

## 上机实训

### 9.1 实验 1 Windows Server 2003 操作系统的使用

#### 一、实验目的：

1. 练习Windows Server 2003操作系统的常用操作。
2. 练习文件夹、文件的属性设置和共享设置

#### 二、实验准备知识

Windows Server 2003是Windows NT的最新版本。在最初开发它时，曾把它叫作Windows NT Server5.0。但后来在正式发布它之前，又把它重新命名为Windows Server 2003。这一版本的操作系统对原操作系统的每一个方面几乎都进行了修改，并对原有功能也重新进行了修正，使之更加容易使用和掌握。并且，还另外添加了数百个新功能。

尽管如此，Windows Server 2003和Windows NT Server4.0、Windows 98的操作界面非常类似，用户可以很快熟悉它的基本操作。下面对于Windows Server 2003的新功能做简要说明：

##### 1) 网络功能

Windows Server 2003 最显著的革新是对网络功能的增强和集成，它具体表现在Internet Explorer5.0、通信簿、Microsoft Chat、Microsoft NetMeeting、Outlook Express以及媒体播放器等网络组件上。通信簿能够帮助用户记录复杂、多样的网络地址；Internet Explorer 5.0是WWW浏览器的最新版本之一；Microsoft Chat 为网络聊天与交流提供了方便的场所；Microsoft NetMeeting用来进行远程呼叫、召开网络会议；Outlook Express进行邮件的收发和新闻组的讨论。

##### 2) 查找界面

在“开始”菜单中，查找界面的更新是比较突出的。依次选择“开始”|“搜索”|“文件或文件夹”之后，打开“搜索结果”对话框，如图1-1所示，该对话框采用Windows NT资源管理器的界面，在左侧的窗格内填写、设置搜索选项，右侧的窗格显示搜索的结果，它与Windows 95/98、Windows NT 4.0的“查找”对话框完全不同。

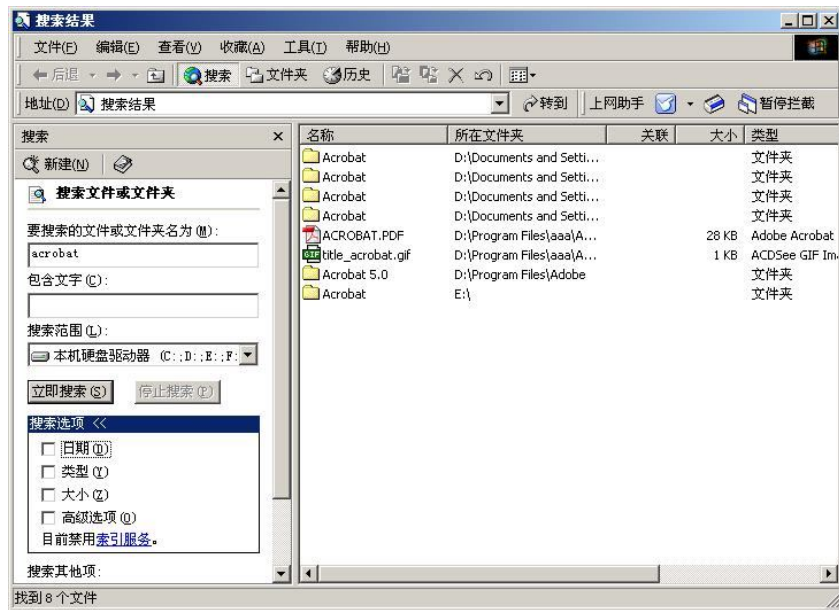


图9-1 “搜索结果”对话框

### 3) 菜单内容

在桌面上打开“开始”菜单，它的内容同Windows NT4.0相比已经发生了很大的变化。“管理工具”出现在“程序”菜单的最前列，在默认的情况下，“管理工具”是由“认证管理器”、“计算机控制台”、“事件查看器”、“索引服务器控制台”、“可移动存储控制台”等五部分内容组成的，并且绝大部分系统管理工具都被集中在“计算机控制台”内。

为了方便用户的管理，并使整个过程更加合理，Windows Server 2003采用了“管理控制台”技术，它继承了Windows资源管理器的一贯风格，由树状节点结构组成，将它所包含的子工具逐一展开，可以看到该控制台所具有的强大功能，本地计算机或域内所有计算机的资源都在它的管理范围之内。

在“开始”菜单中的“附件”中，也会看到较大的变化。例如，依次选择“开始”、“程序”、“附件”、“系统工具”之后，将打开如图所示的菜单选项。在此仅有“备份”、“磁盘清理”、“磁盘碎片整理程序”、“任务计划”与“字符映射表”等五项内容，而其它的“磁盘空间管理”、“维护向导”、“压缩代理”、“计划任务”等都被分配到“计算机管理控制台”内。



图9-2 “系统工具”菜单

在Windows Server 2003桌面上选择“开始”菜单，并逐级打开其中的选项，当选中最低一层的菜单命令时，菜单命令将高亮显示，将鼠标停留片刻，将出现一条提示信息，

内容为该菜单命令的功能说明。例如，选择“开始”|“程序”|“管理工具”|“计算机管理控制台”时，提示信息将告诉用户使用计算机管理控制台来管理本地或者远程计算机。

#### 4) 控制面板



图9-3 “控制面板”窗口

“控制面板”是Windows Server 2003的功能控制中心，选择“开始”|“设置”|“控制面板”之后，将打开如图1-3所示的“控制面板”窗口。在地址栏内选择其它选项，可以导航到其它的资源。在控制面板中，新添加、改进的工具具有管理工具、硬件向导、网络和拨号连接、扫描仪和照相机、SQL服务器客户端配置等。

仔细观察可以发现，用于配置网络属性的“网络”图标并没有出现在控制面板内，这是因为桌面上的“网络邻居”完全可以胜任网络的配置、名称的标识等操作。在Windows Server 2003的控制面板左侧有两个站点连接：“Windows Update”和“技术支持”，它们分别与<http://windows.update.microsoft.com/default.com>、<http://support.microsoft.com/support>两个主页保持链接，通过这两个页面可获得Windows Server 2003最新版本和技术支持。

#### 5) 活动目录

除了查看磁盘的文件夹和文件的传统目录树外，管理员还可以浏览活动目录，对域用户、用户组和网络资源进行管理。活动目录能够把域划分为不同的组织单元，这意味着网络在目录树中直接具有部门的名称。域的树显示多个对象进行划分后的结果，每一部分都有自己的安全性、委托关系和用户许可证等特性。

另外，系统管理员还可以通过活动目录指定局部管理员，每人以自己的安全性及许可权限进行管理分类。局部管理员有能力建立自己的组织单元并把对用户的分配权力直接拖放到目录的域中，这时所指定的帐号信息及许可权将得到自动复制。

#### 6) 连接网络

在桌面上双击“网络邻居”图标之后，即可浏览和使用网上的计算机资源，而无需通过驱动器映射来连接某个网络。要共享某个文件夹或打印机时，可右击文件夹或打印机，执行“共享”命令，配置共享属性。通过共享口令、用户列表的设置，可将访问用户进行有效的控制。需要配置网络时，可双击控制面板的“网络和拨号连接”图标，打开如图所示的“网络和拨号连接”窗口。在默认的情况下，该窗口包括“新建连接”和“本地连

接”两个选项，前者用于创建新的拨号连接，后者代表本机所在局域网的配置情况。

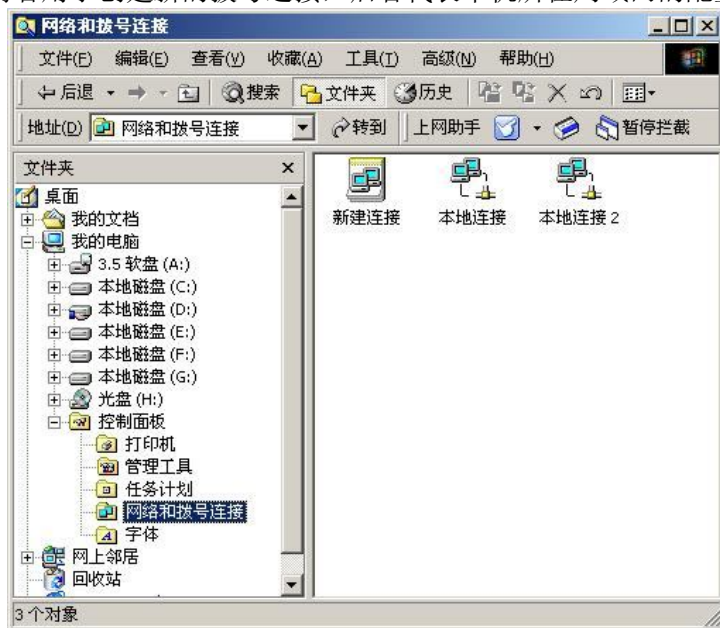


图9-4 “网络和拨号连接”窗口

在网络上只要有一台调制解调器，就可以使用拨号网络，通过向服务器和带调制解调器的计算机呼叫。可以访问该计算机以及与之相连的网络，并使用其中的共享资源。使用“拨号网络”时，可双击控制面板的“电话和调制解调器”图标，打开如图所示的“位置信息”对话框，确定本地计算机所在的地区、使用的电话号码。

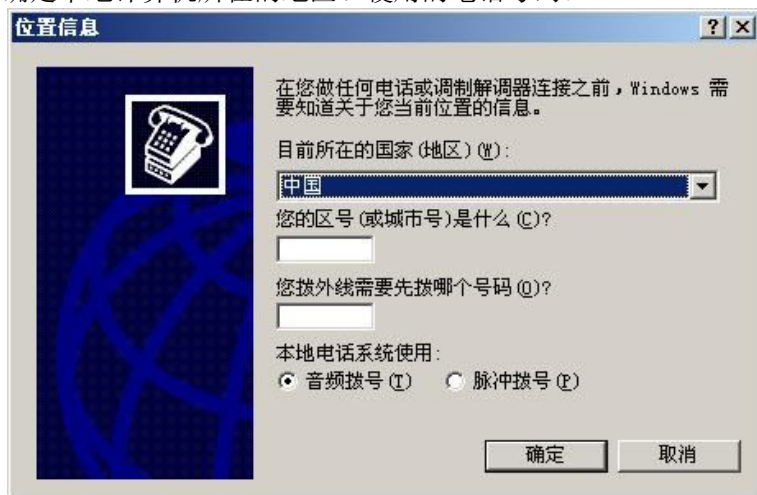


图9-5 “位置信息”对话框

### 9.1.3 实验步骤

Windows Server 2003的常用操作和Windows 98基本一样。这里仅对资源管理器中新增加的功能以及对文件夹、文件的属性设置、共享设置方面做简单介绍。

