Лекция №10

Управление программами

Программа

Программа – представленная в объективной форме совокупность данных и команд, предназначенных для функционирования ЭВМ и других компьютерных устройств с целью получения определённого результата.

Программа – совокупность файлов, содержащих код (исходные тексты, объектные и исполняемые файлы).

Типы программ

- Системные программы.
- Службы (демоны) – неинтерактивные процессы, выполняемые в фоновом режиме.
- DLL («динамически подключаемая библиотека») – динамическая библиотека, позволяющая многократное использование различными программными приложениями.
- Прикладные программы.

Службы

- Службы – механизм, запускающий в ОС процессы, предоставляющие услуги, не увязанные с интерактивным пользователем.
- Службы часто используются для реализации серверной части клиент-серверных приложений.
- Информация о сервисах (пути к исполняемым файлам, конфигурационные настройки) хранится в реестре по адресу: HKLM\SYSTEM\CurrentControlSet\Services.

Управление службами

- Настройка действий восстановления в случае сбоя в работе службы – например, автоматический повторный запуск службы или перезагрузка компьютера.
- Выполнение службы в контексте безопасности учётной записи пользователя, отличающейся от пользователя, вошедшего в систему, или учётной записи компьютера по умолчанию.

Параметры службы

- Тип службы (выполняется в собственном процессе или совместно с другими службами).
- Местонахождение исполняемого файла.
- Тип запуска.
- Код реагирования на ошибку.

Необязательные параметры:

- экранное имя;
- имя и пароль для запуска;
- момент запуска относительно других сервисов (при автоматическом запуске).

Типы запуска службы

- Низкоуровневые драйверы, например, драйверы дисков, которые загружаются на самом раннем этапе загрузки – загрузке ядра.
- Драйверы, которые загружаются после инициализации ядра ОС.
- Службы, которые должны быть загружены Диспетчером Управления Сервисами (Автоматически).
- Службы, запускаемые Диспетчером Управления Сервисами, только в случае получения явной инструкции на загрузку (Вручную).
- Службы, которые не загружаются (Отключено).

Учётные записи для служб

- Учётная запись локальной системы – выполняются базовые компоненты пользовательского режима; обладает наибольшими возможностями.
- Сетевая служба – предназначена для служб, которым нужно аутентифицироваться в сети по учётной записи компьютера, но не требуется членство в административных группах или привилегий учётной записи локальной системы.
- Локальная служба – в отличие от сетевого сервиса позволяет обращаться только к сетевым ресурсам, разрешающим анонимный доступ.

Характеристики учётной записи локальной системы (System)

- Обладатель – член группы локальных администраторов.
- Задание практически любых привилегий.
- Полный доступ к большинству файлов и разделов реестра.
- Применяется профиль пользователя по умолчанию, поэтому конфигурационная информация других пользователей недоступна.

Характеристики учётной записи сетевой службы (NetworkService)

Выполняемые процессы используют профиль учётной записи, расположенный в:

- \Documents and Settings\NetworkService
- HKU\S-1-5-20

Характеристики учётной записи локальной службы (LocalService)

