

实验名称： 操作系统综合实验

Flower Manager 资源管理器

学 院： 计算机学院

专 业： 计算机科学与技术

班 级： 07111503

组 长： 1120151764 陈牧乔

小组成员： 1120151769 侯思凡

1120151775 罗 薇

1120151782 王紫薇

指导教师： 王全玉

日 期： 2018 年 5 月 10 日

目录

一、	实验目的.....	3
二、	实验内容.....	3
1.	项目介绍.....	3
2.	阶段说明.....	4
三、	实验环境及方法.....	4
四、	实验方法和实验步骤.....	5
1.	总框架的搭建.....	5
2.	实现菜单功能.....	5
3.	“进程”信息查看的实现.....	7
4.	“性能”信息查看的实现.....	8
5.	“网络”信息查看的实现.....	10
6.	“文件浏览”及文件复制的实现.....	14
7.	“文件清理”的实现.....	17
8.	“详细信息”页的实现.....	18
9.	“内存”信息查看的实现.....	20
五、	实验结果和分析.....	21
1.	总框架.....	21
2.	“菜单”和最小化到托盘.....	21
3.	“进程”.....	23
4.	“性能”.....	23
5.	“网络”.....	24
6.	“文件浏览”.....	24
7.	“文件清理”.....	27
8.	“详细信息”.....	30
9.	“内存”.....	32
六、	讨论、心得.....	33
1.	实验中遇到的问题.....	33
2.	小组成员的心得体会.....	37

<侯思凡>

一、 实验目的

1. 熟悉了解操作系统相关知识，包括进程控制、进程通信、并发控制、虚拟内存机制、文件操作等内容；
2. 熟悉了解 MFC（微软基础类库），能够结合相关知识搭建项目框架；
3. 设计并实现一个具有独立功能的程序，要求设计操作系统相关原理；

二、 实验内容

1. 项目介绍

(1) 简介

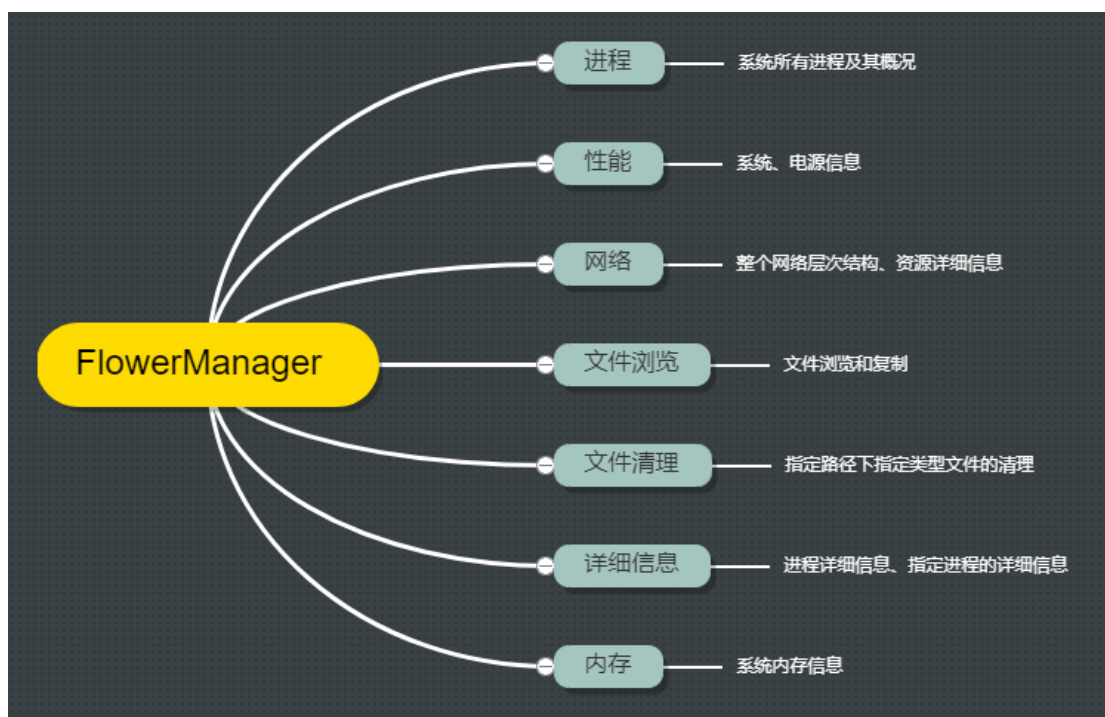


Figure 1

(2) 模块及其功能说明

Table 1 主模块

编号	模块名	功能描述
0	资源管理器	在图形界面实现浏览页面的切换，同时调用相应的子模块实现子模块间的切换，以此实现了用户切换标签页时，能够查看相应模块的信息，并进行该模块下需要的操作；同时提供退出、置顶/取消置顶、弹出“关于”框和最小化到托盘并允许恢复的功能。

Table 2 子模块

编号	模块名	功能描述
----	-----	------

1	进程	查看系统所有进程，查看其 ID、父进程、线程数等信息。
2	性能	查看有关系统和电源的性能信息；为实现实时查看，提供刷新页面功能。
3	网络	查看当前整个网络清晰的层次结构；并且提供关于网络的更多详细信息；为实现实时查看，提供刷新页面功能。
4	文件浏览	查看系统的文件系统，能够体现清晰的层次结构；提供查看文件属性的功能；能够实现指定目录（包含该目录下的文件及嵌套目录等）的复制和删除。
5	文件清理	根据用户指定路径和指定文件类型清理相应类型的文件。
6	详细信息	查看系统进程的详细信息，包括 PID、内存、描述信息等；为实时监控提供刷新功能；查看指定进程的虚拟内存分配信息；结束指定进程。
7	内存	查看系统内存内存，包括物理内存使用率、实际物理总内存、当前可用物理内存等信息；为实现实时监控提供刷新功能。

2. 阶段说明

（1）项目构思

第一阶段，选题与设计。经小组协商，根据目前所掌握的语言及擅长方向，结合操作系统课程设计的课题要求，小组决定采用 C++ 语言，运用 MFC 框架，搭建一个资源管理器平台，该平台实现进程查看、进程控制、内存监视、文件处理等功能。

（2）框架搭建

第二阶段，完成项目总框架的搭建。根据项目设计，首先搭建项目框架，主模块用于实现子模块调度功能，完成七个子模块间的切换。故在总框架搭建阶段，设计并完成了主对话框和七个子对话框的 Dialog 控件布局。

其中，主对话框 FlowerMangerDlg 内使用 MFC 的 Tab Control 控件，该控件将七个子对话框填充进主对话框中，以实现窗口的切换；由于七个子对话框间相互独立，故将在第三阶段独立地实现各个子模块的功能。

（3）模块实现

第三阶段，小组成员对 7 个模块进行独立实现。

（4）模块集成

第四阶段，将独立的七个子模块与主模块进行集成。由于小组成员在项目第三阶段基于总框架进行七个子模块的独立实现，子模块间相互独立。集成阶段，对资源文件进行重新复制构建，尤其严格控件 ID 与小组成员自行设计的模块中数据交互的 ID 号匹配；功能实现部分，沿用每个成员的 C++ 代码即可。

三、 实验环境及方法

1. Windows 版本 : Windows10 企业版
2. 处理器 : Intel (R) Core (TM) i5-5200U CPU @2.20GHz 2.20GHz
3. 已安装的内存 : 4.00GB

- 4. 系统类型 : 64 位操作系统, 基于 x86 的处理器
- 5. 集成开发环境 : Visual Studio 2015
 - (1) 解决方案配置 : Debug
 - (2) 解决方案平台 : x64

</侯思凡>

四、 实验方法和实验步骤

<罗薇>

1. 总框架的搭建

- (1) 总框架整体设计构思

创建一个 MFC 控制台应用项目, 对进程、文件、内存等模块进行管理。仿照 win10 任务管理器的界面: 在主界面上方设几个菜单项, 下方是几个可切换的标签页。
- (2) 新建 8 个 Dialog, 添加标签页

新建一个 Dialog 作为主窗体, 向其中填充一个 Tab Control 控件。除主窗体外, 另新建 7 个 Dialog 作为资源管理器的七个标签页进程、性能、网络、文件浏览、文件清理、详细信息和内存, 并且在主窗体中填充一个 Tab Control 控件用于切换标签页。添加 CTabCtrl 类变量 m_tab, 在 DoDataExchange()中写入 DDX_Control(pDX, IDC_TAB1, m_tab), 将变量 m_tab 和控件 IDC_TAB1 进行绑定。
- (3) 实现标签页的切换

当点击标签页的 Tab 时, 添加事件处理程序 ON_NOTIFY(TCN_SELCHANGE, IDC_TAB1, &FlowerManagerDlg::OnTcnSelchangeTab1), 用 m_tab.GetCurSel()函数获取当前选择的标签页号 nSel, 对 nSel 用 Switch case 语句进行判断, 用 ShowWindow(SW_SHOW)函数显示当前选择页 (如 Process.ShowWindow(SW_SHOW)显示进程页), 用 ShowWindow(SW_HIDE)函数隐藏其他标签页。

</罗薇>

<侯思凡>

2. 实现菜单功能

Table 3 自定义的函数

编号	格式	功能	实现
1	BOOL FlowerManagerDlg::OnInitDialog()	初始化主对话框, 主要工作有对对话框中加入菜单、设置标签名、放置子对话框。	1.插入的标签页的覆盖范围应适合 标签控件大小; 2.子对话框初始时默认显示第一页。
2	void FlowerManagerDlg::OnSysCommand(其中添加了对最小化	

	UINT nID, LPARAM lParam)	到托盘的函数的调用。	
3	void FlowerManagerDlg::OnTcnSelchangeTab1(NMHDR *pNMHDR, LRESULT *pResult)	实现标签页的切换。	先获取当前用户点选的 tab, 根据 tab 选择显示和隐藏哪些页面。
7	void FlowerManagerDlg::OnExit()	响应“退出”项被点击事件。	弹出确认退出的消息框。
8	void FlowerManagerDlg::OnUpdateTop(CCmdUI *pCmdUI)	置顶/取消置顶。	此处定义一个静态全局变量 TopClick, 初值为 0, 表示置顶功能项从未被点击, 该变量计数“置顶(T)”项被用户点击的次数。对 TopClick 进行奇偶判断——奇数次点击将窗体置顶, 偶数次点击取消置顶。
9	void FlowerManagerDlg::ToTray()	最小化到托盘: 当用户点击最小化按钮, 对话框隐藏, 并且在托盘区显示 Flower Manager 的 logo, 设计鼠标滑过该图标时显示程序的名字。	1.NOTIFYICONDATA 结构体的成员设置实现在托盘区显示图标即项目名; 2.隐藏当前窗口。
10	LRESULT FlowerManagerDlg::OnShowTask(WPARAM wParam, LPARAM lParam)	托盘区图标右键可选“关闭”程序; 左键双击则正常显示对话框。	接收鼠标对托盘区图标的行为, 对行为进行判断并作出反应——关闭或恢复窗口。
11	void FlowerManagerDlg::DeleteTray()	即程序恢复窗口显示后应删除托盘区的图标。	
12	void FlowerManagerDlg::OnAbout()	弹出“关于 Flower Manager”的对话框, 显示设计人员的信息。	运行关于对话框的模块进程。

</侯思凡>

<王紫薇>

3. “进程”信息查看的实现

这个对话框的内容结合【实验四 内存监视】获取进程的相信信息的内容，查看进程的名称、PID、父进程 ID、线程基本优先级等信息，初步了解 List Control 控件的使用方法，将获取到的进程信息填入其中。

