# Процессы и потоки

1. Что такое процесс?

Экземпляр выполняемой программы, включая текущее значение счетчика команд, состояние регистров, адресное пространство.

1. Как происходит работа с процессами?

Псевдопараллельное при наличии в системе одноядерного процессора и параллельно при многопроцессорной\многоядерной системе.

1. Чем отличается процесс от программы?

Программа - набор инструкций, в то время как процесс - выполняемый набор инструкций

1. События для создания процессов(4 варианта)

* Инициализация системы
* Вызов от работающего процесса
* Запрос пользователя
* Инициализация пакетного задания

1. Демоны - что такое?

Фоновые процессы для обработки активной деятельности

1. Виды завершения процесса:

* Обычный выход(добровольно)
* При ошибке(добровольно)
* Фатальная ошибка(принудительно)
* Уничтожение другим процессом(принудительно)

1. Виды состояний процесс

* Выполняемый
* Готовый
* Заблокированный



1. Таблица процессов

Таблица, где каждая запись содержит информацию о контексте определенного процесса.

1. Средства взаимодействия процессов(IPC):

* Сигналы

Сигнал – это нечто, что может быть послано процессу системой или

другим процессом. С сигналом не связано никакой информации, кроме номера, указывающего, какой именно тип сигнала посылается.

При получении сигнала процесс прерывает свою текущую работу и переходит на выполнение функции, определенной как обработчик сигналов данного типа.

* Сообщения

Сообщения не прерывают работу процесса-получателя. С сообщением может быть связана информация, передаваемая получателю. Этот метод взаимодействия процессов использует два примитива, send и receive, которые являются системными вызовами.

* Разделяемая память

Два или более процессов могут совместно использовать некоторую область адресного пространства

* Каналы (трубы), это псевдофайл, в который один процесс пишет, а другой читает.

При использовании программных каналов для выполнения обмена

используются функции чтения/записи в файл.

* Сокеты - поддерживаемый ядром механизм, скрывающий особенности среды и позволяющий единообразно взаимодействовать процессам, как на одном компьютере, так и в сети.
* Почтовые ящики (только в Windows), однонаправленные, возможность широковещательной рассылки.
* Вызов удаленной процедуры, процесс А может вызвать процедуру в процессе В, и получить обратно данные.

1. Векторные прерывания, что такое?

Запись в таблице векторов прерываний, которая связывает адрес обработчика прерывания с номером прерывания.

1. Что такое поток?

Наименьшая единица обработки последовательности команд.

1. Преимущества использования потоков

1.Использования общего адресного пространства.

2. Быстрота создания потока, по сравнению с процессом.

3. Возможность одновременно выполнять вычисления на процессоре и операцию ввода/вывода.

1. Чем отличается процесс от потока?

* Процесс группирует ресурсы в единый образ
* Поток - сущность, разделяемая для работы на ЦП

1. Какие элементы могут принадлежать процессу/потоку?

| **Элементы процесса** | **Элементы потока** |
| --- | --- |
| Адресное пространство | Счетчик команд |
| Глобальные переменные | Регистры |
| Открытые файлы | Стек |
| Дочерние процессы / потоки | Состояние |
| Необработанные аварийные сигналы |  |
| Сигналы и их обработчики |  |
| Информация об использовании ресурсов |  |

1. Множества разделяемых ресурсов для процесса и потоков. В чем отличия между ними?

Потоки в рамках одного процесса разделяют общее адресного пространства, глобальные переменные, различную служебную информацию (сигналы).

Процессы разделяют процессорное время и оперативную память

Основное различие в том, что у потоков общее адресное пространство.

1. Атрибуты в составе блока управления процессом

* Регистры
* Счетчик команд
* Указатель стека
* Состояние процесса
* Приоритет
* Параметры планирования
* Идентификатор процесса
* Привилегии процесса
* Номер процесса
* Учетная информация
* I / O Информация о состоянии - список устройств ввода / вывода, назначенных
* Информация о межпроцессном взаимодействии - флаги, сигналы и сообщения, связанные с обменом данными между независимыми процессами

1. Проблемы потоков

* Проблема с дочерними процессами( передача родительских ресурсов ).
* Совместное использование областей памяти.
* Состояние гонки
* Взаимная блокировка

1. Где могут быть реализованы потоки?

Пользовательское пространство / ядро ОС.

1. Каковы преимущества организации решения задачи на компьютере в варианте приложения с несколькими потоками по сравнению с вариантом с несколькими процессами.

* Использование единого адресного пространства !
* Легкость создания
* Возможность одновременно выполнять вычисления на процессоре и операцию ввода/вывода.
* Возможность параллельных вычислений.

1. +/- реализации потоков на уровне пользователя

| ***+*** | ***-*** |
| --- | --- |
| Реализуем в системе не поддерживающей  потоки.(потоки как библиотека) | Системный вызов блокирует не только работающий поток, но и все потоки того процесса, к которому он относится; |
| Управление потоками возлагается на программу пользователя. | Нужна таблица потоков |
| Возможность использования любых алгоритмов планирования потоков | Внутри одного потока нет прерываний по таймеру |
| Скорость переключения между потоками выше | ОС может принимать плохие решения по управлению потоками. |
| Не требует обращения к ядру |  |

1. +/- реализации потока на уровне ядра

| + | - |
| --- | --- |
| Планирование нескольких потоков одного процесса на разных процессорах ! | Создание/удаление требует больших затрат. |
| Эффективное управление потоками | Неизвестно какой поток будет обрабатывать сигнал. |
| При блокировании одного из потоков процесса ядро может выбрать другой поток этого же (или другого) процесса |  |

1. Активация планировщика что такое и в чём главная суть?

Имитация функциональности пото­ков ядра, но с большей производительностью и гибкостью, свойственной потокам уровня пользователя, достигается избавлением от ненужных переходов между пространствами.

Суть: если ядро знает что поток заблокирован, оно уведомляет процесс, передавая через

стек, в качестве параметров номер потока и описание события.

1. Какой аппаратный компонент необходим для реализации вытесняющей многозадачности?

Аппаратный таймер, обработка прерывания которого обеспечивает возможность операционной системе осуществлять переключение между процессами.

1. Что такое вытесняющая многозадачность?

Это вид многозадачности, при которой планировщик выбирает процесс и предоставляет ему возможность работать до истечения некоторого строго определенного периода времени. Если до окончания этого периода он все еще будет работать, планировщик приостанавливает его работу и выбирает для запуска другой процесс.

1. Многозадачность -возможность операционной системы параллельного выполнения нескольких программ (процессов).
2. Что такое кооперативная многозадачность?

Это вид многозадачности, при котором планировщик выбирает запускаемый процесс, а затем даёт ему возможность выполняться до тех пор, пока он не заблокируется, или до тех пор, пока он добровольно не осовобит ЦП.

1. Всплывающие потоки (что такое, преимущества/недостатки)

Событие вызывает новый поток для его обработки

+ Этот поток не имеет истории(стек, регистры…)

- Предварительное планирование. ?

1. Состязательная ситуация (Состояние гонки)

Когда два или более процесса используют общий ресурс, и окончательный результат работы будет не детерминированным из-за борьбы за этот ресурс.

1. Пояснить термин “взаимоблокировка”

ситуация, при которой несколько процессов находятся в состоянии ожидания ресурсов, занятых друг другом, и ни один из них не может продолжать свое выполнение.

1. Для чего предназначен алгоритм банкира и почему его использование на практике ограничено?

Для избежания взаимоблокировок. Проверяет, ведет ли выполнение каждого запроса к безопасному состоянию.

Не используется, потому что обычно для процессов неизвестно максимальное количество необходимых ресурсов. Число процессов динамически изменяется, соответственно изменяются требования к объему ресурсов.

