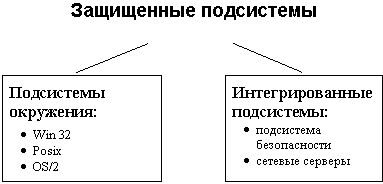
**Множественные прикладные среды Windows NT**

Виктор Олифер



Защищенные подсистемы взаимодействуют путем передачи сообщений, используя механизм LPC

LPC - Local Procedure Call - вызов локальных процедур

Цели подсистем окружения:

Обеспечить несколько программных интерфейсов

Экранировать базовую ОС от изменений API

Объединить общие для всех API глобальные данные, уменьшая избыточность

Обеспечить возможность включения новых API

Защита окружений от приложений

Создание процесса

Множественные прикладные среды обеспечивают совместимость на ДВОИЧНОМ уровне

Цели:

возможность выполнять программы, написанные для других ОС и процессоров

встроенность средств обеспечения совместимости в ОС

приемлемая скорость выполнения приложений

Примеры ОС, содержащих встроенные средства обеспечения множественных прикладных сред:

OS/2 2.x

Workplace OS

Windows NT

PowerOpen

некоторые версии UNIX

Реализация Windows NT на базе API OS/2

Пример различия в системных вызовах:

|  |  |
| --- | --- |
| fork()   Наследует адресное пространство родителя   Имеет одну нить   При завершении потомка нужно послать сигнал родителю | DosExecPgm()   Адресное пространство создается заново на основе файла prog.exe   Имеет несколько нитей   При завершении потомка созданного с опцией EXEC\_SYNC идентификатор процесса нельзя повторно использовать |

Реализация Windows NT на базе двух равноправных API

Цели разработки микроядра Mach

Обеспечение базовых функций для создания других операционных систем (например, UNIX)

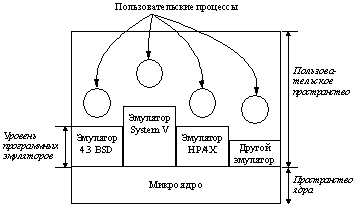
Поддержка больших разряженных адресных пространств

Обеспечение прозрачного доступа к сетевым ресурсам

Поддержка параллелизма как в системе, так и в приложениях

Обеспечение переносимости Mach на различные типы компьютеров

Абстрактная модель эмуляции UNIX на основе Mach



Функции микроядра Mach:

управление процессами,

управление памятью,

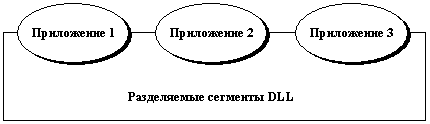
коммуникации

функции ввода-вывода

Функции управления файлами, каталогами и другие традиционные для операционных систем функции выполняются в пользовательском пространстве эмуляторами различных ОС

Функционирование защищенных систем Windows NT

Модель API на основе DLL



Модель API Windows NT

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Системные сервисы | | | | | |
| Менеджер объектов | Монитор ссылокбезопасности | Менеджер процессов | Средство вызова локальных процедур | Менеджер виртуальной памяти | Менеджер ввода-вывода |
| Ядро | | | | |  |

Два способа вызова системных функций

Обращение к системным сервисам в традиционных ОС

аппаратное прерывание

процессор переходит в режим ядра

выполняется требуемая системная функция

возврат процессора в пользовательский режим

выполнение нити с прерванного места

Вызов системной функции (API Win32) в Windows NT

Динамическая библиотека DLL Win32 обращается к системному сервису NT с просьбой послать сообщение серверу, выполняющему требуемую функцию

Сервис посылает сообщение и ждет ответ

Сервер получает сообщение, выполняет функцию и отсылает ответ

NT-executive выполняет следующую последовательность действий:

сохраняет контекст клиентской нити

выбирает серверную нить для выполнения и загружает ее контекст

выполняет функцию API Win32, используя серверную нить

сохраняет контекст серверной нити

загружает контекст клиентской нити и обрабатывает результаты выполнения функции API

Оптимизация

некоторые функции API реализованы внутри библиотеки заглушек

некоторые данные Win32 хранятся в адресном пространстве NT-executive

запросы приложений на выполнение функций API объединяются в пакеты

Типичные функции, которые подсистемы окружения должны выполнять по управлению "своими" процессами и нитями

Создание и завершение процессов и нитей

Регистрация и управление взаимоотношениями между процессами

Чтение, запись и другие действия с адресными пространствами процессов - клиентов

Останов нити клиента, изменение пользовательского контекста нити, рестарт этой нити

