# 20-21 final

**1.** Compare and contrast **internal fragmentation** and **external fragmentation**. Explain the circumstances where one might be preferred over the other. (6 marks)

### 内部碎片（Internal Fragmentation）

内部碎片是指在内存分配过程中，由于分配的内存块比实际需要的内存大而造成的内存浪费。这种情况通常出现在固定分区分配方案中，即内存被划分为固定大小的分区，当一个进程请求内存时，会分配一个足够大的分区，如果该分区比实际需求大，就会产生内部碎片。

### 外部碎片（External Fragmentation）

外部碎片是指在内存分配过程中，由于多次分配和释放内存块，导致内存中出现许多小的、不连续的空闲块，虽然总的空闲内存足够，但无法分配给新的进程所需的连续内存块。

- 内部碎片优先：当系统中的进程请求内存大小变化不大，并且请求次数较少时，使用固定分区分配或页式存储管理可以减少管理复杂性，虽然会产生一些内部碎片，但总体上管理效率较高。
- 外部碎片优先：当系统中的进程请求内存大小差异较大，并且请求次数频繁时，使用可变分区分配或段式存储管理可以更灵活地利用内存，尽管会产生外部碎片，但可以通过内存压缩或其他内存管理技术来减轻其影响。

**2.** What is a **virtual machine**? Briefly discuss each of the two components that make up a virtual machine. (6 marks)

虚拟机是一种软件计算机，它在物理计算机上运行，能够模拟一台独立的物理计算机。虚拟机能够运行操作系统和应用程序，就像它们在一台真正的物理机器上运行一样。虚拟机提供了一种隔离的环境，使得不同的操作系统和应用程序可以在同一台物理机器上并存且互不干扰。

主机（Host）

主机是指运行虚拟化软件（如虚拟机监视器或Hypervisor）的物理计算机。主机提供了底层的硬件资源，如CPU、内存、存储和网络连接，这些资源被虚拟化软件分配给虚拟机

客机（Guest）

客机是指运行在虚拟机内的操作系统和应用程序。每个客机都认为自己运行在独立的硬件上，尽管实际上它们共享主机的物理资源。

**3.** The **file system** resides permanently on secondary storage. Briefly discuss each of the three major methods of allocating disk space. (12 marks)

1. 连续分配（Contiguous Allocation）

连续分配是将文件存储在连续的磁盘块中，即文件的所有块在磁盘上是连续排列的。文件目录表记录文件的起始位置和文件长度。

优点：

- 访问速度快：由于文件是连续存储的，顺序访问速度非常快。
- 简单性：文件管理和空间分配算法简单，容易实现和维护。

缺点：

- 外部碎片：随着文件的不断创建和删除，会产生外部碎片，导致磁盘空间浪费。
- 文件大小限制：如果文件需要的空间超过了连续的可用磁盘块，可能无法分配给文件。

2. 链接分配（Linked Allocation）

链接分配是将文件存储在任意的磁盘块中，每个块包含文件的一部分，同时指向下一个块的位置。文件目录表记录文件的起始块位置，文件的每个块包含一个指向下一个块的指针。

优点：

- 没有外部碎片：文件块可以存储在任意位置，不会因为空间不足而无法分配。
- 文件大小灵活：文件可以动态扩展，不受连续空间的限制。

缺点：

- 访问速度慢：由于需要通过指针逐块访问，顺序访问速度较慢。
- 可靠性问题：如果某个块的指针损坏，可能导致文件的其余部分无法访问。

3. 索引分配（Indexed Allocation）

索引分配使用一个索引块来存储文件的所有块地址，索引块中包含指向文件各个块的指针。文件目录表记录文件的索引块位置。

优点：

- 快速访问：可以直接通过索引块访问任意一个文件块，访问速度较快。
- 灵活性：文件块可以分散存储，不受连续空间的限制。

缺点：

- 索引开销：每个文件都需要一个索引块，增加了额外的存储开销。
- 索引块限制：对于大文件，索引块可能不足以存储所有指针，需要多级索引。

**4.** Describe in your own words how a parent and child are created from the **fork ()** system call and why you think the child process will or will not continue if the parent process is terminated. (6 marks)

