

**INSTITUTO POLITÉCNICO NACIONAL.**

ESCUELA SUPERIOR DE CÓMPUTO.

**SISTEMAS OPERATIVOS.**

**PRÁCTICA 5**

Hilos

**Integrantes del equipo:**

* Chavarría Vázquez Luis Enrique.
* Juárez Espinoza Ulises.
* Machorro Vences Ricardo Alberto.
* Pastrana Torres Víctor Norberto.





**Unidad 2**

**2CM6**

Índice de contenido.

[**Glosario de términos.** 1](#_Toc55951868)

[**Hilos.** 1](#_Toc55951869)

[**Hilo de usuario.** 1](#_Toc55951870)

[**Hilos de kernel.** 1](#_Toc55951871)

[**Procesamiento asíncrono.** 1](#_Toc55951872)

[**Contenido (Investigación)** 2](#_Toc55951873)

[**Consideraciones.** 2](#_Toc55951874)

[**Importancia de los hilos.** 2](#_Toc55951875)

[**Tipos de hilos.** 2](#_Toc55951876)

[Hilos del usuario. 2](#_Toc55951877)

[Hilos del kernel. 3](#_Toc55951878)

[**Diferencia entre los hilos y los procesos** 3](#_Toc55951879)

[**Problema de usar hilos.** 4](#_Toc55951880)

[**Usos comunes de los hilos.** 4](#_Toc55951881)

[ Procesamiento asíncrono. 4](#_Toc55951882)

[ Aceleración de la ejecución. 4](#_Toc55951883)

[ Trabajo interactivo y en segundo plano. 4](#_Toc55951884)

[**Creación de hilos.** 4](#_Toc55951885)

[ Identificación de nuestro hilo. 4](#_Toc55951886)

[ Creación de nuestro hilo. 5](#_Toc55951887)

[**Gestión de hilos.** 6](#_Toc55951888)

[Modelos de subprocesos múltiples. 6](#_Toc55951889)

[Modelo de muchos a muchos. 6](#_Toc55951890)

[Modelo de muchos a uno. 7](#_Toc55951891)

[Modelo de uno a uno. 8](#_Toc55951892)

[**Códigos y ventanas de ejecución** 9](#_Toc55951893)

[**Programa51.c** 9](#_Toc55951894)

[**Código explicado por partes.** 9](#_Toc55951895)

[**Código completo.** 12](#_Toc55951896)

[**Explicación de manera global del código.** 14](#_Toc55951897)

[**Ejecución:** 14](#_Toc55951898)

[**Programa52.c** 16](#_Toc55951899)

[**Código explicado por partes.** 16](#_Toc55951900)

[**Código completo.** 16](#_Toc55951901)

[**Explicación de manera global del código.** 16](#_Toc55951902)

[**Ejecución:** 16](#_Toc55951903)

[**Programa53.c** 17](#_Toc55951904)

[**Código explicado por partes.** 17](#_Toc55951905)

[**Código completo.** 17](#_Toc55951906)

[**Explicación de manera global del código.** 17](#_Toc55951907)

[**Ejecución:** 17](#_Toc55951908)

[**Programa54.c** 18](#_Toc55951909)

[**Código explicado por partes.** 18](#_Toc55951910)

[**Código completo.** 18](#_Toc55951911)

[**Explicación de manera global del código.** 18](#_Toc55951912)

[**Ejecución:** 18](#_Toc55951913)

[**Programa55.c** 19](#_Toc55951914)

[**Código explicado por partes.** 19](#_Toc55951915)

[**Código completo.** 19](#_Toc55951916)

[**Explicación de manera global del código.** 19](#_Toc55951917)

[**Ejecución:** 19](#_Toc55951918)

[**Conclusiones.** 20](#_Toc55951919)

[**Chavarría Vázquez Luis Enrique.** 20](#_Toc55951920)

[**Juárez Espinoza Ulises.** 22](#_Toc55951921)

[**Machorro Vences Ricardo Alberto.** 22](#_Toc55951922)

[**Pastrana Torres Victor Norberto.** 22](#_Toc55951923)

[Bibliografía 23](#_Toc55951924)

Índice de figuras

[Ilustración 1 Creación de código. 5](file:///C:\Users\Luis%20Enrique\OneDrive\Escritorio\COVID\QUINTO%20SEMESTRE\S.O\PRACTICAS\Práctica%205_unidad2\Práctica5%201.0.docx#_Toc55951925)

[Ilustración 2 Código completo. 5](#_Toc55951926)

[Ilustración 3 Foto referente a lo explicado arriba de muchos a muchos 7](#_Toc55951927)

[Ilustración 4 Foto referente a lo explicado arriba de muchos a uno. 7](#_Toc55951928)

[Ilustración 5 Foto referente a lo explicado arriba de uno a uno 8](#_Toc55951929)

[Ilustración 6 Inclusión de bibliotecas y definición de la estructura. 9](#_Toc55951930)

[Ilustración 7 Generando las referencias a los almacenes de las variables. 10](#_Toc55951931)

[Ilustración 8 Validación e impresión de los valores de los 2 hilos. 11](#_Toc55951932)

[Ilustración 9 Impresión de los datos del HILO principal. 11](#_Toc55951933)

[Ilustración 10 Ejecución del programa programa52.c 15](#_Toc55951934)

Índice de tablas

[Tabla 1 Diferencias entre el nivel de usuario y nivel de kernel. 3](#_Toc55938178)

# **Glosario de términos.**

## **Hilos.**

Los hilos son básicamente una tarea que puede ser ejecutada en paralelo con otra tarea; teniendo en cuenta lo que es propio de cada hilo es el contador de programa, la pila de ejecución y el estado de la CPU (incluyendo el valor de los registros).

