梦幻诛仙安装更新方案概要设计说明书

|  |  |  |
| --- | --- | --- |
| 版本 | 日期 | 作者 |
| 0.1.0 | 2008年12月8日 | 刘东学 |
| 0.1.1 | 2008-12-17 | 刘东学 |
| 0.1.2 | 2008-12-18 | 刘东学 |

# 静态结构模型

* 1. 打包工具(PackZip)

打包工具完成程序和资源文件的打包过程，其结果是安装工具的输入。

打包工具的输出分2种格式

* + 1. 安装包(data.pfs)
    2. 资源包(xxxx.pfs)

它同时负责生成安装工具所需要的安装包中的文件元信息。

TODO:详细说明

* 1. 安装工具(Install)

安装工具负责将打包工具生成的安装包(data.pfs)安装到用户目录上。

它同时负责生成目标运行时目录的版本信息和文件元信息。这些信息供更新程序使用。

TODO:详细说明

* 1. 安装文件结构(pfs)

TODO:详细说明

* 1. 文件元信息(.files.meta)

TODO:详细说明

* 1. 版本信息定义(.version.meta)

.version.meta文件为XML格式。示例如下:

<data>

<child version=”1.0.0” url=”http://172.16.x.x/mhzx/gameclient” hash=”xxxxxxxxx”/>

</data>

* 1. 卸载工具(Uninstall)

TODO:卸载工具详细说明

# 动态更新模型

* 1. 差异更新

为了支持动态更新，启动和加载过程需要特殊设计。现确定方案如下:

Launcher

Patcher

GameClient

Patcher.1

Launcher和Patcher模型

上图中Launcher.1为Launcher的临时版本。它由Launcher拷贝Patcher而生成。

策略如下:

Patcher为旧版本或当前唯一版本。

启动程序复制Patcher目录到Patcher.1中。

然后Launcher启动Patcher.1为目录下的Patcher.exe。

Patcher目录游戏客户端原始安装目录，同时也是更新程序从服务器上下载的新Patcher的目录。

这种方案不许要对更新程序的更新逻辑做特殊处理，

即所有Patcher的新程序都将放置在Patcher目录下，而非Patcher.1目录下。

* 1. 差异下载和安装

差异下载和安装过程由Patcher完成。

它从本地运行目录读取当前安装的版本，获取更新服务器的url，从更新服务器读取新版本信息，确定是否有新版本。若有新版本存在，再下载需要更新的文件列表，下载文件，替换本地文件。

* 1. 完全更新

Launcher负责提供一个强制完全更新支持。

当本地版本过旧时，或差异更新出现异常时，可由用户执行完全更新。

完全更新模式等价于一般意义的在线安装。

用户只需下载一个很小在线安装程序，执行该程序由其完成安装程序的下载过程，并启动安装程序完成安装过程。

从下载效率上讲，现在流行的网络下载程序Flashget/迅雷/电驴等依靠多线程下载和断点续传等能力备受用户青睐，除非我们能够提供更高效率的在线安装程序，且一定要做到断点续传功能方可以为用户提供此程序，否则费力不讨好。

* 1. 差异包更新

# 更新服务器

更新服务器架设在Linux的Web Server上。

更新程序通过Http请求连接并下载更新包。

更新服务器的url由打包程序生成并保存到.version.meta文件中.

更新服务器上的文件以分散的目录结构存放，但文件内容可能为压缩格式。

更新程序将文件从服务器下载到本地临时目录中，执行Translate过程，将压缩格式的文件还原为非压缩格式，放置到运行目录中。

运行目录中的.pfs包都对应于更新服务器上的一个目录。

# 国际化

## 中文URL问题和解决方案

打包工具使用936代码转换页将路径转换成ANSI字符串(咱们的系统环境都为GBK编码)，打包进pfs(zip)包，置文件系统属性为0x0B14 - 0x0B含义NTFS, 0x14指代版本2.0。其结论是：

A. 使得Windows平台下的WinRAR可以完全识别中文路径，并且无障碍解包，方便策划部门测试安装包，避免我们为不兼容的pfs包写Windows下的解包程序。

B. 使得Linux平台下的unzip命令行工具可以完全无障碍解pfs包，且解包过程不依赖于Linux环境的locale环境.

当Linux的locale环境设成zh\_CN.GBK之后，中文路径可以正常显示。

最重要的一点是，解包后的文件可以通过中文URL经GBK编码后无障碍访问下载。

IE/FireFox浏览器缺省将非ASCII字符编码成UTF8方式发送到Web服务器，不经修改，IE/FireFox无法下载含中文的URL资源，改成不使用URL访问后才可以下载。

