

INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

SISTEMAS OPERATIVOS

ADMINISTRADOR DE PROCESOS EN LINUX Y WINDOWS

*Práctica #4. Administrador de Procesos en
Linux y Windows (2)*

INTEGRANTES DEL EQUIPO:

COLIN RAMIRO JOEL
HERNÁNDEZ REYES JULIO CÉSAR
MALDONADO CERÓN CARLOS
MENDOZA GARCÍA ELIÚ EDUARDO

Grupo: 4CM1

PROFESOR: CORTÉS GALICIA JORGE

5 Octubre, 2021

I.INTRODUCCIÓN TEÓRICA

Podemos definir a un proceso como un programa ejecutándose dentro de su propio espacio de direcciones. O también lo podemos ver como instrucciones de un programa destinadas a ser ejecutadas por el microprocesador. Estos procesos se puede decir, son los “supervisores” de hilos de ejecución **fig 1**

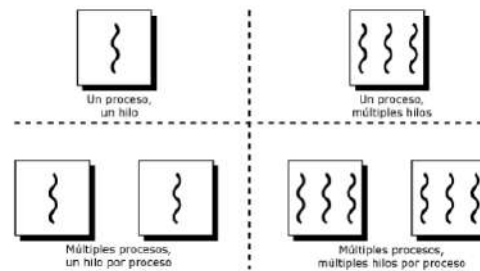


fig 1

Ahora bien un hilo es una unidad básica de utilización de CPU, la cual contiene un ID de hilo, su propio contador de programa,, un conjunto de registros, y una pila; que se representa a nivel del sistema operativo con una estructura llamada TCB (thread control block). Estos elementos comparten con otros hilos que pertenecen al mismo proceso la sección de código, la sección de datos, entre otras cosas. Si un proceso tiene múltiples hilos, puede realizar más de una tarea a la vez, esto es real cuando se posee más de un CPU.

Ventajas de la implementación de los hilos

- **Respuesta:** Su tiempo de respuesta mejora, ya que el programa puede continuar ejecutándose, aunque parte de él esté bloqueado.
- **Compartir recursos:** Los hilos comparten la memoria y los recursos del proceso al que pertenecen, por lo que se puede tener varios hilos de ejecución dentro del mismo espacio de direcciones.
- **Economía:** Es más fácil la creación, cambio de contexto y gestión de hilos que de procesos.
- **Utilización múltiples CPUs:** Permite que hilos de un mismo proceso se ejecuten en diferentes CPUs a la vez. En un proceso mono-hilo, un proceso se ejecuta en una única CPU, independientemente de cuántas tenga a su disposición.

Hasta ahora nos hemos referido a los hilos como algo genérico, pero a nivel práctico los hilos se pueden implementar de dos formas:

1. **Hilos a nivel Kernel.-** El SO es el que lo crea, planifica y gestiona los hilos. Se reconocen tantos hilos como se hayan creado.

Su beneficio principal es que pueden aprovechar mejor las arquitecturas de multiprocesadores, y que proporcionan un mejor tiempo de respuesta, ya que si un hilo se bloquea, los otros pueden seguir ejecutando.

2. **Hilos a nivel Usuario.-** Son implementados en alguna librería. Estos hilos se gestionan sin soporte del SO, el cual solo reconoce un hilo de ejecución.

Su principal beneficio es que su cambio de contexto es más sencillo que el cambio de contexto entre hilos de kernel. Además, se pueden implementar aún si el SO no utiliza hilos a nivel de kernel. Otro de los beneficios consiste en poder planificar diferente a la estrategia del SO.

II.DESARROLLO EXPERIMENTAL

1. A través de la ayuda en línea que proporciona Linux, investigamos el funcionamiento, sus argumentos y sus respectivos retornos de las funciones: `pthread_create()`, `pthread_join()`, `pthread_self()`, `pthread_exit()`, `scandir()`, `stat()` :

- ***pthread_create()***

- **Descripción.-** Inicia un nuevo hilo en la llamada al proceso. El nuevo hilo comienza la ejecución invocando ***start_routine ()***;
- **Argumentos.-** Son 4
 - ***pthread_t *thread***
 - ***const pthread_attr_t *attr***
 - ***void *(*start_routine)(void *)***
 - ***void *arg*** ;
- **Retorno.-** Si es exitoso, retorna un 0. Si ocurre un error, retorna el número del error y el contenido de **thread* será indefinido.



```
Joel@Joel-VirtualBox: ~
PTHREAD_CREATE(3)
NAME
pthread_create - create a new thread
SYNOPSIS
#include <pthread.h>

int pthread_create(pthread_t *thread, const pthread_attr_t *attr,
void *(*start_routine) (void *), void *arg);

Compile and link with -pthread.
DESCRIPTION
The pthread_create() function starts a new thread in the calling process. The new thread starts execution by invoking start_routine(); arg is passed as the sole argument of start_routine().

The new thread terminates in one of the following ways:

* It calls pthread_exit(3), specifying an exit status value that is available to another thread in the same process that calls pthread_join(3).

Manual page pthread_create(3) line 1 (press h for help or q to quit)
```

- ***pthread_join()***

- **Descripción.-** Espera por el hilo a que termine. Si ya ha terminado entonces la función regresa inmediatamente.
- **Argumentos.-** Son 2
 - ***pthread_t *thread***
 - ***void **retval***;
- **Retorno.-** Si es exitoso regresa un 0. Si ocurre un error retorna el número del error.

```
joel@joel-VirtualBox: ~  
PTHREAD_JOIN(3)      Linux Programmer's Manual      PTHREAD_JOIN(3)  
  
NAME  
    pthread_join - join with a terminated thread  
  
SYNOPSIS  
    #include <pthread.h>  
  
    int pthread_join(pthread_t thread, void **retval);  
  
    Compile and link with -pthread.  
  
DESCRIPTION  
    The pthread_join() function waits for the thread specified by thread to terminate. If that thread has already terminated, then pthread_join() returns immediately. The thread specified by thread must be joinable.  
  
    If retval is not NULL, then pthread_join() copies the exit status of the target thread (i.e., the value that the target thread supplied to pthread_exit(3)) into the location pointed to by retval. If the target thread was canceled, then PTHREAD_CANCELED is placed in the location pointed to by retval.  
  
Manual page pthread_join(3) line 1 (press h for help or q to quit)
```

- **pthread_self()**

- **Descripción.-** Devuelve el ID del hilo llamado.
- **Argumentos.-** Es 1
 - pthread_t pthread_self(void)
- **Retorno.-** Ya que la función siempre es exitosa, devuelve el ID del hilo llamado.

```
joel@joel-VirtualBox: ~  
PTHREAD_SELF(3)      Linux Programmer's Manual      PTHREAD_SELF(3)  
  
NAME  
    pthread_self - obtain ID of the calling thread  
  
SYNOPSIS  
    #include <pthread.h>  
  
    pthread_t pthread_self(void);  
  
    Compile and link with -pthread.  
  
DESCRIPTION  
    The pthread_self() function returns the ID of the calling thread. This is the same value that is returned in *thread in the pthread_create(3) call that created this thread.  
  
RETURN VALUE  
    This function always succeeds, returning the calling thread's ID.  
  
ERRORS  
    This function always succeeds.  
  
Manual page pthread_self(3) line 1 (press h for help or q to quit)
```

- **pthread_exit()**

- **Descripción.-** Termina el hilo llamado. Y devuelve un valor que está disponible para otro hilo.
- **Argumentos.-** Es 1
 - void *retval;
- **Retorno.-** No retorna ningún valor.

```
joel@joel-VirtualBox: ~  
PTHREAD_EXIT(3)      Linux Programmer's Manual      PTHREAD_EXIT(3)  
  
NAME  
    pthread_exit - terminate calling thread  
  
SYNOPSIS  
    #include <pthread.h>  
  
    void pthread_exit(void *retval);  
  
    Compile and link with -pthread.  
  
DESCRIPTION  
    The pthread_exit() function terminates the calling thread and returns a  
    value via retval that (if the thread is joinable) is available to an-  
    other thread in the same process that calls pthread_join(3).  
  
    Any clean-up handlers established by pthread_cleanup_push(3) that have  
    not yet been popped, are popped (in the reverse of the order in which  
    they were pushed) and executed. If the thread has any thread-specific  
    data, then, after the clean-up handlers have been executed, the corre-  
    sponding destructor functions are called, in an unspecified order.  
  
Manual page pthread_exit(3) line 1 (press h for help or q to quit)
```

- **scandir()**

- **Descripción.-** Escanea el directorio *dirp*, llamando *filter* en cada entrada del directorio.
- **Argumentos.- Son 4**
 - **const char *dirp**
 - **struct dirent ***namelist**
 - **int (*filter)(const struct dirent *)**
 - **int (*compar)(const struct dirent **, const struct dirent **)**
- **Retorno.-** Devuelve el número de entradas del directorio seleccionado, en caso de error se devuelve un -1.

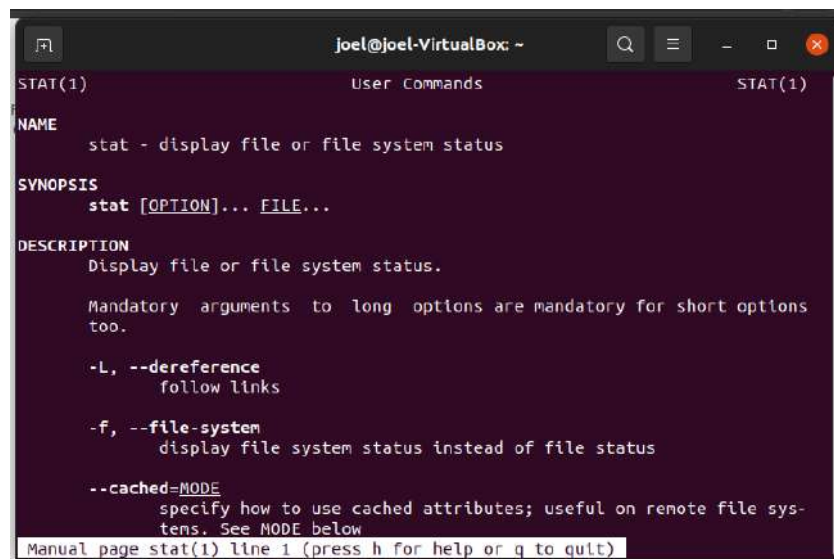
```
joel@joel-VirtualBox: ~  
SCANDIR(3)      Linux Programmer's Manual      SCANDIR(3)  
  
NAME  
    scandir, scandirat, alphasort, versionsort - scan a directory for  
    matching entries  
  
SYNOPSIS  
    #include <dirent.h>  
  
    int scandir(const char *dirp, struct dirent ***namelist,  
                int (*filter)(const struct dirent *),  
                int (*compar)(const struct dirent **, const struct dirent **));  
  
    int alphasort(const struct dirent **a, const struct dirent **b);  
  
    int versionsort(const struct dirent **a, const struct dirent **b);  
  
    #include <fcntl.h>      /* Definition of AT_* constants */  
    #include <dirent.h>  
  
    int scandirat(int dirfd, const char *dirp,  
                  struct dirent ***namelist,  
                  int (*filter)(const struct dirent *),  
                  int (*compar)(const struct dirent **, const struct dirent **));  
  
Manual page scandir(3) line 1 (press h for help or q to quit)
```

