Урок 2. Процессы

Основные определения

Программа - статический объект, представляющий собой файл или совокупность файлов с кодами и данными. Для того чтобы программа могла быть запущена на выполнение, операционная система должна создать окружение или среду выполнения задачи, включающую возможности доступа к различным системным ресурсам (память, устройства ввода-вывода, файлы и т.д.). Такое окружение получило название процесса.

Процесс представляет собой исполняемый образ программы, включающий отображение в памяти исполняемого файла, полученного в результате компиляции и связывания исходного кода программы, кодов данных и библиотек, стека, а также ряда структур данных ядра, необходимых для управления процессом.

В целом, процесс можно представить как совокупность данных ядра системы, необходимых для описания образа программы в памяти и управления ее выполнением.

Процессы можно условно разбить на три категории:

- системные;
- фоновые (демоны);
- прикладные (пользовательские).

Системные процессы являются частью ядра ОС и всегда расположены в оперативной (основной) памяти. Выполняемые инструкции и данные этих процессов находятся в ядре системы, и поэтому они могут вызывать функции и обращаться к данным, недоступным для остальных процессов, например диспетчер страничного замещения, диспетчер памяти ядра, диспетчер буферного кэша и другие.

Фоновые процессы или демоны - это неинтерактивные процессы, которые обычно запускаются при инициализации системы (после инициализации ядра) и обеспечивают работу различных подсистем.

Например, системы терминального доступа, системы печати, системы сетевого доступа и другие. Демоны не связаны с пользовательскими сеансами работы и не могут непосредственно управляться пользователями.

Прикладные процессы, как правило, порождаются в рамках пользовательского сеанса. Они могут выполняться как в интерактивном, так и в фоновом режимах.

Большинство процессоров поддерживают два режима работы:

привилегированный, или режим ядра, и ***пользовательский***, или режим задачи. Определенные команды выполняются только в привилегированном режиме, например команды управления памятью и вводом-выводом. Режим работы устанавливается в регистре слова состояния процессора (PSW) битом режима выполнения, который может быть изменен при наступлении некоторых событий. Например, если в результате прерывания управление пользовательским процессом переходит к процедуре ОС, данная процедура изменяет режим выполнения на привилегированный. Перед возвращением управления пользовательскому процессу режим выполнения изменяется обратно на пользовательский. Программы, выполняющиеся в режиме ядра, обладают полным контролем над процессором и имеют доступ ко всем ячейкам памяти.

Модель процесса

Можно рассматривать все функционирующее на компьютере программное обеспечение, включая операционную систему, в виде набора процессов. Каждый процесс можно описать набором параметров, включая текущие значения счетчика команд, регистров и переменных. С позиций данной абстрактной модели у каждого процесса есть собственный виртуальный центральный процессор. В действительности реальный процессор переключается с одного процесса на другой. Это переключение и называется многозадачностью или мультипрограммированием.

Для реализации модели процессов операционная система содержит системную таблицу процессов (массив структур) с одним элементом для каждого процесса. Данный элемент называется блоком управления или ***дескриптором*** процесса. В дескрипторе процесса прямо или косвенно (через указатели на связанные с процессом структуры) содержится информация о состоянии процесса, его приоритетах, идентификаторе, параметрах планирования, о расположении образа процесса в оперативной памяти и на диске, об ожидаемых процессом событиях, а также другая оперативная информация, необходимая ядру системы в течение всего жизненного цикла процесса, независимо от того, находится ли процесс в активном или пассивном состоянии. При управлении процессами ядро ОС кроме дескриптора использует другую информационную структуру, называемую ***контекстом процесса***. Контекст процесса содержит более объемную часть информации о процессе, необходимую для возобновления выполнения прерванного процесса. Эта информация включает содержимое регистров процессора, данные об открытых файлах и незавершенных операциях ввода-вывода, коды ошибок выполняемых системных вызовов и другие данные, характеризующие состояние вычислительной среды в момент прерывания.

Каждому процессу операционной системой выделяется ***виртуальное адресное пространство***, представляющее собой набор виртуальных адресов, необходимых для выполнения процесса. Для прикладных программ эти адреса первоначально назначаются транслятором при создании сегментов кода и данных. Затем, при создании процесса, ОС фиксирует назначенное виртуальное адресное пространство в собственных системных таблицах. В ходе выполнения процесс может увеличить размер назначенного виртуального адресного пространства, запросив у ОС создания дополнительных сегментов или увеличения существующих. Максимальный размер виртуального адресного пространства ограничивается разрядностью адреса данной архитектуры компьютера. Например, для 32-разрядных процессоров Intel Pentium ОС может предоставить каждому процессу виртуальное адресное пространство до 4 Гбайт. Содержимое назначенного процессу виртуального адресного пространства представляет собой образ процесса.

