1. **操作系统进程／线程间通信机制有哪几种，它们的基本工作原理和适用场合是什么**

操作系统中进程/线程间通信（IPC，Inter-Process Communication）的机制多种多样，每种机制都有其适用场合和优缺点。以下是一些常见的IPC机制：

1. 管道（Pipe）

- 原理：管道是一种半双工的通信方式，数据只能单向流动，可以将输出流定向到另一个进程的输入流。

- 适用场合：适合有父子或生产者-消费者关系的进程间通信。

2. 命名管道（Named Pipe）

- 原理：与匿名管道类似，但它是一种文件形式存在，不同于匿名管道只能用于有亲缘关系的进程，命名管道允许无亲缘关系的进程间通信。

- 适用场合：适合本机的不同进程间的通信。

3. 信号（Signal）

- 原理：信号是一种比较简单的通信方式，用于通知接收进程某个事件已经发生。

- 适用场合：适合处理异步事件，如终止信号。

4. 消息队列（Message Queue）

- 原理：消息队列允许不同进程发送和接收消息，数据是以消息为单位进行传输的，避免了管道中可能出现的消息边界问题。

- 适用场合：适合不同进程间的复杂通信，如传递较大的数据或结构化数据。

5. 共享内存（Shared Memory）

- 原理：允许多个进程访问同一块内存空间，是最快的IPC方式，因为进程可以直接对内存进行操作，无需通过内核。

- 适用场合：适合传输大量数据的场合。

6. 信号量（Semaphore）

- 原理：主要用于进程间的同步，而非直接传递数据。信号量可以是一个计数器，用于控制多个进程对共享资源的访问。

- 适用场合：适合多个进程共享资源的场合。

7. 套接字（Socket）

- 原理：套接字可以实现不同机器间的进程通信，支持基于流和基于数据报的通信。

- 适用场合：适合网络间或一台机器上不同应用程序之间的通信。

8. 互斥锁（Mutex）、读写锁（RWLock）、条件变量（Condition Variable）

- 原理：这些都是同步原语，用于线程间的同步，以确保数据的一致性和防止竞态条件。

- 适用场合：适合多线程程序中，保护临界区资源不被并发访问。

9. 文件映射（Memory-mapped File）

- 原理：通过将文件内容映射到进程的地址空间，实现文件在内存中的直接访问，可以被多个进程共享。

- 适用场合：适合需要共享大型数据集的场合，类似于共享内存的使用场景。

每种IPC机制都有其适用的场景，选择合适的IPC机制通常取决于需要解决的问题、性能要求、程序的复杂性以及是否跨机器通信等因素。

**2、介绍 FAT文件系统与NTFS文件系统各自的特点，以及二者各自的优缺点是什么？**

FAT（File Allocation Table）文件系统和NTFS（New Technology File System）文件系统是Windows操作系统中常见的两种文件系统。它们各自的特点、优缺点如下：

\*\*FAT文件系统\*\*

特点：

1. 简单且广泛兼容：FAT文件系统是较早的文件系统之一，由于其简单性，它被广泛应用于各种设备上，包括USB闪存驱动器、SD卡等。

2. 支持的操作系统多：不仅Windows系统，许多其他操作系统（如macOS、Linux）也支持读写FAT文件系统。

3. 有多个版本：FAT12、FAT16和FAT32是最常用的版本，随着版本的更新，支持的磁盘大小和单个文件大小限制在增加。

优点：

1. 简单易用：FAT文件系统结构简单，易于实现和维护。

2. 兼容性好：几乎所有的操作系统都支持FAT文件系统。

3. 适合小容量磁盘：对于小容量存储设备（如闪存驱动器），FAT文件系统性能良好。

缺点：

1. 文件大小限制：FAT32文件系统单个文件最大仅支持4GB大小。

2. 磁盘大小限制：FAT32文件系统支持的磁盘大小有限，最大支持2TB。

3. 缺乏高级特性：不支持文件权限管理、加密、磁盘配额等高级特性。

4. 效率问题：随着存储容量的增加，FAT文件系统的效率会下降。

\*\*NTFS文件系统\*\*

特点：

1. 支持大容量存储：NTFS文件系统没有FAT32那样的磁盘和文件大小限制，可以支持更大的文件和磁盘。

2. 高级文件系统特性：支持文件加密、磁盘配额、文件压缩、稀疏文件、硬链接等。

3. 安全性更强：支持文件级别的权限控制和访问控制列表（ACL）。

优点：

1. 支持大文件：理论上单个文件大小可达16EB（Exabytes）。

2. 支持大容量磁盘：可以处理大于2TB的磁盘。

3. 高级特性：文件加密、文件权限控制、数据恢复、磁盘配额等。

4. 更好的数据恢复：日志文件系统可以在系统崩溃后恢复文件系统到一致状态。

缺点：

1. 兼容性问题：虽然大多数现代操作系统都可以读取NTFS文件系统，但写入支持可能有限，尤其是在非Windows系统中。

2. 效率：在小文件传输和小容量存储设备上，NTFS可能不如FAT文件系统高效。

3. 复杂性：NTFS的高级特性增加了其结构的复杂性，对文件系统的实现和维护提出了更高的要求。

总的来说，FAT文件系统由于其简单性和良好的兼容性，适合用于小容量存储设备和需要在多种操作系统之间移动文件的场景。而NTFS文件系统则适合现代大容量硬盘驱动器，特别是当需要高级特性和安全性时。

**3、 文件表中需要为这个打开的文件重新建立表项吗？**

A进程：

fd1=open(“/etc/passwd”,O\_RDONLY);

fd2=open(“local”,O\_RDWR);

fd3=open(“/etc/passwd”,O\_WRONLY);

B进程：

fd1=open(“/etc/passwd”,O\_RDONLY);

fd2=open(“private”,O\_RDONLY);

文件表(file table)：（是一个全局表，为系统中所有的进程共享。

对每个活动的open, 它都包含一个条目。

每个系统文件表的条目都包含文件偏移量、访问模式（读、写、or 读-写）以及指向它的文件描述符表的条目计数。

以Linux为例，已打开的文件在内核中用file结构体表示，文件描述符表中的指针指向file结构体）

如果B进程是A进程通过fork系统调用产生的子进程；而且，父进程A在fork子进程B之前已经打开/etc/passwd文件，**试问：文件表中需要为这个打开的文件重新建立表项吗？**

在Linux操作系统中，当一个进程（如A进程）通过fork系统调用创建一个子进程（如B进程）时，子进程会继承父进程的文件描述符表。这意味着在fork调用之后，父进程和子进程将共享相同的文件表项（即file结构体）。

因此，对于由A进程打开的`/etc/passwd`文件，不会因为B进程的创建而在文件表中重新建立一个新的表项。子进程B将拥有指向相同file结构体的文件描述符，即：

- A进程的fd1和B进程的fd1都指向`/etc/passwd`文件的同一个file结构体。

- 同样，如果其他文件在fork调用之前被父进程打开，子进程也会继承这些文件描述符和它们对应的file结构体指针。

综上所述，文件表中不需要为这个已打开的文件重新建立表项。相反，父进程和子进程会共享相同的file结构体，包括文件偏移量、访问模式以及指向它的文件描述符表的条目计数。

需要注意的是，虽然父子进程共享file结构体，它们各自有自己独立的文件描述符表。因此，父子进程可以独立地操作文件描述符，如关闭文件描述符或改变文件偏移量，但这些操作会影响到彼此，因为它们是通过相同的file结构体进行的。如果需要父子进程各自独立操作文件，通常需要在fork后立即在子进程中使用dup2或者close等系统调用来复制或关闭文件描述符。

**4、桌面虚拟化与服务器虚拟化之间关系是什么？**

PC/服务器虚拟化：系统虚拟化的最大价值所在。

桌面虚拟化：解决个人电脑的桌面环境（包括应用程序和文件等）与物理机之间的耦合关系。经过虚拟化的桌面环境被保存在远程的服务器上，当用户使用具有足够显示能力的兼容设备（比如PC, 智能手机等）在桌面环境上工作时，所有的程序与数据都运行和最终保存在这个远程的服务器上.

请问桌面虚拟化与服务器虚拟化之间关系是什么？

桌面虚拟化和服务器虚拟化都是虚拟化技术的应用，但它们关注的领域和目标不同。这两种虚拟化形式之间的关系主要体现在它们共同使用底层虚拟化技术的基础上，以及它们在IT架构中的相互作用。

\*\*服务器虚拟化\*\*

服务器虚拟化通常涉及到在物理服务器上创建多个隔离的虚拟机（VM），每个VM都可以运行自己的操作系统和应用程序，彼此独立。服务器虚拟化的关键点在于资源的最优化和灵活性，使得单个物理服务器的资源可以被多个虚拟机共享，从而提高资源利用率，简化管理，增强灾难恢复能力等。

\*\*桌面虚拟化\*\*

桌面虚拟化则是指将用户的桌面环境从物理机中抽象出来，并在服务器中进行集中管理和运行。用户可以通过网络，使用不同的终端设备访问自己的虚拟桌面。桌面虚拟化的核心价值在于提供灵活的工作环境、增强数据安全性和便于管理。

\*\*两者之间的关系\*\*

1. 技术基础共享：桌面虚拟化和服务器虚拟化都建立在虚拟化技术之上，使用了相似的底层架构（如使用虚拟机监控器或者超级管理程序）来创建和管理虚拟机。

2. 交互作用：在实际部署中，桌面虚拟化往往依赖于服务器虚拟化提供的基础设施。虚拟化的桌面通常托管在虚拟化的服务器上，服务器虚拟化提供了必要的计算、存储和网络资源，以支持大量桌面虚拟机的运行。

3. 管理和维护：从管理的角度来看，桌面虚拟化可以利用服务器虚拟化的管理工具，实现集中管理和维护，减轻IT部门的管理负担。

4. 安全和合规：通过在服务器端集中托管用户桌面，桌面虚拟化可以利用服务器虚拟化的安全和合规性能力，确保数据安全和遵守相关政策。

总之，桌面虚拟化和服务器虚拟化虽然关注的场景不同，但它们都使用虚拟化技术来提高资源的灵活性和效率，同时也相互依赖，以实现整个IT系统的优化。

**5、内存虚拟化的直接模式和影子页表模式各自的工作原理和区别？**

内存虚拟化方法：

VMM维护一个虚拟机内存管理数据结构——影子页表(shadow page table)。VMM通过影子页表给不同的虚拟机分配机器的内存页，如操作系统虚拟内存一样，VMM能将虚拟机内存换页到磁盘，因此，虚拟机申请的内存可以超过机器的物理内存。VMM也可以根据每个虚拟机的要求,动态地分配相应的内存。

请问：内存虚拟化的直接模式和影子页表模式各自的工作原理和区别？

内存虚拟化是虚拟机管理器（VMM）或超级管理程序（Hypervisor）用来管理物理内存资源并为虚拟机（VM）提供虚拟内存空间的一种技术。在内存虚拟化中，有两种主要的技术：直接模式（也称为硬件辅助分页）和影子页表模式。

\*\*直接模式（硬件辅助分页）\*\*

工作原理：

直接模式利用现代CPU提供的硬件辅助虚拟化功能，例如Intel的Extended Page Tables（EPT）或AMD的Rapid Virtualization Indexing（RVI）。这些技术通过硬件支持，在物理内存和虚拟机的虚拟内存之间提供直接的映射。

1. 每个虚拟机拥有自己的一套页表，这些页表由虚拟机的操作系统管理，直接将虚拟地址（VA）映射到虚拟机的物理地址（guest physical address, GPA）。

2. VMM不需要介入每次的内存访问转换，因为CPU硬件会自动使用另一层页表（例如EPT或RVI）将GPA映射到宿主机的物理地址（host physical address, HPA）。

区别：

- 直接模式减少了VMM在内存访问中的介入，降低了内存管理的开销，提高了性能。

- 直接模式需要CPU具备硬件辅助虚拟化功能，这在较老的CPU中可能不可用。

\*\*影子页表模式\*\*

工作原理：

影子页表是VMM用来管理内存访问的数据结构。在不支持硬件辅助虚拟化的环境中，VMM会拦截和处理所有的内存访问，然后通过影子页表来维护VA到HPA的映射。

1. 虚拟机的操作系统会像在非虚拟化环境中一样维护自己的页表，将VA映射到GPA。

2. VMM为每个虚拟机维护一套影子页表，将VA直接映射到HPA。

3. 当虚拟机的操作系统尝试更新它的页表时，VMM会捕获这些操作并同步更新影子页表，以确保VA到HPA的映射是正确的。

区别：

- 影子页表需要VMM在内存访问路径中更频繁地介入，这可能导致性能开销较大。

- 影子页表不依赖特定的硬件辅助，因此可以在没有这些功能的较老CPU上使用。

总结来说，硬件辅助分页（直接模式）通过硬件提供更高效的内存虚拟化，而影子页表模式则依靠软件在VMM中管理内存映射。随着硬件辅助虚拟化技术的普及，直接模式成为了现代虚拟化环境中的首选方法，它可以提供更接近物理机性能的内存访问速度。影子页表模式在没有硬件支持的环境中仍然有其应用场景。

