

网络通信软件设计指南

朱三元 杨 明 薛 钊 编著
白英彩 主审

清华大学出版社

(京)新登字 158 号

内 容 简 介

作者根据多年从事计算机网络通信软件设计与研制的成果和经验,并综合国内外同行的新成果,著述成这本实用性很强的设计指南。内容包括进程间通信 IPC 及其调用、传送层接口程序设计、网络系统远程处理、行式打印机假脱机、网关和网际、异种机联网技术、TCP/IP 协议系列的 UNIX 实现、NetWare 和 LAN Manager 驱动程序设计等。

版权所有,翻印必究。

本书封面贴有清华大学出版社
激光防伪标签,无标签者不得销售。

网络通信软件设计指南

朱三元 杨 明 薛 钊 编著

白英彩 主审

责任编辑 蔡鸿程

☆

清华大学出版社出版

北京 清华园 邮编 100084

国防工业出版社印刷厂印刷

新华书店总店科技发行所发行

☆

开本: 787×1092 1/16 印张: 26 字数: 613 千字

1994 年 2 月第 1 版 1995 年 3 月第 2 次印刷

印数: 5001—10000

ISBN 7-302-01387-X/TP·533

定价: 26.00 元

代 序

计算机网络是电子计算机技术与通讯技术日益发展及其密切结合的产物,网络化是计算机技术 90 年代的重要发展趋势之一。所谓计算机网络通常是指地理上分离的、具有独立功能的多个计算机系统通过各种通信手段、按不同的拓扑结构连接起来的网络,其主要特点是:多个计算机系统结合在一起,实现彼此通信和资源共享;不受地理和环境的限制;同时为多个用户服务。

90 年代计算机网络的新发展是:异机种网络和异网互联将有较大突破;复杂的网络系统与相适配的数据库系统密切结合构成实现办公自动化(OA)、工厂自动化(FA)的集成化系统;将在网络方面普遍地采用 Client-Server(客户机-服务器)先进结构,它构成一种日趋流行的分布式运算环境;若干个局部网络(LAN)彼此互连或者他们与大主机相连,构成规模更大的网络或形成广域网(WAN);在 LAN 方面,目前 Novell 公司的 NetWare 局网占据了市场的较大份额,但 Microsoft 公司的 LAN Manager 局网将后来居上,这两种局网形成世界局网的两大支柱。

计算机网络大发展的结果,也要求各种各样的网络通信软件有一个飞跃的发展,本书作者正是为了适应这样的形势需要而编著的,作者亲自投身于网络通信软件的设计与研制,本书作者杨明率领一个研制小组在一个较短的时间内研制出我国拥有自己版权的第一个 TCP/IP 软件包业已投放到国际市场,受到国内外专家的好评。这足以证明我国在发展软件产业方面有着巨大潜力和广阔前景!在编著本书过程中,陈宇同志给予了许多帮助,并完成了大量工作,尤其要感谢责任编辑同志对于本书在编著过程给予的诚挚的支持和具体指导。由于时间仓促,作者和审者水平有限,书中定有不少欠妥之处,恳请读者不吝指正。

白英彩

1993 年 4 月

目 录

第一章 绪论	1
1.1 概述	1
1.2 网络通信规程	3
1.3 开发网络通信软件的平台	4
第二章 进程间通信 IPC 及其调用	5
2.1 概述	5
2.2 文件和记录锁定	5
2.2.1 示例程序及其说明.....	5
2.2.2 锁定中的几个概念.....	7
2.2.3 System V Release 2 的咨询锁定	7
2.2.4 4.3 BSD 的咨询式锁定.....	8
2.2.5 UNIX 的其它上锁技术.....	9
2.3 管道.....	13
2.4 FIFOs	17
2.5 与 IPC 有关的概念和结构定义	20
2.5.1 流和消息	20
2.5.2 名字空间	21
2.5.3 IPC_perm 结构及 IPC 通道的创建	22
2.6 消息队列.....	25
2.7 信号灯.....	32
2.8 共享内存.....	44
第三章 传送层接口程序设计	53
3.1 BSD 4.3 的套接字接口 socket	53
3.1.1 概述	53
3.1.2 套接字地址	54
3.1.3 基本的套接字系统调用	56
3.1.4 高级套接字系统调用	60
3.1.5 套接字使用示例	72
3.2 System V 的传送层接口 TLI	82
3.2.1 概述	82
3.2.2 传送端点地址	83
3.2.3 基本 TLI 函数	84
3.2.4 高级 TLI 函数	93

