

```

code/ecf/sigint.c

1  #include "csapp.h"
2
3  void sigint_handler(int sig) /* SIGINT handler */
4  {
5      printf("Caught SIGINT!\n");
6      exit(0);
7  }
8
9  int main()
10 {
11     /* Install the SIGINT handler */
12     if (signal(SIGINT, sigint_handler) == SIG_ERR)
13         unix_error("signal error");
14
15     pause(); /* Wait for the receipt of a signal */
16
17     return 0;
18 }
code/ecf/sigint.c

```

图 8-30 一个用信号处理程序捕获 SIGINT 信号的程序

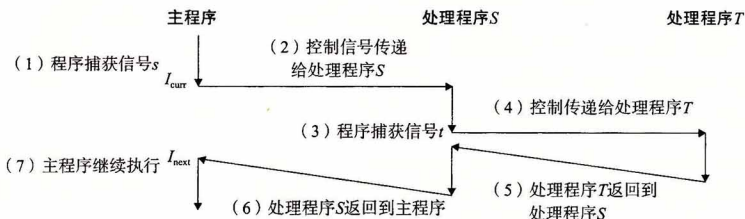



图 8-31 信号处理程序可以被其他信号处理程序中中断

 **练习题 8.7** 编写一个叫做 `snooze` 的程序，它有一个命令行参数，用这个参数调用练习题 8.5 中的 `snooze` 函数，然后终止。编写程序，使得用户可以通过在键盘上输入 `Ctrl+C` 中断 `snooze` 函数。比如：

```

linux> ./snooze 5
CTRL+C                               User hits Ctrl+C after 3 seconds
Slept for 3 of 5 secs.
linux>

```

8.5.4 阻塞和解除阻塞信号

Linux 提供阻塞信号的隐式和显式的机制：

隐式阻塞机制。内核默认阻塞任何当前处理程序正在处理信号类型的待处理的信号。例如，图 8-31 中，假设程序捕获了信号 s ，当前正在运行处理程序 S 。如果发送给该进程另一个信号 s ，那么直到处理程序 S 返回， s 会变成待处理而没有接收。

显式阻塞机制。应用程序可以使用 `sigprocmask` 函数和它的辅助函数，明确地阻塞和解除阻塞选定的信号。