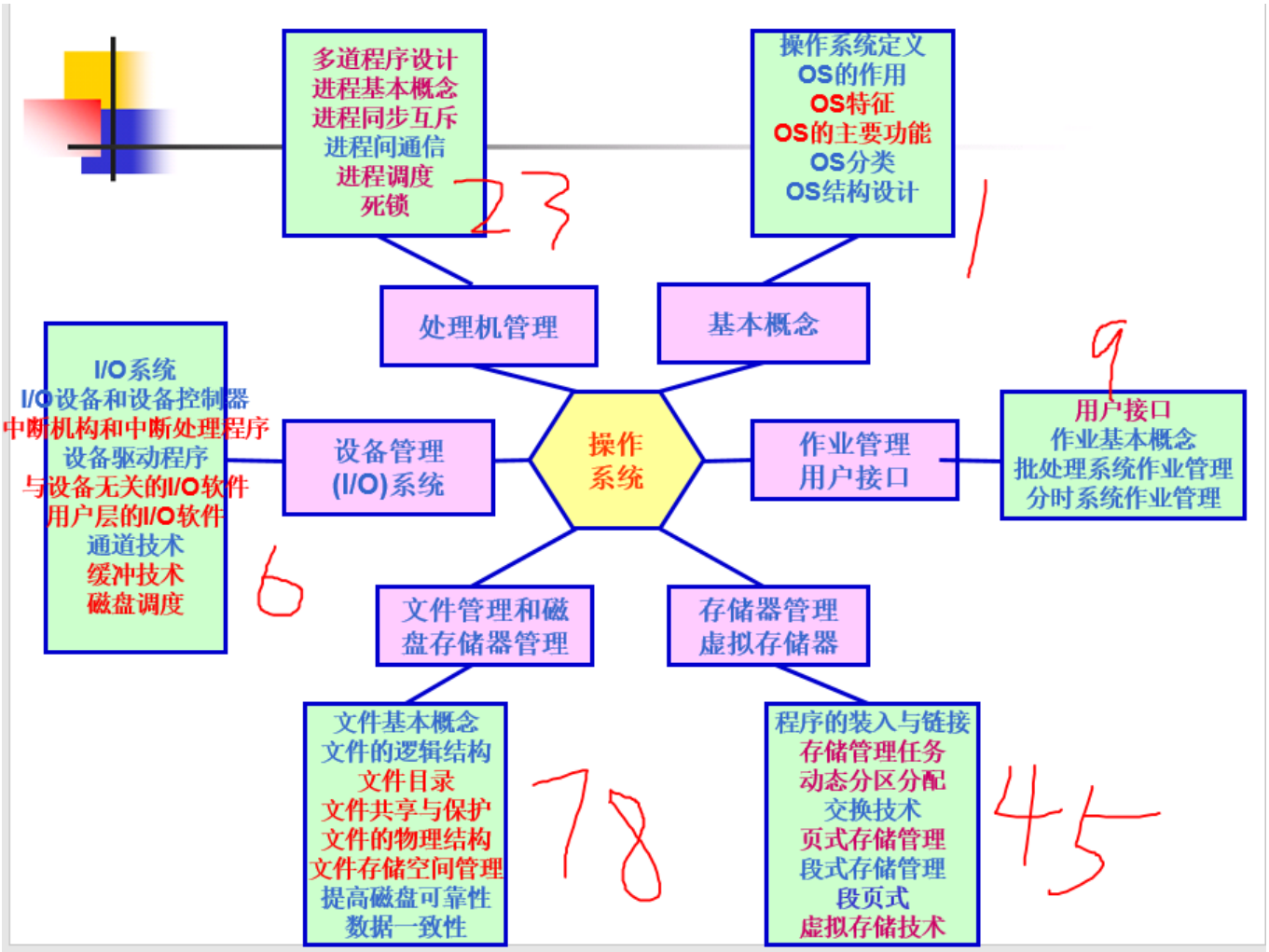
OS



一、操作系统引论

1.1 操作系统的目标和作用

1. OS的主要目标：有效性、方便性、可扩充性、开放性
2. os的作用
   - 作为用户和计算机硬件系统之间的接口
   - 作为计算机系统资源的管理者
   - 实现了计算机资源的抽象
3. 推动操作系统发展的主要动力：
   - 不断提高计算机资源利用率

    - 方便用户
    - 器件的不断更新换代
    - 计算机体系结构的不断发展
    - 不断提出新的应用需求

1.2 操作系统的发展过程

- 未配置操作系统的计算机系统
- 单道批处理系统
- 多道批处理系统
- 分时系统
- 实时系统

脱机I/O：
为了解决人机矛盾及CPU和I/O设备之间速度不匹配的矛盾。
事先将装有用户程序和数据的纸带装入纸带输入机，再一台外围控制机的控制下，把纸带卡片上的数据输入到磁带上。

单道批处理系统：为了解决人机矛盾和CPU与I/O设备速度不匹配矛盾的过程中形成的。缺点就是，系统中的资源得不到充分利用。

多道批处理系统：

- 优缺点：
    1. 资源利用率高
    2. 系统吞吐量大
    3. 平均周转时间长
    4. 无交互能力
- 需要解决的问题：
    1. 处理机争用问题
    2. 内存分配和保护问题
    3. I/O设备分配问题
    4. 文件的组织和管理问题
    5. 作业管理问题
    6. 用户和系统接口问题

分时系统：
关键问题：及时接收、及时处理
特征：多路性、独立性、及时性、交互性

实时系统：
硬实时任务和软实时任务：硬实时任务是指系统必须满足任务对截至时间的要求，否则可能出现难以预料的后果；软实时任务是指任务时间上执行并不严格，偶尔错过任务的截至时间，对系统的影响也不会会太大。

微机os的发展：
1.单用户单任务操作系统：CP/M、MS-DOS（IBM首先推出个人PC）
2.单用户多任务操作系统：windows95
3.多用户多任务操作系统：UNIX OS

1.3 操作系统的基本特性

- 并发性:
    - 并行和并发
    - 引入进程
- 共享性：

    - 互斥共享
    - 同时访问
- 虚拟性
    - 时分复用
    - 空分复用
- 异步性

1.4 操作系统的主要功能

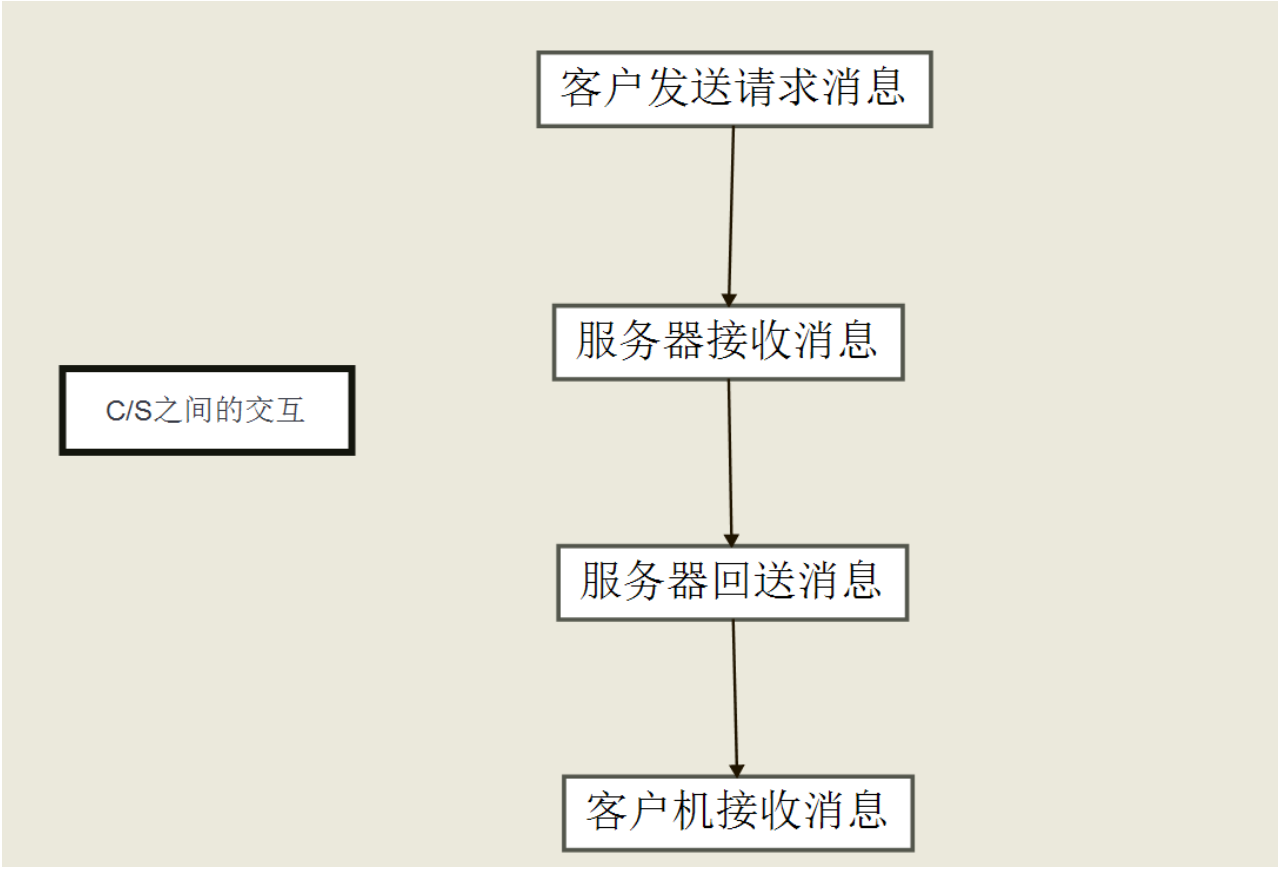
1. 处理机管理功能
    - 进程控制
    - 进程同步
    - 进程通信
    - 调度
2. 存储器管理功能
    - 内存分配
    - 内存保护
    - 地址映射
    - 内存扩充
3. 设备管理功能
    - 缓冲管理
    - 设备分配
    - 设备处理
4. 文件管理功能
    - 文件存储空间的管理
    - 目录管理
    - 文件的读/写管理和保护
5. 操作系统与用户之间的接口
    - 用户接口
    - 程序接口