**6、请阅读GB17859并完成以下任务：**

**-（1）可信计算基、引用监控器概念是如何定义的？**

**–（2）五个可信等级是什么，划分的原则是什么？**

**–（3）涉及的安全功能主要要求包括哪些？**

**–（4）区分第二级与第三级安全功能要求的异同之处？**

**–（5）区分第三级与第四级安全功能要求的异同之处？**

–（4）区分第二级与第三级安全功能要求的异同之处？

–（5）区分第三级与第四级安全功能要求的异同之处？

由于我没有直接访问GB17859标准文档的能力，我无法提供具体的、基于该标准的第二级与第三级、第三级与第四级之间的安全功能要求的异同点。然而，我可以提供一个一般性的解释，这些通常是基于通用的安全等级划分原则所共有的。

在许多安全等级框架中，例如美国国家安全局（NSA）的评估标准（如Common Criteria）或早期的美国国防部安全模型（如Orange Book），安全等级通常按照从低到高的顺序排列，每个等级都加强了特定的安全措施。以下是它们可能如何区分的一般描述：

\*\*(4) 第二级与第三级安全功能要求的异同\*\*

第二级可能提供基础的访问控制和审计功能，以及一定程度的认证和完整性保护。在这一级别上，安全措施可能仍然是相对简单的，以用户为中心，并且多侧重于防止意外的安全漏洞。

第三级通常会引入更为严格和精细的安全控制措施，可能包括对访问控制和审计功能的加强。这个级别可能会要求更复杂的认证机制，如双因素认证，以及对数据和操作的更细粒度的控制。此外，第三级可能会要求系统能够保护机密数据，实施严格的数据分类和安全标记，并提供更全面的审计记录。

\*\*(5) 第三级与第四级安全功能要求的异同\*\*

第三级与第四级之间的主要区别通常在于安全措施的复杂性和强制性。第四级可能会要求实施更高级的安全策略，包括更高级的访问控制机制（如基于角色的访问控制或强制访问控制），以及强化的系统完整性和可靠性措施。

在第四级，可能还需要实现更完善的安全保障措施，如对抗恶意软件和内部威胁的防护，高级加密应用，以及高度复杂的安全管理过程。此外，第四级可能会要求系统的安全设计和实施需要经过更为严格的验证和测试，以确保在高威胁环境中的可靠性。

请注意，这些描述是基于通用的理解和不同安全等级框架的相似之处。具体到GB17859标准的详细要求和定义，您需要查阅官方发布的标准文档或咨询具有相关专业知识的专家。

第一级（C1级）：支持基本安全需求

C1级适用于对计算机系统安全保护要求相对较低的场景，强调对非授权访问

的防范和控制，要求基本的访问控制、身份鉴别等功能。

第二级（C2级）：支持自主访问控制

C2级适用于需要比较高安全防范的系统，要求更严格的访问控制、用户账号

管理、文件系统安全等功能，以及审计和可靠的安全审计记录等。

第三级（B1级）：支持离散保密需求

B1级适用于对安全防范需求较高的系统，要求在访问控制、数据加密、身份

鉴别等方面具备更严格的安全要求，以满足对离散保密需求的系统要求。

第四级（B2级）：支持严格的安全防范需求

B2级适用于对安全防范需求更为严格的系统，要求在访问控制、身份鉴别、

数据完整性、安全审计等方面具备更高的安全功能，以保护系统免受更严重的威

胁。

第五级（B3级）：支持高度安全防范需求

B3级适用于对严格安全防范需求的系统，要求在访问控制、数据完整性、安

全通信、安全审计等方面具备更高的安全保障能力，并要求有相应的安全设计和

评估文档

**7、**

**1.可信计算芯片提供了那些基本功能？**

**2.可信计算芯片如何度量平台完整性？**

**3.可信计算芯片如何报告平台的状态？**

**4.基于可信计算芯片的远程证实协议？**

**5.可信计算芯片提供哪三个信任根？**

**6.数据封装（seal）操作原理如何？**

可信计算芯片，通常指的是可信平台模块（TPM, Trusted Platform Module），它是一种专门设计用来增强计算设备安全性的硬件组件。以下是TPM提供的一些基本功能，以及它如何执行这些功能：

