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 0SLab 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。

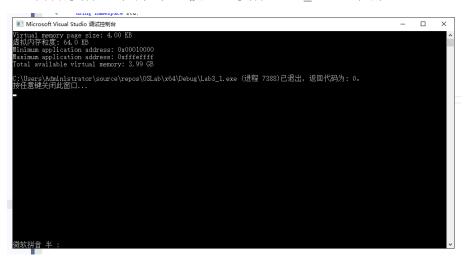


图 3-1 程序 Lab3 1 运行结果

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: 0xfffeffff
- 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 Virtual Query EX() 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 0SLab 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 定义成连续快或具有相同状态(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。

地址	大小	虚拟空间类型	访问权限	描述
00010000 755 0000	(1.00CP)	Г	NO A COEGG	具有 NOACCESS 权限的自
00010000-7ffe0000	(1.99GB)	Free	NOACCESS	由内存区
7550000 (00000	(2.66CP)	E		具有 NOACCESS 权限的自
7fff0000-6ae00000	(3.66GB)	Free	NOACCESS	由内存区
c1 200000 25 c20000		E	No Address	具有 NOACCESS 权限的自
6b300000-35630000	(3.15GB)	Free	NOACCESS	由内存区
25.4 1000 25.50000	(12 OKD)	Г	NO A COEGG	具有 NOACCESS 权限的自
3564d000-35650000	(12.0KB)	Free	NOACCESS	由内存区
35666000-35670000 (40	(40 OKD)	E	NO A CCEGG	具有 NOACCESS 权限的自
	(40.0KB)	Free	NOACCESS	由内存区

表 3-1 实验记录

		T	T	_
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
33001000-33070000	(00.0141)	1100	NOTICELSS	由内存区
25602000 256,0000	(56 OVD)	Erro	NOACCEGG	具有 NOACCESS 权限的自
35692000-356a0000	(56.0KB)	Free	NOACCESS	由内存区
256-1000 2561,0000	(12 OKD)	F	NO A COEGG	具有 NOACCESS 权限的自
356ad000-356b0000	(12.0KB)	Free	NOACCESS	由内存区
25(1,1000,256,0000	(60 0KD)	T.	NO V CCEGG	具有 NOACCESS 权限的自
356b1000-356c0000	(60.0KB)	Free	NOACCESS	由内存区
25611000 256 0000	(60 OMD)		NO F CCE.	具有 NOACCESS 权限的自
356d1000-356e0000	(60.0KB)	Free	NOACCESS	由内存区
254 2000 2547000				具有 NOACCESS 权限的自
356e2000-356f0000	(56.0KB)	Free	NOACCESS	由内存区
27.521.000.2771.0000	(124KB)	Free	NOACCESS	具有 NOACCESS 权限的自
356f1000-35710000				由内存区
250 11000 250 0000	(60 OVD)		NO A CODEG	具有 NOACCESS 权限的自
358d1000-358e0000	(60.0KB)	Free	NOACCESS	由内存区
25, 60000 25, 70000	(22 01/D)	Frag	NOACCESS	具有 NOACCESS 权限的自
35a68000-35a70000	(32.0KB)	Free	NOACCESS	由内存区
251 51000 25, 00000	(60 0KD)	Г	NOAGGEGG	具有 NOACCESS 权限的自
35bf1000-35c00000	(60.0KB)	Free	NOACCESS	由内存区
2721 1000 7510000	(1.75CD)	Г	NOAGGEGG	具有 NOACCESS 权限的自
372bd000-a7510000	(1.75GB)	Free	NOACCESS	由内存区
7642000 0620000	(15.03.50)	T.	NO A GGEGG	具有 NOACCESS 权限的自
a7643000-a8630000	(15.9MB)	Free	NOACCESS	由内存区
0.50000 420 0000	(2.4067)	T.	NOACCESS	具有 NOACCESS 权限的自
a8659000-420c0000	(2.40GB)	Free		由内存区
40000000 40000000	(50 0175)		No i dense	具有 NOACCESS 权限的自
42282000-42290000	(56.0KB)	Free	NOACCESS	由内存区

		T	T	_
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
	(¬2.5IVID)	1100	HOMECESS	由内存区
50d64000 5f060000	(42 OMD)	Evas	NOACCESS	具有 NOACCESS 权限的自
5cd64000-5f960000	(43.9MB)	Free	NOACCESS	由内存区
550 J-000 C1200000	(25 (MD)	Г	NO A CCEGG	具有 NOACCESS 权限的自
5f9da000-61380000	(25.6MB)	Free	NOACCESS	由内存区
6120-000 61450000	(712VD)		NO A COEGG	具有 NOACCESS 权限的自
6139e000-61450000	(712KB)	Free	NOACCESS	由内存区
C141-000 C1C0000	(1.00) (1)	Г	NO A COEGG	具有 NOACCESS 权限的自
614ba000-61690000	(1.83MB)	Free	NOACCESS	由内存区
Z10 1000 Z101 0000	(12.0)(D)	_	NOACCEGG	具有 NOACCESS 权限的自
618ad000-618b0000	(12.0KB)	Free	NOACCESS	由内存区
	(28.0KB)	Free	NOACCESS	具有 NOACCESS 权限的自
61959000-61960000				由内存区
61h7a000 611 00000	(16 OVD)	Г	NO A CCECC	具有 NOACCESS 权限的自
61b7c000-61b80000	(16.0KB)	Free	NOACCESS	由内存区
C1-75000 C2020000	(20.73.47)	Erro	NOACCESS	具有 NOACCESS 权限的自
61c75000-63930000	(28.7MB)	Free	NOACCESS	由内存区
620°£000 62 20000	(4.10MD)	F	NO A COEGG	具有 NOACCESS 权限的自
639ef000-63e20000	(4.19MB)	Free	NOACCESS	由内存区
C2541000 C4200000	(4.20MD)	E.	NOACCEGG	具有 NOACCESS 权限的自
63f41000-64390000	(4.30MB)	Free	NOACCESS	由内存区
C4455000 C4500000	(44.0175)		NO A COESS	具有 NOACCESS 权限的自
644f5000-64500000	(44.0KB)	Free	NOACCESS	由内存区
647 0000 647 0000	(0.6.0127)	T.	NO 4 CCESS	具有 NOACCESS 权限的自
647c8000-647e0000	(96.0KB)	Free	NOACCESS	由内存区
£1020000 £10 0000	(410777)		No + GGTGG	具有 NOACCESS 权限的自
64839000-648a0000	(412KB)	Free	NOACCESS	由内存区

