一种基于虚拟机的操作系统内核调试方法

杨 杰 成新民
(湖州师范学院信息与工程学院，浙江湖州 313000)

[摘 要] 由于操作系统自身的特点，调试操作系统需要构建特殊的环境。通过构建基于虚拟机的调试环境，给出在该环境下操作系统内核的调试方法。与传统调试环境相比，虚拟机调试环境更接近于操作系统真实的运行环境，提供了更强、更直接的调试功能。

[关键词] 操作系统；虚拟机；远程调试；linux0.11 内核

1 引言

阅读、修改和增强其源代码是学习和研究操作系统的最好方法，也就是所谓的操作系统实验方法，其中调试环境是操作系统实验环境最重要的组成部分。与应用程序相比，进行操作系统实验要困难得多。主要原因是调试器需要调试的目标正是它所在的运行环境，对调试环境的调试很有可能会影响到本身的运行情况。因此构建操作系统调试环境的关键是实现调试程序所在的环境和调试目标相分离，为此通常调试操作系统采用硬件调试器或远程调试方式。

2 传统的操作系统调试环境

除了使用硬件调试器，传统的操作系统实验环境通常采用远程调试方式来满足调试环境和调试目标相分离的需求。其具体方案如下，首先把一个称为 stub 的程序插入到待调试操作系统中，经重新编译生成一个具有远程调试功能的调试目标（待调试的操作系统），其中 stub 程序负责控制调试目标、搜集调试目标的状态信息以及与远程机器进行通讯。然后在另外一台机器上运行 gdb 程序，因为 gdb 中内嵌有串口通讯协议，并且规范了远程机器调试命令的一些数据传输包的格式，所以只要调试目标中的 stub 程序遵守相同的协议和数据包格式，就可以实现远程调试的功能。图 1 是远程调试方式的示意图。

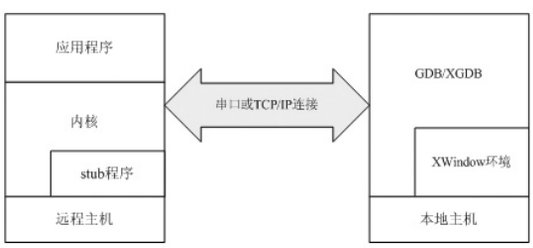

图 1 远程调试方式

但是并不是所有的操作系统和版本都提供现成的 stub 程序，大多数情况下需要自己编程实现，这项工作通常需要熟悉该操作系统的系统程序员来完成，对大多数想通过实验来学习、研究该操作系统的人来说，是不能所能胜任的。考虑到调试器都是通过父进程（或硬件）监控子进程的动作和状态来实现的，要调试操作系统就意味着需要有一个内核的父进程，所以为实现调试环境和调试目标相分离的目的，可将待调试的操作系统在虚拟机上运行，此时虚拟机即为待调试操作系统内核的父进程。

3 基于 bochs 虚拟机的内核调试方法

bochs 虚拟机提供了两种调试方式，当使用内部调试器时，其功能相当于一个硬件调试器，具有极为强大的软件调试能力，可以不受任何限制地调试操作系统的所有部分，包括内存管理、中断处理、设备管理、进程管理、文件系统等等，甚至还能调试系统的启动过程。文中介绍的调试方法是使用内部调试器进行的。

Linux0.11 内核作为一个早期的 Linux 版本，虽然简陋且没有了使用价值，但由于它简单以及包含了大量现代 Linux 的实现原理，所以成为很好的学习和研究操作系统的样本。文中也选择 Linux0.11 内核作为对象来展示操作系统的调试方法。

3.1 C 语言源代码级调试功能的实现

首先在编译操作系统时，除生成正常使用的内核外，再使用 - g 参数，生成一个带调试信息的内核副本，该内核副本无须执行，我们只使用它包含的调试信息。然后在 gdb 中读入内核副本中的符号表，通过这些调试信息就可以方便地获得源代码中每一条语句在运行时的地址和对应的汇编指令。例如要调试 main 函数，可以在 gdb 中用命令 list main，得到 main 函数源代码和对应行号。

```
104 void main(void)       /* This really IS void, no error here. */
105 {      /* The startup routine assumes (well,...) this */
……
110         ROOT_DEV= ORIG_ROOT_DEV;
111 drive_info= DRIVE_INFO;
112         memory_end= (1<<20)+ (EXT_MEM_K<<10);
……
127         trap_init();
128         blk_dev_init();
……
```

