

重庆邮电大学

学生实验实习报告册

课堂编号: SK13182A2130330002

学年学期: 2018 -2019 学年 ☒春 ☐秋学期

课程名称: 操作系统

学生学院: 软件工程学院

专业班级: 13001603

学生学号: 2016214052

学生姓名: 姜文泽

联系电话: 17783101834

重庆邮电大学教务处制

课程名称	操作系统	课程编号	A2130330
实验地点	综合实验楼 A511/A512	实验时间	2019-05-27
校外指导教师		校内指导教师	常光辉
实验名称	实验三 存储管理		
评阅人签字		成绩	

一、实验目的

实验 3.1 Windows Server 2016 内存结构

1) 通过实验了解 windows Server 2016 内存的使用，学习如何在应用程序中管理内存、体会 Windows 应用程序内存的基本原理以及使用方法。

2) 了解 windows Server 2016 的内存结构和虚拟内存的管理，进而了解 Windows 为使用内存而提供的一些扩展功能。

实验 3.2 Windows Server 2016 虚拟内存

1) 通过实验了解 Windows Server 2016 内存的使用，学习如何在应用程序中管理内存，体会 Windows 应用程序内存的简单性和自我防护能力。

2) 学习检查虚拟内存空间或对其进行操作。

3) 了解 Windows Server 2016 的内存结构和虚拟内存的管理，进而了解进程堆和 Windows 为使用内存而提供的一些扩展功能。

二、工具/准备工作

1. 回顾教材相关内容；
2. 安装有 Windows Server 2016 的计算机或虚拟机；
3. 系统中装有 Visual Studio 或 Visual C++ 6.0 或其他 C++编译器。

三、实验环境

操作系统：Windows Server 2016（虚拟机）

编程语言：C++

集成开发环境：Visual Studio 2019

四、实验步骤与实验过程

实验 3.1 Windows Server 2016 内存结构

Windows 提供了一个 API 即 `GetSystemInfo()`，以便用户能检查系统中虚拟内存的一些特性。程序 5-1 显示了如何调用该函数以及显示系统中当前内存的参数。

步骤 1: 登录进入 Windows Server 2016。

步骤 2: 在“开始”菜单中单击“程序”-“Microsoft Visual Studio 2019”，进入 Visual Studio 2019 的窗口。

步骤 3: 在 Visual Studio 2019 窗口的工具栏中单击“文件”按钮，在解决方案 OSLab 中新建一个项目，命名为 Lab3_1，并在源文件中添加新建项，命名为 Lab3_1.cpp，将实验指导书中的代码拷贝至该文件中。

步骤 4: 单击“生成”菜单中的“编译”命令，系统对 Lab3_1.cpp 进行编译。

步骤 5: 编译完成后，单击“生成”菜单中的“生成”命令，建立 Lab3_1.exe 可执行文件。

操作能否正常进行？如果不行，则可能的原因是什么？

由于程序代码是由 pdf 文件复制出来的，调整缩进与分号等规范后，使用了命名空间 std，减少代码冗余，调整过后，程序代码可以调通。

步骤 6: 在解决方案资源管理器中右键当前项目，并设置为启动项目。在工具栏单击“调试”-“开始执行（不调试）”按钮，执行 Lab3_1.exe 程序。

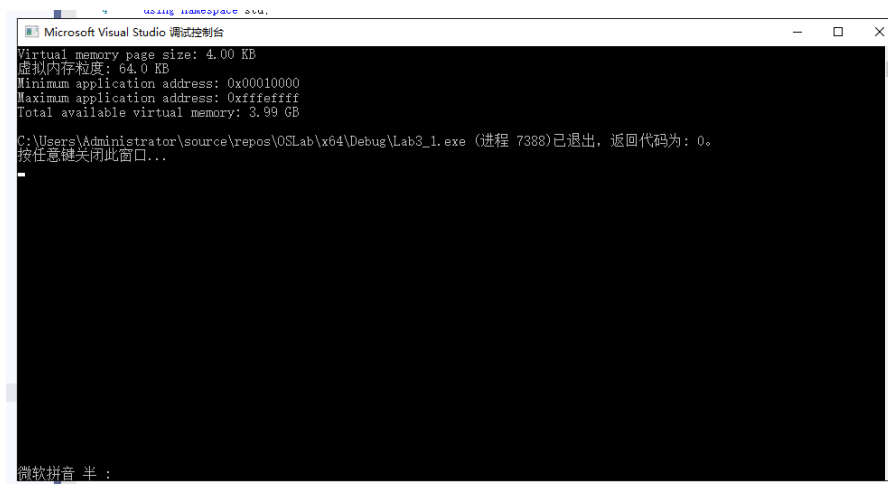


图 3-1 程序 Lab3_1 运行结果

运行结果（分行书写。如果运行不成功，则可能的原因是什么？）

- 1) 虚拟内存每页容量为: 4.00 KB
- 2) 最小应用地址: 0x00010000
- 3) 最大应用地址为: 0xfffffff
- 4) 当前可供应用程序使用的内存空间为: 3.99 GB

5) 当前计算机的实际内存大小为: 4 GB

阅读和分析程序 Lab3_1, 请回答问题:

1) 理论上每个 windows 应用程序可以独占的最大存储空间是: 4 GB

2) 在程序 Lab3_1 中, 用于检索系统中虚拟内存特性的 API 函数是:
GetSystemInfo()。

提示: 可供应用程序使用的内存空间实际上已经减去了开头与结尾两个 64KB 的保护区。虚拟内存空间中的 64KB 保护区是防止编程错误的一种 Windows 方式。任何对内存中这一区域的访问(读、写、执行)都将引发一个错误陷阱, 从而导致错误并终止程序的执行。也就是说, 假如用户有一个 NULL 指针(地址为 0), 但仍试图在此之前很近的地址处使用另一个指针, 这将因为试图从更低的保留区域读写数据, 从而产生意外错误并终止程序的执行。

实验 3.2 Windows Server 2016 虚拟内存

1. 虚拟内存的检测

清单 3-2 所示的程序使用 VirtualQueryEX() 函数来检查虚拟内存空间。

步骤 1: 登录进入 Windows Server 2016。

步骤 2: 在“开始”菜单中单击“程序”-“Microsoft Visual Studio 2019”, 进入 Visual Studio 2019 的窗口。

步骤 3: 在 Visual Studio 2019 窗口的工具栏中单击“文件”按钮, 在解决方案 OSLab 中新建一个项目, 命名为 Lab3_2, 并在源文件中添加新建项, 命名为 Lab3_2.cpp, 将实验指导书中的代码拷贝至该文件中。

