3、4

选择：2 \* 15

填空：1 \* 15

简答：5 \* 5

问答：4个 共30

**Part 01：操作系统概述**

**-----------------------------------------------------------**

* 性质：系统软件
* 用户观点：与计算机的接口
* 应用程序员：协调多任务运行
* 功能：处理机管理、存储器管理、设备管理、信息管理
* 类型：实时操作系统、批处理操作系统、分时操作系统（兼有≥2项的为通用OS）
* 多道处理：将多个作业放入主存，同时运行并共享资源（硬件支持：通道和中断机构）
* 分时策略：时间片轮转
* 分时特征：多路性、交互性、独占性
* 不需要硬件支持的功能组成部分：进程调度
* 计算机系统 = 软件系统 + 硬件系统
* 基本职责：管理资源管理器、给用户提供一个管理平台
* 基本特征：并发 + 共享
* 性能参数：吞吐率 + 利用率
* 描述观点：虚拟机（极大扩展了裸机功能） + 资源管理
* 资源抽象：将现实硬件资源抽象成另一形式，可以使用户屏蔽底层细节，更好地使用P25

**-----------------------------------------------------------**

死锁避免方法、解决方法

**Part 02：体系结构部分**

**-----------------------------------------------------------**

* 作业：计算机系统按指定的步骤，为用户一次上机解题所完成工作的总和（组成：源程序、数据、作业处理说明，具体内容和具体步骤）
* 作业管理：用户提交作业后，OS对其的管理和控制（控制方式：控制卡、说明书）
* OS提供接口：命令（操作命令）、程序（系统调用命令）、图形（图形界面）
* 系统调用过程：
* 遇到调用命令
* 访管中断，处理机变管态，控制权交访管中断处理程序
* 按相应系统调用命令，通过地址变换找到相应的例行子程序入口去执行
* 退出中断、返回现场、继续执行

**-----------------------------------------------------------**

**Part 03：进程管理**

**-----------------------------------------------------------**

**进程：**

* 是一个具有一定独立功能的程序在一个数据集合上的一次动态执行过程
* 是分配资源的独立单位
* 组成：代码&数据（代码&数据&PCB）
* 特征：动态、并发、独立、异步、结构
* 状态及转换条件
* 顺序执行特征：可再现性、并发执行特征：资源共享
* 通信方式：直接、间接
* 创建：申PCB、赋pid、配空间、初始化PCB、链接到就绪队列。后随着作业运行结束而撤销
* PCB表大小：决定了系统中最多可同时存在的进程数，也叫系统的并发度
* 僵尸进程：子进程终止但父进程未调用wait()（父亲不知孩子已死，如同对待僵尸一样）
* 孤儿进程：父进程未调用wait()就终止（不先把孩子安顿好就离去，孩子变成了孤儿。Linux、UNIX将init（0号进程）作为孤儿进程的父进程）
* 关系：相交和无关（有无逻辑关系）、直接作用（同步，通信）和间接作用（互斥、共享）（直接和间接都属于相交）

**进程间通信：**

* 类型：低级（定长消息：状态、整数值）、高级（变长消息）
* 方式：消息传递、共享内存（共享数据结构：低级通信；共享存储区：高级通信）、管道通信（共享文件）、网络通信
* 消息传递：直接（消息队列）、间接（信箱）。有两个原语send、receive，实现不同
* 共享内存：内核仅参与创建共享内存区
* 管道通信：允许两个进程进行通信，实现时要考虑问题

**进程与程序区别：**

* 动态与静态：进程是在处理机上的一次运行过程；程序没有运行的含义
* 暂时与永久：进程有生命期；程序是永久的
* 内容不同：程序是指令的有序集合；进程包括代码、数据和PCB
* 同一程序可以运行于不同数据集合上，属于不同的进程；一个进程可以执行多个程序

