

СС 2

Дружина Елена Владимировна  
elen-vd17@mail.ru

организация обмена по прерываниям  
в микропроцессорах Intel x86 и  
ищу: - история Е.В.

USB-порт: способ обмена (структура)

MySQL. High Performance (курс Бжоррррр)

Салаева, 43502/4.

Реализация процессов в UNIX-системах

отображение в памяти исполняемого  
файла после компиляции (+ свои данные,  
код, в т.ч. библиотек, структуры данных  
ядро, набор для упр-о процессами)

Дерево процессов включает parents  
and sons

parent - при загрузке ОС (init)

son - создается программистом

8 Инфраструктура процесса

структура proc - запись системной  
таблицы процессов, в ядре 034.

Запись для выполняющего  
процесса x-я системная  
переменная struct proc.

Переименование процесса - то  
изменение sys-proc.

В дескрипторе proc хранится информация

достаточно для разрешения всего  
поискового процесса.



\* выносятся раздельно на диск

и - area - кодовая часть структуры

а) заголовок исполняемого файла  
(4 - exdata)

б) n-cdir - текущий каталог

в) n-dir - путь к каталогу

г) двоичный сигнал (n-signal) -  
указание на то, что реакция  
с сигналами летящими  
взаимодействий (на реакцию),  
уверенный.

д) регистровой (аппаратной) кодовой  
(n-uid)

е) системный стек

+ переменные окружения.

Стек ядра - стек фиксированного  
размера, внутри и - area

Используется и - area многими  
системами ядра



Инфа, хранящаяся в табличке `sysproc` `new struct`.

char p\_stat  
char p\_pri  
unsigned int p\_flag  
unsigned short p\_uid  
unsigned short p\_suid  
int p\_sid (number process or вычисляется по какому session принадлежит)  
short p\_ppid  
short p\_pid  
short p\_ppid  
signed int p\_sig  
unsigned int p\_size  
time\_t p\_utime  
time\_t p\_stime  
char\* p\_fname  
struct p\_region & p\_region  
short p\_vstat  
unsigned p\_utel[] - массив функций tracing for u-area

100 - количество записей в таблице

Процессинг континента осуществляется до прерыва процесса в user mode (тогда записаны все данные системы)

Остановка процесса

вызван сбросом (системный вызов `exit()`)

итоговое завершение

по сигналу от системы (принудительное)

освобождение ресурсов процесс, кроме файла возврата и статистики выполнения с сохранением структуры данных → kernel в соед. zombie parent выключает `sig. block mask` `child & process & m...`

Создание процесса зависит от ядра: 1) прерывание процесса системным вызовом `fork()`

2) загрузка программы, привязанная к процессу, не 1-й раз ресурс

Например, идентификатор функции, структуры, диспозиция сигналов в обработчике, ограничено на процесс, текущий корневой каталог, маска созданных файлов, управляющий термиднал.

Виртуальная память не описана в привязан. режиме. не не стем, что и у родителя.

После возврата из вызова `fork()` значение переходит из процесса родителя в

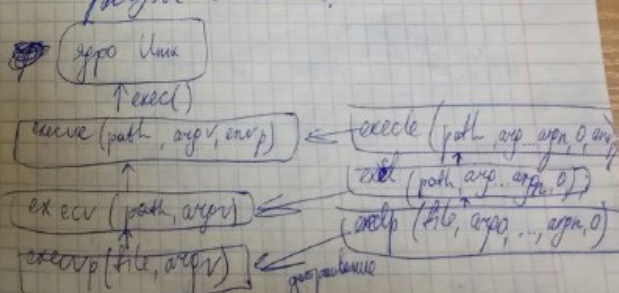
процесс `parent`

`parent returns 0`  
`son returns ppid`

Влияние родителя на потомка:

