第一部分

学生用书

第零章 简 介

0.1 简 介

- 名字
- 工作单位
- 职务
- 从事的工作
- 有关 Windows 的经验
- 期望和建议

0.2 课程纲要

- 第一章：移植
- 第二章：设计目标、对象和注册
- 第三章：结构异常处理
- 第四章：创建和启动进程
- 第五章：线程和同步
- 第六章：标准 I/O
- 第七章：内存管理
- 第八章：动态链接库(DLL)
- 第九章：基于 Win32 应用程序的性能测试工具
- 第十章：远程过程调用(RPCs)
- 第十一章：国际化和 Unicode
- 第十二章：Win32 API 战略

0.2.1 第一天

Microsoft Win32 程序设计课程介绍
- 期望与要求
- 课程纲要

第 课程材料
- 课程设置
- 环境标志

第一章 移植
- 编译
- 头文件
- 移植工具
- PORT.INI
- 移植步骤
- 主要的移植问题
- Microsoft Visual C++ for Windows NT Tools

实验一
- 把 16 位应用程序移植到 32 位平台上

第二章 设计目标、对象和注册

第 Windows NT 操作系统的设计目标
- 对象
- 注册

实验二
- 在注册时映射私有简要表函数
- 如何使用注册来注册一个应用程序

第三章 结构异常处理
- 定义
- 语法
- 异常处理搜索顺序

骤 处理器特性
- 异常流程图
- 异常处理和展开
- 异常过滤
- 异常信息
- 不能处理的异常过滤
- 终止流程图
- 复习

实验三

- 实现结构异常处理
- 使用缺省的异常过滤
- 实现一个过滤程序

0.2.2 第二天

第四章 创建和启动进程

- 体系结构概述
- 对象处理
- 进程创建
- 设置启动窗口信息
- 返回进程信息
- 结束进程
- 多种 API 函数
- 进程间通信对象模型
- 继承
- 管道

实验四

- 创建和启动进程

第五章 线程和同步

- 体系结构概述
- 为什么使用多线程
- 线程额外开销
- 线程创建函数
- 线程 ID 和句柄
- 线程终止
- C 运行时库
- 同步
- 线程和消息队列

实验五

- 创建并运行线程
- 创建和同步多线程
- 解决练习二中的竞态条件

第六章 标准 I/O

- 体系结构概述
- 文件 I/O API
- I/O 处理概述
- Win32 中重叠 I/O 方法

实验六
■ 完成异步文件 I/O

0.2.3 第三天

第七章 内存管理
■ 体系结构概述
■ API 函数
■ 内存共享

实验七
■ 使用堆 API 管理内存
■ 通过内存映射文件共享内存

第八章 动态链接库(DLL)
■ 与基于 Windows 3.1 的 DLL 的区别
■ DLL 入口和出口点
■ 线程局部存储
■ 建立 32 位 DLL

实验八
■ 创建 32 位的 DLL

第九章 基于 Win32 应用程序的性能测试工具
■ 进程观察(PView)
■ 性能监视
■ 性能统计(PStat)
■ Profilers

演示
■ 使用 Win32 性能测试工具

0.2.4 第四天

第十章 远程过程调用(RPCs)
■ 体系结构概述
■ 运行时体系结构
■ MIDL 编译程序
■ 建立 RPC 应用程序

实验九
■ 实现 RPC 应用程序
■ 使用 RPC 关联句柄(选做)

第十一章 国际化和 Unicode
■ 什么是 Unicode

■ 在 Win32 中编程 Unicode
■ 数据类型
■ 基本转换步骤
■ C 运行时扩充
■ 字节顺序标记

实验十
■ 把应用程序转换为 Unicode

第十二章 Win32 API 战略
■ 当前产品状况
■ 当前 API 战略
■ 未来产品状况
■ 未来 API 战略
■ 开发资源提交

实验十一
■ 使用 Win32s(课外作业)

参考资料
这本课程参考的主要资料有:
■ Microsoft Win32 Software Development Kit 联机帮助
■ Microsoft Win32 Programmer's Referencce, Volumes 1-5, Microsoft Press
■ Inside Windows NT, Helen Custer, Microsoft Press

