Instituto Politécnico Nacional

Escuela Superior de Cómputo

HILOS (THREADS)
EN LENGUAJE C
ENFOCADOS A LINUX

Integrantes:

Ayona López Eugenio Milton

Sistemas Operativo

Profesor:

JUÁREZ MÉNDEZ ANA BELEM

Grupo: 2CM8 Fecha de entrega: 27 de marzo del 2020

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…

DIFERENCIAS A MI PUNTO DE VISTA:

- 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 se me hace extraña todavía.
- El tiempo de ejecución en un proceso es mucho más lento que un hilo.
- Funcionamiento extraño.

SIMILITUDES A MI PUNTO DE VISTA:

- 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

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:

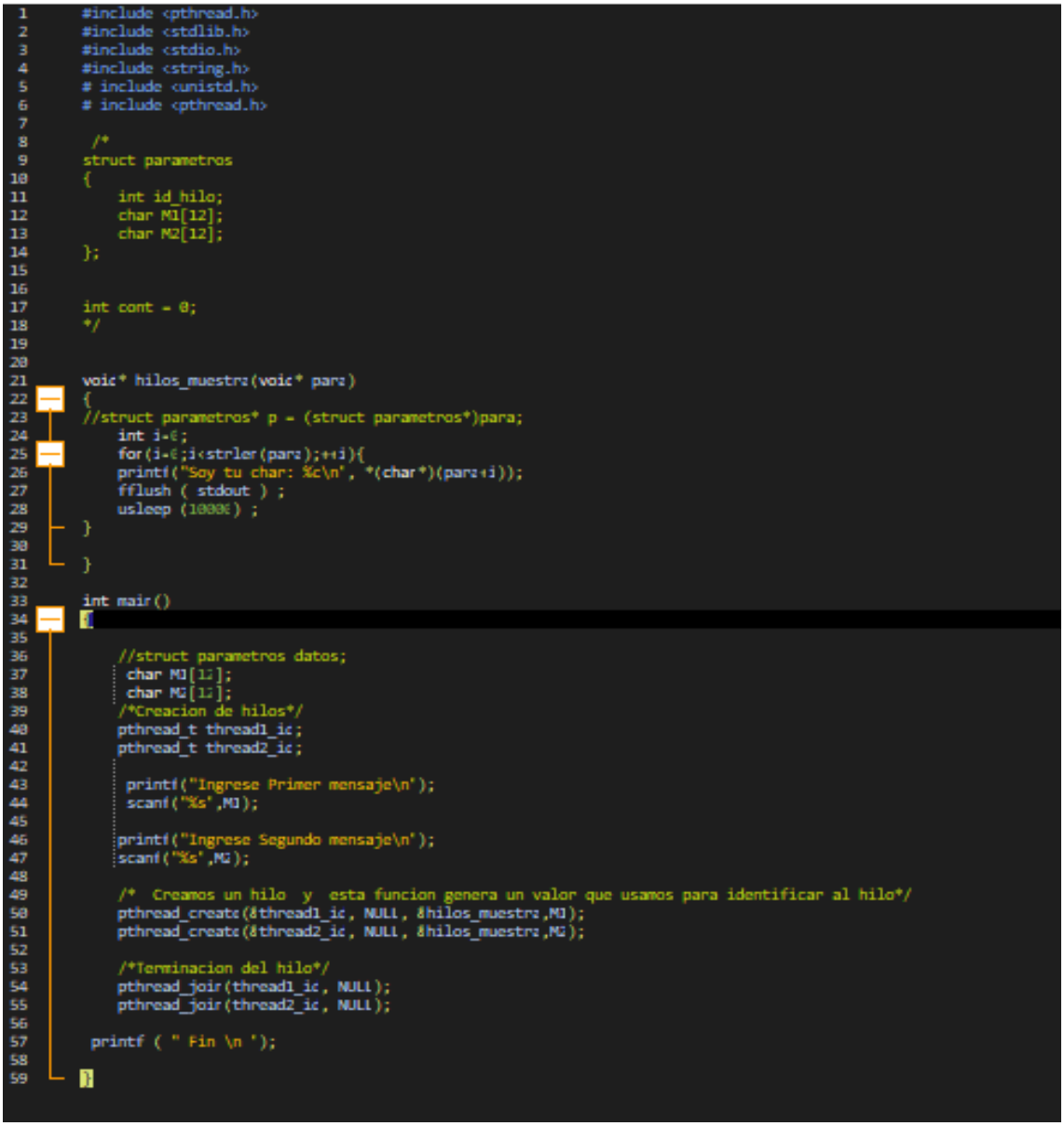
Problema 1. _ Dado dos mensajes M1 y M2, los cuales respectivamente se deben pasar en dos hilos H1 y H2 y como último paso cada hilo debe imprimir en la pantalla el mensaje respectivo.