3.2.5	流的概念	96
3.2.6	TLI 使用示例	98
第四章	网络系统远程处理	106
4.1	引言	106
4.2	远程命令执行	106
4.2.1	rcmd 函数和 rshd 服务器	107
4.2.2	rexec 函数和 rexecd 服务器	114
4.3	远程登录	115
4.3.1	终端行律与伪终端	115
4.3.2	终端方式字与控制终端	119
4.3.3	rlogin 概述	122
4.3.4	窗口环境	122
4.3.5	流控制与伪终端方式字	123
4.3.6	rlogin 客户程序	125
4.3.7	rlogin 服务器	126
4.4	远程介质的访问	128
4.4.1	UNIX 磁带驱动器的处理	129
4.4.2	rmt 协议	130
4.4.3	rmt 服务器设计分析	132
4.5	远程过程调用	135
4.5.1	远程过程调用的处理过程	136
4.5.2	远程过程调用的有关问题	137
4.5.3	远程过程调用传送协议	138
4.5.4	Sun RPC	139
4.5.5	Xerox Courier	143
4.5.6	Apollo RPC	146
4.5.7	小结	146
第五章	行式打印机假脱机	147
5.1	设计原则	147
5.2	打印机配置及分类	148
5.3	4.3 BSD 打印假脱机单机系统	148
5.4	4.3 BSD 打印假脱机单机系统示例	150
5.5	4.3 BSD 假脱机系统远程打印	152
5.6	远程打印示例	155
5.7	远程打印的总体设计	156
5.8	System V 打印假脱机系统	156
5.9	System V 打印假脱机系统示例	158
第六章	Gateway (网关)与 Internet(网际)	160

6.1	核心网关系统	160
6.1.1	网关与路由选择表	160
6.1.2	网关到网关协议(GGP)	161
6.1.3	GGP 报文格式	162
6.2	自治系统与联盟(EGP)	164
6.2.1	自治系统的概念	164
6.2.2	外部网关协议	166
6.2.3	EGP 报文报头	166
6.2.4	EGP 相邻搜索报文	167
6.2.5	EGP 相邻可达性报文	167
6.2.6	EGP 轮询请求报文和路由选择更新报文	168
6.2.7	EGP 的关键限制	170
6.2.8	自治联盟的概念	171
6.3	内部网关协议	171
6.3.1	静态和动态内部路由	171
6.3.2	路由选择信息协议(RIP)	172
6.3.3	RIP 报文格式	173
6.3.4	HELLO 协议	174
第七章	异种机联网技术	176
7.1	异种机互连的概念	176
7.2	信关与桥	177
7.3	协议转换	178
7.4	典型的网际互连例子	178
第八章	TCP/IP 协议系列的 UNIX 实现	180
8.1	DOD 网络体系结构	180
8.2	4.3 BSD UNIX 网络环境简介	180
8.3	4.3 BSD UNIX 网络环境剖析	181
8.4	网际协议 IP	186
8.4.1	IP 在 UNIX 互连网域中的作用	186
8.4.2	IP 与互连网域上层协议的连接	187
8.4.3	IP 与互连网域下层子网协议的连接	189
8.4.4	IP 的报文格式	190
8.4.5	IP 的分段和重装算法	192
8.4.6	IP 的路由	195
8.4.7	IP 的选项	199
8.4.8	IP 协议的完整实现	203
8.4.8.1	IP 协议模块的初始化函数 ip_init	203
8.4.8.2	IP 协议模块的输出函数 ip_output	203

