

# 实验三：UNIX V6++完整的进程图象

## 一、实验目的

结合课程所学知识，通过编写一个简单的 C++代码，并在 UNIX V6++中编译和运行调试。通过查找关键地址单元的值，绘制出整个进程的图象，进而加深对 UNIX 进程图象的理解，特别是对逻辑地址空间与物理地址空间的理解。

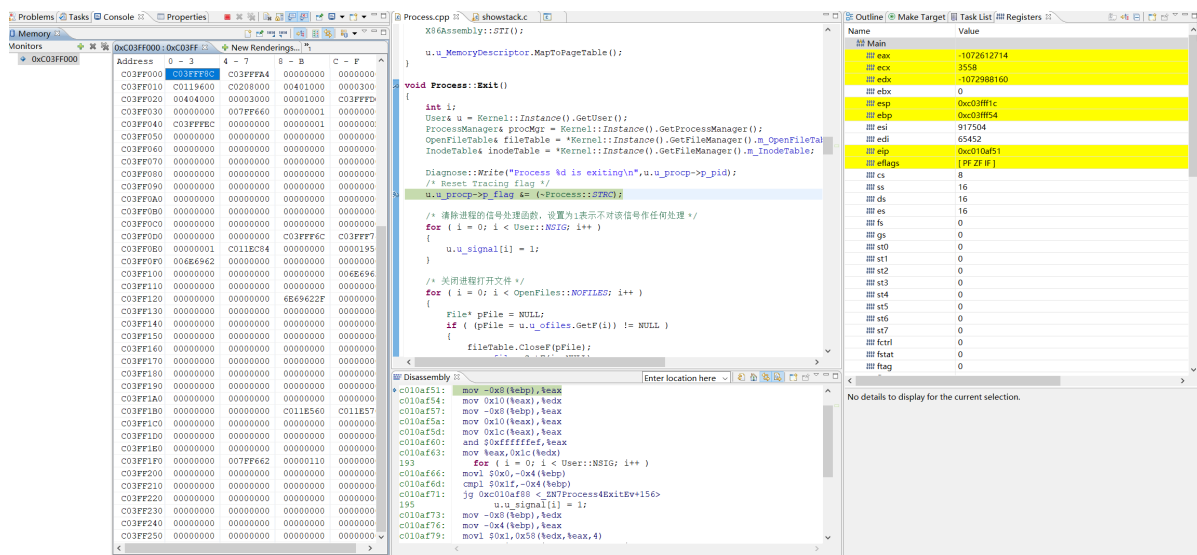
## 二、实验内容

### 1.获取进程USER结构，Proc结构和Text结构

#### 获取USER结构

以调试模式启动UNIX V6++，将项目的调试对象设置为Kernel.exe，并在在Process::Exit()函数中设置断点，如图所示，调试正常开始，在UNIX V6++中输入指令

cd bin 和showstack.exe，程序执行完输出语句后停在断点处。



```
1 public:
2 static const unsigned long USER_ADDRESS = 0x400000 - 0x1000 + 0xc0000000; /*
  0xc03ff000 */
3 User& Kernel::GetUser()
4 {
5 return *(User*)USER_ADDRESS;
6 }
```

由kernel中提供的GetUser函数设置User结构的逻辑地址可知，UNIX V6++的进程 User 结构逻辑地址是固定的，始终为 0xC03FF000 于是我们打开memory窗口，可以依次查看User结构中各变量的值

