sigaction()

Examines or changes the action taken by a process when it receives a specific signal.

```
#include <signal.h>

/**
 * "signum" is the signal number (e.g. SIGINT, SIGTERM...)
 * "act" is a pointer to a struct that specifies
 *      the new action for the signal
 * "oldact" a pointer to a struct where the previous action
 *         for the signal is stored (can be NULL)
 *
 * returns 0 on success. On failure, returns -1 and sets "errno"
 */
int sigaction(int signum, const struct sigaction *act,
                                             struct sigaction *oldact);
```

Enquire about the current handler

- If “act” is NULL, the handler of a given signal is not changed
- If “oldact” is not NULL, the function can be used to enquire about the current handling of a given signal
    - When this happens, “sigaction” fills the “oldact” structure with the current signal handling information

    - If “act” is NULL and “oldact” is not, the function can enquire about the current handler of a given signal without changing it

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

int main(void)
{
    struct sigaction old_action;

    // Query the current handling of SIGUSR1
    if (sigaction(SIGUSR1, NULL, &old_action) == -1)
    {
        perror("sigaction");
        return (1);
    }
    // Print the current handler
    if (old_action.sa_handler == SIG_DFL)
        printf("SIGUSR1 is using the default handler.\n");
    else if (old_action.sa_handler == SIG_IGN)
        printf("SIGUSR1 is being ignored.\n");
    else
        printf("SIGUSR1 has a custom handler.\n");
    return 0;
}
```

SA_SIGINFO

The `SA_SIGINFO` flag is a bitmask that can be set in the “sa_flags” member of the “sigaction” struct to specify how the signal handler should behave.

> ⚠️ Memory from “sa_handler” and “sa_sigaction” may overlap, so the application shouldn’t use both simultaneously.

- If the “SA_SIGINFO” flag is cleared in the “sa_flags” member of the “sigaction” structure:
    - The “sa_handler” member is used to identify the action to be associated with the given signal
    - The signal handler has the following prototype:

      ```
      void handler(int signum);
      ```

    - This is the simpler form of a signal handler
- On the other hand, if “SA_SIGINFO” flag is set in the “sa_flags” member:
    - The “sa_sigaction” member is used to specify a signal-catching function
    - The signal handler has the following prototype:

      ```
      void handler(int signum, siginfo_t *info, void *context);
      ```

    - This is the more advanced form of a signal handler
    - It provides access to the “siginfo_t” structure containing additional info about the signal