§5 Понятие процесса в ОС

5.1 Понятие процесса. Блок управления процессом. Операции над процессом.

5.2 Классификация процессов.

5.3 Схема состояний процесса.

5.4 Планирование и взаимодействие процессов.

5.5 Планирование операций над процессом. Стратегии планирования.

5.6 Прерывания. Механизм обработки прерываний.

5.1 Понятие процесса

Процесс – это программный модуль, выполняемый в текущий момент времени.

Действия ОС по управлению процессами:

-Создание и удаление
-Планирование
-Синхронизация
-Коммуникация
-Разрешение тупиковых ситуаций

Программа – план действий

Процесс – действие

Процесс, связан с программным кодом исполняемого модуля и рассматривается ОС как заявка на потребление всех видов ресурсов, кроме процессорного времени.

Поток представляет собой, способ распараллеливания вычислений. Поток – последовательность команд, выполняемых процессором.

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

Блок управления процессом (Process Control Block)

Для того чтобы операционная система могла выполнять операции над процессами, каждый процесс представляется в ней некоторой структурой данных.

Эта структура содержит информацию, специфическую для данного процесса:

- состояние, в котором находится процесс ;
- программный счетчик процесса или, другими словами, адрес команды, которая должна быть выполнена для него следующей;
- содержимое регистров процессора;
- данные, необходимые для планирования использования процессора и управления памятью (приоритет процесса, размер и расположение адресного пространства, какие ресурсы нужны и на какое примерное время, допускается ли совместно использовать ресурс с другими процессами или нет и т. д.);
- учетные данные (идентификационный номер процесса, какой пользователь инициировал его работу, общее время использования процессора данным процессом и т. д.);
- сведения об устройствах ввода-вывода, связанных с процессом (например, какие устройства закреплены за процессом, таблицу открытых файлов).

Блок управления процессом является моделью процесса для операционной системы.

Информацию, для хранения которой предназначен блок управления процессом, удобно для дальнейшего изложения разделить на две части.

Содержимое всех регистров процессора (включая значение программного счетчика) будем называть *регистровым контекстом* процесса, а все остальное – *системным контекстом* процесса.

Знания регистрового и системного контекстов процесса достаточно для того, чтобы управлять его работой в операционной системе, совершая над ним операции.

С точки зрения пользователя, наоборот, наибольший интерес представляет содержимое адресного пространства процесса, возможно, наряду с регистровым контекстом определяющее последовательность преобразования данных и полученные результаты.

Код и данные, находящиеся в адресном пространстве процесса, будем называть его *пользовательским контекстом*.

Совокупность регистрового, системного и пользовательского контекстов процесса для краткости принято называть просто *контекстом процесса*.

В любой момент времени процесс полностью характеризуется своим контекстом.

Операции над процессами

- создание процесса – завершение процесса ;
- приостановка процесса (перевод из состояния исполнение в состояние готовность);
- запуск процесса (перевод из состояния готовность в состояние исполнение);
- блокирование процесса (перевод из состояния исполнение в состояние ожидание);
- разблокирование процесса (перевод из состояния ожидание в состояние готовность).

5.2 Классификация процессов

1. По временным характеристикам:

- *Интерактивные (*время существования интерактивного процесса определяется реакцией ЭВМ на запрос обслуживания и составляет секунды).
- *пакетные (*запускаются один вслед за другим и время реакции часы и более).
- *процессы реального времени (*имеют гарантированное время окончания работы и время реакции мсек).

2. По генеалогическому признаку: *порождающие и порожденные процессы.*

3. По результативности:

-*эквивалентные (*реализовываются как на одном, так и на многих процессорах по одному или разным алгоритмам, то есть они имеют разные трассы, которые определяют порядок и длительность пребывания процесса в разных состояниях)

- *тождественные (*реализуются по одной и той же программе, но имеют разные трассы).

- *равные процессы (*реализуются по одной программе и имеют одинаковые трассы).

Все они имеют одинаковый конечный результат, но эквивалентные процессы могут реализовываться как на одном, так и на нескольких процессорах по одному или разным алгоритмам, т.е. имеют разные трассы, которые определяют порядок и длительность процесса в разных состояниях.

4. По времени развития: *последовательные, параллельные и комбинированные* (для последних есть точки, в которых существуют оба процесса, и точки, в которых существует только один процесс).

5. По месту развития: *внутренние* (реализуются на центральном процессоре) и *внешние* (реализуются на внешних процессорах).

