Специальность 09.02.07
«Информационные системы и программирование»

«Внедрение и поддержка компьютерных систем»

Модуль №2

Группа
С 121к, С 122

Модуль 2 Методы решения проблем совместимости

- Анализ приложений с проблемами совместимости.
- Использование динамически загружаемых библиотек.
- Механизм решения проблем совместимости на основе «системных заплаток».
- Разработка модулей обеспечения совместимости

Тема 1 Совместимость программного обеспечения

- Анализ приложений с проблемами совместимости.

Тема 1 Совместимость программного обеспечения

- Использование динамически загружаемых библиотек.

Использование динамически загружаемых библиотек.

Понятие динамически подключаемых библиотек

- **Библиотека DLL представляет собой коллекцию подпрограмм, которые могут быть вызваны на выполнение приложениями или подпрограммами из других библиотек.**

- Подобно модулям, библиотеки DLL содержат разделяемый (sharable) код или ресурсы. Однако, в отличие от модулей, библиотеки содержат отдельно откомпилированный исполняемый код, который подключается к приложению динамически на этапе его выполнения, а не компиляции.

- **Для того чтобы библиотеки можно было отличить от самостоятельно выполняемых приложений, они имеют расширение .dll.**

Использование динамически загружаемых библиотек.

Понятие динамически подключаемых библиотек

- Библиотеки могут содержать два вида подпрограмм: экспортируемые и внутренние.
- Экспортируемые подпрограммы могут вызываться процессом, подключающим библиотеку, а внутренние могут быть вызваны только внутри библиотеки.
- Библиотека загружается в адресное пространство процесса, который ее вызвал.
- Подпрограммы библиотеки также используют стековое пространство процесса и его динамическую память.

Использование динамически загружаемых библиотек.

Понятие динамически подключаемых библиотек

- Преимущества использования динамических библиотек (по сравнению со статическим подключением подпрограмм на этапе сборки приложения) следующие:

- Процессы, которые загрузили библиотеку по одному и тому же базовому адресу, пользуются одной копией их кода, но имеют свои собственные копии данных. Благодаря этому снижаются требования к объему памяти и снижается своппинг.
- Модификация подпрограмм библиотек не требует повторной компиляции приложений до тех пор, пока остается неизменным формат их вызова.
- Библиотеки обеспечивают также послепродажную поддержку (after-market support). Например, дисплейный драйвер, предоставляемый библиотекой, может быть обновлен для того, чтобы поддерживать дисплей, который не существовал в момент продажи приложения.
- Программы, написанные на разных языках программирования, могут использовать одну и ту же библиотечную функцию, если они следуют соглашению по вызову, предусмотренному функцией.

Использование динамически загружаемых библиотек.

Понятие динамически подключаемых библиотек

- Интерфейс прикладных программ (Microsoft Win32 application programming interface – API) реализован как набор DLL–библиотек.
- Недостатком использования библиотек является то обстоятельство, что приложение не является самодостаточным и зависит от наличия отдельного файла(ов) библиотеки.

- **В случае отсутствия статически загружаемой библиотеки система прерывает приложение и выводит сообщение об ошибке.**
- **В случае отсутствия динамически загружаемой библиотеки приложение не прерывается, но функции библиотеки являются, разумеется, недоступными для приложения**

Использование динамически загружаемых библиотек.

Вызов подпрограмм из библиотек DLL

- Прежде чем программа сможет вызвать из библиотеки подпрограмму, ее необходимо импортировать.
- Сделать это можно двумя путями: объявить подпрограмму с директивой external или использовать прямой вызов с помощью Windows API.
- При этом в любом случае подпрограмма подключается к приложению только на этапе выполнения приложения. Из этого также следует, что на этапе компиляции нет проверки корректности вызова подпрограммы.

Использование динамически загружаемых библиотек.

Статическая загрузка подпрограмм.

- Простейший способ импорта подпрограммы из библиотеки состоит в объявлении ее с помощью директивы external, например:

```
procedure DoSomething; external 'MyLib.dll';
function MessageBox(HWnd: Integer; Text, Caption: PChar; Flags: Integer):
Integer; stdcall; external 'user32.dll' name 'MessageBoxA';
// MessageBoxA – 'настоящее' имя функции в библиотеке
```

- Если включить эти объявления в программу, библиотеки MyLib.dll и user32.dll будет загружена в память при запуске программы на выполнение.
- В течение всего времени выполнения программы идентификатор DoSomething или MessageBox будет ссылаться на одну и ту же точку входа в той же библиотеке.

Использование динамически загружаемых библиотек.

Статическая загрузка подпрограмм.

- Объявления импортируемых подпрограмм можно поместить непосредственно в программу или модуль, которые их используют.
- Вместе с тем использование библиотечных подпрограмм можно упростить, если собрать их объявления в отдельном модуле, который бы содержал также описания, необходимые для использования подпрограмм из библиотек.
- Примером может служить модуль Windows, предоставляющий доступ к функциям Windows API.
- Другие модули теперь будут просто подключать модуль импорта и вызывать необходимые подпрограммы обычным путем.

