

**操作系统实验报告**

**实验题目**  实验6 时间片轮转调度

**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12月14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**

调试 EOS 的线程调度程序，熟悉基于优先级的抢先式调度。

为 EOS 添加时间片轮转调度，了解其它常用的调度算法。

1. **实验原理**

基于优先级的抢占式调度：

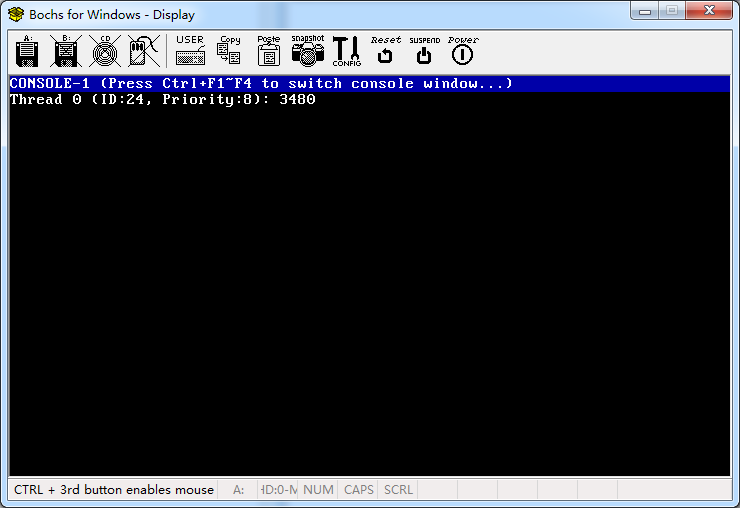
(1) 每个任务赋予唯一的一个优先级（有些操作系统可以动态地改变任务的优先级）；