Принципы алгоритма банкира следующие:

* Каждый процесс должен априорно обозначить свои потребности в ресурсах по максимуму.
* Когда процесс запрашивает ресурс, ему, возможно придется подождать (выделение ресурсов по запросу не всегда может произойти немедленно).
* Когда процесс получает требуемые ресурсы, он должен их вернуть системе за ограниченный период времени.

1. Критическая область(секция)

участок исполняемого кода программы, в котором производится доступ к общему ресурсу (данным или устройству), который не должен быть одновременно использован более чем одним потоком выполнения

1. Условия отсутствия состояния гонки

* Только один процесс может находится в критической секции для ресурса.
* Никакой процесс вне критической области не может блокироваться другим процессом
* Процессы не должны находиться в вечном ожидании входа в критические области

1. Виды взаимных исключений

* Запрещение прерываний
* Блокирующие переменные
* Строгое чередование
* Алгоритм Петтерсона (Команда TSL)

1. Запрещение прерываний

Запрещение прерывания каждого процесса сразу после входа в критическую область и их разрешение сразу же после выхода из критической области.

1. Блокирующие переменные

Программное решение, в котором используется одна общая (блокирующая) переменная, исходное значение которой равно нулю. Когда процессу требуется войти в свою критическую область, сначала он проверяет значение блокирующей переменной. Если оно равно 0, процесс устанавливает его в 1 и входит в критическую область. Если значение уже равно 1, процесс просто ждет, пока оно не станет равно нулю

1. Строгое чередование

Подход будет также использовать общую для обоих процессов переменную с начальным значением 0. Только теперь она будет играть не роль замка(mutex) для критического участка, а явно указывать, кто может следующим войти в него.

1. Алгоритм Петтерсона

Перед входом в свою критическую область каждый процесс вызывает функцию enter\_region(команда TSL(атомарность последовательности команд)), передавая ей в качестве аргумента свой собственный номер процесса. Этот вызов

заставляет процесс ждать, если потребуется, безопасного входа в критическую область. После завершения работы с ресурсом процесс, чтобы показать это и разрешить вход другому процессу, если ему это требуется, вызывает функцию leave\_region.

1. Задача производителя и потребителя

Два процесса используют общий буфер фиксированного размера. Один из них, производитель, помещает информацию в буфер, а другой, потребитель, извлекает ее оттуда. Проблемы возникают в тот момент, когда производителю требуется поместить новую запись в уже заполненный буфер. Решение заключается в блокировании производителя до тех пор, пока потребитель не извлечет как минимум одну запись. Также, если потребителю нужно извлечь запись из буфера и он видит, что буфер пуст, он блокируется до тех пор, пока производитель не поместит что-нибудь в буфер и не активизирует этого потребителя.

1. Что такое условная переменная и какие возможности ей предоставляются?

Условные переменные - механизм, позволяющий потомкам блокироваться до выполнение некоторого условия. Условные переменные и мьютексы всегда используются вместе. Не запоминают сигналы.

1. Что такое семафор и какие возможности им предоставляются (в терминах программного интерфейса)?

Целочисленная переменная для подсчёта количества активаций, отложенных на будущее. Операции над семафором должны быть атомарными.

Предоставляет операции up и down.

* Операция down выясняет, отличается ли значения семафора от 0, если отличается, то она уменьшает это значение на 1 и продолжает свою работу. В противном случае, процесс приостанавливается, не завершая в этот раз операцию down.
* Операция up увеличивает значение, адресуемое семафором на 1. Если с этим семафором связаны один или более приостановленных процессов, способных завершить ранее начатые операции down, система выбирает один из них и позволяет ему завершить его операцию down.

1. Что такое мьютекс и какие возможности им предоставляются (в терминах программного интерфейса)?

Мьютекс — механизм синхронизации, обеспечивающий взаимное исключение исполнения критических участков кода.

Интерфейс по POSIX:

* init - Создание мьютекса.
* destroy - Уничтожение мьютекса.
* lock - Перевод мьютекса в заблокированное состояние (захват мьютекса).
* trylock - Попытка перевода мьютекса в заблокированное состояние, и возврат ошибки в случае, если должна произойти блокировка потока из-за того, что у мьютекса уже есть владелец.
* timedlock - Попытка перевода мьютекса в заблокированное состояние, и возврат ошибки в случае, если попытка не удалась до наступления указанного момента времени.
* unlock - Перевод мьютекса в незаблокированное состояние (отпускание мьютекса)

#### Фьютекс

Фьютекс состоит из очереди ожидания в пространстве ядра, которая присоединена к целому числу в пользовательском пространстве. Несколько процессов или потоков работают с целым числом полностью в пользовательском пространстве и прибегают только к относительно дорогостоящим системным вызовам для запроса операций в очереди ожидания.

1. Монитор

Монитор – это конструкция языка программирования, поддерживающая управляемый доступ к разделяемым данным. Монитор инкапсулирует:

* разделяемые данные;
* функции, использующие разделяемые данные;
* синхронизацию выполнения параллельных потоков, вызывающих указанные функции.

Доступ к данным, расположенным в мониторе, реализуется только посредством вызова предоставленных функций. Только один поток может находиться в мониторе в любой момент времени, если второй поток пытается вызвать метод монитора, он переходит в состояние ожидания до выхода из монитора первого потока.

1. Барьер

Некоторые приложения разбиты на фазы и следуют правилу, согласно которому ни один из процессов не может перейти к следующей фазе, пока все процессы не будут готовы перейти к следующей фазе. Добиться выполнения этого правила можно с помощью барьеров, поставленных в конце каждой фазы. Когда процесс достигает барьера, он блокируется до тех пор, пока этого барьера не достигнут все остальные процессы. Это позволяет синхронизировать группы процессов.

1. Работа без блокировок

Обеспечение, чтобы каждый считывающий процесс читал либо старую, либо новую версию данных.

1. Планировщик

компонент ОС, планирующий и осуществляющий перевод процессов из состояния готовности в состояние выполнения.

1. Ограничения на процесс:

* ограниченный скоростью вычислений
* ограниченный скоростью работы I/O
* ограничение на использование оперативной памяти

1. Момент принятия решения планировщиком (когда планировать?):

* при создании нового процесса (какой выполнять, родительский или дочерний);
* при завершении процесса; Если готовых процессов нет, то запускается холостой процесс, представляемый ОС
* при блокировании процесса;
* при возникновении прерывания (запускать того, кто был активен, или того, кого это прерывание могло разблокировать.)

1. Категории алгоритмов планирования(системы, использующие данные алгоритмы)

* Пакетная: алгоритмы без переключений или с переключениями, но с большим временем, отведенным каждому процессу
* Интерактивная: приоритетные алгоритмы планирования с переклю­чениями, чтобы предотвратить захват процессора одним процессом.
* Реального времени - приоритетность не обязательная; процессы знают, что их время ограничено, и бы­стро выполняют работу, а затем блокируются.

1. Задачи алгоритмов планирования

* Все системы
  + Равнодоступность — предоставление каждому процессу справедливой доли процессорного времени
  + Принудительное применение политики — контроль за выполнением принятой политики
  + Баланс — поддержка занятости всех частей системы
* Системы пакетной обработки данных
  + Производительность — максимальное количество задач в час
  + Оборотное время — минимизация времени между представлением задачи и ее завершением;
* Интерактивные системы
  + Время отклика — быстрые ответы на запросы
  + Пропорциональность — выполнение пожеланий пользователя
* Системы реального времени
  + Окончание работы к сроку — предотвращение потери данных
  + Предсказуемость — предотвращение ухудшения качества в мультимедийных системах.