C404 000 C4 70000	(1.14) (D)	Г	NO A CCEGG	具有 NOACCESS 权限的自
6494c000-64a70000	(1.14MB)	B) Free NOACO	NOACCESS	由内存区
64aa4000 64ba0000	(1.10MD)	Erron	NOACCESS	具有 NOACCESS 权限的自
64aa4000-64bc0000	(1.10MB)	Free NOACCESS	由内存区	
64bee000-64bf0000	(8.00KB)	Free	具有 NOACCESS 权限的 由内存区	具有 NOACCESS 权限的自
040ee000-04010000	(8.00KD)	riee		由内存区
64c42000-64de0000	(1.61MB)	Free	NOACCESS	具有 NOACCESS 权限的自
04042000-04060000				由内存区
64e7e000-64e80000	(0,00KD)	Б	NOACCESS	具有 NOACCESS 权限的自
04676000-04680000	(8.00KB)	Free	NOACCESS	由内存区
65052000 fff-0000	(2.42GP)	Free	NOACCESS	具有 NOACCESS 权限的自
65052000-fffe0000	(2.42GB)	Fiee		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
75-0000 75-1000	(4.00KD)	Committed	DEADONIV	具有 READONLY 权限的
7ffe0000-7ffe1000	(4.00KB)	Committed	READONLY	己调配私有内存区
60064000 6006h000	(28 OVD)	Committed	READWRITE	具有 READWRITE 权限
6ae64000-6ae6b000	(28.0KB)	Committed	READWRITE	的已调配私有内存区
			GUARD	具有 GUARD
6b0f9000-6b0fc000	(12.0KB)	Committed	READWRITE	READWRITE权限的已调
				配私有内存区
6b0fc000-6b100000	-6b100000 (16.0KB) Committed READWRITE	READWRITE	具有 READWRITE 权限	
	(16.0KB)	Committee	KEADWKITE	的已调配私有内存区
6b1fc000-6b1ff000			GUARD	具有 GUARD
	(12.0KB)	Committed	READWRITE	READWRITE 权限的已调
			READWRITE	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
			CILADO	具有 GUARD
6b2fc000-6b2ff000	(12.0KB)	Committed	GUARD	READWRITE 权限的已调
			READWRITE	配私有内存区
d 200000 d 200000	(4.0017D)	G 24 1	DE A DIVIDIE	具有 READWRITE 权限
6b2ff000-6b300000	(4.00KB)	Committed	READWRITE	的已调配私有内存区
25620000 25640000	(64 OVD)	Committed	DE A DWDITE	具有 READWRITE 权限
35630000-35640000	(64.0KB)	Committed	READWRITE	的已调配 Mapped 内存区
25640000 25642000	(8 00KD)	Committed	READWRITE	具有 READWRITE 权限
35640000-35642000	(8.00KB)	Committed	READWRITE	的已调配私有内存区
25(50000 25(((000	(00 OVD)	Ci4-1	DEADONIA	具有 READONLY 权限的
35650000-35666000	(88.0KB)	Committed	READONLY	已调配 Mapped 内存区
35670000-35674000	(16 OVD)	Committed	READONLY	具有 READONLY 权限的
330/0000-330/4000	(16.0KB)	Committed	READONLI	已调配 Mapped 内存区
35680000-35681000	(4.00KD)	G iv 1	DEADONIV	具有 READONLY 权限的
33080000-33081000	(4.00KB)	Committed	READONLY	己调配 Mapped 内存区
35690000-35692000	(0,00KD)	G iv 1	READWRITE	具有 READWRITE 权限
33090000-33092000	(8.00KB)	Committed	READWRITE	的已调配私有内存区
356a0000-356a1000	(4.00KB)	Committed	READWRITE	具有 READWRITE 权限
33040000-33041000	(4.00 <b>Kb</b> )	Committee	READWRITE	的已调配私有内存区
356b0000-356b1000	(4.00KB)	Committed	READWRITE	具有 READWRITE 权限
33000000-33001000	(4.00KB)	Committee	READWRITE	的已调配私有内存区
356c0000-356c7000	(28.0KB)	Committed	READWRITE	具有 READWRITE 权限
33000000-33007000	(26.0Kb)	Committee	READWRITE	的已调配私有内存区
25640000 25641000	(4 00KD)	Committed	READWRITE	具有 READWRITE 权限
356d0000-356d1000	(4.00KB)	Committed	READWRITE	的已调配私有内存区
256,0000 256,2000	(8 00KB)	Committed	DEADONIV	具有 READONLY 权限的
356e0000-356e2000	(8.00KB)	Committed	READONLY	已调配映射内存区
254f0000 254f1000	(4 00KD)	Committed	DEADWRITE	具有 READWRITE 权限
356f0000-356f1000	(4.00KB)	Committed	READWRITE	的已调配映射内存区

