Занятие 09: Управление процессами

Понятия «процесс», «поток» и «ресурс» **1**
Процессы и потоки 1
Понятие ресурса 2
Основные задачи управления процессами **3**
Реализация многозадачного режима **4**
Состояния процесса 4
Невытесняющая многозадачность 6
Вытесняющая многозадачность 8
Резюме **9**
Понятия процесса и ресурса. Основные задачи ОС по управлению процессами 9
Основные состояния процесса и переходы между ними 10
Невытесняющая и вытесняющая многозадачность 11

Литература:

ОИ1. Батаев А.В., Налютин Н.Ю., Синицын С.В. Операционные системы и среды: учебник для студ. учреждений сред. проф. образования. – М.: Изд. центр «Академия», 2015. – 272 с. – С. 70-81 (глава 4 «Процессы»).

ДИ2. Дроздов С.Н. Операционные системы: Конспект лекций. – Таганрог: Таганрогский гос. радиотехн. ун-т, 2008. – 136 с.

Понятия «процесс», «поток» и «ресурс»

Процессы и потоки

http://life-prog.ru/1_56260_ponyatia-protsess-i-potok.html

Процессом является выполняемая программа, включающая текущее значение счетчика команд, регистров и переменных.

Согласно определению, данному Шоу еще на заре компьютерной эры («Логическое проектирование операционных систем». – М.: Мир, 1981. – 360 с.),

последовательный процесс (иногда называемый «задача») есть работа, производимая последовательным процессором при выполнении программы с ее данными

Это определение подчеркивает последовательный характер процесса, т.е. выполнение команд в определенном порядке. Термин «задача» мы будем понимать как синоним термина «процесс». Далее, процесс – понятие динамическое. Программа – это текст, процесс – выполнение этого текста. Конечно, на практике мы часто говорим: «программа вызывает функцию», «программа ждет ввода» и т.п., однако, строго говоря, правильнее было бы «процесс, выполняющий программу, вызывает…».

Еще один важный момент в определении – упоминание данных. В многозадачных системах зачастую одна и та же программа может запускаться несколько раз (например, можно несколько раз открыть текстовый редактор Notepad для разных файлов). Это означает, что несколько процессов могут использовать одну и ту же программу, но с разными данными.

Поток – это последовательность исполняемых команд.

В чем состоят принципиальные отличия в понятиях «процесс» и «поток»?

В операционных системах, где существуют и процессы, и потоки, процесс рассматривается операционной системой как заявка на потребление всех видов ресурсов, кроме одного — процессорного времени. Этот последний важнейший ресурс распределяется операционной системой между другими единицами работы — потоками, которые и получили свое название благодаря тому, что они представляют собой последовательности (потоки выполнения) команд.

Элементы, которые потоки используют совместно:

1) Адресное пространство; 2) Глобальные переменные; 3) Открытые файлы; 4) Дочерние процессы; 5) Необработанные аварийные сигналы; 6) Сигналы и их обработки; 7) Информация об использовании ресурсов.

Понятию «поток» соответствует последовательный переход процессора от одной команды программы к другой.

Элементы индивидуальные для каждого потока:

1) Элементы потока; 2) Счетчик команд; 3) Регистры; 4) Стек; 5) Состояние.

ОС распределяет процессорное время между потоками. Процессу ОС назначает адресное пространство и набор ресурсов, которые совместно используются всеми его потоками.

Для того чтобы процессы не могли вмешаться в распределение ресурсов, а также не могли повредить коды и данные друг друга, важнейшей задачей ОС является *изоляция* одного процесса от другого. Для этого операционная система обеспечивает каждый процесс отдельным виртуальным адресным пространством, так что ни один процесс не может получить прямого доступа к командам и данным другого процесса.

На этом занятии для простоты будем рассматривать процессы, состоящие из одного потока, так что понятия «процесс» и «поток» будут означать одно и то же.

Понятие ресурса

Другим основополагающим понятием, тесно связанным с управлением процессами, является понятие ***ресурса***.

Под ресурсом понимается любой аппаратный или программный объект,

который может понадобиться для работы процессов и доступ к которому может при этом вызвать конкуренцию процессов.

