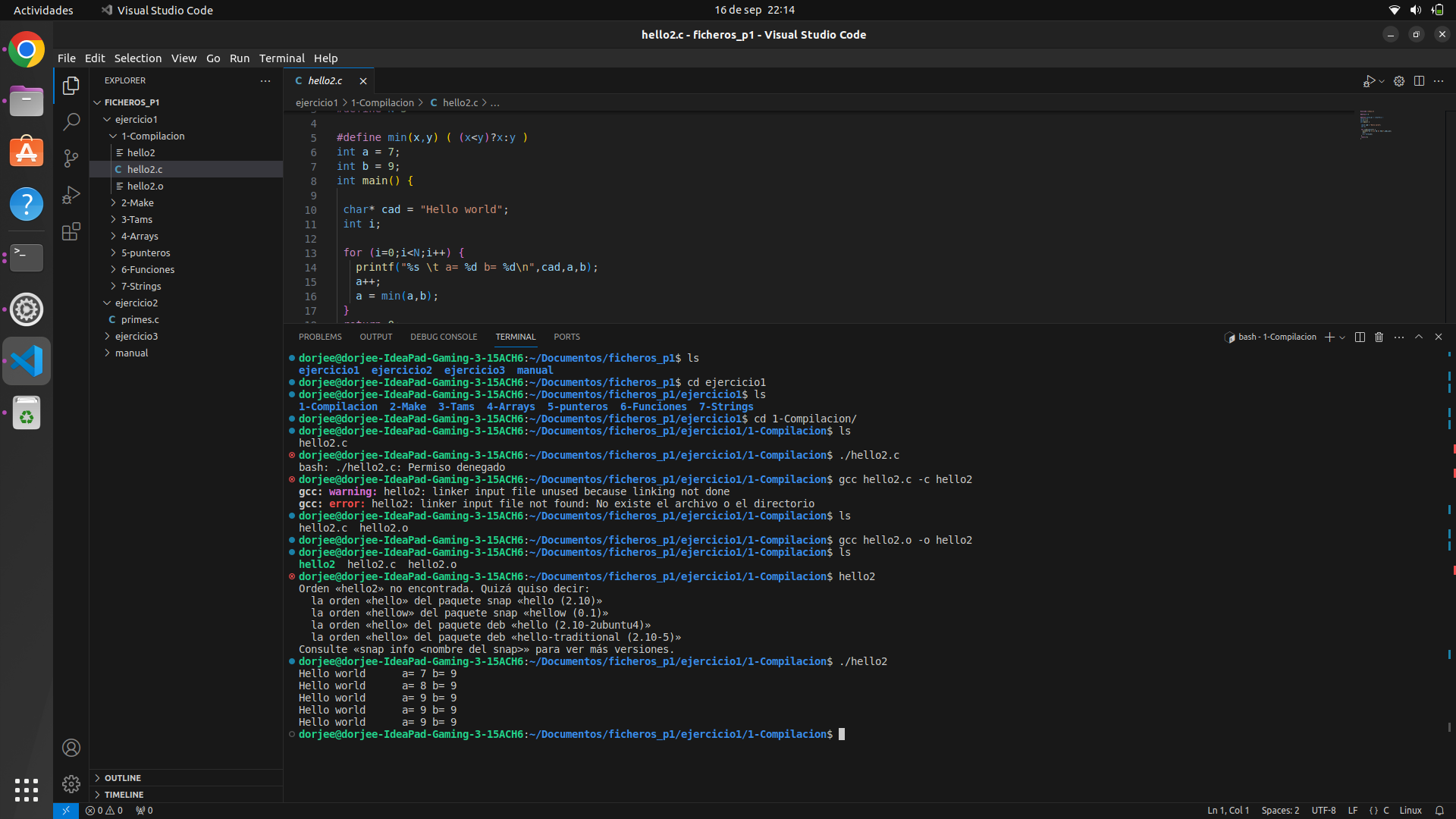
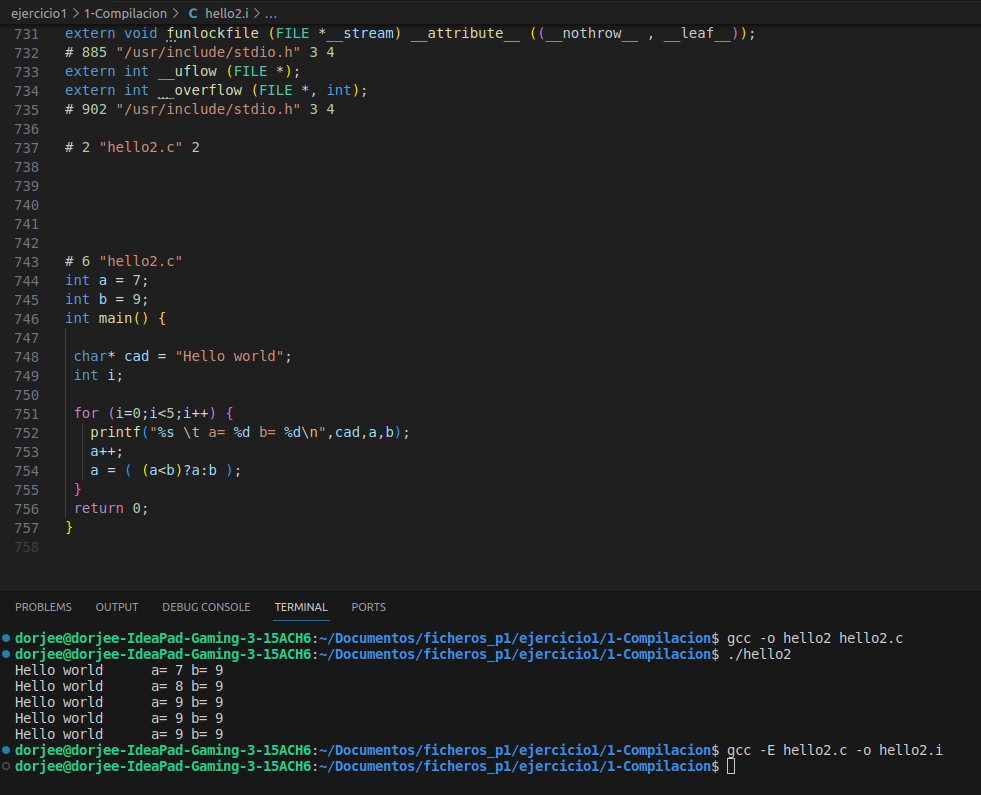
Práctica 1 Sistemas Operativos



