的，能够避免死锁的发生。在这个问题中，哲学家在获取资源（筷子）时会等待，而且通过有限的资源控制，能够防止所有哲学家同时进餐，从而有效地避免了死锁。

3.5.1.2. 方案二（设定拿筷子的规则，避免死锁）：

```
semaphore chopstick[5] = {1, 1, 1, 1, 1};
void philosopher(int i) {
    while (true) {
        if (i < (i+1)%5) {
    ① P(chopstick[i]);
            P(chopstick[(i+1)%5]);
        } else {
            P(chopstick[(i+1)%5]);
            P(chopstick[i]);
        }
        // 进餐操作
        V(chopstick[i]);
        V(chopstick[(i+1)%5]);
        // 思考操作
    }
}
```

解释：
初始化： semaphore chopstick[5] = {1, 1, 1, 1, 1}; 初始化了 5 个信号量，每个信号量对应一个筷子，初始值为 1，表示筷子可用。

互斥性： 这段代码确保哲学家在进餐前会同时获取左右两只筷子，以确保互斥性。通过比较 i 和(i + 1) % 5 的大小关系，决定先获取左边的筷子还是右边的筷子。

信号量操作：
P(chopstick[i]); 和 P(chopstick[(i + 1) % 5]); 分别是对左手边和右手边的筷子进行等待操作。
V(chopstick[i]); 和 V(chopstick[(i + 1) % 5]); 分别是对左手边和右手边的筷子进行释放操作。
循环： while (true) 表示哲学家会一直循环进行思考、进餐和思考的过程。

这段代码确保了哲学家在进餐时会先获取左右两只筷子，确保了互斥性，从而避免了死锁。这种策略被称为“左右等待”或“左右规则”，通过一个规定的获取筷子顺序，确保哲学家不会陷入循环等待的死锁情况。

3.6. 直接通信，间接通信；

1.直接通信是指发送进程把信件直接发给接受进程。对称指名通信：Send(P,信件)，

receive(Q,信件)。非对称指名通信方式：发送者先向系统申请缓冲区，把信件存入缓冲区。接受进程被唤醒，接收信件。

2.间接通信指把信件发送到共享的信箱中，接受进程到信箱取信件。

3.7. 死锁的定义，四个必要条件

一组并发进程彼此相互等待对方占有的资源，在得到对方的资源之前不会释放自己占有的资源，造成这组进程都不能继续向前推进的情况。
必要条件：互斥、部分分配、不可抢占、循环等待。

3.8. 死锁的防止

1.资源静态分配（一个进程必须在执行前申请所需的全部资源，直到所需的资源得到满足后才开始执行）

2.资源层次分配（把资源分成多个层次，一个进程得到某一层资源后只能申请更高层资源，释放资源时先释放高层资源）。

3.9. 死锁的避免与银行家算法

银行家算法

3.10. 死锁的检测与恢复

死锁检测：资源分配图与等待图。
死锁解除（恢复）：撤销进程法、剥夺资源法。

死锁例题

某一系统在T0时刻进程的资源分配“瞬间状态”为：

进程	已占有资源量				最大资源需求量				剩余资源量			
	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4
P1	0	0	0	1	5	5	3	2	3	2	1	0
P2	1	1	2	3	3	4	5	5				
P3	2	1	1	0	4	3	5	6				
P4	1	2	3	2	4	4	4	2				

使用银行家算法回答：

(1)系统是否安全？如果安全，请给出一个安全执行序列，如果

不安全，说明理由。要求给出计算过程。

(2)如果在 T0 时刻进程 P1 申请资源(0，1，1，1)，系统能否立即

满足进程的要求？请说明理由。

【解】(1)、安全的。(2 分)

先计算各个进程还需要的资源：

P1：(5 5 3 1)，P2:(2 3 3 2)，P3: (2 2 4 6)，P4:(3 2 1 0)；

- 系统剩余资源为(3 2 1 0)，仅能满足进程 P4 对资源的最大需要，故分配 P4 所需要的资源，P4 在得到所需全部资源后应该在有限的时间内执行结束，并归还它所占用的全部资源；
- 系统现在可用资源为(4 4 4 2)，仅能满足进程 P2 对资源的最大需要，在 P2 执行结束并归还它所占用的全部资源后系统资源为(5 5 6 5),此时可满足 P1 对资源的最大需求；
- 在 P1 执行结束并归还它所占用的全部资源后系统资源为(5 5 6 6)，此时可满足 P3 对资源的最大需求；
- 因此系统是安全的，安全执行序列为 P4P2P1P3(6 分)。