清单 3-2 中显示一个 walkVM() 函数开始于某个进程可访问的最低端虚拟地址处, 并在其中显示各块虚拟内存的特性。虚拟内存中的块由 VirsualQueryEX() API 定义成连续块或具有相同状态(自由区, 已调配区等)的内存, 并分配以一组统一的保护标志(只读、可执行等)。

步骤 4: 单击“生成”菜单中的“编译”命令, 系统对 Lab3_2.cpp 进行编译。

步骤 5: 编译完成后, 单击“生成”菜单中的“生成”命令, 建立 Lab3_2.exe 可执行文件。

操作能否正常进行? 如果不行, 则可能的原因是什么?

由于程序代码是由 pdf 文件复制出来的, 调整缩进与分号等规范后, 使用了命名空间 std, 减少代码冗余, 调整过后, 程序代码可以调通。

步骤 6: 在解决方案资源管理器中右键当前项目，并设置为启动项目。在工具栏单击“调试” - “开始执行（不调试）”按钮，执行 Lab3_2.exe 程序。

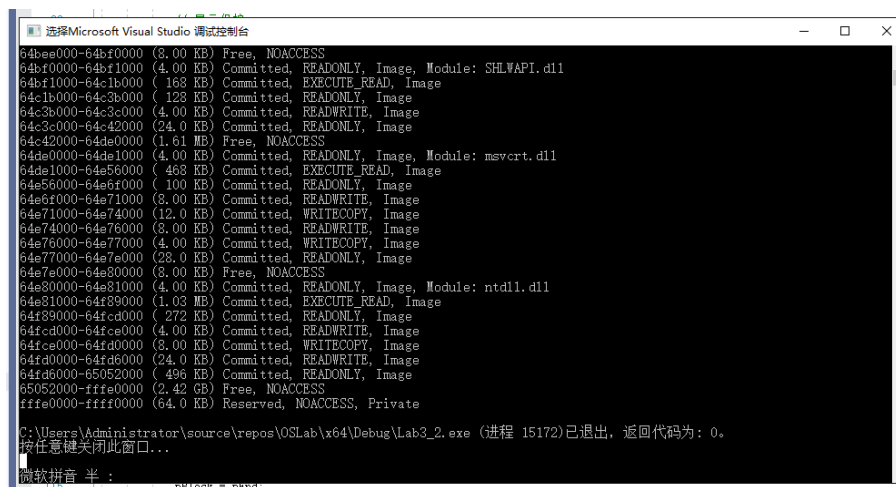


图 3-2 程序 Lab3_2 运行结果

1) 分析运行结果（如果运行不成功，则可能的原因是什么）

按 committed, reserved, free 等三种虚拟地址空间分别记录实验数据，其中“描述”是对该组数据的简单描述，例如，对下列一组数据：

00010000 - 00012000 <8.00KB> Committed, READWRITE, Private 可描述为：具有 READWRITE 权限的已调配私有内存区。

将系统当前的自由区（Free）虚拟地址空间填入表 3-1 中。

表 3-1 实验记录

地址	大小	虚拟空间类型	访问权限	描述
00010000-7ffe0000	(1.99GB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
7fff0000-6ae00000	(3.66GB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
6b300000-35630000	(3.15GB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
3564d000-35650000	(12.0KB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
35666000-35670000	(40.0KB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区

35674000-35680000	(48.0KB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
35681000-35690000	(60.0KB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
35692000-356a0000	(56.0KB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
356ad000-356b0000	(12.0KB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
356b1000-356c0000	(60.0KB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
356d1000-356e0000	(60.0KB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
356e2000-356f0000	(56.0KB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
356f1000-35710000	(124KB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
358d1000-358e0000	(60.0KB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
35a68000-35a70000	(32.0KB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
35bf1000-35c00000	(60.0KB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
372bd000-a7510000	(1.75GB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
a7643000-a8630000	(15.9MB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
a8659000-420c0000	(2.40GB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
42282000-42290000	(56.0KB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区

42386000-5a130000	(381MB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
5a2b5000-5cd40000	(42.5MB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
5cd64000-5f960000	(43.9MB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
5f9da000-61380000	(25.6MB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
6139e000-61450000	(712KB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
614ba000-61690000	(1.83MB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
618ad000-618b0000	(12.0KB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
61959000-61960000	(28.0KB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
61b7c000-61b80000	(16.0KB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
61c75000-63930000	(28.7MB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
639ef000-63e20000	(4.19MB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
63f41000-64390000	(4.30MB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
644f5000-64500000	(44.0KB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
647c8000-647e0000	(96.0KB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
64839000-648a0000	(412KB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区

6494c000-64a70000	(1.14MB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
64aa4000-64bc0000	(1.10MB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
64bee000-64bf0000	(8.00KB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
64c42000-64de0000	(1.61MB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
64e7e000-64e80000	(8.00KB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区
65052000-fffe0000	(2.42GB)	Free	NOACCESS	具有 NOACCESS 权限的自由内存区

将系统当前的已调配区（Committed）虚拟地址空间填入表 3-2 中。

表 3-2 实验记录

地址	大小	虚拟空间类型	访问权限	描述
7ffe0000-7ffe1000	(4.00KB)	Committed	READONLY	具有 READONLY 权限的已调配私有内存区
6ae64000-6ae6b000	(28.0KB)	Committed	READWRITE	具有 READWRITE 权限的已调配私有内存区
6b0f9000-6b0fc000	(12.0KB)	Committed	GUARD READWRITE	具有 GUARD READWRITE 权限的已调配私有内存区
6b0fc000-6b100000	(16.0KB)	Committed	READWRITE	具有 READWRITE 权限的已调配私有内存区
6b1fc000-6b1ff000	(12.0KB)	Committed	GUARD READWRITE	具有 GUARD READWRITE 权限的已调配私有内存区
6b1ff000-6b200000	(4.00KB)	Committed	READWRITE	具有 READWRITE 权限

