**01.12.2020 ИС2К Операционные системы**

**«Основы операционных систем» К.А. Коньков** <https://yadi.sk/i/SojomTcdoQ2v_g>

 **Тема:** Лекция 4: **Кооперация процессов и основные аспекты ее логической**

 **организации**

 (Учебник «Основы операционных систем» К.А.Коньков стр. 40-48)

**Ответить на вопросы: (кратко 3-4 предложения максимум)**

1. **Причины совместной деятельности процессов**
2. **Категории средств обмена информацией**
3. **Способы адресации**
4. **Особенности передачи информации с помощью линий связи**

**1.Поток ввода/вывода**

**2.Сообщения**

1. **Надежность средств связи на кого возложена**
2. **Процесс и нити исполнения**

**Ответы присылать** kuzn117@yandex.ru **можно в ВК -**  **id480169637**

 **Пишите тему письма и файл именуйте: Фамилия-ОС-0112**

**Для тех у кого проблемы с Яндекс диском сегодняшнюю тему**

 **(**Лекция 4:

Кооперация процессов и основные аспекты ее логической организации**)**

 **выкладываю ниже**

**4.Лекция: Кооперация процессов и основные аспекты ее логической организа-
ции**

Одной из функций операционной системы является обеспечение санкционированного взаимодействия
процессов. Лекция посвящена основам логической организации такого взаимодействия. Рассматривается
расширение понятия процесс - нить исполнения (thread).

Взаимодействие процессов в вычислительной системе напоминает жизнь в коммунальной квартире. По-
стоянное ожидание в очереди к местам общего пользования (процессору) и ежедневная борьба за ресур-
сы (кто опять занял все конфорки на плите?). Для нормального функционирования процессов операци-
онная система старается максимально обособить их друг от друга. Каждый процесс имеет собственное
адресное пространство (каждая семья должна жить в отдельной комнате), нарушение которого, как пра-
вило, приводит к аварийной остановке процесса (вызов милиции). Каждому процессу по возможности
предоставляются свои дополнительные ресурсы (каждая семья предпочитает иметь собственный холо-
дильник). Тем не менее для решения некоторых задач (приготовление праздничного стола на всю квар-
тиру) процессы могут объединять свои усилия. В настоящей лекции описываются причины взаимодейст-
вия процессов, способы их взаимодействия и возникающие при этом проблемы (попробуйте отремонти-
ровать общую квартиру так, чтобы жильцы не перессорились друг с другом).

**Взаимодействующие процессы**

Для достижения поставленной цели различные процессы (возможно, даже принадлежащие разным поль-
зователям) могут исполняться псевдопараллельно на одной вычислительной системе или параллельно на
разных вычислительных системах, взаимодействуя между собой.

Для чего процессам нужно заниматься совместной деятельностью? Какие существуют причины для их
кооперации?

• Повышение скорости работы. Пока один процесс ожидает наступления некоторого события (на-
пример, окончания операции ввода-вывода), другие могут заниматься полезной работой, направ-
ленной на решение общей задачи. В многопроцессорных вычислительных системах программа
разбивается на отдельные кусочки, каждый из которых будет исполняться на своем процессоре.

• Совместное использование данных. Различные процессы могут, к примеру, работать с одной и той
же динамической базой данных или с разделяемым файлом, совместно изменяя их содержимое.

• Модульная конструкция какой-либо системы. Типичным примером может служить микроядерный
способ построения операционной системы, когда различные ее части представляют собой отдель-
ные процессы, взаимодействующие путем передачи сообщений через микроядро.

• Наконец, это может быть необходимо просто для удобства работы пользователя, желающего, на-
пример, редактировать и отлаживать программу одновременно. В этой ситуации процессы редак-
тора и отладчика должны уметь взаимодействовать друг с другом.

Процессы не могут взаимодействовать, не общаясь, то есть не обмениваясь информацией. "Общение"
процессов обычно приводит к изменению их поведения в зависимости от полученной информации. Если
деятельность процессов остается неизменной при любой принятой ими информации, то это означает, что
они на самом деле в " общении" не нуждаются. Процессы, которые влияют на поведение друг друга путем

**Основы операционных систем**

**41**

обмена информацией, принято называть **кооперативными** или взаимодействующими процессами, в от-
личие от независимых процессов, не оказывающих друг на друга никакого воздействия.

