



INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO



SISTEMAS OPERATIVOS

PROFESOR: CORTES GALICIA JORGE

CICLO ESCOLAR: 2024-2

PRACTICA 6 – ADMINISTRADOR DE PROCESOS EN LINUX Y
WINDOWS (2)

20 DE MAYO DE 2024

INTRODUCCION

En esta práctica de sistemas operativos, se profundiza en la gestión de hilos y la manipulación de archivos y directorios, aspectos fundamentales para la programación concurrente y el manejo eficiente de recursos en un sistema. La capacidad de crear, controlar y sincronizar hilos permite a las aplicaciones ejecutar múltiples tareas simultáneamente, mejorando su rendimiento y capacidad de respuesta.

Las funciones de la biblioteca pthread en Linux, tales como **pthread_create**, **pthread_join**, **pthread_self** y **pthread_exit**, proporcionan un marco robusto para el manejo de hilos. **pthread_create** permite iniciar nuevos hilos dentro del mismo proceso, mientras que **pthread_join** facilita la sincronización, asegurando que un hilo principal pueda esperar la finalización de otros hilos. **pthread_self** devuelve el identificador del hilo actual, y **pthread_exit** permite la terminación controlada de un hilo, devolviendo un valor que puede ser capturado por otros hilos. Estas funciones son esenciales para construir aplicaciones multihilo eficientes.

Además, se exploran funciones del sistema para la manipulación de archivos y directorios, como **scandir** y **stat**. **scandir** permite escanear un directorio y crear una lista de sus entradas, aplicando filtros y ordenaciones según sea necesario, lo que resulta útil para aplicaciones que necesitan interactuar con el sistema de archivos de manera dinámica y organizada. Por otro lado, **stat** proporciona información detallada sobre archivos, incluyendo su tamaño, permisos y fechas de modificación, permitiendo a los programas gestionar y verificar el estado de los archivos de manera precisa.

En el contexto de Windows, se utiliza la función **CreateThread** para la creación de hilos. Aunque su uso y algunos aspectos de implementación difieren de las funciones pthread en Linux, **CreateThread** cumple un rol similar en la gestión de la concurrencia dentro de un proceso, facilitando el desarrollo de aplicaciones multihilo en esta plataforma.

Un aspecto importante de la práctica es la comparación de tiempos de ejecución entre programas que utilizan procesos y aquellos que utilizan hilos. Los hilos, al compartir el mismo espacio de direcciones, suelen ser más ligeros y rápidos de crear y destruir en comparación con los procesos, que requieren mayor overhead para mantener espacios de direcciones separados. Esta comparación es crucial para entender las ventajas y limitaciones de cada enfoque y elegir el más adecuado para optimizar el rendimiento de las aplicaciones.

DESARROLLO

1.- A través de la ayuda en línea que proporciona Linux, investigue el funcionamiento de las funciones: **pthread_create()**, **pthread_join()**, **pthread_self()**, **pthread_exit()**, **scandir()**, **stat()**. Explique los argumentos y retorno de cada función.

Pthread create:

Crea un nuevo hilo en el proceso actual, el nuevo hilo empieza su ejecución invocando **start_routine()**; **arg** se pasa como el único argumento de **start_routine()**.

```
1 #include <pthread.h>
2
3 int pthread_create(pthread_t *thread, const pthread_attr_t *attr, void *(*start_routine) (void *), void *arg);
```

- **thread**: Es un puntero a una variable de tipo **pthread_t** donde se almacenará el identificador del hilo creado.
- **attr**: Es un puntero a una estructura de atributos que especifica los atributos del hilo que se está creando. Puede ser **NULL** para utilizar los atributos por defecto.
- **start_routine**: Es una función que será ejecutada por el hilo. Debe ser de tipo **void *función(void *)** y tomar un puntero **void** como argumento y devolver un puntero **void**.
- **arg**: Es un puntero que se pasará como argumento a la función **start_routine**.

El nuevo hilo termina su ejecución de las siguientes maneras:

- Llama **pthread_exit(3)**, especificando un valor de estado de salida que en el mismo proceso que llama a **pthread_join(3)**.
- Se retorna de **start_routine()**. Este es el equivalente a llamar a **pthread_exit(3)** con el valor declarado en el sentencia **return**.
- Se cancela llamando **pthread_cancel(3)**.
- Cualquiera de los hilos llama **exit(3)**, o el hilo principal realiza un retorno desde la función **main()**. Esto causa la terminación de todos los hilos en el proceso.

La función retorna **0** cuando se ejecuta exitosamente, cuando llega a un error devuelve un valor de error y el contenido de ***thread** es indefinido.

Pthread join:

Espera a que el hilo especificado por **thread** termine de ejecutarse. Si ese hilo ya terminó su ejecución, entonces la función retorna un valor de manera inmediata.

- **thread**: Es el identificador del hilo que se desea esperar.
- **retval**: Es un puntero a un puntero donde se almacenará el valor de retorno del hilo. Puede ser **NULL** si no se necesita el valor de retorno.

```
1 #include <pthread.h>
2
3 int pthread_join(pthread_t thread, void **retval);
```

La función bloqueará la ejecución del hilo que la llama hasta que el hilo especificado por **thread** haya terminado su ejecución. Devuelve **0** si tiene éxito y un valor distinto de **0** en caso de error.

Si **retval** no es **NULL**, entonces la función copia el estado de salida del hilo objetivo (el valor que el hilo objetivo le envió a **pthread_exit(3)**) a la localidad de memoria a la que apunta **retval**.

Si varios hilos hacen una llamada a **pthread_join** con el mismo hilo, los resultados son indefinidos; si el hilo que llama a la función es cancelado, entonces al hilo objetivo se le podrá hacer join.

Pthread self:

Esta función devuelve el ID del hilo que la llama.

```
1 #include <pthread.h>
2
3 pthread_t pthread_self(void);
```

Esta función siempre se ejecuta de manera exitosa, devolviendo el identificador único del hilo.

Pthread exit:

Se utiliza para terminar la ejecución del hilo que la llama y devuelve un valor por medio de **retval**, ese valor podrá ser accedido por otro hilo que esté utilizando la función **pthread_join()** en el hilo terminado.

```
1 #include <pthread.h>
2
3 void pthread_exit(void *retval);
```

- **retval:** Es un puntero que especifica el valor de salida del hilo. Puede ser **NULL** si no se necesita devolver ningún valor.

Esta función siempre se ejecuta de manera exitosa y nunca devuelve nada al hilo que la llama.

Scandir:

Se utiliza para escanear un directorio y crear una lista de las entradas (archivos y subdirectorios) que contiene ese directorio, filtrando opcionalmente las entradas mediante una función de selección proporcionada por el usuario.

```
1 #include <dirent.h>
2
3 int scandir(const char *dir, struct dirent ***namelist, int (*filter)(const struct dirent *), int (*compar)(const struct dirent **, const struct dirent **));
```

- **dir:** Es una cadena de caracteres que especifica la ruta del directorio que se va a escanear.
- **namelist:** Es un puntero a un puntero a una matriz de estructuras **dirent**. Después de que **scandir** tenga éxito, esta matriz contendrá las entradas del directorio.