				的已调配私有内存区
6b2fc000-6b2ff000	(12.0KB)	Committed	GUARD READWRITE	具有 GUARD READWRITE 权限的已调 配私有内存区
6b2ff000-6b300000	(4.00KB)	Committed	READWRITE	具有 READWRITE 权限 的已调配私有内存区
35630000-35640000	(64.0KB)	Committed	READWRITE	具有 READWRITE 权限 的已调配 Mapped 内存区
35640000-35642000	(8.00KB)	Committed	READWRITE	具有 READWRITE 权限 的已调配私有内存区
35650000-35666000	(88.0KB)	Committed	READONLY	具有 READONLY 权限的 已调配 Mapped 内存区
35670000-35674000	(16.0KB)	Committed	READONLY	具有 READONLY 权限的 已调配 Mapped 内存区
35680000-35681000	(4.00KB)	Committed	READONLY	具有 READONLY 权限的 已调配 Mapped 内存区
35690000-35692000	(8.00KB)	Committed	READWRITE	具有 READWRITE 权限 的已调配私有内存区
356a0000-356a1000	(4.00KB)	Committed	READWRITE	具有 READWRITE 权限 的已调配私有内存区
356b0000-356b1000	(4.00KB)	Committed	READWRITE	具有 READWRITE 权限 的已调配私有内存区
356c0000-356c7000	(28.0KB)	Committed	READWRITE	具有 READWRITE 权限 的已调配私有内存区
356d0000-356d1000	(4.00KB)	Committed	READWRITE	具有 READWRITE 权限 的已调配私有内存区
356e0000-356e2000	(8.00KB)	Committed	READONLY	具有 READONLY 权限的 已调配映射内存区
356f0000-356f1000	(4.00KB)	Committed	READWRITE	具有 READWRITE 权限 的已调配映射内存区

35710000-35733000	(140KB)	Committed	READWRITE	具有 READWRITE 权限的已调配私有内存区
35810000-358d1000	(772KB)	Committed	READONLY	具有 READONLY 权限的已调配映射内存区
358e0000-358e9000	(36.0KB)	Committed	READONLY	具有 READONLY 权限的已调配映射内存区
35a60000-35a65000	(20.0KB)	Committed	READONLY	具有 READONLY 权限的已调配映射内存区
35a70000-35bf1000	(1.50MB)	Committed	READONLY	具有 READONLY 权限的已调配映射内存区
35c00000-35c81000	(516KB)	Committed	READONLY	具有 READONLY 权限的已调配映射内存区
37000000-372bd000	(2.73MB)	Committed	READONLY	具有 READONLY 权限的已调配映射内存区
a7510000-a7515000	(20.0KB)	Committed	READONLY	具有 READONLY 权限的已调配映射内存区
a7610000-a7643000	(204KB)	Committed	READONLY	具有 READONLY 权限的已调配映射内存区
a8630000-a8631000	(4.00KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
a8631000-a8641000	(64.0KB)	Committed	EXECUTE_WRITECOPY	具有 EXECUTE_WRITECOPY 权限的已调配映像内存区
a8641000-a864b000	(40.0KB)	Committed	EXECUTE_READ	具有 EXECUTE_READ 权限的已调配映像内存区
a864b000-a864f000	(16.0KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
a864f000-a8650000	(4.00KB)	Committed	READWRITE	具有 READWRITE 权限的已调配映像内存区
a8650000-a8655000	(20.0KB)	Committed	READONLY	具有 READONLY 权限的

				已调配映像内存区
a8655000-a8656000	(4.00KB)	Committed	WRITECOPY	具有 WRITECOPY 权限的已调配映像内存区
a8656000-a8659000	(12.0KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
420c0000-420c1000	(4.00KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
420c1000-42214000	(1.32MB)	Committed	EXECUTE_READ	具有 EXECUTE_READ 权限的已调配映像内存区
42214000-4226e000	(360KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
4226e000-42271000	(12.0KB)	Committed	READWRITE	具有 READWRITE 权限的已调配映像内存区
42271000-42282000	(68.0KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
42290000-42291000	(4.00KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
42291000-42329000	(608KB)	Committed	EXECUTE_READ	具有 EXECUTE_READ 权限的已调配映像内存区
42329000-42374000	(300KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
42374000-42375000	(4.00KB)	Committed	READWRITE	具有 READWRITE 权限的已调配映像内存区
42375000-42376000	(4.00KB)	Committed	WRITECOPY	具有 WRITECOPY 权限的已调配映像内存区
42376000-42378000	(8.00KB)	Committed	READWRITE	具有 READWRITE 权限的已调配映像内存区
42378000-42386000	(56.0KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
5a130000-5a131000	(4.00KB)	Committed	READONLY	具有 READONLY 权限的

				已调配映像内存区
5a131000-5a1c1000	(576KB)	Committed	EXECUTE_READ	具有 EXECUTE_READ 权限的已调配映像内存区
5a1c1000-5a205000	(272KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
5a205000-5a206000	(4.00KB)	Committed	READWRITE	具有 READWRITE 权限的已调配映像内存区
5a206000-5a2b5000	(700KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
5cd40000-5cd41000	(4.00KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
5cd41000-5cd59000	(96.0KB)	Committed	EXECUTE_READ	具有 EXECUTE_READ 权限的已调配映像内存区
5cd59000-5cd5e000	(20.0KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
5cd5e000-5cd5f000	(4.00KB)	Committed	READWRITE	具有 READWRITE 权限的已调配映像内存区
5cd5f000-5cd64000	(20.0KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
5f960000-5f961000	(4.00KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
5f961000-5f99f000	(248KB)	Committed	EXECUTE_READ	具有 EXECUTE_READ 权限的已调配映像内存区
5f99f000-5f9bc000	(116KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
5f9bc000-5f9be000	(8.00KB)	Committed	READWRITE	具有 READWRITE 权限的已调配映像内存区
5f9be000-5f9da000	(112KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
61380000-61381000	(4.00KB)	Committed	READONLY	具有 READONLY 权限的