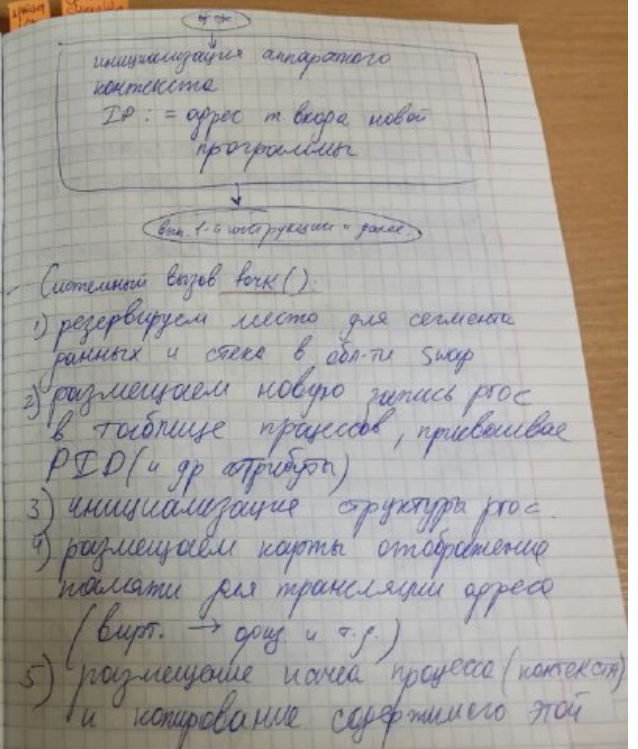
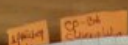
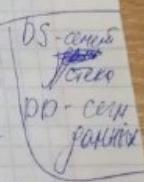
- 1) идентификация
- 2) обнуление сигналов поставки
- 3) son получает копию u-area от parent на первое время.
- 4) список файловых дескрипторов при `fork` не файловый таблиц.

Временная статистика обнуляется, атрибут остается.





CP-Bd  
Stereoplate





- области с ресурсами
- а) создаются соответствующие объекты процесса: часть оборудования с ресурсами
  - б) реализация аппаратного уровня: также контролируется с процессом
  - в) установка в ОВ возвращаемого значения: факт, вызов, результат РЭО по ~~ка~~ <sup>ка</sup>
  - г) процесс помещается как "готовый к запуску" и реализуется в операционной системе "готовый к выполнению"

### Объектная модель

Все ресурсы системы рассматриваются как объекты. Это означает, что объект имеет атрибуты и пр.

⊕ Все ресурсы - каталоги, структуры, с типами атрибутов.

### Процессы в Windows NT

реализуются как объекты, с ресурсами, подготовкой объектов, сервисов. В адресном пространстве процесс может выполняться. Несколько потоков.

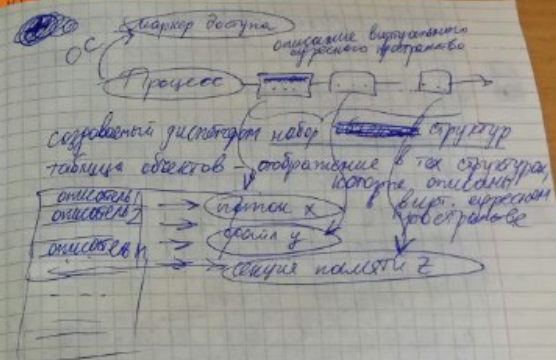
Процесс, объект и поток имеют встроенные средства синхронизации. Диспетчер процессов не предоставляет методику создания объектов процессами отпавления типа процесс-поток.

\* Все это характерно для базового ядра NT. Процесс NT, который достраивается к существующей системе - исполняемое программное обеспечение с закрытым набором памяти, системными ресурсами (соединения, порты, командная строка, драйверы).

### Поток исполнения - внутрипроцессная

структура, направленная на выполнение. Структура прав, реализуемая за счет средств виртуальной памяти. Процесс получает логич. образ памяти, несоответствующий с фактом  $\approx 4/5$

- Процесс изолирован друг от друга.
- Пользовательские приют-е ~~не~~ обращаются к защищенной системе/серверу.
- ~~Диспетчер~~ <sup>Диспетчер</sup> реализующий др-ч системы (APR).
- упрощается базовая ОС.
- ⊕: появляется расширяемость и др. защита виртуального пространства, другие механизмы и приемы.