(2) 假如有几个任务同时处于就绪状态，优先级最高的那个将被运行；

(3) 只要有一个优先级更高的任务就绪，它就可以中断当前优先级较低的任务的执行；

1. **实验内容**

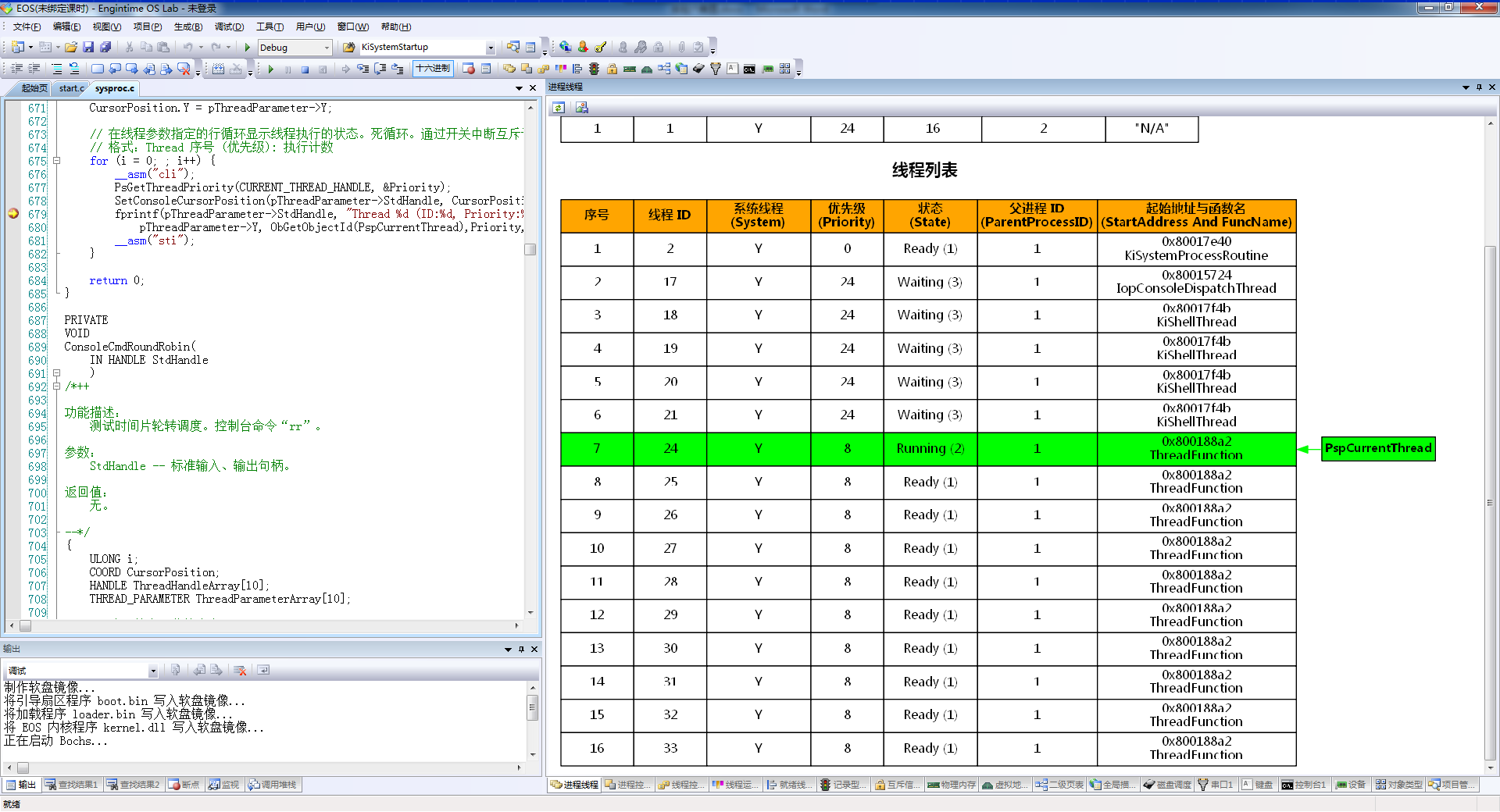
1、执行控制台命令“rr”，查看其在没有时间片轮转调度时的执行效果：



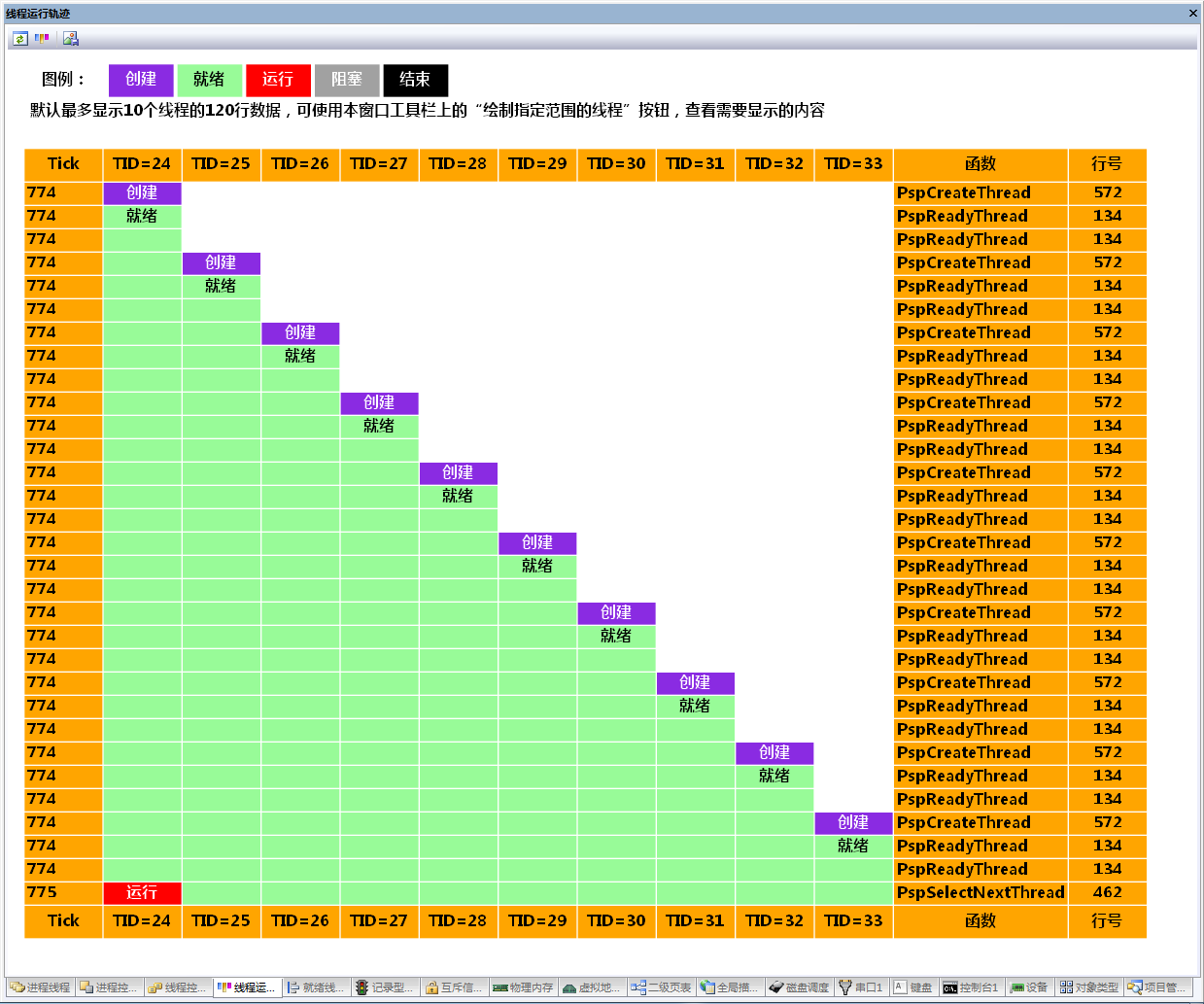
2、调试线程调度程序

（1）调试当前线程不被抢先的情况

结束之前的调试。在 ke/sysproc.c 文件的 ThreadFunction 函数中，调用 fprintf 函数的代码行（第 679 行）添加一个断点。按 F5 启动调试。待 EOS 启动完毕，在 EOS 控制台中输入命令“rr”后按回车。“rr”命令开始执行后，会在断点处中断。刷新“进程线程”窗口，可以看到如图 14-2 所示的内容。其中，从 ID 为 24 到 ID 为 33 的线程是“rr”命令创建的 10 个优先级为 8 的线程，ID 为 24 的线程处于运行状态，其它的 9 个线程处于就绪状态。



（2）在“线程运行轨迹”窗口点击其工具栏上的“绘制指定范围线程”按钮，输入起始线程 ID 为 24和结束线程 ID 为 33（需根据当前实际创建的线程 ID 进行调整），点击“绘制”按钮，可以查看这 10 个线程的运行状态和调度情况，如图所示。