-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



```c
#include <pthread.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
# include <unistd.h>
# include <pthread.h>

 /*
struct parametros
{
    int id_hilo;
    char M1[12];
    char M2[12];
};


int cont = 0;
*/


void* hilos_muestra(void* para)
{
//struct parametros* p = (struct parametros*)para;
    int i=0;
    for(i=0;i<strlen(para);++i){
    printf("Soy tu char: %c\n", *(char*)(para+i));
    fflush ( stdout ) ;
    usleep (1000) ;
}

}

int main()
{

    //struct parametros datos;
     char M1[12];
     char M2[12];
     /*Creacion de hilos*/
    pthread_t thread1_id;
    pthread_t thread2_id;

     printf("Ingrese Primer mensaje\n");
     scanf("%s",M1);

    printf("Ingrese Segundo mensaje\n");
    scanf("%s",M2);

    /*  Creamos un hilo  y  esta funcion genera un valor que usamos para identificar al hilo*/
    pthread_create(&thread1_id, NULL, &hilos_muestra,M1);
    pthread_create(&thread2_id, NULL, &hilos_muestra,M2);

    /*Terminacion del hilo*/
    pthread_join(thread1_id, NULL);
    pthread_join(thread2_id, NULL);

 printf ( " Fin \n ");

}
```

Descripción:

Bibliotecas utilizadas en el programa

Solo se llaman las bibliotecas que vamos a utilizar.

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, es muy importante que los parametros que regrese y reciba esta función sean apuntatores a un nulo (lo dice en las especificaciones).
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.

