85_programando_en_C__cabgome mi Cabo Gomis, Emilio

Emilio Cabo Gomis

Indice

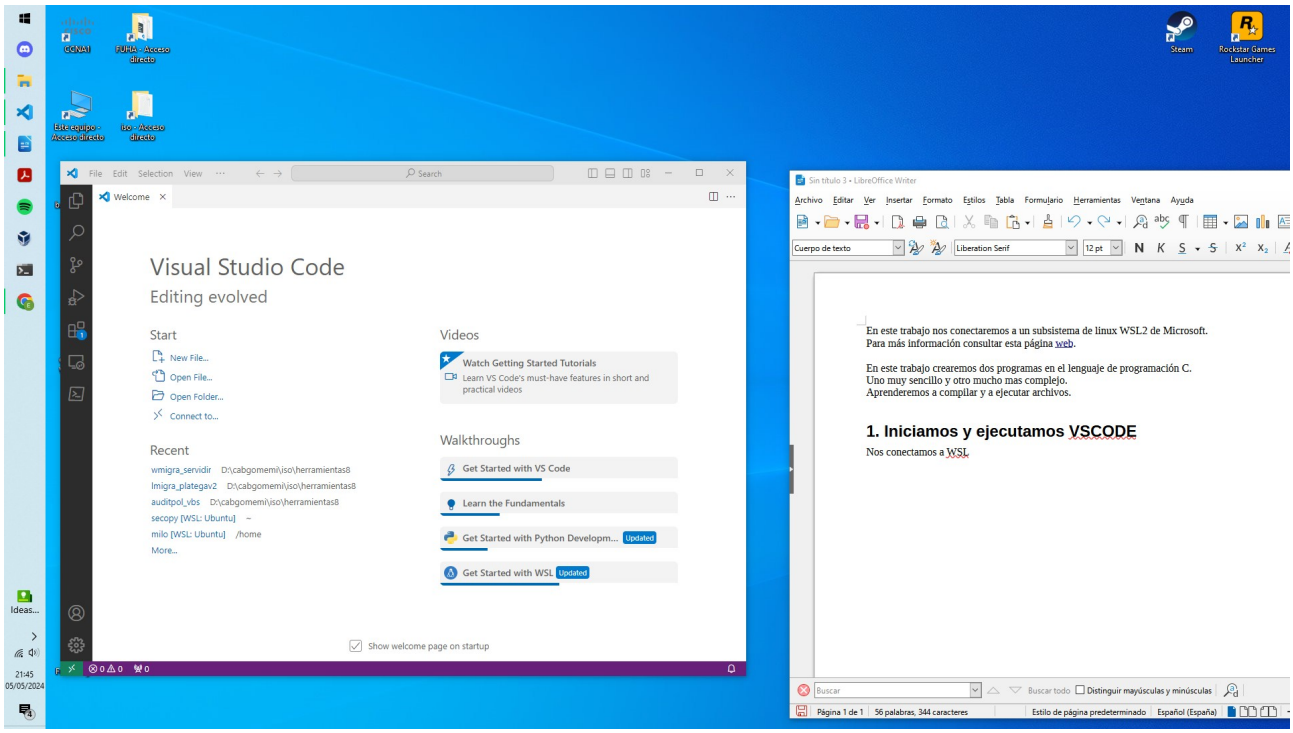
1. Iniciamos y ejecutamos VSCODE....2
Creación primer programa....3
    Compilado....5
Segundo programa....9

En este trabajo nos conectaremos a un subsistema de linux WSL2 de Microsoft.
Para más información consultar esta página web.

En este trabajo crearemos dos programas en el lenguaje de programación C.
Uno muy sencillo y otro mucho mas complejo.
Aprenderemos a compilar y a ejecutar archivos.

