一、OS概述

1.1 操作系统的概念

- 定义
  - 从计算机系统组成观点——操作系统是系统软件
  - 从资源管理观点——操作系统是系统资源管理者
    - 软件资源：信息（数据和程序）
    - 硬件资源: I / O设备、存储器、处理器
  - 从软件分层、扩允机器的观点——操作系统是扩充裸机的第一层系统软件
- 从服务用户的观点——操作系统是用户与裸机之间接口
  - **命令接口、程序接口**
- **操作系统的特征: 并发，共享，虚拟，异步**
  - 并发是同一时间间隔，并行是同一时刻
  - 虚拟是指将一个物理实体变为若干个逻辑上的对应物
- 操作系统的目标：有效性、方便性、可扩充性、开发性
- 联机操作与脱机操作
  - 联机操作：输入输出由主机控制完成，再由主机完成运算
  - 脱机操作：利用卫星机完成输入输出功能，主机完成计算

1.2 操作系统的类型（概念及特点）

- 微机操作系统:
  - 单用户单任务操作系统
  - 单用户多任务操作系统
  - 多用户多任务操作系统
- 批处理操作系统
  - 单道批处理操作系统：脱机I/O + 批处理技术
    - 优点是系统吞吐量大，资源利用率高。
    - 缺点是作业的周转时间长，用户无法实现对作业的控制。
    - CPU和I/O设备使用忙闲不均
  - 多道批处理操作系统：脱机I/O + 多道程序设计技术
    - 优点是资源利用率高，作业吞吐量大
    - 缺点是用户交互性差，作业平均周转时间长
- 分时操作系统：在一台主机上连接了多个带有显示器和键盘的终端，它同时允许许多用户通过终端以交互方式使用计算机共享主机中的资源
  - 多路性、独立性、交互性、及时性
  - 响应时间 = 用户数 * 每个用户的运行时间片
- 实时操作系统：对随机发生的外部事件作出及时响应并对其进行处理

    - 快速的响应时间、有限的交互能力、高可靠性
- 网络操作系统：使网络上各计算机能方便地进行网络通信、有效地共享网络资源，为网络用户提供所需各种服务的软件和有关规程的集合。
    - 内装网络
- 分布式操作系统：能直接对系统中各类资源进行动态分配和管理，有效控制和协调诸任务的并行执行，允许系统中的处理单元无主、次之分，并向系统提供统一的、有效的接口的软件集合。
    - **与网络操作系统的区别：**
        - **分布性：处理上的分布性是分布式操作系统的最基本特征。网络虽有分布处理的功能，但网络的控制功能，则大多集中在某个主机或服务器上，它的控制方式集中的，分布式系统的处理分布是资源、功能、任务和控制都是分布的。**
        - **透明性：分布式OS通常很好地隐藏系统内部的实现细节，如对象的物理位置、并发控制、系统故障等对用户都是透明的。**
        - **统一性：分布式系统要求一个统一的操作系统，实现系统操作的统一性，而网络系统一般是在各自操作系统基础上加上负责网络访问功能模块，网络各系统操作不一致。**
        - **健壮性：由于分布式系统的处理和控制功能是分布的，设备出现故障时，可通过容错技术实现系统重构，从而仍保持系统的正常运行，因而系统具有键壮性，即具有较好的可用性和可靠性。而网络系统其控制功能大多集中在主机和服务器中，使系统具有潜在的不可靠性。**

1.3 操作系统的功能

- **处理机(进程)管理：**
    - **任务：**处理机管理的主要任务是对处理机进行分配，并对其运行进行有效的控制和管理。
    - **功能：**控制、进程同步、进程通信和调度
- **存储器管理：**
    - **任务：**存储器管理的主要任务是为多道程序的运行提供良好的环境，方便用户使用存储器，提高存储器的利用率，以及能从逻辑上来扩充内存
    - **功能：**内存分配、内存保护、地址映射、内存扩充
- **I/O设备管理：**
    - **任务：**设备管理体制的任务是登记各I/O设备状态，管理并完成用户提出的I/O请求，按一定的策略为用户分配I/O设备。同时提高CPU和I/O设备的利用率，提高I/O速度，方便用户使用I/O设备
    - **功能：**缓冲器管理、设备分配、设备处理、虚拟设备
- **信息管理(文件系统管理)：**
    - **任务：**文件系统的任务是对用户文件和系统文件进行管理，以方便用户使用，并保证文件的安全性
    - **功能：**对文件存储空间的管理、目录管理、文件共享和保护
- **用户接口：**该接口分成二种：一种是作业级接口，它提供一组键盘命令，供用户去组织和控制作业的运行。另一种是程序级接口，它提供一组系统调用供其它程序调用。