1. Упреждающее планирование - осуществления прерываний с фиксированной частотой, а не по требованию.
2. Алгоритмы планирования

1. В пакетных системах

* Планирование «первым пришел — первым обслужен» (FCFS)
  + Задания выполняются в порядке поступления.
  + Легко понять и реализовать.
  + Реализация основана на очереди FIFO.
  + Низкая производительность, так как среднее время ожидания велико.
* Планирование Shortest-Job-Next (SJN)
  + Неупреждающий алгоритм планирования
  + Это также называется “первый - самый короткий”, или SJF
  + Лучший подход для минимизации времени ожидания.
  + Процессор должен заранее знать, сколько времени займет процесс.
* Приоритетное планирование
  + Алгоритм без вытеснения
  + Каждому процессу назначается приоритет. Процесс с наивысшим приоритетом должен выполняться первым и так далее.
  + Процессы с одинаковым приоритетом выполняются по принципу «первым пришел — первым обслужен».
  + Приоритет может быть решен на основе требований к памяти, времени или любых других ресурсов.
* Самое короткое оставшееся время(SRTF)
  + Метод планирования, который является упреждающей версией SJN
  + Выполняется процесс с наименьшим количеством времени, оставшимся по определению, и поскольку это время должно только уменьшаться по мере выполнения, процесс будет либо выполняться до его завершения, либо будет вытеснен, если будет добавлен новый процесс, требующий меньшего количества времени.
  + Система также требует очень небольших накладных расходов, поскольку она принимает решение только тогда, когда процесс завершается или добавляется новый процесс
  + Невозможно реализовать в интерактивных системах, где необходимое время ЦП неизвестно.

2. В интерактивных системах

* Round Robin (RR) (Циклическое планирование)
  + Алгоритм планирования вытесняющих процессов.
  + Каждому процессу предоставляется определенное время для выполнения(квант).
  + Как только процесс выполняется в течение заданного периода времени, он прерывается, и другой процесс выполняется в течение заданного периода времени.
* Планирование многоуровневых очередей
  + Для процессов с общими характеристиками поддерживается несколько очередей.
  + Каждая очередь может иметь свои собственные алгоритмы планирования.
  + Приоритеты присваиваются каждой очереди.
* Гарантированное планирование
  + Давать пользователям обещания относительно времени выполнения процессов
* Лотерейное планирование
  + Раздача процессам «лотерейных билетов» и когда приходит время выбирать следующий процесс случайно выбирать из готовых.

1. В каких условиях алгоритм планирования по принципу FIFO будет наиболее эффективен с точки зрения среднего времени отклика и почему?

Время выполнения задач примерно одинаковое.

Если время выполнения задач примерно одинаково, то в других алгоритмах планирования тратится время на поиск процесса, который нужно запустить, а FIFO просто возьмёт первый, который придёт.

1. Предложите не менее двух недостатков алгоритма планирования “первый - самый короткий”

* Все задачи должны быть доступны одновременно.
* Необходимо знать время выполнения задач.
* Он выполняется медленнее, чем SRJF.
* Возможна ситуация ресурсного голодания.

1. Разделение механизма и политики планирования

Предположим, что ядро ОС применяет алгоритм приоритетного планирования, но предоставляет системный вызов, с помощью которого процесс может установить приоритеты своих дочерних процессов. Таким образом родительский процесс может управлять порядком планирования работы дочерних процессов, хотя он сам планированием не занимается.

Механизм находится в ядре, а политика устанавливается пользовательским процессом.

1. Планирование потоков

* Если потоки на уровне пользователя, то система дает время процессу, и он

внутренним планировщиком выбирает поток для выполнения. При этом выбранный

поток будет работать в режиме кооперативной многозадачности, но это не

отразится на других потоках.

* Потоки на уровне ядра выбирает само ядро, что чередует потоки.

1. Задача "Читатели-Писатели" заключается в обеспечении согласованного доступа нескольких потоков к разделяемым данным. Корректное решение должно удовлетворять следующим условиям:

* потоки выполняются параллельно;
* во время выполнения потоком операции записи, данные не могут использоваться другими потоками;
* во время выполнения потоком операции чтения, другие потоки также могут выполнять операцию чтения;
* потоки должны завершить работу в течение конечного времени.

1. Как использовать код возврата программы?

Код возврата программы позволит ОС определить корректность завершения работы данной программы, так как для ОС программа является дочерним процессом. Если же мы самостоятельно создаем в нашей программе дочерний процесс, то это позволит нам уже отследить корректность выполнения дочернего процесса.

1. Функции операционной системы:

- Первая функция – менеджер ресурсов. ОС управляет распределением ресурсов между программами как во времени (устройства ввода-вывода, микропроцессор), так и в пространстве (оперативная память, жесткий диск).

- Вторая функция – виртуальная машина, скрывающая от программиста сложность работы с аппаратными средствами.

1. В чем различие между режимом ядра и пользовательским режимом? Почему это различие представляет важность для операционной системы?

Программы, запущенные в режиме ядра защищены от попыток их изменения на аппаратном уровне. Это очень важно для ОС, т.к. данный режим уберегает ее от несанкционированного вмешательства клиентских приложений.

## Нужные определения

* ***Системный вызов*** — способ обращения программы пользовательского пространства к пространству ядра.
* **Ядро операционной системы** (Kernel) - часть операционной системы: постоянно находящаяся в оперативной памяти, управляющая всей операционной системой, содержащая: драйверы устройств, подпрограммы управления памятью, планировщик заданий, реализующая системные вызовы и т.п
* ***Оборотное время*** — это среднестатистическое время от момента передачи задания на выполнение до момента завершения его выполнения.
* ***Время отклика*** — это время между выдачей команды и получением результата.
* ***Взаимное исключение*** — свойство построения параллельных программ, которое используется в целях предотвращения состояния гонки; Оно требует, чтобы один поток исполнения никогда не входил в критическую секцию одновременно с тем, как другой параллельный поток выполнения вошел в свою критическую секцию.
* ***Пул потоков*** предназначен для упрощения многопоточной обработки. Программист выделяет фрагменты кода, которые можно выполнять параллельно. Планировщик оптимальным образом распределяет рабочие элементы по рабочим потокам пула.

Преимущества:

* Вопросы эффективной загрузки оптимального числа потоков решаются планировщиком.
* Уменьшение накладных расходов, связанных с ручным созданием и завершением потоков для каждого фрагмента кода, допускающего распараллеливание.
* **Счетчик команд** – адрес текущей выполняемой команды процесса; обычно хранится в специальном системном регистре аппаратуры.
* **Атомарная операция** — это операция, которая выполняется за один раз, не прерываясь ничем.(пока она выполняется — система не будет отвлекаться на выполнение каких-либо других операций)
* Интерфейс прикладного программирования (application programming interface – **API**) - набор библиотечных функций, реализующий некоторую функциональность, используемую программой.
* **Блок управления процессом** (Process Control Block – PCB) – системная структура данных, используемая ОС для управления процессом.
* **Диспетчеризация** – поочередное выделение процессам планировщиком операционной системы квантов процессорного времени.
* **Долговременный планировщик** (планировщик заданий) - планировщик ОС, определяющий, какие процессы должны быть перемещены в очередь готовых процессов.
* **Исполняемый** (running) - Состояние процесса, при котором исполняются его команды на процессоре или процессорах компьютерной системы под управлением ОС.
* **Готовый к выполнению** (ready) – Состояние процесса, при котором он ожидает получения ресурсов процессора для его исполнения.
* **Новый** (new) - Состояние процесса, при котором он создан операционной системой, но еще не начал выполняться.
* **Ожидающий** (waiting) – состояние процесса, при котором он ожидает наступления некоторого события, например, завершения ввода-вывода. В состоянии ожидания процесс не занимает процессор.
* **Переключение контекста** (context switch) – действия ОС при переключении с одного процесса на другой по сохранению состояния старого процесса и загрузки сохраненного состояния для нового процесса.
* **Поток ядра** (kernel thread) - низкоуровневый системный поток, поддержанный и использующийся на уровне ядра операционной системы; используется для реализации потоков пользовательского уровня.
* **Волокно (fiber)** — облегченный поток, выполняемый в режиме пользователя.
* **Голодание** (starvation) - ситуация в системе, когда процессы с низким приоритетом длительное время ждут и не получают квантов времени процессора.
* **Регистр процессора** — поле заданной длины во внутрипроцессорной оперативной памяти.
* **Unix** — семейство переносимых, многозадачных и многопользовательских операционных систем, которые основаны на идеях оригинального проекта AT&T Unix.