## **Hilo de usuario.**

En una aplicación ULT pura, todo el trabajo de gestión de hilos lo realiza la aplicación y el núcleo o kernel no es consciente de la existencia de hilos. Es posible programar una aplicación como multihilo mediante una biblioteca de hilos. La misma contiene el código para crear y destruir hilos, intercambiar mensajes y datos entre hilos, para planificar la ejecución de hilos y para salvar y restaurar el contexto de los hilos.

## **Hilos de kernel.**

Para cada hilo que existe en el espacio de usuario, hay un hilo del kernel correspondiente. Dado que estos subprocesos son administrados por el kernel, siguen una multitarea preventiva en la que el programador puede adelantarse a un subproceso en ejecución con un subproceso de mayor prioridad que está listo para la ejecución.

## **Procesamiento asíncrono.**

Un proceso asíncrono es un proceso o una función que ejecuta una tarea "en segundo plano" sin que el usuario tenga que esperar a que finalice la tarea.

# **Contenido (Investigación)**

## **Consideraciones.**

Un hilo de ejecución se considera a menudo como la unidad más pequeña de procesamiento en la que trabaja un planificador.

Un proceso puede tener varios subprocesos de ejecución que se ejecutan de forma asincrónica.

Esta ejecución asincrónica brinda la capacidad de cada hilo de manejar un trabajo o servicio en particular de forma independiente. Por lo tanto, varios subprocesos que se ejecutan en un proceso manejan sus servicios, lo que en general constituye la capacidad completa del proceso. [1]

## **Importancia de los hilos.**

Ahora, uno se preguntaría por qué necesitamos múltiples subprocesos en un proceso. ¿Por qué no se puede utilizar un proceso con un solo hilo principal (predeterminado) en cada situación?

Supongamos que hay un proceso, que recibe entradas en tiempo real y corresponde a cada entrada que tiene para producir una determinada salida. Ahora bien, si el proceso no tiene varios subprocesos, es decir, si el proceso no incluye varios subprocesos, todo el procesamiento del proceso se vuelve sincrónico. Esto significa que el proceso toma una entrada, la procesa y produce una salida.

La limitación en el diseño anterior es que el proceso no puede aceptar una entrada hasta que haya terminado de procesar la anterior y, en caso de que el procesamiento de una entrada tarde más de lo esperado, la aceptación de más entradas queda en espera.

Para considerar el impacto de la limitación anterior, si asignamos el ejemplo genérico anterior con un proceso de servidor de socket que puede aceptar la conexión de entrada, procese y proporcione al cliente de socket la salida. Ahora, si al procesar cualquier entrada, si el proceso del servidor toma más tiempo del esperado y mientras tanto, otra entrada (solicitud de conexión) llega al servidor de socket, entonces el proceso del servidor no podrá aceptar la nueva conexión de entrada ya que ya está bloqueada. procesando la antigua conexión de entrada. Esto puede dar lugar a un tiempo de espera de conexión en el cliente de socket que no se desea en absoluto. [2]

Esto muestra que el modelo de ejecución sincrónico no se puede aplicar en todas partes y, por lo tanto, se sintió el requisito del modelo de ejecución asincrónico que se implementa mediante el uso de subprocesos.

## **Tipos de hilos.**

### Hilos del usuario.

Estos subprocesos no son conocidos por el kernel y, por lo tanto, el kernel no está involucrado en su procesamiento. Estos subprocesos siguen la multitarea cooperativa en la que un subproceso libera la CPU por su propio deseo, es decir, el planificador no puede apropiarse del subproceso. [3]

### Hilos del kernel.

Para cada hilo que existe en el espacio de usuario, hay un hilo del kernel correspondiente. Dado que estos subprocesos son administrados por el kernel, siguen una multitarea preventiva en la que el programador puede adelantarse a un subproceso en ejecución con un subproceso de mayor prioridad que está listo para la ejecución. [3]

En la siguiente tabla podemos ver precisamente la comparativa de todo lo que hemos mencionado los puntos anteriores de manera un poco más desmenuzada y por tanto más simplificada.

Tabla Diferencias entre el nivel de usuario y nivel de kernel.

|  |  |  |
| --- | --- | --- |
| # | Hilos de nivel de usuario | A nivel de kernel |
| 1 | Los subprocesos a nivel de usuario son más rápidos de crear y administrar. | Los hilos a nivel de kernel son más lentos de crear y administrar. |
| 2 | La implementación se realiza mediante una biblioteca de subprocesos a nivel de usuario. | El sistema operativo admite la creación de hilos del kernel. |
| 3 | El hilo a nivel de usuario es genérico y puede ejecutarse en cualquier sistema operativo. | El hilo de nivel de kernel es específico del sistema operativo. |
| 4 | Las aplicaciones multiproceso no pueden aprovechar el multiprocesamiento. | Las propias rutinas del kernel pueden ser multiproceso. |

## **Diferencia entre los hilos y los procesos**

Mostraremos algunas de las principales diferencias entre el hilo y los procesos: [4]

* Los procesos no comparten su espacio de direcciones, mientras que los subprocesos que se ejecutan bajo el mismo proceso comparten el espacio de direcciones.
* Desde el punto anterior, está claro que los procesos se ejecutan de forma independiente entre sí y la sincronización entre procesos la realiza el kernel solo, mientras que, por otro lado, la sincronización de subprocesos debe ser atendida por el proceso bajo el cual se ejecutan los subprocesos.
* El cambio de contexto entre hilos es rápido en comparación con el cambio de contexto entre procesos.
* La interacción entre dos procesos se logra solo a través de la comunicación estándar entre procesos, mientras que los subprocesos que se ejecutan bajo el mismo proceso pueden comunicarse fácilmente ya que comparten la mayoría de los recursos como la memoria, el segmento de texto, etc.

