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

Динамически подключаемые библиотеки

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. Программы, написанные на Object Pascal, могут использовать библиотеки, написанные на других языках. С другой стороны, приложения Windows могут использовать библиотеки, написанные на Object Pascal.

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.

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

1 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.

2 Модификация подпрограмм библиотек не требует повторной компиляции приложений до тех пор, пока остается неизменным формат их вызова.

3 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.

4 Программы, написанные на разных языках программирования, могут использовать одну и ту же библиотечную функцию, если они следуют соглашению по вызову, предусмотренному функцией.

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.

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.

Необходимо отметить, что Object Pascal не поддерживает импорт переменных из DLL.

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.

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.

Пример использования статической библиотеки (полный текст см. в проекте UseDll):

```
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.
```

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 *LoadLibrar*y и *FreeLibrar*y. Пример их использования (полный пример см. в проекте UseDll_2):

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

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

1 В каталоге, из которого было запущено приложение.

2 В текущем каталоге.

3 В каталоге System (Windows 95) или в каталоге System32 (Windows NT). Для получения пути к этому каталогу можно использовать функцию GetSystemDirectory.

4 Для Windows NT: в каталоге SYSTEM (16-битная версия).

5 В каталоге Windows. Для получения пути к этому каталогу можно использовать функцию GetWindowsDirectory.

6 В каталогах, перечисленных в переменной окружения PATH.

При попытке повторной загрузки библиотеки ошибки не происходит и библиотека не загружается.

В любой момент временибиблиотеку можно выгрузить с помощью процедуры *FreeLibrar*y.

Решение проблем совместимости приложений.

Введение

Одним из самых важных новшеств в Microsoft® Windows® XP стало добавление целого ряда технологий совместимости приложений, доступных даже конечным пользователям через оболочку Windows XP. Распространение исправлений совместимости приложений на большом количестве компьютеров может быть трудным или невыполнимым, если оно предоставлено каждому пользователю компьютера. К счастью, есть более простой способ собирать группы исправлений совместимости и распределять их путем автоматической установки на компьютеры, работающие под управлением Windows XP.

Администратор совместимости

После установки группы необходимых исправлений совместимости, Вы можете воспользоваться Администратором совместимости, чтобы скомпоновать исправления

совместимости для распространения на других компьютерах, работающих под управлением Windows XP.

Создание собственных оболочек совместимости с помощью Администратора совместимости

В этом разделе обсуждается как можно создавать и подготавливать файлы собственной базы данных с помощью Администратора совместимости, для поддержания множества приложений на одном или нескольких компьютерах, работающих под управлением Windows XP.

Администратор совместимости может компоновать исправления и оболочки совместимости для множества приложений в один файл базы данных совместимости (*.sdb), который потом может быть перенесен на другие компьютеры, работающие под управлением Windows XP. Это особенно полезно в большом сетевом окружении, где несколько человек должны обеспечивать поддержку программного обеспечения огромному числу пользователей.

Установка Администратора совместимости

Администратор совместимости, поставляемый с операционной системой Windows XP, может быть найден в папке Support Tools на установочном компакт-диске. Администратор совместимости распространяется как часть Пакета средств обеспечения совместимости приложений (Application Compatibility Toolkit) версии 2.0 и выше.

Для установки Пакета средств обеспечения совместимости приложений (Application Compatibility Toolkit) в Вашей ОС Windows XP:

1. Вставьте установочный компакт-диск Windows XP в привод компакт-дисков
2. Используя Мой компьютер (My Computer) или Проводник (Windows Explorer), перейдите на привод, в который Вы вставили диск с ОС Windows XP, и откройте папку Support Tools.
3. Щелкните дважды файл ACT. EXE для начала установки программы. Примите настройки, предложенные по умолчанию программой установки.

После установки Пакета средств обеспечения совместимости приложений (Application Compatibility Toolkit) его можно будет найти в меню Пуск. Администратор совместимости находится в группе Пакета средств обеспечения совместимости приложений (Application Compatibility Toolkit) в меню Пуск.