Диспетчер объектов, процессов и вирт. памяти работают раздельно.

Диспетчер вирт. памяти - поддерживает в памяти и записи описателей.

Диспетчер объектов отвечает за создание/уничтожение процессов.

Маркер доступа - один из описателей таблицы, который реализует доступ к данным.



Удобно открыть маркер доступа  
нужно сделать большим правым

### Доп. ресурсы:

- приоритет
- процессорное время на каких  
ядрах процессора выполняются  
потомки процессов)

У-б объект имеет:

1) заголовок (сод. инт. обьекта),  
куда входят имя и  
адрес обьекта, дескриптор  
защиты обьекта.

2) атрибуты - тело обьекта

3) сервисы тела обьекта

Именно диспетчер процессов  
определяет атрибуты процесса и обьекта

34 Возможные сервисы.

Сервисы:

- 1) создание, открытие процесса, запросы  
о нем интроду, завершение,  
работа с виртуальной  
памятью.

Маркер доступа - обьект исполняемой  
системы, содержащий интроду  
о привах зарегистрированного  
в системе пользователя.

~~Размер и атрибуты~~

Размер и атрибуты -  
Максимальный обьем резервированной  
и непрерывной памяти, размер  
обьекта (порядки ~~сравнов~~  
процессорное + выделяемое  
памятью на выполнение.

- ~~текущий~~ - общее + выполнение  
всех потоков процесса.
- Счетчик ввода-вывода -  
переменные, в которых записываются  
число и тип операций  
ввода-вывода для каждого  
из типов процесса.

- Сетчик адресов виртуальной  
исполняемой памяти - канал, по  
которому диспетчер процесса  
посылает сообщения если  
есть из потока ~~виртуальной~~  
исключения (прерывания).

Процессорные прерывания запрещены!

Сервисы

- Текущий процесс - получение описания  
завершение - завершение открытия  
описателей, уничтожение  
вир. адресного пространства

Поток - набор ресурсов, принадлежность  
того, кем обьект ~~назначен~~ потоку.

Состав потока

- 1) идентификатор имени
- 2) содержимое набора регистров
- 3) 2 обьекта: Kernel mode  
User mode
- 4) собств. область памяти для  
использования порошителями  
(#IDE)

П. 2-4 - это контекст потока.

Фактически данные зависят  
от х-к системы.



## Общие у потоков процессы:

- адресное пространство
- маркер доступа
- базовый приоритет
- описатели объектов таблицы вирт. пространства

Дальнейшее развитие гипотетически идет на базовых -

ресурсах базовых:

- + стек + полнота ресурсов регистров
- + данные при создании базовых

Предоставление потока на объект:

- заголовки
- атрибуты (идентификаторы)
- сервисы
  - контент потока
  - набор значений регистра
  - динамический приоритет (на текущий момент)
  - базовый приоритет минимален

переходящий процесс

процессорное время

выполнения потока - общее

время в режиме ядра и пользовательского режима

отсутствие обслуживания для обработки асинхронного базового (APC) процедур (за системных вызовов)

система приостановки выполнения потоков без возобновления исполнения

маркер имперсонации - временный маркер доступа, который используется для выполнения операций от имени другого пользователя

завершение - канал коммуникации, по которому передается инв-я по завершению потока

код завершения потока

сервисы

создать поток

запросить инв-я о потоке

запросить / установить инв-я

остановить поток

возобновить поток

и др.