Memory					
Monitors					
0xC03FF000 : 0xC03FF					
New Renderings...					
»1					
Address	0 - 3	4 - 7	8 - B	C - F	
0xC03FF000					
0x0040F000					
0xC0119600					
0xC011AE94					
C03FEFD0	00000000	00000000	00000000	00000000	
C03FEFE0	00000000	00000000	00000000	00000000	
C03FEFF0	00000000	00000000	00000000	00000000	
C03FF000	C03FFF8C	C03FFFA4	00000000	00000000	
C03FF010	C0119600	C0208000	00401000	00003000	
C03FF020	00404000	00003000	00001000	C03FFFD0	
C03FF030	00000000	007FF660	00000001	00000000	
C03FF040	C03FFFE0	00000000	00000001	00000000	
C03FF050	00000000	00000000	00000000	00000000	
C03FF060	00000000	00000000	00000000	00000000	
C03FF070	00000000	00000000	00000000	00000000	
C03FF080	00000000	00000000	00000000	00000000	
C03FF090	00000000	00000000	00000000	00000000	
C03FF0A0	00000000	00000000	00000000	00000000	
C03FF0B0	00000000	00000000	00000000	00000000	
C03FF0C0	00000000	00000000	00000000	00000000	
C03FF0D0	00000000	00000000	C03FFF6C	C03FFF70	
C03FF0E0	00000001	C011EC84	00000000	00001950	
C03FF0F0	006E6962	00000000	00000000	00000000	
C03FF100	00000000	00000000	00000000	006E6960	
C03FF110	00000000	00000000	00000000	00000000	
C03FF120	00000000	00000000	6E69622F	00000000	
C03FF130	00000000	00000000	00000000	00000000	
C03FF140	00000000	00000000	00000000	00000000	
C03FF150	00000000	00000000	00000000	00000000	
C03FF160	00000000	00000000	00000000	00000000	
C03FF170	00000000	00000000	00000000	00000000	
C03FF180	00000000	00000000	00000000	00000000	
C03FF190	00000000	00000000	00000000	00000000	
C03FF1A0	00000000	00000000	00000000	00000000	
C03FF1B0	00000000	00000000	C011E560	C011E570	
C03FF1C0	00000000	00000000	00000000	00000000	
C03FF1D0	00000000	00000000	00000000	00000000	
C03FF1E0	00000000	00000000	00000000	00000000	
C03FF1F0	00000000	007FF662	00000110	00000000	

Process\* u\_procp = 0xC0119600

MemoryDescriptor u\_MemoryDescriptor

PageTable\*m\_UserPageTableArray=0xC0208000 相对映射表首地址

unsigned long m\_TextStartAddress =0x00401000=4M+4K 代码段起始地址

unsigned long m\_TextSize 代码段长度=0x00003000=12K

unsigned long m\_DataStartAddress =0x00404000=4M+16K 数据段起始地址

unsigned long m\_DataSize =0x00003000=12K 数据段长度

unsigned long m\_StackSize =0x00001000=4K 栈段长度

## 获取proc结构

由Process\* u\_procp = 0xC0119600，可以得到进程proc的逻辑地址，同样查看memory窗口可以得到proc结构数据成员的值，而p\_addr中的值是物理地址，所以其为User结构的物理地址

Monitors					
0xC0119600 <Hex>					
0xC0119600 : 0xC0119619					
»1					
◆ 0xC03FF000	Address	0 - 3	4 - 7	8 - B	C - F
◆ 0x0040F000	C0119600	00000000	00000002	00000001	0040F000
◆ 0xC0119600	C0119610	00005000	C011AE94	00000003	00000000
	C0119620	00000065	0000001C	00000000	00000000
	C0119630	00000000	00000000	C0120DA0	00000000
	C0119640	00000000	00000000	FFFFFFFF	00000000
	C0119650	00000000	00000000	00000000	00000000
	C0119660	00000000	00000000	00000000	00000000
	C0119670	00000000	00000000	00000000	00000000
	C0119680	00000000	00000000	FFFFFFFF	00000000
	C0119690	00000000	00000000	00000000	00000000
	C01196A0	00000000	00000000	00000000	00000000
	C01196B0	00000000	00000000	00000000	00000000
	C01196C0	00000000	00000000	FFFFFFFF	00000000
	C01196D0	00000000	00000000	00000000	00000000
	C01196E0	00000000	00000000	00000000	00000000
	C01196F0	00000000	00000000	00000000	00000000
	C0119700	00000000	00000000	FFFFFFFF	00000000
	C0119710	00000000	00000000	00000000	00000000
	C0119720	00000000	00000000	00000000	00000000
	C0119730	00000000	00000000	00000000	00000000
	C0119740	00000000	00000000	FFFFFFFF	00000000
	C0119750	00000000	00000000	00000000	00000000
	C0119760	00000000	00000000	00000000	00000000
	C0119770	00000000	00000000	00000000	00000000