### 1. Compilación

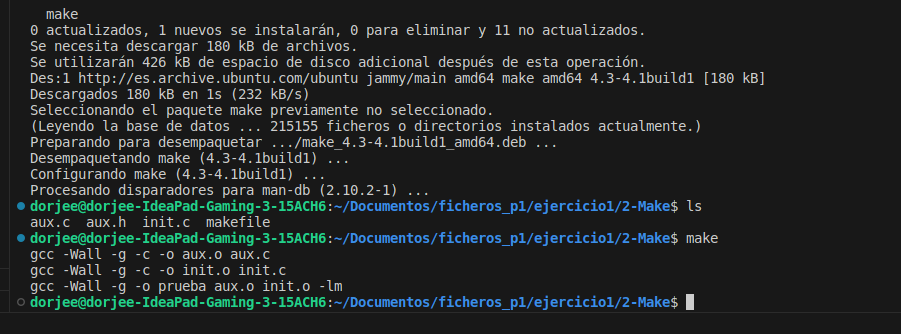
* Compila el código del ejercicio y ejecútalo
* Obtén la salida de la etapa de pre-procesado (opción -E o la opción --save-temps para obtener la salida de todas las etapas intermedias) y en un fichero *hello2.i*
* ¿Qué ha ocurrido con la “llamada a min()” en *hello2.i*?
* ¿Qué efecto ha tenido la directiva #include <stdio.h>?

La llamada min() ha sido reemplazada por la expansión de la macro. Como se puede ver en el fichero .i:



la directiva #include <stdio.h> ha insertado todas las declaraciones necesarias para que puedas usar las funciones estándar de entrada/salida, y definido tipos de variables.

MakeFile:



Con la primera ejecución de make ocurre lo siguiente:

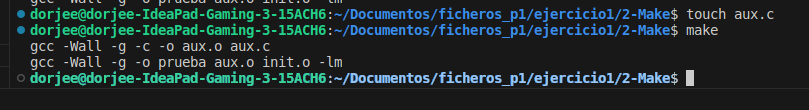
1. Compiló aux.c para producir aux.o.
2. Compiló init.c para producir init.o.
3. Enlazó aux.o e init.o para producir el ejecutable prueba, utilizando la biblioteca math -lm.