(2)、若在 T0 时刻进程 P1 申请(0，1，1，1)，系统不能立即满足进程的要求。因为，虽然系统剩余资源能够满足进程 P1 当前对资源申请需求，但不满足其对资源的最大需求，如果分配给 P1，那么系统剩余资源不能满足任何一个进程对资源的最大需求，没有一个进程能够安全执行结束，从而造成相互等待的情况，导致系统的不安全性(4 分)。

死锁检测：资源分配图和进程等待图

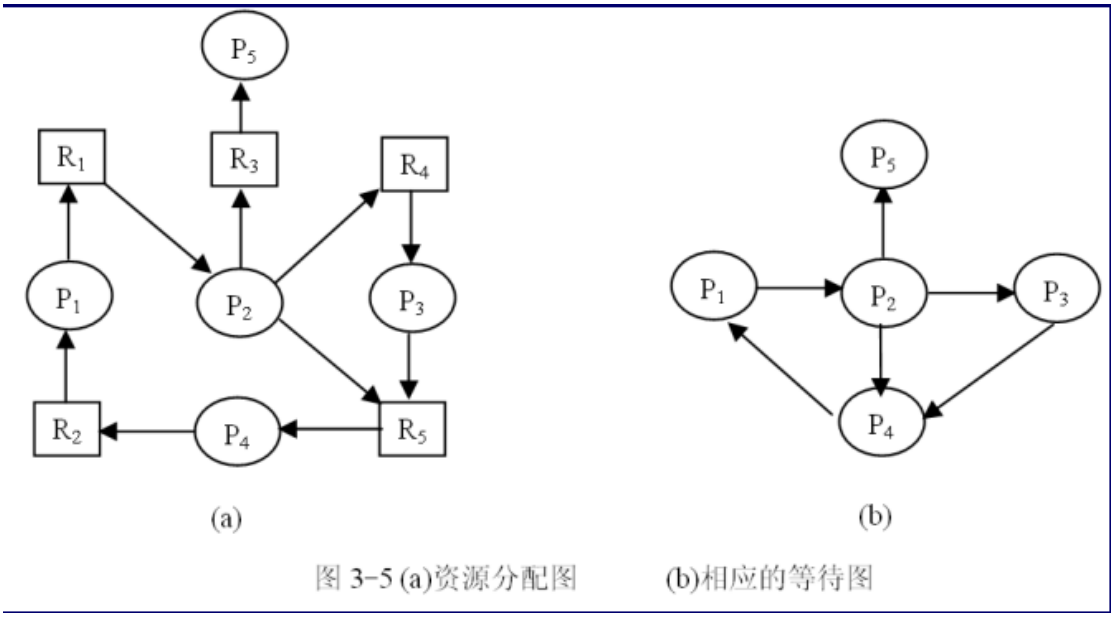

图 3-5 (a)资源分配图　　(b)相应的等待图

4.　处理器调度

4.1. 处理器调度层次

1.长程调度（决定作业是否能创建进程运行、分配资源、创建后是就绪或就绪挂起）

2.中程调度（将内存调到外存对换区挂起）

3.短程调度（哪个进程获得处理机）。

4.2. 调度准则

资源（cpu）利用率、平衡资源、响应时间、周转时间、吞吐率、公平性。

4.3. 短程调度算法

单处理器调度算法、多处理器调度算法。

4.4. 单处理器调度算法

单处理机：先来先服务、时间片轮换、短作业优先、优先级调度、多级反馈队列轮换法。

4.5. 多处理器调度算法

多处理机：负载共享调度算法、组调度算法、专用处理机分配调度算法。

4.6. 实时调度

有限等待时间、有限响应时间、用户控制、可靠性高、系统处理错误能力强。算法：频率单调调度算法、时限调度算法、最少裕度法。

5. 存储管理

5.1. 存储管理的功能

存储空间的分配与回收、地址映射与重定位、存储共享与保护。

5.2. 存储分配的三种方式

1.直接存储分配方式（直接操作物理地址）。

2.静态存储分配方式（使用逻辑地址，程序一直占据存储空间直到结束）。

3.动态存储分配方式（一部分一部分动态装入，利用覆盖与交换）。

5.3. 重定位的定义

由于用户程序装入而引起的地址空间中的相对地址转化为存储空间中绝对地址的地址变换过程，称为地址重定位。

5.4. 两种重定位的特点与区别

1.静态地址重定位是用户程序在装入时由装配程序一次完成，以后不变。

2.动态地址重定位是程序执行过程中，cpu 要对存储器访问时通过硬件地址变换机构将程序和数据地址转换成主存地址，逻辑地址 MA=(重定位寄存器 BR)+(相

对地址寄存器 VR)。

5.5. 覆盖与交换

1.覆盖指一个用户程序不在一开始把全部程序数据装入主存，把程序划分成若干个相互独立的程序段，让那些不会同时被 CPU 执行的程序段共享同一个主存区。某一程序段执行时将该程序段装入主存覆盖以前某个程序段。 2.交换指将系统暂时不用的程序数据从主存中调出，腾出更大存储空间将系统要求使用的程序数据调入主存并将控制权转交给它。在主存和外存交换程序数据。

5.6. 分区存储管理(各种方法采用的分配回收算法，数据结构，地址变换过程，共享与保护，优缺点比较）

1.固定分区法采用分区说明表将主存的用户可用区域划分为若干个大小不等的区域，每个进程占用一个或多个区域。优缺点：实现简单，但存在内部碎片。

