```c
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
#include <signal.h>

volatile int flag = 0;

// signal handler
void interrupted(int val) {
    flag = 1;
}

int main()
{
    struct sigaction old_sig_int, new_sig_int;
    int res;

    //
    // get the old handler
    res = sigaction (SIGINT, NULL, &old_sig_int);
    if (0 != res) {
        perror("could not obtain old handler");
        return -1;
    }

    //
    // setup new handler
    // function pointer for caught signal
    new_sig_int.sa_handler = interrupted;
    new_sig_int.sa_flags = 0;

    // install the new handler
    res = sigaction(SIGINT, &new_sig_int, NULL);
    if (0 != res) {
        perror("could not install new handler");
        return -2;
    }
    char buffer[100];
    flag = 0;
    ssize_t result = read(0, buffer, 100);

    // check for errors
    int error_val = errno;
    if (error_val != 0) {
        printf("\n");
        printf("error_val: %d\n", error_val);
        printf("read() was interrupted by a signal\n");
        printf("flag is: %d\n", flag);
        perror("hmm errno non zero ---> ");
    }

    fprintf(stderr, "managed to read: %d characters\n",
     result);

    printf("buffer contains: ");
    int i;
    for (i = 0; i < result + 20; ++i)
        printf("_%c", buffer[i]);
    printf("\n");


    return 0;
}
```