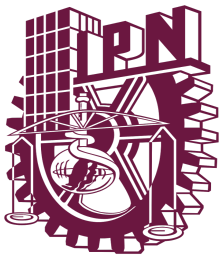


INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

U.A.: APLICACIONES PARA COMUNICACIONES DE RED

PROFESOR: RANGEL GONZALEZ JOSUE

GRUPO: 3CV7

PRACTICA 1

ALUMNOS:

ALDO SUAREZ CRUZ

JOSUÉ ARTURO ALFARO BRACHO

VLADIMIR AZPEITIA HERNÁNDEZ

Introducción

El presente trabajo sobre threads forma parte de una de las bases para poder entender el funcionamiento de temas más complejos que iremos abordando durante este curso. Lo primero que hay que entender antes de abordar el concepto de thread es donde se almacenan y donde se ejecutan dentro de una computadora. Sabemos que una computadora tiene un procesador que es quien ejecuta las instrucciones de los programas que están cargados en la memoria RAM, por el procesador pasan todas las instrucciones que hacen funcionar una computadora. Dentro de un procesador existen módulos llamados núcleos, cada núcleo ejecutará miles de instrucciones, estas instrucciones son ejecutadas en cada ciclo de reloj. La finalidad de tener varios núcleos tiene que ver con mejorar el rendimiento de la computadora.

Un thread no forma parte físicamente de un procesador o núcleo, se encuentra almacenado dentro de núcleos. Un thread o hilo se define como un medio que permite administrar las tareas que debe realizar un procesador de manera que las instrucciones se ejecuten de manera eficiente. De esta manera es posible procesar varias tareas al mismo tiempo. Para poder cargar un programa este debe cargarse en la memoria RAM mediante procesos, este concepto comúnmente es confundido con thread, por lo que más adelante se muestran diferencias entre estos.

Los procesos necesitan de ciertos recursos para poder ejecutarse, estos son contador de programa y una pila de registros. Entonces cada programa se divide en procesos, y cada proceso se ejecuta de forma independiente dentro de la memoria

Diferencias entre procesos y threads

La relación que existe entre thread y proceso es que un thread es por así decirlo una unidad de ejecución de un proceso, esto se refiere a que el proceso puede dividirse en subprocesos, y cada uno de estos subprocesos es conocido como un thread. Se pensaría que un proceso y un thread son casi iguales ya que los dos se ejecutan de manera similar y llevan casi las mismas tareas, pero tiene varias diferencias:

- El proceso es un programa en ejecución, mientras que el hilo es un proceso ligero o parte de un proceso.
- Un proceso se encuentra completamente aislado y no comparte memoria, el thread por su parte puede compartir memoria con los demás threads.
- Un proceso consume más memoria que un thread.
- Un proceso requiere más tiempo de creación que un thread, de la misma forma para finalizar un proceso requiere más tiempo que un thread.

- Un thread no puede existir de manera individual, ya que es parte de un proceso, siempre es adjuntado a un proceso, pero un proceso si puede existir de manera individual.
- Cuando un thread termina, su pila puede ser recuperada ya que cada uno puede tener su propia pila, pero en el caso del proceso si termina, todos los threads terminan incluido el proceso.

Estados de un thread

Un thread pasa por cuatro estados:

- New o creación: todos los threads se encuentran en este estado la primera vez que son creados hasta que son llamados. En este punto ya se han inicializado, pero aun no has sido notificados para iniciar una tarea.
- Running: en este punto el thread ya se encuentra ejecutándose.
- Not Running: se aplica a todos los threads que se encuentran detenidos por alguna razón, los hilos que se encuentran este estado pueden cambiar a estado running en cualquier momento.
- Dead: Los threads entran este estado cuando ya no son necesarios, se puede llegar a este estado cuando se han terminado de ejecutar o son detenidos.

Información de hilos

Un hilo contiene la información necesaria para representar un contexto de aplicación dentro de un proceso. Ésta es:

- ID del hilo. No son únicos dentro del sistema, sólo tienen sentido en el contexto de cada proceso.
- Stack pointer
- Un conjunto de registros
- Propiedades de itineración (como política y prioridad)
- Conjunto de señales pendientes y bloqueadas.
- Datos específicos del hilo.

Manejo de Threads en UNIX

POSIX significa **P**ortable **O**perating **S**ystem **I**nterface (for **U**nix). Es un estándar orientado a facilitar la creación de aplicaciones confiables y portables. La mayoría de las versiones populares de UNIX (Linux, Mac OS X) están cumpliendo este estándar en gran medida. La biblioteca para el manejo de hilos en POSIX es pthread, este estándar define entre otras funciones para el manejo de threads creación, destrucción de atributos de creación de threads, terminación y espera a la terminación de un thread e Identificación de threads. Durante el desarrollo de este trabajo se explicarán las sintaxis y propiedades de las instrucciones para el manejo de threads.

