Deadlock - (взаємне блокування)

Що таке Deadlock (взаємне блокування)?

Deadlock (дедлок) — це стан у багатопоточних або багатозадачних системах, коли дві або більше задач блокують одна одну, чекаючи на ресурси, які вже зайняті іншими задачами. У такій ситуації задачі не можуть завершити роботу, і система «зависає».

Умови виникнення Deadlock (Coffman conditions):
Щоб виник deadlock, повинні виконуватися **чотири умови одночасно**:

1. **Взаємне блокування ресурсів (Mutual Exclusion):** Ресурс може бути використаний лише одним процесом у певний момент часу.
2. **Утримання та очікування (Hold and Wait):** Процес, який вже утримує один ресурс, запитує додаткові ресурси, не відмовляючись від уже зайнятих.
3. **Відсутність примусового звільнення (No Preemption):** Ресурс не може бути примусово відібраний у процесу, він звільняється лише добровільно після завершення процесу.
4. **Циклічне очікування (Circular Wait):** Утворюється цикл процесів, де кожен процес чекає ресурс, зайнятий іншим процесом у цьому циклі.

2. Dining Philosophers Problem (Задача обідаючих філософів)
Ця задача — класичний приклад дедлоку. Умови:

- Є N філософів, які сидять за круглим столом.
- Перед кожним філософом є тарілка їжі та по одній виделці між кожною парою філософів.
- Щоб їсти, філософу потрібно взяти дві виделки (ліву та праву).
- Якщо кожен філософ бере одну виделку, але чекає на другу, виникає дедлок.

Рішення:

1. **Ієрархічне замовлення (Hierarchical Ordering):** Філософи беруть виделки в певному порядку, щоб уникнути циклу.
2. **Чергування (Alternating Strategy):** Кожен другий філософ змінює порядок, у якому бере виделки (спочатку ліву, потім праву).
3. **Використання таймаутів:** Якщо виделка не доступна певний час, філософ відкладає свою спробу.

Щоб симулювати deadlock (взаємне блокування) в задачі про філософів або ресурсів, потрібно створити сценарій, де два або більше процесів чекають ресурси, захоплені іншими процесами. У моєму випадку це можна зробити шляхом коректного налаштування команд у файлах процесів (a0.dat, a1.dat і т.д.).

Про deadlock simulator(GUI), він для і що показує:

1. Структура виводу

Програма виводить інформацію про стан ресурсів і процесів у симуляції на кожному кроці:
time = <час> available = <доступно ресурсу 1> <доступно ресурсу 2> blocked = <кількість заблокованих процесів>

2. Що це означає

1. time = <час>:
   - Показує, на якому "кроці часу" знаходиться симуляція.
2. available = <доступно ресурсу 1> <доступно ресурсу 2>:
   - Кількість доступних одиниць кожного ресурсу в системі.
   - Наприклад, available = 5 5 означає, що ресурс 1 має 5 доступних одиниць, і ресурс 2 також має 5 одиниць.
3. blocked = <кількість заблокованих процесів>:
   - Кількість процесів, які заблоковані через брак ресурсів.

3. Як інтерпретувати(це при початкових даних з умови лабораторної роботи у файлах a0.dat, a1.dat)

- time = 0 ... time = 9:
  - Усі ресурси доступні (available = 5 5).
  - Немає заблокованих процесів (blocked = 0).
- time = 10 ... time = 19:
  - Один із процесів захопив 2 одиниці ресурсу 1 (available = 3 5).
  - Процеси все ще виконуються нормально, і блокувань немає (blocked = 0).
- time = 20:
  - Ресурс 1 повернув свої одиниці (available = 5 5).
  - Жоден процес не блокується.

4. Що далі?
Цей вивід показує, що система працює без взаємного блокування (deadlock) або заблокованих процесів(starvation).

Про Xeyes та XQuartz

Що таке XQuartz?

XQuartz — це реалізація **X11** (графічного сервера) для macOS. X11 — це протокол, який дозволяє графічним програмам працювати в мережі або віддалено (наприклад, з контейнерів Docker або Linux-систем).
Роль XQuartz у лабі:

- Docker-контейнер (Ubuntu) запускає програму з графічним інтерфейсом.
- Програма, така як `xeyes` або симуляція дедлоку, потребує графічного дисплея.
- Оскільки Docker-контейнер працює у "головному режимі" (headless), він не має свого дисплея. XQuartz дозволяє контейнеру підключитися до графічного середовища Mac.

Що таке Xeyes?

Xeyes — це проста програма для тестування X11-з'єднань. Вона показує пару очей, які слідкують за рухами курсора.
Роль Xeyes у лабі:

- Використовується для перевірки, чи ваш Docker-контейнер правильно підключений до XQuartz.
- Якщо `xeyes` працює, це означає, що графічний інтерфейс симуляції також зможе працювати.

Docker:

Роль Docker у моїй лабі:

- Я використовувала Docker для створення середовища Ubuntu на Mac.
- У контейнері встановила Java, X11 та інші інструменти для запуску симуляції дедлоку.

Робота з файлами з умов лаби у терміналі через докер:

Використовую nano

Nano: Простий текстовий редактор у терміналі
Nano — це легкий і зручний текстовий редактор для термінала. Його основна перевага — простота використання. Nano встановлений майже у всіх дистрибуціях Linux за замовчуванням і є ідеальним інструментом для редагування текстових файлів прямо з командного рядка.

Завдання Nano у лабораторній роботі

- Nano використовувався для редагування файлів `Kernel.java`, `a0.dat` тощо.
- Я використовувала його для виправлення помилок у коді та додавання нових рядків.
- Він дозволив швидко зберігати зміни й перевіряти їх у середовищі Docker.

Ubuntu

Ubuntu — це популярна дистрибуція Linux, яка широко використовується для розробки, навчання та серверних задач.
Роль Ubuntu у лабі:

- Ubuntu в Docker забезпечує ізольоване середовище для запуску Java-програм, таких як ваша симуляція.
- Це дозволяє уникнути проблем із конфігурацією, які могли б виникнути на macOS.

Теоретичні кроки виконання лабораторної роботи

(1) Встановлення середовища

- Використання Docker для створення контейнера з Ubuntu.
- Встановлення Java Development Kit (JDK) і X11-програм.

(2) Перевірка X11

- Тестування підключення XQuartz через xeyes.

(3) Запуск симуляції дедлоку

- Використання програми java deadlock для симуляції дедлоку.
- Налаштування початкових параметрів (наприклад, кількість процесів, ресурси).

(4) Аналіз результатів

- Результат у графічному інтерфейсі (run, step, reset).
- Логічний аналіз: коли обидва процеси очікують ресурси, які недоступні, виникає дедлок.

(5) Збереження результатів

- Перенаправлення текстового виводу в файл (results.txt).
- Збереження резервної копії директорії з проєктом.

Як усе пов’язано між собою?

1. **Docker створює ізольоване середовище (Ubuntu), де можна виконувати програми.**
2. **Ubuntu забезпечує платформу для запуску Java та інших інструментів.**
3. **XQuartz дозволяє Ubuntu-контейнеру підключитися до графічного дисплея Mac.**
4. **Xeyes — це тестова програма для перевірки роботи XQuartz.**
5. **Deadlock-симулятор демонструє дедлок як приклад некоректної взаємодії між процесами та ресурсами.**