Различные процессы в вычислительной системе изначально представляют собой обособленные сущно-
сти. Работа одного процесса не должна приводить к нарушению работы другого процесса. Для этого, в
частности, разделены их адресные пространства и системные ресурсы, и для обеспечения корректного
взаимодействия процессов требуются специальные средства и действия операционной системы. Нельзя
просто поместить значение, вычисленное в одном процессе, в область памяти, соответствующую пере-
менной в другом процессе, не предприняв каких-либо дополнительных усилий. Давайте рассмотрим ос-
новные аспекты организации совместной работы процессов.

**Категории средств обмена информацией**

Процессы могут взаимодействовать друг с другом, только обмениваясь информацией. По объему переда-
ваемой информации и степени возможного воздействия на поведение другого процесса все средства та-
кого обмена можно разделить на три категории.

• Сигнальные. Передается минимальное количество информации - один бит, "да" или "нет". Ис-
пользуются, как правило, для извещения процесса о наступлении какого-либо события. Степень
воздействия на поведение процесса, получившего информацию, минимальна. Все зависит от того,
знает ли он, что означает полученный сигнал, надо ли на него реагировать и каким образом. Не-
правильная реакция на сигнал или его игнорирование могут привести к трагическим последстви-
ям. Вспомним профессора Плейшнера из кинофильма "Семнадцать мгновений весны". Сигнал
тревоги - цветочный горшок на подоконнике - был ему передан, но профессор проигнорировал
его. И к чему это привело?

• Канальные. "Общение" процессов происходит через линии связи, предоставленные операционной
системой, и напоминает общение людей по телефону, с помощью записок, писем или объявлений.
Объем передаваемой информации в единицу времени ограничен пропускной способностью линий
связи. С увеличением количества информации возрастает и возможность влияния на поведение
другого процесса.

• Разделяемая память. Два или более процессов могут совместно использовать некоторую область
адресного пространства. Созданием разделяемой памяти занимается операционная система (если,
конечно, ее об этом попросят). "Общение" процессов напоминает совместное проживание студен-
тов в одной комнате общежития. Возможность обмена информацией максимальна, как, впрочем, и
влияние на поведение другого процесса, но требует повышенной осторожности (если вы перело-
жили на другое место вещи вашего соседа по комнате, а часть из них еще и выбросили). Исполь-
зование разделяемой памяти для передачи/получения информации осуществляется с помощью
средств обычных языков программирования, в то время как сигнальным и канальным средствам
коммуникации для этого необходимы специальные системные вызовы. Разделяемая память пред-
ставляет собой наиболее быстрый способ взаимодействия процессов в одной вычислительной сис-
теме.

**Логическая организация механизма передачи информации**

При рассмотрении любого из средств коммуникации нас будет интересовать не их физическая реализа-
ция (общая шина данных, прерывания, аппаратно разделяемая память и т. д.), а логическая, определяю-
щая в конечном счете механизм их использования. Некоторые важные аспекты логической реализации
являются общими для всех категорий средств связи, некоторые относятся к отдельным категориям. Да-
вайте кратко охарактеризуем основные вопросы, требующие разъяснения при изучении того или иного
способа обмена информацией.

**Как устанавливается связь?**

Могу ли я использовать средство связи непосредственно для обмена информацией сразу после создания
процесса или первоначально необходимо предпринять определенные действия для инициализации обме-
на? Например, для использования общей памяти различными процессами потребуется специальное об-
ращение к операционной системе, которая выделит необходимую область адресного пространства. Но

**Основы операционных систем**

**42**

для передачи сигнала от одного процесса к другому никакая инициализация не нужна. В то же время пе-
редача информации по линиям связи может потребовать первоначального резервирования такой линии
для процессов, желающих обменяться информацией.

К этому же вопросу тесно примыкает вопрос о способе адресации при использовании средства связи. Ес-
ли я передаю некоторую информацию, я должен указать, куда я ее передаю. Если я желаю получить не-
которую информацию, то мне нужно знать, откуда я могу ее получить.

