

Sumario

1.Descripción de la tarea:.....	2
2.Instrucciones paso a paso:.....	2
2.1.Crear el archivo fuente en C.....	2
2.2.Cambiar los permisos del archivo para que sea ejecutable.....	3
2.3.Preprocesado.....	3
2.4.Compilación.....	5
2.5.Ensamblado.....	5
2.6.Enlazado.....	6
2.7.Ejecutar el programa.....	6

ED

Entornos de Desarrollo (ED)

Unidad de Trabajo: UT1

Tarea 0103: Análisis de la traducción de un programa en C

1.Descripción de la tarea:

En esta tarea, aprenderás a analizar y documentar el proceso de compilación de un programa escrito en C, desde la creación del archivo hasta la ejecución del programa final. Este proceso incluye varias fases: **preprocesado, compilación, ensamblado** y **enlazado**. A lo largo de la tarea, te guiaremos paso a paso en la creación y ejecución de un programa C en un entorno Linux, usando la terminal.

2.Instrucciones paso a paso:

2.1.Crear el archivo fuente en C

Lo primero que debes hacer es crear un archivo que contenga el código fuente en C. Para ello, utilizaremos el editor de texto nano, que puedes ejecutar desde la terminal. Sigue estos pasos:

- Abre la terminal de Linux.

- Para crear un archivo llamado `main.c`, escribe el siguiente comando y presiona **Enter**:

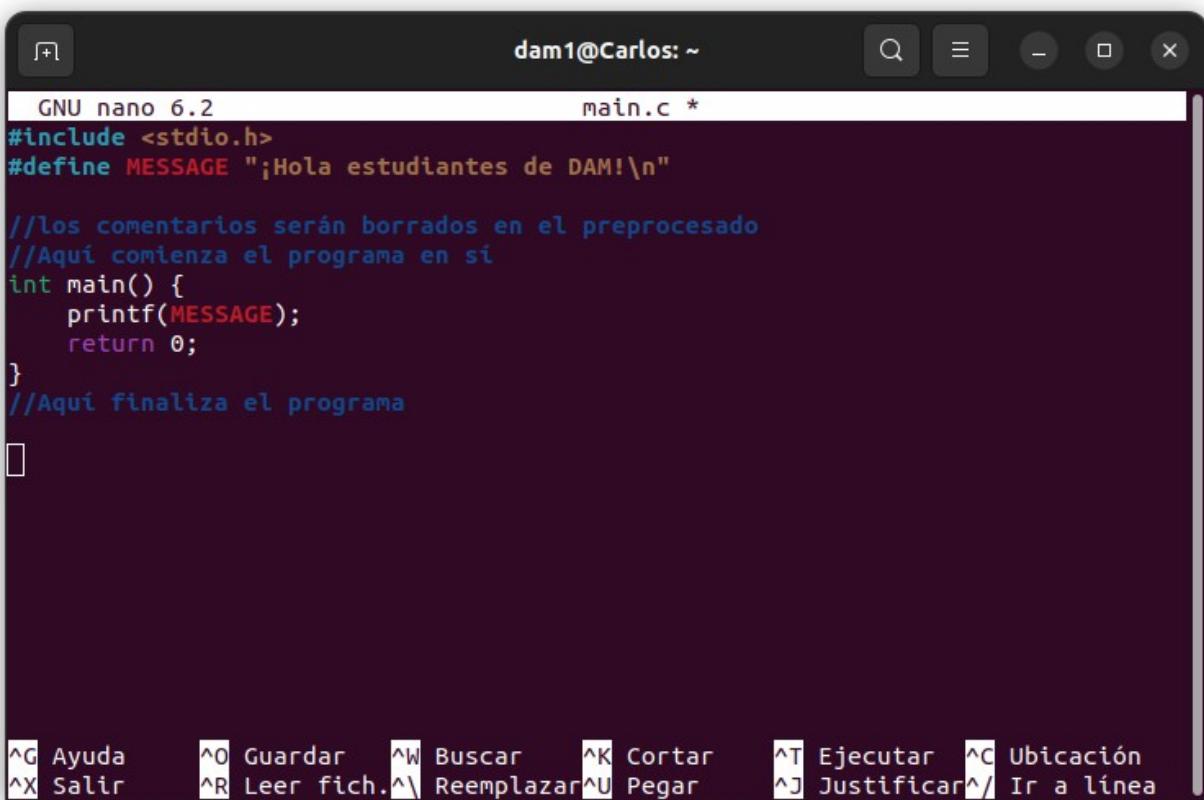
```
nano main.c
```

- Dentro del editor de texto `nano`, escribe el siguiente código:

```
#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
```

