Операційні системи

Лекція 10

Керування введенням-виведенням

План лекції

- Завдання керування введенням-виведенням
- Фізична організація пристроїв введення-виведення
- Організація програмного забезпечення введення-виведення
- Синхронне та асинхронне введення-виведення

Завдання керування введенням-виведенням

- Забезпечення доступу до зовнішніх пристроїв з прикладних програм
  - Передавати пристроям команди
  - Перехоплювати переривання
  - Обробляти помилки
- Забезпечення спільного використання зовнішніх пристроїв
  - Розв’язувати можливі конфлікти (синхронізація)
  - Забезпечувати захист пристроїв від несанкціонованого доступу
- Забезпечення інтерфейсу між пристроями і рештою системи
  - Інтерфейс повинен бути універсальним для різних пристроїв

Фізична організація пристроїв введення-виведення

- Два типи пристроїв – *блок-орієнтовані* (або блокові) та *байт-орієнтовані* (або символьні)
- Блок-орієнтовані пристрої
  - зберігають інформацію у блоках фіксованого розміру, які мають адресу
  - дозволяють здійснювати пошук
  - *Приклад:* диск
- Байт-орієнтовані пристрої
  - не підтримують адреси
  - не дозволяють здійснювати пошук
  - генерують або споживають послідовність байтів
  - *Приклади:* термінал, клавіатура, мережний адаптер
- Існують пристрої, які не належать ні до перших, ні до других
  - *Приклад:* таймер
- До якого типу належить оперативна пам’ять?

Контролери пристроїв

- Пристрій може мати механічний компонент
- Пристрій обов'язково має електронний компонент
  - Електронний компонент пристрою, що взаємодіє з комп'ютером, називають *контролером* (також – *адаптером*)
  - Контролер завжди має кілька *регістрів*
- ОС взаємодіє саме з контролером
- Незалежно від типу пристрою, контролер, як правило:
  - Під час операції введення перетворює потік біт у блоки, що складаються з байтів (під час виведення – навпаки)
  - Здійснює контроль і виправлення помилок

Робота з регістрами

- Операції
  - Виведення даних – це записування даних у *регістр виведення* контролеру (data out)
  - Введення даних – це зчитування даних з *регістра введення* контролеру (data in)
  - Керування пристроєм – це записування даних у *регістр керування* контролеру (control)
  - Перевірка стану пристрою і результату попередньої операції – це зчитування даних з *регістра статусу* контролеру (status)
- Варіанти архітектури комп'ютерів:
  - Регістри контролерів є частиною адресного простору оперативної пам'яті (memory-mapped I/O)
  - Регістри контролерів утворюють власний адресний простір – порти введення-виведення (I/O port)

Драйвери пристроїв

- Драйвер пристрою – це програмний модуль, що керує взаємодією ОС із конкретним зовнішнім пристроєм
  - точніше – з його контролером
- Драйвер можна розглядати як транслятор, який:
  - отримує на свій вхід команди високого рівня, які зумовлені його (універсальним) інтерфейсом з ОС
  - на виході генерує інструкції низького рівня, специфічні для конкретного контролера
- Драйвери практично завжди виконують у режимі ядра
  - Драйвери можуть бути завантажені у пам’ять і вивантажені з неї під час виконання

Способи виконання операцій введення-виведення (1)

- Опитування пристроїв
  - Драйвер у циклі зчитує біт *busy* регістру статусу, поки він не буде знятий
  - Драйвер встановлює біт *write* керуючого регістра і записує байт даних у регістр виведення
  - Драйвер встановлює біт *cready*
  - Контролер виявляє встановлений біт *cready* і встановлює біт *busy*
  - Контролер виявляє встановлений біт *write*, зчитує регістр виведення і передає отриманий байт пристрою
  - Контролер знімає біти *cready* і *busy* (операція завершена)

Способи виконання операцій введення-виведення (2)