用户ID short p\_uid 0

进程标识数int p\_pid 2

父进程标识数int p\_ppid 1

user 结构即ppda 区的物理地址unsigned long p\_addr 0x0040F000

除共享正文段的长度，以字节单位unsigned int p\_size 0x00005000=20K

指向代码段Text 结构的逻辑地址

进程调度状态ProcessState p\_stat 3=SRUN

进程标志位int p\_flag 1=SLOAD

进程优先数int p\_pri 65

cpu 值，用于计算int p\_cpu p\_pri 19

进程优先数微调参数int p\_nice 0

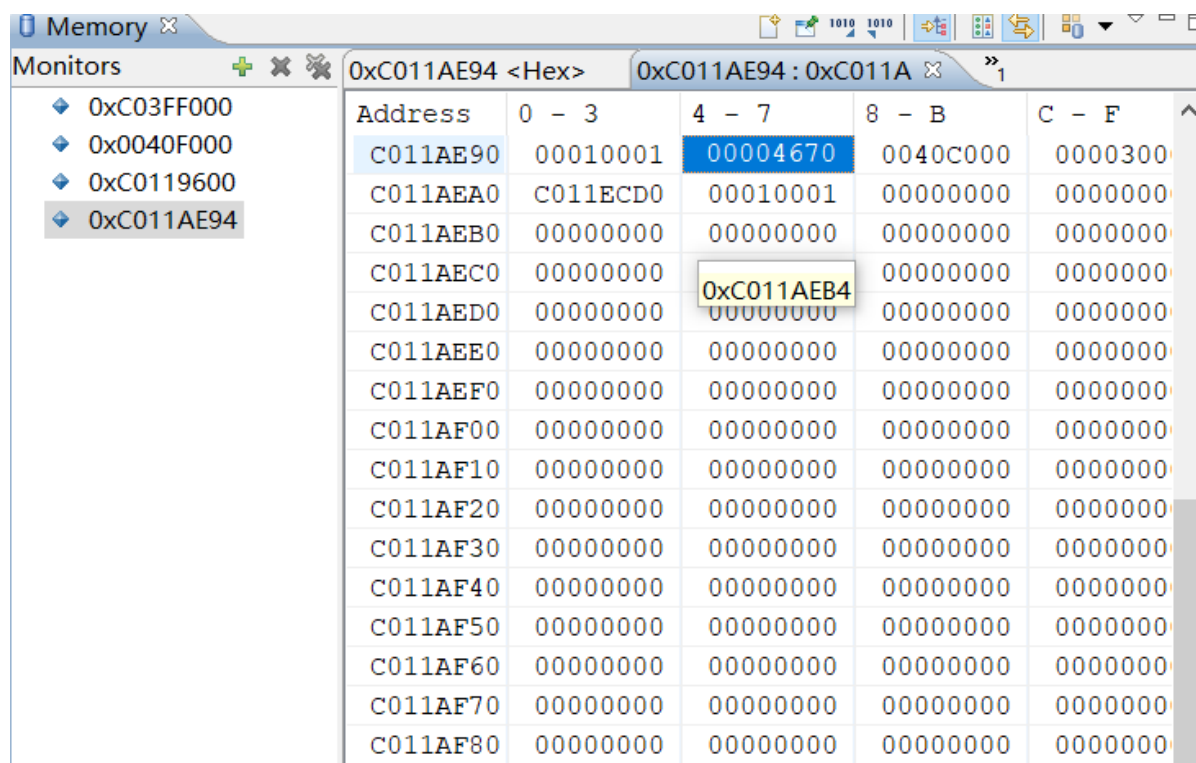
进程在盘上(内存内)驻留时间int p\_time 0

进程睡眠原因unsigned long p\_wchan 0

## 获取Text结构

由Text\* p\_textp= 0xC011AE94可以得到代码段的逻辑地址，同样通过memory窗口可以得到各数据成员的值。如下图

代码段在盘交换区上的地址 `int x_daddr 0x00004670`  
 代码段起始地址（物理地址） `unsigned long x_caddr 0x0040C000`  
 代码段长度，以字节为单位 `unsigned int x_size 0x00003000 = 12K`  
 内存inode 地址 `Inode* x_iptr 0xC011ECD0`  
 共享正文段的进程数 `Unsigned short x_count 1`  
 共享该正文段且图像在内存的进程数 `Unsigned short x_ccount 1`



