**NAME**
> sigqueue - queue a signal and data to a process

**SYNOPSIS**
> **#include <signal.h>**
>
> **int sigqueue(pid_t** *pid***, int** *sig***, const union sigval** *value***);**

Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

> **sigqueue**(): _POSIX_C_SOURCE >= 199309L

**DESCRIPTION**
> **sigqueue**() sends the signal specified in *sig* to the process whose PID is given in *pid*. The permissions required to send a signal are the same as for kill(2). As with kill(2), the null signal (0) can be used to check if a process with a given PID exists.
>
> The *value* argument is used to specify an accompanying item of data (either an integer or a pointer value) to be sent with the signal, and has the following type:
>
>> union sigval {
>> int sival_int;
>> void *sival_ptr;
>> };
>
> If the receiving process has installed a handler for this signal using the **SA_SIGINFO** flag to sigaction(2), then it can obtain this data via the *si_value* field of the *siginfo_t* structure passed as the second argument to the handler. Furthermore, the *si_code* field of that structure will be set to **SI_QUEUE**.

**RETURN VALUE**
> On success, **sigqueue**() returns 0, indicating that the signal was successfully queued to the receiving process. Otherwise, -1 is returned and *errno* is set to indicate the error.

**ERRORS**
> **EAGAIN**
>> The limit of signals which may be queued has been reached. (See signal(7) for further information.)
>
> **EINVAL**
>> *sig* was invalid.
>
> **EPERM**
>> The process does not have permission to send the signal to the receiving process. For the required permissions, see kill(2).
>
> **ESRCH**
>> No process has a PID matching *pid*.

**VERSIONS**
> **sigqueue**() and the underlying **rt_sigqueueinfo**() system call first appeared in Linux 2.2.

**ATTRIBUTES**
> For an explanation of the terms used in this section, see attributes(7).

| Interface | Attribute | Value |
|-----------|-----------|-------|
| **sigqueue**() | Thread safety | MT-Safe |

**CONFORMING TO**
> POSIX.1-2001, POSIX.1-2008.

**NOTES**
> If this function results in the sending of a signal to the process that invoked it, and that signal was not blocked by the calling thread, and no other threads were willing to handle this signal (either by having it unblocked, or by waiting for it using sigwait(3)), then at least some signal must be delivered to this thread

before this function returns.

**C library/kernel differences**

On Linux, **sigqueue**() is implemented using the rt_sigqueueinfo(2) system call. The system call differs in its third argument, which is the *siginfo_t* structure that will be supplied to the receiving process's signal handler or returned by the receiving process's sigtimedwait(2) call. Inside the glibc **sigqueue**() wrapper, this argument, *uinfo*, is initialized as follows:

```
uinfo.si_signo = sig; /* Argument supplied to sigqueue() */
uinfo.si_code = SI_QUEUE;
uinfo.si_pid = getpid(); /* Process ID of sender */
uinfo.si_uid = getuid(); /* Real UID of sender */
uinfo.si_value = val; /* Argument supplied to sigqueue() */
```

**SEE ALSO**

kill(2), rt_sigqueueinfo(2), sigaction(2), signal(2), pthread_sigqueue(3), sigwait(3), signal(7)

**COLOPHON**

This page is part of release 4.10 of the Linux *man-pages* project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at https://www.kernel.org/doc/man-pages/.