**Вопросы по лекции №9**

**Взаимодействие процессов по информации. Правила**

**защиты критических ресурсов для ПО с параллельными процессами.**

**Задачи синхронизации процессов**

Бригада СанУ11: Ковалев К.А., Терентьев Е.А., Щекочихин Д.А., Оганисян Т.С., Шабакаев А.Р.

Выполнил: Ковалев К.А.

Группа: 6411

**1.В каких случаях возникает задача обмена информацией между процессами.**

Задача обмена информацией между процессами возникает при работе в централизованной (нераспределенной) системе ПО, когда обмен данными между процессами должен проводиться через общую память.

**2. Как ОС обеспечивает решение этой задачи. Три варианта решения этой задачи**

1) Через файловую систему 2) Через специальные вызовы - средства ОС 3) Через разделяемую память задач - один процесс создает специальным вызовом ОС сегмент разделяемой памяти и сообщает её идентификатор взаимодействующим процессам.

**3.Какой из рассмотренных вариантов обмена информацией между процессами самый**

**быстрый? Какой самый медленный?**

Через файловую систему – самый медленный, через разделяемую память задач – быстрый.

**4. Какие процессы и задачи мы называем параллельными и почему не одновременными?**

Два процесса или задачи, или программы называются параллельными, если их выполнение может перекрываться во времени и второй процесс начинается до завершения первого. В случае многозадачной работы ПО на однопроцессорной ЦВМ задачи решаются последовательно, но переключения с задачи на задачу происходят как бы незаметно для задач, и мы говорим о квазиодновременном решении задач.

**5.Что такое критический ресурс ЦВМ?**

Критический ресурс – ресурс к которому в каждый момент времени может

иметь доступ только один процесс, хотя претендентов на доступ много.

**6. Каково основное правило защиты критического ресурса и его модификация?**

Основное правило защиты критических ресурсов – процесс не должен получать доступа к

ресурсу пока он используется другим процессом вне зависимости от того хочет ли

первый процесс менять состояние ресурса, поскольку не всегда эти намерения известны.

**7.Два способа решения задачи синхронизации процессов.**

**«Грубая синхронизация»** процессов, не учитывающая их внутреннюю работу и

возможность их обращения к общим ресурсам, обеспечивается программами

комплексного функционирования и планирования работы программного комплекса,

которые вызывают на исполнение по определенной временной диаграмме

функциональные программы.

При этом **«тонкая синхронизация»** работы программ в многозадачном комплексе

требует более безопасного логического разрешения. При решении вопросов тонкой

синхронизации не должно быть заложено никаких предположений о времени

исполнения процессов.

**8.Перечислите задачи синхронизации процессов с кратким их содержанием.**

**1. Взаимное исключение.** Это задача согласования работы n ≥ 2 параллельных процессов при использовании некоторых критических ресурсов ЦВМ, которые не могут использоваться одновременно, а должны использоваться последовательно таким образом, чтобы: одновременно к критическому ресурсу допускалось не более одного процесса, освобождение критического ресурса должно быть произведено процессом (программой), использующим его, за определенное конечное время.

**2. Читатели-писатели.** Программы-читатели могут считывать информацию из зоны памяти (базы данных) асинхронно, по мере возникновения необходимости.Программы – писатели могут записывать информацию в эту же зону памяти, то есть в общем случае обновлять ее также асинхронно с программами-читателями. Если одна задача читает информацию из зоны памяти и будет прервана, переписав её к себе частично, а прервавшая её задача является задачей-писателем, и заменит информацию в зоне памяти, то после завершения обслуживания прерывания и восстановления ОС контекста задачи-читателя она продолжит чтение информации и завершит его прочитав перезаписанную информацию.

**3. Обедающие-философы и т.п. -тупики.** Ситуация в многозадачной среде, при которой несколько [процессов](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) находятся в состоянии ожидания [ресурсов](https://ru.wikipedia.org/w/index.php?title=%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D1%80%D0%B5%D1%81%D1%83%D1%80%D1%81%D1%8B&action=edit&redlink=1), занятых друг другом, и ни один из них не может продолжать свое выполнение.

**4. Производители-потребители (гонки данных).** Возникают тогда, когда несколько процессов работают с разделяемыми данными и конечный результат зависит в конечном итоге от взаимного порядка выполнения ими операций.

