БИЛЕТ № 13

1) Основные структуры ОС и их назначение.

Если программа меняет информацию, с которой она работает (помимо своей собственной), то это *обрабатывающая* программа (компиляторы, загрузчики, редакторы связей), если не меняет, – то *управляющая*.

Управляющие программы выполняют следующие действия:

1. **Управление** – на уровне ядра, в режиме супервизора осуществляется:
   - отслеживание входа и выхода задания в системе;
   - отслеживание выполнения собственной задачи и управление ее выполнением;
   - обеспечение работы систем управления файлами, памятью, внешними устройствами.
2. **Управление задачами.** Следит за выполнением задачи на ресурсах системы
3. **Система управления файлами.**
4. **Управление памятью.**
   - программное (обеспечивает эффективное функционирование памяти в соответствии с ее организацией; программа решает свои задачи на уровне процесса, то есть память выделяется процессу);
   - микропрограммное (программа запрашивает у админа, какие разделы необходимо выполнять; у каждого раздела – своя очередь);
5. **Управление внешними устройствами.**
6. Управление заданиями отслеживает прохождение задания от входа до выхода

Современные ОС имеют сложную структуру, каждый элемент которой выполняет определенные функции по управлению компьютером.

- Управление файловой системой. Процесс работы компьютера сводится к обмену файлами между устройствами. В ОС имеются программные модули, управляющие файловой системой.
- Командный процессор. В состав ОС входит специальная программа – командный процессор, – которая запрашивает у пользователя команды (запуск программы, копирование, удаление и т.д.) и выполняет их.
- Драйверы устройств. В состав ОС входят драйверы устройств, специальные программы, которые обеспечивают управление работой устройств и согласование информационного обмена с другими устройствами, позволяют производить настройку параметров устройств. Каждому устройству соответствует свой драйвер. Программы (драйверы) для поддержки наиболее распространенных устройств входят в Windows, а для остальных устройств поставляются вместе с этими устройствами или контроллерами. Можно самому установить или переустановить драйвер.
- Графический интерфейс. Для упрощения работы пользователя в состав ОС входят программные модули, создающие графический пользовательский интерфейс. Пользователь может вводить команды с помощью мыши, а не с клавиатуры.
- Сервисные программы. (утилиты). Позволяют обслуживать диски (проверять, сжимать, дефрагментировать), выполнять операции с файлами (архивировать и т.д.), работать в сетях и т.д.
- Справочная система. Позволяет оперативно получить необходимую информацию о функционировании ОС.

1 Виды модулей. Виды: 1) исходный (текст программы); 2) объектный; 3) загрузочный; 4) исполняемый (абсолютный). Системные программы (компилятор, редактор связей, загрузчик) используются для преобразования от 1) к 4).

2 Отличие загрузочного модуля от объектного.Загрузочный модуль имеет все для своего выполнения . Написан на машинно-ориентированном языке, но быть выполненным не может. При использовании компилятора из исходного модуля получается объектный модуль, при использовании редактора связи из объектного получается загрузочный, далее из него при использовании загрузчика получается исполняемый модуль.

3 Отличие загрузочного модуля от абсолютного. Загрузочный модуль имеет все для своего выполнения . Написан на машинно-ориентированном языке, но быть выполненным не может. Нужно настроить адресные константы. Абсолютный – все адресные константы уже настроены.

4 Дать определение – резидентный модуль. модуль, который резидентно находится в памяти. (постоянно)Резидентные проги. – находятся в ОП, поддерживают быстрый отклик системы, Проги обработки прерываний, Процессы 4/ работы с ВУ, управление памятью, управление процессами, начальная прога. Управления заданиями.

5 Дать определение – транзитный модуль. Программы, связанные с выполнением функций ОС, но не находящиеся постоянно в ОП называются транзитными. Эти программы вызываются в ОП по мере необходимости. – проги., которые могут понадобится 4/ выполнения ф-ций ОС (но не резидентные) – могут вызываться по оверлейной || динамически-последовательной схеме в транзитную зону.

