**同济大学计算机系**

**操作系统课程实验报告**

****

**学 号 2251557**

**姓 名 代文波**

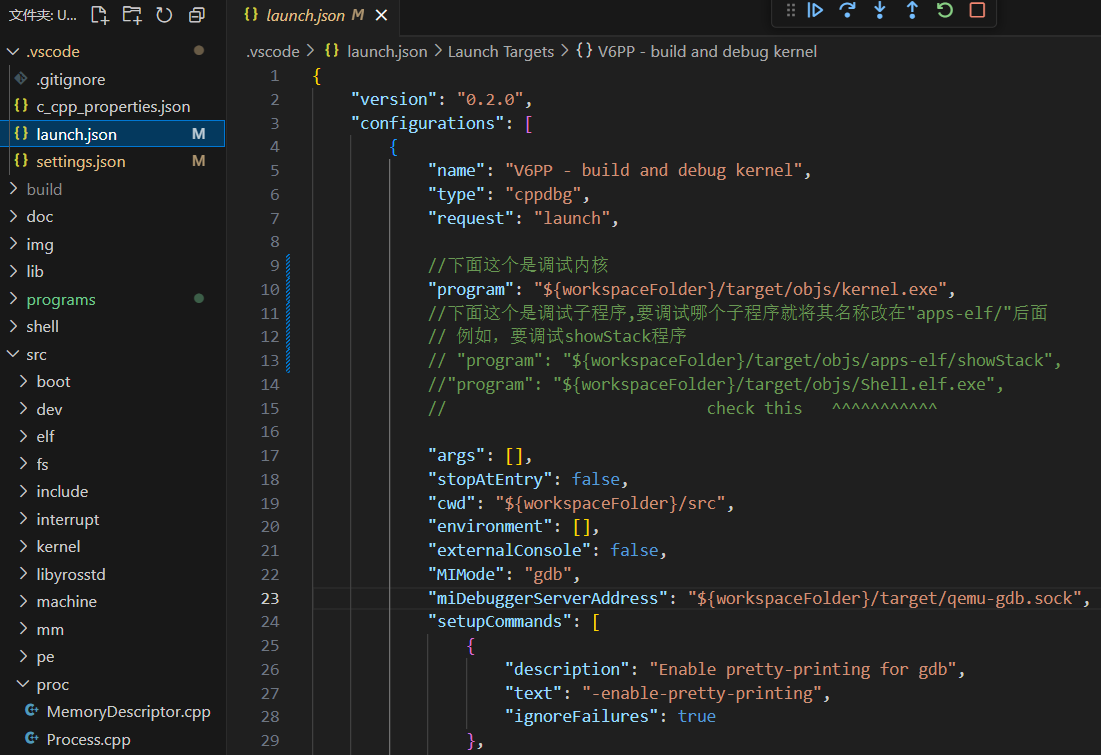
**专 业 计算机科学与技术**

**授课老师 方钰**

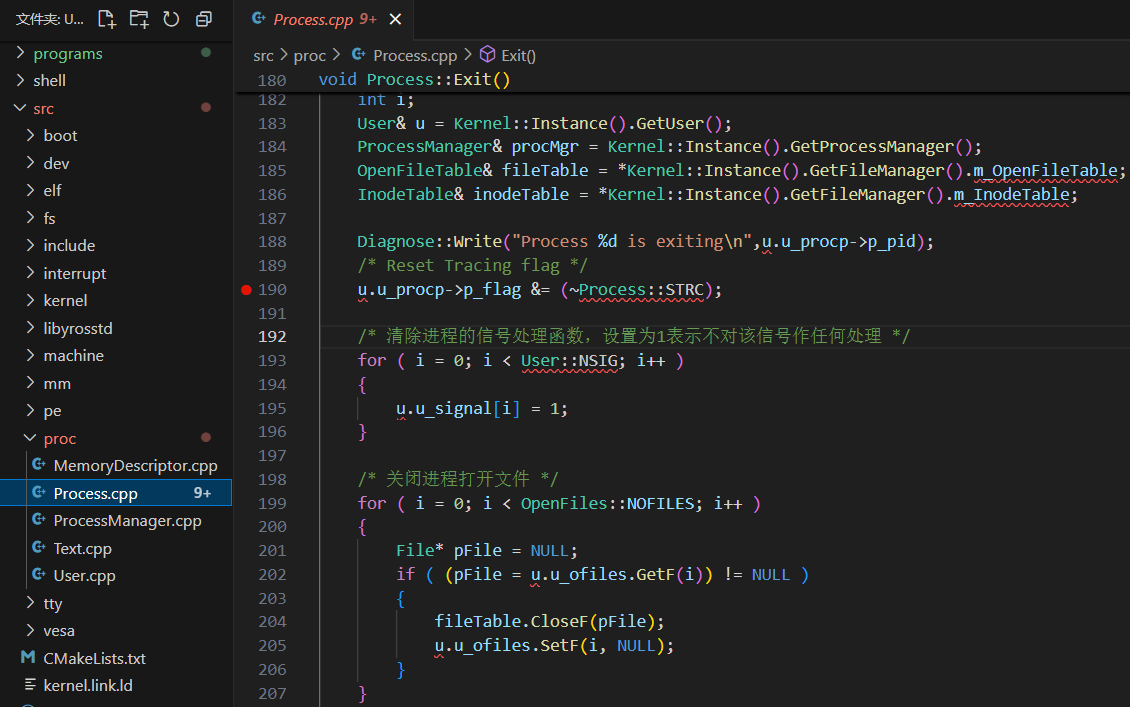
一、实验4.1~4.2 基础内容

1.1准备工作

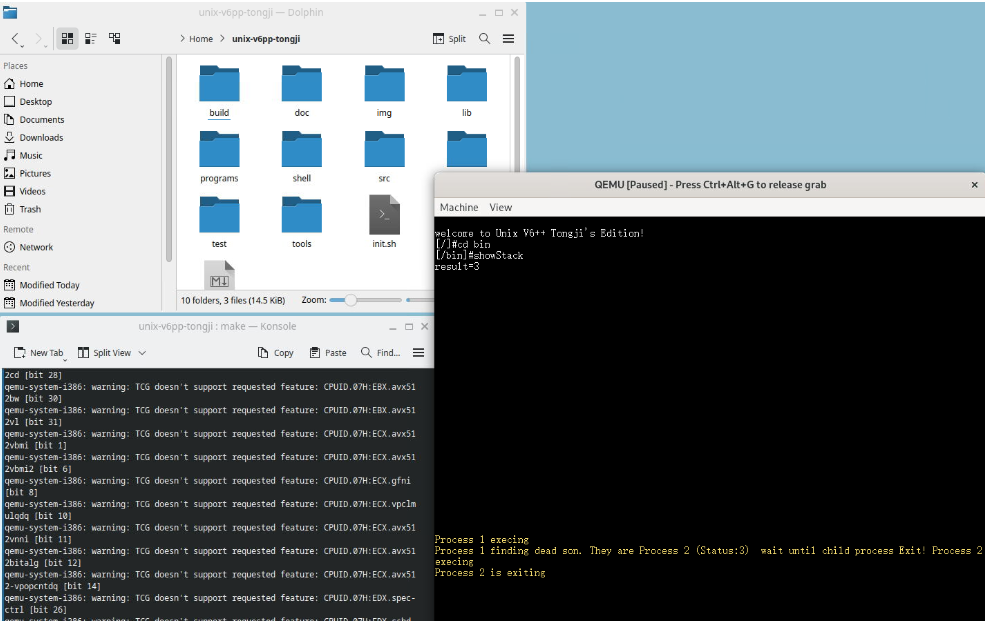
（1）设置调试对象为Kernel.exe

