Тема №5: Взаємодія процесів через спільні змінні. Завдання взаємного виключення і синхронізації та засоби її вирішення: атомарні змінні, семафори, мютекси, події, критичні секції, монітори

Питання:

1. Завдання взаємного виключення і синхронізації та засоби її вирішення
2. Синхронізація по подіях
3. Критична секція
4. Семафор
5. М'ю́текс
6. Монітор
7. Атомарні змінні

1. Завдання взаємного виключення і синхронізації та засоби її вирішення

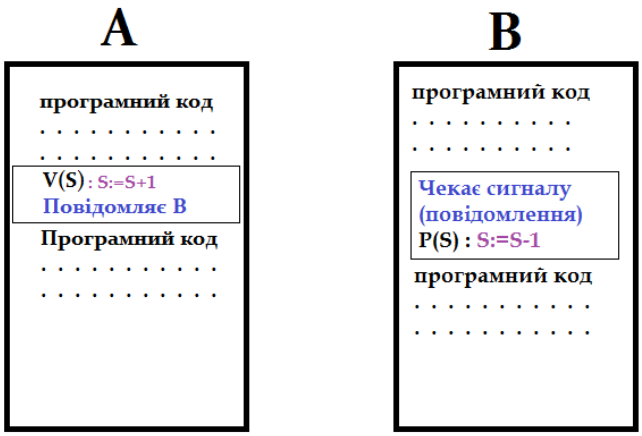
Синхронізація процесів — приведення двох або декількох процесів або потоків (нитей) до такого їхнього протікання, коли певні стадії різних процесів здійснюються в певному порядку, або одночасно, для уникнення конкуренції потоків або взаємного блокування. Загальна ідея полягає в тому, що в певних точках процесам необхідно разом домовитися про певний порядок дій зі спільними ресурсами.

Синхронізація необхідна у випадках, коли паралельно протікаючим процесам (або потокам одного процесу) необхідна взаємодія.

Задача синхронізації

Постановка задачі: є два потоки, що виконуються паралельно. Перший потік (назвемо його А), виконується і в певному місці коду (критична ділянка коду) має відправити повідомлення другому потоку (нехай буде В). Потік В, виконується і дійшовши до певної частини коду(критична ділянка коду), блокується, поки не отримає повідомлення від потоку А. Невідомо, який з потоків першим дійде до критичної частини коду. Потрібно їх синхронізувати.

Розв"язок задачі: Для розв'язання цієї задачі достатньо використати двійковий семафор(S=0,1). На початку обов'язково семафор S встановити в нуль:



Оскільки потоки виконуються паралельно, тобто водночас, то заздалегідь невідомо, який з потоків дійде до критичної ділянки коду першим. Існує два випадки:

Нехай першим дійшов потік А. Операція V(S) збільшить семафор на 1 і потік А продовжить виконуватися. Тоді, коли до критичної ділянки дійде потік В, він продовжить своє виконання. При цьому операція P(S) зменшить семафор на 1.

Нехай першим дійде потік В. Перевіривши семафор, він блокується, поки не буде повідомлення(сигналу) від потоку А. Тобто, поки семафор дорівнює нулю. Коли потік А

відправить сигнал, тобто, збільшить семафор на 1, тоді продовжить виконуватися потік B, зменшивши при цьому семафор на 1.

Прикладом такої синхронізації може бути процес: Нехай потоку B для виконання потрібен ресурс, який формує потік A. Тому потоку B доводиться чекати, поки потік A не сформує цей ресурс.

Засоби синхронізації

- Найпоширеніші засоби синхронізації такі:
- сигнали і повідомлення,
- критичні секції,
- м'ютекси
- семафори
- події
- бар'єри,
- канали (англ. pipe).

2. Синхронізація по подіях

Найпростішою функцією, яка очікує завершення переходу заданого об’єкта у сигнальний стан, є WaitForSingleObject.