Различные упр-е процессами в разных средах

- наличие поддержки многопоточности
- отношения между процессами (иерархичность построения процесса, способность функционировать по принципу "родитель-потомок")
- система наследования - способ передачи информации между процессами
- наличие сессии процесса обслуживания по группам признаков
- правила создания новых процессов и формат исполнения со своей семантикой, исполнения

Семантика создания процессов в разных средах

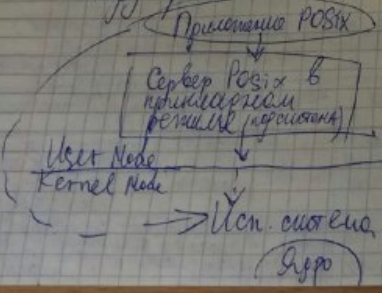
Ф-и	Win 32/64	POSIX	OS/2	Базовый процесс
API	CreateProcess()	fork()	Dos Exec Pgm	NT CreateThread()
переход процесс	нет семантики отношения "родитель-потомок"	иерархичность "родитель-потомок"	процесс-16 "потомок"	создается новый процесс, который не от базового, а от базового процесса, который не от базового, а от базового
последующие	наследуются от базового процесса	наследуются от базового процесса	наследуются от базового процесса	наследуются от базового процесса



инициализация адресного пространства	процесс инициализации адресного пространства	процесс инициализации адресного пространства	процесс инициализации адресного пространства
инициализация пространства памяти	инициализация пространства памяти	инициализация пространства памяти	инициализация пространства памяти
память	память	память	память

### Система

На протяжении всего времени о  
множестве серверов, процессах и  
их отношениях  $\odot$  действует соглашение  
между процессами



Каждое серверное приложение в Windows  
в привилегированном режиме системных  
сервисов предоставляет 2 метода  
реализации прил.-я:

- 1) Обеспечение использования сегментов
- 2) С загрузкой памяти прил.-я

(аналогично микроверсии системы)  
Прил.-я формирует запрос на загрузку  
параметров прил.-я в  
загрузчике  $\rightarrow$  вызов  $\rightarrow$  ядро (LPC)  $\rightarrow$   
 $\rightarrow$  процесс на защищенном (сервер)  $\rightarrow$   
 $\rightarrow$  доставка сообщения и его  
обработка через исполняющую  
подсистему

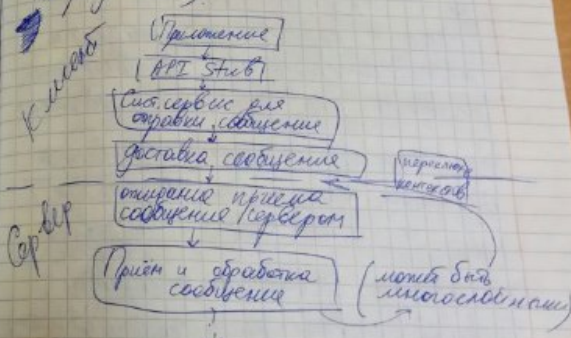
\* Связывание прил.-я с API происходит  
через загрузчик (точка входа)  
Вызовы раз-тов происходят тем же путем.

### ① Модель с OLL загрузкой:

- 1) защита модальных структур данных  
серверов и прил.-я
- 2) абстракция серверов (со своей  
структурой данных, системной  
процессов)
- 3) сервера - прил.-я на уровне ядра  
режима не могут вызывать  
внутренние др.-и ОС напрямую,  
а только через ядро
- 4) Единовременный доступ и ядро  
вызов системных сервисов  
(механизм загрузки)
- 5) Клиент - сервера обеспечивают  
разделение ядра и исп. системы
- 6) Возможность привлекать  
работу с сервером с помощью  
средств

### ② Модель с OLL загрузкой

- 1) обеспечение производительности за  
счет промежуточного слоя и  
его реализации механизмов  
LPC (сообщение всегда через  
посредника)



Для получения сообщения и его  
обработки требуется переключить контекст  
(клиент сервер), что "нежелательно".



- Для приложения клиента требуется:
- 1) скрывать клиентский код
  - 2) выбрать поток сервера (или оба записки планировщик)
  - 3) адреса текущего контекста на серверной.
  - 4) фр-я API выполняется
  - 5) скар-е серверного контекста
  - 6) переключение на клиентский контекст для обработки сообщения.

