Концепция семафора и ее использование для синхронизации процессов

Операционные системы

Серебрякова Дарья. Студентка НКАбд-04-24

Содержание

1 Введение **5**

2 Что такое семафор **7**

3 Бинарные семафоры **8**

4 Счетные семафоры **9**

5 Классические задачи синхронизации с использованием концепции семафоров **10**

6 Использование семафоров **12**

7 Преимущества семафоров **13**

8 Недостатки семафоров **14**

9 Заключение **15**

Список литературы **16**

Список иллюстраций

1.1 Эдсгер Дейкстрой 6

Список таблиц

1 Введение

Понятие семафора было введено в 1965 году нидерландским учёным Эдсгером Дейкстрой (рис. 1.1), а в 1968 году он предложил использовать два семафора для решения задачи производителя и потребителя

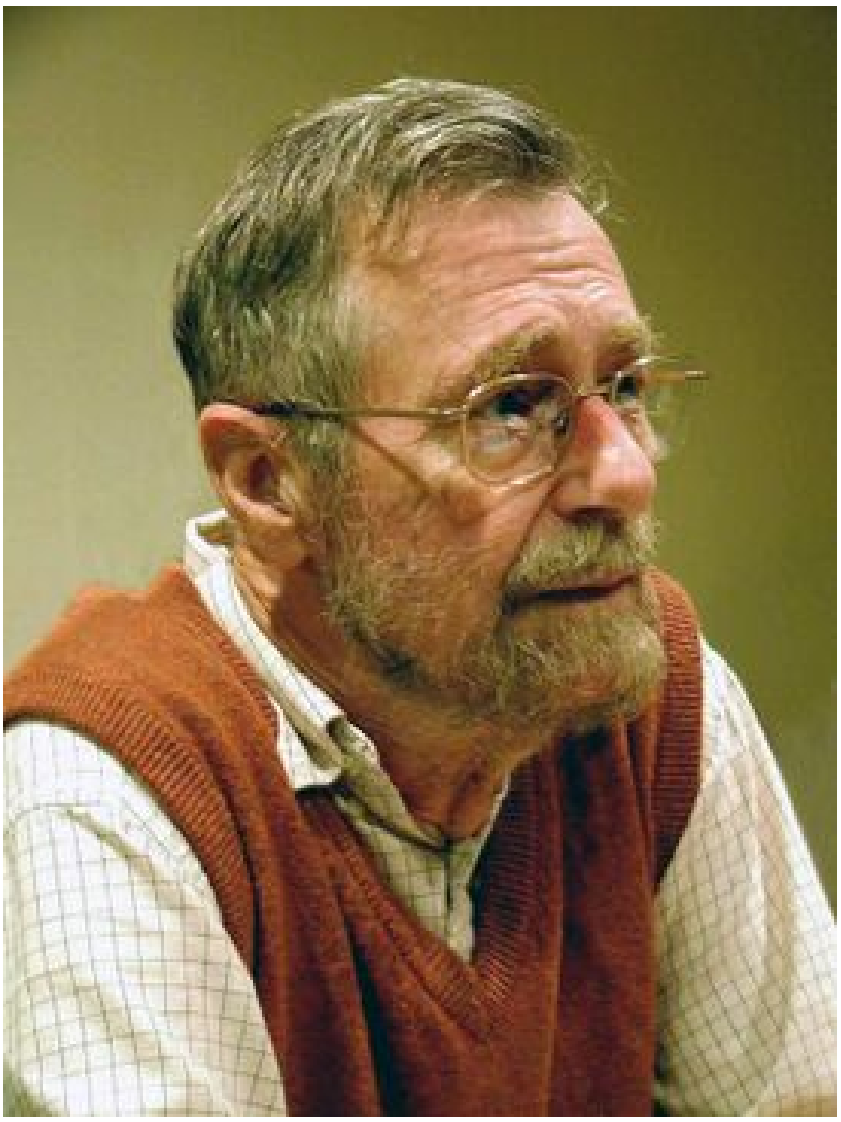

Рис. 1.1: Эдсгер Дейкстрой

2 Что такое семафор

Семафор — это синхронизирующий объект, который используется в многопоточных и многопроцессных системах для управления доступом к общим ресурсам. Он помогает избежать проблем, связанных с одновременной работой нескольких процессов с одинаковыми ресурсами, что может привести к гонкам данных и другим ошибкам.

Семафор представляет собой сигнальный механизм, в котором задача, находящаяся в режиме ожидания (waiting state), сигналит об этом другой задаче. К примеру, если задача task1 заканчивает свою работу, она передает флаг или инкремент флага на 1 и когда этот флаг принимается другой задачей (task2), это говорит ей о том, что теперь она может выполнять свою работу. Когда задача task2 заканчивает свою работу значение флага уменьшается на 1. Таким образом, мы имеем дело с механизмом “передать” и “принять” (“Give” and “Take” mechanism), а семафор является переменной целого типа, которая используется для синхронизации доступа к ресурсам. [1]

Существуют семафоры двух типов: [2]

1. Бинарные семафоры (Binary Semaphore)
2. Счетные семафоры (Counting Semaphore)

3 Бинарные семафоры

Бинарные семафоры могут принимать значения 0 и 1. В некотором смысле такой семафор похож на очередь длины 1

К примеру, у нас есть две задачи - task1 и task2. Task1 передает данные задаче task2, при этом задача task2 непрерывно проверяет, равен ли элемент очереди 1. Если он равен 1, то task2 может считывать данные, иначе должна подождать до тех пор, пока он не равен 1. После получения данных task2 декрементирует (уменьшает на 1) элемент очереди, таким образом, он становится равным 0. Это будет означать что задача task1 снова может передавать данные задаче task2

