绝密★启用前

2022 年河南工业大学期末考试

操作系统

注意事项：

1. 答卷前，考生务必将自己的姓名和准考证号填写在答题卡上。
2. 回答选择题时，选出每小题答案后，用铅笔把答题卡对应题目的答案标号涂黑。如需改动，用橡皮擦干净后，再选涂其它答案标号。回答非选择题时，将答案写在答题卡上。写在本试卷上无效。
3. 考试结束后，将本试卷和答题卡一并交回。请认真核对监考员在答题卡上所粘贴的条形码上的姓名、准考证号与您本人是否相符。

一、 选择题

1. 操作系统最主要的任务是( )。

   A. 管理资源　　B. 并发　　C. 异步　　D. 虚拟

2. ( )是操作系统形成的标志。

   A. 单道批处理系统　　B. 多道批处理系统　　C. 分时系统　　D. 实时系统

3. 进程从运行状态进入就绪状态的原因可能是( )。

   A. 被选中占有处理机　　B. 等待某一事件　　C. 等待的事件已发生　　D. 时间片用完

4. 在信号量机制中, 哪种信号量的 wait 操作, 不满足“让权等待”?( )

   A. 整型信号　　B. 记录型信号量　　C. And 型信号量　　D. 一般信号量集

5. 操作系统中的存储管理是指对( )的管理。

   A. 主存　　B. 辅存　　C. Cache　　D. 以上都不是

6. 在请求分页管理方式中, 页表中的状态位(存在位)用来指示对应页( )。

   A. 是否被修改过　　B. 是否允许动态增长　　C. 是否已调入内存　　D. 是否已置换

7. 可变式分区分配方案中, 某一作业完成后, 系统收回其主存空间, 并与相邻空闲区合并, 为此需修改空闲区表, 造成空闲区数减 1 的情况是( )。

   A. 无上邻空闲区, 也无下邻空闲区　　B. 有上邻空闲区, 但无下邻空闲区

   C. 有下邻空闲区, 但无上邻空闲区　　D. 有上邻空闲区, 也有下邻空闲区

8. 典型的分时系统采用的进程调度算法是( )。

   A. 先来先服务　　B. 短进程优先　　C. 高优先权优先　　D. 时间片轮转

9. 通道是一种( )。

   A. I/O 端口　　B. 数据通道　　C. 软件工具　　D. I/O 专用处理器

10. 基本分段存储管理方式中, 逻辑地址的地址格式是( )地址。

A. 线性　　B. 一维　　C. 二维　　D. 三维

11. 动态重定位是在作业( )中进行的。

A. 编译过程　　B. 链接过程　　C. 装入过程　　D. 执行过程

12. 为实现设备的分配, 应为每系统中所有的设备配置一张( )。

A. 逻辑设备表　　B. 设备控制表　　C. 系统设备表　　D. 设备开关表

13. 位示图法可用于( )。

A. 磁盘空间的管理　　B. 磁盘的驱动调度

C. 文件目录的查找　　D. 页式虚拟存贮管理中的页面调度

14. 哪一种文件的共享方式, 可以允许文件主随时删除共享文件?( )

A. FCB 拷贝复制共享　　B. 基于索引结点的共享方式

C. 基于符号链的共享方式　　D. 以上都不对

15. 文件系统中, 目录管理的最基本功能是( )。

A. 实现虚拟存储　　B. 实现文件的按名存取

C. 提高外存的读写速度　　D. 用于存储系统文件

二、 填空题 (共 10 空, 每空 1 分, 共 10 分)

1. 进程存在的唯一标志是 _____。
2. 处理机调度的层次包括：高级调度、低级调度、_____。
3. 进程调度包括两种调度方式:非抢占式进程调度方式、_____ 进程调度方式。
4. 若 P、V 操作的信号量 S 初值为 2, 当前值为–1, 则表示有 ____ 个等待进程。
5. 数据传输控制方式可分为程序 I/O 控制方式、中断控制方式、_____ 和 I/O 通道控制方式。
6. 为了能对一个文件进行正确的存取, 必须为文件设置用于描述和控制文件的数据结构, 称之为 _____。
7. 设备驱动程序是 I/O 进程和 _____ 之间的通信程序。
8. 文件系统管理的对象包括：文件、目录、_____。
9. 从用户的角度, 观察到的文件的组织形式叫做文件的 _____。
10. 在引入线程的 OS 中, 独立调度和分派的基本单位是线程, 资源分配的单位是 _____。

三、 简答题 (共 3 题, 每题 5 分, 共 15 分)

1. 什么是操作系统? 操作系统的特征有哪些?
2. 请画出三种基本状态的转换图, 并注明各状态转换时的条件。
3. 什么是虚拟存储器? 它具有哪些基本特征?

