第一章　简介

1.1　Macintosh方式

现在，Macintosh系列产品取得了巨大的成功，受到了高度的评价，且物有所值，因而被许多厂家模仿。从PowerBooks到功能强劲的Quadra工作站，Macintosh是许多人的首选目标。然而，在1984年，第一台Macintosh出现时，人们为之困惑不已：她不象他们曾见过的任一种计算机一棕灰色的盒子上有一个小银屏，并接着一个鼠标。人们称Macintosh是一个玩具，因为她有一个图形用户接口（graphical user interface，简称GUI），而图形在当时不是计算机交互的通常方式。

自从80年代起，我们已经走过了一个漫长的道路。Microsoft、惠普，最后是IBM都已走进了Apple的大门。GUI和鼠标日益普及，Apple支持的硬件标准，例如NuBus和SCSI（Small Computer Systems Interface）已经在工业界广为传播。Apple的反对者消失了，代之以公司推销员声称他们公司的计算机工作起来“就像一台Mac一样”。如果Mac仅是另一种漂亮的接口的话，他们也许是正确的。

Mac的优雅、易用及强大功能源自一种界面（interface）、工具箱（Toolbox）和资源（Resources）的完美的组合。

· **Macintosh界面**：任何使用过Macintosh的人都对Mac的界面十分熟悉。下拉菜单、可移动的窗口、滚动条和图形，所有这些的组合使Macintosh成为现有计算机中交互最友好的一种。

· **工具箱**：Macintosh ROM中定义了综合性的例程（routines），它既驱动了界面，又允许软件设计人员开发出功能强大、使用方便的应用程序。Macintosh的一个突出的特点是具有良好的用户界面，而这一点正是得益于它丰富的系统软件。

· **资源的使用**：Macintosh上的所有软件的建立模块集 — 资源，在编程文件中的一系列模板里存储着编程信息，简化了Mac程序的创建和修改。

以上三种成份：界面、工具箱和资源，组合在一起，使Macintosh成为已有计算机中最多才多艺的。一些高级的功能，如影像的录制和重放（一种“Mac TV”）、虚拟内存和个人文件分享等，不仅仅是90年代新型Mac的特点，而且甚至对于十年前生产的Mac也可以通过装载Macintosh操作系统的新版本来获得这些功能。由于Mac机种的不断进化和提高，这种对最初Mac机仔细的设计计划在现在看来已经取得了很好的效果。

Mac未来的编程者们如果想为Macintosh写出成功的应用程序，就必须理解界面、工具箱和资源是如何共同工作的。首先，让我们看看三者中最常见的一部分：Macintosh用户界面。

■ Macintosh图形用户接口（界面）

Macintosh总以它复杂的用户界面给新用户留下深刻的第一印象。图1-1中显示出了Mac“外貌”中的一些独具特色的要素。因为初学者能够直观地理解和使用Mac应用程序中的窗口和菜单，所以可以较快地掌握。Macintosh界面代表了一个显著的改进，不论是对基于命令的界面（如“MS-DOS”），还是基于窗口的界面（如建立在MS-DOS之上的Microsoft Windows）。界面中的每一个要素 — 窗口、菜单、对话框和图标都有一个特定的功能与之相联，并且配有针对每种要素的正确使用方法的广泛指导。使用了System7.0之后，界面看上去会变得更加令人兴奋，它不但有色彩明丽的图标，而且当你用鼠标在你想知道的某东西上按一下时，会出现友好的文字说明块即Ball。

当然，光有漂亮的图形是不够的。Mac界面真正的优点在于它是如何被创立的。界面的每一部分都是由Macintosh ROM里的一系列例程产生的。例如，你能够调用Macintosh ROM里的一个例程来创建一个应用程序的窗口。

