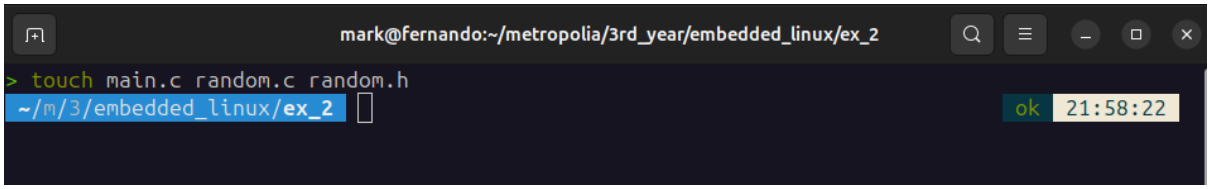
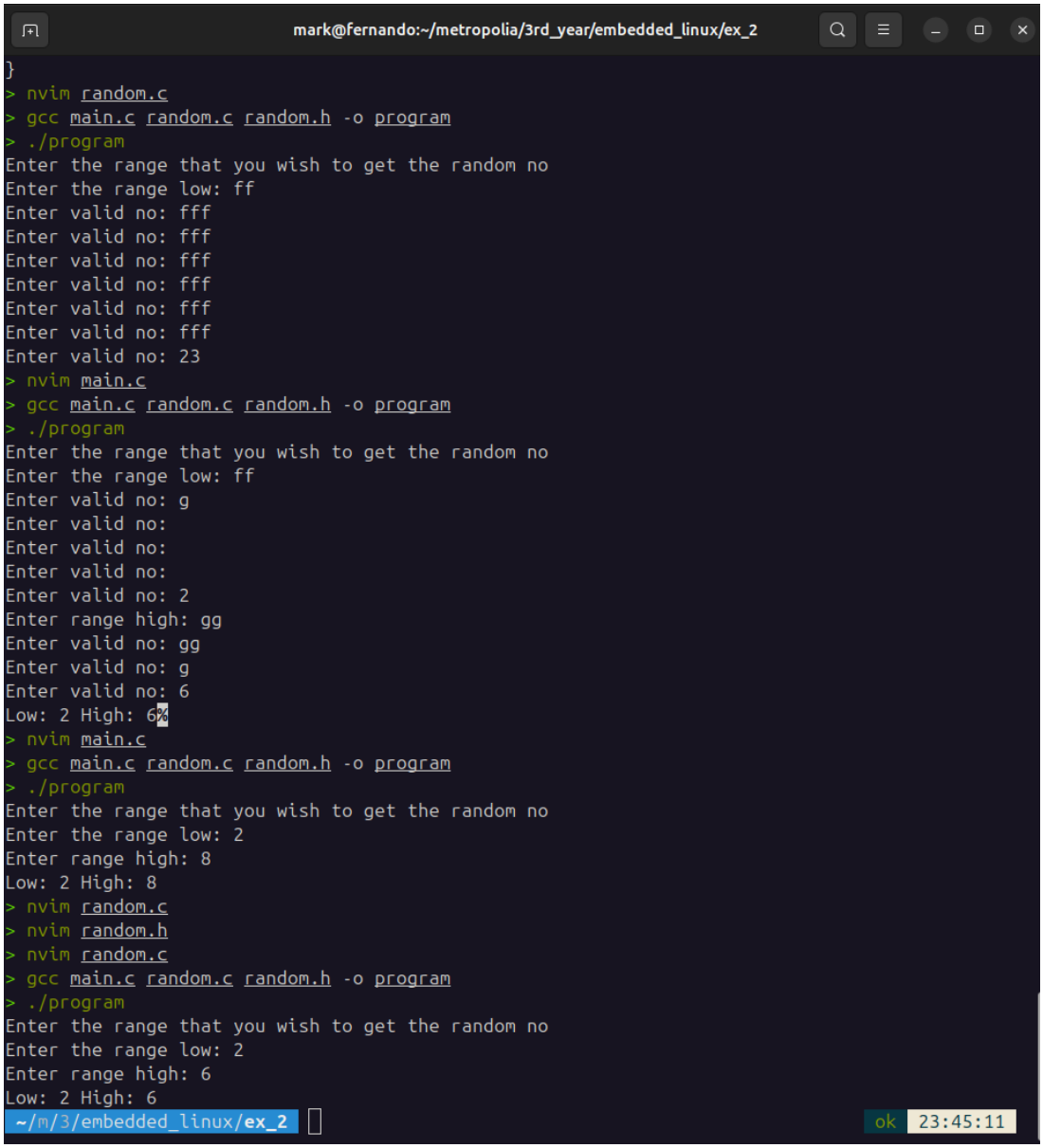
Managing Linux with the Embedded Perspective Exercise 2

Created necessary files.



