```c
/* A simple illustration of exit handlers.  Note that exit handlers
are
 * pushed onto a stack and thus execute in reverse order.
 *
 * Illustrate exiting at different times by invoking
 * this program as
 * ./a.out            exit handlers invoked after return from main
 * ./a.out 1          exit handlers invoked from within func
 * ./a.out 1 2        no exit handlers invoked
 * ./a.out 1 2 3      we call abort(3), no exit handlers invoked
 *
 *
 * */

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

void
my_exit1(void) {
        (void)printf("first exit handler\n");
}

void
my_exit2(void) {
        (void)printf("second exit handler\n");
}

void
func(int argc) {
        (void)printf("In func.\n");
        if (argc == 2) {
                exit(EXIT_SUCCESS);
        } else if (argc == 3) {
                _exit(EXIT_SUCCESS);
        } else if (argc == 4) {
                abort();
        }
}


int
main(int argc, char **argv) {
        (void)argv;
        if (atexit(my_exit2) != 0) {
                perror("can't register my_exit2\n");
                exit(EXIT_FAILURE);
        }

        if (atexit(my_exit1) != 0) {
                perror("can't register my_exit1");
                exit(EXIT_FAILURE);
        }

        if (atexit(my_exit1) != 0) {
                perror("can't register my_exit1");
                exit(EXIT_FAILURE);
        }

        func(argc);

        (void)printf("main is done\n");

        return EXIT_SUCCESS;
}
```