如果需要知道 trap_init()语句在运行时的地址和对应的汇编指令，只要使用命令 info line 127，获得该语句运行时的地址：

Line 127 of “init/main.c” starts at address 0x669f and ends at 0x66a4。

然后使用反汇编命令 disassemble 0x669f 0x66a4，获得该语句对应的汇编指令：

0x0000669f <main+131>: call 0x7448 <trap_init>。

同样我们可以获得 110 行语句 ROOT_DEV= ORIG_ROOT_DEV 对应的汇编指令和运行时的地址：

0x00006621 <main+5>: movzwl 0x901fc, %eax

0x00006628 <main+12>: mov %eax, 0x1b990。

通过上述方法，解决了在使用内部调试器时只具有汇编级调试能力的问题，在发挥内部调试器强大的调试功能的同时实现了 c 语言源码级的调试功能。

3.2 调试系统启动过程

通常系统启动过程时，还没有任何应用程序或者监控程序，当然也就没有任何基于软件的调试器可用。如果不用硬件调试器的话是无法对启动代码进行跟踪调试的。然而当操作系统在虚拟机中运行时，启动过程作为虚拟机调试器的子进程，它的动作和状态完全处在虚拟机调试器的监控之下，所以在虚拟机中可以实现分析、调试操作系统的启动过程。

x86 体系结构计算机的启动过程是：系统加电后，由 BIOS 中的启动例程把启动盘上引导扇区的数据加载到绝对地址 0x7c00- 0x7dff 处，然后从 0x7c00 处开始执行引导扇区中的装载程序，继续完成操作系统的装入过程。因此为了跟踪、调试操作系统的启动代码，使用命令 pb 0x7c00，在物理地址 0x7c00 处设置断点。然后键入命令 c，让系统继续执行。当系统在断点 0x7c00 处暂停时，就是我们需要调试的操作系统启动代码的入口。接着可以使用各种调试命令来完成启动代码的调试。

3.3 查看内核工作环境

与 DOS 和 MINIX 等工作在实模式下的操作系统不同，linux 工作

在386保护模式下，使用存储器分页管理机制来实现虚拟存储器。因此在内核进入真正的工作前，还需要进行大量的初始化工作，以此为操作系统的执行建立必要的工作环境，特别是全局描述符表、中断描述符表以及页目录表和页表的设置，决定了整个操作系统的工作过程和方式。了解工作环境的设置，对于理解整个操作系统的工作是至关重要的。

通过源代码可知，当内核把自己转换到用户态，并进入进程0执行时，表示它已经完成了初始化，将进入工作状态。因此可在此暂停程序的执行，以便查看操作系统的初始工作环境。当然查看操作系统工作环境的设置，可以在其初始化完成后的任何时候进行。

查源代码知，断点应该设置在main.c的138行，使用4.1中介绍的方法可以知道该断点的地址是0x66f9，当系统在断点处暂停时，执行查看cpu寄存器命令，得到部分数据如下所示：

gdtr:base=0x00005cb8,limit=0x7ff

idtr:base=0x000054b8,limit=0x7ff

cr3:0x00000000

第一行显示的是全局描述符表寄存器的值，它说明全局描述符表的起始地址在0x5cb8处,长度为2K。第二行中断描述符表寄存器的值，给出了中断描述符表的起始地址在0x54b8处,长度为2K。有了上述的信息后，可接着使用查看物理内存命令pb来获得操作系统运行时全局描述符表和中断描述符表的设置情况。X86体系结构计算机的分页管理机制采用二级页映射方式，第三行控制寄存器CR3的值给出了页目录表在内存中的起始地址在0x0处。执行查看物理内存命令xp/8wx 0x0，得到页目录表的部分内容（表的前八项数据）：