1.4 微内核

- 概念：微内核操作系统是一种操作系统设计架构，其中内核仅包含最基本的功能，如进程调度、内存管理和进程间通信（IPC）等核心服务，其它功能则采用进程方式运行，运行在用户态，系统进程与内核是一种客户/服务器模式，采用消息机制通信
- 优点：可扩展性更好、可靠性高、可移植性强、支持分布式、面向对象技术引入
- 缺点：内核态和用户态切换及消息传递导致运行效率低

二、进程管理

2.1 进程的描述

- 进程与程序的区别：动态与静态、进程具有并发性、是资源分配的基本单位，是调度的基本单位，进程有生命特征
- 进程的特征：结构，动态，并发，独立，异步
- **进程的状态：运行态、就绪态、阻塞态**
  - **进程调度**
    - **运行态→阻塞态：处于运行态的进程在运行过程中需要等待某一事件发生后（例如因I / O请求等待I / O完成后），才能继续运行**
    - **阻塞态→就绪态：处于阻塞态的进程等待的事件已经发生**
    - **运行态→就绪态：处于运行状态的进程在其运行过程中，分给它的处理机时间片已用完**
    - **就绪态→运行态：当处理机空闲时，进程调度程序必将处理机分配给一个处于就绪态的进程**
  - 需要注意进程数在状态之间的数量关系
- **进程控制模块PCB(Process Control Block)：进程存在的唯一标志**
  - 信息
    - 进程标识符：用于唯一地标识一个进程。它由外部标识符（由字母组成，供用户使用）和内部标识符（由整数组成，为方便系统管理而设置）两种。
    - 处理机状态信息：它由处理机各种寄存器（通用寄存器、指令计数器、程序状态字PSW、用户栈指针等）的内容所组成，该类信息使进程被中断后重新执行时能恢复现场从断点处继续运行。
    - 进程控制信息：它包括程序和数据的地址、I / O资源清单，保证进程正常运行的同步和通信机制等。
    - 家族信息：它包括该进程的父、子进程标识符、进程的用户主等。
  - 组织方式
    - 线性方式
    - 链接方式
    - 索引方式
- 进程 = 程序 + 数据 + PCB
- **进程上下文切换：现场信息， 地址空间等，保存在PCB中，进程恢复执行时从PCB中恢复**

2.2 进程的控制

- 内核
  - 概念
  - **核心态和用户态的转换：通过系统调用，用户态变成核心态， 系统调用返回，则由核心态变成用户态**
  - 原语
- “挂起”和“激活”操作
  - 当进程处于运行态和活动就绪态时，执行挂起操作，进程状态转换为静止就绪态
  - 当进程处于活动阻塞态时，执行挂起操作，进程状态转换为静止阻塞态
- 进程控制原语：创建原语、撤消原语、阻塞原语、唤醒原语、挂起原语、激活原语
  - 注意执行原语的进程和原语操作的进程间的关系

2.3 线程的描述与控制

- **线程具有唯一的标志符和线程控制块(TCB)，线程控制块中包含有线程的一切私有信息**
- 线程的特征：并发、共享、动态、结构
- 线程的状态及状态变化
- 线程库
- **线程的实现方式：内核级线程，用户级线程、混合式线程**
  - **在支持线程的系统中，资源是以进程为基本单位进行分配，而线程是进行调度的基本单位**
  - 用户级线程
    - 优点：创建和切换速度快，线程管理灵活，线程数量不受操作系统限制
    - 缺点：单一线程阻塞，进程下的全部线程都不能执行，无法利用多处理器并行执行，无法利用内核优化
  - 内核级线程
    - 优点：线程可以在多处理器上并行执行，利用内核优化，一个线程阻塞不影响同一进程中的其它线程。
    - 缺点：创建和切换开销大，线程管理开销，线程数量受操作系统限制。

2.4 进程同步

- 进程间制约关系
  - 资源共享关系：互斥、死锁、饥饿
  - 相互合作关系
- 临界资源：一次只允许一个进程使用的资源称为临界资源
- 临界区：把各进程分解，把访问临界资源的那段代码（称为临界区）与其它段代码分割开来，只对各种进程进入自己的临界区加以限制，即各进程互斥地进入自己的临界区
- **进程同步机制：空闲让进、忙则等待、有限等待、让权等待**

