ОСИ БИЛЕТЫ

Nikita Lukinov

1 Этапы развития вычислительной техники и программного обеспечения.

- 1 поколение - электронно-вакуумные лампы, 40-50 годы 20 века.
- 2 поколение - полупроводниковые элементы (транзисторы, диоды), конец 50-х - начало 60-х.
- 3 поколение - интегральные схемы, конец 60-х - начало 70-х
- 4 поколение - большие интегральные схемы и дальнейшее развитие вычислительных мощностей, приведшее к современному понятию ОС.

2 Ресурсы ВС - физические ресурсы, виртуальные ресурсы. Уровень операционной системы.

Вычислительная система структурно разделяется на несколько уровней: прикладные системы, системы программирования, уровень ОС, аппаратные средства (компоненты). (сверху вниз)

Физическим ресурсом (или устройством) вычислительной системы называют аппаратную компоненту ВС, используемую более высокими уровнями этой системы, и оказывающую на них влияние. Физический ресурс обычно описывается тремя характеристиками: интерфейс программного взаимодействия, объемные (производительные) свойства, степень используемости. Эти характеристики не определяются однозначно и могут зависеть, например, от метода подключения устройства.

Виртуальным (логическим) ресурсом ВС называют ресурс, некоторые или все характеристики которого реализованы программно.

Драйвером физического ресурса называется программа, обеспечивающая управление ресурсом и обмен данными с ним, виртуального - программа, обеспечивающая его существование и использование.

Ресурсами ВС называется совокупность физических и логических ресурсов.

К уровню ОС в вышеупомянутой иерархии относится управление ресурсами ВС.

3 Ресурсы ВС - физические, виртуальные. Уровень систем программирования.

(здесь стоит повторить все, что касается иерархии и ресурсов из в. 2)

Система программирования - программа (комплекс программ), поддерживающая жизненный цикл программы в вычислительной системе (проектирование, кодирование, тестирование, отладка). Уровень системы программирования представляет собой инструментальные средства разработки, основанные на доступе к уровню ОС.

4 Ресурсы ВС - физические ресурсы, виртуальные ресурсы. Уровень прикладных систем.

(та же ремарка, что и в в. 3)

Прикладная система - программная система, предназначенная для решений задач конкретной предметной области.

Уровень прикладной системы, соответственно - это высший уровень в иерархии ВС, использующий ресурсы нижележащих уровней и предоставляющий программистам и иным пользователям возможность решать задачи из необходимой области.

5 Основные компоненты и характеристики. Структура и функционирование ЦП.

В современном компьютере обязательно находятся следующие три составляющие: ЦП, оперативная память (RAM), соединенная с ЦП шиной данных и, наконец, внешние устройства (напр., жесткий диск).

ЦП состоит из арифметико-логического устройства, обрабатывающего машинные инструкции, управляющего устройства, регистровой памяти и кэша первого уровня.

NB: все вышеперечисленное – базовые вещи, которые есть практически во всех современных компьютерах, но ни о какой полноте здесь речи не идет.

6 Основные компоненты и характеристики. Оперативное запоминающее устройство. Расслоение памяти.

(опять же, повторить первый абзац в. 5)

Основное назначение ОЗУ - хранение исполняемой в данной момент программы. ОЗУ состоит из ячеек памяти, которые, в свою очередь, состоят из служебной информации и машинного слова.

Машинное слово - поле программно изменяемой информации. В машинном слове могут храниться либо инструкции программы, либо ее данные.

Под служебной информацией понимается тэг, используемый для контроля корректности и доступности данных. Например, однобитный тэг можно использовать для контроля целостности данных по четности, а также для избежания ошибок, связанных, например, с передачей управления в секцию данных.

Расслоение ОЗУ - одно из оптимизационных решений, призванное сгладить разницу между временем рабочего цикла ОЗУ (мин. время между двумя обращениями) и временем доступа к ячейке. Оно состоит в следующем: все адресное пространство ОЗУ разбивается на $K = 2^L$ независимых подустройств, называемых банками. Банки нумеруются с нуля так, чтобы ячейки с последовательными адресами хранились в соседних банках (по модулю K). Младшие

L битов адреса при этом будут кодировать номер банка.

Как видим, такой подход к организации действительно более эффективен, ведь с вышеупомянутой проблемой мы будем сталкиваться только в том случае, когда нам необходимо будет обратиться к одному и тому же банку несколько раз.

7 Основные компоненты и характеристики. Кэширование ОЗУ.

(та же ремарка, что в 6)

Чтобы еще сильнее ускорить и оптимизировать работу с данными, в процессор интегрировано быстрое устройство хранения данных, называемое кэшем. Основная идея кэширования состоит в том, чтобы хранить наиболее актуальные фрагменты ОЗУ в процессорной памяти, сокращая таким образом количество обращений к оной. В современных компьютерах для удобства в процессоре хранится два кэша - для данных и для команд.

Общая схема работы кэша выглядит так: при очередном обращении к оперативной памяти вначале происходит поиск данных в кэше (для этого используется тэг данных). При нахождении (cache hit) алгоритм останавливается. При промахе (cache miss) из кэша вытесняется по определенной стратегии, зависящей от процессора, какой-либо элемент и заменяется на запрашиваемый.

Стоит также обратить внимание на разные стратегии согласования данных между кэшем и ОЗУ (поддержания cache coherency - связности кэша). Одна из таких стратегий называется сквозным кэшированием и состоит в изменении данных как в кэше, так и в ОЗУ. Другая называется кэшированием с обратной связью и состоит в использовании в ячейке памяти бита модификации, в случае взведенности которого при записи соотвествующая запись в ОЗУ обновляется - это минимизирует количество запросов записи.

8 Аппарат прерываний. Последовательность действий в вычислительной системе при обработке прерываний.

Прерыванием называется определенное событие в компьютере, на возникновение которого предусмотрен стандартный ответ. В обработке прерываний следует различать два этапа - аппаратный и программный. Первый из них подразумевает реакцию процессора на прерывание, второй - программно описанные методы обработки, интегрированные в ОС.

Необходимость в аппарате прерываний естественным образом возникает при решении вопроса обработки ошибочных и аномальных ситуаций в работе устройства ПК. Этап аппаратной обработки выглядит так: завершается выполнение текущей команды (кроме случаев некорректного выполнения команды), вызывается программа-обработчик, при этом также происходит сохранение некото-

рых регистров процессора, необходимых программе-обработчику, и, возможно, самой программе в случае возвращения в нее.

Программный же этап заключается в определении типа прерывания, и, в зависимости от него, либо возвращения в программу (короткое), либо аварийного выхода (фатальное), либо же сохранения регистров и дальнейшего анализа (ошибки вроде выхода за адресное пространство программы, или обращение к системному вызову). Во всех случаях происходит также снятие режима блокировки прерываний, необходимого для того, чтобы при обработке одного прерывания мы не вылетели в другое, потеряв тем самым данные.