8.4.8.3	IP 协议模块的接收函数 ipintr	203
8.4.8.4	IP 协议模块的超时处理函数 ip_slowtimo	204
8.4.8.5	IP 协议模块的异常空间释放函数 ip_drain	204
8.5	网际控制报文协议 ICMP	204
8.5.1	ICMP 在 UNIX 互连网域中的作用	204
8.5.2	ICMP 的报文格式	205
8.5.3	ICMP 协议的实现	210
8.5.3.1	ICMP 协议模块的输入函数 icmp_input	210
8.5.3.2	ICMP 协议模块的套接字用户接口函数 raw_usrreq	211
8.5.3.3	ICMP 协议模块的输出函数 rip_output	212
8.5.3.4	raw_input	213
8.5.3.5	rawintr	213
8.6	用户数据报协议 UDP	213
8.6.1	UDP 在 UNIX 互连网域中的作用	213
8.6.2	UDP 的报文格式	214
8.6.3	UDP 协议的协议控制块	214
8.6.4	UDP 协议的实现	215
8.6.4.1	UDP 协议模块的初始化函数 udp_init	215
8.6.4.2	UDP 协议模块的输入函数 udp_input	215
8.6.4.3	UDP 协议模块的套接字用户接口函数 udp_usrreq	215
8.6.4.4	UDP 协议模块的输出函数 udp_output	218
8.6.4.5	UDP 协议模块的控制输入函数 udp_ctlinput	218
8.7	运输控制协议 TCP	218
8.7.1	TCP 在 UNIX 互连网域中的作用	218
8.7.2	TCP 的报文格式	218
8.7.3	TCP 协议的状态转换图	221
8.7.4	TCP 协议的协议控制块	223
8.7.5	TCP 的协议机制及实现策略	225
8.7.5.1	序号	225
8.7.5.2	初始序号及“平静”时间	226
8.7.5.3	连接建立	227
8.7.5.4	连接关闭	229
8.7.5.5	窗口式流量控制	231
8.7.5.6	“推进”数据	233
8.7.5.7	紧急数据	233
8.7.5.8	会话连接的重置	234
8.7.5.9	TCP 的多路复用机制	235

8.7.5.10	TCP 的优先级和安全性	235
8.7.5.11	TCP 的内部时钟	235
8.7.6	TCP 协议的具体实现	237
8.7.6.1	TCP 协议模块的初始化函数 tcp_init	237
8.7.6.2	TCP 协议模块的输入函数 tcp_input	237
8.7.6.3	TCP 协议模块的套接字用户接口函数 tcp_usrreq	243
8.7.6.4	TCP 协议模块的输出函数 tcp_output	246
8.7.6.5	TCP 协议模块的“快超时”处理函数 tcp_fasttimo	249
8.7.6.6	TCP 协议模块的“慢超时”处理函数 tcp_slowtimo	249
8.7.6.7	TCP 协议模块的内部时钟超时处理函数 tcp_timer	249
8.7.6.8	TCP 协议模块的控制输入函数 tcp_ctlinput	251
8.7.6.9	TCP 协议模块的异常空间释放函数 tcp_drain	251
第九章	X.25 在 UNIX 操作系统上的实现	252
9.1	X.25 建议	252
9.1.1	X.25 的发展动态	252
9.1.2	分组交换的概念	252
9.1.3	X.25 建议的内容	253
9.1.3.1	DTE/DCE 物理级的接口特性	253
9.1.3.2	通过 DTE/DCE 接口的链路接入规程	254
9.1.3.3	分组级 DTE/DCE 接口描述及虚电路业务规程	259
9.1.3.3.1	虚呼叫的建立和消除过程	260
9.1.3.3.2	数据传输和流量控制	260
9.1.3.3.3	复位和重新启动过程	260
9.2	基于 UNIX 的 X.25 的实现	261
9.2.1	物理级的实现	261
9.2.1.1	SUN 工作站 A/B 口及 Zilog 8530 芯片介绍	261
9.2.1.2	与物理级实现有关的数据结构	261
9.2.1.3	中断系统及中断服务程序结构	263
9.2.2	链路级的实现	264
9.2.2.1	链路级的总体结构	264
9.2.2.2	链路级总控模块的状态转换图及程序结构	264
9.2.2.3	链路级协议控制块的结构	266
9.2.3	分组级的实现	267
9.2.3.1	分组级的总体结构	267
9.2.3.2	分组级的状态转换图	268
9.2.3.3	重新开始规程控制块	269
9.2.3.4	虚电路业务规程控制块	270
9.2.4	X.25 与 UNIX 操作系统的接口模块	272