**进程调度：**

* 类型：长程调度（形成作业队列）、中程调度（进程在内外存的交换）、短程调度（进程进CPU运行）
* 短程调度模式：非抢占式（简单、开销小、实时性差）、抢占式（优先级原则、时间片原则）
* 目的：最大化CPU利用率
* 进程工作方式：CPU和I/O交替进行（CPU或I/O burst cycle）
* 调度发生时刻（进程状态改变）：终止、运行-等待、运行-就绪、等待-就绪、创建-就绪（判断是否为抢占式：就绪队列变化时是否可以发生调度）
* 调度准则：系统角度（CPU使用率、吞吐量）、用户角度（周转时间、等待时间、响应时间）。本书为最小化平均等待时间
* 周转时间：进程提交到完成
* 等待时间：在就绪队列中等待的时间总和
* 响应时间：从提交到第一次响应的时间（是否为抢占影响最大）
* 调度算法：
  + FCFS（先到先服务）：护航效果（短的等在长的后面导致平均等待时间长）、非抢占、不适合分时系统
  + SJF（最短作业优先）：非抢占式（根据到来时间和长度排列组合）、抢占式（当新进程来到后，剩余burst时间最短的抢占）、适合长程不适合短程、实际上由于无法预测burst时间而无法实现
  + Priority Scheduling（优先级调度）：会有饥饿的问题（老化：时间↑，优先级↑）
  + RR（轮转调度）：时间片大小影响切换次数（大：退化为FCFS；小：切换次数增多）、轮转给每一个进程
  + Multilevel Queue（多级队列）：等待队列分成多队列，每个队列有自己的算法。进程可以在队列间转移（示例：q = 8、q = 16、FCFS）
* 多核调度：考虑亲和性（减少处理器间迁移开销）和负载均衡
* ！！！**实时CPU调度**：每个进程分配处理时间t、截止期限d、周期p（0≤t≤d≤p）
  + 单调速率调度（周期↓优先级↑）
  + 最早截止期限优先（截止期限早优先级↑）

**进程同步：**

临界资源：一次只允许一个进程使用的资源

临界区：进程中涉及到临界资源的程序段（进程其他部分叫剩余区）

控制原则：**互斥访问、有空让进、有限等待、让权等待**、无空等待、多中择一

**进程同步经典问题：**

**生产者-消费者问题：**

* empty：表示为空的资源数，初始为N
* full：表示为满的资源数，初始为0
* mutex：控制读写缓冲区的互斥量，初始为1

|  |  |
| --- | --- |
| **Producer** | **Consumer** |
| P(empty)  P(mutex)  one unit -> buffer  V(mutex)  V(full) | P(full)  P(mutex)  one unit <- buffer  V(mutex)  V(empty) |

**读者-写者问题（读者优先）：**

有读者在读，写者等待，但此时新读者可以进行读，写者接着等待

最后一个读者完成访问后，发现写者在等，则唤醒其中一个写者

* readcount：记录当前访问数据的读者数量，初始为0
* rmutex：控制各读者互斥地修改readcount
* wmutex：控制读写进程之间地互斥

|  |  |
| --- | --- |
| **reader** | **writer** |
| P(rmutex)  readcount++  if (readcount == 1) then P(wmutex)  V(rmutex)  read  P(rmutex)  readcount--  if (readcount == 0) then V(wmutex)  V(rmutex) | P(wmutex)  write  V(wmutex) |

**哲学家就餐问题：**

* chopstick：表示每根筷子是否被使用地互斥信号量数组，共4个，初始均为1

|  |
| --- |
| **philosopher** |
| P(chopstick[i])  P(chopstick[(i + 1) % 5])  eat  V(chopstick[(i + 1) % 5])  V(chopstick[i])  think |

**原语：**

是一定会有结果且不会被打断的操作

**P、V原语，操作一个信号量S，初值为n：**

* 是低级进程通信原语，管理和控制进程
* S种类：公用（控制进程间互斥，初始为1）、私用（控制进程间同步，初始为0或n）
* S含义：可用资源的数目（>0），因请求该资源而被阻塞的进程数（<0）
* 可以解决一切互斥问题（理论上！！！在复杂问题上实现起来很麻烦）
* 根据上下文推断变化范围
* 含有等待进程队列操作的P、V实现方法
* **完成例题**

