基于实时 Linux 计算机联锁系统实时性分析与改进*

由建宏[1]， 董 昱[1]， 李智军[2]

(1. 兰州交通大学 电子与信息工程学院,甘肃 兰州 730070;2. 青岛理工大学 计算机工程学院,山东 青岛 266033)

摘 要:计算机联锁系统是保障列车在车站范围内安全、高效运行的实时控制系统,它必须具有非常高的安全性、可靠性. 这首先应该解决实时性的问题,针对基于 Linux 下的计算机联锁系统的实时性进行分析,并结合计算机联锁系统的实时性模块给出分析,采用实时 Linux 系统进行设计与实现,并对改进后的系统与原系统进行测试比较.

关键词:RTLinux;实时性;内核;进程调度;计算机联锁

中图分类号:TP316.2 **文献标识码**:A

近几年,随着铁路运输朝着高速、重载、高密度方向的发展,对相应的铁路行车控制系统的可靠性和安全性的要求也越来越高[1]. 而如何提高计算机联锁控制系统的实时性成为提高整个系统可靠性和安全性的核心问题. 但标准 Linux 是为通用操作系统而设计,它应用于计算机联锁的实时环境有许多缺陷. 为了使 Linux 能适应实时应用的要求,出现了以 Linux 为基础,在其上进行实时化改造的方案,并已成为主流发展方向.

1 计算机联锁的实时控制软件结构

在计算机联锁系统软件结构中,进路处理的 5 个阶段的处理模块:选排一致检查及道岔控制命令生成模块、进路锁闭模块、信号开放模块、信号保持开放模块、进路自动解锁模块都是实时模块. 根据中华人民共和国铁道行业标准,计算机联锁系统开关量信息采集周期应适应列车最高运行速度的要求,以前是 250 ms. 对于时速 160 km/h 的运行,响应时间为 1.08 s,对于时速 250 km/h 的运行,响应时间为 0.648 s. 对于计算机联锁系统来讲,其主要功能是完成对联锁进路的控制,为了使联锁机的工作具有实时性,必须不失时机的采集输入数据的变化情况.

2 影响 Linux 系统实时性的因素

实时性就是指工业控制计算机系统应该具有的能够在限定的时间内对外来事件作出反应的特征. 计算机联锁系统必须保证实时任务在规定时间内完成. 标准 Linux 内核是为通用操作系统而设计,虽然它采取了许多技术来加快系统的运行和反应速度,但它本质上并不是一个实时系统,并不能完全满足计算机联锁系统的实时性要求.

2.1 进程调度

Linux 有 2 种类型的进程: 普通进程和实时进程. 在 Linux 系统中实时进程的优先级比一般进程的优先级高. 由此可见,Linux 中已经有了一些实时操作系统的特征. 但这还不能满足计算机联锁系统实时应用的要求.

Linux 内核中有 3 种不同的进程调度策略:

1) SEHED_FIFO:先进先出策略,一种实时进程的调度策略. 采用这个策略,实时进程的调度之取决于它的优先级,只要没有更高优先级的进程进入运行队列,该实时进程就可以一直运行下去. 相同优先级的实时进程采取先来先服务的原则.

2) SCHED_RR:时间片轮转策略,也是一种实时进程调度策略. 每一个进程都有一个时间片,当进程的时间片用完时,就把它置于就绪队列尾部,并重新为其分配时间片.

3) SCHED_OTHER:非实时进程的基于优先级的轮转策略.

2.2 计时器精度

计时器精度对实时性也有较大的影响. Linux 通过对硬件时钟的编程产生周期为 100 Hz 的时钟中断,因此任务调度的时间精度最高能达到 10 ms,

这远远无法满足一些对时间精度要求苛刻的实时应用. 当然精度越高,意味着时钟中断越频繁,而花在中断处理上的时间越多. 操作系统必须对时间精度和时钟中断处理的时间开销进行折衷考虑.

2.3 中断处理

大部分内核进程是不能中断的,即一旦一个进程进入到核心模式,它将运行到系统调用的完成或者被阻塞为止. 如果低优先级的进程关闭了中断,那么即使有高优先级的实时进程的中断发生系统也无法响应. 这种情况在实时系统中是不允许发生的.