- **信号量的含义：** `n = mutex.value`
  - `n > 0`：表示n个某类资源空闲，可供使用
  - `n = 0`：表示资源已被占用，无其它进程等待
  - `n < 0`：表示资源已被占用，还有-n个进程因等待资源而阻塞
- **P-V操作：代表申请和释放资源**

```
proceduce P(S)
    var S: semaphore;
begin
    S.value = S.value - 1;
    if S.value < 0 then
        block(S.L);
end
```

```
Procefuce V(S)
var S: semaphore;
begin
    S.value = S.value + 1;
    if S.value <= 0 then
        wakeup(S.L);
end
```

- **利用信号量实现进程互斥：为使多个进程能互斥地访问某临界资源，只需为该资源设置一个互斥信号量mutex，并设其初值为1，然后将各进程的临界区CS置于P(mutex)和V(mutex)操作之间即可。**

```
var mutex: semaphore:= 1 ;
begin
    parbegin
        A:begin
            Input datd 1 from I/O 1;
            Compute……;
            P(mutex);
            Print results1 by printer;
            V(mutex);
        end
    parend
end
```

- **利用信号量实现进程同步：设置一个同步信号量full，它代表的资源是缓冲器满，初值为0，一个同步信号量empty，它代表的资源是缓冲器空，初值为1，写入数据的程序执行P(empty)和V(full)，读取数据的程序执行P(full)和V(empty)**

```
var empty, full: semaphore:= 1, 0;
begin
    parbegin
        I: begin
            repeat
                Compute next number;
                P(empty);
                Add to buffer;
```

```
                V(full);
            until false;
        end

        O: begin
            repeat
                P(full);
                Take from Buffer;
                V(empty);
                Print last number;
            until false;
        end
    parend
end
```

- 生产者-消费者问题
- 读者-写者问题

2.5 进程通信

- **高级通信机制**
  - **共享存储器系统**
  - **消息传送系统**
    - 直接通信方式
    - 间接通信方式：通过信箱这个2传手来交换信件
  - **管道通信系统**

2.6 进程调度

- 处理机三层调度
  - 高级(Long-term)调度——作业调度
  - 中级(Medium-term)调度——对换
  - 低级(Short-term)调度——进程调度
- **作业调度**
  - 作业控制块（JCB）：作业存在的标志，包含作业的名称、作业对资源的需求信息、作业的资源使用信息、作业的控制方式、作业类型、作业优先级和作业状态
  - 作业的状态：提交，后备，运行，完成
    - 作业的运行状态是宏观状态，其实对应的进程有时确实占用CPU运行，有时却在阻塞或就绪状态
  - **调度算法：**
    - **先来先服务（优点：公平，缺点，没有考虑缩短周转时间，对短作业不利）**
    - **短作业优先（照顾短作业，存在饥饿现象，平均周转时间最小）**
    - **响应比高者优先（兼顾短作业、先进先出、长作业）**
      - 响应比 $=\frac{\text{等待时间}+\text{处理时间}}{\text{处理时间}}$

        - **优先级（紧急作业）**
        - **多队列（平衡资源的使用）**
- **进程调度**
    - **进程调度必须具备的基本机制：排队，分派和上下文切换**
    - 非抢占式调度：处理器分配给进程后，一直到进程结束或进程阻塞，进程才自动放弃处理器
    - 抢占调度：一个进程正在处理器中运行时，操作系统可以根据规定的抢占原则`(时间片，优先级，短进程)`，将已经分配给进程的处理器从进程剥夺，并分配给其他的进程
    - 与非抢占调度相比，抢占调度不但要选取一个就绪进程来运行，同时把现运行进程强制变成就绪状态
    - **调度算法：**
        - **先进先出**
        - **短进程优先（照顾短进程）**
        - **优先级（用于紧急进程，需要配合抢占调度方式）**
        - **时间片轮转法（用于人机交互系统、分时系统）**
        - **多级队列调度**
        - **多级反馈调度**
- **评价调度算法好坏的指标：周转时间（用于评价作业调度），响应时间（进程调度），截止时间（实时调度）**