Її опис мовою C[1]:

```
DWORD WINAPI WaitForSingleObject(
 __in  HANDLE hHandle,
 __in  DWORD dwMilliseconds
);
Опис мовою Delphi[2]:
function WaitForSingleObject(
  hHandle: THandle;          // Дескриптор об’єкта
  dwMilliseconds: DWORD      // Час очікування
): DWORD;
```

Функція очікує переходу об'єкта з дескриптором hHandle у сигнальний стан протягом dwMilliseconds мілісекунд. Якщо цим параметром передати значення INFINITE, функція буде чекати протягом необмеженого часу. Якщо dwMilliseconds дорівнює 0, то функція перевіряє стан об'єкта й негайно повертає керування. Дескриптор повинен бути відкритий з правом доступу SYNCHRONIZE.

Функція повертає одне з перелічених у таблиці значень.

Значення	Опис
WAIT_OBJECT_0	Об'єкт перейшов у сигнальний стан
WAIT_ABANDONED	Означає, що заданий об’єкт є м'ютексом, і потік, який володів ним, завершився, не звільнивши його. Власником м'ютекса стає викликаючий потік, а сам м'ютекс переводиться у несигнальний стан
WAIT_TIMEOUT	Вийшов час очікування
WAIT_FAILED	Відбулася помилка (наприклад, невірне значення hHandle)

Синхронізація кількох об'єктів

Для синхронізації кількох об'єктів використовують фунцію API WaitForMultipleObjects. ЇЇ опис мовою C:

```
DWORD WINAPI WaitForMultipleObjects(
 __in  DWORD nCount,
```

```
  __in  const HANDLE *lpHandles,
  __in  BOOL bWaitAll,
  __in  DWORD dwMilliseconds
);
```

Вказівник lpHandles посилається на масив, який містить дескриптори об’єктів очікування. Параметр nCount задає кількість елементів цього масиву. Якщо параметру bWaitAll задати значення true, функція очікуватиме на перехід у сигнальний стан усіх перелічених у масиві lpHandles об’єктів. Коли цей параметр дорівнює false, функція чекатиме на сигнальний стан одного з них. Параметр dwMilliseconds задає час очікування (як для функції WaitForSingleObject). Дескриптори повинні мати право доступу SYNCHRONIZE. Функція повертає одне з перелічених у таблиці значень.

3. Критична секція (англ. critical section) — об'єкт синхронізації нитей у Windows, що дозволяє запобігти одночасному виконанню деякогокритичного набору операцій (зазвичай пов'язаних з доступом до даних) кількома нитями.

Об'єкт критичної секції представляє ділянку сирцевого коду, яка в кожен окремий момент може виконуватися лише одною ниттю. Критичні секції можна створювати і знищувати, але вони не мають дескрипторів, тому не можуть використовуватись різними процесами. У той же час, для синхронізації нитей одного процесу критичні секції – один з найшвидших і найпростіших способів. Об'єкти критичних секцій не є об'єктами ядра, а розміщуються у користувацькому адресному просторі процесу[1].

В більшості реалізацій з кожною критичною секцією пов'язано лічильник, і захоплення критичної секції можливе лише за нульового значення цього лічильника. Захоплення означає атомарну зміну лічильника на 1. Щоразу, коли нить, що володіє критичною секцією, знов входить до неї, лічильник збільшується на 1, при виході — зменшується. Перехід до чекання на події ядра здійснюється лише у випадку, якщо значення змінної лічильника до спроби захоплення було вже більше 1, тобто критична секція *зайнята* і є реальне «змагання» двох або більше нитейза ресурс критичної секції.

Аналогічний об'єкт в Linux називається ф'ютекс.

Порівняння критичних секцій та м'ютексів

Критична секція значною мірою виконує ті ж завдання, що і м'ютекс.

У операційних системах Microsoft Windows головна відмінність між м'ютексом і критичною секцією в тому, що м'ютекс є об'єктом ядра і може бути використаний кількома процесами одночасно, критична секція ж належить процесу і служить для синхронізації тільки його нитей. Процедура входу і виходу критичних секцій зазвичай займає менший час, ніж аналогічні операції м'ютекса.

