1. Влияние размера страницы виртуальной памяти на ОС.Стратегии ОС по работе с виртуальной памятью.虚拟内存页面大小对操作系统的影响。操作系统使用虚拟内存的策略。

开发操作系统时，需要决定页面的大小

需要考虑的总要事项有：

·页面的大小

·内部碎片

·地址转化期间的页面错误数（page-fault）

·与辅助储存器交互的速度（和块的大小）

·数据局部性（在下面的多线程应用程序中）

·TLB未命中次数，TLB大小，L1 L2 L3缓存大小等

现代处理器可以处理不同页面的大小（large, huge pages）

从辅助内存获取页面的策略

·按需预取

安置策略

·在NUMA系统上，靠近进程/线程处理器的速度更快

清理策略（上传到辅助内存）

·按需和预清理

多任务管理

·卸载整个进程

替换策略

管理进程的常见部分

1. Стратегии замещения страниц ОС. Часовой Алгоритм.Управление резидентной частью процесса操作系统页面替换策略。 时钟算法. 进程驻留部分的管理

正在运行的进程中应该选择哪些帧来被其他进程替换和使用？

·我们将卸载近期不会访问的框架

·并非所有页面都可以调出（部分内核、缓冲区……）

·如果一个框架被共享多次（例如共享库代码），那么卸载它可能会影响许多进程

·必须考虑锁定帧

先进先出（FIFO）：最早进入内存的页面被替换。这种策略简单易懂，但可能导致“Belady现象”，即增加页面数量并不一定会减少页面错误率。

最近最少使用（LRU）：替换最近最长时间未被使用的页面。实现时可以通过计数器、栈或者队列等数据结构来实现。

最不经常使用（LFU）：替换最不经常被访问的页面。需要对每个页面进行频率计数。

时钟算法（Clock）：使用一个环形队列的指针，当某个页面被访问时，将该页面的标志位置为“已访问”，当需要替换页面时，从当前指针位置开始，找到第一个“未访问”标志的页面进行替换。

最佳（OPT）：理论上最优的替换算法，选择未来最长时间内不会被访问的页面进行替换。但是实现起来需要对未来的访问模式进行预测，通常是不可行的。

进程驻留部分的管理

内存管理：

内存是计算机系统中的一项关键资源，用于存储程序代码、数据和执行中的进程。

操作系统需要管理内存，以便在有限的内存空间中为多个程序提供服务，并确保它们之间不会相互干扰。

用户空间和内核空间：

大多数操作系统将内存划分为两个部分：用户空间和内核空间。

用户空间用于存储用户程序的代码和数据，而内核空间用于存储操作系统内核和系统服务的代码和数据。

页面调度和置换：

当用户程序需要执行时，操作系统会将其从磁盘加载到内存中。但是，内存是有限的，当内存不足时，操作系统需要选择哪些页面置换出去，以便为新的页面腾出空间。

页面调度和置换算法负责决定哪些页面应该被置换出去。常见的页面置换算法包括最近最少使用（LRU）、先进先出（FIFO）、时钟算法等。

进程调度：

操作系统还需要决定哪些进程可以在内存中运行，并在必要时进行进程切换和上下文切换。

进程调度算法负责决定进程的优先级和调度顺序，以便充分利用CPU和内存资源。

内存分配：

当用户程序请求内存时，操作系统需要分配内存空间给这些程序，并确保它们不会相互干扰。

内存分配算法负责管理内存的分配和释放，以便最大限度地提高内存的利用率。

1. Виды планирования процессов. Критерии краткосрочного планирования. Приоритеты.流程规划的类型，短期规划的标准，优先事项。

规划类型

·长期规划：决定将进程添加到正在运行的进程中

·中期规划：决定将进程添加到完全或部分位于主内存中的进程数量中

·短期规划：决定处理器将执行哪些可用进程

·I/O调度：决定哪个进程中的I/O请求将由可用的I/O设备处理

短期规划的标准:首先他是与性能相关的一组自定义标准.