2.7 进程死锁

- **死锁的原因**
    - **竞争资源引起死锁**
    - **进程推进顺序不当引起死锁**
- **产生死锁的必要条件：互斥、不可抢占、请求和保持、环路等待**
- **死锁的预防方法：**
    - 破坏互斥条件
    - 破坏不可抢占条件
    - **破坏请求和保持条件：一次性分配资源**
    - **破坏循环等待条件：按序分配资源**
- **死锁的避免：银行家算法**
    - 数据结构：
        - 可用资源向量 Available[m]
        - 最大需求矩阵 Max[n, m]
        - 分配矩阵 Allocation[n, m]
        - 需求矩阵 Need[n, m]
    - **银行家算法：假设在进程并发执行时进程i提出请求j类资源k个后，表示为$Request_i[j]=k$**
        1. **如果$Request_i \le Need_i$则继续以下检查，否则显示需求申请超出最大需求值的错误**
        2. **如果$Request_i \le Available$则继续以下检查，否则显示系统无足够资源，P_i阻塞等待**

3. **系统试探把要求的资源分配给进程i并修改有关数据结构的值:**

   $Available = Available - Request_i$;

   $Allocation_i = Allocation_i + Request_i$;

   $Need_i = Need_i - Request_i$;

4. **系统执行安全性算法，检查此次资源分配后，系统是否处于安全状态，若安全，才正式将资源分配给进程i，以完成本次分配；否则将试探分配作废，恢复原来的资源分配状态，让进程Pi等待**

- **安全性算法**

  1. **初始化设置工作向量Work[m]表示系统可提供的各类资源数目，用以保护原数据结构有关值。Work = Available，设置完成标志向量 Finish[n]。Finish[i] = false 表示i进程尚末完成，如值为true则表示进程i已完成**
  2. **从进程集合n中找到一个能满足下述二个条件: Finish[i] = false和$Need_i \le Work$，如找到则执行步骤3，如找不到同时满足以上二条件的进程则执行步骤4**
  3. **当进程i获得资源后可顺利执行直到完成，并释放出分配给它的资源，表示如下：Work = Work + $Allocation_i$ ； Finish[i]=true ； 转执行步骤2**
  4. **如果所有的Finish[i] = ture，则表示系统处于安全状态，否则系统处于不安全状态。**

- 不安全状态并不等于死锁，只能说可能死锁，原因是假设已分配资源不归还的清下，可能存在已分配资源归还的情况， 还存在进程异常终止

三、存储管理

3.1 存储管理概念

- 地址
  - 符号地址：在源程序中，是通过符号名来访问子程序和数据的，这些符号名实际代表了地址，称为符号地址。例如，变量，子程序名，函数名，标号等
  - 逻辑地址：程序中使用的从0开始进行编址的地址，可以是一维或二维地址
  - 物理地址：内存单元的地址
  - 符号地址→逻辑地址由编译程序或者汇编程序加上链接程序共同完成；逻辑地址→物理地址由装入程序或者动态重定位机构完成
  - 链接的方式：静态链接，装入时动态链接，运行时动态链接
  - 装入的方式：绝对装入，静态重定位装入，动态重定位装入
- 地址重定位
  - 静态重定位：静态重定位是在程序执行之前进行重定位
  - 动态重定位：动态重定位是指在程序执行过程中进行地址重定位
  - 分页、分段、段页式、可重定位动态分区等采用动态地址重定位，固定分区、动态分区一般采用静态地址重定位
  - 静态重定位和动态重定位的区别主要是重定位时机不同，静态重定位是发生在装入内存之后运行之前，而动态重定位是在运行时，在执行每条指令期间进行的
- **零头：不可用的存储空间**

- **内零头：固定分区中未利用部分**
- **外零头：动态分区中太小的分区，以至于无法利用**
- **内零头采用动态分区消除，外零头采用动态重定位分区解决，通过移动合并消除**

3.2 存储方式

- 单一连续分配
- 固定分区分配
- 动态分区分配：可变式分区是指在作业装入内存时，从可用的内存中划出一块连续的区域分配给它，且分区大小正好等于该作业的大小。
  - **分区分配算法：**
    - **最佳适应算法：**它从全部空闲区中找出能满足作业要求的、且大小最小的空闲分区，这种方法能使碎片尽量小。
    - **首次适应算法：**从空闲分区表的第一个表目起查找该表，把最先能够满足要求的空闲区分配给作业，这种方法目的在于减少查找时间。
    - **循环首次适应算法：**该算法是首次适应算法的变种。在分配内存空间时，不再每次从表头（链首）开始查找，而是从上次找到的空闲区的下一个空闲区开始查找，直到找到第一个能满足要求的的空闲区为止
    - **最坏适应法：**从所有未分配的分区中挑选最大的且大于和等于作业大小的分区分给要求的作业
  - UNIX分配回收算法
- 动态重定位分区分配：与动态分区一样，但分区可以移动以便把较小的空闲分区合并成一个较大的空闲分区，因而需要动态重定位的支持。
- 覆盖和对换技术