- **stat()**

- **Descripción.-** Muestra el archivo o el estado del sistema de archivos
- **Argumentos.- Son 2**
 - **const char *path**

- **struct stat *buf**

- **Retorno.-** Devuelve el archivo o el estado del sistema de archivos.



```
STAT(1)                                User Commands                                STAT(1)

NAME
    stat - display file or file system status

SYNOPSIS
    stat [OPTION]... FILE...

DESCRIPTION
    Display file or file system status.

    Mandatory arguments to long options are mandatory for short options too.

    -L, --dereference
        follow links

    -f, --file-system
        display file system status instead of file status

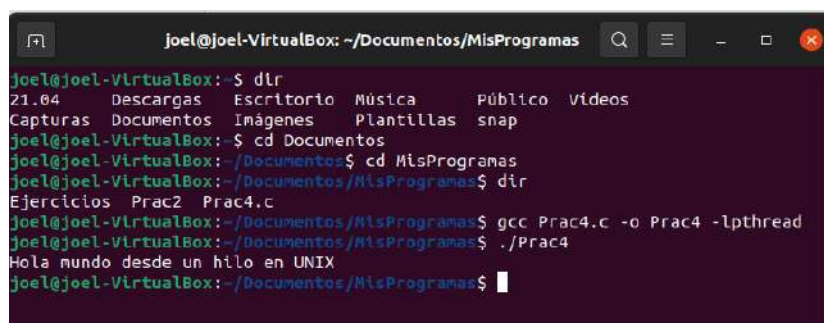
    --cached=MODE
        specify how to use cached attributes; useful on remote file systems. See MODE below

Manual page stat(1) line 1 (press h for help or q to quit)
```

2.Capturamos,compilamos y ejecutamos el programa de creación de un nuevo hilo en Linux.



```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <pthread.h>
4
5 void *hilo(void *arg);
6 int main(void){
7     pthread_t id_hilo;
8     pthread_create(&id_hilo, NULL, (void*)hilo, NULL);
9     pthread_join(id_hilo, NULL);
10    return 0;
11 }
12 void *hilo(void *arg){
13
14     printf("Hola mundo desde un hilo en UNIX\n");
15
16
17     return NULL;
18 }
```

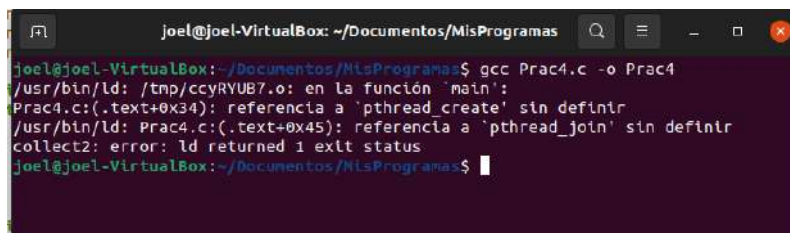


```
joel@joel-VirtualBox: ~/Documentos/MisProgramas
joel@joel-VirtualBox:~$ dir
21.04  Descargas  Escritorio  Música    Público  Videos
Capturas  Documentos  Imágenes   Plantillas snap
joel@joel-VirtualBox:~$ cd Documentos
joel@joel-VirtualBox:~/Documentos$ cd MisProgramas
joel@joel-VirtualBox:~/Documentos/MisProgramas$ dir
Ejercicios Prac2 Prac4.c
joel@joel-VirtualBox:~/Documentos/MisProgramas$ gcc Prac4.c -o Prac4 -lpthread
joel@joel-VirtualBox:~/Documentos/MisProgramas$ ./Prac4
Hola mundo desde un hilo en UNIX
joel@joel-VirtualBox:~/Documentos/MisProgramas$
```

Podemos observar que en la función main se invoca a las llamadas pthread_create cuyo argumento manda a llamar a la función previamente declarada en el programa **void *hilo** y pthread_join espera a que termine la ejecución para poder imprimir en pantalla el mensaje previamente establecido.

Algo que observamos en la ejecución del programa es que debimos agregar la sentencia **-lpthread** para su funcionamiento ya que de lo contrario marcaba un error, más específicamente, este error mencionaba que no había nada que referenciara a las funciones

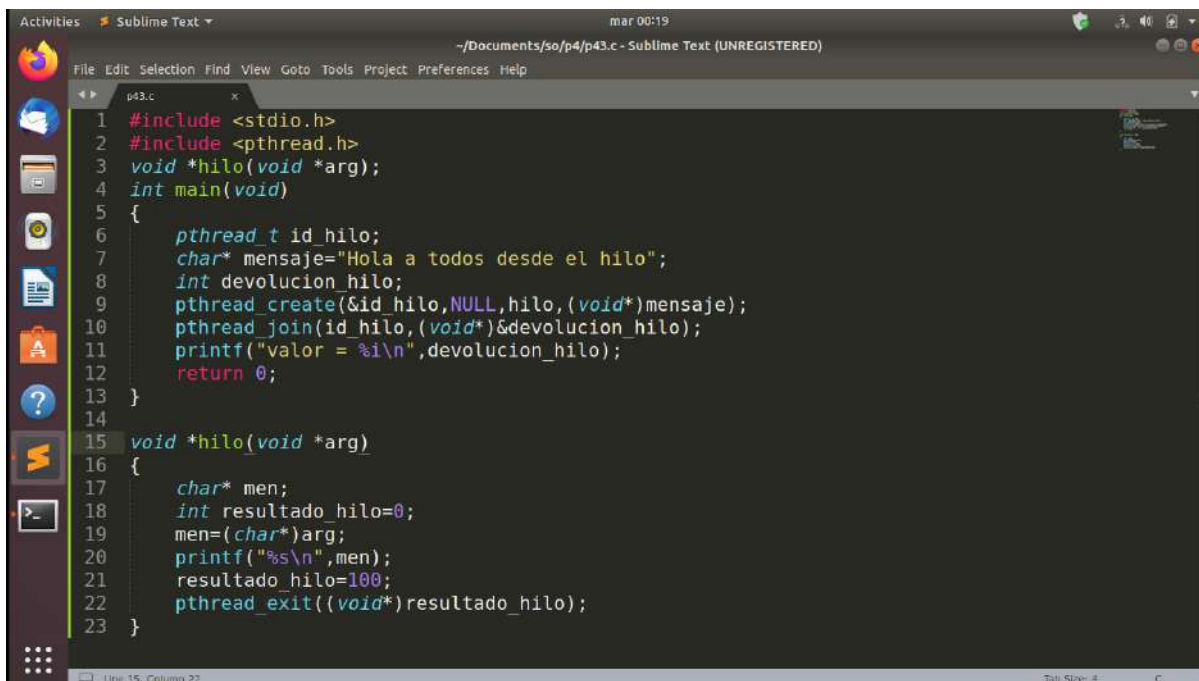
pthread_create y pthread_join más concretamente no había una definición de estas funciones, para su funcionamiento, al agregar esta sentencia, funcionaba correctamente el programa.



```
joel@joel-VirtualBox: ~/Documentos/MisProgramas
joel@joel-VirtualBox:~/Documentos/MisProgramas$ gcc Prac4.c -o Prac4
/usr/bin/ld: /tmp/ccyRYUB7.o: en la función 'main':
Prac4.c:(.text+0x34): referencia a 'pthread_create' sin definir
/usr/bin/ld: Prac4.c:(.text+0x45): referencia a 'pthread_join' sin definir
collect2: error: ld returned 1 exit status
joel@joel-VirtualBox:~/Documentos/MisProgramas$
```

3. Capturar, compilar y ejecutar el programa de creación de hilos en Linux. Observar su funcionamiento.

Captura de código



```
1 #include <stdio.h>
2 #include <pthread.h>
3 void *hilo(void *arg);
4 int main(void)
5 {
6     pthread_t id_hilo;
7     char* mensaje="Hola a todos desde el hilo";
8     int devolucion_hilo;
9     pthread_create(&id_hilo,NULL,hilo,(void*)mensaje);
10    pthread_join(id_hilo,(void*)&devolucion_hilo);
11    printf("valor = %i\n",devolucion_hilo);
12    return 0;
13 }
14
15 void *hilo(void *arg)
16 {
17     char* men;
18     int resultado_hilo=0;
19     men=(char*)arg;
20     printf("%s\n",men);
21     resultado_hilo=100;
22     pthread_exit((void*)resultado_hilo);
23 }
```

Compilación y ejecución



```
conner02kent@B04-VirtualBox: ~/Documents/so/p4
conner02kent@B04-VirtualBox:~/Documents/so/p4$ gcc -pthread p43.c
p43.c: In function 'hilo':
p43.c:22:15: warning: cast to pointer from integer of different size [-Wint-to-p
ointer-cast]
pthread_exit((void*)resultado_hilo);
^
conner02kent@B04-VirtualBox:~/Documents/so/p4$ ls
a.out  p43.c
conner02kent@B04-VirtualBox:~/Documents/so/p4$ ./a.out
Hola a todos desde el hilo
valor = 100
conner02kent@B04-VirtualBox:~/Documents/so/p4$
```

Similar al punto anterior, las impresiones en pantalla del mensaje y el valor, se imprimen desde el hilo.

Para poder compilar, se añadió -pthread a la compilación, de lo contrario, se obtenía el error de referencia no definida.

4. Capturar, compilar y ejecutar el programa de creación de hilos en Windows. Observar su funcionamiento.

Captura de código

```
p44.c
1  #include <windows.h>
2  #include <stdio.h>
3  DWORD WINAPI funcionHilo(LPVOID lpParam);
4  typedef struct Informacion info;
5  struct Informacion
6  {
7      int val_1;
8      int val_2;
9  };
10 int main(void)
11 {
12     DWORD idHilo;           //Identificador del Hilo
13     HANDLE manHilo;         //Manejador del Hilo
14     info argumentos;
15     argumentos.val_1=10;
16     argumentos.val_2=100;
17     // Creacion del hilo
18     manHilo=CreateThread(NULL, 0, funcionHilo, &argumentos, 0, &idHilo);
19     // Espera la finalización del hilo
20     WaitForSingleObject(manHilo, INFINITE);
21     printf("Valores al salir del Hilo: %i %i\n", argumentos.val_1, argumentos.val_2);
22     // Cierre del manejador del hilo creado
23     CloseHandle(manHilo);
24     return 0;
25 }
26
27 DWORD WINAPI funcionHilo(LPVOID lpParam)
28 {
29     info *datos = (info *)lpParam;
30     printf("Valores al entrar al Hilo: %i %i\n", datos->val_1, datos->val_2);
31     datos->val_1*=2;
32     datos->val_2*=2;
33     return 0;
34 }
```

Compilación y ejecución

```
C:\Users\Windows 10\Documents\ESCOMaywey\4\SOperativos\p4>gcc p44.c
C:\Users\Windows 10\Documents\ESCOMaywey\4\SOperativos\p4>a
Valores al entrar al Hilo: 10 100
Valores al salir del Hilo: 20 200
C:\Users\Windows 10\Documents\ESCOMaywey\4\SOperativos\p4>
```

El código resulta algo intuitivo y sencillo de entender.

Los valores iniciales se envían al hilo, durante la ejecución del hilo, se imprimen dichos valores iniciales y se multiplican por 2. Así al salir del hilo, volver al main y volver a imprimir los valores, estos ya no serán igual a los iniciales, sino que habrán sido modificados dentro del hilo.