C. 现已经编译并生成了Windows平台下的unzip命令行工具，它也可以无障碍解pfs包，解包后的文件完全正确。

下面列出与中文字符集和国际化有关的代码页标识符ID

Simplified Chinese Code Page Identifiers:

936 ANSI/OEM - Simplified Chinese (PRC, Singapore)

10008 MAC - Simplified Chinese (GB 2312)

20936 Simplified Chinese (GB2312)

50227 ISO 2022 Simplified Chinese

50935 Simplified Chinese Extended and Simplified Chinese

50936 Simplified Chinese

51936 EUC - Simplified Chinese

52936 HZ-GB2312 Simplified Chinese

54936 Windows XP: GB18030 Simplified Chinese (4 Byte)

Unicode Code Page Identifiers:

1200 Unicode UCS-2 Little-Endian (BMP of ISO 10646)

1201 Unicode UCS-2 Big-Endian

12000 Unicode UCS-4 Little-Endian

12001 Unicode UCS-4 Big-Endian

65000 Unicode UTF-7

65001 Unicode UTF-8

## 相关参考、研究、测试、代码和结论：

//

// Code Page Default Values.

//

#define CP\_ACP 0 // default to ANSI code page

#define CP\_OEMCP 1 // default to OEM code page

#define CP\_MACCP 2 // default to MAC code page

#define CP\_THREAD\_ACP 3 // current thread's ANSI code page

#define CP\_SYMBOL 42 // SYMBOL translations

#define CP\_UTF7 65000 // UTF-7 translation

#define CP\_UTF8 65001 // UTF-8 translation

//

// Country/Region Codes.

//

#define CTRY\_DEFAULT 0

**#define CTRY\_PRCHINA 86 // People's Republic of China**

IsValidCodePage

The **IsValidCodePage** determines whether a specified code page is valid.

**BOOL IsValidCodePage(**

**UINT** *CodePage* // code page

**);**

Parameters

*CodePage*

[in] Specifies the code page to check. Each code page is identified by a unique number.

Return Values

If the code page is valid, the return values is a nonzero value.

If the code page is not valid, the return value is zero.

Remarks

A code page is considered valid only if it is installed in the system.

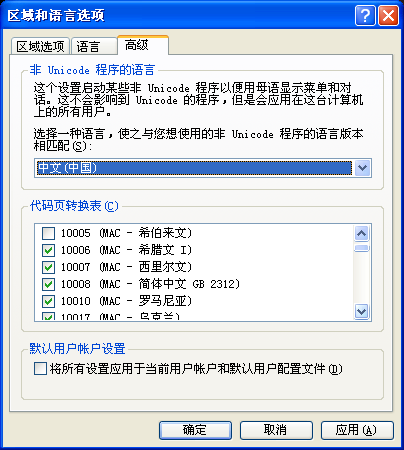
See [Code Page Identifiers](ms-help://MS.MSDNQTR.v80.en/MS.MSDN.v80/MS.WIN32COM.v10.en/intl/unicode_81rn.htm) for a list of code pages.

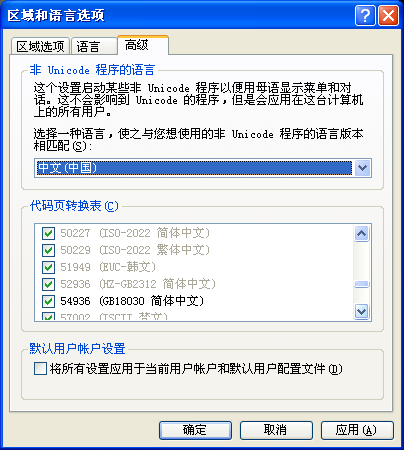
**Windows 95/98/Me:** **IsValidCodePage** is supported by the Microsoft Layer for Unicode. This version of the API also supports CP\_UTF7 and CP\_UTF8. To use this version, you must add certain files to your application, as outlined in Microsoft Layer for Unicode on Windows 95/98/Me Systems.