Desarrollo

Esta práctica abordará el tema de threads en dos ejercicios diferentes, uno desarrollado en lenguaje c, y otro desarrollado con lenguaje java; aunque en el lenguaje java el manejo de hilos tiene una sintaxis diferente, el funcionamiento y los conceptos son muy similares.

Ejercicio 1

En un directorio copiar 10 imágenes, cada una de estas imágenes serán asignadas a un hilo. El hilo abrirá la imagen como una matriz RGB y generará 3 copias de dicha imagen, la primera copia asignará el valor de 255 a R, la segunda copia asignará el valor de 255 a G y la tercera copia asignará el valor de 255 a B. Cada una de las copias modificadas será guardada nuevamente como una imagen, lo cual provocará que al final se tengan por cada imagen original 3 copias saturadas en colores diferentes, Rojo, Verde y Azul


Inicio
+ Número de Hilos
+ Directorio de imagenes
+ Tipo de saturación
Se lee el numero de imagenes en la carpeta
Se crea el número de hilos a partir de las imagenes
Se satura la imagen de cada uno de los colores
Se guarda la nueva imágen
Se espera a que temrinen los hilos
Terminador

- Al inicio del programa se define la carpeta que se va a utilizar para traer las imágenes y por cada elemento de este se inicia un hilo los parámetros de este hilo son la imagen que va a trabajar y con este parámetro se inicia el hilo

```
public static void main(String[] args) {
    //CARPETA DONDE ESTAN GUARDADAS LAS IMAGENES
    String path = "/home/xubuntu/Desktop/redes2/practica01/imagenesRGB/img/";
    File directoryPath = new File(path); //ABRIMOS LA CARPETA
    String contents[] = directoryPath.list(); //LISTAMOS EL CONTENIDO

    for (String imageName : contents) { //OBTENEMOS LOS NOMBRES DE LAS IMAGENES
        ImageRGB thread = new ImageRGB(directoryPath.getAbsolutePath()+ "/" +imageName);
        thread.start();
    }
}
```

- Cada hilo realiza la saturación de la imagen que le tocó en los 3 colores, esto se hace llamando a una función para hacer la saturación correspondiente de cada uno

```
public void run() {
    System.out.println("Corriendo hilo para saturar la imagen " + directoryImage);
    try {
        File file = new File(directoryImage); //ABRIMOS EL ARCHIVO
        try{
            saturateImage(file, ImageRGB.RED_COLOR); //SATURAMOS LA IMAGEN EN ROJO
            saturateImage(file, ImageRGB.GREEN_COLOR); //SATURAMOS LA IMAGEN EN VERDE
            saturateImage(file, ImageRGB.BLUE_COLOR); //SATURAMOS LA IMAGEN EN AZUL
        }catch(IOException e){
            e.printStackTrace();
        }
        Thread.sleep(50); //ESPERAMOS A QUE EL HILO FINALIZE
    } catch (InterruptedException e) {
        System.out.println("El hilo con la imagen " + directoryImage + " se ha interrumpido.");
    }
    System.out.println("El hilo con la imagen " + directoryImage + " finalizó.");
}
```

