alarm()用来设置信号SIGALRM在经过参数seconds指定的秒数后传送给目前的进程。如果参数seconds 为0,则之前设置的闹钟会被取消,并将剩下的时间返回。信号正常发送时返回值为0。若新的报警设置在前一次设定尚未结束时,发送,则返回前一次设置所剩时间。返回之前闹钟的剩余秒数,如果之前未设闹钟则返回0。

alarm()执行后,进程将继续执行,在后期(alarm以后)的执行过程中将会在seconds秒后收到信号SIGALRM并执行其处理函数。

使用alarm函数可以设置一个计时器,在将来某个指定的时间该计时器会超时。当计时器超时时,产生SIGALARM信号。如果不捕捉或不忽略此信号,则其默认动作是终止调用该alarm函数的进程。

每个进程只能有一个闹钟时钟,如果在调用alarm时,已经为该进程设置过闹钟时钟,而且它还没有超时,则将该闹钟时钟的余留值作为本次alarm函数调用的返回值。以前登记的闹钟时钟将被新值代替。

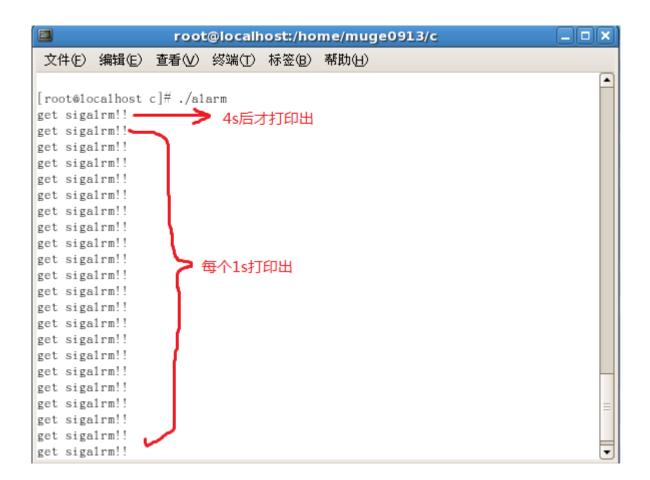
如果有以前为进程登记的尚未超过的闹钟时钟,而且本次调用的 seconds为0,则取消以前的闹钟时钟,其余留值仍作为alarm的返回 值。

(以上这几段话摘自《unix环境高级编程》)

[cpp] view plaincopyprint?

```
1. #include <stdio.h>
2. #include <signal.h>
3. #include <time.h>
4.

5. void func();
6. void main()
7. {
8. signal(SIGALRM, &func);
9. alarm(4);
```



signal

表头文件 #include(signal.h)

功能:

设置某一信号的对应动作

函数原型:

```
void (*signal(int signum, void(* handler)(int)))(int);
```

或者: typedef void(*sig_t) (int); sig_t signal(int signum, sig_t handler);

参数说明:

第一个参数signum指明了所要处理的信号类型,它可以取除了SIGKILL和SIGSTOP外的任何一种信号。

第二个参数handler描述了与信号关联的动作,它可以取以下三种值: