## **ПОНЯТИЕ ПРОЦЕССА**

**Процессом** называется некоторая деятельность, выполняемая на процессоре. **Процессором** в широком смысле называется любое устройство в составе ЭВМ, способное автоматически выполнять определенную последовательность действий по программе, хранимой в памяти ЭВМ. Можно утверждать, что архитектура современной ЭВМ является многопроцессорной.

Процесс, выполняемый на центральном процессоре (внутренний процесс), представляет собой программу. При исполнении программ на центральном процессоре выделяются следующие допустимые состояния:

1. *порождение* – подготавливаются условия для первого исполнения программы на процессоре;
2. *активность* – программа исполняется на процессоре;
3. *ожидание* – программа не исполняется на процессоре из-за отсутствия какого-либо требуемого ресурса;
4. *готовность* – программа не исполняется, имея все необходимые ресурсы, кроме ЦП;
5. *окончание* – нормальное или аварийное завершение программы, после чего ЦП и другие ресурсы ей не предоставляются.

Интервал времени между порождением и завершением называется *интервалом существования процесса*.

Допустимые состояния процесса и переходы между ними удобно изображать в виде *графа существования процесса* (рис. 1).

**Рис.1.** Граф существования процесса

Рассмотрим классификацию процессов по ряду признаков (рис. 2).

1. Процессы, которые необходимо завершить до наступления конкретного момента времени, называются **процессами реального времени.**
2. Процессы, время существования которых не превышает времени допустимой реакции на запросы пользователя, называются **интерактивными**.
3. Остальные процессы называются **пакетными**.

Процессы, развивающиеся на ЦП, называется **внутренними**, или **программными**. Процессы, развивающиеся на других процессорах, отличных от ЦП, называются **внешними** (процессы ввода-вывода).

Программные процессы, в свою очередь, делятся на **системные** (исполняется программа из состава ОС) и **пользовательские** (исполняется прикладная программа).

Два процесса, имеющие одинаковый конечный результат при одинаковых исходных данных называются **эквивалентными**. Если при этом используется одна и та же программа, то процессы называются **тождественными**. Если совпадают трассы процессов, то они **равные**. Во всех остальных случаях процессы **различные**.

Если интервалы существования двух процессов не пересекаются во времени, то это **последовательные** процессы. В противном случае – **параллельные**. Если на части рассматриваемого интервала процессы развиваются одновременно, то они **комбинированные**.

Если между процессами возникают связи всевозможных типов (функциональные, пространственно-временные, управляющие, информационные и т.д.), то процессы являются **взаимосвязанными**.

Процесс, по требованию которого образуется другой процесс, называется **порождающим**. Образованный процесс называется **порожденным**. Управляющий тип связи определяется отношением порождающего к порождаемому.

Если два процесса совместно используют некоторые ресурсы, но не обмениваются информацией, они называются **информационно**-**независимыми**. При наличии информационной связи, они являются взаимодействующими.

Если необходимо акцентировать внимание на совместном использовании двумя процессами одного и того же ресурса, их называют **конкурирующими**.

Принадлеж

ность к ЦП

Внутренние

Внешние

**Принадлежность к ОС**

Системные

Пользовательские

**Динамический признак**

Последователь

ные

Параллельные

Комбинирован

ные

###### Связность

Изолированные

Информационно независимые

Взаимодействующие

Конкурирующие

**Результатив**

**ность**

Различные

Эквивалентные

Тождественные

Равные

Генеалоги

ческий признак

Порождающие

Порожденные

##### Процессы

**Рис.2.** Схема классификации процессов

Допустимые отношения между взаимосвязанными процессами определяются тремя синхронизирующими правилами:

1. Отношение **предшествования**. Первый из двух процессов должен переходить в активное состояние всегда раньше второго.
2. Отношение **приоритетности**. Процесс, имеющий приоритет **P**, активизируется при соблюдении двух условий:
* в состоянии готовности нет процессов с большим приоритетом;
* процессор либо свободен, либо используется менее приоритетным, чем **Р** процессом.

3. Отношение **взаимного исключения**. Если два процесса используют общий ресурс, допускающий только последовательное использование (**критический ресурс**), то действия одного процесса над этим ресурсом не должны выполняться одновременно с действиями другого процесса. Сами эти действия процесса над критическим ресурсом называются **критической областью**.