_			ı	_
35710000-35733000	(140KB)	Committed	nmitted READWRITE	具有 READWRITE 权限
	(110111)	Committee		的已调配私有内存区
35810000-358d1000	(772KB)	Committed	READONLY	具有 READONLY 权限的
33810000-33801000	(772 <b>KB</b> )	Committee	READONLI	已调配映射内存区
250,0000,250,0000	(26 OVD)	Q 'w 1	DEADONIN	具有 READONLY 权限的
358e0000-358e9000	(36.0KB)	Committed	READONLY	已调配映射内存区
25 (0000 25 (5000	(20.01/D)	G1	DEADONIA	具有 READONLY 权限的
35a60000-35a65000	(20.0KB)	Committed	READONLY	已调配映射内存区
25 70000 251 51000	(1.50) (D)	G 20 1	DEADONIA	具有 READONLY 权限的
35a70000-35bf1000	(1.50MB)	Committed	READONLY	已调配映射内存区
25,00000 25,01000	(51 (UD)	G to 1	DE L DONAN	具有 READONLY 权限的
35c00000-35c81000	(516KB)	Committed	READONLY	已调配映射内存区
27000000 2721 1000	(2.72) (D)	G to 1	DE L DONAN	具有 READONLY 权限的
37000000-372bd000	(2.73MB)	Committed	READONLY	已调配映射内存区
7510000 7515000	(= 0 0 )		DE L DOMAY	具有 READONLY 权限的
a7510000-a7515000	(20.0KB)	Committed	READONLY	已调配映射内存区
7610000 7642000	(20.4KD)	C 'w 1	DEADONIA	具有 READONLY 权限的
a7610000-a7643000	(204KB)	Committed	READONLY	已调配映射内存区
-9/20000 -9/21000	(4.00KD)	C:44-4	DEADONIA	具有 READONLY 权限的
a8630000-a8631000	(4.00KB)	Committed	READONLY	已调配映像内存区
			EVECUTE WAIT	具有
a8631000-a8641000	(64.0KB)	Committed	EXECUTE_WRIT	EXECUTE_WRITECOPY
			ECOPY	权限的已调配映像内存区
0641000 0641000	(40 0KD)	C 'w 1	EXECUTE_REA	具有EXECUTE_READ权
a8641000-a864b000	(40.0KB)	Committed	D	限的已调配映像内存区
0.41.000 0.4000	(16 000)	0 20 1	DEADONA	具有 READONLY 权限的
a864b000-a864f000	(16.0KB)	Committed	READONLY	已调配映像内存区
0.640000 0.550000	(4.00177)	G :	READWRITE	具有 READWRITE 权限
a8641000-a8650000	(4.00KB)	Committed		的已调配映像内存区
a8650000-a8655000	(20.0KB)	Committed	READONLY	具有 READONLY 权限的
a864f000-a8650000 a8650000-a8655000	(4.00KB)	Committed  Committed		的已调配映像内存区