6 Связь модулей по управлению. Какие операции, какими программами выполняются. Связь по управлению предусматривает сохранение состояния модуля на момент перехода на вызываемый модуль, причем действия по этому сохранению выполняет вызываемый модуль. Операции: возврат из i-1 модуля только в i+1; возврат из i+1 в i.

Вызывающий модуль после передачи управления стирается. Виды программ могут быть: повторно не исполняемые, повторно исполняемые, чистые процедуры. (via common regs) Связь по управлению – связь для организации возврата. Возврат модуль В должен иметь команду, делающую так, что в области сохранения модуля А сохраняется состояние регистров. В модуле В есть адрес области сохранения в модуле А. Сохраняется адрес возврата, для возврата адреса следующей команды.

7 Связь модулей по данным. Виды связей Виды: по данным и по управлению. По данным может быть через общие регистры или через адрес списка параметром. Связь по данным – данные через общий регистр || через системный регистр передается адрес списка параметров.

Свойства модуля. Стандартность внутренней структуры, функциональная завершенность, параметрическая универсальность, взаимная независимость.

8 Виды загрузчиков Отличаются выполнением основных 4-х функций: распределения памяти, настройки, редактирования и загрузки. Загрузчики делятся: Абсолютный загрузчик, настраивающий загрузчик, непосредственно - связывающий загрузчик.

9 Функции абсолютного загрузчика Распределение памяти, настройка, редактирование;

10 Функции настраиваемого загрузчика Распределение памяти, настройка, редактирование, загрузка. Инфу для него готовит компилятор.

11 Какие топы загрузчика работают с модулями COM и EXE. С COM - абсолютный; с EXE - настраивающий.

12 Какую информацию и как компилятор передает настраивающему загрузчику Настраивающий – работает в загр. Модуле которому больше ничего не надо. Постепенно его ф-ции взял на себя редактор связей. Непосредственно в настраивающем загрузчике каждый модуль может транслироваться отдельно. Чтобы передать сообщение редактору связей надо ему непосредственно указать, что надо транслировать. В каждом модуле в начале трансляции выделяются вектора перехода, внешние и внутренние.(Экспорт и Импорт процедур и ф-ций). Кроме того для выполнения настройки каждая команда отмечается битом переместимости. ОС выделяет и пользуетс глобально выделенной памятью, а загрузчик с локальной.

13 Функции редактора связей Редактирование связей.(Выполнение связывания подпрограмм являющихся внешними по отношению к загружаемому модулю). (С помощью редактора связи мы получаем из объектного модуля загрузочный модуль имеющий всё для своего исполнения).

14 Выходная и входная информация редактора связей Входная - объектный модуль, выходная - загрузочный модуль.

15 Дать определение – обрабатывающие программы ОС Программы выполнения стандартных (в рамках ОС) функций, обработки исключительных ситуаций. Обработка – изменение ин-фы с которой работает прога в составе ОС (дрова, загрузчик)

16 Дать определение – управляющие программы ОС Программы постоянно находящиеся в памяти (резидентные) организующие корректное выполнение процессов и функционирование всех устройств системы при решении задач. Составляют ядро ОС. Управление : Заданиями – слежение за прохождением заданий от входа до выхода на всех этапах его выполнения. Задачами – (Процессами) – слежение за всеми задачами активизированными в системе и процессами их выполнения на ресурсах. Памятью – решение задач эффективного u/ mem (internal) в соответствии с ее организацией.(Защита – согласование ин-фы в кешах) Данными – эффективное размещение и u/ данных на внешних носителях (проблема эффективности u/ процессора) Внешними ус-вами …

17 Основное отличие компилятора от интерпретатора Интерпретатор выполняет перевод части проги в машинные команды и тут же ее выполняет, выполняя все необходимые настройки адресных констант. Компилятор создает объектный модуль, который затем обрабатывается редактором связей. Во время работы интерпретатора исполняемый код программы записывается в фиксированное место и управление передается на стартовый адрес программы.

2) Виды организации памяти.

Управление памятью.

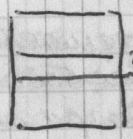
УП подразумевает методы загрузки, распределение, расар смены страниц (вирт дал.), методы повышения эффективности работы памяти.