# Управление процессами

Процесс – программа в стадии выполнения. С каждым процессом связывается его адресное пространство: список адресов от минимума до максимума, который процесс может прочесть и в которые он может писать. Адресное пространство: сама программа, данные к ней, ее стек.

Со всяким процессом связывается некий набор регистров, включая счетчик команд, указатель стека и др. + вся необходимая информация для запуска программы, кроме регистров.

В период своего существования процесс проходит через ряд дискретных состояний:

- выполняется, если в данный момент времени ему выделен ЦП

- готов, если он мог бы сразу использовать ЦП, предоставленный в его распоряжение

- блокирован, если он ожидает появление какого-либо события

Когда в систему поступает некоторое задание, она создает соответствующие процессы, которые затем устанавливаются в конец списка готовых процессов. Этот процесс постепенно продвигается к головной части списка по мере завершения выполнения предыдущих процессов. Когда процесс окажется первым в списке готовых и когда освобождается ЦП, этому процессу выделяется ЦП, и говорят что происходит смена состояния процессов (переходит из состояния готовности в состояние выполнения). Чтобы предотвратить либо случайный, либо умышленный монопольный захват ресурсов компьютера каким-то одним процессом, ОС устанавливает в специальном аппаратном таймере прерываний временной интервал, в течении которого любому процессу разрешается занимать ЦП. Если процесс добровольно не освобождает ЦП в течение указанного временного интервала, таймер вырабатывает сигнал прерывания, по которому управление будет передано ОС. После этого ОС переведет ранее выполнявшийся процесс в состояние готовности, а первый процесс списка готовых – в состояние выполнения.

Представителем процесса в ОС является блок управления процессом (дескриптор процесса). Это структура данных, содержащая: идентификатор процесса, приоритет процесса, состояние, указатели памяти, указатели выделенных ресурсов. Область сохранения регистров.

Когда ОС переключает ЦП с процесса на процесс, она использует области сохранения регистров, предусмотренные в блоке управления процессом, чтобы запомнить ? информацию, необходимую для рестарта каждого процесса, необходимую при следующем обращении процесса к ЦП.

ОС предусматривает следующие операции над процессами:

- Создание/образование

- Уничтожение

- Возобновление

- Блокирование

- Пробуждение

- Запуск/выбор

### Создание процесса:

- присвоение имени

- включение этого имени в список имен известных ОС

- определение начального приоритета процесса

- формирование блока управления процесса

- выделение процессу начальных ресурсов

Любой процесс может породить процесс

Диаграмму состояния можно представить в виде:



Итоги понятия процесса. Понятие процесса базируется на двух независимых концепция: группирование процессов и выполнение программы.

С одной стороны, у процесса есть адресное пространство, содержащее текст программы и данные, а также другие ресурсы.

Такие как:

- открытые файлы

- дочерние процессы

- обработчики сигналов

- необработанные аварийные сообщения

- учетная информация и др.

С другой стороны, процесс можно рассматривать как поток исполняемых команд. У потока есть счетчик команд, регистры, в которых хранятся текущие переменные, стек, содержащий протокол выполнения процесса. Хотя поток должен исполняться внутри процесса, следует различать концепции потока и процесса. Процесс используется для группирования ресурсов, а потоки являются объектами поочередно исполняющимися на ЦП.

Концепция потока добавляет процессу возможность выполнения в одной и той же среде процесса нескольких программ, в достаточной степени независимых.

Приостановленный процесс состоит из собственного адресного пространства, обычно называемого образом памяти или core-файлом.