				已调配映像内存区
61381000-6138a000	(36.0KB)	Committed	EXECUTE_READ	具有 EXECUTE_READ 权限的已调配映像内存区
6138a000-61397000	(52.0KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
61397000-61398000	(4.00KB)	Committed	READWRITE	具有 READWRITE 权限的已调配映像内存区
61398000-6139e000	(24.0KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
61450000-61451000	(4.00KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
61451000-614a1000	(320KB)	Committed	EXECUTE_READ	具有 EXECUTE_READ 权限的已调配映像内存区
614a1000-614b4000	(76.0KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
614b4000-614b5000	(4.00KB)	Committed	READWRITE	具有 READWRITE 权限的已调配映像内存区
614b5000-614ba000	(20.0KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
61690000-61691000	(4.00KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
61691000-61761000	(832KB)	Committed	EXECUTE_READ	具有 EXECUTE_READ 权限的已调配映像内存区
61761000-6187e000	(1.11MB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
6187e000-61882000	(16.0KB)	Committed	READWRITE	具有 READWRITE 权限的已调配映像内存区
61882000-61883000	(4.00KB)	Committed	WRITECOPY	具有 WRITECOPY 权限的已调配映像内存区
61883000-618ad000	(168KB)	Committed	READONLY	具有 READONLY 权限的

				已调配映像内存区
618b0000-618b1000	(4.00KB)	Committed	READONLY	具有 READONLY 权限的 已调配映像内存区
618b1000-61923000	(456KB)	Committed	EXECUTE_READ	具有 EXECUTE_READ 权限的已调配映像内存区
61923000-61947000	(144KB)	Committed	READONLY	具有 READONLY 权限的 已调配映像内存区
61947000-61948000	(4.00KB)	Committed	READWRITE	具有 READWRITE 权限的 已调配映像内存区
61948000-61949000	(4.00KB)	Committed	WRITECOPY	具有 WRITECOPY 权限的 已调配映像内存区
61949000-61953000	(40.0KB)	Committed	READONLY	具有 READONLY 权限的 已调配映像内存区
61953000-61954000	(4.00KB)	Committed	WRITECOPY	具有 WRITECOPY 权限的 已调配映像内存区
61954000-61959000	(20.0KB)	Committed	READONLY	具有 READONLY 权限的 已调配映像内存区
61960000-61961000	(4.00KB)	Committed	READONLY	具有 READONLY 权限的 已调配映像内存区
61961000-61a1f000	(760KB)	Committed	EXECUTE_READ	具有 EXECUTE_READ 权限的已调配映像内存区
61a1f000-61ac0000	(644KB)	Committed	READONLY	具有 READONLY 权限的 已调配映像内存区
61ac0000-61ac4000	(16.0KB)	Committed	READWRITE	具有 READWRITE 权限的 已调配映像内存区
61ac4000-61ae0000	(112KB)	Committed	READONLY	具有 READONLY 权限的 已调配映像内存区
61ae0000-61ae1000	(4.00KB)	Committed	READONLY	具有 READONLY 权限的 已调配映像内存区
61ae1000-61b32000	(324KB)	Committed	EXECUTE_READ	具有 EXECUTE_READ 权

			D	限的已调配映像内存区
61b32000-61b71000	(252KB)	Committed	READONLY	具有 READONLY 权限的 已调配映像内存区
61b71000-61b72000	(4.00KB)	Committed	WRITECOPY	具有 WRITECOPY 权限 的已调配映像内存区
61b72000-61b75000	(12.0KB)	Committed	READWRITE	具有 READWRITE 权限 的已调配映像内存区
61b75000-61b7c000	(28.0KB)	Committed	READONLY	具有 READONLY 权限的 已调配映像内存区
61b80000-61b81000	(4.00KB)	Committed	READONLY	具有 READONLY 权限的 已调配映像内存区
61b81000-61c2c000	(684KB)	Committed	EXECUTE_READ	具有 EXECUTE_READ 权 限的已调配映像内存区
61c2c000-61c65000	(228KB)	Committed	READONLY	具有 READONLY 权限的 已调配映像内存区
61c65000-61c68000	(12.0KB)	Committed	READWRITE	具有 READWRITE 权限 的已调配映像内存区
61c68000-61c75000	(52.0KB)	Committed	READONLY	具有 READONLY 权限的 已调配映像内存区
63930000-63931000	(4.00KB)	Committed	READONLY	具有 READONLY 权限的 已调配映像内存区
63931000-639b9000	(544KB)	Committed	EXECUTE_READ	具有 EXECUTE_READ 权 限的已调配映像内存区
639b9000-639de000	(148KB)	Committed	READONLY	具有 READONLY 权限的 已调配映像内存区
639de000-639e1000	(12.0KB)	Committed	READWRITE	具有 READWRITE 权限 的已调配映像内存区
639e1000-639ef000	(56.0KB)	Committed	READONLY	具有 READONLY 权限的 已调配映像内存区
63e20000-63e21000	(4.00KB)	Committed	READONLY	具有 READONLY 权限的

				已调配映像内存区
63e21000-63f00000	(892KB)	Committed	EXECUTE_READ	具有 EXECUTE_READ 权限的已调配映像内存区
63f00000-63f2a000	(168KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
63f2a000-63f2c000	(8.00KB)	Committed	READWRITE	具有 READWRITE 权限的已调配映像内存区
63f2c000-63f41000	(84.0KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
64390000-64391000	(4.00KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
64391000-6442e000	(628KB)	Committed	EXECUTE_READ	具有 EXECUTE_READ 权限的已调配映像内存区
6442e000-6444b000	(116KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
6444b000-6444d000	(8.00KB)	Committed	READWRITE	具有 READWRITE 权限的已调配映像内存区
6444d000-644f5000	(672KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
64500000-64501000	(4.00KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
64501000-646c9000	(1.78MB)	Committed	EXECUTE_READ	具有 EXECUTE_READ 权限的已调配映像内存区
646c9000-6477e000	(724KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
6477e000-6477f000	(4.00KB)	Committed	READWRITE	具有 READWRITE 权限的已调配映像内存区
6477f000-64780000	(4.00KB)	Committed	WRITECOPY	具有 WRITECOPY 权限的已调配映像内存区
64780000-64783000	(12.0KB)	Committed	READWRITE	具有 READWRITE 权限

