

**操作系统实验报告**

**实验题目**  实验 3 进程的创建

**学生姓名**  余梓俊

**学 号**  2018211991

**专业班级** 计算机科学与技术18-3班

**指导教师**  田卫东

**完成日期**  2020年11月30日

**合肥工业大学 计算机与信息学院**

* 1. **实验目的和任务要求**

练习使用 EOS API 函数 CreateProcess 创建一个进程，掌握创建进程的方法，理解进程和程序的区别。

调试跟踪 CreateProcess 函数的执行过程，了解进程的创建过程，理解进程是资源分配的基本单位。

调试跟踪 CreateThread 函数的执行过程，了解线程的创建过程，理解线程是调度的基本单位。

* 1. **实验原理**

一、进程概念和程序概念最大的不同之处在于：

1、进程是动态的，而程序是静态的。

2、进程有一定的生命期，而程序是指令的集合，本身无“运动”的含义。没有建立进程的程序不能作为1个独立单位得到操作系统的认可。

3、1个程序可以对应多个进程，但1个进程只能对应1个程序。进程和程序的关系犹如演出和剧本的关系。

4、进程和程序的组成不同。从静态角度看，进程由程序、数据和进程控制块（PCB）三部分组成。而程序是一组有序的指令集合。

程序是对于用户而言，也可以叫做软件。进程是对于程序员而言，是操作系统的概念。

进程（Process）是计算机中的程序关于某数据集合上的一次运行活动，是系统进行资源分配和调度的基本单位，是操作系统结构的基础。

程序，是管理方式的一种，是能够发挥出协调高效作用的工具。

二、进程控制块(PCB)是系统为了管理进程设置的一个专门的数据结构。系统用它来记录进程的外部特征，描述进程的运动变化过程。同时，系统可以利用PCB来控制和管理进程，所以说，PCB（进程控制块）是系统感知进程存在的唯一标志。

进程控制块包含三类信息：

1.标识信息。用于唯一地标识一个进程，常常分由用户使用的外部标识符和被系统使用的内部标识号。几乎所有操作系统中进程都被赋予一个唯一的、内部使用的数值型的进程号，操作系统的其他控制表可以通过进程号来交叉引用进程控制表。常用的标识信息包括进程标识符、父进程的标识符、用户进程名、用户组名等。

2.现场信息。用于保留一个进程在运行时存放在处理器现场中的各种信息,任何一个进程在让出处理器时必须把此时的处理器现场信息保存到进程控制块中,而当该进程重新恢复运行时也应恢复处理器现场。常用的现场信息包括通用寄存器的内容、控制寄存器(如PSW寄存器)的内容、用户堆战指针、系统堆饺指针等。

3.控制信息。用于管理和调度一个进程。常用的控制信息包括：l)进程的调度相关信息,如进程状态、等待事件和等待原因、进程优先级、队列指引元等2)进程组成信息,如正文段指针、数据段指针:引进程间通信相关信息,如消息队列指针、信号量等互斥和同步机制4)进程在辅存储器内的地址5)CPU资源的占用和使用信息,如时间片余量、进程己占用CPU的时间、进程己执行的时间总和,记账信息6)进程特权信息,如在内存访问和处理器状态方面的特权7)资源清单,包括进程所需全部资源、已经分得的资源,如主存资源、I/0设备、打开文件表等。

进程的创建来源于以下四个事件：

1.提交一个批处理作业。

2.在终端上交互式的登录。

3.操作系统创建一个服务进程。

4.存在的进程孵化(spawn)新的进程。

进程的创建过程如下描述：

1.在主进程表中增加一项,并从PCB池中取一个空白PCB。

2.为新进程的进程映像中的所有成分分配地址空间。对于进程孵化操作还需要传递环境变量,构造共享地址空间。

3.为新进程分配资源,除内存空间外,还有其它各种资源。

4.查找辅助存储器,找到进程正文段并装入到正文区。

5.初始化进程控制块,为新进程分配一个唯一的进程标识符,初始化PSW。

6.把进程加入某一就绪进程队列,或直接将进程投入运行。

7.通知操作系统的某些模块,如记账程序、性能监控程序。

三、线程与进程的区别

1）进程是一个独立的运行环境，它可以被看作是一个程序或者一个应用。而线程是在进程中执行的一个任务。

2）进程是操作系统进行资源分配的基本单位，而线程是操作系统进行调度的基本单位。

3）进程让操作系统的并发性成为可能，而线程让进程的内部并发成为可能。

线程的创建：

1）继承Thread类;

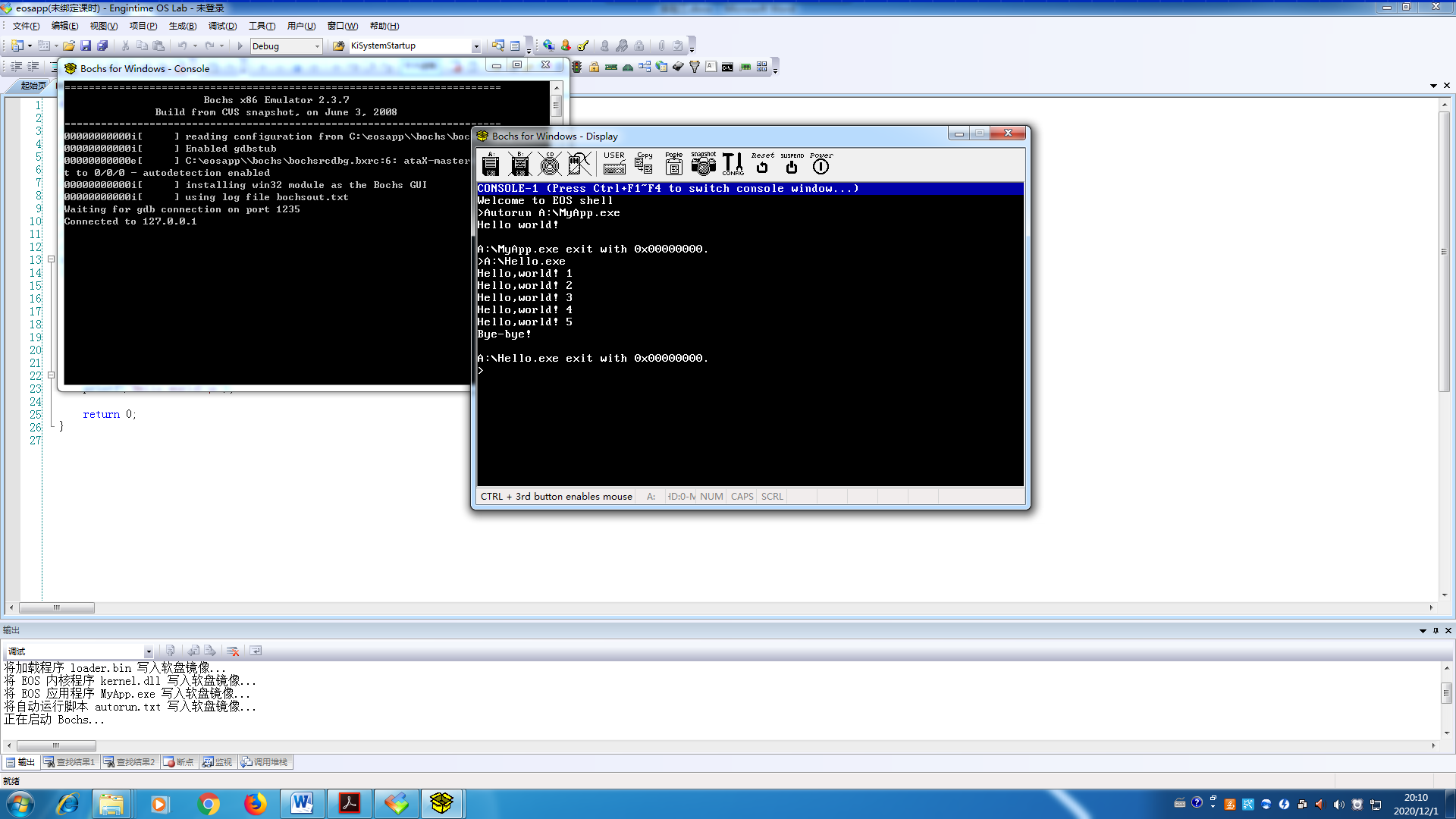
2）实现Runnable接口;

3）实现Callable接口;

4) 通过线程池创建线程；

* 1. **实验内容**

1、使用控制台命令创建一个 EOS 应用程序的进程



2、通过编程的方式让应用程序创建另一个应用程序的进程

使用 OS Lab 打开本实验文件夹中的 NewProc.c 文件（将此文件拖动到 OS Lab 窗口中释放即可），仔细阅读此文件中的源代码和注释，main 函数的流程图可以参见图 11-3。