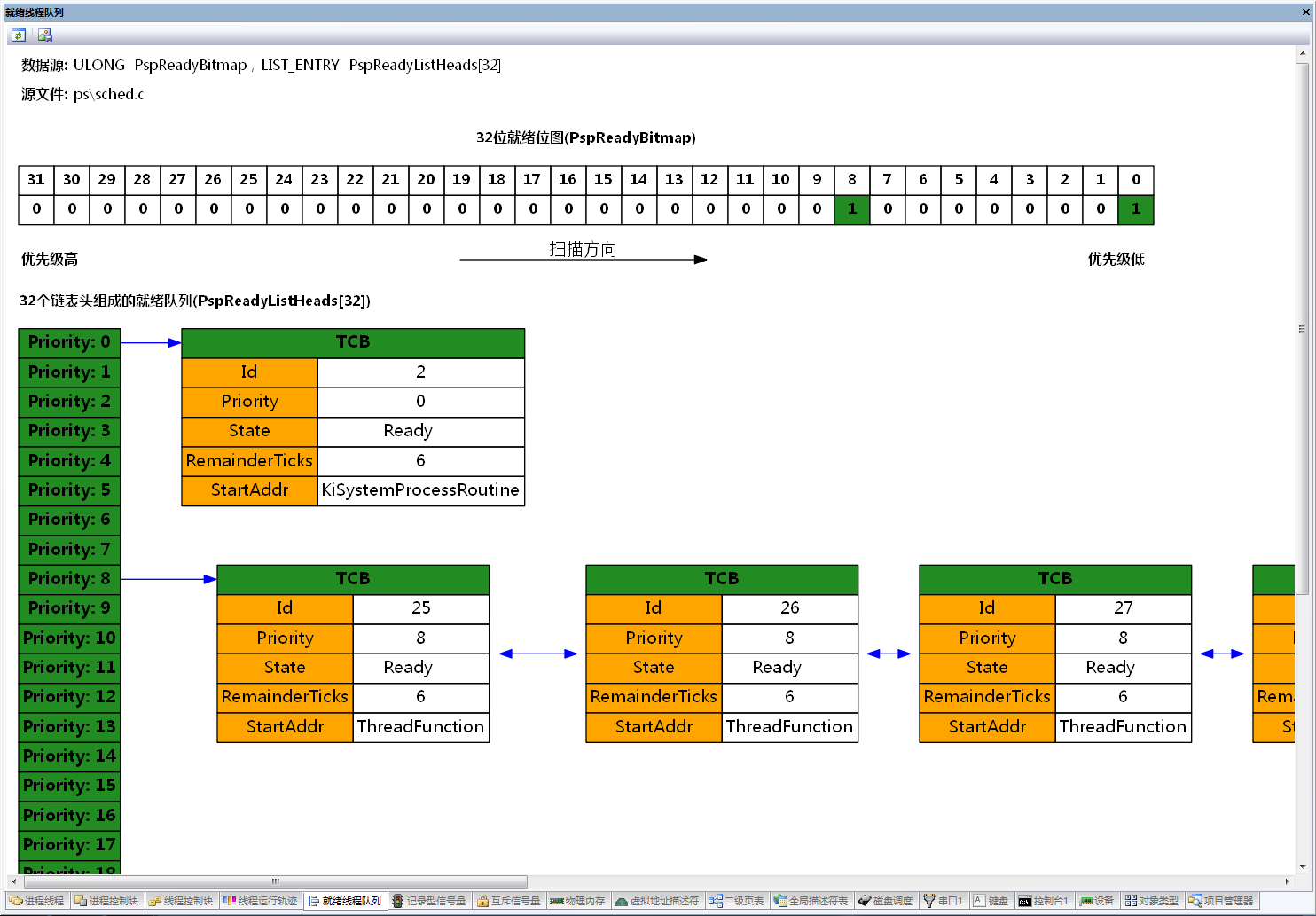
刷新“就绪线程队列”窗口，可以看到在就绪位图中，第 0 位和第 8 位为 1，其它位都为 0，相对应的，在就绪队列中只有优先级为 0 和优先级为 8 的就绪队列中挂接了处于就绪状态的线程，其它优先级的就绪队列都为空，如图 14-4 所示。其中，优先级为 0 的线程是空闲线程，9 个优先级为 8 的线程是从 ID 为 25 到 ID 为 33 的线程，而 ID 为 24 的线程此时正处于运行状态，所以不在优先级为 8 的就绪队列中。

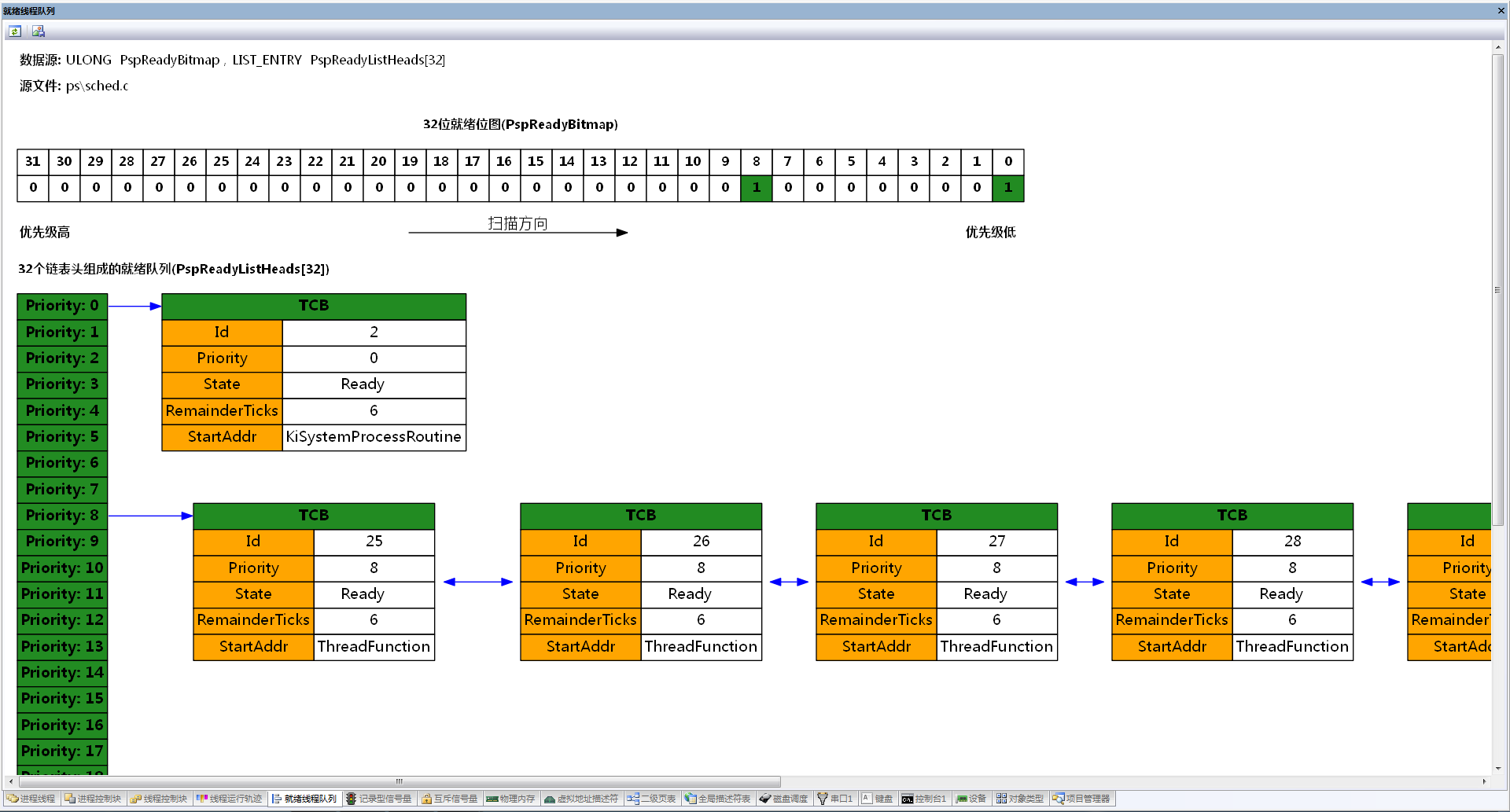
查看 ThreadFunction 函数中变量 pThreadParameter->Y 的值应该为 0，说明正在调试的是第 0个新建的线程。