La segunda vez que ejecuto la instrucción make obtengo lo siguiente:

gcc -Wall -g -c -o aux.o aux.c

gcc -Wall -g -o prueba aux.o init.o -lm

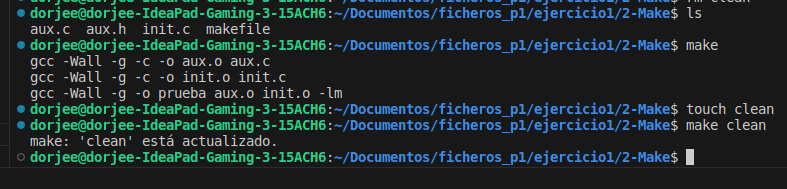
Se salta una instrucción, esto es porque al hacer touch aux.c, make “reconoce” que se ha modificado, y lo vuelve a recompilar.



Al ejecutar make clean sale lo siguiente:

-rm \*.o prueba

Significa que todos los archivos con la declinación .o serán eliminados del directorio.



Creo un archivo llamado clean con el comando touch clean.

Luego, al intentar ejecutar make clean, el mensaje make: 'clean' está actualizado. indica que make cree que el objetivo clean ya está "al día" debido a la existencia del archivo clean, y no ejecuta las acciones definidas para ese objetivo. Eso es precisamente porque el objetivo clean no está marcado como .PHONY en el Makefile, está comentado.

En un Makefile, .PHONY se utiliza para decirle a make que cierto objetivo no representa un archivo real, sino más bien una acción. Sin .PHONY, si existe un archivo con el mismo nombre que el objetivo (en este caso, clean), make pensará que ese objetivo ya está actualizado y no ejecutará sus acciones asociadas.

* Comenta la línea LIBS = -lm poniendo delante una almoadilla (#). Vuelve a contruir el proyecto ejecutando make (haz un *clean* antes si es necesario). ¿Qué sucede? ¿Qué etapa es la que da problemas?

dorjee@dorjee-IdeaPad-Gaming-3-15ACH6:~/Documentos/ficheros\_p1/ejercicio1/2-Make$ make

gcc -Wall -g -c -o aux.o aux.c

gcc -Wall -g -c -o init.o init.c

gcc -Wall -g -o prueba aux.o init.o

/usr/bin/ld: aux.o: en la función `sinDegrees':

/home/dorjee/Documentos/ficheros\_p1/ejercicio1/2-Make/aux.c:7: referencia a `sin' sin definir

/usr/bin/ld: aux.o: en la función `cosDegrees':

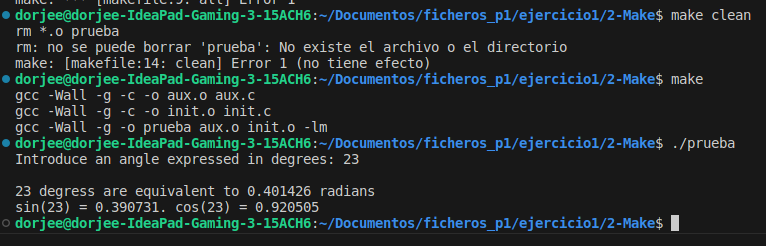
/home/dorjee/Documentos/ficheros\_p1/ejercicio1/2-Make/aux.c:12: referencia a `cos' sin definir

collect2: error: ld returned 1 exit status

make: \*\*\* [makefile:9: all] Error 1

dorjee@dorjee-IdeaPad-Gaming-3-15ACH6:~/Documentos/ficheros\_p1/ejercicio1/2-Make$

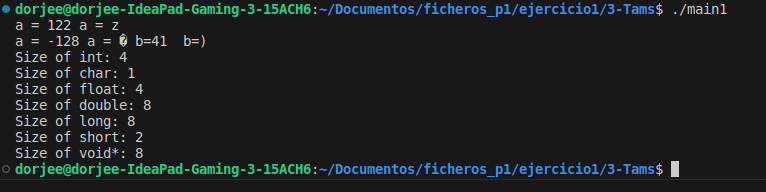
Las líneas de error están indicando que las funciones sin y cos no están definidas. Esto sucede porque al comentar la línea LIBS = -lm, se elimina la referencia a la biblioteca matemática (-lm), que es necesaria para enlazar con las funciones matemáticas.



### 3. Tamaño de variables

Compila y ejecuta el código de cada uno de los ejemplos proporcionados y responde a las preguntas proporcionadas para ellos.