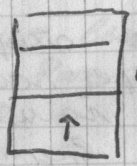
Следует различать и рассматривать вместе управление и организацию памяти.

Организация памяти.

С точки зрения орган. памяти можно различать:

1. однопрограммная





2. многопрограммная, память надо ещё делить между несколькими программами.
   а) разделение памяти с фиксированными разделами (МФТ)
   б) МПТ - - // - с переменными разделами.

Типичные форматы сис-м с управлением памятью

Страничная

№стр	Смещ

Управл. блок	№ кадра

Сегментная

№ сегм	Смещение

Упр блок	Длина	адрес табл. страниц / адрес сегмента

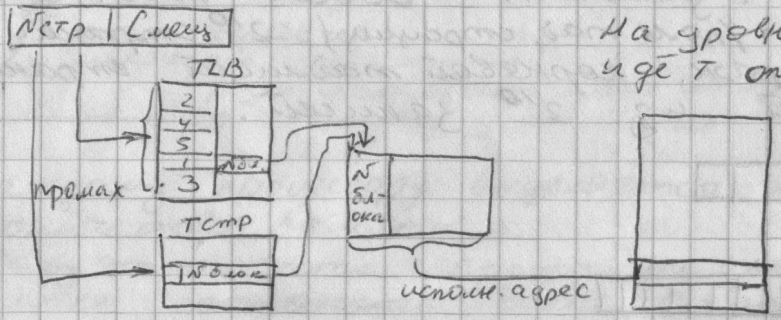
Сегм - стр

№сегм	№стр	смещен.

Упр. блок	Длина	Адр. сегмента

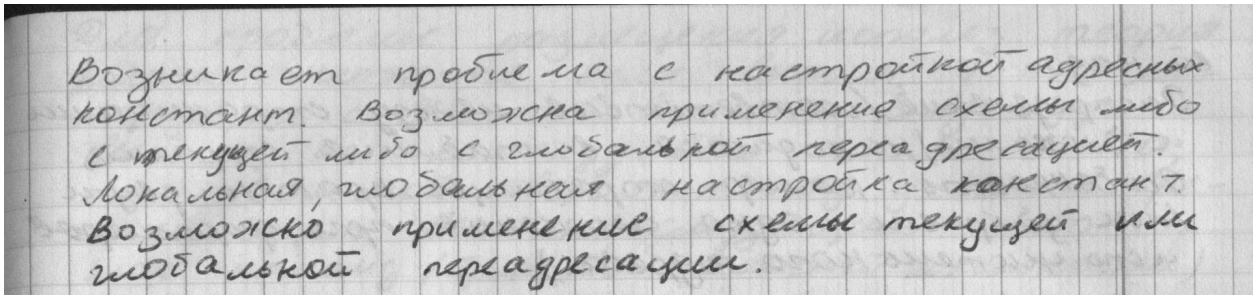
Упр. блок	№ кадра

В машинах преобразов. адресса идёт на аппаратном уровне. В машинах исп. двух уровневые кэши. Один из кэшей явл. ассоциативная память в которой реализ. TLB-буфер (буфер быстрого преобр. адреса). Ассоциативная память ведёт обращение не по адресу, а по содержимому.



На уровне регистров идёт опер. сложения.

Если идёт промах, тогда ищется нужн. страница в таблице страниц.
Если нету в Табл. страниц и TLB, то идет страничное прерывание и подкач в ОП измен. содержимое TLB и Табл. страниц. Если нету в Табл. страниц, то образ из TLB.



3 Методы повышения эффективности носителей данных.

Доп.инфа.