Переключение контекстов осуществляется:

- 1) аппаратное переключение (обязательно!) для получения нового регистрового контекста.
- 2) числом обращений к серверу.
- 3) LRC (система передачи сообщений между процессами)
  - доставка коротких сообщений
  - быстрое (для нативной Win32)

Скрытие чужих обращений к серверу возможно:

- 1) реализация фр-я API на в DLL на стороне клиента, что возможно только для тех API, кот. не обр-т и мод. данных.
- 2) хранение данных сервера в ядре или их интр. сист. или их кэширование в DLL с клиентской стороны.
- 3) пакетирование клиентских запросов API их отправка серверу единым сообщением.

# мод. данных: - число окон на экране Windows  
- таблицы трансляции списков  
- способ. управления процессами и т.д.

- Фр-я фр-я: вставки интр-а с сервера  
потом процесс.
- модификация х-я процесс/потока
- # фр-я No light-версия (с точки зрения ресурсов)
- создание процесса
  - открытие файла

Раздел не передает с клиентом 4  
передачи сообщения меньше чем  
на само создание процессов.

- 2) ~~Данные сервера~~  
# данные сервера, требующиеся клиенту  
- ID устройств (наименование объектов)
- 3) мало обращений.

### LRC

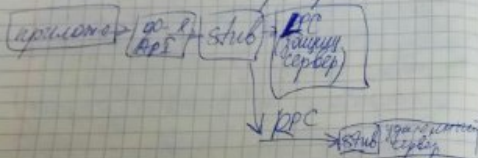
- 1) Описание потока одного процесса (ссылка на объект, адрес, адрес)
- 2) Поток привязывается к процессу (доступ между потоками по данным осуществляется с использованием общей памяти (разделенной), копирование из одного в другой)
- 3) Реализация клиент-серверной передачи данных.

### LRC

- 1) доставка сообщения в объект-порт, связанный с серверным процессом
- 2) Доставка в порт сервера данных на сообщение, а передача сообщения через объектную используемую память.
- 3) передача отп. потоку сервера через свободную область совместно используемой памяти.



Вот три способа передачи данных  
установка канала связи между  
клиентом и сервером.



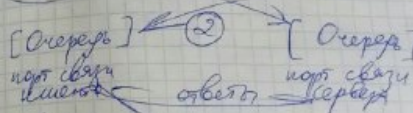
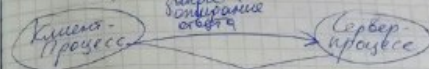
Ф. в порта соединения приел  
запрос клиента на установление  
канала связи с сервером.



Открытие объектов портов не  
последуется.

Механизм

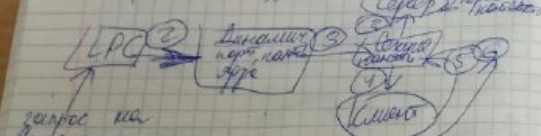
1) Контроль в порте запрос отправляется  
в очередь



- а) LPC помещает сообщение в  
один из объектов портов
- б) уведомление контактов клиент-сервер
- в) переключение контактов ядра
- г) поток сервера копирует сообщение  
в адресное пространство и обрабатывает
- д) результат идет в порт связи клиента.

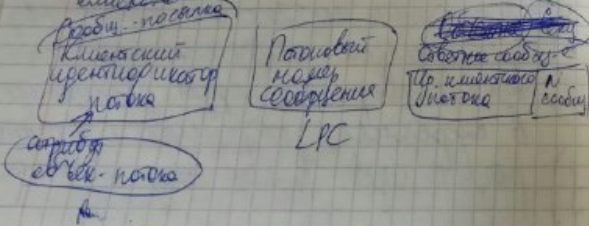
\* Если объект порта находится в  
мемории клиента.

LPC берет данные из памяти  
→ размер блока ограничен  
(256 байт)



запрос на  
выделение  
объекта

LPC не сразу формирует объект связи.  
Все управление полностью на  
клиенте



## LPC (Linux)

1) передача данных  
2) управление состоянием  
3) управление ресурсами  
4) управление процессами

### Типы LPC

- 1) Сигналы
- 2) Каналы (использование / неиспользование  
протокольных объектов, адресов  
удаленных объектов)

FIFO - дисциплина обслуживания, клиент  
имеет в отличие от PIPE  
доступ из независимых  
процессов (Можно быть открыт  
для чтения и записи)

### Базовые правила

- 1) При чтении меньшей порции, чем  
необходимо перерыв, порция  
остается в канале для  
предупреждающего оповещения  
в канале.
- 2) В противном случае возвращает  
число байт с учетом буферизации.