(1) 流程图

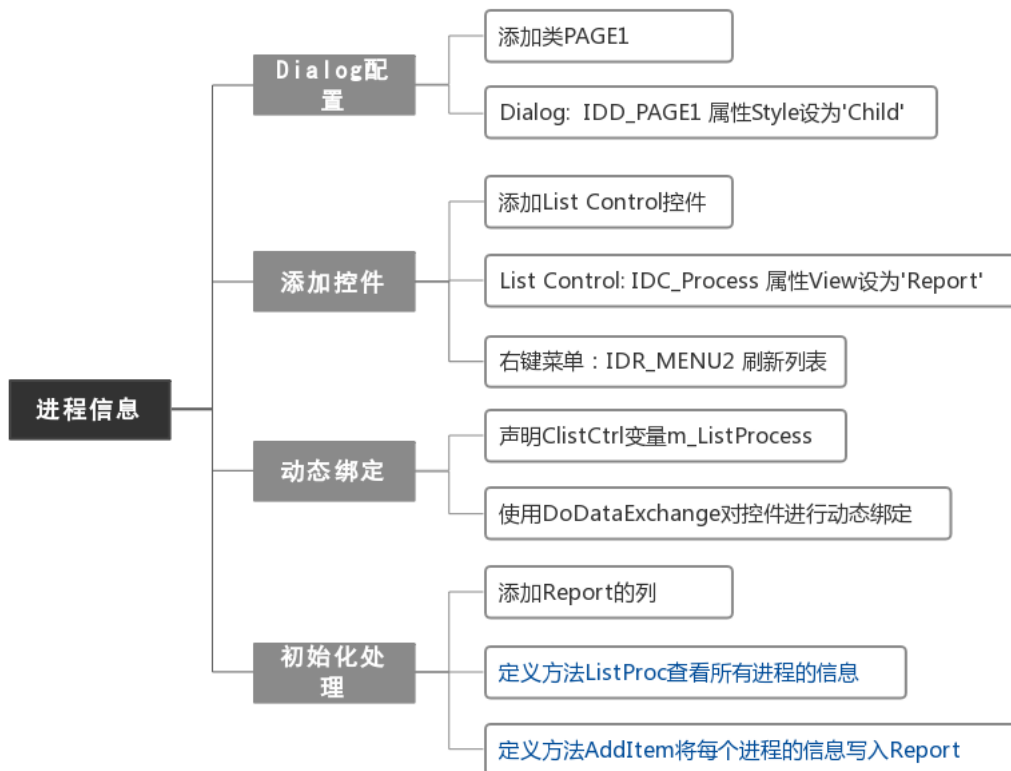


Figure 2 进程信息

(2) 函数说明

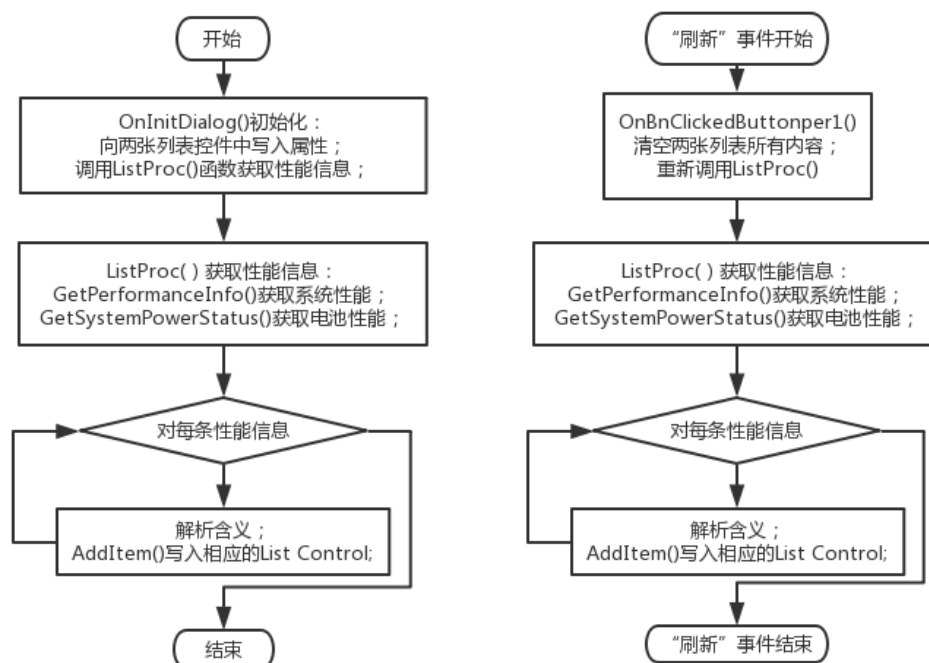
- 定义方法 ListProc 查看所有进程的信息
 - 1) 声明进程入口的 PROCESSENTYR 32 结构体变量
 - 2) 获取系统中当前进程的快照
 - 3) 使用系统 API: Process32First 获取系统快照中的第一个信息
 - 4) 循环读取进程的信息，对信息进行字符串处理后并调用 AddItem 函数填入 Report，直到 Process32Next 函数返回空，即当前进程信息全部读完。
- 定义方法 AddItem 将每个进程的信息写入 Report
 - 1) 使用 InsertItem 插入行，并设置第一列数据
 - 2) 使用 SetItemText 设置其他列
- OnRclickListProcess、OnProUpdate 提供右键弹出菜单，刷新列表功能
 - 1) 添加 Menu 资源 IDR_MENU2
 - 2) 添加刷新选项 ID_PRO_Update
 - 3) 添加右键响应函数 OnRclickListProcess，获取鼠标位置后弹出子菜单
 - 4) 添加命令 OnProUpdate，点击属性选项后进入该函数，清空当前 List Control 中所有项目并重新载入

</王紫薇>

<侯思凡>

4. “性能”信息查看的实现

1)流程:



“性能”页信息填充大致如上图，程序初始化列表控件，调用用系统 API 获取信息并将信息填入列表中。

另外，“刷新”按钮响应后删除列表中所有内容，大致以类似于上图的流程充填列表项。

2)函数说明:

Table 4 自定义函数

编号	格式	功能	实现
1	<code>BOOL PAGE2::OnInitDialog()</code>	初始化 PAGE3 对话框，写入系统、电源性能 List Control 的属性。	对基类虚函数的覆盖。
2	<code>void PAGE2::ListProc()</code>	调用获取系统性能、电源信息的 API，获取性能相关信息，判断该信息标志的意义。	1.将数值应转化为表示大小的字符串；2.将性能的含义写入字符串；3.以同一的标准调用向 List 写入信息。
3	<code>void PAGE2::AddItem(bool flag, TCHAR* name, TCHAR* value)</code>	向 List Control 写入性能信息。	根据 flag 标志判断该写入系统和电源哪张表。

4	<pre>void PAGE2::OnBnClickedButtonper1()</pre>	对“刷新”按钮单击的响应函数，重新获取性能信息并反馈给用户。	控件事件。1.当用户点击该按钮后，清空当前两张 List Control 中的所有条目，2.并且重新调用 List Proc() 函数重读性能相关信息并重填 List Control 表格
5	<pre>int CALLBACK PAGE2::MyCompareProc2(LPARAM lParam1, LPARAM lParam2, LPARAM lParamSort) int CALLBACK PAGE2::MyCompareProc2(LPARAM lParam1, LPARAM lParam2, LPARAM lParamSort)</pre>	比较函数，用于列表控件中某一列的排序	能够实现递增/递减交替排序，通过定义布尔型静态全局变量 method1 和 method2，每次点击对变量取反，排列对象分为整型和字符串型。
6	<pre>void PAGE2::OnLvnColumnClickListper(NMHDR *pNMHDR, LRESULT *pResult) void PAGE2::OnLvnColumnClickListpower(NMHDR *pNMHDR, LRESULT *pResult)</pre>	响应点击某列头并对此列进行排序	获取被点击列的索引、行数，并进行比较和排序

Table 5 用到的控件相关函数

编号	格式	功能
1	<pre>int InsertColumn(int nCol, LPCTSTR lpszColumnHeading, int nFormat = LVCFMT_LEFT, int nWidth = -1, int nSubItem = -1);</pre>	向列表视图控件插入新列
2	<pre>int CListCtrl::GetItemCount()</pre>	检索列表视图控件的项的数目
3	<pre>int CListCtrl::InsertItem(int nItem, LPCTSTR lpszItem);</pre>	向列表视图控件插入项
4	<pre>BOOL SetItemText(int nItem, int nSubItem,</pre>	更改列表视图项目或子项的文本

	<code>LPCTSTR lpszText);</code>	
5	<code>BOOL CListCtrl::DeleteAllItems()</code>	从列表视图控件删除所有项
6	<code>CString GetItemText(int nItem, int nSubItem)</code>	获取指定列指定行的内容
7	<code>int CompareNoCase(PCWSTR psz)</code>	不区分大小写比较字符串
8	<code>CListCtrl::SortItems(PFNLVCOMPARE pfnCompare, DWORD_PTR dwData)</code>	对 List Control 的排序

Table 6 用到的系统 API

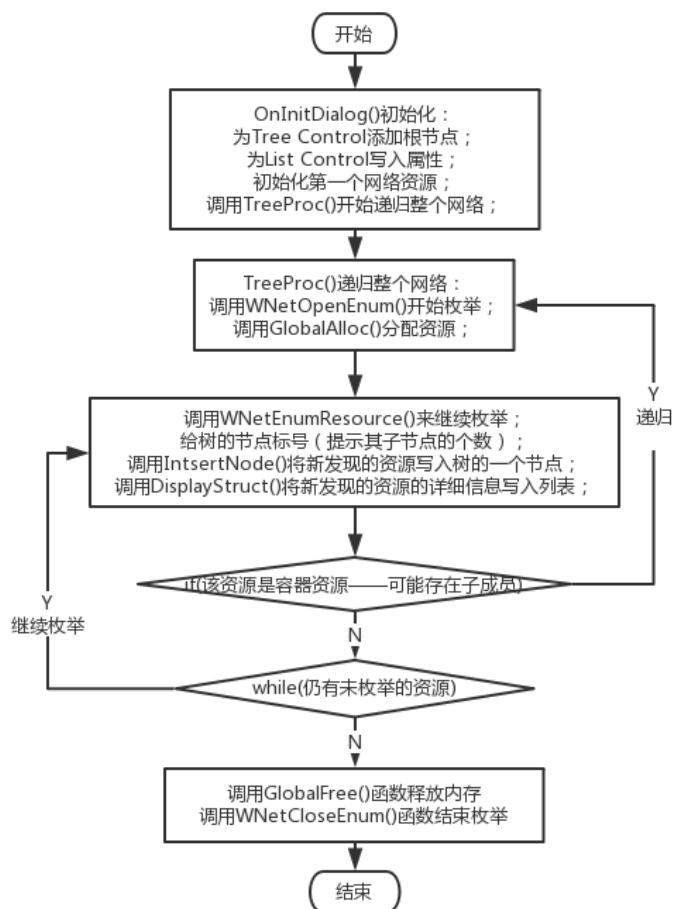
编号	格式	说明
1	<code>BOOL WINAPI GetPerformanceInfo (PPERFORMANCE_INFORMATION pPerformanceInformation, DWORD cb);</code>	该用于函数获取系统性能函数, <code>pi</code> 是一个指向 <code>PERFORMANCE_INFORMATION</code> 结构的指针, 该结构体内的成员表明系统页面、物理内存、Cache 等信息。
2	<code>LWSTDAPI_(PSTR) StrFormatByteSize (DWORD dw, (cchBuf) PSTR pszBuf, UINT cchBuf);</code>	将数字值转换为字符串, 该字符串表示以字节, 千字节, 兆字节或千兆字节为单位的大小值表示的数字, 具体取决于大小。
3	<code>int sprintf(char* const _Buffer, char const* const _Format, ...)</code>	字符串格式化命令, 主要功能是把格式化的数据写入某个字符串中。 <code>sprintf</code> 是个变参函数。使用 <code>sprintf</code> 对于写入 <code>buffer</code> 的字符数是没有限制的
4	<code>BOOL WINAPI GetSystemPowerStatus (LPSYSTEM_POWER_STATUS lpSystemPowerStatus);</code>	该函数用于获取电源信息, <code>power</code> 是一个指向 <code>SYSTEM_POWER_STATUS</code> 的指针, 该结构体内的成员表明电源是否插电、电池充电状态、剩余电量百分比等信息

“性能”页包含计算机系统和电源性能, 故该 Dialog 下含两张 List Control 控件, 调用 `DDX_Control` 函数实现两张 List 和资源 `PAGE2` 类的数据关联。

为方便用户理解“性能”的含义, 程序解析系统 API 返回结构体各个成员的含义, 并将其翻译为易于用户理解的文字, 使用 `StrFormatByteSize()` 和 `sprintf()` 函数格式化写入字符串, 然后将信息填入列表中。

5. “网络”信息查看的实现

1) 流程



“网络”页的信息获取、处理和显示流程大致如上，程序初始化标签页中的 Tree Control 和 List Control，为树新建根节点，为列表设置属性。然后调用 TreePro()递归整个网络，调用 WNetOpenEnum()和 WNetEnumResource()函数获取网络邻居中的资源信息，并按照此网络的层次结构构件一棵树，同时将资源的详细信息填入列表控件。

为更人性化，程序为树的每个节点标号，标号为该节点的子节点个数，并未其编写代码当鼠标滑过该节点时显示标号，以提示用户该资源下有几个子资源。另外本页面下方有提示框，用于提示用户当前正选中哪个节点。

2) 函数说明：

Table 7 自定义函数

编号	格式	功能	实现
1	<code>BOOL PAGE3::OnInitDialog()</code>	初始化函数，为树建立根节点；为列表写上属性。	1.调用 TreeProc() 开始遍历整个网络 2.默认初始状态展开根节点
2	<code>void PAGE3::TreeProc(HTREEITEM hRoot, LPNETRESOURCE lpr,</code>	递归整个网络；同时仿造网络结构构造一棵树，并且将资源的信息写入列表控件	对每一层容器类资源调用 OpenNetEnum()开

	<code>int depth)</code>	中。	始枚举；调用 <code>GlobalAlloc()</code> 来分配资源；对类型为容器的资源进行递归枚举；结束枚举后释放内存关闭枚举；对每个节点进行备注其子节点个数。
3	<code>HTREEITEM PAGE3::IntsertNode(HTREEITEM hRoot, LPNETRESOURCE lpnrLocal, int depth)</code>	向树中指定深度插入一个节点。	返回新节点，因为该节点可能存储一个容器类型的资源，则将来也会被作为根。
4	<code>void PAGE3::DisplayStruct(int i, LPNETRESOURCE lpnrLocal)</code>	将 <code>LPNETRESOURCE</code> 成员的含义翻译为便于用户理解的表述，调用 <code>AddItem()</code> 函数向列表中写入内容。	<code>Switch case</code> 语句对结构体每个成员的宏定义值进行翻译。
5	<code>void PAGE3::AddItem(TCHAR* num, TCHAR* remotename, TCHAR* type, TCHAR* displaytype, TCHAR* usage, TCHAR* localname, TCHAR* scope, TCHAR* comment, TCHAR* provider)</code>	将资源的详细信息写入列表控件。	
6	<code>void PAGE3::OnTvnGetInfoTipTreenet(NMHDR *pNMHDR, LRESULT *pResult)</code>	鼠标划过根节点外的某个树节点时，提示节点的备注——该资源下有几个子资源。	将该节点的附加 32 位数据格式化为字符串提示用户。
7	<code>void PAGE3::OnTvnSelchangedTreenet(NMHDR *pNMHDR, LRESULT *pResult)</code>	提示选中的节点，将提示结果显示提示框中。	获取当前选中节点的句柄，根据句柄获取节点的标签文本字符串，将字符串写入只读编辑框中。
8	<code>void PAGE3::OnBnClickedButtonnet()</code>	“刷新”事件响应。	清空树和列表中所有内容；重新遍历整个网络并填入新的信息。

Table 8 用到的控件相关函数

(对于“性能”页中已经使用的函数不再提及)

编号	格式	说明
1	HTREEITEM CTreeCtrl::InsertItem(LPCTSTR lpszItem, int nImage, int nSelectedImage, HTREEITEM hParent, HTREEITEM hInsertAfter)	向树视中插入一个节点
2	BOOL CTreeCtrl::Expand(HTREEITEM hItem, UINT nCode)	该节点默认为展开
3	BOOL CTreeCtrl::SetItemData(HTREEITEM hItem, DWORD_PTR dwData)	为该节点绑定一个内容，本是杨中绑定为该节点的子节点个数
4	HTREEITEM CTreeCtrl::GetRootItem()	获取根项目的句柄
5	CStringT<BaseType, StringTraits>::Format(PCXSTR pszFormat, ...)	
6	BOOL WINAPI SetDlgItemText(HWND hDlg, int nIDDlgItem, LPCSTR lpString);	将字符串显示到编辑框

Table 9 用到的系统 API

编号	格式	说明
1	DWORD APIENTRY WNetOpenEnum(DWORD dwScope, DWORD dwType, DWORD dwUsage, LPNETRESOURCEA lpNetResource, LPHANDLE lphEnum);	启动对网络资源进行枚举的过程
2	HGLOBAL WINAPI GlobalAlloc(UINT uFlags, SIZE_T dwBytes);	从堆中分配一定数目的字节数
3	DWORD APIENTRY WNetEnumResource(HANDLE hEnum, LPDWORD lpcCount, (*lpBufferSize) LPVOID lpBuffer, LPDWORD lpBufferSize	继 WNetOpenEnum()之后继续对网络资源的枚举

