

**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](#_Toc55904861)

[**Hilos.** 1](#_Toc55904862)

[**Contenido (Investigación)** 2](#_Toc55904863)

[**Consideraciones.** 2](#_Toc55904864)

[**Importancia de los hilos.** 2](#_Toc55904865)

[**Tipos de hilos.** 2](#_Toc55904866)

[Hilos del usuario. 2](#_Toc55904867)

[Hilos del kernel. 3](#_Toc55904868)

[**Diferencia entre los hilos y los procesos** 3](#_Toc55904869)

[**Códigos y ventanas de ejecución** 4](#_Toc55904870)

[**Programa51.c** 4](#_Toc55904871)

[**Código explicado por partes.** 4](#_Toc55904872)

[**Código completo.** 6](#_Toc55904873)

[**Explicación de manera global del código.** 8](#_Toc55904874)

[**Ejecución:** 8](#_Toc55904875)

[**Programa52.c** 9](#_Toc55904876)

[**Código explicado por partes.** 9](#_Toc55904877)

[**Código completo.** 9](#_Toc55904878)

[**Explicación de manera global del código.** 9](#_Toc55904879)

[**Ejecución:** 9](#_Toc55904880)

[**Programa53.c** 9](#_Toc55904881)

[**Código explicado por partes.** 9](#_Toc55904882)

[**Código completo.** 9](#_Toc55904883)

[**Explicación de manera global del código.** 9](#_Toc55904884)

[**Ejecución:** 9](#_Toc55904885)

[**Programa54.c** 10](#_Toc55904886)

[**Código explicado por partes.** 10](#_Toc55904887)

[**Código completo.** 10](#_Toc55904888)

[**Explicación de manera global del código.** 10](#_Toc55904889)

[**Ejecución:** 10](#_Toc55904890)

[**Programa55.c** 10](#_Toc55904891)

[**Código explicado por partes.** 10](#_Toc55904892)

[**Código completo.** 10](#_Toc55904893)

[**Explicación de manera global del código.** 10](#_Toc55904894)

[**Ejecución:** 10](#_Toc55904895)

[**Conclusiones.** 11](#_Toc55904896)

[**Chavarría Vázquez Luis Enrique.** 11](#_Toc55904897)

[**Juárez Espinoza Ulises.** 11](#_Toc55904898)

[**Machorro Vences Ricardo Alberto.** 12](#_Toc55904899)

[**Pastrana Torres Victor Norberto.** 12](#_Toc55904900)

[Bibliografía 12](#_Toc55904901)

Índice de figuras

**No se encuentran elementos de tabla de ilustraciones.**

Índice de tablas

**No se encuentran elementos de tabla de ilustraciones.**

# **Glosario de términos.**

## **Hilos.**

# **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.

## **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.

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.

### 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.

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

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

* 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.

## **Usos comunes de 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.

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.

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:



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.



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.

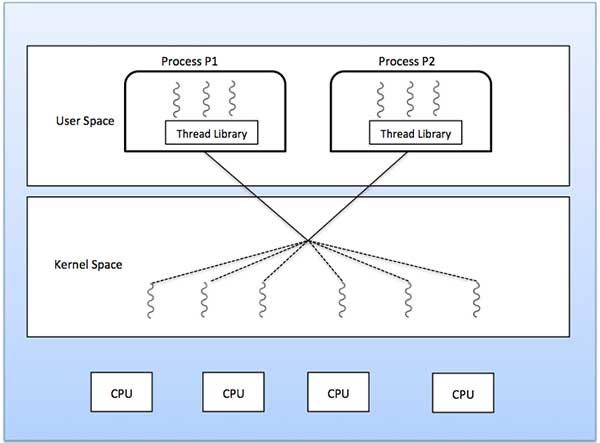
Algunos sistemas operativos proporcionan un subproceso de nivel de usuario combinado y un servicio de subproceso de nivel de kernel. Solaris es un buen ejemplo de este enfoque combinado. En un sistema combinado, varios subprocesos dentro de la misma aplicación pueden ejecutarse en paralelo en varios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

* 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.

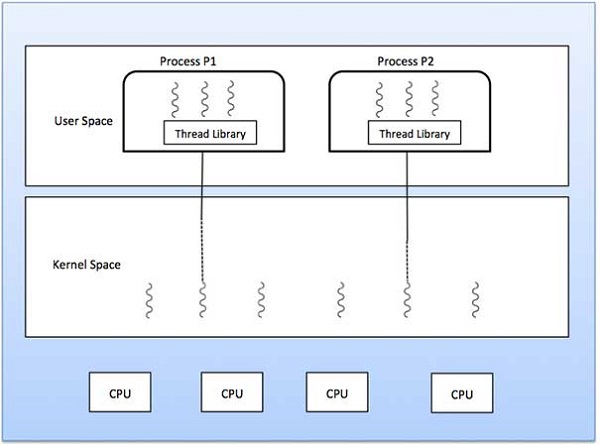
El siguiente diagrama muestra el modelo de subprocesos de muchos a muchos donde 6 subprocesos a nivel de usuario se multiplexan con 6 subprocesos a nivel de kernel. En este modelo, los desarrolladores pueden crear tantos subprocesos de usuario como sea necesario y los subprocesos de Kernel correspondientes 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.