这些支持界面的例程，如创建窗口、控制打印机和绘制菜单等总起来称为**Macintosh工具箱**。

Macintosh图形用户界面中有一个重要部分，叫做Finder，实际上它不是系统软件的一部分，只是一个应用程序。Finder负责跟踪文件并管理用户的桌面。

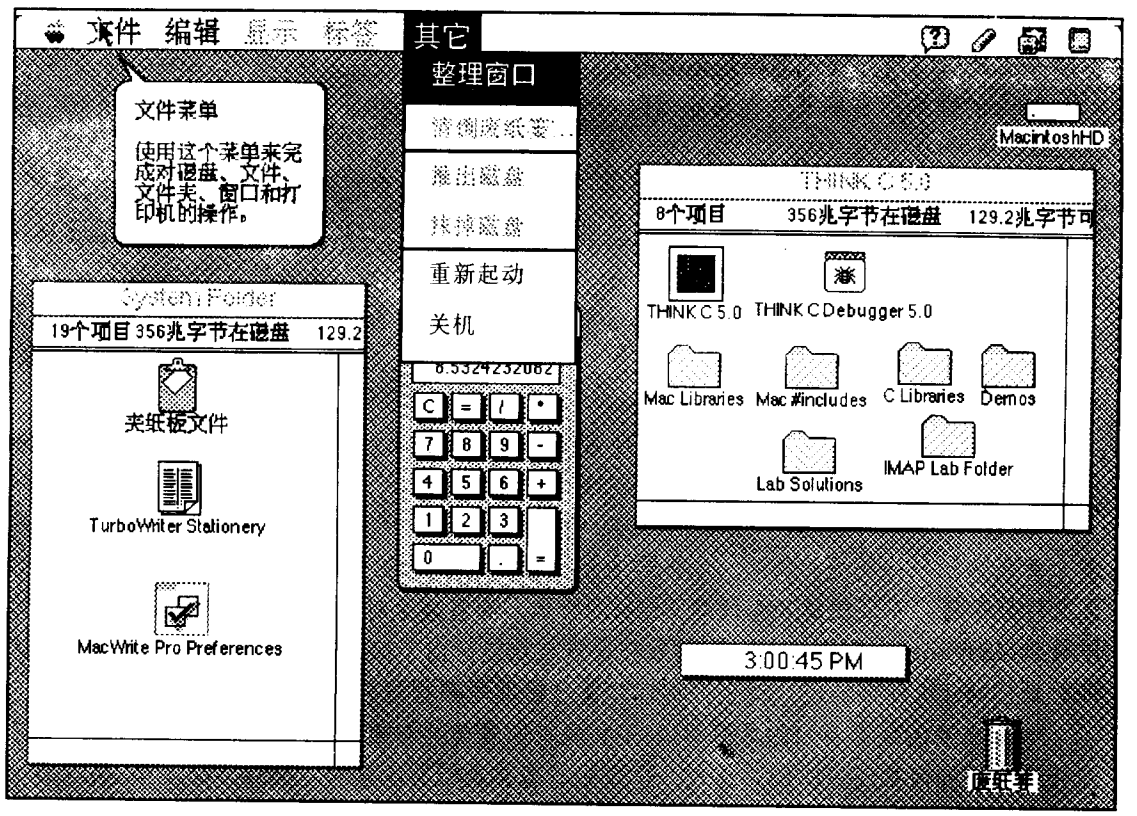

图1-1 Macintosh桌面

■ Macintosh工具箱

Macintosh工具箱提供一系列过程，应用程序可以调用这些过程来实现各种用户界面。工具箱保证了用户界面的方便和一致性，也减小了应用程序的规模，缩短了开发时间，而且使应用程序更易于维护。

使用工具箱来创建你的应用程序的结果是使该应用程序具有一个独具特色的Mac外观和感觉。对于大部分应用程序来说，操作是通用的，如剪贴、拷贝和粘贴等，这使学习一个新的应用程序变得十分简单。

工具箱的例程按功能组装进了管理器（Managers），每个管理器对Macintosh环境中的一个部分负责（如图1-2）。

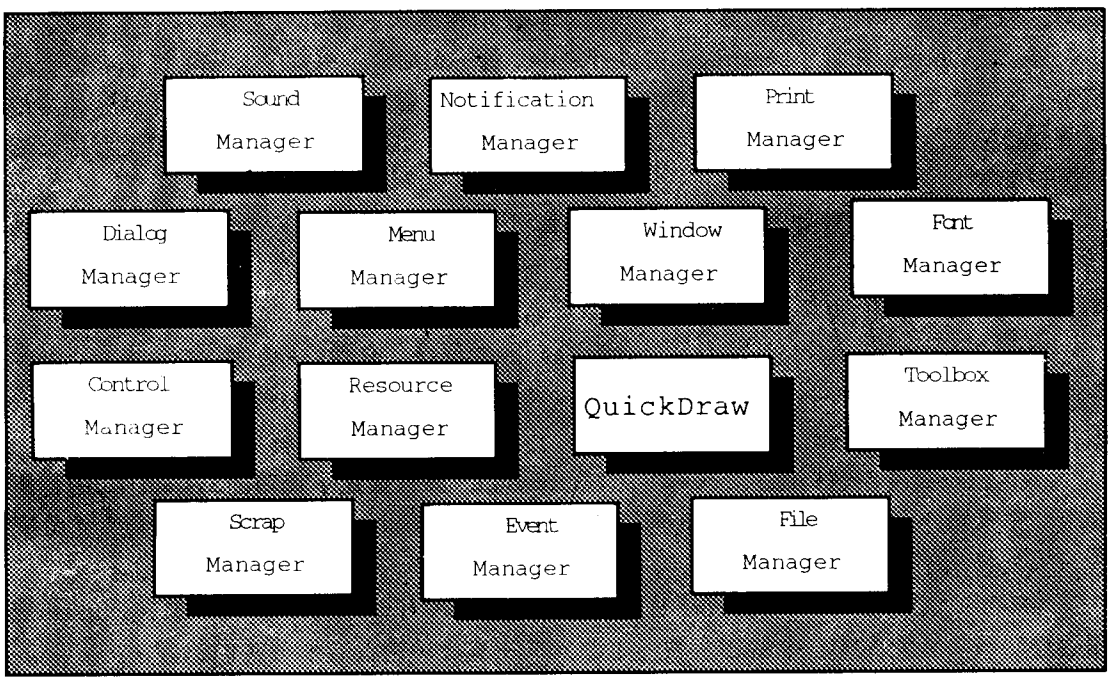

图1-2 工具箱的组成部分

Macintosh工具箱经历着不断的更新和修改，每次新的系统修正都会给你一些耀眼的新工具，并能够和其它旧的一起很好的工作。随着新的例程被添加进工具箱，Apple扫清了与旧例程之间的问题。

这一切把我们带进了System7.0。

Apple已经提供了崭新的工具箱调用程序，彻底改进了编程人员控制文本、声音、动画甚至Mac屏幕上的实时影像的能力。这些功能都能仅用几行编码就植入你的程序。同时，低层例程使你能够很好地把这些新特征组合在一起。本书后续章节中将详细讲述Mac在针对这些新提供的系统工具编程中的改革。

Macintosh的图形界面和工具箱是使Mac独领风骚的两大特点。第三个特点就是资源的

概念。

■ 资 源

如果工具箱是生成Macintosh界面的例程库，那么资源就是你的程序用来执行这些库调用的数据。GetNewWindow( )，这是用来创建一个新窗口的工具箱例程，它要求你指定窗口参数，诸如尺寸、位置和窗口类型等。要达到这一目的，你可以创建一个包含这些信息的资源，并把该资源传送给GetNewWindow( )。GetNewWindow( )用这个资源就可创建出所要求的窗口。