Address	0 - 3	4 - 7	8 - B	C - F
C011AE90	00010001	00004670	0040C000	00003000
C011AEA0	C011ECD0	00010001	00000000	00000000
C011AEB0	00000000	00000000	00000000	00000000
C011AEC0	00000000	0x0011AEB4	00000000	00000000
C011AED0	00000000	00000000	00000000	00000000
C011AEE0	00000000	00000000	00000000	00000000
C011AEF0	00000000	00000000	00000000	00000000
C011AF00	00000000	00000000	00000000	00000000
C011AF10	00000000	00000000	00000000	00000000
C011AF20	00000000	00000000	00000000	00000000
C011AF30	00000000	00000000	00000000	00000000
C011AF40	00000000	00000000	00000000	00000000
C011AF50	00000000	00000000	00000000	00000000
C011AF60	00000000	00000000	00000000	00000000
C011AF70	00000000	00000000	00000000	00000000
C011AF80	00000000	00000000	00000000	00000000

在 UNIX V6++中获取进程的代码段和可交换部分起始位置的逻辑地址和物理地址的方法：

首先找到固定逻辑地址的user结构，然后再user结构中找到procp成员找到proc表位置，在proc表中找到该进程的proc结构，然后根据proc结构中p\_flag是否具有SLOAD标志位可判断该进程图像的可交换部分是否在内存。如果在内存，则p\_addr为进程图像可交换部分在内存的起始地址，如果在盘交换区，则p\_addr为进程图像可交换部分在盘交换区上的盘块号。

根据proc结构中p\_texp找到该进程代码段的TEXT结构，并根据其中x\_ccount的值，判断代码段是否在内存。如果x\_ccount>=1，则代码段在内存，x\_caddr为其内存起始地址；如果x\_ccount==0，则代码段在盘交换区，x\_daddr为其在盘交换区的起始盘块号。

0xC0208000处内存的情况，是进程相对虚实地址映射表的位置

Monitors

+

×

0xC0208000 <Hex>

0xC0208000 : 0xC0208000 <H>

»1

◆ 0xC03FF000

◆ 0x0040F000

◆ 0xC0119600

◆ 0xC011AE94

◆ 0xC0208000

Address	0 - 3	4 - 7	8 - B	C - F	^
C0208000	00000004	00000004	00000004	0000500	
C0208010	00000004	00000004	00000004	0000500	
C0208020	00000004	00000004	00000004	0000500	
C0208030	00000004	00000004	00000004	0000000	
C0208040	00000004	00000004	00000004	0000000	
C0208050	00000004	00000004	00000004	0000000	
C0208060	00000004	00000004	00000004	0000000	
C0208070	00000004	00000004	00000004	0000000	
C0208080	00000004	00000004	00000004	0000000	
C0208090	00000004	00000004	00000004	0000000	
C02080A0	00000004	00000004	00000004	0000000	
C02080B0	00000004	00000004	00000004	0000000	
C02080C0	00000004	00000004	00000004	0000000	
C02080D0	00000004	00000004	00000004	0000000	
C02080E0	00000004	00000004	00000004	0000000	
C02080F0	00000004	00000004	00000004	0000000	
C0208100	00000004	00000004	00000004	0000000	
C0208110	00000004	00000004	00000004	0000000	
C0208120	00000004	00000004	00000004	0000000	
C0208130	00000004	00000004	00000004	0000000	
C0208140	00000004	00000004	00000004	0000000	
C0208150	00000004	00000004	00000004	0000000	
C0208160	00000004	00000004	00000004	0000000	