### 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.

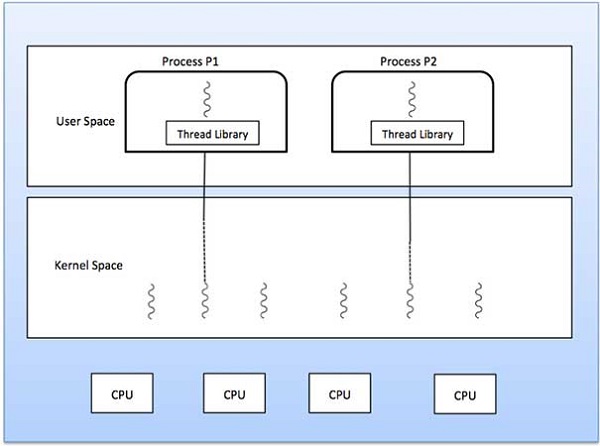
Si las bibliotecas de subprocesos a nivel de usuario se implementan en el sistema operativo de tal manera que el sistema no las admite, entonces los subprocesos del Kernel utilizan los modos de relación de muchos a uno.



### 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.

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.

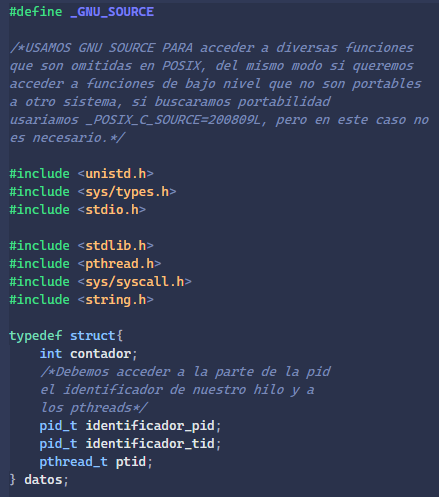


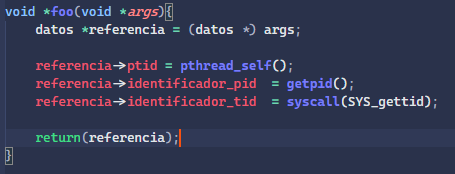
# **Códigos y ventanas de ejecución**

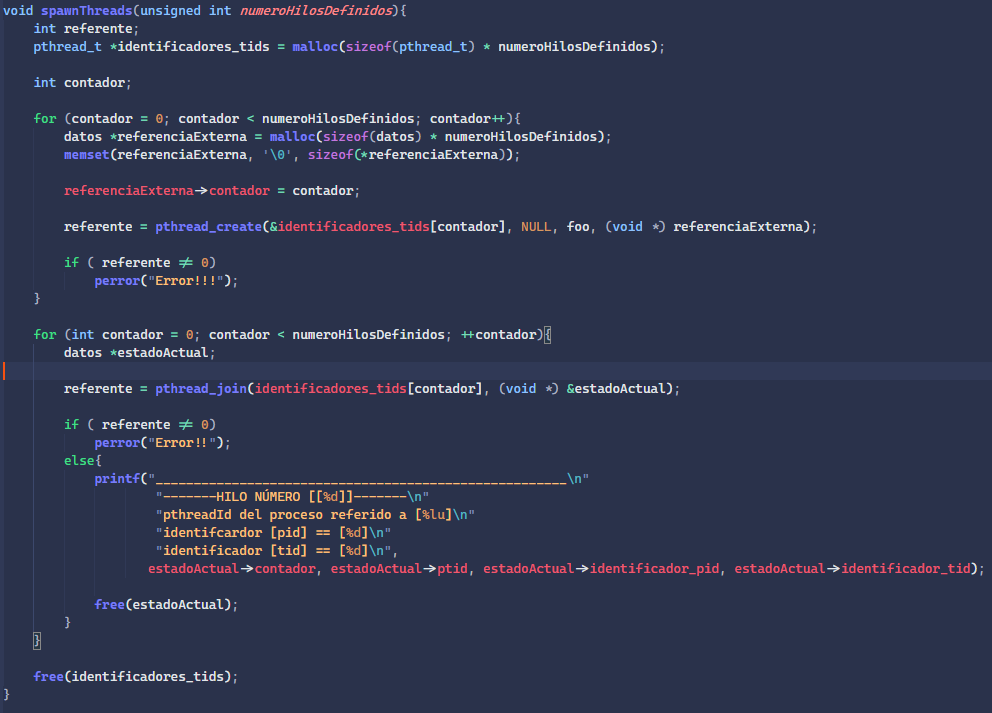
## **Programa51.c**

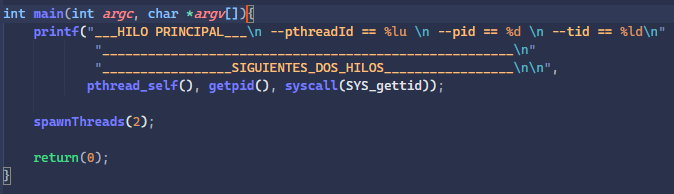
