

**Instituto Politécnico Nacional**

**Escuela Superior de Cómputo**

**Sistemas Operativo Practica 5**

**HILOS (THREADS) EN LENGUAJE C**

**ENFOCADOS A LINUX**

*Integrantes:*

*Ayona López Eugenio Milton*

*Morales Blas David Israel*

Profesora:

JUÁREZ MÉNDEZ ANA BELEM

Grupo: 2CM8

Fecha de entrega: 27 de marzo del 2020

# Objetivo

Utilizar los conocimientos de hilos en aplicaciones de sistemas operativos.

# Introducción

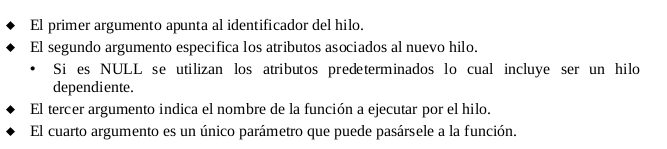
Un hilo es una manera de programar, que se puede considerar demasiado eficiente, debido al principal factor que nos proporciona, lo cual es correr tareas casi simultaneas, esto lo digo porque una computadora no puede ejecutar dos o mas tareas a la vez, hay un cierto margen en cada “proceso ligero”, pero es tan pequeño que nos da la ilusión de simultaneidad.

Si vamos a conocimientos anteriores y nos posicionamos en el punto en donde vimos el uso de la función fork(), notamos que hay varias similitudes y que hasta podríamos llegar a pensar que son las mismas cosas (lo cual yo llegue pensar al inicio), pero esto no se debe ver así, ya que puede ocasionar demasiados problemas a la hora de querer implementar ya sea los procesos y los hilos.

La mejor manera que yo encontré para encontrar las diferencias, fue claramente llenarme de información, que después utilizaba para visualizar estos conceptos al momento de empezar a programar. Pero en general quiero describir las diferencias que note en la parte de abajo…

Para crear un hilo es necesario usar la función siguiente:

**int pthread\_create(pthread\_t \*thread, pthread\_attr\_r \*attr, void\*(\*start\_routine)(void\*),void \*arg);**



Para poder obserbar el resultado de los hilos es necesario usar la funcion pthread\_join, está funcion cabe destacar sólo se usa en hilos independientes es decir el segundo parametro de **pthread\_create** es **NULL.**

Por utlimo es necesario cerrar el hilo cuando se termine de usar esto se hace con la función **int pthread\_exit(void \*value);**

## **DIFERENCIAS ENTRE UN PROCESO Y HILO:**

* Un proceso es algo más complejo.
* En los procesos la memoria esta limitada, ya que se establece un bloque en donde este solo puede trabajar y en los hilos la memoria es independiente.
* Trabajar con hilos se me facilito mas debido a que podemos separar lo que nosotros queramos ejecutar en el en una función independiente, tanto podemos utilizar la misma función como crear otras para los hilos que creamos; Por tanto, la manera en como un proceso trabaja puede ser mas engorrosa es decir al llamar a fork n veces se crean 2^N procesos donde n es el numero de llamadas a la función, controlar cuantos procesos hijos se requieren necesita 2 fors y su control es más complicado de cierta forma.
* El tiempo de ejecución en un proceso es mucho más lento que un hilo.

**SIMILITUDES ETRE HILO Y PROCESO:**

* Ejecución de tareas casi simultaneas.
* Programación de los tiempos en que los hilos o procesos son sincronizados.
* Comunicación un poco mas abstracta con nuestro sistema operativo.
* Procesamiento de señales.

En general los hilos y procesos son dos estilos que puedes utilizar independientemente a la hora de la resolución de problemas que implican el uso de tareas simultaneas.

# **DESARROLLO DE LA PRACTICA**

***El ejercicio 1 cuenta con dos soluciones debido a que en una no se aprecia bien como cada hilo imprime la palabra, en la segunda solución si es posible obserbar, además en uno los valores se piden desde el main y en otro se mandan desde ejecución.***