Использование Администратора совместимости

Windows XP содержит информацию о распространенных проблемах совместимости, которые могут происходить с некоторыми приложениями. Исправления совместимости приложений, предоставляемые Microsoft в Windows XP, созданы, чтобы помочь Windows XP должным образом поддерживать нормальную работу этих приложений, не ставя при этом под угрозу стабильность системы.

Четыре библиотеки DLL содержат все исправления совместимости Четыре библиотеки DLL, расположенные в папке % WINDIR% AppPatch, содержат все исправления совместимости. Файлы APPHELP.SDB и SYSMAIN.SDB обеспечивают

работу справочных сообщений приложений, а исправления приложений являются частью Windows XP.

Справочные сообщения приложений содержат информацию, которая отображается при запуске действительно несовместимого с Windows XP приложения. Примеры приложений, которые могут вызвать появление Справочных сообщений приложений, включают:

- Антивирусные программы
- Программы, которые требуют доступа на уровне ядра операционной системы
- Программы, которые устанавливают специфические драйверы файловой системы

Причины неправильной работы приложений
Приложения, которые были созданы для работы с предыдущими версиями Windows, могут неправильно работать в ОС Windows XP Professional. Причины, по которым это может происходить:

- Приложение не запускается, когда Windows сообщает ему о новой версии операционной системы. Зачастую приложение работает нормально, если пользователь сможет обойти этот момент.
- Приложение обращается к старым версиям функций Win32 API, которые возвращают непредсказуемые значения на компьютерах с большим количеством ресурсов, таких как дисковое пространство
- Приложение ожидает старых форматов данных Windows.
- Приложение ожидает, что информация пользователя, такая как личные и временные папки, будет в определенном месте или в определенном формате.

Для устранения этих проблем с помощью Администратора совместимости Вам необходимо создать собственную базу данных, содержащую информацию об исправлениях совместимости, необходимую Вашим приложениям, а также информацию о соответствии файлов, которая позволяет Windows XP однозначно распознать приложение, требующее поддержки.

Создание собственной базы данных совместимости

Администратор совместимости позволяет Вам просматривать исправления совместимости приложений, хранящиеся в защищенных системой базах данных, чтобы применять нужные исправления для сотен приложений. Основной интерфейс Администратора позволяет контролировать приложения с исправлениями совместимости путем просмотра их в базе данных ОС Windows XP Professional. Эта информация отображается в верхней левой части (части системной базы данных) окна Администратора совместимости.

Системная база данных совместимости является составляющей операционной системы Windows XP Professional, обеспечивающей идеальную совместимость для сотен Windows-приложений. Эта база данных и соответствующие компоненты защищены операционной системой.

Как только Вы определили и проверили исправления для определенного приложения, можно запустить Администратор совместимости для создания базы данных исправлений. Вы можете создать базу данных, которая содержит приложения,

поддерживаемые оболочками совместимости, или приложения, поддерживаемые определенными исправлениями совместимости.

Чтобы создать новую собственную базу данных с помощью Администратора совместимости:

1. Откройте Администратор совместимости выбрав в меню Пуск (Start), Программы(All Programs), Пакет средств обеспечения совместимости приложений (Application Compatibility Toolkit), Администратор совместимости
2. Если у Вас открыта собственная база данных, в меню Файл (File) выберите Новый (New).
3. Зайдите в меню База данных (Database) и нажмите Изменить название базы данных (Change Database Name). Как только Вы измените название базы данных, оно будет отображаться в заголовке собственной базы данных. Если пункт менюИзменить имя базы данных (Change Database Name) не активен, щелкните по области базы данных окна.
4. В меню Файл (File) нажмите Сохранить (Save) и дайте название своему .sdb файлу. Теперь можно добавить исправления в Вашу собственную базу данных.

Как только Вы создали собственную пустую базу данных, которая будет содержать Ваши исправления совместимости приложений, можно добавить оболочку совместимости.