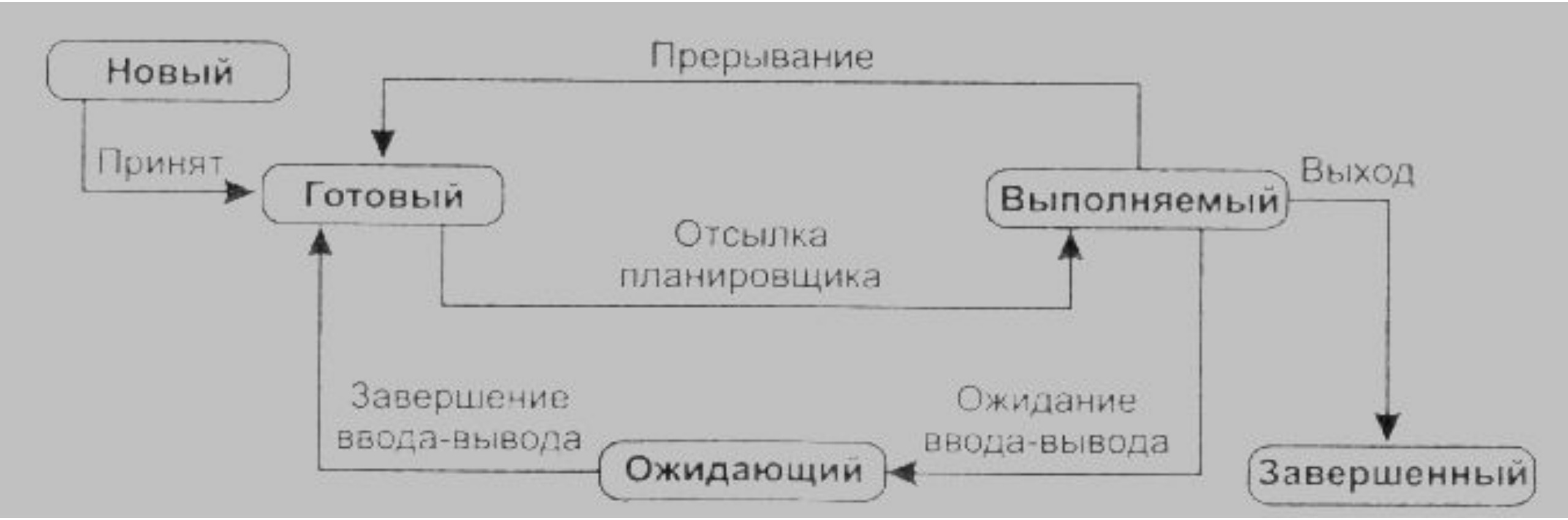
6. По принадлежности к операционной системе: *системные* (исполняют программу из состава операционной системы) и *пользовательские.*

7. По связности различают процессы:

- *взаимосвязанные,* которые имеют какую-то связь (пространственно-временную, управляющую, информационную);
- *изолированные* — слабо связанные;
- *информационно-независимые*, которые используют совместные ресурсы, но имеют собственные информационные базы;
- *взаимодействующие* — имеют информационные связи и разделяют общие структуры данных;
- *взаимосвязанные по ресурсам;*
- *конкурирующие.*

Порядок взаимосвязи процессов – **отношения между процессами**:
- предшествования — один всегда находится в активном состоянии раньше, чем другой;
- приоритетности — когда процесс может быть переведен в активное состояние только в том случае, если в состоянии готовности нет процессов с более высоким приоритетом, или процессор свободен, или на нем реализуется процесс с меньшим приоритетом;
- взаимного исключения - в процессе используется общий критический ресурс, и процессы не могут развиваться одновременно: если один из них использует критический ресурс, то другой находится в состоянии ожидания.

5.3. Схема состояний процесса (классическая)



Каждый процесс в ВС, проходит определенную схему состояний.

Для появления в вычислительной системе процесс должен пройти через состояние **Новый** (рождение). В этом состоянии формируется блок управления процессом (PCB), процесс получает в свое распоряжение адресное пространство, в которое загружается программный код процесса ; ему выделяются стек и системные ресурсы; устанавливается начальное значение программного счетчика этого процесса и т. д.

После выделения ОП, процесс переводится в состояние **Готовый**. В этом состоянии процессу присваивается приоритет (число, характеризующее привилегию процесса при выделении ресурсов), определяется алгоритм обработки процесса и процесс определяется в очередь к нужному ресурсу (ЦП).

При выделении требуемого ресурса процесс переходит в состояние **Выполняемый.** В этом состоянии происходит непосредственное выполнение программного кода процесса. Выйти из состояния выполнения процесс может по трем причинам:

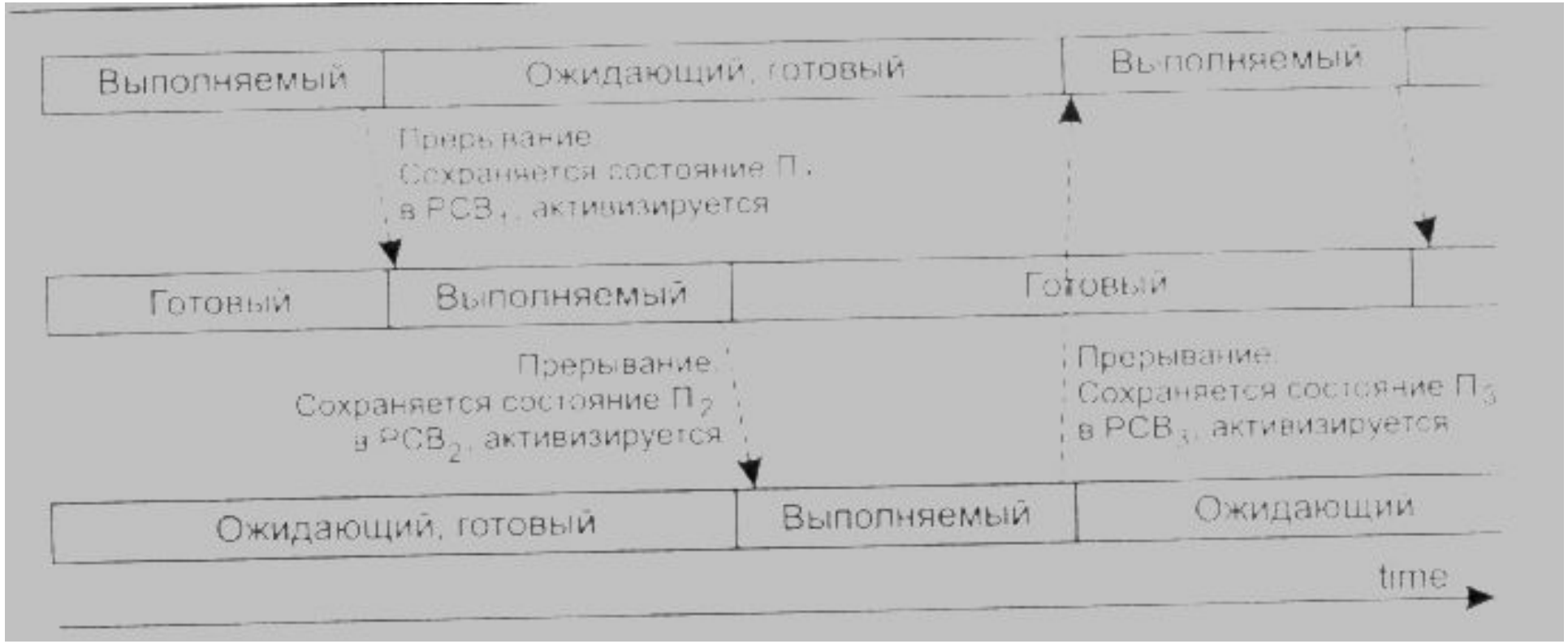
- операционная система прекращает его деятельность;
- он не может продолжать свою работу, пока не произойдет некоторое событие, и операционная система переводит его в состояние **Ожидающий** (процесс ожидает наступления некоторого события, чаще всего операций ввода-вывода, затем переходит в состояние готовый);
- в результате возникновения прерывания в вычислительной системе (например, прерывания от таймера по истечении предусмотренного времени выполнения, поступление высокоприоритетного процесса и т.д.) его возвращают в состояние готовность.

Если все требуемые процессом ресурсы предоставились, то процесс переходит в состояние **Завершенный** (завершил работу).

Для мультипрограммных вычислительных систем псевдопараллельная обработка нескольких процессов достигается с помощью переключения процессора с одного процесса на другой.

Пока один процесс выполняется, остальные ждут своей очереди.

Представление процесса в ОС –
таблица управления процессом
(ТУП – PCB – process control blok)



5.4 Планирование и взаимодействие процессов

Вопрос (о планировании)	Ответ
1. Как называется действие: распределение процессов между имеющимися ресурсами?	Планирование процессов
2. Что такое **«метод очередей ресурсов»**? Опишите суть данного метода.	Метод планирования процессов, ориентированный на эффективную загрузку ресурсов. Суть метода в том, что в определенных состояниях образуется очередь процессов к ресурсам. В состоянии новый - процессы ждут выделения ОП, в состоянии готовности процессы выстраиваются в очередь к процессору, в состоянии ожидания – в очередь к устройствам ввода-вывода или к данным.
3. Где располагаются и чем связаны готовые к выполнению процессы?	Основная память, очередь готовых процессов
4. Как называется программа, управляющая перемещением процесса между очередями?	планировщик
5. Долгосрочный планировщик решает -	Какой из процессов, находящийся во входной очереди, должен быть переведен в очередь готовых процессов в случае освобождения ресурсов памяти
6. Краткосрочный планировщик решает -	Какой из процессов, находящийся во очереди готовых процессов, должен быть передан на управление ЦП
7. В чем заключается основное отличие между разными видами планировщиков	Частота запуска: краткосрочный – каждые 100мс, долгосрочный 1 раз в неск.мин

Вопрос (о взаимодействии)	Ответ
8. В каких отношения могут находится выполняемы процессы?	Независимы, взаимодействующие
9. По какой схеме описывают взаимодействие процессов?	«Производитель-потребитель»
10. В чем **суть взаимодействия** процессов?	Передача данных между процессами или совместное использование ресурсов
11. Перечислите механизмы взаимодействия процессов: ?	Транспортер(канал), очередь, сигнал, семафор
12. Опишите транспортер:	-средство взаимодействия родственных процессов -область памяти с файловой организацией, для которой обеспечены запись и считывание данных -порядок записи данных неизменен, не допускается повторное считывание -размер задает вызвавший процесс, -дочерние процессы могут использовать родительский транспортер
13. Опишите очередь:	-обеспечивает передачу или использование общих данных без перемещения данных, с передачей элемента очереди, содержащего указатель данных и объем массива данных -чтение элементов только создающий процесс, остальные – запись в очередь -чтение с уничтожением или без из очереди -доп.функции: определение количества элементов в очереди в текущий момент, очистка очереди создавшим процессом

Вопрос (о взаимодействии)	Ответ
14. Опишите сигнал:	-передача требований одного процесса к другому на немедленное выполнение действия -действия: обработка системной ошибки при появлении сигнала, блокирование сигнала, передача управления подпрограмме
15. Опишите семафор:	-передача сообщений от одного потока к другому о наступлении некоторого события -виды: системные (ОС контролирует завершение каждого процесса, владеющего системным семафором) оперативной памяти (устанавливаются в определенное состояние, не обслуживаются ОС, ОС не сообщает об их освобождении или захвате) - Функции для управления семафорами: установка с целью сигнализации, ожидание вызывающим потоком пока не будет выключен, ожидание потоком выключения одного из нескольких семафоров

Пример: интерфейс межпрограммного взаимодействия **– буфер обмена**

5.5 Планирование операций над процессами

Планирование работы планировщика.