* *main1.c*
  + ¿Por qué el primer printf() imprime valores distintos para ‘a’ con los modificadores %d y %c?
  + ¿Cuánto ocupa un tipo de datos char?
  + ¿Por qué el valor de ‘a’ cambia tanto al incrementarlo en 6? (la respuesta está relacionada con la cuestión anterior)
  + Si un “long” y un “double” ocupan lo mismo, ¿por qué hay 2 tipos de datos diferentes?
* *main2.c*
  + ¿Tenemos un problema de compilación o de ejecución?
  + ¿Por qué se da el problema?. Soluciónalo, compila y ejecuta de nuevo.
  + a,b,c, y x están declaradas de forma consecutiva. ¿Son sus direcciones concecutivas?
  + ¿Qué significa el modificar "%lu" en printf()?
  + ¿A qué dirección apunta "pc"? ¿Coincide con la de alguna variable anteriormente declarada? Si es así, ¿Coinciden los tamaños de ambas?
  + ¿Coincide el valor del tamaño de array1 con el número de elementos del array? ¿Por qué?
  + ¿Coinciden las direcciones a la que apunta str1 con la de str2?
  + ¿Por qué los tamaños (según sizeof()) de str1 y str2 son diferentes?



printf("a = %d a = %c \n", a, a);

%d imprime el número entero (122), mientras que %c imprime el carácter ASCII 122 que es z.

¿Cuánto ocupa un tipo de datos char? Ocupa 1 byte (8 bits).

¿Por qué el valor de ‘a’ cambia tanto al incrementarlo en 6? (la respuesta está relacionada con la cuestión anterior) Por los rangos de representación, al sumar 6, se convierte en -128. Esto hace que imprima un símbolo representado con una interrogación, significa que no es un carácter ASCII imprimible.

Si un “long” y un “double” ocupan lo mismo, ¿por qué hay 2 tipos de datos diferentes?

“Naturaleza de los datos:

long: Es un tipo de dato entero. Se utiliza para representar números enteros sin parte decimal.

double: Es un tipo de dato de punto flotante. Se utiliza para representar números reales que pueden tener una parte decimal.

Representación en memoria:

long: Los valores enteros se almacenan en un formato que simplemente representa el número en binario.

double: Los valores de punto flotante se almacenan utilizando un formato estandarizado (usualmente el estándar IEEE 754 para punto flotante de doble precisión) que separa el número en tres componentes: signo, exponente y fracción (o mantisa). Esto permite que double pueda representar tanto números muy grandes como muy pequeños, así como números con partes decimales.

Rango y precisión:

long: Puede representar números enteros en un rango específico con una precisión exacta.

double: Puede representar números en un rango muy amplio, pero con una precisión limitada. Además, debido a la forma en que funcionan los números de punto flotante, no todos los números reales pueden representarse exactamente, lo que puede conducir a errores de redondeo.

Operaciones:

Las operaciones matemáticas realizadas con long son operaciones enteras, mientras que las realizadas con double son operaciones de punto flotante. Estas últimas pueden ser más lentas, dependiendo de la arquitectura de hardware, aunque muchos sistemas modernos tienen hardware dedicado para operaciones de punto flotante que hace que estas operaciones sean rápidas.”

