复习指南

🧩 一般题目结构与考点对应关系

题号	题型风格	对应重点知识模块	常见考查内容总结
第1题	**进程与调度分析/计算题**	✅ CPU虚拟化（进程调度）	- RR/SJF/SRTF 调度流程图- 中断间隔、上下文切换、耗时计算- 调度算法效果对比（如等待时间）
第2题	**死锁状态分析/银行家算法**	✅ 并发与进程同步	- 银行家算法（安全状态判断、请求是否可接受）- 死锁四条件分析- 等资源分配图状态分析
第3题	**分页/段式内存管理计算题**	✅ 内存虚拟化（分页/段页式管理）	- 虚地址到物理地址转换流程- TLB 命中/缺失延迟计算- 分页层次结构及页表项含义分析
第4题	**内存分配或页面置换策略题**	✅ 内存虚拟化（内存分配/置换）	- 可变分区首次适应/最佳适应分配模拟- 页表缺页统计与LRU/FIFO/OPT比较- 页框数量影响命中率（Belady异常）
第5题	**文件系统一致性检查/修复**	✅ 文件系统与持久化	- 块分配表错误类型识别- 指针/文件控制块（inode）结构- 修复建议与一致性分析（如 fsck 模拟）
第6题	**磁盘调度策略分析题**	✅ 磁盘持久化与I/O调度	- FCFS, SSTF, SCAN, C-SCAN 的路径比较- 磁道移动总距离计算- 实现伪代码或逻辑流程

🧠 补充说明

📌 第1题（调度类）

- 每年必考
- 通常给定一个时间片或调度顺序，要求你画出调度过程、计算平均等待时间、响应时间等
- 有时结合中断上下文切换耗时分析

📌 第2题（死锁&银行家算法）

- 银行家算法为高频题（2020/2023年均考）
- 判断当前是否为安全状态、模拟资源请求后的安全性

📌 第3题（分页&虚存）

- 涉及地址变换、TLB 命中、页表项结构
- 有时嵌套二级页表结构，考查理解+计算

📌 第4题（内存分配/页面置换）

- 内存分区策略考察常见分配算法
- 页面置换则关注页面访问序列与命中率比较
- 有时出现在简答题中考“为何某算法更优”

📌 第5题（文件系统）

- 不常出计算题，偏向原理+故障识别
- 多数题目要求你指出磁盘块/文件目录结构的错误并给出修复建议

📌 第6题（I/O设备调度）

- 磁盘调度路径比较（常见4种策略）为常考点
- 伪代码实现有一定可能性，需要理解调度方向切换的逻辑

🎯 应试策略建议

模块	题号	备考建议
调度	第1题	模拟调度流程、画甘特图、精通RR/SJF/SRTF
死锁	第2题	熟练掌握银行家算法操作表格+安全性判断流程
分页	第3题	多做地址转换+TLB命中分析题，牢记页表结构
分区/置换	第4题	模拟算法步骤，注意对比算法命中差异
文件系统	第5题	记住inode结构，熟悉常见文件系统错误形式
磁盘调度	第6题	用图画移动轨迹，比较调度算法逻辑与移动距离

有操作系统编程题虽然每年题干不同，但**考查套路非常固定**，主要集中在**进程同步与并发控制**方面，几乎都可归结为“临界区 + 信号量”控制问题。

🎯 编程题套路总结（基于2020–2023年）

✅ 高频考点：

类型	常考题型例子	对应机制	要求实现/思路
并发同步题	哲学家进餐、猴子过桥问题	P/V 信号量	用信号量控制资源数、防止死锁或饥饿
互斥题	多线程输出顺序、单资源访问	mutex 互斥锁	用信号量或布尔标志位控制访问
条件同步题	生产者消费者、读者写者	wait/signal	按照逻辑顺序控制多个线程之间的依赖
死锁避免题	银行家算法伪代码实现	安全状态检查	动态检查请求是否合法，避免系统进入死锁

🧩 编程题套路模板

🎯【套路1：信号量+循环体实现并发控制】

> 多个线程同时运行，有顺序或资源限制。你要设置信号量、在关键区前后加 P/V 操作。

常规模板：

```
sem_t mutex;

void* thread_func(void* arg) {
    while (1) {
        sem_wait(&mutex);      // P操作
        // 临界区：访问共享资源
        sem_post(&mutex);      // V操作
        // 非临界区
    }
}
```

🎯【套路2：计数信号量控制资源数量】

> 如过桥、猴子问题、最大连接数，要求同时不超过N个线程访问

```
sem_t slots;

void* monkey(void* arg) {
    sem_wait(&slots);      // 占用一个名额
    // 过桥过程
    sem_post(&slots);      // 释放名额
}
```

🎯【套路3：条件变量模拟线程协作】

> 如生产者消费者，需要 empty/full 信号量

```
sem_t empty, full, mutex;

void* producer(void* arg) {
    while (1) {
        sem_wait(&empty);  // 等待空位
        sem_wait(&mutex);
        // 放入产品
        sem_post(&mutex);
        sem_post(&full);   // 增加产品数
    }
}

void* consumer(void* arg) {
    while (1) {
        sem_wait(&full);   // 等待产品
        sem_wait(&mutex);
        // 取出产品
        sem_post(&mutex);
        sem_post(&empty);  // 增加空位
    }
}
```

🧠 出题特点与解题套路总结

特点	解题建议
场景化（猴子、哲学家）	抽象出资源与线程行为，分类建模
明确同步目标（无碰撞、无死锁）	写出并发行为图，标明限制关系
考查语义正确性（不丢资源）	用信号量精准表示并发条件
不考代码细节	伪代码写清逻辑/变量名/语义足矣

✅ 考前建议训练

- 熟练写出**哲学家、生产者消费者、读写者**三大经典模型；
- 练习**猴子过桥/过河/上下楼梯**等"资源配额问题"；
- 模拟题目的"角色-资源-动作"三元关系图，有助快速写信号量方案；
- 用"初始化信号量 + while + P/V控制"三段式结构答题。

如果你给出你今年实验课的编程练习题，我可以更精准地帮你整理"最可能出编程题变体"的预测模型。要继续吗？