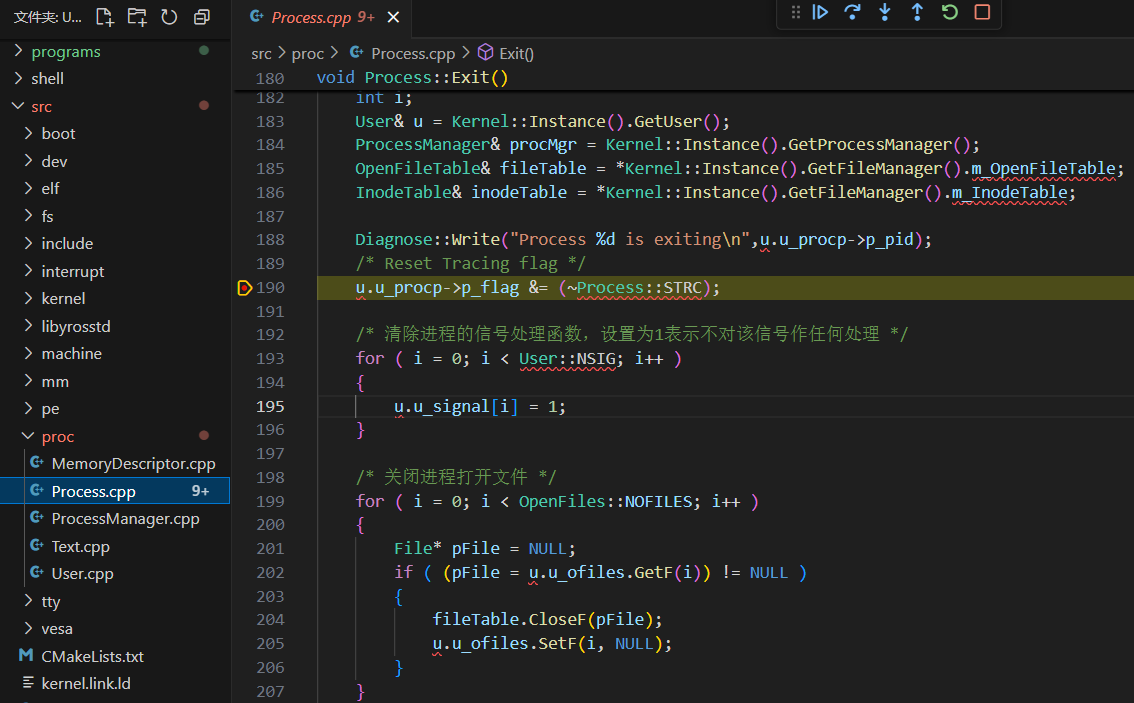


（2）设置断点位置



（3）调试停在断点位置处

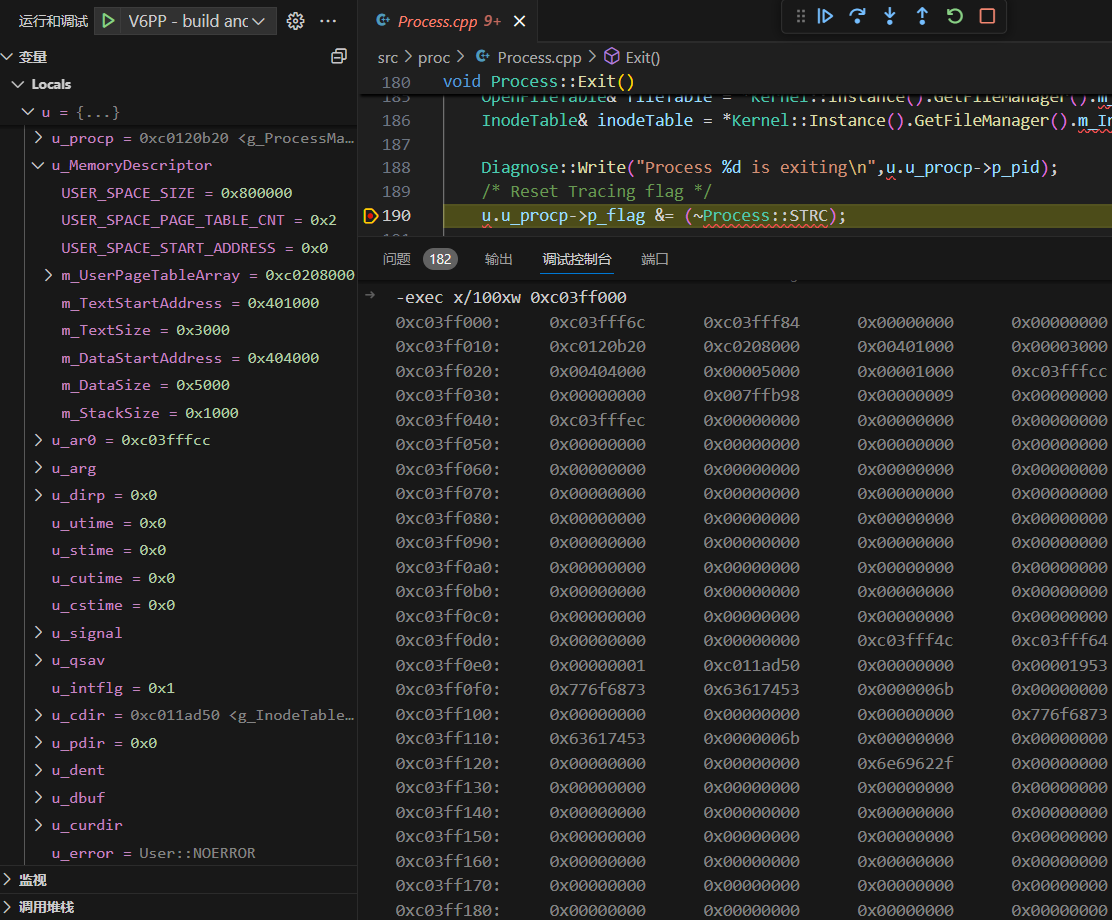




下面将逐步获取User结构、Proc结构、Text结构的内容。

1.2 User结构

UNIX V6++的进程 User 结构逻辑地址是固定的，始终位于 3G ~ 3G+4M 部分的最后一页，即：0xC03FF000（3G+4M-4K）。我们利用该逻辑地址，通过-exec x /100xw 0xC03FF000 命令可以找到进程的User结构的具体内容。

