БИЛЕТ № 21

1)Особенности ОС с микроядерной структурой. Недостатки и достоинства.

Современная тенденция в разработке операционных систем состоит в перенесении значительной части системного кода на уровень пользователя и одновременной минимизации ядра. Речь идет о подходе к построению ядра, называемом микроядерной архитектурой (microkernel architecture) операционной системы, когда большинство ее составляющих являются самостоятельными программами. В этом случае взаимодействие между ними обеспечивает специальный модуль ядра, называемый микроядром. Микроядро работает в привилегированном режиме и обеспечивает взаимодействие между программами, планирование использования процессора, первичную обработку прерываний, операции ввода-вывода и базовое управление памятью.

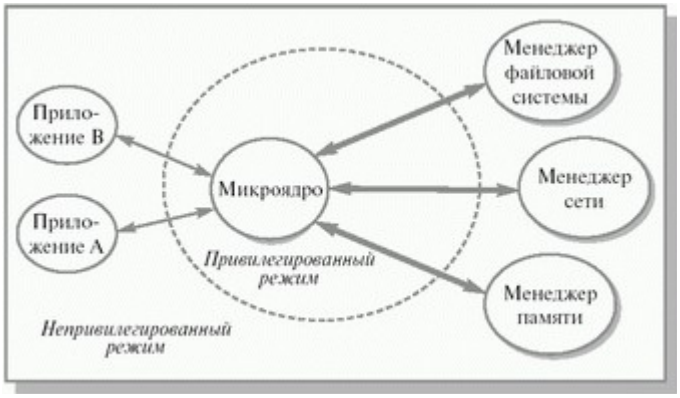


Рис. 1.4. Микроядерная архитектура операционной системы

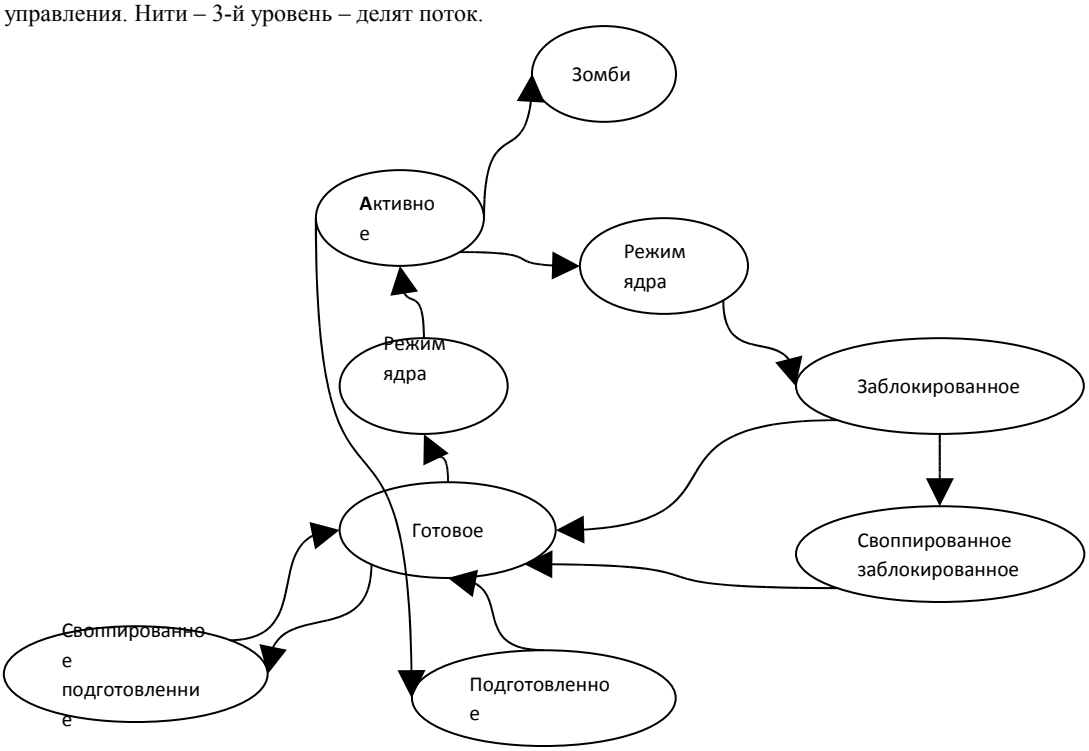
Остальные компоненты системы взаимодействуют друг с другом путем передачи сообщений через микроядро.