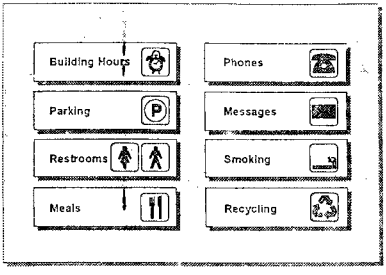
0.3 课程材料

■ 姓名卡片
■ 学生用书
■ 实验手册
■ 评估表格
■ 参考资料

0.4 课程设置

- Microsoft MS-DOS Operating System
- Microsoft Windows
- Microsoft Visual Basic
- Microsoft C and C++ Programming
- Microsoft Excel Programming
- Microsoft LAN Manager
- Microsoft Mail
- Microsoft SQL Server
- Microsoft Education for Client-Server Computing

0.5 环境标志



第一章 移 植

1.1 本章内容

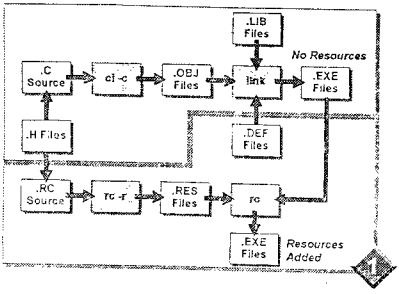
- 编译
- 头文件
- 移植工具
- PORT.INI
- 移植步骤
- 主要的移植问题
- Microsoft Visual C++ for Windows NT Tools

在第一章结束，你将能够：

- 了解应用程序从 16 位平台移植到 32 位平台的一些关键性的问题
- 使用移植工具，把基于 Windows 的代码从 16 位环境移植到 32 位环境
- 只做很少的必要的改动，就能够把基于 Windows 的代码从 16 位环境移植到 32 位环境
- 编译，连接基于 Windows 的 32 位的可执行文件

1.2 浏览 Windows 3.1 的编译过程

幻灯目标：
复习 16 位的 Windows 编译处理过程，并将它与 Win32 编译过程相比较。



关键事项：

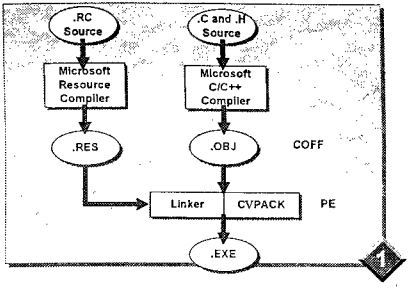
两个独立的连接步骤：

1. 把源文件与库和 .DEF 文件连接起来。
2. 启动 RC 编译器，把资源合并到可执行文件。

编译一个 Windows 3.1 下的应用程序，首先需要使用 Microsoft C Compiler(CL)，把 C 源文件和头文件编译成目标文件，然后将其与库连接。这一系列的操作产生一个没有资源的 .EXE 文件。对于基于 Windows 3.1 的应用程序的资源，可以使用 Microsoft Windows Resource Compiler (RC) 来进行编译，然后将其与已存在的 .EXE 文件相连接，最后形成一个完整的基于 Windows 3.1 的应用程序。

1.3 浏览 Win32 的编译过程

幻灯目标：
连接器只运行一次。CVTRES 把一个资源编译二进制目标文件转换成一个 COFF 格式的目标。



关键事项：

资源目标文件与库和 .DEF 文件一样，一起放在连接命令行上。建立一个应用程序需要较少的过程和时间。

基于 Win32 的应用程序一般只需要连接一次。C 源文件被编译成一个目标文件，与被编译过的资源文件相连接，最后与相应的 Win32 库相连接。

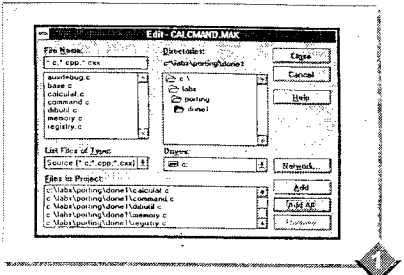
COFF ——公共目标文件格式
PE ——可移植的执行文件

关键事项：

为了与 .OBJ 文件保持一致，.RES 文件已经被改名为 .RBJ 文件。由资源编译器(RC.EXE)产生的 .RES 文件与 Win32 使用的资源格式不一样。可以使用 CVTRES.EXE 把 .RES 文件转变为 Win32 的格式。RC 与在 16 位环境中所起的作用一样，CVTRES 处理由 Win32 带来的一些问题——尤其是，被 LINK32 所需要的 COFF 目标。

