КЕРУВАННЯ ПРИСТРОЯМИ ВВЕДЕННЯ/ВИВЕДЕННЯ

Виконали студенти групи ТР-51

Заковоротний О. І.

Донцов М. Д.

Талах О. М.

Рівні переривань

- Насамперед необхідно мати можливість скасовувати або відкладати обробку переривань під час виконання важливих дій.
- Виходячи з цього, переривання поділяють на рівні відповідно до їхнього пріоритету (Interrupt Request Level, IRQL — рівень запиту переривання). Окремі фрагменти коду ОС можуть маскувати переривання, нижчі від певного рівня, скасовуючи їхнє отримання. Виділяють, крім того, немасковані переривання, отримання яких не можна скасувати (апаратний збій пам'яті тощо).

Особливості реалізації оброблювачів

Оброблювачі переривань — це звичайні послідовності інструкцій процесора; їх можна розробляти як на асемблері, так і мовами програмування високого рівня.

В оброблювачах дозволено виконувати більшість операцій за деякими винятками:

- не можна обмінюватися даними із адресним простором режиму користувача,
- оскільки він не виконується у контексті процесу;
- не можна виконувати жодних дій, здатних спричинити очікування

Символьні, блокові та мережні драйвери пристроїв

- Для блокових пристроїв дані зберігають блоками однакового розміру, при цьому кожен блок має свою адресу, і за допомогою відповідного драйвера до нього можна отримати прямий доступ. Основним блоковим пристроєм є диск.
- Символьні пристрої розглядають дані як потік байтів, при цьому окремий байт адресований бути не може. Прикладами таких пристроїв є модем, клавіатура, миша, принтер тощо.
- Базовими системними викликами для символьних пристроїв є виклики читання і записування одного байта.
- Окремою категорією є мережні пристрої, які надаються прикладним програмаму вигляді мережних інтерфейсів зі своїм набором допустимих операцій, які відображають специфіку мережного введення-виведення (наприклад, ненадійність зв'язку).

Контролер прямого доступу до пам'яті (Direct Memory Access, DMA)

- Такий контролер сам керує пересиланням блоків даних від пристрою безпосередньо у пам'ять, не залучаючи до цього процесора. Блоки даних, які пересилають, завжди набагато більші, ніж розрядність процесора, наприклад вони можуть бути завдовжки 4 Кбайт.

Контролер прямого доступу до пам'яті (Direct Memory Access, DMA)

Схема введення-виведення при цьому наприклад, буде такою:

- процесор дає команду DMA-контролеру виконати читання блоку від пристрою, разом із командою він відсилає контролеру адресу буфера для введення-виведення (такий буфер має бути у фізичній пам'яті);
- DMA-контролер починає пересилання, процесор у цей час може виконувати інші інструкції;
- після завершення пересилання всього блоку DMA-контролер генерує переривання;
- оброблювач переривання (нижня половина) завершує обробку операції читання, наприклад переміщуючи дані із фізичного буфера у сторінкову пам'ять.

Процесор тут бере участь тільки на початку операції та в кінці — за все інше відповідає контролер прямого доступу до пам'яті.

ПІДСИСТЕМА ВВЕДЕННЯ-ВИВЕДЕННЯ ЯДРА

Планування операцій введення-виведення

- Планування введення-виведення звичайно реалізоване як середньотермінове планування. Як відомо, з кожним пристроєм пов'язують чергу очікування, під час виконання блокувального виклику (такого як read()) потік поміщають у чергу для відповідного пристрою, з якої його звичайно вивільняє оброблювач переривання. Різним пристроям можуть присвоювати різні пріоритети.

Буферизація і кешування

- Буферизацію варто відрізняти від кешування. Основна відмінність між ними полягає в тому, що буфер може містити єдину наявну копію даних, тоді як кеш за визначенням зберігає у більш швидкій пам'яті копію даних з іншого місця.
- З іншого боку, ділянку пам'яті в деяких випадках можна використати і як буфер, і як кеш. Наприклад, якщо після виконання операції введення із використанням буфера надійде запит на таку саму операцію, дані можуть бути отримані із буфера, який при цьому буде частиною кеша.

Спулінг

- Спулінг (spooling) — технологія виведення даних із використанням буфера, що працює за принципом FIFO. Такий буфер називають спулом (spool) або ділянкою спула (spool area).
- Спулінг використовують тоді, коли виведення даних має виконуватися неподільними порціями (роботами, jobs). Неподільність робіт полягає в тому, що їхній вміст під час виведення не перемішується (тільки після виведення всіх даних однієї роботи має починатися виведення наступної).
- Роботи надходять у спул і в ньому вишиковуються у FIFO-чергу (нові роботи додаються у її хвіст). Як тільки пристрій вивільняється, роботу із голови черги передають пристрою для виведення.

Обробка помилок

- У підсистемі введення-виведення під час роботи виникають різні помилки, які можна віднести до кількох категорій.
- Помилки в програмному коді введення-виведення (доступ до відсутнього пристрою, недопустимі дії із пристроєм тощо). Реакцією на такі помилки звичайно є повернення коду помилки в застосування. Введення-виведення при цьому зазвичай не виконують.