A diferencia de la compilación en linux, en windows no fue necesario añadir algo como -pthread en la compilación.

5. Programe una aplicación (tanto en Linux como en Windows), que cree un proceso hijo a partir de un proceso padre, el hijo creado a su vez creará 15 hilos. A su vez cada uno de los 15 hilos creará 10 hilos más. A su vez cada uno de los 10 hilos creará 5 hilos más. Cada uno de los hilos creados imprimirá en pantalla “Práctica 4” si se trata de un hilo terminal o los identificadores de los hilos creados si se trata de un proceso o hilo padre.

Código en Linux

```

Actividades Editor de textos mar 16:11
punto5.c
Guardar

//Si es un hilo terminal imprimir "Practica4"
//Si es un hilo padre imprimir los id de los hilos creados si es un hilo padre

//pid == 0 <- Proceso Hijo
//pid < 0 <- Proceso Padre
//pid == -1 <-Error
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

#include <unistd.h>
#include <sys/types.h>
#include <string.h>

void *hilopadre(void *threadid);
void *hilointermedios(void *threadid);
void *hiloterminales(void *threadid);

int main(){
    pid_t pid;

    pid = fork();
    pthread_t hilospadre[15];

    if(pid == 0){
        printf("Soy el proceso hijo(%d), hijo de(%d).\n",getpid(),getppid());
    }
}
C Anchura del tabulador: 8 Ln 40, Col 24 INS

```

```

Actividades Editor de textos mar 16:20
punto5.c
Guardar

long i;
for( i = 1; i <= 15; i++){
    pthread_create(&hilospadre[i],NULL,hilopadre,
    pthread_join(hilospadre[i],NULL);
    pthread_exit(NULL);
}

} else{
    printf("Soy el proceso padre(%d), hijo de(%d).\n",getpid(),getppid());
    if (pid != 0){
        wait();
    }
}

void *hilopadre(void *threadid){
    long tid;
    tid = (long)threadid;
    printf("Hilo: %ld, Hilo ID ->%lu\n",tid,thread_self());
    //Creacion de los 10 hilos
    pthread_t hilos10[10];
    long i;
    for( i = 1; i <= 10; i++){
        pthread_create(&hilos10[i],NULL,hilointermedios,(void *) i);
    }
    pthread_join(hilos10[i],NULL);
}
C Anchura del tabulador: 8 Ln 40, Col 24 INS

```

```

Actividades Editor de textos mar 16:21
punto5.c
Guardar

*) i);
pthread_join(hilos10[i],NULL);
pthread_exit(NULL);

pthread_exit(NULL); //Finaliza la ejecución del hilo
}

void *hilointermedios(void *threadid){
    long tid;
    tid = (long)threadid;
    printf("Hilo: %ld, Hilo ID ->%lu\n",tid,thread_self());
    //Creacion de los 5 hilos
    pthread_t hilos5[5];
    long i;
    for( i = 1; i <= 5; i++){
        pthread_create(&hilos5[i],NULL,hiloterminales,(void *) i);
    }
    pthread_join(hilos5[i],NULL);
    pthread_exit(NULL); //Finaliza la ejecución del hilo
}

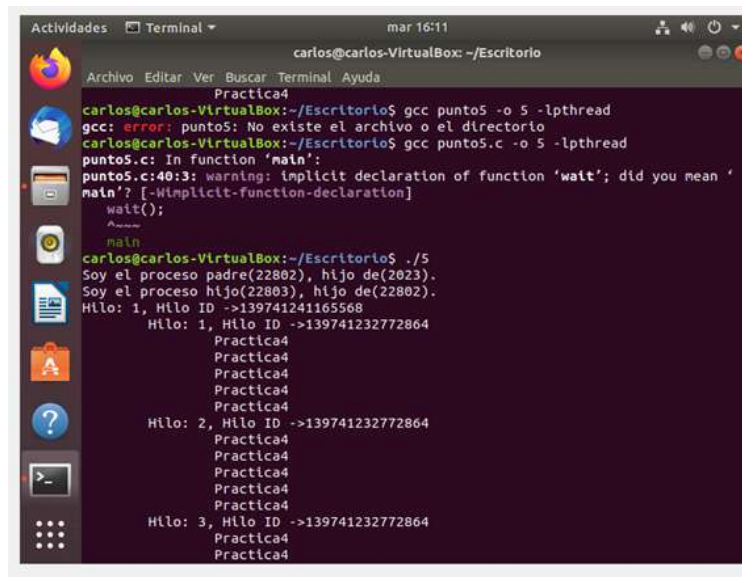
void *hiloterminales(void *threadid){
    printf("Practica4\n");
    pthread_exit(NULL); //Finaliza la ejecución del hilo
}
C Anchura del tabulador: 8 Ln 40, Col 24 INS

```

para compilar el archivo escribimos lo siguiente:

gcc punto5.c -o (nombre del ejecutable) -pthread

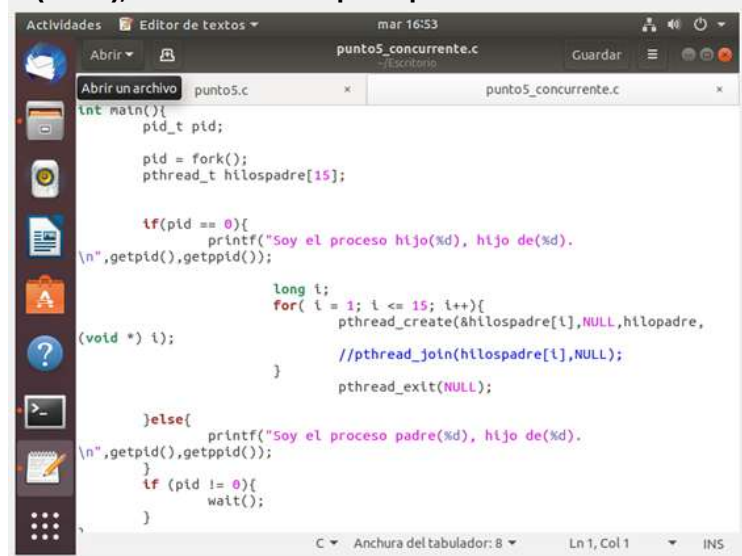
./(ejecutable)



```
Actividades Terminal mar 16:11
carlos@carlos-VirtualBox: ~/Escritorio
Practica4
carlos@carlos-VirtualBox:~/Escritorio$ gcc punto5 -o 5 -pthread
gcc: error: punto5: No existe el archivo o el directorio
carlos@carlos-VirtualBox:~/Escritorio$ gcc punto5.c -o 5 -pthread
punto5.c:40:3: warning: implicit declaration of function 'wait'; did you mean '
main'? [-Wimplicit-function-declaration]
    wait();
    ^~~~~
    main
carlos@carlos-VirtualBox:~/Escritorio$ ./5
Soy el proceso padre(22802), hijo de(2023).
Soy el proceso hijo(22803), hijo de(22802).
Hilo: 1, Hilo ID ->139741241165568
    Hilo: 1, Hilo ID ->139741232772864
        Practica4
        Practica4
        Practica4
        Practica4
        Practica4
    Hilo: 2, Hilo ID ->139741232772864
        Practica4
        Practica4
        Practica4
        Practica4
        Practica4
    Hilo: 3, Hilo ID ->139741232772864
        Practica4
        Practica4
```

Código concurrente

La cuestión para que quede concurrente es que los pthread _create pueden primero y los pthread_join queden hasta el final. Pero como este caso es un ciclo no se usan los pthread_join sino un pthread_exit(NULL); al final del ciclo para que todos los hilos terminen.



```
Actividades Editor de textos mar 16:53
punto5_concurrente.c
Abrir un archivo punto5.c x punto5_concurrente.c x
Guardar
int main(){
    pid_t pid;

    pid = fork();
    pthread_t hilospadre[15];

    if(pid == 0){
        printf("Soy el proceso hijo(%d), hijo de(%d).
\n",getpid(),getppid());

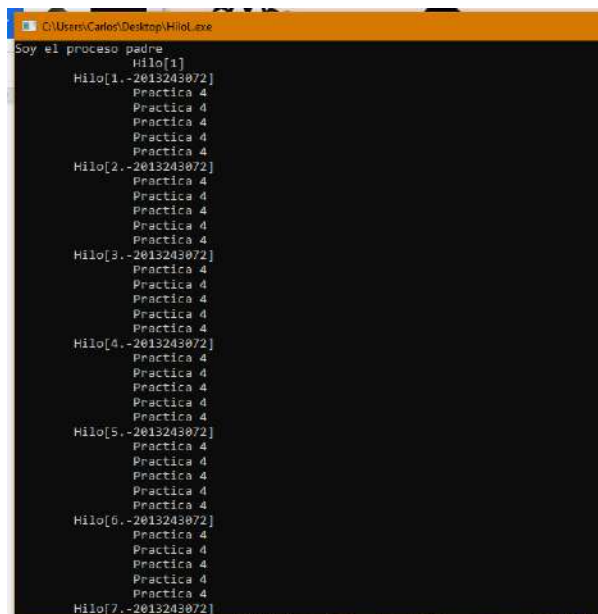
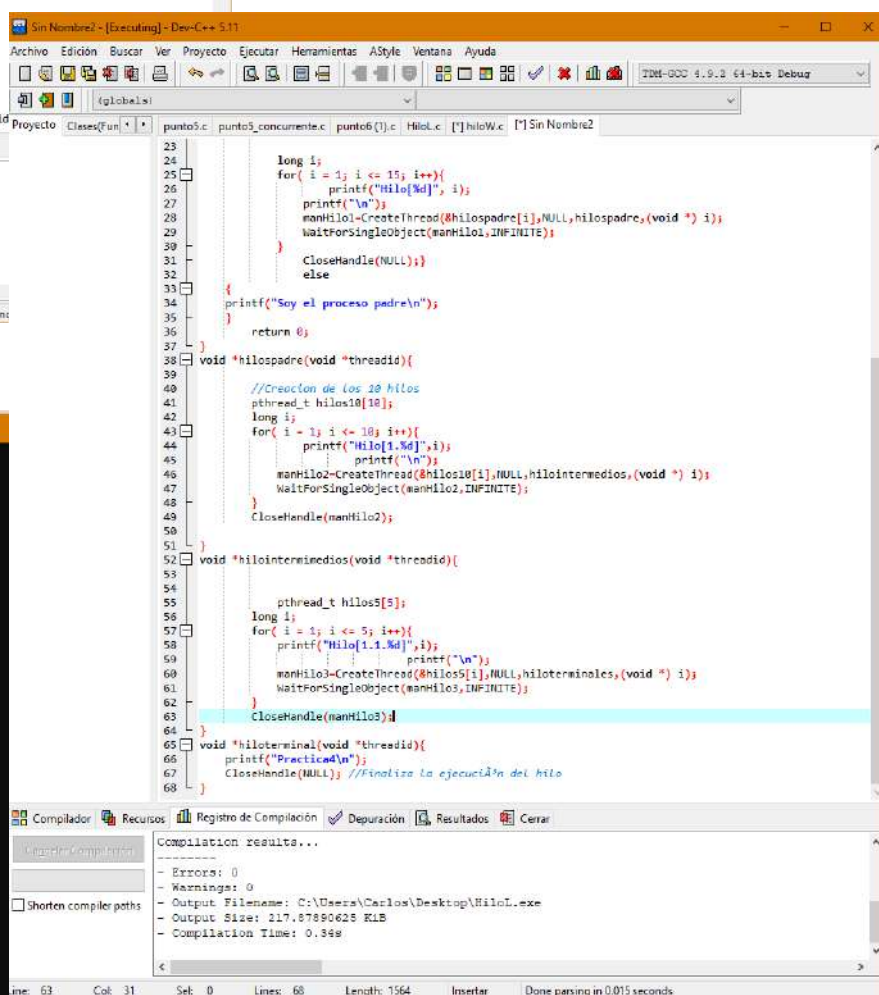
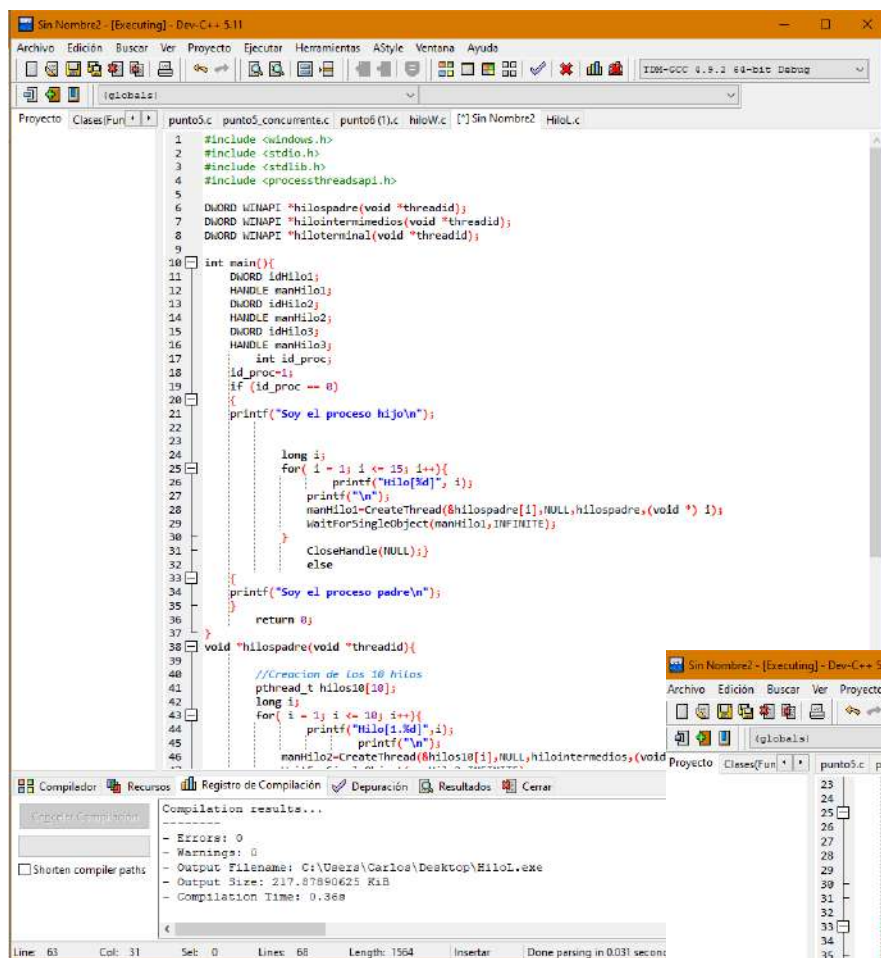
        long i;
        for( i = 1; i <= 15; i++){
            pthread_create(&hilospadre[i],NULL,hilopadre,
(void *) i);

            //pthread_join(hilospadre[i],NULL);

        }
        pthread_exit(NULL);

    }else{
        printf("Soy el proceso padre(%d), hijo de(%d).
\n",getpid(),getppid());
    }
    if (pid != 0){
        wait();
    }
}
```