Критерии сравнения краткосрочных планировщиков:

1.Утилизация CPU (использование процессора)
   0-100% - теоретически
   40% - легко загружен
   90% - тяжело загружен

2. Пропускная способность CPU (кол-во выполняемых процессов в единицу времени)

3. Время оборота (время выполнения: входная очередь – завершение)
4. Время ожидания (суммарное время нахождения процесса в очереди готовых процессов)
5. Время отклика (время между входной очередью до первого обращения)

Стратегии планирования работы процессора

1. **Первый пришел – первый обслуживается, FIFO – first come – first served (FCFS)**
   - велико среднее время ожидания

2.«наиболее коротка работа выполняется первой», SJF - Shortrest Job First
- трудность за ранее определить величину времени последующего обслуживания

3.Приоритетное планирование
- каждому процессу присваивается приоритет , определяющий очередность

Приоритет – целое положительное число, некоторого диапазона
Факторы, влияющие на назначение приоритета:
Внешние:
- Требования к памяти
- Количество открытых файлов
- Отношение среднего времени ввода-вывода к среднему времени использования ресурса CPU

Внутренние:
- Важность процесса
- Тип и величина файлов, используемых для оплаты
- Отделение, выполняющее работы

Недостаток: блокирование на неопределенно долгое время низкоприоритетных процессов
Выход: автоматическое повышение приоритета после некоторого времени ожидания

4.«карусельная» стратегия планирования (RR-Round Robin)

- Применяется в системах разделения времени
- Циклическое перемещение процессов, CPU используется в течение 1 кванта времени t = 10…100мс
- Производительность процессора 1/N

5.Планирование с использованием многоуровневой очереди (Multi-level queue scheduling)

- Для процессов, классифицируемых на группы (интерактивные и пакетные (фоновые)
- Правило: ни один процесс с более низким приоритетом не может быть запущен, пока не выполнятся процессы во всех очередях с более высоким приоритетом

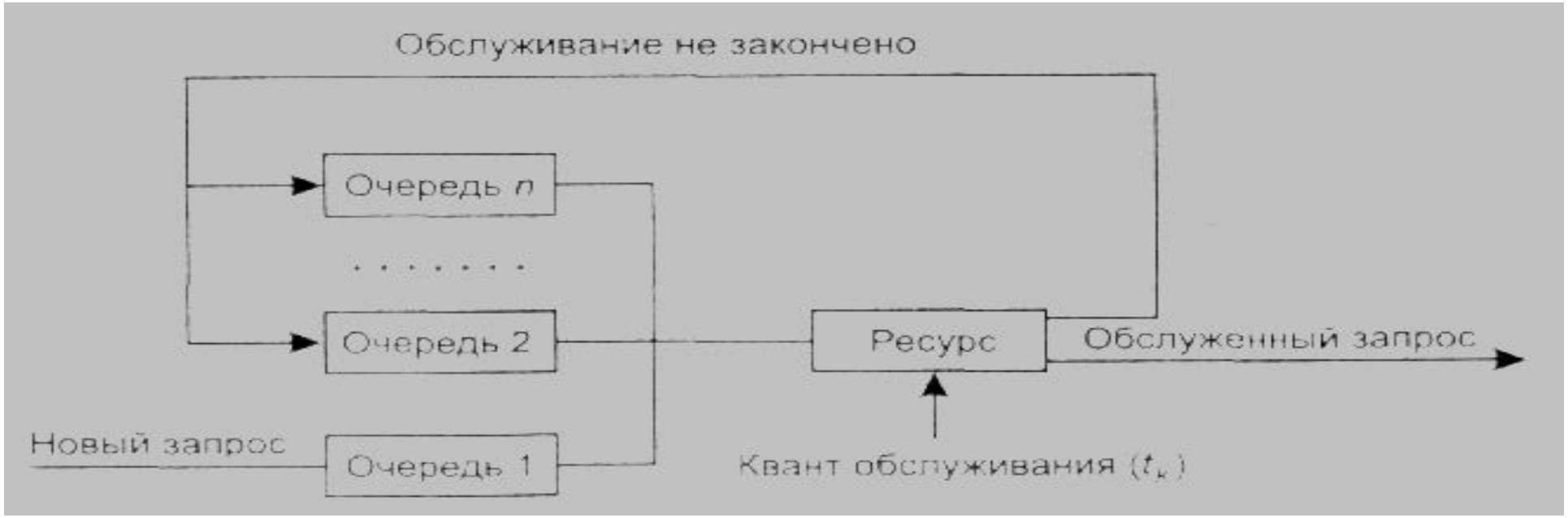
6.Многоуровневая очередь с обратными связями (multilevel feedback queue sheduling) - предполагает, что процессы при определенных условиях могут перемещаться между очередями.