1.5 OS结构设计

传统操作系统结构：无结构操作系统、模块化操作系统
客户/服务器模式：

- C/S模式的由来、组成、类型：客户机、服务器、网络

- C/S之间的交互：



- C/S模式的优点：
  - 数据的分布处理和存储
  - 便于集中管理
  - 灵活性和可扩充性
  - 易于改变应用软件
    面向对象的程序设计
    微内核OS结构：

1. 足够小的内核
2. 基于C/S模式
3. 应用“机制与策略分离”原理
4. 采用面向对象技术

微内核的基本功能：

1. 线程（进程）管理
2. 低级存储器管理
3. 中断和陷入处理

微内核优点：

1. 提高系统的可扩展性
2. 增强系统的可靠性
3. 可移植性强
4. 提供对分布式系统的支持
5. 融入了面向对象技术

二、进程的描述与控制

2.1 前趋图和程序执行

2.1.1 前趋图：有向无循环图
2.1.2 程序顺序执行
2.1.3 程序的并发执行

2.2 进程的描述

多道程序下，进程的执行是并发的，为了使参与并发执行的每个程序(含数据)都能独立运行，在操作系统中必须为之配置一个专门的数据结构，称为进程控制块(PCB)。系统利用PCB来描述进程的基本情况和活动过程，进而控制和管理进程。

进程：进程实体就叫做进程，由程序段、相关的数据段和PCB三部分便构成了进程实体(进程映像)。创建进程，实质上是创建进程实体中的PCB；撤销进程，就是撤销进程的PCB。**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**
进程的特征：动态性、并发性、独立性、异步性

2.2.2进程的基本状态和转换：

进程的三种基本状态：就绪、执行、阻塞

三种基本状态的转换：

创建状态和终止状态：
为了满足进程控制块对数据及操作的完整性要求以及增强管理的灵活性，引入了创建状态和终止状态：

1. 创建状态
   引入创建状态是为了保证进程的调度必须在创建工作完成后进行，以确保对进程控操作的完整性。同时，创建状态的引入增加了管理的灵活性，OS可以根据系统性能或主存容量的限制推迟新进程的提交(创建状态)。对于创建状态的进程，当其获得了所需的资源以及对其PCB的初始化工作完成后，便可由创建状态转入就绪状态。
2. 终止状态
   进程的终止要等到操作系统进行善后处理，最后将其PCB清零，并将PCB空间返还系统。

2.2.3挂起操作和进程状态的转换
为了系统和用户观察和分析进程的需要，引入了对进程的重要操作——挂起操作。与挂起操作相对应的是激活操作。挂起操作，意味着进程处于静止状态。
引入挂起操作的原因：

- 终端用户的需要
- 父进程请求
- 负荷调节的需要
- 操作系统的需要

引入挂起原语操作后三个进程状态的转换：
在引入挂起原语Suspend和激活原语Active后，在它们的作用下，进程将可能发生以下几种状态：

- 活动就绪（Readya）->静止就绪（Readys）
- 活动阻塞（Blockeda）—>静止阻塞（Blockeds）
- 静止就绪->活动就绪
- 静止阻塞->活动阻塞

> 当进程处于未被挂起的就绪状态时，称此为活动就绪状态，表示为Readya,此时进程可以接受调度。当用挂起原语Suspend将该进程挂起之后，该进程便转变成静止就绪状态，表示为Readys，处于Readys状态的进程不再被调度执行。

当进程处于未被挂起的阻塞状态时，称它是处于活动阻塞状态，表示为Blockeda。当用Suspend原语将它挂起后，进程便转变为静止阻塞状态，表示为Blockeds。处于该状态的进程在其所期待的事件出现后，它将从静止阻塞变为静止就绪Readys状态。

2.2.4 进程管理中的数据结构

为了便于对计算机资源中的各类资源(包括硬件和信息)的使用和管理，OS将它们抽象为相应的各种数据结构，以及提供一组对资源进行操作的命令，用户可利用这些数据结构及操作命令来执行相关的操作，而不关心其实现的具体细节。另一方面，操作系统作为计算机资源管理者，尤其是为了协调诸多用户对系统中共享资源的使用，它还必须记录和查询各种资源的使用及各类进程运行情况的信息。