**5.Процессы в состоянии ожидания или «потерянные сигналы».** Поток переходит в состояние ожидания, когда для продолжения выполнения ему требуется наступление некоторого события внешнего по отношению к нему. При этом часто возникает ситуация, когда поток ожидает некоторого события, которое уже произошло до того, как поток пришел в состояние ожидания события для его дальнейшей обработки. В результате этой ошибочной ситуации поток никогда из состояния ожидания не выйдет.

**6.«Заброшенные замки».** Ситуация возникает, когда поток захватил критическую секцию или мьютекс и после этого был снят с исполнения по той или иной причине. В результате этот общий ресурс не будет никогда освобожден. Если этот ресурс нужен другому потоку, он будет ожидать его бесконечно.

**9. В чем опасность задачи синхронизации « читатели – писатели» и как защититься от ошибки?**

Опасность заключается в том, что задача-читатель получит не ожидаемую информацию, а ту, которая была получена путем изменения ее задачей-писателем, она будет разнородна.

Метод предотвращения опасности - если программа-читатель захватила ресурс (читает), то должен быть выставлен запрет прерывания до тех пор, пока зона памяти (база данных) не будет

прочитана до конца. Либо писатель должен ждать пока читатели не прочитают БД, а затем

только начать изменение базы данных. Эту работу можно реализовать, используя мьютексы.

Выполнил: Оганисян Т.С.

Группа: 6411

**1.В каких случаях возникает задача обмена информацией между процессами.**

Задача обмена информацией между процессами возникает, тогда, когда данные, сформированные в одной задаче необходимы другим задачам.

**2. Как ОС обеспечивает решение этой задачи. Три варианта решения этой задачи**

3 варианта: через файловую систему, специальные вызовы - средства ОС, разделяемую память задач.

**3.Какой из рассмотренных вариантов обмена информацией между процессами самый быстрый? Какой самый медленный?**

Разделяемая память является наиболее быстрым вариантом взаимодействия процессов.

Через файловую систему взаимодействие получается самое медленное.

**4. Какие процессы и задачи мы называем параллельными и почему не одновременными?**

Два процесса или задачи называются параллельными, если их выполнение может перекрываться во времени и второй процесс начинается до завершения первого.

Нельзя называть одновременными, потому что задачи решаются последовательно, но переключения с задачи на задачу происходят как бы незаметно для задач, и мы говорим о квазиодновременном решении задач.

**5.Что такое критический ресурс ЦВМ?**

Критический ресурс – ресурс, к которому в каждый момент времени может иметь доступ только один процесс, хотя претендентов на доступ много.

**6. Каково основное правило защиты критического ресурса и его модификация?**

Основное правило защиты критических ресурсов – процесс не должен менять состояние ресурса, пока любой другой процесс имеет к нему доступ. Если процессы при доступе к ресурсу могут менять его состояние, то можно переформулировать: процесс не должен получать доступа к ресурсу пока он используется другим процессом вне зависимости от того хочет ли первый процесс менять состояние ресурса, поскольку не всегда эти намерения известны.

**7. Два способа решения задачи синхронизации процессов.**

«Грубая синхронизация» процессов, не учитывающая их внутреннюю работу и возможность их обращения к общим ресурсам, обеспечивается программами комплексного функционирования и планирования работы программного комплекса, которые вызывают на исполнение по определенной временной диаграмме функциональные программы.

«Тонкая синхронизация» работы программ в многозадачном комплексе требует более безопасного логического разрешения. При решении вопросов тонкой синхронизации не должно быть заложено никаких предположений о времени исполнения процессов.

**8. Перечислите задачи синхронизации процессов с кратким их содержанием.**

1.взаимное исключение;

Это задача согласования работы n ≥ 2 параллельных процессов при использовании некоторых критических ресурсов ЦВМ, которые не могут использоваться одновременно, а должны использоваться последовательно таким образом, чтобы: - одновременно к критическому ресурсу допускалось не более одного процесса, - освобождение критического ресурса должно быть произведено процессом (программой), использующим его, за определенное конечное время.

2.читатели-писатели;

«Программы-читатели» могут считывать информацию из зоны памяти (базы данных) асинхронно, по мере возникновения необходимости. «Программы – писатели» могут записывать информацию в эту же зону памяти (базу данных), то есть в общем случае обновлять ее также асинхронно с «программами-читателями».

3.обедающие-философы и т.п. -тупики.