- **filter:** Es un puntero a una función que se utilizará para filtrar las entradas del directorio. Puede ser **NULL** si no se desea aplicar ningún filtro.
- **compar:** Es un puntero a una función que se utiliza para comparar las entradas del directorio para ordenarlas en la lista resultante. Puede ser **NULL** si no se desea ordenar la lista.

La función devuelve el número de entradas escaneadas si tiene éxito, o -1 si ocurre un error. La lista de entradas escaneadas se almacena en **namelist**.

Stat:

Se utiliza para obtener información sobre un archivo especificado por su nombre de ruta. Esta función obtiene y almacena información sobre el archivo en una estructura **struct stat**.

```
1 #include <sys/types.h>
2 #include <sys/stat.h>
3 #include <unistd.h>
4
5 int stat(const char *pathname, struct stat *statbuf);
```


- **pathname:** Es una cadena de caracteres que especifica la ruta del archivo del cual se desea obtener información.
- **statbuf:** Es un puntero a una estructura **struct stat** donde se almacenará la información sobre el archivo.

La estructura **struct stat** contiene varios campos que proporcionan información sobre el archivo, como el tipo de archivo, los permisos, el tamaño, el tiempo de modificación, etc. Algunos de los campos comunes en la estructura **struct stat** incluyen:

- **st_mode:** Modo de archivo (permisos y tipo).
- **st_size:** Tamaño del archivo en bytes.
- **st_mtime:** Hora de la última modificación del archivo.
- **st_uid:** Identificador de usuario del propietario del archivo.
- **st_gid:** Identificador de grupo del propietario del archivo.

La función **stat** devuelve **0** si tiene éxito y **-1** si ocurre un error. En caso de error, se puede consultar la variable global **errno** para determinar la causa específica del error.

2.- Capture, compile y ejecute el programa de creación de un nuevo hilo en Linux. Observe su funcionamiento.



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

```
vboxuser@UBUNTU:~/Escritorio/PRACTICA6$ ./hilos_1
Hola mundo desde un hilo en Linux
```

3.- Capture, compile y ejecute el siguiente programa de creación de hilos en Linux. Observe su funcionamiento.

```
1  #include <stdio.h>
2  #include <pthread.h>
3
4  void *hilo(void *arg);
5
6  int main(void)
7  {
8      pthread_t id_hilo;
9      char *mensaje = "Hola a todos desde el hilo";
10     int devolucion_hilo;
11
12     pthread_create(&id_hilo, NULL, hilo, (void *)mensaje);
13     pthread_join(id_hilo, (void *)&devolucion_hilo);
14
15     printf("valor = %i\n", devolucion_hilo);
16
17     return 0;
18 }
19
20 void *hilo(void *arg)
21 {
22     char *men;
23     int resultado_hilo = 0;
24
25     men = (char *)arg;
26     printf("%s\n", men);
27
28     resultado_hilo = 100;
29     pthread_exit((void *)resultado_hilo);
30 }
```

```
vboxuser@UBUNTU:~/Escritorio/PRACTICA6$ ./hilos_2
Hola a todos desde el hilo
valor = 100
```

4.- A través del sitio MSDN, investigue el funcionamiento de la llamada al sistema `CreateThread()`.

Función `CreateThread`

Crea un hilo para ejecutar dentro del espacio de direcciones virtuales del proceso de llamada. Para crear un hilo que se ejecute en el espacio de direcciones virtuales de otro proceso, utilice la función **`CreateRemoteThread`**.

Sintaxis:

```
HANDLE CreateThread(
    [in, optional] LPSECURITY_ATTRIBUTES  lpThreadAttributes,
    [in]           SIZE_T                 dwStackSize,
    [in]           LPTHREAD_START_ROUTINE lpStartAddress,
    [in, optional] __drv_aliasesMem LPVOID lpParameter,
    [in]           DWORD                  dwCreationFlags,
    [out, optional] LPDWORD                lpThreadId
);
```

Parámetros:

- **[in, optional] lpThreadAttributes:** Un puntero a una estructura **SECURITY_ATTRIBUTES** que determina si los procesos secundarios pueden heredar el identificador devuelto. Si **lpThreadAttributes** es **NULL**, el identificador no se puede heredar.
- **[in] dwStackSize:** El tamaño inicial de la pila, en bytes. El sistema redondea este valor a la página más cercana. Si este parámetro es cero, el nuevo hilo usa el tamaño predeterminado para el ejecutable.
- **[in] lpStartAddress:** Puntero a la función definida por la aplicación que ejecutará el subproceso. Este puntero representa la dirección inicial del hilo.
- **[in, optional] lpParameter:** Un puntero a una variable que se pasará al hilo.
- **[in] dwCreationFlags:** Las banderas que controlan la creación del hilo.
- **[out, optional] lpThreadId:** Puntero a una variable que recibe el identificador del hilo. Si este parámetro es **NULL**, no se devuelve el identificador del hilo.

Valor de retorno:

- Si la función tiene éxito, el valor de retorno es un identificador del nuevo hilo.
- Si la función falla, el valor de retorno es **NULL**. Para obtener información de error ampliada, llame a **GetLastError**.

5.- Capture, compile y ejecute el siguiente programa de creación de hilos en Windows. Observe su funcionamiento.

```
1 #include <windows.h>
2 #include <stdio.h>
3
4 DWORD WINAPI funcionHilo(LPVOID lpParam);
5
6 typedef struct Informacion info;
7
8 struct Informacion
9 {
10     int val_1;
11     int val_2;
12 };
13
14 int main(void)
15 {
16     DWORD idHilo;
17     HANDLE manHilo;
18     info argumentos;
19
20     argumentos.val_1 = 10;
21     argumentos.val_2 = 100;
22
23     /* Identificador del Hilo */
24     /* Manejador del Hilo */
25
26     // Creacion del hilo
27     manHilo = CreateThread(NULL, 0, funcionHilo, &argumentos, 0, &idHilo);
28
29     // Espera la finalización del hilo
30     WaitForSingleObject(manHilo, INFINITE);
31
32     printf("Valores al salir del Hilo: %i %i\n", argumentos.val_1, argumentos.val_2);
33
34     // Cierre del manejador del hilo creado
35     CloseHandle(manHilo);
36
37     return 0;
38 }
39
40 DWORD WINAPI funcionHilo(LPVOID lpParam)
41 {
42     info *datos = (info *)lpParam;
43
44     printf("Valores al entrar al Hilo: %i %i\n", datos->val_1, datos->val_2);
45
46     datos->val_1 *= 2;
47     datos->val_2 *= 2;
48
49     return 0;
50 }
```

```
C:\Users\jarro\OneDrive\Escritorio\SISTEMAS OPERATIVOS\PRACTICA 6>hilos_1
Valores al entrar al Hilo: 10 100
Valores al salir del Hilo: 20 200
```

