1. Video: <https://www.bilibili.com/video/av17676450/>
2. Teacher: 张英涛;
3. Ep 01: software->os->hardware; 系统软件, 应用软件; 系统软件最重要的是操作系统; 头颅和思想; 接口, 软件接口更多; 操作系统引论; 进程管理; 处理机的调度和死锁; 存储器管理; 设备管理; 文件管理; 什么是操作系统; UNIX, DOS, windows, Linux; 一组控制和管理计算机硬件和软件资源, 合理地对各类作业进行调度, 以及方便用户使用的程序集合; 方便性, 有效性, 可扩充性, 开放性; 方便性, 0/1化为语言; 有效性, 资源有效利用; 可扩充性, 适应硬件和网络, 层次性结构; 开放性, 遵循标准; 用户与硬件系统的接口, 软件接口, 命令方式, 系统调用方式, 图形窗口方式; 四类资源的管理者, 处理机, 存储器, 设备, 文件; 扩充机器, 裸机, 虚拟机; 单道批处理系统, 50年代; 多道批处理系统, 60年代; 分时系统; 实时系统; 自动, 顺序, 单道; 摩尔定律; 多道批处理系统; 内存中同时存放多个相互独立的程序; (2018-7-30)
4. Ep 02: 多道, 内存中同时存放多个相互独立的程序; 共享; 多道; 无序; 调度, 作业调度(外存->内存), 分配处理机; 资源利用高; 系统吞吐量大; 平均周转时间长; 无交互能力; 分时系统; 一台主机多终端; 分开时间片; 多路性; 独立性; 及时性; 实时系统; 实时控制; 实时信息处理; 并发性, 多个事件在同一时间间隔发生; 共享性; 虚拟性; 异步性; (2018-3-5)
5. Ep 03: 并发性(最重要); 并行性, 同一时刻; 并发性, 同一间隔; 单处理机系统, 宏观并发; 多处理机, 可能有并行; 两种资源共享的方式; 互斥共享; 同时访问方式; 虚拟, 把一个物理实体变为若干个逻辑上的对应物; 虚拟通过分时实现; 异步性, 程序以不可预知的速度向前推进; 时钟信号类似于节拍鼓点; 资源管理, 处理器, 存储器, 设备, 文件; 用户接口, 命令接口, 程序接口, 图形接口; 联机用户接口, 是否能和主机实时交互; 脱机则有一个预输入的过程; 命令+终端处理程序+命令解释程序; UNIX是分时系统; 联机命令, log in, log out; 命令解释程序, 操作系统的最高层; command.com, shell(外壳); 脱机用户接口, 批处理用户接口; JCL(作业控制语言)+作业说明书+命令解释程序; 程序接口, 一组系统调用; 系统调用, 一组能完成特定功能的子程序; 利用库函数调用系统功能; 类似于中断; 无结构, 模块化, 分层; 微内核; (2018-7-30)
6. ep 04: 进程管理; 进程的基本概念; 程序在并发环境中的执行过程; 资源分配和独立运行的基本单位; 程序的顺序执行; 顺序性, 封闭型(不受外界因素影响), 可再现性; 程序并发执行; 前趋图, 有向无环; 程序并发执行的特征; 间断性, 共享合作制约, 执行暂停执行; 失去封闭性, 资源状态由多程序改变; 不可再现性; 进程的五大特征; 结构特征, PCB(动态特征的集中反映, 常驻内存), 程序段(描述要完成的功能), 数据段(操作对象及工作区); 动态性, 最基本的特征, 进程的生命周期, 创建而产生, 调度而执行, 撤销而消亡; 并发性, 多个进程同在内存; 独立性, 独立运行, 独立分配资源, 样本空间, 样本点; 几何概型;独立接受调度; 异步性, 独立而不可预知; 进程定义, 进程实体的运行过程, 系统进行资源分配和调度的基本单位; 进程和程序的关系; 进程动态, 程序(菜谱)静态; 进程并行, 程序没有; 进程是竞争资源的基本单位; 一个程序对应多个进程; 就绪, 执行, 阻塞, 针对处理机; 就绪, 万事俱备, 只差CPU, 就绪队列; 阻塞状态, 由于某事件无法继续执行, 等待状态, 自己阻塞自己; 请求I/O, 申请缓存; (2018-7-31)
7. Ep 05: 进程的基本状态转换; 三种基本状态的转换; 执行只有一个, 单处理机; 就绪, 万事俱备, 只缺处理机; 阻塞; 就绪to执行, 进程调度; 执行to阻塞, I/O请求; 阻塞不能直接回到执行; 阻塞to就绪, I/O完成; 就绪不能到阻塞; 执行to就绪, 时间片用完; 挂起状态; 引入原因, 终端用户请求, 父进程请求, 负荷调节需要, 操作系统需要; 挂起状态, 静止状态; 非挂起状态, 活动状态; 执行to静止就绪, 挂起; 静止就绪to活动就绪, 激活; 活动就绪to执行, 调度; 活动就绪to静止就绪, 挂起; 执行to活动阻塞, I/O请求; 活动阻塞to静止阻塞, 挂起; 静止阻塞to活动阻塞, 激活; 活动阻塞to活动就绪; 静止阻塞to静止就绪; PCB, 程序段, 数据段; PCB是OS中最重要的记录型结构; OS用PCB对并发进程进行管理和控制; PCB是进程存在的唯一标志; PCB常驻内存; OS专门开辟PCB区将所有的PCB组成若干链表或队列; 结构体structure; PCB中的信息, 进程标识符(内部标识符, 唯一的数字编号, 外部标识符, 数字字母组成), 处理机状态(主要寄存器, 通用寄存器8-32, 指令寄存器, 下一条指令地址, 程序状态字PSW, 条件码系统态用户态中断屏蔽标志, 用户栈指针, 地址), 进程调度信息, 进程控制信息; 进程调度信息; 进程状态, 阻塞就绪; 进程优先级, 一个整数越大越高; 与调度算法有关信息; 事件, 阻塞原因; 进程控制信息; 程序和数据地址; 进程同步和通信机制; 资源清单, 除CPU以外的所需资源与易经分配资源清单; 链接指针, 本进程PCB所在队列的下一个地址; PCB的组织方式; 链接方式, 同一状态的PCB链接成队列; 索引方式; (2018-7-31)
8. Ep 06: 进程管理最基本的功能是进程控制; 进程的创建终止状态转变; OS内核实现; 进程图, 进程的家族关系, 任何一个进程都可以产生它的子进程; 引起创建进程的事件, 用户登录, 作业调度(外存选一个放进内存), 提供服务(打印计算), 应用请求(由自己创建自己的子进程); create原语创建一个新进程, 申请空白PCB, 为进程分配资源写进PCB, 初始化PCB(标识, 处理机状态, 资源), 将新进程插入就绪队列; PCB的初始化, 初始化标识信息, 初始化处理机状态信息, 初始化处理机控制信息; 引起进程终止的事件, 正常结束, 异常结束, 外界干预; 进程的终止过程; 根据内部标识检索PCB, 从中读出该进程状态; 若执行, 终止执行, 调度标志为真; 杀子孙; 归还资源; PCB移出队列或链表; 引起阻塞和唤醒的事件; 请求系统服务, 启动某种操作, 新数据还没到, 无新工作可做; 进程阻塞过程; 入口->保存CPU现场->置该进程状态->进入等待队列->转进程调度(CPU不能空闲); 进程唤醒过程; 入口->从等待队列中摘下被唤醒进程->置该进程为就绪态->进入就绪队列->转进程调度或返回; block和wakeup作用相反, 成对出现; (2018-7-31)
9. Ep 07: 进程的挂起; 挂起原语suspend; suspend的执行过程; 活动就绪to静止就绪, 活动阻塞to静止阻塞, 正在执行to重新调度; 进程的两种制约关系; 间接制约, 共享, 互斥; 直接制约, 合作, 同步; 互斥, 并发执行的多个进程由于竞争同一资源而产生的相互排斥的关系; 同步, 进程间共同完成一项任务时直接发生相互作用的关系; 执行时间上必须按照一定顺序协调进行; 临界资源, 一次仅允许一个进程使用的共享资源; 临界区, 每个进程中访问临界资源的那段程序; 进程必须互斥进入临界区; 访问临界区的循环进程描述; repeat, 进入区(检查临界资源是否能访问), 临界区, 退出区(将临界区标志设为未访问), 剩余区, until false; 同步机制遵循的原则, 空闲让进, 忙则等待, 有限等待, 让权等待(让出CPU); 信号量机制; 信号量是一种数据结构(整型, 数组); 信号量的值与相应资源的使用情况有关; 信号量的值仅由PV操作改变; (2018-8-1)
10. Ep 08: 整型信号量; 整型数, <=0不可用(红灯), >0可用(绿灯); P操作, wait; V操作, signal; wait(S); while s<=0 do no-op(循环检测, 等待); s=s-1(把其他信号量关在外面); signal(S); s=s+1; wait和signal是原子操作; 只要S<=0就不断测试, 不满足让权等待; 记录型信号量, 用得最多一种信号量; type semaphore=record; value: integer; L: list of process; end; s, value, array; s.value为资源信号量其初值, 当>=0时, 代表系统中某类资源的数目; s.value<0时, 代表等待使用资源的进程个数; wait, 申请一个单位资源; wait(s); Var S: semaphore; s.value = s.value-1; if s.value<0, then block(s, L); 阻塞队列不止一个, 有等待打印机的, 有等待磁带机的; signal操作, 释放一个单位资源; procedure signal(s); var s: semaphore; s.value=s.value+1; if s.value<=0, then wakeup(s, L); s.value>=0, 可用资源数量; s.value<0, 已经阻塞的进程数量; s.value初值为1, 只允许一个进程访问临界资源, 是互斥信号量; AND型信号量, 所需要的所有资源一次性全部分配, 使用完后一起释放; swait; if s1>=1 and sn>=1 then for i=1 to n, do si=si-1; else, 当发现第一个si<1就把该进程放入等待队列并将它的程序计数器置于swait操作的开始位置; ssignal, for i=1 to n, do si=si+1, 将所有等待si的进程由等待队列取出放入到就绪队列; (2018-8-1)
11. Ep 09: 用信号量实现互斥; var mutex: semaphore=1 (互斥的时候必然都是1); parbegin; process1: begin; repeat; wait(mutex), p操作, 申请资源; critical section; signal(mutex), 退出区, 释放资源; remainder section, 剩余区; until false; process2: begin; repeat; wait(mutex) critical section; signal(mutex) remaindersection; until false; 两个没有区别, 一模一样; wait和signal必须成对出现; 初始状态, mutex=1, 没有并发进程使用临界区; 申请成功, 进程使用临界区, 执行了一个wait语句p操作, mutex=0; 另一个进程也申请临界区, mutex=0, 申请失败, 放进阻塞队列; 释放资源, signal操作v, mutex=0; 减一之后大于等于0则成功; 经典的同步问题; 生产者-消费者问题; 输入时, 输入进程是生产者, 计算进程是消费者; 输出时, 计算进程是生产者, 打印进程是消费者; 制约关系, 不允许消费者从空缓冲中取出产品; 不允许生产者进程到一个满的缓冲区放东西; 用记录型信号量解决生产者消费者问题; 利用资源信号量empty和full, 资源信号量是做同步用的; mutex=1, empty=n, full=0; buffer[n]; (2018-8-2)
