1. Структура ядра операционной системы.

Операции, связанные с процессами, входят в базовое обеспечение машины. Они включаются в комплекс используемых средств с помощью программ и микропрограмм, совокупность которых составляет **ядро ОС**. Таким образом, все операции, связанные с процессами, осуществляются под управлением ядра ОС, которое представляет лишь небольшую часть кода ОС в целом. Поскольку эти программы часто используются, то резидентно размещаются в ОП. Другие же части ОС перемещаются в ОП по мере необходимости. Ядро ОС скрывает от пользователя частные особенности физической машины, предоставляя ему все необходимое для организации вычислений:

- само понятие процесса, а значит и операции над ним, механизмы выделения времени физическим процессам;
- примитивы синхронизации, реализованные ядром, которые скрывают от пользователя физические механизмы перестановки контекста при реализации операций, связанных с прерываниями.

Выполнение программ ядра может осуществляться двумя способами:

1. вызовом примитива управления процессами (создание, уничтожение, синхронизация и т.д.); эти примитивы реализованы в виде обращений к супервизору;
2. прерыванием: программы обработки прерываний составляют часть ядра, т.к. они непосредственно связаны с операциями синхронизации и изолированы от высших уровней управления.

Таким образом, во всех случаях вход в ядро предусматривает сохранение слова состояния (при переключении контекста в PSW) и регистров процессора (в PCB), который вызывает супервизор или обрабатывает прерывание.

Функции ядра ОС

Ядро ОС содержит программы для реализации следующих функций:

- обработка прерываний;
- создание и уничтожение процессов;
- переключение процессов из состояния в состояние;
- диспетчеризацию заданий, процессов и ресурсов;
- приостановка и активизация процессов;
- синхронизация процессов;
- организация взаимодействия между процессами;
- манипулирование PCB;
- поддержка операций ввода/вывода;
- поддержка распределения и перераспределения памяти;
- поддержка работы файловой системы;
- поддержка механизма вызова— возврата при обращении к процедурам;
- поддержка определенных функций по ведению учета работы машины;

Одна из самых важных функций, реализованная в ядре — обработка прерываний.

В систему поступает постоянный поток прерываний и требуется быстрая реакция на них с тем, чтобы эффективно использовались ресурсы системы, и пользователь как можно быстрее получал ответ (на запрос в виде прерывания). При обработке (дешифрации) текущего прерывания ядро запрещает другие прерывания и разрешает их только после завершения обработки текущего. При большой интенсивности прерываний может сложиться такая ситуация, когда ядро будет блокировать прерывания в течение значительной части времени, т.е. не будет иметь возможности эффективно реагировать на прерывания. Поэтому, ядро ОС обычно осуществляет лишь минимально возможную предварительную обработку каждого прерывания, а затем передает это прерывание на дальнейшую обработку соответствующему системному процессу, после начала работы которого ядро могло бы реагировать на последующие прерывания. Таким образом улучшается реакция системы на прерывания (сигнал прерывания).

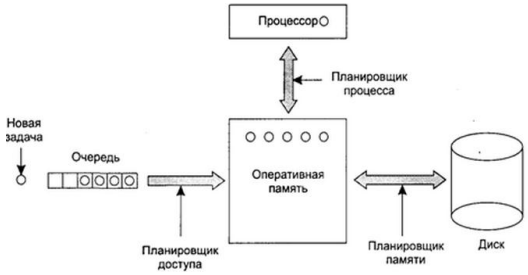
2. Процедура ввода задания в ВС. Участие системных программ. Особенности реализации в различных ОС.

Задание – внешняя единица работы системы, на которую система ресурсы не выделяет. Работа, которую мы хотим дать ОС, чтоб она её выполнила. Задания накапливаются в буферах. В задании должна быть указана программа и данные. Как только появляются ресурсы, задание расшифровывается. Формируется task control block (TCB), он же process control block (PCB).

Задача – внутренняя единица системы, для которой система выделяет ресурсы (активизирует задание).

Процесс – любая выполняемая программа системы; это динамический объект системы, которому она выделяет ресурсы; траектория процессора в адресном пространстве машины. ОС всегда должна знать, что происходит с процессом.

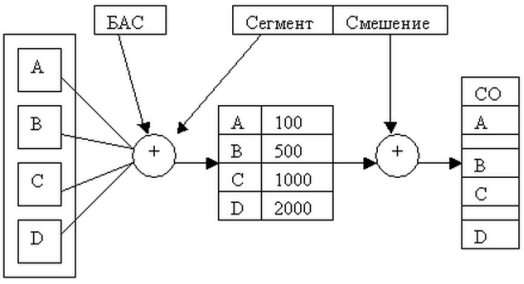
Система управления заданиями предназначена для управления прохождением задач на многопроцессорных вычислительных установках (в том числе кластерных). Она позволяет автоматически распределять вычислительные ресурсы между задачами, управлять порядком их запуска, временем работы, получать информацию о состоянии очередей.