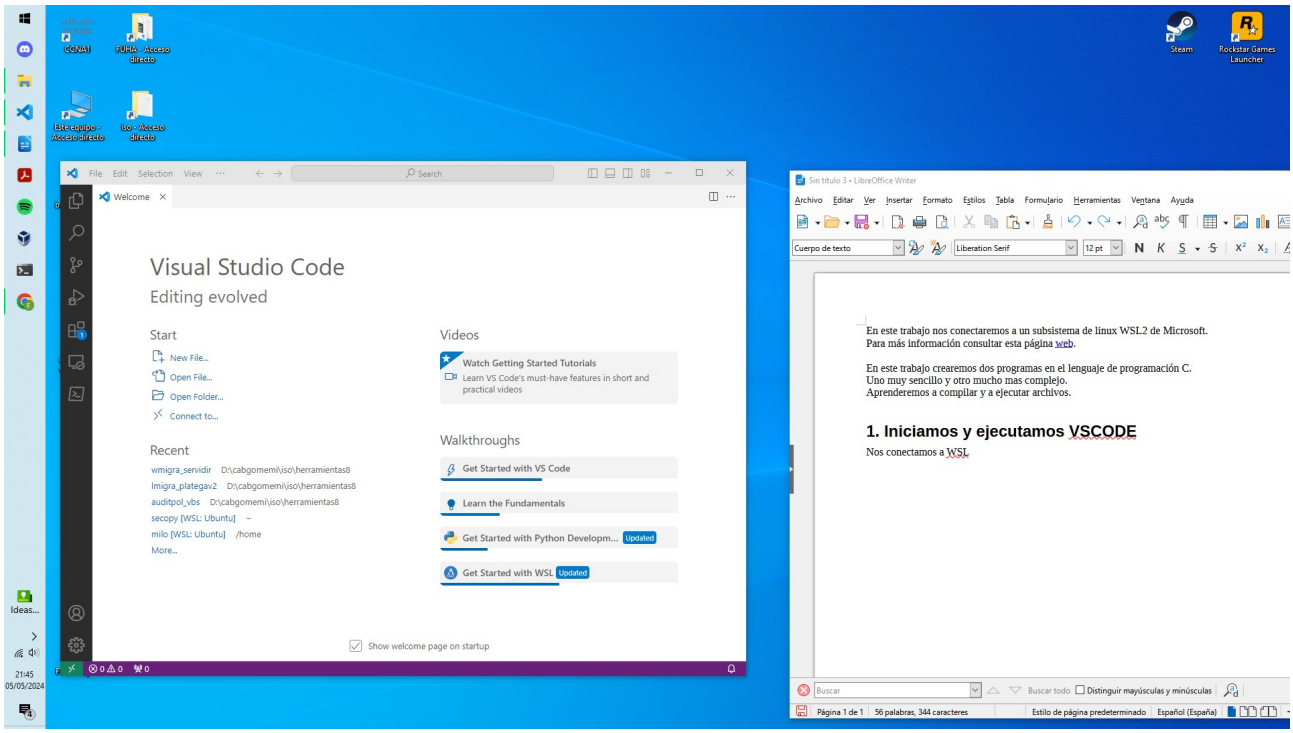
1. Iniciamos y ejecutamos VSCODE

Nos conectamos a WSL



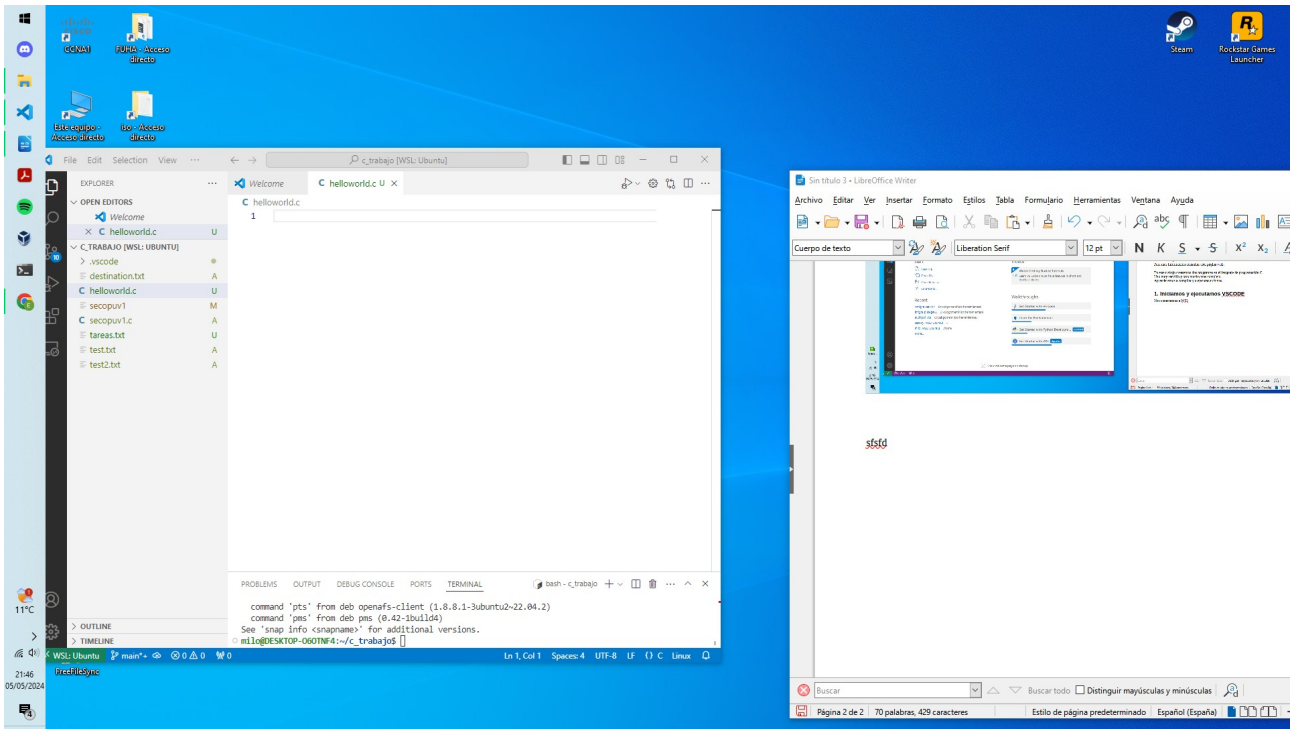
Se inicia el subsistema

Abrimos la carpeta que hemos creado para este propósito.



