signal()函数

参数一：我们要进行处理到函数 kill -l 可查看64个

参数二：处理方式（SIG_ING忽略、SIG_DFL默认、pvoid不带任何参数的函数指针）

```c
#include<stdio.h>
#include<unistd.h>
#include<signal.h>
#include<memory.h>
#include<errno.h>
#include<stdlib.h>

void ouch(int sig)
{
    printf("I got signal %d\n", sig);
    return;

}


int main(int argc,char *argv[])
{
    signal(SIGINT,SIG_ING);
    signal(SIGINT,ouch);
    while(1)
    {
        fprintf(stdout,"continue!\n");
        sleep(1000);
    }
    return 0;
}

```

sigaction()

```c
struct sigaction {
               void     (*sa_handler)(int);
               void     (*sa_sigaction)(int, siginfo_t *, void *);
```

```
    sigset_t   sa_mask;
    int        sa_flags;
    void     (*sa_restorer)(void);
};
```

1.阻塞，sigaction函数有阻塞功能，比如SIGINT信号来了，进入信号处理函数，默认情况下，在信号处理函数未完成之前，如果又来了一个SIGINT信号，其将被阻塞，只有信号处理函数处理完毕，才会对后来的SIGINT再进行处理，同时后续无论来多少个SIGINT，仅处理一个SIGINT，sigaction对后续SIGINT进行排队合并处理

2.sa_mask，信号屏蔽集，通过函数sigemptyset/sigaddset等来清空和增加需要屏蔽的信号，上面代码中，对信号SIGINT处理时，如果来信号SIGQUIT，其将被屏蔽，如果在处理SIGQUIT，来了SIGINT，首先处理SIGINT，接着处理SIGQUIT

3.sa_flags如果取值为0，表示默认行为

SA_RESETHAND:当调用信号处理函数时，将信号的处理函数重置为缺省值

SA_RESTART:如果信号中断了进程的某个系统调用，则系统自动启动该系统调用

SA_NODEFER:一般情况下，当信号处理函数运行时，内核将阻塞该给定信号，但是如果设置了SA_NODEFER标记，那么在该信号处理函数运行时，内核将不会阻塞该信号

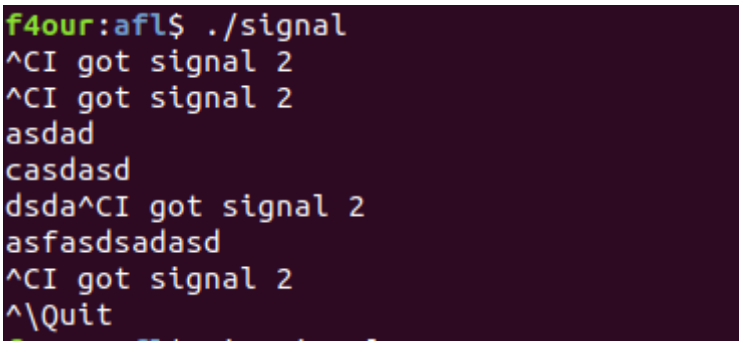
```c
#include<stdio.h>
#include<unistd.h>
#include<signal.h>
#include<memory.h>
#include<errno.h>
#include<stdlib.h>

#define BUF_SIZE 2048

void ouch(int sig)
{
    printf("I got signal %d\n", sig);
    return;
}

int main(int argc,char *argv[])
{
    char buffer[BUF_SIZE] = {0};
```

```
19
20      struct sigaction sa;
21      sa.sa_handler = NULL;
22      sa.sa_flags = SA_RESTART;
23      sa.sa_sigaction = NULL;
24
25      sigemptyset(&sa.sa_mask);
26
27      sa.sa_handler = ouch;
28      sigaction(SIGINT,&sa,NULL);
29
30      fgets(buffer,BUF_SIZE,stdin);
31      fprintf(stdout,"%s\n",buffer);
32
33      return 0;
34  }
35
```



当处理SIG_INT并不会中断read函数系统调用