3,4

资源稀缺。

1从什么角度看os。--定义

用户角度操作界面操作方式，用户环境。开发者，虚拟机，给予的资源和函数调用方式。资源管理者。作业组织者。

目标，方便性，高效性。

2发展

单批处理：内存中一个任务（作业） 监控系统--人工取代 问题：slow performance

cpu的发展solution：多道程序批处理

多批：cpu切换--作业调度的出现 早期多任务雏形，仍是操作员和程序员，无人机交互不灵活。

对os提出要求：io cpu的使用都有多个用户（是否可同时），多个设备资源分配--冲突问题；内存管理（不同机器环境不同方式）；cpu调度（考虑代价等问题）。

工作示例图（时间）。

人机交互：操作员输入数据时切换时死的。关机重启重新排序来做。

解决：分时系统（多任务系统）。作业装到内存中可能和外存切换（swap in、swap out），虚拟内存机制--本质目的：充分利用内存资源。直接的人机交互。用户访问数据非常方便。

dos是单任务，考虑市场成本。多任务在商业机有。命令行操作，无任务切换。

pc 键盘，显示器等配备的外设。用户界面等等考虑。

并行系统：紧耦合和松耦合两种。

紧耦合：多处理器，多核。如何和内存进行沟通--出现并行系统。增加吞吐量，经济，可靠度增加-不依赖于某一运算单元。

紧耦合分为对称和非对称。主处理器和协处理器（不同特征处理器做专属工作）的分别。

云计算，单位时间内干活的人多，吞吐量增大。

松耦合（分布式系统、网络系统）-云计算。加入即是组织成员，脱离仍能工作。客户端服务器端通过网络进行沟通。 安全性。

集群系统。

实时系统。及时反馈、快速。硬实时（只考虑实时性，达不到实时性要求则系统崩溃，军火导弹实时计算--微内核）和软实时（进程切换，响应时间等问题，用户体验）。

3发展规律

操作系统可裁剪可定制，可大可小。

作业-任务-进程。swap交换--内存扩展不同于切换。

4体系结构：硬件：cpu 内存 外设。cpu 和外设（端口地址）通过系统总线（数据、地址总线、控制总线（进出忙闲读写））、设备控制器交流。cpu和内存直接通过总线打交道。cpu中有指令队列，从内存取指令放入指令队列。外设控制器里的缓存当成local buffer。cpu管理外设的方式，典型为中断。

外设：同步方式，异步方式。硬件分级，硬件保护，四级，一般只用两级管态-内核态（0和3）（设备驱动，中断处理，硬件），用户态。用户发出对内核空间数据请求，用户是否等待。

3,9

和外设打交道：中断

两种I/O方法：同步/异步（关联度的时间节奏上，相互依赖与否）。系统级指令（特权级指令）/用户指令。

用户程序请求访问，通过操作系统来访问。同步时（资源利用率低，处理快），用户程序在wait。异步时，不等待该干嘛干嘛，之后再处理（有切换成本，处理复杂，增加并发度【等待队列，设备状态表，，】）。

线代操作系统的进程管理里多个线程，所以可以线程级切换。现代多任务多线程的多用异步。

* I/O技术：程序i/o，中断（现代os,重点），DMA（直接内存访问），Channel

程序i/o，也叫轮询：实现简单方便，占用cpu时间（很多时候白白查看空轮询，实时性取决于轮询时间，现在基本不用，除非数据特别有规律和定时）--有专门io访问指令，定期查看。

x86专门输入输出指令读取端口。（arm？）pad无特殊专门io指令（精简指令集），简单物美价廉。

中断：解决程序io问题。处理事件突发性。

软中断：中断指令，中断号，不同号不同处理程序。中断处理程序驻留在内存中，在内存中位置：中断程序的入口地址叫中断向量。中断号对应中断向量，将号排列成表，叫中断向量表。--8086的方式。0-ff最多256个中断向量，在内存最低地址空间，访问很快，加快响应。

在中断响应过程（尽量快）（还没调出中断处理程序）中不允许中断，然后再把中断打开，此时，中断处理程序和用户程序在多任务系统中并行执行。中断响应和中断处理是两个过程。

中断响应是？原子操作：不能再分。

除0操作，特殊中断处理。

不同机制和系统对中断处理不同，因为cpu和什么什么在发展。

中断处理器--硬中断。

cpu里脑细胞最重要：寄存器，一套但多个，越多则脑细胞越多。程序从S转到T时，记录下上下文。

寄存器越多，位越多cpu越好。位多则寄存器好好在哪？？

中断处理**上下文切换**，单核单处理器则当前程序停下。**上下文切换**保存当前状态：pc程序计数器（mostimportant），一套各个寄存器的值都要记录--保存到程序的栈里，即当前现场压栈。

内存里：程序段和数据段：动态区和静态区。堆栈。初始化和未初始化。，，？？

DMA:刚切换完又有中断请求，中断机制效率仍旧不高。DMA处理器：中断来了把剩下的工作直接交给DMA处理器，无论是io还是内存访问都交给DMA控制器。cpu解放数据传送工作。

Channel:比DMA更高级，相当于协处理器。DMA只是控制器。

* 存储为什么结构化？：工作分级。cpu主要做运算。系统设计时需要考虑的因素。

01位存储，读取速度speed；cpu设计制造的复杂度导致的成本；存储容量；volatility易失性，内存（靠电）断电数据不存在、外存（磁性）不易失。根据这些决定设计存储。