Аппаратно прерывания можно реализовать по-разному, например, через специальный регистр, кодируя разные прерывания разными битами и выполняя необходимые действия в случае их взведения.

9 Внешние устройства. Организация управления и потоков данных при обмене с внешними устройствами.

При работе с внешними устройствами и обменом с ними данными существует несколько различных подходов, но в любом случае существует два потока данных - управляющих и обрабатываемых.

Самый простой метод организации работы с ВУ - прямое управление процессором. Это означает, по сути, что API устройства интегрировано в процессор, и оба потока данных проходят непосредственно через него. Такой метод крайне неэффективен, так как нагружает процессор и требует синхронного доступа.

Вслед за прямым управлением возникло синхронное - для него используют специальные устройства, называемые контроллерами. Они обеспечивают разгрузку процессора при обмене данными, поскольку передают информацию не непосредственно, а большими блоками.

Следующий метод управления называется асинхронным - он подразумевает использование специального устройства DMA (direct memory access), которое, как следует из названия, имеет прямой доступ к ОЗУ, исключая таким образом участие процессора в обработке потока данных.

Наконец, последняя модель управления вводит так называемый процессор ввода-вывода – специальный компьютер между ЦП и устройством, обрабатывающий уже высокоуровневые запросы на обмен.

10 Иерархия памяти.

Следующее перечисление идет в порядке убывания скорости доступа, увеличения объема, времени хранения информации, времени доступа: регистры, кэш L1, кэш L2, RAM, внешнее запоминающее устройство с буферизацией, ВЗУ без буферизации, ВЗУ долгосрочного хранения.

11 Мультипрограммный режим.

Мультипрограммным режимом, как следует из названия, называется режим работы ВС, при котором поддерживается единовременная обработка нескольких программ. Корректно реализованным мультипрограммным режимом вполне естественно считать тот, при котором программы функционируют вне зависимости от существования и действий других программ (если только сам пользователь не захочет обратного).

Во-первых, вполне очевидна нужда в аппарате защиты памяти – ведь ситуация, при которой одна программа может читать или изменять данные в адресном пространстве другой, катастрофична.

Во-вторых, из только что выведенной нами необходимости в защите памяти следует необходимость и в аппарате прерываний – ведь если программа обратилась к памяти, ей не принадлежащей, то эта ситуация ошибочна. Совершенно точно требуется, по крайней мере, прерывание по таймеру – ведь при зацикливании той или иной программы нам необходимо эту программу остановить, иначе зависнет вся система.

В-третьих, наконец, должен поддерживаться привилегированный режим работы программы (режим ОС, kernel space), допускающий доступ ко всем командам процессора, в том числе управлением устройствами, памятью и т.д., и непривилегированный (user space) – тот, что допускает исполнение лишь некоторого подмножества этих команд.

Вышеперечисленные требования являются минимальными для корректного мультипрограммного режима.

12 Организация регистровой памяти ЦП.

Примечание: здесь кратко приведены сведения из курса архитектуры ЭВМ, так как вопрос относится скорее к нему.

Процессорные регистры разделяют на несколько типов. Во-первых, это регистры общего назначения – непосредственно используемые при программных расчетах регистры, доступные самой программе. Во-вторых, регистр указателя стека – программа может его изменить, но содержимое этого регистра контролируется аппаратом защиты памяти, в-третьих – регистр указателя инструкции – в нем хранится адрес следующей инструкции для выполнения, программа не может его изменять (вернее говоря, может, но, опять же, с ограничениями аппарата защиты и не тем же образом, что и РОН). Наконец, существует также специальный регистр флагов, или регистр состояния, позволяющий контролировать результаты вычислений. Также в процессоре существуют так называемые сегментные регистры – в них хранятся адреса сегментов текущего процесса.

13 Виртуальная оперативная память.

Необходимость в аппарате виртуальной адресации возникает, когда решаются проблемы фрагментации и перемещаемости. Вторую из них решает так

называемое базирование – отображение виртуального адреса в физический с помощью прибавления к нему некоторого базового адреса – таким образом непрерывная область памяти процесса будет определяться лишь базовым регистром и своим размером.

Однако такой подход не решает проблему фрагментации – потому вводится более продвинутый подход страничной организации. Оперативная память разделяется на блоки равного размера (обычно степень двойки), называемые страницами, и виртуальный адрес определяется следующим образом – последние биты задают смещение относительно страницы, а первые (слева) – номер. При отображении в физический адрес используется таблица страниц, которая может быть организована по-разному, но смысл ее в том, что из виртуального номера страницы получается физический, а смещение остается неизменным. Данный подход позволяет размещать программы по страницам, а более того – не хранить всю программу в ОЗУ целиком единовременно, так что это решает проблему фрагментации.

14 Пример организации страничной виртуальной памяти.

В целом, описан выше.

15 Многомашинные, многопроцессорные ассоциации. Классификация. Примеры.

Классификация Флинна **(здесь S - single, I - instruction, M - multiple, D - data stream):**

1. SISD
2. SIMD
3. MISD
4. MIMD

Машины класса 1 – простейшие устройства, в которых единственный поток данных обрабатывается единственным процессором, первые компьютеры.

Машины класса 2 – к примеру, процессоры с расширениями AVX, SSE.

Машины класса 3 – существование таковых спорно, к ним можно условно отнести некоторое специальное медиаоборудование.

Машины класса 4 – на сегодняшний день самые широко используемые, имеющие несколько подвидов по организации ОЗУ и архитектуре самой системы. В пример можно привести распределенные системы, работающие синхронизированно.

16 Терминальные комплексы. Компьютерные сети.

Терминальным комплексом называется многомашинная ассоциация, призванная решить следующую задачу - организация массового доступа удаленных и локальных пользователей к ресурсам некоторой ВС. Для организации такой системы используются следующие устройства:

- основная ВС
- локальные мультиплексоры
- локальные терминалы
- модемы
- удаленные терминалы
- удаленные мультиплексоры.

Локальные устройства подключаются к ВС непосредственно или мультиплексированно. Удаленные устройства подключаются к ВС через коммуникационную среду.

Компьютерная сеть – объединение компьютеров, взаимодействующих с помощью коммуникационной среды.

Коммуникационная среда – совокупность каналов связи и иных средств передачи данных.

17 Операционные системы. Основные компоненты и логические функции. Базовые понятия: ядро, процесс, ресурс, системные вызовы. Структурная организация ОС.

Операционная система — это комплекс программ, в функции которого входит обеспечение контроля за существованием, использованием и распределением ресурсов вычислительной системы.

Процессом называется совокупность машинных команд и данных, обладающая правом доступа к некоторым ресурсам ВС.

Ядром называется некоторая резидентная (т.е., встроенная) часть ОС, реализующая некую базовую ее функциональность и работающая в привилегированном режиме.

Под ресурсами ОС понимаются подконтрольные ей ресурсы ВС.

