山东大学 软件 学院

操作系统 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：201400301291 | 姓名： 雷超 | | 班级： 四班 |
| 实验题目：死锁问题实验 | | | |
| 实验学时：2 | | 实验日期： 2016-05-10 | |
| 实验目的：  通过本次实验观察死锁的产生现象，考虑死锁问题的方法。进一步加深对于死锁问题的理解。掌握解决死锁问题的集中算法编程和调试技术，练习怎样构造管程和条件变量，利用管程机制类避免死锁和饥饿问题的发生。 | | | |
| 硬件环境：  00:00.0 Host bridge: Intel Corporation Xeon E3-1200 v3/4th Gen Core Processor DRAM Controller (rev 06)  00:01.0 PCI bridge: Intel Corporation Xeon E3-1200 v3/4th Gen Core Processor PCI Express x16 Controller (rev 06)  00:02.0 VGA compatible controller: Intel Corporation 4th Gen Core Processor Integrated Graphics Controller (rev 06)  00:03.0 Audio device: Intel Corporation Xeon E3-1200 v3/4th Gen Core Processor HD Audio Controller (rev 06)  00:14.0 USB controller: Intel Corporation 8 Series/C220 Series Chipset Family USB xHCI (rev 05)  00:16.0 Communication controller: Intel Corporation 8 Series/C220 Series Chipset Family MEI Controller #1 (rev 04)  00:1a.0 USB controller: Intel Corporation 8 Series/C220 Series Chipset Family USB EHCI #2 (rev 05)  00:1b.0 Audio device: Intel Corporation 8 Series/C220 Series Chipset High Definition Audio Controller (rev 05)  00:1c.0 PCI bridge: Intel Corporation 8 Series/C220 Series Chipset Family PCI Express Root Port #1 (rev d5)  00:1c.1 PCI bridge: Intel Corporation 8 Series/C220 Series Chipset Family PCI Express Root Port #2 (rev d5)  00:1c.2 PCI bridge: Intel Corporation 8 Series/C220 Series Chipset Family PCI Express Root Port #3 (rev d5)  00:1c.4 PCI bridge: Intel Corporation 8 Series/C220 Series Chipset Family PCI Express Root Port #5 (rev d5)  00:1d.0 USB controller: Intel Corporation 8 Series/C220 Series Chipset Family USB EHCI #1 (rev 05)  00:1f.0 ISA bridge: Intel Corporation HM86 Express LPC Controller (rev 05)  00:1f.2 SATA controller: Intel Corporation 8 Series/C220 Series Chipset Family 6-port SATA Controller 1 [AHCI mode] (rev 05)  00:1f.3 SMBus: Intel Corporation 8 Series/C220 Series Chipset Family SMBus Controller (rev 05)  01:00.0 3D controller: NVIDIA Corporation GM107M [GeForce GTX 860M] (rev a2)  08:00.0 Network controller: Intel Corporation Wireless 3160 (rev 93)  09:00.0 Ethernet controller: Realtek Semiconductor Co., Ltd. RTL8111/8168/8411 PCI Express Gigabit Ethernet Controller (rev 10)  0a:00.0 Unassigned class [ff00]: Realtek Semiconductor Co., Ltd. RTS5249 PCI Express Card Reader (rev 01) | | | |
| 软件环境：  Linux EVA 4.6.2-gentoo-EVOLUTION x86\_64 Intel(R) Core(TM) i7-4710HQ CPU @ 2.50GHz GenuineIntel GNU/Linux  dev-libs/gmp-6.0.0a  dev-libs/mpfr-3.1.3\_p4  dev-libs/mpc-1.0.2-r1  sys-devel/gcc-4.9.3 , 5.3.0 , 5.4.0  sys-libs/glibc-2.22-r4  sys-devel/clang-3.5.0-r100  app-editors/emacs-24.5-r1  sys-kernel/gentoo-sources-4.6.2  sys-kernel/linux-headers-4.6  sys-apps/systemd-226-r2  ABI\_X86=”32 64” | | | |
| 实验步骤与内容：  单桥问题：两个城市南北方向之间存在一条铁路，多列火车可以分别从两个城市的车站排队等待进入车道向对方城市行驶，铁路在同一时间，只能允许同一方向上行车，否则会装车。模拟现实两个方向上行车，并且阻止饥饿，使用管程构建。 | | | |
| 结论分析与体会：  此次实验的重点应该在对于防止饥饿上，当然，实验的难度也由解决饥饿所采用的算法的复杂性所决定。我在此次实验中采用的是两套机制来阻止饥饿问题的发生。同时也相应的解决产生的死锁的问题。  首先，当对面车道有车辆在等待的时候，某一方的的车辆同事达到该车道上所能行驶的车辆的最大值，并且另一方的车道上没有等待的车辆，这个时候设定的标志FLAG改变，当己方的车辆就绪时，读到的FLAG和自己的方向不符，阻塞到队列里面；直到另一方的车辆上车道并且达成相同的条件的时候改变FLAG同时唤醒己方的车辆。  第二套解决方案是设定阈值，即某一个如果己方在行驶后，发现对方有车辆等待，对计数器进行自加操作，知道到达设定的阈值，这里默认的阈值是峰值的两倍，保证长时间即使没有达到峰值也能自动阻止一方持续占用车道。  综上这个就是我阻止饥饿所使用的两套技术解决方案。相对与普通的实现需要更长时间的编程和磨练，当然处理出所能达到的效果也会更好，尤其在允许的车辆越多的情况下越能体现出来效果。  计算机系统中，如果系统的资源分配策略不当，更常见的可能是书写的程序有错误等，则会导致进程因[竞争资源](http://baike.baidu.com/view/1260087.htm" \t "/home/elvis/Documents\\x/_blank)不当而产生死锁的现象。我在进行实验的时候所遇到的死锁不是主要解决的目标，但是使用管程进行封装处理后，顺利的解决了互斥条件不满足的问题和请求和保持的问题。解决了死锁。 | | | |

附件：