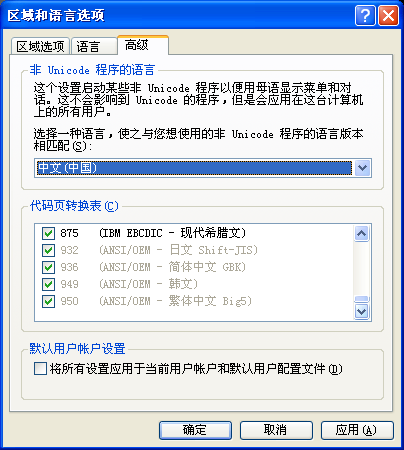
Requirements

**Windows NT/2000/XP:** Included in Windows NT 3.1 and later.  
**Windows 95/98/Me:** Included in Windows 95 and later.  
**Header:** Declared in Winnls.h; include Windows.h.  
**Library:** Use Kernel32.lib.

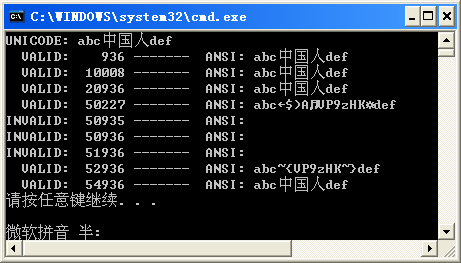
在我的Windows XP操作系统上，语言配置信息如下：







测试UNICODE中文字符串使用指定的Code Page进行转换，结果为：



相关代码：

namespace URLTEST

{

class CStr

{

public:

static bool WideToMutilByte( const std::wstring& \_src, std::string& des, UINT cp = CP\_ACP )

{

des.resize( \_src.size());

int len = ::WideCharToMultiByte( cp, 0, \_src.c\_str(), (int)\_src.size(), const\_cast<LPSTR>( des.c\_str()), (int)des.size(), 0, FALSE);

bool bFailed = ( 0 == len);

if (bFailed)

{

if( ::GetLastError() == ERROR\_INSUFFICIENT\_BUFFER)

{

int nBufSize = WideCharToMultiByte( cp, 0, \_src.c\_str(), (int)\_src.size(), NULL, 0, 0, FALSE);

if( 0 == nBufSize)

return false;

des.resize( nBufSize);

bFailed = ( 0 == ::WideCharToMultiByte( cp, 0, \_src.c\_str(), (int)\_src.size(), const\_cast<LPSTR>( des.c\_str()), nBufSize, 0, FALSE));

}

}

else

{

des.resize( len);

}

return !bFailed;

}

static bool MutilByteToWide( const std::string& \_src, std::wstring& des, UINT cp = CP\_ACP)

{

des.resize( \_src.size() \* 2);

int len = ::MultiByteToWideChar( cp, 0, \_src.c\_str(), (int)\_src.size(), const\_cast<LPWSTR>( des.c\_str()), (int)des.size());

bool bFailed = ( 0 == len);

if (bFailed)

{

if( ::GetLastError() == ERROR\_INSUFFICIENT\_BUFFER)

{

int nBufSize = MultiByteToWideChar( cp, 0, \_src.c\_str(), (int)\_src.size(), NULL, 0);

if( 0 == nBufSize)

return false;

des.resize( nBufSize);

bFailed = ( 0 == ::MultiByteToWideChar( cp, 0, \_src.c\_str(), (int)\_src.size(), const\_cast<LPWSTR>( des.c\_str()), nBufSize));

}

}

else

{

des.resize( len);

}

return !bFailed;

}

}

void StringTranslationTest(const std::wstring& strW, UINT cp = CP\_ACP )

{

std::string strA;

CStr::WideToMutilByte( strW, strA, cp);

std::cout << "ANSI: " << strA << std::endl;

}

void CodePageTest()

{

UINT rgCodePage [] =

{

936 ,

10008,

20936,

50227,

50935,

50936,

51936,

52936,

54936

};

std::wstring strW = L"abc中国人def";

std::wcout << L"UNICODE: " << strW << std::endl;

for ( int i = 0; i < \_countof(rgCodePage); i ++ )

{

if( IsValidCodePage( rgCodePage[i] ) )

{

std::cout << " VALID: " << std::setw(6) << rgCodePage[i] << " ------- ";

}

else

{

std::cout << "INVALID: " << std::setw(6) << rgCodePage[i] << " ------- ";

}

StringTranslationTest(strW, rgCodePage[i] );

}

}

void Test()

