OS hw1

T1

> 请分别从系统和用户的角度，阐述操作系统的功能，并具体描 述操作系统需要提供哪些服务

从系统的角度来看操作系统的主要功能：

1. 管理硬件资源：操作系统需要管理计算机的硬件资源，包括处理器、内存、磁盘、网络等，以保证它们能够高效地协同工作。
2. 提供应用程序接口：操作系统需要为应用程序提供接口，使它们可以访问硬件资源和其他系统服务。这些接口包括系统调用、库函数和设备驱动程序等。
3. 提供文件系统：操作系统需要管理计算机上的文件和目录，以便用户可以方便地创建、读取、修改和删除文件。
4. 提供网络支持：操作系统需要提供网络支持，使计算机能够与其他计算机通信，包括网络协议、网络接口和网络服务等。
5. 提供安全保护：操作系统需要提供安全保护机制，保护计算机和用户的数据不受恶意软件和黑客的攻击。

从用户的角度来看操作系统的主要功能：

1. 提供友好的用户界面：操作系统需要提供一种易于使用的用户界面，使用户可以方便地执行各种任务，如启动应用程序、打印文档等。
2. 管理应用程序：操作系统需要管理应用程序的安装、启动、关闭和卸载等操作，以便用户可以方便地使用它们。
3. 管理文件和文件夹：操作系统需要管理文件和文件夹，使用户可以方便地组织和访问自己的数据。
4. 提供网络连接：操作系统需要提供网络连接，使用户可以连接互联网，浏览网页、发送电子邮件等。

操作系统需要提供服务：

1. 进程管理服务：管理应用程序的创建、运行、暂停和终止等操作。
2. 内存管理服务：管理计算机内存的分配、回收和交换等操作。
3. 文件管理服务：管理文件和文件夹的创建、读取、写入和删除等操作。
4. 设备管理服务：管理计算机设备的驱动程序和设备资源的分配和回收等操作。
5. 网络管理服务：管理网络协议和网络连接的建立和关闭等操作。
6. 安全管理服务：提供身份验证、权限控制和数据加密等安全机制，以保护计算机和用户的数据安全。

T2

> 请阐述 multi-programming 和 multi-tasking 的概念与设计目的

Multi-programming

概念：

在计算机内存中同时存储多个程序，并让它们在同一时间内轮流执行。在这种情况下，操作系统会分配给每个程序一个时间片，也就是一小段时间的CPU时间，然后让它们依次执行，以此类推。当一个程序等待某个资源（比如等待I/O操作完成）时，CPU可以立即切换到另一个程序，继续执行。

设计目的：

Multi-programming强调的是让多个程序轮流执行，减少CPU的浪费，提高CPU的利用率

Multi-tasking

概念：

则是指在同一个计算机系统中，同时运行多个任务，这些任务可以是不同的程序，也可以是同一个程序的不同实例。在这种情况下，操作系统会将CPU时间切分为多个时间片，同时为每个任务分配一个时间片，并根据优先级和任务的需求进行调度，使得各个任务可以并发执行。例如，在OS中，我们可以在typora中编辑文档的同时播放音乐。

设计目的：

Multi-tasking强调的是在同一时间内让多个任务并发执行，满足用户多任务处理的需求，提高系统的响应速度和用户体验。

T3

> 请阐述缓存的思想以及工作原理

思想：

以空间换取时间，将经常访问的数据存储在快速访问的地方，以便在下一次访问时能够更快地获取

工作原理：

缓存的工作原理可以简单地描述为“缓存命中”和“缓存未命中”。当程序需要访问数据时，先检查缓存中是否已经存储了该数据。如果缓存中已经存在该数据，则称为“缓存命中”，数据可以快速地被访问。如果缓存中没有该数据，则称为“缓存未命中”，此时程序需要从慢速存储介质（例如磁盘或网络）中获取数据，并将其存储到缓存中以供以后访问。

T4

> 请阐述什么是系统调用，以及系统调用与 API 的逻辑关系

系统调用

系统调用是操作系统提供给用户程序的一种接口，通过这种接口，用户程序可以请求操作系统执行一些特权操作，例如打开文件、读写数据、创建进程等。系统调用提供了一种机制，让用户程序能够在保持操作系统安全的前提下，访问操作系统底层的功能。

系统调用与API的逻辑关系

API（应用程序编程接口）是一组定义在软件中的接口，用于访问软件提供的功能。系统调用是操作系统提供的一种接口，也是 API 的一部分。

一般来说，API 是一组库函数，这些函数提供了一些高层次的抽象接口，以方便用户程序进行开发。这些库函数通常会调用底层的系统调用来完成它们的工作。例如，一个文件操作的库函数可能会提供打开、读取、写入等操作的高层次接口，而这些操作在内部会通过系统调用来实现。