## **Problema de usar hilos.**

Muchos sistemas operativos no implementan subprocesos como procesos, sino que los ven como parte del proceso principal. Considerando otro posible problema podría surgir son los problemas de concurrencia. Dado que los subprocesos comparten todos los segmentos (excepto el segmento de la pila) y el programador puede adelantarlos en cualquier etapa, cualquier variable global o estructura de datos que pueda dejarse en un estado inconsistente al adelantar un subproceso podría causar problemas graves cuando la siguiente prioridad alta thread ejecuta la misma función y usa las mismas variables o estructuras de datos. [5]

## **Usos comunes de los hilos.**

Acorde con lo encontrado en [6] podemos ver los diversos usos que tienen los hilos.

### Procesamiento asíncrono.

Un ejemplo es como los software de procesamiento de texto guardan archivos temporales cuando se esta trabajando en dicho programa. Se crea un hilo que tiene como función guardar una copia de respaldo mientras se continúa con la operación de escritura por el usuario sin interferir en la misma.

### Aceleración de la ejecución.

Se ejecutan lotes de código de forma totalmente simultanea.

### Trabajo interactivo y en segundo plano.

En un programa de hoja de cálculo un hilo puede estar visualizando los menús y leer la entrada del usuario mientras que otro hilo ejecuta las órdenes y actualiza la hoja de cálculo.

* Estructuración modular de los programas.

Puede ser un mecanismo eficiente para un programa que ejecuta una gran variedad de actividades, teniendo las mismas bien separadas mediante a hilos que realizan cada una de ellas.

## **Creación de hilos.**

### Identificación de nuestro hilo.

Así como un proceso se identifica a través de un ID de proceso, un hilo se identifica mediante un ID de hilo. Pero curiosamente, la similitud entre los dos termina aquí. [2]

1. Un ID de proceso es único en todo el sistema, mientras que un ID de subproceso es único solo en el contexto de un único proceso.
2. Un ID de proceso es un valor entero, pero el ID de subproceso no es necesariamente un valor entero. Bien podría ser una estructura.
3. Un ID de proceso se puede imprimir muy fácilmente, mientras que un ID de hilo no es fácil de imprimir.

### Creación de nuestro hilo.

Basado en lo encontrado en [7], normalmente, cuando un programa se inicia y se convierte en un proceso, comienza con un hilo predeterminado. Entonces podemos decir que cada proceso tiene al menos un hilo de control. Un proceso puede crear subprocesos adicionales utilizando la siguiente función:



Ilustración Creación de código.

El primer argumento es una dirección de tipo pthread\_t. Una vez que la función se llama con éxito, la variable cuya dirección se pasa como primer argumento contendrá el ID del hilo del hilo recién creado. El segundo argumento puede contener ciertos atributos que queremos que contenga el nuevo hilo. Podría ser una prioridad, etc. El tercer argumento es un puntero de función. Esto es algo a tener en cuenta que cada hilo comienza con una función y que la dirección de las funciones se pasa aquí como tercer argumento para que el kernel sepa desde qué función iniciar el hilo. Como la función (cuya dirección se pasa en el tercer argumento anterior) puede aceptar algunos argumentos también, podemos pasar estos argumentos en forma de puntero a un tipo void.

Ahora procedemos a aplicar lo visto en el siguiente programa.



Ilustración Código completo.

De manera resumida, lo que el programa mostrado en la parte superior hace es lo siguiente.

1. Utiliza la función pthread\_create () para crear dos hilos.
2. La función de inicio para ambos hilos se mantiene igual.
3. Dentro de la función 'doSomeThing ()', el hilo usa las funciones pthread\_self () y pthread\_equal () para identificar si el hilo en ejecución es el primero o el segundo creado.
4. Además, dentro de la misma función 'doSomeThing ()' se ejecuta un bucle for para simular un trabajo que consume mucho tiempo.

Ahora bien ejecutan el código tnemos lo siguiente.

./threads

Hilo creado de manera satisfactoria.

Primer hilo procesado.

Hilo creado de manera satisfactoria.

Segundo hilo procesado.

## **Gestión de hilos.**

### Modelos de subprocesos múltiples.

Algunos sistemas operativos proporcionan un subproceso de nivel de usuario combinado y un servicio de subproceso de nivel de kernel. Solaris es un buen ejemplo de este enfoque combinado. En un sistema combinado, varios subprocesos dentro de la misma aplicación pueden ejecutarse en paralelo en varios procesadores y una llamada al sistema de bloqueo no necesita bloquear todo el proceso. Los modelos de subprocesos múltiples son de tres tipos [8]:

* Relación de muchos a muchos.
* Relación de muchos a uno.
* Relación uno a uno.

### Modelo de muchos a muchos.

El modelo de muchos a muchos multiplexa cualquier número de subprocesos de usuario en un número igual o menor de subprocesos del núcleo [8].

El siguiente diagrama muestra el modelo de subprocesos de muchos a muchos donde 6 subprocesos a nivel de usuario se multiplexan con 6 subprocesos a nivel de kernel. En este modelo, los desarrolladores pueden crear tantos subprocesos de usuario como sea necesario y los subprocesos de Kernel correspondientes pueden ejecutarse en paralelo en una máquina multiprocesador. Este modelo proporciona la mejor precisión en la concurrencia y cuando un hilo realiza una llamada al sistema de bloqueo, el kernel puede programar otro hilo para su ejecución. El asiento info tomemos el ejemplo de lo que un mencionado.

