БИЛЕТ № 15

1) Особенности задач, решаемых в распределенных ОС.

Конспект

Распределенная система подразумевает наличие ресурсов, которые для пользователей как один ресурс.
Характеристики распределенных систем: прозрачность, масштабируемость, открытость.
Подключение дополнительных устройств для сохранения характеристики системы.
Прозрачность доступа – обеспечение прозрачности разности представления данных.
Прозрачность местоположения – ресурс может находиться, где угодно.
Прозрачность переноса ресурса – изменение местоположения ресурса, а пользователь этого не знает.
Прозрачность репликации – скрыто, что пользуетесь копией, а не оригиналом.
Прозрачность параллельного доступа – система работает с разделяемыми данными.
Прозрачность отказа – скрыт отказ ресурса.
Прозрачность по сохранности данных – пользователь не знает, где находятся данные (в какой памяти).

Доп. Инфа. Свойства распределенных операционных систем:

- Отсутствие общей памяти приводит к тому, что нельзя определить общее состояние системы с помощью множества совместных переменных, а невозможность совместного обращения к памяти и различия в задержке передач сообщений приводят к тому, что при определении состояния какого-либо элемента системы из двух различных точек можно получить разные результаты, в зависимости от порядка предшествующих событий;
- Распределенная система распределяет выполняемые работы в узлах системы, исходя из соображений повышения пропускной способности всей системы;
- Распределенные системы имеют высокий уровень организации параллельных вычислений.

Два фактора увеличивают вероятность отказов отдельных элементов (по сравнению с централизованными системами):

1. Большое число элементов системы.
2. Надежность систем связи обычно меньше, чем надежность информационных систем.

Однако надежность всей системы в целом в распределенных операционных системах весьма высока за счет специфических свойств, позволяющих исключить или ослабить эффект отказа отдельных элементов системы (исправление вышедших из строя элементов, переконфигурация системы и т.д.).

2) Методы повышения эффективности организации вычислительного процесса ВС.

Приемы, методы повышения эффективности вычислительной машины(за счет организации выч. процесса):

- применение дополнительных уровней памяти(многоуровневая память).
- просмотр команд вперед(подкачка). Буфер – либо одна самая длинна, либо 6 коротких
- секционирование памяти
- конвейерный принцип
- многопрограмный режим работы

3) Особенности организации NTFS

Фундаментальная структура системы файлов Windows 2000 (NTFS) – том (volume). Том создается утилитой администрирования диска. Структура тома основана на логическом диске (partition). Том может занимать часть диска, целый диск или распределяться по нескольким дискам (рис. 28.1). Все метаданные, такие как информация о томе, хранятся в обычном файле.

NTFS использует кластеры как базовую единицу выделения дисковой памяти. Кластер – число секторов диска, размер которого – степень двойки. Поскольку размер кластера меньше, чем в FAT16, внутренняя фрагментация уменьшается.

NTFS использует логические номера кластеров - logical cluster numbers (LCN) в качестве дисковых адресов. Файл в NTFS – не просто байтовый поток, как в MS-DOS или в UNIX, но структурированный объект, состоящий из атрибутов. Каждый файл в NTFS описывается одной или несколькими записями в массиве, хранящемся в специальном файле, называемом Master File Table (MFT). Каждый файл в томе NTFS имеет уникальный идентификатор (ID), называемый ссылкой на файл - file reference. Это 64-битовое число, состоящее из 48-битового номера файла и 16-битового номера последовательности. Ссылка на файл может использоваться для выполнения внутренних проверок целостности.

Пространство имен NTFS организовано в иерархию директорий. Индексный корень (index root) содержит верхний уровень B+ - дерева.