Различают два способа адресации: прямую и непрямую. В случае прямой адресации взаимодействующие
процессы непосредственно общаются друг с другом, при каждой операции обмена данными явно указы-
вая имя или номер процесса, которому информация предназначена или от которого она должна быть по-
лучена. Если и процесс, от которого данные исходят, и процесс, принимающий данные, указывают имена
своих партнеров по взаимодействию, то такая схема адресации называется симметричной прямой адре-
сацией. **Ни один другой процесс не может вмешаться в процедуру симметричного прямого общения
двух процессов**, **перехватить посланные или подменить ожидаемые данные**. Если только один из
взаимодействующих процессов, например передающий, указывает имя своего партнера по кооперации, а
второй процесс в качестве возможного партнера рассматривает любой процесс в системе, например ожи-
дает получения информации от произвольного источника, то такая схема адресации называется асиммет-
ричной прямой адресацией.

При непрямой адресации данные помещаются передающим процессом в некоторый промежуточный
объект для хранения данных, имеющий свой адрес, откуда они могут быть затем изъяты каким-либо дру-
гим процессом. Примером такого объекта может служить обычная доска объявлений или рекламная газе-
та. При этом передающий процесс не знает, как именно идентифицируется процесс, который получит
информацию, а принимающий процесс не имеет представления об идентификаторе процесса, от которого
он должен ее получить.

При использовании прямой адресации связь между процессами в классической операционной системе
устанавливается автоматически, без дополнительных инициализирующих действий. Единственное, что
нужно для использования средства связи, - это знать, как идентифицируются процессы, участвующие в
обмене данными.

При использовании непрямой адресации инициализация средства связи может и не требоваться. Инфор-
мация, которой должен обладать процесс для взаимодействия с другими процессами, - это некий иден-
тификатор промежуточного объекта для хранения данных, если он, конечно, не является единственным и
неповторимым в вычислительной системе для всех процессов.

**Информационная валентность процессов и средств связи**

Следующий важный вопрос - это вопрос об информационной валентности связи. Слово "валентность"
здесь использовано по аналогии с химией. Сколько процессов может быть одновременно ассоциировано
с конкретным средством связи? Сколько таких средств связи может быть задействовано между двумя
процессами?

Понятно, что при прямой адресации только одно фиксированное средство связи может быть задейство-
вано для обмена данными между двумя процессами, и только эти два процесса могут быть ассоциирова-
ны с ним. При непрямой адресации может существовать более двух процессов, использующих один и тот
же объект для данных, и более одного объекта может быть использовано двумя процессами.

К этой же группе вопросов следует отнести и вопрос о направленности связи. Является ли связь однона-
правленной или двунаправленной? Под однонаправленной связью мы будем понимать связь, при которой
каждый процесс, ассоциированный с ней, может использовать средство связи либо только для приема
информации, либо только для ее передачи. При двунаправленной связи каждый процесс, участвующий в
общении, может использовать связь и для приема, и для передачи данных. В коммуникационных систе-
мах принято называть однонаправленную связь симплексной, двунаправленную связь с поочередной пе-
редачей информации в разных направлениях - полудуплексной, а двунаправленную связь с возможно-

**Основы операционных систем**

**43**

стью одновременной передачи информации в разных направлениях - дуплексной. Прямая и непрямая
адресация не имеет непосредственного отношения к направленности связи.

**Особенности передачи информации с помощью линий связи**

Как уже говорилось выше, передача информации между процессами посредством линий связи является
достаточно безопасной по сравнению с использованием разделяемой памяти и более информативной по
сравнению с сигнальными средствами коммуникации. Кроме того, разделяемая память не может быть
использована для связи процессов, функционирующих на различных вычислительных системах. Воз-
можно, именно поэтому каналы связи из средств коммуникации процессов получили наибольшее рас-
пространение. Коснемся некоторых вопросов, связанных с логической реализацией канальных средств
коммуникации.

**Буферизация**

Может ли линия связи сохранять информацию, переданную одним процессом, до ее получения другим
процессом или помещения в промежуточный объект? Каков объем этой информации? Иными словами,
речь идет о том, обладает ли канал связи **буфером** и каков объем этого буфера. Здесь можно выделить
три принципиальных варианта.

1. Буфер нулевой емкости или отсутствует. Никакая информация не может сохраняться на линии
связи. В этом случае процесс, посылающий информацию, должен ожидать, пока процесс, прини-
мающий информацию, не соблаговолит ее получить, прежде чем заниматься своими дальнейшими
делами (в реальности этот случай никогда не реализуется).