Основное достоинство микроядерной архитектуры – высокая степень модульности ядра операционной системы. Это существенно упрощает добавление в него новых компонентов. В микроядерной операционной системе можно, не прерывая ее работы, загружать и выгружать новые драйверы, файловые системы и т. д. Существенно упрощается процесс отладки компонентов ядра, так как новая версия драйвера может загружаться без перезапуска всей операционной системы. Компоненты ядра операционной системы ничем принципиально не отличаются от пользовательских программ, поэтому для их отладки можно применять обычные средства. Микроядерная архитектура повышает надежность системы, поскольку ошибка на уровне непривилегированной программы менее опасна, чем отказ на уровне режима ядра.

В то же время микроядерная архитектура операционной системы вносит дополнительные накладные расходы, связанные с передачей сообщений, что существенно влияет на производительность. Для того чтобы микроядерная операционная система по скорости не уступала операционным системам на базе монолитного ядра, требуется очень аккуратно проектировать разбиение системы на компоненты, стараясь минимизировать взаимодействие между ними. Таким образом, основная сложность при создании микроядерных операционных систем – необходимость очень аккуратного проектирования.

2) Особенности переключения процессов между состояниями.

Процесс – любая исполняемая программа.
Процесс – траектория процессора в адресном пространстве машины.
В рамках одного процесса активизируются разные потоки. ОС знает о процессе, а о потоке не знает. Есть проблема управления. Нити – 3-й уровень – делят поток.



128 Дать определение – процесс Это любая выполняемая работа в системе; это динамический объект (внутренняя единица работы) системы, которому она выделяет ресурсы;

129 Условие перехода процесса из подготовительного состояния в готовность. Выделение ресурсов.

130 Условие перехода процесса из состояния готовности в активное. Выделение кванта времени процессора, запуск.

131 Условие перехода процесса из активного состояния в блокированное. Ожидание события, освобождения ресурса, конца операции ввода/вывода.

132 Условие перехода процесса из блокированного состояния в готовое. Наступление события, освобождения ресурса, завершение ввода/вывода.

133 Условие перехода процесса из активного состояния в готовое. Истечение кванта времени. (Выделенное время процессора для выполнение этого процесса завершилось и процессор занят другим процессом).

134 Условие создания процесса. Корректность описания процесса на языке описания процессов и выделение ресурсов

135 Отличие создания процесса при рекурсивном обращении (?) Хранить все локальные переменные в стеке.

14 ПРОЦЕСС – Любая вычислительная задача решаемая системой (динамическим объектом системы, которому выделяются ресурсы, это траектория процессора в адресном пространстве ВС)

По способу создания процессы делятся на:

- Системные процессы. Они создаются при загрузке системы, имеют оверлейную или динамически-последовательную структуру

- Проблемные (пользовательские) процессы. Создаются при активизации работы (задачи пользователя) операционной системой. Под активизацией процесса здесь понимается начало выполнения программы, подчиненной этому процессу, т.е. начало процесса. Пользовательский процесс может быть активизирован только другим процессом

По способу существования процессы делятся на:

- Последовательные. Это процессы, которые выполняются друг за другом, т.е. когда закончится выполнение первого процесса, начинается второй и т.д.

- Параллельные. Процессы, которые могут выполняться одновременно. Параллельные процессы могут быть:

o Независимые, т.е. выполняют разные операции и не имеют общих частей.

o Асинхронные ? это процессы, которые должны синхронизироваться и взаимодействовать друг с другом.

