操作系统课程设计实验报告

宋振华

2018 年 11 月 19 日

1 简述

2 阅读源码

2.1 各部分功能

3 选择可视化模块

4 提取数据

4.1 提取什么数据

4.2 提取数据方式

4.3 提取数据细节

对于使用 gdb 脚本调试, 应当注意以下几个方面:

4.3.1 添加断点

断点不宜太多 在 gdb 脚本中, 不能一次性添加太多断点, 否则会严重影响执行速度. 原因如下:

1. 频繁执行 gdb 脚本, 耗费大量时间在 gdb 输出, 保存断点等方面;

   举例来说, console.c 中的所有函数, 在系统启动时, 总计被调用了 5000 多次.

如果在诸多函数添加断点, 系统启动过程, 断点可能被执行到几十万次.

2. 在执行 gdb 脚本时, bochs 虚拟机时钟滴答应该并没有停止, 这会引起大量的时钟中断, 从而导致 linux0.11 大量时间片轮转操作.

解决方案 如果需要大量记录数据, 可以将大量断点分成若干批次, 每次执行只加入其中一部分断点.

优点: 可以极大减少 gdb 调试引起的 linux0.11 时间片轮转操作, 使 gdb 调试过程更接近于正常启动过程.

缺点: 不能完全保证两次执行操作完全相同, (如时钟中断时机不一定完全相同).

相比而言, 由于 linux 具有很好的模块性, 各模块之间耦合性较低, 分别调试问题不大.

加断点需谨慎 在 linux0.11 中, 有些函数执行极其频繁, 如 sched.c 中 void schedule(void) 函数 (该函数用于时间片轮转). 如需要 gdb 连续地执行代码 (区别与单步调试), 这些函数会频繁被执行, 从而严重影响速度. 原因同上.

汇编语言添加断点 汇编语言中定义的函数, 类似于这样:

```
keyboard_interrupt:
        pushl %eax
        pushl %ebx
        pushl %ecx
        pushl %edx
        push %ds
        push %es
        movl $0x10,%eax
        mov %ax,%ds
        mov %ax,%es
        xor %al,%al             /* %eax is scan code */
        inb $0x60,%al
        cmpb $0xe0,%al
```

其中 keyboard_interrupt 为函数入口, 随后几条 pushl 等语句, 为函数参数初始化过程. gdb 单步调试, 会将函数初始化过程一步完成, 从而不会在参数初始化过程停留.

因此, 在 pushl %eax 这一条语句上添加断点, 是不会被执行到的.

解决方案 使用 gdb 图形化工具, 在汇编语言函数入口处, 多设几个断点, 观察会在哪里暂停. 会暂停的位置, 可以在 gdb 脚本中设为断点.

4.4 关于 Makefile

及时清理生成代码 代码生成过程会有许多中间文件, 不要将中间文件错误地当做源代码文件. 如 kernel/chr_drv/kb.S(源代码) 和 keyboard.s(中间代码).

亲测在 keyboard.s 中添加断点, 会输出很多奇怪的东西.

解决方案 及时执行 make distclean 操作, 清除中间文件.

4.5 提取数据脚本

4.5.1 gdb 脚本技巧

1. gdb 脚本可以添加函数, 以简化代码;
2. gdb 脚本 source 语句可以导入别的 gdb 脚本, 类似于 C 语言的 #include 语句.
3. gdb 脚本中 set 定义的变量, 是全局变量;
4. 在 bochs 环境中, gdb 脚本某一处出现 bug, 可能导致 bochs 退出. 建议对 gdb 脚本做好备份.

5 可视化方案

5.1 编程语言

可视化展示界面使用 HTML5+JavaScript+CSS. 优点: 设计界面较为快捷.

5.2 界面设计

6 最终效果

7 感想

7.1 收获

7.2 对课程的建议