9.2.4.1	与互连网域的连接方法	272
9.2.4.2	创建新的 X.25 域	273
第十章	NetWare 驱动程序设计指南	274
10.1	NetWare 协议和驱动程序简介	274
10.2	NetWare ODI 的服务器驱动程序	276
10.2.1	概述	276
10.2.2	OS 驱动程序中用到的数据结构	276
10.2.3	开发者需研制的处理过程	278
10.2.4	驱动程序的支持处理过程	300
10.2.5	NetWare 服务器驱动程序的制作	301
10.3	NetWare ODI 的工作站驱动程序	302
10.3.1	概述	302
10.3.2	MSM 的数据结构和变量	304
10.3.3	开发者在〈HSM〉中需开发的处理过程	305
10.3.4	DOS ODI LAN 驱动程序的制作	311
第十一章	LAN Manager 驱动程序设计指南	313
11.1	概论	313
11.1.1	设备驱动程序介绍	313
11.1.2	设备驱动程序的安装	320
11.1.3	DevHlp 服务的作用	319
11.2	NDIS(网络驱动程序接口规范)	319
11.2.1	NDIS 与 OSI 参考模型	319
11.2.2	NDIS 驱动程序的构成说明	321
11.2.3	协议管理程序	321
11.2.4	VECTOR 和动态装配	324
11.2.5	协议与 MAC 模块间的交互机制	328
11.3	LAN Manager 驱动程序中用到的数据结构	330
11.3.1	模块特性表	330
11.3.2	公共特性表	331
11.3.3	MAC 特定服务特性表	332
11.3.4	MAC 特定服务状态表	336
11.3.5	802.3 特定介质统计表	337
11.3.6	802.5 特定介质统计表	338
11.3.7	MAC 高层调度表	338
11.3.8	协议特定服务特性表	339
11.3.9	协议低层调度表	339
11.3.10	NetBIOS 驱动程序的特性表	339
11.3.11	帧数据描述	341

11.3.12	PROTOCOL.INI	342
11.3.13	配置存储器影象	344
11.4	协议/MAC 间的交互原语	347
11.4.1	直接原语	349
11.4.2	通用请求	354
11.4.3	通用请求确认	362
11.4.4	状态指示	363
11.4.5	Interrupt	366
11.4.6	系统请求	366
11.4.7	协议管理程序原语	369
11.5	MAC 驱动程序的编写	378
11.5.1	MAC 驱动程序的主要构成	378
11.5.2	策略程序和初始化程序的编写	379
11.5.3	初始化程序	379
11.5.4	系统请求程序的编写	387
11.5.5	指示程序的编写	389
11.5.6	Transmit Chain 程序的编写	391
11.5.7	中断程序的编写	393
11.5.8	通用请求程序的编写	402
参考文献		403

第一章 绪 论

1.1 概 述

所谓计算机网络就是通过通信线路互相连接最小自治的计算机的集合。它是由计算机及其外围设备、数据通信和终端设备等构成的一个群体。目前,计算机网络大部分都是多台计算机系统之间能互连、通信,达到资源共享的目的网络系统,它是电子计算机及其应用技术与通信技术日益发展且两者密切结合的产物。

计算机的通信通常有两种方式:其一,通过双绞线、同轴电缆、电话线或光缆等有形传输介质而互连实现通信;其二,通过激光、微波、地球卫星等无形介质实现无线通讯。前者是目前使用得最普通的形式,后者将是今后的最主要发展方向,因为九十年代以来,出现了各种各样的便携式微机,诸如 Notebooks 和由一种 Minimodules 构成的嵌入式微型计算机(其 Coremodule 尺寸为 90mm×96mm),各种运输设备(如飞机,舰船等)均可以嵌入这种计算机或相应的嵌入式操作系统,实现调度和控制作用,甚至这种微小型化的计算机可在野外旅途中以及偏远的乡村使用。这就对采用无线通信联网产生极为迫切的需求,可以预计,随着微机的这种进一步小型化和嵌入式结构的发展,无线网络将会进一步的发展。