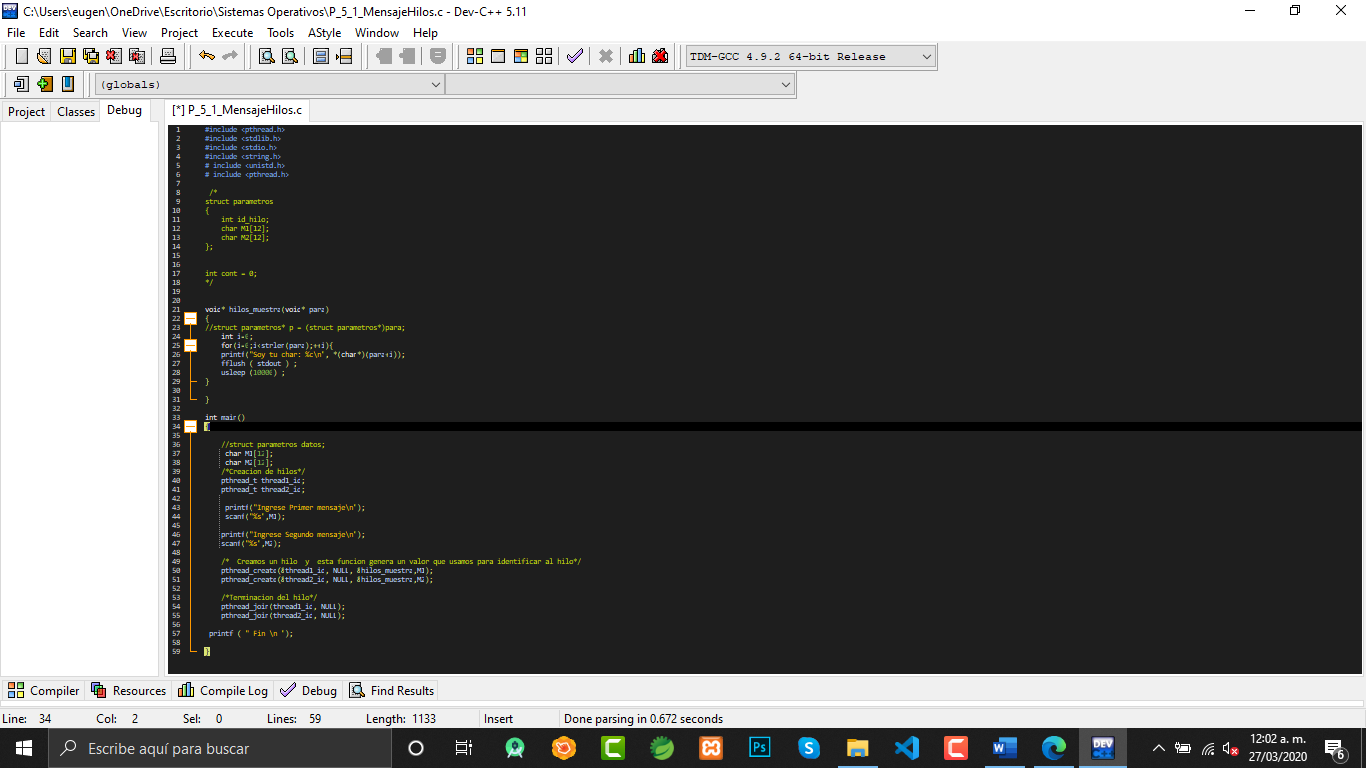
## **Ejercicio 1 Solución 1**

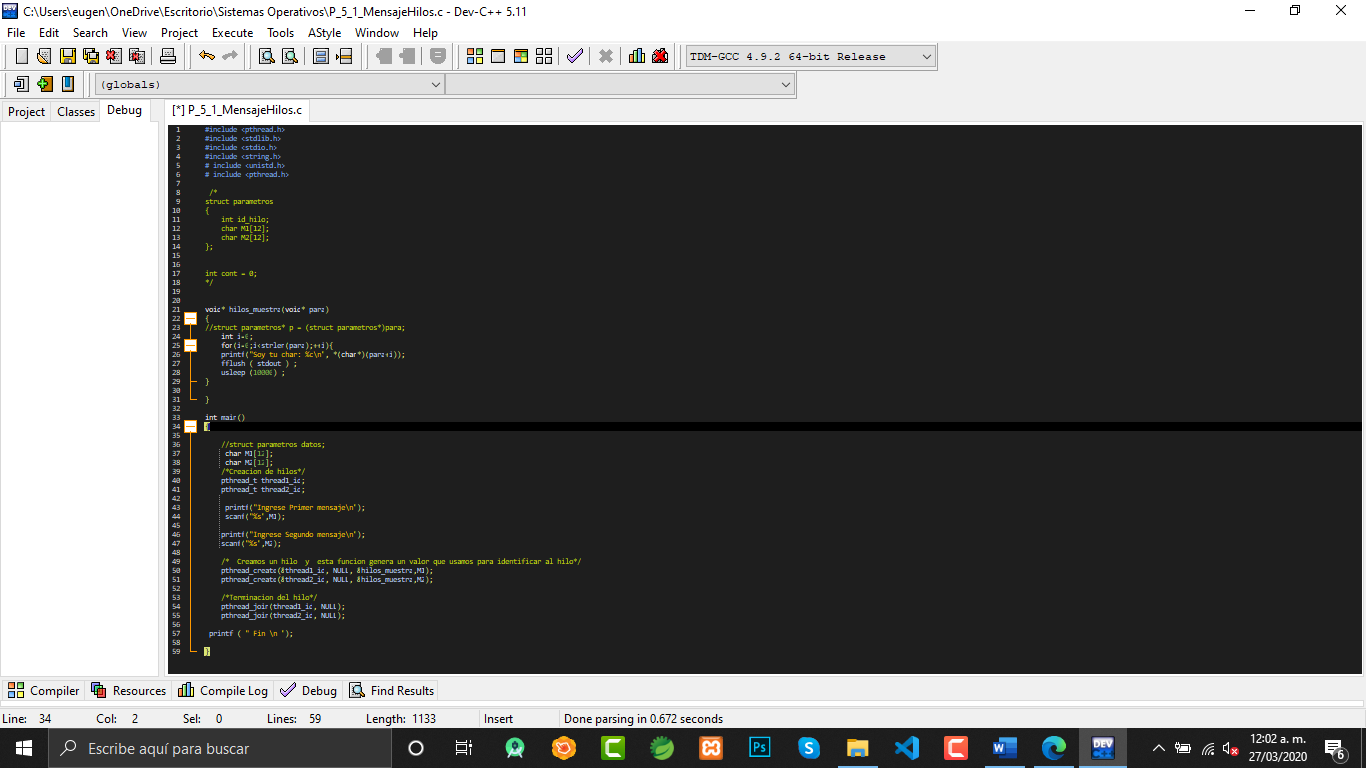
Se nos ha dado la tarea de resolver la practica 5 en la cual incluye 4 problemas, que implican el uso de hilos, los que a continuación se describen:

*Problema 1. \_* Dado dos mensajes M1 y M2, los cuales respectivamente se deben pasar en dos hilos H1 y H2 y como último paso cada hilo debe imprimir en la pantalla el mensaje respectivo. Además de agregar al final la palabra FIN cuando se terminen de ejecutar los dos procesos.