Говоря упрощенно, ресурс – это нечто дефицитное в вычислительной системе. К важнейшим ресурсам любой системы относятся процессор (точнее сказать, процессорное время), основная память, периферийные устройства, файлы. В зависимости от конкретной ОС, к дефицитным ресурсам могут относиться места в таблице процессов или в таблице открытых файлов, буферы кэша, блоки в файле подкачки и другие системные структуры данных.

Если два процесса никак не связаны логикой своей работы, то никакой синхронизации или обмена данными между ними нет, однако такие процессы все же могут оказывать косвенное влияние друг на друга вследствие ***конкуренции за ресурсы***. Многозадачная ОС управляет доступом процессов к ресурсам. В некоторых случаях система временно закрепляет ресурс за одним процессом, отказывая другим процессам в доступе или заставляя их ждать освобождения ресурса. В других случаях оказывается возможным совместный доступ нескольких процессов к одному ресурсу.

Основные задачи управления процессами

Под управлением процессами понимаются процедуры ОС, обеспечивающие запуск системных и прикладных программ, их выполнение и завершение.

В однозадачных ОС управление процессами решает следующие задачи:

загрузка программы в память, подготовка ее к запуску и запуск на выполнение;
выполнение системных вызовов процесса;
обработка ошибок, возникших в ходе выполнения;
нормальное завершение процесса;
прекращение процесса в случае ошибки или вмешательства пользователя.

Все эти задачи решаются сравнительно просто. В многозадачном режиме добавляются значительно более серьезные задачи:

эффективная реализация параллельного выполнения процессов на единственном процессоре, переключение процессора между процессами;
выбор очередного процесса для выполнения с учетом заданных приоритетов процессов и статистики использования процессора;
исключение возможности несанкционированного вмешательства одного процесса в выполнение другого;
предотвращение или устранение тупиковых ситуаций, возникающих при конкуренции процессов за системные ресурсы;

обеспечение синхронизации процессов и обмена данными между ними.

Реализация многозадачного режима

При описании работы многозадачных систем основное внимание уделяется вопросам, связанным с ***параллельным выполнением***, т.е. с одновременной работой нескольких процессов.

Состояния процесса

Любой процесс в многозадачной ОС многократно испытывает переход из одного ***состояния*** в другое.

Основных состояний всего три.

Работа (running) – в этом состоянии находится процесс, программу которого в данный момент выполняет процессор. Работающий процесс иногда удобно называть также ***текущим*** процессом.
Готовность (ready) – состояние, их которого процесс может быть переведен в состояние работы, как только это сочтет нужным сделать ОС.
Блокировка или, что то же самое, ***сон*** (sleeping, waiting) – состояние, в котором процесс не может продолжать выполнение, пока не произойдет некоторое *внешнее* по отношению к процессу событие.

Первые два состояния часто объединяют понятием ***активного*** состояния процесса.

На рис. 1 показаны основные состояния процесса и переходы между ними. Этот рисунок кочует из книги в книгу, поскольку он действительно наглядно отражает самую суть работы многозадачных систем.

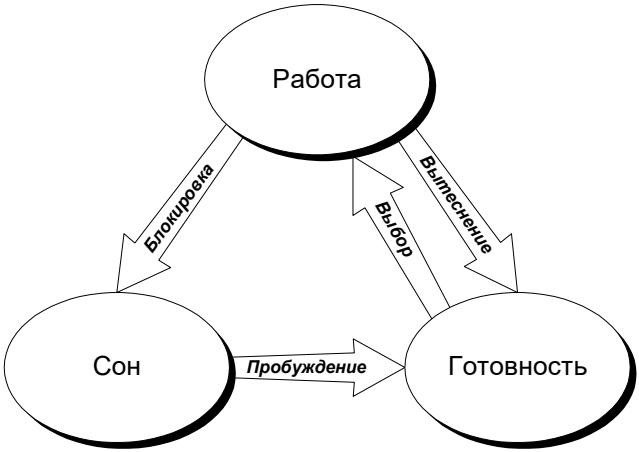


Рис. 1. Основные состояния процесса

Для состояний готовности и сна общее то, что процесс не работает. В чем разница между этими двумя «способами не работать»?