Testing the code



I took out all errors before doing the preprocessor. Now I have only the warnings.

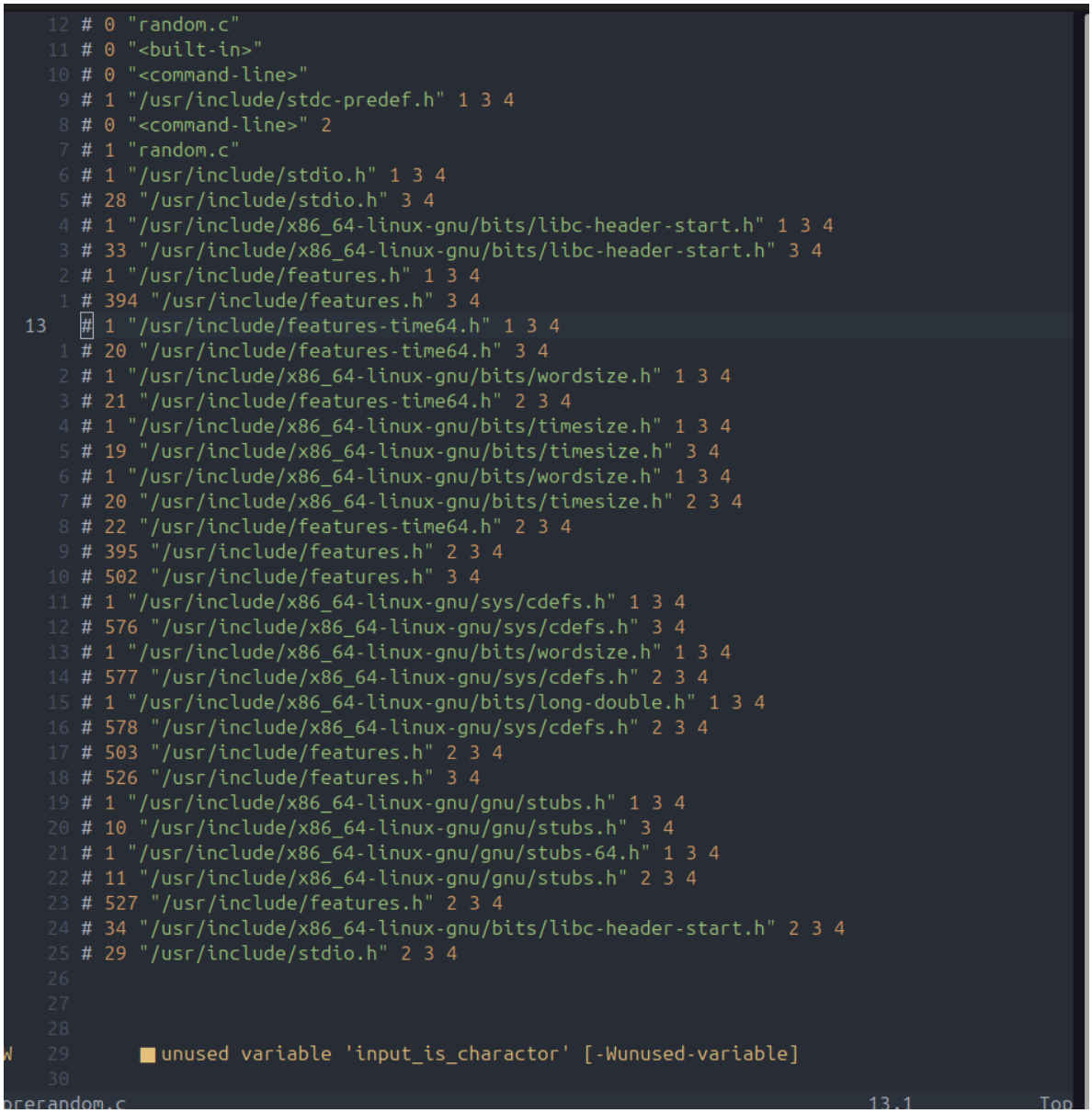
```
mark@fernando:~/metropolia/3rd_year/embedded_linux/ex_2
> gcc -E main.c > premain.c
> gcc -E random.c > prerandom.c
> gcc -E random.h > prerandomh.h
~/m/3/embedded_linux/ex_2                                   ok  15:23:13
```