2. Буфер ограниченной емкости. Размер буфера равен n, то есть линия связи не может хранить до
момента получения более чем n единиц информации. Если в момент передачи данных в буфере
хватает места, то передающий процесс не должен ничего ожидать. Информация просто копирует-
ся в буфер. Если же в момент передачи данных буфер заполнен или места недостаточно, то необ-
ходимо задержать работу процесса отправителя до появления в буфере свободного пространства.

3. Буфер неограниченной емкости. Теоретически это возможно, но практически вряд ли реализуемо.
Процесс, посылающий информацию, никогда не ждет окончания ее передачи и приема другим
процессом.

При использовании канального средства связи с непрямой адресацией под емкостью буфера обычно по-
нимается количество информации, которое может быть помещено в промежуточный объект для хранения
данных.

**Поток ввода/вывода и сообщения**

Существует две модели передачи данных по каналам связи - поток ввода-вывода и сообщения. При пе-
редаче данных с помощью потоковой модели операции передачи/приема информации вообще не интере-
суются содержимым данных. Процесс, прочитавший 100 байт из линии связи, не знает и не может знать,
были ли они переданы одновременно, т. е. одним куском или порциями по 20 байт, пришли они от одно-
го процесса или от разных. Данные представляют собой простой поток байтов, без какой-либо их интер-
претации со стороны системы. Примерами потоковых каналов связи могут служить pipe и FIFO, описан-
ные ниже.

Одним из наиболее простых способов передачи информации между процессами по линиям связи являет-
ся передача данных через pipe (канал, трубу или, как его еще называют в литературе, конвейер). Пред-
ставим себе, что у нас есть некоторая труба в вычислительной системе, в один из концов которой процес-
сы могут "сливать" информацию, а из другого конца принимать полученный поток. Такой способ реали-
зует потоковую модель ввода/вывода. Информацией о расположении трубы в операционной системе об-
ладает только процесс, создавший ее. Этой информацией он может поделиться исключительно со своими
наследниками - процессами-детьми и их потомками. Поэтому использовать pipe для связи между собой
могут только родственные процессы, имеющие общего предка, создавшего данный канал связи.

Если разрешить процессу, создавшему трубу, сообщать о ее местонахождении в системе другим процес-
сам, сделав вход и выход трубы каким-либо образом видимыми для всех остальных, например, зарегист-

**Основы операционных систем**

**44**

рировав ее в операционной системе под определенным именем, мы получим объект, который принято
называть FIFO или именованный pipe. Именованный pipe может использоваться для организации связи
между любыми процессами в системе.

В модели сообщений процессы налагают на передаваемые данные некоторую структуру. Весь поток ин-
формации они разделяют на отдельные сообщения, вводя между данными, по крайней мере, границы со-
общений. Примером границ сообщений являются точки между предложениями в сплошном тексте или
границы абзаца. Кроме того, к передаваемой информации могут быть присоединены указания на то, кем
конкретное сообщение было послано и для кого оно предназначено. Примером указания отправителя мо-
гут служить подписи под эпиграфами в книге. Все сообщения могут иметь одинаковый фиксированный
размер или могут быть переменной длины. В вычислительных системах используются разнообразные
средства связи для передачи сообщений: очереди сообщений, sockets (гнезда) и т. д. Часть из них мы рас-
смотрим подробнее в дальнейшем, в частности очереди сообщений будут рассмотрены в лекции 6, а
гнезда (иногда их еще называют по транслитерации английского названия - сокеты) в лекции 14.

И потоковые линии связи, и каналы сообщений всегда имеют буфер конечной длины. Когда мы будем
говорить о емкости буфера для потоков данных, мы будем измерять ее в байтах. Когда мы будем гово-
рить о емкости буфера для сообщений, мы будем измерять ее в сообщениях.

**Надежность средств связи**

Одним из существенных вопросов при рассмотрении всех категорий средств связи является вопрос об их
надежности. Мы все знаем, как бывает тяжело расслышать собеседника по вечно трещащему телефону
или разобрать, о чем сообщается в телеграмме: "Прибду пыездом в вонедельник 33 июня в 25.34. Пама".

Мы будем называть способ коммуникации надежным, если при обмене данными выполняются четыре
условия.

1. Не происходит потери информации.

2. Не происходит повреждения информации.

3. Не появляется лишней информации.

4. Не нарушается порядок данных в процессе обмена.