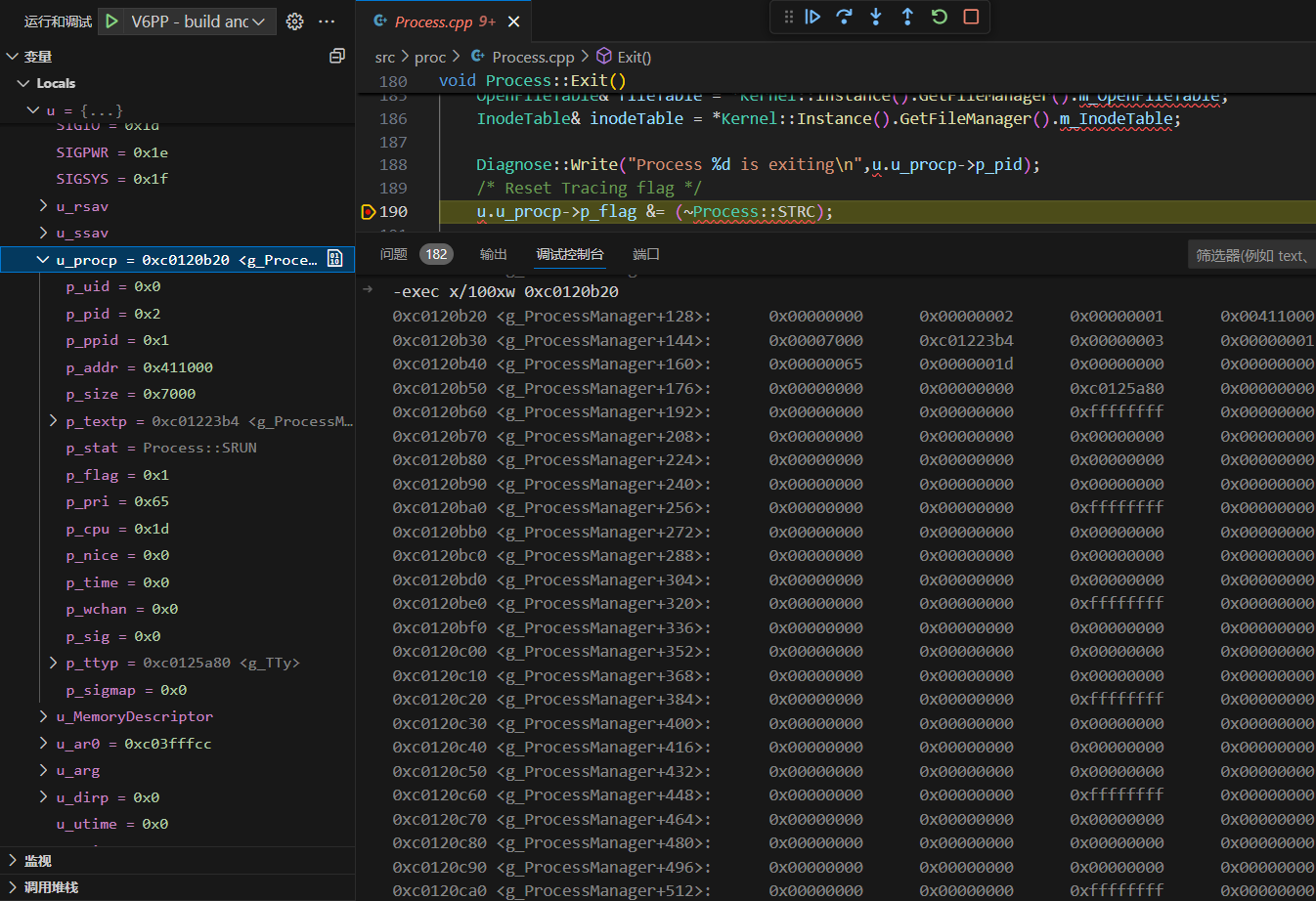


根据信息绘制表格如下：

|  |  |  |
| --- | --- | --- |
| User结构 | | |
| 变量名称 | 含义 | 值 |
| Process\* u\_procp | Proc结构的逻辑地址 | 0xC0120B20 |
| MemoryDescriptor u\_MemoryDescriptor（具体内容如下，此处均为逻辑地址） | | |
| PageTable\* m\_UserPageTableArray | 相对虚实映射表首地址 | 0xC0208000 |
| unsigned long m\_TextStartAddress | 代码段起始地址 | 0x401000=4M+4K |
| unsigned long m\_TextSize | 代码段长度 | 0x3000=12K |
| unsigned long m\_DataStartAddress | 数据段起始地址 | 0x404000=4M+16K |
| unsigned long m\_DataSize | 数据段长度 | 0x5000=20K |
| unsigned long m\_StackSize | 栈段长度 | 0x1000=4K |

1.3 Proc结构

通过观察User结构可以获得的一个重要信息是u\_procp的值，即进程proc结构的逻辑地址为0xC0120B20。我们利用该逻辑地址，通过-exec x /100xw 0xC0120B20命令可以找到进程的Proc结构。

