

2023-2024 学年《操作系统》A 卷
参考答案与评分标准

一、教学目标 1 (35 分)

1. 解答：（共 8 分）

- 1) 0644 （2 分）
- 2) 备份的文件描述符 $fd_backup = 4$ （3 分）
- 3) 文件描述符 $fd = 3$ （3 分）

2. 解答：（共 5 分）

内部碎片发生在分配给进程的内存块内部。当内存块分配给进程的尺寸大于进程实际所需的尺寸时，未使用的部分就形成了内部碎片。

外部碎片是指分配和释放内存块时在内存中形成的空闲空间小洞。这些小空间单独来看可能无法满足新的内存请求，即使它们的总和可能足够大。

内部碎片和外部碎片都会导致内存资源的浪费，减少了系统的效率和性能。内部碎片主要影响内存的利用率，而外部碎片则可能导致内存分配请求失败，增加了系统运行的不稳定性。

3. 解答：共（8 分）

1) 直接地址指向的块数：前 8 个地址直接指向 8 个数据块。

一级索引指向的块数：第 9 个地址指向一个索引块，该索引块能存储 256 个磁盘块号，因此通过一级索引可以访问 256 个数据块。

二级索引指向的块数：第 10 个地址是二级索引指针，指向一个索引块，该索引块包含 256 个一级索引块的地址。每个一级索引块又指向 256 个数据块，因此二级索引可以指向 $256 * 256 = 65536$ 个数据块。

最大磁盘块数量： $8 + 256 + 65536 = 65800$ （3 分）

2) 每个数据块的大小为 1KB（1024 字节），所以文件的最大大小为：最大字节容量 = $65800 \text{ 块} \times 1024 \text{ 字节/块} = 67379200 \text{ 字节}$ （2 分）

3) 最少磁盘块数：如果要读取的字节位于前 8 个直接地址指向的块中，只需要读取 1 个数据块。（1 分）

最多磁盘块数：如果要读取的字节位于通过二级索引指向的块中，首先需要读取二级索引块（第 10 个地址指向），然后读取从二级索引块指向的一个一级索引块，最后读取目标数据块。这涉及 3 个磁盘块的读取。（2 分）

4. 解答：（共 5 分）

虚拟存储器允许仅将最近需要运行的程序代码和数据载入内存而不会严重影响进程运行速度的原因，主要基于程序运行的局部性原理：时间局部性和空间局部性，时间局部性指的是如果一个数据项被访问，那么它很可能在不久的将来再次被访问，空间局部性是指如果一个数据项被访问，那么其附近的数据项也很可能在不久的将来被访问。在虚拟存储器中，基于局部性原理，操作系统会预加载那些预测将会被访问的页面到物理内存。同时，操作系统使用页面替换算法（如最近最少使用（LRU）或先进先出（FIFO））来管理内存，保持经常访问的数

据在内存中，而将不常用的数据交换到磁盘。这种按需加载和智能页面替换确保了物理内存的有效使用，而不需要一次性将整个程序或数据集加载到内存中，从而优化了性能并减少了内存需求。