#### 1) 资源管理器新增功能

Windows Server 2003的资源管理器在标准工具栏上增加了三个按钮，即“搜索”、

“文件夹”、“历史”。这些按钮可以帮助用户更方便快捷地找到所要的文件。

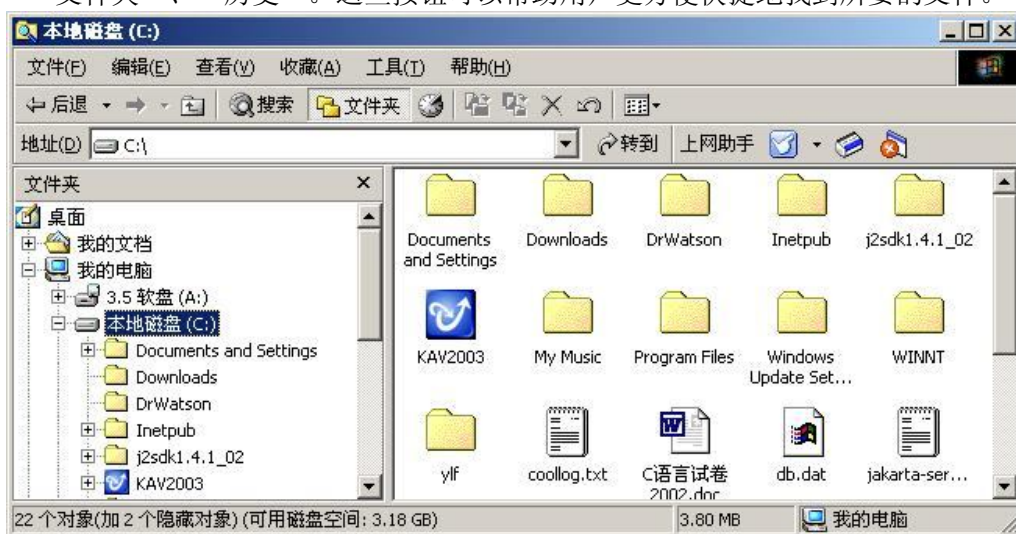


图9-6资源管理器窗口

“历史纪录”栏显示出用户以前浏览过的文件或web网页，排列方式也有多种方案，可以按时间的先后、站点的名称，也可以按访问次数的多少排序。图1-7就是资源管理器的“历史”栏。



图9-7资源管理器的“历史”栏。

资源管理器的“搜索”功能很强，可以根据用户的需要进行三种类型的搜索：搜索文件或文件夹、搜索计算机、搜索Internet。

## 2) 设置文件夹选项

Windows Server 2003提供了统一的“文件夹选项”对话框来管理设置资源管理器和文件夹的显示风格。在资源管理器窗口打开“工具”|“文件夹选项”时，就会出现如下图所示“文件夹选项”对话框。其中包括“常规”、“查看”、“文件类型”、“脱机显



示”等四个标签，分别对应不同的设置。

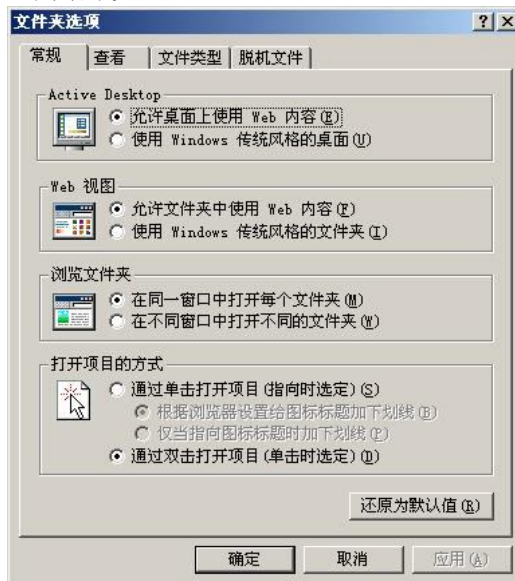


图9-8 “文件夹选项”对话框

其中“常规”、“查看”与Windows 9X功能基本一致。“文件类型”标签功能是查看、更改、新建文件的打开方式。在“文件夹选项”对话框中单击“文件类型”标签，将打开图1-7所示选项卡，已经注册的文件类型将出现在选项卡内，而且与文件的扩展名对应。



图9-9 “文件类型”选项卡

在Windows Server 2003中，如果要更改一个文件的类型，采用的步骤如下：

- a) 从资源管理器或控制面板中运行“文件夹选项对话框”。
- b) 选中“文件类型”标签。
- c) 单击所要更改的文件类型，
- d) 单击“高级”按钮。
- e) 如有必要，更改文件类型的描述，并单击“更改”图标。
- f) 在“动作”中，单击所要更改的命令，然后单击“编辑”、“删除”、“设置默认”或者单击“新建”来为“动作”中的列表添加一条新的命令。
- g) 重复步骤f)为该文件类型更改所需要的全部动作。

在Windows Server 2003中，为了将文件类型与应用程序关联起来，可采用如下步骤：

- a) 从资源管理器或控制面板中运行“文件夹选项”对话框。
- b) 选中“文件类型”标签。
- c) 单击“新建”按钮。
- d) 键入一个新的或已有的文件扩展名，然后单击“高级”。
- e) 在“关联的文件类型”中，单击“新建”以创建一种与该文件扩展名相关联的新的文件类型。

在Windows Server 2003中，可以更改特定文件类型与特定程序的关联，其步骤是：

- a) 从资源管理器或控制面板中运行“文件夹选项”对话框。
- b) 选中“文件类型”标签。
- c) 单击“更改”按钮。
- d) 在“打开方式”对话框中，选择想要用于打开该文件的应用程序，或者单击“其他”以选择一个不在列表中的程序。

### 3) 压缩文件和文件夹

有时为了节约磁盘空间，需要对一些文件进行压缩。在Windows Server 2003中NTFS驱动器上，可直接对文件或文件夹进行压缩，步骤如下：

- a) 运行“资源管理器”。
- b) 选择想要进行压缩的文件或文件夹。
- c) 单击右键，从快捷菜单中选择“属性”。
- d) 单击“高级”按钮。
- e) 选中“压缩内容以节省磁盘”复选框。
- f) 如果压缩的是一个文件夹，则在提示确认属性变化时单击所要的选项。

文件夹被压缩后，添加或拷贝到该文件夹中的文件都将被自动压缩，从不同NTFS驱动器上移动来的文件也会被自动压缩。但从同一个NTFS驱动器中移动过来的文件不会被压缩。

### 4) 共享文件或文件夹

在资源管理器中设置文件或文件夹的共享，操作与Windows98下基本相同。

## 9.2 实验 2 Windows Server 2003 操作系统安装

利用光驱直接从 Windows Server 2003 光盘启动安装,这是在没有操作系统或者操作系统损坏的情况下常用的安装方式。当然,如果计算机里有一个完好的操作系统也可以来用这样的安装方式,这也是笔者向大家推荐的安装方式。下面是这种安装方式的详细过程。

### 一、实验目的:

#### 1. Windows Server 2003 操作系统安装。

### 二、创建磁盘分区

这一步是选择将 Windows Server 2003 操作系统安装在哪个磁盘分区上,如果磁盘上没有任何分区,则必须新建一个。安装中的选择:

ENTER=安装。用箭头键选择要安装 Windows Server 2003 的已经建立好的分区,然后按下 Enter 键。

C=创建磁盘分区。用箭头键选择未划分的空间,然后按下 C 键,将创建新的磁盘分区。

D=删除磁盘分区。用箭头键选择所要删除的现存分区,然后按下 D 键。

F3=退出。退出安装程序。

在安装过程中所做出的选择,取决于硬盘的状态:

(1) 如果硬盘未分区,则必须为 Windows Server 2003 创建分区并为 Windows Server 2003 设定适当的空间大小。

(2) 如果硬盘已有分区,并有足够大的未分区空间,可以考虑为 Windows Server 2003 在未分区的空间上创建新的分区。

(3) 如果硬盘中已经分区空间足够大,那么可以利用已经分区空间来安装 Windows Server 2003。

(4) 如果硬盘中分区空间都不能满足 Windows Server 2003 的需要,可以通过删除更多的分区来创建新的来分区磁盘空间。然后用它来安装 Windows Server 2003。

由于安装的是 Windows Server 2003 的企业版,所以安装系统的分区大小不应该小于 1.5GB,如果你的硬盘已经有了一个分区,则你可以通过将它删除来创建更多的未分区的磁盘空间,然后用它来创建 Windows Server 2003 的分区。

提示:也可以利用安装程序创建其他分区,但向大家推荐只创建要安装 Windows Server 2003 的分区并适当地设置大小。其余的分区最好是在安装完 Windows Server 2003 之后,用 Windows Server 2003 自带的图形界面的磁盘管理工具来划分。

### 三、格式化磁盘分区并选择文件系统

选择新建立的磁盘分区,然后按下 Enter 键。由于此系统刚刚创建还没有格式化,这时系统会提示用那种文件系统格式化此分区。一共有四个选择,其中有两个是快速格式



化。快速格式化相对常规格式化的速度要快一些，尤其是分区空间特别大的时候更加明显。对于新硬盘或者怀疑有问题的硬盘建议使用常规格式化，因为常规格式化虽然速度慢，但会检测硬盘是否有坏道，还会将坏道标志排除。

用箭头键选择“用 NTFS 文件系统格式化磁盘分区（快）”，按下 Enter 键会出现磁盘格式化的界面，

提示：如果是在真正的服务器上，为了提高文件的安全性，建议使用 NTFS 文件系统，如果是学习或者做实验，建议使用 FAT32 文件系统。

#### 四、复制文件并重新启动

格式化分区结束以后，安装程序将文件复制到 Windows 的安装文件夹，然后出现红色等待 1.5s 的倒计时状态条，如果不想等待，只要按任意键就可以马上重新启动计算机。