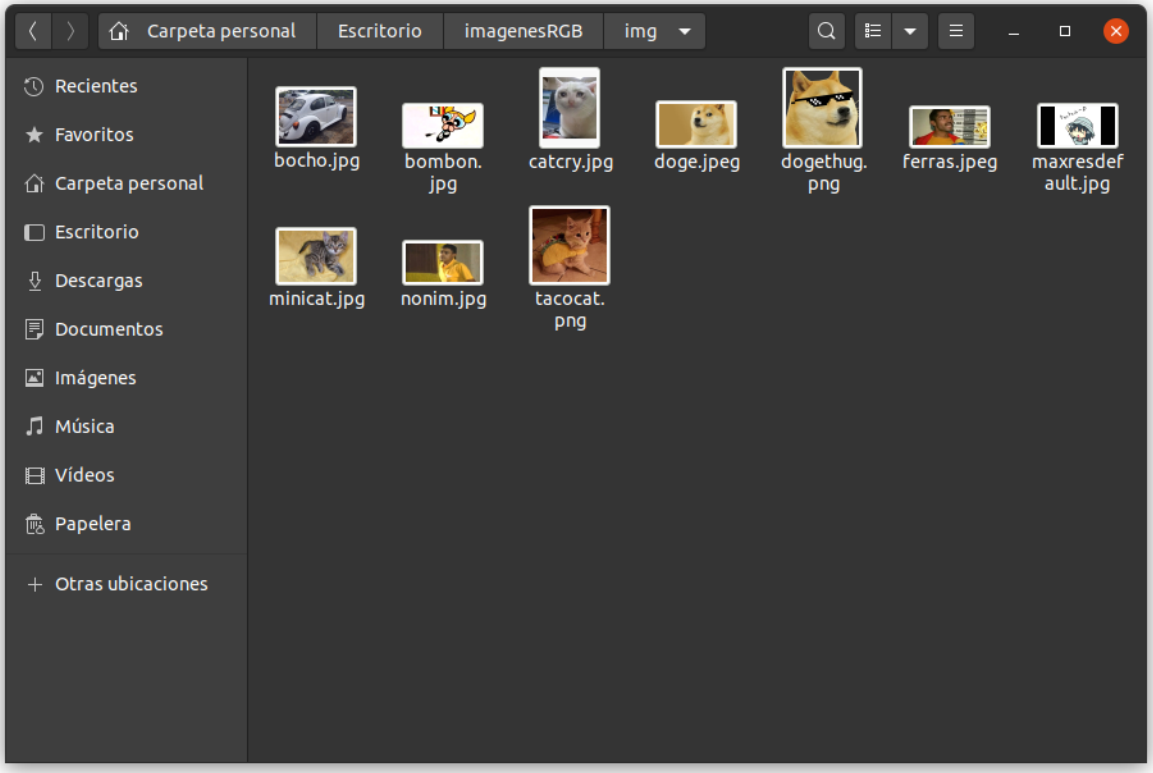
- En esta parte se toma la imagen para hacer el cambio se toma la imagen y se transforma en una matriz rgb y dependiendo de los parámetros se realiza la saturación a rojo, verde o azul, una vez hecha la saturación se guarda la imagen con el mismo nombre agregando el color correspondiente y como vemos en la imagen anterior se pone el thread.sleep para que terminen los hilos

```java
private void saturateImage(File file, String optionColor) throws IOException{
    BufferedImage img = ImageIO.read(file); //Abrimos la imagen en un buffer
    for (int y = 0; y < img.getHeight(); y++) {
        for (int x = 0; x < img.getWidth(); x++) {
            // Obtener color del pixel en la posicion xy
            int pixel = img.getRGB(x, y);
            // Lo pasamos a la clase Color
            Color color = new Color(pixel, true);
            // Separamos RGB con Alpha
            int red = color.getRed();
            int green = color.getGreen();
            int blue = color.getBlue();
            int alpha = color.getAlpha();

            //SE SATURA DEPENDIENDO DEL COLOR QUE SE ELIGIO DESDE LOS PARAMETROS
            if (optionColor == RED_COLOR) {
                color = new Color(255, green, blue, alpha); //SATURAR EN ROJO
                img.setRGB(x, y, color.getRGB());
            } else if (optionColor == GREEN_COLOR) {
                color = new Color(red, 255, blue, alpha); //SATURAR EN VERDE
                img.setRGB(x, y, color.getRGB());
            } else if (optionColor == BLUE_COLOR) {
                color = new Color(red, green, 255, alpha); //SATURAR EN AZUL
                img.setRGB(x, y, color.getRGB());
            }

        }
    }
    //SEPARAMOS LA RUTA DEL NOMBRE DEL ARCHIVO
    String path = file.getAbsolutePath().replaceFirst("[.][^.]+$", "");
    //SEPARAMOS LA EXTENCION DEL ARCHIVO DEL NOMBRE
    String fileExt = file.getName().replaceAll("^.*\\.(.*)$", "$1");

    //CREAMOS UNA COPIA DEL ARCHIVO AGREGANDO EL COLOR QUE SE SATURO AL FINAL DEL NOMBRE ORIGINAL
    // IMAGE.JPG ---> IMAGENBLUE.JPG
    File fileCopy = new File(path + optionColor+ "." + fileExt);
    //GUARDAMOS LA IMAGEN DEL BUFFER EN LA COPIA DEL ARCHIVO
    ImageIO.write(img, fileExt, fileCopy);
}
```

Una vez hecho esto primero nos aseguramos que en el main del código este la carpeta donde estan las imagenes con la ruta correcta

```java
public static void main(String[] args) {
    //CARPETA DONDE ESTAN GUARDADAS LAS IMAGENES
    String path = "/home/parzival/Escritorio/imagenesRGB/img/";
    File directoryPath = new File(path); //ABRIMOS LA CARPETA
    String contents[] = directoryPath.list(); //LISTAMOS EL CONTENIDO
```