Обробка помилок

Помилки, викликані апаратними проблемами. Серед них розрізняють:

- викликані тимчасовими причинами (високе навантаження на мережу, сигнал «зайнято» для модему); для цих помилок звичайною реакцією є повторна спроба виконання введення-виведення;
- що вимагають втручання користувача (відсутність дискети в дисководі,відсутність паперу у принтері); за такої помилки зазвичай потрібно попросити користувача виконати певні дії;
- викликані некоректною роботою апаратного забезпечення (збій контролера, дефектні сектори на диску); у цьому разі важливим є надання користувачу якомога більше повної інформації про помилку

Введення-виведення у режимі користувача

- Тут розглянемо взаємодію підсистеми введення-виведення із процесами режиму користувача.

Синхронне введення-виведення

- У більшості випадків введення-виведення на рівні апаратного забезпечення кероване перериваннями, а отже є асинхронним. Однак це використовувати таку обробку складніше ніж синхронну, тому найчастіше введення виведення в ОС реалізоване у вигляді набору блокувальних або синхронних системних викликів, подібних до read(), write().
- Під час виконання такого виклику поточний потік призупиняють, переміщуючи в чергу очікування для цього пристрою. Після завершення операції введення-виведення і отримання всіх даних від пристрою потік переходить у стан готовності та може продовжити своє виконання.

Синхронне введення-виведення

Однак синхронне введення-виведення підходить не для всіх застосувань, а зокрема для таких категорій програм:

- Серверів, що обслуговують багатьох клієнтів;
- Застосувань, що працюють із журналом;
- Мультимедійних застосувань(відіславши запит на читання одного кадру, потрібно одночасно показувати інші).

Багатопотокова організація введення-виведення

- Цей підхід полягає у тому, що за необхідності виконання асинхронного введення-виведення у застосуванні створюють новий потік, у якому виконуватиметься звичне, синхронне введення виведення. При блокуванні цього потоку вихідний потік продовжуватиме своє виконання.
- Переваги: простота реалізації та низькі вимоги до ресурсів.
- Недоліки: у недостатній маштабованості(за великої кількості одночасних запитів витрати на створення потоків для кожного із них можуть спричиняти зменшення продуктивності.

Введення-виведення з повідомленням

- Першою технологією, яку можна використати для організації введення-виведення без блокування і яка не вимагає організації багатопотоковості є введення-виведення із повідомленням.
- Якщо потрібно в циклі виконати блокувальний виклик (наприклад, read ()) для кількох файлових дескрипторів, може трапитися так, що один із викликів заблокує поточний потік у той момент, коли на дескрипторі, який використовується в іншому виклику, з'являться дані. Доцільно організувати одночасне очікування отримування даних із кількох дескрипторів. Це і є основним мотивом розробки даної категорії засобів введення-виведення.

Введення-виведення з повідомленням

У цьому разі виконання введення-виведення поділяють на кілька етапів.

- 1. Спеціальний системний виклик визначає, чи можна виконати синхронне введення-виведення хоча б для одного дескриптора із заданого набору без блокування потоку.
- 2. Як тільки хоча б один дескриптор із набору стає готовий до введення-виведення без блокування, виклик повідомлення повертає керування; при цьому поточний потік може визначити, для яких саме дескрипторів може бути виконане введення-виведення або які з них змінили свій стан (тобто отримати повідомлення про стан дескрипторів).
- 3. Потік, що викликає, може тепер у циклі обійти всі дескриптори, визначені внаслідок повідомлення на етапі 2, і виконати введення-виведення для кожного з них, блокування поточного потоку ця операція в загальному випадку не спричинить.

Введення-виведення з повідомленням про стан дескрипторів

- Основною особливістю введення-виведення із повідомленням про стан дескрипторів є те, що в разі його використання не зберігається стан. Кожен виклик повідомлення вимагає передавання всього набору дескрипторів і повертає «миттєвий знімок» стану цих дескрипторів. Це потребує повного обходу цього списку як всередині виклику повідомлення, так і в коді, що його викликав.
- Такий обхід може серйозно позначитися на продуктивності у разі великої кількості дескрипторів. Таким чином, інформація про неактивні дескриптори даремно копіюватиметься у ядро і назад під час кожного виклику повідомлення.

Введення-виведення з повідомленням про події

- Основною відмінністю цього підходу є збереження у ядрі інформації про набір дескрипторів, зміна стану яких становить інтерес. Унаслідок цього з'являється можливість повертати інформацію про стан не всіх дескрипторів, а тільки про ті з них, які перейшли у стан готовності з моменту останнього виклику функції повідомлення (тобто, про всі події зміни стану).

Асинхронне введення-виведення

Основна ідея тут полягає в тому, що потік, який почав виконувати введення-виведення, не блокують до його завершення.

Асинхронне введення-виведення зводиться до виконання таких дій.

- Потік виконує системний виклик асинхронного введення або виведення, який ставить операцію введення-виведення в чергу і негайно повертає керування.
- Потік продовжує виконання паралельно з операцією введення-виведення.
- Коли операція введення-виведення завершується, потік отримує про це повідомлення.

Порти завершення введення-виведення

- Порт завершення введення-виведення (I/O completion port) — поєднує багатопотоковість із асинхронним введенням-виведенням для вирішення проблем розробки серверів, що обслуговують велику кількість одночасних запитів.
- в момент запуску серверного застосування заздалегідь створюють набір потоків (пул), кожен із яких готовий обслуговувати запити. Коли приходить запит, перевіряють, чи є в пулі вільні потоки, якщо є, з нього вибирають потік, який починає обслуговувати запит. Після виконання запиту потік повертають у пул. Коли з появою нового запиту вільних потоків у пулі немає, запит поміщають у чергу, і він там очікує, поки не вивільниться потік, що може його обслужити.

ДЯКУЄМО ЗА УВАГУ!