3.3 分页存储管理

- 概念：分页存储管理是将一个进程的地址空间划分成若干个大小相等的区域，称为页，相应地，将内存空间划分成与页相同大小的若干个物理块，称为块或页框。在为进程分配内存时，将进程中的若干页分别装入多个不相邻接的块中。
- 地址：分页系统的地址由页号和位移量(页内地址)组成。

  系统为每个进程建立了一张页面映射表，简称页表。每个页在页表中占一个表项，记录该页在内存中对应的物理块号
- 地址变换机构：地址变换机构的基本任务是利用页表把用户程序中的逻辑地址变换成内存中的物理地址，实际上就要将用户程序中的页号变换成内存中的物理块号。

  为了实现地址变换功能，在系统中设置页表寄存器，用来存放页表的始址和页表的长度。

  1. 当进程被调度时，就将对应的页表的始址和长度装入页表寄存器
  2. 在进行地址变换时，先判断页号是否越界，越界则产生越界中段
  3. 未越界则根据页表寄存器中的页表始址和页号计算出该页在页表项中的位置，得到该页的物理块号，将此物理块号装入物理地址寄存器中
  4. 将有效地址（逻辑地址）寄存器中页内地址直接装入物理地址寄存器的块内地址字段中，这样便完成了从逻辑地址到物理地址的变换

> 假设每页有1KB，现有逻辑地址2500:`页号2 | 业内地址452`，页表:`0|2` `1|4` `2|5`
>
> 1. 首先将对应的页表的始址和长度装入页表寄存器
> 2. 然后根据将页号愚页表长度进行比较，没有越界，则计算得到物理块号5
> 3. 则可以得到物理地址为 5 × 1024 + 452 = 5572

- 具有快表的地址变换机构：为了提高地址变换的速度，在地址变换机构中增设了一个具有并行查询功能的特殊的高速缓冲存储器，称为“联想存储器”或“快表”，用以存放当前访问的那些页表项
- 两级页表机制

3.4 分段存储管理

- 概念：在分段存储管理方式中，作业的地址空间被划分为若干个段，每个段是一组完整的逻辑信息，如有主程序段、子程序段、数据段及堆栈段等，每个段都有自己的名字，都是从零开始编址的一段连续的地址空间，各段长度是不等的
- 地址：分段系统的地址由段号(名)和段内地址组成

  在系统中为每个进程建立一张段映射表，简称为“段表”。每个段在表中占有一表项，在其中记录了该段在内存中的起始地址（又称为“基址”）和段的长度
- 地址变换机构：与分页存储管理中的类似，只不过又加了一个段内地址和对应段的段长的比较来判断是否越界
- 分页和分段的比较

	分页	分段
目的	为了提高内存的利用率	为了能更好地满足用户的需要
页/段单位划分	页是信息的物理单位，页的大小是固定的，而且由系统确定	段是信息的逻辑单位,它含有一组意义相对完整的信息。段的长度是不固定的，由用户确定
地址	单一的线性地址空间	二维的，标识一个地址需给出段名和段内地址
共享和存取访问控制	在同一页面中包含共享的程序和私用的数据，使共享和存取访问控制困难	便于共享段逻辑上完整信息共享有价值，提高主存利用率；便于控制存取访问，段是逻辑上完整信息可根据各段信息决定存取访问权
动态链接	\	提供动态连接的便利，运行中不用的模块可以不连接调入，节省内存空间
动态增长	\	便于处理变化的数据结构段，可动态增长

3.5 段页式存储管理

- 段页式系统的基本原理是先将整个主存划分成大小相等的存储块（页架），把用户程序按程序的逻辑关系分为若干个段，并为每个段赋予一个段名，再把每个段划分成若干页，以页架为单位离散分配。
- 地址：由段号、段内页号和页内地址三部分组成