5. 解答（共 9 分）

1) 逻辑地址的有效位数为 32 位。（2 分）

2) 物理地址需要 28 位。（2 分）

3) 第二次访问地址 3A24H 时的物理地址为 47652（十六进制为 0xBA24）。（3 分）

当前在内存中的页面：页面 3（帧 11），页面 2（帧 12），页面 4（帧 10）（2 分）

二、教学目标 2（35 分）

1.（15 分，每小题 3 分）参考答案：

1) 4

2) $x=2$

3) 子进程

4) $x=0$

5) 会

2.（第一问 3 分，后面每个问题 2 分）

参考答案：因为操作系统是根据 PCB 的内容来管理进程的。

PID：标识进程身份

优先级：用于调度算法中确定进程执行顺序。

程序计数器（PC）：记录进程下一条要执行的指令地址。

3.（每个问题 2.5 分）

参考答案：1、I/O 完成，2、进程调度，3、时间片用完，4、I/O 请求。

三、教学目标 3（20 分）

1.（5 分）处理机调度中“可抢占”和“非抢占”两种方式，哪一种系统的开销更大？为什么？

答：

可抢占式会引起系统的开销更大。（2 分）

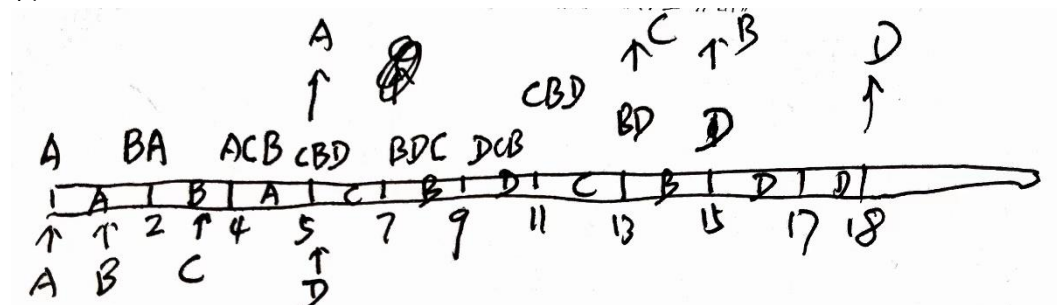
可抢占式调度是严格保证任何时刻，让具有最高优先权的进程占有处理机运行，因此增加了处理机调度的时机，引起为退出处理机的进程保留现场，为占有处理机的进程恢复现场等时间（和空间）开销增大。（3 分）

2.（5 分）假设一个系统中有 5 个进程，它们的到达时间和服务时间如下表所示，忽略 I/O 以及其他开销，按时间片轮转调度算法，时间片为 2，请计算各进程的周转时间和平均周转时间。

作业	到达时间	服务时间	开始时间	结束时间	周转时间
----	------	------	------	------	------

A	0	3	(1)	(2)	(3)
B	1	6	(4)	(5)	
C	3	4			
D	5	5	9	18	13

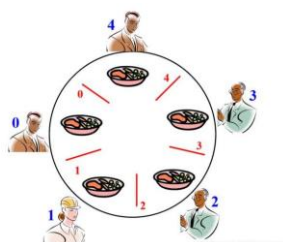
答:



(1) 0 (2) 5 (3) 5 (4) 2 (5) 15

(评分标准: 每空 1 分)

3. (5 分) 分析哲学家就餐问题中是否可能产生死锁, 如果是, 请设计一种解决方案来防止死锁的发生, 可用文字或代码表述。



答:

有可能产生死锁。(1 分)

死锁产生的场景是每个哲学家同时拿起左手边的筷子, 当伸手去拿右边筷子时, 都失败。(1 分)

解决方案一, 静态资源分配法: 每个人同时伸出两只手, 去拿左右两边的筷子, 只要有一只筷子不在, 就不拿了, 稍后再试。

解决方案二, 有序资源分配法: 给五根筷子编号, 要求每个人先拿编号小的筷子, 再拿编号大的筷子。(3 分)

4. (5 分) 死锁预防和死锁避免两种解决死锁问题的方案中, 哪种方法有更好的资源利用率, 为什么?

答:

死锁避免有更好的资源利用率。(2 分)

死锁避免不对资源申请方式做任何限制, 进程任何时候按顺序、数量申请资源, 而死锁预防不是这样。(3)

四、教学目标 4（10 分）

1.（10 分）

答：

// 定义两个信号量

sem_t sem1, sem2;

int counter = 1; // 定义计数器

```
void* printHello(void* arg) {
    for (int i = 0; i < 5; i++) {
        sem_wait(&sem1); // 等待信号量 sem1
        printf("Hello%d", counter);
        sem_post(&sem2); // 释放信号量 sem2
    }
    return NULL;
}
```

```
void* printWorld(void* arg) {
    for (int i = 0; i < 5; i++) {
        sem_wait(&sem2); // 等待信号量 sem2
        printf("World%d", counter);
        counter++; // 增加计数器
        sem_post(&sem1); // 释放信号量 sem1
    }
    return NULL;
}
```

```
int main() {
    pthread_t thread1, thread2;

    // 初始化信号量
    sem_init(&sem1, 0, 1); // 初始值为 1, 表示线程 1 可以先执行
    sem_init(&sem2, 0, 0); // 初始值为 0

    // 创建线程
    pthread_create(&thread1, NULL, printHello, NULL);
    pthread_create(&thread2, NULL, printWorld, NULL);

    // 等待线程完成
    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);

    // 销毁信号量
    sem_destroy(&sem1);
    sem_destroy(&sem2);

    printf("\n");
    return 0;
}
```