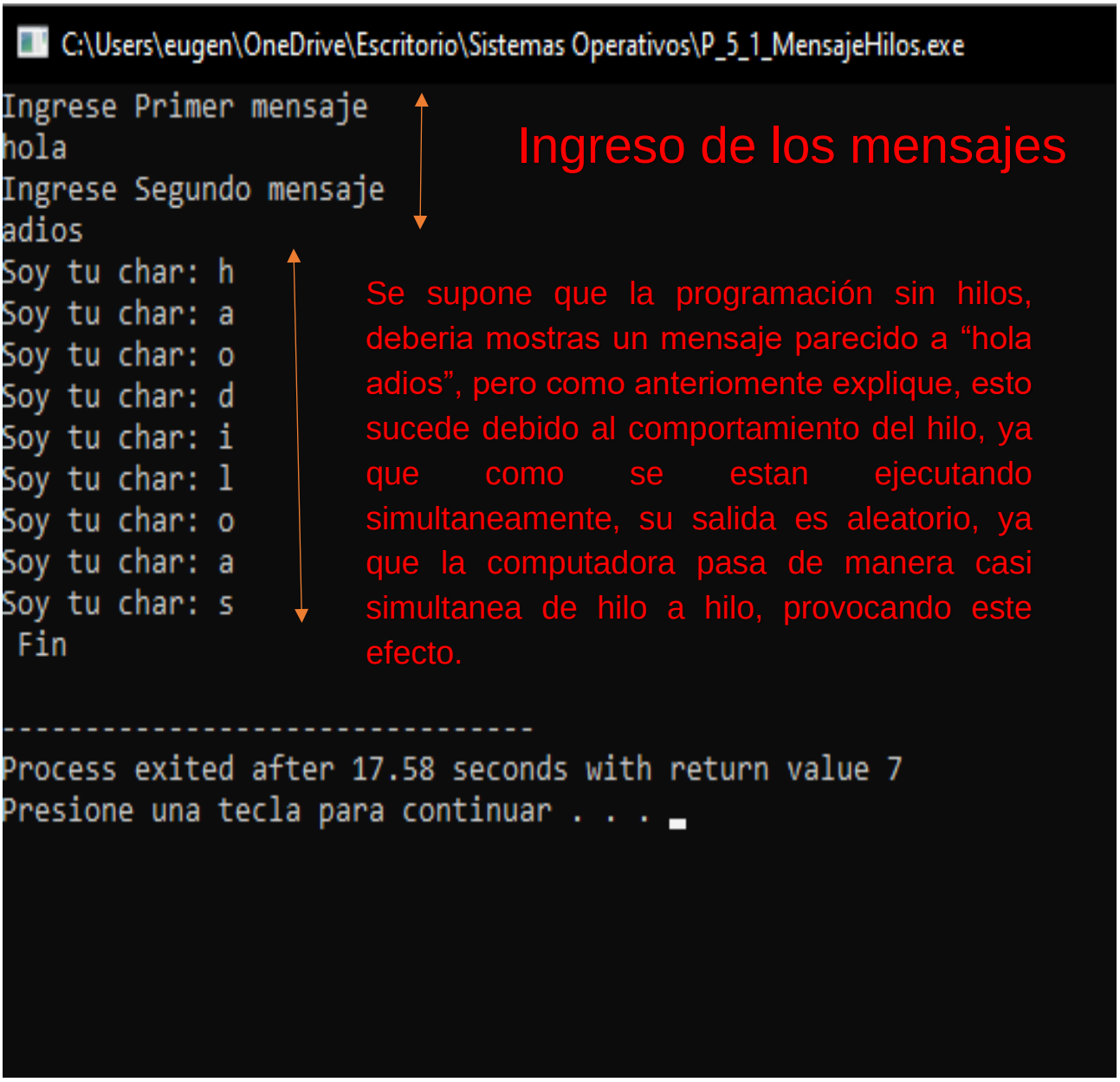
Indicaciones

- 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



Problema 2. _ Dado tres ingresados desde consola, encontrar el producto del producto de los dos primeros números, como suma sucesivas, las cuales se deben de dividir entre los hilos que el tercer número nos indique.

-A mi punto de vista este programa es el que mas dificultad tiene debido a la cantidad de aspectos que se deben de tomar en cuenta, ya que no es solo sumar, sino que analizar como los hilos se repartirán el trabajo de la suma y que al final, coincida la multiplicación de los dos primeros números con la suma de los trabajos de los hilos.

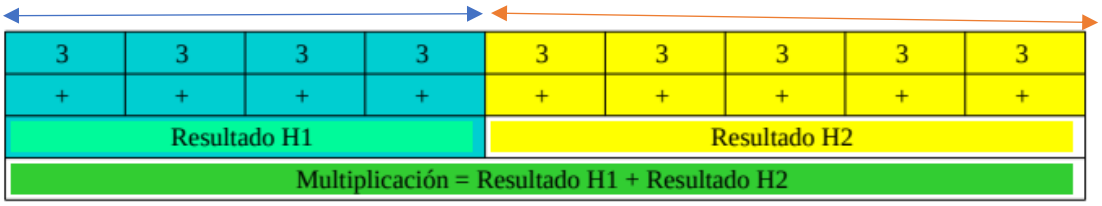
Aspectos a tomar en cuenta

- Ingresar datos desde consola.

-Verificar de cuantos dígitos será el número.
-Cuantos números ingresaras.
-Las banderas que permite leer el número desde consola.
-Extraer el numero de la consola y convertirlo a un tipo útil.

- Verificar que el dato E2 no sea mayor que N hilos
- Checar que el módulo de E2/N sea cero, ya que esto cambia cómo será el ciclo de trabajo de la suma que se hará en cada hilo.
  -Si el módulo es cero entonces el ciclo de trabajo en cada hilo será igual
  -Si es diferente a cero el ciclo de trabajo será igual hasta N-1, en N el ciclo de trabajo varia, y es debido a que la división de E2/N se toma como entero lo cual hace que los caracteres fraccionarios se trunquen, lo cual si no lo consideramos puede ocasionar errores.

A ESTO LLAMO CICLO DE TRABAJO DE CADA HILO



Las entradas era E1: 3 E2: 9 N:2 como E2/N su modulo no es cero, entonces observamos que el último ciclo de trabajo no es el mismo

- Verificar la sincronización de los hilos, ya que como se ejecutan simultáneamente y a veces ese comportamiento no es deseado, se debe meter un temporizador para que se ejecuten con ritmo.
- Al mandar los datos a la función es conveniente, crear una estructura, para que podamos pasarlos por referencia, lo que hace que su manipulación se más sencilla.
- Mostrar las veces que hará el ciclo de trabajo en nuestro hilo, para observar este comportamiento.