(一) 计算机网络的研制始于 60 年代中期,至今已有 20 多年的历史,其网络技术发展及其应用已十分普及,已渗到各个领域。并正在日益显示着它对信息化社会所带来的影响和深远的意义。

在网络发展上,最早出现的是分布在很大的地理范围内的远程网络(Wide Area Network, WAN),例如美国国防部高级研究计划局首先研制的 ARPA 网。它从 1969 年建立,至今已发展成为跨越几大洲的巨型网络。

70 年代中期由于微型计算机的出现和微处理机的出现,以及短程通讯技术的飞快发展,两者相辅相成,又促进以微机为基础的各种局部网络(Local Area Network, LAN)的飞快发展,1975 年美国 Xerox 公司首先推出了 Ethernet,与此同时英国剑桥大学研制成剑桥环网,它们是 LAN 的代表。

LAN 与 WAN 有所区别,其特点是:

- (1) 有限的地理范围,通常网内的计算机限于一个大楼、楼群或一个企业及单位。
- (2) 较高的通信速率,大都在每秒 1—100 Mbps,而 WAN 多在几十 Kbps。
- (3) 通讯介质多样。
- (4) 通常为一个部门所拥有。

特别是 80 年代以来,以微机为基础 LAN 技术有了极其迅速的发展。

(二) 90 年代计算机网络化趋势尤为明显。据称 1978 年全世界约有 700 万人每天使用计算机,而到 1988 年上升到 5000 万人,目前全世界已拥有计算机逾一亿台,预计每

天上机的人次可达 2 亿以上。计算机的性能价格比以每年 25% 的速度在提高。微机的应用已渗透到国民经济的各个部门,乃至家庭和个人。这标志着正步入信息时代,世界范围内的社会信息数据正以每年增长 40% 到 45% 的年增长率在增加,这就是迫切实现网络化的动力源泉,据称,约有 65% 的计算机要联网或已联网,以求彼此通信,达到资源共享的目标。

90 年代计算机网络化更加向深度和广度方向发展。人们要求网络传输的内容范围增加,诸如数据之外,还需传输声音、图形、图像和文字,这就是以网络为基础的多媒体技术,使网络的应用广度更加扩大,并最终为信息化社会的实现所必须的 ISDN(Integrated Service Data Network)奠定基础。

(三)当前国际 LAN 的市场上,两雄称霸,龙争虎斗的局面,将可能持续相当长一段时间。

正如大家知道那样,80 年代中后期美国 Novell 公司先是以“一花独秀,压倒群芳”之势占据了国际 LAN 市场的 60% 以上,一路领先,扶摇直上,尤其是 NetWare 386 V3.11 版推出之后,受到普遍的注目;随后,国际上的软件公司魁首—Microsoft 公司先后推出了 LAN Manager V1.0(即 LAN 3+Open)、LAN Manager V2.0 和 V2.1,后来居上,成为当前世界 LAN 的两大支柱。1992 年 10 月 Microsoft 又抢先宣布了 LAN Manager V2.2,已更加领先于 Novell 的 NetWare 386 V3.11,但后者宣布今年初夏将推出 NetWare 4.0。可见,“龙争虎斗”一定要持续一个相当长的时期。

就 InfoWorld 杂志对 NetWare V3.11 和 LAN Manager V2.1 两个性能相近的网络技术性能进行了定量地全面评价,前者得分总和 5.8,而 LAN Manager 得分为 6.8。可见 LAN Manager 略高一筹,尽管 Novell 网当今在国际 LAN 市场上所占份额很大,但今后 LAN Manager 肯定是要雄心勃勃地争霸这个市场的。