12. Ep 10: procedure, begin; repeat, procedure an item nextp, 生产下一个产品; wait(empty), 申请一个缓冲区; wait(mutex), 互斥信号量都设成mutex; buffer(in)=nextp; in=(in+1) mod n; signal(mutex), 释放缓冲池, 告诉消费者可以取了; signal(full); until false; end; 消费者进程; consumer begin; repeat; wait(full), 申请一段满的缓冲区; wait(mutex); nextc=buffer(out); out=(out+1) mod n; signal(mutex), 告诉生产者可以放了 ; signal(empty); consumer the item in nextc; until false; end; 每个程序中互斥的wait和signal必须成对出现; empty和full成对出现, 只不过处于不同的进程中; 唤醒阻塞进程的是它的合作进程; V(signal)操作可以颠倒, P(wait)操作不能颠倒; 哲学家进餐问题; 筷子是临界资源, 在一段时间内只允许一个哲学家使用; 用一个信号量表示一只筷子, 由五个信号量构成信号量组, 初值为1; 用记录型信号量解决哲学家进餐问题; repeat; wait(c[i]); wait(c[(i+1) mod 5]); eat; signal(c[i]); signal(c[(i+1) mod 5]); think; until false; 同时拿起左边, 则死锁; 至多四个人同时拿起左边的筷子; 仅左右均可用, 才允许进餐; 奇数号先做后右; 用AND型信号量解决哲学家进餐问题; array[0..4] = (1, 1, 1, 1, 1); repeat; think; swait(c[(i+1) mod 5], c[i]); eat; ssignal(c[(i+1) mod 5], c[i]); think; until false; 读者写者问题; 一个数据文件被多个进程读写; reader进程; writer进程; 保证一个writer进程与其他进程互斥地访问共享对象的同步问题; 信号量设置, Wmutex; 设置Readercount表示正在读的进程数目; Readercount是一个可被多个Reader进程访问的临界资源, 为它设置互斥信号量Rmutex; 仅当Readercount=0表示没有reader进程在读, reader才需要执行p操作; 若p操作成功, reader进程便可去读, 使Readercount+1; Readercount是普通的整型量, 不是信号量; readercount>0, 说明已有Reader进程在安全的读数据; (2018-8-2)
13. Ep 11: 进程通信; 进程通信是指进程之间的信息交换; 低级通信, 进程的互斥和同步, 交换的信息量非常少; 高级通信, 利用os提供的一组通信命令, 搞笑传输大量数据; 对用户透明; 高级通信分类; 共享存储器系统; 消息传递系统; 管道通信; 共享存储器系统; 共享数据结构的通信方式, 进程之间通过某种数据结构, 如缓冲池进行通信属于低级通信方式; 共享存储区通信方式, 为了传送大量信息, 在存储器中划出一块共享存储区, 进程可通过对共享存储区进行读写来实现通信; 消息传递系统; 信息交换的单位是消息或报文, 分成两种, 直接通信方式, 间接通信方式; 计算机网络中将消息称为报文; 直接通信方式; send(receiver, message); receive(send, message); send(p2, m1), p1调用; receive(p1, m1); 解决生产者消费者进程; repeat; produce an item in nextp; send(consumer, nextp); until false; repeat; receive(producer, nextp); consumer the item in nextc; until false; 间接通信方式; 进程之间的通信需要通过某种中间实体; 悄悄话是直接, 传纸条是间接; 信箱; 只允许核准的目标用户随时读取; 信箱的创建和撤销; 信箱创建原语; 信箱名字, 信箱属性, 共享者的名字; send(maibox, message); receive(mailbox, message); 信箱分类; 私用信箱, 公用信箱, 共享信箱; 私用信箱; 用户进程建立, 作为进程的一部分; 订牛奶订报纸的信箱; 进程结束时信箱也消失, PCB消失则进程消失; 公用信箱; OS创建; 提供给系统中的所有核准进程进程使用; 采用双向链路的信箱来实现; 共享信箱; 某进程创建, 提供共享进程(用户)名字; 拥有者和共享者都有权利从信箱中去走发送给自己的消息; 一对一, 专用的通信链路; 多对一, 客户/服务器; 一对多, 广播; 多对多, 公用; 管道通信; 利用共享文件来连接两个相互通信的进程; 写进程->管道->读进程; 管道通信必需的协调能力; 互斥, 一个进程读写管道, 另一个进程必须等待; 同步; 写入管道, 睡眠等待, 直到读进程取走数据唤醒; (2018-8-3)
14. Ep 12: 在原语操作过程中不允许中断; 线程, 用于多CPU操作系统和网络操作系统; 进程, 使多个程序并发执行, 以提高资源利用率和系统吞吐量; 引入线程, 是为了减少程序在并发执行时所付出的时空开销, 使OS具有更好的并发性; 引入线程的目的; 进程是可拥有资源的独立单位和可独立调度和分派的基本单位; 创建撤销和切换, 系统必须付出较大的时空开销, 所以进程不能过多, 切换频率不能过高; 线程的属性; 轻型实体, 线程中的实体基本上不拥有系统资源; 独立调度和分派的基本单位, 切换迅速开销小; 可并发执行; 共享进程资源; 操作系统是控制和管理各种硬件和软件资源, 有效组织多道程序运行的系统软件, 是用户与计算机之间的接口; 操作系统的基本职能是控制和管理系统内各种资源, 有效地组织多道程序的运行; 操作系统的基本特征是并发共享虚拟异步; 操作系统引入进程概念的主要目的是描述程序动态执行过程的性质; 从阻塞变为就绪; PCB是为所有进程; 进程和程序多对多; 同步, 互斥, 互斥; p操作, 减一; 小于零, 阻塞态, PCB连入末尾, 放弃处理机; 进程和程序的主要区别; 进程动态, 程序静态; 进程并发, 程序顺序; 进程独立; 进程和程序没有一一对应; 进程异步, 相互制约; P(mutex); apply for printer; use printer; signal(mutex); 一百个进程也是一样; (2018-8-3)
15. Ep 13: 处理机的调度和死锁; 三级调度; 高级调度, 作业调度, 从外存处于队列中的作业拿到内存创建进程分配资源排在就绪队列准备执行, 长程调度, 接纳调度; 分时系统实时系统不需要作业调度; 低级调度, 进程调度, 短程调度, 决定就绪队列中的哪个进程应获得处理机, 然后由分派程序把处理机分配给该进程的具体操作, 在OS中都必须配置; 中级调度; 调度方式, 非抢占方式, 抢占方式; 非抢占方式, 一旦分配, 一直执行, 直到完成或阻塞; 非抢占引起进程调度的因素; 进程执行完毕, I/O请求暂停执行, 执行了某种原语操作(wait, block, wakeup); 抢占方式, 允许暂停某个正在执行的进程; 抢占原则; 优先权原则, 优先权高的进程抢占处理机; 短作业优先原则; 时间片原则; 中级调度, 中程调度, 提高内存利用率和系统吞吐率; 暂时不能运行的进程暂时从内存调到外存, 又称对换, 挂起; 调度队列模型; 仅有进程调度的队列模型; FIFO队列, 排在就绪队列的末尾, 时间片轮转; 经常用于分时系统; 时间片内完成, 释放后进入完成; 时间片内未完成, OS把任务放入末尾; 遇到阻塞, 被OS放入阻塞队列; 用户->就绪队列->进程调度->CPU; 有高级和低级调度的队列模型; 同时具有三级调度的队列模型; 有高级和低级调度的调度队列模型; 就绪队列的形式; 批处理系统中最常用的优先权队列, 也可采用无序链表方式; 优先权越大越往前; 设置多个阻塞队列; 有三级调度队列模型; 调出时, 内存阻塞to外存阻塞, 内存就绪to外存就绪; 外存就绪to内存就绪; (2018-8-3)
16. Ep 14: 选择调度方式和调度算法的准则; 面向用户的准则, 周转时间段, 响应时间快, 截止时间保证, 优先权准则; 面向系统的准则, 系统吞吐量高, 处理机利用率好, 资源的平衡利用; 周转时间; 从作业被提交给系统开始, 到作业完成为止, 称为作业周转时间; 在外存后备队列等待调度的时间, 在就绪队列上等待调度的时间, 在CPU上执行的事件, 等待I/O操作完成的时间; 平均周转时间, T=1/n\*sigma(Ti); 带权周转时间; W=T/Ts, T作业的周期时间, Ts系统为它提供服务的事件, T>Ts; 平均带权周转时间; 提交是放在外存; 加上开始时间和结束时间; 周转=完成-提交; 带权周转=周转/运行时间; 平均周转时间, 平均带权周转时间; 响应时间; 从用户通过键盘提交请求开始直到系统首次响应为止的时间间隔, 从键盘到处理机的事件, 处理机对请求信息进行处理的时间, 响应信息回送到显示器的时间; 截止时间, 某任务必须开始执行的最迟时间, 或必须完成的最迟时间; 对于严格的实时系统, 调度方式和调度算法必须能保证这一点; 吞吐量, 单位时间内系统所完成的作业数; 评价批处理系统性能的重要指标; (2018-8-4)
17. Ep 15: 高级调度是作业外存进内存, 低级调度是内存中就绪队列选择一个进程, 中级调度是内存外存对换, 挂起; 调度算法, 资源分配策略, CPU(低级)和内存(高级); 不同的系统和系统目标, 通常采用不同的调度算法; 先来先服务算法; 选择最先进入该队列的作业调入内存, 分配资源创建进程放入就绪队列; 从就绪队列中, 选择一个最先进入该队列的进程分配处理机使之运行, 知道完成或阻塞, 才放弃处理机; 作业调度中每次从后被作业队列, 选择一个或多个最先进入该队列的作业调入内存, 为它们分配资源创建进程放入就绪队列; 进程调度每次从就绪队列, 选择一个最先进入该队列的进程分配处理机使之运行, 知道完成或阻塞后, 才放弃处理机; 最简单的调度算法, 可用于作业调度也可用于进程调度; first come first serve; 有利于长作业, 而不利于短作业; 有利于CPU繁忙作业, 不利于I/O繁忙作业(周转时间远远大于运行时间); 完成时间=开始+运行; 周转时间=完成时间-到达事件; 带权时间=周转时间/运行时间; 短作业(进程)优先法, 从后备队列选择一个或若干个估计运行时间最短的作业调入内存运行; 短进程有限法调度算法; 周转时间和带权周转时间都小, 有利于短作业, 不利于长作业, 将导致长作业长期不被调度; 不能保证紧迫性作业优先处理; 作业的长短是估计的; 高优先权优先调度算法; 优先权调度算法的类型; 优先权的类型; 高响应比优先调度算法; (2018-8-4)
18. Ep 16: 平均周转时间越短, 客户越满意; 带权时间越短, CPU利用率越高; 优先权调度算法类型; 非抢占式优先权算法; 把处理机分配给就绪队列中优先权最高的进程就一直执行下去; 用于批处理系统和某些对实时性要求不严的实时系统中; 抢占式优先权调度算法; 只要出现优先权更高的进程, 就重新将处理机分配给新到的优先权最高的进程; 能满足紧迫作业的要求, 常用于要求严格的实时系统, 以及对性能要求较高的批处理和分时系统中; 优先权的类型; 静态优先权, 在创建进程时确定, 在整个运行期间保持不变, 一般用某一范围的整数表示; 确定优先权依据; 系统高于用户, 资源要求少的进程较高优先权, 用户进程的紧迫程度及付费多少; 静态优先权的优缺点, 简单系统开销小, 不精确仅在要求不高的系统中使用; 动态优先权, 随进程推进或随等待时间的增加而改变, 以便获得更好的调度性能; 高响应比优先调度算法, 引入动态优先权, 并使作业优先级随着等待时间的增加而以速率a提高; 优先权=(等待时间+要求服务时间)/要求服务时间; 优先权=响应时间/要求服务时间; 分析, 作业等待时间相同, 则有利于短作业; 要求服务时间相同, 实现的是先来先服务; 长作业也可以获得处理机; 优点兼顾了长短作业, 缺点增加了系统复杂度; 时间片轮转法; 分时系统中采用时间片轮转法; 把就绪进程组织成FIFO队列; 把CPU分配给队首进程; 规定它执行一个时间片; 时间片执行完排在就绪队列的末尾, 重新把处理机分配给队首进程, 也执行一个时间片; 所有进程在一定时间内, 均可获得一个时间片的CPU时间; 多级反馈队列的调度算法, 为多个就绪队列赋予不同的优先级, UNIX系统使用; 第一个队列的优先级最高, 其余逐个降低; 各队列时间片也不同, 优先权越高时间片越小; 新进程进入内存, 放入第一队列末尾, FCFS等待调度, 如果未完成, 转入第二队列末尾; 执行一次, 优先权降低一点; 新进程抢占处理机, 原进程重新排入末尾; 多级反馈队列调度算法的性能; 终端作业用户, 作业小, 第一队列就能完成; 短批处理作业用户, 周转时间短; 长批处理作业用户, 不必担心作业长期得不到处理; 希望短作业优先安排, 长作业避免饥饿; (2018-8-5)