Windows Código/Ejecución



6. Se programó la misma aplicación del punto 5 de la práctica 3 pero utilizando hilos en vez de procesos.

La aplicación crea 5 hilos

Hilo 1: Suma de dos matrices de 7x7 elementos de tipo entero

Hilo 2: Resta sobre esas mismas matrices

Hilo 3: Multiplicación de matrices

Hilo 4: Obtener la transpuesta de cada matriz

Hilo 5: Leer los archivos de salida de los hilos anteriores y mostrarlos en pantalla

VERSIÓN LINUX

CÓDIGO En forma secuencial

```
punto6.c
1 //Aplicacion que crea 5 hilos
2 //Hilo 1: Suma de dos matrices de 7x7 elementos de tipo entero
3 //Hilo 2: Resta sobre esas mismas matrices
4 //Hilo 3: Multiplicación de matrices
5 //Hilo 4: Obtener la transpuesta de cada matriz
6 //Hilo 5: Leera lo archivos de resultados y los mostrara en pantalla
7 //Los hilos del 1 -> 4 deberan crear un .txt con las salidas
8 //-----
9 #include <stdio.h>
10 #include <stdlib.h>
11 #include <pthread.h>
12 void hilo1();
13 void hilo2();
14 void hilo3();
15 void hilo4();
16 void hilo5();
17
18 int m1[7][7] = {
19     {1,2,3,4,5,6,7},
20     {2,4,5,6,2,1,4},
21     {7,4,2,6,3,1,3},
22     {5,1,2,6,3,2,4},
23     {1,3,2,5,4,7,6},
24     {3,4,2,3,1,5,2},
25     {7,6,5,4,3,2,1}
26 };
27 int m2[7][7] = {
28     {7,4,2,6,3,1,3},
29     {5,1,2,6,3,2,4},
30     {1,2,3,4,5,6,7},
31     {7,6,5,4,3,2,1},
32     {3,4,2,3,1,5,2},
33     {1,3,2,5,4,7,6},
34     {2,4,5,6,2,1,4}
35 };
36 int mr[7][7];
37
38 int main(){
39
40     pthread_t th1,th2,th3,th4,th5; //Direcciones de los 5 hilos(threads)
41
42     //Creacion de los hilos
43     pthread_create(&th1,NULL,(void*)hilo1,NULL); //Hilo de la Suma
44     pthread_join(th1,NULL); //Esperar a hilo1
45
46     pthread_create(&th2,NULL,(void*)hilo2,NULL); //Hilo de la Resta
47     pthread_join(th2,NULL); //Esperar a hilo2
48
49     pthread_create(&th3,NULL,(void*)hilo3,NULL); //Hilo de la Multiplicación
50     pthread_join(th3,NULL); //Esperar a hilo3
51
52     pthread_create(&th4,NULL,(void*)hilo4,NULL); //Hilo de la Transpuesta
```



```

53 pthread_join(th4, NULL); //Esperar a hilo4
54
55 //Se crea el hilo 5 y se espera su finalización
56 pthread_create(&th5, NULL, (void*)hilo5, NULL); //Hilo de los .txt
57 pthread_join(th5, NULL); //Esperar a hilo5
58
59
60 }
61 void hilo1(){
62
63     FILE *th1 = NULL;
64     th1 = fopen("Hilo1.txt", "w+");
65     if(th1 == NULL){
66         printf("No fue posible abrir el archivo\n");
67     }
68     //Se imprime el Id. del Hilo
69     fprintf(th1, "Soy el primer Hilo: %lx\n", pthread_self());
70     fprintf(th1, "Suma de matrices\n");
71
72     for(int i = 0; i < 7; i++){
73         //matriz 1
74         for(int j = 0; j < 7; j++){
75             if(j == 0){
76                 fprintf(th1, "| ");
77             }
78             fprintf(th1, "%d ", m1[i][j]);
79             if(j == 6){
80                 fprintf(th1, " |");
81             }
82         }
83         if (i == 3){fprintf(th1, " + ");} else {fprintf(th1, " ");}
84         //matriz 2
85         for(int j = 0; j < 7; j++){
86             if(j == 0){
87                 fprintf(th1, "| ");
88             }
89             fprintf(th1, "%d ", m2[i][j]);
90             if(j == 6){
91                 if(i == 3){
92                     fprintf(th1, " | =\n");
93                 }
94                 else{
95                     fprintf(th1, " |\n");
96                 }
97             }
98         }
99     }
100     fprintf(th1, "\n");
101     for(int i = 0; i < 7; i++){
102         for(int j = 0; j < 7; j++){
103             if(j == 0){
104                 fprintf(th1, "| ");
105             }
106             mr[i][j] = m1[i][j] + m2[i][j];
107             fprintf(th1, "%d ", mr[i][j]);
108         }
109         fprintf(th1, " |\n");
110     }
111     fclose(th1); //Se cierra el archivo.txt
112
113     printf("Inicio Hilo 1\n");
114     printf("Archivo creado\nFin Hilo 1\n");
115     printf("\n");

```

```

116     pthread_exit(NULL); //Finaliza la ejecución del hilo
117 }
118
119 void hilo2(){
120
121     FILE *th2 = NULL;
122     th2 = fopen("Hilo2.txt","w+");
123     if(th2 == NULL){
124         printf("No fue posible abrir el archivo\n");
125     }
126     //Se imprime el Id. del Hilo
127     fprintf(th2,"Soy el segundo Hilo: %lx\n",pthread_self());
128     fprintf(th2,"Resta de matrices\n");
129
130     for(int i = 0; i < 7; i++){
131         //matriz 1
132         for(int j = 0; j < 7; j++){
133             if(j == 0){
134                 fprintf(th2,"|   ");
135             }
136             fprintf(th2,"%d ",m1[i][j]);
137             if(j == 6){
138                 fprintf(th2," |");
139             }
140         }
141         if (i == 3){fprintf(th2," - ");}else{fprintf(th2," ");}
142         //matriz 2
143         for(int j = 0; j < 7; j++){
144             if(j == 0){
145                 fprintf(th2,"|   ");
146             }
147             fprintf(th2,"%d ",m2[i][j]);
148             if(j == 6){
149                 if(i == 3){
150                     fprintf(th2," | =\n");
151                 }
152                 else{
153                     fprintf(th2," |\n");
154                 }
155             }
156         }
157     }
158     fprintf(th2,"\n");
159     for(int i = 0; i < 7; i++){
160         for(int j = 0; j < 7; j++){
161             if(j == 0){
162                 fprintf(th2,"|   ");
163             }
164             mr[i][j] = m1[i][j] - m2[i][j];
165             fprintf(th2,"%d ",mr[i][j]);
166         }
167         fprintf(th2,"|\n");
168     }
169     fclose(th2); //Se cierra el archivo.txt
170
171     printf("Inicio Hilo 2\n");
172     printf("Archivo creado\nFin Hilo 2\n");
173     printf("\n");
174     pthread_exit(NULL); //Finaliza la ejecución del hilo
175 }
176 void hilo3(){
177
178     FILE *th3 = NULL;