四、 算法综合题 (共 6 题, 每题 10 分, 共 60 分)

1. 假设一个系统有 5 个进程, 它们的到达时间和服务时间如表 1 所示, 忽略 I/O 及其他开销时间, 计算先来先服务 FCFS、短进程优先 SPF 进行 CPU 调度, 请完成表 2。(10 分)

进程	到达时间	服务时间
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

表 1　表 1 进程到达和服务时间

	FCFS					SPF				
进程	**A**	**B**	**C**	**D**	**E**	**A**	**B**	**C**	**D**	**E**
完成时间										
周转时间										
带权周转时间										

表 2　表 2 (数值若无法除尽, 直接用分数表示即可)

2. 系统有(A, B, C, D)四种资源和五个进程, 在银行家算法中, 某时刻出现下述资源分配情况

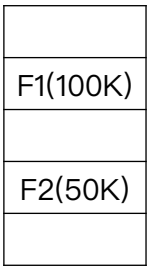
Process	Allocation (A,B,C,D)	Need (A,B,C,D)	Available (A,B,C,D)
P0	(0, 0, 3, 2)	(0, 0, 1, 2)	(1, 6, 2, 2)
P1	(1, 0, 0, 0)	(1, 7, 5, 0)	
P2	(1, 3, 5, 4)	(2, 3, 5, 6)	
P3	(0, 3, 3, 2)	(0, 6, 5, 2)	
P4	(0, 0, 1, 4)	(0, 6, 5, 6)	

试问:
(1) 该状态是否安全? 请写出分析过程。(6 分)
(2) 若进程 P2 提出请求 Request(1, 2, 2, 2), 系统能否分配给它资源? 写出分析过程。(4 分)

3. 在一个请求分页系统中, 假如一个作业的页面走向为 0, 1, 4, 2, 0, 2, 6, 5, 1, 2, 3, 2, 1, 2, 6, 2. 当分配给该作业的物理块数为 3 时, 采用最近最久未使用(LRU)页面替换算法, 计算访问过程中所发生的缺页次数。(写出具体过程) (10 分)

4. 对于移动头磁盘, 假设磁头现在位于 25 号磁道上, 且基于磁道号的磁盘访问请求序列(按提出时间的先后次序排列)为 39, 62, 18, 28, 100, 130, 90. 试采用最短寻道时间优先(SSTF)调度算法和电梯调度(SCAN)算法(规定先向磁道号变小的方向移动), 分别给出相关磁盘访问请求处理的先后次序, 并计算相应的平均寻道长度(若无法除尽, 直接用分数表示)。(10 分)

5. 设作业 A(30K), B(70K) 和 C(50K) 依次请求内存分配, 内存现有 F1(100K), F2(50K) 两个空闲区, 如图所示。分别采用最佳适应算法和最差适应算法, 画出每一步的内存分配情况示意图。(10 分)



6. 利用信号量机制写出一种不会发生死锁的“哲学家就餐问题”解决方案。(10 分)