Использование динамически загружаемых библиотек.

Статическая загрузка подпрограмм.

- Пример использования статической библиотеки

```
Unit Unit1;
interface
uses Windows, …;
type
TForm1 = class(TForm)
…
end;
{Импортируемыефункции}
Function Max(a,b:integer):integer; external 'MyDll.dll';
Function Min(a,b:integer):integer; external 'MyDll.dll';
…
var
Form1: TForm1;
implementation
…
end.
```

Использование динамически загружаемых библиотек.

Динамическая загрузка библиотек.

- Приложение может самостоятельно загружать и выгружать библиотеки и получать доступ к их подпрограммам с использованием функции GetProcAddress. Для целей загрузки и выгрузки библиотек предназначены подпрограммы LoadLibrary и FreeLibrary

- Пример их использования далее на слайдах

- **Unit Unit1;**
- **interface**
- **uses Windows, …;**
- **const DllName = 'MyDll.dll'; //можно указать 'MyDll'**
- **type**
- **TForm1 = class(TForm)**
- **…**

Использование динамически загружаемых библиотек.

Динамическая загрузка библиотек.

- **Procedure FormCreate (Sender : TObject);**
- **Procedure FormClose (Sender: TObject; var Action: TCloseAction);**
- **public**
- **MyHandle : integer; {для получения результата выполнения функции**
- **LoadLibrary}**
- **end;**
- **{Тип TFunction необходим для того, чтобы воспользоваться функцией**
- **GetProcAddress для получения адреса необходимой функции из библиотеки}**
- **TFunction = Function (a,b:integer) : integer;**
- **var**
- **Form1: TForm1;**
- **FMax,FMin : TFunction;**
- **implementation**
- **{$R *.DFM}**

Использование динамически загружаемых библиотек.

Динамическая загрузка библиотек.

```
Procedure TForm1.FormCreate ( Sender : TObject );
Begin
...
MyHandle := LoadLibrary ( DllName );
if MyHandle = 0 then
begin
ShowMessage('Ошибка при загрузке библиотеки ' + DllName);
Application.Terminate;
end;
@FMax := GetProcAddress(MyHandle,'Max');
@FMin := GetProcAddress(MyHandle,'Min');
if (@FMax=nil) or (@FMin=nil) then
begin
ShowMessage('Не найдены необходимые функции в библиотеке ' + DllName);
FreeLibrary(MyHandle);
Application.Terminate;
end;
End;
```

Использование динамически загружаемых библиотек.

Динамическая загрузка библиотек.

```
...
{теперь подпрограммы можно вызывать используя имена FMax и FMin}
...
Procedure TForm1.FormClose( Sender : TObject; var Action: TCloseAction);
Begin
{Выгружаем библиотеку из памяти}
FreeLibrary(Handle);
End;
...
END.
```

- В этом способе использования библиотеки ее загрузка в память не выполняется до вызова функции LoadLibrary, что позволяет экономить память. Кроме того, в этом случае можно выполнять программу даже в том случае, если какие-либо библиотеки не найдены (если, разумеется, в этом есть какой-либо смысл). Чтобы получить более подпробную информацию об ошибке, надо использовать функцию GetLastError.

Использование динамически загружаемых библиотек.

Динамическая загрузка библиотек.

- В этом способе использования библиотеки ее загрузка в память не выполняется до вызова функции LoadLibrary, что позволяет экономить память.

- Кроме того, в этом случае можно выполнять программу даже в том случае, если какие-либо библиотеки не найдены (если, разумеется, в этом есть какой-либо смысл).

- Чтобы получить более подпробную информацию об ошибке, надо использовать функцию GetLastError.

Использование динамически загружаемых библиотек.

Динамическая загрузка библиотек.

- Поиск загружаемой библиотеки, если ее имя не содержит пути, выполняется по следующим маршрутам (см. Help для функции LoadLibrary):

- В каталоге, из которого было запущено приложение.
- В текущем каталоге.
- В каталоге System (Windows 95) или в каталоге System32 (Windows NT). Для получения пути к этому каталогу можно использовать функцию GetSystemDirectory.
- Для Windows NT: в каталоге SYSTEM (16-битная версия).
- В каталоге Windows. Для получения пути к этому каталогу можно использовать функцию GetWindowsDirectory.
- В каталогах, перечисленных в переменной окружения PATH.

Использование динамически загружаемых библиотек.

Динамическая загрузка библиотек.

- При попытке повторной загрузки библиотеки ошибки не происходит, и библиотека не загружается.
- В любой момент времени библиотеку можно выгрузить с помощью процедуры FreeLibrary.

Тема 1 Совместимость программного обеспечения

- Механизм решения проблем совместимости на основе «системных заплаток».

Тема 1 Совместимость программного обеспечения

- Разработка модулей обеспечения совместимости