Наконец, системными вызовами называются некоторые средства обращения к ядру с целью выполнения той или иной задачи. Интерфейс системных вызовов и их набор зависит от системы.

Структурно ОС состоит из API системных вызовов, драйверов ресурсов, ядра.

В логические функции ОС входят следующие:

- управление процессами
- управление оперативной памятью
- планирование процессов
- управление устройствами и ФС
- управление сетевыми взаимодействиями
- обеспечение безопасности.

18 Пакетная ОС, ОС разделения времени, ОС реального времени, распределенные и сетевые ОС.

- Пакетная ОС – система, которая ставит в приоритет минимизацию накладных расходов, т.е. максимальное приближение времени работы процессора ко времени исполнения пользовательских программ. Пакетом программ называется совокупность программ, необходимых ОС для обработки. В таких системах переключение рабочего процесса происходит в одном из трех случаев: завершение процесса, прерывание по вводу-выводу, зацикливание процесса. Такая схема делает эту систему организации действительно эффективной.

- ОС разделения времени – модификация пакетной ОС, вводящая понятие кванта времени и стремящяяся уже к минимизации времени ответа на запрос пользователя. К вышеупомянутым причинам переключения пакетной ОС следует добавить также и истечение выделенного процессу кванта времени.

- ОС реального времени – специализированные системы, призванные взаимодействовать с реальными технологическими процессами и требующими, соотвественно, тонкого подхода к разделению времени. В таких системах планирование жестко связано с некоторым набором событий, при возникновении любого из которых обработка гарантированно будет занимать не более чем какой-либо наперед заданный временной промежуток.

- Сетевые и распределенные ОС – системы, предназначенные для синхронного управления компьютерными сетями и многомашинными комплексами. Сетевая ОС ставится на каждый компьютер в сети и использует их совместно, распределенная ОС позволяет использовать некоторые свои функции в распределенном, соответственно, виде.

19 Эталонная модель ISO/OSI. Протокол, интерфейс. Стек протоколов. Логическое взаимодействие сетевых устройств.

Модель OSI (Open Systems Interconnection) является скорее эталоном и рекомендацией, чем реальным стандартом, так как нигде и никогда не была реализована в полном объеме. Модель OSI рассматривает сетевое взаимодействие компьютеров на 7 уровнях:

1. Физический – непосредственно передача неструктурированного потока двоичных данных в физической среде, определение стандартов соединений, сигналов.

2. Канальный – решение задачи обеспечения доступности, синхронизации и борьбы с ошибками. Канальный уровень работает с порциями данных, называемыми кадрами. В таких порциях присутствует информация в том числе и служебная, для исправления ошибок.

3. Сетевой – решение задачи взаимодействия сетей: управление операциями сети, управление движением пакетов.

4. Транспортный – решение задачи обеспечения корректной транспортировки данных, возможно, выявление и обработка ошибок при передаче.

5. Сеансовый – обеспечение контроля сеансов связи: аутентификация, определение активной стороны, установление контрольных точек и меток, позволяющих в случае ошибок смягчить ущерб.

6. Представительный – обеспечение унификации используемых форматов данных.

7. Прикладной – правила взаимодействия с прикладными системами.

Протоколом называется формальное описание сообщений и правил, по которым устройства обмениваются информацией (также можно считать, что правила взаимодействия одноименных уровней тоже называются протоколом).

Стеком протоколов называется множество протоколов, реализованных в данной системе (от первого до максимально реализованного).

Интерфейсом называются правила взаимодействия вышестоящего уровня с нижестоящим.

В логической схеме взаимодействия сетевых устройств, устройство-отправитель по коммуникационной среде передает данные с i-го, допустим, протокола (предварительно опустив на первый), устройство-приемник поднимает их на тот же уровень и работает с ними.

20 Семейство протоколов TCP/IP, соответствие модели ISO/OSI. Взаимодействие между уровнями протоколов семейства TCP/IP. IP адресация. (очень плохой билет)

Семейство протоколов TCP/IP – классическая четырехуровневая модель взаимодействия. Она состоит из следующих уровней:

1. Доступ к сети – соотвествует физическому и канальному уровням, специфицирует доступ к физической сети.

2. Межсетевой – соответствует сетевому уровню, решает проблемы адресации и маршрутизации, но не устанавливает соединений с другими машинами.

3. Транспортный – соответствует сеансовому и транспортному уровням. В него не всегда входит работа по обработке ошибок и их поиску. На этом уровне могут использоваться виртуальные протоколы передачи, устанавливающие виртуальное соединение или нет.

4. Прикладных программ – решает задачи уровней представления и прикладного. В отличие от OSI-модели, задача стандартизации ложится на сами программы.

Эти уровни являются пакетными, то есть, каждый из них оперирует определенных свойств порциями данных.

Уровень доступа к сети решает задачу предоставления системы средств для передачи данных.

Межсетевой уровень выполняет задачу маршрутизации – по имеющему IP-адресу получателя прокладывает маршрут следования пакета.

Транспортный уровень содержит в себе два важнейших протокола: TCP и UDP. Первый из них передает данные по сети последовательно в виде пакетов и состоит в прокладывании виртуального канала, тогда как UDP является более быстрой, но менее потереустойчивой альтернативой.

Уровень прикладных программ специфицирует протоколы построения распределенных сетевых приложений, некоторые из которых опираются на TCP, некоторые – на UDP.

IP-адресом устройства называется 32- или 64-битное число, кодирующее номер сети и номер хоста. 32-битные IP-адреса разделяются на следующие 5 классов:

- Класс A - 0(7 бит - номер сети)(24 бита - номер хоста)
- Класс B - 10(14 бит - номер сети)(16 бит - номер хоста)
- Класс C - 110(21 бит - номер сети)(8 бит - номер хоста)
- Класс D - 1110(номер группы)
- Класс E - 1111(номер группы).

21 Определение процесса, типы. Жизненный цикл, состояния процесса. Свопинг. Модели жизненного цикла процесса. Контекст процесса.

Как было сказано ранее, процессом называется совокупность машинных команд и данных, обладающих доступом к некоторым ресурсам ВС. Можно условно разделить процессы на полновесные и легковесные, называемые также нитями или потоками.

Жизненным циклом процесса называется цепочка этапов, который он проходит от своего порождения до завершения включительно. На разных системах и при разных реализациях она может варьироваться, но можно выделить следующие типичные этапы: формирование, обработка, ожидание постановки на исполнение, завершение.

В модельных операционных системах, рассматриваемых в курсе, выделяется два основных состояния процесса: нахождение в буфере ввода процессов и нахождение в буфере обрабатываемых процессов. Второе из них, по сути, является совокупностью различных состояний, множество которых определяется реализацией системы.

Свопингом называется техника, состоящая в откачке какого-либо процесса из очереди выполнения во внешнюю память и замену его на какой-либо другой оттуда же.