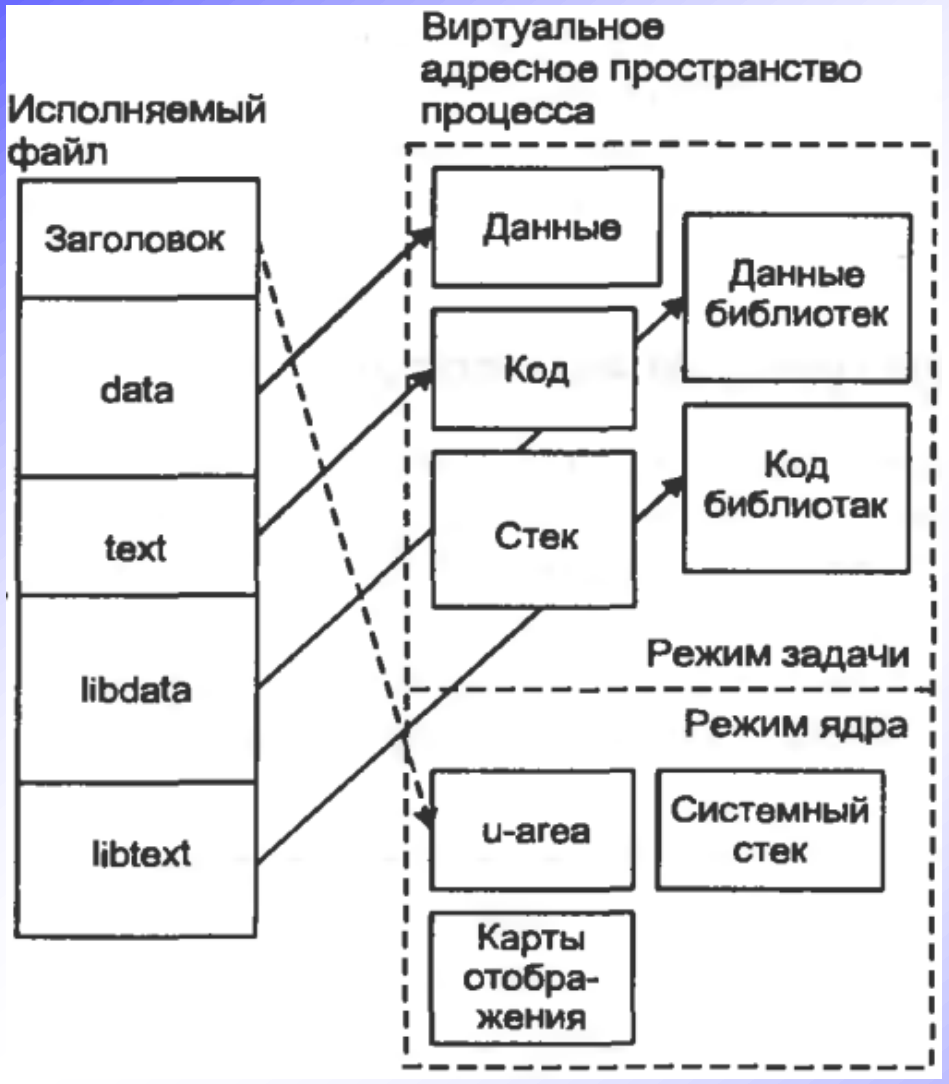
Выполняемые процессы используют профиль учётной записи, расположенный в:

- \Documents and Settings\LocalService
- HKU\S-1-5-19

Примеры служб

- Диспетчер печати – spoolsv.exe.
- Диспетчер учётных записей безопасности – lsass.exe.
- Plug-and-Play, Планировщик заданий, Питание и др. – svchost.exe.

Загрузка программы



Заголовок исполняемого файла содержит общую информацию о файле (номер версии; время и дата создания; размер раздела инструкций; размер раздела данных и т.д.).

Структура образа исполняемой программы

Выполнение программы

- Системные вызовы – обращение прикладной программы к ядру операционной системы для выполнения какой-либо операции.
- С точки зрения программиста системный вызов обычно выглядит как вызов подпрограммы или функции из системной библиотеки.
- Обработка системного вызова требует перевода системы в привилегированный режим работы.

Интерфейс программирования приложений (API)

- API – набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) для использования во внешних программных продуктах.
- Главный API операционных систем – это множество системных вызовов.

Задачи API-функций

- Функции API позволяют разработчику строить результирующую прикладную программу так, чтобы использовать средства целевой вычислительной системы для выполнения типовых операций.
- Разработчик программы избавлен от необходимости создавать исходный код для выполнения этих операций.

Примеры API

API операционных систем:

- POSIX;
- Windows API.

API графических интерфейсов:

- OpenGL;
- X11;
- Direct3D и DirectDraw (части DirectX).

API аутентификационных систем:

- BioAPI;
- PAM.