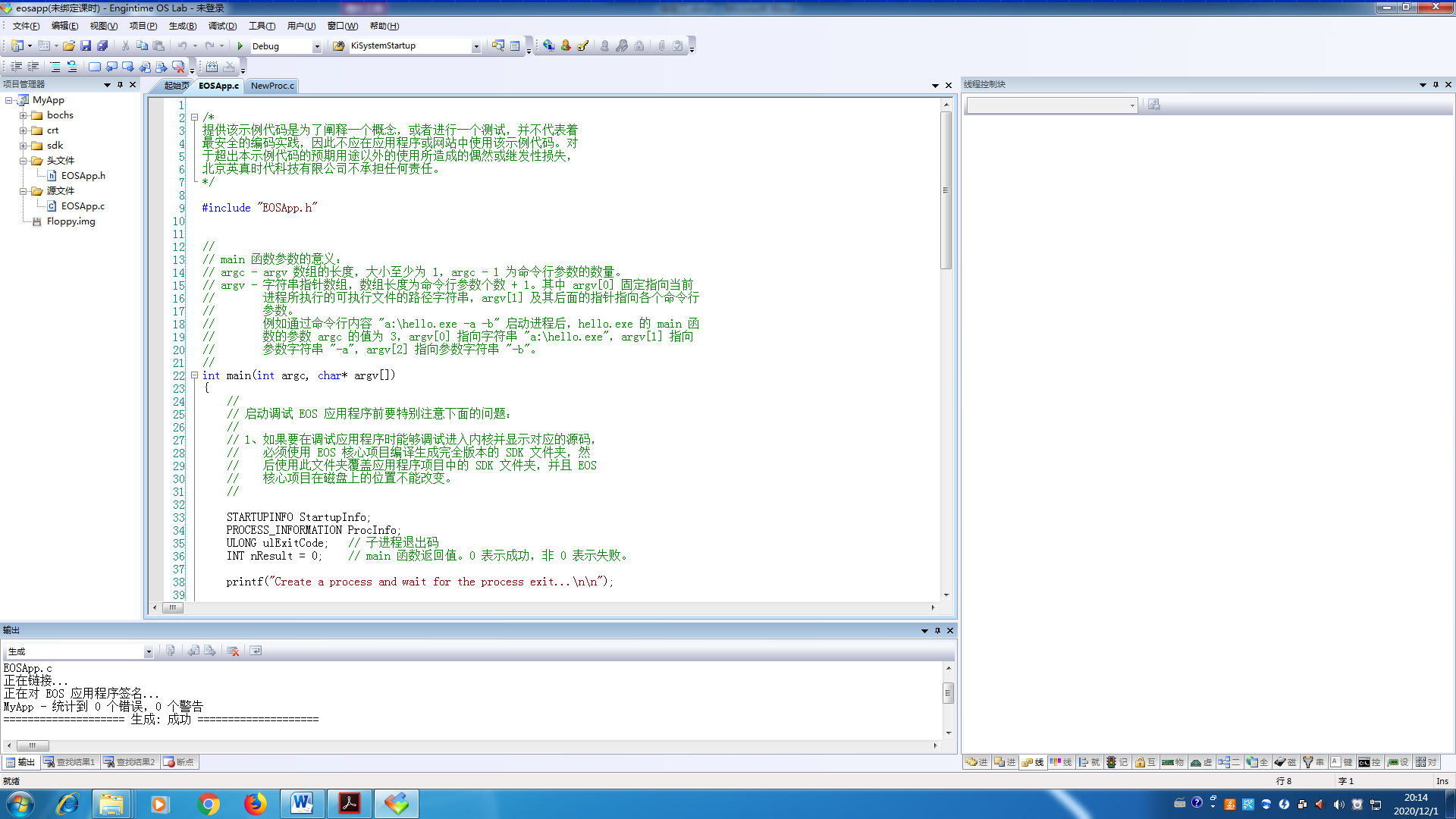
按照下面的步骤查看让一个应用程序创建另一个应用程序的进程的执行结果：

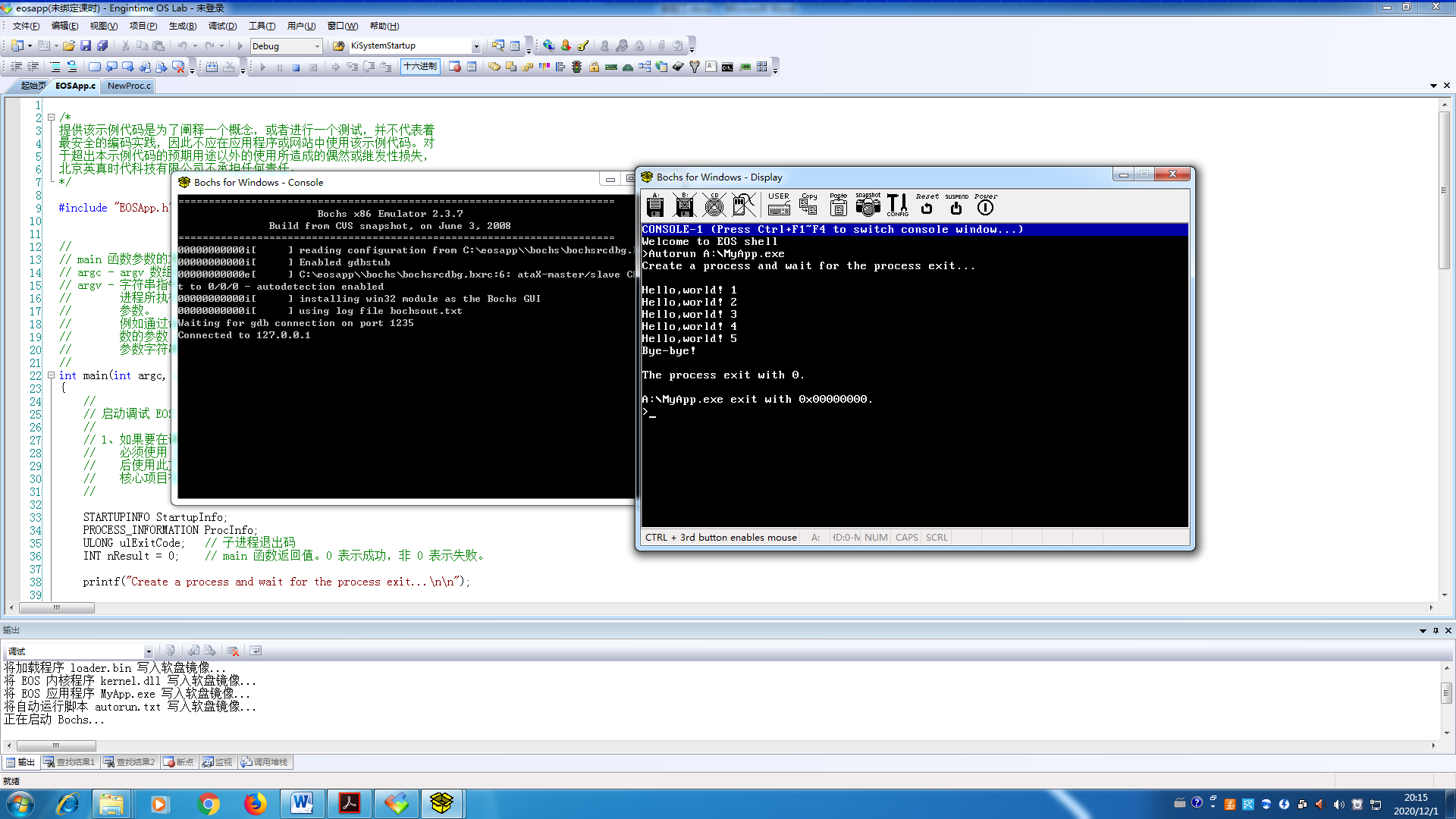
1. 使用 NewProc.c 文件中的源代码替换之前创建的 EOS 应用程序项目中 EOSApp.c 文件的源代码。

2. 按 F7 生成修改后的 EOS 应用程序项目。

3. 按 F5 启动调试。

4. 激活虚拟机窗口查看应用程序输出的内容，如图 11-2 所示。结合图 11-2，可以看到父进程（EOSApp.exe）首先开始执行并输出内容，当父进程创建了子进程（Hello.exe）后，子进程开始执行并输出内容，待子进程结束后父进程再继续执行。