);	
4	HGLOBAL WINAPI GlobalFree(HGLOBAL hMem);	释放指定的全局内存块
5	DWORD APIENTRY WNetCloseEnum(HANDLE hEnum);	结束对网络资源的枚举过程

</侯思凡>

<王紫薇>

6. “文件浏览”及文件复制的实现

“文件浏览”对话框的实现结合【实验五 文件复制】，通过三个 Dialog：PAGE4、PAGE4_ATTR 和 PAGE4_COPY。实现文件夹（包含该目录下的文件及嵌套目录和文件的浏览、复制以及删除。

控件和响应函数图示如下：

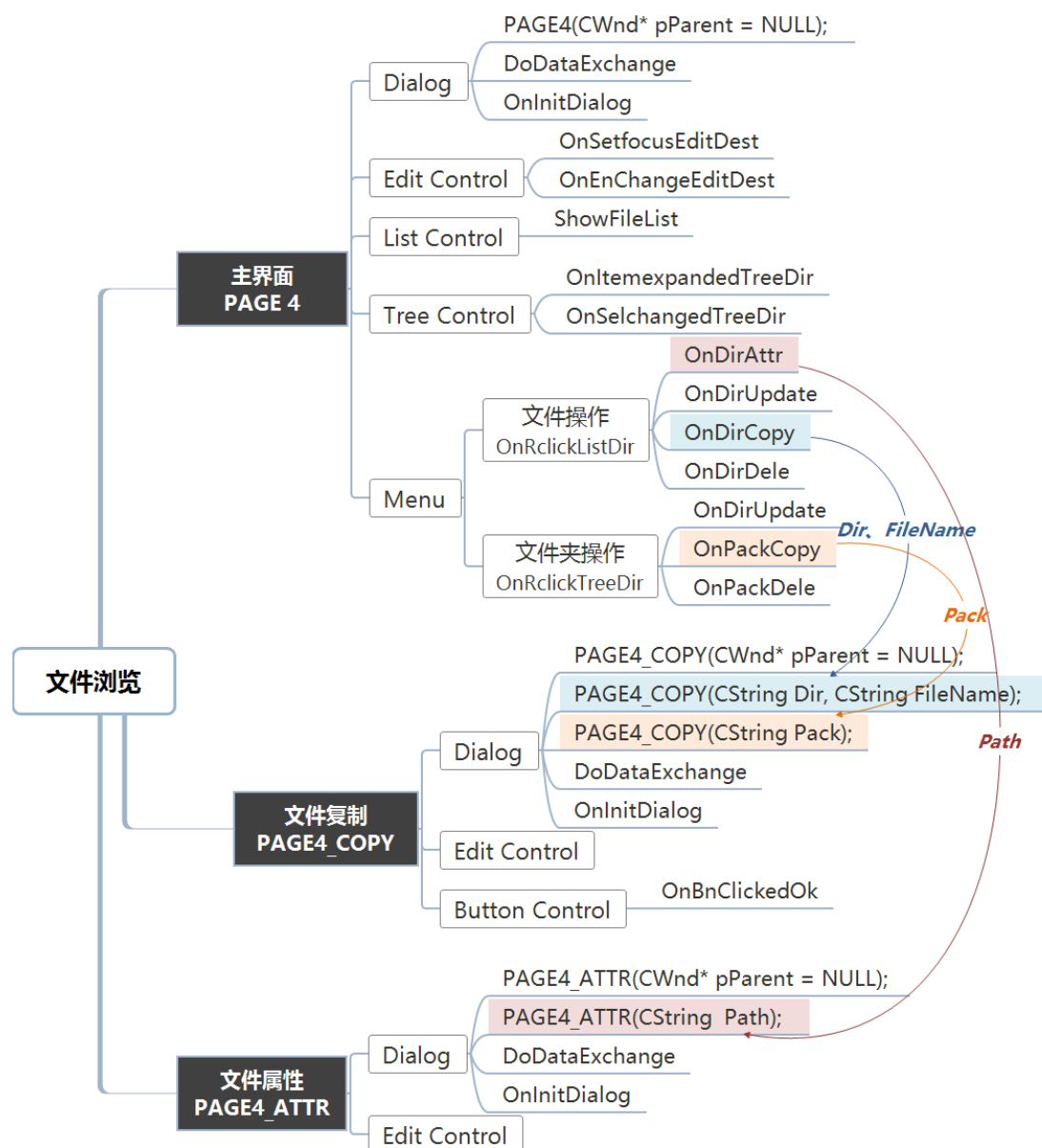


Figure 3 文件操作的控件与响应函数

(1) 主界面 PAGE4

主界面显示地提供文件浏览、文件夹查看、指定路径文件夹查看功能。在树和列表中点击弹出菜单，提供其他功能。

1) 文件浏览、文件夹查看

此功能通过 Tree Control & List Control 控件实现：Tree Control 实现文件资源的浏览，运用 FindFile、FindNextFile 函数，以递归的方式获取指定目录下的所有文件夹、文件的遍历。我们希望在点击 Tree Control 中文件夹时，触发响应函数，将该文件夹下的所有文件以图标的形式显示在 List Control 中，在点击 Tree Control 中的“+”以扩展树时不更新 List Control 界面。主要的方法如下图：

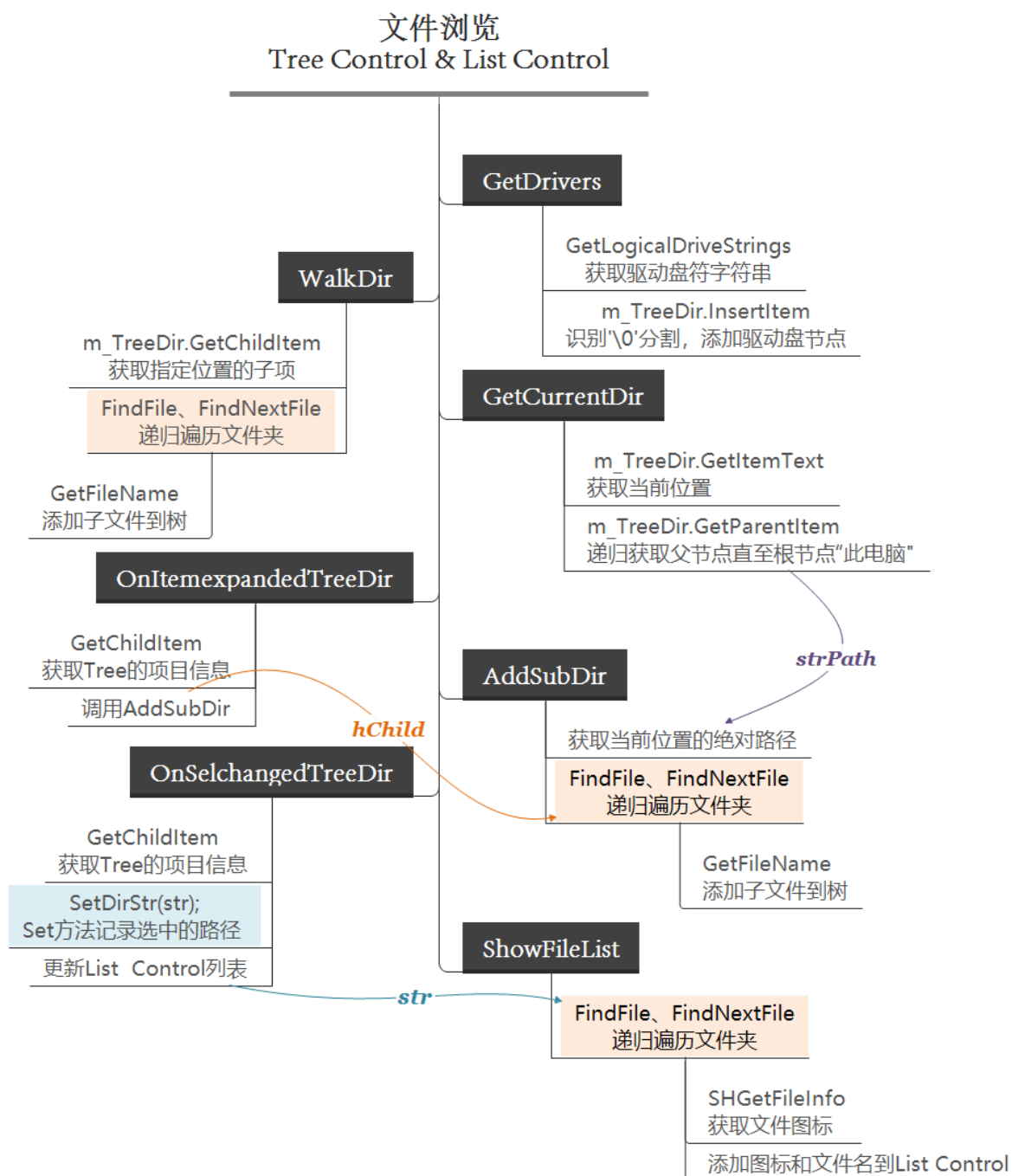


Figure 4 Tree Control & List Control 实现文件浏览

2) Edit Control

Edit Control 提供一种直接浏览指定目录下文件的功能。由确认按钮触发响应函数 OnClickedListOk, 获取键入的目录调用 ShowFileList 显示文件。

3) 右键菜单

在 Tree Control 和 List Control 中点击右键将弹出菜单, 菜单选项在 IDR_MENU2 的子菜单 0 和子菜单 2 中定义。菜单选项见“控件和响应函数图示”。删除和刷新文件夹和文件的响应函数在 PAGE4 的对话框中即可实现, 复制和属性查看将调用新的

对话框 PAGE4_COPY 和 PAGE4_ATTR。

(2) 文件、文件夹复制 PAGE4_COPY

我们希望当点击复制选项时，弹出新的对话框显示源路径（只读），用户输入目标路径，点击确认完成复制。因此首先实现不同对话框之间的参数传输。结合面向对象的知识，对构造函数进行重载，传入不同的参数，分别复制文件和文件夹。

	文件复制	文件夹复制
对话框	PAGE4_COPY	
响应函数	OnDirCopy	OnPackCopy
构造函数	PAGE4_COPY(CString Dir, CString FileName);	PAGE4_COPY(CString Pack);
拷贝函数	CopySingleFile	CopyFolder
拷贝思想	直接拷贝	递归

Figure 5 文件、文件夹复制对比

(3) 文件属性查看 PAGE4_ATTR

该对话框的主体是 Static Control 做标题，文件的属性信息调用 GetStatus 函数将信息填充到 CfileStatus 结构体，格式化输出到 Edit Control。

</王紫薇>

<罗薇>

7. “文件清理”的实现

(1) 文件清理页框架的搭建

Page5 页面整体设计构思：一个用于选择待清理文件路径的文本框，一个用于选择待清理文件类型的文本框，一个用于显示扫描后获得全部符合条件文件的 list 表格，以及扫描和文件清理两个按钮。

具体实现：

- 1). 在 Page5 的 dialog 资源视图上方填充两个 Text Control 控件,两个 Edit Control 控件,两个 Button Control 控件，分别用于显示和获取待清理的文件路径和类型。
- 2). 在下方填充一个 List Control 控件，扫描之后的所有文件将填入 List 中。
- 3). 添加一个 Check Box Control 控件，用于全选/全不选 list 中的文件。
- 4). 添加两个 Button Control 控件分别用于扫描和清理文件。
- 5). 添加 CListCtrl 类变量 m_list_page5, CEdit 类变量 m_edit_fileroute, CEdit 类变量 m_edit_filetype, 在 DoDataExchange()中写入 DDX_Control(pDX, IDC_EDIT_FILEROUTE, m_edit_fileroute) , DDX_Control(pDX, IDC_EDIT_FILETYPE, m_edit_filetype) , DDX_Control(pDX, IDC_LIST_PAGE5, m_list_page5) , 将变量 m_edit_fileroute 、 m_edit_filetype、m_list_page5 分别和控件 IDC_EDIT_FILEROUTE、IDC_EDIT_FILETYPE、IDC_LIST_PAGE5 进行绑定。

(2) 选择待清理文件夹路径

点击第一个...按钮添加事件处理程序。声明一个库中含有的 CFolderPickerDialog 类的实体，对其进行初始化，并调用其 DoModal()方法打开文件选择对话框。调用 GetPathName()方法获得用户选择的文件夹的路径，并调用 SetDlgItemText()将文件夹路径填入相应编辑框。

(3) 文件类型选择页框架的搭建

在 CHOOSE EXTENDED DIALOG 中填充一个 List Control 控件，显示可供用户选择的文件类型。添加一个 Check Box Control 控件，用于全选/全不选 list 中的文件。添加一个 Check Box Control 控件，点击时保存用户的选择状况并返回上一对话框。

(4) 选择待清理文件类型

1). 为文件类型选择对话框添加 ChooseExtension 类

在初始化 OnInitDialog() 中，使用 InsertColumn() 插入表头，使用 InsertItem()、SetItemText()、SetCheck() 插入每行的文件类型、描述以及设置 Checkbox 选项。Checkbox 选项保存在 ChooseExtension 类的 BOOL extState[34] 成员变量中。

为全选按钮添加事件处理程序。调用 GetCheck() 方法获得全选 CheckBox 状态，并遍历 list 的每一行对其 CheckBox 状态进行检查与更改。

为确定按钮添加事件处理程序。首先遍历列表，获得每行的状态，并将状态存入 extState[] 数组。然后调用 OnCancel() 关闭当前对话框，退回上一对话框。

2). 实现清理文件类型的选择

为第二个...按钮添加事件处理程序。声明一个 ChooseExtension 类的实体，将其 extState[] 初始化，使用 DoModal() 方法弹出该对话框。

用户单击 ChooseExtension 对话框确认按钮后的处理，获得该类实体中 extState[] 中表示的文件类型选中状态，将其存入 extend[] 结构体数组，同时生成 Edit 框回显字符串。调用 SetDlgItemText 方法将选中文件类型字符串填入 Exit 框。

(5) 按用户选择的文件路径和类型扫描

为扫描按钮添加事件处理程序。首先调用 DeleteAllItems() 方法清空文件路径 List。使用 GetWindowText() 获得文件路径 Edit 框中的路径，根据此路径使用自定义的 searchFile() 函数进行文件扫描。扫描结束后使用 MessageBox 弹出提示框。