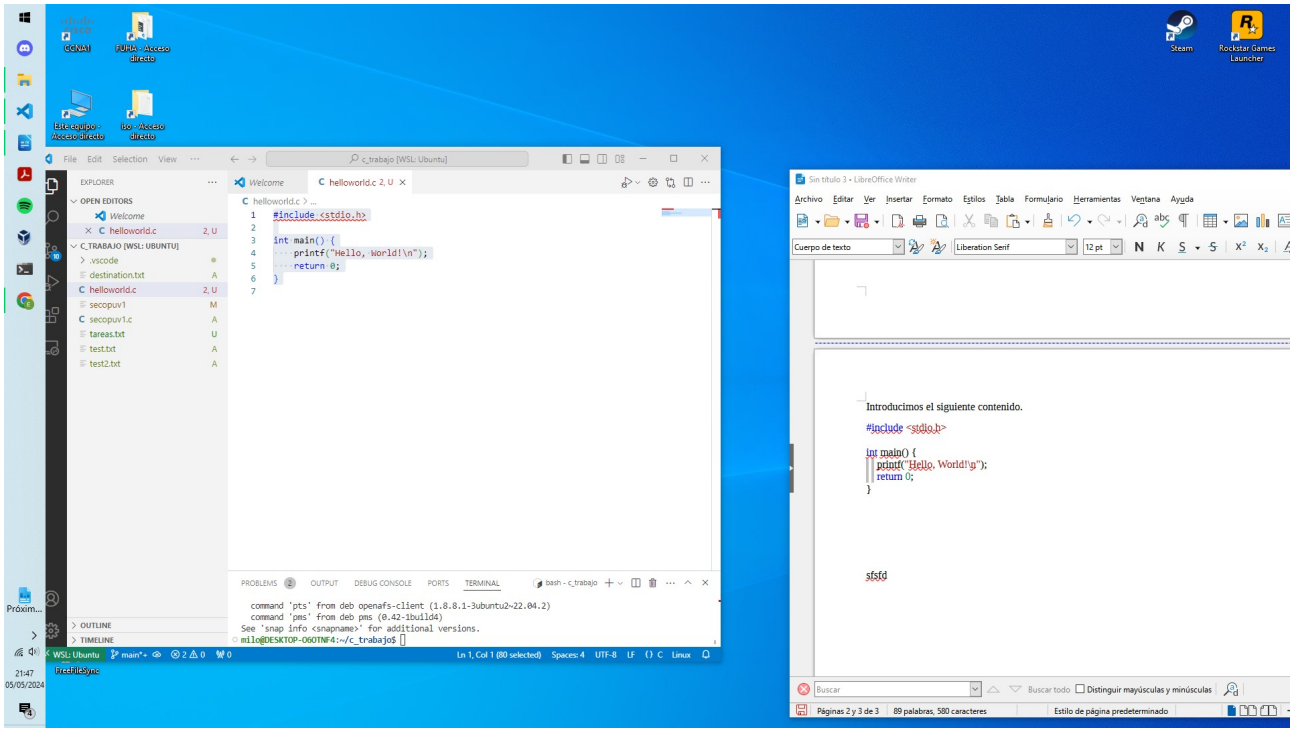
Creación primer programa

Creamos un archivo llamado helloworld.c



Introducimos el siguiente contenido.

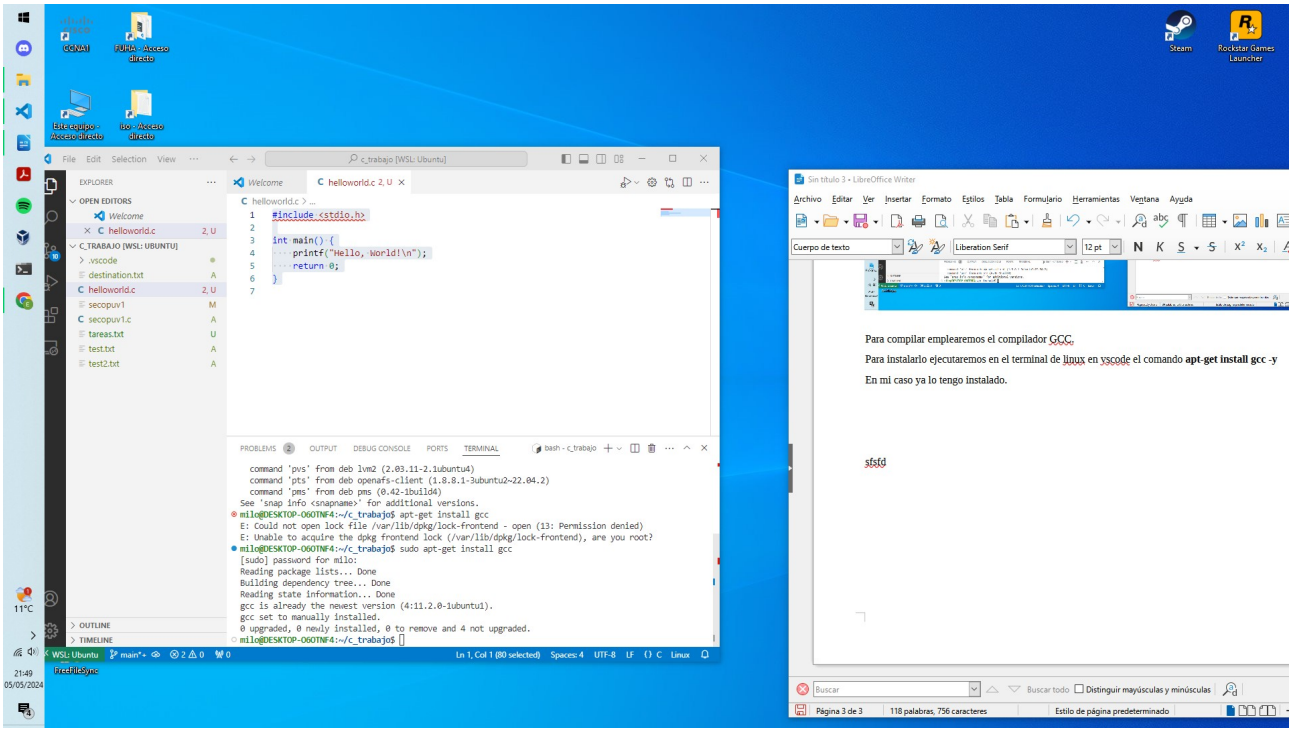
```
#include <stdio.h>

int main() {
    printf("Hello, World!\n");
    return 0;
}
```



Para compilar emplearemos el compilador GCC.

Para instalarlo ejecutaremos en el terminal de linux en vscode el comando **apt-get install gcc -y**

En mi caso ya lo tengo instalado.



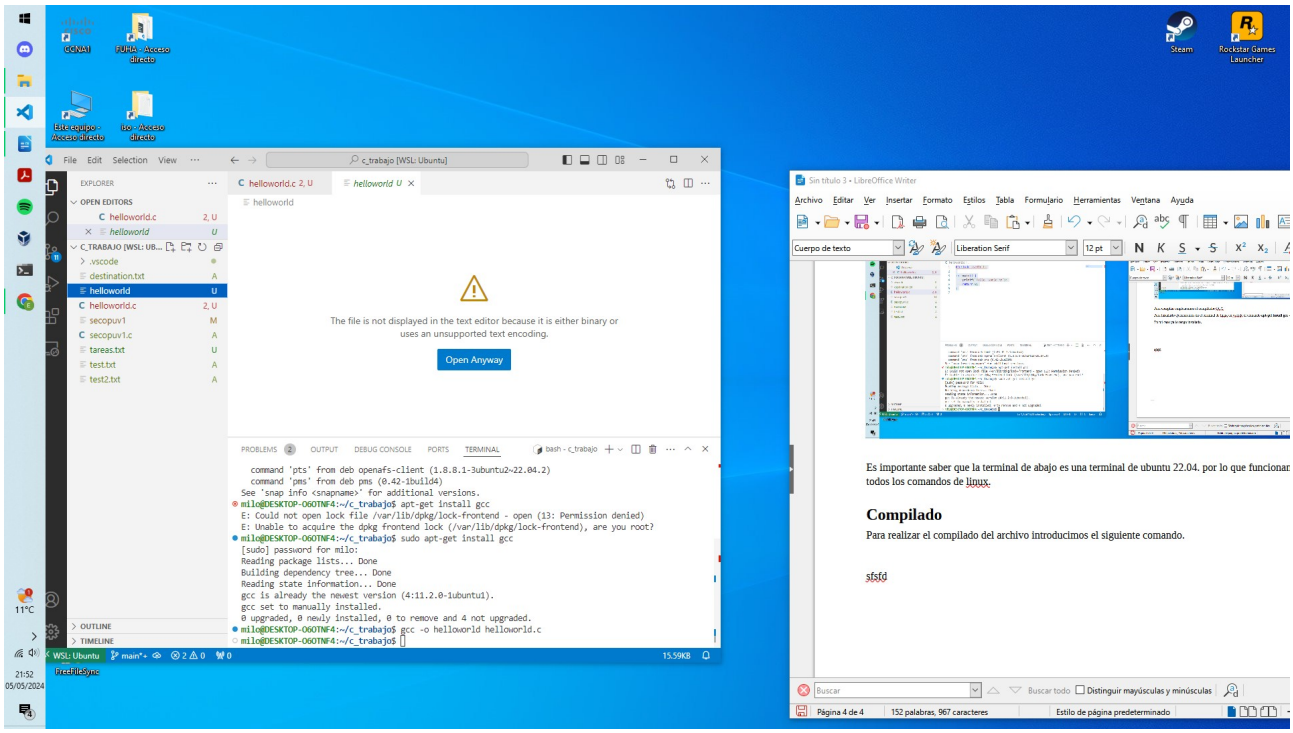
Es importante saber que la terminal de abajo es una terminal de ubuntu 22.04. por lo que funcionan todos los comandos de linux.