资源有多种类型，每种类型与Macintosh界面上的一个特定要素相联系。例如，一个类型为WIND的资源包含了创建一个窗口所需的所有信息。也许有许多的资源都是关于窗口信息的，但是仅有一个WIND类型，它对所有Mac应用程序来说都是同等的。

资源被集成起来置于Macintosh的设计之中。每一个Mac应用程序文件也许拥有数十个资源。这大大简化了应用程序设计者的许多任务。比如，资源使一个程序在不同地区的本地化变得很简单。比方说你要在法国出售你的程序，那么，用法文版的资源代替英文版的就相对要容易了许多。

在开发驱动Macintosh界面和硬件的复杂编码时，资源也很重要。因为他们能够简单地从一个程序拷贝到另一个，菜单和对话框无需再创建一遍。你在建立一个程序集之后，创建新程序也许就从你较满意的程序集中的一个剪贴部分开始。

为了编辑资源，Apple开发出一个名为ResEdit的程序，它使你能够编辑Macintosh基本程序中的任一资源。你可以使用ResEdit去探索其他Macintosh应用程序，甚至是系统文件！因为这些资源是已完成的应用程序的一部分，所以你无需重新编译就可以编辑。

我们在Mac基本教程中广泛使用了2.1版ResEdit，并且描述了成功的System 7.0编程所要求的新资源。即使你从未使用过ResEdit，你也将发现Macintosh基本教程中的命令十分完善，而且易于学习。

Macintosh界面、工具箱和资源是我们使用THINK C和ResEdit创建独立的Macintosh应用程序时将要介绍到的三个紧密相联的主题。下一节将讨论我们学习它们的方法。

1.2 关于本书

大多数Macintish参考书，如Inside Macintosh和Macintosh Revealed，都是面向那些已经掌握Macintosh编程的人的好书。然而，对于初学者来说，它们可能太难了一些。Mac基本教程将为那些刚刚学习Mac编程基础的人架起一座桥梁。

我们的目标是帮助你书写结构正确的Mac应用程序。如果你已习惯于在基于MS-DOS或UNIX的计算机上编程，那么Mac基本教程是你开始Mac编程的理想起点。过去，我们的时间花在了像PDP-II和VAX-11/780这样的机器上的UNIX编程，后来我们也用了很多时间在IBM-PC及其兼容机的编程上。因此，在写Macintosh编程基本教程时，我们的脑子时时在为读者您考虑。

如果你已经在Macintosh上编过程，但却未接触过System 7.0，那么你将会在教程中找

到许多有用的代码，帮助你执行你的应用程序的新System 7.0友好版本。

■ 你需要知道些什么

阅读本书有两个先决条件，即你应该已经具备基本的Mac经验：你应能运行Macintosh应用程序并对Mac用户界面有一个良好的感性认识；另外，你还应有一些编程语言方面的经验，例如C、Pascal或BASIC等。如果你没有编程经验，或者你的计算机语言技巧很糟，那不妨找本有关C语言方面的书，做为本书的补充。

本书示例均用C语言，在THINK C开发环境下写成。然而，我们总的方法是强调为针对Mac工具箱编程的技巧。这些技巧，不论你将来打算用哪种编程语言或环境，都会使你受用无穷。

■ 为何选择THINK C

Mac编程者可以选择的开发环境有许多种，如Macintosh Progarmmers's Workshop（MPW）。它是一套由Apple编写并经销的复杂而又强大的开发系统。大多数Apple的内核是在MPW下开发的，因而许多大型 Macintosh 软件开发商都以MPW为他们的第一选择。MPW为软件开发提供了齐备完善的环境。其基本系统包含一个编辑器外壳，让你能够编辑你的代码，同时可编译、连接和执行复杂的命令。在MPW下你可以做任何事情，但它的确不适合于初学者。另外，除了要学习编辑器和外壳，你必须安装配置你所选择的编译器，当然，还要付钱。你可以买C、Pascal，甚至FORTRAN的MPW编译器。MPW对于复杂的多语言开发工作十分理想，但不适于学习Macintosh编程。

THINK C是一套功能强大而且友好的开发环境。它有扼要、准确的文档。对于那些不可避免的错误，它有市场上最好的调试实用工具。对于专为System 7.0编程的人员，它也给予十分优秀的支持。

最后一个因素，那就是THINK C的价格十分合理。

■ 使用THINK C

THINK C是一个集成开发环境。源代码编辑器遵从所有标准的Macintosh风格，因而非常易于使用。它的编译器很聪明：它跟踪你当前的工作文件，注意那些上次编译后已被修改过的文件，THINK C只重新编译那些必需的文件。

THINK C有一个考虑周密的Macintosh界面。例如，要创建一个独立的应用程序，拉下**Project**菜单并选择**Build Application**命令即可。安装也十分简单：取出源盘，把文件拷贝到你的硬盘，然后就可以使用了！

THINK C还具备集成调试工具，使你能够试运行你的程序同时在另一个窗口监视其过程。该调试工具还能与其他Macintosh调试工具一起工作，诸如MacsBug和TMON。

■ THINK C的内部

项目（project）文件是Symantec的C和Pascal开发环境的独特之处，它除包括你的应用程序的名字，还包含所有源代码文件的名称。同时项目文件还包含每个源文件的编译信息，例如编译成功后的代码长度（如图1-3）。

THINK C带有类似于MacApp的类库，MacApp是Apple原装用户界面例程库。THINK C的调试功能是无可匹敌的。你能够使用THINK C去编写程序，并充分利用Mac OS的最高级的特点。所有这些特点都是以Apple设想的方式给予支持。THINK C还提供例程支持扩展的Apple的**HyperCard**或Silicon Beach的**SuperCard**。

