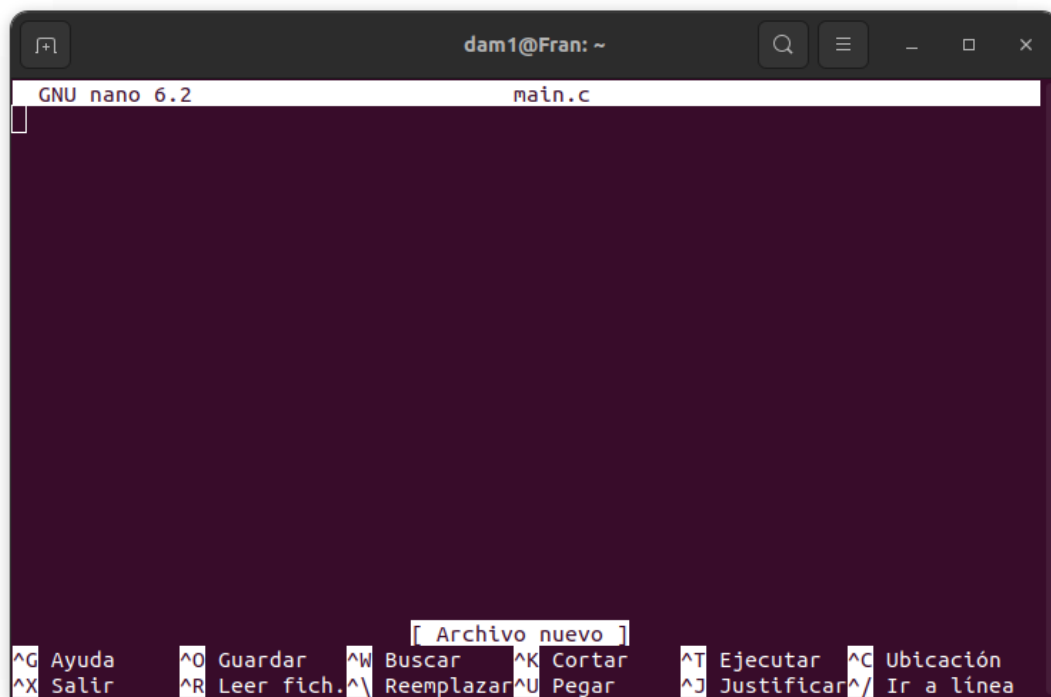


1. Crear el archivo fuente en C:

Aquí hemos creado un archivo que contiene el código fuente en C, mediante el editor de texto nano del terminal. Hemos creado el archivo con el siguiente código, lo hemos guardado y hemos cerrado el editor de texto.

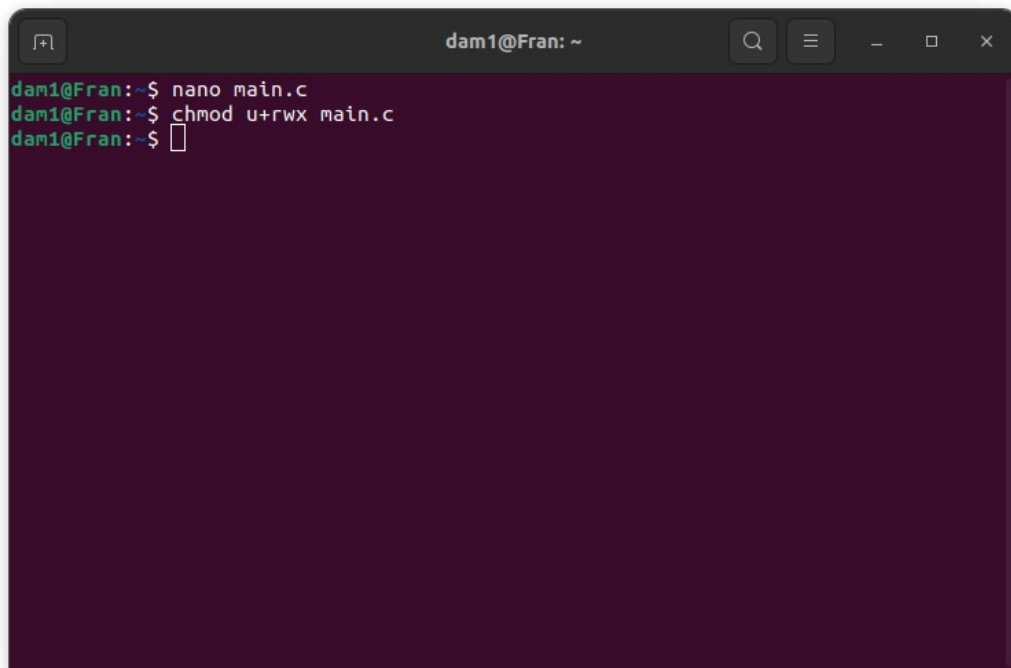
```
#include <stdio.h>
#define MESSAGE "¡Hola estudiantes de DAM!\n"

//los comentarios serán borrados en el preprocesado
//Aquí comienza el programa en sí
int main() {
    printf(MESSAGE);
    return 0;
}
//Aquí finaliza el programa
```