2.4 非抢占式内核

Linux 的内核是非抢占式的. 当一个任务调用系统计如内核态运行时,它是不可被抢占的,这导致实时任务执行时间的不确定性. 这在实时系统中是不能容忍的.

2.5 虚拟内存

采取了"虚拟内存"的内存管理方式. 即利用交换空间让进程运行在一个比实际内存大的虚拟内存空间里. 当进程访问的虚拟内存的内容在交换空间里时,就要把在交换空间里的页面交换到实际内存中来. 这是一个费时的操作,它使得访问这些数据的时间具有不可预测性.

3 实时 Linux 的设计与实现

由上所述,Linux 存在 3 个不利于实现实时性的不足:过长的中断封锁时间,非抢占式的 Linux 内核,耗尽式的、机会均等的进程调度策略. 为此,针对这些不足,通过提高中断频率,设计实时性内核来提高基于 Linux 的计算机联锁系统的实时性. 下面就介绍一种实现硬实时指标的实时 Linux 方案.

3.1 实时 Linux 的内核原理与系统结构

实时内核都是由中断服务例程(ISR)处理异步事件的,它始终不关闭硬件中断[2],可以接受所有的中断信号[3]. 当中断信号需要实时进程来处理时,实时进程将抢占 Linux 内核(在实时 Linux 中把原来的 Linux 内核作为普通进程对待,并将其优先级设置为最低). 若中断信号需要原来的 Linux 内核来处理,则由实时内核将信号传递给 Linux 内核. 同时,在实时内核中提供标志字,来模拟原来 Linux 内核的关中断情况. 这个标志字在 Linux 打开中断时置"1",关中断时置"0". 实时内核在中断到来时检查标志字,若置"1",则立刻将中断传给 Linux 内核;否则,将所有待处理的中断放入一个队列中,直到 Linux 打开中断时才将它们传给 Linux 内核. 实时 Linux 的系统结构如图 1 所示.

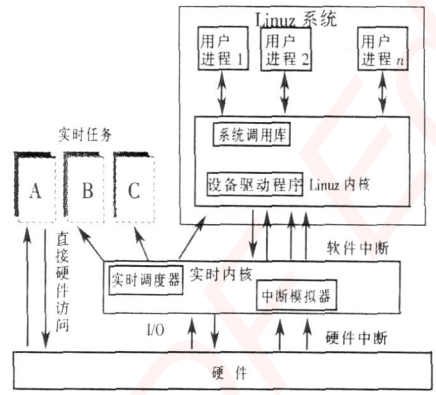

图 1 实时双核结构的实现机理

Fig. 1 Mechanic realization of real-time double kernel

3.2 实时 Linux 中断处理的实现

基于实时 Linux 的设计思想,在 Linux 系统上实现实时 Linux 的关键在于实现实时接收外部中断的中断处理模块(rtlkm.o),以实现相应进程调度的实时进程调度模块. 由于实时中断处理程序可以在任何时候抢占非实时进程,因此普通内核不能屏蔽中断,它所发出的 cli(关中断)、sti(开中断)和 iret(中断返回),都被重新定义为宏 S_CLI、S_STI、S_IRET. 若普通内核发出"S_CLI",并不真正去关闭中断,而只是很简单的改变一个标志变量(如设置 status = 0). 当一个中断发生后,实时内核捕获后决定如何处理,如果该中断有相应的实时处理程序,该程序就被调用. 如果该中断是一个应该有普通内核处理的中断,那么检查 status 的值,如果是 0,则只设置该中断的对应标志,表明该中断被挂起,如果 status = 1,则调用该中断对应的中断服务程序. 当普通内核发出"S_STI"时,只简单地改变 status 的值为 1,然后对所有被挂起的中断,调用其中断服务程序. S_CLI 和 S_STI 的宏定义如下:

```
S_CLI: mov1  $0,status
S_STI: pushf1
  push1  $KERNEL_CS
  push1  $next_sti
  S_IRET
next_sti:
```

S_IRET 完成初始化工作后,检查所有的中断标志位. 如果没有挂起的中断,设置状态变量 mov1 $0,status,之后执行硬件中断返回 iret. 若发现存在被挂起的中断,则转到相应的中断在处理程序执行. 返回后,依次执行其余的待处理的中断,直到没有挂起的中断为止.