- Введення-виведення, кероване перериваннями
  - У регістри записують команди та(або) дані
  - Контролер працює
  - Контролер викликає переривання
  - Обробник переривання перевіряє результати операції
- Прямий доступ до пам’яті (DMA)
  - Процесор дає команду DMA-контролеру виконати зчитування блоку даних з пристрою, при цьому він передає контролеру адресу буфера у фізичній пам’яті
  - DMA-контролер починає пересилання (процесор може виконувати інші інструкції)
  - Після завершення пересилання усього блоку (наприклад, 4 кБ), DMA-контролер генерує переривання
  - Оброблювач переривання завершує обробку операції

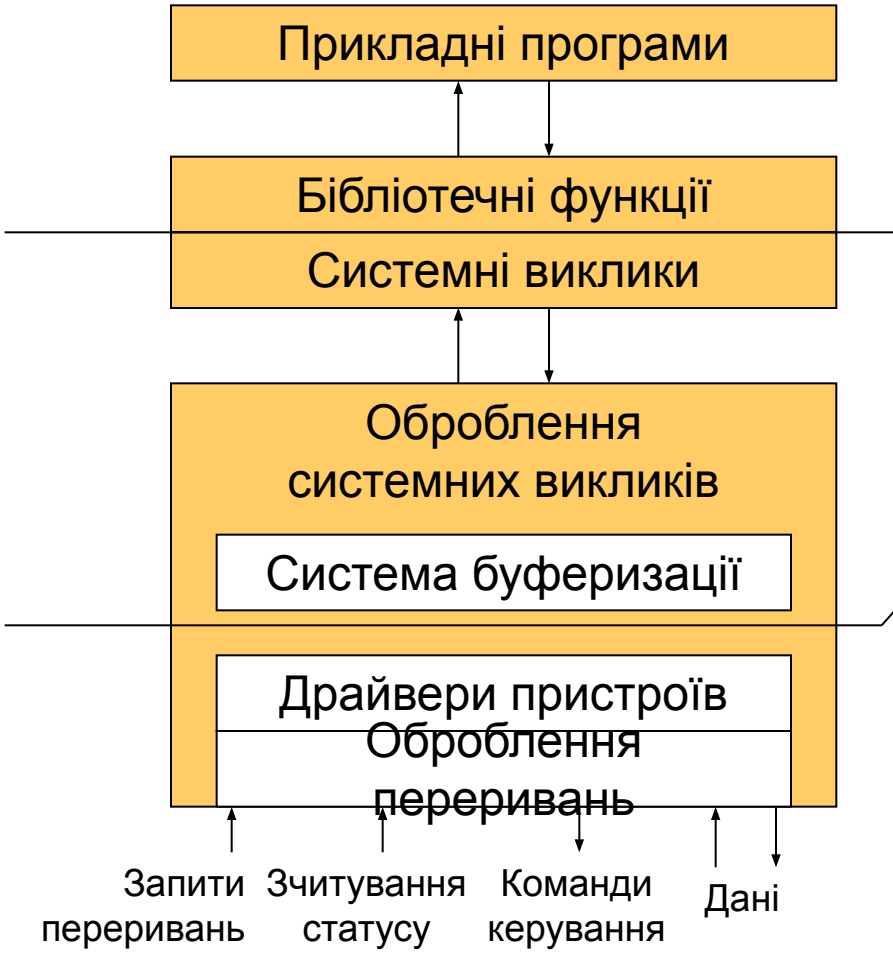
Особливості оброблення переривань

- Рівні переривань
  - Деякі переривання більш важливі, деякі – менш
  - Переривання поділяють на рівні відповідно до їхнього пріоритету (IRQL – *Interrupt Request Level*)
  - Можна *маскувати* переривання, які нижчі певного рівня (тобто, скасовувати їх оброблення)
- Встановлення оброблювачів переривань
  - Апаратні переривання обробляються *контролером переривань*, який надсилає процесору сигнал різними лініями (IRQ line)
    - Раніше контролери переривань були розраховані на 15 ліній
    - Сучасні контролери розраховані на більше ліній (Intel – 255)
  - Оброблювач переривання встановлюється драйвером пристрою, при цьому драйвер повинен визначити лінію IRQ, яку буде використовувати пристрій
    - Раніше користувачі мали вручну обрати номер лінії IRQ
    - Для деяких стандартних пристроїв номер лінії IRQ документований і незмінний
    - Драйвер може виконувати *зондування* (*probing*) пристрою. Цей підхід вважається застарілим
    - Пристрій сам повідомляє номер лінії. Цей підхід є найкращим