Novell LAN 采用了“将网络协议软件与网络操作系统 NetWare 密切结合起来的设计思想”,可达到节省开销、提高运行效率之目标,Novell LAN 的最大特点是与其底层的网络适配器(Adapter 或称网卡)的无关性,即是说 NetWare 可虚拟地在所有流行的 LAN 上运行,使它成为一个理想的开发网络应用软件的平台,吸引了广大用户软件人员为之开发逾来逾多的应用软件。反过来又推动其发展,同时,Novell LAN 采用了开放性协议技术(OPT),允许各种网络协议紧密结合,进而在 NetWare 386 V3.11 版中采用了 NLM 模块的组合技术,可以实现异机种联网的难题。此外,Novell LAN 不需专用服务器,占用工作站的内存最小,使用方便,功能强,效率高,兼容性强,可靠性高,保密性强,容错性好。尤其在 NetWare 386 V3.11 版中实现了服务器软件的“分布式结构策略”、“横向信息共享”、“报文传送”技术、增添了“TCP/IP 栈”、实现了“SNA 协议栈”和“开放式数据链路接口”等一系列新技术,使 Novell LAN 更深入人心,扩大了市场。

与此同时,Microsoft 公司的 LAN Manager V2.1 和 V2.2 版除了具备 Novell LAN 一些通常的优点之外,还采用了“客户机/服务器”(Client-Server)的先进网络体系结构,以及基于多用户、多任务并发操作系统 OS/2 作为服务器的强大功能,并以 OS/2、Unix、VMS 和 Windows N.T. 作为开发平台,更便于异机种联网和异网互连。由于 LAN Manager 与 Windows(已公认优于 DOS)的紧密结合,使它具有更好的性能价格比,在技术

上胜过一筹,且随 Windows 逐步取代 DOS 的过程,LAN Manager 便很自然地渗透到 LAN 市场中去的策略,也将是不可抗拒的。

(四)在网络化技术迅速发展的今天,实用性能强的 TCP/IP 协议立下了汗马功劳。起先,TCP/IP(Transmission Control Protocol/Internet Protocol)是美国国防部于 70 年代提出的重大决策之一,将网络(当时主要是中大型机器连成的网络)互连起来,并按 TCP/IP 协议实现异网之间“数据通讯和资源共享”,接着美国国防部高级计划局(DARPA)于 70 年代末便提出了一系列的网际互联(Internet)技术。使得在科学研究、军事和社会生活迫切需要的大范围实现资源共享和交换信息得以实现。

TCP/IP 协议的基本思想是通过网间连接器(Gateway)将各种不同的网络连接起来,在各个网络的低层协议之上构造一个虚拟的大网,使用户与其他网的通讯就象与本网的主机通信一样方便。实现这一思想的重要协议是 TCP/IP,此外还有 ICMP (Internet Control Message Protocol)、UDP (User Datagram Protocol)、SMTP (Simple Mail Transfer Protocol)、ARP (Address Resolution Protocol)、RARP (Reverse Address Resolution Protocol)、GGP (Gateway-Gateway Protocol)、EGP (Exterior Gateway Protocol)、Telenet、FTP (File Transfer Protocol)、NSP (Name Server Protocol)等 10 多种协议,构成一个协议系列,它们互相协调来实现异网通信和资源共享。在本书中以及有关参考书中还将详细地阐明 TCP/IP 的机理。

国际标准化组织(ISO)对网络标准提出了 OSI/RM(开放系统互连七层协议的参考模型),这七层自低向高分别为物理层、数据链路层、网络层、传送层、会话层、表示层和应用层。而在此之前,DARPA 提出的 TCP/IP 仅仅提供了高四层标准,对低三层没有定义,因此 TCP/IP 在设计时必须解决与低层的接口问题。这也在本书中作了详细的介绍。

值得指出的是,作为大型网络互联的 TCP/IP 协议系列早已实现且有相应产品,但就其微机领域中的各种以微机为基础的不同 LAN 的互联所用到的 TCP/IP 工程设计,直到 90 年代伊始才提到议事日程上来的。因此,书中提供的通信程序设计思想、方法和实现原则是新颖而又极具参考价值的。

1.2 网络通信规程

网络通信规程(Protocol)又称通信协议。它是网络标准化的基础。

通信协议就是一组约定的集合,由它决定经由通信网络传输的信息或存储在报文和数据库中的信息格式和控制方式,这些信息可以是声音、报文、数据和图象。每一种类型的信息都需要一个为用户宜于了解的清晰的程序来定义。