				已调配映像内存区
				具有 WRITECOPY 权限
a8655000-a8656000	(4.00KB)	Committed	WRITECOPY	的已调配映像内存区
0.575000 0.570000	(10.0375)	a	DE 1 D 0 1 W 1	具有 READONLY 权限的
a8656000-a8659000	(12.0KB)	Committed	READONLY	已调配映像内存区
120,000,120,100	(4.00775)	a	DE 1 D 0 1 W 1	具有 READONLY 权限的
420c0000-420c1000	(4.00KB)	Committed	READONLY	已调配映像内存区
420c1000-42214000	(1.22MD)	Committed	EXECUTE_REA	具有 EXECUTE_READ 权
42001000-42214000	(1.32MB)	Committed	D	限的已调配映像内存区
42214000-4226e000	(360KB)	Committed	READONLY	具有 READONLY 权限的
42214000-42206000	(300Kb)	Commuted	READONLI	已调配映像内存区
4226e000-42271000	(12.0KB)	Committed	READWRITE	具有 READWRITE 权限
42206000-42271000	(12.0 <b>Kb</b> )	Commuted	READWRITE	的已调配映像内存区
42271000 42282000	(69 OVD)	Committed	READONLY	具有 READONLY 权限的
42271000-42282000	(68.0KB)			已调配映像内存区
42290000-42291000	(4.00KD)	G 244 1	READONLY	具有 READONLY 权限的
42290000-42291000	(4.00KB)	Committed	READONLI	已调配映像内存区
42291000-42329000	(COOKE)	Committed	EXECUTE_REA	具有 EXECUTE_READ 权
42291000-42329000	(608KB)	Committee	D	限的已调配映像内存区
42220000 42274000	(200KD)	C:44-4	READONLY	具有 READONLY 权限的
42329000-42374000	(300KB)	Committed	KEADONLI	已调配映像内存区
42274000 42275000	(4 00KB)	Committed	DEADWDITE	具有 READWRITE 权限
42374000-42375000	(4.00KB)	Commuted	READWRITE	的已调配映像内存区
42275000 42276000	(4 00KB)	Committed	WRITECOPY	具有 WRITECOPY 权限
42375000-42376000	(4.00KB)	Committee	WRITECUPY	的已调配映像内存区
42276000 42278000	(0.00KD)	C:44-4	DE A DWDITE	具有 READWRITE 权限
42376000-42378000	(8.00KB)	Committed	READWRITE	的已调配映像内存区
42279000 42297000	(ECOUD)	C1	DEARCONN	具有 READONLY 权限的
42378000-42386000	(56.0KB)	Committed	READONLY	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
5 121000 5 1 1000	(57.6WD)	C :: 1	EXECUTE_REA	具有EXECUTE_READ权
5a131000-5a1c1000	(576KB)	Committed	D	限的已调配映像内存区
	()			具有 READONLY 权限的
5a1c1000-5a205000	(272KB)	Committed	READONLY	已调配映像内存区
	(1.00777)			具有 READWRITE 权限
5a205000-5a206000	(4.00KB)	Committed	READWRITE	的已调配映像内存区
	(500110)		DE LE COMM	具有 READONLY 权限的
5a206000-5a2b5000	(700KB)	Committed	READONLY	已调配映像内存区
5 140000 5 141000	(4.0017D)	G1	DEADONIA	具有 READONLY 权限的
5cd40000-5cd41000	(4.00KB)	Committed	READONLY	已调配映像内存区
5 141000 5 150000	(0.6 OVD)	G1	EXECUTE_REA	具有EXECUTE_READ权
5cd41000-5cd59000	(96.0KB)	Committed	D	限的已调配映像内存区
5 150000 5 15 000	(20.01/D)	Committed	READONLY	具有 READONLY 权限的
5cd59000-5cd5e000	(20.0KB)			已调配映像内存区
5 15 000 5 15000				具有 READWRITE 权限
5cd5e000-5cd5f000	(4.00KB)	Committed	READWRITE	的已调配映像内存区
5 15000 5 164000	(= 0 0 )	a	DEADONIV	具有 READONLY 权限的
5cd5f000-5cd64000	(20.0KB)	Committed	READONLY	已调配映像内存区
500,0000, 500,01000	(4.0017D)	G1	DEADONIA	具有 READONLY 权限的
5f960000-5f961000	(4.00KB)	Committed	READONLY	已调配映像内存区
5 m < 1000 5 m 0 m 0 0	(0.401/D)	G1	EXECUTE_REA	具有EXECUTE_READ权
5f961000-5f99f000	(248KB)	Committed	D	限的已调配映像内存区
5,000,000,5,001,000	(11 CVD)	G ''' 1	DEADONIA	具有 READONLY 权限的
5f99f000-5f9bc000	(116KB)	Committed	READONLY	已调配映像内存区
5f0L-000 5f0L-000	(0 00KB)	G- 111 1	DE 4 DWD IEE	具有 READWRITE 权限
5f9bc000-5f9be000	(8.00KB)	Committed	READWRITE	的已调配映像内存区
501 000 500 1 000	(11000)	G 11		具有 READONLY 权限的
5f9be000-5f9da000	(112KB)	Committed	READONLY	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
	(2 ¢ 0¥FD)		EXECUTE_REA	具有EXECUTE_READ权
61381000-6138a000	(36.0KB)	Committed	D	限的已调配映像内存区
6120 000 61207000	(52 OVD)	G to 1	DEADONAN	具有 READONLY 权限的
6138a000-61397000	(52.0KB)	Committed	READONLY	已调配映像内存区
	(4.00717)			具有 READWRITE 权限
61397000-61398000	(4.00KB)	Committed	READWRITE	的已调配映像内存区
61200000 6120 000	(24.01/P)	G to 1	DEADONAN	具有 READONLY 权限的
61398000-6139e000	(24.0KB)	Committed	READONLY	已调配映像内存区
c1.450000 c1.451000	(4.0017D)	G1	DEADONIA	具有 READONLY 权限的
61450000-61451000	(4.00KB)	Committed	READONLY	已调配映像内存区
C1451000 C14 1000	(220KD)	G1	EXECUTE_REA	具有EXECUTE_READ权
61451000-614a1000	(320KB)	Committed	D	限的已调配映像内存区
C14 1000 C141 4000	(76.0KB)	Committed	READONLY	具有 READONLY 权限的
614a1000-614b4000				已调配映像内存区
C141-4000 C141-5000	(4.00775)	a		具有 READWRITE 权限
614b4000-614b5000	(4.00KB)	Committed	READWRITE	的已调配映像内存区
C1415000 C141-000	(20 01/D)	G in I	READONLY	具有 READONLY 权限的
614b5000-614ba000	(20.0KB)	Committed	READONLY	已调配映像内存区
c1c00000 c1c01000	(4.00KD)	C:44-4	READONLY	具有 READONLY 权限的
61690000-61691000	(4.00KB)	Committed	KEADONLI	已调配映像内存区
61691000-61761000	(832KB)	Committed	EXECUTE_REA	具有 EXECUTE_READ 权
01091000-01701000	(632 <b>N</b> D)	Commuted	D	限的已调配映像内存区
61761000-6187e000	(1.11MD)	Committed	DE A DONI V	具有 READONLY 权限的
61/61000-618/6000	(1.11MB)	Committed	READONLY	已调配映像内存区
6187e000-61882000	(16 OVD)	Committed	READWRITE	具有 READWRITE 权限
018/6000-01882000	(16.0KB)	Committed	READWRITE	的已调配映像内存区
Z1002000 Z1002000	(4 00KD)	Committee 1	WRITECOPY	具有 WRITECOPY 权限
61882000-61883000	(4.00KB)	Committed		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
				具有 READONLY 权限的
618b0000-618b1000	(4.00KB)	Committed	READONLY	已调配映像内存区
618b1000-61923000	(456VD)	Committed	EXECUTE_REA	具有EXECUTE_READ权
61801000-61923000	(456KB)	Committed	D	限的已调配映像内存区
(1022000 (1047000	(1.4.4KD)	G1	DEADONIA	具有 READONLY 权限的
61923000-61947000	(144KB)	Committed	READONLY	已调配映像内存区
610.47000 610.40000	(4.0017D)	G1	DE A DIVIDITE	具有 READWRITE 权限
61947000-61948000	(4.00KB)	Committed	READWRITE	的已调配映像内存区
< 10.40000 < 10.40000	(4.0017D)	G1	WIDITECODY	具有 WRITECOPY 权限
61948000-61949000	(4.00KB)	Committed	WRITECOPY	的已调配映像内存区
610 40000 610 <b>52</b> 000	(40.01/D)	G to 1	DEADONAN	具有 READONLY 权限的
61949000-61953000	(40.0KB)	Committed	READONLY	已调配映像内存区
	(4.00KB)	Committed	WRITECOPY	具有 WRITECOPY 权限
61953000-61954000				的已调配映像内存区
(1054000 (1050000	(20.0KB)	Committed	READONLY	具有 READONLY 权限的
61954000-61959000				已调配映像内存区
(10,0000,010,01000	(4.0017D)		DE A DOMEN	具有 READONLY 权限的
61960000-61961000	(4.00KB)	Committed	READONLY	已调配映像内存区
C10C1000 C1 1000	(7.60VD)	G1	EXECUTE_REA	具有EXECUTE_READ权
61961000-61a1f000	(760KB)	Committed	D	限的已调配映像内存区
C1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	(CAAVD)	G '' 1	DEADONIA	具有 READONLY 权限的
61a1f000-61ac0000	(644KB)	Committed	READONLY	已调配映像内存区
C1 0000 C1 4000	(1.6.017D)	G1	DE A DIVIDITE	具有 READWRITE 权限
61ac0000-61ac4000	(16.0KB)	Committed	READWRITE	的已调配映像内存区
C1 4000 C1 0000	(1101/2)	Q 11	DEADONAN	具有 READONLY 权限的
61ac4000-61ae0000	(112KB)	Committed	READONLY	已调配映像内存区
<b>61</b> 0000 61 1005	(4.00777)	G	READONLY	具有 READONLY 权限的
61ae0000-61ae1000	(4.00KB)	Committed		已调配映像内存区
61ae1000-61b32000	(324KB)	Committed	EXECUTE_REA	具有EXECUTE_READ权