1.4 可视工作平台 Project Edit 对话框

幻灯目标：
介绍如何在 Visual Workbench 中使用项目编辑对话框，该对话框主要用于在项目中增加或删除文件。



※ 在项目中增加文件

1. 从 Project 菜单，选择 Edit。此时，Edit-项目名对话框出现。
2. 使用 List Files Of Type 下拉式列表框选择出现在 File Name 列表框中的文件类型。
3. 从 File Name 列表框所列的文件中选取要加进项目中的文件，然后再选择 Add 命令，或者双击在文件列表框的文件。
4. 对每一个想加入项目的文件重复步骤 2 和 3。
5. 当完成编辑项目文件之后，选择 Close 命令。这将引起相关文件被重新建立。

※ 从项目中移去文件

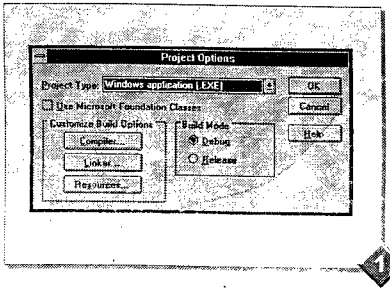
1. 从 Project 菜单，选择 Edit。此时，Edit-项目名对话框出现。
2. 从 Files In Project 列表框中选择要删除的文件，然后选取 Remove 按钮。也可以双击在 Files In Project 列表框中的文件，对其进行删除。
3. 对于项目中每一个要移去的文件，重复步骤 2 和 3。
4. 当完成编辑项目文件之后，选择 Close 按钮。这将引起相关文件被重新建立。

注意：所有的项目编辑都应该通过 Project Edit 菜单。尽量不要手工修改 MSVCNT 项目文件，否则它们可能无法正常运行。

Microsoft Visual C/C++ 32 位版本提供了一个例子代码，它可以把 Microsoft Visual C/C++ 16 位版本的项目文件转变为 32 位版本的文件。这个例子应用程序，包括所有的源代码，可以在 \MSCVNT\SAMPLES\CVTMAKE 中找到。

1.5 可视工作平台 Project Options 对话框

幻灯目标：
在 Project Options 对话框中，可以自定义编译器，连接器和资源编译器的选项，还可以选择 Debug 或 Release 构造模式。



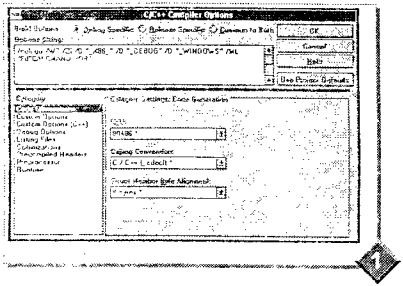
※ 编辑项目的编译器、连接器或资源编译器选项

1. 从 Options 菜单上选择 Project。Project Options 对话框出现。
2. 在 Customize Build Options 中选择按钮
   - 选择 Compiler 按钮将会改变项目的编译器选项
   - 选择 Linker 按钮将会改变项目的连接器选项
   - 选择 Resource 按钮将会改变项目的资源编译器选项

※ 设置 Debug 或 Release 构造选项

1. 从 Options 菜单上选择 Project。Project Options 对话框出现。
2. 在 Build Mode 中选择 Debug 或 Release。
3. 选择 OK 按钮。
4. 构造或重新构造项目。

1.6 可视工作平台 Project C/C++ Compiler Options 对话框



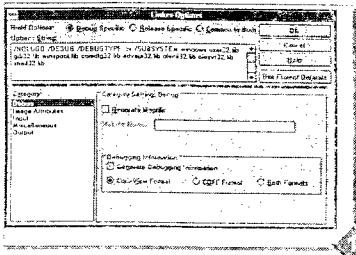
幻灯目标：
在这个对话框中显示了 MSVCNT 中定义的编译选项，对话框的下半部分显示了选项的种类，它们的变化直接反映在上半部分的 Options String 框中。

※ **编辑项目的编译选项**

1. 从 Options 菜单，选择 Project。Project Options 对话框出现。
2. 选择 Complier 按钮。C/C++ Compiler Options 对话框出现。
3. 从三个 Build 选项 Debug Specific、Release Specific 或者 Common To Both 中选择一个。这样可以将编译选项设置为只跟踪、只释放或者既是跟踪又是释放等构造环境。
4. 从 Category 列表框，选择想要编辑的种类。Category Settings 动态反应在 Category 列表框中选择的结果。
5. 对于需要编辑的每一种类重复步骤 4。
6. 选择 OK 按钮。