与性能相关的标准:周转时间,即工作的总时间,响应时间,截止时间-处理时间限制(此标准常用于实时系统) ,可预测性

此外系统必须依赖与系统性能相关的短期规划标准: 吞吐量-带宽,处理器利用率-处理器负载,公平-公平,强制执行优先级-强制优先级排序,平衡资源-负载平衡

进程被分配了数字优先级, 不同的操作系统使用不同的优先级方案

1. Использование приоритетов.使用优先顺序

为了隔离进程,现代所有操作系统都使用优先级,将优先级高的先执行,把优先级相同的进程可以位于同一个队列中,然后可以进一步选择. 如果进程的优先级较低,会发生饥饿,导致进程无法获得处理器,并且执行速度大幅降低. 现代操作系统可以动态更改. 还有优先级继承,避免了优先级倒置的问题.

1. Стратегии планирования FCFS, RR, SPN, SRT, HRRN,Feedback.规划策略

FCFS(先到先得)策略:选择等待时间最长的策略

RR循环策略:每个进程被给予相同时间以一定顺序循环.

SPN:更短的进程向前移

SRT:最短时间完成

HRRN:最高响应率

反馈:根据当前进程的操作状态更改优先率的一种策略 如果进程主动使用处理器,那么他的优先级会降低 所以进程在I/O上等待的时间越长,优先级越高.(使用进程的反馈将进程拉回执行区域)

1. сро. Feedback планировщик и классы планирования ОС UNIX SVR4

сро. 反馈调度程序和调度类 UNIX SVR4

反馈调度器：

UNIX SVR4 中的反馈调度器是一种动态调度算法，其特点是根据进程的执行情况动态调整进程的优先级。

反馈调度器通过监视进程的运行情况来收集反馈信息，例如执行时间、等待时间等，并根据这些信息动态调整进程的优先级。

如果一个进程执行时间较长或者经常被阻塞，那么它的优先级可能会下降；相反，如果一个进程执行时间较短或者经常处于就绪状态，那么它的优先级可能会提升。

调度类别：

在 UNIX SVR4 中，进程被分为不同的调度类别，每个类别有不同的调度要求和优先级。

常见的调度类别包括实时进程、交互式进程和批处理进程等。

实时进程通常具有较高的优先级，以确保它们能够及时响应外部事件；交互式进程通常具有中等优先级，以保证用户体验流畅；批处理进程通常具有较低的优先级，以确保它们不会占用过多系统资源。

1. Справедливое планирование.公平规划

公平调度是一种调度概念,允许我们以捆绑方式分配处理器时间.把任务按比例分配给各个组.如果工程组没有任务,系统中的其他组就会占用他的时间量,根据初始的负载公平分配给其他组.

一旦任务来自工程师,他们就会拿走他们在总处理器能力中的份额,包括所有处理器.

1. Планирование в многопроцессорных системах. Типы многопроцессорных систем с точки зрения организации планирования. Гранулярность и проектирование планировщиков процессов и потоков для многопроцессорных систем.多处理器系统中的调度。 从规划组织的角度来看多处理器系统的类型。 多处理器系统的进程和线程调度程序的粒度和设计。

多处理器调度:松耦合(分布式多处理器丶集群)

松耦合系统:有自己的主存储器和输入输出结构的独立系统合集.

在这个设备集群中,必须在系统之间分配任务.

确保一般数据和结果的确切存储位置.

功能专用的多处理器系统:存在协调和以某种方式分配任务的特殊处理器

执行计算的从属处理器的可用性.

紧耦合处理器: 拥有共享内存和缓存的系统,其中每个内核都有自己的一级缓存和共享的二级缓存,第三缓存级别,联合处理器上的所有内核.由一个操作系统控制

多处理器系统的进程调度:对于多处理器,调度原则的选择不如在单处理器中重要.因此在多处理器中系统中使用简单的FCFS就足够了.

