信号

- 信号是 Linux 进程间通信的最古老的方式之一，是事件发生时对进程的通知机制，有时也称之为软件中断，它是在软件层次上对中断机制的一种模拟，是一种异步通信的方式。信号可以导致一个正在运行的进程被另一个正在运行的异步进程中断，转而处理某一个突发事件。
- 发往进程的诸多信号，通常都是源于内核。引发内核为进程产生信号的各类事件如下：
  - 对于前台进程，用户可以通过输入特殊的终端字符来给它发送信号。比如输入Ctrl+C通常会给进程发送一个中断信号。
  - 硬件发生异常，即硬件检测到一个错误条件并通知内核，随即再由内核发送相应信号给相关进程。比如执行一条异常的机器语言指令，诸如被 0 除，或者引用了无法访问的内存区域。
  - 系统状态变化，比如 alarm 定时器到期将引起 SIGALRM 信号，进程执行的 CPU 时间超限，或者该进程的某个子进程退出。
  - 运行 kill 命令或调用 kill 函数。kill -9(-9就是一个信号)

- 使用信号的两个主要目的是：
  - 让进程知道已经发生了一个特定的事情。
  - 强迫进程执行它自己代码中的信号处理程序。
- 信号的特点：
  - 简单
  - 不能携带大量信息
  - 满足某个特定条件才发送
  - 优先级比较高
- 查看系统定义的信号列表：`kill -l`
- 前 31 个信号为常规信号，其余为实时信号。

02 / Linux 信号一览表

编号	信号名称	对应事件	默认动作
1	SIGHUP	用户退出shell时，由该shell启动的所有进程将收到这个信号	终止进程
2	杀不死非前台的进程 **SIGINT**	当用户按下了<Ctrl+C>组合键时，用户终端向正在运行中的由该终端启动的程序发出此信号	终止进程
3	**SIGQUIT**	用户按下<Ctrl+\>组合键时产生该信号，用户终端向正在运行中的由该终端启动的程序发出些信号	终止进程
4	SIGILL	CPU检测到某进程执行了非法指令	终止进程并产生core文件
5	SIGTRAP	该信号由断点指令或其他 trap指令产生	终止进程并产生core文件
6	SIGABRT	调用abort函数时产生该信号	终止进程并产生core文件
7	SIGBUS	非法访问内存地址，包括内存对齐出错	终止进程并产生core文件
8	SIGFPE	在发生致命的运算错误时发出。不仅包括浮点运算错误，还包括溢出及除数为0等所有的算法错误	终止进程并产生core文件

02 / Linux 信号一览表

编号	信号名称	对应事件	默认动作
9	**SIGKILL**	无条件终止进程。该信号不能被忽略，处理和阻塞	终止进程，可以杀死任何进程
10	SIGUSE1	用户定义的信号。即程序员可以在程序中定义并使用该信号	终止进程
11	**SIGSEGV**	指示进程进行了无效内存访问（段错误）	终止进程并产生core文件
12	SIGUSR2	另外一个用户自定义信号，程序员可以在程序中定义并使用该信号	终止进程
13	**SIGPIPE**	Broken pipe向一个没有读端的管道写数据	终止进程
14	SIGALRM	定时器超时，超时的时间 由系统调用alarm设置	终止进程
15	SIGTERM	程序结束信号，与SIGKILL不同的是，该信号可以被阻塞和终止。通常用来要示程序正常退出。执行shell命令Kill时，缺省产生这个信号	终止进程
16	SIGSTKFLT	Linux早期版本出现的信号，现仍保留向后兼容	终止进程

