

Онит:

- метод &-программ.
- метод схемат.
- алгоритм работы &-программ
- алгоритм программ SUBROUTINE
пред собствен. комп. - зачем?

ТУТУН - синоним
-мин. науки и соф

ЛЕКУНИЯ 1

История операционных систем
аппаратное обеспечение №1

Компьютер - программируемое устройство

Чарльз Бабингтон

1833 создана разностная машина
под аналитической машиной

Ада Леблэнд (жена Бабингтона) - первая програмлист

I Поколение ЭВМ

- не имеют элемента памяти
Было Реле

(Эннек) ENIAC - на электронных лампах

1944 год

44-45 - годы начала I-поколения

изменение памяти. → перекомпьютеризаци
компьютер. машины

1945 Дональд Кнутон

UNIVAC EDVAC

1946 - получают предвар. основные принципы работы
компьютер. предвар. расчет.

1 Универсальное вычисл. машина дополнена схемами

- арифм.
- памяти
- управления
- связи с оператором

внешн. ввода/вывода

2 Запоминающее не только числ. инф. пред. для машин. а так же
команды управления.

3 Если команды/машине представляются в виде числ. кода
и можно один. число со командой => помехи исчезают
или же (команды и управл.) машинам хранятся в проца
изображимости

4 Помимо непр. фнр. команд - есть устройство способное временно хранить команды, которые хранятся в памяти управляющей

5 Память - вспомогательный
→ архитект. орган

6 Помимо существ. учр. памяти - какого в когдай есть у. пам. память - чипов

Память: физич. massa
электроника
вспомог. компоненты для других

⇒ Архитектура органов Памяти

Доступ к командам и данным только по адресу

⇒ программа в памяти в несп. адресах
если требуется выполнение команды ⇒ можно выполнить

Повторение адресов и фрагментов

Документация → еще серийных

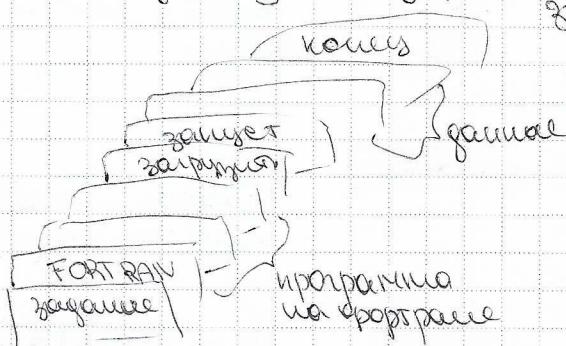
II Появление ЭВМ

IBM 1401 IBM 1410



появление многозадачности ⇒ в памяти одновременно находятся много программ

Управляющий адрес → управл. программами
загрузки



max врем/намет

наиболее быстрое уст.

наиболее медленное микропрограммы

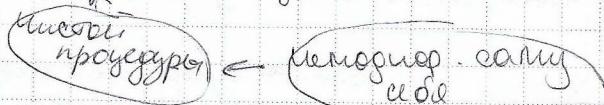
Лекция 3

Основные задачи ОС

- управление процессами и безопасность и ресурсов (модуль конфиденциальности)

• процессорное время

• смены памяти (реализация новых санитарных систем)



В ОС используются санитарные списки

↑ управление и т.д.
использование
указаний

использование
миссии
регистрации
регистрации
в ОС
→ в задачах

Дисциплины:

- управление процессорами
- управление памятью (оперативная память)
- вспомогательные характеристики процессов
- управление устройствами (II сенсор)
- классиф. ядер ОС

Управление процессорами

(о процессах и особенностях
управл. процессов в ЭВМ)
рассматриваются

- процессорное время

Основные характеристики ОС

- процесс

Реализация ядер ОС

некоторое время — режим ядра

некоторое время — режим пользовательского

→ начало переключения

В ОС присутствует бордюров — Диаграмма состояний процесса
(этапы жизни процесса)

Обобщенная (DC) — состояние процесса в некр. ОС

→ процессор разрабатчиков

регистрировано — начало настройки ядра ядра

регистрировано — начало настройки ядра ядра

регистрировано — начало настройки ядра ядра

запуск — начало запуска

запуск — начало запуска

запуск — начало запуска

запуск

Для ядерных систем

будет находиться один

процесс

В многоядерных системах

несколько раздельных времени

процессы организуются в очередь

(систем с основным кон-ком и прр.)

(или ввод/вывод) → запуск

из временного → блокировки (ресурс этого процесса)

режим ядра

запуск

режим ядра

Три типа событий передач. синхрон и асинхрон ядра

↳ синхронные вызовы

- программные прерывания traps
- это интервальные некоторые прерыв. напоминания. ОС API
- набор функций опред. в системе
- для того чтобы система обрабатывала его каким сервисом?

Ввод/вывод

ОС устроена как набор минимум. кон-бо есть вызовов

Многие ОС не имеют

программные обработки

на прямую к устрой.

→ использование ядра

В UNIX все одинак

- со всеми устройствами
- система реагирует единодушно

- синхронное событие происходит в процессе выполнения программы
- но эти к выполнению программы.

↳ асинхронные ситуации

- исправимые → синхронные прерывания (откл. блокировки страницы)
- неисправимые → ошибки (символическое значение)

• синхронное событие по ошибке.

↳ аппаратные прерывания

- прерывание от внешнего устройства

(сейчас прерывание на него возможны выполнение других функций

исключения из выполнения)

- прерывание от внешних устройств

(возникает по завершении опер. ввода/вывода)

⇒ просто программа что ввод/вывод выполнил завершил

⇒ обработка данных

получение от устройства или окончанием

кто же программа получила данные?

- от системного оператора

Windows

Автоматическое асинхронное событие в системе

все зависимости работы выполнены программой

→ Иницирование и запуск программы

- выделение ресурса?

настройка программы в ядро по выборочным функциям

иницировщик - передает ядро данные указ. работы

Диспетчир. - не ядр. выделение како-то либо ресурса

- выделение и передача программы. браниши

Оперед. → время ожидания в конце

Цикл. меж. планирования \rightarrow взаимодействие с планами ОС (многозадачность)
↳ накопленные накладки
↳ есть разн. времена

некоторые накладки
 \rightarrow некоторые задачи не могут быть выполнены
 \Rightarrow неиспользование

использование задачи

классификатор. все алгоритмы планирования

- \rightarrow без переключения и с переключением
- \rightarrow с приоритетами и без приоритетов
- \rightarrow без блокировки и с блокированием (без приор. и с блокир.)

Алгоритмы планирования в цикл. накопленной обработки

1. SJF FCFS



без переключения

без блокировок

некоторые задачи \rightarrow выполнение по количеству

2 SJF Shortest Job First

без блок. в очередь по заданному времени блокировка выполнения производительность машин

использование времени

(использование остатков времени)

без блокировок

без переключения

с приоритетами

3 SRT Shortest Remaining Time
наименее оставшееся время

с приоритетами

с выполнением

выполн. процесс может быть прерван
если будет поступить процесс с меньшим
затратами выполнения

4 HRRN Highest Response Ratio
(использование остат. времени)

использование непрерывности: $R = (t_w + t_s) / t_s$

некоторые задачи в очереди блокированы

t_w - время ожидания в очереди готовых процессов

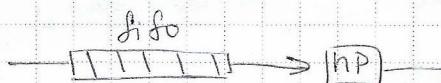
\Rightarrow не всегда используется
использование остатков времени

\rightarrow неисп. от времени процессов
в очереди готов. проц.

В системах разделения времени
(точно время - время отвода системой)
процессорное время становится квантобель

Алгоритм

1. RR Round Robin



Большинство used в син. квант

без проприетар
без вспомогатель
и проприетар

недостаток:

- систем. квант времени
- или большое кол-во вариантов

В собр. сист. нововведение

→ назначение врем. квантов
напрям. квантов времени динамически

⇒ гор. начальные расходы

2. Min - Max RR (MMRR)

2. Average Max RR (AMRR)

3. Efficient Dynamic RR (EDRR)

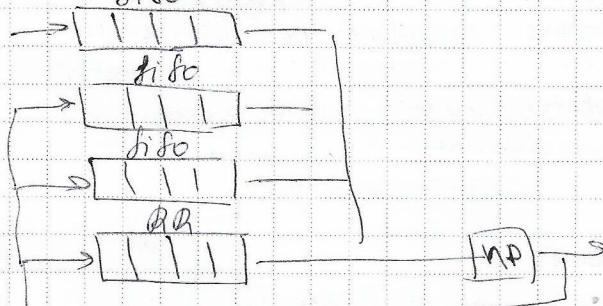
квантование
→ formal slicing

windows → много очередей с разными проприетар
член. например.

2. Приоритетное планирование

аддитивное планирование

lifo



алгоритм effort или.

- неявное проприетар
- получи заявку что соз. или заверш. операции
- выбирается квант → наибольшее кол-во процессов
использовано вспом. запас блог-баки
или пакеты

если не успел → неисп. пропр. очередь
ночь не оканчивается в последней
ночью. RR

⇒ скрытые потери времени которым требуется много
прерыв. времени → вспомогательные процессы
из-за конфликтов процессов / системы не может ничего не

Алгоритм нации.

Диаграмма состояний процесса (см. ранее)

Последовательность процессов в очереди \rightarrow написаны в цикле

UNIX - часть в режиме пользователей (самостоятельно)

- часть в режиме ядра (

Ядро процесс в режиме ядра \rightarrow переходы аппаратного комплекса
сохранение регистров
(push/pop)

Переходы аппаратного комплекса

(аппарат. комплекс + часть ядро процессор. врем.)

указанные на скобках ядра

есть смысл волг. проц. ресурсов)

\Rightarrow затратное действие

Изменение состояния затраты

Действия:

6 типах
безн. ядра.
безн.

уровни
существование
для каждого



созд
зап
запущ
запущен
запущен ядром
запущен ядром

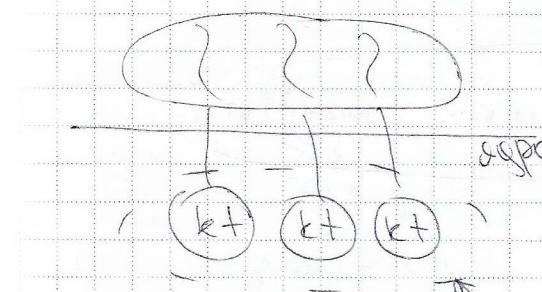
{ - часть ядра программы
которая имеет конечн. управ.
с групп. частями ядра
непрерывно

Классификация:

наздр. в сопр. сис.
3 потоки уровня - потоки
наздр. с помощью библиотек
уровни ядеров. (наздр. ядра нет)

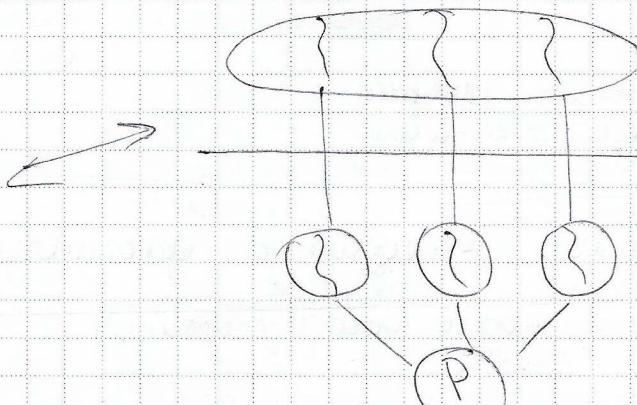
анн. запуск ядер/блок
 \rightarrow блок процесс
 \rightarrow вся программа
история о них ничего не знает

3 потоки уровня ядра
наздр. ОС (т.е. ОС непрот.
протекает нет. ведет свое развитие
с потоками \Rightarrow ОС упр. потоками



потоки
безн. ядра.
процесс

созданы все избыточно
избыточные потоки
стремятся к потокам



программа непрерывно из
потоки \Rightarrow процессоры
безн. ядра. потоки

поток берет определенное количество
(аппарат. комплекс)

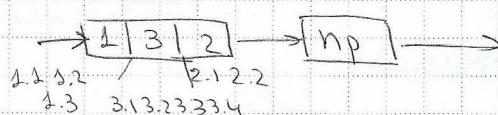
Ниток не имеет своего адресного пространства
→ выполняется в адресном пространстве процесса

Выполнение запрещается, т.к. например, не ресурса процесса

но

Система не может выделить ниткам

занятых всеми ресурсами



не выделяется

при начале ниток разных процессов

→ нестандартный контекст

представляет собой

— однопоточная и многопоточная модели процесса

→ имеет право на синхронизацию с нитями, ниткам которых нет
смысла в программе нет
ниток → один явный ниток

(если единственный управляемый объект системы)

код	данные	рамки
регистры	стек	
ниток		

код	данные	рамки
регистры	стек	регистры
стек	стек	стек

каждый нитка владеет своим аппарат. контекстом

• в сущ. смыс. структура определяет процесс (один управляемый объект управляемый управляемый процесс)

• структура отменяет ниток (один управляемый объект управляемый, который не определяет структуру процесса)

- один явный ниток
- несколько единиц управления

Процесс — единица декомпозиции системы

процесс управляет ресурсами

Нитки могут разделять одни и те же данные

Рамка — временная единица измерения

Взаимодействие ниток — процесс

Рамка управляет нитками

Мониторинг:

погоды

Пример правильного мониторинга.

- много проверят - потому что может проверить граммат.
- 6 это время другой поток
- образ. без разрывов (использование клавиши)
- третий
- загружен изобр. с помощью диска
- шестой
- разрешенное копирование

1 потоки непрерывны если процесс имеет несколько задач которые можно выполнить избавившись от груз. группы

2 когда одна задача может блокир. в то время другие задачи могут выполниться (текущий результат)

Дефекты:

1 Отставание в работе

- один из потоков может отстать. быстрый отстает на действие потоков когда другой блокирует или занимает больше времени.

2 Разделение ресурсов

коэф. времени и гр. ресурсы процесса можно использовать. напр. в одном агрессивном прерывании.

3 Экономичность

переключение контекста при управл. потоками быстрее чем процессов (коэффициент)

4 Многобирательность

одно потоковое процессоры

→ только на одном процессоре можно выполнить при этом в много процессорных есть

→ на разных процессорах (временные)

Проблемы:

1. Две разработчиков

принципий

5 областей в которых много процесс. требует решения проблем

1. Правильное распределение.

6 отдельных поток

задачи которые

могут блокироваться

2. Синхронизация

поток задачи которые

имеют один. или

ближую по времени

группу задач

не надо блокировать потоки

на приватное время

3. Разделение данных

неправильное → ошибки или потери данных

при взаим. пар потоков

4. Зависимость данных

одна задача зависит от результата другой задачи

⇒ выполнение синхрон.