页表 + 段表（内容改为页表始址和页表长度）
- 地址变换机构：综合分页和分段的

3.6 虚拟存储器的概念

- 定义：虚拟存储器是具有请求调入功能和置换功能，能仅把作业的一部分装入内存便可运行作业的存储器系统，它能从逻辑上对内存容量进行扩充的一种虚拟的存储器系统。
- 实现方式
  - 请求分页系统：它是在分页系统的基础上，增加了请求调页功能和页面置换功能所形成的页式虚拟存储系统
  - 请求分段系统：它是在分段系统的基础上，增加了请求调段和分段置换功能所形成的段式虚拟存储系统
  - 请求段页式系统：它是在段页式系统的基础上，增加了请求调页和页面置换功能所形成的段页式虚拟存储系统
- 特征：离散性、多次性、对换性、虚拟性
- **普通分页存储管理的地址变换目前实际是采用动态重定位方式进行地址变换，理论上是可以采用静态地址重定位进行的；但是虚拟分页存储管理由于页面会调进调出，页面存放的物理块可能会变化，相当于程序在内存要移动，所以必须采用动态地址重定位**

3.7 请求分页存储管理

- 需在页表中再增加：状态位P、访问字段A、修改位M、外存地址
- 缺页中断机构：在请求分页系统中，每当所要访问的页面不在内存时，便要产生一缺页中断，请求OS将所缺页调入内存
- 地址变换机构：增加了产生和处理缺页中断，以及从内存中换出一页等功能
- 页面的分配和置换策略
  - 固定分配局部置换策略
  - 可变分配全局置换策略
  - 可变分配局部置换
- 页面分配算法
  - 平均分配
  - 按比例分配
  - 按照优先权的分配
- 页面调入策略：预调页策略、请求调页策略
- 页面调入的位置：一般未修改过从文件区调入，修改过，从对换区调入
- **页面置换算法**
  - **“抖动”现象：页面频繁调进调出，以致一个进程在运行中把大部分时间花费在完成页面置换的工作上**
  - **最佳置换算法**：选择那些永不使用的，或者是在最长时间内不再被访问的页面置换出去
  - **先进先出（FIFO）置换算法**：该算法总是淘汰最先进入内存的页面
    - 存在Belady异常现象：分配块增加，缺页次数增加

    - **最近最久未使用置换算法(LRU)**：该算法是选择最近最久未使用的页面予以淘汰，系统在每个页面设置一个访问字段，用以记录这个页面自上次被访问以来所经历的时间T，当要淘汰一个页面时，选择T最大的页面。
    - **最近未用算法(NRU)**：这是一种LRU的近似算法。该算法为每个页面设置一位访问位，将内存中的所有页面都通过链接指针链成一个循环队列。当某页被访问时，其访问位置“1”。在选择一页淘汰时，就检查其访问位，如果是“0”，就选择该页换出；若为“1”，则重新置为“0”，暂不换出该页，在循环队列中检查下一个页面，直到访问位为“0”的页面为止
    - 最少使用置换算法(LFU)
    - 页面缓冲算法
- **虚拟存储管理中的管理算法实际上是牺牲时间性能换取内存空间性能**
- **性能分析**
    - 在请求分页系统中，假设存储器的访问时间ma为100ns（一般为10ns～几百ns），缺页率为p，缺页中断时间为25ms，则有效访问时间就可以表示为：

      有效访问时间 = (1 - p) × ma + p × 缺页中断时间

      = 0.1 + 24999.9p

3.8 请求分段存储管理

- 需在段表中再增加：存取方式、访问字段A、修改位M、存在位P、增补位、外存地址
- 缺段中断机构：当进程所要访问的段未调入内存时，便由缺段中断机构产生一缺段中断信号，由缺断中断处理程序将所需的段调入内存
- 地址变换机构：与请求分页的类似
- 段的动态链接：在一个程序运行开始时，只将主程序装配好并调入主存，其它各段的装配是在主程序段运行过程中逐步进行的，每当需要调用一个新段时，再将该段装配好，并与主程序段链接

四、设备管理

4.1 设备管理的概述

- I/O系统组成：微机I/O、主机I/O
- 设备控制器
- **设备的分类：**
    - **块设备，有磁盘和磁带**
    - **字符设备，有交互式终端、打印机等**
- **I/O控制方式**
    - **程序I/O方式：**也称为忙-等待方式，即在CPU向设备控制器发出一条I/O指令启动I/O设备进行数据传输时，要同时把状态寄存器中的忙/闲标志busy置为1，然后便不断地循环测试busy。当busy=I时，表示该I/O设备尚未输入完一个字(符)，CPU应继续对该标志进行测试，直至busy=0，表示该I/O设备已将输入数据送入到I/O控制器的数据寄存器中，于是CPU将从数据寄存器中取出数据，送入内存的指定单元，接着，再启动去读下一个数据，并置busy=1。