En cuanto sepamos eso pasaremos a ver la carpeta como estan las imagenes



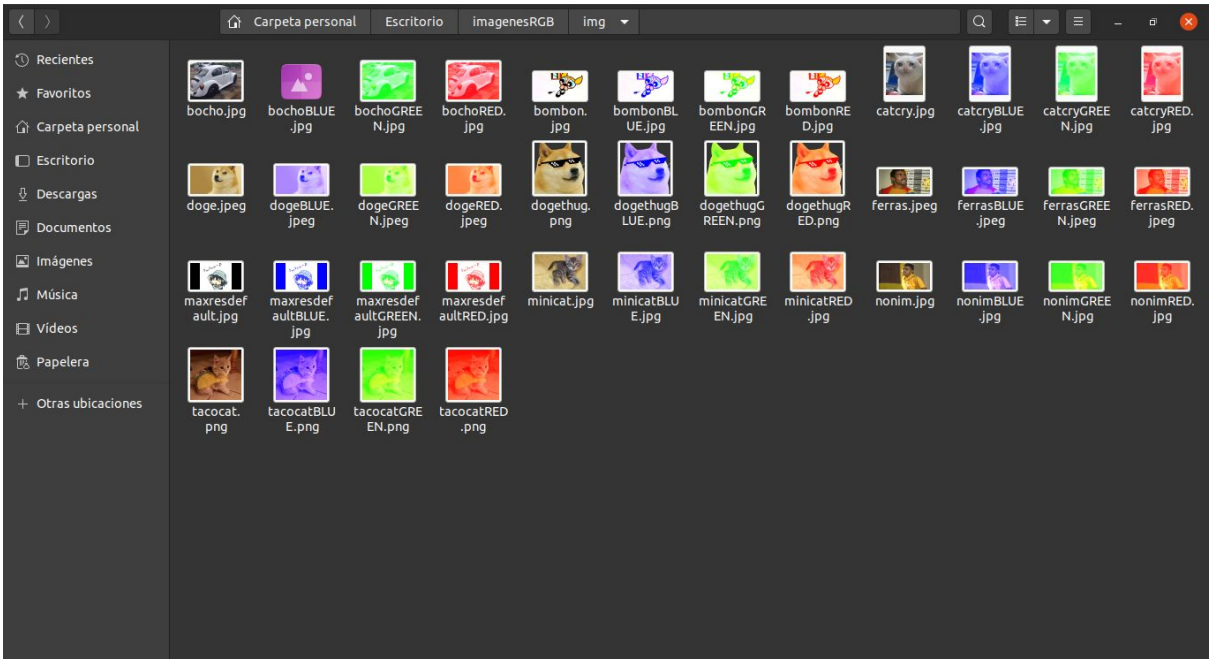
Pasamos a compilar y ejecutar el código en ubuntu lo hacemos de la siguiente manera ya estando en la donde se encuentra el archivo

```
parzival@parzival-Lenovo-YOGA-510-14ISK:~/Escritorio/imagenesRGB$ java ImageRGB.
java
```

una vez hecho esto comenzará la ejecución

```
parzival@parzival-Lenovo-YOGA-510-14ISK: ~/Escritorio/imagenesRGB
parzival@parzival-Lenovo-YOGA-510-14ISK:~/Escritorio/imagenesRGB$ java ImageRGB.java
Creando hilo para la imagen /home/parzival/Escritorio/imagenesRGB/img/catcry.jpg
Empezando hilo de /home/parzival/Escritorio/imagenesRGB/img/catcry.jpg
Creando hilo para la imagen /home/parzival/Escritorio/imagenesRGB/img/doge.jpeg
Empezando hilo de /home/parzival/Escritorio/imagenesRGB/img/doge.jpeg
Creando hilo para la imagen /home/parzival/Escritorio/imagenesRGB/img/dogethug.png
Empezando hilo de /home/parzival/Escritorio/imagenesRGB/img/dogethug.png
Corriendo hilo para saturar la imagen /home/parzival/Escritorio/imagenesRGB/img/catcry.jpg
Corriendo hilo para saturar la imagen /home/parzival/Escritorio/imagenesRGB/img/doge.jpeg
Creando hilo para la imagen /home/parzival/Escritorio/imagenesRGB/img/bombon.jpg
Empezando hilo de /home/parzival/Escritorio/imagenesRGB/img/bombon.jpg
Corriendo hilo para saturar la imagen /home/parzival/Escritorio/imagenesRGB/img/dogethug.png
Creando hilo para la imagen /home/parzival/Escritorio/imagenesRGB/img/nonim.jpg
Empezando hilo de /home/parzival/Escritorio/imagenesRGB/img/nonim.jpg
Corriendo hilo para saturar la imagen /home/parzival/Escritorio/imagenesRGB/img/bombon.jpg
Creando hilo para la imagen /home/parzival/Escritorio/imagenesRGB/img/bocho.jpg
Empezando hilo de /home/parzival/Escritorio/imagenesRGB/img/bocho.jpg
Corriendo hilo para saturar la imagen /home/parzival/Escritorio/imagenesRGB/img/nonim.jpg
Creando hilo para la imagen /home/parzival/Escritorio/imagenesRGB/img/maxresdefault.jpg
Empezando hilo de /home/parzival/Escritorio/imagenesRGB/img/maxresdefault.jpg
Corriendo hilo para saturar la imagen /home/parzival/Escritorio/imagenesRGB/img/bocho.jpg
Creando hilo para la imagen /home/parzival/Escritorio/imagenesRGB/img/tacocat.png
Corriendo hilo para saturar la imagen /home/parzival/Escritorio/imagenesRGB/img/maxresdefault.jpg
Empezando hilo de /home/parzival/Escritorio/imagenesRGB/img/tacocat.png
Creando hilo para la imagen /home/parzival/Escritorio/imagenesRGB/img/ferras.jpeg
Empezando hilo de /home/parzival/Escritorio/imagenesRGB/img/ferras.jpeg
Corriendo hilo para saturar la imagen /home/parzival/Escritorio/imagenesRGB/img/tacocat.png
Creando hilo para la imagen /home/parzival/Escritorio/imagenesRGB/img/minicat.jpg
Empezando hilo de /home/parzival/Escritorio/imagenesRGB/img/minicat.jpg
Corriendo hilo para saturar la imagen /home/parzival/Escritorio/imagenesRGB/img/ferras.jpeg
Corriendo hilo para saturar la imagen /home/parzival/Escritorio/imagenesRGB/img/minicat.jpg
El hilo con la imagen /home/parzival/Escritorio/imagenesRGB/img/ferras.jpeg finalizó.
El hilo con la imagen /home/parzival/Escritorio/imagenesRGB/img/minicat.jpg finalizó.
El hilo con la imagen /home/parzival/Escritorio/imagenesRGB/img/catcry.jpg finalizó.
El hilo con la imagen /home/parzival/Escritorio/imagenesRGB/img/dogethug.png finalizó.
El hilo con la imagen /home/parzival/Escritorio/imagenesRGB/img/bombon.jpg finalizó.
El hilo con la imagen /home/parzival/Escritorio/imagenesRGB/img/maxresdefault.jpg finalizó.
El hilo con la imagen /home/parzival/Escritorio/imagenesRGB/img/tacocat.png finalizó.
El hilo con la imagen /home/parzival/Escritorio/imagenesRGB/img/nonim.jpg finalizó.
```