线程调度:

负载分配:进程不分配到某个特定的处理器,每个处理器只要空闲就从队列中选一个线程.

组调度:一组相关的线程基于一对一原则,同时调度到一组处理器上运行.

专用处理器分配:这种方法与负载分配相反,它通过线程指定到处理器来定义隐式的调度.每个程序执行时,都分配一组处理器,处理器数量和线程数相等.程序结束,处理器返回处理器池,等待分配给下一个程序.

同步粒度是系统中进程之间同步的频率.

粒度有:

很好(Fine):在单个机器指令级别执行并计算时.这种情况下,同步程度使下一次同步发生平均需要十几个或者几十个指令. 细粒度只存在特定的应用程序:游戏,视频等

平均(Medium)粒度是普通应用程序的典型,当有20~200条命令时,执行完后线程必须相互同步.

不同进程相互交互时通常使用粗粒度级别,例如,共享内存区域

非常粗(Very Coarse)粒度应用于分布式系统,当有集群向另一个集群发消息时,很少需要同步(2000~1000000)个命令内.

独立粒度(indenpendent)-典型用于完全没有同步的独立系统,但是同步可以在操作系统级别进行.

静态分配,在处理器分配到整个进程时执行.上下文切换相关的消耗减少了,进程运行的更快.

动态分配:在系统内创建一个大型进程队列,然后分配到处理器之间. 所有进程功的动态共享队列,如果进程相互关联,这种分布就会无效.

动态负载平衡系统:允许在多处理器系统上工作时将进程从一个处理器转移到其他处理器的节点

1. ОС реального времени и планировщики.Deadline-планирование.实时操作系统和调度程序。最后期限规划。

负载共享:允许使用一个全局队列调度所有进程,优点:在处理器之间平均分配负载.这种方法的性能负面影响:中央队列锁(当调度算法本身需要重新配置,有必要获取锁), 并且进程和处理器经常会遇到阻塞,频繁的缓存出错是难免的.

组调度(Gang Scheduling)--线程被分配到相关的处理器,这种方法可以大大减少细和中等粒度的”开销”.处理器在相同的缓存上工作,可以帮助我们避免不必要的内存访问.

专用处理器分配:分配的处理器池等于线程数.这将增加处理器的停机时间,这种调度原则完全消除开关,大大提高了工作速度,因为每个进程都将在系统的最短时间内执行.

动态调度:如果调度器已经在进程之间分配了处理器,那么在新任务到来且没有空闲资源的情况下,可以从多处理器的进程中收回一个处理器并分配给新进程.

实时操作系统分为:硬实时和软实时

硬实时(Hard RealTime)是一个要求非常高的系统,没有按时完成会出现非常严重的后果,提前完成则不会出事情.

实时系统有一个重要特征:频率,所有实时都被划分为段,在段结束前应该检查内容是否完成并作出一些决定.

实时系统的要求:确定性-明确定义中断发生的时间,在处理前报告事情,还会确定最后期限.

响应时间(Responsiveness)---确定接受信号到产生响应所需的时间

用户控制--对操作系统提出要求,用户必须足够精细才能定义进程的特征,以便调度程序可以更好的执行. 实时调度程序支持用户更改大量的参数.

可靠性(reliability)---任何实时系统与外部对象一起工作,因此比传统操作系统更可靠

Fail-soft operation 软故障操作--如果发生错误,隔离有错误的部分,使系统有错误的部分变成未使用的部分,并尝试修复.并且不适用内核转储.

Fail-save 故障保存 防止了错误的可能性和后果

实时系统对调度程序的要求还意味着更高优先级的任务到达的情况下严格使用抢占优先级,以及最小化调度中的延迟.