THINK C同时带有全套的实用程序，包括前面提到的资源编辑器ResEdit，以及基于各种Mac项目上的有用代码，包括文本编辑器、控制面板设备（Cdevs）和桌面附件等。

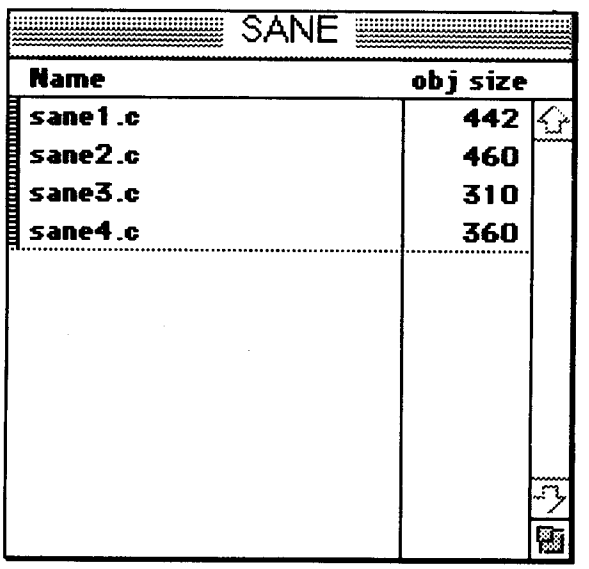

图1-3 THINK C项目窗口

■ 编写Macintosh应用程序

大多数Macintosh应用程序都使用一个基本结构（图1-4）。它们都从对支持Macintosh用户界面的工具箱的数据结构和例程的初始化开始，然后，应用程序进入一个事件循环，耐心地等待用户做些什么：按一下键，移动一下鼠标或其他一些动作。应用程序之外的事件也同样被检测：桌面附件可能被使用或插入磁盘等事件。不论Macintosh程序有多么复杂，这个简单的结构仍被保持着。这种以用户为中心的设计，其基础是："事件驱动编程模式"（event-driven programming model），使用户感到你的应用程序随时都准备为他服务。

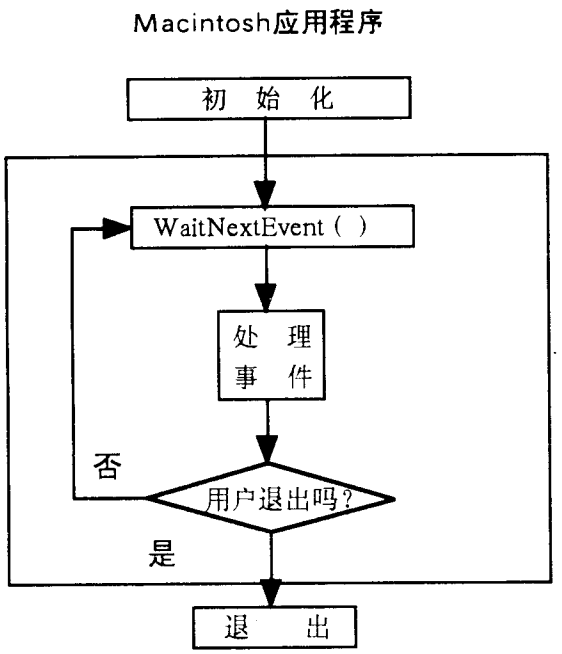

图1-4 Macintosh应用程序的一般流程

Macintosh 编程基本教程的核心里有 7 个范例应用程序。每一个都建立在基本的程序结构上，提供对工具箱越来越复杂的高级的使用，每一个新章节建立一个更强大的基本程序结构的实施方案。第三章的程序将示范如何创建窗口，并在窗口里面绘图。第四章说明了如何处理事件（包括 Apple 事件）。第五章实现菜单，第六章使用交互对话。它们被设计用于说明 Macintosh 工具箱的不同部分。

每个 Mac 基本教程的例子程序都尽可能完整，并且每个程序清单都有大致的解析。每章包含完整的指令和数据，供输入、编译和执行 THINK C 的程序。

1.3 准备知识

这一部分内容介绍了本书将要用到的工具软件，同时还讲解了几个在Mac平台上采用C语言开发的问题。

■ 安装THINK C 5.0

THINK C是本书的编程环境，因此，我们要学会THINK C的安装。它的安装实现很简单，即把THINK C从磁盘上拷贝到硬盘上即可。我们要注意的是在拷贝之前应完成几项准备工作。

第一件事是备份你的THINK C源盘，然后用你的备份盘进行安装。

第二件事即为查看你的硬盘，看看是否已有低版本的THINK C，若有则删除。注意勿删除你的源文件或项目文件，你可以把它们存在一个临时文件夹中或拖至桌面上，请把新版THINK C安装完毕后，再拖回。在THINK C文件夹同级目录下创建一个名为Development的文件夹，这个文件夹将用来保存你的所有THINK C源文件。最后，就是要确保你的硬盘可用空间至少还有5M，这时你才可安装THINK C。

插入 THINK C 2 号磁盘，打开磁盘标识文件夹，窗口中的那三个文件均为压缩文件（Self-extracting archive）。点按其中一个文件Headers & Libs. sea就会弹出一个对话框 。

对话框要求你指定压缩文件释放“容器”，这里你把此“容器”定为刚建立的 Development 文件夹，注意要使对话框中的弹出菜单内容为Development。一旦确认放置“对象容器”是Development文件夹，点按**Extract**按钮，就出现一个自动释放窗口（AutoExtractorTM），同时告知你还有多少个文件待释放。直到此压缩文档中的所有文件都已释放后，再去打开下一个文档THINK C 5.0 Demos. sea重复上述操作，最后打开DALcdev Tools. sea文档。