Compilado

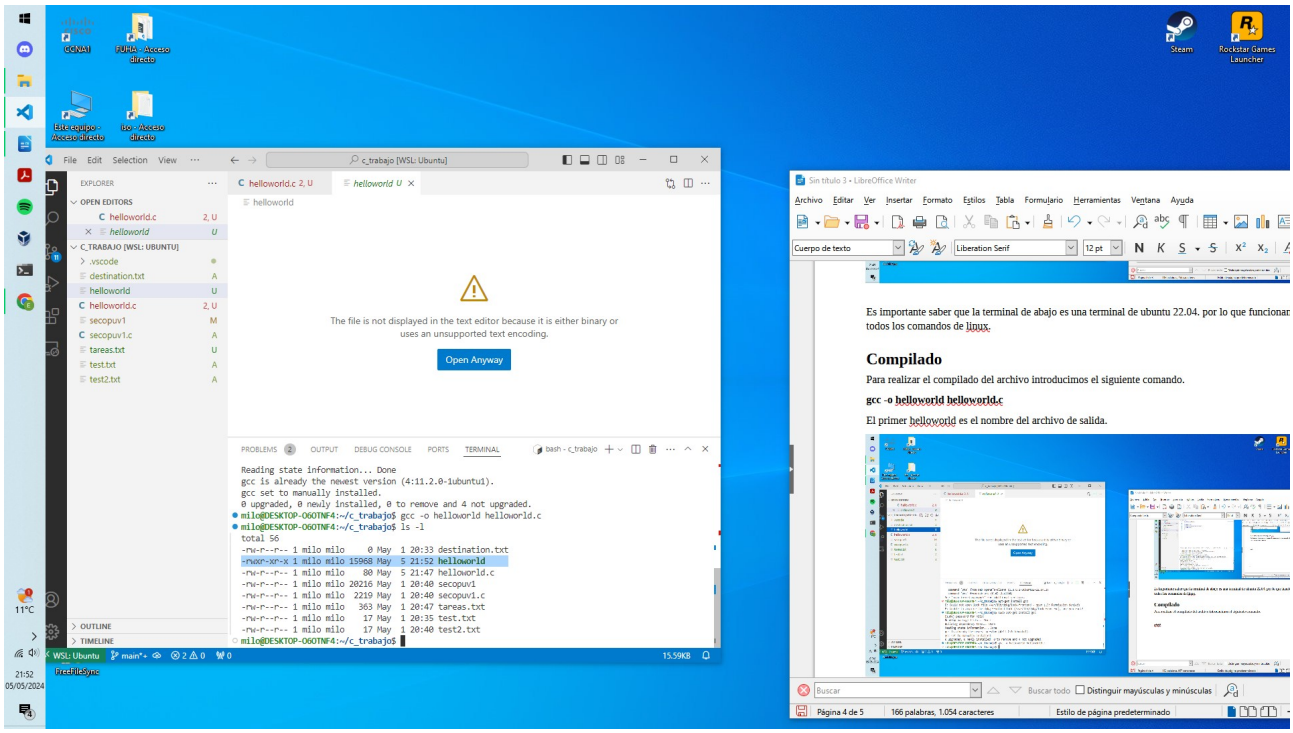
Para realizar el compilado del archivo introducimos el siguiente comando.

gcc -o helloworld helloworld.c

El primer helloworld es el nombre del archivo de salida.

