MIS 版本自动维护系统的设计

邓 鹰 （湖南省娄底师专 417000）

摘要： 本文针对Client/Server模式下的大型管理信息系统，详细论述了其应用软件系统实现版本自动维护（或者系统自动升级）的设计思路，及其实现方法。

关键词： C/S结构 MIS系统 应用系统版本 自动维护

1 前言

在我们设计、开发一个管理信息系统（MIS）时，尤其对于一个大型MIS系统，其开发工作很难一次完善，在系统的使用过程中，会不断发现新问题、提出新需求，因而需要进行不断的修改、升版。在C/S模式下，应用系统分别安装在Server及Client上，而Client站点往往分布范围较广，这就造成对Client端的维护工作繁杂，系统管理员疲于应付。

为了减轻系统管理员的负担，同时更为了保证使整个企业用户同步使用系统的最新软件版本，在此我们提出了一种让Client端自动进行版本维护的技术方案。

2 设计思想

为了实现Client端的版本自动维护，我们可以在Server端的数据库中新增一个专用数据表，用来保存应用系统的最新版本（即二进制的可执行文件、动态连接库等等）。当系统每次有新的版本编译出来后，可以通过系统维护工具，将最新版本更新到数据库中，而Client端的应用软件，在其每次启动时首先检测自己的版本（INI文件），如果发现有更新的版本存在，则先下载新版本的文件，然后进行版本升级，升级完成后才进入系统并将新的版本信息写入INI文件。其流程如图1所示。

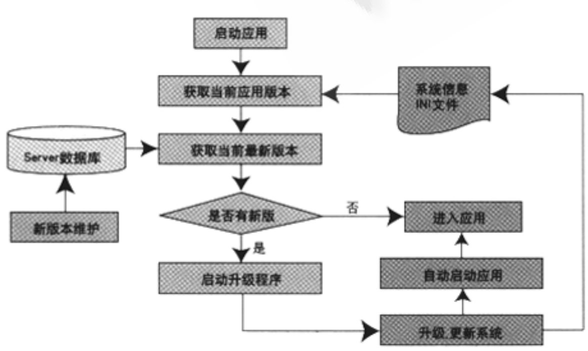

图 1

2.1 Server端设计

实现应用系统的版本自动维护，其Server端的设计，主要是在相应的MIS数据库中增加一个数据表，用来保存应用系统的最新版本，该表只能由系统开发人员（部门）来更新维护。其表(sysAppVersion)结构定义为下表：

序号	域名	域名含义	类型长度	PK	FK	NULL
1	AppName	应用名称	VarChar(32)	Y	N	
2	FileName	应用文件名称	VarChar(32)	Y	N	
3	FileSize	文件大小	Integer			
4	Version	对应的版本号	Decimal(6,2)			
5	FileDate	文件时间	Datetime			
6	FileData	实际文件的二进制数据	Blob			
7	UpDate	维护时间	Datetime			
8	Staff	维护人工号	VarChar(5)			
9	Ws_name	维护使用工作站	VarChar(64)			

表中，AppName、FileName两列组合为主关键字，这是因为一个应用可能包含多个具体文件如可执行文件、动态连接库、以及其他文件（也可能没有）。

2.2 Client端设计

Client端设计则主要是在应用中增加版本检测代码：前端在每次启动时首先检查应用的当前版本，以及数据库sysAppVersion表中相应应用的版本，如果一致，则进入应用之功能操作，如果存在新版本，则进入版本更新处理模块。

3 具体实现

考虑到在实际MIS系统的开发设计中，Powersoft公

司的 PowerBuilder 是最为常用的开发工具，这里就以 PowerBuilder 为环境，给出了应用系统中版本自动维护的具体实现。

3.1 应用软件的设计

3.1.1 进行全局变量的定义

```
gs_AppName: string // 保存当前应用名称
gs_AppPath: string // 保存当前应用的工作路径
gs_IniFile: string // 保存当前应用的初始化文件名称
```

3.1.2 申明外部函数

```
function long ExitWindowsEx(long uFlag, long dwReserved) library "user32.dll"
function long GetWindowsDirectoryA(ref string lpBuffer, long nSize) library "user32.dll"
```

3.1.3 编写自定义函数

```
function decimal f_CheckVersion()
decimal ld_ver1, ld_ver2
// 从系统 INI 文件中获取当前本地的软件版本
ld_ver1 = Dec(ProfileString(gs_IniFile, gs_AppName, version, ""))
SELECT MAX(version) INTO :ld_ver2 FROM sysAppVersion
WHERE AppName = :gs_AppName;        //获取数据库中的软件版本
If IsNull(ld_ver2) then ld_ver2 = 0.0
if ld_ver2 > ld_ver1 then
  return ld_ver2   // 需要更新
else
  return 0 // 不需要更新
// End of function f_CheckVersion
```