Preprocessor

Premain.c

```
nvim premain.c
  1   # 0 "main.c"
  1 # 0 "<built-in>"
  2 # 0 "<command-line>"
  3 # 1 "/usr/include/stdc-predef.h" 1 3 4
  4 # 0 "<command-line>" 2
W   5 # 1 "main.c"      unused variable 'buffer' [-Wunused-variable]
  6 # 1 "/usr/include/stdio.h" 1 3 4
  7 # 28 "/usr/include/stdio.h" 3 4
  8 # 1 "/usr/include/x86_64-linux-gnu/bits/libc-header-start.h" 1 3 4
W   9 # 33 "/usr/include/x86_64-linux-gnu/bits/libc-header-start.h" 3 4     unused variable
W  10 # 1 "/usr/include/features.h" 1 3 4     unused variable 'high_state' [-Wunused-variabl
 11 # 394 "/usr/include/features.h" 3 4
 12 # 1 "/usr/include/features-time64.h" 1 3 4
 13 # 20 "/usr/include/features-time64.h" 3 4
 14 # 1 "/usr/include/x86_64-linux-gnu/bits/wordsize.h" 1 3 4
 15 # 21 "/usr/include/features-time64.h" 2 3 4
 16 # 1 "/usr/include/x86_64-linux-gnu/bits/timesize.h" 1 3 4
 17 # 19 "/usr/include/x86_64-linux-gnu/bits/timesize.h" 3 4
 18 # 1 "/usr/include/x86_64-linux-gnu/bits/wordsize.h" 1 3 4
 19 # 20 "/usr/include/x86_64-linux-gnu/bits/timesize.h" 2 3 4
 20 # 22 "/usr/include/features-time64.h" 2 3 4
 21 # 395 "/usr/include/features.h" 2 3 4
 22 # 502 "/usr/include/features.h" 3 4
 23 # 1 "/usr/include/x86_64-linux-gnu/sys/cdefs.h" 1 3 4
 24 # 576 "/usr/include/x86_64-linux-gnu/sys/cdefs.h" 3 4
 25 # 1 "/usr/include/x86_64-linux-gnu/bits/wordsize.h" 1 3 4
 26 # 577 "/usr/include/x86_64-linux-gnu/sys/cdefs.h" 2 3 4
 27 # 1 "/usr/include/x86_64-linux-gnu/bits/long-double.h" 1 3 4
 28 # 578 "/usr/include/x86_64-linux-gnu/sys/cdefs.h" 2 3 4
 29 # 503 "/usr/include/features.h" 2 3 4
 30 # 526 "/usr/include/features.h" 3 4
 31 # 1 "/usr/include/x86_64-linux-gnu/gnu/stubs.h" 1 3 4
 32 # 10 "/usr/include/x86_64-linux-gnu/gnu/stubs.h" 3 4
 33 # 1 "/usr/include/x86_64-linux-gnu/gnu/stubs-64.h" 1 3 4
 34 # 11 "/usr/include/x86_64-linux-gnu/gnu/stubs.h" 2 3 4
 35 # 527 "/usr/include/features.h" 2 3 4
 36 # 34 "/usr/include/x86_64-linux-gnu/bits/libc-header-start.h" 2 3 4
 37 # 29 "/usr/include/stdio.h" 2 3 4
 38
 39
 40
 41
 42
premain.c                                                   1,1           Top
```