-Este problema es fundamental, ya que nos enseña el uso, creación y el efecto que tiene el uso de hilos.

**Aspectos a tomar en cuenta**

* Como pasar parámetros al momento de crear el hilo
* Tiempo de espera para que un hilo realice su tarea **Código C**

****

Invocación de los hilos y envio de datos

Terminación del hilo

Bibliotecas utilizadas en el programa

Ingreso de datos

Variables

Función ejecutada por el hilo

**Descripción:**

Bibliotecas necesarias para uso de hilos

#include <pthread.h>

## Variables

Podemos observar que se encuentran uno que ya conocemos que es char y uno que no es muy conocido llamado pthread\_t.

pthread\_t: Variable ubicada en pthread utilizada para la invocación de un hilo.

## Ingreso de datos

Aquí ya es un uso mas funcional en nuestra práctica, ya que en esta sección pedimos los mensajes que nuestro usuario va ingresar, que este caso pueden tener una longitud máxima de 12 caracteres.

### Invocación de los hilos y envio de datos

Al igual que la variable pthread\_t, nuestra funcion pthread\_create se encuentra en la biblioteca pthread.

pthread\_create : Funcion engargada de invocar nuestro hilo, recibe cuatro parametros, que se enlistan abajo:

1. Se encarga de recibir la dirección de nuestro hilo en especifico.
2. Personalizacion de los atributos que nuestro hilo va a tener, como damos un NULL, se generan los atributos que vienen por defecto.
3. Recibi la direccion de la función que nuestro hilo va ejecutar, en este la funcion es **hilos\_muestra**
4. Los parametros que va a mantar a nuestra funcion que se ejecuta en el hilo, en este caso son los mensajes que el usuario ha ingresado.

## Función ejecutada por el hilo

La parte mas importante de nuestro código es esta debido a que aquí se encuentra la lógica de nuestro programa.

Uso: solo muestra en pantalla el mensaje que se envió en cada hilo.

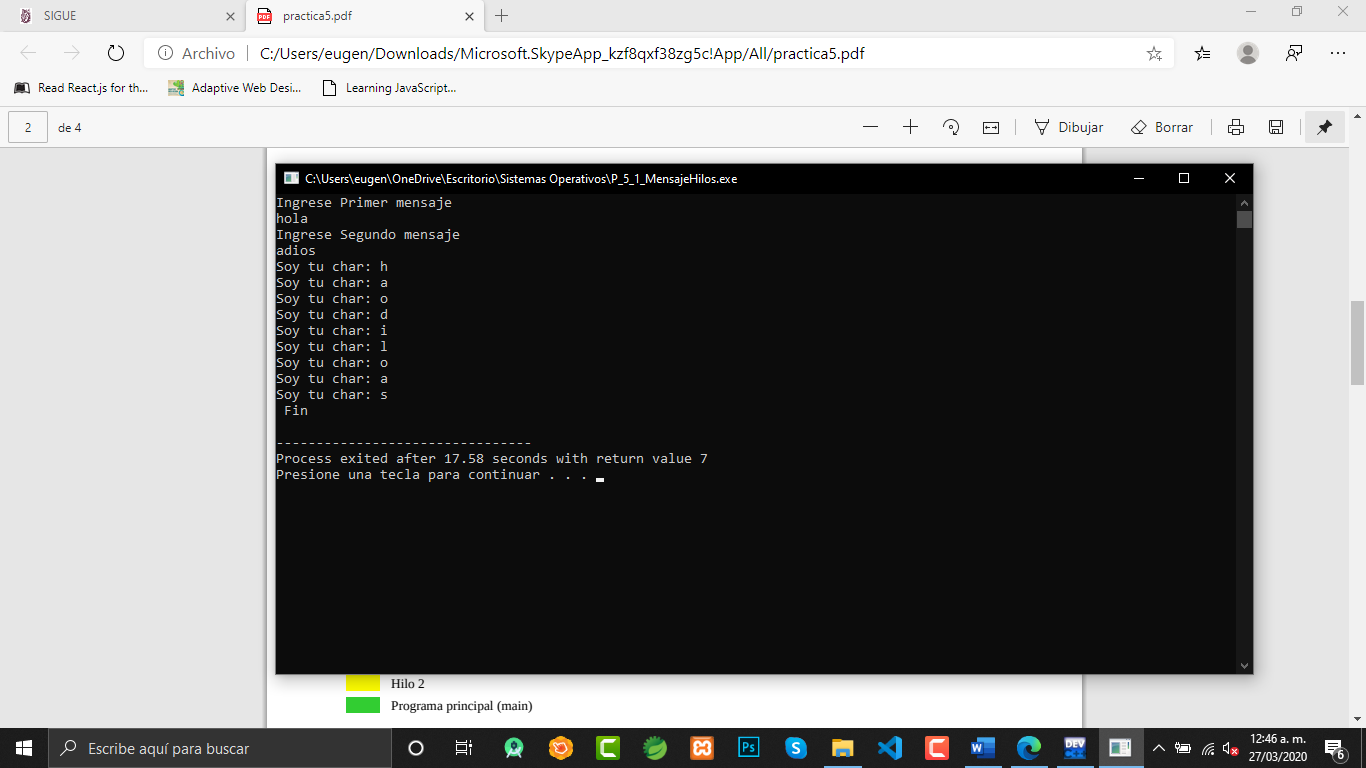
Indicaciones

* Debido a que nuestro mensaje se encuentra en un apuntador a un tipo void, hay que castearlo para que se revele el mensaje lo cual hacemos con este código \*(char\*)(para+i)