			D	限的已调配映像内存区
411 22 C C C C C C C C C C C C C C C C C	(252KB)	Committed		具有 READONLY 权限的
61b32000-61b71000			READONLY	已调配映像内存区
	(4.00317)			具有 WRITECOPY 权限
61b71000-61b72000	(4.00KB)	Committed	WRITECOPY	的已调配映像内存区
	(10.0117)			具有 READWRITE 权限
61b72000-61b75000	(12.0KB)	Committed	READWRITE	的已调配映像内存区
(11.75000 (11.7.000	(20.01/D)		DEADONIA	具有 READONLY 权限的
61b75000-61b7c000	(28.0KB)	Committed	READONLY	已调配映像内存区
<11 00000 <11 01000	(4.00KD)	G 'w 1	DEADONIA	具有 READONLY 权限的
61b80000-61b81000	(4.00KB)	Committed	READONLY	已调配映像内存区
C11-01000 C1-2-000	(C94VD)		EXECUTE_REA	具有 EXECUTE_READ 权
61b81000-61c2c000	(684KB)	Committed	D	限的已调配映像内存区
61c2c000-61c65000	(228VD)	Committed	READONLY	具有 READONLY 权限的
6102000-61065000	(228KB)	Committed		已调配映像内存区
61c65000-61c68000	(12 OKB)	Committed RE	READWRITE	具有 READWRITE 权限
01003000-01008000	(12.0KB)	Committee	KE/ID WRITE	的已调配映像内存区
61c68000-61c75000	(52.0KB)	Committed	READONLY	具有 READONLY 权限的
01008000-01073000	(32.0 <b>Kb</b> )	Committee	KEADONLI	己调配映像内存区
63930000-63931000	(4 00KB)	Committed	READONLY	具有 READONLY 权限的
03930000-03931000	(4.00KB)	Committee		已调配映像内存区
63931000-639b9000	(544KB)	Committed	EXECUTE_REA	具有 EXECUTE_READ 权
	(344KD)	Committee	D	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
	(12.0KB)			的已调配映像内存区
639e1000-639ef000	(56.0KB)	Committed	READONLY	具有 READONLY 权限的
03701000-03761000	(50.0KD)	Commuted	KLADONLI	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_REA	具有EXECUTE_READ权
			D	限的已调配映像内存区
	(4.10777)			具有 READONLY 权限的
63f00000-63f2a000	(168KB)	Committed	READONLY	已调配映像内存区
	(0,00717)			具有 READWRITE 权限
63f2a000-63f2c000	(8.00KB)	Committed	READWRITE	的已调配映像内存区
6252 000 62541000	(0.4.0IVD)	G1	DEADONIA	具有 READONLY 权限的
63f2c000-63f41000	(84.0KB)	Committed	READONLY	已调配映像内存区
C4200000 C4201000	(4.00KD)	G 24 1	DEADONIA	具有 READONLY 权限的
64390000-64391000	(4.00KB)	Committed	READONLY	已调配映像内存区
64391000-6442e000	(C20VD)		EXECUTE_REA	具有 EXECUTE_READ 权
64391000-6442e000	(628KB)	Committed	D	限的已调配映像内存区
6442-000-64445000	(116VD)	Committed	READONLY	具有 READONLY 权限的
6442e000-6444b000	(116KB)	Committed		已调配映像内存区
6444b000-6444d000	(8 00KD)	Committed READWRITE	DE A DWDITE	具有 READWRITE 权限
04440000-04440000	(8.00KB)		的已调配映像内存区	
6444d000-644f5000	(672KB)	Committed	READONLY	具有 READONLY 权限的
0444000-04413000	(0/2 <b>ND</b> )	Committee	KEADONLI	己调配映像内存区
64500000-64501000	(4.00KD)	Committed	READONLY	具有 READONLY 权限的
	(4.00KB)	Committee		已调配映像内存区
64501000-646c9000	(1.78MB)	Committed	EXECUTE_REA	具有 EXECUTE_READ 权
	(1.76WID)	Committee	D	限的已调配映像内存区
646c9000-6477e000	(724KB)	Committed	READONLY	具有 READONLY 权限的
	(724KB)			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
	(4.00KD)	Committee	WRITECOLL	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
	(4.00KB)			具有 WRITECOPY 权限
64783000-64784000		Committed	WRITECOPY	的已调配映像内存区
	(050110)		PE1 PO1 WY	具有 READONLY 权限的
64784000-647c8000	(272KB)	Committed	READONLY	已调配映像内存区
647,0000,647,1000	(4.001/P)		具有 READONLY 权限的	
647e0000-647e1000	(4.00KB)	Committed	READONLY	已调配映像内存区
647e1000-64812000	(10.CVD)	C:44-4	EXECUTE_REA	具有 EXECUTE_READ 权
64761000-64812000	(196KB)	Committed	D	限的已调配映像内存区
64912000 64920000	(120KB)		DEADONIV	具有 READONLY 权限的
64812000-64830000	(120KB)	Committed	READONLY	已调配映像内存区
64920000 64921000	(4.00KB)	G in 1		具有 READWRITE 权限
64830000-64831000	(4.00KB)	Committed	READWRITE	的已调配映像内存区
64921000 64922000	(4.00KB)	Committed	WRITECOPY	具有 WRITECOPY 权限
64831000-64832000				的已调配映像内存区
C4922000 C4922000	(4.00KD)	G in I DEAL	READWRITE	具有 READWRITE 权限
64832000-64833000	(4.00KB)	Committed	Committee READWRITE	的已调配映像内存区
C4922000 C4920000	(24 OKD)	C:44-4	DEADONIV	具有 READONLY 权限的
64833000-64839000	(24.0KB)	Committed	READONLY	已调配映像内存区
648a0000-648a1000	(4.00KD)	Committed	READONLY	具有 READONLY 权限的
04880000-04881000	(4.00KB)	Committed		已调配映像内存区
648a1000-64914000	(460VD)	Committed	EXECUTE_REA	具有 EXECUTE_READ 权
04881000-04914000	(460KB)	Committed	D	限的已调配映像内存区
64014000 64042000	(100VD)	Committed	READONLY	具有 READONLY 权限的
64914000-64943000	(188KB)			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
