

Linux/UNIX

系统编程手册 (下册)

THE **LINUX**
PROGRAMMING
INTERFACE

A Linux and UNIX[®] System Programming Handbook

[德] Michael Kerrisk 著
郭光伟 陈舸 译



Linux/UNIX 系统编程手册 (下册)

THE LINUX
PROGRAMMING
INTERFACE

A Linux and UNIX[®] System Programming Handbook

[德] Michael Kerrisk 著
郭光伟 陈舸 译



人民邮电出版社
北京

目 录

第 34 章	进程组、会话和作业控制	573
34.1	概述	573
34.2	进程组	575
34.3	会话	577
34.4	控制终端和控制进程	578
34.5	前台和后台进程组	580
34.6	SIGHUP 信号	581
34.6.1	在 shell 中处理 SIGHUP 信号	581
34.6.2	SIGHUP 和控制进程的终止	583
34.7	作业控制	585
34.7.1	在 shell 中使用作业控制	585
34.7.2	实现作业控制	587
34.7.3	处理作业控制信号	591
34.7.4	孤儿进程组 (SIGHUP 回顾)	594
34.8	总结	598
34.9	习题	599
第 35 章	进程优先级和调度	600
35.1	进程优先级 (nice 值)	600
35.2	实时进程调度概述	603
35.2.1	SCHED_RR 策略	604
35.2.2	SCHED_FIFO 策略	605
35.2.3	SCHED_BATCH 和 SCHED_IDLE 策略	605
35.3	实时进程调用 API	605
35.3.1	实时优先级范围	606
35.3.2	修改和获取策略和优先级	606
35.3.3	释放 CPU	611

35.3.4	SCHED_RR 时间片	611
35.4	CPU 亲和力	612
35.5	总结	614
35.6	习题	615
第 36 章	进程资源	617
36.1	进程资源使用	617
36.2	进程资源限制	619
36.3	特定资源限制细节	623
36.4	总结	627
36.5	习题	627
第 37 章	DAEMON	628
37.1	概述	628
37.2	创建一个 daemon	629
37.3	编写 daemon 指南	632
37.4	使用 SIGHUP 重新初始化一个 daemon	632
37.5	使用 syslog 记录消息和错误	635
37.5.1	概述	635
37.5.2	syslog API	636
37.5.3	/etc/syslog.conf 文件	640
37.6	总结	641
37.7	习题	641
第 38 章	编写安全的特权程序	642
38.1	是否需要一个 Set-User-ID 或 Set-Group-ID 程序?	642
38.2	以最小权限操作	643
38.3	小心执行程序	645
38.4	避免暴露敏感信息	646
38.5	确定进程的边界	647
38.6	小心信号和竞争条件	647
38.7	执行文件操作和文件 I/O 的缺陷	648
38.8	不要完全相信输入和环境	648
38.9	小心缓冲区溢出	649
38.10	小心拒绝服务攻击	650
38.11	检查返回状态和安全地处理失败情况	651
38.12	总结	651
38.13	习题	652
第 39 章	能力	653
39.1	能力基本原理	653

39.2	Linux 能力	654
39.3	进程和文件能力	654
39.3.1	进程能力	654
39.3.2	文件能力	655
39.3.3	进程许可和有效能力集的目的	657
39.3.4	文件许可和有效能力集的目的	657
39.3.5	进程和文件可继承集的目的	658
39.3.6	在 shell 中给文件赋予能力和查看文件能力	658
39.4	现代能力实现	659
39.5	在 exec() 中转变进程能力	659
39.5.1	能力边界集	660
39.5.2	保持 root 语义	660
39.6	改变用户 ID 对进程能力的影响	661
39.7	用编程的方式改变进程能力	661
39.8	创建仅包含能力的环境	665
39.9	发现程序所需的能力	667
39.10	不具备文件能力的老式内核和系统	667
39.11	总结	669
39.12	习题	669
第 40 章	登录记账	670
40.1	utmp 和 wtmp 文件概述	670
40.2	utmpx API	671
40.3	utmpx 结构	671
40.4	从 utmp 和 wtmp 文件中检索信息	673
40.5	获取登录名称: getlogin()	676
40.6	为登录会话更新 utmp 和 wtmp 文件	677
40.7	lastlog 文件	681
40.8	总结	683
40.9	习题	683
第 41 章	共享库基础	684
41.1	目标库	684
41.2	静态库	685
41.3	共享库概述	686
41.4	创建和使用共享库——首回合	687
41.4.1	创建一个共享库	687
41.4.2	位置独立的代码	687
41.4.3	使用一个共享库	688
41.4.4	共享库 soname	689