2.动态分区法在 os 启动时只存在一个空闲分区。

3.分配程序将该区域依次划分给调度程序选中的进程，分配的大小随用户进程对主存的要求改变，不会出现碎片。

4.数据结构：使用可用分区表、自由链表、请求表。

5.算法使用最先适应法（产生碎片）、最佳适应法（产生碎片）、最坏适应法。

6.动态分区回收（释放区与上下空闲区是否相邻的情况）。

7.地址转换：先判断逻辑地址<限长值？（保护：地址越界中断）再用基址寄存器+逻辑地址=绝对地址。

8.共享：各作业共享存储区域部分有相同基址、限长值。

9.优点：多道程序设计资源利用率高、硬件支持少管理简单。缺点：主存利用率不高、只能覆盖交换。

5.7. 页式存储管理(各种方法采用的分配回收算法，数据结构，地址变换过程，共享与保护，优缺点比较）

1.静态分页管理。利用页表、请求表（包含进程号、请求页面数、页表起始地址、页表长度、状态）、存储页框表（位视图、空闲页框链）。

2.分配：首先从请求表查出进程要求的页框数，由存储页框表检查是否有空闲页框，若有则分配设置页表，填写请求表，查找出空闲页框，将对应页框填入页表。

3.回收：将页表中的页框插入存储页框表中，使之成为空闲页框。

4.地址变换：将逻辑地址分为页号和页内偏移，根据页号找到页框号，将页框号和页内偏移拼接。

5.共享保护：数据共享中，不同作业对共享数据页使用不同页号，各自页表的表目指向共享页框。程序共享必须规定统一页号。

6.保护：保护保护权限域或保护键法。（psw 的键和访问页的保护键核对，相符时能访问）。

7.优点：解决了碎片问题。缺点：作业大小仍受主存页框数限制。

5.8. 虚拟存储器基本思想

虚拟存储器基本思想：把作业的所有分段的副本存放在外存上，当作业被调度投入运行时，首先把当前需要用的一段或几段装入主存，在执行过程中，访问到不在贮存的段时，通过缺段中断机构把它从外存调入。流程：由硬件地址转换机构查快表和段表，若快表命中或该段在主存中，核对存储权限。存储合法，则按静态段式存储管理的地址转换得到绝对地址，否则引发保护中断。若该段不在主存，由硬件产生缺段中断，操作系统处理中断时查看可用分区表或自由链表，找出一个足够大的连续区域装入该分段。若找不到连续分区，则检查空闲区总和，使用移动技术合并一个空闲区将该分段装入主存。若空闲区总和不能满足，则调出段。

5.9. 页式虚拟存储工作流程。

页式虚拟存储是一种常见的存储管理方式，它将虚拟存储空间划分为固定大小的页，并将物理内存空间也划分为相同大小的页框，这样可以更灵活地管理内存空间。下面是页式虚拟存储的工作流程：

(1) 程序页划分：在程序加载到内存时，操作系统将程序划分为若干个页，每个页的大小是固定的，通常是 4KB 或者 8KB。这样可以更好地管理程序的内存空间。

(2) 页表管理：操作系统维护一个页表，用于记录每个页的信息，包括页的逻辑地址、对应的物理页框号、访问权限等。当程序需要访问某个页时，操作系统会根据页表进行地址转换，找到对应的物理内存地址。