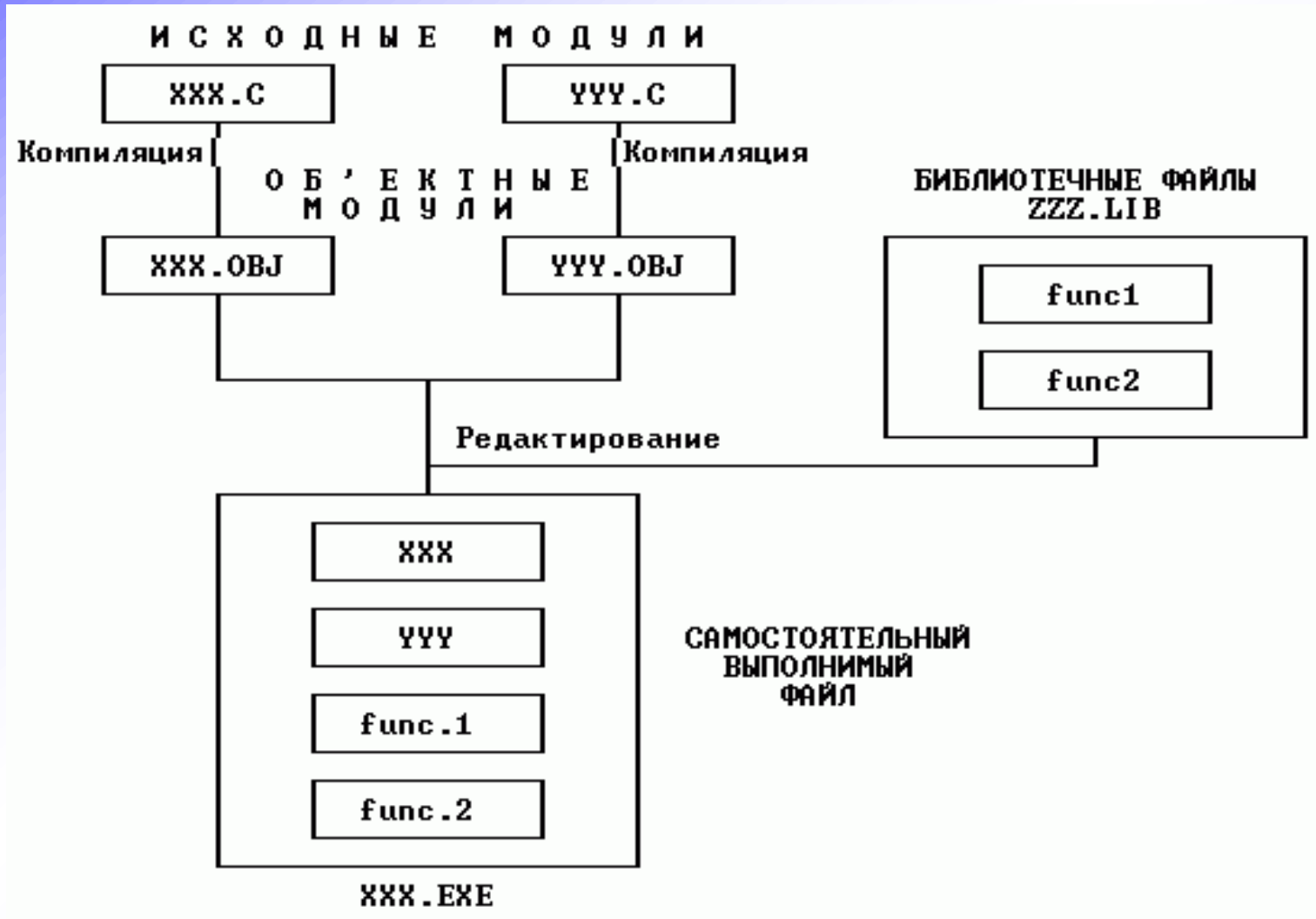
Варианты реализации API-функций

- Реализация на уровне модулей операционной системы (динамические библиотеки).
- Реализация на уровне системы программирования (статические библиотеки).
- Реализация на уровне внешней библиотеки процедур и функций (статические и динамические библиотеки).

Типы библиотек

- Статические библиотеки – исходный текст, подключаемый программистом к своей программе на этапе написания, или объектные файлы, присоединяемые к исполняемой программе на этапе компиляции.
- Динамические библиотеки – часть основной программы, которая загружается в ОС по запросу работающей программы в ходе её выполнения

Организация статических вызовов



Организация динамических вызовов

API операционной системы

- Объектный код, выполняющий API-функции, либо непосредственно входит в состав операционной системы (или даже ядра операционной системы), либо находится в составе динамически загружаемых библиотек, поставляемых вместе с системой.
- Система программирования ответственна только за то, чтобы организовать интерфейс для вызова этого кода.