В разных классах систем, рассмотренных выше, жизненный цикл процесса моделируется по-разному. В пакетной ОС, например, процесс проходит по достаточно простому пути: цикл из обработки ЦП, ожидания операции ввода-вывода, ожидания в очереди на выполнение, по выходу – завершение. ОС с разделением времени позволяет произвольно (в терминах кванта) вытягивать процесс с обработки в очередь и наоборот.

Контекстом процесса называется совокупность данных, описывающих актуальное состяние процесса, и ее можно разделить на три составляющие: пользовательскую, аппаратную, системную.

22 (UNIX) Определение процесса. Контекст, тело процесса. Состояния процесса. Аппарат системных вызовов в ОС UNIX.

Итак, процессом в ОС UNIX называется сущность, порожденная системным вызовом `fork()` (за некоторыми исключениями). С процессом ассоциируется его идентификатор – PID. Этот идентификатор определяет позицию процесса в системной таблице процессов. (есть и определение через таблицу процессов, но это, по сути, тавтология вроде "случайная величина – число, зависящее от воли случая").

В описанной выше модели контекста процесса пользовательская часть называется телом процесса и лежит в адресном пространстве процесса. Аппаратная и системная же составляющие лежат в пространстве ядра и представляют со-

бой образ процессорной памяти при исполнении и системную информацию о процессе соотвественно (заметим, что в состоянии останова аппаратная и системная составляющие совпадают).

Системным вызовом в ОС UNIX называется часть интерфейса ядра, позволяющая обращаться к нему процессам с более низким уровнем привилегий за тем или иным действием, полное множество которых определяется реализацией.

23 Базовые средства управления процессами в ОС UNIX. Загрузка ОС UNIX, формирование нулевого и первого процессов.

Все процессы в ОС UNIX, за исключением специальных с номерами 0 и 1, порождаются системным вызовом `fork()`. Он определен в заголовочном файле `unistd.h` и работает следующим образом: в точке вызова создается копия процесса, сохраняющая следующие составляющие: открытые файлы (кроме тех, чье наследование было запрещено напрямую), переменные окружения, обработчики сигналов, разделяемые ресурсы, текущий каталог и т.д.. Процессу-отцу вызов возвращает при этом PID сына, а процессу-сыну – PID отца, в случае ошибки вызов возвращает отрицательное число и модифицирует переменную `errno` нужным образом.

Стоит также отметить еще одно важное семейство системных вызовов – `exec()`. Каждый из них принимает в аргументе путь к исполняемому файлу и, при успешном отрабатывании, подменяет текущее тело процесса на тело того, что было запрошено. Разные функции этого семейства позволяют по-разному вызывать сам исполняемый файл, а также передавать ему аргументы командной строки и переменные окружения. В случае ошибки этот системный вызов возвращает управление в вызывающую программу.

Базовым средством синхронизации процессов является возможность ожидать в родительском процессе завершения его прямого потомка. Для этого в UNIX существует системный вызов `wait()`, который при вызове блокирует процесс до смерти какого-либо сына. Этот вызов позволяет также считать посмертную информацию сына – например, код его завершения.

Все процессы, как было сказано ранее, в UNIX создаются через идиому fork-exec, но исключением из данного правила являются процессы 0 и 1. При загрузке системы в таблице процессов формируется запись с номером 0 и выделяется некоторое количество памяти под его контекст, но в привычном смысле процессом он не является, удовлетворяя лишь определению.

Формирование процесса 1 начинается с копирования нулевой записи в таблице в первую. Далее, в выделенную для процесса память загружается исполняемый код, содержащий вызов `exec` с целевой программой `etc/init`, что и завершает формирование первого процесса. Все дальнейшие процессы формируются уже по классической схеме, описанной выше.

24 Пример программирования нитей.

В целом, в лекциях показан достаточно простой пример, добавить нечего, а копировать сюда его код бессмысленно.

25 Разделяемые ресурсы. Критические секции. Взаимное исключение. Тупики.

Разделяемым называется ресурс, к которому могут иметь доступ единовременно несколько процессов.

Критической секцией называется часть кода, которая в один момент времени должна исполняться только одним процессом.

Взаимным исключением называется такой подход к работе с ресурсом, который постулирует отсутствие доступа к нему в один момент времени у всех остальных процессов в случае наличие оного у какого-то одного.

Тупик (дедлок) – состояние, в котором каждый из двух процессов, блокирующих два некоторых ресурса, ожидает освобождения ресурса, принадлежащего другому (вообще говоря, их может быть и больше, но простейшая модель такова).

26 Некоторые способы реализации взаимного исключения: семафоры Дейкстры, мониторы, обмен сообщениями.

Первым и простейшим методом синхронизации, которые мы рассмотрим, будут семафоры Дейкстры. Семафор – это разделяемая целочисленная переменная, над которой определены две атомарные операции: up и down. Он работает по следующей схеме: если при выполнении операции down значение семафора будет меньше нуля, процесс, выполняющий операцию, блокируется. При выполнении операции up значение семафора увеличивается на необходимое значение, и какой-то из заблокированных при выполнении операции down процессов продолжит свою работу. (отметим, что есть и концепции семафоров, допускающие отрицательные значения – это используется в более продвинутых синхронизационных задачах).

Хотя в книге и дается другое определение, можно неформально сказать, что монитор – это некоторый вид класса со строго инкапсулированными методами. Говорят, что процесс находится внутри монитора, если в текущий момент времени процесс использует какой-то из этих методов. Особенность монитора в том, что находиться внутри него может единовременно только один процесс. В случае попытки использования занятого монитора может возникнуть жесткая или мягкая блокировка.

Последнее средство, рассмотренное в курсе – это очереди сообщений. Это обобщённое средство, так как, во-первых, синхронизация – это не единственная

его задача, во-вторых, с его помощью можно реализовать оба вышеперечисленных средства. В интерфейсе этого средства нас интересуют 4 основных метода – блокирующие и неблокирующие приемы и передачи сообщений. Именно на них можно выстраивать разные синхронизационные модели. Стоит также отметить, что различают прямую отправку сообщений – то есть, конкретному адресату, и косвенную – то есть, сброс сообщения в некоторый общий пул для его считывания каким-либо из получателей.

27 “Обедающие философы”.

Модельная задача о философах ставится так: есть некоторый круглый стол, за которым сидят философы, между каждыми двумя философами лежит вилка, в центре стоит тарелка со спагетти. Каждый философ ведет себя следующим образом: некоторое время размышляет, затем берет две вилки по обе стороны от себя и некоторое время ест, после этого цикл повторяется. Философы ведут себя независимо друг от друга. Необходимо предложить метод синхронизации, исключающий тупиковые ситуации.

Краткое описание алгоритма:

Создается массив семафоров, символизирующий философов. Также создается массив состояний для философов, его элементы принимают значения для состояний бездействия, ожидания ресурса и использования ресурса. Создается также один двоичный семафор. Далее, цикл работы философа выглядит следующим образом: философ думает, затем пытается взять вилки, что заключается в смене состояния на ожидание ресурса. После этого проверяется состояние соседей: если оба они не едят, то в состояние использования переключается текущий процесс, в противном случае процесс блокируется. После того, как философ поел, производится последовательность действий по освобождению вилок: состояние переключается на бездействие, а соседние процессы, которые ожидали освобождения, получают доступ к вилкам.