- Para guardar el archivo y salir de `nano`, presiona **Ctrl + O** (luego Enter para confirmar el nombre del archivo), y después **Ctrl + X** para cerrar el editor.



The screenshot shows a terminal window titled "dam1@Carlos: ~". The window contains the following text:

```
GNU nano 6.2                               main.c *
#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
```

At the bottom of the terminal, there is a menu bar with various keyboard shortcuts:

^G Ayuda	^O Guardar	^W Buscar	^K Cortar	^T Ejecutar	^C Ubicación
^X Salir	^R Leer fich.	^V Reemplazar	^U Pegar	^J Justificar	^/ Ir a línea

Insera el contenido del `main.c` (Supongo que es un código con el lenguaje de programación .c que permite escribir un mensaje y que salga por la pantalla).

The screenshot shows a code editor window with the following details:

- File menu: Abrir (Open), Guardar (Save).
- Title bar: main.c, ~/Descargas.
- Code content:

```
1 #include <stdio.h>
2 #define MESSAGE ";Hola estudiantes de DAM!\n"
3
4 //los comentarios serán borrados en el preprocesado
5 //Aqui comienza el programa en si
6 int main() {
7     printf(MESSAGE);
8     return 0;
9 }
10 //Aqui finaliza el programa
11
12
```
- Status bar: C (font size), Anchura del tabulador: 8 (Tab width), Ln 12, Col 1, INS (insert mode).

También lo hice desde el editor de texto de linux para comprobar varias maneras.

2.2.Cambiar los permisos del archivo para que sea ejecutable

En Linux, los archivos creados no son ejecutables de forma predeterminada. Antes de compilar y ejecutar el programa, deberás cambiar los permisos del archivo. En la terminal, escribe el siguiente comando para asegurarte de que tengas permisos de lectura y escritura:

```
chmod u+rwx main.c
```

The terminal window shows the following session:

```
dam1@Carlos: ~$ nano main.c
dam1@Carlos: ~$ chmod u+rwx main.c
```

Mediante este comando, lo que estoy haciendo es darle permisos al archivo para que me permita ejecutar el archivo que estoy creando, ya que predeterminadamente no es posible .

2.3.Preprocesado

Ahora que tienes el archivo fuente, el primer paso del proceso de compilación es el **preprocesado**. El preprocesador en C maneja directivas como `#include` y `#define`. Para realizar este paso, utiliza el siguiente comando en la terminal:

```
gcc -E main.c -o main.i
```

