1.**操作系统的功能：**

用户视角：

[1] 对于 PC 用户，操作系统的目的是优化用户进行的工作（游戏），使用户使用方便。

[2] 对于大型机或小型机用户，操作系统的目的是优化资源利用率，确保所有的 CPU 时间、内存和 IO 都能得到有效使用，并且确保没有用户使用超出限额以外的资源。

[3] 对于工作站用户，操作系统需要兼顾使用方便性和资源利用率。

[4] 移动计算机资源匮乏，针对可用性和电池寿命进行了优化。

[5] 某些计算机的用户界面很少或没有，例如设备和汽车中的嵌入式计算机。

系统视角：

[1] 操作系统是与硬件紧密相连的程序，可以看作资源分配器，操作系统管理 CPU 时间、内存空间、文件存储空间、IO 设备等资源。面对许多甚至冲突的资源请求，操作系统应考虑如何为各个程序和用户分配资源，以便计算机系统能有效公平的运行。

[2] 操作系统是一个控制程序，管理用户程序的执行，防止计算机资源的错误或不当使用。

- System View
  - OS is a **control program**
    - Controls execution of programs to prevent errors and improper use of the computer
  - OS is a **resource allocator**
    - Manages all resources
    - Decides between conflicting requests for efficient and fair resource use
- User View
  - PC users want convenience, **ease of use** and **good performance**, don't care about resource utilization
  - But shared computer such as mainframe or minicomputer must keep all users happy: maximize **resource utilization**
  - Users of dedicate systems such as workstations have dedicated resources but frequently use shared resources from servers: **tradeoff**
  - Mobile computers are resource poor, optimized for **usability and battery life**
  - Some computers have little or no user interface, such as embedded computers in devices and automobiles

2.multi-programming（多道程序设计）

通过安排作业（编码和数据）使得 CPU 总有一个执行作业，从而提高 CPU 的利用率。

设计目的:解决人机矛盾及 CPU 和 IO 设备之间的速度不匹配的矛盾。

multi-tasking（分时系统）

是 multi-programming 的自然延伸。对于分时系统，虽然 CPU 还是通过切换作业来执行多个作业，但是由于切换频率很高，用户可以在程序运行时与其交互。

设计目的：多个用户通过终端共享一台主机，这些终端连接在主机上，用户可以与主机进行交互操作而互不干扰。

3.存储层次：

寄存器、高速缓存、内存、固态磁盘、硬盘、光盘、磁带

缓存的思想：

信息通常保存在存储系统中（如内存），使用时，它会被临时复制到更快存储系统，即高速缓存。当需要特定信息时，首先检查它是否处于告诉缓存，如果是，可以直接使用高速缓存的信息，如果否，就使用位于源地的信息，同时将其复制到高速缓存以便下次再用。

4.系统调用：

系统调用提供操作系统服务接口，为用户程序提供手段，以便请求操作系统完成完成某些特权任务，是进程和系统内核的编程接口，用户可以通过这个接口访问内核空间。

系统调用与 API 的逻辑关系：

API 是应用编程接口，与内核无关。系统调用通过中断向内核发出请求，实现内核提供的某些服务。系统调用的实现是在内核中完成的，API 是在函数库中实现的。

5. Dual Mode 的工作机制：

分为内核模式和用户模式，计算机硬件通过一个模式位来表示当前模式：内核模式（0），用户模式（1）。用户通过系统调用时切换到内核模式，返回时重置为用户模式。硬件只有在特权模式下才能执行特权指令。

原因：

提供保护手段，防止操作系统和用户程序受到错误用户程序的影响。

6.操作系统需要提供的服务：

用户界面、程序执行、IO 操作、文件系统操作、通信、错误检测、资源分配、记账、保护与安全

7.

	优点	缺点
Monolithic 结构	系统调用接口和内核通信开销很小。	难以实现与维护
层次化结构	隐藏底层实现，易于构建和调试	难以定义各种层次，效率较低
微内核结构	容易扩展，将服务添加到用户空间，无序更改内核。更可靠、更安全	用户空间和内核空间的通信开销大
模块化结构	类似于分层系统，更灵活。类似于微内核系统，更高效。	模块间的接口规定很难满足对接口的实际需求。各模块设计者齐头并进，每个决定无法建立在上一个已验证的正确决定基础上，无法找到可靠的

		决定顺序。模块划分的大小，也会影响模块内聚性和耦合度。

8. 例子：

对于 CPU 定时器机制，可根据用户对响应高低的需求改变定时器时长

该设计的好处：

在机制保持不变的情况下可以改变策略。