3、从应用程序的角度理解进程的创建过程

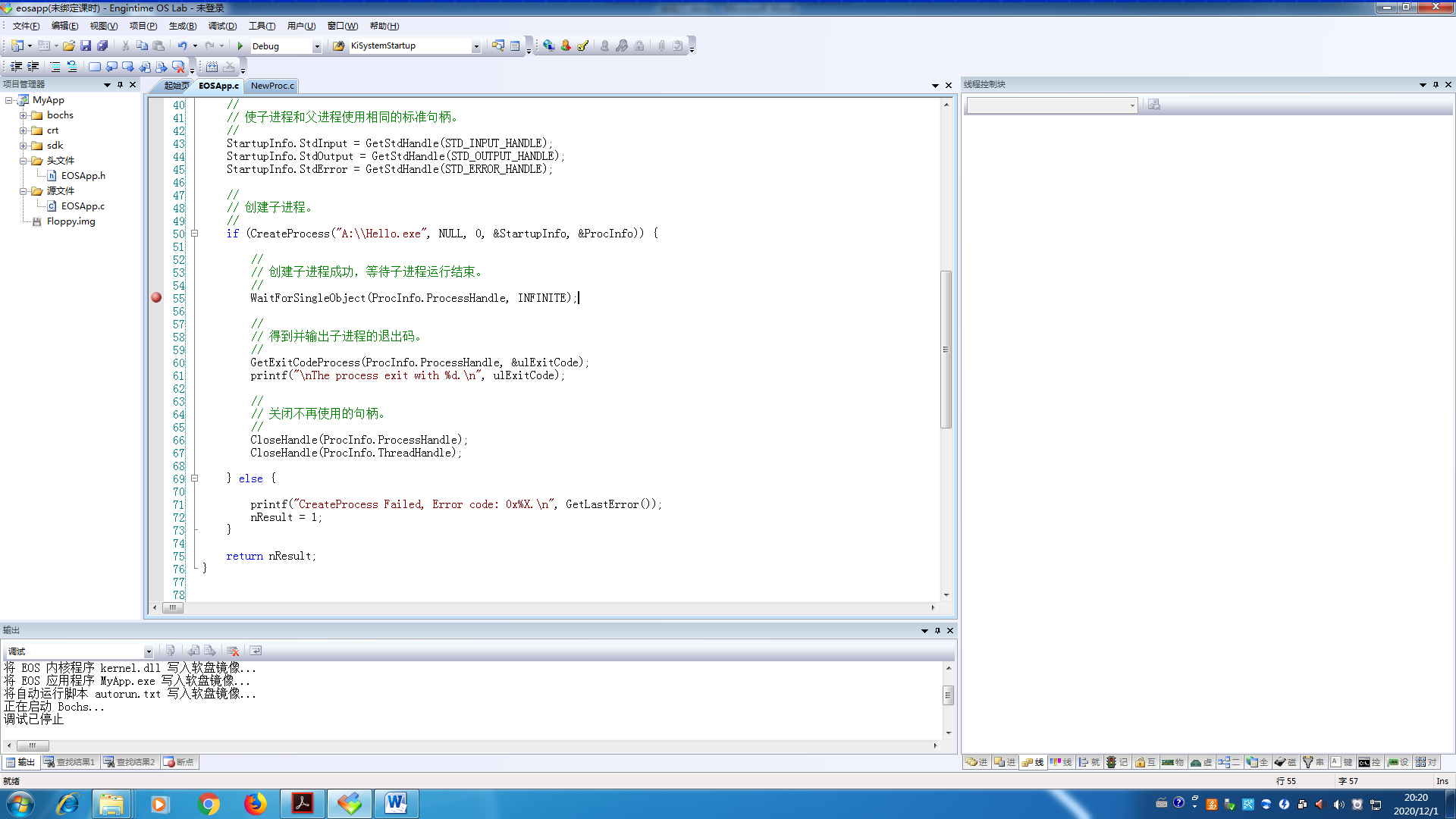
按照下面的步骤调试，从应用程序的角度理解进程的创建过程

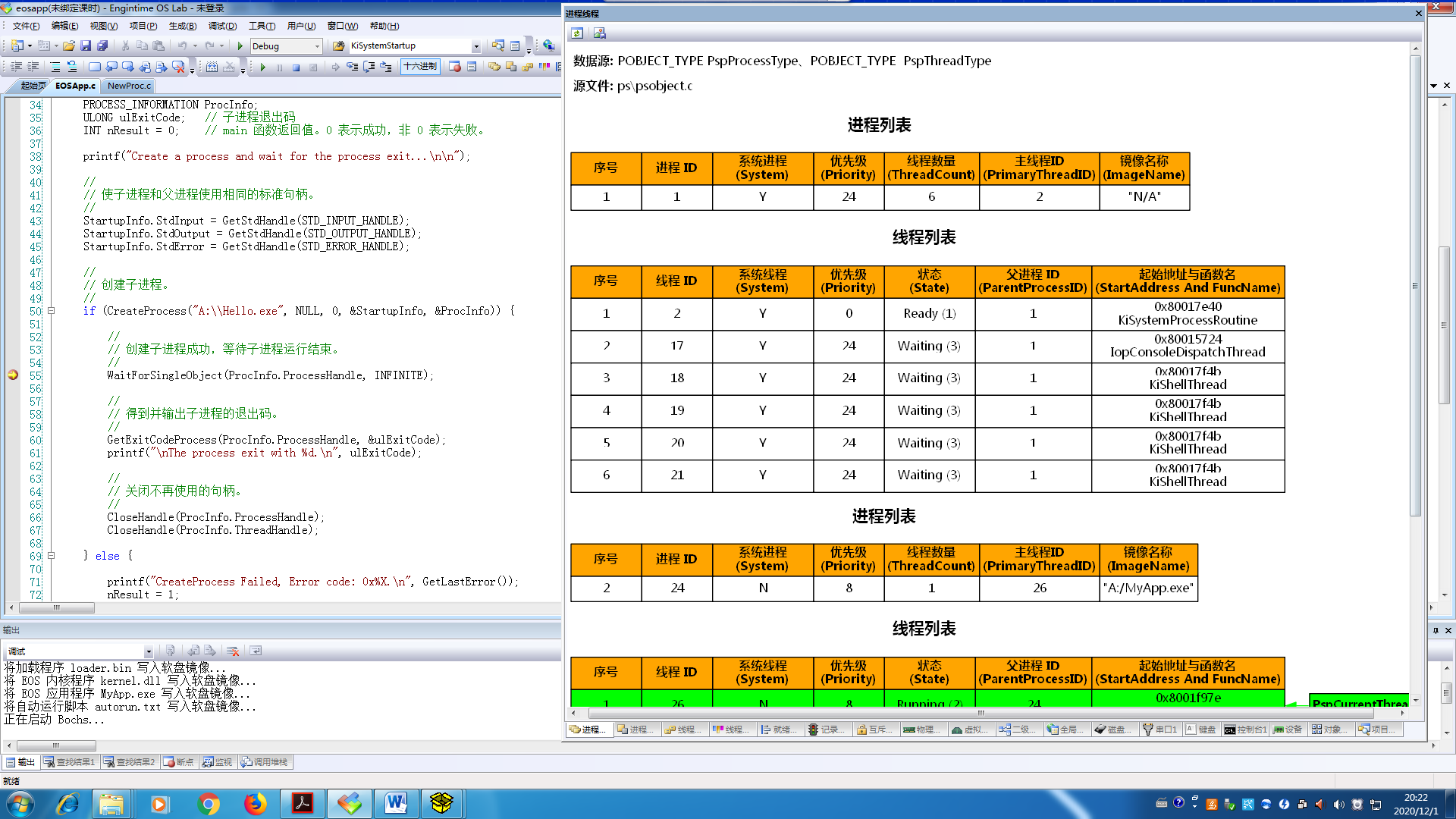
1. 打开EOS 应用程序的EOSApp.c 文件 ，在 main 函数中调用WaitForSingleObject 函数的代码行（第