Між м'ютексом і критичною секцією є й термінологічні відмінності. Процедура, аналогічна захопленню м'ютекса, називається входом у критичну секцію, зняття блокування м'ютекса — виходом з критичної секції.

Для кожної критичної секції система підтримує лічильник входжень, тому нить повинна звільнити її стільки разів, скільки захоплювала. Недотримання цієї умови приведе до блокування нитіі ("зависання").

Слід також пам'ятати, що спроба виходу з критичної секції, якою нить не володіє, також приведе до блокування викликаючої нитіі.

Програма може здійснити спробу захоплення об'єкта критичної секції без зупинки нитіі. Для цього використовують функцію TryEnterCriticalSection.

4. Семафор — це універсальний механізм для організації взаємодії процесів (в термінології операційних систем сімейства Windows —потоків).

Розв'язує задачі взаємного виключення та синхронізації потоків. Він є одним з найстаріших засобів розподілення доступу процесів, що працюють паралельно, до критичних ресурсів. Семафори використовуються для контролю доступу до спільного ресурсу, або для синхронізації процесів (потоків).

Визначення семафору зроблено нідерландським вченим Едсгером Дейкстрою, деякий час використовувався термін *Семафор Дейкстри*.

Семафор — це об'єкт ядра ОС, який можна розглядати як лічильник, що містить ціле число в діапазоні від 0 до заданого при його створенні максимального значення[1]. При досягненні семафором значення 0 він переходить у несигнальний стан, при будь-яких інших значеннях лічильника – його стан сигнальний.

Традиційне позначення семафора : S. Операції, які можна виконати над семафором:

Ініціалізація - встановлення початкового значення семафору.

Операція P(S): Вона перевіряє стан семафору. Якщо семафор не рівний нулю, то виконується операція S:=S-1. Інакше, процес блокується, поки S=0.

Операція V(S): Ця операція збільшує значення семафору на 1. Тобто виконується операція S:=S+1.

Типи семафорів

- В залежності від значень, які може приймати семафор він поділяється на:
- Двійковий : здатний приймати значення 0 та 1.
- Трійковий : здатний приймати значення 0, 1 та 2.

5. М'ю́текс (англ. mutex, від англ. mutual exclusion — об’єкт взаємного виключення (mutual exclusion)), призначений для захисту певного об’єкта у потоці від доступу інших потоків. М'ютекс є одним із засобів синхронізації роботи потоків або процесів.

М'ютекси є одним з варіантів семафорних механізмів для організації взаємного виключення. Вони реалізовані в багатьох ОС, їхнє основне призначення — організація взаємного виключення для потоків з одного і того ж або різних процесів.

М'ютекси — це прості двійкові семафори, які можуть перебувати в одному з двох станів, — сигнальному або несигнальному (відкритий і закритий відповідно). Коли потік стає власником м'ютекса, він переводиться в несигнальний стан.

Організація послідовного доступу до ресурсів з використанням м'ютексів стає нескладною, оскільки в кожен конкретний момент тільки один потік може володіти цим об'єктом. Для того, щоб об'єкт mutex став доступний потокам, що належать різним процесам, при створенні йому необхідно присвоїти ім'я. Потім це ім'я потрібно передати «у спадок» завданням, які повинні його використовувати для взаємодії. Для цього вводяться спеціальні системні виклики (наприклад CreateMutex у Windows), в яких указується початкове значення м'ютекса і його ім'я. Для роботи з м'ютексом є кілька функцій. Крім вже згаданої функції створення такого об'єкта (CreateMutex), є функції відкриття (OpenMutex) і функція звільнення цього об'єкта (ReleaseMutex). Конкретні звернення до цих функцій і переліки передаваних і отримуваних параметрів потрібно дивитися в документації на відповідну ОС.

Порівняння м'ютексів та критичних секцій

Порівняно з критичною секцією його функціональність дещо розширена. М’ютекси можуть мати імена і дескриптори. Вони є об’єктами ядра (в той час як критичні секції належать процесу), тому їх можна використовувати для синхронізації потоків із різних процесів. Наприклад, двапроцеси, які розділяють спільну пам’ять через відображення файлів, можуть використовувати м’ютекс для синхронізації доступу до неї. Так як м’ютекси є об’єктами ядра і мають складнішу реалізацію, на їх обробку, як правило, витрачається більше часу порівняно з критичними секціями. Тому, якщо потрібно синхронізувати роботу потоків одного процесу, варто користуватися критичними секціями.