Выполнение процесса может происходить в двух режимах: в ***режиме ядра (kernel mode)*** или ***режиме задачи (user mode)***.

В режиме задачи процесс выполняет инструкции прикладной программы, допустимые на непривилегированном уровне защиты процессора. При этом процессу недоступны системные структуры данных. Для получения услуг ядра процессу необходимо сделать системный вызов, после чего выполнение процесса переходит на

привилегированный уровень (в режим ядра). Таким образом, ядро системы защищает собственное адресное пространство от доступа прикладного процесса, который может нарушить целостность структур данных ядра. Соответственно и образ процесса состоит из двух частей: данных режима задачи и данных режима ядра. Образ процесса в режиме задачи состоит из сегментов кода, данных, стека, библиотек. Образ процесса в режиме ядра состоит из структур данных, которые используются ядром для управления процессом.

Управление процессами

Для управления процессами и выделяемыми им ресурсами ОС использует четыре вида управляющих таблиц.

Таблицы памяти - используются для отслеживания оперативной и виртуальной памяти. Содержат следующую информацию:

1) объем оперативной и виртуальной памяти, отведенной процессу;
2) атрибуты защиты блоков памяти (указывают, какой из процессов имеет доступ к той или иной совместно используемой памяти);
3) данные, необходимые для управления виртуальной памятью.

Таблицы ввода-вывода - используются для управления устройствами ввода-вывода. В каждый момент времени устройство ввода-вывода может быть либо свободно, либо занято определенным процессом.

Таблицы файлов - содержат информацию о существующих файлах, их расположении, текущем состоянии и других атрибутах. Основная часть этой информации поддерживается файловой системой.

Таблицы процессов - содержат указатели на образы процессов.

Все четыре вида таблиц связаны между собой (рис.2.1) и имеют перекрестные ссылки. Данные таблицы создаются при генерации ОС на основе информации, содержащейся в конфигурационных файлах.

Управление процессами осуществляется ОС в режиме ядра и включает следующие функции:

- создание и завершение процессов;
- планирование и диспетчеризация процессов;
- переключение процессов;
- синхронизация и обмен информацией между процессами;
- организация управляющих блоков процессов.