Ситуация в многозадачной среде, при которой несколько процессов находятся в состоянии ожидания ресурсов, занятых друг другом, и ни один из них не может продолжать свое выполнение.

Например, если философы одновременно проголодаются и попытаются взять одновременно, например, левую вилку и не будут её освобождать, ожидая освобождения правой (она занята соседом). Возникает тупиковая ситуация, в результате которой философы умрут с голода. (ресурсное голодание)

4.Производители-потребители (гонки данных).

Возникают тогда, когда несколько процессов работают с разделяемыми данными и конечный результат зависит в конечном итоге от взаимного порядка выполнения ими операций.

5.Процессы в состоянии ожидания или «потерянные сигналы».

Поток переходит в состояние ожидания, когда для продолжения выполнения ему требуется наступление некоторого события внешнего по отношению к нему. Если пребывание потока в состоянии ожидания продолжается слишком долго и превышает допустимое время, то это – ошибочная ситуация. При этом часто возникает ситуация, когда поток ожидает некоторого события, которое уже произошло до того, как поток пришел в состояние ожидания события для его дальнейшей обработки

6.«Заброшенные замки».

Ситуация возникает, когда поток захватил критическую секцию или мьютекс и после этого был снят с исполнения по той или иной причине. В результате этот общий ресурс не будет никогда освобожден. Если этот ресурс нужен другому потоку, он будет ожидать его бесконечно.

**9. В чем опасность задачи синхронизации «читатели – писатели» и как защититься от ошибки?**

Если «задача-читатель» считывает информацию в отрезке времени t0-t2, и в этот промежуток времени, например t1 начнет запись «задача-писатель», то на выходе «задача-читатель» получит разнородную информацию, относящуюся к разным моментам времени t0-t1 и t1-t2 соответственно.

Метод защиты: если программа-читатель захватила ресурс (читает), то должен быть выставлен запрет прерывания до тех пор, пока зона памяти (база данных) не будет прочитана до конца. Либо писатель должен ждать пока читатели не прочитают БД, а затем только начать изменение базы данных. Для этого можно воспользоваться, например, мьютексами.

Выполнил: Шабакаев А.Р.

Группа: 6411

**1. В каких случаях возникает задача обмена информацией между процессами?**

Задача *обмена информацией* между процессами возникает при работе в централизованной (нераспределенной) системе ПО, когда обмен данными между процессами должен проводиться через общую память. При этом надо преодолеть изоляцию памяти процессов, к которой мы стремились с точки зрения безопасности работы ПО.

**2. Как ОС обеспечивает решение этой задачи. Три варианта решения этой задачи**

ОС обеспечивает преодоление изоляции памяти работы процессов, и с точки зрения безопасности работы ПО, ОС обеспечивает изоляцию «всех от всех»: пользователей друг от друга, пользователей от программ, программ - процессов друг от друга, программ от аппаратуры ЦВМ.

Существуют 3 варианта решения задачи обмена информацией между процессами:

через файловую систему, при помощи использования средств, через разделяемую память.

**3. Какой из рассмотренных вариантов обмена информацией между процессами самый быстрый?** **Какой самый медленный?**

Самым быстрым вариантом обмена информацией является взаимодействие процессов на основе разделимой памяти.

Такое взаимодействие не требует каких-либо накладных расходов, процессы обмениваются информацией со скоростью обращения к памяти.

Самым медленным является взаимодействие процессов через файловую систему.

**4. Какие процессы и задачи мы называем параллельными и почему не одновременными?**

Два процесса или задачи, или программы называются параллельными, если их выполнение *может перекрываться во времени*, и второй процесс начинается до завершения первого.

В случае многозадачной работы ПО на однопроцессорной ЦВМ задачи решаются последовательно, но переключения с задачи на задачу происходят как бы незаметно для задач, и мы говорим о квазиодновременном решении задач.

**5. Что такое критический ресурс ЦВМ?**

Критический ресурс – ресурс к которому в каждый момент времени может иметь доступ только один процесс, хотя претендентов на доступ много.

**6. Каково основное правило защиты критического ресурса и его модификация?**

Основное правило защиты критических ресурсов – процесс не должен менять состояние ресурса, пока любой другой процесс имеет к нему доступ.

