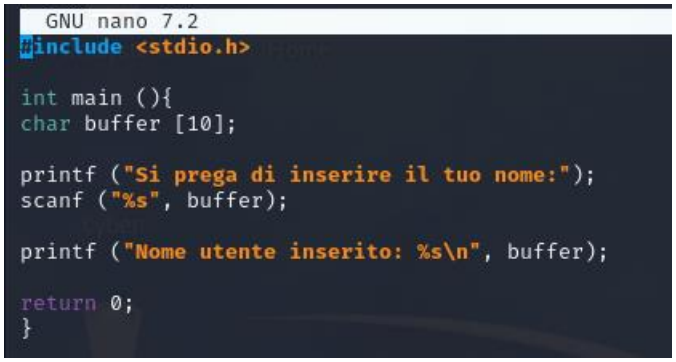
ESERCITAZIONE S7L4

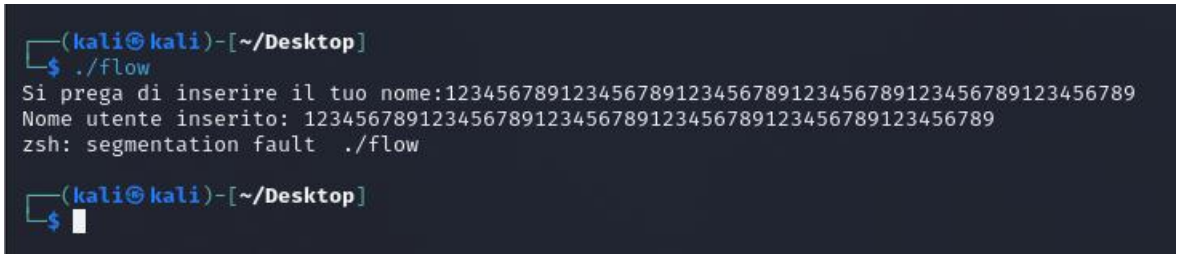
Partendo dal codice originale:



```
 GNU nano 7.2
#include <stdio.h>

int main (){
char buffer [10];

printf ("Si prega di inserire il tuo nome:");
scanf ("%s", buffer);

printf ("Nome utente inserito: %s\n", buffer);

return 0;
}
```

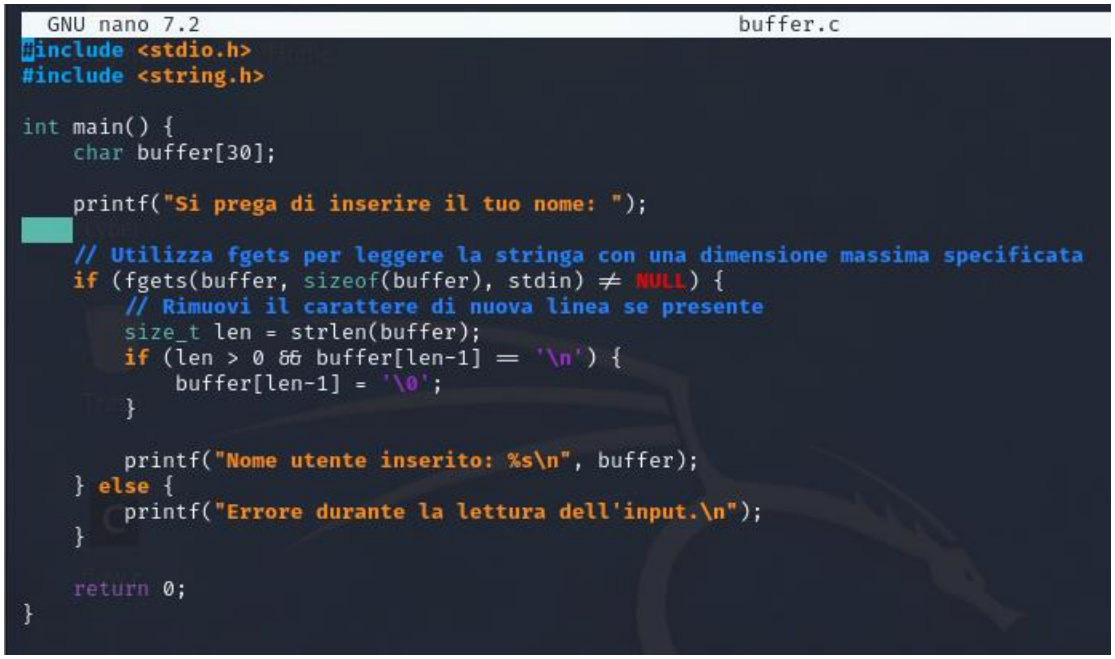
È sufficiente modificare la dichiarazione del buffer, da 10 a 30.

Inserendo una stringa maggiore di 30 caratteri, si può notare che l'errore di segmentazione viene riprodotto correttamente



```
┌──(kali㉿kali)-[~/Desktop]
└─$ ./flow
Si prega di inserire il tuo nome:123456789123456789123456789123456789123456789123456789
Nome utente inserito: 123456789123456789123456789123456789123456789123456789
zsh: segmentation fault  ./flow

┌──(kali㉿kali)-[~/Desktop]
└─$
```

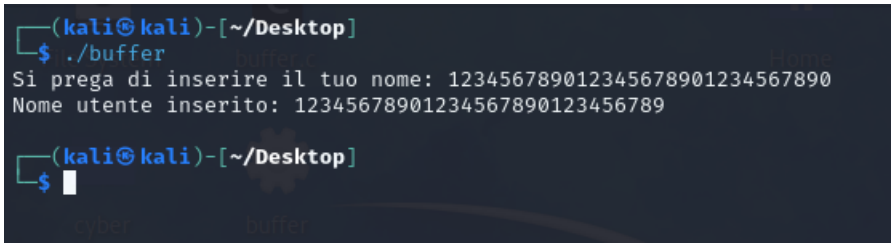
Per evitare questo tipo di errore, conosciuto come buffer overflow, possiamo modificare il codice come segue



```
 GNU nano 7.2                                  buffer.c
#include <stdio.h>
#include <string.h>

int main() {
    char buffer[30];

    printf("Si prega di inserire il tuo nome: ");

    // Utilizza fgets per leggere la stringa con una dimensione massima specificata
    if (fgets(buffer, sizeof(buffer), stdin) != NULL) {
        // Rimuovi il carattere di nuova linea se presente
        size_t len = strlen(buffer);
        if (len > 0 && buffer[len-1] == '\n') {
            buffer[len-1] = '\0';
        }

        printf("Nome utente inserito: %s\n", buffer);
    } else {
        printf("Errore durante la lettura dell'input.\n");
    }

    return 0;
}
```

La funzione fgets specifica la dimensione massima dell'input, impedendo così che venga superato il buffer. La variabile input è dichiarata come un array di caratteri di dimensione MAX_INPUT_LENGTH + 1.
Nel caso in cui l'utente inserisca più di MAX_INPUT_LENGTH caratteri, fgets leggerà e memorizzerà solo i primi MAX_INPUT_LENGTH caratteri nel buffer input, e i caratteri successivi verranno ignorati. Pertanto, non avrai un buffer overflow in questo caso. La precauzione principale è assicurarsi che la dimensione specificata a fgets sia sufficiente per memorizzare l'input. In questo esempio, ciò è gestito correttamente grazie all'uso di sizeof(input), che riflette la dimensione dell'array.

Eseguendo il codice possiamo notare come il carattere in più viene scartato, prendendo in input solo i primi 30



I caratteri in overflow vengono scritti in un'allocazione di memoria che si può rintracciare con l'esecuzione in debug del programma