Отметим, что вышеупомянутый двоичный семафор здесь использовался в функциях освобождения и занятия вилок, чтобы исключить одновременное разрешение на взятие для двух процессов.

28 “Читатели и писатели”.

Задача о читателях и писателях заключается в следующем: существует некоторая информационная система и два типа процессов для работы с ней: читающие и пишущие. В произвольный момент времени читать информацию может любое количество процессов, но в случае, если писатель начнет свою работу, все остальные процессы (обоих видов) будут заблокированы. Предложить метод организации такой системы.

Краткое решение: будем ставить читателей в приоритет писателям, то есть, писатели должны будут ожидать, пока читатели выйдут из системы. Для этого потребуется два двоичных семафора и глобальная переменная-счетчик читателей. Первый семафор будет использоваться, чтобы корректно вести подсчет

процессов-читателей, второй – чтобы блокировать доступ к системе в случае, когда первый читатель заходит в нее. Иными словами, цикл работы читателя выглядит так: в начале захватывается семафор на переменную-счетчик, она инкрементируется, в случае равенства ее 1 (первый читатель) доступ к системе блокируется. Далее читатель выполняет свою работу, декрементирует счетчик, в случае, если он последний закончивший читатель, доступ к базе открывается. Цикл жизни писателя выглядит намного проще: он захватывает семафор системы, работает, затем выходит из нее, отпуская семафор.

29 «Спящий парикмахер».

Пусть есть парикмахерская, в которой работает один цирюльник. В парикмахерской также есть N мест для ожидания. Если парикмахер не обслуживает клиента, он спит, и ожидающему клиенту надо его разбудить. Если в зале ожидания не хватает места, то клиенту в услуге будет отказано. Необходимо, опять же, запрограммировать корректное поведение всех сторон.

Создадим семафор с количеством клиентов и обычную переменную с тем же значением. Также создадим семафор для состояния парикмахера. Цикл для решения выглядит так: если клиентов нет, блокируемся, в противном случае уменьшаем количество ожидающих клиентов на 1 и выставляем готовность к работе. Какой-то из клиентов займет парикмахера и будет обслужен, затем все повторится.

30 Сигналы. Примеры программирования.

Сигналом называется один из системных методов воздействия одних процессов на другие. Аппарат сигналов родственен аппарату прерываний в том, что он также используется для извещения о происхождении какого-либо события. При получении процессом сигнала может произойти несколько ситуаций: сигнал может быть обработан по умолчанию, проигнорирован или обработан функцией-обработчиком.

Для работы с сигналами в UNIX существует несколько системных вызовов. Во-первых, чтобы послать сигнал процессу по его номеру, используется вызов `kill()`. Во-вторых, для установления обработчика есть вызов `signal(int signo, void *(int) Hndl)`, который установит в качестве обработчика для сигнала с номером из первого аргумента функцию Hndl или будет блокировать или игнорировать его в случае подачи соотвествующих констант. Существует также более современный вызов `sigaction()`, который, в отличие от первого, не сбрасывает обработчик при срабатывании и вообще позволяет более тонко работать с сигналами.

31 Неименованные каналы. Примеры программирования.

Неименованным каналом, или трубой, или pipe-ом, называется организованная по принципу FIFO система, позволяющая передавать информацию разным процессам друг другу.

В UNIX пайпы реализуются через пару файловых дескрипторов: один используется для записи информации, другой – для чтения. Для создания такой пары используется системный вызов `pipe()`. Стоит четко понимать, что между каналами и регулярными файлами существует довольно много отличий: к каналам можно обращаться только через ФД, каналы не являются файлами произвольного доступа, канал существует в системе до тех пор, пока существуют использующие его процессы.

32 Взаимодействие процессов по схеме "подчиненный-главный". Общая схема трассировки процессов.

Необходимость в такой схеме взаимодействия возникает главным образом при отладке.

В ОС UNIX для организации такой схемы используется функция `ptrace()`, которая позволяет читать данные из сегментов кода и данных отслеживаемого процесса, некоторые данные из его контекста, в состоянии останова модифицировать эти данные, исполнять процесс в пошаговом режиме.

Отметим, что в UNIX трассировка может вестись только между родителем и его потомком (прямым) и только после того, как тот даст на это разрешение.

Общая схема работы выглядит так: выполнение подчиненного процесса приостанавливается при получении им какого-либо сигнала, а также при вызове `exec()`. Если контролирующий процесс при этом выполняет вызов `wait()`, он немедленно возвращает управление.

33 Именование разделяемых объектов. Очереди сообщений. Пример.

В системе SystemV был введен интерфейс IPC, позволяющий обмениваться информацией произвольным процессам, не обязательно являющимся родственными.

Этот интерфейс включает в себя объекты трех типов: очередь сообщений, массив семафоров и разделяемая память. Для обращения к объектам существует унифицированная схема именования по ключу.

Вообще говоря, ключи IPC могут выбираться произвольным образом, но для исключения коллизий в UNIX существует специальный вызов `ftok(char *filename, int proj)`. Этот вызов является, по сути, некоторой хэш-функцией,

генерирующей ключ на основании системных характеристик файла вроде номера индексного дескриптора или устройства, на котором он расположен (отсюда следует, что файл должен существовать).

Итак, очередь сообщений IPC – объект, который, в целом, описывается своим названием (FIFO-организованная структура передачи информации). Однако стоит отметить одну важную деталь: сообщения в очереди типизированы, то есть, при вызове функции-передатчика `msgsnd()` необходимо передать ей кроме самого сообщения также его тип, что реализовано в языке Си специальной структурой.

Функция приёма сообщений `msgrcv()`, в свою очередь, может быть настроена таким образом, что будет принимать сообщения конкретного типа или сообщения множества типов – это контролируется одним из ее аргументов. Таким образом, очередь сообщений представляет по сути не очередь в классическом понимании, а, скорее, набор очередей, определяемый множеством типов.

34 Именование разделяемых объектов. Разделяемая память. Пример.

(повторить первые 3 абзаца из 33)

Разделяемая память, опять же – достаточно самоочевидный объект. В UNIX для работы с разделяемой памятью необходимо сначала получить к ней доступ по ключу, затем с помощью специальной функции `shmat()` ассоциировать ее с существующем в виртуальном адресном пространстве процесса указателем.

После окончания работы с помощью другой специальной функции `shmdt()` ассоциация удаляется.

35 Именование разделяемых объектов. Массив семафоров. Пример.

(то же самое, что в 34)

Массив семафоров IPC устроен с точки зрения создания/получения так же, как и остальные IPC-объекты. Контроль за этим массивом ведется следующим образом: каждый его элемент можно проинициализировать с помощью функции `semctl()`, а операции на его элементах производить функцией `semop()`. В смысле синхронизации он полностью реализует концепцию семафора Дейкстры.