推出2号磁盘，插入THINK C的 3号磁盘，把磁盘中的两个压缩文档释放到Development文件夹中，先释放 THINK Class Library 1.1.sea 文档，再释放 TCL 1.1 Demos. sea文档。推出3号磁盘，插入 4 号磁盘，重复上述操作，把磁盘中的 Resource Utilities. sea 和 THINK C Utilities. sea 文件释放到 Development 文件夹。

完成上述工作后，推出4号磁盘，插入1号磁盘，把磁盘中的两个文件：THINK C 5.0 和 THINK C Debugger 5.0 拷贝到 Development 文件夹中的 THINK C 5.0 文件夹。到此，我们就完成了 THINK C 5.0 的安装工作。如图1-5所示：

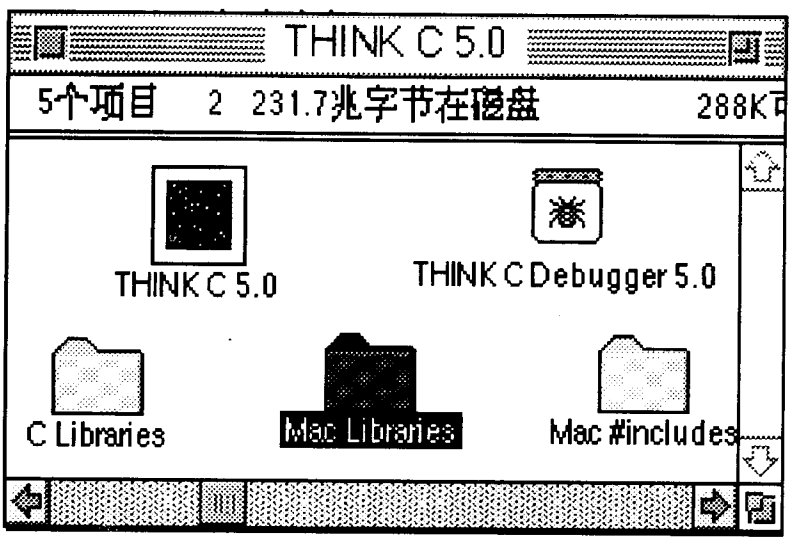

图1-5 安装好的 THINK C 5.0 文件夹内容

■ 访问工具箱的C语言

每一个Macintosh都内嵌一个Mac工具箱，它是由1400多个例程构成的一个功能程序库。这些例程可以绘制窗口，控制菜单甚至还可以按时更换日期等等，正是因为有了这些例程，才使得Mac用户界面保持一致性和直观性。不论你是在使用Claris MacDrow还是Deneba的Canvas时，你总是要用到这些例程。当你点按菜单并拖出一个相应的菜单时也都是在调用同一个例程。这也就是为什么每一个应用程序的菜单都很类似的原因所在。我们对滚动条窗口，对话框示警框等等也采用同样的处理方法。

如果你已在Inside Macintosh资料书中阅读过工具箱调用的内容，你会注意到它的例子都是用Pascal语言写成的，例如GetNewWindow( )函数的调用。

```
FUNCTION GetNewWindow (WindowID:   INTEGER;
                       WStorage:    Ptr;
                       behind:       Windowptr ); WindowPtr
```

调用语句均以函数（FUNCTION）或过程（PROCEDURES）为标识符，FUNCTION返回数值，而PROCEDURES则不然。如上例GetNewWindow( )函数返回一个Windowptr类型的值，下面是一个程序中调用GetNewWindow( )的例子：

```
WindowPtr   myNewWindow, myoldWindow;
Ptr   mystonage = nil;
int    myWindowID = 400;
myNewWindow = GetNewWindow ( myWindowID, myStorage, myoldWindow );
```

用Pascal语言调用，GetNewWindow( )函数返回一个WindowPtr类型的值，在编程中，我们把GetNewWindow( )的返回值myNewWindow变量定义为WindowPtr类型，Inside Macintosh中大部分数据类型在THINK C中都是有效的。（如果你有兴趣的话，你可以查看THINK C5.0文件夹中的Mac #includes文件夹内容，那里对所有类型都进行了定义）。差异如下：

Pascal数据类型		THINK C语言中的等效类型
INTEGER	2 字节	Short
LONGINT	4字节	long
CHAR	2 字节	short
BOOLEAN	1字节	Boolean
STR255	0 ~ 255	unsigned char[256]

我们用布尔型变量举例比较：

Pascal语言：

```
FUNCTION Button :  BOOLEAN;
```

C语言：

```
Boolean     isButton;

isButton = Button( );
if  ( isButton == TRUE )
      SysBeep (20);
```

Pascal不区分大小写，而C语言却区分大小写，Boolean与BoolEAN是不相同的，对于编

程人员来说，THINK C为这种Pascal类型的使用提供了便利。即使就像Button( )，虽无形参，但仍要求使用括弧，否则编译就通过不了。

你还可以直接把数值传递给调用函数，比如前面所举的GetNewWindow( )例程的调用就可写做:

```
WindowPtr   newWindow, oldWindow;
myNewWindow = GetNewWindow( 400, nil, oldWindow );
```

这样也能正常运行，为了提高可读性，建议你采用 #define 语句来定义参数常量。

· #include, #define和extern语句

#include "文件名"

#include语句在编译时，则是将其指定的文件名用相应的源代码替换掉。如:

```
#include "BigFile. h"
```

#define标识符字符串（常量）

#define语句则是在程序中用前者替换后者，提高可读性。如:

```
#define MAXFILES 20
```

extern变量类型变量名

这是一个存储类区分符，用于变量说明，所定义的变量为外部变量，即在整个程序的执行过程中都存在且保持它们的数值。如:

```
extern Boolean done;
```