自定义 searchFile 函数的框架用到了实验五复制文件的内容，不同之处在于取消了目标文件路径的设置以及更改了检测到文件后的处理内容。首先调用 isWantedFileType 自定义方法判断查找到的文件是否为要清理的文件类型。如果是要清理的文件类型，就将该文件的路径填入文件路径 List。

对于判断文件类型的自定义方法 isWantedFileType，将给定的文件名与 extend[] 结构体中每个置为 true 的文件类型进行正则匹配，其中的正则匹配通过调用 regex 头文件中的 regex_match 方法实现。一旦找到匹配的文件类型则返回相应的序号，若没有匹配的文件类型则返回-1。

(6) 实现对指定文件的清理

为清理按钮添加事件处理程序。调用 List 的 GetItemCount 方法，若 List 中没有条目，则弹窗通知并返回。遍历 List 中的所有项，若 GetCheck() 判断为选中项，则 GetItemText() 获得文件路径，使用 DeleteFile 删除相应文件，并使用 DeleteItem() 删除 List 中的条目。若没有清理项，则弹窗提示并返回。反之，弹窗提示清理文件数。

</罗薇>

<陈牧乔>

8. “详细信息” 页的实现

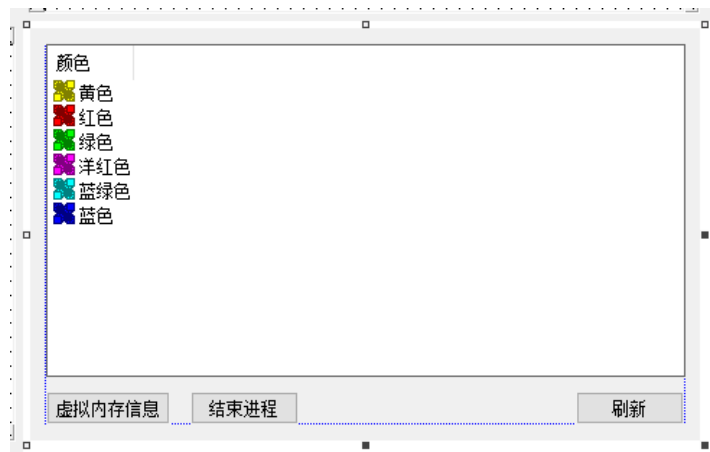
(1) 绘制详细信息页框架

如图。List 用于列出当前计算机中的进程，包括进程名称、进程 PID、进程内存大小、进程描述等。

左下角虚拟内存信息按钮用于弹出虚拟内存信息对话框，显示选定进程的虚拟内存信息。

下方结束进程按钮用于结束选定的进程。

右下角刷新按钮用于重新获取系统的进程列表。



(2) 实现对进程及其信息的获取

自定义 ListProc() 方法。

框架用到了实验四内存监视中查看进程部分的内容，使用 `CreateToolhelp32Snapshot()` 获得进程快照，使用 `Process32First()` 获得第一个进程后循环使用 `Process32Next()` 遍历每一个进程。遍历过程中通过快照获得进程名、PID，使用 `OpenProcess()` 打开进程句柄，并使用 `GetProcessMemoryInfo()` 获得进程的大小。

自定义 `GetProcessFilePath()` 方法通过进程句柄获得进程地址，再使用自定义方法 `GetFileVersion()` 通过进程地址获得描述信息。

(3) 绘制虚拟内存信息对话框

如图。上方进程 PID、进程名称、进程大小对应的 Edit 框分别显示选中进程的相关信息。

中间的 List 显示进程的虚拟内存块信息，包括块的起始地址、结束地址、大小、状态、保护、类型、名称。



(4) 实现虚拟内存信息的获取

为虚拟内存信息按钮创建事件处理程序。获取所选行的 PID、进程名称、进程内存大小，新建 VMWalker 类实体并将上述字段写入实体其中。使用 DoModal() 弹窗。

在 VMWalker 类中，实现通过进程 PID 遍历虚拟内存块的功能。这用到了实验四内存监视中显示进程虚拟地址空间布局的内容。不同之处在于要将获得的信息做适当的格式转换，以便填入 List 中。

(5) 实现结束进程功能

为结束进程按钮创建事件处理程序。首先 OpenProcess() 根据 PID 打开指定进程获得句柄，接着使用 TerminateProcess() 根据句柄结束进程。接下来刷新列表。

(6) 实现刷新功能

为刷新按钮创建事件处理程序。调用列表实体的 DeleteAllItems() 方法清空列表，接着调用 ListProc() 重新列出进程列表。

(7) 实现排序功能

为 List 创建事件处理程序 OnLvnColumnclickListDetail()。调用 CListCtrl 的 SortItems() 方法实现对指定列的正序或倒序排序，需要自定义 MyCompareProc() 比较方法。

在 MyCompareProc() 中提取两列的相关内容，并按照排序关键字的不同进行整数型或字符串型的比较。

定义 BOOL 类关键字 method 用于指示排序方式，即正序或倒序。每次排序后都将 method 取反，并在比较方法中对其进行判断，以便实现不同字段的不同顺序的排序。

</陈牧乔>

<罗薇>

9. “内存”信息查看的实现

(1) 内存页框架的搭建

在 Page7 的 dialog 资源视图中填充一个 List Control 控件，添加 CListCtrl 类变量 m_ListSystemInfo。在 DoDataExchange() 中写入 DDX_Control(pDX, IDC_LIST_SystemInfo, m_ListSystemInfo)，将变量 m_ListSystemInfo 和控件 IDC_EDIT 进行绑定。

(2) 获取系统内存信息，在 list 中填充内容

在 List Control 控件中添加两列，分别表示系统内存信息的分类及其对应数据。自定义函数 ListProc()，调用 GlobalMemoryStatusEx(&statex) 函数获取内存使用率、物理内存、可用物理内存、提交的内存限制、虚拟内存、可用虚拟内存 5 项信息，m_ListSystemInfo.InsertItem(int nItem, LPCTSTR lpszItem) 函数将获取的信息填入 List Control 控件中。

(3) 实现系统内存信息的刷新

添加一个 Button 控件，更改名字为 IDC_BUTTON_REFRESH，用来实现刷新功能。添加 button 单击响应事件 OnBnClickedButtonRefresh()，当触发时调用 m_ListSystemInfo.DeleteAllItems() 清空当前 list，重新调用 ListProc() 函数获取内存信息再填入。

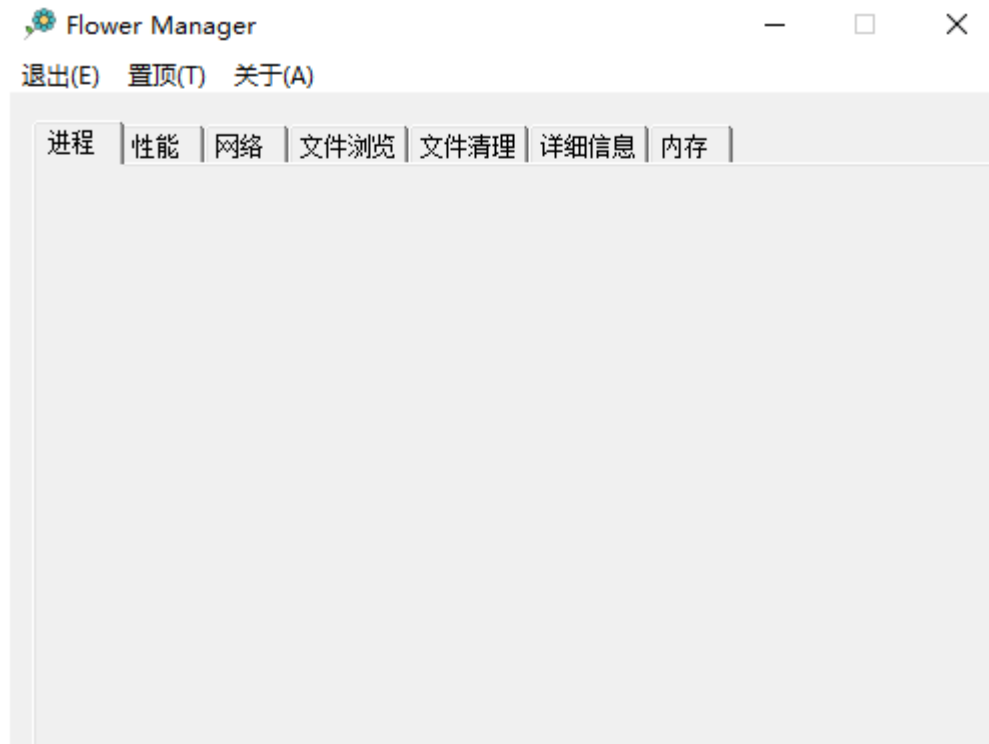
</罗薇>

五、 实验结果和分析

<罗薇>

1. 总框架

本项目总框架如下。在菜单项和标签页内容补全后，点击菜单项可显示相应菜单内容，点击标签页可切换，并显示相应标签内容。

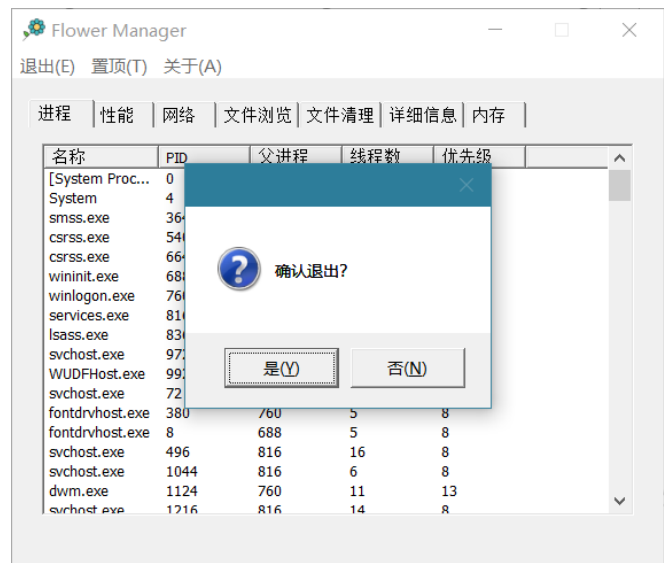


</罗薇>

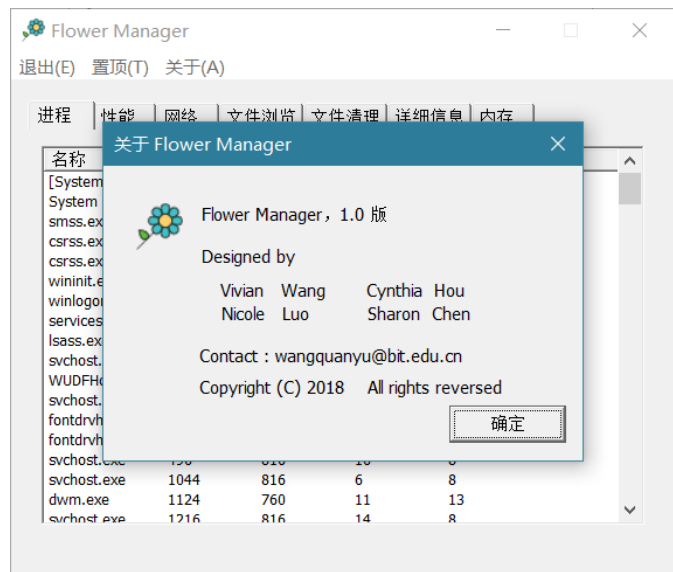
<侯思凡>

2. “菜单”和最小化到托盘

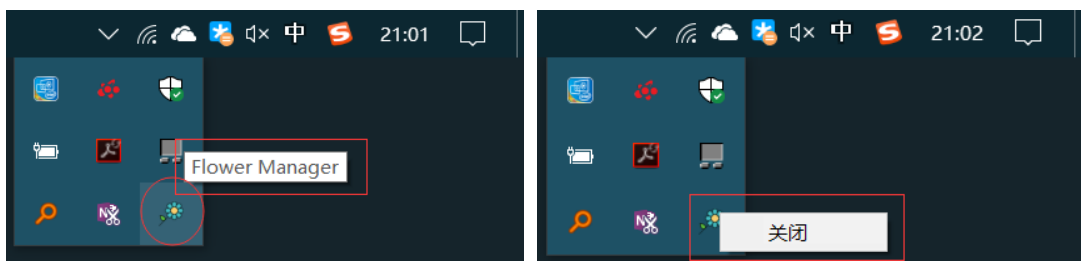
- 1) 点击菜单的“退出”项，将弹出确认退出的消息框，选择“是”将退出 Flower Manager，选择否将回到管理器界面：



- 2) 点击“置顶”项，Flower Manager 窗体将被置顶；在此点击将取消置顶。重复多次，则奇数次点击置顶，偶数次点击取消置顶。
- 3) 点击“关于”项，弹出本项目设计人员相关信息：



- 4) 点击最小化，窗体被最小化到托盘。到托盘区查看该图标提示为“Flower Manager”；右键提示“关闭”功能；双击可恢复正常显示：

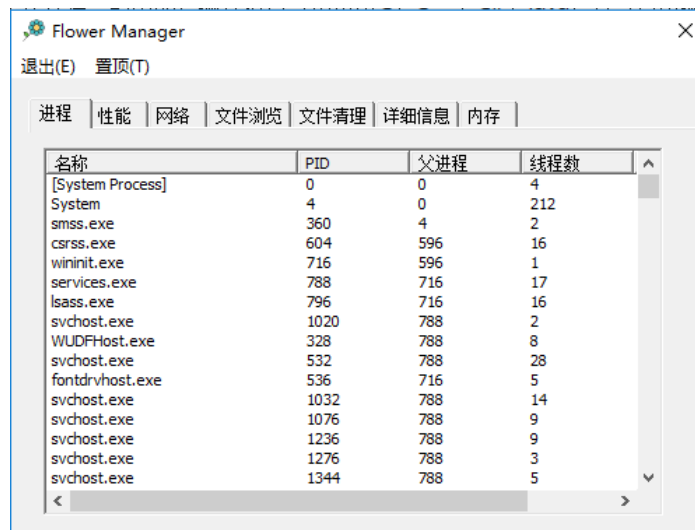


</侯思凡>

<王紫薇>

3. “进程”