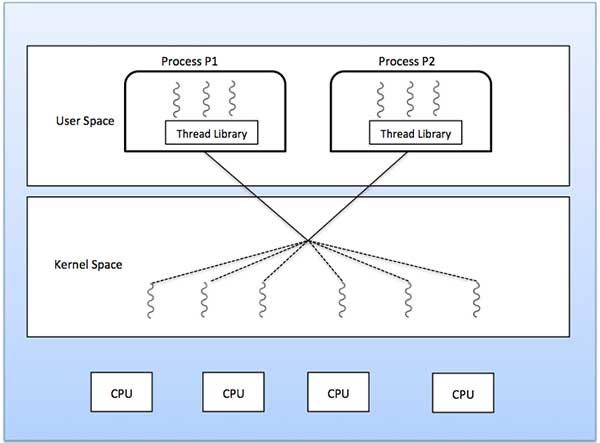


Ilustración Foto referente a lo explicado arriba de muchos a muchos

### Modelo de muchos a uno.

El modelo de varios a uno asigna muchos subprocesos de nivel de usuario a un subproceso de nivel de Kernel. La gestión de subprocesos se realiza en el espacio del usuario mediante la biblioteca de subprocesos. Cuando el hilo realiza una llamada al sistema de bloqueo, todo el proceso se bloqueará. Solo un subproceso puede acceder al Kernel a la vez, por lo que varios subprocesos no pueden ejecutarse en paralelo en multiprocesadores. [8]

Si las bibliotecas de subprocesos a nivel de usuario se implementan en el sistema operativo de tal manera que el sistema no las admite, entonces los subprocesos del Kernel utilizan los modos de relación de muchos a uno. La siguiente foto vemos el ejemplo de lo mencionado.

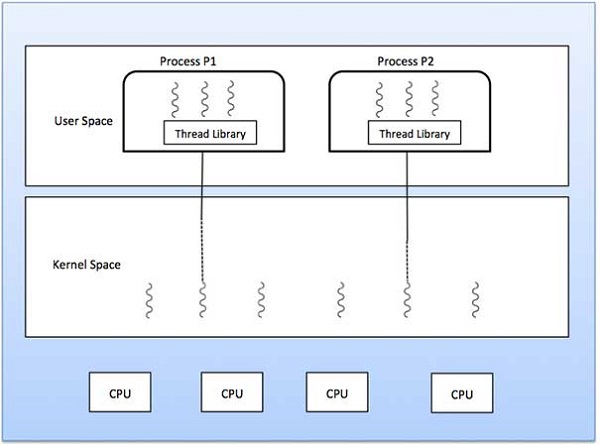


Ilustración Foto referente a lo explicado arriba de muchos a uno.

### Modelo de uno a uno.

Existe una relación de uno a uno entre el subproceso a nivel de usuario y el subproceso a nivel de kernel. Este modelo proporciona más simultaneidad que el modelo de varios a uno. También permite que se ejecute otro hilo cuando un hilo realiza una llamada al sistema de bloqueo. Admite múltiples subprocesos para ejecutarse en paralelo en microprocesadores. [8]

La desventaja de este modelo es que la creación de un hilo de usuario requiere el hilo de Kernel correspondiente. OS / 2, Windows NT y Windows 2000 utilizan un modelo de relación uno a uno. En la siguiente foto vemos el ejemplo de lo que hemos mencionado.

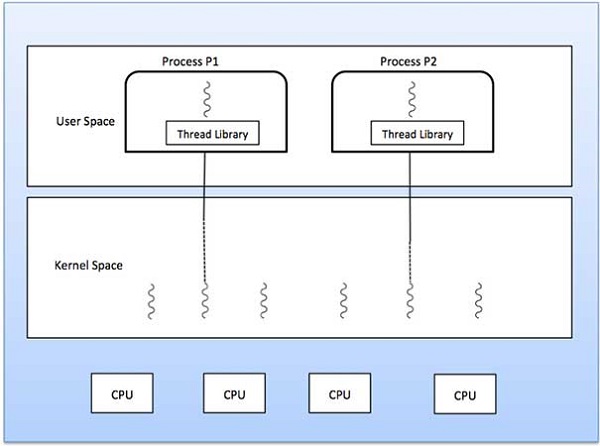


Ilustración Foto referente a lo explicado arriba de uno a uno

El

# **Códigos y ventanas de ejecución**

## **Programa51.c**

Realizar un programa que inicie un hilo principal que a su vez crea dos hilos. El hilo principal espera hasta que ambos hilos terminen y después finaliza. Los hilos sólo deben de mostrar algún mensaje en pantalla y terminar.

### **Código explicado por partes.**

En esta primera parte del código podemos ver como se hicieron todas las importaciones de las librerías necesarias para la ejecución del programa, dentro de las cuales destacan una que se emplea para el uso de precisamente la implementación de los hilos en nuestro código, también tenemos otro para el manejo de cadenas de caracteres y algunas del sistema. Cabe destacar que también he hecho una definición de GNU\_SOURCE el cual nos ayuda a acceder a diversas funciones que son omitidas en POSIX para el manejo precisamente de los hilos y nos permite el acceso algunas funcionalidades de bajo nivel dentro de nuestro sistema operativo.

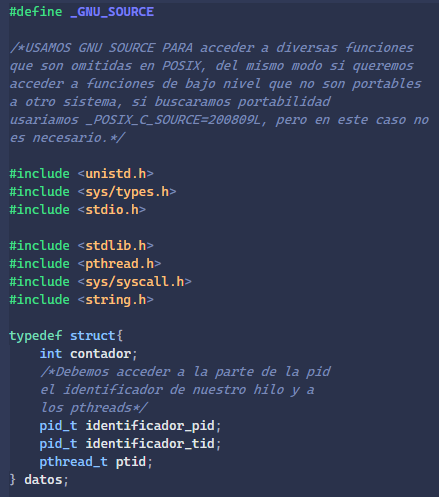


Ilustración Inclusión de bibliotecas y definición de la estructura.

Ahora bien, también la primera imagen se muestra como se hizo la definición de nuestra estructura en la cual vamos a ingresar los datos básicos como un contador, y los identificadores de cada uno de los hilos que en mostraremos esto con la finalidad de poder recordar y visualizar en consola que hilo corresponde acorde con su identificador.

También en ese de mencionar qué necesitamos precisamente apuntar a dichos identificadores como para lo cual hemos implementado el siguiente código.

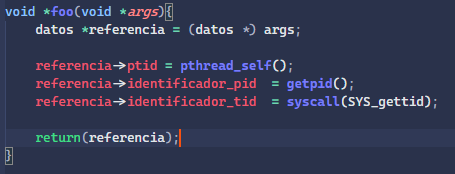


Ilustración Generando las referencias a los almacenes de las variables.