(3) 页置换：当物理内存空间不足时，操作系统会进行页置换，将一些页从内存中交换出去，以释放内存空间或者将其存储到磁盘上。常见的页置换算法包括最近最少使用（LRU）、先进先出（FIFO）等。

(4) 虚拟内存管理：页式虚拟存储可以实现虚拟内存管理，当程序需要的页不在内存中时，操作系统可以将其从磁盘中加载到内存中，这样可以实现更大的虚拟内存空间。

(5) 内存保护：操作系统可以通过页式虚拟存储实现内存保护，对每个页进行访问权限的控制，防止程序越界访问或者非法访问内存空间。

5.10. 常用的页面置换算法

（FIFO, LRU, OPT,CLOCK, NRU,LFU)

5.11. 段式存储管理的思想

把作业的所有分段的副本存放在外存上，当作业被调度投入运行时，首先把当前需要用的一段或几段装入主存，在执行过程中，访问到不在贮存的段时，通过缺段中断机构把它从外存调入。流程：由硬件地址转换机构查快表和段表，若快表命中或该段在主存中，核对存储权限。存储合法，则按静态段式存储管理的地址转换得到绝对地址，否则引发保护中断。若该段不在主存，由硬件产生缺段中断，操作系统处理中断时查看可用分区表或自由链表，找出一个足够大的连续区域装入该分段。若找不到连续分区，则检查空闲区总和，使用移动技术合并一个空闲区将该分段装入主存。若空闲区总和不能满足，则调出段。

5.12. 段式虚拟存储管理流程

段式虚拟存储管理是一种操作系统中常见的存储管理方式，它将虚拟存储空间划分为若干个段，每个段对应一个逻辑单位，这种方式可以更灵活地管理内存空间。下面是段式虚拟存储管理的流程：

(1) 程序段划分：在程序加载到内存时，操作系统将程序划分为若干个段，如代码段、数据段、堆栈段等，每个段对应一个逻辑单位，这样可以更好地管理程序的内存空间。

(2) 段表管理：操作系统维护一个段表，用于记录每个段的信息，包括段的起始地址、长度、访问权限等。当程序需要访问某个段时，操作系统会根据段表进行地址转换，找到对应的物理内存地址。

(3) 段页式管理：有些系统采用段页式管理，将每个段进一步划分为若干个页，这样可以更灵活地管理内存空间。操作系统会维护一个页表，用于记录每个页的信息，包括页的起始地址、长度、访问权限等。

(4) 虚拟内存管理：在需要时，操作系统可以将某些段或页从内存中交换出去，以释放内存空间或者将其存储到磁盘上，这样可以实现更大的虚拟内存空间。

(5) 内存保护：操作系统可以通过段式虚拟存储管理实现内存保护，对每个段进行访问权限的控制，防止程序越界访问或者非法访问内存空间。

页面置换：

例：设某计算机的逻辑地址空间和物理地址空间均为 64KB，按字节编址。若某进程最多需要 6 页数据存储空间，页的大小为 4KB。操作系统采用固定分配局部置换策略为此进程分配 4 个页框。如果该进程依次要访问的地址序列是：0123H，1234H，2345H，3456H，1357H，0586H，4ABCH，3088H，1235H，0987H。试问分别采用 FIFO、LRU 和 OPT 算法时，各产生多少次缺页中断？并计算相应的缺页中断率，同时写出在这三种调度算法下产生缺页中断时淘汰的页面号和在主存的页面号。

解：根据页式管理的工作原理，应先考虑页面大小，以便将页号和页内位移分解出来。页面大小为 4KB，即 212，则得到页内位移占虚拟地址的低 12 位，页号占剩余高位。因此地址序列：0123H，1234H，2345H，3456H，1357H，0586H，4ABCH，3088H，1235H，0987H 对应的页号是 0,1,2,3,1,0,4,3,1,0。（3 分）

FIFO 共 6 次缺页中断，缺页中断率=6/10=60%（3 分）

访问次序	0	1	2	3	1	0	4	3	1	0
主存页号				3	3	3	3	3	3	3
			2	2	2	2	2	2	2	2
		1	1	1	1	1	1	1	1	0
	0	0	0	0	0	0	4	4	4	4
淘汰							0			1