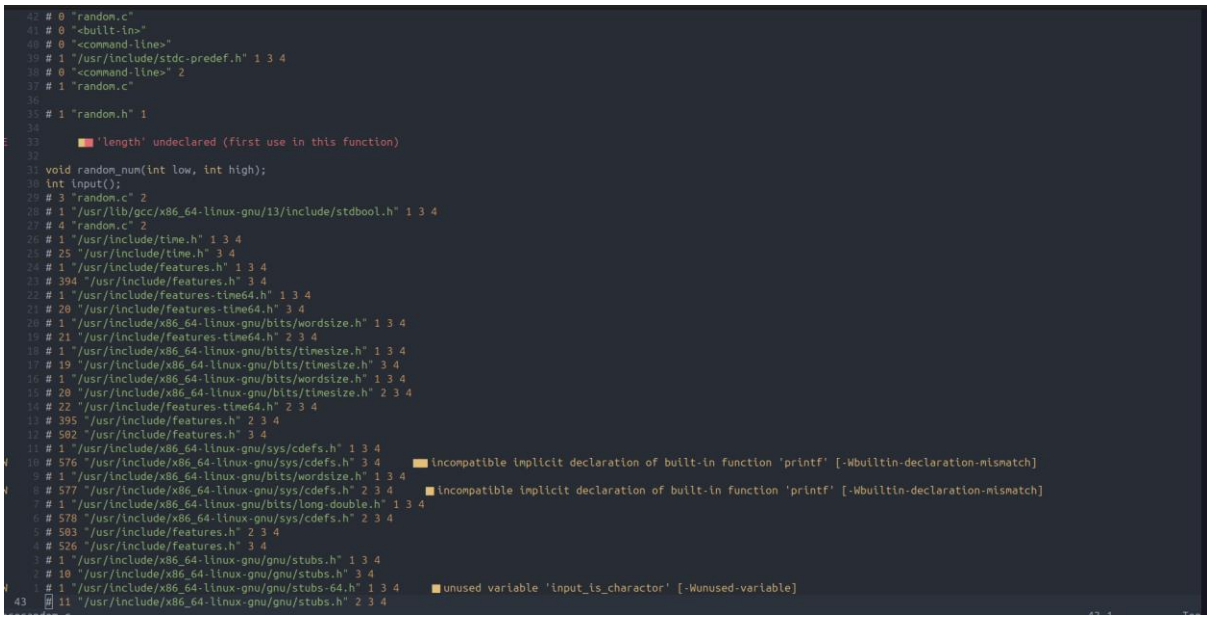
Prerandom.c



In the preprocessing stage all #include are replaced with full header files and Macros are expanded. Also, comments are removed. As a result, the preprocessor file is larger than the original c file.

I could see the warnings, but I didn't see any errors since I had already cleared them before running the preprocessor step. So, I intentionally broke the code to generate errors, so I could study them.

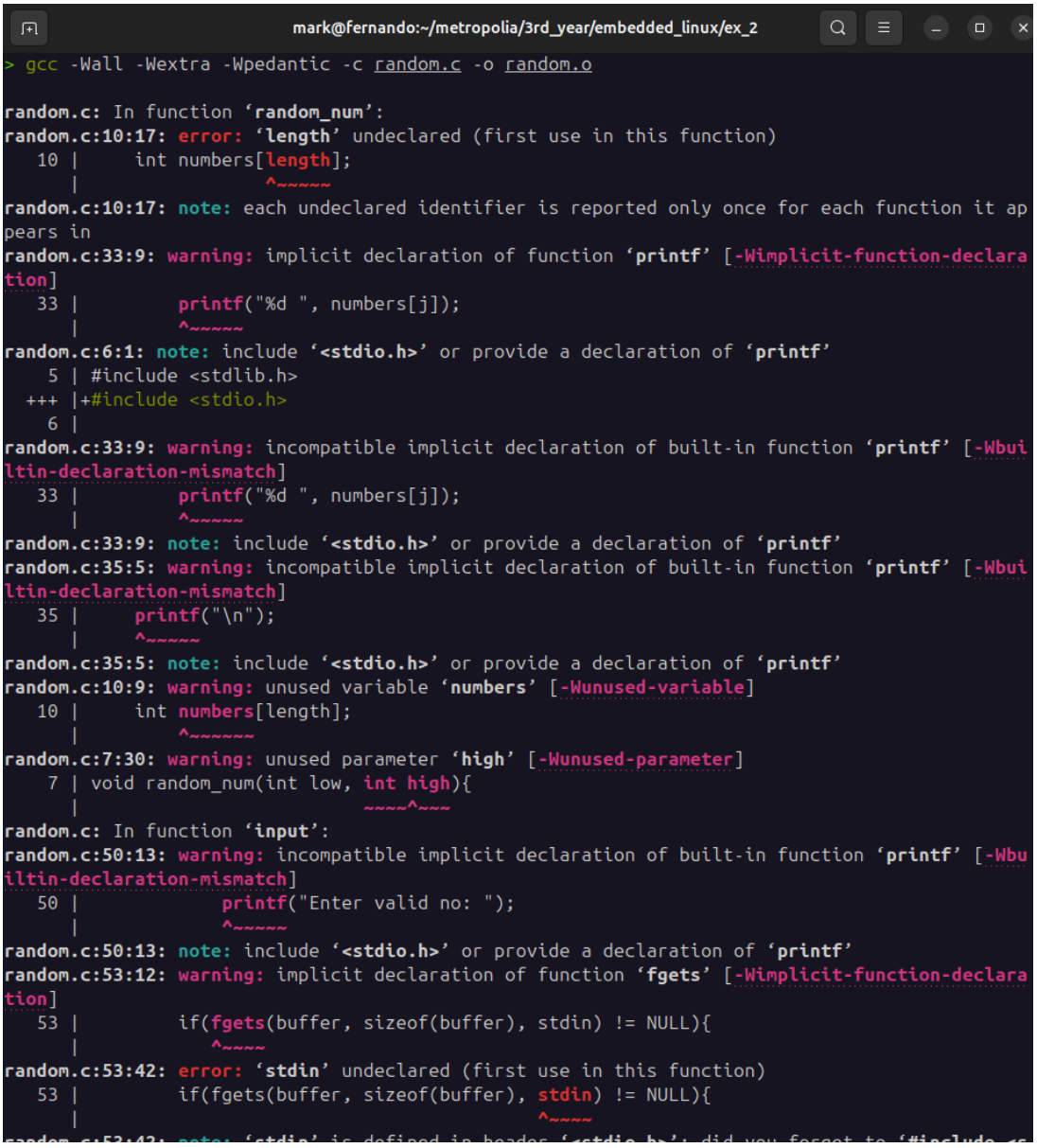
I broke the random.c file.