Ahora ya dentro de nuestra función para poder mostrar los hilos, necesitamos primero que nada ingresar a la memoria dinámica y posteriormente por medio de un ciclo iremos realizando un conteo y un futuro es iteraciones para poder acceder a número de hilos que nosotros hemos definido en un comienzo, ya que este programa nos ayuda a poder determinar la cantidad de ideas que queremos que se muestra en pantalla y en nuestro caso se trata solamente de dos hilos derivados.

Cabe destacar que en esta sección básicamente se hicieron toda la validaciones como por ejemplo en caso de que no tengamos una definición de los creada lo mostro mensaje de error, pero también como lo es el caso al detener una definición de la cantidad de hilos que queremos entonces se limitará a mostrarnos simplemente el el mensaje que nosotros queremos con información que nosotros le estamos pidiendo.

Al estar trabajando directamente lenguaje de programación C, en la impresión directamente se pusieron los símbolos de carácter guión con la finalidad de poder mantener todo la consola un poco más organizado y no confundir los datos al momento del impresión, desta manera tenerlos bien identificados en todo momento.

Queremos recalcar que hemos ingresado estos datos y los humos impresiona terminan de el sistema operativo por el simple motivo de que es imprescindible que tengamos bien identificados a cada uno de los hilos y desde luego también tenemos datos del padre para poder verificar que en efecto esto serán derivados de un hilo principal.

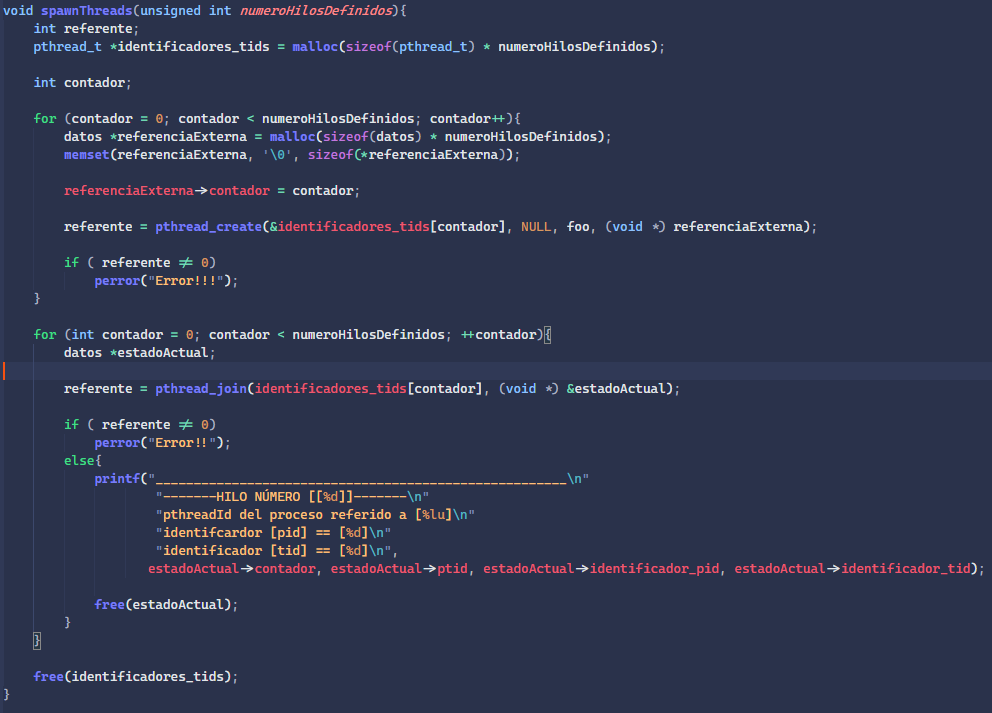


Ilustración Validación e impresión de los valores de los 2 hilos.

Finalmente en nuestro main, basta con que haga monde impresión del hilo principal, lo cual es bastante sencillo porque simplemente tenemos que hace referencia a ellos en la impresión respetando la estructura que hemos nosotros definido y por último simplemente hacemos la llamada a nuestra función para hacer aparecer los hilos que nosotros queramos, es importante recalcar que debemos pasar del parámetro de la cantidad de hilos que queramos generar partiendo del primer hilo, lo bonito de este código es que nos permite y nos da la posibilidad de crear una cantidad n de hilos lo cual está bastante bien si es que queremos ver como es que se generan y poder entender mucho mejor pues mucho más de sus datos el información que el programa que hemos creado nos proporciona.

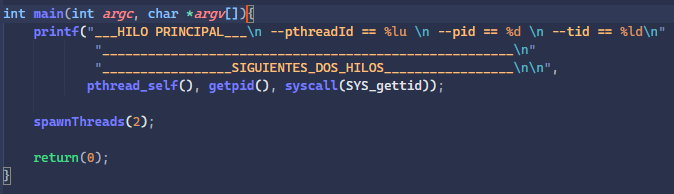


Ilustración Impresión de los datos del HILO principal.

### **Código completo.**

Es destacable que este nuestro código completo, en formato de texto puesto dentro de nuestra práctica pero la explicación general del código se encuentre la parte de abajo, aunque ya lo hemos desmenuzado la sección derribar la cual la modificación de código explicado por partes.