这时候系统已经完成了收集硬盘信息的过程，接下来将进入 Windows Server 2003 的图形界面安装阶段。

#### 五、进入图形界面安装

计算机重新启动以后马上就会进入图形安装界面，此时，系统提示安装程序已经运行到哪个步骤，还需要多少时间才能完成安装，并通过屏幕展示 Windows Server 2003 在各个方面的优点。

注意：如果计算机在重新启动以后，出现蓝屏状态，一般情况下是硬件导致的，解决方法是更换硬件，比如内存、硬盘。

提示：如果是中文版可以使用默认值，直接单击【下一步】按钮。如果是英文版则要选择语言，否则将来如果遇到中文汉字还需要重新配置才能支持。

安装程序总共完成 4 项任务：

安装【开始】菜单

注册组件

保存设置

删除任何用过的临时文件

当这 4 项任务完成以后，整个安装程序结束，整个安装过程因计算机的配置不同大概需要 40-60min 的时间。

重新启动后，按 Ctrl-Alt-Del 组合键，用户名是 Administrator，输入安装时设置的管理员密码，然后按回车键，就可以登录到 Windows Server 2003，然后就可以使用了。

### 9.3 实验 3： Windows Server 2003 磁盘管理

#### 一、实验目的

1. 练习 Windows Server 2003 的磁盘管理；
2. 建立主磁盘分区和逻辑分区、镜像分区

## 二、实验准备知识

磁盘管理是一项使用计算机时的常规任务，Windows Server 2003在磁盘管理方面提供了强大的功能。Windows Server 2003的磁盘管理任务是以一组磁盘管理实用程序的形式提供给用户的，它们位于“计算机管理”控制台中，包括查错程序、磁盘碎片整理程序、磁盘整理程序等。这些工具在Windows NT相应程序的基础上做了很大的改进，使得用户可以使用这些方便、强劲的磁盘管理工具对本地磁盘进行各种操作。

用户使用磁盘管理程序之前，有必要首先了解一些有关磁盘管理的基础知识以及Windows Server 2003采用的磁盘管理新技术。例如：

- 多种不同的文件系统：FAT、FAT32、NTFS等特点。
- 磁盘分区和卷的区别，以及将分区转换为卷时两个不同磁盘组织的对应关系。

采用的磁盘管理新技术有：

- 动态存储。利用动态存储，不用关闭系统或打断用户就可以完成管理任务。例如，不用重新启动系统，就可以创建、扩充或监视卷。不用重新启动也可以添加新磁盘。多数配置改变几乎可以立即生效。

- 本地和网络驱动器管理。如果是管理员，可以管理运行Windows Server 2003或Windows NT4.0 的任何网络计算机。

- 简化任务和直觉的用户接口。磁盘管理易于使用。菜单显示了在选定对象上执行的任务，向导引导你创建分区和卷并初始化或更新磁盘。

- 驱动器路径。可以使用磁盘管理将本地驱动器连接或固定一个本地NTFS 格式卷的空文件夹上。

## 三、实验步骤

### 1. 启动“磁盘管理”

在Windows Server 2003中，有关磁盘管理的各项事务都可通过“磁盘管理”程序来完成。启动“磁盘管理”的步骤是：

- 1) 以Administrator的身份登录系统。
- 2) 从“开始”菜单中点击“程序”，然后点击“管理工具”。
- 3) 在“管理工具”菜单中单击“计算机管理”，然后在出现的“计算机管理”窗口中打开“存储”下的“磁盘管理”，窗口如图4-1所示。“磁盘管理”使用状态栏和图例来显示计算机的磁盘资源，通过定制颜色和图样来显示磁盘区域的使用情况。

### 2. 建立主磁盘分区

一台基本磁盘内最多可以有4个主磁盘分区。创建主磁盘分区的步骤如下：

- 1) 启动“磁盘管理”。
- 2) 选取一块未指派的磁盘空间，如图9-13所示，这里我们选择“磁盘1”。

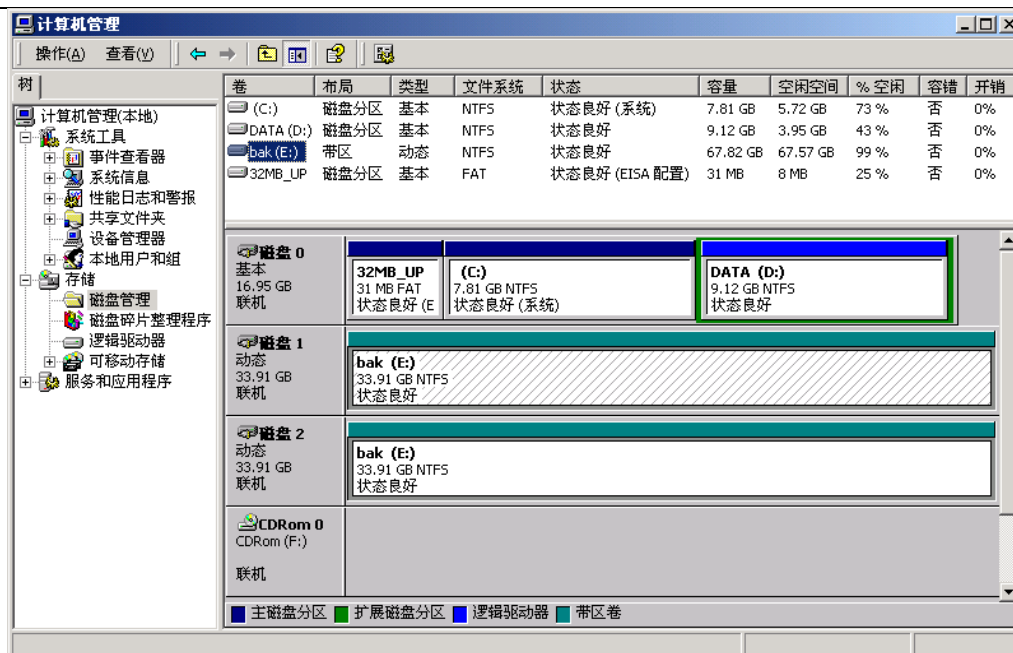


图 9-11 磁盘管理

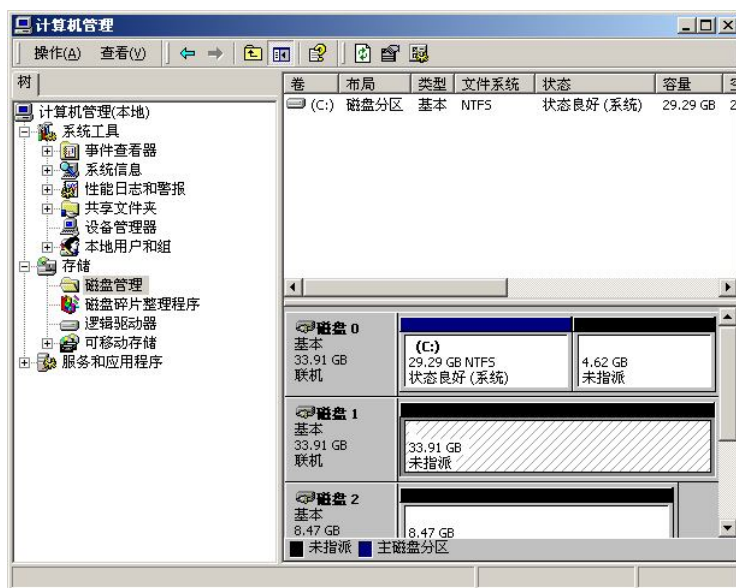


图 9-12 选择未指派的空间

3) 用鼠标右击该空间，在弹出的菜单中选择“创建磁盘分区”，在出现“欢迎使用创建磁盘分区向导”对话框时，单击“下一步”按钮。

4) 在如图 9-13 所示的“选择分区类型”对话框中，选择“主磁盘分区”，单击“下一步”按钮。

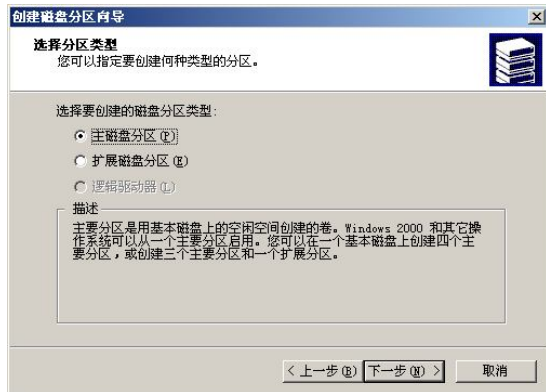


图 9-13 选择分区类型



图 9-14 指定分区大小

5) 在图 9-14 所示的“指定分区大小”对话框中，输入该主磁盘分区的容量，此例子我们输入“600M”。完成后单击“下一步”按钮。

6) 在图 9-15 所示的“指派驱动器号和路径”对话框中，完成其中的单选框选择，单击“下一步”按钮。其中每个单选框的含义如下：

**指派驱动器号：**指定一个磁盘驱动器号来代表该磁盘分区，例如 E:、F:。

**将这个卷装入一个支持驱动器路径的空文件夹中：**例如，利用 E:\backup 来代表该磁盘分区，则以后所有要存储到 E:\backup 的文件，都会被存储到该磁盘分区内，而不是 E: 盘的 backup 文件夹。注意该文件夹必须是空的文件夹，也就是其中不可已有任何文件，并且该文件夹必须是位于 NTFS 卷内。这个功能特别适用于 26 个磁盘驱动器号 (A: 到 Z:) 不够使用时。