### Понятие ядра ОС.

Все операции, связанные с процессами выполняются под управлением той части ОС, которая называется ядром. Ядро представляет собой небольшую часть кода ОС в целом. Однако оно относится к числу наиболее интенсивно используемых компонент системы. По этой причине ядро обычно резидентно размещается в ОП, в то время как другие части ОС перемещаются во внешнюю память и обратно по мере необходимости. Ядро ОС, как правило, содержит программы для реализации следующих функций:

- обработка прерываний

- создание и уничтожение процессов

- переключение процессов из состояния в состояние

- диспетчирование процессов

- приостановка и активизация процессов

- синхронизация процессов

- организация взаимодействия между процессами

- поддержка операций ввода/вывода

- поддержка распределения или перераспределения памяти

- поддержка работы файловой системы

- поддержка механизма вызова-возврата при обращении к процедурам

- поддержка определенных функций по учету работы ЭВМ

# Процессы.

Понятие процесса относятся к одному из основополагающих в ОС. Существует много определений термина процесс в том числе:

1. Выполняющаяся прикладная программа пользователя
2. Экземпляр программы, выполняющийся на компьютере
3. Объект, который можно индетифицировать и выполнять на процессоре
4. Единица активности, которую можно охарактеризовать единой цепочкой последовательных действий, текущим состоянием и связанных с ней набором системных ресурсов.

Как понятие процесс является определенным типом абстракции, и обычно следует придерживаться следующего неформального определения.

Последовательный процесс (задача) – выполнение отдельной программы и ее данные на последовательном процессоре.

В качестве примера можно назвать следующие процессы:

1. Выполняющаяся прикладная программа пользователя
2. Утилит
3. Трансляция программ
4. Компоновка, выполнение

Определение понятия процесс ставит цель выработать механизм распределения и управления ресурсами. Понятие ресурс как и понятие процесса является основным при рассмотрении ОС. Термин ресурс применяется к повторно используемым, относительно стабильным и часто недостающем объектом, которые запрашиваются, используются и освобождаются процессами в период их активности, т.е. ресурсом называется всякий объект, который может распределяться внутри системы.

Мысленно процесс можно разделить на 3 компонента:

1. Выполняющаяся программа
2. Данные нужные для работы
3. Контекст выполняющейся программы (execution context) или состояние процесса (process state) Включат в себя всю информацию нужную ОС для управления процессами и процессору для его выполнения. Данные, характеризующие это состояние, включают в себя содержимое различных регистров процессора, таких как программный счетчик и регистры данных, приоритет процесса и сведения о том, находится ли данный процесс в состоянии ожидания вв-выв.

1-список процессов

## А-процесс а

В-процесс В

2-индекс процесса

3-PC счётчик команд

4-базовый регистр

5-граничный регистр

На рис. Показан пример реализации процесса. Два процесса А и В находятся в различных областях основной памяти. Каждому процессу отведён блок памяти, в к-ом создаётся код программы, данные и информация о состоянии процесса. Каждый процесс заносится в список процессов, к-ый создаётся и поддерживается OC. Часть этого списка, соответствующая определённому процессу, содержит указатель размещения этого процесса в памяти. В регистре индекса процесса содержится индекс выполняющегося в текущий момент времени процесса, идентифицирующий его в списке процессов. Содержимое команд (PC) указывает на очередную инструкцию, к-ую нужно выполнить. Базовый и граничный регистры задают область памяти, занимаемую процессом. В базовый регистр заносится адрес начальной ячейки памяти, а в граничный- её размер в байтах.

Содержимое програмного счётчика и всех ссылок на данные отсчитывается от значения базового регистра. По своей величине эти ссылки не могут превосходить значения граничного регистра. Регистр индекса процесса показывает, что выполняется процесс В, до этого выполняется процесс А, но он временно прерван. Содержимое всех регистров в момент прекращения этого процесса записано в виде данных о состоянии процесса. Впоследствии ОС сможет к выполнению процесса А. При этом будет сохранён контекст выполнения процесса В и восстановлен контекст процесса А. Когда в програмный счётчик загрузится значение, указывающее на область кода, программа процесса Аавтоматически возобновиться выполнение этого процесса. Таким образом процесс реализируется в виде структуры данных. Он может выполнятьсяили находиться в состоянии ожидания. Состояние процесса в каждый момент времени заносится в специально отведённую область данных. Использование структуры позволяет развить мощные методы координации и взаимодействия процесса. Расширяя и добавляя данные о состоянии процесса доп. информации можно разработать новые возможности ОС. В обычных ОС общего назначения процес может находиться в активном и пассивном состоянии.