- организуется *N* очередей.
- все новые запросы поступают в конец первой очереди.
- первый запрос из *i*-й очереди поступает на обслуживание лишь тогда, когда все очереди от 1-й до *i* — 1-й пустые.
- на обслуживание выделяется квант времени t_k. Если за это время обслуживание запроса завершается полностью, то он покидает систему. В противном случае недообслуженный запрос поступает в конец *i* + 1-й очереди
- после обслуживания запроса из *i*-й очереди система выбирает для обслуживания запрос из непустой очереди с самым младшим номером.

Таким запросом может быть следующий запрос из очереди *i* или из очереди *i* + 1 (при условии, что после обслуживания запроса из очереди *i* последняя оказалась пустой). Новый запрос поступает в 1-ю очередь (*i* = 1). В такой ситуации после окончания времени t_k, выделенного для обслуживания запроса из очереди *i*, будет начато обслуживание запроса первой очереди. Если система выходит на обслуживание заявок из N-й очереди, то они обслуживаются либо по дисциплине FIFO (каждая заявка обслуживается до конца), либо по циклическому алгоритму. Данная система наиболее быстро обслуживает все короткие по времени обслуживания запросы.

Недостаток системы заключается в затратах времени на перемещение запросов из одной очереди в другую.

Данная стратегия является универсальной и **сочетает:** FCFS, SJF, приоритетная, Round Robin, многоуровневая очередь.


Обслуживание не закончено
Очередь n
Очередь 2
Очередь 1
Новый запрос
Ресурс
Обслуженный запрос
Квант обслуживания ($t_к$)

7.Приоритетная многоочередная дисциплина обслуживания

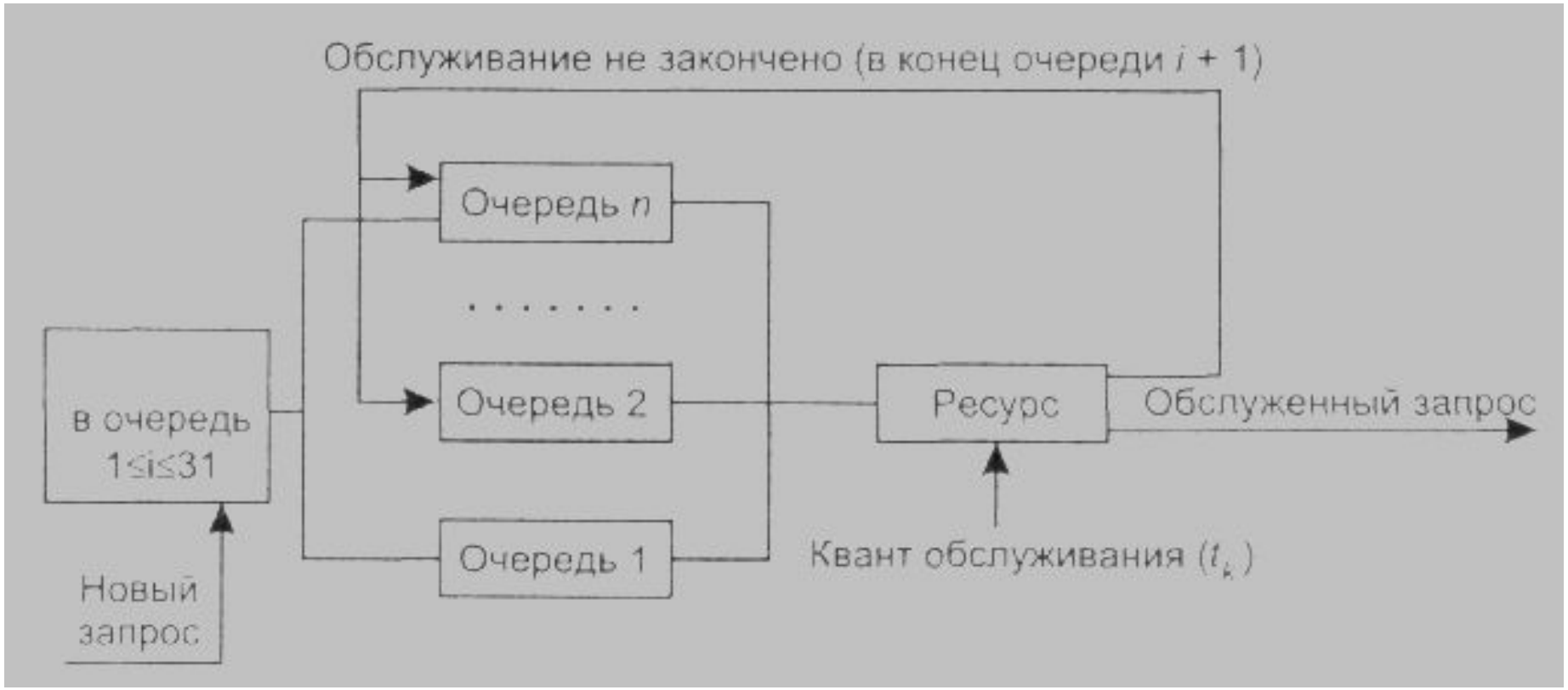
- вновь поступающие в систему запросы устанавливаются не обязательно в 1-ю очередь, а в очередь в соответствии с имеющимися приоритетами, которые определяются параметрами обслуживания процессов.
- приоритетные многоочередные дисциплины обслуживания могут использовать обслуживание с абсолютным и относительным приоритетом.
- при обслуживании с абсолютным приоритетом приоритет определяется номером очереди, и первыми обслуживаются запросы, обладающие наивысшим приоритетом (из очереди с меньшим номером запрос из очереди *i* — 1 будет прерывать обработку запроса из очереди *i*).