CÓDIGO C

```c
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
#include<unistd.h>
#include<pthread.h>
#include<math.h>

struct datos{
    int suma; //almacena la suma total de cada hilo
    int ciclo;//Veces en que E1 se sumara
    int E1;//Numero de E1 para que el valor que se sumara n ciclos
};
 int  hilo=0;
void* hilos_muestra(void* para)
{

struct datos* p = (struct datos*)para;
printf("E1= %d  se sumara: %d veces en este hilo\n ",p->E1, p->ciclo);

int i;
for(i=1;i<=(p->ciclo); ++i){
p->suma+= p->E1;
}

++hilo;

printf("HILO:%d Suma=  %d\n\n",hilo,(p->E1*p->ciclo));

}

int main(int argc,char *argv[])
{
    struct datos parametros;
    int N[3]; //usado para almacenar los datos que recibimos desde consola

int cad,contcad,reinicio;
char numeros[3][10];//el primer arreglo indica los numeros que vamos a recibir y el segundo son los digitos de cada numero

for(cad=0,contcad=0;argv[1][contcad] != 0;) {
    for(reinicio=0;argv[1][contcad]!='*';++reinicio)
    {
        numeros[cad][reinicio]= argv[1][contcad];

            ++contcad;
    }

        ++contcad;
       ++cad;

    // Use c as you need
}

int num;
for(num=0;num<3;++num){
  N[num]=atoi(numeros[num]);
}

printf("E1: %d E2: %d N: %d\n",N[0],N[1],N[2]);
```

Bibliotecas que utilizamos

Estructura con los datos que utiliza el hilo.(datos)

Funcion que muestra la suma de los hilos y realiza el “ciclo de trabajo” mostrando el aporte en cada hilo.(hilos_muestra)

Variables

Analiza los datos ingresados desde consola,guardandolo en un arreglo.

Convierte el arreglo que obtuvimos desde la consola a un tipo util (int) y muestra los datos que ingresamos.

```
61      while(N[2]>N[1])
62      {
63          printf("No se puede tener una cantidad de hilos mayor E2\n");
64          printf("Ingrese Numero hilos <E2\n");
65              scanf("%d",&N[2]);
66      }
67
68  /*Creacion de hilos*/
69     pthread_t thread[N[2]];
70
71
72  /*  Creamos un hilo  y  esta funcion genera un valor que usamos para identificar al hilo*/
73
74  int i=0;
75  double partent;
76  parametros.ciclo=N[1]/N[2];
77  parametros.E1=N[0];
78  parametros.suma=0;
79
80  while(i<N[2])
81  {
82              if((N[1]%N[2])==0)
83                 { //Si el modulo E1 y el numero de ciclos es par, esto significa que a cada hilo, le tocara un ciclo de trabajo igual
84                  parametros.ciclo=N[1]/N[2];//Ciclo de trabajo par
85                  pthread_create(&thread[i], NULL, &hilos_muestra,&parametros);
86                  usleep (10000) ;
87                 }else
88                   {//Si el modulo no fuera par, el ciclo de trabajo del hilo seria igual hasta i-1, en i su ciclo de trabajo se vuelve diferente
89                    if(i<N[2]-1)//Este if evalua a todo los N-1, siendo el ciclo de trabaja igual para cada uno
90                      {
91                       parametros.ciclo=N[1]/N[2];
92                       pthread_create(&thread[i], NULL, &hilos_muestra,&parametros);
93                       usleep (10000) ;//Es importante el tiempo para que alcancehacertoda la tarea
95                  }else//Cuando se rompa la condicion anterior osea que estemos en el ultimo ciclo, este se tiene un ciclo de trabajo diferente
96                      {
97                        parametros.ciclo=  parametros.ciclo + (N[2]*modf((double)N[1]/N[2], &partent)) + 1e-9; //el 1e-9 es un redondeo ya que la funcion fmod trabaja  con double y
98                        pthread_create(&thread[i], NULL, &hilos_muestra,&parametros);
99
100                     }
101             }
102     ++i;
103 }
104
105 i=0;
106 while(i<N[2]){
107   pthread_join(thread[i], NULL);
108   ++i;
109 }
110
111 printf("\n\n La suma total de los %d hilos es %d",N[2],parametros.suma);
112 exit(0);
113
114 }
115
```

Prueba que el numero E2 no sea mayor que N.

