《计算机系统设计》平时作业 6

计算机科学与技术 2110939 李颖

1. 当系统发生缺页中断时，①CPU 和操作系统分别做了哪些具体操作？②如果此时有外部键盘按键事件，系统如何响应，具体的处理过程是什么？

(1)当操作系统发生缺页中断时，CPU 会检测到缺页中断，即程序需要访问某个虚拟内存地址的数据时，如果对应的页面不在物理内存中，就会触发缺页中断。之后，CPU 便会保存当前的现场（程序计数器、标志寄存器等），分析中断原因，再转去执行操作系统提供的针对缺页中断事件的处理程序。

操作系统内核接管控制后，开始处理缺页中断。OS 先根据缺页中断的信息确定缺失页面的位置。若缺失页面在磁盘上，OS 会使用页面置换算法选择合适的页面进行置换，并将磁盘上的页面加载到物理内存中。OS 还会更新进程的页表，将缺失页面的地址映射到页面在物理内存中的正确位置。

CPU 恢复保存的上下文，恢复被中断程序的执行，让程序继续运行。

(2) 中断嵌套处理是指在处理一个中断时，系统又发生了一个更高优先级的中断，需要暂时中断正在进行的处理，转而处理更高优先级的中断。这种情况下，系统需要正确保存和恢复各个中断的现场信息，并按照中断的优先级顺序进行响应。

而出现缺页中断时，由 CPU 的内存管理单元 MMU 发出缺页中断；外部键盘按键事件则由键盘控制器发出一个中断请求。而系统则会根据中断的优先级来决定先处理哪个中断。具体的优先级设定因系统而异，但一般来说，缺页中断的优先级可能会高于普通的设备中断（如键盘中断），即键盘中断是可屏蔽的，OS 可根据中断屏蔽标志来决定是否响应。

若响应，则发生中断嵌套。系统保存当前缺页中断的上下文，然后跳转到键盘中断的处理程序。键盘中断处理程序会处理键盘事件，录入读取按键的扫描码，然后执行相应的处理逻辑。键盘中断处理完成后，CPU 会恢复之前保存的缺页中断的上下文，并继续执行缺页中断的处理程序。

1. 某应用程序涉及大量读写操作，如果将其从 Linux 系统移植到目标平台，例如可信执行环境（Trust Execution Environment, TEE)，需要进行哪些改进、修正？如果参考 POSIX Specification，可做哪些调整或设计？

可以进行的改进和修正为：

(1) 优化 I/O 性能：在 TEE 中，由于安全性的限制，I/O 操作可能会有额外的开销。因此，需要优化 I/O 性能。可以使用缓冲 I/O 来减少系统调用的次数，或考虑使用异步 I/O 来重叠 I/O 操作和计算操作，进一步提高程序的吞吐量。

(2) 调整缓存策略：TEE 通常具有较小的内存和资源限制，因此需要优化应用程序的内存使用和资源管理。对于大量读写操作，需要设计高效的内存管理策略，以减少内存占用和资源消耗。针对大量的读写操作，可能需要实现更智能的缓存策略来减少磁盘访问的次数。可以使用内存映射文件来将文件的一部分或全部映射到进程的地址空间中，从而允许程序像操作内存一样操作文件。

(3) 增强安全性：由于 TEE 提供更高级别的安全性和隔离性，需要确保读写操作的安全性。可能需要加密或验证读写操作，以保护敏感数据免受恶意访问或篡改。考虑在读写操作中实现安全的访问控制，以确保只有经授权的实体可以进行读写操作。

(4) 调整文件系统：TEE 可能具有限制的文件系统访问权限，因此需要调整应用程序的文件读写操作以适应 TEE 的限制。可能需要使用 TEE 提供的安全存储接口或安全文件系统来进行读写操作。

如果参考 POSIX Specification，可以做的调整为：

(1) 安全隔离性设计：充分利用 TEE 提供的安全隔离特性，确保应用程序的关键数据和代码在 TEE 中执行，并与外界隔离。评估应用程序的敏感数据，并确定哪些数据需要在 TEE 中处理。可能需要修改应用程序的数据流，以确保敏感数据不会泄露到 TEE 外部。

(2) I/O 性能优化：使用 POSIX 标准的 I/O 函数（如 read(), write(), open(), close() 等），并考虑使用非阻塞 I/O、异步 I/O 或批量 I/O 来提高性能。如果 TEE 提供了特定的 I/O 优化机制（如加密文件系统、高速缓存等），考虑使用这些机制来

进一步提高 I/O 性能。

(3) 并发和同步机制：评估应用程序的并发读写需求，并确定如何在 TEE 中实现有效的并发控制和同步机制。使用 POSIX 提供的线程和同步原语（如互斥锁、条件变量、信号量等）来管理并发操作，确保数据的一致性和完整性。如果 TEE 提供了更高级的并发控制机制（如事务内存、分布式锁等），可以考虑使用这些机制来简化并发编程并提高性能。

(4) 错误处理和恢复：在移植过程中，重新评估并调整应用程序的错误处理和恢复机制。使用 POSIX 标准的错误码和错误消息来报告和处理错误。考虑实现重试机制，以处理可能因网络问题、硬件故障等导致的临时性错误。在 TEE 中，可能需要额外的错误处理逻辑来处理与 TEE 特定的错误条件（如安全违规、资源限制等）。

(5) API 兼容性：评估应用程序使用的 API 与 TEE 的兼容性。虽然 TEE 可能提供了与 POSIX 兼容的 API 子集，但仍然存在一些差异。如果 TEE 提供了特定的 API 或功能来增强安全性或性能，考虑使用这些 API 来替代 POSIX 标准 API。修改应用程序的代码以使用 TEE 提供的 API，并确保代码的可移植性和可维护性。