Un aspecto a tomar en cuenta es (para+i) talvez esta no sea una manera muy conocida, pero tiene el mismo funcionamiento a si estuviéramos haciendo M[indice].

* strlen() esta función nos devuelve el tamaño de nuestro mensaje, esta recibe como parámetros un apuntador en este caso al mensaje, nos daría la longitud del mensaje, la que usamos como condición en nuestro “for”, para que imprima carácter por carácter, hasta el tamaño final.
* Y por ultimo esta las funciones usleep() y fflush, que se encarga de dormir a nuestro hilo unos milisegundos con el fin de observar el funcionamiento casi simultaneo de nuestro hilo y para limpiar la salida de nuestro buffer, respectivamente.

**SALIDA EN CONSOLA**



Ingreso de los mensajes

Se supone que la programación sin hilos, deberia mostras un mensaje parecido a “hola adios”, pero como anteriomente explique, esto sucede debido al comportamiento del hilo, ya que como se estan ejecutando simultaneamente, su salida es aleatorio, ya que la computadora pasa de manera casi simultanea de hilo a hilo, provocando este efecto.

### EJERCICIO 1 Solucion 2

Representación del mismo problema pero con un código donde se puede apreciar de mejor manera como cada hilo ejecuta la palabra carácter por carácter.

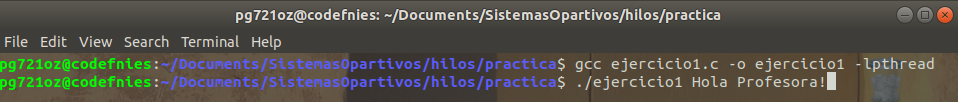
Código en c:

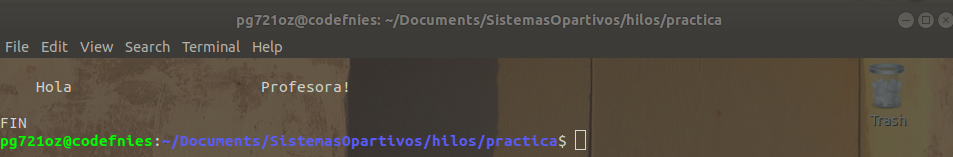


Como se puede obserbar la funcion gotoxy es la encargada de cambiar la posicion de impresion de la funcion \*saludo

Aqui los valores se piden desde la ejecución es decir char \*argv[] es el encargado de tomar los dos valores.