**死锁：**

* 必要条件：
  + 互斥（资源一次被一个进程占有）
  + 占有并等待（进程A占着资源A并等着资源B，而资源B被进程B所拥有）
  + 非抢占（资源不能被抢占）
  + 循环等待（一个等另一个占有的资源，形成闭环）
* 资源分配图：
  + 结点：进程、资源
  + 边：申请边（进程指向资源，只指到边框）、分配便（资源指向进程，从内部圆点出发）
  + 调整：申请资源时加申请边；得到满足时转换为分配边；进程不再需要资源时删除分配边
* 性质：环的存在是死锁的**必要条件**
* 处理方法：
  + 通过协议预防：确保4个必要条件不出现，开销大
  + 避免：检测到非安全状态时，将系统恢复到安全状态（资源分配图算法、银行家算法）
  + 允许死锁出现并检测恢复：
    - 检测：将分配图的资源结点去掉并合并边得到等待图，等待图出现环是死锁出现的**充要条件**；
    - 恢复：可以通过终止进程恢复，**中止所有进程**（死锁进程可能已经计算许久却放弃了，开销大）或**一次中止一个进程**直至打破死锁的等待循环（每中止一个就要调用死锁检测算法，开销相当大）
  + 忽视（多数OS使用）

**线程：**

* 是调度的基本单位
* 被进程所拥有，不能独立存在
* 共享同一进程的资源
* 多线程优点：响应性（部分阻塞或有冗长操作时仍可运行）、资源共享（共享所属进程的资源）、经济（创建、切换线程开销小）、可伸缩性（线程可以在多核上运行）
* 分用户和内核线程，关系模型：
  + 多对一（线程库管理，效率高；一个线程阻塞，其他都阻塞，没有实际意义上的并行）
  + 一对一（适用于多核处理器；线程数量受硬件限制）
  + 多对多（单层模型、双层模型）

**Part 04：存储管理**

**-----------------------------------------------------------**

**重定位：**

* 定义：将作业地址空间中的逻辑地址变成内存中物理地址
* 种类：静态（程序装入时进行）、动态（程序运行时进行）

**存储管理方案：**

* 连续分配（单一连续、分区）
* 离散分配（段式、页式、段页式）
* 虚拟存储

**单一连续：**

将内存分为系统区和用户区，适合单用户单任务的OS，易于管理。但程序全装入导致浪费

**分区：**

* 有分区表（占用表、空闲表），表项有始址、长度和标志
* 类型：
  + 固定分区（大小相等、大小不等）：内存利用率↑，支持多道且简单；但作业需预估占用的大小空间，有内碎片且限制了并发程序数
  + 动态分区：解决了内碎片；但有外碎片
* 分配算法：
  + 最先适配：找到符合要求的第一个位置
  + 循环最先适配：从上次分配的分区开始，找到符合要求的第一个位置
  + 最佳适配：在大于等于要求长度的分区中选最小的
  + 最坏适配：直接选择最大的空闲区
  + 问题：会有许多的小碎片，需要紧凑技术将其合并，开销大（引出了离散分配）

**内存扩充：**

* 提出背景：作业的程序地址空间大于内存可使用空间
* 技术：
  + 覆盖技术：
    - 部分装入，必要部分常驻内存，可选部分到外存。
    - 增加了内外存交换的开销，且覆盖方法过于复杂，加重程序员负担
  + 交换技术：
    - 内存中暂时不能运行/不可用的程序和数据到外存上，进程状态增加挂起状态（在外存），通过激活回到内存
    - 交换单位可以是整个进程（整体交换），也可以是段或页（部分交换）
    - 并发程序数增多，响应时间好，不影响程序结构；但交换的控制开销大