在本书介绍的例子中因只有一个源程序所以并未用到extern说明语句，如果你的文件太大，想把它们分成几个小文件时，你就要用外部变量在其中做通讯作用。

■ C字符串和Pascal字符串

C语言与Pascal在处理字符串时采用了不同的方法，Pascal字符串的起始位为长度位（length byte），它表明这个字符串的长度。C字符串则是在其末位加上一个 "0" 值作为字符串的结束标志，例如: "Hello, World!" 字符串，Pascal与C的实际存储格式为:

Pascal语言:

13	H	e	l	l	o	,		W	o	r	l	d	!

C语言:

H	e	l	l	o	,		W	o	r	l	d	!	0

Macintosh工具箱例程采用Pascal字符串，即str255数据类型。由于1个字节的数值范围为0～255，所以Pascal字符串长度最长为255位（不包括长度位）。

在THINK C中采用Pascal字符串就比较容易，THINK C编译器会自动地把以“\P”开始的C字符串转换为Pascal格式。例如:

```
DrawString ("\PHello, World!");
```

你还可以调用CtoPstr( )和PtoCstr( )例程来对C与Pascal格式进行转换。这些例程仅为THINK C所提供的，并不是Macintosh工具箱中的。

■ 编程方面的几点补充

在编程时应考虑到程序的易读性，在保持你的风格的同时要尽量采用标准结构格式，下面所举的例子就是保持花括弧（{}）对应在同一列来提高易读性。

```
main( )
{
    int i;

    for (i = 0; i<10; i++)
    {
        DoNastyStuff( );
    }
    wrapItup( );
}
DoNastyStuff( )
{
  DoOneNastyThing( );
}
```

有的编程人员也喜欢采用如下方式:

```
main( ){
```

```
int  i;

for  ( i = 0; i < 10; i++ ) {
   DoNastyStuff (  );
}

DoNastyStuff (  ){
   DoOneNastyThing (  );
 }
```

不论你采用什么样的格式，应保持住它的一致性。

我们在定义变量和例程时，也应考虑到易读性，建议遵循以下几点规则:

· 如果你给变量命名，第一个单词为小写，后面的单词均以大写第一字母来分隔开来，如:

i, myWindow和bigDateStructure。

· 如果命名的变量为全局变量，就在变量名前加一个“g”字母，如:

gCurrentWindow 和gDone。

· 如果变量是通过#define语句来定义说明的，则应在该变量名前加一个“k”字母，如:

#define kNumRecords 20

· 如果你命名的对象是例程，那么构成例程名的那几个单词应以大写字母分隔开来，如: MainLoop( ), DeleteEverything( )和 PutThatDown( )。

■ ResEdit

你会在Development文件夹中的THINK C 5.0 Utilities文件夹中找到ResEdit，这是Apple较为流行的资源编辑器。点按ResEdit文件夹，从File菜单中选择Get Info菜单命令相应地出现一个ResEdit Info窗口，如图1-6所示（确保你自己使用的ResEdit版本高于2.1）。

利用 ResEdit 编辑资源将能大大地减少内存开支，在 ResEdit Info 窗口右下角的自定域，标明至少减少了 1500K 内存开销。

你可以利用ResEdit来创建和制作你所需要的资源。下面介绍几种本书将要用到的资源。

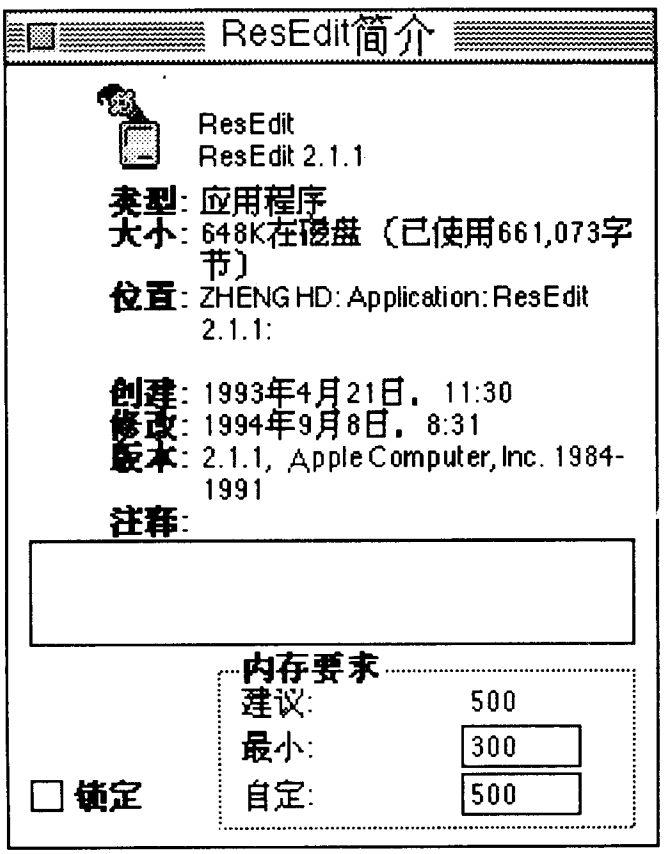

图1-6 ResEdit信息窗口

■ 资 源

正如我们前面所介绍的那样，资源中存储着程序的大部分描述信息。我们可以定义它们的类型、ID值以及它们的名字。

每一个资源都对应一个类型，每一种类型都具有特殊的功能，例如 WIND 类型的资源就包含了创建一个窗口所需的所有信息。MENU 资源描述的是屏幕顶端菜单条中显示的菜单。图1-7 展示了你在本书将要用的资源类型。