五、 参考答案

5.1. 选择题答案

1. 答案：A

操作系统的主要任务是管理资源。

2. 答案：B

多道批处理系统是操作系统形成的标志。

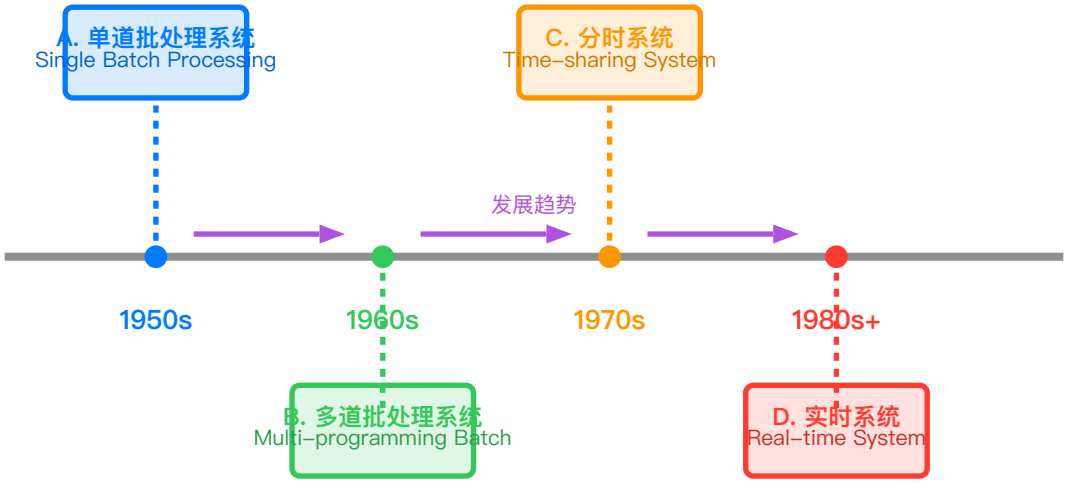


图 1　操作系统发展历程

，

3. 答案：D

进程从运行状态进入就绪状态的原因可能是时间片用完。不做过多解释。

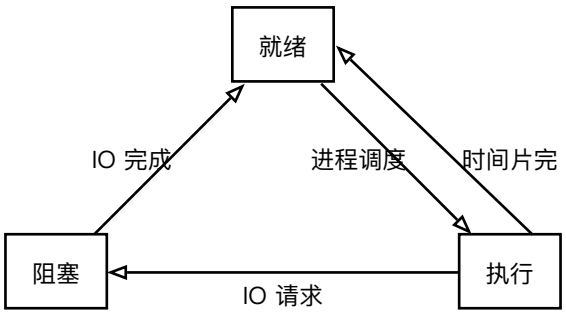


图 2　三种基本状态及转换

4. 答案：A

“让权等待”（blocking wait）指的是： 当进程执行 wait() 操作时，如果资源不可用，该进程会让出 CPU，进入等待队列，直到条件满足再被唤醒。也就是主动“让出执行权”，不再占用 CPU 时间片。
整型信号 • 最简单的信号量实现，实际上就是一个整数变量。wait 操作通常是不释放 CPU，而是不断地轮询变量

```
wait(S) {
  while (S <= 0); // busy wait
  S--;
}
```

5. 答案：A

主存的存储管理是操作系统的核心任务之一。主存是计算机中速度最快的存储器，操作系统需要对其进行有效的管理，以确保各个进程能够高效地运行。

6. 答案：C

页表中的状态位（存在位）用来指示对应页是否已调入内存。若该位为 1，则表示该页已在内存中；若为 0，则表示该页不在内存中。

7. 答案：D

8. 答案：D

分时系统（Time-Sharing System）采用时间片轮转调度算法：

1. 把处理器时间划分成很短的时间片，按顺序轮流分配给多个用户或进程执行，使每个用户都感觉自己独占系统。
2. 分时系统的主要特点：
   - 时间片机制：系统将 CPU 时间划分成一段段时间片
   - 多用户共享：支持多个用户/进程同时登录与操作
   - 响应快速：每个用户响应时间通常小于 1 秒
   - 并发执行：支持多个程序“同时”运行（逻辑并发）
   - 自动切换：系统负责在多个用户/进程之间快速切换
3. 调度过程示例：三个用户运行程序 A、B、C，时间片设为 100 毫秒

   ```
   时间轴: | A | B | C | A | B | C | A | B | ...
         ↑每100ms切换一次↑
   ```

4. 系统类型对比：

类型	用户互动	响应速度	适用场景
分时系统	有	快（秒级）	多用户共享计算机
批处理系统	无	慢（分钟级）	数据处理任务
实时系统	有	很快（毫秒级）	工业控制、航空航天