В активном состоянии процесс участвует в конкуренции за исп-ие ресурса выч системы, а в пассивном- он только известен системе, но в конкуренции не участвует. В свою очередь активный процесс м/б в одном из следующих состояний:

1. *Выполнение*- все затребованные процессом ресурсы выделены. В этом состоянии в каждый момент времени может находиться только один процесс (В однопроцессорной выч системе)

2. *Готовность к выполнению*- ресурсы м/б предоставлены, тогда процесс перейдёт в состояние выполнения

3.*Блокирование или ожидание*- затребованные ресурсы не м/б предоставлены или не завершена операция вв/выв.

В большенстве ОС последнее состояние подразделяется на мн-во состояний ожиданий, соответствующих определённому виду ресурсов из-за отсутствия которого процесс переходит в заблокированное состояние.

В обычных ОС процесс появляется при запуске какой-нибудь программы ОС пораждает для нового процесса соответствующий дескриптор процесса и процесс начинает выполняться. Поэтому пассивного состоянияне существует. В ОС реального времени уже заранее бывает известен состав программ которые должны выполняться. Известны и многие их параметры которые необходимо учитывать при распределении ресурса. Например: объём памяти, приоритет, средняя длительность выполнения, открываемые файлы, исп-ые устройства и т. д. Поэтому для них заранее заводят дискрипторы процесса, чтобы в последствии не тратить время на организацию дискрипторов таким образом в ОСРВ многие процессы находятся в состоянии бездействия.

Процесс состояния бездействия может перейти в состояние готовности в следующих случаях:

1. По команде оператора.

2. При выборе из очереди планировщиком.

3. По вызову из другой задачи. Один процесс может создать, инециировать, приостановить, уничтожить другой процесс.

4. По прерыванию от внешнего инециативного устройства. (Сигнал о завершении некоторого события может запускать соответствующий процесс).

5. При наступлении запланированного времени запуска программ.

Последние два способа запуска характерны для ОСРВ (реал. времени).

Из состояния выполнения процесс может выйти по одной из следующих причин:

1)Процесс завершается, при этом он передаёт управление ОС и сообщает о своём завершении.

2)Процесс переводится ОС в состояние готовности к выполнению в связи с появлением более приоритетной задачи и в связи с окончанием выделенного ему кванта времени.

3)процесс блокируется (переводится в состояние ожидания) либо в силу невозможности предоставить ему ресурс, запрошенный в настоящий момент. При наступлении соответствующего события (завершилась операция вв/выв, освободился затребованный ресурс в опер. памяти, загружена необходимая страница виртуальной памяти и т. д. ) процесс деблокируется и переводится в состояние готовности к исполнению. Таким образом движущей силой, меняющей состояние процесса является событие. Один из ост. видов- прерывание.

Для того чтобы ОС могла управлять процессами, она должна распологать необходимой для этого информацией. С этой целью на каждый процесс заводится специальная специальная информационная структур, называемая диструктором процесса. (описатель задачи, блок управления задачей) Деструктор содержит следующую информацию:

1. Идентификатор процесса (Process Identificator (ID))

2. Тип или класс процесса, который определяет для ОС некоторые правила предоставления ресурсов.

3. Приоритет процесса. В соответствии с которым ОС предоставляет ресурсы В рамках одного класса процессов в первую очередь обслуживается более приоритетный процесс.

4. Переменную состояния, которая определяет в каком состоянии находится процесс (готов кработе, состояние выполнения, ожидание устройства ввода/вывода и т. д.)

5. Защищённую область памяти в которой хранится текущее значение регистров процессора, если прерывается не закончив работу. Эта информация называется контекстом процесса(задачи).

6. Информацию о ресурсах, которыми процесс владеет и имеет право пользоваться (указатели на открытые файлы, инфа о независимых операциях вв/выв и т. т.)

7. Место памяти или адрес этого места для организации общения с другими процессами.

8. Параметры времени запуска. (момент времени, когда процессор должен активизироваться и периодичность этой операции).

Дескрипторы процесса постоянно находятся в опер. памяти, чтобы ускорить работу ОС, которая организует их в списки (очереди) и отображает изменение состояния процесса, перемещением соответствующего дескриптора из одного списка в другой. В некоторых ОС количество дескрипторов определяется жёстко и заранее на этапе генерации варианта ОС или в конфигурационном файле, который исполняется при загрузке ОС. В других ОС по мере необходимости система выделяет участки памяти под новые дескрипторы.