6. 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 20 hilos. A su vez cada uno de los 20 hilos creará 15 hilos más. A su vez cada uno de los 15 hilos creará 10 hilos más. Cada uno de los hilos creados imprimirá en pantalla “Práctica 6 Hilo Terminal” si se trata de un hilo terminal o los identificadores de los hilos creados si se trata de un proceso o hilo padre.

```
1  #include<stdio.h>
2  #include <unistd.h>
3  #include<stdlib.h>
4  #include<sys/types.h>
5  #include<sys/wait.h>
6  #include <string.h>
7  #include <pthread.h>
8
9  pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
10
11 void *hilo3(void *arg)
12 {
13     pthread_mutex_lock(&mutex);
14     char* men=(char*)arg;
15     printf("\t\tPractica 6 hilo terminal\n");
16     pthread_mutex_unlock(&mutex);
17     pthread_exit(NULL);
18 }
19
20 void *hilo2(void *arg)
21 {
22     int id= pthread_self();
23
24     printf("\t2- %d\n",id);
25
26     pthread_t id_hilo[10];
27
28     for(int i=0;i<10;i++)
29     {
30         pthread_create(&id_hilo[i],NULL,(void*)hilo3,NULL);
31     }
32
33     for(int i=0;i<10;i++)
34     {
35         pthread_join(id_hilo[i],NULL);
36     }
37
38     pthread_exit(NULL);
39 }
40
41 void *hilo(void *arg)
42 {
43     int id= pthread_self();
44
45     printf("1- %d\n",id);
46
47     pthread_t id_hilo[15];
48
49     for(int i=0;i<15;i++)
50     {
51         pthread_create(&id_hilo[i],NULL,(void*)hilo2,NULL);
52     }
53
54     for(int i=0;i<15;i++)
55     {
56         pthread_join(id_hilo[i],NULL);
57     }
58
59     pthread_exit(NULL);
60 }
61
62 int main(void)
63 {
64     int id_proc;
65
66     id_proc=fork();
67
68     if(id_proc==0)
69     {
70         pthread_t id_hilo[20];
71
72         for(int i=0;i<20;i++)
73         {
74             pthread_create(&id_hilo[i],NULL,(void*)hilo,NULL);
75         }
76
77         for(int i=0;i<20;i++)
78         {
79             pthread_join(id_hilo[i],NULL);
80         }
81     }
82     else wait(0);
83
84     return 0;
85 }
```


Salida recortada (extensión completa muy grande):

[illegible]



```
1 #include <windows.h>
2 #include <stdio.h>
3
4 int main(int argc, char *argv[])
5 {
6     STARTUPINFO si; /* Estructura de información inicial para Windows */
7     PROCESS_INFORMATION pi; /* Estructura de información del adm. de procesos */
8     int i;
9
10    ZeroMemory(&si, sizeof(si));
11    si.cb = sizeof(si);
12    ZeroMemory(&pi, sizeof(pi));
13
14    if(argc != 2)
15    {
16        printf("Usar: %s Nombre_programa_hijo\n", argv[0]);
17        return 1;
18    }
19
20    // Creación proceso hijo
21    if(!CreateProcess(NULL, argv[1], NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))
22    {
23        printf("Fallo al invocar CreateProcess (%d)\n", GetLastError());
24        return 1;
25    }
26
27    // Proceso padre
28    WaitForSingleObject(pi.hProcess, INFINITE);
29
30    // Terminación controlada del proceso e hilo asociado de ejecución
31    CloseHandle(pi.hProcess);
32    CloseHandle(pi.hThread);
33
34    return 0;
35 }
```



```
1 #include <windows.h>
2 #include <stdio.h>
3
4 DWORD idHilo;
5 HANDLE manHilo;
6
7 DWORD WINAPI hilo3(LPVOID lpParam)
8 {
9     printf("\t\t\tPractica 6 hilo terminal\n");
10    return 0;
11 }
12
13 DWORD WINAPI hilo2(LPVOID lpParam)
14 {
15     printf("\t2- %d\n", GetCurrentThreadId());
16
17     for(int i=0; i<10; i++)
18     {
19         manHilo=CreateThread(NULL, 0, hilo3, NULL, 0, &idHilo);
20     }
21
22     return 0;
23 }
24
25 DWORD WINAPI hilo(LPVOID lpParam)
26 {
27
28     printf("1- %d\n", GetCurrentThreadId());
29
30     for(int i=0; i<15; i++)
31     {
32         manHilo=CreateThread(NULL, 0, hilo2, NULL, 0, &idHilo);
33     }
34
35     return 0;
36 }
37
38
39
40 int main(void)
41 {
42     for(int i=0; i<20; i++)
43     {
44         manHilo=CreateThread(NULL, 0, hilo, NULL, 0, &idHilo);
45     }
46     WaitForSingleObject(manHilo, INFINITE);
47     CloseHandle(manHilo);
48     return 0;
49 }
```


7.- Programe la misma aplicación del punto 5 de la práctica 5 en su sección de Linux pero utilizando hilos (tanto en Linux como en Windows) en vez de procesos, obtenga el tiempo de ejecución del programa con hilos. Compare ambos programas en su tiempo de ejecución (el creado en la práctica 5 y el creado en esta práctica), y dé sus observaciones tanto de funcionamiento como de los tiempos de ejecución resultantes.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <sys/types.h>
5 #include <sys/wait.h>
6 #include <string.h>
7 #include <pthread.h>
8 #include <time.h>
9
10 #define N 10
11
12 // Declaracion de las matrices
13 int MatrizUno[N][N];
14 int MatrizDos[N][N];
15
16 // Declaracion de la matriz que almacenar algunos de los resultados
17 int Resultado[N][N];
18 int TraspuestaUno[N][N];
19 int TraspuestaDos[N][N];
20 double InversaUno[N][N];
21 double InversaDos[N][N];
22
23 // Funcion para llenar una matriz con valores aleatorios
24 void LlenarMatriz(int matriz[N][N])
25 {
26     int i, j;
27     for (i = 0; i < N; i++)
28     {
29         for (j = 0; j < N; j++)
30         {
31             matriz[i][j] = rand() % 101; // Genera números aleatorios entre 0 y 100
32         }
33     }
34 }
35
36 // Funcion para imprimir matriz entera
37 void ImprimirMatriz(int matriz[N][N])
38 {
39     for (int i = 0; i < N; i++)
40     {
41         for (int j = 0; j < N; j++)
42             printf("%d\t", matriz[i][j]);
43         printf("\n");
44     }
45 }
46
47 // Funcion que escribe los resultados de las operaciones en archivos de texto
48 void Archivo(void *matriz, char *nombre_archivo, char *operacion, char tipo)
49 {
50     FILE *archivo = fopen(nombre_archivo, "w");
51     if (archivo == NULL)
52     {
53         printf("Error al abrir el archivo %s\n", nombre_archivo);
54         exit(1);
55     }
56
57     fprintf(archivo, "\n--- Operacion---: %s\n", operacion);
58
59     if (tipo == 'i') {
60         int (*matriz_int)[N] = (int (*)[N]) matriz;
61         for (int i = 0; i < N; i++)
62         {
63             for (int j = 0; j < N; j++)
64             {
65                 fprintf(archivo, "%d ", matriz_int[i][j]);
66             }
67             fprintf(archivo, "\n");
68         }
69     } else if (tipo == 'd') {
70         double (*matriz_double)[N] = (double (*)[N]) matriz;
71         for (int i = 0; i < N; i++)
72         {
73             for (int j = 0; j < N; j++)
74             {
75                 fprintf(archivo, "%lf ", matriz_double[i][j]);
76             }
77             fprintf(archivo, "\n");
78         }
79     }
80
81     fclose(archivo);
82 }
```

