Белгородский Государственный Технологический Университет им. В. Г. Шухова

Кафедра программного обеспечения вычислительной техники  
и автоматизированных систем

## Лабораторная работа №5 по теме: «Операции с файлами»

Выполнил:  
студент группы ПВ-31  
Адаменко И. И.

Проверил:  
старший преподаватель  
Гарибов А. И.

Белгород  
2015

Цель работы: получение практических навыков по использованию Win32 API для работы с файлами.

## Задание

Реализовать программу экспорта выбранной ветки системного реестра в бинарный файл, а также программу для просмотра созданного файла.

## Отчёт

### Основные понятия

Файлы инициализации  
Windows даёт приложениям возможность легко сохранить данные инициализации или состояния в текстовых файлах стандартного формата, называемых файлами инициализации. Самыми известными примерами файлов инициализации являются WIN.INI и SYSTEM.INI. Они используются некоторыми версиями Windows для сохранения системных настроек (например, информации о шрифтах и устройствах), а также приложениями для сохранения их собственных данных инициализации. Windows также позволяет каждому приложению создавать и поддерживать собственные файлы инициализации.

Файлы инициализации (или профильные файлы, как их иногда называют) имеют стандартный формат, описанный в документации Windows. Каждый файл инициализации делится на секции. Каждой секции присваивается имя, заключённое в квадратные скобки: [имя секции]

Секция содержит записи, которые состоят из двух частей, разделённых знаком =. Первая часть называется именем ключа. Вторая часть содержит строковое значение ключа. Таким образом, обобщённый формат записи выглядит так: имя ключа = строковое значение ключа.

В Windows NT большинство секцийWIN.INI и других системных файлов инициализации отображается на реестр. Если секция INI-файла отображается на реестр, функции работы с INI-файлами записывают данные прямо в реестр, и эти данные вообще не попадают в файлы инициализации.

Функции API для работы с файлами инициализации перечислены в следующей таблице:

|  |  |
| --- | --- |
| Функция | Описание |
| GetProfileInt | Читает целый параметр из файла инициализации WIN.INI |
| GetProfileSection | Читает секцию из файла инициализации WIN.INI |
| GetProfileString | Читает строку из файла инициализации WIN.INI |
| GetPrivateProfileInt | Читает целый параметр из файла инициализации WIN.INI |
| GetPrivateProfileSection | Читает секцию из приватного файла инициализации |
| GetPrivateProfileString | Читает строку из приватного файла инициализации |
| WritePrivateProfileSection | Записывает секцию в приватный файл инициализации |
| WritePrivateProfileString | Записывает строку в приватный файл инициализации |
| WriteProfileSection | Записывает секцию в файл инициализации WIN.INI |
| WriteProfileString | Записывает строку в файл инициализации WIN.INI |

Системный реестр  
Одна из отличительных особенностей DOS и других операционных систем того же класса заключается в том, что пользователь работает с приложением, а приложения, в свою очередь, работают с данными. Иначе говоря, когда вы хотите что-нибудь сделать, вы выбираете приложение, с которым вы собираетесь работать. Затем при помощи приложения вы обращаетесь к файлам на диске, содержащим нужные данные.

Windows обеспечивает другой, документно-ориентированный подход. Вместо того чтобы выбирать приложение, вы выбираете данные, с которыми собираетесь работать. Далее операционная система выбирает нужное приложение.

Чтобы операционная система могла узнать, какое приложение связывается с тем или иным типом объектов, она должна располагать многочисленными данными и об объектах, и о приложениях. Системе необходимы средства уникальной регистрации и идентификации любых объектов — определяемых как системой, так и отдельными приложениями. Нужно уметь определять, какое приложение связано с конкретным объектом. Нужно знать, где на диске хранятся документы и объекты, чтобы их можно было быстро найти. Нужно следить за общими компонентами, чтобы при удалении приложения не удалялись его компоненты, используемые в других приложениях.