#define **\_GNU\_SOURCE**

*/\*USAMOS GNU SOURCE PARA acceder a diversas funciones*

*que son omitidas en POSIX, del mismo modo si queremos*

*acceder a funciones de bajo nivel que no son portables*

*a otro sistema, si buscaramos portabilidad*

*usariamos \_POSIX\_C\_SOURCE=200809L, pero en este caso no*

*es necesario.\*/*

#include <unistd.h>

#include <sys/types.h>

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <sys/syscall.h>

#include <string.h>

typedef struct{

    int contador;

*/\*Debemos acceder a la parte de la pid*

*el identificador de nuestro hilo y a*

*los pthreads\*/*

    pid\_t identificador\_pid;

    pid\_t identificador\_tid;

    pthread\_t ptid;

} data;

void \***foo**(void \**args*){

    data \*referencia = (data \*) args;

    referencia->ptid = **pthread\_self**();

    referencia->identificador\_pid  = **getpid**();

    referencia->identificador\_tid  = **syscall**(SYS\_gettid);

    return(referencia);

}

void **spawnThreads**(unsigned int *numeroHilosDefinidos*)

{

    int referente;

    pthread\_t \*identificadores\_tids = **malloc**(sizeof(pthread\_t) \* numeroHilosDefinidos);

    int contador;

    for (contador = 0; contador < numeroHilosDefinidos; contador++)

    {

        data \*referenciaExterna = **malloc**(sizeof(data) \* numeroHilosDefinidos);

**memset**(referenciaExterna, '\0', sizeof(\*referenciaExterna));

        referenciaExterna->contador = contador;

        referente = **pthread\_create**(&identificadores\_tids[contador], NULL, foo, (void \*) referenciaExterna);

        if ( referente != 0)

**perror**("Error!!!");

    }

    for (int contador = 0; contador < numeroHilosDefinidos; ++contador)

    {

        data \*estadoActual;

        referente = **pthread\_join**(identificadores\_tids[contador], (void \*) &estadoActual);

        if ( referente != 0)

**perror**("Error!!");

        else

        {

**printf**("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n"

                    "-------HILO NÚMERO [[%d]]-------\n"

                    "pthreadId del proceso referido a [%lu]\n"

                    "identifcardor [pid] == [%d]\n"

                    "identificador [tid] == [%d]\n",

                   estadoActual->contador, estadoActual->ptid, estadoActual->identificador\_pid, estadoActual->identificador\_tid);

**free**(estadoActual);

        }

    }

**free**(identificadores\_tids);

}

int **main**(int *argc*, char \**argv*[])

{

**printf**("\_\_\_HILO PRINCIPAL\_\_\_\n --pthreadId == %lu \n --pid == %d \n --tid == %ld\n"

            "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n"

            "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_SIGUIENTES\_DOS\_HILOS\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n\n",

**pthread\_self**(), **getpid**(), **syscall**(SYS\_gettid));

**spawnThreads**(2);

    return(0);

}

### **Explicación de manera global del código.**

Como se mencionó anteriormente en nuestro código, lo primero que se hace es la implementación de las diversas librerías y desde luego de una definición especial para poder tener acceso funciones de bajo nivel y también poder manejar algunas funcionalidades omitidas en POSIX, todo con la mera finalidad de poder trabajar mucho mejor con nuestros hilos. Es bastante interesante ver también que en no solamente se puede crear un programa estático en el cual se generan dos hilos y ya esta, sino que también es posible como fue nuestro caso crear un programa que tienen una cantidad determinada por el mismo usuario de hilos, lo cual es bastante interesante porque ya así no mejoras a la implementación que nosotros deseamos podemos inclusive pedir al usuario que no se diera una cantidad por medio de la terminal y entonces demostrará en un ciclo cierta cantidad de y dos en una determinada cantidad ocasiones mientras se lo decida con un menú y un interfaz más detallada, pero en este caso nuestra finalidad era mostrar simplemente como en el funcionamiento de un hilo principal y dos hilos secundarios, tal cual estaba pedido los requerimientos de la práctica. Nos aseguramos de que el mensaje que mostraron estos signos fuera precisamente mensajes con información de alto valor los cuales son identificadores que nos permite entender precisamente de dónde vienen y también poder tener bien visualizado cuáles son esos hilos porque supongamos que queremos hacer que este programa se pudiera utilizar para genera mucho más hilos desde luego que tendríamos que llevar un control mucho más detallado de cada uno de los valores identificadores que estos tienen dentro de nuestro sistema operativo.

### **Ejecución:**

En la siguiente imagen podemos apreciar la ejecución de nuestro código, a nosotros como equipo nos llamó bastante la atención como es que se compilar los programas de hilos ya que es muy interesante ver que en realidad no solamente basta con hacer nuestra compilación de forma tradicional como lo haríamos en este lenguaje programación, sino que además de ello tenemos que ingresar la directiva -lpthread para indicarle de manera directa al sistema que estaremos trabajando precisamente con hilos los cuales son por así decirlo funciones de un tanto bajo nivel, por lo que es necesario indicarlo además de que también estamos utilizando cierta tuberías que nos permiten el manejo de los mismos por lo cual es importante que lo hagamos esta forma, lo que ocurre sino usamos dicha directiva es que simplemente nuestro programa nos va compilar o posiblemente nos de un archivo un formato .out el cual tendrá que ser ejecutado con esa terminación, pero nuestra intención es tener el ejecutable directamente con el código compilado de forma tradicional como lo tenemos en un programa común y corriente C, entonces ya entendido esto ahora sí procedemos explica que es lo que estamos viendo en la terminal.

Principalmente tenemos la impresión de nuestro primer hilo el cual evidentemente tiene una id y única, pero también tiene un pid y tid que lo hacen también único, esto es porque se trata del hilo principal pero cuando nosotros vemos los otros dos hilos que hemos definido los cuales ya se han ingresado por medio del parámetro que le pasamos a la función de la creación de hilos el cual fue dos, podemos ver que precisamente el pid coincide con el del hilo principal y a su vez los otros identificadores simplemente que no es un porque están ocupando por así decirlo un espacio diferente dentro del sistema operativo, también cabe destacar que cada uno de los hilos tiene un identificador referido único.

Como lo mencionamos hace unos momentos es bastante interesante porque este programa que hemos arrollado sí lo pensamos lo podemos escalada muchas más cosas o inclusive tener una gestión mucho más eficiente de una cantidad ilimitada y dos, claro está ilimitada en términos de los recursos que nuestra computadora nos permita.