内部总线，外部总线，放在cpu内则通过内部总线更快。

cache：在cpu内，几级则增加复杂度，理论不变。寄存器不够，为了比访问memory加快一级速度。这是种思想，内存相当于磁盘的cache。访问磁盘需要调入内存。这个机制很多地方用到：123处。命中率问题？？系统结构讲。

寄存器的使用os无法干预。外存在硬件层面属于外设，软件层面属于文件系统。mainmemory是唯一一个能cpu访问的比较大的存储体。

从cpu角度，会把cache看成内存。cache和内存？管理是系统结构管辖范围（怎么映射？？），os站在系统结构上层。

程序对内存的需求在不断增加，因为功能越来越多。软件发展快于硬。外存以T，内存仍多数8G。save将内存中编辑的数据转入磁盘，断电无所谓。内存是灵活的线下，内存管理课时。从工作来说计算机只需要cpu和内存，无需人机交互或永久保存什么。

辅助存储器是对内存的扩展以来永久存储数据。没失磁啊什么，也有寿命。在外设讲。如何驱动磁盘？磁头磁臂扇区，，

数据一致性问题：对值更新一直要到magnetic disk的值更新完。数据是一级一级拷贝到不同存储级别。回写时出错则非一致。回写方式不同（只更新cache级，结束后再更新disk或同时更新完），一致性和速度的平衡。

cache和buffer机制一样。

3，11

数据一致性问题，也属于系统结构问题。

a<--b中间有个cache，因为a快b慢，程序有一定规律，把可能要用的代码放到cache，a直接从cache得到，得到叫做命中。不命中的代价等等。但整体提高的系统性能，平衡了ab速度差需求问题。

* hardware protection（硬件层面讲保护，ch3中软件层面进行保护）

单任务无此问题。多程序，要保证过程之间不能相互干扰。从系统层面，系统软件用户软件之间不能相互干扰。且用户程序是否合法可信，病毒软件。能够健壮稳定可靠地运行需要保护机制---1两种模式：系统模式，用户模式。2io保护3内存保护（各个程序有自己的地址空间）4cpu保护（权限）

1使用者复杂，需要对程序可信度判断。03，12没用。mode bit：0monitor 1user，两种状态切换图，privileged instructions、切换的权限只能是monitor。否则通过中断或陷进fault报错。

计算机网络的端口概念不同于这里的端口--io端口，一个device一套，端口地址就相当于device里面的register位置。

2io是privileged instructions。普通用户访问io如何调用--通过系统调用的方式来访问再把数据传送给你？

3基址寄存器和长度寄存器--硬件资源提供，简单的内存管理。作业进行时cpu都要判断地址是否在范围内，越界则报错。

内存的使用方式越复杂则对硬件机制要求越高。

4timer：每个任务什么时候开始用cpu可以用多久，超出时间释放资源--时间片概念。

否则如果死循环则cpu没有工作量了。

ch3操作系统体系结构

操作系统包含的模块、模块关系；设计角度提供什么的服务给用户；编程角度考虑os接口；普通用户的基本应用……

1component

外存管理，io系统管理，文件管理--3个有关联。

2process management

process是动态，running状态下的program。根本区别：将装到内存里，需要资源--由os决定装在哪以什么方式装入内存--内存管理里讲。

多个进程时的问题：cpu到底执行哪一，执行多长时间，怎么执行。进程和io打交道，和文件，都会需要解决问题。

作业：单道多道里概念，完成一个大的任务大的程序。

任务：非规范概念，可能指进程可能指线程。是：执行实体。

进程和线程概念比较规范。

进程管理相关内容：os负责进程创建（用户双击--命令发出，os要创建跟踪管理，用户把需求给出，os有各自方法来创建），结束进程，切换（挂起，唤醒），进程间通信机制(进程间是协作)，“死锁”，出错处理……

1控制2同步3通信4调度

3内存管理：资源稀缺--有使用也有释放，使用共享，地址划分单元。os负责内存分配，有相应记录--那些使用那些没使用，如何使用。管理也需要资源。在动态情况下必须对内存有精确记录机制。

单任务无此问题。并发执行共享内存，复杂度提高。

1分配2保护（系统和用户空间）3地址映射（物理地址：内存装载存储空间，逻辑：动态时？？）4内存扩展（虚拟存储技术）

4外存管理：用户决定，os以文件系统方式管理外存。12位置分配3磁盘调度（速度快点）

5io管理：速度，方式，特点类很多。HAL 硬件抽象层。123

直接io命令或者把映射到内存，？两种思路。

端口com口（地址是写死的，设置好的），设备管理器里。

提高处理机和io设备利用率--提高程度不大了。

1缓冲管理（串行并行通信？）2345

6文件管理：逻辑 文件名 文件属性 文件id

创建删除移动等操作

7保护：

现在网络应用重要非常，os必要支持network。

图形界面消耗资源，指令效率更高。

8基本服务和增加服务（分配资源，记录，保护）

支持高级语言的系统调用，比如copy文件

systemcall（os提供，非我们编程人员可控）：传参到寄存器（最快的方式，）；store 到内存的一个表格table，表格地址放在寄存器；push到栈，并由os pop off。

9systemcall里的communication：两种方式：消息传递（轻量级的消息，通过内核来ab间传递）和内存共享（进程ab有关联，一部分内存可能两者都可读写或一读一写）。

10apps：自带的浏览器，画板等等

11以上各种不同模块怎么组织？理想方式是计算机网络的层次模型（好在模块独立，不会相互牵扯关联），但未实现。

微内核：vxworks，硬实时非常著名。导弹，航天飞机等不同设备的二次开发。

12虚拟机 ：虚拟机上装另一个系统。用软件模拟硬件。