编号	信号名称	对应事件	默认动作
17	**SIGCHLD**	子进程结束时，父进程会收到这个信号	忽略这个信号
18	**SIGCONT**	如果进程已停止，则使其继续运行	继续/忽略
19	**SIGSTOP**	停止进程的执行。信号不能被忽略，处理和阻塞	为终止进程
20	SIGTSTP	停止终端交互进程的运行。按下<ctrl+z>组合键时发出这个信号	暂停进程
21	SIGTTIN	后台进程读终端控制台	暂停进程
22	SIGTTOU	该信号类似于SIGTTIN，在后台进程要向终端输出数据时发生	暂停进程
23	SIGURG	套接字上有紧急数据时，向当前正在运行的进程发出些信号，报告有紧急数据到达。如网络带外数据到达	忽略该信号
24	SIGXCPU	进程执行时间超过了分配给该进程的CPU时间，系统产生该信号并发送给该进程	终止进程

编号	信号名称	对应事件	默认动作
25	SIGXFSZ	超过文件的最大长度设置	终止进程
26	SIGVTALRM	虚拟时钟超时时产生该信号。类似于SIGALRM，但是该信号只计算该进程占用CPU的使用时间	终止进程
27	SGIPROF	类似于SIGVTALRM，它不公包括该进程占用CPU时间还包括执行系统调用时间	终止进程
28	SIGWINCH	窗口变化大小时发出	忽略该信号
29	SIGIO	此信号向进程指示发出了一个异步IO事件	忽略该信号
30	SIGPWR	关机	终止进程
31	SIGSYS	无效的系统调用	终止进程并产生core文件
34 ~ 64	SIGRTMIN ~ SIGRTMAX	LINUX的实时信号，它们没有固定的含义（可以由用户自定义）	终止进程

- 查看信号的详细信息：man 7 signal
- 信号的 5 中默认处理动作
  - Term（terminate）　终止进程
  - Ign（ignore）　当前进程忽略掉这个信号
  - Core　终止进程，并生成一个Core文件 为了对程序的错误进行调试
  - Stop　暂停当前进程
  - Cont（continue）　继续执行当前被暂停的进程
- 信号的几种状态：产生、未决、递达
- SIGKILL 和 SIGSTOP 信号不能被捕捉、阻塞或者忽略，只能执行默认动作。

- `int kill(pid_t pid, int sig);`
- `int raise(int sig);`
- `void abort(void);`
- `unsigned int alarm(unsigned int seconds);`
- `int setitimer(int which, const struct itimerval *new_val, struct itimerval *old_value);`

- `sighandler_t signal(int signum, sighandler_t handler);`
- `int sigaction(int signum, const struct sigaction *act, struct sigaction *oldact);`

- 许多信号相关的系统调用都需要能表示一组不同的信号，多个信号可使用一个称之为信号集的数据结构来表示，其系统数据类型为 sigset_t。
- 在 PCB 中有两个非常重要的信号集。一个称之为“阻塞信号集”，另一个称之为“未决信号集”。这两个信号集都是内核使用位图机制来实现的。但操作系统不允许我们直接对这两个信号集进行位操作。而需自定义另外一个集合，借助信号集操作函数来对 PCB 中的这两个信号集进行修改。
- 信号的“未决”是一种状态，指的是从信号的产生到信号被处理前的这一段时间。
- 信号的“阻塞”是一个开关动作，指的是阻止信号被处理，但不是阻止信号产生。
- 信号的阻塞就是让系统暂时保留信号留待以后发送。由于另外有办法让系统忽略信号，所以一般情况下信号的阻塞只是暂时的，只是为了防止信号打断敏感的操作。