41.5	使用共享库的有用工具	691
41.6	共享库版本和命名规则	692
41.7	安装共享库	694
41.8	兼容与不兼容库比较	696
41.9	升级共享库	697
41.10	在目标文件中指定库搜索目录	698
41.11	在运行时找出共享库	700
41.12	运行时符号解析	700
41.13	使用静态库取代共享库	701
41.14	总结	702
41.15	习题	703
第 42 章	共享库高级特性	704
42.1	动态加载库	704
42.1.1	打开共享库: <code>dlopen()</code>	705
42.1.2	错误诊断: <code>dlderror()</code>	706
42.1.3	获取符号的地址: <code>dlsym()</code>	707
42.1.4	关闭共享库: <code>dlclose()</code>	709
42.1.5	获取与加载的符号相关的信息: <code>dladdr()</code>	710
42.1.6	在主程序中访问符号	710
42.2	控制符号的可见性	710
42.3	链接器版本脚本	711
42.3.1	使用版本脚本控制符号的可见性	712
42.3.2	符号版本化	713
42.4	初始化和终止函数	715
42.5	预加载共享库	716
42.6	监控动态链接器: <code>LD_DEBUG</code>	716
42.7	总结	717
42.8	习题	718
第 43 章	进程间通信简介	719
43.1	IPC 工具分类	719
43.2	通信工具	720
43.3	同步工具	721
43.4	IPC 工具比较	723
43.5	总结	727
43.6	习题	727
第 44 章	管道和 FIFO	728
44.1	概述	728

44.2	创建和使用管道	730
44.3	将管道作为一种进程同步的方法	735
44.4	使用管道连接过滤器	737
44.5	通过管道与 Shell 命令进行通信: popen()	739
44.6	管道和 stdio 缓冲	743
44.7	FIFO	743
44.8	使用管道实现一个客户端/服务器应用程序	745
44.9	非阻塞 I/O	751
44.10	管道和 FIFO 中 read()和 write()的语义	752
44.11	总结	753
44.12	习题	754
第 45 章	System V IPC 介绍	756
45.1	概述	757
45.2	IPC Key	759
45.3	关联数据结构和对象权限	761
45.4	IPC 标识符和客户端/服务器应用程序	763
45.5	System V IPC get 调用使用的算法	764
45.6	ipcs 和 ipcrm 命令	766
45.7	获取所有 IPC 对象列表	767
45.8	IPC 限制	767
45.9	总结	768
45.10	习题	768
第 46 章	System V 消息队列	769
46.1	创建或打开一个消息队列	769
46.2	交换消息	771
46.2.1	发送消息	772
46.2.2	接收消息	774
46.3	消息队列控制操作	777
46.4	消息队列关联数据结构	778
46.5	消息队列的限制	780
46.6	显示系统中所有消息队列	781
46.7	使用消息队列实现客户端/服务器应用程序	783
46.8	使用消息队列实现文件服务器应用程序	784
46.9	System V 消息队列的缺点	790
46.10	总结	790
46.11	习题	791
第 47 章	System V 信号量	792
47.1	概述	793

47.2	创建或打开一个信号量集	795
47.3	信号量控制操作	796
47.4	信号量关联数据结构	798
47.5	信号量初始化	801
47.6	信号量操作	803
47.7	多个阻塞信号量操作的处理	809
47.8	信号量撤销值	810
47.9	实现一个二元信号量协议	811
47.10	信号量限制	814
47.11	System V 信号量的缺点	815
47.12	总结	816
47.13	习题	817
第 48 章 System V 共享内存		818
48.1	概述	818
48.2	创建或打开一个共享内存段	819
48.3	使用共享内存	820
48.4	示例：通过共享内存传输数据	821
48.5	共享内存存在虚拟内存中的位置	825
48.6	在共享内存中存储指针	828
48.7	共享内存控制操作	829
48.8	共享内存关联数据结构	830
48.9	共享内存的限制	832
48.10	总结	833
48.11	习题	833
第 49 章 内存映射		835
49.1	概述	835
49.2	创建一个映射：mmap()	837
49.3	解除映射区域：munmap()	840
49.4	文件映射	840
49.4.1	私有文件映射	841
49.4.2	共享文件映射	842
49.4.3	边界情况	845
49.4.4	内存保护和文件访问模式交互	846
49.5	同步映射区域：msync()	847
49.6	其他 mmap() 标记	848
49.7	匿名映射	849
49.8	重新映射一个映射区域：mremap()	852
49.9	MAP_NORESERVE 和过度利用交换空间	853
49.10	MAP_FIXED 标记	854