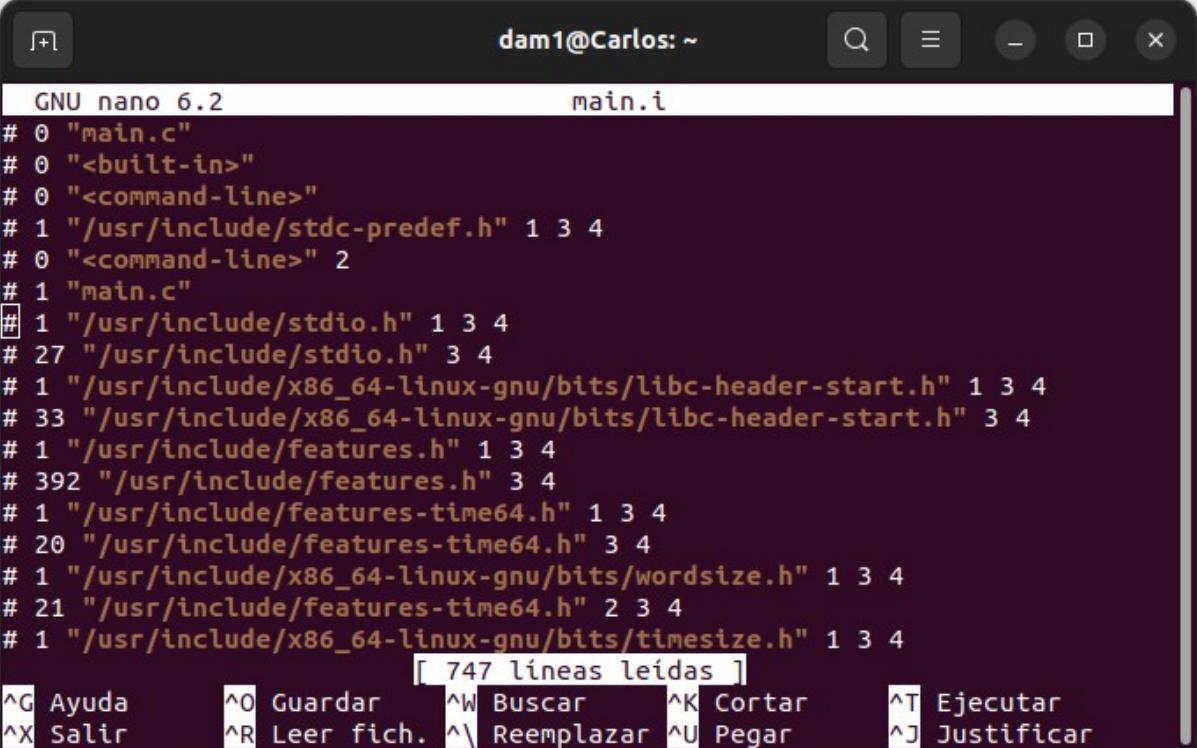
Este comando genera el archivo `main.i`, que contiene el código preprocesado. Puedes abrirla con el editor nano para ver cómo las macros y las cabeceras han sido reemplazadas:

```
nano main.i
```



```
dam1@Carlos:~$ nano main.c
dam1@Carlos:~$ chmod u+rwx main.c
dam1@Carlos:~$ gcc -E main.c -o main.i
dam1@Carlos:~$ nano main.i
dam1@Carlos:~$
```

Mediante el comando **gcc -E main.c -o main.i**, estoy generando un nuevo archivo main.i con el código del anterior archivo preprocesado antes de pasar a la compilación.



```
GNU nano 6.2                         main.i
# 0 "main.c"
# 0 "<built-in>"
# 0 "<command-line>"
# 1 "/usr/include/stdc-predef.h" 1 3 4
# 0 "<command-line>" 2
# 1 "main.c"
# 1 "/usr/include/stdio.h" 1 3 4
# 27 "/usr/include/stdio.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/libc-header-start.h" 1 3 4
# 33 "/usr/include/x86_64-linux-gnu/bits/libc-header-start.h" 3 4
# 1 "/usr/include/features.h" 1 3 4
# 392 "/usr/include/features.h" 3 4
# 1 "/usr/include/features-time64.h" 1 3 4
# 20 "/usr/include/features-time64.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/wordsize.h" 1 3 4
# 21 "/usr/include/features-time64.h" 2 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/timesize.h" 1 3 4
[ 747 líneas leídas ]
^G Ayuda      ^O Guardar      ^W Buscar      ^K Cortar      ^T Ejecutar
^X Salir      ^R Leer fich.  ^\ Reemplazar  ^U Pegar      ^J Justificar
```