2. Cambiar los permisos del archivo para que sea ejecutable

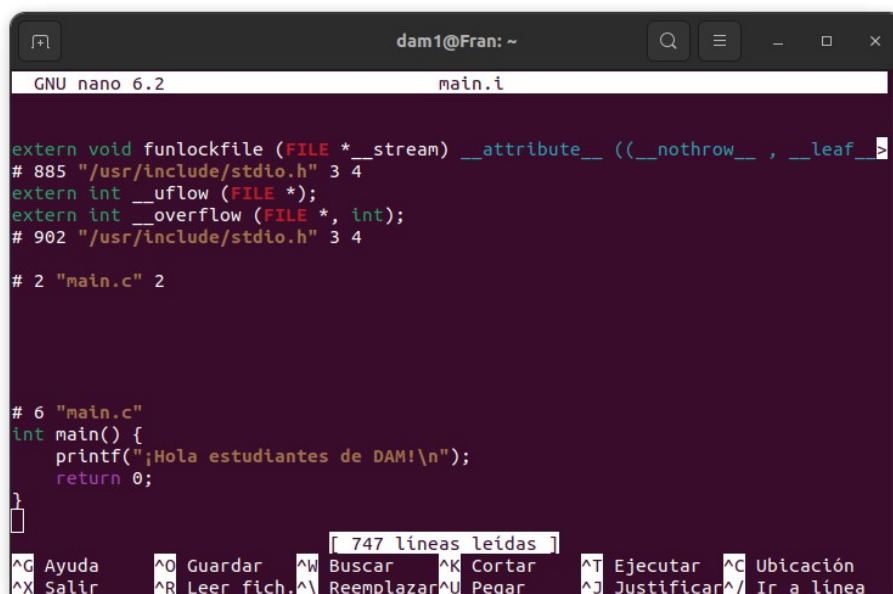
En este paso hemos cambiado los permisos de ejecución mediante el siguiente comando: `chmod u+rx main.c`



```
dam1@Fran: ~  
dam1@Fran:~$ nano main.c  
dam1@Fran:~$ chmod u+rx main.c  
dam1@Fran:~$
```