期限(Deadline)调度程序:

就绪时间--任务准备好执行的时间

开始截止时间--任务开始的时间限制;

完成期限--任务完成的时间限制

Processing Time-完成任务所需的时间

资源需求--完成任务所需的资源清单

优先级

子任务结构--将进程划分为若干子任务.可以标记任务中那些是必要的,那些是可选的

1. Проблема инверсии приоритетов, типы инверсии и способы решения в планировщике.调度器中的优先级倒置问题、倒置类型及解决方案。

优先级反转问题:

有界优先级反转:当一个高优先级进程需要公用的内核锁,但是被低优先级的进程占用,那么高优先级的进程需要等低优先级进程的锁被释放,才能执行,但是低优先级的占用的时间可能很长.

无界优先级反转:基于上面的情况,又出现一个进程T3基于两者之间,他会导致低优先级的进程从处理器中移位到等待区,原则上不行启动,因为处理器的时间被高优先级的进程T3占用.

解决方法:

优先级继承:当出现高优先级进程因为低优先级占用资源阻塞,就先讲低优先级提升到等待进程中的最高级

最高优先级:将低优先级进程提升到当前资源的最高优先级.

1. Ввод-вывод. Современные устройства и скорости обмена, развитие способов ввода-вывода, логическая структура ввода-вывода. 输入-输出。 现代设备和汇率、投入产出方法的发展、投入产出的逻辑结构

可编程I/O:处理器通过总线和寄存器之间控制外设或连接到总线并具有一组控制寄存器的控制器.

I/O使用中断:在控制器中添加了中断.不包括等待.

直接内存访问:

控制器中增加寄存器和计数器,保证控制器中的缓冲区转移到内存区

控制器变成带有处理器和命令系统的独立计算模块(输入和输出的通道)

控制器通过额外的一组设备(I/O处理器)控制I/O

为了实现直接内存访问(Direct Memory Access),在控制器结构中加入了寄存器和计数器,以保证控制器缓冲区到内存区的转移.

控制器有了缓冲区,现代操作系统必须将其映射到用户地址空间.或者将此缓冲区复制到其他系统区域(缓冲区的缓存中),然后交给用户.以便用户接受必要的数据并作出决定,之间访问内存这种传输(将数据复制到公共内存区域)可以由控制器本身执行.

控制器进一步发展,数据传输效率有所提高,控制器渐渐的变成了复杂的设备,成为计算模块(I/O通道),由单独的命令系统控制.

要进行通信,进程必须有逻辑I/O模块,它必须从操作系统中侦听改模块,内核必须用Logical I/O来组织信息交换的逻辑,确定数据传输的顺序和相应的缓冲区. 意思就是提供与设备通信的dev I/O设备驱动程序,一级提供I/O调度的模块.

如果有问题的程序进行网络交换,那么内核内部有很大一部分成为TCP/IP(通讯加购)

通讯架构在驱动程序方面有非常复杂的模块化架构,TCP/IP中的基础设备频繁更换,导致在驱动程序更改方面交换非常困难

I/O请求和处理器一样是一种资源,计算设备发展的深入,它们能处理的输入和输出量也越大. 硬件也会自我调整.

1. Буферизация ввода вывода. Ввод-вывод в UNIX SVR4. I/O缓冲。 UNIX SVR4 中的 I/O

全缓冲：数据会被先存储在内存的缓冲区中，直到缓冲区满了才进行实际的 I/O 操作。

行缓冲：数据会被存储在内存的缓冲区中，直到遇到换行符（'\n'）才进行实际的 I/O 操作。

无缓冲：数据不会被缓存，每次操作都会直接进行实际的 I/O 操作。

在 UNIX SVR4 中，I/O 操作是通过文件描述符来实现的。文件描述符是一个整数，代表了打开文件或其他 I/O 设备的引用。UNIX SVR4 提供了一系列系统调用和库函数来进行输入输出操作，包括 open、close、read、write 等。这些函数会在内核中触发相应的 I/O 操作，而具体的 I/O 缓冲策略则由操作系统来管理和实现。