Основные библиотеки Windows

- Kernel32.dll – управление памятью, процессами и потоками.
- User32.dll – поддержка пользовательского интерфейса, в том числе функции, связанные с созданием окон и передачей сообщений.
- GDI32.dll – графика и вывод текста.

Реализация API на уровне системы программирования

- API-функции разных операционных систем, выполняя одну и ту же операцию (открытие файла, выделение памяти и т.п.), могут иметь различные названия и параметры.
- Система программирования предоставляет разработчику единую функцию, независящую от типа операционной системы.

API-функции внешних библиотек

- При реализации API-функций с помощью внешних библиотек эти функции предоставляются пользователю в виде библиотеки процедур и функций, созданной сторонним разработчиком.
- Система программирования ответственна только за то, чтобы подключить объектный код библиотеки к результирующей программе. Причём внешняя библиотека может быть и динамически загружаемой во время выполнения программы.

Управление процессами

(часть 1)

Задачи подсистемы управления процессами

- Генерация и хранение данных о потребности процесса в ресурсах и о фактически выделенных ресурсах.
- Выделение оперативной памяти, процессорного времени и др. ресурсов для работы процесса.
- Поддержание очередей заявок процессов на ресурсы.

Задачи подсистемы управления процессами

- Защита ресурсов, выделенных процессу, от вмешательства других процессов и организация совместного доступа к ресурсам.
- Синхронизация работы процессов при доступе к совместно используемым ресурсам (остановка процесса до наступления какого-либо события).
- Реализация межпроцессного взаимодействия.

Процессы и потоки

Понятие «процесс»

Процесс – единица работы в ОС, предназначенная для обеспечения многозадачности.

На каждый процесс выделяются:

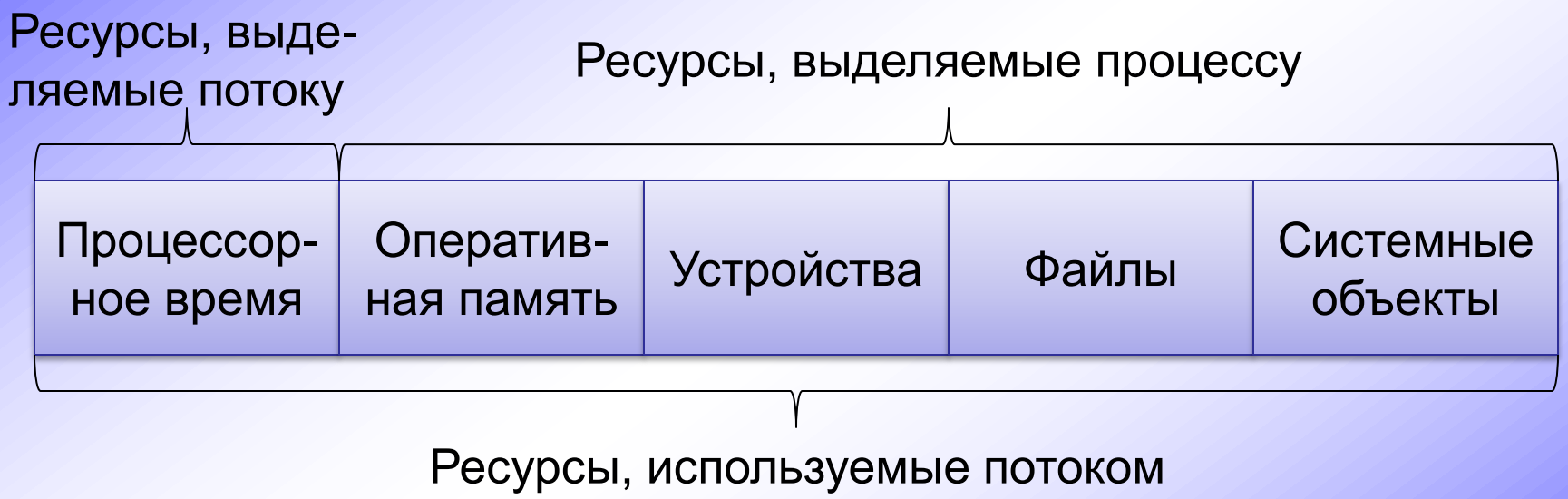
- собственное адресное пространство в оперативной памяти;
- ресурсы ОС, необходимые для работы соответствующей программы.