Для добавления оболочки совместимости

1. Выберите Создать исправление приложения (Create Application Fix) в меню База данных. Появится диалоговое окноСоздание исправления приложения (Create an Application Fix).
2. Выберите Применить режим совместимости (Apply Compatibility Mode) и нажмите кнопку Далее (Next).
3. Введите название приложения, для которого Вы будете определять режим совместимости, и нажмите кнопку Далее(Next).
4. Введите название файла, к которому будет применен режим совместимости. Вы можете набрать название файла вручную или использовать кнопку Обзор (Browse), чтобы указать его.
5. Выберите из выпадающего списка режим совместимости, который нужно применить, и нажмите Далее (Next).
6. Нажмите кнопку Добавить файл (Add File), чтобы выбрать файлы, которые помогут точно определить нужный файл на целевых компьютерах (Выберите файлы, связанные с приложением, которые будут установлены в то же место. Например, выберите файл .hlp, находящийся в одной папке с .exe файлом. Постарайтесь однозначно определить Ваш файл, не выбирая большое количество соответствующих файлов).
7. Когда выберете все необходимые файлы, нажмите Далее (Next).
8. Если Вы хотите проверить приложение с примененным исправлением, нажмите Выполнить тестирование (Test Run). В противном случае нажмите Готово (Finish).

Тот же процесс может быть использован для добавления индивидуальных исправлений совместимости в собственную базу данных, за исключением того, что в окне Создать исправление приложения (Create an Application Fix) Вы должны выбрать вариантПрименить определенное исправление совместимости

(Apply Specific Compatibility Fix). Как только все исправления и оболочки будут добавлены в базу данных, сохраните базу данных и проверьте приложение. Совпадение имен файлов

Применение собственной базы данных к системе

Как только Вы создали Вашу собственную базу данных исправлений совместимости приложений, она должна быть применена к системе компьютера, на котором это приложение будет работать. Общий процесс развертывания исправлений совместимости на нескольких компьютерах под управлением Windows XP включает следующие действия:

- Определите и проверьте исправления для необходимых приложений.
- Создайте файл выборочной базы данных с нужными исправлениями.
- Перенесите .sdb файл на нужные компьютеры под управлением Windows XP.
- Используйте команду SDBINST.EXE, чтобы зарегистрировать базу данных. Она автоматически установит и добавит информацию об исправлениях в реестр на выбранных компьютерах.

Перенос файла собственной базы данных на другие компьютеры под управлением Windows XP
Перенос файла собственной базы данных на другие компьютеры под управлением Windows XP может быть проведена разными способами:

- Можно поместить файл базы данных в программу установки и распространить его с помощью Групповой политики в сети с Active Directory, но это требует дополнительной работы.
- Файл может быть скопирован вручную на каждый удаленный компьютер, или это можно сделать с помощью сценария входа в систему.
- Еще одной возможностью является размещение файла .sdb на общем сетевом ресурсе, к которому имеют доступ все пользователи Windows XP.

Несмотря на то, что файл перенесен на удаленные компьютеры, содержащаяся в нем информация должна быть зарегистрирована на каждом компьютере. Это делается с помощью запуска команды SDBINST.EXE из командной строки, за которой следует полный путь и имя созданного .sdb файла. Например:

```
Sdbinst c:WindowsAppPatchmyapp.sdb
```

Как только база данных зарегистрирована на компьютере, информация о совместимости будет использоваться каждый раз при запуске приложения.

Заключение

Windows XP предоставляет улучшенную поддержку приложений по сравнению с предыдущими версиями операционных систем Windows. Помимо встроенной поддержки для решения огромного разнообразия известных проблем совместимости приложений, новые средства, включая Пакет средств обеспечения совместимости приложений (Application Compatibility Toolkit), позволяют системным администраторам осуществлять поддержку всех их приложений. Администратор совместимости является инструментом из Пакета средств обеспечения совместимости приложений. Администратор совместимости позволяет системным

администраторам брать информацию, полученную путем тестирования, и упаковывать её в индивидуальную базу данных совместимости. Эта база данных может использоваться для поддержки множества приложений, и может быть легко распространена на другие компьютеры, нуждающиеся в исправлениях совместимости. Для регистрации файла базы данных совместимости на удаленных компьютерах используется команда SDBINST.EXE, после чего информация будет доступна в Windows XP каждый раз при запуске приложения.