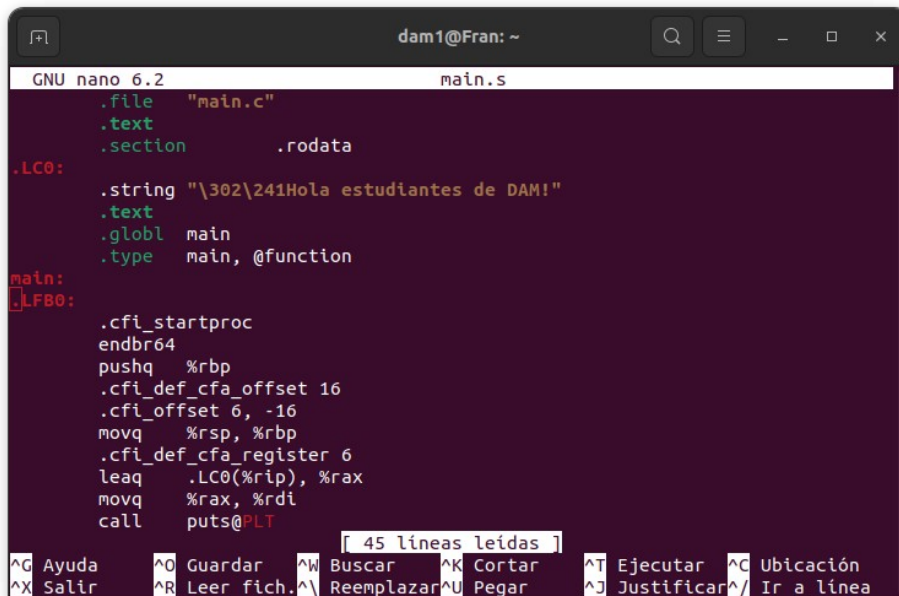
3. Preprocesado

En este paso vamos a adaptar el código fuente eliminando los comentarios y revisando posibles errores, generando el archivo **main.i** cuyo contenido es el código ya preprocesado.



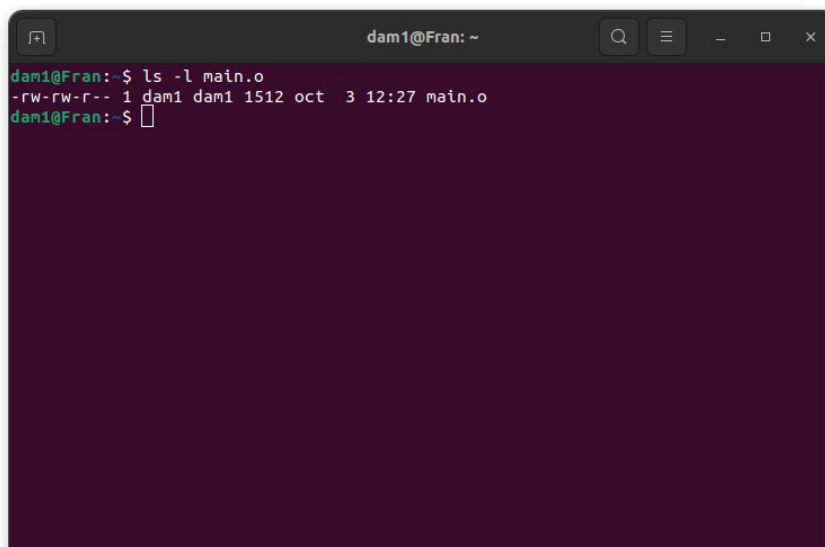
```
GNU nano 6.2 main.i  
  
extern void funlockfile (FILE *__stream) __attribute__((__nothrow__ , __leaf__  
# 885 "/usr/include/stdio.h" 3 4  
extern int __uflow (FILE *);  
extern int __overflow (FILE *, int);  
# 902 "/usr/include/stdio.h" 3 4  
  
# 2 "main.c" 2  
  
# 6 "main.c"  
int main() {  
    printf("¡Hola estudiantes de DAM!\n");  
    return 0;  
}  
  
[ 747 líneas leídas ]  
^G Ayuda ^O Guardar ^W Buscar ^K Cortar ^T Ejecutar ^C Ubicación  
^X Salir ^R Leer fich. ^_ Reemplazar ^U Pegar ^J Justificar ^_ Ir a línea
```

4.Compilación: En este paso vamos a convertir el código procesado en código ensamblador, donde empieza a haber dependencia con el SO y el Software. Aquí generaremos el archivo main.s



```
dam1@Fran: ~
GNU nano 6.2 main.s
.file "main.c"
.text
.section .rodata
.LC0:
.string "\302\241Hola estudiantes de DAM!"
.text
.globl main
.type main, @function
main:
.LFB0:
.cfi_startproc
endbr64
pushq %rbp
.cfi_def_cfa_offset 16
.cfi_offset 6, -16
movq %rsp, %rbp
.cfi_def_cfa_register 6
leaq .LC0(%rip), %rax
movq %rax, %rdi
call puts@PLT
[ 45 líneas leídas ]
^G Ayuda ^O Guardar ^W Buscar ^K Cortar ^T Ejecutar ^C Ubicación
^X Salir ^R Leer fich. ^_ Reemplazar ^U Pegar ^J Justificar ^_/ Ir a línea
```