C4044000 C404 000	(2.2	Committee 1	DEV DOMI A	具有 READONLY 权限的
64944000-6494c000	(32.0KB)	Committed	READONLY	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_REA	具有EXECUTE_READ权
			D	限的已调配映像内存区
	(100777)			具有 READONLY 权限的
64a83000-64a9e000	(108KB)	Committed	READONLY	已调配映像内存区
	(4.00775)	Committed READWRITE	具有 READWRITE 权限	
64a9e000-64a9f000	(4.00KB)		READWRITE	的已调配映像内存区
64 0000 64 4000	(20.01/D)	G1	DEADONIA	具有 READONLY 权限的
64a9f000-64aa4000	(20.0KB)	Committed	READONLY	已调配映像内存区
C41-2000 C41-1000	(4.00KD)	C:44-4	DEADONIA	具有 READONLY 权限的
64bc0000-64bc1000	(4.00KB)	Committed	READONLY	已调配映像内存区
C41-1000 C41-11000	(112KD)		EXECUTE_REA	具有 EXECUTE_READ 权
64bc1000-64bdd000	(112KB)	Committed	D	限的已调配映像内存区
C41-1-1000 C41-4000	(= 0 0777)	Committed	READONLY	具有 READONLY 权限的
64bdd000-64be4000	(28.0KB)			已调配映像内存区
C41- 4000 C41- 5000	(4.00KD)	Committed READWRITE		具有 READWRITE 权限
64be4000-64be5000	(4.00KB)		的已调配映像内存区	
C41-5000 C41000	(2.6.0VD)	Committed	DE A DONI V	具有 READONLY 权限的
64be5000-64bee000	(36.0KB)	Committed	READONLY	已调配映像内存区
64bf0000-64bf1000	(4.00KD)	Committed	READONLY	具有 READONLY 权限的
04010000-04011000	(4.00KB)			已调配映像内存区
64bf1000-64c1b000	(169VD)	Committed	EXECUTE_REA	具有 EXECUTE_READ 权
04011000-04010000	(168KB)	Committed	D	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
(4-2-000 (4-42000	(24.277	Committed	READONLY	具有 READONLY 权限的
64c3c000-64c42000	(24.0KB)	Committed	KEADUNLY	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_REA	具有EXECUTE_READ权
			D	限的已调配映像内存区
64.56000 64.60000	(10017D)		DEADONAN	具有 READONLY 权限的
64e56000-64e6f000	(100KB)	Committed	READONLY	已调配映像内存区
	(0.00417)	Committed		具有 READWRITE 权限
64e6f000-64e71000	(8.00KB)		READWRITE	的已调配映像内存区
	(10.0375)		WID WEEL GODY	具有 WRITECOPY 权限
64e71000-64e74000	(12.0KB)	Committed	WRITECOPY	的已调配映像内存区
C1 71000 C1 7C000	(0,00MD)	G 1	DE A DIVIDITE	具有 READWRITE 权限
64e74000-64e76000	(8.00KB)	Committed	READWRITE	的已调配映像内存区
	(1.00777)	~		具有 WRITECOPY 权限
64e76000-64e77000	(4.00KB)	Committed	WRITECOPY	的已调配映像内存区
	(28.0KB)	Committed		具有 READONLY 权限的
64e77000-64e7e000			READONLY	已调配映像内存区
		Committed READONLY	具有 READONLY 权限的	
64e80000-64e81000	(4.00KB)		READONLY	已调配映像内存区
			EXECUTE_REA	具有EXECUTE_READ权
64e81000-64f89000	(1.03MB)	Committed	D	限的已调配映像内存区
			DE L DOMAN	具有 READONLY 权限的
64f89000-64fcd000	(272KB)	Committed	READONLY	已调配映像内存区
	(4.00KB)	Committed	READWRITE	具有 READWRITE 权限
64fcd000-64fce000				的已调配映像内存区
	(0.00117)	Committed	WID WELL GODY	具有 WRITECOPY 权限
64fce000-64fd0000	(8.00KB)		WRITECOPY	的已调配映像内存区
	(2.1.277		READWRITE	具有 READWRITE 权限
64fd0000-64fd6000	(24.0KB)	Committed		的已调配映像内存区
	(496KB) Committed		READONLY	具有 READONLY 权限的
64fd6000-65052000		Committed		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
7ff-1000 7fff0000	(60 OMB)	(60.0KB) Reserved READONLY	PE - PO - WW	具有 READONLY 权限的保
7ffe1000-7fff0000	(60.0KB)		READONLY	留私有内存区
6 00000 6 64000		Reserved	READONLY	具有 READONLY 权限的保
6ae00000-6ae64000	(400KB)			留私有内存区
6 61 000 61 000000	(1.50MD)	Reserved	READONLY	具有 READONLY 权限的保
6ae6b000-6b000000	(1.58MB)			留私有内存区
d 000000 d 00000	(00 (VP)		DEADONIA	具有 READONLY 权限的保
6b000000-6b0f9000	(996KB)	Reserved	READONLY	留私有内存区
Ch 100000 Ch 15-000	(0.00MD)	D 1	DEADONLY	具有 READONLY 权限的保
6b100000-6b1fc000	(0.98MB)	Reserved	READONLY	留私有内存区
(1.200000 (1.25-000	(0.98MB)	Reserved	READONLY	具有 READONLY 权限的保
6b200000-6b2fc000				留私有内存区
25.42000 25.44000	(44.0KB)	Reserved	READONLY	具有 READONLY 权限的保
35642000-3564d000				留私有内存区
256 1000 256 1000	(40 OKD)	D 1	DEADONIV	具有 READONLY 权限的保
356a1000-356ad000	(48.0KB)	Reserved	READONLY	留私有内存区
256 7000 256 10000	(26 0KD)	ъ 1		具有 READONLY 权限的保
356c7000-356d0000	(36.0KB)	Reserved	READONLY	留私有内存区
25722000 25010000	(00 AKD)	ъ 1	DEADONIN	具有 READONLY 权限的保
35733000-35810000	(884KB)	Reserved	READONLY	留私有内存区
259-0000 25 (0000	(1.46MB)	Reserved	READONLY	具有 READONLY 权限的保
358e9000-35a60000				留映射内存区
	(12.0KB)	Reserved	READONLY	具有 READONLY 权限的保
35a65000-35a68000				留映射内存区
25 01000 2500000	(10.47.77)		DE LOCUME	具有 READONLY 权限的保
35c81000-37000000	5c81000-37000000 (19.4MB) Reserved R	READONLY	留映射内存区	