36 Коммуникационный домен. Схема работы с сокетами с установлением соединения. Пример программирования с установлением соединения.

Как было сказано ранее, коммуникационным доменом называется совокупность правил именования и области взаимодействия сокетов. При программировании сокетов возможны 2 основных типа соединений: с использованием виртуального канала и датаграмм.

Выше уже было описано, что каждое из них из себя представляет, здесь это остается актуальным.

Интерфейс создания сокета в Си следующий: вызывается функция `socket(int domain, int type, int socket)`, возвращающая файловый дескриптор сокета в случае успеха.

Сокет предварительно устанавливает соединение в случае, если он имеет тип виртуального канала. В случае датаграммы соединение, как правило, устанавливается каждый раз, когда передается какое-либо сообщение.

После создания сокета необходимо произвести связываение – ассоциацию этого сокета с каким-либо конкретным адресом. В UNIX для этого используется функция `bind()`.

Далее, клиент-сторона отправляет по сокету запрос на соединение (connect()). Сервер-сторона при этом принимает запрос (accept()) и начинается обмен сообщениями с помощью функций send и recv.

После обмена сокет закрывается функцией shutdown.

37 Схема работы с сокетами без установления соединения. Пример программирования без установления соединения.

Все сказанное про связывание и создание из прошлого вопроса остается актуальным. Для программирования сокетов без установления соединения используются функции sendto и rcvfrom, позволяющие указывать реквизиты адресатов прямо на месте.

38 Мьютексы. Пример синхронизации нитей.

В программировании нитей одну из важнейших ролей играют мьютексы – более упрощенный вид двоичного семафора Дейкстры, позволяющий реализовывать взаимное исключение в нитях.

Основное их отличие от обычного семафора Дейкстры заключается в том, что отпустить мьютекс может только та нить, что его захватила – при попытке отпустить мьютекс из другой нити может произойти UB или ошибка (в зависимости от атрибутов мьютекса).

Влияние же нитей на мьютексы, захваченные в других, возлагается на условные переменные (condvar), которые в этом курсе лекций рассмотрены не были.

В остальном мьютексы работают аналогично двоичным семафорам – над ними определены атомарные операции мягкого и жесткого захвата и поднятия.

39 Общая классификация средств взаимодействия процессов в ОС UNIX.

Честно говоря, не вполне понятно, что здесь говорить. Я бы разделил средства взаимодействия на те, что позволяют работать друг с другом родственным процессам: пайпы, mmap, отслеживание и те, что позволяют взаимодействовать произвольным: IPC, сокеты, сигналы, FIFO-файлы.

40 Структурная организация файлов. Атрибуты файлов. Основные правила работы с файлами. Типовые программные интерфейсы работы с файлами.

Для описания структуры файла существует несколько моделей. Во-первых, это модель последовательности байт – система возлагает задачу интерпретации содержимого полностью на пользователя, трактуя файлы лишь как абстрактный набор байтов. Сегодня такая модель пользуется достаточно сильной известностью. Во-вторых, файл можно представить как последовательность записей переменной и постоянной длины – такая интерпретация хорошо ложится на аппаратную реализацию, так как, по сути, отображается в те же перфокарты или магнитные ленты. В-третьих, существует иерархическая модель, в которой файл рассматривается как некоторая сложная структура, позволяющая организовывать динамическую работу с данными. Такая модель требует достаточно сложной реализации.

Совокупность параметров файла, описывающих его свойства и состояния, называют атрибутами файла. Основные атрибуты:

- имя
- права доступа
- принадлежность
- тип
- размер блока (если файл имеет блочную организацию)
- размер файла
- указатель чтения/записи.

Основные операции работы с файлами выглядят следующим образом: операция открытия, блок операций, изменяющих его содержимое и атрибуты, операция закрытия.

При открытии файла создается системная файловая структура, называемая файловым дескриптором – она содержит информацию об актуальном состоянии открытого файла.

При закрытии файла система уведомляется о закрытии файлового дескриптора, связанного с файлом (поскольку файл может быть открыт несколько раз даже в рамках одной программы, и на каждое открытие будет выделен свой дескриптор).

41 Модели реализации файловых систем. Понятие индексного дескриптора.

Мы будем основываться на предположении о том, что каждое системное устройство имеет унифицированную структуру. Конкретно, каждое устройство в начальном блоке имеет так называемый основной программный загрузчик (MBR), инициализирующий начальные данные системы. Им может быть как загрузчик одной операционной системы, так и некоторый обобщенный, позволяющий выбирать из нескольких установленных на компьютере систем.

Следом за загрузчиком в устройстве идет таблица разделов (и непосредственно они), каждый из которых состоит из загрузчика ОС, суперблока, свободного пространства, файлов.

Существует несколько подходов к реализации файлов. Самый тривиальный из них – модель непрерывных файлов, в соответствии с которой файлы размещаются в непрерывных областях памяти. Такой подход очевидным образом влечет проблему фрагментации.

Далее предлагается модель организации в виде связного списка – то есть, каждый блок файловой системы будет также содержать ссылку на следующий. Недостатки такой реализации состоят в отсутствии прямого доступа, фрагментации файла по диску и хранении в блоке системной информации.

Еще одна модель, призванная решить эти проблемы, называется File Allocation Table (FAT) и устроена следующим образом: для каждого файла в каталоге хранится начальная позиция в таблице, в этой позиции хранится информация об актуальном состоянии блока и о следующей позиции, в которой хранится то же самое. Индикатором конца является нулевая ссылка на следующий блок. Такой подход позволяет организовывать прямой доступ и решает проблему хранения системной информации в блоке. Недостатком такой таблицы является ее размер, ведь он должен быть равен количеству блоков, доступных для хранения файлов, и при больших объемах данных держать такую таблицу, например, в оперативной памяти становится весьма проблематичным.

Наконец, существует концепция индексных дескрипторов, подразумевающих добавление к атрибутам файла список блоков, хранящих его. То есть, при открытии файла мы сразу получаем список номеров блоков, но, опять же, при больших размерах файла размер индексного дескриптора будет приближать-

ся к размеру FAT. Эта проблема решается введением уровней косвенности в индексном дескрипторе: то есть, какое-то количество блоков будет содержаться непосредственно в нем, а после этих блоков, если файл не исчерпывается, будут содержаться уже ссылки на остальные. Такой подход позволяет при фиксированном размере индексного дескриптора существенно увеличить предельный размер файла.

42 Координация использования пространства внешней памяти. Квотирование пространства ФС. Надежность ФС. Проверка целостности ФС.

При решении вопроса о координации с внешней памятью встают две проблемы: выбор размера блока и метод учета свободных блоков. Первая из них достаточно ситуативно решаема, то есть, в зависимости от параметров системы и ожидаемых с ней действий, можно выбирать размер блока так, как это было бы наиболее эффективно.