2.4.Compilación

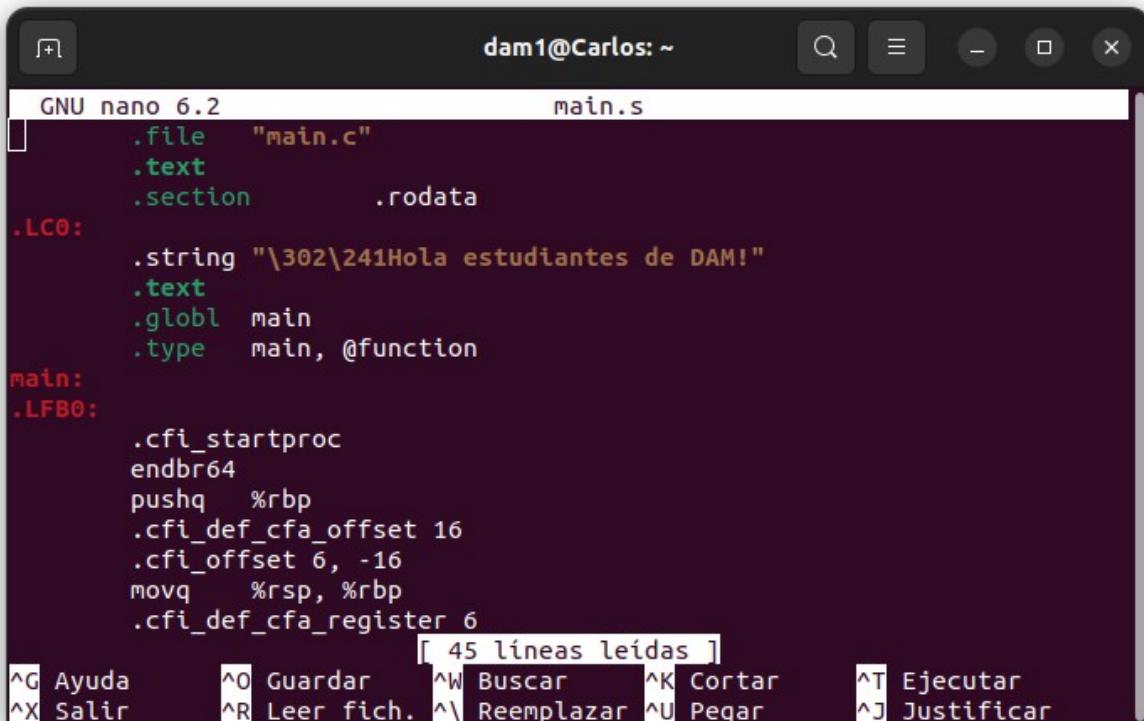
El siguiente paso es compilar el código preprocesado para convertirlo en código ensamblador. Ejecuta el siguiente comando para generar un archivo llamado `main.s`:

```
gcc -S main.i -o main.s
```

Abre el archivo ensamblador con `nano` para ver el resultado:

```
nano main.s
```

En este archivo podrás observar instrucciones en lenguaje ensamblador, que son más cercanas a las instrucciones que entiende el procesador.



The screenshot shows the nano text editor window with the file `main.s`. The assembly code is as follows:

```
GNU nano 6.2                         main.s
[ 45 líneas leídas ]
[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
```

The status bar at the bottom shows keyboard shortcuts for various operations like Ayuda (Help), Guardar (Save), Buscar (Search), Cortar (Cut), Ejecutar (Execute), Salir (Exit), Leer fich. (Read file), Reemplazar (Replace), Pegar (Paste), and Justificar (Justify).

En este paso, convierto el archivo `main.i` en un archivo con lenguaje ensamblador que esta contenido dentro de un nuevo archivo llamado `main.s`. Este lenguaje permite al ordenador comprender el contenido del archivo.