Готовый к выполнению процесс не выполняется только потому, что есть другие не менее готовые процессы, по мнению системы более достойные занимать сейчас процессорное время. В каждый момент времени выбор одного из готовых процессов на роль работающего определяется логикой работы ОС. Этот выбор должен обеспечивать эффективную квазипараллельную работу готовых процессов.

В отличие от этого, спящий процесс – это всегда процесс, ожидающий некоторого конкретного события. Спящий процесс не сможет заработать, даже если процессор вдруг окажется свободным. Такой процесс, в соответствии со своей собственной логикой, ждет чего-то, что должно произойти.

Чего он может ждать? Ну, например:

- завершения начатой операции синхронного ввода/вывода (т.е., например, процесс ждет нажатия клавиши **`Enter`** или окончания записи на диск);
- освобождения запрошенного у системы ресурса (например, дополнительной области памяти или открытого файла);
- истечения заданного интервала времени («*посплю-ка я минут десять!*») или достижения заданного момента времени («*разбудите меня ровно в полночь!*») (в обоих случаях процесс ждет сигнала от запрограммированного таймера);
- сигнала на продолжение действий от другого, взаимосвязанного процесса;
- сообщения от системы о необходимости выполнить определенные действия (например, перерисовать содержимое окна).

В любом из названных (и многих неназванных) случаев должно произойти некоторое событие, источник которого лежит *вне* данного процесса.

Чисто условно можно сказать, что если бы в вычислительную систему вдруг было добавлено еще несколько процессоров, то «готовые» процессы могли бы сразу перейти в состояние «работа», но «спящие» продолжили бы свой сон.

Любая многозадачная ОС предоставляет в распоряжение прикладных программ набор функций, переводящих вызвавший их процесс в состояние сна, в котором процесс не пытается использовать процессорное время (другими словами, состояние сна есть состояние ***пассивного ожидания***). Такие системные функции называются ***блокирующими***. К их числу относятся функции синхронного ввода/вывода, запроса ресурсов, приостановки до заданного времени, получения сообщений и многие другие.

Поскольку ОС берет на себя блокировку, «усыпление» процесса, она должна обеспечить и его разблокировку, «пробуждение». Чтобы это стало возможным, система должна для каждого спящего процесса помнить, «чего он ждет», т.е. помнить условия пробуждения процесса. Система отслеживает все

события, способные разблокировать какой-либо процесс (во многих случаях используя для этого аппаратные прерывания) и, когда для одного или сразу нескольких процессов наступает ожидаемое событие, переводит эти события из состояния сна в состояние готовности.

Рассмотрим возможные переходы между состояниями процесса, показанные на рисунке стрелками.

Переход *Работа* → *Сон* представляет собой блокировку процесса, которая может произойти при вызове блокирующей системной функции.

Переход *Сон* → *Готовность* – это пробуждение процесса, оно выполняется системой при возникновении соответствующего условия.

Переход *Работа* → *Готовность* ранее не рассматривался.

Переход *Работа* → *Готовность* называется ***вытеснением*** процесса и выполняется системой, когда она принимает решение о смене текущего процесса.

Для обратного перехода *Готовность* → *Работа* нет общепринятого термина. Будем называть его ***выбором*** процесса для выполнения. Отметим, что этот переход почти всегда связан либо с блокировкой, либо с вытеснением прежнего текущего процесса.

Двух стрелок нет на диаграмме. Прямой переход от сна к работе нелогичен, т.к. он совмещал бы два совершенно разных действия. *Каких именно?*

Переход от готовности ко сну невозможен в принципе.

Кстати, почему?

Помимо трех основных состояний, в различных ОС могут использоваться и другие состояния.

Состояние ***старта*** означает, что процесс находится на этапе создания и пока не готов вступить в работу.
Состояние ***завершения*** (в UNIX оно почти официально называется «зомби») означает, что процесс завершил свою работу, но пока присутствует в системе в виде записи о результатах и причине завершения.
Состояние ***приостановки*** (suspended) означает, что выполнение процесса временно прервано оператором (или, может быть, другим процессом) и позднее должно быть им же возобновлено.