Производительность файловых систем:

Во многих файловых системах применяют оптимизацию, для увеличения производительности:

1. **Кэширование.** Для минимизации количества обращений к диску применяют блочный или буферный кэш. Кэш – набор блоков, логически принадлежащих диску, но хранящиеся в оперативке. Происходит захват всех запросов чтения к диску и проверке требуемой информации в КЭШе. Если блок присутствует – запрос чтения происходит без обращения к диску. Иначе блок считывается с диска в кэш, оттуда копируется по нужному адресу памяти.
2. **Опережающее чтение блока.** Метод заключается в попытке получить блоки блоки диска в кэш прежде чем они потребуются. Применимо только для последовательного считывания файлов.
3. **Снижение времени перемещения блока головок.** Блоки с высокой вероятностью доступа помещаются близко друг к другу, желательно на один цилиндр.
4. **Перемещение i-узлов в середину диска.** Среднее расстояние перемещения блока головок уменьшается вдвое.
5. **Разбиение диска на группы цилиндров, каждая со своими i-узлами, блоками и списками свободных блоков.** При создании нового файла может быть выбран любой i-узел, но предпринимается попытка найти блок в той же группе цилиндров, что и i-узел.

Оптимизация доступа к диску

Методы оптимизации:

1. **FCFS** (первый пришел – первый обслужился);
2. **SSTF** (поиск с найменшим временем);
3. **SCAN** (сканирующий алгоритм, принцип действия заключается в том, что головка двигается внутри и по дороге обслуживает все запросы);
4. **C-SCAN** (запросы обрабатываются от внешней дорожки к внутренней до конца, потом идет перескок на внешнюю дорожку, и так до конца запроса);
5. **N-STEP-SCAN** (точно также как SCAN, но после начала движения запросы ставятся в отдельную очередь);
6. **SLTF** (с наименьшим временем ожидания первого; обрабатываются запросы под головкой в рамках всего цилиндра);

Схема Эшенбаха пытается обработать только одну дорожку.

Приемы повышения эффективности работы с внешними носителями

Существует 2 подхода к повышению эффективности работы с внешними носителями:

1.Оптимизация физического обращения к диску (перемещения головок,…)

2.Оптимизация логического обращения (RAID)

В первом случае алгоритмы делятся на 2 вида.

Когда текущая дорожка неизвестна используются следующие алгоритмы:

1.Random access

2.FIFO – самый простой – операционная система ловит запросы к диску

3.LIFO

Если текущая дорожка известна то используются:

1.SSTF(стратегия наименьшего времени обслуживания) , еще до ввода-вывода определяется текущее перемещение головки. Из очереди заявок выбирается заявка с ближайшим перемещением.

Недостаток: при частом обращении некоторые заявки долго ожидают.

2.SCAN – головка перемещается в одном направлении, по пути обслуживая заявки. Все вновь поступившие заявки включаются в обслуживание. Т.е происходит обратный проход.

3.C-SCAN – SCAN без обратного прохода.

4.NstepSCAN. Используются 2 очереди. В начале сканируется очередь, она фиксируется и новые заявки поступают в дополнительную очередь. При скачке назад очереди перезаписываются.

5. По приоритету.

При втором подходе(RAID).RAID-массив – избыточный массив независимых дисков ,рассматриваемых ОС как один диск. Обеспечивается либо параллельный , либо независимый доступ.

RAID0 – чисто параллельный доступ .Избыточность не используется.

Disk1	Disk2	Disk3	Disk4
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

RAID1 – «зеркало».

Disk1	Disk2	Disk3	Disk4
0	1	0	1
2	3	2	3
…	…	…	…

RAID2 – второй диск хранит код Хемминга(исправляет 1 ошибку и обнаруживает 2) первого диска.

Disk1	Disk2	Disk3	Disk4
0	F(0)		

RAID3 - Используется четность с чередующимися битами.

RAID4 – Используется технология независимых дисков.

RAID5 – Защита чередующимися блоками.

Disk1	Disk2	Disk3	Disk4	Disk5
0	1	2	3	F(0-3)
4	5	6	F(4-7)	7
8	9	F(8-11)	10	11

12	13	14	F(12-15)	15

RAID6

Disk1	Disk2	Disk3	Disk4	Disk5	Disk6
0	1	2	3	F(0-2)	F(0-3)
4	5	6	F(4-6)	F(4-7)	7

4) Структура загрузчика MS-DOS. Назначение элементов.