### Resultados en pantalla código 1 solución 2.



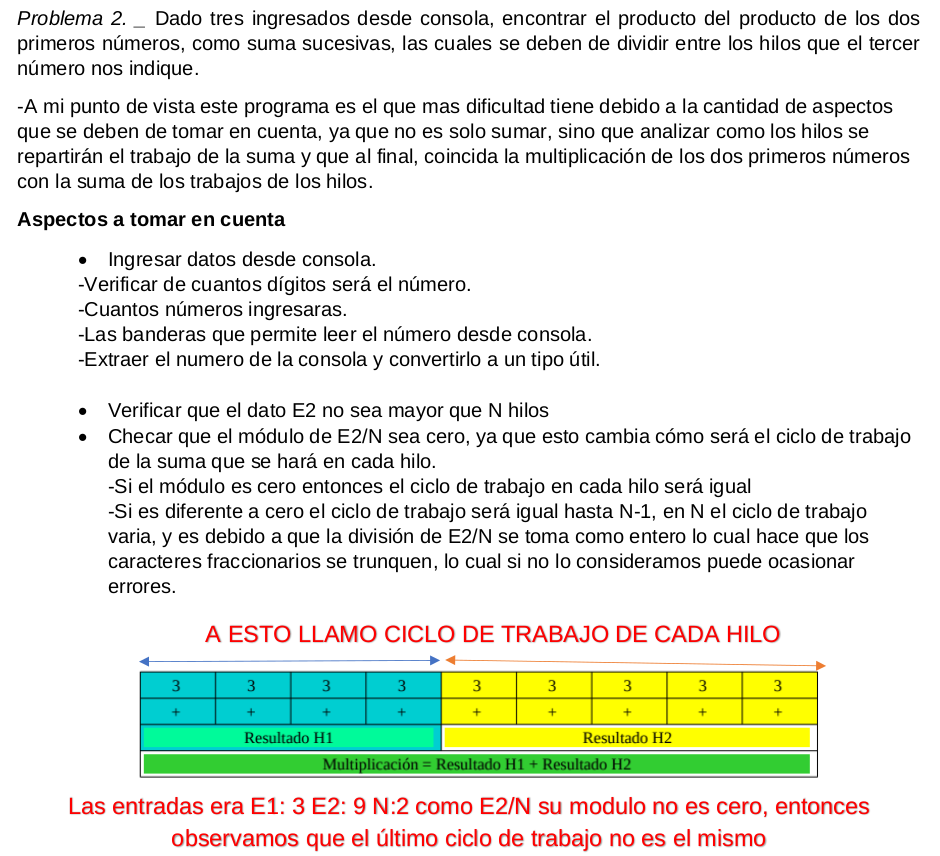


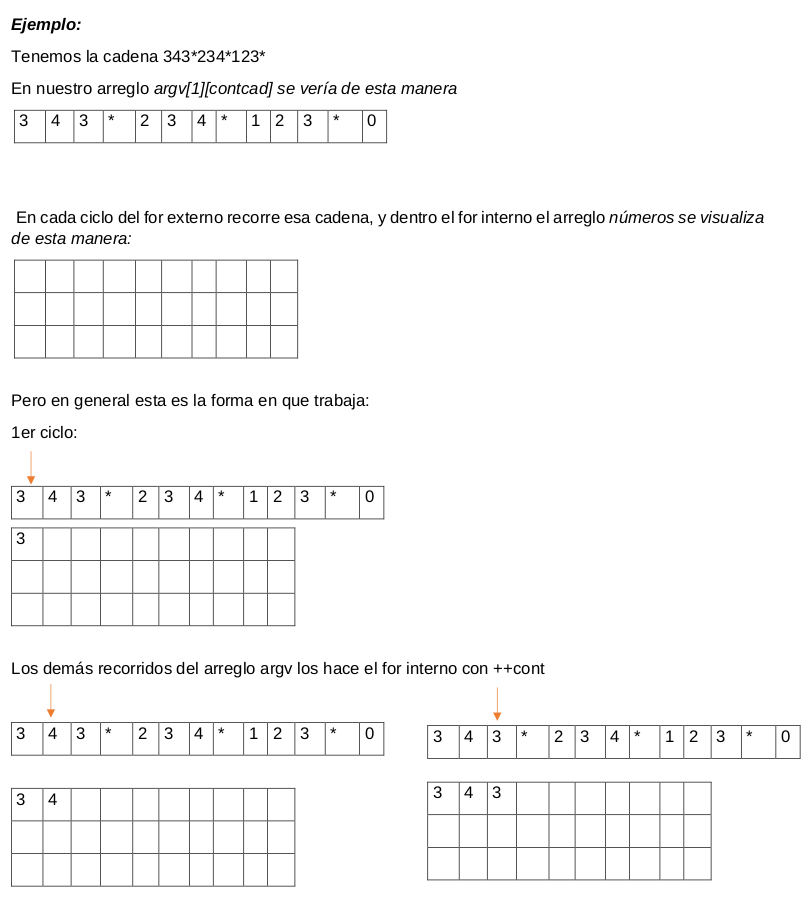
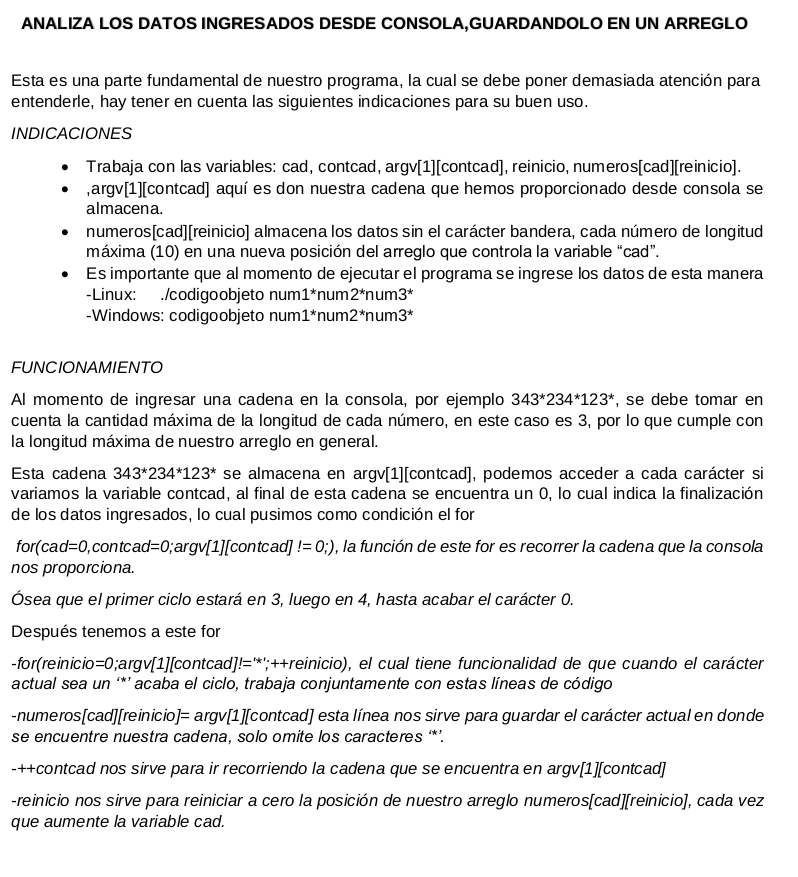
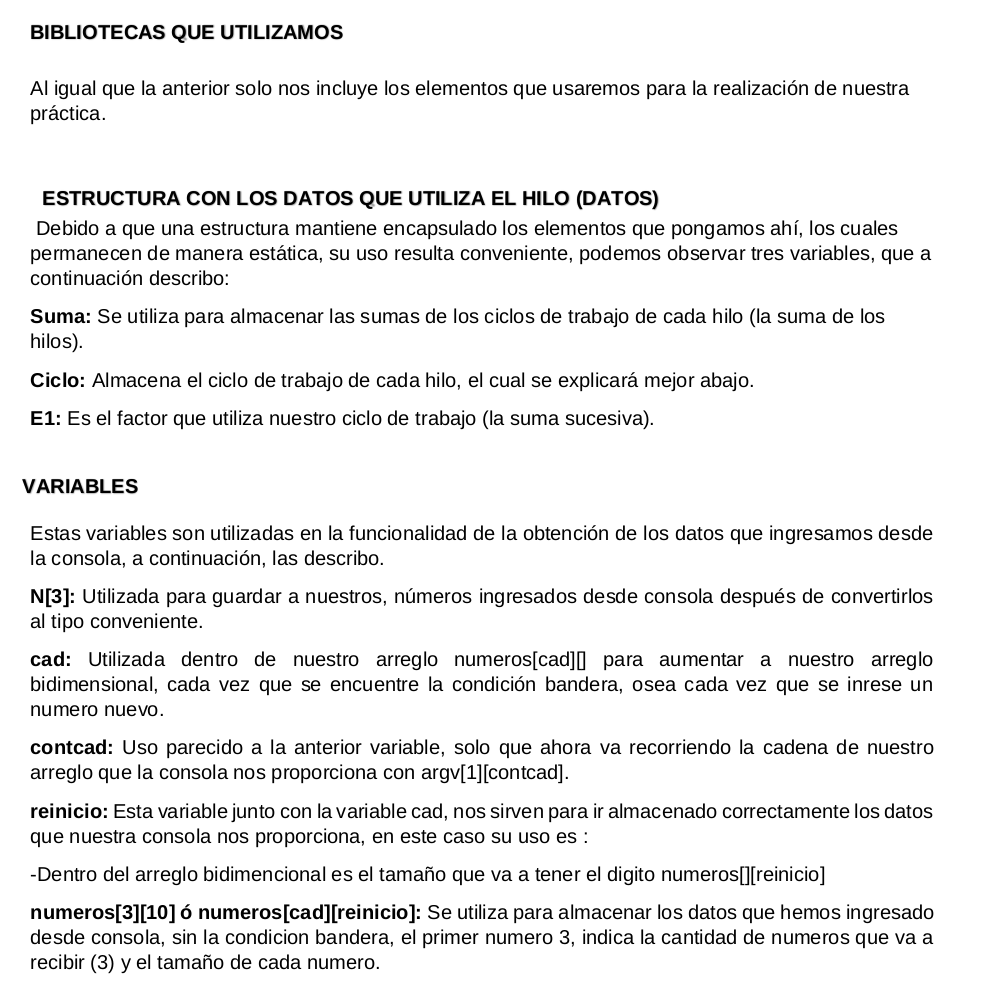
### Conclusión ejercicio 1 solucion 2