1025#号页表的起始地址，可以看到1025-1027号页表为代码段，1028-1030位为数据段

Memory	Monitors	0xC0209004 <Hex>	0xC0209004 : 0xC0209004 <H>	+ New Renderings...																																													
	<ul style="list-style-type: none"> <li>0xC03FF000</li> <li>0x0040F000</li> <li>0xC0119600</li> <li>0xC011AE94</li> <li>0xC0208000</li> <li>0xC0209004</li> </ul>	<table> <tr> <th>Address</th><th>0 - 3</th><th>4 - 7</th><th>8 - B</th><th>C - F</th></tr> <tr><td>C0209000</td><td>00000004</td><td>00000005</td><td>00001005</td><td>00002005</td></tr> <tr><td>C0209010</td><td>00001007</td><td>00002007</td><td>00003007</td><td>00000004</td></tr> <tr><td>C0209020</td><td>00000004</td><td>00000004</td><td>00000004</td><td>00000004</td></tr> <tr><td>C0209030</td><td>00000004</td><td>00000004</td><td>00000004</td><td>00000004</td></tr> <tr><td>C0209040</td><td>00000004</td><td>00000004</td><td>00000004</td><td>00000004</td></tr> <tr><td>C0209050</td><td>00000004</td><td>00000004</td><td>00000004</td><td>00000004</td></tr> <tr><td>C0209060</td><td>00000004</td><td>00000004</td><td>00000004</td><td>00000004</td></tr> <tr><td>C0209070</td><td>00000004</td><td>00000004</td><td>00000004</td><td>00000004</td></tr> </table>	Address	0 - 3	4 - 7	8 - B	C - F	C0209000	00000004	00000005	00001005	00002005	C0209010	00001007	00002007	00003007	00000004	C0209020	00000004	00000004	00000004	00000004	C0209030	00000004	00000004	00000004	00000004	C0209040	00000004	00000004	00000004	00000004	C0209050	00000004	00000004	00000004	00000004	C0209060	00000004	00000004	00000004	00000004	C0209070	00000004	00000004	00000004	00000004		
Address	0 - 3	4 - 7	8 - B	C - F																																													
C0209000	00000004	00000005	00001005	00002005																																													
C0209010	00001007	00002007	00003007	00000004																																													
C0209020	00000004	00000004	00000004	00000004																																													
C0209030	00000004	00000004	00000004	00000004																																													
C0209040	00000004	00000004	00000004	00000004																																													
C0209050	00000004	00000004	00000004	00000004																																													
C0209060	00000004	00000004	00000004	00000004																																													
C0209070	00000004	00000004	00000004	00000004																																													

查看堆栈段的地址空间，为2047号页表，标志位为7，而1031-2046号页表全部标志位为4

Monitors

0xC0209FFC <Hex>

0xC0209FFC : 0xC0209FFC <H>

New Renderings...

0xC03FF000

0x0040F000

0xC0119600

0xC011AE94

0xC0208000

0xC0209004

0xC0209FFC

Address	0 - 3	4 - 7	8 - B	C - F	
C0209FF0	00000004	00000004	00000004	00004007	
C020A000	00000004	00000004	00000004	00004007	
C020A010	00000004	00000004	00000004	00004007	
C020A020	00000004	00000004	00000004	00004007	
C020A030	00000208	00004000	00000400	00003200	
C020A040	00000000	00000000	00000000	C0600040	
C020A050	00000000	00000000	00000000	C0600040	
C020A060	00000600	00003600	00000000	00000000	
C020A070	00000000	40400040	7373622E	00000000	

补全进程的相对虚实地址映射表



页号	地址	值		
		高 20 位页框号	低 12 位标志位 (u/s r/w p)	
0#	0xC0208000~0xC0208003			
	.....			
1024#	0xC0208000~0xC0208003			
1025#	0xC0209004~0xC0209007	0x420	005 (0000 0000 0101)	代码段
1026#	0xC0209008~0xC020900B	0x421	005 (0000 0000 0101)	
1027#	0xC020900C~0xC020900F	0x422	005 (0000 0000 0101)	
1028#	0xC0209010~0xC0209013	0x441	007 (0000 0000 0111)	数据段
1029#	0xC0209014~0xC0209017	0x442	007 (0000 0000 0111)	
1030#	0xC0209018~0xC020901B	0x443	007 (0000 0000 0111)	
1031#	0xC020901C~0xC020901F		004 (0000 0000 0100)	
	.....		.....	
2047#	0xC0209FFC~0xC0209FFF	0x444	007 (0000 0000 0111)	堆栈段