Исходя из представленного примера можно сказать, что бинарный семафор используется для синхронизации между задачами или для синхронизации между задачей и ее прерыванием

4 Счетные семафоры

Счетный семафор может принимать не только значения равные 0 и 1, поэтому он эквивалентен очереди с длиной больше 1 элемента

Этот вид семафоров используется для событий счета. В данном случае обработчик события будет “передавать” семафор каждый раз когда происходит событие (инкрементирование значения счета семафора), а обработчик задачи будет “принимать” семафор каждый раз, когда он обрабатывает событие (декрементирование значения счета семафора). Таким образом, значение счета, будет равно разнице между числом событий, которые случились (произошли), и числом событий, которые были обработаны

5 Классические задачи синхронизации с использованием концепции семафоров

1. Проблема производителя и потребителя

Проблема производителя-потребителя связана с двумя типами процессов: производителями, которые генерируют данные, и потребителями, которые обрабатывают данные. Проблема производителя и потребителя подобна ресторану, где шеф-повар (производитель) готовит еду, а клиент (потребитель) ее ест. Прилавок (буфер: очередь фиксированного размера, в которой производители размещают товары, а потребители убирают их) временно удерживает продукты. Специальный замок (семафор) гарантирует, что шеф-повар не переполнит прилавок, а клиент не возьмет еду, когда ее нет в наличии. Таким образом, все проходит гладко и эффективно и дает более быстрые результаты

Используемые Семафоры: - пустой: подсчитывает количество пустых слотов в буфере - заполнен: подсчитывает количество заполненных слотов в буфере - мьютекс: блокирует доступ к буферу (взаимное исключение)

2. Управление светофором

Светофоры на перекрёстке можно использовать для управления потоком транспорта и обеспечения безопасного перехода

Пример: Семафоры представлены в виде семафоров, которые управляют зелёным, жёлтым и красным сигналами для разных направлений Используемые семафоры: каждое направление света управляется семафором, который регулирует время и переходы между состояниями света Контроллер светофора: использует семафоры для переключения между зеленым, желтым и красным светом. Контроллер гарантирует, что только одно направление будет иметь зеленый свет в данный момент, и управляет переходами, чтобы избежать конфликтов

3. Проблема читателя и писателя

Проблема в том, что происходит синхронизация доступа к общим данным, когда несколько устройств чтения могут считывать данные одновременно, но для их изменения устройствам записи требуется эксклюзивный доступ. Проще говоря, представьте библиотеку, куда приходят несколько читателей и авторов, и все читатели хотят прочитать книгу, а некоторые люди (писатели) хотят обновить или отредактировать книгу. Вот почему нам нужна система, гарантирующая, что эти действия выполняются плавно, без ошибок или конфликтов

Значение семафора для читателя: процессы, которые считывают общие данные, могут получить доступ одновременно (значение > 1, более одного считывателя могут получить доступ одновременно) Значение семафора для писателей: процессы, которые изменяют общие данные, получают эксклюзивный доступ (значение = 1, по одному за раз)

6 Использование семафоров

1. Взаимное исключение: семафор гарантирует, что только один процесс одновременно получает доступ к общему ресурсу
2. Синхронизация процессов: семафор координирует порядок выполнения нескольких процессов
3. Управление ресурсами: ограничивает доступ к ограниченному набору ресурсов, таких как принтеры, устройства и т.д.
4. Проблема «читатель-писатель»: позволяет нескольким читателям, но ограничивает писателей до тех пор, пока не останется ни одного писателя
5. Предотвращение взаимоблокировок : предотвращает взаимоблокировки за счет контроля порядка выделения ресурсов

7 Преимущества семафоров

1. Семафор - это простой и эффективный механизм синхронизации процессов
2. Поддерживает координацию между несколькими процессами, контролируя доступ к критическим секциям. Семафоры помогают управлять несколькими процессами, не мешая им взаимодействовать друг с другом
3. При правильном использовании семафоры помогают избежать взаимоблокировок, эффективно управляя доступом к ресурсам и гарантируя, что ни один процесс не будет заблокирован на неопределённый срок в доступе к необходимым ресурсам
4. Семафоры помогают предотвратить состояние гонки, гарантируя, что только один процесс может одновременно получить доступ к общему ресурсу
5. Обеспечивает гибкий и надежный способ управления общими ресурсами

8 Недостатки семафоров

1. Это может привести к снижению производительности из-за накладных расходов, связанных с операциями ожидания и передачи сигналов
2. Если не управлять семафорами должным образом, они могут привести к зависанию. Это часто происходит, когда семафоры не освобождаются должным образом или когда процессы получают семафоры в непоследовательном порядке
3. Отладка и поддержка могут быть затруднены. Отладка систем, в которых активно используются семафоры, может быть сложной задачей, поскольку трудно отслеживать состояние каждого семафора и обеспечивать правильную синхронизацию всех процессов
4. При неправильном использовании он может привести к состояниям гонки и другим проблемам с синхронизацией
5. Он может быть уязвим для определённых типов атак, таких как атаки типа «отказ в обслуживании»

9 Заключение

Синхронизация процессов — важный аспект параллельных вычислений, обеспечивающий выполнение нескольких процессов без помех друг для друга. Семафоры можно использовать для решения классических проблем синхронизации в вычислениях. Они играют важную роль в управлении доступом к общим ресурсам и координации действий параллельных процессов. Они также помогают повысить производительность системы, сохранить целостность данных и устранить конфликты между процессами. Эффективная синхронизация процессов обеспечивает согласованность, стабильность и эффективность в многопоточных и многопроцессорных системах

Список литературы

1. MicroKontroller.ru.
2. GeeksforGeeks.