Особливості реалізації оброблювачів переривань

- В оброблювачах дозволено виконувати більшість операцій, за деякими винятками:
  - Не можна обмінюватись даними з адресним простором режиму користувача (оброблювач не виконується у контексті процесу)
  - Не можна виконувати жодних дій, здатних спричинити очікування
    - Явно викликати sleep()
    - Звертатись до синхронізаційних об'єктів з викликами, які можна заблокувати
    - Резервувати пам'ять за допомогою операцій, що призводять до сторінкового переривання
- Відкладене оброблення переривань
  - Для запобігання порушенню послідовності даних оброблювач переривань має на певний час блокувати (маскувати) переривання
  - Оброблювач повинен швидко завершувати свою роботу, щоби не тримати заблокованими наступні переривання
  - Для підвищення ефективності у сучасних ОС код оброблювача поділяють на дві половини:
    - *Верхня половина* (*top half*) – безпосередньо оброблювач переривання
    - *Нижня половина* (*bottom half*) – реалізує *відкладене оброблення переривання*. Не маскує переривання!

Ієрархія рівнів програмного забезпечення введення-виведення



- Користувальницький рівень ПЗ
- Незалежний від пристроїв рівень
- Драйвери пристроїв
- Оброблення переривань

Рівень, незалежний від пристроїв

- Забезпечення спільного інтерфейсу до драйверів пристроїв
- Іменування пристроїв
- Захист пристроїв
- Буферизація
- Розподіл і звільнення виділених пристроїв
- Повідомлення про помилки

Синхронні та асинхронні операції введення-виведення

- Введення-виведення на рівні апаратного забезпечення є керованим перериваннями, а отже – асинхронним
- На користувальницькому рівні організувати синхронне оброблення даних значно простіше, ніж асинхронне
  - Потік робить блокувальний (синхронний) виклик
  - Ядро переводить потік у стан очікування пристрою
  - Після завершення операції введення-виведення викликається переривання
  - Обробник переривання переводить потік у стан готовності
- Синхронне введення-виведення не підходить для:
  - Серверів, що обслуговують багатьох клієнтів
  - Програм, що працюють з журналом
  - Мультимедійних застосувань

Асинхронне введення-виведення на користувальницькому рівні (1)

- Багатопотокова організація
  - У процесі створюють новий потік, який виконує звичайне синхронне введення-виведення
  - Проблеми:
    - Потрібно реалізовувати синхронізацію потоків
    - Може знизитись надійність прикладної програми
- Введення-виведення з повідомленнями (notification-driven I/O, I/O multiplexing)
  - Застосовується тоді, коли треба у циклі виконати блокувальний виклик для кількох файлових дескрипторів
    - Один із викликів може заблокувати потік тоді, коли на іншому дескрипторі є дані
  - Реалізують спеціальний виклик, який дозволяє потоку отримати повідомлення про стан дескрипторів
    - Після цього можна у циклі працювати з усіма дескрипторами, для яких у поточний момент введення-виведення можливе (блокування потоку не виникне)

Асинхронне введення-виведення на користувальницькому рівні (2)

- Асинхронне введення-виведення
  - Основна ідея – потік, який почав виконувати введення-виведення, не блокують
  - Послідовність:
    - Потік виконує системний виклик асинхронного в-в, який ставить операцію в-в у чергу і негайно повертає керування
    - Потік продовжує виконання паралельно з операцією в-в
    - Коли операцію завершено, потік отримує повідомлення
  - У разі асинхронного в-в не гарантується послідовність операцій
    - Якщо потік виконує поспіль операції 1 і 2, то цілком ймовірно, що операція 2 буде виконана до операції 1
- Порти завершення введення-виведення (I/O completion port)
  - Поєднує багатопотоковість з асинхронним в-в
  - Має переваги для серверів, що обслуговують велику кількість одночасних запитів
  - Принцип:
    - Створюють *пул потоків* (*thread pool*)
    - Кожний потік готовий до обслуговування запитів в-в
    - Кількість потоків обирають виходячи з кількості наявних процесорних ядер