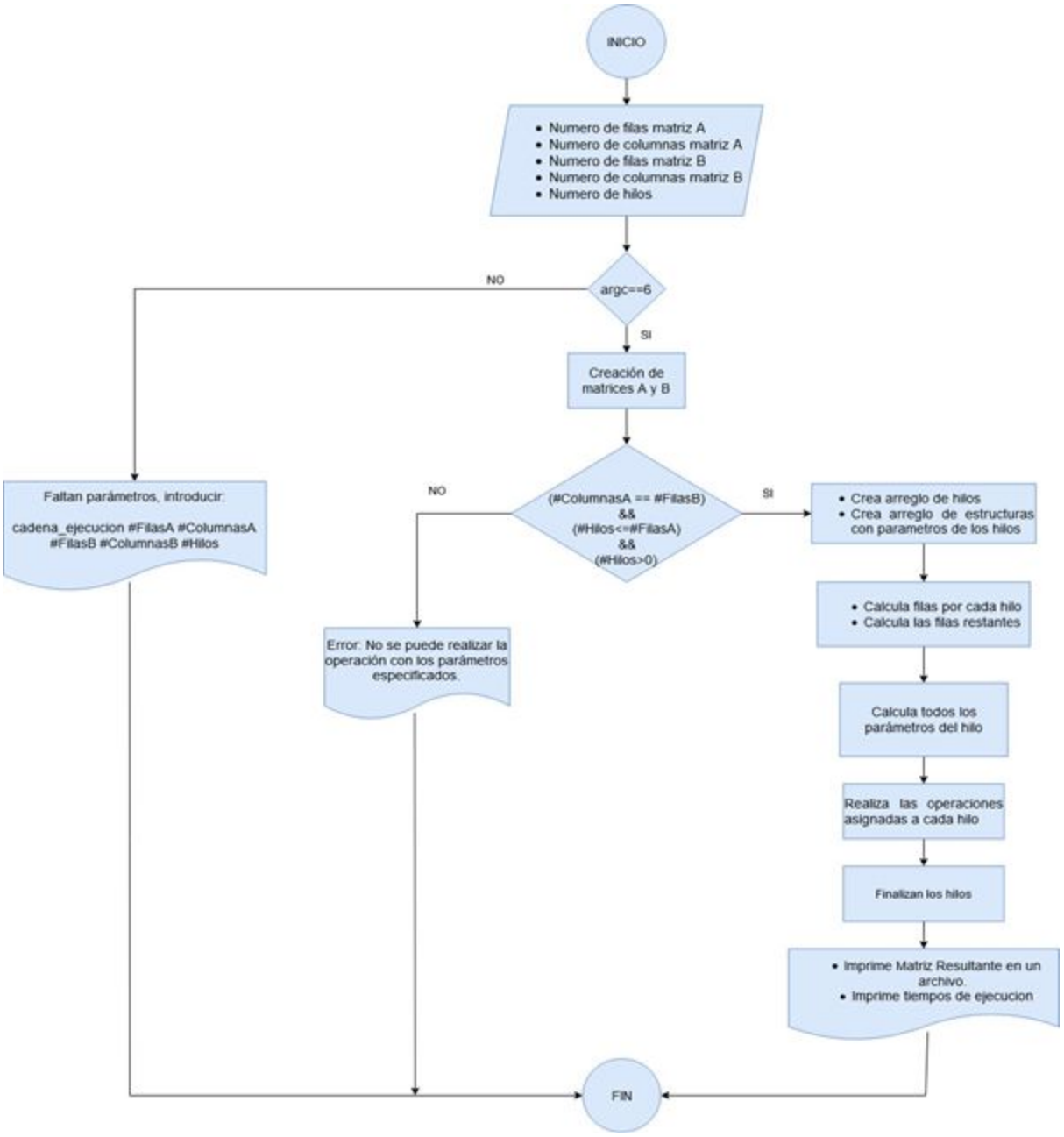
Para comprobar pasamos a ver la carpeta la cual se veran todos los archivos originales con todas sus formas saturadas