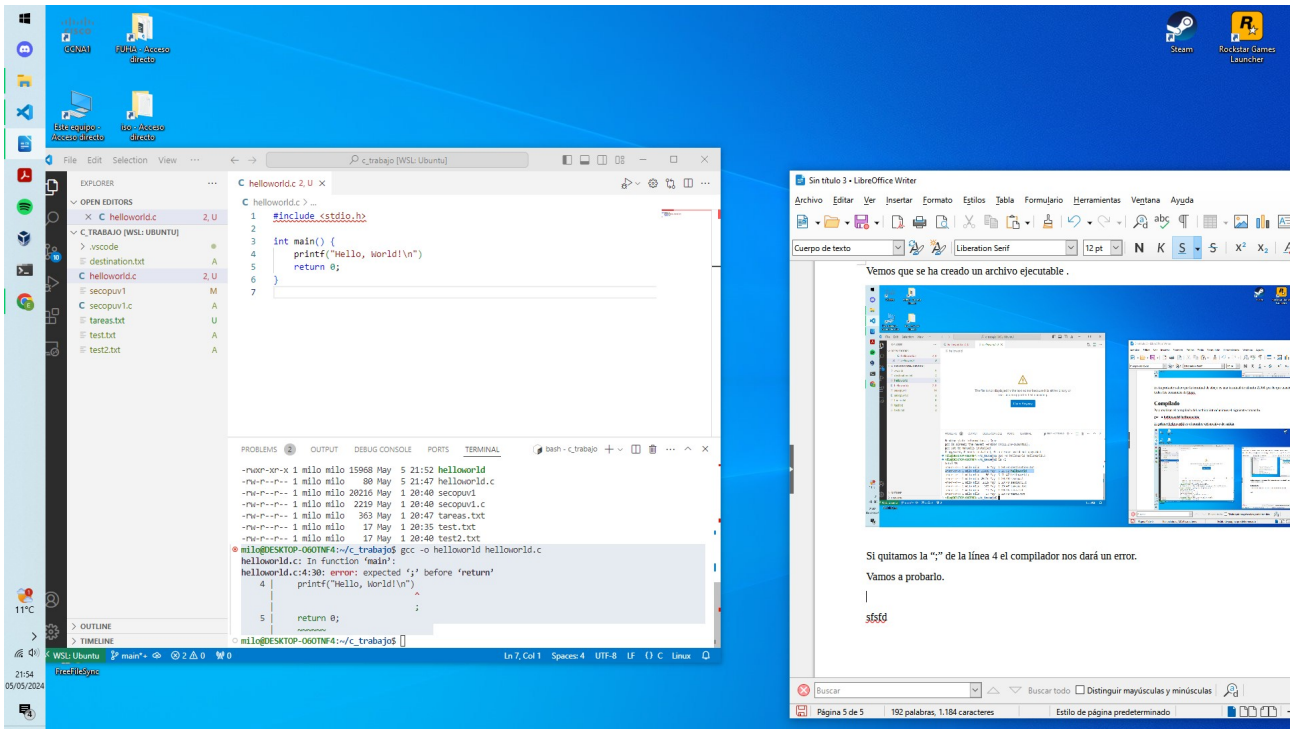


Vemos que se ha creado un archivo ejecutable .

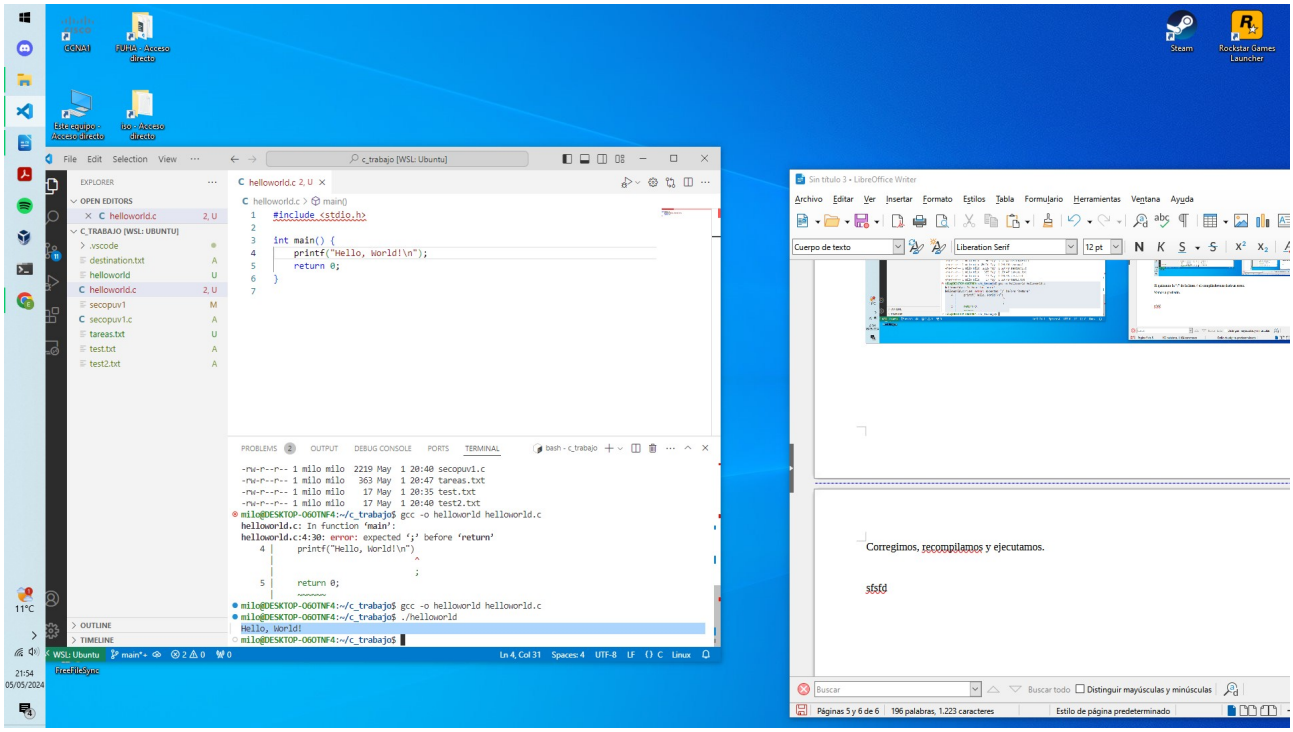


Si quitamos la “;” de la línea 4 el compilador nos dará un error.

Vamos a probarlo.


helloworld.c
#include <stdio.h>
int main() {
printf("Hello, World!\n")
return 0;
}
milo@DESKTOP-O6OTNF4:~/c_trabajo$ gcc -o helloworld helloworld.c
helloworld.c: In function 'main':
helloworld.c:4:30: error: expected ';' before 'return'
printf("Hello, World!\n")
return 0;
milo@DESKTOP-O6OTNF4:~/c_trabajo$
Sin título 3 - LibreOffice Writer
Vemos que se ha creado un archivo ejecutable .
Si quitamos la ";" de la línea 4 el compilador nos dará un error.
Vamos a probarlo.
sfsfd

Corregimos, recompilamos y ejecutamos.