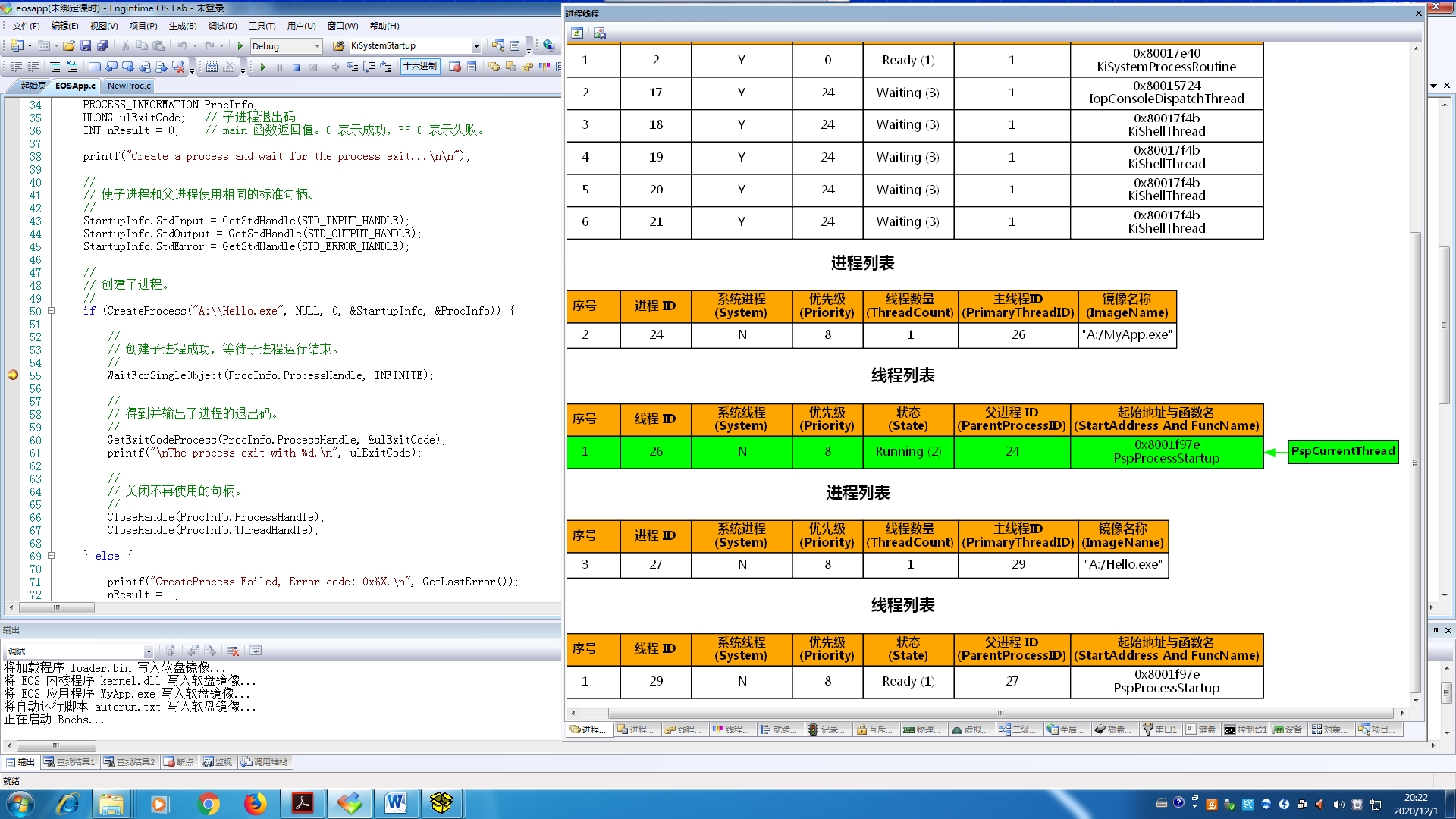
54 行）添加一个断点。

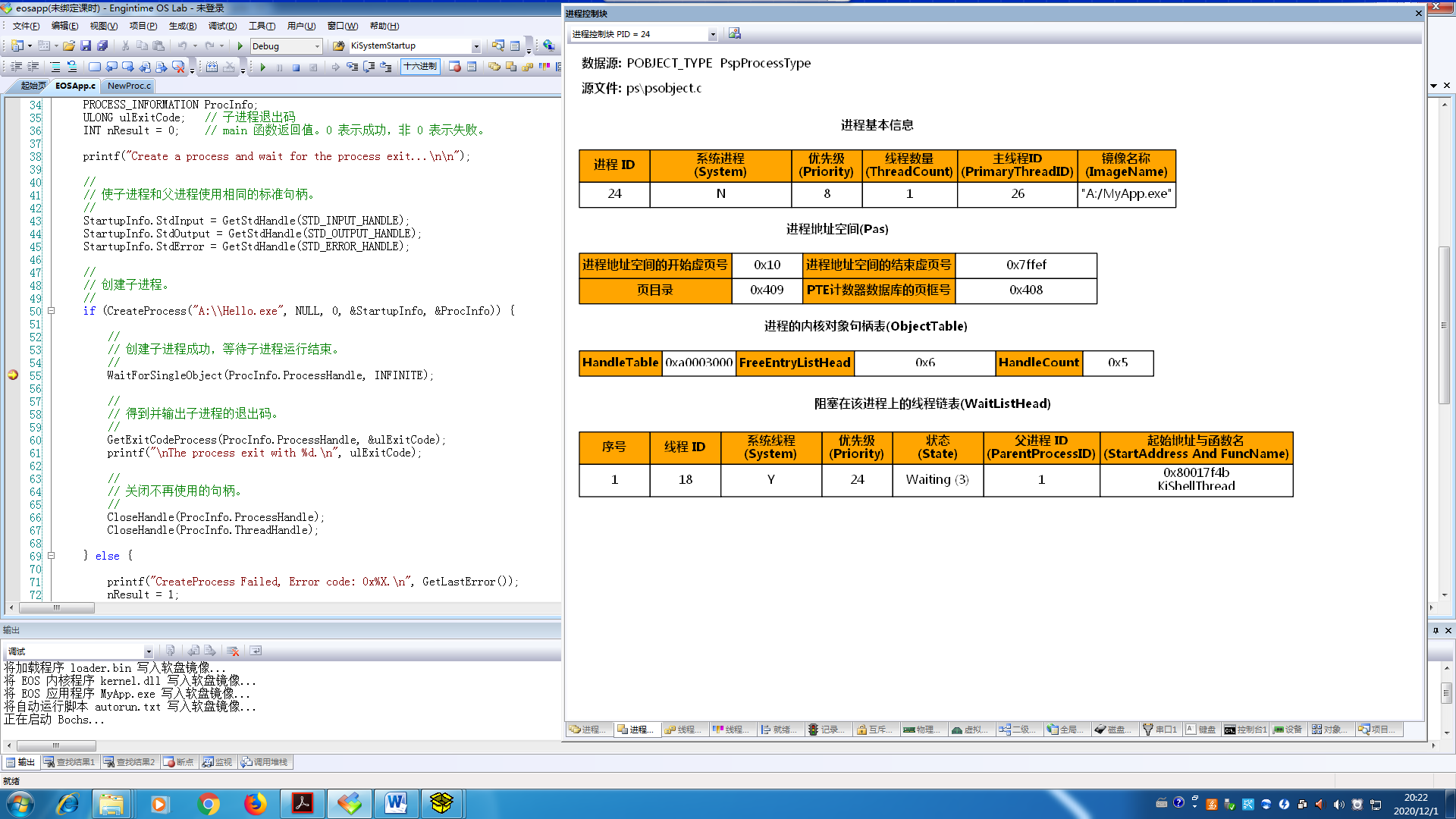
2. 按 F5 启动调试，会在刚刚添加的断点处中断。

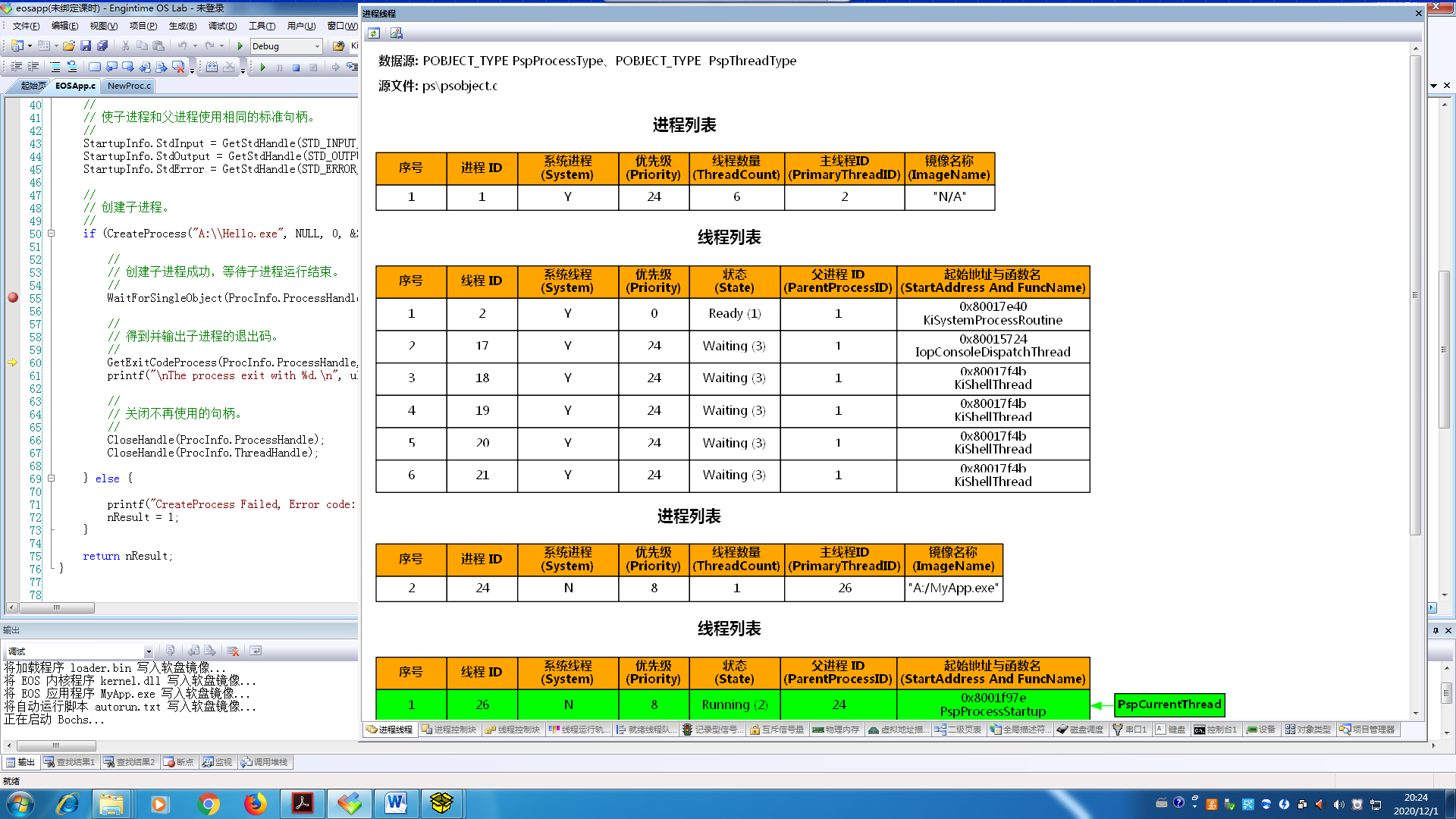
3. 选择“调试”菜单“窗口”中的“进程线程”菜单打开“进程线程”窗口，点击此窗口工具栏上的“刷新”按钮，会显示如图 11-4 所示的内容。可以看到当前系统中有三个进程，其中第一个是系统进程，镜像名称为空；另外两个是刚刚创建的应用程序的进程，镜像名称分别为“A:\EOSApp.exe”（会根据读者创建的项目名称变化）和“A:\Hello.exe”。其中，A:\EOSApp.exe进程是父进程，其主线程处于运行状态（正准备调用 WaitForSingleObject 函数，但是命中了断点）；A:\Hello.exe 进程是子进程，其主线程处于就绪状态，当父进程的主线程通过调用WaitForSingleObject 函数进入阻塞状态让出处理器后，这个处于就绪状态的线程就会占用处理器开始运行。这两个应用程序的进程和线程的优先级都为 8。