19. Ep 17: 实时调度的基本条件; 提供必要的信息, 就绪时间, 开始截止时间和完成截止时间, 处理时间, 资源要求; 系统处理能力强; 采用抢占式调度机制; 具有快速切换机制; 优先级; 系统的限制条件, 系统忙不过来; 缩短处理时间, 增加CPU; 快速切换机制, 对外部中断的快速响应能力, 响应禁止中断的时间间隔尽可能短; 实时调度算法分类; 硬实时调度算法; 软实时调度算法; 非抢占调度算法, 完成或阻塞或时间片用完; 抢占调度算法; 静态调度算法, 动态调度算法; 非抢占式调度算法, 简单, 用在小型实时系统或要求不严的实时控制系统; 非抢占式轮转调度算法, 用于工业群控系统, 一台计算机控制多个对象调度程序每次选择队列中第一个任务运行, 完成后挂在轮转队列的末尾, 等待下一次调度, 可获得数秒或数十秒的响应时间; 非抢占式优先调度算法, 针对有一定要求的系统; 为实时要求高的任务赋予较高的优先级, 优先安排在就绪队列队首, 待当前任务结束后, 被调度执行; 响应时间为数秒至数百毫秒; 抢占式调度算法; 应用于响应时间在数十毫秒以下的系统; 根据抢占发生时间不同分类, 基于时钟中断的抢占式优先权调度算法, 等待时钟中断到来时再重新分配处理机; 立即抢占的优先权调度算法, 只要当前任务不处于临界区就立即把处理机分配给它; 实时性能好不好取决于系统响应时间; 常用的实时调度算法; 最早截止时间优先算法EDF; 最低松弛优先算法LLF; (2018-8-5)
20. Ep 18: EDF算法; 根据任务的开始截止时间确定优先级, 截止时间越早优先级越高; 系统中保持一个实时任务优先级就绪队列, 调度程序选择对首任务分配处理机; 可采取抢占式和非抢占式调度; LLF算法; 根据任务的紧急程度确定优先级, 紧急程度越高优先级越高; 系统中保持一个实时任务优先就绪队列, 调度程序选择队首任务分配处理机; 可采用抢占式和非抢占式, 取决于响应要求; 多处理器系统MPS; 提高系统性能途径, 提高元器件的运行速度特别是处理器芯片的速度(材料和工艺), 改进计算机系统的体系结构特别是在系统中引入多个处理器或多台计算机; 多处理机系统类型, 紧密耦合MPS/松弛耦合MPS, 对称MPS/非对称MPS; 紧密耦合MPS, 通常通过高速总线或高速交叉开关实现多个处理器互联; 地址总线, 数据总线, 控制总线; 它们共享主存和I/O, 并要求将主存划分为若干个能独立访问的存储器模块, 以便多个处理器能同时对主存进行访问; 系统中所有的资源和进程, 都由操作系统实施统一的控制和管理; 松弛耦合MPS; 通常通过通道或通信线路实现多台计算机的互联; 每台计算机都有自己的存储器和I/O设备, 并配置了OS来管理本地资源和在本地运行的进程; 每台计算机都能独立工作, 必要时可通过通信线路与其他计算机交换信息; 对称多处理器系统; 系统中处理器单元在功能和结构上都相同, SR/6000 MODEL F50; 非对称多处理器系统; 系统中有多种类型的处理单元, 它们的功能和结构各不相同; 只有一个主处理器, 有多个从处理器; 进程分配方式; 多处理器系统中进程的调度与系统结构有关; 同构系统中进程可以分配到任一处理器上; 非对称系统中进程只能分配到某一合适运行的处理器上; 多处理器, 多计算机; 对称多处理器系统的进程分配, 把所有处理器作为一个处理器池; 由调度程序将任何一个进程分配给池中的任何一个处理器, 静态分配方式, 动态分配方式; 静态分配方式, 进程被固定分配到一个处理器上, 与单机进程调度方式相同, 优点开销小, 缺点各处理机忙闲不均衡; 动态分配方式, 设置一个公共就绪队列, 进程可被分配到任一处理器上, 优点消除了有的忙有的闲; 非对称MPS中的进程分配方式; 多采用主从式OS, 即OS的核心部分驻留在一台主机上, 从机上只是用户程序, 进程调度只由主机执行; 优点, 系统处理简单, 因为进程分配由主机独自处理, 使进程间的同步问题得以简化; (2018-8-6)
21. Ep 19: 实验一; 计算机执行的任何程序都必须首先读入内存, CPU通过内存来访问程序, 而安装操作系统实际上是把存放在光盘上的操作系统执行代码存入硬盘的过程; 从硬盘上加载程序到内存很方便; 操作系统中的文件系统主要是靠硬盘提供物理支持; 安装操作系统到硬盘, 在硬盘上建立文件系统, 把操作系统的全部内容事先存放在硬盘上备用; 当使用计算机时, 从硬盘上加载操作系统到内存, 然后将机器控制权转给操作系统内核来执行; 多重开机安装, 首先对硬盘进行分区; 实验2, 进程控制与描述; windows任务管理器的进程管理; 应用程序, 进程, 性能; windows 2000编程; 进程(线程)调度方式; 自调度方式; 成组调度方式; 专用处理机分配方式; 自调度方式; 自调度机制; FCFS, FPF和抢占式FPF都可以用; FCFS是一种好的调度算法; 整个系统中只设置一个就绪队列, 供多个处理器共享, 这些处理器必须互斥访问该队列; 自调度优点, 容易将单处理机环境下的调度机制移植到多处理机系统中, 处理器利用率高, 不会出现处理机空闲, 也不会出现有的忙有的闲; 自调度缺点, 瓶颈问题, 一个就绪队列必须互斥访问, 当处理器数目在数十乃至数百, 会产生严重的瓶颈问题; 低效性, 线程在生命周期中, 可能要多次更换处理器, 使高速缓存的使用效率很低; 线程切换频繁, 多个相互合作的线程很难同时获得处理器, 将会使某些线程阻塞, 被切换下来; 成组调度方式; (2018-8-6)
22. Ep 20: 成组调度方式; 将一个进程中的一组线程, 分配一组处理器上去执行; 可用两种方式分配处理器时间, 面向所有应用程序平均分配处理器时间, 面向所有线程平均分配处理器时间; 面向所有应用程序平均分配处理器时间, N个处理器和M个程序, 每个程序有1/M的时间去占有N个处理机; 成组优点, 减少线程切换, 总体性能优于自调度; 专用处理机分配方式, 专门为该应用程序分配一组处理器, 每一个线程一个处理器, 供应用程序专用直到应用程序完成; 缺点, 可造成单个处理机的浪费; 引入原因, 多处理机系统中单个处理机的利用率不是很重要, 在一个应用程序的运行中完全避免了进程或线程的切换, 大大加速运行; 死锁, 多个进程在运行过程中因竞争资源而造成的一种僵局; 各并发进程彼此等待对方拥有的资源, 且在得到对方资源前不释放自己的资源; 产生死锁的原因, 竞争资源, 进程间推进顺序非法; (2018-8-7)
23. Ep 21: 竞争资源引起死锁; 可剥夺和非可剥夺资源; 竞争非剥夺性资源; 竞争临时性资源; 可剥夺资源, 该资源可以被其他进程剥夺, 如CPU, 主存; 不可剥夺资源, 如磁带机, 打印机; 竞争非剥夺资源, 非剥夺资源的数量不能满足进程运行的需要, 会使进程在运行过程中, 因争夺这些资源而陷入僵局; I/O设备共享时的死锁情况, 资源分配有环路; 竞争临时性资源; 临时性资源, 进程产生的消息, 可能引起死锁; 画出资源分配图, 矩形资源, 圆形进程; 产生死锁的必要条件, 互斥条件, 请求和保持条件, 不剥夺条件, 环路等待条件; 互斥条件, 一段时间内某资源只由一个进程占有; 请求和保持条件, 进程已经保持至少一个资源, 但又提出了新的资源请求, 而该资源已被其他进程占有; 不剥夺条件, 进程已获得的资源, 在未使用完之前, 不能被剥夺, 只能使用完自己释放; 环路等待条件; 处理思索的基本方法; 预防死锁(严格条件限制), 避免死锁, 检测死锁, 解除死锁; 预防死锁, 设置某些限制条件, 破坏四个必要条件中的一个或几个, 简单容易实现; 避免死锁, 在资源动态分配过程中, 用某种方法去防止系统进入不安全状态, 可获得较高的资源利用率和系统吞吐量, 实现上有一定难度; 检测死锁, 允许死锁, 通过检测机构, 精确确定与死锁有关的进程和资源, 然后从系统中清除掉死锁; 解除死锁, 撤销或挂起一些进程; 预防死锁, 互斥条件不可以, 请求和保持条件可以, 不剥夺条件可以, 环路等待条件可以; (2018-8-7)
24. Ep 22: 抛弃”请求和保持”, 系统规定所有进程在开始运行之前, 都必须一次性地申请整个运行过程所需的所有资源; 优点, 简单安全易实现; 缺点, 资源被严重浪费, 恶化了系统的利用率, 使进程延迟运行; 抛弃不剥夺条件; 当一个进程已经保持了某些资源的进程, 再提出新的资源请求而不能立即得到满足时, 必须释放它已经保持了的所有资源, 待以后需要时再重新申请; 进程已经占有的资源, 在运行过程中会被暂时释放, 可被认为是剥夺; 缺点, 实现复杂代价大, 可能因为反复申请和释放资源, 而使进程的执行无限推迟, 延长了进程的周转时间增加系统开销, 降低系统吞吐量; 抛弃环路等待条件, 系统将所有的资源按类型进行线性排队, 并赋予不同的序号, 所有进程请求资源严格按资源序号递增的次序提出, 防止出现环路; 缺点, 序号必须稳定, 限制了新设备类型的增加, 作业使用资源顺序与系统规定的顺序不同而造成资源的浪费, 限制了用户编程; 系统安全状态, 系统能按某种顺序, 来为每个进程分配所需的资源, 直到最大需求, 使每个进程都可以顺序完成, 若需通不存在这样一个安全序列, 则称系统处于不安全状态; 系统进入不安全状态后可能进入死锁; 避免死锁的市值, 如何使系统不进入不安全状态; 只要能找出一个安全序列就行了, <P2, P1, P3>; 谁要求的少分配给谁, 谁归还的多优先安排给谁; (2018-8-7)
25. Ep 23: 银行家算法避免死锁; 阴阳家算法中的数据结构; 银行家算法, 安全性算法, 银行家算法举例, 银行家算法数据结构, 可利用资源向来Available(数组, 含有m个元素, 每个元素代表一类可利用的资源数目, 初值是该类全部可用资源的数目, 数值随该类资源的分配和回收而动态改变, available[j]=k, 表示系统中现有j类资源k个), 最大需求矩阵Max(n\*m的矩阵, 定义了n个进程中的每一个进程对m类资源的最大需求, 分配矩阵allocation, allocation[i, j]=K, 表示进程i当前分到j类资源的数目是k), 分配矩阵Allocation, 需求矩阵Need(n\*m矩阵, 表示每一个进程还要的各类资源数, 如need[i, j]=K, 表示进程i还需要j类资源k个); 三个矩阵关系, need[i, j] = max[i, j] - allocation[i, j]; Requence[j]=k; requence[i, j]<=need[i, j], 则转向步骤2, 否则error; 请求不能超过最大量; requence[i, j]<=available[j], 则转向步骤3, 否则等待; 系统试探着把资源分配给pi, available[j]=available[j]-requenst[i, j], allocation[i, j] = allocation[i, j] + request[i, j], need[i, j] = need[i, j] - request[i, j]; 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, 安全才分配, 否则让等待; 安全性算法, 设置两个向量, 工作向量work, 供进程继续运行的各类资源数; 工作向量work, 供进程继续运行的各类资源数; bool finish, 表示系统是否有足够的资源分配给进程, finish[i]=true; 寻找finish[i]=false, need[i, j]<=work[j]; 如果所有进程finish[i]=true, 则系统处于不安全状态; need=max-allocation; (2018-8-8)