49.11	非线性映射: <code>remap_file_pages()</code>	855
49.12	总结.....	857
49.13	习题.....	858
第 50 章	虚拟内存操作	859
50.1	改变内存保护: <code>mprotect()</code>	859
50.2	内存锁: <code>mlock()</code> 和 <code>mlockatt()</code>	861
50.3	确定内存驻留性: <code>mincore()</code>	864
50.4	建议后续的内存使用模式: <code>madvise()</code>	866
50.5	小结.....	868
50.6	习题.....	868
第 51 章	POSIX IPC 介绍	869
51.1	API 概述.....	869
51.2	System V IPC 与 POSIX IPC 比较.....	872
51.3	总结.....	873
第 52 章	POSIX 消息队列	874
52.1	概述.....	874
52.2	打开、关闭和断开链接消息队列.....	875
52.3	描述符和消息队列之间的关系.....	877
52.4	消息队列特性.....	878
52.5	交换消息.....	882
52.5.1	发送消息.....	882
52.5.2	接收消息.....	883
52.5.3	在发送和接收消息时设置超时时间.....	885
52.6	消息通知.....	886
52.6.1	通过信号接收通知.....	887
52.6.2	通过线程接收通知.....	889
52.7	Linux 特有的特性.....	891
52.8	消息队列限制.....	892
52.9	POSIX 和 System V 消息队列比较.....	893
52.10	总结.....	894
52.11	习题.....	894
第 53 章	POSIX 信号量	895
53.1	概述.....	895
53.2	命名信号量.....	895
53.2.1	打开一个命名信号量.....	896
53.2.2	关闭一个信号量.....	898
53.2.3	删除一个命名信号量.....	898

53.3	信号量操作	899
53.3.1	等待一个信号量	899
53.3.2	发布一个信号量	901
53.3.3	获取信号量的当前值	901
53.4	未命名信号量	903
53.4.1	初始化一个未命名信号量	904
53.4.2	销毁一个未命名信号量	906
53.5	与其他同步技术比较	906
53.6	信号量的限制	907
53.7	总结	908
53.8	习题	908
第 54 章	POSIX 共享内存	909
54.1	概述	909
54.2	创建共享内存对象	910
54.3	使用共享内存对象	913
54.4	删除共享内存对象	915
54.5	共享内存 APIs 比较	915
54.6	总结	916
54.7	习题	917
第 55 章	文件加锁	918
55.1	概述	918
55.2	使用 flock()给文件加锁	920
55.2.1	锁继承与释放的语义	922
55.2.2	flock()的限制	923
55.3	使用 fcntl()给记录加锁	923
55.3.1	死锁	928
55.3.2	示例: 一个交互式加锁程序	928
55.3.3	示例: 一个加锁函数库	931
55.3.4	锁的限制和性能	933
55.3.5	锁继承和释放的语义	934
55.3.6	锁定饿死和排队加锁请求的优先级	935
55.4	强制加锁	935
55.5	/proc/locks 文件	938
55.6	仅运行一个程序的单个实例	939
55.7	老式加锁技术	941
55.8	总结	942
55.9	习题	943
第 56 章	SOCKET: 介绍	945