**不指派驱动器号或路径：**如果在创建分区时选择此项，则可以在创建完分区以后再指定磁盘驱动器代号或者利用一个空文件夹来代表此磁盘分区。选择相应分区，鼠标右键点击选择“更改驱动器名和路径”，可完成修改工作。

这里，选择指派一个驱动器号 E 来代表该主磁盘分区。

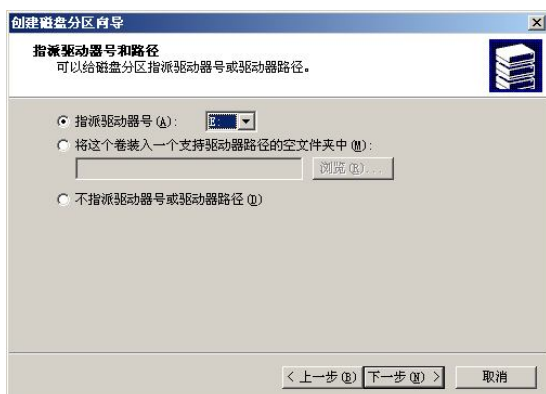


图 9-15 指派驱动器号和路径



图 9-16 格式化分区

7) 在如图 9-16 所示的“格式化分区”对话框中，可以选择是否格式化该分区，若选择格式化该分区，则需要设置如下内容：

**使用的文件系统：**可选择 FAT、FAT32 或 NTFS。

**分配单位大小：**一般建议选用默认值，系统会根据该分区的大小自动设置。

**卷标：**为该磁盘分区设置一个名称。

**执行快速格式化：**选此选项时，系统只是重新创建 FAT、FAT32 或 NTFS 格式，不去检查是否有坏扇区，同时磁盘内原有文件不会真正的被删除。

**启动文件及文件夹压缩：**选此选项，可将该磁盘设为“压缩磁盘”，以后添加到该磁盘分区中的文件及文件夹都会被自动压缩。

8) 上述所有内容设置完成，系统进入安装向导最后“完成”对话框，并列出了用户所设置的所有参数。单击“完成”按钮，系统开始格式化该分区。

由于用户各自使用的计算机中的硬件和软件的配置都有所差别，这使得用户在配置自己的磁盘分区时也应有所不同。用户应根据实际的需要和现有的硬盘配置，合理地规划对磁盘分区进行规划和调整。通常情况下，用户一般在安装 Windows Server 2003 的过程中对磁盘分区进行配置。

在安装过程中用户可以有如下选择：

- 如果硬盘未分区，可以创建 Windows Server 2003 分区并划分其大小。
- 如果现有的分区足够大，可以在该分区上安装 Windows Server 2003。
- 如果现有的分区太小，但还有足够大的未分区空间，可以创建一个新的 Windows Server 2003 分区；
- 如果硬盘有一个已经存在的分区，可以删除它以便得到足够多的未分区空间创建 Windows Server 2003 分区。删除一个已有的分区将同时删除该分区上的所有数据。
- 如果为 Windows Server 2003 设置了双重启动，一定要把 Windows Server 2003 安装到它自己的分区上。若把 Windows Server 2003 安装到另一个操作系统所在的分区上，可能导致安装程序覆盖由另一个操作系统安装的文件。

## 9. 4 实验 4 Windows 中的进程

### 一、背景知识

Windows可以识别的应用程序包括控制台应用程序、GUI应用程序和服务应用程序。控制台应用程序可以创建GUI(Graphics User Interface),GUI应用程序可以作为Windows服务来运行,可以向标准的输出流写入数据。不同类型应用程序间的唯一重要区别是其启动方法不同。

Windows 它提供了创建控制台应用程序的能力,使用户可以利用标准的 C++工具,来创建小型应用程序。当控制台应用程序运行时,Windows 服务会给用户提供服务。

服务应用程序需要ServiceMail()函数,由服务控制管理器(SCM)加以调用。SCM是操作系统的集成部分,启动后开始服务,负责响应服务应用程序请求或接受另一个服务中的请求。通常,服务需要登录到特殊的LocalSystem账号下,此账号具有与开发人员创建的进程不同的权限。

当令 C++编译器创建可执行程序时,编译器将源代码编译成 OBJ 文件,然后将其与标准库相链接,产生的 EXE 文件。EXE 文件是装载器指令、机器代码、应用程序的代码和数据的集合。装载器指令告诉系统从哪里装载机器代码。另一个装载器指令告诉系统从哪里开始执行进程的主线程。在进行某些设置后,进入开发者提供的 main()、ServiceMain()或 WinMain()函数的低级入口点。机器代码中包括控制逻辑,它所做的工作包括跳转到 Windows API 函数、进行计算或向磁盘写入数据等。Windows 允许开发人员将大型应用程序分为较小的、互相有关系的服务模块,即动态链接库(DLL)代码块,在其中包含应用程序所使用的机器代码和应用程序的数据。

### 二、实验目的

通过对Windows编程,进一步熟悉操作系统的基本概念,较好地理解Windows的结构,理解进程。

### 三、工具/准备工作

在开始本实验之前,请回顾教科书的相关内容。

您需要做以下准备:

- (1) 一台运行Windows操作系统的计算机。
- (2) 计算机中需安装Visual C++6.0专业版或企业版。

### 四、实验内容与步骤

#### 1. 简单的控制台应用程序

我们创建一个名为HelloWorld.cpp的应用程序。

步骤1:登录进入Windows操作系统。

步骤2:在"开始"菜单中单击"程序",启动Visual C++6.0。



步骤3:选择菜单项文件——新建——C++ Source File, 并选择文件名为HelloWorld.cpp和存储目录, 然后按确定按钮, 进入编程界面。

步骤4: 将清单4-1中的程序键入。

```
//清单4-1 一个简单的Windows控制台应用程序HelloWorld.cpp
#include <iostream.h>

void main()
{
    cout << "Hello,World!"<< endl;
}
```

步骤5: 选择菜单项编译, 编译程序HelloWorld.cpp。

步骤6: 选择菜单项编译, 运行HelloWorld.cpp程序, 产生一行文字"Hello,World"。

## 2.GUI应用程序

在下面的实验中, C++编译器创建一个GUI应用程序, 代码中包括了WinMain()方法, 是GUI类型的应用程序的标准入口点。

步骤1:在"开始"菜单中单击"程序",启动Visual C++6.0。

步骤2:选择菜单项文件——新建——工程——Win32 Application, 选择工程名为msgbox和存储目录, 然后按确定按钮。

步骤3:选择菜单项文件——新建——C++ Source File, 并选择文件名为msgbox.cpp, 将该文件加入工程msgbox, 然后按确定按钮, 进入编程界面。

步骤4: 将清单4-2中的程序键入。

```
//清单4-2 WindowsGUI应用程序msgbox.cpp
# include<windows.h> //标准的include
// 告诉连接器与包括MessageBox API函数的user32库进行连接
# pragma comment(lib,"user32.lib")
// 这是一个可以弹出信息框然后退出的简单的应用程序
int APIENTRY WinMain(HINSTANCE, /* hInstance */
                     HINSTANCE, /* hPrevInstance */
                     LPSTR, /* lpCmdLine */
                     int ) /* nCmdShow */
{
    ::MessageBox(
        NULL, /*没有父窗口*/
        "Hello,Windows", //消息框中的文本
        "Greetings, ", //消息框标题
        MB_OK) //其中只有一个OK按钮
    //返回0以便通知系统不进入消息循环
    return(0);
}
```

步骤5: 选择菜单项编译, 编译程序msgbox.cpp,并重新构件msgbox.exe。

步骤6: 选择菜单项编译, 运行msgbox.exe程序, 弹出一个对话框。

在清单4-2的GUI应用程序中, 首先需要Windows.h头文件, 以便获得传送给WinMain()和MessageBox()API函数的数据类型定义。接着的pragma指令指示编译器/连接器找到

User32.LIB 库文件并将其与产生的 EXE 文件连接起来。如果没有 `pragma` 指令，则 `MessageBox()` API 函数就成为未定义的了。这一指令是 Visual Studio C++ 编译器特有的。接下来是 `WinMain()` 方法。其中有四个由实际的低级入口点传递来的参数，`hInstance` 参数用来装入与代码相连的图标或位图一类的资源，无论何时，都可用 `GetModuleHandle()` API 函数将这些资源提取出来。系统利用实例句柄来指明代码和初始的数据装在内存的何处。句柄的数值实际上是 EXE 文件映像的基地址，通常为 `0x00400000`。下一个参数 `hPrevinstance` 是为向后兼容而设的，现在系统将其设为 `NULL`。应用程序的命令行(不包括程序的名称)是 `lpCmdLine` 参数。另外，系统利用 `nCmdShow` 参数告诉应用程序如何显示它的主窗口(选项包括最小化、最大化和正常)。最后;程序调用 `MessageBox()` API 函数并退出。如果在进入消息循环之前就结束运行的话，最后必须返回 0。

### 3. 进程对象

操作系统将当前运行的应用程序看作是进程对象。利用系统提供的唯一的称为句柄 (HANDLE) 的号码，就可与进程对象交互，这一号码只对当前进程有效。

本实验表示了一个简单的进程句柄的应用。在系统中运行的任何进程都可调用 `GetCurrentProcess()` API 函数，此函数可返回标识进程本身的句柄。然后就可以利用这一句柄提供该进程的有关情况。

步骤1:在"开始"菜单中单击"程序",启动 Visual C++6.0。

步骤2:选择菜单项文件——新建——C++ Source File，并选择文件名为 `prohandle.cpp`，然后按确定按钮，进入编程界面。

步骤3: 将清单4-3中的程序键入。

```
//清单4-3 获得和使用进程的句柄prohandle.cpp
#include<windows.h>
#include<iostream.h>
// 确定自己的优先权的简单应用程序
void main()
{
    // 从当前进程中提取句柄
    HANDLE hProcessThis = :: GetCurrentProcess();
    // 请求内核提供该进程所属的优先权类
    DWORD dwPriority=:: GetPriorityClass(hProcessThis);
    // 发出消息，为用户描述该类
    cout<< "Current process priority:";
    switch(dwPriority)
    {
        case HIGH_PRIORITY_CLASS:
            cout<<"High" ;
            break;
        case NORMAL_PRIORITY_CLASS:
            cout<<"Normal" ;
            break;
        case IDLE_PRIORITY_CLASS:
            cout<<"Idle" ;
            break;
    }
}
```

```

        case REALTIME_PRIORITY_CLASS:
            cout<<"Realtime";
            break;

        default:
            cout<<"unknown ";
            break;
    }
    cout << endl;
}

```

步骤4: 选择菜单项编译, 编译程序msgbox.cpp。

步骤5: 选择菜单项编译, 运行msgbox.exe程序, 显示当前进程的优先权为Normal。

清单4-3中列出的是一种获得进程句柄的方法。进程句柄可以作为参数传送给应用程序调用的其他API。正如清单3-3中对GetPriorityClass()API函数的调用那样, 通过进程句柄, 系统向内"窥视"进程对象以决定其优先级, 然后将此优先级返回给应用程序。OpenProcess()和CreateProcess()API函数也可以用于提取进程句柄。前者提取的是已经存在的进程的句柄, 而后者创建一个新进程, 并将其句柄提供出来。