Como se puede obserbar el código limpia el la pantalla con la funcion system(“clear”);

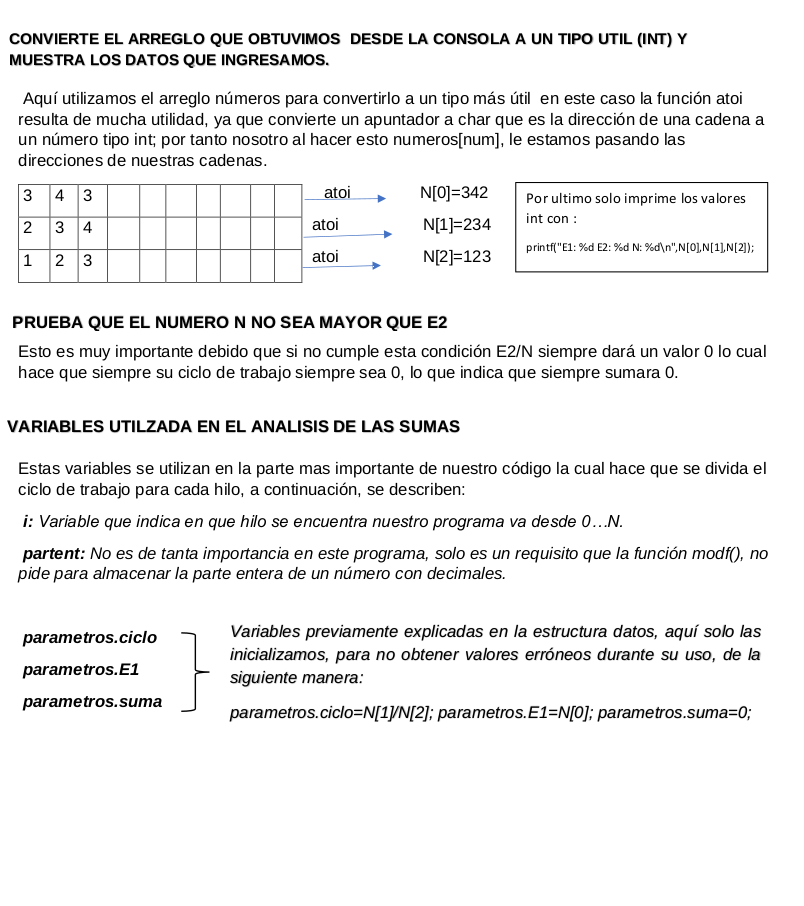
por ello en la screanshoot se obserba que se escirbe lo de los dos hilos en la parte de arriba y posterior escribe la palabra fin una vez que los hilos se temrinaron de ejecutar, cabe destacar que dar un tamaño de palabra como argumento muy grande debido a que la impresion se encuentran en la misma linea sobreescribirá en la salida la letra, es decir no se podría apreciar bien, pero su funcionameinto sigue siendo correcto.