56.1	概述	945
56.2	创建一个 socket: <code>socket()</code>	948
56.3	将 socket 绑定到地址: <code>bind()</code>	948
56.4	通用 socket 地址结构: <code>struct sockaddr</code>	949
56.5	流 socket	950
56.5.1	监听接入连接: <code>listen()</code>	951
56.5.2	接受连接: <code>accept()</code>	952
56.5.3	连接到对等 socket: <code>connect()</code>	952
56.5.4	流 socket I/O	953
56.5.5	连接终止: <code>close()</code>	953
56.6	数据报 socket	953
56.6.1	交换数据报: <code>recvfrom</code> 和 <code>sendto()</code>	954
56.6.2	在数据报 socket 上使用 <code>connect()</code>	955
56.7	总结	956
第 57 章	SOCKET: UNIX DOMAIN	957
57.1	UNIX domain socket 地址: <code>struct sockaddr_un</code>	957
57.2	UNIX domain 中的流 socket	959
57.3	UNIX domain 中的数据报 socket	962
57.4	UNIX domain socket 权限	965
57.5	创建互联 socket 对: <code>socketpair()</code>	965
57.6	Linux 抽象 socket 名空间	966
57.7	总结	967
57.8	习题	967
第 58 章	SOCKET: TCP/IP 网络基础	968
58.1	因特网	968
58.2	联网协议和层	969
58.3	数据链路层	971
58.4	网络层: IP	971
58.5	IP 地址	973
58.6	传输层	975
58.6.1	端口号	975
58.6.2	用户数据报协议 (UDP)	976
58.6.3	传输控制协议 (TCP)	977
58.7	请求注解 (RFC)	979
58.8	总结	980
第 59 章	SOCKET: Internet DOMAIN	982
59.1	Internet domain socket	982

59.2	网络字节序	982
59.3	数据表示	984
59.4	Internet socket 地址	986
59.5	主机和服务转换函数概述	988
59.6	inet_pton()和 inet_ntop()函数	989
59.7	客户端-服务器示例 (数据报 socket)	990
59.8	域名系统 (DNS)	992
59.9	/etc/services 文件	994
59.10	独立于协议的主机和服务转换	995
59.10.1	getaddrinfo()函数	996
59.10.2	释放 addrinfo 列表: freeaddrinfo()	998
59.10.3	错误诊断: gai_strerror()	999
59.10.4	getnameinfo()函数	999
59.11	客户端-服务器示例 (流式 socket)	1000
59.12	Internet domain socket 库	1006
59.13	过时的主机和服务转换 API	1010
59.13.1	inet_aton()和 inet_ntoa()函数	1010
59.13.2	gethostbyname()和 gethostbyaddr()函数	1010
59.13.3	getserverbyname()和 getserverbyport()函数	1012
59.14	UNIX 与 Internet domain socket 比较	1013
59.15	更多信息	1014
59.16	总结	1014
59.17	习题	1015
第 60 章	SOCKET: 服务器设计	1016
60.1	迭代型和并发型服务器	1016
60.2	迭代型 UDP echo 服务器	1016
60.3	并发型 TCP echo 服务器	1019
60.4	并发型服务器的其他设计方案	1021
60.5	inetd (Internet 超级服务器) 守护进程	1023
60.6	总结	1027
60.7	练习	1027
第 61 章	SOCKET: 高级主题	1028
61.1	流式套接字上的部分读和部分写	1028
61.2	shutdown()系统调用	1030
61.3	专用于套接字的 I/O 系统调用: recv()和 send()	1033
61.4	sendfile()系统调用	1034
61.5	获取套接字地址	1036
61.6	深入探讨 TCP 协议	1039

61.6.1	TCP 报文的格式	1039
61.6.2	TCP 序列号和确认机制	1041
61.6.3	TCP 协议状态机以及状态迁移图	1041
61.6.4	TCP 连接的建立	1043
61.6.5	TCP 连接的终止	1044
61.6.6	在 TCP 套接字上调用 shutdown()	1045
61.6.7	TIME_WAIT 状态	1045
61.7	监视套接字: netstat	1047
61.8	使用 tcpdump 来监视 TCP 流量	1048
61.9	套接字选项	1049
61.10	SO_REUSEADDR 套接字选项	1050
61.11	在 accept()中继承标记和选项	1051
61.12	TCP vs UDP	1052
61.13	高级功能	1053
61.13.1	带外数据	1053
61.13.2	sendmsg()和 recvmsg()系统调用	1053
61.13.3	传递文件描述符	1054
61.13.4	接收发送端的凭据	1054
61.13.5	顺序数据包套接字	1055
61.13.6	SCTP 以及 DCCP 传输层协议	1055
61.14	总结	1056
61.15	练习	1056
第 62 章	终端	1058
62.1	整体概览	1059
62.2	获取和修改终端属性	1060
62.3	stty 命令	1062
62.4	终端特殊字符	1063
62.5	终端标志	1068
62.6	终端的 I/O 模式	1073
62.6.1	规范模式	1073
62.6.2	非规范模式	1074
62.6.3	加工模式、cbreak 模式以及原始模式	1075
62.7	终端线速 (比特率)	1081
62.8	终端的行控制	1082
62.9	终端窗口大小	1084
62.10	终端标识	1085
62.11	总结	1086
62.12	练习	1087