激活虚拟机窗口，可以看到第 0 个新建的线程还没有在控制台中输出任何内容，原因是 fprintf函数还没有执行。

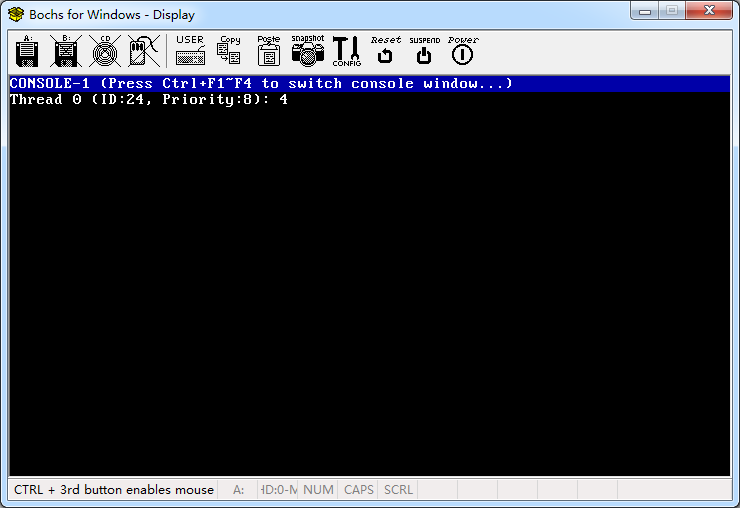
激活 OS Lab 窗口后按 F5 使第 0 个新建的线程继续执行，又会在断点处中断。再次激活虚拟机窗口，可以看到第 0 个新建的线程已经在控制台中输出了第一轮循环的内容。可以多按几次 F5 查看每轮循环输出的内容。

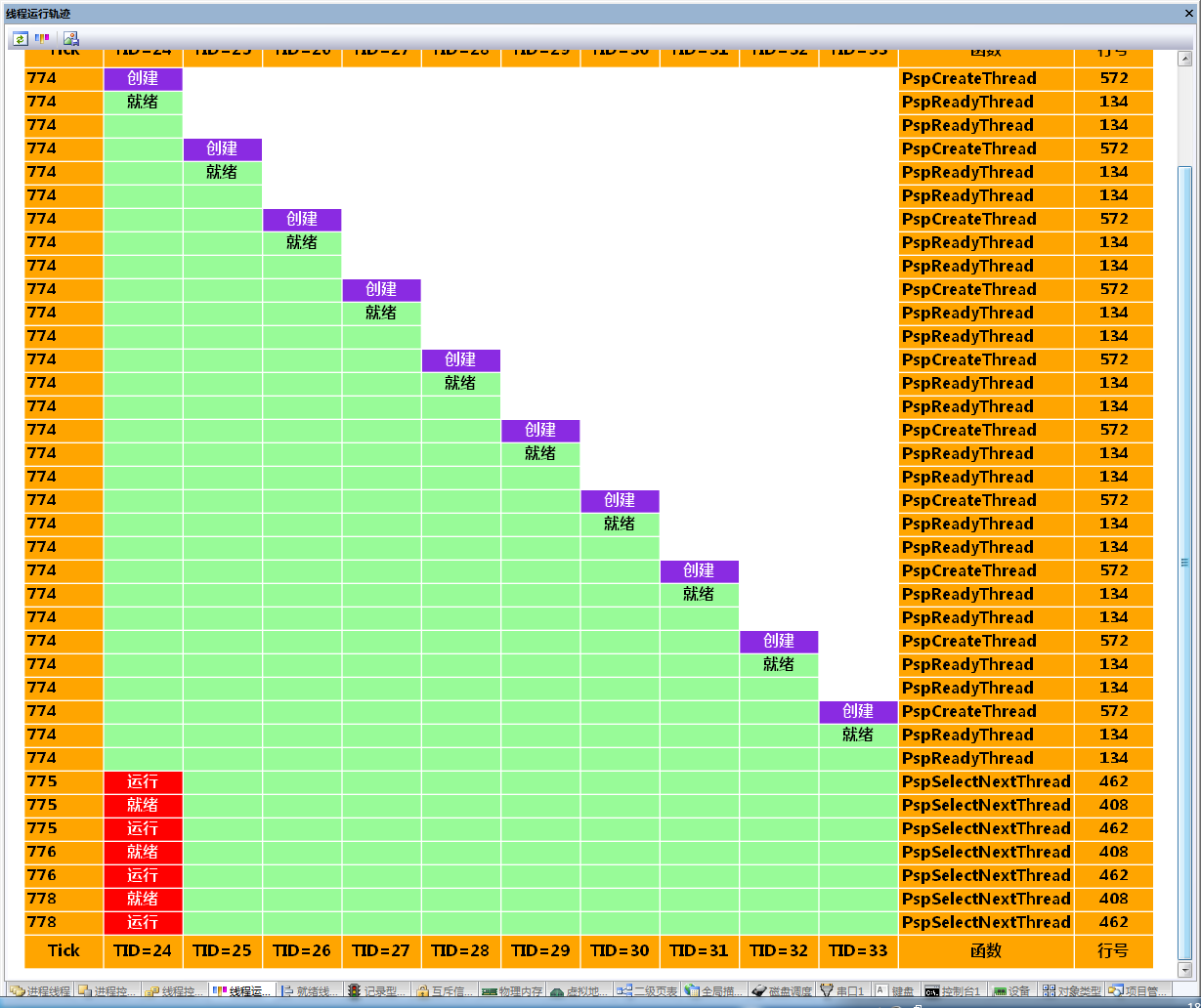
再次在“线程运行轨迹”窗口中点击工具栏上的“绘制指定范围线程”按钮，输入起始线程 ID为 24 和结束线程 ID 为 33,点击“绘制”按钮，可以看到仍然是只有 ID 为 24 的线程处于运行状态，其它 9 个线程处于就绪状态。