Очевидно, что передача данных через разделяемую память является надежным способом связи. То, что
мы сохранили в разделяемой памяти, будет считано другими процессами в первозданном виде, если, ко-
нечно, не произойдет сбоя в питании компьютера. Для других средств коммуникации, как видно из при-
веденных выше примеров, это не всегда верно.

Каким образом в вычислительных системах пытаются бороться с ненадежностью коммуникаций? Давай-
те рассмотрим возможные варианты на примере обмена данными через линию связи с помощью сообще-
ний. Для обнаружения повреждения информации будем снабжать каждое передаваемое сообщение неко-
торой контрольной суммой, вычисленной по посланной информации. При приеме сообщения контроль-
ную сумму будем вычислять заново и проверять ее соответствие пришедшему значению. Если данные не
повреждены (контрольные суммы совпадают), то подтвердим правильность их получения. Если данные
повреждены (контрольные суммы не совпадают), то сделаем вид, что сообщение к нам не поступило.
Вместо контрольной суммы можно использовать специальное кодирование передаваемых данных с по-
мощью кодов, исправляющих ошибки. Такое кодирование позволяет при числе искажений информации,
не превышающем некоторого значения, восстановить первоначальные неискаженные данные. Если по
прошествии некоторого интервала времени подтверждение правильности полученной информации не
придет на передающий конец линии связи, будем считать информацию утерянной и пошлем ее повторно.
Для того чтобы избежать двойного получения одной и той же информации, на приемном конце линии
связи должен осуществляться соответствующий контроль. Для гарантии правильного порядка получения
сообщений будем их нумеровать. При приеме сообщения с номером, не соответствующим ожидаемому,
поступаем с ним как с утерянным и ждем сообщения с правильным номером.

Подобные действия могут быть возложены:

**Основы операционных систем**

**45**

• на операционную систему;

• на процессы, обменивающиеся данными;

• совместно на систему и процессы, разделяя их ответственность. Операционная система может об-
наруживать ошибки при передаче данных и извещать об этом взаимодействующие процессы для
принятия ими решения о дальнейшем поведении.

**Как завершается связь?**

Наконец, важным вопросом при изучении средств обмена данными является вопрос прекращения обме-
на. Здесь нужно выделить два аспекта: требуются ли от процесса какие-либо специальные действия по
прекращению использования средства коммуникации и влияет ли такое прекращение на поведение дру-
гих процессов. Для способов связи, которые не подразумевали никаких инициализирующих действий,
обычно ничего специального для окончания взаимодействия предпринимать не надо. Если же установле-
ние связи требовало некоторой инициализации, то, как правило, при ее завершении бывает необходимо
выполнить ряд операций, например сообщить операционной системе об освобождении выделенного
связного ресурса.

Если кооперативные процессы прекращают взаимодействие согласованно, то такое прекращение не
влияет на их дальнейшее поведение. Иная картина наблюдается при несогласованном окончании связи
одним из процессов. Если какой-либо из взаимодействующих процессов, не завершивших общение, на-
ходится в этот момент в состоянии ожидания получения данных либо попадает в такое состояние позже,
то операционная система обязана предпринять некоторые действия для того, чтобы исключить вечное
блокирование этого процесса. Обычно это либо прекращение работы ожидающего процесса, либо его из-
вещение о том, что связи больше нет (например, с помощью передачи заранее определенного сигнала).

**Нити исполнения**

Рассмотренные выше аспекты логической реализации относятся к средствам связи, ориентированным на
организацию взаимодействия различных процессов. Однако усилия, направленные на ускорение решения
задач в рамках классических операционных систем, привели к появлению совершенно иных механизмов,
к изменению самого понятия "процесс".

В свое время внедрение идеи мультипрограммирования позволило повысить пропускную способность
компьютерных систем, т. е. уменьшить среднее время ожидания результатов работы процессов. Но лю-
бой отдельно взятый процесс в мультипрограммной системе никогда не может быть выполнен быстрее,
чем при работе в однопрограммном режиме на том же вычислительном комплексе. Тем не менее, если
алгоритм решения задачи обладает определенным внутренним параллелизмом, мы могли бы ускорить
его работу, организовав взаимодействие нескольких процессов. Рассмотрим следующий пример. Пусть у
нас есть следующая программа на псевдоязыке программирования:

Ввести массив a
Ввести массив b
Ввести массив c
a = a + b
c = a + c
Вывести массив c

При выполнении такой программы в рамках одного процесса этот процесс четырежды будет блокиро-
ваться, ожидая окончания операций ввода-вывода. Но наш алгоритм обладает внутренним параллелиз-
мом. Вычисление суммы массивов a + b можно было бы выполнять параллельно с ожиданием окончания
операции ввода массива с.