```



```

179 th3 = fopen("Hilo3.txt","w+");
180 if(th3 == NULL){
181     printf("No fue posible abrir el archivo\n");
182 }
183 //Se imprime el Id. del Hilo
184 fprintf(th3,"Soy el tercer Hilo: %lx\n",pthread_self());
185 fprintf(th3,"Multiplicación de matrices\n");
186
187 for(int i = 0; i < 7; i++){
188     //matriz 1
189     for(int j = 0; j < 7; j++){
190         if(j == 0){
191             fprintf(th3,"| ");
192         }
193         fprintf(th3,"%d ",m1[i][j]);
194         if(j == 6){
195             fprintf(th3," |");
196         }
197     }
198     if (i == 3){fprintf(th3," * ");}else{fprintf(th3," ");}
199     //matriz 2
200     for(int j = 0; j < 7; j++){
201         if(j == 0){
202             fprintf(th3,"| ");
203         }
204         fprintf(th3,"%d ",m2[i][j]);
205         if(j == 6){
206             if(i == 3){
207                 fprintf(th3," | =\n");
208             }
209             else{
210                 fprintf(th3," |\n");
211             }
212         }
213     }
214 }
215 fprintf(th3,"\n");
216 for(int a = 0; a < 7; a++){
217     for(int i = 0; i < 7; i++){
218         int suma = 0;
219         for(int j = 0; j < 7; j++){
220             suma +=(m1[i][j] * m2[j][a]);
221         }
222         mr[i][a] = suma;
223     }
224 }
225
226 for(int i = 0; i < 7; i++){
227     for(int j = 0; j < 7; j++){
228         if(j == 0){
229             fprintf(th3,"| ");
230         }
231         fprintf(th3,"%d ",mr[i][j]);
232     }
233     fprintf(th3,"|\n");
234 }
235 fclose(th3);//Se cierra el archivo.txt
236
237 printf("Inicio Hilo 3\n");
238 printf("Archivo creado\nFin Hilo 3\n");
239 printf("\n");
240 pthread_exit(NULL); //Finaliza la ejecución del hilo

```

```

241 }
242 void hilo4(){
243
244     FILE *th4 = NULL;
245     th4 = fopen("Hilo4.txt","w+");
246     if(th4 == NULL){
247         printf("No fue posible abrir el archivo\n");
248     }
249     //Se imprime el Id. del Hilo
250     fprintf(th4,"Soy el cuarto Hilo: %lx\n",pthread_self());
251     fprintf(th4,"Transpuesta de matrices\n");
252
253     //Transpuesta de la matriz 1
254     for(int i = 0; i < 7; i++){
255         for(int j = 0; j < 7; j++){
256             if(j == 0){
257                 fprintf(th4,"|  ");
258             }
259             fprintf(th4,"%d  ",m1[i][j]);
260             if(j == 6){
261                 fprintf(th4,"  |");
262             }
263         }
264         if (i == 0){fprintf(th4,"T  ");}
265         else if (i == 3){fprintf(th4," = ");}
266         else{fprintf(th4,"  ");}
267         //matriz 2
268         for(int j = 0; j < 7; j++){
269             if(j == 0){
270                 fprintf(th4,"|  ");
271             }
272             fprintf(th4,"%d  ",m1[j][i]);
273             if(j == 6){
274                 fprintf(th4,"  |\n");
275             }
276         }
277     }
278     fprintf(th4,"\n");
279     //Transpuesta de la matriz 2
280     for(int i = 0; i < 7; i++){
281         for(int j = 0; j < 7; j++){
282             if(j == 0){
283                 fprintf(th4,"|  ");
284             }
285             fprintf(th4,"%d  ",m2[i][j]);
286             if(j == 6){
287                 fprintf(th4,"  |");
288             }
289         }
290         if (i == 0){fprintf(th4,"T  ");}
291         else if (i == 3){fprintf(th4," = ");}
292         else{fprintf(th4,"  ");}
293         //matriz 2
294         for(int j = 0; j < 7; j++){
295             if(j == 0){
296                 fprintf(th4,"|  ");
297             }
298             fprintf(th4,"%d  ",m2[j][i]);
299             if(j == 6){
300                 fprintf(th4,"  |\n");
301             }
302         }
303     }
304 }

```

```

304     fprintf(th4, "\n");
305
306     fclose(th4); //Se cierra el archivo.txt
307
308     printf("Inicio Hilo 4\n");
309     printf("Archivo creado\nFin Hilo 4\n");
310     printf("\n");
311     pthread_exit(NULL); //Finaliza la ejecución del hilo
312 }
313 void hilo5(){
314     printf("Inicio Hilo 5\n");
315     //Se imprime el Id. del Hilo
316     printf("Soy el quinto Hilo: %lx\n", pthread_self());
317     printf("Lectura de los archivos creados de los hilos anteriores\n");
318
319     char c;
320     FILE *archivo1 = fopen("Hilo1.txt", "r");
321     if(archivo1 != NULL){
322         printf("\n***Hilo1.txt***\n");
323         while(1){
324             c = fgetc(archivo1);
325             if(feof(archivo1)){
326                 break;
327             }
328             printf("%c", c);
329         }
330     }else{
331         printf("Hubo un error al abrir el archivo o es inexistente.");
332     }
333     fclose(archivo1);
334
335     FILE *archivo2 = fopen("Hilo2.txt", "r");
336     if(archivo2 != NULL){
337         printf("\n***Hilo2.txt***\n");
338         while(1){
339             c = fgetc(archivo2);
340             if(feof(archivo2)){
341                 break;
342             }
343             printf("%c", c);
344         }
345     }else{
346         printf("Hubo un error al abrir el archivo o es inexistente.");
347     }
348     fclose(archivo2);
349
350     FILE *archivo3 = fopen("Hilo3.txt", "r");
351     if(archivo3 != NULL){
352         printf("\n***Hilo3.txt***\n");
353         while(1){
354             c = fgetc(archivo3);
355             if(feof(archivo3)){
356                 break;
357             }
358             printf("%c", c);
359         }
360     }else{
361         printf("Hubo un error al abrir el archivo o es inexistente.");
362     }
363     fclose(archivo3);
364
365     FILE *archivo4 = fopen("Hilo4.txt", "r");

```



```

366     if(archivo4!=NULL){
367         printf("\n***Hilo4.txt***\n");
368         while(1){
369             c=fgetc(archivo4);
370             if(feof(archivo4)){
371                 break;
372             }
373             printf("%c",c);
374         }
375     }else{
376         printf("Hubo un error al abrir el archivo o es inexistente.");
377     }
378     fclose(archivo4);
379
380     printf("Fin Hilo 5\n");
381     pthread_exit(NULL); //Finaliza la ejecución del hilo
382 }

```

COMPILACIÓN Y EJECUCIÓN

```

cesar@cesar-HP-Notebook: ~/Documentos/SO/Practica4
cesar@cesar-HP-Notebook:~/Documentos/SO/Practica4$ gcc punto6.c -o punto6 -lpthread
cesar@cesar-HP-Notebook:~/Documentos/SO/Practica4$ ./punto6
Inicio Hilo 1
Archivo creado
Fin Hilo 1

Inicio Hilo 2
Archivo creado
Fin Hilo 2

Inicio Hilo 3
Archivo creado
Fin Hilo 3

Inicio Hilo 4
Archivo creado
Fin Hilo 4

```

```

Inicio Hilo 5
Soy el quinto Hilo: 7f8d2e11d700
Lectura de los archivos creados de los hilos anteriores

***Hilo1.txt***
Soy el primer Hilo: 7f8d2e11d700
Suma de matrices

```

| | | | | | | | | | | | | | | | |
|---|----|---|----|---|----|---|---|----|---|---|----|---|----|---|--|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | | 7 | 4 | 2 | 6 | 3 | 1 | 3 | |
| 2 | 4 | 5 | 6 | 2 | 1 | 4 | | 5 | 1 | 2 | 6 | 3 | 2 | 4 | |
| 7 | 4 | 2 | 6 | 3 | 1 | 3 | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
| 5 | 1 | 2 | 6 | 3 | 2 | 4 | + | 7 | 6 | 5 | 4 | 3 | 2 | 1 | |
| 1 | 3 | 2 | 5 | 4 | 7 | 6 | | 3 | 4 | 2 | 3 | 1 | 5 | 2 | |
| 3 | 4 | 2 | 3 | 1 | 5 | 2 | | 1 | 3 | 2 | 5 | 4 | 7 | 6 | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | | 2 | 4 | 5 | 6 | 2 | 1 | 4 | |
| | | | | | | | | | | | | | | | |
| | 8 | | 6 | | 5 | | | 10 | 8 | | 7 | | 10 | | |
| | 7 | | 5 | | 7 | | | 12 | 5 | | 3 | | 8 | | |
| | 8 | | 6 | | 5 | | | 10 | 8 | | 7 | | 10 | | |
| | 12 | | 7 | | 7 | | | 10 | 6 | | 4 | | 5 | | |
| | 4 | | 7 | | 4 | | | 8 | 5 | | 12 | | 8 | | |
| | 4 | | 7 | | 4 | | | 8 | 5 | | 12 | | 8 | | |
| | 9 | | 10 | | 10 | | | 10 | 5 | | 3 | | 5 | | |

Hilo2.txt

Soy el segundo Hilo: 7f8d2e11d700

Resta de matrices

| | | | | | | | | | | | | | | | | |
|---|----|---|----|---|----|---|---|--|----|---|----|---|----|---|----|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | | | 7 | 4 | 2 | 6 | 3 | 1 | 3 | |
| 2 | 4 | 5 | 6 | 2 | 1 | 4 | | | 5 | 1 | 2 | 6 | 3 | 2 | 4 | |
| 7 | 4 | 2 | 6 | 3 | 1 | 3 | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
| 5 | 1 | 2 | 6 | 3 | 2 | 4 | - | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | = |
| 1 | 3 | 2 | 5 | 4 | 7 | 6 | | | 3 | 4 | 2 | 3 | 1 | 5 | 2 | |
| 3 | 4 | 2 | 3 | 1 | 5 | 2 | | | 1 | 3 | 2 | 5 | 4 | 7 | 6 | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | | | 2 | 4 | 5 | 6 | 2 | 1 | 4 | |
| | | | | | | | | | | | | | | | | |
| | -6 | | -2 | | 1 | | | | -2 | | 2 | | 5 | | 4 | |
| | -3 | | 3 | | 3 | | | | 0 | | -1 | | -1 | | 0 | |
| | 6 | | 2 | | -1 | | | | 2 | | -2 | | -5 | | -4 | |
| | -2 | | -5 | | -3 | | | | 2 | | 0 | | 0 | | 3 | |
| | -2 | | -1 | | 0 | | | | 2 | | 3 | | 2 | | 4 | |
| | 2 | | 1 | | 0 | | | | -2 | | -3 | | -2 | | -4 | |
| | 5 | | 2 | | 0 | | | | -2 | | 1 | | 1 | | -3 | |

Hilo3.txt

Soy el tercer Hilo: 7f8d2e11d700

Multiplicación de matrices

| | | | | | | | | | | | | | | | | |
|---|-----|---|-----|---|----|---|---|--|-----|---|----|---|-----|---|-----|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | | | 7 | 4 | 2 | 6 | 3 | 1 | 3 | |
| 2 | 4 | 5 | 6 | 2 | 1 | 4 | | | 5 | 1 | 2 | 6 | 3 | 2 | 4 | |
| 7 | 4 | 2 | 6 | 3 | 1 | 3 | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
| 5 | 1 | 2 | 6 | 3 | 2 | 4 | * | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | = |
| 1 | 3 | 2 | 5 | 4 | 7 | 6 | | | 3 | 4 | 2 | 3 | 1 | 5 | 2 | |
| 3 | 4 | 2 | 3 | 1 | 5 | 2 | | | 1 | 3 | 2 | 5 | 4 | 7 | 6 | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | | | 2 | 4 | 5 | 6 | 2 | 1 | 4 | |
| | | | | | | | | | | | | | | | | |
| | 83 | | 102 | | 92 | | | | 133 | | 79 | | 105 | | 110 | |
| | 96 | | 85 | | 83 | | | | 115 | | 75 | | 73 | | 89 | |
| | 129 | | 99 | | 81 | | | | 130 | | 74 | | 64 | | 81 | |
| | 103 | | 95 | | 78 | | | | 111 | | 65 | | 64 | | 73 | |
| | 90 | | 102 | | 91 | | | | 135 | | 81 | | 104 | | 108 | |
| | 76 | | 65 | | 57 | | | | 102 | | 65 | | 71 | | 82 | |
| | 125 | | 90 | | 76 | | | | 139 | | 89 | | 87 | | 106 | |