```

85 // Hilos para cada unas de las operaciones
86
87 // Hilo para la suma
88 void *hilosuma(void *arg)
89 {
90     printf("\nHilo de la suma\n");
91
92     for (int i = 0; i < N; i++)
93     {
94         for (int j = 0; j < N; j++)
95         {
96             Resultado[i][j] = Matrizuno[i][j] + Matrizdos[i][j];
97         }
98     }
99
100     Archivo(Resultado, "Suma.txt", "Suma de matrices", 'i');
101     return NULL;
102 }
103
104 // Hilo para la resta
105 void *hiloresta(void *arg)
106 {
107     printf("\nHilo de la resta\n");
108
109     for (int i = 0; i < N; i++)
110     {
111         for (int j = 0; j < N; j++)
112         {
113             Resultado[i][j] = Matrizuno[i][j] - Matrizdos[i][j];
114         }
115     }
116
117     Archivo(Resultado, "Resta.txt", "Resta de matrices", 'i');
118
119     return NULL;
120 }
121
122 // Hilo para la multiplicacion
123 void *hilomultiplicacion(void *arg)
124 {
125     printf("\nHilo de la multiplicacion\n");
126     for (int i = 0; i < N; i++)
127     {
128         for (int j = 0; j < N; j++)
129         {
130             Resultado[i][j] = 0;
131             for (int k = 0; k < N; k++)
132             {
133                 Resultado[i][j] += Matrizuno[i][k] * Matrizdos[k][j];
134             }
135         }
136     }
137
138     Archivo(Resultado, "Multiplicacion.txt", "Multiplicacion de matrices", 'i');
139
140     return NULL;
141 }
142
143 // Hilo para la traspuesta
144 void *hilotraspuesta(void *arg)
145 {
146     printf("\nHilo de la traspuesta\n");
147
148     for (int i = 0; i < N; i++)
149     {
150         for (int j = 0; j < N; j++)
151         {
152             TraspuestaUno[j][i] = Matrizuno[i][j];
153             TraspuestaDos[j][i] = Matrizdos[i][j];
154         }
155     }
156
157     Archivo(TraspuestaUno, "Traspuesta1.txt", "Traspuesta de la Matriz 1", 'i');
158     Archivo(TraspuestaDos, "Traspuesta2.txt", "Traspuesta de la Matriz 2", 'i');
159
160     return NULL;
161 }

```

```

163 // Proceso para calcular la inversa de la matriz
164
165 void Cofactor(int matriz[N][N], int temp[N][N], int p, int q, int n) {
166     int i = 0, j = 0;
167     for (int fila = 0; fila < n; fila++)
168     {
169         for (int col = 0; col < n; col++)
170         {
171             if (fila != p && col != q)
172             {
173                 temp[i][j++] = matriz[fila][col];
174                 if (j == n - 1)
175                 {
176                     j = 0;
177                     i++;
178                 }
179             }
180         }
181     }
182 }
183
184 int DeterminanteMatriz(int matriz[N][N], int n)
185 {
186     int determinante = 0;
187
188     if (n == 1)
189         return matriz[0][0];
190
191     int temp[N][N]; // Para almacenar cofactores
192     int signo = 1; // Para almacenar el signo del cofactor
193     for (int f = 0; f < n; f++)
194     {
195         Cofactor(matriz, temp, 0, f, n); // Calculando el cofactor
196         determinante += signo * matriz[0][f] * DeterminanteMatriz(temp, n - 1);
197         signo = -signo;
198     }
199     return determinante;
200 }
201
202
203 void AdjuntaMatriz(int matriz[N][N], double adjunta[N][N], int n) {
204     if (n == 1)
205     {
206         adjunta[0][0] = 1;
207         return;
208     }
209
210     int temp[N][N]; // Para almacenar cofactores
211     int signo = 1; // Para almacenar el signo del cofactor
212     for (int i = 0; i < n; i++)
213     {
214         for (int j = 0; j < n; j++)
215         {
216             Cofactor(matriz, temp, i, j, n);
217             signo = ((i + j) % 2 == 0) ? 1 : -1;
218             adjunta[j][i] = (signo) * (DeterminanteMatriz(temp, n - 1));
219         }
220     }
221 }
222
223 // Hilo para la inversa
224 void *hiloInversa(void *arg)
225 {
226     printf("\nHilo de la inversa\n");
227     double determinanteuno = (double)DeterminanteMatriz(Matrizuno, N);
228     double determinantedos = (double)DeterminanteMatriz(Matrizdos, N);
229
230     if (determinanteuno == 0)
231     {
232         printf("La matriz es singular, no se puede calcular la inversa.\n");
233         return NULL;
234     }
235
236     double adjuntauno[N][N];
237     double adjuntados[N][N];
238
239     AdjuntaMatriz(Matrizuno, adjuntauno, N);
240     AdjuntaMatriz(Matrizdos, adjuntados, N);
241
242     for (int i = 0; i < N; i++)
243     {
244         for (int j = 0; j < N; j++)
245         {
246             InversaUno[i][j] = adjuntauno[i][j] / determinanteuno;
247             InversaDos[i][j] = adjuntados[i][j] / determinantedos;
248         }
249     }
250
251     Archivo(InversaUno, "Inversa1.txt", "Inversa de la Matriz 1", 'd');
252     Archivo(InversaDos, "Inversa2.txt", "Inversa de la Matriz 2", 'd');
253
254     return NULL;
255 }

```



```

257 // Hilo para mostrar los resultados
258 void *hiloarchivos(void *arg)
259 {
260     execlp("cat", "cat", "Suma.txt", "Resta.txt", "Multiplicacion.txt", "Transpuesta1.txt", "Transpuesta2.txt", "Inversa1.txt", "Inversa2.txt", NULL);
261 }
262 return NULL;
263 }
264
265 int main(void)
266 {
267     srand(time(NULL));
268
269     // Se llenan las matrices con valores aleatorios
270     LlenarMatriz(Matrizuno);
271     LlenarMatriz(Matrizdos);
272
273     // Muestra en pantalla las dos matrices generadas
274     printf("Matriz 1:\n");
275     ImprimirMatriz(Matrizuno);
276     printf("\nMatriz 2:\n");
277     ImprimirMatriz(Matrizdos);
278
279     // Llamada a 6 hilos diferentes
280     pthread_t id_hilo;
281
282     pthread_create(&id_hilo, NULL, (void*)hilosuma, NULL);
283     pthread_create(&id_hilo, NULL, (void*)hiloresta, NULL);
284     pthread_create(&id_hilo, NULL, (void*)hilomultiplicacion, NULL);
285     pthread_create(&id_hilo, NULL, (void*)hilotraspuesta, NULL);
286     pthread_create(&id_hilo, NULL, (void*)hiloinvertida, NULL);
287     pthread_create(&id_hilo, NULL, (void*)hiloarchivos, NULL);
288     pthread_join(id_hilo, NULL);
289
290     return 0;
291 }

```