1. Диски и дисковое планирование.磁盘和磁盘调度。

硬盘（磁盘）：

硬盘是计算机系统中用来存储数据的主要设备之一，它具有大容量、持久性和相对较低的成本等特点。

硬盘通常被分为多个扇区和磁道，操作系统通过逻辑块号（LBA）来管理和寻址这些扇区，从而实现对数据的读写操作。

磁盘调度：

磁盘调度是指操作系统中用来管理对硬盘访问的过程。由于硬盘的访问速度相对较慢，磁盘调度算法的目标是尽可能地减少硬盘的寻址时间和旋转延迟，以提高磁盘的性能和效率。

常见的磁盘调度算法包括先来先服务（FCFS）、最短寻道时间优先（SSTF）、扫描（SCAN）、循环扫描（C-SCAN）、LOOK、C-LOOK 等。

磁盘缓存：

为了进一步提高磁盘的性能，操作系统通常会使用磁盘缓存来缓存最近访问过的数据块。磁盘缓存可以减少对物理硬盘的实际访问次数，加快数据的读写速度。

磁盘缓存通常位于内存中，大小由操作系统动态管理。

优化策略：

磁盘调度算法的选择取决于不同系统的需求和性能目标。某些算法可能更注重响应时间，而另一些算法可能更注重吞吐量和效率。

操作系统的磁盘调度器会根据具体的情况选择合适的磁盘调度算法，并根据系统的负载和性能指标动态调整。

1. Концепции RAID. RAID概念

（Redundant Array of Independent Disks——独立磁盘冗余阵列）。

1. RAID 是物理磁盘的集合，操作系统将其视为单个逻辑磁盘。

2. 数据分布在阵列的物理磁盘上。

3、多余的磁盘容量用于存储控制信息，以保证磁盘发生故障时数据的恢复。

拥有更多磁盘可以提高性能，但会增加发生故障的可能性。 这就是 RAID 提供存储的原因

数据恢复信息。

使用多个磁盘而不是一个磁盘是因为辅助存储设备的性能增长速度比其他组件慢，并且多个磁盘允许同时执行多个 I/O 操作。

在七个 RAID 级别中，最常用的是 0、1、5 和 6。

RAID 2 和 3 不被使用，因为它们设计用于容易出错的环境，而磁盘的可靠性很高。

1. RAID-0, 1, 10, 0+1

0 不是真正的 RAID 级别，因为它不以任何方式使用冗余。 RAID-0 的主要优点是其非常高的 I/O 速度。 这是因为数据被剥离并立即写入所有可用磁盘

由于这种写入方式，损坏一个磁盘会导致所有数据损坏。

1 镜像数据（镜像、创建镜像）。 冗余不是通过计算实现的，而是简单地通过数据复制来实现的。 这使用与 RAID-0 相同的方案。

优点包括：

易于数据恢复，能够在磁盘之间进行选择（访问时，选择因旋转而导致的延迟最小且搜索时间最短的磁盘。

记录可以在两个磁盘上并行更新（因此时间由较慢操作的速度决定）

缺点是它需要两倍的磁盘，因此 RAID-1 仅用于最重要的文件。

RAID-1 的读取性能更好，在面向事务的环境中速度可以是 RAID-0 的两倍。

RAID 10（也称为 RAID 1+0）是一种基于 RAID（Redundant Array of Independent Disks，独立磁盘冗余阵列）技术的存储方案。RAID 10 结合了 RAID 1 和 RAID 0 的特性，提供了高性能和数据冗余的优势。在 RAID 10 中，磁盘组被分成至少四个硬盘，并且数据被复制到多个硬盘上。