Here the error says, “**Implicit declaration of function ‘fgets’ [-Wimplicit-function-declaration]**”. It means compiler saying you called fgets but it didn’t declare. This happens because it couldn’t find **stdio.h**

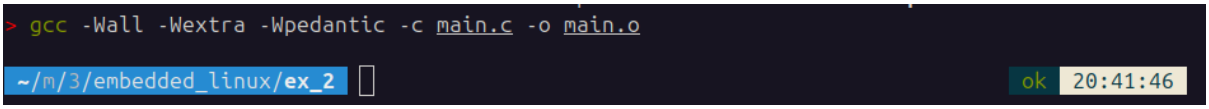
Compilation

Then I compiled both main .c and random.c

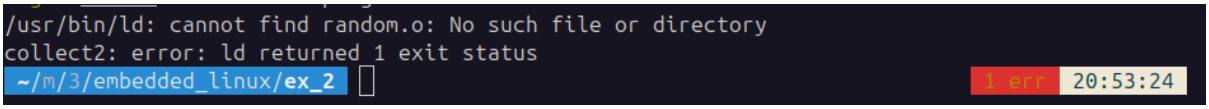


The major issue was that it threw a bunch of errors related to everything connected to stdio.h. It also showed an error saying it couldn’t find the length variable.

I have noticed an interesting thing here.



When I tried to compile main.c without fixing the errors in random.c, it didn't give any errors. What I understood is that the compiler compiles files individually, and each .c file is handled independently. Since main.c had everything, it needed, it compiled correctly. However, issues came up during linking. Even though main.c compiled successfully, random.c didn't. So, when I tried to link, I got output as follows.



We don't have the random.o file because the compiler terminated while compiling random.c.

Flags

-Wall --> Enables common warnings

-Wextra --> Enables additional warnings not included in –wall (e.g. Unused variables)

-Wfatal-erros --> Stops compilation immediately after the first error.

-Wpedantic --> Warns about non-standard practices. It helps developers to write clean code and standard-compliant code.

Other flags

-g, -pg --> debugging and information

-o, os --> Optimization

-std=c99, -std=c11, -ansi --> language standards

-l<library> --> Linking and libraries

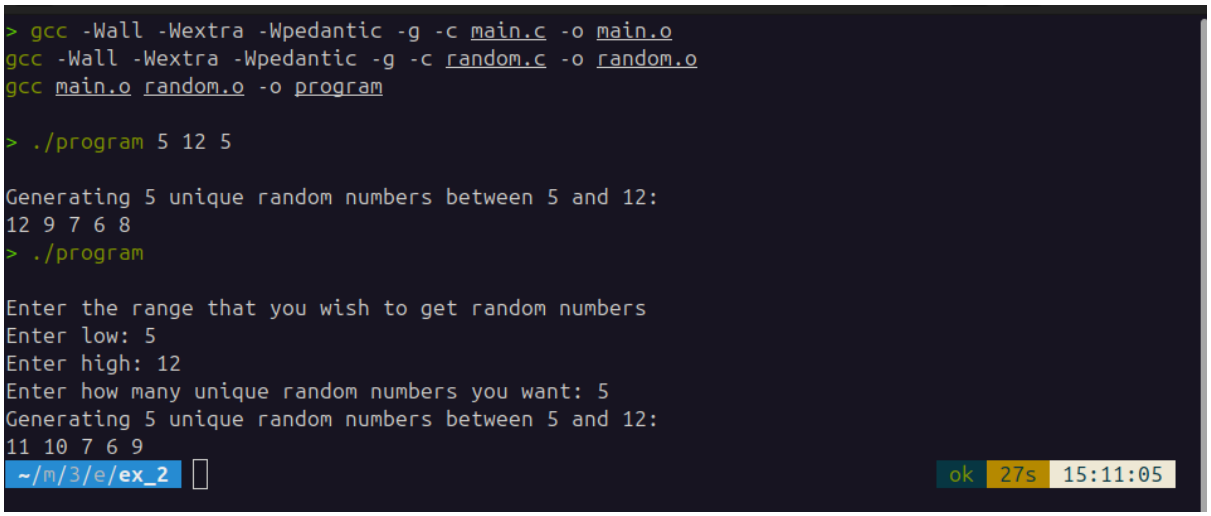
-Wshadow, -Wconversion, -Wunreachable-code --> warnings and errors