a7515000-a7610000	(0.00 <b>MD</b> )	D 1		具有 READONLY 权限的保
	(0.98MB)	Reserved	READONLY	留映射内存区
fffe0000-ffff0000	(CA OVD)	Deserved NOACCES	NO A CCECC	具有 NOACCESS 权限的保
	(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。

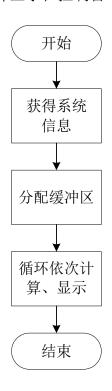


图 3-3 Lab3\_2 流程图

#### 2. 虚拟内存的分配与释放

能正确使用系统函数 GetMeoryStatus()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 VirsualAlloc()函数和 VirsualFree()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 0SLab 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?):

# <u>Current Memory Status is :</u>

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 Virsual memory is 134217712 MB

Memory Load is 50 %

实验结果如图 3-4 所示。

```
The entry point of the compute approaches.

Total Physical Memory is 4094 MB
Total Page File is 5950 MB
Available Page File is 1980 MB
Total Virtual Memory is 194217727 MB
Available Virsual memory is 194217713 MB

Now Allocate 32M Virsual Memory is 2044 MB
Total Physical Memory is 194217727 MB
Available Virsual memory is 194217727 MB

Now Release 32M Virsual Memory is 2044 MB
Total Page File is 5950 MB
Available Virsual memory is 194217727 MB

Now Release 32M Virsual Memory is 194217727 MB

Now Release 32M Virsual Memory is 194217727 MB
Memory Load is 50 %

No Total Physical Memory is 194217727 MB
Available Page File is 5950 MB
Available Physical Memory is 2046 MB
Total Virtual Memory is 194217727 MB
Available Virsual Memory is 2046 MB
Total Physical Memory is 194217727 MB
Available Virsual memory is 134217712 MB
Memory Load is 50 %

C:\Users\Administrator\source\repos\OSLab\x64\Debug\Lab3_3.exe (ibta 8440) 已退出,返回代码为: 0.

按任意键关闭此面口...
```

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:

<u>分配虚拟内存: BaseAddr = ::VirtualAlloc(NULL, 1024 \* 1024 \* 32, MEM RESERVE</u>

| 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。

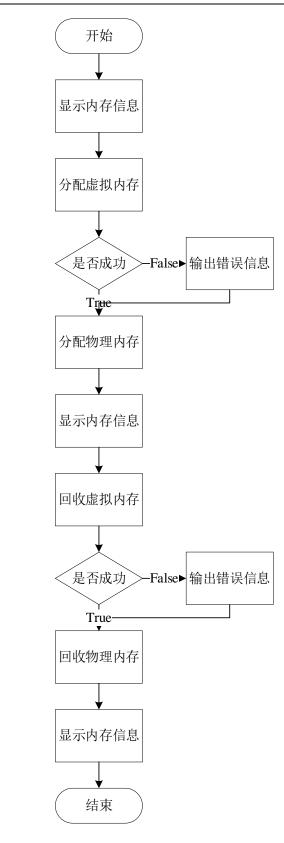


图 3-5 Lab3\_3 流程图

# 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

```
// 工程 vmeminfo
2. #include <windows.h>
3.
   #include <iostream>
4. using namespace std;
5.
   #include <shlwapi.h>
6. #include <iomanip>
    #pragma comment(lib, "shlwapi.lib")
7.
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
      TCHAR szPageSize[MAX_PATH];
16.
17.
      ::StrFormatByteSize(si.dwPageSize, szPageSize, MAX_PATH);
18.
      TCHAR dwAllocationGranularity[MAX_PATH];
      ::StrFormatByteSize(si.dwAllocationGranularity, dwAllocationGranularity, MAX_PATH);
19.
      DWORD dwMemSize = (DWORD)si.lpMaximumApplicationAddress - (DWORD)si.lpMinimumApplication
20.
    Address + 1;
      TCHAR szMemSize[MAX_PATH];
21.
      ::StrFormatByteSize(dwMemSize, szMemSize, MAX_PATH);
22.
23.
24.
      // 将内存信息显示出来
25.
      cout << "Virtual memory page size: " << szPageSize << endl;</pre>
      cout << "虚拟内存粒度: " << dwAllocationGranularity << endl;
26.
27.
28.
      cout << "Minimum application address: 0x" << hex << setw(8)</pre>
29.
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;</pre>
   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>
   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) ) \
           {cout << ", " << #type;}
   19.
   20.
   21. void ShowProtection(DWORD dwTarget)
   22. {
           SHOWMASK(dwTarget, READONLY);
   23.
   24.
          SHOWMASK(dwTarget, GUARD);
           SHOWMASK(dwTarget, NOCACHE);
   25.
           SHOWMASK(dwTarget, READWRITE);
   26.
   27.
           SHOWMASK(dwTarget, WRITECOPY);
          SHOWMASK(dwTarget, EXECUTE);
   28.
           SHOWMASK(dwTarget, EXECUTE_READ);
   29.
           SHOWMASK(dwTarget, EXECUTE_READWRITE);
   30.
           SHOWMASK(dwTarget, EXECUTE WRITECOPY);
   31.
           SHOWMASK(dwTarget, NOACCESS);
   32.
   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) {
           // 首先,获得系统信息
   37.
   38.
          SYSTEM INFO si;
   39.
           ::ZeroMemory(&si, sizeof(si));
```

```
40.
        ::GetSystemInfo(&si);
41.
        // 分配要存放信息的缓冲区
42.
43.
        MEMORY_BASIC_INFORMATION mbi;
44.
        ::ZeroMemory(&mbi, sizeof(mbi));
45.
        // 循环整个应用程序地址空间
46.
        LPCVOID pBlock = (LPVOID)si.lpMinimumApplicationAddress;
47.
        while (pBlock < si.lpMaximumApplicationAddress)</pre>
48.
49.
            // 获得下一个虚拟内存块的信息
50.
51.
            if (::VirtualQueryEx(
52.
                hProcess, // 相关的进程
                pBlock, // 开始位置
53.
                &mbi, // 缓冲区
54.
                sizeof(mbi)) == sizeof(mbi)) // 大小的确认
55.
56.
                // 计算块的结尾及其大小
57.
                LPCVOID pEnd = (PBYTE)pBlock + mbi.RegionSize;
58.
                TCHAR szSize[MAX_PATH];
59.
                ::StrFormatByteSize(mbi.RegionSize, szSize, MAX_PATH);
60.
61.
62.
                // 显示块地址和大小
63.
                cout.fill('0');
                cout << hex << setw(8) << (DWORD)pBlock</pre>
64.
                    << "-" << hex << setw(8) << (DWORD)pEnd
65.
                    //<< " (" << szSize
66.
                    << (::strlen(szSize) == 7 ? " (" : " ( ") << szSize
67.
68.
                    << ") ";
69.
70.
                // 显示块的状态
71.
                switch (mbi.State) {
72.
                case MEM COMMIT:
73.
                    cout << "Committed";</pre>
74.
                    break;
75.
                case MEM_FREE:
76.
                    cout << "Free";</pre>
77.
                    break;
78.
                case MEM_RESERVE:
79.
                    cout << "Reserved";</pre>
80.
                    break;
81.
                }
82.
                // 显示保护
83.
                if (mbi.Protect == 0 && mbi.State != MEM FREE)
84.
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";</pre>
   94.
                       break;
                   case MEM_MAPPED:
   95.
                       cout << ", Mapped";</pre>
   96.
   97.
                       break;
   98.
                   case MEM_PRIVATE:
                       cout << ", Private";</pre>
   99.
   100.
                       break;
   101.
                   }
   102.
                   // 检验可执行的影像
   103.
   104.
                   TCHAR szFilename[MAX_PATH];
                   if (::GetModuleFileName(
   105.
   106.
                       (HMODULE)pBlock, // 实际虚拟内存的模块句柄
                       szFilename, // 完全指定的文件名称
   107.
   108.
                      MAX_PATH) > 0) // 实际使用的缓冲区大小
   109.
                   {
                      // 除去路径并显示
   110.
                       ::PathStripPath(szFilename);
   111.
                       cout << ", Module: " << szFilename;</pre>
   112.
   113.
   114.
                   cout << endl;// 移动块指针以获得下一下个块
   115.
                   pBlock = pEnd;
   116.
           }
   117.
           }
   118.}
   119.
   120.void main() {
           // 遍历当前进程的虚拟内存
   121.
   122.
         ::WalkVM(::GetCurrentProcess());
   123.}
清单 3-3
      #include <Windows.h>
   2. #include <iostream>
   3. using namespace std;
```

4. // Defines the entry point for the console application.

5.

```
#ifdef_DEBUG
6.
7.
    #define new DEBUG_NEW
   #undef THIS_FILE
8.
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. {
      int nRetCode = 0;
17.
18.
      LPVOID BaseAddr;
       char* str;
19.
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");
      str = (char*)malloc(1024 * 1024 * 2); //分配内存
28.
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. {
      MEMORYSTATUS MemInfo;
44.
45.
       GlobalMemoryStatus(&MemInfo);
46.
       printf("Current Memory Status is :\n");
47.
      printf("\t Total Physical Memory is %d MB\n", MemInfo.dwTotalPhys / (1024 * 1024));
48.
       printf("\t Available Physical Memory is %d MB\n", MemInfo.dwAvailPhys / (1024 * 1024));
49.
50.
       printf("\t Total Page File is %d MB\n", MemInfo.dwTotalPageFile / (1024 * 1024));
```

```
printf("\t Available\ Page\ File\ is\ \%d\ MB\n",\ MemInfo.dw Avail Page File\ /\ (1024*1024));
51.
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));
       printf("\t Memory\ Load\ is\ \%d\ \%\%\n\n",\ MemInfo.dwMemoryLoad);
54.
55. }
```