Операционные системы семейства Unix характеризуются модульным дизайном, в котором каждая задача выполняется отдельной утилитой, взаимодействие осуществляется через единую файловую систему, а для работы с утилитами используется командная оболочка.

# Управление Памятью

1. Концепция замены данных(свопинг)

Операционная система должна сохранить образ некоторого процесса в файле на диске, а затем по потребности загрузить и запустить программу.

1. Статическое перемещение

Выделение ресурсов производится перед выполнением программы, в результате которого программа “привязывается” к определенному месту в памяти вычислительной машины.

1. Недостатки работы без абстракций

Если пользовательские программы будут обращаться к каждому байту памяти, они могут испортить операционную систему, раздробить ее код и довести до остановки работы. Во-вторых, при использовании этой модели сложно организовать одновременную работу нескольких программ.

1. Адресное пространство

Набор адресов, который может быть использован процессом для обращения к памяти. У каждого процесса имеется собственное адресное пространство.

1. Какие недостатки присущи схеме изоляции памяти процесса, использующей два параметра - базу и границу?

* Постоянные операции сравнения и сложения при каждом обращении к памяти.
* Требуется оснащение ЦП двумя специальным регистрами.

1. Динамическое перераспределение памяти(базовый, ограниченный регистры)

Адресное пространство каждого процесса просто проецируется на различные части

физической памяти.

При запуске процесса в базовый регистр загружается физический адрес, с

которого начинается размещение программы в памяти, а в ограничительный

регистр загружается длина программы.

При каждой ссылке процесса на память аппаратура центрального процессора перед выставлением адреса на шине памяти добавляет к адресу, сгенерированному процессом, значение базового регистра. Одновременно аппаратура проверяет, не равен ли

предлагаемый адрес значению ограничительного регистра или не превышает ли он

это значение.

1. Свопинг vs Виртуальная память

* Свопинг заключается в размещении в памяти всего процесса целиком, его запуске на некоторое время, а затем сбросе на диск.
* Виртуальная память позволяет программам запускаться даже в том случае, если они находятся в оперативной памяти лишь частично.

1. Задачи операционной системы по управлению оперативной памятью

* Отображение адресного пространства процесса на конкретные области физической памяти
* Распределение памяти между конкурирующими процессами
* Контроль доступа к адресным пространствам процессов
* Выгрузка процессов во внешнюю память, когда в оперативной памяти недостаточно места
* Учет свободной и занятой памяти

1. Уплотнение памяти

Когда в результате свопинга в памяти создаются несколько свободных областей, их можно объединить в одну большую за счет перемещения при первой же возможности всех процессов в нижние адреса.

1. Управление памятью с помощью битовых матриц

При использовании битовых матриц память делится на единичные блоки размером от нескольких слов до нескольких килобайт. С каждым единичным блоком соотносится один бит в битовой матрице, который содержит 0, если единичный блок свободен, и 1, если он занят (или наоборот).

1. Управление памятью с помощью связных списков

Введение связанных списков сегментов памяти, где сегмент либо содержит процесс, либо является пустым пространством между двумя процессами.

Каждая запись в списке содержит:

* информацию о том является ли область памяти свободной или занятой процессом
* адрес, с которого начинается эта область
* ее длину
* содержит указатель на следующую запись.

1. Какие две структуры данных наиболее широко используются для учета свободного пространства оперативной памяти?

* Битовые матрицы
* Связные списки

Алгоритмы для нахождения свободного участка памяти

1. “Первое подходящее”

Диспетчер памяти сканирует список сегментов до тех пор, пока не найдет пустое пространство подходящего размера.

1. “Следующее подходящее”

Он работает так же, как и «первое подходящее», но отслеживает свое местоположение, как только находит подходящее пустое пространство. При следующем вызове для поиска пустого пространства он начинает поиск в списке с того места, на котором остановился в прошлый раз.

1. “Наименее подходящее”

Выбор самого большого подходящего пустого пространства, чтобы вновь образующееся пустое пространство было достаточно большим для дальнейшего использования.

1. “Наиболее подходящее”

При нем поиск ведется по всему списку, от начала до конца, и выбирается наименьшее соответствующее пустое пространство.

1. "Быстро искомое подходящее”

Его использование предусматривает ведение отдельных списков для некоторых наиболее востребованных искомых размеров.

1. Суть метода виртуальная память

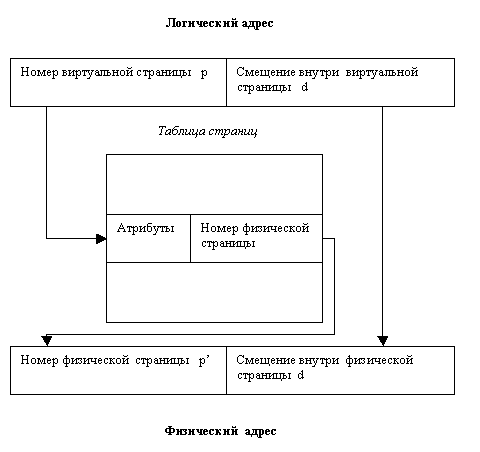
Виртуальная память разбивается на участки, называемые страницами. Каждая страница представляет собой непрерывный диапазон адресов. Эти страницы отображаются на физическую память, но для запуска программы одновременное присутствие в памяти всех страниц необязательно.

1. Оверлей

метод программирования, позволяющий создавать программы, занимающие меньше оперативной памяти, чем установлено в системе. Метод заключается в разделении программы на исполняемые блоки, которые поочередно записываются в одну область памяти поверх предыдущих, выполняют свои функции и перезаписываются следующими.

1. Описание страничной и сегментной организации памяти

* При реализации страничной виртуальной памяти виртуальное адресное пространство делится на части одинакового фиксированного размера, называемые виртуальными страницами. Физическая оперативная память также делится на части такого же размера, называемые физическими страницами. Копия всех виртуальных страниц хранится на диске. Страничное распределение памяти предусматривает только механическое разбиение на страницы.
* При сегментном распределении памяти виртуальное адресное пространство делится на части, называемые сегментами, размер которых определяется смысловым значением содержащейся в них информации. Сегменты никак не упорядочиваются друг относительно друга.
* Виртуальный адрес задается парой чисел: номером сегмента и смещением внутри сегмента. При загрузке процесса в оперативную память помещается только часть его сегментов, полный образ виртуального адресного пространства находится на диске. Для каждого загружаемого сегмента подыскивается непрерывный участок свободной физической памяти достаточного размера.



1. Бит присутствия-отсутствия страницы, ошибка отсутствия страницы

Оборудование отслеживает присутствие конкретных страниц в физической памяти за счет бита присутствия-отсутствия. Диспетчер памяти замечает, что страница не отображена, и заставляет центральный процессор передать управление операционной системе. Это системное прерывание называется ошибкой отсутствия страницы (page fault).

1. Таблица страниц, структура записи

Таблица страниц соотносит виртуальные адреса страниц и страничных блоков.

Биты защиты сообщают о том, какого рода доступ разрешен(разрешения чтения, записи и исполнения страницы).

Когда в страницу осуществляется запись, аппаратура автоматически устанавливает **бит модификации**.

**Бит ссылки** устанавливается при обращении к странице для чтения или записи.

Оставшийся **бит** позволяет **блокировать кэширование** страницы.