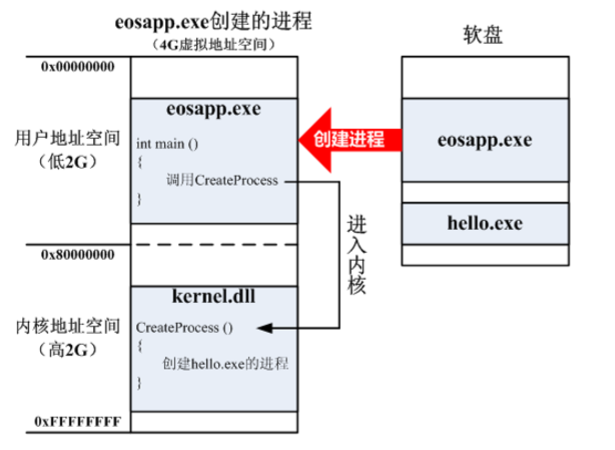






4、从内核的角度理解进程的创建过程

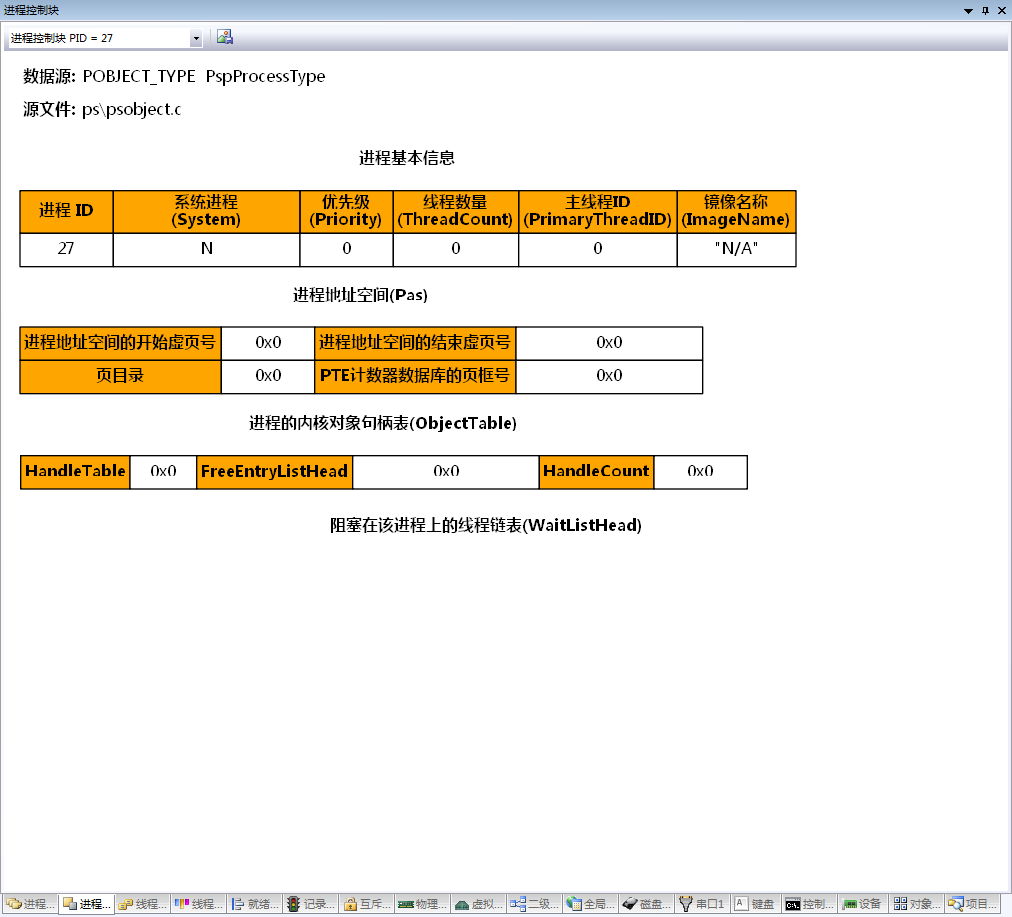
（1）调试 CreateProcess 函数



创建进程最主要的操作就是创建进程控制块(PCB),并初始化其中的各种信息（也就是为进程分配各种资源）。所以在 PsCreateProcess 函数中首先调用了 PspCreateProcessEnvironment 函数来创建进程控制块。

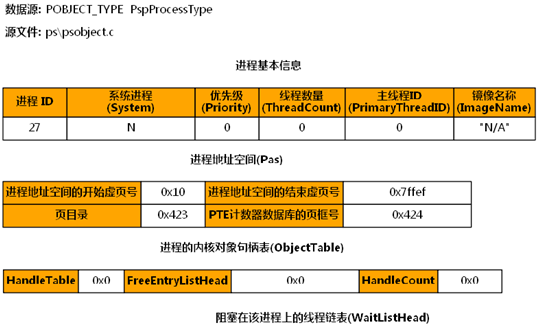
1. 调试进程控制块

“进程控制块”窗口，在该窗口工具栏的下拉列表中选择“进程控制块 PID=27”，可以看到该进程控制块中各个成员变量的值。由于目前只是新建了进程控制块，还没有为其中的成员变量赋值，所以值都为 0。

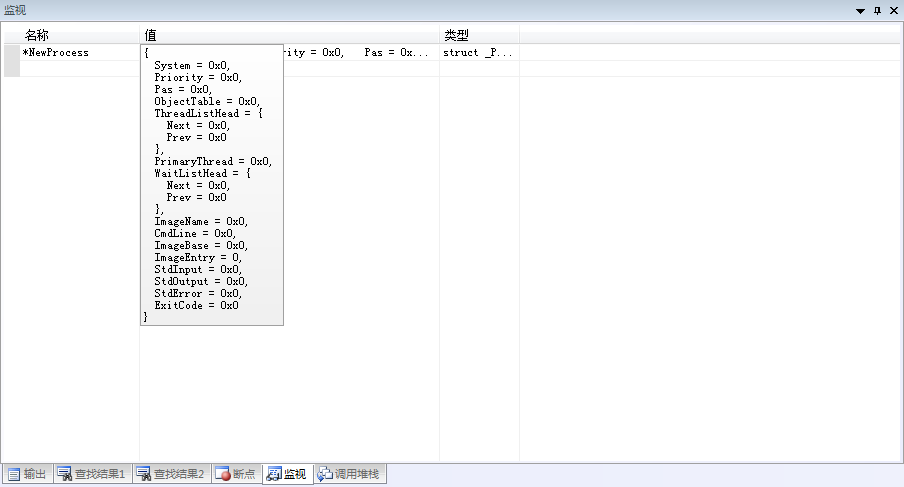


（4）初始化进程控制块中各个成员变量

在“进程控制块”窗口中，选择“进程控制块 PID=27”，可以看到该进程控制块的“进程地址空间”的值已经不再是 0。说明已经初始化了进程的 4G 虚拟地址空间。还可以在“监视”窗口中查看进程控制块的成员变量 Pas 的值已经不再是 0。

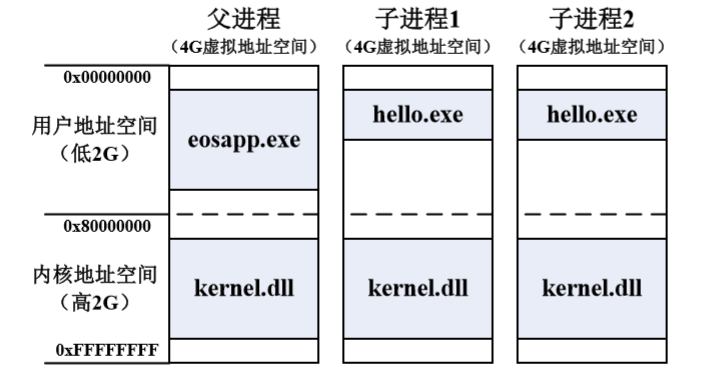


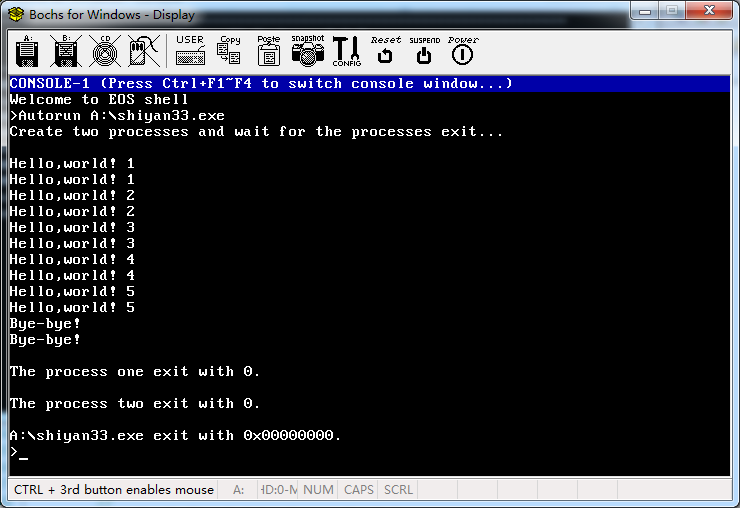
（5）当从 PspCreateProcessEnvironment 函数返回到 PsCreateProcess 函数后，停止按 F10。此时“监视”窗口中已经不能再显示表达式\*NewProcess 的值了，在 PsCreateProcess 函数中是使用ProcessObject 指针指向进程控制块的，所以将表达式\*ProcessObject 添加到“监视”窗口中就可以继续观察新建进程控制块中的信息。