1. \*\*可信计算芯片提供的基本功能：\*\*

- 安全存储：提供一个安全的环境来保存加密密钥和其他敏感数据。

- 随机数生成：生成高质量的随机数，用于加密操作。

- 加密引擎：支持密钥生成和加密/解密操作。

- 完整性度量和存储：记录系统启动时的软件和硬件组件的度量值。

- 远程证实：能够向远程实体证明系统的完整性和状态。

2. \*\*如何度量平台完整性：\*\*

TPM通过启动时的度量来度量平台完整性。在系统启动过程中，TPM会记录（度量）BIOS、引导加载程序、操作系统，以及其他启动过程中涉及的软件组件的哈希值。这些哈希值（度量值）会被存储在TPM的平台配置寄存器（PCR）中。

3. \*\*如何报告平台的状态：\*\*

TPM可以通过远程证实（Remote Attestation）过程报告平台状态。在这个过程中，TPM会使用其存储的密钥对PCR值进行签名，生成一个报告，证明当前的系统状态。这个签名的报告可以提供给请求方，以便验证系统的完整性。

4. \*\*基于可信计算芯片的远程证实协议：\*\*

远程证实协议通常涉及以下步骤：

- 请求方发起证实请求。

- TPM使用私有密钥对PCR中的完整性度量值进行签名。

- TPM将签名的完整性报告返回给请求方。

- 请求方使用TPM的公钥验证签名和度量值的有效性。

5. \*\*提供的三个信任根：\*\*

- 硬件信任根：TPM芯片本身，作为一个物理设备，提供了一个安全的起点。

- 软件信任根：通过在启动过程中度量和验证软件来建立信任。

- 报告信任根：TPM能够生成并签名报告，以证明平台的完整性和状态。

6. \*\*数据封装（seal）操作原理：\*\*

封装操作是指将数据与特定的平台状态关联起来，只有在这些条件满足时，数据才能被访问（解封装）。TPM的封装功能会将数据加密，并将解密操作与PCR中的特定度量值关联起来。如果TPM检测到PCR的值与封装数据时的值不匹配，数据将保持加密和不可访问状态。这确保了只有在预期的安全状态下，敏感数据才能被访问。

请注意，以上信息是基于通用的TPM功能和操作原理。不同的TPM版本和制造商可能会有额外的特性和功能。

**8、**

**【Q1】：你了解Linux内核“主体”的数据结构吗?其中uid、euid、**

**suid、fsuid各自的含义和用途是什么？与标识和鉴别机制的关系？**

**• 【Q2】：你了解Linux内核“客体”(如文件、目录、特殊文件、设**

**备、IPC客体-消息队列+共享内存+信号量、信号等)的数据结构吗?**

**如果要为这些客体打安全标签（如安全级），应该如何添加？**

**• 【Q3】：Linux的“块设备”与“字符设备”区别是什么，相关的数**

**据结构如何？举例说明。**

**• 【Q4】: 你了解Linux2.6系统中与DAC相关的命令吗？如熟悉chmod,**

**chown, setfacl, chfacl等命令的使用吗?**

**• 【Q5】：你能在Linux2.6以上内核中找出客体（如文件/目录）相关**

**的DAC数据结构吗（传统9Bit模式、ACL模式）？**

**• 【Q6】: 你了解Linux的SETUID机制吗？chmod命令中的特殊权限位**

**的S\_ISUID, S\_ISGID,S\_ISVTX(STICKY位）是什么含义？**

【Q1】Linux内核中的"主体"通常指的是进程，它在操作系统中代表执行中的程序。Linux内核为了实现权限控制，为每个进程维护了一组身份标识：

- uid：User ID，用户ID，是进程拥有者的实际ID。

- euid：Effective User ID，有效用户ID，用于决定进程对资源的访问权限。

- suid：Saved User ID，保存的用户ID，当执行具有set-user-ID权限的程序时，suid将保存被执行程序的所有者ID。

- fsuid：Filesystem User ID，文件系统用户ID，专门为文件系统操作使用，用于覆盖euid。

其中，标识（Identification）是指如uid这样的身份标识，而鉴别（Authentication）通常是指系统如何确认一个用户的身份，例如通过密码等方式。在权限检查时，Linux内核会使用这些ID来确定进程是否有权进行特定的操作。

【Q2】Linux内核中的"客体"包括文件、目录、特殊文件、设备、IPC对象（消息队列、共享内存、信号量）以及信号等。它们在内核中通常通过相应的数据结构进行管理，如`inode`结构体用于文件和目录，`ipc\_namespace`用于IPC对象。

如果要为这些客体打安全标签（例如安全级），可以通过扩展现有的数据结构或使用安全模块框架（如SELinux）来添加安全相关的属性。在SELinux中，这通常是通过将安全上下文关联到相应的内核数据结构来实现的。