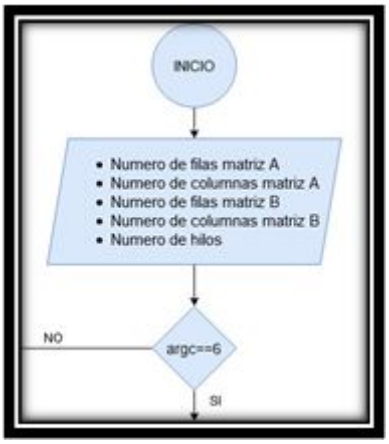
Ejercicio 2

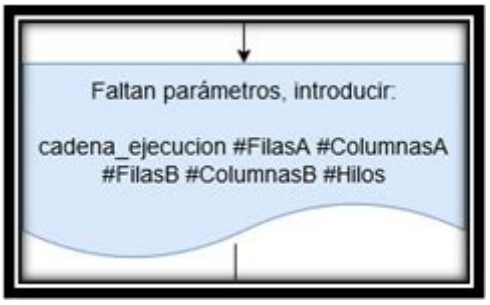
Generar la multiplicación de matrices donde A será una matriz de tamaño 2000 x 1500 y B será de tamaño 1500 x 2000. Se ejecutarán diferentes cantidades de hilos que resuelvan la matriz, la cantidad de hilos serán 2, 4, 8, 16 y 32. En cada ejecución se deben medir los tiempos y al final se debe generar una conclusión, ¿Qué sucede cuando mas hilos ejecutan la matriz?, ¿Vale la pena crear grandes cantidades de hilos para resolver un problema?

La solución al problema planteado en el enunciado anterior se modela por medio de un diagrama de flujo, en este diagrama se modelan algunos procesos de manera general y después del diagrama se van explicando detalladamente cada uno.



- El inicio del diagrama no tiene muchas complicaciones se introducen los parámetros para la ejecución del programa, se introduce el número de filas y columnas para las matrices A y B, además se introduce el número de hilos que ejecutan las operaciones. Si la cantidad de argumentos que se especifican en la línea de ejecución es diferente de 6 indica que faltó algún parámetro y se indica que parámetros se deben introducir, de lo contrario sigue la ejecución del programa.





- Una vez que han sido verificadas la cantidad de argumentos se crean las matrices con los parámetros introducidos. El primer paso es reservar memoria para la cantidad de datos que llevará cada matriz. Luego de reservar la memoria comienza a llenarse las dos matrices con valores aleatorios dentro de un rango de 0 a 6.

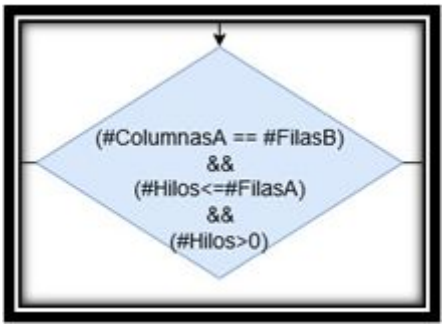
- El siguiente paso es validar los parámetros que se introdujeron, esto se logra solo si las tres condiciones se cumplen:

    1. El número de columnas de A es igual al número de filas de B, esta condición está definida por la multiplicación de matrices.

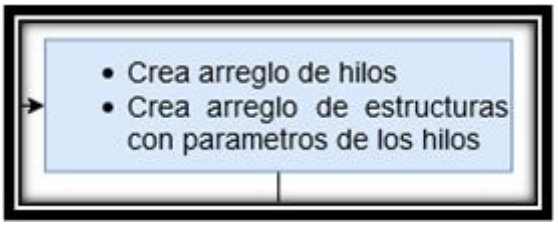
2. El número de hilos es menor o igual al número de filas de A, con esto aseguramos que no haya hilos que queden sin realizar operaciones.

3. El número de hilos sea mayor a cero, con esto aseguramos que por lo menos un hilo ejecute las operaciones.

En caso de que alguna de las condiciones anteriores no se cumplan el programa indicará que alguno de los parámetros es incorrecto y terminará la ejecución.