```

vboxuser@UBUNTU:~/Escritorio/PRACTICA6$ ./hilos_4
Matriz 1:
44 6 1 83 45 92 58 47 63 62
2 16 13 9 37 74 83 49 29 74
34 86 86 96 62 61 19 33 47 82
38 57 89 5 39 100 63 63 46 26
24 48 42 3 23 45 44 73 60 73
46 94 24 98 90 86 59 75 18 5
57 22 62 11 28 68 10 91 30 22
83 21 36 91 24 59 36 68 31 96
6 43 56 30 41 11 15 100 52 0
71 8 89 99 86 83 66 96 74 63

Matriz 2:
84 56 84 19 47 74 45 49 8 42
10 14 86 32 11 93 9 93 58 62
59 95 36 48 93 22 30 59 84 70
88 68 93 71 53 5 44 98 54 18
40 30 33 92 29 10 84 38 2 7
66 62 1 69 76 94 57 5 18 6
42 5 74 34 42 94 39 53 91 59
71 30 55 70 21 84 80 71 89 83
44 20 10 45 89 86 5 11 91 23
18 32 96 58 32 37 51 71 56 42

Hilo de la suma
Hilo de la resta
Hilo de la multiplicacion
Hilo de la traspuesta
Hilo de la inversa

--- Operacion---: Suma de matrices
28652 20285 27003 29150 25937 30461 24239 24583 25678 16027
17824 12714 20375 22172 18117 26443 19422 18462 22625 15590
30429 28097 35946 34082 30362 30874 25591 36496 32425 23693
27224 23048 24233 27791 28581 36018 23142 25050 30216 23419
20113 15636 23618 22982 20865 29768 18980 22869 27571 20510
32805 23933 34133 34562 25583 35727 29056 34871 28818 22680
23839 19278 19084 22148 20955 26333 20277 19862 21785 18657
31600 24840 35318 29664 26254 30442 26282 32162 28455 21981
19262 14325 17502 20689 16936 21551 16573 20617 24143 18112
42903 33707 38936 42493 39141 40076 36085 38460 40332 28800

--- Operacion---: Resta de matrices
28652 20285 27003 29150 25937 30461 24239 24583 25678 16027
17824 12714 20375 22172 18117 26443 19422 18462 22625 15590
30429 28097 35946 34082 30362 30874 25591 36496 32425 23693
27224 23048 24233 27791 28581 36018 23142 25050 30216 23419
20113 15636 23618 22982 20865 29768 18980 22869 27571 20510
32805 23933 34133 34562 25583 35727 29056 34871 28818 22680
23839 19278 19084 22148 20955 26333 20277 19862 21785 18657
31600 24840 35318 29664 26254 30442 26282 32162 28455 21981
19262 14325 17502 20689 16936 21551 16573 20617 24143 18112
42903 33707 38936 42493 39141 40076 36085 38460 40332 28800

--- Operacion---: Multiplicacion de matrices
28652 20285 27003 29150 25937 30461 24239 24583 25678 16027
17824 12714 20375 22172 18117 26443 19422 18462 22625 15590
30429 28097 35946 34082 30362 30874 25591 36496 32425 23693
27224 23048 24233 27791 28581 36018 23142 25050 30216 23419
20113 15636 23618 22982 20865 29768 18980 22869 27571 20510
32805 23933 34133 34562 25583 35727 29056 34871 28818 22680
23839 19278 19084 22148 20955 26333 20277 19862 21785 18657
31600 24840 35318 29664 26254 30442 26282 32162 28455 21981
19262 14325 17502 20689 16936 21551 16573 20617 24143 18112
42903 33707 38936 42493 39141 40076 36085 38460 40332 28800

```

```

--- Operacion---: Traspuesta de la Matriz 1
44 2 34 38 24 46 57 83 6 71
6 16 86 57 48 94 22 21 43 8
1 13 86 89 42 24 62 36 56 89
83 9 96 5 3 98 11 91 30 99
45 37 62 39 23 90 28 24 41 86
92 74 61 100 45 86 68 59 11 83
58 83 19 63 44 59 10 36 15 66
47 49 33 63 73 75 91 68 100 96
63 29 47 46 60 18 30 31 52 74
62 74 82 26 73 5 22 96 0 63

--- Operacion---: Traspuesta de la Matriz 2
84 10 59 88 40 66 42 71 44 18
56 14 95 68 30 62 5 30 20 32
84 86 36 93 33 1 74 55 10 96
19 32 48 71 92 69 34 70 45 58
47 11 93 53 29 76 42 21 89 32
74 93 22 5 10 94 94 84 86 37
45 9 30 44 84 57 39 80 5 51
49 93 59 98 38 5 53 71 11 71
8 58 84 54 2 18 91 89 91 56
42 62 70 18 7 6 59 83 23 42

```

```

--- Operacion---: Inversa de la Matriz 1
-2.124810 0.945493 3.010529 0.135808 -3.019649 3.042341 -2.306860 1.143906 5.409870 0.797789
-0.791215 -5.747867 -1.373579 6.498105 -4.786028 -5.992182 1.524573 5.804643 -3.692272 -1.504818
2.860328 -4.909619 3.656390 -6.144741 0.615969 -4.229177 6.482089 2.416312 5.881973 -3.344393
-4.686244 3.028313 -1.984120 -1.482844 1.257027 0.341042 4.738730 -2.118427 4.385227 -4.432398
-2.136612 2.566951 3.993021 -6.319751 4.139022 5.964007 -3.655178 -3.837498 -3.002987 -0.943524
-3.082087 2.410722 -3.461462 1.657698 5.795040 4.161262 -4.400487 -4.031970 2.000961 -4.354147
-2.584262 3.447938 6.327904 4.307529 -0.848754 1.904487 -4.311665 5.823264 4.980020 -3.748410
5.861002 -5.476912 -5.614851 1.673347 5.839445 -2.968655 6.167339 -2.991640 3.667037 -5.794684
5.954418 1.812953 -0.125090 -4.956946 -5.234744 3.980472 -3.248433 -2.006373 -4.451685 3.613346
-0.086650 5.644020 1.556124 5.669020 3.144850 -5.154993 6.530013 4.246026 -3.993434 -4.400632