下面程序4-4是上面程序4-3的深入, 清单4-4显示如何找出系统中正在运行的所有进程, 如何利用OpenProcess()API函数来获得每一个访问进程的进一步信息。

步骤1: 选择菜单项文件——新建——C++ Source File, 并选择文件名为prolist.cpp, 然后按确定按钮, 进入编程界面。

步骤2: 将清单4-4中的程序键入。

清单4-4 利用句柄查出进程的详细信息

```

#include<windows.h>
#include<tlhelp32.h>
#include<iostream.h>

DWORD GetKernelModePerCentage(const FILETIME &ftKernel,const FILETIME &ftUser)
{
    // 将 FILETIME 结构转化为64位整数.
    ULONGLONG qwKernel =(((ULONGLONG) ftKernel.dwHighDateTime)<< 32) +
        ftKernel.dwLowDateTime;
    ULONGLONG qwUser =
        (((ULONGLONG) ftUser.dwLowDateTime) <<32)+
        ftUser.dwLowDateTime;
    //将消耗时间相加, 然后计算消耗在内核模式下的时间百分比
    ULONGLONG qwTotal=qwKernel+qwUser;
    DWORD dwPct =(DWORD)(((ULONGLONG) 100*qwKernel)/ qwTotal);
    return(dwPct);
}

// 以下是将当前运行进程名和消耗在内核模式下的时间百分数都显示出来的应用程序
void main()
{
    // 对当前系统中运行的进程拍取"快照"
    HANDLE hSnapshot =:: CreateToolhelp32Snapshot(
        TH32CS_SNAPPROCESS,          // 提取当前进程

```

```

    0);           //如果是当前进程，就将其忽略
// 初始化进程入口
PROCESSENTRY32 pe;
:: ZeroMemory(&pe,sizeof(pe));
pe.dwSize=sizeof(pe);
// 按所有进程循环
BOOL bMore = :: Process32First(hSnapshot,&pe);
while(bMore)
{
    // 打开用于读取的进程
    HANDLE hProcess = :: OpenProcess(
        PROCESS_QUERY_INFORMATION,    // 指明要得到信息
        FALSE,                        // 不必继承这一句柄
        pe.th32ProcessID); // 要打开的进程
    if(hProcess!=NULL)
    {
        // 找出进程的时间
        FILETIME ftCreation,ftExit,ftKernelMode,ftUserMode;
        :: GetProcessTimes(
            hProcess,           // 所感兴趣的进程
            &ftCreation,        // 进程的启动时间(绝对的)
            &ftExit,            // 结束时间(如果有的话)
            &ftKernelMode,      // 在内核模式下消耗的时间
            &ftUserMode);       // 在用户模式下消耗的时间
        // 计算内核模式消耗的时间百分比
        DWORD dwPctKernel = :: GetKernelModePerCentage(
            ftKernelMode,       // 在内核模式上消耗的时间
            ftUserMode);        // 在用户模式下消耗的时间
        // 向用户显示进程的某些信息
        cout<<"ProcessID:"<<pe.th32ProcessID
            <<" ,EXE file:"<< pe.szExeFile
            <<" ,in kernel mode: "<< dwPctKernel
            <<endl;
        // 消除句柄
        :: CloseHandle(hSnapshot);
    }
    // 转向下一个进程
    bMore = :: Process32Next(hSnapshot, &pe);
}
}

```

步骤3: 选择菜单项编译，编译程序prolist.cpp。

步骤4: 选择菜单项编译，运行prolist.exe程序。

清单4-4的程序首先利用工具帮助库来获得当前运行的所有进程的快照。然后应用程序进入快照中的每一个进程，得到其以PROCESSENTRY32结构表示的属性，这一结构用来

向OpenProcess()API函数提供进程的ID。Windows跟踪每一进程的有关时间，示例中是通过打开的进程句柄和GetProcessTimes()API来查询得到有关时间的。接下来，一个定制的帮助函数取得了几个返回的数值，然后计算进程在内核模式下消耗的时间占总时间的百分比。程序的其余部分比较简单，只是将有关信息显示给用户，清除进程句柄，然后继续循环，直到所有进程都计算过为止。

## 9.5 实验5 Windows进程的“一生”

### 一、背景知识

Windows所创建的每个进程都从调用CreateProcess()API函数开始，该函数的任务是在对象管理器子系统内初始化进程对象。每一进程都以调用ExitProcess()或TerminateProcess()API函数终止。通常应用程序的框架负责调用ExitProcess()函数。对于C++运行库来说，这一调用发生在应用程序的main()函数返回之后。

#### 1. 创建进程

CreateProcess()调用的核心参数是可执行文件运行时的文件名及其命令行。表9-5-1详细地列出了每个参数的类型和名称。

表9-5-1 CreateProcess()函数的参数

参数名称	使用目的
LPCTSTR lpApplicationName	全部或部分地指明包括可执行代码的EXE文件的文件名
LPCTSTR lpCommandLine	向可执行文件发送的参数
LPSECURITY_ATTRIBUTES lpProcessAttributes	返回进程句柄的安全属性。主要指明这一句柄是否应该由其他子进程所继承
LPSECURITY_ATTRIBUTES lpThreadAttributes	返回进程的主线程的句柄的安全属性
BOOL bInheritHandle	一种标志，告诉系统允许新进程继承创建者进程的句柄
DWORD dwCreationFlags	特殊的创建标志（如CREATE_SUSPENDED）的位标记
LPVOID lpEnvironment	向新进程发送的一套环境变量；如为null值则发送调用者环境
LPCTSTR lpCurrentDirectory	新进程的启动目录
STARTUPINFO lpStartupInfo	STARTUPINFO结构，包括新进程的输入和输出配置的详情
LPPROCESS_INFORMATION lpProcessInformation	调用的结果块；发送新应用程序的进程和主线程的句柄和ID

可以指定第一个参数，即应用程序的名称，其中包括相对于当前进程的当前目录的全路径或者利用搜索方法找到的路径；lpCommandLine参数允许调用者向新应用程序发送数据；接下来的三个参数与进程和它的主线程以及返回的指向该对象的句柄的安全性有关。然后是标志参数，在dwCreationFlags参数中指明系统应该给予新进程什么行为。经常使用的标

志是 `CREATE_SUSPENDED`，告诉主线程立刻暂停。当准备好时，应该使用 `ResumeThread()` API 来启动进程。另一个常用的标志是 `CREATE_NEW_CONSOLE`，告诉新进程启动自己的控制台窗口，而不是利用父窗口。这一参数还允许设置进程的优先级，用以向系统指明，相对于系统中所有其他的活动进程来说，给此进程多少 CPU 时间。接着是 `CreateProcess()` 函数所需要的三个通常使用缺省值的参数。第一个参数是 `lpEnvironment` 参数，指明为新进程提供的环境；第二个参数是 `lpCurrentDirectory`，用于向主进程发送新进程的目录；第三个参数是 `STARTUPINFO` 数据结构，用于在必要时指明新应用程序的主窗口的外观。`CreateProcess()` 的最后一个参数是用于新进程对象及其主线程的句柄和 ID 的返回值缓冲区。以 `PROCESS_INFORMATION` 结构中返回的句柄来调用 `CloseHandle()` API 函数非常重要，因为如果不将这些句柄关闭的话，有可能危及主进程终止之前的任何未释放的资源。

## 2. 正在运行的进程

在系统中正在运行的进程至少拥有一个执行线程。通常这种进程使用主线程来指示它的存在。当主线程结束时，调用 `ExitProcess()` API 函数，通知系统终止它拥有的所有正在运行、准备运行或正在挂起的其他线程。当进程正在运行时，可以查看它的许多特性，其中少数特性也允许加以修改。

首先可查看的进程特性是系统进程标识符 (PID)，这可以利用 `GetCurrentProcessId()` API 函数。使用 `GetCurrentProcess()` 返回的 PID 在整个系统中都可使用。其他可显示当前进程信息的 API 函数还有 `GetStartupInfo()` 和 `GetProcessShutdownParameters()`，它们可给出进程在存活期内的配置详情。

通常，一个进程需要知道它的运行环境信息。例如 `GetModuleFileName()` API 函数和 `GetCommandLine()`，可以给出 `CreateProcess()` 中的参数以启动应用程序。在创建应用程序时可使用的另一个 API 函数是 `IsDebuggerPresent()`。另外，可利用 API 函数 `GetGuiResources()` 来查看进程的 GUI 资源，此函数既可返回指定进程中的打开的 GUI 对象的数目，也可返回指定进程中打开的 USER 对象的数目。进程的其他性能信息可通过 `GetProcessIoCounters()`、`GetProcessPriorityBoost()`、`GetProcessTimes()` 和 `GetProcessWorkingSetSize()` API 得到。以上这几个 API 函数都只需要具有 `PROCESS_QUERY_INFORMATION` 访问权限。

另一个可用于进程信息查询的 API 函数是 `GetProcessVersion()`，此函数只需进程的 PID (进程标识号)。

## 3. 终止进程

所有进程都是以调用 `ExitProcess()` 或者 `TerminateProcess()` 函数结束的。但最好使用前者而不要使用后者，因为进程在正常结束时要调用前者，此时它完成了它所有的关闭“职责”。而进程异常终止时通常调用后者，由于此时关闭时的途径不太正常，有可能引起错误的行为。