**离散分配方式：**

* **页式存储管理：**
  + OS有一个物理页面表，描述物理内存空间的分配情况。固定分区用位图，动态分区用链表
  + 用户空间按逻辑页划分成大小相等的部分。逻辑地址中，高位为页号，低位为页内地址
  + 内存空间按页划分成大小相等的部分，又称物理页、页框、实页。**逻辑上相邻的页，物理上不一定相邻**
  + 系统为每个进程创建一个页表，在内存中，作为现场信息
  + 管理页表过程：
    - 计算一个作业所需页数N
    - 查位图，看是否有N个空闲页
    - 若有，则申请页表空间，将页表始址和N填入PCB
    - 分配N个空闲页，将块号和页号填入
    - 更新位图
  + 硬件支持
    - 页表长度寄存器
    - 页表始址寄存器
    - TLB：关联存储器，从逻辑页号映射到物理页号，缩短查找时间
  + 地址映射（设页大小为P，作业有N个页，主存空间M）：
    - 逻辑地址：高位log2 N位，低位log2 P位
    - 对某逻辑地址loc：
* loc >= N \* P，则发生越界中断
* 否则先求逻辑页号**p = loc / P**，页内偏移**q = loc - p \* P**。找到p对应的物理页r，物理地址为：**r \* P + q**
  + 逻辑地址空间大会造成页表过大，提高存储开销。可通过多级页表、哈希页表、反向页表解决
  + 优点：解决了碎片问题，便于管理
  + 缺点：不适合程序共享和动态链接
* **段式存储管理：**
  + 用户空间按程序自身逻辑划分成若干段。逻辑地址中，高位为段号，低位为段内地址
  + 内存空间被动态划分成大小不等的部分，又称物理段。物理段占据连续空间，但各段之间可以不连续
  + 系统为每个进程创建一个段表，在内存中，作为现场信息
  + OS段表会记录占用段的信息，空闲段信息可以结合到里面。分配算法就是分区管理的分配算法
  + 地址映射（设作业有N个段）：
    - 对某逻辑地址：段号为d，段内偏移b
    - d >= N，则因段号越界发生越界中断
    - 找到d对应的物理段q，其始址为s，长度为l
    - b >= l，则因段内偏移越界发生越界中断
    - 否则物理地址为s + b
  + 硬件支持
    - 页表长度寄存器
    - 页表始址寄存器
    - 联想存储器：进程段表的子集（表项加了状态位和访问位）
  + 优点：管理和使用统一，便于共享和动态链接
  + 缺点：产生碎片
* **段页式存储管理：**
  + 用户空间按段式划分，内存空间按页式划分，分配以页为单位
  + 逻辑地址：段号+段内地址（段内地址有页号+页内地址）

**虚拟存储：**

* 特征：不连续性、部分交换、大空间（物理内存+快速外存交换区）、实际使用再分配内存（先声明，需要再定义）
* 技术：
  + 虚拟段式
  + 虚拟页式：
    - 原理：需要某个不在内存中的块时就将其调入
    - 页表项：
* 页号
* 中断位：页面是否存在
* 保护位：有关页面的读写控制
* 访问位：页面是否正在被使用，用于置换
* 修改位：页面是否被修改过
* 页表：多级结构+TLB
* 缺页中断：在页表中发现所需页不在内存中。OS调出缺页中断程序调块
* 进程对块进行引用
* 从页表中发现不在内存中，陷入OS
* OS在外存中找到对应块
* 将块调入内存的一个空闲帧
* 更新页表状态位
* 重启指令
* 置换算法：
* **先进先出**：维护队列
* 最佳：无法实现，作为性能评价标准
* **最近最久未使用**：维护特殊栈，访问时移到栈顶
* 最不常用：维护每页的访问计数器
* 轮转：维护一个指针。从当前位置开始寻找use = 0的位置替换，期间经过的位置use设为0，替换后该位置use设为1，指针停在下一个位置
* 缺页率：缺页次数/访问总次数
* Belady现象：分配的页面数增多，缺页率反而增多（原因是被替换的页仍旧有可能被进程访问）
* 调入策略：
  + 请求调页：只调需要的页（容易实现；I/O多，开销大）
  + 预调页：调需要的页及其相邻几页（提高I//O效率；需要预测，因此一般用于程序装入时的调页）
* 系统颠簸：刚被替换的页很快又被访问，需要重新调入；但调入的页很快又再次被替换。如此反复导致系统的页面替换过于频繁。
  + 解决方式：局部置换（进程抖动时不从其他进程取页，限制抖动的范围）、工作集、挂起某些进程（优先级低、缺页率高、最大）