26. Ep 24: 死锁的解除; 剥夺资源, 从其它进程剥夺足够数量的资源给死锁进程, 以解除死锁状态; 撤销进程, 杀死所有死锁进程, 按照某种顺序逐个撤销进程, 直到有足够资源可用; 存储器管理; 用户程序的主要处理阶段, 连续分配方式, 虚拟存储器的基本特征, 分页分段存储管理技术; 存储管理的功能, 内存分配(为每个进程分配一定的内存空间), 地址映射(把程序中所用的相对地秩转换成内存的物理地址), 内存保护(检查地址的合法性, 防止越界访问), 内存扩充(解决求大于供的问题, 采用虚拟存储技术); 程序的装入和链接, 编译阶段, 连接, 装入, 运行; 编译程序, 将用户源代码编译成若干个目标模块; 链接, 将一组目标模块及它们所需要的库函数链接在一起, 形成一个完成的装入模块; 装入, 将装入模块装入内存; 内存空间(物理空间, 绝对空间), 由内存一系列存储单元所限定的地址范围; 逻辑地址空间(相对空间), 由程序中逻辑地址组成的地址范围; 相对地址(逻辑地址), 用户程序经编译之后的每个目标模块都以0为基地址顺序编址; 绝对地址, 内存中各物理存储单元的地址是从统一的基地址顺序编址; (2018-8-8)
27. Ep 25: 程序的装入; 绝对装入方式, 可重定位装入方式, 动态运行装入方式; 绝对装入方式, 逻辑地址与实际地址相同, 要求程序员熟悉内存的使用情况, 通常在程序中采用符号地址; 可重定位装入方式, 目标模块从0编址, 其他地址相对于起始地址计算; 重定位, 装入时对目标程序中指令和数据的修改过程; 动态运行时装入方式, 在程序执行时将相对地址转换成为绝对地址, 允许程序在内存中移动; 程序的链接, 静态链接, 装入时动态链接, 运行时动态链接; 静态链接, 运行前形成就是静态, 执行前将目标模块和它们的库函数, 连接成一个完整的装配模块; 两个问题, 对相对地址进行修改, 变换外部调用标号; 运行时动态链接, 在执行过程中, 若发现一个被调用模块尚未装入内存, 由操作系统去找到该模块, 将它装入内存, 并把它链接到调用者模块上; 连续分配方式, 为一个用户程序分配一个连续的内存空间, 分为单一连续分配, 固定分区分配, 动态分区分配, 动态重定位分配; 单一连续分配, 内存分系统区和用户区, 系统区供os使用, 在内存中仅驻留一道程序, 整个用户区被一用户独占, 这种分配方式仅能用于单用户单任务os中, 如MS-DOS, CP/M; 多道, 内存中有多个程序; 固定分区分配, 将内存分为几个固定大小的区域, 每个区域装入一道作业; 划分分区的方法, 分区大小相等, 缺乏灵活性, 群控系统, 分区大小不等, 根据程序大小分配适当分区; 内存分配, 将分区按大小进行分配, 建立分区使用表, 表项包含分区的起始地址大小状态, 由小到大排列; (2018-8-8)
28. Ep 26: 动态分区分配; 数据结构, 分区分配算法, 分配与回收操作; 分区分配中的数据结构; 空闲分区表; 每个分区占一个表目, 包含分区序号, 分区起始地址, 分区大小; 空闲分区链, 在每一个分区的起始部分设置用于控制分区的信息向前指针, 在分区尾部设置一个向后指针, 形成双向链表; 分区分配算法; 首次适应算法FF, 循环首次适应算法, 最佳适应算法; 首次适应算法FF, 空闲分区链以地址递增的次序链接; 从链首开始顺序查找, 直到找到一个大小能满足要求的空闲分区为止, 再按作业大小, 从该分区中划出一块内存空间分配给请求者, 余下的空闲分区仍留在空闲链中; 内存利用率低, 低地址密集, 空闲区不均匀; 循环首次适应算法, 每一次从上次找到的下一个分区开始查找, 直到找到一个能满足要求的分区; 能使空闲分区均匀, 但缺乏大的空闲分区; 最佳适应算法, 将所有的空闲分区按容量从小到大形成一空闲分区链, 第一次找到的空闲区必然是最佳的; (2018-8-9)
29. Ep 27: 单一连续分配; 分区分配操作; 分配内存, 设请求的分区大小u.size, 空闲分区大小m.size, 不可再分割的剩余分区大小size, 如果m-u<=size, 将整个分区分配给请求者, 否则剩余部分留在空闲分区链表中; 将分区的首地址返回给调用者; 回收内存; 当进程释放内存时, 系统根据回收区的首值, 从空闲区链表中找到相应的插入点, 回收区可能出现四种情况, 与插入点的前一个空闲区F1相邻接, 与插入点后一个空闲区F2相邻接, 同时邻接, 都不邻接; F1/回收区->F1; 回收区/F2->回收区; F1/回收区/F2->F1; 动态重定位的引入; 碎片, 内存中不能被利用的小分区; 分散的小分区拼接成一个大分区的方法, 称为拼接或紧凑; 动态重定位, 地址变换过程是在程序执行期间, 随着对每条指令或数据的访问自动进行的; 增设硬件机构, 重定位寄存器; 静态地址是把逻辑地址变成物理地址; LOAD 1, 2500; 动态重定位分区分配算法; 增加了紧凑功能, 请求大小u.size, 空闲分区大小m.size; 对换; 把内存中暂时不能运行的进程调到外存, 把具备条件的进程调入内存; 对换分类; 整体对换, 以进程为单位, 又称进程对换; 部分对换, 页面对换, 分段对换; 对换空间的管理; 外存, 文件区/对换区, 为提高空间利用率采用离散分配方式, 为提高对换速度采用连续分配方式; (2018-8-9)
30. Ep 28: 进程换出过程; 选择处于阻塞状态且优先级最低的进程作为换出进程; 然后启动盘块, 将该进程的程序和数据传送到磁盘的对换区上; 对换区在外存上; 若传送过程未出现错误, 便可回收该进程所占用的内存空间, 并修改PCB; 进程换入过程, 从外存对换区调入内存; 系统定时查看所有进程的状态, 找出就绪但已经换出的进程, 将其中换出时间最久的进程作为换入进程, 将之换入, 直到再没有可换入的进程或无可换出的进程为止; 离散分配方式; 将一个进程分散装入不相邻的分区中; 分页存储管理方式, 分段存储管理方式; 基本分页存储管理方式; 不具备页面对换功能, 又称纯分页存储管理方式, 不支持虚拟存储器功能; 分页存储管理, 将一个进程的逻辑地址空间分成若干个大小相等的片, 称为页面, 并从0开始编号; 把内存空间分成与页面相同大小的若干个存储块, 称块或页框, 也加以编号; 以块为单位将进程中的若干个页分别装入到多个可以不相邻的物理块中; 块是物理, =页是逻辑; 页内碎片, 由于进程的最后一页进程不满而形成的不可利用的碎片; 页面大小, 通常1KB-8KB; 页面小, 页内碎片小, 提高内存利用率, 但页表过长, 占内存, 降低对换效率; 页号P, 位移量W; 逻辑地址A, 页面大小L, 页号P, 页内地址d; P=[A/L], d=A mod L; 页表, 页面系统为了能在内存中找到每个页面对应的物理块而为进程建立的一张页面映像表, 简称页表; 页表作用, 实现从页号到物理块号的地址映射; 页表在内存中; 第一次访问页表, 第二次访问内存; 表项中常设有存取控制字段; 内存分配, 连续, 离散; 离散, 将进程分散在内存不同位置; 分页, 分段; 基本分页管理(纯), 不具备对换功能; 页是逻辑地址划分片; (2018-8-10)
31. Ep 29: 存储器管理; 连续分配, 离散分配(分页, 分段); 地址变换机构; 页是大小相等的; 地址变换机构, 实现从逻辑地址到物理地址的转换, 将逻辑地址的页号, 转换成内存中的物理块号; 地址变换任务是借助页表(是个硬件)来完成的; 页表由一组专门的寄存器实现, 一个页表项用一个寄存器; 页表大多数驻留在内存中; 系统中只设置一个页表寄存器PTR, 存放页表在内存的始值和页表的长度; 一个进程一个页表; 进程未执行时, 页表的始值和页表长度存放在PCB中, 当调度到进程时装入页表寄存器中; 块表引入原因; CPU存取一个数据要两次访问内存; 第一次访问页表, 找到指定页的物理块号, 再将块号与业内偏移量W拼接形成物理地址; 第二次访问内存是从所得地址中获得所需数据; 为提高地址变换速度, 增设一个具有并行查询能力的高速缓冲寄存器, 称块表, 用于存放当前访问的页表项; 地址变换过程, CPU给出有效地址, 由地址变换机构自动将页号p送入高速缓冲存储器, 并将此页号与高速缓冲中的所有页号进行比较; 首先在块表中查询; 访问内存2或3次; 块表通常存放16-512个页表项; 大型作业只能将其中一部分页表项放入其中; 块表命中率90%; 如果找到, 20+100=120; 如果不能, 20+100+100=220; 有效访问时间T=h\*t1+(1-h)\*t2; 解决方法, 用离散分配方式解决难以找到连续的大内存空间的问题; 只将当前需要的部分页表项调入内存, 其余驻留在磁盘上; 两级页表, 为离散分配的页表再建立一张页表, 记录页表页面的物理块号; (2018-8-10)
32. Ep 30: 逻辑地址结构; 外层页号, 外层页内地址, 页内地址; 外部页号+外部地址寄存器=外部页表, 外部页内地址+外部页表=页表; 基本分段存储管理方式; 分段存储的引入, 分段系统的基本原理, 信息共享, 段页式存储管理方式; 分段存储的引入, 方便编程, 信息共享, 信息保护, 动态增长, 动态链接; 分段, 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, 每个段定义一组逻辑信息; 每个段都从零开始编址, 采用一段连续的地址空间, 段的长度由相应的逻辑信息组的长度决定, 因而段长不等; 整个作业的地址空间分成多个段, 是二维的; 逻辑地址结构, 段号, 段内地址; 段表, 系统为每一个进程建立一张段映射表, 简称段表, 用于实现逻辑段到物理内存的映射; 分段的地址变换; 分页和分段的区别; 页是信息的物理单位, 分页是由于系统管理的需要; 段是信息的逻辑单位, 分段是为了满足用户的需要; 页的大小固定且由系统决定, 段的长度不固定, 决定于用户所编写的程序, 通常由编译程序在对源程序进行编译时, 根据信息的性质来划分; 分页的作业地址空间是一维的, 程序员只需利用一个记忆符, 即可表示一个地址; 分段的作业地址空间是二维的, 程序员在标识一个地址时, 既要给出段名, 又需给出段内地址; (2018-8-11)
33. Ep 31: 可重入代码, 一种允许多个进程同时访问但不允许任何进程对它进行修改的代码; 编译器就是纯代码; 分页式信息共享; 段在内存中是连续存放的; 段页式存储管理方式, 既具有分段系统便于实现分段共享易于保护动态链接灯优点; 链接, 把分散的逻辑函数和库连在一起, 动态链接就是一边执行一边链接; 又能像分页系统解决内存的外部碎片问题, 以及可为各个分段离散分配内存等问题; 段页式存储管理的基本原理, 现将用户分成若干段, 再把每个段分成若干页, 并为每一个段赋予一个段名; 主程序段, 子程序段, 数据段; 段没有标号, 而是段名; 段页式地址结构, 段号, 段内页号, 页内地址; 段表寄存器; 三次访问内存; 第一次访问段表, 从中取得页表起始值; 第二次访问页表, 从中取出该页所在的物理块号, 并将该块号与页内地址一起形成指令或数据的物理地址; 第三次真正访问指令或数据; 地址变换机构; 内存容量不够带来的问题; 大作业要求的内存空间超过了内存容量不能全部被装入致使该作业无法运行; 其他大量作业留在外存上等待; 常规存储器管理方式特征; 一次性, 作业在运行前需要一次性全部装入内存; 驻留性, 作业装入内存后, 便一直驻留在内存中, 直到作业运行结束; 局部性原理, 一段较短时间内, 程序的执行仅限于某个部分而且所访问的存储空间也局限于某个区域; 局部性原理的几个论点; 程序多数情况是顺序执行的; 过程调用的深度大多不超过5, 程序将会在一段时间内都局限在这些过程的范围内运行; 循环结构虽然只由少数指令构成, 但是它们将多次执行; 程序包括许多对数据结构的处理; 时间局限性, 某指令一旦执行, 则不久后该指令可能再次执行, 某数据被访问过, 则不久后该数据可能再次被访问; 空间局限性, 程序在一段时间内所访问的地址, 可能集中在一定的范围之内; 虚拟存储器定义, 具有请求调入功能和置换功能, 能从逻辑上对内存容量加以扩充的一种存储器系统; 逻辑容量=内存+外存, 运行速度接近内存, 每位成本接近外存; (2018-8-11)