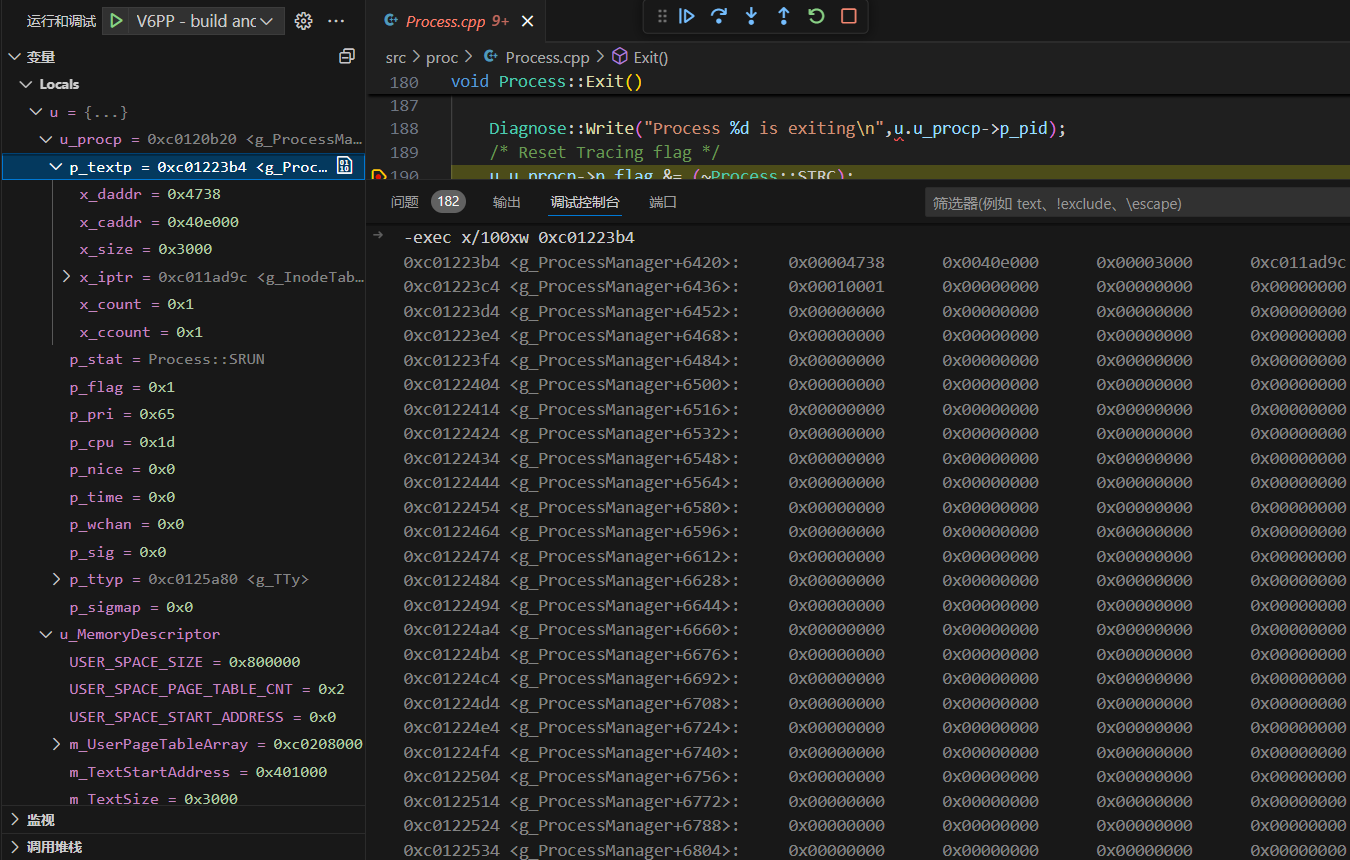


根据信息绘制表格如下：

|  |  |  |
| --- | --- | --- |
| Proc结构 | | |
| 变量名称 | 含义 | 值 |
| short p\_uid | 用户ID | 0 |
| int p\_pid | 进程标识数 | 2 |
| int p\_ppid | 父进程标识数 | 1 |
| unsigned long p\_addr | User结构即ppda区物理地址 | 0x411000 |
| unsigned int p\_size | 除共享正文段的长度，以字节为单位 | 0x7000=28K |
| Text\* p\_textp | 指向代码段Text结构的逻辑地址 | 0xc01223B4 |
| ProcessState p\_stat | 进程调度状态 | 3=SRUN |
| int p\_flag | 进程标志位 | 1 |
| int p\_pri | 进程优先数 | 0x65 |
| int p\_cpu | cpu值，用于计算p\_pri | 0x1d |
| int p\_nice | 进程优先数微调参数 | 0x0 |
| int p\_time | 进程在盘上（内存里）驻留时间 | 0x0 |
| unsigned long p\_wchan | 进程睡眠原因 | 0x0 |

1.4 Text结构

通过观察Proc结构可以获得的一个重要信息是p\_textp的值，即进程代码段逻辑地址Text结构的逻辑地址为0xc01223B4。我们利用该逻辑地址，通过-exec x /100xw 0xc01223B4命令可以找到进程代码段的Text结构。



根据信息绘制表格如下：

|  |  |  |
| --- | --- | --- |
| Text结构 | | |
| 变量名称 | 含义 | 值 |
| int x\_daddr | 代码段在盘交换区的地址 | 0x4738 |
| Unsigned long x\_caddr | 代码段的起始地址（物理地址） | 0x40E000 |
| Unsigned int x\_size | 代码段长度 | 0x3000 |
| Inode\* x\_iptr | 内存Inode地址 | 0xc011AD9C |
| Unsigned short x\_count | 共享正文段的进程数 | 1 |
| Unsigned short x\_ccount | 共享正文段且图像在内存的进程数 | 1 |

1.5 总结

经过上面的实验，我们找到了进程图像的两大部分——进程图像的可交换部分和代码段，在逻辑地址空间和物理地址空间的分布。

由此，读者可根据课程所学知识，绘制出进程的相对虚实地址映射表和物理页表，并通过后续实验，验证是否正确。

进程图像的完整信息：

|  |  |  |  |
| --- | --- | --- | --- |
| 进程图像的完整信息 | | | |
| 名称 | 逻辑地址 | 物理地址 | 大小 |
| 代码段 | 0x401000=4M+4K | 0x40E000 | 12k |
| 可交换部分 | 0xC03FF000=3G+4M-4K | 0x411000 | 28k |
| ppda区 | 0xC03FF000=3G+4M-4K | 0x411000 | 4k |
| 数据段 | 0x404000=4M+16K | 0x412000 | 20k |
| 堆栈段 |  | 0x417000 | 4k |

经验总结：

（1）可交换部分

逻辑地址：

固定值，始终位于3G~3G+4M的最后一页，即0xc03FF000=3G+4M-4K

物理地址：

通过固定的User结构的逻辑地址查看到u\_procp（进程Proc结构的逻辑地址），然后找到Proc结构的p\_addr（ppda区的物理地址）。

（2）代码段

逻辑地址：

通过固定的User结构找到u\_MemoryDescriptor.m\_TextStartAddress（代码段的逻辑地址）

物理地址：

通过固定的User结构的逻辑地址（0xc03FF000）查看到u\_procp（进程Proc结构的逻辑地址），然后找到Proc结构的p\_textp（进程Text结构的逻辑地址），之后找到Text结构的x\_caddr（代码段在内存的物理地址）或者x\_daddr（代码段在盘交换区的物理地址）。