2.5.Ensamblado

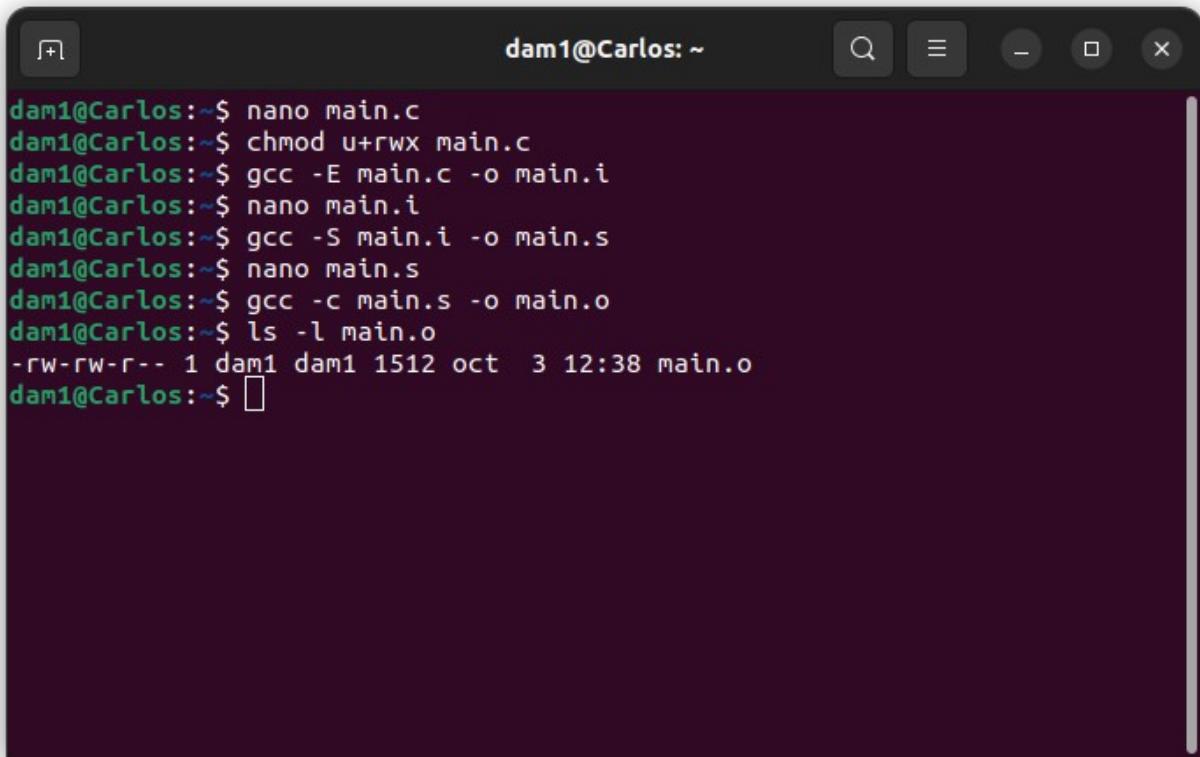
El código ensamblador ahora debe ser traducido a un archivo objeto (binario intermedio) que el sistema pueda manejar. Usa el siguiente comando para generar el archivo `main.o`:

```
gcc -c main.s -o main.o
```

El archivo `main.o` es un archivo objeto que contiene el código en formato binario. Aunque no es necesario abrir este archivo, puedes verificar que se ha generado correctamente utilizando el siguiente comando:

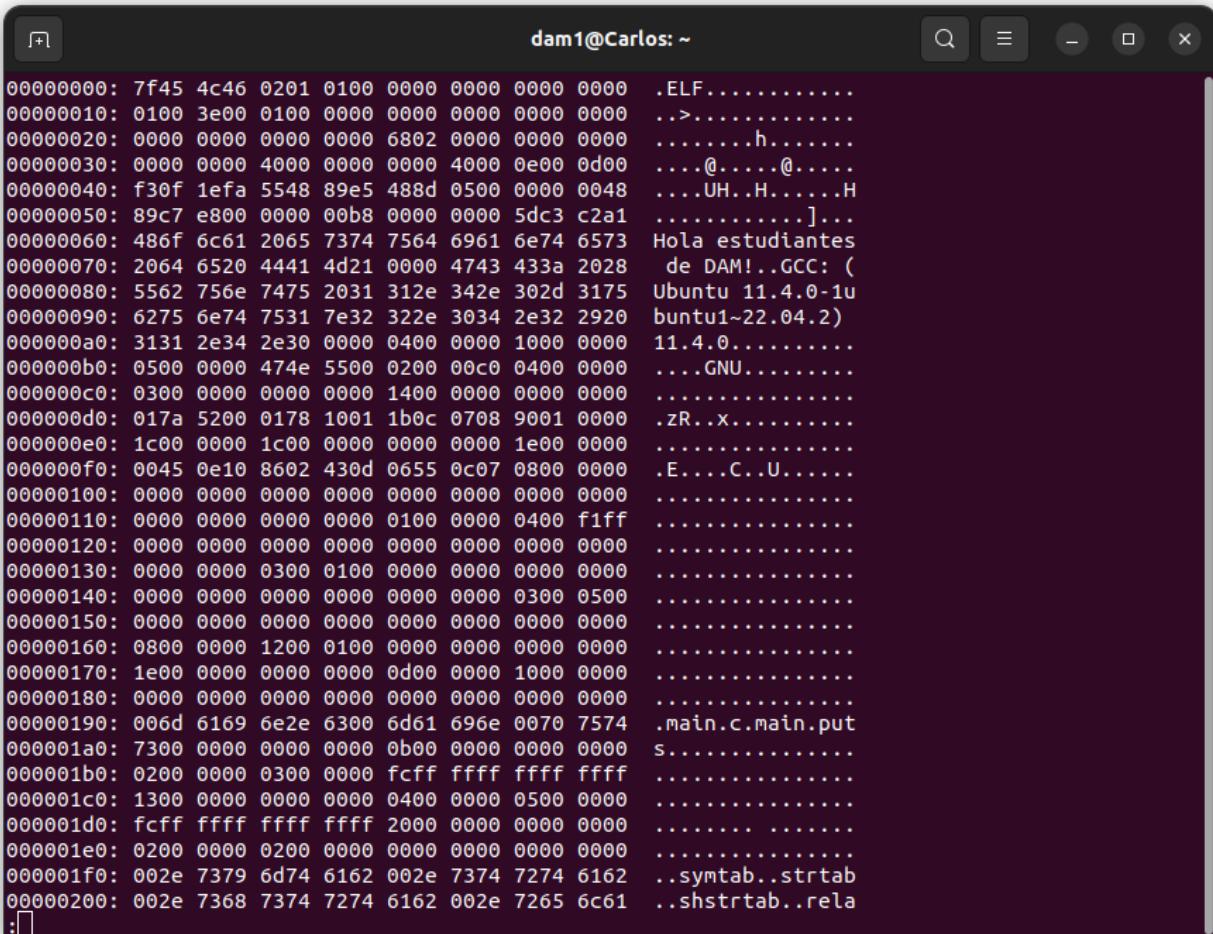
```
ls -l main.o
```

También puedes investigar cómo mostrar el contenido en binario.



```
dam1@Carlos:~$ nano main.c
dam1@Carlos:~$ chmod u+rwx main.c
dam1@Carlos:~$ gcc -E main.c -o main.i
dam1@Carlos:~$ nano main.i
dam1@Carlos:~$ gcc -S main.i -o main.s
dam1@Carlos:~$ nano main.s
dam1@Carlos:~$ gcc -c main.s -o main.o
dam1@Carlos:~$ ls -l main.o
-rw-rw-r-- 1 dam1 dam1 1512 oct  3 12:38 main.o
dam1@Carlos:~$
```