Понятие «поток»

Поток – единица работы ОС, предназначенная для обеспечения параллельного выполнения задач в рамках процесса.

Процессорное время распределяется между потоками.

Распределение ресурсов между процессами и потоками



Потоки, запущенные в рамках процесса, получают доступ ко всем его ресурсам. Таким образом, потоки одного процесса используют одни и те же ресурсы.

Особенности использования потоков

Преимущества:

- использование одного адресного пространства в рамках процесса;
- более быстрое создание и уничтожение потоков;
- увеличение быстродействия при одновременной работе устройств ввода-вывода и вычислений;
- параллельное выполнение в многопроцессорных системах.

Недостатки:

- усложнение программной модели;
- необходимость учитывать совместное использование ресурсов.

События, приводящие к созданию процессов

- Инициализация ОС.
- Создание работающим процессом дочернего.
- Запрос пользователя на создание процесса (запуск программы).
- Инициирование пакетного задания.

Причины завершения процесса

- Обычный выход по окончании работы процесса.
- Выход по ошибке при работе устройством.
- Выход по ошибке в работе программы.
- Уничтожение другим процессом.

Создание процесса

- Создание дескриптора процесса.
- Выделение процессу адресного пространства.
- Загрузка кодов и данных исполняемой программы в оперативную память.

Дескриптор процесса

Дескриптор процесса – описание, содержащее информацию необходимую на протяжении всего жизненного цикла процесса.

Элементы дескриптора процесса:

- идентификатор процесса;
- открытые файлы;
- дочерние процессы;
- глобальные переменные;
- расположение в памяти и др.

Дескриптор потока

Элементы дескриптора потока:

- идентификатор потока;
- счетчик команд;
- указатель стека;
- приоритет;
- состояние потока и др.

Сохранение и восстановление процессов

Причины переключения процессов

Выбор на выполнение потока другого процесса в случае:

- перехода текущего потока в состояние ожидания запрошенного ресурса;
- завершения текущего потока;
- окончания выполнения системного вызова, сгенерированного текущим потоком;
- окончания обработки прерывания, возникшего во время выполнения текущего потока.

Компоненты контекста процесса

Содержимое регистров (заменяется при любом переключении потока):

- счётчик команд, указывающий адрес следующей команды, которую будет выполнять центральный процессор;
- указатели стека ядра и пользовательского стека;

Компоненты контекста процесса

- регистр состояния процессора (описание результата последних вычислений, уровень прерывания процессора, текущий режим выполнения процесса – ядра/пользовательский и т.д.);
- регистров общего назначения.

Компоненты контекста процесса

Статическая часть контекста процесса:

- номер записи в таблице процессов (состояние процесса, управляющая информация, постоянно необходимая ядру – приоритет и др.);
- управляющая информация, необходимая ядру только во время выполнения процесса (используемые устройства, файлы, идентификатор пользователя и др.);
- каталог таблиц страниц процесса.

Компоненты контекста процесса

Динамическая часть контекста процесса – уникальная для каждого потока данного процесса (заменяется при любом переключении потока):

- стек ядра;
- пользовательский стек.

Переключение контекста

Типовое переключение контекста требует сохранения и восстановления следующих данных:

- состояния регистров;
- пользовательский стек;
- каталог таблиц страниц процесса; таблица дескрипторов ресурсов (файлов, устройств и т.п.), предоставленных процессу.

Сохранение и восстановление контекста

- Необходимые данные сохраняются в текущем стеке ядра. При этом обновляется указатель стека ядра.
- Указатель стека ядра устанавливается на стек ядра нового потока.
- Загружается контекст нового потока.
- Если новый поток принадлежит другому процессу, то происходит загрузка адреса его каталога страниц.

Рассмотренные вопросы

- Типы программ и представление программ в виртуальном адресном пространстве.
- Организация динамических и статических вызовов.
- Понятия «процесс» и «поток».
- Дескрипторы процесса и потока.
- Сохранение и восстановление контекстов процессов.

Всем спасибо – все свободны,

если нет вопросов