-o <file>, -c, -s, -E --> Output control

Linking

gcc main.o random.o -o program

Once we fix all the errors, it compiles and links properly.



Debugging

I used debugger –g flag

gcc -Wall -Wextra -Wpedantic -g -c main.c -o main.o

gcc -Wall -Wextra -Wpedantic -g -c random.c -o random.o

gcc main.o random.o -o program

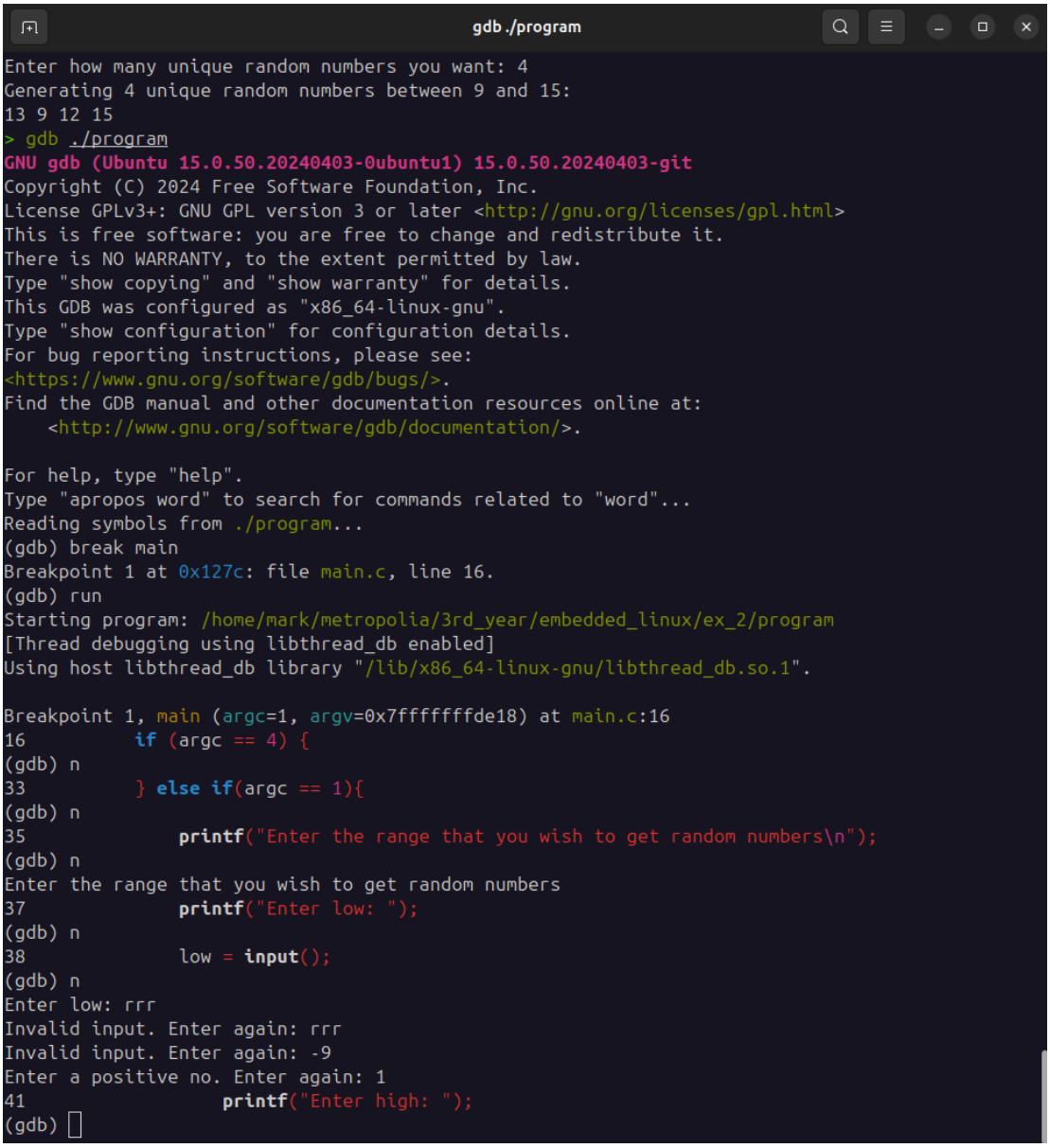
Commands that I used to debug

(gdb)break main --> goes to the main

(gdb) run --> run the debugger

(gdb) next --> next line

(gdb) step --> goes into the function

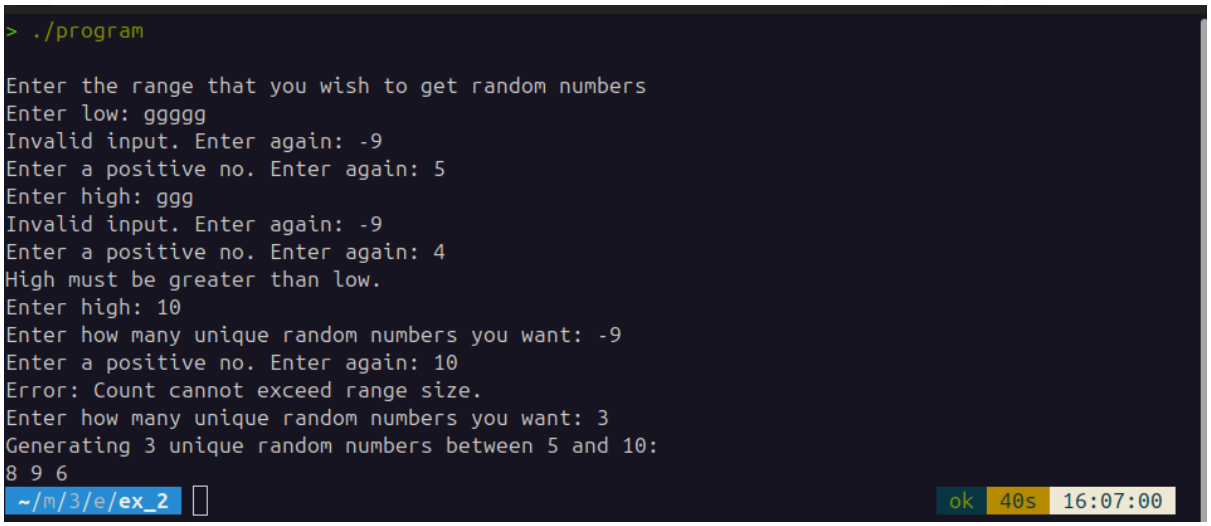


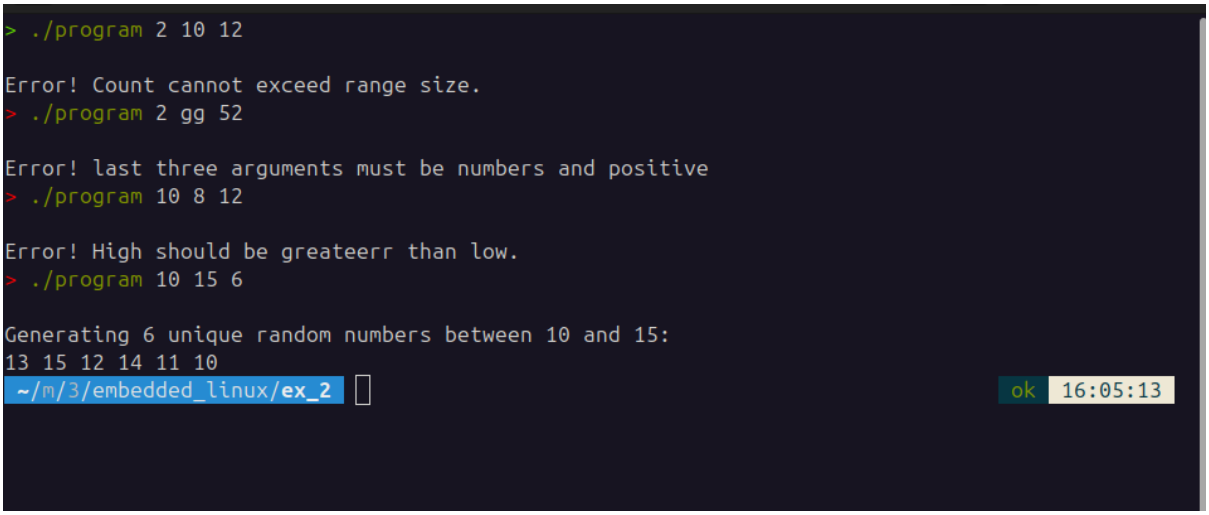
I played around with debugger with different kind of inputs.