    - **中断控制方式：**当某进程要启动某个I/O设备时，便由CPU向相应的设备控制器发出一条I/O命令，然后立即返回继续执行原来的任务。设备控制器便按照该命令的要求去控制I/O设备。此时， CPU与I/O设备处于并行工作状态
    - **DMA(Direct Memory Access)控制方式：**
        - 它作为高速的外围设备与内存之间成批的数据交换，但是不对数据再做加工处理，数据传输的基本单位是数据块，I/O操作的类型比较简单
        - 它需要使用一个专门的DMA控制器（DMAC）。DMAC中有控制、状态寄存器、传送字节计数器、内存地址寄存器和数据缓冲寄存器
        - 它采用盗窃总线控制权的方法，由DMAC送出内存地址和发出内存读、设备写或设备读、内存写的控制信号来完成内存与设备之间的直接数据传送，而不用CPU的干预
        - 仅在传送一个或多个数据块的开始和结束时，才需 CPU干预，整块数据的传送是在控制器的控制下完成的
    - **I/O通道控制方式：**在大、中型计算机系统中，普遍采用由专用的I/O处理机来接受CPU的委托，独立执行自己的通道程序来实现I/O设备与内存之间的信息交换，这就是通道技术
- 通道的类型：
    - 字节多路通道（共享）
    - 数组多路通道（独占）
    - 数组选择通道（独占）
- 缓冲技术的作用：提高CPU与I/O并行程度，缓解CPU 与I/O速度不匹配的问题
    - **缓冲区管理实际上就是牺牲内存空间来换取时间效率**
- **I/O软件的分层结构：**

  从高到低的次序分别是：用户层软件、设备独立性软件，设备驱动程序、中断处理程序

4.2 设备的分配

- 分配策略：独享、共享、虚拟
- **在通道系统和主机系统中，分配设备的同时，还要分配与之相连的控制器和通道。**
- 设备独立性：用户程序不直接使用物理设备名（或设备的物理地址），而只能使用逻辑设备名；而系统在实际执行时，将逻辑设备名转换为某个具体的物理设备名，实施I/O操作
- **SPOOLing技术：利用多道程序技术，设置输入输出进程来模拟过去的卫星机/外围机，在CPU的控制下实现类似过去的脱机输入输出**
    - **组成：输入进程、输出进程；输入井、输出井；输入缓冲区、输出缓冲区，井管理程序**
    - 避免等待，提高了并发程度，提高了系统吞吐量

4.3 磁盘I/O

- 磁盘的类型：固定磁头、移动磁头
- **磁盘访问时间：寻道时间 + 旋转延迟时间 + 读写时间/传输时间**
- **磁盘调度算法**
    - **先来先服务FCFS**
    - **最短寻道时间优先法SSTF**

        - **最短寻道时间优先算法存在饥饿现象**
    - **扫描法SCAN / 电梯算法：**磁头从一端向另一端移动，若在移动方向上发现有I/O请求，则响应；若移动方向上无I/O请求时，便返回，向另一端移动
    - **循环扫描法CSCAN：**磁头从一端向另一端移动，若在移动方向上发现有I/O请求，则响应；若移动方向上无I/O请求时，便返回另一端上有请求的最小磁道上
    - N_step_SCAN算法
    - FSCAN算法
- 磁盘高速缓存
    - **RAID优点：速度快（因为具有交叉并行存取能力），容量大（多个硬盘），可靠性高（可以恢复数据）**

五、文件管理

5.1 文件系统

- 文件

  UNIX 文件系统将文件分成四类：普通文件、目录文件、设备文件（特殊文件）和符号连接文件(Symbolic link)
- 文件系统的功能：从系统角度来看，文件系统是对文件存储器的存储空间进行组织、分配和回收，负责文件的存储、检索、共享和保护。从用户角度来看，文件系统主要是实现“按名取存”，文件系统的用户只要知道所需文件的文件名，就可存取文件中的信息，而无需知道这些文件究竟存放在什么地方
- 常见文件系统：FAT文件系统、扩展文件表系统、NTFS（NT文件系统）、S51K/S52K（sysv）、ext2（二级扩展文件系统）、HPFS（高性能文件系统、hpfs）、CD-ROM文件系统(iso9660)、UDF通用磁盘格式文件系统
- 文件操作：