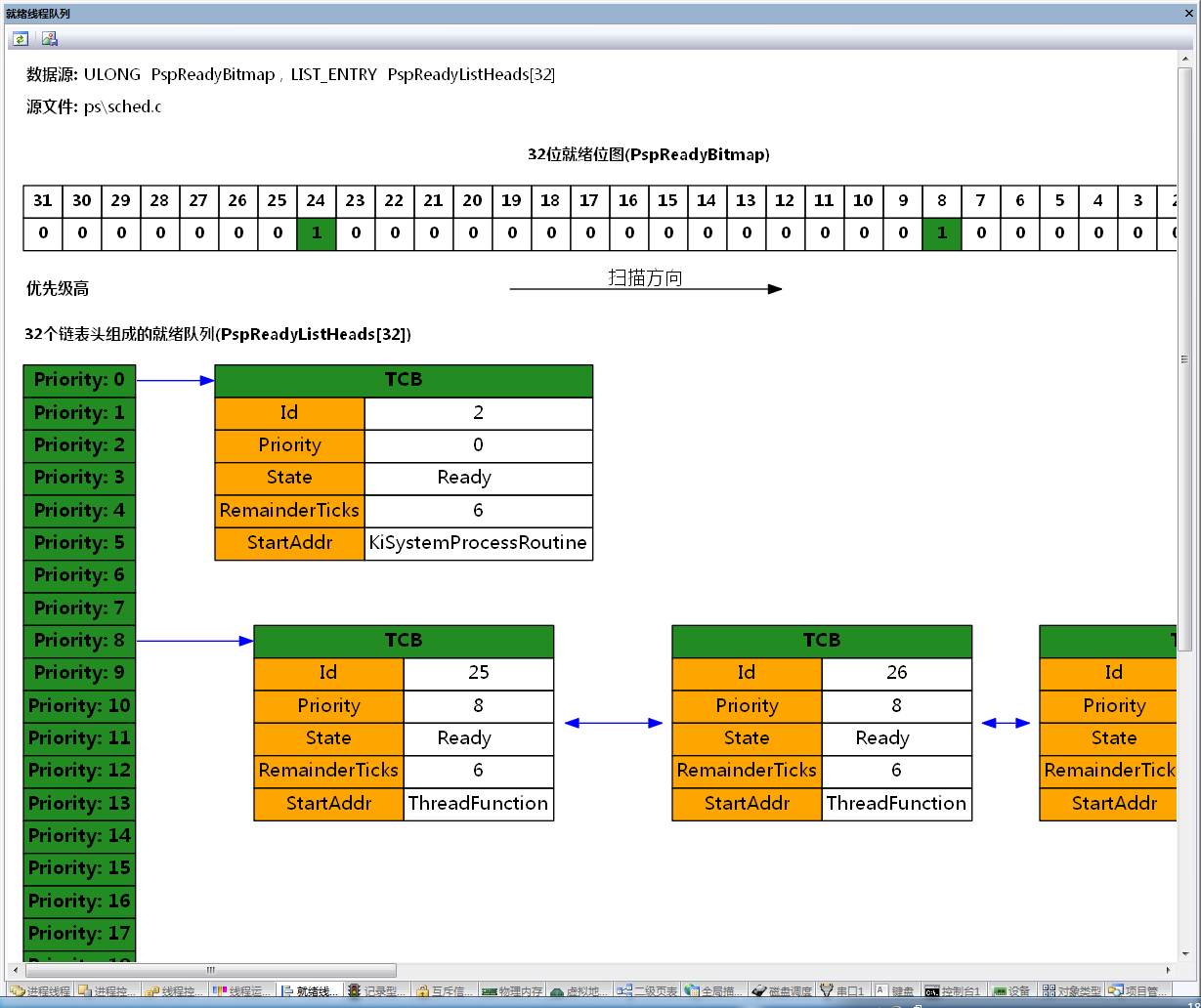




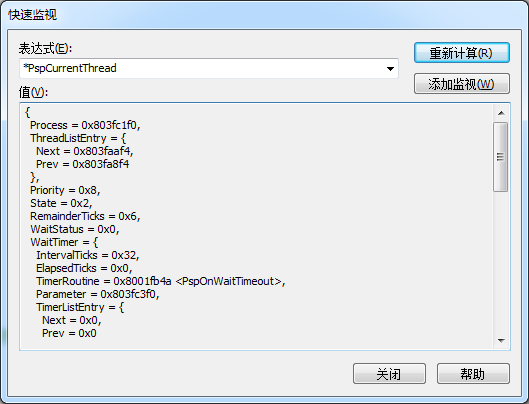
继续之前的调试。在 ps/sched.c 文件的 PspSelectNextThread 函数中，调用 BitScanReverse 函数扫描就绪位图的代码行（第 391 行）添加一个断点。按 F5 继续执行。因为每当有定时计数器中断发生时（每 10ms 一次）都会触发线程调度函数PspSelectNextThread，所以很快就会在刚刚添加的断点处中断。此时，刷新“就绪线程队列”窗口，仍然会显示如图所示的内容。

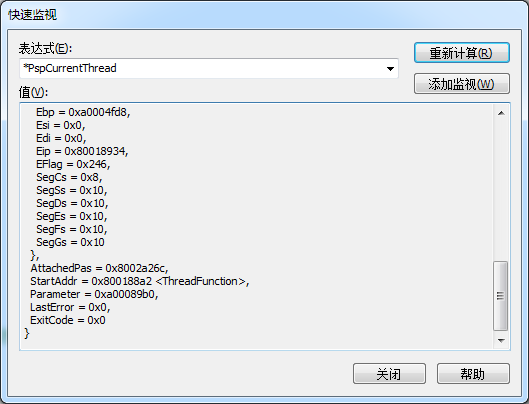






在“调试”菜单中选择“快速监视”，在“快速监视”对话框的“表达式”中输入表达式“\*PspCurrentThread”后，点击“重新计算”按钮，可以查看当前正在运行的线程（即被中断的线程）的线程控制块中各个域的值。其中优先级（Priority 域）的值为 8；状态（State 域）的值为 2（运行状态）；时间片（RemainderTicks 域）的值为 6；线程函数（StartAddr 域）为ThreadFunction。综合这些信息即可确定当前正在运行的线程就是“rr”命令新建的第 0 个线程。



