信号机制

概念：信号是在软件层次上对中断机制的一种模拟，是一种异步通信方式

所有信号的产生及处理全部都是由内核完成的

信号的产生：

1 按键产生

2 系统调用函数产生（比如raise， kill）

3 硬件异常

4 命令行产生 （kill）

5 软件条件（比如被0除，访问非法内存等）

信号处理方式：

1 缺省方式

2 忽略信号

3 捕捉信号

常用信号：

|  |  |  |
| --- | --- | --- |
| 信号名 | 含义 | 默认操作 |
| SIGHUP | 该信号在用户终端关闭时产生，通常是发给和该  终端关联的会话内的所有进程 | 终止 |
| SIGINT | 该信号在用户键入INTR字符(Ctrl-C)时产生，内  核发送此信号送到当前终端的所有前台进程 | 终止 |
| SIGQUIT | 该信号和SIGINT类似，但由QUIT字符(通常是  Ctrl-\)来产生 | 终止 |
| SIGILL | 该信号在一个进程企图执行一条非法指令时产生 | 终止 |
| SIGSEV | 该信号在非法访问内存时产生，如野指针、缓  冲区溢出 | 终止 |
| SIGPIPE | 当进程往一个没有读端的管道中写入时产生，代  表“管道断裂” | 终止 |

|  |  |  |
| --- | --- | --- |
| 信号名 | 含义 | 默认操作 |
| SIGKILL | 该信号用来结束进程，并且不能被捕捉和忽略 | 终止 |
| SIGSTOP | 该信号用于暂停进程，并且不能被捕捉和忽略 | 暂停进程 |
| SIGTSTP | 该信号用于暂停进程，用户可键入SUSP字符(  通常是Ctrl-Z)发出这个信号 | 暂停进程 |
| SIGCONT | 该信号让进程进入运行态 | 继续运行 |
| SIGALRM | 该信号用于通知进程定时器时间已到 | 终止 |
| SIGUSR1/2 | 该信号保留给用户程序使用 | 终止 |
| SIGCHLD | 是子进程状态改变发给父进程的。 | 忽略 |

信号命令：

kill [-signal] pid

killall [-u user | prog]

信号的函数：

int kill（pid\_t pid, int signum）

功能：发送信号

参数：

pid: > 0:发送信号给指定进程

= 0：发送信号给跟调用kill函数的那个进程处于同一进程组的进程。

< -1: 取绝对值，发送信号给该绝对值所对应的进程组的所有组员。

= -1：发送信号给，有权限发送的所有进程。

signum：待发送的信号

int raise(int sig);

给自己发信号，等价于kill(getpid(), signo);

定时器函数

unsigned int alarm(unsigned int seconds);

功能：定时发送SIGALRM给当前进程

参数： seconds：定时秒数

返回值：上次定时剩余时间。

ualarm （循环发送）

useconds\_t ualarm(useconds\_t usecs, useconds\_t interval);

以useconds为单位，第一个参数为第一次产生时间，第二个参数为间隔产生

int setitimer(int which, const struct itimerval \*new\_value, struct itimerval \*old\_value);

功能：定时的发送alarm信号

参数：

which：

ITIMER\_REAL：以逝去时间递减。发送SIGALRM信号

ITIMER\_VIRTUAL: 计算进程（用户模式）执行的时间。 发送SIGVTALRM信号

ITIMER\_PROF: 进程在用户模式（即程序执行时）和核心模式（即进程调度用时）均计算时间。 发送SIGPROF信号

new\_value： 负责设定 timout 时间

old\_value： 存放旧的timeout值，一般指定为NULL

struct itimerval {

struct timeval it\_interval; // 闹钟触发周期

struct timeval it\_value; // 闹钟触发时间

};

