Using raise()

```
#include <signal.h>
#include <stdio.h>

void signal_catchfunc(int);

int main()
{
   int ret;

  ret = signal(SIGINT, signal_catchfunc);

  if( ret == SIG_ERR)
  {
    printf("Error: unable to set signal handler.\n");
    exit(0);
  }
  printf("Going to raise a signal\n");
  ret = raise(SIGINT);
  if( ret !=0 )
  {
    printf("Error: unable to raise SIGINT signal.\n");
    exit(0);
  }

  printf("Exiting...\n");
  return(0);
}

void signal_catchfunc(int signal)
{
  printf("!! signal caught !!\n");
}
```

Using Signal

```
/* set CTRL-C and CTRL-\ to be trapped by a function called signal_catcher */
#include <stdio.h>
#include <signal.h>
#include <unistd.h>
#include <stdlib.h>

int main (void)
{
     int i;
     void signal_catcher(int);

     if (signal(SIGINT,signal_catcher)==SIG_ERR)
     {
         perror("Sigset cannot set SIGINT");
         exit(SIGINT);
     }
     if (signal(SIGQUIT, signal_catcher)==SIG_ERR)
     {
         perror("Sigset can not set SIGQUIT");
         exit(SIGQUIT);
     }
     for(i=0;; ++i)
     {
         printf("%i\n",i);
         sleep(1);
     }
}
void signal_catcher(int the_sig)
{
     //The following line is commented out, but may be necessary in
     //some implementations. Otherwise, the signal may return to its
     //default action after one occurance of the signal is handled
     //signal(the_sig, signal_catcher); //reset
     printf("\nSignal %d received. \n", the_sig);
     if (the_sig == SIGQUIT)
         exit(1);
}
```

Using sigprocmask()

```
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc,  char *argv[]) {
int i;
sigset_t intmask;

if ((sigemptyset(&intmask) == -1) || (sigaddset(&intmask, SIGINT) == -1)){
   perror("Failed to initialize the signal mask");
   return 1;
}

for ( ; ; ) {
   printf("Entering BLOCK state\n");
   if (sigprocmask(SIG_BLOCK, &intmask, NULL) == -1)
      break;
   fprintf(stderr, "SIGINT signal blocked\n");
   sleep(3);

   printf("Leaving Blocking State & Entering UNBLOCK state\n");
   if (sigprocmask(SIG_UNBLOCK, &intmask, NULL) == -1)
```

```
      break;
   fprintf(stderr, "SIGINT signal unblocked\n");
   sleep(2);
}
perror("Failed to change signal mask");
return 1;
}
```

Using Signal sets

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>

void print( sigset_t set, int signo )
  {

    printf( "Set %8.8lx. Signal %d is ", set, signo );
    if( sigismember( &set, signo ) )
      printf( "a member.\n" );
    else
      printf( "not a member.\n" );
  }

int main( void )
  {
    sigset_t set;

        printf("Calling sigemptyset\n");
    sigemptyset( &set );
    print( set, SIGINT );

        printf("Calling sigfillset\n");
    sigfillset( &set );
    print( set, SIGINT );
//
        printf("Calling sigdelset\n");
    sigdelset( &set, SIGINT );
    print( set, SIGINT );

        printf("Calling sigaddset\n");
    sigaddset( &set, SIGINT );
    print( set, SIGINT );
   return EXIT_SUCCESS;
  }
```