【Q3】Linux中的“块设备”和“字符设备”的区别主要在于数据的处理方式：

- 块设备：允许存储和随机访问固定大小的数据块，例如硬盘、SSD等。相关的数据结构是`block\_device`，它包含了与块设备相关的操作和属性。

- 字符设备：提供顺序访问的接口，通常用于打印机、串口等。相关的数据结构是`cdev`，包括了字符设备的操作函数指针和其他信息。

例如，硬盘通常作为块设备，而键盘或鼠标可能被实现为字符设备。

【Q4】在Linux 2.6系统中，与DAC（Discretionary Access Control，自主访问控制）相关的命令包括：

- chmod：更改文件或目录的权限。

- chown：更改文件或目录的所有者和所属组。

- setfacl：设置文件或目录的访问控制列表（ACL）。

- getfacl：获取文件或目录的访问控制列表。

这些命令用于管理文件和目录的权限，确保只有授权的用户可以访问特定资源。

【Q5】在Linux 2.6及以上内核中，客体（如文件/目录）相关的DAC数据结构包括：

- 传统9Bit模式：每个文件或目录的`inode`结构体中都有一个`mode`字段，它包含了设置文件权限的9个位，分别表示文件所有者、所属组成员和其他用户的读、写、执行权限。

- ACL模式：如果使用了ACL，相关的数据结构会包含更为详细的权限信息。在文件系统中，这通常是通过额外的数据结构来实现的，如`posix\_acl\_entry`。

【Q6】Linux的SETUID机制允许用户以文件所有者的权限来执行程序。chmod命令中的特殊权限位含义如下：

- S\_ISUID（Set User ID）：当一个可执行文件具有这个权限位时，运行该文件的进程将具有文件所有者的euid。

- S\_ISGID（Set Group ID）：与S\_ISUID类似，但是应用于进程的egid。

- S\_ISVTX（Sticky Bit，粘滞位）：通常用于目录，确保只有文件的所有者、目录的所有者或root可以删除或重命名目录下的文件。

这些特殊权限位用于控制文件的特殊访问权限。

**9、 根据中国墙模型思想，假定公司A、公司B同属于一个利益**

**冲突类P；公司X、公司Y同属于另一个利益冲突类Q，试**

**问：**

**（1）假定用户Alice读取过公司A的数据和公司X的数据，**

**用户Bob读取过公司A的数据和公司Y的数据，那么接下来，**

**用户Alice能否写公司A的数据？为什么？请说明理由。**

**（2） 假定用户Alice读取过公司A的数据，以及公司X已经**

**清洗过(sanitized)的数据，用户Bob读取过公司A的数据和公**

**司Y的数据，那么接下来，用户Alice能否读取公司Y的数据？**

**为什么？请说明理由。**

中国墙模型（Chinese Wall Model）是一种动态的访问控制策略，主要用于防止利益冲突和信息流动所可能导致的泄露问题。该模型假定存在多个利益冲突类，每个类包含了存在潜在利益冲突的实体（比如公司）。

1. 在中国墙模型中，一旦用户访问了某个利益冲突类中的公司数据，他就不能访问该类中其他公司的数据，以防止可能的利益冲突。但是，用户可以访问不同利益冲突类中的公司数据。

（1）根据上述规则，用户Alice已经读取过公司A的数据（属于利益冲突类P），根据中国墙模型，她在不违反访问控制策略的情况下，可以对公司A的数据进行写操作。这是因为Alice访问公司A的数据并不会产生新的利益冲突，她没有试图访问同一利益冲突类P中其他公司（比如公司B）的数据。

（2）用户Alice读取过公司A的数据，这意味着她已经被限制在了利益冲突类P中，但她也读取了公司X的已经清洗过的数据，这里假设“清洗过”的数据不会引入新的利益冲突，且不会违反中国墙模型的策略。如果公司Y属于另一个利益冲突类Q，并且Alice没有读取过类Q中其他公司（比如公司X）的敏感或未清洗的数据，那么理论上，根据中国墙模型，Alice应该能够读取公司Y的数据。这是因为Alice没有访问过利益冲突类Q中的敏感数据，因此不存在直接的利益冲突。

需要注意的是，实际应用中是否允许用户访问某些数据，可能还受到其他安全策略、法规要求或组织内部规定的影响。而“清洗过的数据”具体指的是什么，以及它在该组织中如何被定义和处理，也可能对访问控制决策产生影响。