第 63 章 其他备选的 I/O 模型	1088
63.1 整体概览	1088
63.1.1 水平触发和边缘触发	1091
63.1.2 在备选的 I/O 模型中采用非阻塞 I/O	1092
63.2 I/O 多路复用	1092
63.2.1 select()系统调用	1092
63.2.2 poll()系统调用	1097
63.2.3 文件描述符何时就绪?	1101
63.2.4 比较 select()和 poll()	1103
63.2.5 select()和 poll()存在的问题	1105
63.3 信号驱动 I/O	1105
63.3.1 何时发送 “I/O 就绪” 信号	1109
63.3.2 优化信号驱动 I/O 的使用	1110
63.4 epoll 编程接口	1113
63.4.1 创建 epoll 实例: epoll_create()	1113
63.4.2 修改 epoll 的兴趣列表: epoll_ctl()	1114
63.4.3 事件等待: epoll_wait()	1115
63.4.4 深入探究 epoll 的语义	1120
63.4.5 epoll 同 I/O 多路复用的性能对比	1121
63.4.6 边缘触发通知	1122
63.5 在信号和文件描述符上等待	1124
63.5.1 pselect()系统调用	1125
63.5.2 self-pipe 技巧	1126
63.6 总结	1128
63.7 练习	1129
第 64 章 伪终端	1130
64.1 整体概览	1130
64.2 UNIX98 伪终端	1133
64.2.1 打开未使用的主设备: posix_openpt()	1134
64.2.2 修改从设备属主和权限: grantpt()	1135
64.2.3 解锁从设备: unlockpt()	1135
64.2.4 获取从设备名称: ptsname()	1136
64.3 打开主设备: ptyMasterOpen()	1136
64.4 将进程连接到伪终端: ptyFork()	1138
64.5 伪终端 I/O	1140
64.6 实现 script(1)程序	1142
64.7 终端属性和窗口大小	1146
64.8 BSD 风格的伪终端	1146

64.9 总结.....	1148
64.10 练习.....	1149
附录 A 跟踪系统调用.....	1151
附录 B 解析命令行选项.....	1153
附录 C 对 NULL 指针做转型.....	1159
附录 D 内核配置.....	1161
附录 E 更多信息源.....	1162
附录 F 部分习题解答.....	1167

第 34 章

进程组、会话和作业控制

进程组和会话在进程之间形成了一种两级层次关系：进程组是一组相关进程的集合，会话是一组相关进程组的集合。读者通过本章的学习就能弄清楚两个术语中“相关”的含义。

进程组和会话是为支持 shell 作业控制而定义的抽象概念，用户通过 shell 能够交互式地在前台或后台运行命令。术语“作业”通常与术语“进程组”作为同义词来看待。

本章将介绍进程组、会话和作业控制。

34.1 概述

进程组由一个或多个共享同一进程组标识符 (PGID) 的进程组成。进程组 ID 是一个数字，其类型与进程 ID 一样 (`pid_t`)。一个进程组拥有一个进程组首进程，该进程是创建该组的进程，其进程 ID 为该进程组的 ID，新进程会继承其父进程所属的进程组 ID。

进程组拥有一个生命周期，其开始时间为首进程创建组的时刻，结束时间为最后一个成员进程退出组的时刻。一个进程可能会因为终止而退出进程组，也可能会因为加入了另外一个进程组而退出进程组。进程组首进程无需是最后一个离开进程组的成员。

会话是一组进程组的集合。进程的会话成员关系是由其会话标识符 (SID) 确定的，会话标识符与进程组 ID 一样，是一个类型为 `pid_t` 的数字。会话首进程是创建该新会话的进程，其进程 ID 会成为会话 ID。新进程会继承其父进程的会话 ID。