34. Ep 32: 分页请求系统; 在分页系统的基础上, 增加了请求吊页功能页面置换功能所形成的页式虚拟存储系统; 装入若干页的用户程序和数据, 便可启动运行; 再通过调页功能和页面置换功能, 陆续把即将要运行的页面调入内存, 同时把暂不运行的页面换出到外存上, 置换时以页面为单位; 分页请求系统硬件支持; 请求分页的页表机制, 缺页中断机构, 地址变换机构; 后10位是页内地址, 拼接(物理块号+页内地址); 请求分段系统; 在分段系统的基础上, 增加了请求调段及分段置换功能后, 所形成的段式虚拟存储系统; 允许只装入若干段的用户程序和数据, 即可启动运行; 再通过调段功能和段的置换功能, 将暂不运行的段调出, 同时调入即将运行的段, 置换是以段为单位进行的; 请求分段的硬件支持; 请求分段的段表机制, 缺段中断机制, 地址变换机制; 虚拟存储器特征, 多次性(分多次调入内存), 对换性(不用的调出来), 虚拟性(逻辑容量大于实际容量); 请求分页存储管理方式; 请求分页中的硬件支持, 内存分配策略和分配算法, 调页策略; 页表机制, 基本作用是将逻辑地址变换为物理地址, 在页表中再增加若干项, 供换进换出时参考; 状态位P, 用于指示该页是否已经调入内存, 供程序访问时参考; 访问字段A, 用于记录本页在一段时间内被访问的次数, 或记录本页最近已有多长时间未被访问, 供选择换出页面时参考; 修改位M, 该页调入内存后是否被修改过; 外存地址, 用于指出该页在外存上的地址, 通常是物理块号, 供调入该页时参考; 页号物理块号是最基本的; 缺页中断机构, 请求分页系统每当要访问的页面不在内存时, 便要产生缺页中断, 请求将所缺之页调入内存; 与一般中断的区别, 在指令执行期间产生和处理中断信号, 一条指令在执行期间, 可能产生多次缺页中断; 内存分配的三个问题; 最小物理块数, 能保证进程正常运行所需的最少物理块数; 最少物理块数与计算机的硬件结构有关, 取决于指令的格式功能和寻址方式; 物理块的分配策略, 固定分配局部置换, 可变分配全局置换, 可变分配局部置换; 固定分配局部置换, 为每个进程分配一固定页数的内存空间, 在整个运行期间都不改变; 如果缺页, 则只能从该进程的页面中选出一页换出, 再调入一页; 困难, 应该为每个进程分配多少个面的内存难以确定; 可变分配全局置换, 先为每个进程分配一定数目的物理块, os保持一个空闲物理块队列; 缺页时, 系统从空闲物理块队列中, 取出一个物理块分配给该进程, 并将想要调入的缺页装入其中; 当空闲物理块队列空时从内存中选择一页调出; 可变分配局部置换, 基于进程的类型或程序员的要求, 为每个进程分配一定数目的物理块; 缺页时从该进程的页面中选出一页换出; 如果进程频繁发生缺页中断, 则再为该进程分配附加的物理块; (2018-8-12)
35. Ep 33: 物理块分配算法, 平均分配算法, 按比例分配算法; 平均分配算法, 将系统中所有可供分配的物理块, 平均分配给各个进程; 按比例分配算法, 根据进程的大小按比例分配物理块; 考虑优先权的分配算法, 物理块分成两部分, 一部分按比例分配给进程, 另一部分根据各进成的优先权, 适当增加响应份额后, 分配给各进程; 有的系统可能是完全按优先权分配物理块; 调页策略; 何时调入页面, 从何处调入页面, 页面调入过程; 调页时机; 预调页策略, 将那些预计在不久之后便会被访问的页面, 预先调入内存; 主要用于进程首次调入; 请求调页策略, 绝大多数情况; 运行中要访问某程序和数据时, 若其页面不在内存, 立即提出请求, os调入; 从何处调入页面; 有足够对换区空间, 可全部从对换区调入所需页面; 无足够对换区空间, 不会被修改的文件从文件区调入, 会被修改的文件从对换区调入; UNIX方式, 未运行过的页面, 从文件区调入, 运行过被换出的页面从对换区调入; 页面调入过程; 程序发出缺页中断, 中断处理程序保留cpu环境, 转入缺页中断处理程序; 查页表, 求该页的外存物理块号; 如果内存未满, 则调入内存修改页表; 如果内存已满, 从内存中选出一页换出; 页面置换算法; 进程访问的页面不在内存而内存已无空闲空间时, 系统必须从内存中调出一页送磁盘的对换区中; 应将以后不会访问的页面或较长时间内不会再访问的页面调出; 抖动, 刚被换出的页很快又被访问, 需重新调入, 又需再选一页调出; 抖动导致进程在运行中, 把大部分时间花费在页面置换上; 最佳置换算法; 选择被淘汰页是永不使用的, 或者是在最长时间内不再被访问的页面; 可保证获得最低的缺页率, 无法实现; 可利用这个算法去评价其他算法; 有就不用置换了, 踢出最远的; (2018-8-12)
36. Ep 34: 先进先出页面置换算法; 淘汰最先进入内存的页面, 即选择在内存中驻留时间最久的页面淘汰; 新进来放在最底下; 最近最久未使用置换算法; 根据页面调入内存后的使用情况进行决策; 由于无法预测各页面将来的使用情况, 只能利用最近的过去作为最近的将来的近似, 选择最近最久未使用的页面淘汰; 该算法赋予每个页面一个访问字段, 用来记录一个页面自上次被访问以来所经历的事件T, 淘汰T值最大的; 简单clock算法; 每页设置一位访问位, 再将内存中的所有页面都通过链接指针连成一个循环队列; 当某页被访问时, 其访问位位置1, 淘汰时检查其访问位, 如果是0就换出, 若为1则重新归零; 再按FIFO算法检查下个页面; 又称最久未用算法; 最少使用置换算法; 为每个页面设置一个移位寄存器, 用来记录该页面被访问的频率; 选择在最近时期使用最少的页面作为淘汰页; 页面缓冲算法; 采用可变分配和局部置换方式; 如果页面未被修改, 就将它直接放入空闲链表, 否则, 放入已经修改页面的链表中; 页面在内存中并不做物理移动而只是将页表中的表项移到链表之中; (2018-8-13)
37. Ep 35: 请求分段存储管理; 请求分段的硬件支持; 分段的共享和保护; 段表机制; 缺段中断机制; 地址变换机制; 基本, 基地址, 段长; 存取方式, 表示本段属性, 只读; 访问字段A, 访问频率; 修改位M, 用于表示该页进入内存后, 是否被修改过, 供置换时参考; 存在位P, 指示本段是否已经调入内存, 供程序访问时参考; 存盘是存放在外存; 增补位, 特有的字段, 用于本段在运行过程中, 是否做过动态增长; 外存始值, 本段在外存中的起始地址, 即起始盘块号; 缺段中断处理过程; 请求分段地址变换过程; 共享段表; 共享进程计数count; 存取控制字段, 对于一个共享段, 应给不同的进程不同的存取权限; 段号, 对于一个共享段, 不同的进程可以用不同的段号去共享该段; 共享段的分配, 对第一个请求使用该共享段的进程, 为该段分配一个物理区把共享段调入该区, 同时将该区的起始地址填入该进程的段表中; 在共享段表中增加一表项, 把count设为1; 其他进程在使用该共享段时, 在进程的段表中, 增加一表项, 填入该共享段的物理地址; 共享段的回收; 取消在该进程段表中共享段所对应的表项; count=count-1; 若结果为0则系统回收该共享段的物理内存, 并取消共享段表中该段所对应的表项; 分段保护; 越界检查, 将逻辑地址空间的段号与段表长度进行比较, 如果段号大于等于段表长度, 将发出地址越界中断信号, 检查段内地址是否大于等于段长; 存取控制检查, 只读; 环保护机构, 低编号的环具有高优先权; 一个程序可以访问驻留在相同环或较低特权环中的数据; 一个程序可以调用驻留在相同环或较高特权环中的服务, 用户调用操作系统; (2018-8-13)
38. Ep 36: 设备管理; 设备管理的主要功能, 缓冲区管理, 设备分配, 设备处理, 虚拟设备, 实现设备独立性; I/O系统, 实现信息输入输出和存储的系统; 包括, I/O设备, 总线, 设备控制器, I/O通道, I/O处理机; I/O设备的类型; 可以按照重要的性能指标分类, 数据传输速率, 数据传输单位, 设备共享属性; 按传输速率分类, 低速设备, 每秒几字节至数百字节, ex 键盘鼠标语音输入输出; 中速设备, 每秒数千字节至数万字节, ex 行式打印机激光打印机; 高速设备, 每秒数百千字节以上, ex 磁带机磁盘机光盘机; 按信息交换单位分类; 块设备, 字符设备; 用于存储信息, 属有结构类型, ex 磁盘, 盘块大小512B-4KB; 磁盘设备的基本特征是传输速率比较高, 每秒钟几兆位, 可寻址, 即对它可随机读写任一块; 磁盘设备的I/O常采用DMA方式; 字符设备, 用于字符的输入和输出; 属无结构类型; 种类多, 如交互式终端打印机等; 基本特征是传输速率低, 通常为几个字节至数千字节, 另一特征是不可寻址, 即输入输出时不能指定数据的输入源地址及输出的目标地址; 输入输出时, 常采用中断驱动方式; 按设备的共享属性分类; 独占设备, 共享设备, 虚拟设备; 独占设备, 指在一段时间内只允许一个用户或进程访问的设备, 即临界资源; 共享设备, 指在一段时间内允许多个进程同时访问的设备, 如磁盘; 设备与控制器接口; 在该接口中有三种类型的信号, 数据信号; 控制信号, 设备控制器发送给设备的用于规定设备执行读写操作的信号; 状态信号, 用于指示设备的当前状态; 设备控制器, 是CPU与I/O设备的接口, 分成两大类, 用于控制字符设备的控制器, 用于控制块设备的控制器; 微型机和小型机中的控制器常做成印制电路卡形式, 称接口卡; 设备控制器的功能, 接受和识别功能, 数据交换, 标识并报告设备的状态, 地址识别(分辨多个设备), 数据缓冲, 差错控制; 设备控制器的组成; 设备控制器与处理机接口, 设备控制器与设备接口, I/O逻辑; 与处理机接口, 三类信号线, 数据线, 地址线, 控制线; 两类寄存器, 数据寄存器, 控制状态寄存器; 与设备接口, 一个接口连接一台设备, 每个接口中含有数据地址控制信号, 控制器的I/O逻辑根据处理机发的地址信号选择设备接口; I/O逻辑, 通过一组控制线与处理机交互, 处理机利用它向控制器发送I/O命令, I/O逻辑对收到的命令进行译码; I/O通道; I/O通道是特殊的处理机, 它具有执行I/O指令的能力, 并通过执行通道程序来控制I/O操作; 指令类型单一, 通道硬件比较简单; 通道没有自己的内存, 与CPU共享内存; 通道类型; 根据信息交换方式分三种类型, 字节多路通道, 数组选择通道, 数组多路通道; 字节多路通道, 含有非分配型子通道, 每个子通道连接一台I/O设备; 子通道按时间片轮转方式共享主通道; 第一个子通道控制其I/O设备完成一个字节的交换后, 便立即腾出字节多路通道, 让给第二个子通道使用, 所有通道轮转一周后重返回; 只要扫描通道速度快, 而设备速率小, 不丢数据; 连接低速或中速设备时, 不丢信息; 数组选择通道, 含有一个非分配型子通道, 可以连接多台高速设备; 一段时间内只能执行一道通道程序控制一台设备进行数据传送; 利用率很低; 数组多路通道, 含有多个非分配型子通道, 结合前两者的优点; (2018-8-14)
39. Ep 37: 瓶颈问题; I/O性能经常成为系统的瓶颈; CPU性能不等于系统性能, 响应时间也是一个重要因素; CPU性能越高, 与I/O差距越大; 弥补, 更多的进程; 进程切换多, 系统开销大; 通道价格昂贵, 使机器中的通道数量较少, 这成为瓶颈; 解决瓶颈最有效的方法, 便是增加设备到主机间的通路而不增加通道; 单通道I/O, 设备到主机只有一条路; 存储器, 通道, 控制器, 设备; 交叉连接, 多通道I/O; 总线, 将计算机系统中的各个子系统相互连接, 且连接是共享的; 优点, 低成本, 一线多用; 灵活性, 容易增加设备, 便于两个系统共享外设; 总线的缺点, 本身形成了通讯瓶颈, 限制I/O吞吐量; 总线分类, 数据地址控制; 总线系统, ISA, EISA, VESA, PCI; ISA总线, 为80286设计的总线结构, 带宽8位, 最高速率2MB/s; EISA总线, 带宽32位, 最高传输速率32MB/s, 能连接12台设备; 局部总线; VESA总线, 带宽32位, 最高速率132MB/s, 用于486微机中; 缺点, 连接的设备数2-4, 在控制器中无缓冲, 难适应处理器速度的提高, 不支持奔腾; PCI总线; 支持64位系统; CPU和外设间插入一复杂的管理层, 用于协调数据传输和提供一致的接口; 管理层中有数据缓冲, 将线路的驱动能力放大, 使PCI最多能支持10种外设, 并使高时钟频率的CPU能很好运行; 既可连接ISA, EISA等传统总线, 又可支持奔腾的64位系统; 缓冲的引入; 缓和CPU与I/O设备间速度不匹配的矛盾; 减少对CPU的中断频率, 放宽对CPU中断响应时间的限制; 提高CPU与I/O设备的并行性; 缓冲区设置; 硬缓冲, 在设备中设置缓冲区, 由硬件实现; 软缓冲, 在内存中开辟一个空间当缓冲区; 缓冲区管理; 单缓冲, 双缓冲, 循环缓冲, 缓冲池; 单缓冲; 块设备工作过程, 用户发出I/O请求, OS在主存中分配一缓冲区; max(c, t) + M; 单缓冲块设备工作过程; 单缓冲; 字符设备工作过程, 用户发出I/O请求, os在主存中分配一缓冲区用于暂存用户输入的一行数据; 输入期间, 用户进程被挂起以等待数据输入完毕; 输出时, 用户进程将一行数据输入到缓冲区后, 继续执行处理, 当用户已有第二行数据输出时, 如果第一行数据尚未提取完毕, 则用户进程阻塞; 双缓冲, 也称缓冲对换; 输入时, 将数据送入第一缓冲区, 装满后变转向第二缓冲区, 从第一缓冲区移出数据, 并送入用户进程, 接着由CPU对数据进行计算; max(C, T); 字符设备可消除用户等待时间; 发送缓冲区, 接受缓冲区; 为实现双向数据传输必须在每台机器上设置两个缓冲区; (2018-8-14)