Hilo4.txt

Soy el cuarto Hilo: 7f8d2e11d700

Transpuesta de matrices

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|--|--|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | T | 1 | 2 | 7 | 5 | 1 | 3 | 7 | | |
| 2 | 4 | 5 | 6 | 2 | 1 | 4 | | 2 | 4 | 4 | 1 | 3 | 4 | 6 | | |
| 7 | 4 | 2 | 6 | 3 | 1 | 3 | | 3 | 5 | 2 | 2 | 2 | 2 | 5 | | |
| 5 | 1 | 2 | 6 | 3 | 2 | 4 | = | 4 | 6 | 6 | 6 | 5 | 3 | 4 | | |
| 1 | 3 | 2 | 5 | 4 | 7 | 6 | | 5 | 2 | 3 | 3 | 4 | 1 | 3 | | |
| 3 | 4 | 2 | 3 | 1 | 5 | 2 | | 6 | 1 | 1 | 2 | 7 | 5 | 2 | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | | 7 | 4 | 3 | 4 | 6 | 2 | 1 | | |
| | | | | | | | | | | | | | | | | |
| | 7 | 4 | 2 | 6 | 3 | 1 | T | 7 | 5 | 1 | 7 | 3 | 1 | 2 | | |
| | 5 | 1 | 2 | 6 | 3 | 2 | | 4 | 1 | 2 | 6 | 4 | 3 | 4 | | |
| | 1 | 2 | 3 | 4 | 5 | 6 | | 2 | 2 | 3 | 5 | 2 | 2 | 5 | | |
| | 7 | 6 | 5 | 4 | 3 | 2 | = | 6 | 6 | 4 | 4 | 3 | 5 | 6 | | |
| | 3 | 4 | 2 | 3 | 1 | 5 | | 3 | 3 | 5 | 3 | 1 | 4 | 2 | | |
| | 1 | 3 | 2 | 5 | 4 | 7 | | 1 | 2 | 6 | 2 | 5 | 7 | 1 | | |
| | 2 | 4 | 5 | 6 | 2 | 1 | | 3 | 4 | 7 | 1 | 2 | 6 | 4 | | |

Fin Hilo 5

cesar@cesar-HP-Notebook:~/Documentos/S0/Practica4\$

CAPTURAS DE LOS .TXT

The screenshots show the following outputs:

- Hilo1.txt:** "Soy el primer Hilo: 7f8d2e11d700", "Suma de matrices". It displays two 3x6 matrices being added to produce a 3x6 result matrix.
- Hilo2.txt:** "Soy el segundo Hilo: 7f8d2e11d700", "Resta de matrices". It displays two 3x6 matrices being subtracted to produce a 3x6 result matrix.
- Hilo3.txt:** "Soy el tercer Hilo: 7f8d2e11d700", "Multiplicación de matrices". It displays two 3x6 matrices being multiplied to produce a 3x6 result matrix.
- Hilo4.txt:** "Soy el cuarto Hilo: 7f8d2e11d700", "Transpuesta de matrices". It displays a 3x6 matrix being transposed into a 6x3 result matrix.

Código Linux en forma Concurrente

La cuestión para que quede concurrente es que los `pthread_create` queden primero y los `pthread_join` queden hasta el final, excepto por el hilo 5 que necesita que los 4 hilos anteriores hayan finalizado para poder imprimir el contenido de los .txt de salida.

Todo el código restante es exactamente el mismo, por lo que en la ejecución el orden en el que aparecerán los hilos será diferente cada vez que se corra el programa, solo del hilo 1 al 4, el hilo 5 siempre aparecerá al final.

```
38 int main(){
39
40     pthread_t th1,th2,th3,th4,th5; //Direcciones de los 5 hilos(threads)
41
42     //Creacion de los hilos
43     pthread_create(&th1,NULL,(void*)hilo1,NULL); //Hilo de la Suma
44     pthread_create(&th2,NULL,(void*)hilo2,NULL); //Hilo de la Resta
45     pthread_create(&th3,NULL,(void*)hilo3,NULL); //Hilo de la Multiplicación
46     pthread_create(&th4,NULL,(void*)hilo4,NULL); //Hilo de la Transpuesta
47
48     //Se esperan la finalización de los hilos
49     pthread_join(th1,NULL); //Esperar a hilo1
50     pthread_join(th2,NULL); //Esperar a hilo2
51     pthread_join(th3,NULL); //Esperar a hilo3
52     pthread_join(th4,NULL); //Esperar a hilo4
53
54     //Se crea el hilo 5 y se espera su finalización
55     pthread_create(&th5,NULL,(void*)hilo5,NULL); //Hilo de los .txt
56     pthread_join(th5,NULL); //Esperar a hilo5
57 }
```


VERSIÓN WINDOWS

CÓDIGO En forma secuencial

```
punto6.c x
1 //Aplicacion que crea 5 hilos
2 //Hilo 1: Suma de dos matrices de 7x7 elementos de tipo entero
3 //Hilo 2: Resta sobre esas mismas matrices
4 //Hilo 3: Multiplicación de matrices
5 //Hilo 4: Obtener la transpuesta de cada matriz
6 //Hilo 5: Leera lo archivos de resultados y los mostrara en pantalla
7 //Los hilos del 1 -> 4 deberan crear un .txt con las salidas
8 //-----
9 #include <windows.h>
10 #include <stdio.h>
11 #include <stdlib.h>
12
13 DWORD WINAPI hilo1();
14 DWORD WINAPI hilo2();
15 DWORD WINAPI hilo3();
16 DWORD WINAPI hilo4();
17 DWORD WINAPI hilo5();
18
19 int m1[7][7] = {
20     {1,2,3,4,5,6,7},
21     {2,4,5,6,2,1,4},
22     {7,4,2,6,3,1,3},
23     {5,1,2,6,3,2,4},
24     {1,3,2,5,4,7,6},
25     {3,4,2,3,1,5,2},
26     {7,6,5,4,3,2,1}
27 };
28 int m2[7][7] = {
29     {7,4,2,6,3,1,3},
30     {5,1,2,6,3,2,4},
31     {1,2,3,4,5,6,7},
32     {7,6,5,4,3,2,1},
33     {3,4,2,3,1,5,2},
34     {1,3,2,5,4,7,6},
35     {2,4,5,6,2,1,4}
36 };
37 int mr[7][7];
38
39 int main(){
40     DWORD idHilo1;
41     HANDLE manHilo1;
42
43     DWORD idHilo2;
44     HANDLE manHilo2;
45
46     DWORD idHilo3;
47     HANDLE manHilo3;
48
49     DWORD idHilo4;
50     HANDLE manHilo4;
51
52     DWORD idHilo5;
53     HANDLE manHilo5;
54
55     //Creacion de hilo 1
56     manHilo1=CreateThread(NULL,0,hilo1,NULL,0,&idHilo1);
57     //Esperar la finanilación del hilo 1
58     WaitForSingleObject(manHilo1,INFINITE);
59     //Cierre del manejador del hilo 1 creado
60     CloseHandle(manHilo1);
61
62     //Creacion de hilo 2
```

```

63     manHilo2=CreateThread(NULL,0,hilo2,NULL,0,&idHilo2);
64     //Esperar la finalización del hilo 2
65     WaitForSingleObject(manHilo2,INFINITE);
66     //Cierre del manejador del hilo 2 creado
67     CloseHandle(manHilo2);
68
69     //Creacion de hilo 3
70     manHilo3=CreateThread(NULL,0,hilo3,NULL,0,&idHilo3);
71     //Esperar la finalización del hilo 3
72     WaitForSingleObject(manHilo3,INFINITE);
73     //Cierre del manejador del hilo 3 creado
74     CloseHandle(manHilo3);
75
76     //Creacion de hilo 4
77     manHilo4=CreateThread(NULL,0,hilo4,NULL,0,&idHilo4);
78     //Esperar la finalización del hilo 4
79     WaitForSingleObject(manHilo4,INFINITE);
80     //Cierre del manejador del hilo 4 creado
81     CloseHandle(manHilo4);
82
83     //Creacion de hilo 5
84     manHilo5=CreateThread(NULL,0,hilo5,NULL,0,&idHilo5);
85     //Esperar la finalización del hilo 5
86     WaitForSingleObject(manHilo5,INFINITE);
87     //Cierre del manejador del hilo 5 creado
88     CloseHandle(manHilo5);
89
90     return 0;
91 }
92 DWORD WINAPI hilo1(){
93     printf("Inicio Hilo 1\n");
94
95     FILE *th1 = NULL;
96     th1 = fopen("Hilo1.txt","w+");
97     if(th1 == NULL){
98         printf("No fue posible abrir el archivo\n");
99     }
100    //Se imprime el Id. del Hilo
101    fprintf(th1,"Soy el primer Hilo: %lx\n",pthread_self());
102    fprintf(th1,"Suma de matrices\n");
103
104    for(int i = 0; i < 7; i++){
105        //matriz 1
106        for(int j = 0; j < 7; j++){
107            if(j == 0){
108                fprintf(th1,"|   ");
109            }
110            fprintf(th1,"%d ",m1[i][j]);
111            if(j == 6){
112                fprintf(th1," |");
113            }
114        }
115        if (i == 3){fprintf(th1," + ");}else{fprintf(th1,"   ");}
116        //matriz 2
117        for(int j = 0; j < 7; j++){
118            if(j == 0){
119                fprintf(th1,"|   ");
120            }
121            fprintf(th1,"%d ",m2[i][j]);
122            if(j == 6){
123                if(i == 3){
124                    fprintf(th1," |   =\n");
125                }
126                else{

```

```

127         fprintf(th1," |\n");
128     }
129 }
130 }
131 }
132 fprintf(th1,"\n");
133 for(int i = 0; i < 7; i++){
134     for(int j = 0; j < 7; j++){
135         if(j == 0){
136             fprintf(th1,"| ");
137         }
138         mr[i][j] = m1[i][j] + m2[i][j];
139         fprintf(th1,"%d ",mr[i][j]);
140     }
141     fprintf(th1,"|\n");
142 }
143 fclose(th1);//Se cierra el archivo.txt
144
145 printf("Archivo creado\nFin Hilo 1\n");
146 printf("\n");
147
148 return 0;
149 }
150
151 DWORD WINAPI hilo2(){
152     printf("Inicio Hilo 2\n");
153
154     FILE *th2 = NULL;
155     th2 = fopen("Hilo2.txt","w+");
156     if(th2 == NULL){
157         printf("No fue posible abrir el archivo\n");
158     }
159     //Se imprime el Id. del Hilo
160     fprintf(th2,"Soy el segundo Hilo: %lx\n",pthread_self());
161     fprintf(th2,"Resta de matrices\n");
162
163     for(int i = 0; i < 7; i++){
164         //matriz 1
165         for(int j = 0; j < 7; j++){
166             if(j == 0){
167                 fprintf(th2,"| ");
168             }
169             fprintf(th2,"%d ",m1[i][j]);
170             if(j == 6){
171                 fprintf(th2," |");
172             }
173         }
174         if (i == 3){fprintf(th2," - ");}else{fprintf(th2," ");}
175         //matriz 2
176         for(int j = 0; j < 7; j++){
177             if(j == 0){
178                 fprintf(th2,"| ");
179             }
180             fprintf(th2,"%d ",m2[i][j]);
181             if(j == 6){
182                 if(i == 3){
183                     fprintf(th2," | =\n");
184                 }
185                 else{
186                     fprintf(th2," |\n");
187                 }
188             }
189         }
190     }