По мере эволюции Windows объёмы и типы информации, которую приходится хранить для решения этих задач, выходят за пределы возможностей эффективного использования файлами инициализации. По этой причине Microsoft; разработала системный реестр (также называемый «регистрационной базой данных»). Реестр может содержать существенно больше данных, чем файлы инициализации, и не ограничивается строковым типом данных. Он более надёжен и лучше защищён от ошибок, чем отдельные файлы инициализации, и его сложнее отредактировать или удалить. Реестр также содержит некоторые данные конфигурации системы в зависимости от операционной системы. В Windows 3.x почти все данные системной конфигурации хранились в файлах инициализации — при удалении реестра информация практически не терялась. В Windows 95 часть данных системной конфигурации хранится в реестре, но даже в случае его повреждения или удаления система может сохранить работоспособность. В Windows NT потеря реестра является катастрофой — реестр придётся либо восстанавливать с диска аварийного восстановления, либо переустанавливать систему заново, поскольку Windows NT почти не хранит данные в файлах инициализации.

Архитектура реестра  
Реестр чем-то напоминает миниатюрную файловую систему. Вместо файлов он содержит параметры, а вместо каталогов — ключи. Все вместе образует информационную иерархию. Вместо одного корневого каталога в реестре Windows 95 и Windows NT находится несколько стандартных ключей. Функциям, которым передаются дескрипторы ключей, в большинстве случаев также можно передавать константы стандартных ключей, перечисленные в следующей таблице:

|  |  |
| --- | --- |
| HKEY\_CLASSES\_ROOT | Ключ предназначен для хранения информации о классах файлов и объектов. Здесь хранятся данные о связи типов документов с приложениями, а также идентификаторы классов, используемые объектами OLE. К ключу также можно обратиться через HKEY\_LOCAL\_MACHINE\SOFTWARE\Classes. |
| HKEY\_CURRENT\_USER | В ключе хранятся данные конфигурации для текущего пользователя. |
| HKEY\_LOCAL\_MACHINE | Подробная информация об аппаратной и программной конфигурации системы. |
| HKEY\_USERS | Этот ключ верхнего уровня содержит данные конфигурации программ и системы, уникальные для конкретного пользователя. Подключ HKEY\_USERS\Default содержит настройки по умолчанию для новых пользователей, а также пользователей, не вносивших собственные изменения в стандартную конфигурацию. |
| HKEY\_CURRENT\_CONFIG | Ключ используется для хранения общих данных конфигурации системы. |
| HKEY\_DYN\_DATA | Ключ используется для хранения данных, используемых в текущем сеансе, но не сохраняемых между сеансами Windows. Например, здесь находятся данные о быстродействии системы, относящиеся к текущему сеансу. |

Ключи реестра могут содержать данные различных типов, приведённых в следующей таблице:

|  |  |
| --- | --- |
| REG\_BINARY | Двоичные данные |
| REG\_DWORD | 32-разрядное число |
| REG\_DWORD\_BIGENDIAN | 32-разрядное число, байты которого упорядочены так, что старший байт находится по младшему адресу |
| REG\_DWORD\_LITTLE\_ENDIAN | 32-рязрадное целое, байты которого упорядочены так, что младший байт находится по младшему адресу (стандарт Intel x86) |
| REG\_EXPAND\_SZ | Нерасширяемая строка окружения |
| REG\_LINK | Символическая ссылка |
| REG\_NULTI\_SZ | Список строк, разделяемых нуль-символами. Завершается двумя нуль-символами |
| REG\_NONE | Неопределённый тип |
| REG\_RESOURCE\_LIST | Список ресурсов для драйвера устройства |
| REG\_SZ | Строка, завершённая нуль-символом |