通常,通信协议主要应该具有以下功能:数据交换,差错控制,信息编码,线路利用,同步,使通信数据具有透明性等等。采用协议的方式可以在用户分支交换器、计算机、字符处理器、工作站及通信装置间进行信息单元的有序交换。由协议程序制定一种对话,并能在通信链路的任何地方对其加以修正。

在网络领域中,“协议”代表着标准化的意思,这是一种早已为人们所共识的,通常

协议至少有两种功能：一种是通信，包括识别和同步；另一种是信息传输，包括传输正确性的保证，错误检测与修正等等。

通信协议具有层次性、可靠性、有效性等特点。

本书所阐述的网络通信软件的设计都必须是遵照上面协议的功能和特性来进行的。

1.3 开发网络通信软件的平台

准确地说，就是在设计网络通信软件时，一定要针对网络中所包含的机器的类型，即这些机器是分属于哪一类操作系统的。全世界已安装的上亿台机器，主要是 IBM 及其兼容机，它们的连网及其彼此通信的软件开发，也必须是基于 DOS 操作系统作为开发平台进行的；当然，Windows 乃至 Windows N.T. 的出现，并被公认为它优于 DOS 环境，那么今后开发网络通信软件也无疑地会转向以 Windows 为开发平台进行工作，对于 LAN Manager 网络而言，其基础是以能运行 OS/2 操作系统的机器（如高档 PS/2 系列机或高于 IBM 486 的机器）作为服务器的，因此为这种网络开发其通信软件也就自然要以 OS/2 为开发平台。当然，各式各样网络的互联，通过 TCP/IP 协议都可以实现，它是以 BSD 4.3 UNIX 或 SCO UNIX (System V) 为基础发展起来的，因此本书自然应该以大量篇幅围绕 UNIX 作为开发平台实现和解决一系列通信软件的设计问题。

第二章 进程间通信 IPC 及其调用

2.1 概 述

在传统的单任务操作系统上,程序设计的对象一经运行便将独占整个主机资源,程序实体的不同模块之间完全是通过全局变量、函数调用时的参数及返回值来进行通信的。

UNIX 操作系统是一个分时的多任务操作系统,在其上设计的每一个程序,运行后都将成为一个独立的实体,我们称之为进程,它们都在自己的地址空间内运行。这样 UNIX 进程间的通信就由两个不同的概念组成:其一是单一进程内部各模块间的通信,它沿袭了单任务操作系统中的程序模块通信方法;其二是作为独立单位的各不同进程间的通信,它必须以保证各进程在通信过程中互不干扰从而保持其通信的一致性为条件。

为不同进程间的通信提供相应的支持机制是 UNIX 操作系统的主要特点,本章将主要讨论包括文件和记录锁定、管道、FIFOs、消息队列、信号量和共享内存等在内的 UNIX 进程间通信。

2.2 文件和记录锁定

2.2.1 示例程序及其说明

```
#define SEQFILE    "sequo"      /* 文件名 */
#define MAXBUFF    100
main ()
{
    int fd,i,n,pid,seqno;
    char buff[MAXBUFF+1];
    pid=getpid();
    if ((fd=open(SEQFILE,2))<0)
        err_sys("can't open %s",SEQFILE);
    for(i=0;i<5;i++) {
        my_lock(fd);          /* 文件锁定 */
        lseek(fd,0L,0);        /* 文件指针回到文件头 */
        if((n=read(fd,buff,MAXBUFF))<=0)
            err_sys("read error");
        buff[n]='\0';
        if((n=sscanf(buff,"%d\n",&seqno))!=1)
            err_sys("sscanf error");
        printf("pid=%d,seq #=%d\n",pid,seqno);
        seqno++;
        sprintf(buff,"%03d\n",seqno);
        n=strlen(buff);
        lseek(fd,0L,0);
    }
```

```

        if(write(fd,buff,n) != n)
            err_sys("write error");
        my_unlock(fd);    /* 文件解锁 */
    }
    close(fd);
}

my_lock(fd)
int fd;
{
    return;
}

my_unlock(fd)
int fd;
{
    return;
}

```

在上面示例程序中文件锁定(my_lock)和文件解锁(my_unlock)并不执行任何上锁和解锁操作,因此该程序的工作将简化为打开序号文件,把序号文件中的序号加 1 后写回原文件,以及关闭序号文件等。

