么结果可能就不可预知，而且经常是致命的。

我们会在第 12 章详细讲述并发编程。这里我们的目标是给你一些保守的编写处理程序的原则，使得这些处理程序能安全地并发运行。如果你忽视这些原则，就可能有引入细微的并发错误的风险。如果有这些错误，程序可能在绝大部分时候都能正确工作。然而当它出错的时候，就会错得不可预测和不可重复，这样是很难调试的。一定要防患于未然！

- G0. 处理程序要尽可能简单。避免麻烦的最好方法是保持处理程序尽可能小和简单。例如，处理程序可能只是简单地设置全局标志并立即返回；所有与接收信号相关的处理都由主程序执行，它周期性地检查(并重置)这个标志。
- G1. 在处理程序中只调用异步信号安全的函数。所谓异步信号安全的函数(或简称安全的函数)能够被信号处理程序安全地调用，原因有二：要么它是可重入的(例如只访问局部变量，见 12.7.2 节)，要么它不能被信号处理程序中断。图 8-33 列出了 Linux 保证安全的系统级函数。注意，许多常见的函数(例如 printf、sprintf、malloc 和 exit)都不在此列。

_Exit	fexecve	poll	sigqueue
_exit	fork	posix_trace_event	sigset
abort	fstat	pselect	sigsuspend
accept	fstatat	raise	sleep
access	fsync	read	sockatmark
aio_error	ftruncate	readlink	socket
aio_return	futimens	readlinkat	socketpair
aio_suspend	getegid	recv	stat
alarm	geteuid	recvfrom	symlink
bind	getgid	recvmsg	symlinkat
cfgetispeed	getgroups	rename	tcdrain
cfgetospeed	getpeername	renameat	tcflow
cfsetispeed	getpgrp	rmdir	tcflush
cfsetospeed	getpid	select	tcgetattr
chdir	getppid	sem_post	tcgetpgrp
chmod	getsockname	send	tcsendbreak
chown	getsockopt	sendmsg	tcsetattr
clock_gettime	getuid	sendto	tcsetpgrp
close	kill	setgid	time
connect	link	setpgid	timer_getoverrun
creat	linkat	setsid	timer_gettime
dup	listen	setsockopt	timer_settime
dup2	lseek	setuid	times
execl	lstat	shutdown	umask
execle	mkdir	sigaction	uname
execv	mkdirat	sigaddset	unlink
execve	mkfifo	sigdelset	unlinkat
faccessat	mkfifoat	sigemptyset	utime
fchmod	mknod	sigfillset	utimensat
fchmodat	mknodat	sigismember	utimes
fchown	open	signal	wait
fchownat	openat	sigpause	waitpid
fcntl	pause	sigpending	write
fdatasync	pipe	sigprocmask	

图 8-33　异步信号安全的函数(来源：man 7 signal。数据来自 Linux Foundation)