现运行进程的四张页表起始逻辑地址为3G+2M+0K, 3G+2M+4K, 3G+2M+8K, 3G+2M+12K

#### 0x200号页框 页目录 0# 1#和768#号页表

Monitors		0xC0200000 <Hex> 0xC0200000 : 0xC0200000 <H				+ New Renderings...	
		Address	0 - 3	4 - 7	8 - B	C - F	
◆ 0xC03FF000		C0200000	00202027	00203027	00000000	00000000	
◆ 0x0040F000		C0200010	00000000	00000000	00000000	00000000	
◆ 0xC0119600		C0200020	00000000	00000000	00000000	00000000	
◆ 0xC011AE94		C0200030	00000000	00000000	00000000	00000000	
◆ 0xC0208000		C0200040	00000000	00000000	00000000	00000000	
◆ 0xC0209004		C0200050	00000000	00000000	00000000	00000000	
◆ 0xC0209FFC		C0200060	00000000	00000000	00000000	00000000	
◆ 0x00003400		C0200070	00000000	00000000	00000000	00000000	
◆ 0xC0200000		C0200080	00000000	00000000	00000000	00000000	
◆ 0xC0201000		C0200090	00000000	00000000	00000000	00000000	
◆ 0xC0202000		C02000A0	00000000	00000000	00000000	00000000	

Monitors		0xC0200000 <Hex> 0xC0200000 : 0xC0200000 <H				+ New Renderings...	
		Address	0 - 3	4 - 7	8 - B	C - F	
◆ 0xC03FF000		C0200BA0	00000000	00000000	00000000	00000000	
◆ 0x0040F000		C0200BB0	00000000	00000000	00000000	00000000	
◆ 0xC0119600		C0200BC0	00000000	00000000	00000000	00000000	
◆ 0xC011AE94		C0200BD0	00000000	00000000	00000000	00000000	
◆ 0xC0208000		C0200BE0	00000000	00000000	00000000	00000000	
◆ 0xC0209004		C0200BF0	00000000	00000000	00000000	00000000	
◆ 0xC0209FFC		C0200C00	00201023	00000000	00000000	00000000	
◆ 0x00003400		C0200C10	00000000	00000000	00000000	00000000	
◆ 0xC0200000		C0200C20	00000000	00000000	00000000	00000000	
◆ 0xC0201000		C0200C30	00000000	00000000	00000000	00000000	
◆ 0xC0202000		C0200C40	00000000	00000000	00000000	00000000	

## 0x201号页框 内核页表

Monitors	0xC0201000 <Hex>	0xC0201000 : 0xC0201000 <H	+ New Renderings..		
<ul style="list-style-type: none"> <li>0xC03FF000</li> <li>0x0040F000</li> <li>0xC0119600</li> <li>0xC011AE94</li> <li>0xC0208000</li> <li>0xC0209004</li> <li>0xC0209FFC</li> <li>0x00003400</li> <li>0xC0200000</li> <li>0xC0201000</li> <li>0xC0202000</li> </ul>	Address	0 - 3	4 - 7	8 - B	C - F
	C0201000	00000003	00001003	00002003	00003003
	C0201010	00004003	00005003	00006003	00007003
	C0201020	00008003	00009003	0000A003	0000B003
	C0201030	0000C003	0000D003	0000E003	0000F023
	C0201040	00010003	00011003	00012003	00013003
	C0201050	00014003	00015003	00016003	00017003
	C0201060	00018003	00019003	0001A003	0001B003
	C0201070	0001C003	0001D003	0001E003	0001F003
	C0201080	00020003	00021003	00022003	00023003
	C0201090	00024003	00025003	00026003	00027003
	C02010A0	00028003	00029003	0002A003	0002B003
	C02010B0	0002C003	0002D003	0002E003	0002F003
	C02010C0	00030003	00031003	00032003	00033003