一个会话中的所有进程共享单个控制终端。控制终端会在会话首进程首次打开一个终端设备时被建立。一个终端最多可能会成为一个会话的控制终端。

在任一时刻，会话中的其中一个进程组会成为终端的前台进程组，其他进程组会成为后台进程组。只有前台进程组中的进程才能从控制终端中读取输入。当用户在控制终端中输入其中一个信号生成终端字符之后，该信号会被发送到前台进程组中的所有成员。这些字符包括生成 SIGINT 的中断字符 (通常是 Control-C)、生成 SIGQUIT 的退出字符 (通常是 Control-\)、生成 SIGSTP 的挂起字符 (通常是 Control-Z)。

当到控制终端的连接建立起来 (即打开) 之后，会话首进程会成为该终端的控制进程。成为控制进程的主要标志是当断开与终端之间的连接时内核会向该进程发送一个 SIGHUP 信号。

通过检查 Linux 特有的 `/proc/PID/stat` 文件，就能确定任意进程的进程组 ID 和会话 ID。此外，还能确定进程的控制终端的设备 ID（一个十进制数字，包含主 ID 和辅 ID）和控制该终端的控制进程的进程 ID。更多细节信息请参考 `proc(5)` 手册。

会话和进程组的主要用途是用于 shell 作业控制。读者通过一个具体的例子就能够弄清楚这些概念了。如对于交互式登录来讲，控制终端是用户登录的途径。登录 shell 是会话首进程和终端的控制进程，也是其自身进程组的唯一成员。从 shell 中发出的每个命令或通过管道连接的一组命令都会导致一个或多个进程的创建，并且 shell 会把所有这些进程都放在一个新进程组中。（这些进程在一开始是其进程组中的唯一成员，它们创建的所有子进程会成为该组中的成员。）当命令或以管道连接的一组命令以 `&` 符号结束时会在后台进程组中运行这些命令，否则就会在前台进程组中运行这些命令。在登录会话中创建的所有进程都会成为该会话的一部分。

在窗口环境中，控制终端是一个伪终端。每个终端窗口都有一个独立的会话，窗口的启动 shell 是会话首进程和终端的控制进程。

在除任务控制之外的其他场景中也有可能用到进程组，因为进程组具备两个有用的属性：在特定的进程组中父进程能够等待任意子进程（参见 26.1.2 节）和信号能够被发送给进程组中的所有成员（参见 20.5 节）。

图 34.1 给出了执行下面的命令之后各个进程之间的进程组和会话关系。

```

$ echo $$          Display the PID of the shell
400
$ find / 2> /dev/null | wc -l &    Creates 2 processes in background group
[1] 659
$ sort < longlist | uniq -c        Creates 2 processes in foreground group

```

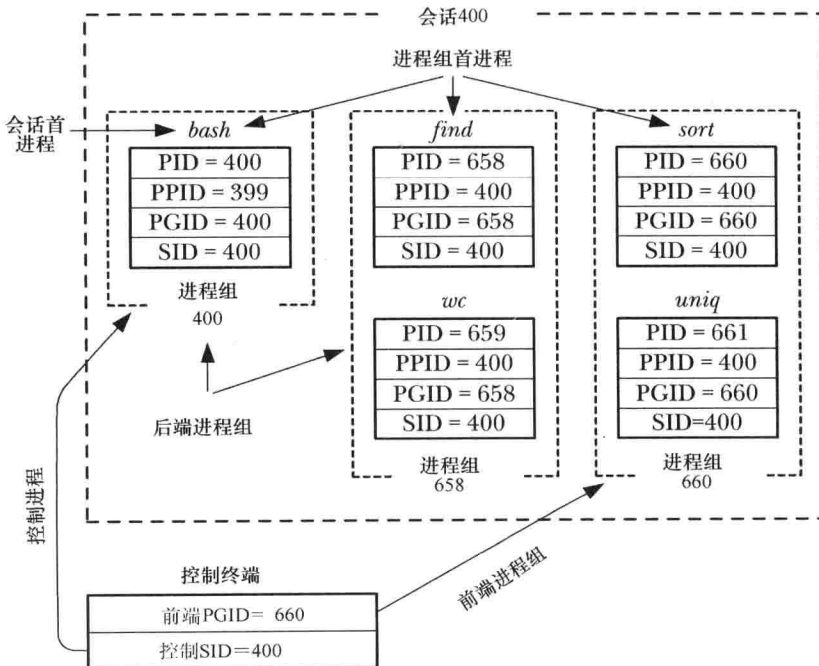


图 34.1 进程组、会话和控制终端之间的关系