LRU 共 5 次缺页中断，缺页中断率=5/10=50%（3 分）

访问次序	0	1	2	3	1	0	4	3	1	0
主存页号				3	1	0	4	3	1	0
			2	2	3	1	0	4	3	1
		1	1	1	2	3	1	0	4	3
	0	0	0	0	0	2	3	1	0	4
淘汰							2			

OPT 共 5 次缺页中断，缺页中断率=5/10=50%（3 分）

访问次序	0	1	2	3	1	0	4	3	1	0
主存页号				3	3	3	3	3	3	3
			2	2	2	2	4	4	4	4
		1	1	1	1	1	1	1	1	1
	0	0	0	0	0	0	0	0	0	0
淘汰							2			

6. 文件管理

6.1. 文件

文件是一组逻辑上相互关联的信息的集合，是计算机系统中信息的一种组织结

构。文件的基本操作有创建、删除、写、读、打开、关闭。

6.2. 文件目录

1.文件目录操作：创建、删除、修改、显示、搜索、创建文件、删除文件。

2.文件控制块：File Control Block FCB。每个文件都有一个与之对应的文件控制块，所有文件的文件控制块按一定方式组合在一起构成文件目录表。每个文件控制块对应目录表中的一个目录项。文件目录在文件名和文件自身提供一种映射。

3.索引节点：i 结点（i-node），将 fcb 除文件名以外的信息放入 i 结点，每个文件对应一个索引节点。文件目录中每个目录项仅由文件名和 i 结点构成，与文件信息分离。

6.3. 文件共享与保护

文件共享

1.符号链接（symbol link）：采用无环图目录结构。系统创建一个 link（lnk）类型的文件，新文件中被写入的是被链接文件的路径名。只有文件主才拥有指向其索引结点的指针，其他用户只有路径名。文件主删除文件后其他用户已找不到文件。优点是网络共享时只要文件路径名，缺点是访问共享文件时要多次读盘。

2.硬链接（hard link）：无环图目录结构。文件的物理地址等属性不在目录项而在 inode 中，利用链接计数（count）。共享文件时，用户 B 的目录中新增一个目录项并设置指针指向该文件的索引结点。文件主删除文件时 count-1，count 为 0 时文件被删除。

文件保护

实现文件保密性、完整性、可用性。技术：用户认证、文件保护、数据备份

6.4. 文件存储空间的管理

1.空闲表：连续分配，与内存的动态分配方式类似，为每个文件分配一块连续的存储空间。为空闲区建立空闲盘块表，包含序号、首块号、空闲块数（长度）等信息。

2.空闲链表：storing the free list on a linked list。空闲盘块链将所有空闲盘块连成一条链。空闲盘区链将空闲盘区（每个空闲区可能含多个盘块，自带盘区大小信息）连成一条链。

3.位示图：bitmap。利用一个二进制位表示存储空间一个盘块状态，某位值为 0 表示该盘块空闲。

4.索引链接表法（成组链接法 Group link）：每个索引表块第 0 个表项作为指向下一个索引表块的指针。链表的头指针在超级块中，包括结构和管理两方面信息。把 n 个空闲扇区地址保存在第一个空闲扇区内。文件分配时从链表头索引块表尾部开始分配，若读入第 0 表项则将该表项指针读入超级块作为索引链表表头指针，将该盘块分配给请求空闲块的文件。

6.5. 文件分配

1.空闲表：连续分配，与内存的动态分配方式类似，为每个文件分配一块连续的存储空间。为空闲区建立空闲盘块表，包含序号、首块号、空闲块数（长度）等信息。

2.空闲链表：storing the free list on a linked list。空闲盘块链将所有空闲盘块连成一条链。空闲盘区链将空闲盘区（每个空闲区可能含多个盘块，自带盘区大小信息）连成一条链。

3.位示图：bitmap。利用一个二进制位表示存储空间一个盘块状态，某位值为 0 表示该盘块空闲。

4.索引链接表法（成组链接法 Group link）：每个索引表块第 0 个表项作为指向下一个索引表块的指针。链表的头指针在超级块中，包括结构和管理两方面信息。把 n 个空闲扇区地址保存在第一个空闲扇区内。文件分配时从链表头索引块表尾部开始分配，若读入第 0 表项则将该表项指针读入超级块作为索引链表表头指针，将该盘块分配给请求空闲块的文件。

题目：

A UNIX file system has 4-KB blocks and 4-byte disk addresses. Assume that there are 10 direct block pointers, and a singly, doubly, and triply indirect pointer in each i-node, then

a. What is the maximum file size supported by this system?

b. Assuming no information other than that the file i-node is already in main memory, how many disk accesses are required to access the byte in position 12345?

c. Assuming no information other than that the file i-node is already in main memory, how many disk accesses are required to access the byte in position 1234567890?