Все изменения структур данных в файловой системе NTFS выполняются как транзакции, для которых используется журнал. Перед тем, как структура данных изменяется, транзакция заносит в журнал специальную запись, которая содержит информацию для повторного выполнения (redo) и отмены (undo) данного изменения. После изменения структуры данных в журнал заносится информация об успешном выполнении операции. В случае порчи информации файловая система может быть восстановлена до целостного состояния с использованием журнальных записей. Эта схема не гарантирует, что все данные пользовательского файла могут быть восстановлены в случае порчи информации, а гарантирует лишь, что все структуры данных о файлах в системе (метаданные) не повреждены и отражают какое-либо целостное состояние данных до порчи информации. Журнал хранится в третьем файле метаданных каждого тома.

Безопасность тома NTFS реализована на основе объектной модели Windows 2000. Каждый файловый объект имеет дескриптор безопасности, хранящийся в записи MFT. Данный атрибут содержит маркер доступа владельца файла, а также список управления доступом, устанавливающий права каждого пользователя для доступа к данному файлу.

FtDisk, дисковый драйвер Windows 2000, устойчивый к сбоям, обеспечивает несколько способов объединения нескольких SCSI-дисков в один логический том. Он логически конкатенирует диски, образуя один логический том (набор дисков тома – volume set). Драйвер выполняет обработку нескольких частей тома по принципу round-robin для формирования "полосатого

множества" (stripe set), или disk striping) – рис. 28.2. Stripe set – набор от 2 до 32 дисков, логически объединенных в единый том. При записи данных в stripe set данные записываются порциями по 64 Кбайта (которые могут распределяться произвольным образом между дисками stripe set). Это позволяет сэкономить время в случае, если работа с дисками stripe set может выполняться параллельно. Такая схема не гарантирует сохранности данных.

Как вариант используется схема stripe set with parity (рис. 28.3), которая позволяет восстановить данные в случае сбоя.

Зеркальное отображение дисков (Disk mirroring) - это надежная схема, использующая множество "зеркал" (mirror set) — две секции одного размера на разных частях диска с идентичным содержимым (рис. 28.4).

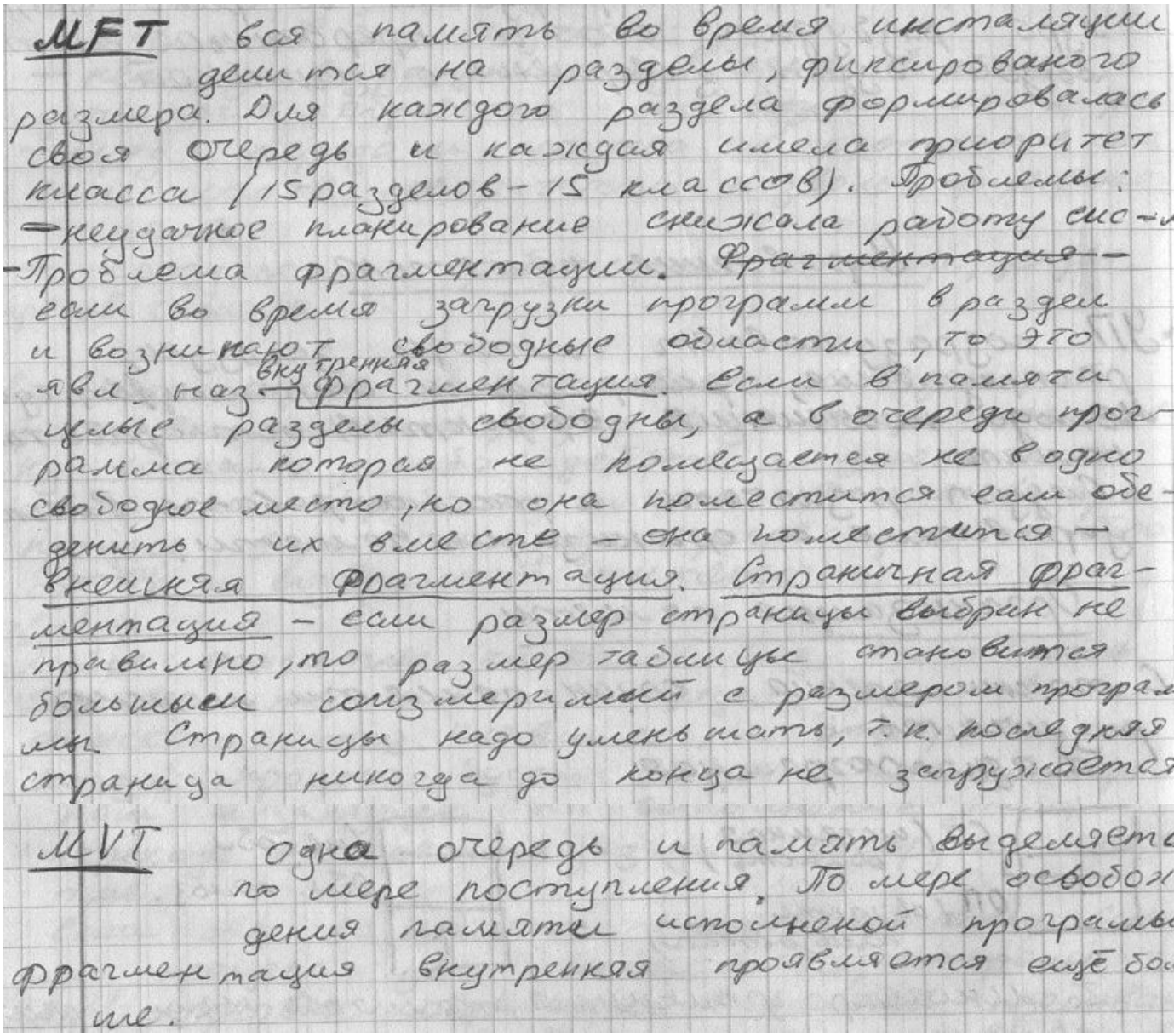
Для обработки запорченных дисковых секторов FtDisk использует аппаратный метод, называемый предохранением секторов (sector sparing), а NTFS использует программный метод, называемый повторным отображением кластеров (cluster remapping).

