**Module: R1: C Programming**

**Section:** Build Systems **Task:** Source to Binary

**Task**

**Source to Binary**

‭The‬ **‭main.c**‬ ‭I will be using throughout this task is ‬‭given by the following program:‬

‭#include <stdio.h>‬

‭int main() {‬

‭printf("Hello World!\n");‬

‭printf("My first makefile!\n");‬

‭return 0;‬

‭}‬

**‭Makefile:‬**

I‭ will just explain the contents of the Makefile first and then proceed to the C compilation‬

‭pipeline.‬

1. ‭‬‭**CC = gcc:** ‬‭It defines the C compiler to be used‬‭as gcc.‬
2. ‭**all: preprocessor compiler assembler linker run:**‬‭It specifies the target all, which‬ ‭depends on preprocessor, compiler, assembler, linker, and run. When we run make all, it‬ ‭will execute all these steps in sequence.‬
3. **‬‭PHONY: all clean:** ‬‭It declares all and clean as phony targets. Phony targets are‬ ‭those that are not actual files but are simply names for tasks. This is basically used to‬ ‭avoid any confusions.‬
4. **‬‭build: preprocessor compiler assembler linker:** ‬‭It specifies the build target, which‬ ‭depends on preprocessor, compiler, assembler, and linker. As this target is not used in‬ the all target, so we have to explicitly invoke‬‭ ***make build‬‭*** to run these steps.‬
5. **‭‭run:** ‬‭It executes the compiled program ‬‭***a.out***.‬
6. **‬‭linker:** ‬‭It compiles **‬‭main.c** ‬‭and links it to produce an executable named‬‭ a.out‬‭.‬
7. ‭**‬‭assembler:** ‬‭It assembles **‬‭main.s** ‬‭to produce an object file **‬‭main.o‬‭**.‬
8. ‭**‬‭compiler:** ‬‭It compiles‬ **‭main.c‬‭** to produce assembly code‬‭ **main.s‬‭**.‬
9. **‬‭preprocessor:** ‬‭It runs the C preprocessor on‬‭ **main.c** ‬‭to produce preprocessed output.‬
10. **‬‭clean:** ‬‭Removes all generated files (.exe, .o, .s, .out) and clears the terminal.‬