9. 答案：D

通道是一种 I/O 专用处理器，用于处理输入输出操作。它可以独立于 CPU 执行 I/O 操作，从而提高系统的整体性能。

10. 答案：C

基本分段存储管理方式中，逻辑地址的地址格式是二维地址。每个段有一个段号和段内偏移量。

🎯 示例逻辑地址

逻辑地址为 (1, 120) 表示： • 访问的是段号 1（数据段） • 在该段的偏移量为 120

11. 答案：D

执行过程中进行动态重定位。操作系统在装入程序时，将逻辑地址转换为物理地址，并进行必要的调整。动态重定位允许程序在运行时根据实际内存情况进行地址调整。

12. 答案：B

设备控制表用于描述和控制系统中所有设备的状态和属性。每个设备都有一个对应的控制块，操作系统通过该表来管理设备的分配和使用。

13. 答案：A

位示图法可用于磁盘空间的管理。它通过使用位图来表示磁盘上每个块的使用情况，1 表示已分配，0 表示空闲。

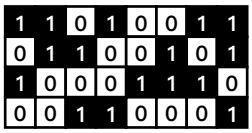


图 3　磁盘空间位示图管理示例

14. 答案：A

FCB 拷贝复制共享方式允许文件主随时删除共享文件。因为每个用户都有自己的文件控制块（FCB）副本，删除操作不会影响其他用户的访问。

15. 答案：B

目录管理的最基本功能是实现文件的按名存取。通过目录结构，用户可以方便地查找和访问文件。

5.2. 填空题答案

1. 答案：进程控制块(PCB)

进程存在的唯一标志是进程控制块(PCB)。PCB 包含了进程的所有信息，如进程状态、程序计数器、寄存器内容等。

```
struct PCB {
    int pid;                    // 进程ID
    int ppid;                   // 父进程ID
    char state;                 // 进程状态
    int priority;               // 优先级
    int* page_table;            // 页表指针
    struct File* open_files[]; // 打开的文件列表
    CPU_Context context;       // CPU寄存器信息（上下文）
};
```

2. 答案：中级调度

处理机调度的层次包括：高级调度、低级调度、中级调度。高级调度决定哪些进程进入就绪队列，低级调度决定哪个就绪进程获得 CPU 时间片，中级调度则在需要时将进程从内存换出或换入。

3. 答案：抢占式

进程调度包括两种调度方式：非抢占式进程调度方式和抢占式进程调度方式。非抢占式调度方式下，进程在运行时不会被强制中断，而抢占式调度方式允许操作系统在任何时候中断正在运行的进程。

4. 答案：3

若 P、V 操作的信号量 S 初值为 2, 当前值为–1, 则表示有 3 个等待进程。因为信号量的当前值为负数时，表示有多少个进程在等待该信号量。

5. 答案：DMA 方式

数据传输控制方式可分为程序 I/O 控制方式、中断控制方式、DMA 方式和 I/O 通道控制方式。DMA（直接内存访问）允许设备直接与内存交换数据，而不需要 CPU 的干预，从而提高了数据传输效率。

5. 答案：DMA 方式

数据传输控制方式可分为程序 I/O 控制方式、中断控制方式、DMA 方式和 I/O 通道控制方式。DMA（直接内存访问）允许设备直接与内存交换数据，而不需要 CPU 的干预，从而提高了数据传输效率。

控制方式	CPU 参与度	数据传输	中断使用	效率	适用场景
程序 I/O	完全参与	CPU 逐字节传输	轮询检查	低	简单设备
中断控制	部分参与	CPU 按块传输	中断通知	中等	键盘、鼠标
DMA 方式	最少参与	硬件直接传输	传输完成中断	高	磁盘、网卡
I/O 通道	几乎不参与	通道处理器控制	通道完成中断	最高	大型机系统

表 4　四种 I/O 控制方式对比

详细说明：

1. **程序 I/O 方式**：CPU 全程参与，效率最低，适合简单设备
2. **中断控制方式**：CPU 在设备就绪时响应中断，减少了等待时间
3. **DMA 方式**：设备控制器直接访问内存，CPU 只需启动和结束时参与
4. **I/O 通道方式**：专用处理器处理 I/O 操作，CPU 几乎完全解放

6. 答案：文件控制块(FCB)

为了能对一个文件进行正确的存取，必须为文件设置用于描述和控制文件的数据结构，称之为文件控制块(FCB)。FCB 包含了文件的元数据，如文件名、大小、位置等信息。

7. 答案：设备控制器

设备驱动程序是 I/O 进程和设备之间的通信程序。它负责将 I/O 请求转换为设备可以理解的格式，并处理设备的响应。 • I/O 进程：指用户程序或系统中的 I/O 管理模块。 • 设备控制器：是连接 CPU 和外设的硬件组件，负责控制具体设备的工作，如磁盘控制器、显示控制器等。 • 设备驱动程序：是操作系统中用于实现 软件与硬件之间通信 的模块，相当于中间翻译器。