**Part 05：文件管理**

**-----------------------------------------------------------**

**磁盘：**

* 性质：直接（随机）存取设备
* 结构：信息记录在磁道上，多个盘片（两面都用），每一面一个磁头
* 物理地址（三维）：
  + 磁道号（柱面号）
  + 磁头号（盘面号）
  + 扇区号
* 调度：
  + 时间开销：
    - 寻道时间：磁头转动
    - 延迟时间：磁盘转动
    - 存取时间：数据传输
  + 过程：
    - 进程阻塞，访问请求加入请求队列
    - 切内核态，运行磁盘驱动程序
    - 读取信息，分析结果送至磁盘控制器
    - 磁盘控制器控制磁头寻道，并进行I/O操作
  + 方法：
    - 先来先服务
    - 最短寻道时间优先
    - 扫描算法（电梯算法）
    - 单向扫描算法
    - LOOK调度（扫描算法中不用到头，到最远的那个请求处即可）
* 格式化：
  + 低级：分扇区、准备备用块（出厂前已完成）
  + 高级：将磁盘分区、逻辑格式化或创建文件系统
* 坏块：
  + 处理：采用备用块逻辑替代坏块（扇区备用/扇区转寄）

**文件：**

* 定义：一组带标识的、在逻辑上有完整意义的信息项的序列
* 分类：
  + 逻辑结构上：流式（无结构）、记录式（有结构）
  + 物理结构上：连续（顺序）、链接、索引
  + ......
* 物理结构：
  + 连续：文件存放在若干连续的物理块中
  + 链接：文件存放在若干离散物理块中，串成链表。文件目录提供首地址和尾地址
  + 索引：为文件创建索引表，表项是块号
* 索引表组织
  + 链接模式
  + 多级链表
  + 综合模式（UNIX三级）

**文件系统：**

* 文件控制块（FCB）：存放为管理文件所需的所有信息，是文件存在的标志
* 目录：
  + 定义：FCB的有序集合（存储这个集合的文件叫**目录文件**）
  + 结构：
    - 一级目录
    - 二级目录（按用户分）
    - 多级目录（树形结构）
    - ......
    - 改进：将FCB分为符号目录项（小，次部）、基本目录项（大，主部）分开存在不同块中，查找文件时只需遍历符号目录项（较少的块），再访问一次基本目录项（较多的块）即可
* 空闲空间管理：
  + 位图
  + 链表（单独链接、成组链接）
* 实现：
  + 系统文件表：OS一张，在内存中，记录所有打开的文件信息
  + 用户文件表：进程一张，在PCB中，记录进程打开的文件信息
  + 用户的表项指向系统的表项，便于共享
* 文件保护：通过存取验证模块实现
  + 存取控制矩阵（一人一文件）
  + 存取控制表（一组用户一文件）
  + 口令和密码
* 文件恢复：
  + 一致性检查
    - 必要性：将数据写回磁盘前可能发生系统崩溃导致不一致情况出现
    - 一种解决：两张表，表项都对应块。一张表示块是否在被使用，一张表示块是否空闲，文件系统通过比对确定当前状态
    - 比对：表项均为一个0一个1——一致；表项均为0——块丢失；空闲表项大于1——空闲表中有重复块；使用中表项大于1——出现重复数据块