1. Вспомогательные свойства страниц при реализации страничной модели управления памятью

* бит присутствия-отсутствия
* бит защиты
* биты модификации
* бит ссылки
* бит,позволяющий блокировать кэширование страниц.

1. Вопросы страничной организации памяти

1) Отображение виртуального адреса на реальный должно быть быстрым

2) Если пространство виртуальных адресов слишком обширное, таблица страниц будет иметь весьма солидный размер.

1. Страничная организация памяти: достоинства/недостатки

* Достоинства:
  + нет проблемы внешней фрагментации
  + никак не ограничены размерами физической памяти
* Недостатки
  + проблема принятия решений об организации таблицы страниц
  + при страничной организации памяти адресное пространство представляет одну модель от 0 до Ν. Т.е. мы работаем с одним пространством адресации в этом процессе. В некоторых ситуациях это бывает не удобно.

1. Сегментная организация памяти: достоинства/недостатки

* Достоинства:
  + простота реализации
  + размер таблицы сегментов может быть много меньше размера таблицы страниц
* Недостатки:
  + наличие внешней фрагментации
  + сегмент рассматривается как единое целое

1. Какие недостатки двух базовых моделей устраняет сегментно-страничная модель управления оперативной памятью?

Логическое пространство процесса разбивается по смыслу на сегменты, каждый из которых разбивается на страницы одинакового размера. Страницы отображаются в фреймах физического пространства. Логический адрес представляет собой тройку чисел: номер сегмента, по которому получается его таблица страниц, номер страницы, по которой получается номер фрейма, в которой она отображена и смещение внутри страницы, с использованием которого получается адрес физической ячейки

Преимущества:

* осмысленность разбиения на сегменты сочетается с отсутствием фрагментации страничного подхода.
* Менее длительное время поиска по сравнению с сегментной организацией памяти



1. Буфер быстрого преобразования памяти(TLB)

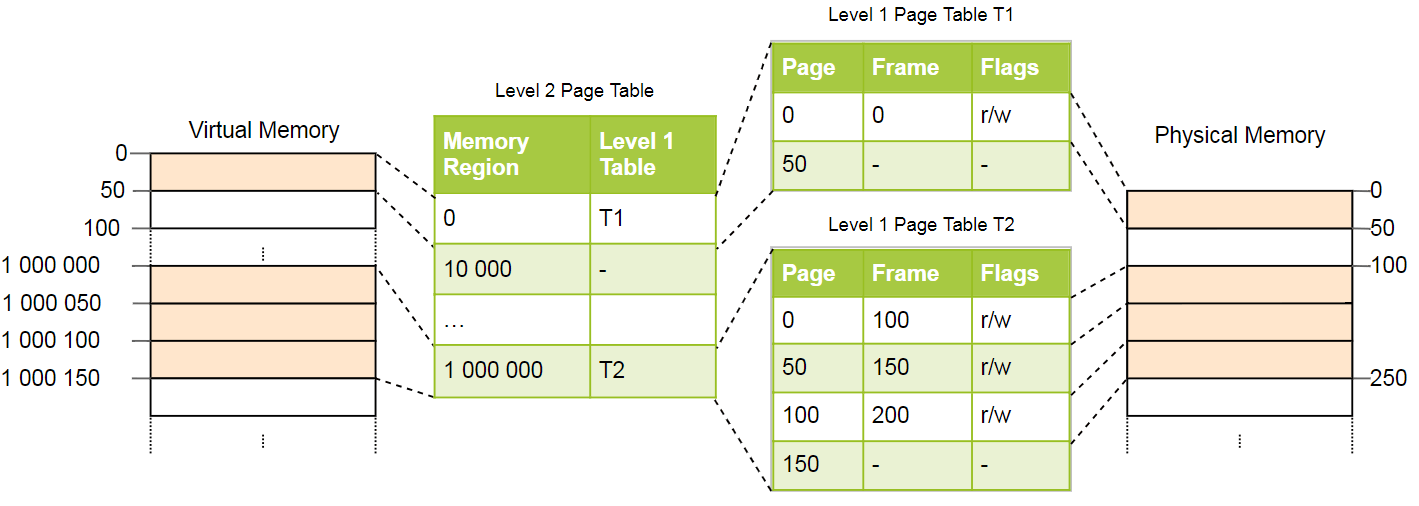
это специализированный кэш центрального процессора, используемый для ускорения трансляции адреса виртуальной памяти в адрес физической памяти.

Это устройство находится внутри диспетчера памяти и состоит из небольшого количества записей. Каждая запись содержит соответствие адреса страницы виртуальной памяти адресу физической памяти.

1. Поверхностно Программное управление буфером TLB и ошибки в ней

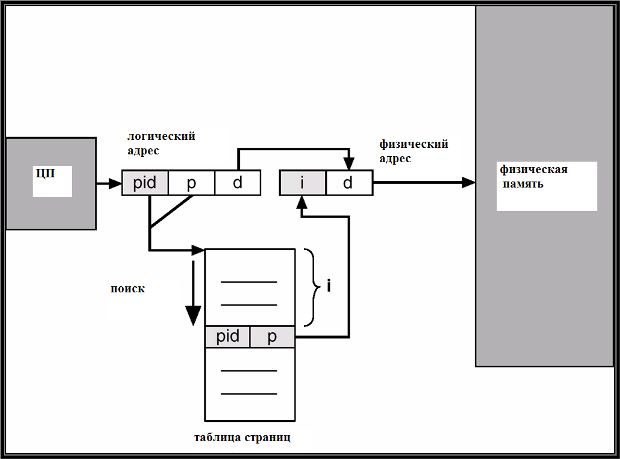
* Программная ошибка: страница, к которой идет обращение, отсутствует в TLB, но присутствует в памяти.
* Аппаратная ошибка: страница отсутствует в памяти.
* Прекращает выполнение программы с выдачей ошибки сегментации: программа просто обратилась по неверному адресу

1. Многоуровневые таблицы страниц

Суть метода заключается в создании дополнительной таблицы, которая называется таблицей страниц *второго уровня*, выполняет преобразование между областями адресов и таблицами страниц первого уровня.