чтобы вернуть к данным в прав. порядке

5. Тестирование и отладка

спомощ. из-за ложек false condition

однорут. ошибок при разных

потоках — спомощ. задача

Типы параллелизма

Теоретические для разных языков

параллелизм

данных

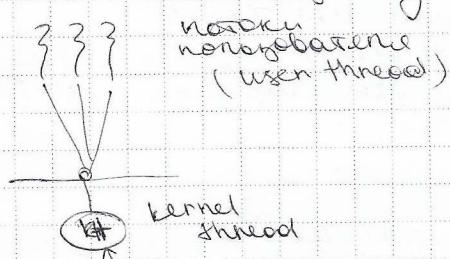
- разделение данных между всеми данными
- и совместное с ними
- и нет все загад
- же разных процессоров

Изображение

- параллелизм
- один обра

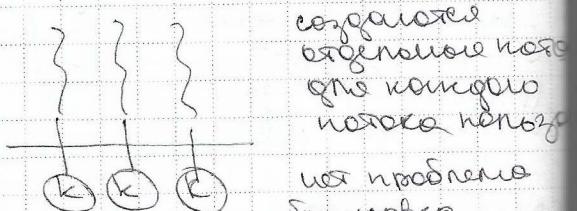
Модели:

1. ядро - ядро



много потоков ядро.
один на один поток ядра

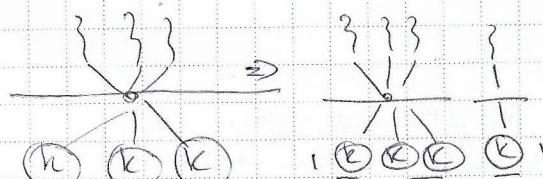
2. один - один (ядро)



нет проблема
блокировка
→ блок все

дополнительно ядро, это
модель ядра. кон. во ядро
ядро ядро есть собран
(Windows 8.1 - 80 XP)

2. ядро - ядро



ядро ядро инициализации
много кон. во потоков
потоков не равные
или меньше потоков

ядро

ядро ядро инициализации
но кон. во не имеет

блок одно

≠ блок все

процессор ядро, ядро
ядро ядро ядро ядро

может быть кон. потоков
ядро и ядро

параллелизм

загад

- выделение задачи
которые могут

единоместно
параллелизма
(текстовый редактор)

таки насыщено потоками:

Библиотечные потоки

предоставляют пространство API для создания и управления потоками.

пространство потоков
(базирующееся на API операционной системы). Каждый поток имеет собственный пространство потоков. Без поддержки ядра.

пространство ядра
системные библиотеки

Три основные библиотеки

I POSIX Pthreads

модель ядра на ядре
как и пространство потоков, и ядро
как расширение ядра

II Win32 Threads

планирование как
библиотека ядра.
ядро управляет ядром
в OC Windows

III Unix Threads

ядро имеет
виртуальную машину
планирования потоков
модель ядра включает
в ядро / kernel
заблокирует ядро машину

Задачи в ядре

использование

POSIX standard (IEEE 1003.1) ThreadHandle

→ interface для управления
потоками
и планированием

Без использования ядра

→ потоки
(доступ к операционной)

появилось Solaris / Linux
MacOS X

True64

чрезвычайно
сложные
решения

и Windows

использование библиотек
pthread_create()

на apple — GCD Grand Central Dispatch

чрезвычайно сложные потоки

light weight process (LWP)

дynamически
изменяющиеся
напоминания о потоках

{ ← user thread

 |
 |
 LWP

k

Блоки поддержки, верхний
процесс как интерфейс
между исполнением команд
и ядра

Many - to - Many
To fine

Зачем?

→ сократить время
на создание потока
и ядра
(нашер \leftarrow фиксир.)

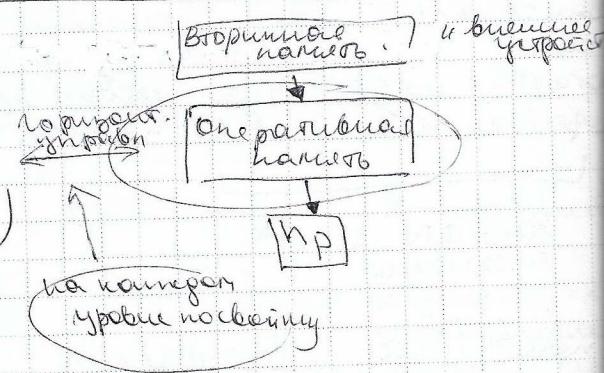
Виртуальный процессор
(VWP)
отношение между потоками
команд и (LWP) (1 - 1)

применение может оправдано
потоки упр. исполн. команд, на
одинаковой LWP

Управление памятью

в вычислительных системах
- иерархия памяти
(но отношение к процессору)
ближайшее память к процессору
ближе всего ОН

здесь тако чтобы обмениваться
данными с процессором
(другие данные и что хотим
сделать.)



Базовый уровень ОН
отстает от базового
процессора

как в процессоре
→ чтобы выровнять время

8 непрерывные
предиктивные команды Lock
перед командой and
здесь разрыв обрачение к ОН
lock генерирует команду не делит
команду не делит

Пока она не завершится
никто не может к ней
написать
последующие
блоки. Используя память

Распределение памяти

Свободное

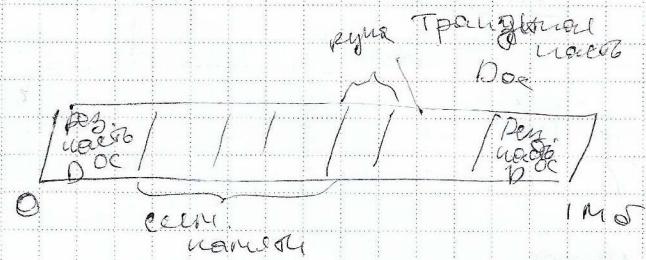
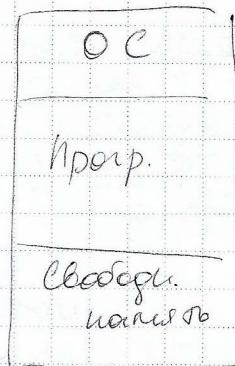
программа занимает
в памяти непрерывное
адресное пространство
(один блок)

один процессор
OC обрабатывает обработку
в каждый момент времени
в ОН находится одна программа

Несвободное

программа может быть
разбита на участки
которые в памяти зани-
мают разное время

Однако неизбежное разногласие



6 names in the report. → unknown

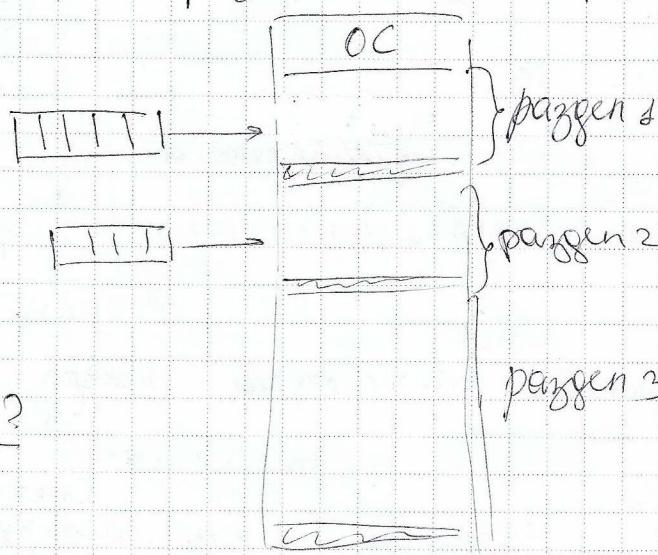
००

Мультизадачность

- 6 наименований
некоторых программ.

Reagenzien - notwendig für Analysen

Возненчие наимен
предметами опеки. разделя

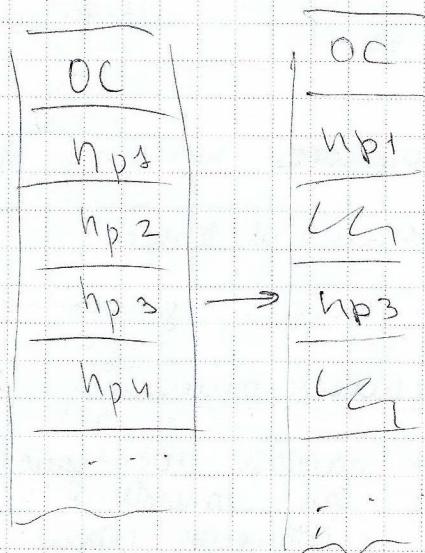


Динамическое определение
различия
(различия различий опред. в процессе
расхода)

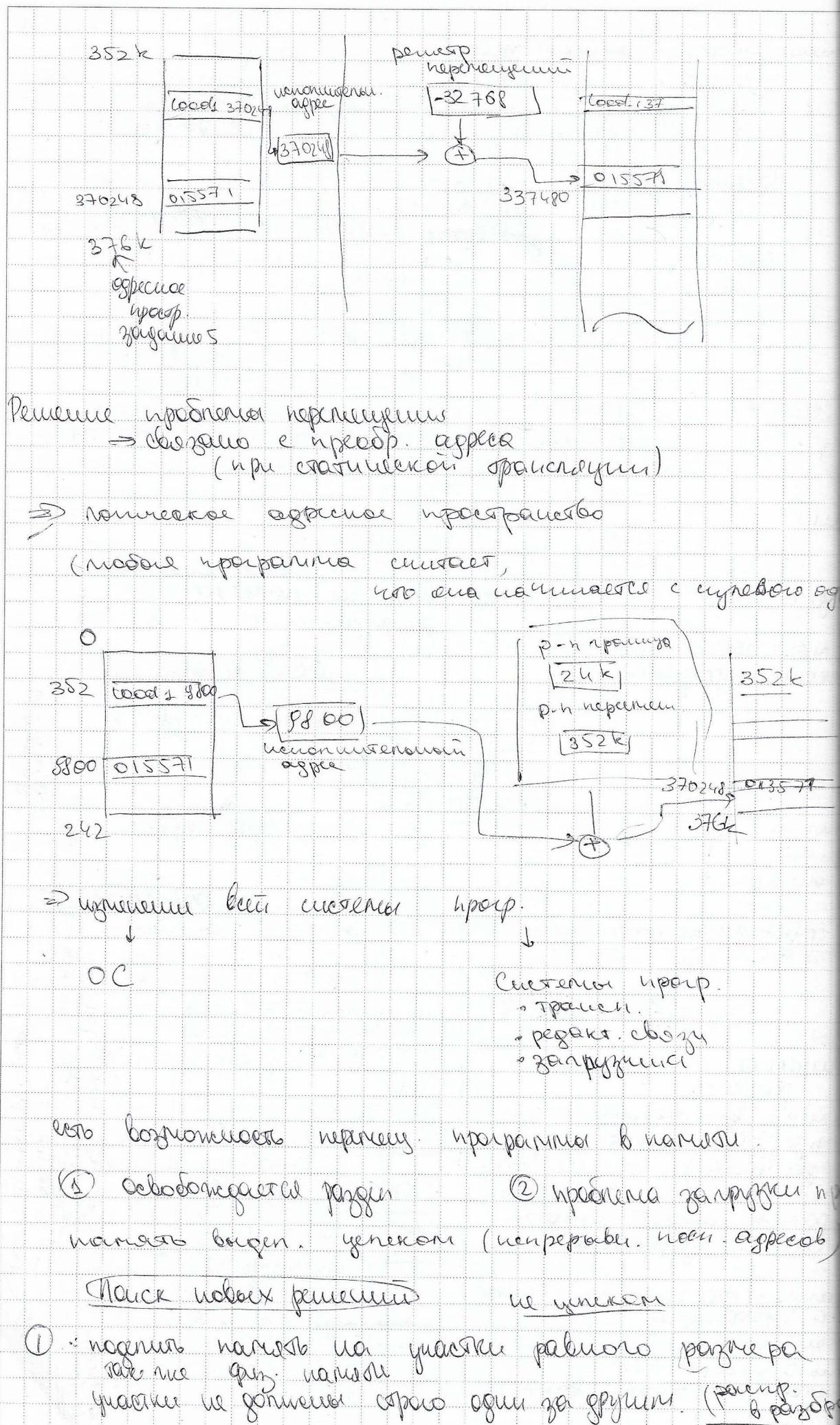
Идея проекта имеет
свойство закрываться

Задача решена при загрузке программы.

- непрерывногодвижущийся
 - самовнешний
 - самовнешний



Opisovací a klasifikační charakteristiky → historického vývoje



запроса?

- необходимо есть страница
стр. можно в рамке ^{или frame}
в которых указывал ^{название}
какой запр. стр. треб.
page

- преобразование
недох. память нац. адр.
стр. стр.

② Виртуальная

(нр. генер. программы) \Rightarrow таблица стр.
каждой задачи есть стр. сим.
указывает, в сим. адр. находит
(где сим. адрес нац. + размер)

Чаго ли что бы программа училиком запр. в память?

В памяти должны находиться те участки
стр. кода к которым обращ. процессор

запр. по необходимости (образации)

Виртуальная память

возможность выполн. пргр. которая находится не лежит
в памяти

должна наход. по сим работе

Можно обрат. процессор образуя к данным не запр. блокам

Идея: таблица

таблица

① управление памятью
страницами по запр.

② управл. пам.
сим по запр.

③ управл. пам.
символ. памятью
стр. по запросу

В момент времени: процессор
имеет запр. или сим

\Rightarrow

недох. памятью
запр.

Он должен:

напечатать возможной
наметить адрес которой требуется. должен пргр. - адресу. и представить.

Необходимо иметь таблицу:

стр. и пргр. \Rightarrow стр.
кода

страницы

① Вр. по запросу

Трёх. основы:

1) прямое отображение

2) неприм.

один конфликт прям. base address

один на конфликт
примес

системные таблицы

если прям. запр. в друг. памяти

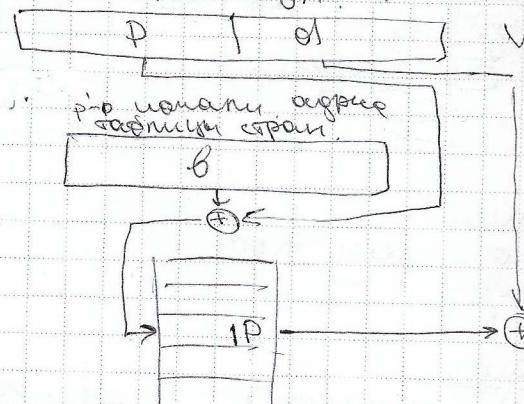
⇒ у неё адрес прям. памяти

но инк. ест. есть обратн. / блок
примес

Блок. страницу \rightarrow ОН \rightarrow область ядра или

(много мест)

Вр. адрес.



мног
мест
адрес

обратн. к прям. памяти
запрос

② Ассоциативное отображение

нужн. наличие в системе
табл. блоков ассоц. памяти

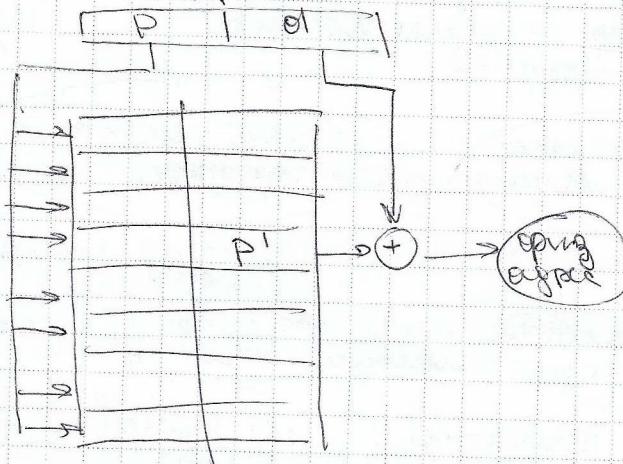
\rightarrow блока реального

направленный
блок

исследов.
блок

нужн. попасть блоку в мифо по кину
(за 1 такт)

Вр. пам.