## 0x202号页框 编译器预留

Monitors	0xC0202000 <Hex>	0xC0202000 : 0xC0202000 <H	+ New Renderings...		
<ul style="list-style-type: none"> <li>0xC03FF000</li> <li>0x0040F000</li> <li>0xC0119600</li> <li>0xC011AE94</li> <li>0xC0208000</li> <li>0xC0209004</li> <li>0xC0209FFC</li> <li>0x00003400</li> <li>0xC0200000</li> <li>0xC0201000</li> <li>0xC0202000</li> </ul>	Address	0 - 3	4 - 7	8 - B	C - F
	C0202000	00000067	00001004	00002004	00003004
	C0202010	00004006	00005006	00006006	00007006
	C0202020	00008006	00009006	0000A006	0000B006
	C0202030	0000C006	0000D006	0000E006	0000F006
	C0202040	00010006	00011006	00012006	00013006
	C0202050	00014006	00015006	00016006	00017006
	C0202060	00018006	00019006	0001A006	0001B006
	C0202070	0001C006	0001D006	0001E006	0001F006
	C0202080	00020006	00021006	00022006	00023006
	C0202090	00024006	00025006	00026006	00027006
	C02020A0	00028006	00029006	0002A006	0002B006
	C02020B0	0002C006	0002D006	0002E006	0002F006
	C02020C0	00030006	00031006	00032006	00033006
	C02020D0	00034006	00035006	00036006	00037006
	C02020E0	00038006	00039006	0003A006	0003B006
	C02020F0	0003C006	0003D006	0003E006	0003F006
	C0202100	00040006	00041006	00042006	00043006
	C0202110	00044006	00045006	00046006	00047006

## 0x203号页框 用户页表

Monitors	0xC0203000 <Hex>		0xC0203000 : 0xC0203000 <H		+ New Renderings...
	Address	0 - 3	4 - 7	8 - B	C - F
0xC03FF000	C0203000	00400006	0040C065	0040D065	0040E065
0x0040F000	C0203010	00410067	00411067	00412067	00412066
0xC0119600	C0203020	00413066	00409006	0040A006	0040B006
0xC011AE94	C0203030	0040C006	0040D006	0040E006	0040F006
0xC0208000	C0203040	00410006	00411006	00412006	00413006
0xC0209004	C0203050	00414006	00415006	00416006	00417006
0xC0209FFC	C0203060	00418006	00419006	0041A006	0041B006
0x00003400	C0203070	0041C006	0041D006	0041E006	0041F006
0xC0200000	C0203080	00420006	00421006	00422006	00423006
0xC0201000	C0203090	00424006	00425006	00426006	00427006
0xC0202000	C02030A0	00428006	00429006	0042A006	0042B006
0xC0203000	C02030B0	0042C006	0042D006	0042E006	0042F006
	C02030C0	00430006	00431006	00432006	00433006
	C02030D0	00434006	00435006	00436006	00437006
	C02030E0	00438006	00439006	0043A006	0043B006
	C02030F0	0043C006	0043D006	0043E006	0043F006
	C0203100	00440006	00441006	00442006	00443006

现运行进程四张页表的相对虚实地址映射表

地址	高20位页框号	标志位
0xC0200000~0xC0200FFC	0x200	007 (0000 0000 0111)
0xC0201000~0xC0201FFC	0x201	003 (0000 0000 0011)
0xC0202000~0xC0202FFC	0x202	/
0xC0203000~0xC0203FFC	0x203	005,007

相对地址映射表				内存	
PBA	u/s	r/w	P	保留区域	
0#				操作系统内核	
1#				(子程序和静态数据块)	
2#				内核对象	
...				0x200	
1023#				0x201	
				0x202	
				0x203	
1025#	0	1	0		
1026#	P	1	0		
1027#	0	1	0		
1028	1	1	1		

现运行进程 页表

4M

4M+6



1029 2 1 1 1  
1030 3 1 1 1

2047 4 1 1 1

页码 0x2005

0x202	1	0	1
0x203	1	0	1
0x204	0	1	1

内存页表 0x201

	PBA	U/S	I/W	P
0	0x0	0	1	1
1	0x1	0	1	1
2	0x2			

0x203 用户页表

PBA U/S I/W P

0#	0x400	1	0	1
1#	0x40D	1	1	1
2#	0x40E	1	1	1
3#	0x40F	1	1	1
4#	0x410	1	1	1
5#	0x411	1	1	1
6#	0x412	1	1	1

1003 1004 0x40F 1 1 1

0x202 用户页表

空

1003 0x413 1 1 1

代码段

PPUA

数据段

堆栈段