{

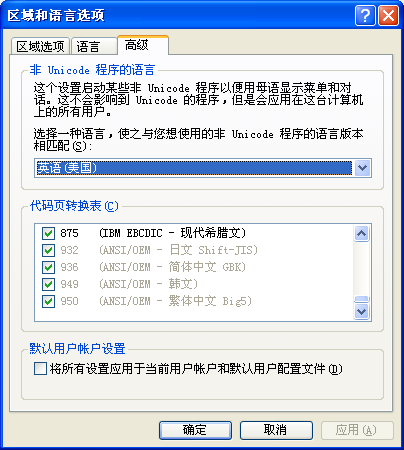
std::wcout.imbue(std::locale("chs"));

CodePageTest();

}

};// namespace URLTEST

当我将操作系统的默认语言更改为英语(美国)并重起计算机之后：



测试结果如下：

下面是控制台窗口的输出内容：

Microsoft Windows XP [Version 5.1.2600]

(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\liudongxue>chcp

Active code page: 437

C:\Documents and Settings\liudongxue>chcp 936

Invalid code page

C:\Documents and Settings\liudongxue>D:

D:\>cd projects\XiaoPang\bin

D:\projects\XiaoPang\bin>pfs

UNICODE: abc╓╨╣·╚╦def

VALID: 936 ------- ANSI: abc╓╨╣·╚╦def

VALID: 10008 ------- ANSI: abc╓╨╣·╚╦def

VALID: 20936 ------- ANSI: abc╓╨╣·╚╦def

VALID: 50227 ------- ANSI: abc←$)A♫VP9zHK☼def

INVALID: 50935 ------- ANSI:

INVALID: 50936 ------- ANSI:

INVALID: 51936 ------- ANSI:

VALID: 52936 ------- ANSI: abc~{VP9zHK~}def

VALID: 54936 ------- ANSI: abc╓╨╣·╚╦def

D:\projects\XiaoPang\bin>pfs > xxx.txt

D:\projects\XiaoPang\bin>chcp 20936

Active code page: 20936

D:\projects\XiaoPang\bin>pfs

UNICODE: abc╓╨╣·╚╦def

VALID: 936 ------- ANSI: abc╓╨╣·╚╦def

VALID: 10008 ------- ANSI: abc╓╨╣·╚╦def

VALID: 20936 ------- ANSI: abc╓╨╣·╚╦def

VALID: 50227 ------- ANSI: abc←$)A♫VP9zHK☼def

INVALID: 50935 ------- ANSI:

INVALID: 50936 ------- ANSI:

INVALID: 51936 ------- ANSI:

VALID: 52936 ------- ANSI: abc~{VP9zHK~}def

VALID: 54936 ------- ANSI: abc╓╨╣·╚╦def

D:\projects\XiaoPang\bin>type xxx.txt

UNICODE: abc╓╨╣·╚╦def

VALID: 936 ------- ANSI: abc╓╨╣·╚╦def

VALID: 10----- ANSI: abc╓╨╣·╚╦def

VALID: 20936 ------- ANSI: abc╓╨╣·╚╦def

VALID:7 ------- ANSI: abc←$)A♫VP9zHK☼def

INVALID: 50935 ------- ANSI:

INVALID: 50936 ------- ANSI:

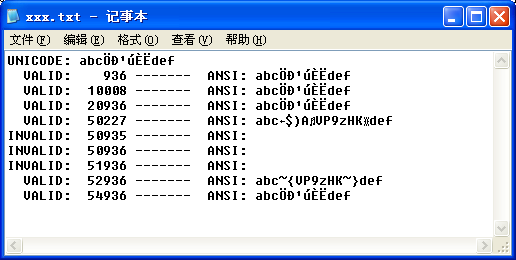
INVALID: 51936 ------- ANSI:

VALID: 52936 ------- ANSI: abc~{VP9zHK~}def

VALID: 54936 ------- ANSI: abc╓╨╣·╚╦de

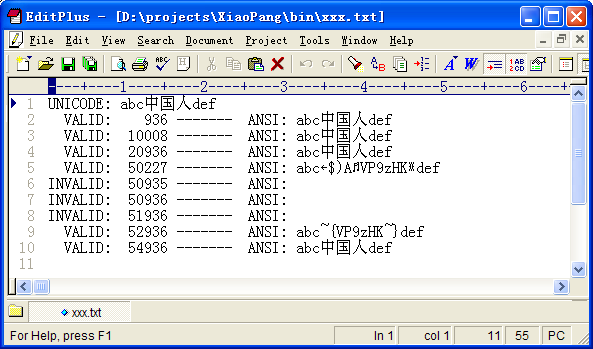
对于xxx.txt:

使用记事本打开看到的结果是（它仅支持ANSI\UNICODE\UTF8\UTF7等代码页，所以无法正常显示中文字符）：

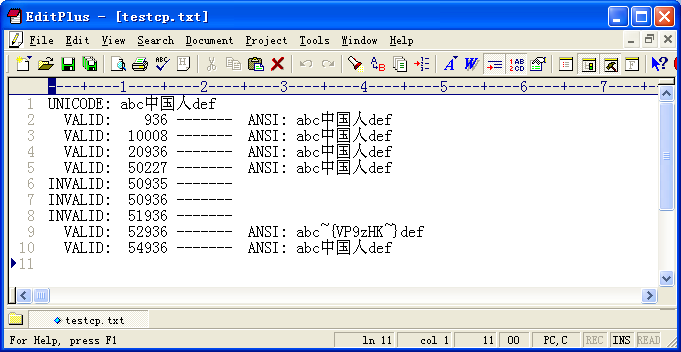


使用EditPlus打开看到的结果是正确的:

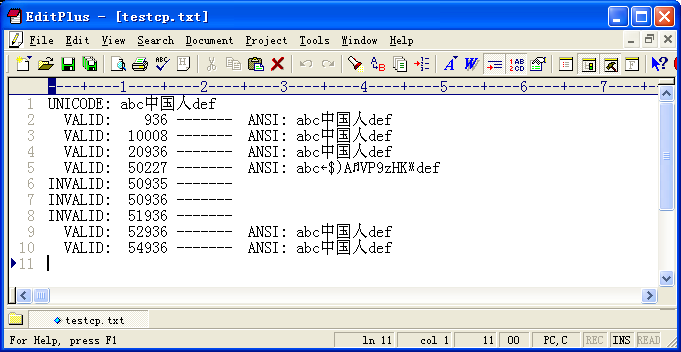
Load As ANSI(GBK936)



Reload AS 50227:



Reload AS 52936:



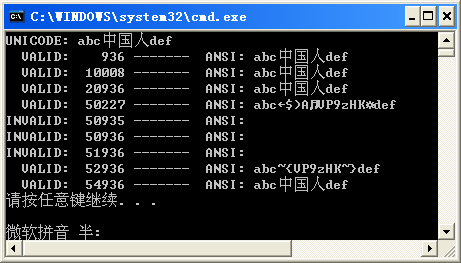
这里汇总以下字符集对应的常见名称及其显示效果：

GB18030 CP54936 Simplified Chinese - GB18030

GB2312 CP20936 Simplified Chinese - GB2312

GBK CP936 Simplified Chinese - GBK

HZ CP52936 Simplified Chinese - HZ-GB2312



可见50227以及52936(HZ-GB2312)和GBK/GB2312/GB18030并不是完全兼容的编码方案。

问题：

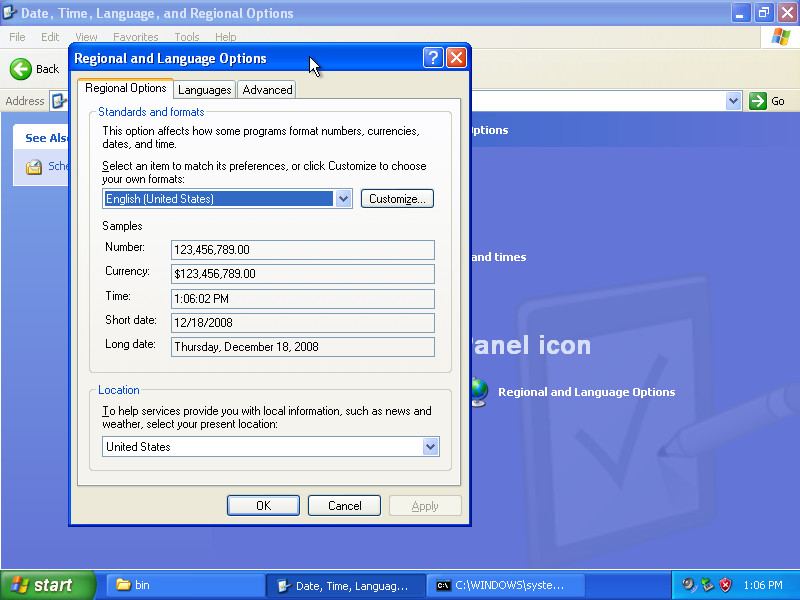
不知道Windows非中文版操作系统是否已安装936代码页。接下来需要针对纯英文操作系统进行测试。