Ответ:

Доп.инфа:

Для функционирования операционной системы MS-DOS в оперативную память программой начальной загрузки загружается обязательная часть MS-DOS: MSDOS.SYS, IO.SYS, COMMAND.COM.

- IO.SYS — обеспечивает логический интерфейс низкого уровня между системой и аппаратурой (через BIOS); содержит набор резидентных драйверов основных периферийных устройств. При инициализации программа определяет состояние всех устройств и оборудования, а также управляет операциями обмена между памятью и внешними устройствами.
- В MSDOS.SYS располагаются программы управления памятью, файлами, данными, внешними устройствами, заданиями, процессами. Это центральный компонент DOS, реализующий основные функции ОС — управление ресурсами ПК и выполняемыми программами. Основу составляют обработчики прерываний верхнего уровня.
- COMMAND.COM — командный процессор, обеспечивающий интерфейс пользователя с вычислительной системой

Файл IO.SYS, загруженный с диска, обычно состоит из двух модулей: BIOS и SYSINIT. SYSINIT вызывается с помощью программы инициализации BIOS. Модуль определяет величину непрерывной памяти, доступной системе, и затем располагает SYSINIT в старшие адреса. Далее модуль переносит ядро системы DOS, MSDOS.SYS, из области ее начальной загрузки в область окончательного расположения.

Далее SYSINIT вызывает программу инициализации в модуле MSDOS.SYS. Ядро DOS инициализирует ее внутренние таблицы и рабочие области, устанавливает вектора прерываний по адресам с 20h по 2Fh и перебирает связанный список резидентных драйверов устройств, вызывая функцию инициализации для каждого из них.

Когда ядро DOS проинициализировано и все резидентные драйверы доступны, модуль SYSINIT может вызвать обычный файловый сервис системы MS-DOS и открыть файл CONFIG.SYS. Драйверы, в соответствии с директивами DEVICE=, указанными в CONFIG.SYS, последовательно загружаются в память, активизируются с помощью вызовов соответствующих модулей инициализации и заносятся в связанный список драйверов. Функции инициализации каждого из них сообщают SYSINIT размер памяти, отведенный под соответствующий драйвер.

После загрузки всех установленных драйверов, SYSINIT закрывает все дескрипторы файлов и открывает консоль, принтер и последовательный порт. Это позволяет замещать резидентные драйверы BIOS стандартных устройств. Модуль SYSINIT перемещает себя к старшим адресам памяти и сдвигает ядро системы DOS в строну младших адресов памяти в область окончательного расположения.

В конце своего выполнения SYSINIT вызывает системную функцию EXEC для загрузки интерпретатора командной строки, т.е. COMMAND.COM.

COMMAND.COM — командный процессор, который обеспечивает интерфейс пользователя с установкой.

---

5) Контекст процесса, состав и назначение.

Контекст процесса включает в себя содержимое адресного пространства задачи, выделенного процессу, а также содержимое относящихся к процессу аппаратных регистров и структур данных ядра. С формальной точки зрения, контекст процесса объединяет в себе пользовательский контекст, регистровый контекст и системный контекст (*). Пользовательский контекст состоит из команд и данных процесса, стека задачи и содержимого совместно используемого пространства памяти в виртуальных адресах процесса. Те части виртуального адресного пространства процесса, которые периодически отсутствуют в оперативной памяти вследствие выгрузки или замещения страниц, также включаются в пользовательский контекст.

Регистровый контекст состоит из следующих компонент:

- Счетчика команд, указывающего адрес следующей команды, которую будет выполнять центральный процессор; этот адрес является виртуальным адресом внутри пространства ядра или пространства задачи.
- Регистра состояния процессора (PS), который указывает аппаратный статус машины по отношению к процессу. Регистр PS, например, обычно содержит подполя, которые указывают, является ли результат последних вычислений нулевым, положительным или отрицательным, переполнен ли регистр с установкой бита переноса и т.д. Операции, влияющие на установку регистра PS, выполняются для отдельного процесса, потому-то в регистре PS и содержится аппаратный статус машины по отношению к процессу. В других имеющих важное значение подполях регистра PS указывается текущий уровень прерывания процессора, а также текущий и предыдущий режимы выполнения процесса (режим ядра/задачи). По значению подполя текущего режима выполнения процесса устанавливается, может ли процесс выполнять привилегированные команды и обращаться к адресному пространству ядра.
- Указателя вершины стека, в котором содержится адрес следующего элемента стека ядра или стека задачи, в соответствии с режимом выполнения процесса. В зависимости от архитектуры машины указатель вершины стека показывает на следующий свободный элемент стека или на последний используемый элемент. От архитектуры машины также зависит направление увеличения стека (к старшим или младшим адресам), но для нас сейчас эти вопросы несущественны.
- Регистров общего назначения, в которых содержится информация, сгенерированная процессом во время его выполнения. Чтобы облегчить последующие объяснения, выделим среди них два регистра - регистр 0 и регистр 1 - для дополнительного использования при передаче информации между процессами и ядром.

Системный контекст процесса имеет "статическую часть" (первые три элемента в нижеследующем списке) и "динамическую часть" (последние два элемента). На протяжении всего времени выполнения процесс постоянно располагает одной статической частью системного контекста, но может иметь переменное число динамических частей. Динамическую часть системного контекста можно представить в виде стека, элементами которого являются контекстные уровни, которые помещаются в стек ядром или выталкиваются из стека при наступлении различных событий. Системный контекст включает в себя следующие компоненты:

- Запись в таблице процессов, описывающая состояние процесса (раздел 6.1) и содержащая различную управляющую информацию, к которой ядро всегда может обратиться.
- Часть адресного пространства задачи, выделенная процессу, где хранится управляющая информация о процессе, доступная только в контексте процесса. Общие управляющие параметры, такие как приоритет процесса, хранятся в таблице процессов, поскольку обращение к ним должно производиться за пределами контекста процесса.
- Записи частной таблицы областей процесса, общие таблицы областей и таблицы страниц, необходимые для преобразования виртуальных адресов в физические, в связи с чем в них описываются области команд, данных, стека и другие области, принадлежащие процессу. Если несколько процессов совместно используют общие области, эти области входят составной частью в контекст каждого процесса, поскольку каждый процесс работает с этими областями независимо от других процессов. В задачи управления памятью входит идентификация участков виртуального адресного пространства процесса, не являющихся резидентными в памяти.
- Стек ядра, в котором хранятся записи процедур ядра, если процесс выполняется в режиме ядра. Несмотря на то, что все процессы пользуются одними и теми же программами ядра, каждый из них имеет свою собственную копию стека ядра для хранения индивидуальных обращений к функциям ядра. Пусть, например, один процесс вызывает функцию creat и приостанавливается в ожидании назначения нового индекса, а другой процесс вызывает функцию read и приостанавливается в ожидании завершения передачи данных с диска в память. Оба процесса обращаются к функциям ядра и у каждого из них имеется в наличии отдельный стек, в котором хранится последовательность выполненных обращений. Ядро должно иметь возможность восстанавливать содержимое стека ядра и положение указателя вершины стека для того, чтобы возобновлять выполнение процесса в режиме ядра. В различных системах стек ядра часто располагается в пространстве процесса, однако этот стек является логически-независимым и, таким образом, может помещаться в самостоятельной области памяти. Когда процесс выполняется в режиме задачи, соответствующий ему стек ядра пуст.
- Динамическая часть системного контекста процесса, состоящая из нескольких уровней и имеющая вид стека, который освобождается от элементов в порядке, обратном порядку их поступления. На каждом уровне системного контекста содержится информация, необходимая для восстановления предыдущего уровня и включающая в себя регистровый контекст предыдущего уровня.

(*) Используемые в данном разделе термины "пользовательский контекст" (user-level context), "регистровый контекст" (register context), "системный контекст" (system-level context) и "контекстные уровни" (context layers) введены автором.