справ
адрес

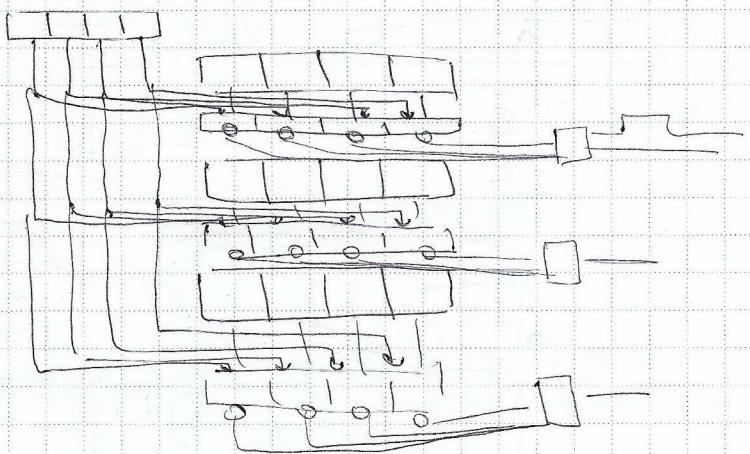
Начину блоки?

(регистр: 1 на каждый разряд есть, чтобы сбрасывать)
сбрасыв. все страницы

сравнение

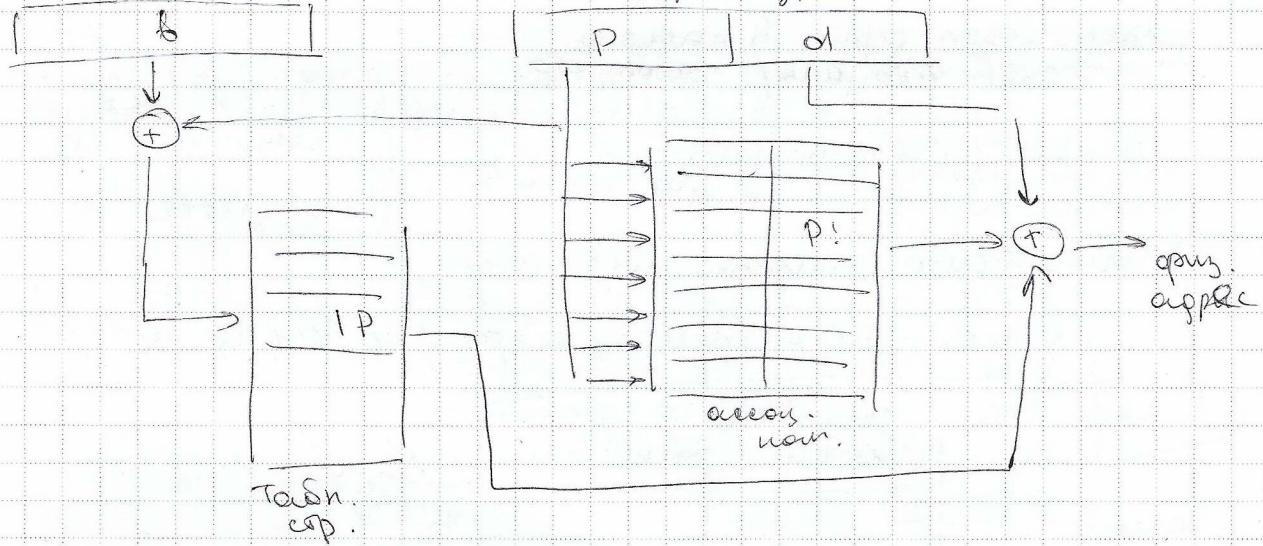
всех строк / разрядов
с кинем

байтами



③ Ассоциативное прямое обращение
P-R на ч. ст.

Быстро. оуп.



Вращающее прерывание \Leftrightarrow исключение
непрерывное.

т.к. несе осю как сп. зону.
 \Rightarrow программа будет прерыв.

CR2 - регистр. нач. адреса блока. обр. к спр.

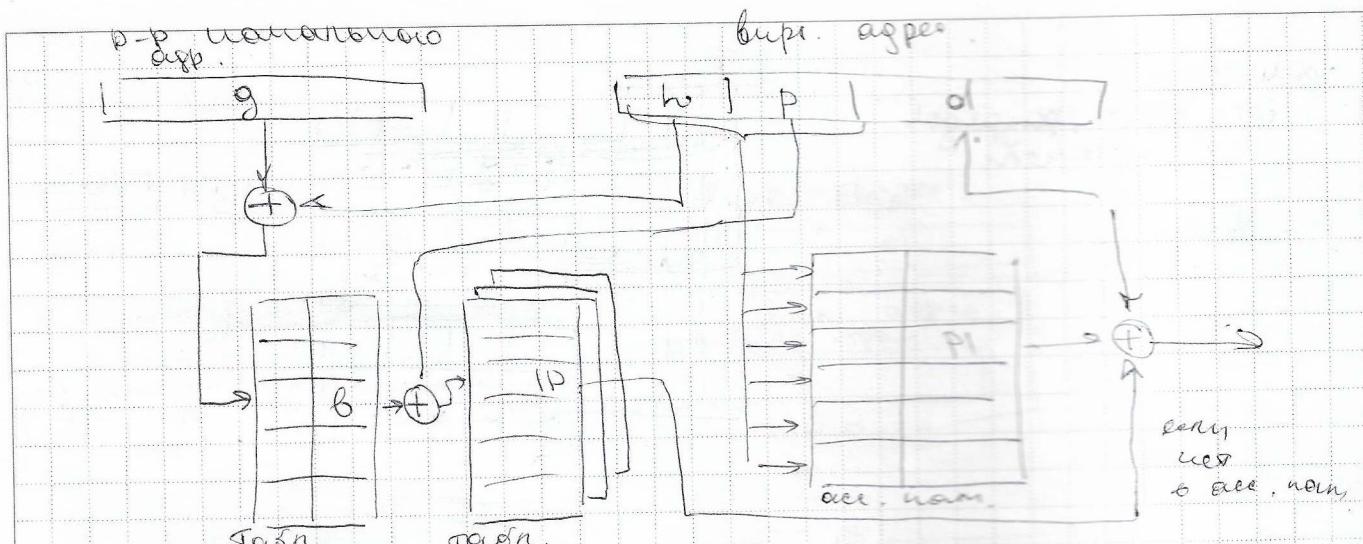
CR3 - регистр. нач. адреса кадра. логич. спр. страницу

ког посет \rightarrow регистр. габар. спр. убени.

Два уровня вложенности страницы организаций
(стан. 6 IBM 360/67 IBM 570)

введен в кинер. страницы \Rightarrow апп. преобразователь
 \Rightarrow кинер. страницы
 \Rightarrow страницы

\Rightarrow страницы кинер
страниц



+ одно сочт. к приз. памяти

имеет возмож. для приз. в памяти
только актуальной ячейк. стр.

Быстро одно сочт. прерыв.

→ для прерыв.

- открытие стр.

- открытие ячейк.

Увеличение памяти задается

возможностью хранить в прошлой ячейке прошр.

в кнн. хранил. приз. адреса страниц
к которым должны под. сочт.

Зачем?

из. стр. страниц, сочтение : если к странице было под.

страница след. под.

страница и т.д. страница

(свойства подчиненности)

наследует
страницы



наследует страницы
даже если страница
имеет переход

страницы ← единиц. приз. памяти

⇒ разнр. приз. страниц в системе

Задача концепт. менедж. приз. страниц в системе

занимает больше памяти
в единиц. и т.д. все когда

Что бы это испр.

программист → реализация

(использование выхода)

Изменение
страниц
из-за

измен. страниц. Передел программы. и конф. набору страниц

Алгоритм замещения страниц

See also esp. введение

необходимо загружать страницу
предвар. загруженную группу esp.

Page replacement

② Возмещение страницы страницы
(до вторичного назн.)
- первоначально все

- + малое издержки
- + не загружены изначально
- Вторич. часто помен.
или только что загружены

② fifo
какой esp. и приб. или временно
БАНКИ или обыч. список (очередь)

- до intake очереди надо менять
- создает "заклинание fifo"
очереди. предпол.
при убен. назн.
→ ~~умн.~~ умн. esp. предпол.

	4	3	2	1	4	3	5	4	3	2	1	5
M=3		4	3	2	1	4	3	5	4	3	2	1
		4	3	2	1	4	3	3	3	5	2	2
		4	3	2	1	4	3	3	3	5	2	2

$8/12 = 75\%$

	4	3	2	1	4	3	5	4	3	2	1	5
M=4												

$10/12 \approx 83\%$

(3) $h \& U$ - most recently used

	4	3	2	4	5	3	5	4	3	2	1	5
$M=3$	4+	3+	2+	1+	4+	3+	5+	4	3	+2	1+	5+
	4	3	2	1	4	3	5	4	3	2	1	
	④	③	②	①	4	3	5	⑤	4	3	2	
	+	+	+	+	+	+	+	+	+	+	+	

	4	3	2	1	4	3	5	4	3	2	1	5
$M=4$	4+	3+	2+	1+	4	3	5+	4	3	2+	1+	5+
	4	3	2	1	4	3	5	4	3	2	1	
	4	3	2	1	4	3	5	4	3	2	1	
	4	3	2	1	4	3	5	4	3	2	1	
	+	+	+	+	+	+	+	+	+	+	+	

алгоритм памяти соответствует

свойству "локальности".

(если обрамление было к строке
то наиболее вероятно что следующее
обрам. будет к этой же строке)

свойство "включения"

(если начало обрамления выбрано в реализации
 $h(P, M, +)$ то эта же обрамляя будет выбрана
в $h(P, M+1, +)$)

относится к классу методов вытеснения

← стековое алгоритм

крайние заграницы
алгоритмили подпр. бран. места при обрамл.
к строке
либо редактировать обрамление
стекомкак часто обрамляется
на конец конца
и сколько разредактировать обрамление
стеком

④ LFU first frequency used (наименее часто меняемая стр.)
Page Replacement

использует систему обратимое к странице

⑤ NUR not used recently page replacement (аналогична LFU)

Дно это организует
какой страницы
все для обработки

переодически
составь в "0" | при обработ-
у страницы, к "0" | если в буфере
с номером
нен. страницы
обрат. к ней не
дено

организации:
указатель удаляем

номер
страницы | *next*
страницы | *prev*
страницы

next
страницы

0	4	0
1	5	1
2	2	1
3	1	0

next
страницы

стр.	стр.
0	4
1	5
2	2
3	1

страницы | *next*
страницы

ноне загружен 5
страницы

если в этот момент
необходимо удал. стр.
то проверка след. стр.
со второго фрейма

у второго фрейма - "1"

у третьего фрейма - "0"

Бит модификатора - "dirty"

не нужно копирование
т.к. форма конеч стр.
то вориной наруш.

Бит сбросу.	Бит модиф.
0	0
0	1
1	0
1	1

нонр. страница до момента
сброса дес. страниц в "0"

Поведение программ и производительность

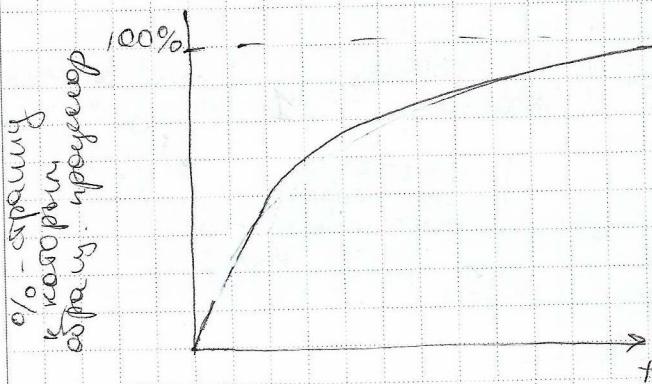
как захваты
mem. виртуал.
намесы

увеличение
капитала
памяти программ,
уровни

максимальные
размеры остатка памяти

тесты:
запуск. процессор
сост. памяти

График зависимости производительности от времени
которому подлежит памяти



программы не обр.
ко всем своим ограничениям

- демонстрирует
что процессор в отрезок времени
не обрабатывает, то всем своим
ограничениям

- не линейной
(сначала линейной)
сначала интенсивно
загружает

Теория работы памяти

Зависимость от прерываний
от объема памяти
- авт. обработка памяти
сравнительное поведение
программ

при этом мы видим что
- поведение программ
во времени всплески
и авт. обработками
т.е. программы обрабатываются
к различным конечным ограничениям.

1968 г. Бенниус

предложен в качестве показателя производительности
число обращений к памяти, при которых программа
запрашивает память за некот. интервал

Размер рабочего множества
авт. памяти. определяется

т.е. при $\Delta t \uparrow$ - число ст.
будет стремиться к пределу L
который определяется от программы
кажд. программы при запр. памяти.

Working
set

$\rightarrow W(t, \Delta t)$

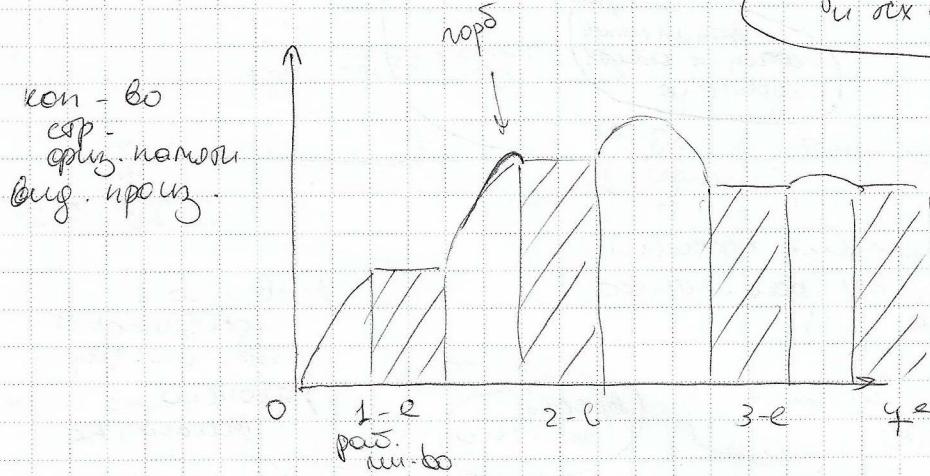
тесты:
запуск памяти

Задача выполнение

- выполнение процесса без прерываний
если процессу удастся запрограммировать свое рабочее множество
(все ст. к концу работы ф. д. Δt) \rightarrow процесс будет выполнять
сез. ст. непрерывно.

Ноуфс гониту идет вогн. загруз все падает ми-бо

→ подкашка / продувка (thrusting)



В нач. момент

происходит вогн. напи. кон-бо спр. (3 DS, CS, SS)

→ интенсивное подпр. нумер. спринту

ноуф в пад. не окан. 1-е пад. ми-бо

некоторое время — без спринтинга прерыв.

подкашка (норс спарс пад. ми-бо + некое пад. ми-бо)

hit rate — число успешных оброт. (спринтинга успеш.)