--- Operacion---: Inversa de la Matriz 2
3.355304 0.559782 -5.141891 -3.425347 2.345166 -4.284068 1.904470 1.387612 0.154384 0.008296
-1.811400 -2.281635 3.836291 -3.764411 -4.498473 -3.011212 3.357409 -3.739315 3.301773 -1.662575
-4.807533 4.333189 1.452576 2.639174 4.136086 -1.785526 -1.864018 2.663642 1.704040 -3.427271
4.813550 -0.642574 2.472623 -1.967385 4.343327 3.610663 -2.787678 1.801259 -0.996466 2.745318
-4.859541 -1.054208 1.510445 -3.948892 0.228717 2.267448 -2.815027 4.755168 -2.161358 -4.569406
3.988649 -2.105669 -0.606417 -0.321721 -1.153404 -0.331024 -2.742913 -3.038255 2.540632 0.905971
-2.257141 4.321440 2.423070 1.684402 -3.590686 3.140135 4.242582 -0.978147 -2.449367 -0.456771
-4.934143 -1.384832 -3.224706 2.648930 3.974033 -1.886522 0.097736 -3.536180 3.502790 -1.803252
1.872690 -3.899204 3.199508 2.203961 0.566772 -0.418678 4.552684 -0.577577 4.873046 1.653143
-1.644044 -2.335997 -1.585772 2.943073 2.087213 0.539947 -5.036978 4.819800 -3.901533 -1.677677

```

```

● ● ●

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <sys/types.h>
4 #include <time.h>
5 #include <windows.h>
6
7 DWORD WINAPI funcionHilo(LPVOID lpParam);
8
9 #define N 10
10
11 // Declaracion de las matrices
12 int Matrizuno[N][N];
13 int Matrizdos[N][N];
14
15 // Declaracion de la matriz que almacenar algunos de los resultados
16 int Resultado[N][N];
17 int TraspuestaUno[N][N];
18 int TraspuestaDos[N][N];
19 double InversaUno[N][N];
20 double InversaDos[N][N];
21
22 // Funcion para llenar una matriz con valores aleatorios
23 void LlenarMatriz(int matriz[N][N])
24 {
25     int i, j;
26     for (i = 0; i < N; i++)
27     {
28         for (j = 0; j < N; j++)
29         {
30             matriz[i][j] = rand() % 101; // Genera números aleatorios entre 0 y 100
31         }
32     }
33 }

```



```

35 // Funcion para imprimir matriz entera
36 void ImprimirMatriz(int matriz[N][N])
37 {
38     for (int i = 0; i < N; i++)
39     {
40         for (int j = 0; j < N; j++)
41             printf("%d\t", matriz[i][j]);
42         printf("\n");
43     }
44 }
45
46 // Funcion que escribe los resultados de las operaciones en archivos de texto
47 void Archivo(void *matriz, char *nombre_archivo, char *operacion, char tipo)
48 {
49     FILE *archivo = fopen(nombre_archivo, "w");
50     if (archivo == NULL)
51     {
52         printf("Error al abrir el archivo %s\n", nombre_archivo);
53         exit(1);
54     }
55
56     fprintf(archivo, "\n--- Operacion---: %s\n", operacion);
57
58     if (tipo == 'i') {
59         int (*matriz_int)[N] = (int (*)[N]) matriz;
60         for (int i = 0; i < N; i++)
61         {
62             for (int j = 0; j < N; j++)
63             {
64                 fprintf(archivo, "%d ", matriz_int[i][j]);
65             }
66             fprintf(archivo, "\n");
67         }
68     } else if (tipo == 'd') {
69         double (*matriz_double)[N] = (double (*)[N]) matriz;
70         for (int i = 0; i < N; i++)
71         {
72             for (int j = 0; j < N; j++)
73             {
74                 fprintf(archivo, "%lf ", matriz_double[i][j]);
75             }
76             fprintf(archivo, "\n");
77         }
78     }
79
80     fclose(archivo);
81 }
82
83
84
85 DWORD WINAPI hilosuma(LPVOID lpParam)
86 {
87     printf("\nHilo de la suma\n");
88
89     for (int i = 0; i < N; i++)
90     {
91         for (int j = 0; j < N; j++)
92         {
93             Resultado[i][j] = Matrizuno[i][j] + Matrizdos[i][j];
94         }
95     }
96
97     Archivo(Resultado, "Suma.txt", "Suma de matrices", 'i');
98
99     return 0;
100 }

```

```

103 DWORD WINAPI hiloresta(LPVOID lpParam)
104 {
105     printf("\nHilo de la resta\n");
106
107     for (int i = 0; i < N; i++)
108     {
109         for (int j = 0; j < N; j++)
110         {
111             Resultado[i][j] = Matrizuno[i][j] - Matrizdos[i][j];
112         }
113     }
114
115     Archivo(Resultado, "Resta.txt", "Resta de matrices", 'i');
116     return 0;
117 }
118
119
120 DWORD WINAPI hilomultiplicacion(LPVOID lpParam)
121 {
122     printf("\nHilo de la multiplicacion\n");
123     for (int i = 0; i < N; i++)
124     {
125         for (int j = 0; j < N; j++)
126         {
127             Resultado[i][j] = 0;
128             for (int k = 0; k < N; k++)
129             {
130                 Resultado[i][j] += Matrizuno[i][k] * Matrizdos[k][j];
131             }
132         }
133     }
134
135     Archivo(Resultado, "Multiplicacion.txt", "Multiplicacion de matrices", 'i');
136
137     return 0;
138 }
139
140
141 DWORD WINAPI hilotraspuesta(LPVOID lpParam)
142 {
143     printf("\nHilo de la traspuesta\n");
144
145     for (int i = 0; i < N; i++)
146     {
147         for (int j = 0; j < N; j++)
148         {
149             TraspuestaUno[j][i] = Matrizuno[i][j];
150             TraspuestaDos[j][i] = Matrizdos[i][j];
151         }
152     }
153
154     Archivo(TraspuestaUno, "Traspuesta1.txt", "Traspuesta de la Matriz 1", 'i');
155     Archivo(TraspuestaDos, "Traspuesta2.txt", "Traspuesta de la Matriz 2", 'i');
156
157     return 0;
158 }
159
160
161 // Proceso para calcular la inversa de la matriz
162
163 void Cofactor(int matriz[N][N], int temp[N][N], int p, int q, int n) {
164     int i = 0, j = 0;
165     for (int fila = 0; fila < n; fila++)
166     {
167         for (int col = 0; col < n; col++)
168         {
169             if (fila != p && col != q)
170             {
171                 temp[i][j++] = matriz[fila][col];
172                 if (j == n - 1)
173                 {
174                     j = 0;
175                     i++;
176                 }
177             }
178         }
179     }
180 }

```

