4. 用 kill 函数发送信号

进程通过调用 kill 函数发送信号给其他进程(包括它们自己)。

```
#include <sys/types.h>
#include <signal.h>

int kill(pid_t pid, int sig);
```

返回：若成功则为 0，若错误则为 −1。

如果 pid 大于零，那么 kill 函数发送信号号码 sig 给进程 pid。如果 pid 等于零，那么 kill 发送信号 sig 给调用进程所在进程组中的每个进程，包括调用进程自己。如果 pid 小于零，kill 发送信号 sig 给进程组 |pid|(pid 的绝对值)中的每个进程。图 8-29 展示了一个示例，父进程用 kill 函数发送 SIGKILL 信号给它的子进程。

code/ecf/kill.c

```
1   #include "csapp.h"
2
3   int main()
4   {
5       pid_t pid;
6
7       /* Child sleeps until SIGKILL signal received, then dies */
8       if ((pid = Fork()) == 0) {
9           Pause();  /* Wait for a signal to arrive */
10          printf("control should never reach here!\n");
11          exit(0);
12      }
13
14      /* Parent sends a SIGKILL signal to a child */
15      Kill(pid, SIGKILL);
16      exit(0);
17  }
```

code/ecf/kill.c

图 8-29　使用 kill 函数发送信号给子进程

5. 用 alarm 函数发送信号

进程可以通过调用 alarm 函数向它自己发送 SIGALRM 信号。

```
#include <unistd.h>

unsigned int alarm(unsigned int secs);
```

返回：前一次闹钟剩余的秒数，若以前没有设定闹钟，则为 0。

alarm 函数安排内核在 secs 秒后发送一个 SIGALRM 信号给调用进程。如果 secs 是零，那么不会调度安排新的闹钟(alarm)。在任何情况下，对 alarm 的调用都将取消任何待处理的(pending)闹钟，并且返回任何待处理的闹钟在被发送前还剩下的秒数(如果这次对 alarm 的调用没有取消它的话)；如果没有任何待处理的闹钟，就返回零。