Недостатки:

- увеличивается степень дискриминации по среднему времени ожидания в очереди между высоко- и низкоприоритетными запросами.

Достоинства:

- время ожидания высокоприоитетных заявок сокращается, но за счет большей задержки в обслуживании низкоприоритетных заявок.
- обслуживание с относительным приоритетом не вызывает прерывания обслуживаемой заявки до ее завершения, даже если она менее приоритетная


Обслуживание не закончено (в конец очереди i + 1)
Очередь n
. . . . . .
Очередь 2
Очередь 1
в очередь
1≤i≤31
Новый
запрос
Ресурс
Обслуженный запрос
Квант обслуживания (t_k)

5.6 Прерывания.

Прерывания представляют собой механизм, позволяющий координировать параллельное функционирование отдельных устройств вычислительной системы и реагировать на особые состояния, возникающие при работе процессора.

Основная цель введения прерываний- реализация асинхронного режима функционирования и распараллеливание работы отдельных устройств вычислительного комплекса.

Прерывание — это принудительная передача управления от выполняемой программы к системе (а через нее — к соответствующей программе обработки прерывания), происходящая при возникновении определенного события.

Виды прерываний

Внешние прерывания вызываются асинхронными событиями, которые происходят вне прерываемого процесса, например:

- прерывания от таймера;
- прерывания от внешних устройств;
- прерывания по нарушению питания;
- прерывания с пульта оператора вычислительной системы;
- прерывания от другого процессора или другой вычислительной системы.

Внутренние прерывания вызываются событиями, которые связаны с работой процессора и являются синхронными с его операциями.

Примерами являются следующие запросы на прерывания:

- при нарушении адресации *(в адресной части выполняемой команды указан запрещенный или несуществующий адрес, обращение к отсутствующему сегменту или странице при организации механизмов виртуальной памяти);*
- при делении на ноль;
- от средств контроля *(например, вследствие обнаружения ошибки четности, ошибок в работе различных устройств);*
- при неправильной работе программ *(программные прерывания)*
- программные прерывания *(нарушение работы программ)*

Механизм обработки прерываний.

1. Установление факта прерывания и идентификация прерывания.
2. Запоминание состояния прерванного процесса вычислений.
3. Управление аппаратно передается на подпрограмму обработки прерывания.
4. Сохранение информации о прерванной программе, которую не удалось спасти с помощью аппаратуры.
5. Выполнение программы, связанной с обработкой прерывания.
6. Восстановление информации, относящейся к прерванному процессу.
7. Возврат на прерванную программу.

Функции механизма прерываний:

- распознавание или классификация прерываний;
- передача управления соответствующему обработчику прерываний;
- корректное возвращение к прерванной программе.