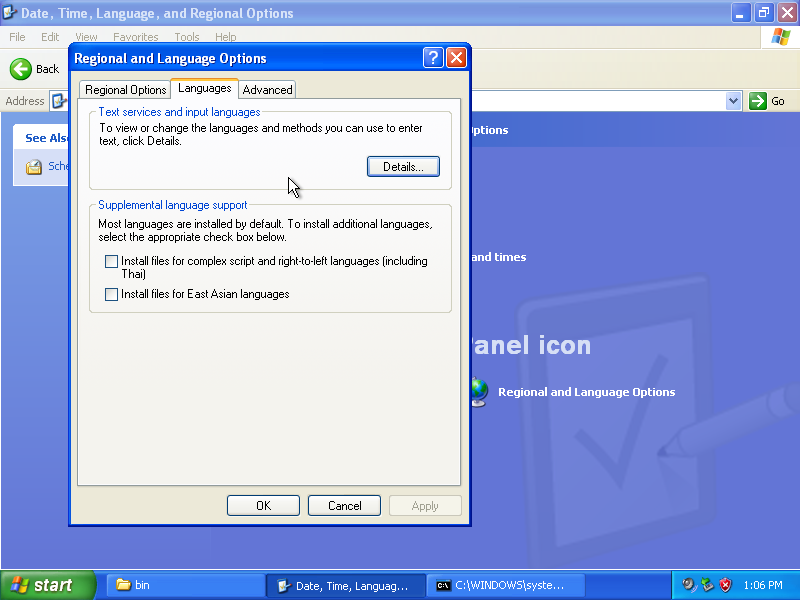
答案：

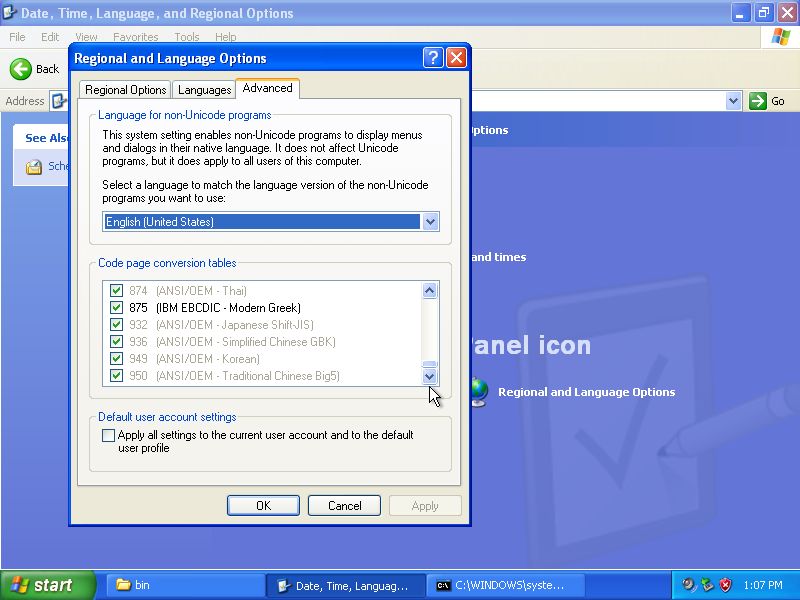
经测试，英文版操作系统也强制安装936代码页。测试结果如下：

通过Virtual Machine方式截屏得到：

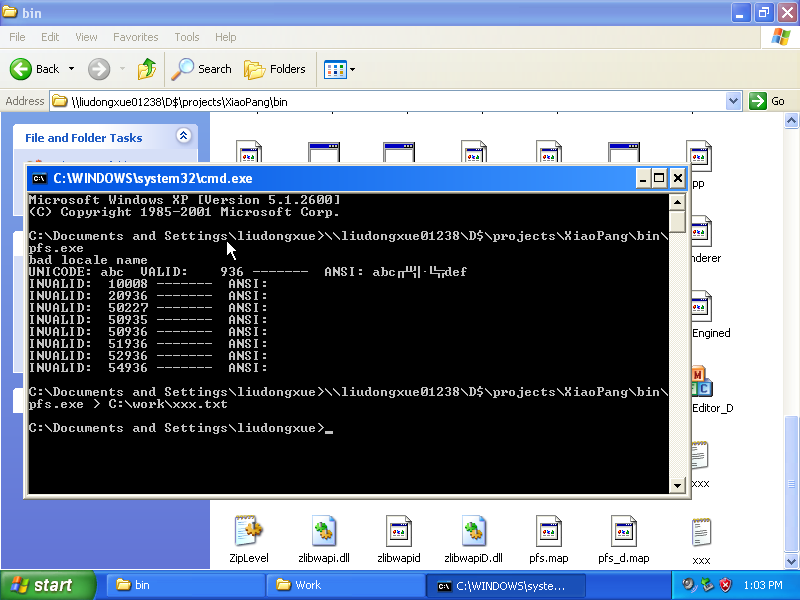
区域和语言配置信息：







执行pfs.exe测试程序:



下面的xxx.txt是重定向pfs.exe输出的内容：



可见，中文多字节处理在936字符集下是没有问题的。

测试程序需要修改一点：

纯英文操作系统不支持chs locale，测试程序在执行时会抛出bad locale name runtime error.所以需要捕获之。

void Test()

{

try

{

std::wcout.imbue(std::locale("chs"));

}

catch( std::runtime\_error re )