1 уровень. Задание кто-то формирует. Программист или система. В первом случае, если задание правильно оформлено (синтаксис), оно накапливается в очереди входных заданий.
Для выбора заданий из очереди может использоваться алгоритм, в котором устанавливается приоритет коротких задач перед длинными. Впускной/входной планировщик волен придержать некоторые задания во входной очереди, а пропустить задание, поступившее позже остальных.
Т.е. на 1 уровне происходит фильтрация заданий, но ресурсы к ним не определяются. В задании должна быть указана программа и данные. В результате, имеем очередь входных заданий.
2 уровень. Если в системе есть свободные ресурсы (см. определение выше), то всплывает планировщик второго уровня. Он выбирает из очереди задание, определяет, можно ли его решить с помощью ресурсов, что есть в системе. Если можно, то вызывает инициализатор/инициатор, который как только определит ресурсы и расшифрует задание, сформирует блок управления задачей/процессом (PCB).
Так исторически сложилось, что сначала был TCB, а потом появился программный режим работы, и появились процессы и PCB.
Таким образом, задание превращается в задачу/процесс. Возможна ситуация, когда процессов слишком много и они все в памяти не помещаются, тогда некоторые из них будут выгружены на диск. Второй уровень планирования определяет, какие процессы можно хранить в памяти, а какие — на диске. Этим занимается планировщик памяти.
Для оптимизации эффективности системы планировщик памяти должен ре шить, сколько и каких процессов может одновременно находиться в памяти. Кол-во процессов, одновременно находящихся в памяти, называется степенью многозадачности.
Планировщик памяти периодически просматривает процессы, находящиеся на диске, чтобы решить, какой из них переместить в память. Среди критериев, используемых планировщиком, есть следующие:
1. Сколько времени прошло с тех пор, как процесс был выгружен на диск или загружен с диска?
2. Сколько времени процесс уже использовал процессор?
3. Каков размер процесса (маленькие процессы не мешают)?
4. Какова важность процесса?

3 уровень. Как только процессор освобожден (либо естественно либо с вытеснением), срабатывает планировщик 3-го (верхнего уровня, он же планировщик процессора) и из готовых процессов/задач он должен выбрать процесс/задачу для выполнения и занять время выполнение в процессоре.
4 уровень. Собственно, его можно не выделять отдельно. На последнем этапе результаты выполнения могут быть сохранены или выведены на внешний носитель.

3. Сегментная организация памяти. Особенности организации. Достоинства и недостатки.



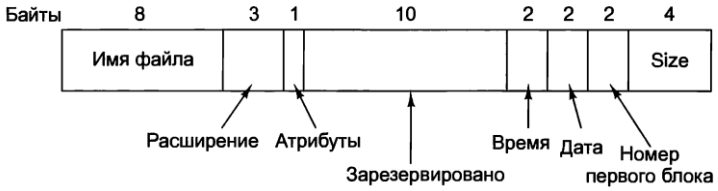
(БАС – базовый адрес сегмента, СО – сист область)
Программа ведется модульно. Каждый модуль занимает непрерывную область памяти, но все программы дробятся при загрузке. При этом возникают проблемы защиты. Требуется таблица сегментов для определения места каждого сегмента. (БАС+имя)-таблица для А получили эффективный адрес и смещение-получили адрес сегмента. Программа состоит из множества модулей. Каждый модуль загружается в любое место.. Базовый адрес определяется соответственно с таблицей сегментов. Чтобы рассчитать настоящий адрес используется таблица страниц. В таблице много программ, для прогр. Табл. сегментов, то появляется табл., в которой адреса таблиц сегментов для процесса.
При помощи атрибутов, записанных в строке таблицы страниц, можно организовать контроль доступа к конкретной странице и ее защиту. (например, биты защиты: 0 - read/write, 1 - read only ...)
Каждый сегмент имеет имя, размер и другие параметры (уровень привилегий, разрешенные виды обращений, флаги присутствия). Каждый сегмент – линейная последовательность адресов, начинающаяся с 0. Максимальный размер сегмента определяется разрядностью процессора (при 32-разрядной адресации это 232 байт или 4 Гбайт). Размер сегмента может меняться динамически (например, сегмент стека). В элементе таблицы сегментов помимо физического адреса начала

сегмента обычно содержится и длина сегмента. Если размер смещения в виртуальном адресе выходит за пределы размера сегмента, возникает исключительная ситуация.
Логический адрес – упорядоченная пара v=(s,d), номер сегмента и смещение внутри сегмента. В системах, где сегменты поддерживаются аппаратно, эти параметры обычно хранятся в таблице дескрипторов сегментов, а программа обращается к этим дескрипторам по номерам-селекторам. При этом в контекст каждого процесса входит набор сегментных регистров, содержащих селекторы текущих сегментов кода, стека, данных и т. д. и определяющих, какие сегменты будут использоваться при разных видах обращений к памяти. Это позволяет процессору уже на аппаратном уровне определять допустимость обращений к памяти, упрощая реализацию защиты информации от повреждения и несанкционированного доступа. Каждый сегмент описывается дескриптором сегмента. ОС строит для каждого исполняемого процесса соответствующую таблицу дескрипторов сегментов и при размещении каждого из сегментов в ОП или внешней памяти в дескрипторе отмечает его текущее местоположение (бит присутствия). Дескриптор содержит поле адреса, с которого сегмент начинается и поле длины сегмента. Благодаря этому можно осуществлять контроль

1) размещения сегментов без наложения друг на друга
2) обращается ли код исполняющейся задачи за пределы текущего сегмента.