1. RAID 4,5,6. Аппаратные дисковые массивы.硬件磁盘阵列

RAID 4：

RAID 4 是一种 RAID 级别，其特点是使用分布式奇偶校验（Distributed Parity）来提供数据冗余。

在 RAID 4 中，数据被分成块，并且每个数据块的奇偶校验信息都存储在一个独立的硬盘上。这个硬盘通常被称为奇偶校验盘（Parity Disk）。

RAID 4 的读性能通常很好，但写性能可能会受到奇偶校验盘的性能瓶颈的影响。

RAID 5：

RAID 5 是一种使用分布式奇偶校验的 RAID 级别，其特点是将奇偶校验信息分布存储在所有硬盘上，而不是单独存储在一个硬盘上。

在 RAID 5 中，数据块和对应的奇偶校验信息被交错存储在多个硬盘上。这种分布式存储提高了写入性能，因为奇偶校验信息的更新可以并行进行。

RAID 5 提供了较好的读性能和中等的写性能，同时具有较高的数据冗余性，即使一个硬盘损坏，数据也可以从其他硬盘和奇偶校验信息中恢复。

RAID 6：

RAID 6 是 RAID 5 的扩展版本，其特点是在每个数据块上使用两个独立的奇偶校验信息。

RAID 6 可以容忍两个硬盘的故障而不丢失数据，提供了更高的数据冗余性。

由于需要计算和存储两个奇偶校验信息，RAID 6 的写入性能可能比 RAID 5 略低。

硬件 RAID 阵列：

硬件 RAID 阵列是通过专用的 RAID 控制器硬件来实现 RAID 功能的存储解决方案。

硬件 RAID 控制器负责管理 RAID 阵列中的数据读写操作，并提供了额外的处理能力和缓存，以提高性能。

硬件 RAID 阵列通常比软件 RAID 更可靠和高效，因为它们可以在硬件级别上执行 RAID 操作，而不会占用主机 CPU 的处理能力。

1. Файловый ввод-вывод, основные определения. Задачи ОС по управлению файлами. Совместное использование файлов.文件 I/O，基本定义。 用于管理文件的操作系统任务。 文件共享。

文件是具有以下属性的数据的集合：

– 结构的可用性（可以很复杂，一个文件中可以有多个）

– 长期存在的可能性

– 进程之间共享的可能性

● 基本文件操作

– 创建、删除、打开、关闭、读、写、位置选择、控制操作、阻塞

字段是记录的单个元素，通常是一个值。

记录 - 一组字段

文件是与同质数据集相关的记录的集合。

数据库是一个具有复杂的、相互依赖的字段和记录结构的文件。

对文件执行的操作会影响其结构

文件可能没有结构（例如字符流）

文件管理的操作系统任务：

● 能够存储文件并对其执行用户操作： 不

– 创建、删除、读取和更改文件

– 管理您自己和其他用户对文件的访问

– 在文件之间移动数据

– 执行文件备份和恢复

– 提供通过用户友好的名称处理文件的能力

● 保证数据正确性

● 确保可接受的性能

● 支持多种类型的存储设备

● 最大限度地减少或消除数据丢失和损坏

● 提供一组基本的I/O 功能

● 启用文件共享

文件的共享：

文件的共享是指多个进程或程序同时访问同一个文件的情况。在多用户和多任务环境中，文件的共享是非常常见的。

操作系统通过提供文件锁（File Lock）机制来实现对文件的共享和保护。文件锁可以防止多个进程同时对同一个文件进行写操作，从而避免数据的混乱和丢失。

文件的共享使得多个进程可以共同使用和修改文件，提高了系统的灵活性和效率。

1. Управление файлами в UNIX SVR4. UNIX SVR4中的文件管理

文件系统：

UNIX SVR4 使用基于层次结构的文件系统来组织文件和目录。文件系统以树状结构的形式存在，根目录为树的顶部，子目录和文件则位于树的分支和叶子节点。

UNIX SVR4 文件系统支持多种文件系统类型，包括标准的 UFS（Unix File System）、NFS（Network File System）等。

文件和目录管理：

UNIX SVR4 提供了一系列命令和系统调用来进行文件和目录的管理。例如，ls 命令用于列出目录中的文件和子目录，cp 命令用于复制文件，mv 命令用于移动文件，rm 命令用于删除文件等。