```



```

191     fprintf(th2, "\n");
192     for(int i = 0; i < 7; i++){
193         for(int j = 0; j < 7; j++){
194             if(j == 0){
195                 fprintf(th2, "|  ");
196             }
197             mr[i][j] = m1[i][j] - m2[i][j];
198             fprintf(th2, "%d ", mr[i][j]);
199         }
200         fprintf(th2, "|\n");
201     }
202     fclose(th2); //Se cierra el archivo.txt
203
204     printf("Archivo creado\nFin Hilo 2\n");
205     printf("\n");
206     return 0;
207 }
208
209 DWORD WINAPI hilo3(){
210     printf("Inicio Hilo 3\n");
211
212     FILE *th3 = NULL;
213     th3 = fopen("Hilo3.txt", "w+");
214     if(th3 == NULL){
215         printf("No fue posible abrir el archivo\n");
216     }
217     //Se imprime el Id. del Hilo
218     fprintf(th3, "Soy el tercer Hilo: %lx\n", pthread_self());
219     fprintf(th3, "Multiplicación de matrices\n");
220
221     for(int i = 0; i < 7; i++){
222         //matriz 1
223         for(int j = 0; j < 7; j++){
224             if(j == 0){
225                 fprintf(th3, "|  ");
226             }
227             fprintf(th3, "%d ", m1[i][j]);
228             if(j == 6){
229                 fprintf(th3, " |");
230             }
231         }
232         if (i == 3){fprintf(th3, " * ");} else {fprintf(th3, " ");}
233         //matriz 2
234         for(int j = 0; j < 7; j++){
235             if(j == 0){
236                 fprintf(th3, "|  ");
237             }
238             fprintf(th3, "%d ", m2[i][j]);
239             if(j == 6){
240                 if(i == 3){
241                     fprintf(th3, " |  =\n");
242                 }
243                 else{
244                     fprintf(th3, " |\n");
245                 }
246             }
247         }
248     }
249     fprintf(th3, "\n");
250     for(int a = 0; a < 7; a++){
251
252         for(int i = 0; i < 7; i++){
253             int suma = 0;

```

```

254         for(int j = 0; j < 7; j++){
255             suma +=(m1[i][j] * m2[j][a]);
256         }
257         mr[i][a] = suma;
258     }
259 }
260 for(int i = 0; i < 7; i++){
261     for(int j = 0; j < 7; j++){
262         if(j == 0){
263             fprintf(th3,"|  ");
264         }
265         fprintf(th3,"%d ",mr[i][j]);
266     }
267     fprintf(th3,"|\n");
268 }
269 fclose(th3);//Se cierra el archivo.txt
270
271 printf("Archivo creado\nFin Hilo 3\n");
272 printf("\n");
273 return 0;
274 }
275 DWORD WINAPI hilo4(){
276     printf("Inicio Hilo 4\n");
277
278     FILE *th4 = NULL;
279     th4 = fopen("Hilo4.txt","w+");
280     if(th4 == NULL){
281         printf("No fue posible abrir el archivo\n");
282     }
283     //Se imprime el Id. del Hilo
284     fprintf(th4,"Soy el cuarto Hilo: %lx\n",pthread_self());
285     fprintf(th4,"Transpuesta de matrices\n");
286
287     //Transpuesta de la matriz 1
288     for(int i = 0; i < 7; i++){
289         for(int j = 0; j < 7; j++){
290             if(j == 0){
291                 fprintf(th4,"|  ");
292             }
293             fprintf(th4,"%d ",m1[i][j]);
294             if(j == 6){
295                 fprintf(th4," |");
296             }
297         }
298         if (i == 0){fprintf(th4,"T  ");}
299         else if (i == 3){fprintf(th4," = ");}
300         else{fprintf(th4,"  ");}
301         //matriz 2
302         for(int j = 0; j < 7; j++){
303             if(j == 0){
304                 fprintf(th4,"|  ");
305             }
306             fprintf(th4,"%d ",m1[j][i]);
307             if(j == 6){
308                 fprintf(th4," |\n");
309             }
310         }
311     }
312     fprintf(th4,"\n");
313     //Transpuesta de la matriz 2
314     for(int i = 0; i < 7; i++){
315         for(int j = 0; j < 7; j++){
316             if(j == 0){
317                 fprintf(th4,"|  ");
318             }
319             fprintf(th4,"%d ",m2[i][j]);
320             if(j == 6){

```

```

321         fprintf(th4, " |");
322     }
323 }
324 if (i == 0){fprintf(th4, "T ");}
325 else if (i == 3){fprintf(th4, " = ");}
326 else{fprintf(th4, " ");}
327 //matriz 2
328 for(int j = 0; j < 7; j++){
329     if(j == 0){
330         fprintf(th4, "| ");
331     }
332     fprintf(th4, "%d ", m2[j][i]);
333     if(j == 6){
334         fprintf(th4, " |\n");
335     }
336 }
337 }
338 fprintf(th4, "\n");
339
340 fclose(th4); //Se cierra el archivo.txt
341
342 printf("Archivo creado\nFin Hilo 4\n");
343 printf("\n");
344 return 0;
345 }
346 DWORD WINAPI hilo5(){
347     printf("Inicio Hilo 5\n");
348     //Se imprime el Id. del Hilo
349     printf("Soy el quinto Hilo: %lx\n", pthread_self());
350     printf("Lectura de los archivos creados de los hilos anteriores\n");
351
352     char c;
353     FILE *archivo1 = fopen("Hilo1.txt", "r");
354     if(archivo1 != NULL){
355         printf("\n***Hilo1.txt***\n");
356         while(1){
357             c = fgetc(archivo1);
358             if(feof(archivo1)){
359                 break;
360             }
361             printf("%c", c);
362         }
363     }else{
364         printf("Hubo un error al abrir el archivo o es inexistente.");
365     }
366     fclose(archivo1);
367
368     FILE *archivo2 = fopen("Hilo2.txt", "r");
369     if(archivo2 != NULL){
370         printf("\n***Hilo2.txt***\n");
371         while(1){
372             c = fgetc(archivo2);
373             if(feof(archivo2)){
374                 break;
375             }
376             printf("%c", c);
377         }
378     }else{
379         printf("Hubo un error al abrir el archivo o es inexistente.");
380     }
381     fclose(archivo2);
382
383     FILE *archivo3 = fopen("Hilo3.txt", "r");

```



```

384     if(archivo3!=NULL){
385         printf("\n***Hilo3.txt***\n");
386         while(1){
387             c=fgetc(archivo3);
388             if(feof(archivo3)){
389                 break;
390             }
391             printf("%c",c);
392         }
393     }else{
394         printf("Hubo un error al abrir el archivo o es inexistente.");
395     }
396     fclose(archivo3);
397
398     FILE *archivo4 = fopen("Hilo4.txt","r");
399     if(archivo4!=NULL){
400         printf("\n***Hilo4.txt***\n");
401         while(1){
402             c=fgetc(archivo4);
403             if(feof(archivo4)){
404                 break;
405             }
406             printf("%c",c);
407         }
408     }else{
409         printf("Hubo un error al abrir el archivo o es inexistente.");
410     }
411     fclose(archivo4);
412
413     printf("Fin Hilo 5\n");
414     return 0;
415 }

```

COMPILACIÓN Y EJECUCIÓN

```

C:\Users\cesar\OneDrive\Documentos\Cuarto Semestre\SISTEMAS OPERATIVOS\Primer Parcial\Practica4\Practica4_Windows\punto6.exe
Inicio Hilo 1
Archivo creado
Fin Hilo 1

Inicio Hilo 2
Archivo creado
Fin Hilo 2

Inicio Hilo 3
Archivo creado
Fin Hilo 3

Inicio Hilo 4
Archivo creado
Fin Hilo 4

Inicio Hilo 5
Soy el quinto Hilo: 5
Lectura de los archivos creados de los hilos anteriores

***Hilo1.txt***
Soy el primer Hilo: 1
Suma de matrices
| 1 2 3 4 5 6 7 | | 7 4 2 6 3 1 3 |
| 2 4 5 6 2 1 4 | | 5 1 2 6 3 2 4 |
| 7 4 2 6 3 1 3 | | 1 2 3 4 5 6 7 |
| 5 1 2 6 3 2 4 | + | 7 6 5 4 3 2 1 | =
| 1 3 2 5 4 7 6 | | 3 4 2 3 1 5 2 |
| 3 4 2 3 1 5 2 | | 1 3 2 5 4 7 6 |
| 7 6 5 4 3 2 1 | | 2 4 5 6 2 1 4 |

| 8 6 5 10 8 7 10 |
| 7 5 7 12 5 3 8 |
| 8 6 5 10 8 7 10 |
| 12 7 7 10 6 4 5 |
| 4 7 4 8 5 12 8 |
| 4 7 4 8 5 12 8 |
| 9 10 10 10 5 3 5 |

```

```

***Hilo2.txt***
Soy el segundo Hilo: 2
Resta de matrices
1 2 3 4 5 6 7 | | 7 4 2 6 3 1 3 |
2 4 5 6 2 1 4 | | 5 1 2 6 3 2 4 |
7 4 2 6 3 1 3 | | 1 2 3 4 5 6 7 |
5 1 2 6 3 2 4 | - | 7 6 5 4 3 2 1 | =
1 3 2 5 4 7 6 | | 3 4 2 3 1 5 2 |
3 4 2 3 1 5 2 | | 1 3 2 5 4 7 6 |
7 6 5 4 3 2 1 | | 2 4 5 6 2 1 4 |

-6 -2 1 -2 2 5 4 |
-3 3 3 0 -1 -1 0 |
6 2 -1 2 -2 -5 -4 |
-2 -5 -3 2 0 0 3 |
-2 -1 0 2 3 2 4 |
2 1 0 -2 -3 -2 -4 |
5 2 0 -2 1 1 -3 |

***Hilo3.txt***
Soy el tercer Hilo: 3
Multiplicaci|n de matrices
1 2 3 4 5 6 7 | | 7 4 2 6 3 1 3 |
2 4 5 6 2 1 4 | | 5 1 2 6 3 2 4 |
7 4 2 6 3 1 3 | | 1 2 3 4 5 6 7 |
5 1 2 6 3 2 4 | * | 7 6 5 4 3 2 1 | =
1 3 2 5 4 7 6 | | 3 4 2 3 1 5 2 |
3 4 2 3 1 5 2 | | 1 3 2 5 4 7 6 |
7 6 5 4 3 2 1 | | 2 4 5 6 2 1 4 |

83 102 92 133 79 105 110 |
96 85 83 115 75 73 89 |
129 99 81 130 74 64 81 |
103 95 78 111 65 64 73 |
90 102 91 135 81 104 108 |
76 65 57 102 65 71 82 |
125 90 76 139 89 87 106 |

***Hilo4.txt***
Soy el cuarto Hilo: 4
Transpuesta de matrices
1 2 3 4 5 6 7 | T | 1 2 7 5 1 3 7 |
2 4 5 6 2 1 4 | | 2 4 4 1 3 4 6 |
7 4 2 6 3 1 3 | | 3 5 2 2 2 2 5 |
5 1 2 6 3 2 4 | = | 4 6 6 6 5 3 4 |
1 3 2 5 4 7 6 | | 5 2 3 3 4 1 3 |
3 4 2 3 1 5 2 | | 6 1 1 2 7 5 2 |
7 6 5 4 3 2 1 | | 7 4 3 4 6 2 1 |

7 4 2 6 3 1 3 | T | 7 5 1 7 3 1 2 |
5 1 2 6 3 2 4 | | 4 1 2 6 4 3 4 |
1 2 3 4 5 6 7 | | 2 2 3 5 2 2 5 |
7 6 5 4 3 2 1 | = | 6 6 4 4 3 5 6 |
3 4 2 3 1 5 2 | | 3 3 5 3 1 4 2 |
1 3 2 5 4 7 6 | | 1 2 6 2 5 7 1 |
2 4 5 6 2 1 4 | | 3 4 7 1 2 6 4 |

Fin Hilo 5
-----
Process exited after 0.1525 seconds with return value 0
Presione una tecla para continuar . . .