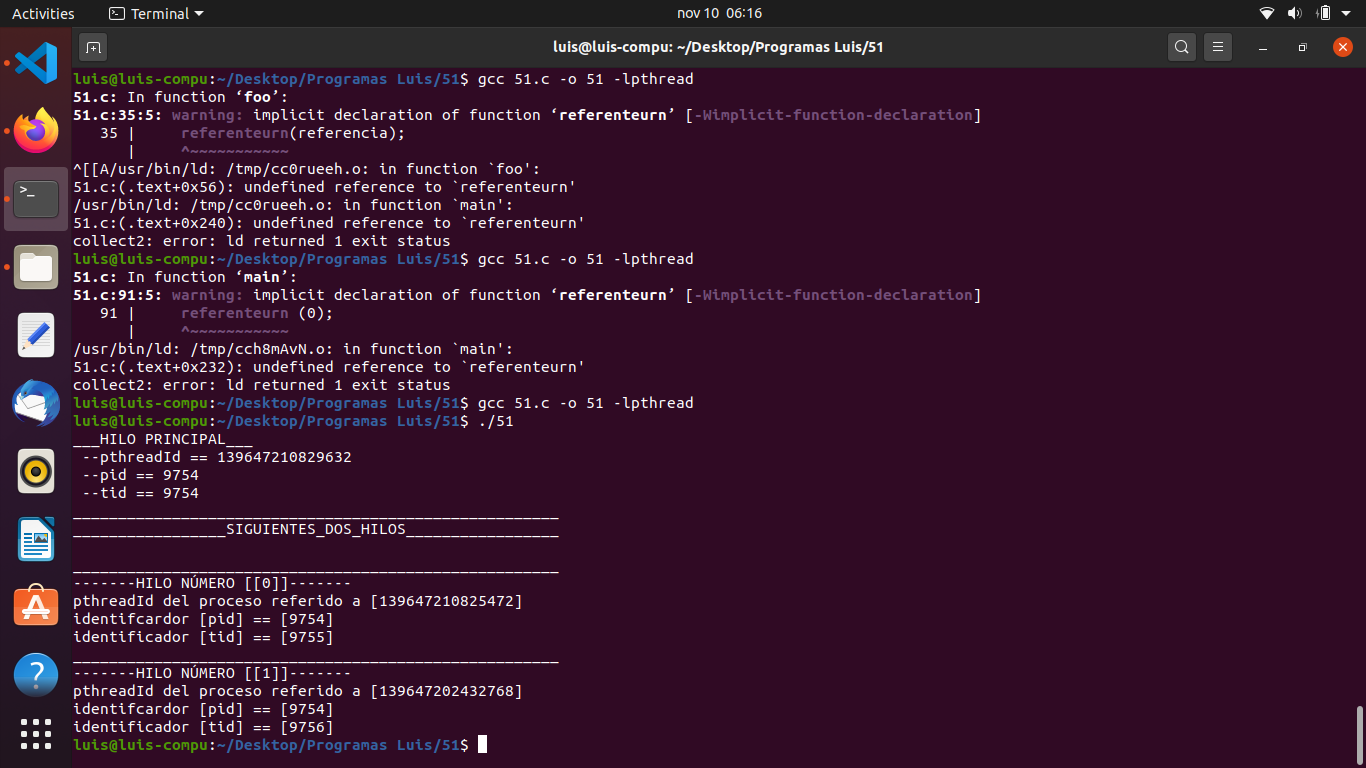


Ilustración Ejecución del programa programa52.c

## **Programa52.c**

Realizar un programa con una variable entera global (fuera de main()) con un valor inicial de cero. Crear un hilo que incremente la variable global en a unidades y crear otro hilo que la disminuya en b unidades. Al final el hilo principal imprimará el valor de la variable global.

### **Código explicado por partes.**

### **Código completo.**

### **Explicación de manera global del código.**

### **Ejecución:**

## **Programa53.c**

Realizar un programa cree un proceso hijo que a su vez creará tres hilos. Cada uno de los tres hilos creará dos hilos más. Cada uno de los hilos creados imprimirá en pantalla sus identificadores.

### **Código explicado por partes.**

### **Código completo.**

### **Explicación de manera global del código.**

### **Ejecución:**

## **Programa54.c**

Realizar un programa que cree tres hilos. El primer hilo se encargará de contabilizar las ocurrencias de una cadena dentro de un archivo especifico y devolver el resultado al programa principal; el segundo hilo copiará los archivos de su directorio actual a un subdirectorio que usted elija devolviendo al programa principal el número de archivos copiados; el tercer hilo generará un archivo donde se reportarán los resultados devueltos por los otros dos hilos.

### **Código explicado por partes.**

### **Código completo.**

### **Explicación de manera global del código.**

### **Ejecución:**

## **Programa55.c**

Realizar un programa donde un hilo se encargará de decir si un número entero dado por el usuario es compuesto, si lo es, otro hilo se encargará de descomponerlo en sus números primos, si no lo es, otro hilo se encargará de decir que si es primo.

### **Código explicado por partes.**

### **Código completo.**

### **Explicación de manera global del código.**

### **Ejecución:**

# **Conclusiones.**

## **Chavarría Vázquez Luis Enrique.**

Esta práctica fue de suma relevancia porque consiguiente entender de forma clara la importancia que tienen los hilos en el desarrollo de software, no solamente en los sistemas operativos ya que tienen un montón de aplicaciones el industria te las cuales muchas veces nosotros ni siquiera somos conscientes y que día a día nos permiten utilizar programas tan fundamentales en nuestra cotidianidad que a veces damos por hecho su uso y ni siquiera pensamos en todo lo que hay detrás de ellos.