1. Инвертированные таблицы страниц

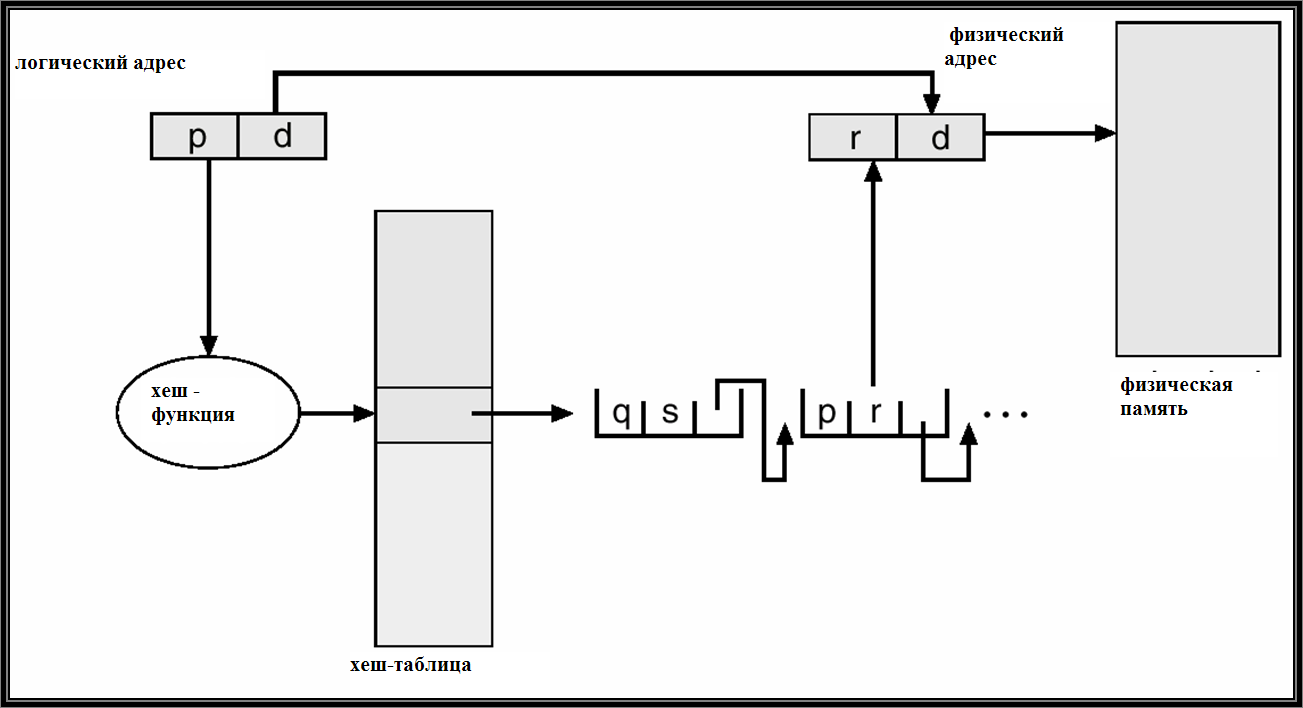
В данной конструкции имеется одна запись для каждого страничного блока в реальной памяти. В каждой записи отслеживается, что именно находится в страничном блоке (процесс, виртуальная страница). Хотя инвертированные таблицы страниц экономят значительное количество пространства, у них есть один серьезный недостаток: преобразование виртуальных адресов в физические становится намного сложнее. Решение этой дилеммы состоит в использовании TLB.



1. Хешированные таблицы страниц

базируется на понятии хеш-функции (hash function) - целочисленной функции hash, определенной на элементах s некоторого пространства поиска S и принимающей значения из отрезка [0, H-1] где H – натуральное число.

Общая идея поиска с помощью хеш-функций заключается в следующем: пространство поиска разбивается на H непересекающихся подмножеств (списков) Sh, в каждом из которых хранятся элементы, имеющие одинаковое значение хеш-функции, равное h.



1. Алгоритмы замещения страницы:

1) Оптимальный алгоритм замещения

Замещается та страница, которая не использовалась в течение наибольшего периода времени.

2) Алгоритм замещения недавно использовавшейся страницы(LRU)

Замещается та страница, которая раньше(недавно) всего использовалась.

3) FIFO

Выбирается фрейм, первым из имеющихся считанный в основную память.

4) Второй шанс

Алгоритм, в котором замещается НЕ та страница, к которой дольше всего не было обращения, а следующая за ней по списку страниц, упорядоченному в порядке возрастания времен обращений.

5) Часы

Все страничные блоки содержатся в циклическом списке в виде часов.

Стрелка указывает на самую старую страницу. При возникновении ошибки

отсутствия страницы проверяется та страница, на которую указывает стрелка.

Если ее бит R имеет значение 0, страница выселяется, на ее место в «циферблате»

вставляется новая страница и стрелка передвигается вперед на одну позицию.

Если значение бита R равно 1, то он сбрасывается и стрелка перемещается на

следующую страницу.

Этот процесс повторяется до тех пор, пока не будет найдена страница с R = 0.

6) Замещение наименее востребованной страницы. **Least Frequently Used (LFU)**

Алгоритм, при котором замещается страница с минимальным значением счетчика обращений (к которой было меньше всего обращений).

7) Рабочий набор

Моделью рабочего набора: отслеживание рабочего набора каждого процесса и обеспечение его присутствие в памяти, перед тем как позволить процессу возобновить работу.

Он был разработан для существенного сокращения количества ошибок отсутствия страниц. Загрузка страниц до того, как процессу будет позволено возобновить работу, называется также опережающей подкачкой страниц.

8) WSClock

Если бит R установлен в 1, значит, страница была использована в течение текущего такта, поэтому она не является идеальным кандидатом на удаление. Затем бит R устанавливается в 0, стрелка перемещается на следующую страницу, и алгоритм повторяется уже для нее. Если у страницы, на которую указывает стрелка, бит R = 0 и ее возраст превышает значение t, страница не изменена, она не относится к рабочему набору и ее точная копия присутствует на диске, то тогда страничный блок просто освобождается и в него помещается новая страница.

Но если страница изменена, ее блок не может быть тотчас же освобождён, поскольку на диске нет ее точной копии. Чтобы избежать переключения процесса, запись на диск планируется, а стрелка перемещается дальше и алгоритм продолжает свою работу на следующей странице.

1. Виртуальная память

* Виртуальная память может быть реализована путем страничной или сегментной организации по требованию.
* Страничная организация по требованию – схема страничной организации, при которой страница подкачивается в основную память только при обращении к ней. При отсутствии адресуемой страницы в основной памяти происходит прерывание – отказ страницы (page fault). Бит valid-invalid элемента таблицы страниц указывает, присутствует ли страница в основной памяти.
* При обработке ситуации отказа страницы ОС находит фрейм в основной памяти и подкачивает в него страницу с диска.
* При отсутствии свободного фрейма выполняется алгоритм замещения страниц – выбор фрейма-жертвы, откачка его на диск и подкачка на освободившееся место адресуемой страницы.
* При замещении страниц на диск откачиваются только модифицированные страницы (для их указания используется бит модификации в элементе таблицы страниц).

1. Организация памяти

* Страничная организация – метод управления памятью, при котором логическая и физическая память делится на страницы одинаковой длины.
* Физические страницы (фреймы), выделяемые для логических страниц процесса, могут располагаться произвольным образом.
* Для трансляции логических адресов в физические используется таблица страниц процесса.
* Логический адрес состоит из номера страницы и смещения внутри страницы.
* ОС хранит список свободных фреймов в основной памяти.
* Для реализации таблицы страниц в системе имеется регистр таблицы страниц, содержащий ее адрес и длину.
* Для ускорения трансляции адресов используется TLB.
* Для защиты памяти процесса используется бит valid – invalid в каждом элементе таблицы страниц, указывающий, принадлежит ли страница логической памяти процесса.
* Таблицы страниц бывают различной структуры – иерархические, хешированные и инвертированные.

1. Размер страницы(внутренняя фрагментация и прочее)

Размер страницы является тем параметром, который должен быть выбран операционной системой. В среднем половина последней страницы останется незаполненной. Оставшееся пространство на этой странице тратится впустую. Эти потери называются внутренней фрагментацией.

1. Совместно использующиеся страницы/библиотеки

Когда два и более процесса совместно используют один и тот же код, возникает проблема совместно используемых страниц. Любые функции, вызываемые в объектных файлах, но не присутствующие в них (например, printf), называются неопределенными внешними функциями и выискиваются в библиотеках. Если они найдены, то их включают в исполняемый двоичный файл. Любые вызываемые, но не присутствующие в них функции также становятся неопределенными внешними функциями.

1. Отображаемые файлы

Идея состоит в том, что процесс может выдать системный вызов для отображения файла на какую-то часть его виртуального адресного пространства.

1. Политика очистки страниц

Замещение страниц лучше всего работает при наличии достаточного количества свободных страничных блоков, которые могут потребоваться при возникновении ошибки отсутствия страницы.

Если заполнен и, более того, изменен каждый страничный блок, то перед помещением в него новой страницы сначала должна быть записана на диск старая страница. Для обеспечения поставки свободных страничных блоков системы замещения страниц, как правило, имеют фоновый процесс, называемый *страничным демоном*, который периодически пробуждается для проверки состояния памяти. Если свободно слишком мало страничных блоков, страничный демон начинает подбирать страницы для выгрузки, используя какой-нибудь алгоритм замещения страниц. Если эти страницы со времени своей загрузки подверглись изменению, они записываются на диск.

