# 信号

## 信号的概念

标准信号

实时信号

kill -l 命令查看系统支持的信号类型和信号编号

信号是软件中断。

## signal（）

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

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

int signum：信号类型

sighandler\_t handler：要执行的动作 SIG\_IGN //忽略信号SIG\_DFL ，自定义函数

信号会打断阻塞的系统调用（ag: sleep open）

1. 信号的不可靠

标准信号会丢失并不是信号不可靠的原因而是信号的行为不可靠（执行现场由内核决定信号处理函数的执行现场不是程序员布置的，而是内核布置的，因为程序中不会有调用信号处理函数的地方。同一个信号处理函数的执行现场会被布置在同一个地方，所以当一次信号处理函数未执行完成时再次触发了相同的信号，信号处理函数发生了第二次调用，则第一次调用的执行现场会被覆盖）

标准信号的响应没有严格的顺序

1. 可重入函数

所有的系统调用和一部分库函数都是可重入的

man手册中后缀为“\_r”的函数表示可重入版本。

1. 信号的响应过程

内核为每个进程维护了两个（32位）位图，MASK和pending，MASK是屏蔽信号位图（响应中断置1，不响应置0），pending是信号是否发生位图（发生了中断为1否则为0）。在每个时间片用完，发生任务调度时，内核会根据这两个位图判断要运行进程是否收到了信号。如果收到了信号，首先更改这两个位图，屏蔽同类型的信号，然后更改PC指针到信号响应函数开始运行。

使用信号和alarm函数完成时间控制的程序，时间误差最小可以控制在10ms。

MASK置0？？

标准信号的丢失是因为，在一个信号未得到响应前，无论重复发多少次相同的信号，都只能响应一次，因为记录信号发生的位图只有1位，多余的信号只是在重复对位图进行设置，发生一次penging置1，发生100次penging还是1。（实时信号不会丢失）

信号的响应由一定的延时，因为需要等待调度到再次运行时，才能判断是否受到信号进行响应。

1. 常用函数

//发送信号给进程

int kill(pid\_t pid, int sig);

//给自己（进程或线程）发信号

int raise(int sig);

//计时seconds后给自己发信号 这个信号的默认功能会杀死进程，可采用signal()函 //数修改信号的功能。

unsigned int alarm(unsigned int seconds);

//当多个alarm存在时只有最后一个alarm生效。也就是alarm只能设置一个计时值。

//单次单值计时

//堵塞的等待一个信号

int pause(void);

int getitimer(int which, struct itimerval \*curr\_value);

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

struct itimerval \*old\_value);

abort

system

sleep

1. 信号集
2. 信号屏蔽字/pending集 的处理
3. sigsuspend sigaction setitimer
4. 实时信号

# 多线程