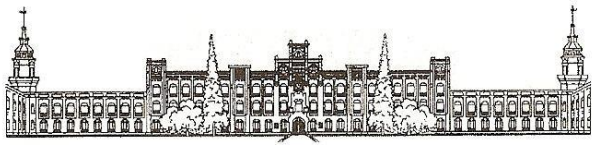

Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут»

Розрахунково-графічна робота

*з дисципліни «**Введення до операційних систем**»*

Виконав студент III курсу

групи: КВ-11

ПІБ: Терентьєв Іван Дмитрович

Перевірив: _____

Київ 2024

Питання за варіантом №23

Питання №1: у вигляді якої програмної структури оформлюються системні виклики вводу-виводу.

Системні виклики введення-виведення зазвичай оформлюються як синхронні процедури, оскільки ці операції займають багато часу, і користувацькому процесу або потоку все одно доведеться чекати результатів для продовження роботи.

Щоб забезпечити ефективне управління операціями введення-виведення, операційні системи використовують різні структури даних:

- Файлові дескриптори: числові ідентифікатори використовуються для позначення відкритих файлів і пристроїв. Вони дозволяють програмам взаємодіяти з конкретними файлами або пристроями за допомогою системних викликів.
- Буфери введення-виведення: для оптимізації продуктивності використовуються буфери, які зберігають тимчасові дані під час читання або запису.

Питання №2: які переваги та недоліки способу простого перерахування номерів кластерів, що зайняті файлом.

Спосіб простого перерахування номерів кластерів, що зайняті файлом, має свої переваги та недоліки, розглянемо їх.

Переваги:

1. ***Простота реалізації:*** перерахування кластерів безпосередньо дозволяє легко реалізувати алгоритм для зберігання даних про цей файл.
2. ***Швидкий доступ:*** оскільки всі кластери файлу перераховані безпосередньо, доступ до даних може бути відносно швидким, якщо не потрібно часто переміщатися між ними.
3. ***Пряма адресація:*** такий спосіб дозволяє прямий доступ до кожного кластеру файлу, що може спростити деякі операції, такі як читання або запис конкретних частин файлу.

Недоліки:

1. ***Великий обсяг метаданих:*** збереження переліку всіх кластерів, зайнятих файлом, може вимагати багато простору, особливо для великих файлів. Це призводить до значного збільшення розміру метаданих.
2. ***Фрагментація:*** якщо файл сильно фрагментований, та кластерів, що його складають багато, і вони розташовані не послідовно, пошук і читання файлу може бути повільнішим, оскільки система постійно буде перемикатися між різними частинами диска.
3. ***Обмежена масштабованість:*** для дуже великих файлів або файлових систем з великою кількістю файлів підтримка переліку кластерів стає все більш обтяжливою і може призвести до погіршення продуктивності.
4. ***Складність управління:*** управління списком кластерів для кожного файлу може стати складним завданням, особливо якщо файл часто змінюється, додаються нові дані та/або видаляються старі.

Висновок: спосіб простого перерахування номерів кластерів підходить для невеликих і не сильно фрагментованих файлових систем. Для великих і складних систем варто розглянути альтернативні методи.

Питання №3: які проблеми вводу-виводу вирішуються використанням багаторівневих драйверів.

Використання багаторівневих драйверів дозволяє вирішувати кілька проблем введення-виведення, підвищуючи ефективність та надійність системи, розглянемо основні проблеми, що вирішуються таким підходом.