```

182 int DeterminanteMatriz(int matriz[N][N], int n)
183 {
184     int determinante = 0;
185
186     if (n == 1)
187         return matriz[0][0];
188
189     int temp[N][N]; // Para almacenar cofactores
190     int signo = 1; // Para almacenar el signo del cofactor
191     for (int f = 0; f < n; f++)
192     {
193         Cofactor(matriz, temp, 0, f, n); // Calculando el cofactor
194         determinante += signo * matriz[0][f] * DeterminanteMatriz(temp, n - 1);
195         signo = -signo;
196     }
197     return determinante;
198 }
199
200
201 void AdjuntaMatriz(int matriz[N][N], double adjunta[N][N], int n) {
202     if (n == 1)
203     {
204         adjunta[0][0] = 1;
205         return;
206     }
207
208     int temp[N][N]; // Para almacenar cofactores
209     int signo = 1; // Para almacenar el signo del cofactor
210     for (int i = 0; i < n; i++)
211     {
212         for (int j = 0; j < n; j++)
213         {
214             Cofactor(matriz, temp, i, j, n);
215             signo = ((i + j) % 2 == 0) ? 1 : -1;
216             adjunta[j][i] = (signo) * (DeterminanteMatriz(temp, n - 1));
217         }
218     }
219 }
220
221 DWORD WINAPI hiloInversa(LPVOID lpParam)
222 {
223     printf("\nHilo de la inversa\n");
224     double determinanteuno = (double)DeterminanteMatriz(Matrizuno, N);
225     double determinantedos = (double)DeterminanteMatriz(Matrizdos, N);
226
227     if (determinanteuno == 0)
228     {
229         printf("La matriz es singular, no se puede calcular la inversa.\n");
230         return 0;
231     }
232
233     double adjuntauno[N][N];
234     double adjuntados[N][N];
235
236     AdjuntaMatriz(Matrizuno, adjuntauno, N);
237     AdjuntaMatriz(Matrizdos, adjuntados, N);
238
239     for (int i = 0; i < N; i++)
240     {
241         for (int j = 0; j < N; j++)
242         {
243             InversaUno[i][j] = adjuntauno[i][j] / determinanteuno;
244             InversaDos[i][j] = adjuntados[i][j] / determinantedos;
245         }
246     }
247
248     Archivo(InversaUno, "Inversa1.txt", "Inversa de la Matriz 1", 'd');
249     Archivo(InversaDos, "Inversa2.txt", "Inversa de la Matriz 2", 'd');
250
251     return 0;
252 }

```

```

255 void mostrar(char *archivo) {
256     FILE *file = fopen(archivo, "r");
257     if (file == NULL) {
258         printf("Error al abrir el archivo %s\n", archivo);
259         return;
260     }
261
262     char line[256];
263     while (fgets(line, sizeof(line), file)) {
264         printf("%s", line);
265     }
266
267     fclose(file);
268 }
269
270 DWORD WINAPI hiloarchivos(LPVOID lpParam)
271 {
272     mostrar("Suma.txt");
273     mostrar("Resta.txt");
274     mostrar("Multiplicacion.txt");
275     mostrar("Traspuesta1.txt");
276     mostrar("Traspuesta2.txt");
277     mostrar("Inversa1.txt");
278     mostrar("Inversa2.txt");
279
280     return 0;
281 }
282
283 int main(void)
284 {
285     srand(time(NULL));
286
287     // Se llenan las matrices con valores aleatorios
288     LlenarMatriz(Matrizuno);
289     LlenarMatriz(Matrizdos);
290
291     // Muestra en pantalla las dos matrices generadas
292     printf("Matriz 1:\n");
293     ImprimirMatriz(Matrizuno);
294     printf("\nMatriz 2:\n");
295     ImprimirMatriz(Matrizdos);
296
297     // Llamada a 6 hilos diferentes
298     DWORD idHilo;
299     HANDLE manHilo;
300
301     manHilo=CreateThread(NULL,0,hilosuma,NULL,0,&idHilo);
302     manHilo=CreateThread(NULL,0,hiloresta,NULL,0,&idHilo);
303     manHilo=CreateThread(NULL,0,hilomultiplicacion,NULL,0,&idHilo);
304     manHilo=CreateThread(NULL,0,hilotraspuesta,NULL,0,&idHilo);
305     manHilo=CreateThread(NULL,0,hiloinvertida,NULL,0,&idHilo);
306     manHilo=CreateThread(NULL,0,hiloarchivos,NULL,0,&idHilo);
307
308     WaitForSingleObject(manHilo,INFINITE);
309     CloseHandle(manHilo);
310
311     return 0;
312 }

```

```
C:\Users\jarro\OneDrive\Escritorio\SISTEMAS OPERATIVOS\PRACTICA 6>hilos_2
Matriz 1:
6      56      31      16      20      23      54      38      5      41
15     79      18      54      9       99     90     100     76     33
40     17      36      59      10      88     78     53     43     15
45     48      4       23      86     36     65     55     5      29
83     5       59      17      76     30     78     63     83     62
30     91      26      3       15     31     81     35     82     98
71     71      0       58     11     11     79     24     70     80
54     9       22      33     20     67     3      50     94     26
1      7       47      30     36     79     98     42     15     16
28     57      92      3       4      72     33     38     91     65

Matriz 2:
3      27      15      50      15      10      96      89      41      6
46     43      7       16     22     58     73      4      57     79
86     2       69      20      1      34     25     55     88     78
78     13      21      56     48     47      8      83      2     57
88     58      9       18      0      59     29     23     16      1
64     9       84     55     59     93     12     62     72     50
43     64     58      27     54     15     52     97     98     56
76     49     11      48     70     61      1      22     61      1
32     25      74     96     97     54     99     92     83     27
21      6      90     23     69     77     31     67     36     79

Hilo de la suma

Hilo de la resta

Hilo de la traspuesta

Hilo de la inversa

Hilo de la multiplicacion
```

```
--- Operacion---: Suma de matrices
28498 19320 27747 27624 23531 17783 34011 15948 28873 33473
18865 10258 18554 15317 24434 17456 29668 15598 18975 14910
23318 20119 26818 22828 29603 21083 39212 23769 26626 27726
24011 22767 23542 25291 24834 17293 32939 18187 30675 32994
31323 19316 32043 32125 27186 25168 37808 17137 33419 35786
31360 26906 24759 31181 32230 24103 40365 22208 34553 35437
22869 20647 20406 19706 24025 20897 35072 20299 20800 25828
34993 25961 29013 32033 32054 19791 39711 23002 34046 35986
26023 21795 21972 23590 25075 24102 32816 19172 22401 25123
31640 19977 29106 32328 31835 23529 40354 19829 35112 35240
```

```
--- Operacion---: Resta de matrices
15971 9896 12679 9042 12368 14988 11035 14958 17675 15396
31162 18711 26465 26290 31923 31715 22865 33601 36498 25571
24185 12776 21563 20439 22162 22037 16630 29884 27377 18931
22097 16052 12923 13456 14809 19706 15863 20454 20131 13692
27587 17999 25851 23904 24951 25468 26766 36413 32898 19610
20875 15500 25654 19612 25991 25695 26407 29717 31590 25660
18816 14923 21029 20582 24316 21815 26456 32343 25938 22560
18573 9229 18912 21280 21141 20945 18357 25186 21955 12812
23153 12412 19593 14245 17108 18969 11133 23800 24887 17120
24396 10949 28230 21298 23912 26519 22931 29349 33192 25100
```