Ввести массив a

Ожидание окончания операции ввода
Ввести массив b

Ожидание окончания операции ввода
Ввести массив с

Ожидание окончания операции ввода a = a + b

c = a + c
Вывести массив с

**Основы операционных систем**

**46**

Ожидание окончания операции вывода

Такое совмещение операций по времени можно было бы реализовать, используя два взаимодействующих
процесса. Для простоты будем полагать, что средством коммуникации между ними служит разделяемая
память. Тогда наши процессы могут выглядеть следующим образом.

Процесс 1

Процесс 2

Ввести массив a
Ожидание окончания
операции ввода
Ввести массив b
Ожидание окончания
операции ввода
Ввести массив с
Ожидание окончания
операции ввода
c = a + c
Вывести массив с
Ожидание окончания
операции вывода

Ожидание ввода
массивов a и b

a = a + b

Казалось бы, мы предложили конкретный способ ускорения решения задачи. Однако в действительности
дело обстоит не так просто. Второй процесс должен быть создан, оба процесса должны сообщить опера-
ционной системе, что им необходима память, которую они могли бы разделить с другим процессом, и,
наконец, нельзя забывать о переключении контекста. Поэтому реальное поведение процессов будет вы-
глядеть примерно так.

Процесс 1 Процесс 2

Создать процесс 2

Переключение контекста

Выделение общей
памяти

Ожидание ввода
a и b

Переключение контекста
Выделение общей памяти
Ввести массив a
Ожидание окончания
операции ввода
Ввести массив b
Ожидание окончания
операции ввода
Ввести массив с
Ожидание окончания
операции ввода

Переключение контекста

a = a + b

Переключение контекста

c = a + c
Вывести массив с
Ожидание окончания
операции вывода

Очевидно, что мы можем не только не выиграть во времени при решении задачи, но даже и проиграть,
так как временные потери на создание процесса, выделение общей памяти и переключение контекста мо-
гут превысить выигрыш, полученный за счет совмещения операций.

Для того чтобы реализовать нашу идею, введем новую абстракцию внутри понятия "процесс" - нить ис-
полнения или просто нить (в англоязычной литературе используется термин thread). Нити процесса раз-
деляют его программный код, глобальные переменные и системные ресурсы, но каждая нить имеет соб-
ственный программный счетчик, свое содержимое регистров и свой стек. Теперь процесс представляется
как совокупность взаимодействующих нитей и выделенных ему ресурсов. Процесс, содержащий всего

**Основы операционных систем**

**47**

одну нить исполнения, идентичен процессу в том смысле, который мы употребляли ранее. Для таких
процессов мы в дальнейшем будем использовать термин "традиционный процесс". Иногда нити называ-
ют облегченными процессами или мини-процессами, так как во многих отношениях они подобны тради-
ционным процессам. Нити, как и процессы, могут порождать нити-потомки, правда, только внутри сво-
его процесса, и переходить из одного состояния в другое. Состояния нитей аналогичны состояниям тра-
диционных процессов. Из состояния **рождение** процесс приходит содержащим всего одну нить исполне-
ния. Другие нити процесса будут являться потомками этой нити-прародительницы. Мы можем считать,
что процесс находится в состоянии **готовность**, если хотя бы одна из его нитей находится в состоянии
**готовность** и ни одна из нитей не находится в состоянии **исполнение**. Мы можем считать, что процесс
находится в состоянии **исполнение**, если одна из его нитей находится в состоянии **исполнение**. Процесс
будет находиться в состоянии **ожидание**, если все его нити находятся в состоянии **ожидание**. Наконец,
процесс находится в состоянии **закончил исполнение**, если все его нити находятся в состоянии **закон-
чила исполнение**. Пока одна нить процесса заблокирована, другая нить того же процесса может выпол-
няться. Нити разделяют процессор так же, как это делали традиционные процессы, в соответствии с рас-
смотренными алгоритмами планирования.