5、通过编程的方式创建一个应用程序的多个进程

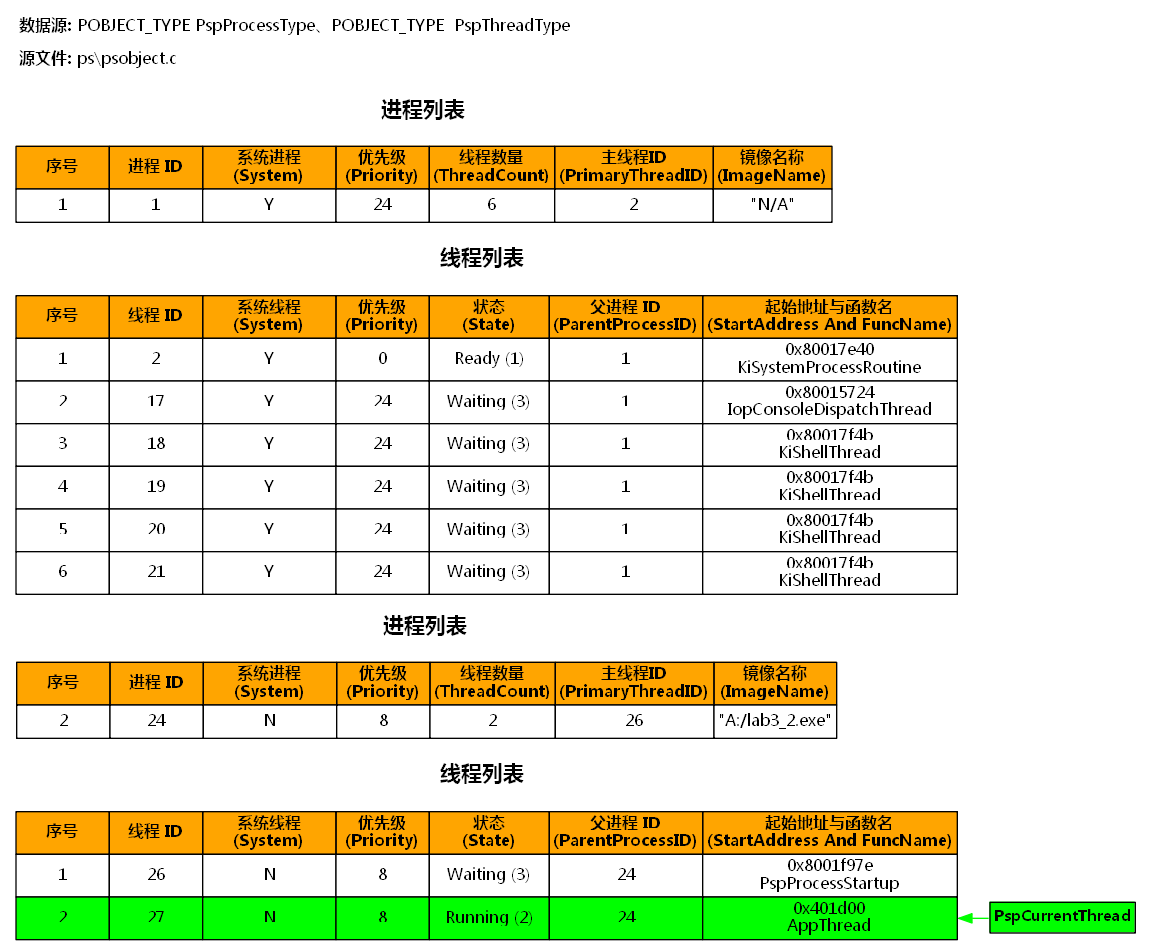
（1）多个进程并发时，EOS 操作系统中运行的用户进程：



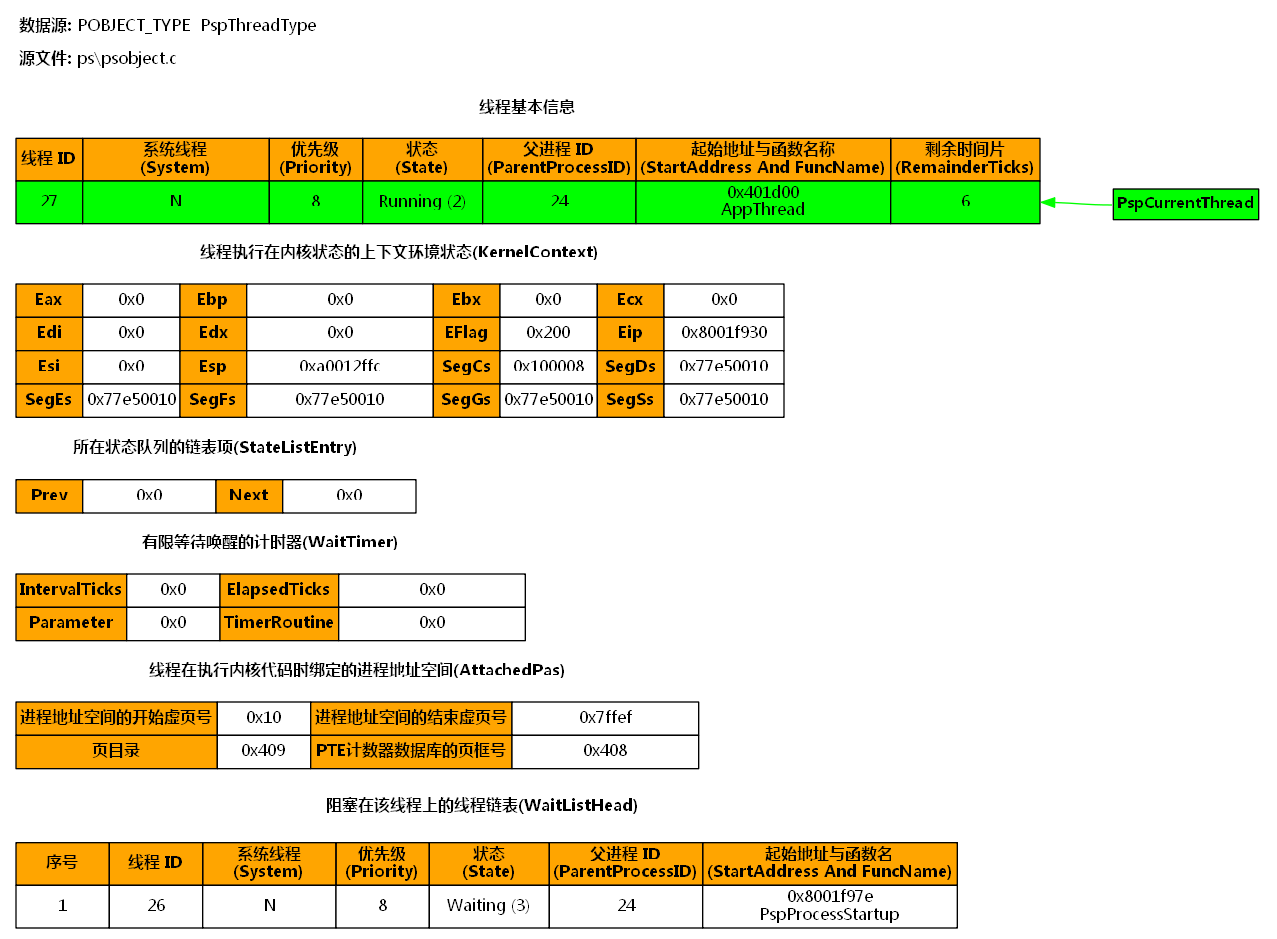


6、在应用程序进程中创建一个工作线程（Worker Thread）

（1）刷新“进程线程”窗口，可以看到如图所示的内容。在应用程序进程的线程列表中，包含两个线程，一个是应用程序的主线程，处于阻塞状态，说明其正在等待工作线程结束；另一个是在应用程序中通过调用 CreateThread 函数创建的工作线程，处于运行状态，说明其正在执行线程函数并命中了断点。