1. **Апаратна абстракція:**
   - ***Проблема:*** без багаторівневих драйверів кожен додаток або система повинні працювати безпосередньо з конкретним апаратним забезпеченням, що ускладнює розробку і підтримку.
   - ***Вирішення:*** багаторівневі драйвери забезпечують абстракцію апаратного забезпечення, що дозволяє програмному забезпеченню взаємодіяти з уніфікованим інтерфейсом, не турбуючись про деталі конкретного пристрою.
2. **Портативність:**
   - ***Проблема:*** різні апаратні платформи вимагають різних драйверів, що ускладнює перенесення програмного забезпечення між системами.
   - ***Вирішення:*** завдяки багаторівневим драйверам можна створити загальний інтерфейс, що спрощує перенесення програмного забезпечення на різні апаратні платформи.
3. **Розширюваність:**
   - ***Проблема:*** Додавання нового апаратного забезпечення часто потребує змін в існуючих драйверах або додатках.
   - ***Вирішення:*** Багаторівневі драйвери забезпечують модульність, дозволяючи додавати нові драйвери без зміни основної логіки системи або додатків.
4. **Підтримка різноманітних пристроїв:**
   - ***Проблема:*** Управління різними типами пристроїв вимагає різних підходів і може бути складним.
   - ***Вирішення:*** Багаторівневі драйвери можуть надавати спеціалізовані шари для різних типів пристроїв, спрощуючи їх підтримку і управління.
5. **Підвищення надійності та стійкості:**
   - ***Проблема:*** Помилки в драйверах можуть призвести до збоїв системи або пошкодження даних.
   - ***Вирішення:*** Розділення функціональності на різні рівні дозволяє локалізувати і легше усувати помилки, не впливаючи на всю систему.

6. **Оптимізація продуктивності:**
    - ***Проблема:*** Помилки в драйверах можуть призвести до збоїв системи або пошкодження даних.
    - ***Вирішення:*** Розділення функціональності на різні рівні дозволяє локалізувати і легше усувати помилки, не впливаючи на всю систему.

Висновок: багаторівневість драйверів дозволяє розробникам зосередитися на своїй області відповідальності і не турбуватися про інші рівні, що робить систему більш гнучкою, надійною та легкою в обслуговуванні.

Питання №4: які переваги та недоліки методу розміщення файлу у вигляді зв'язного списку кластерів.

Метод розміщення файлу у вигляді зв'язного списку кластерів має свої переваги та недоліки, розглянемо їх.

Переваги:

1. ***Гнучкість у використанні дискового простору:***
   - Файл може бути розподілений по кластерам, які не обов'язково знаходяться поруч один з одним. Це дозволяє ефективно використовувати дисковий простір, навіть якщо він фрагментований.
2. ***Легка модифікація файлів:***
   - Додавання нових кластерів до файлу просто здійснюється шляхом оновлення вказівника на наступний кластер у списку. Це особливо корисно для файлів, розмір яких часто змінюється.
3. ***Мінімальні втрати через фрагментацію:***
   - У разі видалення частини файлу або всього файлу, кластери можна використовувати для інших файлів без потреби дефрагментації.

Недоліки:

1. ***Повільний послідовний доступ:***
   - Для доступу до певного кластеру необхідно пройти через всі попередні кластери, що може призвести до значних затримок при читанні або записі великих файлів.
2. ***Збільшення кількості операцій введення-виведення:***
   - Пошук наступного кластеру потребує додаткових операцій введення-виведення, що може негативно вплинути на загальну продуктивність системи, особливо при обробці великих файлів.
3. ***Високі вимоги до метаданих:***
   - Кожен кластер повинен містити посилання на наступний кластер у списку, що збільшує обсяг метаданих і ускладнює структуру файлової системи.
4. ***Проблеми з надійністю:***
   - Кожен кластер повинен містити посилання на наступний кластер у списку, що збільшує обсяг метаданих і ускладнює структуру файлової системи.

Висновок: метод розміщення файлу у вигляді зв'язного списку кластерів є ефективним для гнучкого управління дисковим простором і дозволяє

легко модифікувати файли. Однак він має суттєві недоліки, такі як повільний послідовний доступ і підвищені вимоги до метаданих, що можуть вплинути на загальну продуктивність та надійність файлової системи. Цей метод підходить для середовищ, де фрагментація є серйозною проблемою, але менш придатний для застосувань, які вимагають швидкого доступу до великих обсягів даних.