Variables utilzada en el analisis de las sumas.

Funcion engargada de dividir el trabajo de cada hilo dependiendo de su modulo.

Finaliza nuestros hilos

Muestra suma total

BIBLIOTECAS QUE UTILIZAMOS

Al igual que la anterior solo nos incluye los elementos que usaremos para la realización de nuestra práctica.

ESTRUCTURA CON LOS DATOS QUE UTILIZA EL HILO (DATOS)

Debido a que una estructura mantiene encapsulado los elementos que pongamos ahí, los cuales permanecen de manera estática, su uso resulta conveniente, podemos observar tres variables, que a continuación describo:

Suma: Se utiliza para almacenar las sumas de los ciclos de trabajo de cada hilo (la suma de los hilos).

Ciclo: Almacena el ciclo de trabajo de cada hilo, el cual se explicará mejor abajo.

E1: Es el factor que utiliza nuestro ciclo de trabajo (la suma sucesiva).

VARIABLES

Estas variables son utilizadas en la funcionalidad de la obtención de los datos que ingresamos desde la consola, a continuación, las describo.

N[3]: Utilizada para guardar a nuestros, números ingresados desde consola después de convertirlos al tipo conveniente.

cad: Utilizada dentro de nuestro arreglo numeros[cad][] para aumentar a nuestro arreglo bidimensional, cada vez que se encuentre la condición bandera, osea cada vez que se inrese un numero nuevo.

contcad: Uso parecido a la anterior variable, solo que ahora va recorriendo la cadena de nuestro arreglo que la consola nos proporciona con argv[1][contcad].

reinicio: Esta variable junto con la variable cad, nos sirven para ir almacenado correctamente los datos que nuestra consola nos proporciona, en este caso su uso es :

-Dentro del arreglo bidimencional es el tamaño que va a tener el digito numeros[][reinicio]

numeros[3][10] ó numeros[cad][reinicio]: Se utiliza para almacenar los datos que hemos ingresado desde consola, sin la condicion bandera, el primer numero 3, indica la cantidad de numeros que va a recibir (3) y el tamaño de cada numero.

ANALIZA LOS DATOS INGRESADOS DESDE CONSOLA,GUARDANDOLO EN UN ARREGLO

Esta es una parte fundamental de nuestro programa, la cual se debe poner demasiada atención para entenderle, hay tener en cuenta las siguientes indicaciones para su buen uso.

INDICACIONES

- Trabaja con las variables: cad, contcad, argv[1][contcad], reinicio, numeros[cad][reinicio].
- ,argv[1][contcad] aquí es don nuestra cadena que hemos proporcionado desde consola se almacena.
- numeros[cad][reinicio] almacena los datos sin el carácter bandera, cada número de longitud máxima (10) en una nueva posición del arreglo que controla la variable “cad”.
- Es importante que al momento de ejecutar el programa se ingrese los datos de esta manera
  -Linux: ./codigoobjeto num1*num2*num3*
  -Windows: codigoobjeto num1*num2*num3*

FUNCIONAMIENTO

Al momento de ingresar una cadena en la consola, por ejemplo 343*234*123*, se debe tomar en cuenta la cantidad máxima de la longitud de cada número, en este caso es 3, por lo que cumple con la longitud máxima de nuestro arreglo en general.

Esta cadena 343*234*123* se almacena en argv[1][contcad], podemos acceder a cada carácter si variamos la variable contcad, al final de esta cadena se encuentra un 0, lo cual indica la finalización de los datos ingresados, lo cual pusimos como condición el for

for(cad=0,contcad=0;argv[1][contcad] != 0;), la función de este for es recorrer la cadena que la consola nos proporciona.

Ósea que el primer ciclo estará en 3, luego en 4, hasta acabar el carácter 0.