Segundo programa

```
#include <stdio.h>
#include <sys/stat.h>
#include <fcntl.h>


void copiado(char *source_file_path, char *destination_file_path) {
    FILE *source_file;
    FILE *destination_file;
    char filebuffer[100];
    size_t file1;

    source_file = fopen(source_file_path, "rb"); // Abre el archivo source en binary mode
    destination_file = fopen(destination_file_path, "wb"); // Abre o crea el archivo destination para
escritura.
    if (source_file == NULL || destination_file == NULL) {
        perror("Error opening files");
        printf("error archivo 1");
        return;
    }

    // Copia elementos al buffer de tamaño 100Bytes
    file1 = fread(filebuffer, sizeof(char), 100, source_file);
    // Dump el contenido of the buffer al archivo de destino.
    fwrite(filebuffer, sizeof(char), file1, destination_file);

    //Cerrado de los archivos antes de salir de la funcion.
    //Debe realizarse antes de salir puesto que sino los pointers a memoria no estan declarados.
    fclose(source_file);
    fclose(destination_file);
}


int main () {
    struct stat buffer1;
    struct stat buffer2;
    int status1;
    int status2;

    status1 = stat("./test.txt", &buffer1);
    status2 = stat("./test2.txt", &buffer2);

//Si el programa devuelve el código de error -1 nos imprime el código de error.
    if (status1 == 0 || status2 == 0) {
        printf("========Archivo 1========\n");
```

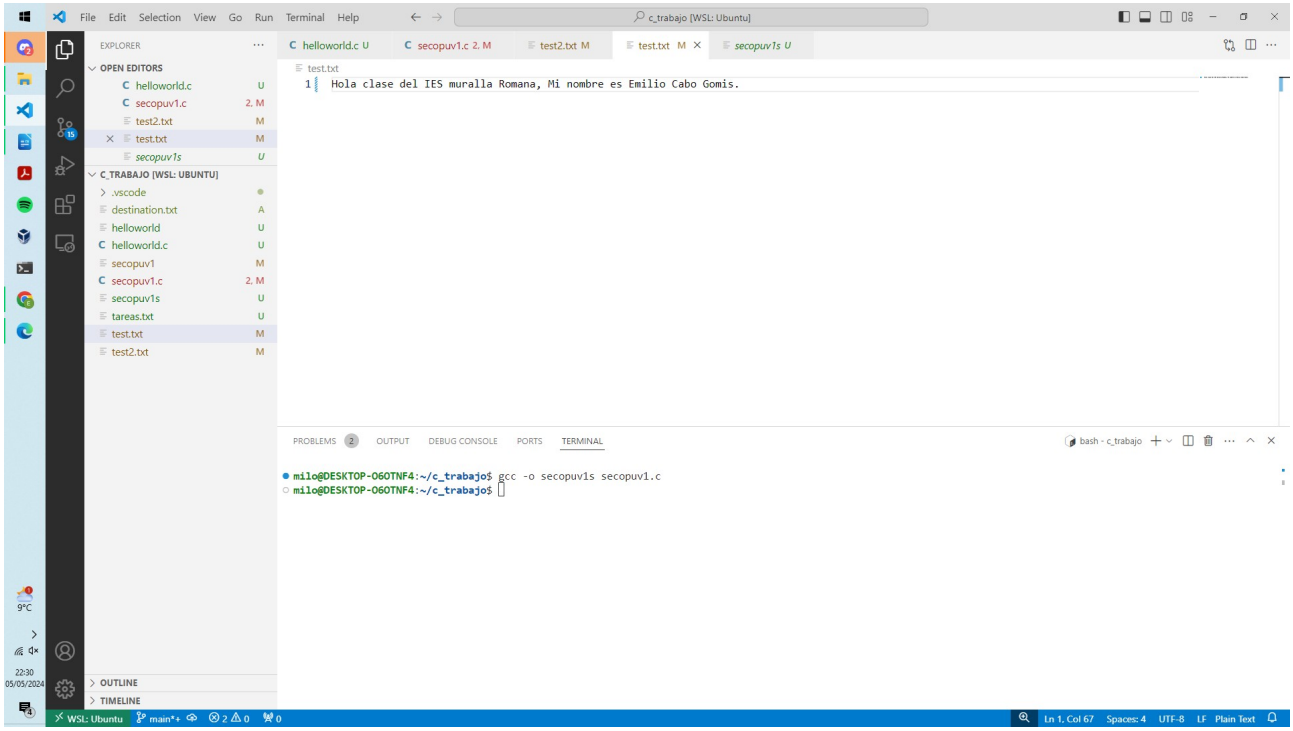
```
        printf("File size is : %ld\n", buffer1.st_size); //bytes es una palabra normal, es el %sd que imprime el contenido después de la ,
        printf("File time of last modif is : %ld\n", buffer1.st_mtime);
        printf("========Archivo 2========\n");
        printf("File size is : %ld\n", buffer2.st_size); //bytes es una palabra normal, es el %sd que imprime el contenido después de la ,
        printf("File time of last modif is : %ld\n", buffer2.st_mtime);
    }

    else {
        perror("stat");
    }

    if (buffer1.st_mtime == buffer2.st_mtime) {
        printf("The size of the files is the same.\n");
    } else if (buffer1.st_mtime > buffer2.st_mtime) {
        copiado("./test.txt", "./test2.txt");
    } else {
        copiado("./test2.txt", "./test.txt");
    }

    return 0;
}
```

Este es un archivo que lee las stats de dos archivos **test.txt** y **test2.txt** y las imprime en la consola.