翻译：

UNIX 文件系统具有 4KB 的块大小和 4 字节的磁盘地址。假设每个 i-node 中有 10 个直接块指针，以及一个单间接、双间接和三间接指针，那么：

a. 此系统支持的最大文件大小是多少？

b. 假设除了文件 i-node 已经在主内存中之外没有其他信息，访问位置 12345 的字节需要多少次磁盘访问？

c. 假设除了文件 i-node 已经在主内存中之外没有其他信息，访问位置 1234567890 的字节需要多少次磁盘访问？

（1）4K/4=1K

10*4K+1K*4K+1K*1K*4K+1K*1K*1K*4K=4T4G4M40KB

（2）因为 12345<40K，故该地址落入直接索引中，由于 i-node 已在内存，因此只需要 1 次访问磁盘块，即一次访问含有该位置数据的数据块。(3 分)

（3）因为 4M40K<1234567890<4G4M40K，故该地址落入二级间接索引中，由于 i-node 已在

内存，因此只需要 3 次访问磁盘块。

7. 第七章 I/O 管理

7.1. I/O 管理概述

1.目标：为用户提供方便、统一的界面，提高资源利用率。

2.功能：设备控制、设备分配与回收、其他功能。

3.组成：设备、控制器、通路、（应用接口）。

7.2. I/O 控制方式

1.程序直接控制方式（查询）。

2.中断。

3.DMA（参数准备、DMA 工作（发 DMA 请求，并用 DMA 控制器控制总线）、结束中断）。传送数据块，直接将数据存入内存不经过 cpu。

4.通道（独立于 CPU 的 I/O 处理机，控制设备与内存直接数据交换，有自己的指令系统）

7.3. I/O 缓冲

1.引入原因：缓解 CPU 与 I/O 设备之间速度不匹配的矛盾，提高 CPU 与 I/O 设备之间的并行性。减少对 CPU 中断频率，放宽对中断响应时间限制。

2.单缓冲：用户进程发出 I/O 请求时操作系统分配一个位于内存中系统部分的缓冲区。读操作时输入数据被放到缓冲区中，缓冲区已满时进程把缓冲区数据移到用户空间。I/O 请求为写操作时进程将输出数据放到缓冲区，缓冲区已满时数

据输出到 I/O 上。只能有一个进程输入输出。

3.双缓冲：在进程对一个缓冲区传送数据时，操作系统清空另一个缓冲区。

4.循环缓冲：多个缓冲区连接起来构成循环缓冲区，一部分用于输入，另一部分用于输出。实现多个设备间、I/O 设备与 CPU 的并行工作能力。

5.缓冲池：由输入缓冲队列、空缓冲队列、输出缓冲队列组成。还有收容输入缓冲区、收容输出缓冲区、提取输入缓冲区、提取输出缓冲区。进程输入时把空缓冲队列队首摘下一个空缓冲区收容输入，输入完成后把它挂到输入队列队尾。同理，进程输出时…收容输出，输出完成后…输出队列队尾。提取输入数据时从输入队列取得一个缓冲区作为工作缓冲区，进程提取数据，用完后挂到空缓冲区队尾。同理，提取输出时从输出队列…

7.4. 设备驱动程序

1.模式：作为内核过程实现，与文件系统一起把设备作为特殊文件处理。或作为独立 I/O 进程实现。

2.功能：1.把抽象要求转为具体要求。2.检查 I/O 命令合法性。3.发 I/O 命令。4.相应中断请求。5.自动构成通道程序

3.特点：是请求 I/O 设备与设备控制器的通信程序。与 I/O 设备特性紧密相关，与 I/O 控制方式紧密相关，基本部分固化在 ROM 中。

4.处理过程：将抽象要求转化为具体要求，检查合法性，读出检查设备状态，传送必要参数，设置工作方式，启咏 I/O 设备。

7.5. 设备分配

1.原则：提高利用率避免死锁、方便用户使用（设备无关性）。

2.分配方式：静态分配、动态分配。

3.安全性：安全分配方式指进程发出 I/O 请求后进入阻塞，直到 I/O 操作完成后被唤醒。不安全分配方式指发出 I/O 请求后又在需要时继续发出请求，当请求设备被另一进程占用时阻塞，可能产生死锁。