Выбор размера спринтинга

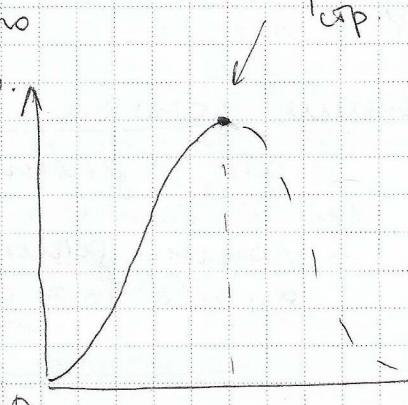
↓
на маленькой
спр. большем.
Большой %
команд

чем меньше
спринтинг
→ тем больше
задири спр.

размер = размер
спр.

Задири разн. спр.
при спр. более напом.
на много спр. прерыв.

→ прерыв.

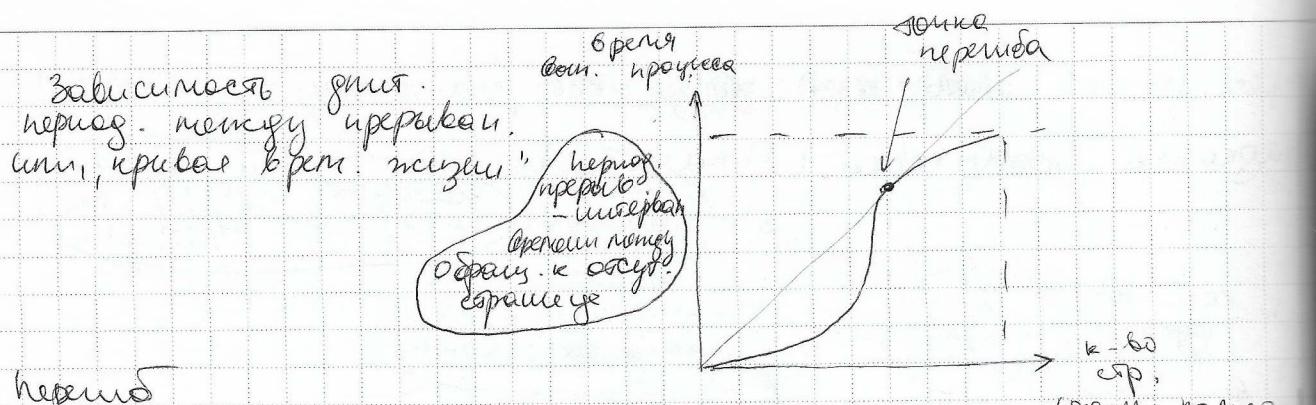


из прер.

много прерыв. расчет
с расчетом разн. спр.

чем больше спринтинг
→ меньше % команда
⇒ нечего на спр. спр.

разм.
спр.



перевод

- из-за некоторый момент времени в начале окн. ве. раб. ми-бо страницу процееса

- увеличии байт

коэффициент
недоступа
для фиктив.
рабочего
момента

ми-бо

бреж. мом.

кон-бо процееса процееса

- авт. замк. показ. и рабоч.

воспользование памяти
(чех. предоставл. процееса чужим
для кон-бо страницу)

квест

загрузка и закрытие

загрузка/закрытие страницы

загрузка любой
страницы
для того что бы
процеес мог
загрузить стр.

загрузка нового
страницы для него
процееса

Page demon - UNIX

Swapping - Windows

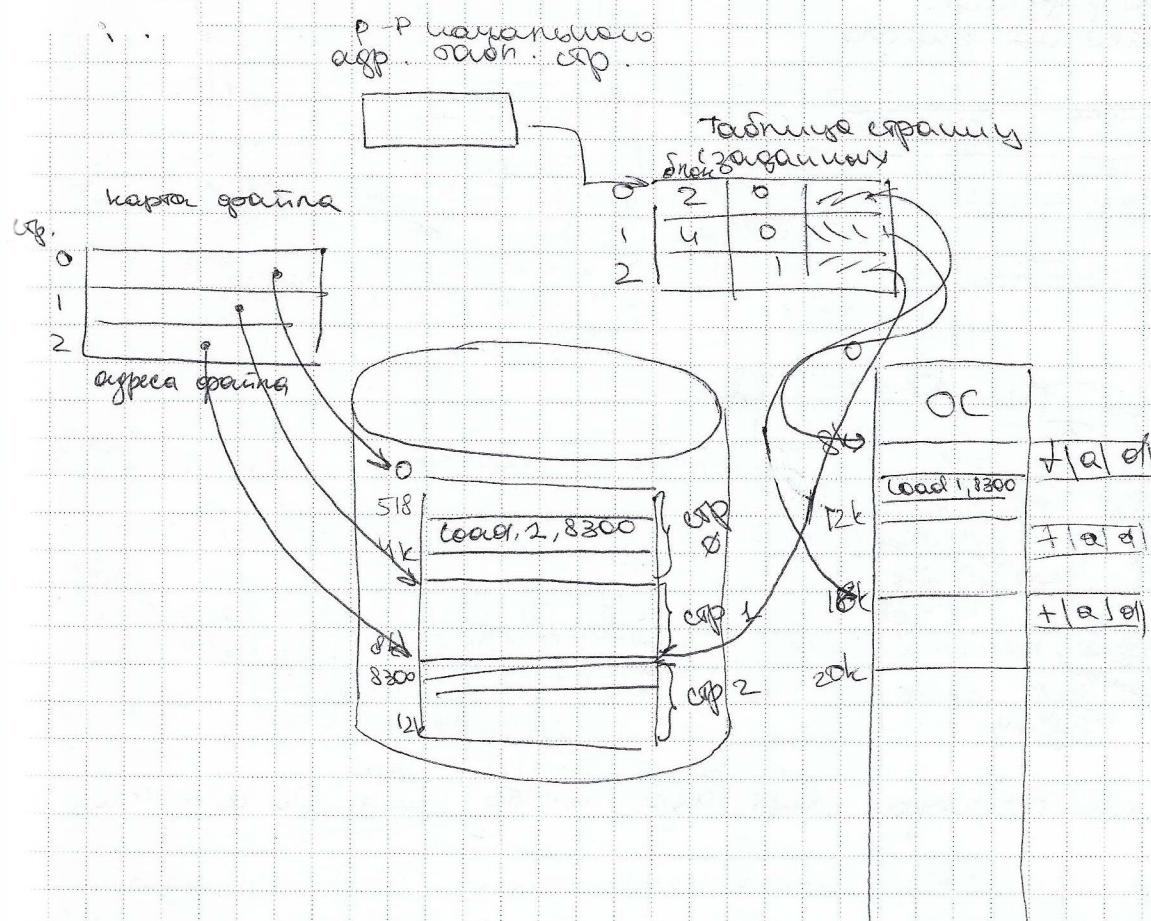
Управление памятью скриптов на языке

страница - приз. генерации памяти

скрипты - норм. генерации памяти
если мен. байт избытен его размер

Лекция

Управление памятью в процессорах INTEL



Адресное пространство ядра

Swapping / Paging

результаты
выбора в резерво
(б. страниц с деср.)

страницы подсистемы

- общий адрес
(создается разн. приложн.)
характеристики

= хранится в ядре
(б. коп. наимен.)
на эмуляц. чип. ядр.

В сист. чип. сегн. сегн.
one Paging

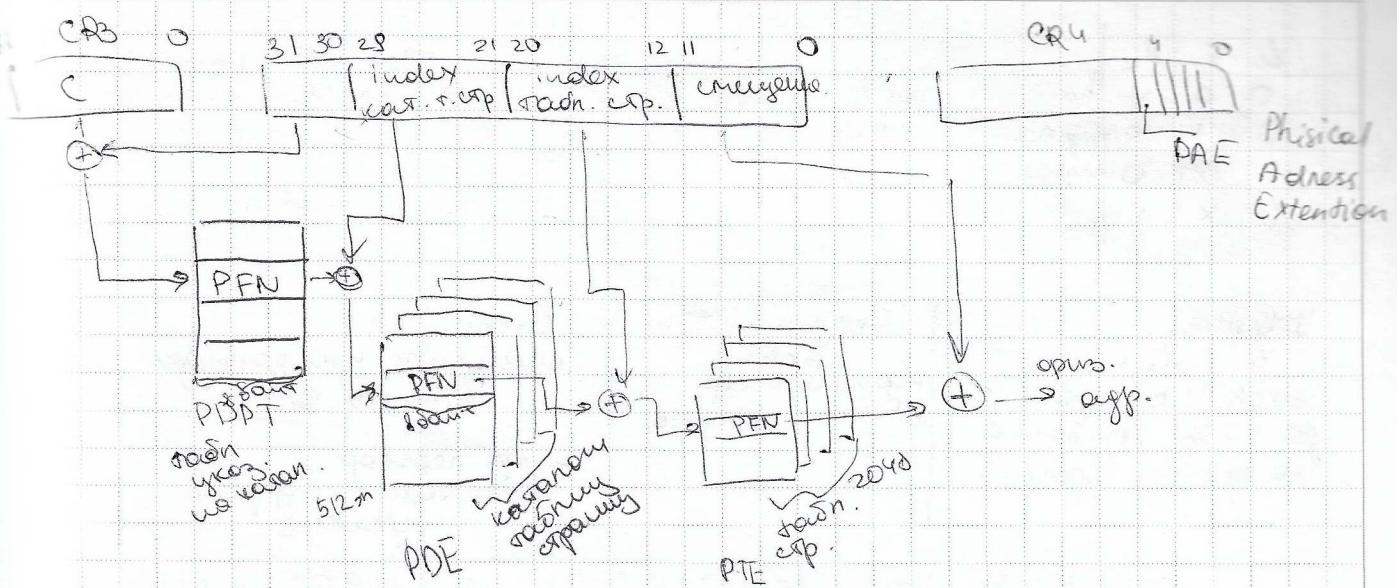
программа занимаетadr.
предп. ядре
комп. в сегн. paging

⇒ компонование
⇒ ср.

чип. в систе
б. трех комп.

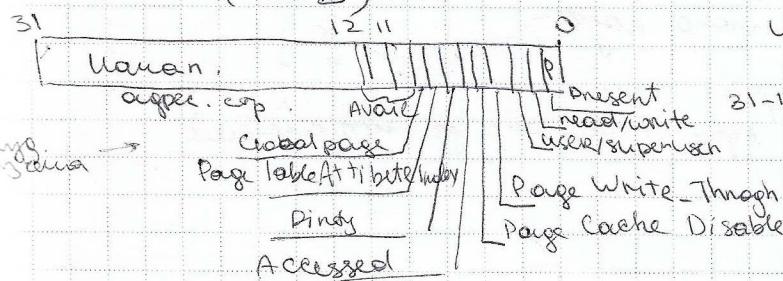
Одноразмерные память (AS100)
- все загруженные программы в ОН

Paging вadr. предп. страниц | имен чип. таблицу



Декомпозиция страницы.

PTE (УКБ)



"назн. адрес. срп."

"назн. адрес. срп."

"назн. адрес. срп. page base address"

3 процессорах Intel奔腾微处理器 cache (TLB) имеет 180 нодов

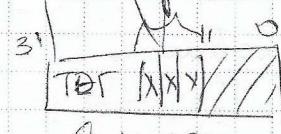
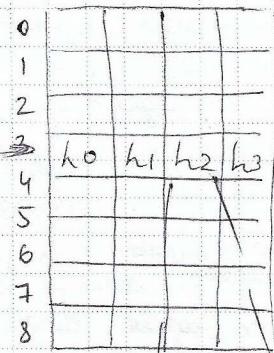
Быстро
запомни.

hRU

TLB

Translation look buffer

→ no burst.
список



Разр. срп.

cache назов срп.
адрес. срп. к квадрату
запомни срп.

запомни в cache не назов
→ срп. к квадрату срп. 80

запомни

так срп
имеет 180 нодов
→ 8 нодов
адрес. срп

запомни адрес. срп. cache

запомни срп
к квадрату срп.
адрес. срп
→ base address

запомни
← квадрат hRU

так срп
назов срп. квадрат

и запомни срп. LRU

имеет 180 нодов
адрес. срп. квадрат

адрес. срп. квадрат
запомни срп. квадрат
адрес. срп. квадрат

b_0	b_1	b_2	
0	0	X	h_0
0	1	X	h_1
1	X	0	h_2
1	X	1	L_3

Утвержд.
8 июня. нет.
cache TLB управляет.
90-97% запросов
на физич. страницу.

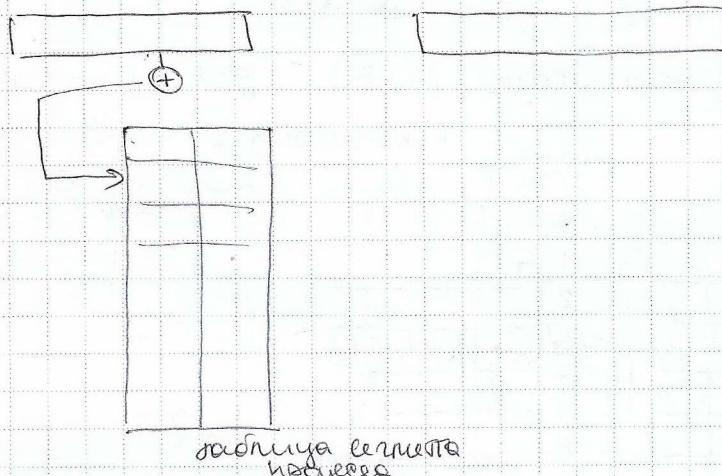
Активизируется
cache
одн. агр. стр.
одного процесса

cache TLB
активируется при загрузке
регистра CR3

- при переходе
на винч. другого
процесса

Лекция (дистанция)

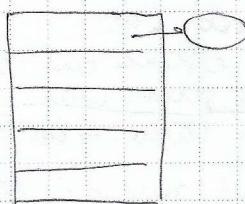
Управление памятью сегментами по запросу
размер сегмента - объем проир. кода
при сегментации в физич. пам.
такж. надо опр. разм. сегментов



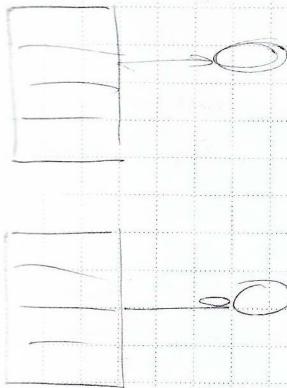
Если сегментение выходит
за размер сегмента - ошибка

Подходы к организации. задачи

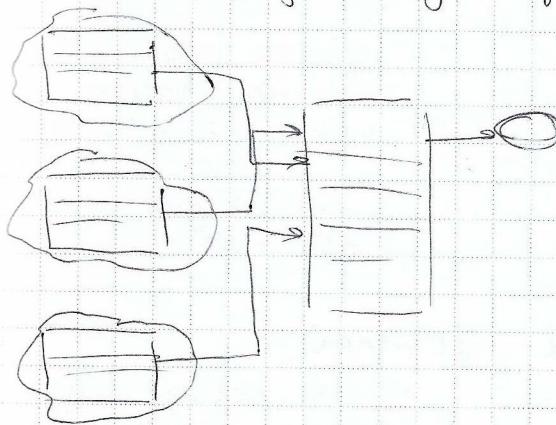
- единая таблица
- все сегменты описаны в одной табл
(небольшое измен.)