Захват и обработка исключительных ситуаций (exeptions), генерируемых клиентскими процессами

Взаимодействие подсистем

Логический вход и подсистема безопасности

Инициализация пользовательской сессии

Отличия 32-битного API Win32 от 16-битного Windows API:

использование 32-битной плоской модели памяти

расширенные функции по управлению вводом-выводом, памятью, объектами

поддержка многонитевости, безопасности

улучшены функции по управлению графикой и окнами

Преемственность API Win32

управление окнами и пользовательским интерфейсом из Windows 3.0

пользовательский интерфейс Windows NT полностью совместим с пользовательским интерфейсом Windows 3.1

графическая часть подсистемы Win32 является полностью новой

новое свойство Win32 - безопасность

Подсистема Win32

Подсистемы окружения MS-DOS и 16-битная Windows

Виртуальная DOS-машина

Модель ввода данных для подсистемы окружения WOW

Подсистема WOW

Подсистема OS/2

символьно-ориентированные приложения OS/2 1.х

компьютеры на базе процессоров х86

запуск из командной строки Windows NT, из Program Manager или косвенно из приложений OS/2 или Win32

распознаются по заголовку исполняемого файла

для загрузки приложения - вызов подсистемы OS/2

запускается процесс OS/2SRV подсистемы окружения OS/2

попытки выполнить сегменты ввода-вывода в кольце 2 завершаются кодом "Общий сбой по защите"

Объекты Windows NT встраиваются внутрь объектов OS/2

Нить получает приоритет и идентификатор, которые являются допустимыми в OS/2

Подсистема окружения OS/2 использует возможности большой памяти Windows NT

Подсистема Posix (Portable Operation System Interface based on UNIX)

запуск из консольного текстового окна Windows NT, с помощью File Manager, Program Manager и косвенно из другого приложения POSIX

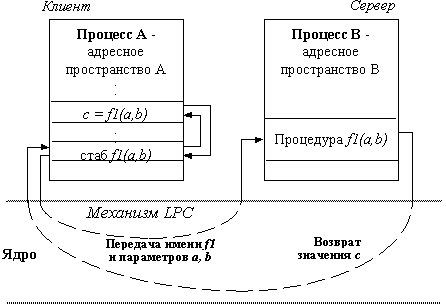
на диске должен находится по крайней мере один раздел NTFS

Подсистема POSIX непосредственно не поддерживает печать

Командный процессор Windows NT поддерживает команды всех подсистем окружения

Механизм вызова локальных процедур (Local Procedure Call, LPC)

Назначение - прозрачный вызов процедур одного процесса из другого процесса внутри одной машины



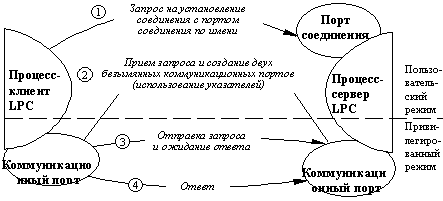
LPC - локальный вариант RPC

Для прикладного программиста совершенно прозрачен

Системный программист оформляет библиотеку стабов LPC и библиотеку функций сервера LPC и регистрирует последнюю в ядре

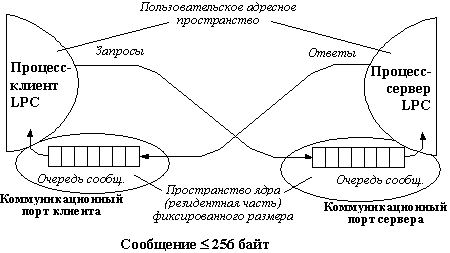
Механизм передачи параметров и результаты в LPC - передача асинхронных сообщений через общую память

Передача сообщений при реализации LPC

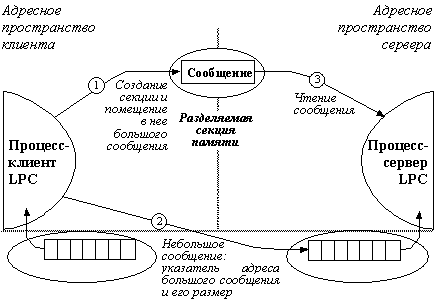


Передача сообщений через коммуникационные порты

Коммуникационные порты - очереди фиксированной длины в виртуальном адресном пространстве ядра.



Передача сообщений через разделяемую секцию памяти



Клиентский стаб сам решает, какого размера сообщения понадобятся для передачи параметров процедуры

Если потребуется сообщение 256 байт, то стаб создает секцию памяти и отображает ее (с помощью менеджера виртуальной памяти) в свое адресное пространство и пространство процесс-сервера