同一文件中类型相同的每一个资源必须有一个唯一的 ID 值相对应，比如某个应用程序的资源文件，有几个 DLOG 类型的资源，如图1-8 所示，它们各自都有一个唯一的ID值，一个为 128，一个为 129，同时这个文件还有一个 WIND 类型的资源，它的 ID 值为 128，所以对于每一个资源来说它的类型与 ID 值构成了它的唯一标识信息。

你在创建资源的时候，应保证资源类型与 ID 值的组合信息的唯一性，如果你打算给资源取名，同样也要考虑到它的唯一性和易读性。

本书中，除 DODE 类型的资源将由 THINK C 创建，其它资源均由 ResEdit 来完成。

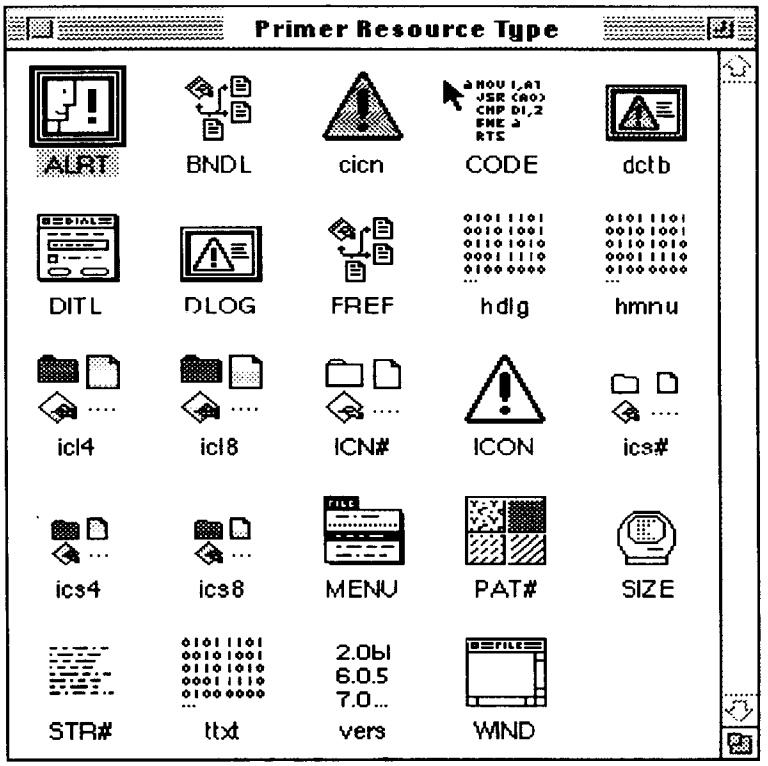

图1-7　本书将要用到的资源

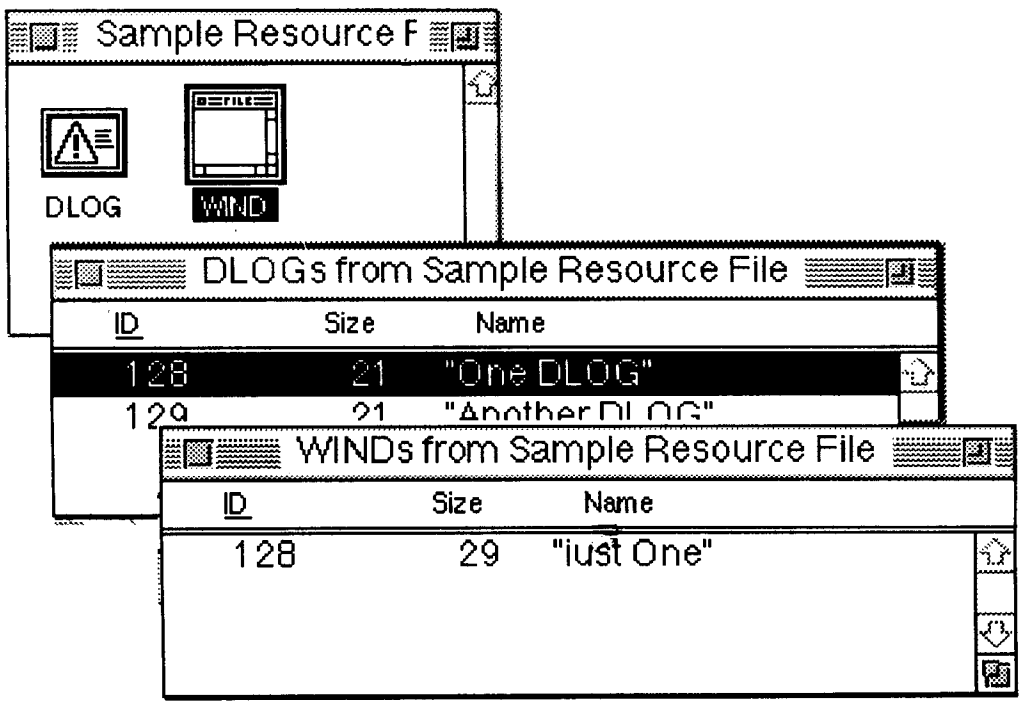

图1-8　资源唯一标识信息举例图示

■ 数据分支和资源分支

Macintosh 创建了一个新概念：把程序与数据分开，这一思想已被许多软件开发商所接受。文件被分为两种类型：文档（document）与应用程序（application）（图1-9）。一般来说，文档是用户可以建立、编辑的文件，可以用两次点按其图标的方式打开。应用程序包括可执行代码及其专用资源（resource）和数据（data）。Macintosh 文件并不像其它操作系统的文件，它包括两个部分：数据分支和资源分支。资源分支中包含文件的资源。如果文件是一个应用程序，其资源分支一般包括用于描述应用程序窗口、菜单、对话框、图标的资源，以及可执行代码本身。应用程序的'SIZE'资源是一个很重要的资源，其中含有应用程序所占空间及其运行时的内存要求。如果文件是一个文档，其资源分支一般包括预置信息、窗口位置以及文档的字体、图标等等。数据分支中包含文件的数据。比如字处理程序把一个文