Невытесняющая многозадачность

Все многозадачные ОС можно разделить на два класса, различающиеся по способам организации переключения процессов.

В системах с ***невытесняющей диспетчеризацией*** (non-preemptive multitasking) работа любого процесса может быть прервана только «по инициативе самого процесса», а точнее – только когда процесс вызывает определенные системные функции.

Если работающий процесс не вызывает системных функций, а занимается, например, долгими расчетами, то все остальные процессы вынуждены простаивать. Когда же системная функция, наконец, вызвана, то система, прежде всего, проверяет, может ли эта функция быть выполнена сразу или предполагает ожидание некоторого события (требуемый ресурс может быть занят другим процессом; операция ввода/вывода обычно требует определенного времени на свое выполнение; очередь сообщений, требующих обработки, может быть пуста). Если требуется ожидание, то система блокирует процесс, выбирая какой-либо другой из готовых процессов для выполнения. Не исключена ситуация, когда заблокированными по разным причинам оказываются все пользовательские процессы.

Но даже если действие, требуемое процессом, может быть выполнено без блокировки, система не обязательно спешит выполнить его и вернуть управление текущему процессу. На самом деле, момент вызова системной функции удобен для того, чтобы система могла решить, не пора ли процессу «отдохнуть», поскольку имеются другие процессы, давно претендующие на процессорное время или обладающие более высоким приоритетом. В этом случае система переводит процесс в состояние готовности к выполнению, а один из других готовых к выполнению процессов становится текущим. Что же касается вызванной системной функции, то ее выполнение будет завершено позднее, когда вызвавший процесс вновь станет текущим.

В предыдущем пункте смена текущего процесса, не связанная с его блокировкой, была названа вытеснением процесса.

> Та часть ОС, которая по определенным в системе правилам выбирает, следует ли вытеснить текущий процесс и какой процесс должен стать следующим текущим, называется ***планировщиком*** (scheduler) или ***диспетчером процессов***.

В системе с невытесняющей диспетчеризацией планировщик получает управление при вызове процессом одной из блокирующих или вытесняющих функций. Планировщик проверяет, не выполнены ли условия активизации одного из спящих процессов, а затем принимает решение о выборе одного из активных процессов для выполнения. Не исключено, что будет опять выбран тот же процесс (если только он не блокируется).

Использование ***невытесняющей диспетчеризации*** позволило разработать достаточно впечатляющие примеры ОС, из которых наиболее известной является Windows версий 1, 2 и 3. Обычно в таких системах большая часть процессов находится в спящем состоянии, ожидая, пока пользователь обратится к соответствующему приложению. Для пользователя это выглядит совершенно естественно.

> Главный недостаток многозадачности невытесняющего типа заключается в том, что любой процесс в принципе имеет возможность полностью и надолго захватить процессор.

Это «многозадачность на честном слове», основанная на предположении, что программы всех процессов написаны так, чтобы достаточно часто вызывать блокирующие функции. Даже если программист пишет программу для выполнения сложных многочасовых вычислений, он должен искусственно вставлять в некоторых местах вызов системных функций, передающих управление планировщику. Если же это не сделано, то система фактически теряет многозадачность и будет выполнять один процесс, не реагируя на действия пользователя, до тех пор, пока либо «нахальный» процесс завершится, либо рассвирепевший пользователь снимет его по Ctrl+Alt+Del.

Вытесняющая многозадачность

Более сложным и совершенным типом многозадачных ОС являются системы с ***вытесняющей диспетчеризацией*** процессов (preemptive multitasking). Их отличие заключается в том, что

> В системах с вытесняющей многозадачностью планировщик вступает в работу не только (и не столько) при вызове системных функций, но и в следующих двух случаях (или хотя бы в одном из них):
> - когда активизируется (т.е. пробуждается или запускается) процесс, обладающий более высоким приоритетом, чем текущий;
> - когда истекает ***квант времени***, выделенный планировщиком для текущего процесса.

Понятие ***приоритета*** процесса будет подробно рассмотрено чуть ниже. Приоритет характеризует относительную важность или срочность данного процесса. Надо отметить, что немедленный вызов планировщика при активизации высокоприоритетного процесса характерен не для всех систем с вытесняющей диспетчеризацией, а только для одного их подкласса – систем с ***абсолютными приоритетами***. В системах с ***относительными приоритетами*** процесс с высоким приоритетом будет все же вынужден подождать истечения кванта времени.