```

CAPTURAS DE LOS .TXT

```

Hilo1.txt Hilo2.txt Hilo3.txt Hilo4.txt
1 Soy el primer Hilo: 1
2 Suma de matrices
3 | 1 2 3 4 5 6 7 | | 7 4 2 6 3 1 3 |
4 | 2 4 5 6 2 1 4 | | 5 1 2 6 3 2 4 |
5 | 7 4 2 6 3 1 3 | | 1 2 3 4 5 6 7 |
6 | 5 1 2 6 3 2 4 | + | 7 6 5 4 3 2 1 | =
7 | 1 3 2 5 4 7 6 | | 3 4 2 3 1 5 2 |
8 | 3 4 2 3 1 5 2 | | 1 3 2 5 4 7 6 |
9 | 7 6 5 4 3 2 1 | | 2 4 5 6 2 1 4 |
10
11 | 8 6 5 10 8 7 10 |
12 | 7 5 7 12 5 3 8 |
13 | 8 6 5 10 8 7 10 |
14 | 12 7 7 10 6 4 5 |
15 | 4 7 4 8 5 12 8 |
16 | 4 7 4 8 5 12 8 |
17 | 9 10 10 10 5 3 5 |

```

```

Hilo1.txt Hilo2.txt Hilo3.txt Hilo4.txt
1 Soy el segundo Hilo: 2
2 Resta de matrices
3 | 1 2 3 4 5 6 7 | | 7 4 2 6 3 1 3 |
4 | 2 4 5 6 2 1 4 | | 5 1 2 6 3 2 4 |
5 | 7 4 2 6 3 1 3 | | 1 2 3 4 5 6 7 |
6 | 5 1 2 6 3 2 4 | - | 7 6 5 4 3 2 1 | =
7 | 1 3 2 5 4 7 6 | | 3 4 2 3 1 5 2 |
8 | 3 4 2 3 1 5 2 | | 1 3 2 5 4 7 6 |
9 | 7 6 5 4 3 2 1 | | 2 4 5 6 2 1 4 |
10
11 | -6 -2 1 -2 2 5 4 |
12 | -3 3 3 0 -1 -1 0 |
13 | 6 2 -1 2 -2 -5 -4 |
14 | -2 -5 -3 2 0 0 3 |
15 | -2 -1 0 2 3 2 4 |
16 | 2 1 0 -2 -3 -2 -4 |
17 | 5 2 0 -2 1 1 -3 |

```

```
Hilo1.txt Hilo2.txt Hilo3.txt Hilo4.txt
1 Soy el tercer Hilo: 3
2 Multiplicación de matrices
3 | 1 2 3 4 5 6 7 | | 7 4 2 6 3 1 3 |
4 | 2 4 5 6 2 1 4 | | 5 1 2 6 3 2 4 |
5 | 7 4 2 6 3 1 3 | | 1 2 3 4 5 6 7 |
6 | 5 1 2 6 3 2 4 | * | 7 6 5 4 3 2 1 | =
7 | 1 3 2 5 4 7 6 | | 3 4 2 3 1 5 2 |
8 | 3 4 2 3 1 5 2 | | 1 3 2 5 4 7 6 |
9 | 7 6 5 4 3 2 1 | | 2 4 5 6 2 1 4 |
10
11 | 83 102 92 133 79 105 110 |
12 | 96 85 83 115 75 73 89 |
13 | 129 99 81 130 74 64 81 |
14 | 103 95 78 111 65 64 73 |
15 | 90 102 91 135 81 104 108 |
16 | 76 65 57 102 65 71 82 |
17 | 125 90 76 139 89 87 106 |
```

```
Hilo1.txt Hilo2.txt Hilo3.txt Hilo4.txt
1 Soy el cuarto Hilo: 4
2 Transpuesta de matrices
3 | 1 2 3 4 5 6 7 | T | 1 2 7 5 1 3 7 |
4 | 2 4 5 6 2 1 4 | | 2 4 4 1 3 4 6 |
5 | 7 4 2 6 3 1 3 | | 3 5 2 2 2 2 5 |
6 | 5 1 2 6 3 2 4 | = | 4 6 6 6 5 3 4 |
7 | 1 3 2 5 4 7 6 | | 5 2 3 3 4 1 3 |
8 | 3 4 2 3 1 5 2 | | 6 1 1 2 7 5 2 |
9 | 7 6 5 4 3 2 1 | | 7 4 3 4 6 2 1 |
10
11 | 7 4 2 6 3 1 3 | T | 7 5 1 7 3 1 2 |
12 | 5 1 2 6 3 2 4 | | 4 1 2 6 4 3 4 |
13 | 1 2 3 4 5 6 7 | | 2 2 3 5 2 2 5 |
14 | 7 6 5 4 3 2 1 | = | 6 6 4 4 3 5 6 |
15 | 3 4 2 3 1 5 2 | | 3 3 5 3 1 4 2 |
16 | 1 3 2 5 4 7 6 | | 1 2 6 2 5 7 1 |
17 | 2 4 5 6 2 1 4 | | 3 4 7 1 2 6 4 |
18
```

Codigo Linux en forma Concurrente

La cuestión para que quede concurrente es que los CreateThread queden primero y los WaitForSingleObject queden hasta el final, excepto por el hilo 5 que necesita que los 4 hilos anteriores hayan finalizado para poder imprimir el contenido de los .txt de salida .

Todo el código restante es exactamente el mismo, por lo que en la ejecución el orden en el que aparecerán los hilos será diferente cada vez que se corra el programa, solo del hilo 1 al 4, el hilo 5 siempre aparecerá al final.

```
39 int main(){
40     DWORD idHilo1;
41     HANDLE manHilo1;
42
43     DWORD idHilo2;
44     HANDLE manHilo2;
45
46     DWORD idHilo3;
47     HANDLE manHilo3;
48
49     DWORD idHilo4;
50     HANDLE manHilo4;
51
52     DWORD idHilo5;
53     HANDLE manHilo5;
54
55     //Creacion de hilo 1
56     manHilo1=CreateThread(NULL,0,hilo1,NULL,0,&idHilo1);
57     //Creacion de hilo 2
58     manHilo2=CreateThread(NULL,0,hilo2,NULL,0,&idHilo2);
59     //Creacion de hilo 3
60     manHilo3=CreateThread(NULL,0,hilo3,NULL,0,&idHilo3);
61     //Creacion de hilo 4
62     manHilo4=CreateThread(NULL,0,hilo4,NULL,0,&idHilo4);
63
64     //Esperar la finalización del hilo 1
65     WaitForSingleObject(manHilo1,INFINITE);
66     //Esperar la finalización del hilo 2
67     WaitForSingleObject(manHilo2,INFINITE);
68     //Esperar la finalización del hilo 3
69     WaitForSingleObject(manHilo3,INFINITE);
70     //Esperar la finalización del hilo 4
71     WaitForSingleObject(manHilo4,INFINITE);
72
73     //Cierre del manejador del hilo 1 creado
74     CloseHandle(manHilo1);
75     //Cierre del manejador del hilo 2 creado
76     CloseHandle(manHilo2);
77     //Cierre del manejador del hilo 3 creado
78     CloseHandle(manHilo3);
79     //Cierre del manejador del hilo 4 creado
80     CloseHandle(manHilo4);
```

```

81
82 //Creacion de hilo 5
83 manHilo5=CreateThread(NULL,0,hilo5,NULL,0,&idHilo5);
84 //Esperar la finalización del hilo 5
85 WaitForSingleObject(manHilo5,INFINITE);
86 //Cierre del manejador del hilo 5 creado
87 CloseHandle(manHilo5);
88
89 return 0;
90 }

```

OBSERVACIONES

En comparación con el punto 5 de la práctica 3 en la cual el programa tiene que hacer lo mismo que el de esta práctica pero con procesos, el hacer las mismas cosas pero con hilos se crean más rápido y supongo más eficientes los archivos con las salidas .txt, también se ocupan menos recursos pues como se sabe los hilos requieren menos para poder funcionar.

La creación de hilos en Linux es un poco menos confusa que en windows pues uno puede crear y manejar hilos de la forma que desee, pero sin perderse en el código pues cada instrucción se puede entender de forma casi inmediata. Sin embargo en Windows si te puedes llegar a confundir si no sabes como trabajar con los hilos de forma adecuada.

Al final lo único que cambia en ambos sistemas es su creación y manipulación pues lo demás que queramos que hagan los hilos sigue siendo el mismo código para ambos.

III.CONCLUSIÓN

En esta práctica trabajamos y analizamos el tópico que revisa a los hilos. Podemos concluir que es uno de los temas más importantes, en la materia de Sistemas Operativos, y en general de las ciencias de la computación.

Dentro de lo que podemos aprender cómo es que el sistema operativo se encuentra estructurado en una primera etapa, el desarrollo de procesos y la comprensión de los hilos de ejecución a través de los cuales, el sistema delega funciones y opera en forma multi funcional. También podemos destacar la forma en la que el sistema gestiona las interrupciones, este se encarga de controlar los accesos al procesador, verificar el estatus de un proceso y determinar su ejecución de acuerdo al nivel de importancia, cabe destacar que no todas las interrupciones son controladas por el SO, ya que existen interrupciones enmascaradas y que son exclusivas del hardware de nuestro ordenador.

IV.REFERENCIAS

1. <https://www.fing.edu.uy/tecnoinf/mvd/cursos/so/material/teo/so05-hilos.pdf>
2. <http://aisii.azc.uam.mx/areyes/archivos/licenciatura/sd/U2/ConceptoHilos.pdf>
3. <https://sites.google.com/site/carlosraulsan2987/home/sistemas-operativos/unidad-2-y-3/procesos-e-hilos>
4. <https://webprogramacion.com/1/sistemas-operativos/procesos-e-hilos.aspx>