На практике: *процесс не должен получать доступа к ресурсу пока он используется другим процессом* вне зависимости от того хочет ли первый процесс менять состояние ресурса, поскольку не всегда эти намерения известны

**7. Два способа** **решения задачи синхронизации процессов.**

«Грубая синхронизация»(временная) процессов, не учитывающая их внутреннюю работу и возможность их обращения к общим ресурсам, обеспечивается программами комплексного функционирования и планирования работы программного комплекса, которые вызывают на исполнение по определенной временной диаграмме функциональные программы.

«Тонкая синхронизация»(логическая) работы программ в многозадачном комплексе требует более безопасного логического разрешения. При решении вопросов тонкой синхронизации не должно быть заложено никаких предположений о времени исполнения процессов. Выполняется путем отслеживания в самом ПО выполнения некоторых логических условий, например, фактического начала или завершения смежных взаимодействующих процессов.

**8. Перечислите задачи синхронизации процессов с кратким их содержанием.**

**8.1. Взаимное исключение**

Это задача согласования работы n ≥ 2 параллельных процессов.

Правило использования ресурсов ЦВМ:

- одновременно к критическому ресурсу допускалось не более одного процесса,

- освобождение критического ресурса должно быть произведено процессом (программой), использующим его, за определенное конечное время.

*Решение: использовать двоичные логические переменные – логические признаки или флаги, которые управляются от процессов и читаются ими.*

В операционных системах данный механизм проработан и принимает стандартную форму путем использования семафоров и мьютексов. Над мьютексом определены две операции – два системных вызова ОС: **Signal** (Set) и Wait(названия обобщенные). Signal всегда устанавливает мьютекс в 1. Мьютекс находится в сигнальном состоянии, если он никому не принадлежит.

Пришедший процесс, содержащий опрос значения Мьютекса и операцию *Wait*, анализирующий состояние мьютекса, обнуляет значение мьютекса, и получает разрешение на исполнение данных. А когда освобождает ресурс выставляет значение мьютекса 1. Следующий за ним процесс при значении мьютекса 0, ожидает освобождение ресурса, и захватывает его, когда мьютекс путем передачи системного вызова переводится процессором в состояние Signal.

Программист должен обеспечивать правильную работу мьютекса и согласованность операций *Signal* и*Wai*t, в противном случае это может привести к тупикам.

**8.2. Читатели-писатели**

Программы-читатели могут считывать информацию из зоны памяти (базы данных) асинхронно.

Программы – писатели могут записывать информацию в эту же зону памяти (базу данных), то есть в общем случае обновлять ее также асинхронно с программами-читателями.

Проблема: читателя во время работы с критическим ресурсом, может прервать писатель для работы с данными, то есть их изменением. После завершения работы писателя, читатель может возобновить работу, но на выходе у него будет пусть и полноценная информация, но несогласованная, ее части будут разнородны по времени. Это приводит к семантической ошибке.

*Решение: если читатель захватывает ресурс, то должен быть выставлен запрет прерывания до тех пор, пока зона памяти (база данных) не будет прочитана до конца. Либо писатель должен ждать пока читатели не прочитают зону памяти (базу данных), а затем только начать изменение базы данных. Эту работу можно реализовать, используя мьютексы.*

**8.3. Обедающие-философы и т.п. -тупики**

У нас есть процессы-«философы». Для его работы нужны 2 вилки.

Вилки в данной ситуации выступают в качестве пересекающихся и разделяемых ресурсов.

Проблема: философ не освободит одну вилку, пока не получит вторую вилку. *Решение:*

*- не брать одну вилку до тех пор, пока не убедился, что обе вилки (оба ресурса) доступны;*

*- не ожидать освобождения ресурса более определенного времени, т.е., если второго ресурса нет в течение определенного времени, то освободить и первый.*

**8.4. Производители-потребители (гонки данных).**

Возникают тогда, когда несколько процессов работают с разделяемыми данными и конечный результат зависит в конечном итоге от взаимного порядка выполнения ими операций.

**8.5. Процессы в состоянии ожидания или «потерянные сигналы».**

Поток переходит в состояние ожидания, когда для продолжения выполнения ему требуется наступление некоторого события внешнего по отношению к нему. Если пребывание потока в состоянии ожидания продолжается слишком долго и превышает допустимое время, то это – ошибочная ситуация, при этом часто возникает ситуация, когда поток ожидает некоторого события, которое уже произошло до того, как поток пришел в состояние ожидания события для его дальнейшей обработки. Эта ошибка происходит из-за неправильной синхронизации. В результате этой ошибочной ситуации поток никогда из состояния ожидания не выйдет.