ОПЕРАЦИИ НАД ПРОЦЕССАМИ

1. Создание процесса включает в себя:

а) присвоение имени процессу,

б) включение этого имени в список имен процессов, известных системе,

в) определение начального приоритета,

г) формирование блока управления процессом,

д) выделение начальных ресурсов.

2. Уничтожение процесса означает его удаление из системы. Ресурсы, выделенные этому процессу, возвращаются системе, имя в системных списках стирается и блок управления процессом освобождается.

3. Изменение приоритета означает просто модификацию значения приоритета в блоке управления данным процессом. В основном увеличивают приоритет у процессов, когда предполагают, что они будут находиться в состоянии бесконечного откладывания или уменьшают приоритет процессу, надолго захватившему процессор.

4. Запуск (выбор) процесса, осуществляемый планировщиком, который выделяет процессу время процессора.

5. Приостановленный процесс не может продолжить свое выполнение до тех пор, пока его не активизирует любой другой процесс (кратковременное исключение определенных процессов в периоды пиковой нагрузки). В случае длительной приостановки процесса, его ресурсы должны быть освобождены. Решение об освобождении ресурса зависит от его природы. Основная память должна освобождаться немедленно, а вот принтер может и не быть освобожден. Приостановку процесса обычно производят в следующих случаях:

а) если система работает ненадежно и может отказать, то текущие процессы надо приостановить, исправить ошибку и затем активизировать приостановленные процессы;

б) если промежуточные результаты работы процесса вызвали сомнение, то нужно его приостановить, найти ошибку и запустить либо сначала, либо с контрольной точки;

в) если ВС перегружена, что вызвало снижение ее эффективности;

г) если для продолжения нормального выполнения процессу необходимы ресурсы, которые ВС не может ему предоставить.

6. Возобновление (или активизация) процесса ? операция подготовки процесса к повторному запуску выполняемая операционной системой, с той точки, в которой он был приостановлен

ИЕРАРХИЯ ПРОЦЕССОВ Процесс может породить новый процесс. При этом первый, порождающий процесс, называется родительским, а второй, порожденный процесс, ? дочерним. Для создания некоторого процесса необходим только один родительский процесс

3) Особенности адресации памяти в защищенном режиме.

Особенности адресации памяти в защищенном режиме.

В защищённом режиме, также как и в реальном, существуют понятия логического и физического адреса. Логический адрес в защищённом режиме (иногда используется термин "виртуальный адрес") состоит из двух 16-разрядных компонент - селектора и смещения. Селектор записывается в те же сегментные регистры, что и сегментный адрес в реальном режиме. Однако преобразование логического адреса в физический выполняется не простым сложением со сдвигом, а при помощи специальных таблиц преобразования адресов.

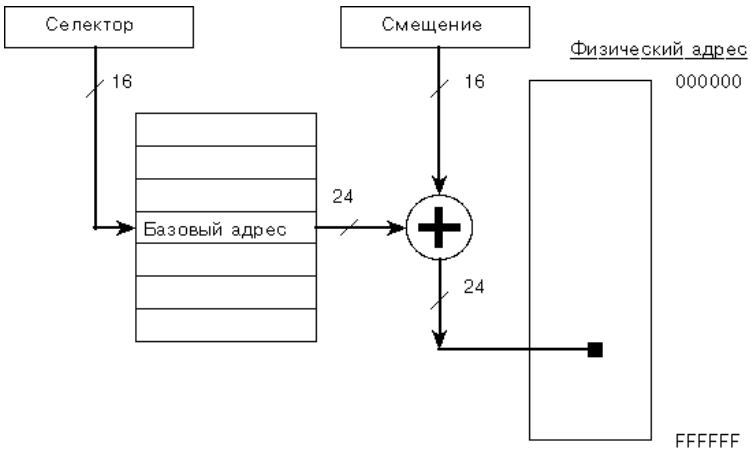
В первом приближении можно считать, что для процессора i80286 селектор является индексом в таблице, содержащей базовые 24-разрядные физические адреса сегментов. В процессе преобразования логического адреса в физический

процессор прибавляет к базовому 24-разрядному адресу 16-разрядное смещение, т.е. вторую компоненту логического адреса. На рис. показана сильно упрощённая схема преобразования логического адреса в физический.