## 二、实验目的

(1) 通过创建进程、观察正在运行的进程和终止进程的程序设计和调试操作，进一步熟悉操作系统的进程概念，理解 Windows 进程的“一生”。

(2) 通过阅读和分析实验程序，学习创建进程、观察进程和终止进程的程序设计方法。



### 三、工具/准备工作

在开始本实验之前，请回顾教科书的相关内容。

您需要做以下准备：

- (1) 一台运行Windows操作系统的计算机。
- (2) 计算机中安装VisualC++6.0专业版或企业版。

### 四、实验内容与步骤

#### 1. 创建进程

步骤1: 在"开始"菜单中单击"程序", 启动Visual C++6.0。

步骤2: 选择菜单项文件——新建——C++ Source File，并选择文件名为procreate.cpp，然后按确定按钮，进入编程界面。

步骤3: 将清单5-1中的程序键入。

```
//清单5-1 创建子进程
#include<iostream.h>
#include<windows.h>
#include<stdio.h>
//创建进程的克隆并赋予其ID值
void StartClone(int nCloneID)
{
//获取当前文件名
TCHAR szFilename[MAX_PATH];
::GetModuleFileName(NULL,szFilename,MAX_PATH);
//得到用于子进程的命令行，并将文件名和ID通知子进程
TCHAR szCmdLine[MAX_PATH];
::sprintf(szCmdLine,"%s\\%s\\%d",szFilename,nCloneID);
//用于子进程的STARTUPINFO结构
STARTUPINFO si;
::ZeroMemory(reinterpret_cast<void*>(&si),sizeof(si));
si.cb=sizeof(si);
//返回用于子进程的进程信息
PROCESS_INFORMATION pi;
//利用同样的可执行文件和命令行创建进程
BOOL bCreateOK=::CreateProcess(
                                szFilename,           // exe文件名
                                szCmdLine,           // 命令行
                                NULL,                 // 缺省的进程安全性
                                NULL,                 // 缺省的线程安全性
                                FALSE,                // 不继承句柄
                                CREATE_NEW_CONSOLE,  // 使用新的控制台
                                NULL,                 // 新的环境
                                NULL,                 // 当前目录
                                &si,                 // 启动信息
                                &pi);                // 返回的进程信息
// 对子进程释放引用
```

```

        if(bCreateOK)
        {
            ::CloseHandle( pi.hProcess );
            ::CloseHandle(pi.hThread);
        }
    }

int main(int argc, char* argv[] )
{
    // 确定进程在列表中的位置
    int nClone(0);
    if(argc>1)
    {
        // 从第二个参数中提取克隆ID
        ::sscanf(argv[1], "%d", &nClone);
    }
    // 显示进程位置
    cout << "Process ID:" << ::GetCurrentProcessId()
        << ", Clone ID: " << nClone
        << endl;
    // 检查是否有创建子进程的需要
    const int c_nCloneMax = 5;
    if(nClone < c_nCloneMax)
    {
        // 发送新进程的命令行和克隆号
        StartClone( ++ nClone );
    }

    //在终止之前暂停一下(1/2秒)
    ::Sleep(500);
    return 0;
}

```

步骤4: 选择菜单项编译, 编译程序procreate.cpp。

步骤5: 选择菜单项编译, 运行procreate.exe程序。

本例展示的是一个简单使用CreateProcess()API函数的例子。首先形成简单的命令行, 提供当前的EXE文件的指定文件名和代表生成克隆进程的号码。CreateProcess()大多数参数可取缺省值, 但是使用了参数CREATE\_NEW\_CONSOLE, 指示新进程使用自己的控制台, 这使得本程序运行时产生了很多命令窗口。然后StartClone()函数关闭句柄返回main()函数。关闭程序之前, 每一个进程的主线程暂停500毫秒, 以便让用户看到程序运行中产生的窗口。

## 2. 正在运行的进程信息

本实验的程序中列出了用于进程信息查询的API函数GetProcessVersion()与GetVersioEx(), 可确定运行进程的操作系统版本号。

步骤1: 在"开始"菜单中单击"程序", 启动Visual C++6.0。

步骤2: 选择菜单项文件——新建——C++ Source File, 并选择文件名为version.cpp, 然

后按确定按钮，进入编程界面。

步骤3: 将清单5-2中的程序键入。

//清单5-2 获得进程和操作系统的版本信息

```
#include <windows.h>
#include <iostream.h>
void main()
{
    // 提取这个进程的ID号
    DWORD dwIdThis = ::GetCurrentProcessId();
    // 获得这一进程的版本，也可以发送0以便指明这一进程
    DWORD dwVerReq = ::GetProcessVersion(dwIdThis);
    WORD wMajorReq = (WORD)(dwVerReq > 16);
    WORD wMinorReq = (WORD)(dwVerReq & 0xffff);
    cout << "Process ID: " << dwIdThis
        << ", requires OS: " << wMajorReq << wMinorReq << endl;
    // 设置版本信息的数据结构，以便保存操作系统的版本信息
    OSVERSIONINFOEX osvix;
    ::ZeroMemory(&osvix, sizeof(osvix));
    osvix.dwOSVersionInfoSize = sizeof(osvix);
    // 提取版本信息和报告
    ::GetVersionEx(reinterpret_cast<LPOSVERSIONINFO>(&osvix));
    cout << "Running on OS: " << osvix.dwMajorVersion << "."
        << osvix.dwMinorVersion << endl;
    // 如果是 NTS (Windows) 系统，则提高其优先权
    if (osvix.dwPlatformId == VER_PLATFORM_WIN32_NT && osvix.dwMajorVersion >= 5)
    {
        // 改变优先级
        ::SetPriorityClass(
            ::GetCurrentProcess(), // 利用这一进程
            HIGH_PRIORITY_CLASS); // 改变为high
        // 报告给用户
        cout << "Task Manager should now now indicate this process is high priority ."
            << endl;
    }
}
```

步骤4: 选择菜单项编译，编译程序version.cpp。

步骤5: 选择菜单项编译，运行version.exe程序。

本例的程序向读者表明了如何获得当前的PID和所需的进程版本信息。为了运行这一程序。接着，程序演示了如何使用GetVersionEx() API函数来提取OSVERSIONINFOEX结构，这一数据块中包括了操作系统的版本信息。最后一段程序利用操作系统的版本信息，以确认运行的是Windows。代码接着将当前进程的优先级提到比正常级别高。

问题1: 按Ctrl+Alt+Del键，进入Windows任务管理器，在"应用程序"选项卡中右键单击本应用任务，在快捷菜单中选择"转到进程"。查看在Windows任务管理器的"进程"选项卡中，与本任务对应的进程映像名称是？

问题2: 右键单击该进程名, 在快捷菜单中选择"设置优先级"命令, 可以调整该进程的优先级, 如设置为"高"后, 重新运行version.exe程序, 屏幕显示有变化吗?为什么?

除了改变进程的优先级以外, 还可以对正在运行的进程执行几项其他的操作, 只要获得该进程句柄即可。SetProcessAffinityMask()API函数允许开发人员将线程映射到处理器上;SetProcessPriorityBoost()API可关闭前台应用程序优先级的提升;还有一个只对当前进程可用的API函数, 即SetProcessShutdownparameters(),可告诉系统如何终止该进程。

## 9. 6 实验 6 Windows 线程同步

### 一、背景知识

为了支持线程之间的相互约束关系, Windows提供了三类常用对象:核心应用服务、线程同步和线程通信, 其中, 开发人员可以使用线程同步对象来协调线程和进程的工作, 以使其共享信息并相互协作执行任务。线程同步对象包括互锁数据、临界段、事件、互斥体和信号等。

多线程编程中关键的一步是保护所有的共享资源, 这主要是用互锁函数、临界段和互斥体等。另一个关键部分是协调线程使其完成应用程序的任务, 为此, 可利用内核中的事件对象和信号。在进程内或进程间实现线程同步的最方便的方法是使用事件对象, 这一组内核对象允许一个线程对其接受信号状态进行直接控制(见表9-6-1)。

表9-6-1 用于管理事件对象的API

API名称	描述
CreateEvent()	在内核中创建一个新的事件对象。此函数允许有安全性设置、手工还是自动重置的标志以及初始时已接收还是未接收信号状态的标志
OpenEvent()	创建对已经存在的事件对象的引用。此API函数需要名称、继承标志和所需的访问级别
SetEvent()	将事件转化为已接收信号状态
ResetEvent()	将事件转化为非接收信号状态
PulseEvent()	将自动重置事件对象转化为已接收信号状态。当系统释放所有的等待它的线程时此种转化立即发生

而互斥体则是另一个可命名且安全的内核对象, 其主要目的是控制对共享资源的访问。拥有资源的线程创建互斥体, 所有想要访问该资源的线程应该在实际执行操作之前获得互斥体, 而在访问结束时立即释放互斥体, 以允许下一个等待线程获得互斥体, 然后接着进行下去。

与事件对象类似, 互斥体容易创建、打开、使用并清除。利用CreateMutex()API可创建互斥体, 创建时还可以指定一个初始的拥有权标志, 通过使用这个标志, 只有当线程完成了资源的所有的初始化工作时, 才允许创建线程释放互斥体。

为了获得互斥体, 首先, 想要访问资源的线程可使用OpenMutex()API来获得指向对象

的句柄;然后,线程将这个句柄提供给一个等待函数。当内核将互斥体对象发送给等待线程时,就表明该线程获得了互斥体的拥有权。当线程获得拥有权时,线程控制了对共享资源的访问。使用完资源后,必须设法尽快地放弃互斥体。放弃共享资源时需要在该对象上调用ReleaseMutex()API。然后系统负责将互斥体拥有权传递给下一个等待着的线程(由到达时间决定顺序)。

## 二、实验目的

在本实验中,通过对事件和互斥体对象的了解,加深对Windows线程同步的理解。

- (1)回顾系统进程、线程的有关概念,加深对Windows线程的理解。
- (2)了解事件和互斥体对象。
- (3)通过分析实验程序,了解管理事件对象的API
- (4)了解在进程中如何使用事件对象。
- (5)了解在进程中如何使用互斥体对象。
- (6)了解父进程创建子进程的程序设计方法。

## 三、工具/准备工作

在开始本实验之前,请回顾教科书的相关内容。

您需要做以下准备:

- (1)一台运行Windows操作系统的计算机。
- (2)计算机中需安装Visual C++6.0专业版或企业版。

## 四、实验内容与步骤

### 1.事件对象

清单6-1的程序展示了如何在进程间使用事件。父进程启动时,利用CreateEvent()API创建一个命名的、可共享的事件和子进程,然后等待子进程向事件发出信号并终止父进程。在创建时,子进程通过OpenEvent()API打开事件对象,调用SetEvent()API使其转化为已接收信号状态。两个进程在发出信号之后几乎立即终止。