Создание в системе виртуальной машины для исполнения приложений.

Режим виртуальных машин для исполнения приложений реального режима

Разработчики рассматриваемого семейства микропроцессоров в своем стремлении обеспечить максимально возможную совместимость архитектуры пошли не только на то, чтобы обеспечить возможность программам, созданным для первых 16-разрядных ПК, без проблем выполняться на компьютерах с более поздними моделями микропроцессоров за счёт введения реального режима работы. Они также обеспечили возможность выполнения 16-разрядных приложений реального режима при условии, что сам процессор при этом функционирует в защищённом режиме работы и операционная система, используя соответствующие аппаратные средства микропроцессора, организует мультипрограммный (мультизадачный) режим. Другими словами, микропроцессоры i80x86 поддерживают возможность создания операционных сред реального режима при работе микропроцессора в защищённом режиме. Если условно назвать 16-разрядные приложения DOS-приложениями (поскольку в абсолютном большинстве случаев это именно так), то можно сказать, что введена поддержка для организации виртуальных DOS-машин, работающих вместе с обычными 32-битовыми приложениями защищённого режима. Это даже нашло отражение в названии режима работы микропроцессоров i80x86 – режим виртуального процессора i8086, иногда (для краткости) его называют режимом V86 или просто виртуальным режимом, – при котором в защищённом режиме работы может исполняться код DOS-приложения. Мультизадачность при выполнении нескольких программ реального режима будет поддержана аппаратными средствами защищённого режима.

Переход в виртуальный режим осуществляется посредством изменения бита VM(virtualmode) в регистре EFLAGS. Когда процессор находится в виртуальном режиме, для адресации памяти используется схема реального режима работы – (сегмент: смещение) с размером сегментов до 64 Кбайт, которые могут располагаться в адресном пространстве размером в 1 Мбайт, однако полученные адреса считаются не физическими, а линейными. В результате применения страничной трансляции осуществляется отображение виртуального адресного пространства 16-битового приложения на физическое адресное пространство. Это позволяет организовать параллельное выполнение нескольких задач, разработанных для реального режима, да ещё и совместно с обычными 32-битовыми приложениями, требующих защищённого режима работы. Естественно, что для обработки прерываний, возникающих при выполнении 16-битовых приложений в виртуальном режиме, процессор возвращается из этого режима в обычный защищённый режим. В противном случае невозможно было бы организовать полноценную виртуальную машину. Очевидно, что обработчики прерываний для виртуальной машины должны эмулировать работу подсистемы прерываний процессораi8086. Другими словами, прерывания

отображаются в операционную систему, работающую в защищённом режиме, и уже основная ОС моделирует работу операционной среды выполняемого приложения.

Вопрос, связанный с операциями ввода/вывода, которые недоступны для обычных приложений, решается аналогично. При попытке выполнить недопустимые команды ввода/вывода возникают прерывания, и необходимые операции выполняются операционной системой, хотя задача об этом и «не подозревает». При выполнении команд IN,OUT,INS,OUTS,CLI,STIпроцессор, находящийся в виртуальном режиме и исполняющий код на уровне привилегий третьего (самого нижнего) кольца защиты, за счёт возникающих вследствие этого прерываний переводится на выполнение высоко привилегированного кода операционной системы.

Таким образом, ОС может полностью виртуализировать ресурсы компьютера: и аппаратные, и программные, создавая иную полноценную операционную среду; при существовании так называемых нативных приложений, создаваемых по собственным спецификациям данной ОС. Очень важным моментом для организации полноценной виртуальной машины является реализация виртуализации не только программных, но и аппаратных ресурсов. Так, например, в OCWindowsNTэта задача выполнена явно неудачно, тогда как в OS/2 имеется полноценная виртуальная машина как для DOS-приложений, так и для приложений, работающих в среде спецификацийWin16. Правда, в последнее время это уже перестало быть актуальным, поскольку появилось большое количество приложений, работающих по спецификациям Win32API.