

# **BAHIR DAR UNIVERSITY**

## **BAHIR DAR INSTITUTE TECHNOLOGY**

### **FACULTY OF COMPUTING**

#### **Software Engineering Department**

**Course : Operating System and System programming**

**Individual assignment ||**

## **System Call**

**Name : Esubalew Kunta**

**ID : BDU1307727**

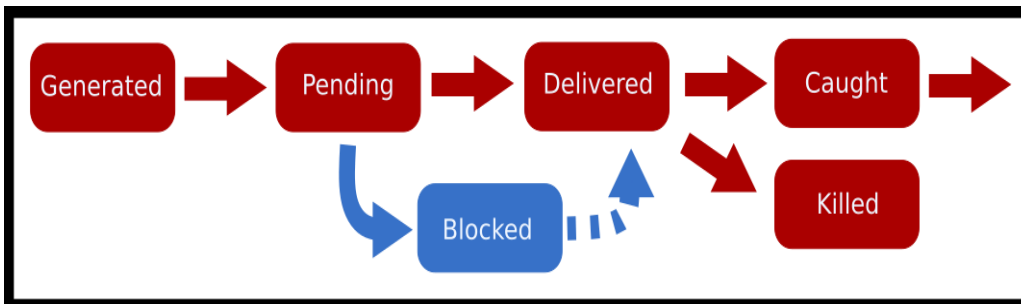
**Submission date : July 23, 2022**

**Submitted to : lecturer Wendimu B.**

# sigprocmask()

## what is sigprocmask() ?

**sigprocmask()** is a function used to fetch and/or change the signal mask of the calling thread. signal mask is the collection of signals that are currently blocked. it is the set of signals whose delivery is currently blocked for the caller. Each process has its own signal mask. When you create a new process it inherits its parent's mask. You can block or unblock signals with total flexibility by modifying the signal mask.



This function allows the calling process to examine (query) or change its signal mask. Each process has a signal mask that specifies a set of signals that cannot be raised. These are called blocked signals. A blocked signal can be sent to a process, but it remains pending until it is unblocked and subsequently raised.

The signals SIGKILL or SIGStop cannot be caught, blocked or ignored. For these signals, the default action has to happen. Any attempt to use **sigprocmask()** to block these signals is simply ignored, and no error is returned.

## Syntax

```
#include <signal.h>
int sigprocmask( int how, const sigset_t *set,
                 sigset_t *oset );
```

## SYNOPSIS

```
#include <signal.h>

/* Prototype for the glibc wrapper function */
int sigprocmask(int how, const sigset_t *restrict set,
                sigset_t *restrict oldset);

#include <signal.h>
```

```
/* Definition of SIG_* constants */
```

```
#include <sys/syscall.h>
```

```
/* Definition of SYS_* constants */
```

```
#include <unistd.h>
```

```
/* Prototype for the underlying system call */
```

```
int syscall(SYS_rt_sigprocmask, int how, const kernel_sigset_t *set, kernel_sigset_t  
*oldset, size_t sigsetsize);
```

```
/* Prototype for the legacy system call (deprecated) */
```

```
int syscall(SYS_sigprocmask, int how, const old_kernel_sigset_t *set,  
old_kernel_sigset_t *oldset);
```

## Parameters

### **how**

(Input) The way in which the signal set is changed, indicates the type of change

It's the manner in which you want to change the set; one of:

- **SIG\_BLOCK** — Indicates that the set of signals given by *set* should be blocked, in addition to the set currently being blocked.  
add the signals pointed to by *set* to the thread mask.
- **SIG\_UNBLOCK** — Indicates that the set of signals given by *set* should not be blocked. These signals are removed from the current set of signals being blocked.  
remove the signals pointed to by *set* from the thread mask.
- **SIG\_SETMASK** — set the thread mask to be the signals pointed to by *set*.  
that the set of signals given by *set* should replace the old set of signals being blocked.

### **\*set**

Points to the new set.

- The *set* parameter points to a signal set giving the new signals that should be blocked or unblocked (depending on the value of *how*), or it points to the new signal mask if the value of *how* was **SIG\_SETMASK**. If *set* is a NULL pointer, the set of blocked signals is not changed. If *set* is NULL, the value of *how* is ignored.

- (Input) A pointer to a set of signals to be used to change the currently blocked set. May be NULL.
- NULL, or a pointer to a `sigset_t` object that defines the signals that you want to change in the thread's signal mask. If this argument is NULL, the *how* argument is ignored.

**\*o`set`**

Is a return pointer to the old set

(Output) A pointer to the space where the previous signal mask is stored. May be NULL.

NULL, or a pointer to a `sigset_t` object that the function sets to indicate the thread's current signal mask.

**Return Value**

0	<code>sigprocmask()</code> was successful.
	<code>sigprocmask()</code> was not successful.
-1	The <code>errno</code> variable is set to indicate the error.

**Errors:**

**EAGAIN**

Insufficient system resources are available to mask the signals.

**EFAULT**

The *set* or *oldset* argument points outside the process's allocated address space.