Сжатие в файловой системе NTFS. Для сжатия файла NTFS разделяет данный файл на модули сжатия (compression units) - блоки по 16 смежных кластеров.

Для не смежно расположенных файлов NTFS использует другой метод экономии памяти. Кластеры, содержащие только нули, фактически не хранятся на диске. Вместо этого, в последовательности виртуальных номеров кластеров оставлены пропуски, информация о которых хранится в элементе MFT для данного файла. При чтении из файла, если найден пропуск в нумерации виртуальных кластеров, NTFS просто заполняет нулями соответствующую часть буфера.

4) Фрагментация, методы борьбы с ней.

Ответ:



Уменьшить фрагментацию при мультипрограммировании с фиксированными разделами можно, если загрузочные модули получать в перемещаемых адресах. Такой модуль может быть загружен в любой свободный раздел после соответствующей настройки.

Борьба с фрагментацией – перемещение всех занятых участков в сторону старших или

младших адресов, так, чтобы вся свободная память образовала свободную единую область(Перемещаемые разделы).

Страничная организация памяти полностью исключает внешнюю фрагментацию

Для борьбы с фрагментацией памяти, а также для решения проблемы перемещения программы по адресному пространству, используется, так называемая, виртуальная память. Суть ее работы заключается в следующем. Пусть имеется некоторое адресное пространство программы, то есть то адресное пространство, в терминах которого оперирует программа. И имеется адресное пространство физическое, которое зависит от времени. Оно характеризует реальное состояние физической оперативной памяти.

В машинах, поддерживающих виртуальную память, существует механизм преобразования адресов из адресного пространства программы в физическое адресное пространство, то есть при загрузке задачи в память машины операционная система размещает реальную задачу в той оперативной памяти, которая является свободной, вне зависимости от того, является ли этот фрагмент непрерывным, либо он фрагментирован. Это первое действие выполняет операционная система. Она знает о состоянии своих физических ресурсов: какие свободны, какие заняты.