Что касается метода учета свободных блоков, существует два основных подхода: использование связных списков и битовых массивов. Первый из них предполагает хранение головы списка, например, в оперативной памяти для быстрого доступа, а все остальные номера – в каком-то количестве блоков файловой системы. Учет же в виде битового массива предполагает расчет нужного количества разрядов и быструю адресацию по ним.

Так как мы говорим о многопроцессных и многопользовательских системах, естественным образом встает необходимость контроля используемых блоков и имен файлов. Здесь эффективной себя показывает модель гибких и жестких лимитов на блоки и файлы: в системе для каждого пользователя ведется подсчет файлов и блоков и устанавливается два предельных значения: жесткое и гибкое. При превышении гибкого лимита пользователю выдается предупреждение, при превышении жесткого лимита пользователь блокируется.

Еще один вопрос, стоящий рассмотрения – устойчивость системы к аварийным ситуациям. Надежность системы вообще включает в себя множество критериев, но главным из них является наличие некоторой избыточной информации, позволяющей свести потенциальные потери к минимуму.

Приведем основные подходы к архивации данных. Избирательной архивацией называется подход, при котором не копируются заведомо восстанавливаемые файлы (например, дистрибутивы ОС). Инкрементным архивированием называется подход, при котором в начальный момент времени создается мастер-копия – копия всех данных, а затем с некоторым периодом сохраняется информация об изменениях в системе.

Для проверки целостности ФС используется две системные таблицы: свободных блоков и занятых (размер – количество блоков, начальные значения – нули). Система вначале производит цикл по структуре, учитывающей свободные блоки и для каждого встреченного свободного блока увеличивает запись в соотвествующей таблице на 1. Далее посредством цикла по массиву индексных дескрипторов та же операция производится на таблице занятых блоков. Далее

таблицы суммируются по ячейкам, если каждая сумма равна 1, система считается исправной. Если какая-то из сумм равна 0, считается, что блок был утерян из списка свободных (некритичная ситуация). Если в таблице свободных блоков находится число, большее 1, считается, что список свободных блоков был нарушен и происходит пересоздание. Если аналогичная ситуация происходит с таблицей занятых блоков, считается, что какие-то два файла владеют одним блоком и ошибка обрабатывается ситуативно.

43 Организация файловой системы ОС UNIX. Виды файлов. Права доступа. Логическая структура каталогов.

Файлы в ОС UNIX имеют следующие типы: регулярный, каталог, устройство, FIFO, сокет, ссылка. Права доступа определяются для каждого файла и разделяются на чтение, запись, исполнение для пользователя, группы и остальных пользователей. Приведем краткое описание содержимого корневого каталога в UNIX-системах:

- bin – исполняемые файлы для команд общего пользования
- etc – системные таблицы, команды для работы с ними
- tmp – временные файлы
- mnt – монтирование других ФС к данной
- dev – устройства
- usr – пользовательская информация.

44 Внутренняя организация ФС. Модель версии UNIX SYSTEM V.

Файловая система s5fs состоит из суперблока, области ИД и собственно блоков файлов. Суперблок состоит из:

- тип ФС
- размер ФС в блоках
- размер области ИД
- число свободных блоков
- число свободных ИД
- флаги

- размер логического блока
- список номеров свободных ИД
- список адресов свободных блоков

Структура ИД описана выше, область блоков файлов содержит собственно их и некоторую системную информацию, не поместившуюся в суперблок.

45 Внутренняя организация ФС. Принципы организации файловой системы FFS UNIX BSD.

В этой системе разделом считается последовательность дисковых цилиндров, разбитых на порции фиксированного размера. В каждую из таких порций, называемых кластерами, помещается копия суперблока, блоки файлов, информация об индексных дескрипторах и свободных ресурсах.

В данной файловой системе используются три основных стратегии размещения: оптимизация размещения каталога (т.е., выбираются кластеры, у которых количество ИД превосходит среднее, из них выбирается тот, в котором количество каталогов минимально); равномерность использования блоков данных (то есть, файл делится на несколько частей, блоки прямой адресации располагаются в том же кластере, что и ИД, а остальные блоки равномерно распределяются по остальным кластерам); размещение блоков файла на диске с некоторым зазором (это сделано для того, чтобы при операции считывания двух соседних блоков за время между выходом из чтения первого и входом в чтение второго диск не успел заехать на начало второго блока, что привело бы к необходимости лишнего оборота).

46 Архитектура организации управления внешними устройствами, основные подходы, характеристики.

Многое из того, что надо сказать в этом вопросе, уже сказано в одном из первых. Здесь, наверное, имеет смысл добавить следующее: среди целей программного управления внешними устройствами можно выделить унификацию интерфейсов, обеспечение конкретной модели синхронизации, обработка ошибок, буферизация, обеспечение стратегии доступа, планирование обменов.

47 Планирование дисковых обменов, основные алгоритмы.

Выделяются следующие алгоритмы для обработки очередей запросов к дорожкам дискового устройства:

- FIFO – по сути, прямой обход очереди в порядке ее следования
- LIFO – обход очереди в обратном порядке
- жадный алгоритм – на каждой итерации выбираем ближайшую дорожку
- PRI – учитывание приоритетов процессов, запрашивающих обмены
- SCAN – проход сначала до одного края диска, затем – до другого
- C-SCAN – проход от минимальной по номеру дорожки до конца диска
- N-step-SCAN – очередь запросов делится на несколько подочередей, обрабатываемых с ситуативным выбором алгоритма.

48 Организация RAID систем, основные решения, характеристики.

RAID-система — это совокупность физических дисковых устройств, которая представляется в операционной системе как одно устройство, имеющее возможность организации параллельных обменов. Помимо этого образуется избыточная информация, используемая для контроля и восстановления информации, хранимой на этих дисках.

Рассмотрим основные модели RAID:

- RAID 0 – решение, родственное расслоению ОЗУ, при котором соседние полосы хранятся на соседних дисках, что позволяет быстро считывать непрерывные порции данных с нескольких дисков.
- RAID 1 – зеркалирующее устройство, хранящее и поддерживающее две копии данных.
- RAID 2 – устройство, хранящее избыточную информацию в виде кода Хэмминга, что позволяет корректировать ошибки.
- RAID 3 – устройство, хранящее сумму по модулю 2 данных, что позволяет восстановить их в случае ошибки.
- RAID 4 – схожее с RAID 3 устройство, требующее ручного поддержания диска четности (не синхронизирован с дисками данных).
- RAID 5 – то же, что и RAID 4, но диски чётности хранятся в другой, более эффективной геометрии.
- RAID 6 – на ряд полос использует по два диска четности для пущей устойчивости.

49 Типы устройств, файлы устройств, драйверы.

Все устройства в ОС UNIX делятся на два типа: блок- и байт-ориентированные. Обмен данными с первыми ведется фиксированными порциями байт, со вторыми – произвольным их количеством. В системе тип взаимодействия устройства и информация о нем определяются драйвером.