Такая схема формирования физического адреса позволяет непосредственно адресовать 16 мегабайт памяти с помощью 16-разрядных компонент логического адреса.

Заметьте, что селектор - это не сегментный адрес. Это индекс, с помощью которого процессор извлекает из специальной таблицы 24-разрядный базовый адрес сегмента. В реальном режиме мы имеем дело с сегментным адресом и смещением, а в защищённом - с селектором и смещением.

На самом деле не все 16 бит селектора используются для индексации по таблице базовых адресов. В качестве индекса выступают старшие 13 бит. Два младших бита (бит 0 и бит 1) используются системой защиты памяти, о чём мы подробно поговорим в следующем разделе. Бит 2 позволяет выбирать для преобразования адреса один из двух типов таблиц преобразования адресов.



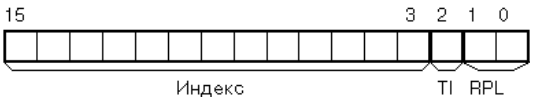


Рис. Формат селектора адреса.

На этом рисунке два младших бита обозначены как RPL (Requested Privilege Level). Это поле является запрошенным программой уровнем привилегий и его мы будем обсуждать позже. Поле TI (Table Indicator) состоит из одного бита. Если этот бит равен нулю, для преобразования адреса используется так называемая глобальная таблица дескрипторов GDT (Global Descriptor Table), в противном случае - локальная таблица дескрипторов LDT (Local Descriptor Table).

Таблица дескрипторов - это просто таблица преобразования адресов, содержащая базовые 24-разрядные физические адреса сегментов и некоторую другую информацию. То есть каждый элемент таблицы дескрипторов (дескриптор) содержит 24-разрядный базовый адрес сегмента и другую информацию, описывающую сегмент.

Таблица GDT - единственная в системе. Обычно в ней находятся описания сегментов операционной системы. Таблиц LDT может быть много. Эти таблицы содержат описания сегментов программ, работающих под управлением операционной системы, т.е. отдельных задач. В каждый данный момент времени процессор может использовать только одну таблицу LDT.

Процессор имеет два регистра, предназначенных для адресации используемых в настоящий момент таблиц GDT и LDT. Регистр GTDR описывает расположение и размер таблицы GDT, а регистр LDTR содержит ссылку на использующуюся в настоящее время таблицу LDT.

На рис. 6 показана уточнённая схема преобразования адресов в защищённом режиме. Из рисунка видно, что регистры процессора GDTR и LDTR определяют расположение в памяти таблиц GDT и LDT соответственно. Таблицы GDT и LDT содержат дескрипторы, описывающие сегменты памяти. В этих дескрипторах, помимо другой информации (заштрихованная область) содержится 24-разрядный базовый адрес сегмента.

Старшие 13 битов селектора (индекс) выбирают элемент из таблицы GDT или LDT в зависимости от состояния бита TI селектора.

Извлечённый из таблицы дескрипторов базовый адрес сегмента складывается процессором для получения 24-разрядного физического адреса.

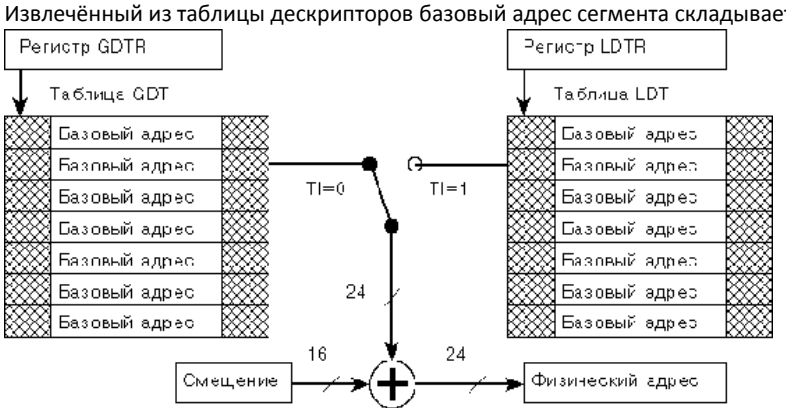


Рис. 6. Уточнённая схема преобразования адресов.