3.3 实时 Linux 多线程调度机制的实现

实时进程调度模块(rtlkm_sched.o)主要任务是保证每个实时任务的时限要求.系统采用固定优先级可抢占调度策略:每个实时任务都有一个固定的优先级,每次时钟中断时,调度器选取可运行队列中的最高优先级的任务投入运行.如果已就绪的任务的优先级比当前任务的优先级高,则当前任务被抢断.

实时任务调度可采用单一比率调度算法(Rate-Monotonic Scheduling Algorithm)来判断任务集的可调度性,时限越小的任务优先级越高.对于相对时限(relative deadline)等于周期的周期性实时任务,若 n 个任务满足以下条件,每个任务的时限将得到保证(C_i 是任务 i 的最坏情况下的执行时间,T_i 是任务 i 的执行周期).

$$\frac{C_1}{T_1}+\frac{C_2}{T_2}+\frac{C_3}{T_3}+\cdots+\frac{C_n}{T_n}\leq n(2^{1/n}-1)$$

对于非周期性的任务,可直接按照最高优先级抢占当前任务,凡每个周期内最多只能执行固定的时间片.

实时模块调度模块采用线程作为调度的基本单位[4],利用多线程实现 I/O 和处理机的并行执行,因为线程之间切换开销小,用规范的线程并行编程与异步调用方式相比,在不增加开销的前提下带来程序结构化好,直观易懂等好处.其原理是在系统中安排许多调度时钟,以调度时钟作为线程调度的参考,以线程创建时指定的优先级作为调度依据.

3.4 保证实时 Linux 实时性的其他模块

为确保硬实时,Linux 实时内核还包括以下模块:

1) rtlkm_time.o 该模块用于控制处理器的时钟,提供一个时钟处理的抽象接口,由于中断延迟可能给系统时钟造成一定的误差,因此设计时可将实时时钟中断的优先级设置的很高,通常仅次于调电中断.2) rtlkm_fifo.o 该模块提供了实时任务间、实时任务和普通 Linux 进程的接口,通过一个普通的 Linux 进程可以读写的 FIFO 设备进行通信.3) rtlkm_posix.o 提供使用 POSIX 方式的设备驱动程序的读、写、打开等系统调用.4) 普通 Linux 内核采用的虚拟内存管理机制影响 linux 的实时性、屏蔽该机制可通过修改文件系统中响应的文件来实现.

4 实时性能测试结果

测试数据来自系统性能分析工具 TraceToolKit 的测试结果(见表 1).

表 1 实时性能测试结果主要数据

Tab.1 Main data of real-time test result ms

测试对象	中断延迟	系统上下文切换时间	系统响应时间
基于普通 Linux 的联锁系统	5	20	40
基于实时 Linux 的联锁系统	<0.012	<0.026	<0.05

从上面的比较可以看出来,实时化改造后的实时 Linux 计算机联锁系统完全可以胜任实时运算操作系统,完全可以满足 250 km/h 以上速度的需要.

参考文献:

[1] 董昱,李敬文,李亮.基于 Linux 联锁计算机系统的实时性和可靠性研究[J].兰州交通大学学报,2004,23(6):70-73

[2] 倪继利.Linux 内核分析及编程[M].北京:电子工业出版社,2005.

[3] 赵志熙.计算机联锁系统技术[M].北京:中国铁道出版社,1999.

[4] 夏卫民,罗宇,吴庆波,等.并行操作系统原理与技术[M].北京:国防出版社,2002.

Analysis and Improvement on Real-time Based on Linux Interlocking Computer System

You Jianhong, Dong Yu, Li Zhijun

(School of Electronic and Information, Lanzhou Jiaotong University, Lanzhou 730070, China;
School of Computer, Qingdao Technological University, Qingdao 266033, China)

Abstract: The microcomputer interlocking system is a kind of control system, which guarantees the train's safety and high efficiency in scope of the station. Therefore, the system should have higher reliability and safety. The first should think about the problem of the real-time. The analysis has about the real-time of the microcomputer interlocking system. The analysis is also given according with the real-time module of the RT Linux, and the new system is compared with the former.