3) Если канал занят (не открыт или с ним каналом не записан), сервером в базе. Если канал открыт, канал будет заблокирован до появления данных.

4) Если процесс одновременно пишущий в канал, их порывы не перемешиваются.

### Особенности работы

5) При записи в базу → разн. оборудования, вызов write блокирует до освобождения памяти.

### Средства сервера

- 1) сервер FIFO (память)
- 2) Открыт на чтение
- 3) проточность ⇒ среда
- 4) закрыт FIFO

### Клиент:

1) открытие для записи.

2) записывание.

3) закрывает → удаляет.

Пространство имен - ил во вдумчивых имен объектов типа IPC для идентификации.

Требуются трансляция имен, сегменты, разделение памяти, многоимен (FIFO) - генерация имен, связь из имени файла и идентификатора процесса.

Пространство имен позволяет создавать и одновременно использовать IPC независимых процессов.

Каждый IPC работает с объектом использует zero.

СИ, пара 12

### Управление (ввод/вывод)

1) контролируемая часть (обслуживание со стороны)

2) функциональная часть (функциональный уровень)

- входы каналов
- принтеры
- мониторы
- сканеры
- адаптеры

### Работа контроллера

- 1) запрос на предоставление в приложении
- 2) его адресирование ОС
- 3) отправка в оборудование
- 4) исполнение программы на стороне клиента.

Для ПО ввода-вывода обслуживаются приложения.

- 1) экранирование от особенностей периферии (аппаратуры) и устройств ввода/вывода
- 2) единоеобразие именования / отображение по ОС и программисту

3) обработка ошибок как логич. ошибки аппаратуры (ошибка чтения ж. диска).

4) обработка ошибок преобразования на низшем уровне.

5) принятие решения о синхронности/асинхронности передачи и необходимости блокировки запроса на ввод-вывод.

### Способы обмена:

- (- директно (накопитель))
- (- байты (массив, строка))
- (- USB-порт)

Прямой  
но прерыванием  
спрос готовности

\* базовый способ обмена - по прерыванию.

Контроль прерывания не обязательно эл.т. Иногда посредники нет, а используется системная шина.



## Функции

разделение  
разные процессы  
могут обращаться  
к одному устройству  
квантами.

# устройство памяти

## Выводы

атомарность обра-  
щения процесса к  
ресурсу.

# драйвер

Для байтовых устройств можно гра-  
низировать разделение доступа  
постройкой семафоризированных данных.

Иерархия ввода-вывода.

I-й слой: в случае асинхронной  
обработки с периферией  
(предполагается по умолчанию)

**ISR** Обработка прерываний - микроядро  
ПО, присутствует почти во всех системах

## Включает:

- свой первичный обработчик  
прерываний
- векторный способ обработки  
таблицу векторов

Первичный обработчик прерываний  
зависит от спец. принципа:

• ДЗ провр. микр. по коду  
(сметание времени записи контроллера)

• обращение к портам  
Большинство ОС не позволяет писать  
собственной обработкой

Осн. назначение TSR: снятие  
блокировки с процесса - зависимость  
обмена (При обработке прерываний)

II-й слой осуществляется рефлексирование  
Драйвер устройств - код (зависимый),  
синхронизирующий работу устройств.

Для систем общего назначения есть  
обобщенные классы драйверов для  
разных классов устройств.

1) Драйверы вызывают критичность запрос  
к устройству программного слоя

2) решают, как его выполнять

3) выполнение осуществляется  
драйвером, если драйвер свободен,  
в противном случае - организуют  
очередь.

1) Преобразование из абстракт-  
ной формы в конкретную.

2) передача команды контроллеру

3) принятие решения о  
необходимости блокировки до  
начала операции.