07 / 阻塞信号集和未决信号集

在PCB中

未决信号集

1	0
2-SIGINT	0
3-SIGQUIT	0
4	0
5	0
6-SIGABRT	0
7	0
8	0
9-SIGKILL	0
10	0
11	0
...	0
64	0

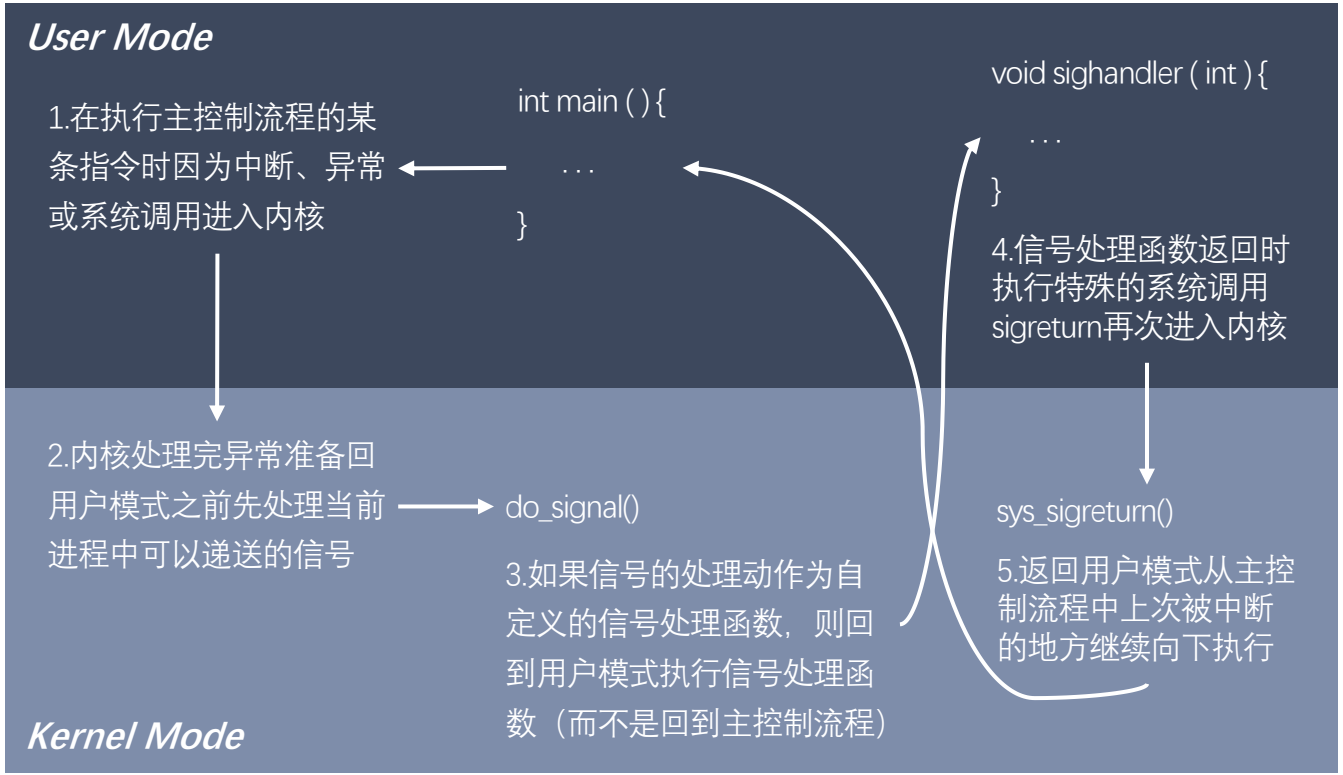
阻塞信号集

1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0
...	0
64	0

- `int sigemptyset(sigset_t *set);`
- `int sigfillset(sigset_t *set);`
- `int sigaddset(sigset_t *set, int signum);`
- `int sigdelset(sigset_t *set, int signum);`
- `int sigismember(const sigset_t *set, int signum);`
- `int sigprocmask(int how, const sigset_t *set, sigset_t *oldset);`
- `int sigpending(sigset_t *set);`

用户区

内核区



10 / SIGCHLD信号 解决僵尸进程

- SIGCHLD信号产生的条件
  - 最常出现 子进程终止时 传递给父进程
  - 子进程接收到 SIGSTOP 信号停止时 暂停进程
  - 子进程处在停止态，接受到SIGCONT后唤醒时
- 以上三种条件都会给父进程发送 SIGCHLD 信号，父进程默认会忽略该信号

牛客大学
-专业求职辅导-

THANKS

关注【牛客大学】公众号
回复“牛客大学”获取更多求职资料