```
--- Operacion---: Multiplicacion de matrices
15971 9896 12679 9042 12368 14988 11035 14958 17675 15396
31162 18711 26465 26290 31923 31715 22865 33601 36498 25571
24185 12776 21563 20439 22162 22037 16630 29884 27377 18931
22097 16052 12923 13456 14809 19706 15863 20454 20131 13692
27587 17999 25851 23904 24951 25468 26766 36413 32898 19610
20875 15500 25654 19612 25991 25695 26407 29717 31590 25660
18816 14923 21029 20582 24316 21815 26456 32343 25938 22560
18573 9229 18912 21280 21141 20945 18357 25186 21955 12812
23153 12412 19593 14245 17108 18969 11133 23800 24887 17120
24396 10949 28230 21298 23912 26519 22931 29349 33192 25100
```

```
--- Operacion---: Traspuesta de la Matriz 1
6 15 40 45 83 30 71 54 1 28
56 79 17 48 5 91 71 9 7 57
31 18 36 4 59 26 0 22 47 92
16 54 59 23 17 3 58 33 30 3
20 9 10 86 76 15 11 20 36 4
23 99 88 36 30 31 11 67 79 72
54 90 78 65 78 81 79 3 98 33
38 100 53 55 63 35 24 50 42 38
5 76 43 5 83 82 70 94 15 91
41 33 15 29 62 98 80 26 16 65
```

```
--- Operacion---: Traspuesta de la Matriz 2
3 46 86 78 88 64 43 76 32 21
27 43 2 13 58 9 64 49 25 6
15 7 69 21 9 84 58 11 74 90
50 16 20 56 18 55 27 48 96 23
15 22 1 48 0 59 54 70 97 69
10 58 34 47 59 93 15 61 54 77
96 73 25 8 29 12 52 1 99 31
89 4 55 83 23 62 97 22 92 67
41 57 88 2 16 72 98 61 83 36
6 79 78 57 1 50 56 1 27 79
```

```

--- Operacion---: Inversa de la Matriz 1
-1.295155 -0.961375 1.462953 -1.065902 1.567570 0.718176 0.042299 2.147281 -2.160547 -1.252494
-1.730197 0.407017 0.695888 -0.991255 -1.464687 -1.943491 2.381167 -0.544487 0.949128 -1.088099
1.431321 -0.258046 1.871612 0.902578 1.708291 -0.968419 1.895639 -0.836886 2.015387 1.849146
-1.303427 -0.282778 -1.665984 -1.317300 -2.196911 -0.360126 1.487425 -2.092960 0.907943 1.610182
2.061673 -1.787989 -1.434558 -0.690589 0.485567 -0.370706 -2.183175 1.202796 1.372408 -1.407268
1.441314 0.778417 1.266094 0.543084 0.890488 -2.176404 1.709165 -1.983588 2.309060 -0.835366
0.003746 0.410786 -1.056188 -1.284715 0.605263 -0.777228 1.344276 -0.239368 2.383067 0.661903
1.429751 1.660223 0.308288 1.299846 -1.854669 -0.125874 -1.513446 2.248771 -0.356061 0.170793
-1.230466 1.381349 -1.385880 1.773891 -2.004046 -0.865336 0.980779 1.029420 1.522043 1.782726
1.996215 0.093204 1.541686 -0.016848 1.888811 -0.293091 -0.970788 1.621358 1.110203 -1.596618

--- Operacion---: Inversa de la Matriz 2
0.753911 0.169394 -0.800693 0.254218 -0.910164 -0.389631 -1.041838 1.042153 -0.085225 0.318865
-0.679093 -0.924489 0.686676 -0.845302 0.889626 1.066566 -0.396925 0.597106 -0.339786 -1.012088
0.848322 -0.743694 0.156327 0.536657 0.192568 -0.552193 -0.872965 -0.555933 -0.592160 -0.054658
0.657406 -0.873338 -0.681033 0.577126 0.863876 -1.060126 -0.685458 -0.668395 -1.066464 0.975933
-0.243900 0.823212 0.346507 -0.286178 -0.678375 0.388957 0.928426 -0.442537 -0.727309 0.663223
0.695886 -0.277183 0.121907 -0.107905 -0.068716 0.036203 -1.072418 1.066396 -0.472585 0.196841
0.183932 0.960220 -0.919977 1.165551 1.005875 0.615023 0.610780 -0.162172 0.300494 -0.297733
0.797698 0.912706 1.048292 0.522224 0.857580 0.945117 -0.477626 0.217073 -0.785205 -0.542093
0.010278 -0.384195 0.565829 -0.730748 0.205822 -0.755035 1.024210 -0.175238 -1.121725 0.639760
1.086748 -0.556756 -0.034140 -0.391033 -0.830637 0.264488 -0.317866 -0.004809 0.135862 -0.500779

```

Comparación y observaciones:

Practica 5

Practica 6

Windows

```

Seconds      : 2
Milliseconds : 374
Ticks        : 23740989

```

```

Seconds      : 0
Milliseconds : 38
Ticks        : 382832

```

Linux

```

real    0m8.643s
user    0m2.249s
sys     0m0.010s

```

```

real    0m0.007s
user    0m0.004s
sys     0m0.004s

```

Los programas realizados con hilos en lugar de procesos, tanto en Linux como en Windows, fueron significativamente más rápidos. Su tiempo de ejecución fue mucho menor debido a que los hilos comparten el mismo espacio de direcciones dentro de un proceso, lo que reduce la sobrecarga de creación y destrucción en comparación con los procesos.

Además, la comunicación y sincronización entre hilos es más eficiente, ya que no requieren mecanismos de comunicación interprocesos más lentos. Los hilos también consumen menos recursos del sistema, permitiendo gestionar un mayor número de ellos de manera más eficiente.

En nuestras pruebas, los programas multihilo completaron sus tareas mucho más rápido que los programas multiproceso, especialmente en escenarios con muchas tareas concurrentes. La eficiencia en la creación, terminación y comunicación de hilos contribuyó significativamente a este mejor desempeño.

CONCLUSION

Durante esta práctica, hemos tenido la oportunidad de experimentar con la programación concurrente y el manejo de archivos en sistemas operativos, tanto en Linux como en Windows. Trabajar con hilos ha sido especialmente interesante, ya que pudimos ver de primera mano cómo los hilos pueden hacer que nuestras aplicaciones sean mucho más rápidas y eficientes al permitir la ejecución de múltiples tareas al mismo tiempo.

Utilizar las funciones de la biblioteca pthread en Linux, como pthread_create y pthread_join, nos ha dado las herramientas necesarias para gestionar hilos de manera efectiva. También exploramos CreateThread en Windows, que, aunque es un poco diferente, cumple la misma función básica de permitirnos trabajar con hilos en nuestras aplicaciones.

Además, trabajar con funciones como scandir y stat nos ha enseñado cómo interactuar con el sistema de archivos de manera más profunda. Estas funciones nos permiten listar y obtener información detallada sobre los archivos y directorios, algo que es fundamental para cualquier aplicación que necesite gestionar grandes cantidades de datos.

Una de las partes más interesantes de la práctica fue comparar el rendimiento entre programas que usan procesos y aquellos que usan hilos. Descubrimos que los hilos son generalmente más rápidos y consumen menos recursos que los procesos, lo cual es un conocimiento valioso para optimizar nuestras aplicaciones en el futuro.