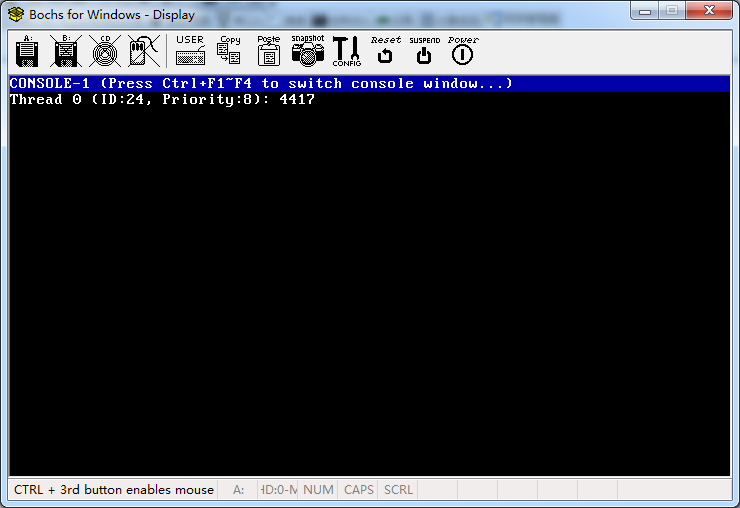


（2）调试当前线程被抢先的情况

选择“调试”菜单中的“删除所有断点”，删除之前添加的所有断点。

在 ps/sched.c 文件的 PspSelectNextThread 函数的第 402 行添加一个断点。

按 F5 继续执行，激活虚拟机窗口，可以看到第 0 个新建的线程正在执行。



在虚拟机窗口中按下一次空格键，使之前处于阻塞状态的控制台派遣线程进入就绪状态，并触发线程调度函数 PspSelectNextThread，就会在刚刚添加的断点处中断。

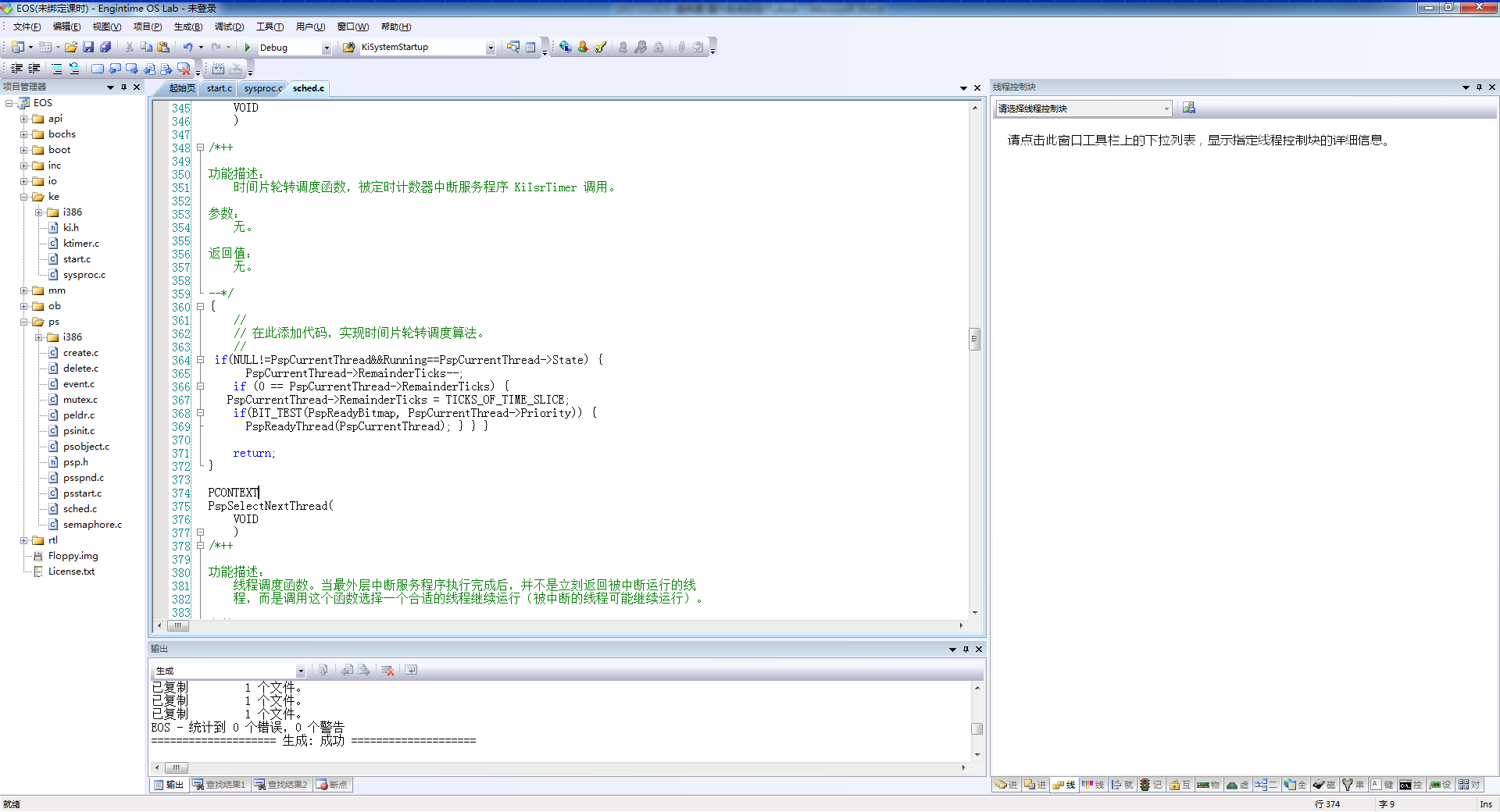
刷新“就绪线程队列”窗口，会显示如图 14-5 所示的内容。可以看到，在 32 位就绪位图中第 24位用绿色高亮显示且值为 1，说明优先级为 24 的就绪队列中存在就绪线程。在“32 个链表头组成的就绪队列”中可以查看优先级为 24 的就绪队列中挂接了一个处于就绪状态的线程，如图所示，通过其线程函数名称可以确认其为控制台派遣线程。