В дескрипторе содержатся также данные о правах доступа к сегменту (запрет на модификацию, можно ли его предоставлять другой задаче), защита.
Сегментация помогает в управлении структурами данных, изменяющими свой размер во время выполнения программы, и упрощает процессы компоновки и совместного доступа. Она также облегчает предоставление различных видов защиты разным сегментам. Иногда сегментация и разбивка на страницы комбинируются, предоставляя двумерную виртуальную память.

4. Особенности управления файлами в ОС с FAT. Достоинства и недостатки.



В зависимости от количества блоков на диске в сист. MS-DOS применяется три версии файл сист. FAT: -12, -16, -32.
FAT-12 (макс размер диска 64 Мбайт, раздела - 16)
FAT-16 (макс размер диска 8 Гбайт, раздела - 2)
FAT-32 (макс размер диска 8 Тбайт, раздела - 2)
В MS-DOS файловые блоки отслеживаются через таблицу размещения файлов (FAT — file allocation table), содержащуюся в оперативной памяти. В записи каталога хранится номер первого блока файла. Этот номер используется в качестве индекса к одному из 64К элементов FAT в оперативной памяти.1 Следуя по цепочке, можно найти все блоки. Существуют три версии файловой системы, использующей FAT: FAT-12, FAT- 16 и EAT-32, в зависимости от разрядности дискового адреса. Вообще-то, название FAT-32 дано несколько неверно, поскольку для адресов диска используются только 28 бит младшего разряда. Ее следовало бы назвать EAT-28, но число, являющееся степенью двойки, куда более благозвучно. Для всех вариантов FAT размер дискового блока может быть кратен 512 байтам (это число может быть разным для каждого раздела) и браться из набора разрешенных размеров блока (которые Microsoft называет размерами кластера), разного для каждого варианта. В первой версии MS-DOS использовалась FAT-12 с блоками по 512 байт, задавая максимальный размер раздела 212 x 512 байт (на самом деле только

5. Виды прерываний и особенности их обработки.

Прерывание — сигнал, сообщающий процессору о наступлении какого-либо события. При этом выполнение текущей последовательности команд приостанавливается и управление передаётся обработчику прерывания, который реагирует на событие и обслуживает его, после чего возвращает управление в прерванный код.

По своему назначению, причине возникновения прерывания делятся на различные **классы**. Традиционно выделяют следующие классы:

1. *Прерывания от схем контроля машины*. Возникают при обнаружении сбоев в работе аппаратуры, например, при несовпадении четности в микросхемах памяти.
2. *Внешние прерывания*. Возбуждаются сигналами запросов на прерывание от различных внешних устройств: таймера, клавиатуры, другого процессора и пр.
3. *Прерывания по вводу/выводу*. Инициируются аппаратурой ввода/вывода при изменении состояния каналов ввода/вывода, а также при завершении операций ввода/вывода.
4. *Прерывания по обращению к супервизору*. Вызываются при выполнении процессором **команды обращения к супервизору** (вызов функции операционной системы). Обычно такая команда инициируется выполняемым процессом при необходимости получения дополнительных ресурсов либо при взаимодействии с устройствами ввода/вывода.
5. *Программные прерывания*. Возникают при выполнении **команды вызова прерывания** либо при обнаружении ошибки в выполняемой команде, например, при арифметическом переполнении.

В последнее время принято прерывания 4 и 5 классов объединять в один класс программных прерываний, причем, в зависимости от источника, вызвавшего прерывание, среди них выделяют такие подтипы:

- прерывание вызванное исполнением процессором *команды перехода к подпрограмме обработки прерывания*
- прерывания, возникающие в результате *исключительной* (аварийной) *ситуации* в процессоре (деление на "0", переполнение и т.д.).

В связи с многообразием различных ВС и их постоянным развитием меняется и организация системы прерываний. Так, с появлением виртуальной памяти, появился класс страничных прерываний, который можно отнести и к классу исключительных ситуаций в процессоре; в системах с кэш-памятью существуют прерывания подкачки страниц в кэш-память и т.д.

Во время выполнения обработчика одного из прерываний возможно поступление другого сигнала прерывания, поэтому система должна использовать определенную *дисциплину обслуживания заявок* на прерывания. Обычно различным классам либо отдельным прерываниям присваиваются различные *абсолютные приоритеты*, т.о. при поступлении запроса с высшим приоритетом текущий обработчик снимается, а его место занимает обработчик вновь поступившего прерывания. Количество уровней вложенности прерываний называют **глубиной системы прерываний**. Обработчики прерываний либо дисциплина обслуживания заявок на прерывание должны иметь также специальные средства для случая, когда при обработке прерывания возникает запрос на обработку прерывания от того же источника.