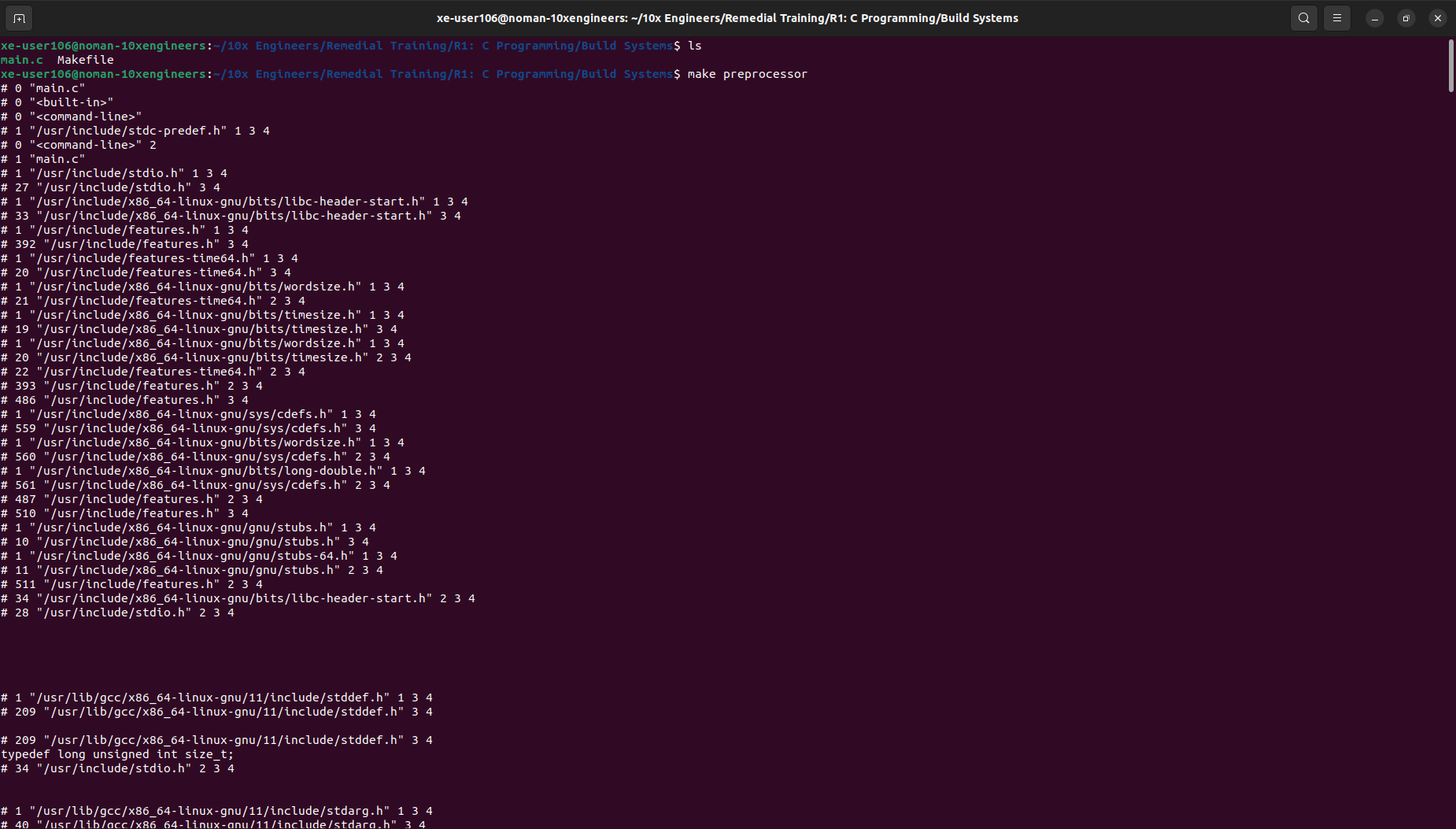
**Compilation Pipeline:**

1. **‭ Preprocessing:‬**

First, let's preprocess main.c using the C preprocessor:‬

*‭make preprocessor‬*

This command will generate a preprocessed version of main.c. Let's take a look at the contents‬ ‭of the preprocessed file. Here’s the output:



‭As we can see, it displays the preprocessed contents of **main.c**, where all preprocessor‬ ‭directives (such as **#include** and **#define**) have been processed.‬

I‭n this file, we can also see the contents of the‬‭ **<stdio.h>** ‬‭header file included and any other‬ ‭preprocessor directives processed.‬