3、为 EOS 添加时间片轮转调度算法

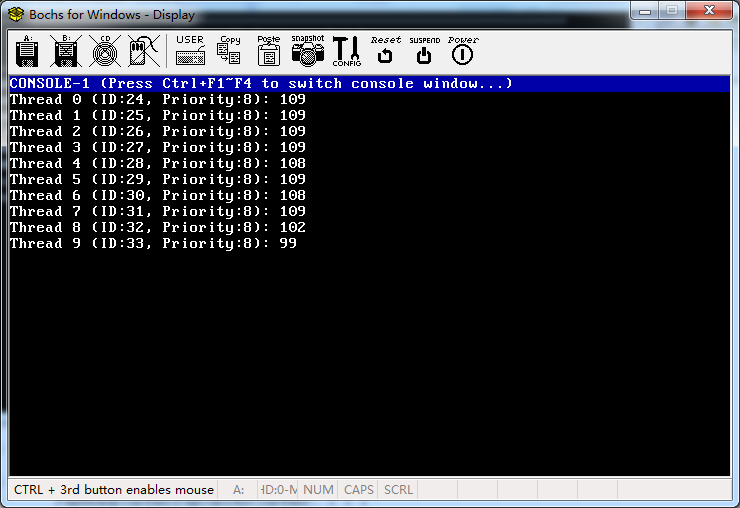
修改 ps/sched.c 文件中的 PspRoundRobin 函数(第 344 行)，在其中实现时间片轮转调度算法。



代码修改完毕后，按 F7 生成 EOS 内核项目。

按 F5 启动调试。

在 EOS 控制台中输入命令“rr”后按回车。应能看到 10 个线程轮转执行的效果，如所示。



4、修改线程时间片的大小

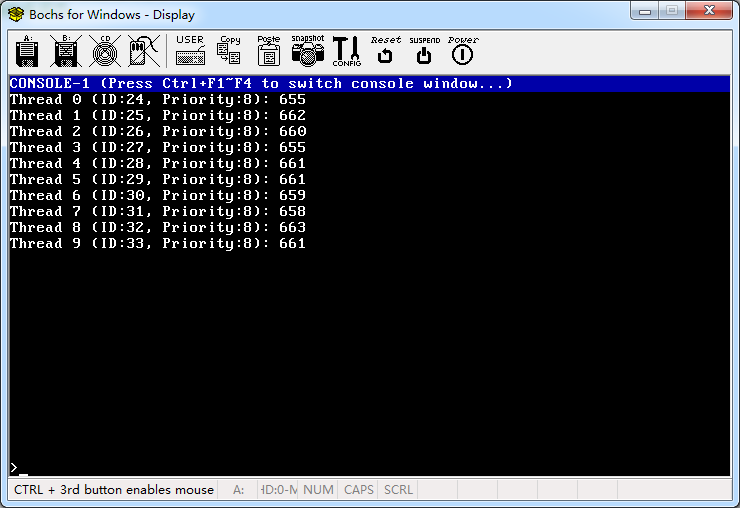
在 OS Lab 的“项目管理器”窗口中找到 ps/psp.h 文件，双击打开此文件。将 ps/psp.h 第 120 行定义的 TICKS\_OF\_TIME\_SLICE 的值修改为 1。