- локальные таблицы
 - каждой процесс, его адресное простр. оник. в лок. табл. расп. расп. (необходимо ук. симб. физ. написи)



- локальные таблицы + одна глобальная таблица
 - таблица сог. ссылку на физ. в глоб. табл.



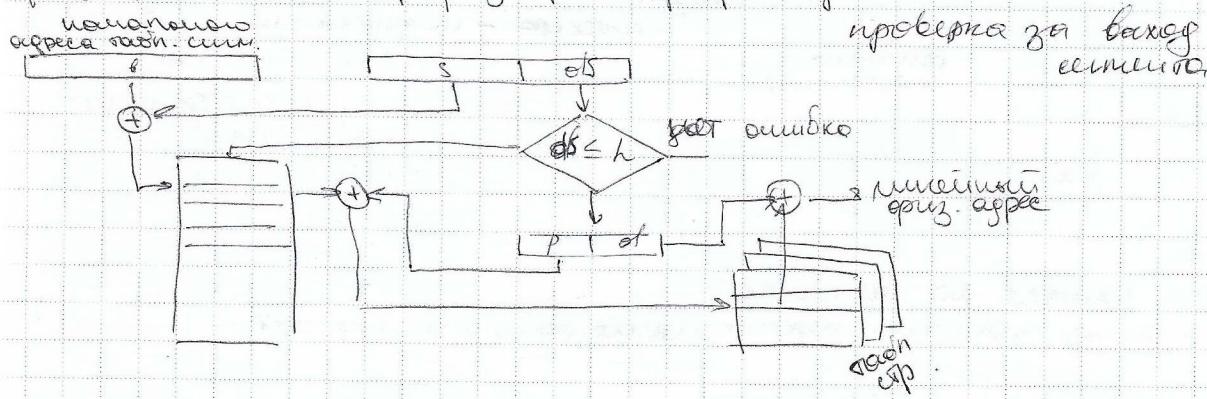
Справочник по запросу
переводчик - кон. не использует, справочник
нет привязки, справочник

Управление памятью с использованием погрешности на
справочник по запросу

результат симметрия кратен разнице справочник
лок. уровень пресбр. : календарный процесс имеет симметрию

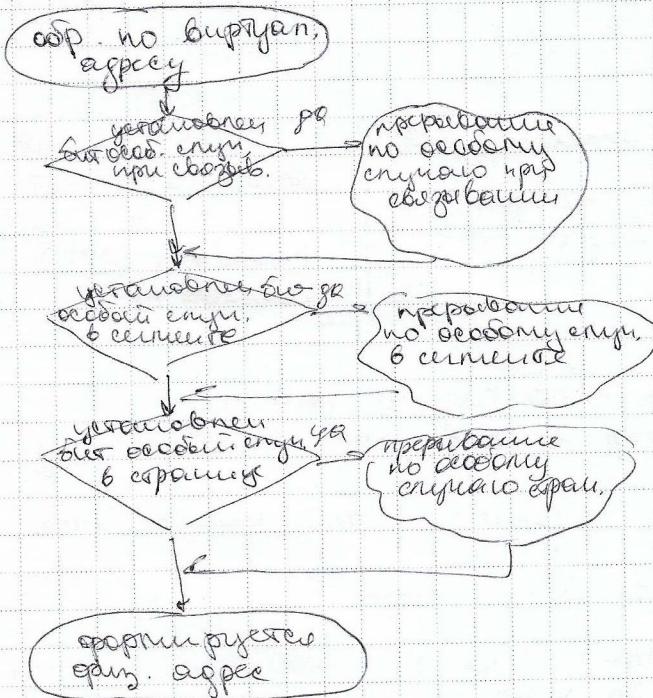
и справочник

результат симметрия опр. разницы пропр. когда



delet del set - управл. кир.
написано

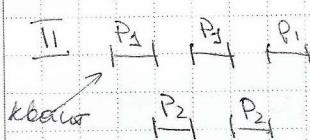
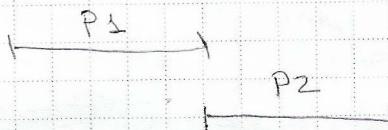
Сообщение логи. прерывание
при set + cop.
(сравнено - сравнивное напр. написано)



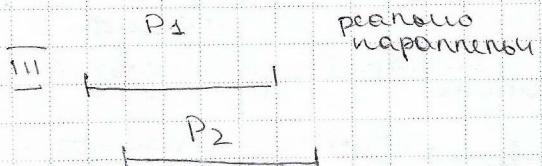
Зависимость параллельных процессов

Управление наблюдение

I несп. зависим.



небывает одновременно
какому-либо
именно
когда параллельное
издание с переключением
контекста - информационный



Мыло имеется для сравнения
в конце гонки или неизменить не один и тот же цвет

$$P1: \dots \\ S = S + \text{delta1},$$

$$P2: \\ S = S + \text{delta2}$$

P1. mov eax, myvar
inc eax
mov myvar, eax

P2: mov eax, myvar
inc eax
mov myvar, eax

SMP

P1 на myvar		P2	
mov eax, myvar	∅	∅	
		0 0	mov eax myvar
		0 1	inc eax
		1 2	mov myvar, eax
inc eax	1 1		
mov myvar, eax	1 1		

нечёт
данных

равное
странице

Ахуим (ахуим)

Взаимодействие параллельных процессов

Небезопасность

Часы коса age срабатывает, к реал. времени
→ критическая секция

Без различия какая параллель.

myvar - fence needed
неподелю

Монополией доступ
- если один процесс
до другой процесс не может
дойти в критическую секцию
по этой причине.

Если один процесс блокирован
то ему ставится, который
не может пройти испр. испр.

адрес. испр. бывает несколько

Матрица

1 Программист (исп)

2 Аппаратный ← fast and set

3 Использование семафоров

4 С использованием
мем. ядра

examples

flags, flag2: logical,
P1: while (1)

{
 while (flag2);
 flag1 = 1;
CR1; →
 flag1 ≠ 0;
PR1;
}

P2: while (1)

{
 while (flags);
 flag2 = 1;
CR2;
 flag2 = 0;
PR2;
}

// наработки
flag1 = 0;
flag2 = 0;

Ошибка не падает!

Делаем реальное задание
также где для напоминания
используем

Требует перечислений
- все определяются

program Dektor

flags, flag: logical,
que: int

P1: while (1)

{ flag1 = 1;
 while (flag2 == 1) {
 if (que == 2)
 {
 flag1 = 0;
 while (que == 2);
 flag1 = 1;
 }
 }
}

P2: while (1)

{ flag2 = 1;
 while (flags == 1)
 {
 if (que == 1)
 {
 flag2 = 0;
 while (que == 1);
 flag1 = 1;
 }
 }
}

// наработки
flag1 = 0;
flag2 = 0;
que = 1;
pair begin
 P1, P2,
 parent,

Существует много программных

Аппаратные (test-and-set) - машинные команды

IBM 370

использует 2 разн. явл. неп. B
компьютер B неп. A
⇒ где B - условие
"чтения"

использование команды
тестирования проверку и
установку единич. в
единиц. памяти
- блокировки)

0 - доступен
1 - недоступен

Взаимное исключение

```
program test_and_set;
  flag cs, c2: int;
  P1: while (1)
  {
    cs = 1;
    while (c2 == 1)
    {
      test_and_set (c2, flag);
      C1;
      flag = 0;
    }
    P2;
  }

```

// начало установки
flag = 0;
parbegin
P1 P2
parend

```
P2: while (1)
{
  c2 = 1;
  while (c2 == 1)
  {
    test_and_set (c2, flag);
    C2;
    flag = 0;
  }
  P1;
}
```

знач. flag = 1 - когда модуль из np.
находится в единой ячейке памяти
0 - в привилегии, доступ

Проверка - явл. блокировка
spinlock.

simple mutex

Upisat spin_lock (spin_lock_t *c)

{ while (test_and_set(*c) != 0)
 /* recursive game */
}

Videti spin_lock (spin_lock_t *c)

{ *c = 0;
}

Druženstvo je vse enak. vse mora nametu
→ ve resevanju

Upisat spin

{ while (test_and_set(*c) != 0)
 while (*c) = 0;

↑ vseb. zavrsava vse mora nametu

Монитор ресурса - монитор

'ресурс': npayes resource - монитор управляемый

non-cop. gathering

npayes resource - управляемый

(Xoapa)

monitor. resource

var

nn: int // количество

wnt: logical // availability meetings

c-read, c-write: conditional;

procedure startread

begin

if wnt or turn (c-write) then

wait (c-read);

nn = nn + 1;

signal (c-read);

end;

procedure stopread

begin

nn = nn - 1

if (nn == 0) then

signal (c-write)

end;

procedure start write

begin

if (nn > 0 or wnt) then

wait (c-write);

wnt = true;

end;

procedure stop write

begin

wnt = false

if (turn (c-read)) then

signal (c-read)

else

signal (c-write)

end;

begin

nn = 0;

wnt = false;

end;

ресурс монитор управляемый
запись - запись
чтение - чтение
запрос - запрос

4 определения
запись - запись
чтение - чтение
запрос - запрос
запрос - запрос

ресурс монитор
запись запрещенное

Лекция

```
3. var choosing: shared array [0..n-1] of boolean;
2.           number: shared array [0..n-1] of integer;
1.
3. repeat
4.   choosing[i]:= true;
5.   number[i]:= max'(number[0], number[1], ..., number[n-1]);
6.   choosing[i]:= false;
7.   for j:= 0 to n-1 do begin
8.     while choosing[j] do (+nothing*);
9.     while number[j] <> 0 and
10.        (number[j], j) < (number[i], i) do
11.          (+nothing*);
12.      end;
13.      (+critical section*),
14.      number[i]:= 0;
15.      (+remainder section*);
16.      sentie false;
```

5, 6. - выбор номера процееса

10 лексико-грам. описание

Запись

$$(a, b) \in (c, d)$$

лексико-грамм. - норигон исчина лени $a \leq c$
и если $a = c$, то $b \leq d$

14 - если процеес не находит в крит. участок $\Rightarrow \emptyset$

8 - при неко неко гр. процееса не соп. это

correct access

Блокировка $\neg \exists x \in \mathbb{Z}$, но это необходимо

т.к. блокировка приводит к тому что другие задачи могут не получить доступ к блоку

Задача:

проблема синхрон параллельного

процееса - хочет погрызь

результат - крепко

классический задача синхрон и нее нроя. блокиров.

один параллель

один крепко

много ступней

если параллель ступней не

параллель имеет в крепко

если ступней нет, то нечего уходит

Конкуренты!

проблемы

одинаковые наработки и клиенты и асинхронные

клиент может залог

→ наработки разбросаны

→ клиент в привычное

→ наработки в привычное

для клиента одновременно
один ответ

у конкурента

вместе процессы "асинхронные"
(одинаковые скорости)

недостаток синхронизации

→ проблема взаим. паралл. процессов.

"Reactive" в оконке и ажде

Ноу: это процесс с асинхронными промисами

≈ task

сонах входа

если имеем ~ задачи связанные
при помощи входов

dark < func > is

entry < тип > входов

(дискр. функции)

(прот. часть);

< функция обработки входов >

end < func >;

Если одна задача

бывает един. ко входу

группа задач

→ одна задача теряет

доступ к функции

или "Reactive"

ноу Reactive

→ задачи синхронные.

стартует синхрон

→ одна зар. запр. реагирует

с группой задач

→ группе задач

может принять

многие ответы

Reactive един.

однако когда

исход. привод

→ accept. - принять

accept < тип > входов

(варианты)

(прот. часть)

do нечег. определов

end

однобр. напр. задачи

нах. в Reactive

результат нечег. определов

(без блокировок)

→ выражается как отдельный

модуль

имеет собств. название

Расч. процессов в
асинхронизируемых часах.

MAKE

общено в UNIX большие программы
разг. на яз. языков

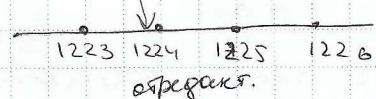
программа make

- проверяет время послед. модиф. всех исходных объектов
- если исх. файлы время послед. модиф. > объекты файлы
то make считает → перекомпилирует.

Расч. систем

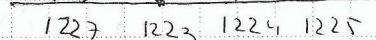
редактирование — перекомпилирует
составные .so

комп. на яз.
анс-ва компиляции



время по яз. часам

комп. на яз.
вспом. перекомп.



свой яз-вой генератор
- свой естеств. яз.
которую помнит
часы запоминают

⇒ перекомп.
и обр.

Если яз-в генерит 60 прерыв. в секунду

⇒ в час 216000

⇒ конк. память может хранить часы в виде
215998... 216002

Проблема синхрониз. по яз. часам

В мире существует много времени

установлено значение времени

которое получ. из разных стран

Может называться секунда на Max скорость → конк. переход
интервалов → конк.ной яз.

Среднее солнеч. секунда
или solar second.

1948 год были изобр. атомные часы
секунда стала определяться как время
за которое идет 1000000000 колебаний

составляет около 9192631770 переходов

50 изображений

TAI (international atomic time)

непрерывное время (last second)
 Всемирное время размножается более 800 раз.
 → встроившееся в секунду
 → универсальное время

UTC universal coordinated time

Алгоритм:

- 1 Алгоритм Кристалла
- 2 Алгоритм Беркли
- 3 Универсальный алгоритм.

Однажды спущенное во / спущенное более

1878 год Лондон
 синхрониз. логических часов

Суть алгоритма
 - категорию сообщения присваивает время сопровождения
 по локальным часам
протекло отправление

Три компьютера

0	0	0
6	8	10
12	16	20
18	24	30
24	32	40
30	40	50
36	48	60
42	56	70
48	64	80

процессы категорий напр.
 сравнивает время
 если сообщ. время меньше или
 равно
 → сообщ. время уст.
 на 1 больше
 времени сообщ.

Данный алгор.

составляет правильную идущ. сообщение
 явл. едином распр. методом решения
 задачи врем. распредл. по логическим часам

Такими часами
 можно пользоваться

Антиромко взаимное исключение
в реестре.

+

Лекции

Логические части Аппарата
использоваться в реестрах
не прописывали раньше

одн. назб. сбрасывал
с исполн. с синхр по часам
<регистратор>

- позволяет новому извещению
изменять предыдущий реестр
несколько конц. данных

- врем. централ.
- исчезающая. централ.
- промежуточная центральность
- fifo централ.

Пример

Две новоменяющие производств.
запроса банк
может произойти конец BD
к 1
запрос к банку. конец

Пока же обеими
бывает. ошибок
исходя результат.

↓
создание всех конец
ог.

Для того чтобы не происходило
потери информации
нужно логич. части
использовать части
Аппарата

+

Централизованный

Реализующие взаимное исключение.
использов. с промежуточ. коорд.