二、实验4.1、4.2进阶

2.1进程的相对虚实地址映射表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 页号 | 地址 | 值 | | | | |
| Page Base Address | 中间标志位 | u/s | r/w | p |
| 0# | 0xC0208000~0xC0208003 | x | x | x | x | x |
| … | … | … | … | … | … | … |
| 1024# | 0xC0208000~0xC0208003 | x | … | x | x | x |
| 1025# | 0xC0209004~0xC0209007 | 0x0 | … | 1 | 0 | 1 |
| 1026# | 0xC0209008~0xC020900B | 0x1 | … | 1 | 0 | 1 |
| 1027# | 0xC020900C~0xC020900F | 0x2 | … | 1 | 0 | 1 |
| 1028# | 0xC0209010~0xC0209013 | 0x1 | … | 1 | 1 | 1 |
| 1029# | 0xC0209014~0xC0209017 | 0x2 | … | 1 | 1 | 1 |
| 1030# | 0xC0209018~0xC020901B | 0x3 | … | 1 | 1 | 1 |
| 1031# | 0xC020901C~0xC020901F | 0x4 | … | 1 | 1 | 1 |
| 1032# | 0xC0209020~0xC0209023 | 0x5 | … | 1 | 1 | 1 |
| 1033# | 0xC0209024~0xC0209027 | x | … | x | x | x |
| … | … | … | … | … | … | … |
| 2047# | 0xC0209FFC~0xC0209FFF | 0x6 | … | 1 | 1 | 1 |

2.2 四张物理页表

2.2.1 目录页（0x200#物理页框）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 页号 | 地址 | 值 | | | | |
| Page Base Address | 中间标志位 | u/s | r/w | p |
| #0 | 0xC0200000~0xC0200003 | 0x202 | … | 1 | 1 | 1 |
| #1 | 0xC0200004~0xC0200007 | 0x203 | … | 1 | 1 | 1 |
| … | … |  | … | … | … | … |
| #768 | 0xC0200C00~0xC0200C03 | 0x201 | … | 0 | 1 | 1 |
| … | … |  | … | … | … | … |

2.2.2 内核页表（0x201#物理页框）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 页号 | 地址 | 值 | | | | |
| Page Base Address | 中间标志位 | u/s | r/w | p |
| #0 | 0xC0201000~0xC0201003 | 0x0 | … | 0 | 1 | 1 |
| #1 | 0xC0201004~0xC0201007 | 0x1 | … | 0 | 1 | 1 |
| … | … |  | … | … | … | … |
| #1023 | 0xC0201FFC~0xC0201FFF | 0x411 | … | 0 | 1 | 1 |

2.2.3 用户页表（0x202、0x203）

（1）用户页表1（0x202）

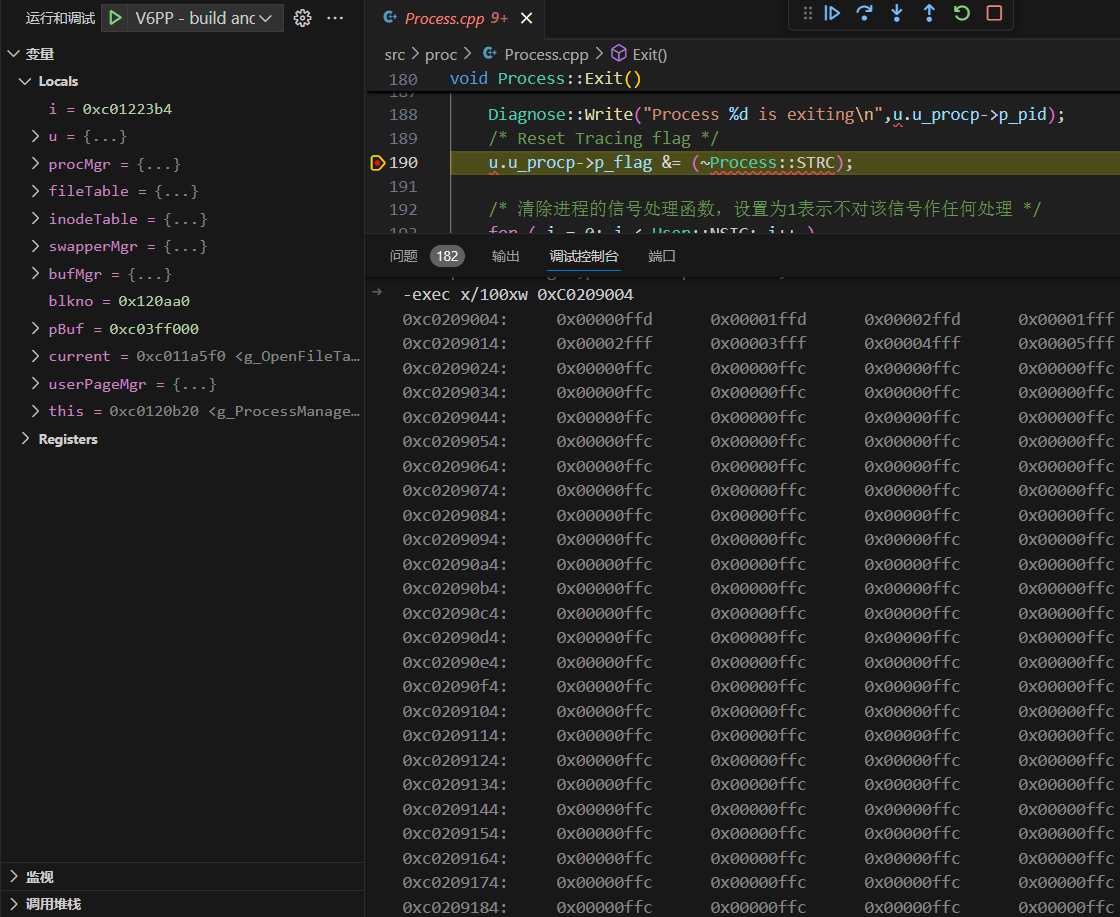
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 页号 | 地址 | 值 | | | | |
| Page Base Address | 中间标志位 | u/s | r/w | p |
| … | … | … | … | … | … | … |