进程信息：



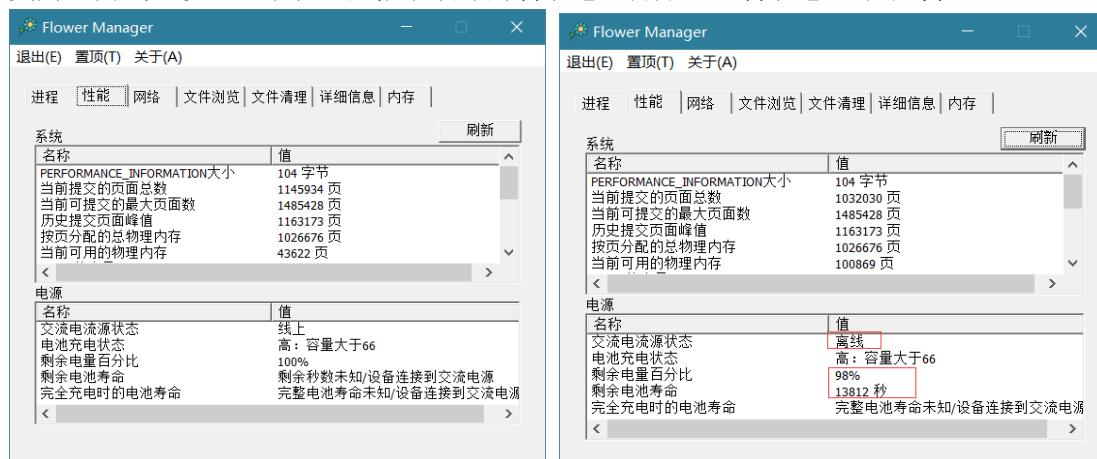
名称	PID	父进程	线程数
[System Process]	0	0	4
System	4	0	212
smss.exe	360	4	2
csrss.exe	604	596	16
wininit.exe	716	596	1
services.exe	788	716	17
lsass.exe	796	716	16
svchost.exe	1020	788	2
WUDFHost.exe	328	788	8
svchost.exe	532	788	28
fontdrvhost.exe	536	716	5
svchost.exe	1032	788	14
svchost.exe	1076	788	9
svchost.exe	1236	788	9
svchost.exe	1276	788	3
svchost.exe	1344	788	5

</王紫薇>

<侯思凡>

4. “性能”

如图，“性能”页显示了关于系统和电源的性能信息，左图为连接电源时，右图为拔掉电源等待一段时间后的刷新效果。可以刷新后看到非分页池大小、句柄数、进程线程个数等系统性能信息变化，因为系统进程处于不断变化中；由于计算机拔出电源，故交流电源由“线上”变为“离线”状态，系统也汇报了预计的剩余电量百分比、剩余电池寿命等信息：



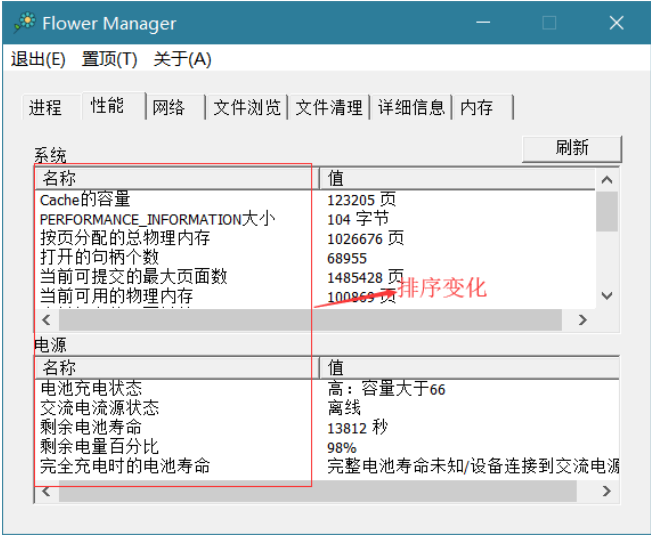
名称	值
PERFORMANCE_INFORMATION大小	104 字节
当前提交的页面总数	1145934 页
当前可提交的最大页面数	1485428 页
历史提交页面峰值	1163173 页
按页分配的总物理内存	1026676 页
当前可用的物理内存	43622 页

名称	值
交流电源状态	线上
电池充电状态	高：容量大于66
剩余电量百分比	100%
剩余电池寿命	剩余秒数未知/设备连接到交流电源
完全充电时的电池寿命	完整电池寿命未知/设备连接到交流电源

名称	值
PERFORMANCE_INFORMATION大小	104 字节
当前提交的页面总数	1032030 页
当前可提交的最大页面数	1485428 页
历史提交页面峰值	1163173 页
按页分配的总物理内存	1026676 页
当前可用的物理内存	100869 页

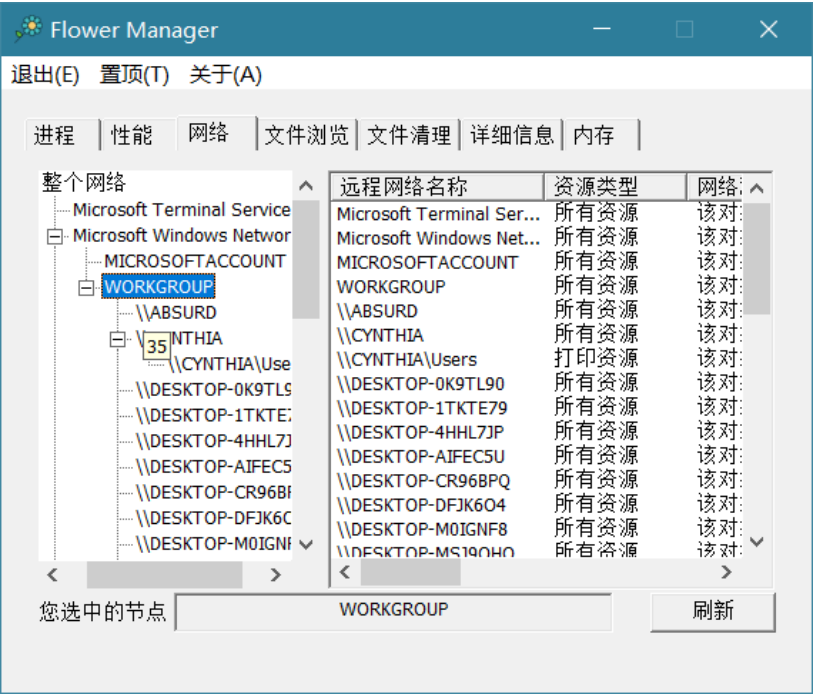
名称	值
交流电源状态	离线
电池充电状态	高：容量大于66
剩余电量百分比	98%
剩余电池寿命	13812 秒
完全充电时的电池寿命	完整电池寿命未知/设备连接到交流电源

点击“名称”列，列表排序发生了变化（“值”列有同样效果）：



5. “网络”

如图为 Flower Manager 获取的网络信息，右侧为网络结构，鼠标滑过节点时，提示该节点下有几个子节点；点击节点在下方只读编辑框中提示您当前选中了哪个节点；右侧为所有资源的详细信息；点击列表头可以对列表内容排序：



</侯思凡>

<王紫薇>

6. “文件浏览”

(1) 文件浏览：

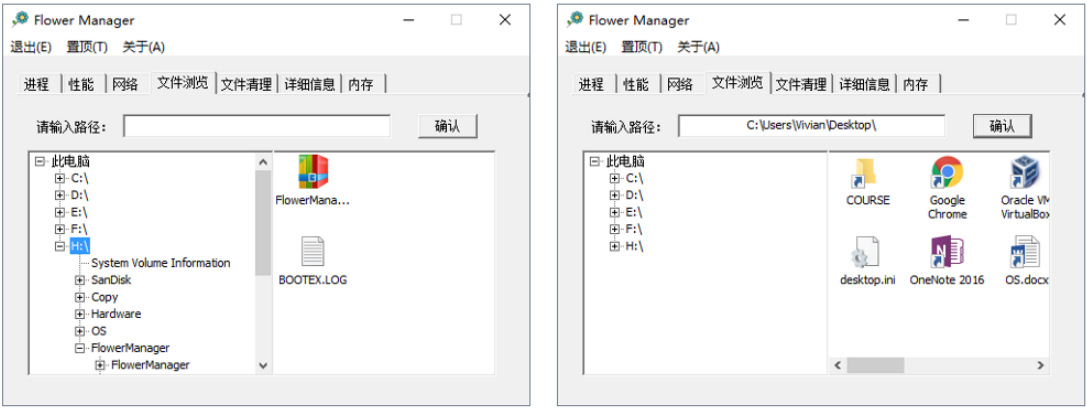


Figure 6.5.1 文件浏览

(2) 文件夹操作

1) 文件夹复制

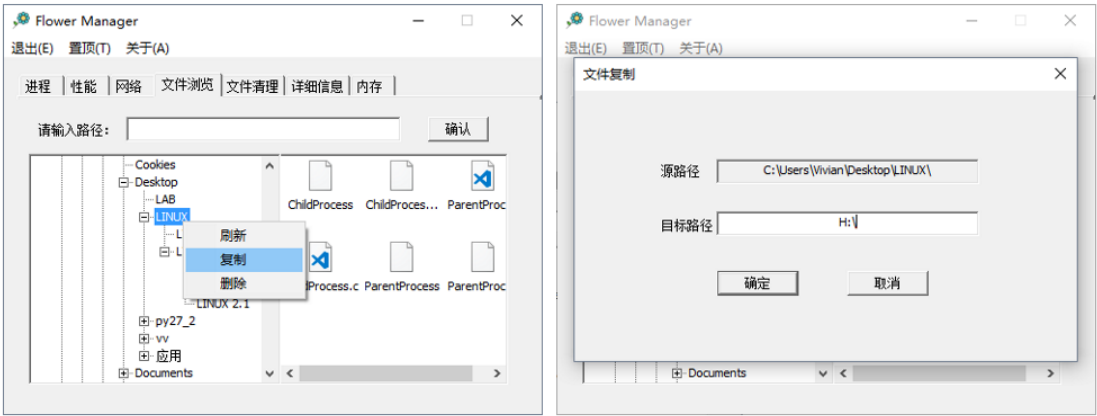


Figure 5.5.2 文件夹复制

复制结果:

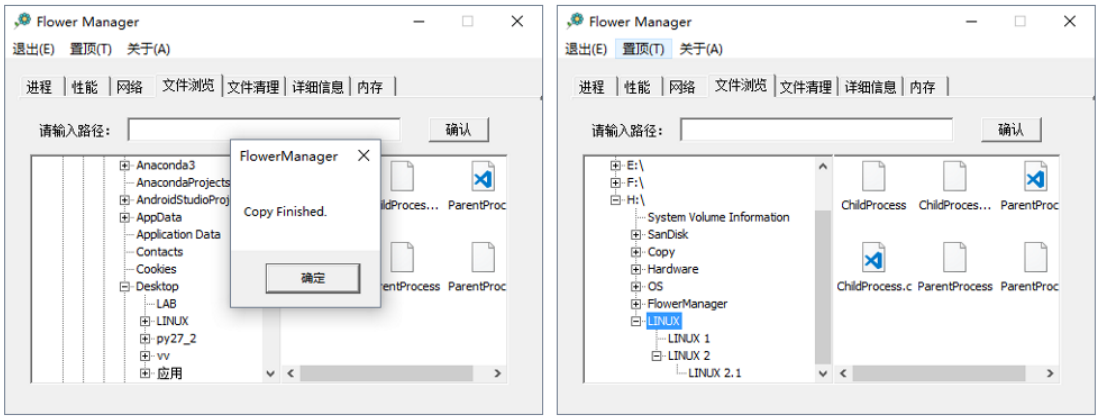


Figure 5.5.3 文件夹复制结果

2) 文件夹删除

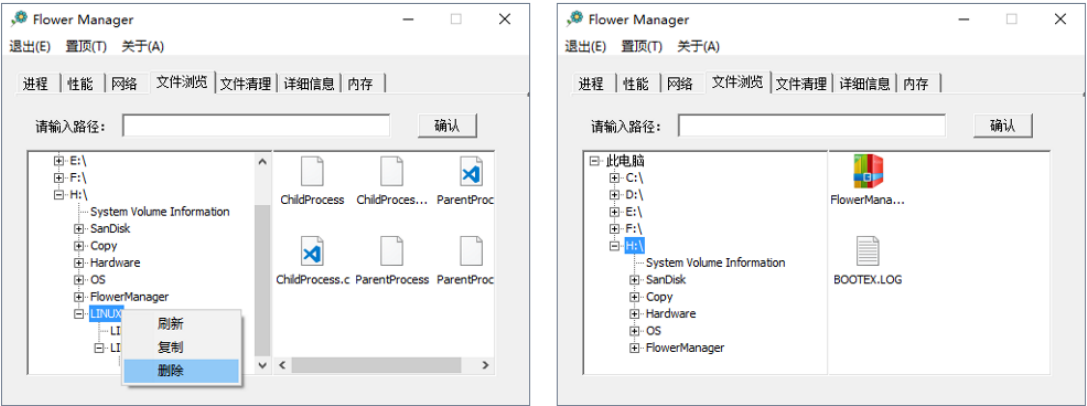


Figure 5.5,4 文件夹删除

(3) 文件操作

1)文件属性查看

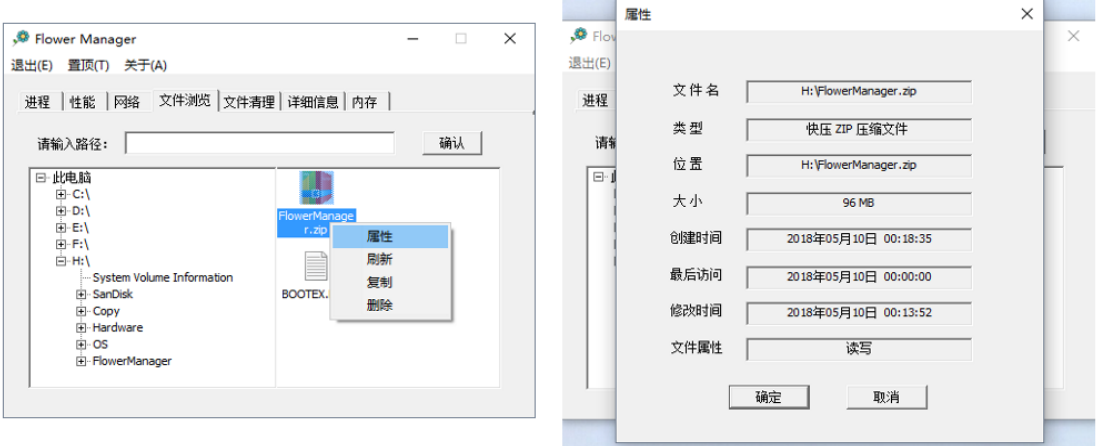


Figure 5.5.5 文件属性

2) 文件复制

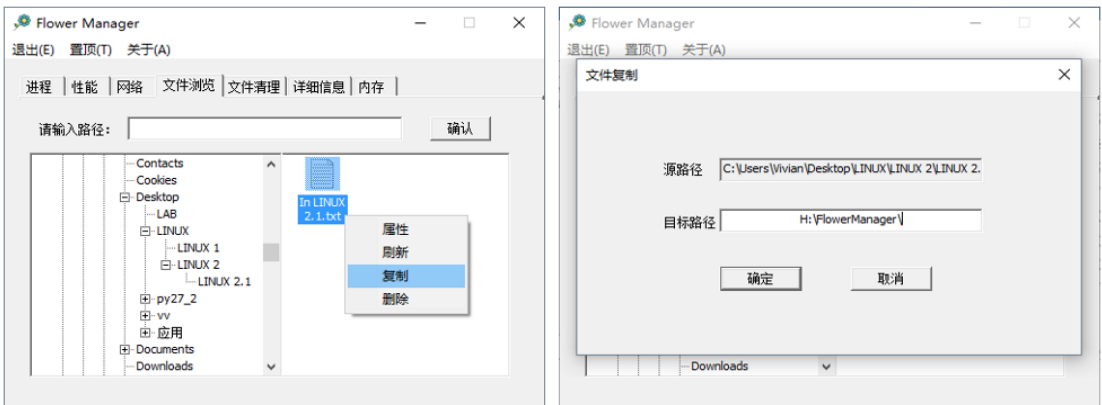


Figure 5.5.6 文件复制

复制结果

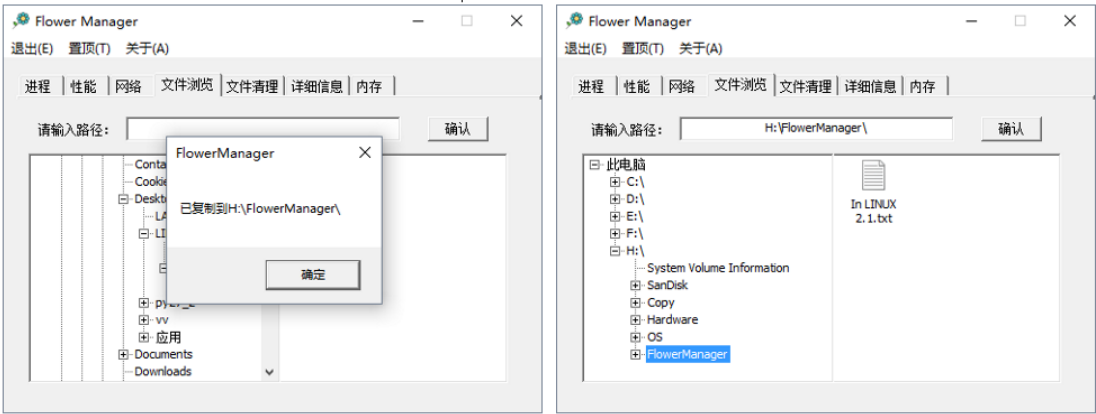


Figure 5.5.7 文件复制结果

3) 文件删除

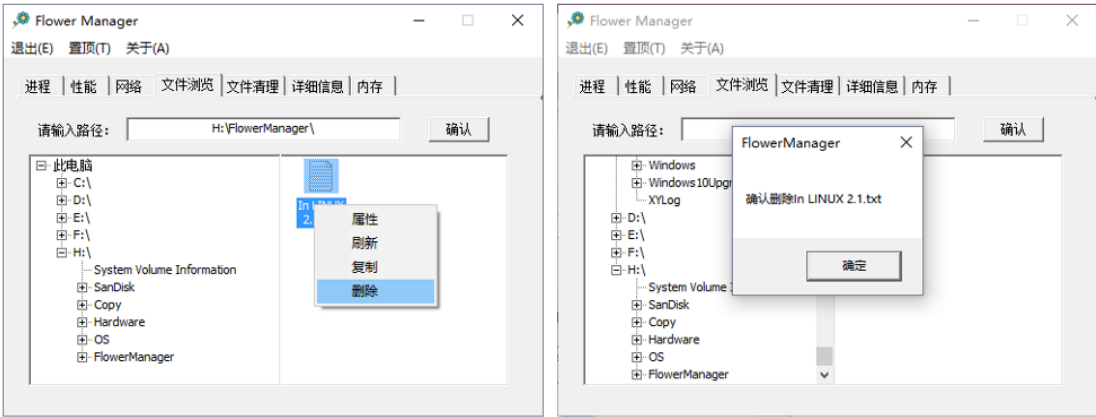


Figure 5.5.8 文件删除

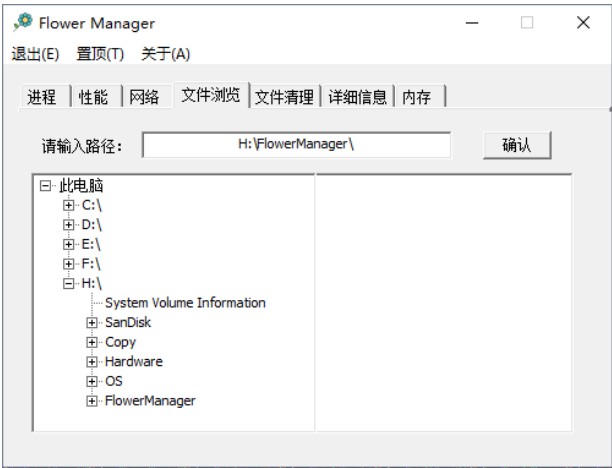


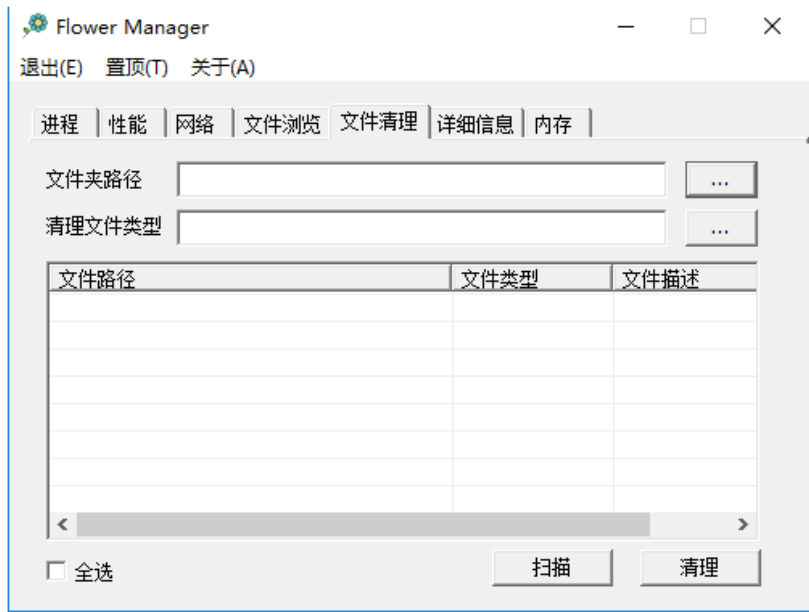
Figure 5.5.9 文件删除结果

</王紫薇>

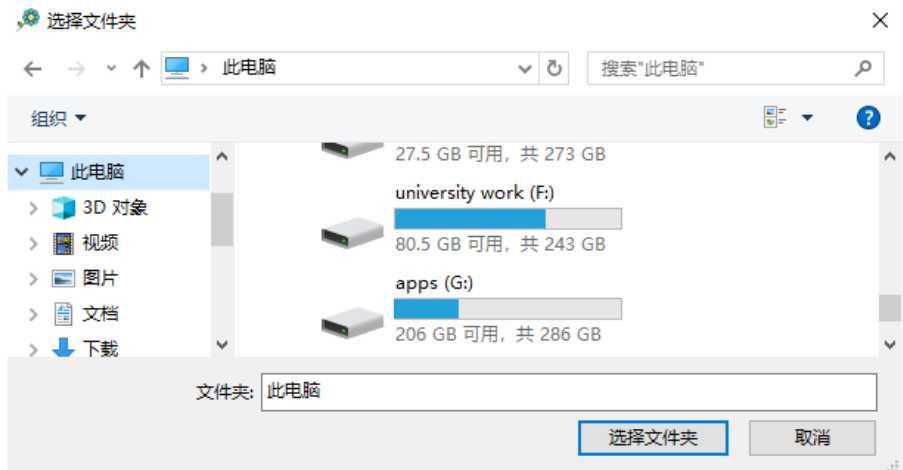
<罗薇>

7. “文件清理”

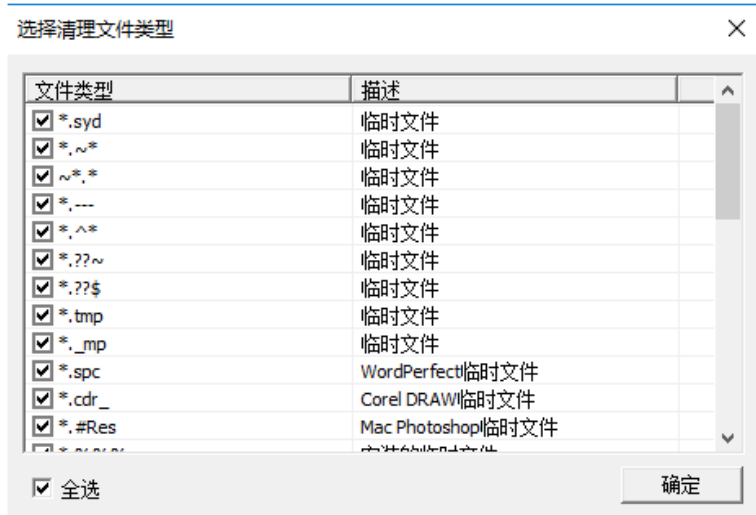
点击“文件清理”切换至文件清理标签页。



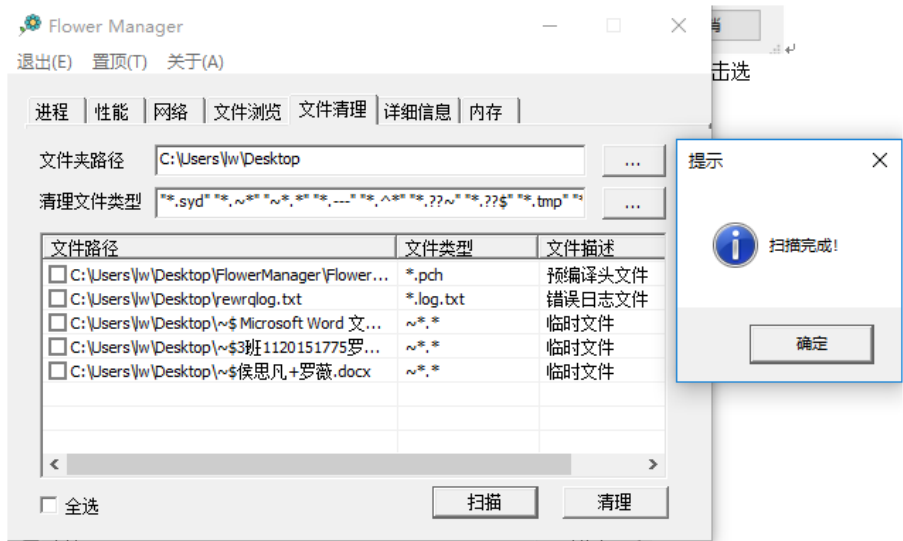
点击第一个...按钮弹出选择待清理文件路径的对话框，选择待清理文件路径后，点击选择文件夹返回上一界面，并将选择的文件夹路径填入第一个文本框。



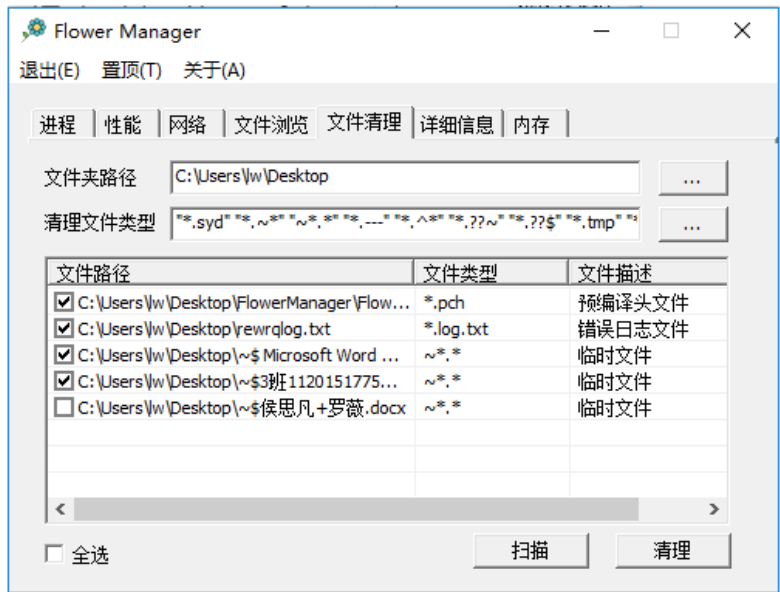
点击第二个...按钮弹出选择待清理文件类型的对话框，选择待清理文件类型后，点击选择文件夹返回上一界面，并将选择的文件类型填入第二个文本框。



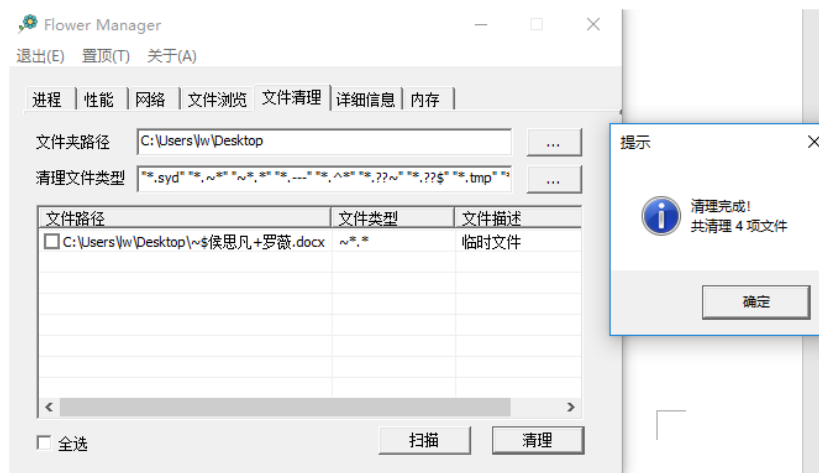
点击扫描按钮，在指定的文件夹路径查找指定类型的文件，并将扫描结果填入 list，完成后提示扫描完成。