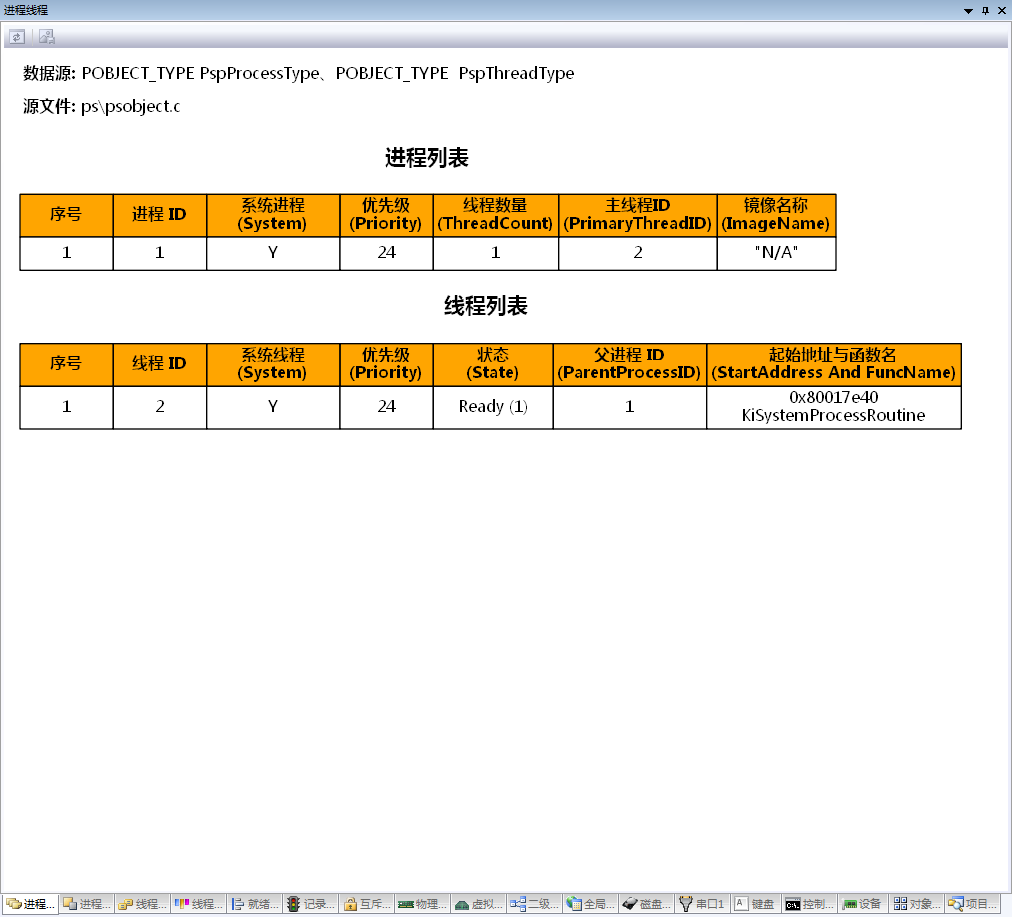
（2）在“线程控制块”窗口工具栏的组合框中选择“线程控制块 TID=27”的选项，可以查看工作线程的详细信息，如图 11-16 所示，包括：线程的基本信息、线程执行在内核状态的上下文环境状态、所在状态队列的链表项（由于该线程处于运行态，也就不在任何状态队列中，所以 Prev 和 Next 指针的值都为 0）、有限等待唤醒的计时器、线程在执行内核代码时绑定的地址空间、阻塞在该线程上的线程链表。其中，阻塞在该线程上的线程链表中显示了应用程序的主线程阻塞在该工作线程上，也就是主线程在等待该工作线程结束。



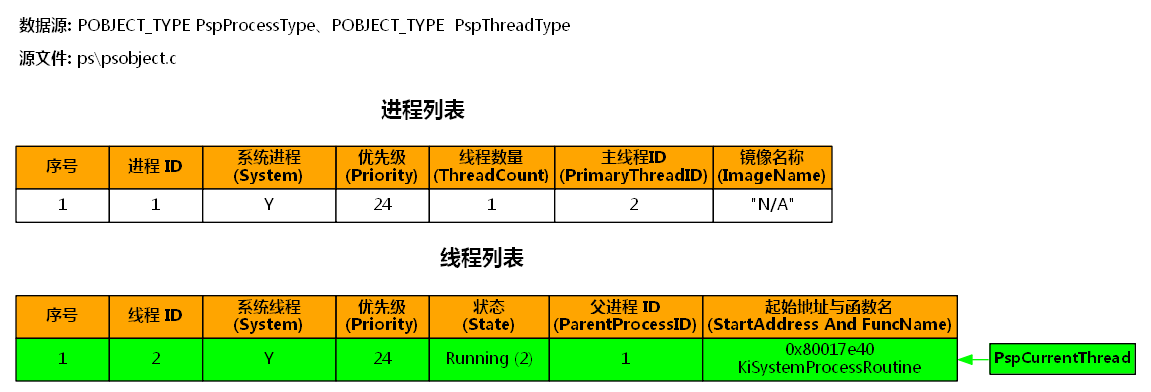
7、系统线程的创建

（1）通过调试 EOS 内核的初始化过程来理解几个重要的系统线程的创建过程，这些线程包括：系统初始化线程、控制台派遣线程、控制台线程。

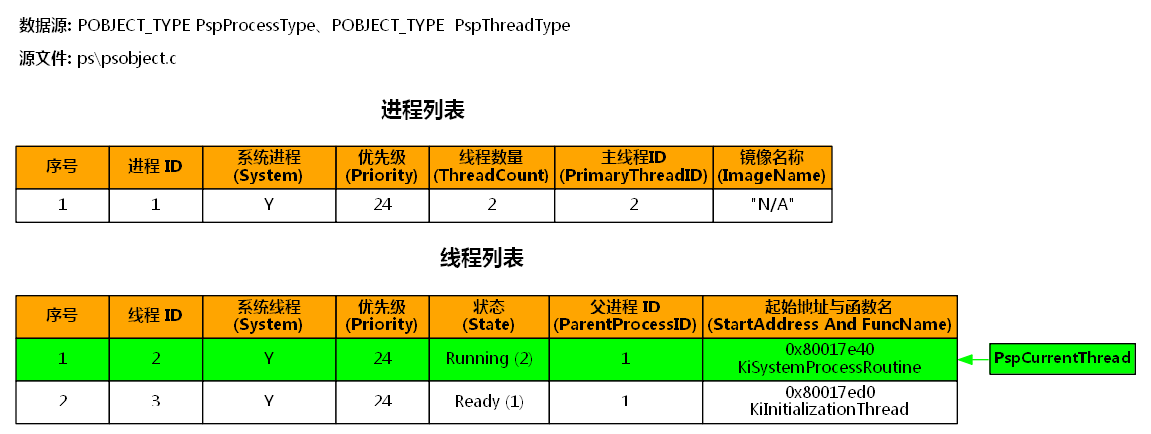
（2）刷新“进程线程”窗口，显示如图所示的内容。可以看到当前系统中，只创建了一个线程， 该线程就是由第 66 行的PsCreateSystemProcess 函数在创建系统进程后，为系统进程创建的第一个子线程，同时也是系统进程的主线程。



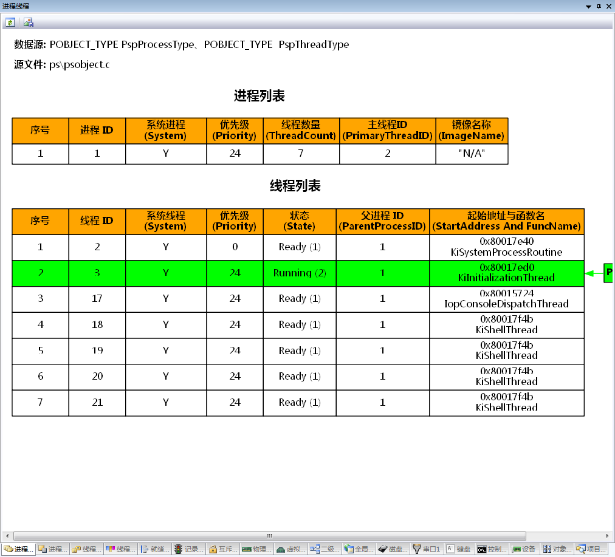
（3）由于第一个系统线程处于就绪状态，并且是当前系统中唯一的线程，所以在执行第 72 行的线程调度函数后，就会开始执行其线程函数。接下来，在其线程函数中添加一个断点：打开.ke\sysproc.c 文件，在 KiSystemProcessRoutine 函数中的（第 123 行代码处）添加一个断点。按 F5 继续调试，在断点处中断。刷新“进程线程”窗口，可以看到系统中唯一的线程处于运行状态，说明该线程正在执行其线程函数并命中了断点。



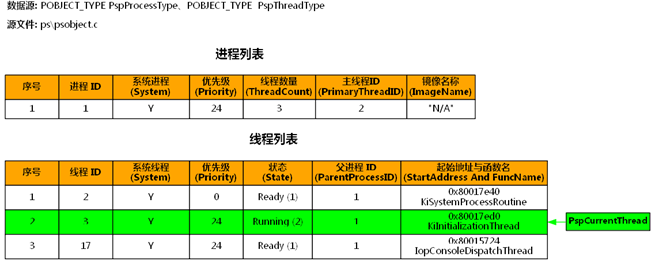
（4）当前中断位置处的代码行用于创建一个新的系统线程，并使用该新建线程继续进行操作系统的初始化工作。所以，按 F10 单步调试一次后，刷新“进程线程”窗口，会显示如图所示的内容，可以看到当前系统中又多出了一个线程，并处于就绪状态。