MAIN2.C

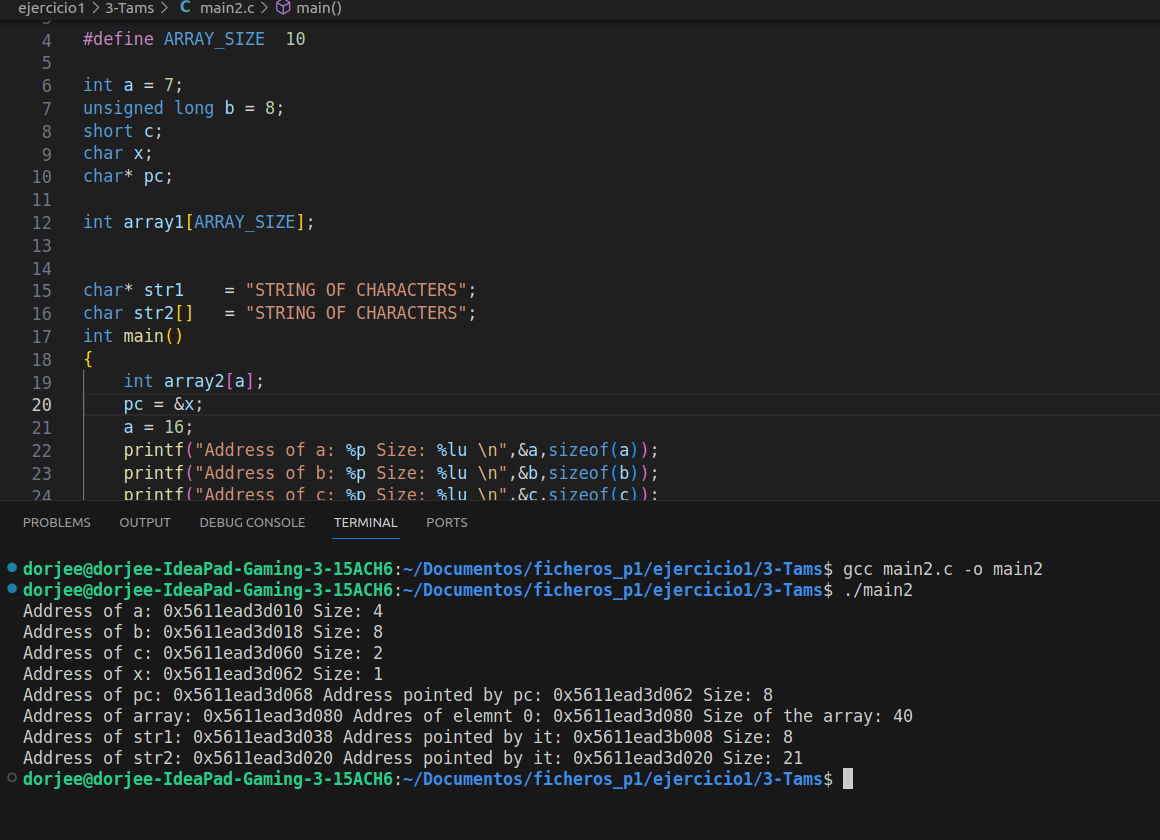
dorjee@dorjee-IdeaPad-Gaming-3-15ACH6:~/Documentos/ficheros\_p1/ejercicio1/3-Tams$ gcc main2.c -o main2

main2.c:13:5: error: variably modified ‘array2’ at file scope

13 | int array2[a];

| ^~~~~~

Solución:



Se da el problema porque aún no se ha declarado a, por eos se debe mover al main, o simplemente pasarle un entero para determinar el valor.

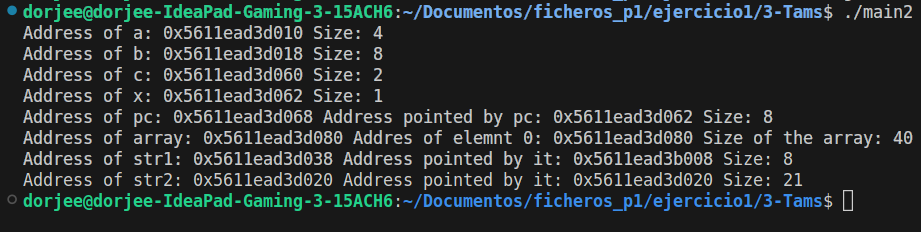
* a,b,c, y x están declaradas de forma consecutiva. ¿Son sus direcciones concecutivas?

Están declaradas consecutivamente, pero esto no garantiza que sus direcciones sean consecutivas.

* ¿Qué significa el modificar "%lu" en printf()?

%lu significa long unsigned, en este caso quiere imprimir un long unsigned number.

* ¿A qué dirección apunta "pc"? ¿Coincide con la de alguna variable anteriormente declarada? Si es así, ¿Coinciden los tamaños de ambas?



**¿Coincide con la de alguna variable anteriormente declarada?** Sí, coincide con la dirección de la variable x, cuya dirección es 0x5611ead3d062.

**¿Coinciden los tamaños de ambas?**

No, ya que x es de tipo char y ocupan 1 byte y pc, que es un puntero ocupa 8 bytes.

**¿Coincide el valor del tamaño de array1 con el número de elementos del array? ¿Por qué?**

Pone que size of array1 es 40 pero no muestra el tamaño del array1, sino el tamaño de bytes que ocupa. Hace esta operación al compilar int array1[ARRAY\_SIZE] = (tipo de variable) x (tamaño) que en este caso sería 4 x 10 = 40.