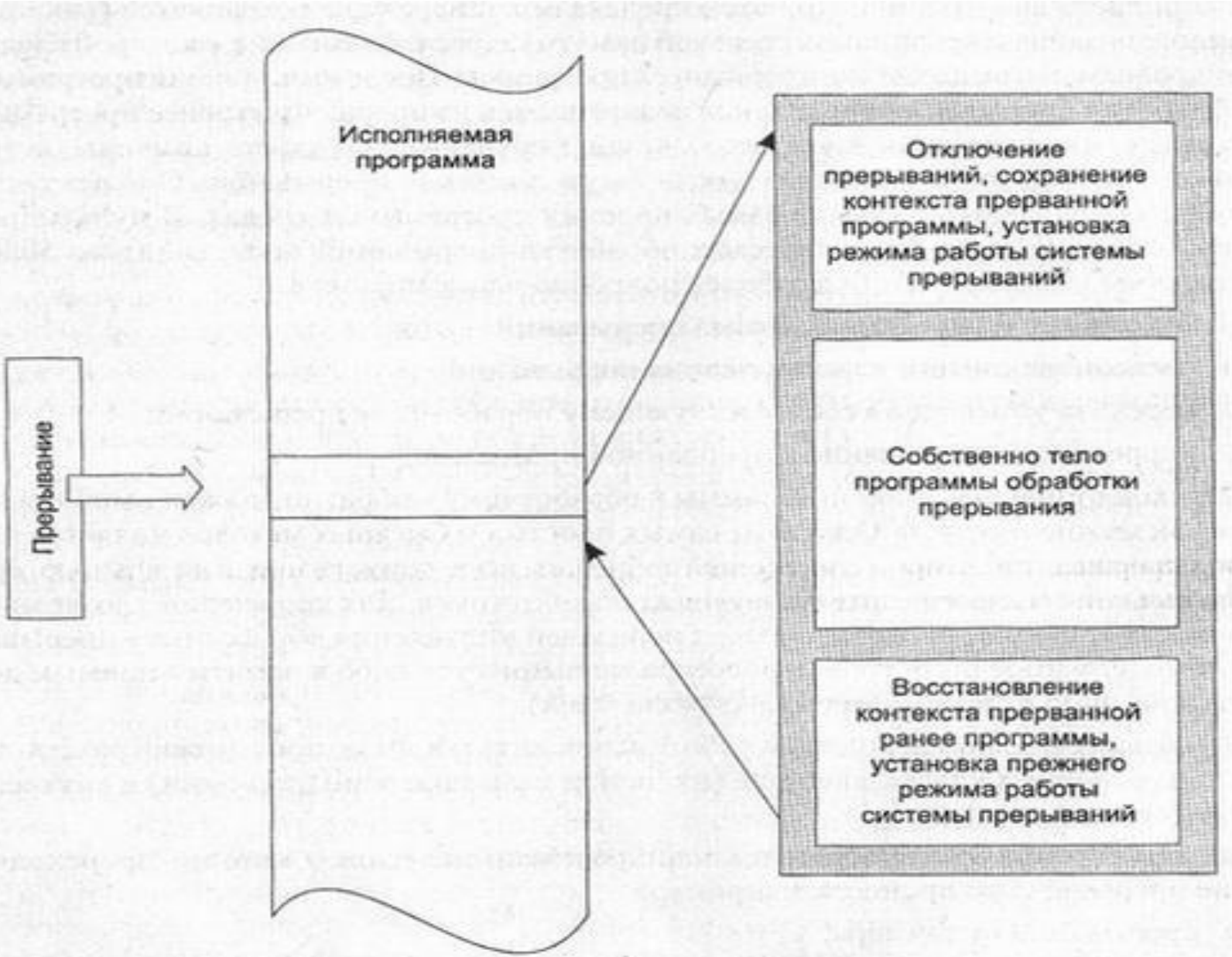
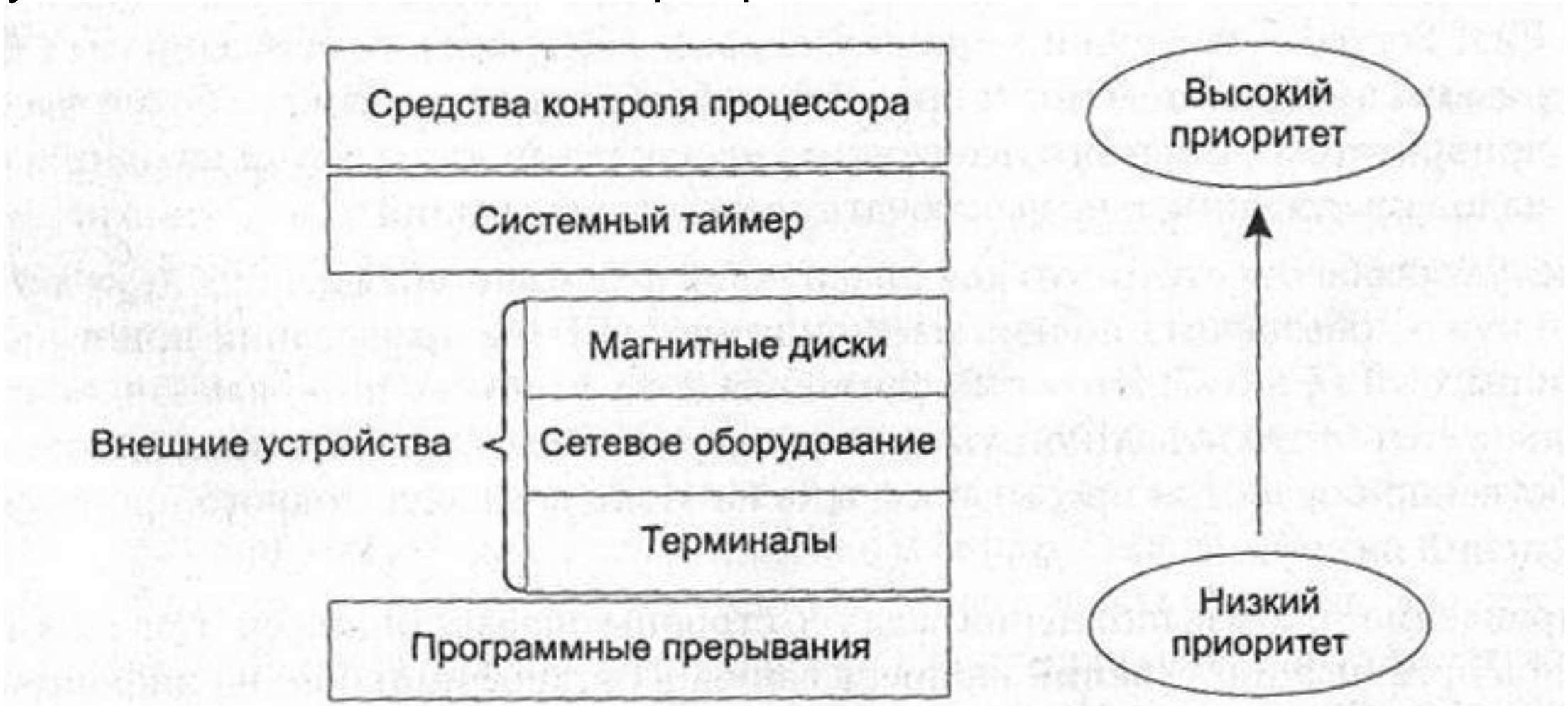


Рис. 1.2. Обработка прерывания

Сигналы, вызывающие прерывания, формируются вне процессора или в самом процессоре, они могут возникать одновременно. Выбор одного из них для обработки осуществляется на основе приоритетов.