4.逻辑设备名到物理设备名映射：将用户程序和具体物理设备隔离。用逻辑设备表 LUT 将逻辑设备名映射为物理设备名。

5.数据结构：设备控制表（DCT）记录本设备状态、控制器控制表指针、设备队列队首指针、重复执行次数。控制器控制表（COCT）记录控制器状态信息和通道的连接情况。通道控制表（CHCT）记录通道状态和设备控制器连接情况。系统设备表（SDT）记录全部设备信息。

6.流程：分配设备、分配控制器、分配通道。

7.SPOOLing(Simultaneous Peripheral Operation On-Line)组成：输入设备输出设备、输入缓冲区输出缓冲区（在内存）、输入井输出井（在磁盘）。作业前先通过输入程序将输入数据存放到输入井，预输入。作业中使用输入数据时从输入井获取。输出时先将数据输出到输出井，输出进程把数据从输出井传送到输出缓冲区，最后输出到输出设备。

7.6. 磁盘存储器的管理,常用调度算法

FCFS、SSTF、SCAN 、LOOK、C-SCAN

7.7. 磁盘阵列(RAID)

‘把多个磁盘组合成一个磁盘阵列组。作为独立系统在主机外连接。磁盘阵列内有缓冲存储器。提高传输速率、提供校验功能。

7.8. 时钟管理和电源管理

xxxxx

题目：

设某磁盘旋转速度为每分钟 6000 转。每个磁道有 100 个扇区，相邻磁道间的平均移动时间为 1ms。若在某时刻，磁头位于 100 号磁道处，并沿着磁道号增大的方向服务，磁道号请求队列为 50、200、80、110、180，对请求队列中的每个磁道需读取 1 个随机分布的扇区，则：

(1)在 SSTF(最短寻道)调度情况下,读完这 5 个扇区总共需要多少时间,要求给出计算过程；

(2)在 SCAN(扫描)调度情况下，读完这 5 个扇区总共需要多少时间，要求给出计算过程；

(3)就以上二种算法进行对比分析，并为用户选择提供建议说明。

【解】

由于转速为 6000r/m，

则平均旋转延迟为 60*1000/(6000*2)=5ms

总的旋转延迟时间为 5*5ms=25ms。

由于旋转一周的时间为 10ms，每个磁道有 100 个扇区，读取一个磁道上一个扇区的平均读取时间为 10ms/100=0.1ms，总的读取扇区的时间=0.1ms*5=0.5ms。

1）SSTF:

执行顺序是 100->110->80->50->180->200

移到磁道长度为 10+60+150=220

故读取上述磁道上所有扇区所花的总时间=220ms+25ms+0.5ms=245.5ms

2）

采用 SCAN 调度算法访问磁道的顺序为 100->110->180->200->80->50

则移到磁道长度为 100+150=250

故读取上述磁道上所有扇区所花的总时间=250ms+25ms+0.5ms=275.5ms

3）两者算法都有寻道时间短、寻道优化的优势，而且两种寻道时间相差 30ms，但相比较而言，SSTF 可能会频繁切换服务方向，而 SCAN 不会。由于磁臂调度是机械运动，存在惯性，频繁切换服务方向会降低机械装置的使用寿命，故实际应用中 SCAN 优于 SSTF

8. 操作系统的安全性

8.1. 安全的计算机系统的三个特性：

保密性、完整性和可用性

8.2. OS 的安全性是计算机系统安全的基础

有选择的访问控制、内存管理与对象重用、审计能力、文件和数据的加密以及进程间安全通信机制。

8.3. 保护机制

保护域、ACL、权能表

8.4. 安全机制

用户认证；恶意代码及时防范、发现和清除；使用情况的监控和审计；加密技术；备份。

9. 新型操作系统

9.1. 嵌入式操作系统

微软嵌入式操作系统、嵌入式 linux 与安卓、TinyOs、VxWorks

9.2. 多媒体操作系统

9.3. 分布式操作系统

集群系统、MapReduce 计算模型

9.4. 集群系统

9.5. MapReduce 计算模型

9.6. 虚拟化技术