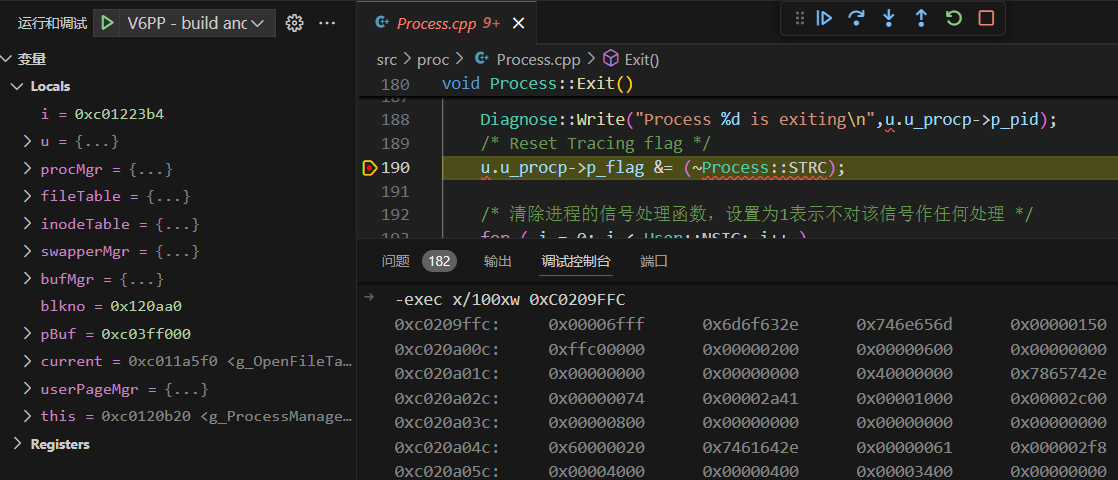
（2）用户页表2（0x203）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 页号 | 地址 | 值 | | | | |
| Page Base Address | 中间标志位 | u/s | r/w | p |
| #0 | 0xC0203000~0xC0203003 | … | … | … | … | … |
| #1 | 0xC0203004~0xC0203007 | 0x40E | … | 1 | 0 | 1 |
| #2 | 0xC0203008~0xC020300B | 0x40F | … | 1 | 0 | 1 |
| #3 | 0xC020300C~0xC020300F | 0x410 | … | 1 | 0 | 1 |
| #4 | 0xC0203010~0xC0203013 | 0x412 | … | 1 | 1 | 1 |
| #5 | 0xC0203014~0xC0203017 | 0x413 | … | 1 | 1 | 1 |
| #6 | 0xC0203018~0xC020301B | 0x414 | … | 1 | 1 | 1 |
| #7 | 0xC020301C~0xC020301F | 0x415 | … | 1 | 1 | 1 |
| #8 | 0xC0203020~0xC0203023 | 0x416 | … | 1 | 1 | 1 |
| #9 | 0xC0203024~0xC0203027 | x | … | x | x | x |
| … | … |  | … | … | … | … |
| #1023 | 0xC0203FFC~0xC0203FFF | 0x417 | … | 1 | 1 | 1 |

三、实验4.3

通过上面的实验，我们知道进程的相对虚实地址映射表位于0xC0208000（逻辑地址）起始的两个4K的页面中。由于第一个页面和第二个页面的第一项都留给编译器了，所以我们先直接从第二个页表的第二项开始看，即通过查看0xC0209004起始的内存单元，来了解相对虚实地址映射表中进程的代码段内容和数据段内容；然后查看第二个页表的最后一项，即通过查看0xC0209FFC起始的内存单元，来了解相对虚实地址映射表中进程堆栈段内容。





|  |  |  |  |
| --- | --- | --- | --- |
| 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 | x | x |
| … | … | … | … |
| 1024# | 0xC0208000~0xC0208003 | x | x |
| 1025# | 0xC0209004~0xC0209007 | 0x0 | 0xFFD（1111 1111 1101） |
| 1026# | 0xC0209008~0xC020900B | 0x1 | 0xFFD（1111 1111 1101） |
| 1027# | 0xC020900C~0xC020900F | 0x2 | 0xFFD（1111 1111 1101） |
| 1028# | 0xC0209010~0xC0209013 | 0x1 | 0xFFF（1111 1111 1111） |
| 1029# | 0xC0209014~0xC0209017 | 0x2 | 0xFFF（1111 1111 1111） |
| 1030# | 0xC0209018~0xC020901B | 0x3 | 0xFFF（1111 1111 1111） |
| 1031# | 0xC020901C~0xC020901F | 0x4 | 0xFFF（1111 1111 1111） |
| 1032# | 0xC0209020~0xC0209023 | 0x5 | 0xFFF（1111 1111 1111） |
| 1033# | 0xC0209024~0xC0209027 | 0x0 | 0xFFC（1111 1111 1100） |
| … | … | … |  |
| 2047# | 0xC0209FFC~0xC0209FFF | 0x6 | 0xFFF（1111 1111 1111） |

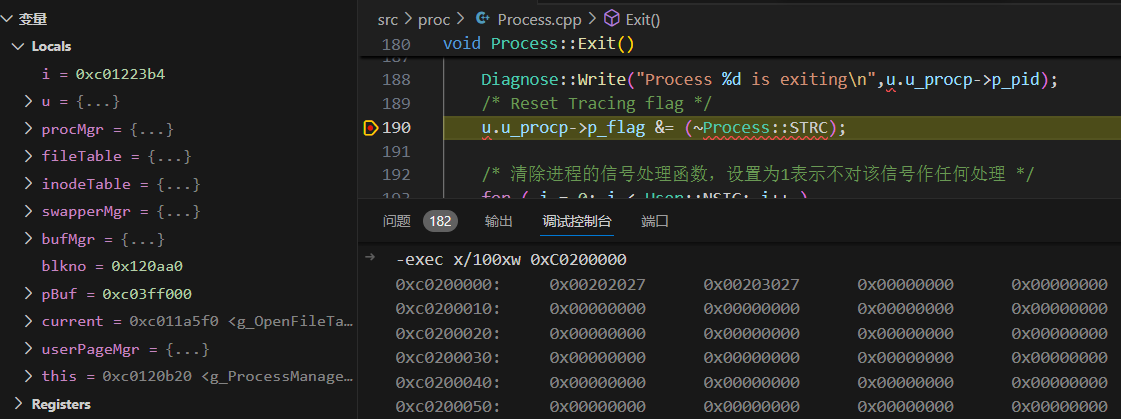
经验证，步骤二中给出的相对虚实地址映射表正确。

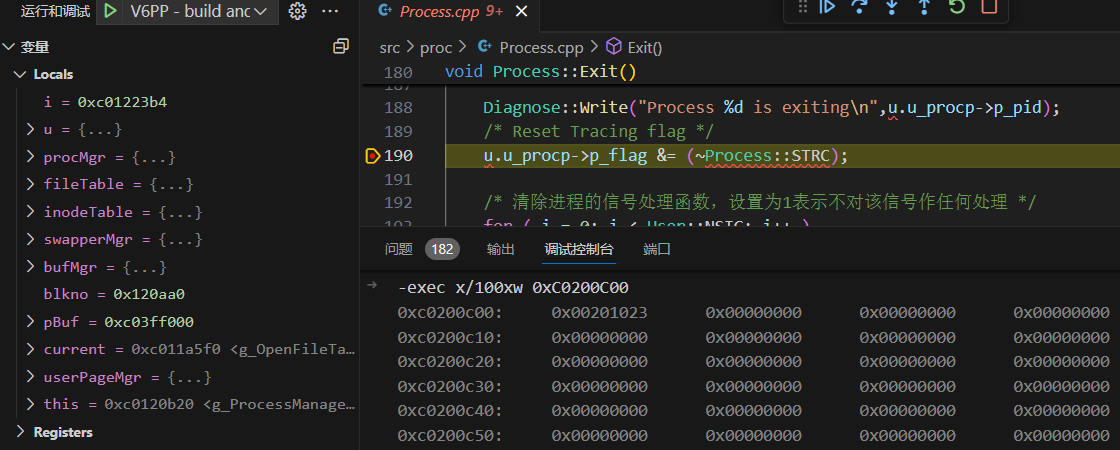
四、实验4.4

这四张表的逻辑地址：

这四张页表按照：目录页、内核页表、用户页表1、用户页表2的顺序排列在逻辑内存的3G+2M~3G+2M-4K的前四个页框内。这四张页表开始位置的逻辑地址依次为：0xC0200000（3G+2M）、0xC0201000（3G+2M+4K）、0xC0202000（3G+2M+8K）、0xC0203000（3G+2M+12K）。