(если операция длительна, то  
до ее окончания драйвер  
блокируется, в противном случае  
без блокировки)

III-й слой: независим от устройства  
реализации набора до-з. имеющих  
общий характер, могут быть  
универсально применимы для  
классов устройств.

# обеспечение общего интерфейса  
к драйверам устройств,  
их инициализация и защита.

Ведение базовых карт - тоже др-то независимого  
слоя.

IV-й слой: пользовательский  
(отвечает за представление  
программной среды API,  
базисных др-и для формирования  
запросов).

⊕ Система Спорушила - способ  
работы с введенными устройствами  
в мультипрограммной среде.



В системе создается процесс с именем, присвоенным на устройство, спец. копией из которого процесс <sup>планирует</sup> отправляет запросы.

Система ввода/вывода Windows

исполнительная система NT.

- есть виртуальный драйвер - источник для внешнего устройства. Направляется идет в драйвер.

- ОС транслирует описанным этого спец. драйверу все необходимые и приемники предоставляются как драйверы.

+ потоки пользовательского режима встраиваются базовые сервисы драйвера системы.

- драйверы реализуются преобразуют запросы и виртуальные драйверы.

- Каждый запрос управляется потоками  
- Драйвер ввода/вывода управляет доставкой запроса ввода/вывода к ОС и драйверами устройств

НО не управляет самим вводом/выводом.

1) реализует общие процедуры для различных драйверов

2) функции универсальности системы ввода/вывода

3) создание потока

4) управление драйверу

5) удаление

Универсальные процедуры:

1) а-и ввода драйверов другими драйверами

2) поддержка потоков ввода/вывода

3) сервис пользователя режима

4) средства API ввода-вывода

5) управление буферами

6) предоставление информации о установленных драйверах

Универсальная модель драйверов: структура и механизм работы драйверов и устройства ввода/вывода

Вид

✓ Драйвер должен быть написан на языке высокого уровня

✓ Структура пакетов многоуровневая

✓ обеспечение стандартизации

✓ драйвер может быть

выяснен потоком более

высокого приоритета

✓ драйверы должны корректно

восстанавливаться после

сбоев и передавать

прерываемые операции ввода/вывода

ввода

Драйверы имеют унифицированный интерфейс

драйверы

однослойные

байт-ориентированные

многоуровневые

байт-ориентированные

39 большого объема

Синхронный режим - более прост.

Асинхронный - более рациональное использование ресурсов.

Сложность синхронизации в многоуровневых драйверах и устройствах ввода/вывода и передающих устройствах и определяется возникающими ситуациями

Каждый процесс обрабатывает различные длинные ситуации на своем канале


стр. 3 страницы  
после 12 страниц



Задача превращения - один из способов  
разрешения противоречия сечения.  
В прикладных работах используются  
различные типы разрешения наиболее трудноразрешимых.

- Бинаризуемое перемножение - форма бинари-  
зации: ~~свободен~~ или занят  
ресурс Минус: все время  
требуется ее считывать.
- Бинаризация Рентона - для спаривания надр-  
работ по бинаризации (ф-и  $P \cdot V$ , где  
 $V$  - время  $S$  увеличивается на 1 ф-и  
перемножения действия  
(свободно, занято, заня-  
то, занято); эти ресурсы  
не могут быть преобразованы.
- $P(S)$  -  $\downarrow S$  на 1, если это возможно.  
(ф-я с условием)  
if  $S \neq 0$ , то уменьшив его  
на 1 и операции прост  
возможности этого уменьшения.

У нас проверка и уменьшение  
надежная опрашивание

Желтые шпорицы. перелетного у  
Фейскарт - это селадитро. 