{

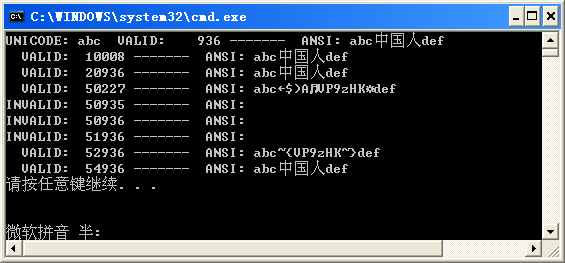
std::wcout<<re.what() << std::endl;

}

CodePageTest();

}

即使在中文操作系统下，如果不置locale为chs，那么UNICODE字符串所包含的中文在输出时将被丢弃。结果如下：



这与在英文操作系统中没有看到中文字符的情况是一致的。

这只会影响标准C++ I/O Stream的输出行为。不影响字符串处理。

将测试程序稍加修改:

void Test()

{

try

{

//std::wcout.imbue(std::locale("chs"));

std::wcout.imbue(std::locale(".936"));

}

catch( std::runtime\_error re )

{

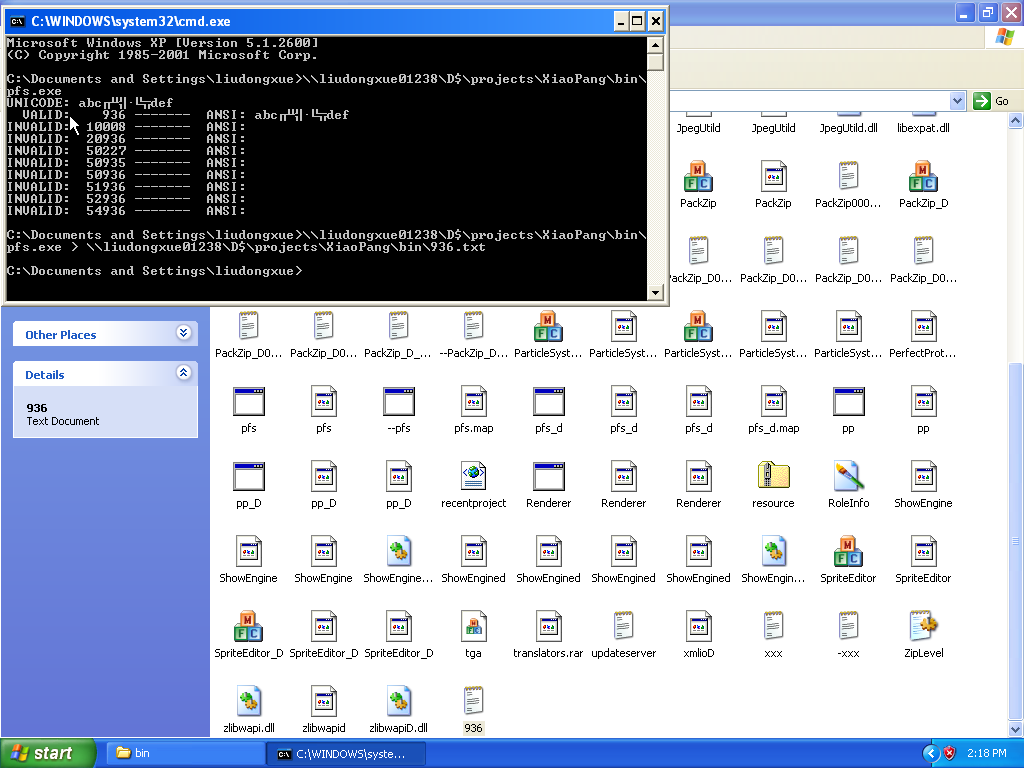
std::wcout<<re.what() << std::endl;