0x00001027　0x00002007　0x00003007　0x00004027

0x00000000　0x00000000　0x00000000　0x00000000

这些数据中的每一项都给出了对应页表的起始地址及属性，如第一项0x00001027,就给出了第一个页表的信息，第二项是第二个页表的，依次类推。但从第四项开始值为全0，表示在初始化时只使用了4个页表。有了每个页表的信息后，再次使用物理内存查看命令来查看具体的页表设置情况，以此了解系统中32位线性地址和物理地址的对应关系。

3.4　查看进程

进程管理是操作系统中的一个重要组成部分，同时进程是程序的一次执行过程，分析和研究一个操作系统需要能全面了解其中各个进程的情况。Bochs虚拟机为查看系统各进程提供了最直接的方法。

下面就选择init进程作为对象来说明具体的查看方法，在linux0.11内核中init进程直接由进程0创建，它负责建立终端环境和创建用于运行shell程序的子进程。linux0.11内核是通过全局任务指针数组task感知进程的，因此查找进程可以通过它进行，查地址映象文件得到全局任务指针数组task的首地址为0x1a180，使用物理内存查看命令查看地址0x1a180-0x1a1a0的数据，我们例子中的数据如下所示：

0x00019160　0x00fff000　0x00ffd000　0x00fbf000

0x00000000　0x00000000　0x00000000　0x00000000

从数据中可以看出当前共有4个进程（指针为0表示进程还没有创建），因为init进程直接由进程0创建，且创建时除了进程0还没有其他进程，所以第二项数据就是init进程的TCB指针。再次使用查看内存命令就能获得其TCB的内容，参照TCB结构各成员的偏移量，就可以了解init进程的所有信息，如进程现在的运行状态、任务时间片、优先级信息等等。有了进程TCB结构的信息，仅仅简单地使用内存查看命令，就可以进一步获得其用户态代码段、数据段、堆栈段的设置情况，以及线性地址到物理地址的转换关系。

另外，查看linux0.11中的进程还可以从其全局描述符表入手，使用4.3中介绍的查看操作系统工作环境的方法，获得其全局描述符表设置如下图所示。

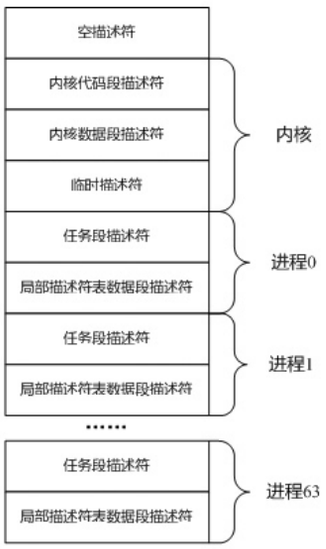

图　2linux0.11内核的全局描述符表

从这些数据中可以获得各进程的局部描述符表和任务状态段的设置情况，从而了解进程的用户态代码段、数据段、堆栈段和核心态堆栈段的设置情况。

4　结语

与传统的调试环境相比，虚拟机调试环境更接近于操作系统真实的运行环境，并且不占用任何系统资源。提供了类似于硬件调试器的功能，可以不受限制地对操作系统的各部分进行分析、调试，包括启动过程、底层的系统调用过程、进程的切换过程等等。并且使用简单，只要掌握设置断点、反汇编、查看内存等3个主要命令就可以完成。虽然当前绝大多数虚拟机只能仿真x86体系结构计算机，但是随着技术的发展虚拟机将成为分析、调试操作系统的主要手段。

参考文献

[1] 李善平,刘文峰. Linux内核2.4版源代码分析大全[M].北京:机械工业出版社,2002.

[2] Kevin Lawton,Bryce Denney. Bochs user Manual [EB/OL]. http://bochs.sourceforge.net,2005.

[3] Gray Nutt.操作系统现代观点[M].北京: 机械工业出版社,2004.

[4] 赵炯. Linux内核完全注释[EB/OL]. www.oldlinux.org,2005.