1. Обработка ошибки отсутствия страницы

Если в таблице страниц имеется ссылка на страницу, отсутствующую в памяти, первое же обращение по такой ссылке приведет к прерыванию и вызову ОС (page fault)

ОС по таблицам определяет, что именно произошло:

* Если имеет место неверная ссылка (на страницу, отсутствующую в логической памяти), то работа программы прекращается.
* Если же имеет место обычное отсутствие страницы в памяти, то ОС должна разместить его в основной памяти. Для этого ОС выполняет следующий алгоритм:
* **Найти незанятый фрейм в основной памяти** ;
* **Считать содержимое страницы в данный фрейм** ;
* **Изменить элемент таблицы страниц**: validation-бит установить равным 1;
* **Продолжить работу программы**.

1. Блокировка страниц памяти

Эта политика определяет, какие учетные записи вправе использовать процесс для хранения данных в физической памяти, что предотвращает страничную запись данных операционной системой в область виртуальной памяти на диск.

1. Резервное хранилище

Это дисковое устройство для хранения информации, доступное по локальной сети или через Интернет.

1. Memory management

* buddy allocator

это алгоритм выделения памяти использует разделение памяти на половины.

Когда больший блок разделяется, он разделяется на два меньших блока, и каждый меньший блок становится уникальным другому. Разделенный блок может быть объединен только с его уникальным блоком друзей, который затем реформирует больший блок, от которого они были отделены.

* slab allocator

Основой этого алгоритма является сохранение выделенной памяти, содержащей объект определённого типа, и повторное использование этой памяти при следующем выделении для объекта того же типа.

Распределитель slab хранит информацию о размещении участков определённого типа и размера, известных также как кэши. Уничтожение объектов не освобождает память, а только открывает слот, который помещается в список свободных слотов распределителем slab.

Slab представляет собой непрерывный участок памяти, обычно составленный из нескольких физических смежных страниц. Кэш состоит из одного или более slab’ов.

1. Процедура загрузки ОС на примере ОС Android

Bootloader — это загрузчик — приложение, которое самым первым запускается на вашем устройстве, считывает свою конфигурации, и на основании её, загружает операционную систему (Android).​

Zygote — ключевой компонент любой Android-системы, который ответственен за инициализацию, старт системных служб, запуск и остановку пользовательских приложений и многие другие задачи.

## Нужные определения

* **Свопинг** — это процесс выгрузки редко используемых областей виртуального адресного пространства программы и/или всей программы на диск или другое устройство внешней памяти.
* **Виртуальная память** – метод распределения памяти, при котором логическая память процесса отделена от физической, превышает физическую по объему, ее образ хранится во вторичной памяти, а конкретный фрагмент памяти подкачивается в основную память при обращении к нему. Такой метод позволяет расширить адресное пространство, обеспечить совместное использование памяти, сэкономить память при создании процессов.
* **Машинное слово** - единица данных, которая выбрана естественной для данной архитектуры процессора.
* **Отказ страницы** ( **page fault) –** прерывание по отсутствию страницы в основной памяти.
* **Бит модификации** - бит элемента таблицы страниц, указывающий на модификацию страницы, с целью откачки на диск только модифицированных страниц.
* **Бит ссылки (reference bit)** – бит элемента таблицы страниц, равный 0, если к странице не было обращения, и 1, если обращение к странице было.
* **Бит "valid/invalid"** – бит элемента таблицы страниц, указывающий на присутствие или отсутствие страницы в основной памяти.
* **Замещение страницы** (page replacement) – подкачка операционной системой страницы, к которой произошло обращение, вместо другой страницы, с откачкой последней, если она требуется.
* **Отказ страницы** ( page fault) – прерывание по отсутствию страницы в основной памяти.
* **Страничная организация по требованию** (paging on demand) – метод организации виртуальной памяти, основанный на страничной организации, при котором каждая страница загружается в память, только если она реально требуется при выполнении программы – содержит код или данные, к которым произошло обращение.
* **Thrashing** – ситуация критической нехватки основной памяти в системе, при которой процессор занят в основном откачкой и подкачкой страниц.
* **Фрейм** – страница физической памяти; образ страницы в основной памяти.
* **Откачка и подкачка** – механизм сброса образов неактивных процессов на диск и, при их активизации, считывания обратно в основную память. Файл откачки – пространство на диске, где система хранит образы откачанных процессов.

# Файловая система

1. С помощью какого механизма реализуется произвольная графовая структура файловой системы?

При помощи жёсткой ссылки (**Жёсткая** **ссылка** может создаваться только в пределах одного логического раздела и только для **файлов**.).

Подобная привязка увеличивает показания файлового счётчика i-узла. В итоге один и тот же файл может появиться в нескольких каталогах, возможно, под разными именами. При помощи символической ссылки. Вместо двух имён, указывающих на одну и ту же внутреннюю структуру данных, представляющую файл, может быть создано имя, указывающее на очень маленький файл, в котором содержится имя другого файла.

Может возникнуть проблема, если владелец файла удалит его (и i-узел тоже), то указатель, каталога содержащего ссылку, будет указывать на не существующий i-узел. Потом может появиться i-узел с тем же номером, а значит, ссылка будет указывать на не существующий файл.

Поэтому в этом случае при удалении файла i-узел лучше не удалять.

Файл будет удален только после того, как файловый счетчик будет равен 0.

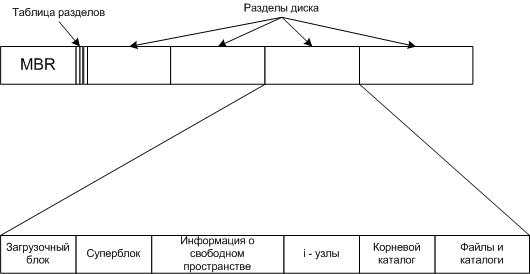
1. Какие ограничения формата разделов MBR устраняет формат GPT?

* Поддержка большого числа разделов (до 128 разделов для Windows).
* Возможность создавать разделы объемом более 2 ТБ.
* Дублирование заголовков и таблицы разделов GPT в конце диска и использование контрольных сумм для проверки корректности содержимого.

1. Функции файловых систем

* Логическая организация данных
* Отображение логической модели файловой системы на физическую организацию хранилища данных
* Разграничение доступа к данным
* Обеспечение целостности данных
* Создание однородной среды для разных носителей информации
* Обеспечение высокой производительности

1. Структура файловой системы



Суперблок - содержит ключевые параметры файловой системы.

**Реализация файлов:**

* Непрерывные файлы(Выделяется каждому файлу последовательность соседних блоков.)
* Связные списки(Файлы хранятся в разных не последовательных блоках, и с помощью связных списков можно собрать последовательно файл.)
* Связные списки при помощи таблиц в памяти(Вся информация о блоках хранится в специальной таблице загружаемой в память.)

**i-узлы**

С каждым файлом связывается структура данных, называемая i-узлом (index-node- индекс узел), содержащие атрибуты файла и адреса всех блоков файла.

**Каталоги**

При открытии файла используется имя пути, чтобы найти запись в каталоге. Запись в каталоге указывает на адреса блоков диска.

В зависимости от системы это может быть:

* дисковый адрес всего файла (для непрерывных файлов)
* номер первого блока (связные списки)
* номер i-узла

**Использование хэш-таблицы для ускорения поиска файла.**

Алгоритм записи файла:

* Создается хэш-таблица в начале каталога, с размером n (n записей).
* Для каждого имени файла применяется хэш-функция, такая, чтобы при хэшировании получалось число от 0 до n-1.
* Исследуется элемент таблицы соответствующий хэш-коду.
* Если элемент не используется, туда помещается указатель на описатель файла (описатели размещены вслед за хэш-таблицей).
* Если используется, то создается связный список, объединяющие все описатели файлов с одинаковым хэш-кодом.