Программное управление специальными регистрами маски (маскирование сигналов прерывания) позволяет реализовать различные дисциплины обслуживания.

- *С относительными приоритетами,* то есть обслуживание не прерывается даже при наличии запросов с более высокими приоритетами.
- *С абсолютными приоритетами,* то есть всегда обслуживается прерывание с наивысшим приоритетом.
- По *принципу стека,* или, как иногда говорят, *по дисциплине LCFS* (Last Come First Served — последним пришел, первым обслужен), то есть запросы с более низким приоритетом могут прерывать обработку прерывания с более высоким приоритетом.

Причины прерываний определяет ОС (модуль - *супервизор прерываний).*

Супервизор прерываний прежде всего сохраняет в дескрипторе текущей задачи рабочие регистры процессора, определяющие контекст прерываемого вычислительного процесса.

Далее он определяет подпрограмму, которая должна выполнить действия, связанные с обслуживанием текущего запроса на прерывание.

После выполнения подпрограммы обработки прерывания управление вновь передается ядру операционной системы (модулю, который занимается диспетчеризацией задач). Диспетчер задач в соответствии с принятой дисциплиной распределения процессорного времени (между выполняющимися вычислительными процессами) восстановит контекст той задачи, которой будет решено выделить процессор.

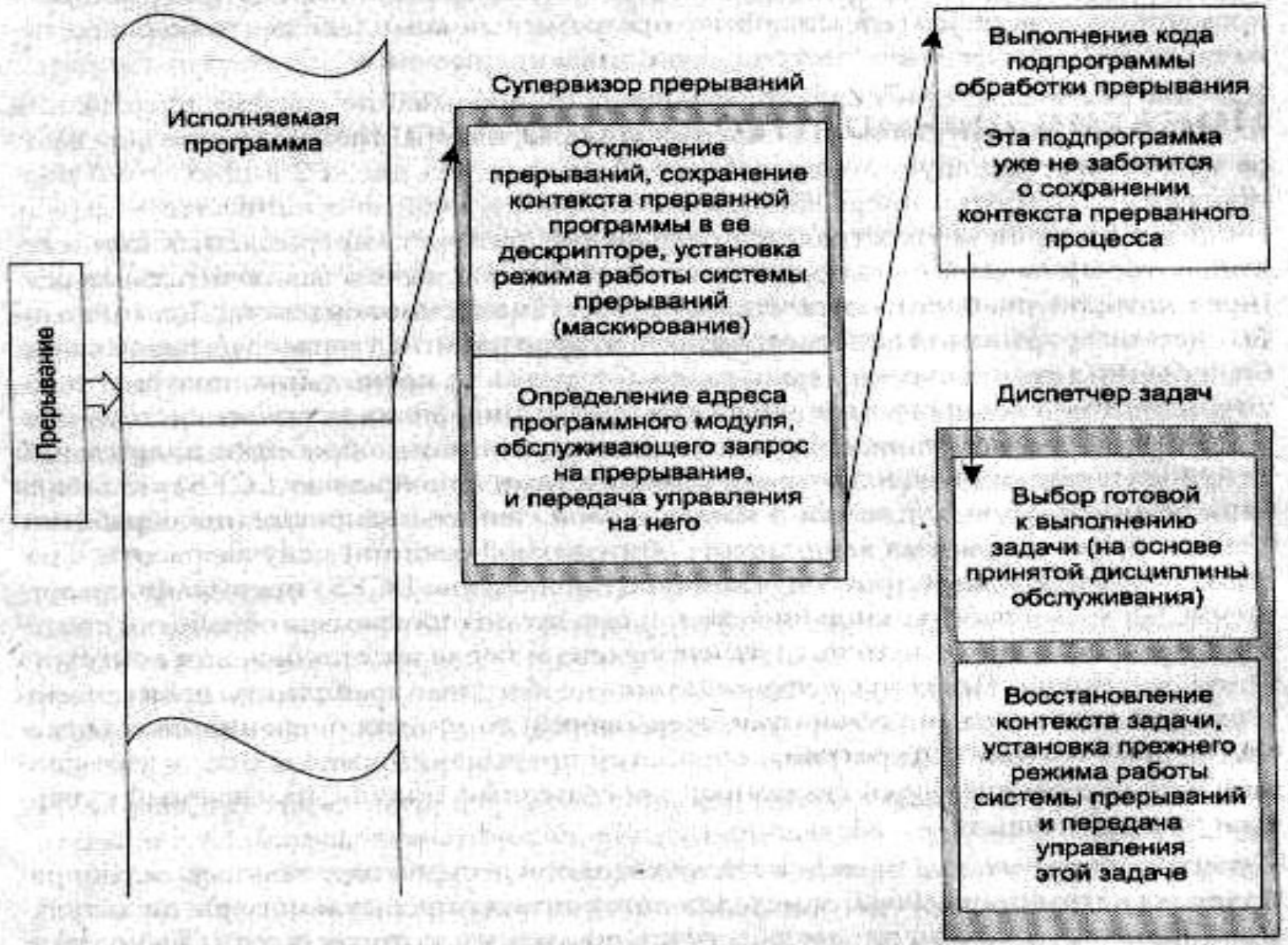


Рис. 1.4. Обработка прерывания при участии супервизоров ОС