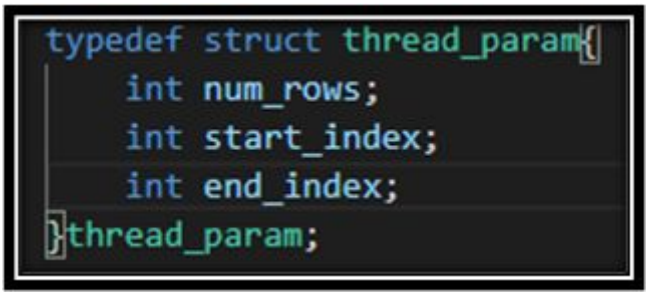


- Una vez que se hayan validado los parámetros introducidos continúa el proceso, primero se crea un arreglo de hilos con el número de elementos indicados por el usuario. En seguida se crea un arreglo de estructuras el cual contiene variables que almacenarán los parámetros que ocupará el hilo para realizar las operaciones.

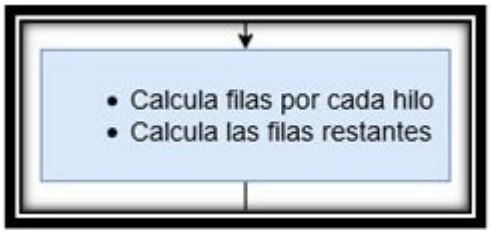



```
pthread_t threads[num_threads];
thread_param params[num_threads];
```

La estructura en la que se almacenan los parámetros que ocupará el hilo se muestra en la imagen. El nombre de la estructura es **thread_param,** contiene tres variables de tipo entero, la variable **num_rows** almacena el número de filas que le corresponde a cada hilo, **start_index** almacena el índice de la primera fila que le corresponde al hilo y la tercera variable **end_index** almacenará el índice de la última fila que le corresponde al hilo.



- El siguiente paso es calcular la cantidad de filas que serán asignadas a cada hilo, se divide primero la cantidad de filas de la matriz A entre la cantidad de hilos y se almacenan en la variable **div.** En caso de que la división no sea exacta se realiza la operación módulo entre la cantidad de filas de la matriz A y el número de hilos.

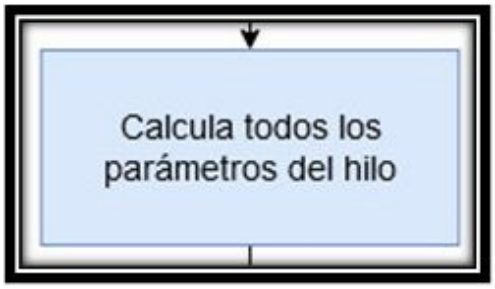



```
int div = A_rows / num_threads;
int res = A_rows % num_threads;
```

- A continuación, se calculan los parámetros para cada hilo, el ciclo for recorre el arreglo de estructuras que se creó para los parámetros de cada hilo. La condición if se ejecuta solo una vez, en esta ejecución se asigna las filas sobrantes siempre al primer hilo. En los ciclos restantes solo se ejecutará el código dentro de else, en este se asigna el número de hilos con el que trabajara, este número se obtuvo de la división de filas entre hilos.

  Después del código de else, se indica a la estructura el índice de la primera fila con la que va a trabajar, este valor se almacena en la variable **position** que en la

primera asignación tiene el valor de 0. A continuación la variable **position** se incrementa en el número de filas que le corresponde, esto para obtener el índice de la última fila que le corresponde. En la ultima línea se asigna el valor de la variable position que indica la ultima fila que le corresponde al hilo. Una vez que se ha recorrido todo el arreglo de estructuras se han llenado los parámetros para todos los hilos.

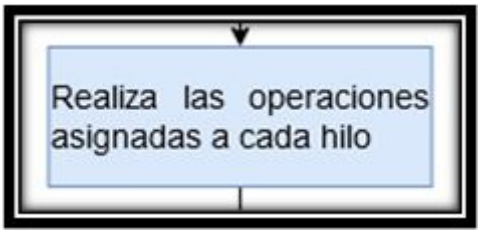



```
for (int i = 0; i < num_threads; i++){
    if(i == 0){
        params[i].num_rows = div + res;
    }else{
        params[i].num_rows = div;
    }

    params[i].start_index = position;
    position += params[i].num_rows;
    params[i].end_index = position;
}
```

- En este proceso se encuentran incluidos varios procesos, la creación de los hilos, finalización de hilos y las instrucciones que llevará a cabo cada uno. Dentro un ciclo for se recorre el arreglo de hilos, se va creando cada hilo del arreglo, este direcciona a la función **threadf,** en la imagen de abajo se muestra el código de la función. Se utiliza otro ciclo for para iniciar el recorrido de las filas de la matriz para realizar las operaciones. El ciclo inicia en el índice almacenado por la estructura de parámetros de cada hilo y va recorriendo hasta el índice del final de los parámetros. A continuación utiliza otro ciclo con el que recorrerá las columnas de la matriz B. En el código se puede observar que las matrices A y B no se encuentran declaradas dentro de la función, esto debido a que están declaradas como variables globales, si recordamos una de las propiedades de los hilos es que varios hilos pueden compartir recursos. En el for más anidado se van realizando las operaciones necesarias de filas y columnas de las matrices A y B, una vez terminado el ciclo el resultado va a colocarse en la matriz C que será la matriz de resultados.