**8.6. «Заброшенные замки».**

Ситуация возникает, когда поток захватил критическую секцию или мьютекс и после этого был снят с исполнения по той или иной причине. В результате этот общий ресурс не будет никогда освобожден. Если этот ресурс нужен другому потоку, он будет ожидать его бесконечно. Вывод – при снятии потока с исполнения, что может быть необходимым в некоторых ситуациях, необходимо предварительно освободить захваченные ресурсы.

**9. В чем опасность задачи синхронизации «читатели – писатели» и как защититься от ошибки?**

Опасность данной задачи синхронизации заключается в том, что задача-читатель получит информацию, состоящую из разнородных частей, относящихся к разным моментам времени. Возникает это из-за того, что читатель во время своей работы может быть прерван писателем, и после выполнения работы писателя, читатель считывает уже измененные данные.

Как защититься от ошибки: если читатель захватывает ресурс, то должен быть выставлен запрет прерывания до тех пор, пока зона памяти (база данных) не будет прочитана до конца. Либо писатель должен ждать пока читатели не прочитают зону памяти (базу данных), а затем только начать изменение базы данных. Эту работу можно реализовать, используя мьютексы.

Выполнил: Терентьев Е.А

Группа: 6411

**1.В каких случаях возникает задача обмена информацией между процессами.**

Задача обмена информацией между процессами возникает, тогда, когда данные, сформированные в одной задаче необходимы другим задачам. При этом важна синхронизация работы задач, чтобы эти связи реализовывались своевременно. Обмен данными между должен проводиться через общую память.

**2. Как ОС обеспечивает решение этой задачи. Три варианта решения этой задачи**

Решение задачи обмена информацией между процессами обеспечивается тремя способами: через файловую систему, специальные вызовы - средства ОС, разделяемую память задач.

**3.Какой из рассмотренных вариантов обмена информацией между процессами самый быстрый? Какой самый медленный?**

При взаимодействии процессов через файловую систему взаимодействие получается медленное. Разделяемая память является наиболее быстрым вариантом взаимодействия процессов. Для операционной системы этот способ является наиболее простым.

**4. Какие процессы и задачи мы называем параллельными и почему не одновременными?**

Два процесса или задачи, или программы называются параллельными, если их выполнение может перекрываться во времени и второй процесс начинается до завершения первого. В случае многозадачной работы ПО на однопроцессорной ЦВМ задачи решаются последовательно, но переключения с задачи на задачу происходят как бы незаметно для задач, и мы говорим о квази-одновременном решении задач.

**5.Что такое критический ресурс ЦВМ?**

Критический ресурс – ресурс к которому в каждый момент времени может иметь доступ только один процесс, хотя претендентов на доступ много.

**6. Каково основное правило защиты критического ресурса и его модификация?**

Основное правило защиты критических ресурсов – процесс не должен менять состояние ресурса, пока любой другой процесс имеет к нему доступ. Другими словами, процесс не должен получать доступа к ресурсу пока он используется другим процессом вне зависимости от того хочет ли первый процесс менять состояние ресурса, поскольку не всегда эти намерения известны.

**7. Два способа решения задачи синхронизации процессов.**

«Грубая синхронизация» процессов, не учитывающая их внутреннюю работу и возможность их обращения к общим ресурсам, обеспечивается программами комплексного функционирования и планирования работы программного комплекса, которые вызывают на исполнение по определенной временной диаграмме функциональные программы.

«Тонкая синхронизация» работы программ в многозадачном комплексе требует более безопасного логического разрешения. При решении вопросов тонкой синхронизации не должно быть заложено никаких предположений о времени исполнения процессов. Выполняется путем отслеживания в самом ПО выполнения некоторых логических условий, например, фактического начала или завершения смежных взаимодействующих процессов.

**8. Перечислите задачи синхронизации процессов с кратким их содержанием.**

Можно выделить несколько различных задач синхронизации, которые обеспечивают защиту общих ресурсов ЦВМ Эти задачи соответствуют реальным и типовым ситуациям в “жизни программ”:

1. Взаимное исключение.

Это задача согласования работы n ≥ 2 параллельных процессов при использовании некоторых критических ресурсов ЦВМ, которые не могут использоваться одновременно, а должны использоваться последовательно таким образом, чтобы:

- одновременно к критическому ресурсу допускалось не более одного процесса,

- освобождение критического ресурса должно быть произведено процессом (программой), использующим его, за определенное конечное время.