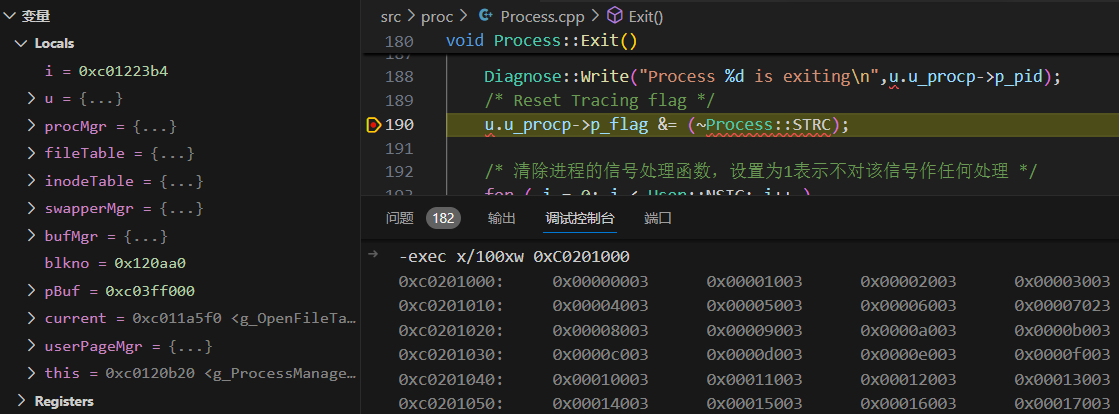
（1）目录页

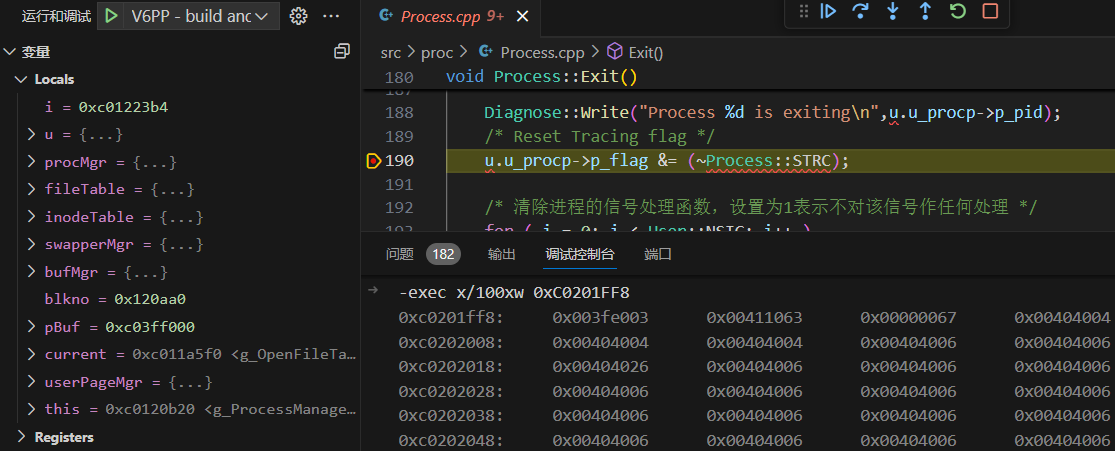




|  |  |  |  |
| --- | --- | --- | --- |
| 页号 | 地址 | 值 | |
| 高20位页编号 | 低12位标志位 |
| #0 | 0xC0200000~0xC0200003 | 0x202 | 0x027（0000 0010 0111） |
| #1 | 0xC0200004~0xC0200007 | 0x203 | 0x027（0000 0010 0111） |
| … | … | … | … |
| #768 | 0xC0200C00~0xC0200C03 | 0x201 | 0x023（0000 0010 0011） |
| … | … | … | … |