**EINVAL**

Encounters when **how** is not a valid value or the kernel does not support the size passed in *sigsetsize*.

The value specified for the argument is not correct.

A function was passed incorrect argument values, or an operation was attempted on an object and the operation specified is not supported for that type of object.

An argument value is not valid, out of range, or NULL.

One of the following has occurred:

- The value of *how* is not equal to one of the defined values.
- The signal set pointed to by *set* contains a signal that is not within the valid range or a signal that is not supported.

- The value of *how* is invalid, or you tried to set SIGKILL or SIGSTOP to something other than SIG\_DFL.

## Examples 1

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

void catcher( int sig ) {
    printf( "inside catcher() function\n" );
}

int main( int argc, char *argv[] ) {

    time_t start, finish;
    struct sigaction sact;
    sigset_t new_set, old_set;
    double diff;

    sigemptyset( &sact.sa_mask );
    sact.sa_flags = 0;
    sact.sa_handler = catcher;
    sigaction( SIGALRM, &sact, NULL );

    sigemptyset( &new_set );
    sigaddset( &new_set, SIGALRM );
    sigprocmask( SIG_BLOCK, &new_set, &old_set);

    time( &start );
    printf( "SIGALRM signals blocked at %s\n", ctime(&start) );

    alarm( 1 );    /* SIGALRM will be sent in 1 second */

    do {
        time( &finish );
        diff = difftime( finish, start );
    } while (diff < 10);

    sigprocmask( SIG_SETMASK, &old_set, NULL );
    printf( "SIGALRM signals unblocked at %s\n", ctime(&finish) );

    return( 0 );
}
```

```
}
```

### **Output :**

```
SIGALRM signals blocked at Sun Jul 24 10:08:38 2022
```

After 10 minutes

```
/tmp/44WfhE1lWa.o
```

```
SIGALRM signals blocked at Sun Jul 24 10:08:38 2022
```

```
inside catcher() function
```

```
SIGALRM signals unblocked at Sun Jul 24 10:08:48 2022
```

### **Example 2**

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

```
#include <unistd.h>
```

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

```
int main()
```

```
{
```

```
    sigset_t old_set,new_set;
```

```
    sigemptyset(&old_set);
```

```
    sigemptyset(&new_set);
```

```
    if(sigaddset(&old_set,SIGSEGV)==0)
```

```
    {
```

```
        printf("sigaddset successfully added for SIGSEGV\n");
```

```
    }
```

```
    sigprocmask(SIG_BLOCK,&new_set,&old_set); // SIGSEGV signal is masked
```

```
    kill(0,SIGSEGV);
```

```
//************************************************************************
```

```

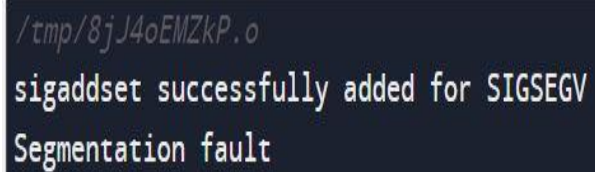
if(sigaddset(&new_set,SIGRTMIN)==0)
{
    printf("sigaddset successfully added for SIGRTMIN\n");
}
if(sigprocmask(SIG_BLOCK,&new_set,&old_set)==-1) // SIGRTMIN signal is masked
{
    perror("sigprocmask");
}
kill(0,SIGSEGV);

**** Unblock all signals ****

if(sigprocmask(SIG_UNBLOCK,&new_set,&old_set)==-1) // SIGRTMIN signal is unmasked
{
    perror("sigprocmask");
}
}

```

Output[



```

/tmp/8jJ4oEMZkP.o
sigaddset successfully added for SIGSEGV
Segmentation fault

```

## Summery

When the **sigprocmask** function is used to change the signal mask of the calling process, it enables the process for signals if the process is not already enabled for signals. If the system has not been enabled for signals, **sigprocmask()** is not successful, and an [ENOTSIGINIT] error is returned.

Typically, `sigprocmask(SIG_BLOCK, ...)` is used to block signals during a critical section of code. At the end of the critical section of code, `sigprocmask(SIG_SETMASK, ...)` is used to restore the mask to the previous value returned by `sigprocmask(SIG_BLOCK, ...)`



## References

<https://pubs.opengroup.org/onlinepubs/7908799/xsh/sigprocmask.html>

<https://www.mkssoftware.com/docs/man3/sigprocmask.3.asp>

<https://man7.org/linux/man-pages/man2/sigprocmask.2.html>

[https://www.ibm.com/docs/en/i/7.3?topic=ssw\\_ibm\\_i\\_73/apis/sigpmsk.htm](https://www.ibm.com/docs/en/i/7.3?topic=ssw_ibm_i_73/apis/sigpmsk.htm)

[https://www.qnx.com/developers/docs/7.1/index.html#com.qnx.doc.neutrino.lib\\_ref/topic/s/sigprocmask.html](https://www.qnx.com/developers/docs/7.1/index.html#com.qnx.doc.neutrino.lib_ref/topic/s/sigprocmask.html)