Селектор 0000h адресует самый первый дескриптор в глобальной таблице дескрипторов GDT. Поле RPL для этого дескриптора равно 0, поле TI также равно 0. Селектор 0008h указывает на второй элемент таблицы GDT, а селектор 0014h указывает на третий дескриптор в локальной таблице дескрипторов LDT, т.к. поле TI в нём равно 1.

4) Принцип борьбы с тупиками.

Ответ:

Тупик – это ситуация, которая происходит, когда в группе процессов каждый получает монопольный доступ к некоторому ресурсу и каждому требуется еще один ресурс, принадлежащий другому процессу в группе.

Способы борьбы с тупиками:

- предотвращение (необходимые методы, чтобы тупик был невозможен, обычно убирают одно из условий возникновения тупика);
- обход;
- обнаружение;
- восстановление системы при обнаружении тупиков.

Способы предотвращения тупиков:

I способ: исключается 2-е условие; процесс запрашивает все ресурсы сразу, но система будет работать не эффективно, кроме того, программист может неправильно предусмотреть количество необходимых ресурсов; может возникнуть ситуация бесконечного откладывания.

II способ: исключает 1-е и 3-е условие; если процесс пытается захватить новые ресурсы, а система не может их предоставить, то у этого процесса отбираются все ресурсы. Недостаток – можем потерять всю работу, которая делалась до этого, опять может возникнуть ситуация бесконечного откладывания. Поэтому, чтобы исключить потери, можно решать задачу модульно.

III способ: исключение четвертого условия; все ресурсы выстраиваются в заранее определенном порядке. Круговое ожидание исключения. Недостаток – нельзя перескакивать через ресурсы (изменять порядок).

IV способ: «*алгоритм банкира*». Рассматривает каждый запрос по мере поступления и проверяет, приведет ли его удовлетворение к безопасному состоянию. Предусматривает:

1) знание количества ресурсов каждого вида; 2) знание max количества ресурсов для каждого процесса;

3) знание, сколько ресурсов занимает каждый процесс; 4) в системе разрешено захватывать и освобождать ресурсы по одному; 5) понятие надежности/ненадежности: система находится в надежном состоянии, если при наличии свободных ресурсов хотя бы 1 процесс может завершиться до конца.

Обход тупика:

Тупики можно предотвратить структурно, построив систему таким образом, что тупиковая ситуация никогда не возникнет по построению (если позволить процессу использовать только один ресурс в любой момент времени, то необходимое для возникновения тупика условие циклического ожидания не возникнет). Тупиков также можно избежать, если перенумеровать все ресурсы и затем требовать от процессов создания запросов в строго возрастающем порядке.

Обнаружение тупика:

Если систему чувствует, что процессы не выходят из системы по квоте и эффективность падает, рисуется граф состояний процессов и ресурсов. Затем, в матричном представлении графа ищем любой процесс, который может быть завершен, и удаляем его. Если граф редуцировался до конца, – мы избежали тупика.

Восстановление системы после обнаружения тупика:

наиболее распространенный способ устранить тупик – завершить выполнение одного или более процессов, чтобы впоследствии использовать его ресурсы. Также в системе можно использовать контрольные точки и откат, нужно сохранять состояния процессов, чтоб иметь возможность восстановления предыдущих действий.