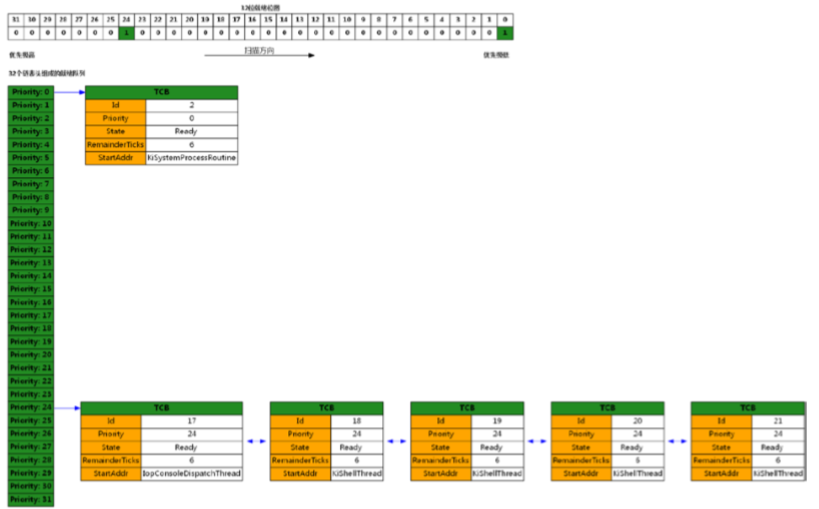
（5）由于第一个系统线程会使用第 140 行的代码将自己的优先级降为 0，退化为空闲线程。这就会让刚刚新建的还处于就绪状态的系统初始化线程抢占处理器开始运行。接下来，在初始化线程的线程函数中添加一个断点：打开 ke\sysproc.c 文件，在 KiInitializationThread 函数中（第 160 行代码处）添加一个断点。按 F5 继续调试，在断点处中断。刷新“进程线程”窗口，可以看到系统初始化线程处于运行状态，说明该线程正在执行其线程函数并命中了断点。



（6）按 F10 单步调试执行 IoInitializeSystem2 函数，完成基本输入输出的初始化。刷新“进程线程” 窗口，显示如图 11-19 所示的内容，可以看到当前系统中已经创建了控制台派遣线程，并处于就绪状态。

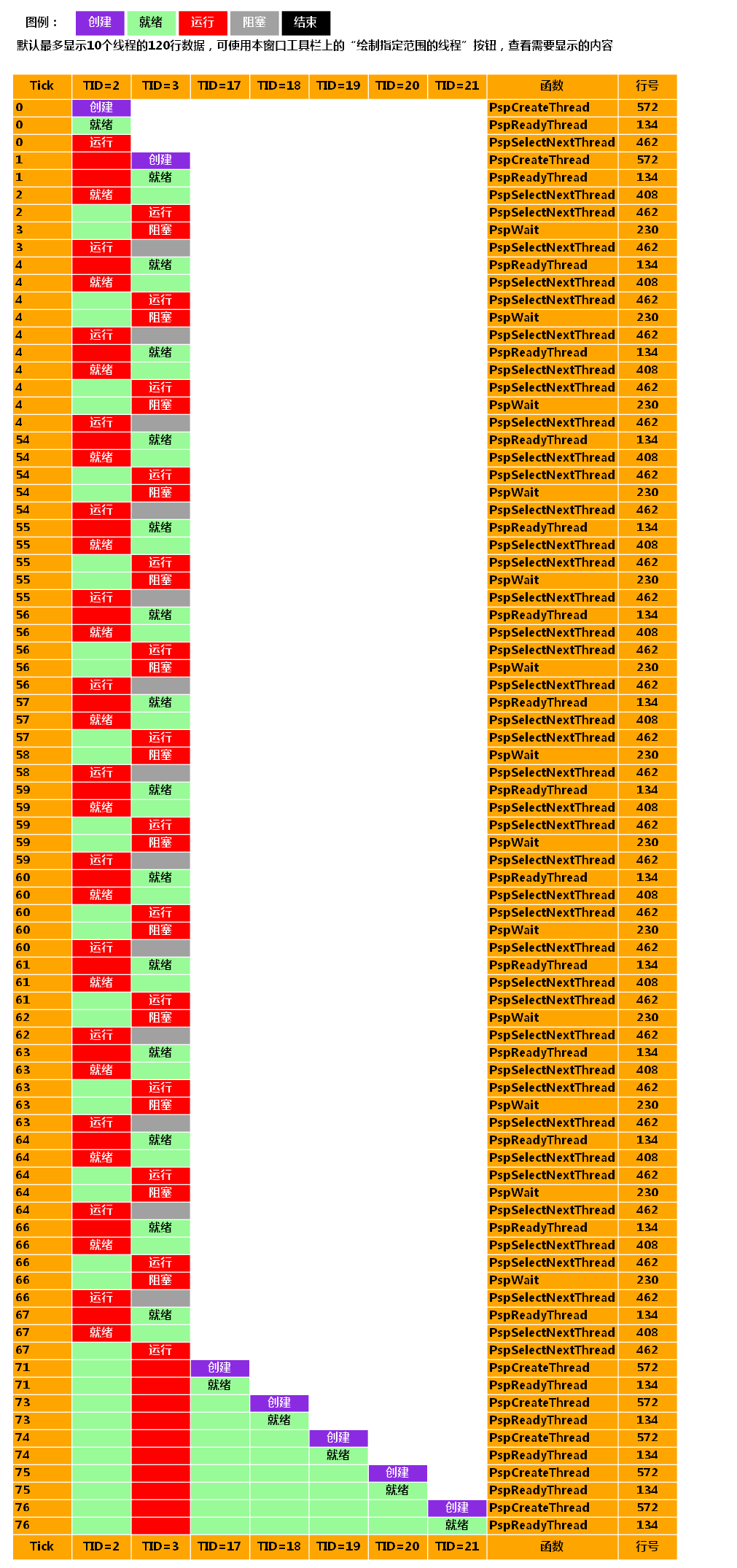


（7）于第 168 行的 for 语句会循环创建四个控制台线程，所以，读者可以在 184 行代码处添加一个断点，按 F5 继续调试。待命中断点后，刷新“进程线程”窗口，会显示如图 11-20 所示的内容， 可以看到当前系统已经完成了五个控制台线程的创建，并且所有的控制台线程都处于就绪状态。



（8）选择“调试”菜单“窗口”中的“就绪线程队列”，打开“就绪线程队列”窗口。点击此窗口工具栏上的“刷新”按钮，会显示如图 11-20 所示的内容。顾名思义，就绪线程队列用于管理那些处于就绪状态的线程控制块，可以看到优先级为 0 的空闲线程，以及优先级为 24 的控制台派遣线程和四个控制台线程分别在其优先级对应的就绪队列中，而当前处于运行态的系统初始化线程（线程 ID 为 2）就不在就绪队列中。

（9）选择“调试”菜单“窗口”中的“线程运行轨迹”，打开“线程运行轨迹”窗口。点击此窗口工具栏上的“刷新”按钮，会显示如图 11-21 所示的内容，可以查看这些系统线程的状态转换过程和运行轨迹。注意，系统初始化线程（TID=3）在初始化的过程中会由于等待一些硬件设备的响应，从而频繁进入阻塞状态。



* 1. **实验的思考与问题分析**

1. 在源代码文件 NewTwoProc.c 提供的源代码基础上进行修改，要求使用 hello.exe 同时创建 10 个进程。提示：可以使用 PROCESS\_INFORMATION 类型定义一个有 10 个元素的数组，每一个元素对应一个进程。使用一个循环创建 10 个子进程，然后再使用一个循环等待 10 个子进程结束，得到退出码后关闭句柄。

答：

if (CreateProcess("A:\\Hello.exe", NULL, 0, &StartupInfo, &ProcInfoOne)

&& CreateProcess("A:\\Hello.exe", NULL, 0, &StartupInfo, &ProcInfoTwo)

&& CreateProcess("A:\\Hello.exe", NULL, 0, &StartupInfo, &ProcInfo3)……

&& CreateProcess("A:\\Hello.exe", NULL, 0, &StartupInfo, &ProcInfo10)) {

WaitForSingleObject(ProcInfoOne.ProcessHandle, INFINITE);

WaitForSingleObject(ProcInfoTwo.ProcessHandle, INFINITE);

WaitForSingleObject(ProcInfo3.ProcessHandle, INFINITE);……

WaitForSingleObject(ProcInfo10.ProcessHandle, INFINITE);

GetExitCodeProcess(ProcInfoOne.ProcessHandle, &ulExitCode);……

GetExitCodeProcess(ProcInfo3.ProcessHandle, &ulExitCode);

printf("\nThe process 3 exit with %d.\n", ulExitCode);……

GetExitCodeProcess(ProcInfo10.ProcessHandle, &ulExitCode);

printf("\nThe process 10 exit with %d.\n", ulExitCode);

CloseHandle(ProcInfoOne.ProcessHandle);……

CloseHandle(ProcInfoTwo.ThreadHandle);

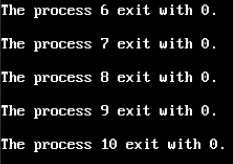
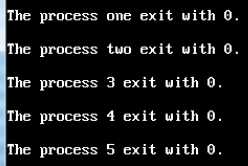
CloseHandle(ProcInfo3.ProcessHandle);

CloseHandle(ProcInfo3.ThreadHandle);……

CloseHandle(ProcInfo10.ProcessHandle);

CloseHandle(ProcInfo10.ThreadHandle);





1. 在 PsCreateProcess 函数中调用了 PspCreateProcessEnvironment 函数后又先后调用了PspLoadProcessImage 和 PspCreateThread 函数，学习这些函数的主要功能。能够交换这些函数被调用的顺序吗？思考其中的原因。

答：

PspCreateProcessEnvironment的主要功能是创建进程控制块 并且为进程创建了地址空间和分配了句柄表oPspLoadProcessImage是将进程的可执行映像加载到了进程的地址空间中。PspCreateThread创建了进程的主线程。这三个函数被调用的顺序是不能够改变的 就向上面描述的 加载可执行映像之前必须已经为进程创建了地址空间 这样才能够确定可执行映像可以被加载到内存的什么位置 在创建主线程之前必须已经加载了可执行映像这样主线程才能够知道自己要从哪里开始执行，执行哪些指令。因此不能交换他们的顺序。