1.7 可视工作平台 Linker Options 对话框

幻灯目标：
这个对话框中显示了 MSCVNT 中声明的不同的连接选项。选项的种类在对话框的下半部分显示，它的任何变化都反应在对话框上半部分的 Options String 框中。

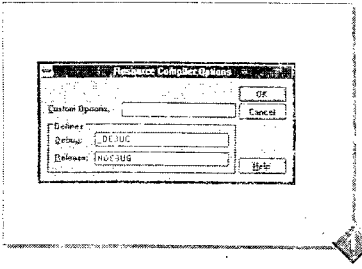


※ 编辑项目的连接选项

1. 从 Options 菜单上选择 Project。Project Options 对话框出现。
2. 选择 Linker 按钮。Linker Opions 对话框出现。
3. 从三个 Build Options：Debug Specific，Release Specific，或者 Common To Both 中选择一个。这样可以将编译选项设置为只跟踪，只释放，或者两者都是等构造环境。
4. 从 Category 列表框选择要编辑的种类。Category Settings 动态反应在 Category 列表框中选择的结果。
5. 对于需要编辑的每一个种类，重复步骤 4。
6. 选择 OK 按钮。

1.8 可视工作平台 Resource Compiler Options

幻灯目标：Resource Compiler 可以为资源编译器声明一些自定义选项，还可以为预处理器定义一些符号。



※ **编辑项目的资源编译选项**

1. 从 Options 菜单上选择 Project。Project Options 对话框出现。
2. 选择 Resource 按钮。Resource Compiler Options 对话框出现。
3. 在 Custom Options 框中为资源编译器加入命令行开关。
4. 可以在 Debug 框中为 Debug 项定义一些常数。
5. 可以在 Release 框中为 Release 项定义一些常数。

1.9 头文件

幻灯目标：
WINDOWS. H 已经从单个较大的头文件变成一组较小的、较好定义的头文件。

Win32 WINDOWS. H 包括

• WINERROR. H	错误 ID
• WINDEF. H	基本类型定义
• WININT. H	Windows NT 定义的类型和常数
• WINUSER. H	Windows Manager 原型和定义
• WINBASE. H	基本函数原型和定义

关键事项：
WINDOWS. H 中有几个头文件。

Win32 WINDOWS. H

Win32 WINDOWS. H 是基于 Win32 应用程序的主要包含文件。在形成的几个较小的头文件中，它的改动比较大。WINDOWS. H 包括所有其他的 WINx. . . x. H 头文件。

WINERROR. H

WINERROR. H 包含了 Win32 API 函数定义的错误代码 。

WINDEF. H

WINDEF. H 对于 Windows 基本的类型有一个类型定义。

注意：
WINBASE. H 里面的函数包括文件 I/O，内容管理等等。
WINDEF. H 有 typedefs 和 #defines。
WINDOWX. H 是可选的，但如果你使用它，它提供了几个可移植的宏。

WINNT. H

WINNT. H 定义了 32 位 Windows 的类型和常数，这些类型和常数由 Windows NT 定义，在 Win32 应用程序接口中使用。

WINUSER. H

WINUSER. H 包括了 Windows NT 下用户组成部分的过程声明，常数定义和宏等。

WINBASE. H

WINBASE. H 定义了 32 位 Windows Base API 函数。

WINDOWSX. H

WINDOWSX. H 提供了一些宏函数，窗口消息和控制函数(帮助和可移植的宏)。这是一个可选的头文件，提供了一些可移植的宏。

1.10 移植工具

幻灯目标：
该工具可以查询源文件，标识出特定的串，意味着该串可能需要进行移植处理。

- 标识出许多代码移植问题
- 包含在 Visual C++ for Windows NT 中(PORTTOOL. EXE)
- 运行于 32 位环境
- 可以后台运行
- 提供上下文敏感的词汇查询
- 通过 PORT. INI 进行自定义
- 包含在一个 DLL 中，可以很容易地引入到用户的编辑器中

关■事项：

该移植工具运行在 32 位环境下，而不是在 16 位环境中。它还可以运行在后台，或者由用户进行交互控制。它不是一个特别灵巧的工具，它提供了词汇查询，但不能从注解中识别相应的代码。