Доп.инфа:

32,33,34

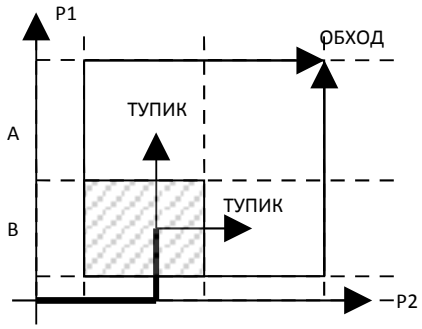
Методы борьбы с тупиками:

- Предотвращение

Нужно удалить одно из условий возникновения. Действия могут быть такими:

1. Процессу выделяются все необходимые ресурсы. Такой подход ресурсоёмок и может привести к бесконечному откладыванию.
2. При запросе отсутствующего ресурса, система забирает у процесса вообще все ресурсы. Таким образом можно делать контрольные точки для отката
3. Удаление циклического ожидания. Ресурсы ранжируются по приоритету и захват ресурсов процессом производиться по возрастанию приоритета.

- Обход



Алгоритм «Банкира»

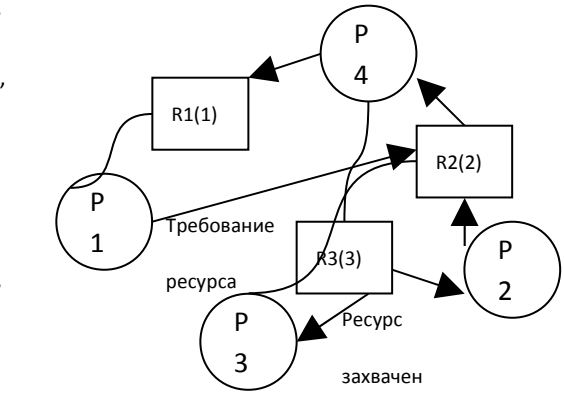
Если в системе известна следующая информация:

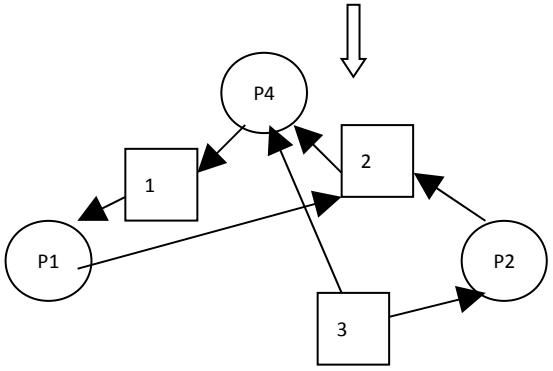
- максимальное количество ресурсов
- количество ресурсов, захваченных системой
- количество ресурсов, которые может потребовать каждый процесс.

Если в системе после выделения доп. ресурсов может завершиться хотябы один процесс, то система находиться в надёжном состоянии. Если же наоборот, ни один процесс не может завершиться, то система – в ненадёжном состоянии. Алгоритм предусматривает удержание системы в надёжном состоянии.

- Обнаружение

Если в системе есть подозрение на тупик, то граф-карта, по ней можно проверяется, хотя-бы ли завершить процесс, один граф если можно –





редуцируется. Если граф редуцируется – то подозрение было напрасно, если нет – то эти процессы – в тупике.

- Восстановление после тупика

Если процессы в тупике, то можно убить 1 либо всен процессы, а потом восстановиться с самого начала или с контрольной точки.

---

5) Особенности и принципы сигналов.

Конспект: К средствам межпроцессорного взаимодействия относятся:

1)сигналы

2)каналы

3)именованные каналы (ФИФО)

4)семафоры (мониторы, мютексы, секвенторы)

5)разделяемые файлы

6)сообщения

7)сокеты

Сигналы были предложены как средства уведомления об ошибках, а ща испол как средства элементарного IPC (Inter Process Communication) для синхр процессов или пересілки єлемент. сообщ. от одного проц. другому.

Однако, применения сигналов ограничевается ввиду их ресурсоемкости.

Появляющееся собітия, которое сопровождается сигналом имеет свой номер. Поєтому, говоря о сигналахнужно разделять его генерацию, отправку сигнала, доставку сигнала и его обработку, т.е. сигнал отправл . когда выполняется событие, а доставл . – когда процесс получит его и обрабатывает.