**¿Coinciden las direcciones a la que apunta str1 con la de str2?**

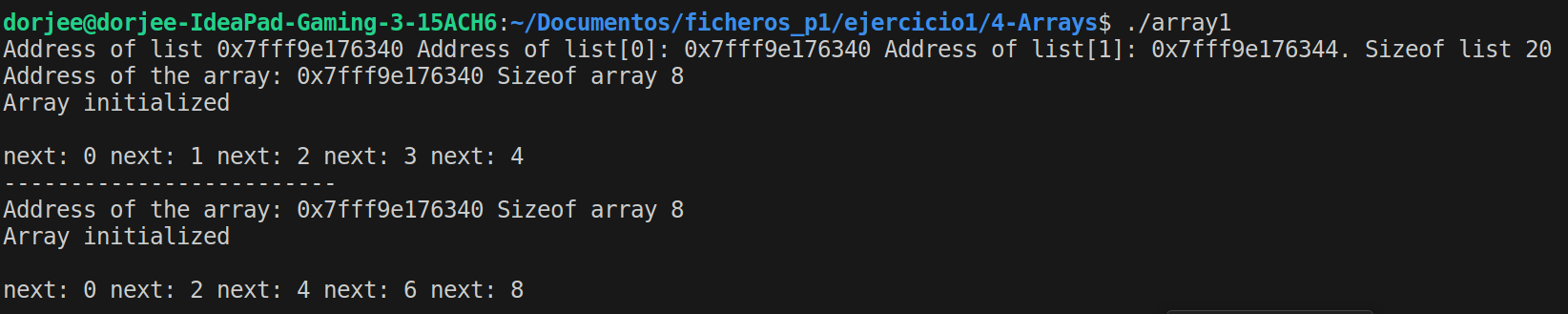
No, porque una es un puntero de tipo char, y la otra es un array de caracteres que necesita más memoria asignada.

**¿Por qué los tamaños (según sizeof()) de str1 y str2 son diferentes?**

Porque str1 ocupa 8 bytes al ser un puntero y str2 ocupa 21 bytes, 20 de caracteres y 1 byte contando el carácter nulo.

### 4. Arrays

Compila y ejecuta el código de los ejemplos proporcionados y responde a las preguntas propuestas para cada uno de ellos.



* *array1.c*
  + ¿Por qué no es necesario escribir "&list" para obtener la dirección del array list?

Porque el nombre list ya es un puntero al primer elemento del array, por eso no es necesario usar &.

* + ¿Qué hay almacenado en la dirección de list?

0x7fff9e176340 un 0 y va hasta 0x7fff9e176350 donde hay un 4

* + ¿Por qué es necesario pasar como argumento el tamaño del array en la función init\_array?

Cuando pasas un array a una función, en realidad estás pasando un puntero al primer elemento del array. La función no tiene información sobre el tamaño del array; solo sabe dónde comienza en la memoria. Por lo tanto, no puedes determinar el tamaño del array dentro de la función usando sizeof de la misma manera que podrías hacerlo fuera de la función.

* + ¿Por qué el tamaño devuelto por sizeof() para el array de la función init\_array no coincide con el declarado en main()?

Porque se le está pasando únicamente un puntero al primer elemento del array.

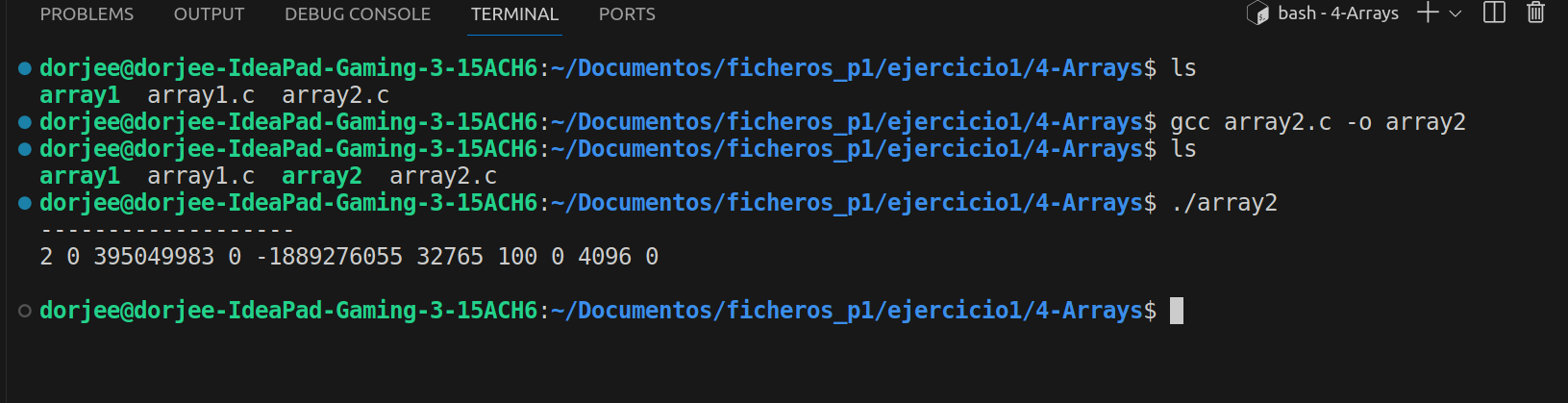
* + ¿Por qué NO es necesario pasar como argumento el tamaño del array en la función init\_array2?