Для аппаратной поддержки работы ОС с дескрипторами задач процессора реализованы соответствующие механизмы. Например процессора intel (i80\*86) имеется специальный регистр Task Register (TR) указывающий местонахождение сигмента состояния задачи Task State Segment (TSS) вкотором при переключении с процесса на процесс автоматически сохраняется содержимое регистров процессора.

# ПОТОКИ

Концепцию процесса можно охарактеризовать двумя параметрами:

1. Владение ресурсами.

Процесс включает виртуальное адресное пространство в котором содержится образ процесса и время от времени может владеть такими ресурсами, как основная память, устройство вв/выв, файлы. ОС выполняет защитные функции предотвращая нежелательные взаимодействия процессов.

2. Планирование и выполнение.

Выполнение процесса осуществляется путём выполнения кода программы при этом выполнение процесса может чередоваться с выполнением других процессов. Таким образом процесс имеет такие параметры как состояние и текущий приоритет, в соответствии с которым ОС осуществляет его планирование и диспетчеризацию.

В большинстве ОС эти две характеристики являются сущностью процесса. Однако они являются независимыми и ОС может рассматривать их отдельно друг от друга. Чтобы различать эти две характеристики единицу диспетчеризации называют потоком (термин нить, волокно), а единицу владения ресурсами – процессом или задачей. Многопоточностью называется способность ОС поддерживать в рамках одного процесса выполнение нескольких потоков.

Традиционный подход, при котором каждый процесс представляет собой единый поток выполнения называется однопоточным. Например MS-DOS.

В однопоточной модели процесса, в его представление входит управляющий блок этого процесса и пользовательского адресного пространства, а также стеки ядра и пользователя, с помощью которого осуществляются вызовы процедур и возвраты из них при выполнении процесса, когда выполнение процесса прерывается, содержимое регистра процессора сохраняе-тся в памяти, в много-пользовательской сре-де с каждым процесс-сом также связан уп-равляющий блок и адресное прост-ранство, но теперь для каждого потока соз-даются свои отдельные стеки, а также свой уп-равляющий блок, в котором содержатся значения регистра, приоритет и другая информация о сос-тоянии потока.

Таким образом все потоки процесса разделены между собой состоянием и ресурсами этого процесса, они нахо-дятся в одном и том же адресном пространстве и имеют доступ к од-ним и тем же данным. Если один поток изме-няет в памяти какие-то данные, то другой поток, во время этого доступа к этим данным , может отследить эти изменения. Если один поток открывает файл с правом чтения, то другие потоки данного процесса могут читать данные из этого файла. Основные преимущества использования потоков:1)создание нового потока в уже существующем процессе занимает меньше времени, чем создание нового процесса2) поток можно завершить быстрее чем процесс3)переключение потоков в рамках одного и того же процесса происходит быстрее4)при использовании потока повышается эффективность обмена информацией между двумя выполняющимися программами. В большинстве о.с. обмен между независимыми процессами происходит с участием ядра, в функции которого входит обеспечение защиты и механизма необходимого для осуществления обмена. Однако благодаря тому, что различные потоки одного и того же процесса используют одну и ту же область памяти и одни и те же файлы, они могут обмениваться информаций без участия ядра.

Таким образом если приложение или функцию надо реализовать в виде набора взаимосвязанных модулей, намного эффективнее реализовать ее в виде потока, чем в виде набора отдельных процессов. Пример программы, где удачно применяются потоки является файловый сервер. При получении каждого нового файлового запроса программа, управляющая файлами, порождает новый поток, из-за того, что серверам приходится обрабатывать очень большое количество запросов за короткий промежуток, будут создаваться и удалятся много потоков. Если такая серверная программа работает на многопроцессорной машине, то на разных процессорах в рамках одного процесса могут одновременно выполнятся несколько потоков, кроме того из-за того , что процессы или потоки файлового сервера должны использовать совместно данные из файлов , рациональнее использовать потоки и общую область памяти, а не процессы и обмен сообщениями. Потоковая конструкция процессов полезна и на однопроцессорных машинах, так как она помогает упростить структуру программы, выполняющую несколько логически различных функций.