2. Читатели-писатели.

Программы-читатели могут считывать информацию из зоны памяти (базы данных) асинхронно.

Программы – писатели могут записывать информацию в эту же зону памяти (базу данных), то есть в общем случае обновлять ее также асинхронно с программами-читателями.

Если одна задача читает информацию из зоны памяти (из базы данных) и будет прервана, переписав её к себе частично, а прервавшая её задача является задачей-писателем, и заменит информацию в зоне памяти, то после завершения обслуживания прерывания и восстановления ОС контекста задачи-читателя она продолжит чтение информации и завершит его прочитав перезаписанную информацию.

3. Обедающие-философы и т.п. -тупики.

Ситуация в многозадачной среде, при которой несколько [процессов](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) находятся в состоянии ожидания [ресурсов](https://ru.wikipedia.org/w/index.php?title=%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D1%80%D0%B5%D1%81%D1%83%D1%80%D1%81%D1%8B&action=edit&redlink=1), занятых друг другом, и ни один из них не может продолжать свое выполнение.

4. Производители-потребители (гонки данных). Возникают тогда, когда несколько процессов работают с разделяемыми данными и конечный результат зависит в конечном итоге от взаимного порядка выполнения ими операций.

5.Процессы в состоянии ожидания или «потерянные сигналы». Поток переходит в состояние ожидания, когда для продолжения выполнения ему требуется наступление некоторого события внешнего по отношению к нему. При этом часто возникает ситуация, когда поток ожидает некоторого события, которое уже произошло до того, как поток пришел в состояние ожидания события для его дальнейшей обработки. В результате этой ошибочной ситуации поток никогда из состояния ожидания не выйдет.

6.«Заброшенные замки». Ситуация возникает, когда поток захватил критическую секцию или мьютекс и после этого был снят с исполнения по той или иной причине. В результате этот общий ресурс не будет никогда освобожден. Если этот ресурс нужен другому потоку, он будет ожидать его бесконечно.

**9. В чем опасность задачи синхронизации «читатели – писатели» и как защититься от ошибки?**

Если одна задача читает информацию из зоны памяти (из базы данных) и будет прервана, переписав её к себе частично, а прервавшая её задача является задачей-писателем, и заменит информацию в зоне памяти, то после завершения обслуживания прерывания и восстановления ОС контекста задачи-читателя она продолжит чтение информации и завершит его прочитав перезаписанную информацию.

Метод предотвращения уязвимости известен и прост: если программа-читатель захватила ресурс (читает), то должен быть выставлен запрет прерывания до тех пор, пока зона памяти (база данных) не будет прочитана до конца. Либо писатель должен ждать пока читатели не прочитают БД, а затем только начать изменение базы данных. Эту работу можно реализовать, используя мьютексы.

Выполнил: Щекочихин Д.А.

Группа: 6411

**1.В каких случаях возникает задача обмена информацией между процессами.**

Связь между задачами возникает, чтобы данные, сформированные в одной задаче были доступны другим задачам. При этом важно, чтоб эти связи реализовывались своевременно.

**2. Как ОС обеспечивает решение этой задачи. Три варианта решения этой задачи**

Решение этой задачи обеспечивается тремя способами: через файловую систему, специальные вызовы - средства ОС, разделяемую память задач.

**3.Какой из рассмотренных вариантов обмена информацией между процессами самый быстрый? Какой самый медленный?**

Разделяемая память является наиболее быстрым вариантом взаимодействия процессов.

При взаимодействии процессов через файловую систему взаимодействие получается медленное.

**4. Какие процессы и задачи мы называем параллельными и почему не одновременными?**

Два процесса или задачи, или программы называются параллельными, если их выполнение может перекрываться во времени и второй процесс начинается до завершения первого. В случае многозадачной работы ПО задачи решаются последовательно, и за счет быстрого переключения с задачи на задачу создается иллюзия параллелизма и это происходит незаметно для задач.

**5.Что такое критический ресурс ЦВМ?**

Критический ресурс – ресурс к которому в каждый момент времени может иметь доступ только один процесс, хотя претендентов на доступ много.