按 F7 生成 EOS 内核项目。

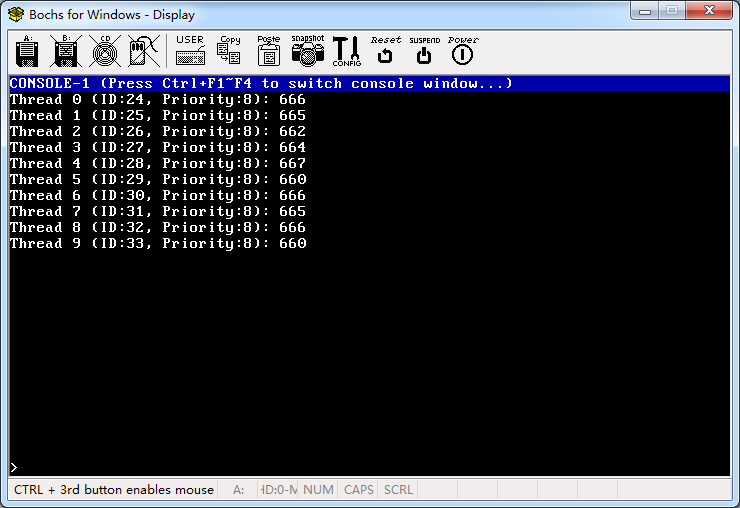
按 F5 启动调试。

在 EOS 控制台中输入命令“rr”后按回车。观察执行的效果。

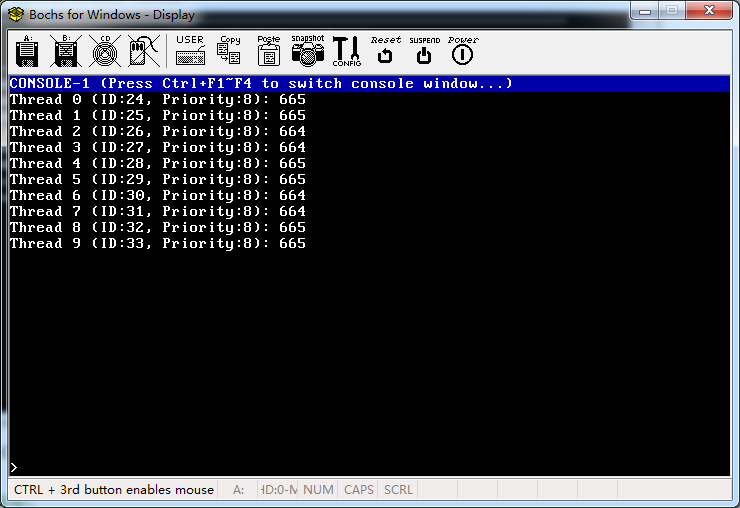
（1）TICKS\_OF\_TIME\_SLICE 1



（2）TICKS\_OF\_TIME\_SLICE 20



（3）TICKS\_OF\_TIME\_SLICE 100



* 1. **实验的思考与问题分析**

1. 结合线程调度执行的时机，说明在 ThreadFunction 函数中，为什么可以使用“关中断”和“开中断”的方法来保护控制台这种临界资源。一般情况下，应该使用互斥信号量（MUTEX）来保护临界资源，但是在 ThreadFunction 函数中却不能使用互斥信号量，而只能使用“关中断”和“开中断”的方法，结合线程调度的对象说明这样做的原因。

答:

EOS会设置CPU停止响应外部设备产生的硬中断，也就不会在由硬中断触发线程调度。开中断和关中断使处理机在这段时间屏蔽掉了外界所有中断，使他线程无法占用资源。使用开中断和关中断进程同步不会改变线程状态，可以保证那些没有获得处理器的资源都在就绪队列中。关中断后CPU就不会响应任何由外部设备发出的硬中断（包括定时计数器中断和键盘中断等）了，也就不会发生线程调度了，从而保证各个线程可以互斥的访问控制台。这里绝对不能使用互斥信号量（mutex）保护临界资源的原因：如果使用互斥信号量，则那些由于访问临界区而被阻塞的线程，就会被放入互斥信号量的等待队列，就不会在相应优先级的就绪列中了，而时间轮转调度算法是对就绪队列的线程进行轮转调度，而不是对这些被阻塞的线程进行调度，也就无法进行实验了。使用“关中断”和“开中断”进行同步就不会改变线程的状态，可以保证那些没有获得处理器的线程都在处于就绪队列中。

1. 时间片轮转调度发现被中断线程的时间片用完后，而且在就绪队列中没有与被中断线程优先级相同的就绪线程时，为什么不需要将被中断线程转入“就绪”状态？如果此时将被中断线程转入了“就绪”状态又会怎么样？可以结合 PspRoundRobin 函数和 PspSelectNextThread 函数的流程进行思考，并使用抢先和不抢先两种情况进行说明。

答:

（1）因为其他优先队列的线程等待时间不能过长。

（2）若将中断线程转入就绪队列，只有当此线程执行完毕之后，其他队列的线程才有机会进入就绪队列，尤其是当其他就绪队列中的线程关于人机交互的时候，会严重影响用户体验。

1. 在 EOS 只实现了基于优先级的抢先式调度时，同优先级的线程只能有一个被执行。当实现了时间片轮转调度算法后，同优先级的线程就能够轮流执行从而获得均等的执行机会。但是，如果有高优先级的线程一直占用 CPU，低优先级的线程就永远不会被执行。尝试修改 ke/sysproc.c 文件中的ConsoleCmdRoundRobin 函数来演示这种情况（例如在 10 个优先级为 8 的线程执行时，创建一个优先级为 9 的线程）。设计一种调度算法来解决此问题，让低优先级的线程也能获得被执行的机会。

答：

解决该问题的最简单方法是实现动态优先级算法。动态优先级是指在创建进程是所赋予的优先级，可随线程的推进而改变，以便获得良好的性能调度。例如，可用规定，在就绪队列中的线程，随着其等待时间的增长，其优先级以速率X增加，并且正在执行的线程，其优先级以速率y下降。这样，在各个线程具有不同优先级的情况下，对于优先级低的线程，在等待足够的时间后，其优先级便可能升为最高，从而获得被执行的机会。此时，在基于优先级的抢占式调度算法、时间片轮转调度算法和动态优先级算法的共同作用下，可防止一个高优先级的长作业长期的垄断处理器。

4. EOS 内核时间片大小取 60ms（和 Windows 操作系统完全相同），在线程比较多时，就可以观察出线程轮流执行的情况（因为此时一次轮转需要 60ms，10 个线程轮流执行一次需要 60×10＝600ms，所以EOS 的控制台上可以清楚地观察到线程轮流执行的情况）。但是在 Windows、Linux 等操作系统启动后，正常情况下都有上百个线程在并发执行，为什么觉察不到它们被轮流执行，并且每个程序都运行的很顺利呢？

答：

在Windows、linux等操作系统中，虽然都提供了时间片轮转调度算法却很少真正被派上用场，下面解释原因，在Windows任务管理器中，即使系统中已经运行了数百个线程，但CPU的利用率仍然很低，甚至为0.因为这些线程在大部分时间都处于阻塞状态，阻塞的原因是各种各样的，最主要的原因是等待I/O完成或者等待命令消息的到达。例如，在编辑Word文档时，每敲击一次键盘，Word就会立即作出反应，并且文档中插入字符。此时会感觉Word运行的非常流畅。事实上，并非如此，Word主线程大部分时间都处于阻塞等待状态，等待用户敲击键盘。在用户没有敲击键盘或没有使用鼠标点击时，Word主线程处于阻塞状态，它将让出处理器给其它需要的线程。当用户敲击一个按键后，Word主线程将会立刻被操作系统唤醒，此时Word开始处理请求。Word在处理输入请求时所用的CPU时间是非常短的（因为CPU非常快），是微秒级的，远远低于时间片轮转调度的时间片大小（Windows下是60毫秒），处理完毕后Word又立刻进入阻塞状态，等待用户下一次敲击键盘。或者拿音乐播放器来分析，表面上感觉播放器在不停地播放音乐，但是CPU的利用率仍然会很低。这是由于播放器将一段声音编码交给声卡，由声卡来播放，在声卡播放完这段声音之前，播放器都是处于阻塞等待状态的。当声卡播放完片段后，播放器将被唤醒，然后它将下一个声音片段交给声卡继续播放。掌握了上面的知识后，就可以很容易解释为什么这么多线程同时在运行而一点都感觉不到轮替现象。