8. 答案：设备控制器

文件系统管理的对象包括：文件、目录、设备控制器。设备控制器是操作系统与硬件设备之间的接口，负责管理和控制外部设备的操作。

9. 答案：逻辑视图

从用户的角度，观察到的文件的组织形式叫做文件的逻辑视图。逻辑视图是用户对文件的抽象表示，与物理存储方式无关。

10. 答案：进程

在引入线程的 OS 中，独立调度和分派的基本单位是线程，资源分配的单位是进程。线程是进程内的执行单元，多个线程可以共享同一进程的资源。

一句话理解：

名称	简单定义
进程（Process）	是程序在操作系统中运行的实例，是资源分配的最小单位。
线程（Thread）	是进程内部的一个执行单元，是 CPU 调度的最小单位。

什么是进程（Process）？

一个运行中的程序就是一个进程。

- 比如你打开了两个 Word 文件，这就是两个进程，虽然程序是同一个。
- 每个进程有自己独立的：地址空间、代码、数据、堆栈、打开的文件、资源权限等。

- 操作系统调度多个进程实现“多任务”。

举例：
- 打开微信 → 一个进程
- 打开浏览器 → 一个进程

什么是线程（Thread）?

线程是进程中的一个执行流，多个线程可以在同一个进程里并行执行。
- 多个线程共享该进程的内存资源（如数据段、堆、文件等），但有自己独立的栈和寄存器上下文。
- 多线程能提高程序的并发性和资源利用率。

举例： 打开微信（主线程） → 接收消息（线程 A）、播放语音（线程 B）、文件下载（线程 C）

对比总结表格：

比较项	进程（Process）	线程（Thread）
定义	程序的执行实例	进程内的执行单元
是否独立	独立单元（互不干扰）	线程间共享进程资源（不独立）
拥有资源	拥有自己的资源（如内存空间）	共享进程资源
通信开销	进程间通信（IPC）开销较大	线程间通信方便（如共享变量）
创建销毁代价	大，操作系统需分配资源	小，开销更低
CPU 调度	是调度单位（早期）	是现代调度的最小单位
举例	浏览器进程、Word 进程等	浏览器标签页线程、下载线程、JS 线程

类比理解：

类比对象	进程	线程
公司/工厂	整个公司（独立空间）	公司内员工（共享资源）
操场跑步	每个跑道是一进程	跑道上的人是线程

5.3. 简答题答案

1. 答案：

操作系统是管理计算机硬件和软件资源的系统软件，为用户提供便利的操作界面。
操作系统的特征包括：资源管理、并发处理、虚拟化、抽象化、可靠性和安全性。

2. 答案：

三种基本状态的转换 图 2 所示

3. 答案：

虚拟存储器是操作系统提供的一种内存管理技术，它允许程序使用比实际物理内存更大的连续地址空间。它的基本特征包括：

特征	具体含义	实现机制
离散性	程序可分散存储	分页/分段技术
多次性	程序可多次调入调出	请求调页/段
对换性	内存与外存间数据交换	页面置换算法
虚拟性	逻辑容量大于物理容量	地址映射机制

表 5　虚拟存储器基本特征详解

5.4. 算法综合题答案

1. 答案：

先来先服务（FCFS）和短进程优先（SPF）调度算法的计算过程如下：

先来先服务（FCFS）算法计算：

按到达时间排序：A(0,3) → B(2,6) → C(4,4) → D(6,5) → E(8,2)

执行过程：

- A: 0→3 (完成时间=3)
- B: 3→9 (完成时间=9)
- C: 9→13 (完成时间=13)
- D: 13→18 (完成时间=18)
- E: 18→20 (完成时间=20)

短进程优先（SPF）算法计算：

按服务时间排序，但需考虑到达时间：

- t=0: A 到达，开始执行 A(0→3)
- t=3: B,C 都已到达，选择服务时间短的 B 开始(3→9)
- t=9: C,D,E 都已到达，按服务时间排序：E(2)<C(4)<D(5)，执行 E(9→11)
- t=11: 执行 C(11→15)
- t=15: 执行 D(15→20)