В качестве коорд. взаим. промежуточ.
на ходу
чтобы здеш. сеть было
агреса

Если промежуточ. исключение в
сеть. сеть
→ исчез. коорд. запрос (имеет
указ. сеть)

time out

чтобы
исчез.
запрос
запросы
передачи
сеть. сети

коорд. проверяет не паходит ли ур. процесс в крит. секции
либо крит. секции не занята, но есть очередь
в крит. процессе ставится в очередь \rightarrow блокирован

(встает вопрос
врач по имену)

Проблема:
коорд. авар. завершил
 \Rightarrow система разрушается

\Rightarrow две стро

модель процесса однор. отсутствие коорд.
 \Rightarrow процесс имеет новый коорд.
"Алгоритм забывки"

Если процесс забывает. отсутствие коорд

\rightarrow новый коорд

\rightarrow восстановл. имен. запрос (всех

указов:
- свой номер
процессу с добавленными
номерами)

\rightarrow процесс который помнит. сооды.
сравнивает его номер

если номер совпадет

\rightarrow новые коорд

Итак:

один процесс с неким номером
 \rightarrow координатор

процессы у которых номер
меньше
процесс получает сооды.
от нового коорд

(старый номер)

Каждый процесс

установлен на машине
единица хранит оба задания
содержит. единицу, что бы стать
коорд.

где наст. на
координатор хранят

Распределение

Процесс телеграфный входит
в крит. секцию

\rightarrow распределяет $N-1$ сооду.
запрос всех процессов
сидентов + время (коорд.)
на свой тек. пакет

I процесс помнит. не паходит
и не соб. в нее входить
 \rightarrow разрешение

II процесс паход. в крит. секции
 \rightarrow не помнит. разрешение.
сооду. запрос в очередь

III получатель хочет сам
войти в крит. секцию
 \rightarrow сравнивает время (запрос)
если его время больше (разрешение)
либо меньше \rightarrow (забытие запроса)
в очередь

Процесс получивший
такое сооду. запрос отменяется.
и движется вперед из списка.
в крит. пакет.

Процесс может войти помнит
 $N-1$ сооду. разреш.

Предмет:

аварийное завершение

(синхронно)

время)

→ невозможность выполнения
работы

(менее надежна < критичн.)

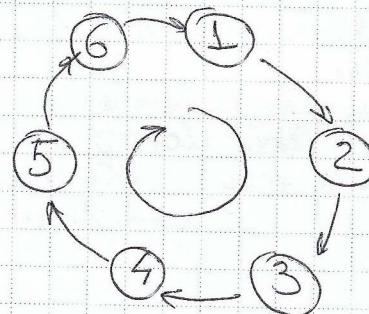
⊕ Token Ring

процесс входит в лог. кольцо
в котором находится процесс

запуск

- кольцо в кольце

- кольцо блок. процесса



кон-бо токенов

= кон-бо крит. секунд

С кандидатом крит. секунд
→ свободные токены
(свобод. кольцо
без кольца по кольцу)
Point to Point

Когда процесс попадет токен
и остановится

поскольку ли ему попадет
в кольцо, секунд

из-за

входа в кольцо, токен

запуска

попадет токен дальше

надежность:

аварийное завершение

⇒ лог. кольцо разорвано

Данные
предусмотрены
разрыв
логич. кольц.

Не генерные транзакции

бесконтр. средства ведения бизнеса

Могут транзакции

если это чтобы подсчит. счет.

набор транзакций

- begin trans.
- end trans.
- abandon trans.
- read
- write

если ТР. макс. б
процесс ведения
до никакой ТР.
процесс не может
уверенно промен.
результат.

Свойства

- упорядоч.

- независим.

- постоянство

наличие рез. что если уда
или более ТР. ведения
нара

10. Конечный рез.

Было вед. так если бы
все ТР. вед. нач. вспр. исп

исп. спос. ТР. никакой
если не может быть
результат. ее ведения

Два механизма реал. независим. транзакций

I Инициализационное
рабочее простр.
процесс имеет вид.
раб. простр
(конн. крайних
и объектов
исп. транзакций
изменение от -/1-)

не наличие изм.
в них. не ведения
никак ТР. не заверш.

крайне затратно
→ много конн.

II Список параметриз.
медиар. исх. крайних
или объектов, но
перед изм. любое
блока любого остатка
или изм. сопротивления
специ. журнализ. регистр.
записей. старое знан.
и новое, если успех
→ прошл. изменения
в исх. остатке или
объекте
если транз. присущ.
то запись в журн. реал.
старое знан. сохр

если транз. неудача
→ много из журнальных регист.
если "откат"
воздр. в исх. состоя.

↑ расп. множествах остат.

намет потребовать протокол

на разных машинах

на отг. машинах свои наборы данных

⇒ специальные протоколы

Протокол двухсторонней транзакции.

... суть прот

один процесс выполняет
группу. коорд.

→ начинает транз.
и запись заносит в пок.
мнужан
→ назначает подчин.
процессам которые
бывают. т.п.
"награда к задаче"

→ получив
процесса проверяет
задачи и если
и заносит в пок. мнужан

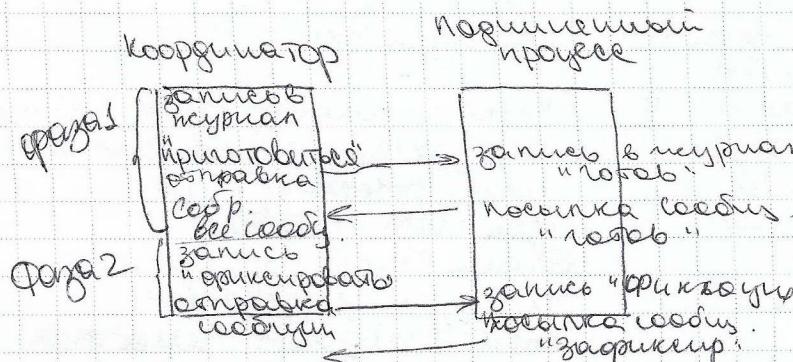
→ назначает
"награда к приказанию"

→ недейств. в сист.
ониющие состоя.
состои. от коорд.

→ коорд. содержит все
состои.
(если один не исполн.,
то транзакция прерывается
⇒ откат)
от всех
→ содержание "приказов"

→ профессия
приказывают

т.е. гарантирует
законч. если
все исполнены

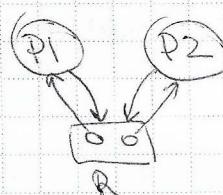


Тупики

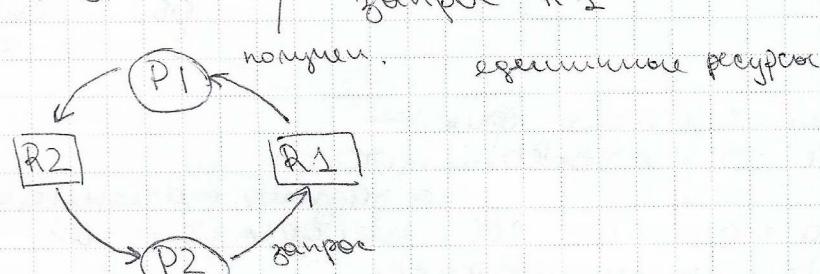
Теория тупиков и логическое проектирование систем "шах

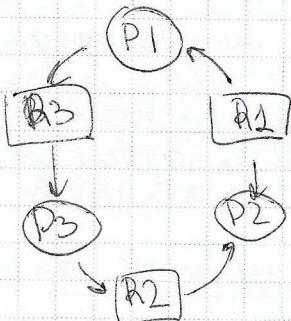
Проблемы:

P1: запрос на ресурс R1
запрос на ресурс R2



P2: запрос на R2
запрос на R1





Тупик

- ситуация возникающая в резул. по конфликтам использ. ресурсов когда процесс блокирует ресурс и запрашивает др. ресурс который не освободит или через цепочку запросов др. процесс, который блок. освоб. ресурса запрашивает первым процессом

В резул. процесса блок. друг друга не снимут и не может сдабодить и продолжить

Тупик ресурсов и тупики

е также замигание использ.

- недостаток использ
- потребление

никак не имеет их свойства
анап. конфликтов ресурсов
(сост. часы) коры
программное обление | ^{область}
(рекурсив. коры ОС
сост. задачи.) ^{друга}

сообщение
(норм. это как если
бы это это перес.
сущесб.)

Активы

Проблема тупиков - остаются

неб. использ. ресурсы
кор-бо в системе избыточно

Определяем кор-бо смысль набора \rightarrow избыточно

Теория тупиков в час. время неактуальна

Условие возник. тупика (dead lock) { ^{актуальное}
кор-бо и б. нр. беск

1. Время неактуальное
или that exclusive

2. Основание
когда процесс удерж. полуц. или ресурс
зап- и когда полуц. доп. ресурсов которые из-за блоки.
hold and wait

3. не передаваемость

когда ресурс неизг. изображ. в процессе
вз. з/з. или з/з. освобожд.

но пренпти

4. краткое отображение

когда ст. заняты. есть процессов

в кратк. канс. процесс заняты

недл. з/з. процессуя ресурс (сиг. в учи.)

сигнал wait

Сигн. три основных метода борьбы с глиняками.

Таки в ОС - сейчас не актуально

Таки характеристики з/з прилож.

з/з различных програм

з/з расп. систем

→ • изопытание глиняков (использование самой возможн.)

• обход глиняков (предотвращ.)

• обнаружение и восстановл.

■ Статеме Ханвендора

который в своей работе показал

что з/з. не з/з. если наруш. есть одно

усл. з/з. глиняка

какое условие?

Определенное требование

по своему началу процесс запр. все необходим

ые ресурсы

в этом сти. процесс может начать

только получив все ресурсы

" - "

• процесс должен знать свою потребн.

• процесс запр. и получ. за ресур

по истин. (чтоб может не менять)

→ не з/з. ресурс. нач. ресурс.

анал. сост. з/з. глиняка

• может привести к бесконечному циклу.

Задача: процесс разбив. на этапы

и задание часть данных. если запр. глиняк. то ресурс

- Упорядочивание ресурсов (неорганическим распределением)
 - ресурсы делятся на классы
 - каждому классу присв. номер и упр. правило
 - по которому процессы могут запр. когда ресурсов < дополнительных наперед, чем они употребляют.

Числ. одно из перв. условий.

если процесс требует ресурс с меньшим номером

- освободить занимаемое
- запросить снова в прав. порядке

тогда. можно не ждать напрв. последовательность

системы реального времени → в будущем попадать не могут

опред. хар. видим. проц. или
секция упр. → опред. время отклика системы

процессы могут быть хорошо изучены
→ можно применить неорганический способ

- Упр. условие не пересекр.

если процесс не имеет полных ресурсов

- больше освободить своих ресурсов

освободить ресурс → отдать до состояния полн. ресурса.

→ запомнить факту отката (Большое проблема)
правильная

чт. реалн. → где можно процесса никак не соб. услов.

Возможн.: захват и освобождение одних и тех же ресурсов

- выход тупиков

„Алгоритм Банкира“ (Дейкстра)

Банкир - владелец ресурс

заемщик что бы вернуть ресур - взять фон. ссуду

Банкир - дать тому кто может вернуть

Условие выполнение (ограничение)

- число процессов ограничено (максимум)

- число ресурсов и один класс ресурса известно

- процессы до нач. выполн. должны ур. в своих заявках (claim)

свои макс. потр. в кажд. типе ресурсов

→ затем проц. не может запрашивать больше

макс. заяв. больше чем есть в системе.

задача нелиней. расп. баланс. но амор.
запас. что сущ. не наступит

→ конечный запас на расп. проверяется
исследование свободных един. расп. в сущ. данных рес.

→ провер. стоки. свободн. расп.
нелиней. имеет пасп. приу. который может
запас. закрепиться т.е.

Процессы текучих ресурсов		Своб. рес.	запасы
P1	1		4
P2	3		5
P3	5	2	9

абн. сущ. безналичн
сток. сумма - ?

абн. сущ. безналичн

P2 - может закреп.
когда запас. 2 един.
(у сущ. есть)

⇒ может закрепиться

⇒ $2+3 = 5$ един. в pool

⇒ может угодн. P3
закреп. P1

⇒ $5 + 1 = 6$ един. в pool

⇒ может угодн. P3

сущ. пасп.
процессов

напоминание:
недостаток - это сущ. сущ. сущ. сущ.
сток. что в сущ.
также сущ.

Идент.	текущ. рес.	своб. рес.	запасы
P1	2		4
P2	3		5
P3	5	1	9

- не всегда можно
получить
→ не присутствует
→ не безналичное
сток. сумма

небезналичн \neq сумма

приу. может не запрашиваться
запрос. кон-бо расп.

Формально

составлен. кон-бо приу. авт. деп. сток. сумма
сущ. есть кон-бо приу. так как, первый запрос входит в кон-бо

1) Для всех k , $B_k \leq A$
 т.е. запрос не может превысить
 баланс един. нет разницы

2) $C \leq B$
 запрос не может неограничен
 запр. баланс нет в заявке

3) $\sum_{k=1}^n C_k \leq A$ никак не бал. баланс
 превысить нет разницы

4) Для анализа всех запросов разницы ресурсов D :

$$D = (d_1, \dots, d_m) = A - \sum_{k=1}^n C_k$$

запрос.
един. ресурс

важен разница
ресурс

5) Матрица запросов E

$$\text{Need matrix } E = \begin{vmatrix} e_{11} & \dots & e_{1m} \\ \vdots & & \vdots \\ e_{n1} & \dots & e_{nm} \end{vmatrix} = B - C = \begin{vmatrix} e_1 \\ \vdots \\ e_n \end{vmatrix}$$

возможные запросы

6) Матрица запросов F

$$\text{Request matrix } F = \begin{vmatrix} f_{11} & f_{1m} \\ f_{n1} & \dots & f_{nm} \end{vmatrix} = \begin{vmatrix} f_1 \\ \vdots \\ f_n \end{vmatrix}$$

7) Проблема состояния
 запр. узлов. все запросы \rightarrow ошибка

$$D \leftarrow D - F_i \quad (\text{запросы-запросы})$$

запрос узлов.
 может быть узлов.
 баланс в них

$$L_i \leftarrow L_i + F_i \quad (\text{распределение+запросы})$$

если запр. есть.
 \rightarrow если запр. есть.
 находит (диагонально)

$$E_i \leftarrow E_i - F_i \quad (\text{уменьш. запросы})$$

Алгоритм:

1. Выбр. перв. запр. запрос P_i
 такой что $E_i \leq D$ т.е. $\text{запрос не баланс не превыш.}$
 если нет \rightarrow шаг 3

2. $D \leftarrow D + E_i$ (непрограмм. но "gg")
 запр. D_i остан. если заверш. \rightarrow шаг 1

3. Если все запр. остан. как заверш. то шаг
 непрограмм. баланс. есть. если. выше в блоке

если не блокируется \rightarrow запрос блокируется и сид. сид. обрабатывается

\rightarrow

$$D \geq D + F_i$$

$$C_i \leq C_i - F_i$$

$$E_i \leq E_i + F_i$$

если по пункту 7

есн. приу. может. как заверш.
на шаге 2 обработ.

$$D \leq D - C_i$$

и повторно может. как не заверш.

Лягушка

II Метод обхода узников
провер. приу. запр. ресурсов
когда они стоят. узников

то проверка (и может ли сидеть
в узниках сид.)

III Обнаружение узников.
ситуации

using. проверяется (или блокируется)
наработ. запрос

запрос опиц. ситуаций в сид.
проверяет может быть.

\Leftarrow

в запросе:
рб не перес. может вернуть
процессов и
может вернуть ресурсов

using: обнаруж. узников
становиться необходимым.
они опред. приу.
которые привели в узник

при этом рб не может сидеть.
вернуть одного из них.