Después tenemos a este for

-for(reinicio=0;argv[1][contcad]!='';++reinicio), el cual tiene funcionalidad de que cuando el carácter actual sea un ‘*’ acaba el ciclo, trabaja conjuntamente con estas líneas de código*

-numeros[cad][reinicio]= argv[1][contcad] esta línea nos sirve para guardar el carácter actual en donde se encuentre nuestra cadena, solo omite los caracteres ‘’.*

-++contcad nos sirve para ir recorriendo la cadena que se encuentra en argv[1][contcad]

-reinicio nos sirve para reiniciar a cero la posición de nuestro arreglo numeros[cad][reinicio], cada vez que aumente la variable cad.

Ejemplo:

Tenemos la cadena 343*234*123*

En nuestro arreglo *argv[1][contcad] se vería de esta manera*

3	4	3	*	2	3	4	*	1	2	3	*	0

En cada ciclo del for externo recorre esa cadena, y dentro el for interno el arreglo *números se visualiza de esta manera:*

Pero en general esta es la forma en que trabaja:

1er ciclo:

3	4	3	*	2	3	4	*	1	2	3	*	0

3									

Los demás recorridos del arreglo argv los hace el for interno con ++cont

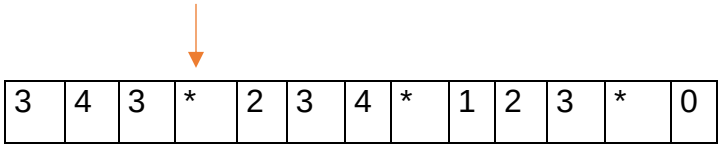
3	4	3	*	2	3	4	*	1	2	3	*	0

3	4	3	*	2	3	4	*	1	2	3	*	0

3	4								

3	4	3							

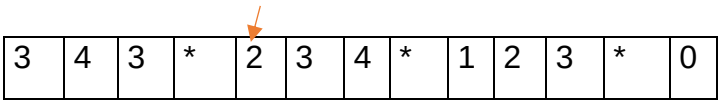
Cuando se encuentra a un carácter asterisco, ya no lo guardaría en el arreglo números, rompe el ciclo del for interno.



3	4	3	*	2	3	4	*	1	2	3	*	0

Y cuando sale del for interno se encuentra con estas líneas de código:

++contcad recorre la posción de nuestro arreglo

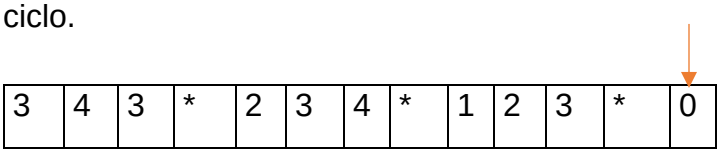


3	4	3	*	2	3	4	*	1	2	3	*	0

++cad aumenta la posición de nuestro arreglo bidimensional, lo cual hace disponible, las casillas a la derecha.

3	4	3							
D	i	s	p	o	n	i	b	l	e

El procedimiento que le sigue es el mismo hasta que llega al carácter 0 o nulo, en donde termina el ciclo.



3	4	3	*	2	3	4	*	1	2	3	*	0

Nuestro arreglo final quedaría de la siguiente manera

3	4	3							
2	3	4							
1	2	3							

CONVIERTE EL ARREGLO QUE OBTUVIMOS DESDE LA CONSOLA A UN TIPO UTIL (INT) Y MUESTRA LOS DATOS QUE INGRESAMOS.

Aquí utilizamos el arreglo números para convertirlo a un tipo más útil en este caso la función atoi resulta de mucha utilidad, ya que convierte un apuntador a char que es la dirección de una cadena a un número tipo int; por tanto nosotro al hacer esto numeros[num], le estamos pasando las direcciones de nuestras cadenas.

3	4	3							
2	3	4							
1	2	3							

atoi → N[0]=342

atoi → N[1]=234

atoi → N[2]=123

Por ultimo solo imprime los valores int con :

```
printf("E1: %d E2: %d N: %d\n",N[0],N[1],N[2]);
```

PRUEBA QUE EL NUMERO N NO SEA MAYOR QUE E2

Esto es muy importante debido que si no cumple esta condición E2/N siempre dará un valor 0 lo cual hace que siempre su ciclo de trabajo siempre sea 0, lo que indica que siempre sumara 0.