3.1.4 在应用启动的 open 事件中增加脚本

```
decimal ld_ver
integer rlt
ld_ver = f_CheckVersion()   // 检查版本
if ld_ver > 0 then
rlt = MessageBox(" 提示 ", " 当前应用系统已经发布了新版本(" + string(ld_ver) + "), 需要更新吗？ ", Question!, YesNo!, 1)
    if rlt = 1 then open(w_upgrade)
end if
```

3.1.5 设计一个窗体（w_upgrade）

窗体 w_upgrade 的界面设计如图 2：

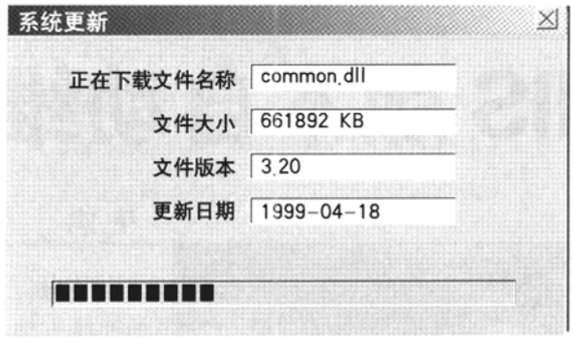

图 2

3.1.6 在窗体 w_upgrade 中定义 Instance 变量：string is_fname []

3.1.7 在窗体 w_upgrad 中定义 wf_SaveFile(string as_fname, blob ab_data) 函数

```
long I, num, lh_file
blob lb_data

lh_file = FileOpen(as_fname, StreamMode!, Write!, LockWrite!, Replace!)
num = Ceiling(len(ab_data) / (64 * 1024))  //计算循环次数，每次读取 64K 字节
for I = 1 to num
  ll_pos  = (I - 1) * 64 * 1024
  lb_data = BlobMid(ab_data, ll_pos, 64 * 1024)
  FileWrite(lh_file, lb_data)
Next
FileClos(lh_file)
```

3.1.8 在窗体 w_upgrade 的 open 事件中增加脚本

```
string ls_fname
long ll_fsize, i
decimal ld_fver
datetime ldt_fdate
blob lb_data
DECLARE c_app CURSOR FOR
SELECT FileName, FileSize, Version, VerDate FROM sysAppVersion
WHERE AppName = :gs_AppName;
OPEN c_app;
FETCH c_app INTO :ls_fname, :ll_fsize, :ld_fver, :ldt_fdate;
i = 1
```

```
do while SQLCA.SQLCode = 0
is_fname [i]    = ls_fname
st_fname.text = ls_fname
st_fsize.text = string(ll_fsize)
st_fver.text  = string(ld_fver)
st_fdate.text = string(ldt_fdate, "yyyy-mm-dd")
SELECTBLOB BinData INTO :lb_data FROM sysAppVersion
WHERE AppName = :gs_AppName AND FileName = :ls_fname;
wf_SaveFile(ls_fname, lb_data)   // 保存到文件
i = i + 1
FETCH c_app INTO :ls_fname, :ll_fsize, :ld_fver, :ldt_fdate;
loop
CLOSE c_app;
```

3.2 版本维护设计

版本维护主要是将系统的最新代码上传到数据库内。一般来说，版本维护程序和自动升级程序都会集中在一个应用系统中，只不过版本维护只对系统管理员或者开发组成员授权使用，而自动升级程序则将在任何一台工作站当系统启动时自动执行。

版本维护的功能主要提供增加、删除功能，其界面设计大致如图 3 所示。

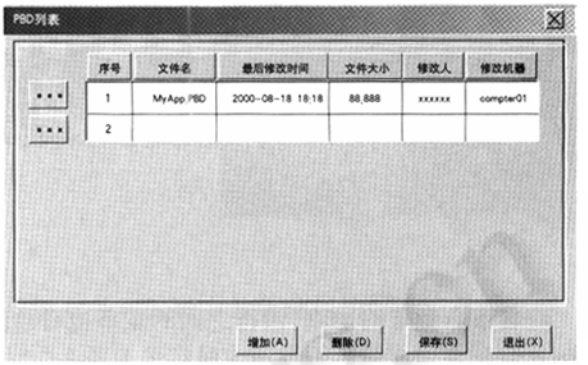

图 3

4 结束语

以上所述，已经基本实现了大型应用系统的版本自动升级管理问题。但如果一个系统非常庞大的时候，简单的文件上载、下载功能可能仍无法满足要求，比如当客户端是通过拨号方式与服务器连接时，升级将耗费大量的代码传输时间，这时，我们还应当考虑将代码文件压缩，压缩处理可以采用高效的 ZIP，或者 WINDOWS 内含的 COMPRESS 函数，有兴趣的读者可以试试。■