Keywords: RTLinux; Real-time; Kernel; Process Schedule; Microcomputer Interlocking

嵌入式资源免费下载

总线协议：

1. 基于 PCIe 驱动程序的数据传输卡 DMA 传输
2. 基于 PCIe 总线协议的设备驱动开发
3. CANopen 协议介绍
4. 基于 PXI 总线 RS422 数据通信卡 WDM 驱动程序设计
5. FPGA 实现 PCIe 总线 DMA 设计
6. PCI Express 协议实现与验证
7. VPX 总线技术及其实现
8. 基于 Xilinx FPGA 的 PCIE 接口实现
9. 基于 PCI 总线的 GPS 授时卡设计
10. 基于 CPCI 标准的 6U 信号处理平台的设计
11. USB30 电路保护
12. USB30 协议分析与框架设计
13. USB 30 中的 CRC 校验原理及实现
14. 基于 CPLD 的 UART 设计
15. IPMI 在 VPX 系统中的应用与设计
16. 基于 CPCI 总线的 PMC 载板设计
17. 基于 VPX 总线的工件台运动控制系统研究与开发
18. PCI Express 流控机制的研究与实现
19. UART16C554 的设计
20. 基于 VPX 的高性能计算机设计
21. 基于 CAN 总线技术的嵌入式网关设计
22. Visual C 串行通讯控件使用方法与技巧的研究
23. IEEE1588 精密时钟同步关键技术研究
24. GPS 信号发生器射频模块的一种实现方案
25. 基于 CPCI 接口的视频采集卡的设计
26. 基于 VPX 的 3U 信号处理平台的设计
27. 基于 PCI Express 总线 1394b 网络传输系统 WDM 驱动设计
28. AT89C52 单片机与 ARINC429 航空总线接口设计
29. 基于 CPCI 总线多 DSP 系统的高速主机接口设计
30. 总线协议中的 CRC 及其在 SATA 通信技术中的应用
31. 基于 FPGA 的 SATA 硬盘加解密控制器设计
32. Modbus 协议在串口通讯中的研究及应用
33. 高可用的磁盘阵列 Cache 的设计和实现
34. RAID 阵列中高速 Cache 管理的优化

35. 一种新的基于 RAID 的 CACHE 技术研究与实现
36. 基于 PCIE-104 总线的高速数据接口设计
37. 基于 VPX 标准的 RapidIO 交换和 Flash 存储模块设计
38. 北斗卫星系统在海洋工程中的应用
39. 北斗卫星系统在远洋船舶上应用的研究
40. 基于 CPCI 总线的红外实时信号处理系统
41. 硬件实现 RAID 与软件实现 RAID 的比较
42. 基于 PCI Express 总线系统的热插拔设计
43. 基于 RAID5 的磁盘阵列 Cache 的研究与实现
44. 基于 PCI 总线的 MPEG2 码流播放卡驱动程序开发
45. 基于磁盘异或引擎的 RAID5 小写性能优化
46. 基于 IEEE1588 的时钟同步技术研究
47. 基于 Davinci 平台的 SD 卡读写优化
48. 基于 PCI 总线的图像处理及传输系统的设计
49. 串口和以太网通信技术在油液在线监测系统中的应用
50. USB30 数据传输协议分析及实现

VxWorks：

1. 基于 VxWorks 的多任务程序设计
2. 基于 VxWorks 的数据采集存储装置设计
3. Flash 文件系统分析及其在 VxWorks 中的实现
4. VxWorks 多任务编程中的异常研究
5. VxWorks 应用技巧两例
6. 一种基于 VxWorks 的飞行仿真实时管理系统
7. 在 VxWorks 系统中使用 TrueType 字库
8. 基于 FreeType 的 VxWorks 中文显示方案
9. 基于 Tilcon 的 VxWorks 简单动画开发
10. 基于 Tilcon 的某武器显控系统界面设计
11. 基于 Tilcon 的综合导航信息处理装置界面设计
12. VxWorks 的内存配置和管理
13. 基于 VxWorks 系统的 PCI 配置与应用
14. 基于 MPC8270 的 VxWorks BSP 的移植
15. Bootrom 功能改进经验谈
16. 基于 VxWorks 嵌入式系统的中文平台研究与实现
17. VxBus 的 A429 接口驱动
18. 基于 VxBus 和 MPC8569E 千兆网驱动开发和实现
19. 一种基于 vxBus 的 PPC 与 FPGA 高速互联的驱动设计方法
20. 基于 VxBus 的设备驱动开发