Поскольку нити одного процесса разделяют существенно больше ресурсов, чем различные процессы, то
операции создания новой нити и переключения контекста между нитями одного процесса занимают зна-
чительно меньше времени, чем аналогичные операции для процессов в целом. Предложенная нами схема
совмещения работы в терминах нитей одного процесса получает право на существование.

Нить 1 Нить 2

Создать нить 2

Переключение контекста нитей

Ожидание ввода а и b

Переключение контекста нитей
Ввести массив а
Ожидание окончания
операции ввода
Ввести массив b
Ожидание окончания
операции ввода
Ввести массив с
Ожидание окончания
операции ввода

Переключение контекста нитей

а = а + b

Переключение контекста нитей
c = а + c
Вывести массив с
Ожидание окончания
операции вывода

Различают операционные системы, поддерживающие нити на уровне ядра и на уровне библиотек. Все
сказанное выше справедливо для операционных систем, поддерживающих нити на уровне ядра. В них
планирование использования процессора происходит в терминах нитей, а управление памятью и другими
системными ресурсами остается в терминах процессов. В операционных системах, поддерживающих ни-
ти на уровне библиотек пользователей, и планирование процессора, и управление системными ресурсами
осуществляются в терминах процессов. Распределение использования процессора по нитям в рамках вы-
деленного процессу временного интервала осуществляется средствами библиотеки. В подобных систе-
мах блокирование одной нити приводит к блокированию всего процесса, ибо ядро операционной систе-
мы не имеет представления о существовании нитей. По сути дела, в таких вычислительных системах
просто имитируется наличие нитей исполнения.

Далее в этой части книги для простоты изложения мы будем использовать термин "процесс", хотя все
сказанное будет относиться и к нитям исполнения.

**Основы операционных систем**

**48**

 **Заключение**

Для достижения поставленной цели различные процессы могут исполняться псевдопараллельно на одной
вычислительной системе или параллельно на разных вычислительных системах, взаимодействуя между
собой. Причинами для совместной деятельности процессов обычно являются: необходимость ускорения
решения задачи, совместное использование обновляемых данных, удобство работы или модульный
принцип построения программных комплексов. Процессы, которые влияют на поведение друг друга пу-
тем обмена информацией, называют кооперативными или взаимодействующими процессами, в отличие
от независимых процессов, не оказывающих друг на друга никакого воздействия и ничего не знающих о
взаимном существовании в вычислительной системе.

Для обеспечения корректного обмена информацией операционная система должна предоставить процес-
сам специальные средства связи. По объему передаваемой информации и степени возможного воздейст-
вия на поведение процесса, получившего информацию, их можно разделить на три категории: сигналь-
ные, канальные и разделяемую память. Через канальные средства коммуникации информация может пе-
редаваться в виде потока данных или в виде сообщений и накапливаться в буфере определенного разме-
ра. Для инициализации "общения" процессов и его прекращения могут потребоваться специальные дей-
ствия со стороны операционной системы. Процессы, связываясь друг с другом, могут использовать не-
прямую, прямую симметричную и прямую асимметричную схемы адресации. Существуют одно- и дву-
направленные средства передачи информации. Средства коммуникации обеспечивают надежную связь,
если при общении процессов не происходит потери и повреждения информации, не появляется лишней
информации, не нарушается порядок данных.

Усилия, направленные на ускорение решения задач в рамках классических операционных систем, приве-
ли к появлению новой абстракции внутри понятия "процесс" - нити исполнения или просто нити. Нити
процесса разделяют его программный код, глобальные переменные и системные ресурсы, но каждая нить
имеет собственный программный счетчик, свое содержимое регистров и свой стек. Теперь процесс пред-
ставляется как совокупность взаимодействующих нитей и выделенных ему ресурсов. Нити могут порож-
дать новые нити внутри своего процесса, они имеют состояния, аналогичные состояниям процесса, и мо-
гут переводиться операционной системой из одного состояния в другое. В системах, поддерживающих
нити на уровне ядра, планирование использования процессора осуществляется в терминах нитей испол-
нения, а управление остальными системными ресурсами - в терминах процессов. Накладные расходы на
создание новой нити и на переключение контекста между нитями одного процесса существенно меньше,
чем на те же самые действия для процессов, что позволяет на однопроцессорной вычислительной систе-
ме ускорять решение задач с помощью организации работы нескольких взаимодействующих нитей.