Win32 содержит большое количество функций для работы с реестром. Эти функции перечислены в следующей таблице:

|  |  |
| --- | --- |
| RegCloseKey | Закрывает открытый ключ |
| RegConnectRegistry | Открывает ключ в удалённой системе |
| RegCreateKey, RegCreateKeyEx | Создаёт новый ключ |
| RegDeleteKey | Удаляет ключ |
| RegDeleteValue | Удаляет параметр |
| RegEnumKey, RegEnumKeyEx | Перебирает подключи заданного ключа |
| RegEnumValue | Перебирает параметры ключа |
| RegFlushKey | Обеспечивает запись на диск изменений в ключе |
| RegGetKeySecurity | Получает данные безопасности для ключа |
| RegLoadKey | Загружает информацию реестра, сохранённую в файле |
| RegNotifyChangeKeyValue | Оповещает приложение об изменениях значения или других атрибутов ключа |
| RegOpenKey, RegOpenKeyEx | Открывает ключ |
| RegQueryInfoKey | Получает информацию о ключе |
| RegQueryValue, RegQueryValueEx | Читает значение параметра |
| RegReplaceKey | Замещает данные реестра данными, сохранёнными в файле |
| RegRestoreKey | Восстанавливает данные реестра по данным, сохранённым в файле |
| RegSaveKey | Сохраняет данные реестра в файле |
| RegSetKeySecurity | Задаёт атрибуты безопасности для ключа |
| RegSetValue, RegSetValueEx | Задаёт значение параметра |
| RegUnloadKey | Выгружает заданный ключ и все его подключи из реестра |

Асинхронные файловые операции  
Одним из самых интригующих и потенциально полезных новшеств, поддерживаемых в Win32, является возможность перекрытия файловых операций. В Win16 вся пересылка данных из файла или в файл, выполняемая средствами API (или средствами Delphi), выполнялась в синхронном режиме. Другими словами, когда вы вызывали функцию пересылки данных, выход из этой функции происходил лишь после завершения пересылки. Длинные операции ввода/вывода приводили к заметным задержкам в работе программы.

Win32 позволяет выполнять файловые операции в асинхронном режиме — выход из функции пересылки данных происходит немедленно, а операционная система завершает операцию в фоновом режиме. Подобные операции называются «перекрывающимися».