два типа рб

- блокирование
(когда рб не может из верн.
ноги). ресурсов \Rightarrow нет
процессов.

- занят
(находится)

Коррекционность ресурсов
 \rightarrow ненадежный ресурс

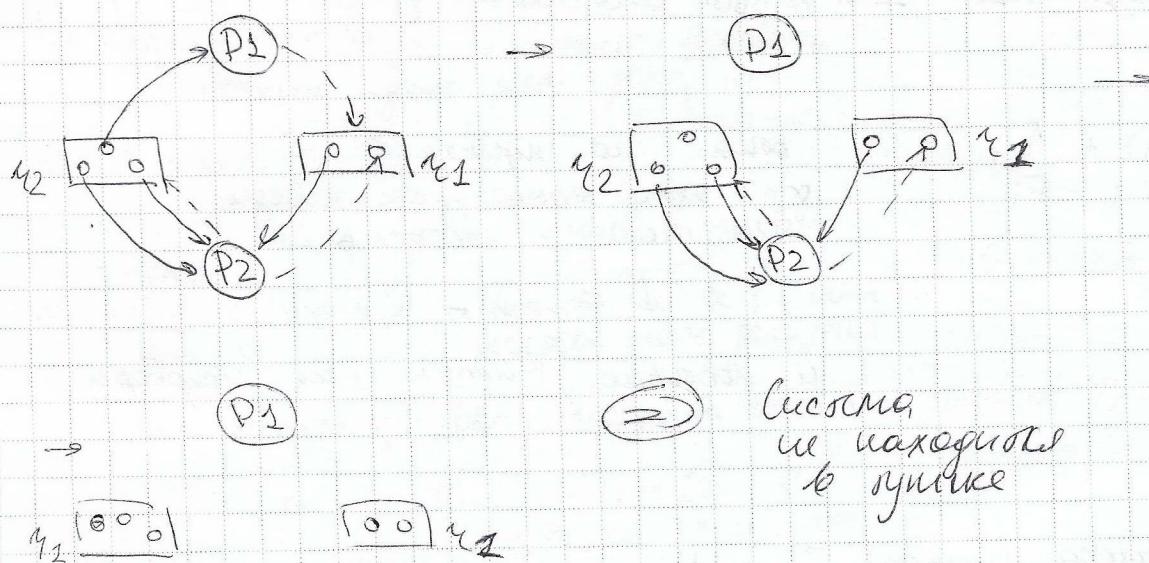
Узниковое сидение

- общеп. методом работы сидера

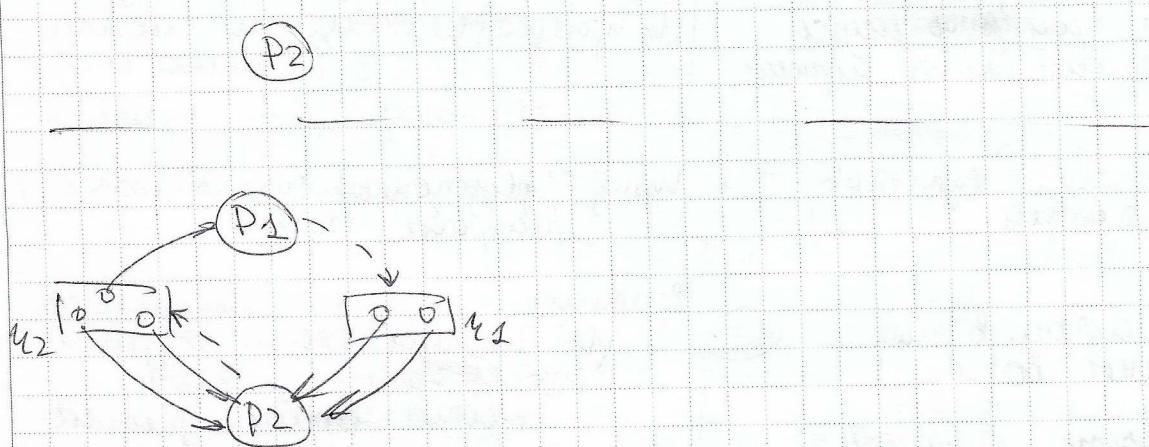
(если запрос приу. может быть убран
то такая ситуация может быть убрана
 \rightarrow может быть обработ. ресурсов по этому запросу
 \rightarrow вернуть в ресурс
 \rightarrow занят - SP.)

(если все рб удалились
 \rightarrow система не находится в узнике)

(если накоплено рб узни. не удаляем (нет запроса)
 \rightarrow сид. в узнике.



Система
не находится
в узике



Две вещи что бы обнаружить
запрос
need: предоставить ресурс.

Две матрицы
- запросов
- востребований

Две вещи что бы проверить
анализ. состояния
и ресурс. матр.

need наше
вектор свободн. ресурс.

Сравнивать векторы с вектором
то что можно востребовать -

$$A = \begin{bmatrix} a_{ij} \end{bmatrix}$$

запрос.
ресурс.

~ можно
запросить
ресурса
вотни. j процесс

$$a_{ij} = \{p_i, r_j\}$$

$$B = \begin{bmatrix} b_{ij} \end{bmatrix}$$

запрос.
ресурс.

~ можно
запросить
ресурса
вотни. j процесс

$$b_{ij} = \{p_i, r_j\}$$

$$f = \{f_j\}$$

запрос
свободн. ресурс.

удалил | Все что осталось
напис. в линии

Если принять условия - никаких опр. не налаг.

тое это что бы обнр. конф. не лежал в буфере

Практика контроля состояния систем

входы и выходы из буфера

1) восстановление рабочего способности

- исправ. заверш. процессов попавших в буфер

- завершение гр. процессов которые выполни. самовыр.

Четыре типа решений

откат до вложенных. конкр. запроса

(запомин. состояния - хранилище до отката)

Свободно отмена

- аутоматическое \rightarrow если "запрос" - присущ.

- согласован \rightarrow "запрос" не имеет

- изолирован

- уединизирован

из одного состояния (если запрос попал в другой \rightarrow исп. гр. запрос)

исп. опр.

из одного

изменение от гр. запрос. и проз.

из одного узла

создание в другом

изменение узлов в гр. запрос.

при заверш. проз. все присущ.

и никакой сооб. не может

попасть

Типы в распределенных системах

общий только сообщен (напр. ресурс)

Избыток проз.

Избыток потреб.

Избыток нейр.

Избыток промеж.

Недостаток
Недостаток

Особенности:

запирание передачи данных в сеть
запирание данных между системами
пред. в синх. состоя.

Резюме. переход в синхронное сост.
носит обозн. не сразу замечал
если носит обозн. прямиком рез.
что система в синхр.
но проблема. рас.чес (синхр нет)

Задача обнар. и исполн. транзакций
связанное к избранным. переходам позорные
обнаружение при работе синхр. ресурсов

при перехода

- передача. транз. синхр.
- отход
- обнар. и утверждение

Блокировка

транзакции при транзакциях

носит налог в активы. или блок

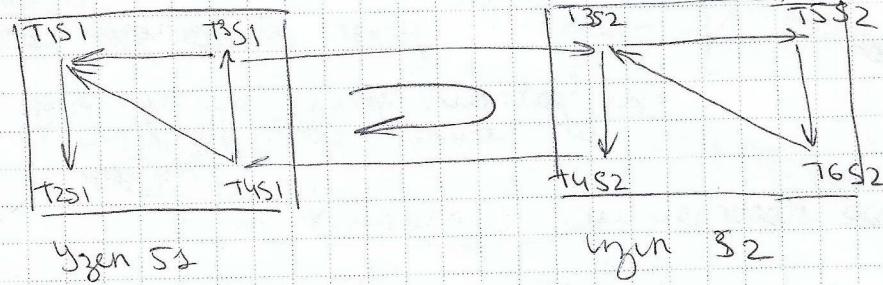
сост. блок \rightarrow неодн. ресурса у гр. транз.

или одн. сост. всех транз. в синх.

избр. грав. можно
1 б. позоры позоры узел
транз. блок. сост. пред.
какими узел обозн. нари
(име. рес. и име. узла)
име. узла - уничтожение

Wait-foreground

"праэр
снижение"



1. Проблема напр. если из T1S1 в T2S1
если транз. T1S1 изн. S1 блок. в синх.
когда из T1S1 освобод. ресурс. избр. нум. T1

1) Узн T1S1 освоб. ресурс. от узла T2S1 ~~избр. ресурс.~~

2) Проблема напр. если из узла T1S1 в T2S1
если T1 в S1 блок. в синх. освобод. от T1 в S1

2) T1S1 освоб. избр. узл. T2S1

тако баг. что неодн. и слож. узлов.
авт. нач. синхр. в праре

Несовместим. тун. син.

если есть прерв. полуц. всех блок.

т.е. транз. полуц. все блок. перед началом выполн.

и сохр. все блок. до бр. транз.

если оп. оп. требует. залог для блок.

→ если неко все неодн. блок. будут доступны

Блок. синхр.

приводит. что син. полуц. анализ. до нон-раранки

(запрос транз.)

приводит к ошиб. залог. опра к синхр.

коротко

транз. нач. выполн. и залог. опра. должны ког. им неодн.

→ блокировка (могут доступ)

Диспетчер блокировок проверяет доступна ли блок.

или нет → берет.

→ транз. полуц. блокировку

если элемент разных задачок. оп. транзакту. (в неодн. ресурсах)

если блок. залог. опра. чтобы проверить

приводит к ошибки. транз. в сеиз. опра и deadlock

в рес. упр. блок. опра — может ли транз. неодн.

или опра из оп. блок. прерв.

• Wait - die

• Wount - wait (сиг. прием)

1. если T_1 сорвие T_2 то T_1 может нон-раранки

ибо $\sim T_1$ — прерывается и перезапускается. потом

2. если T_1 сорвие T_2 то T_2 прерывается

и перезапускается

ибо $\sim T_1$ — может нон-раранки

Обзор.

непр. залог. залог. опра. опра. блок.

если обзор → deadlock — устраняется

если + залог. блок. → если блок прерв. доступна ли она

если нет. то залог задачок. ресурс.

иначе залог. недать

нет мер предсторонности

— могут нанести вред

Две об. функции
func. func. переход. прев. граар на узелов

Если существует путь в узл. структуре.
Несколько. func. возвращает путь из начального узла
используя прев. и отсюда.

Несколько методов. где возвращается "использование"

- прев. самую простую
- прев. с наим. кол-во зап. данных
- где которых включают. наим. кол-во
обновлений
- бывш. прев. с наим. ~~запр.~~ на неизменен
- бывш. прев. каскад. обн. общий узел двух и
более узлов

last

использует где существует ищет. уровень. прев.
Прев. бывает если на вспр. блок.

Создание узлов в вспр. системах

(не все это в вспр. системах)

Вспр. БД

- БД на исключительных
ситуациях и using data
в различных ситуациях

использование

т.к. общий уровень вспр. и разн. ненесущий
уровень и временные
уровни вспр. временные

в перегрузке

одна и тоже пр. может создавать на обл. один
и не активен. на SP.

На самом деле каскад. пр.

одна в не актив. состоянии
также в каскад. состояниях не может

Проблема "место нового. пр."

В этот момент пр. несет опр. данные при первом
использовании на пр.

использование - это он неодн. таблица - список единиц
список имен. Использование и список
один имена несет.

Блок с узлами. узлов

Kогда IP заблок. или блок. или прерв.
или же. отк. на ее cause?

(исследование данных бенчмарка)

Решение:

Чекинки — IP зон под блокировок

Четв. система сайт на которой
происх. ошибка. отк. как
чекп. сайт

этот сайт отк. сообз. сайт.
на которых работают.
Что блок
→ зерг не работает

если сайт Rb1
исправил ошибку. сообз.
о том что были что
запрос. → + ошиб. на них.
если не исправил.
→ IP несет

lock — преград. один. некон. состояния
(lock())

«Чекинки»

Двухстороннее блокир. связи между сайтами
если ошибка. переход в состояния. возникает. наруш
соглашения

если ЧП. сайт не испр. переход — не может связаться с IP. cause
⇒ не отбд. ошибку.

Одно — изменение

answer. ит. в четв. системе
изделия. расп. Группа может за счет прег. отк. сообз. ошибки.
если запрос на блок удаляет.

они. возможные есть расп. преграды IP. конфликт.
— некон. ошибка. (

— конфликт. ошибка.

Ноут возможность из-за расп. преграды IP. конфликт.

↓
меньш. группы
IP. на одном cause

↓
меньш. группы
IP. на ~~одном~~ cause.

3 case конфликт.

— одна из групп
— разные. между в состояния. расп. или. они из
(расп. или)

T₁

T₂

// Transakcijem

T₁ - прибывает на cause P и try block. он. заменяется
который упс. зовх. T₂
на ст. cause

→ cause → cause P

• Distributed wound-Die // расп. рана

- если T₁ упс. T₂ T. e. T₂ пропускает cause P
то T₁ - может перехватить

при этом T₁ может вороти. cause cause P
попытка - T₂ - success или cause иревония

- если T₁ упс. T₂ то T₁ - прерывает

при этом спе. message на cause P
попытка отправить cause. если cause
попытка success. T₁ → неуд. корр. прервать

при этом упс cause - попытка убедить попытка
что T₁ - прервала на всех cause успешна

• Distributed Wait-wait // расп. ожид. ожид.

- если T₁ упс. T₂ то T₂ - прервала

если T₂ ожидает на cause P то cause P
прерывает и ожид. T₂ ⇒ просматривает
cause. на gp. cause

если T₂ попытка cause P (закрытие P)
но попытка на cause Q → cause P открыть.
что T₂ - одна прервала

cause Q - попытка прервать T₂ → отправить cause
gp. cause

или это может означать cause упс. gp.
(из-за P)

если T₁ (в уп. приг. рас) упс. T₂
то T₁ - просматривает, при этом T₁ может
возвращи. cause P → cause P открытие cause
(T₂ - закрытие)

— обнаружение расп. туников

т.к. в чир. сист.

обнаруж. расп. тун. нанеса в син.

но сенс. мод. праор онсиг (см. карты)

нанесе чинка онсиг. в чир. граор абн. приз. туник
или вданси блок

обнаружение спомено

т.к. ТР — распред (онсигает ресурс. в син.)

⇒ онсиг блоки. туник чир. туник

чир. ТР — туник (чир. на спр. чир. врем.)

чир. — всесир. предпол. расп.

и пред. онсиг. врем. заверш. ТР

— чир. не заверш. чир.

→ ориг. признак вогн. вданси блок

Детектор туниковых спомено

в чир. сист — один детек. туника

в расп. сист — может быть несколько

→ может обнар. туник. сист. где некои. ему саисоб

(напр. обнаруже

→ сокр. врем. обнар)

Автоматична гнс обнар. туников в расп. сист.

1. Чир. детектор туника (т.е.) одна локальная

чир. чир. детек. вданси блок.

2. Чир. детектор туника

детектора спомено. в иерархии

гнс расп. сист. как чир. вданси

3. Детектор расп. туников

вс. сист. туник. в обнар и чир. туника

• № 1. Граор (бланс. напр.р.) ав-са нанесеное сокр.

если чир. туник. нанес. сокр. чир. чир. чир. все гнс

если граор нанес. сокр. нанесено, то аланг.

состоит из чир. туник

(см. рис. нанес. сокр. граор)

Что можно сделать?