如果我们把序号文件“seqno”的内容初始化为 1,并且运行一次该程序,我们将得到如下结果:

```

pid=118,seq#=1
pid=118,seq#=2
pid=118,seq#=3
pid=118,seq#=4
pid=118,seq#=5

```

如果我们把序号文件再一次初始化为 1,并且用如下命令运行两次该程序:

a.out&a.out&,我们会看到这样的结果:

```

pid=186,seq#=1
pid=187,seq#=1
pid=187,seq#=2
pid=187,seq#=3
pid=187,seq#=4
pid=187,seq#=5

pid=186,seq#=2
pid=186,seq#=3
pid=186,seq#=4
pid=186,seq#=5

```

分析上面第二个运行结果,我们不难看出:序号文件“seqno”是该程序所产生的两个进程同时要求访问的共享资源。第一个进程读入序号文件输出 1,并准备增值后写回原文件时,第二个进程开始被操作系统调度,它读入“seqno”文件,然后把序号一直修改到 5 后退出,直到此时,第一个进程才重新获得 CPU,它把增值后的 2 写回原文件,继续以下操作,依次输出 2、3、4、5。

第二个结果并不是我们所期望的输出,究其原因,主要是由于在共享资源的访问中没有健全互斥性措施,从而引起了访问和操作的混乱。因此我们要求进程在对共享资源访问前必须对它进行锁定以避免其它进程对它的操作,当然在该进程访问完共享资源后也要进行解锁操作,以使其它进程能有机会占用共享资源。文件和记录锁定就是 UNIX 操作系统为共享资源提供的互斥性保障。

2.2.2 锁定中的几个概念

文件锁定的是整个文件,而记录锁定只锁定文件的某一特定部分。UNIX 的记录指的是从文件的某一相对位置开始的一段连续的字节流,它不同于其它以强制性记录结构组织文件的操作系统。因此,UNIX 记录锁更恰当的称呼应该是范围锁,它是对文件某个范围的锁定。

文件和记录锁定可分为咨询式锁定和强制锁定两种。当正在运行的某一进程对它将要访问的某一文件进行了咨询式锁定后,其它想要访问该文件的进程将被操作系统告知共享文件已经上了锁,但这并不阻止它们对锁定文件的操作。只要有对锁定文件的存取权,这些进程便可忽视咨询式锁定而去写上了锁的文件。强制锁定的含义则要严格多了,当某一共享文件被强制后,操作系统将会对每个读写文件的请求进行核查,只有在确证该请求不会干扰上了锁的文件时,才允许对应的操作。System V Release2 和 4.3BSD 都提供了咨询式锁定方式,而 System V Release 3 既提供了咨询式锁定方式又提供了强制性锁定方式,在缺省情况下,System V Release 3 提供的是咨询式锁定。

2.2.3 System V Release 2 的咨询锁定

System V 的锁函数 `lockf` 具有如下形式:

```
#include <unistd.h>
```

```
int lockf(int fd,int function,long size);
```

`fd` 是在文件打开操作中获得的文件描述符;

`function` 具有如下参数值:

`F_ULOCK` 为一个先前锁定的区域解锁;

`F_LOCK` 锁定一个区域;

`F_TLOCK` 测试并锁定一个区域;

`F_TEST3` 测试一个区域是否已经上锁;

`size` 指明了从文件当前位置开始的一段连续锁定区域的长度,当 `size` 为 0 时,锁定记录将由当前位置一直扩展到文件尾。

函数 `lockf` 即可用来上锁又可用来测试是否已经上了锁。如果 `lockf` 的参数 `FUNCTION` 为 `F_LOCK` 指定文件的对应区域已被其它进程锁定,那么 `lockf` 的调用进程将转为睡眠状态直到该区域解锁。上述情况我们称为阻塞。如果在调用 `lockf` 时把参数 `function` 设为 `F_TLOCK`,那么当被测试的区域上了锁时,`lockf` 便会立即返回 -1,出错返回码将为 `EAGAIN`,它是一个非阻塞调用。

```
if(lockf(fd,F_TEST,size)==0){
```