使用 `fork()` 创建进程

当一个进程调用 `fork()` 时，操作系统会创建父进程的一个副本。子进程会获得一个唯一的进程 ID (PID) 以及自己的内存空间。

如果父进程终止，子进程可以继续执行。因为子进程是一个独立的进程，具有自己的 PID。操作系统会将孤儿进程（即父进程终止后还在运行的子进程）重新分配给 init 进程（通常 PID 为 1）或另一个合适的父进程。

**5.** Describe in your own words why **mutual exclusion** is necessary for multiprogramming systems. (4 marks)

防止竞态条件（Race Conditions）

确保数据一致性

防止死锁和饥饿

提高系统稳定性和可靠性

**6. System protection** is multifaceted. What are the four levels of **security** measures that are necessary for system protection? Name and explain them. (12 marks)

# Security Measure Levels

Security measures at four levels:

- **Physical**
  - Data centers, servers, connected terminals
- **Human**
  - Avoid *social engineering*, *phishing* (involves sending an innocent-looking e-mail), *dumpster diving* (searching the trash or other locations for passwords), *password cracking*, etc.
- **Operating System**
  - System must protect itself from accidental or purposeful security breaches: *runaway processes* (DOS denial of service), *memory-access violations*, *stack overflow violations*, *launching of programs with excessive privileges*, etc.
- **Network**
  - protecting the network itself from attack and protecting the local system from attacks coming in through the network (intercepted communications, interruption, DOS, etc.).

## QUESTION II. CPU scheduling, Memory management, Disk scheduling

**(34 marks)**

**1.** Consider the following scenario of processes:

| Process | Arrival time | Burst time |
|---|---|---|
| P1 | 0 | 9 |
| P2 | 1 | 5 |
| P3 | 2 | 3 |
| P4 | 3 | 4 |

Draw the Gantt chart for the execution of the processes, showing their start time and end time, using ***First-Come First-Served FCFS*** scheduling algorithm. (2 marks)

Calculate average turnaround time, and average waiting time for the system in ***First-Come First-Served FCFS*** scheduling algorithm. (8 marks)

**2.** There is a system with **64 pages** of **512 bytes** page size and a physical memory of **32 frames**. How many bits are required in the logical and physical address? (6 marks)

**3.** Calculate the number of page faults for the following reference string using **First-In, First-Out FIFO algorithm** with frame size as **4**.

**5 0 2 1 0 3 0 2 4 3 0 3 2 1 3 0 1 5**

(6 marks)

**4.** Consider a disk queue with I/O requests on the following cylinders in their arriving order:

**54, 97, 73, 128, 15, 44, 110, 34, 45**

The disk head is assumed to be at cylinder number **23**.

Write the sequence in which requested tracks are serviced using **Shortest-Seek-Time-First (SSTF) algorithm** and calculate the total head movement (in number of cylinders) incurred while servicing these requests. (5 marks)

**5.** Consider a **real-time system** in which there are three processes. Their period and execution time are as follows:

| Process | Execution time, t | Period, p |
|---|---|---|
| P1 | 35 | 100 |
| P2 | 10 | 50 |
| P3 | 30 | 150 |

Calculate the total utilization of CPU. (1 mark)

We assume that all three processes are released at time 0. Explain the **Rate Monotonic Scheduling Algorithm** of the processes. (4 marks)

Show the processes on timing diagram. (2 marks)