Наиболее широкоиспол . сигналы (события):

- Особые ситуации (Пр: деление на нуль)

- Терминал. Прерывания (нажатие соотв. клавиш вызывает терминал, «убивающ», процесс)

- Другие процессы (через спец . сист . вызов проц может послать сигнал другому процу)

- Сигналы, связанные с управлением заданиями (возникает через командные интерпретаторы)

- Сигналы-квоты (если процу заданы квоты (размер диск . пространства, память, …) и он их привысил, то сообщение о превышении квоты)

- уведомление (проц . запрашивает ресурс, чтоб предупредить другой)

- alarm(будильник)

-доставка и обработка сигнала.

Для каждого сигнала сущ . обраб . по умолчанию. Вобщем случае сигнал . вызывает завершение выполнения, игнорировать и поставить обработчик (ПР: sig_kill, sig_stop).

Любая обработка сигнала, в том числе по умолчанию, подразумевает, что проц . ,которому он доставляется – активный, а в сис-х многопрограммного режима может привести к большим задержкам.

Шпоры:

Сигналы

Сигналы изначально принимались, как способы извещения об ошибках. Это средство представляло собой простейшие команды. Однако они ресурсоемкие, так как отправка требует системного вызова, а доставка – прерывания процесса-получателя. Они слабоинформативны и число их ограничено. То есть, они служат для информирования процесса или процессов о наступлении события.

Сигналы бывают синхронными и асинхронными. Поэтому в системе реализован механизм управления сигналами, который состоит из 2-х фаз:

1) Генерация и отправка.
2) Доставка и обработка.

Время между двумя фазами может быть достаточно большим.

Виды сигналов зависят от того, какой процесс генерирует сигнал и кто является получателем.

Причины, вызывающие отправку сигнала:

- особые ситуации (деление на 0);
- терминальные прерывания (нажатия кнопки);
- другие процессы;
- системы управления заданиями;
- сигналы для управления фоновыми и текущими задачами;
- сигналы квоты (если процесс превышает квоту – ему отправляется сигнал);
- сигналы уведомления (о наступлении какого-либо события);
- alarm (связанный со счетчиком таймера).

Доставка и обработка сигналов

Для каждого сигнала предусмотрена обработка по умолчанию. Однако пользователь может предусмотреть по сигналу завершение процесса.

При сигнале процесс может:

- завершить свое выполнение;
- проигнорировать сигнал;
- остановиться и продолжиться потом;
- отложить обработку сигнала на время.

Следует аккуратно относиться к блокированию и игнорированию сигналов. Особенно сигналов исключительных ситуаций. Некоторые сигналы нельзя обрабатывать самим (kill, stop).

Процесс может обработать сигнал только в том случае, если он активен, что может привести к большим задержкам при большом количестве процессов. То есть сигнал не может быть обработан, если нужный процесс не выбран планировщиком и ему не предоставлен ресурс.

Сигнал может быть доставлен, если только ОС от имени процесса сформирует специальный системный вызов для доставки сигнала. Этот вызов **ISSIG()** формируется:

- непосредственно перед возвращением процесса из режима ядра в режим задач, после обработки системного вызова;
- непосредственно перед переходом в состояние сна;
- сразу после пробуждения.

Если процедура ISSIG() обнаруживает сигнал, ожидающий доставки, то ядро вызывает функцию самой доставки, вызов по умолчанию, либо функцию sendsuk (специальная функция обработки).

Типичные ситуации:

user работает с терминалом и нажимает Delete, вызывается аппаратное прерывание, по прерыванию драйвер терминала определяет, что была нажата клавиша и определяет сигнал текущей процедурой, связанной с терминалом. Когда процесс будет запущен планировщиком и запущен на выполнение, то при переходе в режим задачи он обнаружит сигнал и выполнит его; а если он уже был активен, то сигнал будет обработан после перехода этого процесса в режим задачи, после выхода из режима прерывания.