				的已调配映像内存区
64783000-64784000	(4.00KB)	Committed	WRITECOPY	具有 WRITECOPY 权限的已调配映像内存区
64784000-647c8000	(272KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
647e0000-647e1000	(4.00KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
647e1000-64812000	(196KB)	Committed	EXECUTE_READ	具有 EXECUTE_READ 权限的已调配映像内存区
64812000-64830000	(120KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
64830000-64831000	(4.00KB)	Committed	READWRITE	具有 READWRITE 权限的已调配映像内存区
64831000-64832000	(4.00KB)	Committed	WRITECOPY	具有 WRITECOPY 权限的已调配映像内存区
64832000-64833000	(4.00KB)	Committed	READWRITE	具有 READWRITE 权限的已调配映像内存区
64833000-64839000	(24.0KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
648a0000-648a1000	(4.00KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
648a1000-64914000	(460KB)	Committed	EXECUTE_READ	具有 EXECUTE_READ 权限的已调配映像内存区
64914000-64943000	(188KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
64943000-64944000	(4.00KB)	Committed	READWRITE	具有 READWRITE 权限的已调配映像内存区
64944000-6494c000	(32.0KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
64a70000-64a71000	(4.00KB)	Committed	READONLY	具有 READONLY 权限的

				已调配映像内存区
64a71000-64a83000	(72.0KB)	Committed	EXECUTE_READ	具有 EXECUTE_READ 权限的已调配映像内存区
64a83000-64a9e000	(108KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
64a9e000-64a9f000	(4.00KB)	Committed	READWRITE	具有 READWRITE 权限的已调配映像内存区
64a9f000-64aa4000	(20.0KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
64bc0000-64bc1000	(4.00KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
64bc1000-64bdd000	(112KB)	Committed	EXECUTE_READ	具有 EXECUTE_READ 权限的已调配映像内存区
64bdd000-64be4000	(28.0KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
64be4000-64be5000	(4.00KB)	Committed	READWRITE	具有 READWRITE 权限的已调配映像内存区
64be5000-64bee000	(36.0KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
64bf0000-64bf1000	(4.00KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
64bf1000-64c1b000	(168KB)	Committed	EXECUTE_READ	具有 EXECUTE_READ 权限的已调配映像内存区
64c1b000-64c3b000	(128KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
64c3b000-64c3c000	(4.00KB)	Committed	READWRITE	具有 READWRITE 权限的已调配映像内存区
64c3c000-64c42000	(24.0KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
64de0000-64de1000	(4.00KB)	Committed	READONLY	具有 READONLY 权限的

				已调配映像内存区
64de1000-64e56000	(468KB)	Committed	EXECUTE_READ	具有 EXECUTE_READ 权限的已调配映像内存区
64e56000-64e6f000	(100KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
64e6f000-64e71000	(8.00KB)	Committed	READWRITE	具有 READWRITE 权限的已调配映像内存区
64e71000-64e74000	(12.0KB)	Committed	WRITECOPY	具有 WRITECOPY 权限的已调配映像内存区
64e74000-64e76000	(8.00KB)	Committed	READWRITE	具有 READWRITE 权限的已调配映像内存区
64e76000-64e77000	(4.00KB)	Committed	WRITECOPY	具有 WRITECOPY 权限的已调配映像内存区
64e77000-64e7e000	(28.0KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
64e80000-64e81000	(4.00KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
64e81000-64f89000	(1.03MB)	Committed	EXECUTE_READ	具有 EXECUTE_READ 权限的已调配映像内存区
64f89000-64fcd000	(272KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区
64fcd000-64fce000	(4.00KB)	Committed	READWRITE	具有 READWRITE 权限的已调配映像内存区
64fce000-64fd0000	(8.00KB)	Committed	WRITECOPY	具有 WRITECOPY 权限的已调配映像内存区
64fd0000-64fd6000	(24.0KB)	Committed	READWRITE	具有 READWRITE 权限的已调配映像内存区
64fd6000-65052000	(496KB)	Committed	READONLY	具有 READONLY 权限的已调配映像内存区

将系统当前的保留区（Reserved）虚拟地址空间填入表 3-3 中。

表 3-3 实验记录

地址	大小	虚拟空间类型	访问权限	描述
7ffe1000-7fff0000	(60.0KB)	Reserved	READONLY	具有 READONLY 权限的保留私有内存区
6ae00000-6ae64000	(400KB)	Reserved	READONLY	具有 READONLY 权限的保留私有内存区
6ae6b000-6b000000	(1.58MB)	Reserved	READONLY	具有 READONLY 权限的保留私有内存区
6b000000-6b0f9000	(996KB)	Reserved	READONLY	具有 READONLY 权限的保留私有内存区
6b100000-6b1fc000	(0.98MB)	Reserved	READONLY	具有 READONLY 权限的保留私有内存区
6b200000-6b2fc000	(0.98MB)	Reserved	READONLY	具有 READONLY 权限的保留私有内存区
35642000-3564d000	(44.0KB)	Reserved	READONLY	具有 READONLY 权限的保留私有内存区
356a1000-356ad000	(48.0KB)	Reserved	READONLY	具有 READONLY 权限的保留私有内存区
356c7000-356d0000	(36.0KB)	Reserved	READONLY	具有 READONLY 权限的保留私有内存区
35733000-35810000	(884KB)	Reserved	READONLY	具有 READONLY 权限的保留私有内存区
358e9000-35a60000	(1.46MB)	Reserved	READONLY	具有 READONLY 权限的保留映射内存区
35a65000-35a68000	(12.0KB)	Reserved	READONLY	具有 READONLY 权限的保留映射内存区
35c81000-37000000	(19.4MB)	Reserved	READONLY	具有 READONLY 权限的保留映射内存区

a7515000-a7610000	(0.98MB)	Reserved	READONLY	具有 READONLY 权限的保留映射内存区
fffe0000-ffff0000	(64.0KB)	Reserved	NOACCESS	具有 NOACCESS 权限的保留私有内存区

2) 从上述输出结果, 对照分析清单 3-2 的程序, 请简单描述程序运行的流程: 遍历当前进程的整个虚拟内存并显示在控制台中。具体流程图如图 3-3。

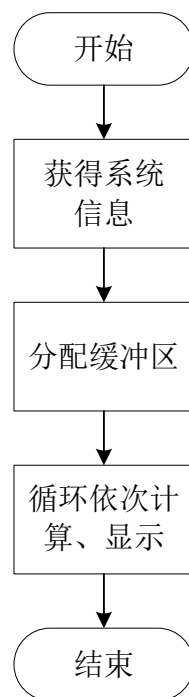


图 3-3 Lab3_2 流程图

2. 虚拟内存的分配与释放

能正确使用系统函数 `GetMemoryStatus()` 和数据结构 `MEMORY_STATUS` 了解系统内存和虚拟存储空间使用情况, 会使用 `VirtualAlloc()` 函数和 `VirtualFree()` 函数分配和释放虚拟内存空间。

步骤 1: 在 Visual Studio 2019 窗口的工具栏中单击“文件”按钮, 在解决方案 OSLab 中新建一个项目, 命名为 Lab3_3, 并在源文件中添加新建项, 命名为 Lab3_3.cpp, 将实验指导书中的代码拷贝至该文件中。并对其进行编译。

步骤 2: 编译完成后, 单击“生成”菜单中的“生成”命令, 建立 Lab3_3.exe 可执行文件。

操作能否正常进行? 如果不行, 则可能的原因是什么?