> Системы, в которых перепланировка процесса выполняется после истечения каждого кванта времени, называют системами с ***квантованием времени***[1] (time slicing).

Если величина кванта достаточно мала, то для пользователя процесс периодической смены текущего процесса будет незаметен и создастся впечатление, что все активные процессы работают как бы одновременно. С другой стороны, чем меньше величина кванта, тем большую долю процессорного времени будет занимать процедура переключения текущего процесса.

---

[1] Иногда используют также термин «разделение времени» (time sharing). Однако этот термин лучше оставить для обозначения многотерминальных ОС. К системам с квантованием времени относятся также ОС реального времени, заметно отличающиеся от систем разделения времени.

Вызов планировщика не обязательно означает смену процесса. Если нет других активных процессов с таким же или более высоким приоритетом, то планировщик может продолжить выполнение того же процесса.

> Принципиальной чертой систем с вытесняющей диспетчеризацией является то, что текущий процесс может быть прерван и вытеснен практически в любой точке своей программы.

Это заметно усложняет реализацию таких систем по сравнению с невытесняющими системами, где смена текущего процесса может произойти только в момент вызова системной функции.

Резюме

Понятия процесса и ресурса. Основные задачи ОС по управлению процессами

Последовательный **процесс** (иногда называемый «задача») есть работа, производимая последовательным процессором при выполнении программы с ее данными.

В многозадачных системах зачастую одна и та же программа может запускаться несколько раз (например, можно несколько раз открыть текстовый редактор Notepad для разных файлов). Это означает, что несколько процессов могут использовать одну и ту же программу, но с разными данными.

Под **ресурсом** понимается любой аппаратный или программный объект, который может понадобиться для работы процессов и доступ к которому может при этом вызвать конкуренцию процессов. Говоря упрощенно, ресурс – это нечто дефицитное в вычислительной системе. К важнейшим ресурсам любой системы относятся процессор (точ-нее сказать, процессорное время), основная память, периферийные устройства, файлы.

Под управлением процессами понимаются процедуры ОС, обеспечивающие запуск системных и прикладных программ, их выполнение и завершение.

В однозадачных ОС управление процессами решает следующие задачи:

- загрузка программы в память, подготовка ее к запуску и запуск на выполнение;
- выполнение системных вызовов процесса;
- обработка ошибок, возникших в ходе выполнения;
- нормальное завершение процесса;
- прекращение процесса в случае ошибки или вмешательства пользователя.

В многозадачном режиме добавляются значительно более серьезные задачи:

эффективная реализация параллельного выполнения процессов на единственном процессоре, переключение процессора между процессами;

выбор очередного процесса для выполнения с учетом заданных приоритетов процессов и статистики использования процессора;

исключение возможности несанкционированного вмешательства одного процесса в выполнение другого;

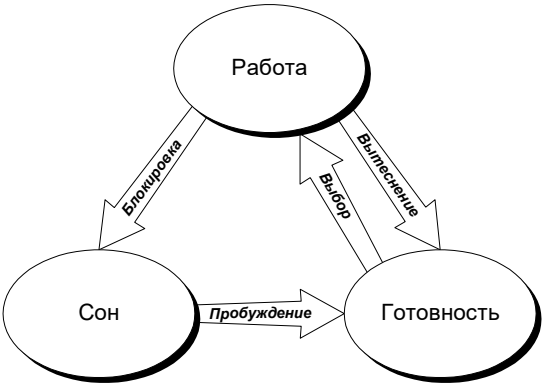
предотвращение или устранение тупиковых ситуаций, возникающих при конкуренции процессов за системные ресурсы;

обеспечение синхронизации процессов и обмена данными между ними.

<Кратко – об основных состояниях процесса>

Основные состояния процесса и переходы между ними

Любой процесс в многозадачной ОС многократно испытывает переход из одного состояния в другое. Основных состояний всего три.



