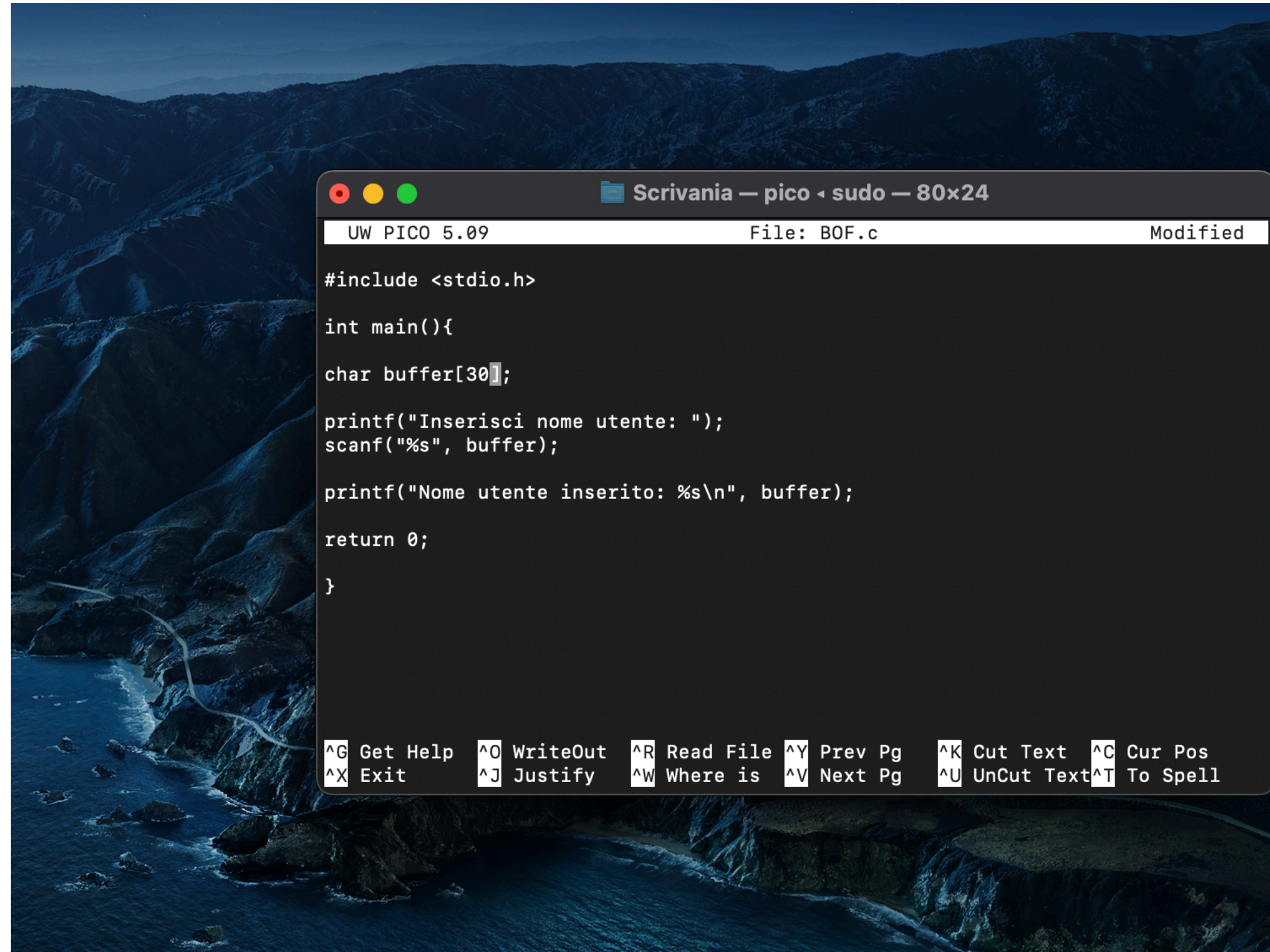


Buffer Overflow



The screenshot shows a terminal window titled "Scrivania — pico « sudo — 80x24". The window contains a C program named "BOF.c". The program's code is as follows:

```
#include <stdio.h>

int main(){

char buffer[30];

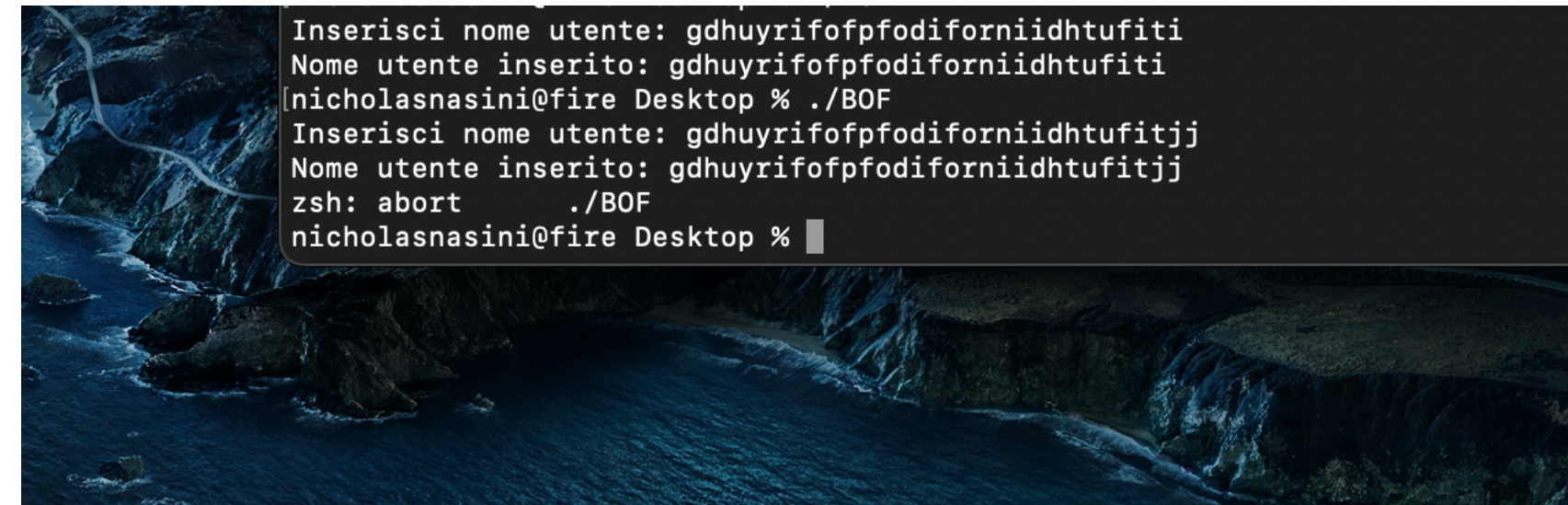
printf("Inserisci nome utente: ");
scanf("%s", buffer);

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

return 0;

}
```

At the bottom of the terminal window, there is a status bar with various keyboard shortcuts: ^G Get Help, ^O WriteOut, ^R Read File, ^Y Prev Pg, ^K Cut Text, ^C Cur Pos, ^X Exit, ^J Justify, ^W Where is, ^V Next Pg, ^U UnCut Text, and ^T To Spell.



The screenshot shows a terminal window with the following text:

```
Inserisci nome utente: gdhuyrifofpfodiforniidhtufiti
Nome utente inserito: gdhuyrifofpfodiforniidhtufiti
nicholasnasini@fire Desktop % ./BOF
Inserisci nome utente: gdhuyrifofpfodiforniidhtufitjj
Nome utente inserito: gdhuyrifofpfodiforniidhtufitjj
zsh: abort      ./BOF
nicholasnasini@fire Desktop %
```

The output shows that the program successfully reads the input string, which is longer than the 30-character buffer, resulting in a segmentation fault (zsh: abort).

- Ho eseguito un semplice codice contenente un char di 30 per dimostrare l'efficacia di un buffer overflow nel momento in cui a seguito di una mancanza di un corretto controllo sia possibile eseguire codice malevolo o sovrascrivere la memoria buffer.
- Nella figura a destra superando il termine di caratteri consentiti il programma ci restituisce un errore di segmentation fault