}

CodePageTest();

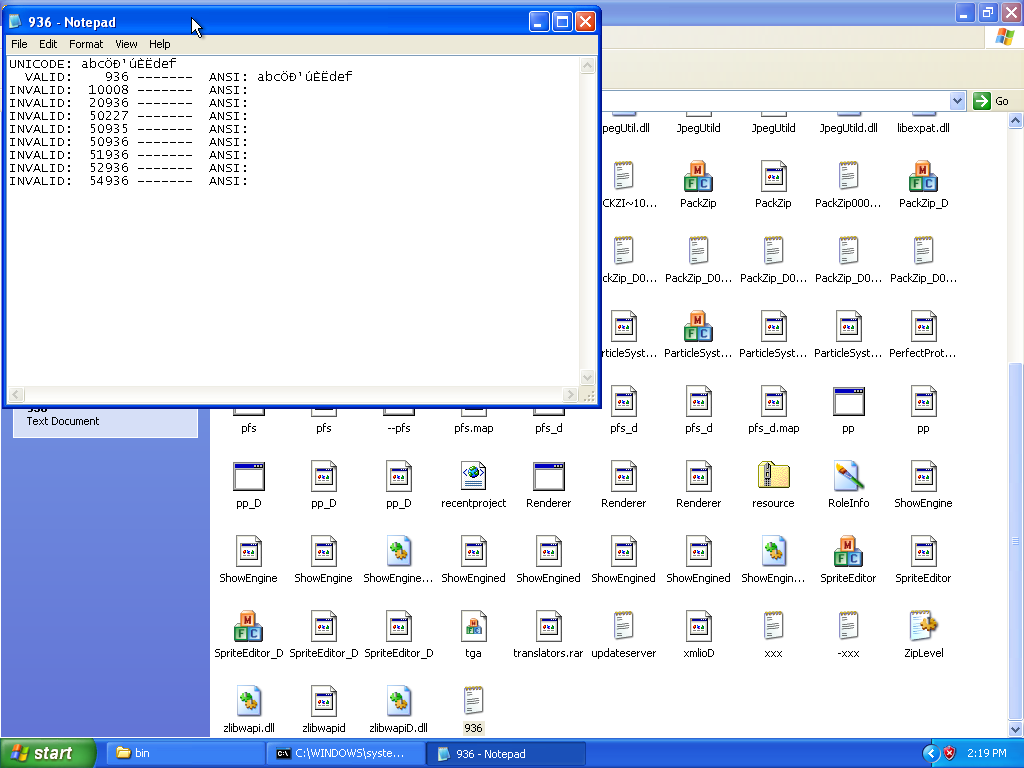
}

在英文操作系统下的输出结果将完全正确:



下面是重定向得到的文本文件在记事本下所显示的效果:

在纯英文操作系统下:



在中文操作系统下:



结论

针对中文路径名、中文URL的国际化方案为：

对于GB2312编码的中文字符串，如果要保证成功转成UNICODE(UTF16-LE)或反向转换成功，需指定codepage为936,而非CP\_ACP(0). 相关API如下：

WideCharToMultiByte

MultiByteToWideChar

对于控制台显示和输出问题,可以通过设定locale为”.936”而非”chs”来保证数据输出时不丢失数据。相关I/O对象和函数如下：

std::wcout.imbue(std::locale(".936"));

std::cout.imbue(std::locale(".936"));

std::setlocale(".936");