勾选想要清理的文件，点击清理按钮，实现文件清理。



如图，清理掉了勾选的前四个文件，并提示清理完成。



</罗薇>

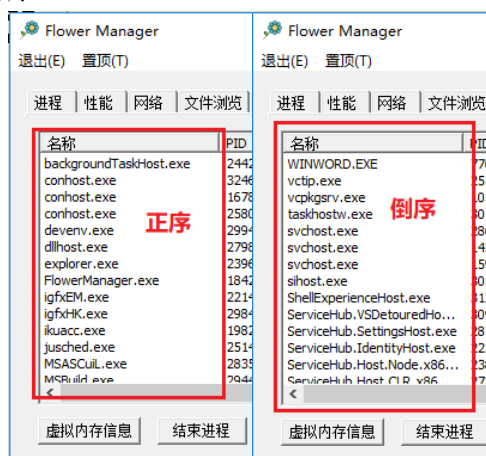
<陈牧乔>

8. “详细信息”

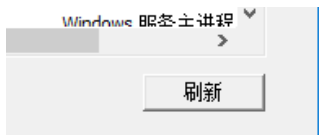
进入详细信息界面，可以看到系统进程列表，包括进程的名称、PID、内存、描述信息。



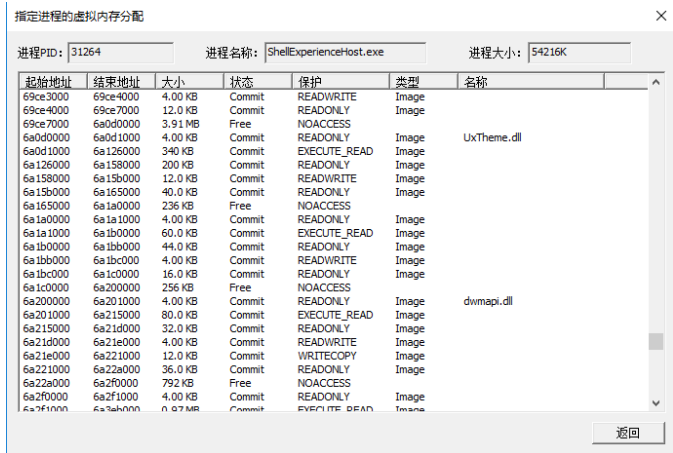
单击表头中的某项，可以对该列进行正序或倒序的排序。名称和描述信息按字符排序，PID 和内存大小按整数大小排序。



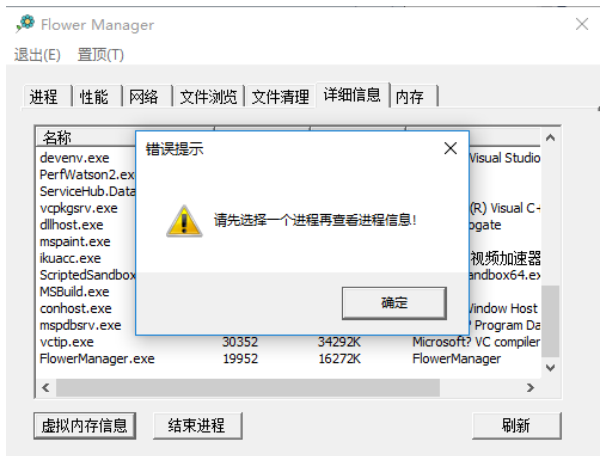
单击刷新按钮，则会重新获取系统的进程并显示在列表中。



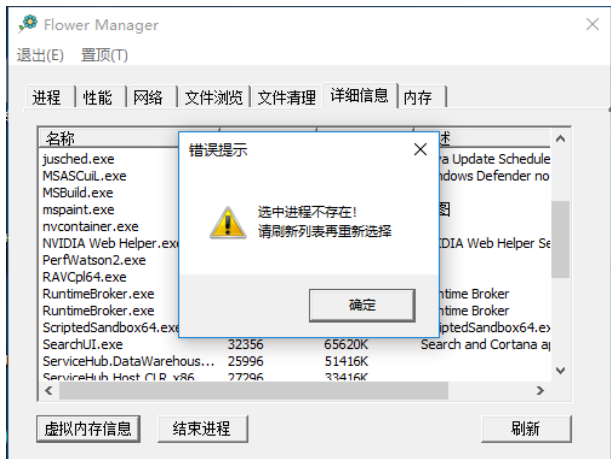
选中某进程并单击虚拟内存信息按钮，则可以在弹出的对话框中查看该进程的虚拟内存分配情况，包括虚拟内存块的起始地址、结束地址、大小、状态、保护、类型及名称。



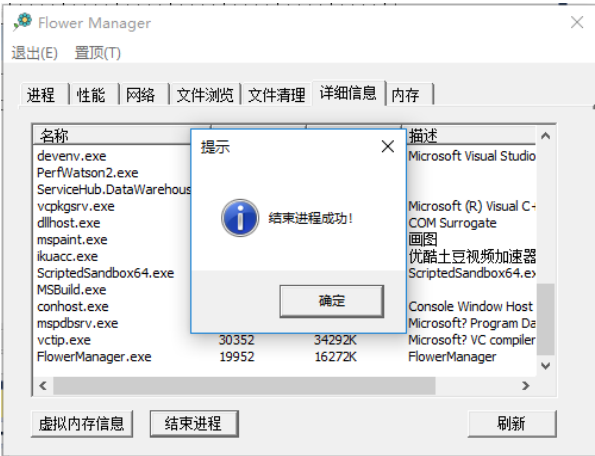
如果用户没有选择一个进程就单击虚拟内存信息按钮，则会弹出提示框提示用户选择一个进程。



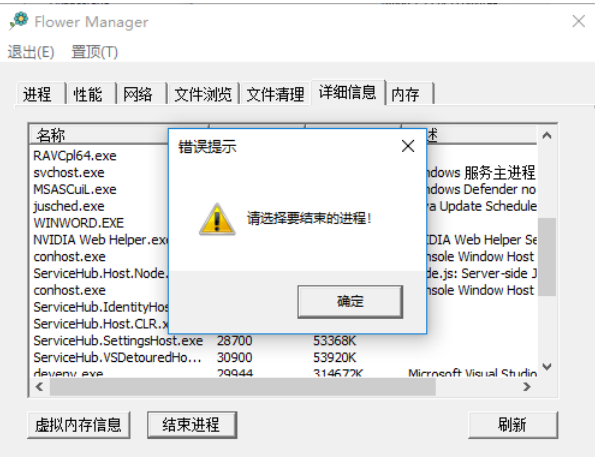
如果用户选中的进程已关闭，但是仍然存在于列表中，则弹窗提示进程不存在。



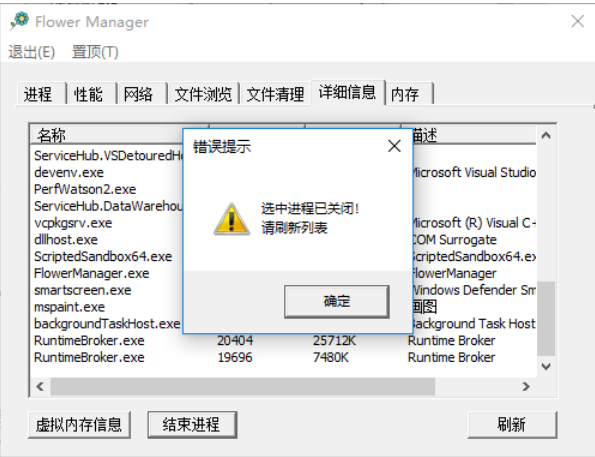
选中一个进程并单击结束进程按钮，如果成功结束进程，就弹窗提示结束成功。



如果用户没有选择进程就单击结束进程按钮，则弹窗提示用户选择进程。



如果用户选中了一个已关闭的进程并单击结束进程按钮，则弹窗提示刷新列表。

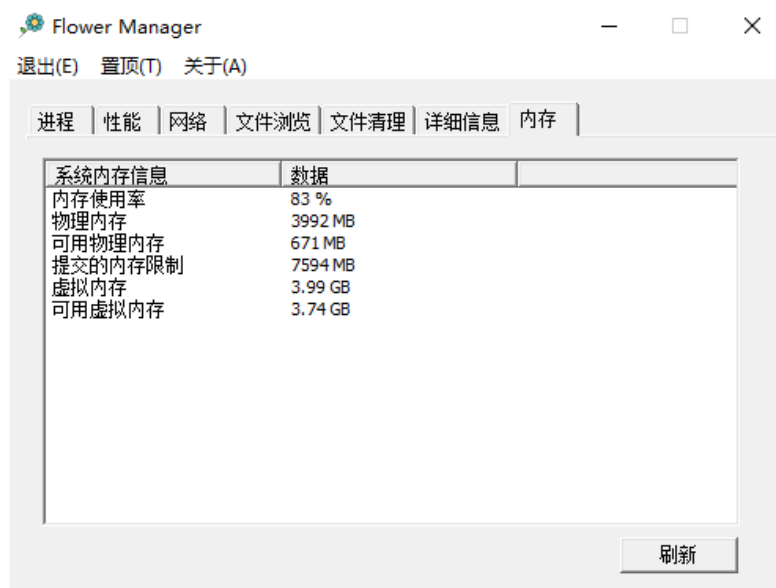


<陈牧乔>

<罗薇>

9. “内存”

点击“内存”切换至内存标签页，显示系统内存信息。点击刷新按钮可刷新。



</罗薇>

六、 讨论、心得

1. 实验中遇到的问题

<侯思凡>

(1) 主对话框搭建时遇到的问题

1) 菜单无法直接通过控件加入

在第一阶段进行主对话框搭建的时候，发现 MFC 提供的工具箱里没有菜单控件。后经查阅资料得知应在资源文件中新建一个 Menu 资源，设计项目所需的菜单，然后在想加入菜单的对话框类中编辑相应的代码添加菜单。

2) Tab Control 无法实现标签页切换

起初模仿查阅的资料，编写 DDX 函数，创建对话框、添加响应事件，但是点击标签切换无效。后经尝试发现，切换标签的响应事件应当在 FlowerManagerDlg.cpp 中编辑，而非为 Tab Control 新建的类。原因未经查实，经实践认为是 Tab Control 和七个子对话框均为填充在 FlowerManagerDlg 中的控件，故对这些控件的事件响应应当在这部分代码中编写。

3) 未知重写字符

此问题为头文件包含问题，解决方法有多种，本实验将头文件在直接引用它的 cpp 中 include，此做法为避免头文件被多次引用而产生位置重写字符的情况。实验中在实现标签页的切换 debug 时又避免了改错误。

(2) “性能”模块实现时遇到的问题

与电源性能相关 API 中的结构体成员标识含义晦涩，故以“用户是傻子”为理念的程序员决定将成员值的含义翻译为用户理解的文字，如值 ACLineStatus 为 0、1、2 分别表示离线、线上、未知状态。

(3) “网络”模块实现时遇到的问题

Tree Control 没有索引，边建立树，边给节点编号。访问节点时获取编号，可以以此实现对树的索引。但后期考虑索引在本项目无多大意义，将编号改为该节点的子节点数，并且通过鼠标滑过显示子节点数提示用户该资源下有几个子资源。

只一棵树信息太过单调，故便枚举资源，边将资源的详细信息写入列表控件，便于用户获得更多信息。

</侯思凡>

<王紫薇>

(4) “文件浏览”模块实现时遇到的问题

<1>文件属性传递参数

➤ 问题描述:

选取 List Control 中文件后由索引获取到了文件名，绝对路径如何获取便出现了问题。因此想到在 Tree Control 中选择文件夹时保存选中的文件夹路径（或有 Edit Control 直接获取的路径）。

➤ 解决方法:

创建 protected 类型变量 DirPath 保存文件所在文件夹的路径，使用 Set 和 Get 方法修改和获取该变量。

<2>在文件复制的基础上增加文件夹复制

➤ 问题描述:

文件复制的实现先于文件夹复制的实现。此时向 PAGE4_COPY 传入的文件夹路径、文件名这两个参数，直接加到本来的构造函数中。之所以需要两个参数而不是‘文件的绝对路径’一个路径这一个参数，是希望在复制时能方便的获取文件名而不用进行字符串的处理。此时再加上文件夹的复制，只需传入文件夹路径一个参量，如果改变之前的显得有些繁琐。

➤ 解决方法

最初在实现文件复制和文件属性查看时，不同对话框之间传递参数写成如下形式（以文件复制 PAGE4_COPY 为例）:

```
PAGE4_COPY(CString Dir, CString FileName, CWnd* pParent = NULL);
```

由于构造函数是搭建 MFC 框架时自动生成的代码，因此并未过多考虑其意义。解决这个问题便可从重载构造函数入手。更改后的构造函数:

```
PAGE4_COPY(CWnd* pParent = NULL);           // 标准构造函数
PAGE4_COPY(CString Dir, CString FileName);   // 文件复制构造函数
PAGE4_COPY(CString Pack);                   // 文件夹复制构造函数
```

此时文件的绝对路径是 Dir|FileName，而文件夹路径 Pack 末尾经处理一定是‘\’，故可在确认 Button 的响应函数中添加判断，就可以用同一个 Dialog 区分得进行文件夹复制和文件复制。

<3>Tree Control 右键获取节点并弹出菜单

➤ 问题描述:

右键获得鼠标当前位置对应元素并弹出的菜单时，菜单没有出现在鼠标点击位置，而是出现在屏幕的左上角。

➤ 解决方法:

查找资料发现，主要因为 CTreeCtrl 的点击测试 HitTest 坐标点是基于自身坐标

系，因此需要用 MapWindowPoints 做坐标系的映射。

<4>dlgdata.cpp line 40 断言失败

➤ 问题描述:

一次编译运行时报错: Debug Assertion Failed, 错误位置 dlgdata.cpp 的 Line 40 这个路径是不存在的。

➤ 解决方法:

导致该错误, 即断言失败, 原因是为一个控件进行变量绑定后, 直接在对话框上把控件删除了, 而关联代码却没删除”。由于项目较大, 有在每个模块完成更改是做备份, 因此回复最近的一次备份重做修改。并注意: 删除控件前, 应先取消对控件的变量绑定, 然后再删除控件

<5>文件夹复制时未创建根目录

➤ 问题描述:

整个文件夹复制时, 结果不是将源文件夹复制到目标路径, 而是将其中的内容复制了过来。

➤ 解决方法:

在 CopyDirectory 自定义函数中增加对目标路径的字符串处理, 识别 Source 最后两个‘\’之间的部分为要复制的文件夹的名称, 加载 Dest 之后加‘\’再复制。

</王紫薇>

<陈牧乔>

(5) “文件清理”模块实现时遇到的问题

1) 无法在 List 控件中嵌套 CheckBox 控件

需要在 List 的每行中添加一个 CheckBox, 以便使用户对需要清理的文件类型以及主页面中扫描到的文件进行选择, 但是始终找不到嵌套添加的方法。后来经过一番搜索, 发现只要在页面初始化时对 List 的属性进行设置, 使用 SetExtendedStyle(LVS_EX_CHECKBOXES)就行了。

2) 文件类型的选择状态无法在文件类型选择对话框和父对话框之间传递

最开始选择采用一个共享的数据结构让父子对话框共同访问, 并创建相应的方法, 这样便于数据的更改。但是由于程序的一些模块特性以及 C++的数据作用域的限制, 经过多次修改, 在编译过程中不是重定义就是找不到定义, 总之程序一直不能正常运行。后来退了一步, 只对指示文件类型选择状态的三十几个 BOOL 变量进行传递, 方式是父子对话框共同访问子对话框的成员变量。这样做带来的问题是造成了一定的代码冗余, 但是还是将这个问题解决了。

3) 无法正确匹配指定文件类型的文件

本来想截取文件的后缀直接进行判断, 但是许多垃圾文件并不是某特定的后缀决定的, 这样就要 if...else 许多步, 逻辑非常混乱。后来选择使用正则表达式, 可以使用统一的方式进行匹配, 大大提高代码的逻辑清晰度。难度在于正则表达式的书写, 非常容易出错, 需要足够细心。

(6) “详细信息”模块实现时遇到的问题

1) 无法插入列表表头

最开始不太熟悉 MFC 的一些控件的使用, 一直找不到代码哪里出错, 后来才发现是列表的属性设置错误, 将 View 的属性从 List 改成 Report 即可。

2) 无法插入正确的进程描述信息

首先是格式问题，在测试时 `cout` 可以正确输出结果，但是插入表中的显示就是乱码，被这个问题困扰了很久。后来对描述信息的数据类型进行研究，发现可能由于 C++ 中特有的 `string` 类型是 C 语言无法识别的，为了填入表中在进行格式化转化时出了错。解决方案是先用 `string` 的 `c_str()` 方法将 `string` 转化成 `char*`，再使用 `_stprintf_s` 转化为 `TCHAR*`。

3) 调用排序函数时，比较函数的实参和形参类型不匹配

`List` 的 `SortItems` 方法有一个参数类型是 `PFNLVCOMPARE`，应该是重写的比较函数的类型，在编译时一直报形参实参不匹配的错，检查了许多遍都不知道哪里出了问题。`PFNLVCOMPARE` 是一个宏定义的函数类型，在查看源码的过程中带来了不少的麻烦。最后发现错误出在函数在类中的声明上，将定义在 `public` 中的 `int CALLBACK MyCompareProc` 改为定义在 `protected` 中的 `static int CALLBACK MyCompareProc` 就不报错了。

4) 在未选择进程的情况下查看虚拟内存信息，程序卡死

这种问题需要增加错误处理机制，在使用虚拟内存查找函数前先对选择的存在性做判断，如果未选择则弹窗提示，不再进行下一步骤。

5) 查看已关闭却仍然在列表中的进程的虚拟内存信息，程序卡死

这个问题的造成是由于进程列表是手动刷新的，有可能列表中的进程已被关闭，但是列表没有刷新。如果用户选择了已经关闭的进程进行虚拟内存信息的查看，就会出现错误。处理方法也是进行预判，先对用户选择的进程进行状态的判断，如果进程仍然处于运行状态则进行下一步，否则弹窗提示刷新并退出。

6) 结束进程功能的错误

最初选择调用系统命令 `system()` 直接对 `PID` 进行操作，但是出现了访问错误。后来使用 C++ 的命令，先 `OpenProcess()` 再 `TerminateProcess()`，就可以实现结束进程功能了。

</陈牧乔>

<罗薇>

(7) “内存”模块实现时遇到的问题

在使用 `GlobalMemoryStatusEx(&statex)` 函数获取虚拟内存和可用虚拟内存两项信息时，发现结果错误。

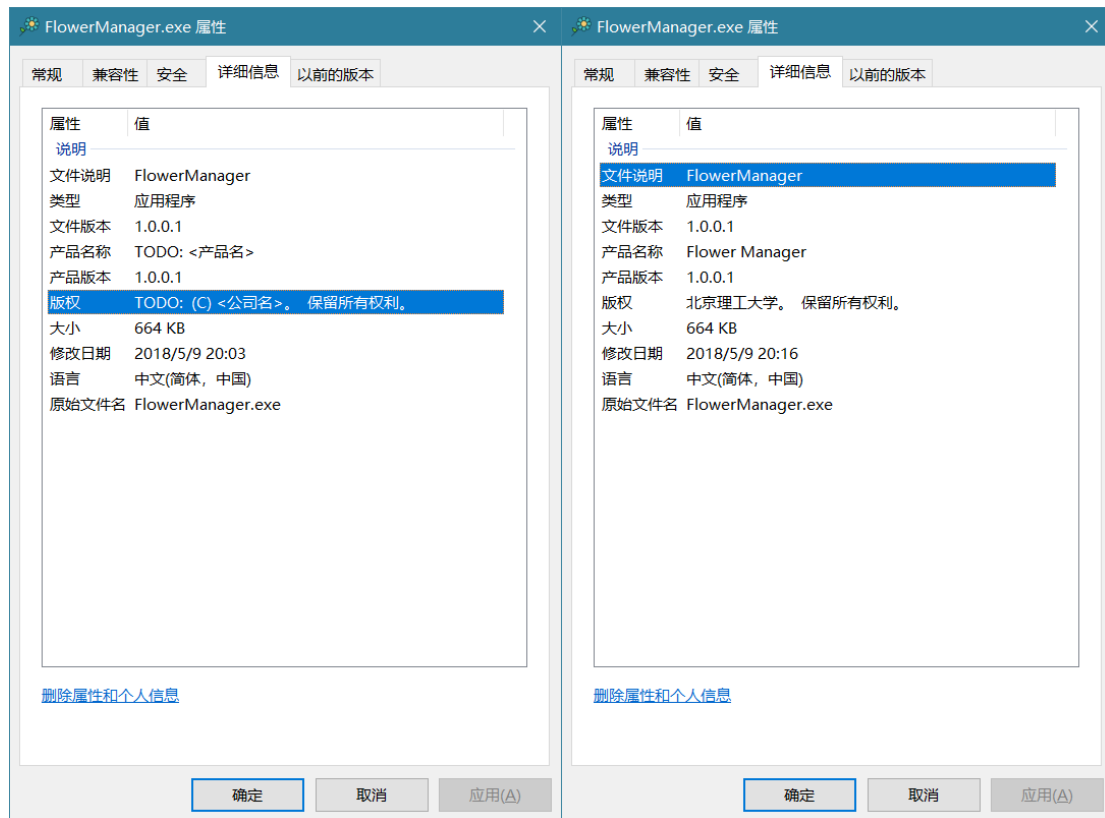
解决方法：将 `statex.ullAvailVirtual` 转为 `TCHAR` 型再填入表格。定义一个 `Tchar` 型的数组 `sz[MAX_PATH]`，再调用 `StrFormatByteSizeW(statex.ullTotalVirtual, sz, MAX_PATH)` 将转换好的数据写入 `sz`，最后将 `sz` 填入 `List Control` 控件。虚拟内存信息即可正确获取，可用虚拟内存的处理同上。

</罗薇>

<侯思凡>

(8) 项目信息填写未完全

如下，左图为修改前的信息，通过修正资源文件的代码完善了正确的详细信息：



</侯思凡>

2. 小组成员的心得体会

<侯思凡>

侯思凡：

1) 宏定义的便利之处

由于各个模块由小组成员独立完成，最终项目便需要集成。集成工作主要是，重新建立一个 MFC 项目，重新建立.rc 文件，代码只需拷贝即可。为了完善资源文件和代码的统一，资源文件的 ID 与组员独立设计的 ID 一样即可。资源文件的 ID 被宏定义在 Resource.h 中，每次新建项目 ID 号的值会有所不同，但是引用宏定义不需要考虑这些问题。

2) MFC

MFC 以 C++类的形式封装了 Windows 的 API，大大减少了程序开发人员的工作量。程序员只需在图形界面添加控件，在相应的类文件中编辑代码即可实现对控件及控件内容的控制。

3) 透明性

在无须了解 MFC 如何实现界面显示的条件下，程序员只需掌握如何添加控件和编辑代码即可实现一个图形界面的项目。对于 MFC 到底是如何实现图形显示的，我们并不需要关心，透明性是不同层面程设很好的隔离，也是很好的起点。使得我们无需关心太多底层，即可从某一高度开始设计实现项目。

4) Deadline 是第一生产力

本次大作业本小组四个人齐心协力共同完成，是本科目前以来较为复杂的一个项目。在队友

的鼓励和互助下，我们完成了共同设计、独立完成模块、集成工作等项目阶段，自己的能力有了极大的提升，代码能力又有了新的飞跃。

</侯思凡>

<王紫薇>

王紫薇：

和舍友兼小组成员一起从无到有完成 Flower Manager，从定选题、大框架到独立任务，相互交流，每个过程都还是历历在目。“进称查看”、“文件浏览、复制”部分由我负责，都是在结合课内实验的基础上，将所有分块的知识点整合到一起，运用 MFC 框架呈现一个完整的结果。期间遇到了不少问题，字符串类型转化的、框架结点元素获取的、C++构造函数的，于是对应到代码，一步一步用 MessageBox 这种最原始的找原因。似乎每一次看自己的代码，基于更深的理解，都能重新发现自己代码中不足和优化的地方，仅仅验收之后到写完报告又修改了 1/3 的代码。至今得到了自己较为满意的结果。

通过这次的大作业，我不仅对课内实验部分的知识有了更深的理解，还将他们灵活运用了面向对象的部分知识，将不同的模块整合到框架中。期间也加深了团队的合作。记得验收之前自告奋勇找了代表宿舍的 LOGO 加在程序中，打开后能真正顺畅的使用，真的觉得付出都是值得的。

</王紫薇>

<罗薇>

罗薇：

（1）MFC 界面与功能实现

MFC 窗口对象是一个 C++ CWnd 类的实例，是程序直接创建的。可在窗体中填充控件，然后结合项目需要在代码中添加事件响应程序或显示信息，简化了工作量。

（2）从需求出发

本次实验是从零开始实现的项目，我们做的是关于进程、文件、内存等内容的管理器，功能性很强。因此在着手编程之前，我们先对将要完成的功能进行了初步需求设计，在主窗体中设计多个标签页，分别实现相应的功能。包括在每一个标签页里面，需要获取哪些信息、填充哪几个控件、以及界面布局，这都是我们需要提前考虑的。

（3）编程规范

本次项目中不同组员编写的标签页中都含有大量控件，因此就牵涉到了很多变量名的问题。如 CListCtrl 类变量，我负责的部分就有 m_list_page5 和 m_ListSystemInfo，如果按数字顺序命名为 m_List_1 和 m_List2，就很有可能会和其他组员的变量重复，在后期整合时带来不必要的麻烦。因此我们在小组共同完成项目时，应保证自己负责部分的编程规范，除了变量取名外，代码整洁、程序可读性等也是需要注意的。

（4）分工合作

在本次项目中，从总框架的搭建、到小组成员独立完成功能页、再到最后的代码整合，组内的协调是必不可少的。前期小组共同商榷项目实现目标，初步规划；中期结合之前讨论的方向选择各自将要完成的内容，分模块进行，模块间互不干扰，仅分别于主窗体进行关联；后期统一整合为一个完整的项目，再一起提出完善项目的建议，直至最终完成。小组成员经常

一起写代码到深夜，除了项目本身带给我们的知识，我觉得大家相互鼓励和帮助的氛围也是我从中的得到的一大笔财富。

</罗薇>

<陈牧乔>

陈牧乔：

这次操作系统的综合实验，从最开始的举棋不定犹豫不决，到后来确定方案开始艰难探索，再到最后完成了一个功能相对完整的资源管理器，可以说是与小组每个成员之间的通力分工合作是分不开的。与之前的实验只实现某几个特定功能不同，这次综合实验需要多个功能模块的相互协调与集成，同时还需要与界面的交互，需要对面向对象编程思想的深入理解和综合运用。

我们选择 MFC 界面编程，因为它封装了许多 Windows API，可以更直观地对界面进行编写，降低了界面的开发难度。由于它是以 C++ 类的形式封装的，而之前的操作系统实验也多是由 C/C++ 编写，这样就可以与之前的课程内容有更紧密的联系。

愿景十分美好，但是初次上手不免遇到很多问题。由于是从应用的角度进行 MFC 相关资料的查找与操作，最开始总是出现控件无法显示等问题。在一次次碰壁和探索中，我对界面与代码、控件与动作的联系有了更深刻的理解，之后渐渐熟悉并得心应手起来。回想之前做过的 C# 和 Android 界面项目，虽然具体的实现有很大的区别，但是基本思想还是融会贯通的。之后遇到的一大难点是将调用的系统函数的结果显示在界面中。之前的实验使用的是控制台程序，对于输出内容的格式要求没有那么严格，而在 MFC 界面程序中则不是这样。我花费了很多精力在字符串格式的转换上，包括 TCHAR*、CHAR*、CString、string 等等字符串类型的相互转换。

在这次实验中我感受到，编程的具体实现固然重要，但是更重要的是编程思想的理解与快速学习的能力。编程具体实现的不熟悉带来的障碍可以通过快速学习在短期内解决，到最后真正制约项目进度的，是编程思想理解的欠缺，而这恰恰是需要长期编程经验积累的。在实验过程中遇到的两个对话框参数传递问题，涉及到类的成员变量、全局变量的数据共享和使用，这就需要面向对象编程思想的深入理解与熟练使用。这一问题当时困扰了很久，很难通过搜索的方式解决，只恨自己的面向对象编程思想没有理解到位。

这次实验中的一些部分确实是两位同学坐在一起共同完成的，但是要求每行代码具体到个人，这就给这种完成方式带来了一些困扰。之前遇到小组项目总是以这种方式完成，这一次也就部分延续了这一传统，但是这一次暴露了一些问题。我想在今后的学习和工作中如果遇到团队合作的部分，应该尽量将每个任务模块具体分配到个人，这样更有利于工作的量化与责任的划分。

</陈牧乔>