Porque N es una constante ya declarada y en el mismo método ya inicializas i = N.

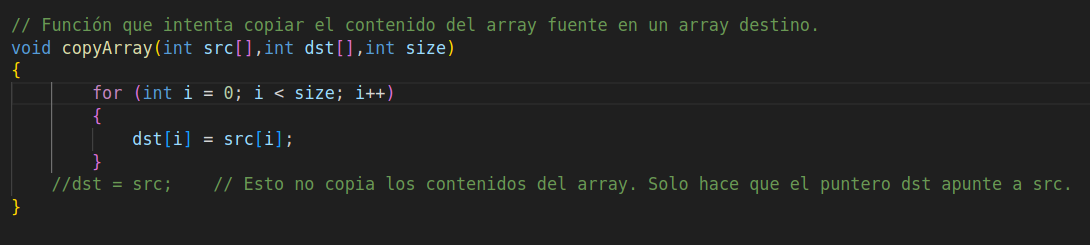
* + ¿Coincide el tamaño devuelto por sizeof() para el array de la función init\_array2 con el declarado en main()?

El Sizeof list 20 muestra que el tamaño total del array list es de 20 bytes (dado que N es 5 y, un int ocupa 4 bytes, el tamaño total es 5 \* 4 = 20).

El Sizeof array 8 muestra que, dentro de las funciones init\_array y init\_array2, el parámetro array se trata como un puntero, y en una arquitectura de 64 bits, el tamaño de un puntero es de 8 bytes.

* *array2.c*
  + ¿La copia del array se realiza correctamente? ¿Por qué?

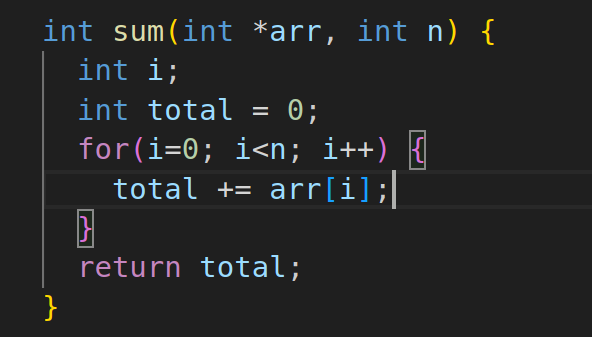
No lo hace, lo único que hace es que dst apunte a src, pero no copia los valores, para eso se haría lo siguiente.