Алгоритм поиска файла:

* Имя файла хэшируется
* По хэш-коду определяется элемент таблицы
* Затем проверяются все описатели файла из связного списка и сравниваются с искомым именем файла
* Если имени файла в связном списке нет, это значит, что файла нет в каталоге.

Атрибуты файлов

● тип файла (обычный файл, каталог, специальный файл и т. п.);

● владелец файла;

● создатель файла;

● пароль для доступа к файлу;

● информация о разрешенных операциях доступа к файлу;

● времена создания, последнего доступа и последнего изменения;

● текущий размер файла;

● максимальный размер файла;

● признак «только для чтения»;

● признак «скрытый файл»;

● признак «системный файл»;

● признак «архивный файл»;

● признак «двоичный/символьный»;

● признак «временный» (удалить после завершения процесса);

● признак блокировки;

● длина записи в файле;

Нужные определения

**Каталог** — специальный файл, содержащий структуру, позволяющую описывать вложенные в него файлы (имена, физическое месторасположение, атрибуты и т. д.)

**Файловая система** (ФС) - это часть операционной системы, назначение которой состоит в том, чтобы организовать эффективную работу с данными, хранящимися во внешней памяти, и обеспечить пользователю удобный интерфейс при работе с такими данными.

**Файл** — это именованная область внешней памяти, в которую можно записывать и из которой можно считывать данные.

**SELinux** реализация системы принудительного контроля доступа, которая может работать параллельно с классической избирательной системой контроля доступа.

Другие вопросы

1. Контейнеры Docker

**Контейнеризация** — это «облегченная» форма виртуализации на уровне операционной системы, которая позволяет запускать приложение и системные библиотеки в изолированной области, «контейнере», каждый из которых содержит все компоненты, необходимые для запуска приложения, не зависит от архитектуры серверной системы и взаимодействует с ОС при помощи стандартных интерфейсов.

Docker — это проект с открытым исходным кодом для автоматизации развертывания приложений в виде переносимых автономных контейнеров, выполняемых в облаке или локальной среде.

1. Стандартизация POSIX

набор стандартов, описывающих интерфейсы между операционной системой и прикладной программой (системный API), библиотеку языка C и набор приложений и их интерфейсов. Стандарт создан для обеспечения совместимости различных UNIX-подобных операционных систем и переносимости прикладных программ на уровне исходного кода.

1. Виртуализация VirtualBox

предоставление набора вычислительных ресурсов или их логического объединения, абстрагированное от аппаратной реализации, и обеспечивающее при этом логическую изоляцию друг от друга вычислительных процессов, выполняемых на одном физическом ресурсе.

1. Гипервизор

программа или аппаратная схема, обеспечивающая одновременное, параллельное выполнение нескольких операционных систем на одном и том же хост-компьютере. Гипервизор также обеспечивает изоляцию операционных систем друг от друга, защиту и безопасность, разделение ресурсов между различными запущенными ОС и управление ресурсами.

1. Журналируемые файловые системы

файловая система, в которой осуществляется ведение журнала, хранящего список изменений и помогающего сохранить целостность файловой системы при сбоях.

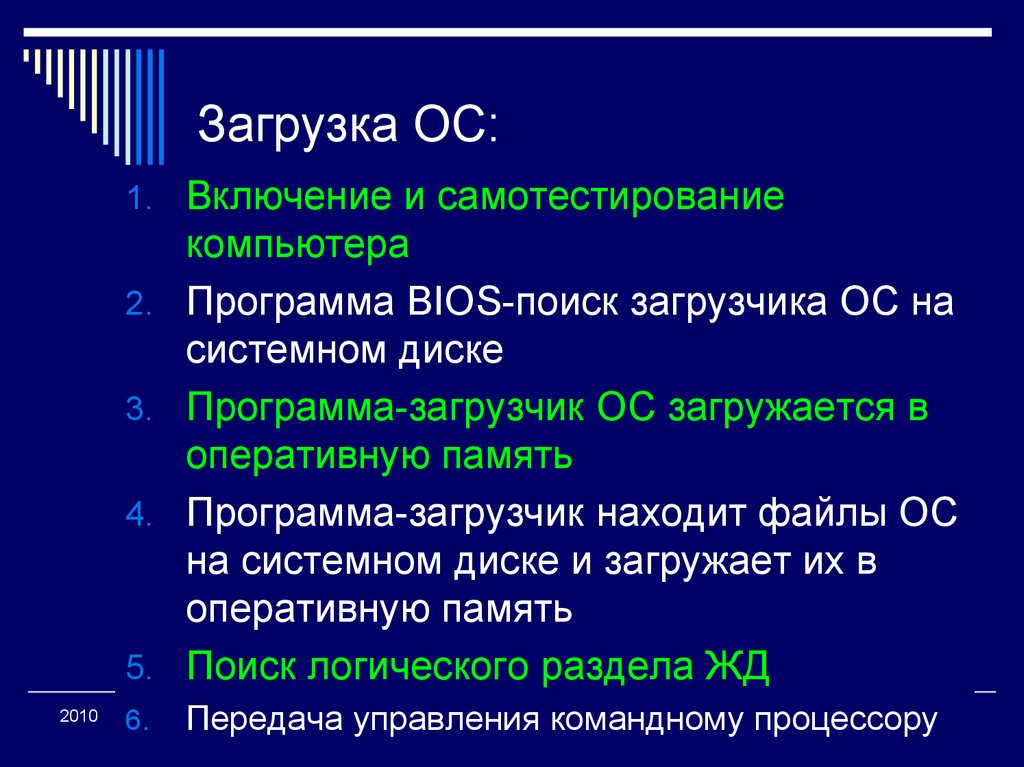
Журналируемая файловая система сохраняет список изменений, которые она будет проводить с файловой системой, перед фактическим их осуществлением.

1. Процедура загрузки ОС

Включение и самотестирование компьютера

Программа BIOS - поиск загрузчика ОС на системном диске

Программа-загрузчик



1. Fiber

Как и thread, fiber разделяют адресное пространство. Однако fiber используют кооперативную многозадачность, в то время как потоки используют вытесняющую многозадачность.

волокна являются конструкцией системного уровня, рассматриваемой как потоки, которые не работают параллельно.

+

конструкции синхронизации не нужны при написании fiber кода, поскольку они неявно синхронизированы.

-

волокна не могут использовать многопроцессорные машины без использования упреждающих потоков

1. Thread affinity

технология, которая обеспечивает закрепление и открепление процесса или потока к конкретному ядру центрального процессора, центральному процессору или набору процессоров, так что процесс или поток будут выполняться только на указанном ядре.

1. Необходимые условия возникновения взаимоблокировки.

* Mutual Exclusion (Взаимное исключение) – по крайней мере один из запрашиваемых ресурсов является неделимым (то есть должен захватываться в эксклюзивное использование).
* Hold and wait (Удержание ресурсов при ожидании) – существует процесс, владеющий некоторым ресурсом и ожидающий освобождения другого ресурса.
* No preemption (Неперераспределяемость ресурсов) – ресурсы не могут быть отобраны у процесса без его желания.
* Circular wait (Циклическое ожидание)

## Нужные определения

Главная загрузочная запись (MBR) — код и данные, необходимые для последующей загрузки операционной системы и расположенные в первых физических секторах на жёстком диске или другом устройстве хранения информации.

MBR содержит небольшой фрагмент исполняемого кода, таблицу разделов диска и специальную сигнатуру.

В процессе запуска компьютера после окончания начального теста базовая система ввода-вывода (BIOS) загружает «код MBR» в оперативную память и передаёт управление находящемуся в MBR загрузочному коду.