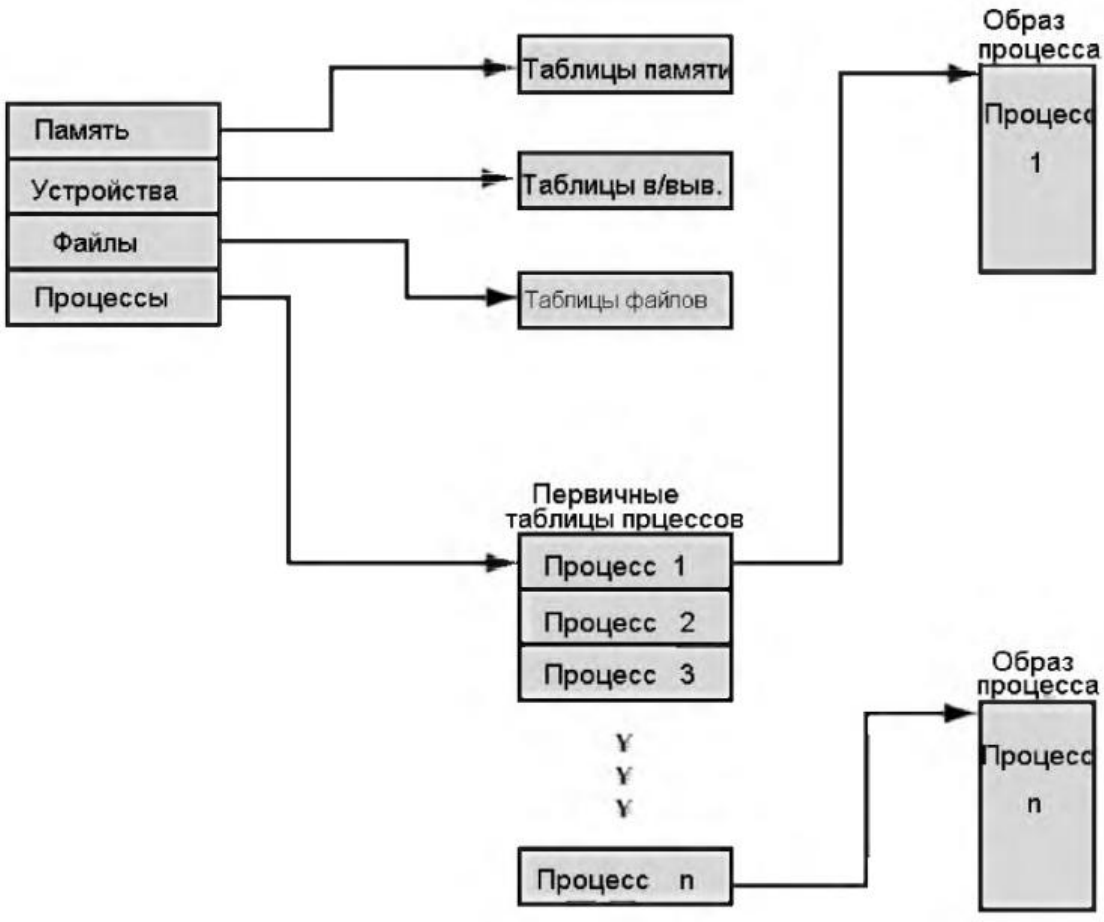


Рис 2.1. Структура управляющих таблиц ОС

Создание и завершение процессов

Для создания нового процесса операционной системе необходимо выполнить определенную последовательность действий:

1) присвоить новому процессу уникальный идентификатор, т.е. занести новую запись в таблицу процессов;
2) выделить пространство для процесса, т.е. выделить адресное пространство для всех элементов образа процесса;
3) инициализировать управляющий блок процесса;
4) поместить процесс в список “готовых ” или “готовых приостановленных процессов ”;
5) загрузить часть кодов и данных процесса в оперативную память.

Информация о состоянии процессора обычно инициализируется нулевыми значениями, за исключением счетчика команд (содержит точку входа в программу) и указателей системного стека (задающих границы стека процесса). Состояние процесса обычно инициализируется значением “готов” или “готов и приостановлен”.

Основными причинами создания процессов являются:

1) запуск задач пользователей и заданий в среде пакетной обработки;
2) поступление запросов от приложений на выполнение некоторых функций;
3) порождение процессов другими процессами.

Когда один процесс порождает другой, то порождающий процесс называется ***родительским (parent)***, а порождаемый процесс называется ***дочерним (child)***. Порождение процессов используется для структурирования приложений или распараллеливания вычислений. Например, файловый сервер может генерировать новый процесс для каждого обрабатываемого им запроса.

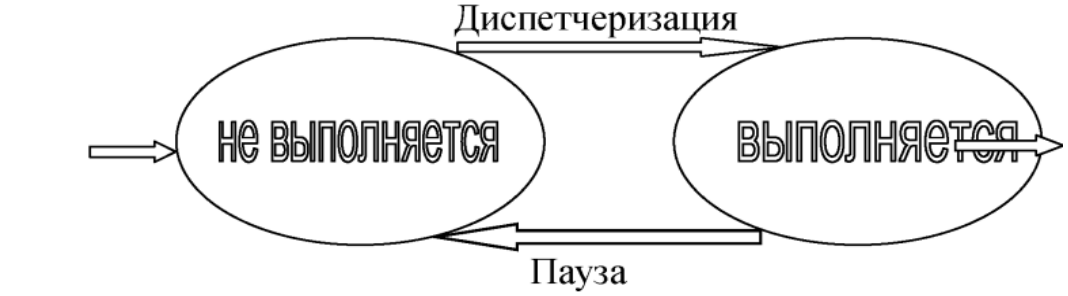
Основными причинами завершения процессов являются:

1) нормальное завершение;
2) превышение предельного лимита времени;
3) превышение лимита отведенной памяти;
4) ошибки при выполнении;
5) вмешательство пользователя, администратора или ОС;
6) завершение родительского процесса.

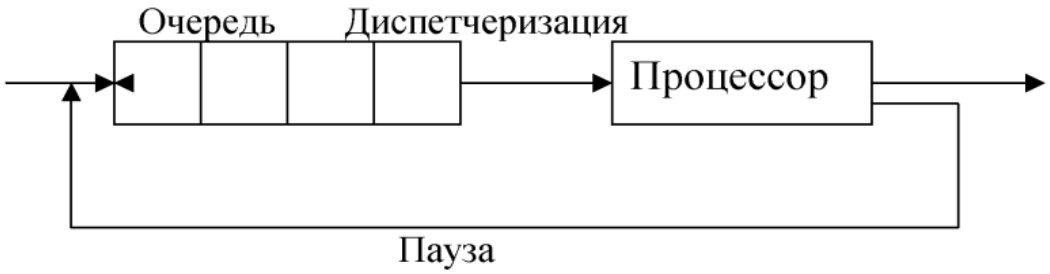
Планирование и диспетчеризация процессов

Самую простую модель диспетчеризации, т.е. определения схемы чередования процессов, можно построить, используя модель процесса с двумя состояниями: выполняющийся или невыполняющийся.