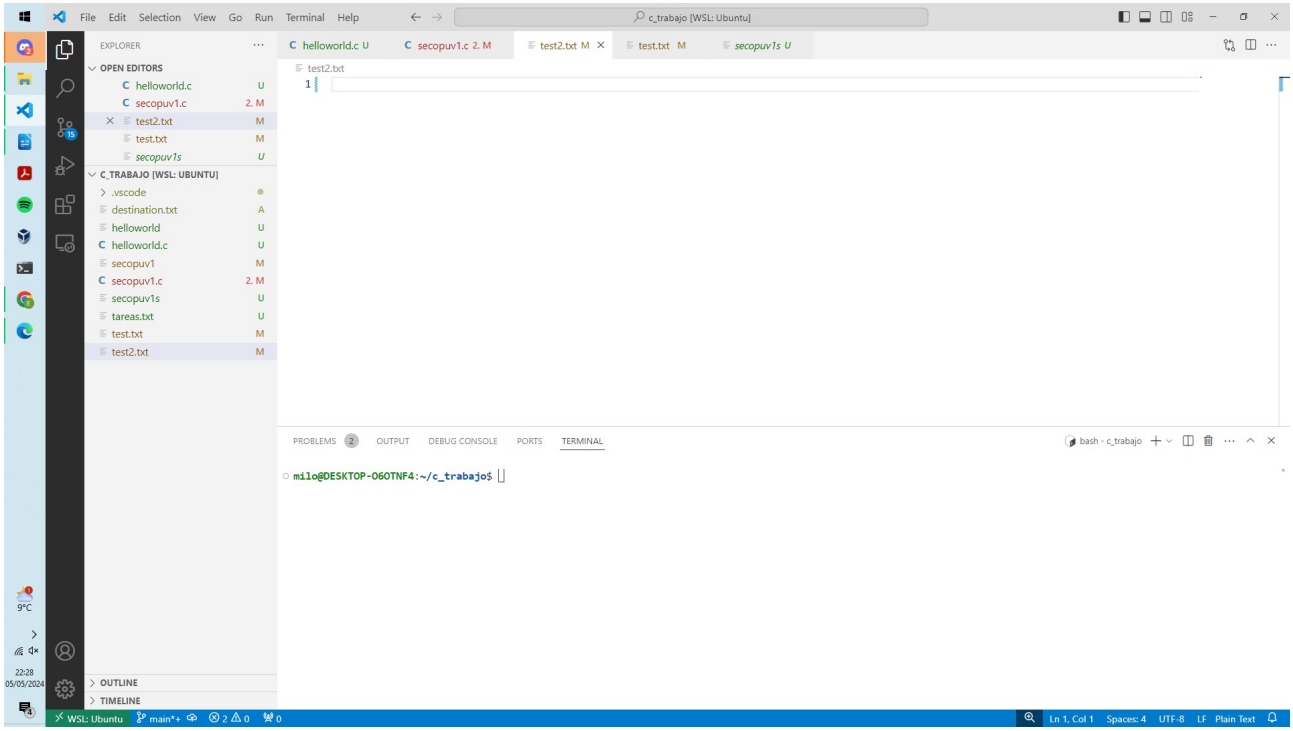
Crea un buffer de memoria de 100B y guarda el contenido del archivo mas nuevo y lo vuelca en el archivo mas antiguo.

Debido a que en C tenemos que manejar la memoria nosotros mismos

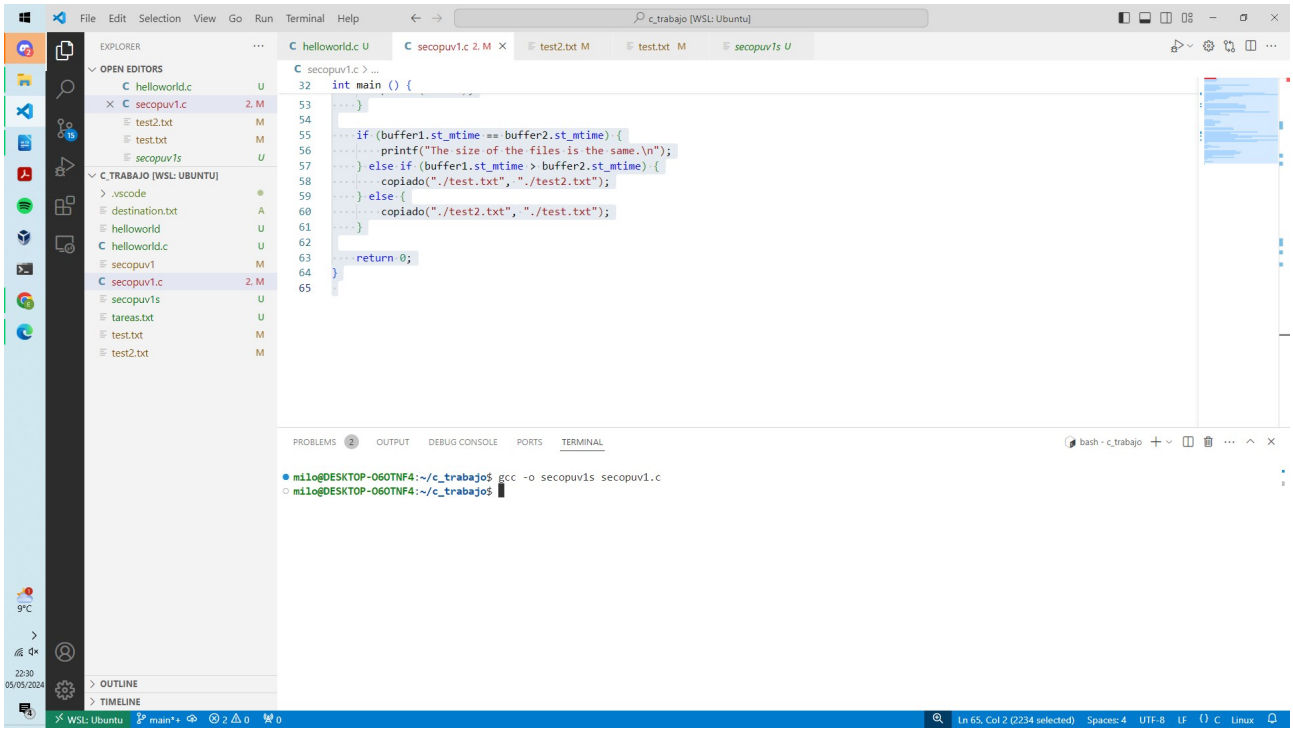
Este es el contenido del primer archivo **test1.txt**



Este es el contenido del segundo archivo.



Compilamos el programa.



Este es el output del programa.