VARIABLES UTILZADA EN EL ANALISIS DE LAS SUMAS

Estas variables se utilizan en la parte mas importante de nuestro código la cual hace que se divida el ciclo de trabajo para cada hilo, a continuación, se describen:

i: *Variable que indica en que hilo se encuentra nuestro programa va desde 0…N.*

partent: *No es de tanta importancia en este programa, solo es un requisito que la función modf(), no pide para almacenar la parte entera de un número con decimales.*

parametros.ciclo

parametros.E1

parametros.suma

Variables previamente explicadas en la estructura datos, aquí solo las inicializamos, para no obtener valores erróneos durante su uso, de la siguiente manera:

parametros.ciclo=N[1]/N[2]; parametros.E1=N[0]; parametros.suma=0;

PARTE ENGARGADA DE DIVIDIR EL TRABAJO DE CADA HILO

Esta sección del código es la fundamental, para que el programa, realice correctamente su funcionalidad, se divide en las siguientes partes.

- Si el modulo de E2 y N es 0, entonces el ciclo de trabajo será igual en todos los hilos.

$$H1 \quad \dots \quad N$$

E1	E1	E1	E1	E1	E1	E1	E1

- Si el módulo de E2 y N es diferente a 0, el ciclo de trabajo será igual hasta N-1, en N el ciclo de traba es diferente y es dado por:

$Ciclo_{Hn} = Ciclo_{Hn-1} + (N * P_d)$ + 1e-9

$$Ciclo_{Hn} = Cantidad\ el\ ciclo\ de\ trabajo\ en\ el\ ultimo\ ciclo$$

$$Ciclo_{Hn-1} = Cantidad\ del\ ciclo\ de\ trabajo\ cuando\ el\ trabajo\ es\ el\ mismo\ en\ cada\ ciclo$$

$$Ciclo_{Hn} = Cantidad\ el\ ciclo\ de\ trabajo\ en\ el\ ultimo\ ciclo$$

$$N = cantidad\ de\ hilo\ ingresados$$

$$P_d = Parte\ decimal\ del\ ciclo\ de\ trabajo\ que\ se\ trunca, cuando\ se\ hace\ la\ division\ entera$$

$$1e-9 = Factor\ de\ correcion\ para\ que\ se\ redonde\ nuestro\ numero$$

Visualmente quedaría así nuestro ciclo de trabajo para cada hilo

$H_{1\dots\dots}H_{N-1} = parte\_enter(\frac{E2}{N})$							$H_N = Nparte\_decima(\frac{E2}{N})$
E1	E1	E1	E1	E1	E1	E1	E1

Esto queda demasiado claro en los if que están en el código en las líneas de código que está aquí:

-parametros.ciclo=N[1]/N[2], para ciclo de trabajo iguales

- parametros.ciclo= parametros.ciclo + (N[2]*modf((double)N[1]/N[2], &partent)) + 1e-9, para el último ciclo de trabajo, cuando el modulo es diferente a 0 y se encuentra en el ultimo hilo.

- En cada if y else se manda invocar el hilo pthread_create(&thread[i], NULL, &hilos_muestra,&parametros), con parámetros como la dirección de cada hilo(la cual cambia mientras i varia), atributos (en este caso los por defecto), la dirección de nuestra función que se describirá su funcionamiento abajo y por ultimo los parámetros que se encuentran en nuestra estructura.
- Hay que tomar en cuenta como son varios hilos que se están ejecutando casi simultáneos, es necesario que se le ponga un sleep para que cada hilo hago su tarea en el debido tiempo.

FUNCION QUE MUESTRA LA SUMA DE LOS HILOS Y REALIZA EL “CICLO DE TRABAJO” MOSTRANDO EL APORTE EN CADA HILO.(HILOS_MUESTRA)

Esta función solo se encarga de sumar todas las aportaciones de los hilos, en la variable que suma que se encuentra adentro de la estructura datos, lo cual hacemos aquí:

```
int i;

for(i=1;i<=(p->ciclo); ++i){

p->suma+= p->E1;

}
```

, mostrar cuantas veces realizara cada ciclo te trabajo, lo cual se encuentra aquí.

```
printf("E1= %d  se sumara: %d veces en este hilo\n ",p->E1, p->ciclo);
```