  Así seguirá realizando todas las operaciones para cada fila asignada a ese hilo, una vez que haya terminado ese hilo de realizar sus operaciones termina el hilo pero sin terminar el proceso.

```c
void *threadf(void *arg) {
    thread_param *param = (thread_param *) arg;
    for (int i = param->start_index; i < param->end_index; i++){
        for (int j = 0; j < B_cols; j++){
            int sum = 0;
            for (int k = 0; k < A_cols; k++){
                sum += matrixA[i][k] * matrixB[k][j];
            }
            matrixC[i][j] = sum;
        }
    }
    sleep(2);
    pthread_exit(NULL);
}
```



- Una vez que terminó las operaciones de cada hilo se recorre el arreglo de hilos con un ciclo for, con este van finalizando los hilos con la función pthread_join.

- Por último se imprimen los tiempos de ejecución que fueron tomados, estos se envían a un archivo y la matriz resultante se imprime en un archivo de texto llamado result.txt.

En la siguiente tabla se muestra los tiempos que se obtuvieron al realizar la multiplicación de las matrices A y B, así como el número de hilos con los que se ejecutan las operaciones. El tamaño de las matrices que se utiliza para este ejercicio es A=2000x1500 B=1500x2000

Número de hilos	Tiempo Total
2	30.9204150000 s
4	30.0372100000 s
8	48.5874450000 s
16	50.3583180000 s
32	53.0682480000 s

La compilacion y ejecucion se realiza mediante un archivo .sh, en este archivo se indican las instrucciones:

```
#!/bin/bash

gcc -pthread -o practica1 mult_matrices.c tiempo.c
./practica1 2000 1500 1500 2000 32 >> nuevo.txt
```

La primera línea realiza la compilación del programa con las librerías necesarias para manejar threads y la segunda es una librería creada para mostrar los tiempo de ejecución, en la segunda línea se ejecuta el programa se indican los argumentos del programa, después por medio de los signos “>>” se indica al programa que envie las impresiones normales (tiempos) a un archivo en lugar de imprimirlos en consola. En consola solo se ejecuta el archivo .sh.

```
~/Escritorio/practica1_redes2$ ./e.sh
~/Escritorio/practica1_redes2$ ./e.sh
~/Escritorio/practica1_redes2$ ./e.sh
```

El formato de impresión de tiempos se muestra en un archivo de la siguiente forma:

```
80 real (Tiempo total)  1604205329.4152779579 s
81 user (Tiempo de procesamiento en CPU) 53.0682480000 s
82 sys (Tiempo en acciónes de E/S)  0.0906070000 s
83 CPU/Wall   0.0000033137 %
84
85
86 real (Tiempo total)  1.6042053294e+09 s
87 user (Tiempo de procesamiento en CPU) 5.3068248000e+01 s
88 sys (Tiempo en acciónes de E/S)  9.0607000000e-02 s
89 CPU/Wall   0.0000033137 %
90
91 Se guardo el resultado en result.txt
```

Conclusiones

ALDO SUAREZ CRUZ: Una de las principales funciones de una computadora es realizar varias tareas al mismo tiempo. Los hilos forman parte de las herramientas con las cuales las computadoras pueden realizar estas tareas, pero al mismo tiempo una cantidad grande de hilos no asegura que estas tareas se realicen más rápido. En el segundo ejercicio se pudo observar el comportamiento mencionado, ya que al incrementar la cantidad de hilos no se realiza más rápido el procesamiento de datos. Las matrices se llevan más tiempo en realizar todas operaciones, con esto podemos concluir que más hilos no significa más velocidad de procesamiento.

JOSUÉ ARTURO ALFARO BRACHO: Durante esta parte del curso y al realizar la práctica las principales dificultades con las que me presente fue al inicio en como pasar los parámetros entre los hilos, en el caso de la matriz de números la parte que se me complico era la parte de guardar los resultados en la matriz resultante ya que me mandaba error de segmento ya que no se creaba bien de inicio ni vacío, con respecto al primer programa fue principalmente entender los hilos vistos desde el punto de vista orientado a objetos pero una vez que se entiende no es complicado manejarlos

VLADIMIR AZPEITIA HERNÁNDEZ: Los hilos nos permite ejecutar rutinas de manera concurrente. Un proceso puede tener uno o más hilos. El ejercicio de la matriz me ayudó a ver que no siempre es útil trabajar con hilos. Y así mismo, el número de hilos que utilizamos dependiendo del problema y de de las capacidades de nuestra computadora, reducirá o aumentará el tiempo que tardará la máquina en realizar el programa. En nuestro caso después de los 4 hilos el tiempo aumentaba.