HEXADECIMAL (xxd main.o | less)



```
dam1@Carlos:~$ xxd main.o | less
00000000: 7f45 4c46 0201 0100 0000 0000 0000 0000 .ELF.....
00000010: 0100 3e00 0100 0000 0000 0000 0000 0000 ..>.....
00000020: 0000 0000 0000 0000 6802 0000 0000 0000 .....h.....
00000030: 0000 0000 4000 0000 0000 4000 0e00 0d00 ....@....@....
00000040: f30f 1efa 5548 89e5 488d 0500 0000 0048 ....UH..H....H
00000050: 89c7 e800 0000 00b8 0000 0000 5dc3 c2a1 .....]...
00000060: 486f 6c61 2065 7374 7564 6961 6e74 6573 Hola estudiantes
00000070: 2064 6520 4441 4d21 0000 4743 433a 2028 de DAM!..GCC: (
00000080: 5562 756e 7475 2031 312e 342e 302d 3175 Ubuntu 11.4.0-1u
00000090: 6275 6e74 7531 7e32 322e 3034 2e32 2920 buntu1~22.04.2)
000000a0: 3131 2e34 2e30 0000 0400 0000 1000 0000 11.4.0.....
000000b0: 0500 0000 474e 5500 0200 00c0 0400 0000 ....GNU.....
000000c0: 0300 0000 0000 0000 1400 0000 0000 0000 .....
000000d0: 017a 5200 0178 1001 1b0c 0708 9001 0000 .zR..x.....
000000e0: 1c00 0000 1c00 0000 0000 0000 1e00 0000 .....
000000f0: 0045 0e10 8602 430d 0655 0c07 0800 0000 .E....C..U...
00000100: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000110: 0000 0000 0000 0000 0100 0000 0400 f1ff .....
00000120: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00000130: 0000 0000 0300 0100 0000 0000 0000 0000 .....
00000140: 0000 0000 0000 0000 0000 0300 0500 ...
00000150: 0000 0000 0000 0000 0000 0000 0000 ...
00000160: 0800 0000 1200 0100 0000 0000 0000 ...
00000170: 1e00 0000 0000 0000 0d00 0000 1000 0000 ...
00000180: 0000 0000 0000 0000 0000 0000 0000 0000 ...
00000190: 006d 6169 6e2e 6300 6d61 696e 0070 7574 .main.c.main.put
000001a0: 7300 0000 0000 0000 0b00 0000 0000 0000 s.....
000001b0: 0200 0000 0300 0000 fcff ffff ffff ffff .....
000001c0: 1300 0000 0000 0000 0400 0000 0500 0000 .....
000001d0: fcff ffff ffff 2000 0000 0000 0000 ...
000001e0: 0200 0000 0200 0000 0000 0000 0000 ...
000001f0: 002e 7379 6d74 6162 002e 7374 7274 6162 ..symtab..strtab
00000200: 002e 7368 7374 7274 6162 002e 7265 6c61 ..shstrtab..rela
:
```

BINARIO (xxd -b main.o | less)

```
dam1@Carlos: ~
00000000: 01111111 01000101 01001100 01000110 00000010 00000001 .ELF..
00000006: 00000001 00000000 00000000 00000000 00000000 00000000 .....
0000000c: 00000000 00000000 00000000 00000000 00000001 00000000 .....
00000012: 00111110 00000000 00000001 00000000 00000000 00000000 >.....
00000018: 00000000 00000000 00000000 00000000 00000000 00000000 .....
0000001e: 00000000 00000000 00000000 00000000 00000000 00000000 .....
00000024: 00000000 00000000 00000000 00000000 01101000 00000010 ....h.
0000002a: 00000000 00000000 00000000 00000000 00000000 00000000 .....
00000030: 00000000 00000000 00000000 00000000 01000000 00000000 ....@.
00000036: 00000000 00000000 00000000 00000000 01000000 00000000 ....@.
0000003c: 00001110 00000000 00001101 00000000 11110011 00001111 .....
00000042: 00011110 11111010 01010101 01001000 10001001 11100101 ..UH..
00000048: 01001000 10001101 00000101 00000000 00000000 00000000 H.....
0000004e: 00000000 01001000 10001001 11000111 11101000 00000000 .H.....
00000054: 00000000 00000000 00000000 10111000 00000000 00000000 .....
0000005a: 00000000 00000000 01011101 11000011 11000010 10100001 ...]...
00000060: 01001000 01101111 01101100 01100001 00100000 01100101 Hola e
00000066: 01110011 01110100 01110101 01100100 01101001 01100001 studia
0000006c: 01101110 01110100 01100101 01110011 00100000 01100100 ntes d
00000072: 01100101 00100000 01000100 01000001 01001101 00100001 e DAM!
00000078: 00000000 00000000 01000111 01000011 01000011 00111010 ..GCC:
0000007e: 00100000 00101000 01010101 01100010 01110101 01101110 (Ubun
00000084: 01110100 01110101 00100000 00110001 00110001 00101110 tu 11.
0000008a: 00110100 00101110 00110000 00101101 00110001 01110101 4.0-1u
00000090: 01100010 01110101 01101110 01110100 01110101 00110001 buntu1
00000096: 01111110 00110010 00110010 00101110 00110000 00110100 ~22.04
0000009c: 00101110 00110010 00101001 00100000 00110001 00110001 .2) 11
000000a2: 00101110 00110100 00101110 00110000 00000000 00000000 .4.0..
000000a8: 00000100 00000000 00000000 00000000 00010000 00000000 .....
000000ae: 00000000 00000000 00000101 00000000 00000000 00000000 .....:
```

```
(xxd -b main.o | cut -d" " -f2-)
```

```
dam1@Carlos: ~
00000000 00000000 00000000 00000000 00000001 00000000 . . .
00111110 00000000 00000001 00000000 00000000 00000000 > . .
00000000 00000000 00000000 00000000 00000000 00000000 . . .
00000000 00000000 00000000 00000000 00000000 00000000 . . .
00000000 00000000 00000000 00000000 01101000 00000010 . . . h .
00000000 00000000 00000000 00000000 00000000 00000000 . . .
00000000 00000000 00000000 00000000 01000000 00000000 . . . @ .
00000000 00000000 00000000 00000000 01000000 00000000 . . . @ .
00001110 00000000 00001101 00000000 11110011 00001111 . . .
00011110 11111010 01010101 01001000 10001001 11100101 .. UH ..
01001000 10001101 00000101 00000000 00000000 00000000 H. . .
00000000 01001000 10001001 11000111 11101000 00000000 . H. . .
00000000 00000000 00000000 10111000 00000000 00000000 . . .
00000000 00000000 01011101 11000011 11000010 10100001 .. ] . .
01001000 01101111 01101100 01100001 00100000 01100101 Hola e
01110011 01110100 01110101 01100100 01101001 01100001 studia
01101110 01110100 01100101 01100011 00100000 01100100 ntes d
01100101 00100000 01000100 01000001 01001101 00100001 e DAM!
00000000 00000000 01000111 01000011 01000011 00111010 .. GCC:
00100000 00101000 01010101 01100010 01110101 01101110 (Ubun
01110100 01110101 00100000 00110001 00110001 00101110 tu 11.
00110100 00101110 00110000 00101101 00110001 01110101 4.0 -1u
01100010 01110101 01101110 01110100 01110101 00110001 buntu1
01111110 00110010 00110010 00101110 00110000 00110100 ~22.04
00101110 00110010 00101001 00100000 00110001 00110001 .2) 11
00101110 00110100 00101110 00110000 00000000 00000000 4.0
```