Второе: операционная система заполняет некоторые аппаратные таблицы, которые обеспечивают соответствие размещения программы в реальной оперативной памяти с адресным пространством, используемым программой. То есть можно определить, где в физической памяти размещена какая часть программы, и какая часть адресного пространства программы поставлена ей в соответствие.

После этого запускается программа, и начинает действовать аппарат (или механизм) виртуальной памяти. Устройство управления выбирает очередную команду. Из этой команды оно выбирает операнды, то есть адреса и те индексные регистры, которые участвуют в формировании адреса. Устройство управления (автоматически) вычисляет исполнительный адрес того значения, с которым надо работать в памяти. После этого автоматически (аппаратно) происходит преобразование адреса исполнительного программного (или виртуального) в адрес исполнительный физический с помощью тех самых таблиц, которые были сформированы операционной системой при загрузке данной программы в память. И продолжается выполнение команды. Аналогично выполняется и, например, команда безусловного перехода на какой-то адрес. Точно так же устройство управления вычисляет сначала адрес исполнительный, после чего он преобразуется в адрес физический, а потом значение этого физического адреса помещается в счетчик команд. Это и есть механизм виртуальной памяти.

0……

5) Виды прерываний, особенности их обработки.

Прерывание — это нарушение последовательности выполнения действий (команд), т.е. после текущего действия (команды) выполняется не следующее (команда), а некоторое другое действие.

Классы прерываний

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.

((((((((((((((((((((((С ШПОР

В системе присутствует 2 приоритетных уровня:

- Приоритет сигналов

  Используется полноупорядоченная схема сигналов (контроллер, его приоритет определяется путём запуска команды смены прерывания) либо частичноупорядоченная (когда прерываний

много – все они разбиваются на классы, а потом производиться сканирование в полносвязной системе).

- Приоритет прерывания программ
  Более важный уровень, на нём выбирается программа с наивысшим приоритетом.

Дополнительные классы прерываний:

- Прерывания по таймеру
- Прерывания по кэш-промаху
- Страничное прерывание
- Прерывание по нажатию клавиши

))

РЕАЛИЗАЦИЯ МЕХАНИЗМА ПРЕРЫВАНИЙ В КОМПЬЮТЕРАХ ТИПА IBM PC

В компьютерах семейства IBM PC, построенных на основе микропроцессора Intel 80x86 и использующих операционную систему MS-DOS, механизм прерываний реализуется как при помощи аппаратных, так и при помощи программных средств.

Микропроцессор поддерживает обработку 256 различных прерываний, каждое из которых идентифицируется номером в диапазоне 00h-FFh. Сегментные адреса обработчиков прерываний, называемые **векторами прерываний**, содержатся в **таблице векторов прерываний**, которая располагается в начале оперативной памяти (по адресу 0000:0000). Каждый вектор прерывания имеет размер 4 байта, таким образом, таблица векторов занимает 1024 байта. Для определения адреса обработчика любого прерывания нужно номер прерывания умножить на 4.

В защищенном режиме старшие модели семейства процессоров вместо таблицы векторов используют **таблицу дескрипторов прерываний,** сходную по своему назначению с таблицей векторов.

Операционная система не контролирует содержимое таблицы векторов, хотя и имеются средства для чтения и изменения векторов. При помощи функций MS-DOS значение векторов можно изменить, поместив нужный адрес непосредственно в таблицу векторов прерываний. В этом случае контроль за содержимым векторов и их соответствием размещению обработчиков в памяти полностью ложится на программиста, а последствия в случае ошибки могут быть самыми непредсказуемыми.