步骤1:在"开始"菜单中单击"程序",启动Visual C++6.0。

步骤2:选择菜单项文件——新建——C++ Source File,并选择文件名为event.cpp,然后按确定按钮,进入编程界面。

步骤3:将清单6-1中的程序键入。

```
//清单6-1 创建和打开事件对象在进程间传送信号
#include <windows.h>
#include <iostream.h>
#include <stdio.h>
// 以下是句柄事件。实际中很可能使用共享的包含文件来进行通信
static LPCTSTR g_szContinueEvent = "w2kdg.EventDemo.event.Continue";
// 本方法只是创建了一个进程的副本,以子进程模式(由命令行指定)工作
BOOL CreateChild()
{
// 提取当前可执行文件的文件名
```

```

TCHAR szFilename[MAX_PATH];
::GetModuleFileName( NULL, szFilename, MAX_PATH);
// 格式化用于子进程的命令行, 指明它是一个EXE文件和子进程
TCHAR szCmdLine[MAX_PATH];
::sprintf(szCmdLine, "\\%s\\\"child\",szFilename);
// 子进程的启动信息结构
STARTUPINFO si;
::ZeroMemory(reinterpret_cast<void*>(&si),sizeof(si));
si.cb= sizeof(si);           //必须是本结构的大小
// 返回的子进程的进程信息结构
PROCESS_INFORMATION pi;
// 使用同一可执行文件和一个命令行创建子进程
BOOL bCreateOK = ::CreateProcess(
    szFilename,           // 生成的可执行文件名
    szCmdLine,           // 指示其行为与子进程一样的标志
    NULL,                // 子进程句柄的安全性
    NULL,                // 子线程句柄的安全性
    FALSE,               // 不继承句柄
    0,                   // 特殊的创建标志
    NULL,                // 新环境
    NULL,                // 当前目录
    &si,                 // 启动信息结构
    &pi);                // 返回的进程信息结构
// 释放对子进程的引用
if(bCreateOK)
{
    ::CloseHandle( pi.hProcess );
    ::CloseHandle( pi.hThread);
}
return(bCreateOK) ;
}
// 下面的方法创建一个事件和一个子进程, 然后等待子进程在返回前向事件发出信号
void WaitForChild()
{
    // create a new event object for the child process to use when releasing this process
    HANDLE hEventContinue = ::CreateEvent(
        NULL,           // 缺省的安全性, 子进程将具有访问权限
        TRUE,           // 手工重置事件
        FALSE,          // 初始时是非接受信号状态
        g_szContinueEvent); // 事件名称
    if (hEventContinue!=NULL)
    {
        cout << "event created " <<endl;
        // 创建子进程
        if(::CreateChild())
        {

```



```

    cout << " child created" << endl;
    // 等待，直到子进程发出信号
    cout << "parent waiting on child." << endl;
    ::WaitForSingleObject(hEventContinue, INFINITE);
    ::Sleep(1500);    // 删去这句试试
    cout << "parent received the event signaling from child" << endl;
}
// 清除句柄
::CloseHandle(hEventContinue);
hEventContinue = INVALID_HANDLE_VALUE;
}
}
// 以下方法在子进程被调用，其功能只是向父进程发出终止信号
void SignalParent()
{
    // 尝试打开句柄
    cout << "child process begining ....." << endl;
    HANDLE hEventContinue = ::OpenEvent(
        EVENT_MODIFY_STATE, // 所需求的最小访问权限
        FALSE,              // 不是可继承的句柄
        g_szContinueEvent); // 事件名称
    if (hEventContinue != NULL)
    {
        ::SetEvent(hEventContinue);
        cout << "event signaled" << endl;
    }
    // 清除句柄
    ::CloseHandle(hEventContinue);
    hEventContinue = INVALID_HANDLE_VALUE;
}
int main( int argc, char * argv[] )
{
    // 检查父进程或是子进程是否启动
    if(argc>1 &&::strcmp(argv[1],"child")==0)
    {
        // 向父进程创建的事件发出信号
        ::SignalParent();
    }
    else
    {
        // 创建一个事件并等待子进程发出信号
        ::WaitForChild();
        ::Sleep(1500);
        cout<< "Parent released" << endl;
    }
    return 0;
}

```

```
}
```

步骤4: 编译event.cpp

步骤5: 运行event.cpp

## 2.互斥体对象

清单6-2的程序中显示的类CCountUpDown使用了一个互斥体来保证对两个线程间单一数值的访问。每个线程都企图获得控制权来改变该数值，然后将该数值写入输出流中。创建者实际上创建的是互斥体对象，计数方法执行等待并释放，为的是共同使用互斥体所需的资源(也就是共享资源)。

步骤1:在"开始"菜单中单击"程序",启动Visual C++6.0。

步骤2:选择菜单项文件——新建——C++ Source File，并选择文件名为mutex.cpp，然后按确定按钮，进入编程界面。

步骤 3: 将清单 6-2 中的程序键入。

```
//清单6-2 利用互斥体保护共享资源
#include <windows.h>
#include <iostream.h>
// 利用互斥体来保护同时访问的共享资源
class CCountUpDown
{
public:
// 创建者创建两个线程来访问共享值
CCountUpDown(int nAccesses) :
    m_hThreadInc( INVALID_HANDLE_VALUE),
    m_hThreadDec( INVALID_HANDLE_VALUE),
    m_hMutexValue( INVALID_HANDLE_VALUE),
    m_nValue(0),
    m_nAccess( nAccesses)
{
    // 创建互斥体用于访问数值
    m_hMutexValue =::CreateMutex(
        NULL,        // 缺省的安全性
        TRUE,        // 初始时拥有，在所有的初始化结束时将释放
        NULL);       // 匿名的
    m_hThreadInc =::CreateThread(
        NULL,        //缺省的安全性
        0,           // 缺省堆栈
        IncThreadProc, // 类线程进程
        Reinterpret_cast <LPVOID> (this), // 线程参数
        0,           // 无特殊的标志
        NULL);       // 忽略返回的id
    m_hThreadDec = ::CreateThread(
        NULL,        // 缺省的安全性
        0,           // 缺省堆栈
        DecThreadProc, // 类线程进程
        Reinterpret_cast <LPVOID>(this), // 线程参数
```

```

0,          // 无特殊的标志
NULL);      // 忽略返回的id
// 允许另一线程获得互斥体
::ReleaseMutex(m_hMutexValue);
}
// 解除程序释放对对象的引用
virtual ~CCountUpDown()
{
::CloseHandle(m_hThreadInc);
::CloseHandle(m_hThreadDec);
::CloseHandle(m_hMutexValue);
}
// 简单的等待方法，在两个线程终止之前可暂停主调者
virtual void WaitForCompletion()
{
// 确保所有对象都已准备好
if(m_hThreadInc!=INVALID_HANDLE_VALUE&& m_hThreadDec!=INVALID_HANDLE_VALUE)
{
// 等待两者完成(顺序并不重要)
::WaitForSingleObject(m_hThreadInc, INFINITE);
::WaitForSingleObject(m_hThreadDec, INFINITE);
}
}
protected:
// 改变共享资源的简单的方法
virtual void DoCount(int nStep)
{
// 循环，直到所有的访问都结束为止
while (m_nAccess > 0)
{
// 等待访问数值
::WaitForSingleObject(m_hMutexValue, INFINITE);
// 改变并显示该值
m_nValue+=nStep;
cout<<"thread: " <<::GetCurrentThreadId()
<< "value: " << m_nvalue
<< "access: " <<m_nAccess <<endl;
// 发出访问信号并允许线程切换
--m_nAccess;
::Sleep(1000); // 使显示速度放慢
// 释放对数值的访问
::ReleaseMutex(m_hMutexValue);
}
}
static DWORD WINAPI IncThreadProc(LPVOID lpParam)
{

```

```

        // 将参数结实为'this' 指针
        CCountUpDown * pThis =reinterpret_cast< CCountUpDown*>(lpParam) ;
        // 调用对象的增加方法并返回一个值
        pThis->DoCount(+1) ;
        return(0);
    }
    static DWORD WINAPI DecThreadProc(LPVOID lpParam)
    {
        // 将参数解释为'this'指针
        CCountUpDown * pThis =reinterpret_cast<CCountUpDown*>(lpParam) ;
        // 调用对象的减少方法并返回一个值
        pThis->DoCount(-1) ;
        return(0) ;
    }
protected:
    HANDLE m_hThreadInc;
    HANDLE m_hThreadDec;
    HANDLE m_hMutexValue;
    int m_nValue;
    int m_nAccess;
}

void main()
{
    CCountUpDown ud(50);
    ud.waitForCompletion();
}

```

步骤4: 编译mutex.cpp

步骤5: 运行mutex.cpp

分析清单6-2的程序的运行结果，可以看到线程(加和减线程)的交替执行(因为Sleep()API允许Windows切换线程)。在每次运行之后，数值应该返回初始值(0)，因为在每次运行之后写入线程在等待队列中变成最后一个，内核保证它在其他线程工作时不会再运行。

## 9.7 实验 7 在 Windows Server 2003 系统应用程序使用内存的情况

### 一、实验目的

通过该实验，学生可以学会使用“性能监视器”来观察进程内存的变化情况，从而加深学生对存储管理的理解。

### 二、实验准备知识

可以参考实验 4 的准备知识和实验步骤。

## 9.8 实验 8 外设与主板的硬件连接和安装

### 一、实验目的

通过本实验，学生可以学会各种外设与微机的硬件连接，加深对设备的认识，学会在 Windows Server 2003 系统中各种设备（声卡、显卡、网卡、打印机等）的驱动程序的安装和设置，能够安装各种设备的驱动软件，加深对设备管理的认识。。

### 二、实验准备知识

各种外设：显示器、软盘驱动器、硬盘、网卡、打印机等等，与微机主板的硬件连接的方法、步骤。

### 三、实验步骤

1. 完成硬件部分的连接。
2. 连接好硬件后，部分硬件能够使用 Windows Server 2003 系统中默认的驱动程序，另外一些硬件设备还要安装专门的驱动程序。驱动程序的安装一般可以通过控制面板中的“添加/删除硬件”来完成。要准备好各种硬件设备的驱动软件。

## 9.9 实验 9 在 Windows Server 2003 系统中安装设备驱动程序

### 9.9.1 实验目的

通过本实验，学生可以学会在 2000 Server 系统中各种设备（声卡、显卡、网卡、打印机等）的驱动程序的安装和设置，能够安装各种设备的驱动软件，加深对设备管理的认识。

### 9.9.2 实验准备知识

上次实验完成了硬件部分的连接。连接好硬件后，部分硬件能够使用 Windows Server 2003Server 系统中默认的驱动程序，另外一些硬件设备还要安装专门的驱动程序。驱动程序的安装一般可以通过控制面板中的“添加/删除硬件”来完成。要准备好各种硬件设备的驱动软件。