40. Ep 38: 双缓冲块设备的工作过程; max(C, T); 循环缓冲的组成; 多个缓冲区, 多个指针; 多个缓冲区, 用于装入数据的空缓冲区R, 已经装满数据的缓冲区G, 计算进程正在使用的现行工作缓冲区C, 每个缓冲区的大小相同; 循环缓冲区的使用; Getbuf过程; Releasebuf过程; Getbuf(); 指针Nextg所指使的缓冲区给进程使用; Releasebuf(); 计算进程把C缓冲区的数据提取完毕时, 便调用Releasebuf过程, 将缓冲区G缓冲; 把该缓冲区由当前工作缓冲区C改为空缓冲区R, 当输入进程把缓冲区装满时, 也应调用Releasebuf过程, 将该缓冲区释放, 并改为G缓冲区; 即插即用技术, 插入就可以用, 不需要进行任何设置操作; PNP技术的产生; PNP技术的特点; 支持I/O设备及部件的自动配置, 使用户能够简单方便使用系统扩充设备; 减少由制造商装入的种种用户支持和限制, 简化部件的硬件跳接设置, 使I/O附加卡和部件不再具有人工跳接线设置电路; 在主机板和附加卡上保存系统资源的配置参数和分配状态, 有利于系统对整个I/O资源的分配和控制; 支持和兼容各种操作系统平台, 具有很强的扩展性和可移植性; 在一定程度上具有热插入热拼接技术; PNP技术的功能; 附加卡的识别与确认, 资源分配, 附加卡自动配置; 多方面的支持; 设备分配; 当某进程向系统提出I/O分配请求时, 设备分配程序按一定策略分配设备控制器和通道, 形成一条数据传输通路, 以供主机和设备间信息交换; 数据结构, 设备控制块DCB, 控制器控制块COCB, 通道控制块CHCB, 系统设备表SDT; 系统设备表SDT, 整个系统一张表, 记录系统中所有I/O设备的信息, 表目包括设备类型设备标识符进程标识符DCT表指针; 设备控制表DCT, 主要内容, 设备类型设备标识符设备状态COCT; (2018-8-15)
41. Ep 39: USB技术; USB通用串行总线, 是一种连接I/O串行设备的技术标准; USB技术突破了计算机发展的两个局限性; 由于I/O设备的接口标准的不一致和有限的接口数量无法满足各种应用的迫切需要; 传统的I/O设备的接口无法满足实时数据传输与多媒体应用的需求; USB是以WDM模型为基础, WDM包含一套通用的I/O服务和二进制兼容的设备驱动程序; USB支持同步数据传输方式和异步数据传输方式, 比标准串口快100倍, 比标准并口快10倍; USB可以主动为外部设备提供电源, 允许外部设备快速连接, 具有即插即用的功能, 允许外部设备的热插拔; USB结构; 控制器, 控制器主要负责执行由控制器驱动程序发出的命令; 控制器驱动程序, 控制器驱动程序在控制器与USB设备之间建立通信信道; USB芯片驱动程序; USB设备分类; USB集线器, 本身可以再连接其他USB外围设备; USB设备, 如鼠标; USB的传输方式; 等时传输方式; 麦克风, 音箱, 电话; 中断传输方式, 键盘鼠标游戏手柄; 控制传输方式; 批传输方式, 打印机, 扫描仪, 数码相机; 设备分配与回收, 当某进程向系统提出I/O请求时, 设备分配程序按一定策略分配设备控制器和通道, 形成一条数据传输通路, 以供主机和设备间信息交换; 设备分配策略; 考虑的因素, I/O设备的固有属性, I/O设备的分配算法, 设备分配的安全性, 与设备的无关性; 独占设备的分配, 就是临界资源; 要考虑充分发挥效率, 避免由于不合理分配策略造成死锁; 静态分配, 在程序运行前, 完成设备分配, 运行结束时, 收回设备; 缺点, 设备利用率低; 动态分配, 在进程运行过程中, 当用户提出设备要求时, 进行分配, 一旦停止使用立即收回; 优点效率好, 缺点分配策略不好时, 产生死锁; 共享设备分配; 要考虑多个访问请求到达时服务的顺序, 使平均服务时间越短越好; 一台可虚拟设备是可共享的设备, 可以将它同时分配给多个进程使用, 并对这些访问给设备的先后次序进行控制; 设备分配算法; 先来先服务; 优先级高优先; 设备分配中的安全性; 安全分配方式; 不安全分配方式; 设备独立性, 应用程序独立于具体使用的物理设备; 向用户屏蔽物理设备, 呈现给用户的一个操作简单的逻辑设备; 抽象的I/O操作; 设备驱动程序; 设备独立性软件; 执行所有设备的公有操作, 向用户层软件提供统一接口; 公有操作, 对独立设备的分配和回收, 将逻辑设备名映射为物理设备名, 找到响应物理设备的驱动程序; 对设备进行保护, 禁止用户直接访问设备; 缓冲管理; 统一接口, read, write; (2018-8-16)
42. Ep 40: spooling技术, 假脱机操作; 利用假脱机技术可把独享设备变成具有共享特征的虚拟设备; 一道输入一道输出; spooling系统的组成, 输入井和输出井(在磁盘上), 输入缓冲区和输出缓冲区(内存上开辟的), 输入进程SPi和输出进程SPo优点, 高速虚拟I/O操作, 实现对独享设备的共享; 打印机设备和可由打印机管理器管理的打印作业队列; 磁盘I/O访问时间的组成; 柱面定位时间, 旋转延迟时间, 数据传送时间; 合理组成磁盘数据的存储位置; 磁盘I/O调度, 不同进程的磁盘IO请求构成一个随机分布的请求队列, 磁盘IO调度的目标就是减少请求队列对应的平均柱面定位时间; 先进先出算法; 短查找时间优先算法; 扫描算法, 电梯算法; 循环扫描算法; N步扫描算法; 双队列扫描算法; (2018-8-16)
43. Ep 41: 文件管理是软件逻辑; 文件系统, 负责信息的组织存储和访问; 文件系统的功能是提供搞笑快速和方便的信息存储和访问功能; 基本概念; 文件, 具有符号名的数据项的集合, 文件名是文件的标识符号; 文件体, 文件本身的信息; 文件说明, 文件存储和管理信息; 文件系统是操作系统中管理文件的机构, 提供文件存储和访问功能; 目录是由文件说明索引组成的用于文件检索的特殊文件; 文件, 记录, 数据项; 文件系统的结构; 应用程序, 多种文件类型, 基本IO管理, 物理IO, 外部存储器; 文件管理的服务功能因素, 文件系统向上层用户提供的服务; 文件访问, 文件的创建打开关闭读写; 目录管理, 用于文件访问和控制的信息; 文件结构管理, 划分记录顺序索引; 限额, 限制每个用户; 审计, 记录对指定文件的使用信息, 保存在日志中; 文件系统的实现功能元素, 文件系统要实现的功能模块; 文件的分块存储, 与外存的存储块相配合; IO缓冲和调度, 性能优化; 文件定位, 在外存上查找文件的各个存储块; 外存存储空间管理, 分配和释放; 外存设备的访问和控制; 文件的组织, 讨论文件的内部逻辑结构, 主要考虑因素是文件存储性能和访问性能; 文件的组织, 文件的组织是指从用户观点出发讨论文件内部的逻辑结构或用户访问模式, 它可以独立于外存上的物理存储; 访问性能, 便于检索, 便于修改; 存储性能, 向物理存储转换方便, 节省空间; 文件的不同组织层次, 域记录文件; 文件区是离散存储; 文件的类型; 无结构文件, 文件体为字节流, 不划分记录, 顺序访问, 每次读写访问可以指定任意数据长度; 累积文件, 文件体为无结构记录序列, 通过特定分隔符来划分记录, 各记录大小和组成可变, 新纪录总是添加到文件末尾检索必须从头开始; 顺序文件, 文件体为大小相同的排序记录序列, 它由一个主文件和一个临时文件组成, 记录大小相同, 记录大小相同, 按某个关键字域排序; 索引文件; 记录大小不必相同, 不必排序; 建立索引, 每个索引指向一个记录, 索引项按照记录中的某个关键字域排序; 索引文件的记录项通常较小, 查找速度快, 便于随机访问; 索引顺序文件, 在顺序文件的基础上, 另外建立索引和溢出文件, 目的是加快顺序文件的检索速度; 可将关键字域中的取值划分若干个区间, 每个区间对应一个索引项; 通过划分层次, 在记录数量较大时, 比顺序文件大大缩短检索时间; 索引对顺序分组, 记录每一组的首地址; 哈希文件或直接文件; 文件目录, 目录是由文件说明索引组成的用于文件检索的特殊文件, 文件目录的内容主要是文件访问的控制信息; 目录的内容是文件属性信息, 其中一部分是用户可获取的; 文件名, 别名的数目, 文件类型; 地址信息, 存放位置, 包括哪个设备或文件卷volume, 以及各个存储块位置; 文件长度; 访问控制信息, 文件所有者, 访问权限; 使用信息, 创建时间, 最后一次读访问的时间和用户, 最后一次写访问的时间和用户; 目录结构类型, 一级目录, 二级目录, 多级目录; 一级目录, 整个目录组织是一个线性结构; 结构简单, 检索时间长, 有命名冲突; 二级目录, 根目录下, 每个用户对应一个目录; 多级目录, tree-like; 目录级别太多时, 会增加路径检索时间; 多级目录特点, 目录名, 目录树, 目录的上下级关系, 路径; (2018-8-17)
44. Ep 42: 改进的多级目录; 符号文件目录, 由文件名和文件内部标识组成的树状结构, 按文件名排序; 基本文件目录, 由其余文件说明信息组成的线性结构; 文件共享的实现, 各用户通过唯一的共享文件的路径名访问共享文件; 速度慢, 适用于不经常访问的文件共享; 利用多个目录中的不同文件名来描述同一共享文件, 即文件别名; 树状结构不允许共享; 基于索引结点的文件别名, 将目录内容分为两部分, 文件名和索引结点; 多个文件名链接到同一个索引结点, 可建立同一个文件的多个彼此平等的别名; 基于符号链接的文件别名; 特殊类型的文件, 内容是到另一个目录或文件路径的链接, 并不影响原文件; 文件和目录的使用, 文件访问, 文件控制, 目录管理; 文件访问; 打开open, 关闭close, 复制文件句柄dup, 读写指针, 执行exec, 修改文件的访问模式; 文件控制, 创建creat, 删除unlink, 获取文件属性stat, 修改文件名rename, 修改文件属主chown, 修改访问权限chmod; 目录管理; 进行文件访问和控制时, 由操作系统自动更新目录内容; 目录创建mkdir, 删除rmdir, 修改目录名rename; 外存存储空间管理, 文件存储空间分配, 空闲空间管理方法, 文件卷; 文件存储空间分配; 预分配; 动态分配; 文件存储单位, 簇cluster; 文件的存储空间通常由多个分立的簇组成, 每个簇包含若干个连续的扇区; 簇较大, 簇较小; 簇大小可变, IO性能好, 文件存储空间管理困难; 簇大小固定, IO性能下降, 文件管理空间开销大; 文件卷容量大, 簇内碎片多; 簇编号位数多; 外存空间空间管理; 磁盘分配表, 基本单位是簇; (2018-8-18)