Realizar un programa que inicie un hilo principal que a su vez crea dos hilos. El hilo principal espera hasta que ambos hilos terminen y después finaliza. Los hilos sólo deben de mostrar algún mensaje en pantalla y terminar.

### **Código explicado por partes.**









### **Código completo.**

#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));

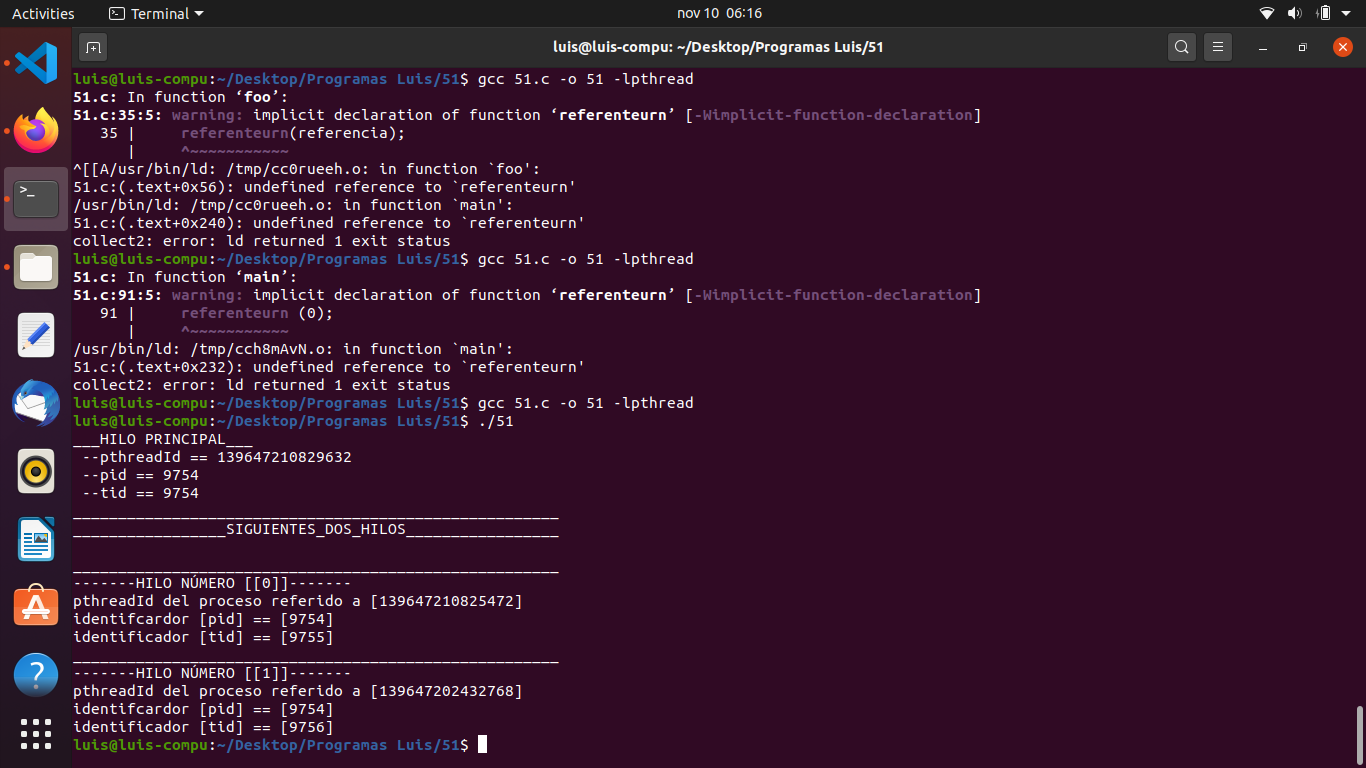
**spawnThreads**(2);

    return(0);

}

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

### **Ejecución:**



## **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.**

## **Juárez Espinoza Ulises.**

## **Machorro Vences Ricardo Alberto.**

## **Pastrana Torres Victor Norberto.**

# Bibliografía

**No hay ninguna fuente en el documento actual.**