## QUESTION III. Resource allocation (10 marks)

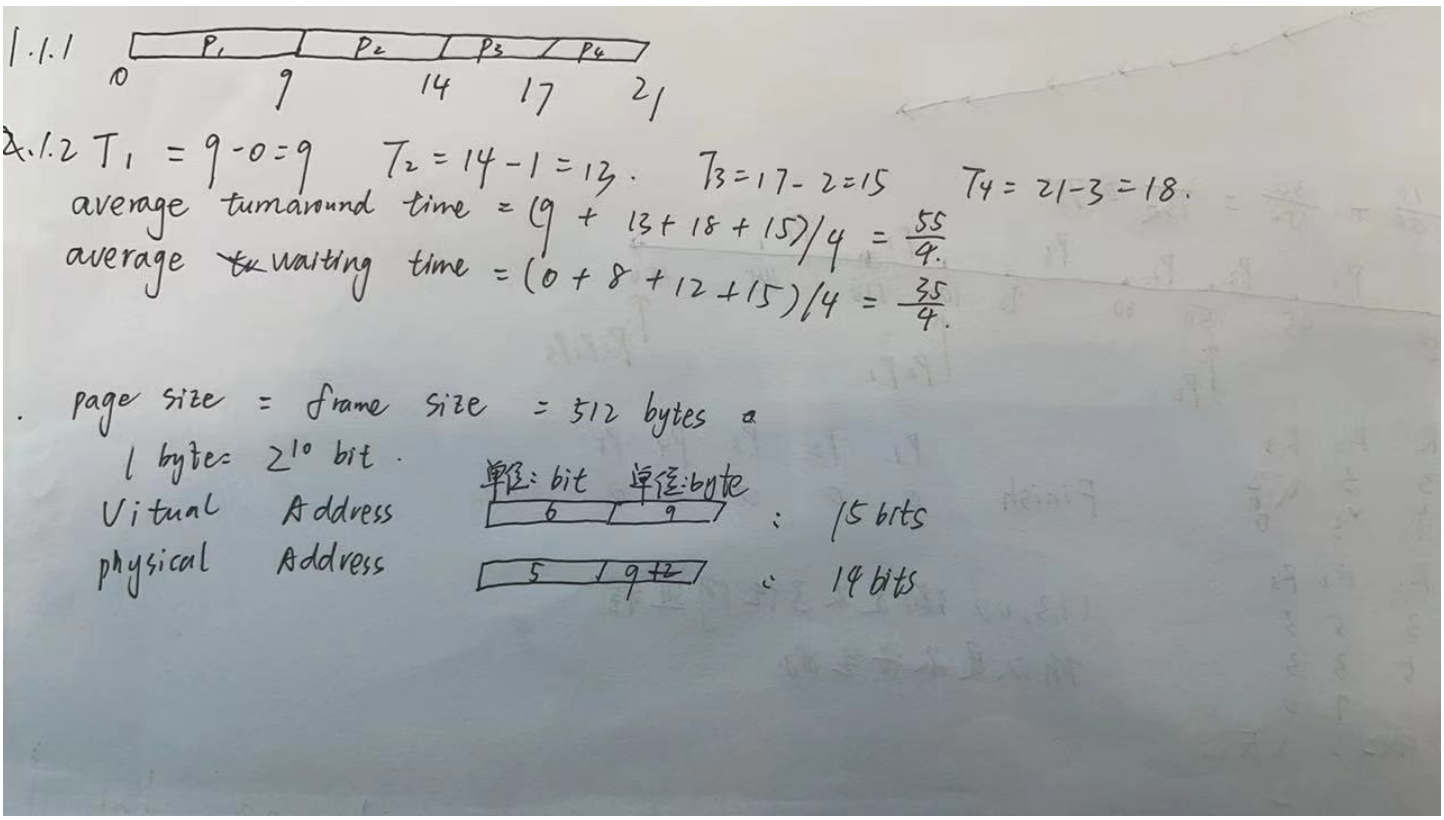
Consider a system with the following information and assume that the system is in safe state.