计算公式：

- 周转时间 = 完成时间 − 到达时间
- 带权周转时间 = 周转时间 ÷ 服务时间

	FCFS					SPF				
进程	**A**	**B**	**C**	**D**	**E**	**A**	**B**	**C**	**D**	**E**
完成时间	3	9	13	18	20	3	9	15	20	11
周转时间	3	7	9	12	12	3	7	11	14	3
带权周转时间	1	7/6	9/4	12/5	6	1	7/6	11/4	14/5	3/2

表 6　表 2 完整计算结果

2. 答案：

银行家算法分析

(1) 安全性分析（6 分）

给定状态：

- Available = (1, 6, 2, 2)
- 各进程的 Allocation 和 Need 如表所示

使用银行家算法检查安全性：

进程	Allocation	Need	Available	可完成	新 Available
P0	(0,0,3,2)	(0,0,1,2)	(1,6,2,2)	✓	(1,6,5,4)
P1	(1,0,0,0)	(1,7,5,0)	(1,6,5,4)	×	–
P2	(1,3,5,4)	(2,3,5,6)	(1,6,5,4)	×	–
P3	(0,3,3,2)	(0,6,5,2)	(1,6,5,4)	✓	(1,9,8,6)
P4	(0,0,1,4)	(0,6,5,6)	(1,9,8,6)	✓	(1,9,9,10)

继续检查剩余进程：

进程	Allocation	Need	Available	可完成	新 Available
P1	(1,0,0,0)	(1,7,5,0)	(1,9,9,10)	✓	(2,9,9,10)
P2	(1,3,5,4)	(2,3,5,6)	(2,9,9,10)	✓	(3,12,14,14)

安全序列：P0 → P3 → P4 → P1 → P2

结论：该状态是安全的。

(2) 请求分析（4 分）

P2 请求 Request(1, 2, 2, 2)：

步骤 1：检查请求合法性

- Request(1,2,2,2) ⩽ Need(2,3,5,6) ✓
- Request(1,2,2,2) ⩽ Available(1,6,2,2) ?
  - 1 ⩽ 1 ✓
  - 2 ⩽ 6 ✓
  - 2 ⩽ 2 ✓
  - 2 ⩽ 2 ✓

步骤 2：试分配

```
Available' = (1,6,2,2) - (1,2,2,2) = (0,4,0,0)
Allocation'[P2] = (1,3,5,4) + (1,2,2,2) = (2,5,7,6)
Need'[P2] = (2,3,5,6) - (1,2,2,2) = (1,1,3,4)
```

步骤 3：安全性检查

新状态下的资源分配：

进程	Allocation’	Need’	Available’
P0	(0,0,3,2)	(0,0,1,2)	(0,4,0,0)
P1	(1,0,0,0)	(1,7,5,0)	
P2	(2,5,7,6)	(1,1,3,4)	
P3	(0,3,3,2)	(0,6,5,2)	
P4	(0,0,1,4)	(0,6,5,6)	

检查各进程是否可完成：

- P0: Need(0,0,1,2) ≤ Available(0,4,0,0) ？ 1 > 0, 2 > 0 ×
- P1: Need(1,7,5,0) ≤ Available(0,4,0,0) ？ 1 > 0, 7 > 4, 5 > 0 ×
- P2: Need(1,1,3,4) ≤ Available(0,4,0,0) ？ 1 > 0, 3 > 0, 4 > 0 ×
- P3: Need(0,6,5,2) ≤ Available(0,4,0,0) ？ 6 > 4, 5 > 0, 2 > 0 ×
- P4: Need(0,6,5,6) ≤ Available(0,4,0,0) ？ 6 > 4, 5 > 0, 6 > 0 ×

结论：没有进程能够完成，系统进入不安全状态。

因此，系统不能分配给 P2 所请求的资源。

3. 答案：

LRU 页面替换算法分析

给定页面访问序列：0, 1, 4, 2, 0, 2, 6, 5, 1, 2, 3, 2, 1, 2, 6, 2

物理块数：3 块

LRU（Least Recently Used）算法：当需要替换页面时，选择最近最久未使用的页面进行替换。

详细执行过程：

访问页面	物理块 1	物理块 2	物理块 3	是否缺页	说明
0	0	–	–	✓	首次访问，缺页
1	0	1	–	✓	新页面，缺页
4	0	1	4	✓	新页面，缺页
2	2	1	4	✓	替换最久未用的 0
0	2	0	4	✓	替换最久未用的 1
2	2	0	4	×	页面 2 已在内存
6	2	0	6	✓	替换最久未用的 4
5	5	0	6	✓	替换最久未用的 2
1	5	1	6	✓	替换最久未用的 0
2	5	1	2	✓	替换最久未用的 6
3	3	1	2	✓	替换最久未用的 5
2	3	1	2	×	页面 2 已在内存
1	3	1	2	×	页面 1 已在内存
2	3	1	2	×	页面 2 已在内存