45. Ep 43: 什么是操作系统; 操作系统五大功能; 并发, 共享, 异步, 虚拟; 三大基本类型; 分时, 实时; 进程的概念; 进程的特征; 进程有三种基本状态以及转换; OS给用户提供三种软件接口方式, 命令方式, 系统调用方式, 图形窗口方式; 两种资源共享方式, 互斥共享同时访问; 并行与并发的区别; 系统调用; 操作系统的发展阶段, 四代; 进程和程序的区别; PCB中的信息; PCB的组织方式, 链接方式, 索引方式; 进程管理中最基本的功能是进程控制; 进程控制的任务; 进程的创建终止, 引起阻塞和唤醒的事件; 进程唤醒过程, 有挂起状态的进程状态图; 进程的两种制约关系; 进程的两大关系, 同步互斥; 临界资源; 临界区; 同步机制遵循的原则, 空闲让进, 忙则等待, 有限等待, 让权等待; 用信号量实现互斥; 互斥信号量的初值固定是1; 记录型信号量解决生产者消费者问题; 用AND型信号量解决哲学家进餐问题; 进程通信分类; 进程通信分类, 低级通信, 高级通信; 高级通信分类, 共享存储器, 消息传递, 管道通信; 直接通信方式; 信箱分类; 引入线程原因(目的); 线程的属性; 三级调度, 每一级的作用; 进程调度的两种调度方式, 哪种系统开销大; 抢占原则, 3个; 调度队列模型, 3个; 面向用户的调度准则; 面向系统的调度准则; 周转时间, 带权周转时间, 响应时间, 吞吐量; 作业和进程调度算法有哪些; 先来先服务调度算法; 短作业(进程)优先法; 高响应比优先调度算法; 时间片轮转法; 多级反馈队列调度算法; 优先权的类型; 静态优先权, 动态优先权; 实时调度算法分类, 3种分类方式; 常用的实时调度算法, EDF算法, LLF算法; 多处理机系统类型; 紧密耦合MPS特点; 松弛耦合MPS特点; 对称MPS特点; 非对称MPS特点; 对称多处理器系统的进程分配方式; 死锁; 产生死锁的原因; 预防死锁, 避免死锁, 检测死锁, 解除死锁; 系统安全状态; 银行家算法避免死锁; 地址空间; 相对地址; 绝对地址; 程序的装入方式, 绝对装入方式, 可重定位装入方式, 动态运行时装入方式; 内存分配方式, 单一连续分配, 固定分区分配, 动态分区分配, 动态重定位分区分配; 分区分配算法, 首次适应算法FF, 循环首次适应算法, 最佳适应算法; 碎片; 动态重定位分区分配算法; 对换; 对换分类; 分页地址结构; 页表定义及作用; 块表引入原因; 分页和分段区别; 纯代码; 段页式存储管理基本原理; 段页式存储管理访问内存次数; 局部性原理, 局限性表现; 虚拟存储器定义; 虚拟存储器特征, 多次性, 对换性, 虚拟性; 请求分页存储物理块的分配策略; 固定分配局部置换, 可变分配全局置换, 可变分配局部置换; 页面置换算法, 最佳置换算法, 先进先出页面置换算法, 最近最久未使用置换算法, 简单CLOCK算法, 最少使用置换算法; 抖动; 请求分段存储器管理的段表结构; I/O设备类型, 数据传输速率, 数据的传输单位, 设备共享属性; 低速中速告诉; 块设备字符设备; 独占设备共享设备虚拟设备; 设备控制器的组成; I/O通道; 与一般处理机不同的两个方面表现; 通道类型, 字节多路通道, 数组选择通道, 数组多路通道; 解决IO瓶颈问题最有效的方法; ISA, EISA, VESA, PCI总线特征; 缓冲的引入原因; 设备分配算法, 先来先服务, 优先级高优先; 设备独立性; spooling技术, 一台设备虚拟成多台; spooling系统的组成; 磁盘IO调度算法; 先进先出算法, 短查找时间优先算法, 扫描算法, 循环扫描算法; 文件, 文件系统, 目录; 文件的类型, 无结构文件, 累积文件, 顺序文件, 索引文件, 索引顺序文件, 哈希文件或直接文件; 目录结构类型, 一级目录, 二级目录, 多级目录; (2018-8-20)
46. Ep 44: 能扩充主存容量的是分页虚拟存储管理; 分时就不能同时了; 分页式是解决碎片; 逻辑地址就是相对地址; 把逻辑地址转变为内存的物理地址的过程称作重定位; 引入进程概念的关键在于共享资源; 操作系统核心部分的主要特点是常驻内存; 系统调用是提供给编程人员的接口; 多道程序设计是指在一台处理机上并发运行多个程序; 进程间的相互制约关系体现为进程的互斥和同步; UNIX操作系统的最大特点是分时多用户操作系统; 一般的分时操作系统无法做实时控制用; 死锁是指两个或多个进程都处于互相等待状态而无法继续工作; 单用户多任务也是多道的; (2018-8-20)
47. // 计算机操作系统, 第四版;
48. Chapter 1: 操作系统引论;
49. Node 1.1: 操作系统的目标和作用;
50. 人机交互性, 实时性, 方便性;
51. List 1.1.1: 操作系统的目标;
52. 方便性, 使用高级语言或各种命令; 有效性, 提高系统资源的利用率, 提高系统的吞吐量; 可扩充性, 模块化结构, 层次化结构, 微内核结构; 开放性, 遵循开放系统互连OSI国际标准; (2018-7-25)
53. List 1.1.2: 操作系统的作用;
54. OS作为用户与计算机硬件系统之间的接口; OS作为计算机系统资源的管理者; OS实现了对计算机资源的抽象; 在裸机上覆盖一层I/O设备管理软件; 在第一层软件(I/O管理软件)上在覆盖一层用于文件管理的软件; (2018-7-25)
55. List 1.1.3: 推动操作系统发展的主要动力;
56. 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, 微机芯片, 外部设备; 计算机体系结构的不断发展, 多处理机系统, 网络系统; 不断提出新的应用需求, 嵌入式操作系统; (2018-7-25)
57. Node 1.2: 操作系统的发展过程;
58. List 1.2.1: 未配置操作系统的计算机系统;
59. 人工操作方式, 穿孔的纸带; 脱机输入输出方式, 减少了CPU的空闲时间, 提高了I/O速度;
60. List 1.2.2: 单道批处理系统;
61. 单道批处理系统的处理过程; 系统中的资源得不到充分的利用; 造成内存的浪费; (2018-7-25)
62. List 1.2.3: 多道批处理系统;
63. 多道程序设计的基本概念; 由作业调度程序按一定的算法, 从后备队列中选择若干个作业调入内存; 资源利用率高; 系统吞吐量大; 平均周转时间长; 无交互能力; 多道批处理系统需要解决的问题; 处理机争用问题; 内存分配和保护问题; I/O设备分配问题; 文件的组织和管理问题; 作业管理问题; 用户与系统的接口问题; (2018-7-25)
64. List 1.2.4: 分时系统;
65. 分时系统的引入; 为了满足用户对人机交互的需求; 分时系统实现中的关键问题; 多个终端, 用户命令能及时接收处理; 及时接收; 多路卡, 分时多路复用; 及时处理; 作业直接进入内存; 每次只能运行一个时间片; 分时系统的特征; 多路性, 允许多台终端同时连接到一台主机上, 并按分时原则为每个用户服务; 独立性, 彼此之间互不干扰; 及时性, 用户的请求能在很短时间内获得响应; 交互性, 通过终端与系统进行广泛的人机对话; (2018-7-25)
66. List 1.2.5: 实时系统;
67. 在规定的时间内完成对该事件的处理; 实时系统的类型; 工业控制系统; 信息查询系统; 多媒体系统; 嵌入式系统; 实时任务的类型; 周期性实时任务和非周期性实时任务; 硬实时任务和软实时任务; 实时系统与分时系统特征的比较; 多路性; 独立性, 互不干扰; 及时性; 交互性; 可靠性; (2018-7-25)
68. List 1.2.6: 微机操作系统的发展;
69. 单用户单任务操作系统; CP/M; MS-DOS; 单用户多任务操作系统; 允许用户把程序分成若干个任务, 使它们并发执行; 代表是windows; 多用户多任务操作系统; 代表是UNIX OS; (2018-7-25)
70. Node 1.3: 操作系统的基本特性;
71. List 1.3.1: 并发;
72. 并行与并发; 引入进程; 分别建立进程, 就可以并发执行; 进程, 系统中能独立运行并作为资源分配的基本单位, 由一组机器指令数据和堆栈组成; (2018-7-25)
73. List 1.3.2: 共享;
74. 互斥共享方式; 规定在一段时间内, 只允许一个进程访问该资源; 临界资源, 大多数物理设备, 栈变量表格; 同时访问方式; 磁盘设备; 并发和共享互为存在; (2018-7-25)
75. List 1.3.3: 虚拟;
76. 虚拟, 将一个物理实体变为若干个逻辑上的对应物的功能; 时分复用技术; 利用某设备为一用户服务的空闲时间, 又转去为其他用户服务; 虚拟机处理技术; 虚拟设备技术; 空分复用技术; 利用存储器的空闲空间; (2018-7-25)
77. List 1.3.4: 异步;
78. 异步运行方式是允许的; (2018-7-25)
79. Node 1.4: 操作系统的主要功能;
80. List 1.4.1: 处理机管理功能;
81. 进程控制; 创建进程, 撤销进程; 进程同步; 进程互斥方式, 锁W; 进程同步方式, 信号量; 进程通信; 实现相互合作进程之间的信息交换; 调度; 作业调度; 进程调度; (2018-7-25)
82. List 1.4.2: 存储器管理功能;
83. 内存分配; 分配内存空间, 提高存储器利用率, 允许申请附加的内存空间; 静态分配方式; 动态分配方式; 内存保护; 确保每道用户程序仅在自己的内存空间内运行; 决不允许用户程序访问操作系统的程序和数据; 设置两个界限寄存器, 分别用于存放正在执行程序的上界和下界; 地址映射; 将地址空间中的逻辑地址转换为内存空间中与之对应的物理地址; 内存扩充; 借助虚拟存储技术, 从逻辑上扩充内存容量; 请求调入功能; 置换功能; (2018-7-25)
84. List 1.4.3: 设备管理功能;
85. 完成用户进程提出的I/O请求, 提高I/O设备利用率; 缓冲管理; 设备分配; 设备处理; (2018-7-25)
86. List 1.4.4: 文件管理功能;
87. 文件存储空间的管理; 为每个文件分配必要的外存空间, 提高外存的利用率; 目录管理; 为每个文件建立一个目录项; 文件的读写管理和保护; 文件的读写管理; 检索文件目录, 利用读写指针; 文件保护; (2018-7-25)
88. List 1.4.5: 操作系统与用户之间的接口;
89. 用户接口; 联机用户接口, 一组键盘操作命令及命令解释程序; 脱机用户接口; 图形用户接口; 程序接口; 每一个系统调用都是一个能完成特定功能的子程序; (2018-7-25)
90. List 1.4.6: 现代操作系统的新功能;
91. 系统安全; 认证技术; 密码技术; 访问控制技术; 反病毒技术; 网络功能和服务; 网络通信; 资源管理; 应用互操作; 支持多媒体; 接纳控制功能; 实时调度; 多媒体文件的存储; (2018-7-25)
92. Node 1.5: OS结构设计;
93. List 1.5.1: 传统操作系统的结构;
94. 无结构操作系统; 模块化结构OS; 模块化程序设计技术的基本概念; 模块-接口法; 内聚性越高, 模块独立性越强; 耦合度越低, 模块独立性越好; 提高正确性可理解性可维护性, 增强可适应性, 加速开发过程; 接口规定很难满足实际需求; 无法寻找一个可靠的决定顺序; 分层式结构OS; 分层式结构的基本概念; 采用bottom-up法铺设中间层; 每一层仅能使用其底层提供的功能和服务; 分层结构的优缺点; 易保证系统的正确性; 易扩充和易维护性; 系统效率降低; (2018-7-26)
95. List 1.5.2: 客户服务器模式简介;
96. C/S模式的由来组成和类型; 客户机; 服务器; 网络系统; 客户服务器之间的交互; 客户发送请求信息, 服务器接受消息, 服务器回送消息, 客户机接受消息; 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, 易于改编应用软件; 存在着不可靠性和瓶颈问题; (2018-7-26)
97. List 1.5.3: 面向对象的程序设计;
98. 面向对象技术的基本概念; 对象; 对象类; 继承; 多重继承; 面向对象技术的优点; 通过重用提高产品质量和生产率; 使系统具有更好的易修改性和易扩展性; 更易于保证系统的正确性和可靠性; (2018-7-26)
99. List 1.5.4: 微内核OS结构;