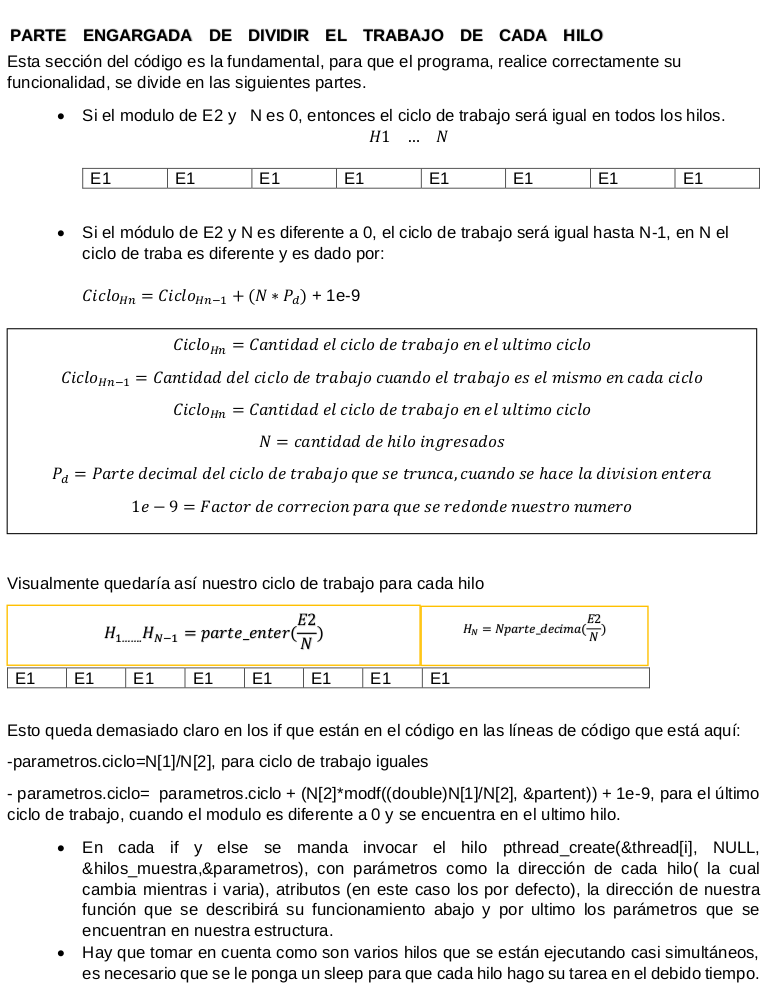
## Ejercicio 2.

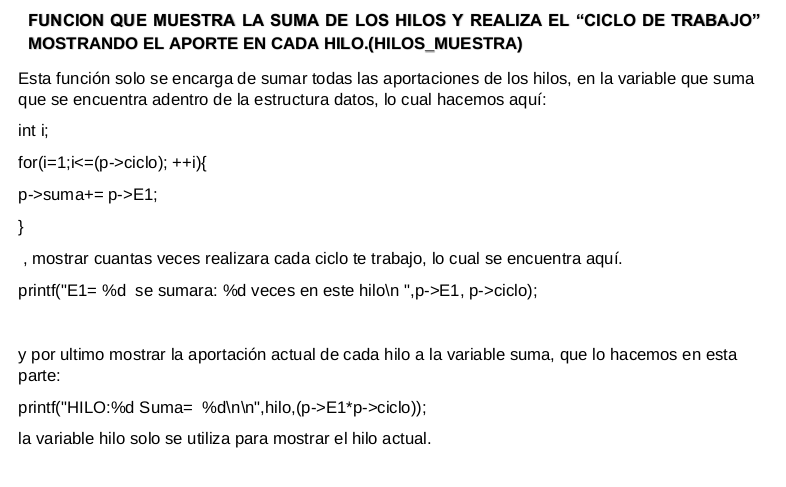




## 







## 

## Ejercicio 3.

*!PROFESORA IMPORTANTE!*

*//Las unidades de incremento y decremento no supimos si tenian que ser ya declaradas o se tenian que mandar desde la ejecución o desde el main.*

*//Las unidades de decremento no sabemos si las mandamos igual pueden ser positivas por lo cual esta el for que rectifica que siempre el segundo valor sea negativo y el primero siempre sea positivo.*

Realizar un programa con una variable entera global con un valor inicial de cero. Crear un hilo que

incremente la variable global en A unidades. Crear otro hilo que la disminuya en B unidades. Al final

el hilo principal(main) imprimirá el valor de la variable global.