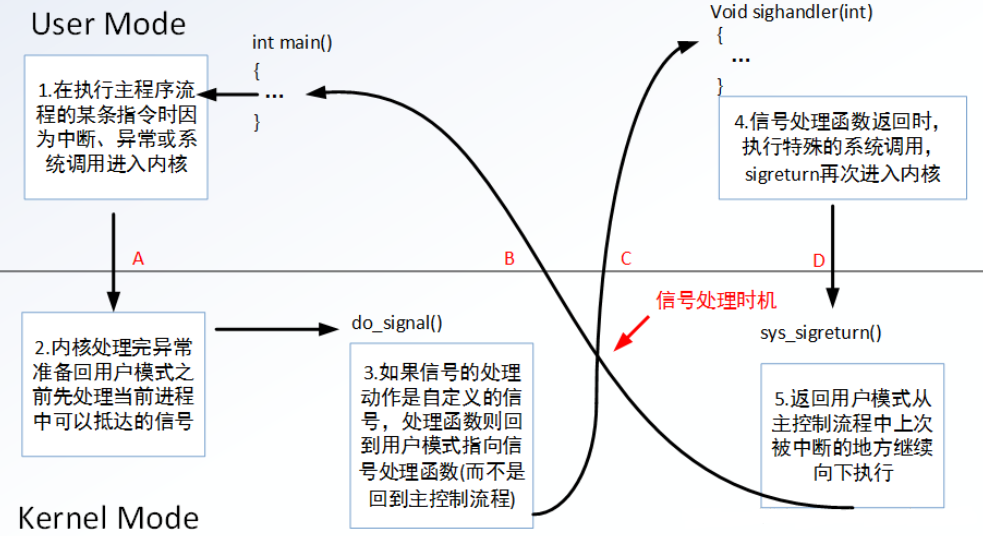
struct timeval {

time\_t tv\_sec; /\* seconds \*/

suseconds\_t tv\_usec; /\* microseconds \*/

};

信号的捕捉



**信号捕捉过程：**

1. 定义新的信号的执行函数handle。
2. 使用signal/sigaction 函数，把自定义的handle和指定的信号相关联。

**signal函数：**

typedef void (\*sighandler\_t)(int);

sighandler\_t signal(int signum, sighandler\_t handler);

功能：捕捉信号执行自定义函数

返回值：成功时返回原先的信号处理函数，失败时返回SIG\_ERR

参数：

signo 要设置的信号类型

handler 指定的信号处理函数: SIG\_DFL代表缺省方式; SIG\_IGN 代表忽略信号;

系统建议使用sigaction函数，因为signal在不同类unix系统的行为不完全一样。

**sigaction函数：**

int sigaction(int signum, const struct sigaction \*act,struct sigaction \*oldact);

struct sigaction {

void (\*sa\_handler)(int);

void (\*sa\_sigaction)(int, siginfo\_t \*, void \*);

sigset\_t sa\_mask;

int sa\_flags;

void (\*sa\_restorer)(void);

}

参数：

signum：处理的信号

act,oldact: 处理信号的新行为和旧的行为，是一个sigaction结构体。

sigaction结构体成员定义如下：

sa\_handler： 是一个函数指针，其含义与 signal 函数中的信号处理函数类似

sa\_sigaction： 另一个信号处理函数，它有三个参数，可以获得关于信号的更详细的信息。

sa\_flags参考值如下：

SA\_SIGINFO：使用 sa\_sigaction 成员而不是 sa\_handler 作为信号处理函数

SA\_RESTART：使被信号打断的系统调用自动重新发起。

SA\_RESETHAND：信号处理之后重新设置为默认的处理方式。

SA\_NODEFER：使对信号的屏蔽无效，即在信号处理函数执行期间仍能发出这个信号。

re\_restorer：是一个已经废弃的数据域

**定时器的实现**

**使用SIGCHLD信号实现回收子进程**

SIGCHLD的产生条件

1子进程终止时

2子进程接收到SIGSTOP信号停止时

3子进程处在停止态，接受到SIGCONT后唤醒时