100. Mach OS, windows 2000/XP; 微内核操作系统的基本概念; 足够小的内核; 与硬件处理紧密相关的部分, 一些较基本的功能, 客户与服务器之间的通信; 基于C/S模式; 应用机制与策略分离原理; 采用面向对象技术; 微内核的基本功能; 进程线程管理; 低级存储器管理; 中断和陷入处理; 微内核操作系统的优点; 提高了系统的可扩展性, 只需在服务器中增加新的功能; 增强了系统的可靠性; 可移植性强; 提供了对分布式系统的支持; 融入了面向对象技术; 微内核操作系统存在的问题; 运行效率有所降低; 好多次内核切换; (2018-7-26)
101. Chapter 2: 进程的描述和控制;
102. Node 2.1: 前趋图和程序执行;
103. List 2.1.1: 前趋图;
104. 偏序或前趋; 前趋图中不允许有循环; (2018-7-26)
105. List 2.1.2: 程序的顺序执行;
106. 程序的顺序执行; 程序顺序执行时的特征; 顺序性; 封闭性; 可再现性; (2018-7-26)
107. List 2.1.3: 程序并发执行;
108. 只有不存在前趋关系的程序才能并发执行; 程序的并发执行; 程序并发执行时的特征; 间断性, 执行暂停执行; 失去封闭性; 不可再现性; (2018-7-26)
109. Node 2.2: 进程的描述;
110. List 2.2.1: 进程的定义和特征;
111. 进程的定义; 进程控制块; 进程的特征; 动态性, 由创建而产生, 由调度而执行, 由撤销而消亡; 并发性; 独立性; 异步性; (2018-7-26)
112. List 2.2.2: 进程的基本状态及转换;
113. 进程的三种基本状态; 就绪状态; 执行状态; 阻塞状态; 三种基本状态的转换; 创建状态和终止状态; 创建状态; 终止状态; (2018-7-26)
114. List 2.2.3: 挂起操作和进程状态的转换;
115. 挂起操作的引入; 终端用户的需要; 父进程的请求; 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; 活动就绪, 静止就绪; 活动阻塞, 静止阻塞; 静止就绪, 活动就绪; 静止阻塞, 活动阻塞; 引入挂起操作后五个进程状态的转换; NULL->创建; 创建->活动就绪; 创建->静止就绪; 执行->终止; (2018-7-26)
116. List 2.2.4: 进程管理中的数据结构;
117. 操作系统中用于管理控制的数据结构; 资源信息表或进程信息表; 内存表, 设备表, 文件表, 进程表; (2018-7-27)
118. 进程控制块PCB的作用; 作为独立运行基本单位的标志; 能实现间断性运行方式, 系统就可将CPU现场信息保存在被中断进程的PCB中; 提供进程管理所需要的信息; 提供进程调度所需要的信息; 实现与其他进程的同步与通信, 信号量; (2018-7-27)
119. 进程控制块中的信息; 进程标识符; 外部标识符, 通常由字母数字组成; 内部标识符, 通常是一个进程的序号; 处理机状态, 由处理机的各种寄存器中的内容组成; 通用寄存器; 指令计数器; 程序状态字PSW; 用户栈指针; 进程调度信息; 进程状态; 进程优先级; 进程调度所需的其他信息; 事件, 阻塞原因; 进程控制信息; 程序和数据的地址; 进程同步和通信机制; 资源清单; 链接指针; (2018-7-27)
120. 进程控制块的组织方式; 线性方式, 适合进程数目不多的系统; 链接方式, 把具有相同状态进程的PCB通过PCB中的链接字链接成一个队列; 索引方式; (2018-7-27)
121. Node 2.3: 进程控制;
122. 创建新进程; 终止已完成的进程; 阻塞进程; 状态转换; (2018-7-27)
123. List 2.3.1: 操作系统内核;
124. 与硬件紧密相关的模块(中断处理程序), 各种常用设备的驱动程序, 运行频率较高的模块(时钟管理, 进程调度); 便于保护, 提高效率; 系统态; 用户态; (2018-7-27)
125. 支撑功能; 中断处理; 时钟管理; 原语操作; 资源管理功能; 进程管理; 存储器管理; 设备管理; (2018-7-27)
126. List 2.3.2: 进程的创建;
127. 进程的层次结构; UNIX, 进程家族; 进程图; 引起创建进程的事件; 用户登录; 作业调度; 提供服务; 应用请求; 进程的创建; 申请空白PCB; 为新进程分配其运行所需的资源, 包括各种物理和逻辑资源, 如内存文件I/O设备CPU时间; 初始化进程控制块PCB; 插入就绪队列; (2018-7-27)
128. List 2.3.3: 进程的终止;
129. 引起进程终止的事件; 正常结束; 异常结束, 越界错, 保护错, 非法指令, 特权指令错, 运行超时, 等待超时, 除零, I/O故障; 外界干预; 进程的终止过程; 检索PCB, 读出状态; 终止执行, 调度为真; 终止子孙; 归还资源; 移出队列; (2018-7-27)
130. List 2.3.4: 进程的阻塞和唤醒;
131. 引起进程阻塞和唤醒的事件; 向系统请求共享资源失败; 等待某种操作的完成; 新数据尚未到达; 等待新任务到达; 进程阻塞过程; 阻塞是进程自身的主动行为; 进程唤醒过程; wakeup, 从阻塞队列移出, 状态改为就绪, 插入就绪队列; block和wakeup作用相反, 成对使用; (2018-7-27)
132. List 2.3.5: 进程的挂起和激活;
133. 进程的挂起; suspend, 活动就绪改为静止就绪, 活动阻塞改为静止阻塞, 把PCB复制到内存区域, 转向重新调度; 进程的激活过程; active, 从外存调入内存, 静止就绪改为活动就绪, 静止阻塞改为活动阻塞, 比较优先级; (2018-7-27)
134. Node 2.4: 进程同步;
135. 硬件同步机制; 信号量机制; 管程机制; (2018-7-27)
136. List 2.4.1: 进程同步的基本概念;
137. 使并发执行的进程之间能按照一定的规则共享系统资源; 两种形式的制约关系; 间接相互制约关系; 直接相互制约关系; 临界资源critical resource; 生产者-消费者问题; 生产者进程和消费者进程互斥访问counter; 临界区critical section; 进入区, entry section; 退出区, exit section; 剩余区; 同步机制应遵循的规则; 空闲让进; 忙则等待; 有限等待; 让权等待; (2018-7-27)
138. List 2.4.2: 硬件同步机制;
139. 关中断; 利用test-and-set指令实现互斥; 利用swap指令实现进程互斥; 用于交换两个字的内容; (2018-7-27)
140. List 2.4.3: 信号量机制;
141. 整型信号量; 记录型信号量; AND型信号量; 将进程在整个运行过程中所需要的所有资源, 一次性全部地分配给进程, 待使用完后再一起释放; swait(simultaneous wait); 信号量集; 在每次分配前, 测试资源的数量, 判断是否大于可分配的下限值; (2018-7-28)
142. List 2.4.4: 信号量的应用;
143. 利用信号量实现进程互斥; mutex=1, 都未进入需要互斥的临界区; mutex=0, 一个进入临界区运行, 另外一个挂入阻塞队列; mutex=-1, 另外一个需要在退出时唤醒; wait(mutex)和signal(mutex)必须成对出现; 利用信号量实现前趋关系; P1, signal(S); P2, wait(S); (2018-7-28)
144. List 2.4.5: 管程机制;
145. 管程monitors;
146. 管程的定义; 管程, 代表共享资源的数据结构以及对该共享数据结构实施操作的一组过程所组成的资源管理程序共同构成了一个操作系统的资源管理模块; 管程的名称, 共享数据结构说明, 对该数据结构进行操作的一组过程, 设置初始值的语句; 管程的特性; 模块化; 抽象数据类型, 不仅有数据, 还有对数据的操作; 信息掩蔽; 管程和进程的不同; 进程是私有PCB, 管程是公共数据结构如消息队列; 进程是顺序执行, 管程主要是同步操作和初始化操作; 进程实现系统的并发性, 管程解决共享资源的互斥使用; 进程调用管程, 进程主动管程被动; 进程能并发, 管程不能和其调用者并发; 进程具有生命, 管程是一个资源管理模块; (2018-7-28)
147. 条件变量; x.wait, 正在调用管程的进程因x条件需要被阻塞或挂起; x.signal, 重新启动一个因x条件而阻塞或挂起的进程; (2018-7-28)
148. Node 2.5: 经典进程的同步问题;
149. List 2.5.1: 生产者-消费者问题;
150. 利用记录型信号量解决生产者-消费者问题; 利用AND信号量解决生产者-消费者问题; 利用管程解决生产者消费者问题; (2018-7-28)
151. List 2.5.2: 哲学家进餐问题;
152. 利用记录型信号量解决哲学家进餐问题; 利用AND信号量解决哲学家进餐问题; (2018-7-28)
153. List 2.5.3: 读者-写者问题;
154. 利用记录型信号量解决读者-写者问题; 利用信号量集机制解决读者-写者问题; (2018-7-28)
155. Node 2.6: 进程通信;
156. 进程间的信息交换; (2018-7-28)
157. List 2.6.1: 进程通信的类型;
158. 共享存储器系统; 基于共享数据结构的通信方式; 基于共享存储区的通信方式; (2018-7-29)
159. 管道(pipe)通信系统; 管道, 用于连接一个读进程和一个写进程以实现它们之间通信的一个共享文件, pipe文件; 互斥; 同步; 确定对方是否存在; (2018-7-29)
160. 消息传递系统(message passing system); 以格式化的消息为单位, 将通信的数据封装在消息中, 利用操作系统提供的通信命令进行消息传递; 利用OS提供的发送原语; 通过邮箱的方式进行消息的发送和接收; (2018-7-29)
161. 客户机-服务器系统; 套接字socket; 包含了通信目的地址, 通信使用的端口号, 通信网络的传输层协议, 进程所在的网络地址; 基于文件型套接字; 基于网络型; 不仅适用于同一台计算机内部的进程通信, 也适用于网络环境中不同计算机间的进程通信; 远程过程调用和远程方法调用; 一台主机系统上的进程调用另一台主机系统上的进程; 每一个能够独立运行的远程过程都拥有一个客户存根; (2018-7-29)
162. List 2.6.2: 消息传递通信的实现方式;
163. 直接消息传递系统; 发送进程利用OS所提供的发送命令, 直接把消息发送给目标进程; 直接通信原语; 对称寻址方式; send(receiver, message); receive(sender, message); 一旦改变进程的名称, 则可能需要检查所有其他进程的定义; 非对称寻址方式; send(P, message); receive(id, message); 消息的格式; 变长的消息格式; 进程的同步方式; send block, receive block; only receive block; no block; 通信链路; 显式的建立连接; 发送命令; 单项通信链路; 双向通信链路; (2018-7-29)
164. 信箱通信; 信箱的结构; 信箱头, 信箱标识符, 信箱的拥有者, 信箱口令; 信箱体, 由若干个可以存放消息的信箱格组成; 信箱通信原语; 邮箱的创建和撤销; 消息的发送和接收; send(mailbox, message), receive(mailbox, message); 信箱的类型; 私用邮箱, 进程自己建立; 公用邮箱, 操作系统创建; 共享邮箱; 一对一; 多对一, 客户/服务器; 一对多, 广播; 多对多; (2018-7-29)
165. List 2.6.3: 直接消息传递系统实例;
166. 消息缓冲队列通信机制中的数据结构; 消息缓冲区, int sender, int size, char \*text, struct message\_buffer; PCB中有关通信的数据项; 发送原语; ;;;;;;;;;;;;
167. -