На рис. 2.2 показана модель процесса с двумя состояниями.



а) Схема перехода состояний



б) Схема использования очереди

Рис 2.2. Модель процесса с двумя состояниями

Поведение диспетчера можно описать следующим образом.

Прерванный процесс переходит в очередь процессов, ожидающих выполнения. В случае завершения процесс выводится из системы. Диспетчер выбирает из очереди следующий процесс для выполнения.

Если бы все процессы всегда были готовы к выполнению, то очередь на рис 2.2 могла бы работать эффективно, т.е. обслуживать имеющиеся в наличии процессы карусельным (round robin) методом, когда каждому процессу отводится определенный

промежуток времени, по истечении которого процесс возвращается обратно в очередь. Однако модель с двумя состояниями не является адекватной. Некоторые из невыполняющихся процессов готовы к выполнению, в то время как другие являются заблокированными и ждут окончания операции ввода-вывода. Поэтому более адекватной реальности является модель процесса с пятью состояниями:

- выполняющийся;
- готовый к выполнению;
- блокированный (процесс не может выполниться до наступления некоторого события, например до завершения операции ввода-вывода);
- новый (только что созданный процесс, который еще не помещен ОС в пул выполнимых процессов и не загружен в ОЗУ);
- завершающийся (процесс, удаленный ОС из пула выполнимых процессов).

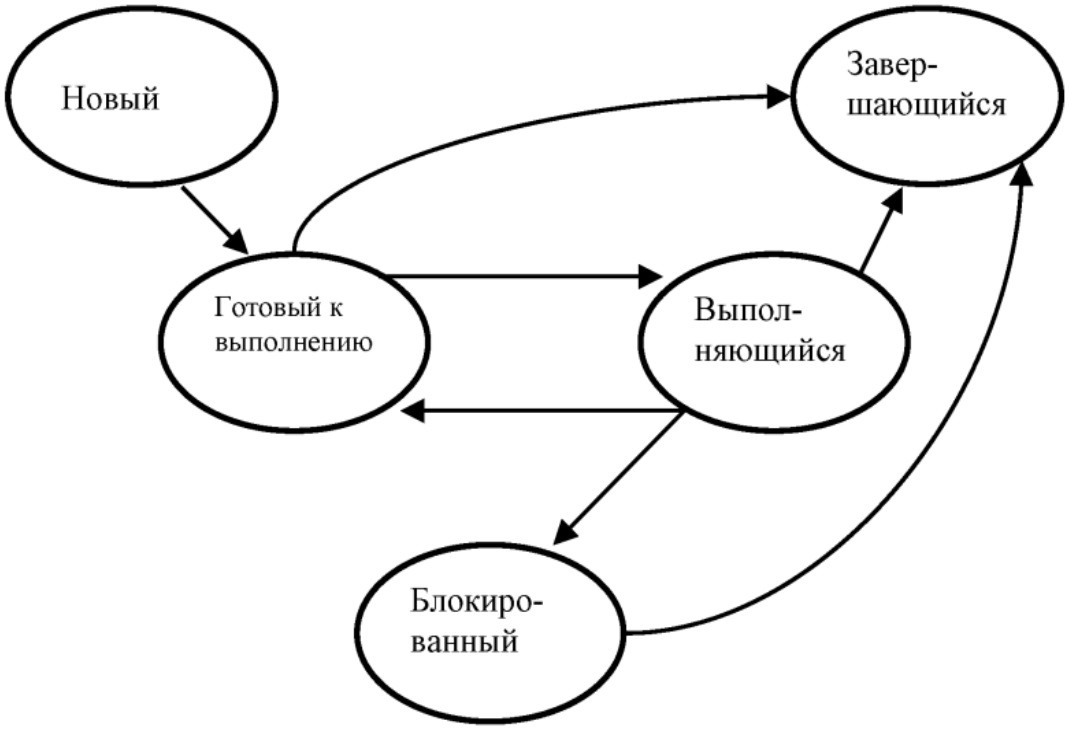
На рис. 2.3 показаны типы событий, соответствующие каждому из возможных переходов из одного состояния в другое. Возможны следующие переходы:

- Нулевое состояние/Новый. ОС выполняет все необходимые действия для создания нового процесса. Присваивает процессу идентификатор, формирует управляющие таблицы процесса.
- Новый/Готовый. ОС переводит процесс в состояние готового к выполнению по мере готовности к обработке дополнительных процессов. Существуют ограничения на количество запущенных процессов и на объем выделяемой для процессов виртуальной памяти (для поддержки нормальной производительности системы).
- Готовый/Выполняющийся. ОС выбирает один из готовых для выполнения процессов в соответствии с принятой стратегией планирования.
- Выполняющийся/Завершающийся. ОС прекращает выполнение процесса по одной из причин завершения процессов.
- Выполняющийся/Готовый. ОС прерывает выполнение процесса или вытесняет процесс по истечении заданного кванта времени или из-за наличия другого процесса с более высоким приоритетом.
- Выполняющийся/Блокированный. Процесс переводится в заблокированное состояние, если для продолжения работы требуется наступление некоторого события (ожидание завершения операции ввода-вывода, сообщения от другого процесса и т.п.).
- Блокированный/Готовый. Заблокированный процесс переходит в состояние готовности к выполнению при наступлении ожидаемого события.
- Готовый/Завершающийся. Родительский процесс может прервать выполнение дочернего процесса. Дочерний процесс может прекратиться при завершении родительского процесса.
- Блокированный/Завершающийся. Аналогично предыдущему пункту.
- Обобщением модели с пятью состояниями является модель с семью состояниями, позволяющая более адекватно смоделировать реализацию операционных систем. В данной модели вводится дополнительно понятие приостановленного процесса и два новых состояния:
- Блокированный/Приостановленный. Процесс, выгруженный на диск и ожидающий какого-то события.

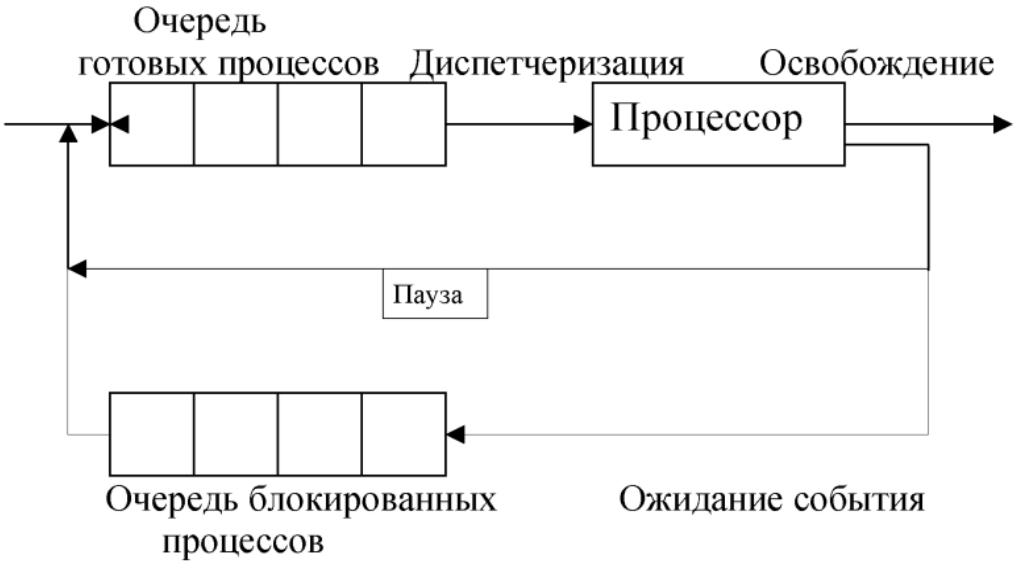
- Готовый/Приостановленный. Процесс, находящийся на диске, но уже готовый к выполнению. Для этого его необходимо только загрузить в основную память.

Приостановленный процесс является процессом, который отсутствует в основной памяти и не может быть запущен немедленно. Причинами приостановки процесса могут быть:

1) необходимость освободить основную память для оптимизации производительности виртуальной памяти;
2) периодический режим выполнения процесса (например, программа анализа использования ресурсов компьютера);
3) приостановка дочерних процессов родительским для их проверки или координации;
4) приостановка пользовательского процесса для отладки программы;
5) приостановка фоновых процессов операционной системой в случае выявления проблем.



а) Схема перехода состояний



б) Схема с одной очередью блокированных процессов

Рис 2.3. Модель процесса с пятью состояниями

Литературные источники

Илюшкин Б.И. Операционные системы. Процессы и потоки: Учеб. пособие. – СПб.: СЗТУ, 2005, - 103 с.