raise() — Raise signal

Last Updated: 2021-06-25

Standards

Standards / Extensions	C or C++	Dependencies
ISO C XPG4 XPG4.2 C99 Single UNIX Specification, Version 3	both	

Format

```
#include <signal.h>

int raise(int sig);
```

General description

Sends the signal *sig* to the program that issued raise(). See Table 1 for the list of signals supported.

You can use signal() to specify how a signal will be handled when raise() is invoked.

In C++ only, the use of signal() and raise() with try(), catch(), or throw() is undefined. The use of signal() and raise() with destructors is also undefined.

Special behavior for POSIX: To obtain access to the special POSIX behavior for raise(), the POSIX runtime option must be set ON, and the version of MVS™ must be 4.3 or higher.

The raise() function sends the signal, sig, to the process that issued the raise(). If the signal is not blocked, it is delivered to the sender before raise() returns. See Table 1 in the description of the sigaction() function for the list of signals supported.

You can use signal() or sigaction() to specify how a signal will be handled when raise() is invoked.

Special behavior for XPG4.2: To obtain access to the special POSIX behavior for raise(), the POSIX runtime option must be set ON, and the version of MVS must be 4.3 or higher.

Several other functions are available to the XPG4.2 application for affecting the behavior of a signal:

- bsd_signal()
- sigignore()
- sigset()

Special behavior for C++: The behavior when mixing signal-handling with C++ exception handling is undefined. Also, the use of signal-handling with constructors and destructors is undefined.

Usage note

The use of the SIGTHSTOP and SIGTHCONT signal is not supported with this function.

Returned value

If successful, raise() returns 0.

If unsuccessful, raise() returns nonzero.

Special behavior for XPG4: The raise() function sets errno to one of the following values:

Error Code	Description
EINVAL	The value of the *sig* argument is an invalid signal number.

Example

CELEBR01

```
/* CELEBR01

   This example establishes a signal handler called sig_hand for the
   signal SIGUSR1.
   The signal handler is called whenever the SIGUSR1 signal is raised and
   will ignore the first nine occurrences of the signal.
```

```
   On the tenth raised signal, it exits the program with an error code of 10.
   Note that the signal handler must be reestablished each time it is called.

 */
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>

#ifdef __cplusplus
extern "C" {
#endif
 void sig_hand(int);
#ifdef __cplusplus
}
#endif
int i;

int main(void)
{
   signal(SIGUSR1, sig_hand);  /* set up handler for SIGUSR1 */
   for (i=0; i<10; ++i)
      raise(SIGUSR1);             /* signal SIGUSR1 is raised */
}                            /* sig_hand() is called */

void sig_hand(int dummy)
{
   static int count = 0;          /* initialized only once */

   count++;
   if (count == 10)   /* ignore the first 9 occurrences of this signal */
   {
      printf("reached 10th signal\n");
      exit(10);
   }
   else
      signal(SIGUSR1, sig_hand);  /* set up the handler again */
}
```

Related information

- signal.h – Exception handling
- bsd_signal() – BSD version of signal()
- kill() – Send a signal to a process
- killpg() – Send a signal to a process group
- pthread_kill() – Send a signal to a thread
- sigaction() – Examine or change a signal action
- sighold() – Add a signal to a thread
- sigignore() – Set disposition to ignore a signal
- signal() – Handle interrupts
- sigprocmask() – Examine or change a thread
- sigset() – Change a signal action or a thread

Parent topic:

→ Library functions

Previous
QueryWorkUnitClassification() – WLM query enclave classification service

Next
rand() – Generate random number