Проблема функций не имеет решения 3-й раз  
предоставление функций

Средства синхронизации

Локальные

2. распределенное

для выявления  
источников  
источников или  
руки  
протекания  
процессов

не парадоксально  
иногда

- микроинструменты
- объектив, рабочее поле
- световые системы
- микроанализатор
- микротом

IPC)

- микроизучение самого
- служба времени
- ~~• микроизучение~~
- исследование содержания

IPC (Pipe)

Силами сердца зависит от "мудра"   
 "сознания"

→ потоки → сильнейшее состояние при завершенном процессе

- процесс по завершении работы
- файл при окончании всех работ

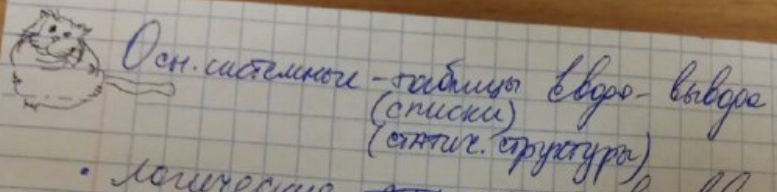
Хронология событий - инвариант, в котором  
каждому элементу можно сопоставить  
его целостную структуру.

Проработав с хвостом одной крылатой самурайской,  
уже не раз решен вопрос разделения,  
справа является нераспространенным  
мощным.

Thread Safe обеспечивает взаимное  
исключение.

- Анкарастые прищипывы браширования
- Мелкие / Милые Блокноты





- ири-го-ч  
о тергуну  
уаредот  
и напаре  
(IRP)

034/64 ~~unsub. & charged~~  
Exhibit ~~unsub.~~

Система вычисления времени  
наблюдения  
фактически  
сегментов  
границы  
сегментов границы

1) разламывание дайлов в 34 разного  
типа (сперматозой, 200 гр)  
2) перемешивание разного моту 34  
3) распределение вып. адреса в 34  
# свитчи, семантики, структуры.

всё время. программа решается по  
страницу как правило, даме.

размер, ~~не превышает~~ ~~в~~ ~~сво~~  
иных странах, недолговечной в  
7 размер → удорожки → плохо.

\* для загрузки процесс чаша гранич  
идет в операцию, а чаша на  
диск - во вкислему.

• В таблице страниц содержится информация  
о том, какие страницы нужны и какие бывают  
ли составлены, что экономит ресурсы

- принцип невзаимности  
(защита не возмущается  
применяя с целью защиты  
информации)

- признак частоты обращения  
сердца к другим



Сторонние производители могут быть вынуждены возмездно финансировать (при отсутствии натурной выплаты сторонами) предоставление помощи в капиталовложениях по развитию персональной страницы в электронной почте на основе странички ее держателя.

Механизм предоставления выплат другим

Механизм преобразования вихрь срыва

Виде сега им е страничен разрез и е изработен в виде парог изпитателна камера за метал + електрич.

2) числовые граничные условия  
( $0, 2^k + \text{разности}$ )

3) при камере обращении к оперативной памяти на основании уникального адреса таблицы номеров байт

8. скрививши своє розум (внутрішній)

4) длина записи в таблице  
Франсуа (вспомогательная),  
определяется следующим.

4) у той записи місцевого іменного  
номер фізич. особи

5) к камере арест. ар. ч. 10  
присоединяется помещение, со склада  
материал разгрузки ввр.  
аррест.  
на

Т.И. Яценко 2-й, помет  
вместо имени

использовать как конспект, что позволит проводить занятия

② Сегментное распределение  
при делении в все внут. окруж.  
пространство, как правило делится  
на равные сегменты ~~по длине~~  
для дифференцированного доступа

к элементу ~~того~~ разного типа  
содержащиеся (кар, бочки и др.)  
теперь ты видишь, на который  
идет процесс, и есть сред-  
не представляя собой нечто  
универсальное явление.

3) Коллектив - старинное

Темные, склеивающиеся фракции, подвергнутся  
в процессе склеивания, и будут  
использованы как фракция 054,  
также как и ранее, без которого продукт  
применяется для склеивания стержней и  
фракции распределения, в котором  
сфера имеет фракцию сферическую  
для всех сегментов прочая.

Д. Свечков

Весь процесс можно изобразить так:  
всплыв. кристалл. состояние, обмен  
пределами границами

2 способа распределения вир. сегмента  
1) помещение сегмента в общую часть виртуального адресного пространства

2)