6. Монітор - в мовах програмування, високорівнева механізм взаємодії і синхронізації процесів, що забезпечує доступ до неподілюваний ресурсам. [1] Підхід до синхронізації двох або більше комп'ютерних завдань, що використовують загальний ресурс, обичноаппаратуру або набір змінних.

При багатозадачності, заснованої на моніторах, компілятор або інтерпретатор прозоро для програміста вставляє код блокування-розблокування в оформлені відповідним чином процедури, позбавляючи програміста від явного звернення до примітивів синхронізації.

Монітор складається з:

- набору процедур, взаємодіючих із загальним ресурсом
- мьютекса
- змінних, пов'язаних з цим ресурсом
- інваріанта, який визначає умови, що дозволяють уникнути стан гонки

Процедура монітора захоплює мьютекс перед початком роботи і тримає його або до виходу з процедури, або до моменту очікування умови (див. Нижче). Якщо кожна процедура гарантує, що перед звільненням мьютекса інваріант правдивий, то ніяка завдання не може отримати ресурс в стані, провідному до гонки.

Простий приклад. Розглянемо монітор, що виконує транзакції банківського рахунку.

```
monitor account {
  int balance := 0

  function withdraw(int amount) {
    if amount < 0 then error "Счёт не может быть отрицательным"
    else if balance < amount then error "Недостаток средств"
    else balance := balance - amount
  }

  function deposit(int amount) {
    if amount < 0 then error "Счёт не может быть отрицательным"
    else balance := balance + amount
  }
}
```

Інваріант тут просто стверджує, що баланс повинен відобразити всі минулі операції до того, як почнеться нова операція. Зазвичай це не виражено в коді, але мається на увазі і може бути згадано в коментарях. Однак, є мови програмування, такі какЕйфель або D, які можуть перевіряти інваріанти. Блокування додана компілятором. Це робить монітори безпечніше і зручніше, ніж інші підходи, що вимагають від програміста вручну додавати операції блокування-розблокування, - оскільки програміст може забути додати їх.

7. Атомарні змінні

Щоб уникати стану активного очікування, процеси повинні сигналізувати один одному про очікувані події. Монітори забезпечують цю можливість за допомогою **атомарніх** змінних. Коли процедура монітора вимагає для подальшої роботи виконання певного умови, вона чекає пов'язану з ним умовну змінну. Під час очікування вона тимчасово відпускає мьютекс і вибуває зі списку виконуються процесів. Будь-який процес, який в подальшому призводить до виконання цієї умови, використовує умовну змінну для оповіщення чекаючого її процесу. Сповіщений процес захоплює мьютекс назад і може продовжувати.

Наступний монітор використовує умовні змінні для реалізації каналу між процесами, який може зберігати одномоментно тільки одне ціле значення.

```
monitor channel {
  int contents
  boolean full := false
  condition snd
  condition rcv

  function send(int message) {
    while full do wait(rcv)   // семантика Mesa: см.ниже
    contents := message
    full := true
    notify(snd)
  }

  function receive() {
    var int received
```

```
      while not full do wait(snd)   // семантика Mesa: см.ниже
      received := contents
      full := false
      notify(rcv)
      return received
    }
  }
```

оскільки очікування умови відпускає блокування, що очікує процес повинен гарантувати дотримання інваріанта перед тим, як почати очікування. У прикладі вище, то ж справедливо і для оповіщення.

<u>Вправи і завдання до теми №5</u>

Дайте відповідь на питання і опишіть роботу наступних засобів синхронізації потоків:

1. **Завдання взаємного виключення і синхронізації та засоби її вирішення**
2. **Синхронізація по подіях**
3. **Критична секція**
4. **Семафор**
5. **М'ю́текс**
6. **Монітор**
7. **Атомарні змінні**