**6. Каково основное правило защиты критического ресурса и его модификация?**

Основное правило защиты критических ресурсов – процесс не должен менять состояние ресурса, пока любой другой процесс имеет к нему доступ. Это правило преформулируется следующим образом: процесс не должен получать доступа к ресурсу пока он используется другим процессом вне зависимости от того хочет ли первый процесс менять состояние ресурса, поскольку не всегда эти намерения известны.

**7. Два способа решения задачи синхронизации процессов.**

«Грубая синхронизация» процессов, не учитывающая их внутреннюю работу и возможность их обращения к общим ресурсам, обеспечивается программами комплексного функционирования и планирования работы программного комплекса, которые вызывают на исполнение по определенной временной диаграмме функциональные программы.

«Тонкая синхронизация» работы программ в многозадачном комплексе требует более безопасного логического разрешения. При решении вопросов тонкой синхронизации не должно быть заложено никаких предположений о времени исполнения процессов. Выполняется путем отслеживания в самом ПО выполнения некоторых логических условий, например, фактического начала или завершения смежных взаимодействующих процессов.

**8. Перечислите задачи синхронизации процессов с кратким их содержанием.**

Можно выделить несколько различных задач синхронизации, которые обеспечивают защиту общих ресурсов ЦВМ. Эти задачи соответствуют реальным и типовым ситуациям в “жизни программ”:

1. Взаимное исключение.

Это задача согласования работы n ≥ 2 параллельных процессов при использовании некоторых критических ресурсов ЦВМ, которые не могут использоваться одновременно, а должны использоваться последовательно таким образом, чтобы:

- одновременно к критическому ресурсу допускалось не более одного процесса,

- освобождение критического ресурса должно быть произведено процессом (программой), использующим его, за определенное конечное время.

2. Читатели-писатели.

Программы-читатели могут считывать информацию из зоны памяти (базы данных) асинхронно.

Программы – писатели могут записывать информацию в эту же зону памяти (базу данных), то есть в общем случае обновлять ее также асинхронно с программами-читателями.

Если одна задача читает информацию из зоны памяти (из базы данных) и будет прервана, переписав её к себе частично, а прервавшая её задача является задачей-писателем, и заменит информацию в зоне памяти, то после завершения обслуживания прерывания и восстановления ОС контекста задачи-читателя она продолжит чтение информации и завершит его прочитав перезаписанную информацию. В результате задача-читатель получит полный по номенклатуре объем информации, но состоящий из двух в общем случае разнородных частей, относящихся к разным моментам времени.

3. Обедающие-философы и т.п. -тупики. Ситуация в многозадачной среде, при которой несколько процессов находятся в состоянии ожидания ресурсов, занятых друг другом, и ни один из них не может продолжать свое выполнение.

4. Производители-потребители (гонки данных). Возникают тогда, когда несколько процессов работают с разделяемыми данными и конечный результат зависит в конечном итоге от взаимного порядка выполнения ими операций.

5.Процессы в состоянии ожидания или «потерянные сигналы». Поток переходит в состояние ожидания, когда для продолжения выполнения ему требуется наступление некоторого события внешнего по отношению к нему. При этом часто возникает ситуация, когда поток ожидает некоторого события, которое уже произошло до того, как поток пришел в состояние ожидания события для его дальнейшей обработки. В результате этой ошибочной ситуации поток никогда из состояния ожидания не выйдет.

6.«Заброшенные замки». Ситуация возникает, когда поток захватил критическую секцию или мьютекс и после этого был снят с исполнения по той или иной причине. В результате этот общий ресурс не будет никогда освобожден. Если этот ресурс нужен другому потоку, он будет ожидать его бесконечно. Вывод – при снятии потока с исполнения, что может быть необходимым в некоторых ситуациях, необходимо предварительно освободить захваченные ресурсы.

**9. В чем опасность задачи синхронизации «читатели – писатели» и как защититься от ошибки?**

Если одна задача читает информацию из зоны памяти (из базы данных) и будет прервана, переписав её к себе частично, а прервавшая её задача является задачей-писателем, и заменит информацию в зоне памяти, то после завершения обслуживания прерывания и восстановления ОС контекста задачи-читателя она продолжит чтение информации и завершит его.

Метод предотвращения ошибки известен и прост: если программа-читатель захватила ресурс (читает), то должен быть выставлен запрет прерывания до тех пор, пока зона памяти (база данных) не будет прочитана до конца. Либо писатель должен ждать пока читатели не прочитают БД, а затем только начать изменение базы данных. Эту работу можно реализовать, используя мьютексы.