Работа (running) – в этом состоянии находится процесс, программу которого в данный момент выполняет процессор. Работающий процесс иногда удобно называть также текущим процессом.

Готовность (ready) – состояние, их которого процесс может быть переведен в состояние работы, как только это сочтет нужным сделать ОС.

Блокировка или, что то же самое, **сон** (sleeping, waiting) – состояние, в котором процесс не может продолжать выполнение, пока не произойдет некоторое *внешнее* по отношению к процессу событие.

Спящий процесс – это всегда процесс, ожидающий некоторого конкретного внешнего события. Спящий процесс не сможет заработать, даже если процессор вдруг окажется свободным

Переход *Работа* → *Сон* представляет собой блокировку процесса, которая может произойти при вызове блокирующей системной функции.

Переход *Сон* → *Готовность* – это пробуждение процесса, оно выполняется системой при возникновении соответствующего условия.

Переход *Работа* → *Готовность* называется вытеснением процесса и выполняется системой, когда она принимает решение о смене текущего процесса.

Для обратного перехода *Готовность* → *Работа* нет общепринятого термина. Можно называть его выбором процесса для выполнения. Этот пере-

ход почти всегда связан либо с блокировкой, либо с вытеснением прежнего текущего процесса.

Помимо трех основных состояний, в различных ОС могут использоваться и другие состояния.

Состояние *старта* означает, что процесс находится на этапе создания и пока не готов вступить в работу.

Состояние *завершения* (в UNIX оно почти официально называется «зомби») означает, что процесс завершил свою работу, но пока присутствует в системе в виде записи о результатах и причине завершения.

Состояние *приостановки* (suspended) означает, что выполнение процесса временно прервано оператором (или, может быть, другим процессом) и позднее должно быть им же возобновлено.

Невытесняющая и вытесняющая многозадачность

В системах с невытесняющей диспетчеризацией (non-preemptive multitasking) работа любого процесса может быть прервана только «по инициативе самого процесса», а точнее – только когда процесс вызывает определенные системные функции.

Если работающий процесс не вызывает системных функций, а занимается, например, долгими расчетами, то все остальные процессы вынуждены простаивать. При вызове системной функции система проверяет возможность выполнения этой функции и затем принимает решение о том, какой процесс будет выполняться дальше.

Та часть ОС, которая по определенным в системе правилам выбирает, следует ли вытеснить текущий процесс и какой процесс должен стать следующим текущим, называется **планировщиком** (scheduler) или диспетчером процессов.

В системе с невытесняющей диспетчеризацией планировщик получает управление при вызове процессом одной из блокирующих или вытесняющих функций. Планировщик проверяет, не выполнены ли условия активизации одного из спящих процессов, а затем принимает решение о выборе одного из активных процессов для выполнения. Не исключено, что будет опять выбран тот же процесс (если только он не блокируется).

Главный недостаток многозадачности невытесняющего типа заключается в том, что любой процесс в принципе имеет возможность полностью и надолго захватить процессор.

В системах с вытесняющей многозадачностью планировщик вступает в работу не только (и не столько) при вызове системных функций, но и в следующих двух случаях (или хотя бы в одном из них):

- когда активизируется (т.е. пробуждается или запускается) процесс, обладающий более высоким приоритетом, чем текущий;
- когда истекает ***квант времени***, выделенный планировщиком для текущего процесса.

Та часть ОС, которая по определенным в системе правилам выбирает, следует ли вытеснить текущий процесс и какой процесс должен стать следу-

ющим текущим, называется **планировщиком** (scheduler) или диспетчером процессов.

Приоритет характеризует относительную важность или срочность данного процесса. Немедленный вызов планировщика при активизации высокоприоритетного процесса характерен не для всех систем с вытесняющей диспетчеризацией, а только для одного их подкласса – систем с ***абсолютными приоритетами***. В системах с ***относительными приоритетами*** процесс с высоким приоритетом будет все же вынужден подождать истечения кванта времени.

Системы, в которых перепланировка процесса выполняется после истечения каждого кванта времени, называют системами с квантованием времени (time slicing).

Принципиальной чертой систем с вытесняющей диспетчеризацией является то, что текущий процесс может быть прерван и вытеснен практически в любой точке своей программы.