Активизация обработчика прерывания может произойти в результате возникновения следующих ситуаций:

- возникновение сигнала внутреннего прерывания внутри микросхемы процессора (**внутренние** или **логические прерывания**)
- поступление в процессор сигнала запроса на прерывание от внешнего (по отношению к процессору) устройства (**аппаратные прерывания**)
- выполнение процессором команды INT вызова процедуры обработки прерывания (**программные прерывания**)

2.9.1. Внутренние прерывания

Для обслуживания внутренних прерываний микропроцессоры i8086 использовали 5 первых векторов прерываний (00h-04h). Прерывания с номерами 05h-31h фирма Intel зарезервировала для использования в дальнейших разработках, однако фирма IBM при создании IBM PC использовала эти вектора по своему усмотрению. В последующих реализациях процессоров эти же вектора были задействованы для обработки новых внутренних прерываний, в результате чего стали возможны конфликты. Поскольку все добавленные прерывания используются в защищенном режиме процессора, то для избежания конфликтов при переходе в защищенный режим контроллер прерываний перепрограммируется таким образом, что при поступлении сигнала на линию IRQ0 вызывается процедура обработки с номером, отличным от 08h (обычно 50h или 60h). Для последующих линий IRQ вызываются процедуры с последующими (относительно базового для IRQ0) номерами.

Сигналы внутренних прерываний возникают при нарушениях в работе самого микропроцессора либо при ошибках в выполняемых командах и формируются внутри схемы микропроцессора при возникновении одной из ситуаций, указанных в Табл. 2.2.

2.9.2. Аппаратные прерывания

Аппаратные прерывания инициируются различными устройствами компьютера, внешними по отношению к процессору (системный таймер, клавиатура, контроллер диска и пр.). Эти устройства формируют сигналы запросов на прерывания (IRQ), поступающие на **контроллер прерываний**.

. Немаскируемые прерывания

Существует одно аппаратное прерывание, отличное от всех остальных. Это **немаскируемое прерывание** *(Non-Maskable Interrupt, NMI).* Его отличие состоит в том, что хоть инициатором его и является внешнее устройство, сигнал запроса поступает в процессор, минуя контроллер прерываний. Поэтому такой запрос на прерывание невозможно замаскировать. Кроме этого, запрос по линии NMI вызывает немедленную реакцию процессора и имеет максимальный приоритет в системе, хотя обработчик NMI может быть прерван любым маскируемым прерыванием, если их разрешить командой STI. Это позволяет использовать данное прерывание в особо критических ситуациях: обычно оно используется при падении напряжения питания и при ошибках памяти, однако на некоторых моделях IBM — совместимых компьютеров NMI используется также для обслуживания сопроцессора, клавиатуры, каналов ввода/вывода, дисковых контроллеров, часов реального времени, таймеров, контроллеров прямого доступа к памяти и пр.

В процессорах i80286 и выше в случае, если во время обработки NMI возникнет повторный запрос на немаскируемое прерывание, он не прервет выполнение текущего обработчика, а запомнится и будет обработан после завершения текущего обработчика. Если повторных запросов во время выполнения обработчика будет несколько, то сохранится только первый, последующие будут проигнорированы. При возврате из обработчика командой IRET процессор не переходит на выполнение следующей команды, а пытается выполнить команду по тому же адресу, что привел к возникновению критической ситуации.

2.9.2.3. Программные прерывания

Программные прерывания инициируются специальной командой процессору INT n, где n указывает номер прерывания, обработчик которого необходимо вызвать. При этом в стек заносится содержимое регистров флагов, указателя команд и сегмента кода, а флаг разрешения прерывания сбрасывается (аналогично тому, как при обработке аппаратных прерываний).

Далее в процессор загружается новое PSW, значение которого содержится в векторе прерывания с номером n, и управление переходит к обработчику прерывания. Требования к обработчику программного прерывания такие же, как и к обработчику аппаратного прерывания, за исключением того, что команда завершения аппаратного прерывания (EOI) не требуется. Обработка прерывания завершается командой IRET, восстанавливающей из стека старое PSW.

Обработка программного прерывания является более мягким видом прерывания по отношению к прерываемой программе, нежели обработка аппаратного прерывания, т.к. аппаратное прерывание выполняется по требованию самой прерываемой программы, и она ожидает от этого какого-либо результата. Аппаратное же прерывание получает управление безо всякого ведома выполняющейся программы и зачастую не оказывает на нее никакого влияния.