5.Ensamblado: Aqui se va a traducir el código ensamblador a un archivo objeto(binario intermedio), y vamos a generar el archivo **main.o**.



```
dam1@Fran: ~
dam1@Fran:~$ ls -l main.o
-rw-rw-r-- 1 dam1 dam1 1512 oct  3 12:27 main.o
dam1@Fran:~$
```

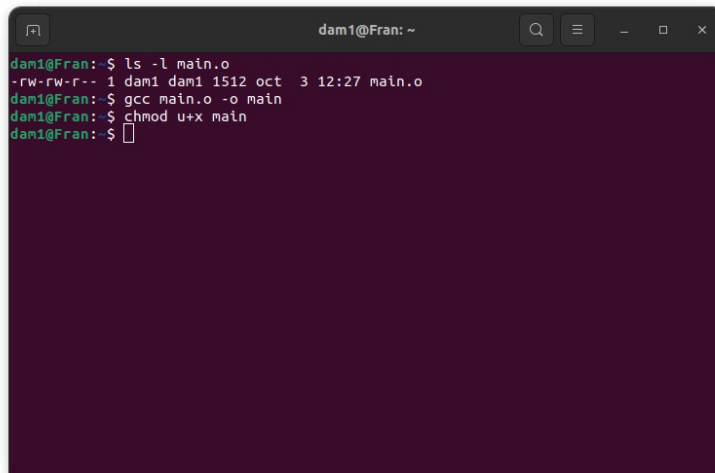
Aquí podemos observar que el código no está en binario por que linux no lo ha sabido interpretar

[illegible]

para poder ver el código en binario debemos introducir el siguiente código en la terminal: **xxd -b main**

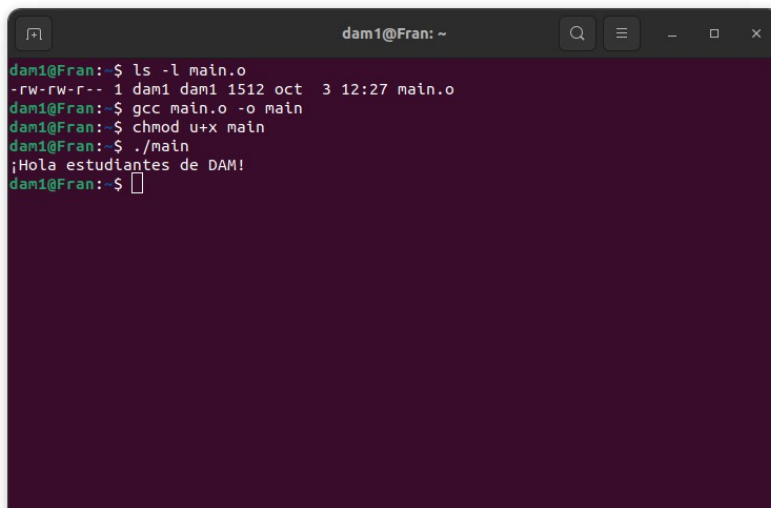
[illegible]

6.Enlazado: Este es el último paso antes de la ejecución, donde enlazaremos las librerías

A terminal window titled 'dam1@Fran: ~' with search, menu, and window control icons. It shows the following commands and output:

```
dam1@Fran:~$ ls -l main.o
-rw-rw-r-- 1 dam1 dam1 1512 oct  3 12:27 main.o
dam1@Fran:~$ gcc main.o -o main
dam1@Fran:~$ chmod u+x main
dam1@Fran:~$
```

7.Ejecutar el programa: En este paso ejecutaremos el programa a través del siguiente comando: `./main`

A terminal window titled 'dam1@Fran: ~' with search, menu, and window control icons. It shows the following commands and output:

```
dam1@Fran:~$ ls -l main.o
-rw-rw-r-- 1 dam1 dam1 1512 oct  3 12:27 main.o
dam1@Fran:~$ gcc main.o -o main
dam1@Fran:~$ chmod u+x main
dam1@Fran:~$ ./main
¡Hola estudiantes de DAM!
dam1@Fran:~$
```