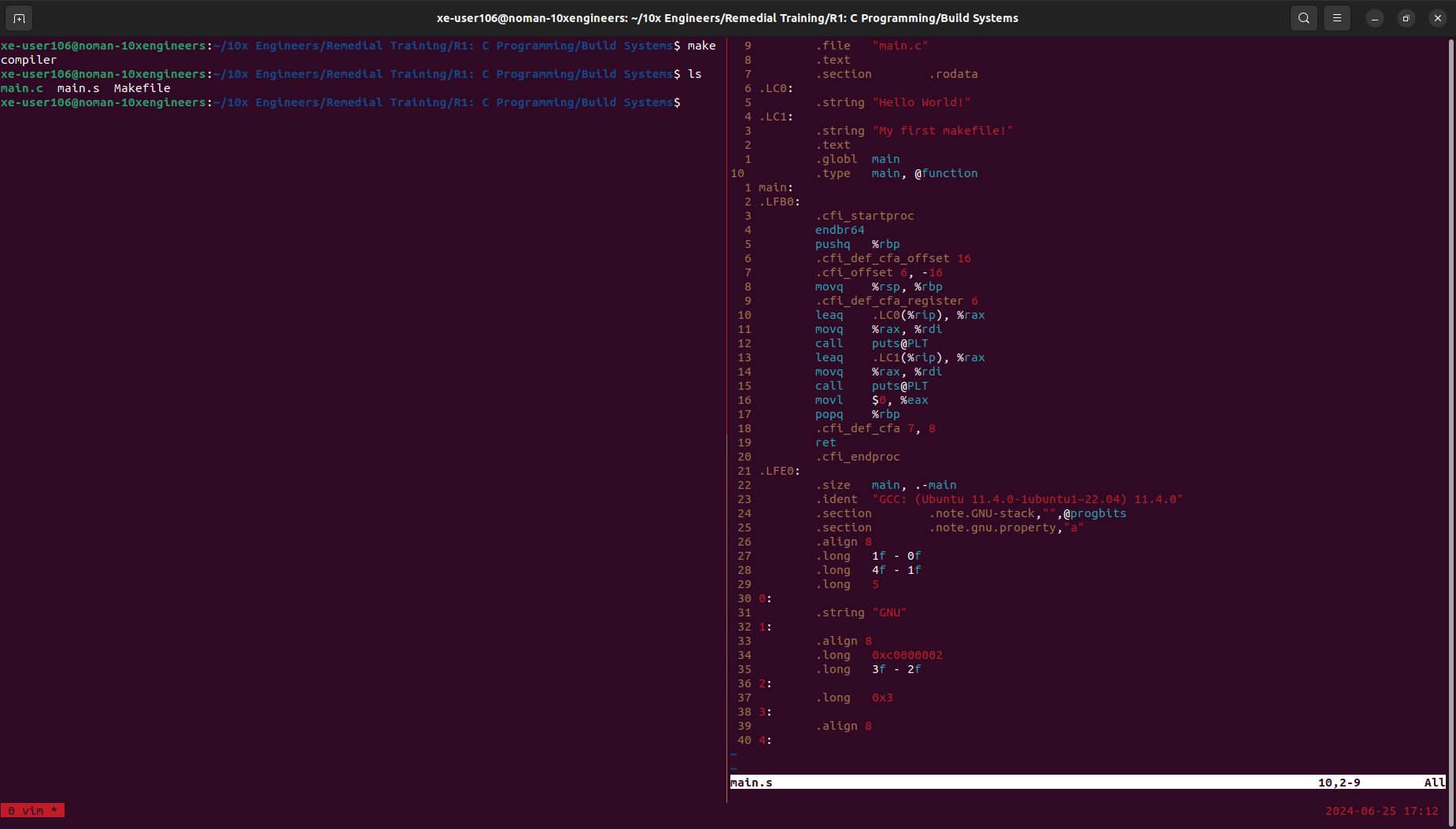
1. **Compiler:**‬

‭Let's compile the preprocessed **main.c** file using the C compiler specified in the Makefile.‬

*‭make compiler*

This command invokes the C compiler to generate assembly code from the preprocessed‬ **‭main.c‬‭**. The output of this command produced an assembly file named **‬‭main.s‬‭**, which contains‬ ‭the assembly code equivalent to the C code in **main.c**.

Let's examine the contents of the generated assembly file. Here's the output:

‬

1. **Assembler:**‬

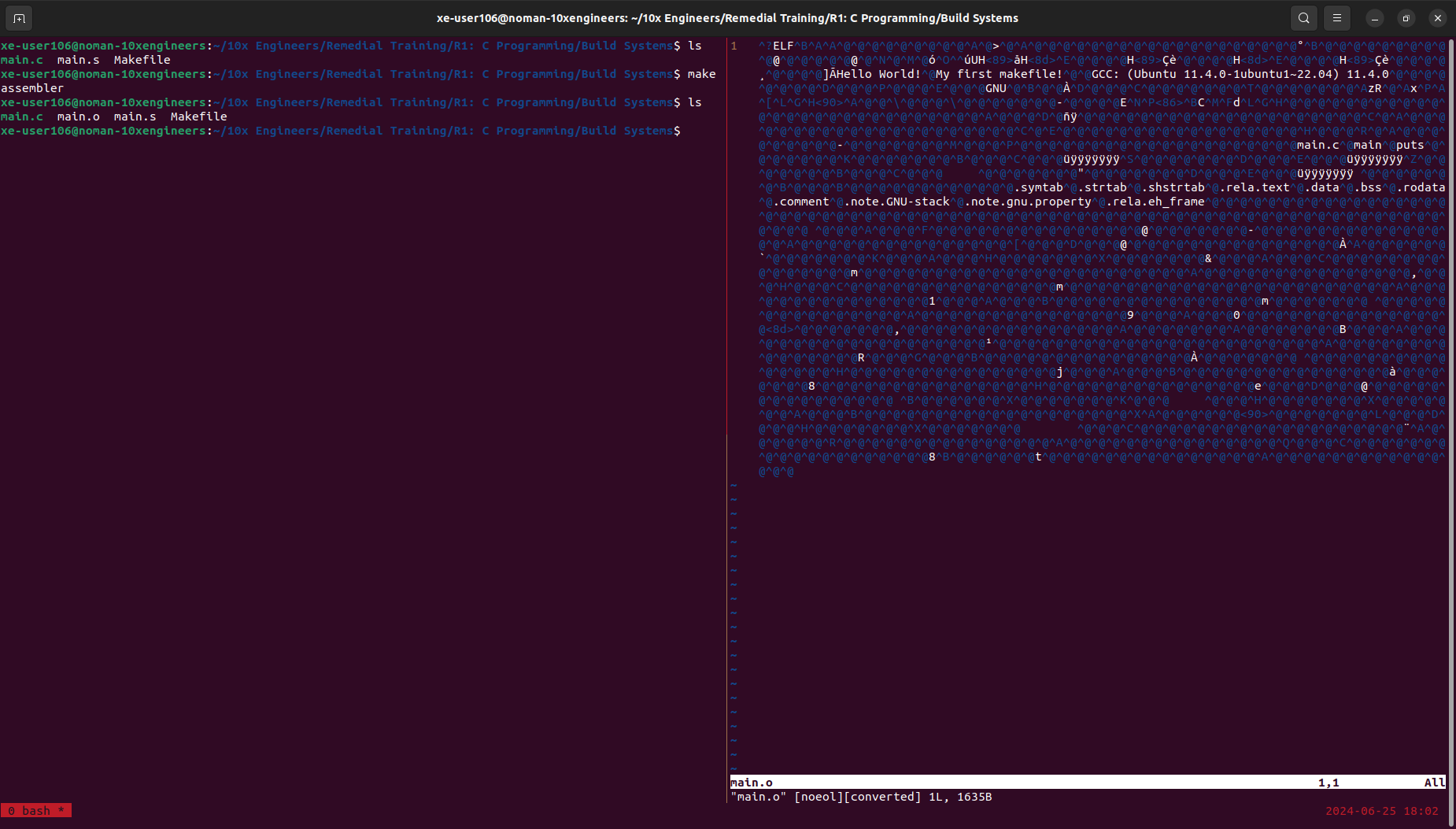
‭The assembling step involves translating the assembly code into machine code (object file).‬

‭Let's assemble the generated **‬‭main.s** ‬‭file using the assembler specified in the Makefile.‬