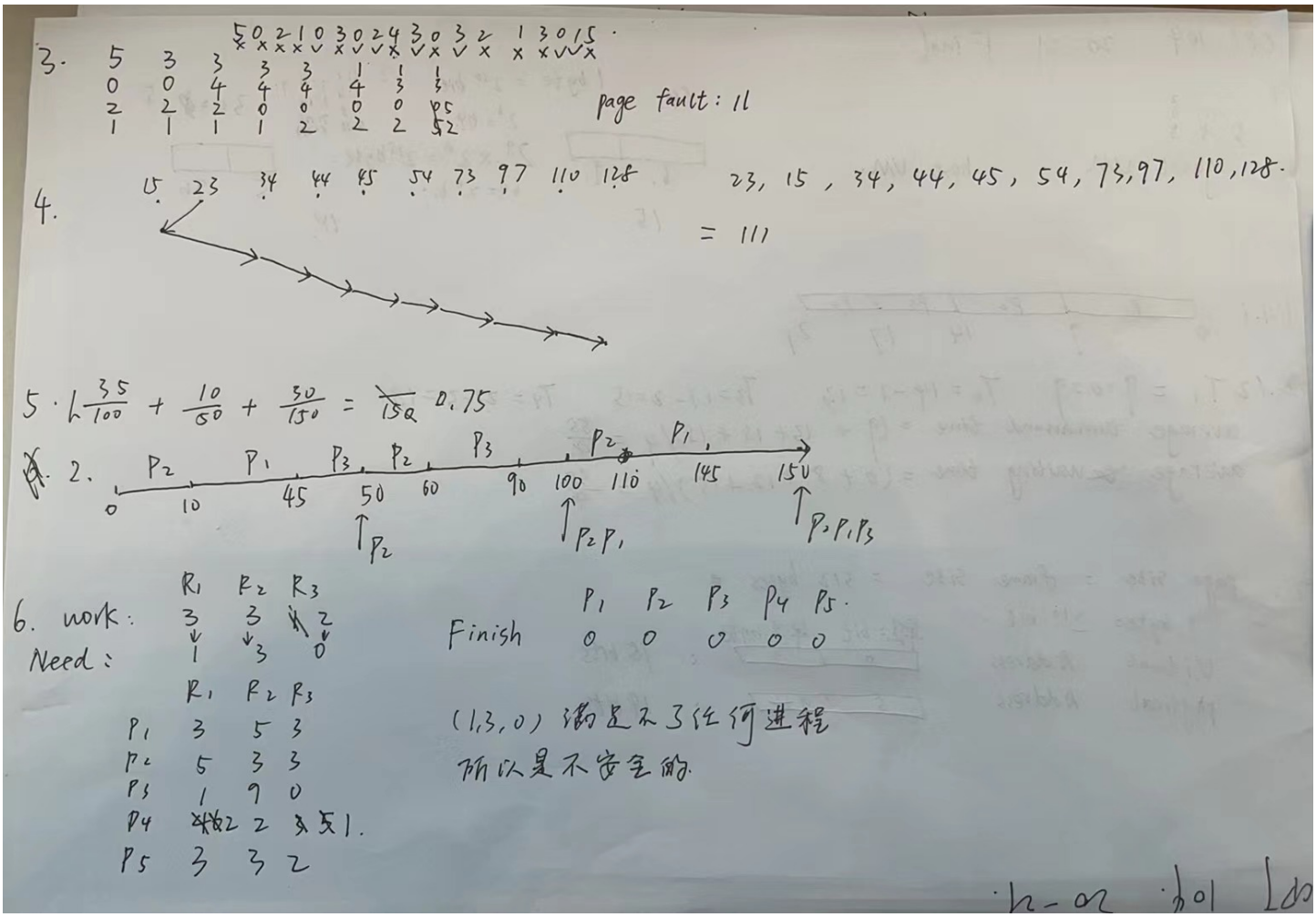
At this moment, if **P4** requests two more instances of **R1** and two instances of **R3**, will the system still be in safe state? Explain the answer.

**Total resources**

| R1 | R2 | R3 |
|---|---|---|
| 15 | 8 | 8 |

| Process | Max | | | Allocation | | |
|---|---|---|---|---|---|---|
| | R1 | R2 | R3 | R1 | R2 | R3 |
| P1 | 5 | 6 | 3 | 2 | 1 | 0 |
| P2 | 8 | 5 | 6 | 3 | 2 | 3 |
| P3 | 4 | 9 | 2 | 3 | 0 | 2 |
| P4 | 7 | 4 | 3 | 3 | 2 | 0 |
| P5 | 4 | 3 | 3 | 1 | 0 | 1 |






## QUESTION IV. Operating System in C Language (10 marks)

If the statement **while** *flag*[j] **and** *turn* = *j* in Peterson's algorithm is changed to **while** *flag[j]* **or** *turn* = *j*, which properties of **Critical Section** implementation are violated by the resulting system? Explain.

```
int turn;
boolean flag[2];
        do
        {
        flag[i] = true;
        turn = j;
        while (flag[j] && turn == j);
        // critical section
        flag[i] = false;
        // remainder section
        }
        while (true);
```

Peterson 算法的原始设计确保了互斥和进程调度属性。原始条件 `while (flag[j] && turn == j);` 确保了以下几点：

- 如果一个进程想进入临界区，它会设置自己的 `flag` 为 `true` 并将 `turn` 设为另一个进程的编号。
- 如果两个进程同时尝试进入临界区，由于 `turn` 变量的存在，只有一个进程可以成功进入临界区，而另一个进程会等待，直到第一个进程离开临界区。

而修改后的条件 `while (flag[j] || turn == j);` 破坏了这种协调机制，导致两个进程可能会同时进入临界区或者都被阻塞，违反了互斥和进程调度的原则。