### Код программы

Часть кода для работы с регистром

|  |  |
| --- | --- |
| 001 | **public** **static** **int** CreateRegKey() |
| 002 | { |
| 003 | **string** nameApplication = *"MyRegApplication"*; |
| 004 |  |
| 005 | **int** newKeyHndl; |
| 006 | **RegResult** regresult; |
| 007 |  |
| 008 | RegCreateKeyEx( |
| 009 | HKEY.HKEY\_CURRENT\_USER, |
| 010 | *"Software\\"* + nameApplication, |
| 011 | 0, |
| 012 | **null**, |
| 013 | **RegOption**.NonVolatile, *// Не удалять после перезагрузки* |
| 014 | RegSAM.AllAccess | RegSAM.WOW64\_32Key, |
| 015 | **IntPtr**.Zero, |
| 016 | **out** newKeyHndl, |
| 017 | **out** regresult); |
| 018 |  |
| 019 | **return** newKeyHndl; |
| 020 | } |
| 021 |  |
| 022 | **public** **static** **void** SaveLocations(**string** saveLocation, **string** loadLocation) |
| 023 | { |
| 024 | **int** hkey = CreateRegKey(); |
| 025 |  |
| 026 | **if** (hkey == 0) { **return**; } |
| 027 |  |
| 028 | **if** (saveLocation != **null** && saveLocation.Length > 0) |
| 029 | { |
| 030 | RegSetValueEx( |
| 031 | hkey, |
| 032 | *"save location"*, |
| 033 | 0, |
| 034 | RegistryValueKind.String, |
| 035 | **Marshal**.StringToHGlobalAnsi((**string**)saveLocation), |
| 036 | ((**string**)saveLocation).Length + 1); |
| 037 | } |
| 038 |  |
| 039 | **if** (loadLocation != **null** && loadLocation.Length > 0) |
| 040 | { |
| 041 | RegSetValueEx( |
| 042 | hkey, |
| 043 | *"load location"*, |
| 044 | 0, |
| 045 | RegistryValueKind.String, |
| 046 | **Marshal**.StringToHGlobalAnsi((**string**)loadLocation), |
| 047 | ((**string**)loadLocation).Length + 1); |
| 048 | } |
| 049 |  |
| 050 | RegCloseKey(hkey); |
| 051 | } |
| 052 |  |
| 053 | **public** **static** **String** GetLocationSave() |
| 054 | { |
| 055 | **int** hkey = CreateRegKey(); |
| 056 |  |
| 057 | **if** (hkey == 0) { **return** *""*; } |
| 058 |  |
| 059 | **StringBuilder** result = **new** StringBuilder(1000); |
| 060 | **int** size = 1000; |
| 061 | RegQueryValueEx( |
| 062 | hkey, |
| 063 | *"save location"*, |
| 064 | 0, |
| 065 | **IntPtr**.Zero, |
| 066 | result, |
| 067 | **ref** size); |
| 068 |  |
| 069 | RegCloseKey(hkey); |
| 070 |  |
| 071 | **return** result.ToString(); |
| 072 | } |
| 073 |  |
| 074 | **public** **static** **String** GetLocationLoad() |
| 075 | { |
| 076 | **int** hkey = CreateRegKey(); |
| 077 |  |
| 078 | **if** (hkey == 0) { **return** *""*; } |
| 079 |  |
| 080 | **StringBuilder** result = **new** StringBuilder(1000); |
| 081 | **int** size = 1000; |
| 082 | RegQueryValueEx( |
| 083 | hkey, |
| 084 | *"load location"*, |
| 085 | 0, |
| 086 | **IntPtr**.Zero, |
| 087 | result, |
| 088 | **ref** size); |
| 089 |  |
| 090 | RegCloseKey(hkey); |
| 091 |  |
| 092 | **return** result.ToString(); |
| 093 | } |
| 094 |  |
| 095 | **public** **static** **bool** DeleteRegKey() |
| 096 | { |
| 097 | **string** nameApplication = *"MyRegApplication"*; |
| 098 |  |
| 099 | **int** result = RegDeleteKeyEx( |
| 100 | HKEY.HKEY\_CURRENT\_USER, |
| 101 | *"Software\\"* + nameApplication, |
| 102 | **RegSAM**.WOW64\_32Key, |
| 103 | 0); |
| 104 |  |
| 105 | **return** result == 0; |
| 106 | } |

Класс, использующий функции для работы с регистром:

|  |  |
| --- | --- |
| 001 | [**Serializable**] |
| 002 | **class** MyReg |
| 003 | { |
| 004 | **public** **String** Root { **get**; **private** **set**; } |
| 005 | **public** **String** Path { **get**; **private** **set**; } |
| 006 | **public** **String** Name { **get**; **private** **set**; } |
| 007 | **public** **List**<MyReg> Keys { **get**; **private** **set**; } |
| 008 | **public** **List**<**String**> Values { **get**; **private** **set**; } |
| 009 |  |
| 010 | **public** MyReg(**String** name, **String** root, **String** path) |
| 011 | { |
| 012 | Name = name; |
| 013 | Root = root; |
| 014 | Path = path; |
| 015 | Keys = **new** List<MyReg>(); |
| 016 | Values = **new** List<**string**>(); |
| 017 | update(); |
| 018 | } |
| 019 |  |
| 020 | **private** **void** update() |
| 021 | { |
| 022 | **UIntPtr** key = getKeyWithPath(); |
| 023 |  |
| 024 | updateValues(key); |
| 025 |  |
| 026 | updateKeys(key); |
| 027 | } |
| 028 |  |
| 029 | **private** **UIntPtr** getKey() |
| 030 | { |
| 031 | **UIntPtr** key = **UIntPtr**.Zero; |
| 032 | **if** (Root == *"HKEY\_CLASSES\_ROOT"*) |
| 033 | { |
| 034 | key = **new**   **UIntPtr**(unchecked((**uint**)win32API.HKEY.HKEY\_CLASSES\_ROOT)); |
| 035 | } |
| 036 | **else** **if** (Root == *"HKEY\_CURRENT\_CONFIG"*) |
| 037 | { |
| 038 | key = **new**  **UIntPtr**(unchecked((**uint**)win32API.HKEY.HKEY\_CURRENT\_CONFIG)); |
| 039 | } |
| 040 | **else** **if** (Root == *"HKEY\_CURRENT\_USER"*) |
| 041 | { |
| 042 | key = **new**   **UIntPtr**(unchecked((**uint**)win32API.HKEY.HKEY\_CURRENT\_USER)); |
| 043 | } |
| 044 | **else** **if** (Root == *"HKEY\_LOCAL\_MACHINE"*) |
| 045 | { |
| 046 | key = **new**   **UIntPtr**(unchecked((**uint**)win32API.HKEY.HKEY\_LOCAL\_MACHINE)); |
| 047 | } |
| 048 | **else** **if** (Root == *"HKEY\_USERS"*) |
| 049 | { |
| 050 | key = **new** **UIntPtr**(unchecked((**uint**)win32API.HKEY.HKEY\_USERS)); |
| 051 | } |
| 052 | **else** |
| 053 | { |
| 054 | key = **new**   **UIntPtr**(unchecked((**uint**)win32API.HKEY.HKEY\_CLASSES\_ROOT)); |
| 055 | } |
| 056 |  |
| 057 | **return** key; |
| 058 | } |
| 059 |  |
| 060 | **private** **UIntPtr** getKeyWithPath() |
| 061 | { |
| 062 | **UIntPtr** key = getKey(); |
| 063 | **UIntPtr** newKey = **UIntPtr**.Zero; |
| 064 |  |
| 065 | win32API.RegOpenKeyEx( |
| 066 | key, |
| 067 | Path, |
| 068 | 0, |
| 069 | (**int**)win32API.Keys.KEY\_READ, |
| 070 | **out** newKey); |
| 071 |  |
| 072 | **return** newKey; |
| 073 | } |
| 074 |  |
| 075 | **private** **void** updateKeys(**UIntPtr** key) |
| 076 | { |
| 077 | **uint** i = 0; |
| 078 | **uint** sbSize = 255; |
| 079 | **StringBuilder** sb = **new** **StringBuilder**((**int**)sbSize); |
| 080 | **List**<**String**> coll = **new** **List**<**string**>(); |
| 081 |  |
| 082 | **while** (**true**) |
| 083 | { |
| 084 | **int** res = win32API.RegEnumKeyEx( |
| 085 | key, |
| 086 | i, |
| 087 | sb, |
| 088 | **ref** sbSize, |
| 089 | **IntPtr**.Zero, |
| 090 | **IntPtr**.Zero, |
| 091 | **IntPtr**.Zero, |
| 092 | **IntPtr**.Zero); |
| 093 |  |
| 094 | **if** (res == 0) |
| 095 | { |
| 096 | coll.Add(sb.ToString()); |
| 097 | i++; |
| 098 | sbSize = 255; |
| 099 | } |
| 100 | **else** |
| 101 | { |
| 102 | **break**; |
| 103 | } |
| 104 | } |
| 105 |  |
| 106 | **foreach** (**String** str **in** coll) |