因此，系统调用和 API 是密切相关的。API 通常会封装底层的系统调用，以方便用户程序进行开发。在操作系统中，系统调用是实现 API 的基础。

T5

> 阐述 Dual Mode 的工作机制，以及采用 Dual Mode 的原因

工作机制

分为用户模式和内核模式，在硬件中提供模式位(Mode bit)使得能够分辨系统是在运行那种模式

有一些指令被认为是特权的，只能在内核模式下执行；特别地，执行系统调用时会设置为内核模式（需要访问内核空间），而在系统返回时重置为用户模式。

而在用户模式下运行时，只能访问用户空间资源。

原因

通过双重模式，特权指令只能只有操作系统内核才能执行，而用户空间不可执行，因此可以有效地保护系统内核代码和系统其他组件的安全性和稳定性，防止用户空间应用程序对内核系统进行非法的修改或者访问

T6

> 分别阐述 Monolithic 大内核结构，层次化结构，模块化结构和微内核结构的特点和优劣

Monolithic 大内核结构

特点

Monolithic 大内核结构是最早的操作系统设计结构之一，整个操作系统内核的所有功能都在同一个地址空间中运行。内核将所有系统资源进行管理，例如文件系统、设备驱动程序和网络协议等

优点

- 在内核态和用户态之间的切换开销较小，因此能够更快地执行系统调用和提供服务
- 系统资源共享和通信更加方便和高效，因为所有的功能都在同一地址空间中运行

缺点

- 整个系统的可靠性和稳定性会受到影响，因为一个模块的故障可能导致整个系统崩溃
- 操作系统代码的复杂性较高，因此不易维护和升级

层次化结构

特点

内核分成若干个层次，每个层次都负责不同的功能。上层次的模块依赖下层次的模块，而下层次的模块提供服务给上层次的模块

优点

- 系统的可靠性和稳定性得到提高，因为每个层次的模块只需关注自己的功能，从而减少了模块之间的耦合性
- 操作系统的可维护性和升级性也得到提高，因为每个层次的模块可以独立地进行开发和测试

缺点

- 需要考虑到每个层次之间的通信和数据传输，增加开销
- 层次化结构的内核设计更加复杂，实现难度也更大

模块化结构

特点

将系统分解成多个模块，每个模块负责不同的功能，模块之间通过接口交互

优点

- 代码结构清晰，易于维护和管理
- 每个模块负责不同的功能，模块之间的耦合度低，易于扩展和升级
- 模块之间的接口清晰，可靠性高

缺点

- 模块化结构下，各个模块之间需要频繁的交互，可能会带来一定的性能损失
- 模块化结构下，系统结构比较复杂，可能会增加系统设计和开发的难度

微内核结构

特点

将操作系统的核心功能封装在内核中，其他功能通过外围服务进程实现，内核只提供最基本的服务接口，其他功能以插件形式存在

优点

- 内核的可靠性和安全性高，可以有效避免因为外围服务进程的错误导致整个系统崩溃的情况
- 内核的功能相对简单，易于维护和管理

缺点

- 服务进程的系统调用频繁，性能较低
- 插件的兼容性处理导致代码难度较大

T7

> 举例说明什么是机制与策略分离的设计原则，并说明该设计的好处

设计原则

机制与策略分离是一种软件设计原则，其核心思想是将系统的机制部分和策略部分分离开来，以便能够更好地维护和修改系统。机制通常指系统中用于实现某些功能的具体实现方式，而策略则指控制系统行为的规则和算法。将这两个部分分离开来，可以使得系统更具灵活性和可扩展性

举例：

一个简单的应用程序可能需要计算某些数据，其中涉及到不同的算法和数据源。使用机制与策略分离的设计原则，我们可以将算法和数据源的实现作为机制，而将计算逻辑作为策略。这样，当我们需要修改算法或者更改数据源时，只需要修改机制部分的代码，而策略部分的代码则不需要做任何修改。

好处

- 灵活性：机制与策略分离可以使系统更加灵活，因为修改机制部分的代码不会影响策略部分的代码，从而使得系统更容易适应变化的需求
- 可扩展性：该原则也可以提高系统的可扩展性，因为当需要添加新的功能时，只需要编写新的机制并将其与现有的策略进行组合即可
- 可维护性：分离机制和策略可以使系统更容易维护，因为在修改机制时，不需要考虑影响策略的代码。同时，由于策略与机制之间的接口清晰，因此也更容易诊断和解决问题