21. 基于 VxBus 的驱动程序架构分析
22. 基于 VxBus 的高速数据采集卡驱动程序开发
23. Vxworks 下的冗余 CAN 通讯模块设计
24. WindML 工业平台下开发 S1d13506 驱动及显示功能的实现
25. WindML 中 Mesa 的应用
26. VxWorks 下图形用户界面开发中双缓冲技术应用
27. VxWorks 上的一种 GUI 系统的设计与实现
28. VxWorks 环境下 socket 的实现
29. VxWorks 的 WindML 图形界面程序的框架分析
30. VxWorks 实时操作系统及其在 PC104 下以太网编程的应用
31. 实时操作系统任务调度策略的研究与设计
32. 军事指挥系统中 VxWorks 下汉字显示技术

Linux：

1. Linux 程序设计第三版及源代码
2. NAND FLASH 文件系统的设计与实现
3. 多通道串行通信设备的 Linux 驱动程序实现
4. Zsh 开发指南-数组
5. 常用 GDB 命令中文速览
6. 嵌入式 C 进阶之道
7. Linux 串口编程实例
8. 基于 Yocto Project 的嵌入式应用设计
9. Android 应用的反编译
10. 基于 Android 行为的加密应用系统研究
11. 嵌入式 Linux 系统移植步步通
12. 嵌入式 CC++语言精华文章集锦
13. 基于 Linux 的高性能服务器端的设计与研究
14. S3C6410 移植 Android 内核
15. Android 开发指南中文版
16. 图解 Linux 操作系统架构设计与实现原理（第二版）
17. 如何在 Ubuntu 和 Linux Mint 下轻松升级 Linux 内核
18. Android 简单 mp3 播放器源码
19. 嵌入式 Linux 系统实时性的研究
20. Android 嵌入式系统架构及内核浅析
21. 基于嵌入式 Linux 操作系统内核实时性的改进方法研究
22. Linux TCP IP 协议详解
23. Linux 桌面环境下内存去重技术的研究与实现
24. 掌握 Android 7.0 新增特性 Quick Settings

25. Android 应用逆向分析方法研究
26. Android 操作系统的课程教学
27. Android 智能手机操作系统的研究
28. Android 英文朗读功能的实现
29. 基于 Yocto 订制嵌入式 Linux 发行版
30. 基于嵌入式 Linux 的网络设备驱动设计与实现
31. 如何高效学习嵌入式
32. 基于 Android 平台的 GPS 定位系统的设计与实现
33. LINUX ARM 下的 USB 驱动开发
34. Linux 下基于 I2C 协议的 RTC 驱动开发
35. 嵌入式下 Linux 系统设备驱动程序的开发
36. 基于嵌入式 Linux 的 SD 卡驱动程序的设计与实现
37. Linux 系统中进程调度策略
38. 嵌入式 Linux 实时性方法

Windows CE：

1. Windows CE.NET 下 YAFFS 文件系统 NAND Flash 驱动程序设计
2. Windows CE 的 CAN 总线驱动程序设计
3. 基于 Windows CE.NET 的 ADC 驱动程序实现与应用的研究
4. 基于 Windows CE.NET 平台的串行通信实现
5. 基于 Windows CE.NET 下的 GPRS 模块的研究与开发
6. win2k 下 NTFS 分区用 ntldr 加载进 dos 源代码
7. Windows 下的 USB 设备驱动程序开发
8. WinCE 的大容量程控数据传输解决方案设计
9. WinCE6.0 安装开发详解
10. DOS 下仿 Windows 的自带计算器程序 C 源码
11. G726 局域网语音通话程序和源代码
12. WinCE 主板加载第三方驱动程序的方法
13. WinCE 下的注册表编辑程序和源代码
14. WinCE 串口通信源代码
15. WINCE 的 SD 卡程序[可实现读写的源码]
16. 基于 WinCE 的 BootLoader 研究
17. Windows CE 环境下无线网卡的自动安装
18. 基于 Windows CE 的可视电话的研究与实现
19. 基于 WinCE 的嵌入式图像采集系统设计
20. 基于 ARM 与 WinCE 的掌纹鉴别系统