## 9.10 实验 10 认知 UNIX 系统

### 一、实验目的

1. 掌握启、关闭 UNIX 系统的过程。
2. 掌握 UNIX 的一些基本操作，初步了解 UNIX 系统及其特性
3. 利用桌面系统对文件和目录进行操作
4. 学习配置图形环境

### 二、实验预备知识

本次实验以常见的SCO UNIX系统为环境，来熟悉对UNIX系统的操作。

SCO UNIX是非常典型的UNIX系统，用户可以通过文本方式和图形方式来操作它。在文本方式下，主要是键入各种shell命令和应用程序来工作。在图形方式下，大多数操作通过鼠标的点击完成。

UNIX的图形环境主要有以下几个成分：

- 1) Xwindows系统。它由美国麻省理工学院开发，是在UNIX系统中应用最广泛的窗口系统。它通过“X服务器”来控制终端硬件，直接与鼠标、键盘、显示器等相互作用，控制字体、颜色和背景。
- 2) 窗口管理器。UNIX中占主导地位的窗口管理器是Motif。它是判别窗口“观感”的一个客户进程，可以调节X服务器和其他客户程序之间的通信、管理运行程序、移动窗口、改变窗口大小等等。
- 3) Desktop（桌面系统）。它是一个客户进程，可以为运行应用程序、在目录间移动以及管理图标提供图形界面。Desktop控制桌面图标和目录的出现位置、桌面和目录菜单的内容，以及控制在桌面图标、目录和菜单上进行击键和拖动所产生的影响。

### 三、实验步骤：

#### 1. 启动系统

把 UNIX 系统安装到硬盘上后，就可以在开机时从硬盘上引导它，从而启动系统工作。启动系统的一般过程如下：

- 1) 打开主机电源
- 2) 硬件自动进行检测，屏幕上会出现一些信息。当屏幕上出现  
SCO OpenServer(TM) Release 5  
Boot  
:  
此时可以按下回车键。
- 3) 系统自动进行引导，并且在屏幕上显示一系列信息，当出现如下提示时：  
INIT : SINGLE USER MODE  
Type CONTROL -d to continue with normal startup,  
(or give the root password for system maintenance) :



如果要进入多用户分方式，则同时按下 Ctrl 键和 “D” 键。如果想以超级用户身份登录并进入系统维护状态，就要输入 root 用户的口令，然后按回车键。

注意：没有特殊任务，不要从 root 登录，以免误操作破坏系统。

- 4) 选择多用户方式后，系统会给出时间、日期提示。如果系统时钟不对，可按照格式 [年月日]时分[秒]输入正确的日期和时间，各用两位十进制数字表示。如 2004 年 3 月 24 日 10 时 28 分 0 秒可写成 040324102800。
- 5) 接下来系统会进行一系列的检查，最后出现用户注册窗口。在 scosysv login 框中输入帐户名，在 Password 框中输入相应口令，如果正确无误，则屏幕上出现桌面环境。

## 2. 关闭系统

要关闭系统，不能简单地关掉计算机电源，否则会损坏系统。关闭系统的方式有三种：利用图形界面，使用 shutdown 命令和 haltsys 命令。

### 1) 利用 haltsys 命令

haltsys 命令能立即终止系统运行，并不对用户发出警告。超级用户可以在控制台输入命令：#/etc/haltsys 或者 #haltsys，如果系统中还有用户在工作，他们会被强行退出，在此期间尚未完成的工作会丢失。系统自动关闭电源。

### 2) 利用 shutdown 命令

以 root 身份注册，在需要关闭系统时，可在#提示符后面输入命令：

```
#shutdown - gn
```

其中 n 代表关闭系统之前留给用户的准备时间是多少分钟。如果需要立即关机，可以输入：shutdown - g0。

### 3) 利用图形界面关闭系统

在桌面环境下，依次双击下面的图标：System Administration->System->System Shutdown Manager，从 Shutdown 菜单中选择 Begin Shutdown。这时系统会向所有用户发送默认广播消息，让各个用户尽快退出系统，然后在 60 秒钟后关闭系统。关机时间可以在该窗口钟重新设定。

当系统给出确认提示信息后，输入 y，然后按回车键，系统继续关闭工作。最后显示信息：

```
**Safe to Power Off**  
--or--  
**Press Any Key to Root**
```

这时候就可以安全地关闭电源。

## 3. 进入和退出图形系统

1) 在文本方式下进行注册，在输入帐户名和口令后，就会出现系统提示符\$（普通用户）或#（root 用户），在提示符后输入命令 startx，然后做出会话选择，单击 OK 后，就可进入桌面系统的主窗口。

2) 如果是在注册屏幕上输入帐户名和口令，则会在显示器上出现两个按钮：Continue my last session 和 Start a new session。单击其中一个按钮，然后单击 OK，同样出现

主窗口。

3) 在图形界面下，单击屏幕左上角的“File”，可以打开 File 菜单，再单击“Exit”，出现询问时单击 OK，就可以退出系统。

4) 图形界面和文本界面的切换。在图形界面下，同时按下 Ctrl、Alt、F1 三键，就切换到文本方式下；在文本方式下，同时按下 Ctrl、Alt、F2 三键，就又切换到图形界面下。

#### 4. 文件和目录操作

在桌面系统中对文件和目录的操作大多与 Windows 系统下的操作相仿。文件和目录都用图标表示，文件和子目录包含在主目录中。

##### 1) 创建、移动、拷贝文件

要创建一个文件，就从 File 菜单中选择 New File。在对话框中输入文件名。

要创建一个目录，就从 File 菜单中选择 New Directory。在对话框中输入目录名和路径，新目录成为原目录的子目录。

文件或目录的移动：用鼠标按钮 1 拖动被选中的文件或目录的图标，到目标目录中。或者把文件或目录的图标放到目标目录的图标上。

文件或目录的拷贝：用鼠标按钮 2 把文件或目录的图标从一个窗口拖到另一个窗口。也可以利用文件或目录的图标菜单来移动和拷贝文件与目录。

##### 2) 删除和恢复文件与目录

删除文件的一种方法是将相应文件的图标放到 Trash 图标上，另一种方法是，选择相应图标，然后从 File 菜单中选择 Discard。

被删除的文件和目录的图标会留在 Trash 桌面中。如果需要恢复，可以打开 Trash 窗口，双击所期望的图标，或者从 File 菜单中选择 Restore，就把文件和目录恢复到原来的位置。

#### 5. 其他操作

在桌面环境下，用户可以根据自己的需要和爱好，对系统的一些外在特性做修改。比如改变颜色、字体、鼠标特性、配置键盘、设置屏幕保护等等。

## 9.11 实验 11 认知 Linux 系统

### 一、实验目的

通过本实验，学生能够进行 Linux 的一些基本操作，初步了解、认识 Linux 操作系统，简单比较它与 UNIX、Windows Server 2003 的异同。

### 二、实验准备知识

LINUX 是一个叫 LINUS 的芬兰人写的操作系统，他的初衷是把只能在大型工作站上运行的 UNIX 移植到个人 PC 上来。本着共享的精神，他在网络上公布了 LINUX 0.01 版，很快，借助于世界各地的电脑高手们的业余“创作”，LINUX 迅速的成长壮大起来，以至于在网络中的很多 PC 机中安家落户，版本也做到了 7.0。

LINUX 的操作风格很类似 UNIX，对网络的支持也很完备。一个 LINUX 系统的用户无论是收发电子邮件还是文件传输，都可以轻松的解决问题。

LINUX 的文件系统与 DOS 基本是一样的，都具有目录和文件的概念，也都有一个根目录，稍稍不同的是目录用“/”代表，而不是 DOS 中的“\”。对文件和目录的操作命令也差别不大。下面是常用的一些命令。

含义	LINUX	DOS
改变当前目录	cd [path]	cd [path]
文件或目录的复制	cp filename1 filename2	copy filename1 filename2
文件目录列表	ls	dir
建立新目录	mkdir	md
分屏显示[文本]文件的内容	more [filename...]	more <[filename]
更改用户的口令	passwd	无
显示当前目录	pwd	cd
删除文件[可有通配符]	rm filename	del filename
删除空目录	rmdir pathname	rd pathname
清屏	clear	cls

Windows Server 2003 与 Linux 最明显的差别是，WINDOWS SERVER 2003 试图将人们熟悉的 Windows 图形用户界面引入到服务器环境中。在理想情况下，网站管理员可以主

要通过鼠标点击来进行 WINDOWS SERVER 2003（以及它捆绑的 Web 服务器软件 Microsoft Internet Information Server，简称 IIS）的管理。WINDOWS SERVER 2003 也捆绑了一整套微软的网站开发工具。

如果说 Windows Server 2003 有一个方面能够超过 Linux 的话，那就是，第三方软件厂商都愿意在 Windows Server 2003 上开发他们的软件。广告软件、搜索软件、数据库、应用服务器，以及电子商务购物软件，几乎肯定会有 Windows Server 2003 版本，而那些大的软件开发商，如 Oracle、Sun 和 IBM，刚刚开始向 Linux 投入力量。

### 三、实验步骤

步骤 1：开机，启动 Red Hat Linux 系统，在系统登录界面的左下方单击“会话”，在屏幕上弹出的菜单中选择会话方式，我们选择“GNOME”，单击“确定”按钮。

步骤 2：使用分配的用户名登录。将这个用户名输入在欢迎界面的“用户”输入框中，并输入口令回车，系统显示 Red Hat Linux 系统桌面。

步骤 3：单击屏幕下方的红帽子，在菜单中单击“系统工具”、“终端”命令，显示“终端”窗口。

步骤 4：在命令行提示符下输入更改口令的命令并更改口令。

在 GNOME 环境下执行以下步骤。

步骤 5：联系使用 GNOME 面板。GNOME 面板左下方有若干个图标，将鼠标移动到每个图标上停一会儿会看到操作提示。

步骤 6：显示面板菜单。GNOME 帮助图标上方有一个带箭头的子面板按钮，单击并查看菜单项。

步骤 7：打开和最小化/最大化视图。单击面板上的“OpenOffice.orgWriter”文字处理图标，打开一个文字处理视窗。

步骤 8：锁定工作站。单击红帽子，选择“锁住屏幕”命令，然后解锁。

步骤 9：使用 Linux 的 Shell。

步骤 10：使用 Linux 的文件系统。

步骤 11：练习 Linux 中常用命令。

由于以上几部的操作方式与 UNIX 相似，因此不再赘述。