操作系统还提供了一系列系统调用，如 open、read、write、close 等，用于在程序中进行文件的打开、读写和关闭操作。

权限和保护：

UNIX SVR4 使用基于权限的文件访问控制机制来保护文件的安全性。每个文件和目录都有一个拥有者和一组权限，包括读取、写入和执行权限。

操作系统通过 chmod 命令和 chown 命令来修改文件的权限和拥有者，以及 chgrp 命令来修改文件的所属组。

文件共享和锁定：

UNIX SVR4 允许多个进程同时访问同一个文件，提供了文件的共享机制。但为了防止多个进程同时对文件进行写操作而导致数据损坏，操作系统支持文件锁定（File Locking）机制，允许进程对文件进行加锁和解锁操作。

1. Каталоги файлов. Элементы каталога, операции ОС. Look

文件目录。 目录项、操作系统操作。 Look

"Каталоги файлов" 指的是操作系统中用于组织和管理文件的特殊类型文件。在大多数操作系统中，文件和目录都是以树状结构的形式组织的，其中顶层目录是根目录，包含其他子目录和文件。每个目录都可以包含其他目录和文件，从而形成了文件系统的层次结构。

"Элементы каталога" 是指目录中的文件和子目录。每个文件和子目录都是目录中的一个元素，它们包含有关文件或子目录的信息，例如名称、大小、权限等。

操作系统对文件目录进行多种操作，包括：

创建目录：创建新的目录以存储文件和其他目录。

删除目录：删除不再需要的目录及其包含的所有文件和子目录。

列出目录内容：显示目录中的所有文件和子目录列表。

移动和重命名目录：将目录移动到新位置或更改其名称。

设置和修改目录权限：设置谁可以访问目录以及执行何种操作。

"Look" 是一个用于搜索以指定字符串开头的单词的命令。它通常用于在文本文件中查找特定单词或短语的出现。使用 "look" 命令，用户可以快速查找和定位包含特定字符串的单词，以便进行进一步的处理或分析。

1. Размещение записей и файлов в блоках данных. Сложность и типы организации размещения.将记录和文件放置在数据块中。 住宿安排的复杂性和类型。

概念解释：

在计算机存储系统中，文件和记录通常被划分为固定大小的数据块，这些块被存储在磁盘或其他存储介质上。

每个数据块可以容纳一个或多个文件记录，这些记录可以是文件的一部分或整个文件。

系统根据文件的大小和访问模式，决定将文件记录存储在哪些数据块中，以便高效地访问和管理文件数据。

组织方式的类型和复杂度：

顺序存储：文件记录按顺序存储在数据块中，一个数据块存满后再存储到下一个数据块。这种组织方式的复杂度较低，适用于顺序访问文件的场景。

链式存储：文件记录通过链式结构连接在一起，每个记录包含指向下一个记录的指针。这种组织方式允许文件记录在任意位置插入和删除，但增加了记录访问的复杂度。

索引存储：每个数据块都有一个索引结构，记录了文件记录的位置和属性。这种组织方式适用于随机访问文件的场景，但需要额外的索引结构来维护文件记录的位置信息，增加了存储开销和管理复杂度。

优缺点：

顺序存储简单直观，适用于顺序访问文件的场景，但不适用于随机访问。

链式存储允许灵活地插入和删除文件记录，但增加了记录访问的复杂度和存储开销。

索引存储适用于随机访问文件的场景，可以快速定位文件记录，但需要额外的索引结构来维护位置信息，增加了存储开销。

1. Непрерывное размещение файлов (на примере ОСRT-11)文件的连续放置（以OSRT-11为例）

68. Цепочечное размещение файлов (на примере DOS FAT)文件的链式放置（以DOS FAT为例）

69. Индексированное размещение (на примере файловой системы UNIX UFS) .索引放置（以 UNIX UFS 文件系统为例）。