PowerPC：

1. Freescale MPC8536 开发板原理图
2. 基于 MPC8548E 的固件设计
3. 基于 MPC8548E 的嵌入式数据处理系统设计
4. 基于 PowerPC 嵌入式网络通信平台的实现
5. PowerPC 在车辆显控系统中的应用
6. 基于 PowerPC 的单板计算机的设计
7. 用 PowerPC860 实现 FPGA 配置
8. 基于 MPC8247 嵌入式电力交换系统的设计与实现
9. 基于设备树的 MPC8247 嵌入式 Linux 系统开发
10. 基于 MPC8313E 嵌入式系统 UBoot 的移植
11. 基于 PowerPC 处理器 SMP 系统的 UBoot 移植
12. 基于 PowerPC 双核处理器嵌入式系统 UBoot 移植
13. 基于 PowerPC 的雷达通用处理机设计
14. PowerPC 平台引导加载程序的移植
15. 基于 PowerPC 嵌入式内核的多串口通信扩展设计
16. 基于 PowerPC 的多网口系统抗干扰设计
17. 基于 MPC860T 与 VxWorks 的图形界面设计
18. 基于 MPC8260 处理器的 PPMC 系统

ARM：

1. 基于 DiskOnChip 2000 的驱动程序设计及应用
2. 基于 ARM 体系的 PC-104 总线设计
3. 基于 ARM 的嵌入式系统中断处理机制研究
4. 设计 ARM 的中断处理
5. 基于 ARM 的数据采集系统并行总线的驱动设计
6. S3C2410 下的 TFT LCD 驱动源码
7. STM32 SD 卡移植 FATFS 文件系统源码
8. STM32 ADC 多通道源码
9. ARM Linux 在 EP7312 上的移植
10. ARM 经典 300 问
11. 基于 S5PV210 的频谱监测设备嵌入式系统设计与实现
12. Uboot 中 start.S 源码的指令级的详尽解析

13. 基于 ARM9 的嵌入式 Zigbee 网关设计与实现
14. 基于 S3C6410 处理器的嵌入式 Linux 系统移植
15. CortexA8 平台的μC-OSⅡ及 LwIP 协议栈的移植与实现
16. 基于 ARM 的嵌入式 Linux 无线网卡设备驱动设计
17. ARM S3C2440 Linux ADC 驱动
18. ARM S3C2440 Linux 触摸屏驱动
19. Linux 和 Cortex-A8 的视频处理及数字微波传输系统设计
20. Nand Flash 启动模式下的 Uboot 移植
21. 基于 ARM 处理器的 UART 设计
22. ARM CortexM3 处理器故障的分析与处理
23. ARM 微处理器启动和调试浅析
24. 基于 ARM 系统下映像文件的执行与中断运行机制的实现
25. 中断调用方式的 ARM 二次开发接口设计
26. ARM11 嵌入式系统 Linux 下 LCD 的驱动设计

Hardware：

1. DSP 电源的典型设计
2. 高频脉冲电源设计
3. 电源的综合保护设计
4. 任意波形电源的设计
5. 高速 PCB 信号完整性分析及应用
6. DM642 高速图像采集系统的电磁干扰设计
7. 使用 COMExpress Nano 工控板实现 IP 调度设备
8. 基于 COM Express 架构的数据记录仪的设计与实现
9. 基于 COM Express 的信号系统逻辑运算单元设计
10. 基于 COM Express 的回波预处理模块设计
11. 基于 X86 平台的简单多任务内核的分析与实现
12. 基于 UEFI Shell 的 PreOS Application 的开发与研究
13. 基于 UEFI 固件的恶意代码防范技术研究
14. MIPS 架构计算机平台的支持固件研究
15. 基于 UEFI 固件的攻击验证技术研究
16. 基于 UEFI 的 Application 和 Driver 的分析与开发
17. 基于 UEFI 的可信 BIOS 研究与实现
18. 基于 UEFI 的国产计算机平台 BIOS 研究
19. 基于 UEFI 的安全模块设计分析
20. 基于 FPGA NiosⅡ的等精度频率计设计
21. 基于 FPGA 的 SOPC 设计
22. 基于 SOPC 基本信号产生器的设计与实现

23. 基于龙芯平台的 PMON 研究与开发

Programming:

1. 计算机软件基础数据结构 - 算法
2. 高级数据结构对算法的优化
3. 零基础学算法
4. Linux 环境下基于 TCP 的 Socket 编程浅析
5. Linux 环境下基于 UDP 的 socket 编程浅析
6. 基于 Socket 的网络编程技术及其实现