  创建、删除、读、写、截断、设置读/写位置、

  打开、关闭、

  创建目录、删除目录、改变当前目录、对文件属性的操作等
- **磁带上的文件只能是连续结构的文件，也只能顺序存取，不支持直接存取（随机存取）**
- 用户程序读写一个文件的步骤：
    1. 用户程序通过文件系统调用read/write,进入操作系统
    2. 然后系统调用在调用对应文件系统的read/write(注意，不同文件系统的read/write是不一样的, 涉及文件的物理结构，物理块的设置等等)
    3. 调用设备驱动程序启动外设
    4. 外设完成工作产生中断（对中断控制方式/DMA/通道）或程序检测状态指导设备完成读写

5.2 文件的逻辑结构

- 从用户观点出发观察到的文件组织结构称为文件的逻辑结构
- **类型**
  - **字符流式文件：它是依次的一串字符流构成的文件**
  - **记录文件：是用户把文件内的信息按逻辑上独立的含义划分信息单位，每个单位称为一个逻辑记录（简称记录）**
- **记录文件有顺序、索引、索引顺序、直接、分区和堆文件几种**
  - 逻辑记录的读写：设置一个读/写指针，Rptr或Wptr，它指向记录的首地址

5.3 文件的目录和管理

- 目录项/文件控制块(FCB)
  - **基本信息**：文件名，文件在外存的地址信息和长度
  - **存取控制信息**
  - **使用信息类：**创建日期，修改日期等
- 多级目录
  - 好处：加快文件或目录检索的速度，允许文件重名

5.4 文件共享

- **基于索引节点的共享方式**
  - 好处：加快文件或目录检索的速度，其次是方便文件共享
  - **基于索引节点的文件共享不能跨越文件系统：**因为每个文件系统具有自己独立的索引节点结构，这些索引节点存储了文件的元数据信息，如文件名、权限、大小等。当文件共享跨越不同的文件系统时，不同文件系统的索引节点结构可能不兼容，导致无法正确地访问和解释文件的元数据信息
- **利用符号链接：**系统为共享的用户创建一个link类型的新文件，将这新文件登录在该用户共享目录项中，这个link型文件包含连接文件的路径名

5.5 文件的物理结构

- **类型**
  - **顺序文件：按照文件逻辑上连续的信息顺序存储到相邻的物理块中**
    - 优点：实现简单，存取记录较快。适合磁带
    - 缺点：增删不方便，需要移动物理块
  - **链接文件：是指将文件中的各个逻辑记录存放在不相邻的物理块中,通过物理块中的链接指针将它们连接在一起的文件形式**
    - 优点：避免了顺序文件要求连续分配存储空间的问题，消除了物理块的外部“碎片”，外存利用率更高；通过指针将物理块链接在一起，使得文件的逻辑记录顺序与外存中记录的物理放置完全独立开来，克服了外存连续分配不能适应文件增长和缩短的缺点

        - 缺点：顺序访问，不能实现随机存取；链接指针信息的存放需要存储空间，使得存储器可利用空间减少；存取时间变长；操作系统可能会由于系统软件或硬件发生故障，导致链接指针丢失或出错，最终引起文件内容丢失或出错
    - **索引文件：索引文件指文件中的各个记录可存储在不相邻的各个物理块中，系统为每个文件建立一张索引表,在索引表中顺序记录该文件存储的物理块号**
        - 优点：可以实现文件的随机存取；文件增删方便
        - 缺点：索引表需要存储空间；查找索引表需要时间
    - 直接文件：直接文件方式采用Hash函数将文件逻辑结构中记录的关键字与存储设备的物理块号直接建立联系，从而提高文件的访问速度
    - 顺序结构存取速度最快
    - **磁带只能是连续结构，磁盘可以是连续、链接、索引、直接（Hash）结构**
- 外存空间
    - **外存空间大小 = 物理块(簇)大小 × 物理块的数量**
    - **根据文件的物理结构、物理块的大小信息确定文件分配外存空间，会计算文件实际占用外存大小，反过来，通过设定最大文件，计算索引结构的索引层次等**
    - **外存空间的管理方法**
        - 空闲表法/空白文件目录法
            - 适合连续分配，索引分配和链式分配
        - 空闲链表法
            - 适合链式分配，索引分配
        - 位示图法
            - 适合连续分配、链式分配、索引分配。占用空间少
        - 成组链接法
            - 适合大型文件系统