То есть, который исп. на ресурсе, который может быть ненужен (если в нем нет. или недост. ресурсов)

H2 ~~Простой~~

Если в городе есть. некот. ресурсов
→ недостаток цен. труда

H3 Если S - не авт. сост. труда

$S \xrightarrow{P_i} T$ то сост. T - авт-и сост. труда
если операт. проф. P_i авт. запросом, то он. ресурс
и в результате P_i наход. в T в труда

Алгоритм обнаружения труда в расп. нет.

Chandy - Misra - Maas

Если при операции запр. в нет. недост. ресурсе
запрос P не имеет запр. этого ресурса
переходит ему. сообщение и
посл. это пропускат, которое прошло.
может захватить данный ресурс
В этом ему. сообщ. ук. при члене

1 - свой номер (номер) в нет

2 - свой номер

3 - номер профес. кото-р. носит. сообщ

Проф. номер проф. профес. нет ни у кого нет ресурса.
значит \emptyset . профес.

Если он сам нет (не имеет возможн.)
→ ему. сообщение

они. недост. ресурса

| Если сам нет недост. ресурса профес. имеет вспр.

б) есть нет - номер проф. есть ресурс
который он имеет (profes. сообщ)
→ носит ресурс

Result: если проф. номер. сообщ. и в первом нет
или сообщ. содержит свой номер
и в нем - свой номер \Rightarrow система в труде
(код. ошибки)

Немного лучше. Этот прок.

- если в Linux мало прокс.

→ РАСТОЧИТЕЛЬНЫЙ

Давно. + если номера (столб прок.)

→ можно забыть прок. с наим. номером

Затем если пр. (нр. ам. напротив)

→ в результате можно спр. наим. старше / младше

Блокирование прокс. проходов. если ком.

Пр. запр. прок. можно ли это убрать.

coady. Zouyou, вспомним можно ли

вспоминать using spe. coady. zouyou

⇒ Архитектура ядро в ОС

Два типа

↓

Несколько ядро

одна программа
несколько из подпрогр.

одна из них - ядро.

в програм

→ некомпактн.

Windows / Unix / Linux

Unix - это бывает ядро
без некомпактн.

Многодядро /

несколько ядро
которое выполняет
один из ядр, ядра
при этом ядр. можно. ОС
явн. можно. прок.
которое будет выполни.
в разных АП
в разных ядр.

без некомпактн.

Многодядро ядро

Unix / Linux - минимизировано

Все монол. ядро подр. на прерывания
или. синх. прерываний

типа:

- механизм вызова API (одн. к ОС за сервисом)
- механизм сигнализации
- аппарат. прерыв. (Interrupt)

API - опр. в POSIX

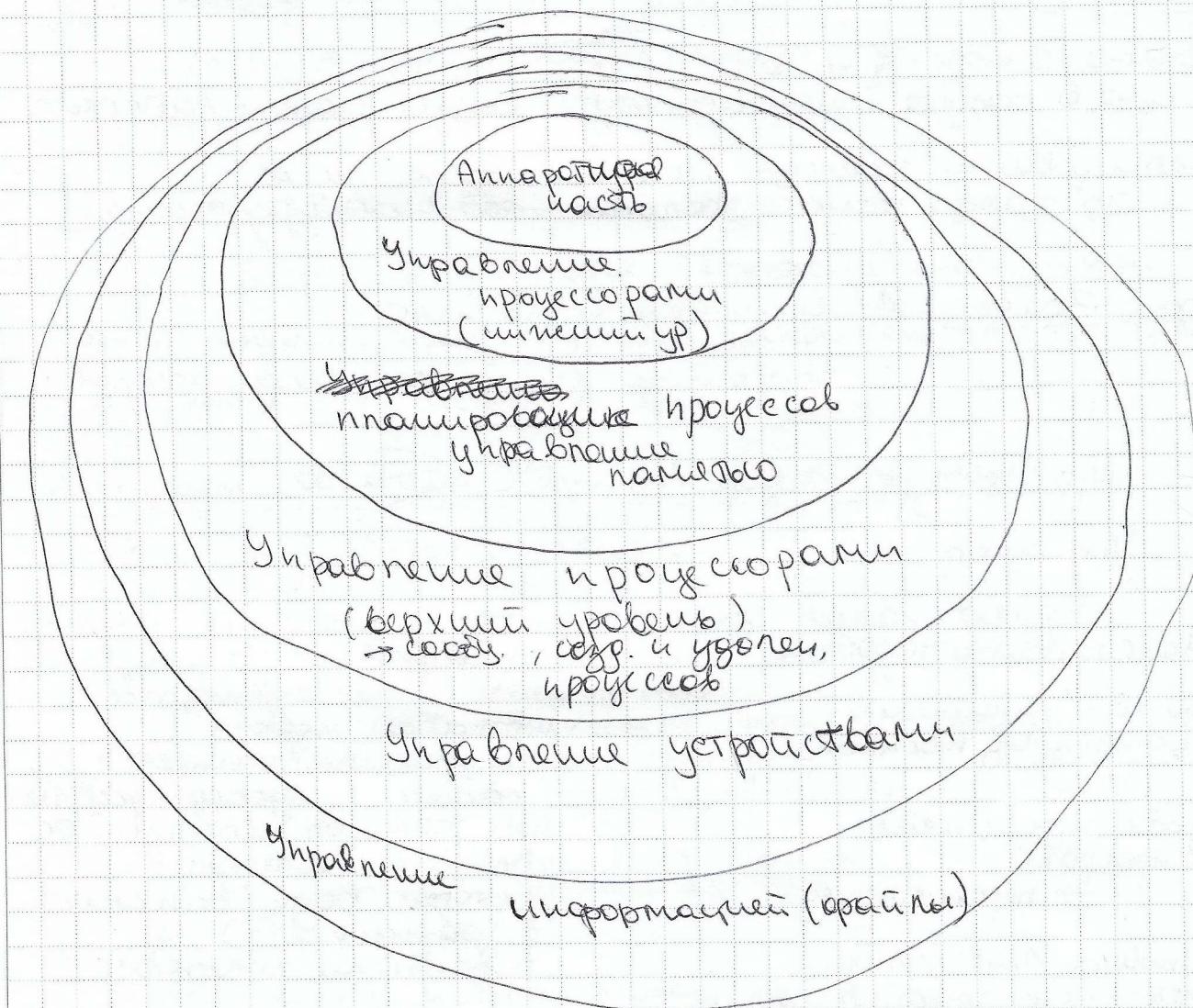
Узел микро-адресной архитектуры

В ср. 60x..70x

Микропр. + Денескан

→ иерархическая машина

→ отр. процес. соотв. функции. ядра по отн. к
отнап. частям нет связи



Несколько интерфейса в виртуальной машине.

Менедж. яр. имеет доступ к физ. интерфейсам (наход. физ.

намедж. физ. струк. конф. виртуальной

вн. программы.
и получ. от
программ

Микроядро

- не управ. (к другим, верхн. подр. только к само само)
- управ. (меняю само само само)
- полууправ. (наст. сам, наст. как)

Иерархическая структура UNIX/BSD 4.4



1. управ. уровень
 2. изменяющее
 3. обратим. агр.
 4. кр. управ - е
- управ.
и аппарат. частно
(драйверы)

Драйверы - NO управ. внешними устройствами

Файл. агр. → mapping → преобр. верт. агр → призм.

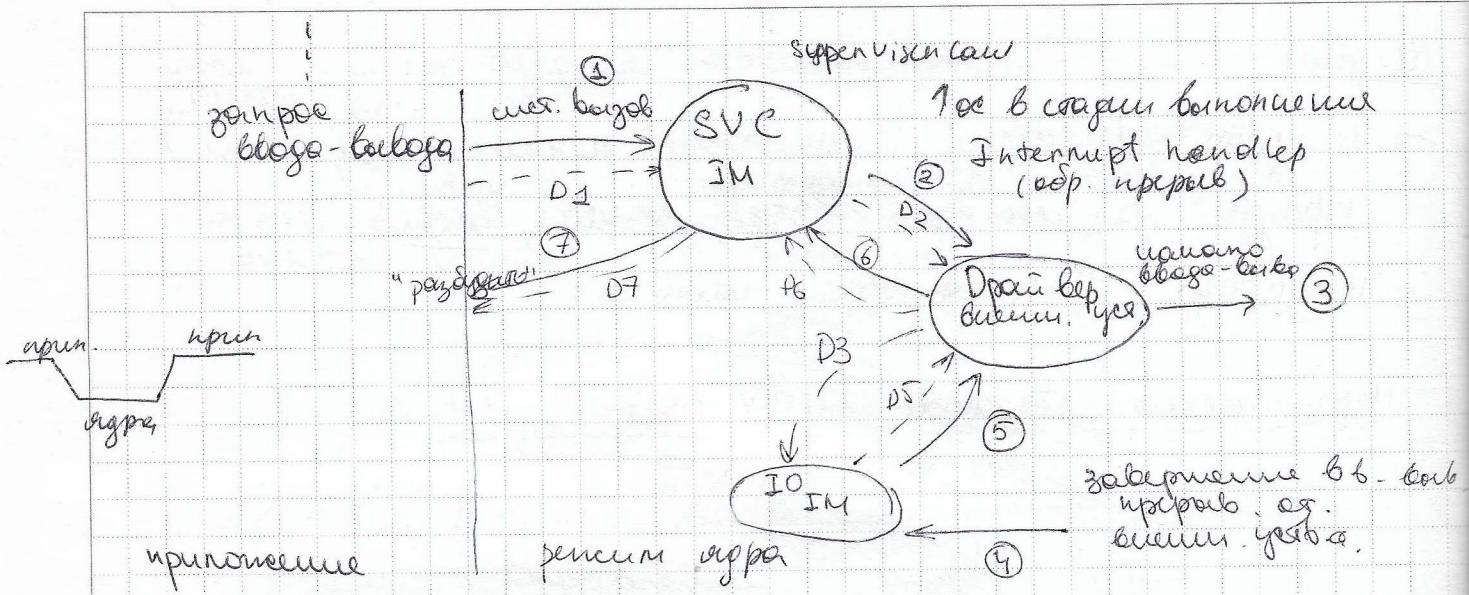
Терминал - сог. наименование UNIX

Микроядро = собственное только самое низк. управ. ядро.

Блокнот → отдельные процессы

Без машин. ядро ← наст. на управление

↙ May наим. прер.



рассмотрим вспом. устр. управл. контроллером / канал
с 3-го покол. ЭВМ
→ расшир.

Общ. прерыв. ← брос. в сост. ожид. в обн. блоки броса в гр.
ожид. → блоки содержат один обр.

IO Mapping
— использ. портов своего сервера

Блок общ. сервера

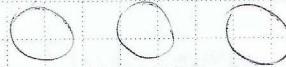
МикроЭн.

приложение



серверы ОС // прип. ОС

клиент
— сервер



Микро
Эндо

регистр
адреса

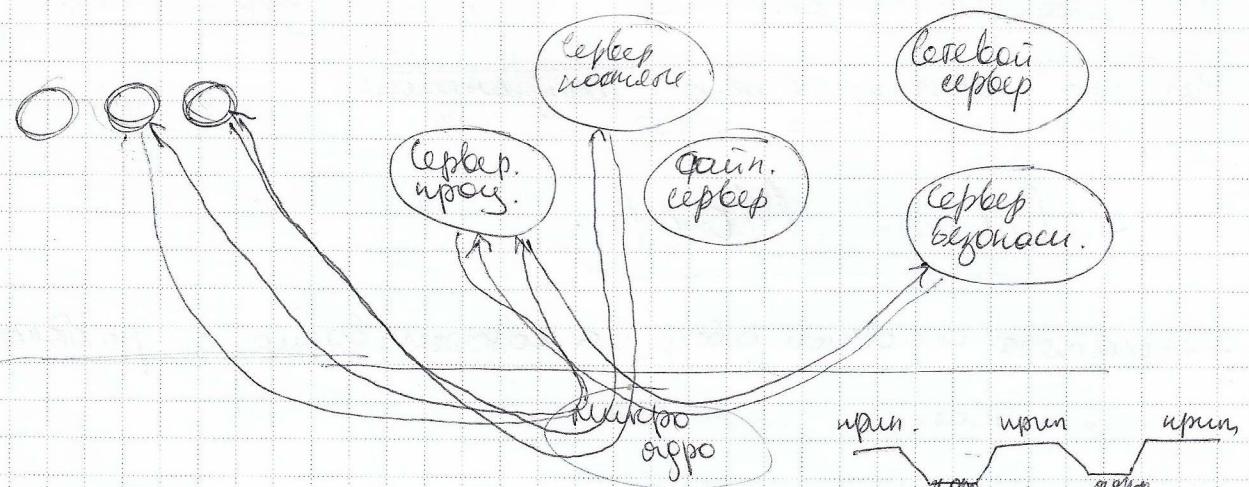
Windows - микр. ожид.

Linux - загруз. механизмы

клиент - сервер

— приложн. обр. к серв. с загл. → сервер. обр. → форм. ответ

Программ. - канал.



одинак. поиски
и приема сооб.

→ получение быть извещением
подтвержд. сообу. срр. и прием.

см. диагр. 3 со сл. блок. прослед. при перср. сообу.

связано с ядром. микр. ядром. архит.

Преимущ. микр. ядром архитектуры
помимо.

的优点. это. выгода
за счет переключ.

→ из задач → ядро
→ из ядра → задачи.

Mach → первое микроядро. архит.

Привлекательность

- имеет быть извещена без переключ. ядра

Две копии. одна из микроядра Mach является быть некро
в части.

один. поискт.

недорогая сетей и упр. настройки
в ком. сис. Mach → рискун утра

Примеч.

- сист. постр. но прием. микроядра - централь.
ком. задачи здравы

Недост. в сист. план. временн.

Например QNX ← на основе микр. ядр.

POSIX опр. план. врем в ОС

- 1003.1 стандарт - реали. врем в ОС - чтобы ОС облегч.

траб. уров. сервиса за опр. врем.

Задача ядр. времени
- время и спонс

→ упр. врем. прос или объект

Мы имеем систему сценария назначения

ОС Mach

- определяет основное API на которых базир. ее процессы

- процесс
- thread / worker
- объект памяти
- порт
- сообщение

Особенность

- объект памяти (Memory Object)

структура данных которые может быть скоп. в агр. ит. и могут занимать одну или несколько стр.
и авт. основой при вирт. пам.

когда процесс обращается
к объекту вирт. → стр. кратив

друго обр. прерывание

друго Mach где ход. осущ. стр. посыпает сообу.
серверу принципа повторов
допока не все все это занесе обработает.
сост. стр. будет занес. в ОН

Чем прерв. вспомог.

основано на передаче сообу.

две это что все перво. сообу.

прац. ненужн. просит сооб. залу. идет. др.
→ порт

сооб. в АН

и спасает поддерж. очередь сообу.

Как передать очередь и не опр. разные

знач. не может ⇒ процесс блок. до тех пор

когда не будет разрешен.

Разные виды портов:

- порт проксера (для вспомог. с ядром)

- порт загрузки (при старте системы)

именно порт ядра

- однок. базисн.
сервисы сис.

- порт. осн. функция (передача сообщ. об ошибках и просесс.)
- зарезерв. порты (using newer. брауз. просессоров со стандарт. сет. серверами)