6	3	6	2	✓	替换最久未用的 1
2	3	6	2	×	页面 2 已在内存

总结：

- 总访问次数：16 次
- 缺页次数：11 次
- 缺页率：11/16 = 68.75%
- 命中次数：5 次
- 命中率：5/16 = 31.25%

LRU 算法特点：

1. 基于时间局部性原理：最近使用的页面很可能再次被使用
2. 需要记录每个页面的使用时间或顺序
3. 算法开销较大，但置换效果较好
4. 适合大多数实际应用场景

4. 答案：

SCAN 调度（又称电梯调度）是一种磁盘调度算法，磁头像电梯一样从一端向另一端移动，依次处理所经过的所有请求，抵达边界后再反向扫描。这样能较均衡地减少平均寻道时间与磁头移动距离。

SCAN 调度过程：

1. 磁头初始位置在 25，向上扫描。
2. 先访问高于 25 的磁道：
   - 从 25 → 28（移动 3）
   - 从 28 → 39（移动 11）
   - 从 39 → 62（移动 23）
   - 从 62 → 90（移动 28）
   - 从 90 → 100（移动 10）
   - 从 100 → 130（移动 30）
3. 已无更高请求后，磁头掉头向下扫描：
   - 从 130 → 18（移动 112）

总移动距离： 3 + 11 + 23 + 28 + 10 + 30 + 112 = 217

5. 答案：

F1(100K)	2B 3C_{Failed}
F2(50K)	1A

表 7　最佳适应

F1(100K)	1A 2B
F2(50K)	3C

表 8　最差适应

6. 答案：

```
  // 初始化
sem forks[5] = {1, 1, 1, 1, 1};
sem room = 4; // 最多允许4位哲学家同时尝试进餐

philosopher(i):
    while (1) {
        think();
        wait(room);           // 进入房间
        wait(forks[i]);       // 拿左筷子
        wait(forks[(i+1)%5]); // 拿右筷子,forks[(i + 1) % N] 为了让数组索引回0的位置，形成一个环形的就餐结构。
        eat();
        signal(forks[i]);     // 放左筷子
        signal(forks[(i+1)%5]); // 放右筷子
```

```
        signal(room);          // 离开房间
    }
```

下面是完整 c 代码,但是不用写这么细

```c
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>

#define N 5  // 哲学家数量

sem_t forks[N];         // 表示每只筷子的信号量
sem_t room;             // 控制最多允许进入就餐的哲学家数（限制为 N-1）

void* philosopher(void* num) {
    int id = *(int*)num;

    while (1) {
        printf("哲学家 %d 正在思考\n", id);
        sleep(1); // 模拟思考

        sem_wait(&room);              // 进入房间限制（最多N-1人）
        sem_wait(&forks[id]);         // 拿起左边的筷子
        sem_wait(&forks[(id + 1) % N]); // 拿起右边的筷子

        printf("哲学家 %d 正在吃饭\n", id);
        sleep(1); // 模拟吃饭

        sem_post(&forks[id]);         // 放下左边的筷子
        sem_post(&forks[(id + 1) % N]); // 放下右边的筷子
        sem_post(&room);              // 离开房间

        printf("哲学家 %d 吃完了，开始思考\n", id);
    }

    return NULL;
}

int main() {
    pthread_t tid[N];
    int ids[N];

    sem_init(&room, 0, N - 1); // 最多允许N-1个哲学家尝试吃饭
    for (int i = 0; i < N; i++) {
        sem_init(&forks[i], 0, 1);
        ids[i] = i;
        pthread_create(&tid[i], NULL, philosopher, &ids[i]);
    }

    for (int i = 0; i < N; i++) {
        pthread_join(tid[i], NULL);
    }

    return 0;
}
```