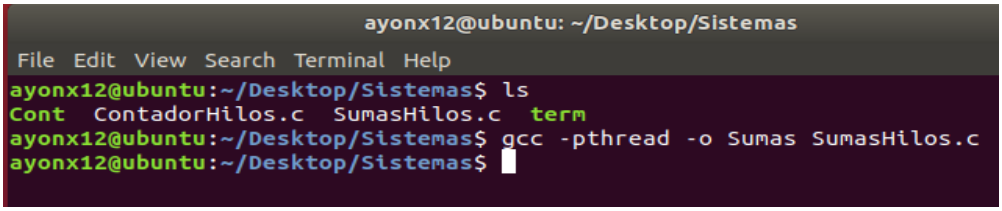
y por ultimo mostrar la aportación actual de cada hilo a la variable suma, que lo hacemos en esta parte:

```
printf("HILO:%d Suma=  %d\n\n",hilo,(p->E1*p->ciclo));
```

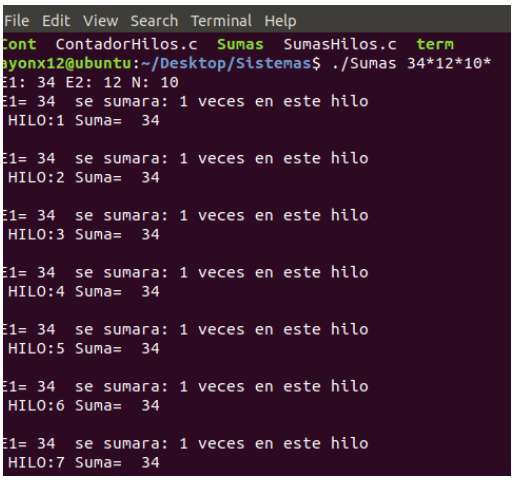
la variable hilo solo se utiliza para mostrar el hilo actual.

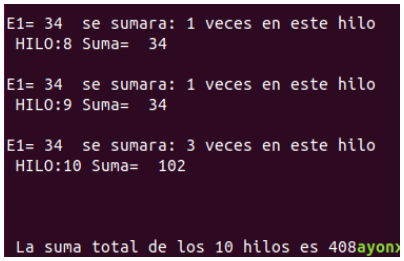
SALIDA EN PANTALLA DEL PROGRAMA

Compilación:

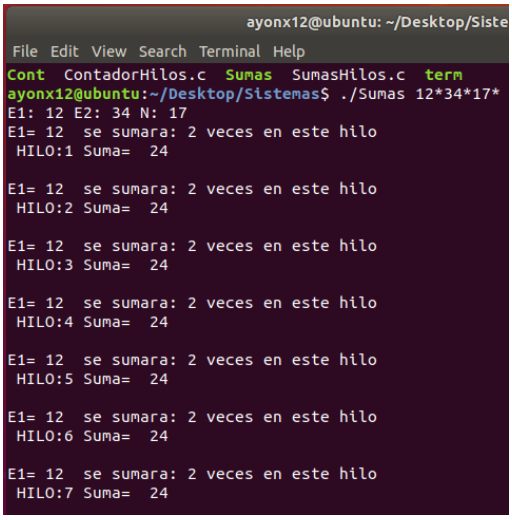


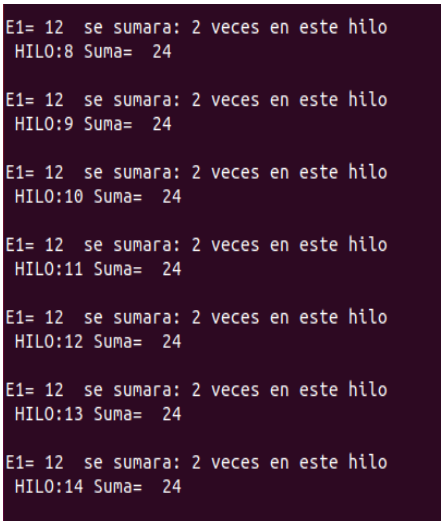
Ejecución con datos 34*12*10*

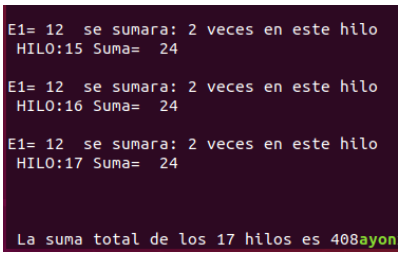




Ejecución con datos 12*34*17*







- No importa cuantos hilos metamos si los factores de la multiplicación dan el mismo resultado, el trabajo de los hilos realizara correctamente la suma.