========Archivo 1========

File size is : 66

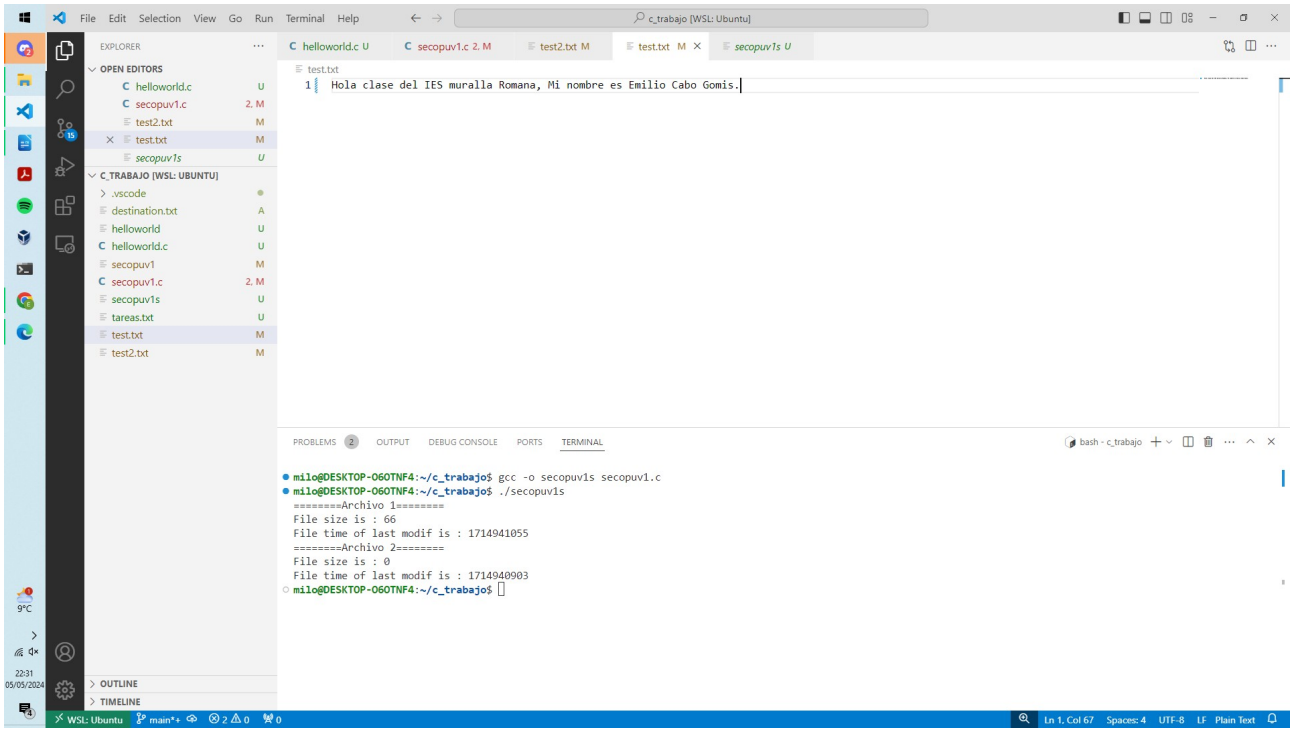
File time of last modif is : 1714941055

========Archivo 2========

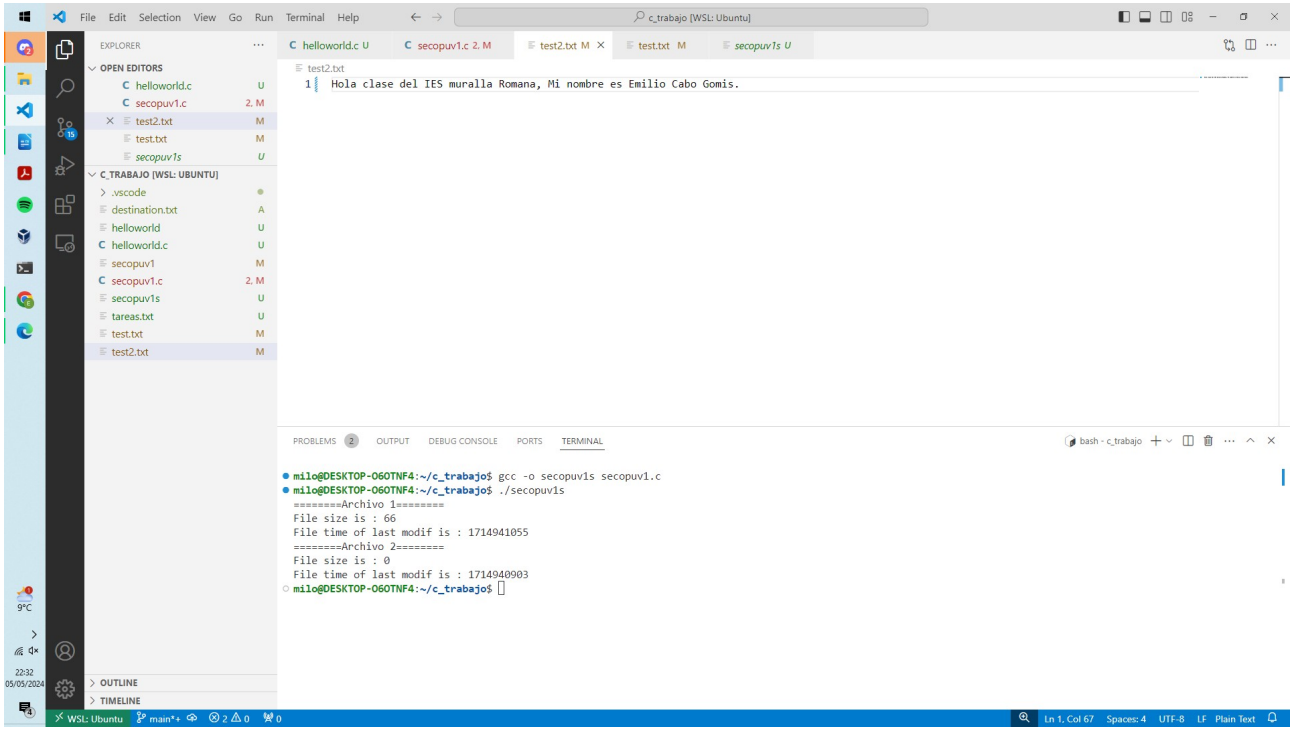
File size is : 0

File time of last modif is : 1714940903

Lo que debería pasar es que el archivo 1 se copiara al archivo 2.



Comprobamos



Como podemos ver el primer archivo tiene un tamaño de 66 Bytes por lo que hemos usado 66% del buffer de memoria que asignamos.

FIN DEL EJERCICIO