I tested out all scenarios.

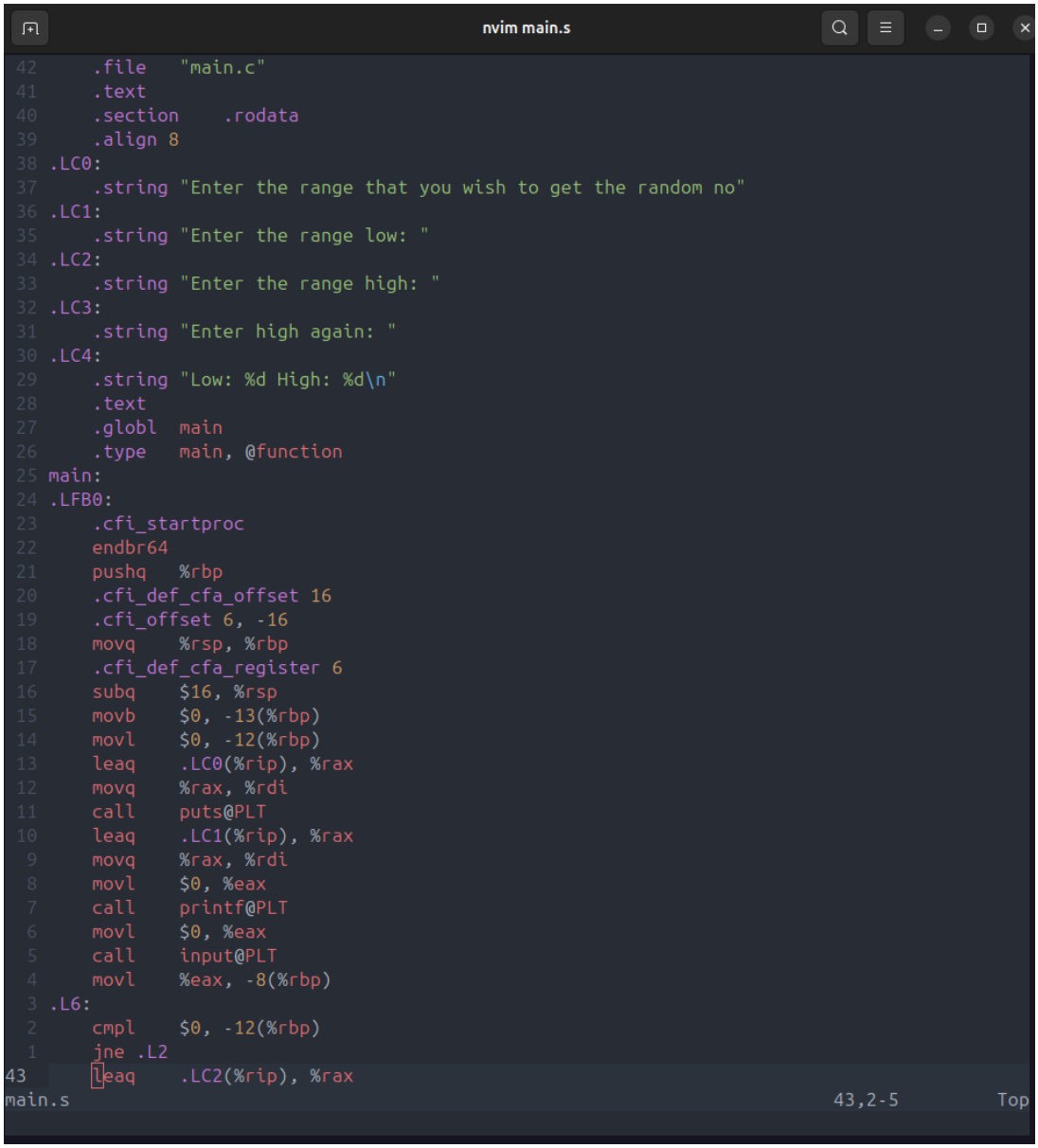
1) Characters
2) negative no
3) For the range, the high number and no of random n must not be less than the low number
4) No of random numbers should be less than the range

Then I tested the program with all possible inputs.





Input validation is fine, and the output is as expected. Then, I also generated the assembly output to see what's inside.



Then I looked at the .o file. Since it’s not a human readable, file I didn't pay much attention to it.