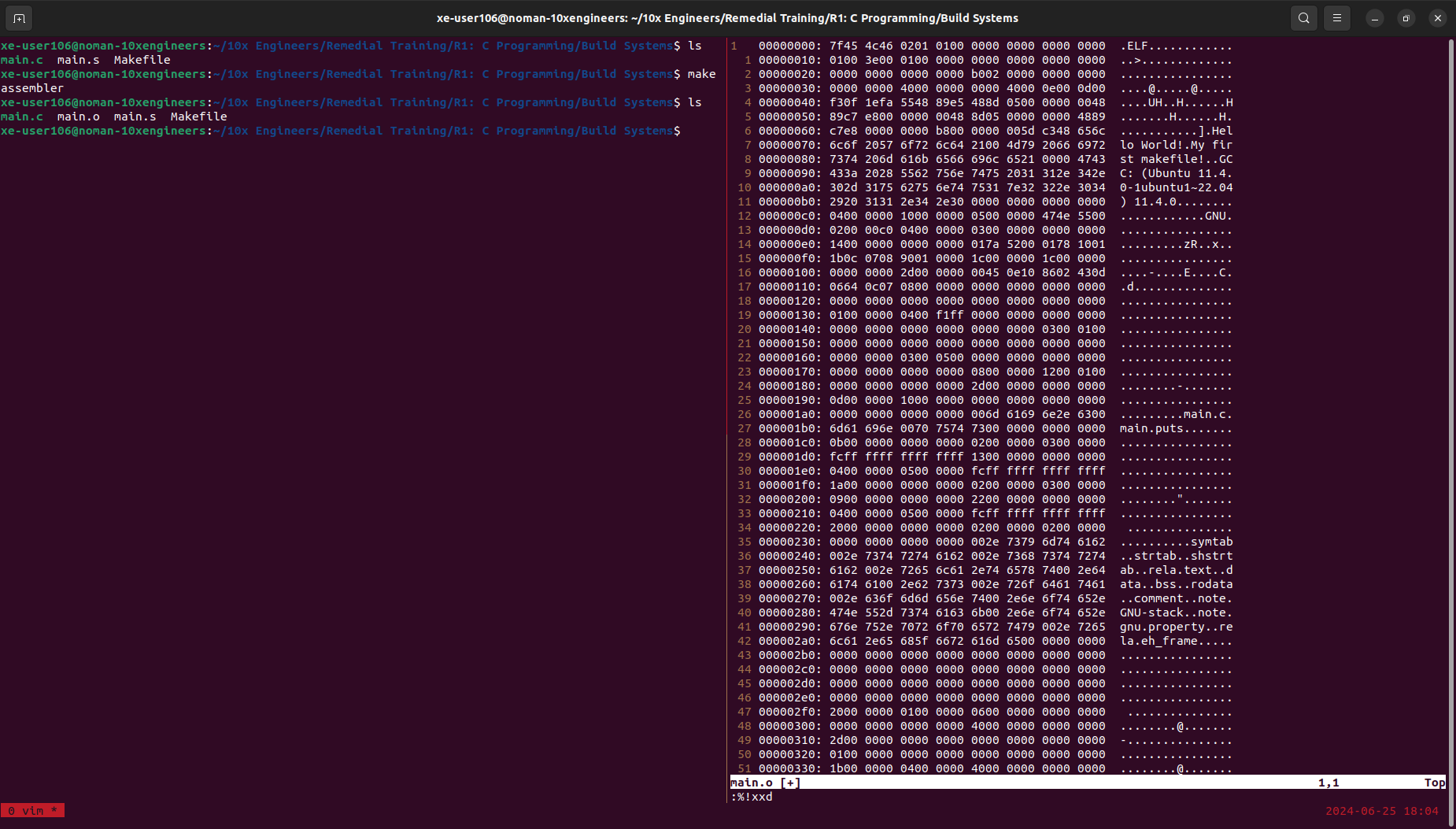
*make assembler*

‭This command invokes the assembler to generate an object file from the assembly file‬‭ **main.s‬‭**.‬

‭Let's examine the contents of the generated object file. Here's the output:

**

*‭*We can convert this binary to hexadecimal information using‬‭ ***%!xxd*** ‬‭command. Here’s a‬ ‭demonstration:



‭The output of the‬‭ **make assembler** ‬‭command produced an object file named‬ **‭main.o‬‭**, which‬ ‭contains the machine code generated from the assembly code in‬‭ **main.s‬‭**.‬

1. **Linker:**‬

‭The linking step involves combining the object file generated from our source file with any‬ necessary system libraries to create an executable program.‬

‭In our case, we'll link the object file‬‭ **main.o** ‬‭to produce the final executable program.

*‭make linker‬*

This command invokes the linker to combine the object file‬‭ **main.o** ‬‭with necessary system‬ libraries to generate the executable program. Let's examine the contents of the generated‬ ‭executable file.‬

‭Here's the output:



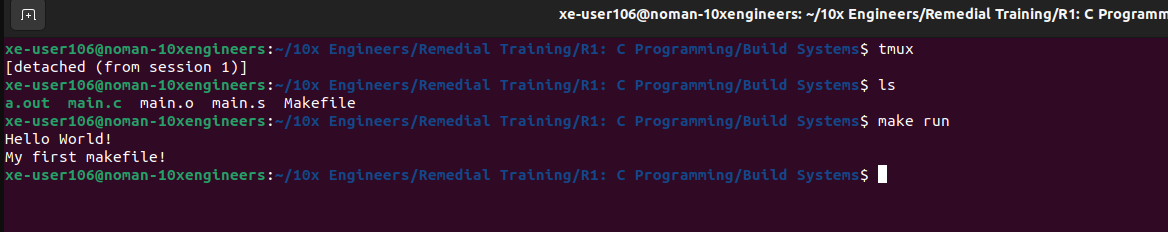
‭The output of this command produced an executable file named‬‭ **a.out‬‭**, which contains the‬ ‭linked program ready for execution.

1. **Run:**‬

‭Now, let's proceed to run the generated executable to verify that it behaves as expected.‬

*‭make run‬*

‭This command will execute the **‬‭a.out** ‬‭file, which is our compiled program. Here’s the output:



This output confirms that our program executed successfully.