档的文本信息存储在文档的数据分支里，而把文档的格式信息存于资源分支中。Hypercard

堆栈的大部分信息，就存于数据分支中。本书的 THINK C 项目只用到资源分支。

将数据存入数据分支还是资源分支，在很大程度上取决于是否能将数据有效地结构化以作为资源。例如，资源数据只需指定资源类型和资源ID值，而无需知道它有多少字节。但象一篇文章，其长度是不定长的，难以放入资源管理器所要求的结构中，因此，存入数据分支中比较合适。

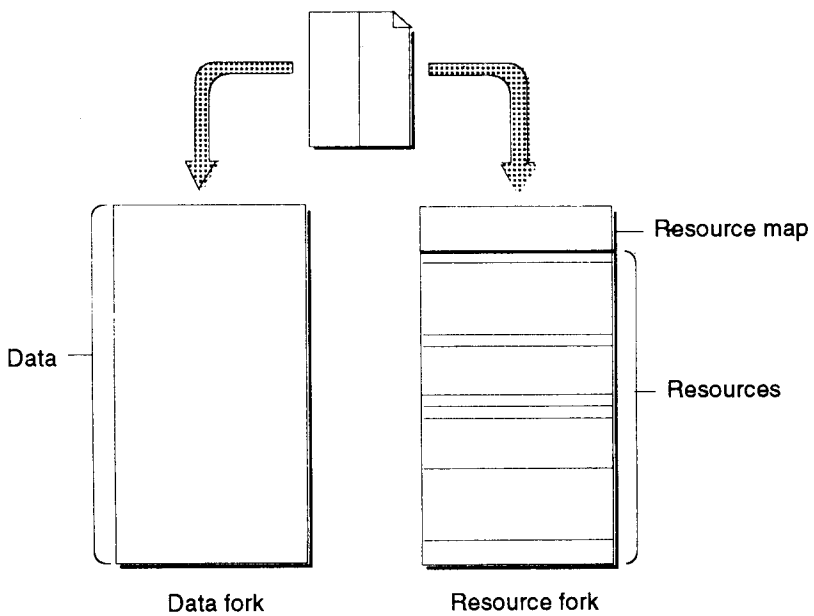

图1-9 Macintosh文件的两个分支

■ 文件系统的层次结构

Macintosh操作系统组织文件的方式称作层次文件系统（hierarch file system，简称HFS）。在HFS中，文件被分组存于不同的目录中，目录称为文件夹（Folder），而若干个目录又可组成上层目录，如图1-10所示。Finder负责管理文件及文件夹。

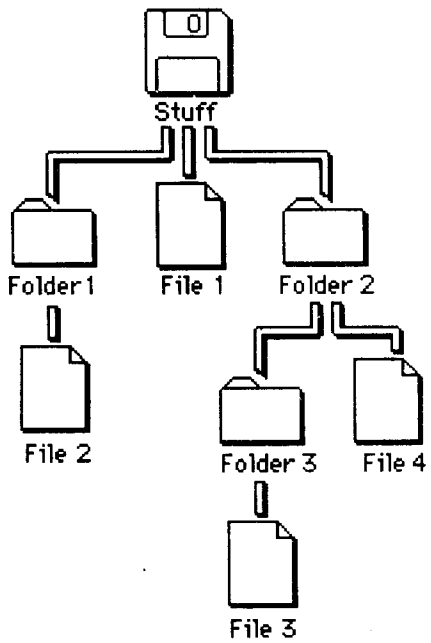

图1-10 层次文件系统

第二章　存储器

本章将针对Macintosh计算机上的存储管理做一个简要的介绍。具体描述你的应用程序开始运行时存储分区（Memory Partition）的组织方式，并解释为Macintosh工具箱和操作系统所使用的基本数据类型。本章还将介绍你怎样才能为特定的目标分配这个存储分区的各部分，以及存储管理器（Memory Manager）怎样协助你维护好一个有秩序的分区。

2.1 存储器概述

在Macintosh操作系统提供的相互协作的多任务环境下，你的应用程序仅能使用全部计算机内可用RAM（内存）中的一部分。有一些可用RAM是为操作系统本身的使用保留的，而剩余的可用内存则被所有打开的应用程序共享。

当操作系统启动时，它把可用RAM分成了两大块。其中有一区域是为操作系统自己保留的，称为系统分区（system partition）系统分区通常从存储器的最低可寻址字节（存储地址为0）开始，而后向上扩展。系统分区包括两个主要部分：

· 一个系统堆（System Heap）

· 一个全局变量（Global Variales）集合

从总体上看，系统分区内的存储器仅供操作系统使用。你的应用程序一般不必读写这部分存储区。

系统分区以外的所有内存都可分配给应用程序或其它软件组成部分使用。在协同多任务环境里，用户可以同时打开多个应用程序，当一个应用程序启动时，操作系统将分配给它一个存储区，称为它的应用分区（application partition）。总之，一个应用程序仅能使用它自己的应用分区内的内存。

图2-1显示了当有若干应用程序同时打开时，内存的组织方式。系统分区占据了内存的最低分区。应用分区占据了一些或所有的剩余空间。注意应用分区首先被分配在高端内存。一个应用分区包括三个主要部分：

· 一个应用程序堆（application heap）

· 一个堆栈

· 一个A5区间（A5 World），它包含了应用程序的全局变量。