| 107 | { |
| 108 | Keys.Add(**new** MyReg(str.ToString(), Root,  (Path == *""* ? str.ToString()  : Path + *"\\"* + str.ToString()))); |
| 109 | } |
| 110 | } |
| 111 |  |
| 112 | **private** **void** updateValues(**UIntPtr** key) |
| 113 | { |
| 114 | **uint** i = 0; |
| 115 | **uint** sbSize = 255; |
| 116 | **StringBuilder** sb = **new** **StringBuilder**((**int**)sbSize); |
| 117 | **List**<**String**> coll = **new** **List**<**string**>(); |
| 118 |  |
| 119 | **while** (**true**) |
| 120 | { |
| 121 | **IntPtr** lpType = **IntPtr**.Zero; |
| 122 | **IntPtr** lpData = **IntPtr**.Zero; |
| 123 | **IntPtr** lpсData = **IntPtr**.Zero; |
| 124 | **int** res = win32API.RegEnumValue( |
| 125 | key, |
| 126 | i, |
| 127 | sb, |
| 128 | **ref** sbSize, |
| 129 | **IntPtr**.Zero, |
| 130 | **ref** lpType, |
| 131 | lpData, |
| 132 | **ref** lpсData); |
| 133 |  |
| 134 | **if** (res == 0) |
| 135 | { |
| 136 | **uint** lpType2; |
| 137 | **uint** sizeData = (**uint**)((lpType.ToInt32() < 3 ||   lpType.ToInt32() == 7)  ? lpсData.ToInt32() \* 2  : lpсData.ToInt32()); |
| 138 | **byte**[] data = **new** **byte**[sizeData]; |
| 139 |  |
| 140 | **int** resy = win32API.RegQueryValueEx( |
| 141 | key, |
| 142 | sb.ToString(), |
| 143 | 0, |
| 144 | **out** lpType2, |
| 145 | data, |
| 146 | **ref** sizeData); |
| 147 |  |
| 148 |  |
| 149 | **var** tt = (lpType.ToInt32() >= 3 && lpType.ToInt32() != 7)   ? BitConverter.ToString(data)  : Encoding.Unicode.GetString(data); |
| 150 |  |
| 151 | coll.Add(sbSize == 0 ? *"(По умолчанию)"*  : sb.ToString() + *" Тип: "* +   typeString(lpType) + *" Значение: "*   + tt); |
| 152 | i++; |
| 153 | sbSize = 255; |
| 154 | } |
| 155 | **else** |
| 156 | { |
| 157 | **break**; |
| 158 | } |
| 159 | } |
| 160 |  |
| 161 | **foreach** (**String** str **in** coll) |
| 162 | { |
| 163 | Values.Add(str); |
| 164 | } |
| 165 | } |
| 166 |  |
| 167 | **private** **String** typeString(**IntPtr** type) |
| 168 | { |
| 169 | **int** typeInt = type.ToInt32(); |
| 170 | **switch** (typeInt) |
| 171 | { |
| 172 | **case** (**int**)win32API.Types.REG\_NONE: **return** *"REG\_NONE"*; |
| 173 | **case** (**int**)win32API.Types.REG\_SZ: **return** *"REG\_SZ"*; |
| 174 | **case** (**int**)win32API.Types.REG\_EXPAND\_SZ:  **return** *"REG\_EXPAND\_SZ"*; |
| 175 | **case** (**int**)win32API.Types.REG\_BINARY: **return** *"REG\_BINARY"*; |
| 176 | **case** (**int**)win32API.Types.REG\_DWORD: **return** *"REG\_DWORD"*; |
| 177 | **case** (**int**)win32API.Types.REG\_DWORD\_BIG\_ENDIAN:  **return** *"REG\_DWORD\_BIG\_ENDIAN"*; |
| 178 | **case** (**int**)win32API.Types.REG\_LINK: **return** *"REG\_LINK"*; |
| 179 | **case** (**int**)win32API.Types.REG\_MULTI\_SZ: **return** *"REG\_MULTI\_SZ"*; |
| 180 | **case** (**int**)win32API.Types.REG\_RESOURCE\_LIST:  **return** *"REG\_RESOURCE\_LIST"*; |
| 181 | **case** (**int**)win32API.Types.REG\_FULL\_RESOURCE\_DESCRIPTOR:  **return** *"REG\_FULL\_RESOURCE\_DESCRIPTOR"*; |
| 182 | **case** (**int**)win32API.Types.REG\_RESOURCE\_REQUIREMENTS\_LIST:   **return** *"REG\_RESOURCE\_REQUIREMENTS\_LIST"*; |
| 183 | **case** (**int**)win32API.Types.REG\_QWORD: **return** *"REG\_QWORD"*; |
| 184 | **default**: **return** *""*; |
| 185 | } |
| 186 | } |
| 187 | } |

### Скриншоты приложения