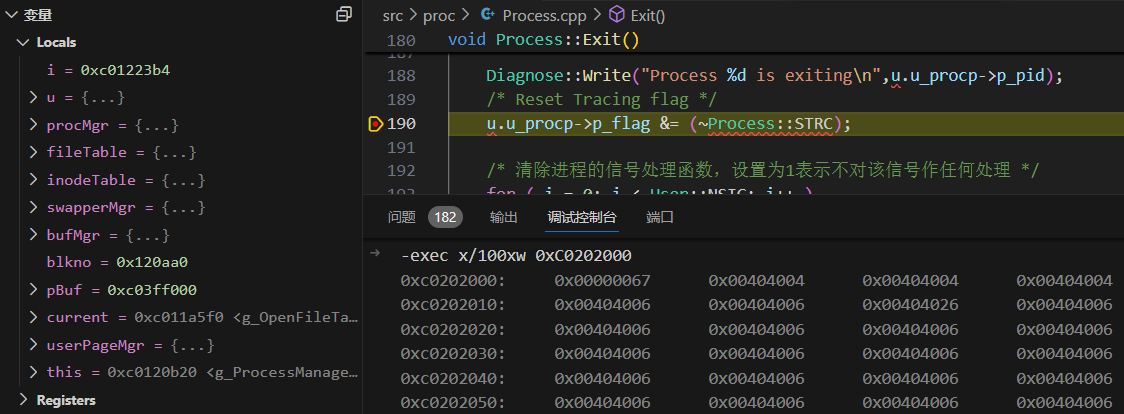
（2）内核页表

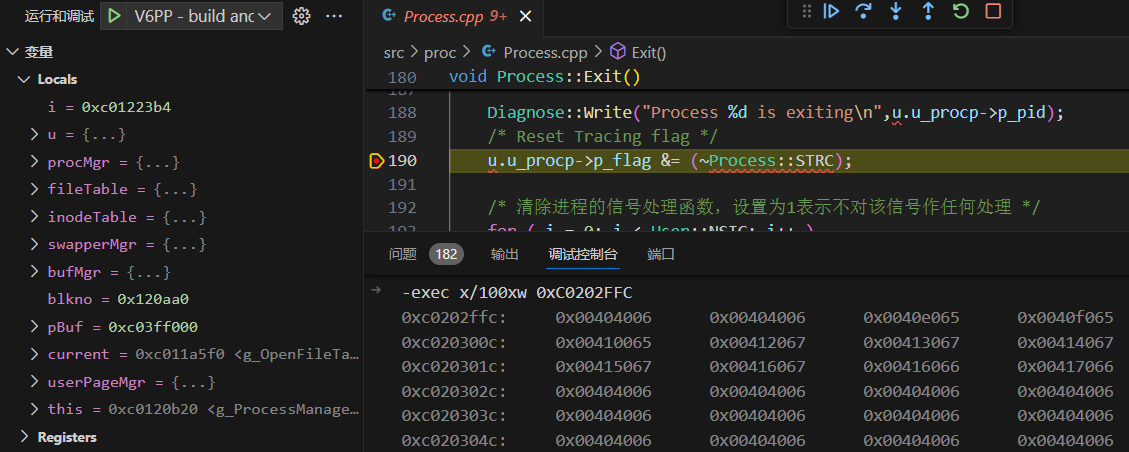




|  |  |  |  |
| --- | --- | --- | --- |
| 页号 | 地址 | 值 | |
| 高20位页编号 | 低12位标志位 |
| #0 | 0xC0201000~0xC0200003 | 0x0 | 0x003（0000 0000 0011） |
| #1 | 0xC0201004~0xC0200007 | 0x1 | 0x003（0000 0000 0011） |
| … | … | … | … |
| #1022 | 0xC0201FF8~0xC0200FFB | 0x3FE（=1022） | 0x003（0000 0000 0011） |
| #1023 | 0xC0201FFC~0xC0201FFF | 0x411 | 0x063（0000 0110 0011） |

（3）用户页表1

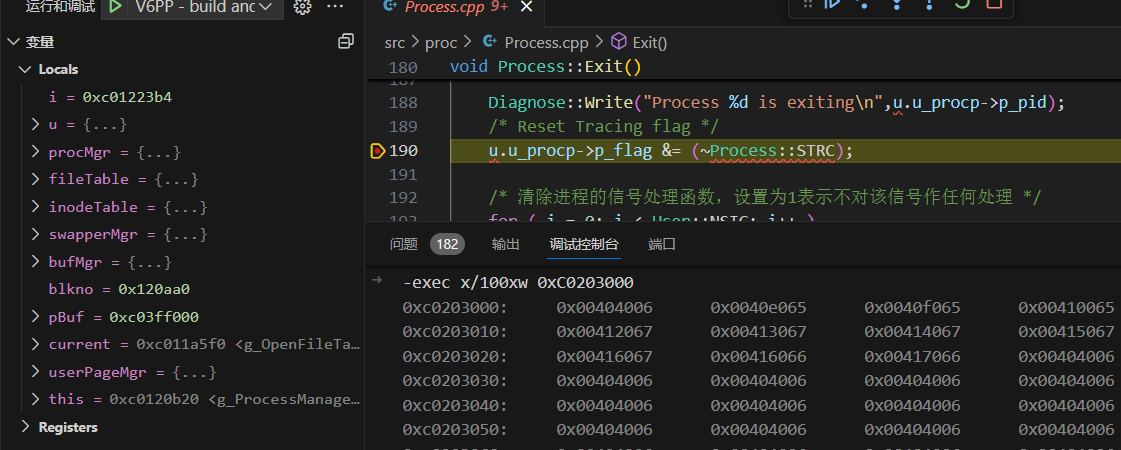


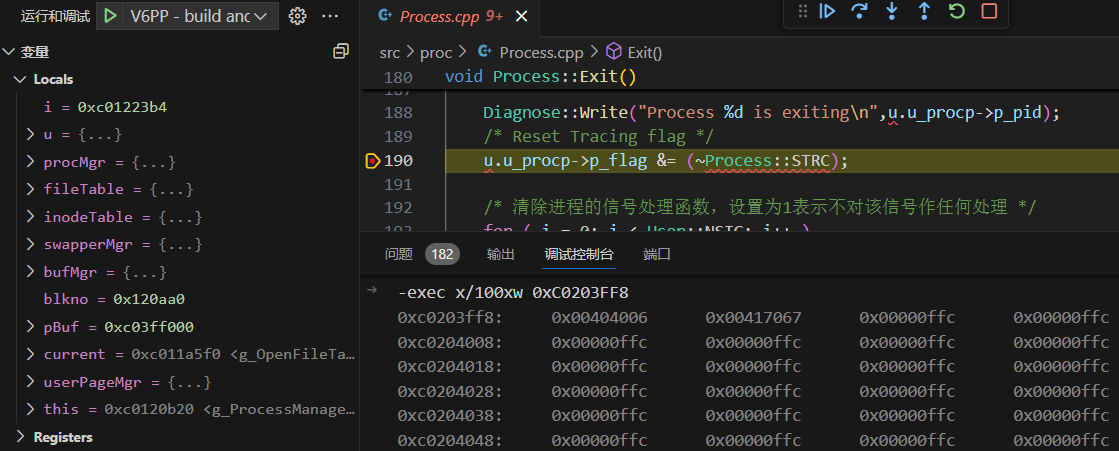


|  |  |  |  |
| --- | --- | --- | --- |
| 页号 | 地址 | 值 | |
| 高20位页编号 | 低12位标志位 |
| #0 | 0xC0202000~0xC0202003 | 0x0 | 0x067（0000 0110 0111） |
| … | … | … | … |
| #1023 | 0xC0202FFC~0xC0202FFF | 0x404 | 0x006（0000 0000 0110） |

由于这一部分是留给编译器使用的，所以这一部分页表的具体内容这里不做仔细研究了。

（4）用户页表2





|  |  |  |  |
| --- | --- | --- | --- |
| 页号 | 地址 | 值 | |
| 高20位页编号 | 低12位标志位 |
| #0 | 0xC0203000~0xC0203003 | 0x404 | 0x006（0000 0000 0110） |
| #1 | 0xC0203004~0xC0203007 | 0x40E | 0x065（0000 0000 0101） |
| #2 | 0xC0203008~0xC020300B | 0x40F | 0x065（0000 0000 0101） |
| #3 | 0xC020300C~0xC020300F | 0x410 | 0x065（0000 0000 0101） |
| #4 | 0xC0203010~0xC0203013 | 0x412 | 0x067（0000 0000 0111） |
| #5 | 0xC0203014~0xC0203017 | 0x413 | 0x067（0000 0000 0111） |
| #6 | 0xC0203018~0xC020301B | 0x414 | 0x067（0000 0000 0111） |
| #7 | 0xC020301C~0xC020301F | 0x415 | 0x067（0000 0000 0111） |
| #8 | 0xC0203020~0xC0203023 | 0x416 | 0x067（0000 0000 0111） |
| #9 | 0xC0203024~0xC0203027 | 0x416 | 0x066（0000 0000 0110） |
| … | … | … | … |
| #1022 | 0xC0203FF8~0xC0203FFB | 0x404 | 0x006（0000 0000 0110） |
| #1023 | 0xC0203FFC~0xC0203FFF | 0x417 | 0x067（0000 0110 0111） |

经检验，步骤二中给出的四张物理页表均正确。