Драйвера в системе регистрируются через две таблицы: bdevsw, cdevsw для блок- и байт-ориентированных соотвественно.

Как было отмечено ранее, файлы устройств хранятся в каталоге /dev. Каждый файл устройства характеризуется двумя числами: номером драйвера в таблице и некоторой системной константой, позволяющей дифференцировать работу с устройством при, например, использовании одного драйвера для нескольких устройств.

50 Системная организация обмена с файлами. Буферизация обменов с блокоориентированными устройствами.

Для организации обменов с файлами в UNIX реализовано несколько системных структур и таблиц. ТОФ (таблица открытых файлов) – это системная таблица, создаваемая для каждого запущенного в системе процесса. Ее размер указывается при инициализации системы и обозначает лимит открытых файлов для одного процесса. Эта таблица, как следует из названия, хранит записи на каждый открытый в процессе файл. В записи ТОФ содержится ссылка на запись в ТФ – таблице файлов. Это системная таблица, хранящая информацию о всех открытых в системе файлах, в частности, ссылки на еще одну таблицу – ТИДОФ – таблицу индексных дескрипторов открытых файлов.

Отметим, что при порождении дочернего процесса этот процесс наследует ТОФ отца, при этом в ТФ у соотвествующей записи счетчик кратности будет увеличен. Однако повторное открытие файла из того же процесса или открытие его из другого приведут к созданию новой записи в ТФ со своим файловым указателем и счетчиком кратности. Запись в ТИДОФ же хранит, ко всему прочему, актуальную копию индексного дескриптора файла и количество записей, соотвествующих ему в ТФ. ТФ и ТИДОФ сохраняются в оперативной памяти.

Буферизацией называется метод оптимизации работы с блочными устройствами, состоящий в том, что при обращении к функциям ввода-вывода устройства данные не передаются сразу на него, а записываются в некоторый буфер, размер которого обычно равен блоку. В определенный момент буфер сбрасывается на устройство, то есть, например, 5 последовательных команд печати повлекут за собой не 5 обращений к интерфейсу устройства, а сохранение выводимых данных в системный буфер и сброс его на устройство в конце выполнения. Отметим, что буферизация в некоторых случаях нарушает необходимый вывод программы, а то и вовсе может заклинить ее, поэтому очень важно учитывать ее при проектировании и сбрасывать или отключать буферы при необходимости.

51 Одиночное непрерывное распределение. Распределение разделами. Распределение перемещаемыми разделами.

Концепция одиночного непрерывного распределения памяти состоит в том, что все адресное пространство разделяется на две компоненты – та, в которой находится ОС и та, в которой находятся иные процессы. Это одна из самых простых концепций разделения памяти.

Распределение разделами состоит в том, что, опять же, адресное пространство делится на часть ОС и часть прикладных процессов, и пользовательская часть еще делится на N частей, называемых разделами. Далее есть два подхода обработки входной очереди процессов – она либо разбивается на N подочередей по, допустим, минимальному размеру раздела, необходимого для умещения того или иного процесса, либо остается единой очередью и обрабатывается на лету, ситуативно. Основной недостаток такого подхода заключается в фрагментации разделов, также стоит отметить, что при таком подходе максимальный размер процесса будет ограничен максимальным размером раздела.

Распределение перемещаемыми разделами развивает предыдущую модель, позволяя через перемещение разделов (а, следовательно, и процессов) производить дефрагментацию пространства. Однако же в таком случае проблемой встают накладные расходы, возникающие при этой дефрагментации.

52 Страничное распределение.

Определение страничного распределения и мотивация к его использованию были даны в начале.

В современных системах мы не можем себе позволить хранить таблицу страниц полностью в оперативной памяти или, тем более, в кэш-памяти, ведь ее размеры для этого слишком велики. Поэтому используется так называемая TLB-таблица (Translation Look-aside Buffer) – некоторого рода кэш, хранящий оперативно необходимые страницы и обновляемый по необходимости.

Более эффективно работать со страницами позволяет также иерархическая модель организации – при таком подходе виртуальный адрес разбивается на несколько полей, каждое из которых кодирует смещение относительно таблицы страниц определенного уровня. Так, например, при двухуровневой организации необходимо выделить из адреса номер таблицы второго уровня, обратиться к ней через таблицу первого уровня и из нее уже извлечь необходимый физический адрес.

Рассмотрим также еще несколько решений к организации таблиц страниц. Инвертированная таблица страниц при адресации опирается на PID процесса, использующего ее. В таком подходе используется единственная системная таблица страниц. Можно также преобразовывать виртуальные адреса посредством использования хэш-функций.

53 Сегментное распределение.

Сегментное распределение концептуально очень похоже на страничное, но с той лишь разницей, что у сегмента может быть произвольный размер. В такой схеме виртуальный адрес состоит из номера сегмента и смещения относительно его начала. При преобразовании по номеру сегмента в специальной таблице сегментов находится его базовый адрес, а также размер. Если смещение не превышает размера, возвращается смещение относительно базы, в противном случае бросается прерывание по защите памяти.

Такой подход к распределению мотивируется в основном тем, что жесткое ограничение на размер страниц при страничном распределении в некоторых случаях доставляет большие неудобства.

54 Кэширование информационных потоков на уровнях аппаратуры и ОС.

Прим.: никакой информации об этом в машбуке нет, как и в лекциях, стоит на консультации уточнить, откуда вообще взялся этот вопрос, здесь же приведу краткие сведения из методички по кэшированию с джаффара.

Кэш – это высокоскоростная структура данных, призванная повысить эффективность обработки информационных потоков через хранение часто используемых ее фрагментов в быстром доступе.

Это достаточно сложная структура данных, существует несколько классических подходов к ее реализации, а именно: кэш прямого отображения, полностью и частично ассоциативный кэш, многоканальный кэш. Мы не будем здесь вдаваться в подробности организации этих видов.

Как известно из курса, кэширование достаточно часто используется при работе с ОЗУ – таким образом засчет более быстрой памяти повышается скорость обработки. Кроме того, буферизация при работе с внешними устройствами также является фактически кэшированием.

Рассмотрим несколько стратегий по вытеснению элемента из кэша:

- LRU (least recently used) – из кэша вытесняется элемент, с момента использования которого прошло наибольшее количество времени
- LFU (least frequently used) – из кэша вытесняется элемент с наименьшим количеством обращений
- элемент для вытеснения выбирается случайно
- MRU (most recently used) – в противовес LRU, из кэша вытесняется элемент, использованный последним.

Комбинированием нужных методов организации и стратегий вытеснения система добивается наибольшей эффективности.

55 Язык программирования C. Общая характеристика. Типы, данные, классы памяти. Правила видимости. Структура программы. Препроцессор. Интерфейс с ОС UNIX.

https://youtu.be/DWSlOCEzRGo?si=torQzIlocckKefC5