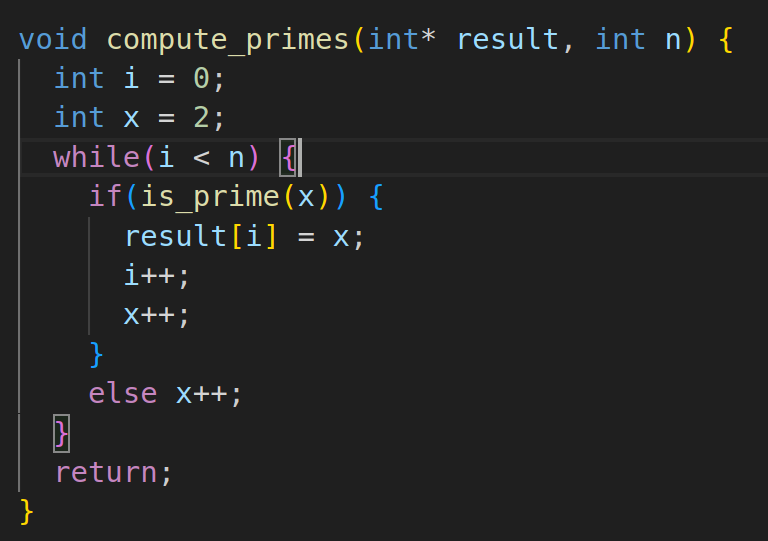


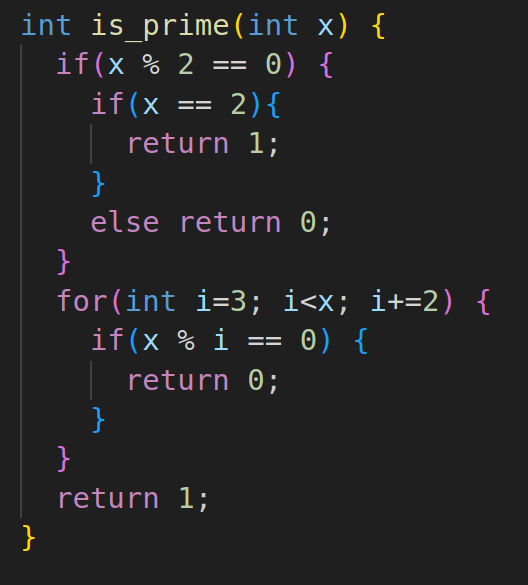
* + Si no es correcto, escribe un código que sí realice la copia correctamente. Hecho como arriba, itera copiando los valores de src[i] a dst[i].
  + Descomenta la llamada a la función tmo en la función main(). Compila de nuevo y ejecuta.
  + El problema que se produce, ¿es de compilación o de ejecución? ¿Por qué se produce?
  + Encuentra un valor de MAXVALID superior a 4 con el que no se dé el problema. ¿Se está escribiendo más allá del tamaño del array? Si es así, ¿por qué funciona el código?

Ejercicio 2: Hecho!

Arreglos:







**Ejercicio 4:**

Para agregar la opción -i <inputfile>, debemos realizar los siguientes pasos:

1. Modificar la estructura de options para incluir una variable que almacene la ruta del archivo de entrada.
2. Modificar la función parse\_passwd para que tome el archivo especificado en options en lugar de /etc/passwd por defecto.
3. Modificar el bucle que analiza las opciones de línea de comandos (getopt) en main para reconocer la opción -i.

**Ejercicio 5:**

1. Usando read para leer y mostrar el contenido de /etc/passwd:

Primero, necesitas entender que read puede tomar varias variables y asignarles valores basados en un delimitador. En el caso de /etc/passwd, los valores están delimitados por :.

Por lo tanto, podemos usar read en un bucle while para leer cada línea del archivo, y luego asignar cada valor delimitado a una variable. Después, podemos usar echo para mostrar estas variables.

Script Básico:

#!/bin/bash

while IFS=':' read -r login\_name encrypted\_passwd uid gid user\_name user\_home user\_shell

do

echo "Login: $login\_name, UID: $uid, Home: $user\_home"

done < /etc/passwd

Este script leerá /etc/passwd y mostrará el nombre de inicio de sesión, UID y directorio de inicio de cada usuario.

2. Mostrar solo entradas con directorio home en /home:

Para esta parte, solo necesitas agregar una condición para verificar si el directorio de inicio ($user\_home) comienza con /home.

Script Modificado:

#!/bin/bash

while IFS=':' read -r login\_name encrypted\_passwd uid gid user\_name user\_home user\_shell

do

if [[ $user\_home == /home/\* ]]; then

echo "Login: $login\_name, UID: $uid, Home: $user\_home"

fi

done < /etc/passwd

Este script filtrará y mostrará solo las entradas donde el directorio de inicio es un subdirectorio de /home.

3. Usando cut y grep para mostrar directorios home que comiencen con /home:

Finalmente, podemos usar comandos de línea de comandos para obtener el mismo resultado. cut se utiliza para seleccionar columnas específicas de un archivo y grep para filtrar líneas basadas en un patrón.

Comando:

cut -d ':' -f 6 /etc/passwd | grep '^/home'

Explicación:

cut -d ':' -f 6 /etc/passwd: Extraerá la sexta columna (que es el directorio de inicio) de /etc/passwd.

grep '^/home': Filtrará las líneas que comienzan con /home.