由于程序代码是由 pdf 文件复制出来的，调整缩进与分号等规范后，使用了命名空间 std，减少代码冗余，调整过后，仍存在问题。

本次实验是在 VS 中进行的，无法直接按照指导书中的要求进行调试，因此替换了头文件之后即可以运行。

步骤 3：在解决方案资源管理器中右键当前项目，并设置为启动项目。在工具栏单击“调试” - “开始执行（不调试）”按钮，执行 Lab3_3.exe 程序。

分析程序 Lab3_3.cpp 的运行结果

1) 请描述运行结果（如果运行不成功，则可能的原因是什么？）：

Current Memory Status is :

Total Physical Memory is 4094 MB

Available Physical Memory is 2046 MB

Total Page File is 5950 MB

Available Page File is 4078 MB

Total Virtual Memory is 134217727 MB

Available Virsual memory is 134217713 MB

Memory Load is 50 %

Now Allocate 32M Virsual Memory and 2M Physical Memory

Current Memory Status is :

Total Physical Memory is 4094 MB

Available Physical Memory is 2044 MB

Total Page File is 5950 MB

Available Page File is 4044 MB

Total Virtual Memory is 134217727 MB

Available Virsual memory is 134217679 MB

Memory Load is 50 %

Now Release 32M Virsual Memory and 2M Physical Memory

Current Memory Status is :

Total Physical Memory is 4094 MB

Available Physical Memory is 2046 MB

Total Page File is 5950 MB

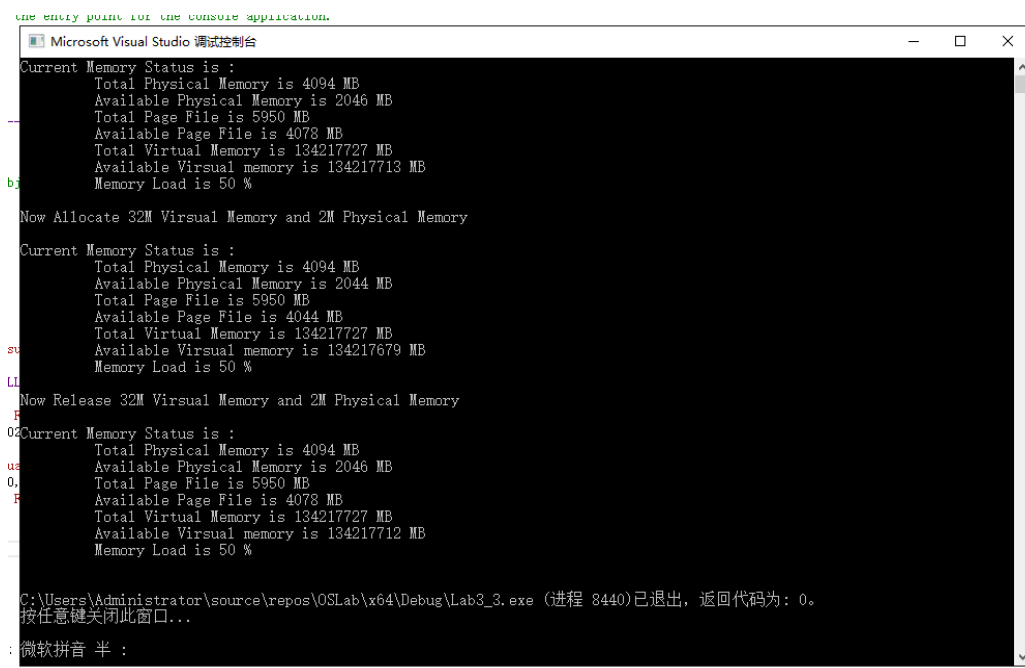
Available Page File is 4078 MB

Total Virtual Memory is 134217727 MB

Available Virtual memory is 134217712 MB

Memory Load is 50 %

实验结果如图 3-4 所示。



```
the entry point for the console application.
Microsoft Visual Studio 调试控制台
Current Memory Status is :
Total Physical Memory is 4094 MB
Available Physical Memory is 2046 MB
Total Page File is 5950 MB
Available Page File is 4078 MB
Total Virtual Memory is 134217727 MB
Available Virtual memory is 134217713 MB
Memory Load is 50 %

Now Allocate 32M Virtual Memory and 2M Physical Memory

Current Memory Status is :
Total Physical Memory is 4094 MB
Available Physical Memory is 2044 MB
Total Page File is 5950 MB
Available Page File is 4044 MB
Total Virtual Memory is 134217727 MB
Available Virtual memory is 134217679 MB
Memory Load is 50 %

Now Release 32M Virtual Memory and 2M Physical Memory

Current Memory Status is :
Total Physical Memory is 4094 MB
Available Physical Memory is 2046 MB
Total Page File is 5950 MB
Available Page File is 4078 MB
Total Virtual Memory is 134217727 MB
Available Virtual memory is 134217712 MB
Memory Load is 50 %

C:\Users\Administrator\source\repos\OSLab\x64\Debug\Lab3_3.exe (进程 8440)已退出, 返回代码为: 0。
按任意键关闭此窗口...
```

图 3-4 程序 Lab3_3 运行结果

2) 根据运行输出结果, 若要改变分配和回收的虚拟内存和物理内存的大小, 要改变程序代码的语句, 分别为:

分配虚拟内存: BaseAddr = ::VirtualAlloc(NULL, 1024 * 1024 * 32, MEM_RESERVE
| MEM_COMMIT, PAGE_READWRITE);