Ядро помещает контекстный уровень в стек при возникновении прерывания, при обращении к системной функции или при переключении контекста процесса. Контекстный уровень выталкивается из стека после завершения обработки прерывания, при возврате процесса в режим задачи после выполнения системной функции, или при переключении контекста. Таким образом, переключение контекста влечет за собой как помещение контекстного уровня в стек, так и извлечение уровня из стека: ядро помещает в стек контекстный уровень старого процесса, а извлекает из стека контекстный уровень нового процесса. Информация, необходимая для восстановления текущего контекстного уровня, хранится в записи таблицы процессов.

На Рисунке 6.8 изображены компоненты контекста процесса. Слева на рисунке изображена статическая часть контекста. В нее входят: пользовательский контекст, состоящий из программ процесса (машинных инструкций), данных, стека и

разделяемой памяти (если она имеется), а также статическая часть системного контекста, состоящая из записи таблицы процессов, пространства процесса и записей частной таблицы областей (информации, необходимой для трансляции виртуальных адресов пользовательского контекста). Справа на рисунке изображена динамическая часть контекста. Она имеет вид стека и включает в себя несколько элементов, хранящих регистровый контекст предыдущего уровня и стек ядра для текущего уровня. Нулевой контекстный уровень представляет собой пустой уровень, относящийся к пользовательскому контексту; увеличение стека здесь идет в адресном пространстве задачи, стек ядра недействителен. Стрелка, соединяющая между собой статическую часть системного контекста и верхний уровень динамической части контекста, означает то, что в таблице процессов хранится информация, позволяющая ядру восстанавливать текущий контекстный уровень процесса. Процесс выполняется в рамках своего контекста или, если говорить более точно, в рамках своего текущего контекстного уровня. Количество контекстных уровней ограничивается числом поддерживаемых в машине уровней прерывания. Например, если в машине поддерживаются разные уровни прерываний для программ, терминалов, дисков, всех остальных периферийных устройств и таймера, то есть 5 уровней прерывания, то, следовательно, у процесса может быть не более 7 контекстных уровней: по одному на каждый уровень прерывания, 1 для системных функций и 1 для пользовательского контекста. 7 уровней будет достаточно, даже если прерывания будут поступать в "наихудшем" из возможных порядков, поскольку прерывание данного уровня блокируется (то есть его обработка откладывается центральным процессором) до тех пор, пока ядро не обработает все прерывания этого и более высоких уровней.

Несмотря на то, что ядро всегда исполняет контекст какого-нибудь процесса, логическая функция, которую ядро реализует в каждый момент, не всегда имеет отношение к данному процессу. Например, если возвращая данные, дисковое запоминающее устройство посылает прерывание, то прерывается выполнение текущего процесса и ядро обрабатывает прерывание на новом контекстном уровне этого процесса, даже если данные относятся к другому процессу. Программы обработки прерываний обычно не обращаются к статическим составляющим контекста процесса и не видоизменяют их, так как эти части не связаны с прерываниями.

Большинство процессоров построены таким образом, что сигнал прерывания проверяется ими только после завершения выполнения текущей команды выполняющегося процесса. Однако, в системе могут существовать прерывания, для которых невозможно ожидание завершения очередной команды, например, если возникает ошибка в самой команде. Поэтому в системе могут использоваться одновременно несколько из перечисленных способов для обработки различных прерываний.

Обнаружив сигнал прерывания, процессор должен запомнить состояние прерванного процесса для того, чтобы продолжить его выполнение после обработки прерывания. После этого в процессор загружается новый процесс, выполняющий обработку прерывания. Эта процедура получила название **переключения контекста** или **переключения состояния**. Она является одной из важнейших в организации работы операционной системы, причем ее реализация требует поддержки и выполнения некоторых функций на аппаратном уровне.

---- --- -- ---- ---- Рассмотрим механизм передачи управления **программе обработки прерывания (IH)**. Как было сказано выше, ОС запоминает состояние прерванного процесса и передает управление IH. Эта операция называется **переключением контекста**. При реализации переключения используются **слова состояния программы (PSW)**, с помощью которых осуществляется управление порядком выполнения команд. В **PSW** содержится информация относительно состояния процесса, обеспечивающая продолжение прерванной программы на момент прерывания.