Uno de sus usos que mí me sorprendió bastante es precisamente la parte de procesamiento asíncrono, que en software como lo son procesadores de texto o u otro software de aplicación como las herramientas para desarrollo de las cuales podemos escribir código, tienen un algoritmo que precisamente se basa en el uso de hilos para generar archivos de guardado temporal de lo que hemos estado trabajando, es super importante esto porque de hecho permite muchas más facilidades al momento de no solamente desplegar la información en pantalla, sino que también tiene respaldos de dicha información en caso de que posibles fallos o posibles errores no sólo dentro del sistema sino también del usuario podía ser ocurrir entonces ofrecer cierta capa de seguridad y de ser test al usuario de que se información se mantendrá salvo.

Eso sólo por arrancar también tenemos otros usos bastante interesantes como lo es la aceleración de la ejecución de los programas, debemos preguntarnos qué es lo que podría pasar en caso de que estamos arrollando un software que realmente ponga el límite las capacidades de algun dispositivo y entonces al no poder modificar el dispositivo de los usuarios o el dispositivo connect contamos para ejecutar dicho programa entonces se los hilos pueden ser de mucha ayuda para optimizar procesos y generar distintas operaciones the manera paralela, esto es bastante interesante porque de hecho permite hacer llegar a multitud de usuarios un software con funcionalidades super complejas que aproveche de forma eficiente las capacidades de los diversos dispositivos sin importar las diferencias que existan entre sus límites.

Ya entrados en esta parte de la optimización, no solamente entran los procesos en segundo plano sino que también podemos de dicho tener una estructura mucho más modular de los programas que realicemos, a lo que me refiero con esto es que podemos tener un mecanismo sumamente eficiente que ejecute teniendo las de forma bien separadas y en caso de que si algún proceso llegase a fallar nosotros podemos tener un plan de gestión en el cual podamos decidir qué hilo debe morir y que iba debe vivir en caso de que ocurre es un fallo esto es sumamente importante porque puede ahorrarle al usuario y realmente al cliente en muchísimo sufrimiento o si es que algún proceso falla, para que este proceso que ha fallado se mantenga dentro de ese mismo hilo y no genere una reacción en cadena dentro del mismo código y todo se mantenga de forma mucho más modular.

Una vez que ya mencionado esto, creo que queda bastante claro la importancia que tienen los hilos en la industria, no solamente en el sistema operativo también en muchas otras ramas de la industria y de hecho se lo pensamos la optimización del software da la pauta para que muchos usuarios puedan disfrutar de funcionalidades que posiblemente no esta demás alcance por limitaciones técnicas de los dispositivos a los que estos corren o inclusive limitaciones temporales en las que los dispositivos simplemente no están listos para poder ejecutar ciertos programas pero gracias a la optimización del software podemos conseguir eso con el uso de los hilos y desde luego muchos otras técnicas más, pero podemos partir de la base, dos hilos mismos.

Quiero resaltar que uno de los aprendizajes principales que me llevaban esta práctica, es precisamente manejar los hilos en el lenguaje de programación C porque en un principio debo admitir que parecía bastante desafiante la parte el implementación, pero al final resultó ser bastante simple es cierto punto intuitivo porque todo momento el código se trató de mantener siempre de forma sumamente estructurada van a poder mantener siempre el recorrido de los datos dentro de nuestro código en las soluciones a los problemas que implementamos.

Quedó bastante satisfecho con la practica, porque como siempre lo mencionado no solamente la parte teórica ha quedado clara sino que también he logrado expandir un poco más mi visión sobre como y para que podemos utilizar los hilos el industria y creo que sería bastante interesante poder implementaron algún proyecto de escala mucho más grande que pudiera satisfacer necesidades de clientes el mercado.

## **Juárez Espinoza Ulises.**

## **Machorro Vences Ricardo Alberto.**

## **Pastrana Torres Victor Norberto.**

# Bibliografía

|  |  |
| --- | --- |
| [1] | TutorialsPoint, «Tutorialspoint,» 2020. [En línea]. Available: https://www.tutorialspoint.com/operating\_system/os\_multi\_threading.htm. [Último acceso: Octubre 2020]. |
| [2] | CEAC, «CEAC,» CEAC, 01 02 2018. [En línea]. Available: https://www.ceac.es/blog/que-es-la-programacion-multihilo-y-que-ventajas-tiene. [Último acceso: 08 2020]. |
| [3] | fingEDU, fingEDU, 2020. [En línea]. Available: https://www.fing.edu.uy/tecnoinf/mvd/cursos/so/material/teo/so05-hilos.pdf. [Último acceso: 2020]. |
| [4] | GURU99, «GURU99,» 2020. [En línea]. Available: https://www.guru99.com/difference-between-process-and-thread.html#:~:text=Process%20means%20a%20program%20is,Lightweight%2C%20whereas%20Threads%20are%20Lightweight.&text=A%20Process%20is%20mostly%20isolated,share%20data%20with%20each%20other.. [Último acceso: Octubre 2020]. |
| [5] | Universidad Carlos III de Madrid, «UC3M,» 2020. [En línea]. Available: http://ocw.uc3m.es/ingenieria-informatica/sistemas-operativos/material-de-clase-1/mt\_t2\_l5.pdf. [Último acceso: Octubre 2020]. |
| [6] | sistemasoper2, «sistemasoper2,» sistemasoper2, 2020. [En línea]. Available: https://sistemasoper2.wordpress.com/2014/10/21/usos-mas-comunes-de-los-hilos/. [Último acceso: Octubre 2020]. |
| [7] | jovenclub, «jovenclub,» jovenclub, 2020. [En línea]. Available: https://gutl.jovenclub.cu/como-crear-hilos-en-c-de-forma-facil/. [Último acceso: Octubre 2020]. |
| [8] | slideshare, «slideshare,» slideshare, 8 Enero 2019. [En línea]. Available: https://es.slideshare.net/EmmanuelGarcaFortuna/procesos-e-hilos-en-los-sistemas-operativos. [Último acceso: 2020]. |