分配物理内存: str = (char*)malloc(1024 * 1024 * 2);

回收虚拟内存: ::VirtualFree(BaseAddr, 0, MEM_RELEASE) == 0

回收物理内存: free(str);

3) 根据运行输出结果, 对照分析 4-2 程序, 可以看出程序运行的流程吗? 请简单描述:

程序运行流程如图 3-5 所示。

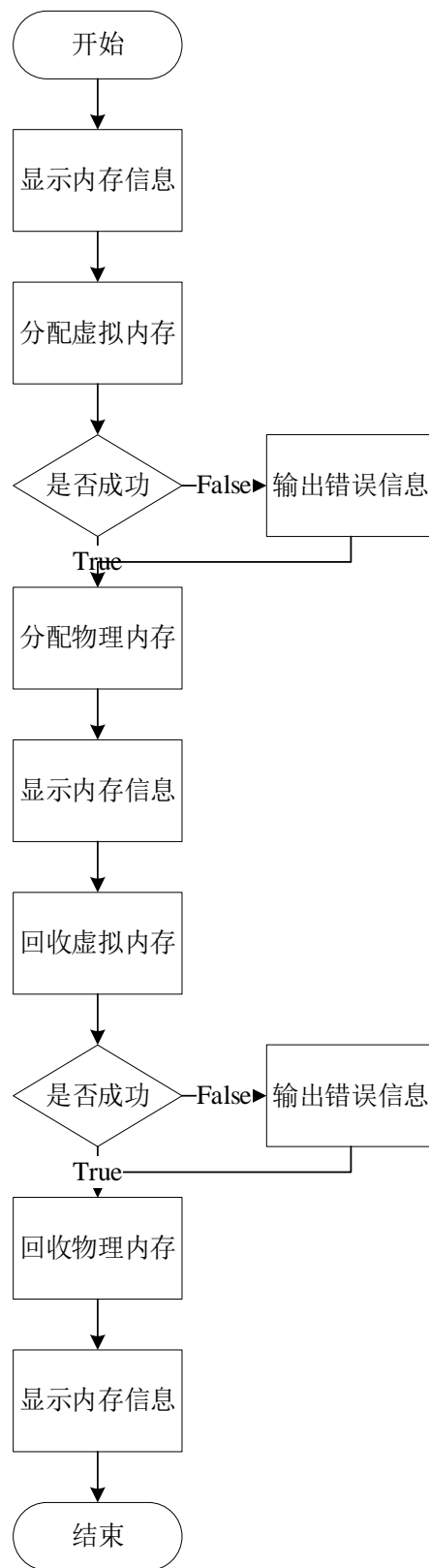


图 3-5 Lab3_3 流程图

五、实验结果与分析

实验结果与分析见实验步骤与实验过程。

六、实验心得体会

通过本次实验我了解了 Windows Server 2016 内存的使用，学习了如何在应用程序中管理内存、体会 Windows 应用程序内存的基本原理以及使用方法。同时了解 windows Server 2016 的内存结构和虚拟内存的管理，进而了解 Windows 为使用内存而提供的一些扩展功能。通过编写程序了解 Windows Server 2016 的内存结构和虚拟内存的管理，进而了解进程堆和 Windows 为使用内存而提供的一些扩展功能。

在实验中，通过绘制流程图对内存管理有了更加深刻的认识。

附录 程序清单

清单 3-1

```
1.  // 工程 vmeminfo
2.  #include <windows.h>
3.  #include <iostream>
4.  using namespace std;
5.  #include <shlwapi.h>
6.  #include <iomanip>
7.  #pragma comment(lib, "shlwapi.lib")
8.
9.  void main() {
10.     // 首先，让我们获得系统信息
11.     SYSTEM_INFO si;
12.     ::ZeroMemory(&si, sizeof(si));
13.     ::GetSystemInfo(&si);
14.
15.     // 格式化
16.     TCHAR szPageSize[MAX_PATH];
17.     ::StrFormatByteSize(si.dwPageSize, szPageSize, MAX_PATH);
18.     TCHAR dwAllocationGranularity[MAX_PATH];
19.     ::StrFormatByteSize(si.dwAllocationGranularity, dwAllocationGranularity, MAX_PATH);
20.     DWORD dwMemSize = (DWORD)si.lpMaximumApplicationAddress - (DWORD)si.lpMinimumApplication
        Address + 1;
21.     TCHAR szMemSize[MAX_PATH];
22.     ::StrFormatByteSize(dwMemSize, szMemSize, MAX_PATH);
23.
24.     // 将内存信息显示出来
25.     cout << "Virtual memory page size: " << szPageSize << endl;
26.     cout << "虚拟内存粒度: " << dwAllocationGranularity << endl;
27.
28.     cout.fill('0');
29.     cout << "Minimum application address: 0x" << hex << setw(8)
30.         << (DWORD)si.lpMinimumApplicationAddress << endl;
```

```

31. cout << "Maximum application address: 0x" << hex << setw(8)
32.     << (DWORD)si.lpMaximumApplicationAddress << endl;
33. cout << "Total available virtual memory: " << szMemSize << endl;
34. }

```

清单 3-2

```

1.  // 检测进程的虚拟地址空间
2.  // 工程 vmwalker
3.  #include <windows.h>
4.  #include <iostream>
5.  using namespace std;
6.  #include <shlwapi.h>
7.  #include <iomanip>
8.  #pragma comment(lib, "Shlwapi.lib")
9.
10. // 以可读方式对用户显示保护的辅助方法。
11. // 保护标记表示允许应用程序对内存进行访问的类型
12. // 以及操作系统强制访问的类型
13. inline bool TestSet(DWORD dwTarget, DWORD dwMask) {
14.     return ((dwTarget & dwMask) == dwMask);
15. }
16.
17. # define SHOWMASK(dwTarget, type) \
18. if (TestSet(dwTarget, PAGE_##type) ) \
19.     {cout << " , " << #type;}
20.
21. void ShowProtection(DWORD dwTarget)
22. {
23.     SHOWMASK(dwTarget, READONLY);
24.     SHOWMASK(dwTarget, GUARD);
25.     SHOWMASK(dwTarget, NOCACHE);
26.     SHOWMASK(dwTarget, READWRITE);
27.     SHOWMASK(dwTarget, WRITECOPY);
28.     SHOWMASK(dwTarget, EXECUTE);
29.     SHOWMASK(dwTarget, EXECUTE_READ);
30.     SHOWMASK(dwTarget, EXECUTE_READWRITE);
31.     SHOWMASK(dwTarget, EXECUTE_WRITECOPY);
32.     SHOWMASK(dwTarget, NOACCESS);
33. }
34.
35. // 遍历整个虚拟内存并对用户显示其属性的工作程序的方法
36. void WalkVM(HANDLE hProcess) {
37.     // 首先, 获得系统信息
38.     SYSTEM_INFO si;
39.     ::ZeroMemory(&si, sizeof(si));