### Aspectos a tomar en cuenta:

\* Uso de hilos funcion pthread\_create

### Bibliotecas necesarias:

#include <pthread.h>

### Variables:

Struct data:

El cual contiene un id y un valor el cual se le asignara por medio del hilo creado para aumentar o disminuir el valor de la variable global

numerG:

Variable global

decremento:

Variable de decremento.

incremento:

Variable de incremento.

statush1:

Valor tipo int devuelto por pthread\_create para verificar si es correcto la creacion del hilo.

statush2:

Valor tipo int devuelto por pthread\_create para verificar si es correcto la creacion del hilo.

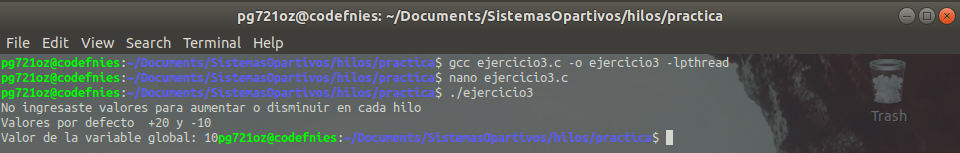
### Código:

### 

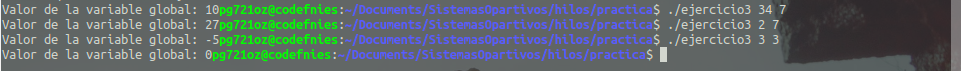
/ *\*Estos dos if hacen que siempre el primer valor sea positivo y el segundo negativo \**/

/ *\*Ya que no sabiamos bien que queria en el 3 por defecto deja un numero de aumento de 20 y uno de decremento de 10\**/

### Salida por defecto es decir al no mandar valores:

Como se aprecia se queda por defecto un 20 y un -10 y el valor de la variable incrementa en 20, valiendo 20 y despues decrementa en 10 valiendo entonces +10

### Salida dando valores

Como se muestra siempre el primer valor lo toma como positivo y el segundo valor negativo.

### Ejercicio 4

NOTA Si el segundo argumento de la función pthread\_create es NULL se utilizan los atributos

predeterminados, lo cual implica ser un hilo dependiente. Si un hilo es dependiente no se liberarán sus

recursos a menos que otro hilo espere su finalización, esto mediante pthread\_join. Se pueden

crear hilos sin necesidad de esperar a que dichos hilos terminen, de ser así deben tener el atributo de

independientes (PTHREAD\_CREATE\_DETACHED).

La función pthread\_attr\_setdetachstate(pthread\_attr\_t\*attr,intdetachstate)

permite establecer el estado de terminación de un hilo, si el segundo argumento es

PTHREAD\_CREATE\_DETACHED, el hilo se considera como independiente.

### Blibliotecas a usar:

#include <pthread.h> → Necesaria para manejar los hilos

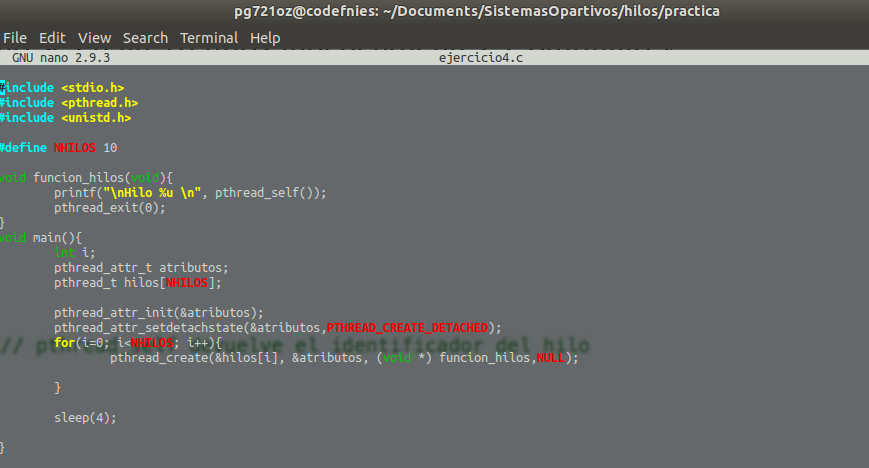
#include <unistd.h> → Para el sleep

### Variables:

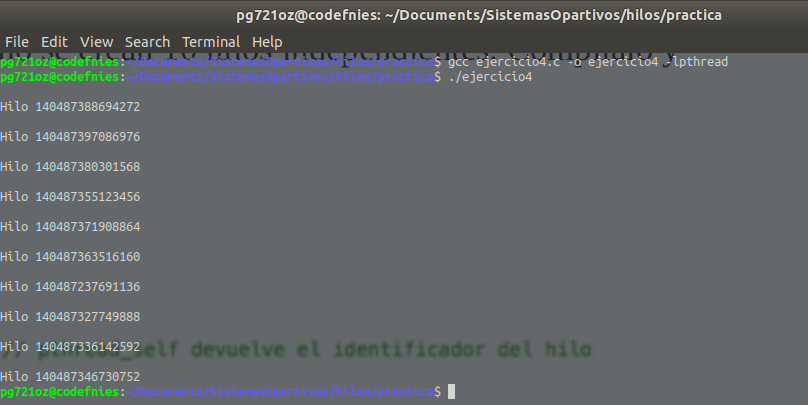
hilos[] → Contenedor para crear los 10 hilos.

NHILOS → int con valor de 10-

Codigo:



Salida del ejercicio 4

Como se puede obserbar se muestran los 10 hilos con su respectivo identificador obtenido por pthread\_self, el cual es necesario imprimirlo con %lu ya que sino tirá warnings.