O (xxd -b main.o | cut -d" " -f2- | tr -d '\n'

2.6. Enlazado

El último paso es enlazar el archivo objeto con las bibliotecas necesarias para crear el ejecutable. Usa el siguiente comando:

```
gcc main.o -o main
```

Esto generará el ejecutable `main`. Sin embargo, si no puedes ejecutarlo directamente, deberás asegurarte de que tiene los permisos correctos. Cambia los permisos del ejecutable utilizando:

```
chmod u+x main
```

```
dam1@Carlos:~$ gcc main.o -o main
dam1@Carlos:~$ chmod u+x main
```

Por último, el archivo binario `main.o` lo paso o genero un ejecutable capaz de realizar la acción que necesito (en este caso, que muestre por la pantalla ¡Hola estudiantes de DAM!).

2.7. Ejecutar el programa

Una vez generado el ejecutable, puedes correr el programa usando el siguiente comando:
`./main`

Si todo ha salido bien, verás que el programa imprime el mensaje **Hello, World!** en la terminal.

```
dam1@Carlos:~$ nano main.c
dam1@Carlos:~$ chmod u+rwx main.c
dam1@Carlos:~$ gcc -E main.c -o main.i
dam1@Carlos:~$ nano main.i
dam1@Carlos:~$ gcc -S main.i -o main.s
dam1@Carlos:~$ nano main.s
dam1@Carlos:~$ gcc -c main.s -o main.o
dam1@Carlos:~$ ls -l main.o
-rw-rw-r-- 1 dam1 dam1 1512 oct  3 12:38 main.o
dam1@Carlos:~$ gcc main.o -o main
dam1@Carlos:~$ chmod u+x main
dam1@Carlos:~$ ./main
¡Hola estudiantes de DAM!
dam1@Carlos:~$
```