```

```

40.     ::GetSystemInfo(&si);
41.
42.     // 分配要存放信息的缓冲区
43.     MEMORY_BASIC_INFORMATION mbi;
44.     ::ZeroMemory(&mbi, sizeof(mbi));
45.
46.     // 循环整个应用程序地址空间
47.     LPCVOID pBlock = (LPCVOID)si.lpMinimumApplicationAddress;
48.     while (pBlock < si.lpMaximumApplicationAddress)
49.     {
50.         // 获得下一个虚拟内存块的信息
51.         if (::VirtualQueryEx(
52.             hProcess, // 相关的进程
53.             pBlock, // 开始位置
54.             &mbi, // 缓冲区
55.             sizeof(mbi)) == sizeof(mbi)) // 大小的确认
56.         {
57.             // 计算块的结尾及其大小
58.             LPCVOID pEnd = (PBYTE)pBlock + mbi.RegionSize;
59.             TCHAR szSize[MAX_PATH];
60.             ::StrFormatByteSize(mbi.RegionSize, szSize, MAX_PATH);
61.
62.             // 显示块地址和大小
63.             cout.fill('0');
64.             cout << hex << setw(8) << (DWORD)pBlock
65.                 << "-" << hex << setw(8) << (DWORD)pEnd
66.                 << "<< " (" << szSize
67.                 << (::strlen(szSize) == 7 ? " (" : " ( ") << szSize
68.                 << ") ";
69.
70.             // 显示块的状态
71.             switch (mbi.State) {
72.                 case MEM_COMMIT:
73.                     cout << "Committed";
74.                     break;
75.                 case MEM_FREE:
76.                     cout << "Free";
77.                     break;
78.                 case MEM_RESERVE:
79.                     cout << "Reserved";
80.                     break;
81.             }
82.
83.             // 显示保护
84.             if (mbi.Protect == 0 && mbi.State != MEM_FREE)
85.             {

```

```

86.         mbi.Protect = PAGE_READONLY;
87.     }
88.     ShowProtection(mbi.Protect);
89.
90.     // 显示类型
91.     switch (mbi.Type) {
92.     case MEM_IMAGE:
93.         cout << ", Image";
94.         break;
95.     case MEM_MAPPED:
96.         cout << ", Mapped";
97.         break;
98.     case MEM_PRIVATE:
99.         cout << ", Private";
100.        break;
101.    }
102.
103.    // 检验可执行的影像
104.    TCHAR szFilename[MAX_PATH];
105.    if (::GetModuleFileName(
106.        (HMODULE)pBlock, // 实际虚拟内存的模块句柄
107.        szFilename, // 完全指定的文件名称
108.        MAX_PATH) > 0) // 实际使用的缓冲区大小
109.    {
110.        // 除去路径并显示
111.        ::PathStripPath(szFilename);
112.        cout << ", Module: " << szFilename;
113.    }
114.    cout << endl; // 移动块指针以获得下一个块
115.    pBlock = pEnd;
116.    }
117.    }
118.}
119.
120.void main() {
121.    // 遍历当前进程的虚拟内存
122.    ::WalkVM(::GetCurrentProcess());
123.}

```

清单 3-3

```

1.  #include <Windows.h>
2.  #include <iostream>
3.  using namespace std;
4.  // Defines the entry point for the console application.
5.

```

```

6.  #ifdef _DEBUG
7.  #define new DEBUG_NEW
8.  #undef THIS_FILE
9.  static char THIS_FILE[] = __FILE__;
10. #endif
11.
12. void GetMemSta(void);
13. //The one and only application object CWinApp theApp;
14.
15. int main()
16. {
17.     int nRetCode = 0;
18.     LPVOID BaseAddr;
19.     char* str;
20.
21.     GetMemSta();
22.
23.     printf("Now Allocate 32M Virsual Memory and 2M Physical Memory\n\n");
24.
25.     BaseAddr = ::VirtualAlloc(NULL, 1024 * 1024 * 32, MEM_RESERVE | MEM_COMMIT, PAGE_READWRITE);
    //分配虚拟内存
26.     if (BaseAddr == NULL)
27.         printf("Virsual Allocate Fail.\n");
28.     str = (char*)malloc(1024 * 1024 * 2); //分配内存
29.
30.     GetMemSta();
31.
32.     printf("Now Release 32M Virsual Memory and 2M Physical Memory\n\n");
33.     if (::VirtualFree(BaseAddr, 0, MEM_RELEASE) == 0) //释放虚拟内存
34.         printf("Release Allocate Fail.\n");
35.     free(str); //释放内存
36.
37.     GetMemSta();
38.
39.     return nRetCode;
40. }
41.
42. void GetMemSta(void)
43. {
44.     MEMORYSTATUS MemInfo;
45.     GlobalMemoryStatus(&MemInfo);
46.
47.     printf("Current Memory Status is :\n");
48.     printf("\t Total Physical Memory is %d MB\n", MemInfo.dwTotalPhys / (1024 * 1024));
49.     printf("\t Available Physical Memory is %d MB\n", MemInfo.dwAvailPhys / (1024 * 1024));
50.     printf("\t Total Page File is %d MB\n", MemInfo.dwTotalPageFile / (1024 * 1024));

```

```
51.  printf("\t Available Page File is %d MB\n", MemInfo.dwAvailPageFile / (1024 * 1024));
52.  printf("\t Total Virtual Memory is %d MB\n", MemInfo.dwTotalVirtual / (1024 * 1024));
53.  printf("\t Available Virsual memory is %d MB\n", MemInfo.dwAvailVirtual / (1024 * 1024));
54.  printf("\t Memory Load is %d %%\n\n", MemInfo.dwMemoryLoad);
55. }
```